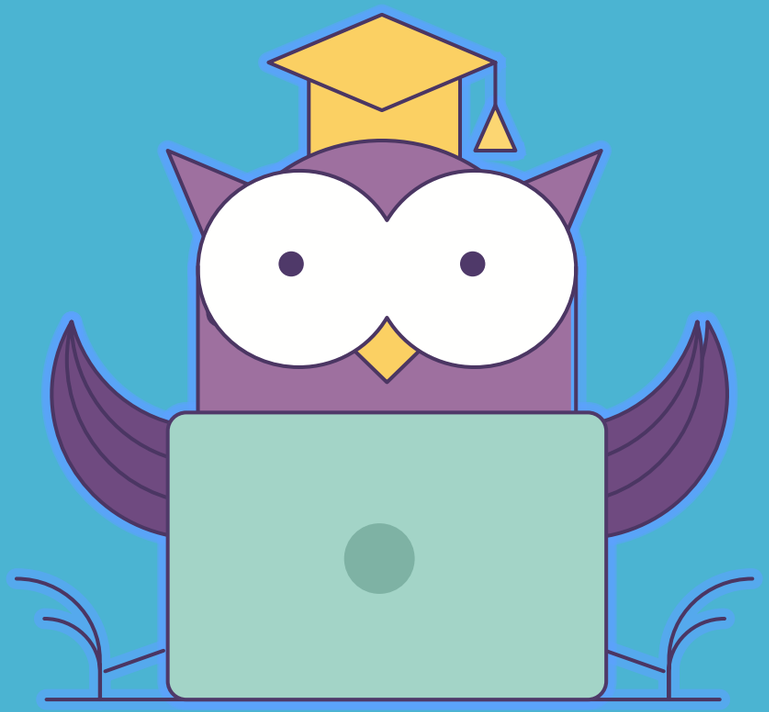
ОНЛАЙН-ОБРАЗОВАНИЕ

OTUS

Как меня слышно и видно?



> Напишите в чат

+ если все хорошо

- если есть проблемы со звуком или с видео

!проверить запись!

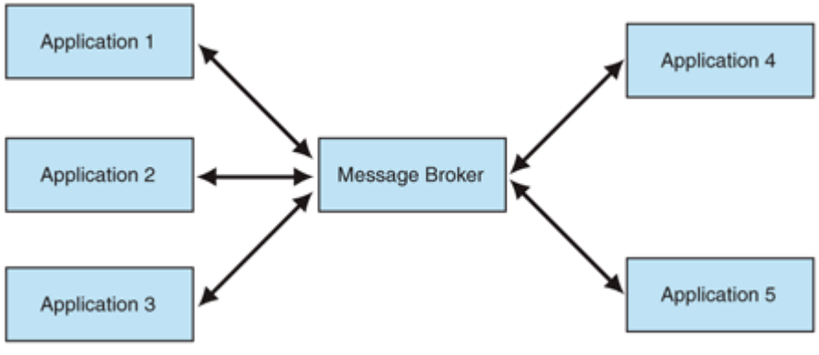
OTUS

Очереди сообщений

Алексей Романовский

План занятия

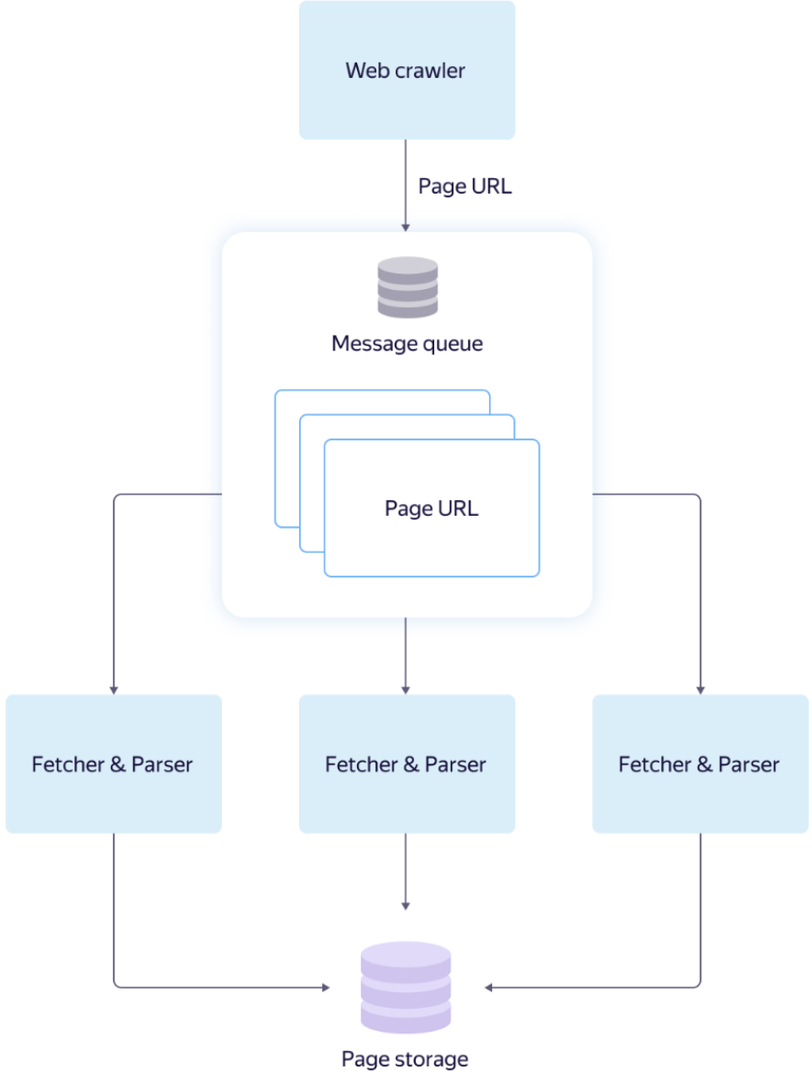
- Очереди сообщений
- Событийно-ориентированная архитектуры
- RabbitMQ
- Использование RabbitMQ
- Пару слов о Kafka



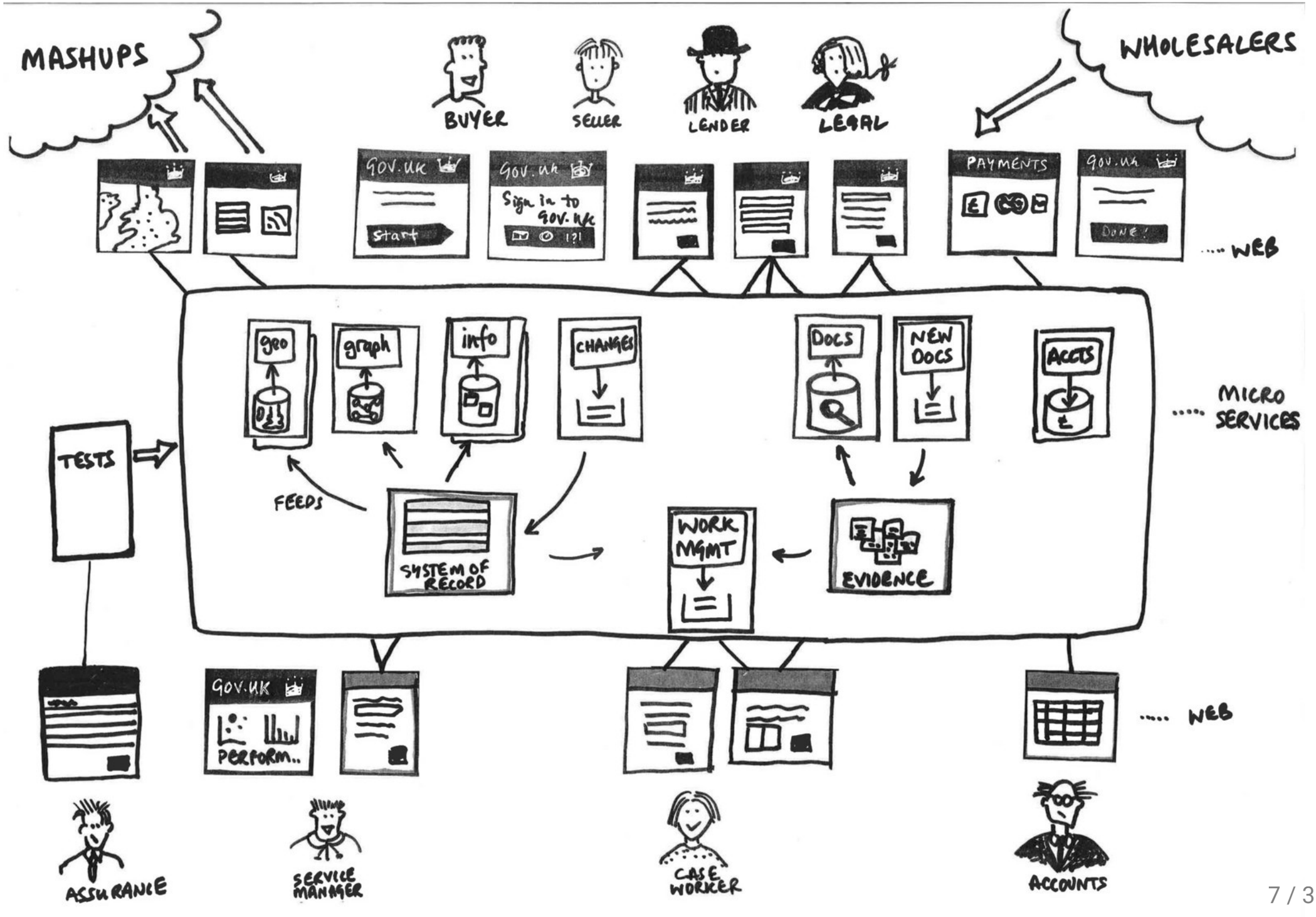
- Yandex Message Queue (https://cloud.yandex.ru/services/message-queue)
- Amazon Web Services (AWS) Simple Queue Service (SQS)
- Apache ActiveMQ
- Apache Kafka
- Redis (pubsub)
- RabbitMQ
- NATS

etc.

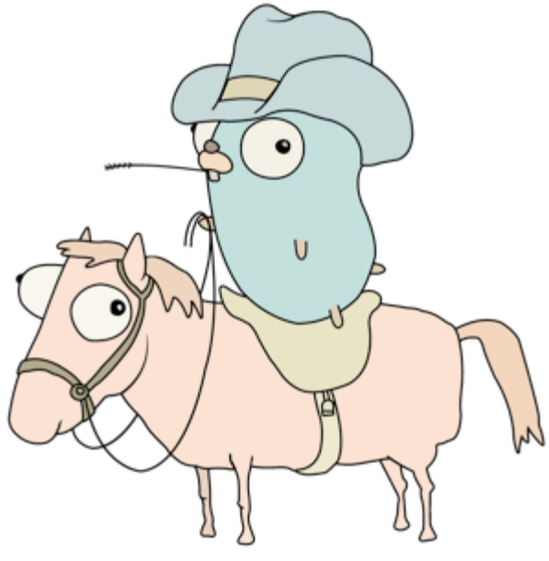
Пример: crawler (поисковый робот)

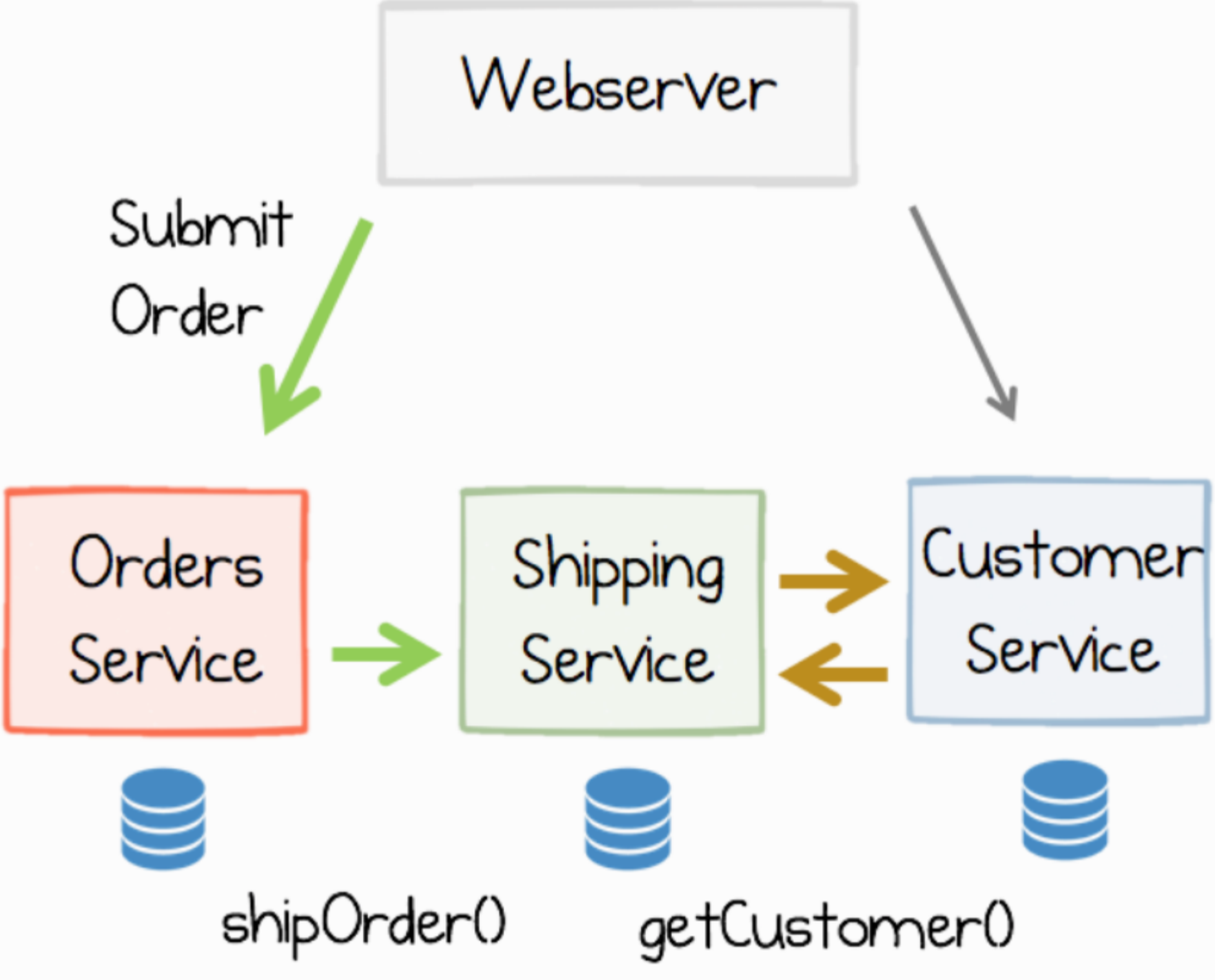


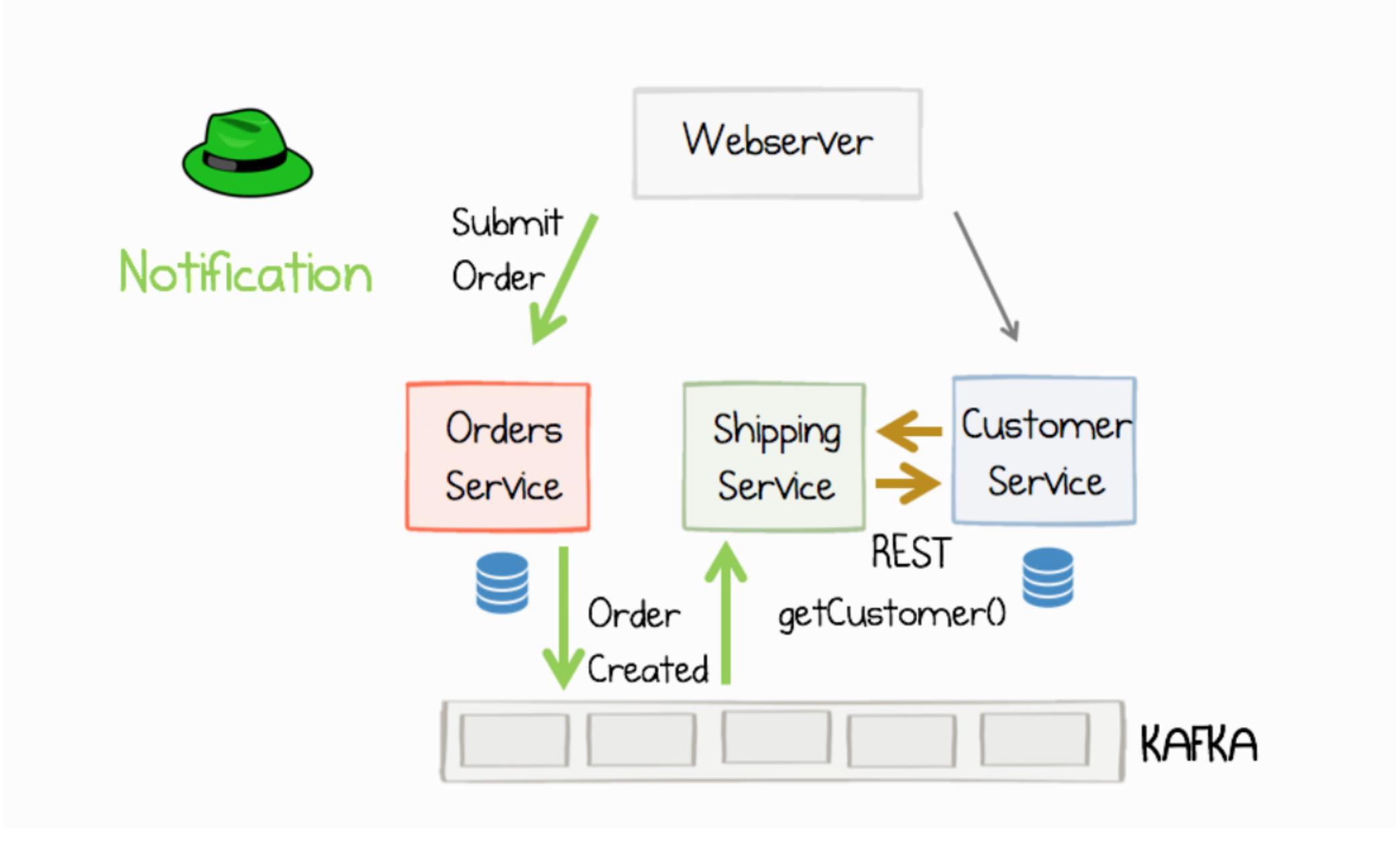
Вернемся к проблемам микросервисов...

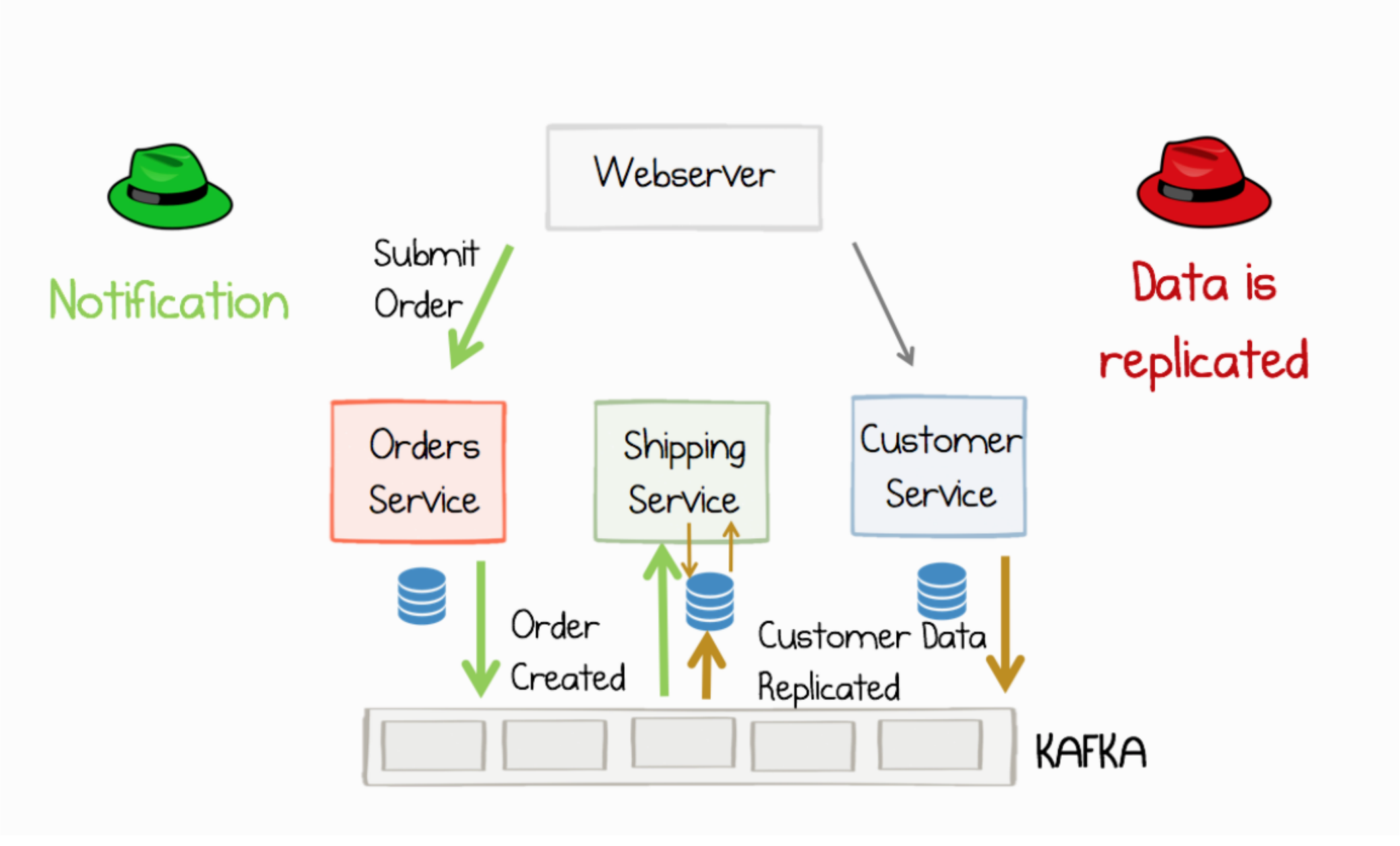


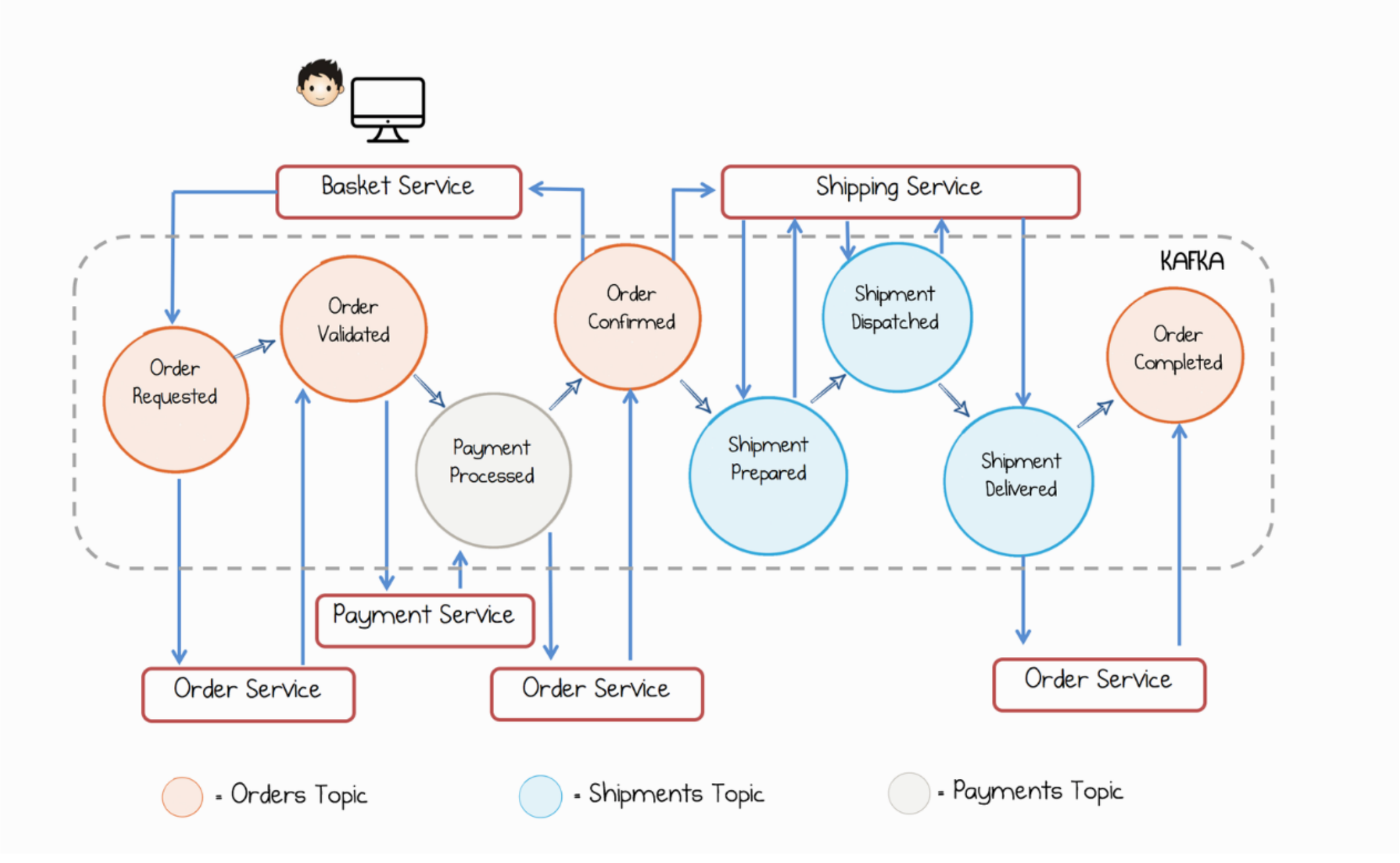
- Слабое связывание
- Масштабируемость*
- Эластичность
- Отказоустойчивость
- Гарантированная доставка*
- Гарантированный порядок доставки*
- Буферизация
- Понимание потоков данных
- Асинхронная связь

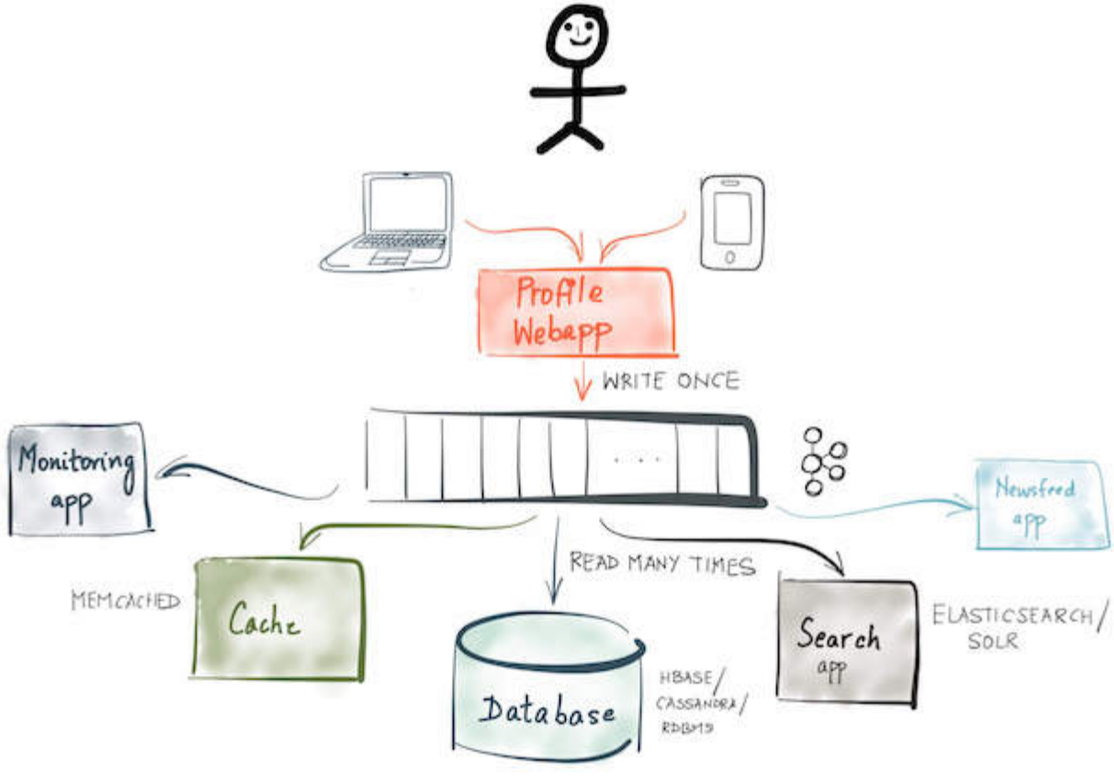


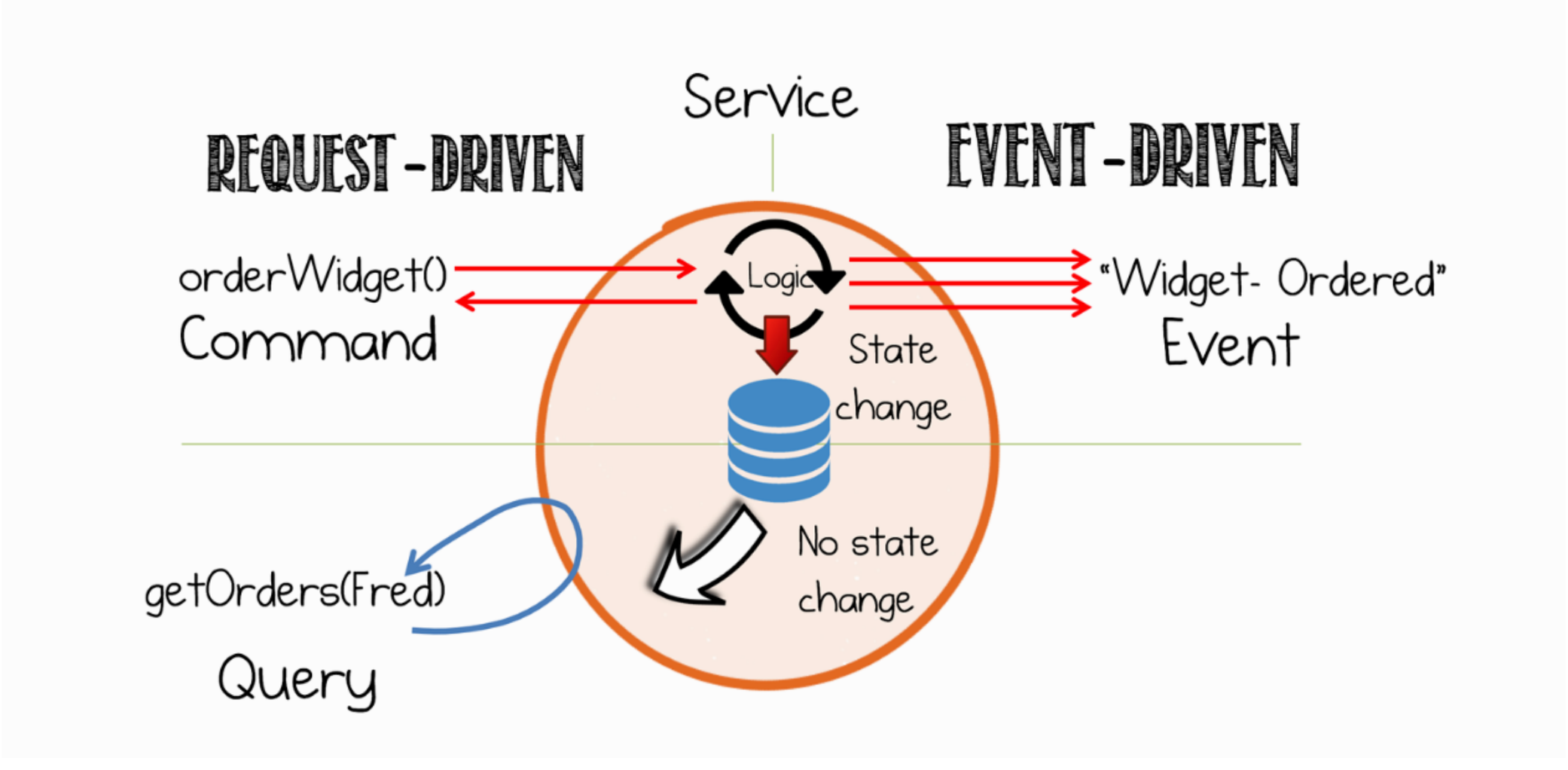

Webserver
Submit
Order
Orders
Service
Shipping
Service
Customer
Service
shipOrder()
getCustomer()


Webserver
Notification
Submit
Order
Orders
Service
Shipping
Service
Customer
Service
REST
getCustomer()
Order
Created
KAFKA

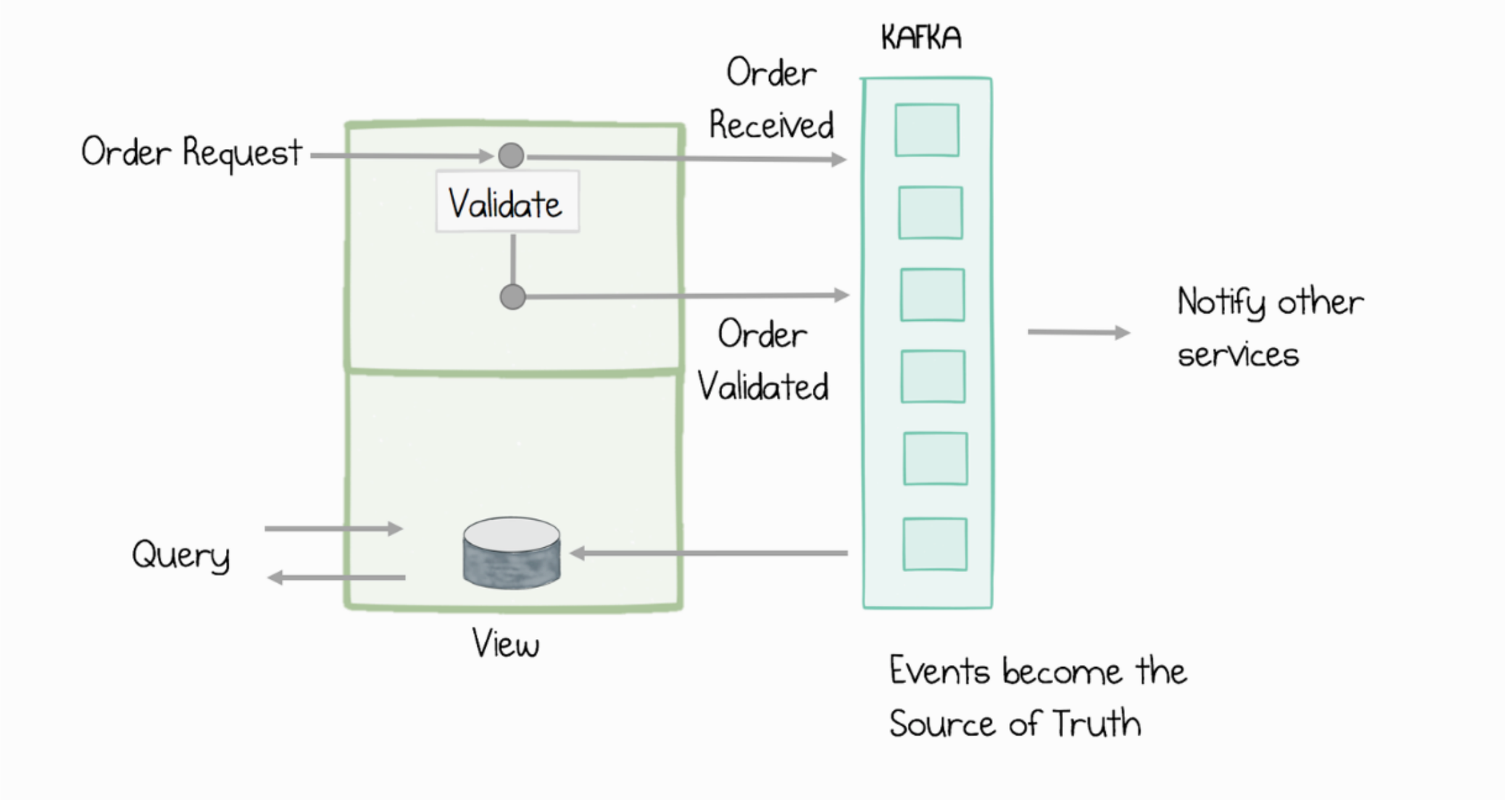

Webserver
Notification
Submit
Order
Data is
replicated
Orders
Service
Shipping
Service
Customer
Service
Order
Created
Customer Data
Replicated
KAFKA


Basket Service
Shipping Service
KAFKA
Order Requested
Order Validated
Payment Processed
Order Confirmed
Shipment Prepared
Shipment Dispatched
Shipment Delivered
Order Completed
Payment Service
Order Service
Order Service
Order Service
= Orders Topic
= Shipments Topic
= Payments Topic


Profile
Webapp
WRITE ONCE
Monitoring
app
Newsfeed
app
READ MANY TIMES
MEMCACHED
Cache
Database
HBASE/
CASSANDRA/
RDBMS
Search
app
ELASTICSEARCH/
SOLR


Service
REQUEST-DRIVEN
EVENT-DRIVEN
orderWidget()
Command
Logic
State
change
"Widget- Ordered"
Event
No state
change
getOrders(Fred)
Query

Command Query Responsibility Segregation

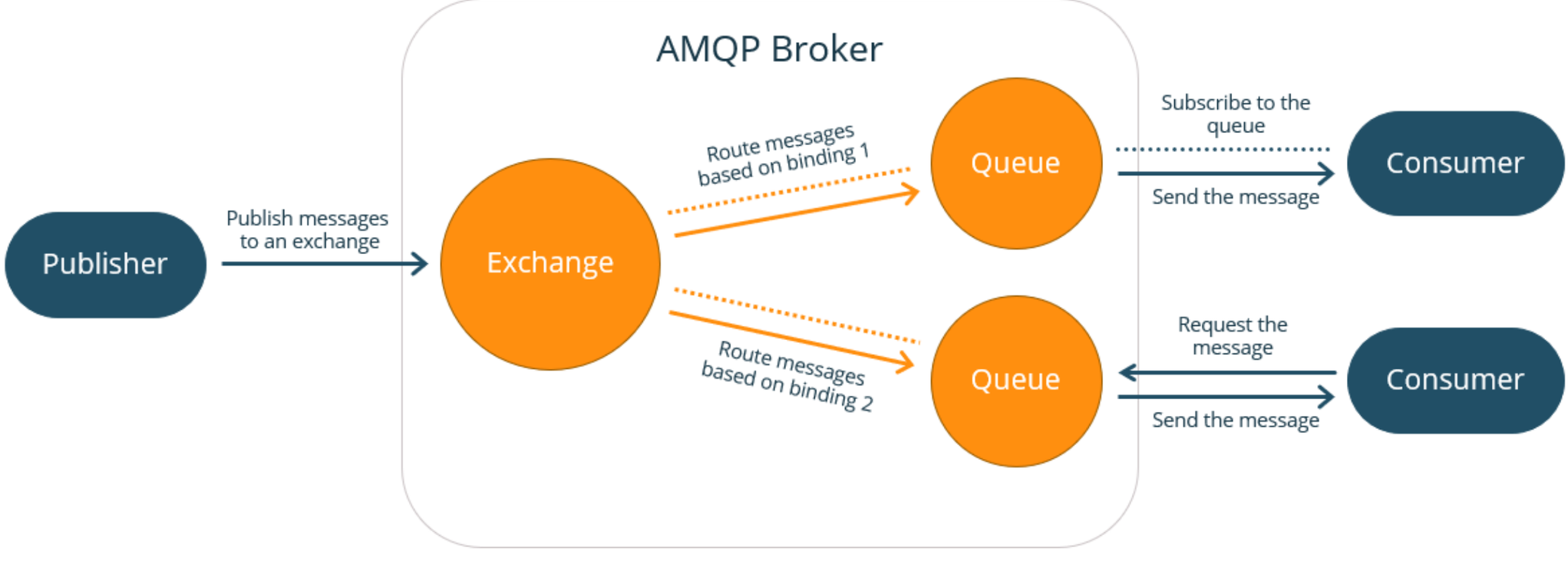


Ссылки

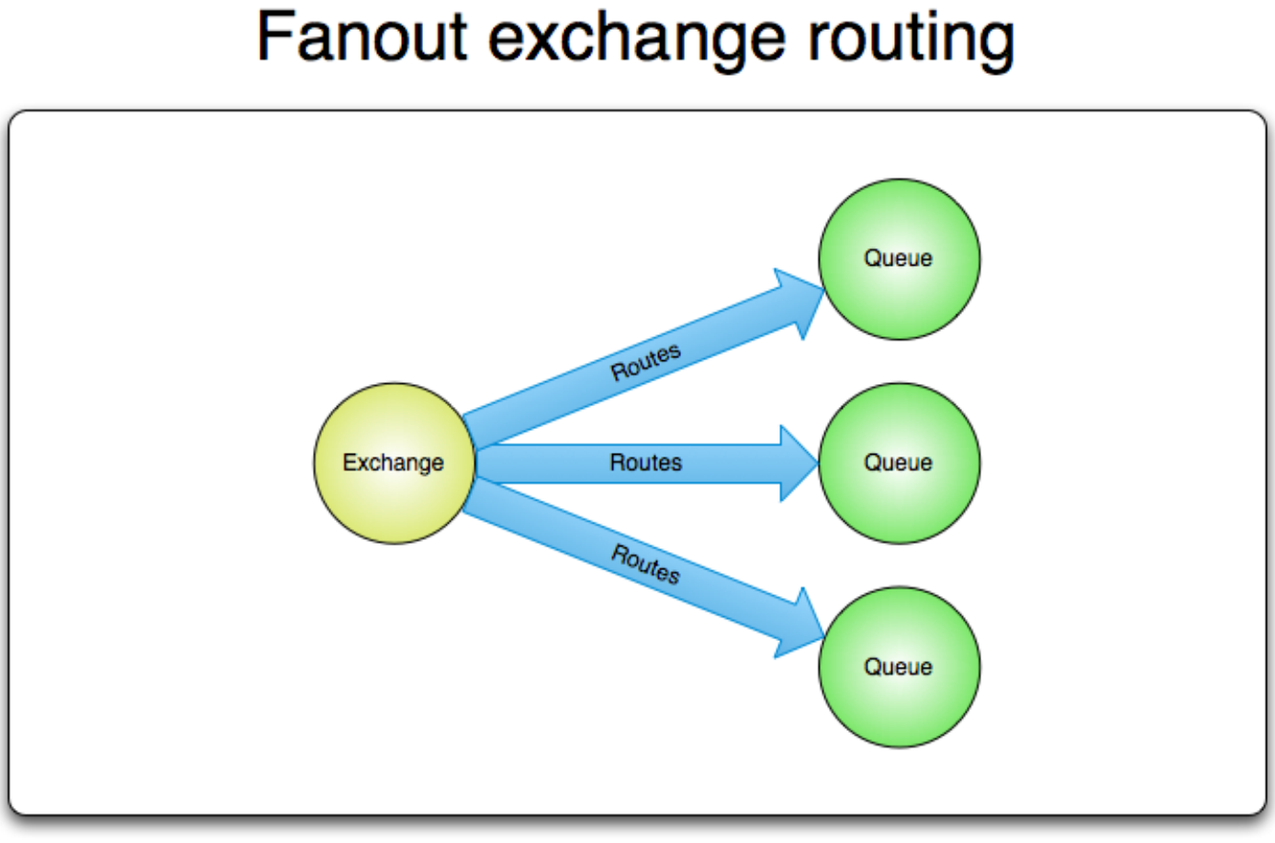
- Разработка транзакционных микросервисов с помощью Агрегатов, Event Sourcing и CQRS
  https://habr.com/ru/company/nix/blog/322214/
- Основы CQRS
  https://habr.com/ru/company/simbirsoft/blog/329970/

RabbitMQ — это распределенная система управления очередью сообщений

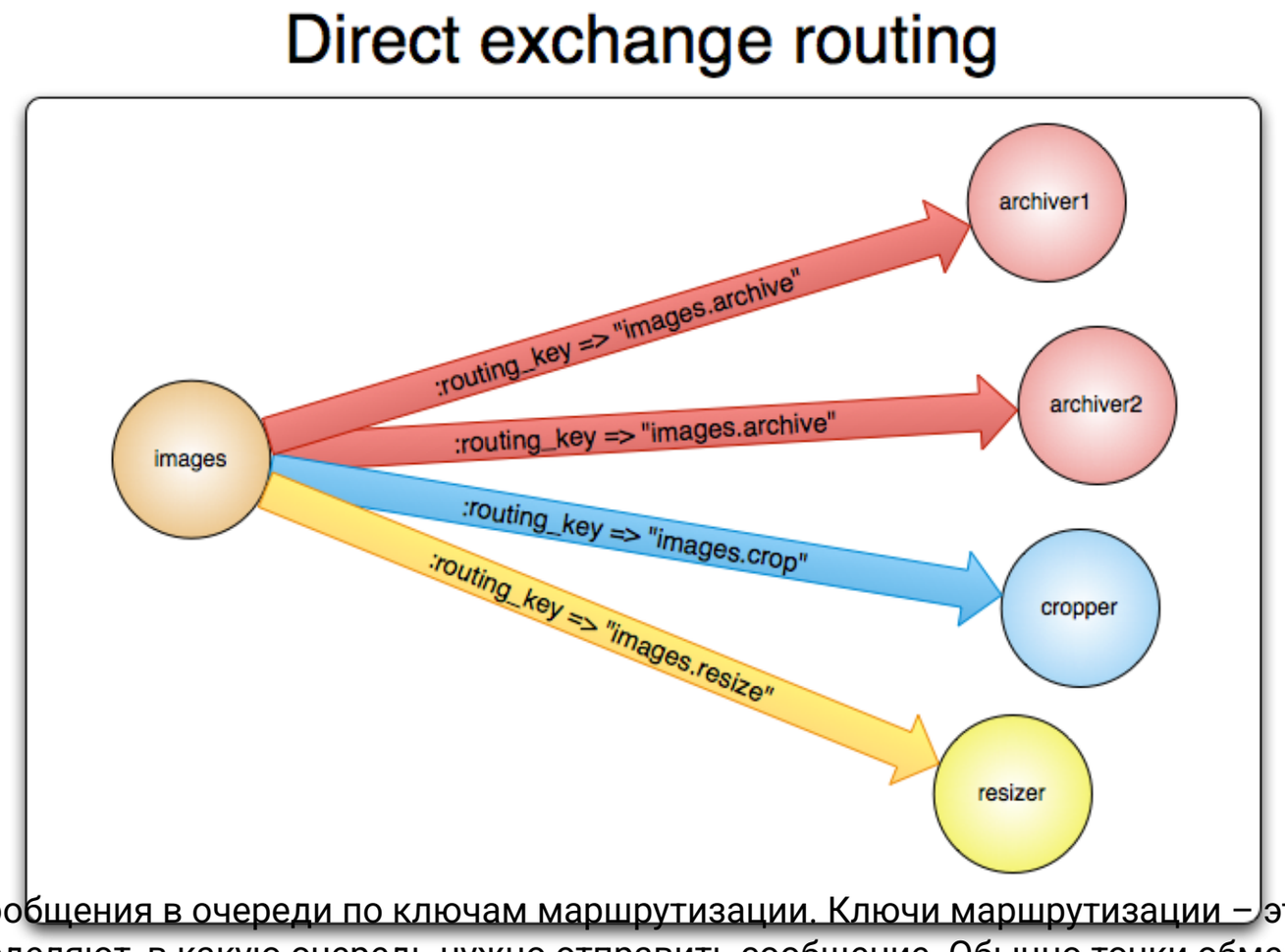
Advanced Message Queuing Protocol (AMQP)



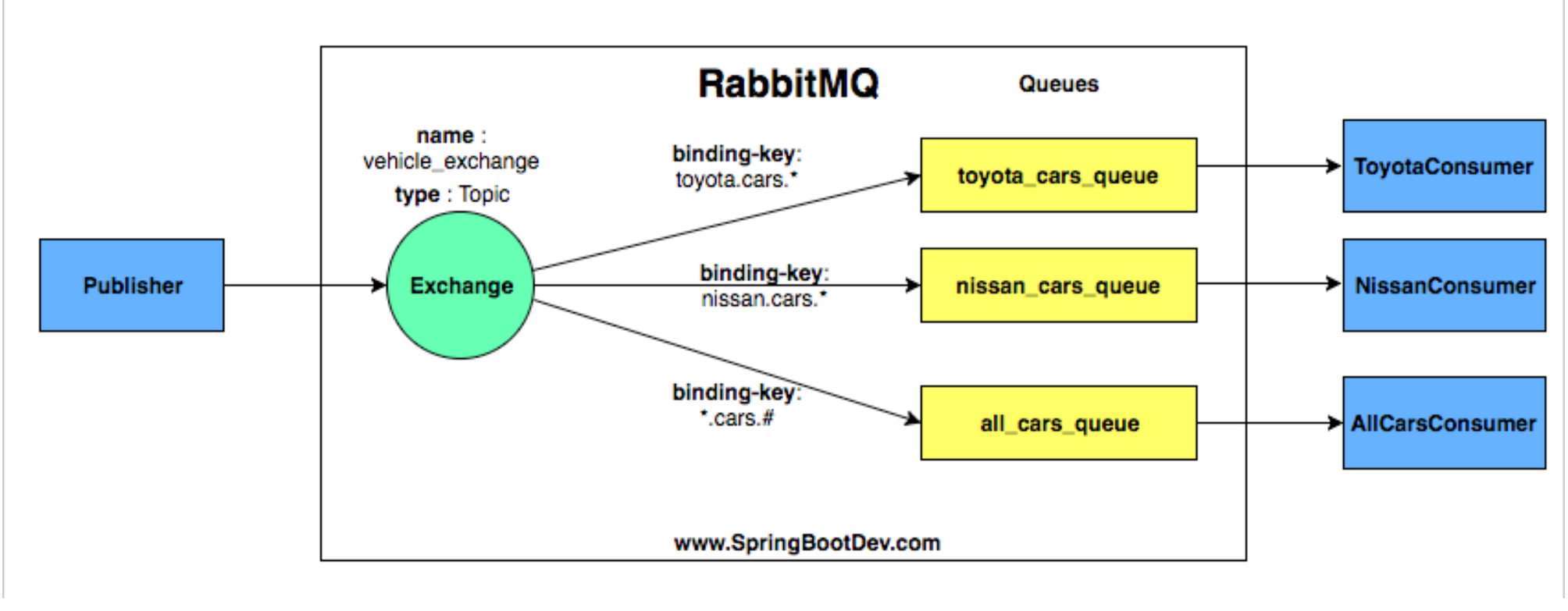
- Сообщение (**message**) — единица передаваемых данных, основная его часть (содержание) никак не интерпретируется сервером, к сообщению могут быть присоединены структурированные заголовки.
- Точка обмена (**Exchange**) — в неё отправляются сообщения. Точка обмена распределяет сообщения в одну или несколько очередей. При этом в точке обмена сообщения не хранятся.
- Очередь (**queue**) — здесь хранятся сообщения до тех пор, пока не будут забраны клиентом. Клиент всегда забирает сообщения из одной или нескольких очередей.
- Связки (**bindings**) - правила для роутинга сообщений



Fanout – полностью игнорирует ключи маршрутизации и отправляет сообщения во все привязанные очереди. Точки обмена этого типа используются для распространения сообщений нескольким клиентам (рассылки уведомлений, обновлений, конфигураций и т.п.).



Direct – доставляет сообщения в очереди по ключам маршрутизации. Ключи маршрутизации – это дополнительные данные, которые определяют, в какую очередь нужно отправить сообщение. Обычно точки обмена такого типа используются в балансировке нагрузки round-robin.



Topic – используется в шаблонах pub/sub. В этом случае ключ маршрутизации используется вместе с привязкой очередей к точке обмена. например, app.notification.sms.# — в очередь будут доставлены все сообщения, отправленные с ключами, начинающимися с app.notification.sms.

Поиграемся с админкой

```
$ docker run -d --name rb -p 15672:15672 -p 5672:5672 rabbitmq:3-management
http://localhost:15672/ guest:guest
```

Посмотрим на код

Краткие выводы

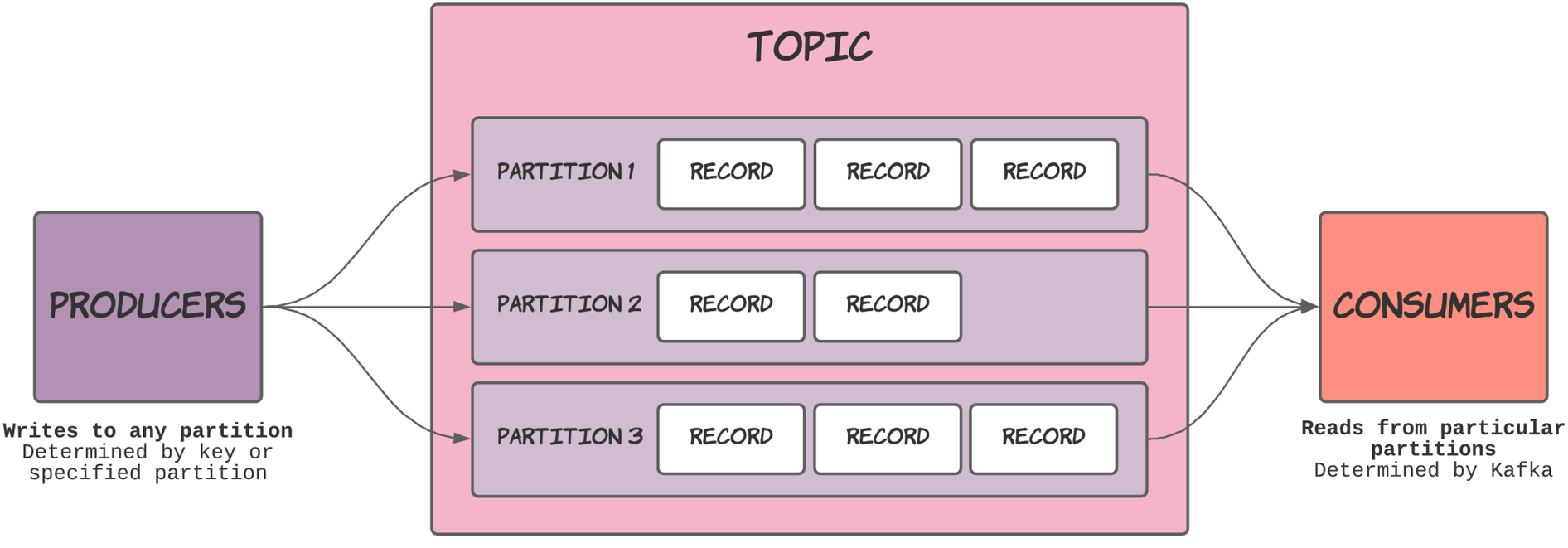
- используем каналы и concurrency паттерны
- не забываем про backoff
- реализуем код исходя из особенностей конкретного брокера

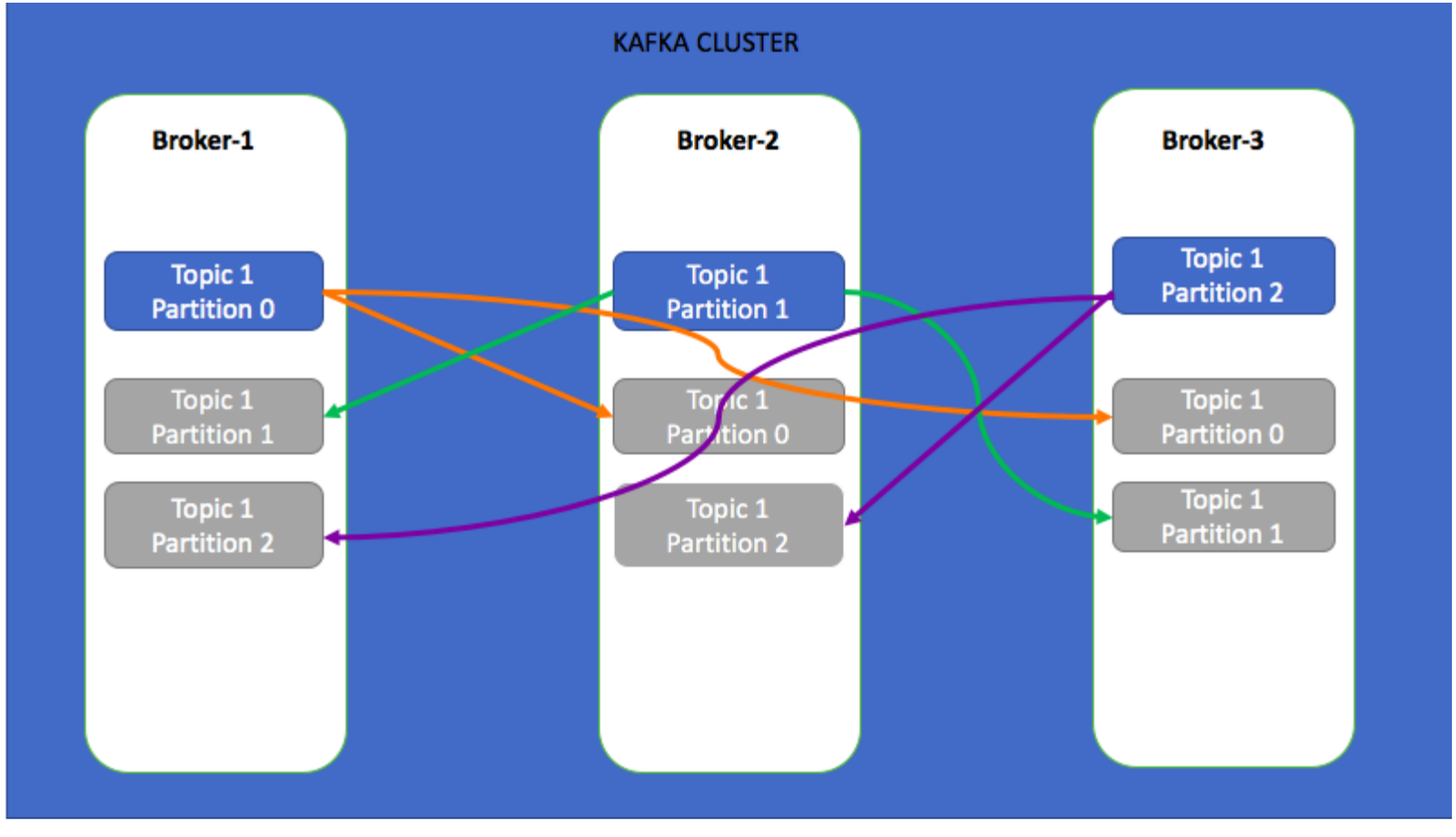
Что ещё из популярного? Kafka!

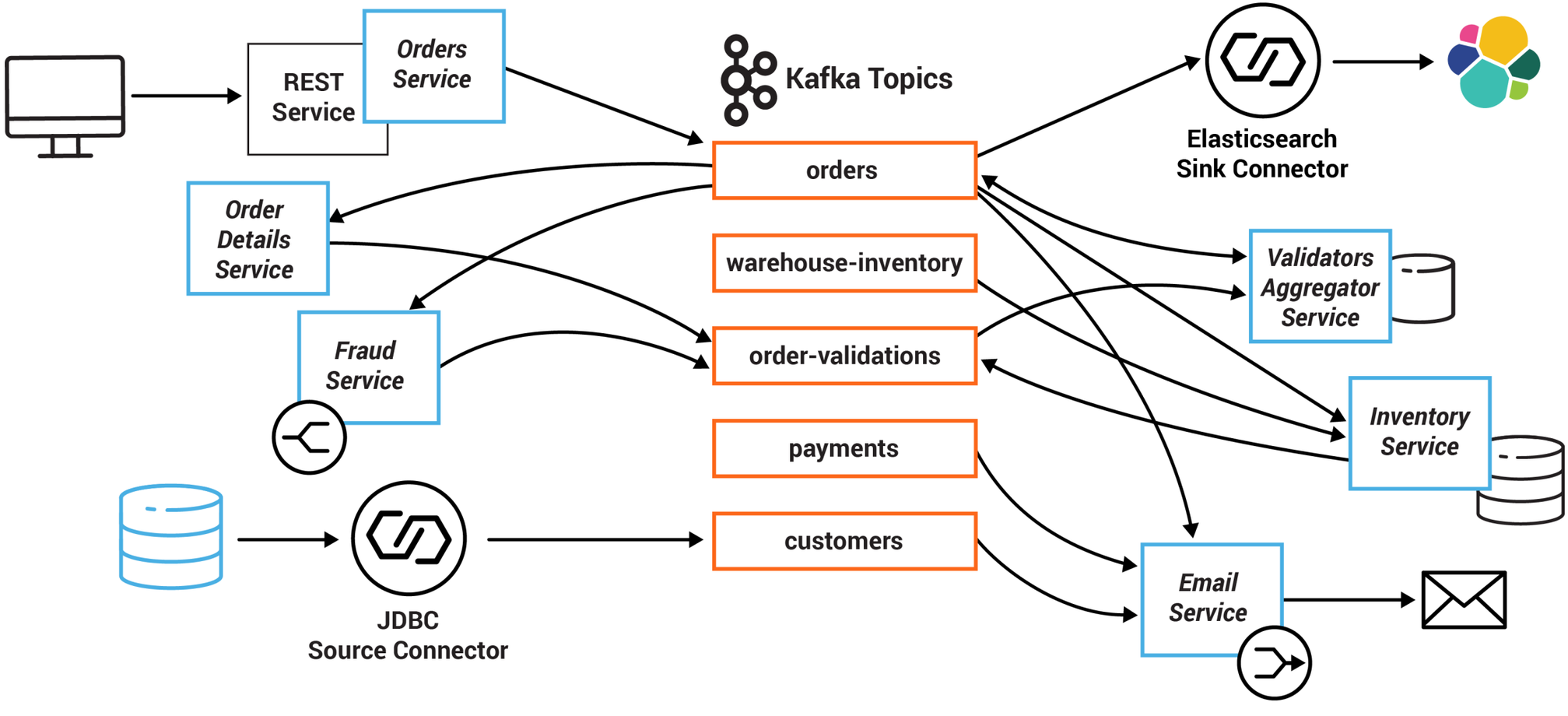
- Распределённый программный брокер сообщений
- Написан на Java/Scala
- Придуман в Linkedin чтобы обрабатывать безумный объем данных
- Есть коммерческая поддержка (Confluent)
- Линейно масштабируемый
- С гаринтией упорядоченности
- После чтения сообщения не удаляются
- Надежный (репликация)
- Высокодоступный (high availability)

Kafka: ключевые термины

- продюсер — приложение или процесс, генерирующий и посылающий данные;
- потребитель — приложение или процесс, который принимает сгенерированное продюсером сообщение;
- сообщение — пакет данных, необходимый для совершения какой-либо операции;
- брокер — узел передачи сообщения от процесса-продюсера приложению-потребителю;
- топик — виртуальное хранилище сообщений (журнал записей) одинакового или похожего содержания.
- партиция - отдельная часть топика (упорядоченность сообщений просиходит на уровне партиций)


TOPIC
PARTITION 1
RECORD
RECORD
RECORD
PARTITION 2
RECORD
RECORD
PARTITION 3
RECORD
RECORD
RECORD
PRODUCERS
Writes to any partition
Determined by key or specified partition
CONSUMERS
Reads from particular partitions
Determined by Kafka


KAFKA CLUSTER
Broker-1
Broker-2
Broker-3
Topic 1 Partition 0
Topic 1 Partition 1
Topic 1 Partition 2
Topic 1 Partition 1
Topic 1 Partition 0
Topic 1 Partition 2
Topic 1 Partition 2
Topic 1 Partition 0
Topic 1 Partition 1


REST Service
Orders Service
Kafka Topics
Elasticsearch Sink Connector
orders
Order Details Service
warehouse-inventory
Validators Aggregator Service
Fraud Service
order-validations
Inventory Service
payments
customers
Email Service
JDBC Source Connector

Kafka: визуализатор работы

Визуализатор полной работы кафка кластера:
https://softwaremill.com/kafka-visualisation/

- Designing Event Driven Systems
  http://www.benstopford.com/2018/04/27/book-designing-event-driven-systems/
- Kafka: The Definitive Guide
  https://www.confluent.io/wp-content/uploads/confluent-kafka-definitive-guide-complete.pdf

RabbitMQ vs Kafka

- https://jack-vanlightly.com/blog/2017/12/4/rabbitmq-vs-kafka-part-1-messaging-topologies
- https://content.pivotal.io/blog/understanding-when-to-use-rabbitmq-or-apache-kafka

И еще почитать

- RabbitMQ против Kafka: два разных подхода к обмену сообщениями
  https://habr.com/ru/company/itsumma/blog/416629/
- Understanding When to use RabbitMQ or Apache Kafka
  https://content.pivotal.io/blog/understanding-when-to-use-rabbitmq-or-apache-kafka
- Apache Kafka: обзор
  http://habr.com/ru/company/piter/blog/352978/
- Kafka и микросервисы: обзор
  https://habr.com/ru/company/avito/blog/465315/
- Apache Kafka и миллионы сообщений в секунду
  https://habr.com/ru/company/tinkoff/blog/342892/
- Apache Kafka и RabbitMQ: семантика и гарантия доставки сообщений
  https://habr.com/ru/company/itsumma/blog/437446/

Домашнее задание

Реализовать "напоминания" о событиях с помощью очереди сообщений (на ваш выбор какой):

- создать процесс, который периодически сканирует основную базу данных, выбирая события о которых нужно напомнить;
- создать процесс, который читает сообщения из очереди и шлет уведомления.

https://github.com/OtusGolang/home_work/blob/master/hw12_13_14_15_calendar/docs/14_README.md
https://github.com/rabbitmq/rabbitmq-tutorials/tree/master/go

Заполните пожалуйста опрос
https://otus.ru/polls/?????/

Спасибо за внимание!