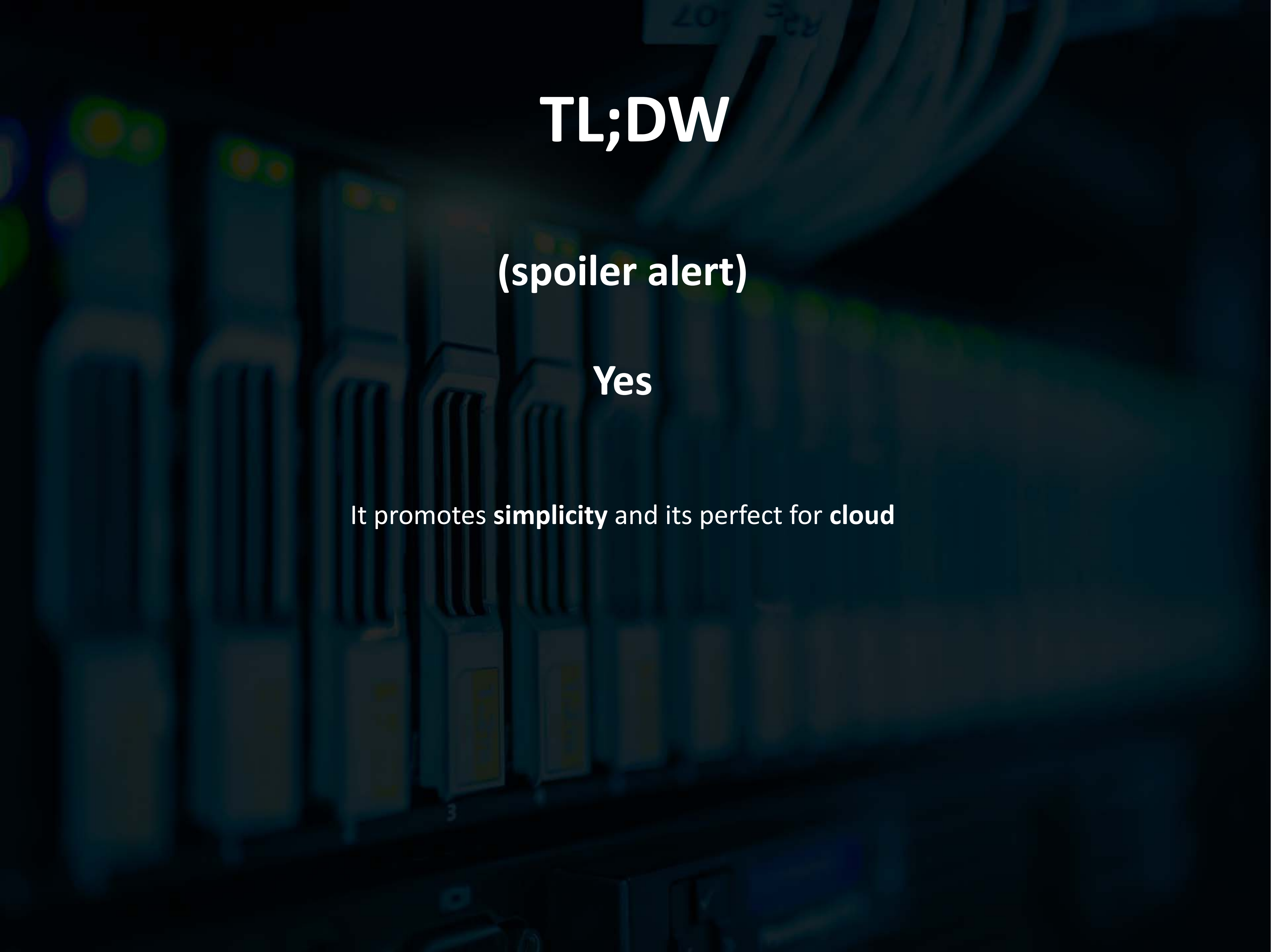
GO

Michał Wyziński

Be

SHAPING THE FUTURE



TL;DW

(spoiler alert)

Yes

It promotes **simplicity** and its perfect for **cloud**

Why

The Go programming language was conceived in late 2007 as an answer to some of the problems we were seeing developing software infrastructure at Google.

The computing landscape today is almost unrelated to the environment in which the languages being used, mostly C++, Java, and Python, had been created.

The problems introduced by **multi core processors, networked systems, massive computation clusters, and the web programming model** were being worked around rather than addressed head-on.

https://go.dev/talks/2012/splash.article
https://go.dev/blog/two-recent-go-articles

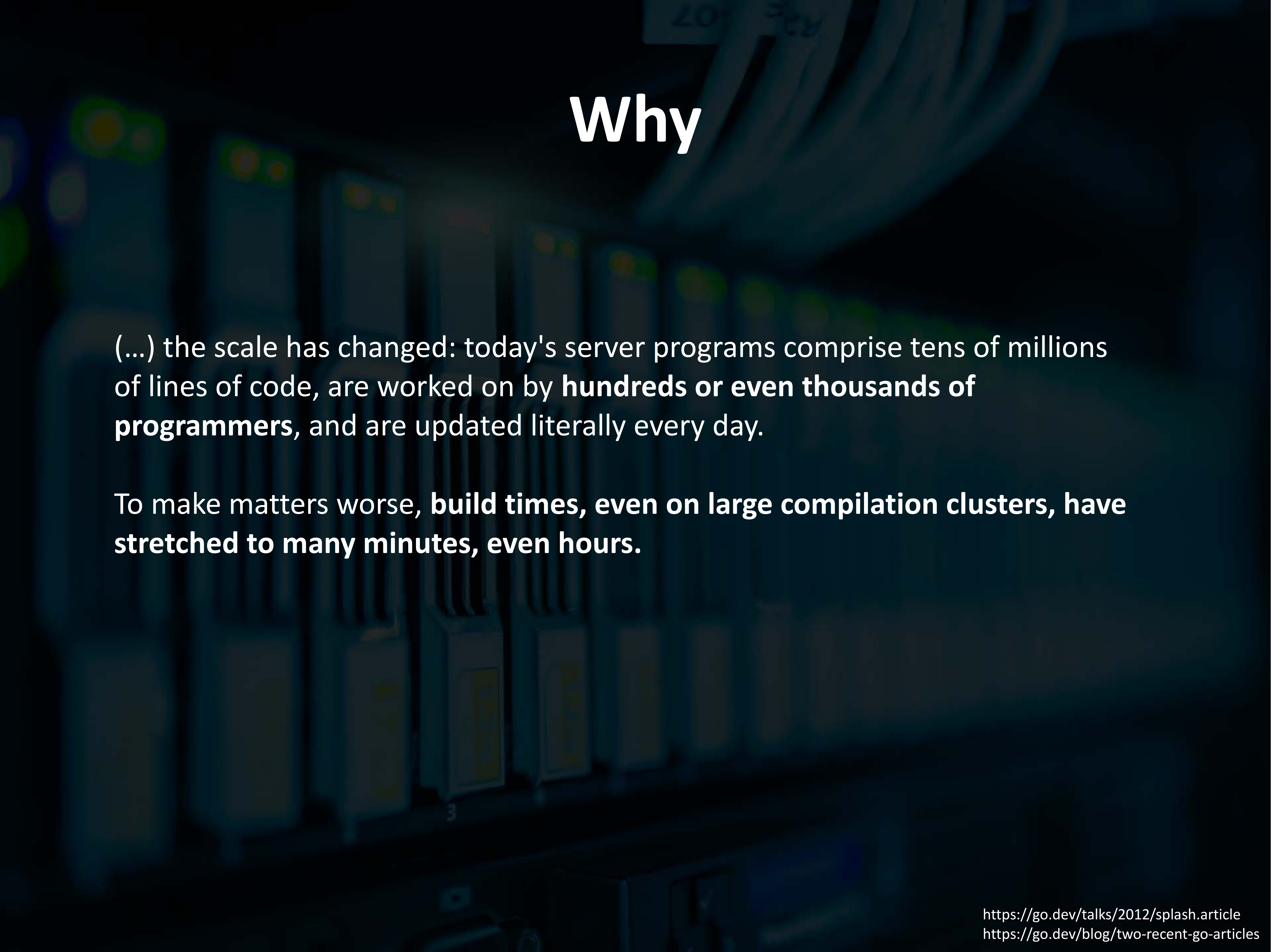
Why

The Go programming language was conceived in late 2007 as an answer to some of the problems we were seeing developing software infrastructure at Google.

The computing landscape today is almost unrelated to the environment in which the languages being used, mostly C++, Java, and Python, had been created.

The problems introduced by **multi core processors, networked systems, massive computation clusters, and the web programming model** were being worked around rather than addressed head-on.

https://go.dev/talks/2012/splash.article
https://go.dev/blog/two-recent-go-articles



Why

(…) the scale has changed: today's server programs comprise tens of millions of lines of code, are worked on by **hundreds or even thousands of programmers**, and are updated literally every day.

To make matters worse, **build times, even on large compilation clusters, have stretched to many minutes, even hours.**

https://go.dev/talks/2012/splash.article
https://go.dev/blog/two-recent-go-articles

Why – Google's Pain points

- Slow builds
- Uncontrolled dependencies
- Each programmer using a different subset of the language
- Poor program understanding (code hard to read, poorly documented, and so on)
- Duplication of effort
- Cost of updates
- Version skew
- Difficulty of writing automatic tools
- Cross-language builds

GO was designed to **address a set of software engineering issues that we had been exposed to in the construction of large server software**.

https://go.dev/talks/2012/splash.article
https://go.dev/blog/two-recent-go-articles

Go Authors

Ken Thompson - C / Unix
Rob Pike - Plan 9
Robert Griesemer - One of primary devs of Hotspot JVM

“(...) we started off with the idea that all three of us had to be talked into every feature in the language, so there was no **extraneous garbage put into the language for any reason**.”

Go's Characteristics

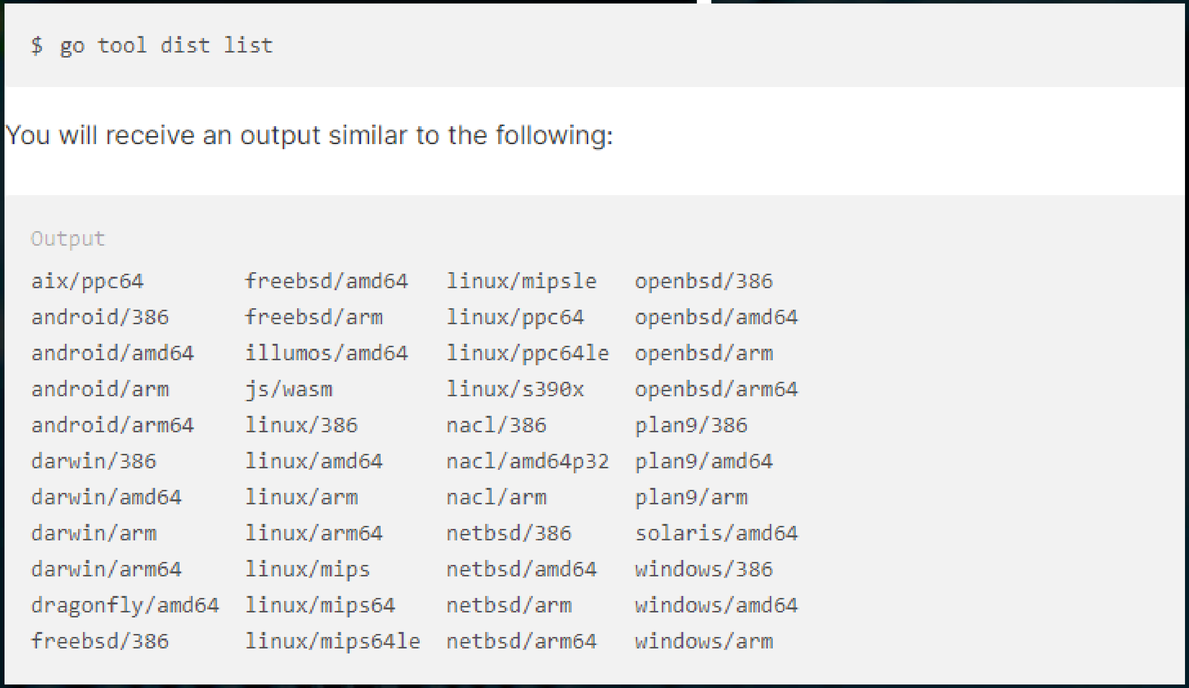
- C-like syntax, type-safety
- Compiled to native code
- Garbage Collected
- Concurrency build in
- Stdlib covering json serialization, http client/server, templates
- Meant to be used (also) in big crowded places like google
- One/Single programming style

Perfect for:

- Distributed systems
- Web/network apis (async I/O – connections, sockets, databases)
- Memory Locality
- Readability
- Low-level interfaces (processes, syscalls)

https://go.dev/talks/2012/splash.article
https://go.dev/blog/two-recent-go-articles

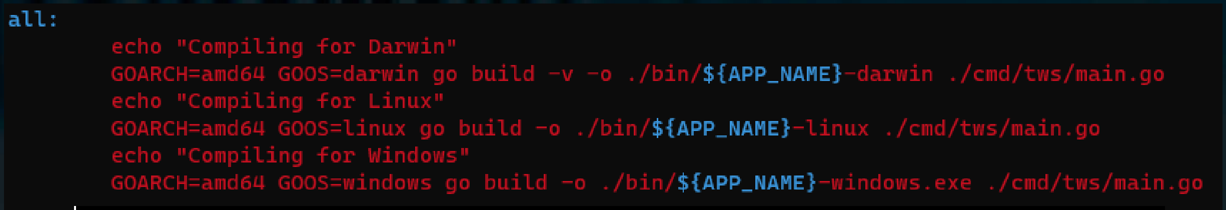
Cross compilation



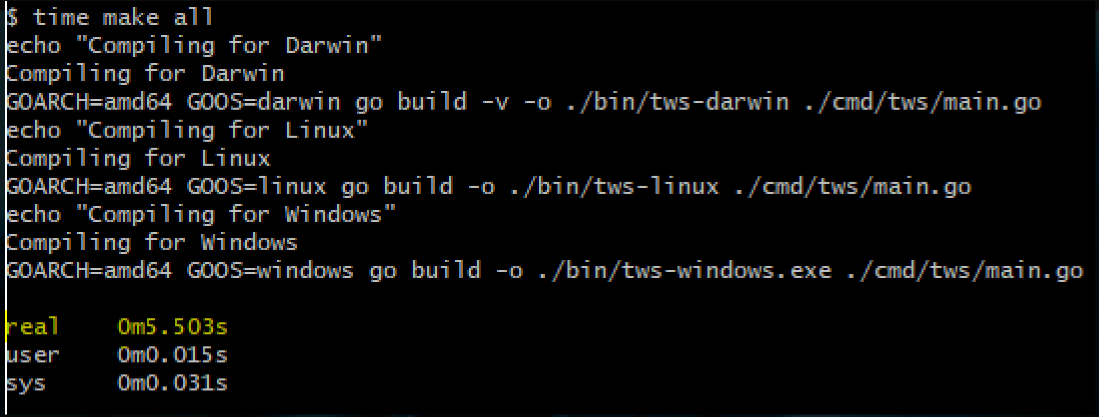
```
$ go tool dist list
```

You will receive an output similar to the following:

```
Output
aix/ppc64        freebsd/amd64   linux/mipsle   openbsd/386
android/386      freebsd/arm     linux/ppc64    openbsd/amd64
android/amd64    illumos/amd64   linux/ppc64le  openbsd/arm
android/arm      js/wasm         linux/s390x    openbsd/arm64
android/arm64    linux/386       nacl/386       plan9/386
darwin/386       linux/amd64     nacl/amd64p32  plan9/amd64
darwin/amd64     linux/arm       nacl/arm       plan9/arm
darwin/arm       linux/arm64     netbsd/386     solaris/amd64
darwin/arm64     linux/mips      netbsd/amd64   windows/386
dragonfly/amd64  linux/mips64    netbsd/arm     windows/amd64
freebsd/386      linux/mips64le  netbsd/arm64   windows/arm
```



```
all:
	echo "Compiling for Darwin"
	GOARCH=amd64 GOOS=darwin go build -v -o ./bin/${APP_NAME}-darwin ./cmd/tws/main.go
	echo "Compiling for Linux"
	GOARCH=amd64 GOOS=linux go build -o ./bin/${APP_NAME}-linux ./cmd/tws/main.go
	echo "Compiling for Windows"
	GOARCH=amd64 GOOS=windows go build -o ./bin/${APP_NAME}-windows.exe ./cmd/tws/main.go
```



```
$ time make all
echo "Compiling for Darwin"
Compiling for Darwin
GOARCH=amd64 GOOS=darwin go build -v -o ./bin/tws-darwin ./cmd/tws/main.go
echo "Compiling for Linux"
Compiling for Linux
GOARCH=amd64 GOOS=linux go build -o ./bin/tws-linux ./cmd/tws/main.go
echo "Compiling for Windows"
Compiling for Windows
GOARCH=amd64 GOOS=windows go build -o ./bin/tws-windows.exe ./cmd/tws/main.go

real    0m5.503s
user    0m0.015s
sys     0m0.031s
```

Readability & Simplicity

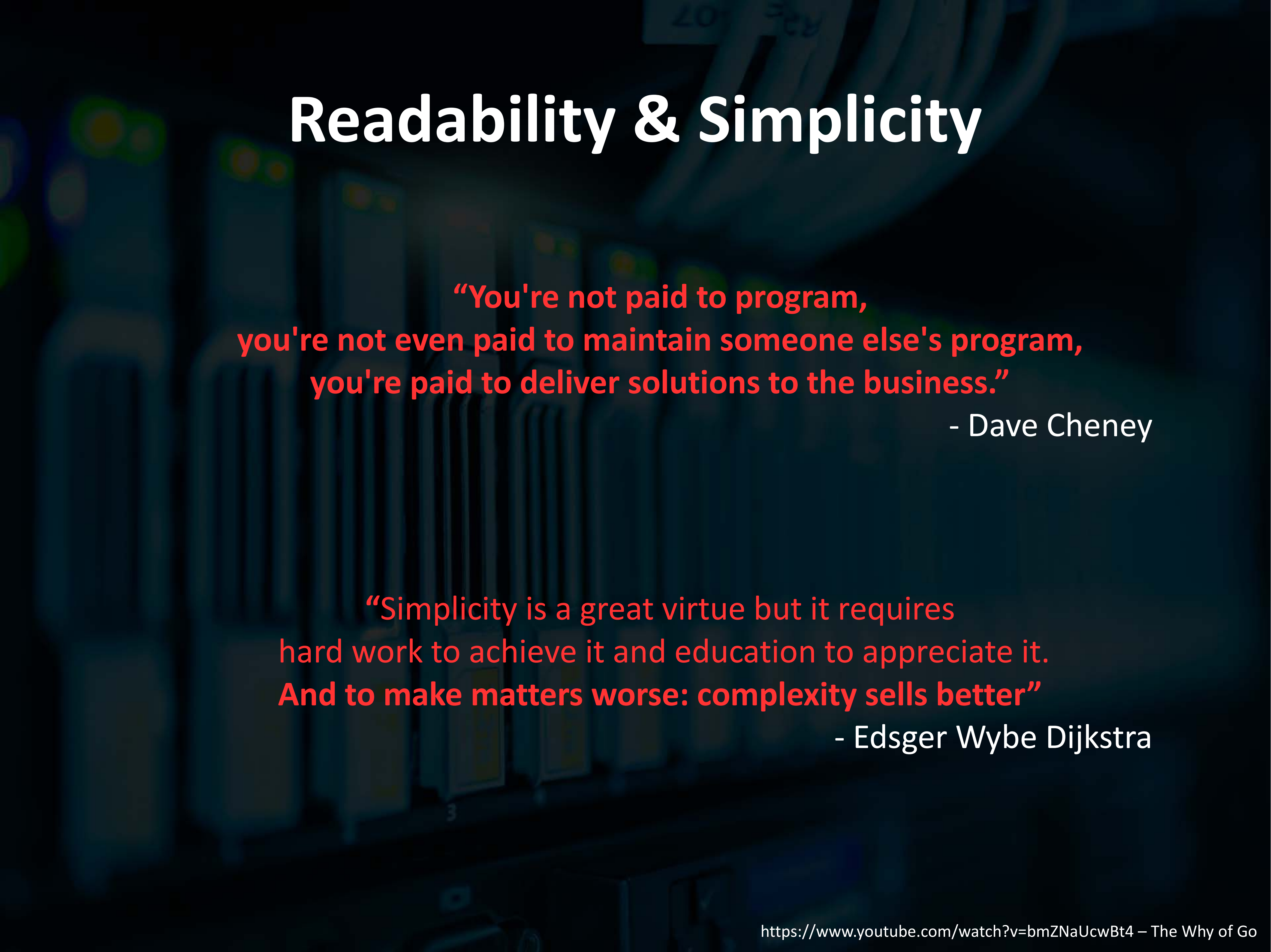
Readability is the defining quality of good code.
(Code is read over and over, written usually once)

Go isn't about revolution. It's about simplicity = Minimalism

You can't add things in order to make something simpler.
(Go does not compete on features)

Clear is better than clever.
Smart, complex code != business problems solved

https://www.youtube.com/watch?v=rFejpH_tAHM&ab_channel=dotconferences - Simplicity is complicated
http://go-proverbs.github.io/



Readability & Simplicity

“You're not paid to program,
you're not even paid to maintain someone else's program,
you're paid to deliver solutions to the business.”

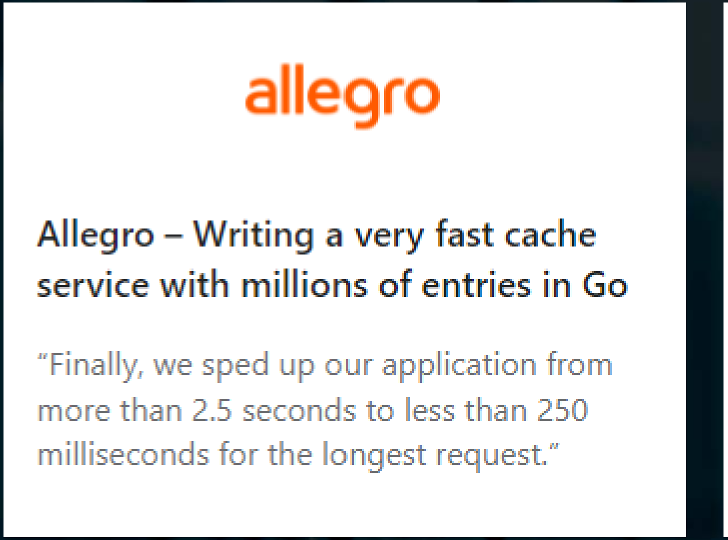
- Dave Cheney

“Simplicity is a great virtue but it requires
hard work to achieve it and education to appreciate it.
And to make matters worse: complexity sells better”

- Edsger Wybe Dijkstra

https://www.youtube.com/watch?v=bmZNaUcwBt4 – The Why of Go docker

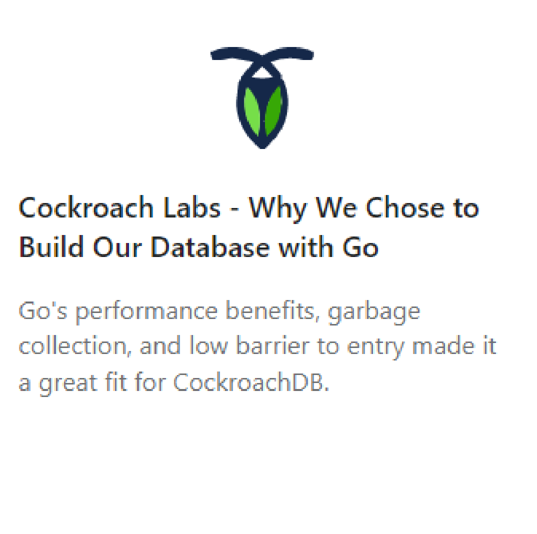
influxdb



allegro

Allegro – Writing a very fast cache service with millions of entries in Go

"Finally, we sped up our application from more than 2.5 seconds to less than 250 milliseconds for the longest request."



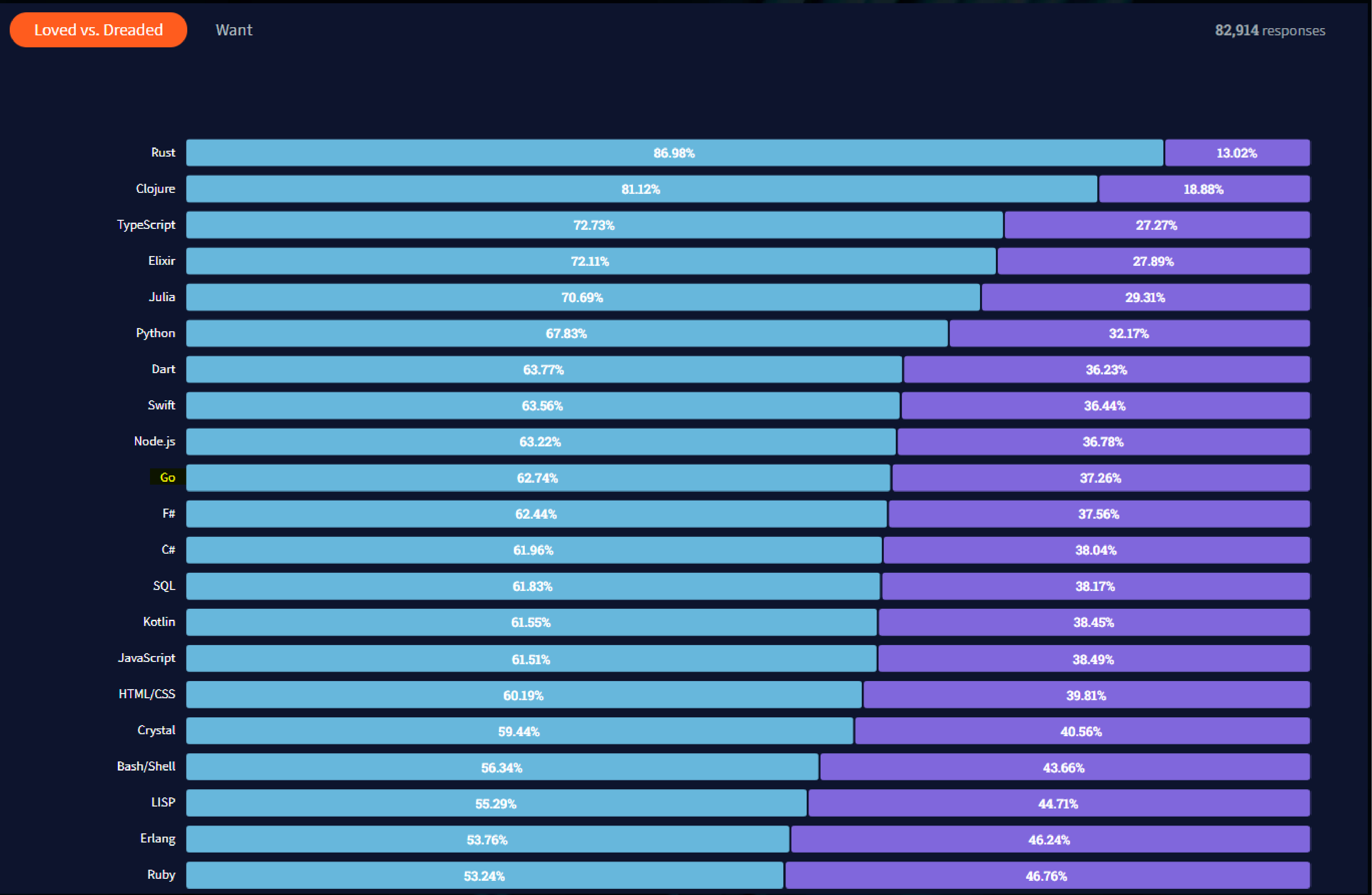
Cockroach Labs - Why We Chose to Build Our Database with Go

Go's performance benefits, garbage collection, and low barrier to entry made it a great fit for CockroachDB.

HashiCorp

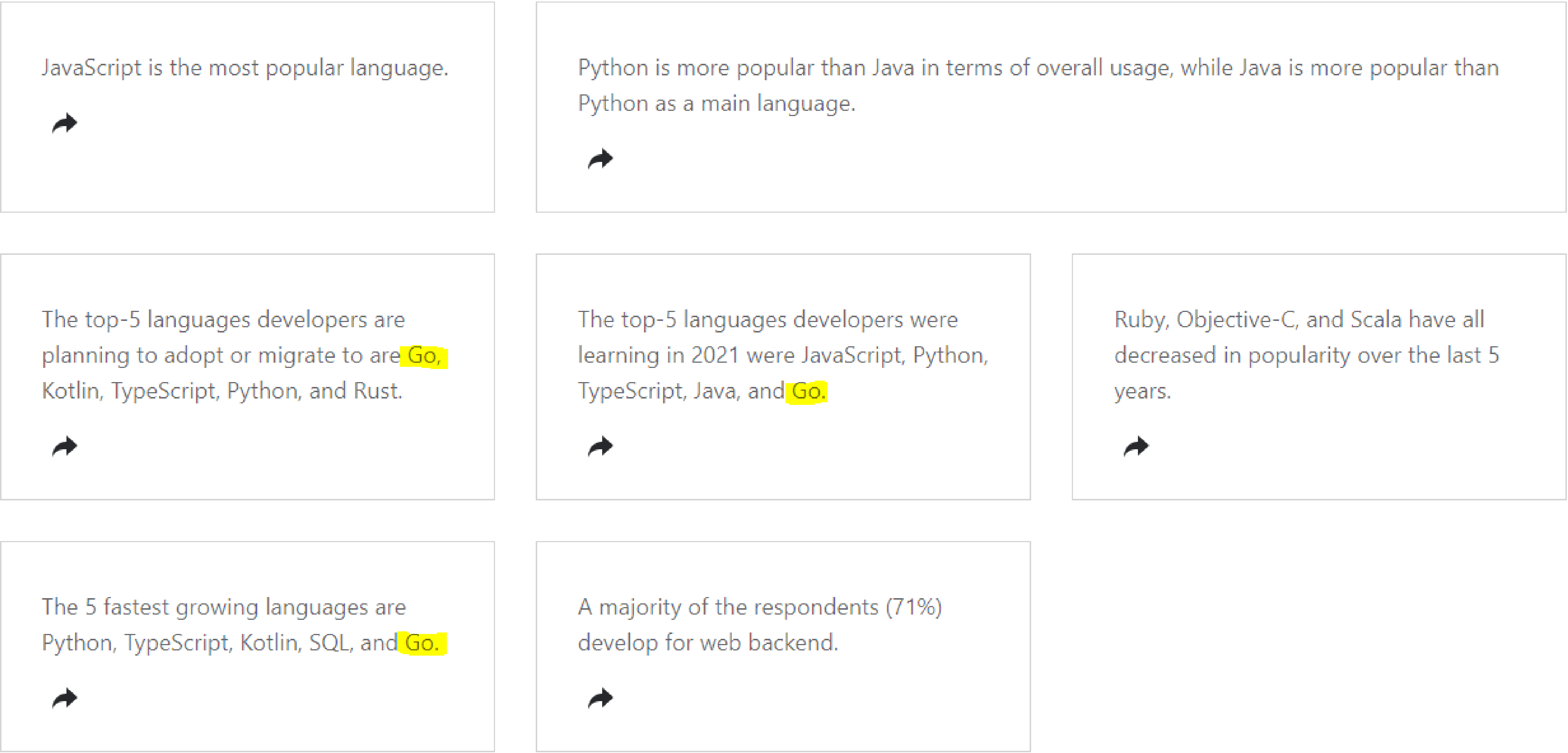
- consul
- nomad
- terraform

https://www.slideshare.net/jpetazzo/docker-and-go-why-did-we-decide-to-write-docker-in-go
https://blog.gopheracademy.com/birthday-bash-2014/kubernetes-go-crazy-delicious/
https://go.dev/solutions/#case-studies



https://insights.stackoverflow.com/survey/2021#most-loved-dreaded-and-wanted-language-love-dread

The State of Developer Ecosystem 2021



JavaScript is the most popular language.

Python is more popular than Java in terms of overall usage, while Java is more popular than Python as a main language.

The top-5 languages developers are planning to adopt or migrate to are Go, Kotlin, TypeScript, Python, and Rust.

The top-5 languages developers were learning in 2021 were JavaScript, Python, TypeScript, Java, and Go.

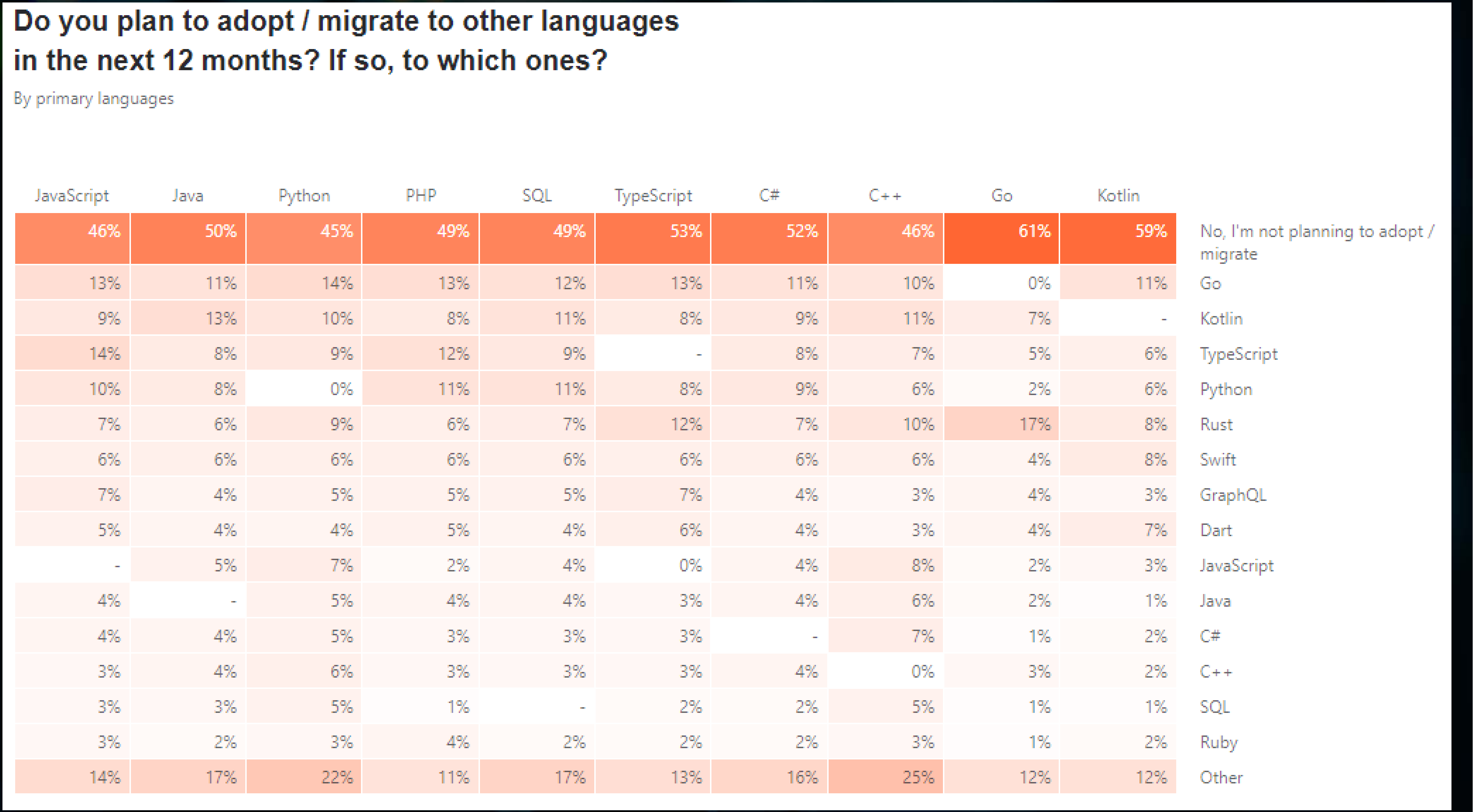
Ruby, Objective-C, and Scala have all decreased in popularity over the last 5 years.

The 5 fastest growing languages are Python, TypeScript, Kotlin, SQL, and Go.

A majority of the respondents (71%) develop for web backend.

https://www.jetbrains.com/lp/devecosystem-2021/

The State of Developer Ecosystem 2021



Do you plan to adopt / migrate to other languages in the next 12 months? If so, to which ones?

By primary languages

JavaScript	Java	Python	PHP	SQL	TypeScript	C#	C++	Go	Kotlin	
46%	50%	45%	49%	49%	53%	52%	46%	61%	59%	No, I'm not planning to adopt / migrate
13%	11%	14%	13%	12%	13%	11%	10%	0%	11%	Go
9%	13%	10%	8%	11%	8%	9%	11%	7%	-	Kotlin
14%	8%	9%	12%	9%	-	8%	7%	5%	6%	TypeScript
10%	8%	0%	11%	11%	8%	9%	6%	2%	6%	Python
7%	6%	9%	6%	7%	12%	7%	10%	17%	8%	Rust
6%	6%	6%	6%	6%	6%	6%	6%	4%	8%	Swift
7%	4%	5%	5%	5%	7%	4%	3%	4%	3%	GraphQL
5%	4%	4%	5%	4%	6%	4%	3%	4%	7%	Dart
-	5%	7%	2%	4%	0%	4%	8%	2%	3%	JavaScript
4%	-	5%	4%	4%	3%	4%	6%	2%	1%	Java
4%	4%	5%	3%	3%	3%	-	7%	1%	2%	C#
3%	4%	6%	3%	3%	3%	4%	0%	3%	2%	C++
3%	3%	5%	1%	-	2%	2%	5%	1%	1%	SQL
3%	2%	3%	4%	2%	2%	2%	3%	1%	2%	Ruby
14%	17%	22%	11%	17%	13%	16%	25%	12%	12%	Other

https://www.jetbrains.com/lp/devecosystem-2021/

NETFLIX Rend

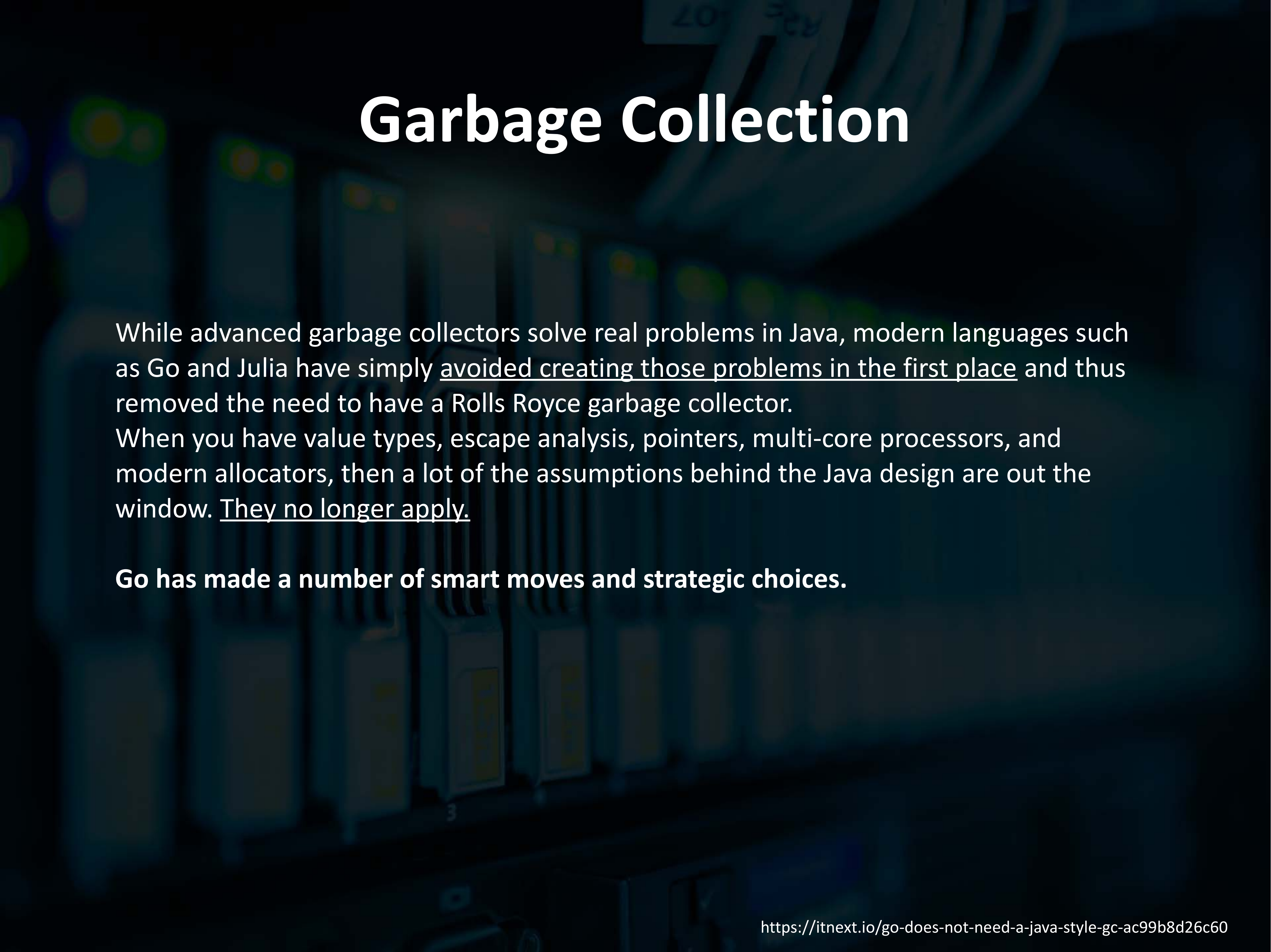
(…) the decision to use Go was deliberate, because we needed something that had lower latency than Java (where garbage collection pauses are an issue) and is more productive for developers than C, while also handling tens of thousands of client connections. **Go fits this space well**.

https://netflixtechblog.com/application-data-caching-using-ssds-5bf25df851ef

twitch

Go's approach to concurrency involves making it very cheap to have a large number of goroutines. While a program using 10,000 OS threads might perform poorly, that number of goroutines is nothing unusual.

https://blog.twitch.tv/en/2016/07/05/gos-march-to-low-latency-gc-a6fa96f06eb7/



Garbage Collection

While advanced garbage collectors solve real problems in Java, modern languages such as Go and Julia have simply avoided creating those problems in the first place and thus removed the need to have a Rolls Royce garbage collector.
When you have value types, escape analysis, pointers, multi-core processors, and modern allocators, then a lot of the assumptions behind the Java design are out the window. They no longer apply.

Go has made a number of smart moves and strategic choices.

https://itnext.io/go-does-not-need-a-java-style-gc-ac99b8d26c60

Concurrency

Using synchronization primitives

Goroutines

- simple model → function executing concurrently with other goroutines in the same address space.

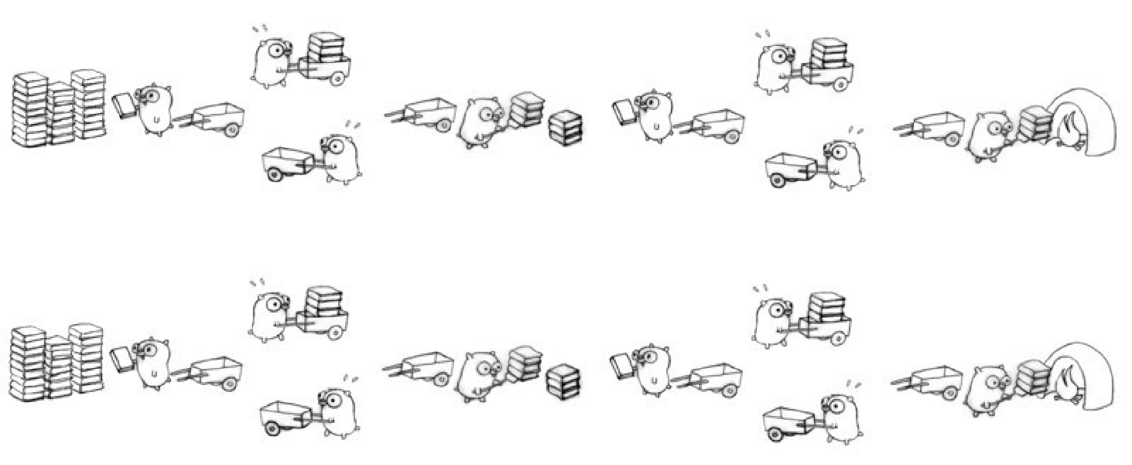
- lightweight, extra cheap

allocating (and freeing) heap storage as required

- multiplexed onto multiple OS threads

(if one should block, such as while waiting for I/O, others continue to run)

- hides many of the complexities of thread creation and management



https://talks.golang.org/2012/waza.slide#1 - Concurrency is not Parallelism

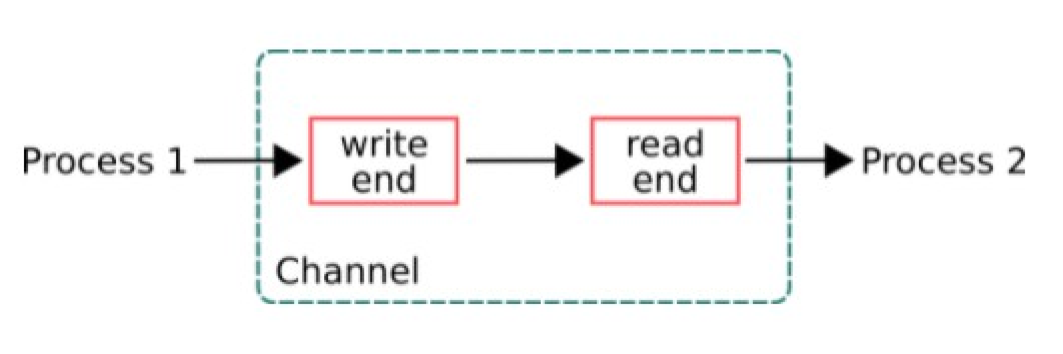
Goroutines

CSP

**Do not communicate by sharing memory;
instead, share memory by communicating.**

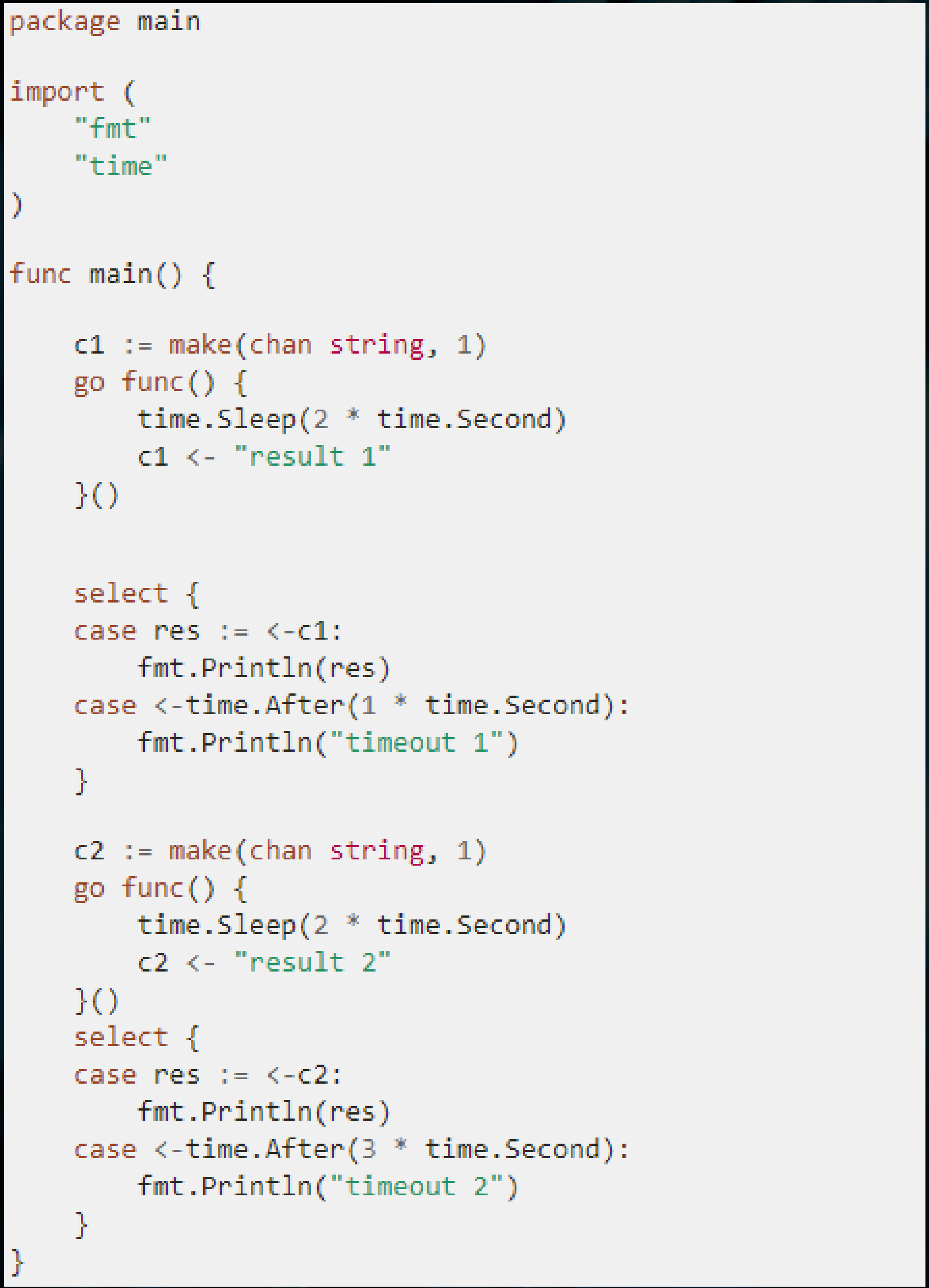
Go's approach to concurrency originates in Hoare's Communicating Sequential Processes (CSP – 1978!), it can also be seen as a type-safe generalization of Unix pipes.

“blocking” semantics by default → linear code



http://www.usingcsp.com/
https://arild.github.io/csp-presentation/#1

Do not communicate by sharing memory; instead, share memory by communicating.



```go
package main

import (
    "fmt"
    "time"
)

func main() {

    c1 := make(chan string, 1)
    go func() {
        time.Sleep(2 * time.Second)
        c1 <- "result 1"
    }()


    select {
    case res := <-c1:
        fmt.Println(res)
    case <-time.After(1 * time.Second):
        fmt.Println("timeout 1")
    }

    c2 := make(chan string, 1)
    go func() {
        time.Sleep(2 * time.Second)
        c2 <- "result 2"
    }()
    select {
    case res := <-c2:
        fmt.Println(res)
    case <-time.After(3 * time.Second):
        fmt.Println("timeout 2")
    }
}
```

https://gobyexample.com/timeouts

Trying Go

As little magic as possible → Magic is bad

“Humble yourself, bite your tongue, and study. You will realize eventually that what you first thought was worth criticizing was **actually a deep work of genius**.”

“If you are coming from another language, be it a dynamic language like Python or Ruby, or a compiled language like Java or C#, **leave your OO baggage at the door**.”

“By removing inheritance from the language, the opportunity to practice the mantra of composition over inheritance is made manifest, and **fighting it will only lead to frustration**.”

“Every language is different and should be considered with a **fresh perspective**. The point here is to try to **share the same view as the language’s creators** (and its core community).”

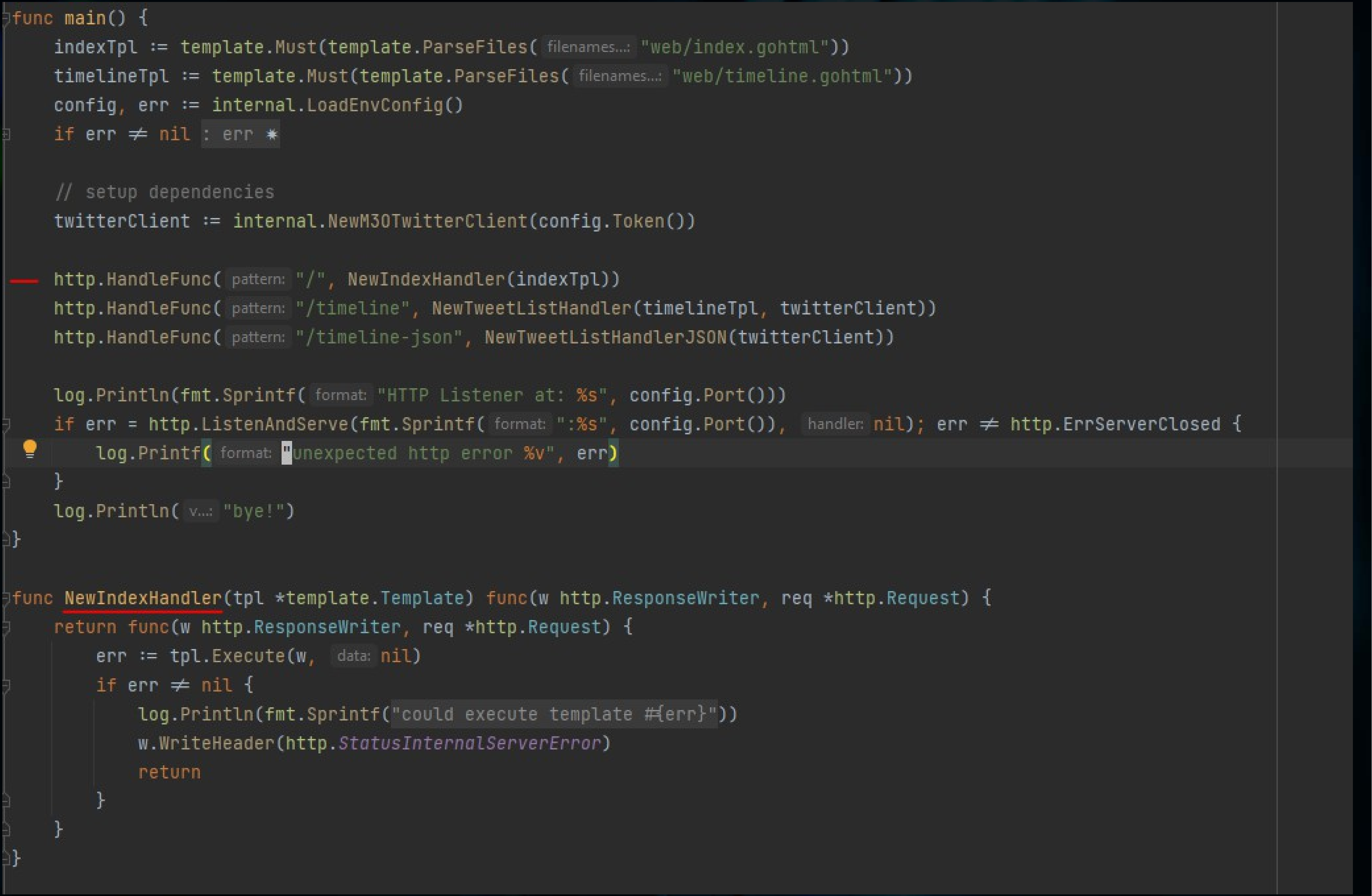
https://www.cloudbees.com/blog/best-practices-for-a-new-go-developer

Trying Go

“Don’t try to force your previous language’s approach into Go.
Each language has different values, a different world view, and you can’t just mix and match based on what you are used to.”

https://www.cloudbees.com/blog/best-practices-for-a-new-go-developer

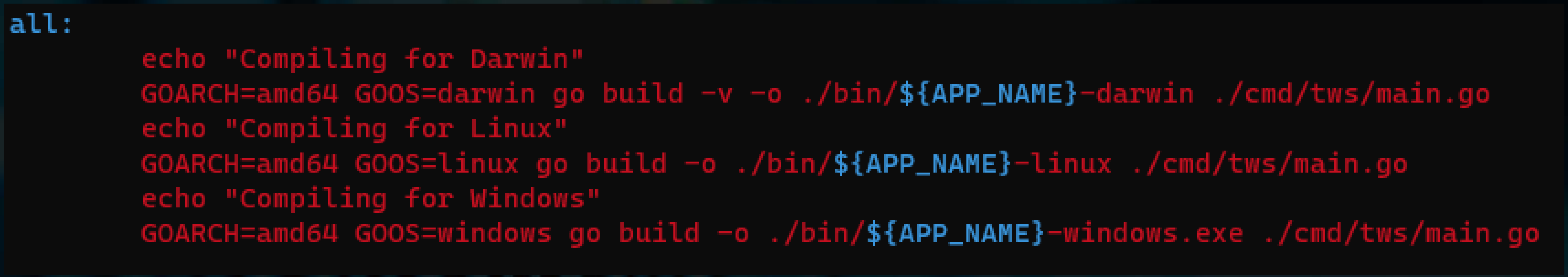
Example



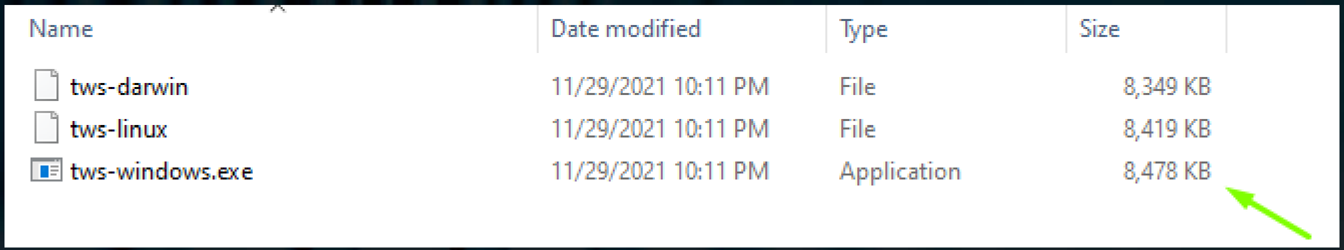
```
func main() {
    indexTpl := template.Must(template.ParseFiles( filenames...: "web/index.gohtml"))
    timelineTpl := template.Must(template.ParseFiles( filenames...: "web/timeline.gohtml"))
    config, err := internal.LoadEnvConfig()
    if err ≠ nil : err *

    // setup dependencies
    twitterClient := internal.NewM3OTwitterClient(config.Token())

    http.HandleFunc( pattern: "/", NewIndexHandler(indexTpl))
    http.HandleFunc( pattern: "/timeline", NewTweetListHandler(timelineTpl, twitterClient))
    http.HandleFunc( pattern: "/timeline-json", NewTweetListHandlerJSON(twitterClient))

    log.Println(fmt.Sprintf( format: "HTTP Listener at: %s", config.Port()))
    if err = http.ListenAndServe(fmt.Sprintf( format: ":%s", config.Port()),  handler: nil); err ≠ http.ErrServerClosed {
        log.Printf( format: "unexpected http error %v", err)
    }
    log.Println( v...: "bye!")
}

func NewIndexHandler(tpl *template.Template) func(w http.ResponseWriter, req *http.Request) {
    return func(w http.ResponseWriter, req *http.Request) {
        err := tpl.Execute(w,  data: nil)
        if err ≠ nil {
            log.Println(fmt.Sprintf("could execute template #{err}"))
            w.WriteHeader(http.StatusInternalServerError)
            return
        }
    }
}
```

https://github.com/mikwys/go-example

Example



```
all:
	echo "Compiling for Darwin"
	GOARCH=amd64 GOOS=darwin go build -v -o ./bin/${APP_NAME}-darwin ./cmd/tws/main.go
	echo "Compiling for Linux"
	GOARCH=amd64 GOOS=linux go build -o ./bin/${APP_NAME}-linux ./cmd/tws/main.go
	echo "Compiling for Windows"
	GOARCH=amd64 GOOS=windows go build -o ./bin/${APP_NAME}-windows.exe ./cmd/tws/main.go
```



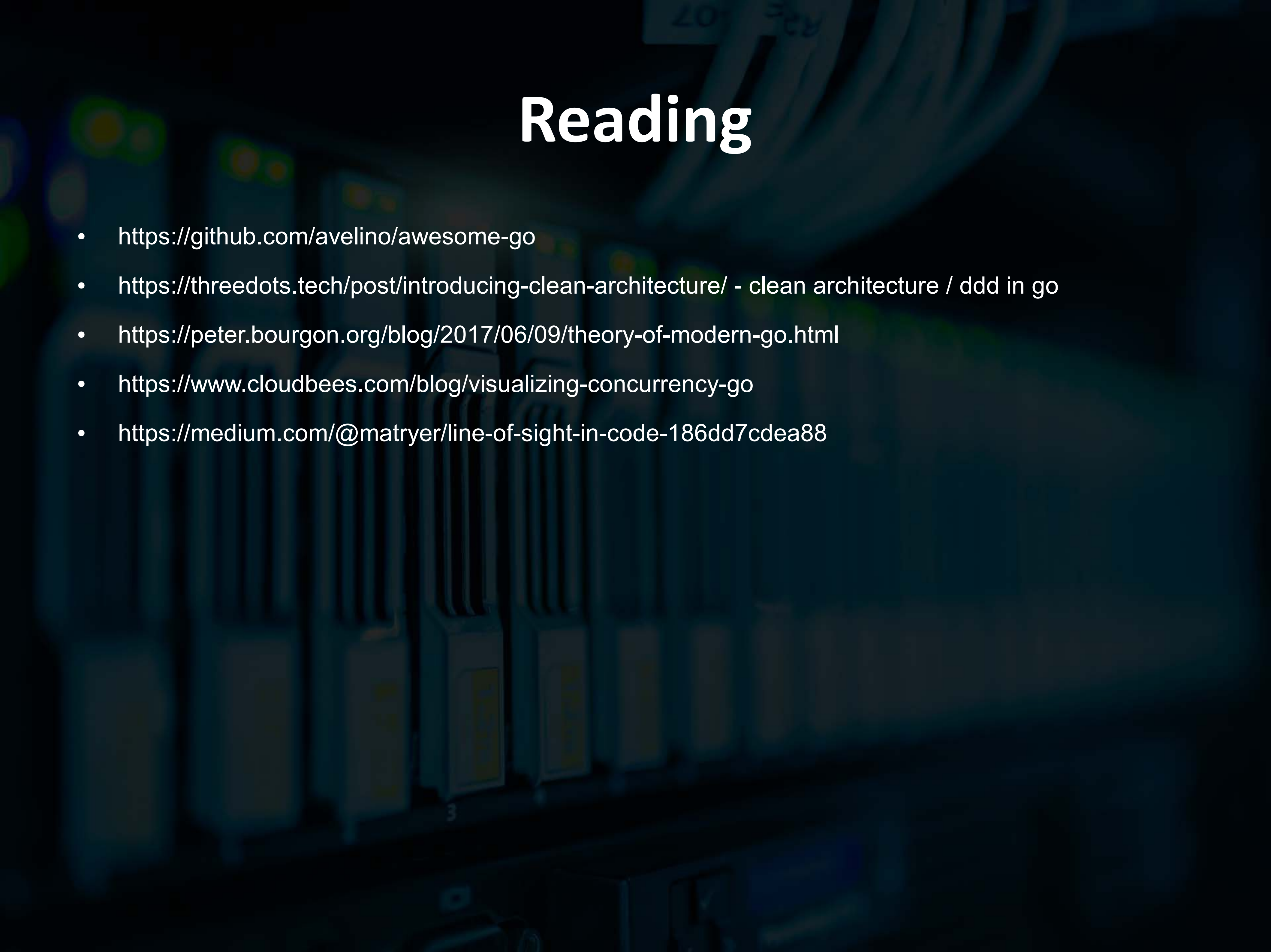
Name	Date modified	Type	Size
tws-darwin	11/29/2021 10:11 PM	File	8,349 KB
tws-linux	11/29/2021 10:11 PM	File	8,419 KB
tws-windows.exe	11/29/2021 10:11 PM	Application	8,478 KB

https://github.com/mikwys/go-example

Should I try?

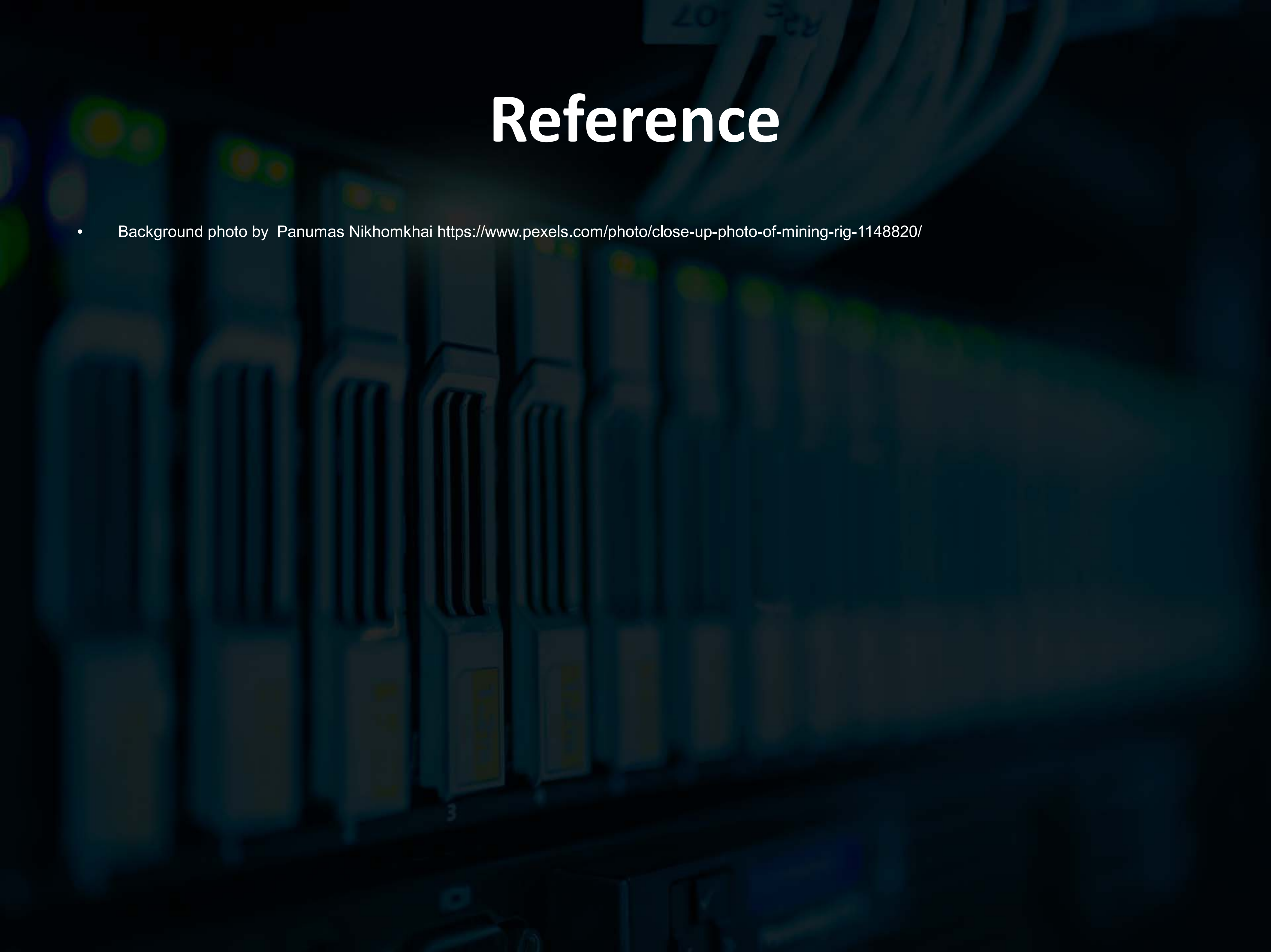
Further learning

- https://go.dev/tour/list - A tour of go
- https://go.dev/doc/effective_go - Effective Go
- https://gobyexample.com/ - Go code examples & snippets
- https://gowebexamples.com/ - Go Web code examples & snippets
- https://www.youtube.com/channel/UC_BzFbxG2za3bp5NRRRXJSw - justforfunc: Programming in Go
- https://github.com/golang/go/wiki/CodeReviewComments - Code style and good practices
- https://github.com/uber-go/guide/blob/master/style.md – Uber Go Style Guide
- https://github.com/avelino/awesome-go



Reading

- https://github.com/avelino/awesome-go
- https://threedots.tech/post/introducing-clean-architecture/ - clean architecture / ddd in go
- https://peter.bourgon.org/blog/2017/06/09/theory-of-modern-go.html
- https://www.cloudbees.com/blog/visualizing-concurrency-go
- https://medium.com/@matryer/line-of-sight-in-code-186dd7cdea88



Reference

- Background photo by Panumas Nikhomkhai https://www.pexels.com/photo/close-up-photo-of-mining-rig-1148820/