OTUS

ОНЛАЙН-ОБРАЗОВАНИЕ

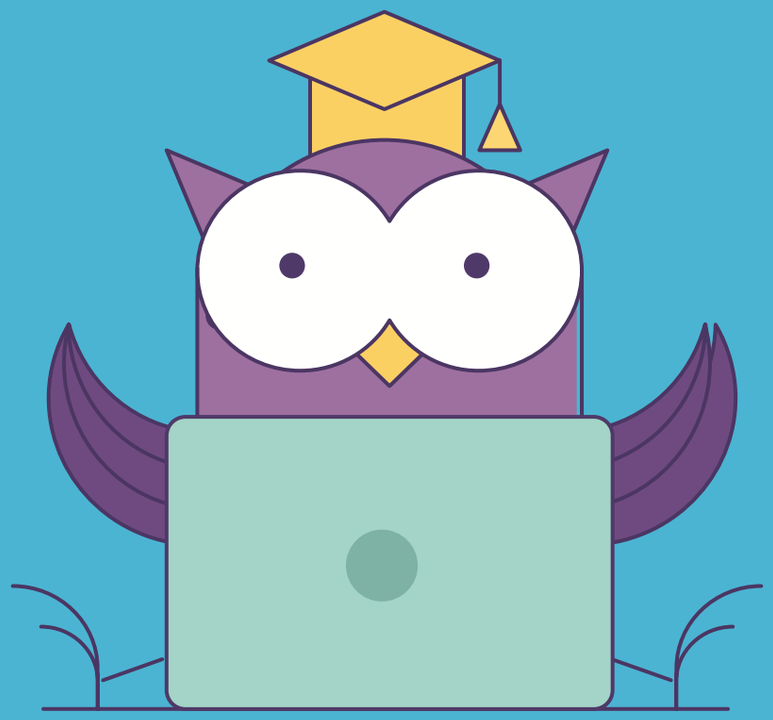
OTUS

RabbitMQ

Курс «Linux. Виртуализация и кластеризация»

Цыкунов Алексей

Меня хорошо слышно && видно?



Напишите в чат, если есть проблемы!

Ставьте + если все хорошо
Ставьте - если есть проблемы

Цели занятия

- Разобрать преимущества брокеров сообщений для высоконагруженных систем
- Разобрать какими инструментами решать задачи отложенного исполнения и построения очередей
- Разобраться в принципах работы инструментов
    - RabbitMQ
    - Kafka
- Понять какие еще возможности есть у этих инструментов
- Понять для чего и в каких случаях вам нужно применять эти инструменты

Примеры задач

- Асинхронные ответы на долгие запросы
    - построение отчетов
    - аналитические, бухгалтерские запросы
- Асинхронная обработка очереди запросов
- Передача данных между приложениями, микросервисами

Варианты решений

- использование базы данных
  - медленно, неэффективно
- использование   IPC, shared memory
  - сложная реализация, блокировки
  - отсутствие кластеризации
- использование систем доставки сообщений
  - Rabbitmq
  - Kafka
  - NATS
  - Redis (pub/sub, streams)

Требования к брокерам

- гарантия доставки сообщений
  - at least once delivery
  - at most once delivery
  - exactly once
- Порядок передачи сообщений
- Управлению размером очереди (ограничение на передачу)
- Зеркалирование
- Масштабирование

протокол AMPQ

AMQP (Advanced Message Queuing Protocol) обеспечивает взаимодействие между **клиентами** и **брокерами** (промежуточным ПО для обмена сообщениями).

Протокол обеспечивает:

- Надежность доставки сообщений
- Высокую скорость доставки сообщений
- Подтверждение приема сообщений

Терминология AMPQ

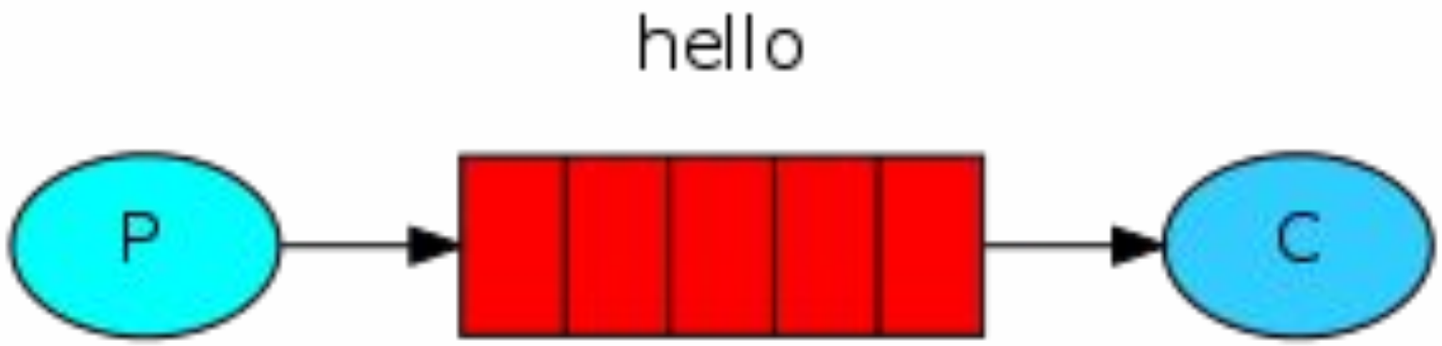
- **Брокер** – это приложение, реализующее модель AMQP, которое принимает соединения клиентов для маршрутизации сообщений и т.п.
- Сообщение (**message**) – это единица передаваемых данных (включая полезные данные и атрибуты сообщения).
- Потребитель (**consumer**) – приложение, которое получает сообщения из очередей.
- Производитель (**producer**)– приложение, которое отправляет сообщения в очередь через обмен.


PRODUCER
Publish
BROKER
RabbitMQ
Consume
Subscribe
CONSUMER

Характеристики rabbitmq

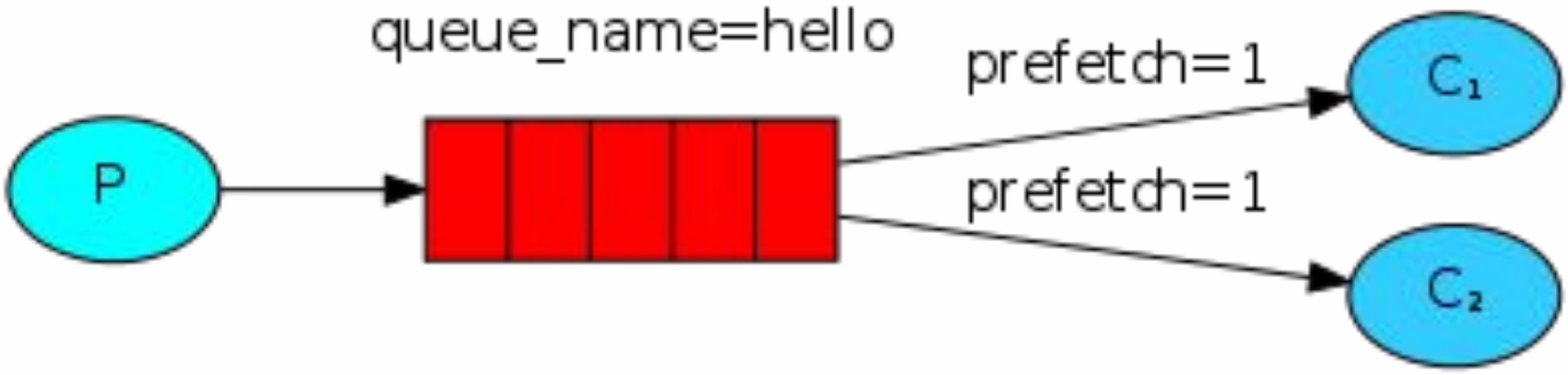
- гарантия доставки сообщений
  - at least once delivery
  - at most once delivery
- Гарантирует порядок передачи сообщений (FIFO)
- Возможность сохранения на диск
- Подтверждение отправки, подтверждение получения
- Ограничение кол-ва отправляемых сообщение
- Управление поведением неполученных сообщений
- Кластеризация
- Зеркалирование

Варианты взаимодействия: consume

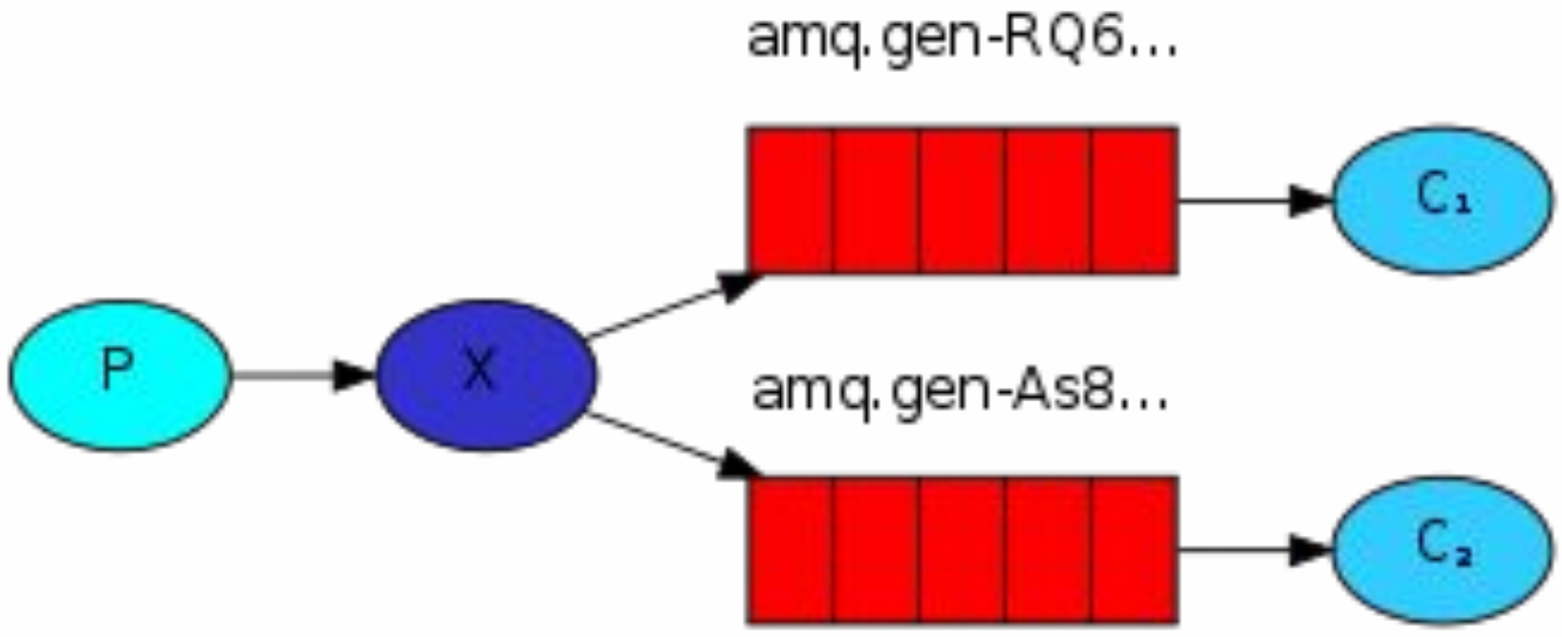


Балансировка

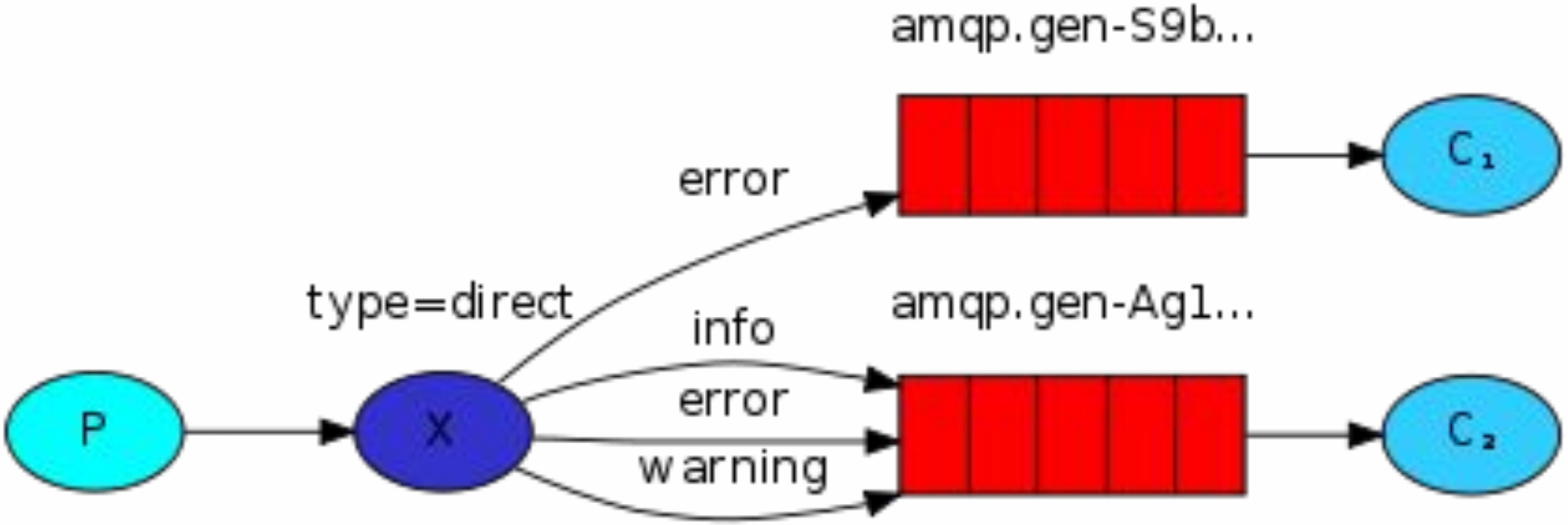
при выставленном **prefetch=1** Rabbit отправляет следующее сообщение только при получении подтверждения о получении предыдущего



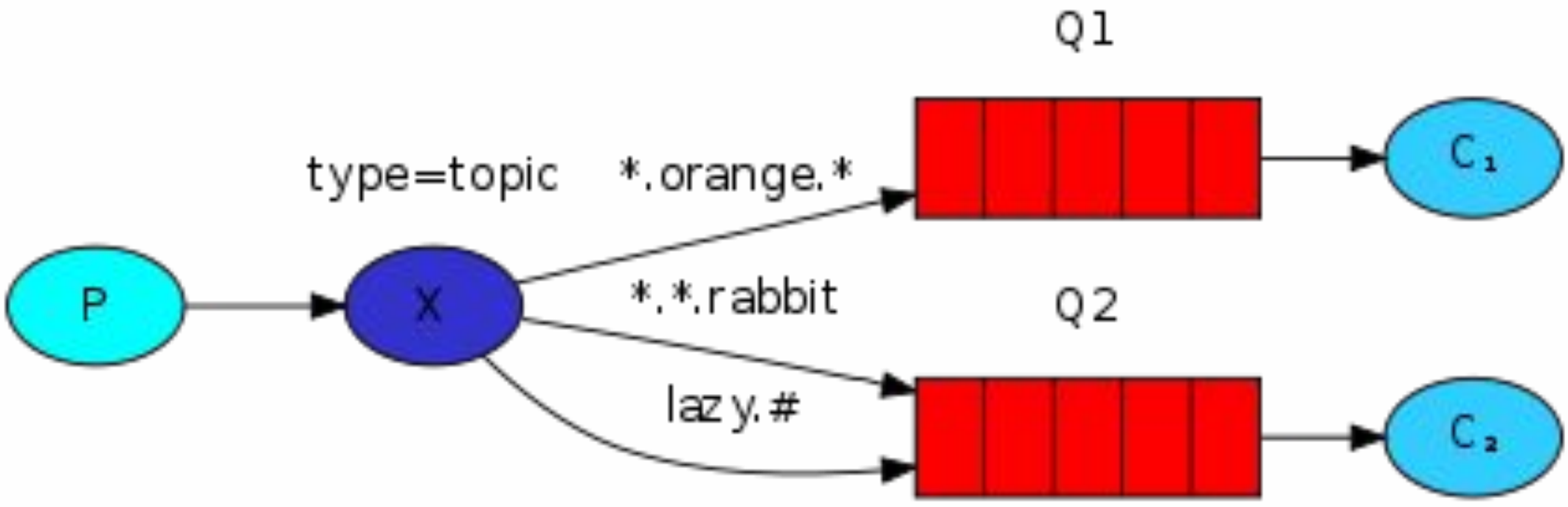
Варианты взаимодействия: pub/sub



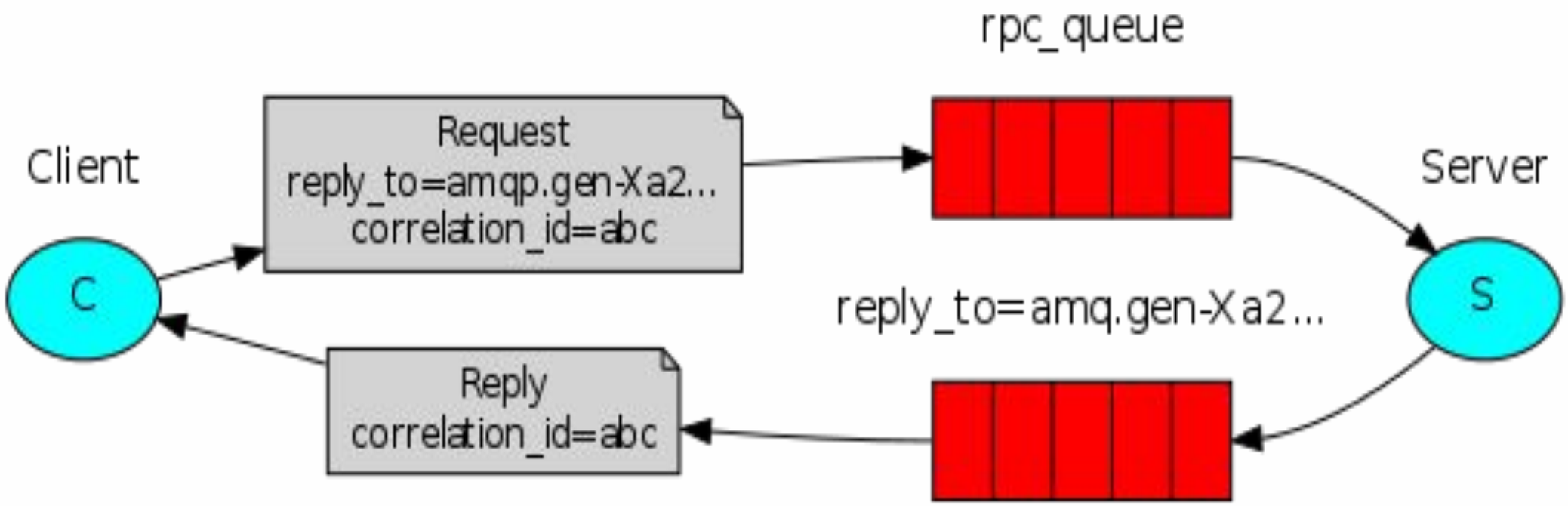
Варианты взаимодействия: роутинг/фильтрация



Варианты взаимодействия: роутинг/фильтрация



Варианты взаимодействия: RPC



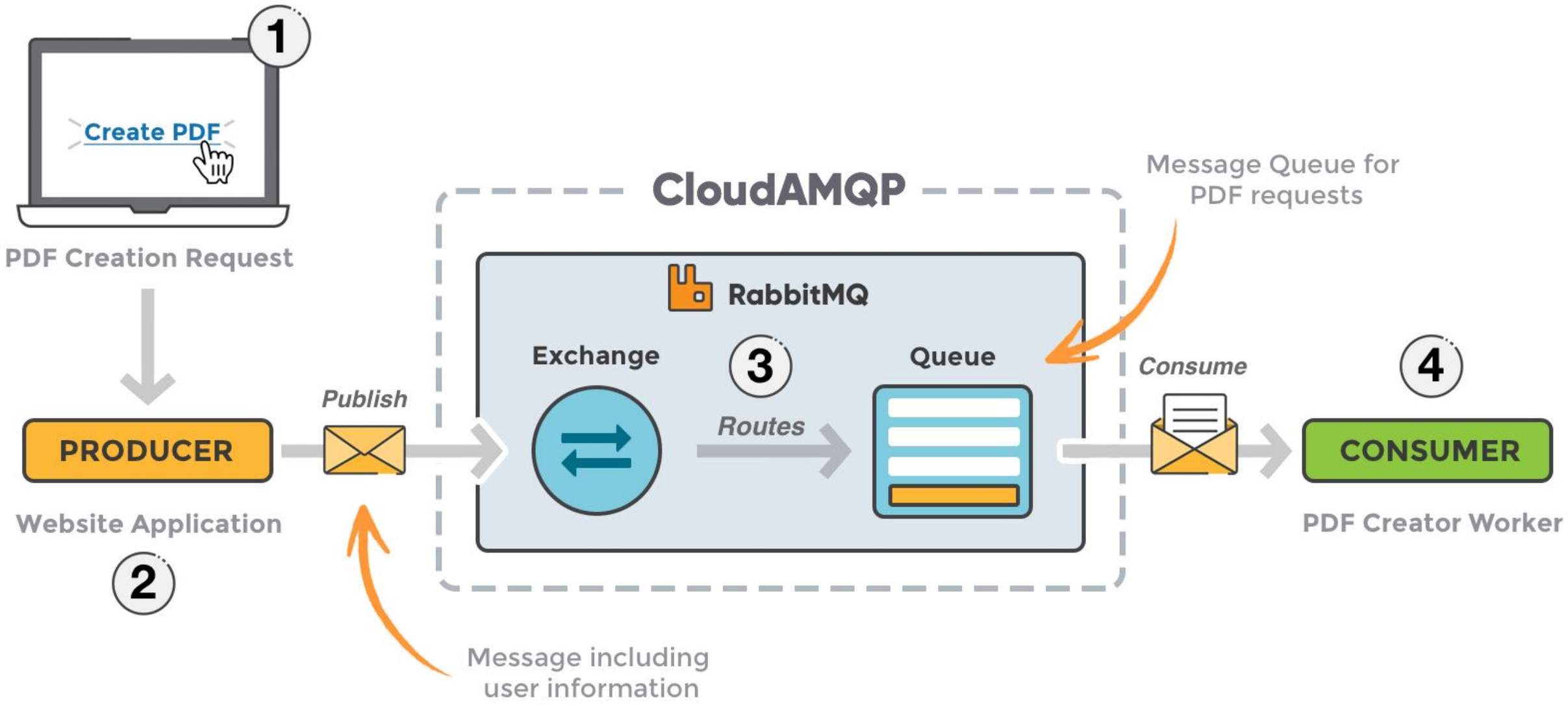
RabbitMQ: Терминология

- **Producer, consumer, message -** (рассмотрели ранее)

- **Queue**: Буфер который хранит сообщения
- **Connection**: TCP соединение между приложениями и менеджером очередей
- **Channel**: Виртуальное соединение внутри соединения. Когда вы публикуете или получаете сообщения через очередь – это все делается в канале.
- **Exchange**: Получает сообщение от поставщика и отправляет его в очередь. Зависит от типа.

RabbitMQ: Терминология

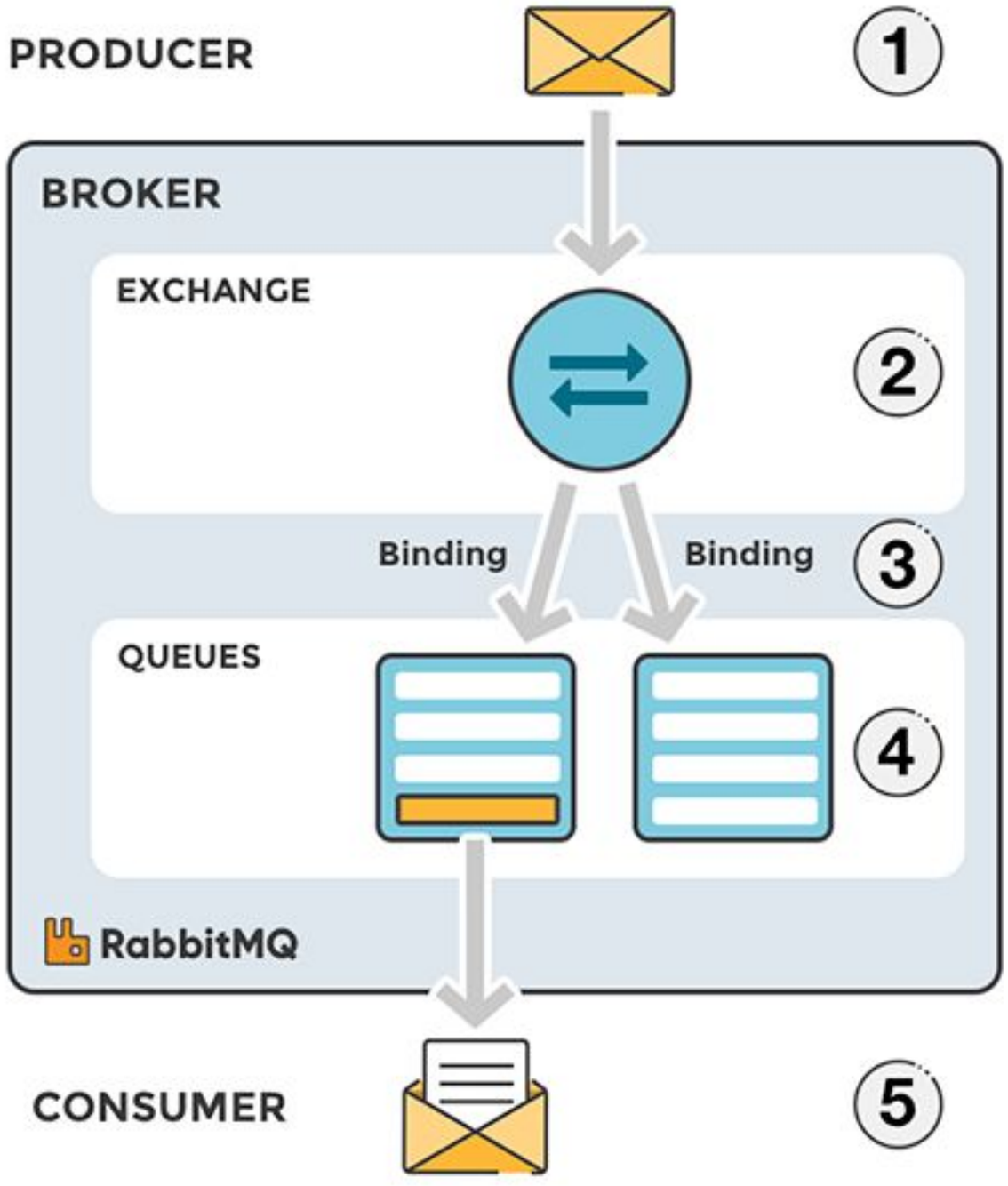
- **Binding**: Связь между очередью и обработчиком сообщений
- **Routing key**: Ключ на который смотрит обработчик и решает в какую очередь перенаправить сообщение
- **Users**: Возможность подключится к брокеру сообщений с помощью имени пользователя и пароля.
- **Vhost, virtual host**: Способ разделения приложений используя один и тот же экземпляр RabbitMQ.


1
Create PDF
PDF Creation Request
CloudAMQP
Message Queue for
PDF requests
RabbitMQ
Exchange
3
Queue
Publish
Routes
Consume
4
PRODUCER
CONSUMER
Website Application
PDF Creator Worker
2
Message including
user information

RabbitMQ: Exchange

Обработчик (EXCHANGE) сообщений

Сообщения публикуются в очередь не на прямую, вместо этого, поставщик шлет сообщение обработчику который отвечает за перенаправление сообщения в нужную очередь с помощью биндинга ключей роутинга.



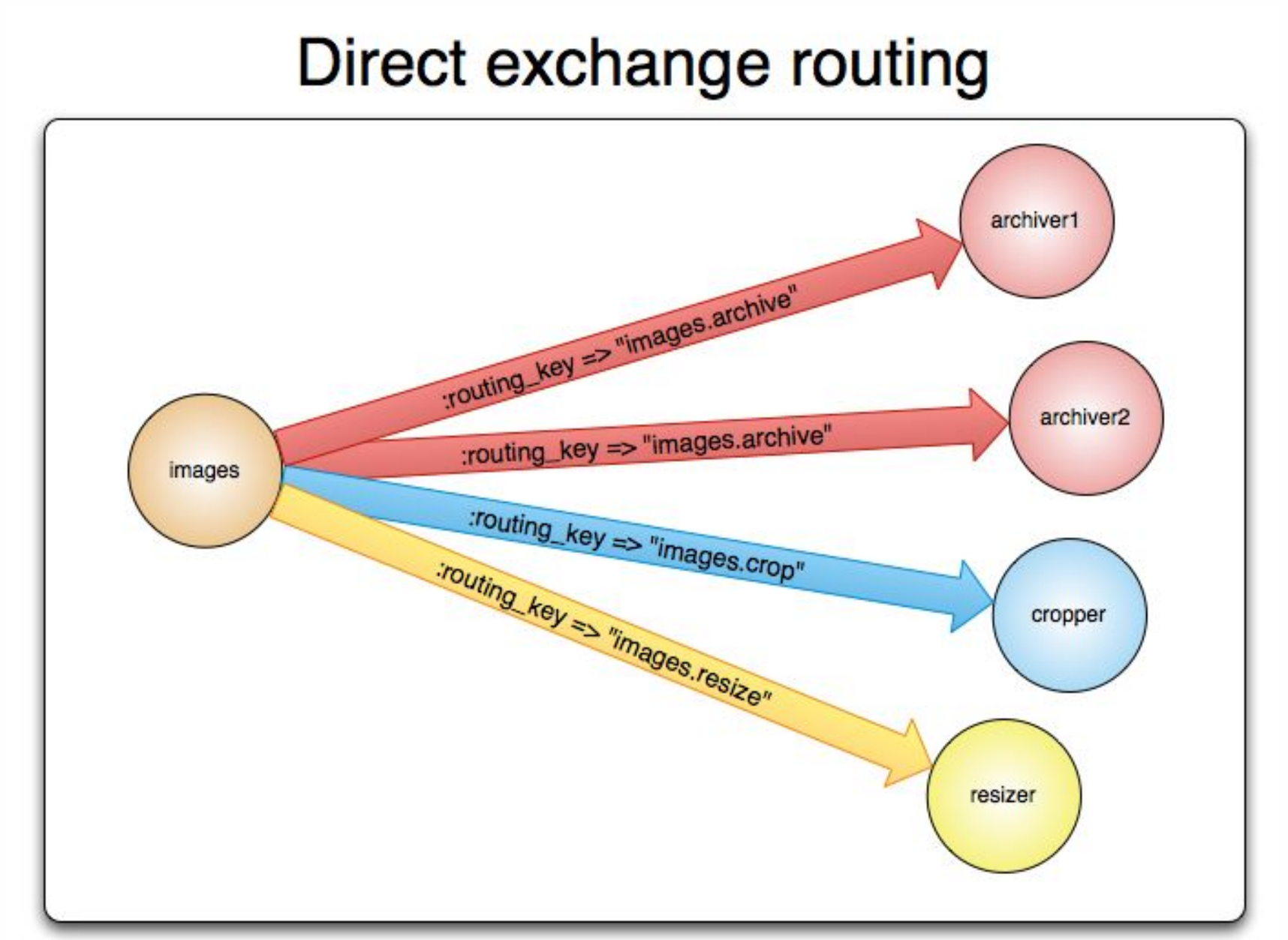
1. Поставщик публикует сообщение в Exchange. Типы exchange: **direct, fanout, topics, headers**
2. Обработчик получает сообщение и отвечает за его перенаправление. Обработчик берет различные атрибуты, такие как, ключ роутинга, зависимость на тип обмена и другие.
3. Создается связь между обработчиком и очередью
4. Сообщение остается в очереди до тех пор пока не будет обработано получателем.
5. Получатель обрабатывает сообщение

Ключ маршрутизации — значение, которое указывается при публикации сообщения в точку обмена, служит для определения очередей в которые попадет сообщение.

Заголовок сообщения — набор аргументов вида ключ-значение связанных с сообщением.

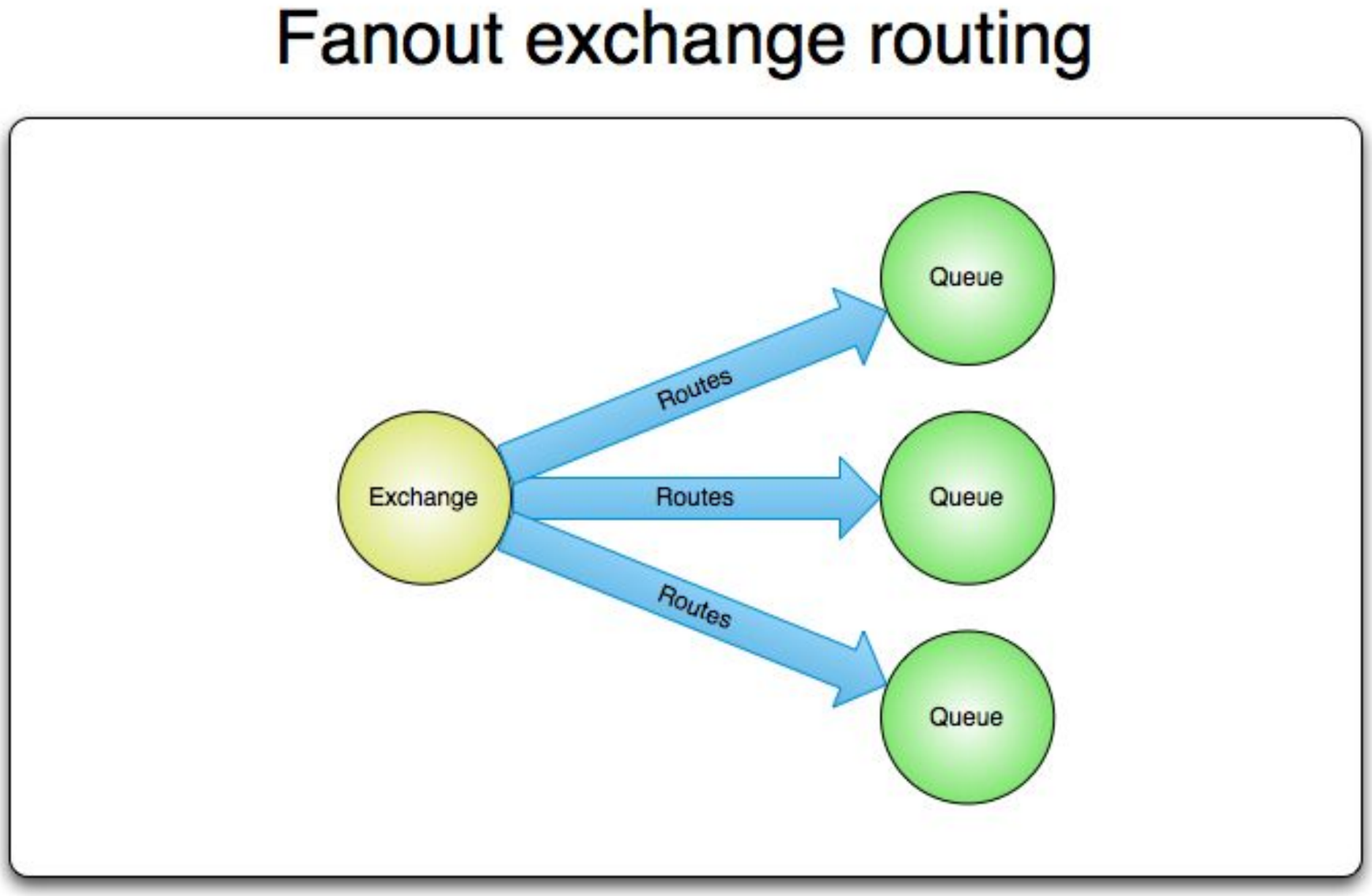
Типы обработчика: direct

- **direct** — сообщения попавшие в эту точку обмена будут скопированы только в те очереди, которые связаны с точкой обмена строгим ключом маршрутизации.



Типы обработчика: fanout

- **fanout** — сообщение поступившее в точку обмена копируется во все привязанные очереди, без проверки ключа маршрутизации или заголовка сообщения.

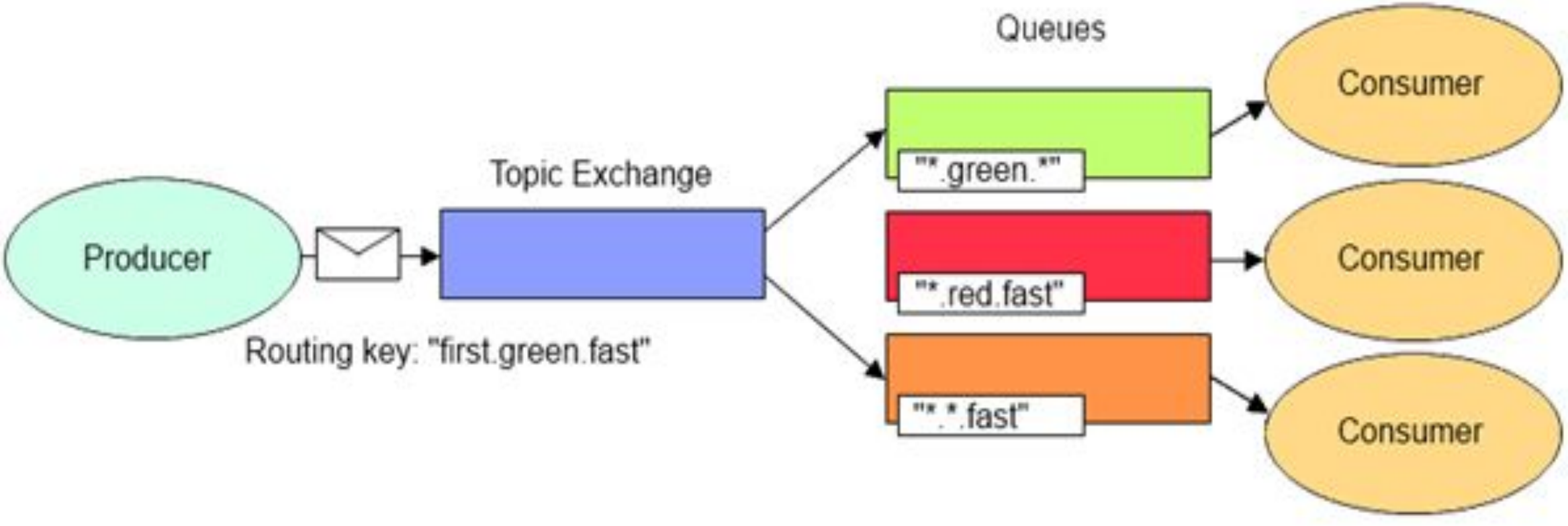


RabbitMQ: Типы обработчика

OTUS

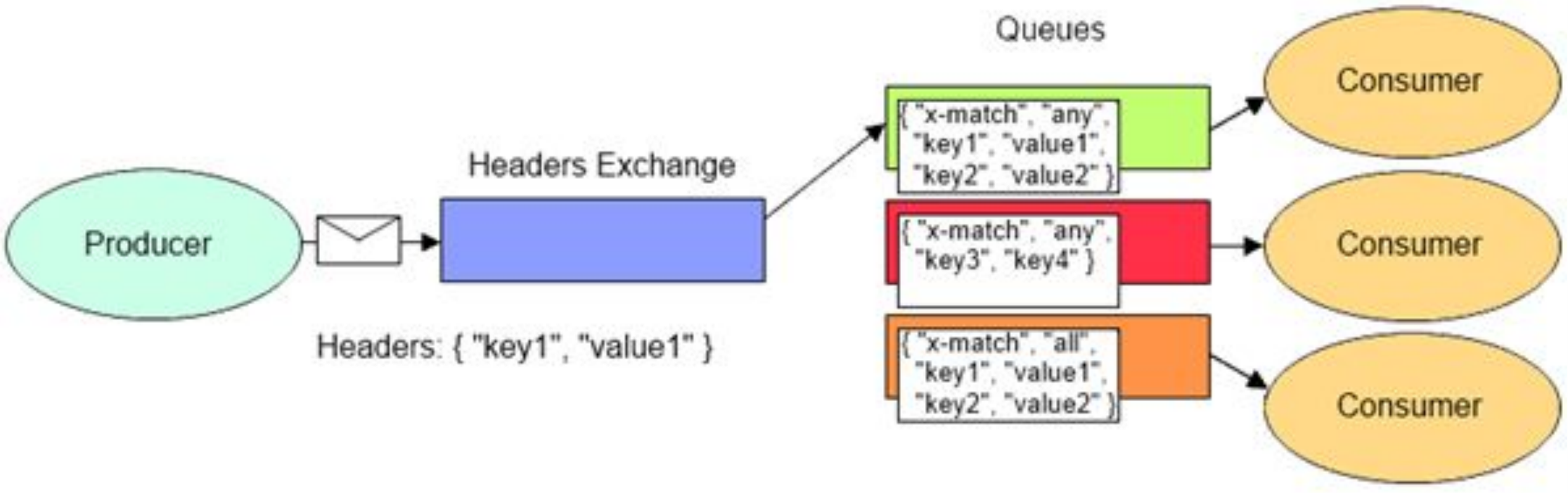
- **topic** — ключ маршрутизации может быть составным, и задаваться в виде паттерна, для чего существуют два специальных символа: * — обозначает одно слово, # — одно или несколько слов. Слова разделяются точкой.

  Пример: routingKey = "*.database"



RabbitMQ: Типы обработчика

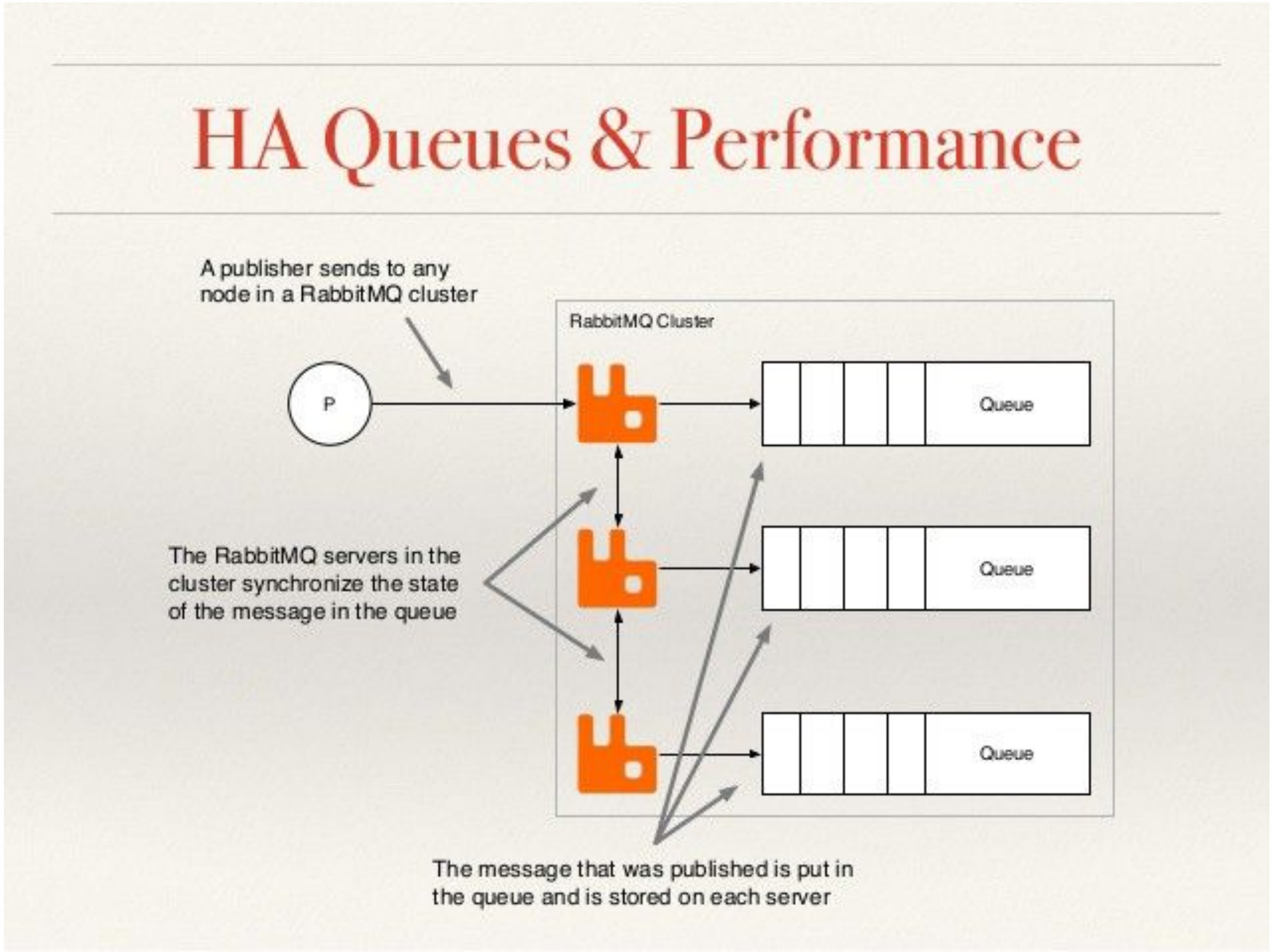
- **headers** — очередь связывается с обработчиком по заголовку сообщения, указывается условие, какие аргументы и их значения ожидаются



RabbitMQ: флаги очередей

- **auto_delete** — если очередь пустая и к ней нет активных подключений, очередь автоматически удаляется
- **durable** — устойчивая очередь, сообщения не теряются при рестарте rabbitMQ (или внезапной перезагрузке), при публикации и до окончания отдачи хранятся в базе данных
- **exclusive** — очередь предназначена для не более чем одного подключения единовременно
- **passive** — при объявлении очереди пассивной, при обращении клиента сервер будет считать что очередь уже создана, т.е. не будет автоматически создавать ее в случае отсутствия, этот вариант нужен если вы хотите обратиться к серверу не изменяя его состояние.
- **internal -** очередь между **exchanges**

RabbitMQ: кластер



RabbitMQ: кластер

Синхронизировать /var/lib/rabbitmq/.erlang.cookie

```
rabbitmqctl stop_app
```

```
rabbitmqctl join_cluster --ram rabbit@master
```

```
rabbitmqctl start_app
```

По умолчанию очереди не синхронизированы!

Включение зеркалирования очередей

```
# rabbitmqctl set_policy ha-all ".*" \
'{"ha-mode":"all","ha-sync-mode":"automatic"}'
```

RabbitMQ: память

По-умолчанию, в случае если RabbitMQ начинает использовать больше 40% от общего объема памяти, то все соединения блокируются.
При высвобождении памяти до требуемого уровня нормальный процесс работы RabbitMQ возобновляется. Порог используемой памяти можно переопределить:

rabbitmqctl set_vm_memory_high_watermark 0.5

или перманентно в **/etc/rabbitmq/rabbitmq.config**

[{rabbit, [{vm_memory_high_watermark, 0.5}]}].

Рефлексия

Напишите, пожалуйста, свое впечатление о вебинаре.

- Отметьте 3 пункта, которые вам запомнились с вебинара.
- Что вы будете применять в работе из сегодняшнего вебинара?

Заполните, пожалуйста, опрос в ЛК о занятии

Спасибо
за внимание!
До встречи в Slack и на вебинаре