MICROSERVICES

# MicroServies là gì?

Nền tẳng của kiến trúc Microservice (MSA) là về việc phát triển một ứng dụng bằng các service nhỏ và độc lập chạy trong tiến trình riêng của chúng. Các service này được phát triển và deploy một cách độc lập.

Ngược lại với kiểu kiến trúc Microservices là kiểu kiến trúc Monolith. Khi xây dựng theo kiểu kiến trúc này, toàn bộ các module (view, business, database, report) đều được gom chung vào một project lớn. Khi deploy, chúng ta sẽ quăng khối code này lên server và config để nó chạy.

Microservies thay vì gom tất cả module thành một khối (monolith), ta tách các module thành những service siêu nhỏ. Mỗi service sẽ được đặt trên một server riêng (Có thể dùng server cloud như AWS hoặc Azure), giao tiếp với nhau thông qua mạng (Gửi nhận message qua giao thức HTTP hoặc sử dụng MessageQueue)...

# Tại sao phải tách hệ thống thành MicroServices?

Tính chất khối Monolithic ảnh hưởng đến cấu trúc Microservices:

* Được thiết kế, phát triển và triển khai dựa theo một khối duy nhất.
* Các ứng dụng monolithic rất phức tạp và nó sẽ gây ra nhiều khó khăn cho quá trình nâng cấp, bảo trì hoặc thêm các tính năng mới.
* Thường rất khó để có thể áp dụng triển khai dựa theo kiểu agile.
* Bạn cần phải triển khai lại toàn bộ một hệ thống mặc dù chỉ phải cập nhật hoặc nâng cấp một phần duy nhất.
* Mở rộng được các khối ứng dụng nếu như gặp khó khăn thì sẽ có các yêu cầu về những tài nguyên khác nhau.
* Một service thường không có độ ổn định nên có thể làm sập cả hệ thống.
* Diagram

  Description automatically generatedKhó đổi mới: Bởi vì ứng dụng monolithic cần phải sử dụng chung một công nghệ vì vậy nó rất khó có thể thay đổi hoặc áp dụng thêm các công nghệ mới.
* Các tính chất giới hạn này của kiến trúc Monolithic dẫn đến sự phát triển của kiến trúc Microservices.

# Ưu điểm, nhược điểm khi sử dụng MicroServices

Ưu điểm nổi trội của Microservices:

* Microservices cho phép dễ dàng continuous delivery và deployment các ứng dụng lớn và phức tạp hơn.
* Có thể cải thiện khả năng bảo trì: bởi vì các service tương đối nhỏ nên dễ hiểu và dễ thay đổi hơn.
* Có khả năng testing dễ dàng: nhờ các services nhỏ
* Có thể triển khai tốt hơn: các services thường rất dễ cho việc triển khai độc lập.
* Cho phép các services được phát triển nhanh chóng bởi những team khác nhau. Khi đó, mỗi team đều sẽ được phát triển và thử nghiệm để triển khai cũng như mở rộng được quy mô của dịch vụ của mình một cách độc lập nhất với tất cả các team.
* Nếu như có lỗi xảy ra trong một service thì chỉ có service đó bị ảnh hưởng và các service khác sẽ thực hiện xử lý các yêu cầu cần thiết. Trong khi đó, thì mỗi một thành phần nếu như hoạt động sai của kiến trúc một khối thì nó sẽ làm ảnh hưởng đến toàn bộ hệ thống.
* Lập trình viên có thể thay đổi dễ dàng bằng cách sử dụng công nghệ mới khi triển khai các service. Tương tự như khi có thay đổi lớn thì các service đều có thể thực hiện và bạn dễ dàng thay đổi được công nghệ hơn.

Nhược điểm còn tồn tại:

* Bởi vì các nhà phát triển thường xuyên phải đối phó với sự phức tạp khi tạo ra một hệ thống phân tán.
* Cần phải implement việc communication giữa các inter-services
* Handle partial failure rất phức tạp bởi vì luồng xử lý cần phải đi qua nhiều service khác nhau.
* Khi thực hiện các requests trải rộng trên nhiều service khó khăn thì điều này cần đòi hỏi sự phối hợp giữa các team.
* Thường rất khó khăn trong việc đảm bảo toàn vẹn cho CSDL nếu như triển khai theo các cấu trúc cơ sở dữ liệu dạng phân vùng.
* Việc triển khai và quản lý microservices nếu như làm thủ công theo cách làm với ứng dụng thì sẽ rất phức tạp.
* Lập trình viên cần phải xử lý các sự cố kết nối chậm, lỗi nếu như thông điệp không được gửi hoặc nếu như thông điệp được gửi đến nhiều đích đến vào các thời điểm khác nhau.

# Thiết kế Microservices

Ví dụ ta có hệ thống bán lẻ bao gồm các dịch vụ:

A red and white sign

Description automatically generated with low confidence

A screenshot of a computer

Description automatically generated with low confidenceÁp dụng kiến trúc MSA, hệ thống bán lẻ có thể được mô tả lại như sau:

Điều quan trọng trong thiết kế MSA là phải quyết định đúng kích thước, phạm vi và khả năng của các microservice. Đây có lẽ là điều khó nhất và cần gặp phải khi thực hiện MSA trong thực tế.

Dưới đây là một vài mối quan tâm chính và quan niệm sai lầm về chúng:

* Số dòng code (line of code)/ kích thước team (team size) là số liệu tệ hại: Có một số tranh luận về việc quyết định kích thước của microservice dựa trên số dòng code của việc triển khai hoặc [team size](http://blog.idonethis.com/two-pizza-team/). Tuy vậy, đây được coi là những số liệu rất không thực tế và tệ hại.
* *Micro* là một thuật ngữ bị hiểu sai: Nhiều lập trình viên có xu hướng nghĩ rằng họ nên cố làm cho service nhỏ nhất có thể. Đây là một hiểu nhầm, không phải càng nhỏ thì càng tốt.
* Trong bối cảnh web service, các service thường được triển khai ở các mức độ chi tiết khác nhau - từ một vài chức năng đến vài chục chức năng. Việc có các webservice và đặt chúng dưới dạng microservice sẽ không đem lại cho bất kỳ lợi ích nào của MSA.

Vậy làm thế nào chúng ta có thể thiết kế service đúng cách trong MSA:

* Một Service có phạm vi và chức năng giới hạn thì việc tập trung vào một nhiệm vụ sẽ giúp cho quá trình phát triển cũng như triển khai dịch vụ trở nên nhanh chóng hơn.
* Khi thiết kế, bạn nên xác định và giới hạn cho các service dựa theo chức năng nghiệp vụ thực tế.
* Hãy đảm bảo microservices có thể phát triển cũng như được triển khai độc lập.
* Mục tiêu thiết kế là đưa ra phạm vi cho một microservices phục cụ cho một nghiệp vụ chứ không đơn giản là làm những dịch vụ nhỏ hơn. Khi đó, kích thước hợp lý của một services đó chính là kích thước đủ để đáp ứng cho các yêu cầu của một chức năng bên trong hệ thống.
* Khác biệt so với services trong SOA thì một microservice không nên có quá nhiều hàm hoặc chức năng hỗ trợ xung quanh cũng như định dạng thông báo gửi hoặc gửi tin đơn giản.

# Các microservices trao đổi với nhau như thế nào?

Trong kiến trúc Monolith, các chức năng nghiệp vụ của các component/ bộ xử lý khác nhau sẽ gọi nhau bằng cách sử dụng các lệnh hàm gọi hoặc các method gọi hàm của ngôn ngữ lập trình. Trong MSA, điều này đã được chuyển sang cách request/response các message service có tính lỏng lẻo hơn, chủ yếu dựa trên SOAP (Simple Object Access Protocol - Giao thức truy cập đối tượng đơn giản) với nhiều giao thức khác nhau như HTTP, XML.

1. Tin nhắn đồng bộ (Synchronous Messaging) – REST, Thift

Đối với tin nhắn đồng bộ ( client yêu cầu phản hồi kịp thời từ các service và thời gian đợi để nhận được nó) trong MSA, REST là lựa chọn dễ nhất vì nó cung cấp một kiểu message đơn giản với các request, respose HTTP. Do đó hầu hết các microservice đều sử dụng HTTP bên cạnh các tài nguyên khác (mỗi chức năng đại diện cho một tài nguyên và các hoạt động sẽ được thực hiện trên các tài nguyên khác).

Timeline

Description automatically generated

1. Tin nhắn không đồng bộ (Asynchronous) - AMQP, STOMP, MQTT

Đối với một số hoàn cảnh chúng ta bắt buộc phải sử dụng các tin nhắn không đồng bộ ( client không muốn phản hồi ngay lập tức hoặc muốn hoàn toàn không phản hồi). Trong những tình huống này, các giao thức tin nhắn không đồng bộ như AMQP, STOPM hay MQTT được sử dụng phổ biến.(như RabbitMQ, Radis,…)

# Tìm vị trí các microservices

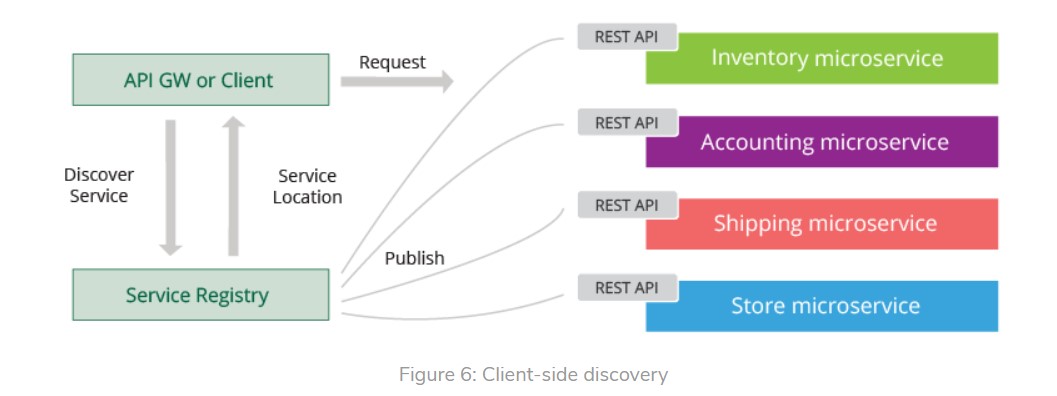
Trong MSA, số lượng các service mà bạn cần để làm việc sẽ là khá nhiều. Chúng thay đổi vị trí linh hoạt do tính chất cần phát triển nhanh của microservice. Do đó, bạn cần tìm vị trí của microservice trong suốt thời gian runtime. Giải pháp cho vấn đề này là sử dụng Service Registry.

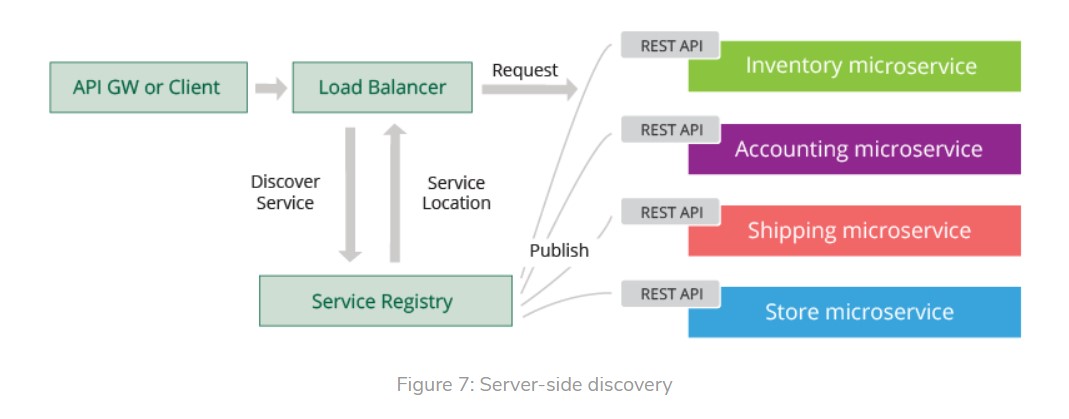
**Service Registry**

Service registry là nơi để chứa các metadata của các microservice instances (bao gồm vị trí location, host port,…). Các microservice instance được đăng ký với service registry khi khởi động và sẽ hủy đăng ký khi bị shut down. Các thành phần khác cần tìm thông tin của một microservice nào đó thì sẽ tìm thông qua service registry.

**Service Discovery**

Để tim các microservice đang hoạt động và vị trí location của nó, chúng ta cần đến cơ chế service discovery. Có 2 loại cơ chế service discovery là client-side discovery và server-side discovery. Chúng ta sẽ xem xét kỹ hơn về các cơ chế này:

Client-side discovery Theo cách tiếp cận này, client hoặc API-gateway sẽ có được vị trí của một service instance bằng cách truy vấn một service registry.

Server-side discovery Với phương pháp này, client/API gateway sẽ gửi một request đến một component (ví dụ như một load balancer) chạy trên một location đã biết. Component đó sẽ gọi đến service registry và xác địh vị trí location mà request cần đến.Các microservice có thể tận dụng các giải pháp deployment như Kubernetes cho service-side discovery.

# Deployment

Khi nói đến MSA, việc deploy các service đóng một vai trò quan trọng và có các yêu cầu chính như sau:

* Khả năng deploy/undeploy độc lập đối với các service khác.
* Phải có khả năng scale cho từng service.
* Deploy nhanh chóng.
* Lỗi trong một service sẽ không được làm ảnh hưởng đến bất kỳ service nào khác. [Docker](https://www.docker.com/) (một công cụ mã nguồn mở cho phép các lập trình viên và quản trị viên hệ thống deploy các container trong mỗi trường Linux) cung cấp một cấp tuyệt vời để deploy các service theo các yêu cầu trên. Các ý chính bao gồm:
* Package của service dưới dạng một Docker image.
* Deploy mỗi service instance bằng một container.
* Scale được thực hiện dựa trên việc thay đổi số lượng container instance.
* Build, deploy và khởi động một service sẽ nhanh hơn nhiều khi sử dụng Docker container (so với các máy ảo VM thông thường).

Graphical user interface, text, application

Description automatically generated

# Tổng kết

Khi bạn muốn kết hợp MSA trong môi trường CNTT doanh nghiệp hiện đại ngày nay, các khía cạnh chính bạn cần quan tâm tổng kết lại sau:

* Microservice không phải là thuốc chữa bách bệnh - nó không giải quyết tất cả các nhu cầu CNTT của doanh nghiệp của bạn, vì vậy chúng ta cần sử dụng nó với các kiến ​​trúc hiện có khác.
* Dùng SOA hợp lý.
* Hầu hết các doanh nghiệp sẽ không thể chuyển đổi toàn bộ hệ thống CNTT doanh nghiệp của họ sang microservice. Thay vào đó, họ sẽ sử dụng microservice để giải quyết một số trường hợp sử dụng kinh doanh trong đó họ có thể tận dụng sức mạnh của MSA.
* Tích hợp doanh nghiệp sẽ không bao giờ biến mất - điều đó có nghĩa là bạn cần phải có phần mềm tích hợp, chẳng hạn như ESB, để phục vụ cho tất cả các nhu cầu tích hợp doanh nghiệp của bạn.
* Tất cả business functions phải được expose dưới dạng API bằng cách tận dụng các kỹ thuật quản lý API.
* Tương tác giữa các microservices nên được thực hiện qua cổng gateway.
* Việc phối hợp service giữa các microservice có thể được yêu cầu cho một số business và có thể được triển khai bên trong một microservice khác hoặc ở tại lớp gateway.