# Go Basic to Advanced

Getting experienced software engineers prepared for building production-ready cloud-native applications

## Objectives

- Understanding Go's syntax
- Working with Go's concurrency and understanding advanced patterns
- Working with Go's inheritance - interfaces and composition
- Structuring and writing testable Go code
- Working with and writing ReSTful applications

## Prerequisites

- At least one year of active programming experience
- Familiarity with client/server Architecture or ReST
- Familiarity with infrastructure tools/platforms like:
  - Linux & Bash
  - Docker or Kubernetes

## Agenda

### Day 1

- Why Go?
- `go build`
  - binaries ( `GOOS` & `GOARCH` / `CGO_ENABLED` )
- `go run`
- Introduce Goroutines (Power of Go)
  - sync.WaitGroup
  - GOMAXPROCS
  - Understanding the Go scheduler
- Syntax overview!
  - Packaging & Imports
    - Directory layout
  - Variables and functions
    - Multiple Returns
    - Zero values
    - Type conversions
    - Constants
    - User-defined types
  - Flow control
    - For
    - if

        - switch
        - range

- Data Structures
    - Arrays
    - Slices

## Day 2

- Data Structures continued
    - Structs
    - Maps
    - Custom: Linked List
- More types
    - Pointers
    - Struct Fields
        - Exporting fields
    - Pointers & structs
- Functions revisited
    - Multiple returns
    - Named return values
    - Variadic functions
- `defer`, `panic` and `recover`
- Higher order functions
    - Functions
    - Understanding Stack vs Heap memory
    - What does the Garbage Collector exactly collect?
- Methods and interfaces
    - Methods and pointer indirection
    - Receiver functions
        - pointer receivers vs value receivers

## Day 3

- `error`
    - Errors in Go 1.13
- interfaces continued
    - implicit implementation
    - nil interface
    - empty interface
- Inheritance in Go
    - Struct Embedding

    - Interface embedding

- Unit Testing & Dependency Management
    - Writing and Running Unit Tests
    - Working with `go mod`
    - Writing assertions using `stretchr/testify`
- Concurrency: Goroutines, Parallelism
    - Concurrency with goroutines
    - Concurrency and Parallelism
- Concurrency: Sync, WaitGroup, Mutexes
    - Sync, WaitGroup
    - Mutexes
    - Deadlocks
    - RW Mutexes
- Concurrency: Handling Race Conditions
    - Example of Race Condition
- Concurrency: Channels
    - Channels
    - Channel Direction
    - Closing Channels
    - Range Over Channels
    - Channels - Select
    - Timeouts
- Concurrency: `context` package
- Writing a ReSTful API
    - Introducing the `gorilla` toolkit