

Embracing the Go Standard Library

A Hero's Quest

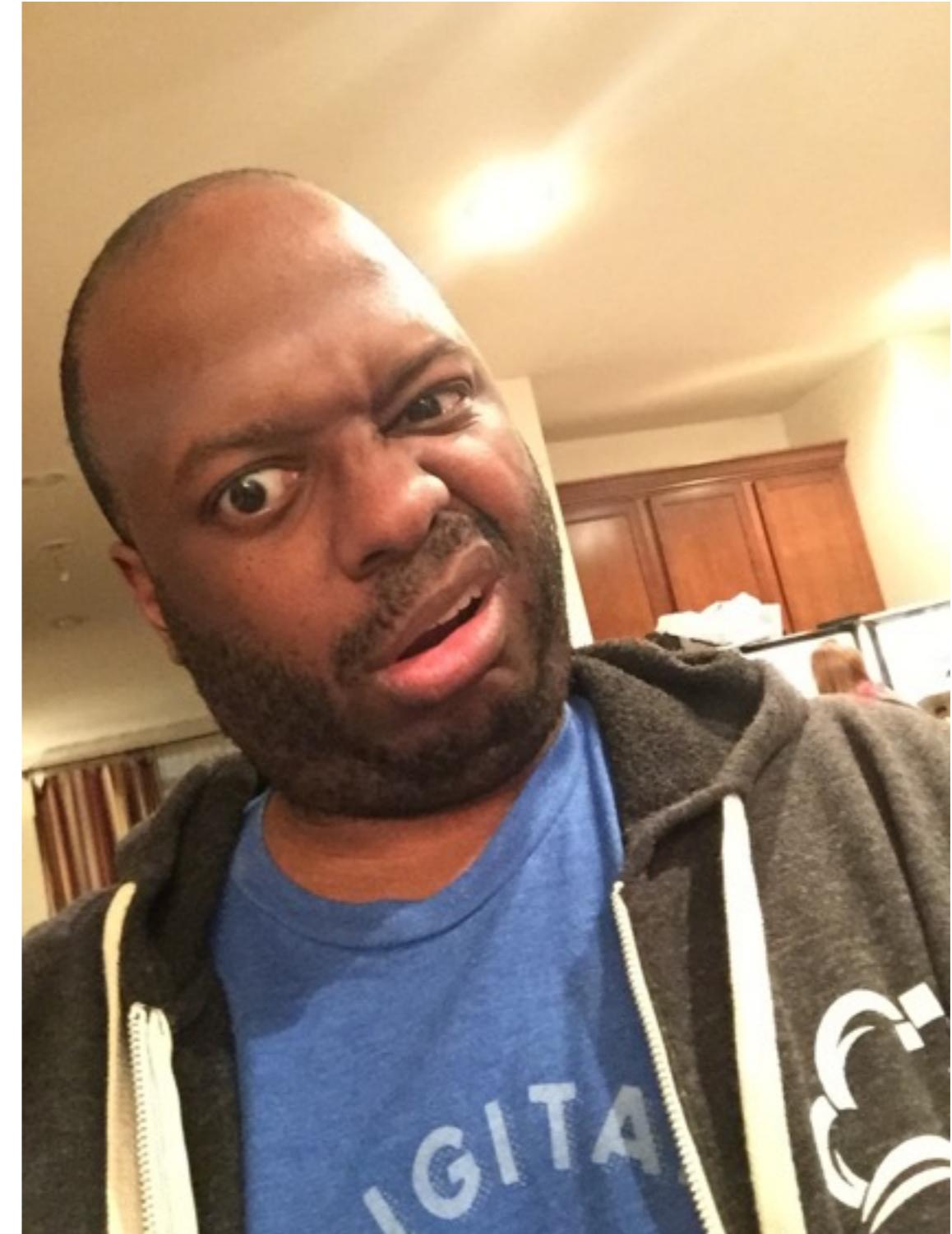
Today's agenda

- An introduction
- My standard library manifesto
- A rant
- No cat pictures
- Some code
- Some new stuff you might not have known
- Some old stuff you may have known
- Pop Quiz!
- Some kind of wrap up. Maybe? Maybe Not.
- 5 minutes of esoteric questions for Bryan

whoami:

☁ engineer
@bryanl
@digitalocean

polyglot, lover
of fine hiphop



Why is the Go standard library so important?

Code Reading

patterns in structure

```
import  
const  
var  
type  
constructors  
functions
```

how terse is too terse?

patterns with errors

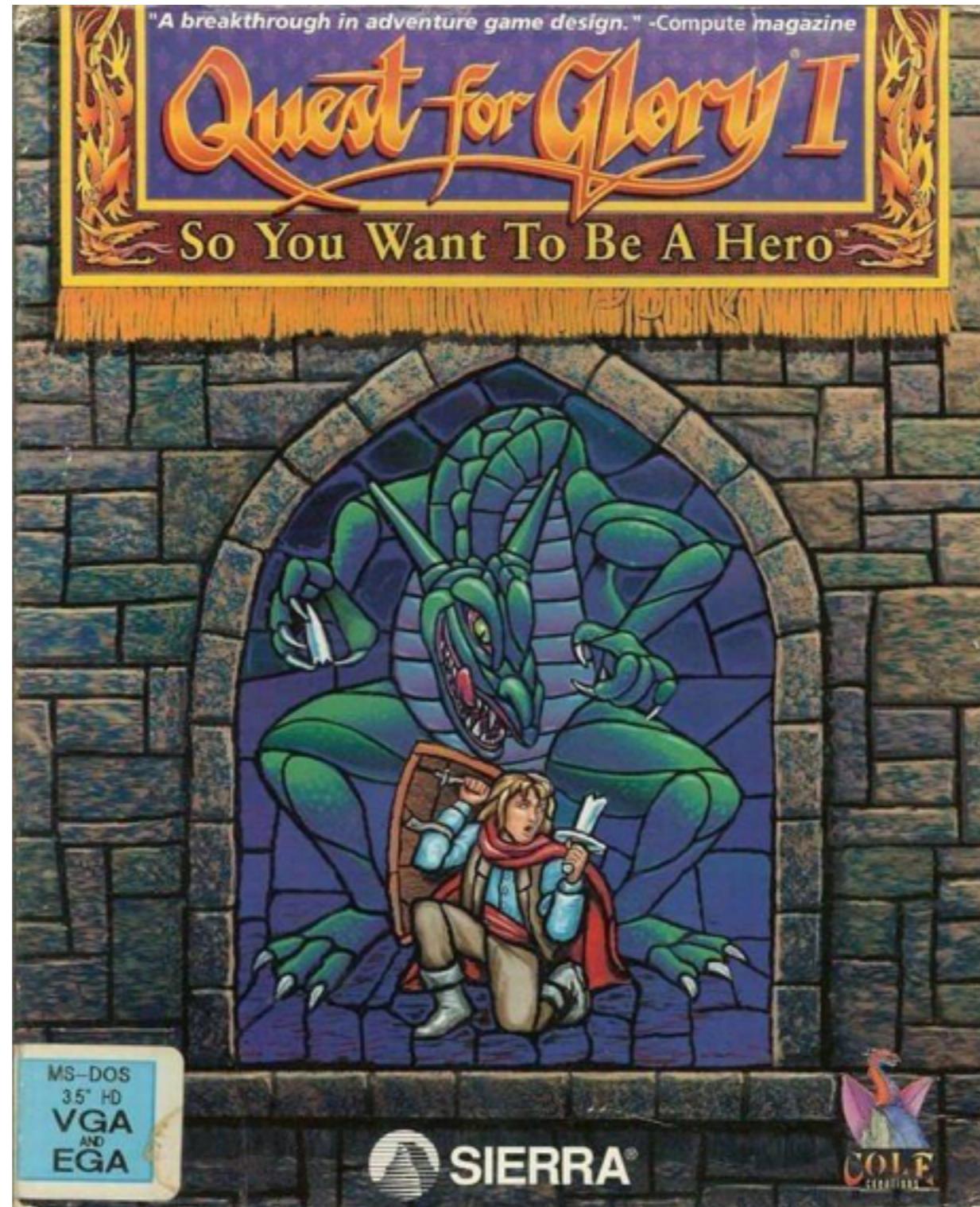
error is an interface

```
type error interface {  
    Error() string  
}
```

use type switching to differentiate
between errors

```
if err != nil {  
    return err  
}
```

```
if err != nil {  
    switch e := err.(type) {  
        case *encodingError:  
        case *transportError:  
        case *gravityError:  
        case *omgWTFBBQError:  
        default:  
    }  
}
```



Using the standard library

http client

```
package main

import (
    "fmt"
    "net/http"
)

func handler(w http.ResponseWriter, r *http.Request) {
    fmt.Fprintf(w, "hi!")
}

func main() {
    http.HandleFunc("/", handler)
    http.ListenAndServe(":8080", nil)
}
```

```
2.. zsh_tmux.plugin.run-new.zshmod
hello1
l
```

3 < ↑ 1d 19h 52m < 2.4 2.4 2.1 < 2015-02-07 < 09:17 Macintosh.home



IT BUILDS? SHIP IT!

WE'LL JUST BLAME ALIENS

TROLL.ME®

```
package main

import (
    "fmt"
    "log"
    "net/http"
)

func stats(h http.Handler) http.Handler {
    return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
        h.ServeHTTP(w, r)
        log.Println("(request stats)")
    })
}

func hiHandler(w http.ResponseWriter, r *http.Request) {
    fmt.Fprintf(w, "hi!")
}

func main() {
    hi := http.HandlerFunc(hiHandler)
    http.Handle("/", stats(hi))
    http.ListenAndServe(":8080", nil)
}
```

```
package main

import (
    "fmt"
    "log"
    "net/http"
)

func stats(h http.Handler) http.Handler {
    return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
        h.ServeHTTP(w, r)
        log.Println("(request stats)")
    })
}

func hiHandler(w http.ResponseWriter, r *http.Request) {
    fmt.Fprintf(w, "hi!")
}

func main() {
    hi := http.HandlerFunc(hiHandler)
    http.Handle("/", stats(hi))
    http.ListenAndServe(":8080", nil)
}
```

```
package main

import (
    "fmt"
    "log"
    "net/http"
)

func stats(h http.Handler) http.Handler {
    return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
        h.ServeHTTP(w, r)
        log.Println("(request stats)")
    })
}

func hiHandler(w http.ResponseWriter, r *http.Request) {
    fmt.Fprintf(w, "hi!")
}

func main() {
    hi := http.HandlerFunc(hiHandler)
    http.Handle("/", stats(hi))
    http.ListenAndServe(":8080", nil)
}
```

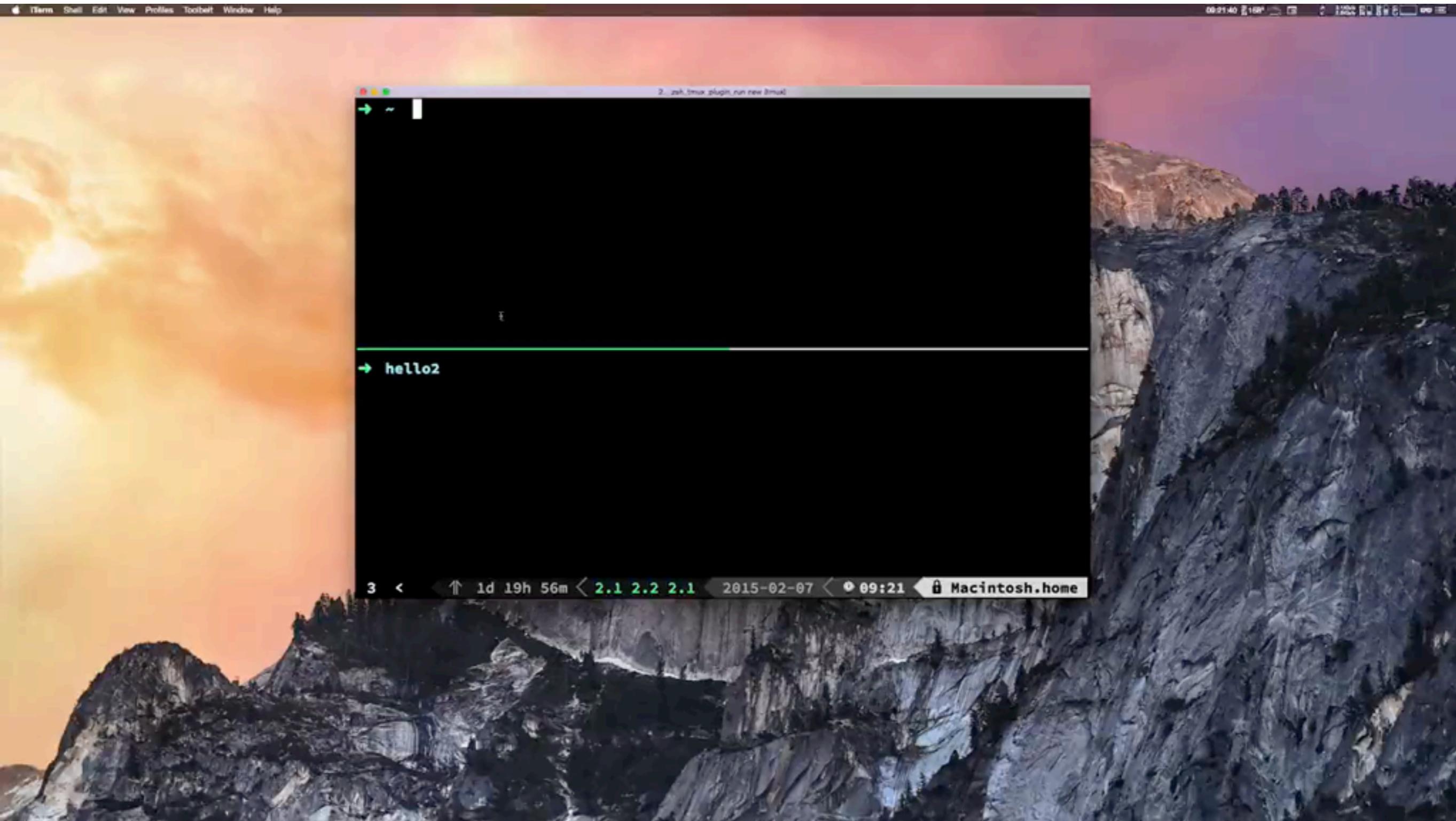
```
package main

import (
    "fmt"
    "log"
    "net/http"
)

func stats(h http.Handler) http.Handler {
    return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
        h.ServeHTTP(w, r)
        log.Println("(request stats)")
    })
}

func hiHandler(w http.ResponseWriter, r *http.Request) {
    fmt.Fprintf(w, "hi!")
}

func main() {
    hi := http.HandlerFunc(hiHandler)
    http.Handle("/", stats(hi))
    http.ListenAndServe(":8080", nil)
}
```



```
func main() {
    hi := http.HandlerFunc(hiHandler)
    http.Handle("/", m1(m2(m3(m4(m5(m6(m7(m8(m9(hi)))))))))))
}
```

```
type app struct{ handler http.Handler }
type handler func(http.Handler) http.Handler

func newApp(h http.Handler) *app {
    return &app{handler: h}
}

func (a *app) middleware(h handler) {
    a.handler = h(a.handler)
}

func (a *app) ServeHTTP(w http.ResponseWriter, r *http.Request) {
    a.handler.ServeHTTP(w, r)
}
```

```
type app struct{ handler http.Handler }
type handler func(http.Handler) http.Handler

func newApp(h http.Handler) *app {
    return &app{handler: h}
}

func (a *app) middleware(h handler) {
    a.handler = h(a.handler)
}

func (a *app) ServeHTTP(w http.ResponseWriter, r *http.Request) {
    a.handler.ServeHTTP(w, r)
}
```

```
type app struct{ handler http.Handler }
type handler func(http.Handler) http.Handler

func newApp(h http.Handler) *app {
    return &app{handler: h}
}

func (a *app) middleware(h handler) {
    a.handler = h(a.handler)
}

func (a *app) ServeHTTP(w http.ResponseWriter, r *http.Request) {
    a.handler.ServeHTTP(w, r)
}
```

```
type app struct{ handler http.Handler }
type handler func(http.Handler) http.Handler

func newApp(h http.Handler) *app {
    return &app{handler: h}
}

func (a *app) middleware(h handler) {
    a.handler = h(a.handler)
}

func (a *app) ServeHTTP(w http.ResponseWriter, r *http.Request) {
    a.handler.ServeHTTP(w, r)
}
```

```
type app struct{ handler http.Handler }
type handler func(http.Handler) http.Handler

func newApp(h http.Handler) *app {
    return &app{handler: h}
}

func (a *app) middleware(h handler) {
    a.handler = h(a.handler)
}

func (a *app) ServeHTTP(w http.ResponseWriter, r *http.Request) {
    a.handler.ServeHTTP(w, r)
}
```

```
func auth(h http.Handler) http.Handler {
    return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
        log.Println("(doing auth)")
        h.ServeHTTP(w, r)
    })
}

func stats(h http.Handler) http.Handler {
    return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
        h.ServeHTTP(w, r)
        log.Println("(request stats)")
    })
}
```

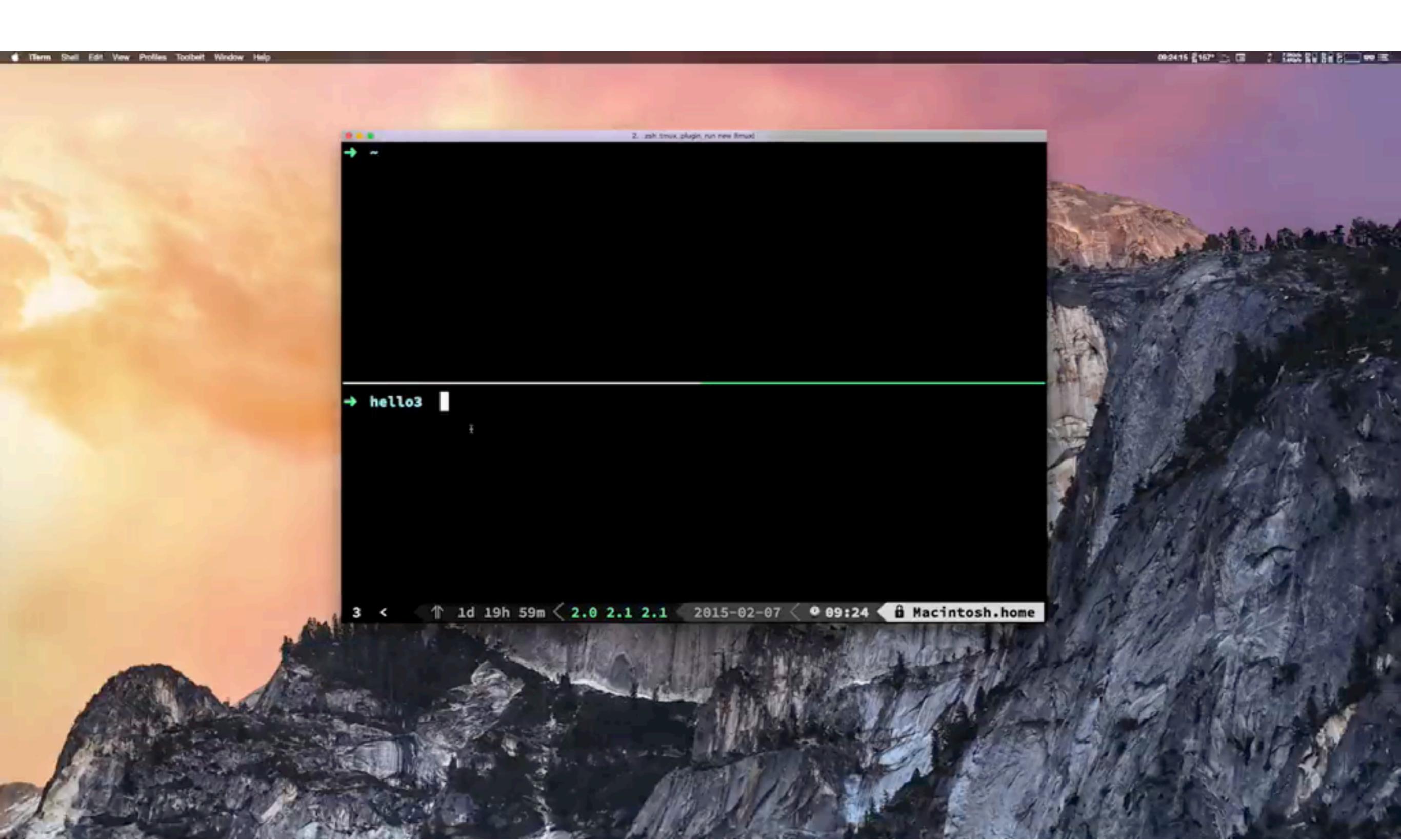
```
func auth(h http.Handler) http.Handler {
    return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
        log.Println("(doing auth)")
        h.ServeHTTP(w, r)
    })
}

func stats(h http.Handler) http.Handler {
    return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
        h.ServeHTTP(w, r)
        log.Println("(request stats)")
    })
}
```

```
func hiHandler(w http.ResponseWriter, r *http.Request) {
    fmt.Fprintf(w, "hi!")
}

func main() {
    hi := newApp(http.HandlerFunc(hiHandler))
    hi.middleware(auth)
    hi.middleware(stats)

    http.Handle("/", hi)
    http.ListenAndServe(":8080", nil)
}
```



digitalocean.com

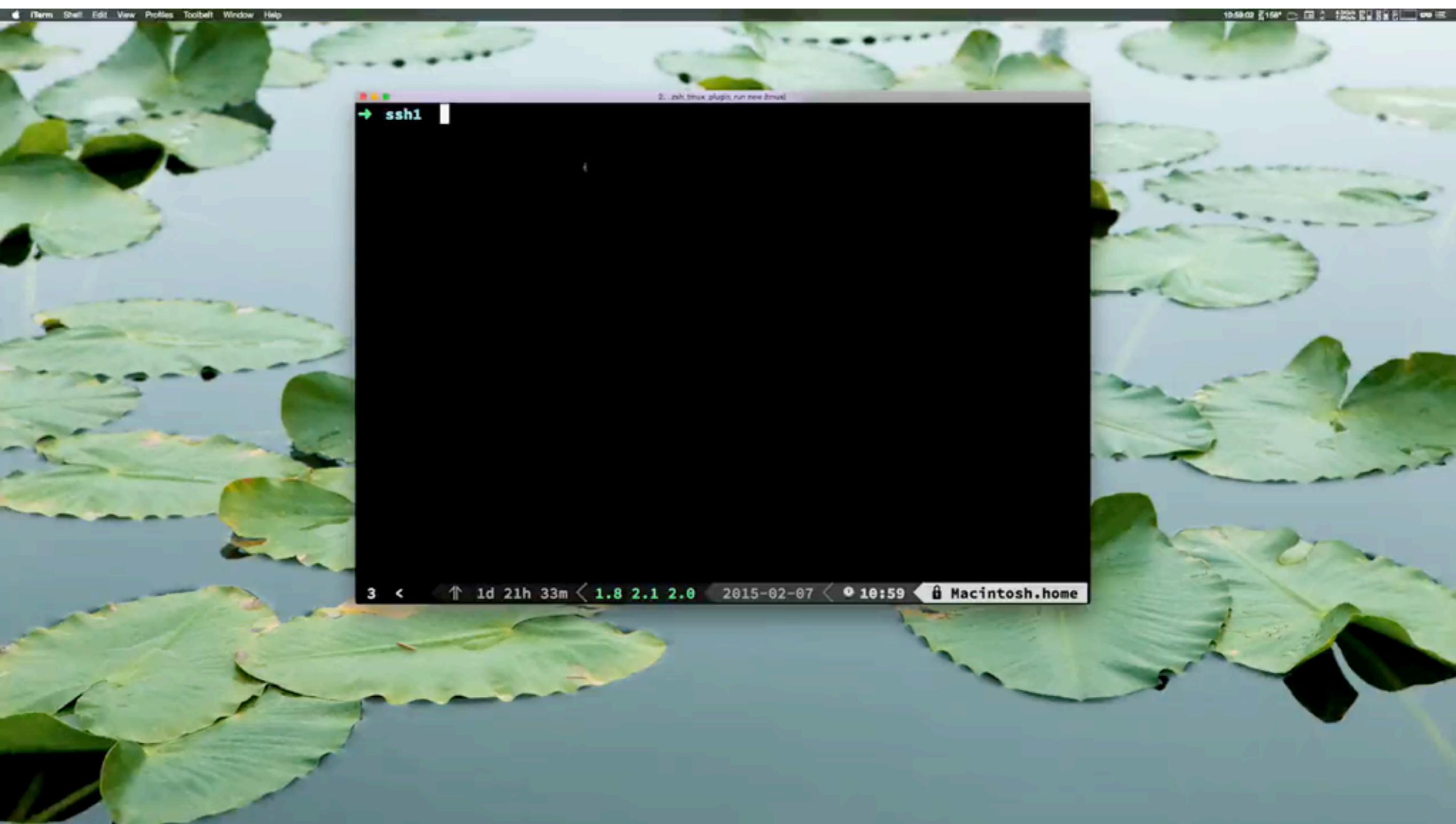
building a ssh client

```
package main

import (
    "fmt"
    "log"
    "os/exec"
)

func main() {
    output, err := exec.Command("ssh",
        "root@104.131.9.72",
        "hostname").Output()
    if err != nil {
        log.Fatalf("run: %v", err)
    }

    fmt.Println(string(output))
}
```



```
func parseKey() (ssh.Signer, error) {
    b, err := ioutil.ReadFile(os.Getenv("KEY_PATH"))
    if err != nil {
        return nil, fmt.Errorf("load private key: %v", err)
    }

    key, err := ssh.ParsePrivateKey(b)
    if err != nil {
        return nil, fmt.Errorf("parse private key: %v", err)
    }

    return key, nil
}
```

```
config := &ssh.ClientConfig{
    User: "root",
    Auth: []ssh.AuthMethod{
        ssh.PublicKeys(signer),
    },
}
```

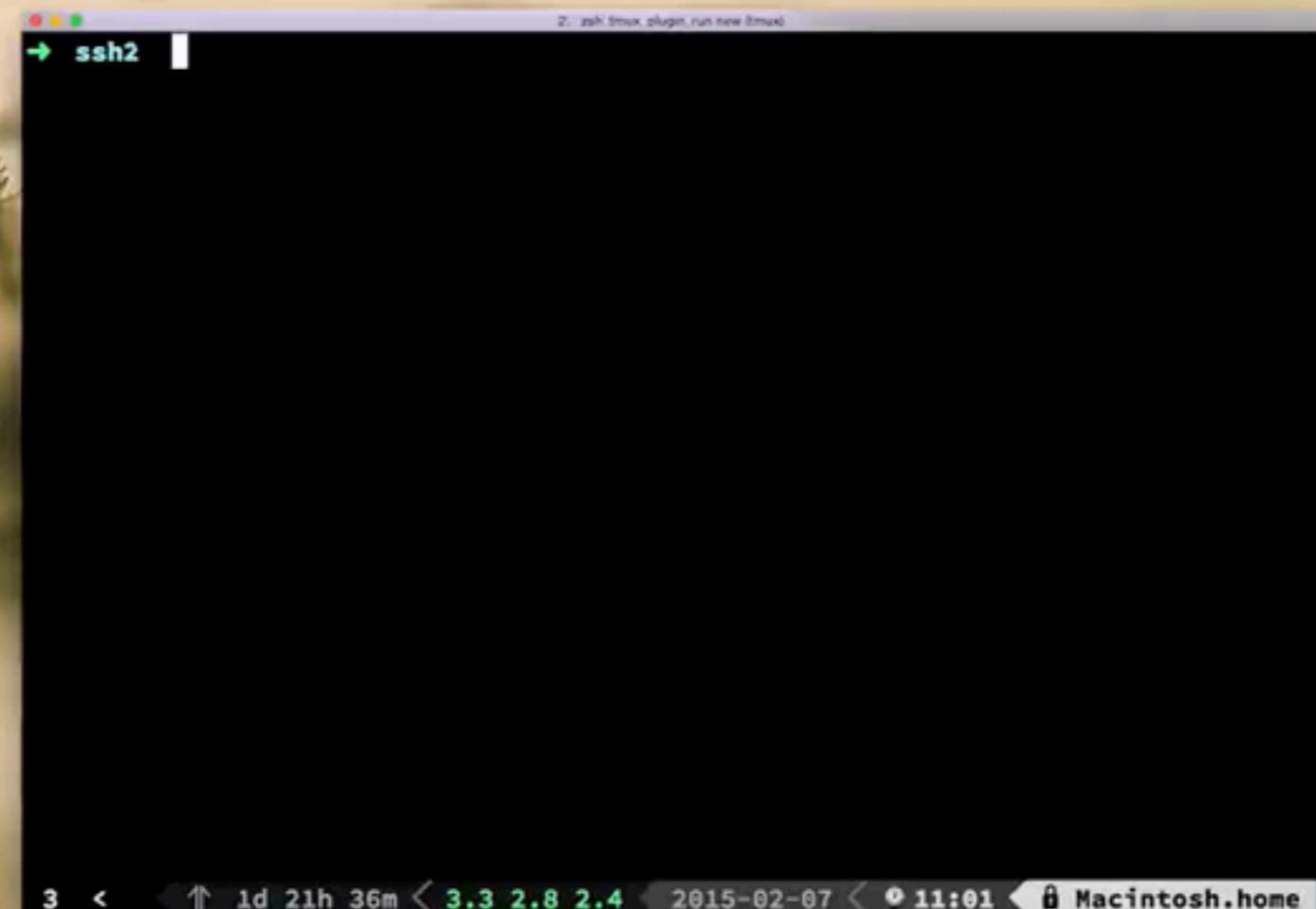
```
client, err := ssh.Dial("tcp", "104.131.9.72:22", config)
if err != nil {
    log.Fatalf("dial: %v", err)
}

sess, err := client.NewSession()
if err != nil {
    log.Fatalf("create session: %v", err)
}

defer sess.Close()

b := bytes.Buffer{}
sess.Stdout = &b
if err := sess.Run("hostname"); err != nil {
    log.Fatalf("run command: %v", err)
}

fmt.Println(b.String())
```



code rewriting

```
var src = `
// Package main is main package.
package main

// we need the fmt
import "fmt"

// main is main
func main() {
    // this prints hello, world
    fmt.Println("hello, world") // all done!
}

`
```

```
func main() {
    fset := token.NewFileSet()
    f, err := parser.ParseFile(fset, "", src,
        parser.ParseComments)
    if err != nil {
        log.Fatalf("ParseFile: %v", err)
    }

    for _, c := range f.Comments {
        for _, l := range c.List {
            fmt.Printf("%+v: %s\n", l.Slash, l.Text)
        }
    }
}
```

2. zsh tmux plugin run new tmux

→ rewrite1

3 < ↑ 1d 22h 19m < 1.9 2.1 2.0 < 2015-02-07 < ⌂ 11:44 ↵ Macintosh.home

```
var src = `
// Package main is main package.
package main

// we need the fmt
import "fmt"

// main is main
func main() {
    // this prints hello, world
    fmt.Println("hello, world") // all done!
}

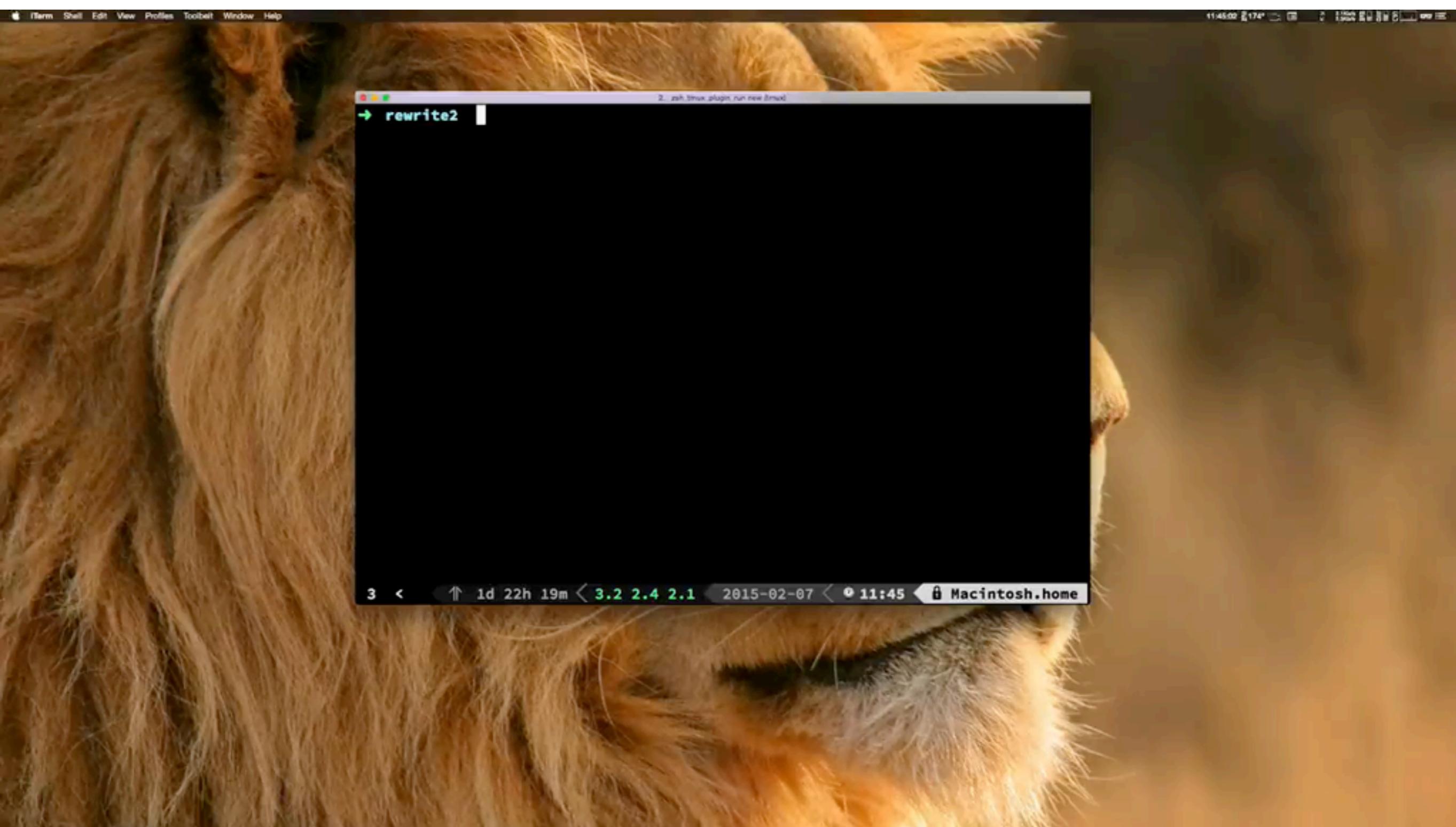
`
```

```
func main() {
    fset := token.NewFileSet()
    f, err := parser.ParseFile(fset, "", src,
        parser.ParseComments)
    if err != nil {
        log.Fatalf("ParseFile: %v", err)
    }

    f.Comments = nil

    var b bytes.Buffer
    if err := format.Node(&b, fset, f); err != nil {
        log.Fatalf("Format: %v", err)
    }

    fmt.Println(b.String())
}
```



but wait, there's more!

MacVim File Edit Tools Syntax Buffers Plugin Window Help 23:23:08 146° 2 3.1Mbps 107.900ms

main.go (~/do/cthulhu/docode/src/services/migration/cmd/migrationapi) - VIM

```
1 // Package main for the migrationapi binary.
2 package main
3
4 import (
5     "net/http"
6     "strings"
7     "time"
8
9     "doge/env"
10    "doge/log"
11    "doge/sentry"
12    "doge/status"
13
14    "services/migration"
15    "services/migration/alpha"
16    "services/migration/api"
17
18    "github.com/ianschenck/envflag"
19    "github.com/stretchr/graceful"
20 )
21
22 func main() {
23     var (
24         alphROUser    = envflag.String("MIGRATE_ALPHA_RO_USER", "root", "Alpha RO DB User")
25         alphROPass    = envflag.String("MIGRATE_ALPHA_RO_PASS", "password", "Alpha RO DB Password")
26         alphROAddr    = envflag.String("MIGRATE_ALPHA_RO_ADDR", "127.0.0.1:3306", "Alpha RO DB Addr")
27         alphROName    = envflag.String("MIGRATE_ALPHA_RO_NAME", "digitalocean", "Alpha RO DB Name")
28         databaseConns = envflag.Int("MIGRATE_ALPHA_MAX_CONN", 2, "max connections for local datastore")
29         addr          = envflag.String("MIGRATE_API_ADDR", ":8888", "listen address")
30     )
31     envflag.Parse()
32
33     status.InitStatusHTTP()
34
35     err := sentry.Init()
36     if err != nil {
37         log.WithError(err).Fatal("Unable to initialize sentry")
38     }
39
40     // Log everything in development mode
41     if env.IsDevelopment() {
42         log.SetLevel(log.Ldebug)
43     } else {
44         log.AddHook(
45             sentry.LogHook("services/migration"),
46             log.Lwarn, log.Lerror, log.Lfatal, log.Lpanic,
47         )
48     }
49
50     dsn := alpha.BuildDSN(*alphROUser, *alphROPass, *alphROName, *alphROAddr)
51     config := alpha.Config{DataDSN: dsn, MaxConn: *databaseConns}
52     dodb, err := alpha.NewDoDB(config)
53     if err != nil {
54         log.WithError(err).WithFields(log.Fields{
55             "alpha.dsm": dsn,
56         }).Fatal("Unable to connect to the Alpha DB")
57     }
58 }
```

NORMAL > master > ./main.go unix < utf-8 < go 15 1:8

**Don't underestimate
the tooling you have**

**Don't forget about
the new stuff**

So You Want To Be A Hero [score 13 of 500]

Hero Status

HP 

SP 

Enemy Status

HP 

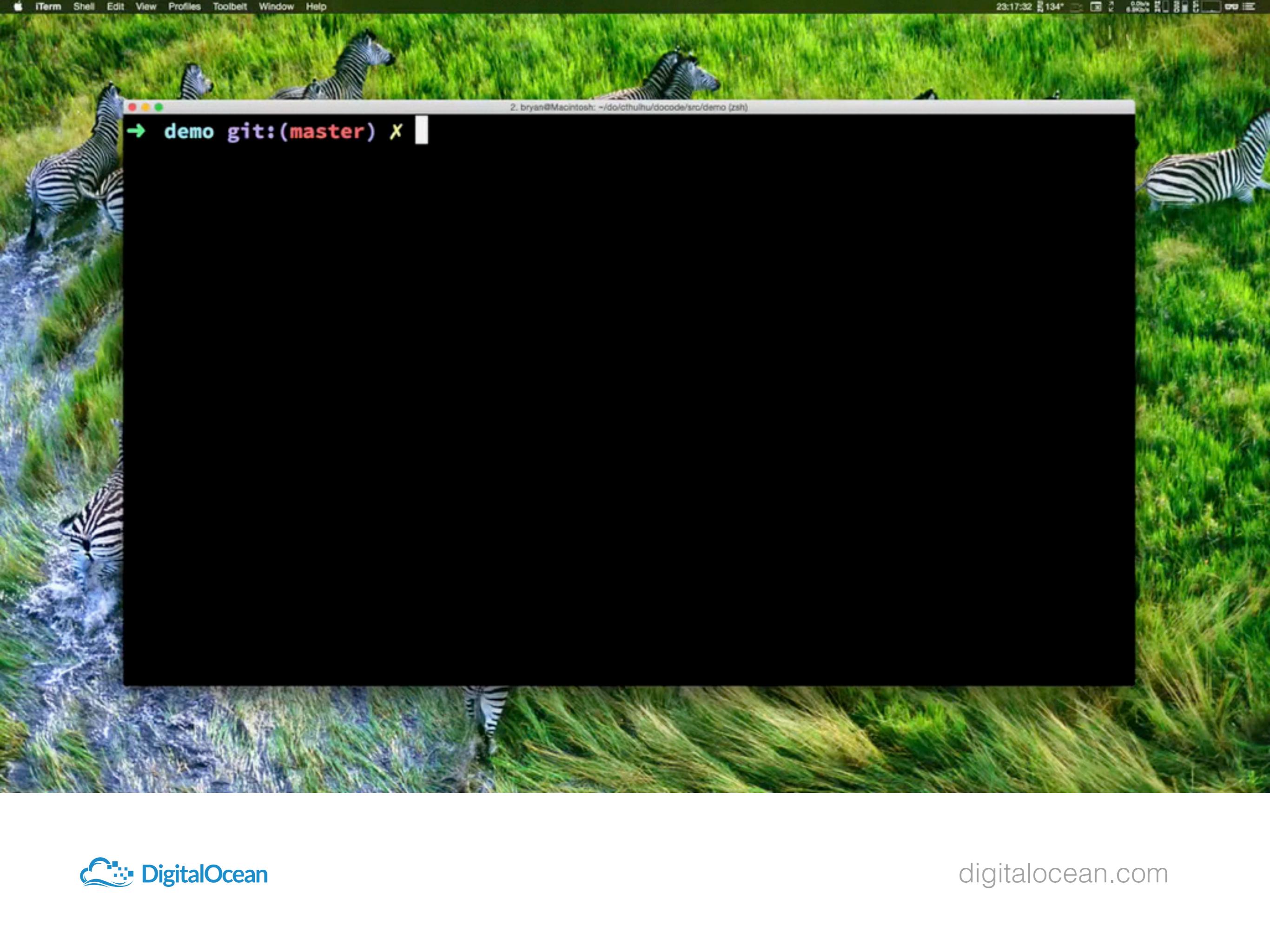


TestMain

```
func TestMain(m *testing.Main) { os.Exit(m.Run()) }
```

```
func TestMain(m *testing.Main) {
    setup()
    exitCode := m.Run()
    shutdown()
    os.Exit(exitCode)
}
```

Code generation



demo git:(master) X

2. bryan@Macintosh: ~/do/cthulhu/docode/src/demo (zsh)

Quest for Glory I

[Score: 21 of 500]

Hero Status	
HP	<div style="width: 50%;">█</div>
SP	<div style="width: 20%;">█</div>
MP	<div style="width: 100%;">██████████</div>

Enemy Status	
HP	<div style="width: 100%;">██████████</div>



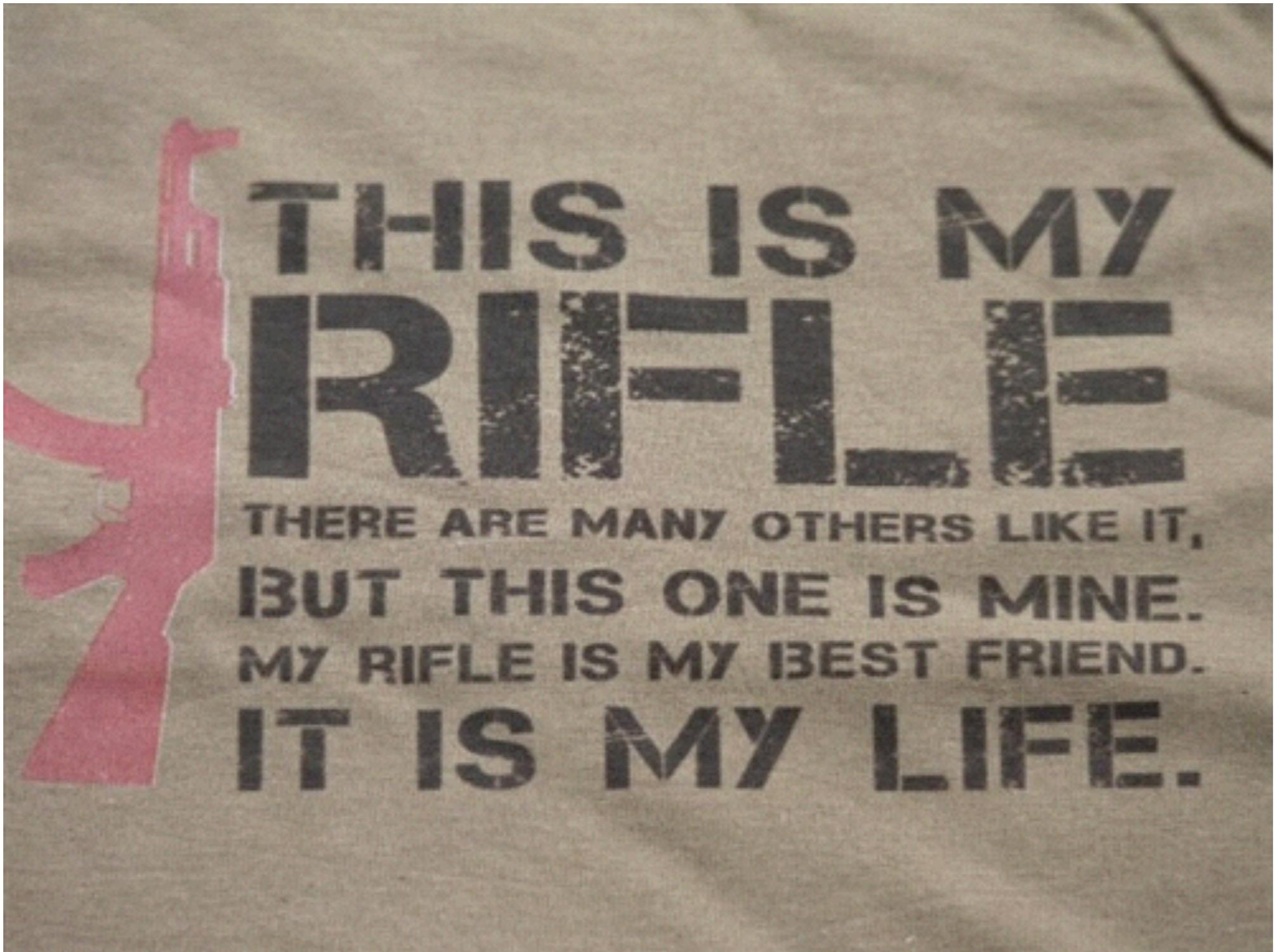
Go isn't boring...

Go could be compared to your **soulmate**. It understands you as a **developer**, but also expects you to understand it as a **programming language**. Go will love you as well as any programming language can by offering **simple constructs** and a **stable standard library**. It also isn't difficult to learn. It will push back when you try to bend its boundaries, but that's a good thing. **You don't want your programming language partner to allow you to bend it to your will.**

<http://blil.es/go-is-not-boring/>

productivity over features

Lessons for neophytes...



editing environment

shell environment

2 minutes on how DigitalOcean does Go

A screenshot of a Mac OS X Finder window titled "cthulhu". The window displays a file list with columns for Name, Date Modified, Size, and Kind. The "Name" column uses a blue folder icon for folders and a white document icon with a blue border for the "README.md" file. The "Date Modified" column shows dates ranging from "1/20/15, 4:47 PM" to "Yesterday, 10:49 AM". The "Size" column shows "5 KB" for the "README.md" file. The "Kind" column shows "Markdown" for the "README.md" file and "Folder" for all other items.

Name	Date Modified	Size	Kind
third_party	Yesterday, 10:49 AM	--	Folder
src	12/22/14, 12:09 PM	--	Folder
pkg	Yesterday, 10:13 AM	--	Folder
bin	Yesterday, 10:49 AM	--	Folder
README.md	1/20/15, 4:47 PM	5 KB	Markdown
docode	Yesterday, 10:16 AM	--	Folder
src	1/11/15, 10:59 AM	--	Folder
pkg	Yesterday, 10:13 AM	--	Folder
bin	Yesterday, 10:19 AM	--	Folder

Macintosh HD ▶ Users ▶ bryan ▶ do ▶ cthulhu ▶ third_party

A screenshot of a Mac OS X Finder window titled "cthulhu". The window displays a file tree under the "Name" column. The structure is as follows:

- third_party (Folder, Today, 12:47 PM)
- src (Folder, 12/22/14, 12:09 PM)
 - gopkg.in (Folder, 12/22/14, 2:14 PM)
 - golang.org (Folder, 12/9/14, 1:29 PM)
 - github.internal.digitalocean.com (Folder, 12/6/14, 4:39 PM)
 - github.com (Folder, Today, 12:45 PM)
 - code.google.com (Folder, 12/6/14, 4:39 PM)
 - bitbucket.org (Folder, 12/6/14, 4:39 PM)
 - pkg (Folder, Yesterday, 10:13 AM)
 - bin (Folder, Yesterday, 10:49 AM)

The status bar at the bottom shows the path: Macintosh HD > Users > bryan > do > cthulhu > docode > src.

Name	Date Modified	Size	Kind
third_party	Today, 12:47 PM	--	Folder
src	12/22/14, 12:09 PM	--	Folder
gopkg.in	12/22/14, 2:14 PM	--	Folder
golang.org	12/9/14, 1:29 PM	--	Folder
github.internal.digitalocean.com	12/6/14, 4:39 PM	--	Folder
github.com	Today, 12:45 PM	--	Folder
code.google.com	12/6/14, 4:39 PM	--	Folder
bitbucket.org	12/6/14, 4:39 PM	--	Folder
pkg	Yesterday, 10:13 AM	--	Folder
bin	Yesterday, 10:49 AM	--	Folder

A screenshot of a Mac OS X Finder window titled "cthulhu". The window displays a file list with columns for Name, Date Modified, Size, and Kind. The "Name" column uses blue folder icons for folders and a white document icon with "MD" for a Markdown file. The "Date Modified" column shows dates like "Yesterday, 10:13 AM" and "1/20/15, 4:47 PM". The "Size" column shows sizes like "--" and "5 KB". The "Kind" column shows types like "Folder" and "Markdown". A blue bar highlights the "src" folder under "decode". The file list includes:

Name	Date Modified	Size	Kind
▶ pkg	Yesterday, 10:13 AM	--	Folder
▶ bin	Yesterday, 10:49 AM	--	Folder
README.md	1/20/15, 4:47 PM	5 KB	Markdown
▼ decode	Today, 12:47 PM	--	Folder
▼ src	1/11/15, 10:59 AM	--	Folder
▶ tools	Today, 12:45 PM	--	Folder
▶ services	Today, 12:45 PM	--	Folder
▶ doge	Today, 12:45 PM	--	Folder
▶ pkg	Yesterday, 10:13 AM	--	Folder
▶ bin	Yesterday, 10:19 AM	--	Folder

The navigation bar at the bottom shows the path: Macintosh HD > Users > bryan > do > cthulhu > decode > src.

```
#!/bin/bash
SCRIPT=`python -c "import os,sys;
print(os.path.realpath(os.path.expanduser(s.argv[1])))" "${0}"`"
CTHULHU_DIR=$(dirname $SCRIPT)

DOCODE=$CTHULHU_DIR/docode
THIRDPARTY=$CTHULHU_DIR/third_party

export TEST_DB_USER=root
export TEST_DB_ADDR=127.0.0.1:3306
export GOPATH="$THIRDPARTY:$DOCODE"
export PATH="$DOCODE/bin:$THIRDPARTY/bin:$PATH"
```

```
2. _zsh_tmux_plugin_run new (tmux)

→ cthulhu git:(master) echo $GOPATH
/Users/bryan/do/cthulhu/third_party:/Users/bryan/do/cthulhu/docode
→ cthulhu git:(master) □
```

3 < ↑ 1d 23h 25m < 2.9 2.3 2.1 < 2015-02-07 < ⏱ 12:50 < 🔒 Macintosh.home

**Back to the
Standard Library ...**

It isn't all roses

```
input := "May 31, 2015 17:35"
format := "Jan 2, 2006 15:04"
t, err := time.Parse(format, input)
```

1/2 3:04:05 2006 -0700

1/2 3:04:05 2006 -0700

Month

1

01

Jan

January

1/2 3:04:05 2006 -0700

Day of month

**2
02**

1/2 3:04:05 2006 -0700

Hour

3

15

1/2 3:04:05 2006 -0700

Minute
04

1/2 3:04:05 2006 -0700

Seconds
05

1/2 3:04:05 2006 -0700

Year
06
2006
Jan
January

1/2 3:04:05 2006 -0700

Time Zone
-0700

if it hurts, don't do it

except for checking errors. always
check for errors.

<https://github.com/kisielk/errcheck>

Finally!

So You've Become A Hero!

[Score: 504 of 500]



**Go and download
Quest for Glory!**

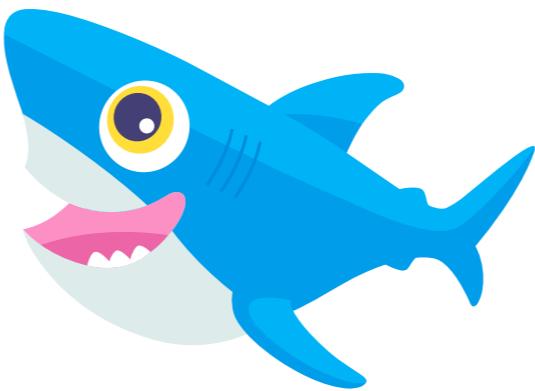
**Go and download
Quest for Glory!**

**Go and steal
Quest for Glory!**

**Go and buy
Quest for Glory!**

PS.

*Consider the standard library
as an adequate solution, not
necessarily a highly
performant one.*



The End.

