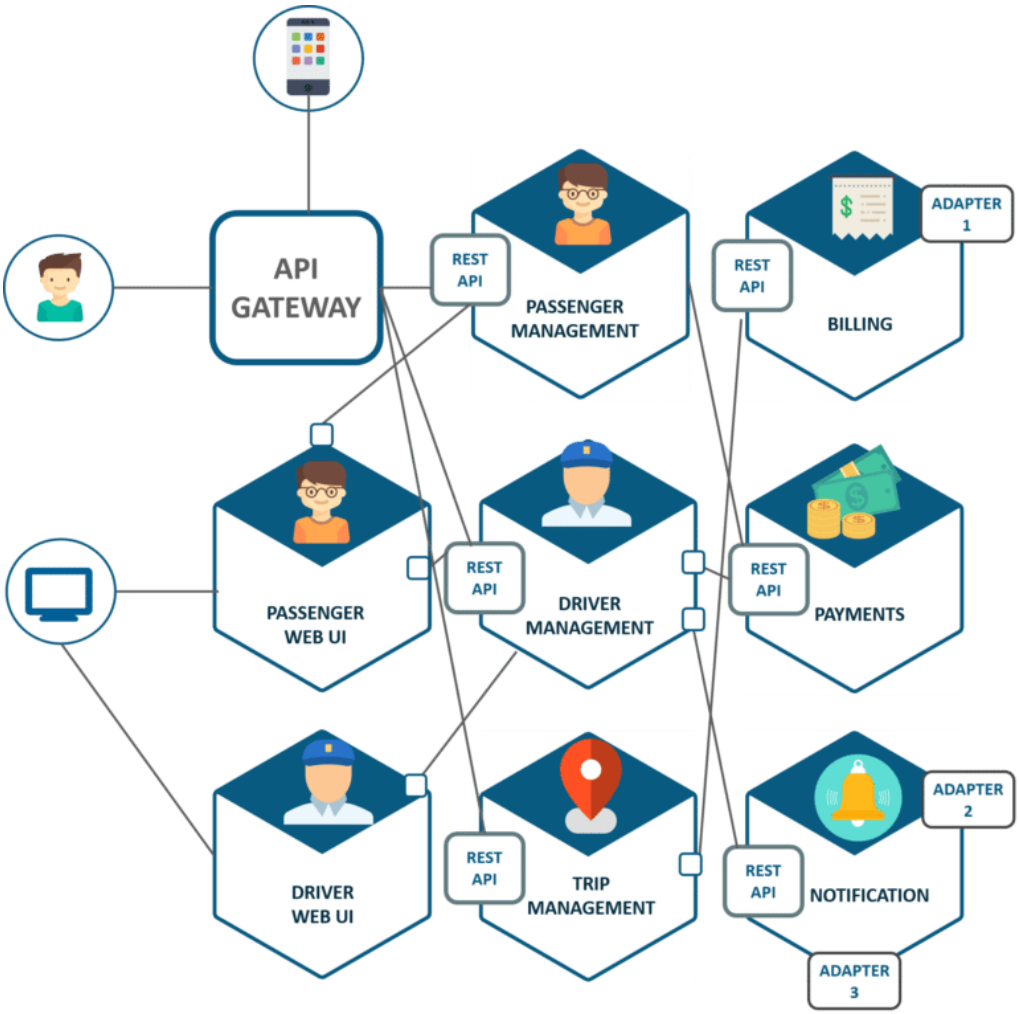
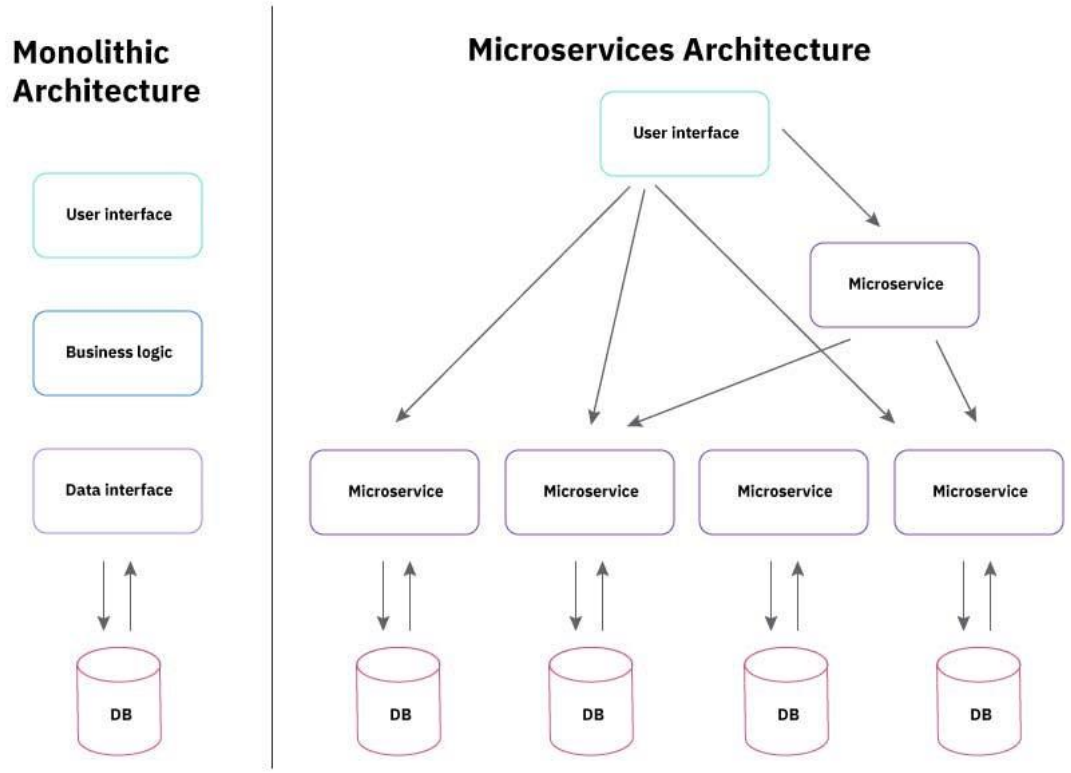
Microservice trong chuyển đổi số?



Theo Wiki Microservices là một kỹ thuật phát triển phần mềm với cấu trúc ứng dụng là một tập hợp các dịch vụ kết hợp lỏng lẻo, nhẹ (services are fine-grained and the protocols are lightweight). Trong Microservices, ứng dụng lớn được phân rã thành các dịch vụ nhỏ hơn, việc này cho phép cải thiện tính mô đun, làm cho ứng dụng dễ hiểu, dễ phát triển, dễ kiểm thử, tạo ra kiến trúc trống chịu tốt hơn với các thay đổi. Microservice cho phép nhiều nhóm có thể phát triển song song một cách độc lập, hiện thực hóa việc triển khai và phân phối liên tục (CI, CD).[1]



Mô hình so sánh kiến trúc truyền thống với Microservice của IBM[2]

TMForum định nghĩa Microservices về cơ bản là các module phần mềm nhỏ, linh hoạt và hoàn toàn khép kín có thể thực hiện một hoạt động cụ thể một cách độc lập, nền tảng này, không giống như việc triển khai phần mềm từ đầu, các dịch vụ nhỏ có thể không cần chỉnh sửa các ứng dụng hiện có tối ưu, nâng cấp. Kết hợp một chức năng mới cũng đơn giản như thêm một thành phần hoặc mô-đun độc lập khác phù hợp với những gì đã có sẵn.[3]

> *"The idea behind microservices is that, rather than building large monolithic applications, development teams instead build apps as collections of loosely coupled services that operate autonomously. They are a special kind of a service-orientated architecture."*

Vì sao nên chuyển đổi Microservice

Microservices hiện đang nhận được rất nhiều sự chú ý: các bài viết, các blog, các cuộc thảo luận trên phương tiện truyền thông, trên mạng xã hội, và các bài thuyết trình hội nghị. Đâu đâu ta cũng có thể bắt gặp những chủ đề liên quan đến Microservices.

Cùng với đó cũng có những hoài nghi trong cộng đồng phần mềm, những người cho rằng microservices không hề có gì mới mẻ. Những người này cho rằng các ý tưởng về

Microservices chỉ là một hình thức làm khác của SOA (Service-Oriented Architecture) hay còn được hiểu là kiến trúc hướng dịch vụ.

Tuy nhiên, bất chấp cả sự cường điệu hóa lẫn những hoài nghi, thì mô hình Microservice mang lại những lợi ích đáng kể trong việc xây dựng những giải pháp phức tạp, đặc biệt là nó tạo điều kiện cho việc áp dụng agile lên project một cách hiệu quả.

Ưu điểm của Microservices

Thứ nhất, giúp đơn giản hóa hệ thống. Với tổng số chức năng không đổi, kiến trúc Microservices chia nhỏ hệ thống ra làm nhiều dịch vụ nhỏ lẽ dể dàng quản lý và triển khai từng phần so với kiến trúc nguyên khối. Mỗi dịch vụ thì được định nghĩa để giao tiếp với các dịch vụ khác thông qua RPC (Remote Procedure Call) hay message-driven API. Kiến trúc Microservices thúc đẩy việc phân tách rạch ròi các dịch vụ nhỏ, việc khó có thể làm nếu xây dựng ứng dụng bằng kiến trúc một khối. Trên hết với mỗi dịch vụ nhỏ, chúng ta sẽ có thời gian phát triển nhanh hơn, dễ nắm bắt cũng như bảo trì hơn.

Thứ hai, kiến trúc này cho phép việc mỗi dịch vụ được phát triển độc lập bởi những team khác nhau. Cũng như cho phép developer có thể tự do chọn lựa technology stack cho mỗi dịch vụ mình phát triển. Dĩ nhiên tự do lựa chọn nhưng không phải là tạo ra một mớ hỗn độn về technology, điều này thì chẳng có một dự án hay sản phẩm nào mong muốn cả. Tuy nhiên, sự tự do này có nghĩa là các developer không còn phải bắt buộc phải sử dụng các công nghệ lỗi thời có thể đã tồn tại vào lúc bắt đầu dự án. Khi viết một dịch vụ mới, họ có tùy chọn của việc sử dụng công nghệ bắt kịp với xu thế. Hơn nữa, vì dịch vụ là tương đối nhỏ, việc viết lại một dịch vụ cũ dựa trên nền tảng công nghệ mới hơn là hoàn toàn khả thi.

Thứ ba, kiến trúc Microservices cho phép mỗi dịch vụ có thể được triển khai một cách độc lập. Cùng với đó thì việc triển khai hệ thống theo kiểu continuous deployment là hoàn toàn có thể. Cuối cùng, kiến trúc Microservices cho phép mỗi dịch vụ có thể thực hiện việc scale một cách độc lập. Bạn có thể scale dễ dàng bằng cách tăng số instance phục vụ cho mỗi dịch vụ lên và phân tải bằng load balancer. Ngoài ra bạn cũng có thể tối ưu chi phí vận hành dịch vụ bằng cách triển khai mỗi dịch vụ lên server có resource thích hợp. Ví dụ về việc này tôi đã nêu ra trong bài địa ngục kiến trúc một khối.

Đánh giá của Gartner

Top 10 Technology Trends Impacting Infrastructure & Operations năm 2016 Gartner đề cập “Trend 3: Containers, Microservices and Application Streams”

Container (ví dụ: Docker) và Microservices là nền tảng ứng dụng mới cho phát triển đám mây. Container cho phép thực hiện phân đoạn quy trình, phù hợp với việc phát triển các dịch vụ nhỏ ở đó các ứng dụng được xây dựng như một bộ dịch vụ nhỏ chạy theo các quy trình riêng biệt. Microservices có thể được triển khai và quản lý độc lập, và một khi được thực thi trong một Container, chúng có ít tương tác trực tiếp với hệ điều hành bên dưới.

TMForum: How to transform OSS/BSS using microservices[4]

Thử thách chính của việc chuyển đổi OSS/BSS sẽ là các hệ thống được chuẩn hóa, mở và linh hoạt như thế nào. Để đảm bảo điều này, CSP nên áp dụng kiến trúc phần mềm microservices. Nó dựa trên nguyên tắc phá vỡ các hệ thống CNTT lớn, phức tạp thành nhiều thành phần dễ quản lý và tự trị, mỗi thành phần giải quyết một yêu cầu kinh doanh cụ thể.

Tại sao lại là microservices

Cách tiếp cận đơn giản hơn để cung cấp các dịch vụ mới và tối ưu hóa các dịch vụ hiện có, mở đường cho việc nâng cấp và bảo trì CNTT nhanh chóng, không có ma sát và tiết kiệm chi phí. Hơn nữa, các dịch vụ nhỏ, thường sử dụng các phương thức nhanh, theo mô hìnhDevOps, có thể giúp CSP mở rộng việc triển khai OSS/BSS của họ phù hợp với các yêu cầu nghiệp vụ biến động liên tục.

Những thách thức của microservices

Tuy nhiên, đối với nhiều CSP, việc chuyển sang thế giới của các dịch vụ nhỏ vẫn là một thách thức, phần lớn vì hai lý do. Thứ nhất, phát triển các ứng dụng như một nhóm các dịch vụ kết hợp lỏng lẻo, các dịch vụ độc lập đòi hỏi một cách tiếp cận phát triển phần mềm cơ bản khác với các phương pháp thông thường được sử dụng để xây dựng các ứng dụng quy mô lớn. Vì vậy, điều này đòi hỏi một skillset hoàn toàn khác nhau.

Thứ hai, văn hóa tại nhiều tổ chức viễn thông có thể không mở rộng cho các cải tiến quy mô rộng phá vỡ các nguyên tắc kiến trúc cơ bản của họ, họ thường lựa chọn giữ nguyên trạng của các khung phần mềm quen thuộc, hiện có.

MuleSoft: Microservices and DevOps: Better together[5]

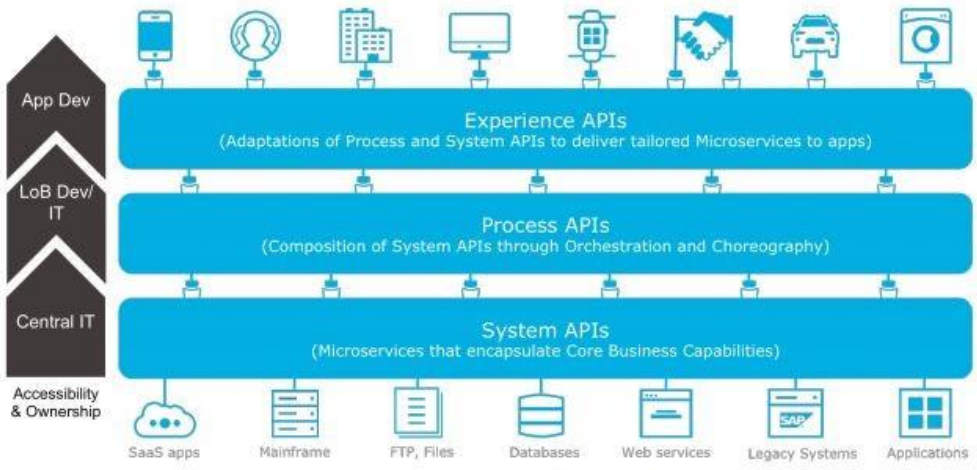


Figure 1. Microservices classification

Kiến trúc dựa trên microservices đáp ứng sự thay đổi, thay đổi thường được đón nhận bởi những người tạo ra các ứng dụng hiện đại. Những người nắm lấy sự thay đổi đó nhận thấy rằng năng suất đang gia tăng ở một tốc độ đáng kinh ngạc, và các giải pháp có thể được chuyển giao nhanh hơn nhiều cho những người yêu cầu các ứng dụng linh hoạt, có khả năng mở rộng. Đối với những người trong lĩnh vực DevOps, microservices mang lại một số lợi ích đáng kể, bao gồm:

- Khả năng triển khai: rút ngắn chu trình xây dựng sản phẩm, dựa trên mô hình nhỏ, hoạt động độc lập.
- Độ tin cậy: Một lỗi với một microservice chỉ ảnh hưởng đến dịch vụ trong Microservice đó và người tiêu dùng của nó. Khi các ứng dụng quy mô lớn gặp lỗi, nhiều khả năng cả ứng dụng bị ảnh hưởng.
- Tính khả dụng: Việc phát hành phiên bản mới của một dịch vụ trong Microservice cụ thể đòi hỏi rất ít thời gian Downtime, trong khi tung ra một phiên bản mới của dịch vụ triển khai trên ứng dụng truyền thống thường đòi hỏi phải khởi động lại toàn bộ hệ thống.
- Khả năng mở rộng: Microservices có thể được thu nhỏ một cách độc lập bằng cách sử dụng pools, clusters, grids. Đặc điểm triển khai này làm cho các dịch vụ nhỏ trở thành một kết hợp tuyệt vời cho tính đàn hồi của đám mây.
- Khả năng thay đổi: Microservices cung cấp sự linh hoạt để tương thích các khung công tác mới, thư viện, nguồn dữ liệu và các tài nguyên khác. Khi được kết hợp lỏng lẻo, các thành phần mô-đun, microservices chứng minh khả năng làm việc linh hoạt hơn.
- Quản lý: Microservices có thể tận dụng phương pháp Agile, ở đó nỗ lực phát triển ứng dụng được chia cho các nhóm nhỏ hơn và hoạt động độc lập hơn.

Microservices mang lại năng suất bổ sung cho DevOps bằng cách sử dụng một bộ công cụ phổ biến, có thể được sử dụng cho cả phát triển và hoạt động. Bộ công cụ phổ biến này thiết lập thuật ngữ chung, cũng như các quy trình cho các yêu cầu, phụ thuộc và các vấn đề. Điều này lại giúp Devs và Ops làm việc với nhau dễ dàng hơn, cho phép các thực thể đó cùng nhau làm việc trên một vấn đề và sửa chữa thành công cấu hình xây dựng hoặc xây dựng tập lệnh. DevOps và microservices hoạt động tốt hơn khi được áp dụng cùng nhau.

Open API

Kiến trúc microservices dựa trên API để cung cấp các dịch vụ microservice cho các ứng dụng khác. TM Forum đã xây dựng và đề xuất các CSP một bộ API mở được ánh xạ tới eTOM và SID.

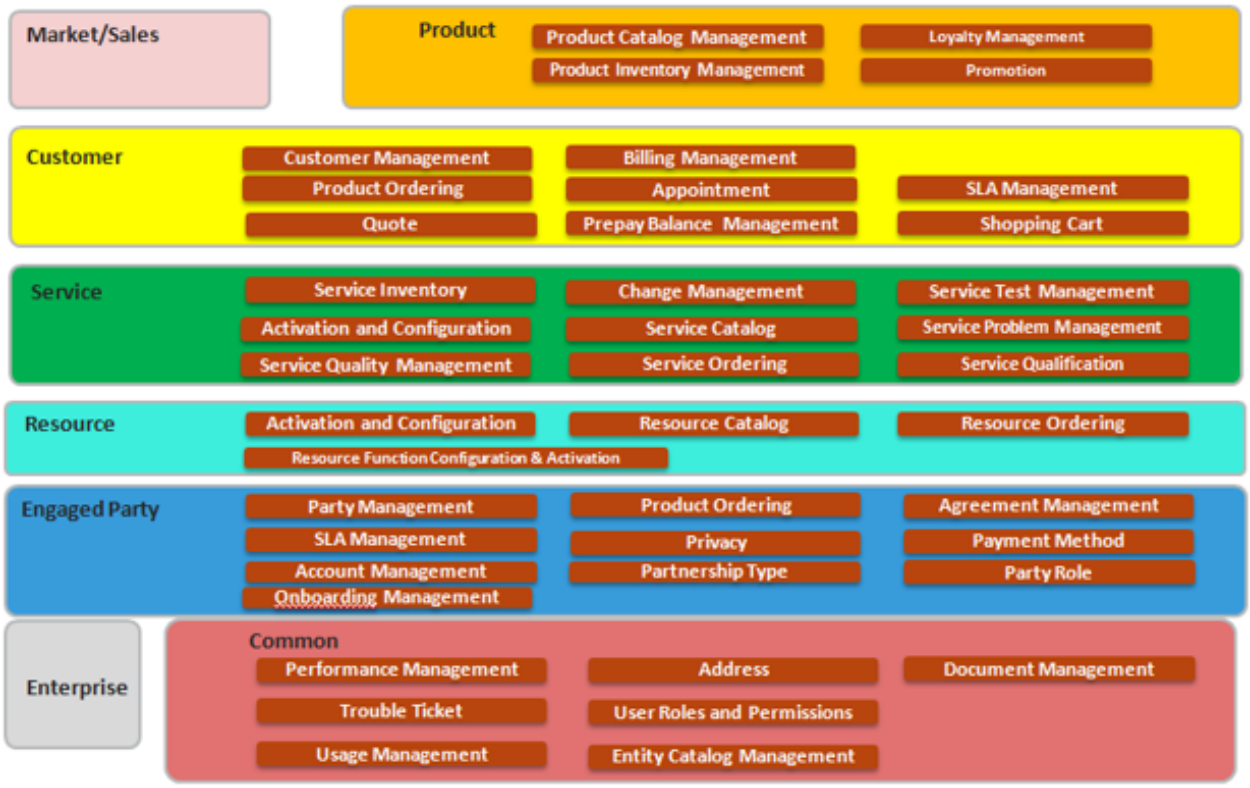
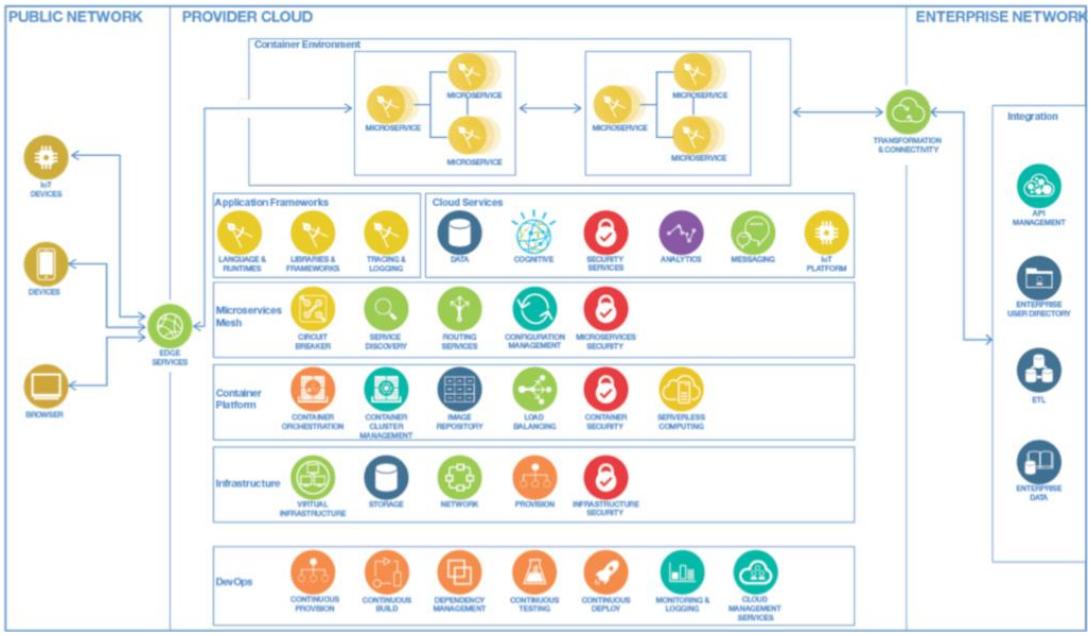


Figure 3- API Map Level 1(Current/Existing)

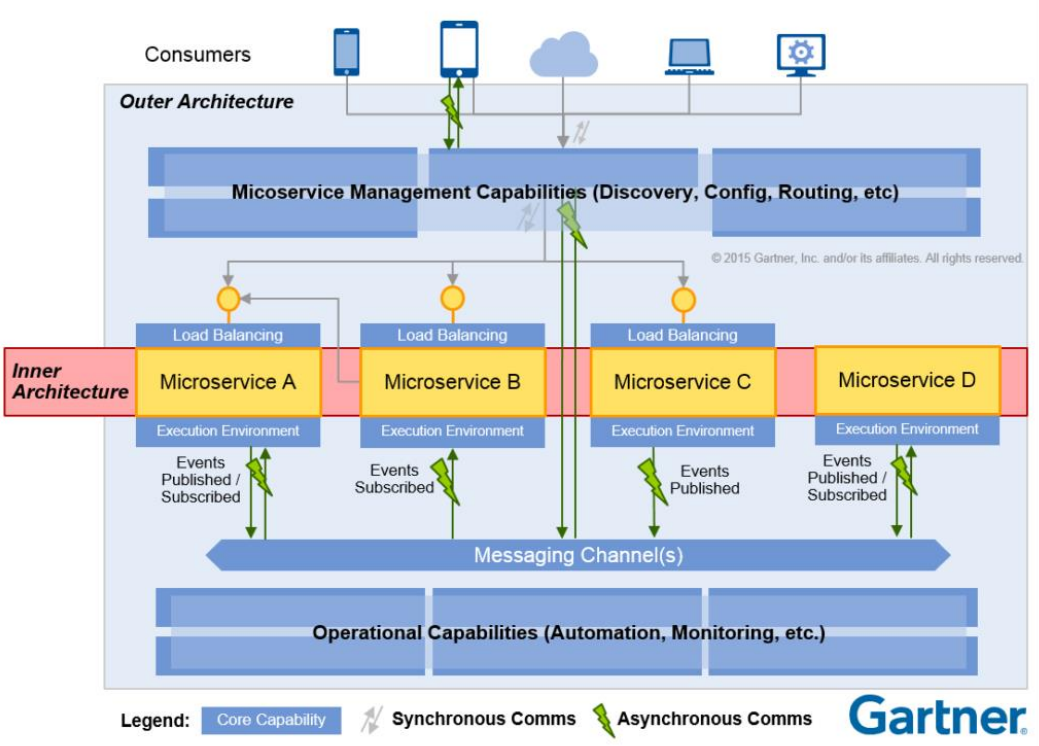
Kiến trúc Micrservices tham chiếu

IBM Microservice Refefence Architecture[6]



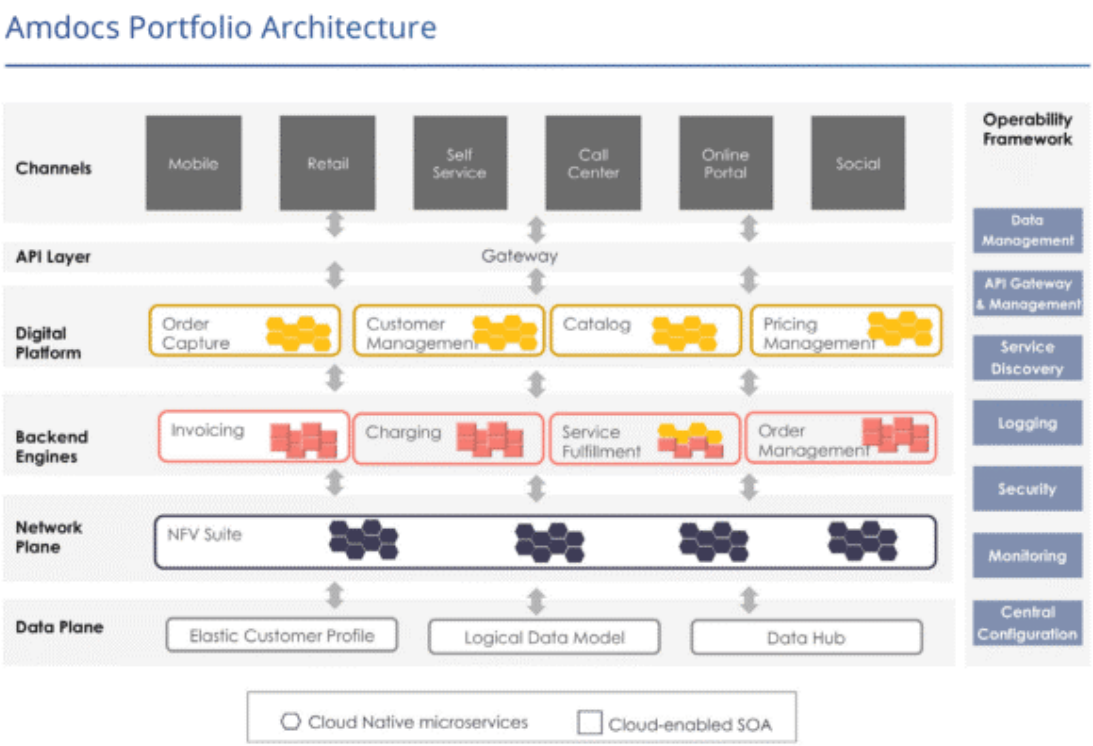
Sự khác biệt lớn nhất giữa hai cách tiếp cận là cách chúng được triển khai. Trong nhiều năm, các ứng dụng đã được đóng gói trong một thời trang nguyên khối – đó là một nhóm các nhà phát triển đã xây dựng một ứng dụng lớn đã làm mọi thứ cần thiết cho một nhu cầu kinh doanh. Sau khi được xây dựng, ứng dụng đó sau đó được triển khai nhiều lần trên một Fame của các máy chủ ứng dụng. Ngược lại, với phong cách kiến trúc microservices, các nhà phát triển độc lập xây dựng và đóng gói một số ứng dụng nhỏ hơn mà mỗi ứng dụng chỉ thực hiện các phần của toàn bộ ứng dụng.

Gartner định nghĩa Microservices Inner and Outer Architecture[7]

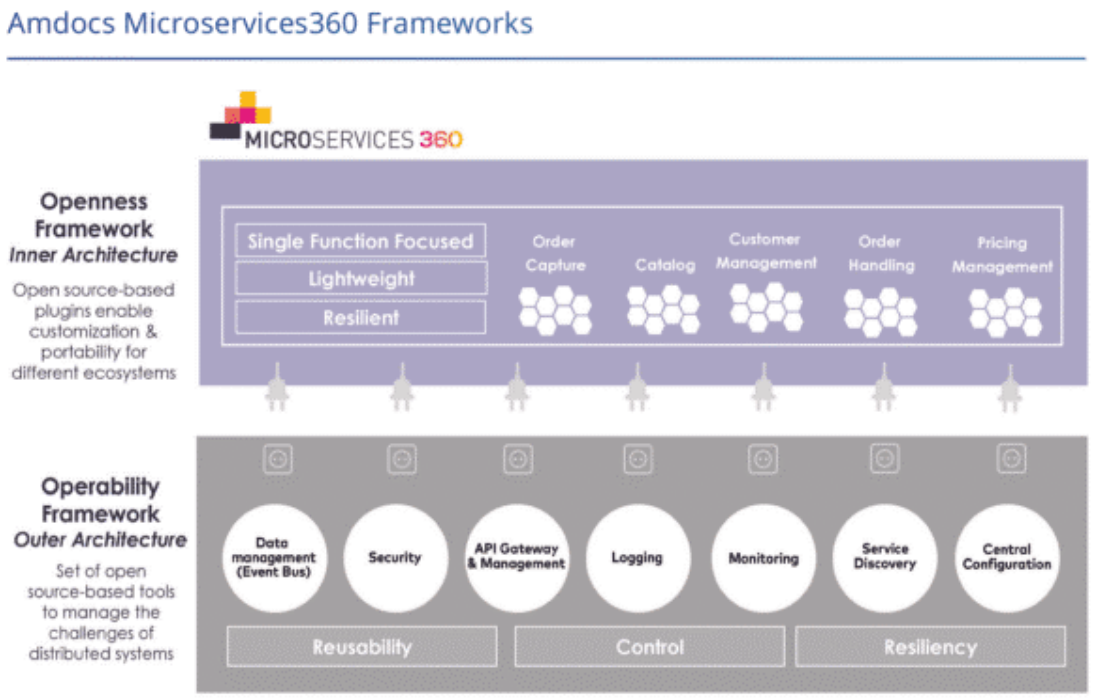


Amdocs đề xuất giải pháp Microservice360[8]

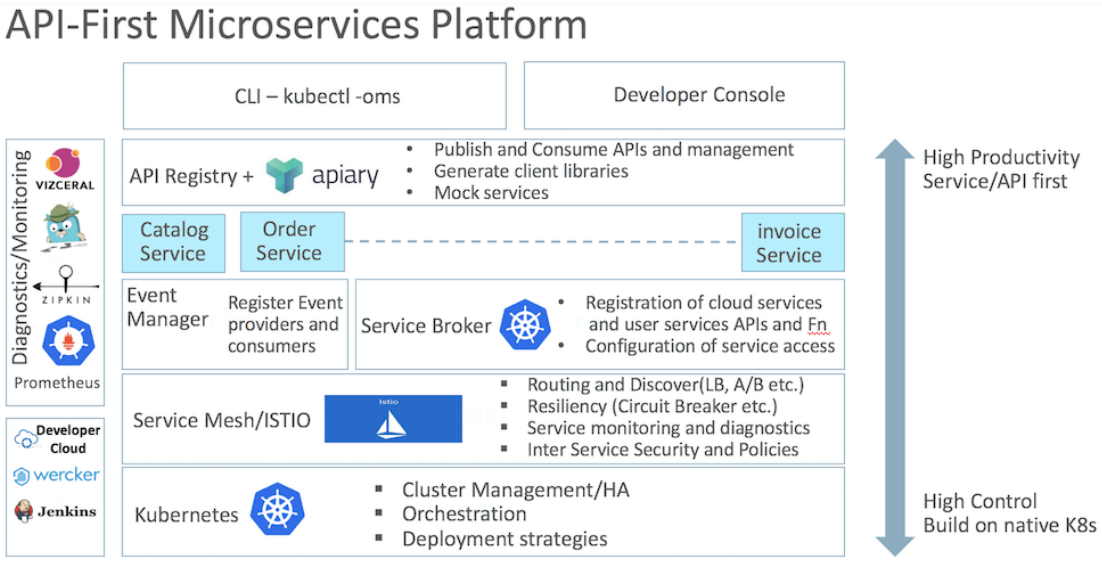
Microservices-based applications: được mô tả như hình vẽ



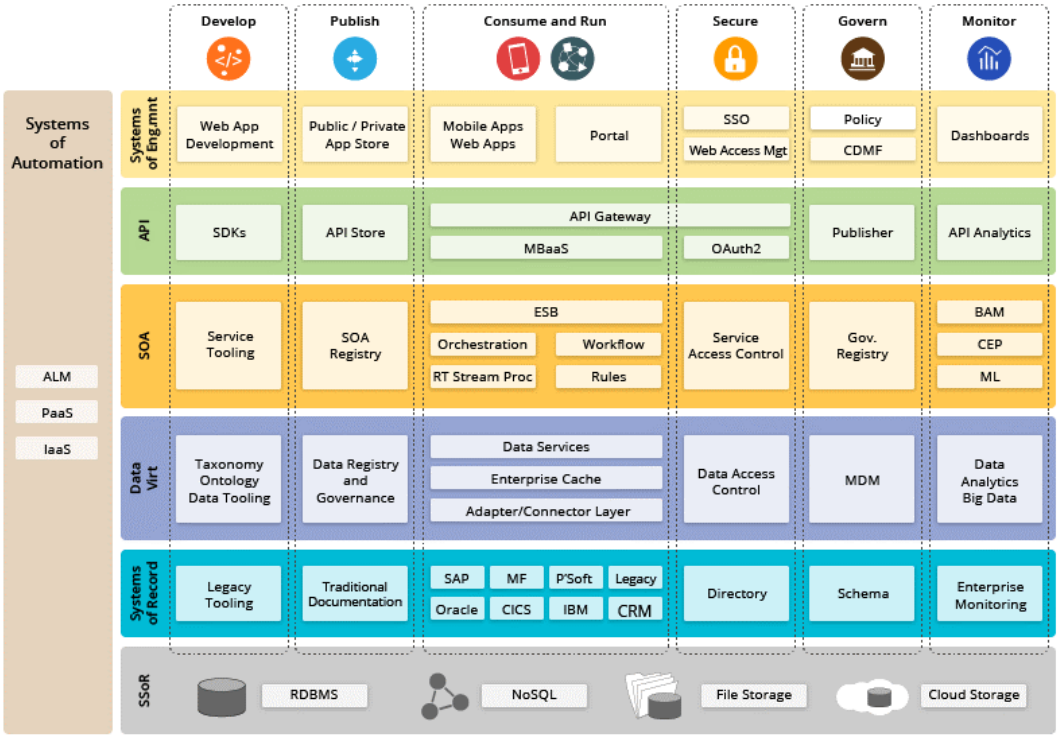
Microservices orchestration and operations platform: Dựa trên công cụ cộng đồng mã nguồn mở hàng đầu bao gồm Kubernetes và Docker, Amdocs Microservices360 framework cung cấp bộ công cụ được chứng nhận cho các hoạt động đóng gói, cấu hình, thời gian chạy và các hoạt động hàng ngày của đám mây. Amdocs Microservices360 framework đảm bảo khả năng hoạt động cho các CSP với một bộ công cụ như phát hiện dịch vụ, bảo mật, giám sát, ghi nhật ký, cấu hình, quản lý dữ liệu và quản lý API, đồng thời đảm bảo sự cởi mở thông qua phương pháp tiếp cận cơ sở hạ tầng cho phép microservices chạy trên bất kỳ hạ tầng đám mây nào.



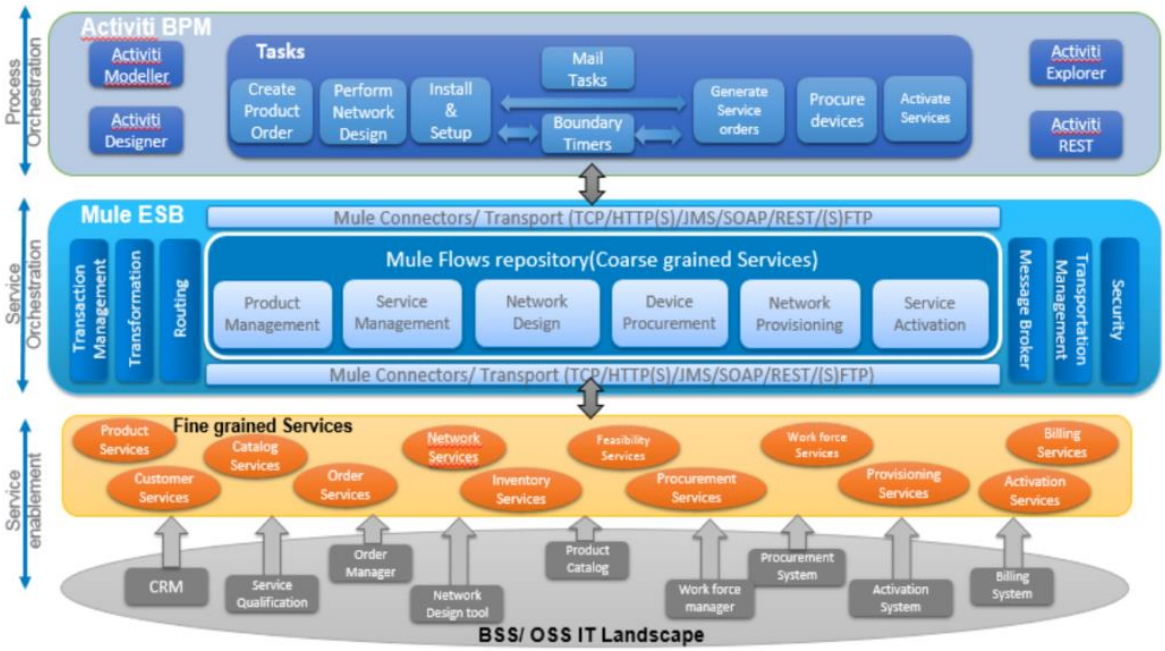
Oracle API-first Microservice Platform[9]



WSO2 Microservice Architecture[10]



Info Sys[11]



Ref:

[2] IBM – What are microservices and why are they important?

[3] TMForum – How to transform OSS/BSS using microservices

[4] TMForum – How to transform OSS/BSS using microservices

[5] MuleSoft – Microservices and DevOps: Better together

[6] IBM – Microservices reference architecture

[7] Gartner – Microservices : Building Services with the Guts on the Outside

[8] Amdocs – Putting Microservices into Practice: Taking microservices from theory into reality to achieve business agility

[9] Oracle – An API First Approach to Microservices Development

[10] WSO2 – WSO2 Microservices Framework for Java

[11] TMForum – Transforming BSS/ OSS systems to Microservices Architecture