1. Giới Thiệu Về Khóa Học

Khóa học giúp nâng cao kỹ năng xây dựng và triển khai các hệ thống microservices theo tiêu chuẩn sản phẩm. Khóa học trang bị kiến thức về Spring Boot, Docker, Kubernetes và các công nghệ hỗ trợ khác.

2. Tầm Quan Trọng Của Microservices

Hiện nay, nhiều doanh nghiệp đang chuyển đổi từ các ứng dụng monolithic sang kiến trúc microservices nhằm tối ưu hóa hiệu suất, tăng tốc độ phát triển và dễ dàng quản lý. Kiến trúc microservices giúp các ứng dụng linh hoạt hơn, tăng tốc triển khai tính năng mới và hạn chế ảnh hưởng khi có lỗi.

3. Nội Dung Khóa Học

- Giới thiệu về microservices: So sánh với kiến trúc monolithic và server-based.
- Xây dựng microservices bằng Spring Boot:
    - Các tiêu chuẩn khi xây dựng REST API.
    - Các kỹ thuật xây dựng API bền vững.
    - Kiểm tra, xác thực và tài liệu hóa API.
- Container hoá với Docker: Triển khai microservices dễ dàng và hiệu quả hơn.
- Quản lý cấu hình với Spring Cloud Config.
- Khám phá Service Discovery và Load Balancing: Sử dụng Eureka Server và Spring Cloud Gateway.
- Tăng tính bền vững: Sử dụng Resilience4j để đảm bảo hệ thống hoạt động ổn định.
- Quan sát và giám sát:
    - Triển khai Grafana, Prometheus.
    - Phân tích logs và giám sát hệ thống.

- Bảo mật Microservices: Tích hợp Oauth2, OpenID và Spring Security Frameworks.
- Xử lý bất đồng bố:
    - Sử dụng RabbitMQ, Spring Cloud Functions.
    - Khám phá Kafka cho xử lý sự kiện.
- Triển khai Kubernetes:
    - Tổ chức Kubernetes Cluster.
    - Sử dụng Helm Package Manager.
    - Deploy microservices trên môi trường cloud.

Giới thiệu về kiến trúc Microservices

Microservices là một trong những phong cách kiến trúc được sử dụng để xây dựng các ứng dụng web hiện đại. Thay vì triển khai toàn bộ hệ thống như một khối duy nhất, kiến trúc microservices chia ứng dụng thành các dịch vụ nhỏ, độc lập và có thể hoạt động riêng lẻ.

Kiến trúc Monolithic

Trước khi tìm hiểu về microservices, cần hiểu về kiến trúc monolithic, một mô hình phổ biến được sử dụng trước đây trong ngành công nghệ phần mềm. Monolithic là cách tiếp cận truyền thống, trong đó tất cả các chức năng của ứng dụng được triển khai như một khối duy nhất.

Cấu trúc của Monolithic

Ví dụ, trong một ứng dụng ngân hàng như EasyBank, hệ thống cung cấp các sản phẩm như tài khoản (Accounts), thẻ (Cards) và khoản vay (Loans). Trong kiến trúc monolithic, tất cả các thành phần này được tích hợp vào một hệ thống duy nhất với các lớp:

- Lớp giao diện người dùng (Presentation Layer): Chứa HTML, CSS, JavaScript.

- Lớp logic nghiệp vụ (Business Logic Layer): Xử lý các quy tắc kinh doanh.
- Lớp truy cập dữ liệu (Data Access Layer): Tương tác với cơ sở dữ liệu chung.

Toàn bộ ứng dụng chạy trên một máy chủ và sử dụng một cơ sở dữ liệu duy nhất để lưu trữ thông tin về tài khoản, thẻ và khoản vay.

Ưu điểm của Monolithic

- Dễ phát triển và triển khai: Với các ứng dụng nhỏ hoặc nhóm phát triển nhỏ, việc triển khai đơn giản vì toàn bộ hệ thống được triển khai cùng một lúc.
- Ít yêu cầu về quản lý các vấn đề cắt ngang (Cross-Cutting Concerns): Các yêu cầu như bảo mật, ghi log, kiểm toán dễ quản lý hơn do tất cả đều nằm trong cùng một hệ thống.
- Hiệu suất cao: Vì tất cả các thành phần đều trong một máy chủ, giao tiếp giữa các bộ phận chỉ là lời gọi phương thức nội bộ mà không có độ trễ mạng.

Nhược điểm của Monolithic

- Khó mở rộng: Khi ứng dụng phát triển, việc bảo trì trở nên phức tạp do mã nguồn ngày càng lớn.
- Khó áp dụng công nghệ mới: Mọi thay đổi đều ảnh hưởng đến toàn bộ hệ thống, khiến việc nâng cấp công nghệ trở nên khó khăn.
- Thiếu khả năng chịu lỗi: Một lỗi nhỏ có thể khiến toàn bộ hệ thống bị sập.
- Khó triển khai cập nhật nhỏ: Một thay đổi nhỏ cũng yêu cầu triển khai lại toàn bộ hệ thống, gây gián đoạn dịch vụ.

Kiến trúc Service-Oriented Architecture (SOA)

Do những hạn chế của monolithic, ngành công nghiệp đã chuyển sang kiến trúc hướng dịch vụ (SOA).

Đặc điểm của SOA

Trong SOA, ứng dụng được chia thành các dịch vụ khác nhau, có thể giao tiếp với nhau thông qua một thành phần trung gian gọi là Enterprise Service Bus (ESB).

- Giao diện người dùng (UI Layer) được triển khai riêng biệt.
- Các dịch vụ nghiệp vụ (Business Services) như tài khoản, thẻ và khoản vay được triển khai tách biệt.
- ESB quản lý giao tiếp giữa UI và các dịch vụ backend.

Ưu điểm của SOA

- Tái sử dụng dịch vụ: Các thành phần có thể được tái sử dụng trong nhiều ứng dụng khác nhau.
- Dễ bảo trì: Mỗi dịch vụ có thể được nâng cấp mà không ảnh hưởng đến toàn bộ hệ thống.
- Phát triển song song: Các nhóm khác nhau có thể phát triển các dịch vụ một cách độc lập.

Nhược điểm của SOA

- Giao tiếp phức tạp: Các dịch vụ sử dụng giao thức SOAP/XML, khiến việc truyền tải dữ liệu trở nên nặng nề.
- Chi phí cao: ESB là một thành phần thương mại, thường yêu cầu đầu tư lớn.

Kiến trúc Microservices

Microservices là một bước tiến xa hơn của SOA, trong đó ứng dụng được chia thành nhiều dịch vụ nhỏ hơn, mỗi dịch vụ đảm nhiệm một chức năng cụ thể và có thể hoạt động độc lập.

Đặc điểm của Microservices

- Tách biệt các thành phần: Mỗi dịch vụ đảm nhiệm một phần nghiệp vụ riêng như tài khoản, thẻ, khoản vay và có thể hoạt động độc lập.

- Mỗi dịch vụ có cơ sở dữ liệu riêng: Không còn chia sẻ chung một database như monolithic hay SOA.
- Giao tiếp giữa các dịch vụ qua API: Thường sử dụng giao thức REST hoặc gRPC thay vì ESB.
- Triển khai độc lập: Mỗi microservice có thể được triển khai riêng trên container hoặc server riêng biệt.

Ưu điểm của Microservices

- Dễ dàng mở rộng: Có thể mở rộng từng dịch vụ mà không ảnh hưởng đến toàn bộ hệ thống.
- Linh hoạt trong công nghệ: Các nhóm phát triển có thể sử dụng công nghệ phù hợp với từng microservice, chẳng hạn như Python cho tài khoản, Java cho khoản vay.
- Triển khai nhanh chóng: Mỗi dịch vụ có thể được cập nhật độc lập, giúp giảm thời gian gián đoạn.
- Khả năng chịu lỗi cao: Nếu một dịch vụ bị lỗi, các dịch vụ khác vẫn hoạt động bình thường.

Nhược điểm của Microservices

- Độ phức tạp cao: Quản lý nhiều dịch vụ độc lập đòi hỏi hệ thống giám sát và điều phối mạnh mẽ.
- Chi phí cơ sở hạ tầng cao: Cần nhiều máy chủ hoặc container hơn để chạy các dịch vụ riêng lẻ.
- Vấn đề bảo mật: Dữ liệu truyền qua mạng giữa các dịch vụ có thể gặp rủi ro về bảo mật.

So sánh các mô hình kiến trúc: Monolithic, SOA và Microservices

Trong bài viết này, chúng ta sẽ so sánh ba mô hình kiến trúc chủ yếu: Monolithic, SOA (Service-Oriented Architecture) và Microservices.

1. Kiến trúc Monolithic

Mô hình Monolithic là cách tiếp cận truyền thống, trong đó toàn bộ mã nguồn của ứng dụng được triển khai trên một máy chủ duy nhất và sử dụng một cơ sở dữ liệu duy nhất. Tất cả các thành phần của ứng dụng, bao gồm giao diện người dùng và logic xử lý, đều được gói gọn trong một khối duy nhất.

- Ưu điểm: Dễ triển khai, bảo trì và quản lý vì không có nhiều thành phần phức tạp.
- Nhược điểm: Khó mã hoá song song, thiếu linh hoạt khi nâng cấp từng thành phần.

2. Kiến trúc SOA (Service-Oriented Architecture)

Mô hình SOA chia giao diện người dùng và logic xử lý thành các dịch vụ độc lập, nhưng vẫn dựa vào một cơ sở dữ liệu duy nhất. Ngoài ra, một middleware trung gian (thường là Enterprise Service Bus - ESB) được sử dụng để kết nối các dịch vụ.

- Ưu điểm: Giảm độ phụ thuộc giữa UI và backend, linh hoạt hơn so với Monolithic.
- Nhược điểm: Middleware phức tạp, tính linh hoạt không cao bằng Microservices.

3. Kiến trúc Microservices

Trong mô hình Microservices, mỗi dịch vụ được xây dựng và triển khai độc lập dựa trên các miàn kinh doanh khác nhau. Mỗi Microservice có cơ sở dữ liệu riêng, và các dịch vụ giao tiếp với nhau qua API.

- Ưu điểm: Linh hoạt, dễ bảo trì, dễ mã hoá song song, độ linh hoạt và tính khả năng mở rộng cao.
- Nhược điểm: Phức tạp trong triển khai và quản lý, đối hỏi kiến thức chuyên sâu.

Định nghĩa Microservice

- Microservices là một phương pháp phát triển ứng dụng theo hướng chia nhỏ một ứng dụng đơn lẻ thành nhiều dịch vụ nhỏ. Định nghĩa này được James Lewis và Martin Fowler đưa ra trong một bài viết chuyên sâu về kiến trúc phần mềm.
- Cụ thể, microservices là một tập hợp các dịch vụ nhỏ, hoạt động độc lập, có thể giao tiếp với nhau thông qua các giao thức nhẹ như REST. Trong một ứng dụng web như Easy Bank, chúng ta có thể chia nhỏ thành các dịch vụ như Tài khoản (Accounts), Khoản vay (Loans) và Thẻ (Cards). Mỗi dịch vụ này chạy trong một tiến trình riêng biệt và được xây dựng dựa trên các miền kinh doanh cụ thể.
- Một trong những ưu điểm quan trọng của microservices là khả năng triển khai độc lập thông qua hệ thống triển khai tự động hoàn toàn. Khi một lập trình viên thực hiện thay đổi và đẩy mã nguồn lên kho lưu trữ, quá trình build sẽ tự động diễn ra và triển khai đến môi trường phát triển (Dev) và kiểm thử (UAT). Nếu cần thiết, chúng ta có thể mở rộng quy trình này để triển khai trực tiếp lên môi trường sản xuất bằng cách sử dụng khái niệm CI/CD (Continuous Integration/Continuous Deployment).
- Microservices mang lại nhiều lợi ích, đặc biệt trong việc tăng tốc độ phát triển, dễ dàng mở rộng và nâng cấp từng dịch vụ mà không ảnh hưởng đến toàn bộ hệ thống. Đây là một cách tiếp cận hiệu quả giúp doanh nghiệp tối ưu hóa quá trình phát triển và vận hành ứng dụng.