Резюме урока:

Клиентские библиотеки:

- Основные способы взаимодействия с кластером.
- Официальным языком и клиентом Kafka является Java. Для других языков также создано множество библиотек, однако в этом курсе мы будем работать преимущественно с Java, как наиболее полным клиентом с точки зрения функционала.
- Все брокеры кластера могут быть помещены за один IP/Load Balancer для Service Discovery. При этом нужно обеспечить прямой доступ клиентов к каждому из них.
- Подробное описание протокола Apache Kafka доступно в официальной документации: https://kafka.apache.org/protocol

Producer:

- Посылает сообщения брокеру батчами, улучшая пропускную способность и степень сжатия данных.
- *linger.ms* — аналог Nagle's Algorithm из TCP.
- Внутренний буфер ограниченного размера — *buffer.memory.* При заполнении блокирует отправку на *max.block.ms.*
- *delivery.timeout.ms* — общий таймаут на доставку, равный 2-м минутам по умолчанию. В пределах этого лимита клиент будет пробовать доставить сообщение заново в случае ошибок.
- При помощи опции *Retries* можно контролировать количество попыток доставки. Важно помнить, что *Retries* могут сломать очередность сообщений если *max.in.flight.requests.per.connection* > 1.

Как бороться с дубликатами:

1. В Consumer, если у каждой записи есть внутренний ID, мы можем использовать его для атомарного сохранения результата в БД. Однако это

не всегда возможно. Например, если Consumer не работает с БД со сторонним АПИ, у которого нет идемпотентности.

2. При помощи настройки enable.idempotence в true: каждый Producer получает свой уникальный ID на брокере, а также инициализируется SEQ number. Producer отправляет сообщения с возрастающим SEQ, а брокер сверяет что каждое полученное сообщение обладает правильным SEQ (0,1,2,3...). Таким образом если отправляется дубликат — брокер отклоняет его, поскольку видит неправильный SEQ. Важно помнить, что используя idempotence, вы не можете выставить max.in.flight.requests.per.connection больше чем 5, а retiries в 0. Кроме того, опция acks должна быть равна all.

Consumer:

- Библиотека Consumer в Kafka используется для чтения данных из топиков. Клиент Consumer автоматически обрабатывает падения брокеров и перемещения лидеров партиций в кластере.
- enable.auto.commit — автоматические коммиты с интервалом auto.commit.interval.ms.
- session.timeout.ms — максимальное время между heartbeat запросами к брокеру *(контролируется брокером).*
- max.poll.interval.ms — максимальное время между вызовами poll *(контролируется самим клиентом).*
- group.id — идентификатор группы.
- group.instance.id — статический идентификатор консьюмера
- Рестарт должен уложиться в session.timeout.ms

Transactions & Exactly Once:

- Созданы для решения проблем обработки в “read-process-write” приложениях, где “read & write” производятся в/из Kafka.
- Традиционно выбор был из двух режимов обработки
- **at-least once** — вначале мы “процессим” сообщение и затем “коммитим” оффсет в Кафку *(возможна повторная обработка);*
- **at-most once** — вначале “коммитим” оффсет и только затем “процессим” сообщение *(возможна потеря данных).*

- Все хотят режим **exactly once** — возможность обработки сообщений только 1 раз. Именно тут на помощь приходят Transactions.

Главные принципы транзакционности в Кафке:

- Атомарная запись в несколько партиций одновременно.
- Защита от “зомби” из коробки.
- Изоляция: Consumers получают сообщения только успешно завершенных транзакций.

“Подводные камни”, связанные с Transactions:

- Транзакционные маркеры — обычные сообщения, которые могут несколько ломать логику поиска сообщений по timestamp *(offsetsByTime()),* если вы пользуетесь *CreateTime.*
- Ущерб производительности на мелких стримах не заметен *(±3% degradation, 1KB of records per/sec)*. Растет нагрузка на брокеры с ростом количества транзакций. Совет: использовать Transactions только там где надо (не включать по-умолчанию).
- Открытые транзакции не дают Consumers с *isolation.level*=read_committed читать сообщения выше Last Stable Offset (LSO).
- Transactions обеспечивают exactly-once только в “read-process-write” приложениях читающих и пишущих в Кафку.