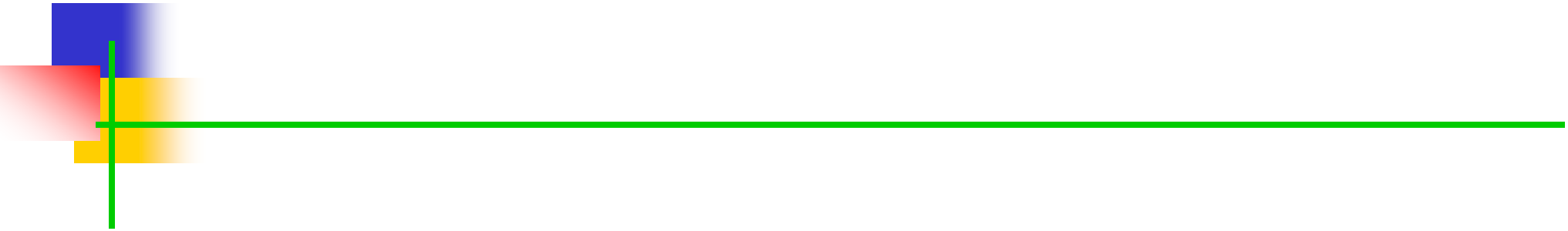
Microservices



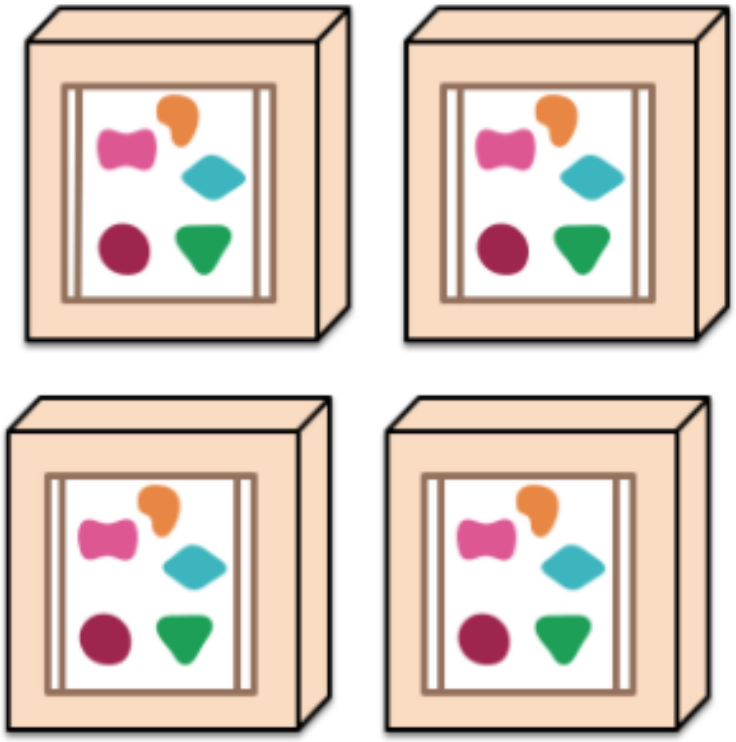
Microservice Architectural Style

- The microservice architectural style is an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API.
- These services are built around business capabilities and independently deployable by fully automated deployment machinery.
- There is a bare minimum of centralized management of these services, which may be written in different programming languages and use different data storage technologies.
- Microservices vs. monolithic style?
- Scaling monolithic applications?

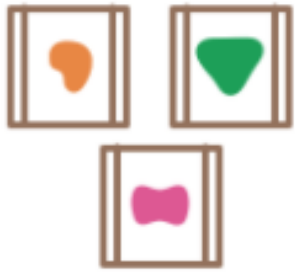
Microservice Architectural Style

A monolithic application puts all its functionality into a single process...

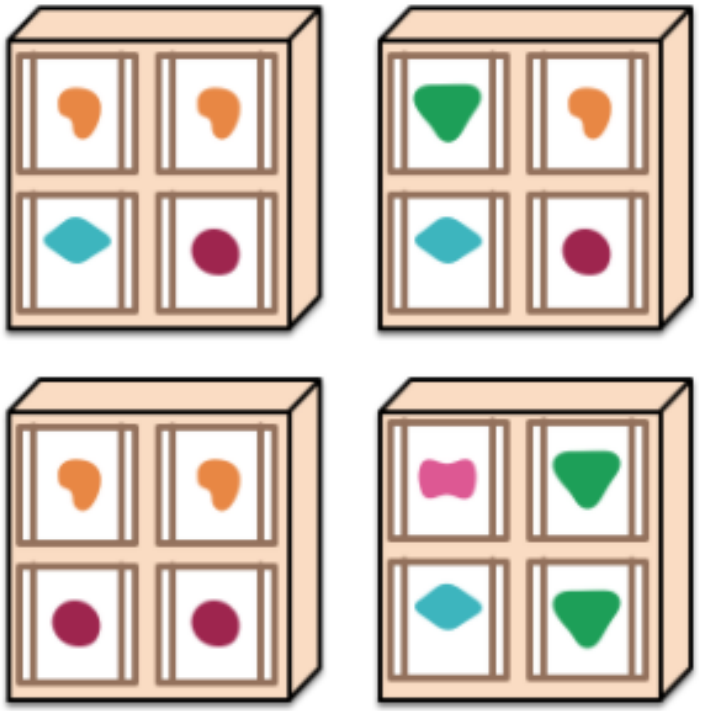
... and scales by replicating the monolith on multiple servers



A microservices architecture puts each element of functionality into a separate service...



... and scales by distributing these services across servers, replicating as needed.



- Limitations of monolithic applications?

- Limitations of monolithic applications?
  - Change cycles
  - Modular structure
  - Scaling
  - Start times
  - Reliability

Microservice Architectural Style

- Many organizations, such as Amazon, eBay, and Netflix, have solved this problem by adopting what is now known as the Microservices Architecture pattern.
- Instead of building a single monstrous, monolithic application, the idea is to split your application into set of smaller, interconnected services.
- A service typically implements a set of distinct features or functionality, such as order management, customer management, etc.

5

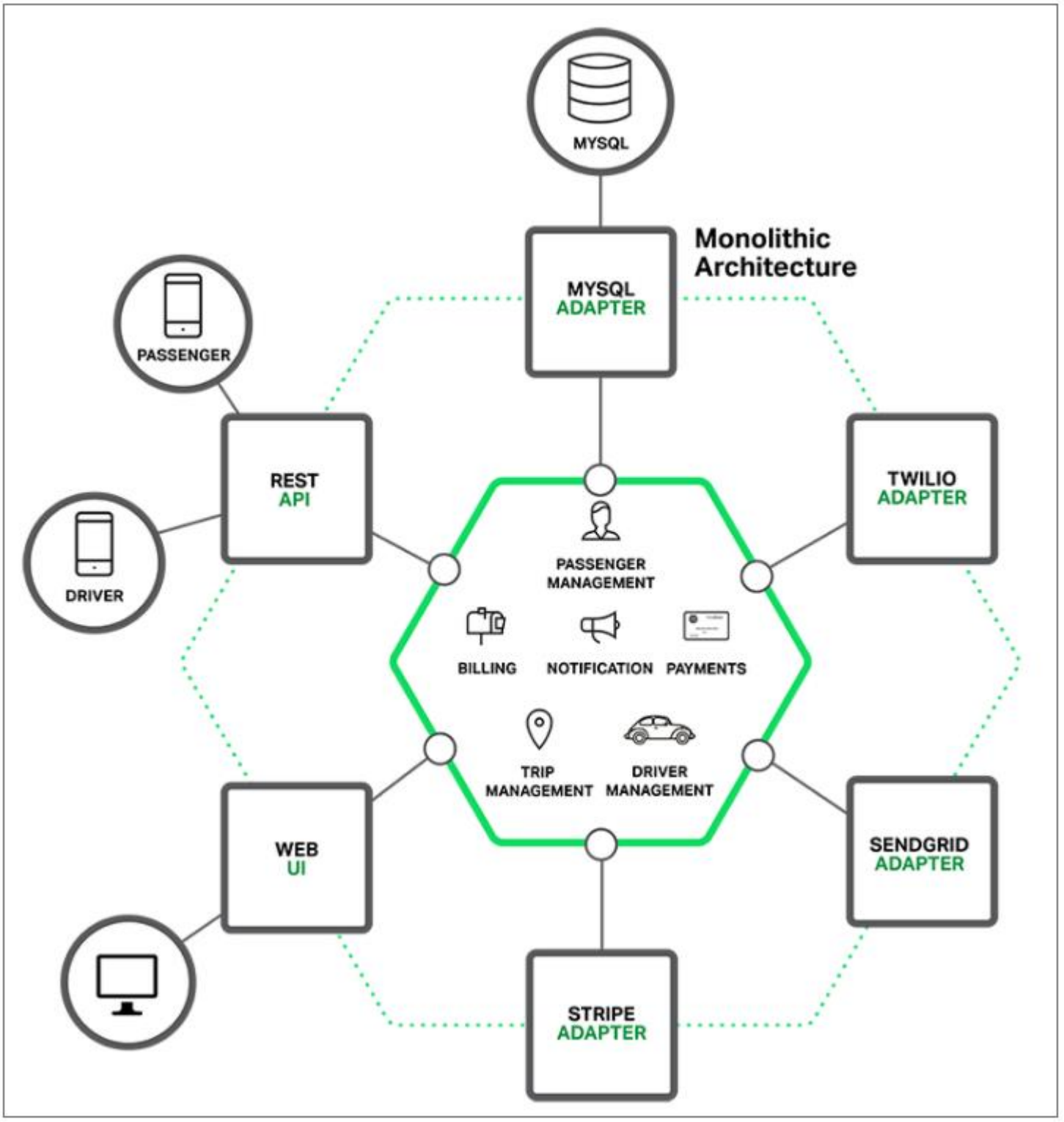
Microservice Architectural Style

- Each microservice is a mini-application that has its own hexagonal architecture consisting of business logic along with various adapters.
- Some microservices would expose an API that's consumed by other microservices or by the application's clients. Other microservices might implement a web UI. At runtime, each instance is often a cloud VM or a Docker container.
- Example:
  - Consider that you are building a competitor application to Uber
  - How would the architecture look?

6

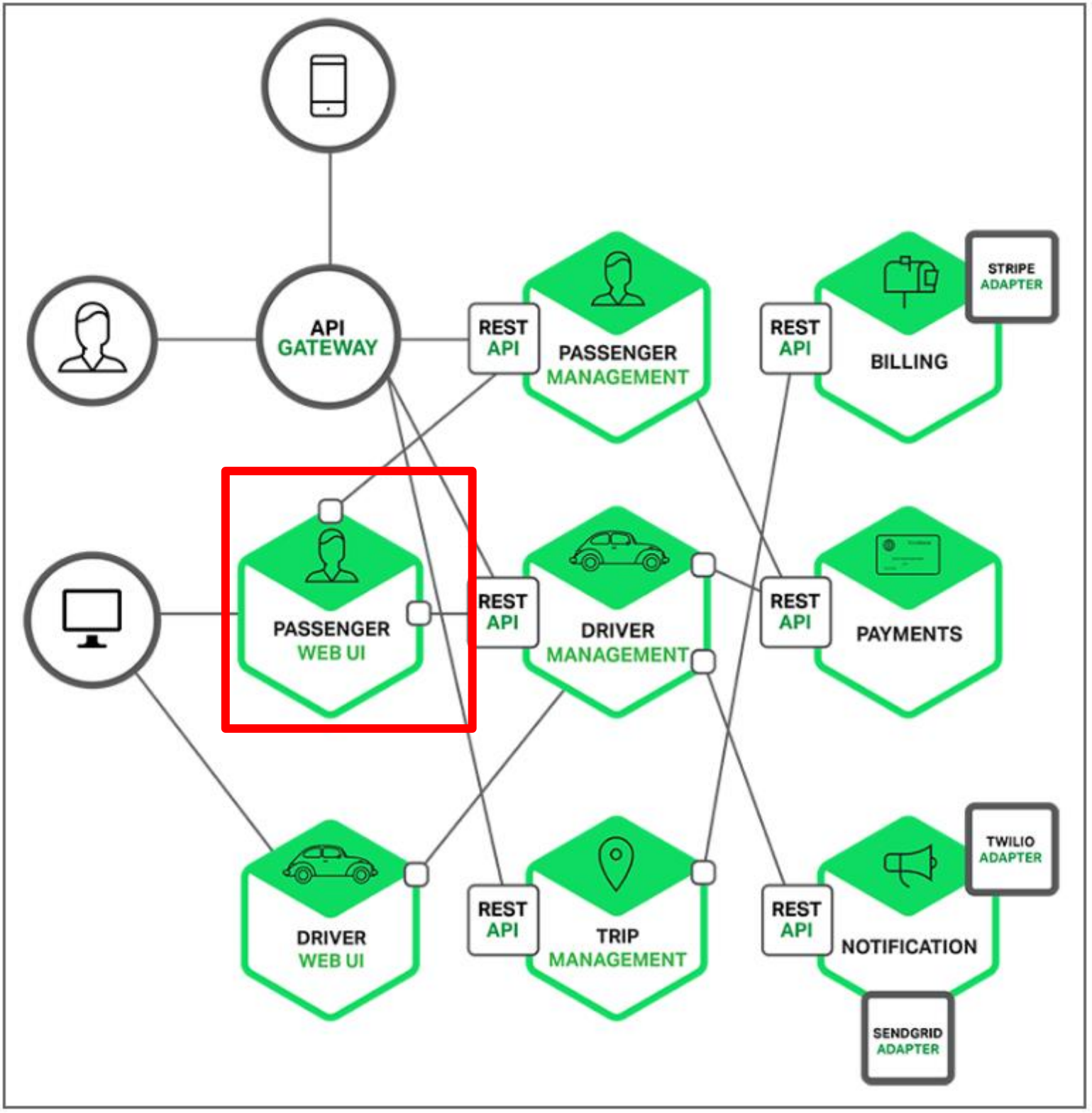
Uber competitor monolithic architecture

Pros?



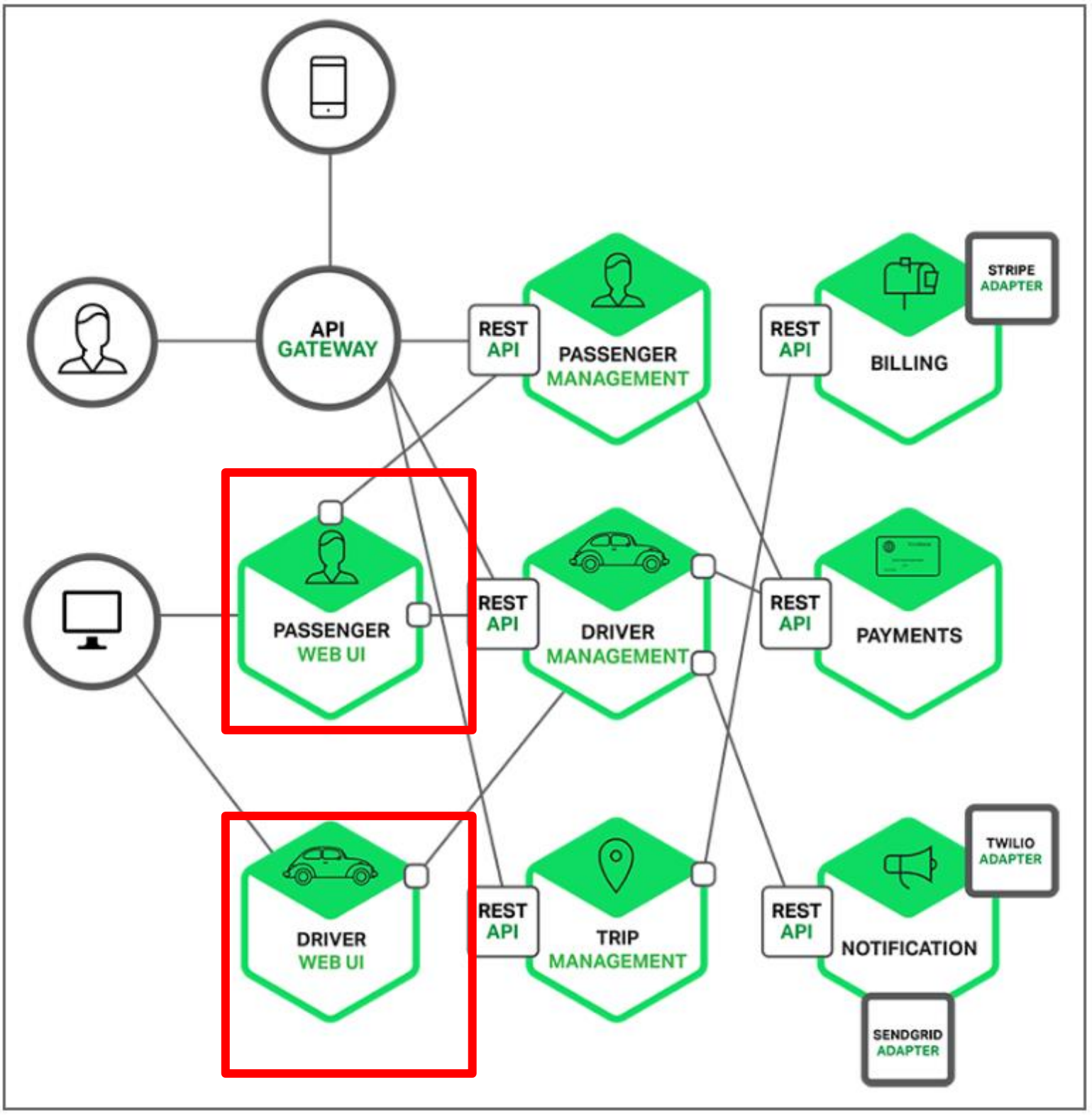
Uber competitor microservice architecture

Pros?



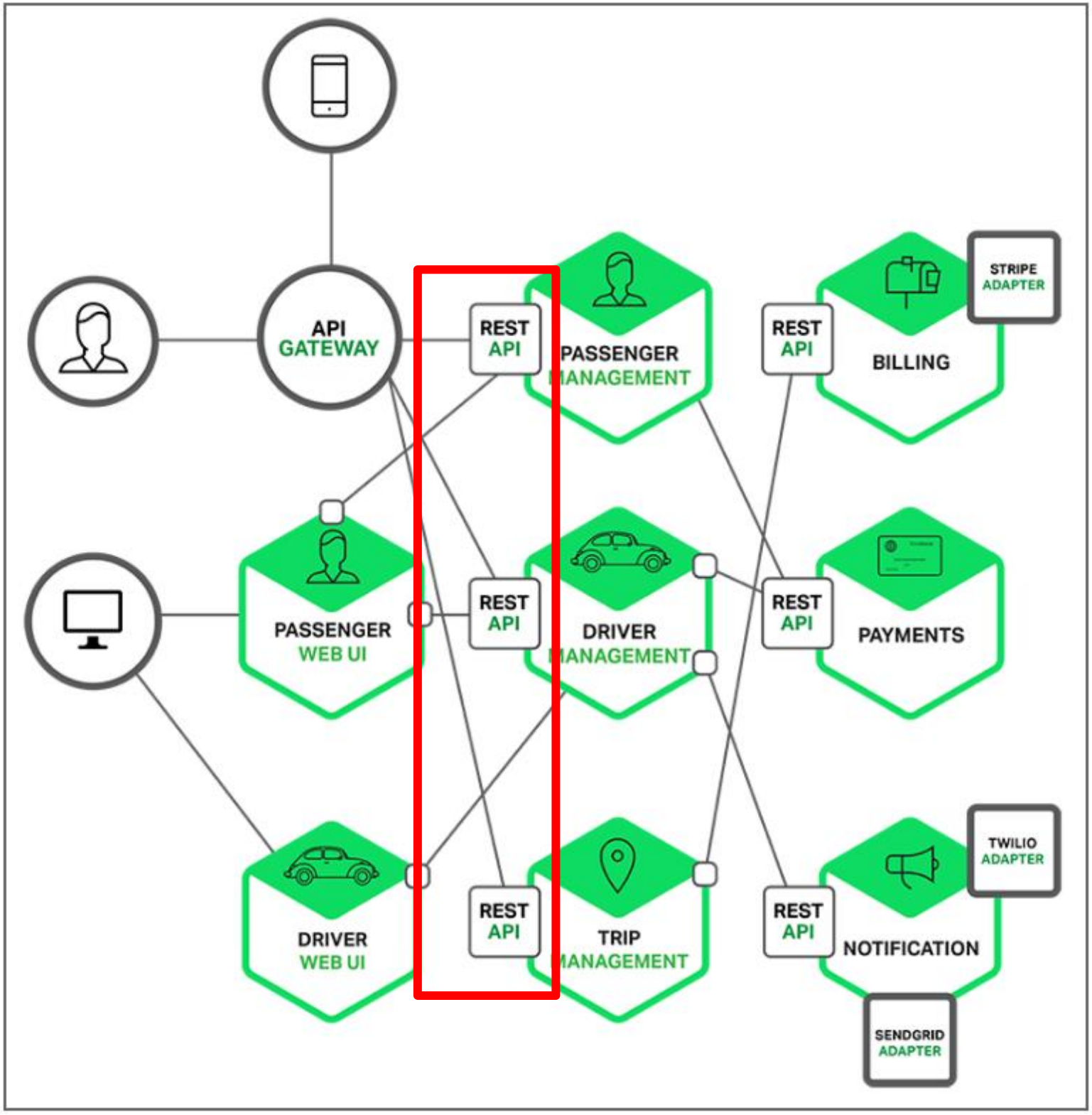
Uber competitor microservice architecture

Pros?



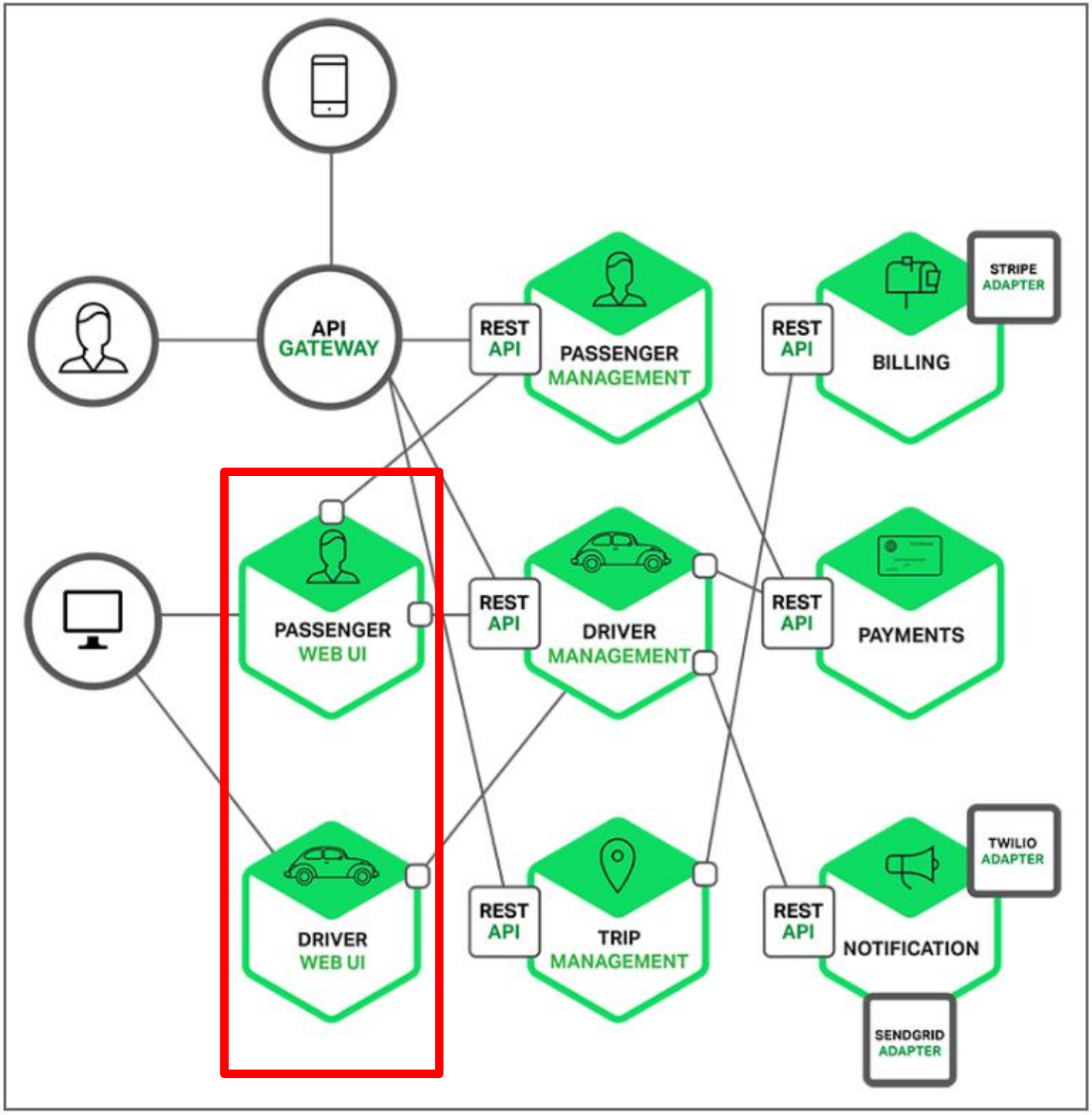
Uber competitor microservice architecture

Pros?



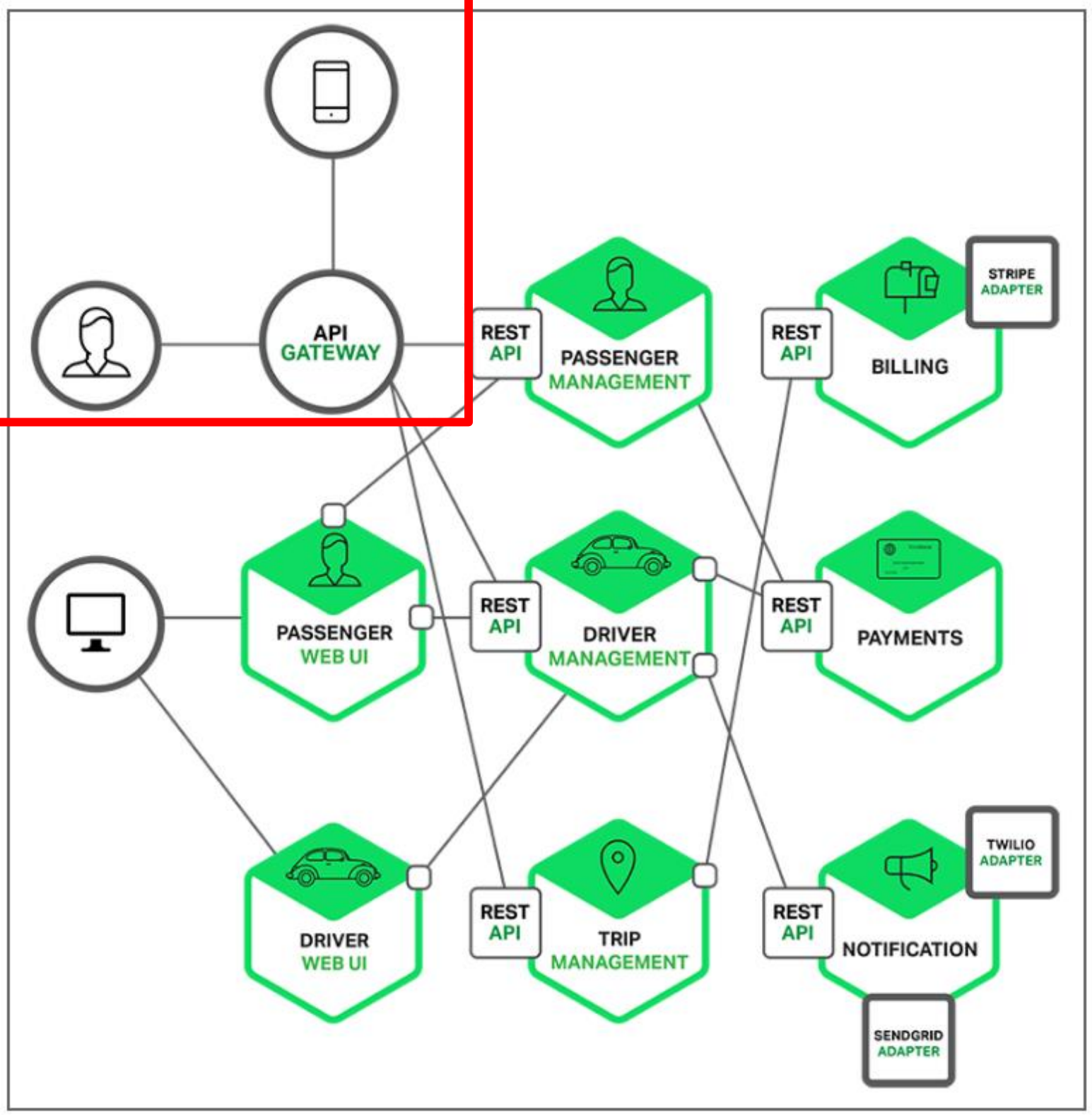
Uber competitor microservice architecture

Pros?

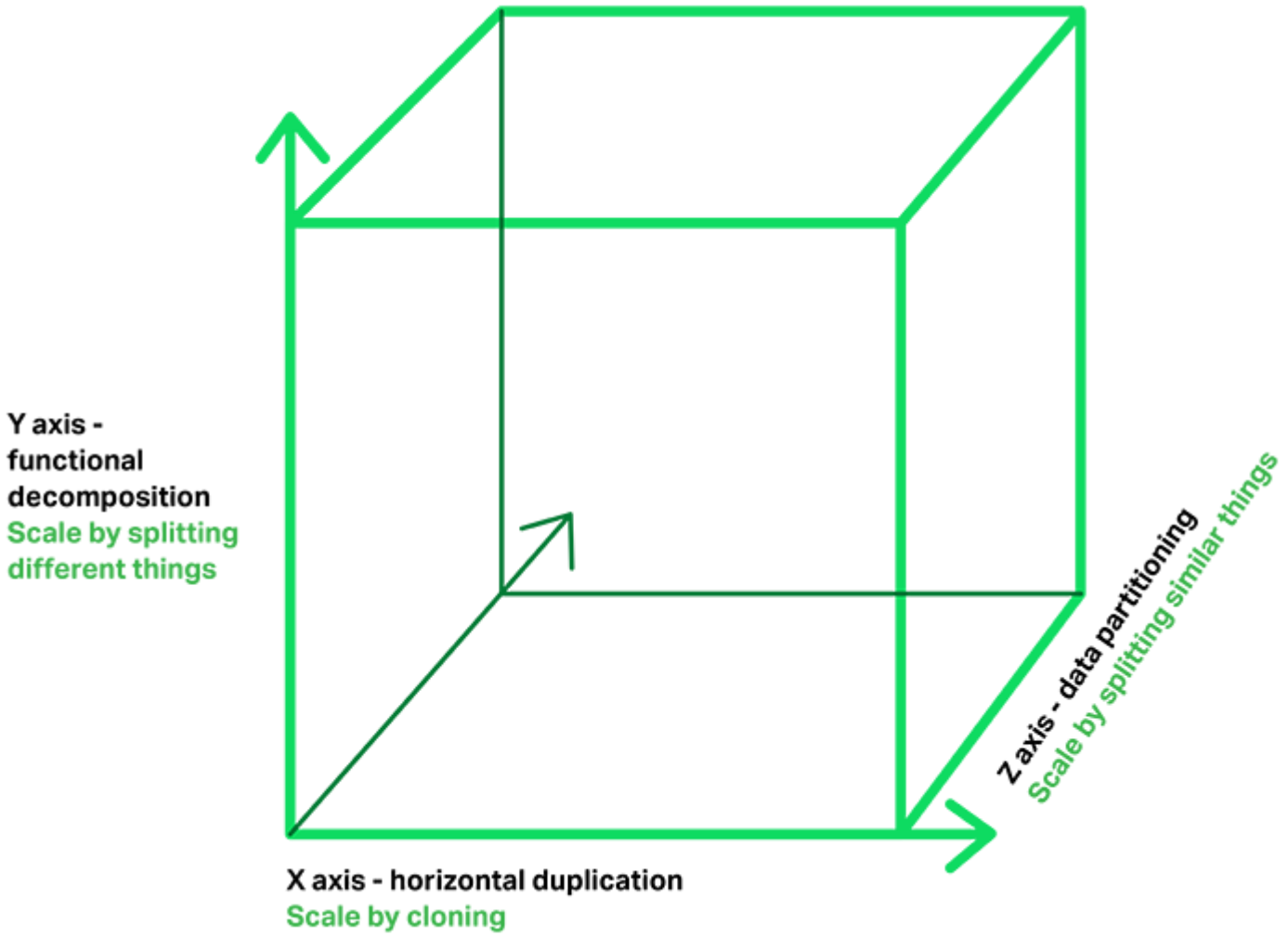


Uber competitor microservice architecture

Pros?



The Scale Cube



From “The Art of Scalability” book

Characteristics of a Microservice Architecture

- Componentization via services
- Organized around business capabilities
- Decentralized governance
- Decentralized data management
- Design for failure

15

Characteristics of a Microservice Architecture
Componentization via Services

- A **component** is a unit of software that is independently replaceable and upgradeable.
- Componentization with microservices involves breaking the software into a set of services.
- Libraries versus components?
- Such services are independently deployable.
- Effects of such componentization?
  - Redeploy rate after changes (how?)
  - More explicit interfaces (how?)

16

Characteristics of a Microservice Architecture

Organized around Business Capabilities

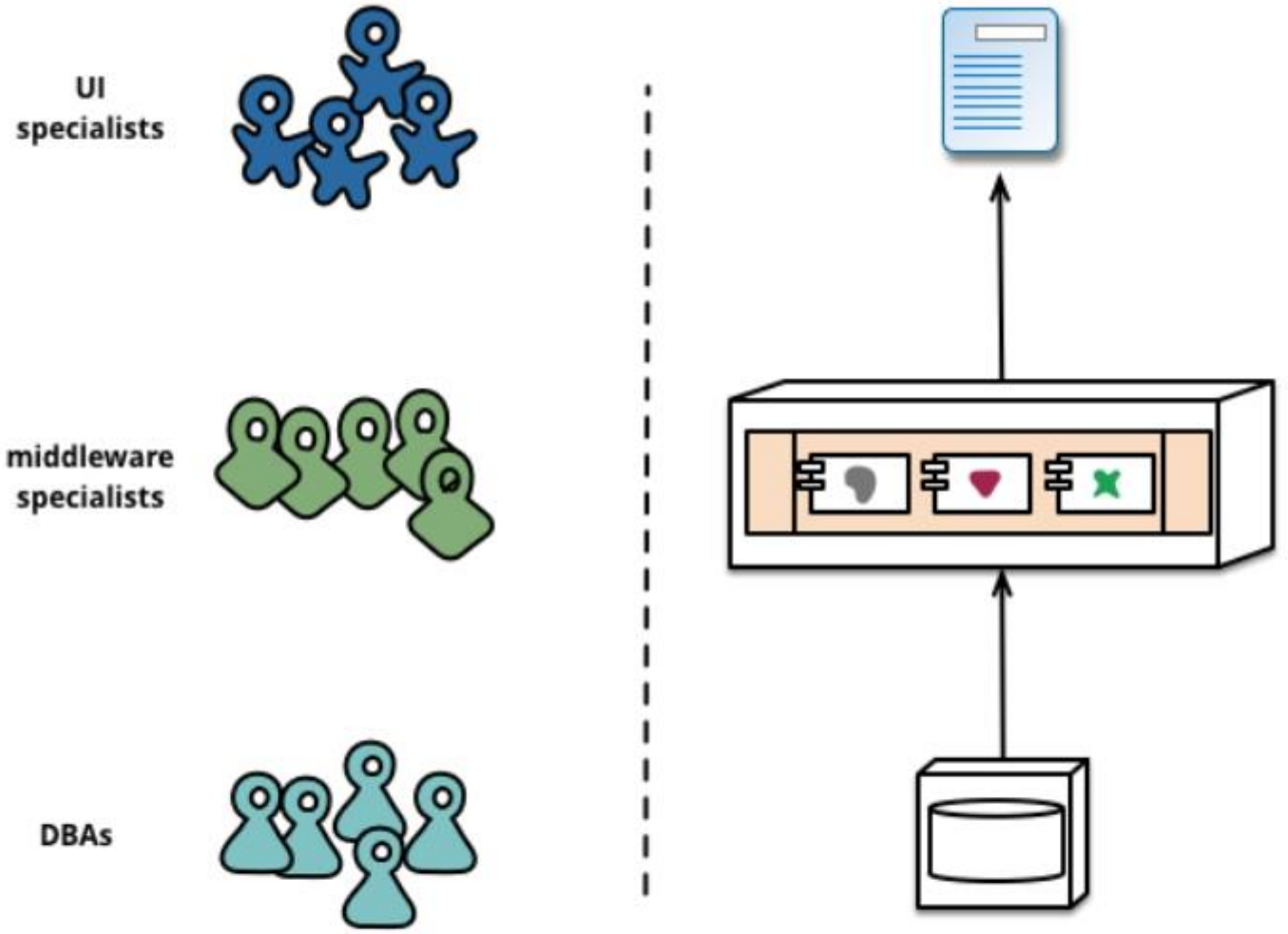


Figure 2: Conway's Law in action

17

Characteristics of a Microservice Architecture

Organized around Business Capabilities

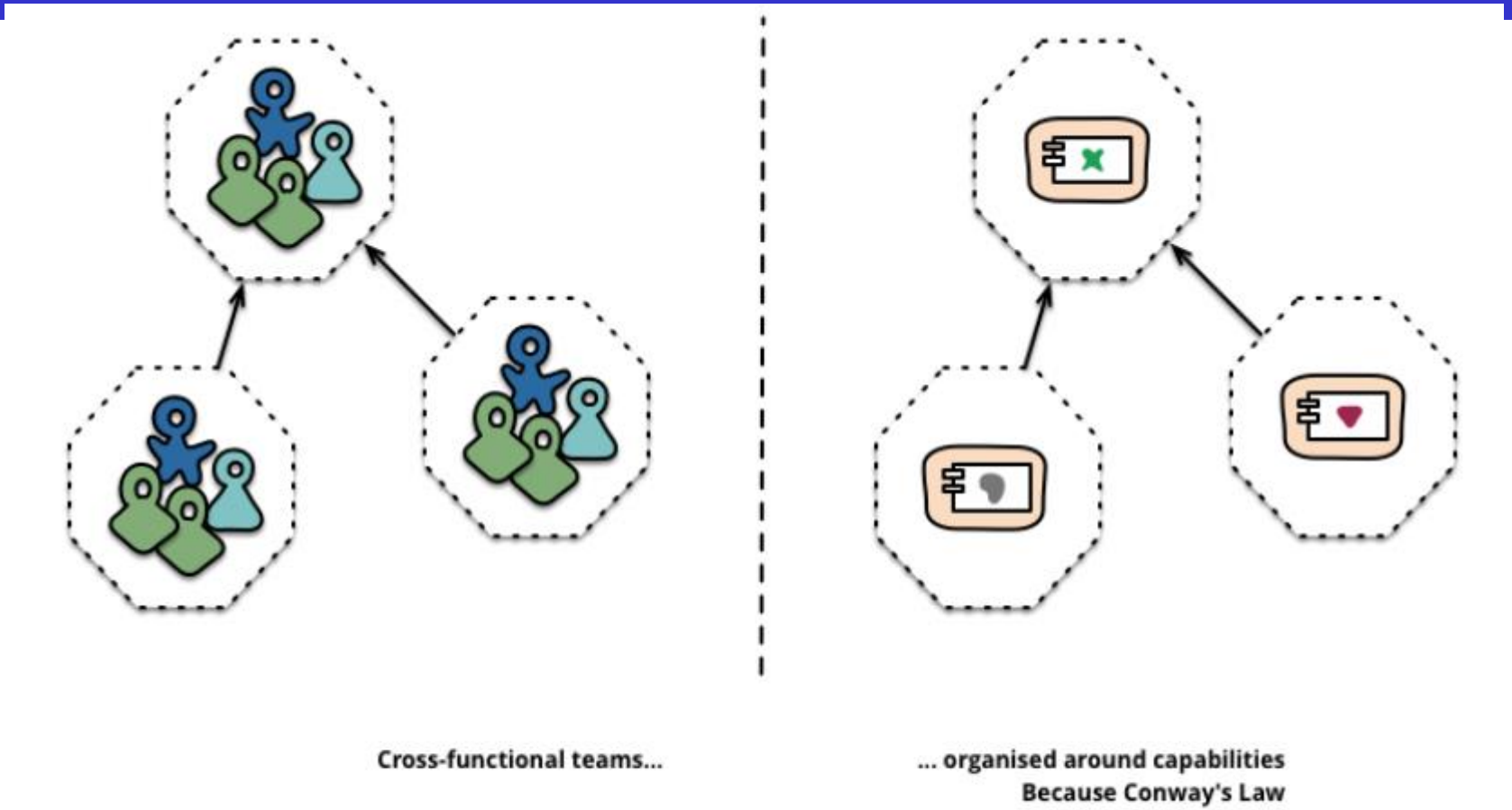


Figure 3: Service boundaries reinforced by team boundaries

Characteristics of a Microservice Architecture
Decentralized Governance

- One of the consequences of centralised governance is the tendency to standardise on single technology platforms.
- Splitting the monolith's components out into services we have a choice when building each of them.

19

Characteristics of a Microservice Architecture
Decentralized Data Management

- Decentralization of data management is present in a number of different ways.
- Consider a large enterprise application that is being built.
- The sales perspective of the customer would differ from the support perspective of the application, hence resulting in different views.
- Such views could have different attributes and common attributes (more problematic) with different semantics. (How?)
- This issue is common between applications, but can also occur *within* applications, particular when that application is divided into separate components.

20

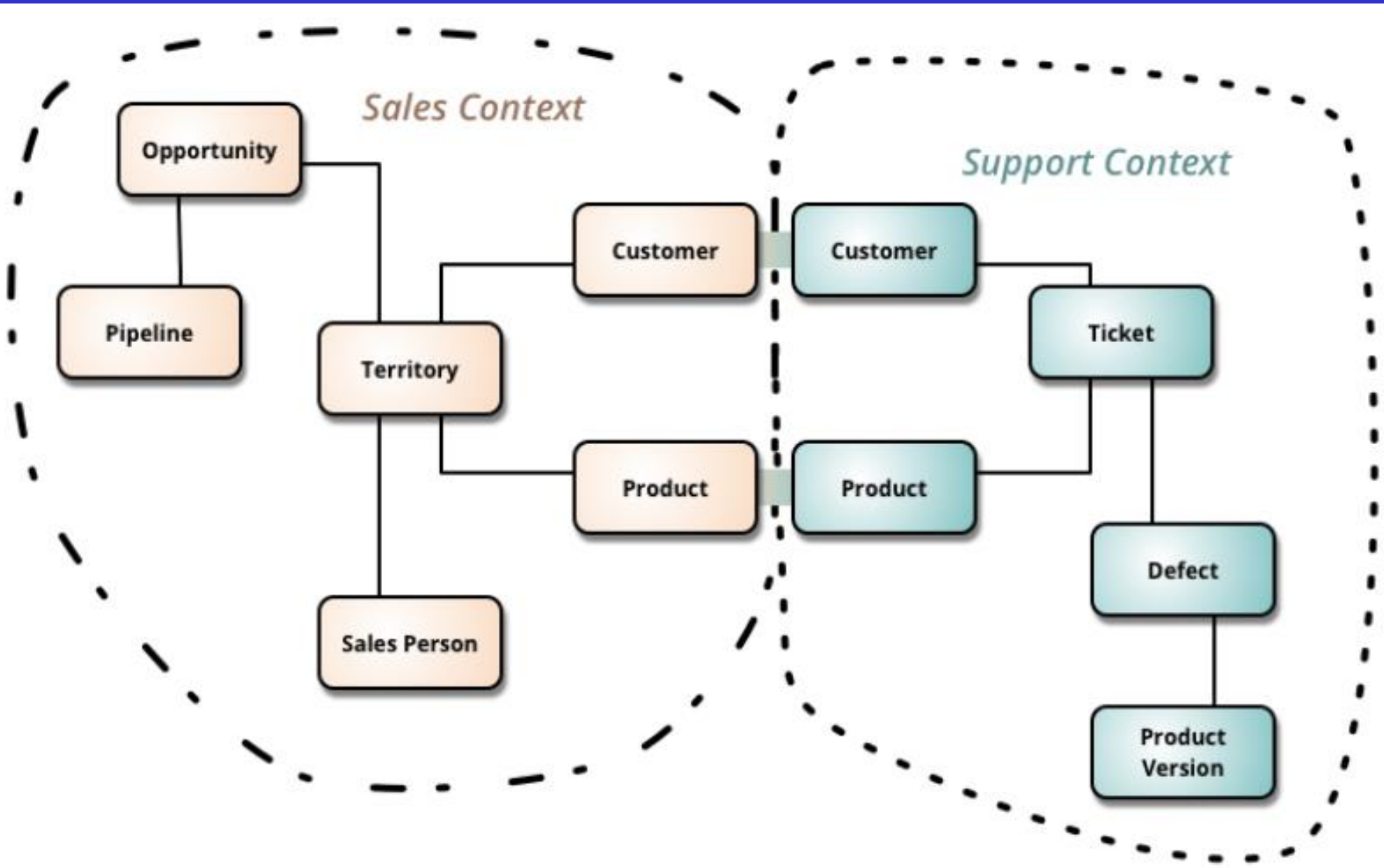
Characteristics of a Microservice Architecture
Decentralized Data Management

- This brings the notion of Domain Driven Design and Bounded Context.
- *Domain Driven Design* (DDD) is a software design technique that focuses on creating *bounded contexts* to understand the natural boundaries between business domains.
- The features and functionalities within a bounded context have high *cohesion*. This keeps similar functions that need similar data close together.
- Bounded contexts only interact with each other in well-defined ways—they use APIs.

21

Characteristics of a Microservice Architecture

Decentralized Data Management

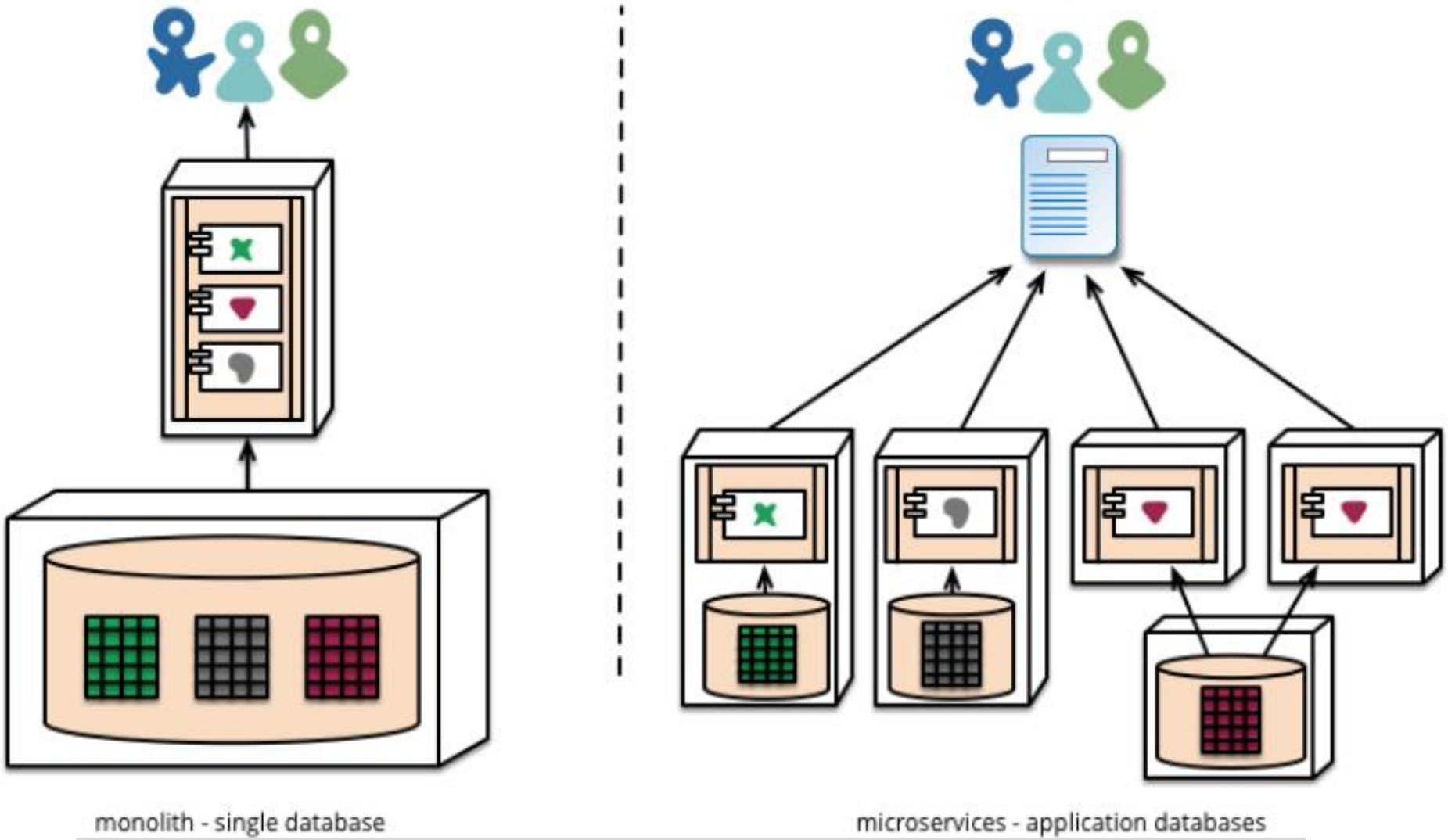


Characteristics of a Microservice Architecture
Decentralized Data Management

- As well as decentralizing decisions about conceptual models, microservices also decentralize data storage decisions.
- Microservices prefer letting each service manage its own database, either different instances of the same database technology, or entirely different database systems.

Characteristics of a Microservice Architecture

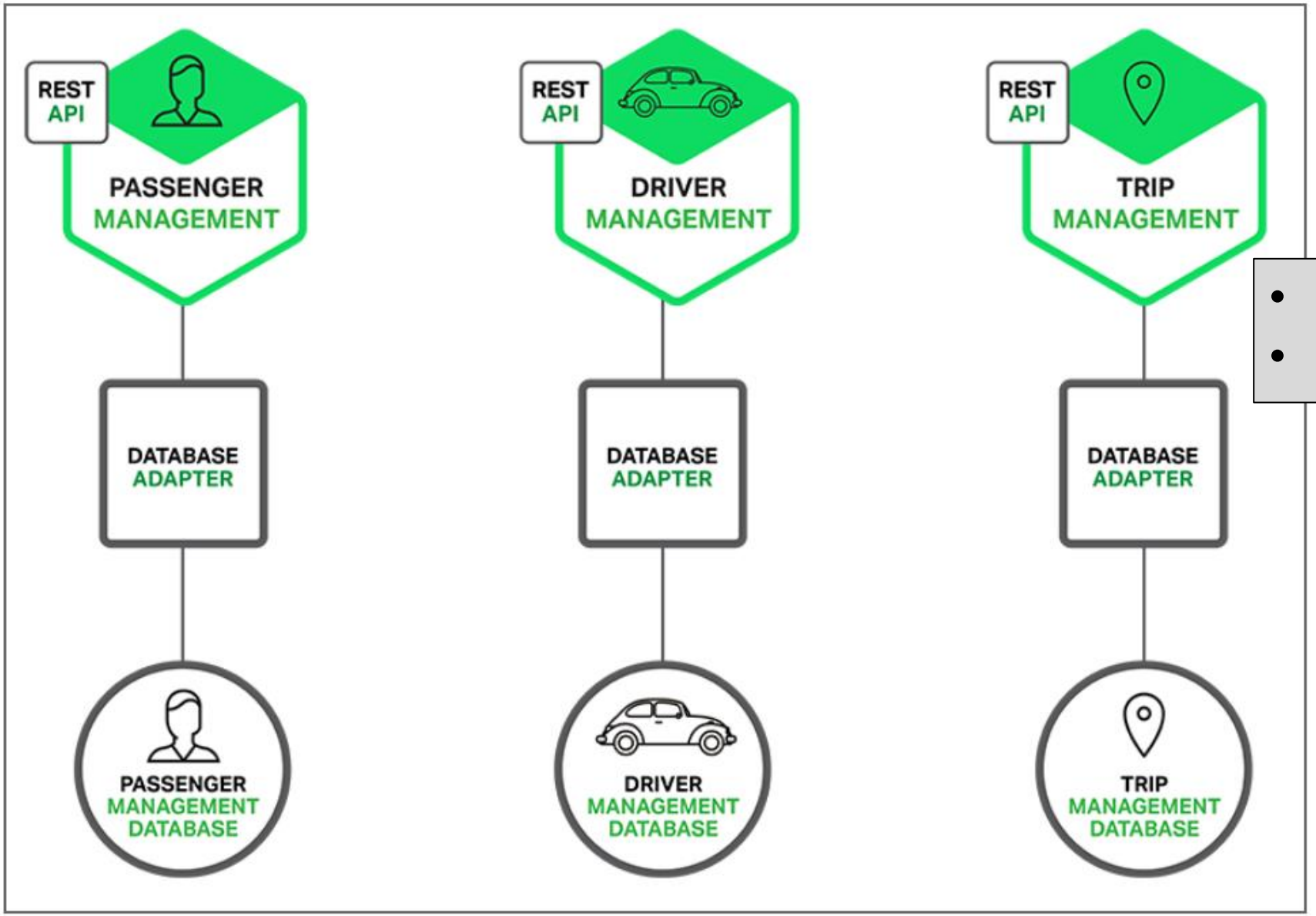
Decentralized Data Management



Issues?

Characteristics of a Microservice Architecture

Decentralized Data Management



- Coupling?
- Duplication?

Characteristics of a Microservice Architecture
Design for Failure

- A consequence of using services as components, is that applications need to be designed so that they can tolerate the failure of services.
- Any service call could fail due to unavailability of the supplier, the client has to respond to this as gracefully as possible.
- This is a disadvantage compared to a monolithic design (why?)
- The consequence is that microservice teams constantly reflect on how service failures affect the user experience.

26

Characteristics of a Microservice Architecture
Design for Failure

- Since services can fail at any time, it's important to be able to detect the failures quickly and, if possible, automatically restore service.
- Microservice applications put a lot of emphasis on real-time monitoring of the application.
  - What can be monitored?

27

Required Readings

- https://martinfowler.com/articles/microservice-trade-offs.html
- https://martinfowler.com/articles/microservices.html
- https://www.nginx.com/blog/introduction-to-microservices/
- https://www.nginx.com/blog/building-microservices-using-an-api-gateway/
- https://www.nginx.com/blog/building-microservices-inter-process-communication/