Go 1.6 Vendor Third Party Packages

Background

- *go `import "pkg"` is mono-revision*
  - no notion of package version
- encouraged workflow is "*mono-repository*"
  - copy 3rd party dependencies into local repo
  - ***reproducible builds***

Background

- *`go import "pkg"` is mono-revision*
  - no notion of package version
- encouraged workflow is "*mono-repository*"
  - copy 3rd party dependencies into local repo
  - ***reproducible builds***
- upstream (theirs) and local (yours) may differ
  - local bug fix may not be pulled
  - upstream may change anytime

Example: fix/changes in go package

- forking: requires path rewriting
  - *before:* `github.com/<theirs>/package`
  - *after:* `github.com/<yours>/package`

Example: fix/changes in go package

- forking: requires path rewriting
  - *before:* `github.com/<theirs>/package`
  - *after:* `github.com/<yours>/package`
- incompatible with Pull Request

Example: fix/changes in go package

- forking: requires path rewriting
  - *before:* `github.com/<theirs>/package`
  - *after:* `github.com/<yours>/package`
- incompatible with Pull Request
- instead:
  - keep fix in vendor/ directory
  - `git remote add myne <your repo URL>`

Scope of Vendor Experiment

- pkgA depends on "*version 2*" of pkgB
  - i.e. "go get" will get wrong version
- put *copy* of v2 pkgB inside pkgA/vendor

Scope of Vendor Experiment

- pkgA depends on "*version 2*" of pkgB
  - i.e. "go get" will get wrong version
- put copy of v2 pkgB inside pkgA/vendor
- export `GO15VENDOREXPERIMENT=1`
  - Go 1.6 sets it to 1 by default
  - Go 1.7 will ignore `GO15VENDOREXPERIMENT`

Example: main needs hello.v2

```
$GOPATH/src/
├── my_main/
│   └── main.go
└── hello/
    └── hello.v5.go
```

Example: main needs hello.v2

```
$GOPATH/src/
└── my_main/
    ├── main.go
    └── vendor/
        └── hello/
            └── hello.v2.go
```

“Vendor” Tool Support

Godep

- github.com/tools/godep (kr)
  - used by Paas buildpacks (Heroku, Appfog)
- snapshots project dependencies
  - writes a *Godeps JSON* for all dependencies
  - can store sources in Godeps/_workspace
- supports vendor/ experiment
  - unless you already use a _workspace

gvt: Go Vendoring Tool

- creates & populates vendor/ folder for you
  - relies on GO15VENDOREXPERIMENT=1
- use inside a package within $GOPATH:
  - resolves & fetches dependencies
  - writes a manifest JSON file

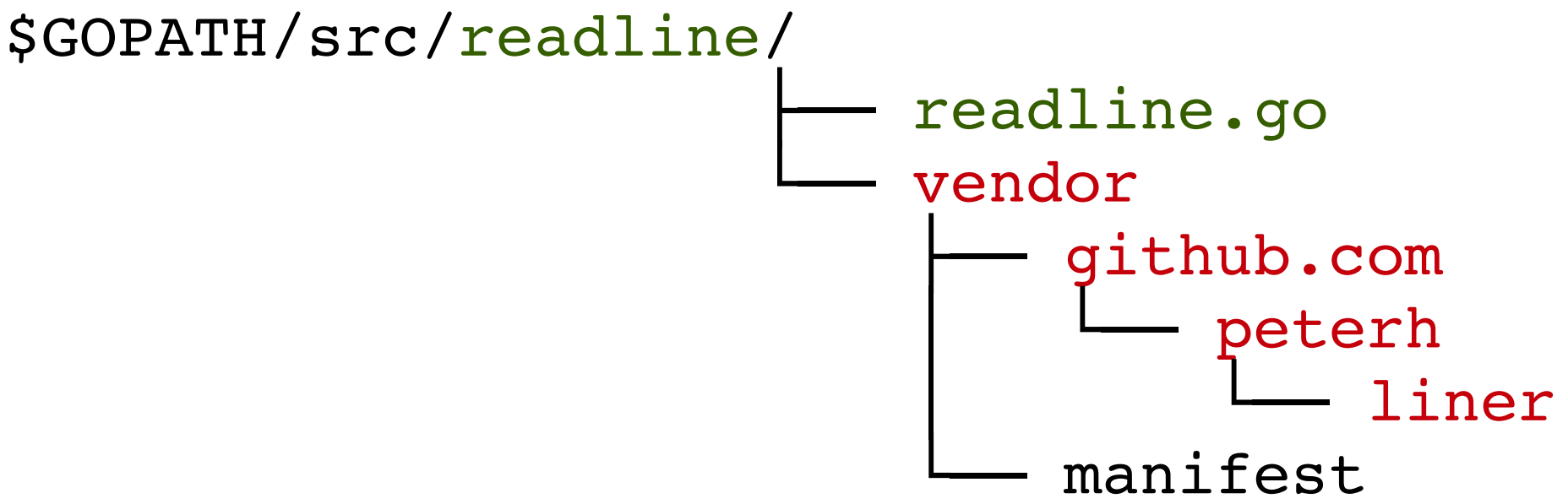
gvt Example

- src/readline depends on github.com/peterh/liner
- Result:

```
$GOPATH/src/readline/
                      ├── readline.go
                      └── vendor
                          ├── github.com
                          │   └── peterh
                          │       └── liner
                          └── manifest
```

gvt Example

- src/readline depends on github.com/peterh/liner
- Result:



govendor

- github.com/kardianos/govendor
  - similar to gvt
  - copies files into vendor/ folder
  - uses vendor/vendor.json

Example:

```
$ govendor init
$ govendor add github.com/peterh/liner
$ tree
   ├── readline.go
   └── vendor
       ├── github.com
       │   └── peterh
       │       └── liner
       │           ...
       └── vendor.json
```

govendor

- github.com/kardianos/govendor
  - similar to gvt
  - copies files into vendor/ folder
  - uses vendor/vendor.json

Example:

```
$ govendor init
$ govendor add github.com/peterh/liner
$ tree
   ├── readline.go
   └── vendor
       ├── github.com
       │   └── peterh
       │       └── liner
       │           ...
       └── vendor.json
```

gb project tool

- https://getgb.io/
  - opinionated project/build tool
  - builds without $GOPATH and go commands
  - influenced thinking about vendoring
- example:
  - `mkdir -p src/...`
  - `git clone ...`
  - `gb build all`
- supports vendoring via `gb-vendor`

Links

https://nathany.com/go-packages/

https://golang.org/s/go15vendor

https://github.com/golang/go/wiki/PackageManagementTools