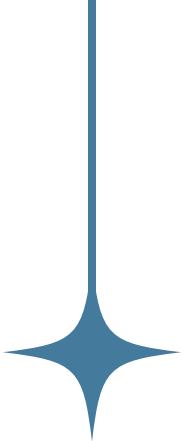


Microserve —& Monolith





1. Microserve

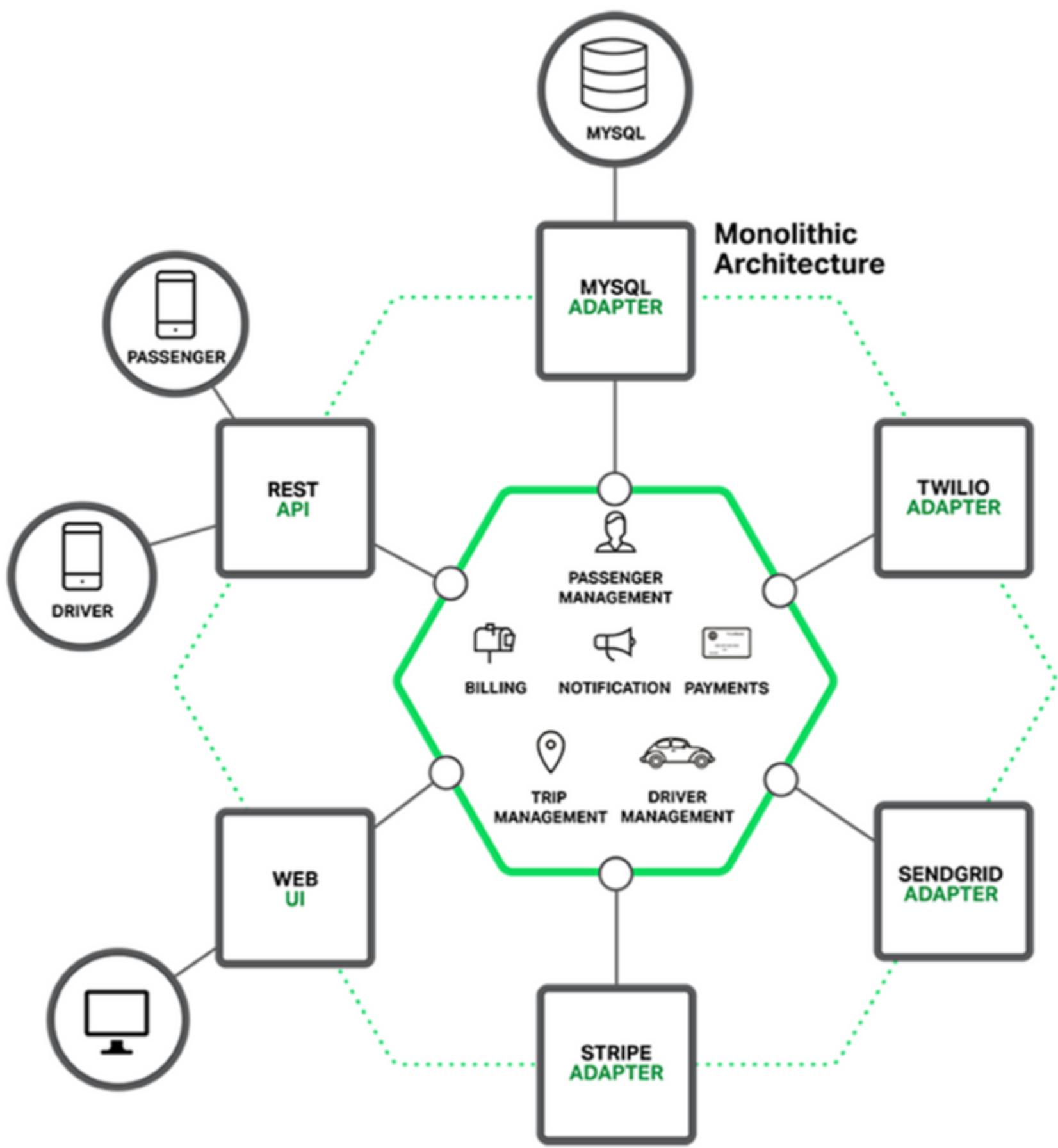




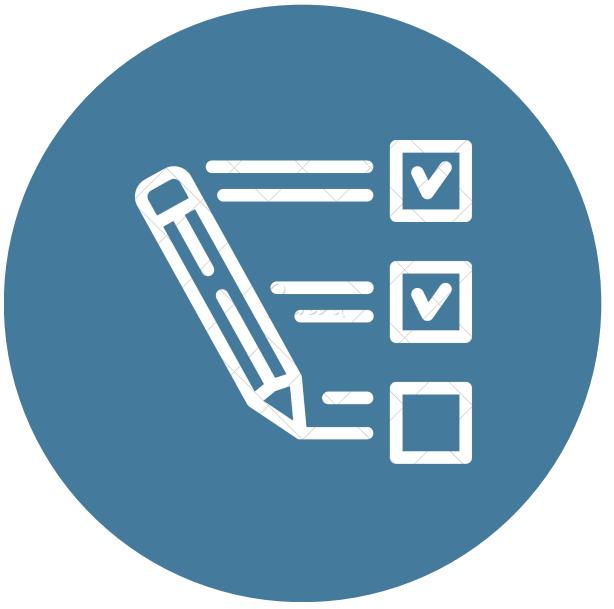
Monolith là một kiến trúc phần mềm truyền thống, trong đó toàn bộ ứng dụng được phát triển, triển khai và triển khai như một thực thể duy nhất.



Sơ đồ Monolith



Ưu Điểm



Kiến trúc Đơn Địệu

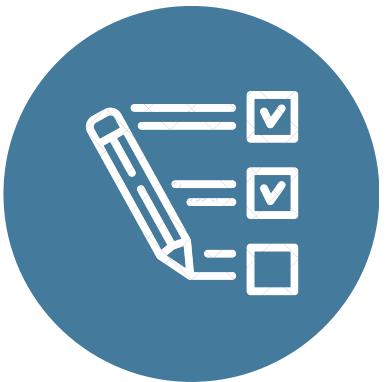


Dễ Phát Triển, Dễ Bảo Trì



Dễ Hiểu

Nhược Điểm



**Khả Năng Mở Rộng
Khó Khăn**



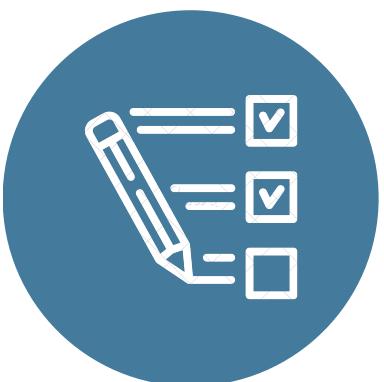
**Phát Triển và Triển
Khai Chậm**



**Quản Lý Codebase
Khó Khăn**



**Phụ Thuộc Chặt
Chẽ**

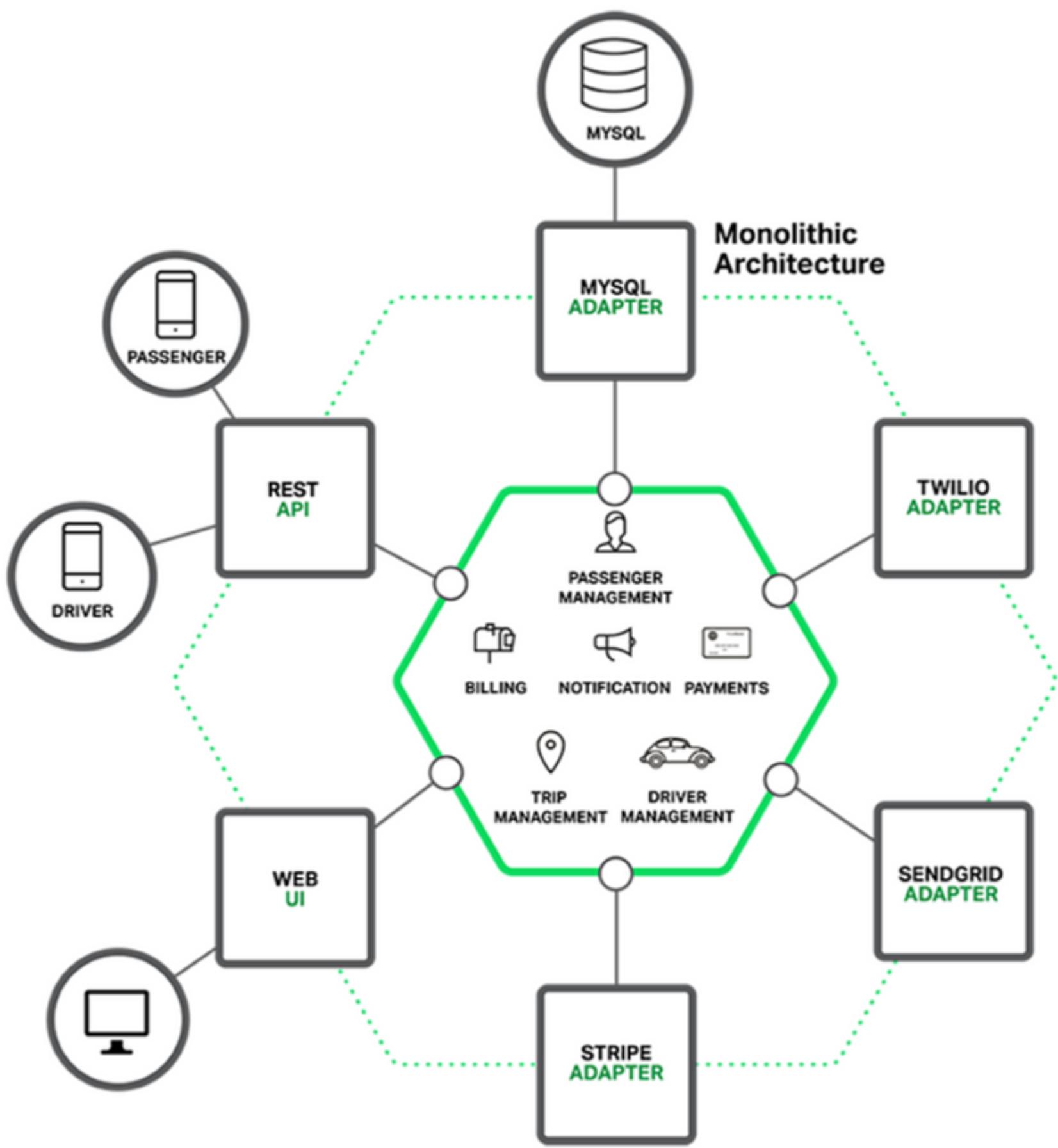


**Khó Tích Hợp Công
Nghệ Mới**



**Khả Năng Tồn Tại
SPOF (Single Point
of Failure)**

Sơ đồ Monolith





Microservice



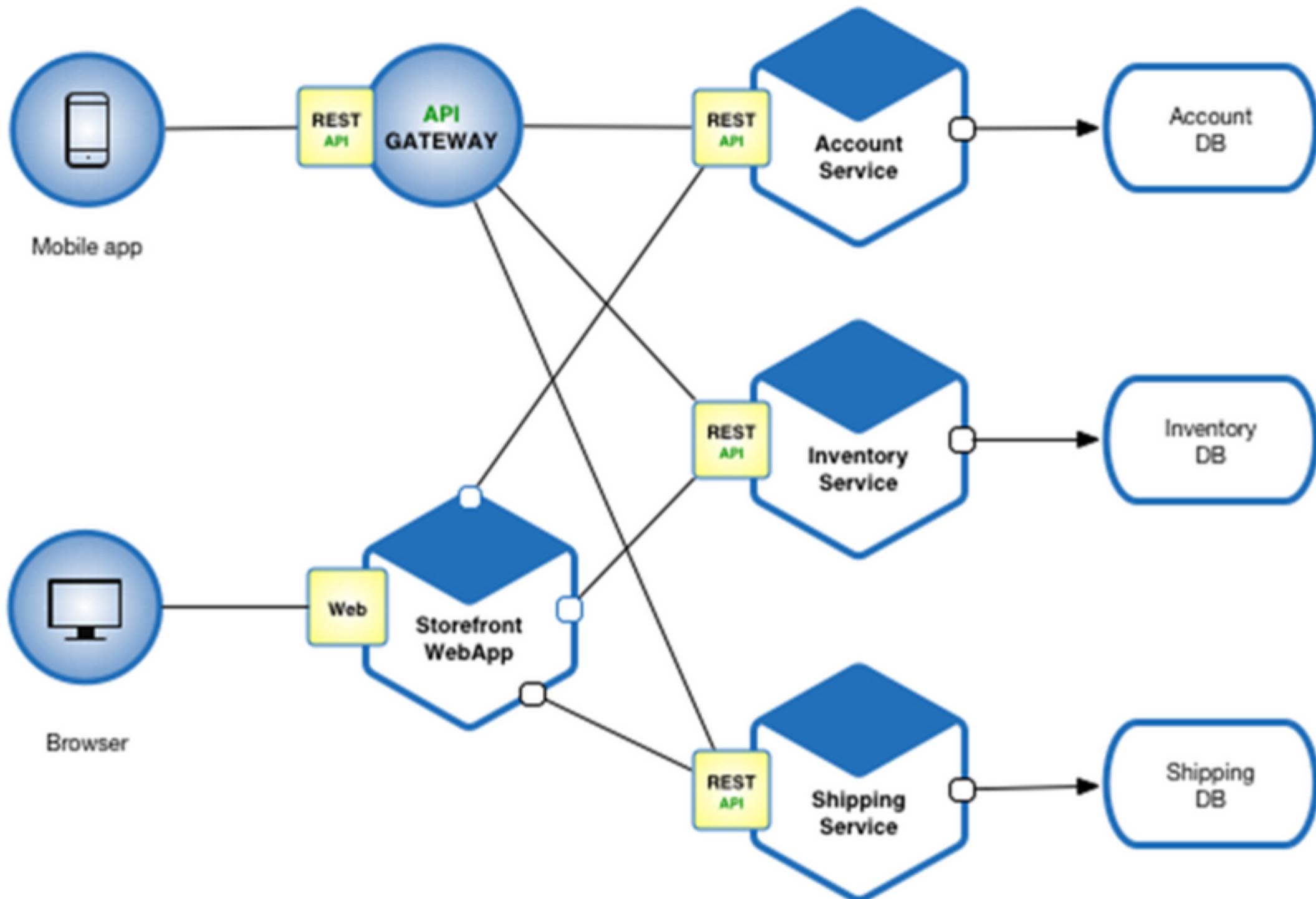


Microservice là gì

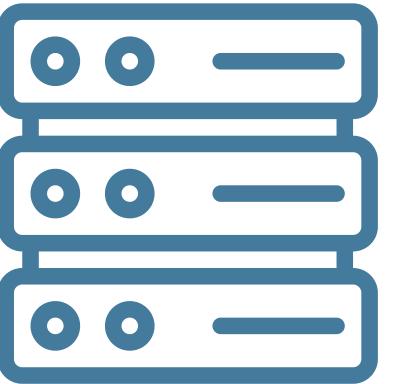
- Microservices là một kiến trúc phần mềm mà ứng dụng được chia thành các dịch vụ nhỏ, độc lập với nhau và chạy trên các quy trình riêng biệt. Mỗi dịch vụ trong kiến trúc Microservices chịu trách nhiệm cho một phần cụ thể của ứng dụng và có thể được triển khai, mở rộng và quản lý độc lập.



Sơ đồ Microservice



Lợi Ích của Microservice



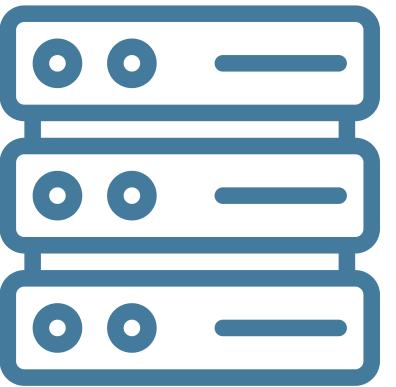
Linh hoạt và mở rộng

Độc lập công nghệ

Phân Chia Chức Năng, công việc

Giao Tiếp Thông Qua API:

Lợi Ích của Microservice



Dễ Quản Lý và Bảo Trì:

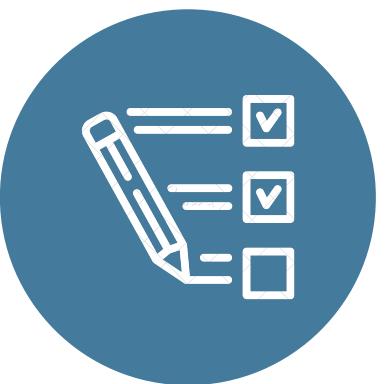
Tính Mở Rộng Tốt:

Độ Đàn Hồi (Resilience):

Linh Hoạt Công Nghệ:

Quản Lý Cơ Sở Dữ Liệu Dễ Dàng Hơn:

Thúc Đẩy Sự Đổi Mới Công Nghệ:
• Sự độc lập giữa các dịch vụ hỗ trợ việc



Phức Tạp Hóa Việc Phát Triển Ban Đầu:

- Xây dựng và triển khai một hệ thống microservices có thể phức tạp hơn so với việc phát triển một ứng dụng monolith.

Kiến trúc Microservices đòi hỏi sự quản lý phức tạp hơn do sự phân tách của các dịch vụ và việc phải xử lý các vấn đề về giao tiếp và đồng bộ giữa chúng. Việc phải duy trì và điều phối các dịch vụ độc lập nhau có thể tạo ra thách thức quản lý và phát triển hệ thống.



Khó hiểu và duy trì

- : Với nhiều dịch vụ riêng biệt, mã nguồn của kiến trúc Microservices có thể trở nên phức tạp và khó hiểu hơn so với Monolith. Sự phân tách của các dịch vụ và các tương tác giữa chúng đòi hỏi sự hiểu biết sâu về kiến trúc và mã nguồn, cũng như khả năng duy trì và khắc phục sự cố trong hệ thống phân tán.



Quản Lý Đồng Bộ Hóa và Giao Tiếp:

- Việc quản lý giao tiếp giữa các dịch vụ và đồng bộ hóa chúng có thể làm phức tạp và đòi hỏi giải pháp chặt chẽ.



Nhược Điểm

- Chi phí triển khai và quản lý có thể tăng lên vì cần phải duy trì và theo dõi nhiều dịch vụ độc lập.
- Khi có lỗi, việc theo dõi và xác định nguồn gốc có thể trở nên phức tạp do sự phân tách giữa các dịch vụ.



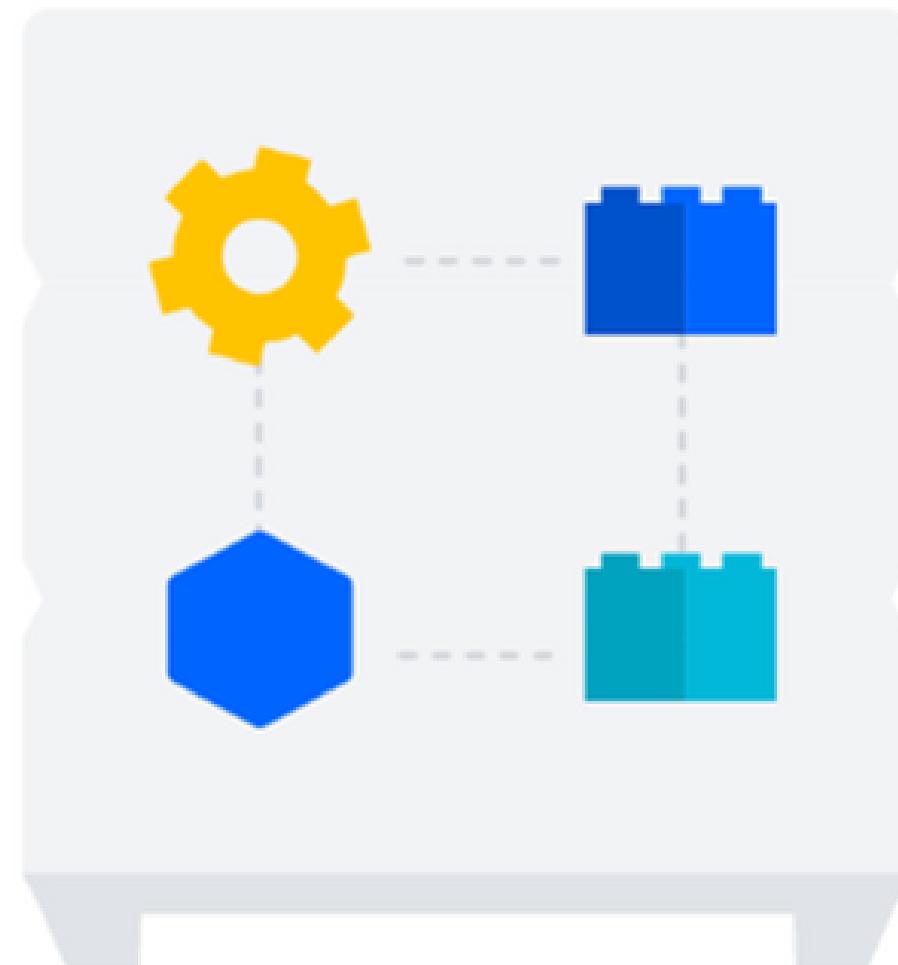
So Sánh

	Monolith	Microservice
Kiến Trúc	<p>Tất cả các tính năng như quản lý người dùng, quản lý sản phẩm, thanh toán và giao hàng đều được tích hợp trong một ứng dụng lớn.</p>	<ul style="list-style-type: none">● Có các Microservices riêng lẻ cho quản lý người dùng, quản lý sản phẩm, thanh toán và giao hàng.● Mỗi Microservice có cơ sở dữ liệu riêng và giao tiếp thông qua API.
Phát Triển và Triển Khai	<ul style="list-style-type: none">● Tất cả các tính năng được phát triển và triển khai cùng một lúc.● Khi có sự thay đổi trong một chức năng, cần phải triển khai lại toàn bộ ứng dụng..	<ul style="list-style-type: none">● Các nhóm phát triển có thể làm việc độc lập trên từng Microservice, giúp giảm xung đột và tăng tốc quá trình phát triển.● Việc triển khai mỗi Microservice không ảnh hưởng đến các dịch vụ khác
Khả Năng Mở Rộng	<p>Nếu muốn mở rộng khả năng xử lý của ứng dụng, bạn cần phải mở rộng toàn bộ hệ thống, bao gồm cả chức năng không cần thiết.</p>	<p>Nếu chỉ cần mở rộng chức năng thanh toán (ví dụ), bạn có thể tăng cường chỉ Microservice quản lý thanh toán, không cần phải mở rộng toàn bộ hệ thống</p>

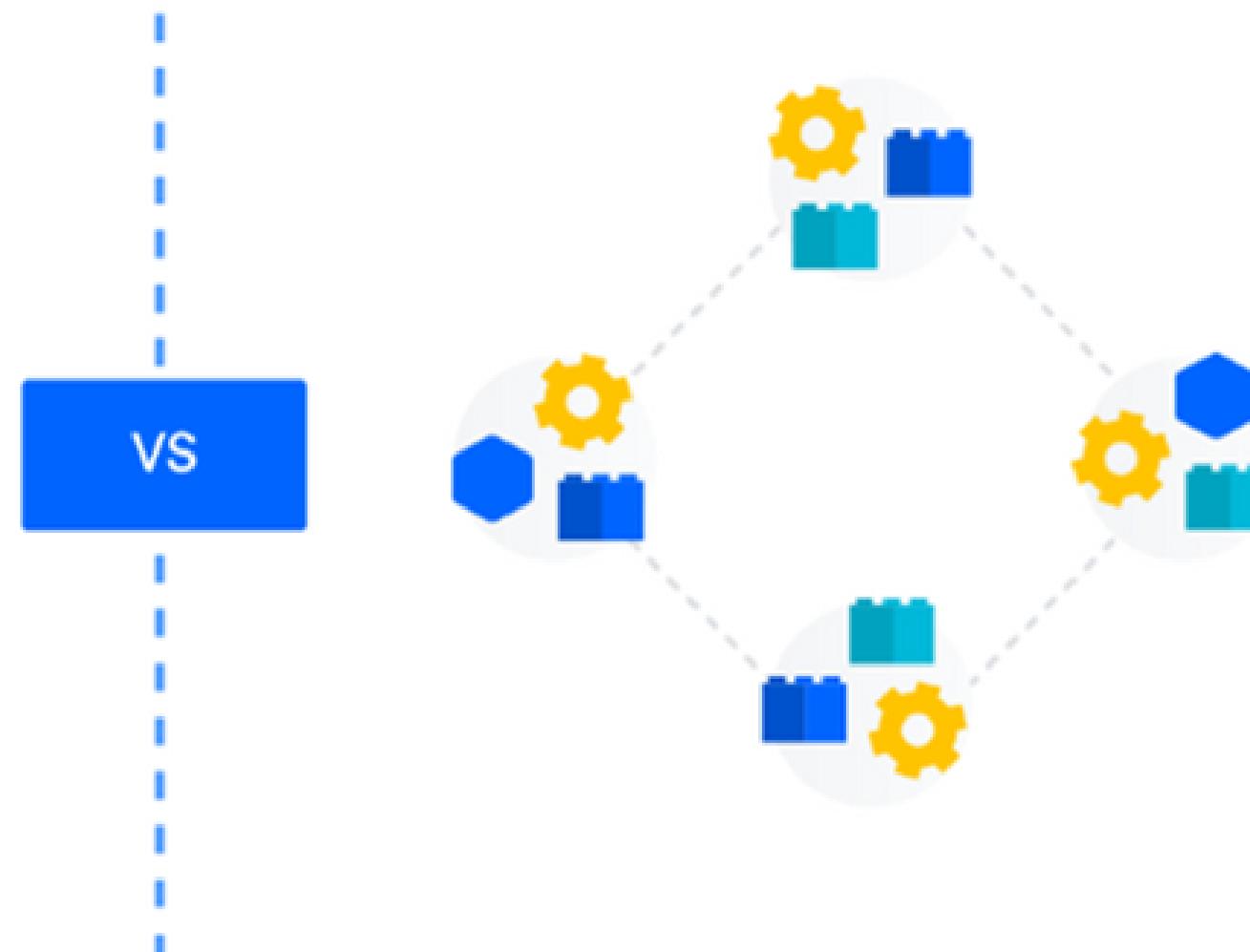
So Sánh



Monolith

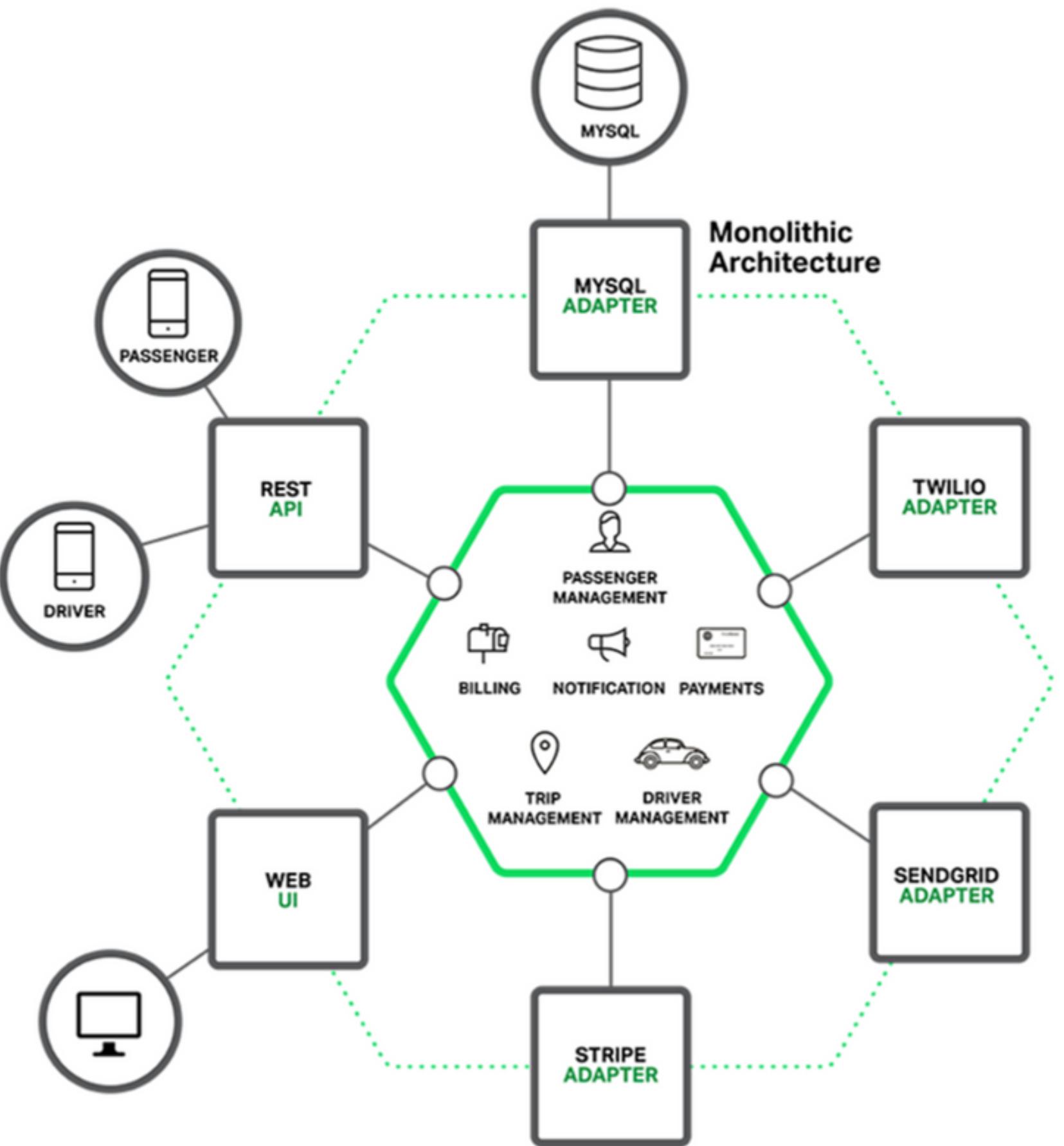


Microservices



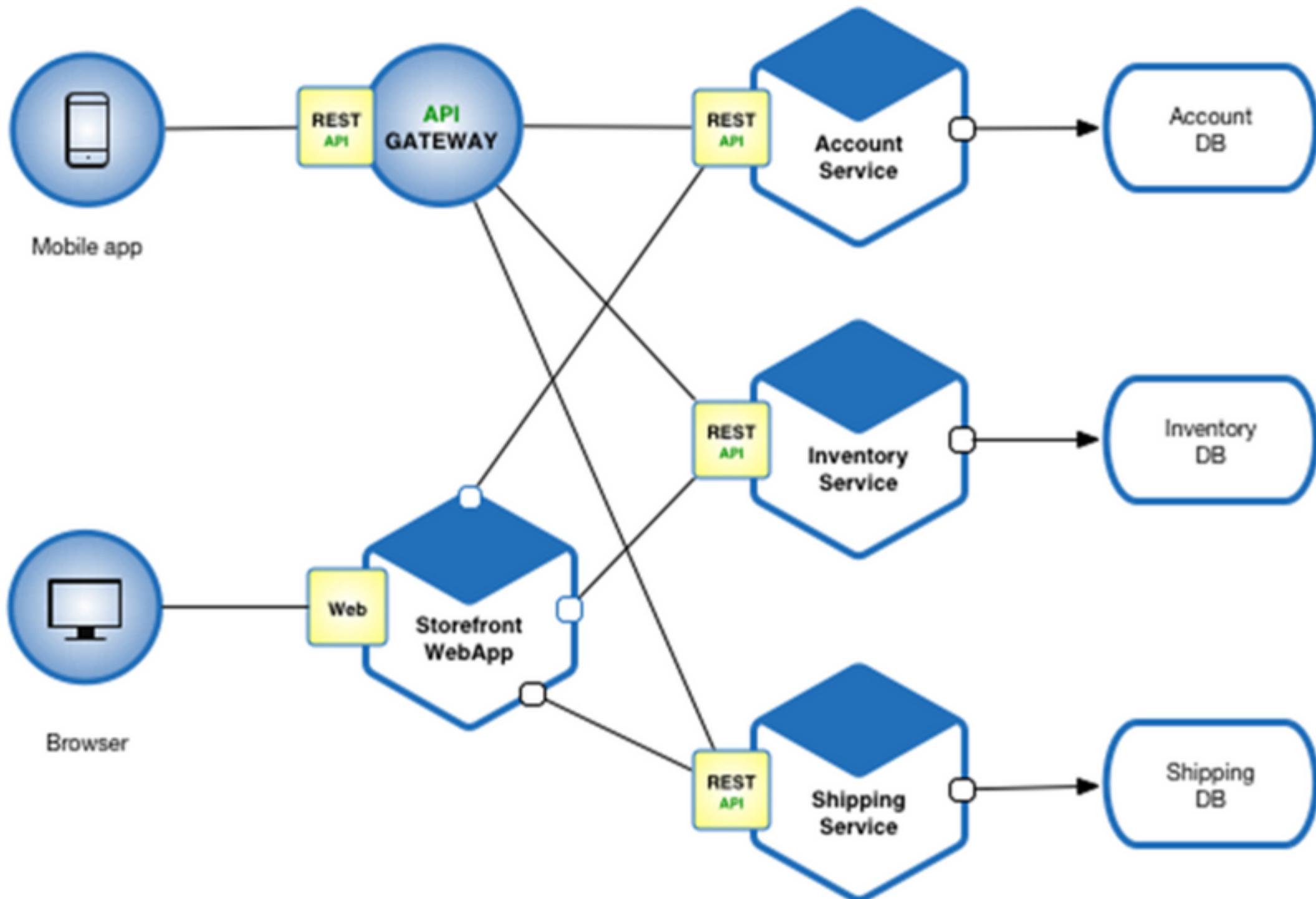
VS

Sơ đồ Monolith



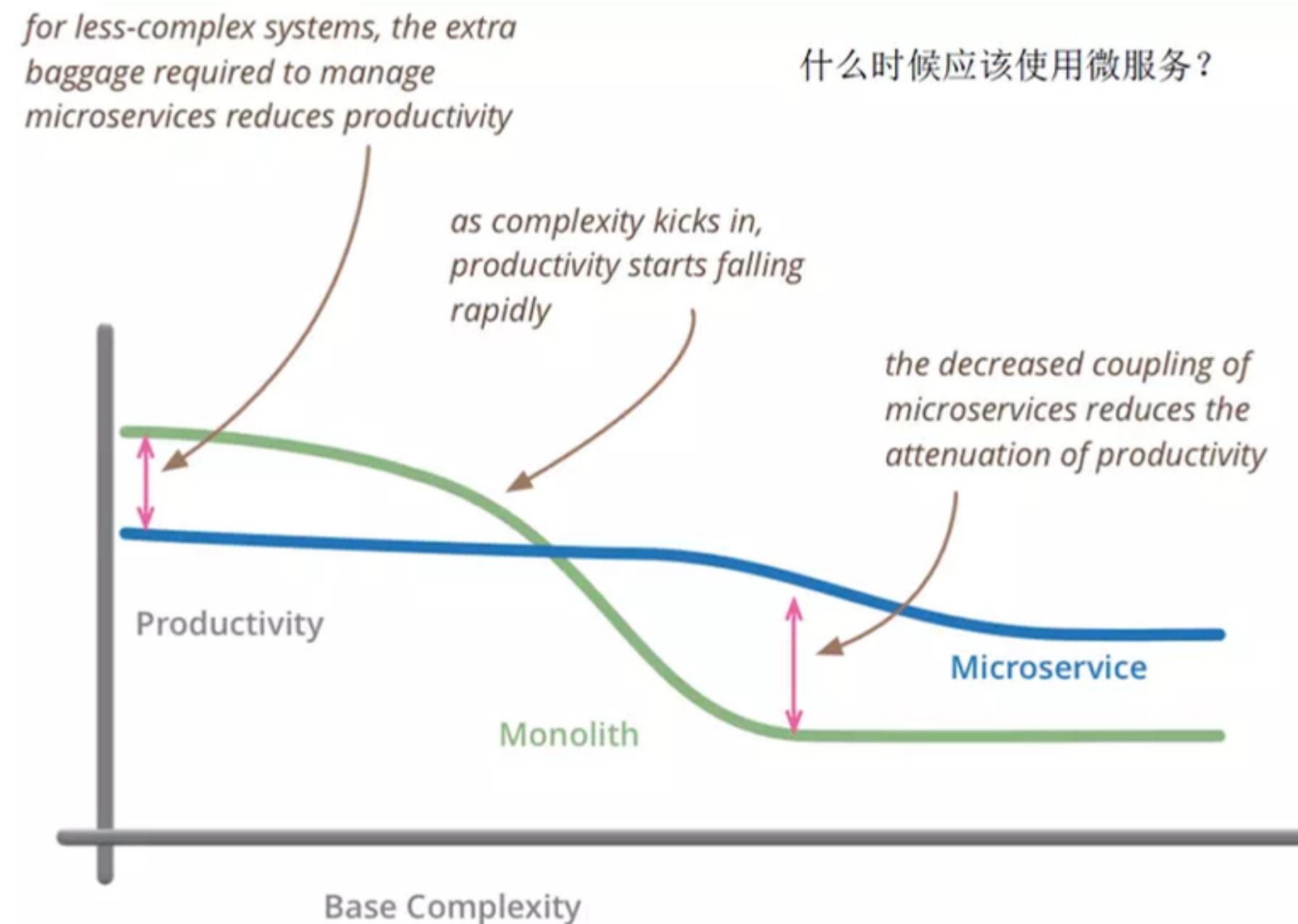


Sơ đồ Microservice





So Sánh

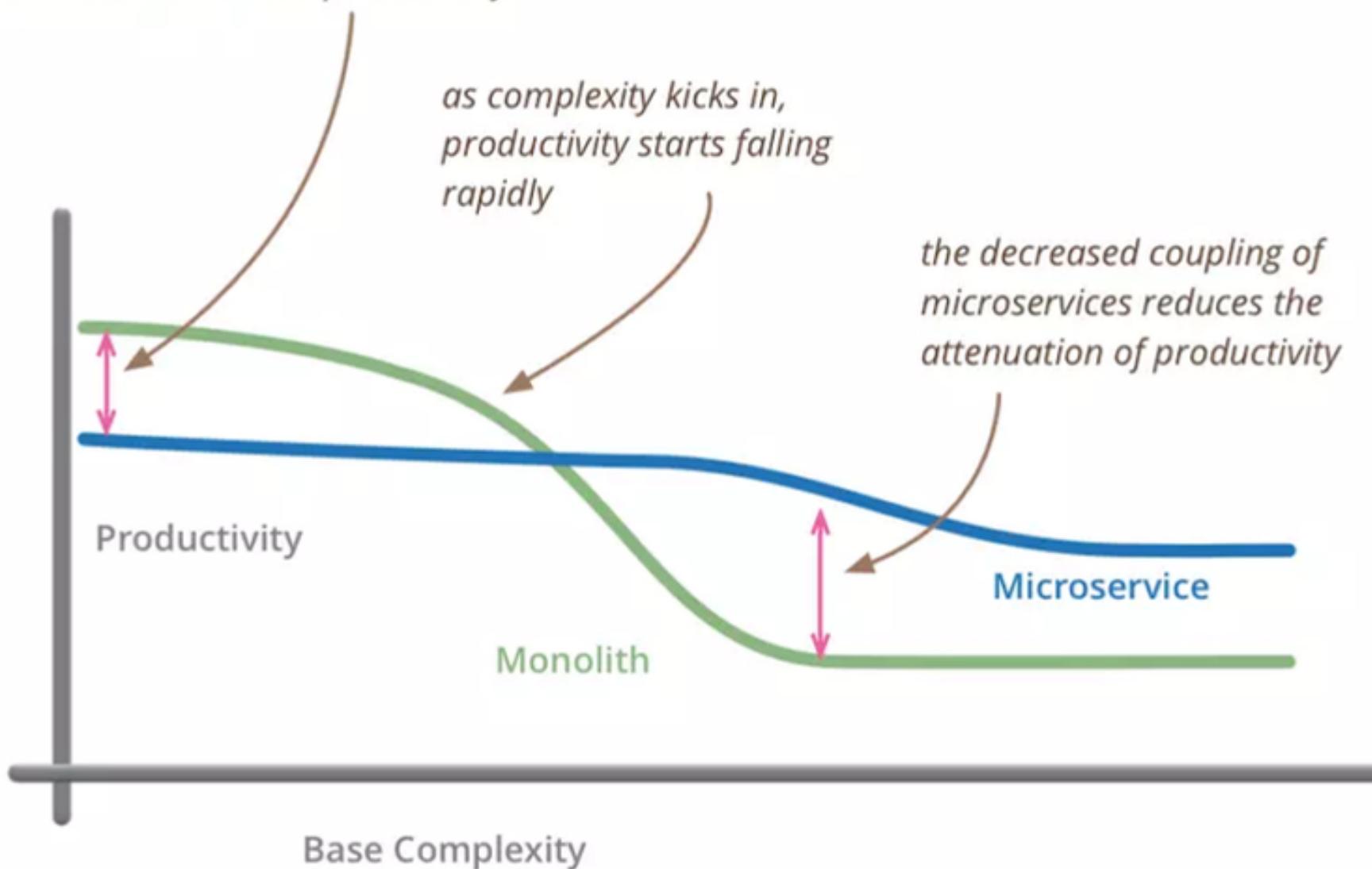




So Sánh

for less-complex systems, the extra baggage required to manage microservices reduces productivity

什么时候应该使用微服务？



but remember the skill of the team will outweigh any monolith/microservice choice



**Thank You
For Your Time!**