Spring Cloud Developer

Duration 3 day(s) (SPRING-CLOUD-DEVELOPER)

Microservices for developer

Official Training

Zenika exclusivity

Description

Cloud-native application architectures and processes are becoming a proven strategy to enable fast delivery of business value. Spring Boot and Spring Cloud are a powerful combination for building modern cloud-native application architectures that leverage industry battle-tested Spring ecosystem and third-party solutions to solve accompanying problems of scaling, availability and fault tolerance. The 3-day Spring Cloud Developer course provides participants with an in-depth coverage of cloud-native and microservices patterns using Spring Cloud and Netflix components to help solve challenges associated with running distributed, cloud-native applications over a microservices architecture.

Course Delivery Options

- Classroom
- Live Online

Goals

- Examine Problems of Distributed Systems and the associated Fault Tolerance patterns
- Examine how Distributed applications contribute to development and runtime of Cloud Native REST applications
- Examine how Distributed applications impact software systems fault tolerance
- Examine development impacts of implementing Spring Cloud solutions
- Contrast the benefits and trade-offs of Spring Cloud solutions
- Implement Spring Cloud solutions

Public

Developers interested in learning how to construct scalable and fault-tolerant cloud-native applications using the Spring Cloud family of projects

Prerequisites

- Completion of the Pivotal Platform Acceleration Lab for Developers (Java) course
- Completion of Pivotal'sSpring Core or Spring Boot course or Spring Boot experience

Structure

33% Theory, 67% Practice

Program

Introduction

- Spring Cloud Introduction
- Distributed Applications
- Spring Cloud Dependencies

Service Discovery and Client Load Balancing

- Service Registry, Load Balancing Patterns
- Eureka Service Registry
- Eureka Server REST Operations
- Service Discovery Clients
- Client Load Balancing
- Observability
- Configurable Load Balancing Algorithm

External Configuration and Distributed Trace

- External Configuration and Distributed Trace Patterns
- Spring Cloud Config Server
- Dynamically Refresh Application Configuration
- Distributed Updates
- Distributed Trace Collection and Visualization with Zipkin

Fault Tolerance – Health Checks

- Fault Tolerance Patterns
- Health Check Pattern
- Eureka Client Health Check
- Load Balancing Client Liveness Check

Retry, Backoff, Timeouts, Fallbacks

- Retry, Timeout and Fallback Patterns
- Retry with Load Balancing Client
- Slow Requests
- Socket Timeouts
- Fail-fast and Retry with Spring Cloud Config Server
- Timeouts with Hystrix

Circuit Breakers and Bulkheads

- Circuit Breaker and Bulkhead Patterns
- Circuit Breaker with Hystrix
- Bulkheads with Hystrix