Concurrent Programming for PI Developers

Christopher Sawyer
Technology Enablement Team

Your Speaker Today

- Christopher Sawyer
- Technology Enablement
- csawyer@osisoft.com

Why This Talk?

Why This Talk?

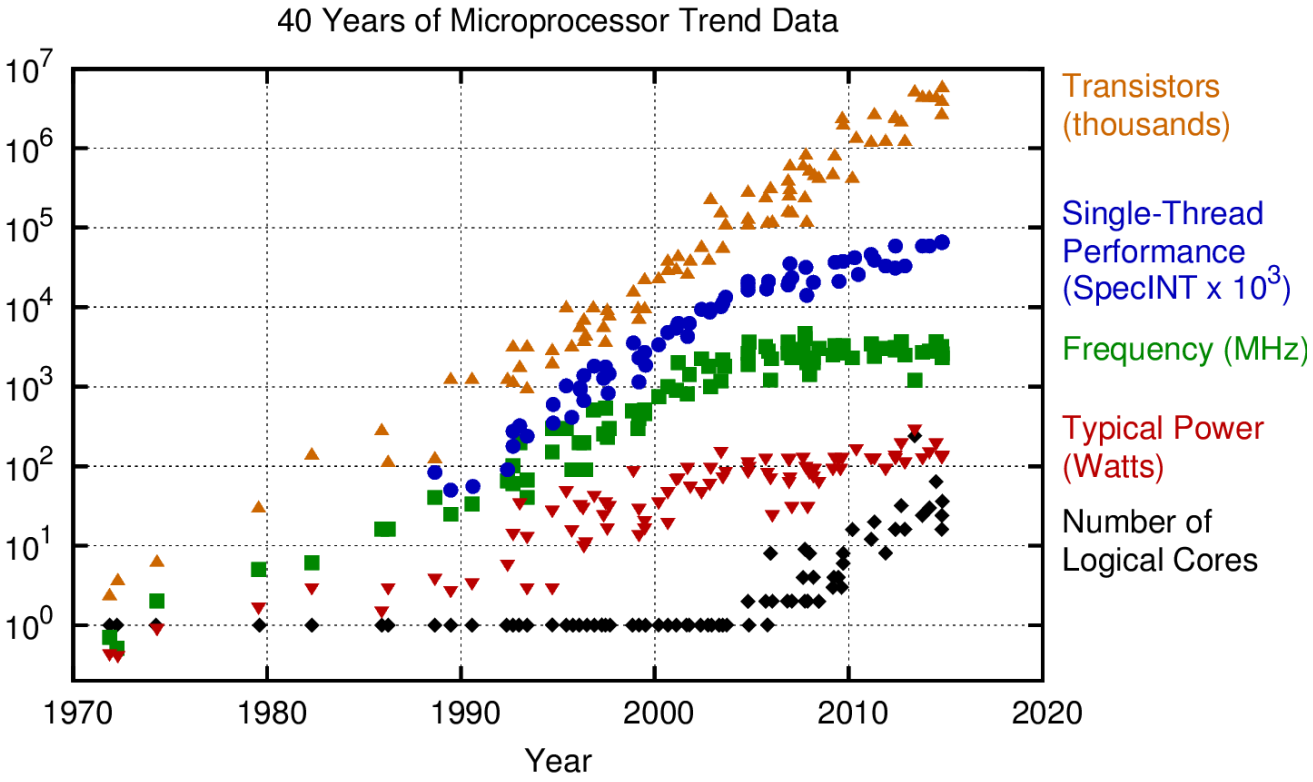
- This talk is an experiment, actually ;-)
- Expose you to something new
- Make it easy to figure out
- Maybe you recycle what you learn back into your own projects
- We learn how to build great software, together

Software is at a crossroad...

Software is at a crossroad...

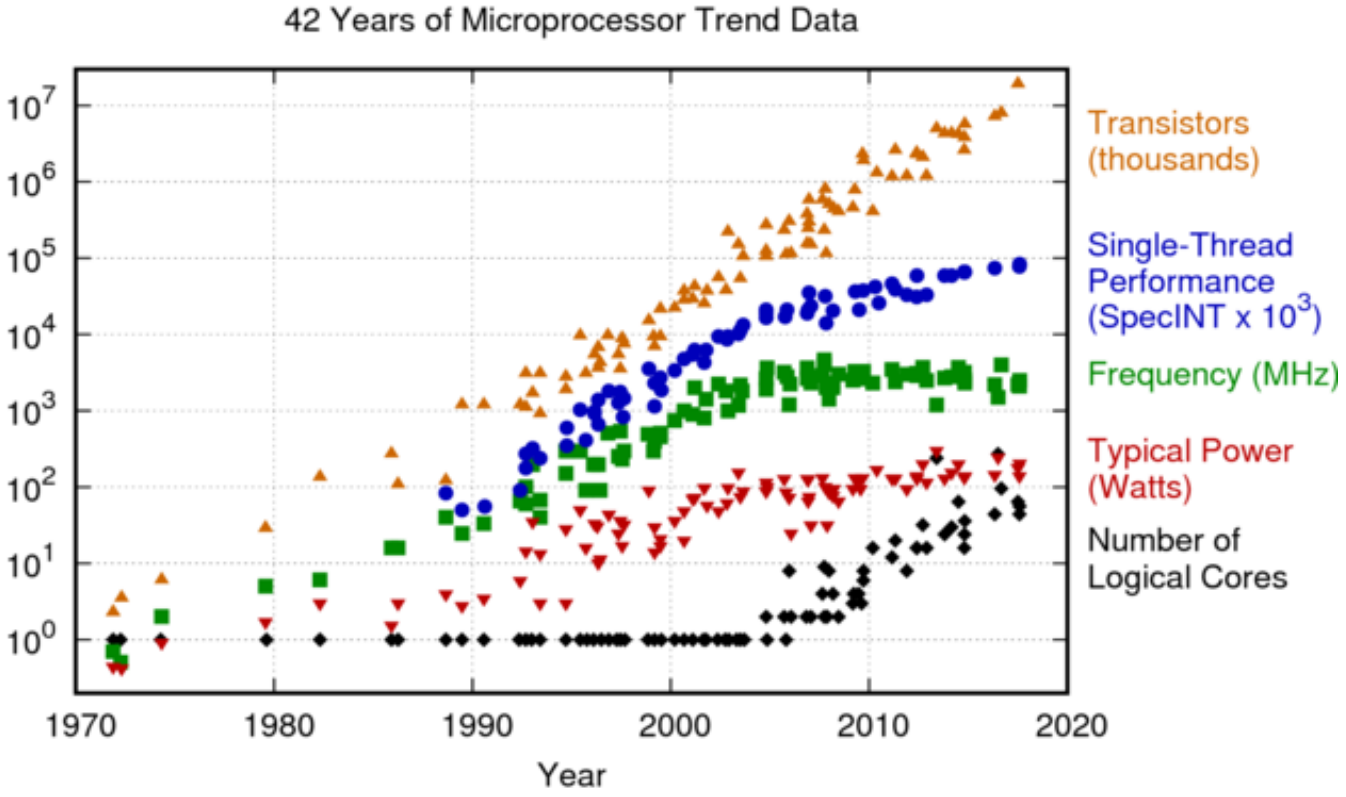
- The software industry has left a lot of programmers behind
- e.g., look at all the people who still write new code in *Visual Basic*
- Multiprogramming is still not easy
- But even with all the fancy toolchains/kits/guides, we're still programming like...

CPU Performance Wall



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2015 by K. Rupp

CPU Performance Wall



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp

We can't code this way anymore :-(

CPU Performance Wall

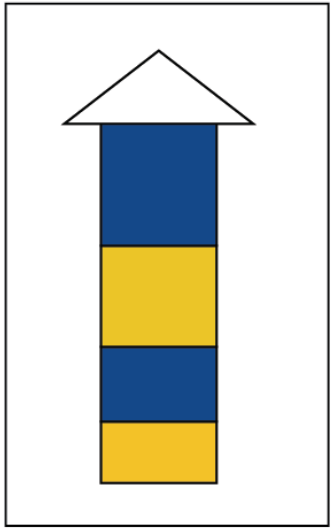
- We're constantly starved for performance
- We ran to cloud computing, which makes sense
  - But depreciating waste with on-prem now means *continuous waste* with cloud
  - *psst, that's also stopping on-prem/cloud transitions!*
- We depended on massive leaps in chip performance to see us through, now that's gone
- Massive, over-complicated APIs now have nowhere to hide

Concurrency

- An app that deals with more than one thing at once
- Maximize the hardware you're paying for
- Being efficient at waiting

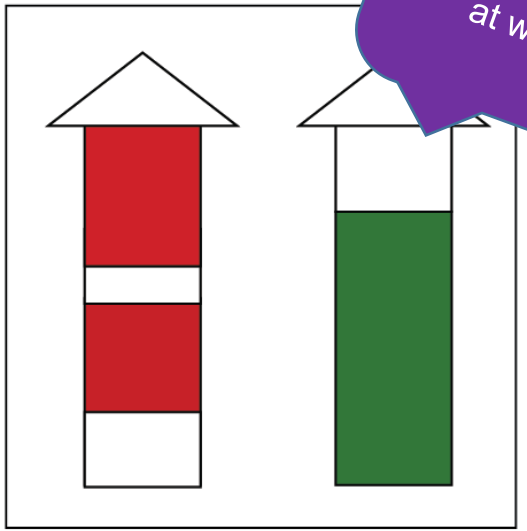
Concurrency vs Parallelism

Concurrency



Concurrency is about *dealing with* lots of things at once

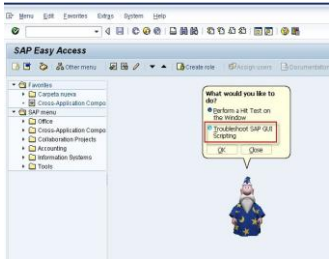
Parallelism



Parallelism is about *doing* lots of things at once

Concurrency is also about how *efficient* you are at waiting

No Concurrency



LongOperation()

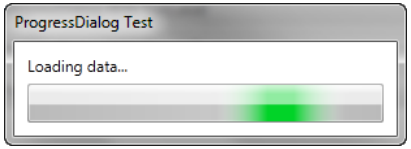
Receive Input	Long Running Operation 1	Long Running Operation 2	Return Value

Time

Concurrency (Async/Await in C#)

MakeLongRequest(LongOperation)

OPComplete(data)



data = await LongOperation()

LongOperation()

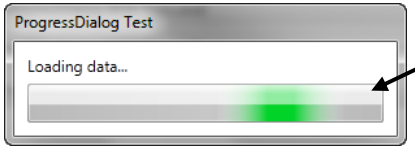
Receive Input	Long Running Operation 1	Long Running Operation 2	Return Value

Time

Concurrency (Async/Await in C#)

MakeLongRequest(LongOperation)

OPComplete(data)



ProgressBar1.BeginInvoke({})...

data = await LongOperation()

LongOperation()

Time

SetProgressThing(15)

SetProgressThing(75)

Remember I said multiprogramming is still not easy?

- Most computer languages *introduced* concurrency as a feature, it wasn't a major focus of design
- You may have to pull in a kit to work with advanced multiprogramming (Python, Java, C, .NET etc.)
- High-level programmers must be very aware of what the processor is doing as well as the SDK, else race-conditions, deadlocks and memory leaks emerge
- Oh yeah, that stuff is hard to debug, too...

Golang

Golang (or, *Go*)

- Developed at Google in 2009-2012
- Used extensively inside Google
- Makes concurrent programming *much, much easier*
- Instead of threads, there's go routines, which are far more efficient than launching threads

Golang (or, *Go*)

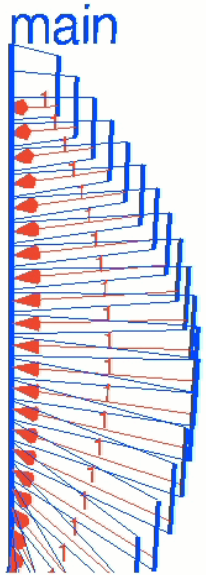
- It mimics C

Golang (or, *Go*)

- It mimics C
- It's not difficult to use (unlike C)
- Strong typing, memory-safety, memory managed for you with garbage collection (also unlike C)
- The types are modern (*really* unlike C)
- Async operations are lightweight (unlike Java/C#)
- Async code is readable (unlike... everything)

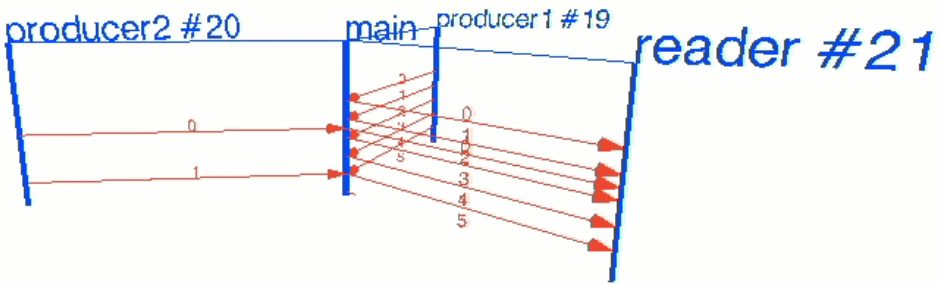
Concurrency Patterns

The Timer Pattern




```go
package main

import "time"

func timer(d time.Duration) <-chan int {
    c := make(chan int)
    go func() {
        time.Sleep(d)
        c <- 1
    }()
    return c
}

func main() {
    for i := 0; i < 24; i++ {
        c := timer(1 * time.Second)
        <-c
    }
}
```

The Fan-In Pattern

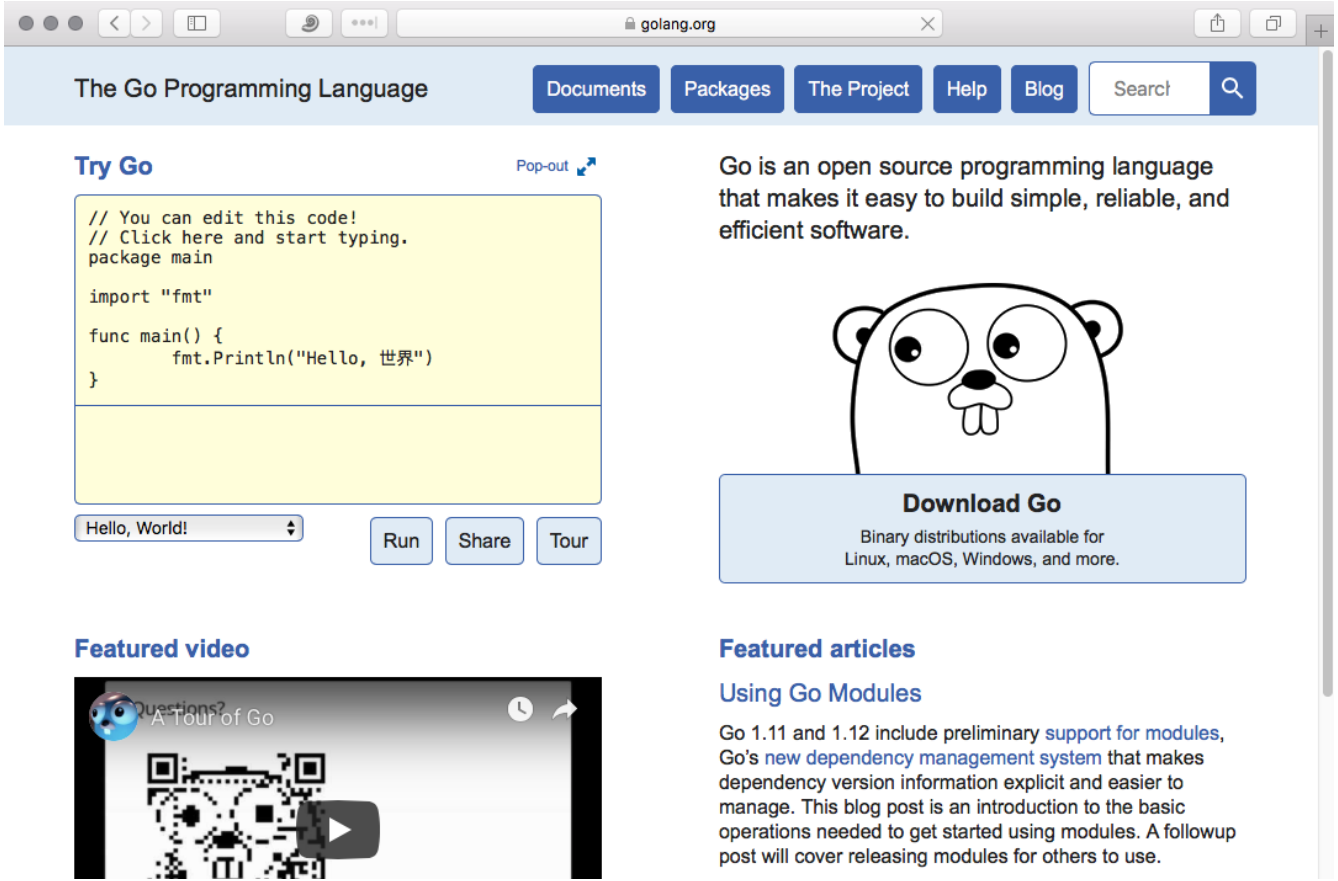


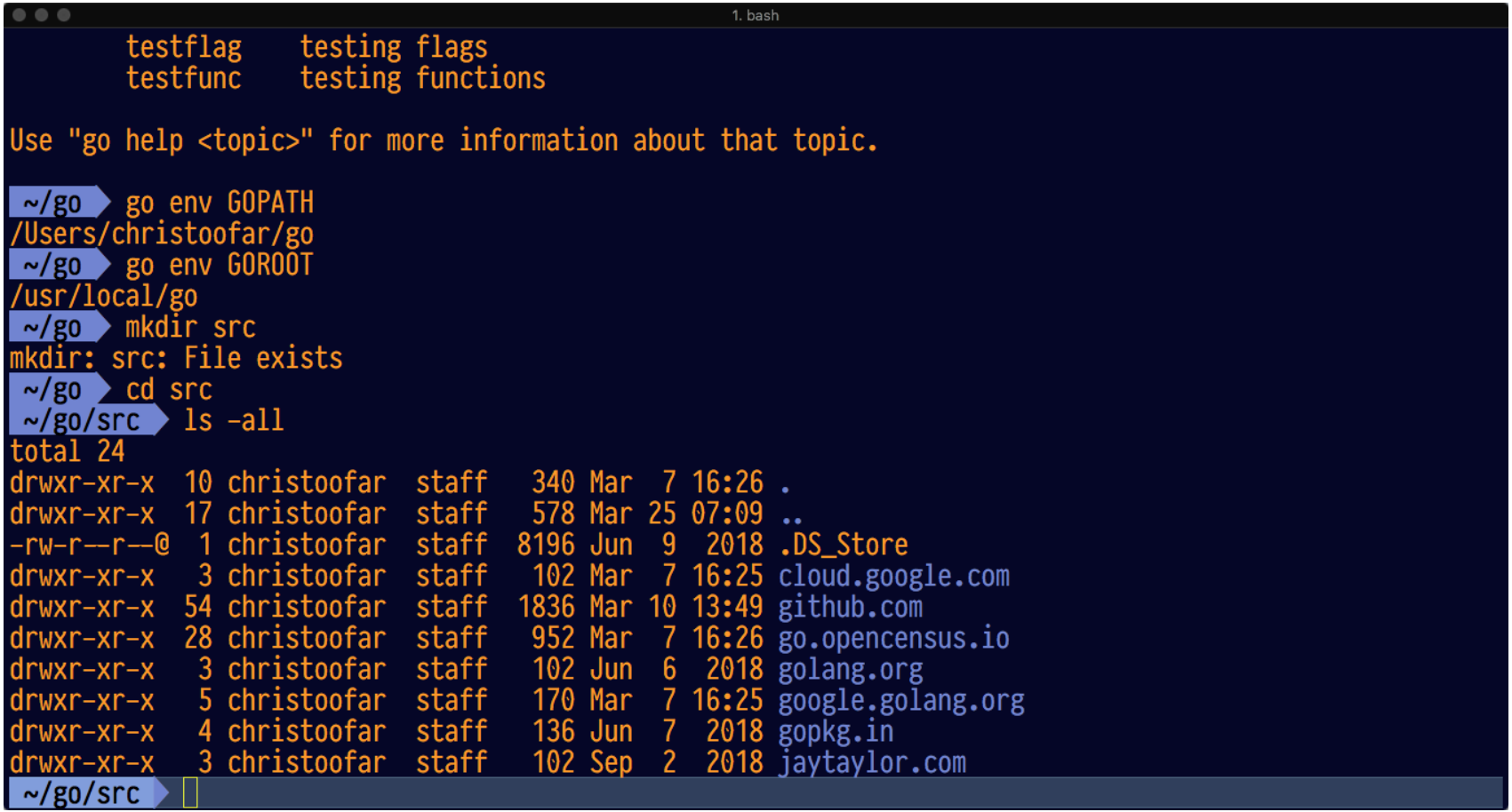

```go
package main

import (
    "fmt"
    "time"
)

func producer(ch chan int, d time.Duration) {
    var i int
    for {
        ch <- i
        i++
        time.Sleep(d)
    }
}

func reader(out chan int) {
    for x := range out {
        fmt.Println(x)
    }
}

func main() {
    ch := make(chan int)
    out := make(chan int)
    go producer(ch, 100*time.Millisecond)
    go producer(ch, 250*time.Millisecond)
    go reader(out)
    for i := range ch {
        out <- i
    }
}
```

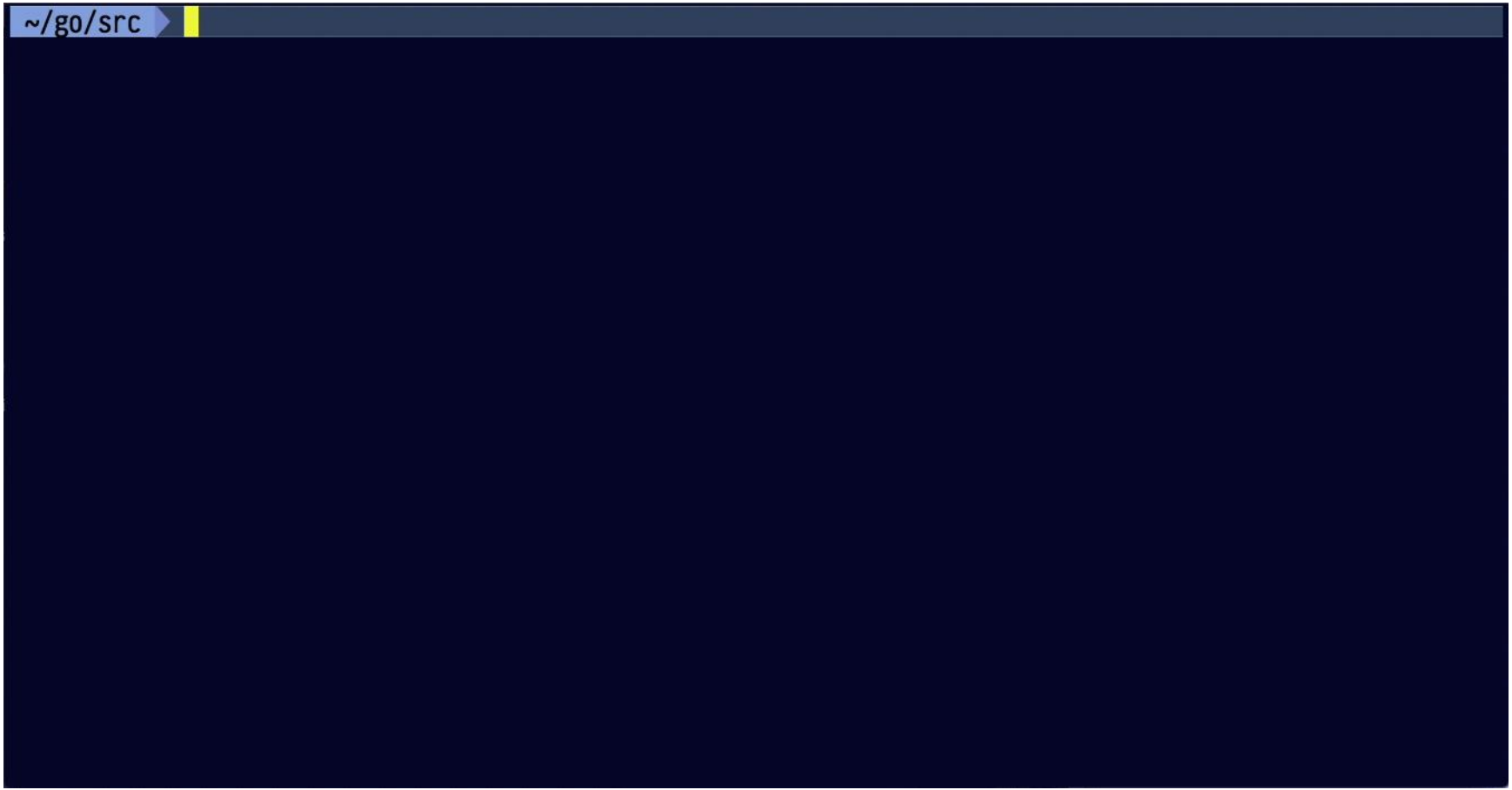
The Worker (Fan Out) Pattern

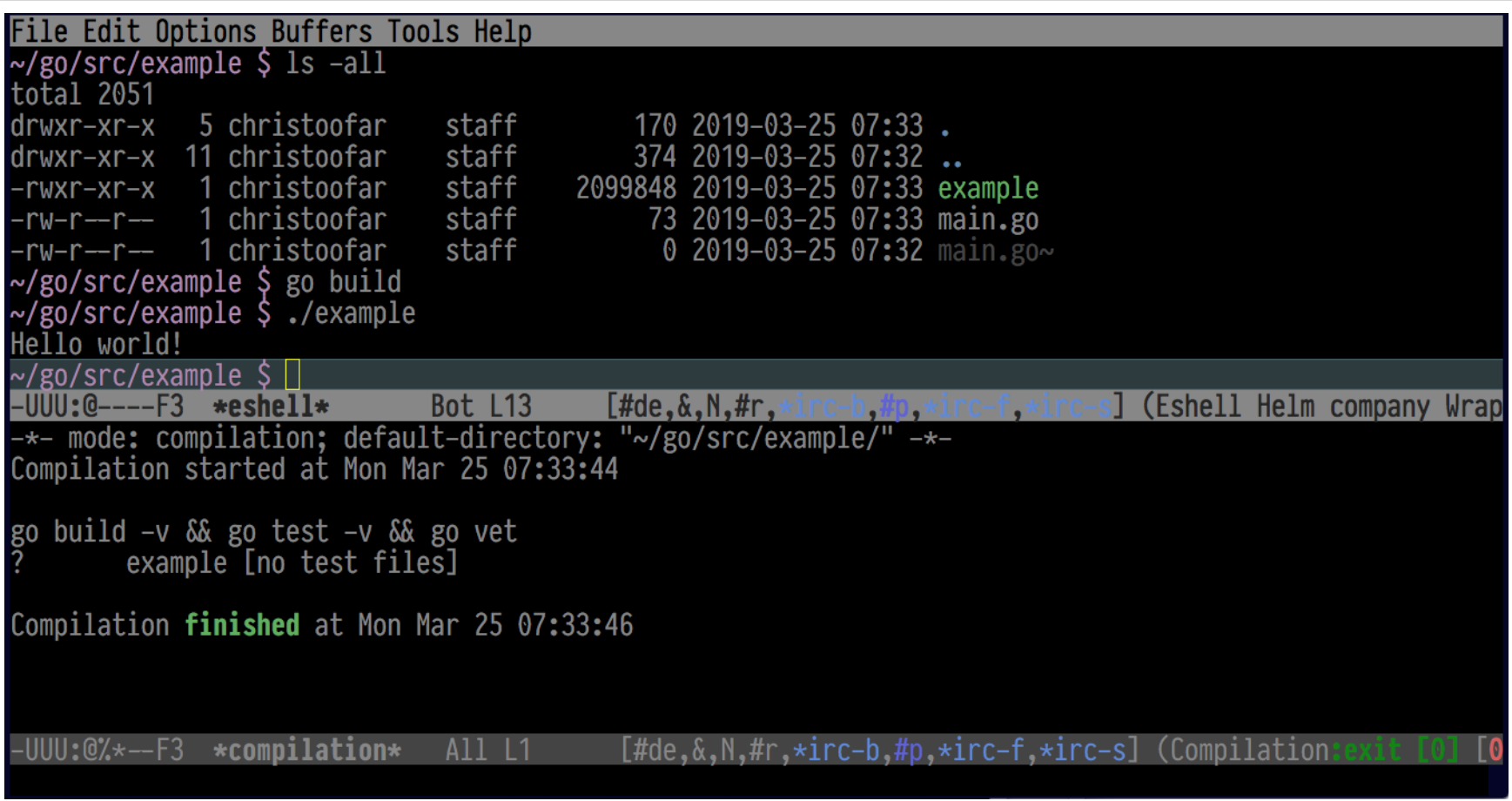

```go
package main

import (
    "fmt"
    "sync"
    "time"
)

func worker(tasksCh <-chan int, wg *sync.WaitGroup) {
    defer wg.Done()
    for {
        task, ok := <-tasksCh
        if !ok {
            return
        }
        d := time.Duration(task) * time.Millisecond
        time.Sleep(d)
        fmt.Println("processing task", task)
    }
}

func pool(wg *sync.WaitGroup, workers, tasks int) {
    tasksCh := make(chan int)

    for i := 0; i < workers; i++ {
        go worker(tasksCh, wg)
    }

    for i := 0; i < tasks; i++ {
        tasksCh <- i
    }

    close(tasksCh)
}

func main() {
    var wg sync.WaitGroup
    wg.Add(36)
    go pool(&wg, 36, 50)
    wg.Wait()
}
```

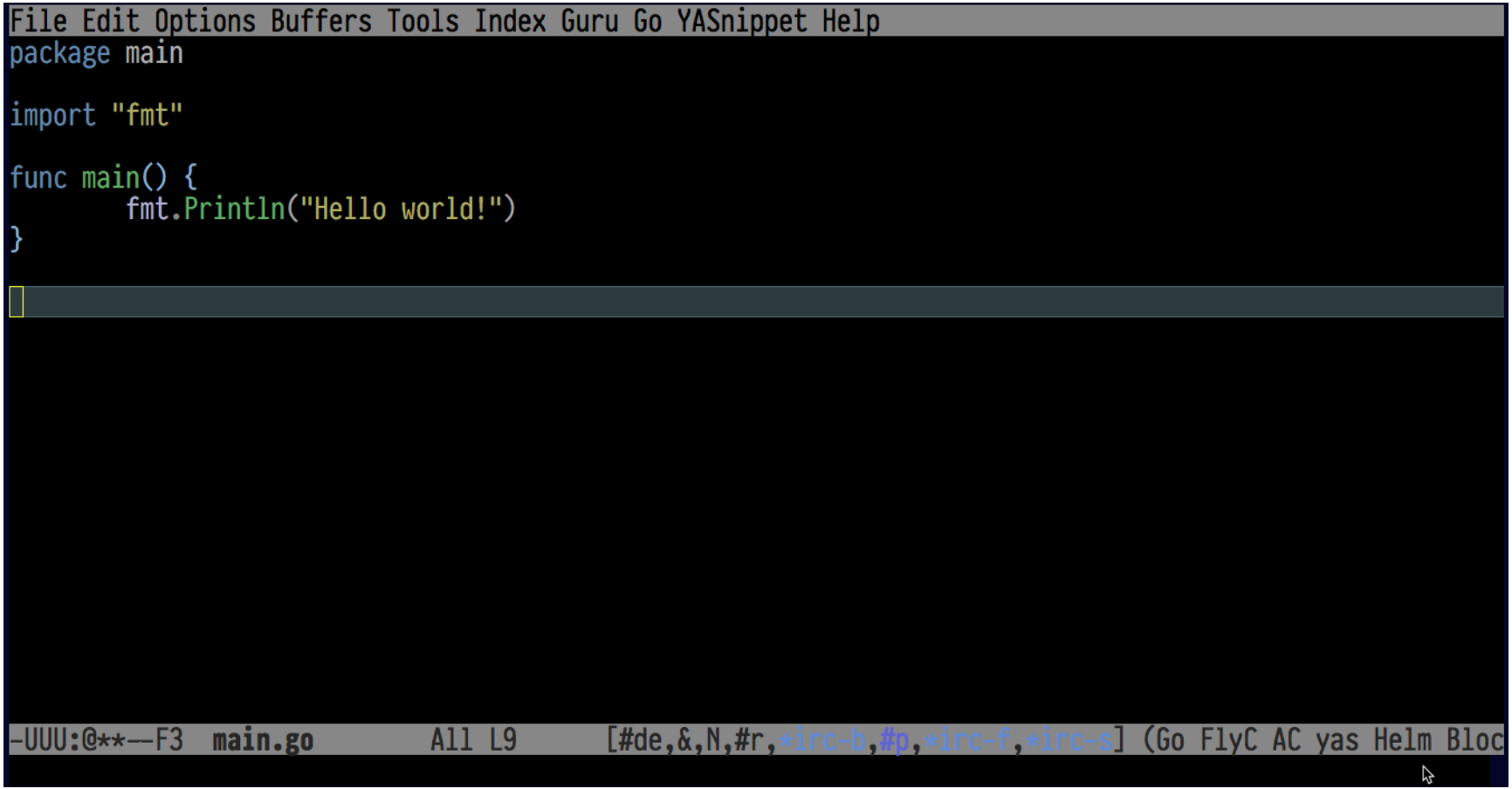
Let's Learn Go in 20 Minutes


golang.org
The Go Programming Language
Documents
Packages
The Project
Help
Blog
Search
Try Go
Pop-out
// You can edit this code!
// Click here and start typing.
package main
import "fmt"
func main() {
fmt.Println("Hello, 世界")
}
Hello, World!
Run
Share
Tour
Go is an open source programming language that makes it easy to build simple, reliable, and efficient software.
Download Go
Binary distributions available for Linux, macOS, Windows, and more.
Featured video
A Tour of Go
Featured articles
Using Go Modules
Go 1.11 and 1.12 include preliminary support for modules, Go's new dependency management system that makes dependency version information explicit and easier to manage. This blog post is an introduction to the basic operations needed to get started using modules. A followup post will cover releasing modules for others to use.


1. bash
testflag testing flags
testfunc testing functions
Use "go help <topic>" for more information about that topic.
~/go go env GOPATH
/Users/christoofar/go
~/go go env GOROOT
/usr/local/go
~/go mkdir src
mkdir: src: File exists
~/go cd src
~/go/src ls -all
total 24
drwxr-xr-x 10 christoofar staff 340 Mar 7 16:26 .
drwxr-xr-x 17 christoofar staff 578 Mar 25 07:09 ..
-rw-r--r--@ 1 christoofar staff 8196 Jun 9 2018 .DS_Store
drwxr-xr-x 3 christoofar staff 102 Mar 7 16:25 cloud.google.com
drwxr-xr-x 54 christoofar staff 1836 Mar 10 13:49 github.com
drwxr-xr-x 28 christoofar staff 952 Mar 7 16:26 go.opencensus.io
drwxr-xr-x 3 christoofar staff 102 Jun 6 2018 golang.org
drwxr-xr-x 5 christoofar staff 170 Mar 7 16:25 google.golang.org
drwxr-xr-x 4 christoofar staff 136 Jun 7 2018 gopkg.in
drwxr-xr-x 3 christoofar staff 102 Sep 2 2018 jaytaylor.com
~/go/src


~/go/src


File Edit Options Buffers Tools Help
~/go/src/example $ ls -all
total 2051
drwxr-xr-x   5 christoofar   staff   170 2019-03-25 07:33 .
drwxr-xr-x 11 christoofar   staff   374 2019-03-25 07:32 ..
-rwxr-xr-x   1 christoofar   staff   2099848 2019-03-25 07:33 example
-rw-r--r--   1 christoofar   staff   73 2019-03-25 07:33 main.go
-rw-r--r--   1 christoofar   staff   0 2019-03-25 07:32 main.go~
~/go/src/example $ go build
~/go/src/example $ ./example
Hello world!
~/go/src/example $
-UUU:@----F3 *eshell*   Bot L13   [#de,&,N,#r,*irc-b,#p,*irc-f,*irc-s] (Eshell Helm company Wrap
-*- mode: compilation; default-directory: "~/go/src/example/" -*-
Compilation started at Mon Mar 25 07:33:44
go build -v && go test -v && go vet
?   example [no test files]
Compilation finished at Mon Mar 25 07:33:46
-UUU:@%*--F3 *compilation*   All L1   [#de,&,N,#r,*irc-b,#p,*irc-f,*irc-s] (Compilation:exit [0] [0


File Edit Options Buffers Tools Index Guru Go YASnippet Help
package main
import "fmt"
func main() {
fmt.Println("Hello world!")
}
-UUU:@**--F3 main.go All L9 [#de,&,N,#r,*irc-b,#p,*irc-f,*irc-s] (Go FlyC AC yas Helm Bloc

```
package main

import "fmt"

func main() {

}
```

```
-UUU:@**--F3  main.go        All L6      [#de,&,N,#r,*irc-b,#p,*irc-f,*irc-s] (Go FlyC AC yas Helm Bloc
No more Flycheck errors
```

```go
package main

import "fmt"

func main() {
        var x int

        x = 3

        fmt.Println(x)
}
```

```
-UUU:@----F3  main.go        All L10     [#de,&,N,#r,*irc-b,#p,*irc-f,*irc-s] (Go FlyC AC yas Helm Bloc
```

```
package main

import "fmt"

func main() {
        var x int = 3
        var y int = 5
        var z = x * y

        fmt.Println(z)
}
```

```
-UUU:@----F3  main.go        All L10     [&,N,#r,*irc-b,#p,*irc-f,*irc-s] (Go FlyC AC yas Helm Blocks c
```

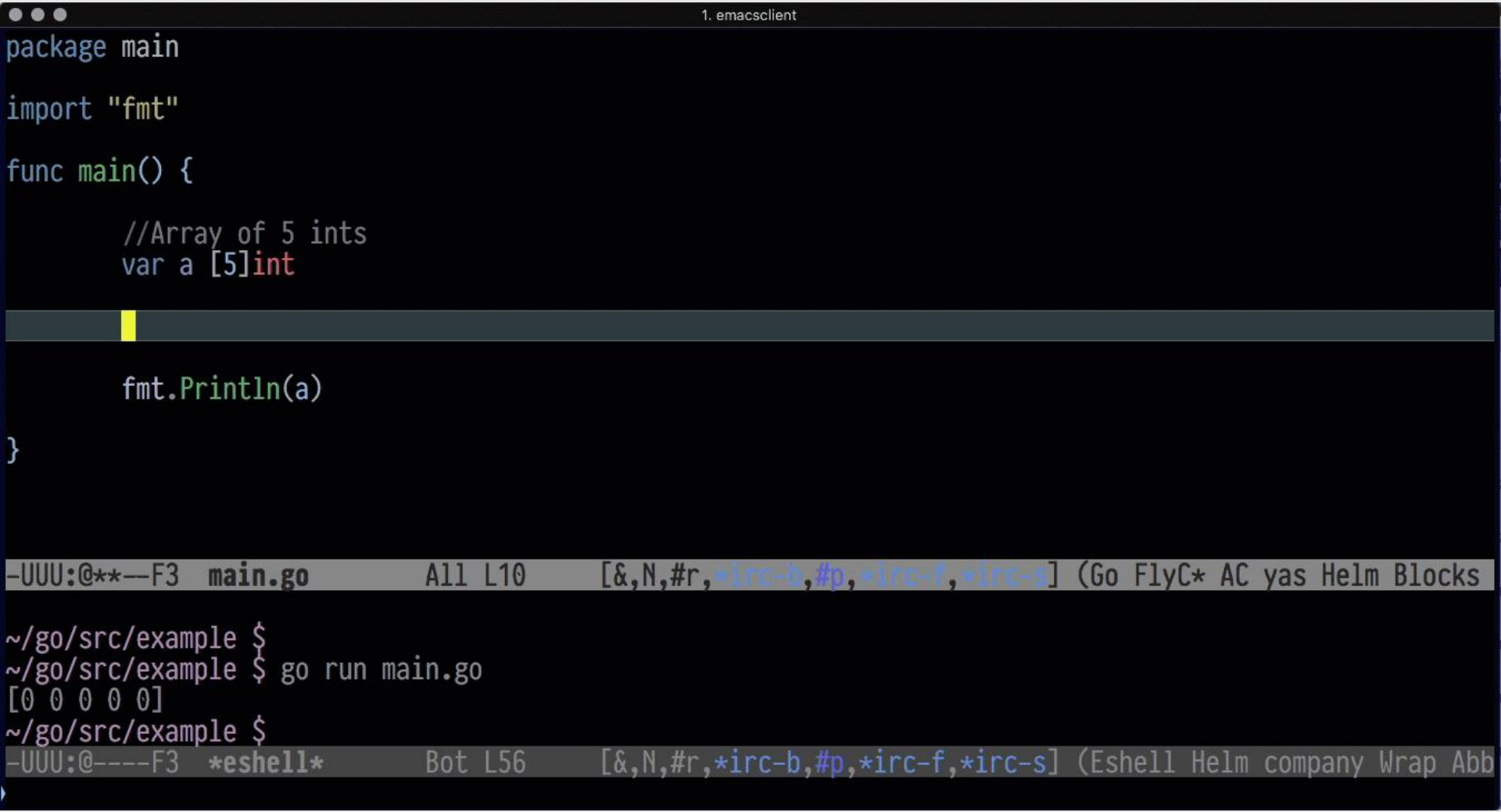
```
package main

import "fmt"

func main() {

        x := 7

}
```

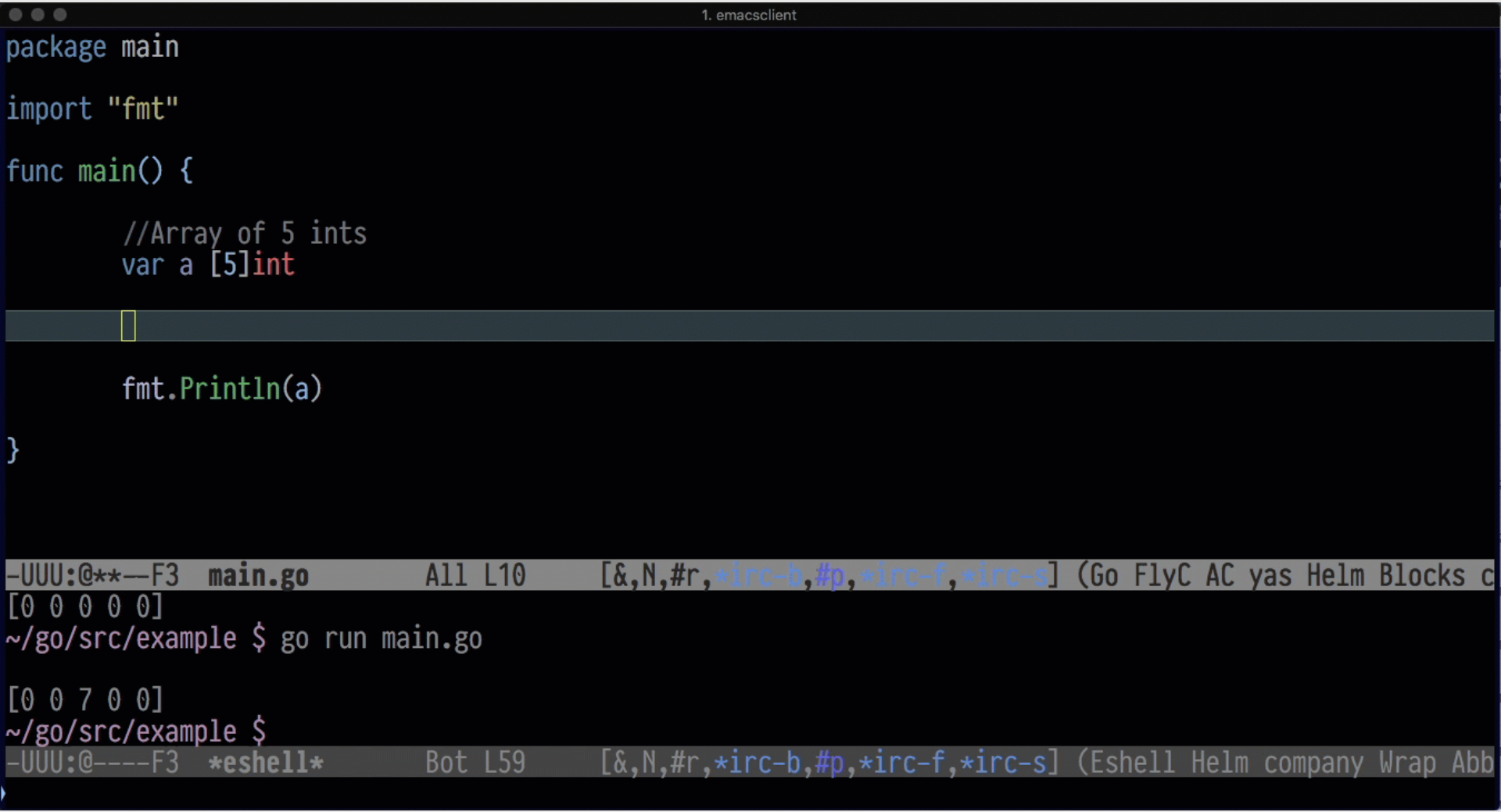

1. emacsclient
package main
import "fmt"
func main() {
x := 7
if x > 6 {
fmt.Println("More than 6!")
}
}
-UUU:@**--F3 main.go All L11 [&,N,#r,*irc-b,#p,*irc-f,*irc-s] (Go FlyC AC yas Helm Blocks c

```
package main

import "fmt"

func main() {

        //Array of 5 ints

}
```

```
-UUU:@**--F3  main.go        All L8      [&,N,#r,*irc-b,#p,*irc-f,*irc-s] (Go FlyC* AC yas Helm Blocks
```

```
~/go/src/example $
~/go/src/example $
-UUU:@----F3  *eshell*       Bot L54     [&,N,#r,*irc-b,#p,*irc-f,*irc-s] (Eshell Helm company Wrap Abb
```


1. emacsclient
package main
import "fmt"
func main() {
//Array of 5 ints
var a [5]int
fmt.Println(a)
}
-UUU:@**--F3 main.go All L10 [&,N,#r,*irc-b,#p,*irc-f,*irc-s] (Go FlyC* AC yas Helm Blocks
~/go/src/example $
~/go/src/example $ go run main.go
[0 0 0 0 0]
~/go/src/example $
-UUU:@----F3 *eshell* Bot L56 [&,N,#r,*irc-b,#p,*irc-f,*irc-s] (Eshell Helm company Wrap Abb


1. emacsclient
package main
import "fmt"
func main() {
//Array of 5 ints
var a [5]int
fmt.Println(a)
}
-UUU:@**--F3 main.go All L10 [&,N,#r,*irc-b,#p,*irc-f,*irc-s] (Go FlyC AC yas Helm Blocks c
[0 0 0 0 0]
~/go/src/example $ go run main.go
[0 0 7 0 0]
~/go/src/example $
-UUU:@----F3 *eshell* Bot L59 [&,N,#r,*irc-b,#p,*irc-f,*irc-s] (Eshell Helm company Wrap Abb


1. emacsclient
package main
import "fmt"
func main() {
// ↓ This is now a slice!
a := []int{5, 4, 3, 2, 1}
fmt.Println(a)
}
-UUU:@**--F3 main.go All L7 [&,N,#r,*irc-b,#p,*irc-f,*irc-s] (Go FlyC AC yas Helm Blocks c
[5 4 3 2 1]
~/go/src/example $ go run main.go
[5 4 3 2 1]
~/go/src/example $
-UUU:@----F3 *eshell* Bot L63 [&,N,#r,*irc-b,#p,*irc-f,*irc-s] (Eshell Helm company Wrap Abb

```
package main

import "fmt"

func main() {

        //    ↓ This is now a slice!
        a := []int{5, 4, 3, 2, 1}

        fmt.Println(a)

}
```

```
-UUU:@**--F3  main.go        All L9      [&,N,#r,*irc-b,#p,*irc-f,*irc-s] (Go FlyC AC yas Helm Blocks c
```

```
~/go/src/example $
~/go/src/example $
-UUU:@----F3  *eshell*       Bot L76     [&,N,#r,*irc-b,#p,*irc-f,*irc-s] (Eshell Helm company Wrap Abb
```


1. emacsclient
package main
import "fmt"
func main() {
}
-UUU:@**--F3 main.go All L7 [&,N,#r,*irc-b,#p,*irc-f,*irc-s] (Go FlyC AC yas Helm Blocks c
~/go/src/example $
~/go/src/example $ go run main.go
[5 4 3 2 1 999]
~/go/src/example $
-UUU:@----F3 *eshell* Bot L78 [&,N,#r,*irc-b,#p,*irc-f,*irc-s] (Eshell Helm company Wrap Abb

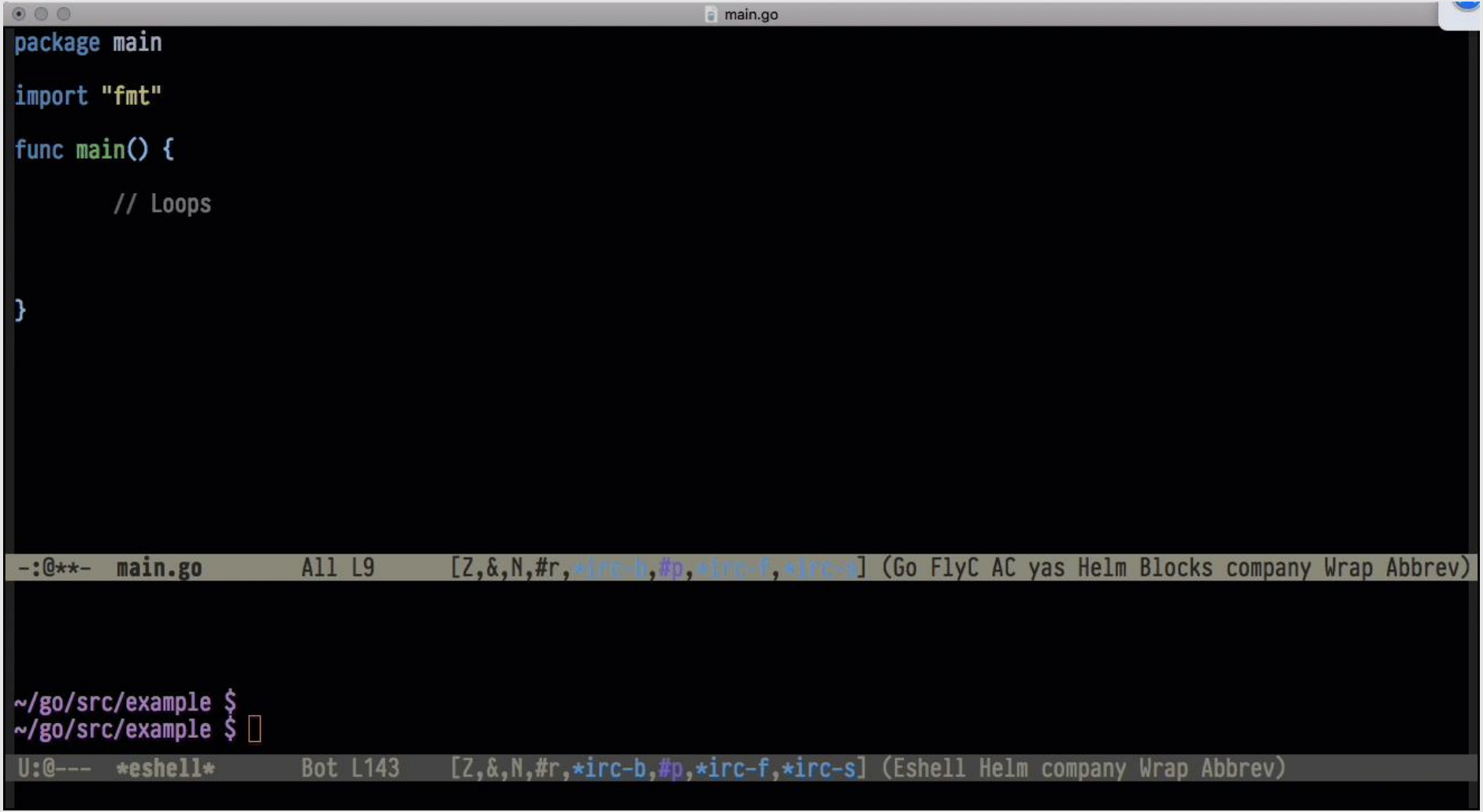
main.go

```go
package main

import "fmt"

func main() {

        vertices := make(map[string]int)

        vertices["triangle"] = 2
        vertices["square"] = 3
        vertices["dodecagon"] = 12

        fmt.Println(vertices)

}
```

-:@--- main.go   All L13   [&,N,#r,*irc-b,#p,*irc-f,*irc-s] (Go FlyC AC yas Helm Blocks company Wrap Abbrev)

```
~/go/src/example $
~/go/src/example $ go run main.go
map[dodecagon:12 square:3 triangle:2]
~/go/src/example $
```

U:@--- *eshell*   Bot L101   [&,N,#r,*irc-b,#p,*irc-f,*irc-s] (Eshell Helm company Wrap Abbrev)


main.go
package main
import "fmt"
func main() {
// Loops
}
-:@**- main.go All L9 [Z,&,N,#r,*irc-b,#p,*irc-f,*irc-s] (Go FlyC AC yas Helm Blocks company Wrap Abbrev)
~/go/src/example $
~/go/src/example $
U:@--- *eshell* Bot L143 [Z,&,N,#r,*irc-b,#p,*irc-f,*irc-s] (Eshell Helm company Wrap Abbrev)

main.go

```
package main

import "fmt"

func main() {

        // Loops (the WHILE loop)

        for i := 0; i < 5; i++ {
                fmt.Println(i)
        }

}
```

-:@**- main.go   All L10   [Z,&,N,#r,*irc-b,#p,*irc-f,*irc-s] (Go FlyC AC yas Helm Blocks company Wrap Abbrev)

```
0
1
2
3
4
~/go/src/example $
```

U:@--- *eshell*   Bot L149   [Z,&,N,#r,*irc-b,#p,*irc-f,*irc-s] (Eshell Helm company Wrap Abbrev)

<nil> <H-mouse-1> is undefined

main.go

```
package main

import "fmt"

func main() {

        array := []string{"Christopher", "Olga", "Bob", "Nguyen"}


}
```

-:@**- main.go   All L8   [Z,&,N,#r,*irc-b,#p,*irc-f,*irc-s] (Go FlyC* AC yas Helm Blocks company Wrap Abbrev)

```
0
1
2
3
4
~/go/src/example $
```

U:@--- *eshell*   Bot L155   [Z,&,N,#r,*irc-b,#p,*irc-f,*irc-s] (Eshell Helm company Wrap Abbrev)

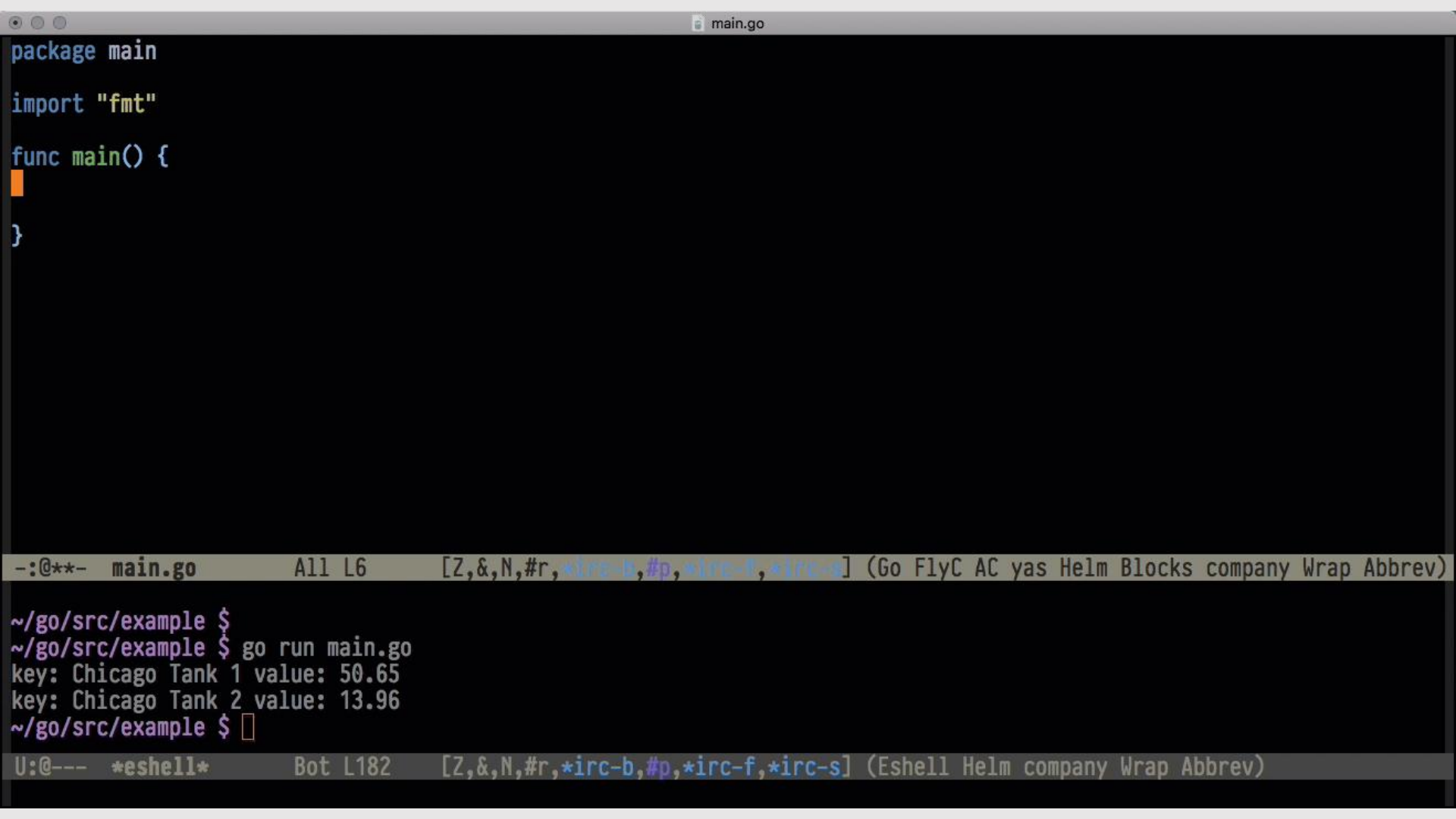
main.go

```
package main

import "fmt"

func main() {

        tankmap := make(map[string]float64)

        for index, value := range array {
                fmt.Println("index:", index, "value:", value)
        }

}
```

-:@**- main.go   All L8   [Z,&,N,#r,*irc-b,#p,*irc-f,*irc-s] (Go FlyC AC yas Helm Blocks company Wrap Abbrev)

```
~/go/src/example $
~/go/src/example $
```

U:@--- *eshell*   Bot L179   [Z,&,N,#r,*irc-b,#p,*irc-f,*irc-s] (Eshell Helm company Wrap Abbrev)


main.go
package main
import "fmt"
func main() {
}
-:@**- main.go All L6 [Z,&,N,#r,*irc-b,#p,*irc-f,*irc-s] (Go FlyC AC yas Helm Blocks company Wrap Abbrev)
~/go/src/example $
~/go/src/example $ go run main.go
key: Chicago Tank 1 value: 50.65
key: Chicago Tank 2 value: 13.96
~/go/src/example $
U:@--- *eshell* Bot L182 [Z,&,N,#r,*irc-b,#p,*irc-f,*irc-s] (Eshell Helm company Wrap Abbrev)

main.go

```
package main

import "fmt"

type tank struct {
        name     string
        capacity float64
}

func main() {

        myTank := tank{name: "Chicago Tank 1", capacity: 50.63}

        fmt.Println(myTank)

}
```

```
-:@---  main.go        All L14     [Z,&,N,#r,*irc-b,#p,*irc-f,*irc-s] (Go FlyC AC yas Helm Blocks company Wrap Abbrev)
~/go/src/example $ go run main.go
key: Chicago Tank 1 value: 50.65
key: Chicago Tank 2 value: 13.96
~/go/src/example $ go run main.go
{Chicago Tank 1 50.63}
~/go/src/example $
U:@---  *eshell*       Bot L184    [Z,&,N,#r,*irc-b,#p,*irc-f,*irc-s] (Eshell Helm company Wrap Abbrev)
```

Almost done! □

Control Flow

main.go

```
package main

import "fmt"

func main() {

}
```

-:@**- main.go All L6 [Z,&,N,#r,*irc-b,#p,*irc-f,*irc-s] (Go FlyC* AC yas Helm Blocks company Wrap Abbrev)

```
~/go/src/example $
~/go/src/example $
```

U:@--- *eshell* Bot L196 [Z,&,N,#r,*irc-b,#p,*irc-f,*irc-s] (Eshell Helm company Wrap Abbrev)

```
package main

import "fmt"

func main() {
        result := sum(2, 3)
        fmt.Println(sum)
}
```



```
~/go/src/example $
~/go/src/example $
```

```
package main

import (
        "errors"
        "fmt"
        "math"
)

func main() {
        result, err := sqrt(-6)
        if err ≠ nil {
                fmt.Println("We can't handle complex numbers 😉")
                return
        }

        fmt.Println(result)
}

func sqrt(x float64) (float64, error) {
        if x < 0 {
```



```
~/go/src/example $
~/go/src/example $ go run main.go
We can't handle complex numbers 😉
~/go/src/example $
```

Pointers

- I don’t have time to show that
- Java / .NET conditioned people to believe pointers are scary (Managed vs. Unmanaged)
- The Go compiler is so advanced; not only are pointers used commonly in Go, but with the garbage collector + async it’s difficult to screw up (*yet again*, unlike C/C++)

You’re now a Go programmer!

Go Routines

Go Routines

```
func main() {
        go doSomethingAsync()
}

func doSomethingAsync() {
        fmt.Println("Hi, I ran inside of a go routine!")
}
```

Go Routines

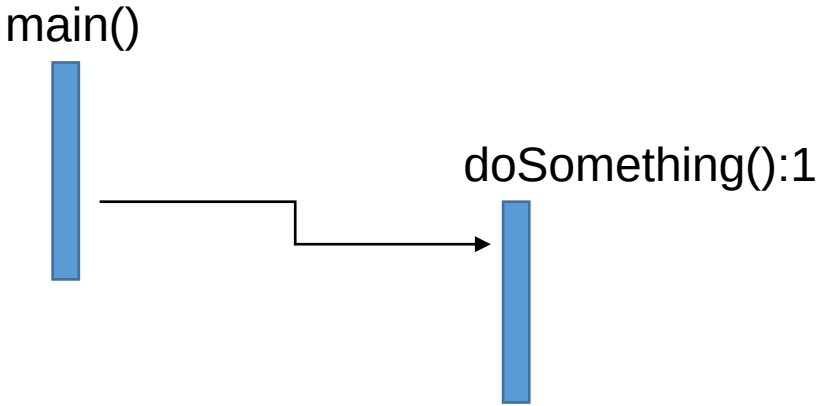
```
package main

import (
        "fmt"
)

func main() {
        go doSomethingAsync()
}

func doSomethingAsync() {
        fmt.Println("Hi, I ran inside of a go routine!")
}
```

```
-:@---  main.go        All L14     [#de,Z,&,N,#r,*irc-b,#p,*
```

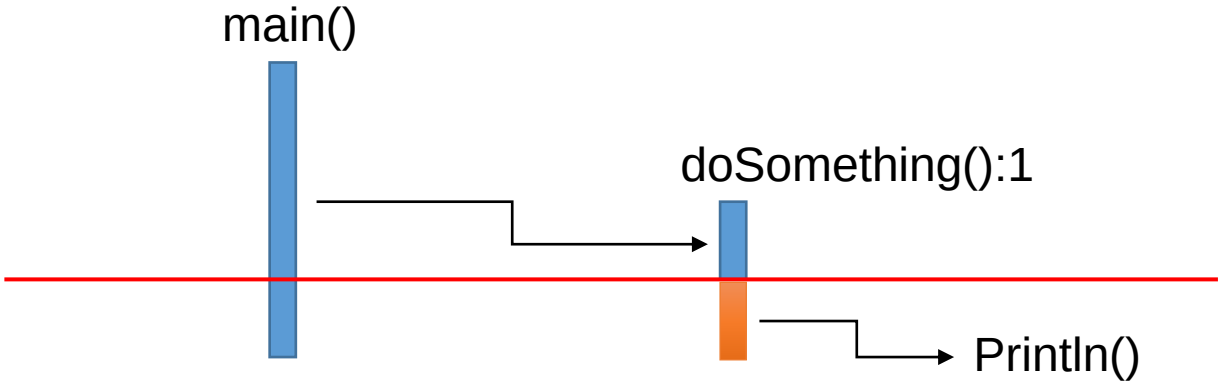
```
~/go/src/example $
~/go/src/example $ go run main.go
~/go/src/example $
```

```
U:@---  *eshell*       Bot L263    [#de,Z,&,N,#r,*irc-b,#p,*irc-f,*irc-s] (Eshell Helm company Wrap Abbrev)
```

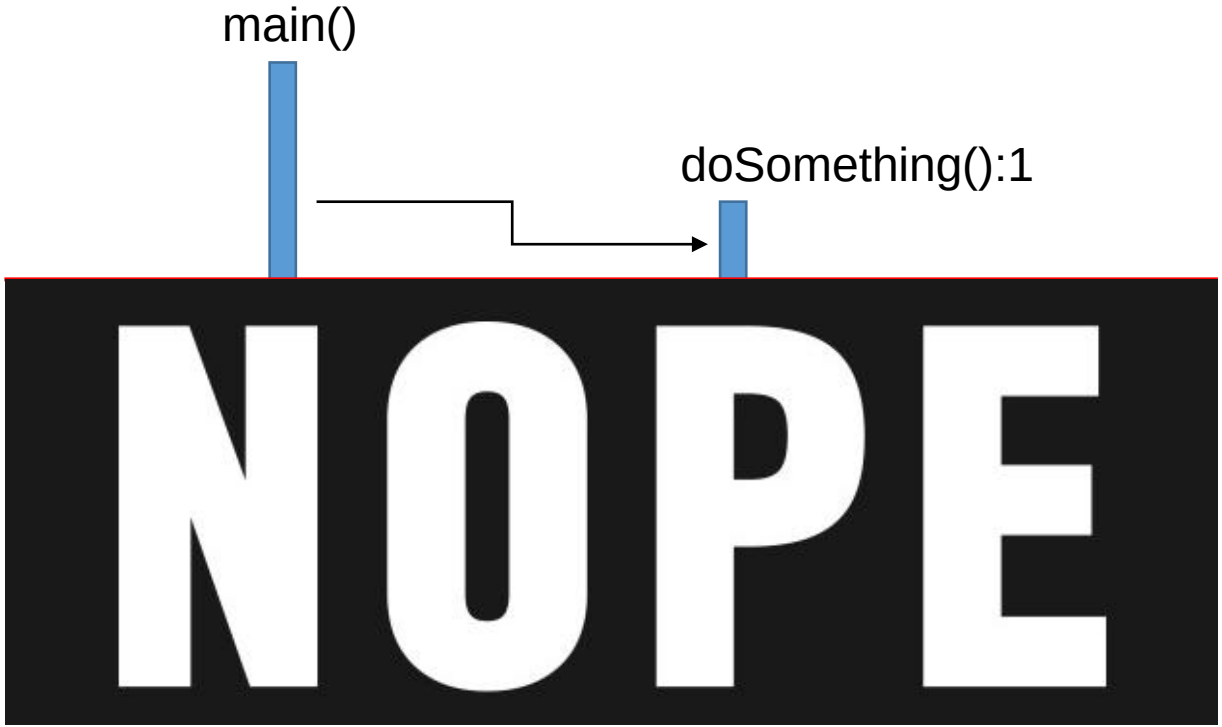
Go Routines



Go Routines



Go Routines




main.go
package main
import (
"fmt"
)
func main() {
go doSomethingAsync()
}
func doSomethingAsync() {
fmt.Println("Hi, I ran inside of a go routine!")
}
-:@--- main.go All L14 [#de,Z,&,N,#r,*irc-b,#p,*irc-f,*irc-s] (Go FlyC AC yas Helm Blocks company Wrap Abbrev)
~/go/src/example $
~/go/src/example $ go run main.go
~/go/src/example $
U:@--- *eshell* Bot L263 [#de,Z,&,N,#r,*irc-b,#p,*irc-f,*irc-s] (Eshell Helm company Wrap Abbrev)

Channels

They’re Just Pipes

Goroutine 1

Channel s

Goroutine 2


eshell
package main
import "fmt"
func main() {
}
-:@**- main.go All L10 [#de,Z,&,N,#r,*irc-b,#p,*irc-f,*irc-s] (Go FlyC AC yas Helm Blocks company Wrap Abbrev)
~/go/src/example $
~/go/src/example $
U:@--- *eshell* Bot L303 [#de,Z,&,N,#r,*irc-b,#p,*irc-f,*irc-s] (Eshell Helm company Wrap Abbrev)

OK, *NOW* you're a Go programmer!

Project: Building a massively-scaling web application server

- Oh god
- I don’t have time for this
- That’s massively complex

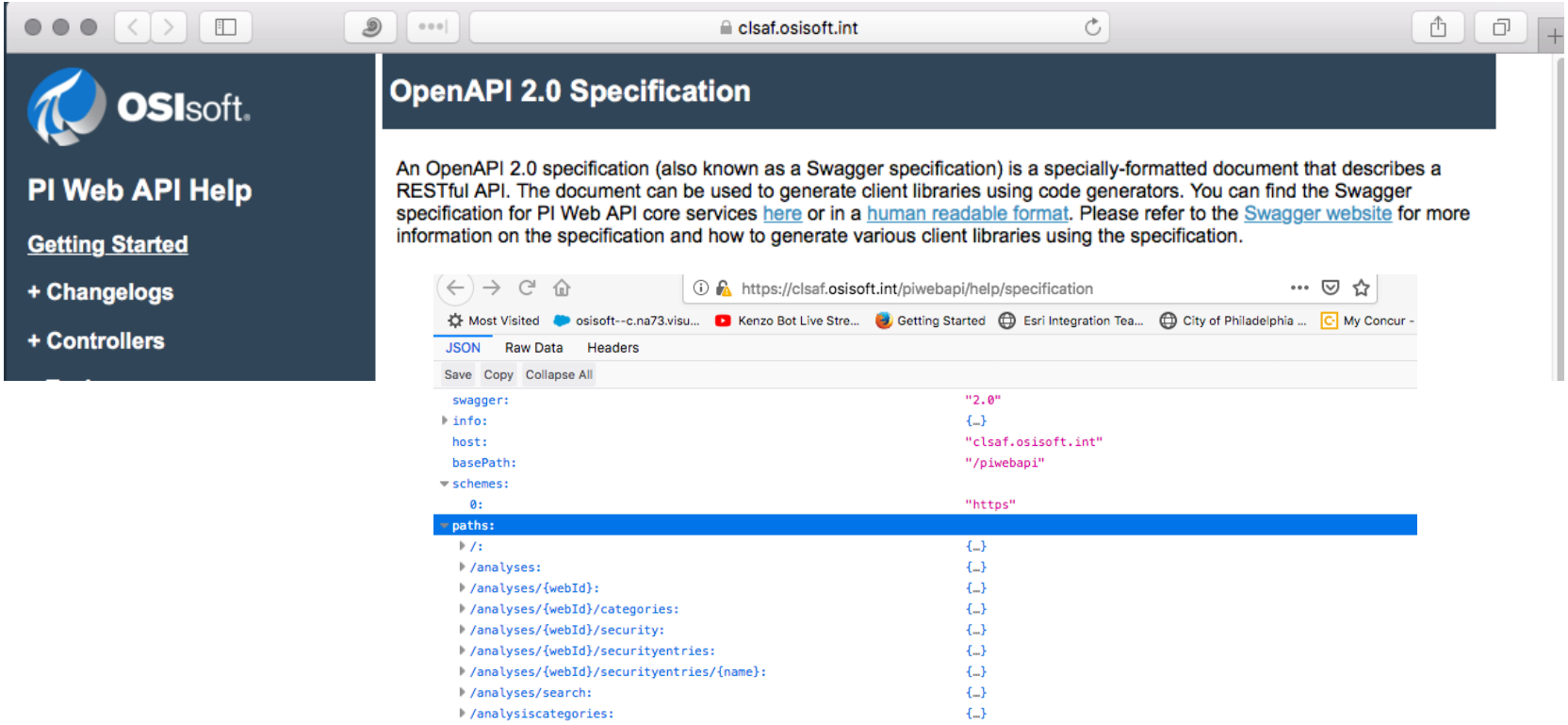
Project: Building a massively-scaling web application server

```
package main

import (
    "encoding/json"
    "log"
    "net/http"
    "github.com/gorilla/mux"
)

// our main function
func main() {
    router := mux.NewRouter()
    log.Fatal(http.ListenAndServe(":8000", router))
}
```

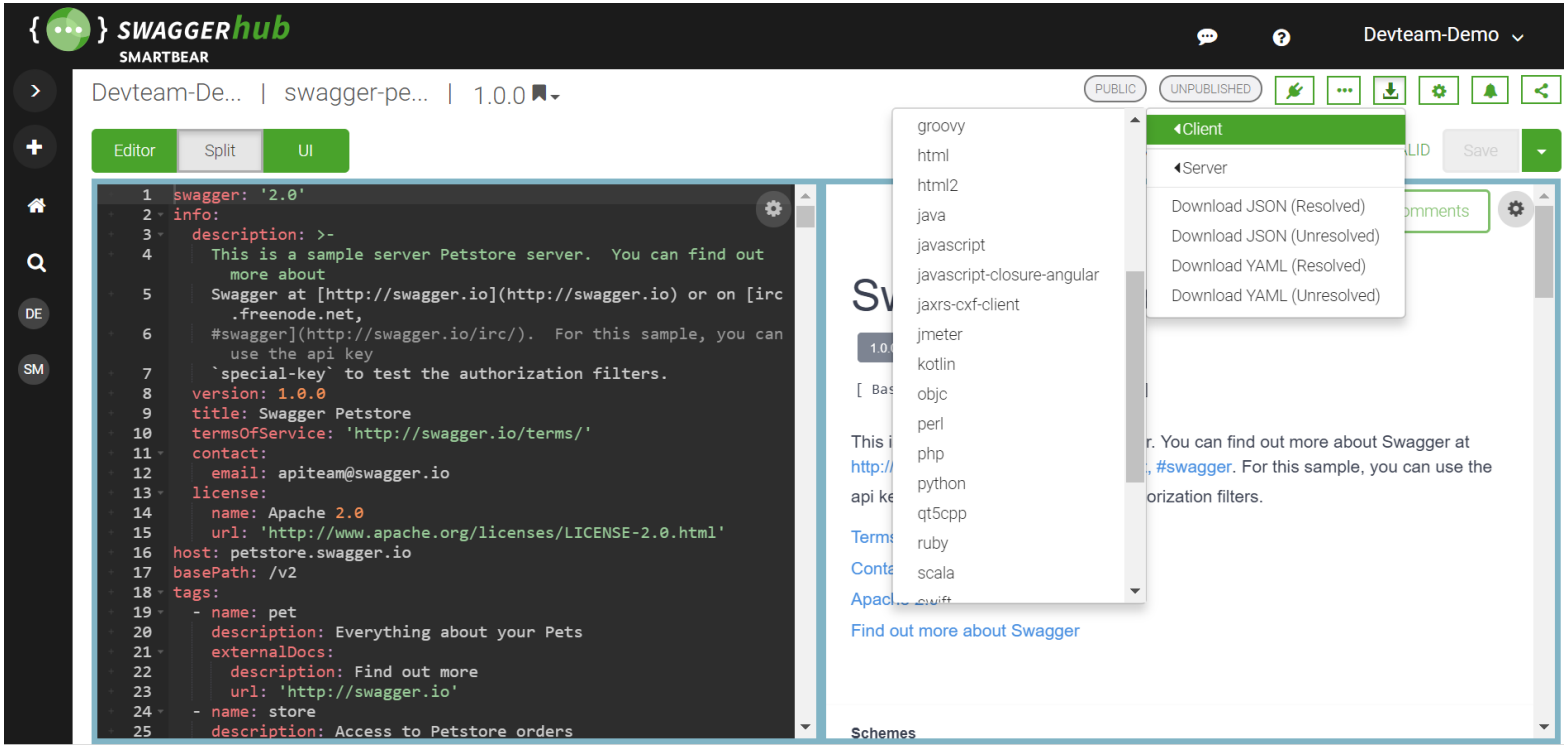
Project: Building a massively-scaling web application server

```
// fun main()
func main() {
    router := mux.NewRouter()
    router.HandleFunc("/people", GetPeople).Methods("GET")
    router.HandleFunc("/people/{id}", GetPerson).Methods("GET")
    router.HandleFunc("/people/{id}", CreatePerson).Methods("POST")
    router.HandleFunc("/people/{id}", DeletePerson).Methods("DELETE")
    log.Fatal(http.ListenAndServe(":8000", router))
}
```

Web API for Go?



Swagger.io



Project Mashup

Corporate API Libraries

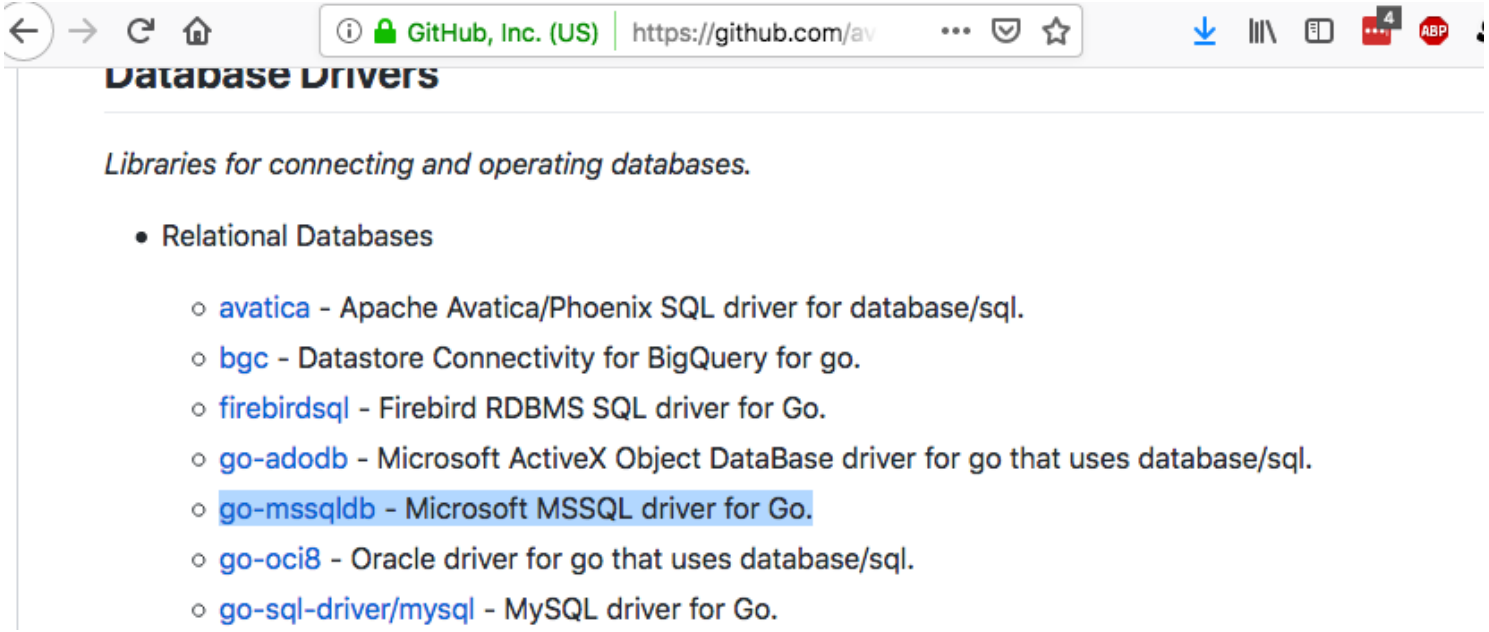
Unification Layer (golang)

Application Server (golang)

Project: Exploiting Concurrency In IoT

- Project CHEAP: lot of sensors chained to one concentrator
- EDGE Calc: do pre-analytics (moving averages) at the edge to reduce server loads
- MEGABUFF: Robust anti-packet-drop buffering with audible alarm on extended uplink loss

Go Awesome



Yes the concurrency is that good…

Why you can have millions of Goroutines but only thousands of Java Threads

Thanks to random internet strangers, you can also read this post in: Chinese Japanese

Many seasoned engineers working in JVM based languages have seen errors like this:

```
[error] (run-main-0) java.lang.OutOfMemoryError: unable to create native thread:
[error] java.lang.OutOfMemoryError: unable to create native thread:
[error]         at java.base/java.lang.Thread.start0(Native Method)
[error]         at java.base/java.lang.Thread.start(Thread.java:813)
...
[error]         at java.base/java.lang.Thread.run(Thread.java:844)
```

`OutOfMemory`…err…out of threads. On my laptop running Linux, this happens after a paltry 11500 threads.

Questions?

Please wait for the **microphone**

State your **name & company**

Please remember