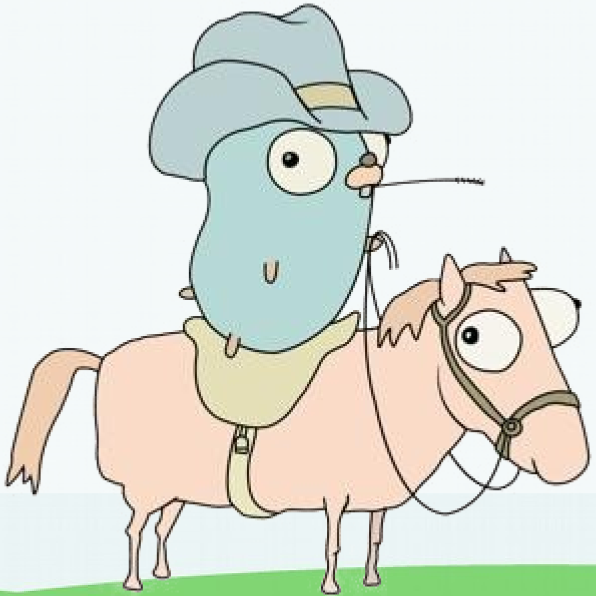
A Simple Introduction to Modules

Chuck England iStreamPlanet

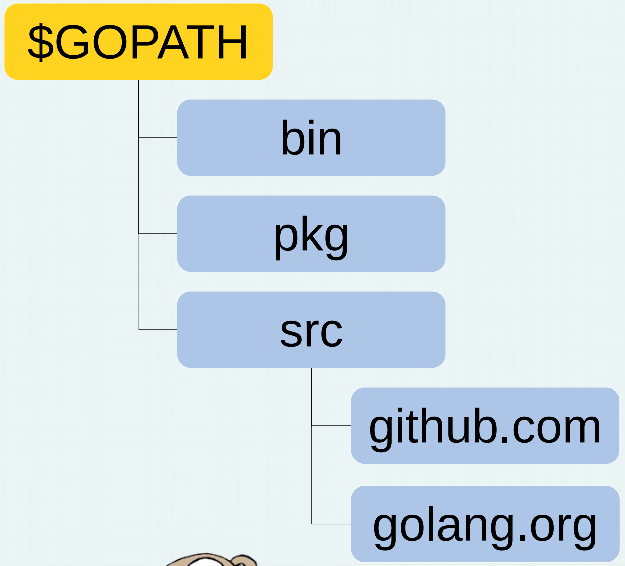
Agenda

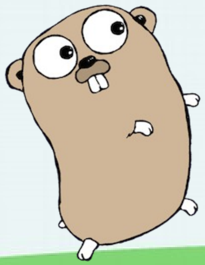
- Dependencies in Go
- Motivation
- Modules



A stroll down memory lane

- Everything lives in the $GOPATH
  - You must develop in a specific place. What, really?
- Dependencies are repositories and pull latest from master
  - Surprises from ref repos
  - Builds are not repeatable

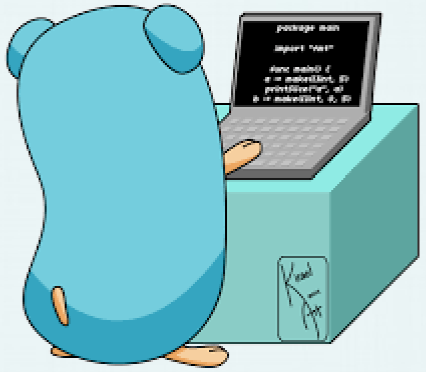




Add-ons to the rescue (sort of)

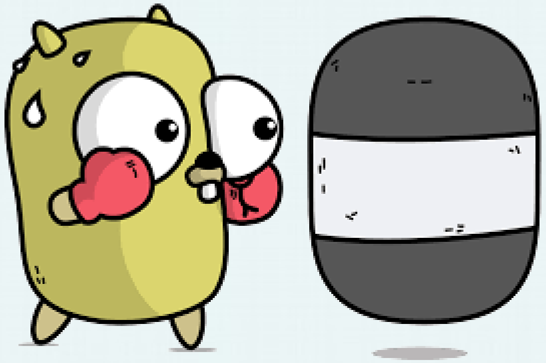
- Why do we need add-ons?
  - Ability to select a branch/commit
  - Ability for references to select their own branch/commit
  - Ability to know all specific dependencies
    - Aka: vendor
- Some add-on examples
  - Glide
  - dep

Modules



- Dependencies are managed by Go
  - Built-in / no extra download to manage
  - Getting dependencies is: **go get ./…**
  - Works like other language environments
- Choose to vendor (or not)
- Develop in any path

Demo



Redux

Currently being edited to demo

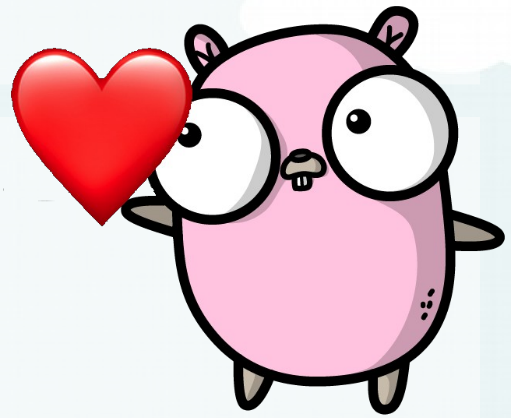
```
go mod init github.com/{me}/{project}
go get ./...
go mod tidy
go mod list -m all
go mod list -u -m all
go mod vendor
go build -mod=vendor ./...
```

Modules

As a result of Semantic Import Versioning, code opting in to Go modules must comply with these rules:

1) Follow semver (with tags such as v1.2.3).
2) If the module is version v2 or higher, the major version of the module must be included as a /vN at the end of the module paths used in go.mod files (e.g., module github.com/my/mod/v2, require github.com/my/mod/v2 v2.0.0) and in the package import path (e.g., import "github.com/my/mod/v2/mypkg").
3) If the module is version v0 or v1, do not include the major version in either the module path or the import path.

Semantic Versioning



Follow semver (with tags such as v1.2.3) (see https://semver.org/)

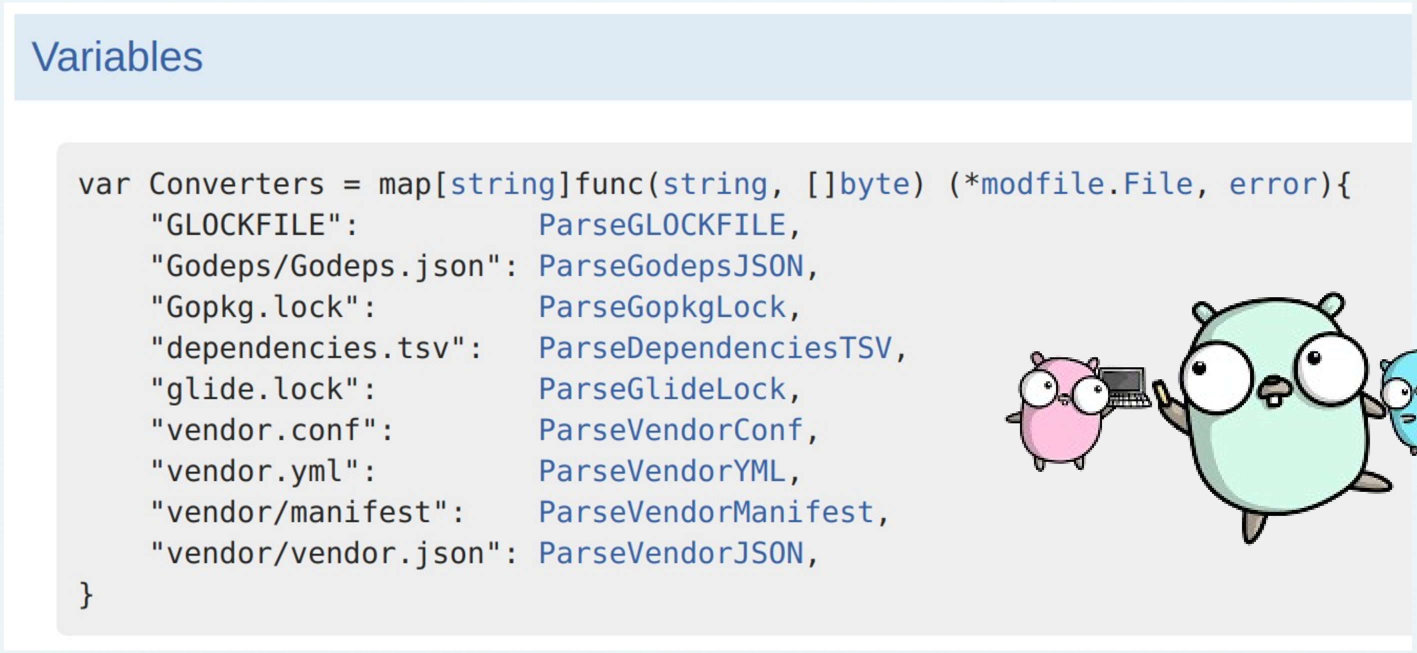
Summary

Given a version number MAJOR.MINOR.PATCH, increment the:

1) MAJOR version when you make incompatible API changes,
2) MINOR version when you add functionality in a backwards-compatible manner, and
3) PATCH version when you make backwards-compatible bug fixes.

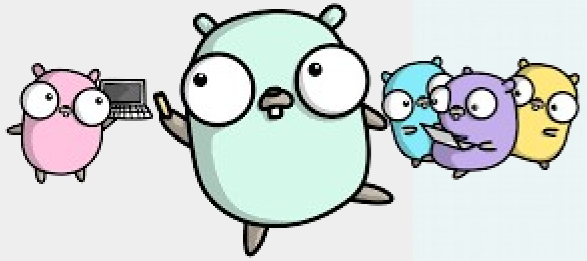
Additional labels for pre-release and build metadata are available as extensions to the MAJOR.MINOR.PATCH format.

Automatic conversions from these



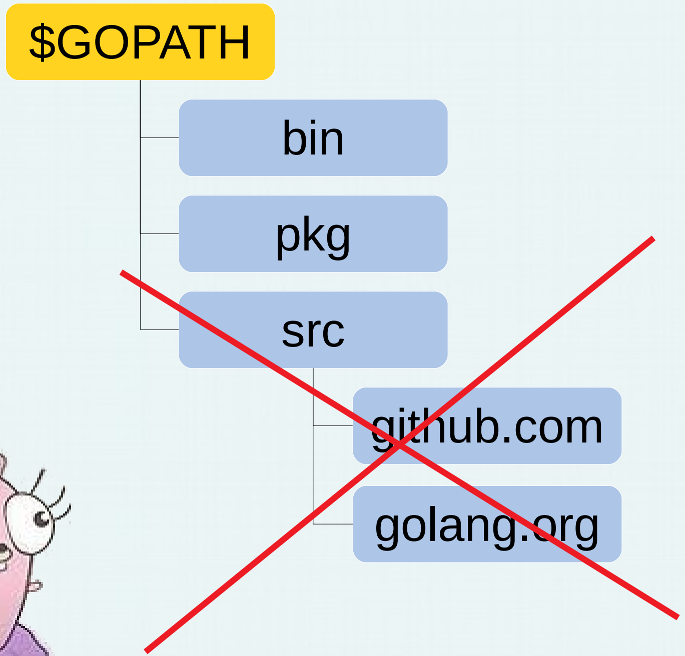
Variables

```
var Converters = map[string]func(string, []byte) (*modfile.File, error){
    "GLOCKFILE":           ParseGLOCKFILE,
    "Godeps/Godeps.json": ParseGodepsJSON,
    "Gopkg.lock":          ParseGopkgLock,
    "dependencies.tsv":   ParseDependenciesTSV,
    "glide.lock":          ParseGlideLock,
    "vendor.conf":         ParseVendorConf,
    "vendor.yml":          ParseVendorYML,
    "vendor/manifest":    ParseVendorManifest,
    "vendor/vendor.json": ParseVendorJSON,
}
```



https://tip.golang.org/pkg/cmd/go/internal/modconv/?m=all#pkg-variables

Fini



cengland@istreamplanet.com

Be sure to check out the
#seattle channel on Slack
https://gophers.slack.com

Want to know more?

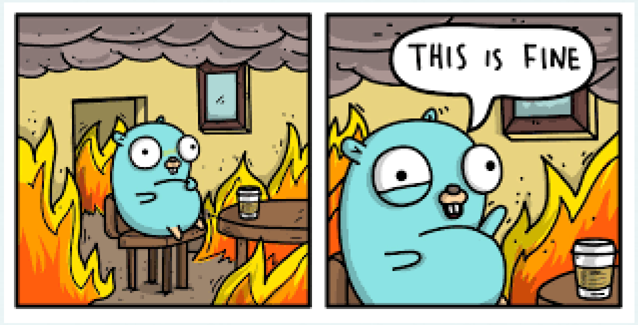
- Explore
  - Modules (https://bit.ly/2AJT5Ei)
  - The module file (go.mod https://bit.ly/2HbE57V)
  - The sum file (go.sum https://bit.ly/2M88X7Z)
  - Vendoring (go mod vendor) (https://bit.ly/2SSrurG)

    Use -mod=vendor or GOFLAGS=-mod=vendor
  - Non-vendored src caching ($GOPATH/pkg/mod)

    https://groups.google.com/forum/#!msg/golang-dev/RjSj4bGSmsw/KMHhU8fmAwAJ
  - “Always on” proxy repos (like Athens) (https://bit.ly/2Hfmegj)

Try it without installing Go:

```
docker run --rm -it golang:1.11
```


THIS IS FINE