

Go Programming Workshop 2019

A programming workshop for building highly scalable backend systems and microservices with Go ecosystem

Go Programming workshop provides highly intensive training on both Go programming

language and the Go ecosystem, which is packed with real-world guidance using code examples. This course lets the audience build real-world backend systems and Microservices using the Go ecosystem. Attendees can dive into production systems after completing this training. This course will also provide architecture guidance for building scalable backend systems in Go.

**Course Duration:** 5 Days **Prerequisites:** Nil **Learning Objectives:**

* Master on Go programming language and its core fundamentals.
* Learn in-depth on Go functions, including functional programming.
* Write highly maintainable and extensible systems with Go's type system.
* Write high-performance, concurrent systems using Go’s concurrency primitives and various concurrency patterns.
* Build real-world web apps and RESTful APIs in Go.
* Persistence with databases such as MongoDB, CockroachDB, InfluxDB and SQL databases.
* Testing Go applications.
* Build highly scalable distributed systems and microservices with gRPC and NATS Streaming Server.

## Course Outline:

|  |  |
| --- | --- |
| **Module No.** | **Module Name and Contents** |
| 1. | **Introduction to Go**   * Introduction to Go * The design philosophy of Go * Go Ecosystem * Setting up Go workspace |
| 2. | **Go Language Fundamentals**   * Core language fundamentals * Functions * Arrays * Slices * Maps * Defer, Panic and Recover * Error handling |
| 3. | **Go Package System and Tooling**   * A deep dive into package ecosystem in Go * Write packages * Go Tools * Using Go standard library packages * Using third-party packages * Managing package dependencies |
| 4. | **User-Defined Type System**   * A deep dive into Go’s type system * Introduction to Structs and Interfaces * Adding Behaviors to Structs * Value Receivers and Pointer Receivers * Using composition pattern for building data model for Go apps * Using interface for building extensible and testable apps |
| 5. | **Concurrency Programming**   * Concurrency in Go * Goroutines * Channels * Unbuffered Channels * Buffered Channels * Channel Select * Advanced Concurrency patterns |
| 6. | **Mutexes**   * Preventing data race conditions with Mutexes |
| 7. | **HTTP Programming**   * A deep dive into Go’s http package * ServeMux and Handler * Creating custom handlers * Extending Go’s http package by using third-party packages * Routing using Gorilla Mux * Writing HTTP middleware * Building RESTful APIs |
| 8. | **Persistence with Databases** |

|  |  |
| --- | --- |
|  | * Working with NoSQL Databases: MongoDB * Working with SQL Databases * Working with Time-Series Databases: InfluxDB * Working with global-scale distributed databases: CockroachDB |
| 9. | **An Architectural Guidance for Building Real-World Backend Systems**   * Demonstrating an end-to-end RESTful API application with Go idioms and best practices * Organizing real-world applications into various packages * Exploring various third-party packages * Exploring various components for building real-world web apps * Implementing authentication for APIs using JWT * Passing values between HTTP handlers * Best practices for writing HTTP middleware * Best practices for building real-world applications * Organizing real-world applications into packages * Dependency management of packages * Writing make files * Deployment strategy for running Go applications |
| 10. | **Testing Go Applications**   * Writing unit tests * Writing Benchmark tests * Testing HTTP applications * Test-Driven Development (TDD) and Behavior-Driven Development (BDD) * Writing BDD-styled testing * A deep dive into Ginkgo BDD test framework * Mocking Go interfaces * GoMock - A mock framework for Go |
| 11. | **Debugging and Profiling Go Programs**   * Debugging Go programs * Profiling Go programs * Tracing Go programs |
| 12. | **Package Context**   * Using context on http applications * Provides deadlines, cancelation signals, and other request-scoped values across API boundaries and between processes |
| 13. | **Building Next-Generation APIs using gRPC and Protocol Buffers**   * Introduction to gRPC * Introduction to Protocol Buffers * Writing APIs with gRPC and Protocol Buffers * Writing gRPC Interceptors |
| 14. | **Building Highly Scalable Distributed Systems and Microservices with NATS Streaming Server**   * Introduction to NATS and NATS related messaging patterns * Introduction to NATS Streaming Server * Building Event-Driven Microservices with NATS Streaming Server |

**Day Wise Training Plan**

**Modules**

**Days**

|  |  |
| --- | --- |
| Day 1 | Module 1, 2, 3 |
| Day 2 | Module 4, 5 |
| Day 3 | Module 6, 7, 8 |
| Day 4 | Module 9, 10, 11 |
| Day 5 | Module 12, 13, 14 |

# Technology Stack



## Important Notes

* This training program is designed to let developers build real-world Go applications. Hence the course will be using code examples and real-world guidance instead of using PPT slides and focusing on basic syntax.
* The trainer will also provide architecture guidance for building highly scalable real- world Go applications.
* The trainer also provides architecture guidance and mentoring for building real- world applications in Go after completing the training program.