

Введение в брокеры сообщений.

В предыдущих уроках мы рассмотрели различные стили интеграции, такие как webAPI (RPC), обмен файлами, общая база данных и коротко затронули обмен сообщениями. Сейчас мы подробно остановимся на стиле интеграции с использованием брокеров сообщений.

Вспомним теорию. Брокерированный обмен сообщениями, или обмен сообщениями через брокеры, - это стиль интеграции, при котором системы взаимодействуют друг с другом посредством асинхронного обмена сообщениями. Он отличается от прямого доступа к общим ресурсам, как в случае с общей базой данных или API.

Брокер сообщений выступает в роли посредника, обрабатывающего и направляющего сообщения между отправителями и получателями. Говоря простыми словами, брокер – это система, которая отвечает за логику отправки, получения и хранения сообщений.

Сначала давайте разберемся, зачем нам вообще нужны брокеры?

- Обмен данными между микросервисами: в мире микросервисной архитектуры, где каждый сервис выполняет свою уникальную функцию, обмен данными становится ключевой частью системы. Брокеры сообщений обеспечивают надежный и эффективный способ обмена информацией между микросервисами.
- Интеграция микросервисов: при использовании брокеров мы можем отправлять и читать нужные нам сообщения, поэтому брокеры сообщений используются для интеграции различных микросервисов, позволяя им работать вместе для достижения определенной цели.
- Реализация асинхронного взаимодействия: в некоторых случаях, особенно при обработке больших объемов данных или выполнении тяжелых операций, синхронное взаимодействие может привести к задержкам и снижению производительности. Брокеры сообщений позволяют реализовать асинхронное взаимодействие, обеспечивая параллелизм и повышение эффективности системы.
- Независимое взаимодействие микросервисов: когда взаимодействующие системы должны работать независимо, брокеры сообщений становятся отличным решением. Они обеспечивают возможность отправки сообщений без необходимости знания о состоянии другой системы (она может быть, например, временно недоступна). Это особенно полезно при масштабировании, когда количество взаимодействующих микросервисов может расти.
- Обработка потоков данных: некоторые системы, например системы анализа логов, часто требуют обработки больших объемов данных от разных источников. Брокеры сообщений обеспечивают эффективную и быструю обработку этих потоков, позволяя системам справляться с большими нагрузками.

- Координация распределенных транзакций: например, в финансовой системе, где одна транзакция может затрагивать несколько сервисов (списание с одного счета, зачисление на другой, регистрация операции в истории транзакций), брокер сообщений помогает координировать эти действия и обеспечивает атомарность транзакции (атомарность рассматривали ранее в девятой главе).
- Обработка и чтение данных в реальном времени: в области здравоохранения, финансов, или других областях, где требуется обработка и чтение больших объемов данных в режиме реального времени, брокер сообщений может использоваться для надежной и быстрой передачи данных между различными устройствами и системами.
- Интернет вещей (IoT): в приложениях IoT, где множество устройств постоянно генерируют данные, брокеры сообщений обеспечивают надежную и масштабируемую систему для сбора, хранения и обработки этих данных.

Из кейсов применений видно, что брокеры сообщений имеют ряд преимуществ. Давайте рассмотрим их подробно, но также затронем недостатки.

Преимущества интеграции через обмен сообщениями:

- Асинхронность: обмен сообщениями позволяет асинхронно обмениваться данными между системами, что уменьшает зависимость между компонентами и обеспечивает более высокую производительность.
- Декуплинг (разъединение): брокеры сообщений позволяют отправителю и получателю работать независимо друг от друга. Это означает, что отправитель не нуждается в информации о получателе, его состоянии или доступности.
- Масштабируемость: брокеры сообщений обеспечивают высокую масштабируемость, позволяя увеличивать объем обрабатываемых данных без значительных изменений в архитектуре системы. Можно добавлять новых получателей сообщений, отправителю сообщений код изменять не придется.
- Распределенная обработка: обмен сообщениями поддерживает распределенную обработку данных, что упрощает интеграцию между различными системами и уменьшает сложность инфраструктуры. Это механизм, позволяющий обрабатывать большое количество сообщений параллельно.

- Надежность и отказоустойчивость: брокеры сообщений могут предложить различные уровни гарантии доставки, которые обеспечивают надежность. Брокеры также обеспечивают персистентность, что означает, что сообщения сохраняются до тех пор, пока не будут успешно обработаны.
- Гибкость: брокеры сообщений могут поддерживать различные форматы и протоколы сообщений, что позволяет интегрировать системы, разработанные с использованием разных технологий и стандартов.
- Упорядочивание: брокеры сообщений могут обеспечивать упорядочивание сообщений, что может быть важно в некоторых системах.

Недостатки интеграции через обмен сообщениями:

- Сложность: обмен сообщениями может быть более сложным для разработки и поддержки по сравнению с другими стилями интеграции, такими как общая база данных или API, из-за асинхронности и большего количества компонентов.
- Задержки в обработке: использование брокера сообщений может внести задержки в обработку данных из-за асинхронного обмена сообщениями, что может быть критичным для некоторых приложений, требующих немедленного взаимодействия между системами. Могут, например, накопиться сообщения и ресурсов сервера не хватит для их своевременной обработки.
- Зависимость от одного брокера: если все обмены информацией в системе происходят через одного брокера, это создает точку отказа. Если брокер сталкивается с проблемами, это может повлиять на все системы, которые от него зависят.
- Сложность обеспечения согласованности: в асинхронной среде, какой является брокер сообщений, поддержание согласованности данных между сервисами может быть сложной задачей.
- Необходимость управления брокером: брокер сообщений требует дополнительных усилий для развертывания, настройки и мониторинга, что может увеличить затраты на инфраструктуру и поддержку.

Необходимо внимательно изучать вашу и сторонние системы, чтобы принять окончательное решение по использованию брокеров. Где-то они принесут пользу, а где-то могут требовать внимательной поддержки и служить слабым звеном в цепочке интеграций.

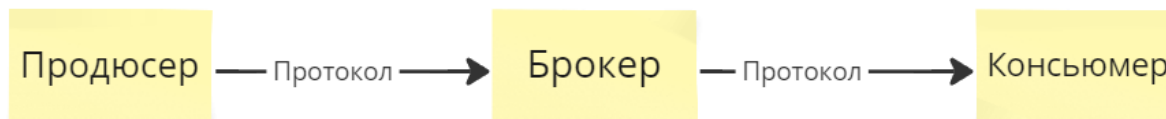
Давайте сразу коротко повторим, как реализуется обмен сообщениями:

1. Выбор брокера сообщений: вначале выбирается подходящий брокер сообщений, который будет использоваться для обмена данными между системами. Примеры популярных брокеров сообщений - RabbitMQ и Apache Kafka.
2. Настройка брокера сообщений: брокер должен быть развернут на сервере и настроен в соответствии с требованиями интеграции, включая настройку очередей, топиков или каналов для обмена сообщениями, а также механизмов обработки ошибок и маршрутизации сообщений.
3. Интеграция систем с брокером: для каждой системы, участвующей в интеграции, настраиваются механизмы отправки и получения сообщений через брокер. Это может включать создание необходимых компонентов на основе технологий и языков программирования, используемых в каждой системе.
4. Обмен сообщениями: в процессе работы системы отправляют сообщения в брокер, который затем обрабатывает и маршрутизирует их к соответствующим получателям. Получатели обрабатывают сообщения и выполняют соответствующие действия, такие как обновление данных или запуск процессов.

Основные термины, которые нужно запомнить:

- Брокер сообщений – уже разбирали ранее, это система, которая отвечает за логику отправки, получения и хранения сообщений.
- Поставщик или отправитель (Producer / «Продюсер») – сервис, отправляющий сообщения в брокер.
- Потребитель или получатель (Consumer / «Консьюмер») – сервис, получающий сообщения из брокера.

Схема взаимодействия выглядит таким образом:



Взаимодействие происходит по выбранному вами протоколу и в зависимости от используемого брокера. Разные брокеры могут поддерживать разные сетевые протоколы – например уже известный нам HTTP/HTTPS или другие, например:

- AMQP (Advanced Message Queuing Protocol): открытый стандартный протокол для обмена сообщениями между системами. RabbitMQ - один из примеров брокера сообщений, который поддерживает AMQP.
- MQTT (Message Queue Telemetry Transport): протокол обмена сообщениями, предназначенный для устройств с ограниченными ресурсами и низкой пропускной способностью, например, для устройств Умного дома, использующий Интернет вещей, IoT.
- STOMP (Simple Text Oriented Messaging Protocol): текстовый протокол обмена сообщениями, который может быть использован с любым совместимым брокером сообщений.
- Kafka Protocol: протокол, разработанный специально для Apache Kafka.

Брокеров существует достаточно много, все они отличаются сложностью реализации, поддержкой определенных протоколов, шаблонами взаимодействия. Полный список можно посмотреть тут - [ссылка на Wiki](#).

В этом блоке мы с вами рассмотрим основные модели работы брокеров, не привязываясь к конкретной реализации.