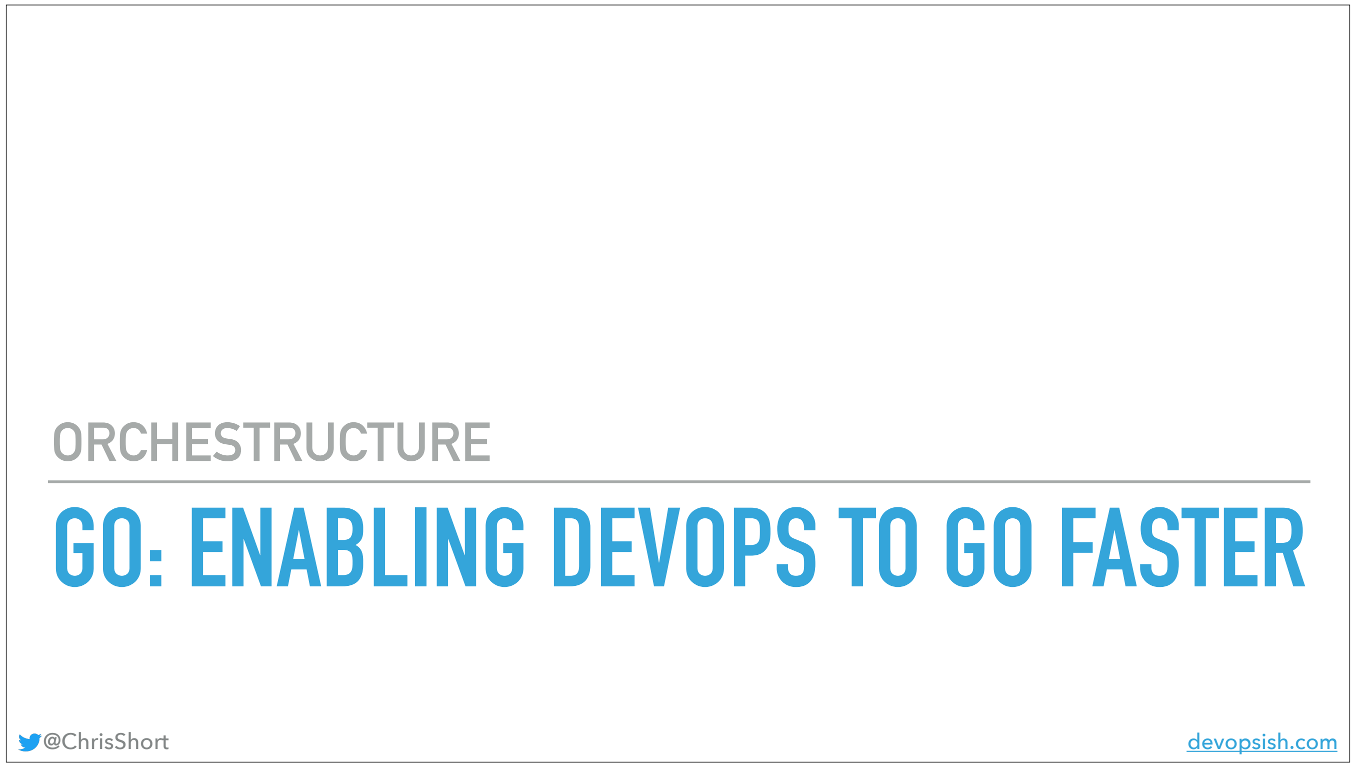


ORCHESTRUCTURE

GO: ENABLING DEVOPS TO GO FASTER

@ChrisShort

devopsish.com

- Hello!

LICENSE AND MATERIALS

- Gopher Artwork from Ashley McNamara: https://github.com/ashleymcnamara/gophers
- All product names, logos, and brands are property of their respective owners. All company, product and service names used in this work are for identification purposes only. Use of these names, logos, and brands does not imply endorsement.
- This presentation is licensed under the Creative Commons Attribution-ShareAlike 4.0 International license.
- You are encouraged to remix, transform, or build upon the material, providing you distribute your contributions under the same license.
- This presentation will be available on chrisshort.net on or after 31 Jan 2018.

INTRODUCTION

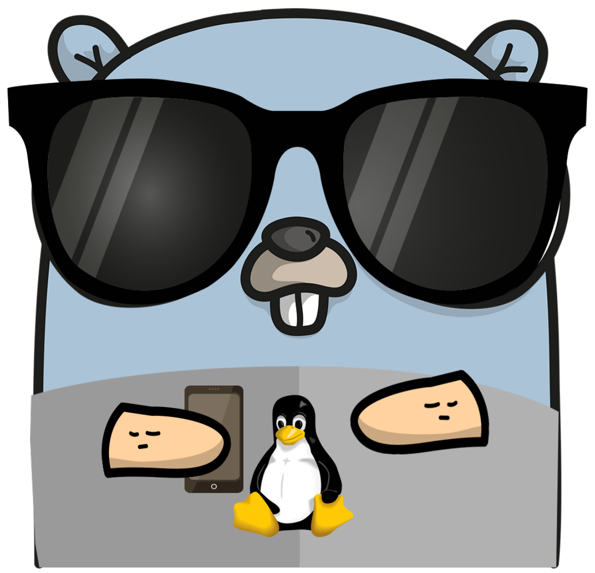
chrisshort.net
opensource.com
DETROIT GOPHERS
CLOUD NATIVE COMPUTING FOUNDATION AMBASSADOR
DevOps'ish
THE OPEN ORGANIZATION GUIDE TO IT CULTURE CHANGE

@ChrisShort

devopsish.com

- Hi I'm Chris Short
- I'm a Senior DevOps Advocate at SJ Technologies
- I write a weekly newsletter called DevOps'ish
- I was recently named a Cloud Native Computing Foundation Ambassador
- I'm co-lead of the DevOps community at opensource.com
- I'm co-lead of the Detroit Go Meetup
- And I have a few other things I do in my non-existent free time

I'M ALSO A GOPHER



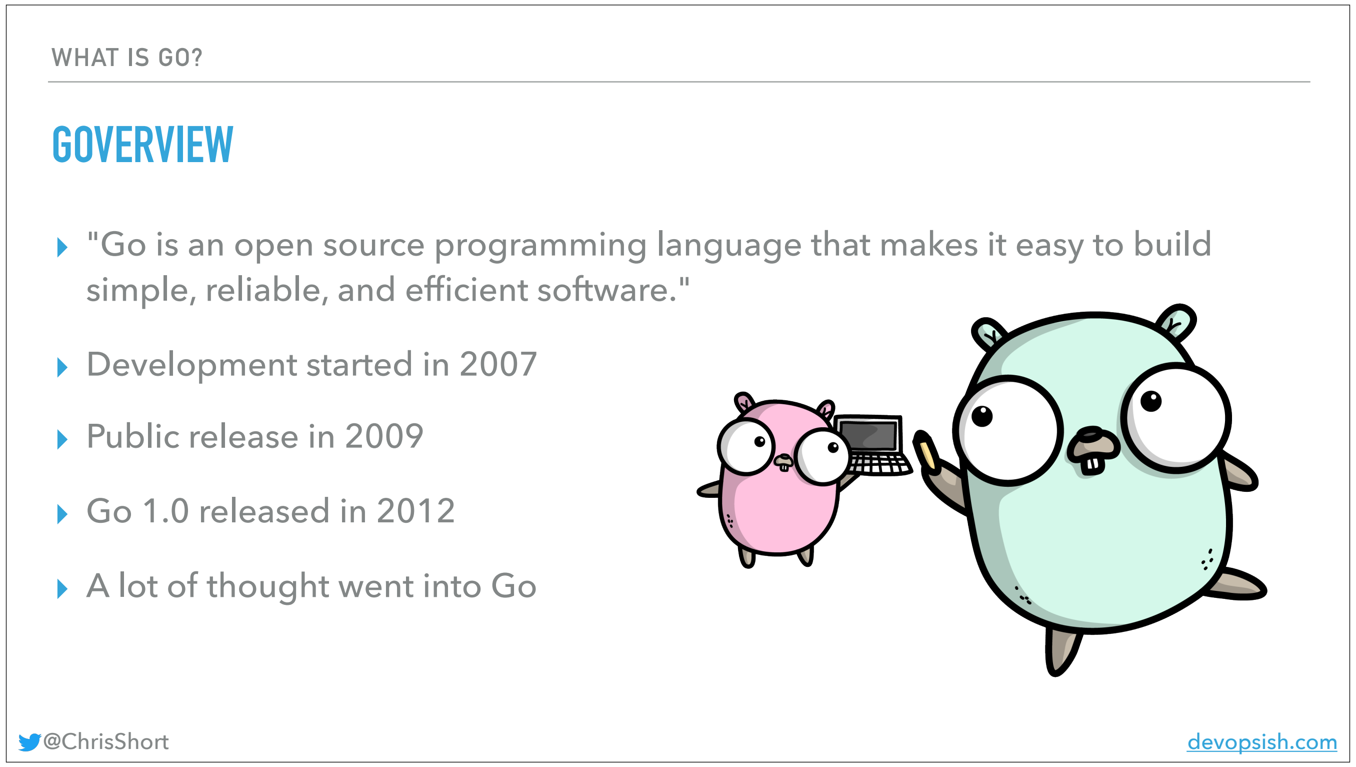
Chris Short in Gopher Form by Gopherize.me

@ChrisShort

devopsish.com

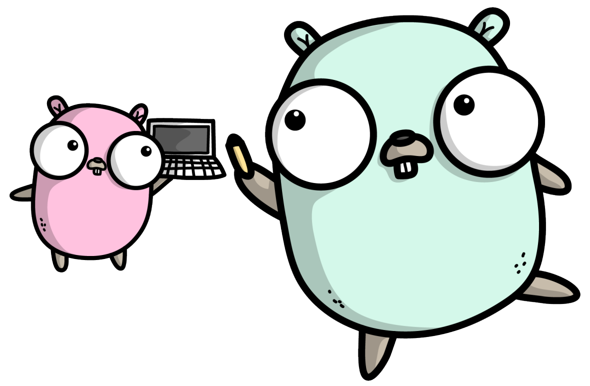
- I'm many things
- One thing I'm not is a coder
- But, Go is starting to change that

WHAT IS GO?

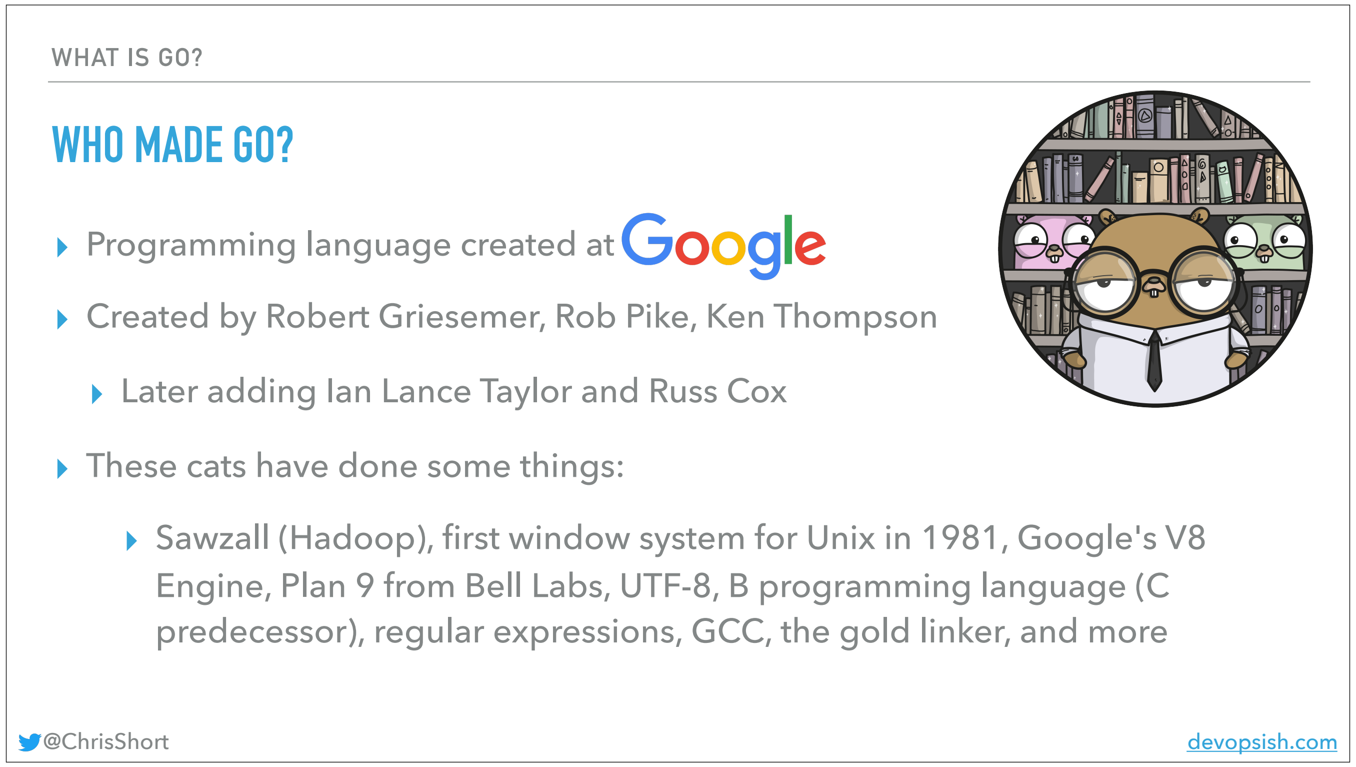


GOVERVIEW

- "Go is an open source programming language that makes it easy to build simple, reliable, and efficient software."
- Development started in 2007
- Public release in 2009
- Go 1.0 released in 2012
- A lot of thought went into Go

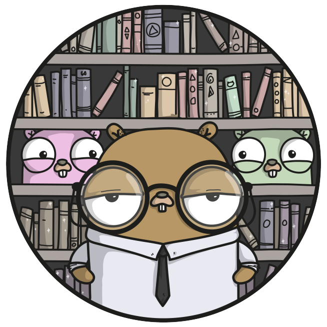


- "Go is an open source programming language that makes it easy to build simple, reliable, and efficient software."
- Development started in 2007
- Publicly released in 2009
- 1.0 in 2012
- Go development is very thoughtful and methodical



WHO MADE GO?

- Programming language created at Google
- Created by Robert Griesemer, Rob Pike, Ken Thompson
  - Later adding Ian Lance Taylor and Russ Cox
- These cats have done some things:
  - Sawzall (Hadoop), first window system for Unix in 1981, Google's V8 Engine, Plan 9 from Bell Labs, UTF-8, B programming language (C predecessor), regular expressions, GCC, the gold linker, and more

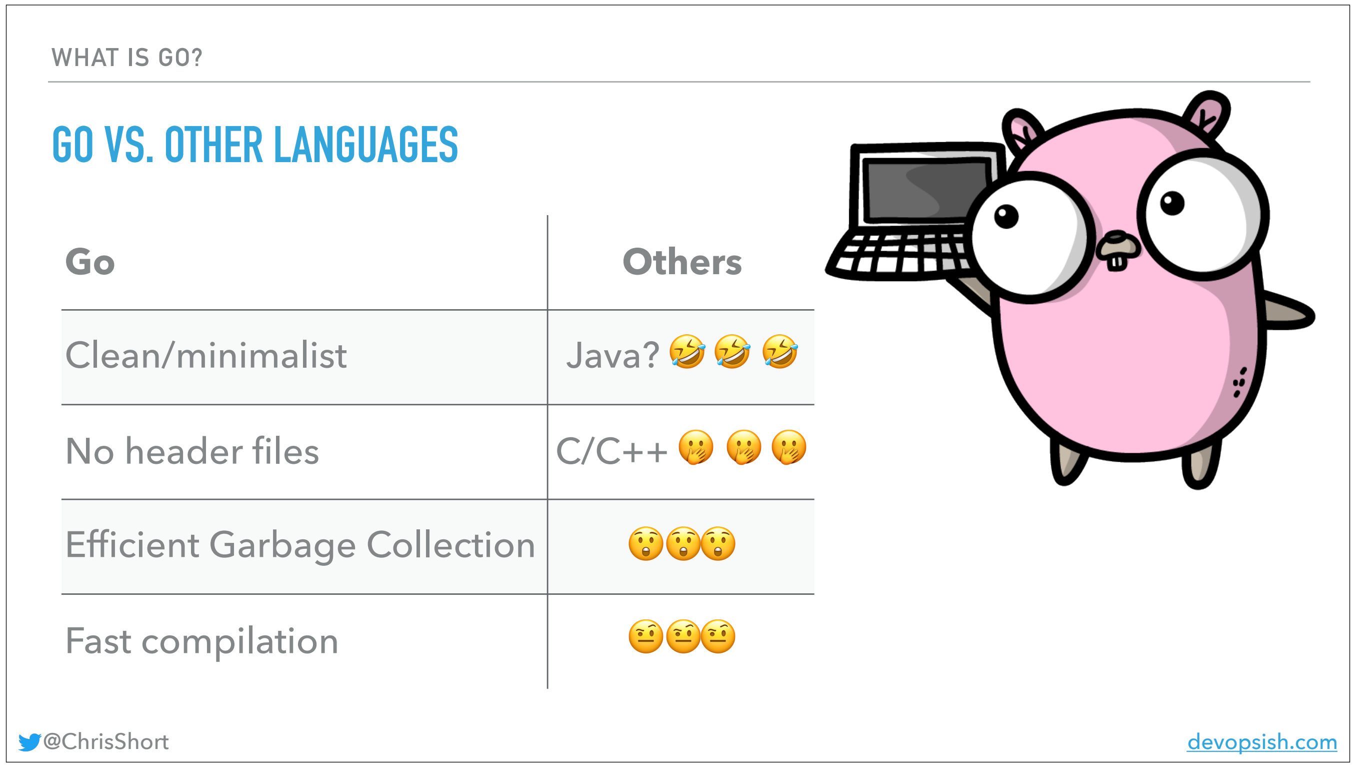


- Created at Google
- A bunch of really smart people made Go
- Robert Griesemer
- Rob Pike
- Ken Thompson
- Ian Lance Taylor
- Russ Cox
- Some seriously brilliant minds

WHY MAKE GO?

- "No new major systems language in a decade." –Rob Pike
- Designed with the following advances in technology in mind:
  - Modern Networking
  - Multi-core CPUs
  - Slowing of Moore's Law
- Improved safety, high speed compilation, and communications

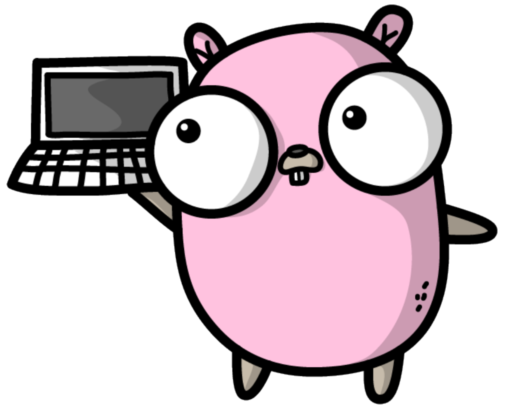
- Think of the language landscape in 2009
- What was the newest language? Python? PHP?
- These languages didn't handle modern computing concepts well



WHAT IS GO?

GO VS. OTHER LANGUAGES

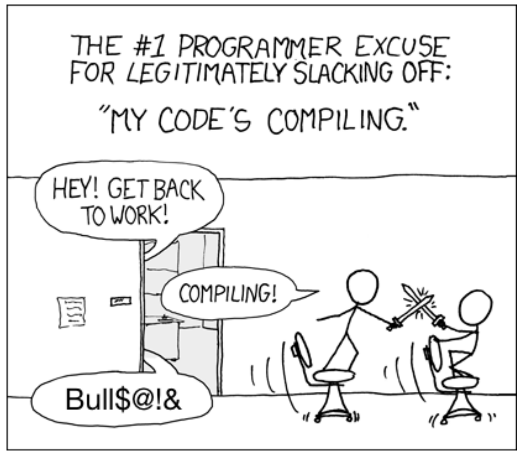
Go	Others
Clean/minimalist	Java? 🤣 🤣 🤣
No header files	C/C++ 🤭 🤭 🤭
Efficient Garbage Collection	😲😲😲
Fast compilation	🤨🤨🤨



@ChrisShort devopsish.com

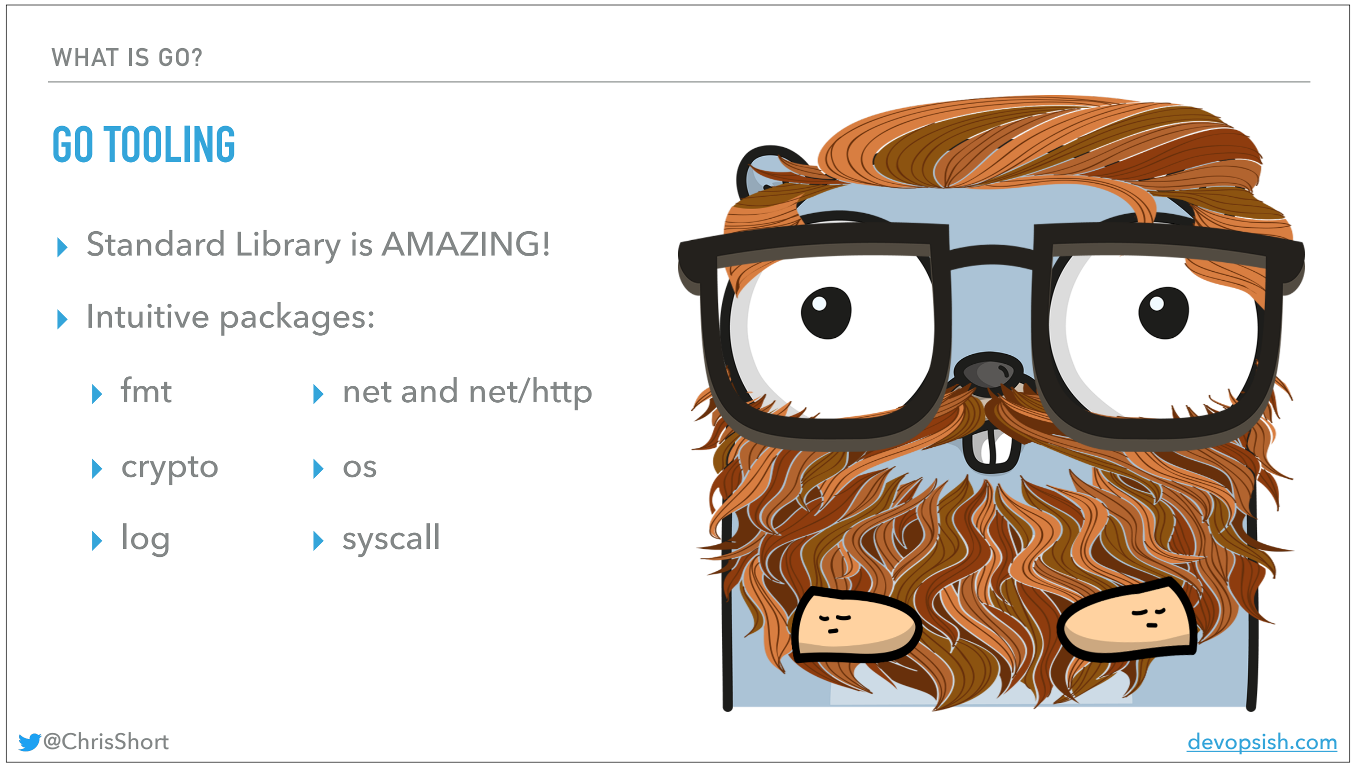
- Go is clean and minimalist
- Makes it extremely easy to learn and use
- It's so easy this Ops turned DevOps clown can talk to you about it
- There aren't a mess of header files like C
- Garbage collection is quick and efficient
- Concurrency enables multiple functions to run simultaneously
- Go programs compile very quickly

SORRY, DEVELOPERS



https://xkcd.com/303/

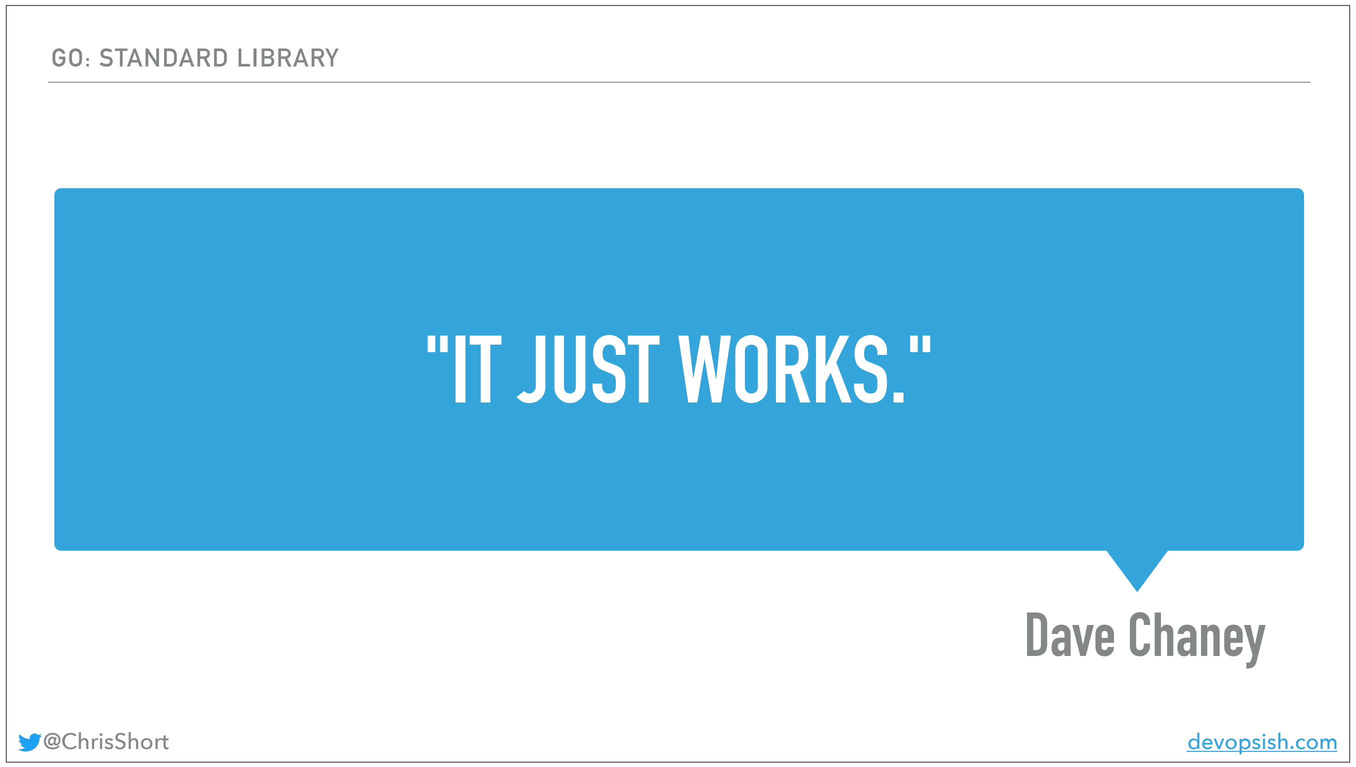
- I fixed this comic for you
- Sorry, devs



GO TOOLING

- Standard Library is AMAZING!
- Intuitive packages:
  - fmt
  - crypto
  - log
  - net and net/http
  - os
  - syscall

- The standard library is really, really good.
- The packages are sane and intuitive
- fmt will literally format your code for you
- crypto is right in line with the RFC
- log allows you to exit the program with an exit code
- net package is very well written
- syscall and os just work


GO: STANDARD LIBRARY

"IT JUST WORKS."

Dave Chaney

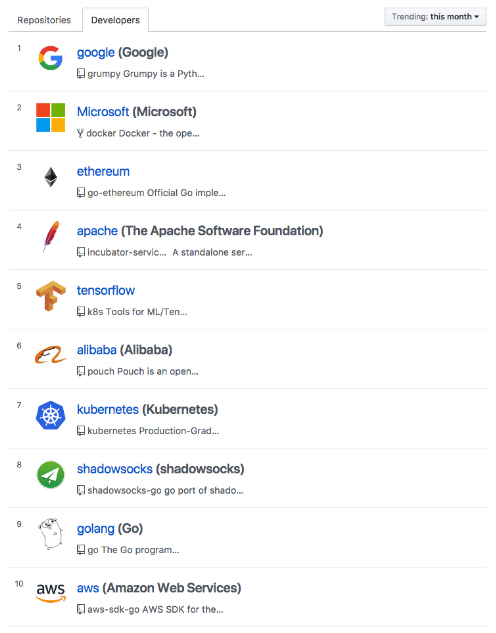
@ChrisShort

devopsish.com

- According to Dave Chaney, "It just works."

WHO CONTROLS GO?

- It's open source! The community!
- Go was developed at Google by Google Folks
- But, look who is writing Go code
  - #2: Microsoft
  - #4: Apache
  - #6 Alibaba



1 google (Google)
grumpy Grumpy is a Pyth...

2 Microsoft (Microsoft)
docker Docker - the ope...

3 ethereum
go-ethereum Official Go imple...

4 apache (The Apache Software Foundation)
incubator-servic... A standalone ser...

5 tensorflow
k8s Tools for ML/Ten...

6 alibaba (Alibaba)
pouch Pouch is an open...

7 kubernetes (Kubernetes)
kubernetes Production-Grad...

8 shadowsocks (shadowsocks)
shadowsocks-go go port of shado...

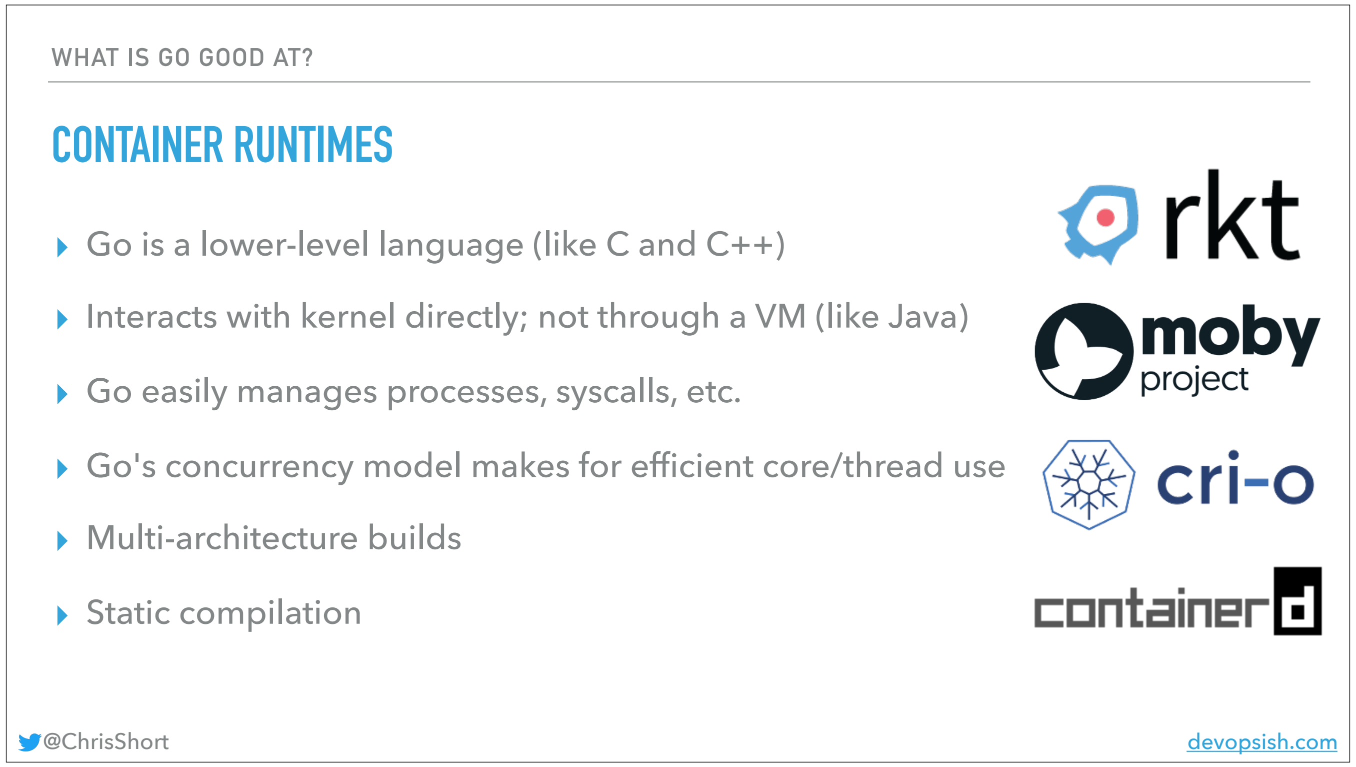
9 golang (Go)
go The Go program...

10 aws (Amazon Web Services)
aws-sdk-go AWS SDK for the...

- Go was created at Google
- I know what you're thinking
- "I don't have Google's problems"
- You don't but you sure can benefit from their work
- Microsoft, Apache, and Alibaba are!

WHAT IS GO GOOD AT?

- There are some obvious projects that use Go
- Let's not talk about the obvious stuff
- That's boring



WHAT IS GO GOOD AT?

CONTAINER RUNTIMES

- Go is a lower-level language (like C and C++)
- Interacts with kernel directly; not through a VM (like Java)
- Go easily manages processes, syscalls, etc.
- Go's concurrency model makes for efficient core/thread use
- Multi-architecture builds
- Static compilation

rkt

moby project

cri-o

containerd

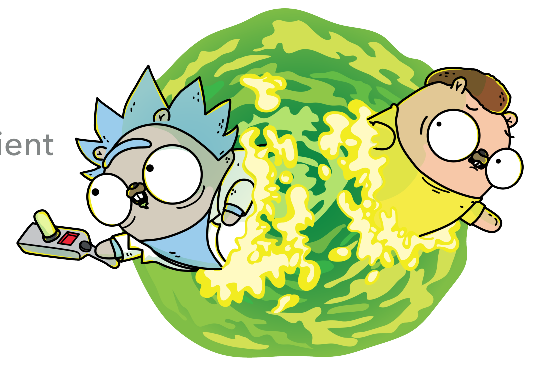
@ChrisShort

devopsish.com

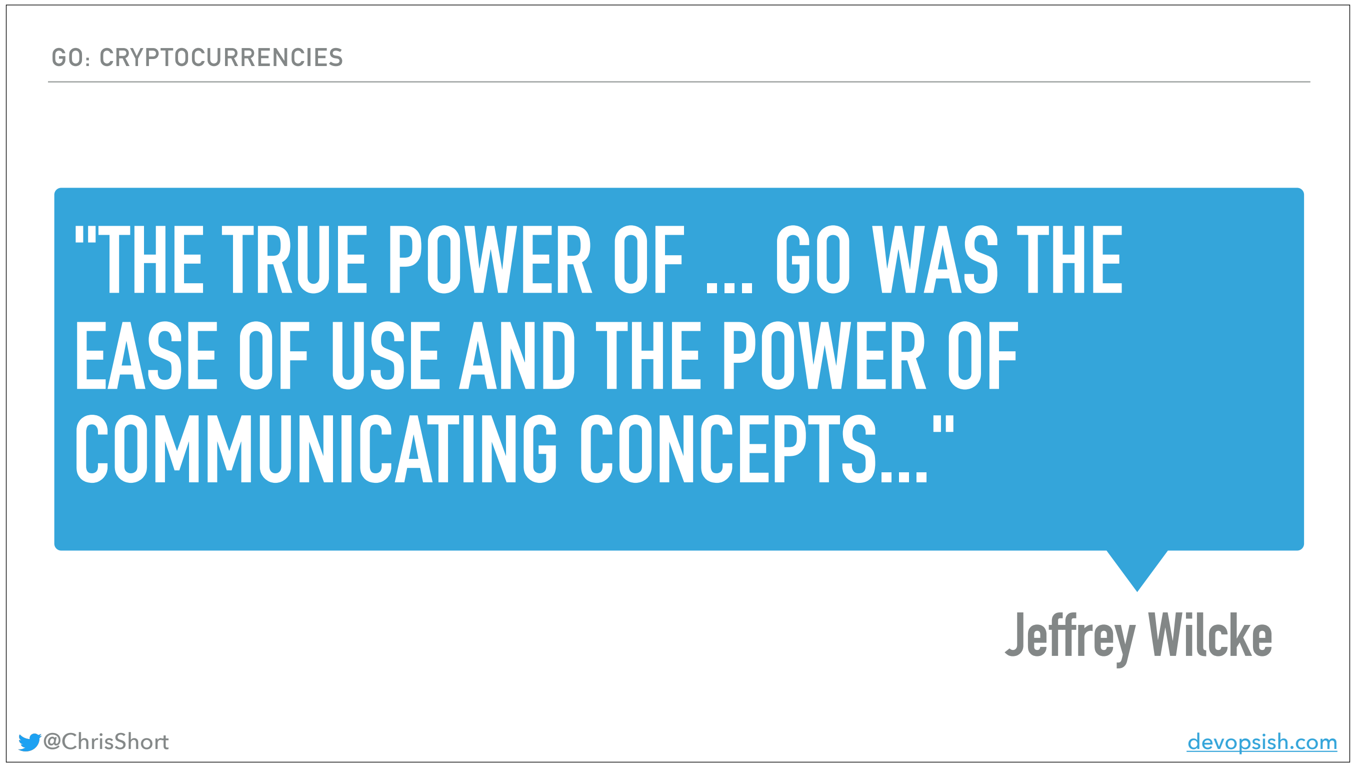
- Container runtimes are generally written in go these days
- Go is a lower-level language
- Go doesn't use a virtual machine
- Go's process handling and straightforward syscall interactions make for clean interactions with the system
- Look up Liz Rice's GopherCon 2017 talk
- She walks through building strace in 60 lines of Go
- Concurrency makes for great resource utilization
- You can compile for x86 as easily as you can arm
- Your output is a static binary
- Everything in one artifact ready for deployment

CRYPTOCURRENCIES

- **Ethereum** has the #3 GitHub project for Go
- geth is the Go implementation of Ethereum client
- geth is the *default* Ethereum client
- geth became the "reference client"



- geth is the go implementation of the Ethereum client
- It's also the default Ethereum client
- geth became the reference client for all other clients written in other languages



GO: CRYPTOCURRENCIES

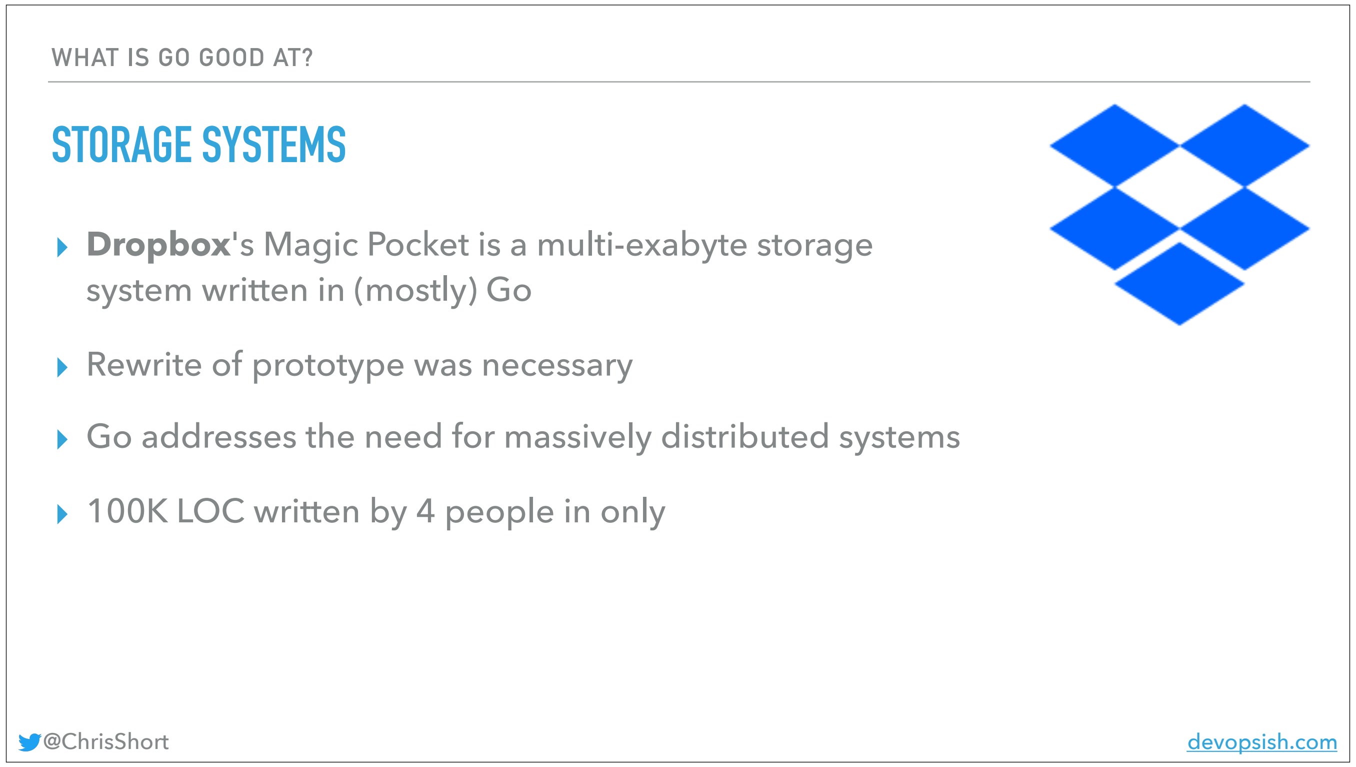
"THE TRUE POWER OF ... GO WAS THE EASE OF USE AND THE POWER OF COMMUNICATING CONCEPTS..."

Jeffrey Wilcke

@ChrisShort

devopsish.com

- Just read this quote.
- That's so amazing.
- It was easier to understand the Ethereum concepts laid out in the white paper in Go than any other language



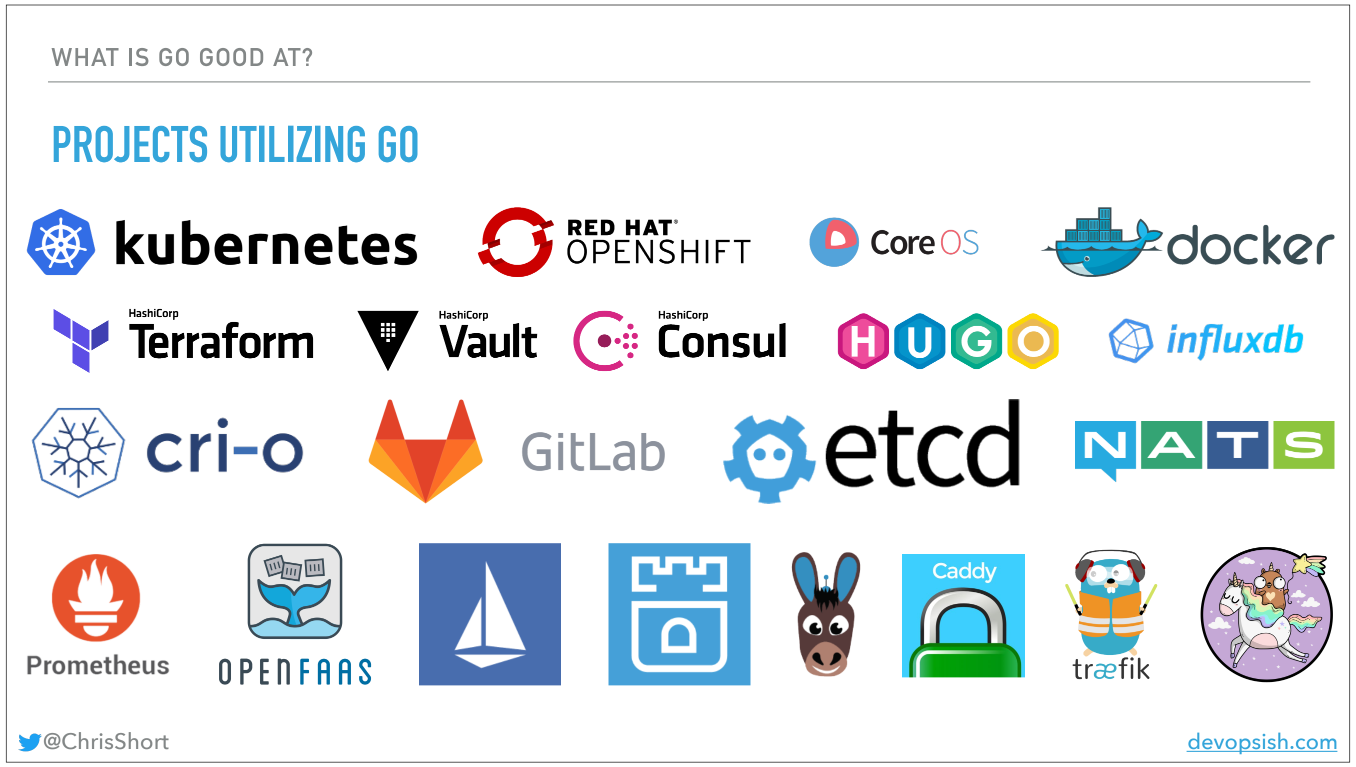
WHAT IS GO GOOD AT?

STORAGE SYSTEMS

- **Dropbox**'s Magic Pocket is a multi-exabyte storage system written in (mostly) Go
- Rewrite of prototype was necessary
- Go addresses the need for massively distributed systems
- 100K LOC written by 4 people in only

@ChrisShort devopsish.com

- When Dropbox pulled out of AWS they landed on Go
- Prototype that was written in Python
- After testing that wasn't going to scale
- Plus they wanted Go for it's type-safe and concurrency
- 4 people re-wrote the prototype in Go
- 100K LOC of Go code were hammered out in six months
- They did end up writing some bits in Rust too


WHAT IS GO GOOD AT?

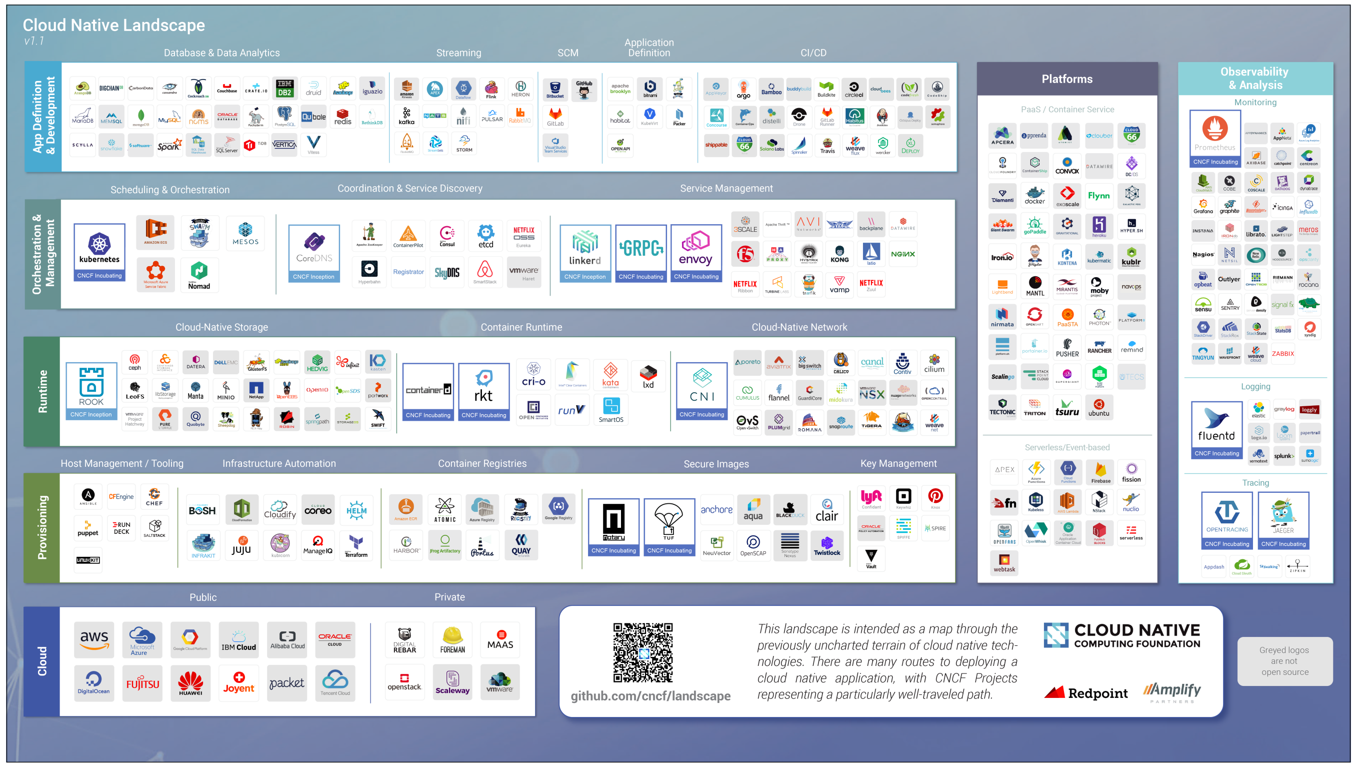
PROJECTS UTILIZING GO

kubernetes
RED HAT OPENSHIFT
CoreOS
docker
HashiCorp Terraform
HashiCorp Vault
HashiCorp Consul
HUGO
influxdb
cri-o
GitLab
etcd
NATS
Prometheus
OPENFAAS
Caddy
træfik

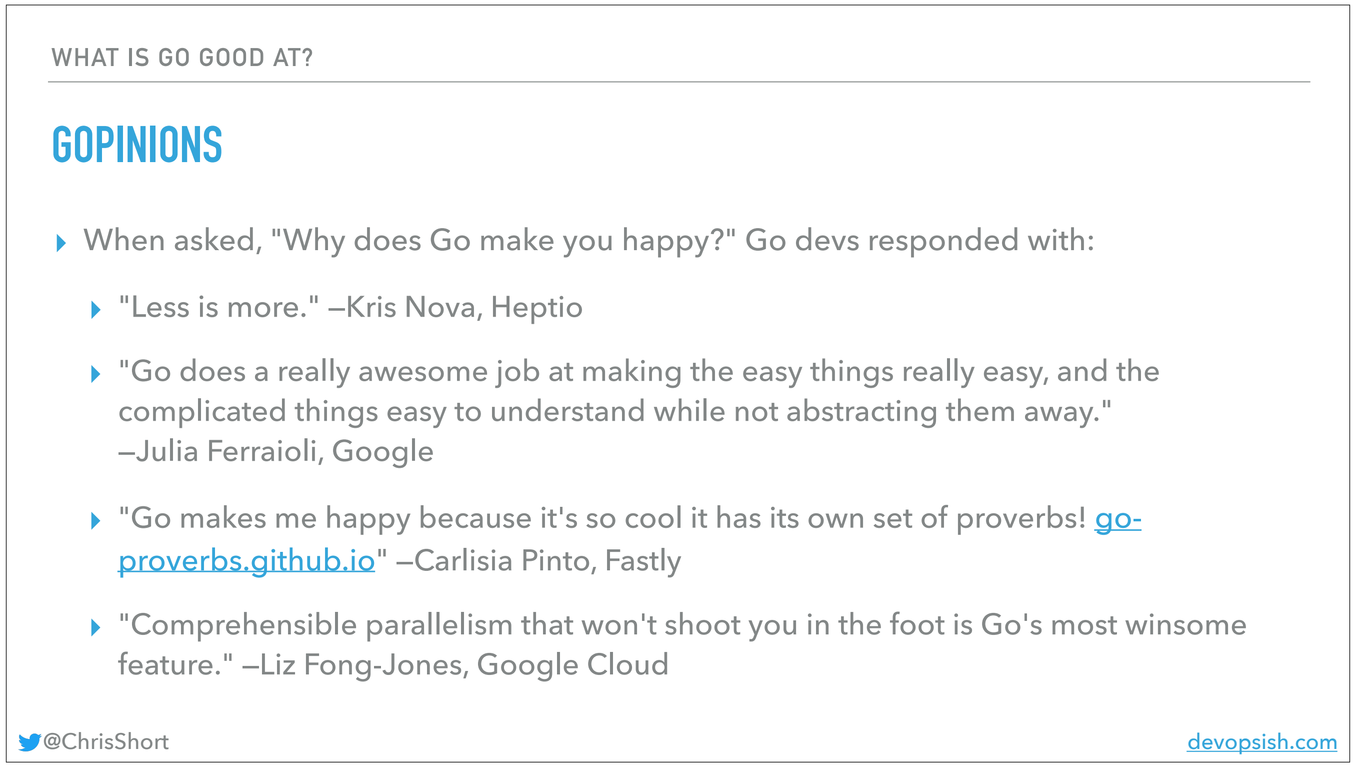
@ChrisShort

devopsish.com

- This is just a snapshot of projects using Go extensively today


Cloud Native Landscape
v1.1
App Definition & Development
Orchestration & Management
Runtime
Provisioning
Cloud
Platforms
Observability & Analysis
github.com/cncf/landscape
This landscape is intended as a map through the previously uncharted terrain of cloud native technologies. There are many routes to deploying a cloud native application, with CNCF Projects representing a particularly well-traveled path.
CLOUD NATIVE COMPUTING FOUNDATION
Redpoint
Amplify
Greyed logos are not open source

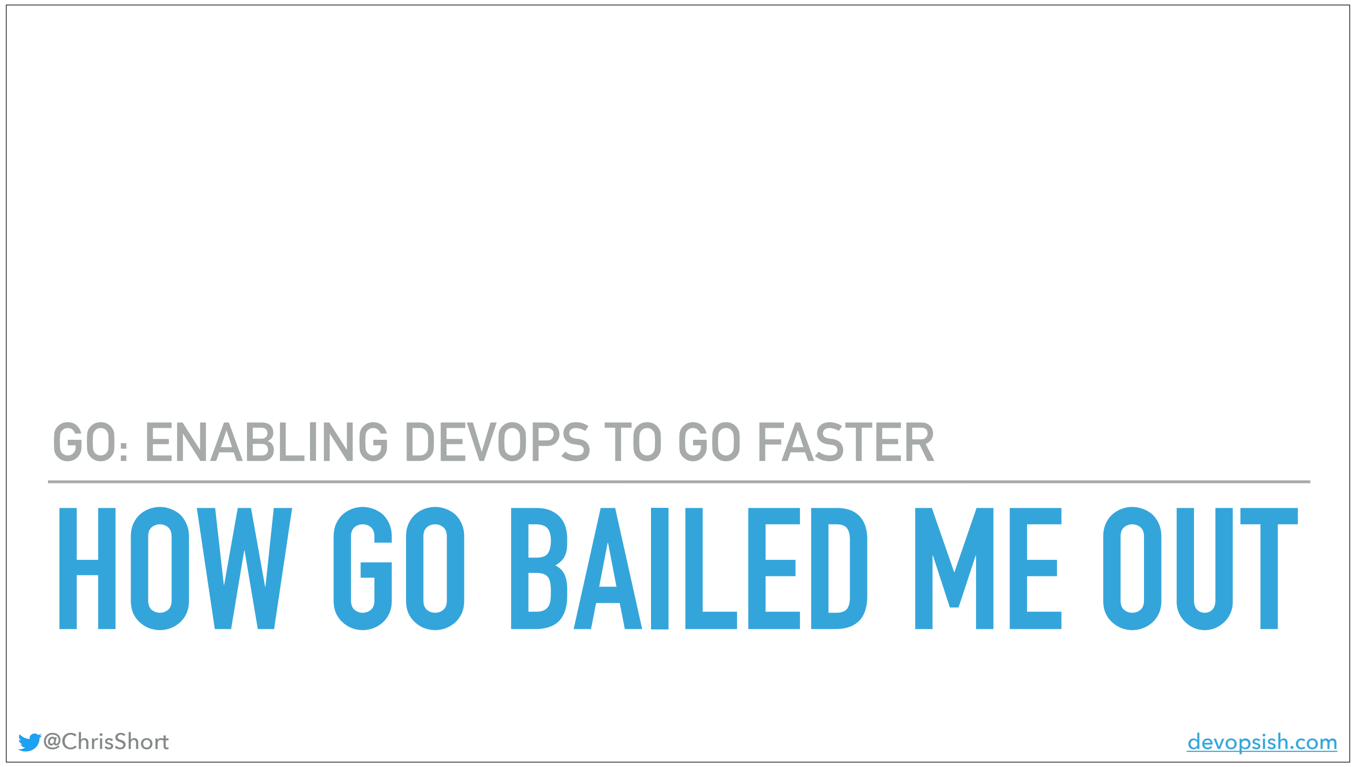
- For those that haven't seen this before this is the Cloud Native Computing Foundation Landscape
- This is a collection of tools that handle Cloud Native concepts very well
- Quite a bit of the CNCF Landscape tooling is written in Go
- Not all of it is though



GOPINIONS

- When asked, "Why does Go make you happy?" Go devs responded with:
  - "Less is more." –Kris Nova, Heptio
  - "Go does a really awesome job at making the easy things really easy, and the complicated things easy to understand while not abstracting them away." –Julia Ferraioli, Google
  - "Go makes me happy because it's so cool it has its own set of proverbs! go-proverbs.github.io" –Carlisia Pinto, Fastly
  - "Comprehensible parallelism that won't shoot you in the foot is Go's most winsome feature." –Liz Fong-Jones, Google Cloud

- I asked some friends why Go makes them happy



GO: ENABLING DEVOPS TO GO FASTER

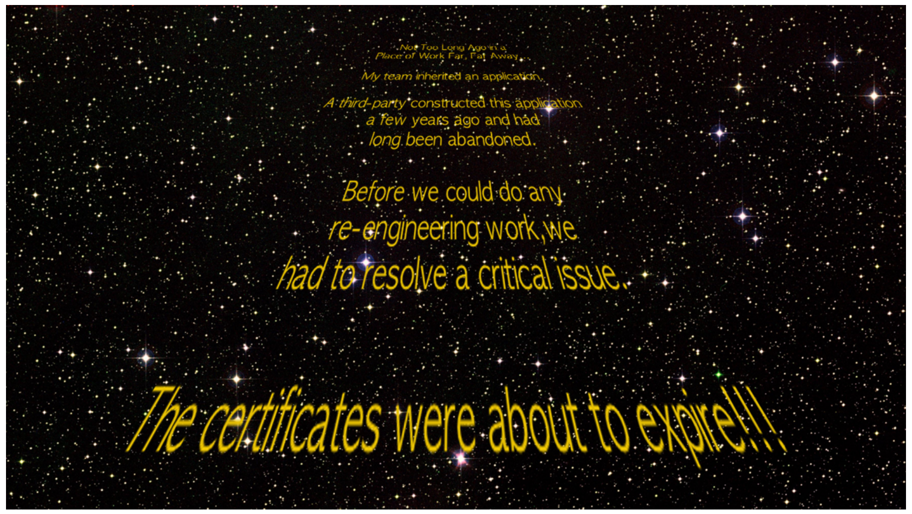
HOW GO BAILED ME OUT

@ChrisShort

devopsish.com

Let me tell you a story of how Go saved my ass once

NOT TOO LONG AGO IN A PLACE OF WORK FAR, FAR AWAY…


Not Too Long Ago in a
Place of Work Far, Far Away…

My team inherited an application.

A third-party constructed this application
a few years ago and had
long been abandoned.

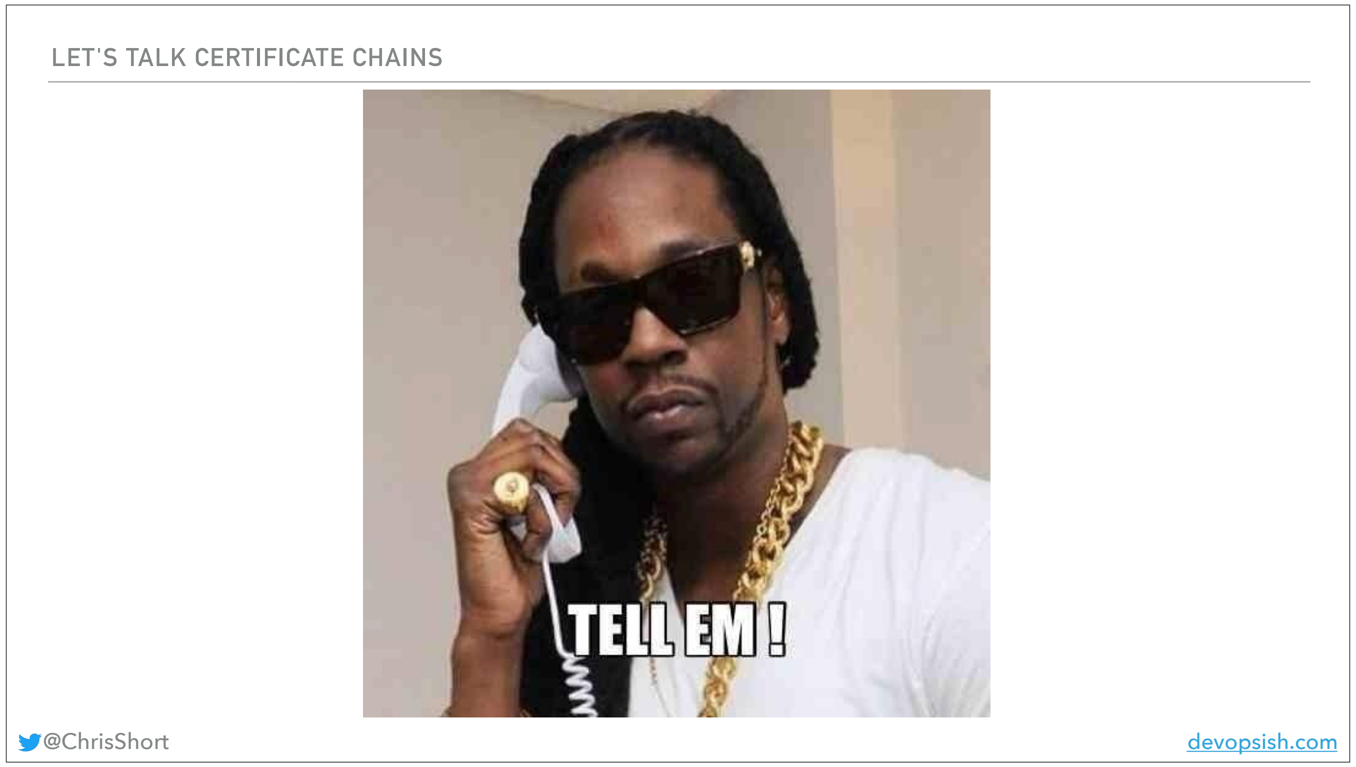
Before we could do any
re-engineering work,we
had to resolve a critical issue.

The certificates were about to expire!!!

- My team of merry DevOps'ers inherited an application
- A third-party built the app a few years ago
- The app had long been abandoned
- Before we could do any re-engineering work, we had to resolve a critical issue.
- The certificates were about to expire!

AND OF COURSE PRODUCTION

- Oh!
- And the only environment this application was in was production
- And there was no time to implement a new key management system
- And it was a pet project of someone in senior leadership

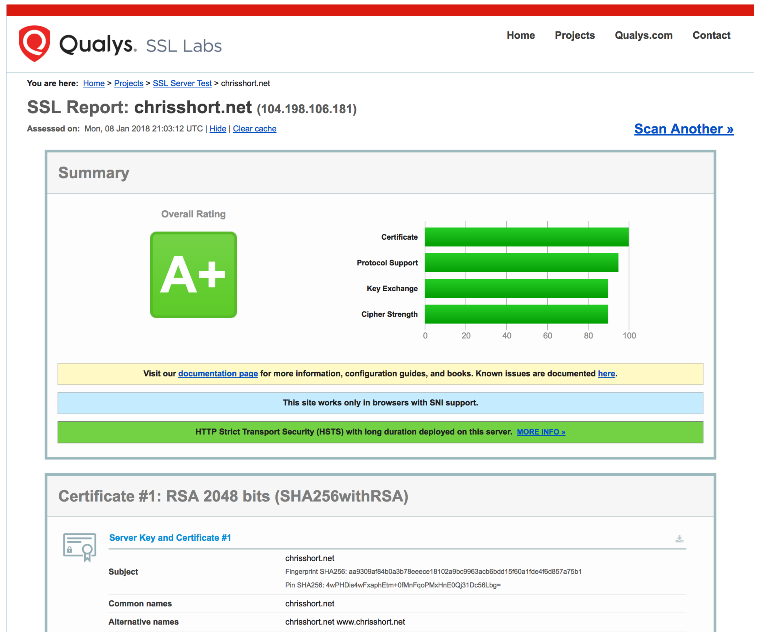


LET'S TALK CERTIFICATE CHAINS

@ChrisShort devopsish.com

- Let's Talk Certificate Chains for a minute
- HTTPS, SSL... it's all TLS now!
- In my opinion, if you're using TLS you MUST have a rock solid configuration
- This means you have to include the certificate chain in the correct order
- This is no longer optional in the post-Heartbleed world
- The Internets are watching and if they find something wrong with your setup they'll exploit it

THIS IS THE GOAL



Home Projects Qualys.com Contact

You are here: Home > Projects > SSL Server Test > chrisshort.net

SSL Report: chrisshort.net (104.198.106.181)

Assessed on: Mon, 08 Jan 2018 21:03:12 UTC | Hide | Clear cache

Scan Another »

Summary

Overall Rating

A+

Visit our documentation page for more information, configuration guides, and books. Known issues are documented here.

This site works only in browsers with SNI support.

HTTP Strict Transport Security (HSTS) with long duration deployed on this server. MORE INFO »

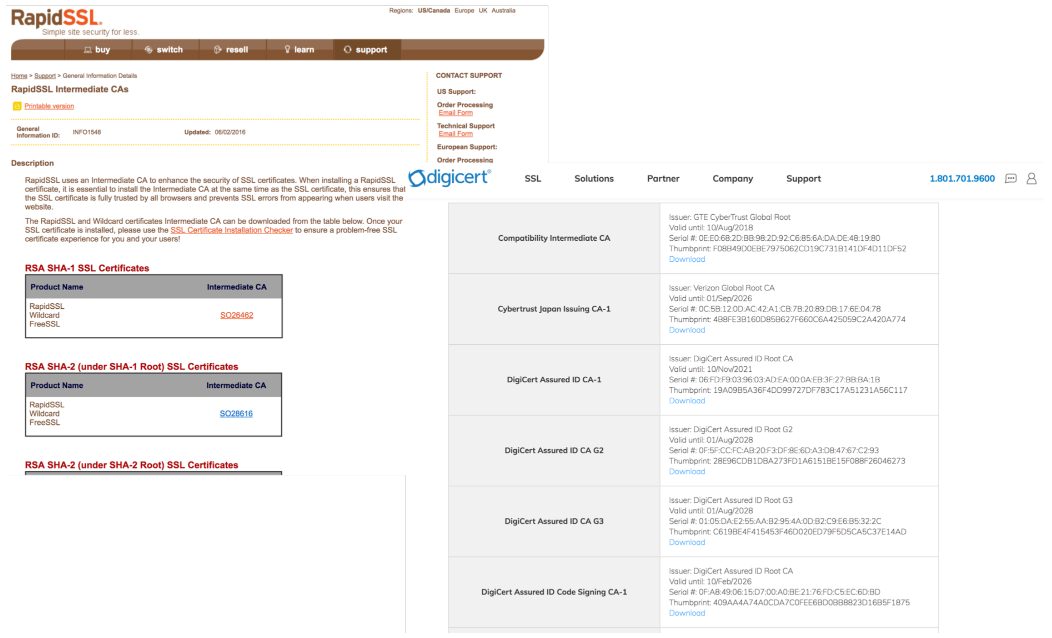
Certificate #1: RSA 2048 bits (SHA256withRSA)

Server Key and Certificate #1

Subject	chrisshort.net Fingerprint SHA256: aa9309af84b0a3b78eeece18102a9bc9963acb6bdd15f60a1fde4f6d857a75b1 Pin SHA256: 4wPHDis4wFxaphEtm+0fMnFqoPMxHnE0Qj31Dc56Lbg=
Common names	chrisshort.net
Alternative names	chrisshort.net www.chrisshort.net

- This is the goal
- If you are going to bother to encrypt your traffic you better do it right
- This is what we're aiming for; an A+
- At this company, we obtained certs from a preferred vendor that our company was cool with
- I prefer Let's Encrypt but some companies aren't comfortable with that yet for various reasons
- The process goes like this...
- You generate your CSR and private key
- You send the CSR to the vendor
- The certificate arrives but usually doesn't have an intermediate key in chain because... vendors be vendoring

💁...😲



Home > Support > General Information Details

RapidSSL Intermediate CAs

Printable version

General Information ID: INFO1548 Updated: 06/02/2016

CONTACT SUPPORT

US Support:

Order Processing
Email Form

Technical Support
Email Form

European Support:

Order Processing

Description

RapidSSL uses an Intermediate CA to enhance the security of SSL certificates. When installing a RapidSSL certificate, it is essential to install the Intermediate CA at the same time as the SSL certificate, this ensures that the SSL certificate is fully trusted by all browsers and prevents SSL errors from appearing when users visit the website.

The RapidSSL and Wildcard certificates Intermediate CA can be downloaded from the table below. Once your SSL certificate is installed, please use the SSL Certificate Installation Checker to ensure a problem-free SSL certificate experience for you and your users!

RSA SHA-1 SSL Certificates

Product Name	Intermediate CA
RapidSSL Wildcard FreeSSL	SO26462

RSA SHA-2 (under SHA-1 Root) SSL Certificates

Product Name	Intermediate CA
RapidSSL Wildcard FreeSSL	SO28616

RSA SHA-2 (under SHA-2 Root) SSL Certificates

Compatibility Intermediate CA	Issuer: GTE CyberTrust Global Root Valid until: 10/Aug/2018 Serial #: 0E:E0:68:2D:BB:98:2D:92:C6:85:6A:DA:DE:48:19:80 Thumbprint: F08B49D0EBE7975062CD19C731B141DF4D11DF52 Download
Cybertrust Japan Issuing CA-1	Issuer: Verizon Global Root CA Valid until: 01/Sep/2026 Serial #: 0C:5B:12:0D:AC:42:A1:CB:7B:20:89:DB:17:6E:04:78 Thumbprint: 4B8FE3B160D85B627F660C6A425059C2A420A774 Download
DigiCert Assured ID CA-1	Issuer: DigiCert Assured ID Root CA Valid until: 10/Nov/2021 Serial #: 06:FD:F9:03:96:03:AD:EA:00:0A:EB:3F:27:BB:BA:1B Thumbprint: 19A09B5A36F4DD99727DF783C17A51231A56C117 Download
DigiCert Assured ID CA G2	Issuer: DigiCert Assured ID Root G2 Valid until: 01/Aug/2028 Serial #: 0F:5F:CC:FC:AB:20:F3:DF:8E:6D:A3:D8:47:67:C2:93 Thumbprint: 28E96CDB1DBA273FD1A6151BE15F088F26046273 Download
DigiCert Assured ID CA G3	Issuer: DigiCert Assured ID Root G3 Valid until: 01/Aug/2028 Serial #: 01:05:DA:E2:55:AA:B2:95:4A:0D:B2:C9:E6:B5:32:2C Thumbprint: C619BE4F415453F46D020ED79F5D5CA5C37E14AD Download
DigiCert Assured ID Code Signing CA-1	Issuer: DigiCert Assured ID Root CA Valid until: 10/Feb/2026 Serial #: 0F:A8:49:06:15:D7:00:A0:BE:21:76:FD:C5:EC:6D:BD Thumbprint: 409AA4A74A0CDA7C0FEE6BD0BB8823D16B5F1875 Download

- No big deal
- Let's go to the vendor's documentation...
- And OMG...
- The vendor docs are terrible
- This is when you learn...
- Cryptography is hard but implementing cryptographic best practices might be even harder

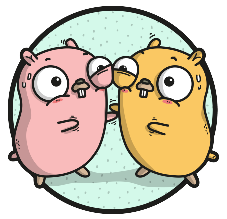
- What do we do?
- Look at statistical probabilities and start shuffling keys around?
- The series of games you have to play with openssl or nginx or some other method aren't intuitive
- Do you know how hard this is to explain to people?

SO WHAT DOES ANY GOOD ENGINEER DO?

- We needed a tool that would fail at the sign of an improper certificate chain
- We needed a lightweight tool that could be publicly accessible
- Conducting a third-party analysis of the certificates and configuration was also a requirement
- There were no tools that I could find meeting these needs
- I decided to build my own tool and I chose to build it with Go

LOG

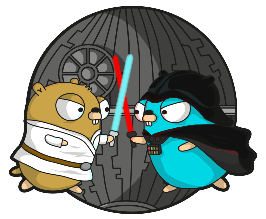
- The Go log package is pretty self explanatory
- Package that enables logging
- Needed a spectacular failure at the sign of trouble
- `log` has three helper functions: `print`, `fatal`, and `panic`



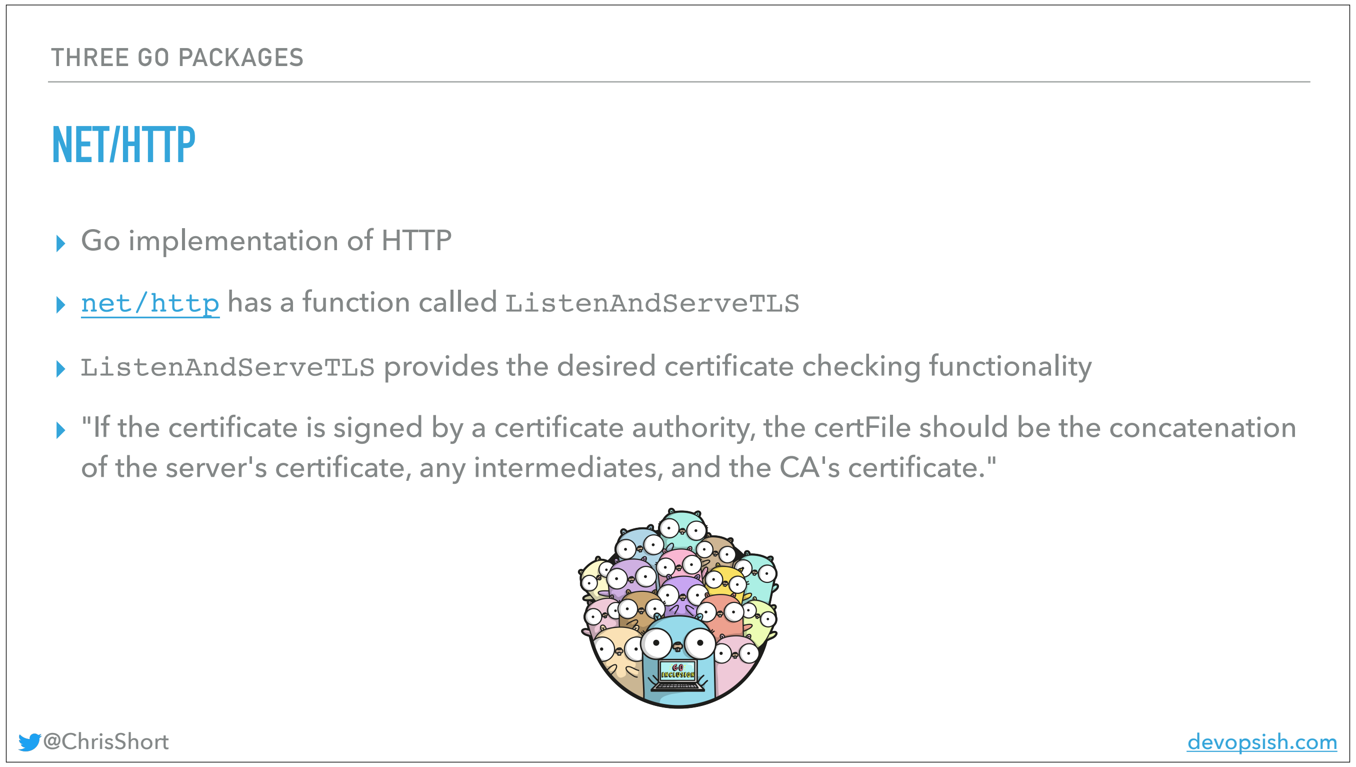
- Let me just say the Go standard library is amazing!
- log is designed beautifully
- Using fatal to break the app and log to stderr if something isn't right is great!

CRYPTO/TLS

- The Go `crypto/tls` package partially implements TLS 1.2, as specified in RFC-5246
- Package configures usable SSL/TLS versions
- Identifies preferred cipher suites and elliptic curves used during handshakes
- This is the package that handles connections securely



- The crypto/tls package is a splendid implementation of the RFC
- "It just works."



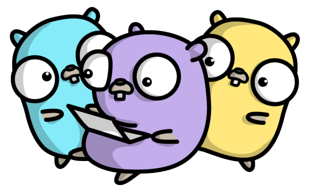
NET/HTTP

- Go implementation of HTTP
- `net/http` has a function called `ListenAndServeTLS`
- `ListenAndServeTLS` provides the desired certificate checking functionality
- "If the certificate is signed by a certificate authority, the certFile should be the concatenation of the server's certificate, any intermediates, and the CA's certificate."

- net/http has the ListenAndServeTLS function and it's awesome
- It fails if your certs aren't up to snuff or ordered properly
- It helps us Gophers out immensely by enforcing best practices

MAIN: MUX, CFG, SRV

- Code creates a `mux`, short for HTTP request multiplexer
- I ❤️ multiplexers (it's a long story that involves analog signals)
- `mux` has a function that creates an HTTP server with headers and content (Hello World!)
- `cfg` brings in all the TLS bits seen in a solid web server config
- `srv` puts the pieces together and defines what port to listen on

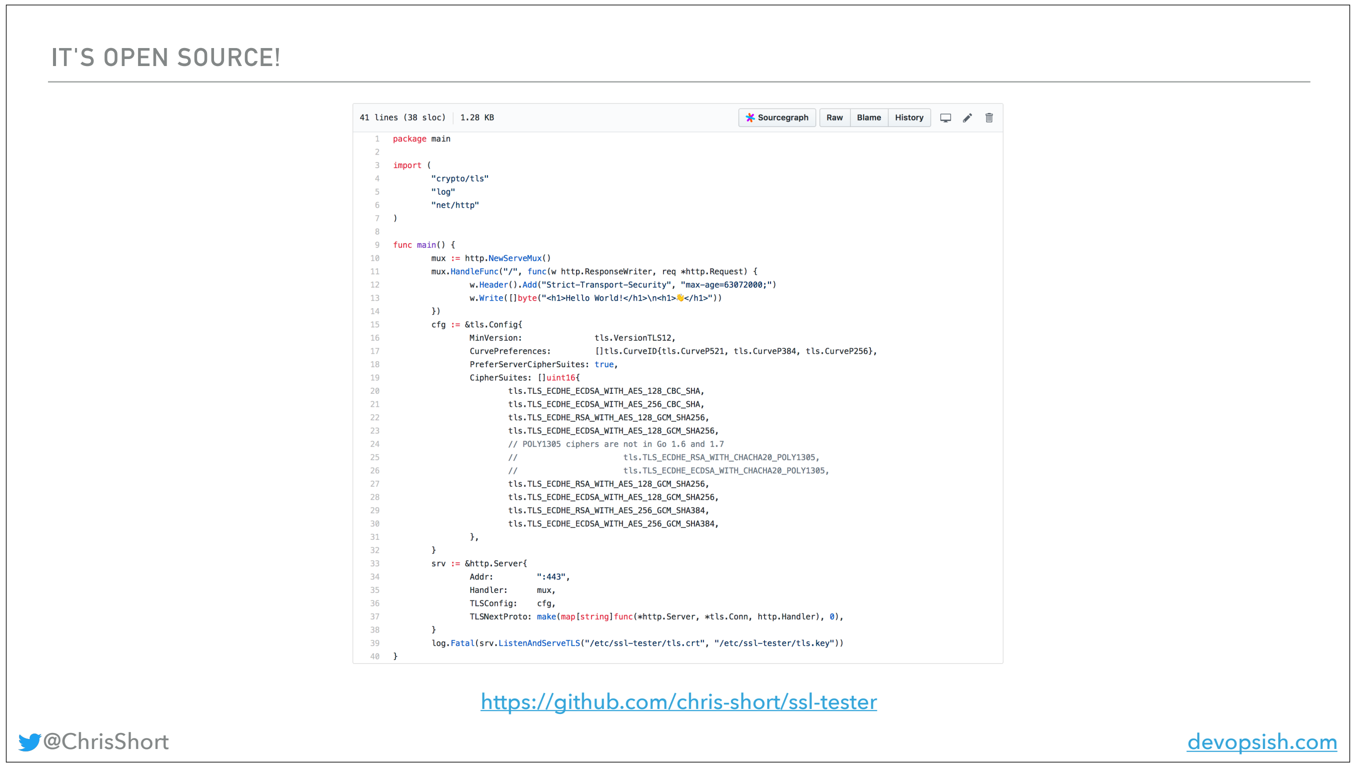


- READ SLIDE

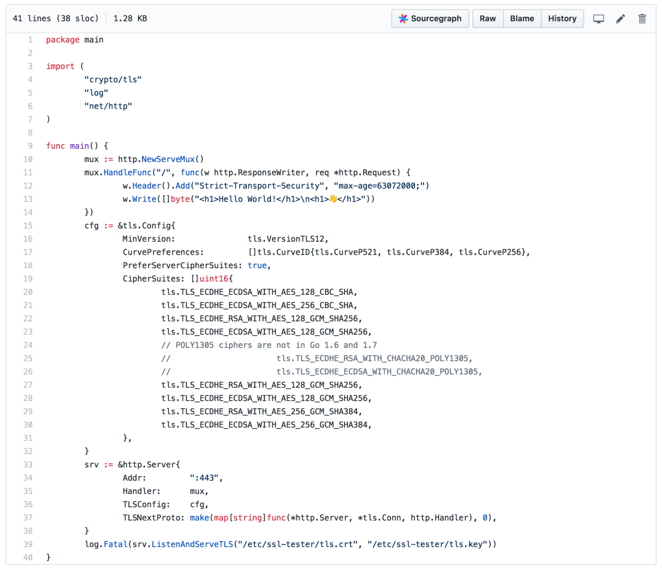
FAIL SPECTACULARLY

- I ❤️ DevOps and I embrace failure
- `log.Fatal(srv.ListenAndServeTLS("/etc/ssl-tester/tls.crt", "/etc/ssl-tester/tls.key"))`
- Defines path of certificate files to use
- Logs a fatal error if certificate is not valid
- Fails Fast

- I love DevOps and I embrace failure
- The code allows us to fail quickly if the certificates aren't in accordance with RFC
- Stuff in the standard library JUST WORKS



IT'S OPEN SOURCE!



41 lines (38 sloc) | 1.28 KB

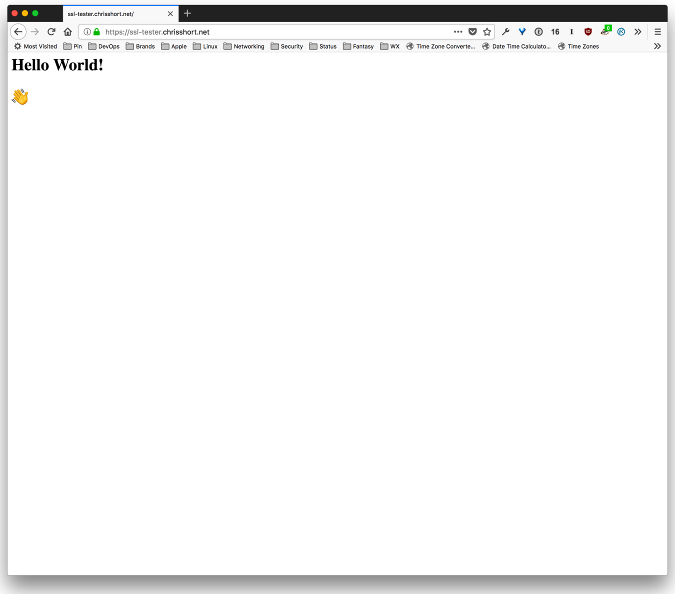
Sourcegraph Raw Blame History

```go
package main

import (
	"crypto/tls"
	"log"
	"net/http"
)

func main() {
	mux := http.NewServeMux()
	mux.HandleFunc("/", func(w http.ResponseWriter, req *http.Request) {
		w.Header().Add("Strict-Transport-Security", "max-age=63072000;")
		w.Write([]byte("<h1>Hello World!</h1>\n<h1>👋</h1>"))
	})
	cfg := &tls.Config{
		MinVersion:               tls.VersionTLS12,
		CurvePreferences:         []tls.CurveID{tls.CurveP521, tls.CurveP384, tls.CurveP256},
		PreferServerCipherSuites: true,
		CipherSuites: []uint16{
			tls.TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA,
			tls.TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA,
			tls.TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,
			tls.TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,
			// POLY1305 ciphers are not in Go 1.6 and 1.7
			//			tls.TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305,
			//			tls.TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305,
			tls.TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,
			tls.TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,
			tls.TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,
			tls.TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,
		},
	}
	srv := &http.Server{
		Addr:         ":443",
		Handler:      mux,
		TLSConfig:    cfg,
		TLSNextProto: make(map[string]func(*http.Server, *tls.Conn, http.Handler), 0),
	}
	log.Fatal(srv.ListenAndServeTLS("/etc/ssl-tester/tls.crt", "/etc/ssl-tester/tls.key"))
}
```

https://github.com/chris-short/ssl-tester

- The code is open source
- Check it out on GitHub
- Throw a star my way if you feel like it

IT WORKS!

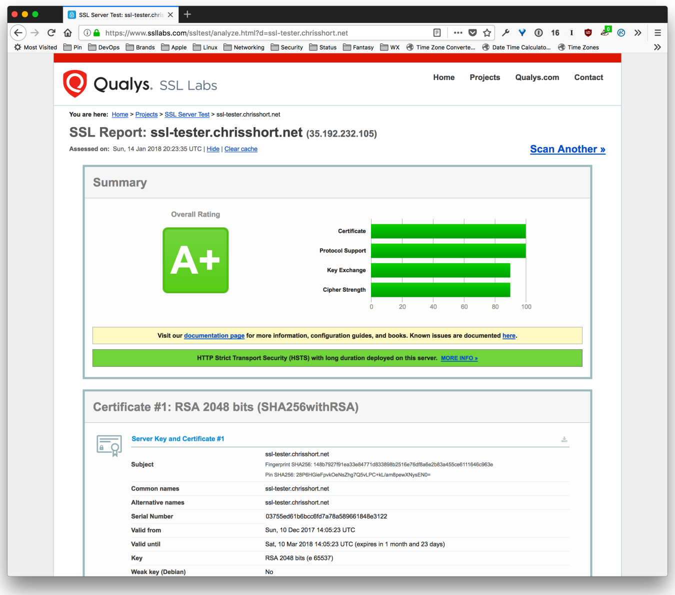


Hello World!

👋

- Yes it works!
- Yes you can access it right now!
- ssl-tester.chrisshort.net sends you to a reference implementation

NO. IT REALLY WORKS!



You are here: Home > Projects > SSL Server Test > ssl-tester.chrisshort.net

SSL Report: ssl-tester.chrisshort.net (35.192.232.105)

Assessed on: Sun, 14 Jan 2018 20:23:35 UTC | Hide | Clear cache

Scan Another »

Summary

Visit our documentation page for more information, configuration guides, and books. Known issues are documented here.

HTTP Strict Transport Security (HSTS) with long duration deployed on this server. MORE INFO »

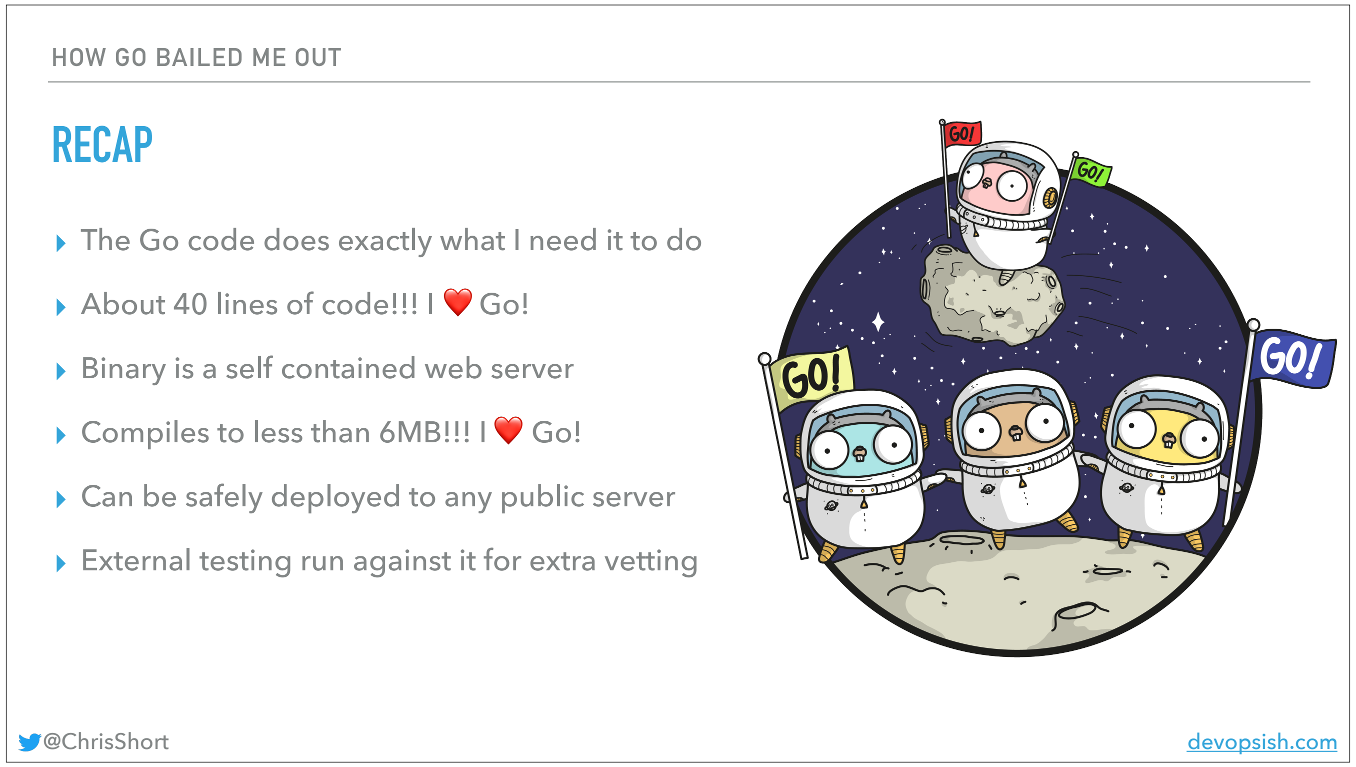
Certificate #1: RSA 2048 bits (SHA256withRSA)

Server Key and Certificate #1

Subject	ssl-tester.chrisshort.net Fingerprint SHA256: 148b7927f91ea33e84771d833898b2516e76df8a6e2b83a455ce6111646c963e Pin SHA256: 28P6HGieFpvkOeNsZhg7Q5vLPC+kL/am8pewXNysEN0=
Common names	ssl-tester.chrisshort.net
Alternative names	ssl-tester.chrisshort.net
Serial Number	03755ed61b6bcc6fd7a78a589661848e3122
Valid from	Sun, 10 Dec 2017 14:05:23 UTC
Valid until	Sat, 10 Mar 2018 14:05:23 UTC (expires in 1 month and 23 days)
Key	RSA 2048 bits (e 65537)
Weak key (Debian)	No

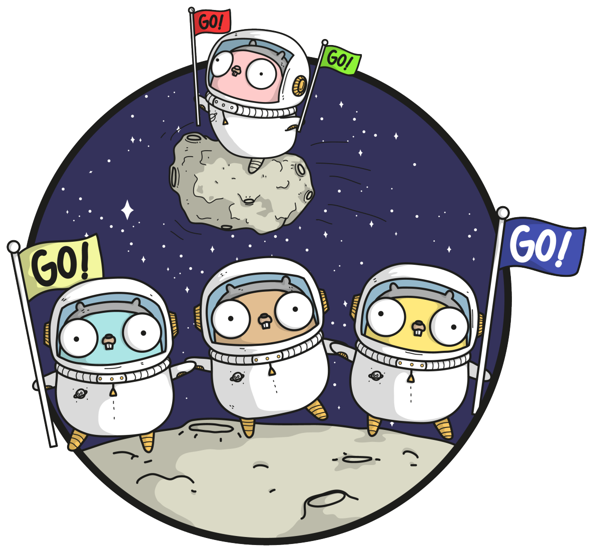
@ChrisShort devopsish.com

- You can even scan it with external tools!
- There's the A+



RECAP

- The Go code does exactly what I need it to do
- About 40 lines of code!!! I ❤️ Go!
- Binary is a self contained web server
- Compiles to less than 6MB!!! I ❤️ Go!
- Can be safely deployed to any public server
- External testing run against it for extra vetting



- The tool does exactly what I need it to do and nothing more
- It fails when the certificate chain provided is incorrect
- It's lightweight and publicly accessible
- I'm able to test via third-parties
- It's a tiny, single binary

@ChrisShort

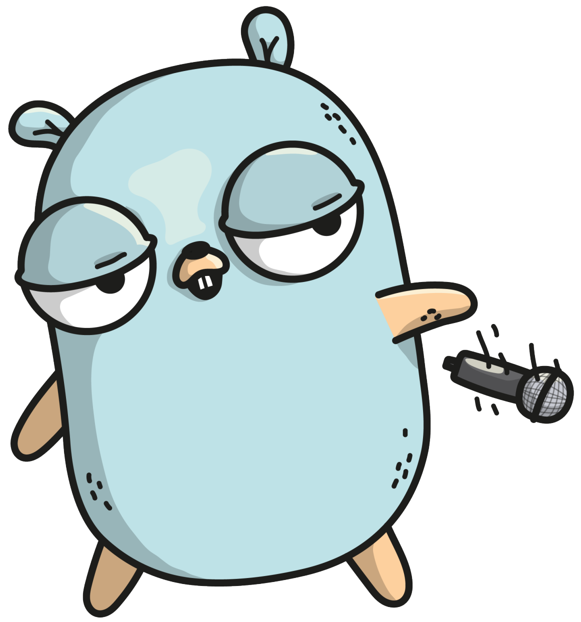
devopsish.com

- If you want to start learning go checkout tour.golang.org
- Or ping me for some book recommendations

CLEAR IS BETTER THAN CLEVER.

Rob Pike

CONCLUSION



@ChrisShort

devopsish.com