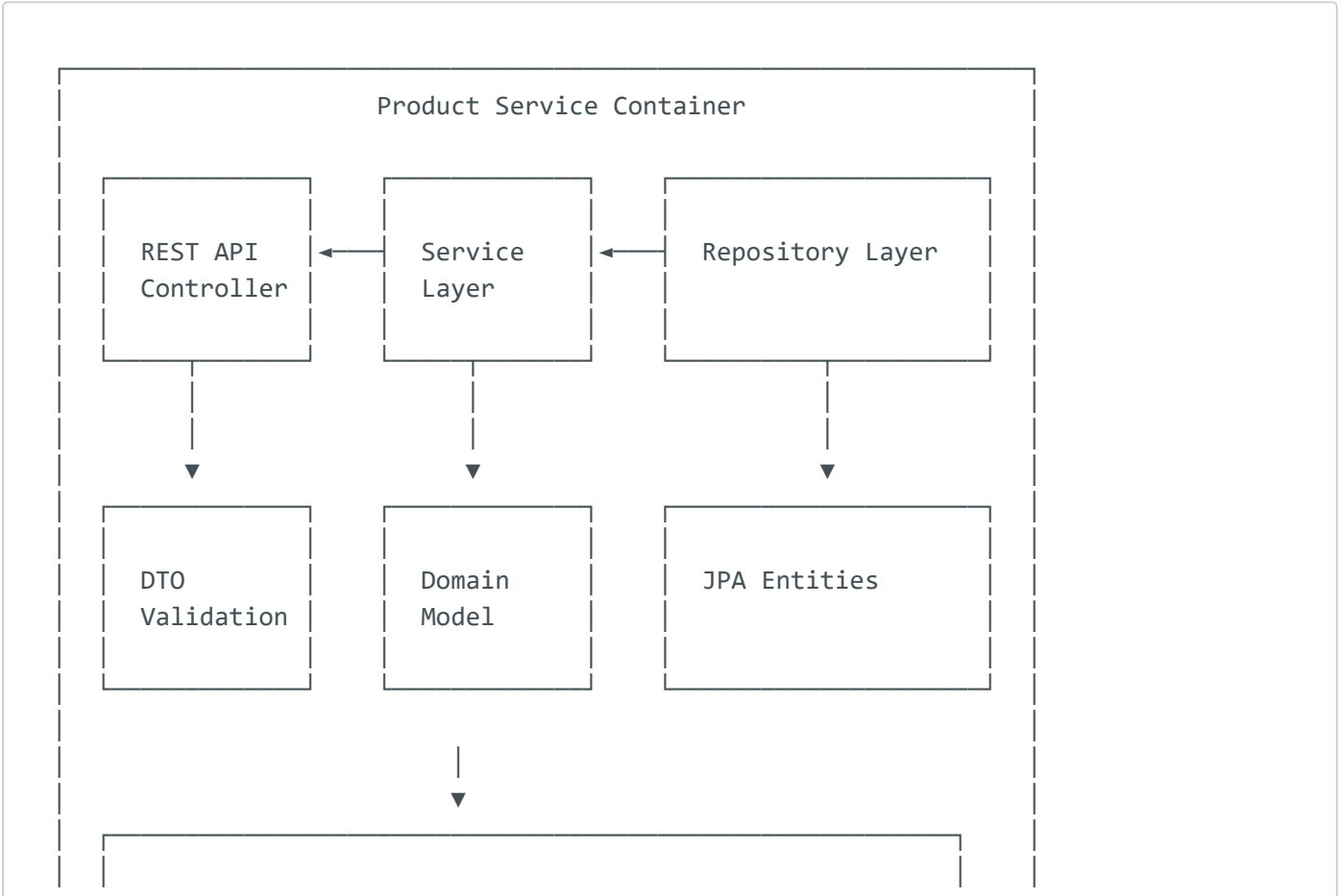
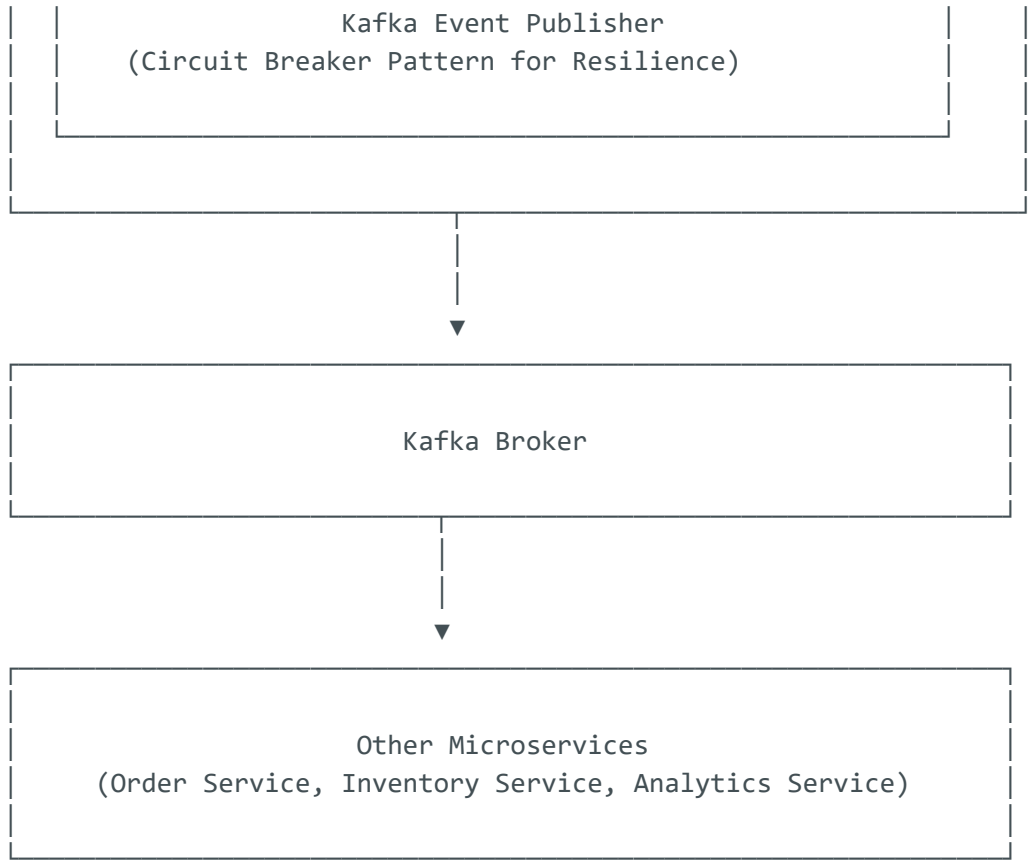
Product Service Architecture

Architecture Overview

The Product Service follows a modern microservice architecture based on twelve-factor app principles, focusing on event-driven communication and containerized deployment. The architecture is designed to be cloud-native, resilient, and maintainable.





Data Flow

1. **REST API Request Flow**:
    - External clients request product data through REST API endpoints
    - Controllers validate input and convert to DTOs
    - Service layer applies business logic
    - Repository layer interacts with PostgreSQL database
    - Response returns through the same layers (Repository → Service → Controller → Client)
2. **Event Publishing Flow**:
    - Product changes trigger events in the Service layer
    - ProductEventPublisher formats and publishes events to Kafka
    - Circuit breaker pattern provides resilience for Kafka communication
    - Other microservices consume events for their specific business needs

Design Patterns

The Product Service implements several key design patterns:

1. **Repository Pattern**
    - Abstracts data access logic
    - Enables clean separation between business logic and data access
2. **Circuit Breaker Pattern**
    - Implemented using Resilience4j
    - Prevents cascading failures when Kafka is unreachable
    - Provides fallback mechanisms for event publishing
3. **Event-Driven Architecture**
    - Decouples components with asynchronous communication
    - Enables real-time data propagation across the ecosystem
    - Supports eventual consistency between services
4. **Dependency Injection**
    - Spring's IoC container manages component lifecycle
    - Enhances testability and maintainability
5. **DTO Pattern**
    - Separates internal domain model from external API representation
    - Provides validation at the boundary

Infrastructure Components

Spring Boot Application

- The core Java application runs in a Docker container
- Exposes REST endpoints for product management
- Handles business logic and data persistence

PostgreSQL Database

- Stores product information
- Runs in its own container
- Volume-mounted for data persistence

Kafka Ecosystem

- Broker: Handles event distribution
- Zookeeper: Manages Kafka cluster state
- Kafka UI: Web interface for topic monitoring and management

Docker Network

- Custom 'product-network' for inter-container communication
- Isolates the application components

Twelve-Factor App Principles Implementation

1. **Codebase**: Single repository for the product service
2. **Dependencies**: Explicitly declared in pom.xml
3. **Config**: Environment-specific configuration in application-{profile}.yml files
4. **Backing Services**: PostgreSQL and Kafka as attached resources
5. **Build, Release, Run**: Distinct stages in Docker build process
6. **Processes**: Stateless application design
7. **Port Binding**: Self-contained HTTP service on port 8080
8. **Concurrency**: Horizontal scaling capability
9. **Disposability**: Fast startup/shutdown in containers
10. **Dev/Prod Parity**: Development environment mirrors production
11. **Logs**: Treated as event streams
12. **Admin Processes**: Management tasks as one-off processes