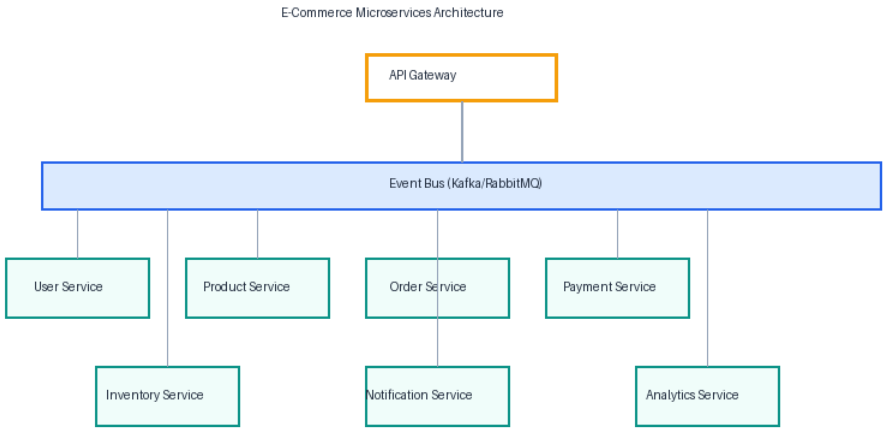
Principles in Practice: E-Commerce Platform Architecture

Scenario: Designing a Scalable E-Commerce Platform

Our hypothetical e-commerce platform serves millions of users globally, requiring high availability, scalability, and maintainability. We implement **Microservices Architecture** combined with **Event-Driven Architecture (EDA)** to handle complex business operations while maintaining loose coupling and system resilience.



Microservices & Event-Driven Architecture

Each microservice owns its domain and database, communicating asynchronously:

- **User Service:** Authentication, profiles
- **Product Service:** Catalog, search
- **Order Service:** Order processing
- **Payment Service:** Transactions, refunds
- **Inventory Service:** Stock management
- **Notification Service:** Email, SMS, push

Event Flow: Order placement triggers 'OrderPlaced' event; services process asynchronously.

Applying SOLID Principles

- **S:** Each service has single responsibility
- **O:** Extend via plugins, not modifications
- **L:** Payment gateways share interface
- **I:** Clients use only needed interfaces
- **D:** Depend on abstractions, not concrete

DRY (Don't Repeat Yourself)

- Centralized Auth Library across services
- Shared validation in domain models
- Reusable event schemas in registry
- Common logging/monitoring configs
- Auto-generated API client SDKs

KISS (Keep It Simple)

- Simple REST APIs with JSON
- Minimal event data; API for details
- Database per service; eventual consistency
- Docker containers; independent deployment
- Simple retry with exponential backoff
- Clear naming conventions

Outcome:

This architecture enables independent scaling, fault isolation, faster development, and easier maintenance. Teams deploy services independently, choose optimal technologies per service, and ensure system resilience.