Дата последней сборки документа:
17 марта 2024 г. в 02:28

Создание микросервисов

Содержание

1. Основы

Микросервисы – это независимо выпускаемые сервисы, которые моделируются вокруг предметной области бизнеса. Сервис инкапсулирует функциональность и делает ее доступной для других сервисов через сети – вы создает более сложную, комплексную систему из этих строительных блоков. Один микросервис может представлять складские запасы, другой – управление заказами и еще один – доставку, но вместе они могут составлять целую систему онлайн-продаж.

Они представляют собой *тип* сервис-ориентированной архитектуры, в которой ключевым фактором выступает возможность независимого развертывания. Они не зависят от технологий, что является одним из преимуществ.

Снаружи отдельный микросервис рассматривается как черный ящик. Он размещает бизнес-функции в одной или нескольких конечных точках сети (например, в очереди или REST API) по любым наиболее подходящим протоколам.

Потребители, будь то другие микросервисы или иные виды программ, получают доступ к этой функциональности через такие точки. Внутренние детали реализации (например, технология, по которой был создан сервис, или способ хранения данных) полностью скрыты от внешнего мира.

Это означает, что в *микросервисных* архитектурах в большинстве случаев *не используются общие базы данных*. Вместо этого каждый микросервис инкапсулирует свою собственную БД там, где это необходимо.

Микросервисы используют концепцию *скрытия информации*. Это означает, что скрытие как можно большего количества информации внутри компонента и как можно меньшее ее раскрытие через внешние интерфейсы.

Реализацию, скрытую от сторонних участников процесса, можно свободно преобразовывать, пока у сетевых интерфейсов, предоставляемых микросервисом, сохраняется обратная совместимость.

Сервис-ориентированная архитектура (SOA, service-oriented architecture) – это подход к проектированию, при котором несколько сервисов взаимодействуют для обеспечения определенного

конечного набора возможностей (*сервис* здесь обычно означает полностью отдельный *процесс ОС*). Связь между этими сервисами осуществляется посредством сетевых вызовов, а не с помощью вызовов методов внутри границ процесса.

Вы должны воспринимать микросервисы как специфический подход к SOA.

1.1. Ключевые понятия микросервисов

Возможность *независимого развертывания* – это идея о том, что мы можем внести изменения в микросервис, развернуть его и предоставить это измененение нашим пользователям без необходимости развертывания каких-либо других микросервисов. Важно не только то, что мы можем это сделать, но и то, что *именно так* вы управляете развертываниеми в своей системе. Подходите к идее независимого развертывания как к чему-то *обязательному*.

Убедитесь, что вы придерживаетесь концепции независимого развертывания ваших микросервисов. Заведите привычку развертывать и выпускать изменения в одном микросервисе в готовом ПО без необходимости развертывания чего-либо еще. Это будет полезно.

Чтобы иметь возможность независимого развертывания, нам нужно убедиться, что наши микросервисы *слабо связаны*, то есть обеспечена возможность изменять один сервис без необходимости изменять что-либо еще. Это означает, что нужны явные, четко определенные и стабильные контракты между сервисами. Некоторые варианты реализации (например, совместное использование баз данных) затрудняют эту задачу.

Моделируя сервисы вокруг предметных областей бизнеса, можно упростить внедрение новых функций и процесс комбинирования микросервисов для предоставления новых функциональных возможностей нашим пользователям.

Развертывание функции, требующей внесения изменений более чем в один микросервис, обходится дорого. Вам придется координировать работу каждого сервиса (и, возможно, отдельных команд) и тщательно отслеживать порядок развертывания новых версий этих сервисов. Это потребует гораздо большего объема работ, чем внесение таких же преобразований внутри одного сервиса. Следовательно, нужно найти способы сделать межсервисные изменения как можно более редкими.

В случае микросервисов мы отдаем приоритет *сильной связности бизнес-функциональности*, а не технической функциональности.

Одна из самых непривычных рекомендаций при использовании *микросервисной* архитектуры состоит в том, что *необходимо избегать использования общих баз данных*. Если микросервис хочет получить доступ к данным, хранящимся в другом микросервисе, он должен напрямую запросить их у него [1, стр. 33].

Если мы хотим реализовать независимое развертывание, нужно убедиться, что есть ограничения на обратно несовместимые изменения в микросервисах. Если нарушить совместимость с вышестоящими потребителями, это неизбежно повлечет за собой необходимость внесения изменений и в них тоже.

1.2. Монолит

Когда все функциональные возможности в системе должны развертываться вместе, такую систему считают *монолитом*. Часто выделяют следующие архитектуры монолитов: однопроцессный, модульный и распределенный монолит.

1.2.1. Однопроцессный монолит

Наиболее распространенный пример, который приходит на ум при обсуждении монолитов, – система, в которой весь код развертывается как *единый процесс*. Может существовать несколько экземпляров этого процесса (из соображений надежности или масштабирования), но в реальности *весь код* упакован в один процесс.

1.2.2. Модульный монолит

Модульный монолит представляет собой систему, в которой один процесс состоит из отдельных модулей. С каждым модулем можно работать независимо, но *для развертывания* их все равно необходимо *объединить*.

Одна из проблем модульного монолита заключается в том, что *базе данных*, как правило, *не хватает декомпозиции*, которую мы находим на уровне кода, что приводит к значительным проблемам, если потребуется разобрать монолит в будущем.

1.2.3. Распределенный монолит

Распределенный монолит – это состоящая из нескольких сервисов система, которая должна быть развернута единовременно. Распределенный монолит вполне подходит под определение SOA, однако он не всегда соответствует требованиям SOA.

Микросервисная архитектура действительно предлагает более конкретные границы, по которым можно определить «зоны влияния» в системе, что дает гораздо больше гибкости.

1.2.4. Преимущества монолитов

Некоторые монолиты, такие как модульные и однопроцессные, обладают целым рядом преимуществ. Их гораздо более простая топология развертывания позволяет избежать многих ошибок, связанных с распределенными системами. Это может привести к значительному упрощению рабочих процессов, мониторинга, устранения неполадок и сквозного тестирования.

Для использования микросервисной архитектуры нужно найти убедительные причины.

1.3. Агрегирование логов и распределенная трассировка

Не используйте слишком много новых технологий в начале работы с микросервисами. Тем не менее *инструмент агрегации логов* настолько важен, что стоит рассматривать его как обязательное условие для внедрения микросервисов.

Такие системы позволяют собирать и объединять логи из всех ваших сервисов, предоставляя возможности для анализа и включения журналов в активный механизм оповещения.

Список литературы

1. *Ньюмен С.* Создание микросервисов. – СПб.: Питер, 2024. – 624 с.