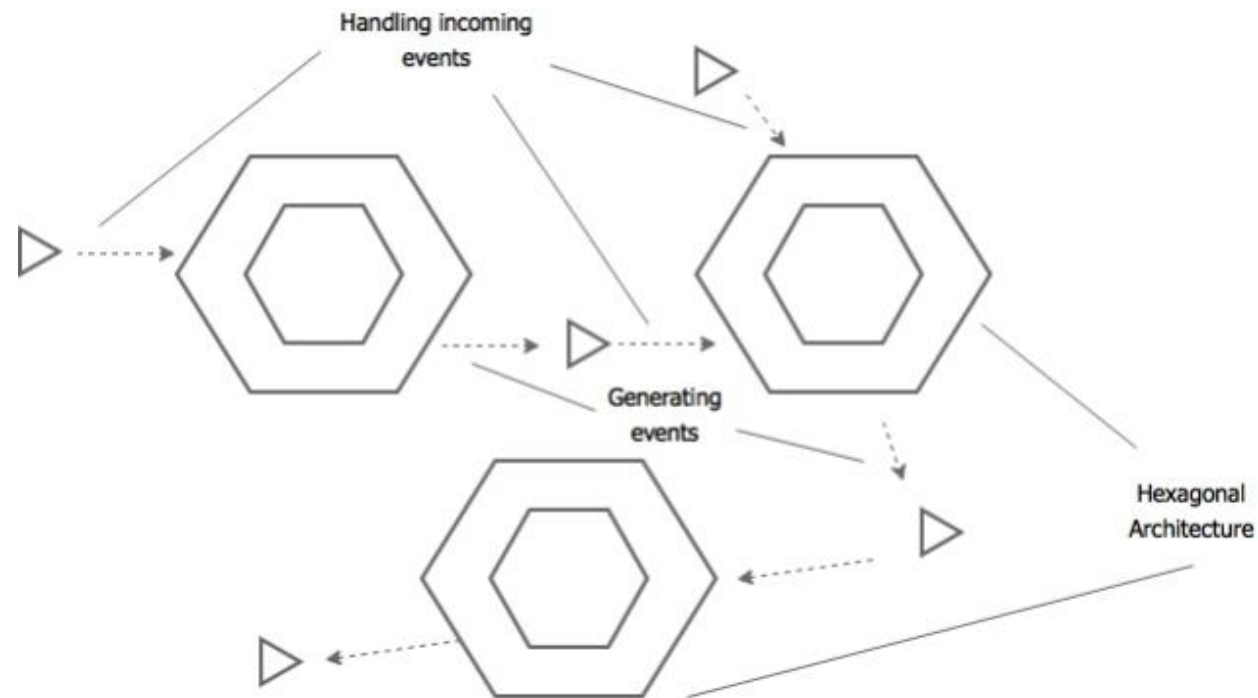


Системный дизайн

event driven architecture

Определение

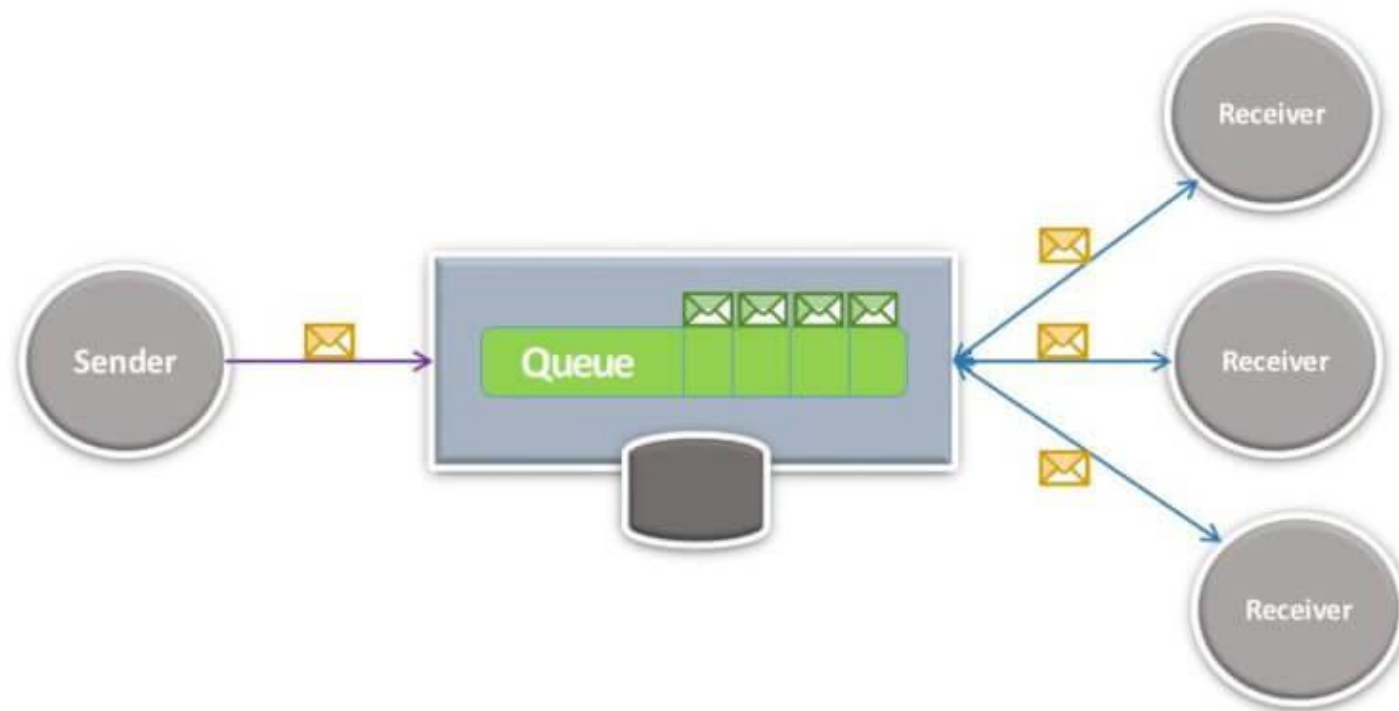
Архитектурный стиль ориентированный на создание, обнаружение, потребление и реакцию на события.



Преимущества EDA

- **Слабая связь между системами.**
Само событие не знает о последствиях своего возникновения.
- **Улучшенная масштабируемость.**
Асинхронные системы, как правило, более масштабируемы, чем синхронные. Отдельные процессы меньше блокируются и имеют меньше зависимостей от удаленных/распределенных процессов.

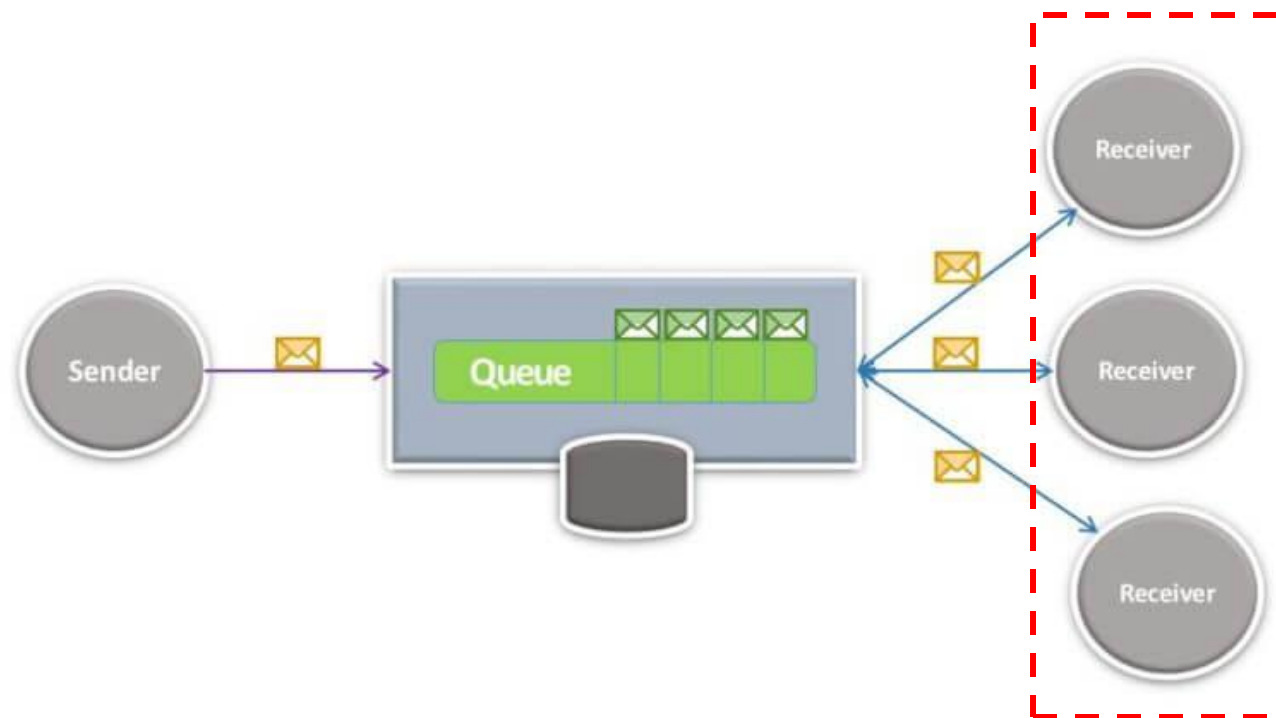
Брокеры сообщений



1 Redundancy via Persistence

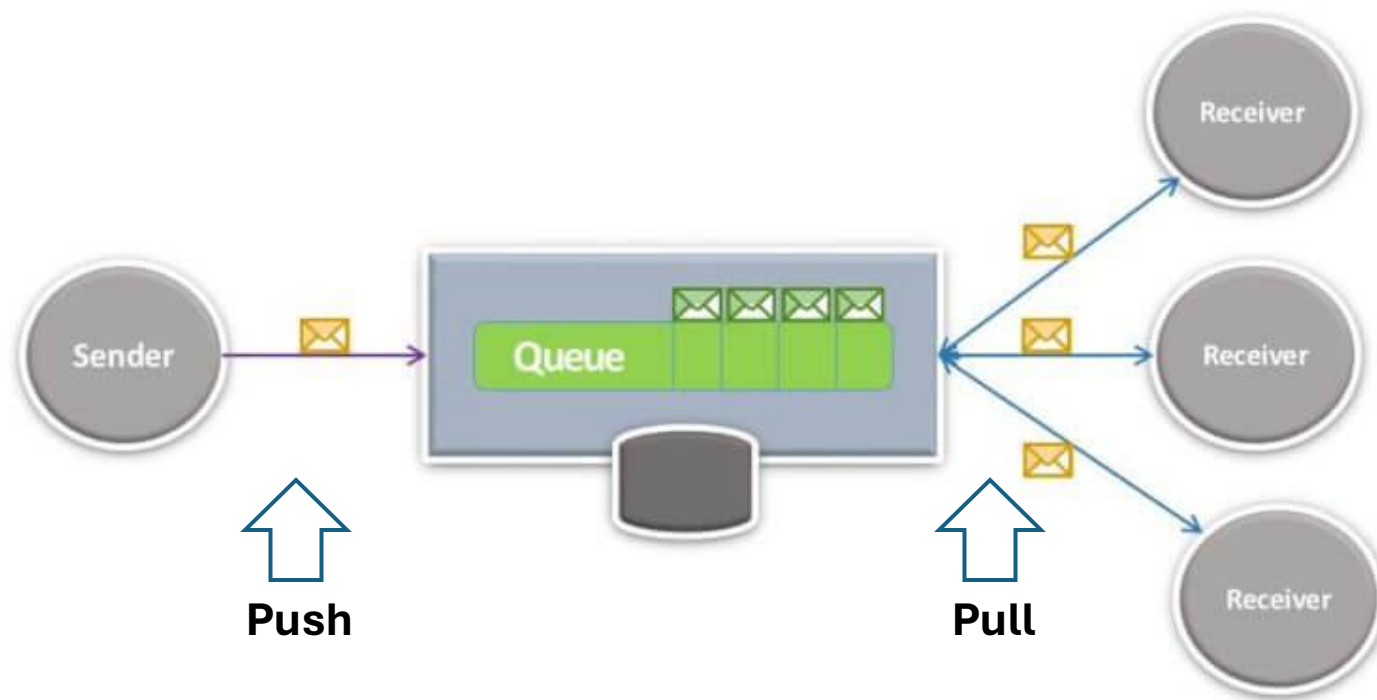
Очереди позволяют масштабировать сервера с целью обеспечения надежности;

Очереди содержат механизмы хранения данных на дисках, что позволяет сделать обработку надежной;



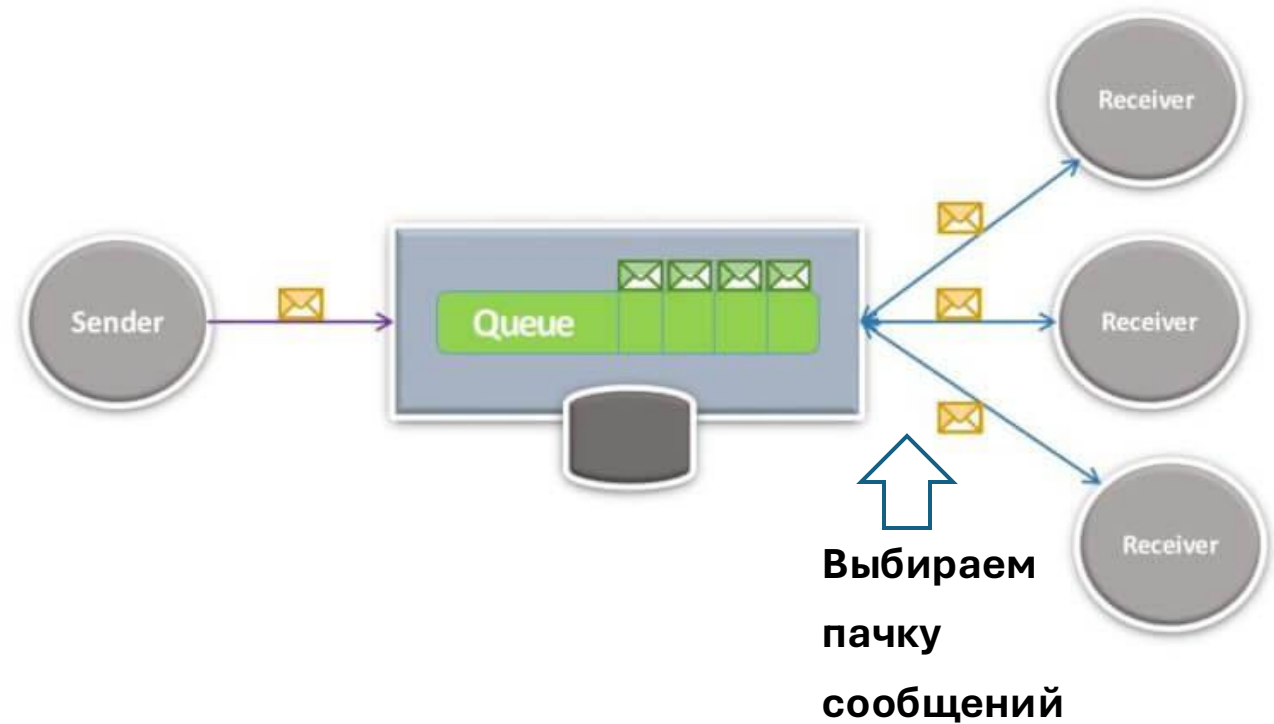
2 Traffic Spikes

Очереди позволяют обрабатывать входящие события со скоростью, которую могут себе позволить сервера. Таким образом сглаживая скачки трафика.



3 Batching for Efficiency

За счет того что мы выполняем запросы не синхронно, а накапливаясь в очередях, мы имеем возможность вычитывать сразу несколько сообщений и обрабатывать их пакетно. Что может дать выигрыш в производительности.



4

Asynchronous Messaging

В том случае, если вычисления надо произвести не сразу, то очереди - это удобный механизм организации такой обработки.

5 Decouple by Using Data Contracts

Это отличный инструмент по уменьшению связанности между системами. Все что их связывает – это «контракт» сообщения.

6 Transaction Ordering and Concurrency Challenges

За счет того что мы управляем тем как обрабатываем данные в очереди мы можем управлять тем сколько параллельных запросов обрабатывается одновременно.

7 Monitoring

Мы можем знать сколько в очереди сообщений и тем самым понимать на сколько производительна наша система.

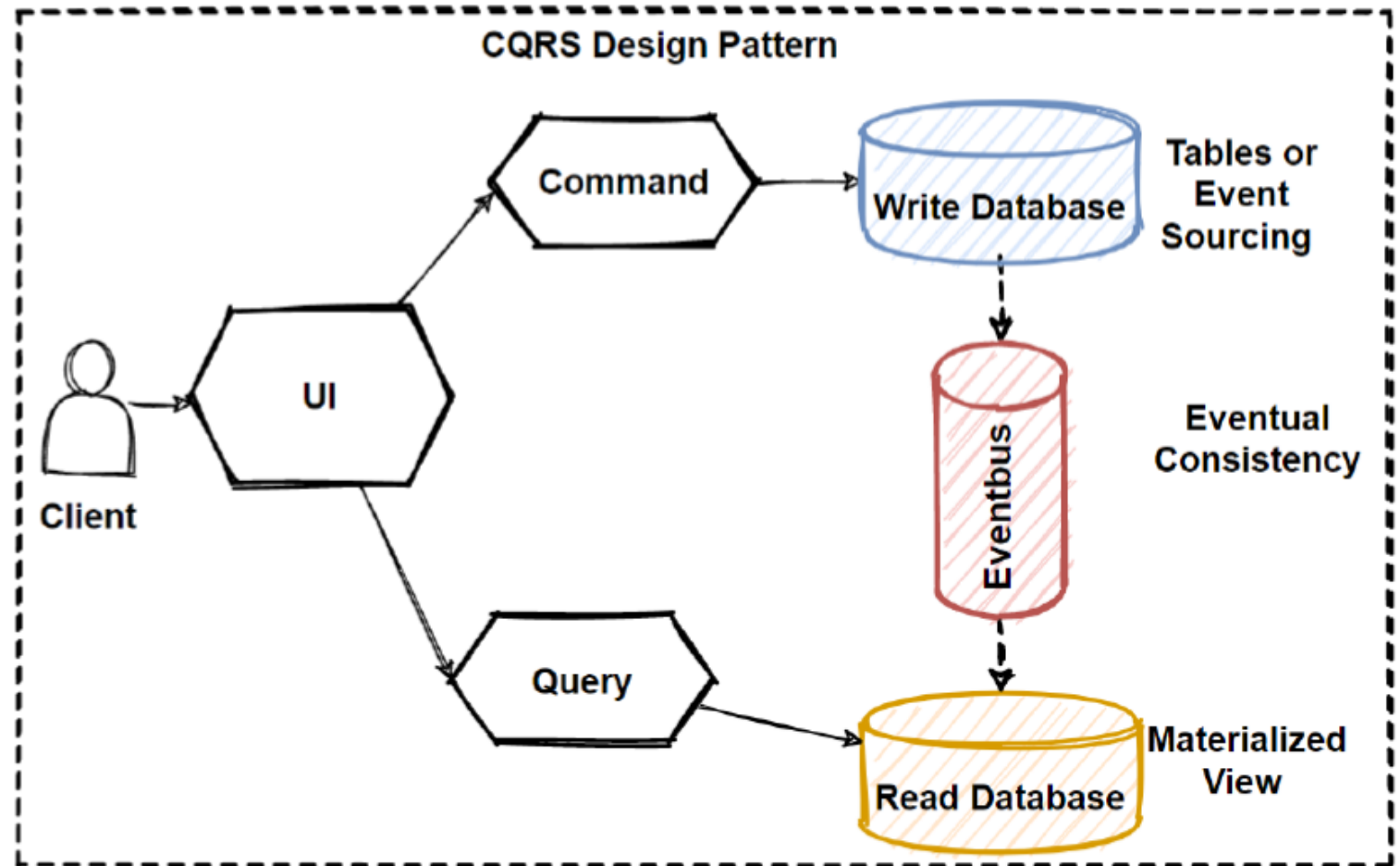
Паттерны асинхронного взаимодействия

Какие задачи решаем

-
1. **Буфер** –быстрый писатель / медленный читатель
 2. **Транспорт** –унификация доставки сообщений
 3. **Диспетчер маршрутов обработки задач** (сообщений)
 4. **Балансировщик**

1 Тактика: «buffer»

Пример: паттерн CQRS



Когда применять

- CQRS можно использовать, когда совместное использование одной и той же модели для чтения и записи затруднено.
- CQRS можно использовать в высокопроизводительных приложениях. Благодаря ему модели чтения и записи масштабируются независимо друг от друга
- CQRS следует использовать только в определенных частях системы (Bounded Context). Каждый ограниченный контекст требует собственных решений о том, как его моделировать.
- CQRS может значительно усложнить систему и должен использоваться с большой осторожностью.

1 Тактика: «buffer»

Плюсы:

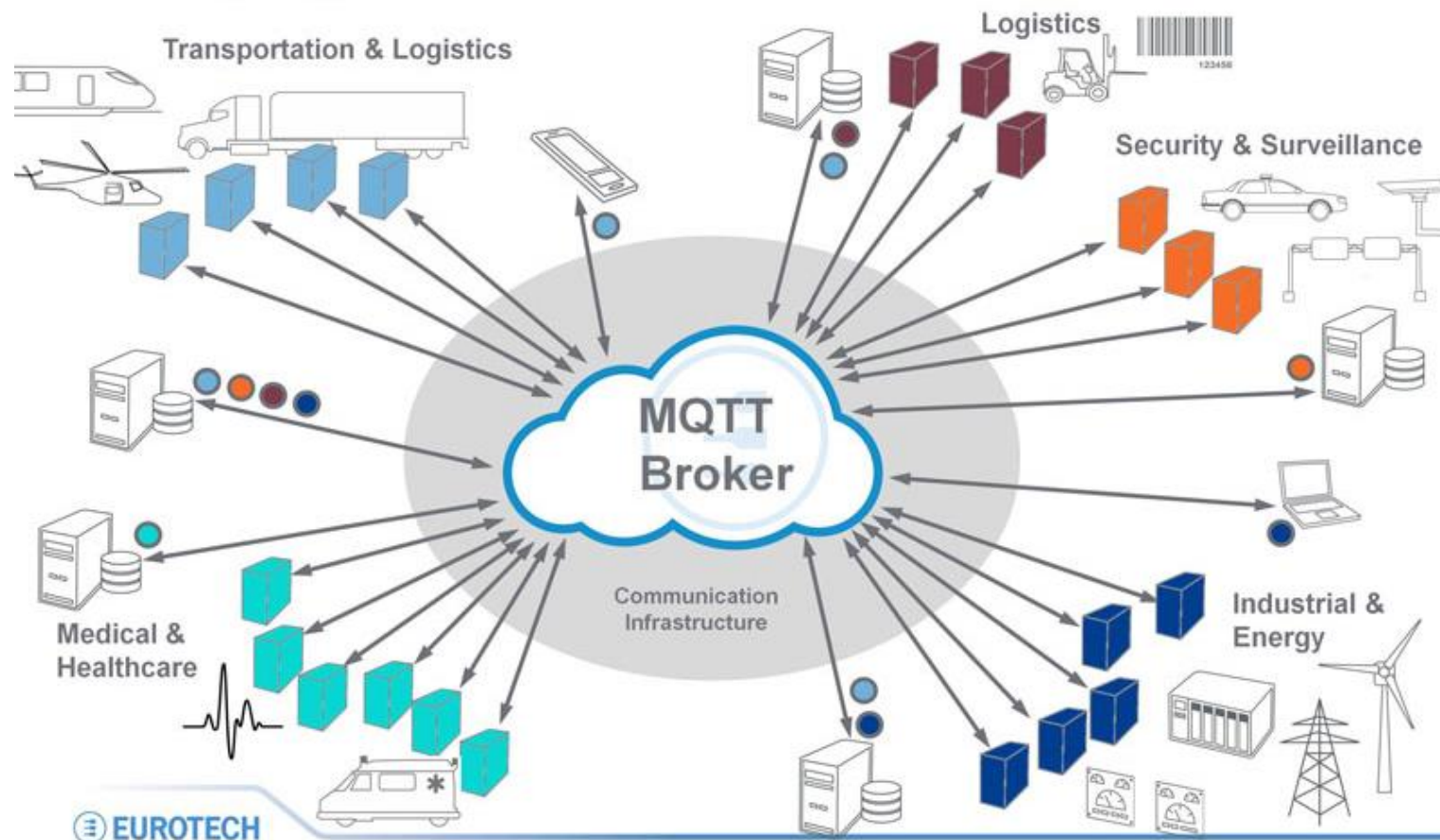
- Возможность выдерживать большие нагрузки при операциях чтения из-за ухода от ресурсоёмких запросов к БД на запись
- Независимое масштабирование операций чтения и записи
- Более простые классы за счет разделения ответственности, а значит проще обслуживать такой код
- Большая гибкость описания прав доступа, так как можно отдельно прописывать права для обращений на чтение и запись

Минусы:

- Не смотря на простоту формулировки CQRS сложность может крыться в деталях. Например, в организации механизма синхронизации БД чтения с БД записи
- Нужно организовывать подсистему обработки событий
- Проблема загрузки пользователем устаревших данных, если БД чтения отстаёт от БД записи

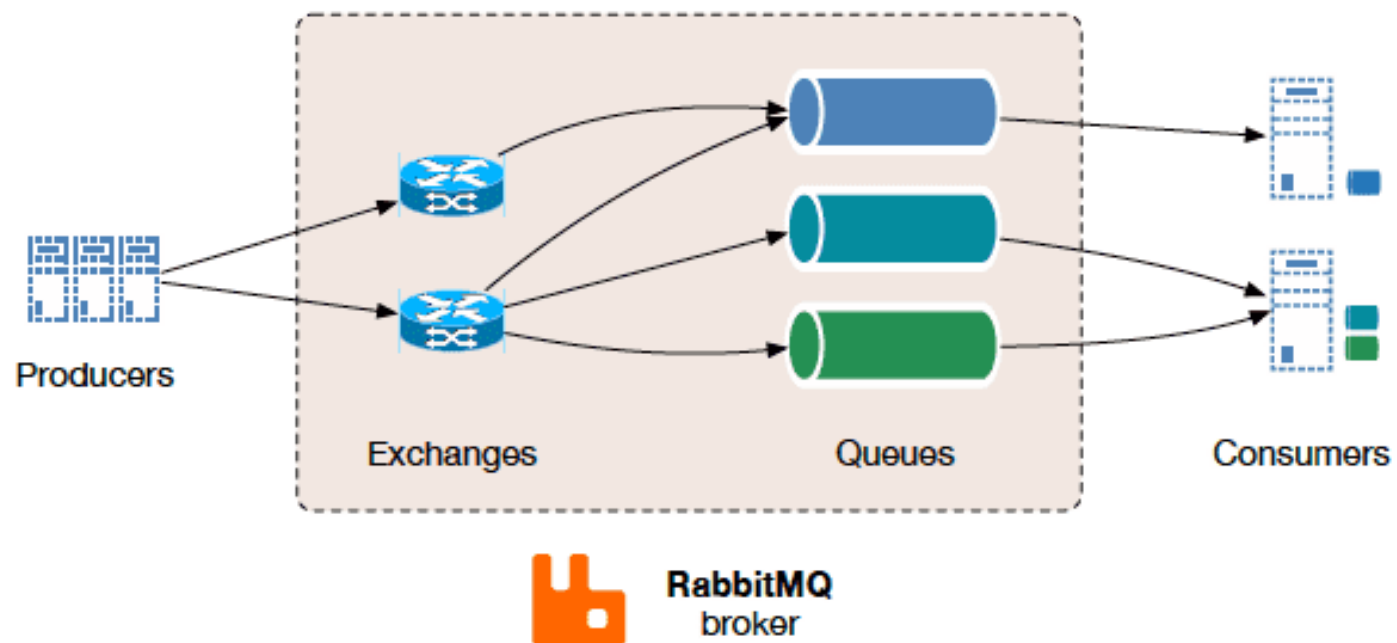
2 Тактика: «transport»

Пример – IOT Hub



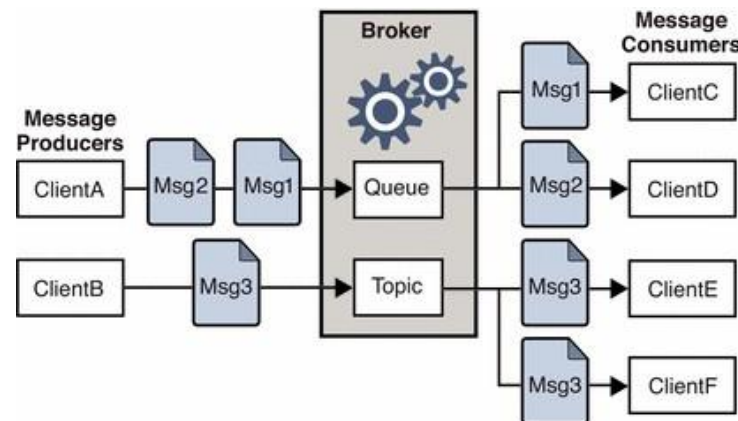
Возможность
устанавливать различные
обработчики разным типам
событий и независимо их
масштабировать

3 Тактика «dispatcher»



4 Очереди - балансировщики

1. Очередь подразумевает что все ресурсы равноправные. По этому то что первый пришедший запрос обрабатывается первым - это нормально.
2. Позволяет ограничивать нагрузку на процессоры.
3. Позволяет балансировать нагрузку между процессорами.
4. Может использоваться как одна очередь с несколькими читателями, так и несколько очередей (для каждого читателя своя).

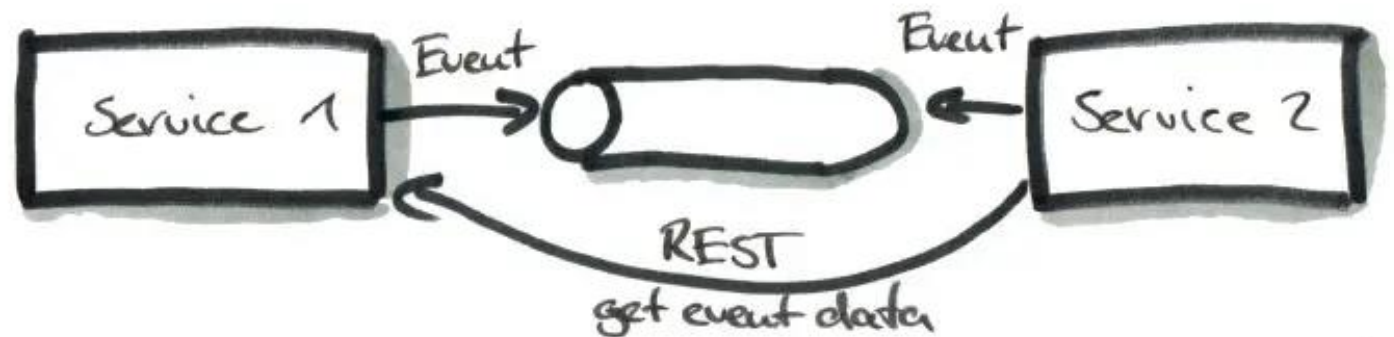


Вопрос?

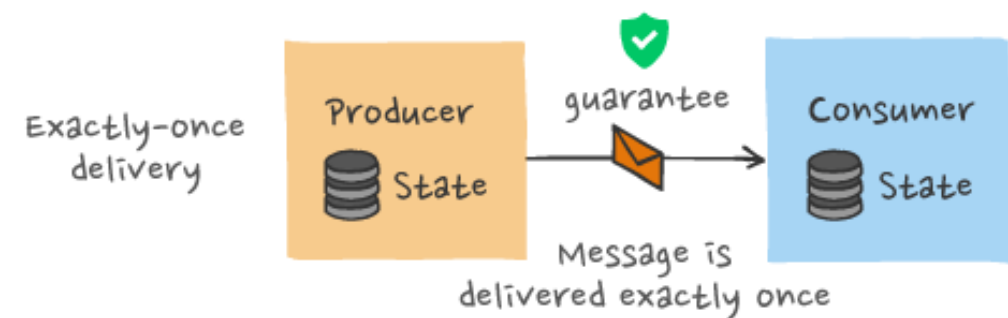
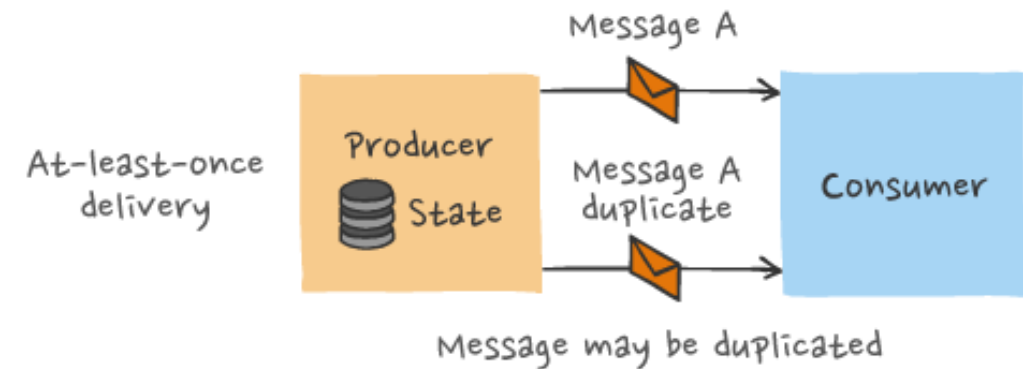
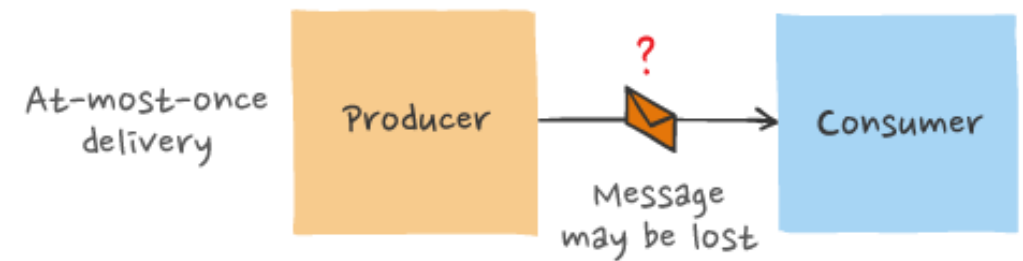
Предположим мы хотим передавать между системами большие файлы. Стоит ли нам отправлять такие файлы с помощью сообщений через очереди?

Zero-Payload Events

- В сообщении отправляется только информация о типе события, идентификатор и ссылка на ресурс
- Сервис-получатель самостоятельно запрашивает нужные данные в сервисе отправителе по ссылке
- Такой подход позволяет не нагружать очередь большими сообщениями и дать возможность получателю всегда обрабатывать актуальные данные



Гарантии доставки сообщений



Проблемы в Event Driven Architecture

- Многократная реакция на сообщение
- «Убийственные» задачи
- Зависание обработчиков
- Персистентность сообщений
- Порядок параллельной обработки сообщений

Сервера

RabbitMQ

Kafka

Redis

NATS

Apache Pulsar

JMS Message Queue

...

<https://ultimate-comparisons.github.io/ultimate-message-broker-comparison/>

Протоколы

- STOMP (Simple Text Oriented Message Protocol)
- MQTT (Message Queue Telemetry Transport)
- AMQP (Advanced Message Queueing Protocol)

```
asyncapi: 3.0.0
info:
  title: Account Service
  version: 1.0.0
channels:
  userSignedup:
    address: 'user/signedup'
    messages:
      userSignedupMessage:
        $ref: '#/components/messages/UserSignedUp'
operations:
  processUserSignups:
    action: 'receive'
    channel:
      $ref: '#/channels/userSignedup'
components:
  messages:
    UserSignedUp:
      payload:
        type: object
        properties:
          displayName:
            type: string
            description: Name of the user
          email:
            type: string
            format: email
            description: Email of the us|
```

Спецификация протокола уровня приложения

-
- <https://www.asyncapi.com/docs/getting-started/coming-from-openapi>
 - <https://github.com/cloudevents/spec>

RabbitMQ



AMQP

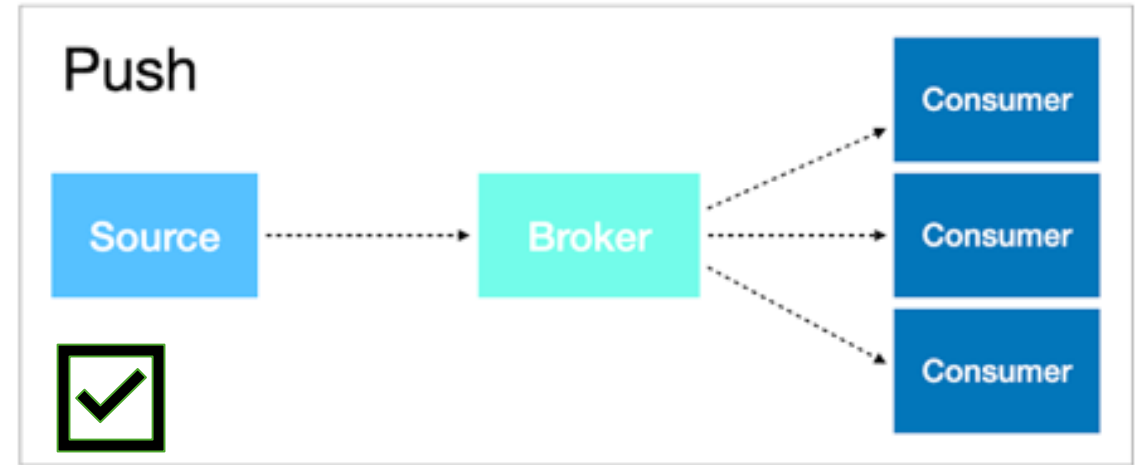
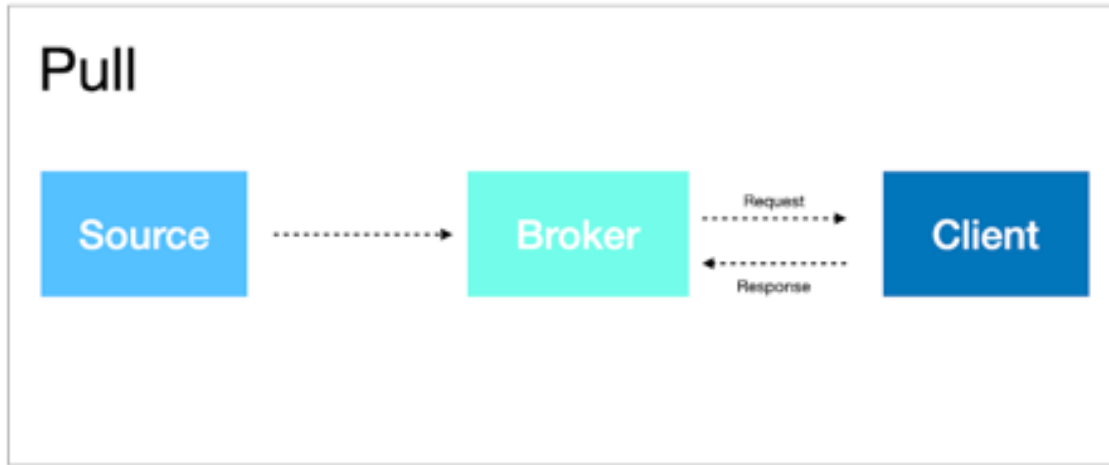
- Международный стандарт.
- Не привязан к поставщику очереди и платформе.
- Спецификация протокола:
<http://docs.oasis-open.org/amqp/core/v1.0/amqp-core-complete-v1.0.pdf>
- Описывает транспортный уровень передачи сообщений.

Свойства RabbitMQ

- Транспорт
- Отказоустойчивость
- Производительность

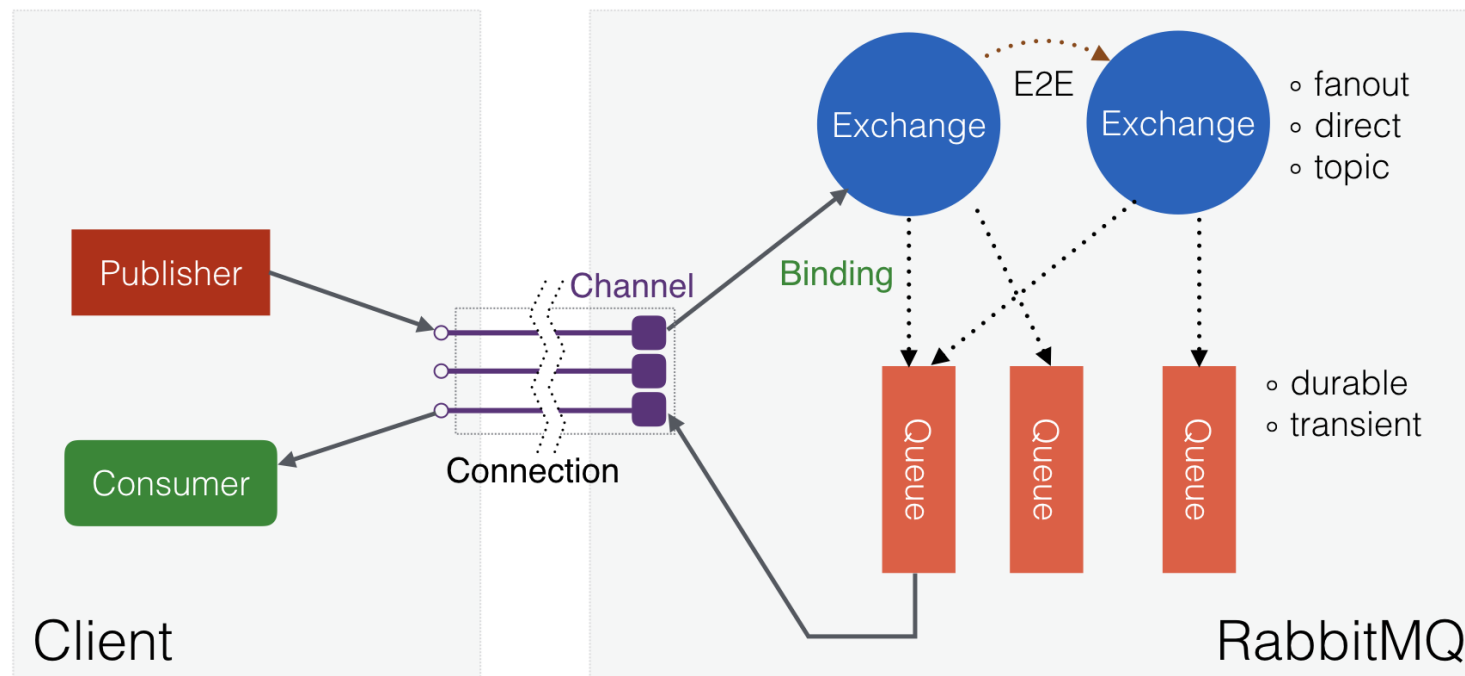
Особенности RabbitMQ

- Используется виртуальная машина Erlang
- Относительная простота настройки и использования
- Поддержка большинства ОС
- Open source
- Использование протокола AMQP
- Применяет Push-модель в доставке сообщений
- Красное название



Push model

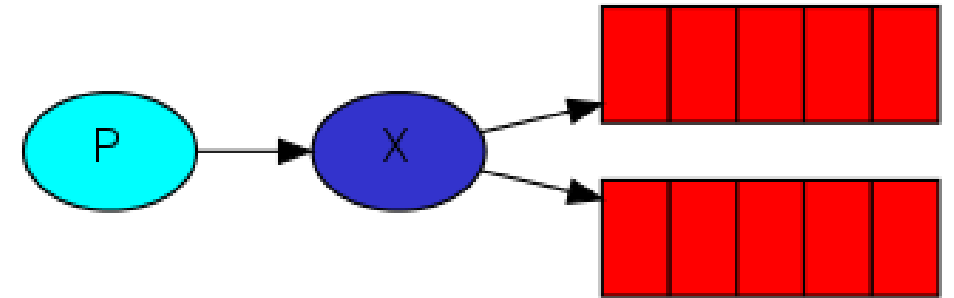
Принцип работы



- **Producer** (поставщик) – программа, отправляющая сообщения
- **Queue** (очередь) – буфер, хранящий сообщение
- **Consumer** (подписчик) – программа, принимающая сообщения.
- Основная идея в модели отправки сообщений Rabbit – Поставщик(producer) никогда не отправляет сообщения напрямую в очередь. Фактически, довольно часто поставщик не знает, дошло ли его сообщение до конкретной очереди. Вместо этого поставщик отправляет сообщение в точку доступа.

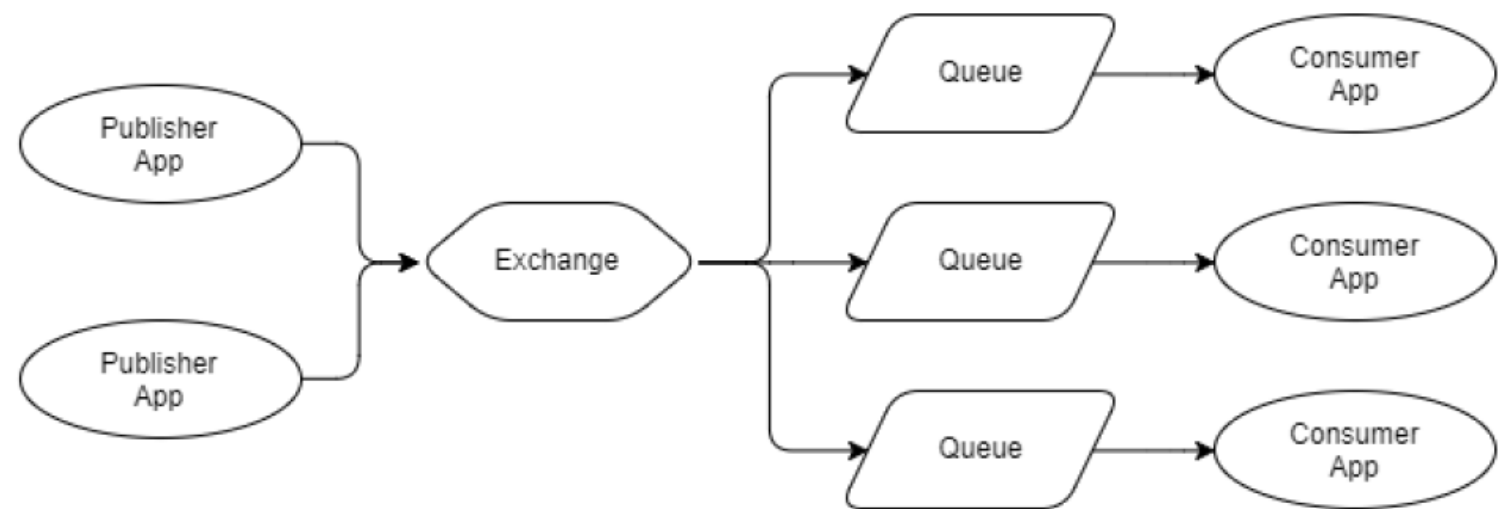
Точка доступа выполняет две функции:

- получает сообщения от поставщика;
- отправляет эти сообщения в очередь.

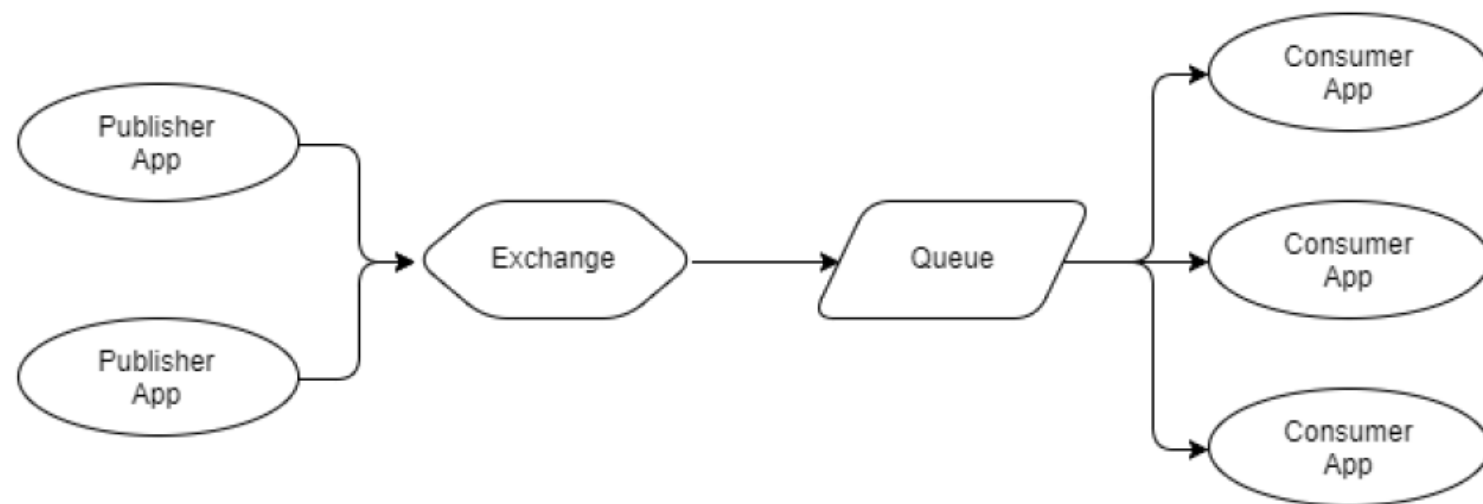


Точка доступа exchange

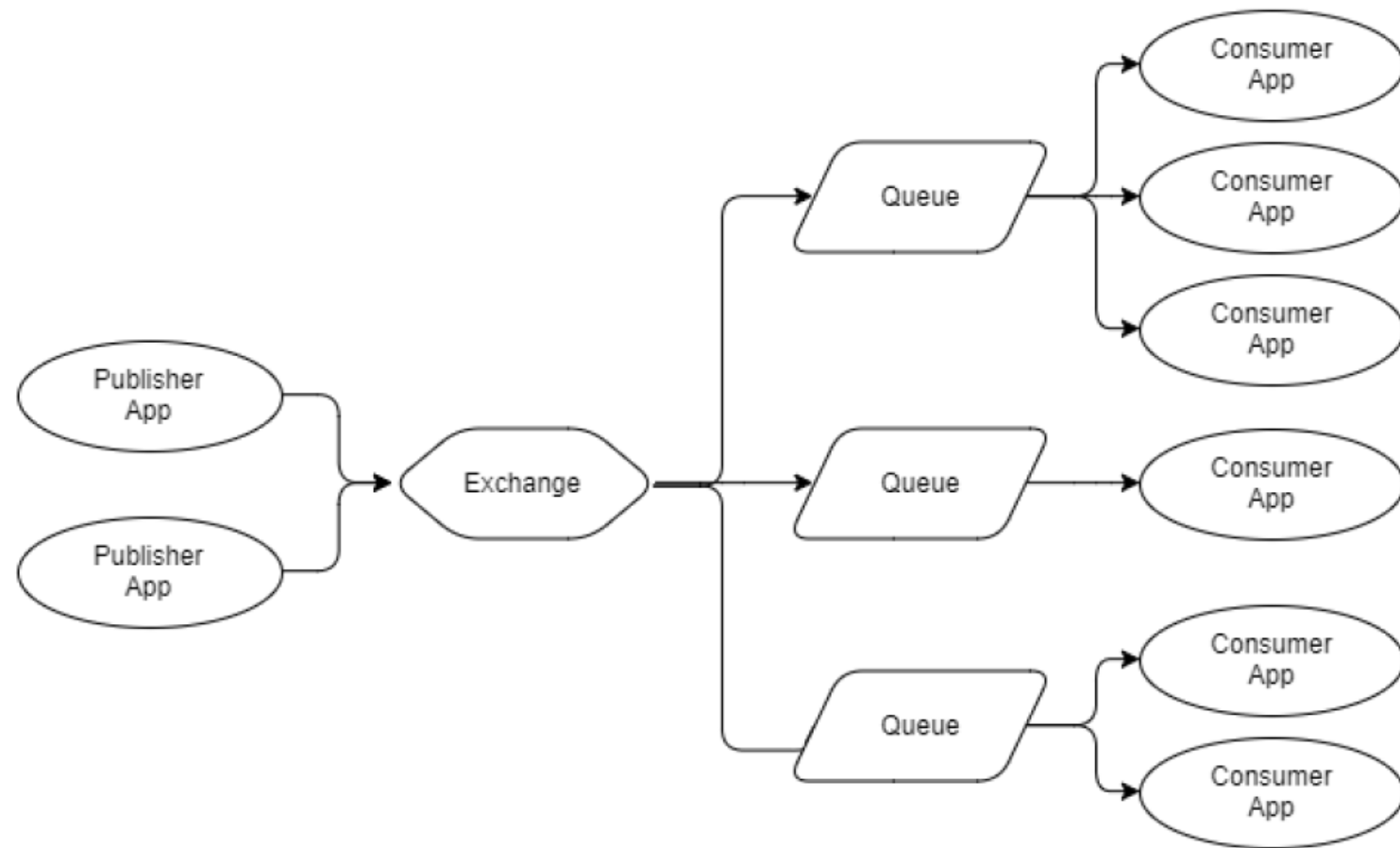
Независимые подписчики



Конкурирующие подписчики



Смешанный вариант



Некоторые типы точек обмена

- **Fanout** –разветвление. Сообщения копируются во все точки обмена и очереди, привязанные к точке обмена
- **Direct** –прямая связь. Маршрутизация по точному соответствию ключа маршрутизации (**routing_key**)
- **Topic** –рассылки. Аналог Direct с проверкой на частичное соответствие

...

Флаги очередей

- **auto_delete**—автоматически удалять, если нет консьюмеров
- **durable**—сообщения не теряются при рестарте RabbitMQ
- **exclusive**—не может быть более одного консьюмера
- **passive**—не будет создаваться автоматически, если её нет
- **internal**—очередь между точками обмена

Память

- По умолчанию, если RabbitMQ начинает использовать больше **40%** от общего объёма памяти, то все соединения блокируются
- Нормальный процесс работы возобновляется после освобождения памяти

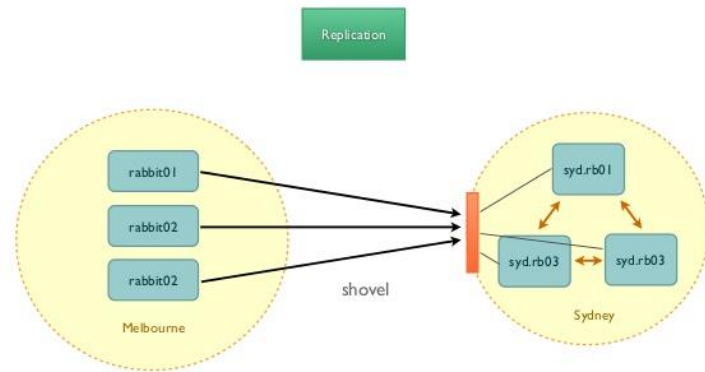
- Переопределение порога:

```
rabbitmqctl set_vm_memory_high_watermark  
0.5
```

или в

```
/etc/rabbitmq/rabbitmq.config  
[{rabbit, [{vm_memory_high_watermark,  
0.5}]}]}
```

Distributed RabbitMQ



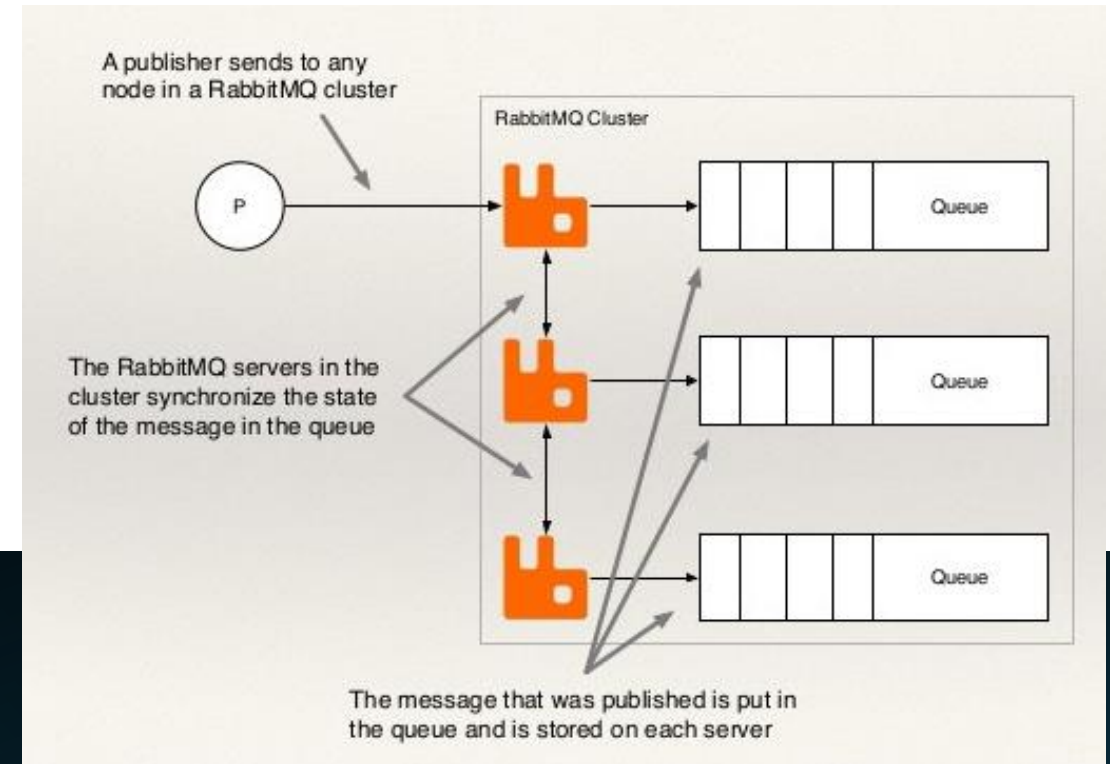
Idea is to have local rabbitmq on web nodes re-publishing messages across WAN to a cluster.

Shovel

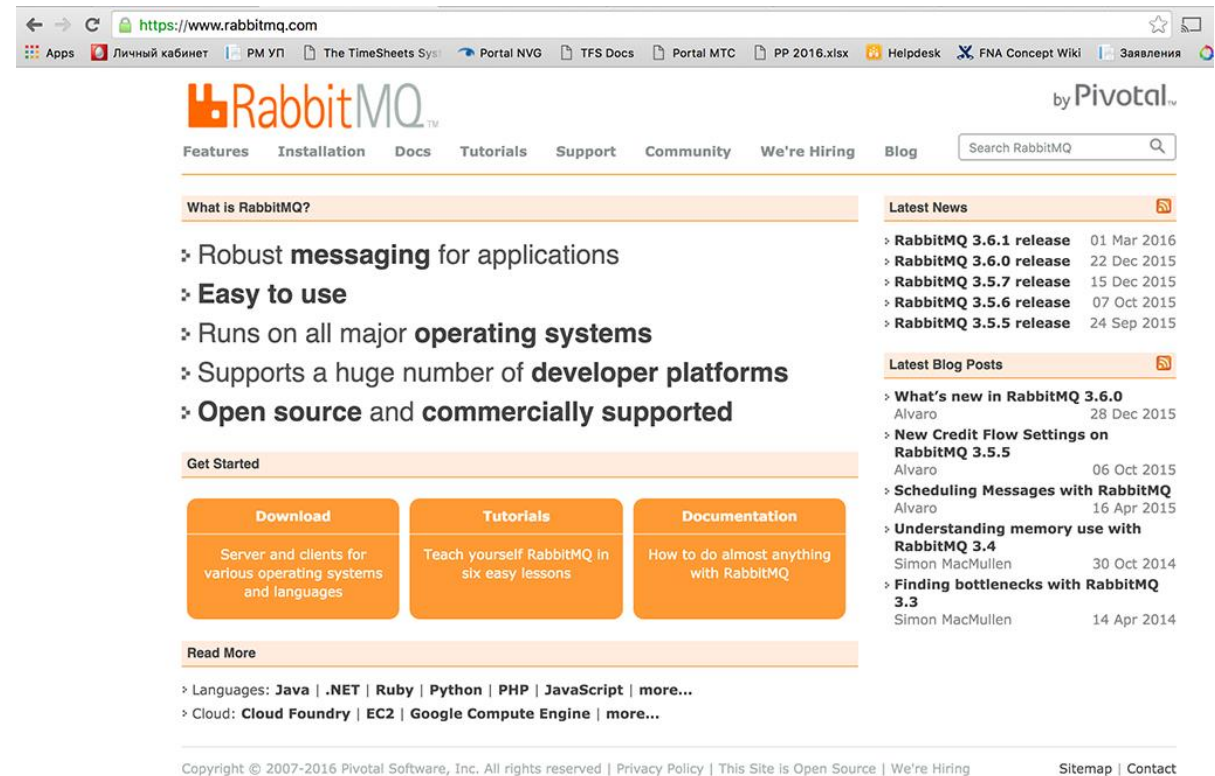
- Основная задача – «транспорт» между серверами очередей
- Нет особых требований к соединению между серверами
- Гарантированная доставка
- <http://www.rabbitmq.com/distributed.html>
- Настройка Shovel
 - <http://www.rabbitmq.com/shovel.html>
 - <http://www.codeproject.com/Articles/309786/Rabbit-Mq-Shovel-Example>

Cluster

<http://www.rabbitmq.com/clustering.html>



Скачиваем с
<http://rabbitmq.com/>



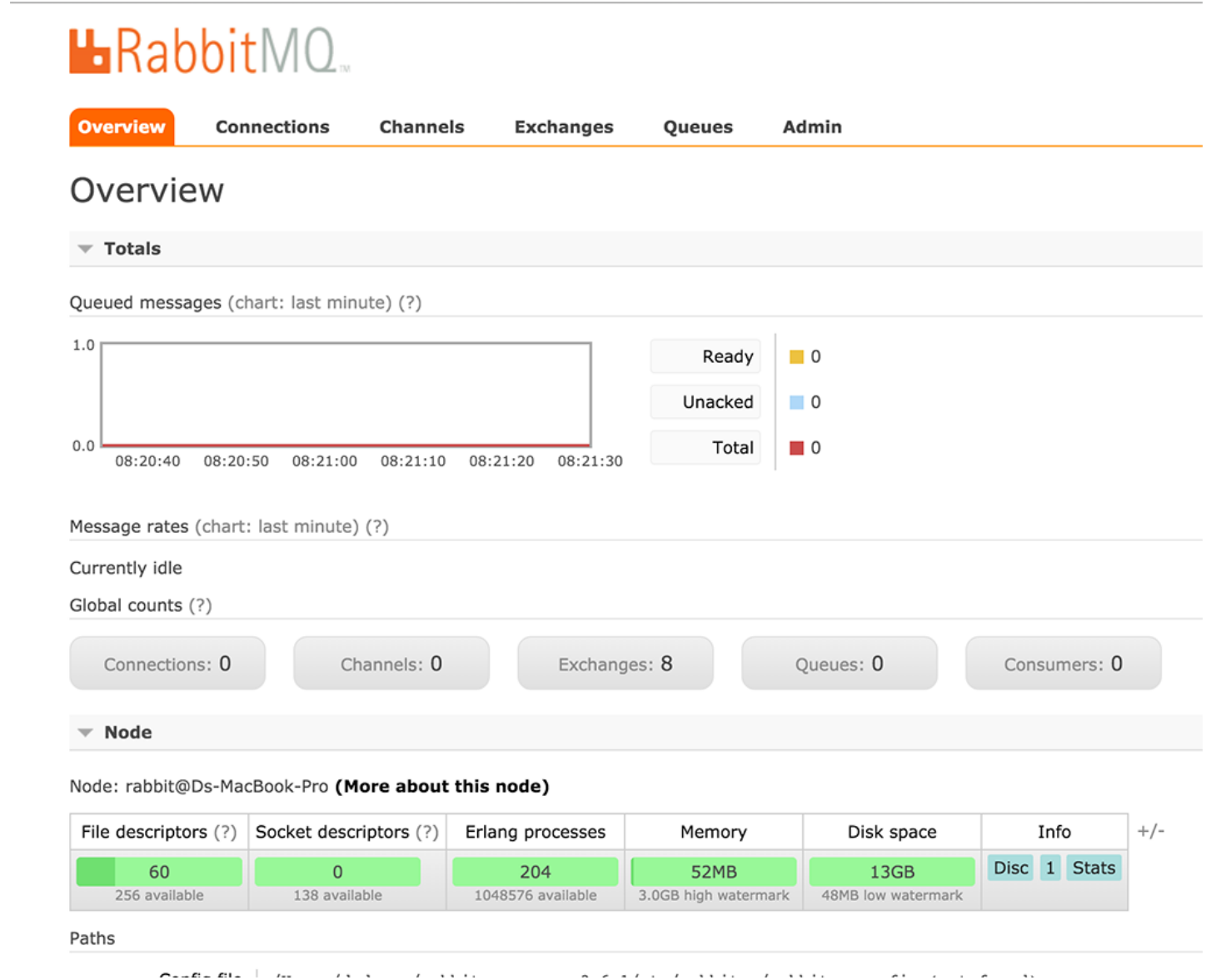
The screenshot shows the RabbitMQ website homepage. The browser's address bar displays <https://www.rabbitmq.com>. The website features a navigation bar with links to Features, Installation, Docs, Tutorials, Support, Community, We're Hiring, and Blog. A search bar is located on the right side of the navigation bar. The main content area is divided into several sections:

- What is RabbitMQ?**: A section with five bullet points highlighting key features:
 - Robust **messaging** for applications
 - Easy to use**
 - Runs on all major **operating systems**
 - Supports a huge number of **developer platforms**
 - Open source** and **commercially supported**
- Get Started**: A section with three orange buttons:
 - Download**: Server and clients for various operating systems and languages
 - Tutorials**: Teach yourself RabbitMQ in six easy lessons
 - Documentation**: How to do almost anything with RabbitMQ
- Read More**: A section with two links:
 - Languages: [Java](#) | [.NET](#) | [Ruby](#) | [Python](#) | [PHP](#) | [JavaScript](#) | [more...](#)
 - Cloud: [Cloud Foundry](#) | [EC2](#) | [Google Compute Engine](#) | [more...](#)
- Latest News**: A section with a list of recent releases and blog posts, including "RabbitMQ 3.6.1 release" (01 Mar 2016) and "What's new in RabbitMQ 3.6.0" (28 Dec 2015).

The footer contains copyright information: "Copyright © 2007-2016 Pivotal Software, Inc. All rights reserved | Privacy Policy | This Site is Open Source | We're Hiring" and links to "Sitemap | Contact".

Настраиваем rabbitmq

1. Добавляем plugin для web-администрирования
`./rabbitmq-plugins enable rabbitmq_management`
2. Запускаем сервер
`./rabbitmq-server`
3. Проверяем статус сервера
`./rabbitmqctl status`
4. Открываем web-browser
<http://localhost:15672/#/>



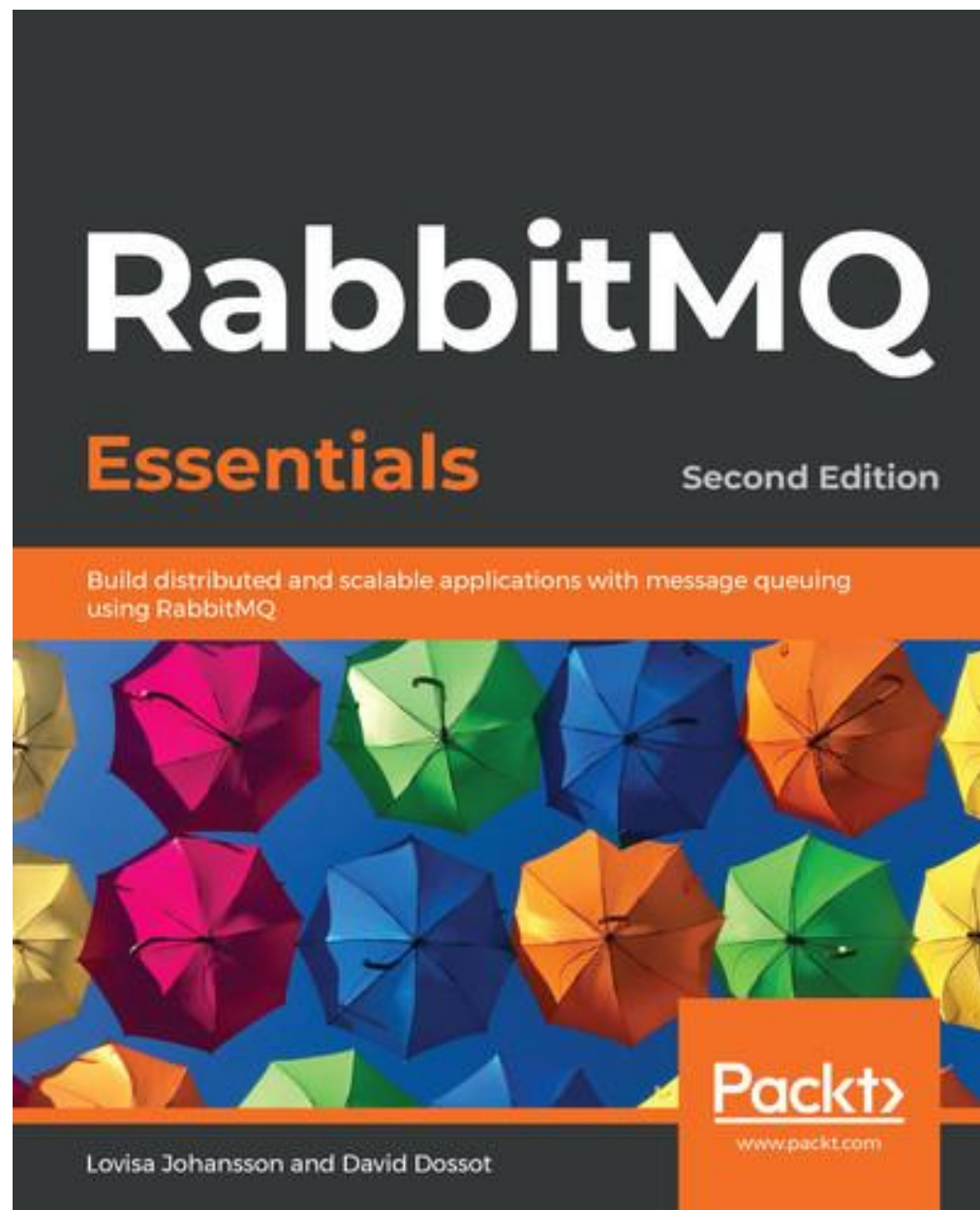


Пример

RabbitMQ итого

- Гибкая маршрутизация
- Гарантия либо at-most-once delivery, либо at-least-once-delivery, но не exactly-once-delivery
- Push сообщений
- Не гарантирует порядок доставки при параллельной обработке «из коробки»

Что
почитать?



Apache Kafka

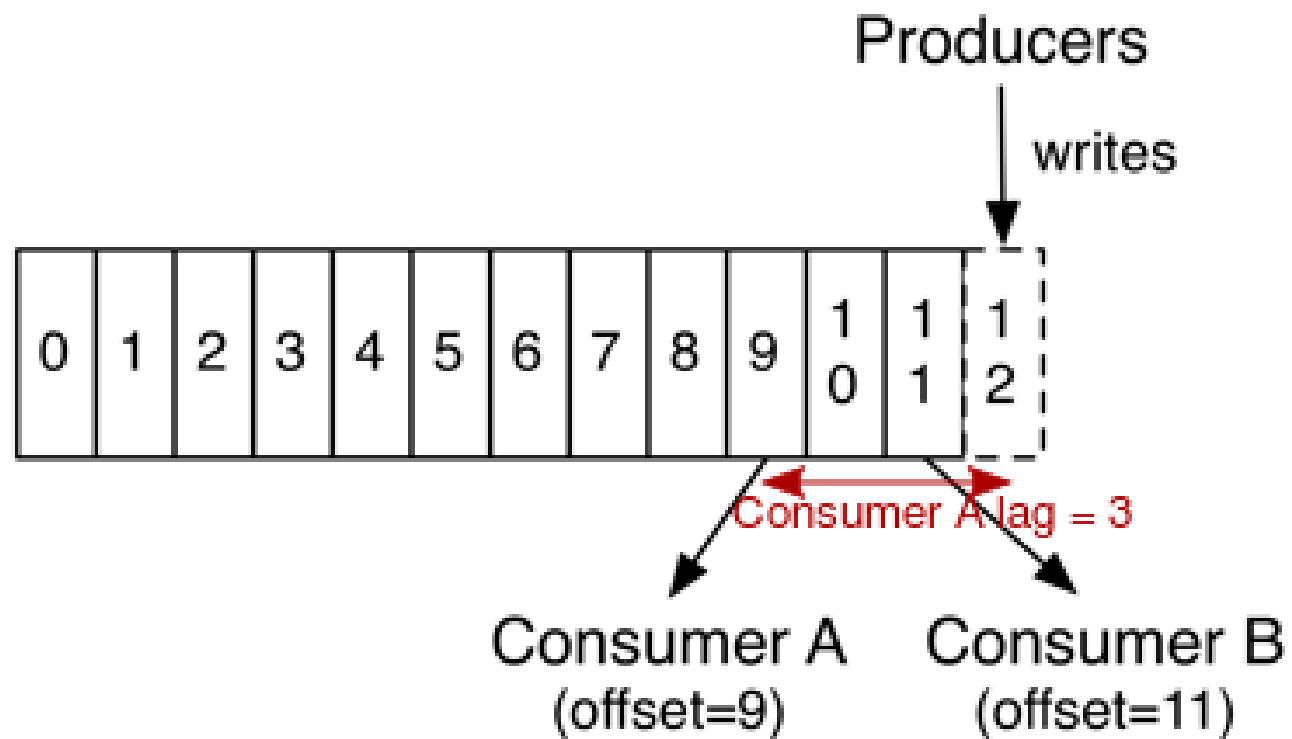
<https://kafka.apache.org/>

Особенности

- Распределённый реплицированный commit log
- Возможность «перемотки назад»
- Pull сообщений
- Гарантирует порядок доставки при параллельной обработке (но не глобально)

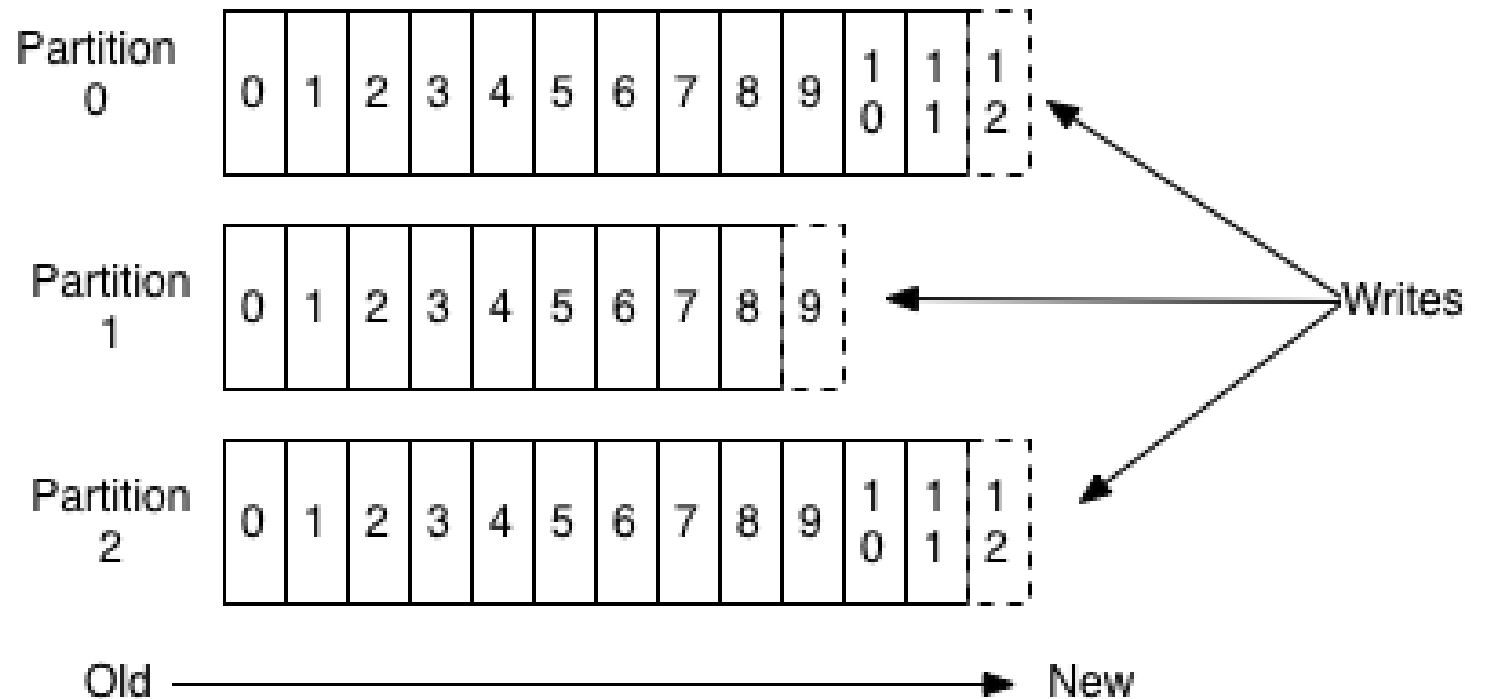
Термины

- Запись (Record)
- Поставщик (Producer)
- Подписчик (Consumer)
- Категория (Topic)
- Позиция записи/чтения (Offset)

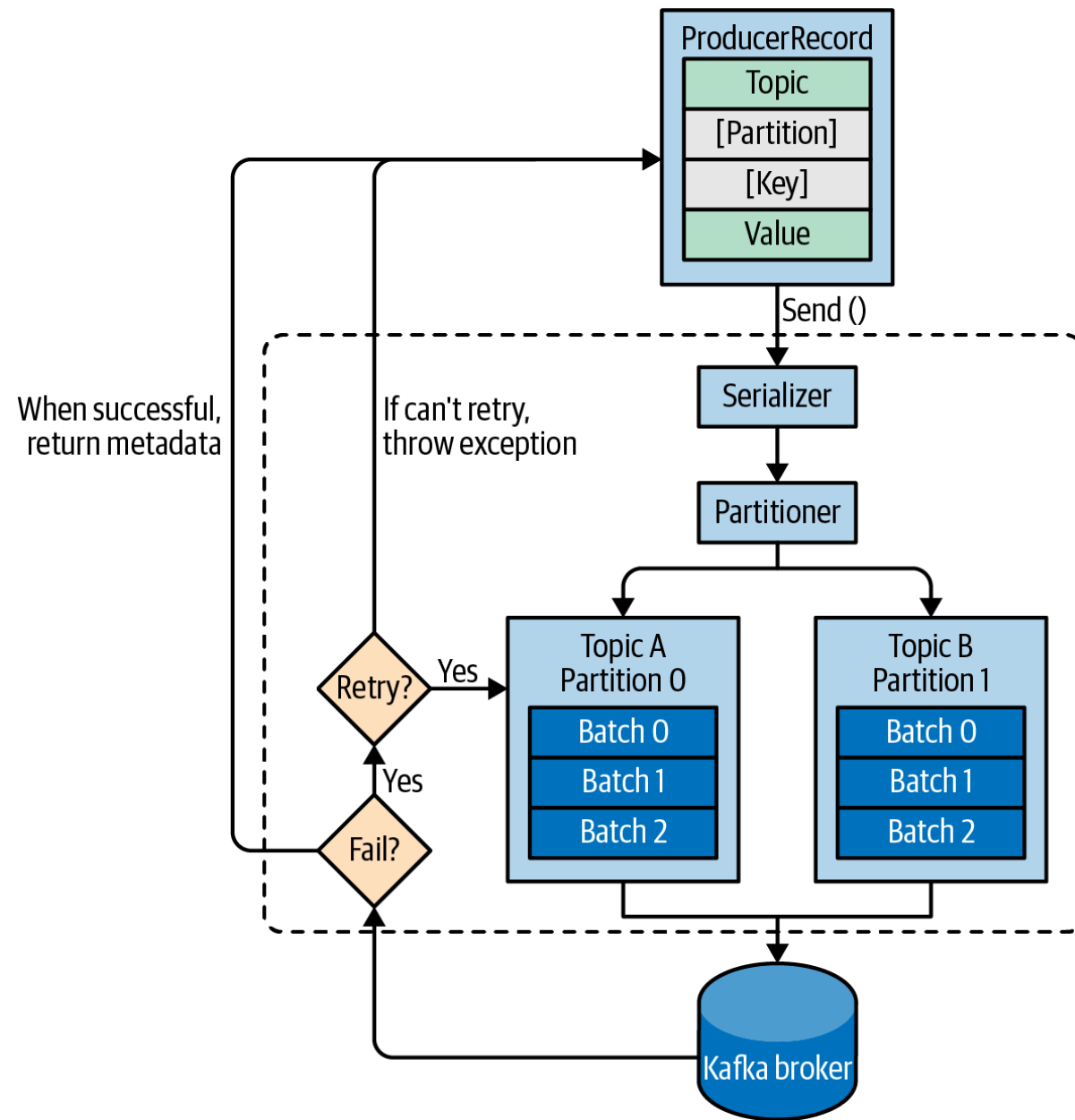


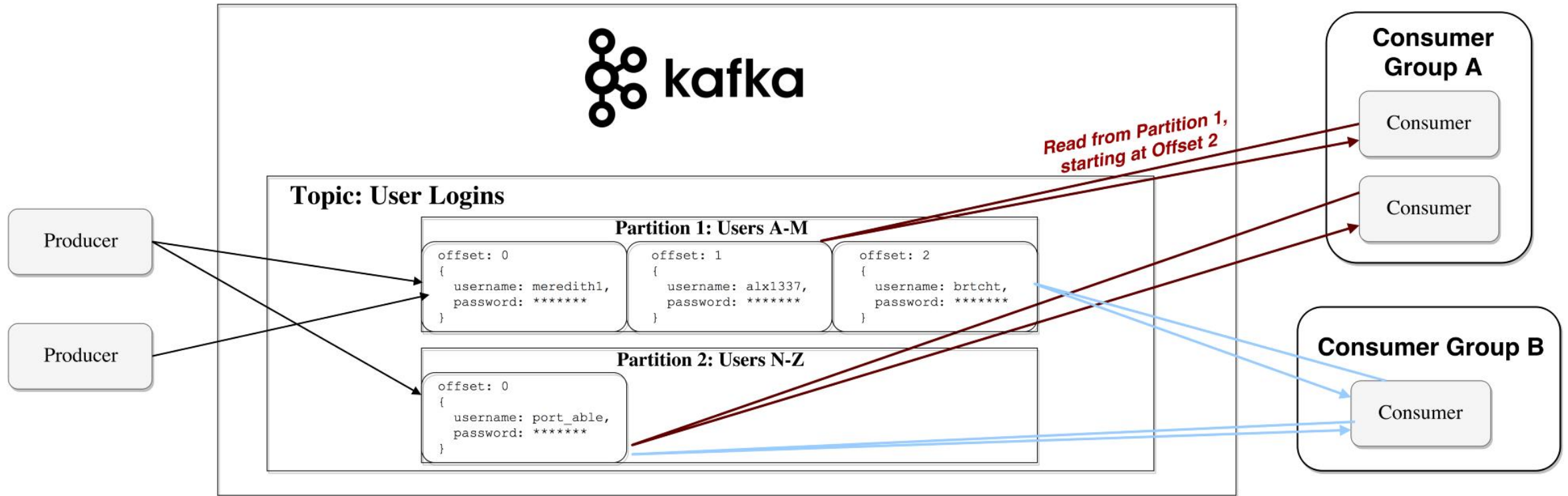
Как устроен topic

Anatomy of a Topic



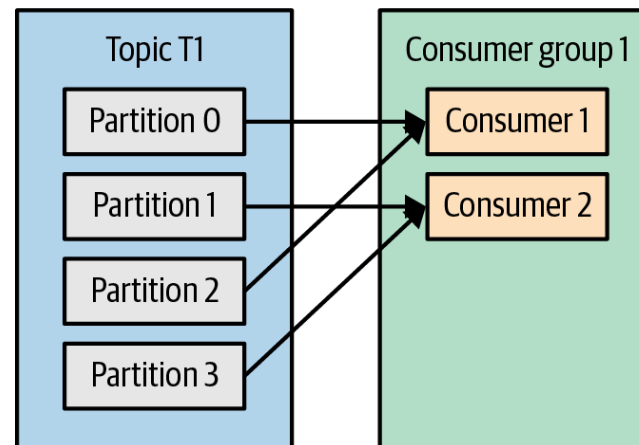
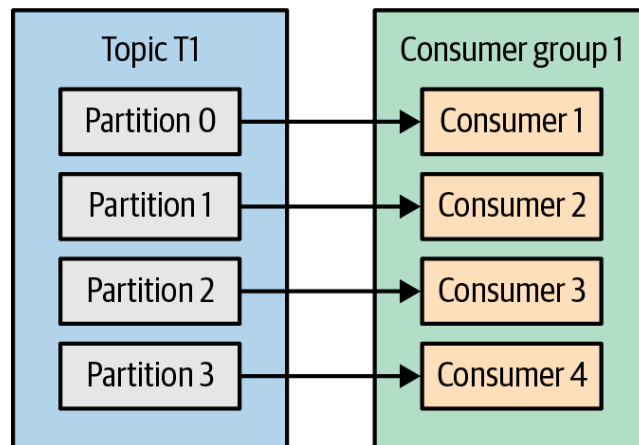
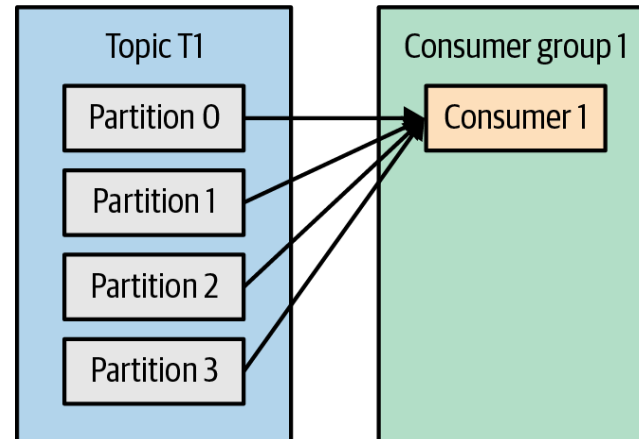
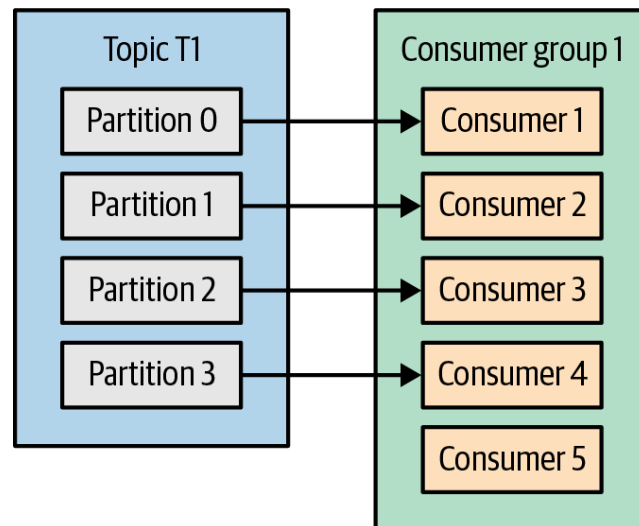
Устройство Producer

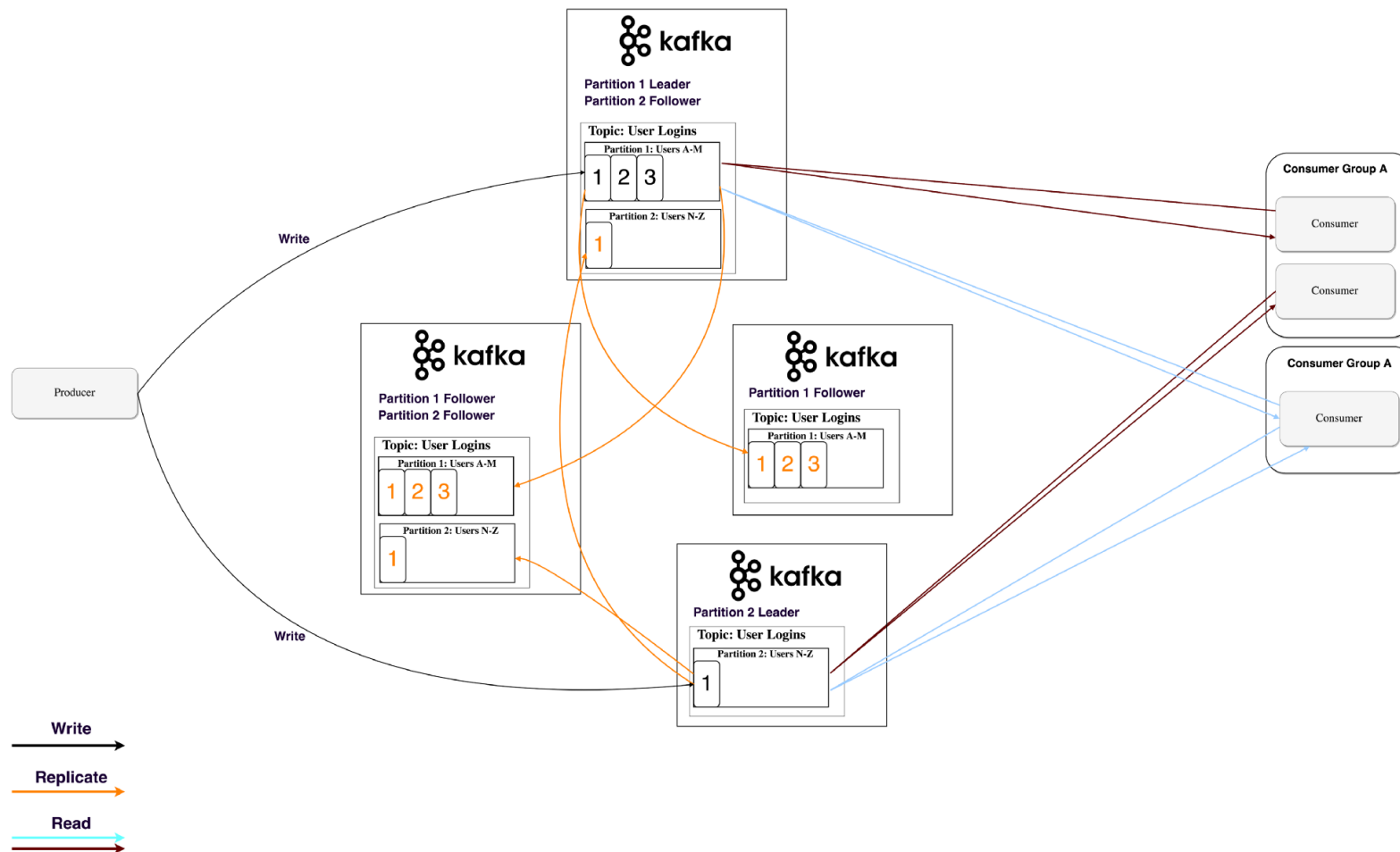




Потоки данных

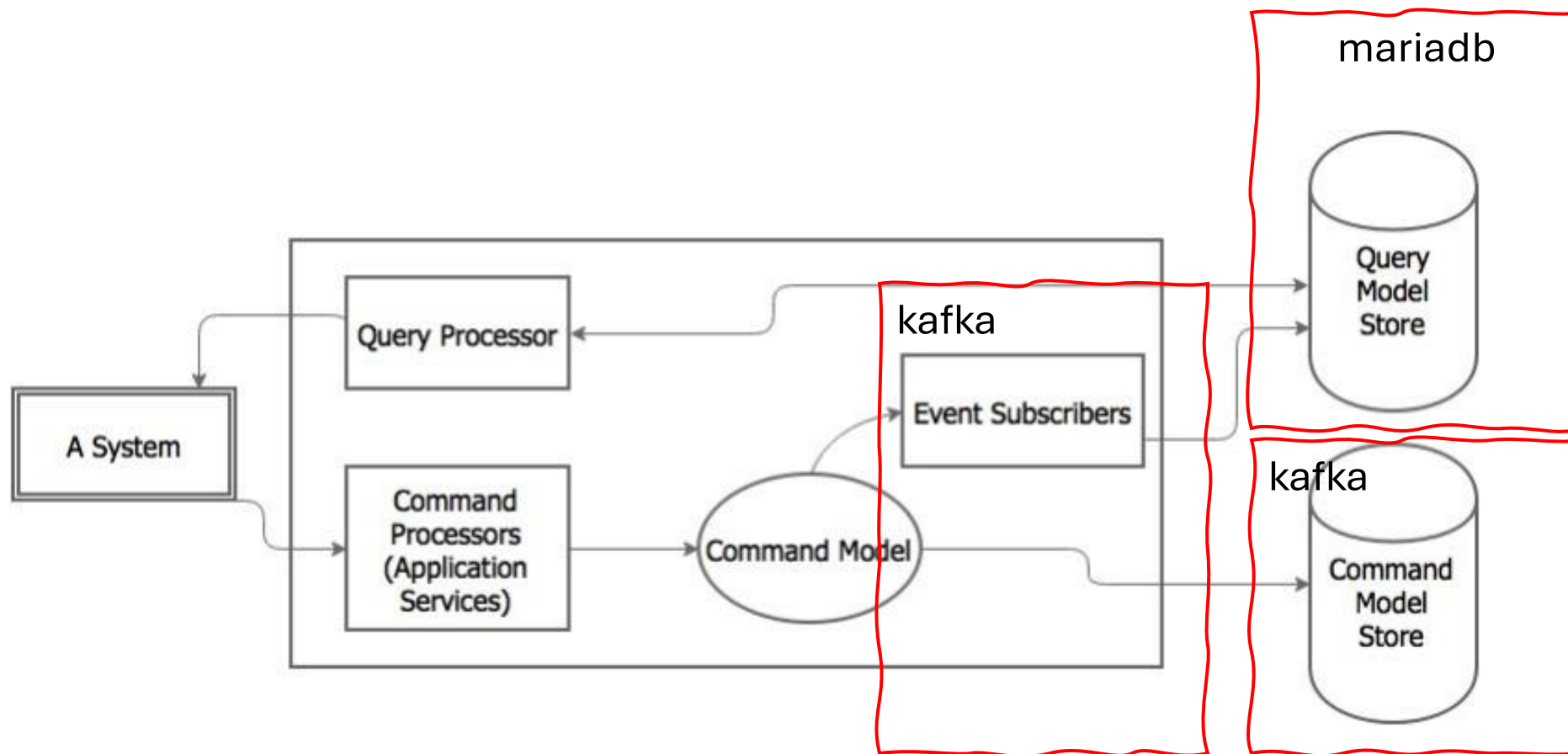
Масштабирование consumer





Репликация

Пример реализации CQRS

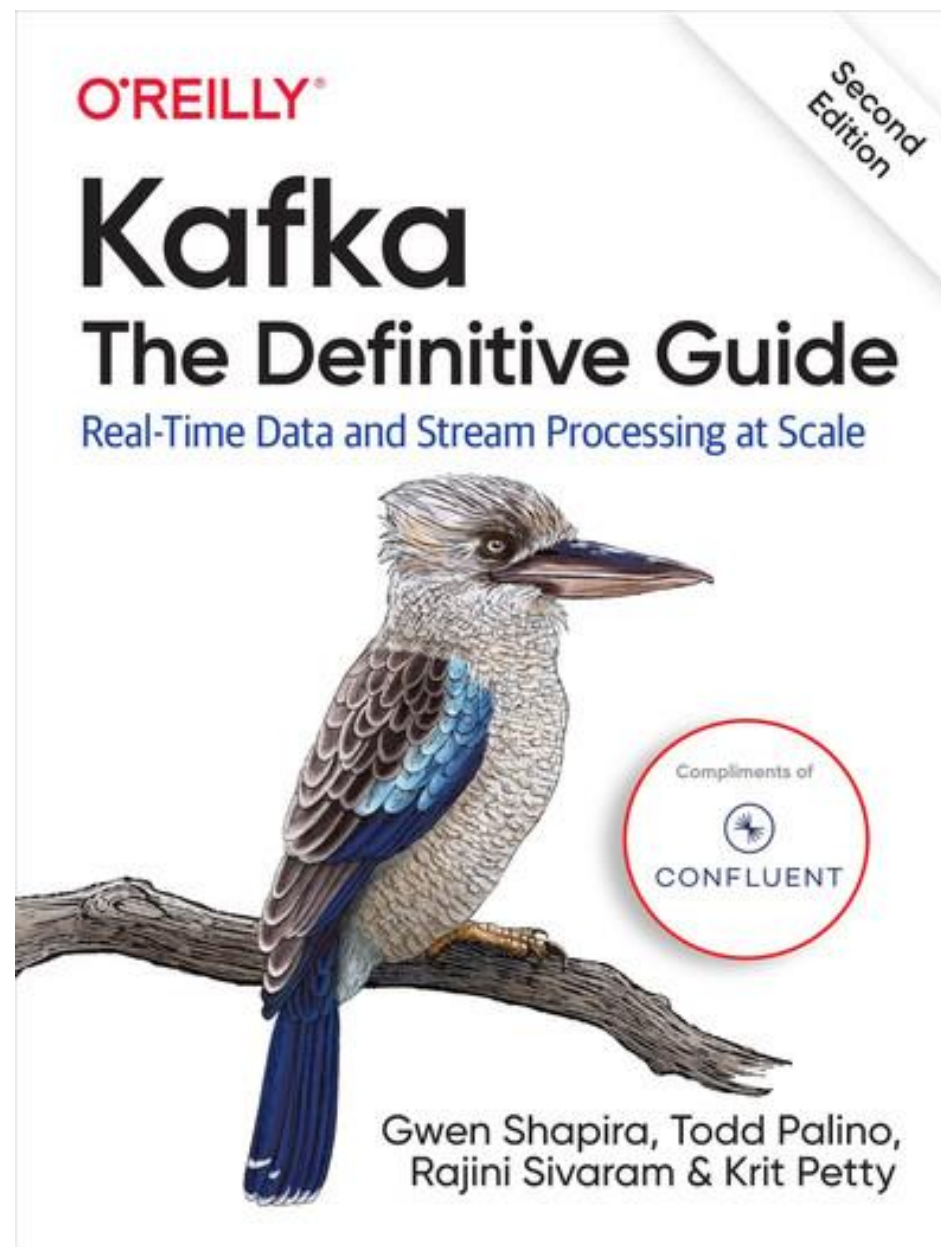


Пример

RabbitMQ vs Kafka

Критерий	RabbitMQ	Apache Kafka
Pull -модель получения сообщений (большая пропускная способность и масштабируемость)	-	+
Push -модель получения сообщений (меньшие задержки)	+	-
Гарантия доставки сообщений в определённом порядке	-	+
Подписка на сообщения с определённого времени	-	+
Повторное чтение сообщений из очереди	-	+
Гибкие возможности маршрутизации сообщений	+	-
Параллелизованная упорядоченная обработка сообщений	-	+
Возможность хранения сообщений очереди в оперативной памяти	+	-
Возможность долговременного хранения сообщений очереди	+/-	+
Поддержка открытых протоколов обмена сообщениями	+ (AMQP, MQTT)	-

Что
почитать?



На сегодня все

ddzuba@yandex.ru