Go Dependency Managers

Joey Gibson
joey@joeygibson.com
@joeygibson

Why use a dependency manager?

- Single list of dependencies for project
- Specify specific versions of dependencies, instead of just `master`
- Repeatable builds!
- No need to check dependencies into VCS

The List

- In roughly the order I heard about them
- godep
- govendor
- glide
- dep
- gb

godep

- https://github.com/tools/godep
- v79, released on 2/1/2017
- `godep save`
  - creates `Godeps` and `vendor` directories
  - both should be checked in to VCS
  - dependencies must already be in $GOPATH
- `godep get <dep>` to install other dependencies into $GOPATH and vendor

govendor

- https://github.com/kardianos/govendor
- 1.0.8, 9/21/2016
- `govendor init`
- creates vendor/vendor.json
    - check this one file into VCS
- `govendor fetch +missing` to fetch dependencies referenced, but not on GOPATH or in vendor
- `govendor fetch <dep>` to add a new dep
- `govendor sync` to pull down what's specified in vendor/vendor.json
- `govendor fetch golang.org/x/net/context@v1` gets the latest v1 build
- `govendor fetch golang.org/x/net/context@=v1` get the tag or branch with that name

glide

- https://github.com/Masterminds/glide
- 0.12.3, 10/3/2016
- `glide create`
    - creates glide.yaml, and vendor/
    - glide.yaml should be committed, not vendor/
- `glide install` will fetch dependencies into vendor/
    - creates glide.lock with current specific versions - check this file in to VCS
    - run `glide install` whenever you update glide.yaml
- `glide up` will update dependencies
- `glide get` will install additional dependencies
    - If dependency uses semver, it will offer to use the latest version
    - if not, will use SHA of head

dep - the “official” one

- https://github.com/golang/dep
- No release yet - still very alpha
- go get -u github.com/golang/dep/...
- `dep init`
    - ~~create manifest.json, lock.json~~ - files and formats still in flux
    - creates Gopkg.toml, Gopkg.lock - commit both to VC
    - adds deps from source to Gopkg.lock, but *not* to Gopkg.toml
- `dep ensure` pulls deps into vendor/
- `dep ensure github.com/sirupsen/logrus@^0.11.4`
    - adds constraints into Gopkg.toml
- `dep ensure github.com/spf13/cobra` will add entry to Gopkg.lock, but NOT to Gopkg.toml, and will give warning about it, since no version specified
- `dep ensure -update` updates all deps within constraints

Which to choose?

- I prefer glide, but I got strange errors sometimes when in a Docker container
- govendor gives fewest unexpected results
  - we use govendor at work
- I use glide for my personal projects

Other Options?

- gb
  - completely different source layout, not compatible with normal Go toolchain
    - `gb build all`, instead of `go build`
  - you still have to manually manage vendoring