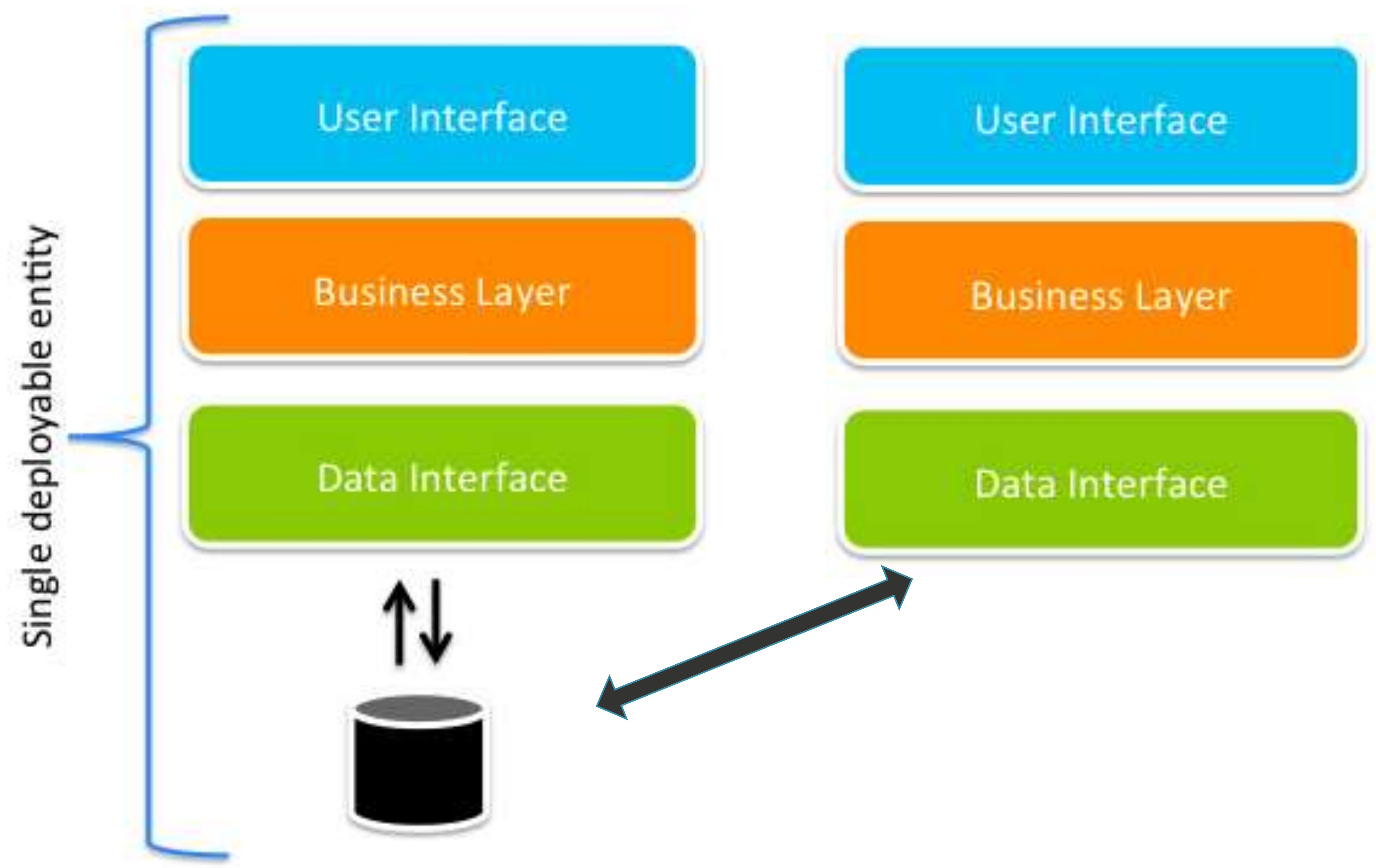
Microservices

Let’s talk about:

- Monolithic architecture and microservices
- Pros and cons. What to choose?
- Microservices architecture patterns

Monolithic Architecture

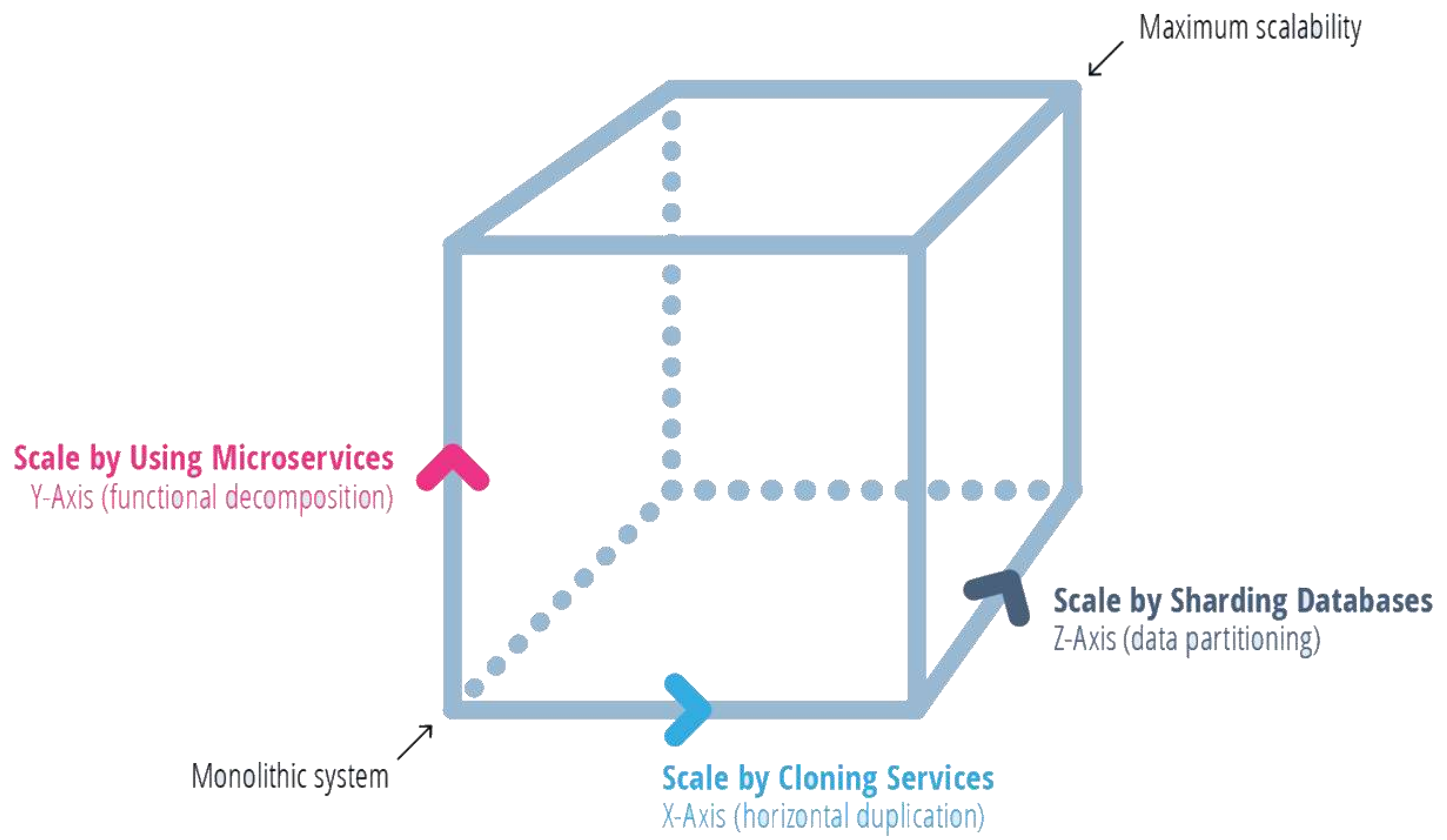


Issues

- System complexity increases
- Support is getting more complex
- No Tests or Testing is limited
- Bugs
- Tech stack becomes outdated
- Release/Testing process

MAGLEBY MAERSK

The Scale Cube and Microservices: 3 Dimensions to Scaling

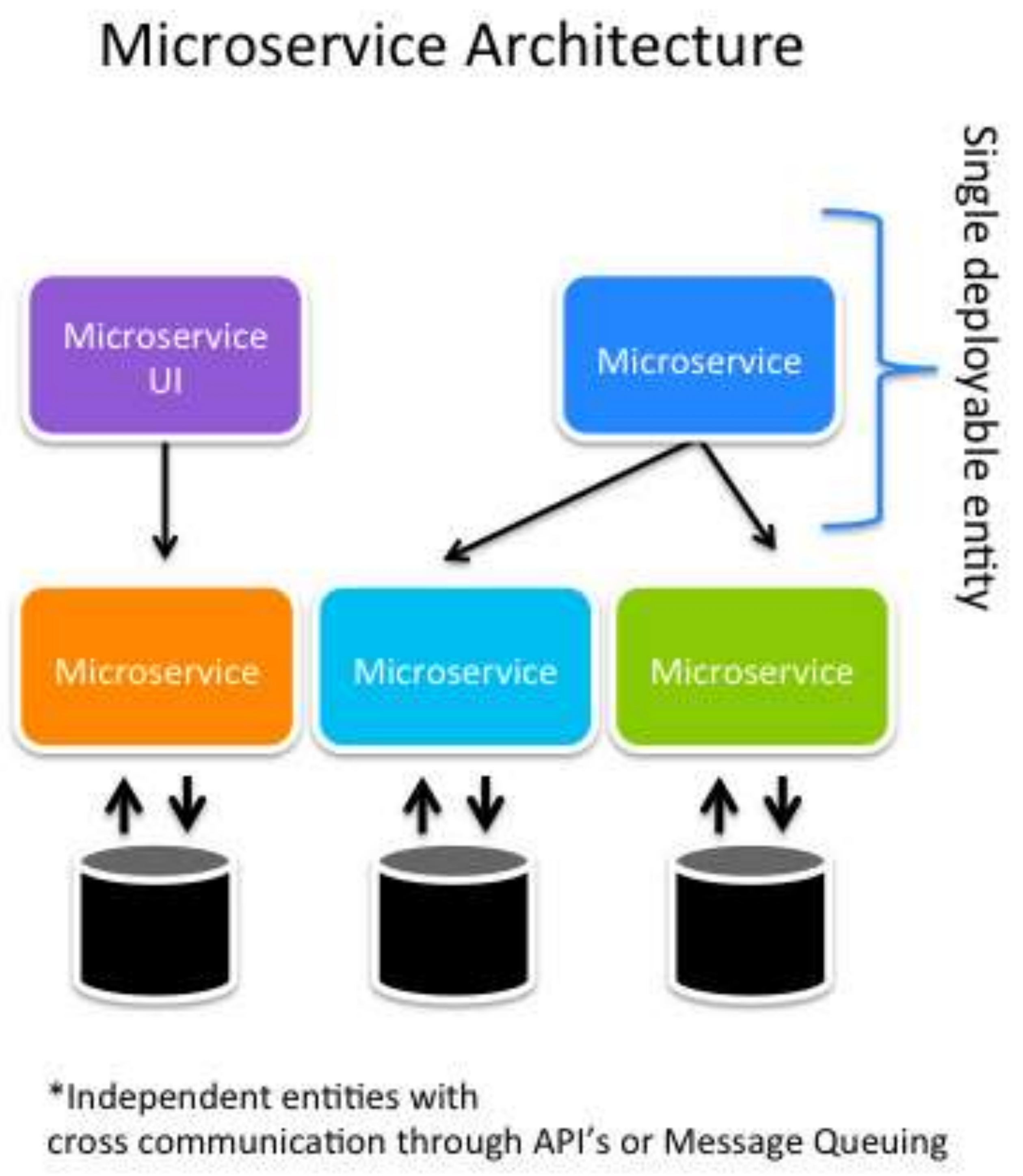

Maximum scalability
Scale by Using Microservices
Y-Axis (functional decomposition)
Scale by Sharding Databases
Z-Axis (data partitioning)
Monolithic system
Scale by Cloning Services
X-Axis (horizontal duplication)

WHEN YOUR BOSS TELLS YOU
WE'RE CONVERTING TO MICROSERVICES

Microservices

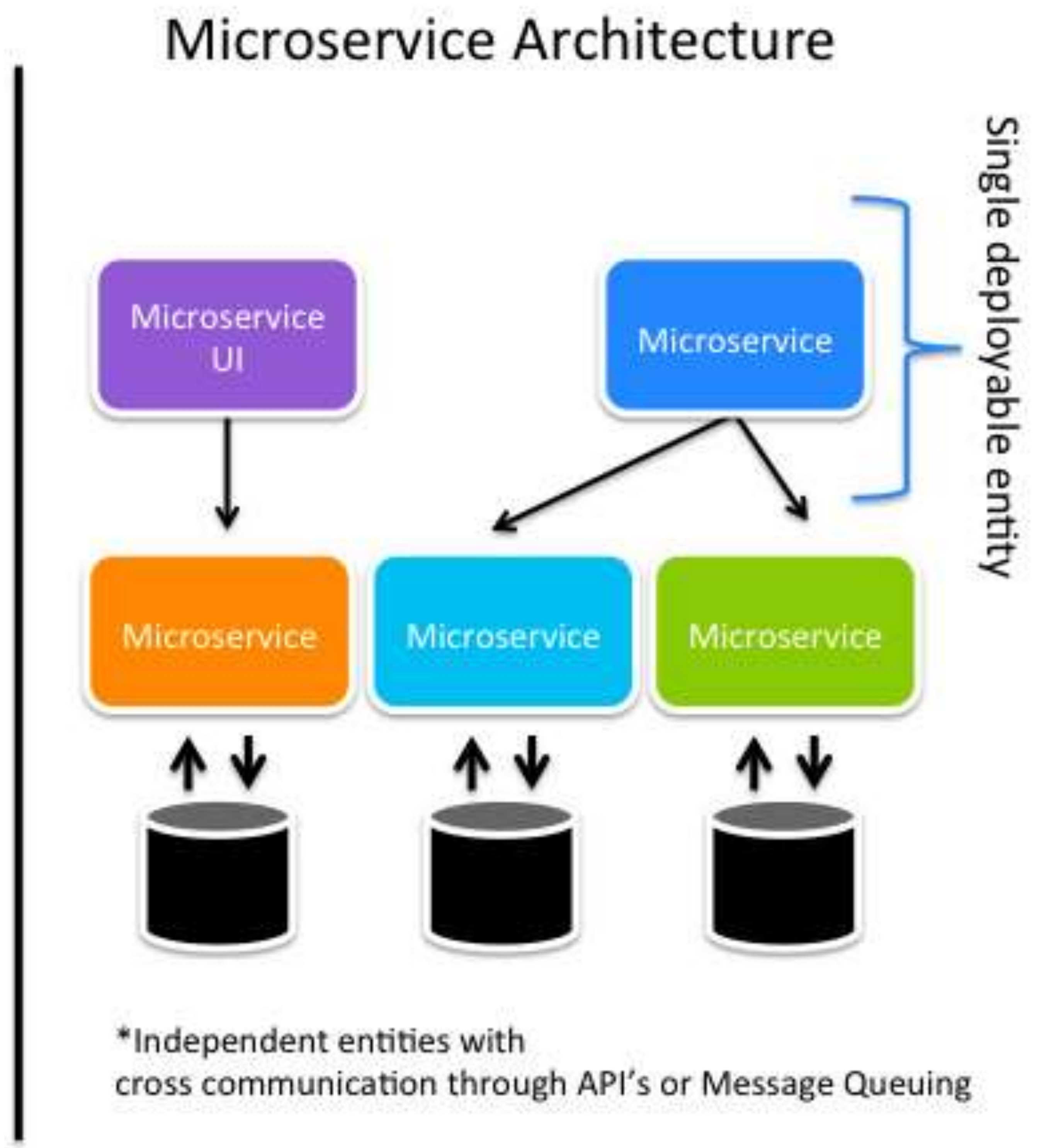
Architecture pattern in which services are:

- Small
- Focused
- Loosely coupled

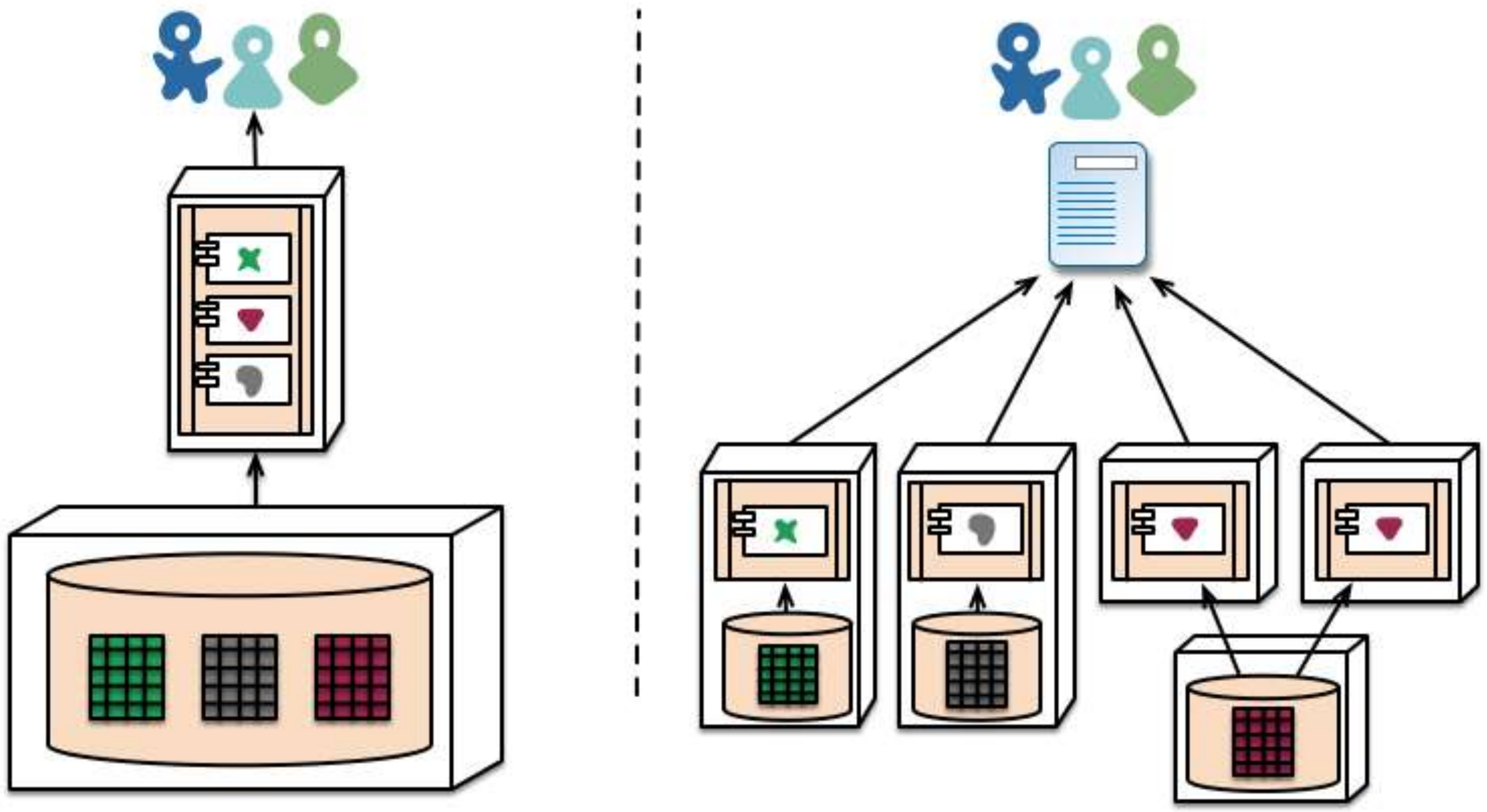

Microservice Architecture
Microservice UI
Microservice
Single deployable entity
Microservice
Microservice
Microservice
*Independent entities with
cross communication through API's or Message Queuing

Main Characteristics

- Componentization, the ability to replace parts of a system, comparing with stereo components where each piece can be replaced independently from the others.
- Organisation around business capabilities instead of around technology.
- Smart endpoints and dumb pipes
- Decentralized data management with one database for each service instead of one database for a whole company.
- Infrastructure automation with continuous delivery being mandatory.
- Design for failure

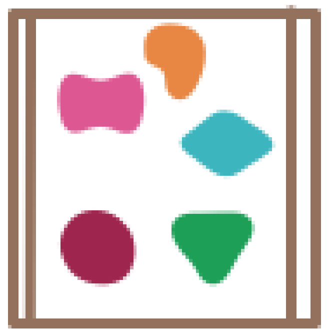

Microservice Architecture
Microservice UI
Microservice
Single deployable entity
Microservice
Microservice
Microservice
*Independent entities with
cross communication through API's or Message Queuing

Monolithic vs Microservices

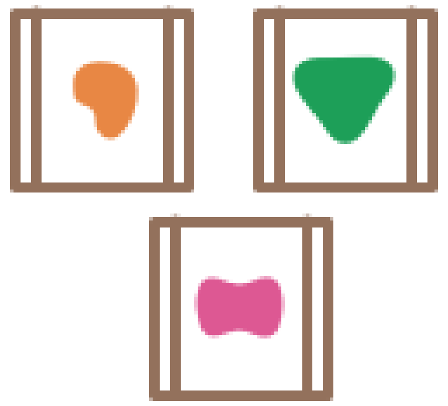

monolith - single database
microservices - application databases

Monolithic vs Microservices. Scaling

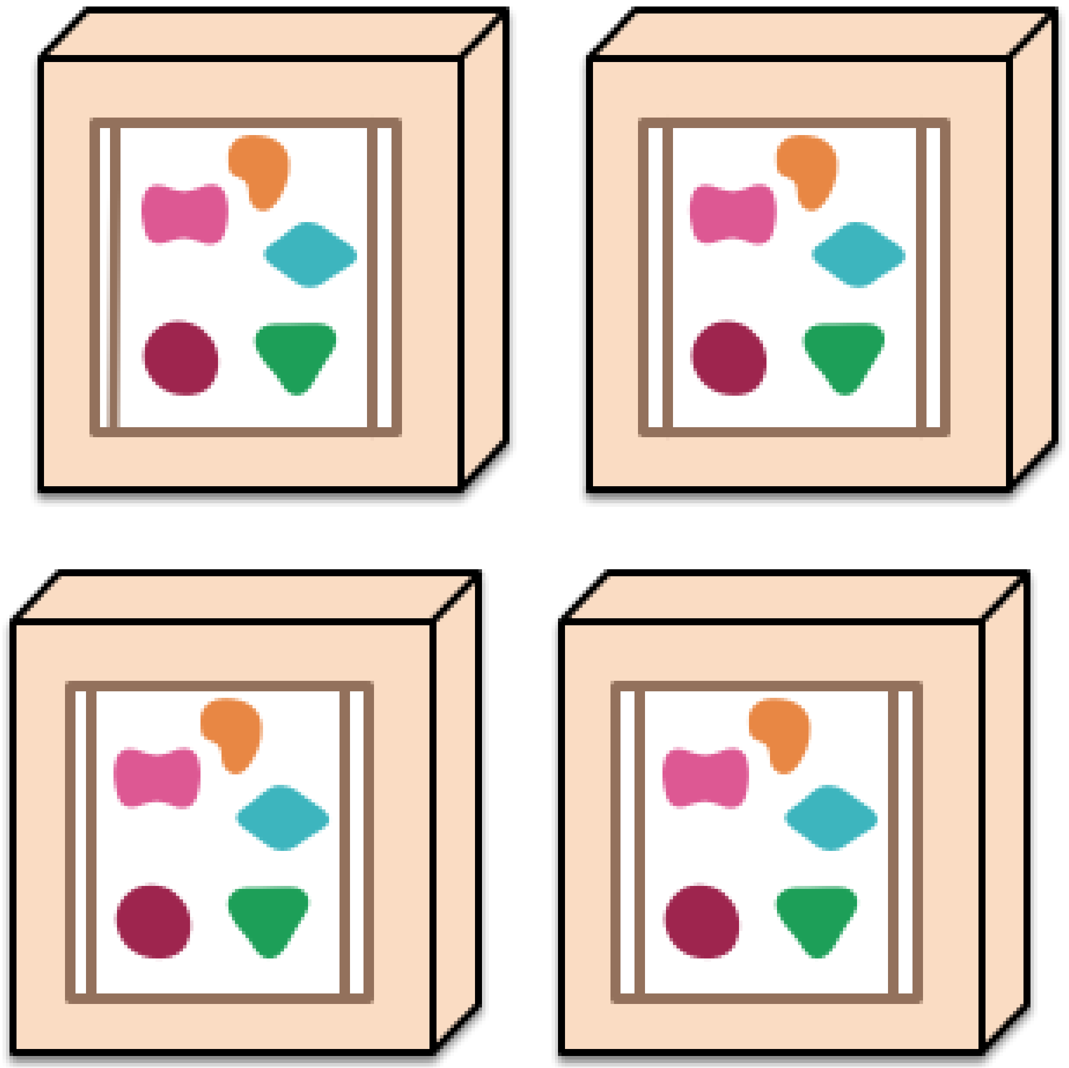
A monolithic application puts all its functionality into a single process...



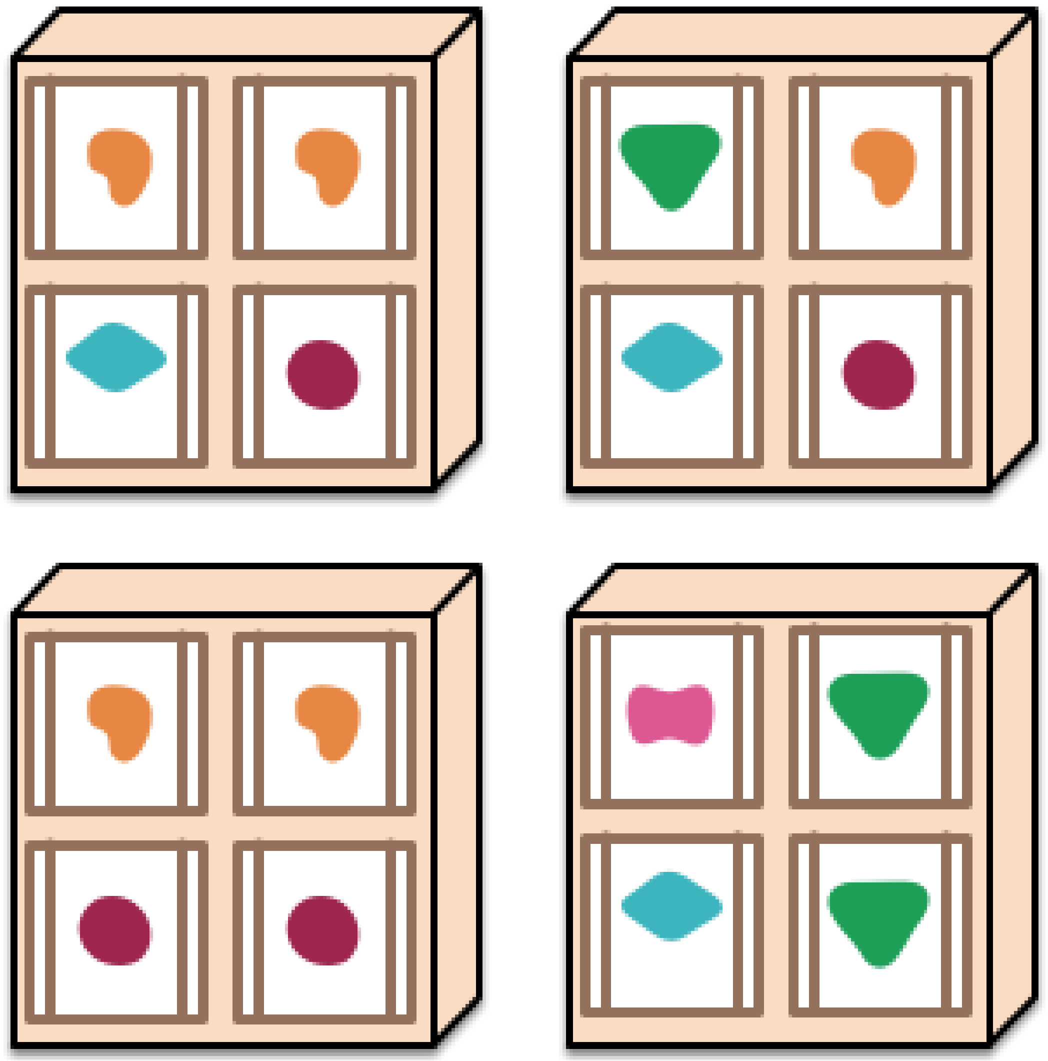
A microservices architecture puts each element of functionality into a separate service...



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



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



Microservices are not a silver bullet

Benefits

- Enables the continuous delivery and deployment of large, complex applications.
- Each microservice is relatively small
- Improved fault isolation. For example, if there is a memory leak in one service then only that service will be affected. The other services will continue to handle requests. In comparison, one misbehaving component of a monolithic architecture can bring down the entire system.
- Eliminates any long-term commitment to a technology stack. When developing a new service you can pick a new technology stack. Similarly, when making major changes to an existing service you can rewrite it using a new technology stack.

Microservices are not a silver bullet

Drawbacks

- Developers must deal with the additional complexity of creating a distributed system.
- Deployment complexity. In production, there is also the operational complexity of deploying and managing a system comprised of many different service types.
- Increased resources consumption. The microservice architecture replaces N monolithic application instances with NxM services instances.

What should I choose?

Monolithic

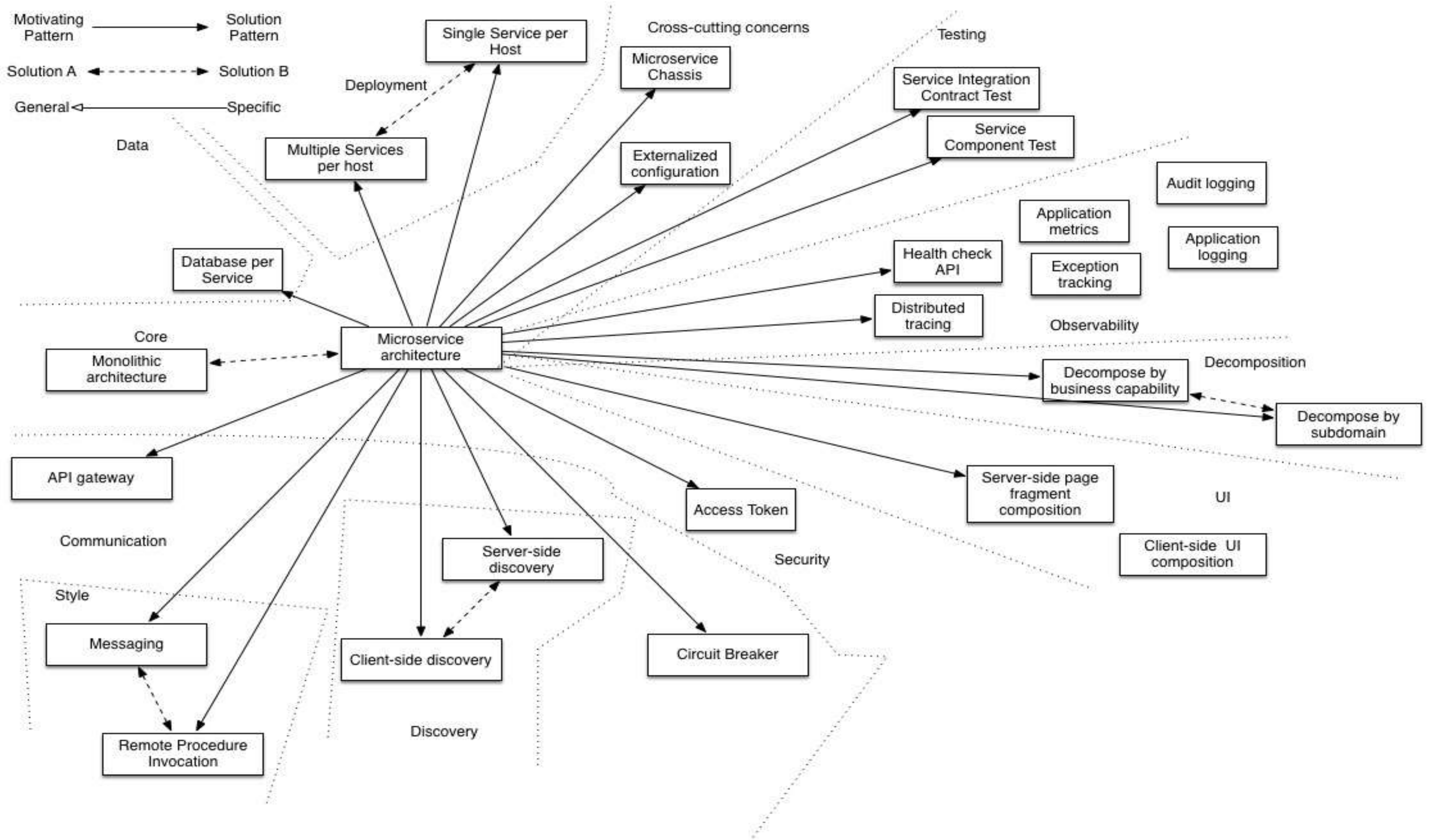
- New business domain, lack of knowledge
- Proof of Concepts
- Lack of qualification
- Fast or Throw-away solutions
- Low budget

What should I choose?

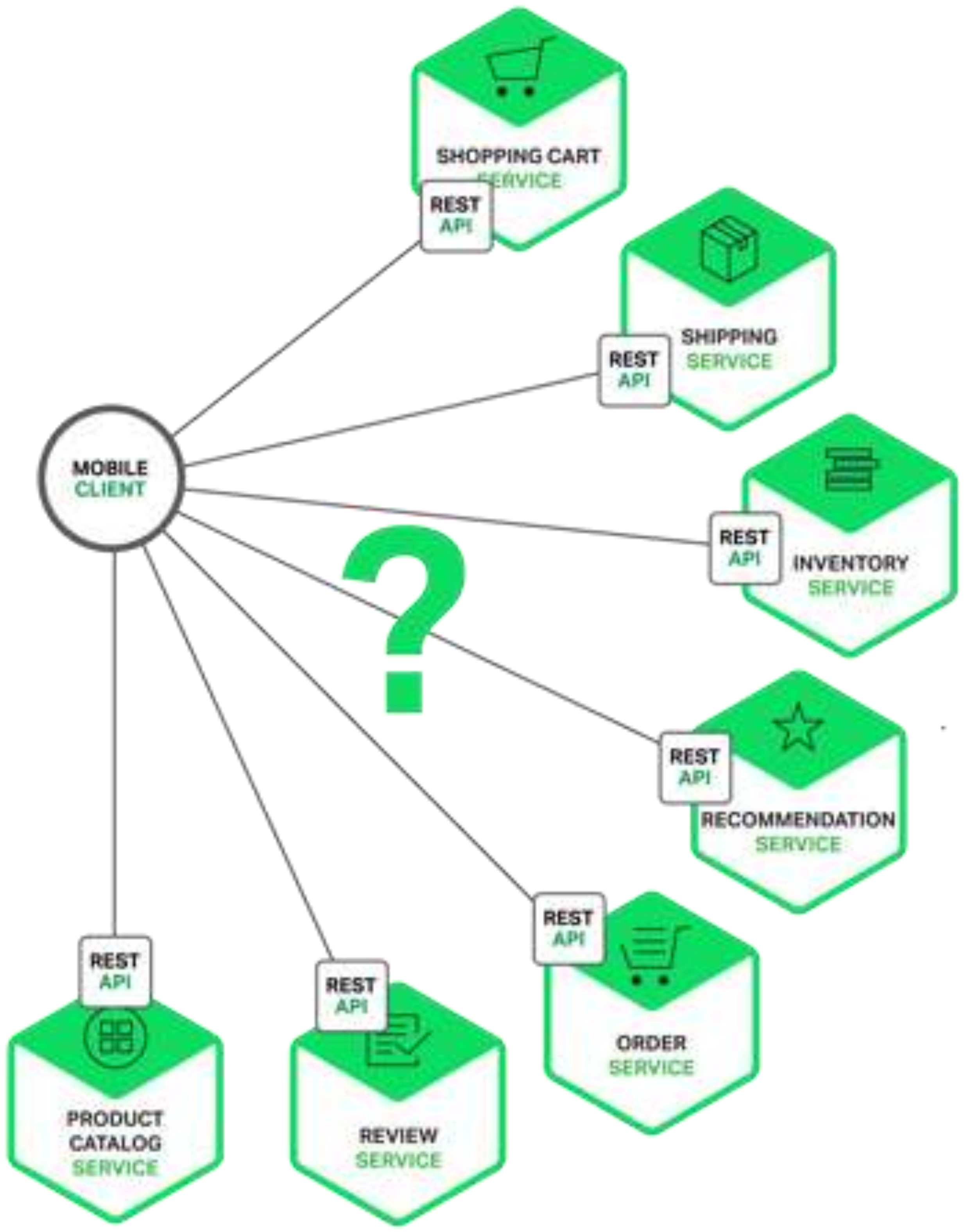
Microservices

- Needs to scale
- You understand business domain
- Big budget
- Ready to invest into infrastructure and CI/CD processes
- Experienced and highly qualified team

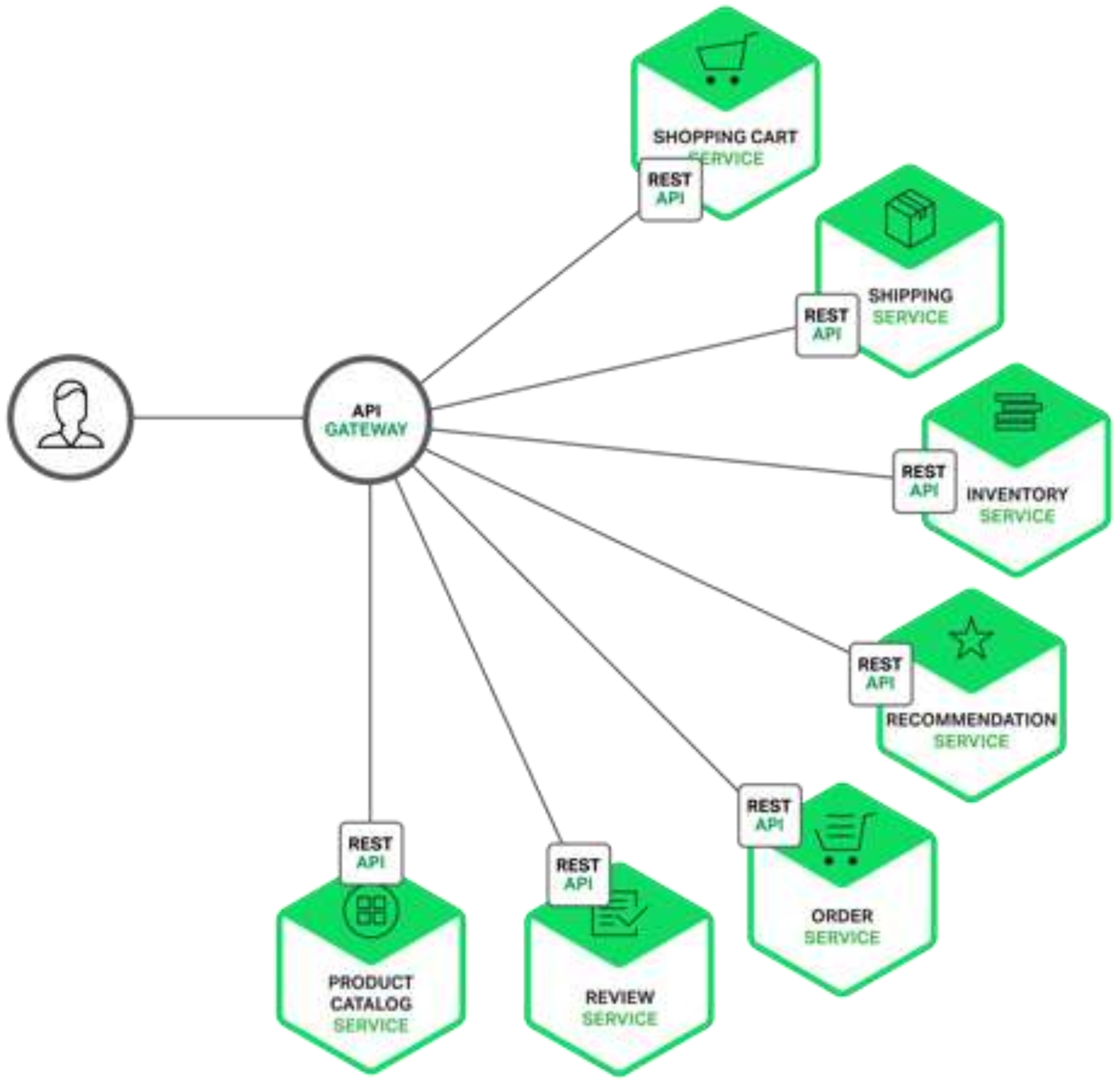
Architecture Patterns



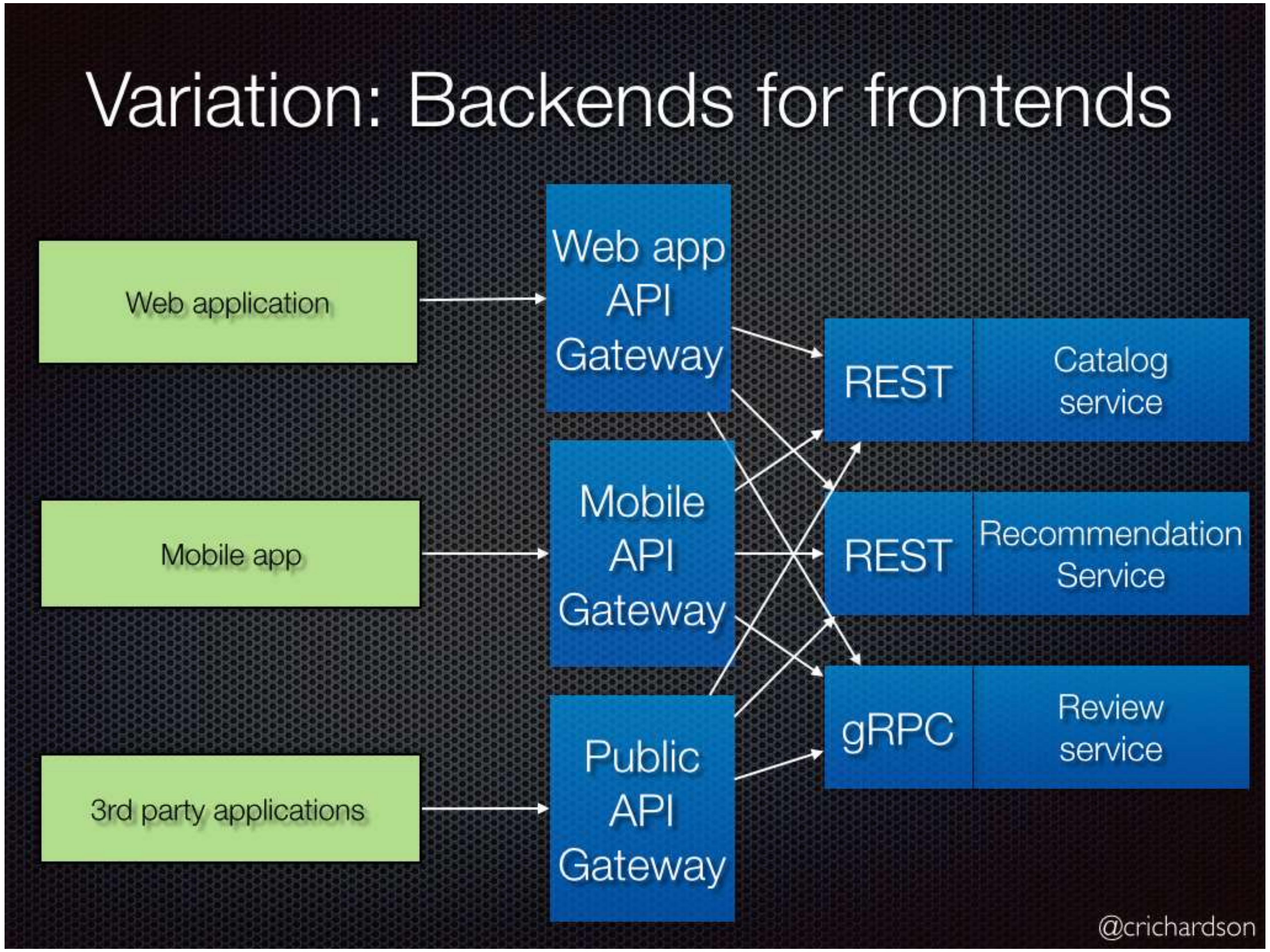
Client Configuration



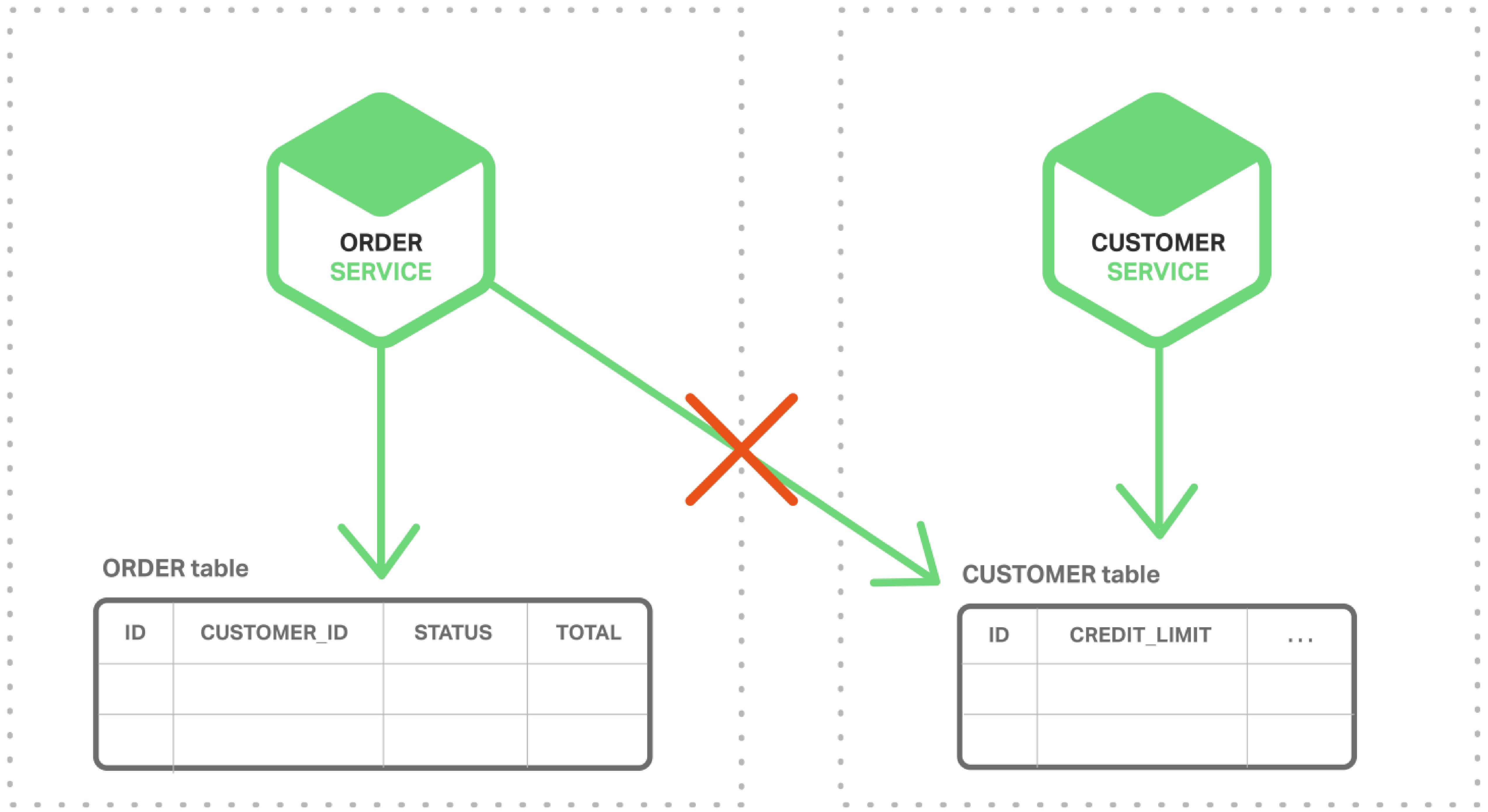
API Gateway


API
GATEWAY
SHOPPING CART
SERVICE
SHIPPING
SERVICE
INVENTORY
SERVICE
RECOMMENDATION
SERVICE
ORDER
SERVICE
REVIEW
SERVICE
PRODUCT
CATALOG
SERVICE
REST
API

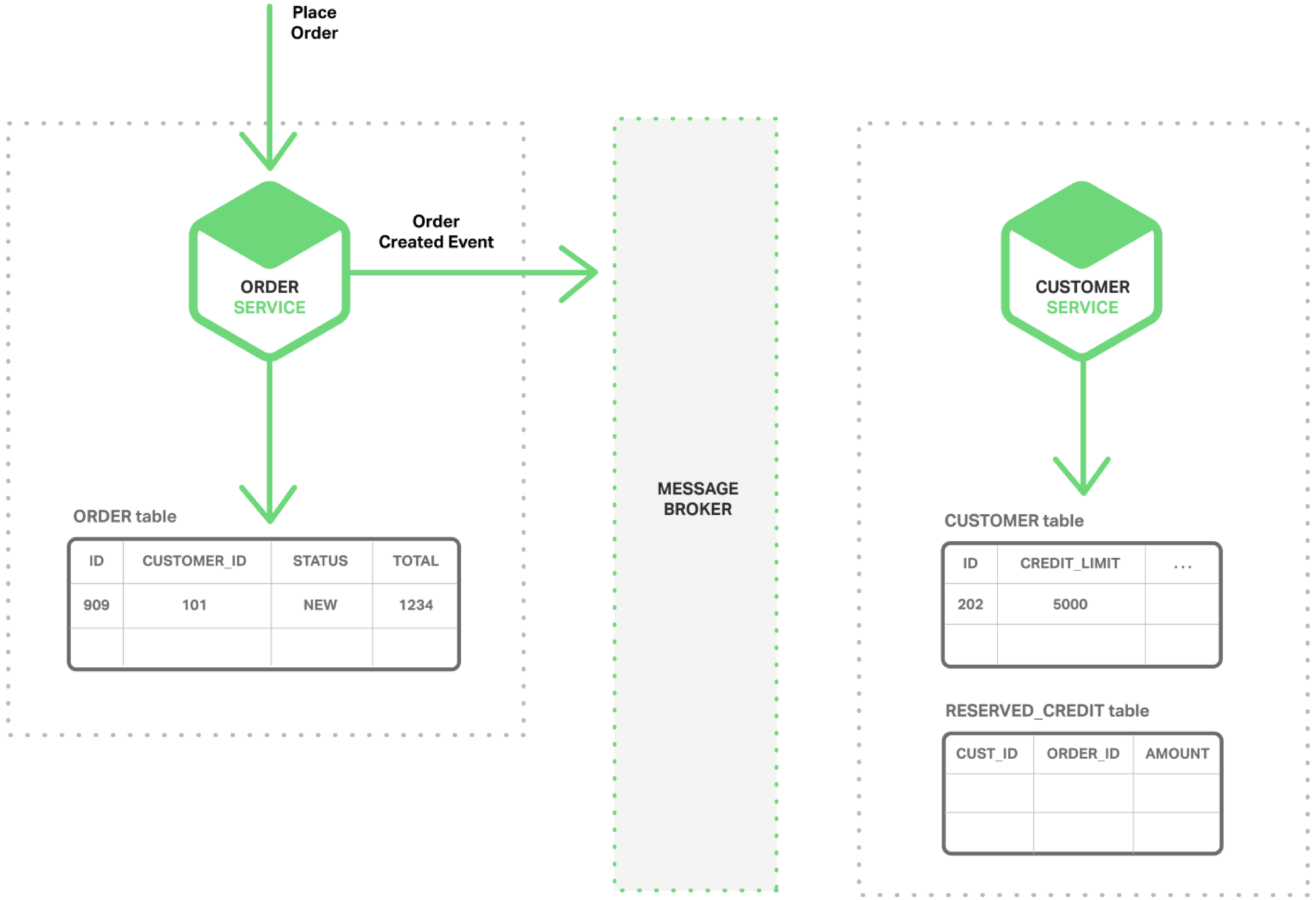
Backend for Frontend



Communication between services



Event Driven (Messaging)



ORDER table

ID	CUSTOMER_ID	STATUS	TOTAL
909	101	NEW	1234

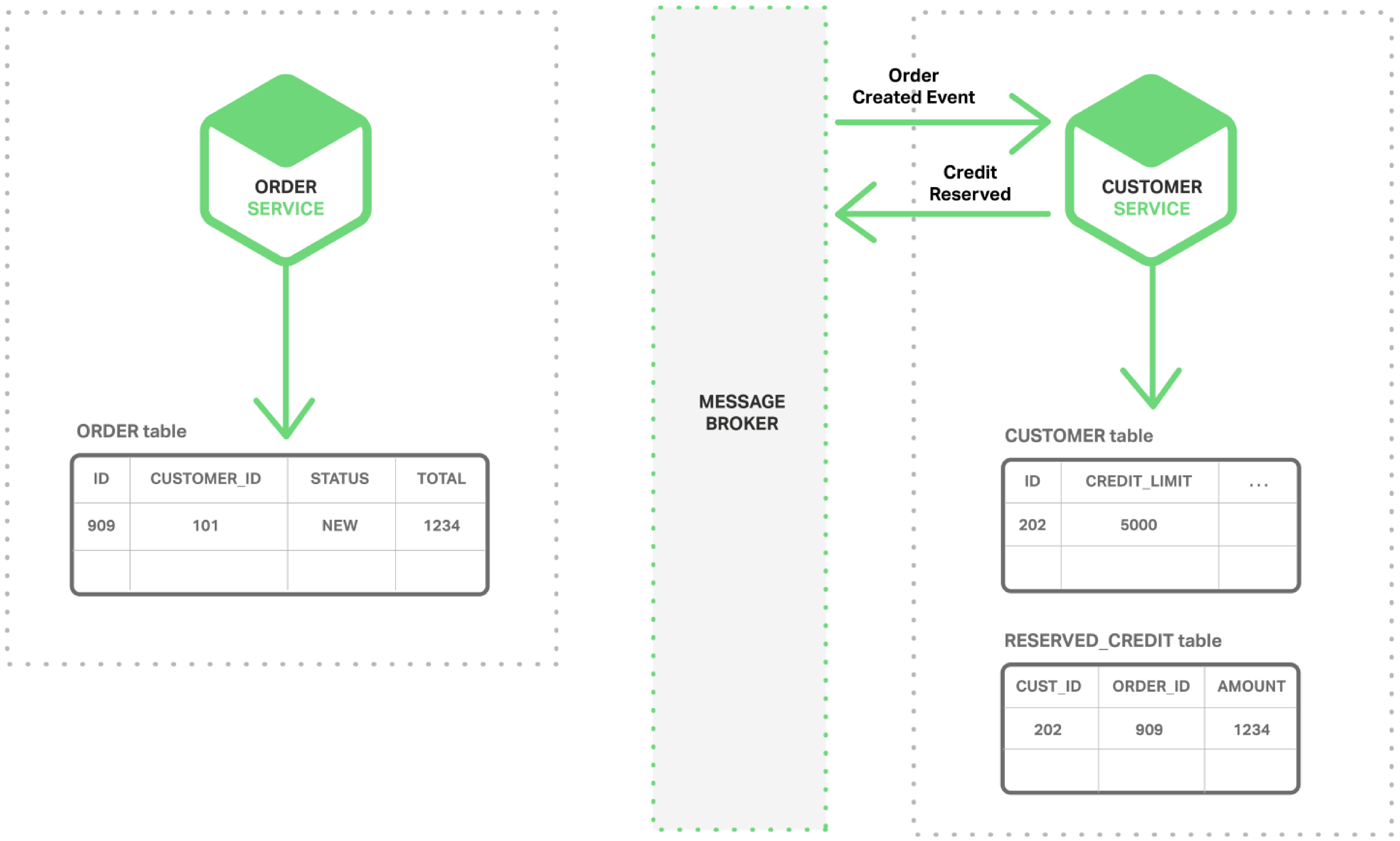
CUSTOMER table

ID	CREDIT_LIMIT	...
202	5000	

RESERVED_CREDIT table

CUST_ID	ORDER_ID	AMOUNT

Event Driven (Messaging)



ORDER table

ID	CUSTOMER_ID	STATUS	TOTAL
909	101	NEW	1234

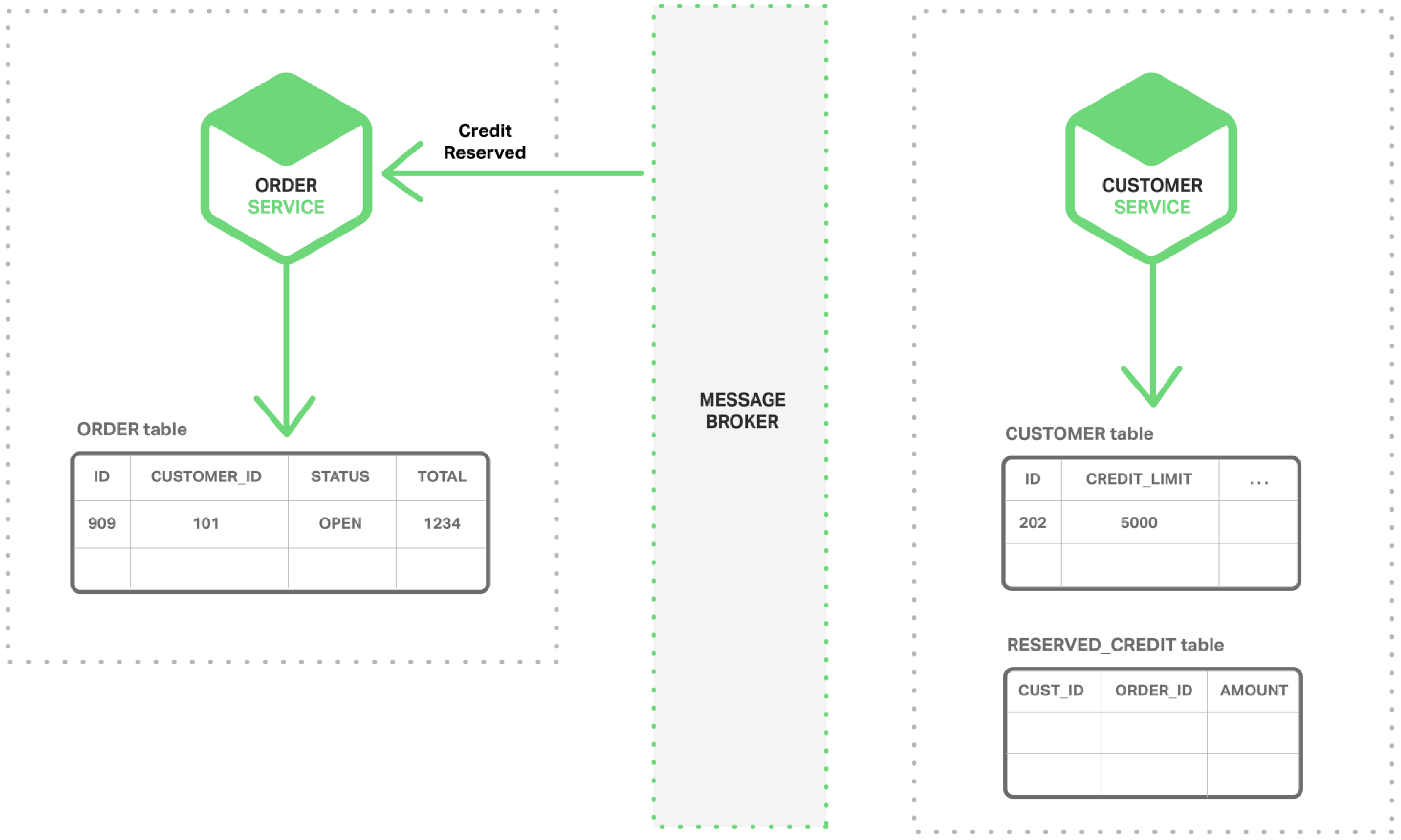
CUSTOMER table

ID	CREDIT_LIMIT	...
202	5000	

RESERVED_CREDIT table

CUST_ID	ORDER_ID	AMOUNT
202	909	1234

Event Driven (Messaging)



ORDER table

ID	CUSTOMER_ID	STATUS	TOTAL
909	101	OPEN	1234

CUSTOMER table

ID	CREDIT_LIMIT	...
202	5000	

RESERVED_CREDIT table

CUST_ID	ORDER_ID	AMOUNT

Event Driven (Messaging)

Benefits:

- Loose coupling since it decouples client from services
- Improved availability since the message broker buffers messages until the consumer is able to process them
- Can scale well
- Lot’s of market solutions

Drawbacks:

- Additional complexity of message broker, which must be highly available
- Requires qualification
- Deployment and support complexity(monitoring and tools)