



ОНЛАЙН-ОБРАЗОВАНИЕ



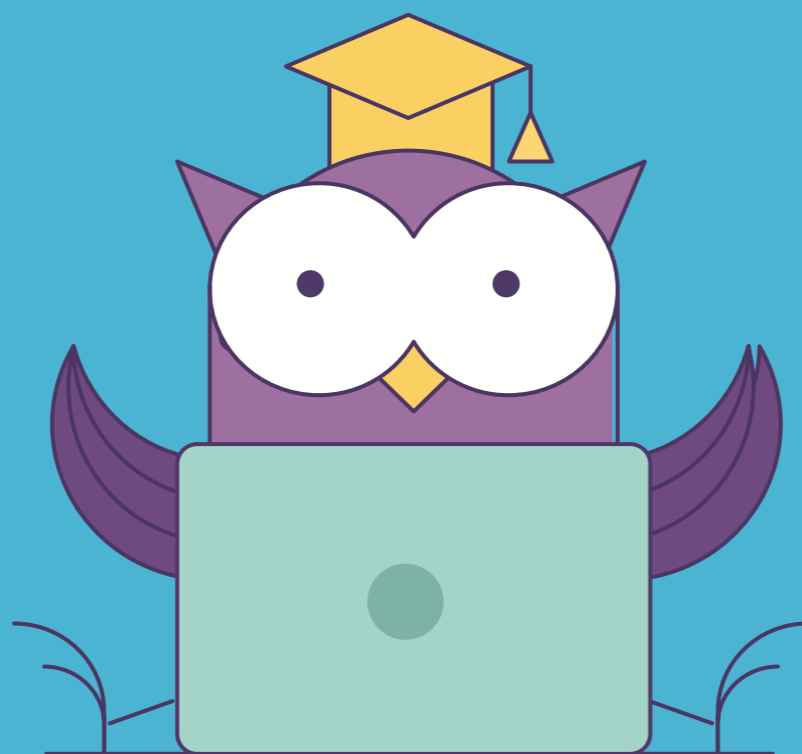
# Очереди и отложенное выполнение

Курс «Архитектор Highload»

Занятие № 15



# 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 AMQP

**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

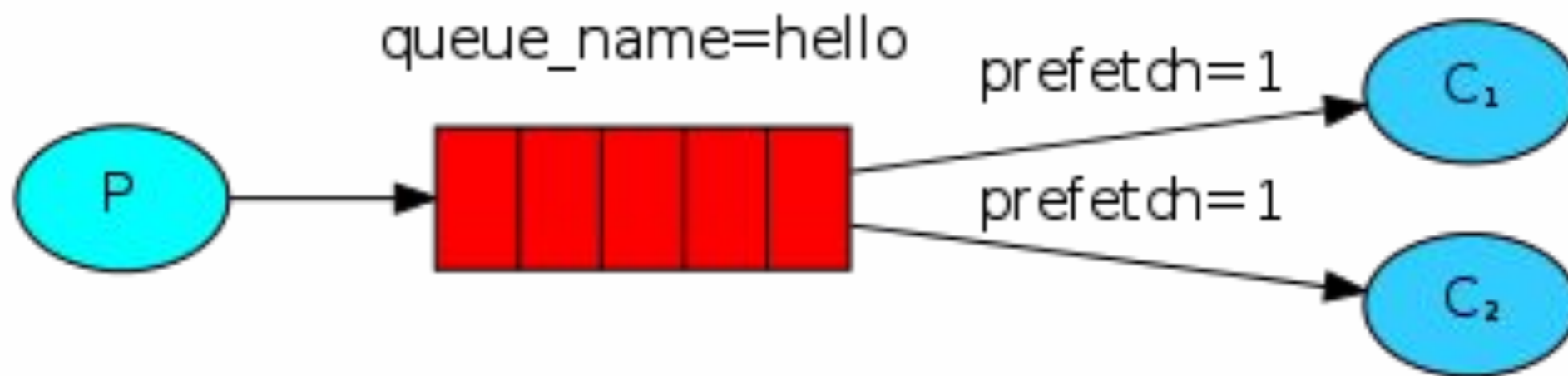
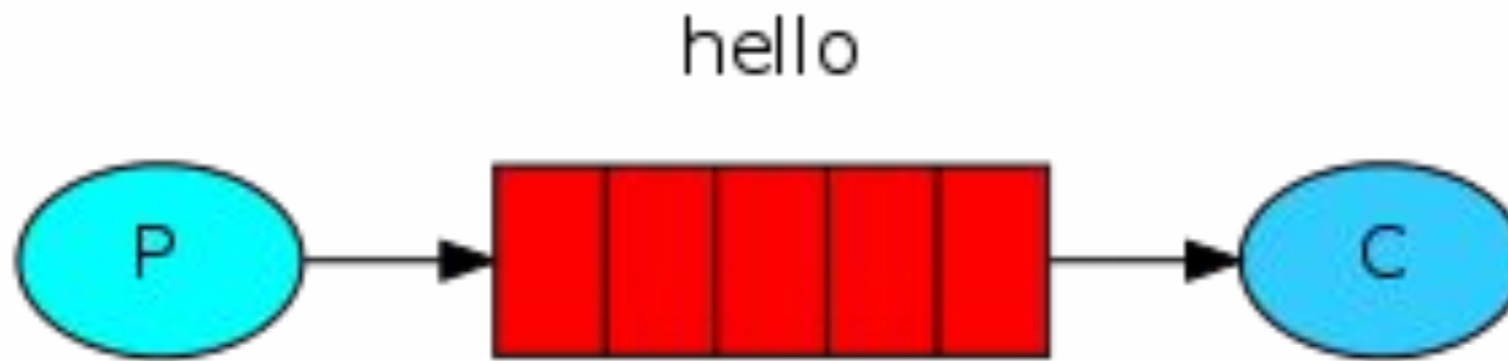
## Терминология AMQP

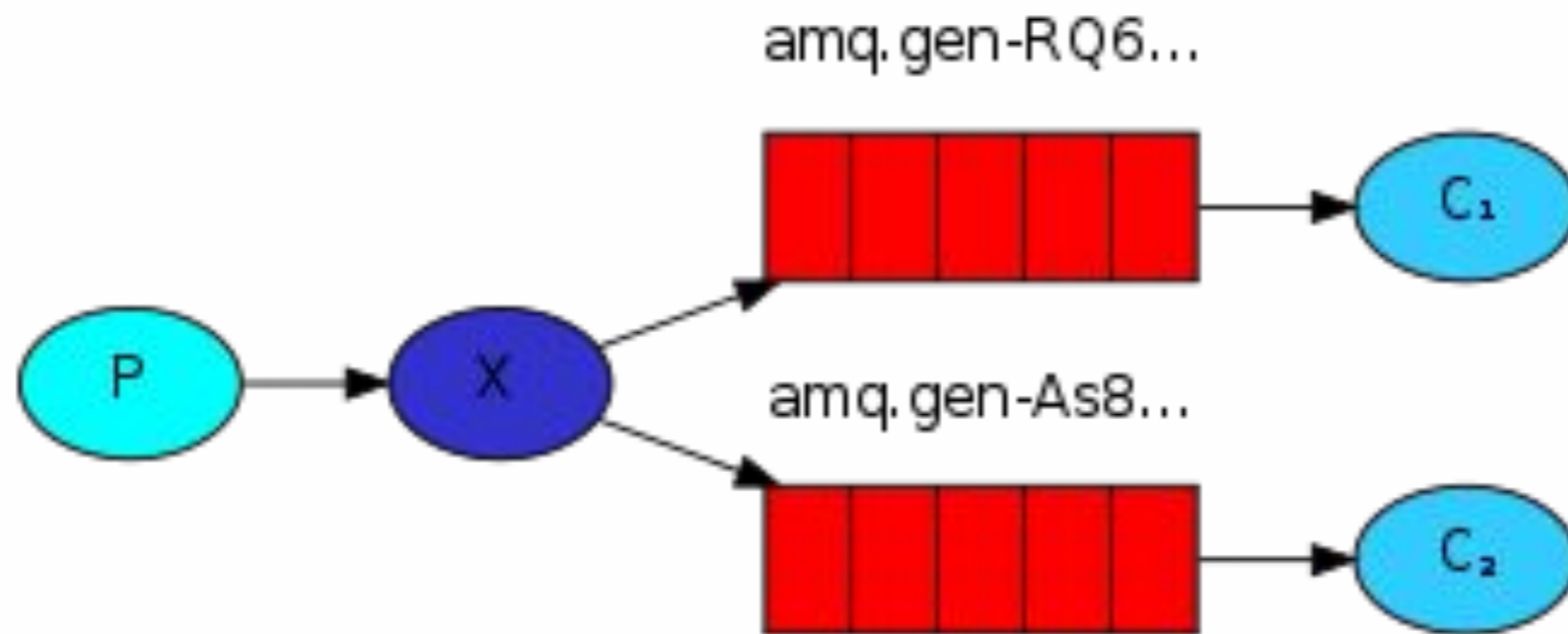
- **Брокер** – это приложение, реализующее модель AMQP, которое принимает соединения клиентов для маршрутизации сообщений и т.п.
- Сообщение (**message**) – это единица передаваемых данных (включая полезные данные и атрибуты сообщения).
- Потребитель (**consumer**) – приложение, которое получает сообщения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

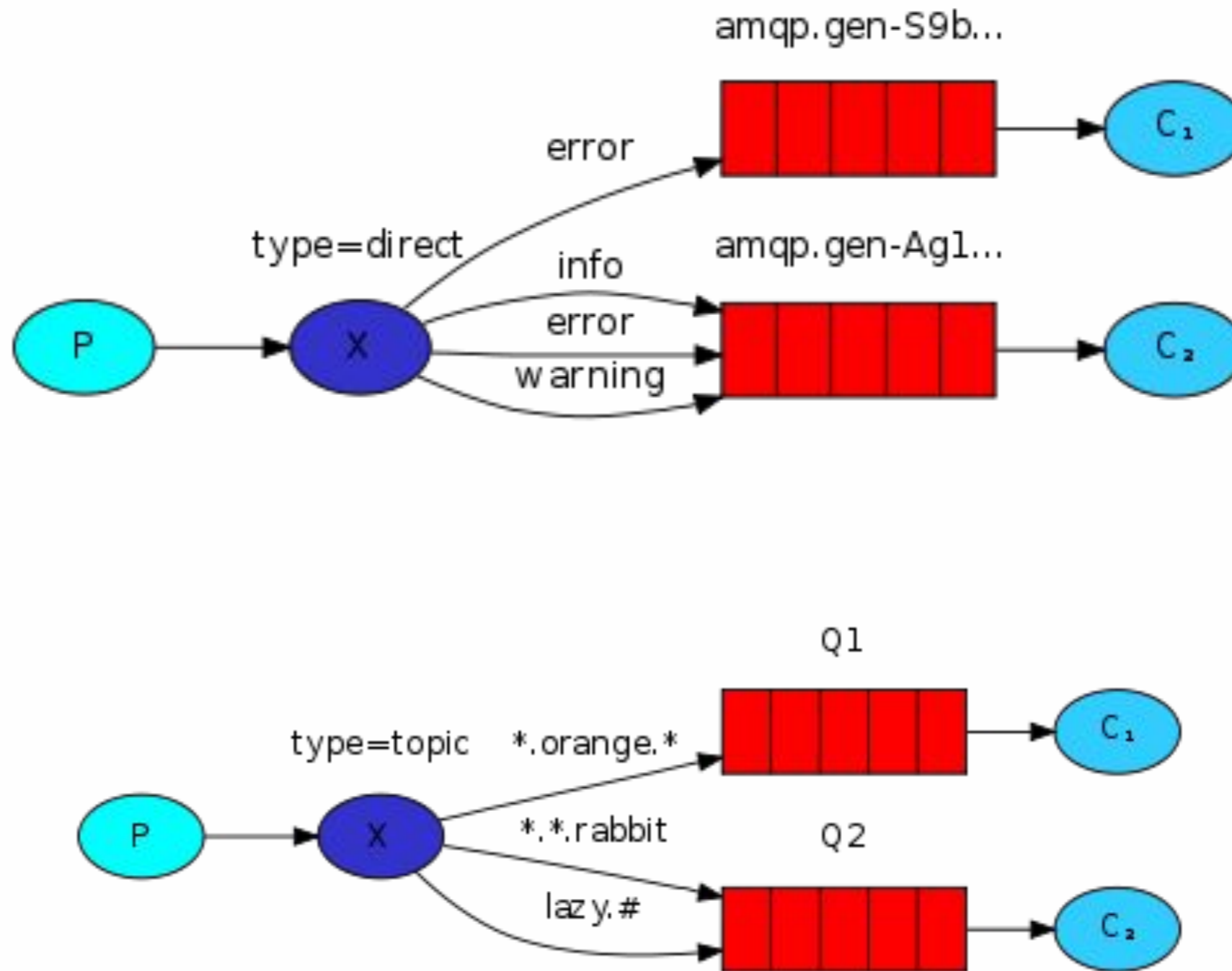


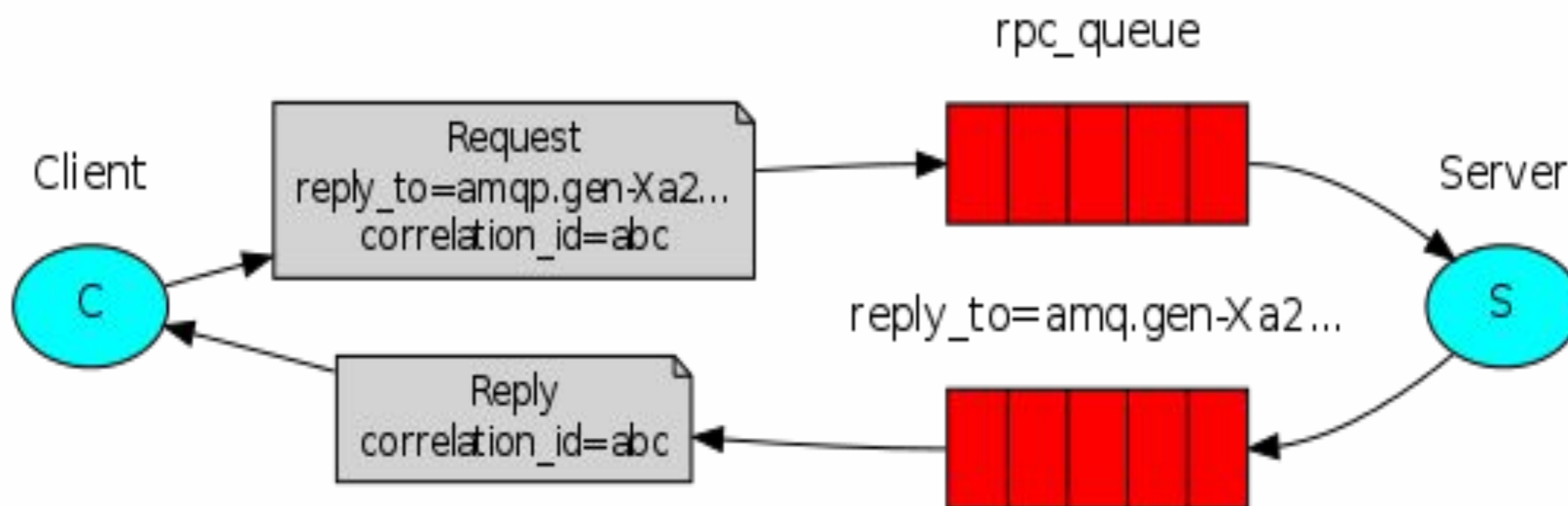
## Характеристики rabbitmq

- 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 сообщений
- Управление поведением не полученных сообщений
- Кластеризация
- Зеркалирование



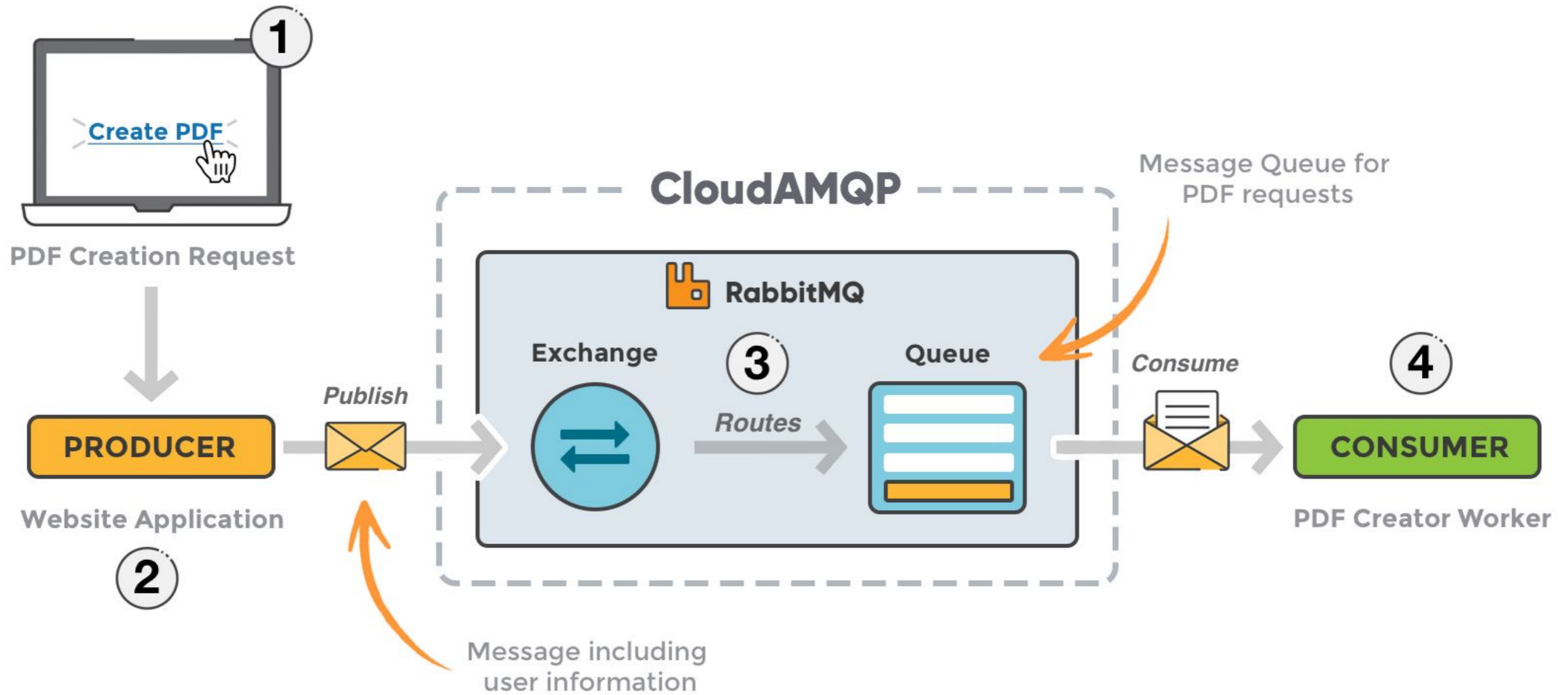






- **Producer, consumer, message** - (рассмотрели ранее)
- **Queue**: Буфер который хранит сообщения
- **Connection**: TCP соединение между приложениями и менеджером очередей
- **Channel**: Виртуальное соединение внутри соединения. Когда вы публикуете или получаете сообщения через очередь – это все делается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

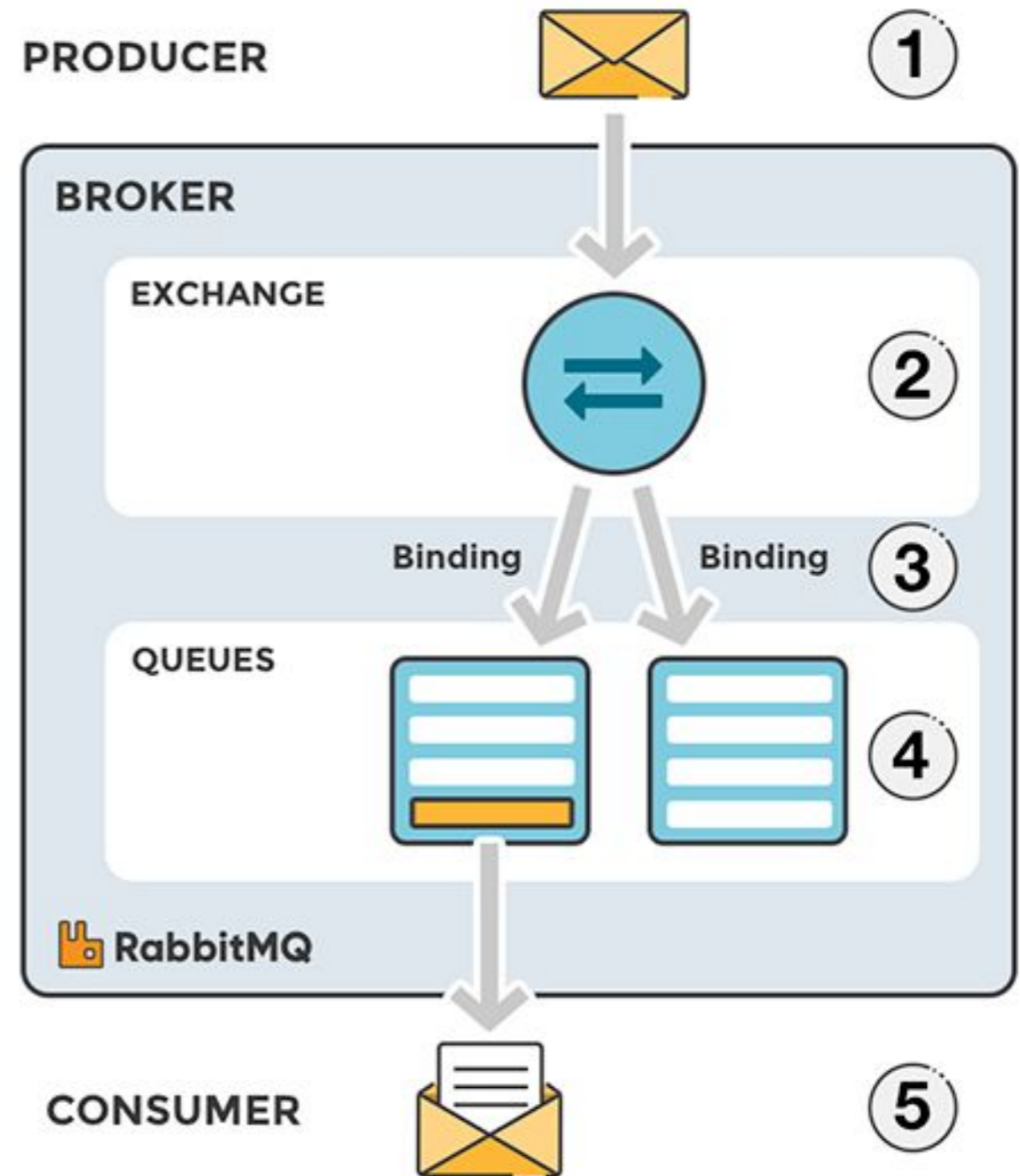
- **Binding:** Связь между очередью и обработчиком сообщений
- **Routing key:** Ключ на который смотрит обработчик и решает в какую очередь перенаправить сообщение
- **Users:** Возможность подключится к брокеру сообщений с помощью имени пользователя и пароля.
- **Vhost, virtual host:** Способ разделения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



# 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.



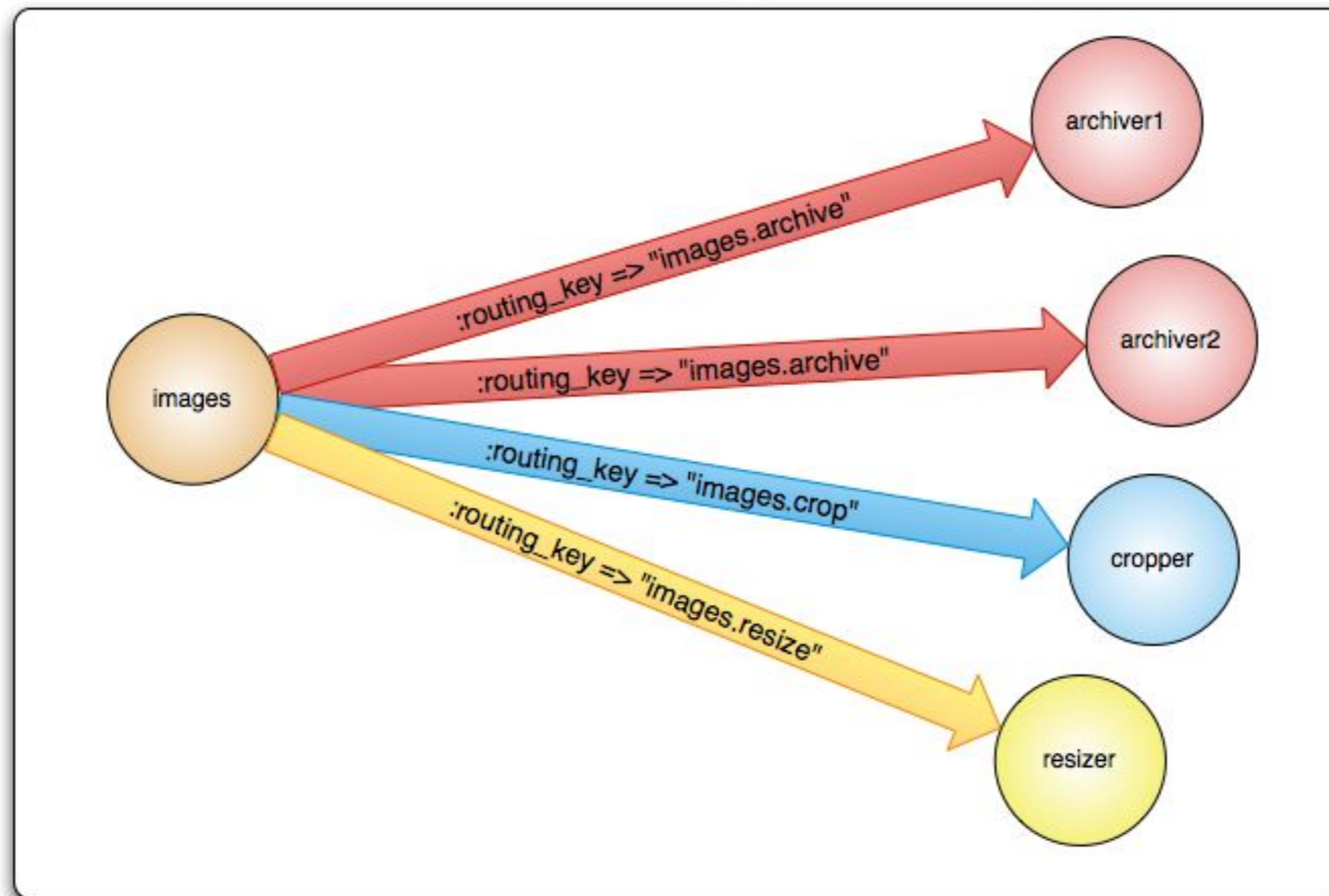
1. Поставщик публикует сообщение в Exchange. Типы exchange: **direct, fanout, topics, headers**
2. Обработчик получает сообщение и отвечает за его перенаправление. Обработчик берет различные атрибуты, такие как, ключ роутинга, зависимость на тип обмена и другие.
3. Создается связь между обработчиком 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.

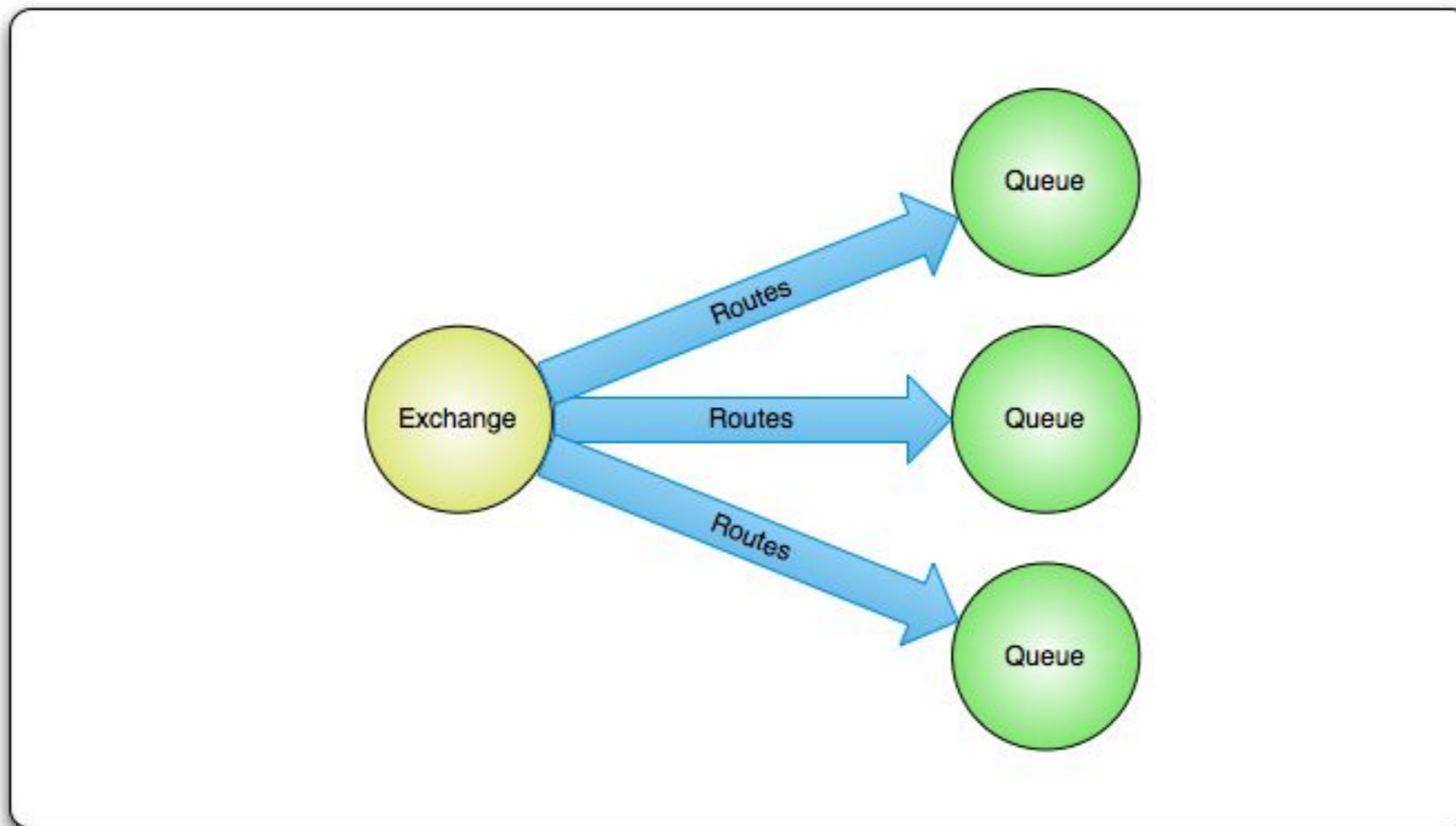
- **direct** — сообщения попавшие в эту точку обмена будут скопированы только в те очереди, которые связаны с точкой обмена строгим ключом маршрутизации.

## Direct exchange routing



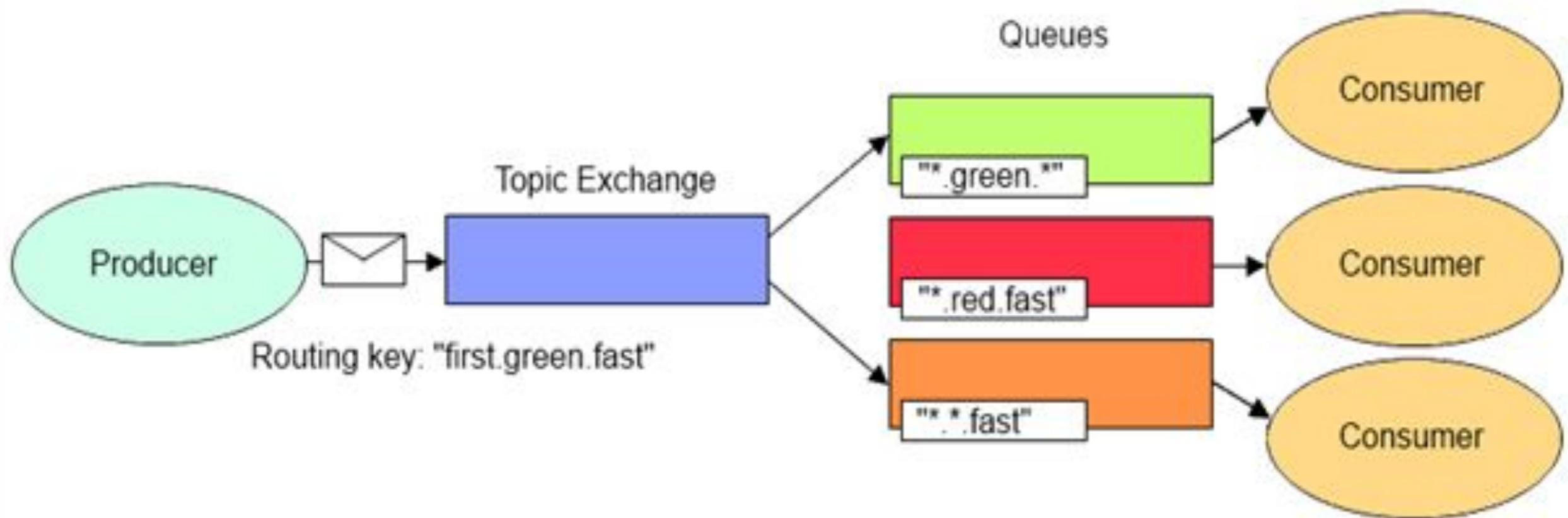
- **fanout** – сообщение поступившее в точку обмена копируется во все привязанные очереди, без проверки ключа маршрутизации или заголовка сообщения.

## Fanout exchange routing

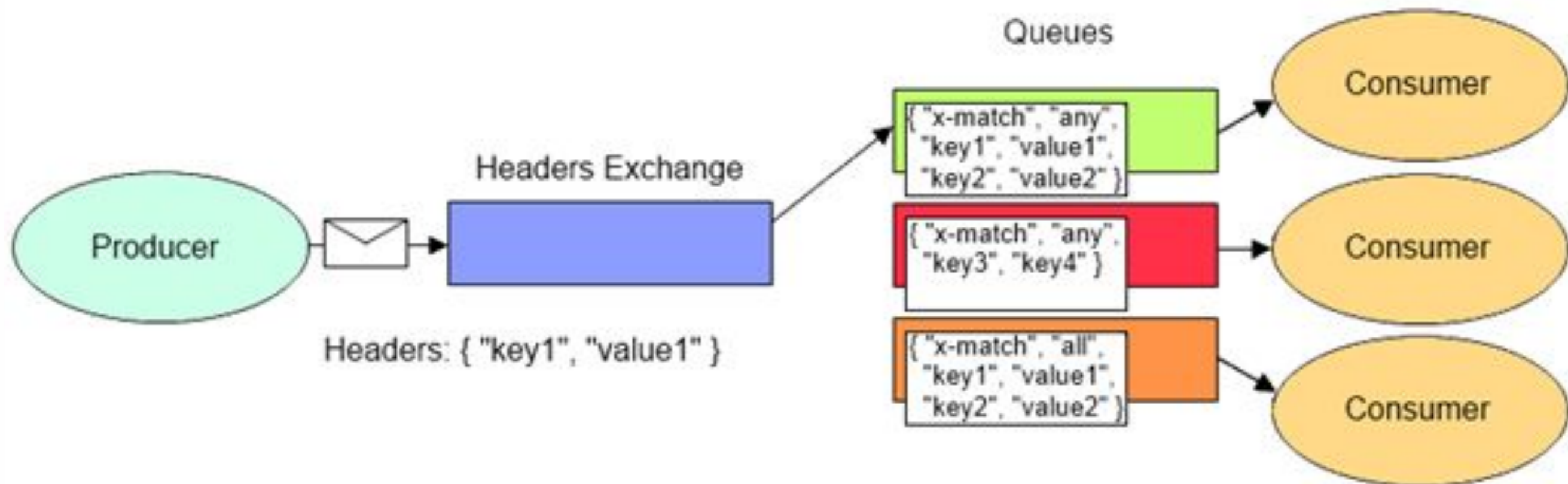


- **topic** — ключ маршрутизации может быть составным, и задаваться в виде паттерна, для чего существуют два специальных символа: \* — обозначает одно слово, # — одно или несколько слов. Слова разделяются точкой.

Пример: routingKey = "\*.database"

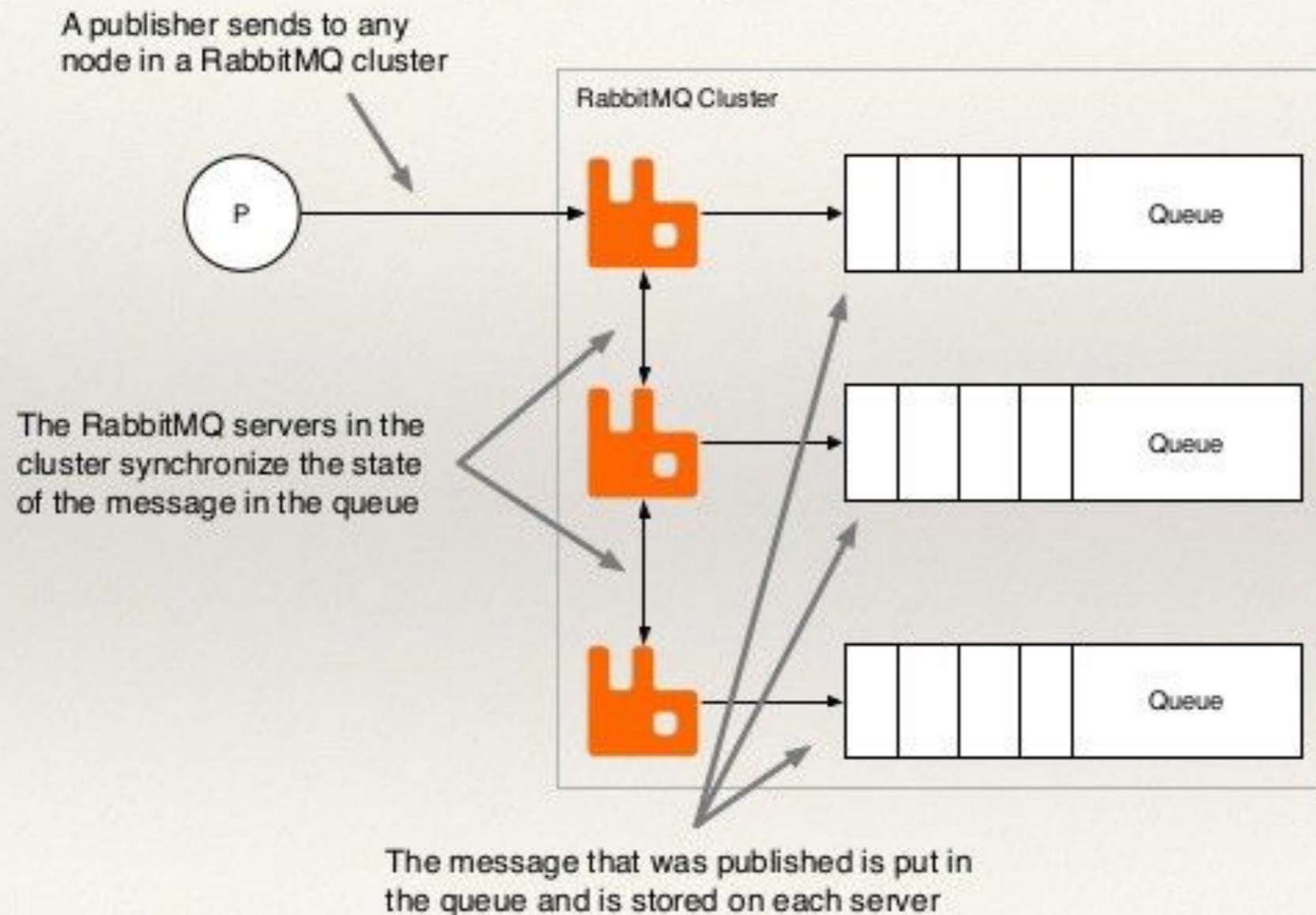


- **headers** – очередь связывается с обработчиком по заголовку сообщения, указывается условие, какие аргументы и их значения ожидаются



- **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 одновременно
- **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**

## HA Queues & Performance



**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"}
```

По-умолчанию, в случае если RabbitMQ начинает использовать больше 40% от общего объема памяти, то все соединения блокируются.

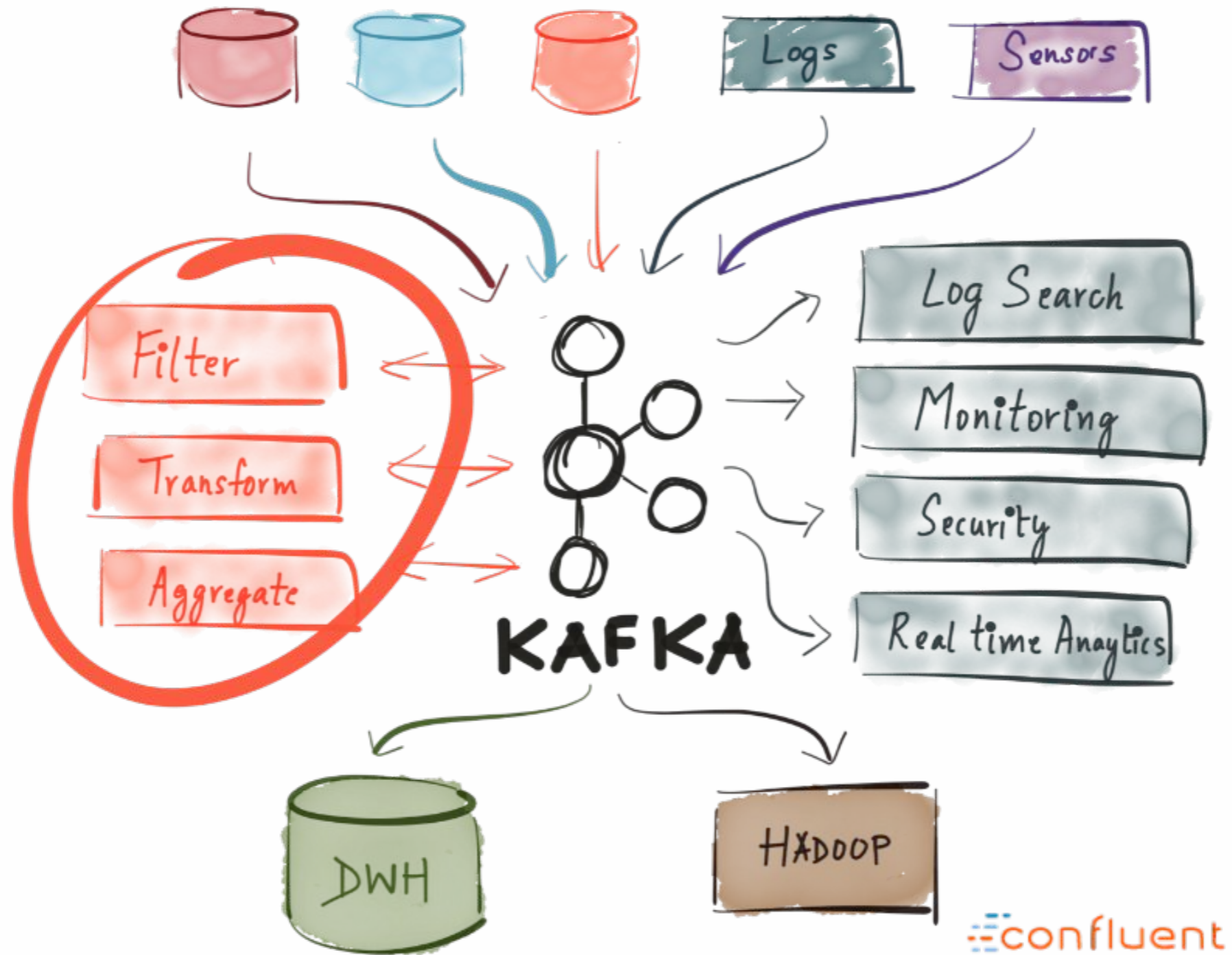
При высвобождении памяти до требуемого уровня нормальный процесс работы RabbitMQ возобновляется. Порог используемой памяти можно переопределить:

```
rabbitmqctl set_vm_memory_high_watermark 0.5
```

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

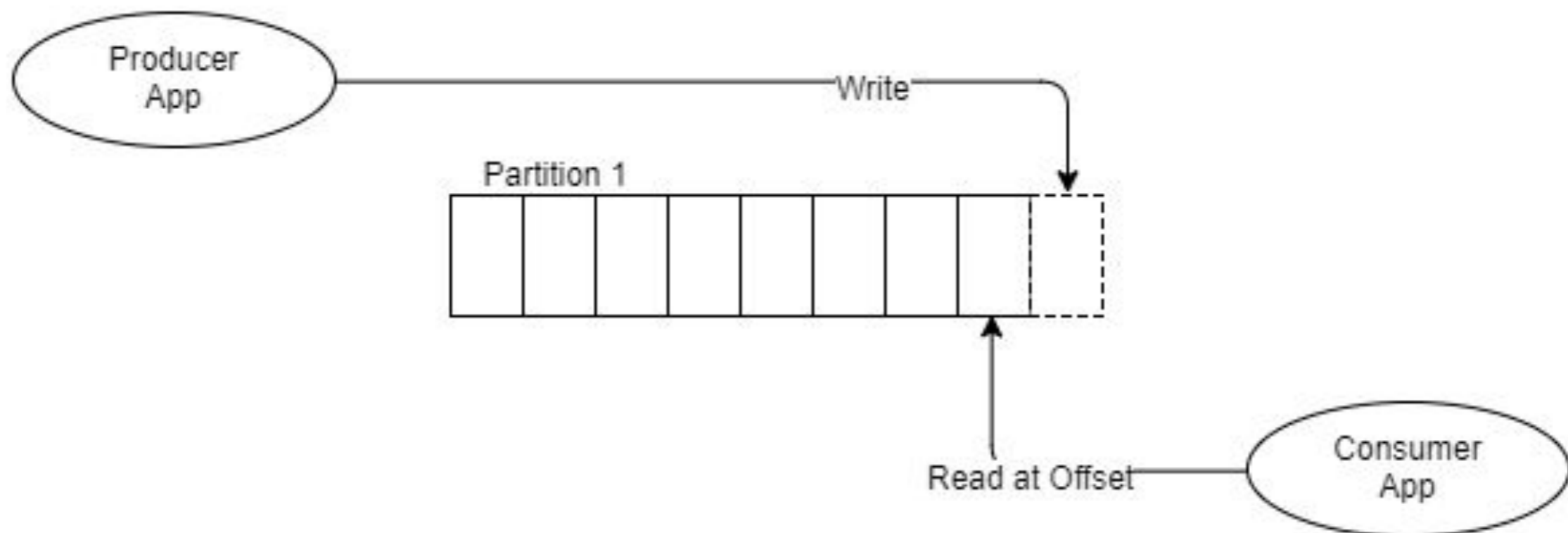
```
[{rabbit, [{vm_memory_high_watermark, 0.5}]}].
```





## Kafka — это

- **распределенный**
- **реплицированный**
- **журнал фиксации изменений (commit log).**

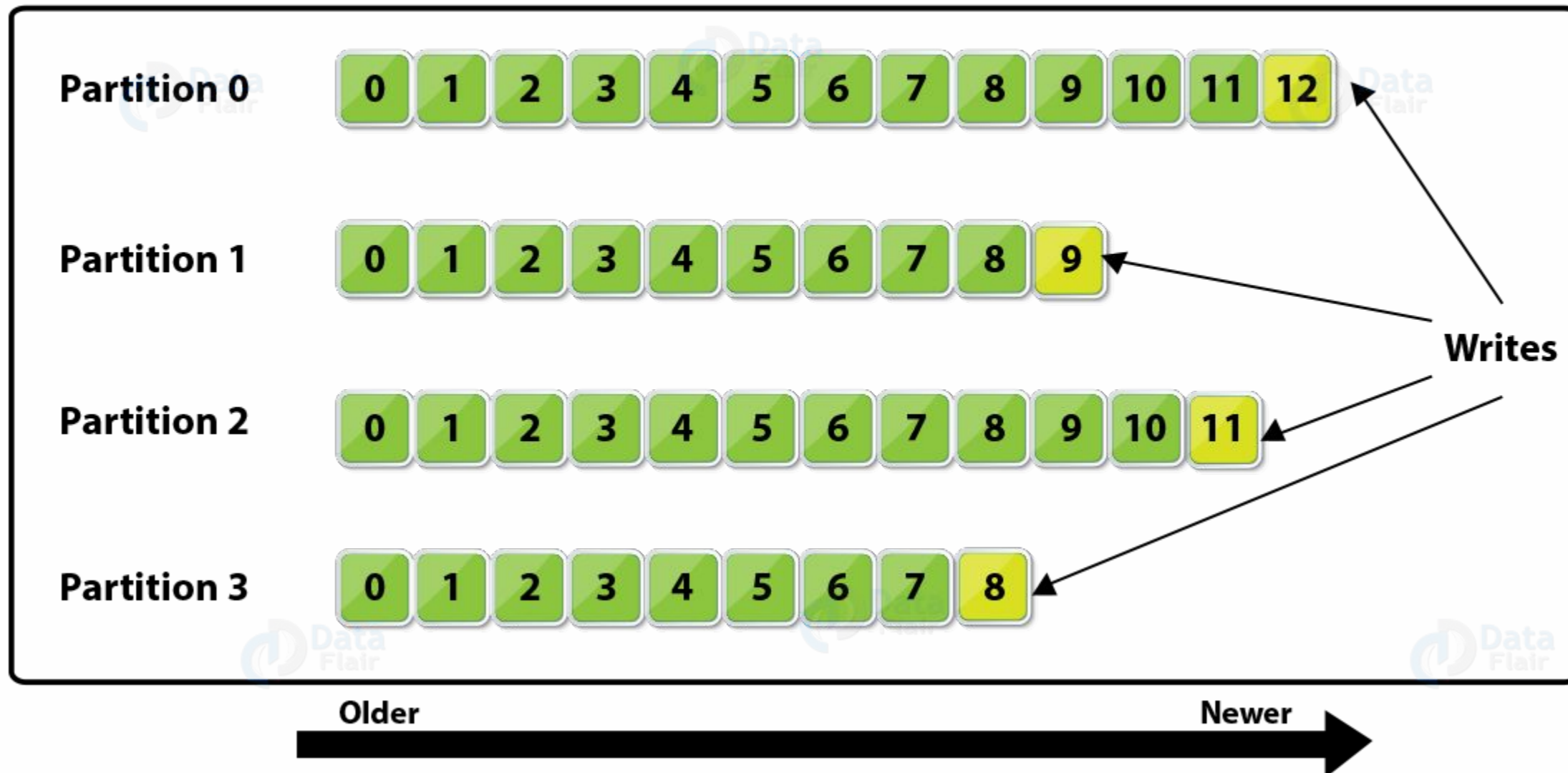


- Гарантия доставки
- Репликация журнала и синхронизация реплик
- Уведомление о смещении(offset) получателя
- Уведомление о получении сообщения для продюсера
- хранит свои записи на диске и ничего не держит в оперативной памяти
  - Операции считывания и записи выполняются за постоянное время  $O(1)$
- Последовательное чтение и запись на диск
- Использует кэш диска
- API для producers/consumers
- поддержка KSQL

- Consumer через pull запрашивает о появлении новых сообщений
- Журнал очищается на основе настроек времени и размера
- Есть возможность сжатия журнала
- Есть возможность сдвига смещения для переигрывания приема сообщений
- Для хранения метаданных используется zookeeper, в последних версиях отдельные топики kafka

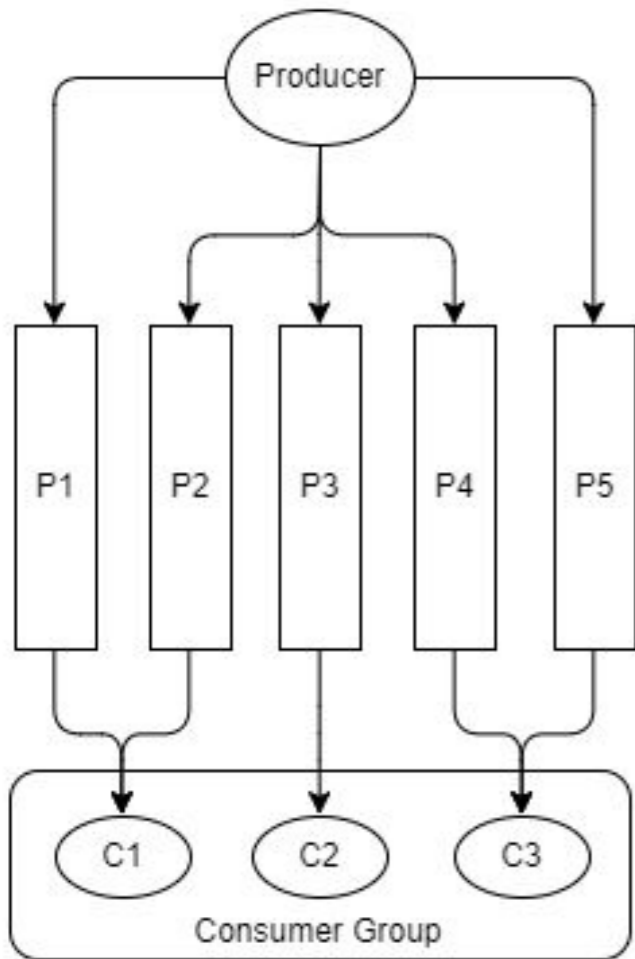
- **Record** – Запись, состоящая из ключа и значения
- **Topic** – категория или имя потока куда публикуются записи
- **Producer** – процесс публикующий данные в топик
- **Consumer** – процесс читающий данные из топика
- **Offset** – позиция записи
- **Partition** – шард топика

## Kafka Topic Partitions Layout

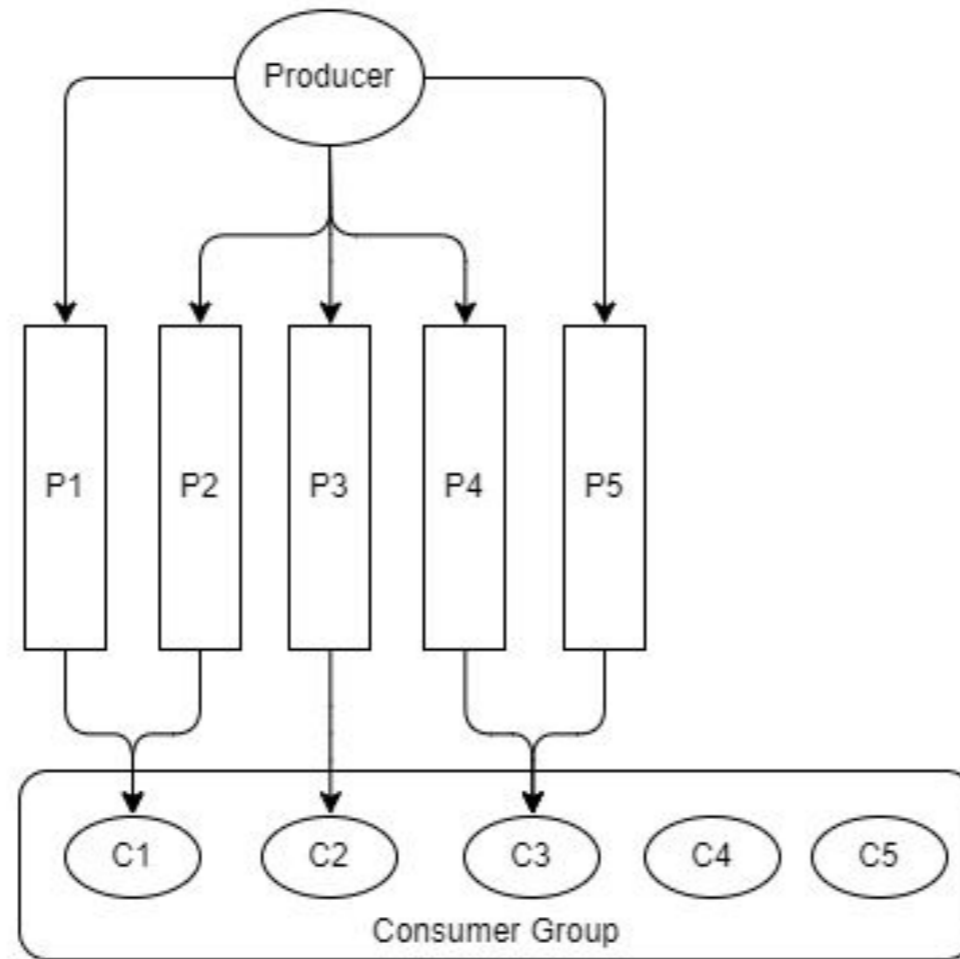


# Kafka: Consumers и партиции

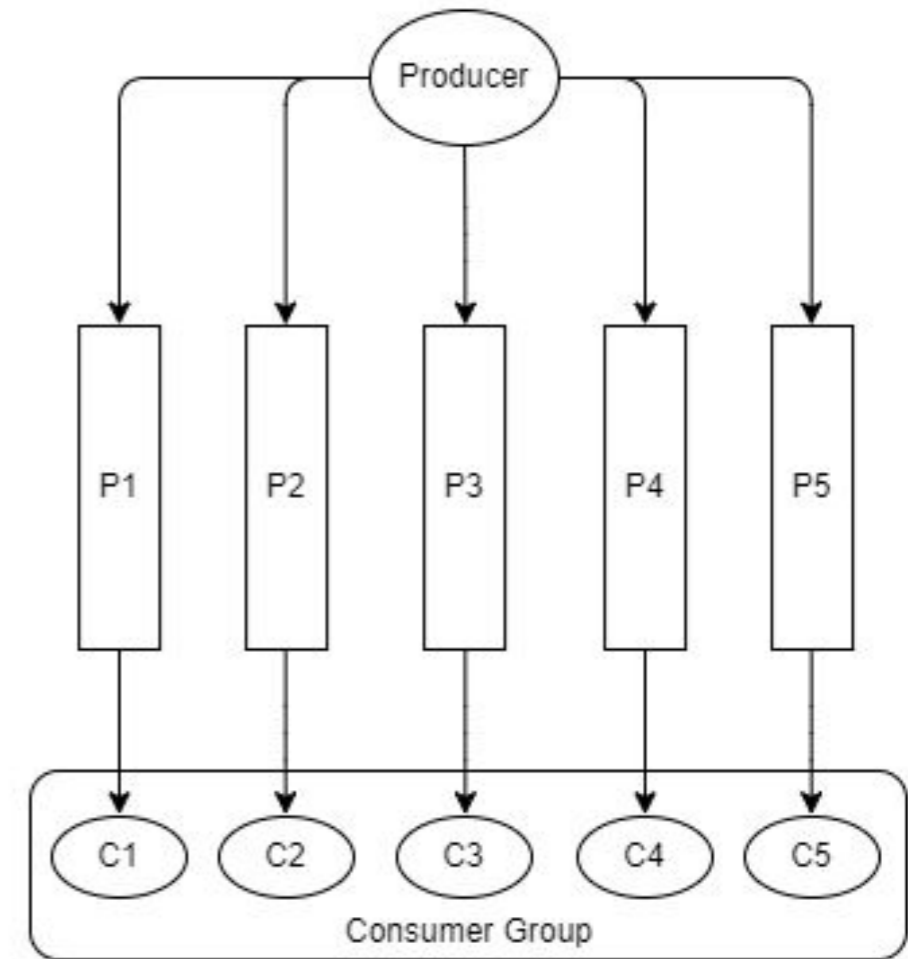
Original State



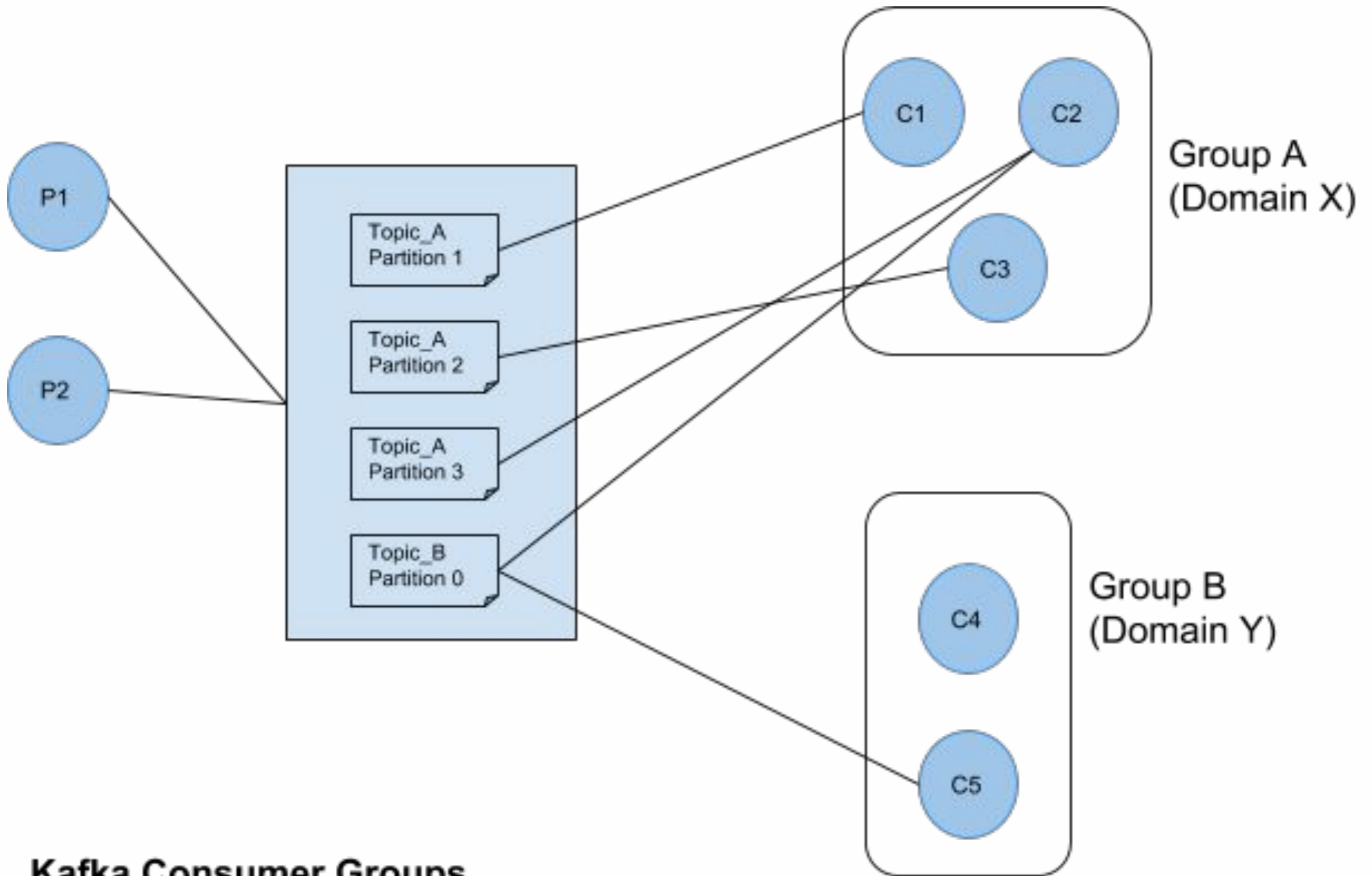
New Consumers, Unbalanced



After Rebalancing

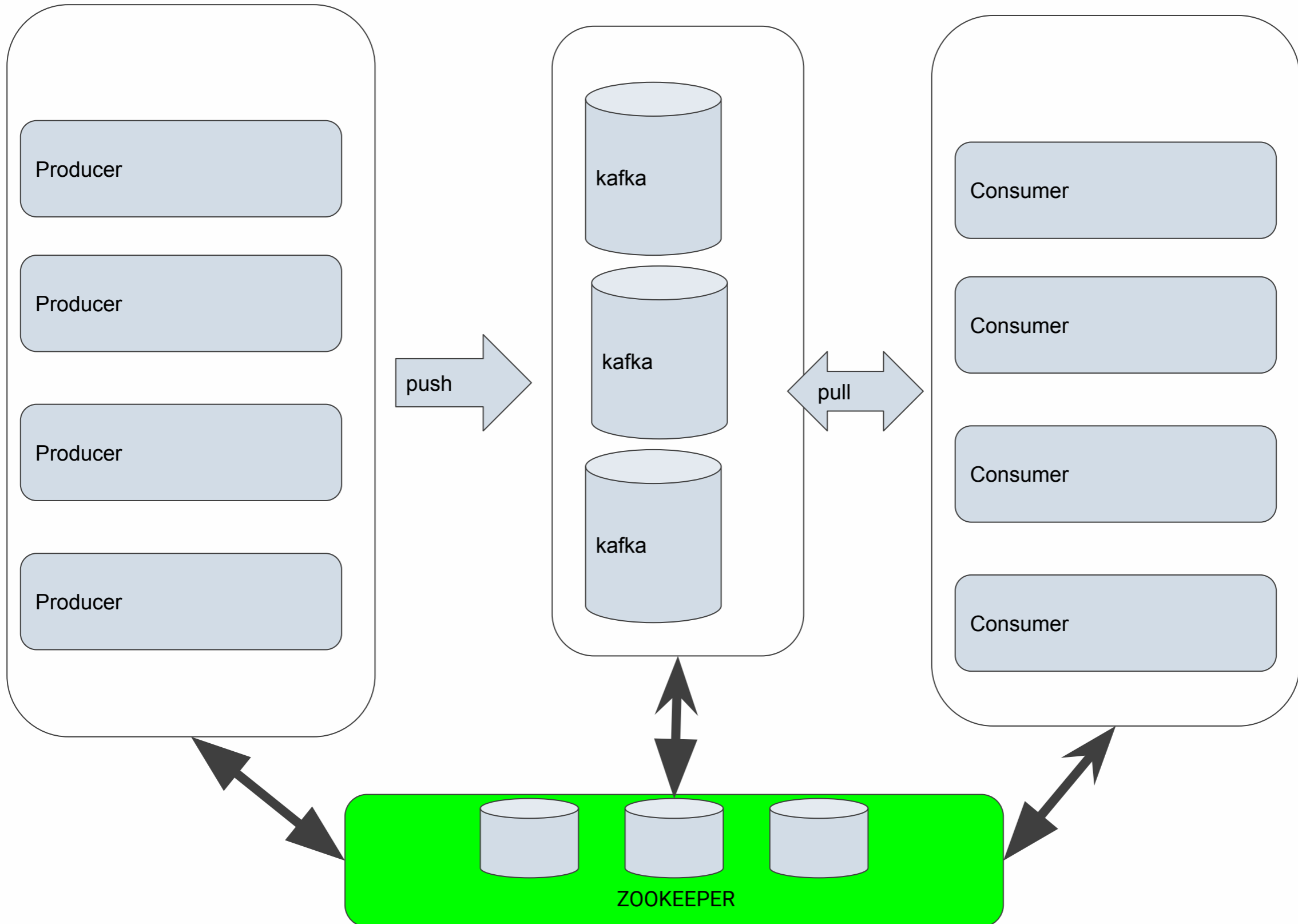


# Kafka: Consumer groups



**Kafka Consumer Groups**

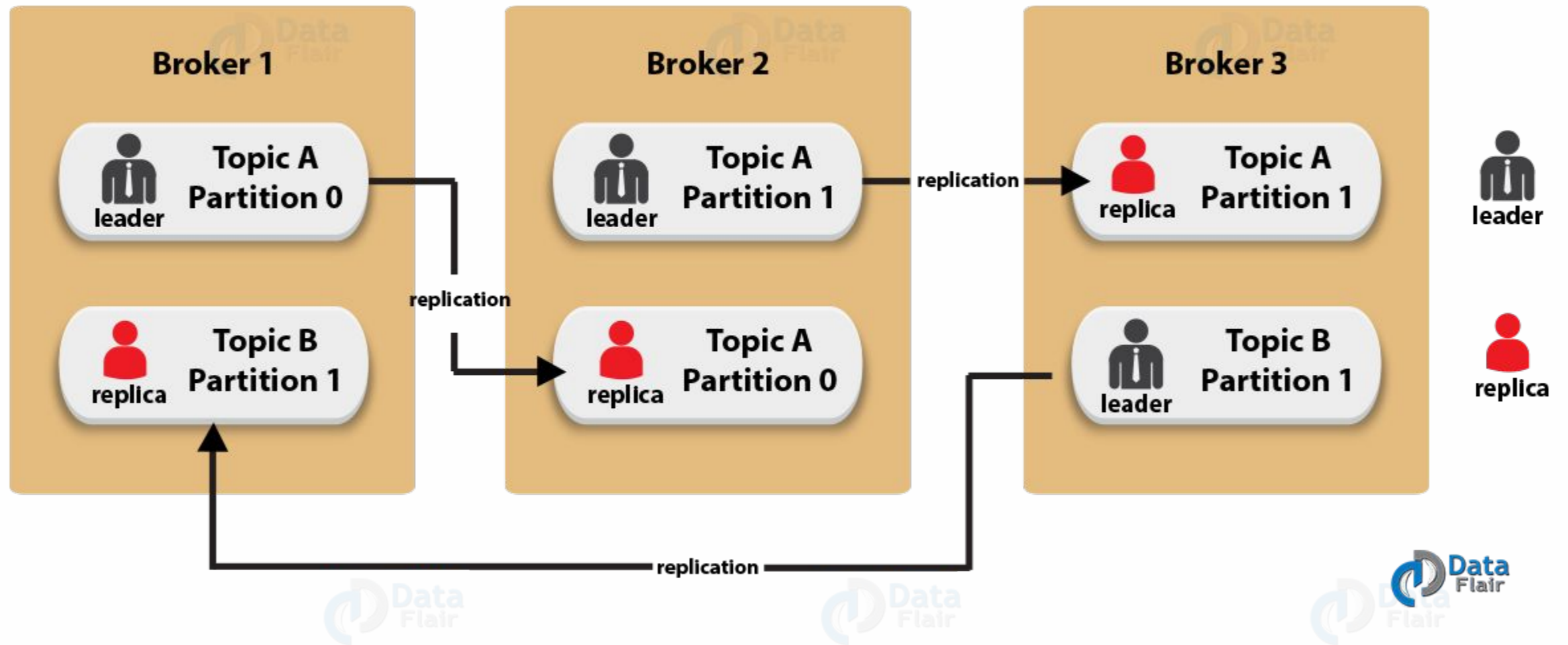
# Kafka: Cluster



Zookeeper – это распределенное хранилище ключей и значений. Оно сильно оптимизировано для считывания, но записи в нем происходят медленнее.

Применяется для хранения метаданных и обработки механизмов кластеризации (heartbeat, распределенные операции обновления, конфигурации, т.д.).

- Offset групп потребителей в рамках секции
- ACL – используются для ограничения доступа /авторизации
- Квоты – максимальные предельные количества сообщений в секунду
- Partition Leaders и уровень их работоспособности



# 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**

