Exactly-once в микросервисной среде

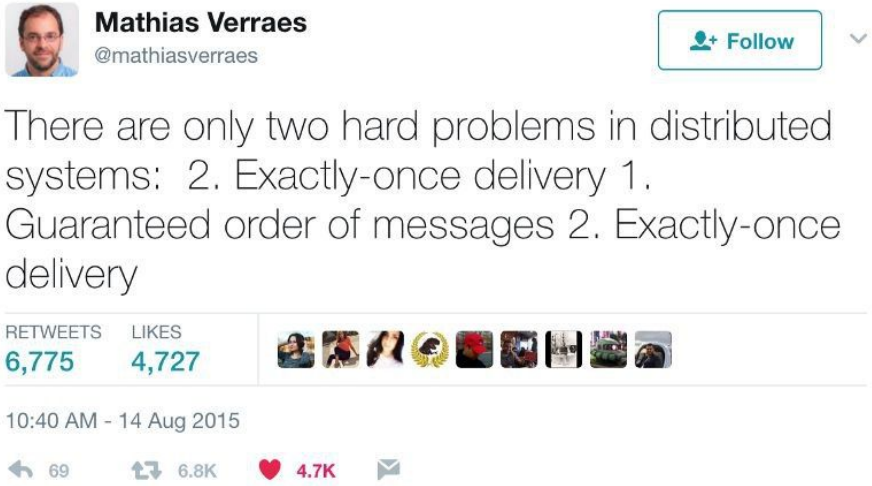
О себе

- 10 лет опыта на .NET
- Работаю в компании Mindbox
- Занимаюсь высоко-нагруженными сервисами

Что будет в докладе

1. Чуть-чуть бизнес
2. Идемпотентность
3. Transactional outbox
4. Составление ключа идемпотентности на кейсах
5. Эволюция

Почему такая тема



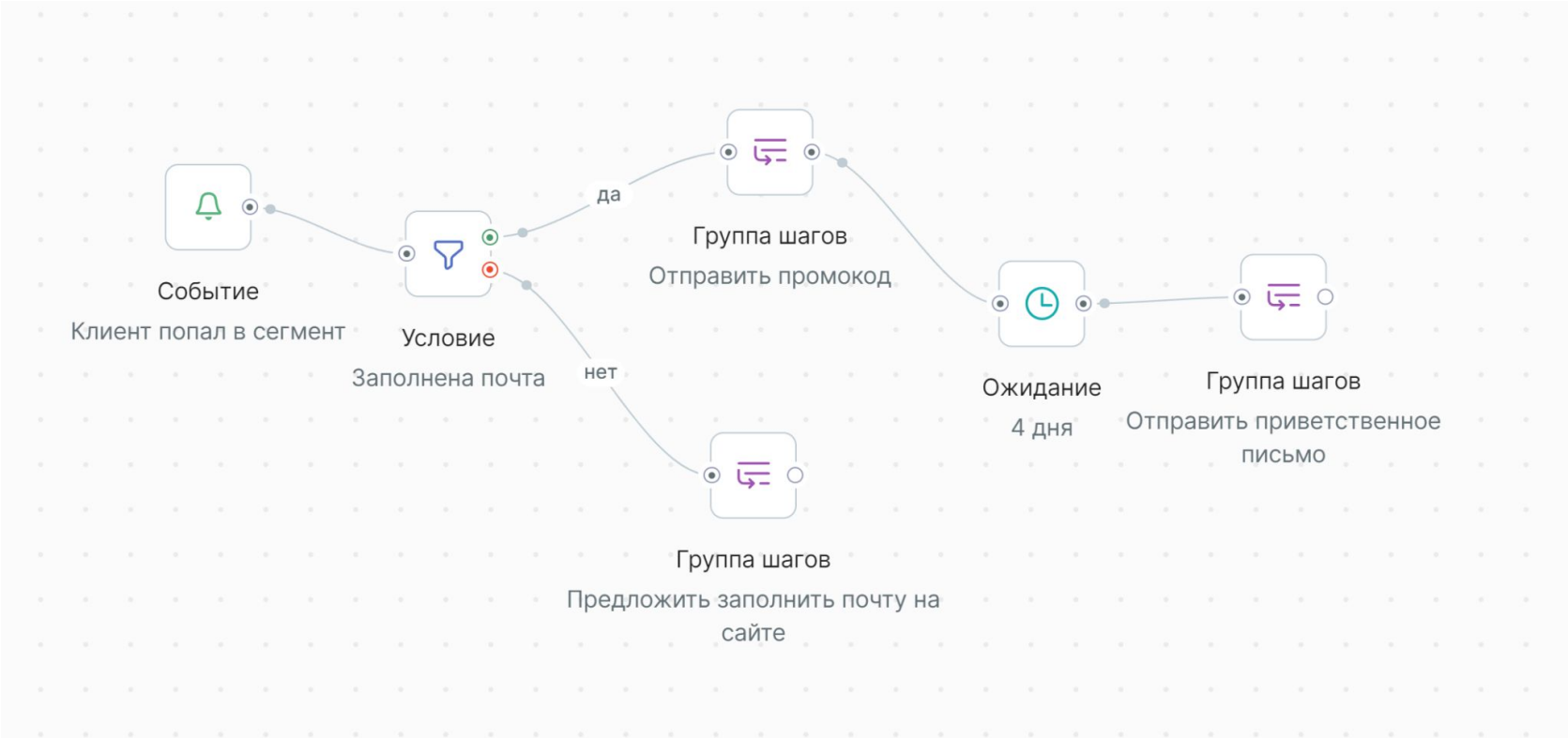
Всё-таки exactly once или at least once?

- не хотим распределенной транзакции
- идемпотентность решает проблемы

Какие ваши доказательства?

- Кейсы
- Кейсы
- Кейсы

Бизнес

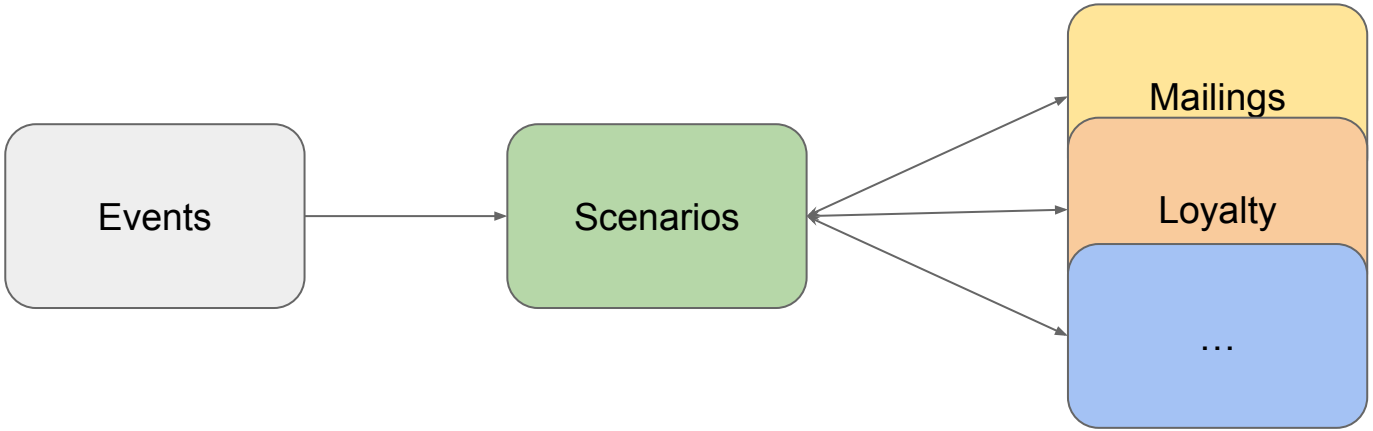

Событие
Клиент попал в сегмент
Условие
Заполнена почта
да
нет
Группа шагов
Отправить промокод
Группа шагов
Предложить заполнить почту на сайте
Ожидание
4 дня
Группа шагов
Отправить приветственное письмо

Микросервис

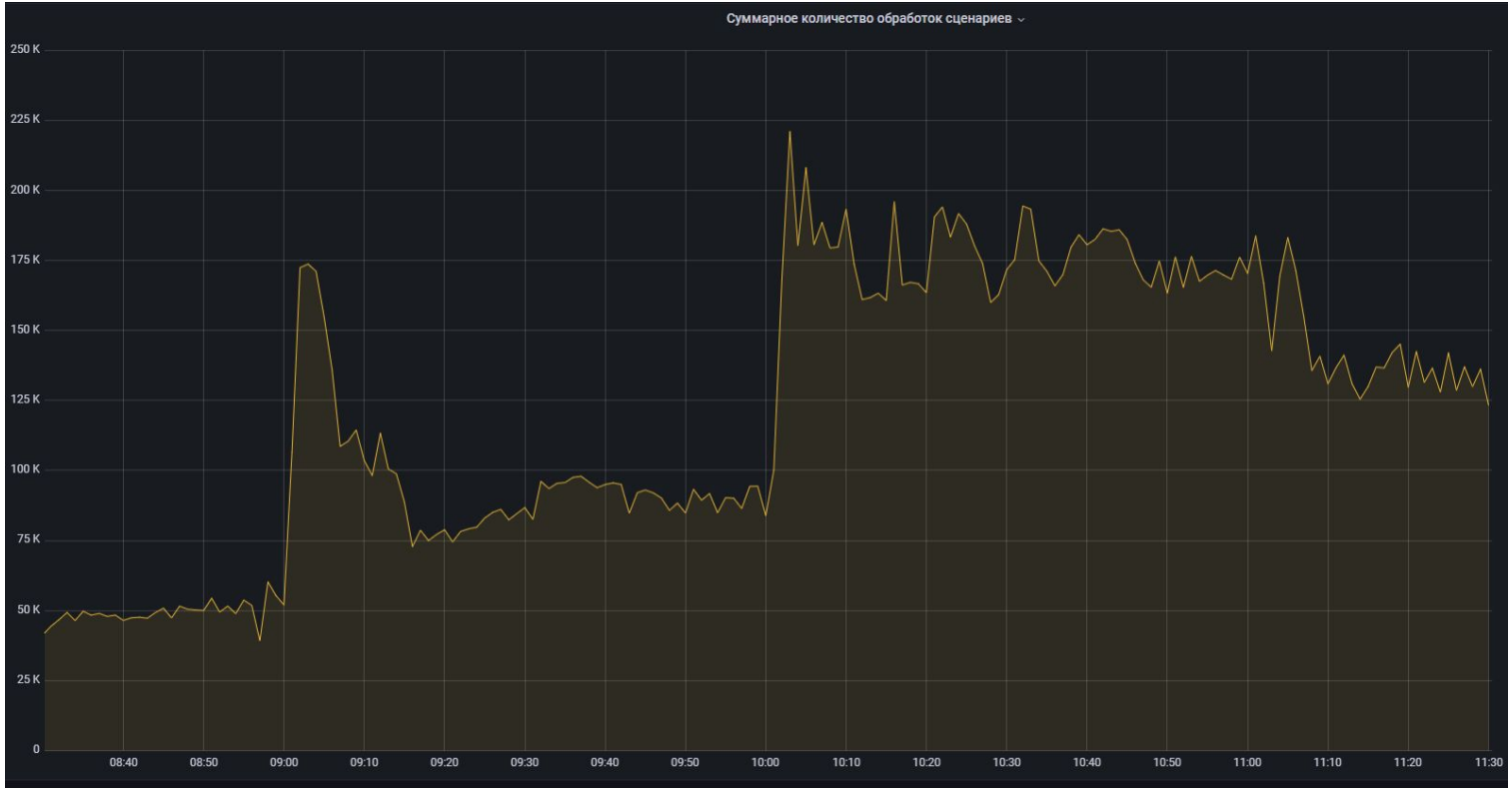
С монолитом будет тяжело

- Кросс-командая фича
- Много разработчиков
- Большая кодовая база

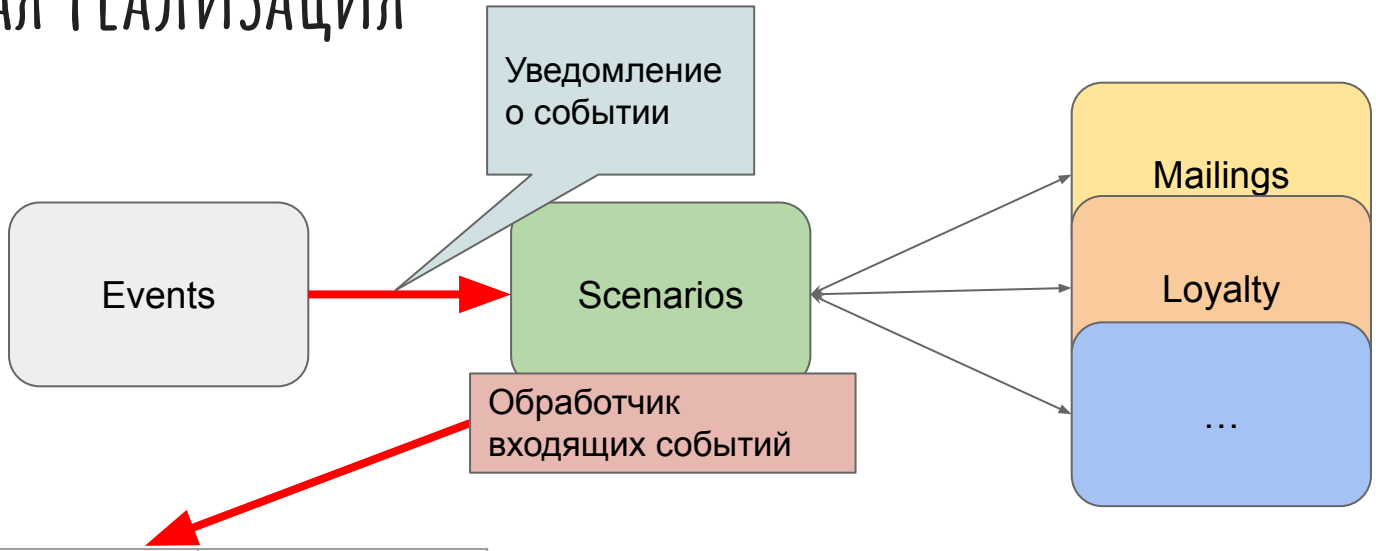
Какие сервисы в итоге получились



Достаточно высокие нагрузки

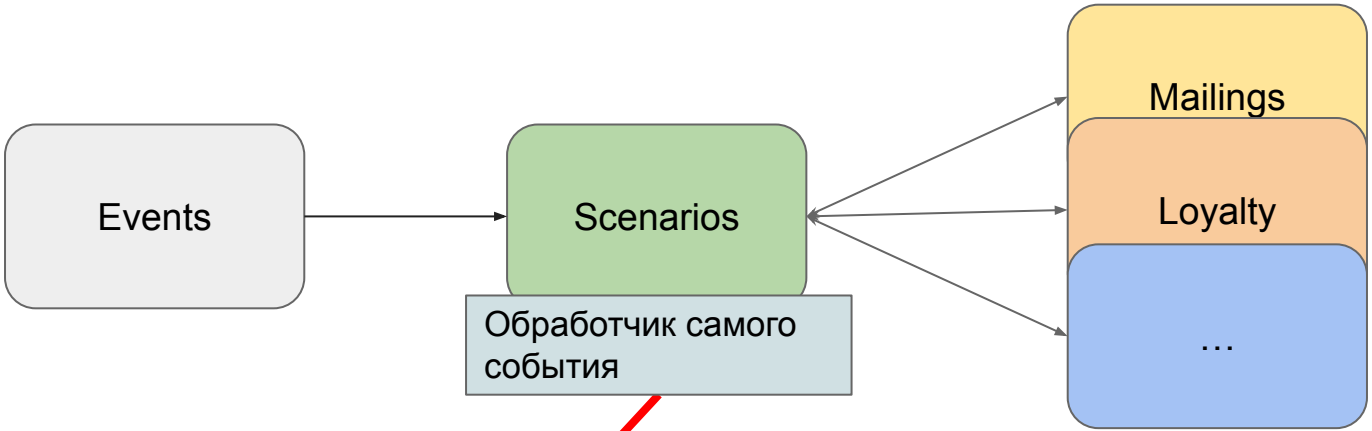


Наивная реализация



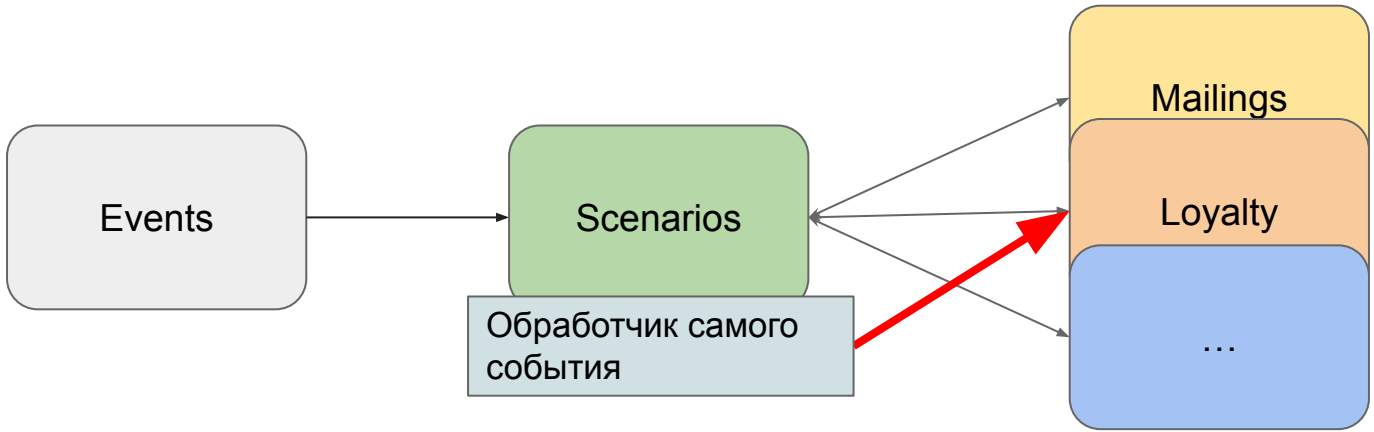
EventData	**IsProcessed**
…	false

Наивная реализация



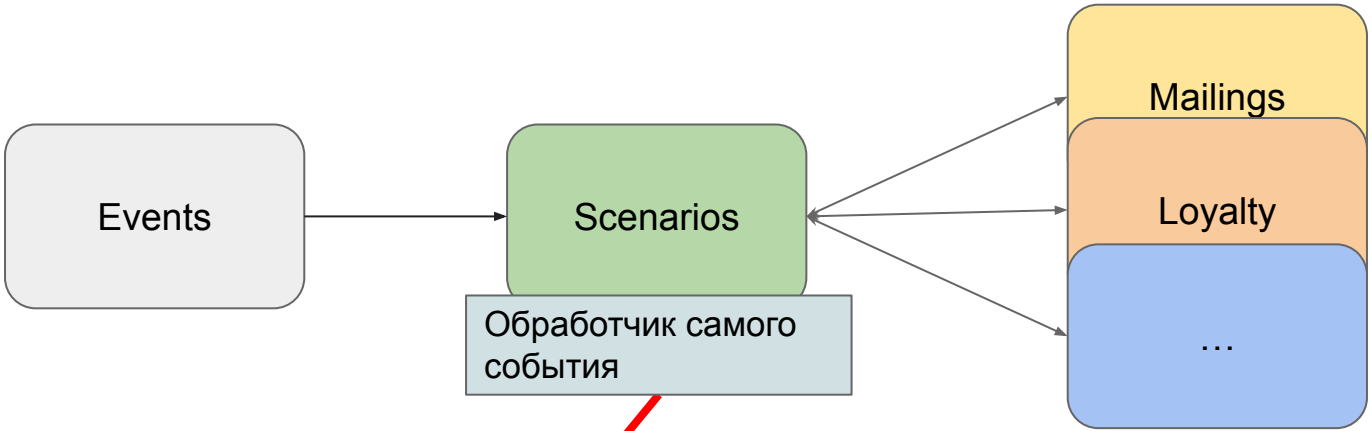
EventData	IsProcessed
…	true

Наивная реализация



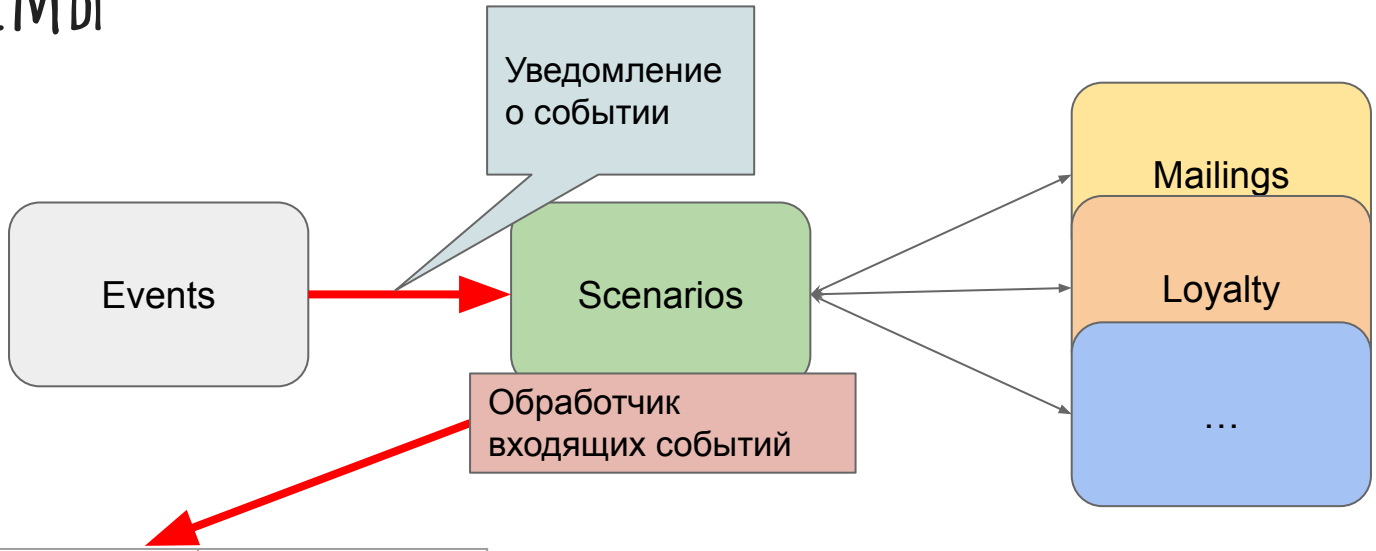
EventData	IsProcessed
…	true

Наивная реализация



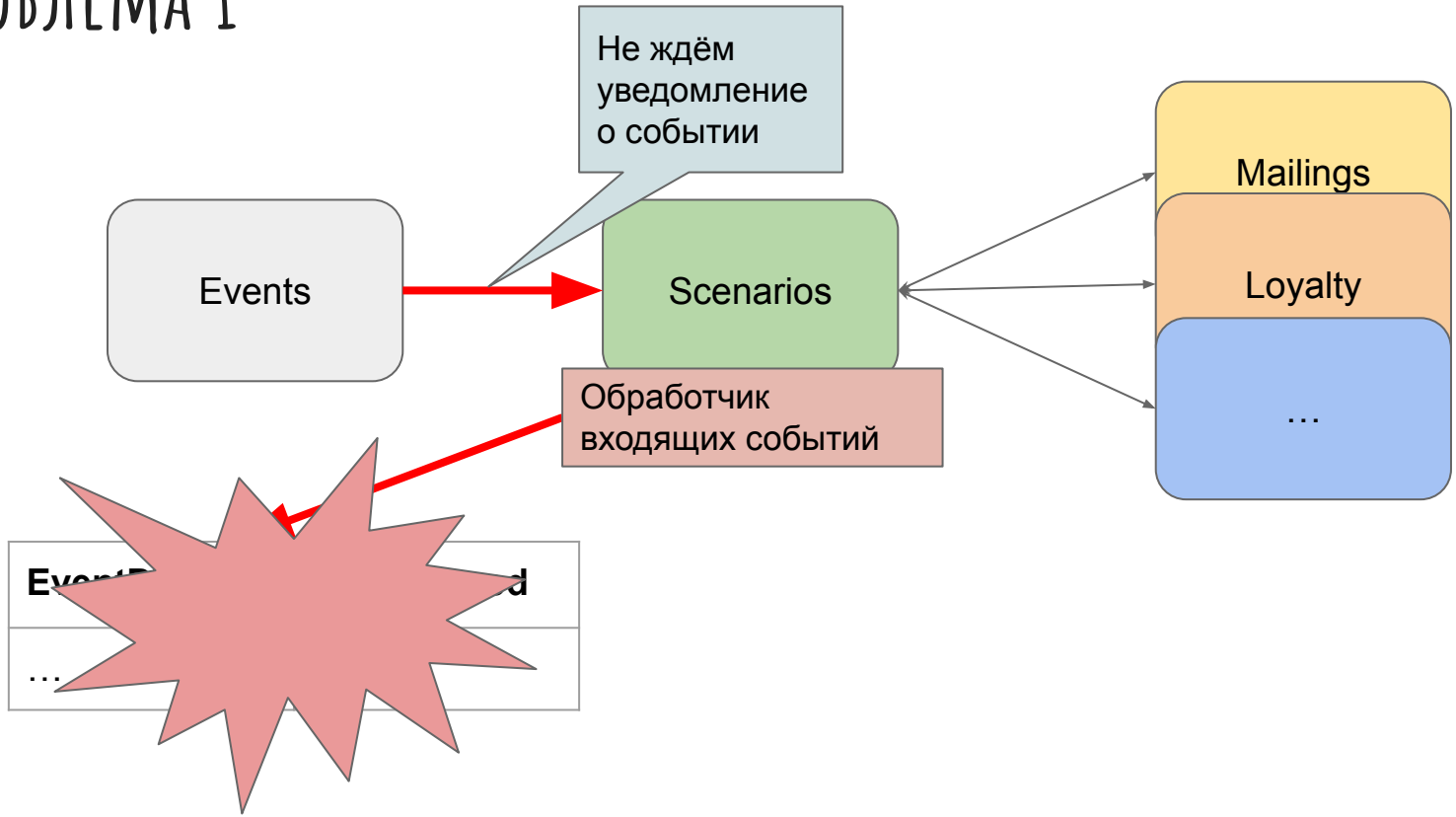
EventData	IsProcessed
…	true

Проблемы

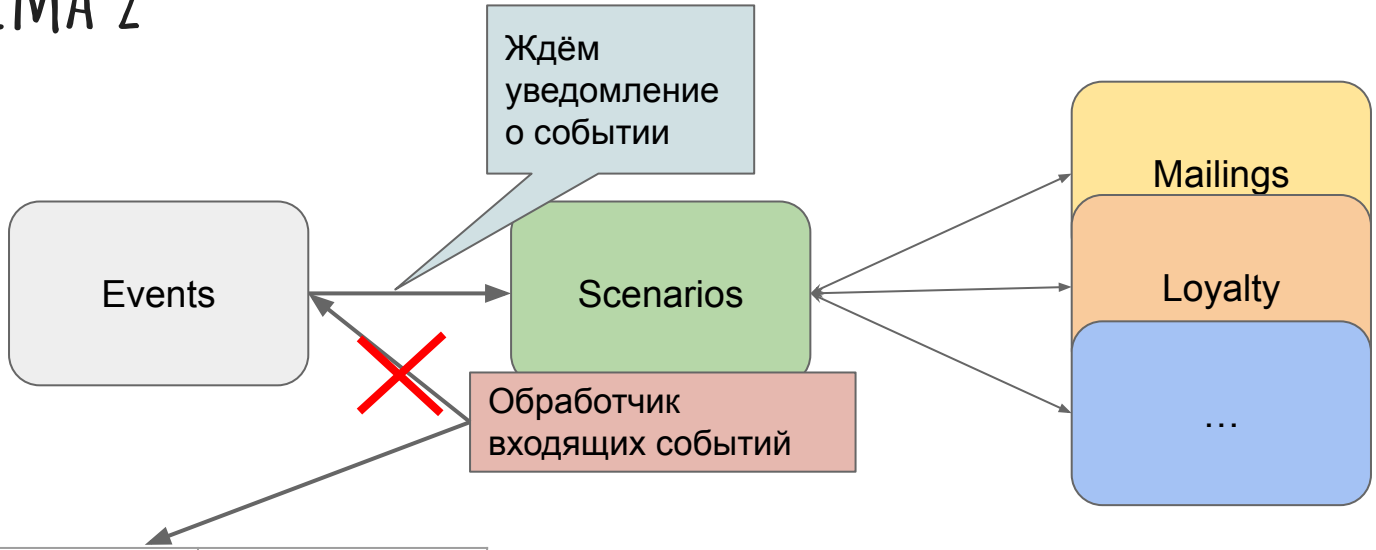


EventData	IsProcessed
…	false

Проблема 1

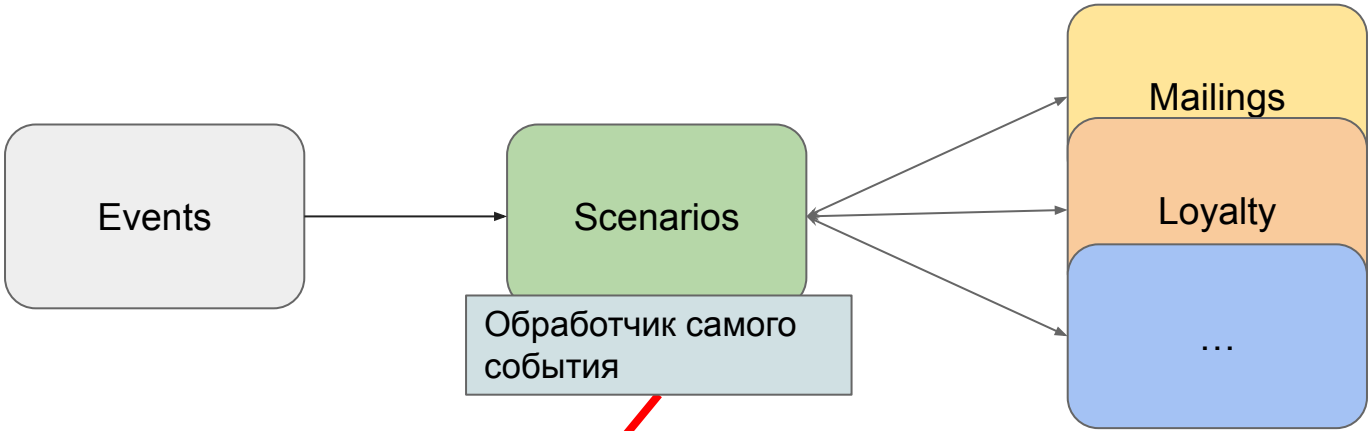

Не ждём уведомление о событии
Events
Scenarios
Mailings
Loyalty
…
Обработчик входящих событий
…

Проблема 2



EventData	IsProcessed
…	false

Проблема 3



EventData	IsProcessed
…	true

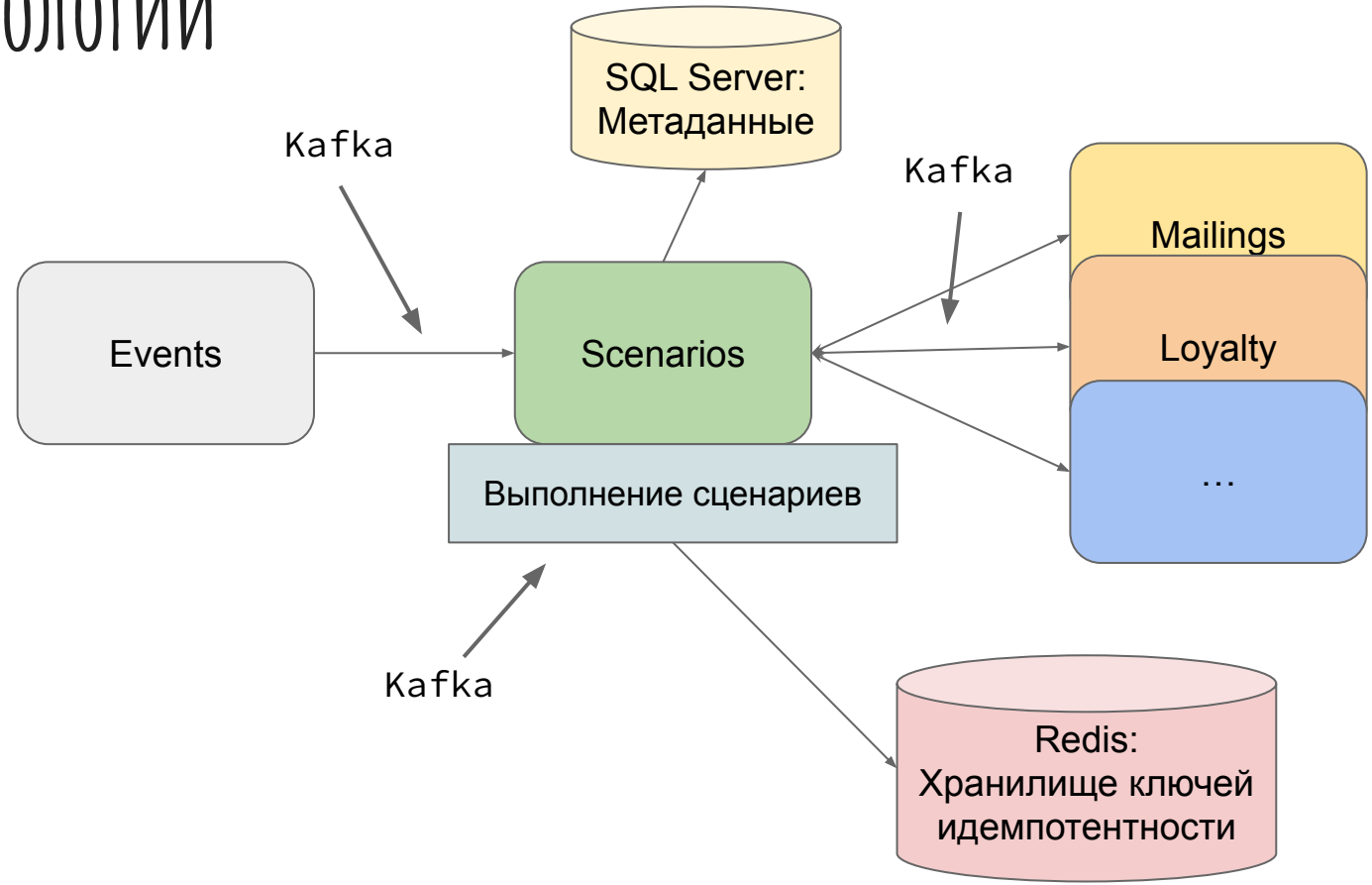
Проблема 4

Не соответствует требованиям по производительности, так как RDBMS не масштабируются горизонтально

Выводы

- Даже в наивной реализации приходится думать о exactly-once гарантиях
- Если требуется высокая производительность и масштабируемость, нужно использовать соответствующие технологии

Технологии


SQL Server:
Метаданные
Kafka
Kafka
Events
Scenarios
Mailings
Loyalty
…
Выполнение сценариев
Kafka
Redis:
Хранилище ключей
идемпотентности

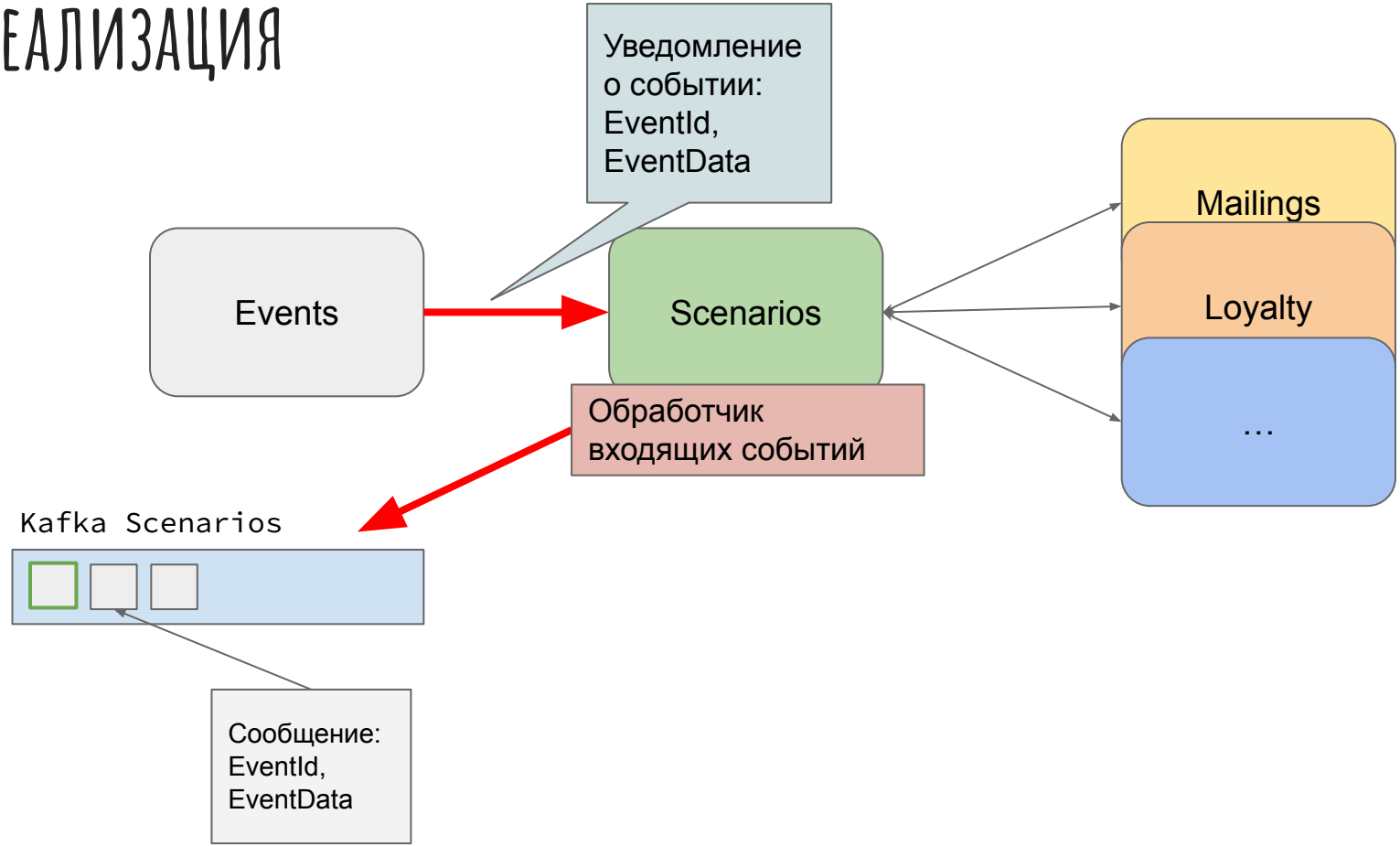
Почему кафка?

- Брокер сообщений, решает похожие с RabbitMQ задачи
- Persistent с репликацией: накопление миллионов необработанных сообщений в топике - не страшно
- Горизонтально масштабируется

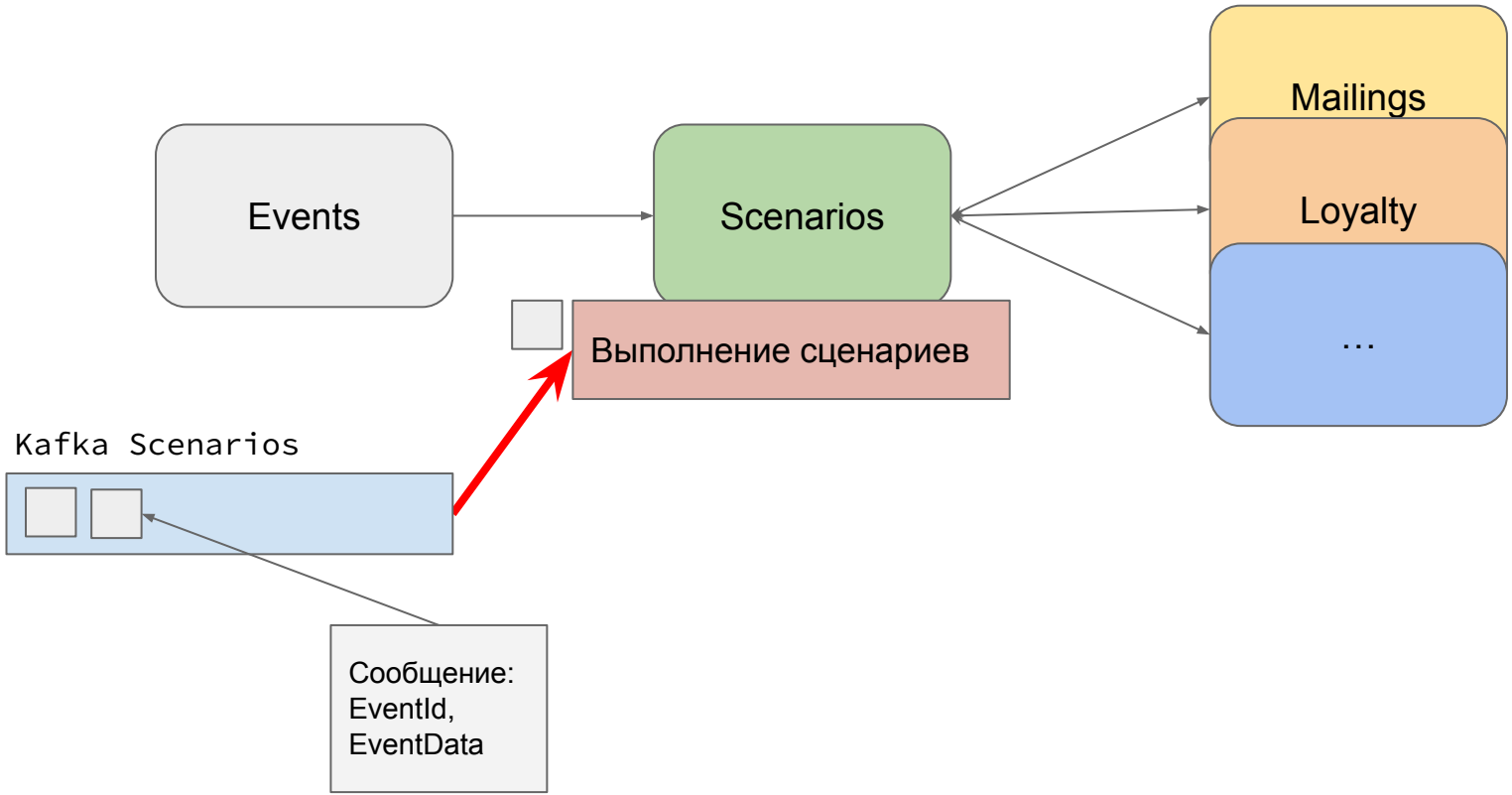
Что такое эта ваша идемпотентность?

Свойство операции, при повторении которой не будет происходить новых сайд-эффектов

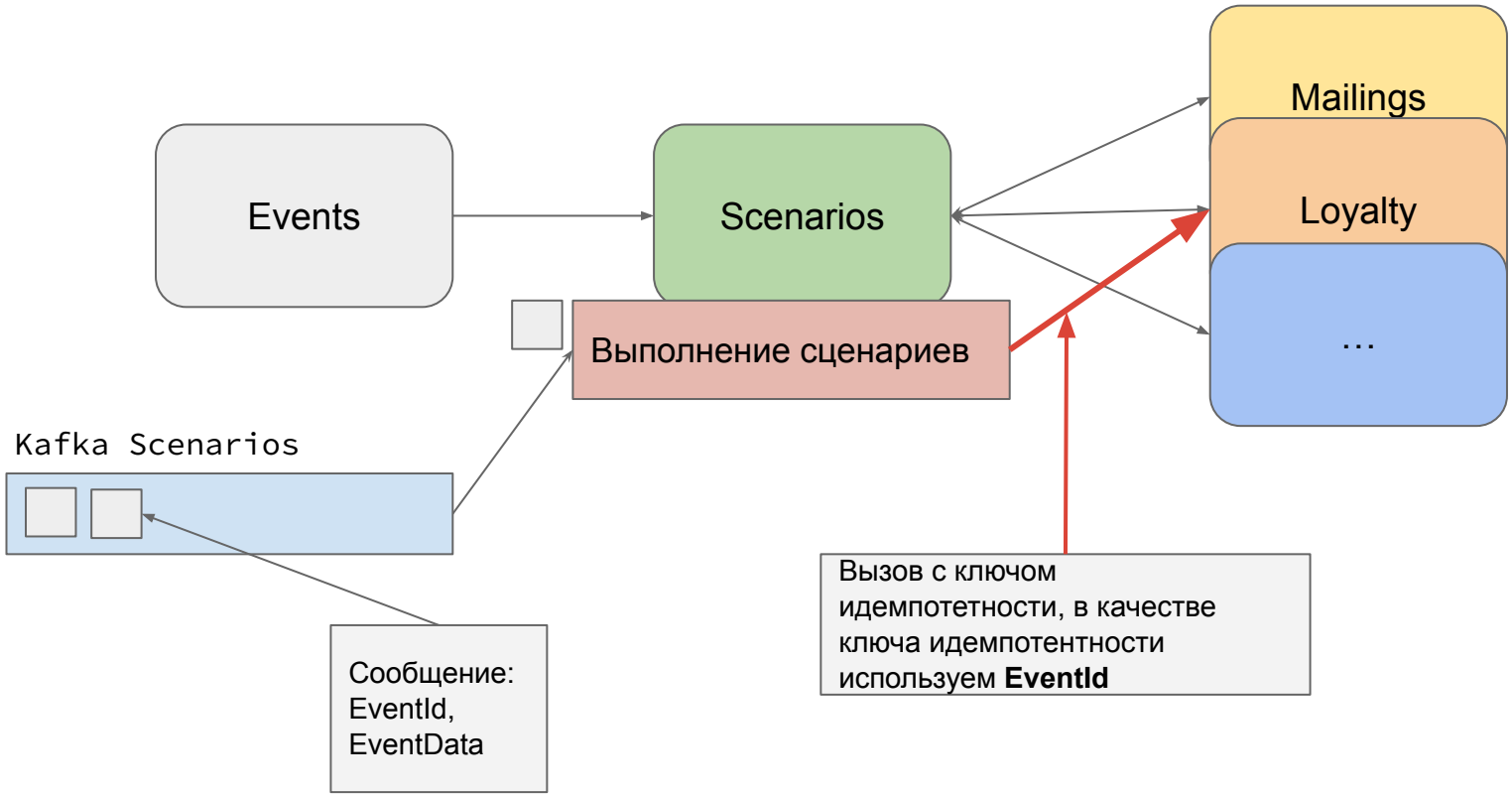
Реализация


Уведомление о событии: EventId, EventData
Events
Scenarios
Mailings
Loyalty
…
Обработчик входящих событий
Kafka Scenarios
Сообщение: EventId, EventData

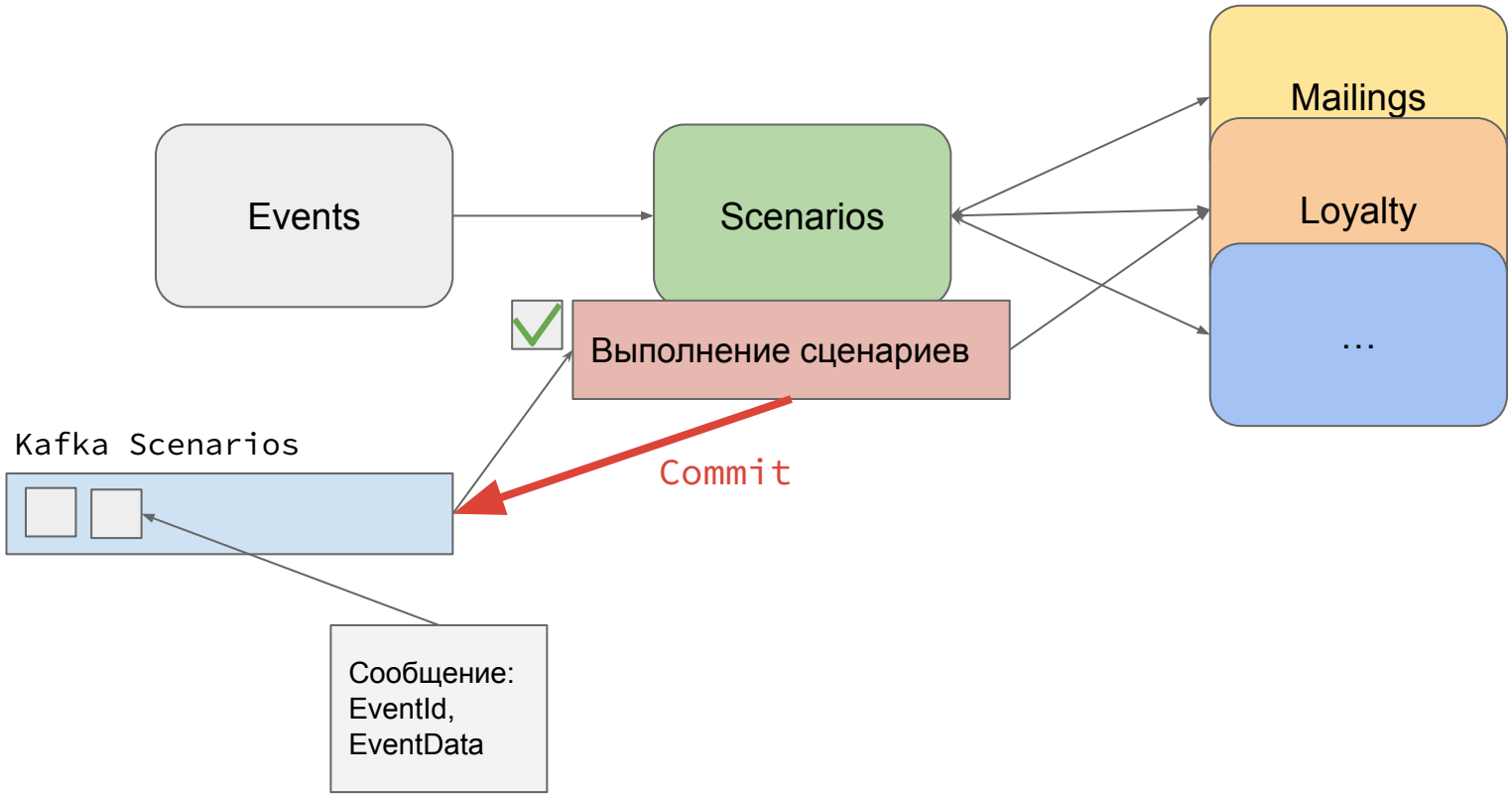
Реализация


Events
Scenarios
Mailings
Loyalty
…
Выполнение сценариев
Kafka Scenarios
Сообщение:
EventId,
EventData

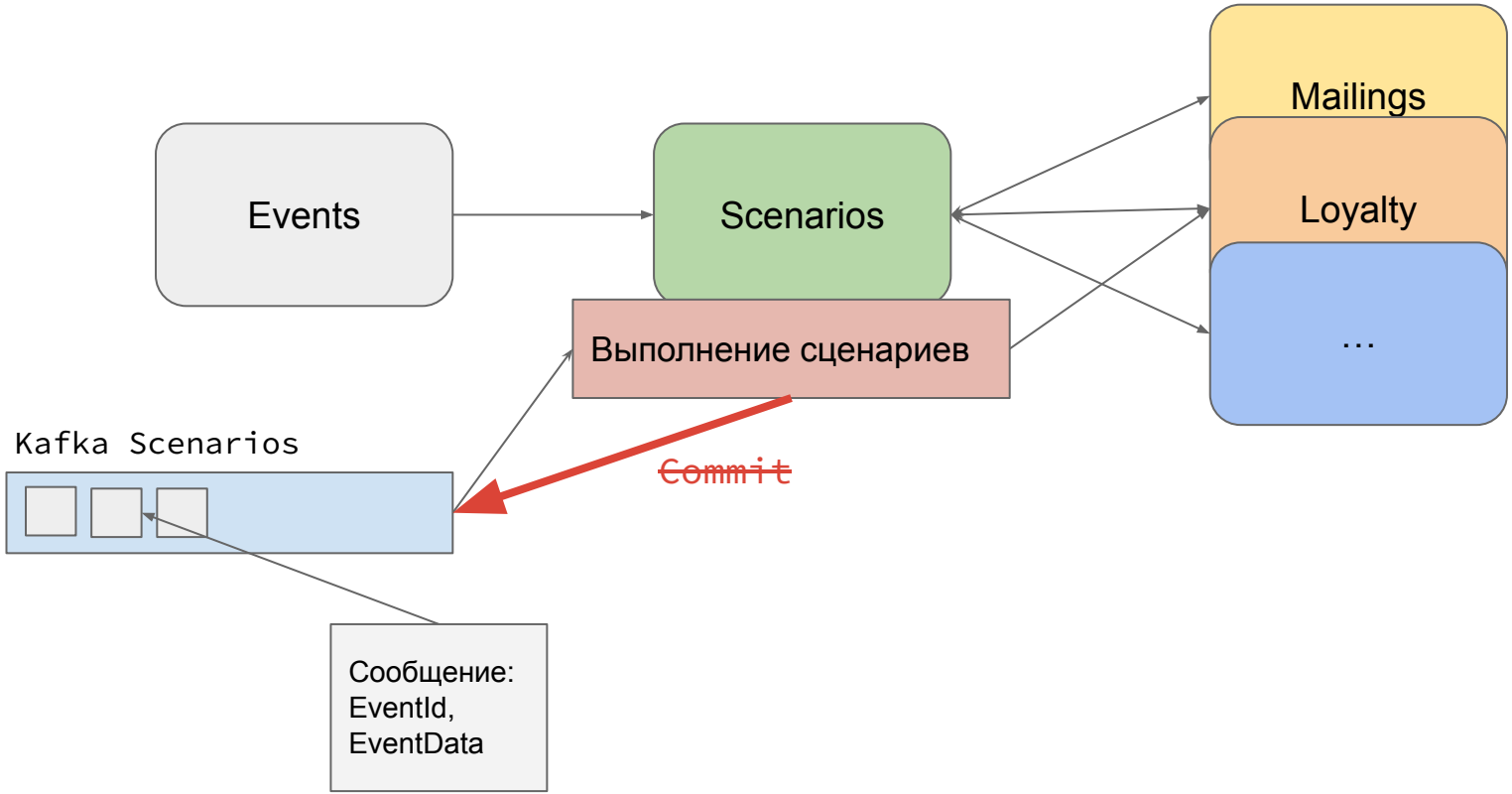
Реализация


Events
Scenarios
Mailings
Loyalty
…
Выполнение сценариев
Kafka Scenarios
Сообщение:
EventId,
EventData
Вызов с ключом идемпотетности, в качестве ключа идемпотентности используем **EventId**

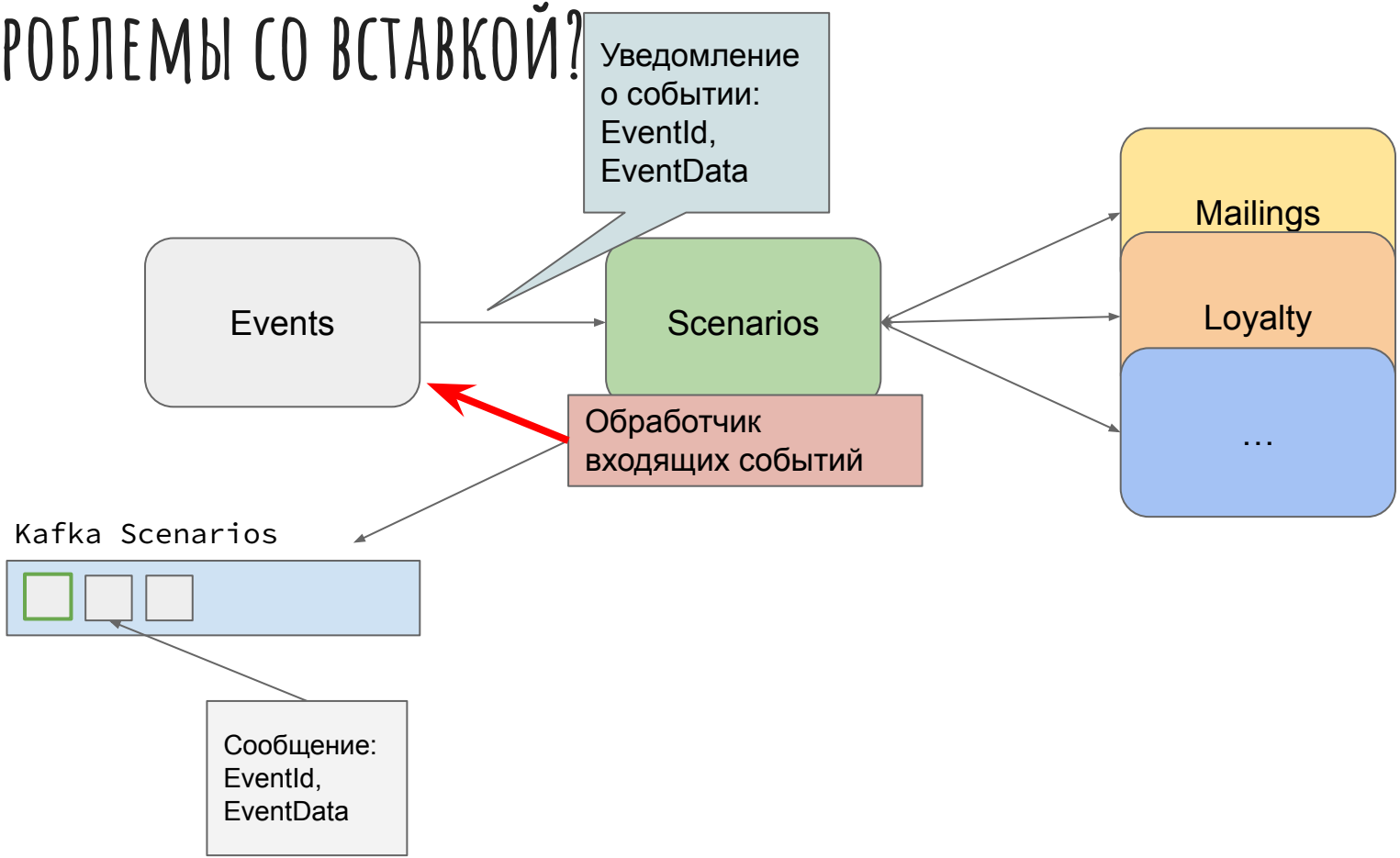
Реализация


Events
Scenarios
Mailings
Loyalty
…
Выполнение сценариев
Kafka Scenarios
Commit
Сообщение:
EventId,
EventData

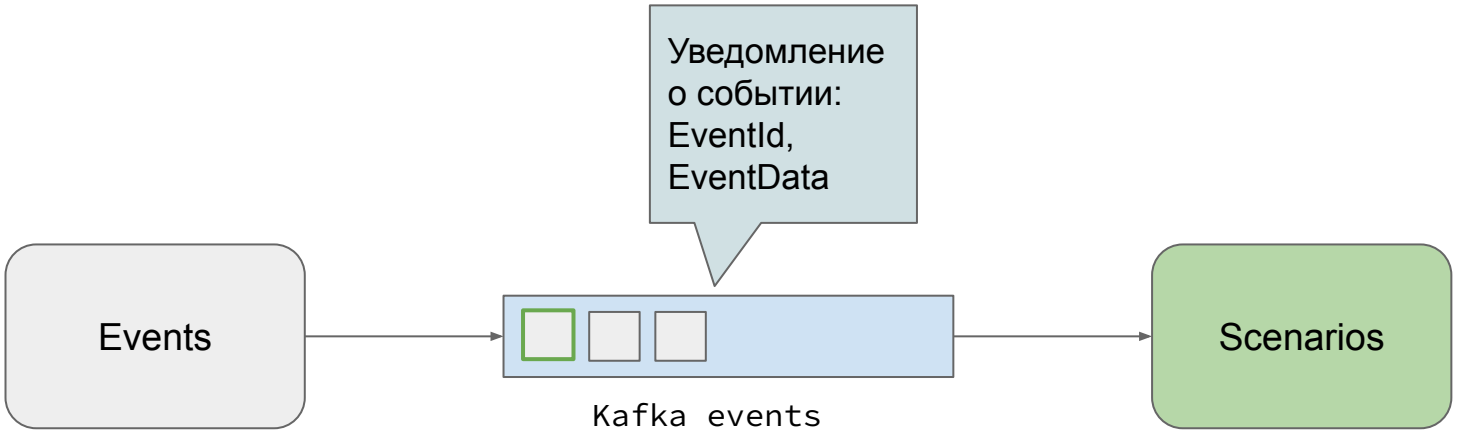
Проблемы с коммитом?



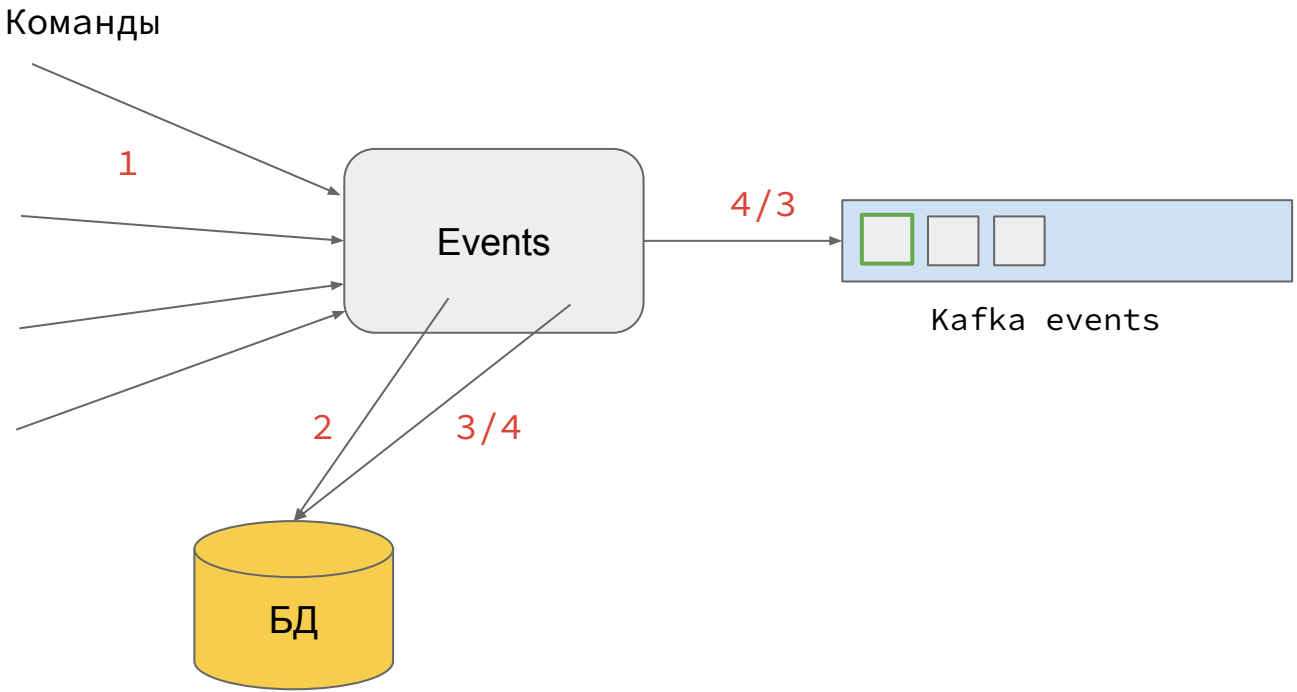
Проблемы со вставкой?



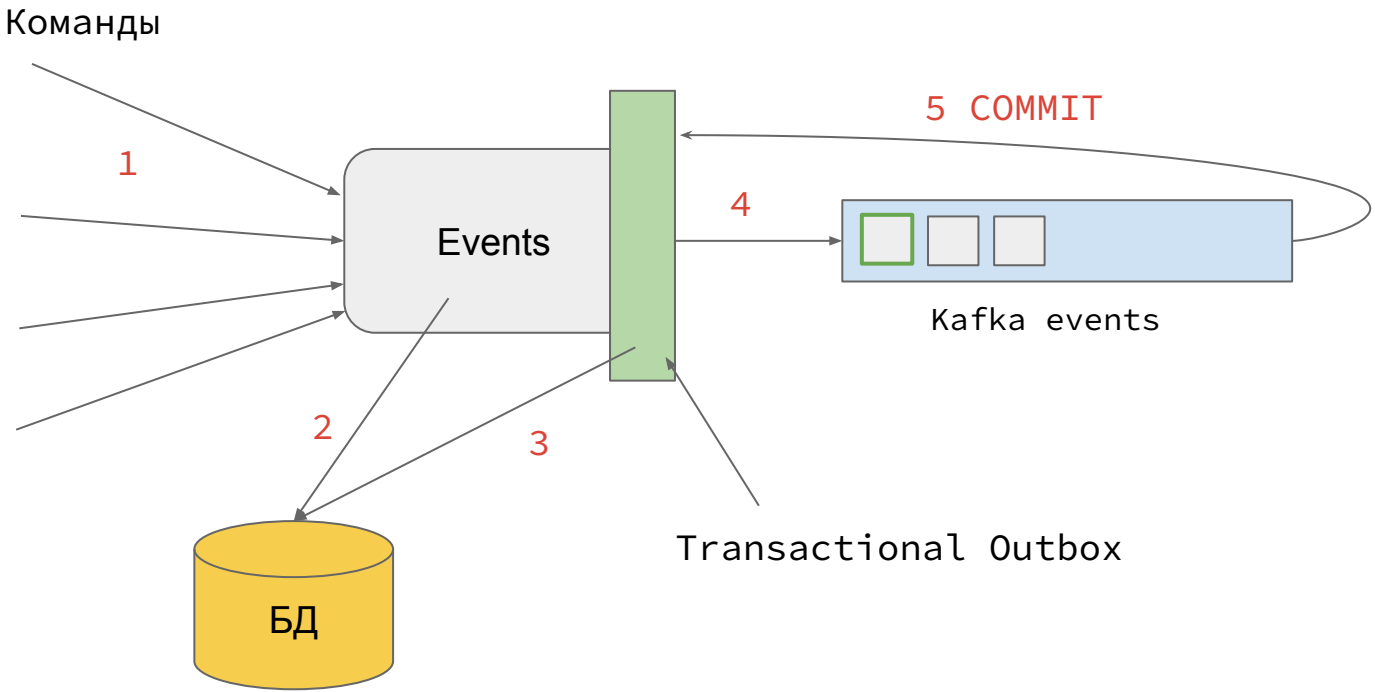
Откуда ещё могут быть дубли сообщений?



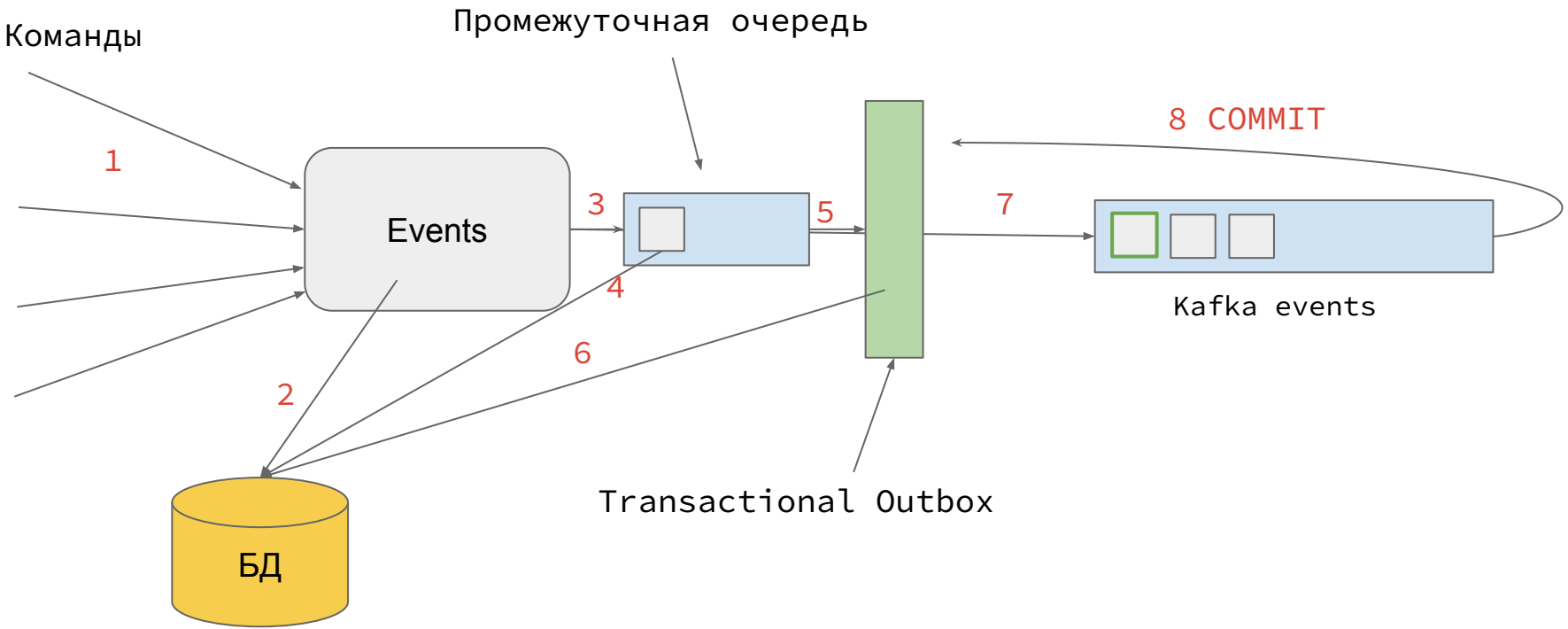
Как уведомляем о событиях?



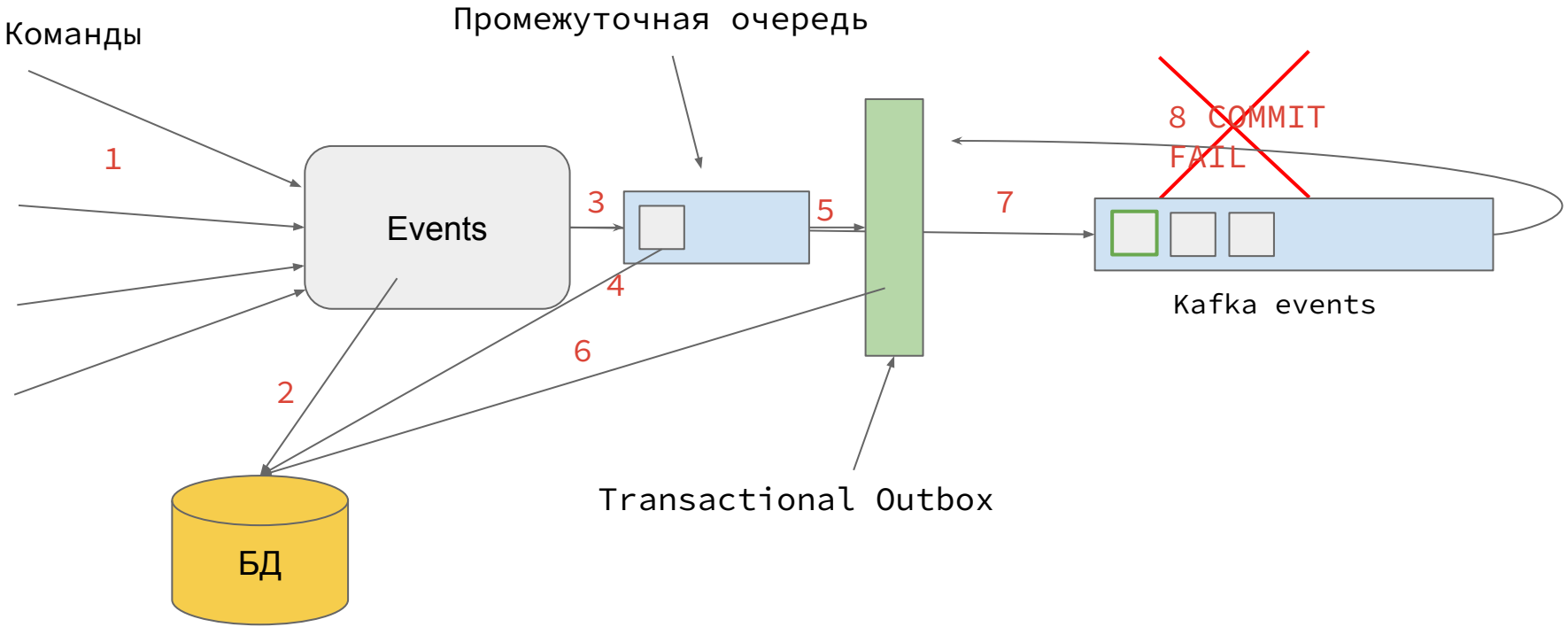
Как уведомляем о событиях?



Как уведомляем о событиях?



Где же тут дубли?



А как же Kafka Idempotent Producer?

Он может только уменьшить количество дублей

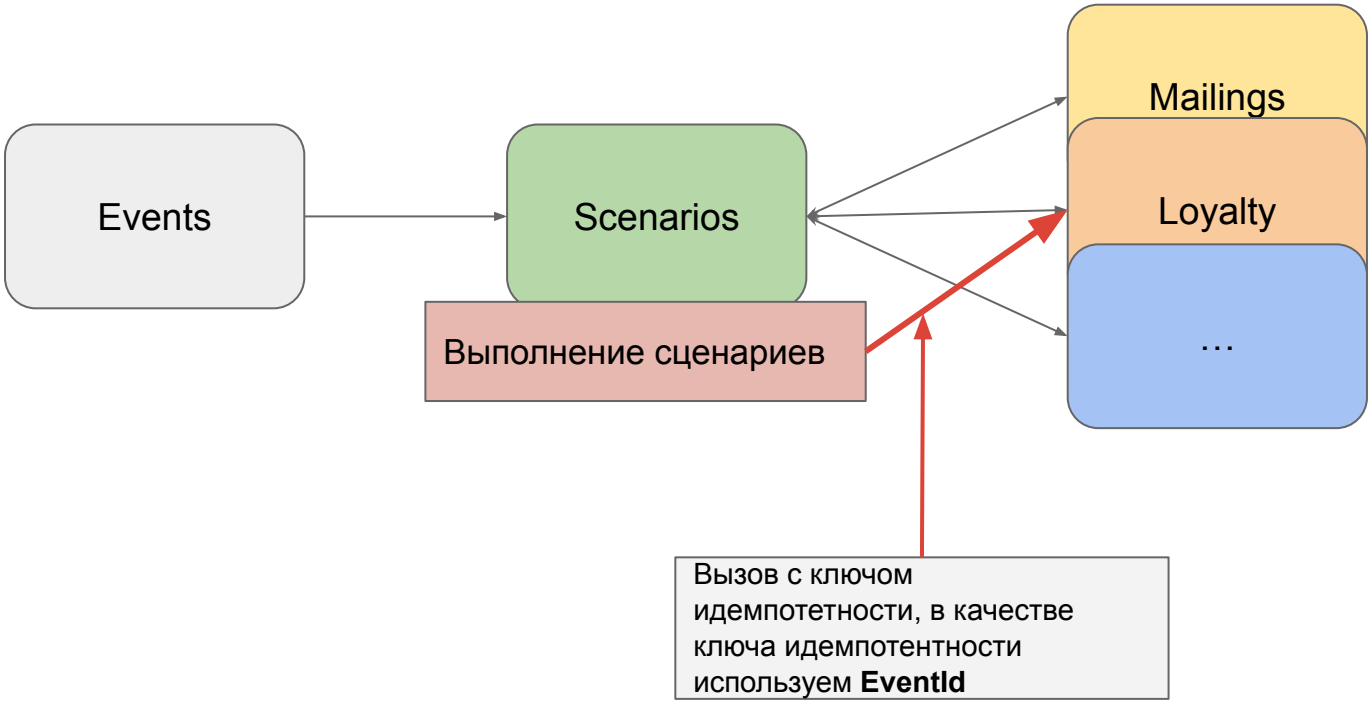
- Падения подов/ноды/сети
- Запрет на самостоятельные ретраи

  When enable.idempotence is set to true, no manual retries are required, in fact performing retries in your application code will still cause duplicates.

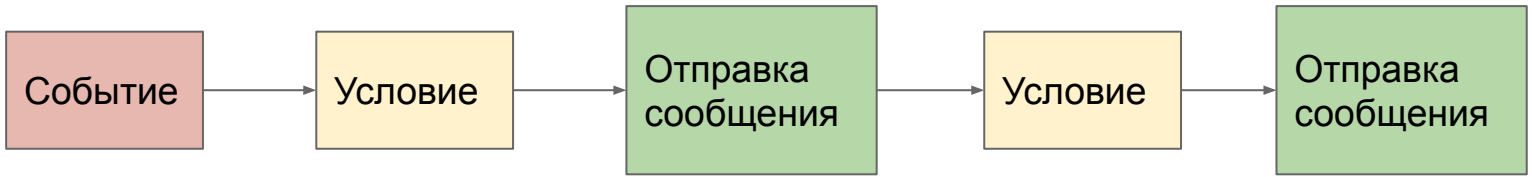
Ещё пара слов о Transactional Outbox

- Дорогая штука
- При больших нагрузках и идемпотентной переобработке мы не используем
- Вместо него можно попробовать transaction log tailing

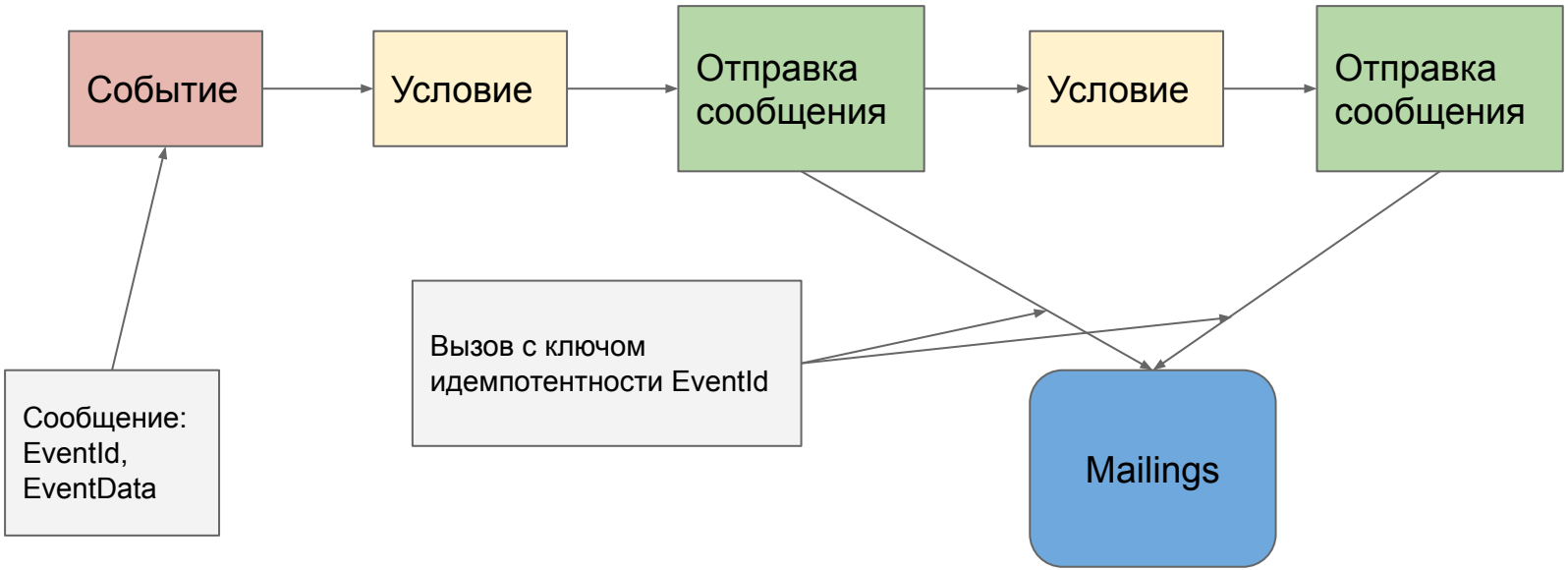
Вернёмся к самим сценариям



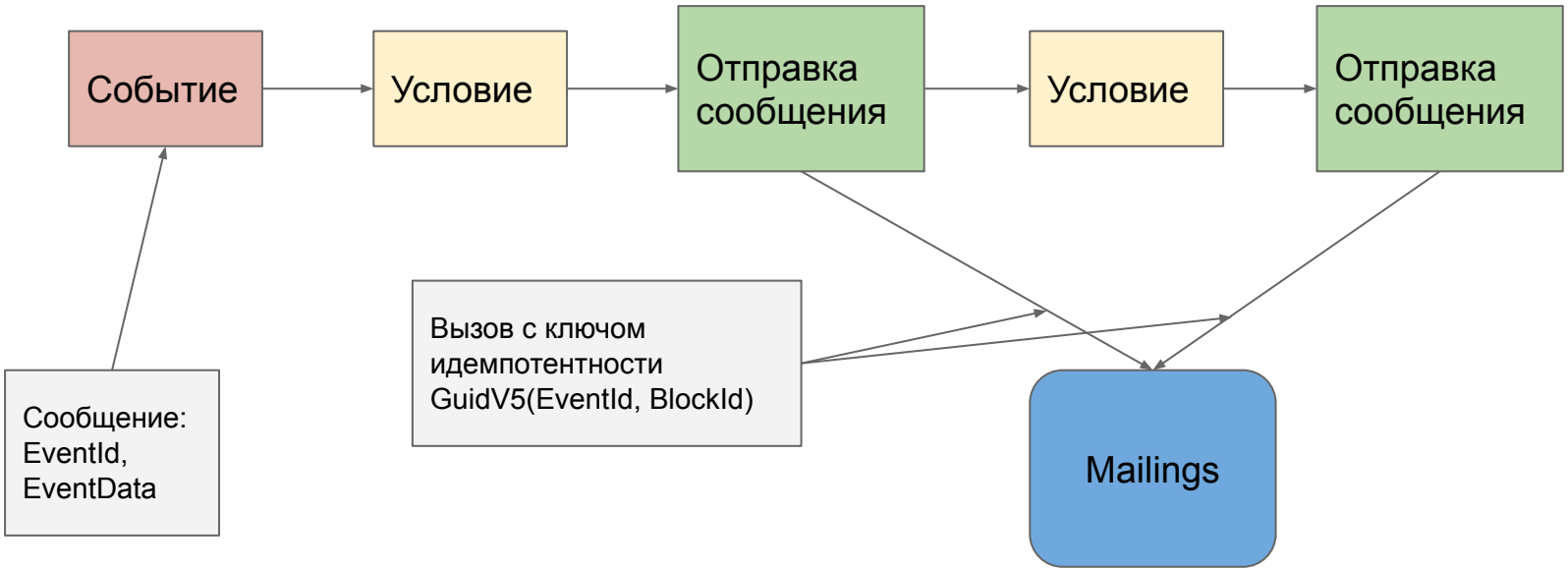
Несколько вызовов сервиса



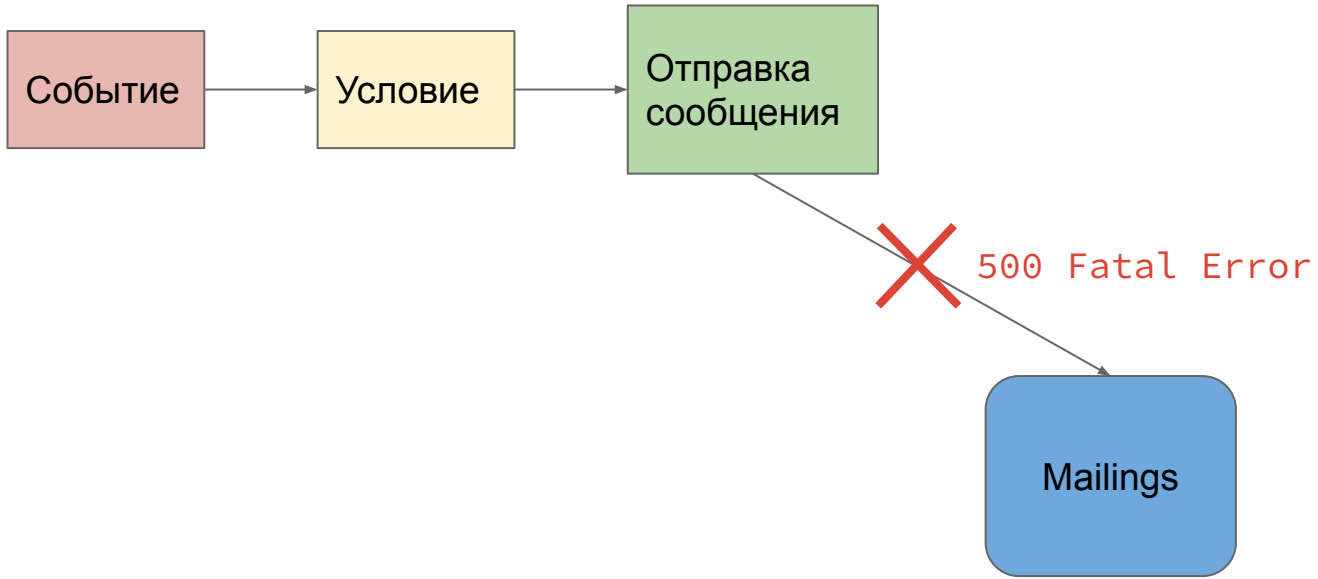
Несколько вызовов сервиса



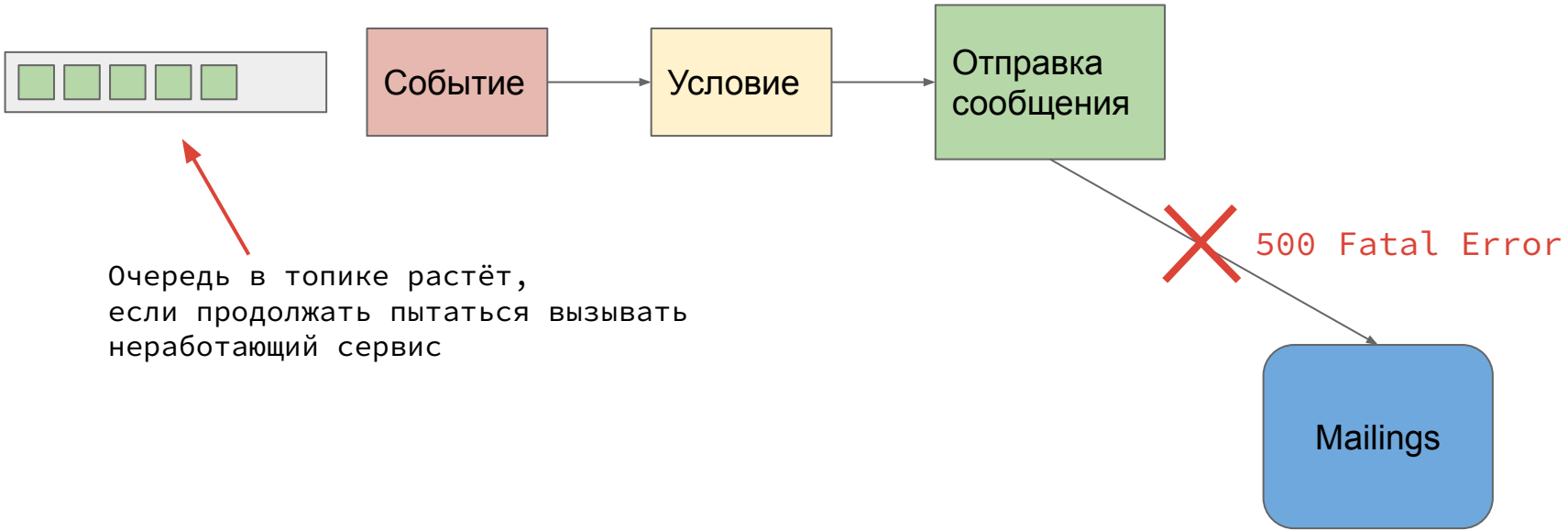
Несколько вызовов сервиса



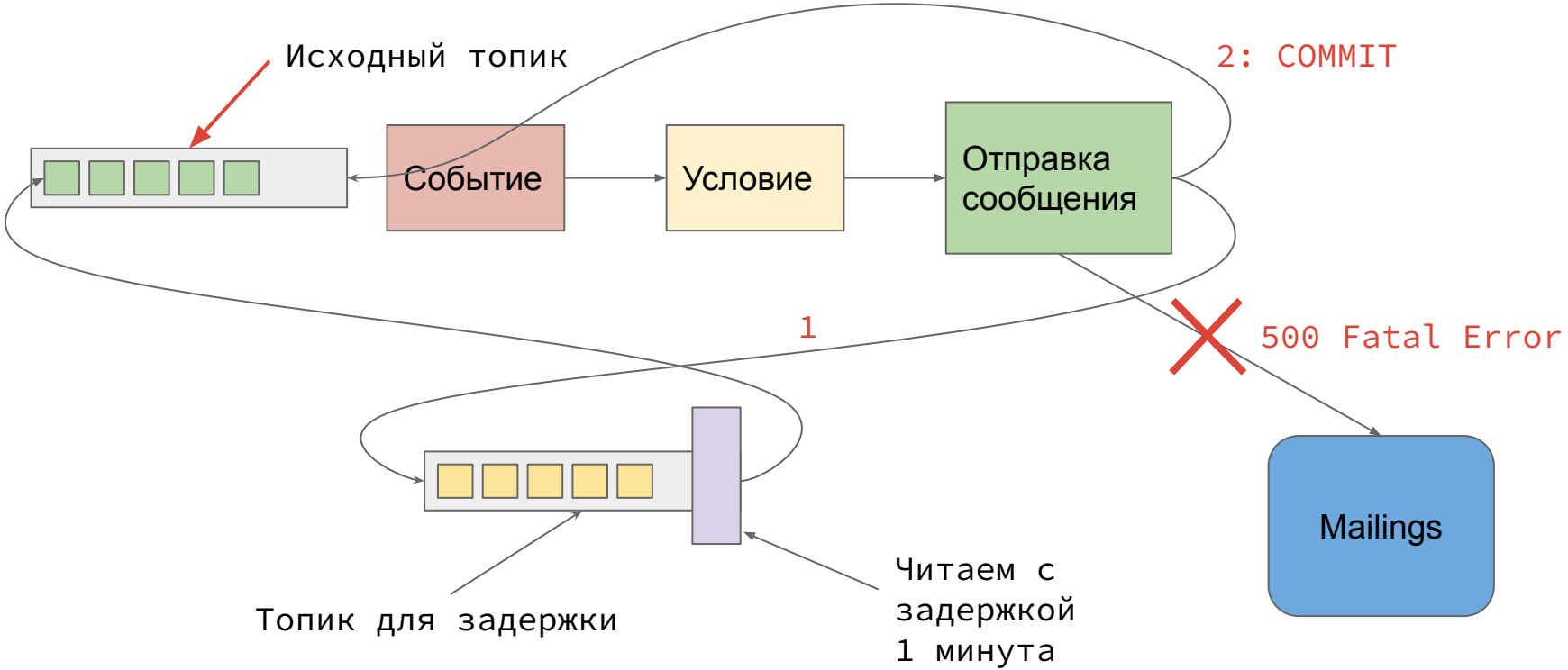
Обработка ошибок



Обработка ошибок



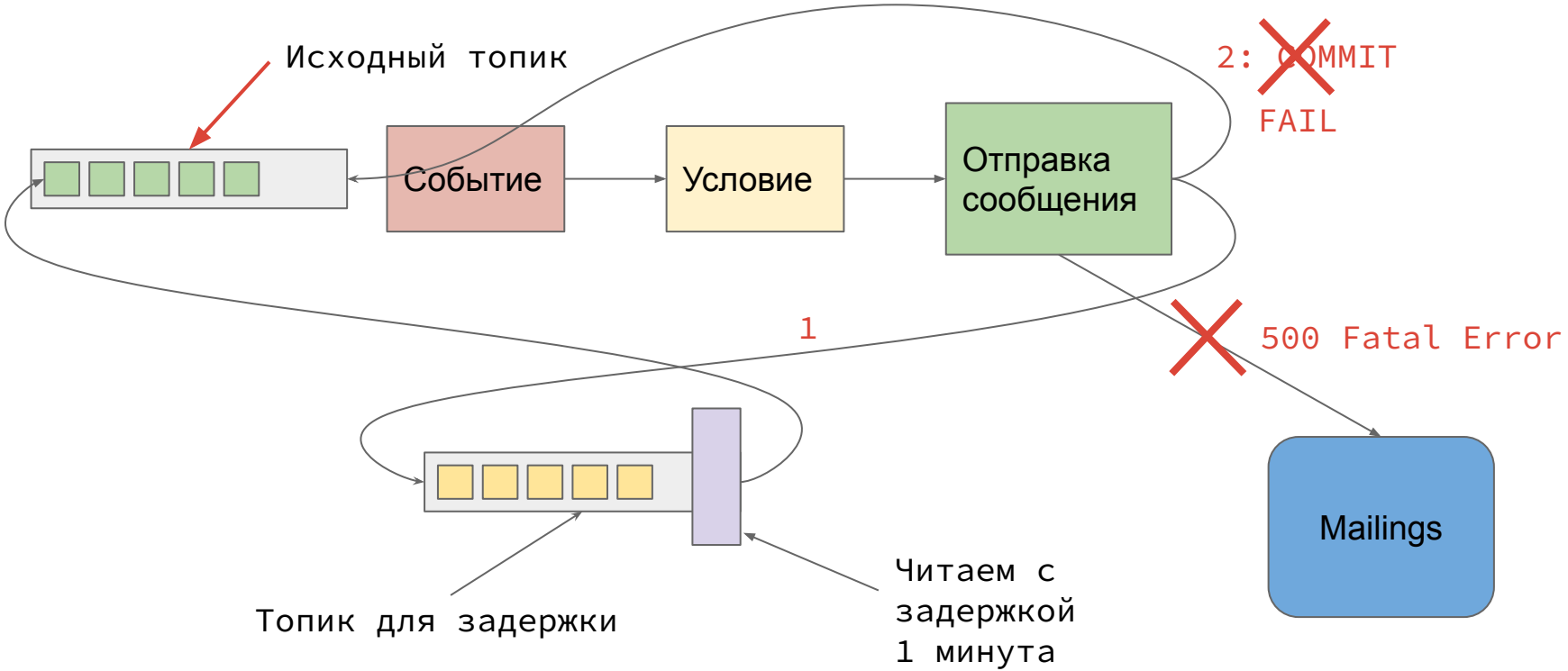
Обработка ошибок



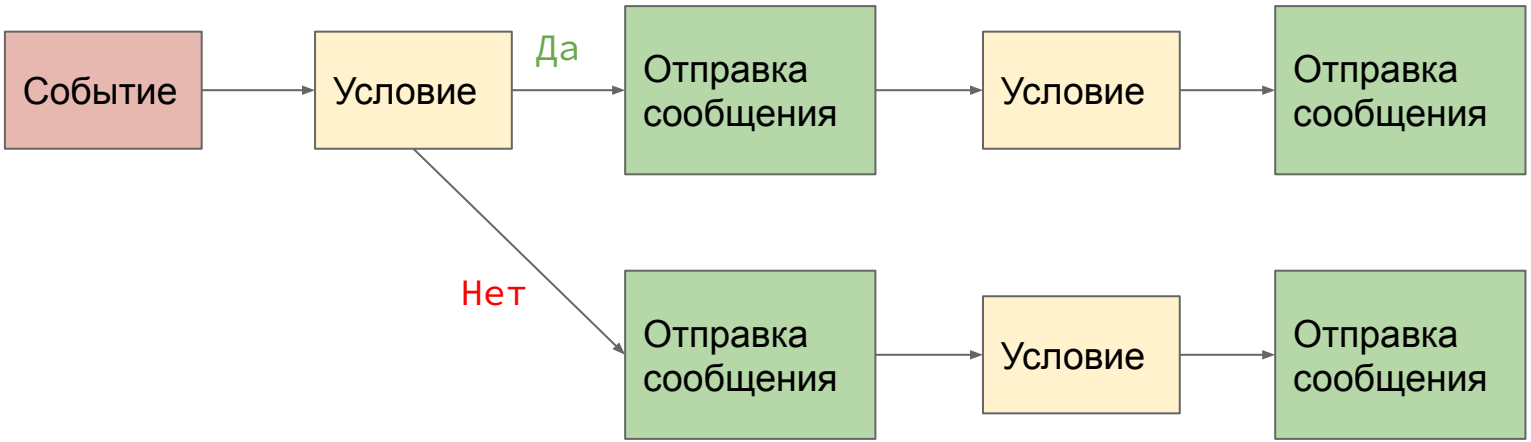
НЮАНСЫ ПОВТОРОВ ДЛЯ ОБРАБОТКИ ОШИБОК

- Порядок будет нарушен
- Константное время повтора – плохо

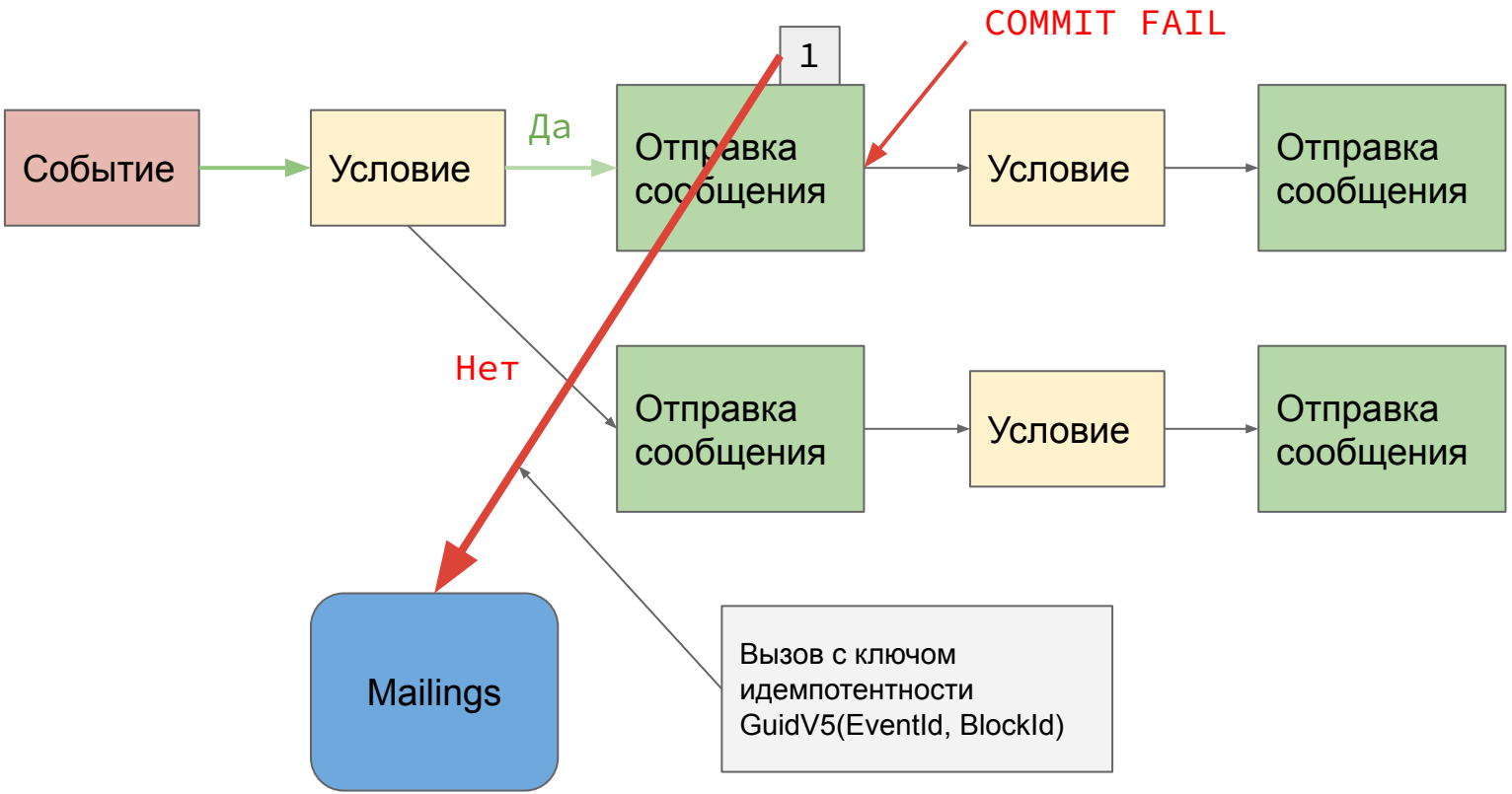
Ещё дубли


Исходный топик
Событие
Условие
Отправка сообщения
2: COMMIT
FAIL
1
500 Fatal Error
Mailings
Топик для задержки
Читаем с задержкой 1 минута

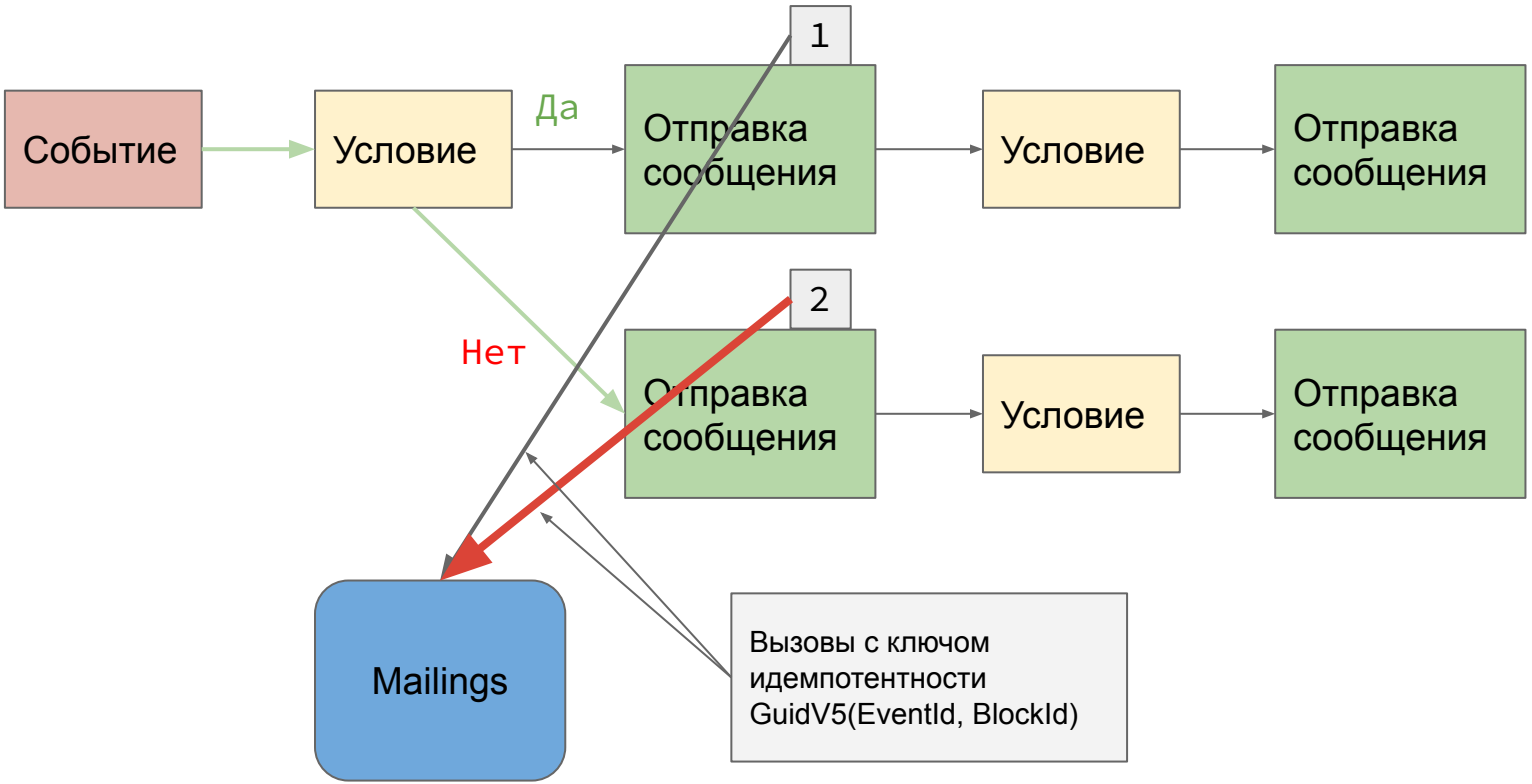
Ветвление


Событие
Условие
Да
Отправка сообщения
Условие
Отправка сообщения
Нет
Отправка сообщения
Условие
Отправка сообщения

Ветвление


COMMIT FAIL
1
Событие
Условие
Да
Отправка сообщения
Условие
Отправка сообщения
Нет
Отправка сообщения
Условие
Отправка сообщения
Mailings
Вызов с ключом идемпотентности GuidV5(EventId, BlockId)

Ветвление


Событие
Условие
Да
Нет
1
Отправка сообщения
Условие
Отправка сообщения
2
Отправка сообщения
Условие
Отправка сообщения
Mailings
Вызовы с ключом идемпотентности GuidV5(EventId, BlockId)

Идея

Идемпотентное применение блоков: сохраняем результат применения блока, при повторе будем проходить по той же цепочке

Результат применения блока

Каждый блок может:

- Остановить выполнение сценария
- Отправить в одно из исходящих соединений

Races

- Множество примеров дублей сообщений и переобработок
- Могут быть параллельным
- Значит необходима синхронизация

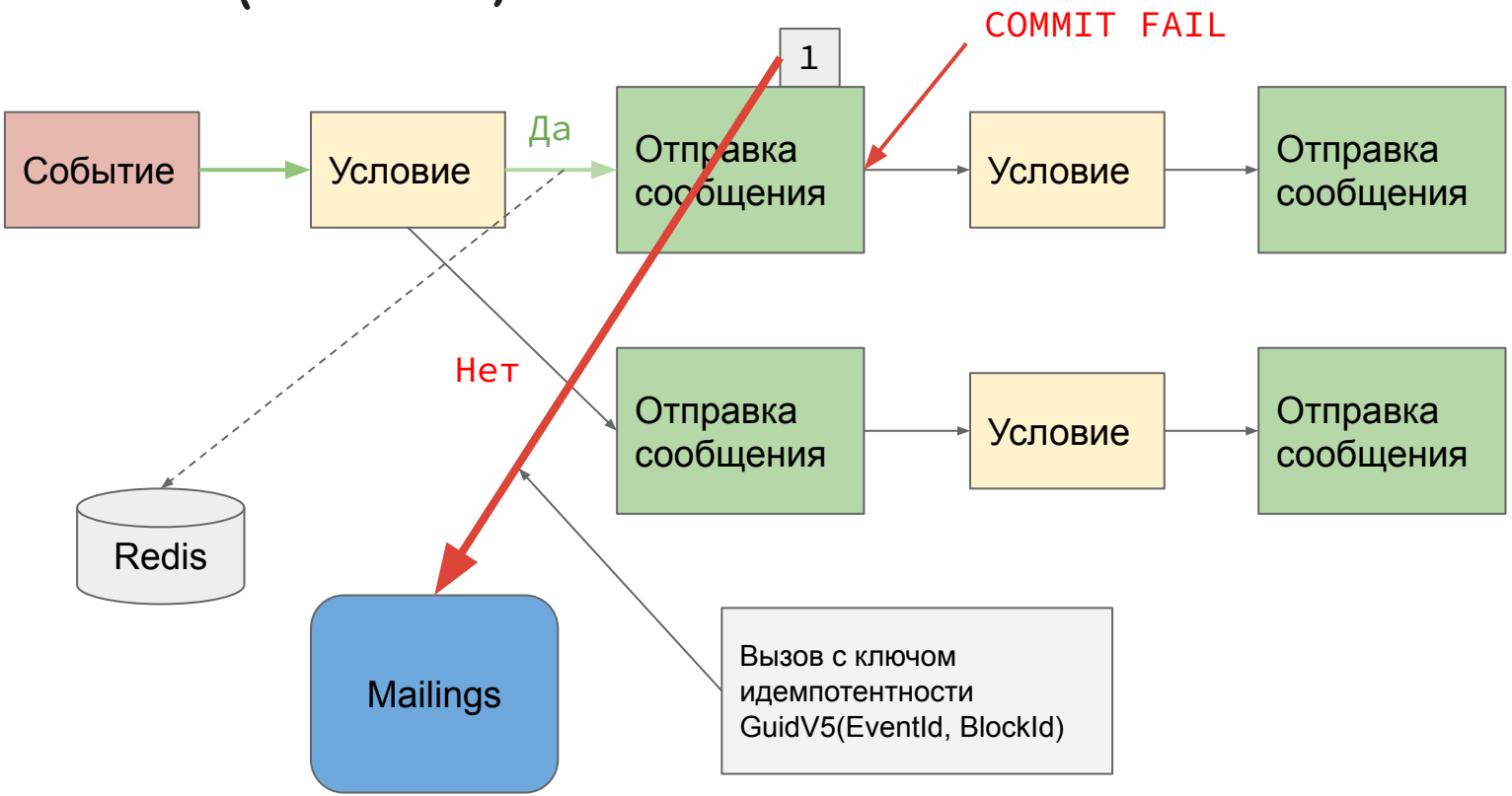
Redis

Быстрое key-value хранилище с горизонтальным масштабированием и (очень ограниченными) транзакциями

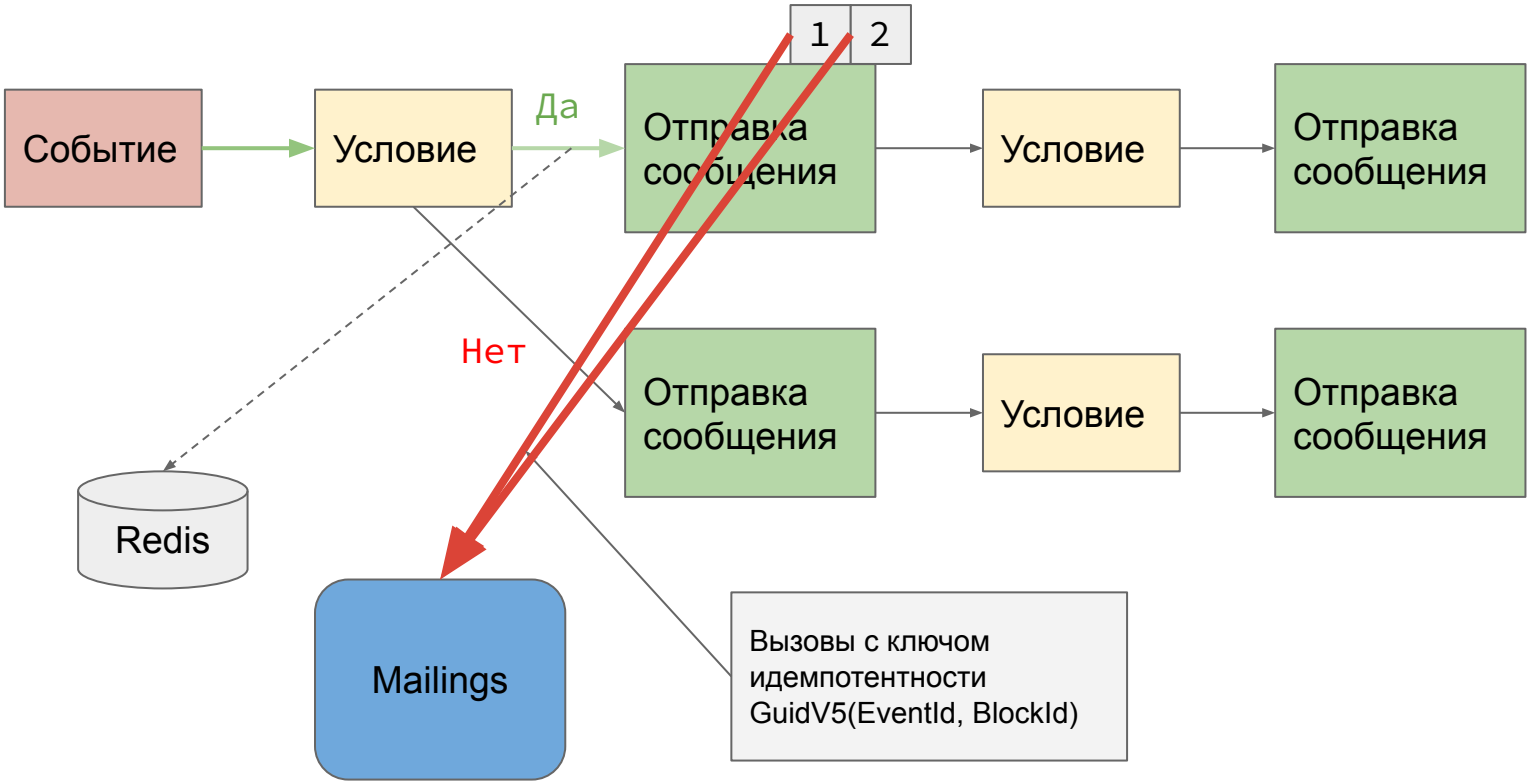
Алгоритм

- Применить блок
- По ключу GuidV5(EventId, BlockId) установить результат через SETNX
- Если не удалось, прочитать по ключу результат и использовать его
- Оптимизация: до применения блока пробовать читать результат

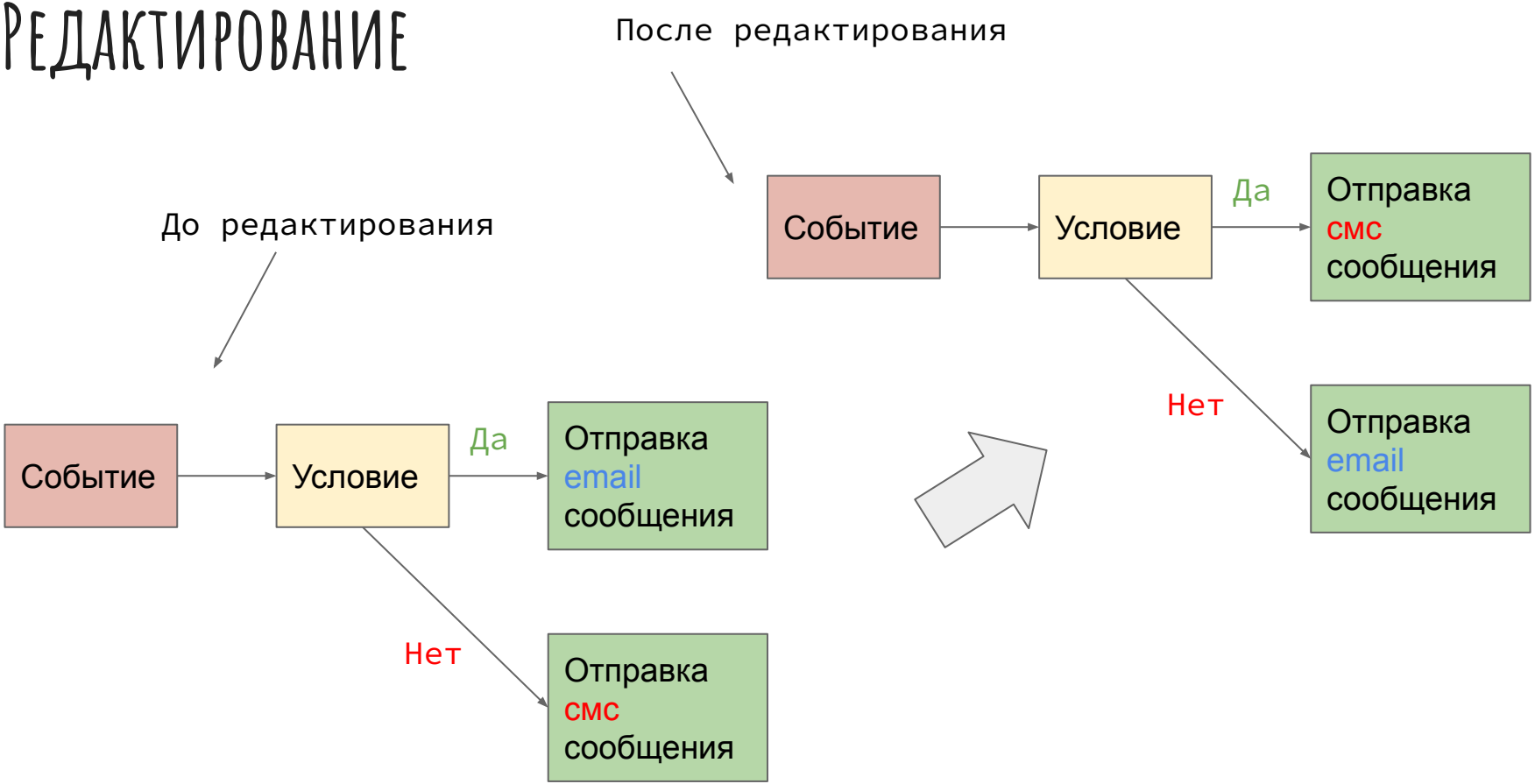
Ветвление (решение)



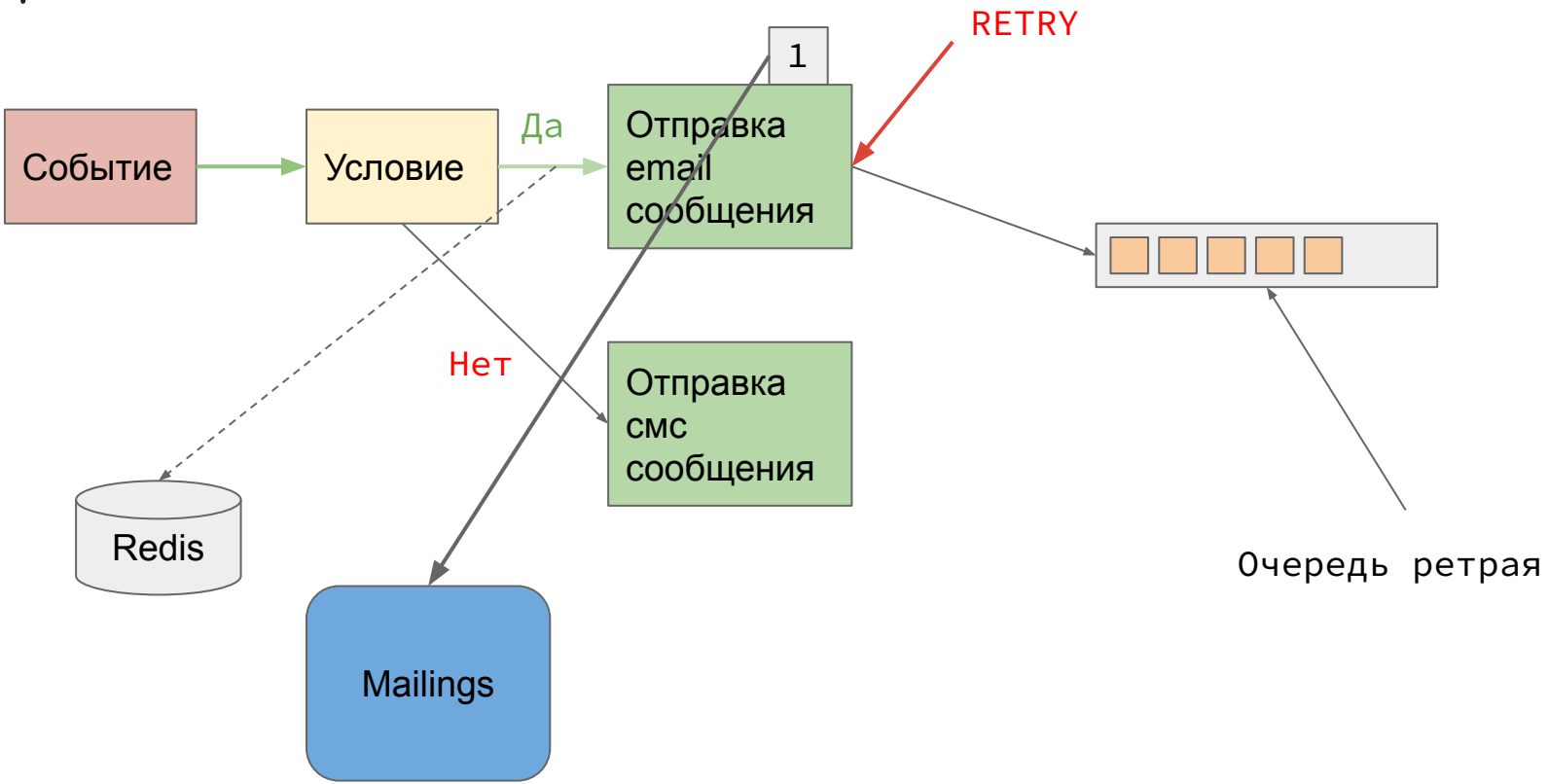
Ветвление (решение)



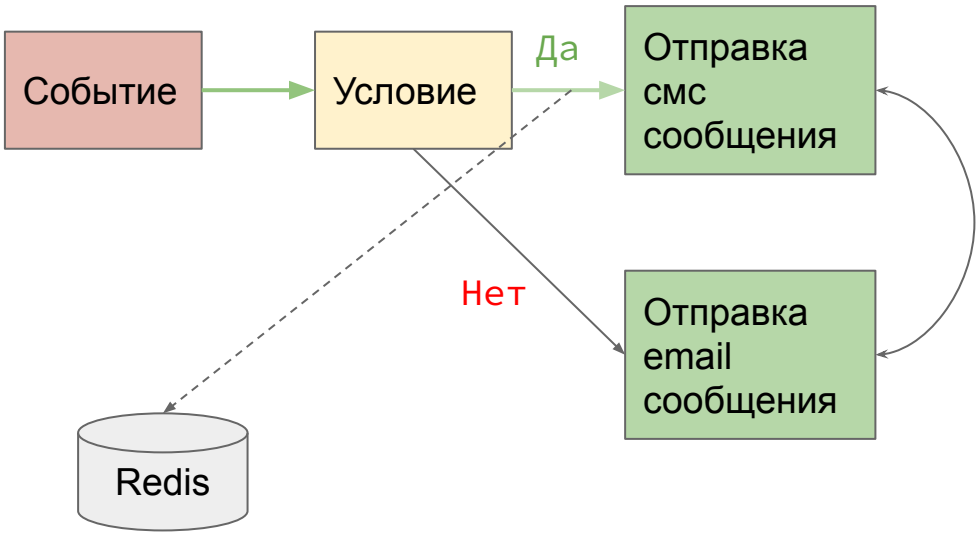
Редактирование

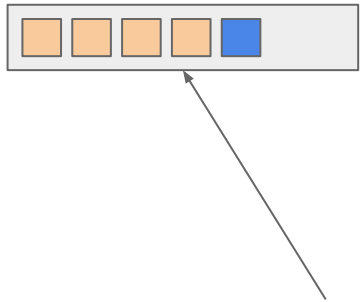


Редактирование



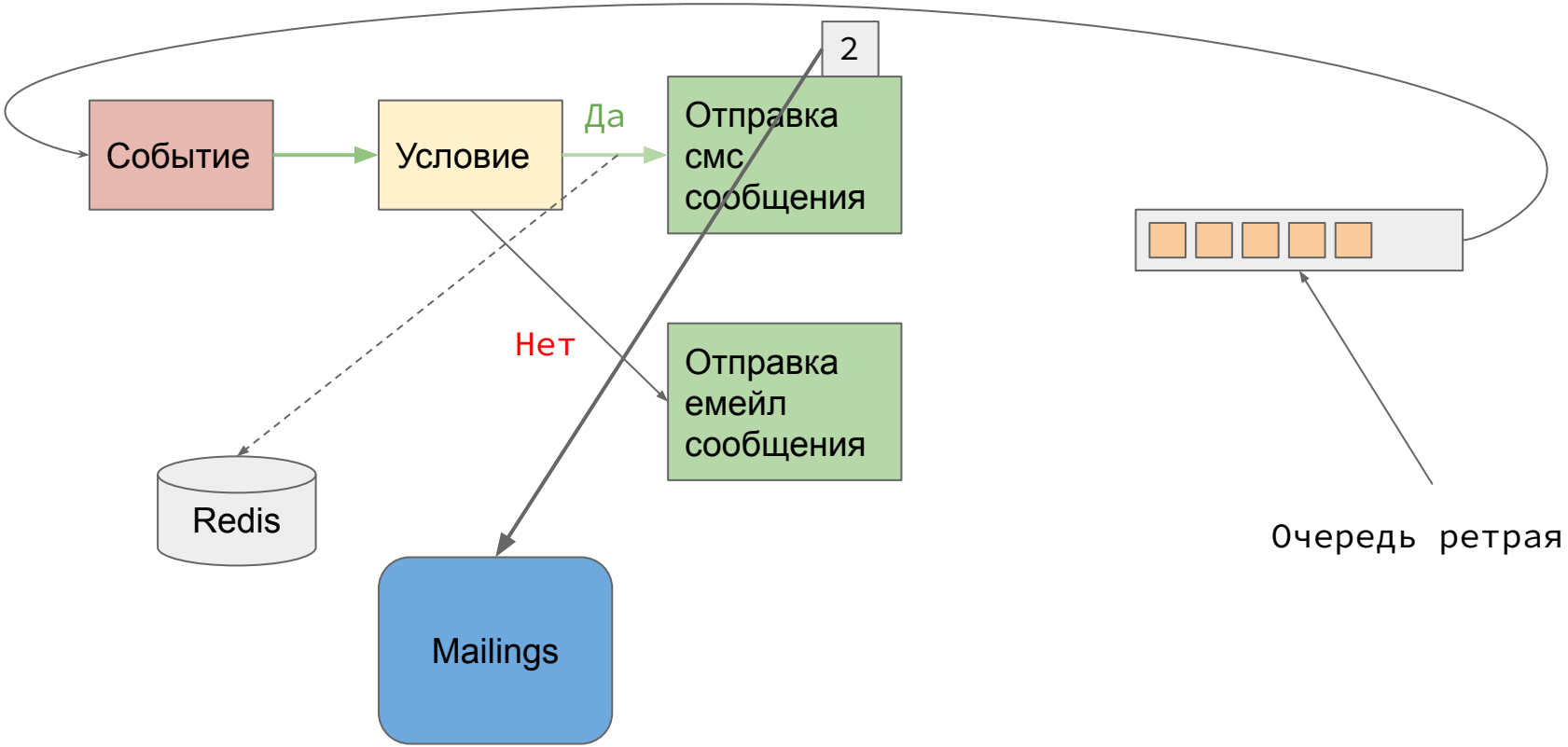
Редактирование





Очередь ретрая

Редактирование

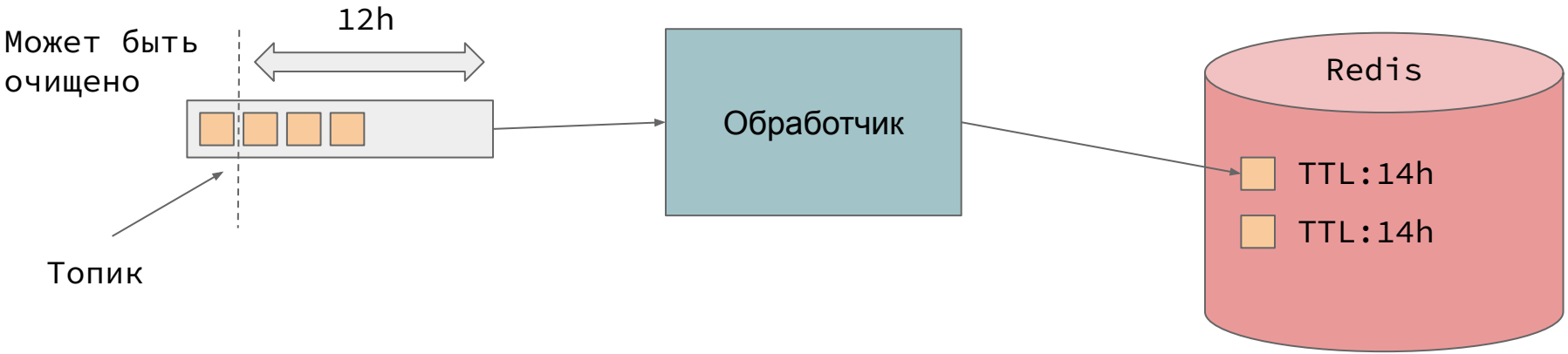


Что делать при редактировании метаданных?

- Хранить историю метаданных
- Гарантировать использование нужной версии метаданных

Сколько хранить ключи идемпотентности?

- Retention в кафке исходя из скорости обработки
- Redis TTL не меньше Retention в кафке из-за ретраев
- Обновление TTL при успешных чтениях (GETEX + PXAT)



Что делать на стороне вызываемого микросервиса?

- Readonly? Ничего (но не всегда).
- Mutable? Транзакция + хранение ключа.
- Внешний вызов? Опционально ТО + идемпотентность.

Хранение ключа прямо в данных

`CustomerMailings`

IdempotentKey	CustomerId	MailingId	DateTimeUtc	…
un42ndt881	1	100	01.01.2020	..
infjs9a3003	2	200	02.01.2020	..

Отдельная табличка для обобщенных транзакций

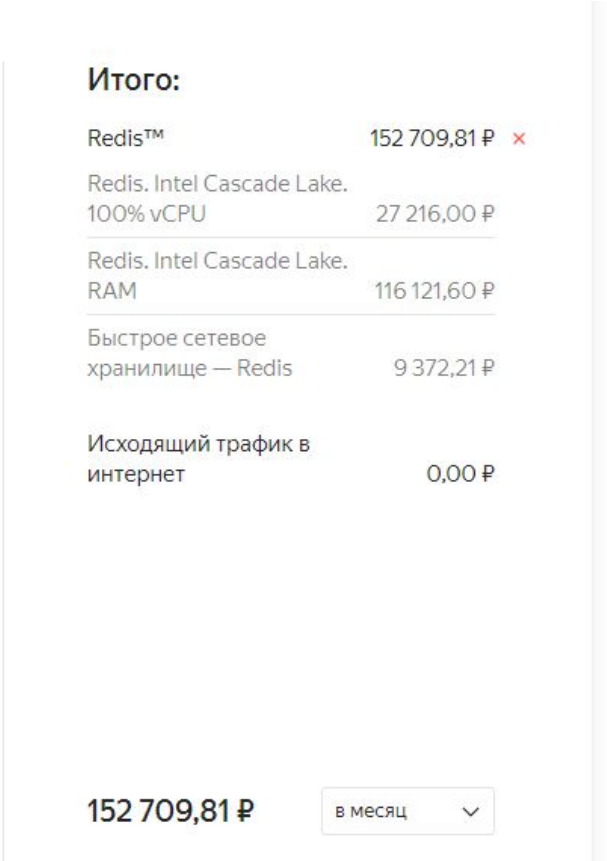
IdempotentKeys

IdempotentKey	DateTimeUtc
un42ndt881	01.01.2020
infjs9a3003	02.01.2020

+ очистка по времени

Что получилось

- Всё работает
- Стоит как каменный мост из-за редиса :)



Ещё раз, а зачем нам редис?

- Борьба с состоянием гонки
- Хранение номера версии сценария
- Хранение результатов применения блоков

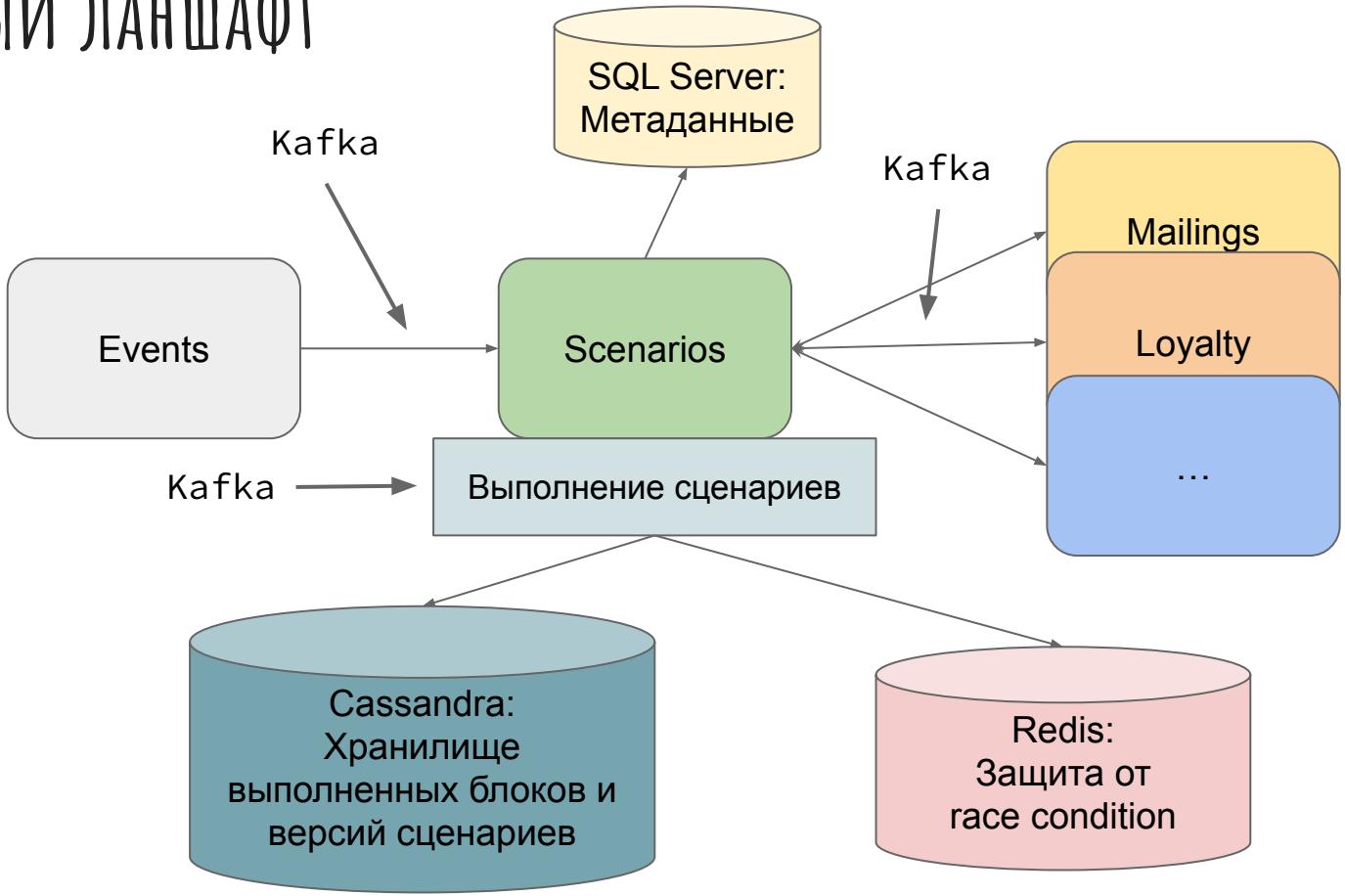
Ещё раз, а зачем нам редис?

- Борьба с состоянием гонки
- ~~Хранение версии сценария~~
- ~~Хранение результатов применения блоков~~

Cassandra: что-то получше

- Горизонтально масштабируется
- Надёжна
- Идеальна для кейса хранения ключ-значения
- Есть TTL
- Есть транзакции, но лучше не использовать

Новый ландшафт



Саммари

Для межсервисного взаимодействия:

- ТО
- Идемпотентность как зомби
- Правильный выбор технологий

Литература

- Доклад по Kafka из первого дня :)
- microservices.io + Microservices Patterns
- Kafka: The Definitive Guide
- Kafka VS Rabbit

Конец

Вопросы приветствуются

Телеграмм **@timramone**