

Golang Development

Course Duration ~60 Hours · 8 Weeks · Live Online · Build-Along · Deployed Services and Tools

Curriculum reviewed: July 2026 — recent Go releases changed production behaviour; this syllabus tracks the current line

Concurrency

Race Detection

gRPC

Profiling & pprof

Cloud-Native

CLI Tools

In short: Learn the language in two weeks. Spend the rest of the course on the part that actually gets you hired — concurrency that's correct, services that stay up, and binaries that start in milliseconds. Go is for infrastructure and high-throughput services: a small language compiled to a single static binary, with concurrency built in and a runtime designed for containers — Docker, Kubernetes, Terraform and Prometheus are all written in it. The "boring language" pitch is a feature: deliberately small, one obvious way to do most things, self-formatting, backward compatible for over a decade. Most Go courses stop at goroutines and a CRUD API; this one goes where recent releases concentrated — container-aware scheduling (why explicit CPU overrides in deployment configs are usually wrong to keep now), deterministic testing of concurrent and time-dependent code with a virtualised clock (no sleeps, no flakes), garbage collector behaviour and memory limits, flight recording that captures what happened before an incident, and a new-generation JSON package. Across 9 modules you ship a load-tested standard-library API, a bounded concurrent pipeline, a race-hunting workshop with detector evidence, a deterministic test suite, a gRPC service pair, a cross-compiled CLI tool, a Kubernetes deployment and an observability build-out — closing with a defended capstone. Choose Go when throughput, latency, memory footprint or operational simplicity matter more than ecosystem breadth; our Node.js, FastAPI and .NET courses cover general application backends. The AI question, sharpened: Go's syntax is small enough that AI writes it fluently — and Go's hard problems are exactly the ones AI gets wrong: data races, goroutine leaks, missing cancellation, unbounded concurrency, channel deadlocks. They fail at 3am under load, not in a demo — and Go is increasingly the language people write AI infrastructure in, not just the language AI writes.

Duration	8 weeks · ~60 hours · live + hands-on	Format	Live online · build-along · deployed services and tools
Prerequisite	Programming experience in any language. No Go, systems or concurrency background required.	Curriculum	9 modules — the language fast, then concurrency, correctness, services, diagnostics and delivery
Portfolio	8 deployed, production-style services and tools plus a defended capstone	Ideal for	Fresh IT graduates, backend developers adding a systems language, and DevOps/SRE engineers who read Go daily

Why Learn Go When AI Writes It Fluently?

Go's syntax is small enough that AI writes it fluently — and Go's hard problems are exactly the ones AI gets wrong: data races, goroutine leaks, missing context cancellation, unbounded concurrency, channel deadlocks. None of these fail loudly in a demo; they fail at 3am under load. The value is concurrency correctness, architecture and operational judgment. You'll use AI tooling as an accelerator you review — and Go's own ecosystem now ships an official SDK for building agent-integration servers: Go is increasingly the language people write AI infrastructure in, not just the language AI writes.

Course Highlights

- Every module leads with what you build and ship, not what you "learn about"
- Concurrency as the core of the course — goroutines, channels, context, and the failure modes that define production Go
- Correctness as a discipline: the race detector in your routine workflow, deterministic clock-driven tests
- Standard library first — frameworks compared honestly, dependencies earned
- Production diagnostics: pprof, GC behaviour and memory limits, flight recording
- Container-aware runtime behaviour — and why explicit CPU overrides are usually wrong now
- Tiny static images, cross-compiled CLI tools, Kubernetes-ready delivery
- AI tooling as an accelerator you review — and Go as the language of AI infrastructure

Hands-on Projects & Portfolio

- **High-Throughput JSON API** — standard library only, load-tested with measured latency percentiles
- **Concurrent Worker Pipeline** — bounded concurrency, context cancellation and graceful shutdown
- **Race-Hunting Workshop** — find and fix real data races and goroutine leaks, with detector and profiler evidence
- **Deterministic Test Suite** — time-dependent concurrent code tested with a virtualised clock: no sleeps, no flakes
- **gRPC Microservice Pair** — protocol buffers, deadlines, retries and circuit breaking
- **Production CLI Tool** — distributed as a single cross-compiled binary
- **Container & K8s Deployment** — minimal image demonstrating correct runtime CPU behaviour
- **Observability Build-Out** — structured logs, metrics, tracing and flight recording wired to an incident workflow
- **Capstone** — production service, deployed, profiled, monitored and defended on architecture

Tools & Stack

Go · goroutines & channels · context · gRPC · Protocol Buffers · PostgreSQL · sqlc · Redis · Docker · Kubernetes · OpenTelemetry · Prometheus · Grafana · testcontainers · golangci-lint · pprof · Cobra · NATS or Kafka · Terraform · GitHub Actions · Git

Career Outcomes

Go occupies a specific and well-rewarded niche — the infrastructure layer. The tools that run modern cloud platforms are written in it, which means Go skills cluster around platform, infrastructure and SRE work rather than general application development. The job market is smaller than for JavaScript or Python but correspondingly less crowded, Go's decade-plus backward-compatibility record means these skills date very slowly, and the language is small enough for an experienced developer to become productive quickly. This course targets:

- Backend Engineer (Go) · Platform Engineer · Infrastructure Engineer · Site Reliability Engineer
- Cloud-Native Developer · Microservices Developer · DevOps Engineer with development depth · Systems Engineer

Who This Course Is For

- Final-year students and fresh IT graduates targeting backend or platform roles
- Backend developers from Python, Node, Java or .NET adding a systems-level language
- **DevOps, SRE and platform engineers who read Go daily and want to write it properly** — a large and highly motivated audience
- Engineers building microservices where latency, throughput or memory cost matters
- Developers who want to contribute to or extend cloud-native and Kubernetes ecosystem tooling
- Anyone who has written Go by pattern-matching and never learned concurrency properly

Module 1: Go Fast

7 Hours

YOU SHIP

- A working CLI utility and library package — idiomatic, formatted, modularised — proving the whole language in one intensive module

TOPICS

- Types, structs, methods, interfaces and implicit satisfaction; slices and maps and their gotchas
- Pointers and value semantics; error values as ordinary returns; packages and modules; the toolchain
- The idioms that make Go code look like Go

Assessment: Quiz + Lab

Module 2: Idiomatic Go & Error Handling

6 Hours

YOU SHIP

- A refactor into consumer-defined interfaces and properly wrapped, inspectable errors — with generics applied only where they remove real duplication

TOPICS

- Composition over inheritance; small interfaces defined by the consumer
- Error wrapping and inspection; sentinel versus typed errors; when panic is and isn't appropriate
- Generics used judiciously; iterators; the standard library as the first place you look

Assessment: Quiz + Lab

Module 3: Concurrency — the Core of the Course**9 Hours****YOU SHIP**

- A concurrent worker pipeline with bounded concurrency, fan-in/fan-out, context cancellation propagated end to end, and graceful shutdown that actually drains

TOPICS

- Goroutines and the scheduler; channels and select; sync primitives and when a mutex beats a channel
- context for cancellation, deadlines and request scoping; worker pools and bounded concurrency; fan-in and fan-out; pipelines; graceful shutdown
- The failure modes that define Go in production: goroutine leaks, unbounded concurrency, deadlocks

Assessment: Quiz + Lab

Module 4: Concurrency Correctness**7 Hours****YOU SHIP**

- The race-hunting workshop: real data races and goroutine leaks found and fixed with detector evidence — plus a deterministic, virtual-clock test suite for time-dependent code with no sleeps and no flakes

TOPICS

- The race detector as a routine part of your workflow; data races and memory visibility
- Deterministic testing of time-dependent and concurrent code with a virtualised clock
- Stress and fuzz testing; reasoning about a concurrent design before you write it

Assessment: Quiz + Lab

Module 5: Building HTTP Services**7 Hours****YOU SHIP**

- A high-throughput JSON API on the standard library — timeouts at every layer, structured errors, graceful shutdown — load-tested with measured latency percentiles

TOPICS

- The standard library router and modern routing patterns; middleware; request lifecycle and context propagation
- JSON handling; validation; structured error responses; timeouts at every layer; graceful shutdown
- An honest look at the popular frameworks — and when the standard library is enough

Assessment: Quiz + Lab

Module 6: Data, Integration & Distributed Communication**7 Hours****YOU SHIP**

- A gRPC microservice pair with protocol buffers, deadlines, retries and circuit breaking — backed by a pooled, transactional data layer and a resilient outbound HTTP client

TOPICS

- Database access with the standard interface and lightweight helpers; connection pooling; transactions; migrations; caching with Redis
- gRPC and protocol buffers for service-to-service calls; message consumption patterns
- HTTP clients with retries, timeouts and circuit breaking; rate limiting

Assessment: Quiz + Lab

Module 7: Testing & Quality

6 Hours

YOU SHIP

- A fully tested service — table-driven tests, interface-based fakes without a mocking framework, containerised database tests, benchmarks — gated by lint and vet in CI

TOPICS

- Table-driven tests as the house style; subtests; interfaces for testability without a mocking framework
- Integration testing with containers; HTTP handler testing; benchmarks
- Coverage that means something; linting and vetting; CI gates

Assessment: Quiz + Lab**Module 8: Performance & Production Diagnostics**

6 Hours

YOU SHIP

- The observability build-out: a slow, allocation-heavy service profiled and fixed with pprof evidence, wired with structured logs, metrics, tracing and flight recording into an incident workflow

TOPICS

- Profiling CPU, memory, blocking and mutex contention; escape analysis and allocation reduction
- Garbage collector behaviour and tuning memory limits; flight recording for post-incident tracing
- Structured logging; metrics and distributed tracing with OpenTelemetry; load testing

Assessment: Quiz + Lab**Module 9: Cloud-Native Delivery & Capstone**

5 Hours

YOU SHIP

- A minimal static container image deployed to Kubernetes demonstrating correct runtime CPU behaviour — and your capstone: one production-grade service, deployed, profiled, monitored and defended

TOPICS

- Building tiny static container images; container and Kubernetes-aware runtime behaviour — why explicit CPU overrides are usually wrong now; health checks and probes
- Configuration and secrets; cross-compilation; production CLI tools; deployment and CI/CD
- Infrastructure and platform tooling, including agent-integration servers; capstone design, build, deploy and defense

Assessment: Capstone Project + Review