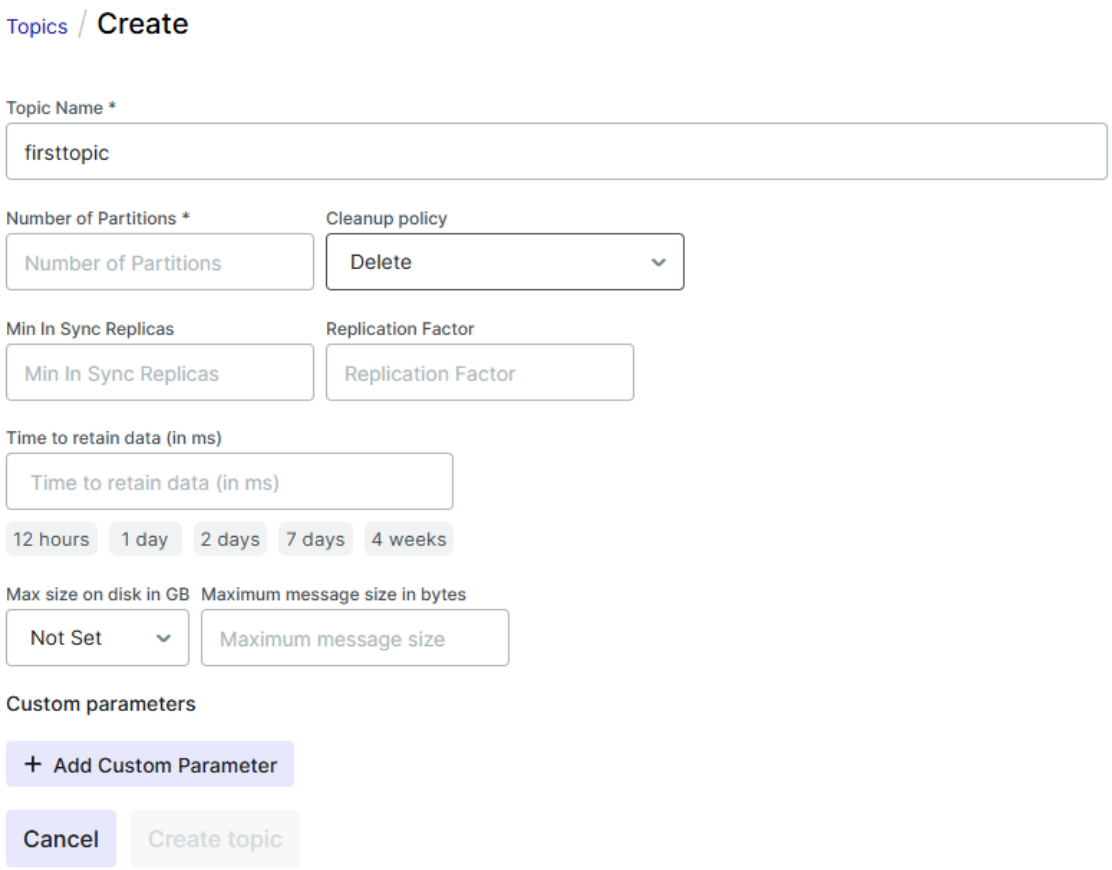
Topics

Создание темы (topic)



Поле	Тип данных	Описание
Topic Name	Строка	Название темы.

Поле	Тип данных	Описание
Number of partitions	Целое число	Количество разделов — управление производительностью и масштабируемостью кластера. Единицы параллелизма в Kafka. Разделы определяют скорость и масштабирование. • Для производителей (Producers): несколько разделов позволяет отправлять данные параллельно. Больше разделов — выше общая пропускная способность на запись. • Для потребителей (Consumers): ключевой момент. В одной группе потребителей (consumer group) каждый раздел может обрабатываться только одним потребителем. Количество разделов определяет максимальное число потребителей в группе, которые могут работать параллельно. **Пример**: для 10 разделов можно запустить до 10 потребителей, и каждый будет обрабатывать свой раздел. Если потребителей больше, чем разделов, то лишние потребители будут простаивать и тратить ресурсы впустую . **Дополнительные факторы** • Гарантируется порядок сообщений только внутри одного раздела. Если критичен порядок событий (например, все действия одного пользователя должны обрабатываться последовательно), используется ключ сообщения (key). Все сообщения с одним ключом всегда попадают в один и тот же раздел, сохраняя порядок. • Каждый раздел это доп. нагрузка (больше файлов, больше метаданных, больше времени на выборы лидера). Слишком большое количество

Поле	Тип данных	Описание
Cleanup policy	Список. • Delete • Compact • Compact, Delete	Политика управления сроком жизни данных. По умолчанию delete. **Политика delete (Удаление)** Удаление старых данных по истечении заданного времени или при превышении лимита размера. **Как это работает**: удаление когда старше retention.ms (по умолчанию 7 дней), или когда общий размер раздела превышает retention.bytes. Происходит в фоновом режиме с заданным интервалом. **Когда применять**: потоки событий и логов, где важна вся история. Например, логи приложений, метрики, трекинговые события. Здесь каждое сообщение самоценно, и старые данные со временем теряют актуальность . **Политика compact (Сжатие)** Эта политика работает по принципу "сохранить только последнее известное состояние" для каждого ключа сообщения . Как это работает: Вместо удаления по времени, Kafka в фоновом режиме "сжимает" лог. Для каждого уникального ключа (key) в разделе она оставляет только последнее по времени сообщение, а все промежуточные удаляет . Важное условие: Сообщения в таких топиках обязаны иметь ключ (key). Сообщения без ключа будут удалены при сжатии . Когда применять: Это идеальный инструмент для топиков, которые хранят состояние или конфигурации. Например: Таблица с актуальными данными пользователей (адрес, почта) . Информация о текущем статусе заказа. Список доступных товаров на складе. Внутренний топик Kafka __consumer_offsets, где хранятся текущие смещения потребителей, по умолчанию использует именно эту политику . ◫ Комбинированная политика delete,compact Вы можете применить обе политики одновременно, указав их через запятую: cleanup.policy=delete,compact . В этом случае Kafka сначала

Поле	Тип данных	Описание

Revision #1
Created 8 September 2026 14:43:18 by Admin
Updated 8 September 2026 15:42:45 by Admin