**Microservices Aracitecture Principles**

5 Nguyên lý cơ bản

1. **1 Microservice chỉ có 1 Single concern (chỉ xử lý đúng 1 nghiệp vụ)**

+ Giống như Single object responsibility (Chỉ làm 1 và chỉ 1).

Ex: Nếu nó là phần customer thì chỉ làm phần constomer – không xử lý phần đơn hàng hay thương mại, bán hàng,…

* Dễ dàng cải thiện code, duy trì, tăng, giảm độ lớn service, dự án,…

1. **1 Microservice phải có tính riêng rẽ**

+ Bản thân nó phải clear tất cả các boundaries separating, tashc biệt ra khỏi môi trường (environment)

+ Nó phải đóng giói 1 cách cẩn thận, đẩy đủ và tốt => Tất cả các login cà data cho microservice phải được đóng giói trong 1 single displacement point => Ex: Khi build 1 dự án sẽ ra 1 file đóng gói nó

+ Development: Nó phải được cách li bởi các microservices khác

+ Production: Các phần microservice sẽ được gắn kết lại với nhau => Trở thành 1 khối duy nhất

1. **1 Microservice có thể được di chuyển, vận chuyển được**

+ Có thể được dịch chuyển, di chuyển từ môi trường này sang 1 môi trường khác mà không bị mất quá nhiều công sức và thời gian

+ Dễ dàng trong quá trình automated test hoặc declarative deployment

Ex: 1 cái customer sau khi triển khai xong sẽ có thể đóng gói toàn bộ lại và team khác hoặc ứng dụng khác cần sử dụng service đó thì có thể pull image đó về và chạy ngay (khái niệm docker container image)

1. **1 Microservice chỉ nên chứa dữ liệu của chính nó**

+ Không chứa dữ liệu của khác service khác và các data của nó cũng phải được cách li khỏi data của các microservice khác

+ Nếu phải shared dữ liệu thì phải shared thông qua việc public interface

+ Vấn đề dư thừa dữ liệu

Ex: 1 service A khi cần 1 data (object, array,…) sẽ consumer tới service B.. Lúc này service B trả 1 object gồm nhiều field (thuộc tính) dữ liệu, nhưng service A chỉ cần mỗi filed Id => Dẫn tới dữ thừa dữ liệu (vấn đề không quá lớn) => Có thể define các Dto, Model trả ra đúng dữ liệu cần sử dụng

* Điều này có thể thường xuyên xảy ra, vi phạm nguyên lý cơ bản của 1 số thiết kế

1. **1 Microservice là 1 ephemeral**

+ 1 service có thể được tạo ra (created), hủy đi (destroyted), tái tạo lại (replenished) tùy theo yêu cầu và không ảnh hưởng tới toàn bộ hệ thống

Ex: Khi 1 nhu cầu vào 1 service tăng cao có thể tạo ra thêm service, khi không cần thiết có thể tự động hủy đi





