En-lighten Technology Ltd

Building Microservices with Go: Design, Implementation and Test

Overview

This course is for developers who want to learn how to design, test and observe microservices using industry best practices using Go. The course uses an API design first approach. Design principles and techniques are introduced for developing efficient, consistent API's specified using Open API. How the specification can be used to drive implementation of both the microservice and consumers whilst ensuring both align with the specification is then covered. Automated testing tools and strategies are introduced that illustrate how to ensure that the API is thoroughly tested to a level of confidence ready for deployment to production. Microservice reliability, performance and observability are also covered.

Following the approach presented in the course will result in a microservice that has a clearly specified API that adheres to design standards and is tested thoroughly both functionally and non-functionally.

Attendees will learn how to

- Design microservices using an API first approach and test using industry best practices
- Use OpenAPI for specifying designs
- Generate server stubs and models from the specification
- Validate an API meets organization design rules automatically
- Apply an API first approach to existing microservices
- Write contract tests that verify request responses against the specification
- Generate client libraries for consuming microservices
- Develop and publish API documentation
- Define KPI and SLO for microservices

Audience and Recommended Background

The course is a hands-on course with practical exercises. The focus is totally on API design with Open API and testing the API. Go is the programming language used in the code examples and programming exercises, but the principles are applicable to any programming languages.

Course Approach

Taught material will be delivered in short sessions and will be followed by the opportunity to practice what has been taught and reflect on how to put it into practice.

Course Duration: 5 days

En-lighten Technology Ltd

Course Content

Introduction to Microservices

- What are Microservices?
- Key benefits
- Challenges of Microservices
- No silver bullet

Modeling Services

- Characteristics of a good service
- Introducing the bounded context
    - Shared and hidden models
    - Modules and services
    - Avoiding premature decomposition
- Communication and the consistent use of language
- Deciding and maintaining technical boundaries
- Asynchronous and synchronous service communication

Decomposing a Monolith

- Why split a monolith?
    - Pace of change
    - Team structure
    - Security
    - Technology
- Starting point for decomposition
- Working with seams
- Working with tangled dependencies
- Transactional boundaries
- Managing shared databases
- Cost of change

Introducing API Design

- What is an API?
- Why design matters
- The elements of API design

API Design

- Designing software interfaces
- Identifying goals
- Synchronous and asynchronous APIs
- Avoiding leaking implementation
- Identifying resources
- Data design
- Design trade-offs

Specifying an API with Open API

- Structure of an OpenAPI document

En-lighten Technology Ltd

- Describing API resources and actions
- API data and JSON schema
    - Query parameters
    - Body parameters
    - Responses
    - Reusing components
    - Path parameters
- Maintaining backward compatibility
    - Breaking changes in output data
    - Breaking changes in input data and parameters
    - Success and error feedback
- Versioning an API
- Designing for extensibility
- Network efficiency
- Validating an Open API specification
- Design rule validation for consistency of API's
- Asynchronous calls and callbacks

Generating Service Models and Stubs

- Why automatic code generation?
- Generating stubs and models from the Open API specification
- Customising output

Working with Existing Microservices

- Steps in moving from code first microservices to API first approach
- Extracting the specification from an existing implementation
- Strategies for managing API changes
- Aligning with design rules and standards
- Generating service stubs and models
- Providing a bridge between generated service, models and existing code

Consumer Driven Contract Testing

- What is consumer driven contract testing?
- A first example using Pact
- Matching requests
- HTTP contracts
- Stateful contracts
- Messaging contracts
- Sharing Pacts with Pact broker
- Maintaining backward compatibility
- Ensuring tests detect contract breaking changes

Observability

- The pillars of observability
    - Tracing
    - Metrics
    - Logging
- Making a service observable

En-lighten Technology Ltd

- RED and USE metrics

Design for Reliability and availability

- Stateless services
- Idempotency
- Upgradable service design
- Patterns for managing failure
- Scalable services

Non-Functional Testing

- Defining KPI’s and SLO's
- Documenting KPIs and resources using Open API extensions
- Load testing to verify KPI’s
- Stress testing

Documenting API Specifications

- Automated document generation with Swagger CodeGen
- Live documentation
- Publishing documentation to an API portal

Consuming an API

- Generating client libraries with Swagger Codegen
- Customising generated code with templates
- Making use of server stubs