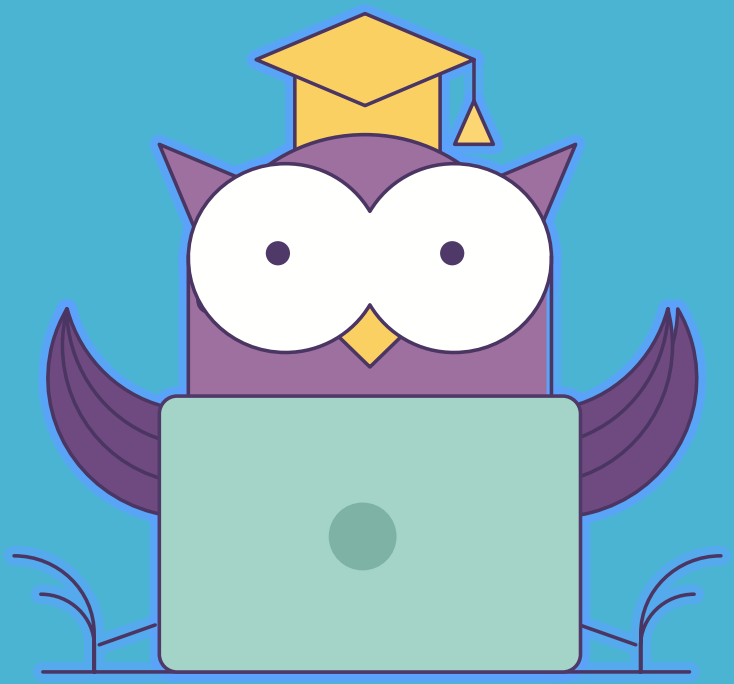
ОНЛАЙН-ОБРАЗОВАНИЕ

OTUS

Как меня слышно и видно?



> Напишите в чат

+ если все хорошо

- если есть проблемы со звуком или с видео

!проверить запись!

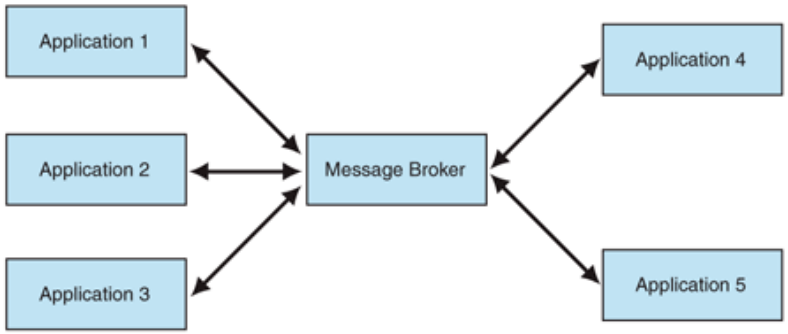
Очереди сообщений

Александр Давыдов

Антон Телышев

План занятия

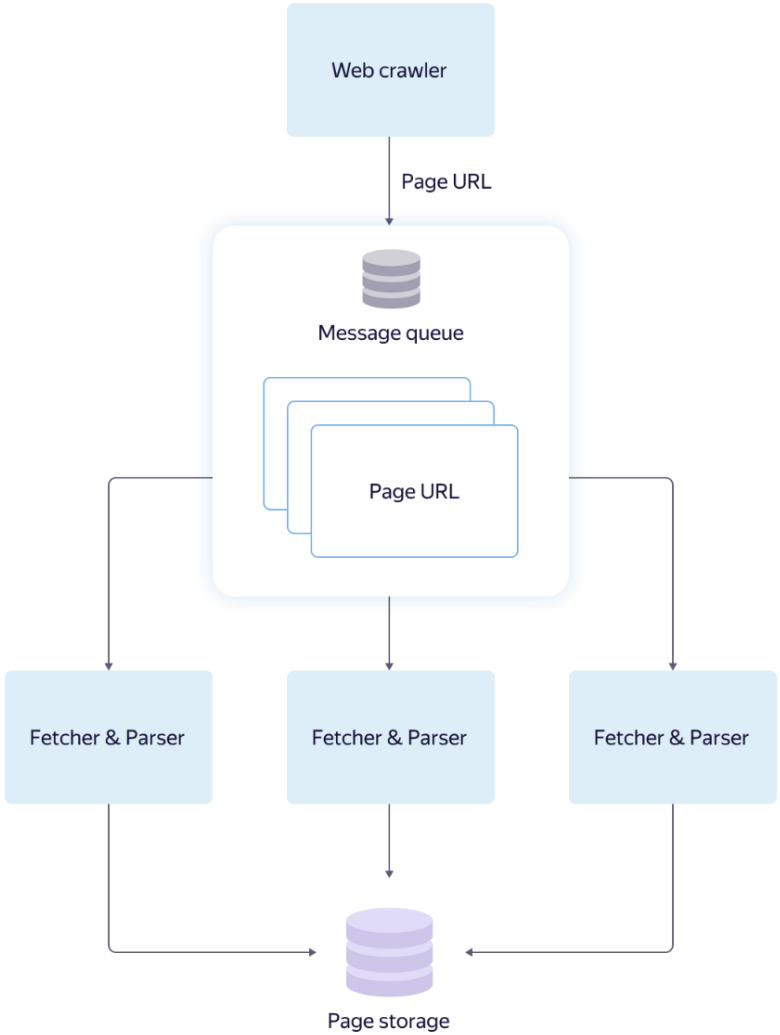
- Очереди сообщений
- Apache Kafka
- Событийно-ориентированная архитектура
- RabbitMQ
- Использование RabbitMQ



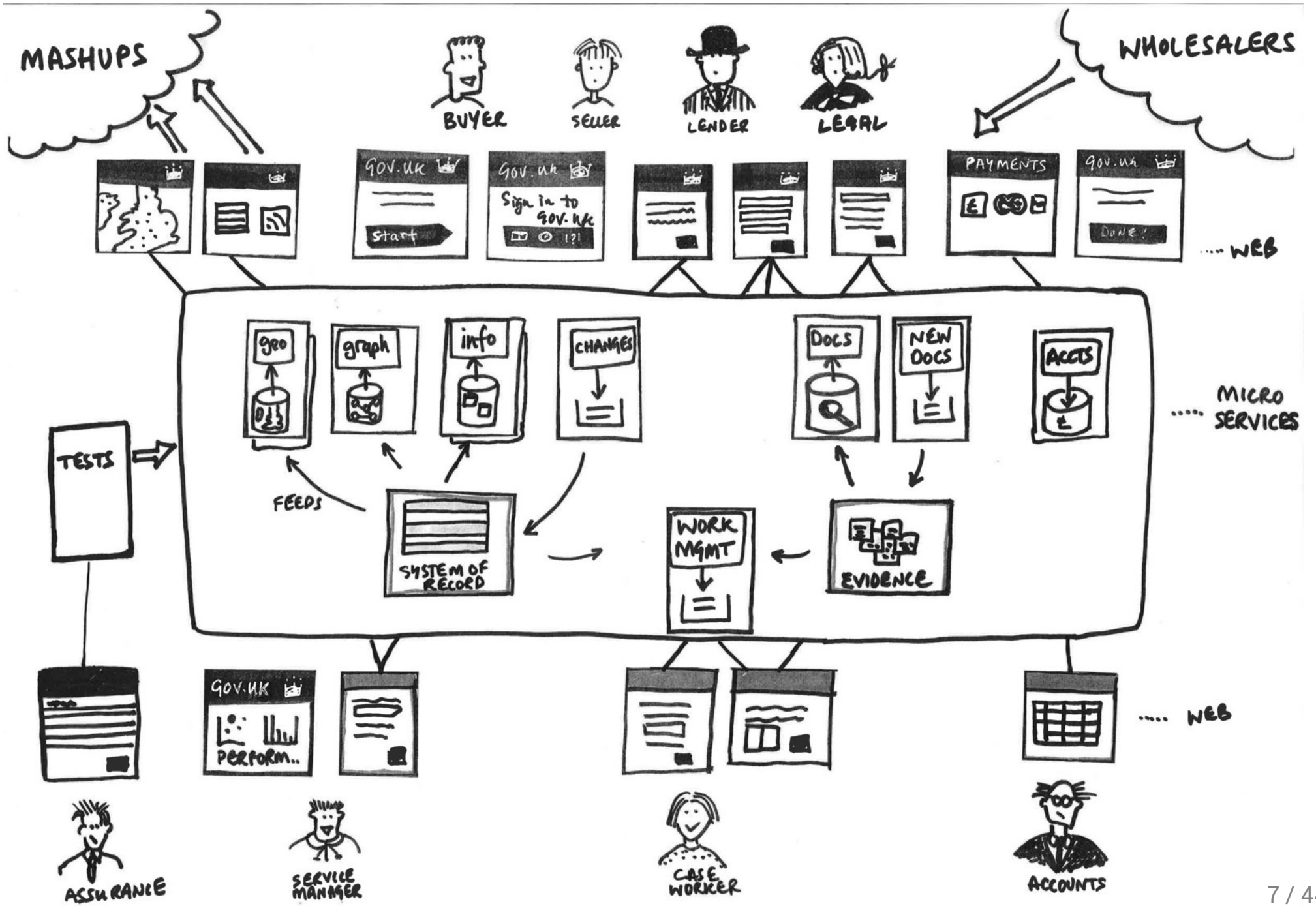
- Yandex Message Queue (https://cloud.yandex.ru/services/message-queue)
- Amazon Web Services (AWS) Simple Queue Service (SQS)
- Apache ActiveMQ
- Apache Kafka
- Redis (pubsub)
- RabbitMQ

etc.

Пример: crawler (поисковый робот)



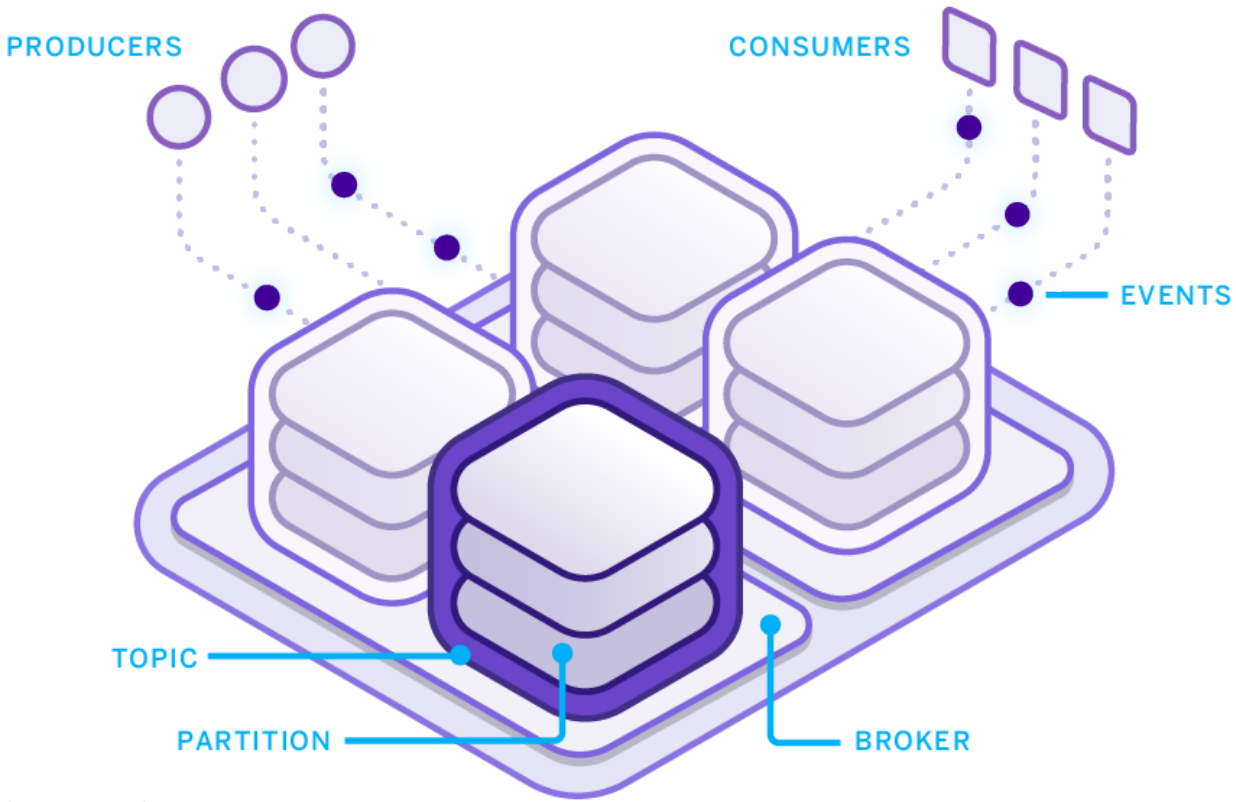
Вернемся к проблемам микросервисов...



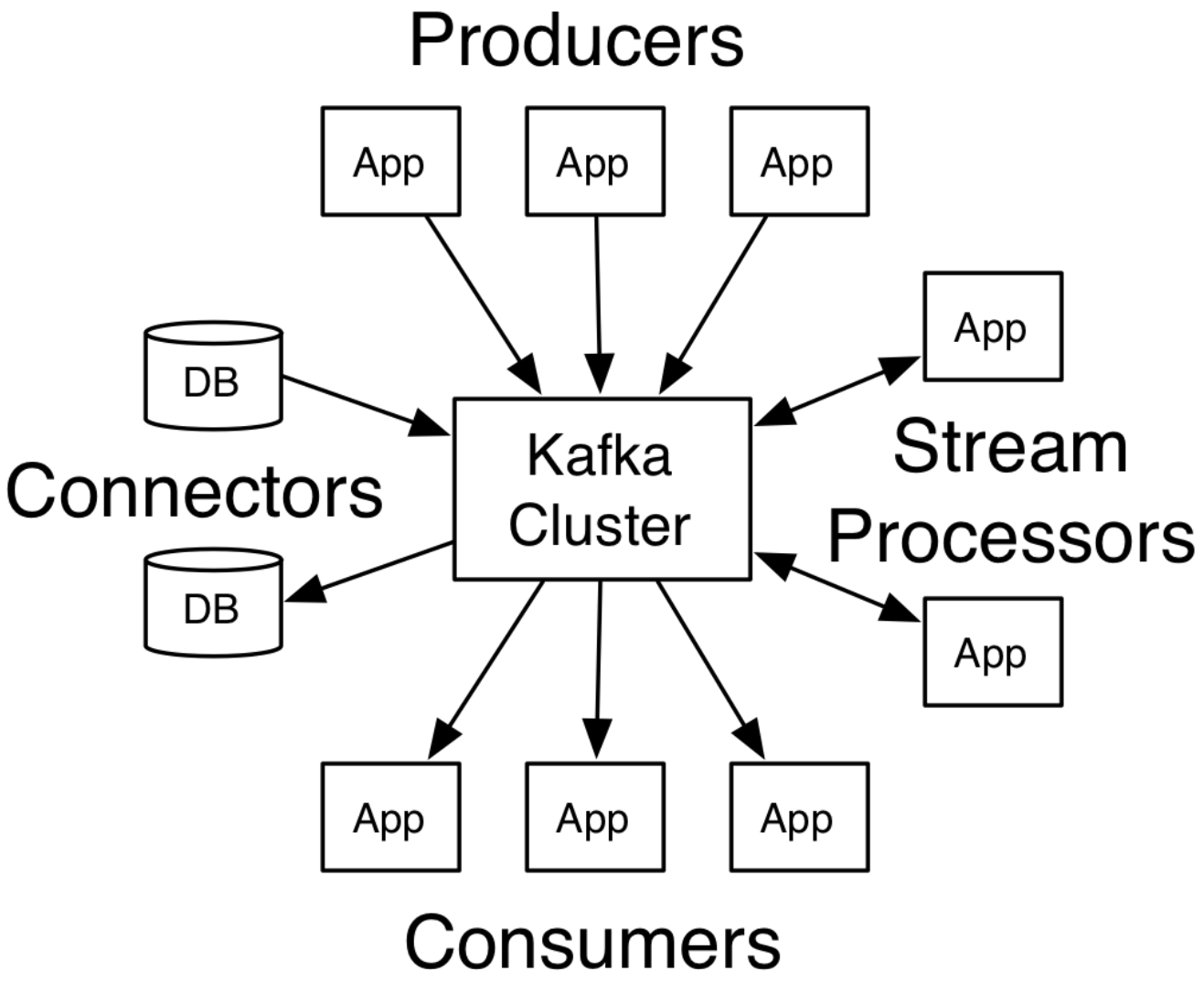
- Слабое связывание
- Масштабируемость
- Эластичность
- Отказоустойчивость
- Гарантированная доставка
- Гарантированный порядок доставки
- Буферизация
- Понимание потоков данных
- Асинхронная связь

- 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
- Надежный (репликация)
- Высокодоступный (high availability)

- 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



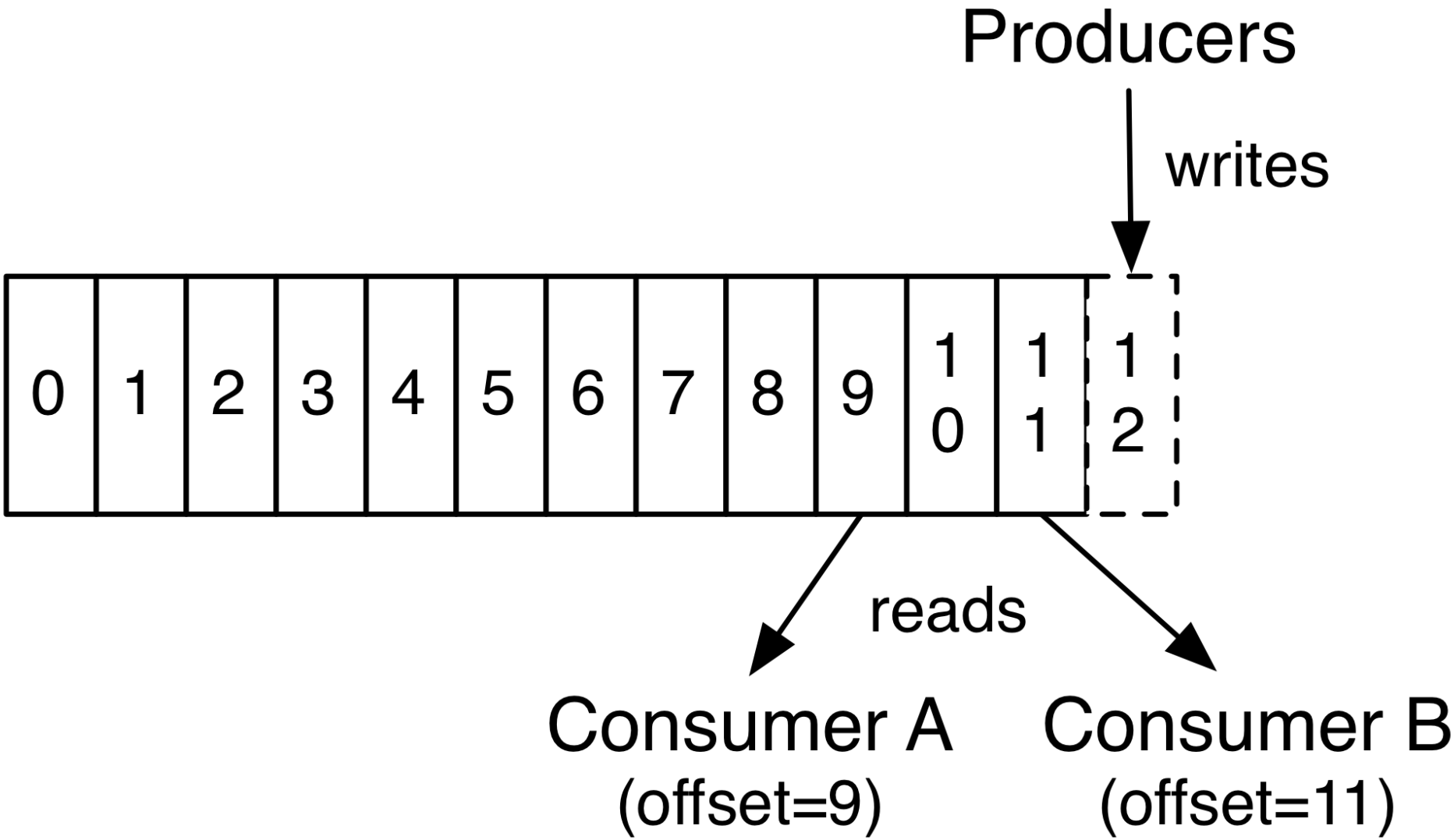
— Издатель (producer)
— Топик (topic), раздел (partition)
— Подписчики (consumer), группа (group)

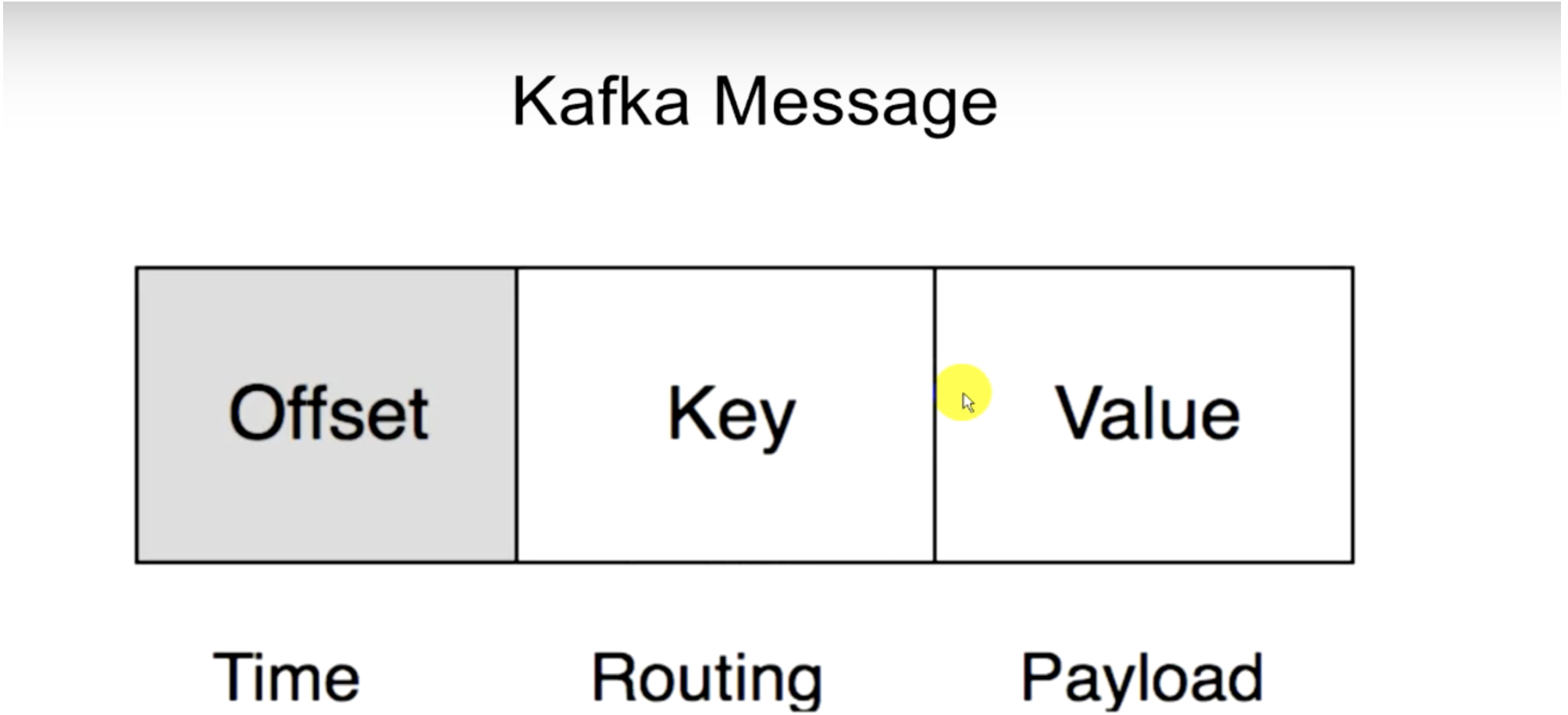

Producers
App
App
App
DB
DB
Connectors
Kafka
Cluster
App
Stream
Processors
App
App
App
App
Consumers

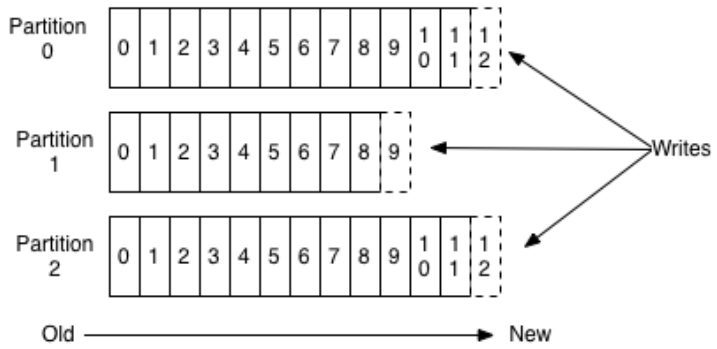
- The Producer API allows an application to publish a stream of records to one or more Kafka topics
- The Consumer API allows an application to subscribe to one or more topics and process the stream of records produced to them
- The Streams API allows an application to act as a stream processor, consuming an input stream from one or more topics and producing an output stream to one or more output topics, effectively transforming the input streams to output streams
- The Connector API allows building and running reusable producers or consumers that connect Kafka topics to existing applications or data systems. For example, a connector to a relational database might capture every change to a table

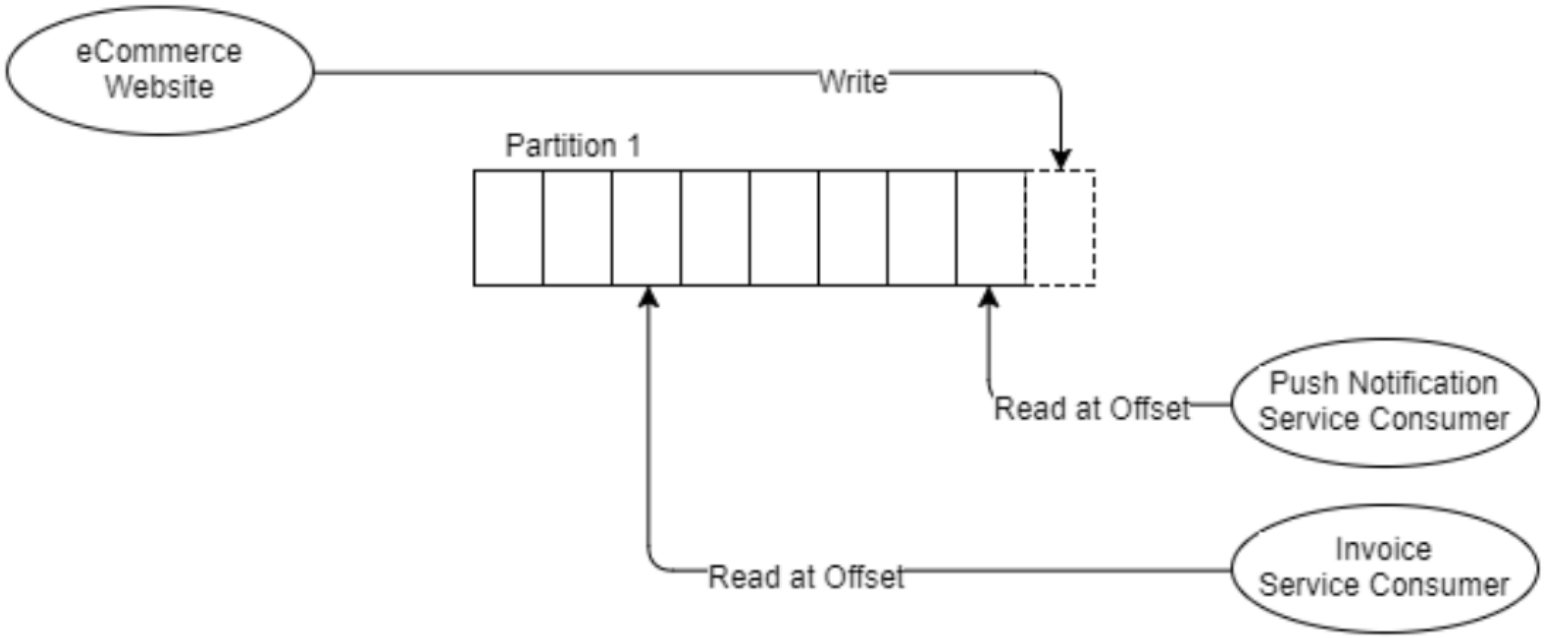
---

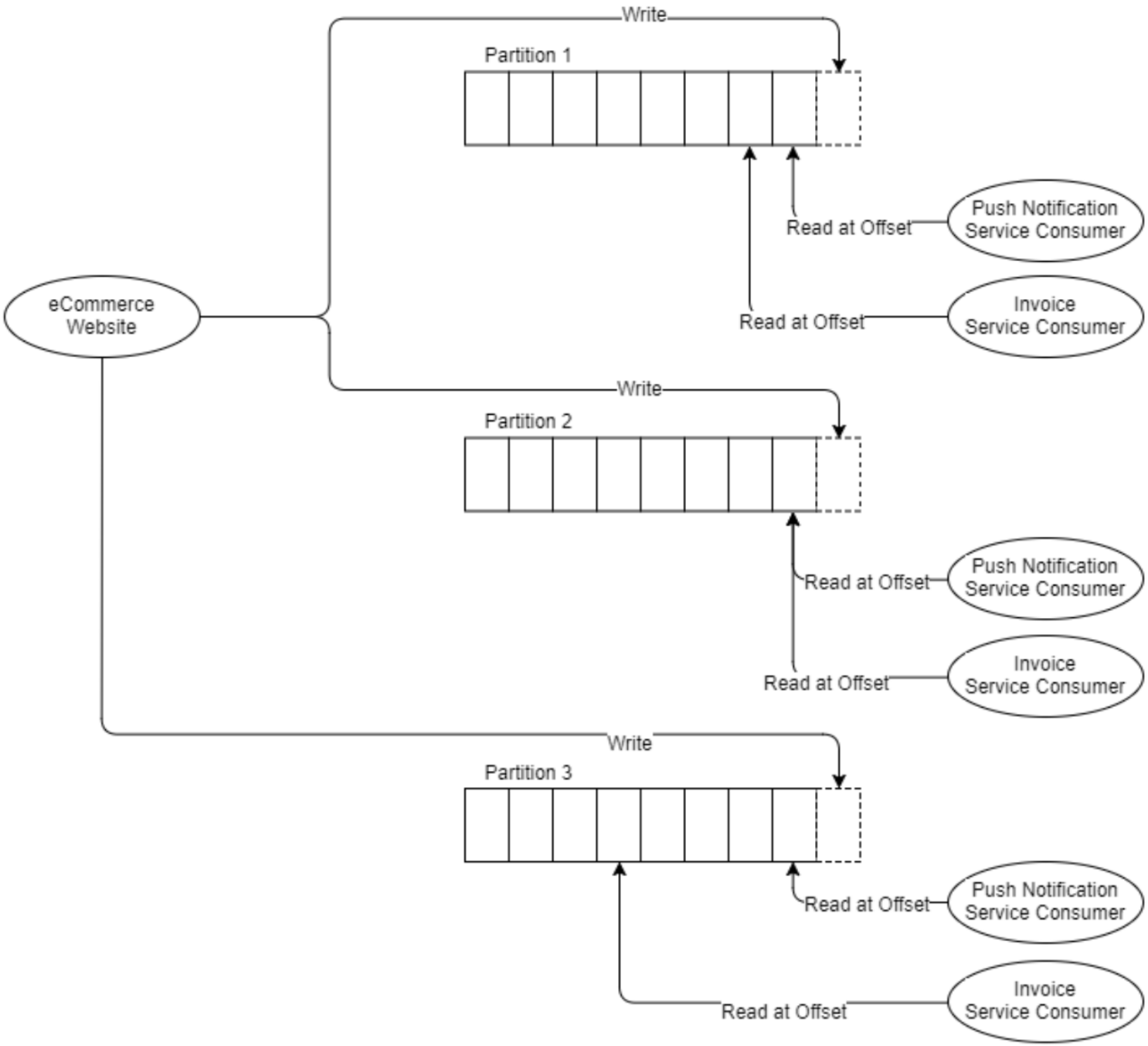
- The Producer API - публикация записей в топики.
- The Consumer API - подпись на топики.
- The Streams API - манипуляции с потоками, конвертация из одного топика в другой
- The Connector API - коннекторы к сторонним системам

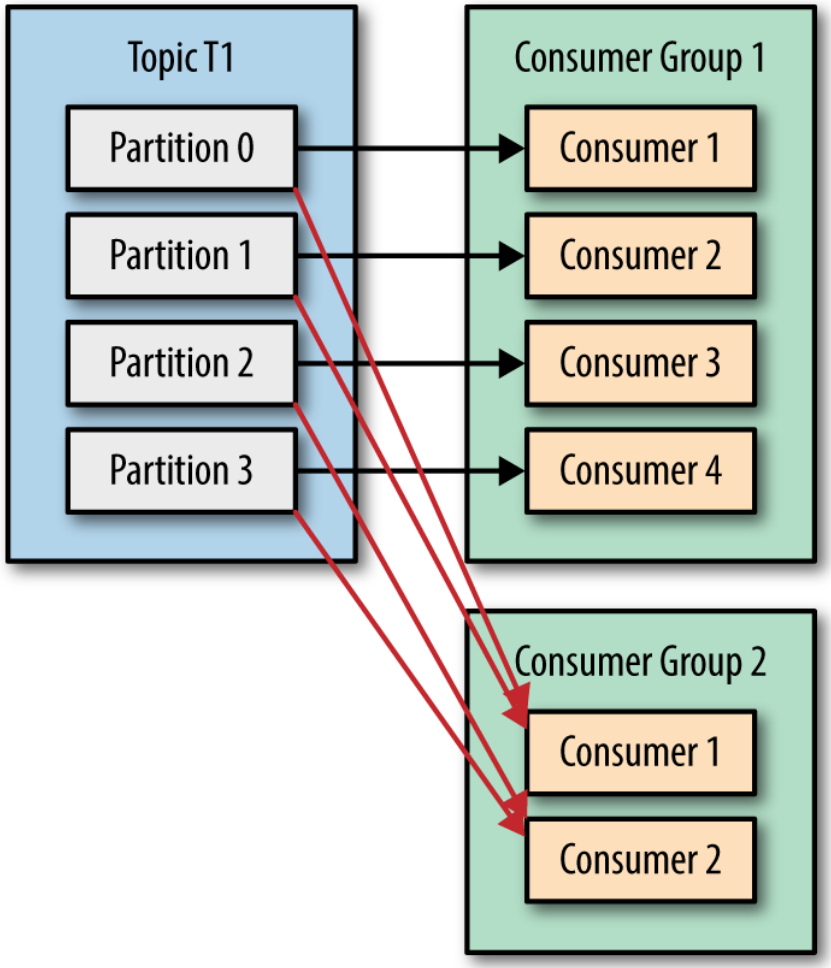

Producers
writes
0
1
2
3
4
5
6
7
8
9
10
11
12
reads
Consumer A
(offset=9)
Consumer B
(offset=11)

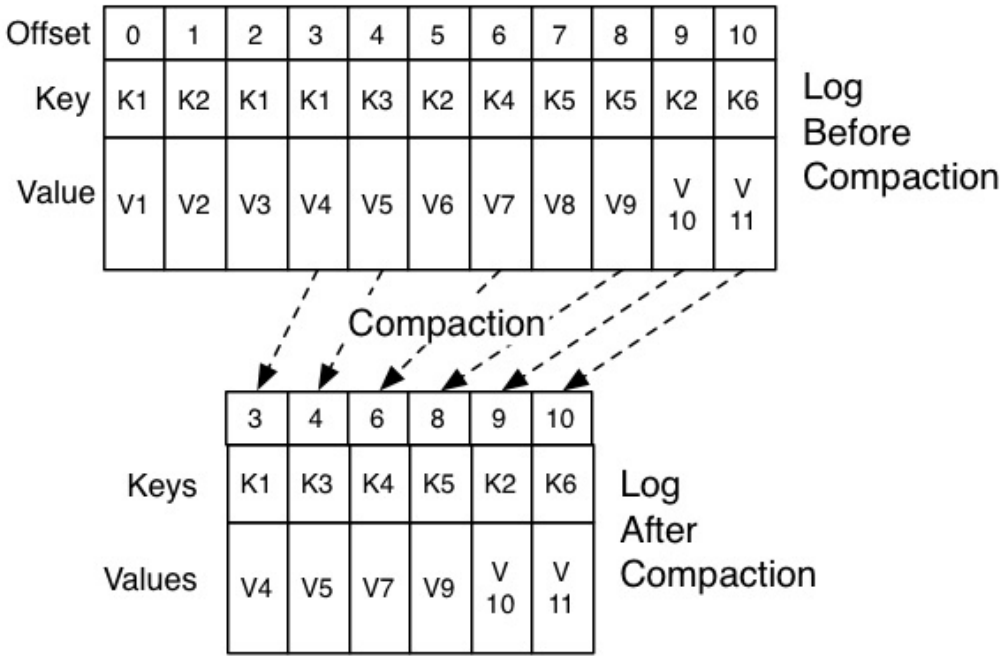

Kafka Message
Offset
Key
Value
Time
Routing
Payload


Partition 0
0 1 2 3 4 5 6 7 8 9 10 11 12
Partition 1
0 1 2 3 4 5 6 7 8 9
Partition 2
0 1 2 3 4 5 6 7 8 9 10 11 12
Writes
Old
New

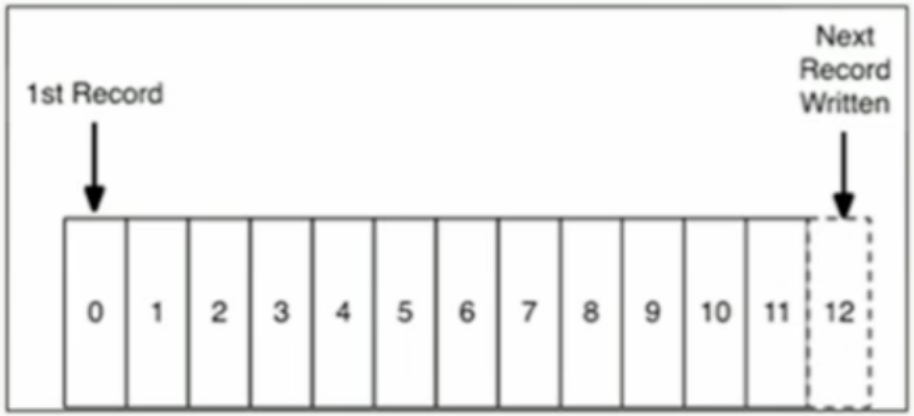

eCommerce Website
Write
Partition 1
Read at Offset
Push Notification Service Consumer
Read at Offset
Invoice Service Consumer


Write
Partition 1
Read at Offset
Push Notification Service Consumer
eCommerce Website
Read at Offset
Invoice Service Consumer
Write
Partition 2
Read at Offset
Push Notification Service Consumer
Read at Offset
Invoice Service Consumer
Write
Partition 3
Read at Offset
Push Notification Service Consumer
Read at Offset
Invoice Service Consumer


Topic T1
Partition 0
Partition 1
Partition 2
Partition 3
Consumer Group 1
Consumer 1
Consumer 2
Consumer 3
Consumer 4
Consumer Group 2
Consumer 1
Consumer 2


Offset
0
1
2
3
4
5
6
7
8
9
10
Key
K1
K2
K1
K1
K3
K2
K4
K5
K5
K2
K6
Value
V1
V2
V3
V4
V5
V6
V7
V8
V9
V 10
V 11
Log Before Compaction
Compaction
3
4
6
8
9
10
Keys
K1
K3
K4
K5
K2
K6
Values
V4
V5
V7
V9
V 10
V 11
Log After Compaction

RECORDS



- Name, Value, Timestamp
- Immutable
- Append Only
- Persisted

AKA: A Log

- Упорядоченность сообщений в партиции
- Консьюмер видит сообщения в порядке попадания их в лог
- Сохранение сообщений при N-1 падениях и replication factor N

- At-most-once delivery (“как максимум однократная доставка"). Сообщение не может быть доставлено больше одного раза. При этом сообщение может быть потеряно.
- At-least-once delivery (“как минимум однократная доставка"). Сообщение никогда не будет потеряно. При этом сообщение может быть доставлено более одного раза.
- Exactly-once delivery (“строго однократная доставка"). Святой грааль систем сообщений. Все сообщения доставляются строго единожды.

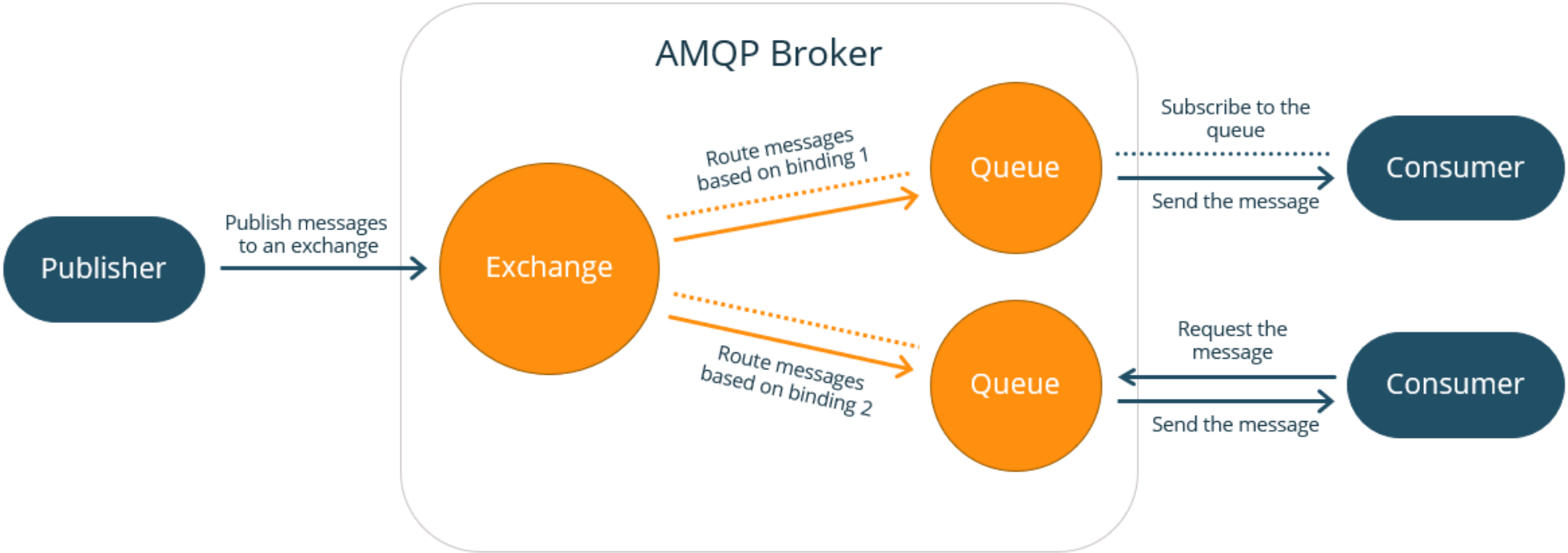
https://dzone.com/articles/interpreting-kafkas-exactly-once-semantics

- Message broker (ActiveMQ / RabbitMQ)
- Трекинг активности в вебе (linkedin)
- Сбор метрик
- Агрегация логов
- Stream processing (Kafka Streams)
- Event sourcing (https://martinfowler.com/eaaDev/EventSourcing.html)
- Storage? https://www.confluent.io/blog/okay-store-data-apache-kafka/

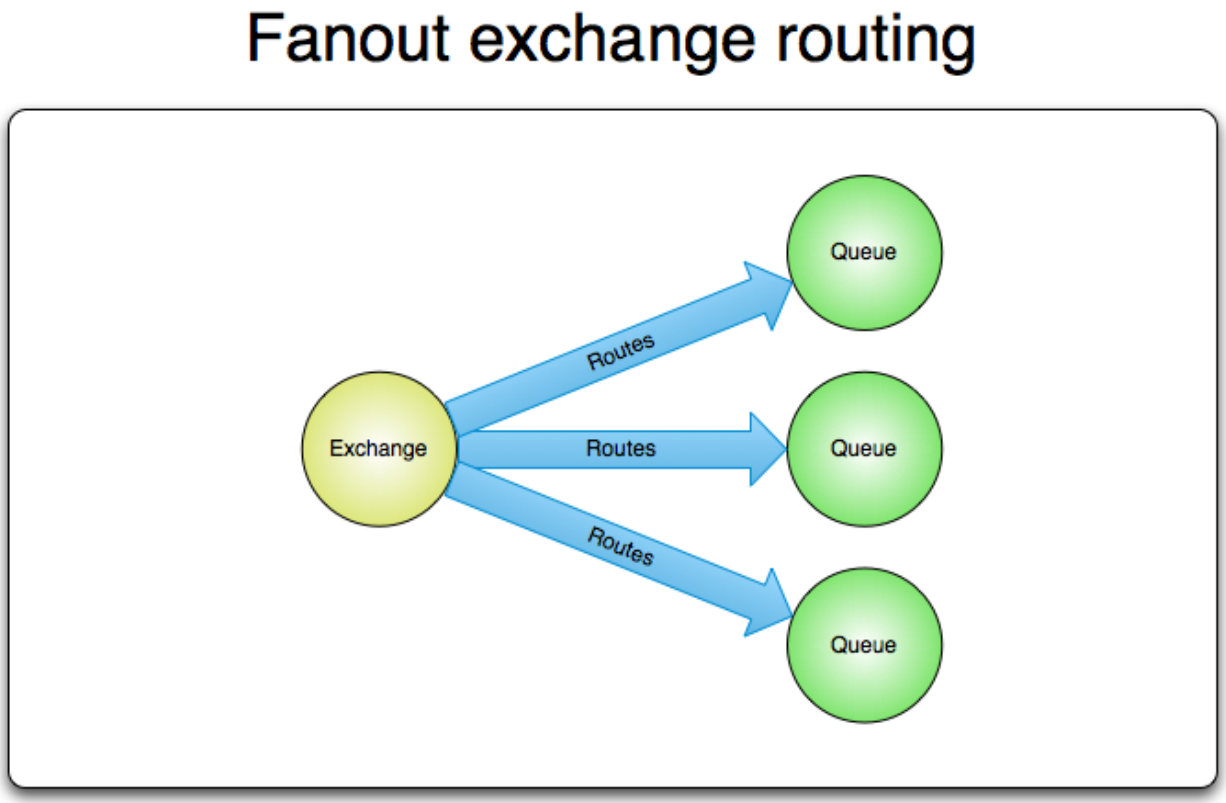
- https://github.com/confluentinc/confluent-kafka-go - отличная дока, librdkafka с lib
- https://github.com/Shopify/sarama - плохая дока, зато чистый go
- https://github.com/segmentio/kafka-go - отличный код, хорошая дока

RabbitMQ — это распределенная система управления очередью сообщений

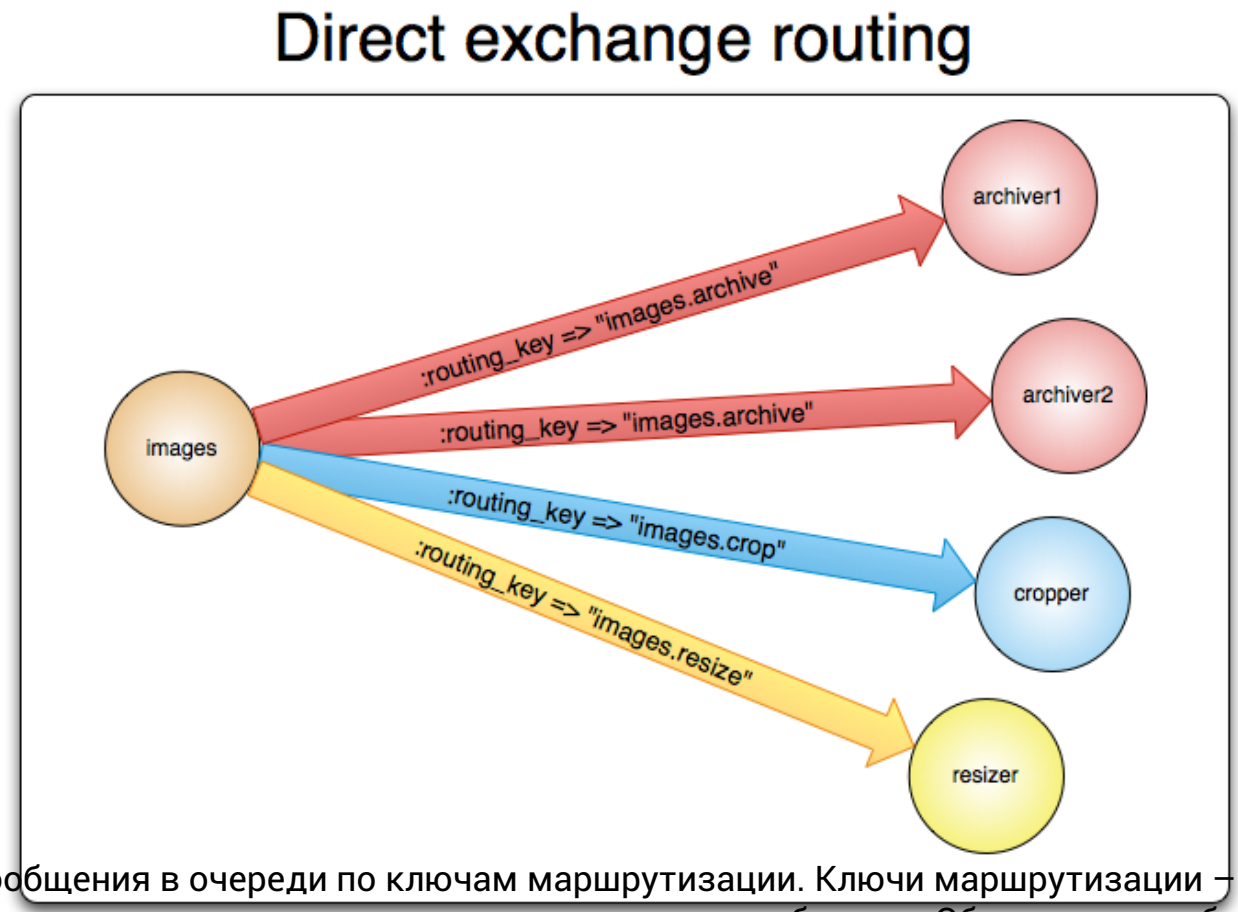
Advanced Message Queuing Protocol (AMQP)



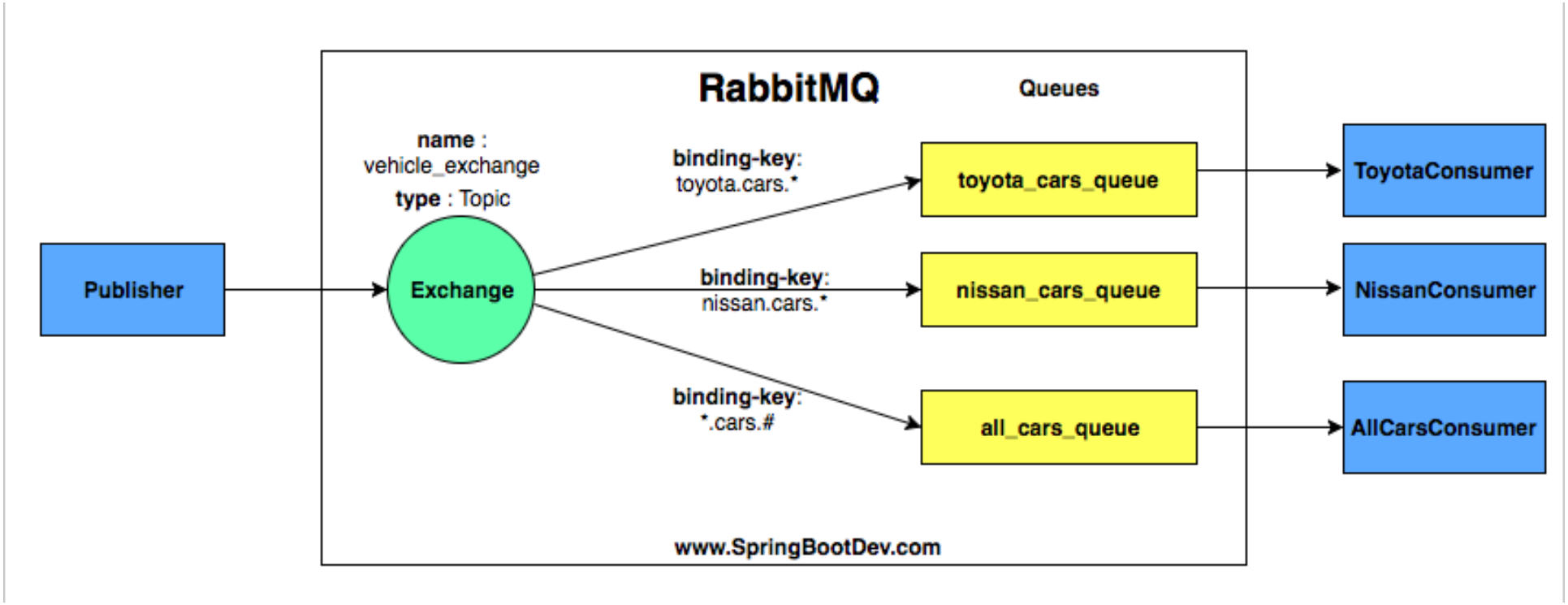
- Сообщение (**message**) — единица передаваемых данных, основная его часть (содержание) никак не интерпретируется сервером, к сообщению могут быть присоединены структурированные заголовки.
- Точка обмена (**Exchange**) — в неё отправляются сообщения. Точка обмена распределяет сообщения в одну или несколько 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.).

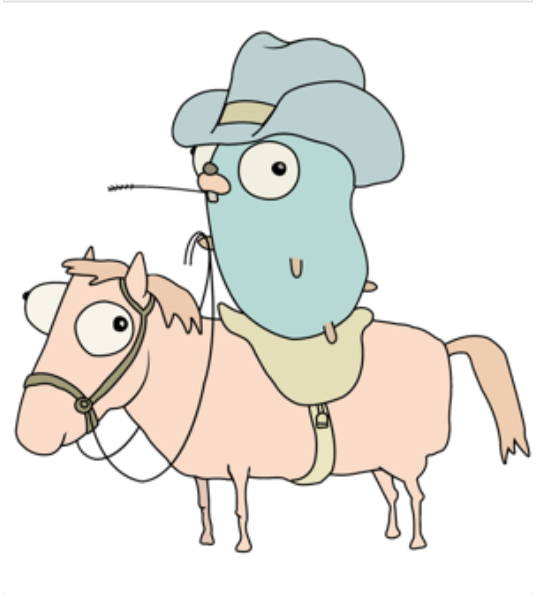


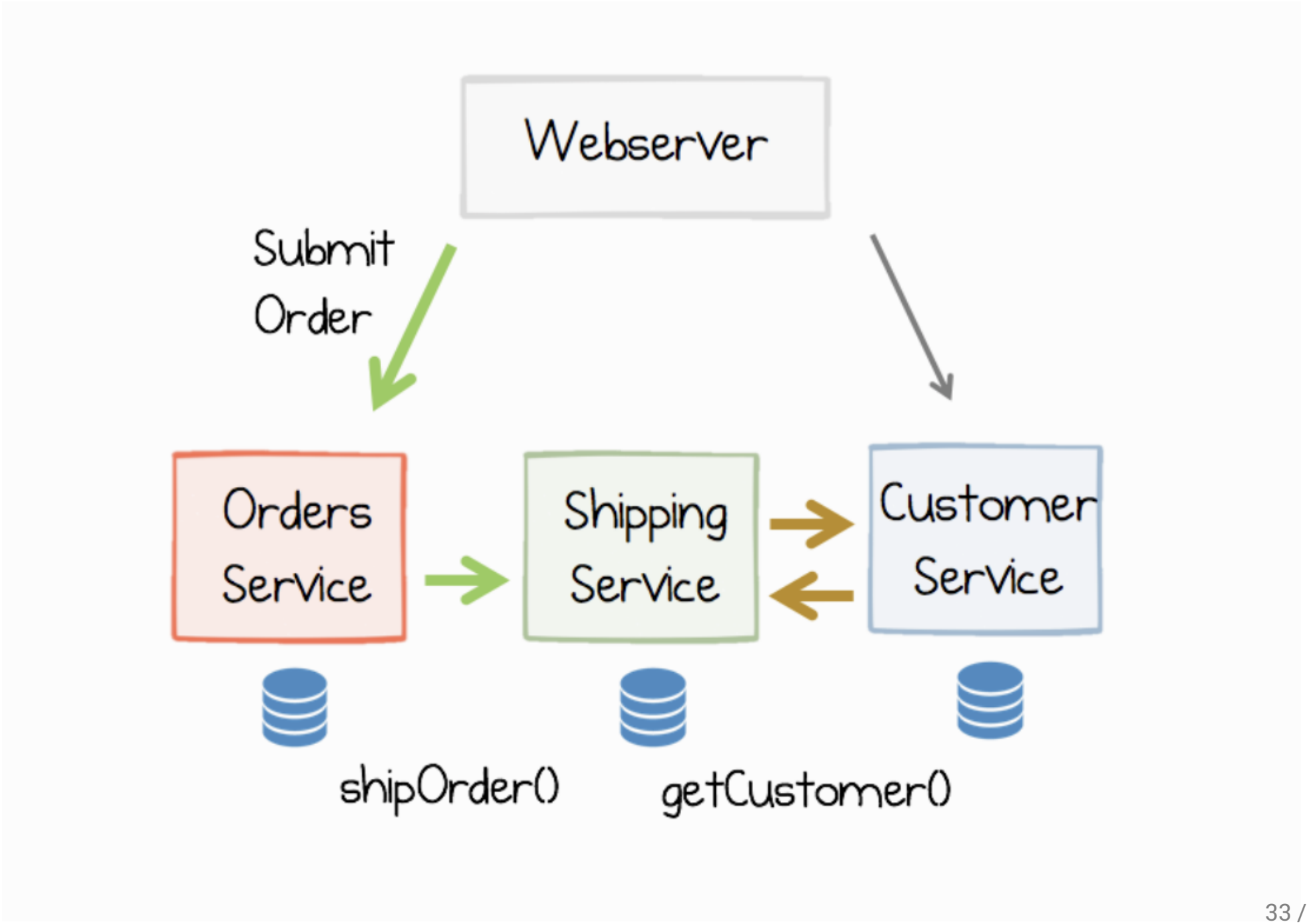
Direct – доставляет сообщения в очереди по ключам маршрутизации. Ключи маршрутизации – это дополнительные данные, которые определяют, в какую очередь нужно отправить сообщение. Обычно точки обмена такого типа используются в балансировке нагрузки round-robin.

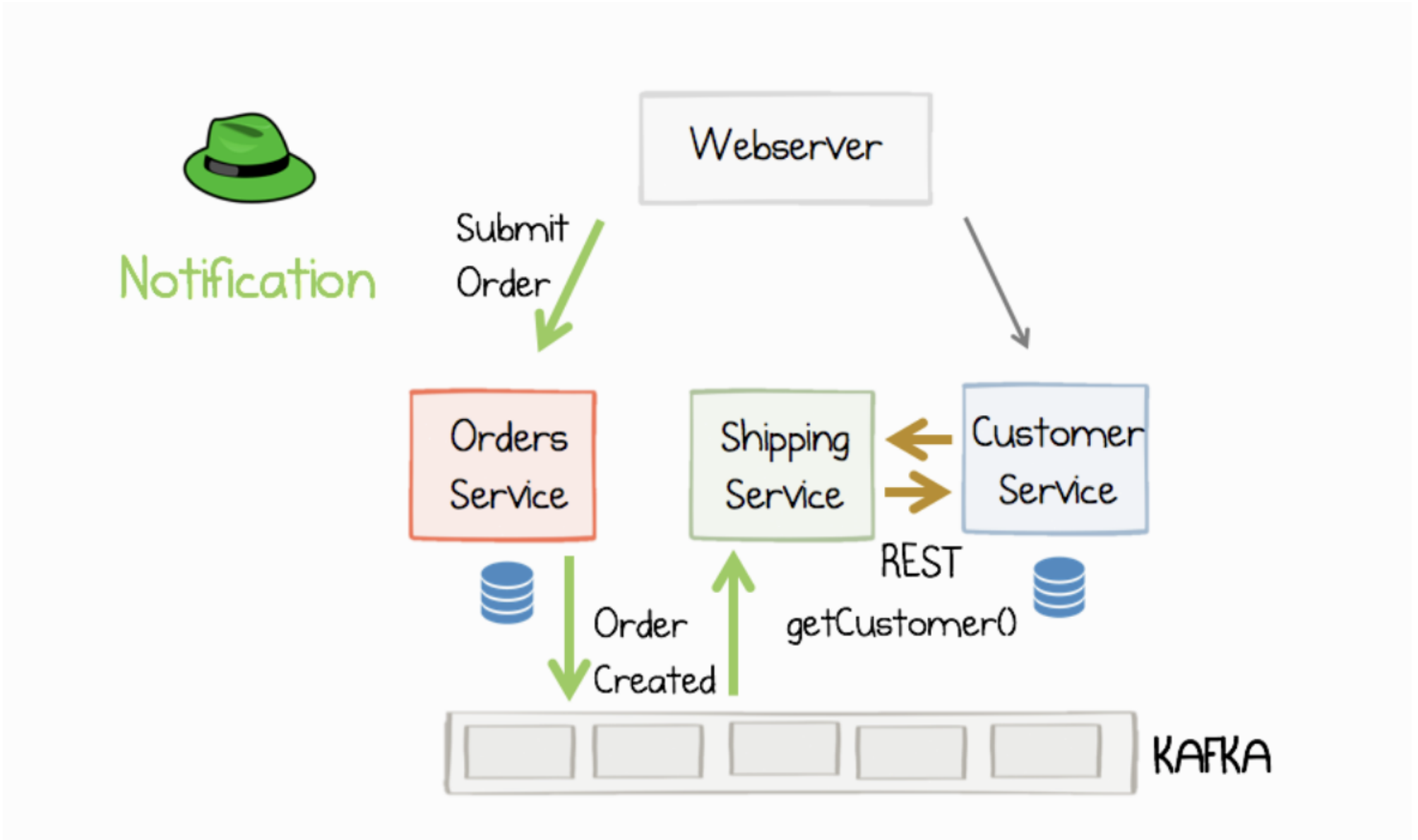


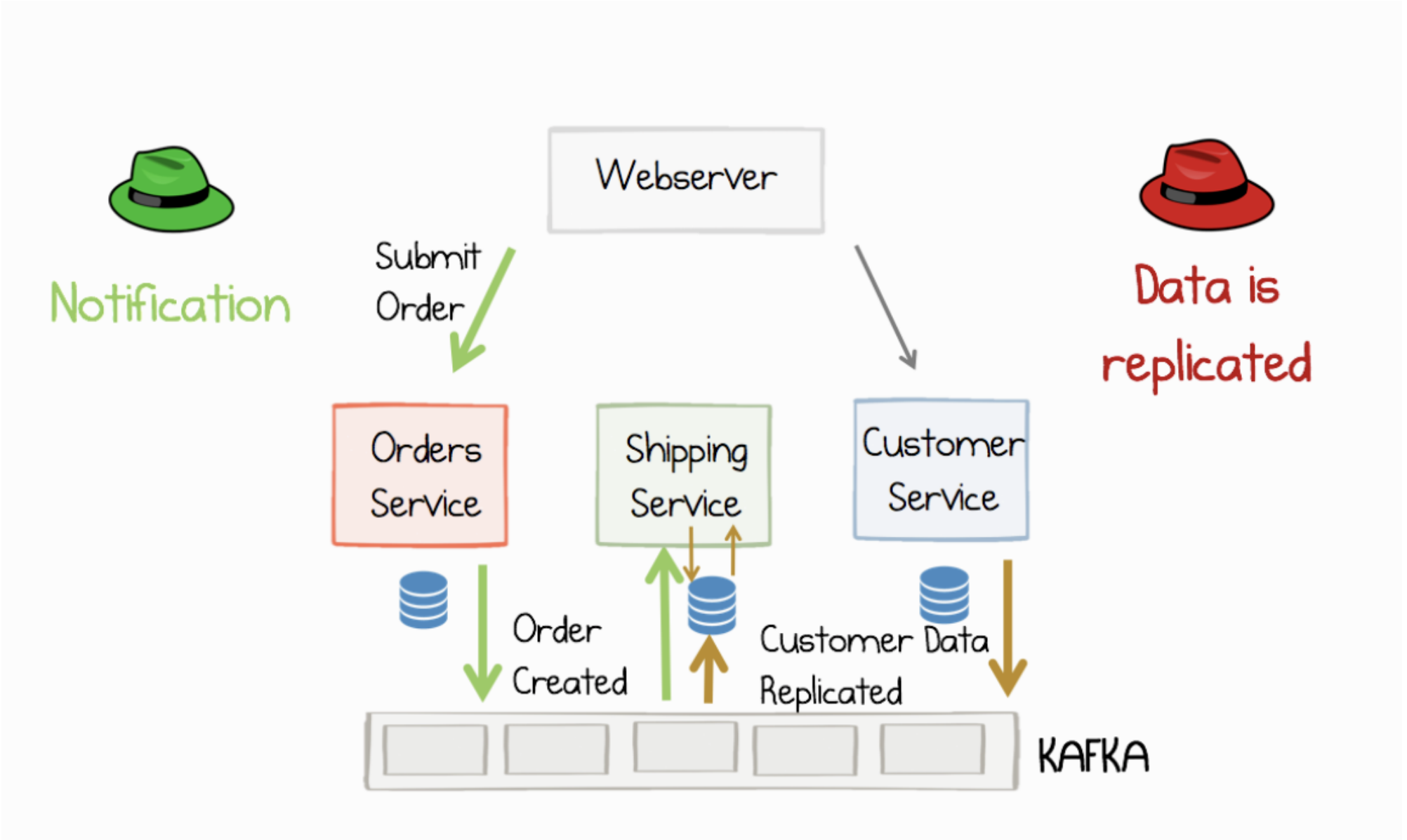
Topic – используется в шаблонах pub/sub. В этом случае ключ маршрутизации используется вместе с привязкой очередей к точке обмена. например, app.notification.sms.# — в очередь будут доставлены все сообщения, отправленные с ключами, начинающимися с app.notification.sms.

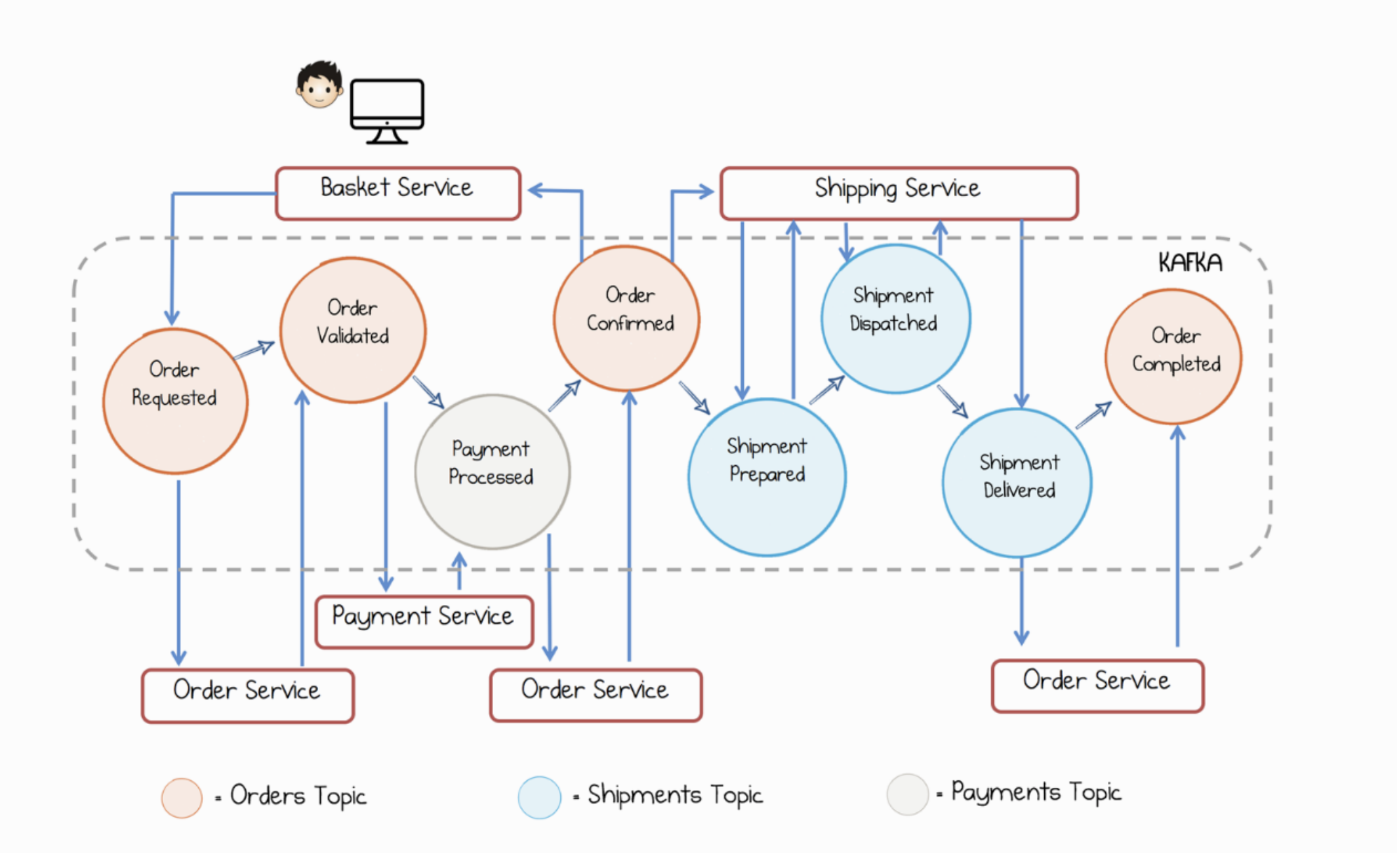
- https://jack-vanlightly.com/blog/2017/12/4/rabbitmq-vs-kafka-part-1-messaging-topologies
- https://content.pivotal.io/blog/understanding-when-to-use-rabbitmq-or-apache-kafka

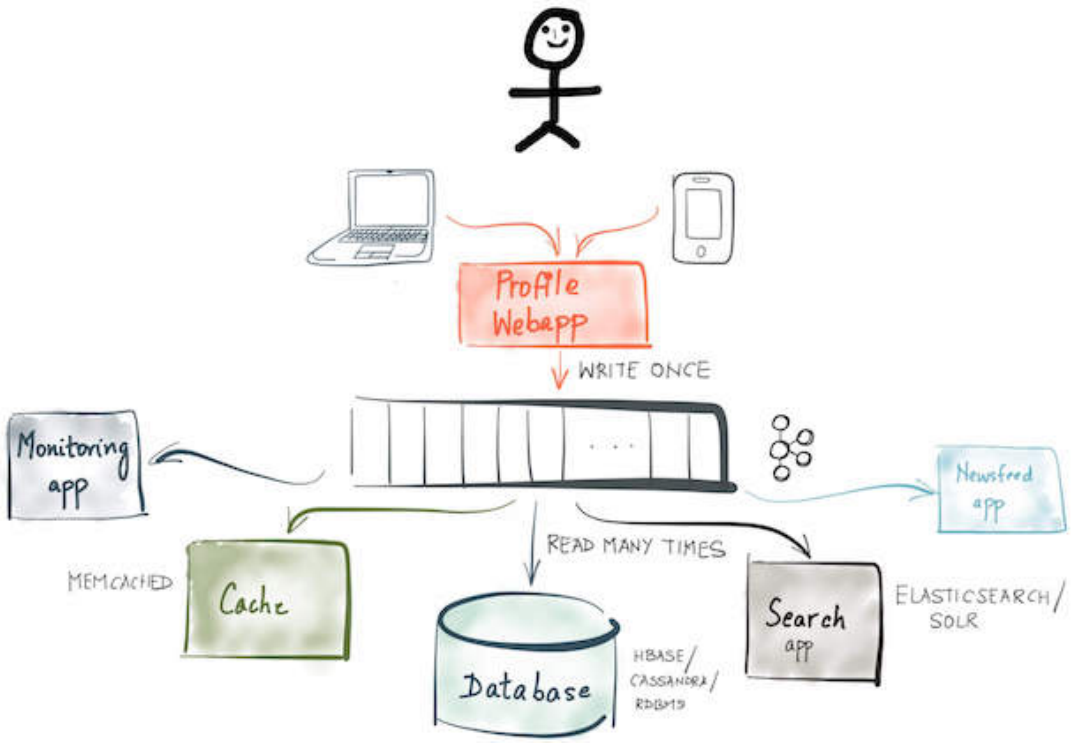


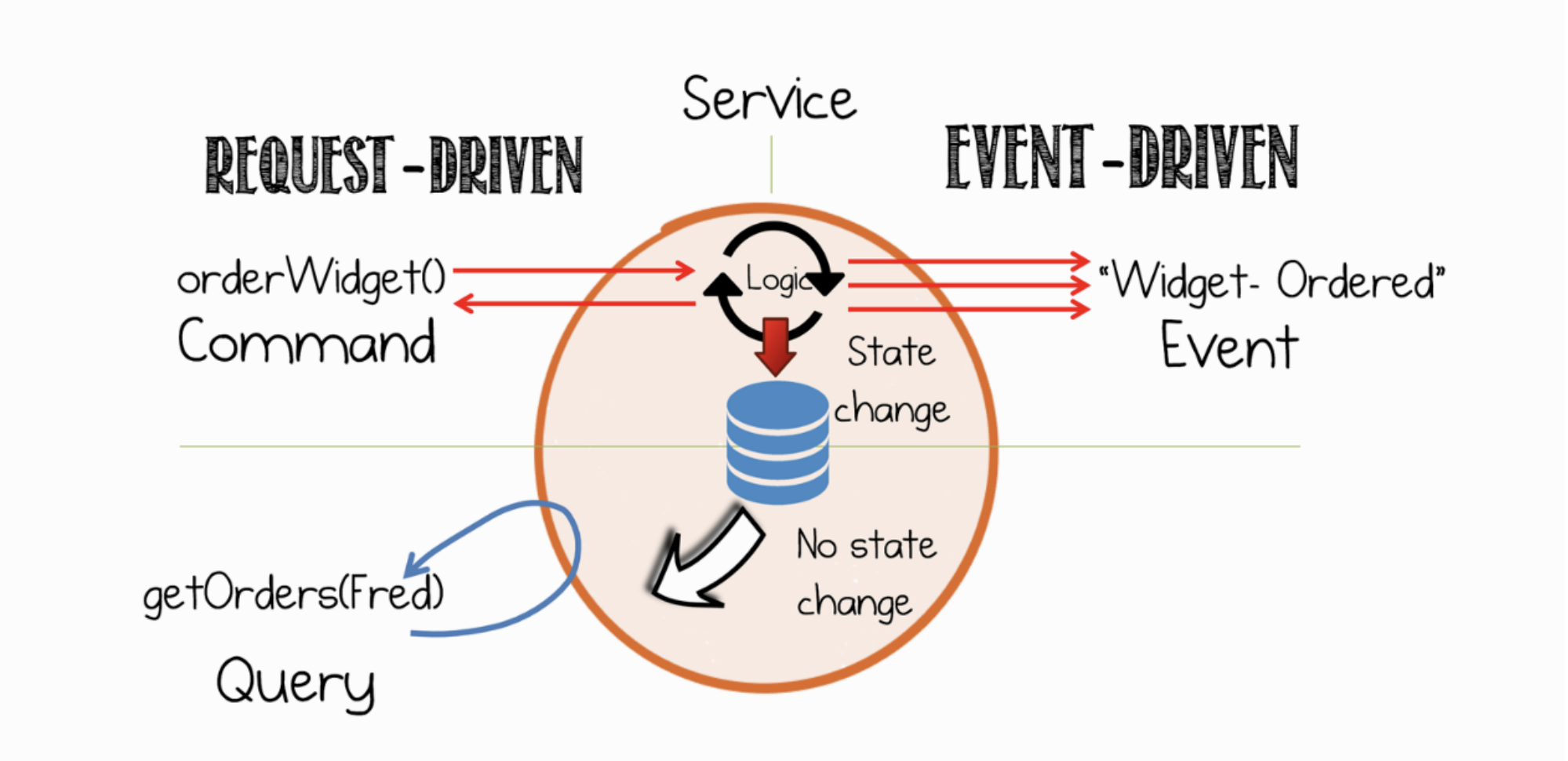

Webserver
Submit
Order
Orders
Service
Shipping
Service
Customer
Service
shipOrder()
getCustomer()


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

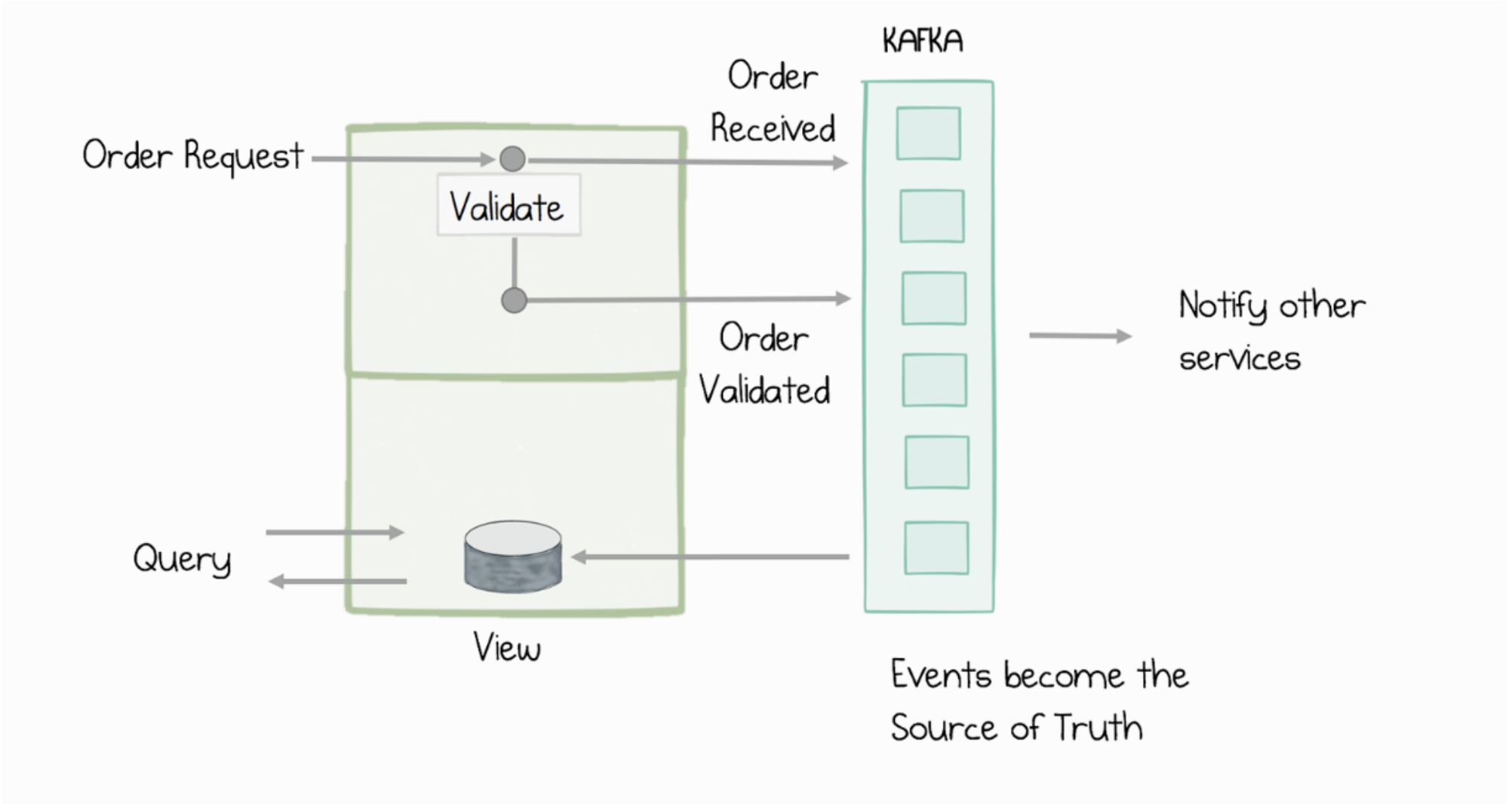

Webserver
Notification
Submit
Order
Data is
replicated
Orders
Service
Shipping
Service
Customer
Service
Order
Created
Customer Data
Replicated
KAFKA


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
"Widget- Ordered"
Event
State
change
No state
change
getOrders(Fred)
Query

Command Query Responsibility Segregation



- 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 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 RabbitMQ:

- 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/rabbitmq/rabbitmq-tutorials/tree/master/go

```
$ docker run -d --name rb -p 15672:15672 -p 5672:5672 rabbitmq:3-management
$ docker ps
```

Опрос

Заполните пожалуйста опрос

https://otus.ru/polls/4915/

Спасибо за внимание!