**Course Title: Mastering Microservices with Spring Boot 3.x, Docker, and Kubernetes**

**Duration**: 10 Days  
**Format**: Instructor-led + Hands-on labs  
**Target Audience**:

* Java Developers familiar with Spring Framework
* System architects, DevOps engineers
* Prerequisite knowledge: Basic knowledge of Java, Spring Boot, and containerization concepts

**Course Prerequisites:**

* **Spring Boot Knowledge**: Understanding of dependency injection, Spring annotations, and Spring Boot application development.
* **Basic Containerization**: Docker and Kubernetes basic understanding, familiarity with kubectl commands.
* **DevOps Fundamentals**: Exposure to CI/CD, Docker, Kubernetes, and cloud-based deployment.

**Lab Requirements:**

* Laptop/PC with at least 32GB RAM.
* Pre-installed Docker Desktop (or Docker CLI), Kubernetes, Minikube, or a Kubernetes cloud provider like GKE (Google Kubernetes Engine).
* Java 21 and Spring Boot 3.x.
* IDE: IntelliJ IDEA or Eclipse.
* Internet connection for accessing DockerHub and other repositories.

**Day-wise Detailed Content**

**Day 1: Introduction to Microservices Architecture**

**Topics Covered**:

* Microservices: Overview and Benefits
* Monolithic vs Microservices Architecture
* Introduction to Spring Boot 3.x for Microservices
* Introduction to Docker and Kubernetes

**Lab**:

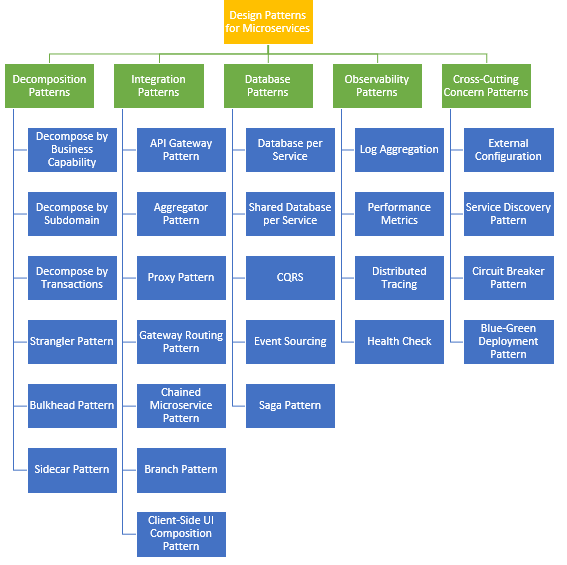
* Create a basic Spring Boot 3.x REST service.
* Containerize a Spring Boot service using Docker.

**Hands-On**:

* Set up a simple microservice and run it using Docker.

**Outcome**: Understand the basics of microservices architecture and the Spring Boot 3.x ecosystem.

Architectural Patterns



**Day 2: Building and Configuring Spring Boot Microservices**

**Topics Covered**:

* Service Layer Design Patterns (DTO, Repository, Service)
* Dependency Injection, Spring Annotations
* Externalized Configuration (YAML, .properties)

**Lab**:

* Create a multi-layered Spring Boot microservice with externalized configuration using Spring profiles.

**Hands-On**:

* Implement service, repository, and controller layers using Spring Boot 3.x.

**Outcome**: Develop a modular, maintainable microservice using best practices and Spring Boot features.

**Day 3: Microservices Communication**

**Topics Covered**:

* Synchronous Communication: REST, Feign Client
* Asynchronous Communication: RabbitMQ, Kafka
* Circuit Breakers: Resilience4J

**Lab**:

* Implement inter-service communication using Feign Client and Resilience4J.

**Hands-On**:

* Create two microservices communicating with each other and implement fault tolerance.

**Outcome**: Master synchronous and asynchronous communication patterns in a microservices environment.

**Day 4: Data Management in Microservices**

**Topics Covered**:

* Database per Service Pattern
* Data Consistency: Sagas and Event Sourcing
* Distributed Transactions (XA vs Saga)
* Spring Data JPA

**Lab**:

* Set up a MySQL/PostgreSQL database for two microservices, using Flyway for schema versioning.

**Hands-On**:

* Create a service that uses Spring Data JPA and manage multiple databases for different microservices.

**Outcome**: Understand and implement database strategies, event sourcing, and data consistency.

**Day 5: Containerizing Microservices with Docker**

**Topics Covered**:

* Dockerizing a Spring Boot Application
* Creating Docker Images and Docker Compose
* Multi-stage builds for optimized Docker images

**Lab**:

* Containerize Spring Boot microservices using Docker and orchestrate them using Docker Compose.

**Hands-On**:

* Build Docker images for multiple microservices, set up Docker Compose for local orchestration.

**Outcome**: Build, package, and orchestrate microservices using Docker.

**Day 6: Kubernetes for Microservices Orchestration**

**Topics Covered**:

* Introduction to Kubernetes (K8s) Architecture
* Deployments, Services, Pods, ConfigMaps, and Secrets
* Scaling and Auto-scaling in Kubernetes

**Lab**:

* Deploy microservices on a local Kubernetes cluster (Minikube or cloud provider).

**Hands-On**:

* Deploy and manage multiple Spring Boot services in a Kubernetes environment.

**Outcome**: Learn the basics of Kubernetes, deploy, and scale microservices.

**Day 7: Service Discovery, API Gateway, and Load Balancing**

**Topics Covered**:

* Service Discovery with Kubernetes (DNS and Service)
* API Gateway: Using Spring Cloud Gateway
* Load Balancing and Service Mesh (Istio Intro)

**Lab**:

* Implement API Gateway using Spring Cloud Gateway for routing and load balancing.

**Hands-On**:

* Set up a Spring Cloud Gateway to route traffic between microservices.

**Outcome**: Build an API Gateway and integrate service discovery.

**Day 8: Security and Authentication in Microservices**

**Topics Covered**:

* Securing Microservices: OAuth2 and JWT
* Spring Security 6.x Integration with Spring Boot 3.x
* Implementing Role-based Access Control

**Lab**:

* Secure a Spring Boot microservice using Spring Security and OAuth2.

**Hands-On**:

* Implement JWT-based authentication and secure endpoints in microservices.

**Outcome**: Secure microservices using industry-standard practices for authentication and authorization.

**Day 9: Logging, Monitoring, and Performance Management**

**Topics Covered**:

* Centralized Logging with ELK Stack (Elasticsearch, Logstash, Kibana)
* Monitoring with Prometheus and Grafana
* Tracing with Jaeger or Zipkin
* Health checks and Actuator

**Lab**:

* Set up Prometheus and Grafana to monitor microservices performance.

**Hands-On**:

* Implement centralized logging, monitoring, and tracing for Spring Boot microservices.

**Outcome**: Gain expertise in setting up monitoring, tracing, and logging systems for microservices.

**Day 10: Hands-On Case Study and DevOps Integration**

**Case Study**:

* Build and deploy a complete microservices-based application.
* Deploy the application in a Kubernetes cluster with logging, monitoring, API Gateway, and security in place.

**Topics Covered**:

* CI/CD Pipeline for Microservices (Jenkins, GitLab CI)
* Testing and Integration: Unit Tests, Contract Testing, and Consumer-driven Testing

**Lab**:

* Build a CI/CD pipeline for automated testing, build, and deployment using Jenkins.

**Hands-On**:

* Develop a microservice, deploy it through a pipeline, and test the full system.

**Outcome**: Successfully apply all concepts to build, deploy, monitor, and manage microservices in a DevOps environment.

**Course Outcome:**

By the end of this course, participants will be able to:

* Architect, build, and deploy microservices using Spring Boot 3.x.
* Utilize Docker and Kubernetes to containerize and orchestrate microservices.
* Implement communication patterns, data management, security, and logging in microservices.
* Set up monitoring, logging, and tracing tools for performance management.
* Build and integrate CI/CD pipelines for continuous deployment.