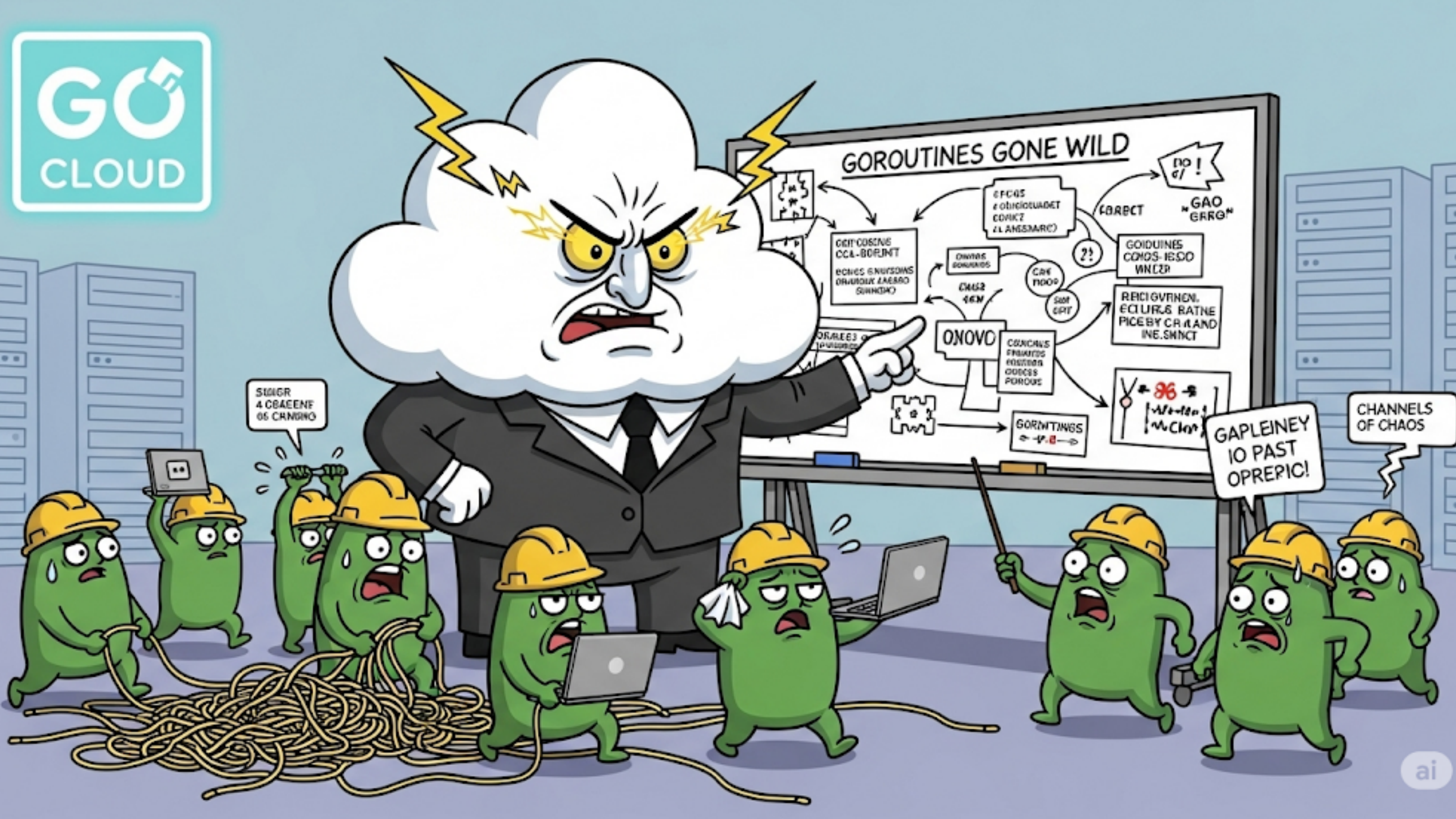
Cloud Computing Technologies

DAT515 - Fall 2025

Introduction to Go Programming

Prof. Hein Meling

University
of Stavanger


GO
CLOUD
GOROUTINES GONE WILD
CHANNELS OF CHAOS

What's Your Programming Languages?

What's Your Programming Languages?

- C
- C++
- Java
- C#
- Go
- Rust
- Zig
- JavaScript / TypeScript
- Python
- PHP
- Perl
- Shell Scripting
- Haskell
- Lisp
- Scala
- Clojure
- Scheme
- Ocaml
- Erlang
- Elixir

Why Go for Cloud Computing?

Why Go for Cloud Computing?

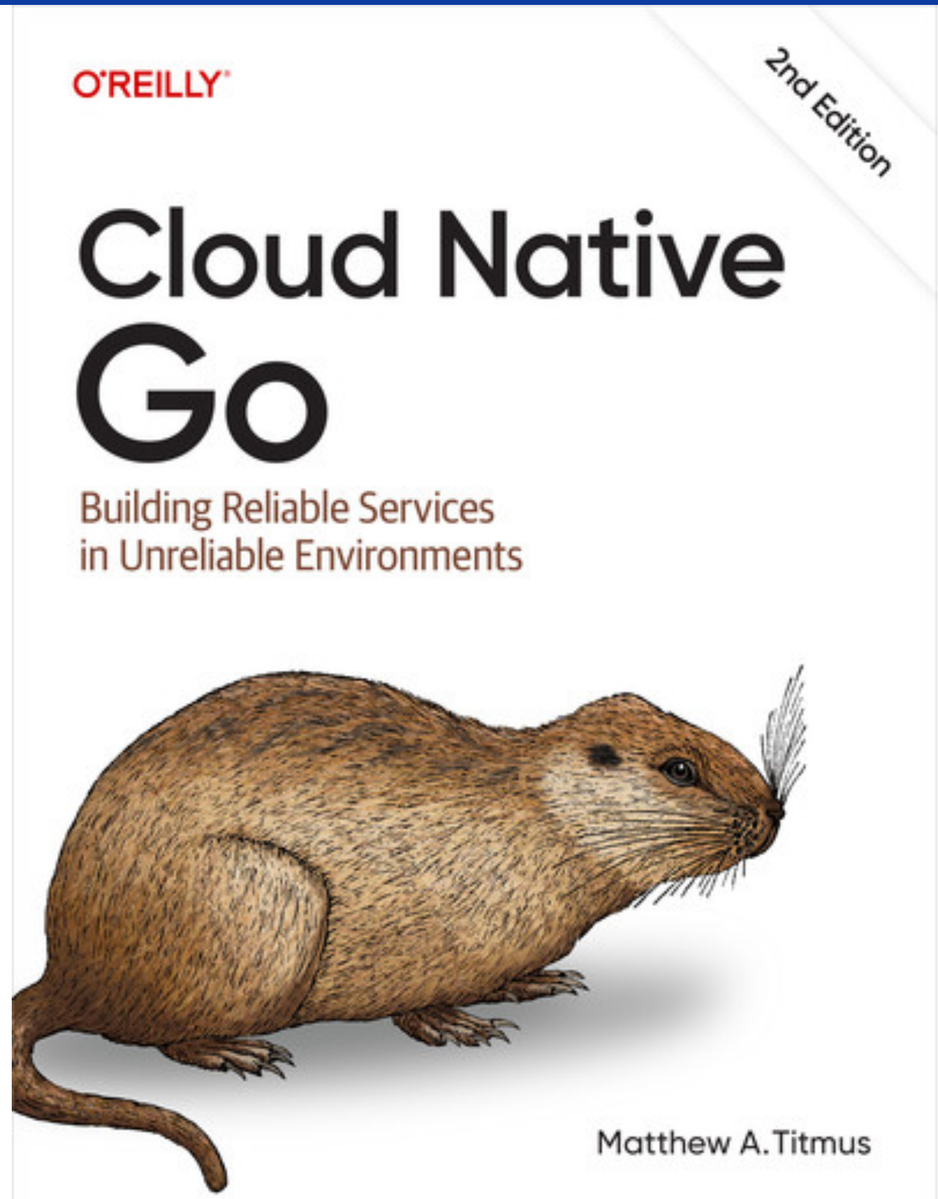
- Type-safe, yet simple and concise language design
- Compiles to static binaries → easy deployment
- Excellent standard library (net/http, encoding/json)
- Concurrency built-in (goroutines, channels)
- Widely adopted: Docker, Kubernetes, Terraform

Why Go for Cloud Computing?

- Generics (since Go 1.18) → type-safe collections and generic functions
- First-class testing support (go test, benchmarks, fuzzing)
- Strong cross-compilation support (Linux, macOS, Windows, ARM)
- Large ecosystem of cloud-related libraries
- Backward compatibility → stable for long-term projects

Book Suggestion

- Recommended if you like books!



Why Go, in General?

Why Go, in General?

- Prepare you for
  - DAT520 Distributed Systems
  - Systems Research
  - Working in companies that doesn't do C#

Go's Tools and Environment

The go command: The Basics

```
$ go run main.go
$ go fmt ./…
$ go test ./…
$ go vet ./…
$ go build
$ go install
$ go doc
$ go doc fmt.Println
```

The go command: Managing Modules and Dependencies

```
$ go mod init github.com/meling/proto2
$ go get github.com/relab/gorums
$ go get -u github.com/google/go-cmp
 go: upgraded github.com/google/go-cmp v0.6.0 => v0.7.0
$ go mod download
$ go mod tidy
```

Go Tools

go help

```
Go is a tool for managing Go source code.

Usage:

        go <command> [arguments]

The commands are:

        bug         start a bug report
        build       compile packages and dependencies
        clean       remove object files and cached files
        doc         show documentation for package or symbol
        env         print Go environment information
        fix         update packages to use new APIs
        fmt         gofmt (reformat) package sources
        generate    generate Go files by processing source
        get         add dependencies to current module and install them
        install     compile and install packages and dependencies
        list        list packages or modules
        mod         module maintenance
        work        workspace maintenance
        run         compile and run Go program
        telemetry   manage telemetry data and settings
        test        test packages
        tool        run specified go tool
        version     print Go version
        vet         report likely mistakes in packages
```

```
meling-labs on  main via 🐹 v1.25.0
❯ pwd
/Users/meling/work/cloud/2025/meling-labs

meling-labs on  main via 🐹 v1.25.0
❯ go version
go version go1.25.0 darwin/arm64

meling-labs on  main via 🐹 v1.25.0
❯ code .

meling-labs on  main via 🐹 v1.25.0
❯
```

VSCode: Go Extension

Go

Go Team at Google go.dev | 16,505,344 | ★★★★½ (262)

Rich Go language support for Visual Studio Code

Disable | Uninstall | Switch to Pre-Release Version | Auto Update

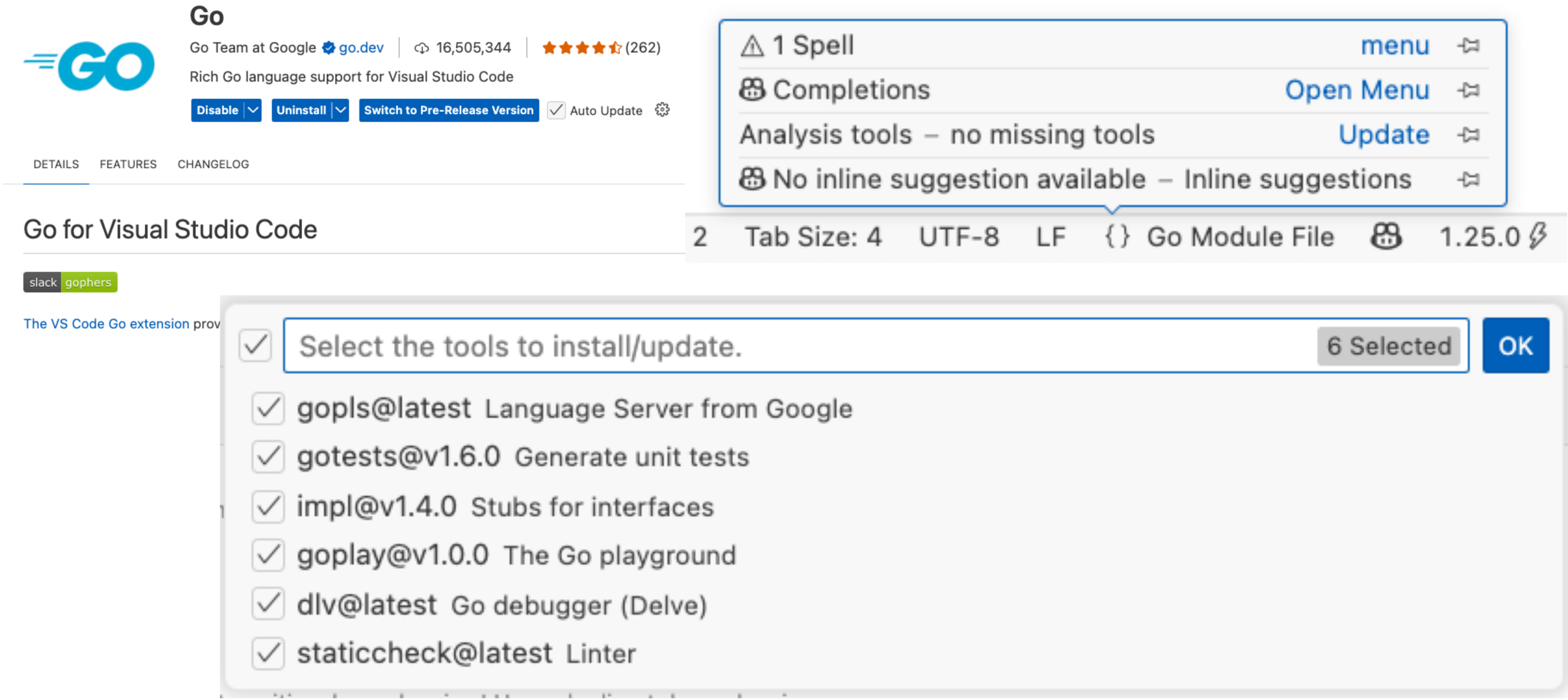
DETAILS FEATURES CHANGELOG

Go for Visual Studio Code

slack gophers

The VS Code Go extension provides rich language support for the Go programming language.

VSCode: Go Extension w/Analysis Tools



VSCode: Go Extension w/Analysis Tools

PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE PORTS GITLENS GITHD SPELL CHECKER Filter

```
2025-08-19 21:19:33.414 [info] All tools successfully installed. You are ready to Go. :)
2025-08-19 21:20:22.913 [info] Tools environment: GOPATH=/Users/meling/go, GOTOOLCHAIN=auto
2025-08-19 21:20:22.913 [info] Installing 6 tools at /Users/meling/go/bin
2025-08-19 21:20:22.913 [info]   gopls
2025-08-19 21:20:22.913 [info]   gotests
2025-08-19 21:20:22.913 [info]   impl
2025-08-19 21:20:22.913 [info]   goplay
2025-08-19 21:20:22.913 [info]   dlv
2025-08-19 21:20:22.913 [info]   staticcheck
2025-08-19 21:20:22.913 [info]
2025-08-19 21:20:23.449 [info] Installing golang.org/x/tools/gopls@latest (/Users/meling/go/bin/gopls) SUCCEEDED
2025-08-19 21:20:23.882 [info] Try to start language server - installation (enabled: true)
2025-08-19 21:20:23.942 [info] Running language server gopls(v0.20.0/go1.25.0)
2025-08-19 21:20:24.088 [info] Installing github.com/cweill/gotests/gotests@v1.6.0 (/Users/meling/go/bin/gotests) SUCCEEDED
2025-08-19 21:20:24.673 [info] Installing github.com/josharian/impl@v1.4.0 (/Users/meling/go/bin/impl) SUCCEEDED
2025-08-19 21:20:25.675 [info] Installing github.com/haya14busa/goplay/cmd/goplay@v1.0.0 (/Users/meling/go/bin/goplay) SUCCEEDED
2025-08-19 21:20:26.157 [info] Installing github.com/go-delve/delve/cmd/dlv@latest (/Users/meling/go/bin/dlv) SUCCEEDED
2025-08-19 21:20:26.519 [info] Installing honnef.co/go/tools/cmd/staticcheck@latest (/Users/meling/go/bin/staticcheck) SUCCEEDED
2025-08-19 21:20:26.519 [info]
2025-08-19 21:20:26.519 [info] All tools successfully installed. You are ready to Go. :)
```

Configuring the Go Extension

.vscode/settings.json

```
{
  "go.useLanguageServer": true,
  "go.coverOnTestPackage": false,
  "go.toolsManagement.autoUpdate": true,
  "gopls": {
    "formatting.gofumpt": true,
    "ui.semanticTokens": true,
    "build.directoryFilters": [
      “-node_modules",
      "-doc"
    ],
    "usePlaceholders": false,
    "staticcheck": true
 },
  "go.lintTool": "golangci-lint",
  "go.lintFlags": [
    "--fast",
    "--errcheck"
 ],
  "go.coverageDecorator": {
    "type": "gutter"
  },
  "git.inputValidation": true,
  "git.inputValidationSubjectLength": 72,
  "git.inputValidationLength": 72,
  "github.copilot.editor.enableCodeActions": false,
  "clang-format.executable": "/opt/homebrew/bin/clang-format",
  "clang-format.style": "{ IndentWidth: 2, BasedOnStyle: google, AlignConsecutiveAssignments: true, ColumnLimit: 120 }",
  "protoc": {
    "compile_on_save": false,
    "options": [
      "--proto_path=${workspaceFolder}",
      "--proto_path=${workspaceRoot}/3net/grpc/proto",
      "--proto_path=${env.GOPATH}/src",
      "--proto_path=`go list -m -f {{.Dir}} google.golang.org/protobuf`"
    ]
  },
  "[proto3]": {
    "editor.defaultFormatter": "zxh404.vscode-proto3"
  },
```

Go Please: gopls

```
$ gopls check 3net/web/server.go
$ gopls references 3net/web/server.go:22:6

# Remove an import statement using vim before doing:
$ gopls check 3net/web/server.go
$ gopls imports

$ gopls help
```

Linters: golangci-lint

```
$ golangci-lint run
$ golangci-lint fmt --diff-colored --enable gofmt
$ golangci-lint run --enable-only godox
$ golangci-lint run --disable errcheck

$ golangci-lint linters
$ golangci-lint formatters
```

Go Environment

```
$ go env
GOARCH='arm64'
GOCACHE='/Users/meling/Library/Caches/go-build'
GOMOD='/Users/meling/work/hotstuff/go.mod'
GOMODCACHE='/Users/meling/go/pkg/mod'
GOOS='darwin'
GOPATH='/Users/meling/go'
GOROOT='/opt/homebrew/Cellar/go/1.25.0/libexec'
GOVERSION='go1.25.0'
```

Joke time

go fmt is like a strict teacher.

"You will write your braces this way... or else."

Testing

Testing

```
$ go test
$ go test -v
$ go test -run TestFibonacci
$ go test -run TestFibonacci -count=10
```

Testing: Writing Tests

```go
// Package fib provides a function to compute Fibonacci numbers.
package fib

func fibonacci(n uint) uint {
    if n == 0 {
        return 0
    }
    if n == 1 {
        return 1
    }
    return fibonacci(n-1) + fibonacci(n-2)
}
```

Testing: Writing Tests

```go
package fib

import "testing"

run test | debug test
func TestFibonacci(t *testing.T) {
    fibonacciTests := []struct {
        in, want uint
    }{
        {0, 0},
        {1, 1},
        {2, 1},
        {5, 5},
        {20, 6765},
    }
    for _, ft := range fibonacciTests {
        got := fibonacci(ft.in)
        if got != ft.want {
            t.Errorf("fibonacci(%d) = %d, want %d", ft.in, got, ft.want)
        }
    }
}
```

Testing: Writing Tests

```go
package fib_test

import (
    "testing"

    "dat515/2go/sequence/fib"
)

run test | debug test
func TestFibonacci(t *testing.T) {
    fibonacciTests := []struct {
        in, want uint
    }{
        {0, 0},
        {1, 1},
        {2, 1},
        {5, 5},
        {20, 6765},
    }
    for _, ft := range fibonacciTests {
        got := fib.Fibonacci(ft.in)
        if got != ft.want {
            t.Errorf("Fibonacci(%d) = %d, want %d", ft.in, got, ft.want)
        }
    }
}
```

Testing: Writing Tests w/Subtests

```go
func TestMapKeysNonDeterminism(t *testing.T) {
    failed := false
    // run 20 tests to detect non-deterministic behavior; if any test fails, stop
    for i := 0; i < 20 && !failed; i++ {
        t.Run(fmt.Sprintf("Run-%d", i+1), func(t *testing.T) {
            x := map[string]int{"a": 1, "b": 2}
            got := keys(x)
            want := []string{"a", "b"}
            if diff := cmp.Diff(got, want); diff != "" {
                failed = true
                t.Error("The keys() function does not sort the keys.")
                t.Errorf("keys() mismatch (-got +want):\n%s", diff)
            }
        })
    }
}
```

Testing with Docker's SDK

Docker Daemon must be running for the following examples

Testing: Official Docker Go SDK

https://pkg.go.dev/github.com/docker/docker

```
# Implement your own CI tool with official Docker SDK
$ mkdir docker-ci; cd docker-ci
$ go mod init github.com/meling/ci
$ go get github.com/docker/docker
$ cat go.mod
```

Testing: Official Docker Go SDK

```go
func TestPing(t *testing.T) {
    c, err := client.NewClientWithOpts(
        client.FromEnv,
        client.WithAPIVersionNegotiation(),
    )
    if err != nil {
        t.Fatalf("Failed to create container client: %v", err)
    }
    if _, err := c.Ping(t.Context()); err != nil {
        t.Fatalf("Failed to ping Docker daemon: %v", err)
    }
    t.Log("Ping successful")
}
```

Testing: Official Docker Go SDK

https://pkg.go.dev/github.com/docker/docker

```
# Implement your own CI tool with official Docker SDK
$ mkdir docker-ci; cd docker-ci
$ go mod init github.com/meling/ci
$ go get github.com/docker/docker
$ cat go.mod
$ go mod tidy
$ go test -v
$ cat go.mod
```

Testing: Simplified Docker Go SDK

https://pkg.go.dev/github.com/relab/container

```
# Implement your own CI tool
$ mkdir ci; cd ci
$ go mod init github.com/meling/ci
$ go get github.com/relab/container
$ cat go.mod
```

Testing: Simplified Docker Go SDK

```go
package ci

import (
    "testing"

    "github.com/relab/container"
)

run test | debug test
func TestPing(t *testing.T) {
    c, err := container.NewContainer()
    if err != nil {
        t.Fatalf("Failed to create container client: %v", err)
    }

    if err := c.Ping(t.Context()); err != nil {
        t.Fatalf("Failed to ping Docker daemon: %v", err)
    }
    t.Log("Ping successful")
}
```

Joke time

I wrote a test yesterday.

→ It passed...

but only because I forgot the t.Fatal()

Benchmarking

Benchmarking

Examples from github.com/relab/bbhash

```
$ go test -run x -bench BenchmarkBBHashMarshal -benchmem
$ benchstat old.txt new.txt
# Advanced usage
$ benchstat -table "" -col /func -row /size bc-bench.txt
```

Questions?