

Подробнее про очереди.

Как вы поняли, очередь в RabbitMQ является последовательным временным хранилищем для необработанных сообщений. Что это значит?

- Последовательность реализуется с помощью паттерна "первый пришел - первый ушел"
- После обработки сообщение удаляется из очереди
- Если сервер упадет и мы не записывали сообщения, они будут потеряны из очереди

Хранение сообщений на диске это опциональная настройка данного брокера. Есть параметр очереди durable, который определяет, сохранится ли очередь после перезагрузки брокера. Важно - для сохранения должен быть включен durable, а также продюсер должен отправлять сообщения с указанием delivery_mode = 2. Если в сообщении будет указано delivery_mode = 1, значит запись на диск не требуется и настройка durable не будет иметь смысла.

Есть три типа очередей:

- Classic Queue (обычная очередь): это наиболее распространенный тип очереди в RabbitMQ. Сообщения сохраняются в обычной очереди и обрабатываются в порядке их поступления. Однако, они могут быть переданы только одному потребителю. В случае, если в очереди есть несколько потребителей, сообщение будет обработано только одним из них.
- Quorum Queue (очередь с обеспечением консистентности): это тип очереди, который предоставляет более надежное обеспечение консистентности (согласованности) данных. Он использует распределенный репликационный механизм для хранения сообщений с учетом возможных потерь данных. Это делает Quorum Queue более устойчивой к сбоям и повышает надежность.
- Stream Queue (очередь типа Stream): очереди типа Stream - это относительно новый тип очередей в RabbitMQ, который был разработан с целью предоставить более мощную обработку потоков данных, аналогичную системе Apache Kafka. Очереди Stream позволяют управлять большим объемом данных и обеспечивают возможность разделения их на разные "потоки". Это позволяет более эффективно обрабатывать высокие нагрузки и сохранять поток данных на более длительное время.

Каждый из этих типов очередей в RabbitMQ имеет свои преимущества и недостатки, и выбор зависит от потребностей вашей системы и специфических требований к обработке и хранению сообщений.

Сообщение в RabbitMQ.

Сообщение для отправки состоит из следующих полей:



- Payload - тело сообщения, сюда вы помещаете свою информацию.
- Routing Key - позволяет маршрутизировать сообщение согласно настройкам binding
- Delivery_mode - обсуждали ранее, ключ отвечает за персистентность
- Headers - заголовки, также позволяют маршрутизировать сообщение согласно настройкам binding. Можно не использовать, если роутинг происходит по Routing Key

Есть и дополнительные параметры, например, можно задать expiration - время жизни сообщения в очереди. Также обсуждали ранее параметр mandatory - отвечает за передачу в отдельную очередь или возврат сообщения, если не найдется настроек маршрутизации.

В самой очереди сообщение хранится с меткой timestamp (время добавления сообщения в очередь), id и другими системными параметрами.

Для передачи сообщений доступны разнообразные форматы данных, рассмотрим самые популярные:

- JSON

- Легко читается, много где используется. Но размер сообщения может быть в два раза больше чем Avro и в несколько раз больше чем Protobuf.
- Avro
 - Тяжело читается, но мало занимает места и быстро обрабатывается.
- Protobuf
 - Тяжело читается, но занимает еще меньше места чем Avro и еще быстрее обрабатывается.

Формат данных обычно определяет разработчик, но вы можете повлиять на это решение. Например, если вам не требуется повышенная производительность, можно убедить оставить формат передачи JSON.