Faysal Aziz

Java Engineering Masterclass

From architecture to deployment in 2 days

FA-0270 · 2 days

Software Development · DevOps, Cloud & Infrastructure

Introduction

In modern software ecosystems, Java continues to anchor enterprise-scale systems — yet excellence now requires mastery far beyond core syntax. Professionals must architect maintainable systems, design asynchronous operations, build secure REST APIs, and ensure seamless deployment through CI/CD pipelines. This advanced two-day course, led by an instructor with **over 30 years of industry experience**, emphasizes **practical, production-oriented approaches** rather than academic exercises. Participants will explore the architecture and engineering patterns demanded by real-world Java solutions.

Learning outcomes

Participants will be able to:

- Design scalable, maintainable Java application architectures.
- Implement asynchronous processing flows using modern Java concurrency.
- Build efficient, secure RESTful APIs.
- Integrate advanced security principles into Java systems.
- Establish automated CI/CD pipelines for Java applications.
- Apply architectural principles that enhance observability, scalability, and maintainability.

Prerequisites

- Strong proficiency in Java (Java 8 or higher).
- Experience developing backend systems (preferably with Spring Boot or similar frameworks).
- Familiarity with RESTful APIs, security concepts, and build tools (Maven or Gradle).

- Basic understanding of Git and CI/CD concepts.

Training outline

1. Architectural Design

Application architecture fundamentals

- Monolithic, microservices, and modular-monolith patterns
- Architectural trade-offs and scalability considerations

Layered and hexagonal architectures

- Domain-driven design principles
- Separation of concerns and service boundaries

Framework and technology selection

- Evaluating Spring Boot, Quarkus, Micronaut, and Jakarta EE

Data architecture considerations

- Schema design and data ownership in microservices
- Avoiding shared database anti-patterns

Integration and communication

- REST vs messaging systems
- Event-driven architectures and service orchestration

Observability and maintainability

- Logging, tracing, metrics, and operational insights
- Managing technical debt and architectural drift

2. Asynchronous Processing

Concurrency paradigms in modern Java

- Threads, executors, futures, and reactive programming

Virtual threads and structured concurrency

- Benefits and limitations of new concurrency models

Asynchronous design patterns

- Producer-consumer, event loops, and task scheduling

Error handling and context propagation

- Managing exceptions, timeouts, and cancellations

Performance and stability considerations

- Thread management, monitoring, and tuning

3. REST API Design

RESTful principles and resource modeling

- Resource identification, HTTP methods, and status codes

API lifecycle management

- Versioning, documentation, and deprecation policies

API security and governance

- Authentication, authorization, and rate limiting

Performance optimization

- Pagination, filtering, caching, and compression

Deployment-oriented API considerations

- Gateways, load balancing, and backward compatibility

4. Security Measures

Secure coding principles in Java

- Avoiding injection, serialization, and validation flaws

Authentication and authorization mechanisms

- OAuth2, JWT, and token lifecycle management

Data security practices

- Encryption at rest and in transit
- Key management and secure configuration

Security testing and monitoring

- Static and dynamic analysis, auditing, and alerts

Regulatory and compliance considerations

- Governance, audit logging, and policy enforcement

5. CI/CD Deployment Experience

CI/CD fundamentals for Java systems

- Build, test, package, deploy, and monitor stages

Continuous integration practices

- Automated testing, code coverage, and quality gates

Deployment automation

- Containerization, artifact management, and versioning

Deployment strategies

- Rolling updates, blue-green, and canary deployments

Infrastructure as Code integration

- Configuration management and environment reproducibility

Operational feedback and observability

- Monitoring, alerting, and post-deployment validation

6. Integration and Synthesis

End-to-end architectural cohesion

- Connecting architecture, asynchronous processing, and APIs
- Alignment of security, performance, and CI/CD

Strategic insights from industry experience

- Lessons learned, best practices, and scalability planning

© 2026 Faysal Aziz · Kuala Lumpur