## 

## Monolith Application là gì? (ứng dụng nguyên)

Bạn đã từng làm việc trong một project:

* Release (trên production) vài tháng một lần
* Có một loạt (rất nhiều) các tính năng và chức năng
* Có team nhiều hơn 50 members
* Debug là một thử thách lớn
* Áp dụng công nghệ mới và quy trình mới là gần như không thể

Đặc trưng của Monolith Application:

* Kích thước ứng dụng lớn
* Thời gian release dài
* Các team lớn (nhiều member)

Các thử thách đi kèm:

* Khả năng mở rộng
* Tương thích với công nghệ mới
* Áp dụng quy trình phát triển mới
* Áp dụng Automation test
* Thích nghi với các bài học phát triển hiện đại
* Tương thích với việc mở rộng thiết bị

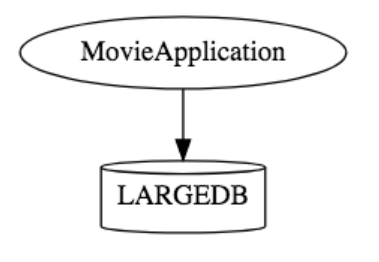
## Microservices là gì?

Microservice architecture đã phát triển như một giải pháp mở rộng và đổi mới đối với Monolith architecture.

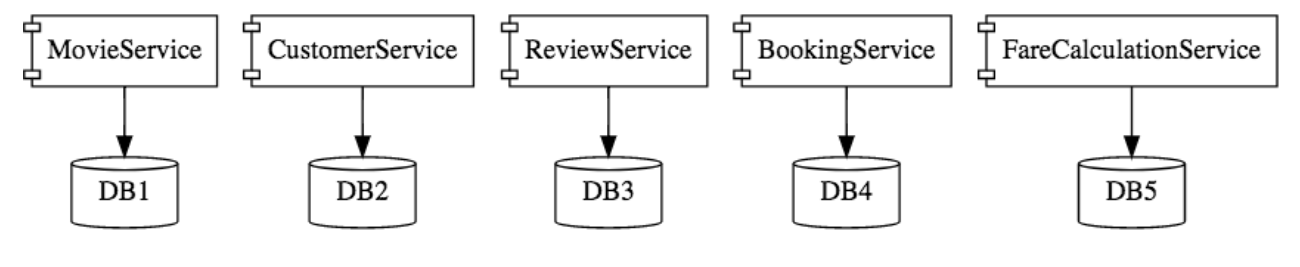
* REST – Xây dựng xung quanh tài nguyên RESTful. Sự giao tiếp có thể là HTTP hoặc event based.
* Các đơn vị có thể lựa chọn triển khai khá nhỏ – giới hạn về phạm vi của đơn vị
* Cloud Enabled – Dynamic Scaling

## Microservice Architecture nhìn như thế nào?

Sơ đồ dưới đây là một ví dụ về Monolith Application, một application cho tất cả mọi thứ.



Và sơ đồ dưới đây mô tả mô hình về một ứng dụng tương tự ví dụ trên sử dụng Microservice Architecture:



**Ưu điểm của Microservice**

* Việc thích ứng với công nghệ và quy trình mới trở nên dễ dàng hơn. Chúng ta có thể thử áp dụng những công nghệ mới với những service mới được tạo ra.
* Thời gian release được rút ngắn lại
* Có thể mở rộng với Cloud

## Những thử thách với Microservice Architecture

Trong khi việc phát triển một vài thành phần nhỏ trở nên dễ dàng hơn, bên cạnh đó có một số thử thách phức tạp liên quan đến Microservice đang tồn tại:

* Cần phải thiết lập nhanh: Chúng ta không thể tốn một tháng để thiết lập cho mỗi microservice. Chúng ta cần phải tạo microservice thật nhanh.
* Tự động hoá: Bằng việc thay thế Monolith Application bằng một số các thành phần nhỏ (microservice) thì chúng ta cần phải tự động hoá tất cả mọi thứ: builds, deploys, monitorings, …
* Tầm nhìn: Chúng ta sẽ có một số các thành phần nhỏ, thậm chí lên tới 100 hay 1000 thành phần. Vì vậy, ta cần phải kiểm soát và xác định vấn đề của chúng một cách tự động. Chúng ta nên hiển thị bao quát tất cả các thành phần để có một tầm nhìn tốt.
* Giới hạn phạm vi: giới hạn phạm vi của microservice không phải là việc dễ. Khởi đầu với **Domain Driven Design** là một sự lựa chọn tốt. Chúng ta phải hiểu về giới hạn phát triển của microservice trong một khoảng thời gian và phải đảm bảo rằng giới hạn phát triển của microservice được đưa ra.
* Quản lý cấu hình: Chúng ta cần maintain cấu hình của nhiều thành phần trong các môi trường khác nhau, vì vậy chúng ta sẽ cần một giải pháp cấu hình để xử lý vấn đề này.
* Mở rộng hoặc thu hẹp quy mô linh động (dynamic scale up and down): Lợi thế của microservice chỉ đem lại giá trị nếu ứng dụng của chúng ta có thể mở rộng hoặc thu hẹp quy mô một cách linh động trong Cloud.
* Khả năng chịu lỗi: nếu một microservice ở cuối cùng của chuỗi cuộc gọi thất bại, nó có thể ảnh hưởng đến tất cả những microservice khác. Microservice cần có khả năng chịu lỗi khi thiết kế.
* Khả năng debug: Khi xảy ra một sự cố cần điều tra, chúng ta có thể cần xem xét trong nhiều service khác nhau của các thành phần khác nhau. Tập trung Log và Dashboard là điều cần thiết để việc debug dễ dàng hơn.
* Tính nhất quán: Chúng ta không có một loạt các tool để giải quyết vấn đề này, thúc đẩy phát triển mới là điều quan trọng. Chúng ta nên có các hệ quản trị khác nhau để quản trị xung quanh các ngôn ngữ, nền tảng, công nghệ và các công cụ được sử dụng cho việc phát triển/triển khai/kiểm soát các microservice để đảm bảo tính nhất quán.

#### Những điều quan trọng từ Spring Cloud Modules

Dynamic Scale Up and Down, sử dụng tập hợp:

* Naming Server (Eureka)
* Ribbon (Client Side Load Balancing)
* Feign (Easier REST Clients)

Hiển thị và Kiểm soát với:

* Zipkin Distributed Tracing
* Netflix API Gateway

Quản lý cấu hình với:

* Spring Cloud Config Server

Khả năng chịu tải với:

* Hystrix