Faysal Aziz

Developing Cloud-Native Applications with Microservices Architectures

FA-0169 · five-day

Software Development · DevOps, Cloud & Infrastructure

Introduction

In the rapidly evolving landscape of software development, the need to create scalable, flexible, and resilient applications is more critical than ever. This five-day intensive course, "Developing Cloud-Native Applications with Microservices Architectures using Python," is designed to equip experienced programmers with the knowledge and skills required to develop and deploy cloud-native applications using microservices architectures. Leveraging Python as the primary programming language, this training will delve into the world of microservices and their underlying principles, ensuring that participants can build applications that are easily adaptable to emerging technologies and market needs.

Throughout this comprehensive course, participants will engage in hands-on exercises and real-world projects, empowering them to build cloud-native applications with confidence. By the end of the training, they will be well-versed in microservices architecture, containerization, API design, deployment, and monitoring, all with the power and flexibility of Python.

Learning outcomes

Upon successful completion of this course, participants will be able to:

- Understand the fundamentals of microservices architecture and its advantages over monolithic applications.
- Design and implement microservices using Python and popular frameworks.
- Utilize containerization technologies, such as Docker, for microservices deployment.
- Design and implement RESTful APIs for communication between microservices.
- Implement service discovery, load balancing, and fault tolerance in microservices architecture.
- Monitor and manage microservices applications using logging, metrics, and tracing.

- Secure microservices and their APIs using authentication and authorization techniques.

Prerequisites

- Proficiency in programming language.
- Familiarity with software development principles and practices.
- Basic understanding of web application development and RESTful APIs.
- Familiarity with version control systems, such as Git.

Training outline

1. Day 1: Introduction to Microservices Architecture and Design Principles

1.1. Introduction to microservices architecture

- Definition and characteristics

Benefits and challenges

- 1.2. Monolithic vs. microservices architecture
- Comparison of architectural styles

When to choose microservices

- 1.3. Microservices design principles
- Single responsibility principle
- Loose coupling and high cohesion

Bounded contexts and domain-driven design

- 1.4. Decomposing monolithic applications into microservices
- Identifying service boundaries

Refactoring strategies

- 1.5. Domain-Driven Design (DDD) for microservices
- Strategic and tactical design patterns

Implementing aggregates, entities, and value objects

- 1.6. Hands-on: Designing a microservices architecture for a sample application

2. Day 2: Implementing Microservices with Python and Frameworks

2.1. Introduction to Python-based microservices frameworks (FastAPI, Flask, Django)

- Overview and comparison of frameworks

Selecting the right framework

- 2.2. Developing microservices with FastAPI
- Setting up a FastAPI project
- Implementing routes and handling requests

Validating input and handling errors

- 2.3. Developing microservices with Flask
- Setting up a Flask project
- Implementing routes and handling requests

Validating input and handling errors

- 2.4. Developing microservices with Django
- Setting up a Django project
- Implementing views and handling requests

Validating input and handling errors

- 2.5. Hands-on: Implementing microservices for the sample application using a chosen Python framework

3. Day 3: Containerization and Deployment of Microservices

3.1. Introduction to containerization and Docker

- Benefits of containerization

Docker architecture and components

- 3.2. Creating Dockerfiles for Python microservices
- Writing a Dockerfile

Building and running Docker images

- 3.3. Deploying microservices using Docker Compose
- Docker Compose basics

Creating a Docker Compose file for the sample application

- 3.4. Container orchestration with Kubernetes

- Kubernetes architecture and components

Deploying and managing microservices with Kubernetes

- 3.5. Hands-on: Containerizing and deploying microservices for the sample application

4. Day 4: Communication, Service Discovery, and Load Balancing

4.1. RESTful API design principles

- Resource identification and naming
- Request and response formats

Versioning and error handling

- 4.2. Implementing RESTful APIs for microservices in Python
- Creating API endpoints
- Handling HTTP methods and status codes

JSON serialization and deserialization

- 4.3. Service discovery patterns and tools (Consul, Eureka, etcd)
- Client-side and server-side discovery
- Service registries and health checks

Implementing service discovery with popular tools

- 4.4. Load balancing and fault tolerance (HAProxy, NGINX, Envoy)
- Load balancing strategies
- Health checks and circuit breakers

Proxy and gateway configurations

- 4.5. Hands-on: Implementing APIs, service discovery, and load balancing for the sample application

5. Day 5: Monitoring, Security, and Best Practices

5.1. Monitoring microservices using logging, metrics, and tracing

- Importance of monitoring in microservices
- Structured logging and log aggregation
- Collecting and analyzing metrics with tools like Prometheus

Distributed tracing with tools like Jaeger and Zipkin

- 5.2. Security concerns in microservices architecture

- Attack vectors and vulnerabilities

Securing communication and data storage

- 5.3. Implementing authentication and authorization in microservices
- Token-based authentication with JWT
- OAuth2 and OpenID Connect

Role-based and attribute-based access control

- 5.4. Best practices for microservices development and deployment
- Continuous integration and continuous deployment (CI/CD)
- Resilience and fault tolerance patterns

Designing for scalability and maintainability

5.5. Hands-on: Adding monitoring and security to the sample application

- 5.6. Course recap and next steps
- Review of key concepts and techniques
- Further learning resources and recommendations

© 2026 Faysal Aziz · Kuala Lumpur