Best Practices

for Production Environments

These slides may not make much sense without a narrative!

For the proper article, see
http://peter.bourgon.org/go-in-production


SOUNDCLOUD

Some Go Projects

Bazooka

Traffic tier

Media streaming

Search & Explore glue

Prometheus

Stream backend*

HLS implementation**

Many others…

Dev environment
Repo structure
Formatting and style
Configuration
Logging and telemetry
Validation and testing
Dependency management ᕕ(ᐛ)ᕗ
Build and deploy

Dev environment

Single `GOPATH` for all projects

Work in `$GOPATH/src/github.com/soundcloud/foo`

vim, Sublime Text, emacs—no IDEs

Repo structure

```
github.com/soundcloud/whatever/
    README.md
    Makefile
    main.go
    support.go
```

```
github.com/soundcloud/whatever/
    README.md
    Makefile
    main.go
    support.go
    foo/
        foo.go
        bar.go
```

```
github.com/soundcloud/whatever/
    README.md
    Makefile
    whatever-server/
        main.go
    whatever-worker/
        main.go
    foo/
        foo.go
        bar.go
```

Formatting and style

`go fmt`

Google's Code Review Guidelines

Avoid named return parameters

Avoid `make` and `new` (unless you know sizes)

Use `struct{}` for sentinel value: sets, signal chans

Break long lines on parameters

```
func process(dst io.Writer, readTimeout,
    writeTimeout time.Duration, allowInvalid bool,
        max int, src <-chan util.Job) {
    // ...
}
```

```
func process(
    dst io.Writer,
    readTimeout, writeTimeout time.Duration,
    allowInvalid bool,
    max int,
    src <-chan util.Job,
) {
    // ...
}
```

```
f := foo.New(foo.Config{
    “zombo.com”,
    conference.KeyPair{
        Key:   “gophercon”, // String value
        Value: 2014,
    },
    os.Stdout,
})
```

Configuration

package flag

```
func main() {
    var (
        foo = flag.String(“foo”, “doch”, “...”)
        bar = flag.Int(“bar”, 34, “...”)
    )
    flag.Parse()
    // ...
}
```

Logging and telemetry

Logging and telemetry

package log

Logging and telemetry

Push model

Graphite
Statsd
AirBrake

vs.

Pull model

expvar
Prometheus
others?

Testing and validation

Testing and validation

package testing

☞`reflect.DeepEqual`☜

```
// +build integration

var fooAddr = flag.String(...)

func TestFoo(t *testing.T) {
    f, err := newFoo(*fooAddr)
    // ...
}
```

Testing and validation

gocov?

on Save

`go fmt` *or*
`goimports`

on Build

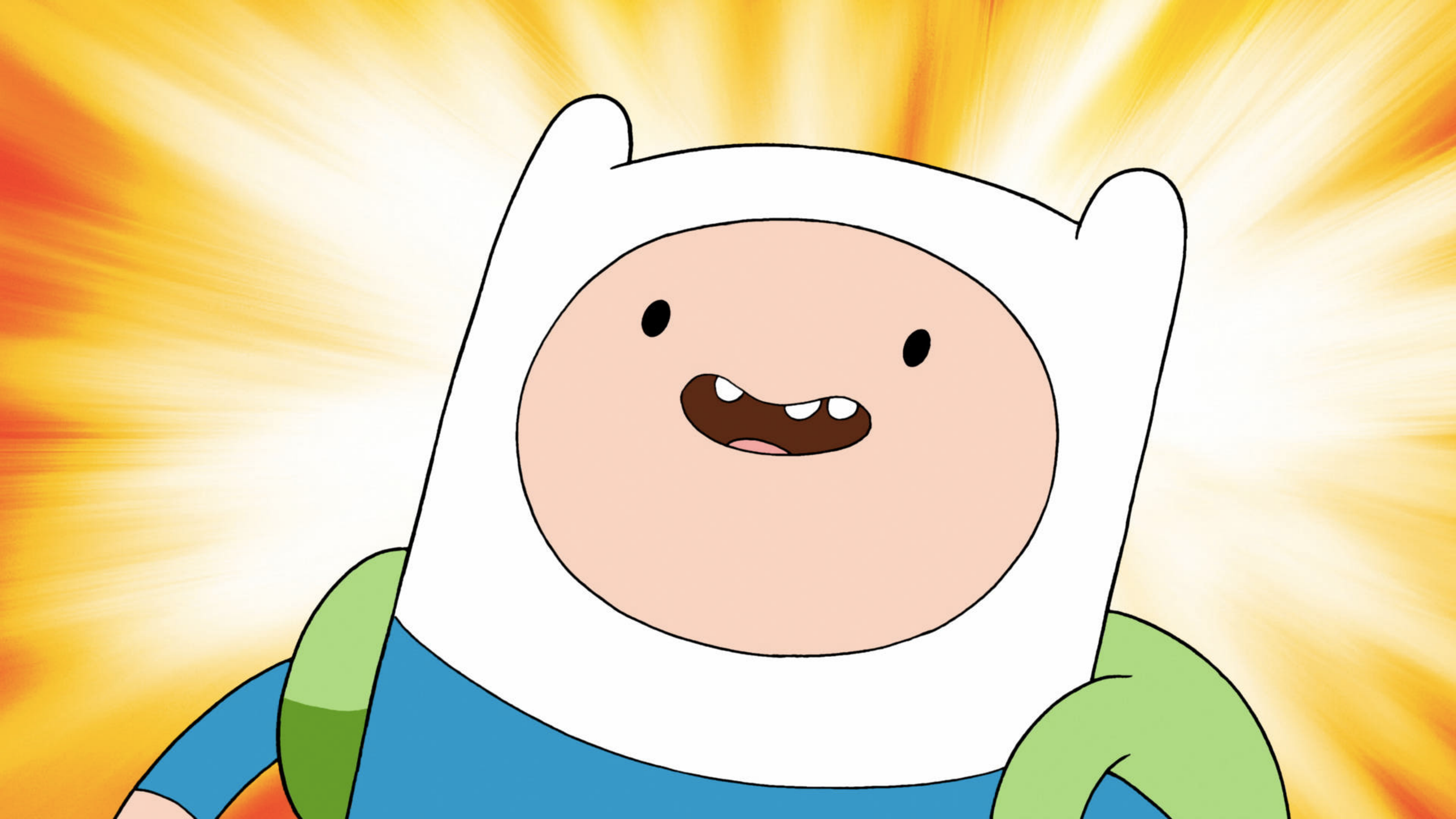
`go vet`
and `golint`
and maybe `go test`

on Deploy

```
go test
 -tags=integration
```

YAGNI

Dependency management



How important is your project?

Eh ☞ `go get -d` *& hope!*

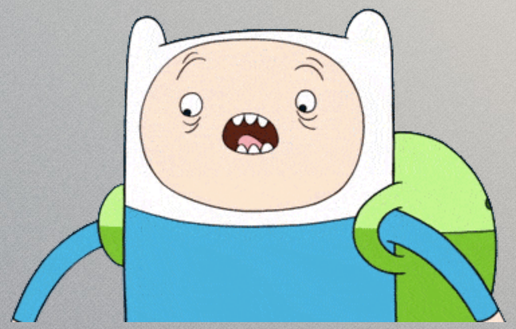
import proxy?

Very ☞ **VENDOR**

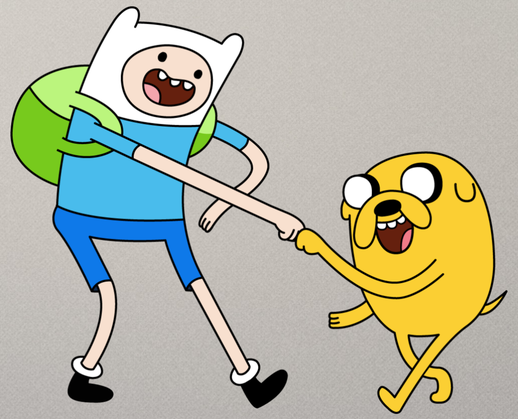
VENDOR

means copy your dependencies

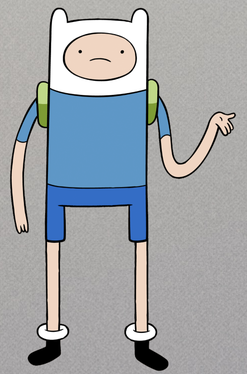
~~with submodules~~



with **subtrees**



with **a tool**



VENDOR

Binary ☞ **`_vendor`** subdirectory + Blessed build, with prefixed **`GOPATH`**

Library ☞ **`vendor`** subdirectory + Rewrite your imports

Build and deploy

Build and deploy

Development 👉 `go build`

“Official” 👉 `make`

```
GOVER=go1.2.1

STAGE=.stage
GOPATH=$(CURDIR)/$(STAGE)/gopath
GOROOT=$(CURDIR)/$(STAGE)/go
GOCC=$(GOROOT)/bin/go
GO=TMPDIR=/tmp GOROOT=$(GOROOT) GOPATH=$(GOPATH) $(GOCC)

OS=$(shell uname)
ARCH=$(shell uname -m)
GOOS=$(subst Darwin,darwin,$(subst Linux,linux,$(OS)))
GOARCH=$(subst x86_64,amd64,$(ARCH))
GOPKG=$(subst darwin-amd64,darwin-amd64-osx10.8,$(GOVER).$(GOOS)-$(GOARCH).tar.gz)

PKGBASE=github.com/soundcloud/goku
PKGPATH=$(GOPATH)/src/$(PKGBASE)

all: build

build: $(GOCC) $(PKGPATH)
    GOPATH=$(GOPATH) GOROOT=$(GOROOT) GO=$(GOCC) make -C roshi-server build
    GOPATH=$(GOPATH) GOROOT=$(GOROOT) GO=$(GOCC) make -C roshi-walker build
    GOPATH=$(GOPATH) GOROOT=$(GOROOT) GO=$(GOCC) make -C reader build
    GOPATH=$(GOPATH) GOROOT=$(GOROOT) GO=$(GOCC) make -C writer build
    GOPATH=$(GOPATH) GOROOT=$(GOROOT) GO=$(GOCC) make -C gap-backfill build

clean:
    GOPATH=$(GOPATH) GOROOT=$(GOROOT) GO=$(GOCC) make -C roshi-server clean
    GOPATH=$(GOPATH) GOROOT=$(GOROOT) GO=$(GOCC) make -C roshi-walker clean
```

Build and deploy

Stateless	vs.	Stateful
Request router		MySQL, Redis
12-Factor model		Any/no model
Scaled		Provisioned
Containers		Containers?

Deploying stateless services

```
$ git push bazooka master
$ bazooka scale -r <new> -n 4 ...
$ # validate
$ bazooka scale -r <old> -n 0 ...
```

Embrace simplicity

Thanks! Questions?

Peter Bourgon
@peterbourgon