Course Icon

Go At Scale: Concurrency, Performance, and Design

- **3 Days**
- **Lecture and Hands-on Labs**

Course Overview

Go at Scale: Concurrency, Performance, and Design helps intermediate Go developers move beyond the basics to build reliable, high-performance services. Through lectures and hands-on labs, students explore idiomatic Go design, concurrency with goroutines and channels, structured error handling, testing, and performance optimization, gaining practical skills for building scalable applications and systems.

Review this course online at https://www.alta3.com/courses/go-idiomatic

Who Should Attend

- Backend Software Engineers
- Platform Engineers
- Site Reliability Engineers (SREs)
- DevOps Engineers

What You'll Learn

- Write idiomatic Go code using interfaces, composition, and effective package design.
- Implement concurrency using goroutines, channels, and common coordination patterns.
- Diagnose and resolve concurrency issues such as race conditions and goroutine leaks.
- Apply structured error handling patterns suitable for production Go services.
- Test, benchmark, and profile Go applications to identify performance bottlenecks.
- Use generics appropriately to build reusable and maintainable components.

Outline

Up and Running with Go

- **Lecture + Lab:** Getting Started with Go

Idiomatic Go - Getting Started

- **Lecture + Lab:** Let gofmt Decide
- **Lecture + Lab:** Package Names Should Be Short, Lowercase, and Boring
- **Lecture + Lab:** MixedCaps Everywhere - No Underscores in Go Names
- **Lecture + Lab:** Semicolons, Newlines, and Why Braces Matter
- **Lecture + Lab:** Doc Comments & Intentional Commentary
- **Challenge:** CHALLENGE - Zero Values & Constructors

Go is Not Object Oriented

- **Lecture + Lab:** Why Go Rejects Classical OOP

Idiomatic Go - Naming

- **Lecture + Lab:** Getters Without “Get” - Export via Capitalization
- **Lecture + Lab:** One-Method Interfaces - Names and Canonical Signatures

Idiomatic Go - Structure

- **Lecture + Lab:** If, Initialization, and Eliminating Else
- **Lecture + Lab:** Redeclaration vs Reassignment

Idiomatic Go - Common Misconceptions

- **Lecture + Lab:** Looping - Unified for, range, and Idiomatic Patterns
- **Lecture + Lab:** Type Switches
- **Lecture + Lab:** Multiple Return Values
- **Lecture + Lab:** Named Result Parameters
- **Lecture + Lab:** Defer to Deterministic Cleanup, LIFO Execution, and Resource Management

Idiomatic Go - Methods and Interfaces

- **Lecture + Lab:** Methods - Pointer Receivers vs Value Receivers
- **Lecture + Lab:** Interfaces - Behavior Over Structure
- **Lecture + Lab:** Conversions, Type Assertions, and Interface Behavior
- **Lecture + Lab:** Blank Identifier - Intentional Discarding Without Sloppiness
- **Lecture + Lab:** Unused Imports and Variables - Why the Compiler is Strict
- **Lecture + Lab:** Import for Side Effect - Trigger init on Purpose
- **Lecture + Lab:** Interface Checks - Compile-Time Guarantees, Run-Time Assertions, and Idiomatic Verification
- **Lecture + Lab:** Embedding - Borrowing Behavior Without Inheritance
- **Lecture + Lab:** Generality - Export Interfaces, Not Implementations

Idiomatic Go - Errors and Handling

- **Lecture:** Panic
- **Lecture:** Defer
- **Lecture:** Recover
- **Lecture + Lab:** Generating and Handling Errors
- **Lecture + Lab:** Errors - Design, Inspection, and Recovery
- **Lecture + Lab:** Panic - When it is Appropriate
- **Lecture + Lab:** Recover - Catching Panic

Idiomatic Go - Testing & Benchmarking

- **Lecture:** Benchmark Testing
- **Lecture + Lab:** Table-Driven Tests
- **Lecture + Lab:** Subtests and Parallel Tests

Idiomatic Go - Channels and Concurrency

- **Lecture + Lab:** Concurrency with Goroutines
- **Lecture:** Channels
- **Lecture + Lab:** Channels
- **Lecture + Lab:** Concurrency - Share Memory by Communicating
- **Lecture + Lab:** Channels of Channels - Request and Response Patterns
- **Lecture + Lab:** Parallelization with Goroutines and Channels
- **Lecture + Lab:** Contexts and Cancellation
- **Lecture + Lab:** A Leaky Buffer Using Channels

Idiomatic Go - Performance and Memory

- **Lecture + Lab:** Garbage Collection Fundamentals
- **Lecture + Lab:** Profiling and Performance Optimization with pprof

Idiomatic Go - Patterns in Practice

- **Lecture + Lab:** SQL and SQL-like Databases
- **Lecture + Lab:** Writing a Minimal Terraform Provider
- **Lecture + Lab:** Structuring Large Go Applications

Appendix

- **Lecture + Lab:** Building an Idiomatic Go Web Server with Templates
- **Lecture + Lab:** Generating and Handling Errors
- **Lecture + Lab:** Interfaces
- **Lecture + Lab:** Sorting
- **Lecture + Lab:** Allocation with new and The Zero-Value-Is-Useful Principle
- **Lecture + Lab:** Constructors and Composite Literals
- **Lecture + Lab:** Allocation with make - Understanding Slices, Maps, and Channels
- **Lecture + Lab:** Arrays - Value Semantics, Type Identity, and Why Slices

Are Preferred
- **Lecture + Lab:** Slices - Reference Semantics, Capacity, Growth, and Two-Dimensional Design
- **Lecture + Lab:** Maps - Reference Semantics, Zero Values, and the Comma-Ok Idiom
- **Lecture + Lab:** Printing - Formatting, Stringers, and Variadic Functions
- **Lecture + Lab:** append - Variadic Elements, Slice Growth, and the ... Operator
- **Lecture + Lab:** Initialization - Constants, iota, Typed Units, and Idiomatic String Formatting
- **Lecture + Lab:** Variables - Runtime Initialization and Idiomatic Usage
- **Lecture + Lab:** init - Controlled Startup
- **Lecture + Lab:** Helpful Links
- **Lecture:** Glossary

Prerequisites

- Go Programming Essentials (https://alta3.com/courses/golang)