

EXPLORING SERVERLESS DESIGN PATTERNS



Building Serverless Solutions
with Azure and .NET

Overview of Serverless Design Patterns

Understanding the Importance and Characteristics of Serverless Design Patterns

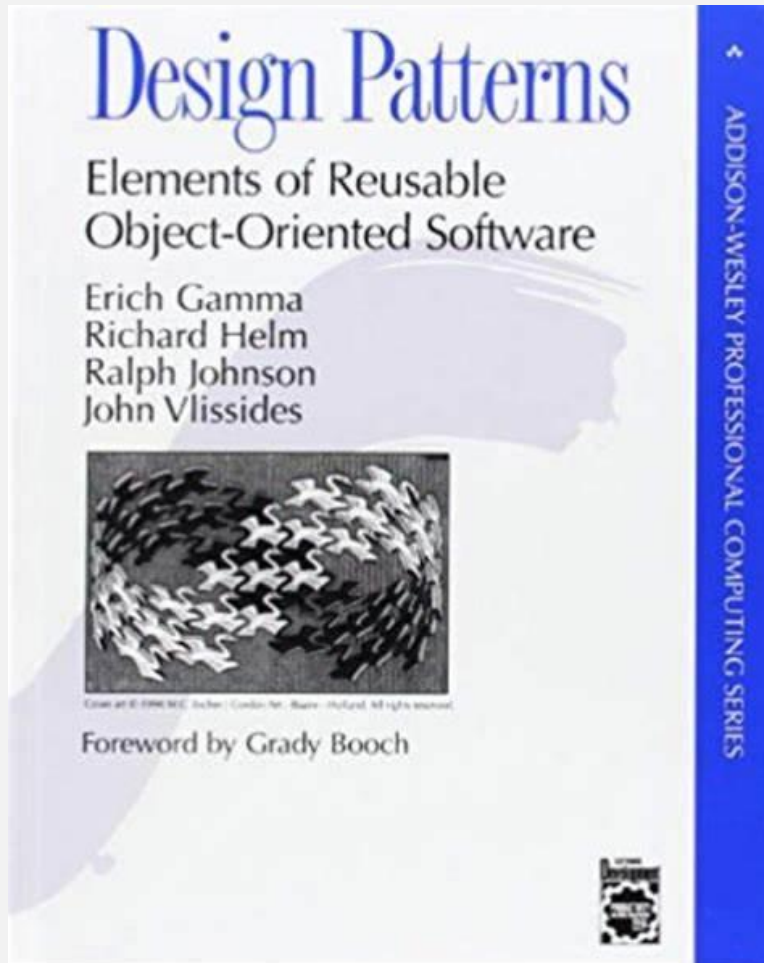
What are design Patterns

- Reusable solutions to common problems
- Best practices and proven solutions
- Building blocks for maintainable, scalable, and robust software

Why Design Patterns Matter

- Address complexity
- Encourage best practices and standardization
- Enhance code readability and maintainability
- Facilitate collaboration

Gang of Four



Types of Design Patterns

Creational

Structural

Behavioral

Concurrency

Architectural

Cloud

Serverless Design Patterns

Benefits

Cost Efficiency

Auto-Scaling

Reduced Operational
Overhead

Faster Time-to-Market

Trigger-Driven
Architecture

High Availability

Micro-Billing

Challenges

Loss of Control

Cold Starts

Usage-Based Pricing

Provider Lock-In

Testing and Debugging

Common Characteristics

Event-Driven

Scalability

Microservices

**Managed
Services**

Exploring Serverless Design Patterns

Sample of Serverless Design Patterns

High-Level Patterns

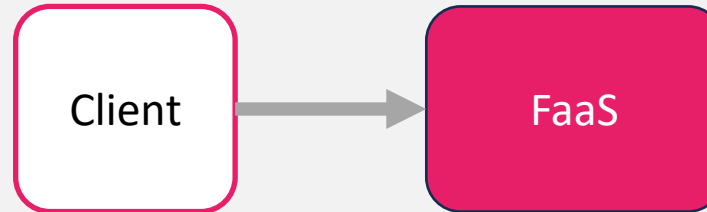


Types of Serverless Architecture

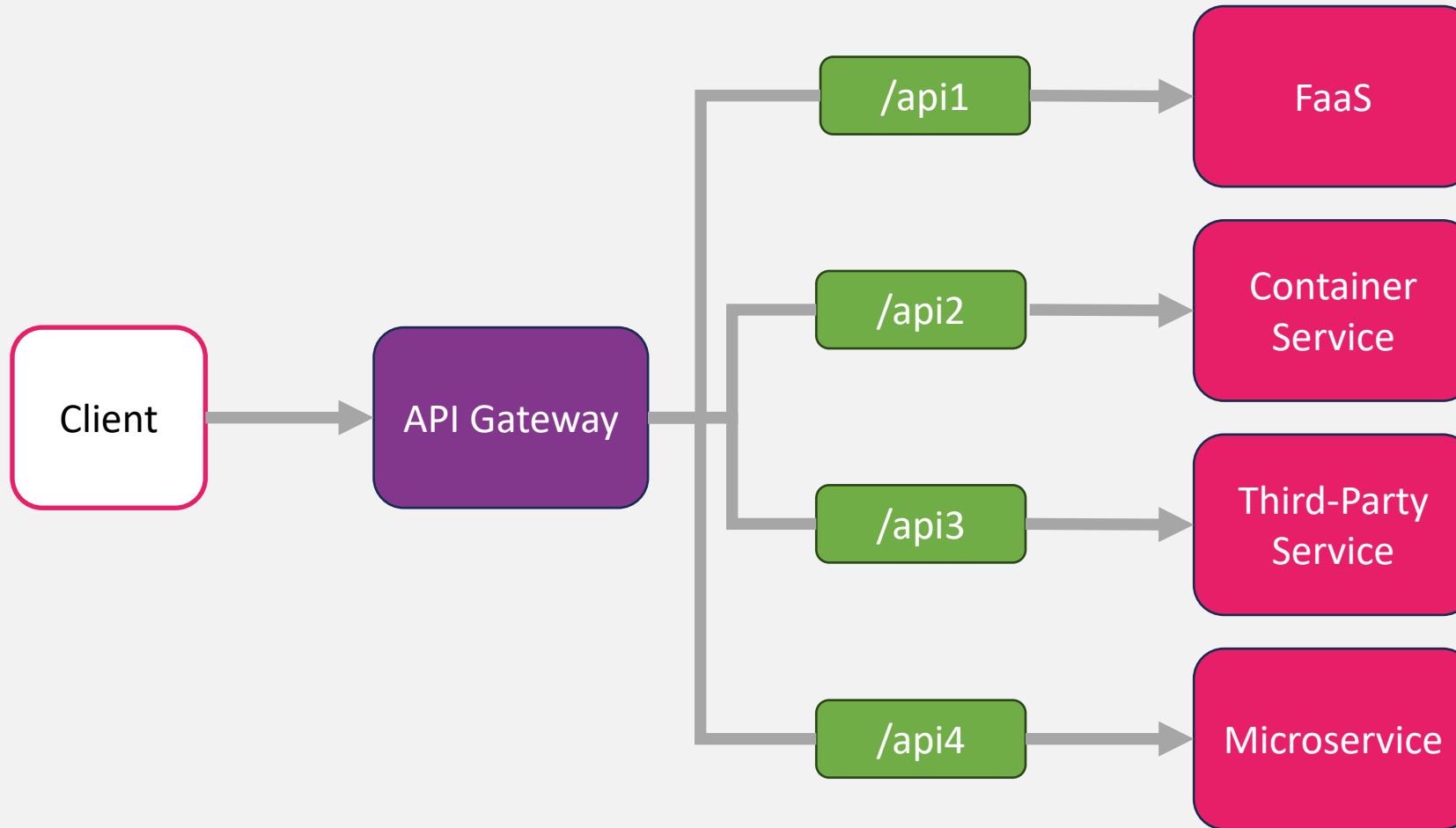
**Function as a Service
(FaaS)**

**Backend as a Service
(BaaS)**

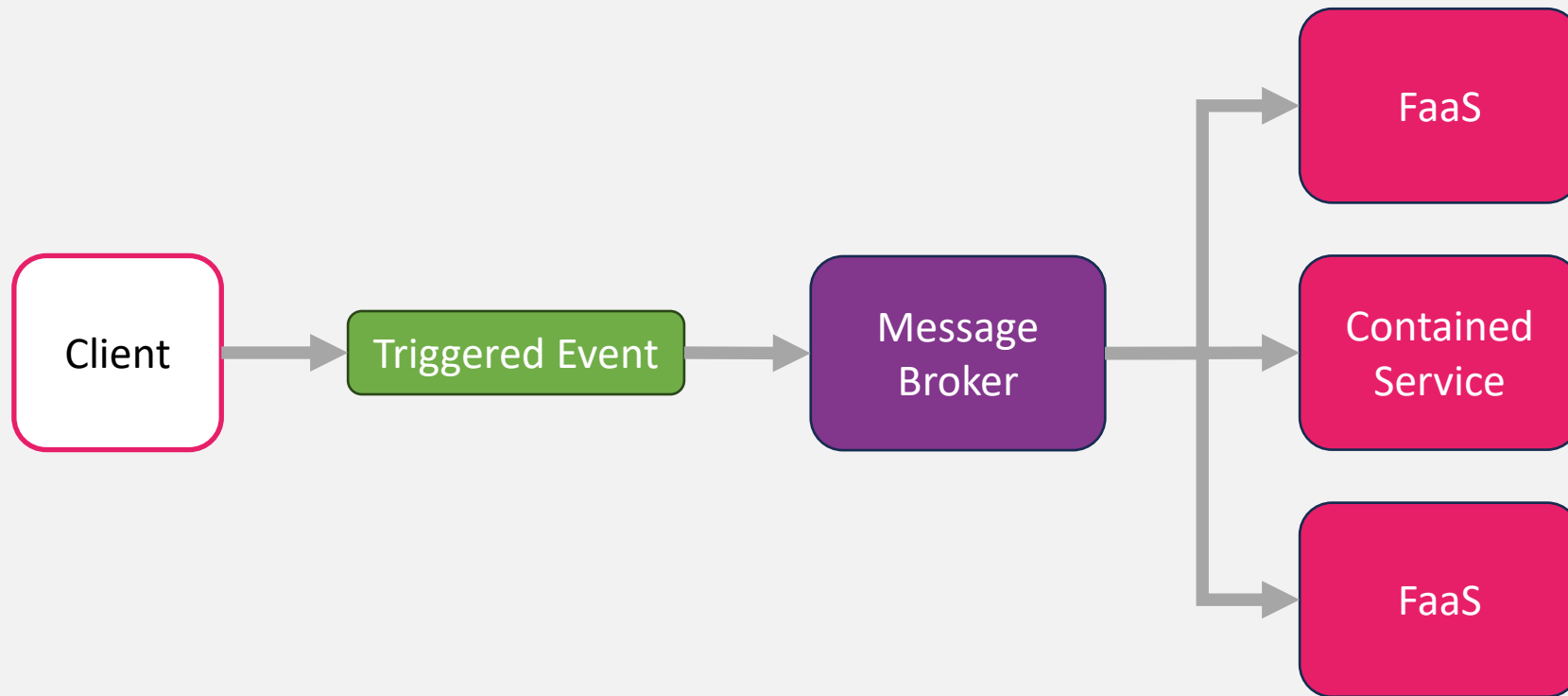
Simple Web Service



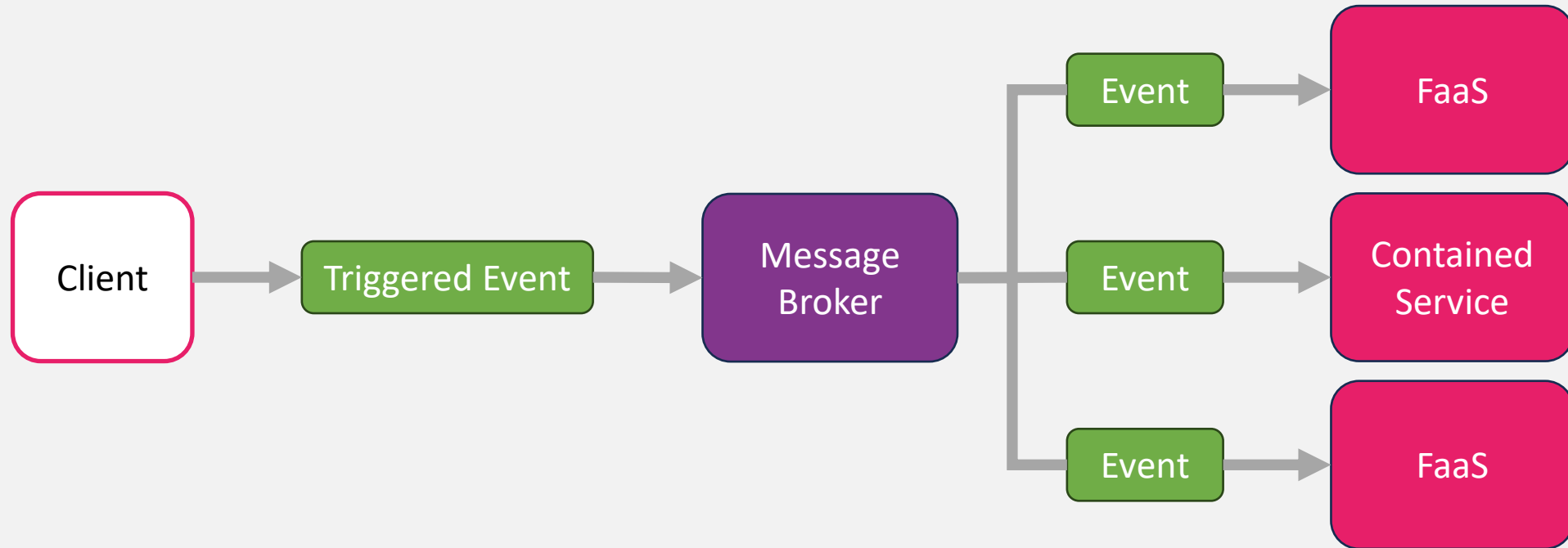
API Gateway



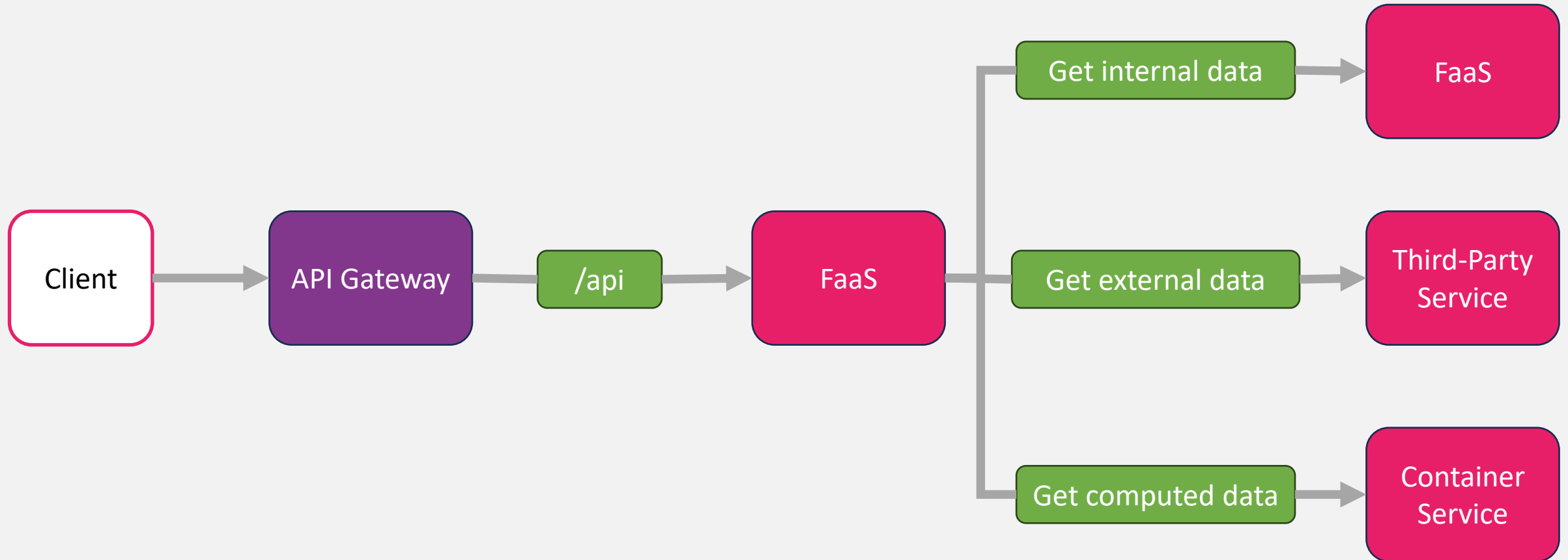
Decoupled Messaging



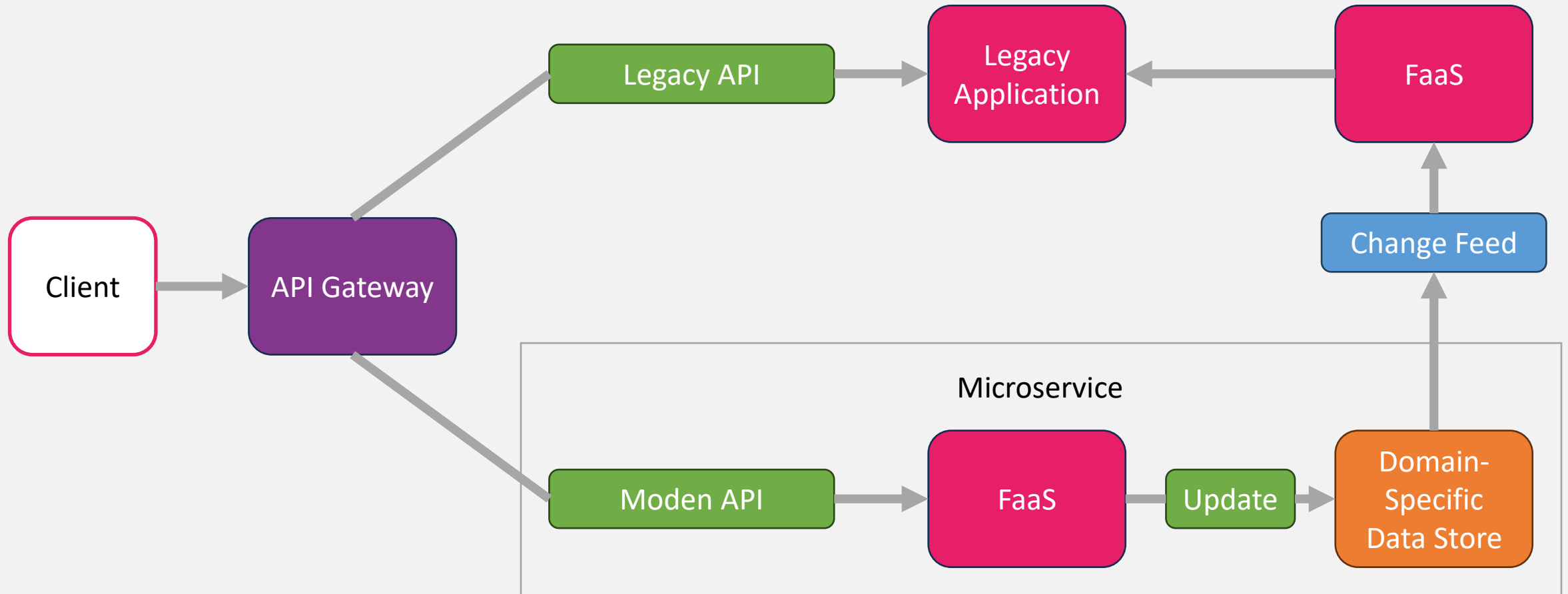
Publish/Subscribe



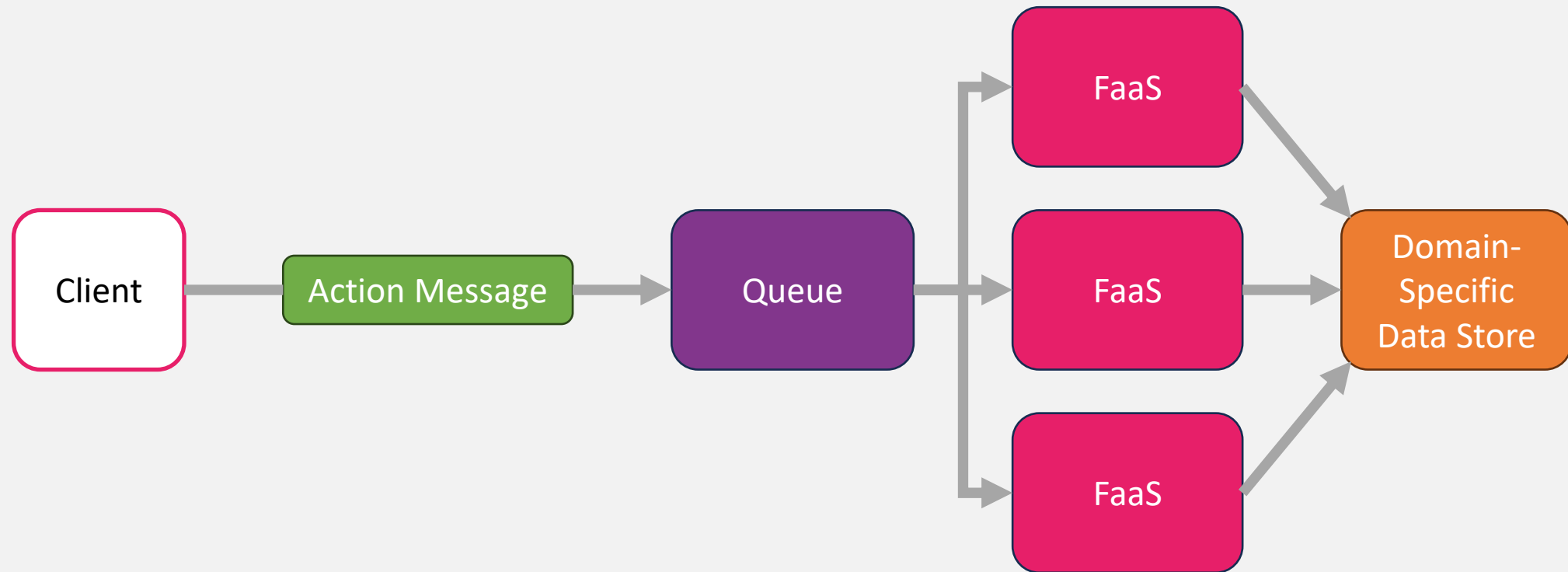
Aggregation



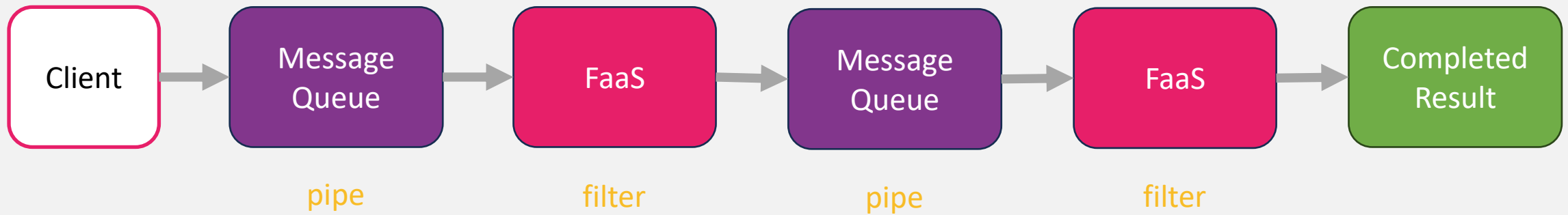
Strangler



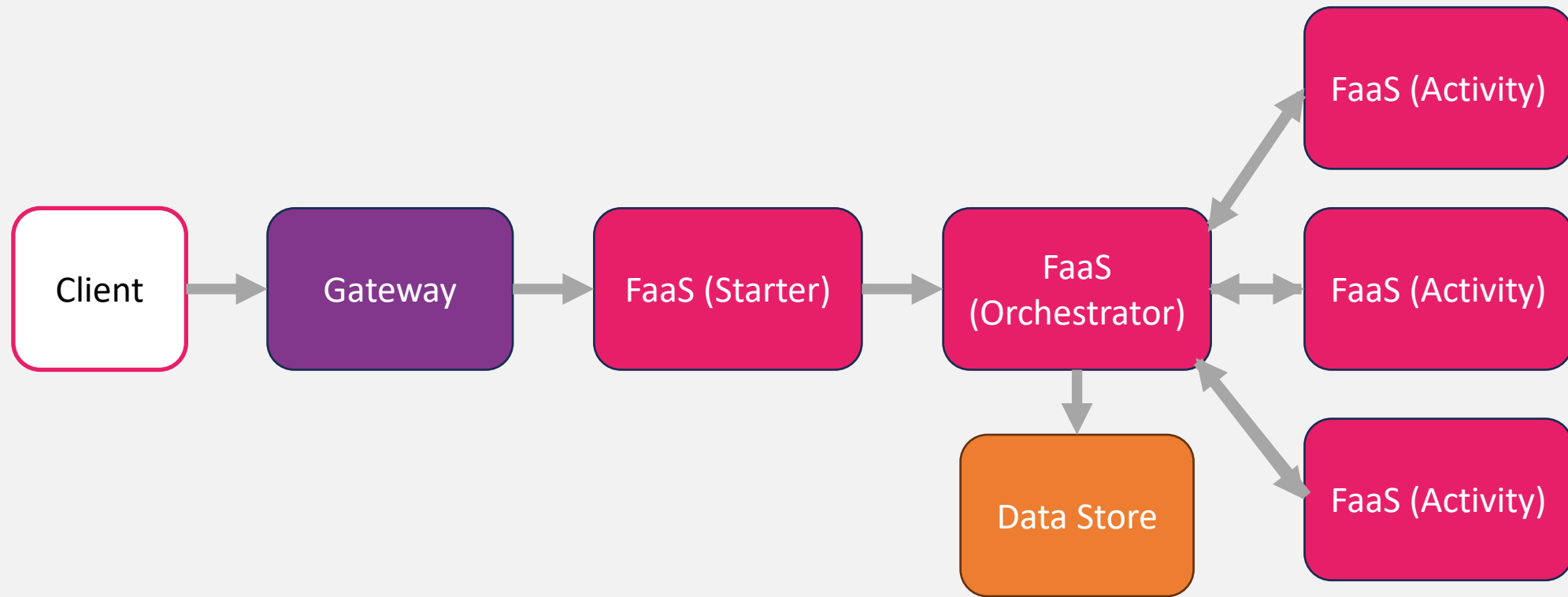
Queue-Based Load Leveling



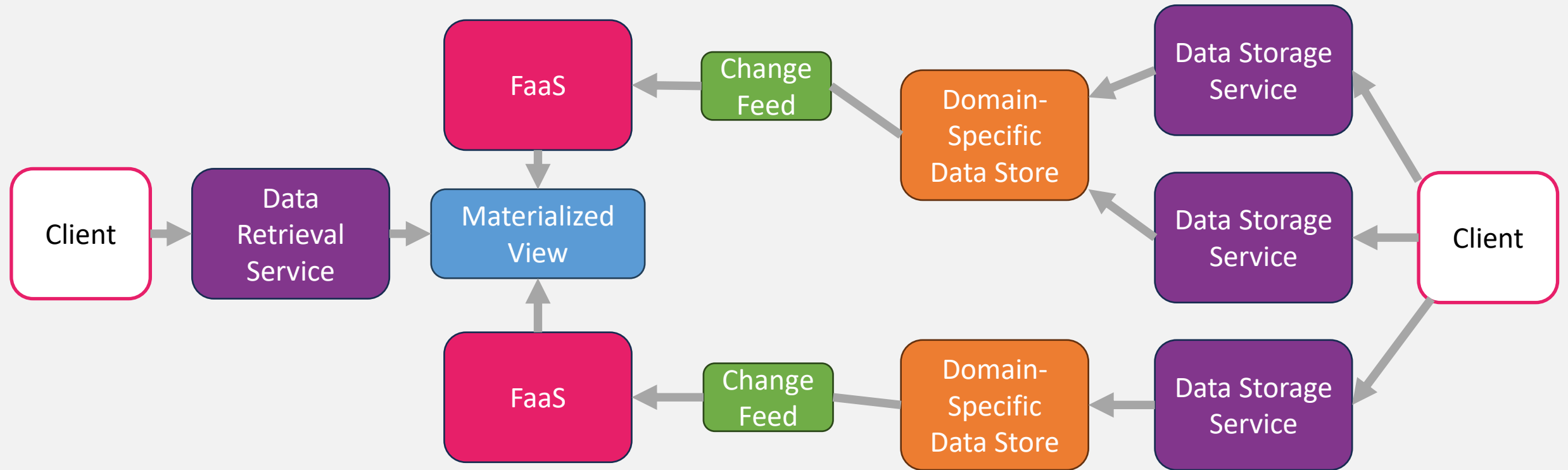
Pipes and Filters



Fan-Out/Fan-In



Materialized Views



LAB SCENARIO

Building Serverless Solutions
with Azure and .NET