Go Vendoring Deconstructed

Wisdom Omuya
MongoDB

What is Vendoring?

Agenda

- Introduction to Go vendoring
- Details of how it works
- Best practices in using vendoring

Project Layout

```
/home/projects/
   mongoproxy/
        main.go
        modules/
               mongod/
                      proxy.go
        vendor/
              src/
                     github.com/go-mgo/mgo
```

Project Layout

```
/home/projects/
   mongoproxy/
        main.go
        modules/
               mongod/
                      proxy.go
        vendor/
              src/
                     github.com/go-mgo/mgo
```

```
export GOPATH=/home/projects/mongoproxy
```

go build…

```
/home/projects/
   mongoproxy/
        main.go
        modules/
               mongod/
                      proxy.go
        vendor/
              src/
                     github.com/go-mgo/mgo
```

export GOPATH=/home/projects/mongoproxy

import "github.com/go-mgo/mgo"

Scaffolding

```
/home/projects/
   mongoproxy/
        .gopath/src/mongoproxy
        main.go
        modules/
              mongod/
                    proxy.go
        vendor/
             src/
                   github.com/go-mgo/mgo
```

Symlinking

```
/home/projects/
   mongoproxy/
        .gopath/src/mongoproxy
        main.go
        modules/
              mongod/
                    proxy.go
        vendor/
             src/
                   github.com/go-mgo/mgo
```

$GOPATH

```
/home/projects/
   mongoproxy/
        .gopath/src/mongoproxy
        main.go
        modules/
              mongod/
                    proxy.go
        vendor/
             src/
                   github.com/go-mgo/mgo
```

```
export GOPATH=`pwd`/.gopath:`pwd`/vendor
```

Build Script?

```
cd /home/projects/mongoproxy

rm -rf .gopath/

mkdir -p .gopath/src/mongoproxy

ln -sf `pwd` .gopath/src/mongoproxy

export GOPATH=`pwd`/.gopath:`pwd`/vendor

go build
```

Can we do better?

Vendoring

Go 1.5 includes experimental support for using local copies of external dependencies to satisfy imports of those dependencies, often referred to as vendoring...

Yeah!

How does this work?

Import resolution

- `$GOROOT`

Import resolution

- `$GOROOT`
- `$GOPATH`

Import resolution

- First in `vendor/` (in $GOPATH)

Import resolution

- First in `vendor/` (in $GOPATH)
- Then in parent's `vendor/` directory

Import resolution

- First in `vendor/` (in $GOPATH)
- Then in parent's `vendor/` directory

Import resolution

- First in `vendor/` (in $GOPATH)
- Then in parent's `vendor/` directory
- `$GOROOT`

Import resolution

- First in `vendor/` (in $GOPATH)
- Then in parent's `vendor/` directory
- `$GOROOT`
- `$GOPATH`

go1.4, 1.5, 1.6, 1.7 …

1.4: N/A
1.5: GO15VENDOREXPERIMENT
1.6: Default
1.7: Point of no Return

The Details...
http://bit.do/govendor

Vendoring: Detail #1

If there is a source directory d/vendor, then, when compiling a source file within the subtree rooted at d, import "p" is interpreted as import "d/vendor/p" if that path names a directory containing at least one file with a name ending in ".go".

Import Path Directory

```
export GOPATH=/home/projects

/home/projects/
    src/
        mongoproxy/
            main.go
            vendor/
                golang.org/x/net/context/
                        context.go
```

Import Path Directory

```
export GOPATH=/home/projects

/home/projects/
    src/
        mongoproxy/
            main.go
            vendor/
                golang.org/x/net/context/
                        context.go

import “mongoproxy/vendor/golang.org/x/net/context”
```

Import Path Directory

```
export GOPATH=/home/projects

/home/projects/
    src/
        mongoproxy/
            main.go
            vendor/
                golang.org/x/net/context/
                        context.go

import "mongoproxy/vendor/golang.org/x/net/context"
```

Import Path Directory

```
export GOPATH=/home/projects

/home/projects/
    src/
        mongoproxy/
            main.go
            vendor/
                golang.org/x/net/context/
                        context.go

import “golang.org/x/net/context”
```

Import Path Directory

```
export GOPATH=/home/projects

/home/projects/
    src/
        mongoproxy/
            main.go
            vendor/
                golang.org/x/net/context/ ✔
                        context.go

import “golang.org/x/net/context”
```

Import Path Directory

```
export GOPATH=/home/projects

/home/projects/
    src/
        mongoproxy/
            main.go
            vendor/
                golang.org/x/net/context
                        context.go

import “golang.org/x/net/context”
```

Import Path Directory

```
export GOPATH=/home/projects

/home/projects/
    src/
        mongoproxy/
            main.go
            vendor/
                golang.org/x/net/context
                        context.go

import “golang.org/x/net/context”
```

Import Path Directory

```
main.go:4:2: cannot find package "golang.org/x/net/
context" in any of:
    /home/projects/src/mongoproxy/vendor/
golang.org/x/net/context (vendor tree)
    /usr/local/opt/go/libexec/src/golang.org/x/net/
context(from $GOROOT)

/home/projects/golang.org/x/net/context (from
$GOPATH)
```

Vendoring: Detail #1

vendored import path must use short form

and contain at least one .go file

Vendoring: Detail #2

When there are multiple possible resolutions, the most specific (longest) path wins.

Multiple possible resolutions

```
$GOPATH/
    src/
        golang.org/x/net/context/
                context.go
        mongoproxy/
            main.go
            vendor/
                golang.org/x/net/context/
                        context.go
```

Multiple possible resolutions

```
$GOPATH/
    src/
        golang.org/x/net/context/
                context.go
        mongoproxy/
            main.go
            vendor/
                golang.org/x/net/context/
                        context.go
```

```
import “golang.org/x/net/context”
```

Multiple possible resolutions

```
$GOPATH/
    src/
        golang.org/x/net/context/
                context.go
        mongoproxy/
            main.go
            vendor/
                golang.org/x/net/context/
                        context.go
```

```
import “golang.org/x/net/context”
```

Multiple possible resolutions

```
$GOPATH/
    src/
        golang.org/x/net/context/
                context.go
        mongoproxy/
            main.go
            vendor/
                golang.org/x/net/context/ ✔
                        context.go
```

```
import “golang.org/x/net/context”
```

Vendoring: Detail #2

Most specific resolution path wins

`vendor/golang.org/x/net/context/`

is longer than

`golang.org/x/net/context/`

Vendoring: Detail #3

The resolution of an import must now take into account the location where that import path was found

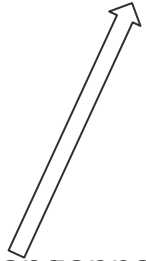
Import Location

```
$GOPATH/
    src/
        golang.org/x/net/context/
                context.go
        mongoproxy/
            main.go
            vendor/
                golang.org/x/net/context/
                        context.go
            server/
                server.go
                vendor/
                        golang.org/x/net/context/
                                context.go
```

```
import “mongoproxy/server”
```

Import Location

```
$GOPATH/
    src/
        golang.org/x/net/context/
                context.go
        mongoproxy/
            main.go
            vendor/
                golang.org/x/net/context/
                        context.go
            server/
                server.go
                vendor/
                        golang.org/x/net/context/
                                context.go
```



```
import “mongoproxy/server”
```

Import Location

```
$GOPATH/
    src/
        golang.org/x/net/context/
                context.go
        mongoproxy/
            main.go
            vendor/
                golang.org/x/net/context/
                        context.go
            server/
                server.go
                vendor/
                        golang.org/x/net/context/
                                context.go
```

```
import “golang.org/x/net/context”
```

Import Location

```
$GOPATH/
    src/
        golang.org/x/net/context/
                context.go
        mongoproxy/
            main.go
            vendor/
                golang.org/x/net/context/
                        context.go
            server/
                server.go
                vendor/
                        golang.org/x/net/context/
                                context.go
```

```
import “golang.org/x/net/context”
```

Import Location

```
$GOPATH/
    src/
        golang.org/x/net/context/
                context.go
        mongoproxy/
            main.go
            vendor/
                golang.org/x/net/context/
                        context.go
            server/
                server.go
                vendor/
                        golang.org/x/net/context/ ✔
                                context.go
```

```
import “golang.org/x/net/context”
```

Vendor Tree

```
$GOPATH/
   src/
        golang.org/x/net/context/
                context.go
        mongoproxy/
           main.go
           vendor/
                golang.org/x/net/context/
                        context.go
                server/
                   vendor/
                        webservice/
                                service.go
```

```
import “golang.org/x/net/context”
```

Vendor Tree

```
$GOPATH/
    src/
        golang.org/x/net/context/
                context.go
        mongoproxy/
            main.go
            vendor/
                golang.org/x/net/context/
                        context.go
                server/
                    vendor/
                        webservice/
                                service.go
```

```
import “golang.org/x/net/context”
```

Vendor Tree

```
$GOPATH/
    src/
        golang.org/x/net/context/
                context.go
        mongoproxy/
            main.go
            vendor/
                golang.org/x/net/context/ ✔
                        context.go
                server/
                    vendor/
                        webservice/
                                service.go
```

```
import “golang.org/x/net/context”
```

Vendoring: Detail #3

`Closest resolution always wins`

Vendoring: Detail #4

If someone wants to vendor (and therefore hide the standard library version of) "math" or even "unsafe", they can.

Will it compile?

```
$GOPATH/
   src/
       fmt/
           fmt.go
       mongoproxy/
           main.go
           vendor/
               github.com/go-mgo/mgo/
                       server.go
```

```
import “fmt”
```

Congratulations!

```
$GOPATH/
   src/
       fmt/ ✔
           fmt.go
       mongoproxy/
           main.go
           vendor/
               github.com/go-mgo/mgo/
                       server.go
```

```
import “fmt”
```

Vendoring: Detail #4

`Don’t shadow stdlib packages`

Vendoring Details

1. Vendored import path *must* use short form
2. Most specific resolution path wins
3. Closest resolution always wins
4. Don’t shadow stdlib packages

The Devil...

Vendored Interfaces

```
mongoproxy/
    mongoproxy.go
    vendor/
        github.com/go-mgo/mgo/
            server.go
```

```
type DBWriter interface {
        Write(b []byte) (n int, err error)
}
```

Vendored Interfaces

```
mongoproxy/
   mongoproxy.go
   vendor/
      github.com/go-mgo/mgo/
         server.go
```

```
type ProxyWriter interface {
    RouteWrite (w mgo.DBWriter) error
}
```

Vendored Interfaces

```
mongoproxy/
    mongoproxy.go
    vendor/
        github.com/go-mgo/mgo/
            server.go
```

```
type ProxyWriter interface {
    RouteWrite (w mgo.DBWriter) error
}
```

Vendored Interfaces

```
main/
    main.go
    vendor/
        github.com/go-mgo/mgo
            server.go
        mongoproxy/
            mongoproxy.go
            vendor/
                github.com/go-mgo/mgo/
                    server.go
```

Vendored Interfaces

main.go

```
import (
    “mongoproxy”
    “mgo”
)

var myProxyWriter mongoproxy.ProxyWriter = &proxyWriter{}
var mgoWriter mgo.DBWriter = &mgoWriter{}
myProxyWriter.RouteWrite(mgoWriter)
```

Vendored Interfaces

```
# command-line-arguments
./main.go:14:
        cannot use myProxyWriter literal (type *proxyWriter)
        as type mongoproxy.ProxyWriter in assignment:
        *myProxyWriter does not implement mongoproxy.ProxyWriter
        (wrong type for RouteWrite method)


        have RouteWrite("main/vendor/mgo".DBWriter) error
        want RouteWrite("mongoproxy/vendor/mgo".DBWriter) error
```

What it does not do

Go Vendoring

Does not prevent duplicate imports

Does not manage your dependencies

Vendoring Package Management

glide
gb

Best Practices

- For libraries, don’t expose - via interfaces, functions, etc - any vendored types
- Avoid vendoring multiple versions of the same package
- Use proven vendor management tooling to manage your dependencies - e.g. glide, gb, etc

Conclusion

Go Vendoring

Does away with import path rewrites

Does resolve multiple candidates

Does provide a standard

Pre: Working Directory

```
/home/projects/
   mongoproxy/
        .gopath/src/mongoproxy
        main.go
        modules/
              mongod/
                    proxy.go
        vendor/
             src/
                   github.com/go-mgo/mgo
```

```
export GOPATH=`pwd`/.gopath:`pwd`/vendor
```

Avoid Symlinking

```
/home/projects/
   mongoproxy/
        .gopath/src/mongoproxy
        main.go
        modules/
              mongod/
                    proxy.go
        vendor/
             src/
                   github.com/go-mgo/mgo
```

```
export GOPATH=`pwd`/.gopath:`pwd`/vendor
```

Avoid Scaffolding

```
/home/projects/src
   mongoproxy/
         .gopath/src/mongoproxy
         main.go
         modules/
                mongod/
                       proxy.go
         vendor/
               src/
                      github.com/go-mgo/mgo
```

```
export GOPATH=`pwd`/.gopath:`pwd`/vendor
```

Avoid GOPATH rewrite

```
/home/projects/src
   mongoproxy/
        main.go
        modules/
               mongod/
                      proxy.go
        vendor/
              github.com/go-mgo/mgo
```

```
export GOPATH=/home/projects
```

Pre: Working Directory

```
/home/projects/
   mongoproxy/
        .gopath/src/mongoproxy
        main.go
        modules/
              mongod/
                    proxy.go
        vendor/
             src/
                   github.com/go-mgo/mgo
```

```
export GOPATH=`pwd`/.gopath:`pwd`/vendor
```

Post: Working Directory

```
/home/projects/src
   mongoproxy/
        main.go
        modules/
               mongod/
                      proxy.go
        vendor/
              github.com/go-mgo/mgo
```

```
export GOPATH=/home/projects
```

Pre: build script

```
cd /home/projects/mongoproxy
rm -rf .gopath/
mkdir -p .gopath/src/mongoproxy
ln -sf `pwd` .gopath/src/mongoproxy
export GOPATH=`pwd`/.gopath:`pwd`/vendor
go build
```

Post: simple command

```
GOPATH=/home/projects go build
```

Thank You!

wisdom@mongodb.com