# Course Design and Outline

## 

| Learning Program | Duration |
| --- | --- |
| **Microservices Intermediate** | 3 days |

## Pre-requisites: Trainees must be proficient with basics of Springboot, Micro-services architecture.

**Lab Setup:**

1. Windows 10 Pro with 16 GB RAM. Proxy-free internet, Admin rights to execute scripts
2. Spring tool Suite
3. Firefox and Chrome browsers
4. JDK 11: <https://www.oracle.com/technetwork/java/javase/downloads/index.html>
5. POSTMAN: <https://www.getpostman.com/downloads/>
6. Spring Tool Suite: <https://spring.io/tools>
7. Docker for Desktop:
   1. windows:<https://download.docker.com/win/stable/Docker%20Desktop%20Installer.exe>
   2. Mac OS: <https://download.docker.com/mac/stable/Docker.dmg>

## Pre-Reading Material

1. [https://docs.spring.io/spring-boot/docs/current/reference/html/](https://docs.spring.io/spring-boot/docs/current/reference/html/%20)
2. <http://spring.io/projects/spring-cloud>

# Learning program overview

|  |  |  |  |
| --- | --- | --- | --- |
| Course name and description | |  |  | | --- | --- | | **Intermediate Microservices using Spring Boot, Spring Cloud and Docker**  Microservices is an architectural style that structures an application as a collection of loosely coupled services, which implement business capabilities. The microservice architecture enables the continuous delivery/deployment of large, complex applications. It also enables an organization to evolve its technology stack.   |  | | --- | |  | | |
| Course learning objectives | * Understand the patterns typical of modern application architectures * Understand Domain Driven Design Concepts * Understand Distributed logging, Monitoring and Tracking * Learn how to build microservices with Docker * Microservices security |
| Effectiveness measure | * Assessments are built for all programs, team members mandatorily need to complete both pre and post assessments, else participation will be considered as void |

# Program outline

|  |  |  |
| --- | --- | --- |
| **Program Name** | **Intermediate Microservices using Spring Boot, Spring Cloud and Docker** | **Est. time (hours)** |
| **Instructional Strategy** | * In this course, we would first introduce the concept, followed by hands on activity in each of the lesson to ensure learners are able to apply the concept into the systems or application. * At the end of every learning objective we will present some quiz questions to participants. * The Assessment which is Post would be directly linked to the Learning Objectives of the program stated above. | |
| **Lesson** | **Topics** |  |
| **Microservices** | **Demystifying Microservices**   * The evolution of microservices * What are microservices? * Microservices – the honeycomb analogy * Principles of microservices * Characteristics of microservices * Challenges for a successful microservice architecture * Microservices examples * Microservices benefits * Microservices vs SOA * Relationship with other architecture styles * Microservice use cases | 4 |
| **Cloud Native Spring Boot** | **Building Microservices with Spring Boot**   * The Spring Boot configuration * REST APIs 101 * Implementing a Simple REST API with Spring boot * Cloud Native Application * Using “Twelve-Factor App” Style Configuration * Read External Configuration * Support multiple environments using Spring Profiles * Implementing Spring Boot Controllers and Exception Handling * Implementing Spring Boot Bean Validation * Implementing Unit testing and Mock Testing for Spring Boot * Implementing Spring Boot Developer Tools | 4 |
| **Spring Boot Deep Dive** | * Rest API and Richardson Maturity Model * Implementing Spring Boot HAL Browser * Implementing Spring Boot JPA using H2 * Implementing Spring Data REST using H2 * Documenting microservices with OpenAPI * Spring Boot actuators | 4 |
| **Spring Cloud** | * Overview of Spring cloud and its modules | 4 |
| **Spring Cloud Eureka** | * Why DiscoveryServer? * Client side load balancing * How to Create a Eureka Server using @EnableEurekaServer or @EnableDiscoveryServer * How to register Eureka Client using @EnableEurekaClient or @EnableDiscoveryClient * How to look up a service from Eureka Server   **LAB : Using Eureka Server and Client LAB : Configuring Eureka cluster** |
| **Spring Cloud Ribbon** | * Why ClientSide Load Balancing? * What is ribbon and how to use spring ribbon? * Using LoadBalanced RestTemplate * Configuring Retries * Using different Load balancing algorithms   **LAB : Implementing Client side load balancing using ribbon** |
| **Spring Cloud Feign** | * What is a Feign? * Implementing REST Clients in declarative approach * Configuring Fallbacks using feign * Hystrix configurations when using feign   **LAB : Implementing REST Client using Feign** |
| **Spring Cloud Hystrix** | * What is a circuit breaker? * Using Hystrix with Ribbon * Various Hystrix configurations * Hystrix Dashboard for monitoring   **LAB : Using Hystrix as circuit breaker**  **LAB : Using Hystrix Dashboard** | 2 |
| **API Gateway and ZUUL** | * Why API Gateway? * What is ZUUL ? * Disabling access to some services at zuul level * Using Zuul Filters   **LAB : Using ZUUL to enable Proxying microservices** |
| **Spring Sleuth and Zipkin** | * What is Tracing? * What is zipkin server?   **LAB : Using sleuth to enable tracing** |
| **Spring Security and OAuth** | * Spring Security Architecture * Authentication and Authorization * Authentication Manager and Access Decision Manager * Http Security * Oauth and Spring Security * Authorization Server and Resource Server * Oauth Token generation and its use   **Lab: Using Oauth between 2 microservices using In memory credential store.** | 4 |
| **Microservices Patterns** | * + Ambassador   + Anti-corruption layer   + Backends for Frontends   + Bulkhead   + Gateway Aggregation   + Gateway Offloading   + Gateway Routing   + Sidecar   + Strangler   Further:   * + Patterns @ microservices.io | 4 |