Faysal Aziz

Software Development Principles for Microservices

Java service design, data boundaries, resilience and Kafka integration

FA-0695 · 5 days

Software Development · DevOps, Cloud & Infrastructure

Introduction

This five-day course explores microservices through software-engineering principles and selected Java exercises. It covers boundaries, APIs, data ownership, security, observability, resilience, Kafka integration and testing.

Microservices are one architectural option, not an automatic improvement over a monolith. Participants evaluate trade-offs and build a bounded sample using core Java; distributed-system patterns are selected according to the problem rather than imposed on every service.

Learning outcomes

- Compare microservices, monoliths and SOA using coupling, cohesion and operational trade-offs.
- Model domains, bounded contexts and API contracts for a sample system.
- Develop selected Java services and explain versioning and contract-testing choices.
- Evaluate data ownership, consistency, event sourcing and CQRS without assuming all patterns are required.
- Apply selected authorisation, token-validation, observability and resilience controls.
- Integrate a sample Java producer/consumer with Kafka and interpret delivery/processing trade-offs.
- Evaluate scaling options and test selected unit, integration, contract and end-to-end behaviours.

Prerequisites

- Basic knowledge of Java programming.
- Familiarity with software architecture concepts.
- Understanding of web services and APIs.

Training outline

1. Day 1 — Architecture and design principles

- Monoliths, microservices and SOA: compare deployment, coordination and data-consistency trade-offs.
- Autonomy, abstraction, encapsulation, modularity, single responsibility, loose coupling and high cohesion.
- Assess example use cases and avoid splitting services solely by technical layers or size.

2. Day 2 — Domains and API contracts

- DDD subdomains, bounded contexts, aggregates, entities, value objects and context maps.
- Define service boundaries and model a selected domain in Java.
- REST principles, OpenAPI documentation, contract-first design and API-versioning choices.
- Consumer-driven contracts and compatibility across independent deployments.

3. Day 3 — Data and security

- Service data ownership versus shared databases, polyglot persistence and consistency trade-offs.
- Event sourcing and CQRS as optional patterns with additional complexity.
- Authentication, authorisation and service-to-service trust; OAuth 2.0 is not itself an identity protocol.
- JWT claim/signature validation, secure token handling and appropriate identity-provider integration.

4. Day 4 — Observability and resilience

- Trace request flow across services; correlate distributed traces, logs and operational signals.
- Centralised logging and diagnostic practices appropriate to sensitive data and system constraints.
- Timeouts, bounded retries and circuit breakers; retries require attention to idempotency and load.
- Bulkheads and isolation to limit cascading failures; exercise a selected failure scenario.

5. Day 5 — Kafka, scaling and testing

- Compare queue and publish/subscribe concepts with Kafka topics, partitions and consumer groups.
- Use core Java producer/consumer clients in a prepared Kafka lab; discuss ordering and processing guarantees in context.
- Horizontal and vertical scaling, load balancing, rate limiting and bottleneck measurement.
- Unit, integration, consumer-contract and end-to-end testing; write selected meaningful tests for the Java sample.
- Review sample deployment choices, architectural trade-offs and further practice.

© 2026 Faysal Aziz · Kuala Lumpur