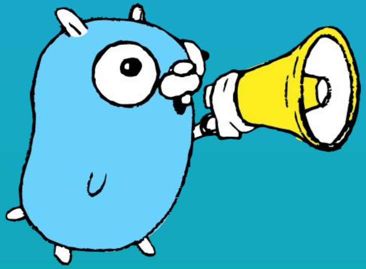
23 August 2018
Go Meetup, Hamburg

Modules

Jan Stamer

red6

@jsTamer21k



Modules

Summary

Modules

- Versioned Packages
- Built Into the Go Tool
- Repeatable Builds
- No GOPATH

The Road Here

Milestones

GOPATH	2012	Go 1.0
goven	2012	
godep	2013	
gopkg.in	2014	
glide	2014	
gb	2015 (April)	
govendor	2015 (April)	
vendor dir	2015 (June)	Go 1.5 -> 1.6
dep	2017	
go mod	2018	Go 1.11 -> 1.12

GOPATH

Configuration by convention enabled

- go get
- go build
- go test

Redesign

GOPATH	2012	Go 1.0
goven	2012	
godep	2013	
gopkg.in	2014	
glide	2014	
gb	2015 (April)	
govendor	2015 (April)	
vendor dir	2015 (June)	Go 1.5 -> 1.6
dep	2017	
go mod	2018	Go 1.11 -> 1.12

Principles

“

Principles of Go Versioning

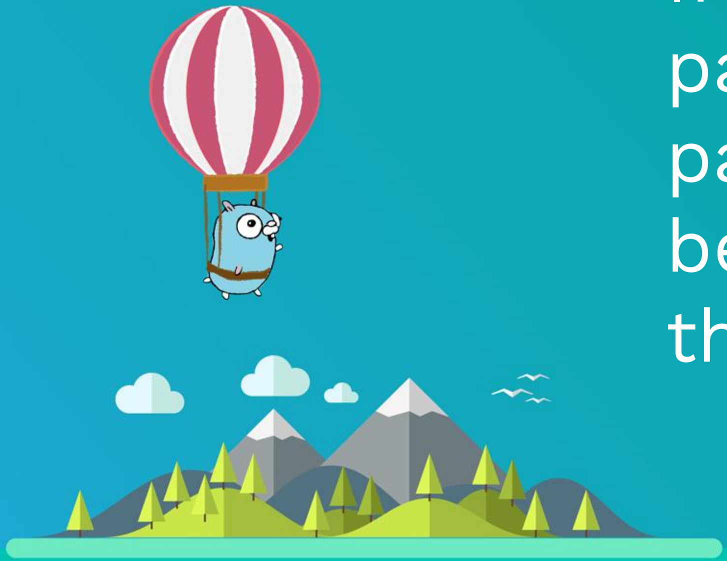
#1 Compatibility

#2 Repeatability

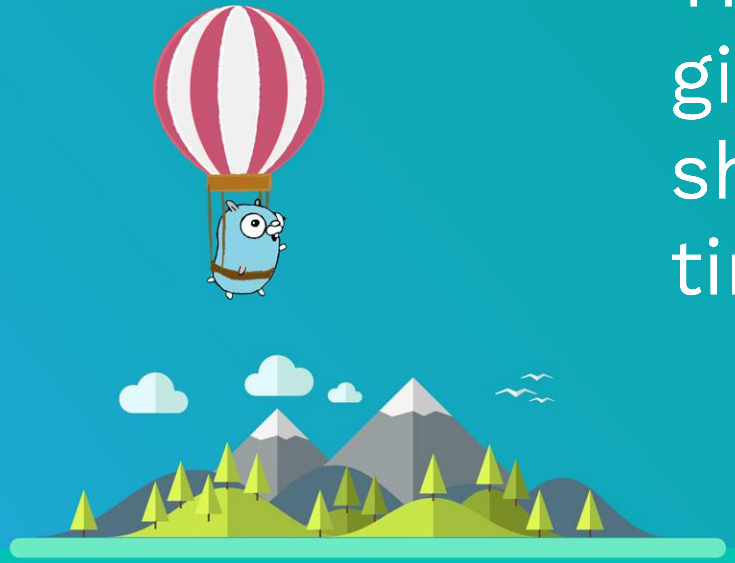
#3 Cooperation

“

If an old package and a new package have the same import path the new package must be backwards compatible with the old package.

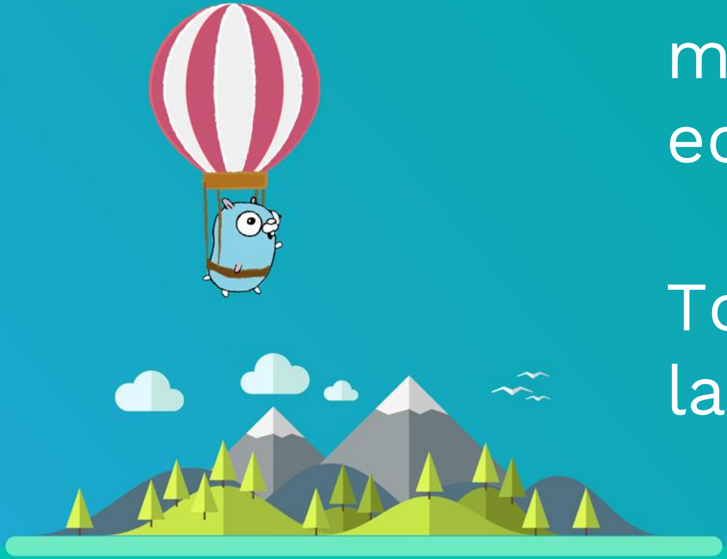


Import Compatiblity Rule ”



“The result of a build of a given version of a package should not change over time.”

Repeatability



“ We must all work together to maintain the Go package ecosystem.

Tools cannot work around the lack of cooperation. ”

Cooperation

Structure

A module is

- A collection of related Go packages
- Unique module path
- The unit of code interchange and versioning
- Semantically versioned

Filesystem Structure

```
<any-path>/
    go.mod                 # module file
    go.sum                 # module checksums
    main.go                # command source
<any-path>/
    .git/                  # optional
    go.mod                 # identifies the module root
    go.sum                 # module checksums
    hello.go               # package source
    hello_test.go          # test source
    stringutil/
        reverse.go         # package source
        reverse_test.go    # test source
```

go.mod

```
module example.com/hello
```

$\leftarrow$ **Module Path**
The import path prefix of the module.

go.mod

```
module example.com/hello

require (
    golang.org/x/text v0.3.0
    gopkg.in/yaml.v2 v2.1.0
)
```

Dependencies
List of modules we depend on and the minimum version we can use.

go.mod

```
module example.com/hello

require (
    golang.org/x/text v0.3.0
    gopkg.in/yaml.v2 v2.1.0
)

exclude github.com/go-stack/stack v1.6.0
```

Exclusions
List of module versions we reject for some reason.

go.mod

```
module example.com/hello

require (
    golang.org/x/text v0.3.0
    gopkg.in/yaml.v2 v2.1.0
)

exclude github.com/go-stack/stack v1.6.0

replace (
    github.com/go-stack/stack v1.4.0 => ../stack/
    golang.org/x/text => github.com/pkg/errors v0.8.0
)
```

Replacements
List of module [versions] we replace with something else.

User Experience

Go 1.11 Module Support

- Inside GOPATH — Status Quo
- Outside GOPATH — Modules reign
- GO111MODULE environment variable
  - auto (default)
  - on
  - off
- Build Cache required for Modules

Commands

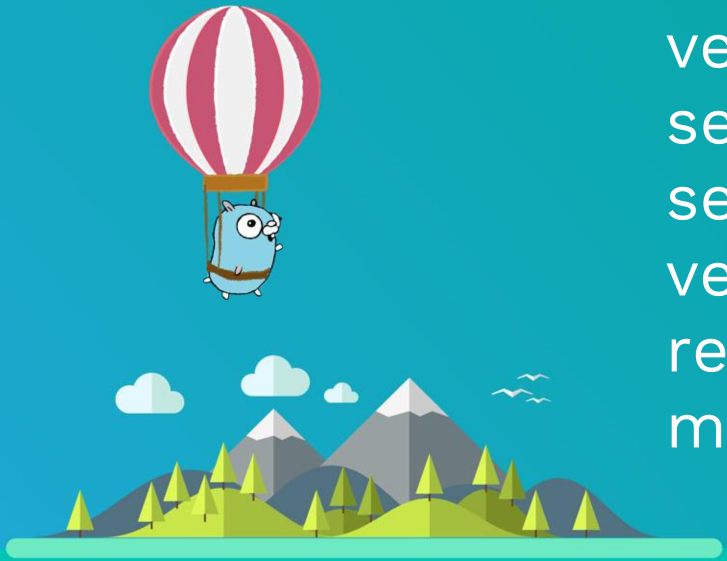
```
go mod init [module-path]
go mod tidy
go mod graph
go get -u[=patch]
go build
go test
```

go get (module remix)

```
go get github.com/gorilla/mux@latest
                        same (@latest is default for 'go get')

go get github.com/gorilla/mux@v1.6.2
                        records v1.6.2

go get github.com/gorilla/mux@e3702bed2
                        records v1.6.2

go get github.com/gorilla/mux@c856192
                        records v0.0.0-20180517173623-c85619274f5d

go get github.com/gorilla/mux@master
                        records current meaning of master
```

Live demo

Inner Workings



“For each module in a build, the version selected by minimal version selection is always the semantically highest of the versions explicitly listed by a require directive in the main module or one of its dependencies.”

Minimal Version Selection

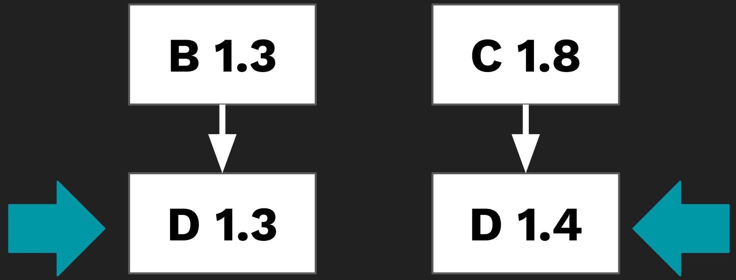
Minimal Version Selection

Requirements

B 1.3 D >= 1.3

C 1.8 D >= 1.4

D 1.4 none



Minimal Version Selection

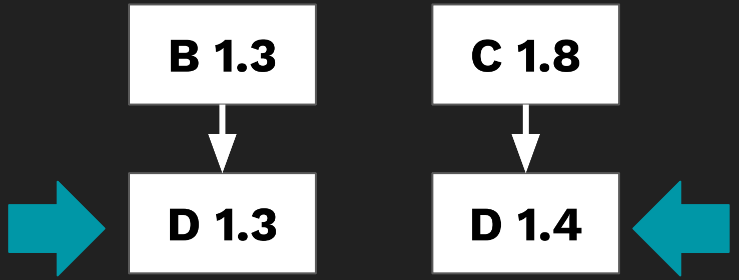
Requirements

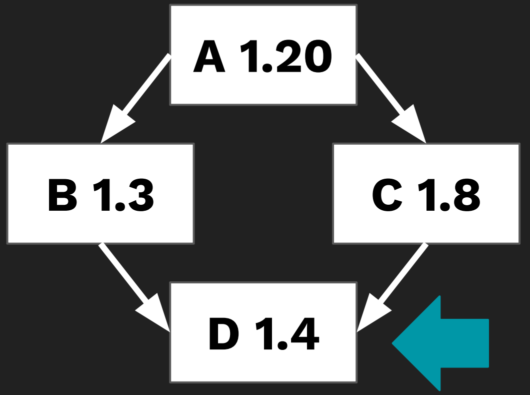
A 1.30 B >= 1.3, C >= 1.8

B 1.3 D >= 1.3

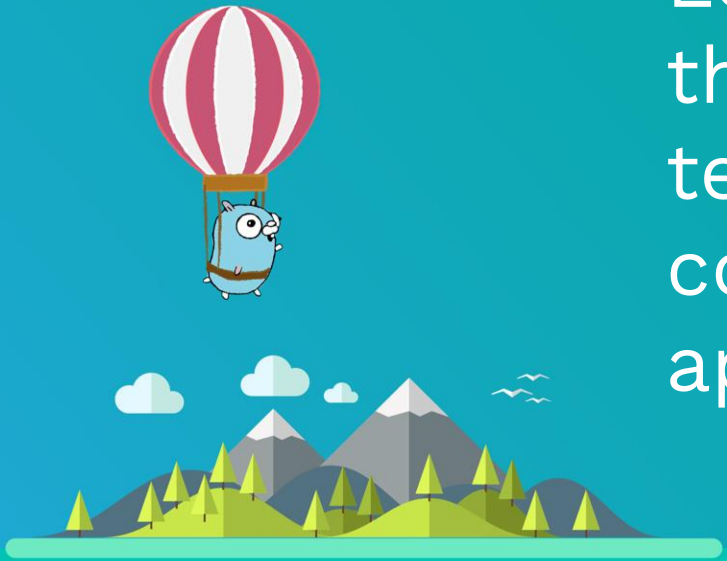
C 1.8 D >= 1.4

D 1.4 none





The Community



“Looking back, as I said at the start, we the core Go team mishandled the community process, and I apologize.”

Russ Cox

Modules

The Road Ahead

Release Plans

- Module support merged July 12
- Opt-in feature in Go 1.11 (end of Aug?)
- Default in Go 1.12

Ultimate Goals

- Add versions to the Go vocabulary of
  - Developers
  - Tools
- Break free of GOPATH
- Minimize need for vendor dirs
- Global Module Registry and Proxy with Athens Project

Links and Literature

- “Principles of Versioning in Go”, Russ Cox
- Blog Posts “Go & Versioning”, Russ Cox
- Golang Tips about Modules