

Microservices

Design Patterns

for

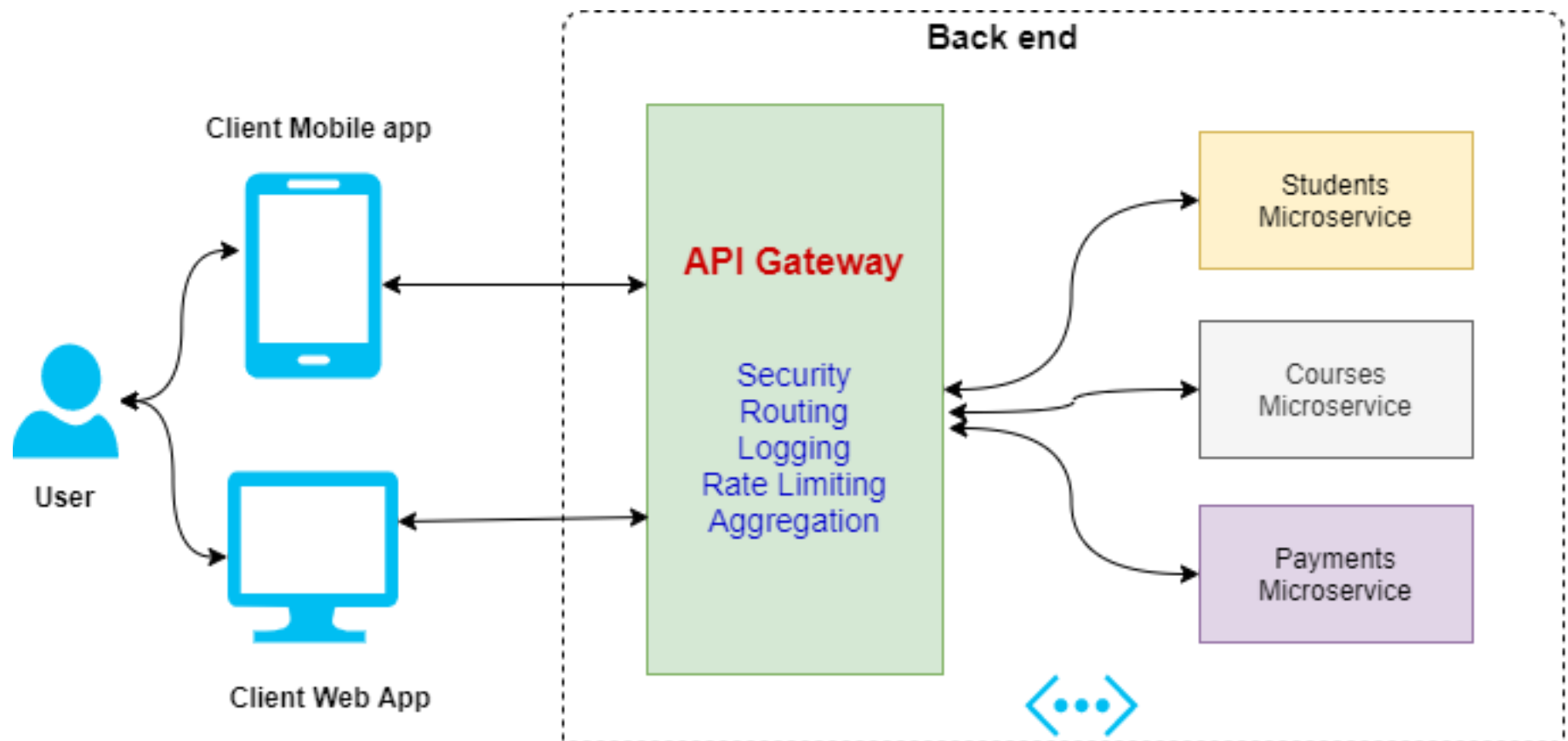


Designing and Implementing

Microservices

Microservices

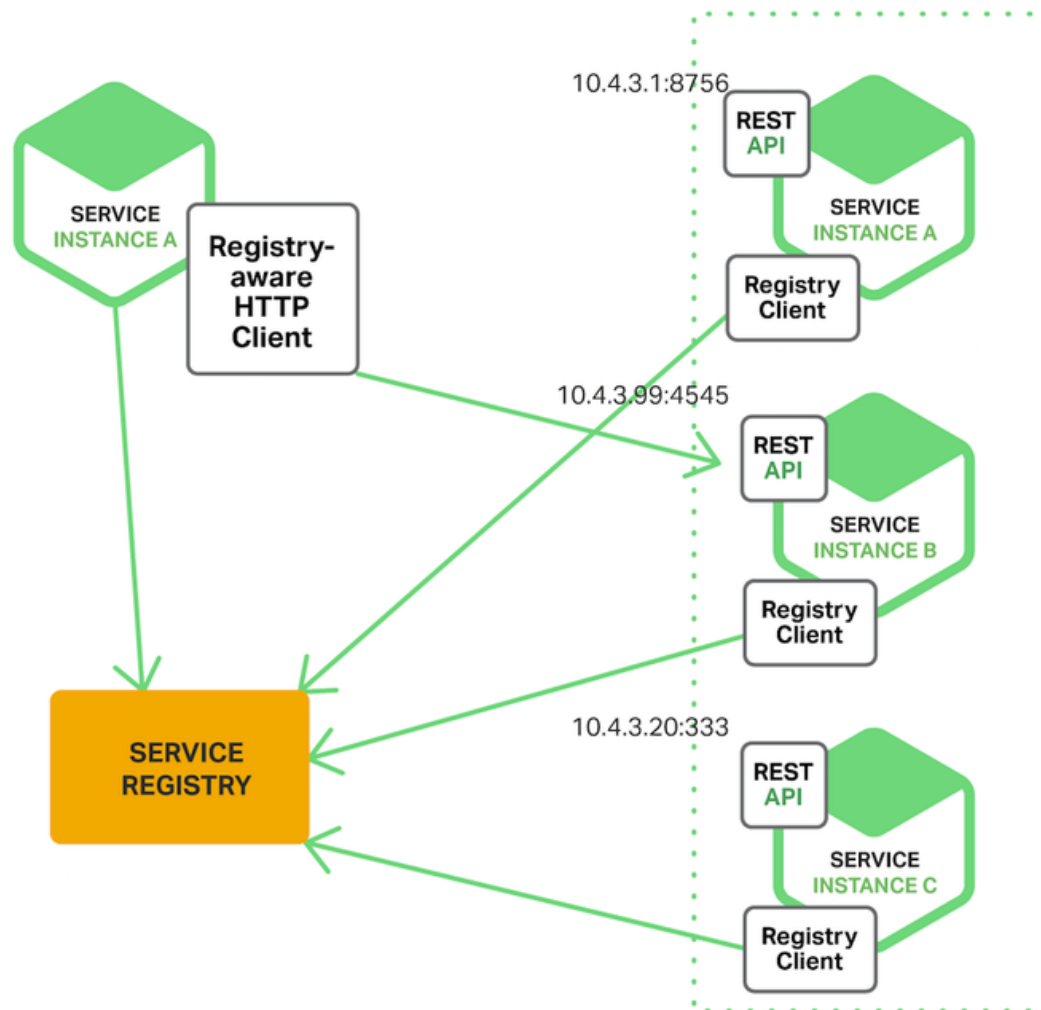
GATEWAY PATTERN



Use an API Gateway to handle client requests and route them to the appropriate microservices. This centralized authentication, load balancing, and routing logic.

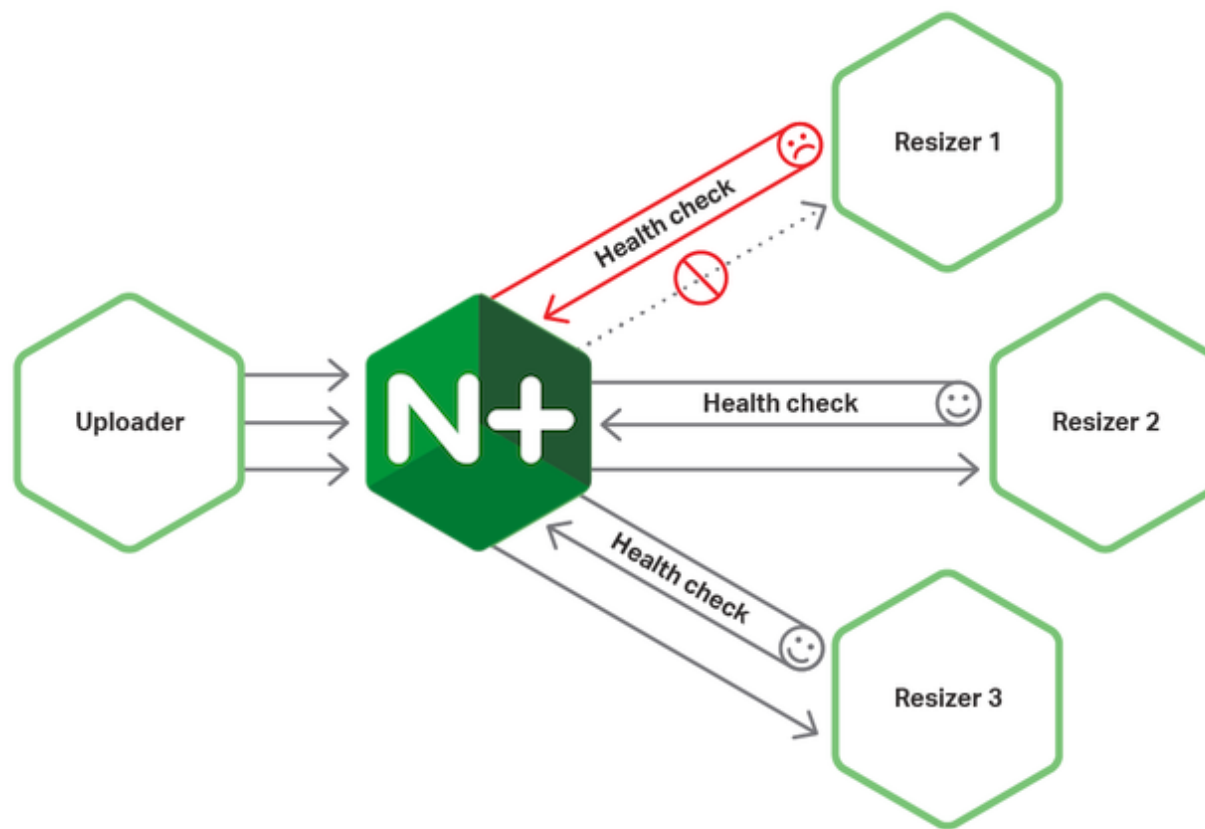
Microservices

SERVICE REGISTRY PATTERN



Implement a service registry to automatically locate and register microservices. This helps in dynamic discovery and communication between services.

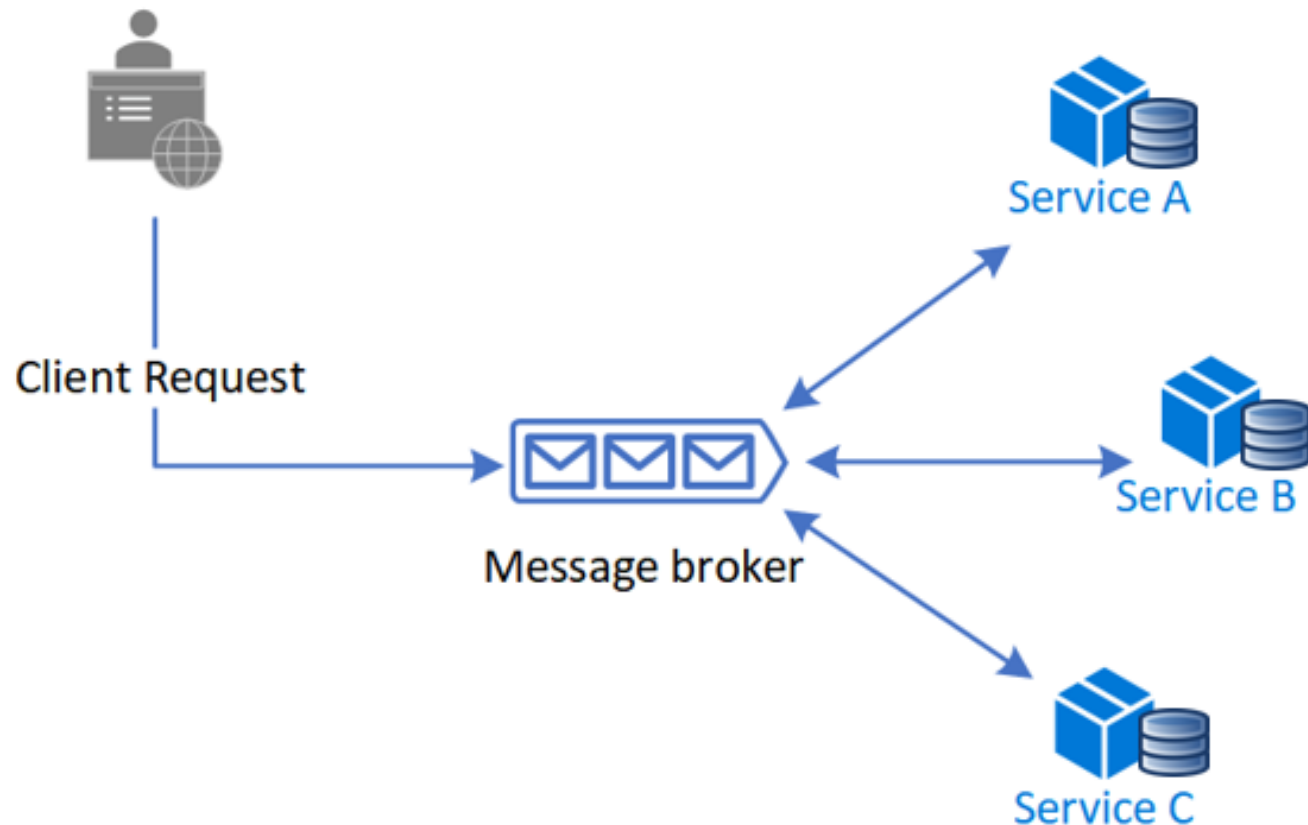
CIRCUIT BREAKER PATTERN



Prevent cascading failures by using a circuit breaker that can temporarily stop requests to a failing service and provide fallback mechanisms.

Microservices

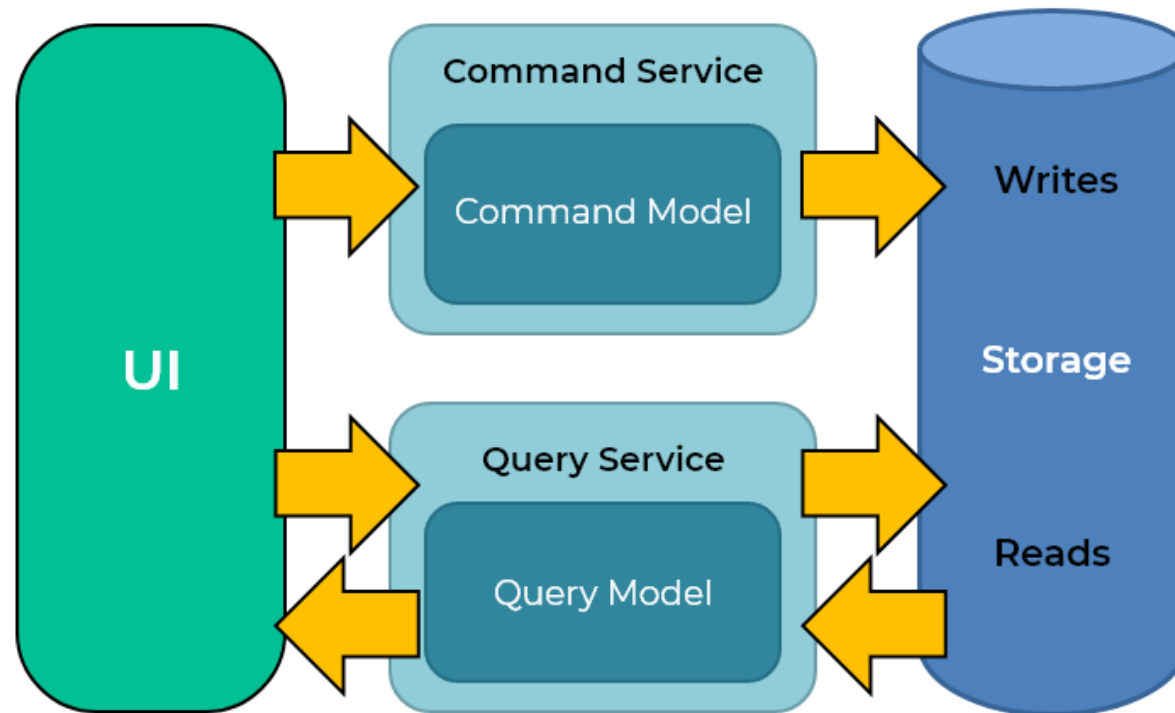
SAGA PATTERN



Manage long-lived transactions across multiple microservices by breaking them down into a sequence of smaller, local transactions.

Microservices

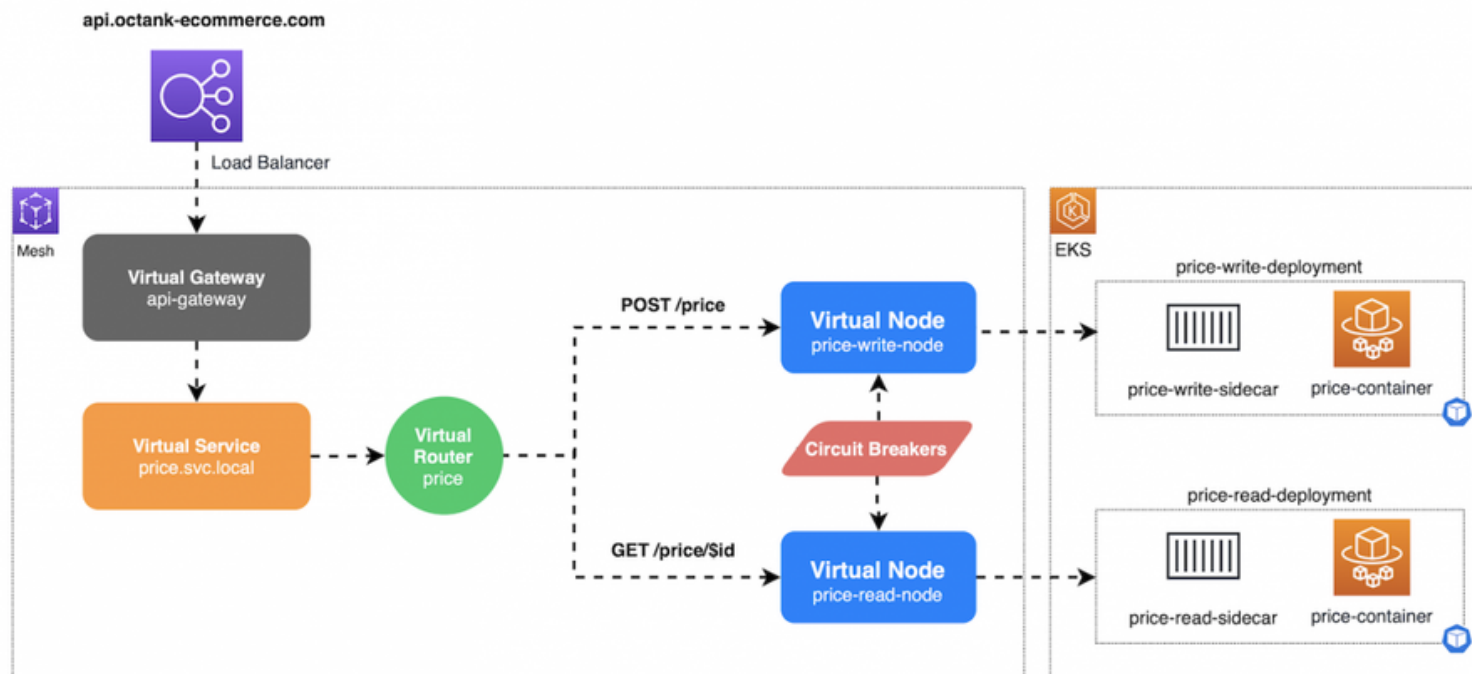
CQRS PATTERN



Separate the read and write responsibilities of a system, allowing for optimized performance and scalability.

Microservices

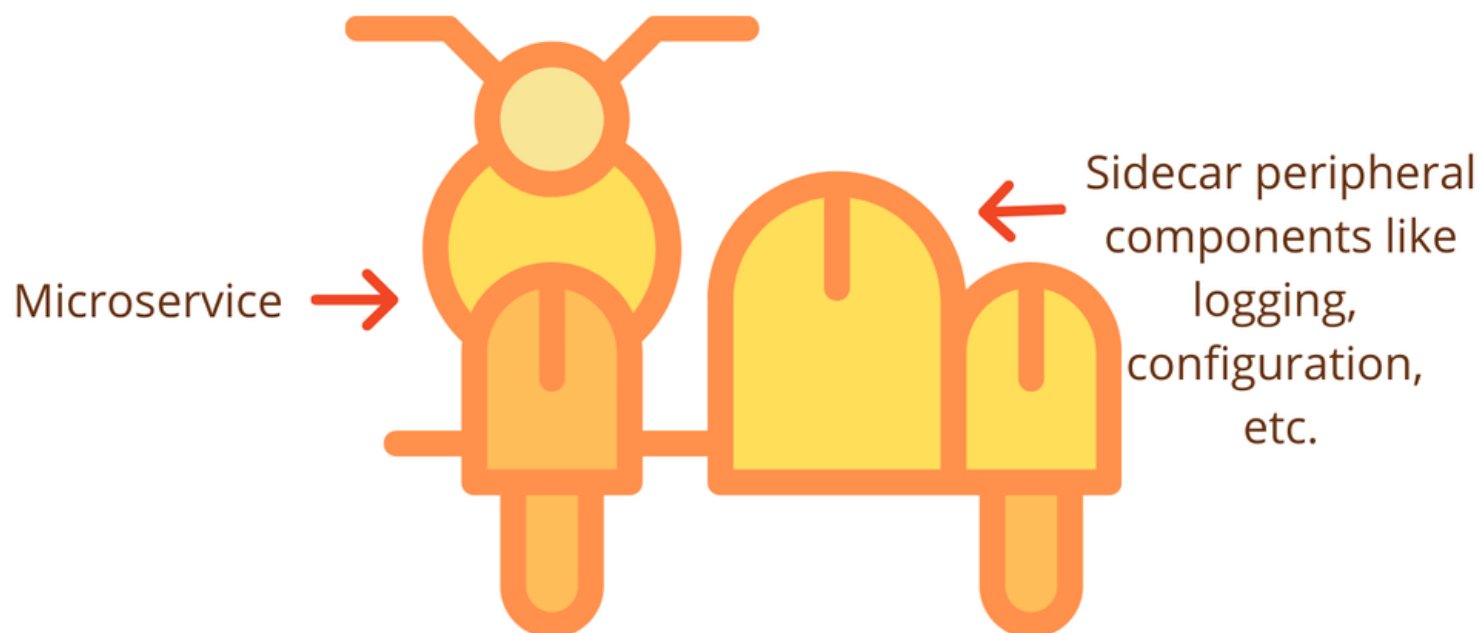
BULKHEAD PATTERN



Isolate failures within separate sections to prevent them from affecting the entire system.

Microservices

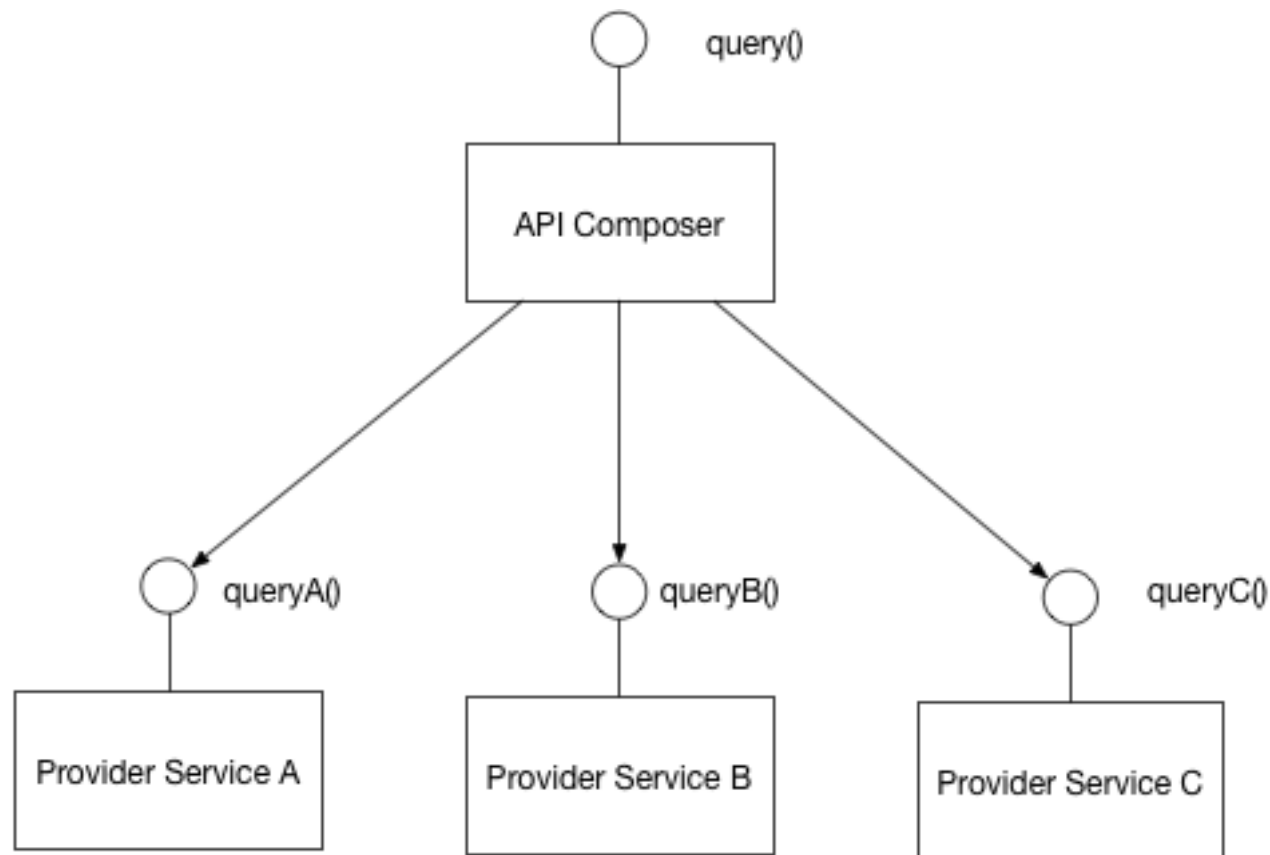
SIDECAR PATTERN



Attach a separate microservice (sidecar) to handle specific tasks like monitoring, logging, or authentication.

Microservices

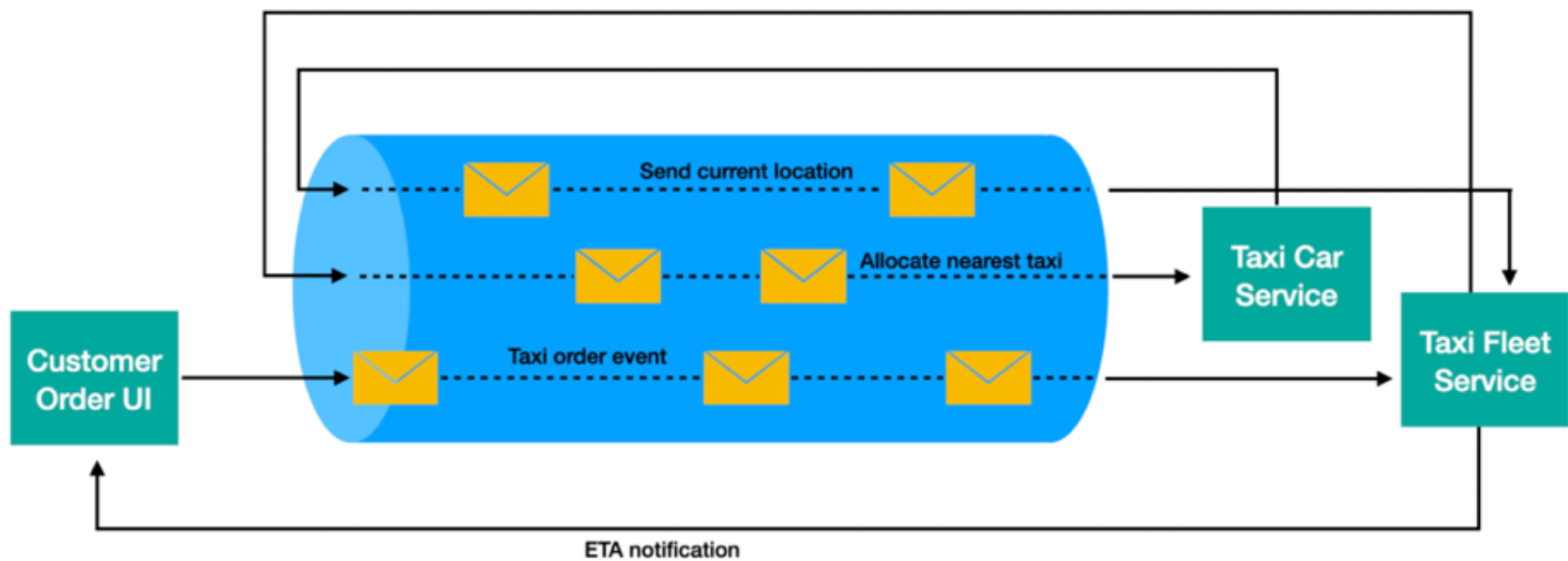
API COMPOSITION PATTERN



Combine multiple microservices to create a more complex and feature-rich API for clients.

Microservices

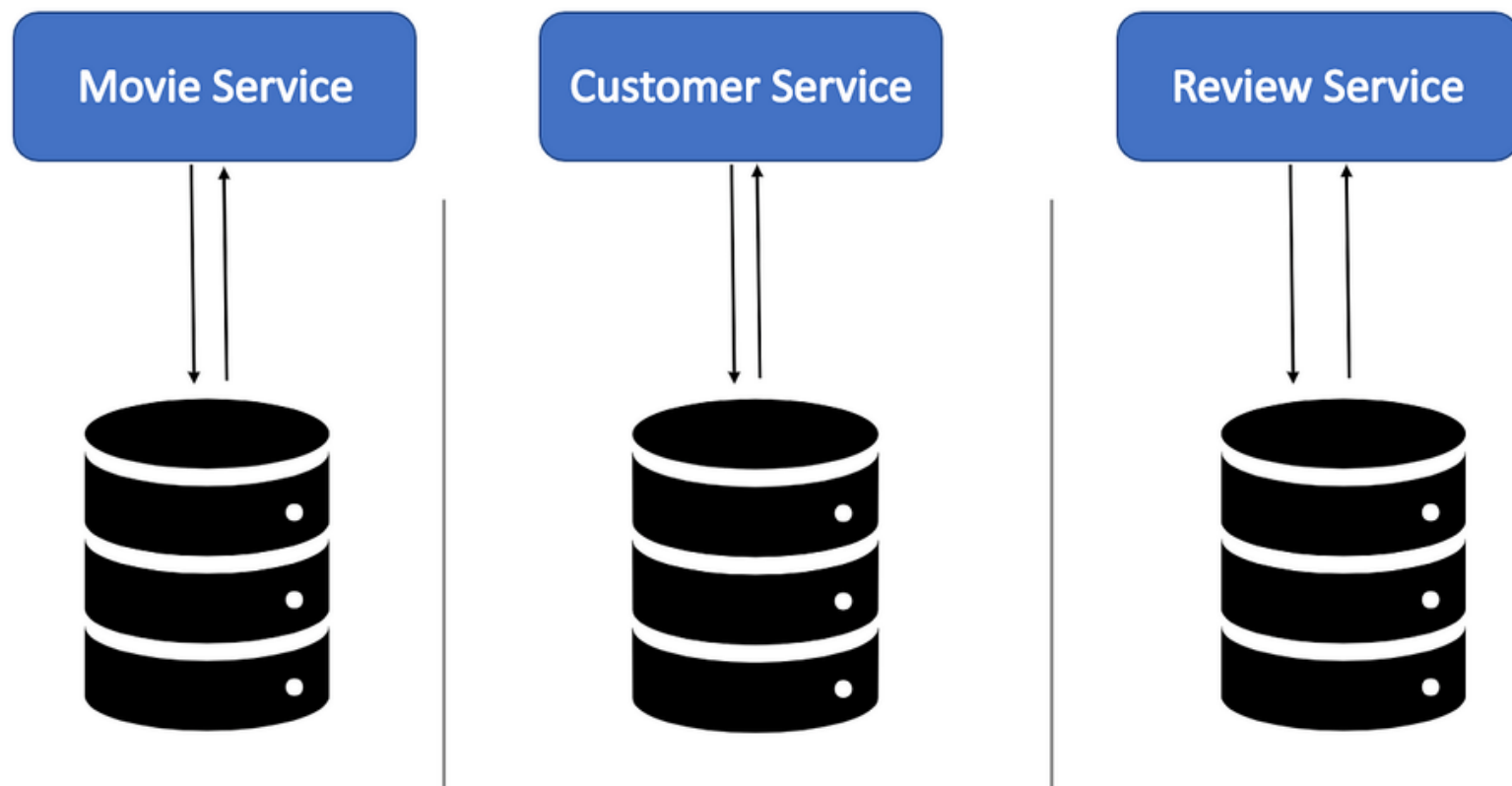
EVENT-DRIVEN ARCHITECTURE PATTERN



Communicate between microservices through events, enabling loose coupling and scalability.

Microservices

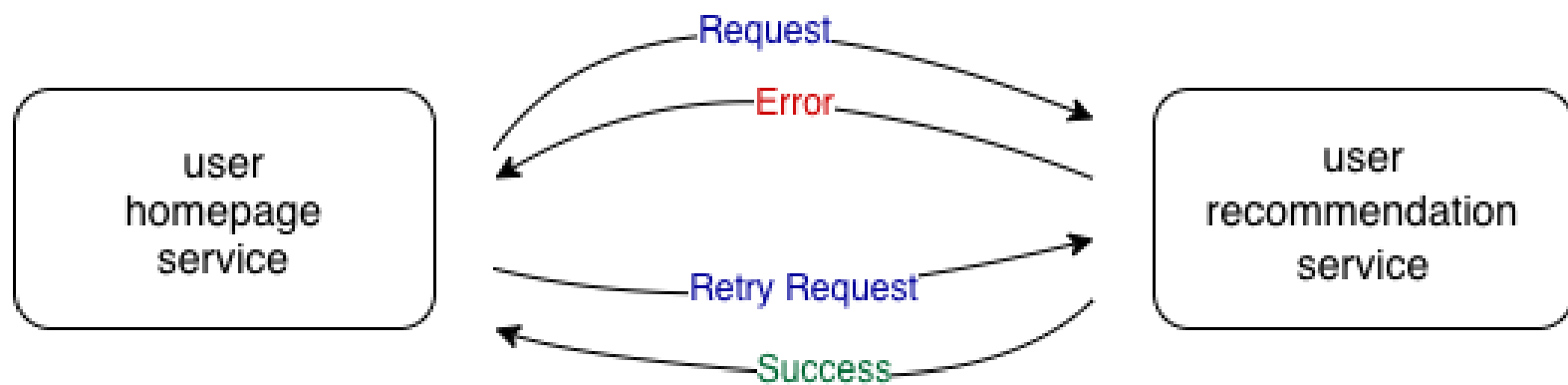
DATABASE PER SERVICE PATTERN



Each microservice has its own dedicated database to ensure loose coupling and autonomy.

Microservices

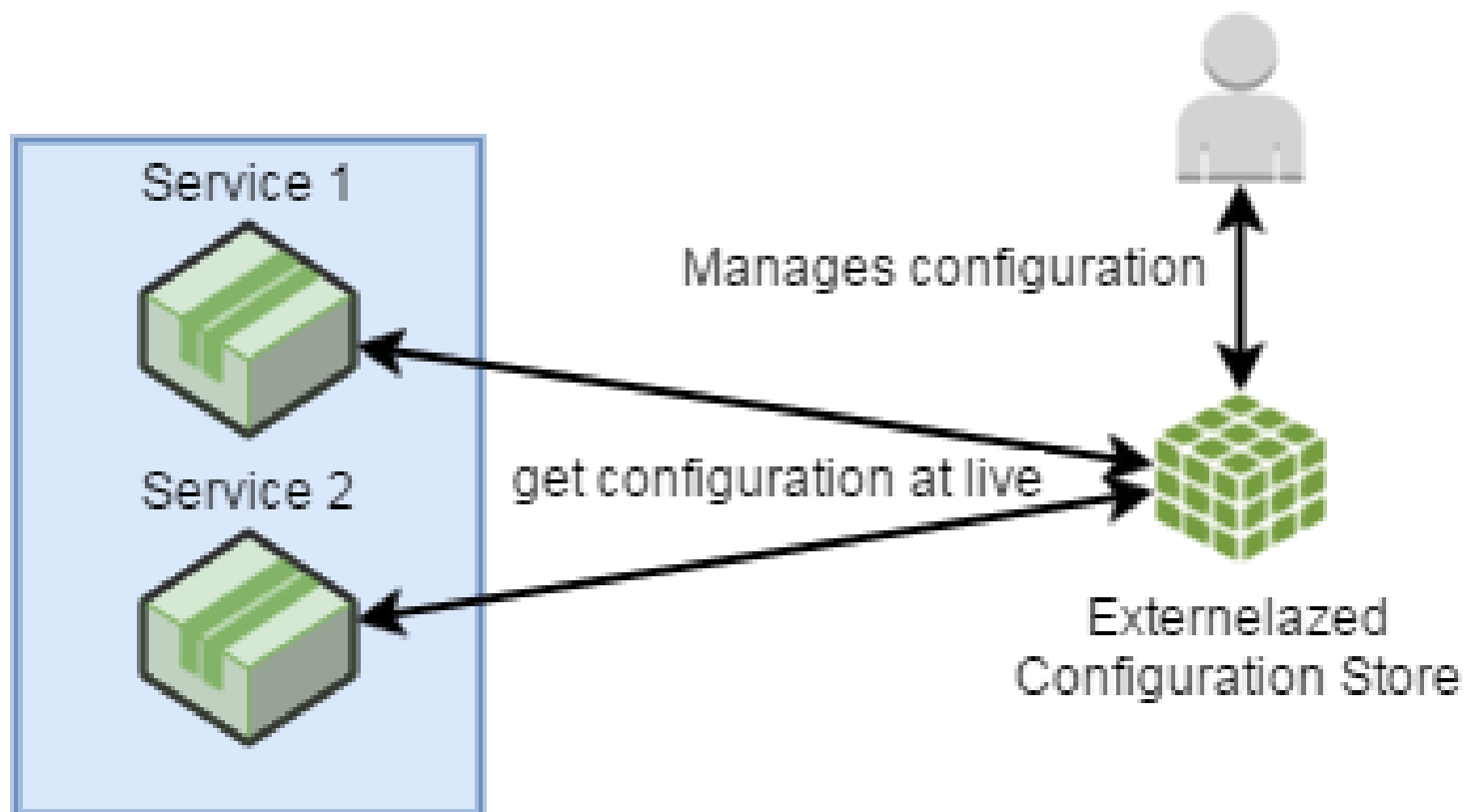
RETRY PATTERN



Automatically retry failed operations to improve the chances of success.

Microservices

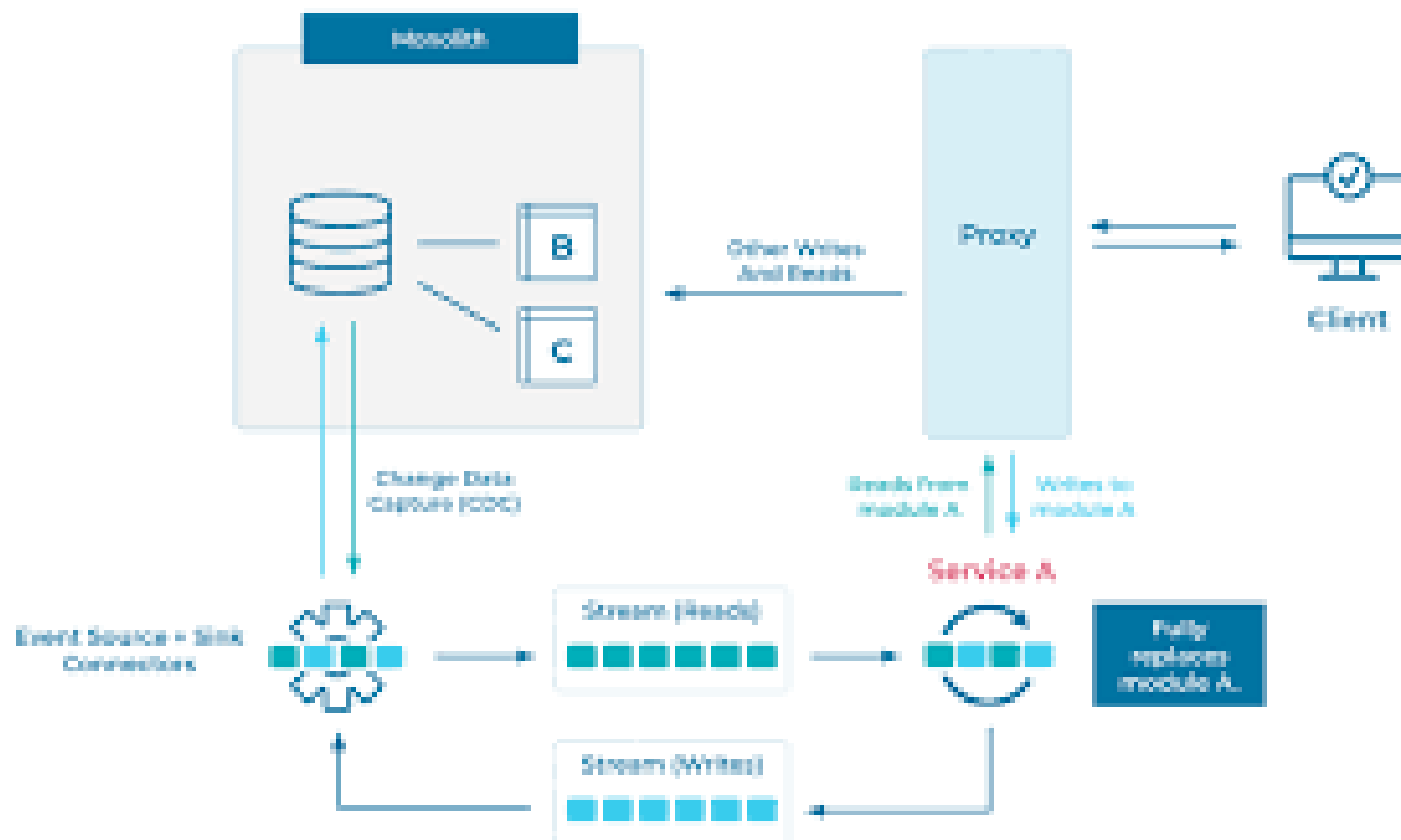
CONFIGURATION EXTERNALIZATION PATTERN



Store configuration settings outside the codebase for easier management and updates.

Microservices

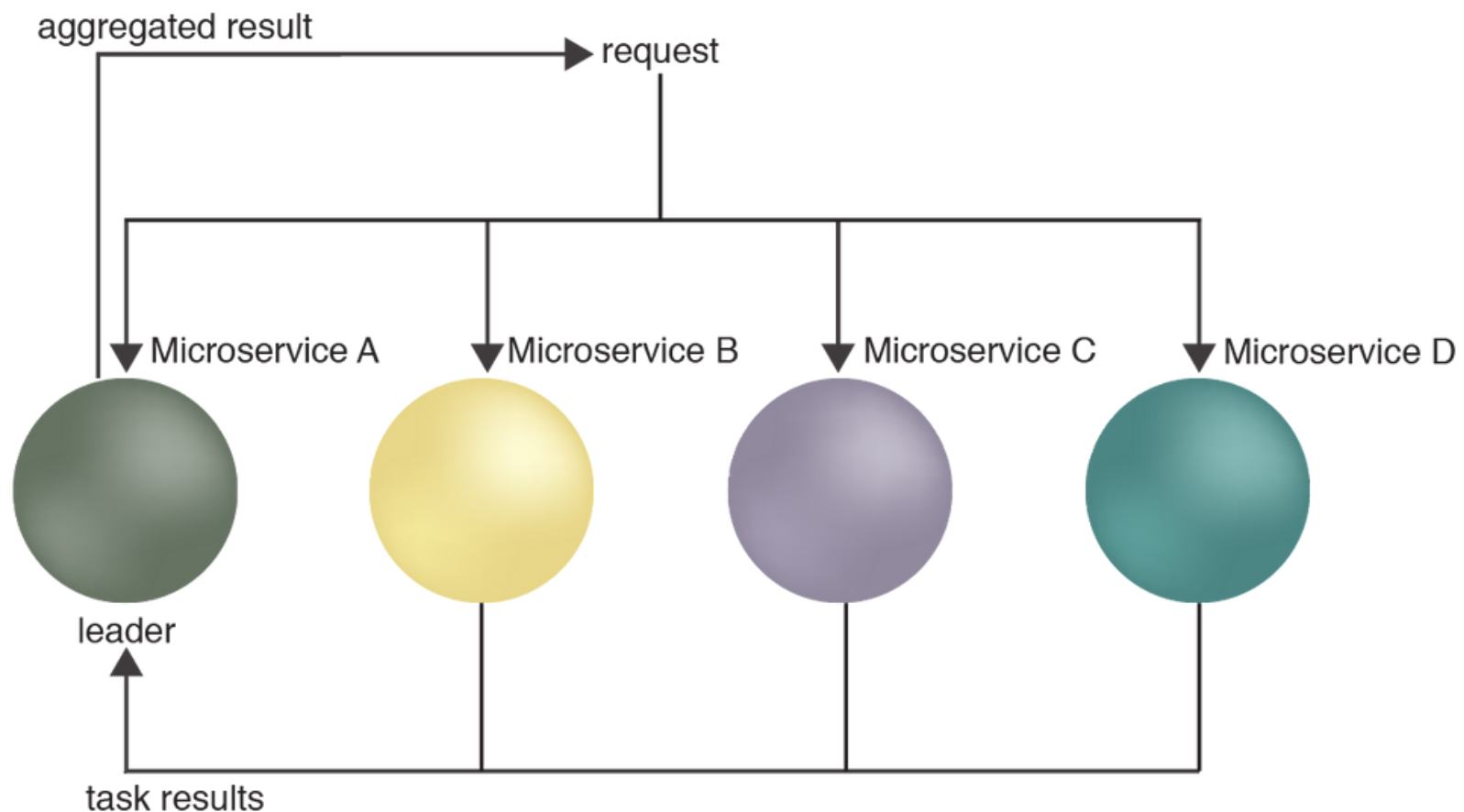
STRANGLER FIG PATTERN



Gradually replace components of a legacy system with new ones until the old system is "strangled" and replaced entirely.

Microservices

LEADER ELECTION PATTERN



Designate a leader among instances of a microservice for tasks like coordination and decision-making.