Симуляция потери данных и почему важен acks

Над текстом нажмите кнопку "**Создать стенд**". До 10 минут создается стенд. Каждый стенд создан под определенный модуль курса (в данном случае это - Клиентские библиотеки), поэтому при выполнении нижеследующих инструкций, нужно запустить отсюда стенд.

Где получить логин/пароль? В настройках профиля или над текстом по кнопке "**Доступы**".

Практический стенд этого урока автоматически запускает кластер Apache Kafka, состоящий из 3 брокеров, установленных на 3 виртуальных машинах.

Каждый брокер адресуется как `node-{1-3}.<ваш номер студента>`

Потеря данных с acks=1

Заходим на наши брокеры в **3-х консольных окнах**, заменив плейсхолдер <ваш номер студента> на ваш личный номер студента:

```
ssh <ваш номер студента>@sbox.slurm.io
ssh node-{1-3}.<ваш номер студента>
export USER=<ваш номер студента>
```

1.

Клонируем репо на одном из брокеров и соберем тестового клиента

```
git clone https://gitlab.slurm.io/edu/kafka.git
cd kafka
```

2. `mvn package`

Создаем топик с 3 репликами для теста и находим текущего лидера, у вас он может отличаться! (см Leader:) Запускать команду можно на любом брокере

```
/opt/kafka_2.13-2.7.0/bin/kafka-topics.sh --bootstrap-server node-1.$USER:9092
--topic test --replication-factor 3 --partitions 1 --create
```

3. `/opt/kafka_2.13-2.7.0/bin/kafka-topics.sh --bootstrap-server node-1.$USER:9092 --topic test --describe`
4. Запускаем продюсера с acks=1 (default) на брокере с собранным клиентом

   ```
   java -cp
   test-clients/target/test-clients-1.0-SNAPSHOT-jar-with-dependencies.jar
   io.slurm.kafka.TestProducer -b node-1.$USER:9092 -c 2000000 -t test --acks
   1
   ```
5. Пока продюсер работает убиваем брокера лидера **в соответствующей ему консоли**

   ```
   sudo systemctl kill -s SIGKILL kafka.service
   ```

Ждем пока продюсер закончит работу и выдаст “Final Stats”

```
15:00:45.842 [main] INFO  io.slurm.kafka.TestProducer - --- Final Stats ---
1994540 records delivered, 5460 records failed, 79378,357942 records/sec…
```

6. `* Если вы не успели остановить брокера до того как продюсер завершил работу, просто пересоздайте топик заново и попробуйте снова!`
7. Запускаем брокера **в соответствующей ему консоли**, который ранее был лидером.

   ```
   sudo systemctl start kafka.service
   ```

Посмотрим, сколько сообщений записалось, на самом деле, на любом из брокеров

```
/opt/kafka_2.13-2.7.0/bin/kafka-run-class.sh kafka.tools.GetOffsetShell
--broker-list node-1.$USER:9092 --time -1 --topic test
```

8. `Test:0:1992356`
   Несмотря на то, что продюсер посчитал, что успешно записал 1994540 сообщений, в логе сохранилось только 1992356, а значит, **2184 сообщения были потеряны** из-за того, что брокер упал до того, как эти сообщения реплицировались фолловерами.

Отсутствие потерь с acks=all

Повторим эксперимент с acks=all (записью на min.insync.replicas). Кластер уже сконфигурирован с min.insync.replicas=2

Удалим и пересоздадим топик **(снова запоминаем лидера)** на любом из брокеров

```
/opt/kafka_2.13-2.7.0/bin/kafka-topics.sh --bootstrap-server node-1.$USER:9092
--topic test --delete
/opt/kafka_2.13-2.7.0/bin/kafka-topics.sh --bootstrap-server node-1.$USER:9092
--topic test --replication-factor 3 --partitions 1 --create
```

1. `/opt/kafka_2.13-2.7.0/bin/kafka-topics.sh --bootstrap-server node-1.$USER:9092 --topic test --describe`
2. Запускаем продюсера с acks=-1 (all) на брокере с собранным клиентом

   ```
   java -cp
   test-clients/target/test-clients-1.0-SNAPSHOT-jar-with-dependencies.jar
   io.slurm.kafka.TestProducer -b node-1.$USER:9092 -c 2000000 -t test --acks
   -1
   ```
3. Пока продюсер работает, убиваем брокера лидера **в соответствующей ему консоли**
   `sudo systemctl kill -s SIGKILL kafka.service`

Ждем пока продюсер закончит работу и выдаст “Final Stats”

```
15:00:45.842 [main] INFO  io.slurm.kafka.TestProducer - --- Final Stats ---
1994540 records delivered, 5460 records failed, 67639,039609 records/sec…
```

4. `* Если вы не успели остановить брокера до того как продюсер завершил работу, просто пересоздайте топик заново и попробуйте снова!`
5. Запускаем брокера **в соответствующей ему консоли**, который ранее был лидером.

   `sudo systemctl start kafka.service`

Посмотрим, сколько сообщений записалось, на самом деле, на любом из брокеров

```
/opt/kafka_2.13-2.7.0/bin/kafka-run-class.sh kafka.tools.GetOffsetShell
--broker-list node-1.$USER:9092 --time -1 --topic test
```

6. `test:0:1994540`
   Видим, что благодаря acks=all **ни одно из сообщений не было потеряно**!*

   * — при этом вполне возможно, что число сообщений которое хранится на брокере будет больше числа успешно записанных сообщений отображенных продюсером. Как вы думаете, откуда может возникнуть эта разница? Все дело в том, что ответ от брокера к продюсеру об успешном сохранении батча сообщений может не успеть быть послан до того, как вы

остановили лидера! Иными словами, сообщения сохранились на min.insync.replicas, но продюсер об этом просто не узнал. Такое поведение может быть причиной появления дубликатов в топике при включенной опции retries. С этим можно побороться включив у продюсера идемпотентность *(ключ "-i" в нашем тестовом клиенте)*.

- Повторите эксперимент с acks=0, какие результаты вы получили? Почему? Какую разницу в latency записи вы заметили (см. Final Stats экспериментов)?
- Удалите все созданные топики.