Эксплуатация Arenadata Streaming (Kafka, NiFi)

ARENADATA

Архитектура и инструменты Apache Kafka: Brocker, Connect, REST Proxy, Schema Regestry, ksqlDB

Agenda

- Базовые концепции и объекты Kafka:
  Consumers, Producers, Topics, Partitions, Segments, Replication, Offset, Messages, consumer groups, Brokers, Schemas
- Topics Kafka:

  основные операции (создание, управление, мониторинг, console CLI, partitions, replication, compaction, retention). Гарантии надежности Kafka (доставка/потребление).
  Лабораторная работа.
- Zookeeper. Выбор лидера
- Producers Kafka:

  Запись сообщений в Kafka, console Producer. Настройка Producers.
- Consumers Kafka:

  Consumer groups, ребалансировка разделов. Настройка Consumers. Изменение параметров Topics, consumer groups, Partitions.
- Kafka Connect:

  Основные понятия и инструменты. FileStream Connectors. CDC Debezium. Kafka ADB Connectors.
- Kafka REST Proxy, ksqlDB:

  Основные возможности и примеры использования.
- Мультикластерные архитектуры:

  Топология мультикластерной репликации. Настройка MirrorMaker, Mirror Connectors.

Базовые концепции и объекты Kafka

Consumers, Producers, Topics, Partitions, Segments, Replication, Offset, Messages, consumer groups, Brokers, Schemas

ARENADATA

История Apache Kafka

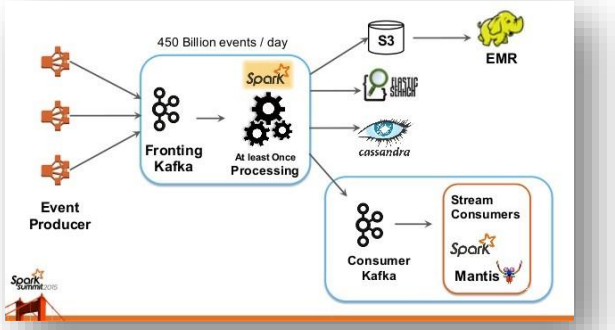
Краткая история Apache Kafka:

Первые разработчики Джей Крепс, *Нархид Ния* и Цзюнь Жао

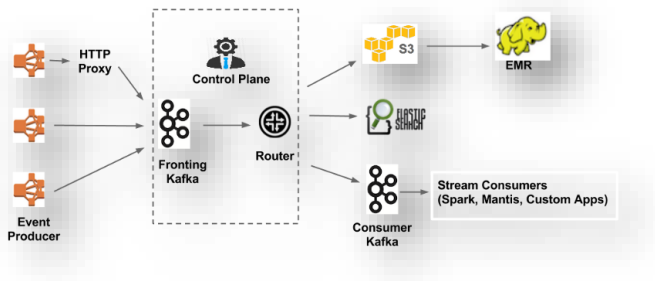
Название "Kafka" было выбрано разработчиками в честь писателя Франца Кафки («because it is – a system optimized for writing», Джей Крепс).

- 2007: Система была задумана как решение для сбора и обработки больших объемов данных в LinkedIn.
- 2010: Kafka была переведена в статус open-source проекта под управлением Apache Software Foundation.
- 2011: Kafka стала доступна как проект Apache Incubator.
- 2012: Kafka стала топ-уровнем проектом Apache
- 2014: Основные авторы покинули LinkedIn и основали компанию Confluent для коммерциализации проекта

Примеры применения: (LinkedIn, Netflix, Uber, AirBnb, Walmart, Twitter (X), The NY Times, ...)

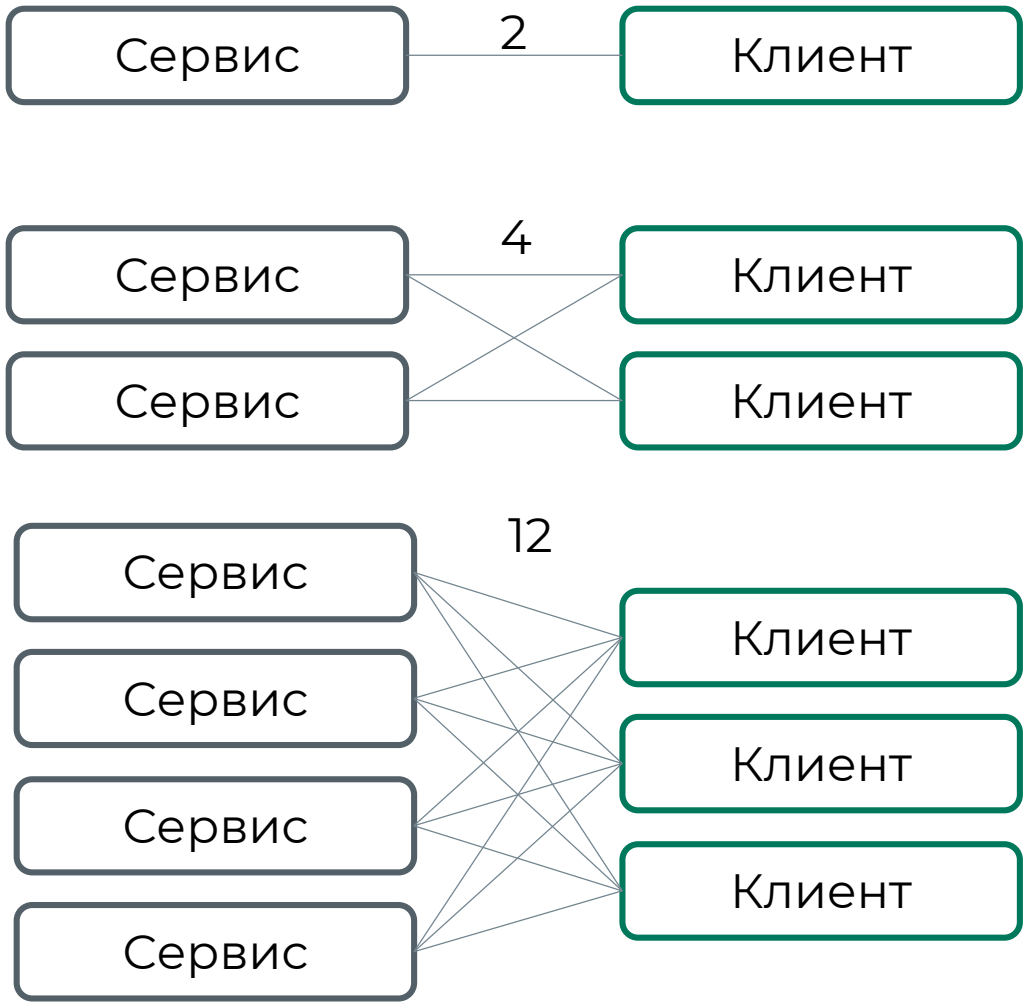


https://www.linkedin.com/pulse/kafkas-origin-story-linkedin-tanvir-ahmed



https://www.loginworks.com/blogs/how-netflix-use-data-to-win-the-race/

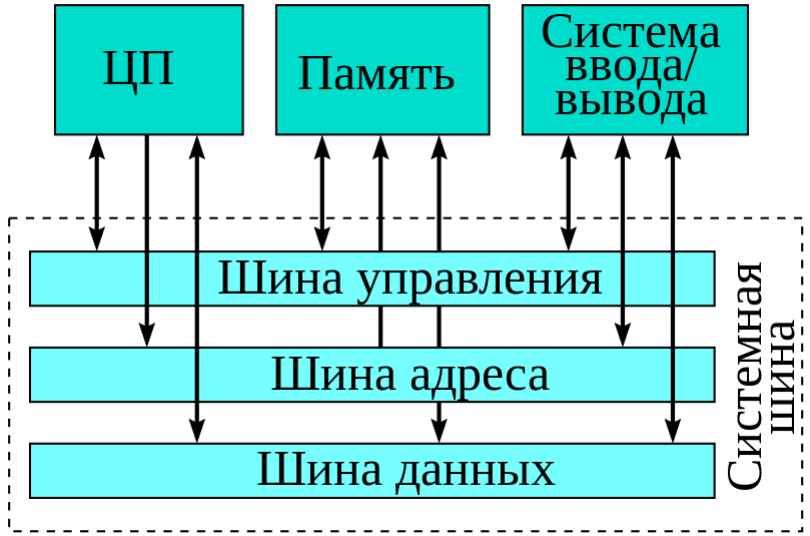
ADS: Apache Kafka

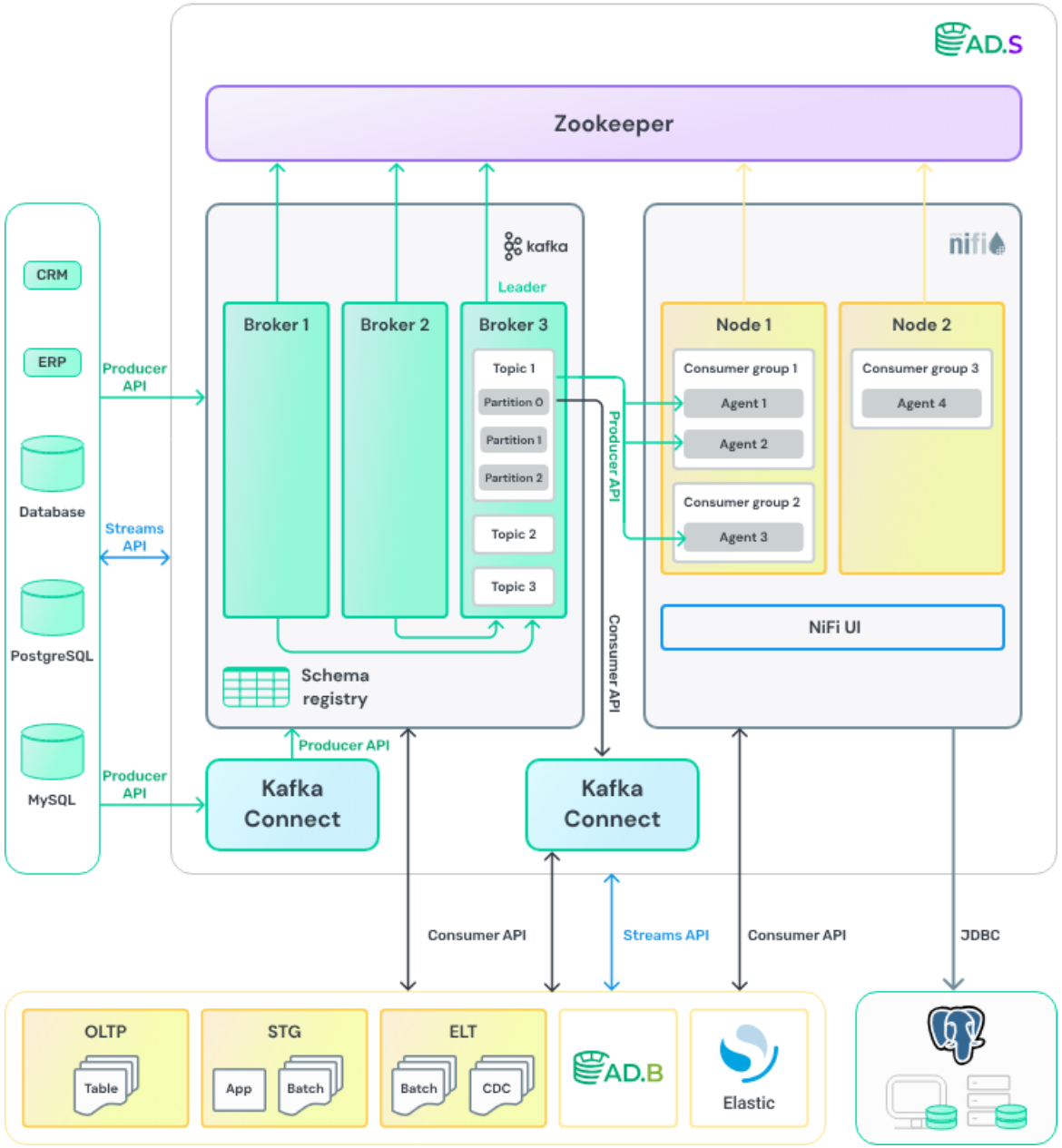


Много подключений!
Как ими управлять?
Как отслеживать их состояние?

Может чего-то не хватает?

ADS: Apache Kafka





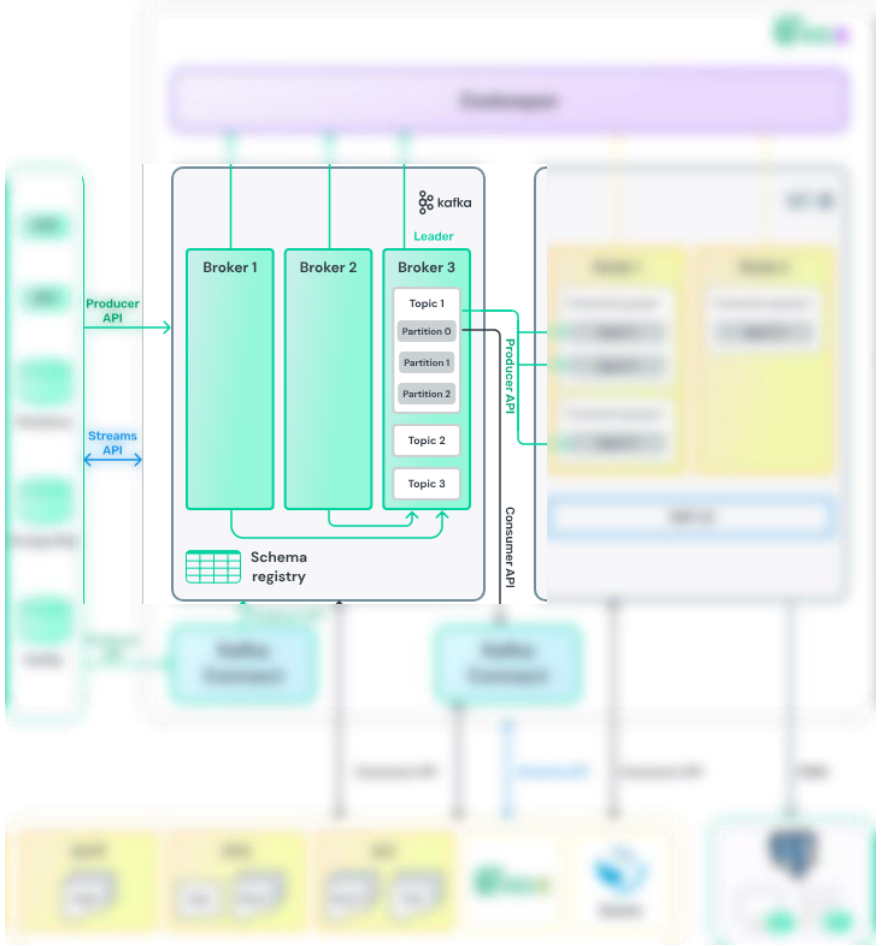
Apache Kafka, RabbitMQ, Redis

Характеристика	Apache Kafka	RabbitMQ	Redis
Основное назначение	Потоковая обработка данных	Брокер сообщений	Хранилище структур данных в памяти
Протоколы	Собственный протокол Kafka	AMQP, MQTT, STOMP	Собственный протокол Redis
Пропускная способность	Высокая	Средняя	Высокая
Масштабируемость	Отличная	Хорошая	Отличная
Долговременное хранение	Да	Нет (без дополнительных настроек)	Нет
Управление очередями	Простое	Гибкое	Простое
Гарантия доставки	Да	Да	Нет
Поддержка транзакций	Да	Да	Нет
Подходит для	Большие объемы данных и потоковая обработка	Сложные маршрутизации и шаблоны обмена сообщениями	Кэширование, управление сессиями, публикация/подписка

Consumers, Producers, Topics, Partitions, Segments

Основные компоненты Kafka

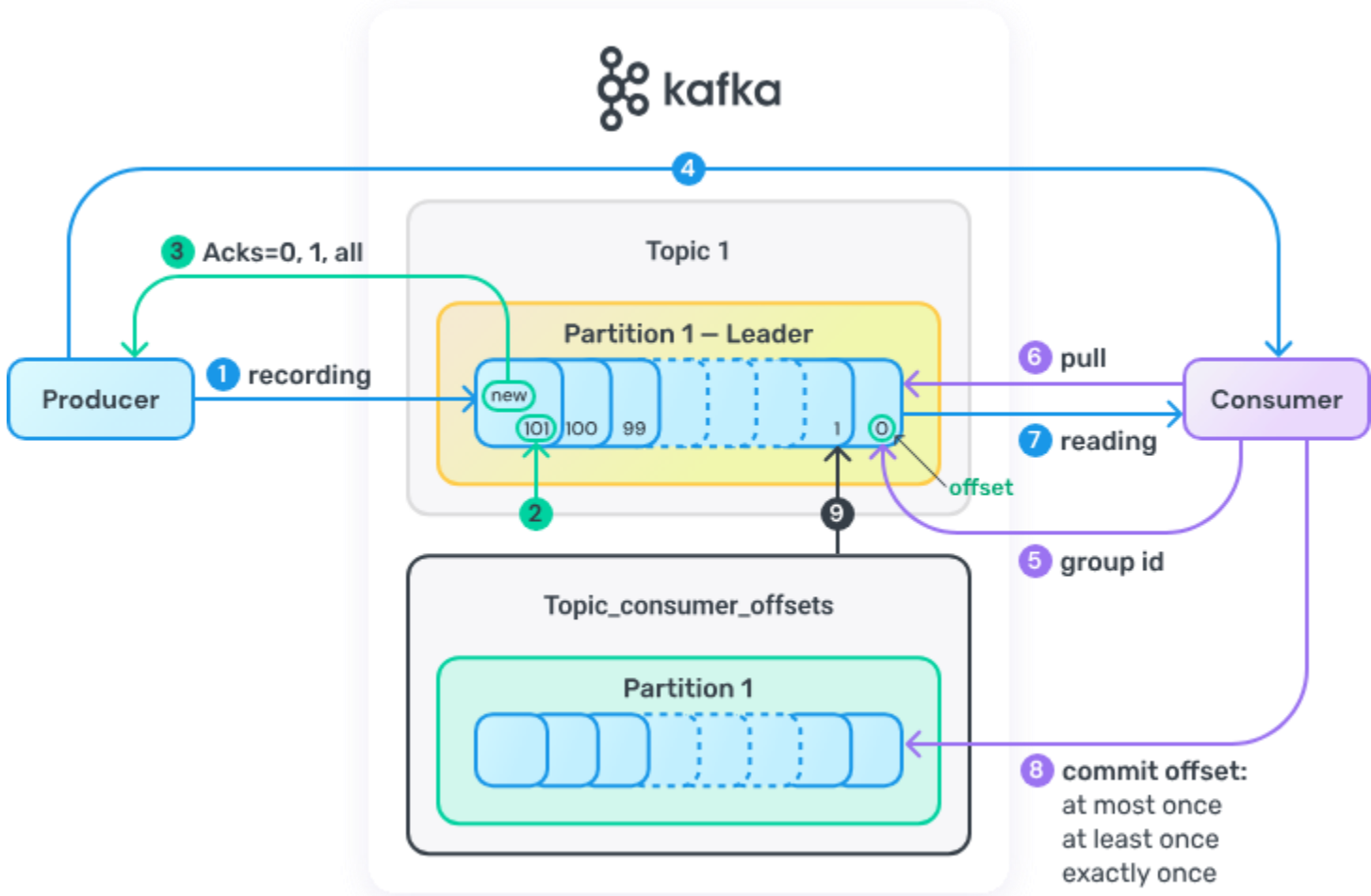
- Сообщение (message) — запись произошедшего события, в том числе состояния объекта, значения физической величины или любого другого параметра, который требует отслеживания, хранения или передачи в другую систему.
- Производитель (producer) — клиентское приложение, публикующее (записывающее) сообщения в Kafka.
- Потребитель (consumer) — клиентское приложение, которое подписывается (читает и обрабатывает) сообщения.
- Топик (topic) — категория (тема), в которой сообщения публикуются в Kafka. В топике сообщения пишутся в журнал коммитов.
- Журнал коммитов (commit log) – упорядоченная структура сообщений, доступная только для записи. В журнале коммитов нельзя ни изменять, ни удалять данные. Большие журналы разделяют на партиции. Если журнал состоит из одной партиции, то он и является партицией.
- Партиция (partition) — упорядоченная неизменяемая последовательность записей, содержимое журнала коммитов.
- Брокер (broker) — сервер, на котором хранятся партиции.
- Сегмент (segment) — часть партиции, хранящаяся в отдельном файле на диске.
- Группа потребителей (consumer group) — несколько потребителей, объединнных в группу для получения сообщений из одного или нескольких топиков. Партиции одного топика автоматически равномерно распределяются между потребителями в одной группе. Каждая партиция доступна только одному потребителю из группы.
- offset — это уникальный идентификатор каждого сообщения в разделе (partition) топика. Offset позволяет потребителям (consumers) отслеживать, какие сообщения были уже обработаны



Consumers, Producers, Topics, Partitions, Segments

Производитель (Producer) при записи сообщений указывает:

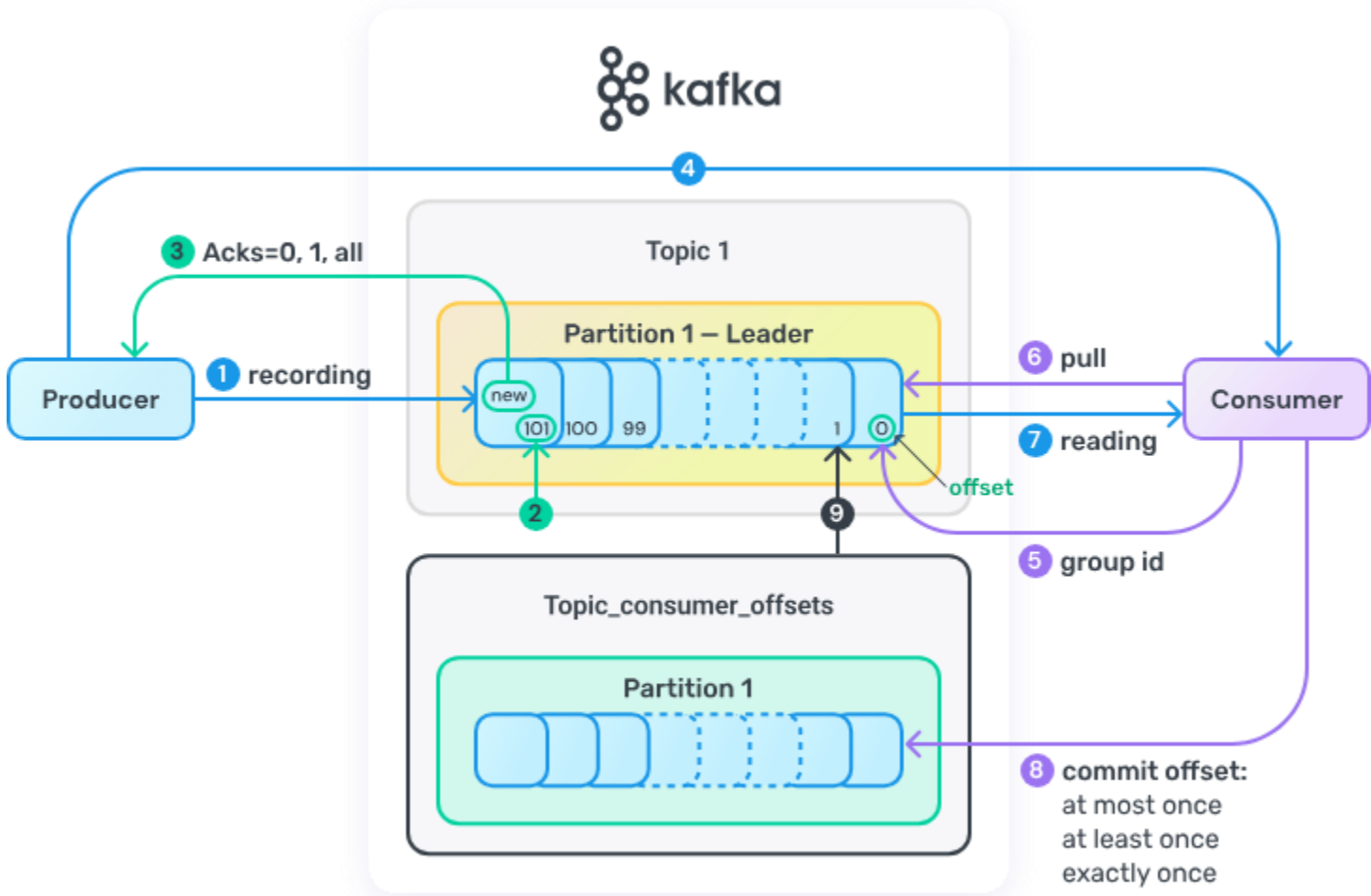
- Топик (topic), в который будут записаны данные;
- Партицию (partition), в которую будут записаны сообщения (messages), имеющие одинаковый ключ (если это необходимо для структурирования данных в топике);
- срок хранения сообщений в Kafka (log.retention.bytes, log.retention.[ms, minutes, hours]);
- необходимое сжатие (compression.type);
- необходимый уровень подтверждения о записи сообщения в партицию и ее реплик – acks (acknowledgement);
- коэффициент репликации партиций (Replication Factor).



Consumers, Producers, Topics, Partitions, Segments

Потребитель для чтения сообщений указывает:

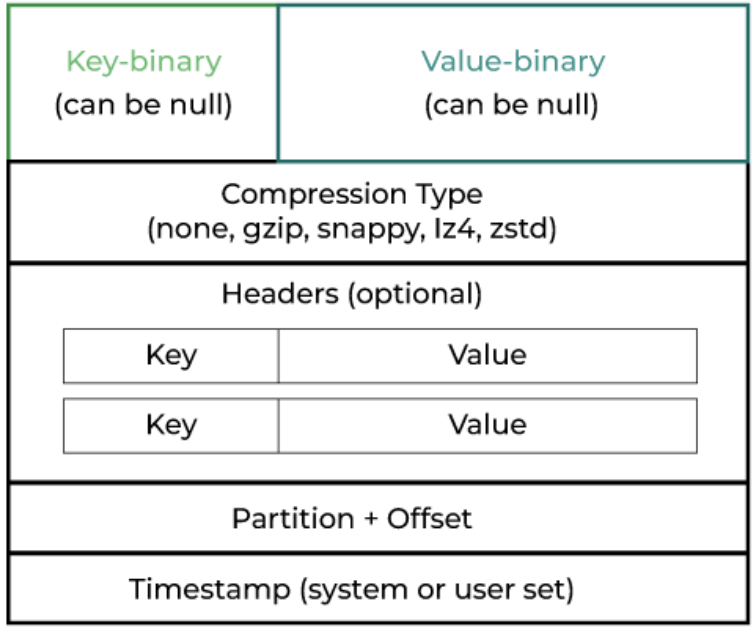
- group_id для каждой из групп потребителя, который определяет, откуда группа будет читать сообщения;
- максимальное время ожидания клиентом ответа на запрос (request.timeout.ms);
- необходимый уровень подтверждения доставки сообщений (семантика):
  - не менее одного раза (at-least-once) – сообщение будет отправляется потребителям до тех пор, пока те не подтвердят его получение;
  - не более одного раза (at-most-once) – сообщение отправляется только один раз и в случае сбоя не отправляется повторно;
  - точно один раз (exactly-once) – потребитель гарантированно получит сообщение ровно один раз



Consumers, Producers, Topics, Partitions, Segments

Потребитель для чтения сообщений указывает:

- **Key (Ключ)**: Опциональный элемент, который используется для определения партиции в топике, куда будет отправлено сообщение. Ключи важны для обеспечения порядка сообщений в пределах определенной партиции.
- **Value (Значение)**: Основные данные сообщения. Это может быть любой набор байтов, и Kafka не интересует формат этих данных – они могут быть текстом, изображением, сериализованным объектом и т.д.



Лишь единицы смогут внести сюда какой-то смысл:

00000000 00000000 00000000
00000000 00000000 00000000
00000000 00000000 00000000

- **Timestamp (Временная метка)**: Время создания сообщения. Может быть установлено производителем или автоматически Kafka при получении сообщения.
- **Headers (Заголовки)**: Опциональные метаданные, которые могут быть добавлены к сообщению. Заголовки представляют собой список пар ключ-значение и могут использоваться для передачи дополнительной контекстной информации, которая не является частью основного тела сообщения. Они могут быть использованы для управления поведением потребителей и производителей, а также для улучшения трассировки и логирования в системах.

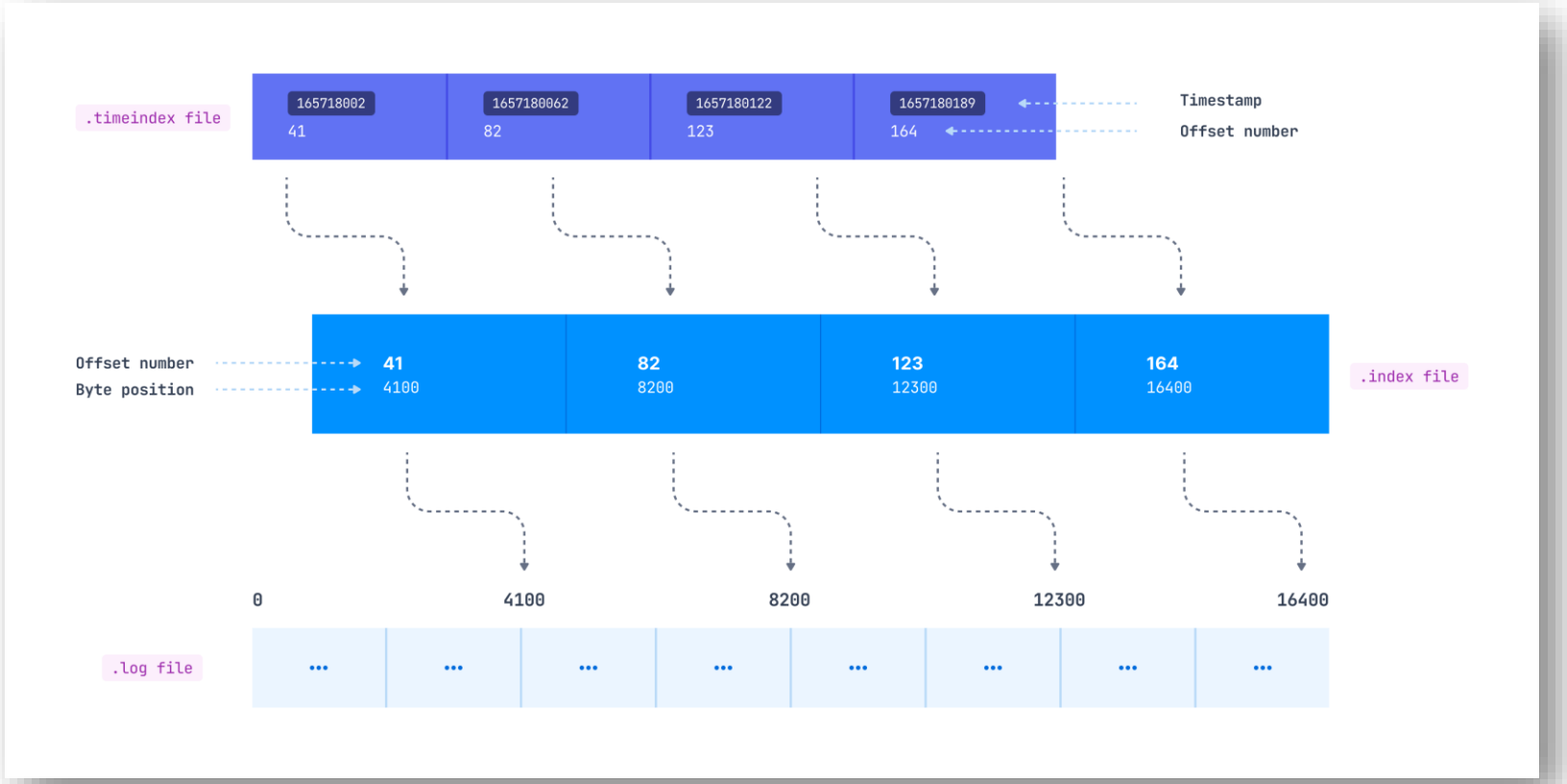
Replicas, Partitions, Segments

- Коэффициент репликации (replication factor) – 3x (рекомендация)
- Каждая партиция имеет три реплики на разных брокерах (но не всегда).
- Брокеры разбивают каждую партицию на сегменты, каждый из которых хранится в отдельном файле данных на диске
- Размер сегмента можно настроить параметром log.segment.bytes. Старые сегменты удаляются или сжимаются в соответствии с политикой хранения, заданной параметрами log.retention.hours, log.retention.bytes и log.cleanup.policy.
- .log: Это основной файл журнала для каждого раздела, который содержит фактические сообщения Kafka. Сообщения записываются в этот файл в порядке их поступления.
- .index: Файл индекса смещений (offset index) помогает Kafka быстро находить сообщения в файле .log по их смещению. Каждая запись в индексе указывает на позицию в файле .log, что позволяет брокеру эффективно получать сообщения по запросу.
- .timeindex: Файл временного индекса (timestamp index) работает аналогично файлу .index, но используется для индексации сообщений по временной метке. Это позволяет выполнять поиск сообщений, основанный на времени, а не на смещении.
- .snapshot – содержит снимок состояния производителя относительно последовательностей IDs используемый во избежание дублирования записей.
- leader-epoch-checkpoint – это файл журнала, который используется для отслеживания эпох лидеров для каждого раздела. Эпоха лидера – это монотонно возрастающий номер, который обновляется каждый раз, когда новый лидер выбирается для раздела.

```
-rwxrwxr-x. 1 kafka kafka 10485760 Jun 16 18:49 00000000000000000000.index
-rwxrwxr-x. 1 kafka kafka      165 Jun 13 17:45 00000000000000000000.log
-rwxrwxr-x. 1 kafka kafka 10485756 Jun 16 18:49 00000000000000000000.timeindex
-rwxrwxr-x. 1 kafka kafka       10 Jun 16 18:12 00000000000000000005.snapshot
-rw-r--r--. 1 kafka kafka       12 Jun 16 18:49 leader-epoch-checkpoint
-rwxrwxr-x. 1 kafka kafka       43 Jun 13 17:44 partition.metadata
```

Replicas, Partitions, Segments

- .log: Это основной файл журнала для каждого раздела, который содержит фактические сообщения Kafka. Сообщения записываются в этот файл в порядке их поступления.
- .index: Файл индекса смещений (offset index) помогает Kafka быстро находить сообщения в файле .log по их смещению. Каждая запись в индексе указывает на позицию в файле .log, что позволяет брокеру эффективно получать сообщения по запросу.
- .timeindex: Файл временного индекса (timestamp index) работает аналогично файлу .index, но используется для индексации сообщений по временной метке. Это позволяет выполнять поиск сообщений, основанный на времени, а не на смещении.



Replicas, Partitions, Segments

- Формат репликации называется ISR (in-sync replicas, min.insync.replicas=2).
- Одна является лидером (leader), а другие последователями (followers). Производитель может управлять синхронностью/асинхронностью репликации при помощи параметра acks(acknowledgement).

get /arenadata/cluster/3/brokers/topics/mm2-offset-syncs.target.internal/partitions/0/state

{"controller_epoch":3,"leader":1002,"version":1,"leader_epoch":6,"isr":[1002,1003,1001]}

cat /kafka-logs/mm2-offset-syncs.target.internal-0/leader-epoch-checkpoint

0

2

0 0

6 133

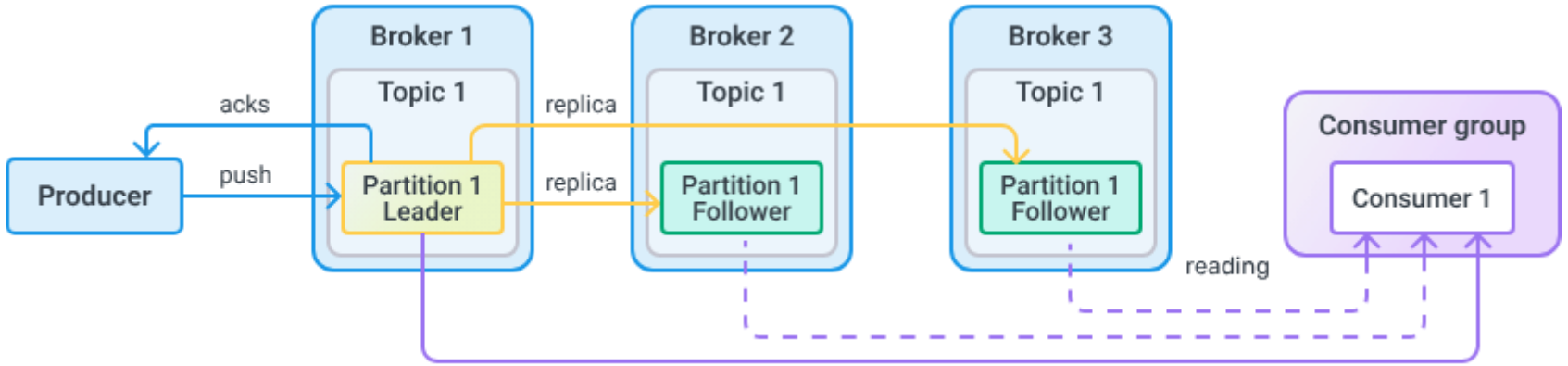
CMAK

Partition	Latest Offset	Leader	Replicas	In Sync Replicas	Preferred Leader?
0	133	1002	(1002,1001,1003)	(1002,1003,1001)	true

- **Синхронная отправка** означает, что продюсер будет ждать подтверждения от брокера перед продолжением. Это обеспечивает более высокую гарантию доставки, но может снизить производительность из-за задержек ожидания подтверждения.
- **Асинхронная отправка** позволяет продюсеру продолжать отправку сообщений без ожидания подтверждения, что увеличивает производительность за счет потенциального риска потери сообщений.
- acks=0: Продюсер не ждет подтверждения (асинхронно).
- acks=1: Продюсер ждет подтверждения только от лидера партиции (может быть как синхронно, так и асинхронно).
- acks=all: Продюсер ждет подтверждения от всех реплик в списке ISR (синхронно).
- max.in.flight.requests.per.connection – определяет максимальное количество неподтвержденных запросов, которые продюсер может отправить на одно соединение до того, как он получит подтверждение от сервера.

Replicas

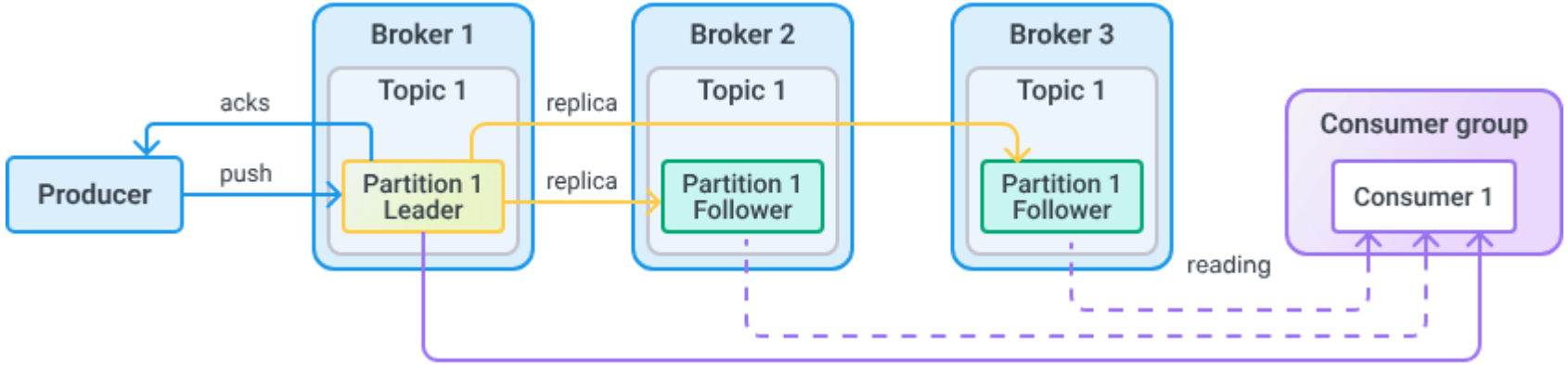
- Операции записи сообщений выполняются через брокера-лидера партиции (leader). Операции чтения сообщений могут быть выполнены как через лидера (по умолчанию), так и через follower (2.4+) .
- Выбор лидера для каждой партиции производится при помощи ZooKeeper
- Фактор репликации (replication factor) – default.replication.factor (определяется на уровне кластера) или может быть задан при создании топика. Можно переопределить позже.
- replica.lag.time.max.ms – это временной период, по истечении которого лидер принимает решение, что реплика записана (если произошло подтверждение записи) или не записана (не произошло подтверждение записи) сообщение.
- min.insync.replicas — минимальное количество реплик, которые должны подтвердить запись. Только при достижении числа реплик, указанного в этом параметре, лидер передает данные о записанной партиции (метаданные) и о созданных ISR в контроллер (Zookeeper).



Replicas

Стандартный рекомендуемый сценарий для обеспечения высоких гарантий надежности:

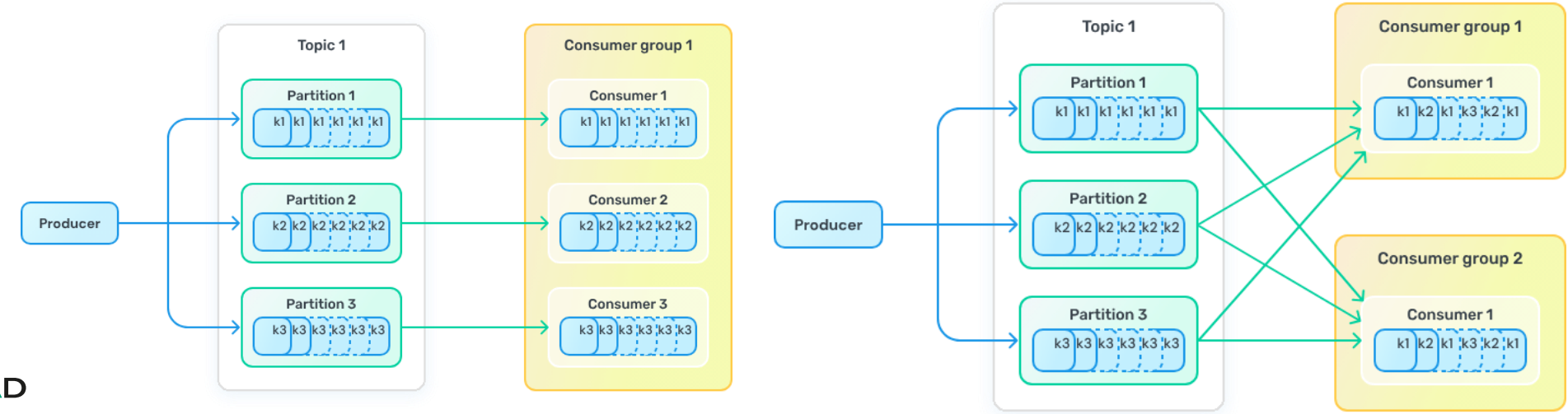
- default.replication.factor = 3
- min.insync.replicas = 2
- acks = all



- Для выбора ближайших синхронизированных реплик используется плагин RackAwareReplicaSelector. Для его работы должны быть настроены параметры:
- broker.rack — указание принадлежности к определенной стойке для брокеров. Для каждого брокера должна быть назначена стойка. Для эффективной работы плагина рекомендуется настроить одинаковое количество брокеров на стойку — таким образом реплики будут равномерно распределены по брокерам.
- client.rack — настройка идентификатора стойки для подключения потребителей.

Consumer Groups

- Группы потребителей в Apache Kafka — это механизм, который позволяет нескольким потребителям совместно обрабатывать данные из одного или нескольких топиков.
- Группы потребителей управляются координатором группы потребителей.
- Координатор группы потребителей — один из брокеров, который делает опрос всех потребителей группы потребителей. Если потребитель перестанет отправлять heartbeats, координатор инициирует ребалансировку потребителей.
- У группы потребителей есть group_id, в котором есть информация, к какому топику подключаться и откуда начинать чтение. Kafka равномерно распределяет все партиции между потребителями одной группы
- **Балансировка нагрузки**
- **Отказоустойчивость**
- **Смещение (Offset)**: Kafka отслеживает смещение для каждого потребителя в группе, что позволяет потребителям продолжить чтение с того места, где они остановились в случае сбоя или перезапуска.
- **Параллельность обработки**: Потребители в разных группах могут независимо читать данные из одних и тех же топиков, не мешая друг другу.



Основные утилиты Kafka (/usr/bin/, /usr/lib/kafka/bin/)

- connect-distributed.sh
- connect-mirror-maker.sh
- connect-standalone.sh
- kafka
- kafka-acls.sh
- kafka-broker-api-versions.sh
- kafka-cluster.sh
- kafka-configs.sh
- kafka-console-consumer.sh
- kafka-console-producer.sh
- kafka-consumer-groups.sh
- kafka-consumer-perf-test.sh
- kafka-delegation-tokens.sh
- kafka-delete-records.sh
- kafka-dump-log.sh
- kafka-features.sh
- kafka-get-offsets.sh
- kafka-leader-election.sh
- kafka-log-dirs.sh
- kafka-metadata-quorum.sh
- kafka-metadata-shell.sh
- kafka-mirror-maker.sh
- kafka-producer-perf-test.sh
- kafka-reassign-partitions.sh
- kafka-replica-verification.sh
- kafka-run-class.sh
- kafka-server-start.sh
- kafka-server-stop.sh
- kafka-storage.sh
- kafka-streams-application-reset.sh
- kafka-topics.sh
- kafka-transactions.sh
- kafka-verifiable-consumer.sh
- kafka-verifiable-producer.sh
- trogdor.sh
- zookeeper-security-migration.sh
- zookeeper-server-start.sh
- zookeeper-server-stop.sh
- zookeeper-shell.sh
- kafka-avro-console-consumer
- kafka-avro-console-producer

- **kafka-topics.sh**: Утилита для создания, удаления, описания или изменения Топиков.
- **kafka-console-producer.sh**: Позволяет отправлять сообщения в тему Kafka из командной строки.
- **kafka-console-consumer.sh**: Утилита для чтения сообщений из темы Kafka, полезна для отладки.
- **kafka-consumer-groups.sh**: Утилита для управления и отслеживания состояния групп потребителей.
- **kafka-acls.sh**: Утилита для управления списками контроля доступа (ACLs) в Kafka.
- **kafka-configs.sh**: Утилита для изменения конфигураций брокеров, тем и клиентов.
- **kafka-avro-console-consumer:**используется для чтения данных из тем Kafka, где сообщения сериализованы с использованием Avro.
- **kafka-reassign-partitions.sh:** используется для балансировки партиций между брокерами в кластере Kafka.
- **kafka-producer-perf-test.sh:** предназначена для тестирования производительности производителей Kafka. Она позволяет измерить скорость отправки сообщений в темы Kafka и оценить производительность и пропускную способность системы.
- ...

Topics Kafka

Основные операции (создание, управление, мониторинг, console CLI, partitions, replication, compaction, retension). Гарантии надежности Kafka (доставка/потребление).
Лабораторная работа

Основные операции: создание

- Console CLI:

```
kafka-topics.sh --create --bootstrap-server localhost:9092 --replication-factor 1 --partitions 3 --topic topic_name
```

- Admin API Java-клиент Kafka:

```
Properties config = new Properties();
config.put(AdminClientConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9092");
AdminClient admin = AdminClient.create(config);
NewTopic newTopic = new NewTopic("topic_name", 3, (short) 1);
admin.createTopics(Collections.singletonList(newTopic));
```

- Автоматическое создание топиков: auto.create.topics.enable = true

```
Properties props = new Properties();
props.put("bootstrap.servers", "localhost:9092");
props.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer");
props.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer");
producer.send(new ProducerRecord<>("topic_name", "key", "value"));
```

- Kafka Streams API
- Kafka Connect
- Kafka REST Proxy:

```
curl -X POST -H "Content-Type: application/vnd.kafka.json.v2+json" \

--data '{"name": "topic_name", "config": {"partitions": 3, "replication-factor": 1, "configs": {"cleanup.policy": "compact"}}}' \

http://localhost:8082/topics
```

- CMAK
- …

Основные операции: создание

- cleanup.policy: Определяет, как Kafka будет удалять или сжимать старые данные (delete: удаляет сообщения старше заданного времени хранения, compact: уплотняет топик путём удаления всех копий сообщения, кроме последней, по ключу).
- Retention Time: Время хранения сообщений в миллисекундах. По истечении этого времени сообщения могут быть удалены.
- Retention Bytes: Максимальный размер лога топика в байтах. По достижении этого размера старые сообщения будут удаляться.
- Max Message Bytes: Максимальный размер сообщения, который может быть отправлен в топик.
- Segment Bytes: Размер сегмента лога в байтах. Когда лог достигает этого размера, он будет закрыт и создан новый сегмент.
- Segment Index Bytes: Размер индекса сегмента лога в байтах. Это помогает ускорить доступ к данным.
- Segment Ms: Время в миллисекундах, после которого текущий сегмент будет закрыт и создан новый, даже если размер сегмента ещё не достиг Segment Bytes.
- Segment Jitter Ms: Время в миллисекундах, на которое Kafka может случайным образом отклоняться от Segment Ms, чтобы избежать одновременного создания сегментов.
- Flush Messages: Количество сообщений, после которого Kafka будет принудительно сбрасывать данные на диск.
- Flush Ms: Время в миллисекундах, после которого Kafka будет принудительно сбрасывать данные на диск.
- ...

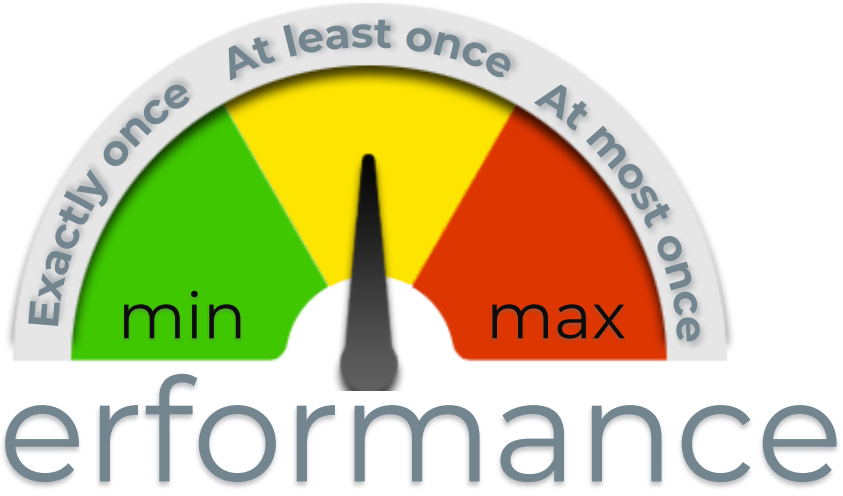
Гарантии надежности Kafka (доставка/потребление)

Проблемы доставки сообщений:

- потеря сообщений при различных сбоях (производителя, потребителя, системы обмена сообщениями, сети);
- дублирование сообщений.

Семантики доставки:

- At most once — сообщения будут обработаны один раз или не будут обработаны вообще (потеряны).
- At least once — сообщения будут обработаны хотя бы один раз.
- Exactly once — каждое сообщение будет обработано один и только один раз.



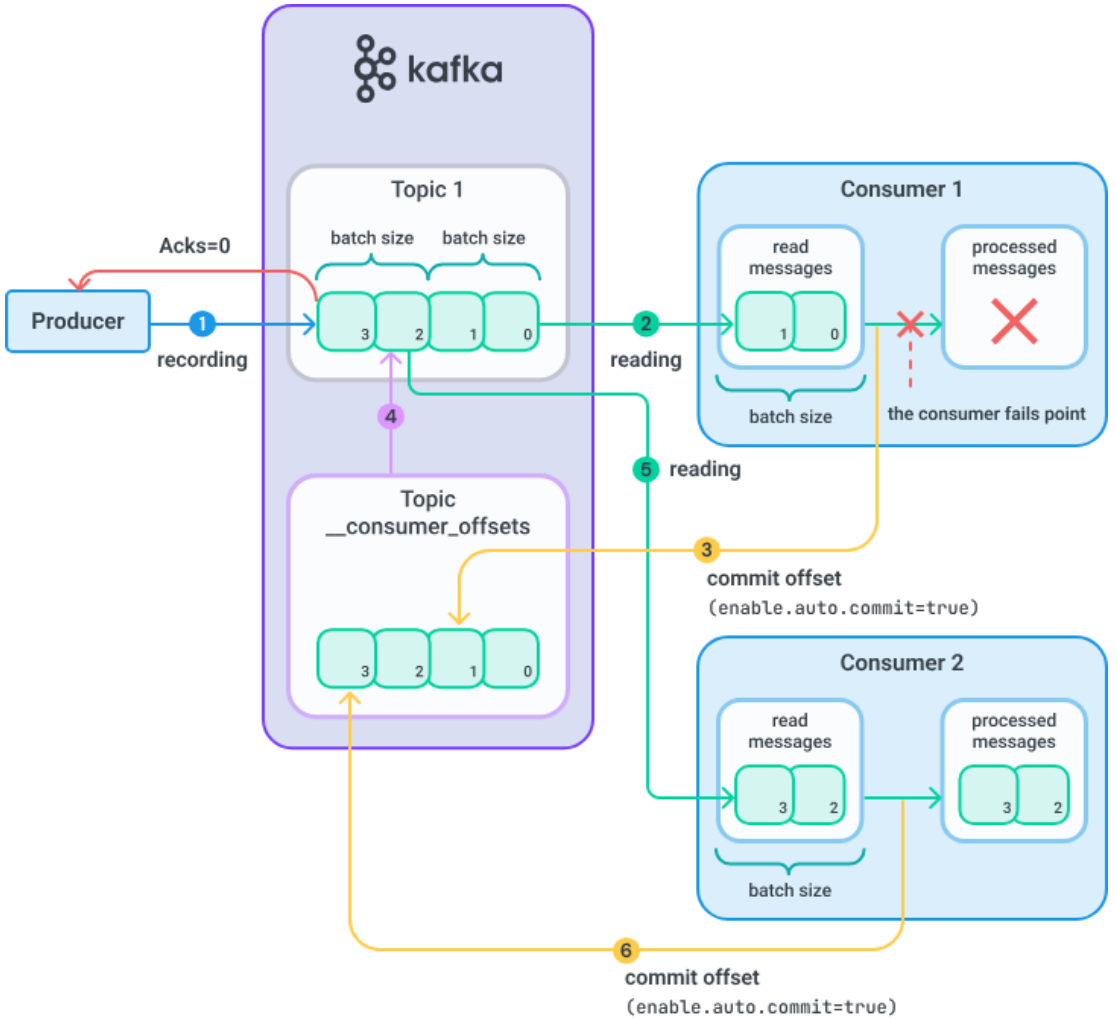
Гарантии надежности Kafka: At most once

At most once — сообщения будут обработаны один раз или не будут обработаны вообще (потеряны).

1. Допустим, что производитель Producer записывает сообщения пакетами (batch) в топик Kafka (размер пакета устанавливается параметром batch.size), не ожидая подтверждения записи (параметр acks = 0). Все сообщения считаются отправленными.
2. Потребитель Consumer 1 читает сообщения также пакетами.
3. На стороне потребителя Consumer 1 автоматически (параметр enable.auto.commit = true) выполняется коммит смещений (commit offset) всех сообщений, которые были прочитаны за определенный период времени (определяется параметром auto.commit.interval.ms).
4. Если после чтения пакета сообщений происходит отказ потребителя, дальнейшее чтение выполняет новый потребитель Consumer 2.
5. Новому потребителю назначается новый пакет сообщений, начиная с последнего зафиксированного смещения.
6. Потребитель Consumer 2 делает коммит смещений и обрабатывает полученные сообщения.

Потеря сообщений может произойти на:

- Producer;
- Broker;
- Consumer.



Гарантии надежности Kafka: At least once

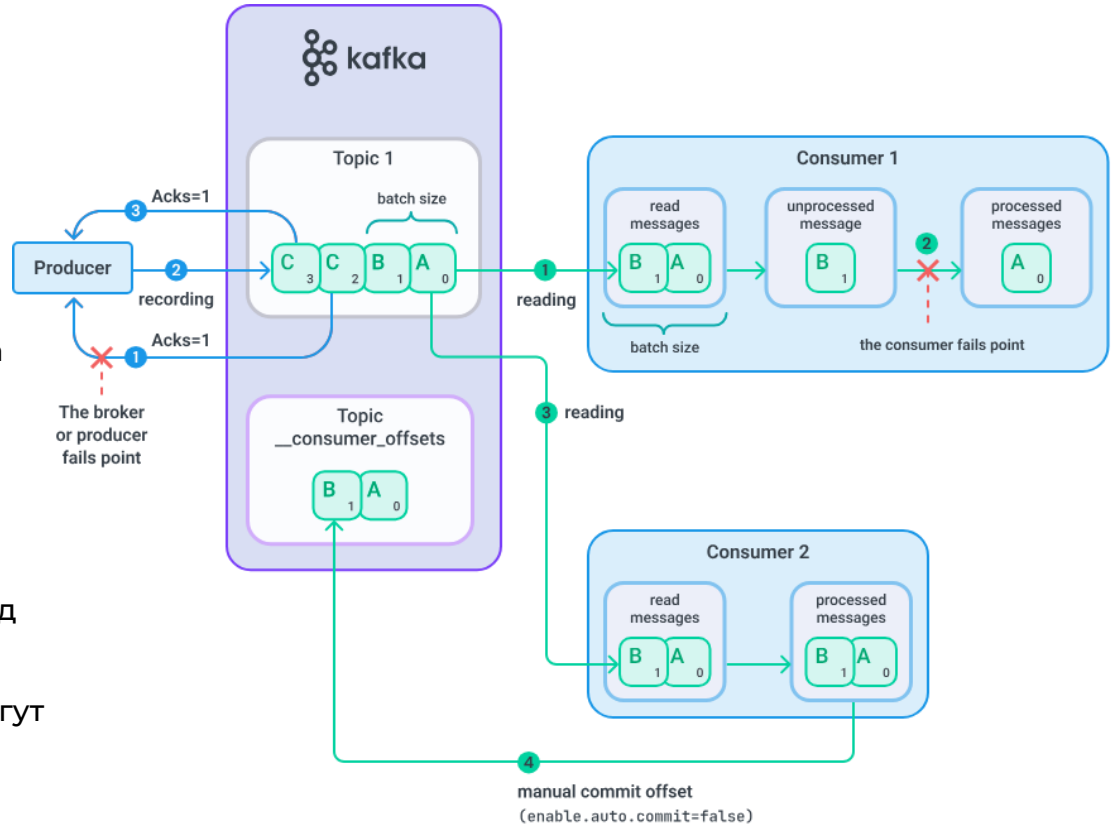
- At least once — сообщения будут обработаны хотя бы один раз.
- Можно отключить коммит смещений (enable.auto.commit = false).
  Ручной коммит смещений в потребителе: Consumer.commitSync/Async

Описание семантики и возможной проблемы на стороне производителя или брокера:

1. Producer записывает сообщение C со смещением 2 в топик Kafka и не получает подтверждение записи от лидера партиции (acks = 1).
2. Производитель Producer осуществляет повторные попытки записи сообщения, пока не истекут значения параметров retries и delivery.timeout.ms.
3. При успешной записи производитель получает подтверждение от брокера.

Проблемы:

- Дублирование сообщений «C»: если после записи сообщения в топик случился отказ брокера или производителя. Появляются два одинаковых сообщения, записанных под разными смещениями.
- Если производитель пишет batch-ами и повторил попытку записи, дублироваться могут несколько сообщений за одну итерацию записи.



Описание семантики и возможной проблемы на стороне потребителя:

1. Потребитель Consumer 1 читает записанные сообщения A и B из Kafka.
2. Представим, что случился отказ потребителя Consumer 1 после обработки A и до обработки B, и они находятся в одном пакете. Тогда в зависимости от логики приложения потребителя, коммит смещения A может быть не выполнен.
3. Так как коммит смещений A и B не выполнен, чтение этих же сообщений выполняет новый потребитель Consumer 2.
4. При успешной обработке сообщений Consumer 2 выполняет коммит смещений A и B.

Проблемы: дублирование обработки сообщений «A», если после обработки произошел отказ потребителя и не удалось выполнить коммит смещения.

Гарантии надежности Kafka: Exactly once. Идемпотентный производитель

- Exactly once — каждое сообщение будет обработано один и только один раз.

Идемпотентный производитель — производитель, каждому из которых назначается идентификатор производителя (PID). PID включается каждый раз, когда данный производитель отправляет сообщения брокеру Kafka (enable.idempotence = true).

- Каждое сообщение от идемпотентного производителя получает последовательно увеличивающийся порядковый номер (sequence), отличный от смещения.
- Для каждой партиции топика, которой производитель отправляет сообщения, поддерживается отдельная последовательность. Брокер для каждой партиции отслеживает наибольшую записанную комбинацию PID+порядковый номер. Если получено сообщение с меньшим порядковым номером, чем предыдущее записанное, оно игнорируется.
- Сообщение будет записано только один раз, независимо от того, сколько попыток отправки было предпринято.

Дополнительные параметры:

max.in.flight.requests.per.connection (определяет максимальное количество неподтвержденных запросов, которые могут быть отправлены на одно соединение) — не более 5 (позволит отправлять до 5 запросов одновременно до того, как будет получено подтверждение для первого из них);

retries > 0;

acks = all.

Гарантии надежности Kafka: Exactly once. Транзакции

- Exactly once — каждое сообщение будет обработано один и только один раз.

Транзакция — атомарная запись в один или несколько топиков и партиций Kafka. Все сообщения, включенные в транзакцию, будут успешно записаны, либо ни одно из них не будет записано. Если транзакция прерывается, все сообщения транзакции недоступны для чтения потребителям.

Работа транзакций в Kafka, термины:

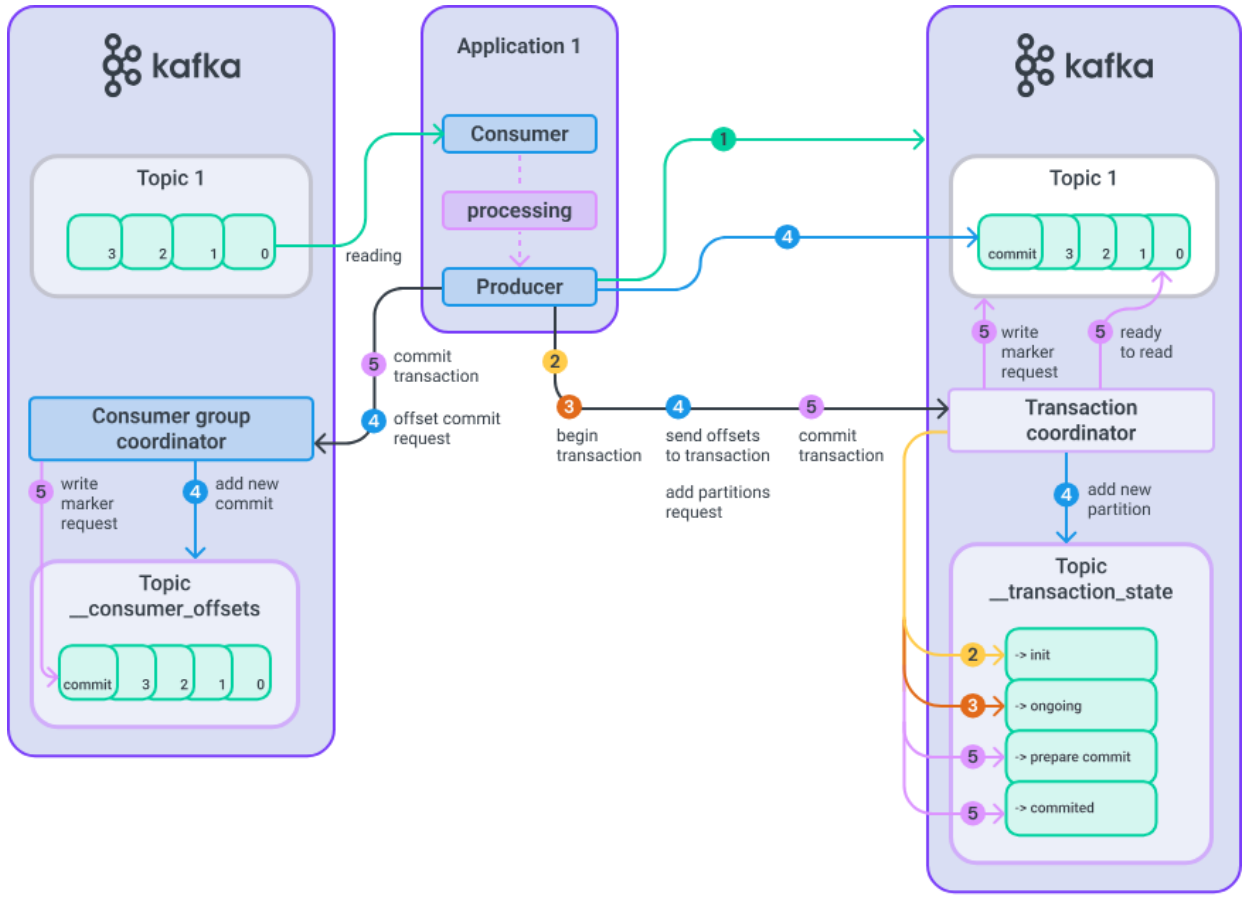
- Идентификатор транзакции (transactional.id) — индивидуальный идентификатор, присваиваемый производителю транзакций.
- Координатор транзакций (transaction coordinator) — модуль, работающий внутри каждого брокера Kafka. Только один координатор владеет зарегистрированным transactional.id. Для надежной работы координатора используется протокол репликации Kafka и процесс выбора лидера. Координатор транзакций — единственный компонент, который может читать и записывать журнал транзакций.
- Журнал транзакций (__transaction_state) — внутренний топик Kafka, который хранит последнее состояние транзакции. Возможные состояния транзакций:
  - ongoing – транзакция открыта и сообщения читаются/пишутся потребителями/производителями,
  - prepare_commit – проверяется, что все записи (сообщения) и смещения готовы к фиксации и что нет ошибок, при последующем завершении транзакции,
  - committed – конечное состояние транзакции. После этого сообщения становятся доступными для других потребителей.
- Каждый координатор транзакций управляет некоторыми партициями в журнале транзакций, является лидером для них.
- Эпоха (epoch) — идентификатор используется для обнаружения устаревших транзакций и предотвращения их влияния на текущее состояние системы.
- Зомби-экземпляр — экземпляр производителя, который начал запись сообщений и потерпел неудачу, не продолжает запись.

Гарантии надежности Kafka: Exactly once. Транзакции

- Exactly once — каждое сообщение будет обработано один и только один раз.

Запись транзакций:

1. Поиск координатора транзакций при помощи запроса FindCoordinatorRequest. Производитель подключается к любому известному брокеру кластера, чтобы узнать местоположение его координатора.
2. Регистрация производителя у координатора транзакций при помощи синхронного запроса InitPidRequest. При этом производитель указывает свой transactional.id.
   Данный запрос выполняет:
   - Возвращение одного и того же PID для одного transactional.id будущим экземплярам производителя.
   - Увеличение и возвращение эпохи PID, при этом любой предыдущий "зомби-экземпляр" производителя блокируется и не может продолжить свою транзакцию.
   - Удаление любой транзакции, оставшейся не завершенной предыдущим экземпляром производителя.
   - Создание записи (init) в журнале __transaction_state для заданного transactional.id, если это первая транзакция для производителя.
3. Запуск транзакции при помощи вызова метода producer.beginTransaction. После запуска транзакции все записанные сообщения будут частью транзакции. В журнале __transaction_state для этого transactional.id меняется состояние транзакции на ongoing.

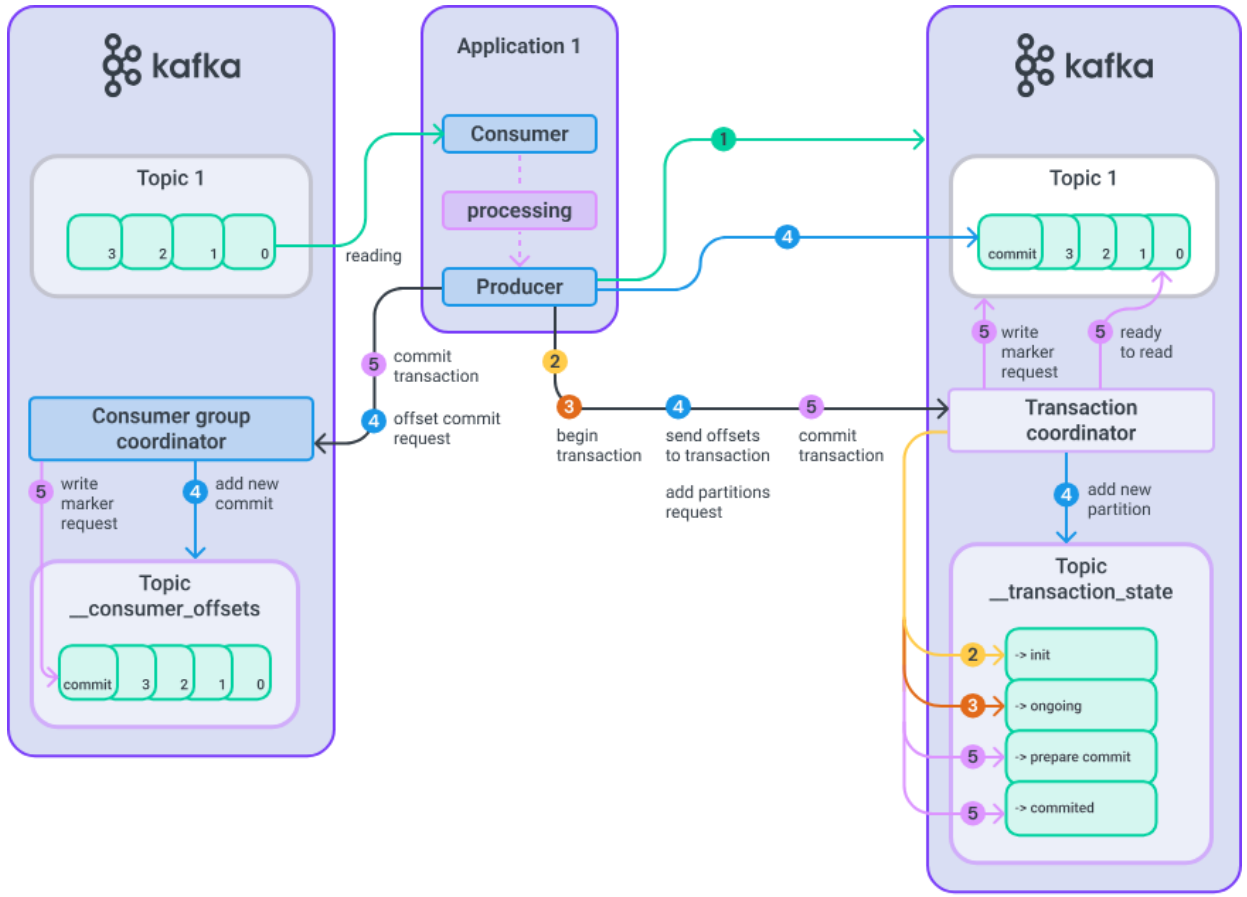


Гарантии надежности Kafka: Exactly once. Транзакции

- Exactly once — каждое сообщение будет обработано один и только один раз.

Запись транзакций (продолжение):

4. Цикл чтение → преобразование → запись сообщений:
   - Регистрация новой партиции в рамках транзакции при помощи запроса AddPartitionsToTxnRequest.
   - Запись сообщений (включая PID, эпоху и порядковый номер) в партицию пользователя при помощи метода producer.send.
   - Отправка смещений прочитанных сообщений в рамках транзакции с идентификатором groupId координатору транзакции при помощи метода producer.sendOffsetsToTransaction для регистрации добавления этой партиции в журнал транзакций.
   - Регистрация смещений в топике __consumer_offsets при помощи запроса TxnOffsetCommitRequest производителя координатору потребителей (включая PID и эпоху). Запись смещений в топике __consumer_offsets является частью транзакции. Координатор потребителей проверяет, что производитель не является "зомби-экземпляром". Записанные смещения не видны потребителям до подтверждения транзакции.
5. Подтверждение транзакции при помощи метода commitTransaction, который выполняет:
   - Запись сообщения prepare_commit в журнал транзакций.
   - Запись маркеров COMMIT в топики пользователя, в том числе и в топик __consumer_offsets.
   - Запись сообщения committed в журнал транзакций.
   - Открытие доступа для потребителей к сообщениям, записанным в топик.



Гарантии надежности Kafka: Exactly once. Транзакции

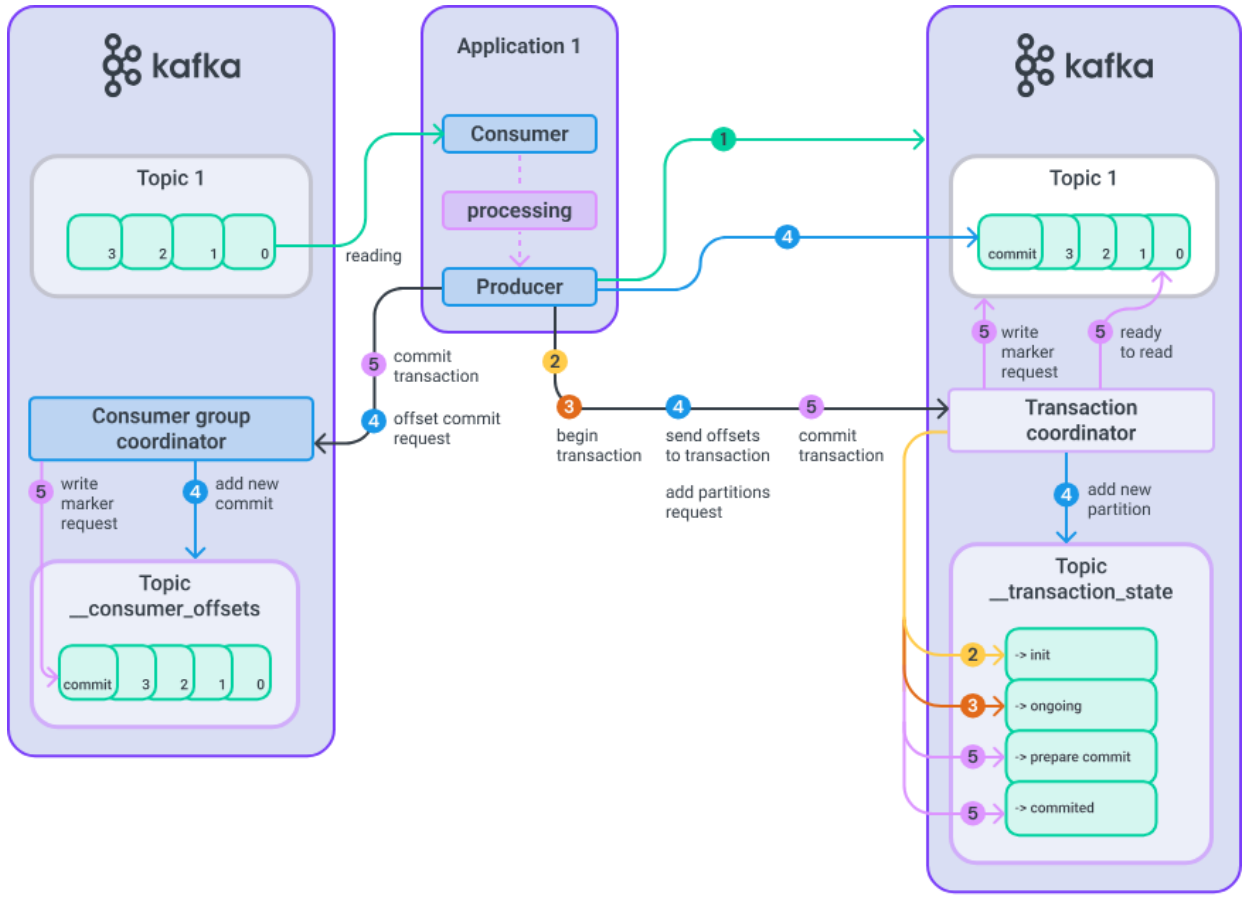
- Exactly once — каждое сообщение будет обработано один и только один раз.

Чтение транзакционных сообщений:

Чтение транзакционных сообщений потребителем регулируется параметром isolation.level:

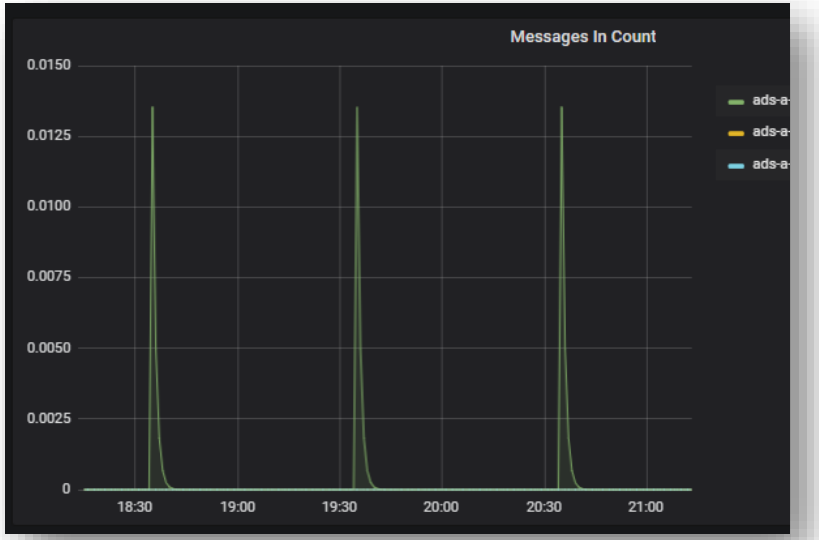
- read_uncommitted: (по умолчанию). Потребители могут читать все сообщения, включая те, которые являются частью незавершенных транзакций, что может привести к чтению "грязных" данных.
- read_committed: Потребители не будут видеть сообщения из незавершенных транзакций.

- Если параметру isolation.level присвоено значение read_committed, потребитель буферизует сообщения, имеющие PID, до тех пор, пока не прочитает сообщение с маркером COMMIT.
- После этого сообщения будут доставлены.
- Сообщения считываются только до последнего стабильного смещения, которое меньше смещения первой открытой транзакции.
- Потребители read_committed не смогут читать топики полностью, пока есть незавершенные транзакции.
- Если параметру isolation.level присвоено значение read_uncommitted, все сообщения видны потребителям, даже если они были частью прерванной транзакции.

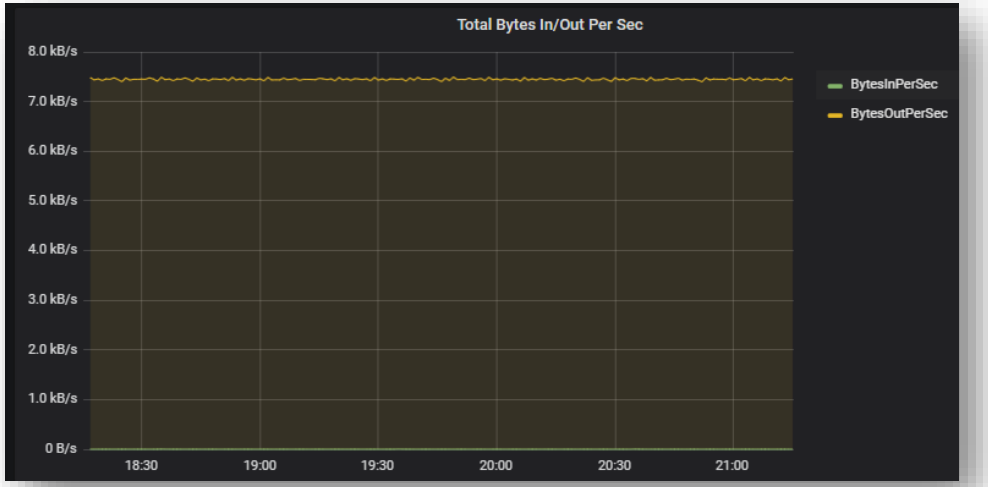


Мониторинг Kafka метрик JMX

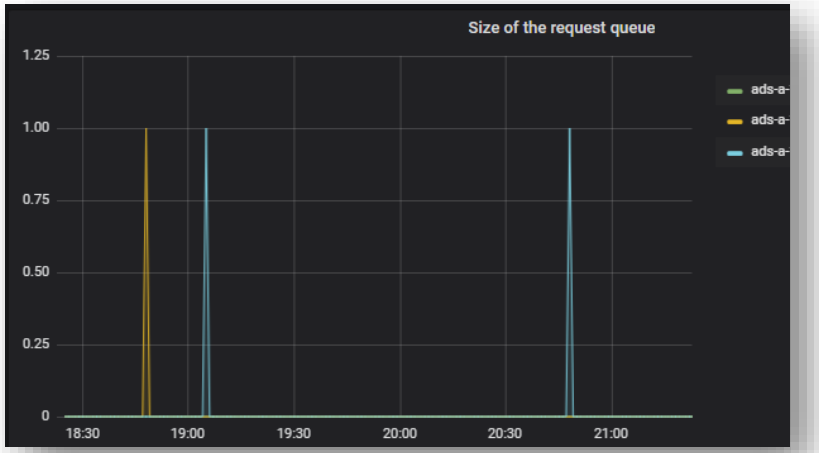
- **kafka.server:type=BrokerTopicMetrics,name=MessagesInPerSec:** показывает скорость, с которой сообщения поступают на брокер. Слишком низкая или высокая скорость может указывать на проблемы.



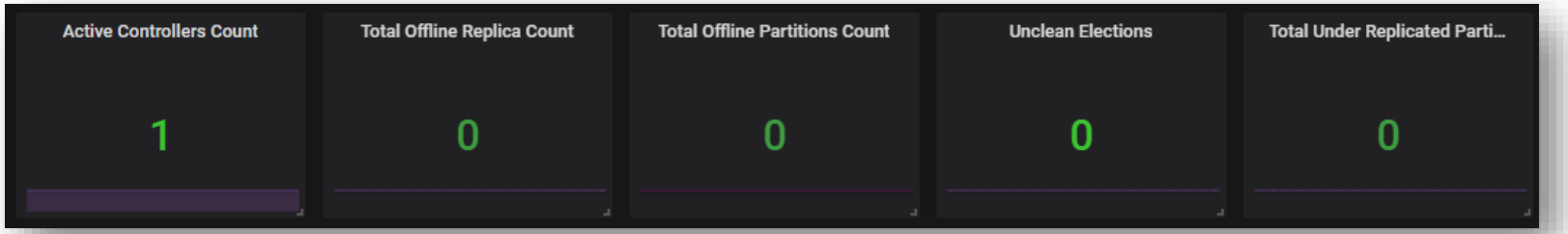
- **kafka.server:type=BrokerTopicMetrics,name=BytesInPerSec:** измеряет объем данных, поступающих на брокер. Аномальные значения могут сигнализировать о проблемах с сетью или приложением.
- **kafka.server:type=BrokerTopicMetrics,name=BytesOutPerSec:** отражает объем данных, отправляемых брокером потребителям. Резкие изменения могут указывать на изменения в паттернах потребления.



Мониторинг Kafka метрик JMX

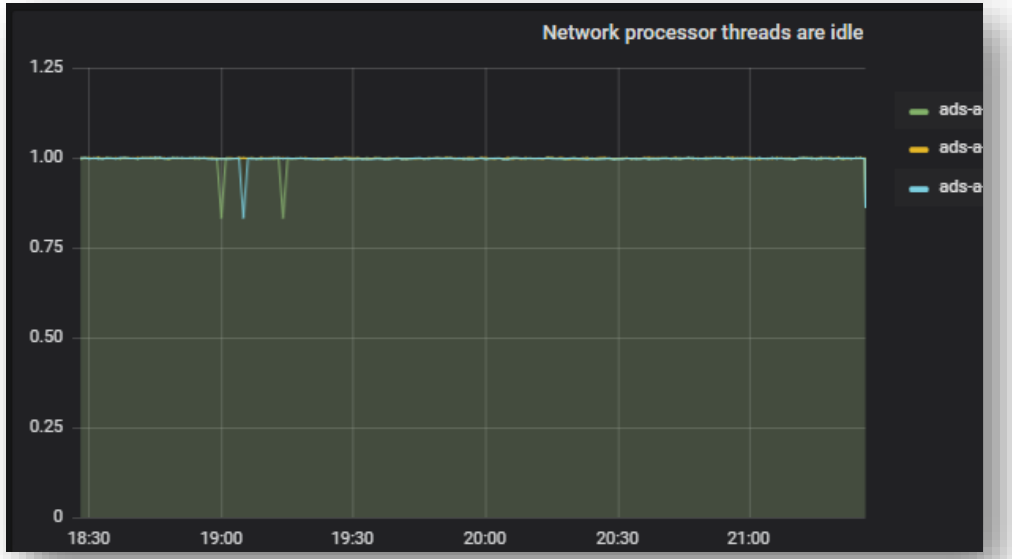


- **kafka.network:type=RequestMetrics,name=RequestQueueSize:** отражает размер очереди запросов на брокере. Эта метрика показывает количество запросов, ожидающих обработки. Если размер очереди увеличивается и не уменьшается со временем, это может указывать на проблемы с производительностью, такие как недостаточная пропускная способность брокера или задержки в обработке сообщений.
- **kafka.controller:type=KafkaController,name=ActiveControllerCount:** количество активных контроллеров в кластере. Должен быть только один с активным управлением!
- **kafka.server:type=ReplicaManager,name=UnderReplicatedPartitions:** количество недостаточно реплицированных партиций. Ненулевое значение может указывать на проблемы с репликацией.



Мониторинг Kafka метрик JMX

- **kafka.network:type=SocketServer,name=NetworkProcessorAvgIdlePercent**. Эта метрика показывает процент свободных сетевых потоков, которые выполняют обработку. Все запросы Kafka проходят через сетевые потоки, так что это важная метрика. 0 — все ресурсы недоступны, а 1 — все ресурсы доступны. Нормальным будет значение выше 30%, чтобы ресурсы кластера не находились на пределе постоянно.



Лабораторная работа

- Создать топик с репликацией 1.
- Увеличить репликацию до 3 (*kafka-reassign-partitions.sh*).
- Изменить параметры топика (cleanup.policy, min.insync.replicas)