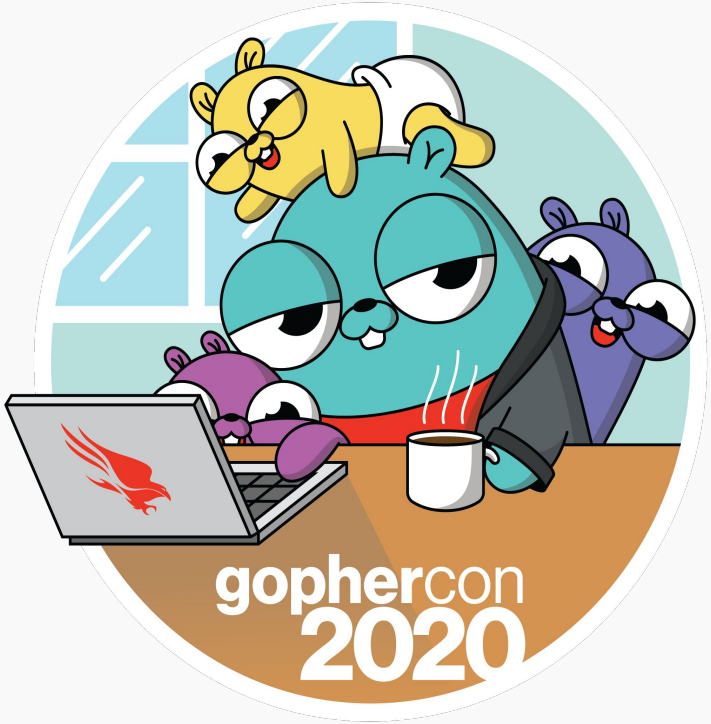



# Untangling the Monorepo

*Moving to Go Modules*

Dylan Bourque and Anthony Lee

CROWDSTRIKE

## Anthony Lee

- Economics, B.A. and Political Science, B.A.
- Minneapolis, MN
- Writing Go since 2016
- Worked in healthcare, logistics, adtech
- Favorite beer of ‘20: Fair State’s *Mirror Universe*

## Dylan Bourque

- Computer Science, B.S.
- Gonzales, LA
- Professional developer since 1998
- Writing Go since 2016
- Worked in banking, healthcare, fin-tech
- Favorite beer of ‘20: Parish Brewing’s *Sacred Islands*

# CROWDSTRIKE


Go!


Cyber Security!

Hiring!

Building Uber's Go Monorepo with Bazel (uber.com)
158 points by dayanruben 4 months ago | hide | past | favorite | 138 comments

Why Google Stores Billions of Lines of Code in a Single Repository (2016) (acm.org)
112 points by tosh 9 months ago | hide | past | favorite | 121 comments

Monorepo or Multirepo? Role-Based Repositories (7mind.io)
141 points by optician_owl 12 months ago | hide | past | favorite | 115 comments

Monorepo is great if you're really good (yosefk.com)
225 points by mr_crankypants on July 30, 2019 | hide | past | favorite | 151 comments

Monorepo: please do (medium.com)
234 points by Soliah on Jan 3, 2019 | hide | past | favorite | 161 comments

Monorepos: Please don't (medium.com)
332 points by louis-paul on Jan 3, 2019 | hide | past | favorite | 391 comments

Ask HN: What are the pros / cons of using monorepos?
20 points by factorialboy 27 minutes ago | hide | past | favorite | 5 comments

# Mono vs Multi Repo

*The right tool for the job*

**Work in the City**
*João Candido Portinari*
*1939, Brazil*

Watercolor, colored pencil

# Monorepo

- Non-intuitive, clumsy interactions with modules
  - Git tags for each module must be prefixed with the relative path to the module from the repo root, i.e. `foo/v1.0.0`, `bar/v1.2.0` and `baz/v1.17.6`
  - What You See is **NOT** What You Get!

```
.../monorepo/
   foo/
      foo.go
      go.mod
   bar/
      main.go
      go.mod
   baz/
      main.go
      go.mod
```

# The Knot 🧬

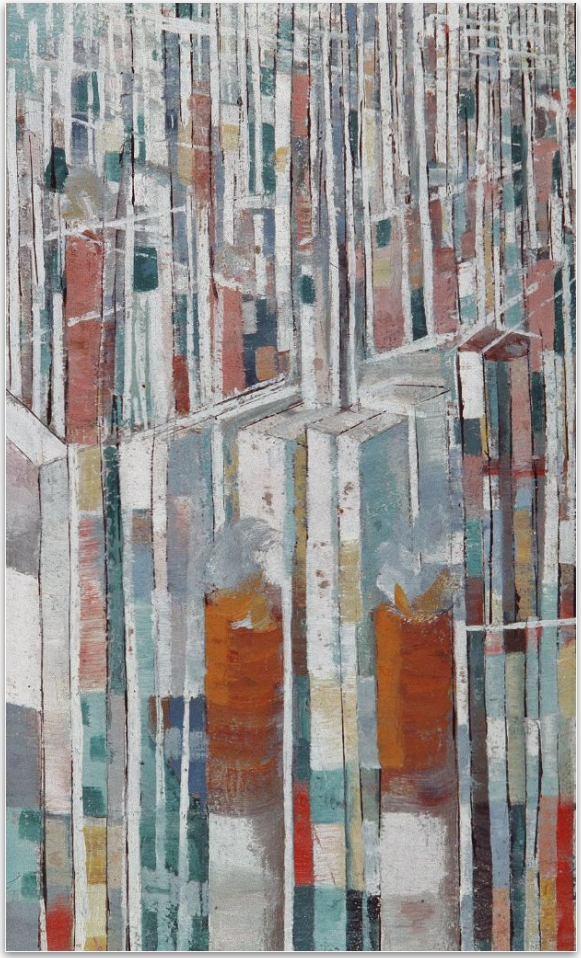
- Several “kitchen sink” packages of loosely related code
- Entwined dependencies from different domains:
  - String manipulation
  - IP parsing
  - AWS SDK wrappers
  - And on and on...
- First step: Put code back where it belongs

# Monorepo

- Non-intuitive, clumsy interactions with modules
- Allows undisciplined dependency management
- Alleviated by sophisticated tooling...
- ...but our time is valuable.

# CrowdStrike's Quirks

- Nearly 400 microservices and libraries, **not** including vendored dependencies
- Committed our entire $GOPATH, which our custom tooling relies on
- Module incompatible import paths (`"cs/foo"`)
- Kitchen sink packages and knotted dependencies



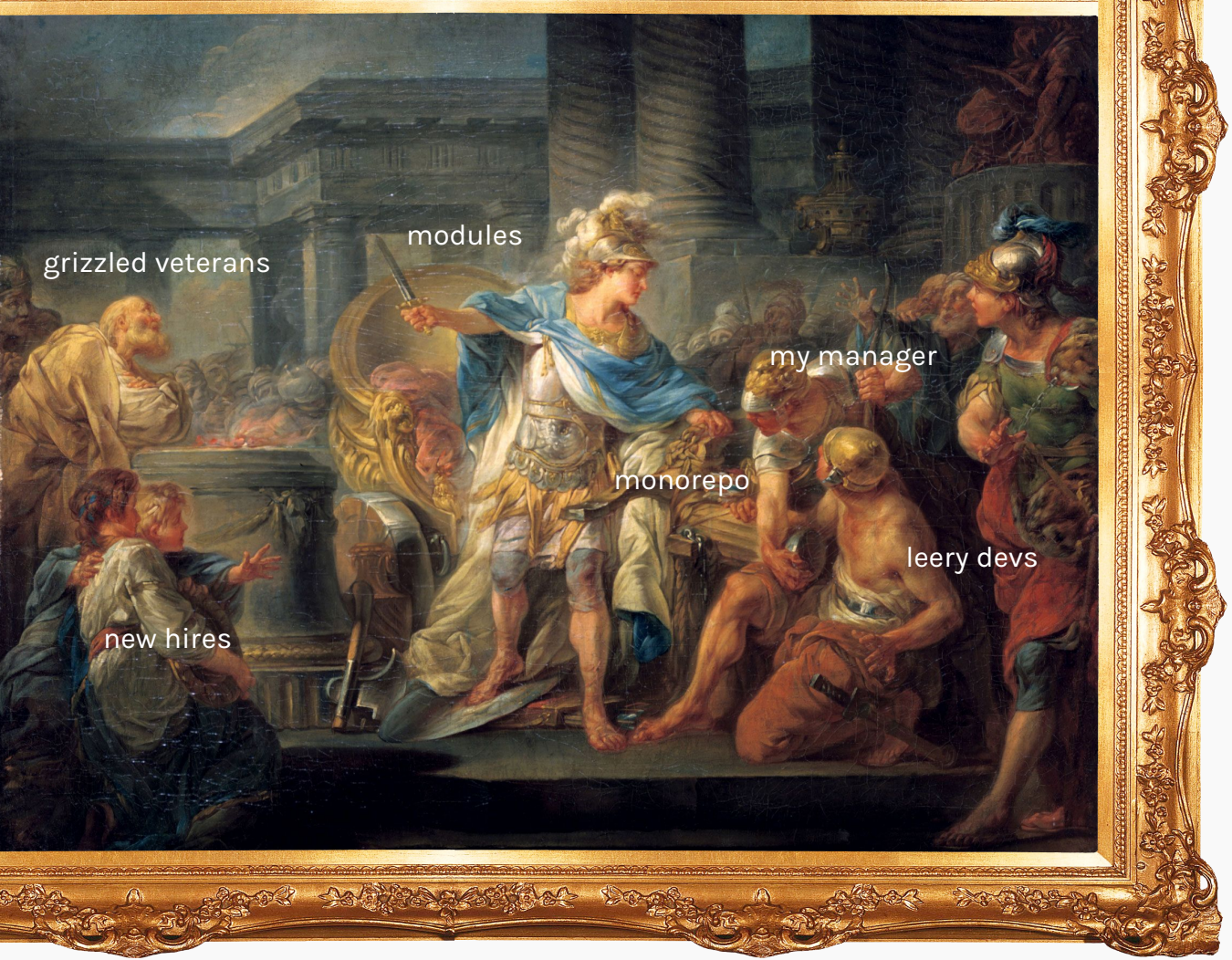
# Multi Repo

- A well understood “default” workflow
- Less need to be aware of downstream deps
- More administrative overhead...
- ...but it’s a common problem

Project/owner level tooling already exists!

**Industrialization of Brazil**
*João Candido Portinari*
*1960, Brazil*

Oil




**CrowdStrike Cuts the Monorepo Knot**
*Anthony Lee and Dylan Bourque*
*2020, USA*

Mixed media; modules toolchain on git

Standard problems have standard solutions

**The Bookworm**
*Carl Spitzweg*
*1850, Germany*

Oil on Canvas

# Decision Time

- How are your teams structured and what are their goals?
- How extensible are your existing CI/CD workflows?

For CrowdStrike, multirepo was a better fit

- Rapidly hiring for increasingly specialized teams
- Increase independence and agility
- Shorten onboarding runway
- Existing CI/CD expertise, not much in the way of Bazel

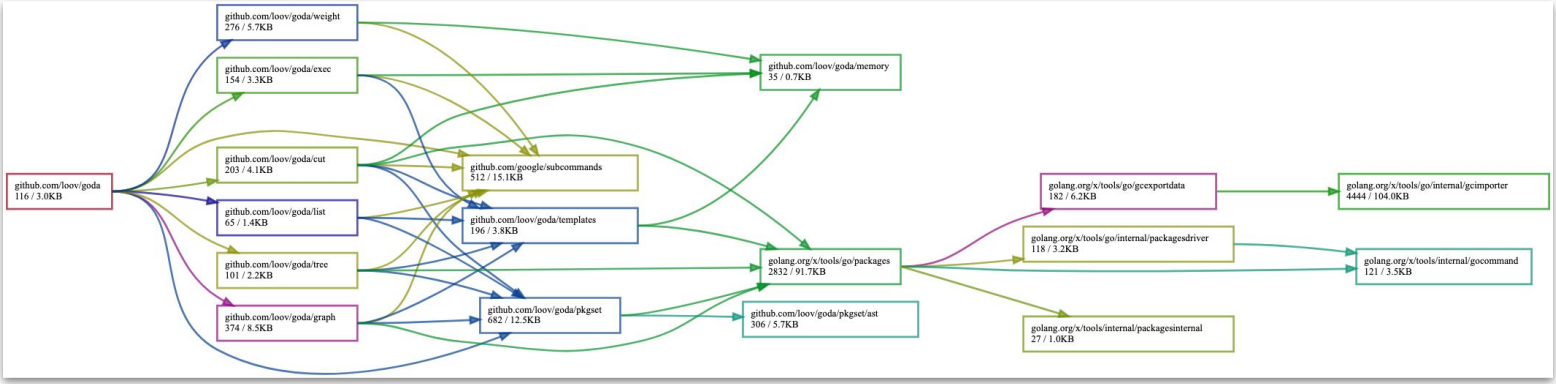
# Deconstructing the Monolith

**I costruttori**
*Massimo Campigli*
*1928, Italy*

Oil on Canvas

# Find your leaf packages

[loov/goda: Go Dependency Analysis toolkit]



**For each leaf package**

1. Create a repository and module for it
2. Vendor it back into the monorepo
3. Adjust imports
   `import “baz”` → `import “foo.bar/baz”`
4. Remove the original package

# Preserve your git history

[newren/git-filter-repo: Quickly rewrite git repository history]

```
git filter-repo --path src/ --to-subdirectory-filter my-module --tag-rename '':'my-module-'
```

- Commit messages are valuable and should be preserved
- git-filter-repo can refactor directories and rewrite git histories

# Pay Down Technical Debt

- Where it is not disruptive, update code to follow current recommended patterns and idioms
- Address linter warnings so that “new” code is clean
- Be careful not to raise the barrier to entry for migration!
  - Do not include behavior changes in the initial converted module
  - Do NOT address cosmetic style. `gofmt/goimports` are your friend

Write shiny, new tools to replace crusty, old tools to make code compatible with boring, standard tools.

```
~/go/src/go.crwd.dev/csi
./csi help
usage: csi [<flags>] <command> [<args> ...]

monorepo/modules transition tool

Flags:
  -h, --help  Show context-sensitive help (also try --help-long and --help-man).

Commands:
  help [<command>...]
    Show help.

  mod [<flags>] [<pkgs>...]
    Modularize a directory. The output will recursively create directories to match the full module name.

  freeze [<flags>] [<pkgs>...]
    Flash freeze and package a dir.

  pack [<flags>] [<pkgs>...]
    Package a module for Athens.

  remote [<flags>] [<repos>...]
    Create a remote for a local repo and push it.

  adjust [<dir>] [<pattern>...]
    Adjust imports.

  skeleton [<flags>] [<name>]
    Create and push a skeleton repo. When no library, service or cli is supplied only boilerplate files will be copied.

  version
    Returns the version of csi.
```

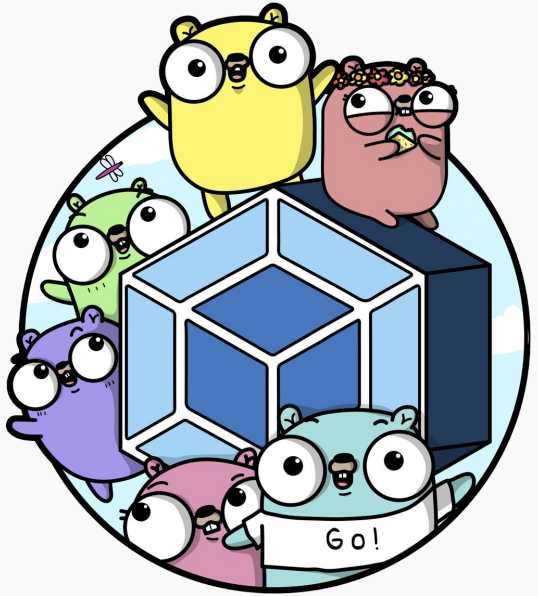
# Freezing Dependencies

**Snow in Beijing**
*Wu Guanzhong*
*1994, China*

Oil on Canvas

`github.com/foo/bar@v0.0.0-crowdstrike`

- Use a prerelease tag to indicate a potentially unstable API
- The suffix indicates that a dep may have been modified or has an unknown version
- `v0.0.0` means that it precedes all other semver tags
- Gives a path forward towards properly versioned deps




# Setting up a Go Proxy

[gomods/athens: A Go module datastore and proxy]
[plexsystems/pacmod: Package your Go modules]

The Go Proxy serves the flash frozen version

# Use a Vanity URL

- Indirect your import paths from your VCS
- Treat yourself to a fancy name
- Make sure you own the actual domain

```
http {
    # This is the Vanity URL site.
    server {
        server_name vanity.foo
        listen 80 default_server;
        location ~* ^\/(?<proj>[^\/]+)\/(?<repo>[^\/]+)(\/.*)?$ {
            # This is a go-get operation.
            if ($arg_go-get = "1") {
                return 200 '
<html><head>
    <meta name="go-import"
        content="vanity.foo/$repo
            git https://vcs.foo/$proj/$repo.git">
    <meta name="go-source"
        content="example.vanity.url.com/$repo
            https://vcs.foo/$proj/$repo
            https://vcs.foo/$proj/$repo/tree/master{/dir}
            https://vcs.foo/$proj/$repo/blob/master{/dir}/{file}#L{line}">
</head></html>';
            }
            # Default to redirecting to VCS for any other request
            return 301 vsc.foo/projects/$proj/repos/$repo;
        }
    }
}
```

# Distribute the correct `go env`

```
GO111MODULE=""

# Hit our internal modules proxy first, then the public proxy, then VCS
GOPROXY="https://proxy.foo.bar,https://proxy.golang.org,direct"

# Skip module proxies for internal/vanity imports
# . Our security policy requires that all source remain on-premise
#   and we are using S3 for Athens storage, so we could not configure
#   Athens to fetch and cache our internal code
# . All go get, etc. operations for internal modules go directly to VCS
GOPRIVATE="vanity.foo/*"
GONOPROXY="vanity.foo/*" # not set explicitly, inferred from $GOPRIVATE

# Handle module sum DB checks for flash-frozen dependencies
# . The complete solution would be to build and run your own checksum DB
# . Need to do this because custom versions will not exist in the
#   public checksum DB
# . Another option would be disable the checksum DB for those deps
GONOSUMDB="github.com/example/dependency"
```


# Impacts at CrowdStrike

- Reduced required toolchain to mostly `git` and `go`
- Clearer lines of ownership across projects and teams
- Significantly quicker test and build times
- Reduced blast radius of problematic changes
- Trade homegrown dep management for modules

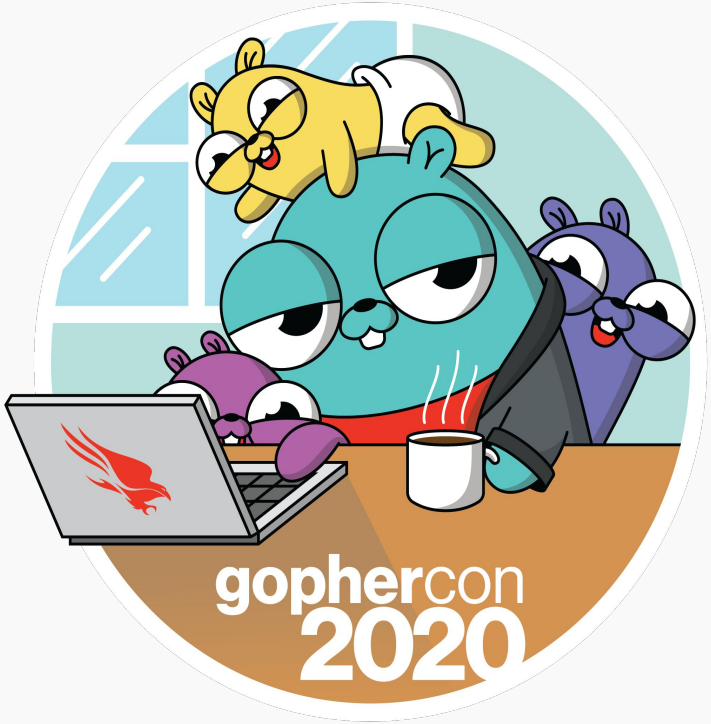
# Lessons Learned

There are no silver bullets, but...

- Properly vendor or fork external dependencies
- Don’t create kitchen sink utils packages
- Be conscientious about major versions and API changes
- Be prepared for a lot of hands on code wrangling

Thanks for joining in!

www.crowdstrike.com/careers/




CROWDSTRIKE

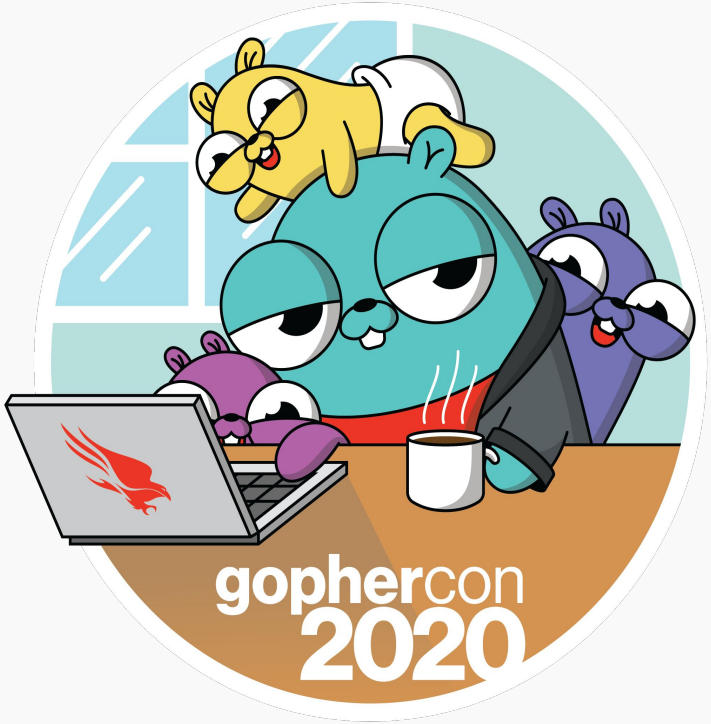
# References


- Google Arts and Culture
- loov/goda: Go Dependency Analysis toolkit
- gomods/athens: A Go module datastore and proxy
- plexsystems/pacmod: Package your Go modules
- ryancurrah/gomodguard: Allow and block list linter for direct Go module dependencies
- myrepos: a tool to manage all your version control repositories


Join CrowdStrike!

# Q & A

Thanks for joining in!




CROWDSTRIKE