

# **Sustainability and Security in the Go ecosystem**

Martin Czygan [martin.czygan@gmail.com](mailto:martin.czygan@gmail.com)

2023-03-11

# Hello!

- open data engineer at [Internet Archive](#), working on [scholar.archive.org](#) and [rclone](#) and software developer at [Leipzig University Library](#) on index metadata for [libraries in Germany](#)
- learned about [Go](#) in 11/2009 through Google Tech Talks
- active user since 2013
- since 2019 co-host of [Leipzig Gophers Meetup](#)
- open source [contributions](#), “data space”



A Library in Space

## Leipzig Gophers

- est. 2019, [golangleipzig.space](https://golangleipzig.space)
- blog, 30+ (hybrid) events, [500+ members](#)
- gave away tech books, JetBrains subs, [Zimaboard](#), and more swag, ...



Leipzig Gopher

We talk about language features, libraries, cloud tools, databases and in the past collaborated with interesting companies using Go, like [edgeless systems](#), [Gridfuse](#), [deta](#), [CodeNotary](#) and others.

Reach out!

## Overview

- sustainability, as in **maintenance**
- security, as in **bugs**
- sustainability, as in **resource efficiency**

# Surviving Software Dependencies

Surviving Software Dependencies, Cox, 2019

The Copay and Equifax attacks are clear warnings of real problems in the way software dependencies are consumed today.

## Version control systems package repository

Package names are locators. A decentralized infrastructure. There is not package *central*. There is an aggregation, however, [pkg.go.dev](https://pkg.go.dev).

```
package main

import "github.com/fatih/color"

func main() {
    color.Yellow("alert")
}
```

## Go modules are the defacto standard

Relevant files are `go.mod` and `go.sum`, checked into version control. Both are mostly edited by tools, e.g. `go mod ...`

```
$ ls -l
main.go  # your program
go.mod   # dependencies
go.sum   # checksums
```

No lockfile. Run `go mod tidy` and you're good.

## Example go.mod

The latest tag is used as version, but any commit id or pseudo-version would work.

```
module yellowalert
```

```
go 1.20
```

```
require github.com/fatih/color v1.14.1
```

```
require (  
    github.com/mattn/go-colorable v0.1.13 // indirect  
    github.com/mattn/go-isatty v0.0.17 // indirect  
    golang.org/x/sys v0.3.0 // indirect  
)
```



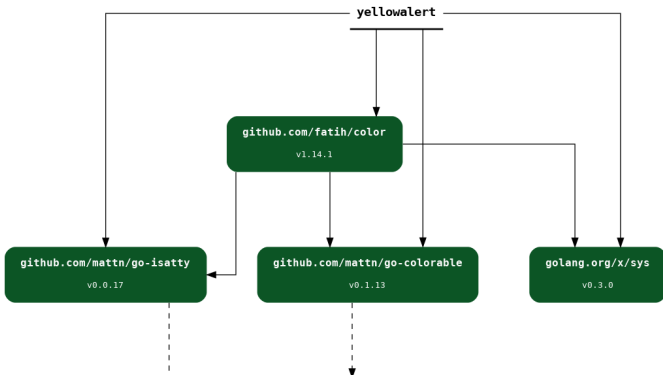
## Visually

Go + tools = ?

```
$ go install github.com/lucasepe/modgv@latest
```

Generate a dependency graph from go.mod, e.g. via:

```
$ go mod graph | modgv | dot -Tpng > gomod.png
```



## Example go.sum

Currently, SHA256 (h1) is used; hashes over files or trees. Not a lock file.

```
github.com/fatih/color v1.14.1 h1:qfhVLaG5s+nCR0l1zJs...  
github.com/fatih/color v1.14.1/go.mod h1:2oHN61fhTpgc...  
...
```

Once a dependency has been fetched and its hash computed, it cannot be changed without notice.

- flip a bit (e.g. git would allow us to move a tag, ...)

```
$ go mod verify  
github.com/fatih/color v1.14.1: dir has been modified (...)
```

## Any domain works

Not tied to any source code host, or any domain for that matter.

```
$ go install golangleipzig.space/clt23@latest
```

Details: **CLT23** - that site is run by a static-site-generator and redirects to the actual repository.

# Leftpad!

Any problem ... can be solved by another level of indirection.

- **Go Module Proxy**, launched 2019-08-29

The go tool will ask **proxy.golang.org** first, use `GOPRIVATE="*"` to disable.  
Run your own proxy, if you run a company.

## Minimal Version Selection

An algorithm to resolve dependencies. Fast (not NP-complete), does not require lock files.

- choose minimal version required for any dependency
- depends on *import compatibility rule*

A human element required for any dependency management (e.g. we expect 1.2.3 be compatible with 1.2.4, and we expect a v2 to be backwards incompatible).

Other ecosystems are curious, e.g. cargo:

```
$ cargo -Z help | grep minimal-versions
```

```
-Z minimal-versions -- Resolve minimal dependency versions inst
```

## Major Version Update

Go has a strict recommendation, when it comes to major version upgrades: you should use a different name - i.e. a different import path, typically `.../v2`, `.../v3`, ...

- `golangleipzig.space/clt 1.0.0`
- `golangleipzig.space/clt/v2 2.0.0`

## The story so far

- cannot change code, once required (and `go.sum` is checked in)
- very slim chance for **leftpad** (when using a proxy)
- very **fast** dependency resolution and high-fidelity builds
- with vendoring, we get **reproducible builds**