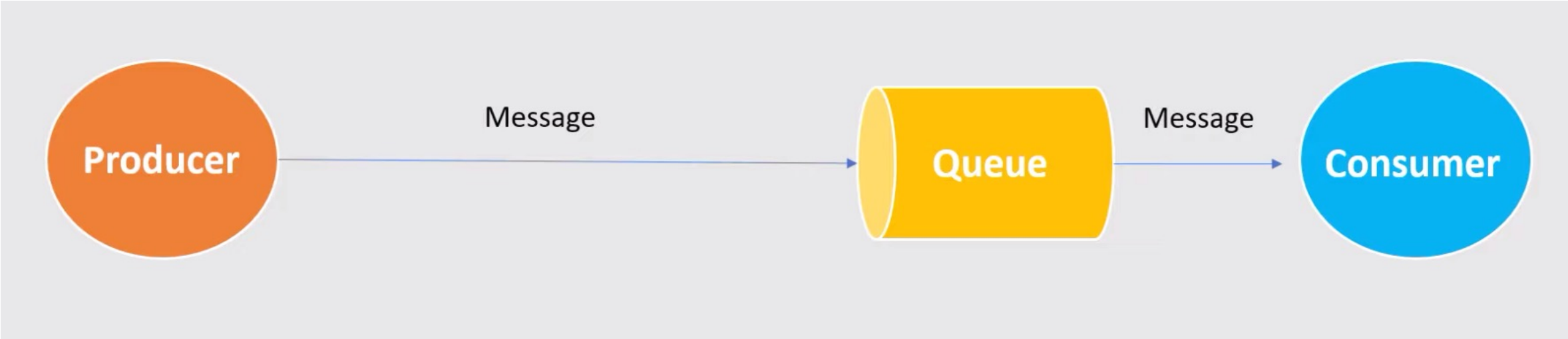
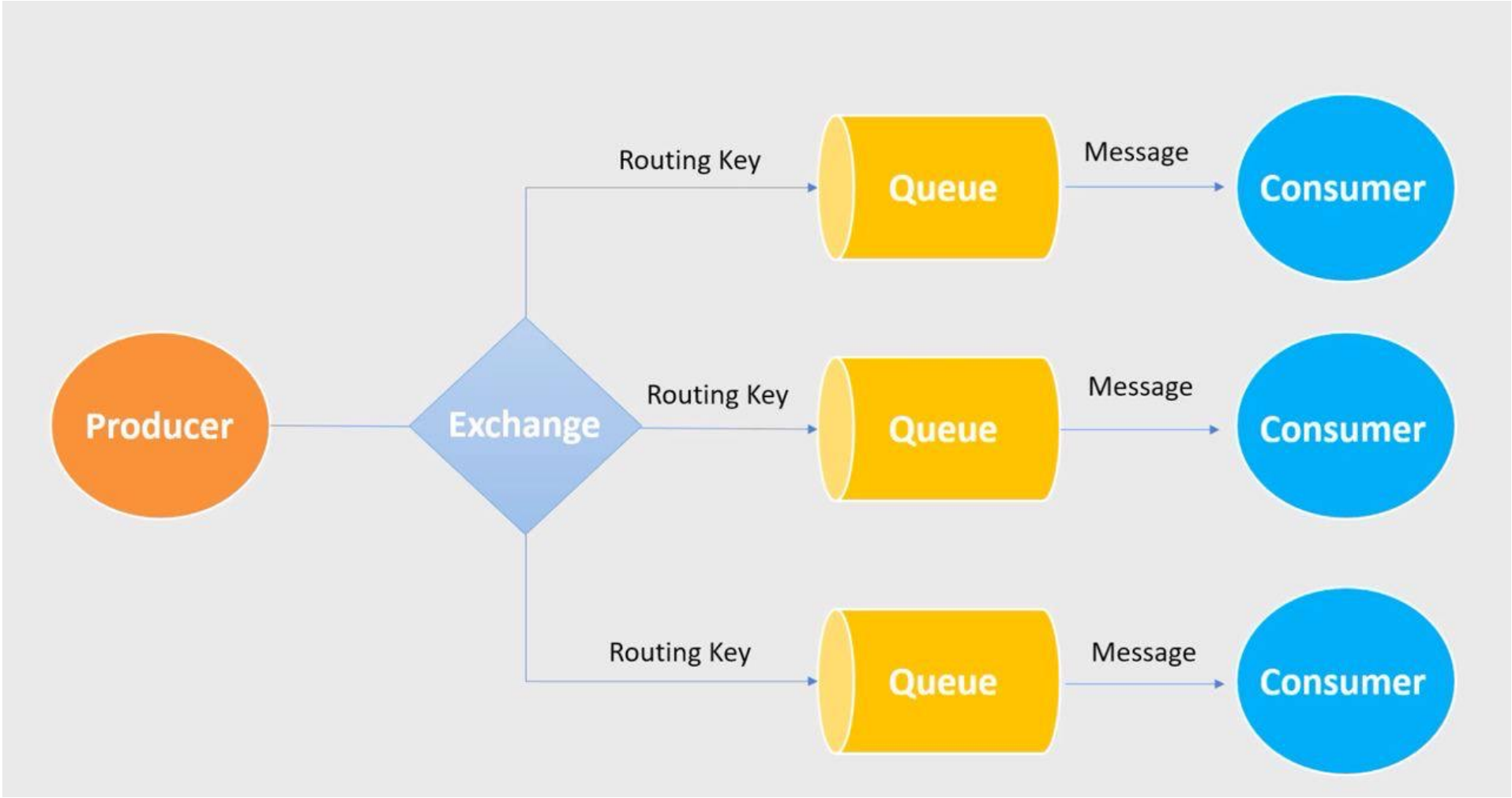
RabbitMQ

Кратко о приятном

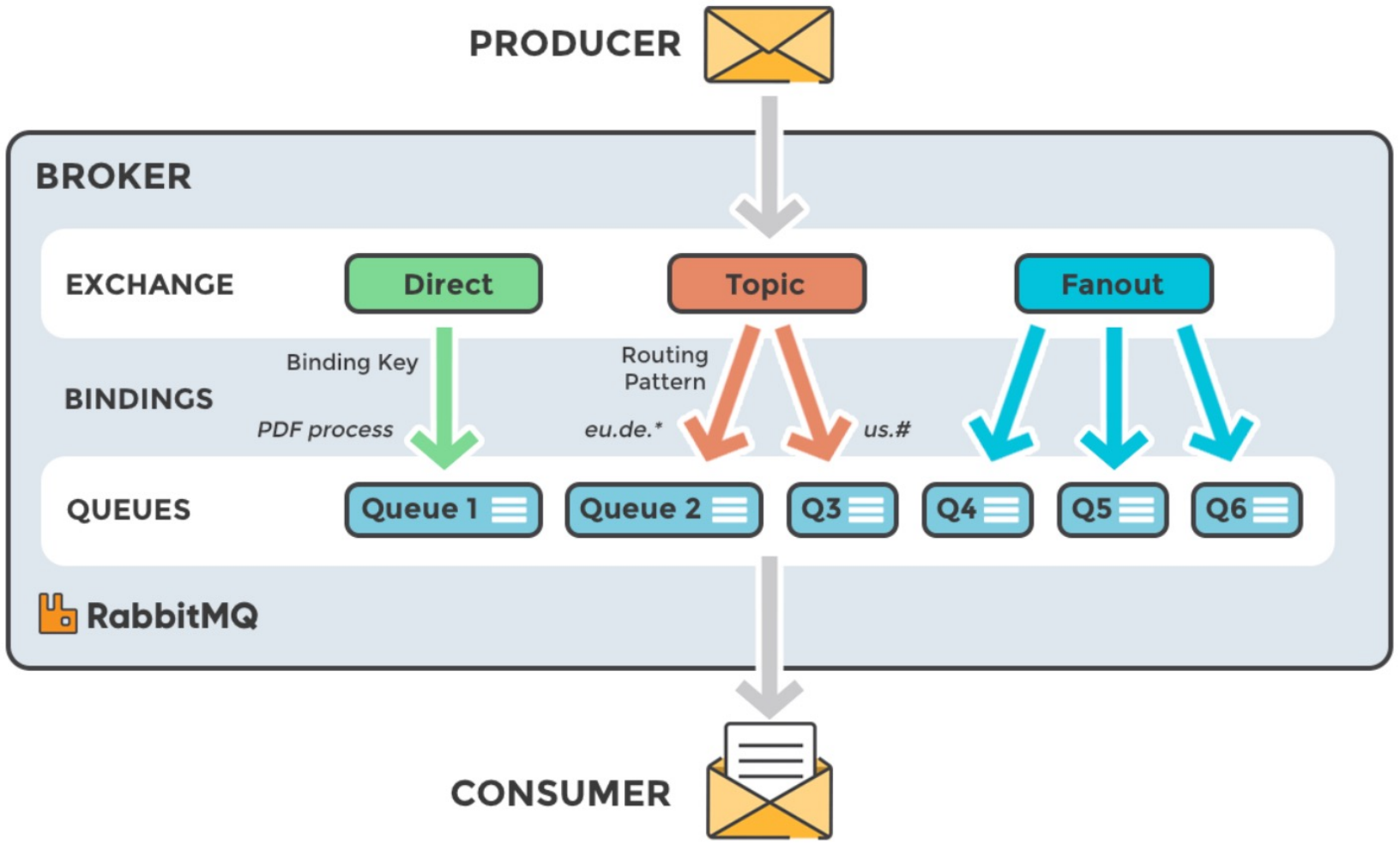
Традиционная система очередей



Структура брокера ч.1

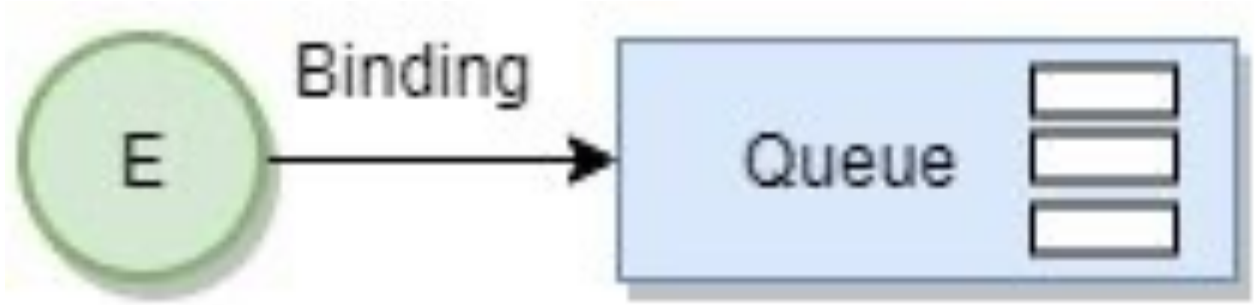


Структура брокера ч.2



AMQP

AMQP (Advanced Message Queuing Protocol) — открытый протокол для передачи сообщений между компонентами системы.

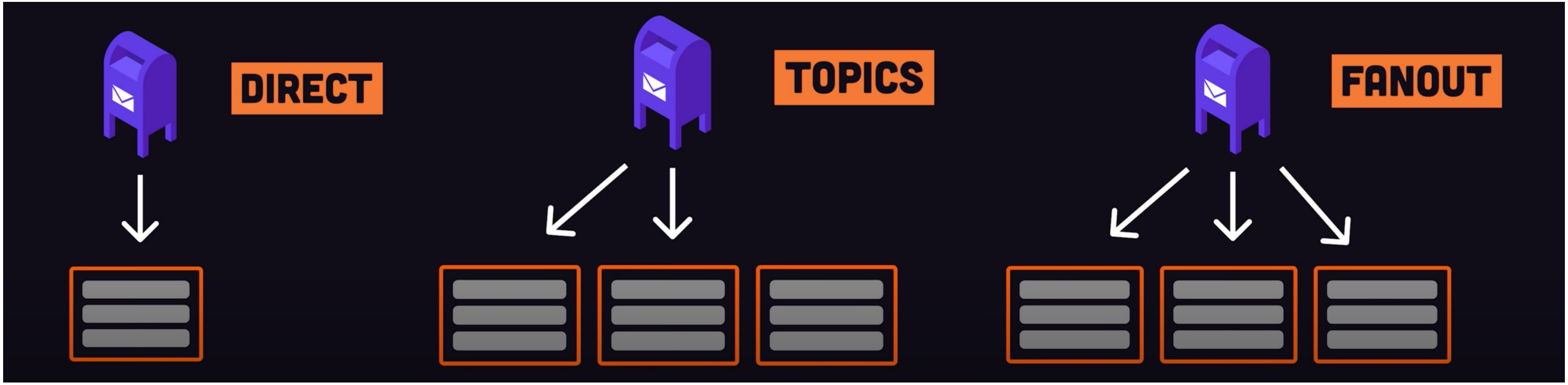


Протокол работает поверх TCP/IP.

3 понятия AMQP

- exchange (обменник или точка обмена) — в неё отправляются сообщения. Обменник **распределяет сообщение** в одну или несколько очередей. Он **маршрутизирует сообщения в очередь** на основе созданных связей (binding) между ним и очередью
- queue (очередь) — структура данных на диске или в оперативной памяти, которая **хранит ссылки на сообщения и отдает копии сообщений consumers** (потребителям). Одна очередь может использоваться несколькими потребителями
- binding (привязка) — правило, которое **сообщает точке обмена в какую из очередей эти сообщения должны попадать**. Обменник и очередь могут быть связаны несколькими привязками

Обмен сообщениями



Обмен может перенаправляться в конкретную очередь или в несколько с общим шаблоном, используя темы или в каждую очередь, используя разветвления

Что такое RabbitMQ?

RabbitMQ – это брокер сообщений с открытым исходным кодом. Он маршрутизирует сообщения по всем базовым принципам протокола AMQP. Отправитель передает сообщение брокеру а тот доставляет его получателю. Архитектура RabbitMQ-server основана на Erlang и BEAM.

Основная идея модели обмена сообщениями в RabbitMQ заключается в том, что producer (издатель) не отправляет сообщения непосредственно в очередь.

Вместо этого издатель может отправлять сообщения только на обмен. С одной стороны, обмен получает сообщения от издателей, а с другой — отправляет их в очереди.

Как работает RabbitMQ?

1. Издатель отправляет сообщение определенному обменнику
2. Обменник, получив сообщение, маршрутизирует его в одну или несколько очередей в соответствии с правилами привязки между ним и очередью
3. Очередь хранит ссылку на это сообщение. Само сообщение хранится в оперативной памяти или на диске
4. Как только потребитель готов получить сообщение из очереди, сервер создает копию сообщения по ссылке и отправляет
5. Потребитель получает сообщение и отправляет брокеру подтверждение
6. Брокер, получив подтверждение, удаляет копию сообщения из очереди. Затем удаляет из оперативной памяти и с диска

Где используется RabbitMQ?

В контексте микросервисов протокол AMQP и его реализацию в RabbitMQ часто используют для **асинхронного взаимодействия** между сервисами.

RabbitMQ используют для обмена данными между серверами (сервер-сервер).

Брокер сообщений – стратегия «издатель-подписчик» позволяет создавать новостные ленты, групповые чаты

Ключевые функции

- **Безопасность** - поддержка аутентификации, авторизации, LDAP и TLS через Плагины RabbitMQ.

- **Надежность** - подтверждает, что сообщение было успешно доставлено посреднику сообщений, и подтверждает, что сообщение было успешно обработано потребителем.

- **Интероперабельность** - сообщение передается в виде потока байтов, поэтому любые клиенты могут работать с ним независимо от каких-либо языков.

RabbitMQ vs Apache Kafka: что выбрать?

RabbitMQ

- Вам важна гибкость в маршрутизации сообщений внутри системы. Инструментарий для построения путей доставки данных в RabbitMQ способен решить даже самые хитрые сценарии в организации потоков событий.
- Вам важен сам факт доставки сообщений, но порядок доставки не принципиален. Настройки RabbitMQ его не гарантируют: если вы сначала отправили сообщение №1, а затем сообщение №2, то подписчику они могут прийти в обратном порядке.

RabbitMQ vs Apache Kafka: что выбрать?

Apache Kafka

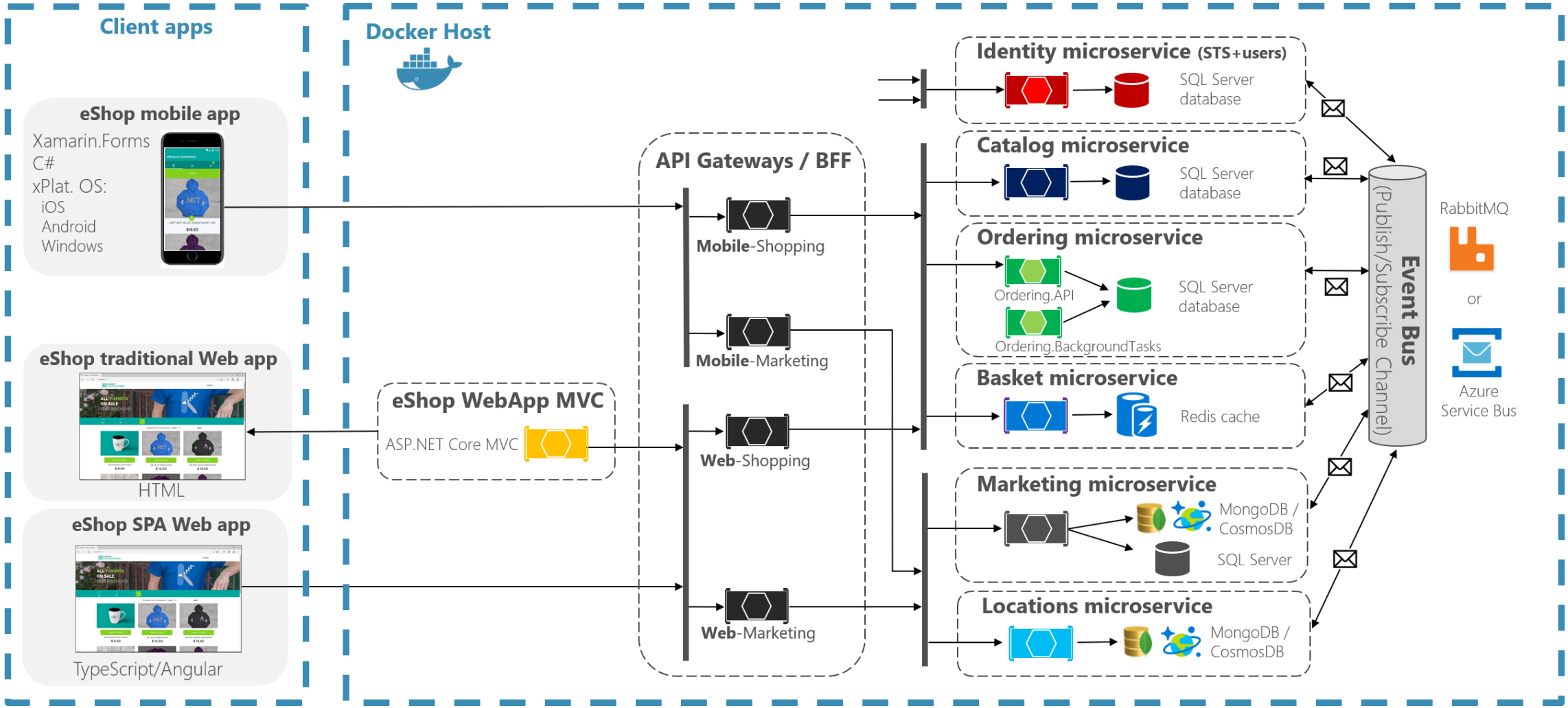
- Apache Kafka однозначно подходит, если вы работаете с big data. Репликация и параллельная обработка теоретически могут масштабировать вашу систему до бесконечности. Плюс Kafka выигрывает по производительности: может достичь пропускной способности в миллионы сообщений в секунду даже при ограниченных ресурсах.
- Решение позволяет программистам извлекать из очереди сообщения (Message Queuing) за любой момент времени.
- В некоторых системах порядок совершения событий имеет критическое значение — например, груз не может быть доставлен до того, как был отправлен. В таких системах необходимо использовать Apache Kafka: он гарантирует порядок доставки сообщений.

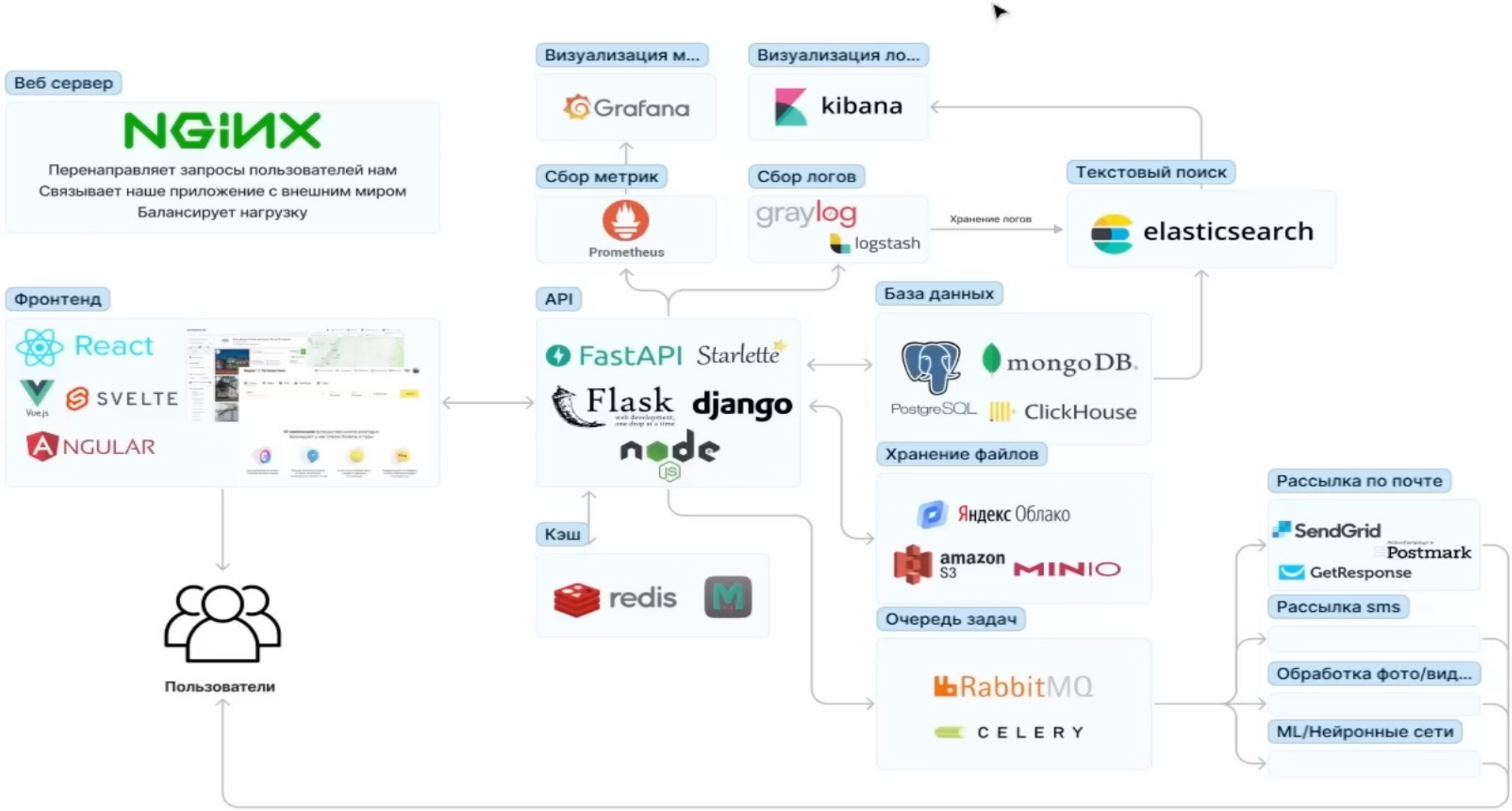
Итог

Выбирайте **RabbitMQ**, если вам нужна надежность и гибкость маршрутизации, а порядок доставки сообщений безразличен. **Apache Kafka** подойдет, если работаете с большими нагрузками, вам важна масштабируемость, доставка сообщений в правильном порядке и возможность просматривать историю сообщений.

eShopOnContainers reference application

(Development environment architecture)




Веб сервер
NGINX
Перенаправляет запросы пользователей нам
Связывает наше приложение с внешним миром
Балансирует нагрузку
Визуализация м...
Grafana
Визуализация ло...
kibana
Сбор метрик
Prometheus
Сбор логов
graylog
logstash
Хранение логов
Текстовый поиск
elasticsearch
Фронтенд
React
Vue.js
SVELTE
ANGULAR
API
FastAPI
Starlette
Flask
django
node
JS
База данных
mongoDB.
PostgreSQL
ClickHouse
Хранение файлов
Яндекс Облако
amazon S3
MINIO
Кэш
redis
Очередь задач
RabbitMQ
CELERY
Рассылка по почте
SendGrid
Postmark
GetResponse
Рассылка sms
Обработка фото/вид...
ML/Нейронные сети
Пользователи

Благодарю
за внимание