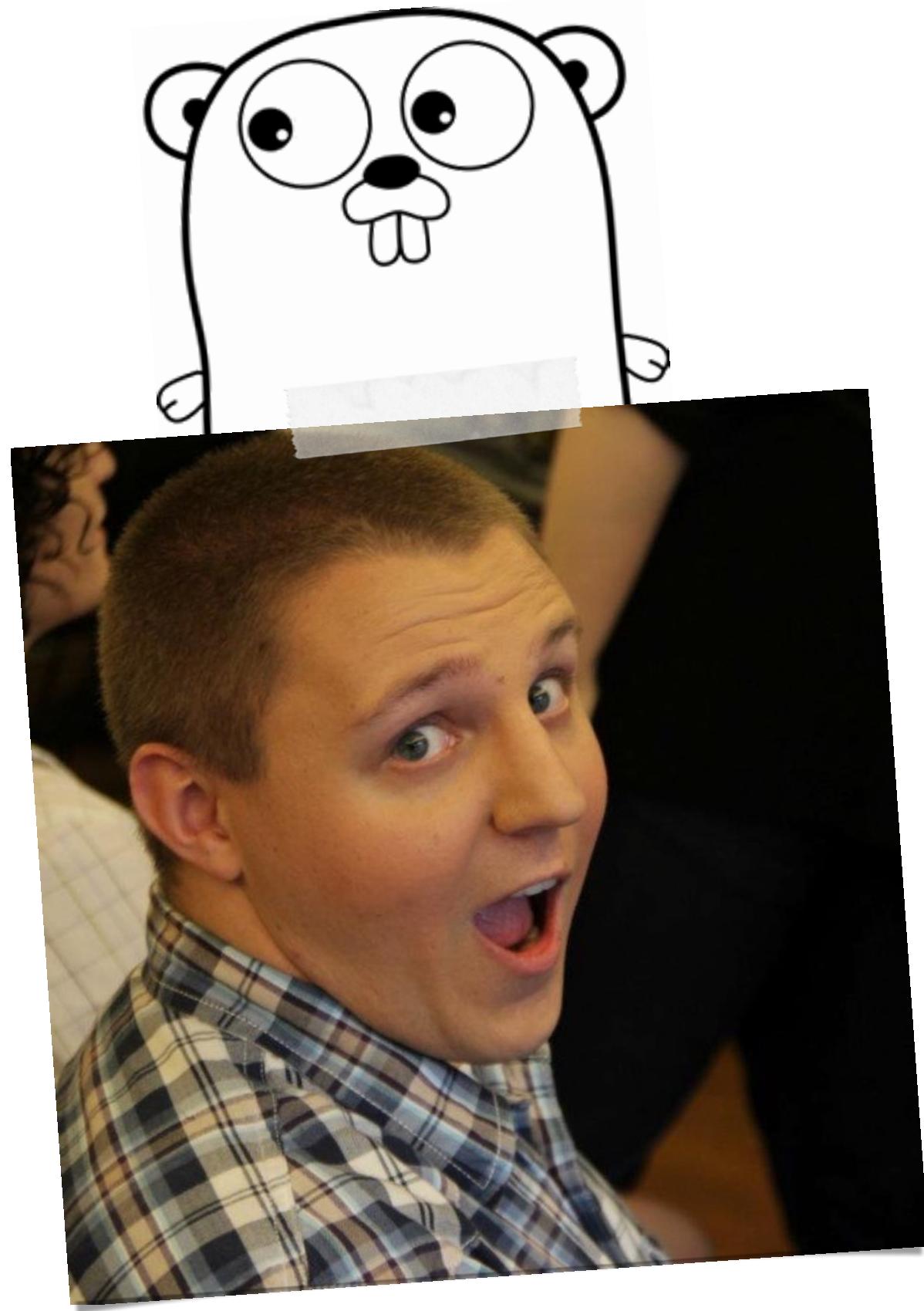
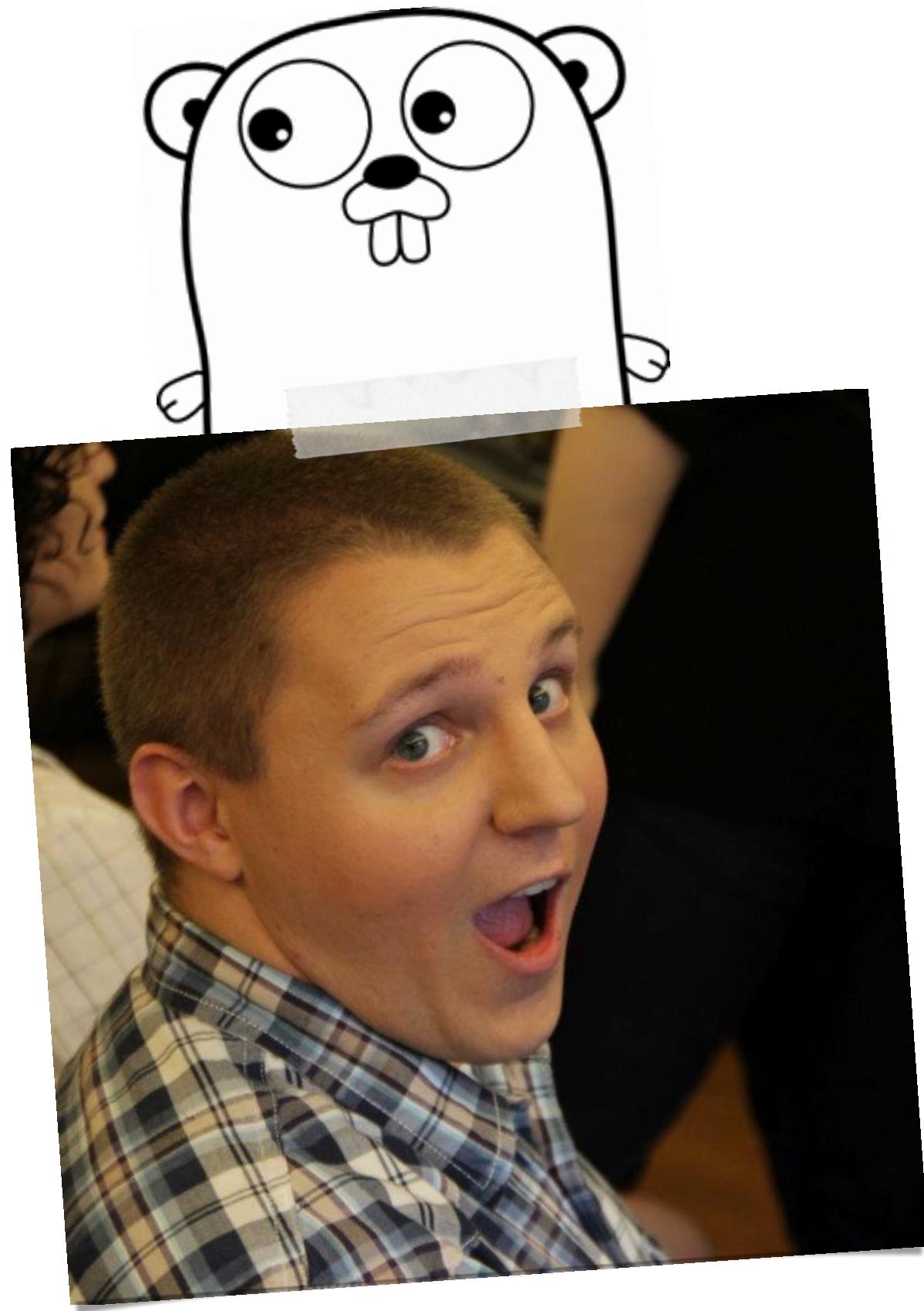


INTRO TO GO



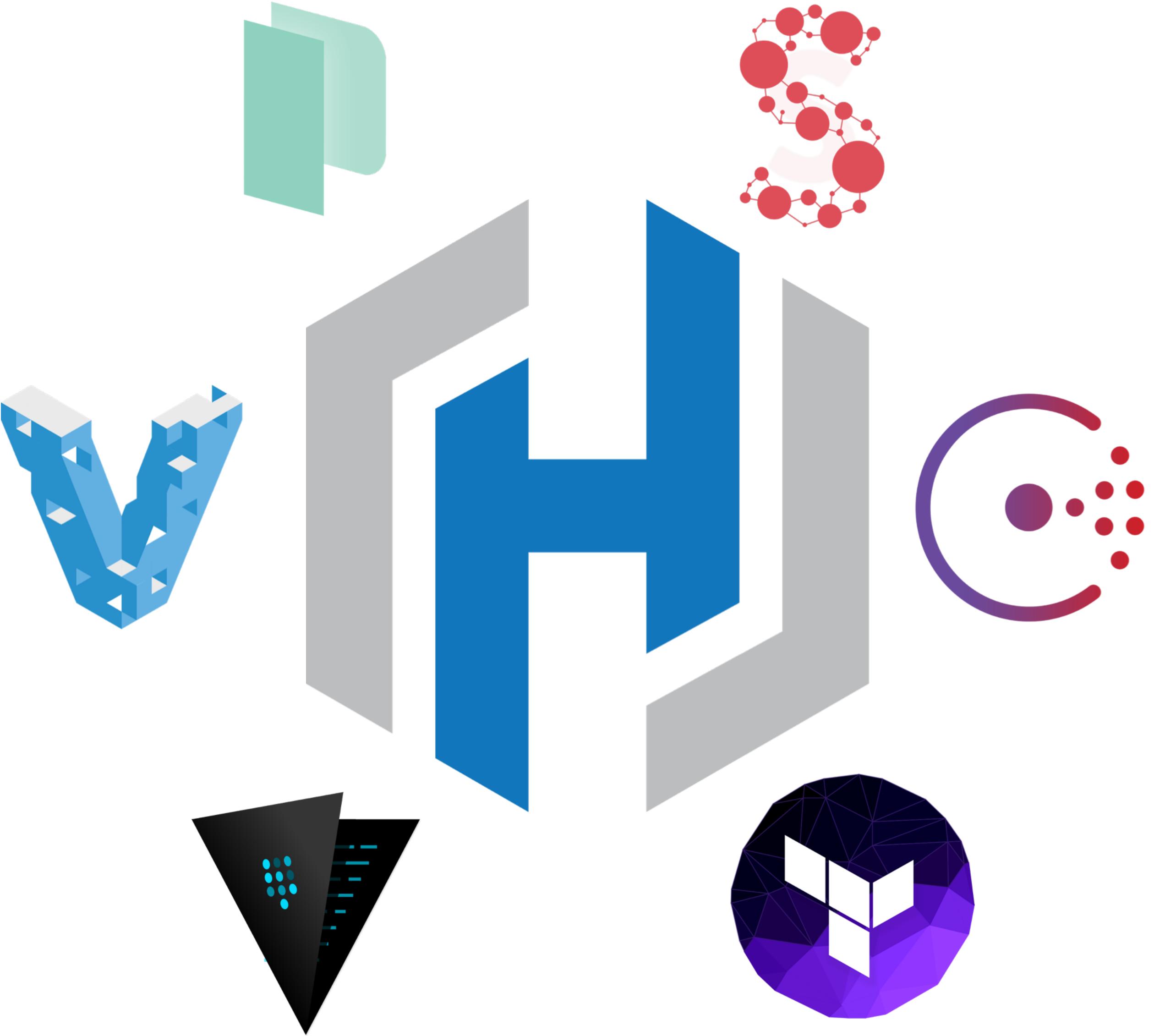
SETH VARGO
@sethvargo

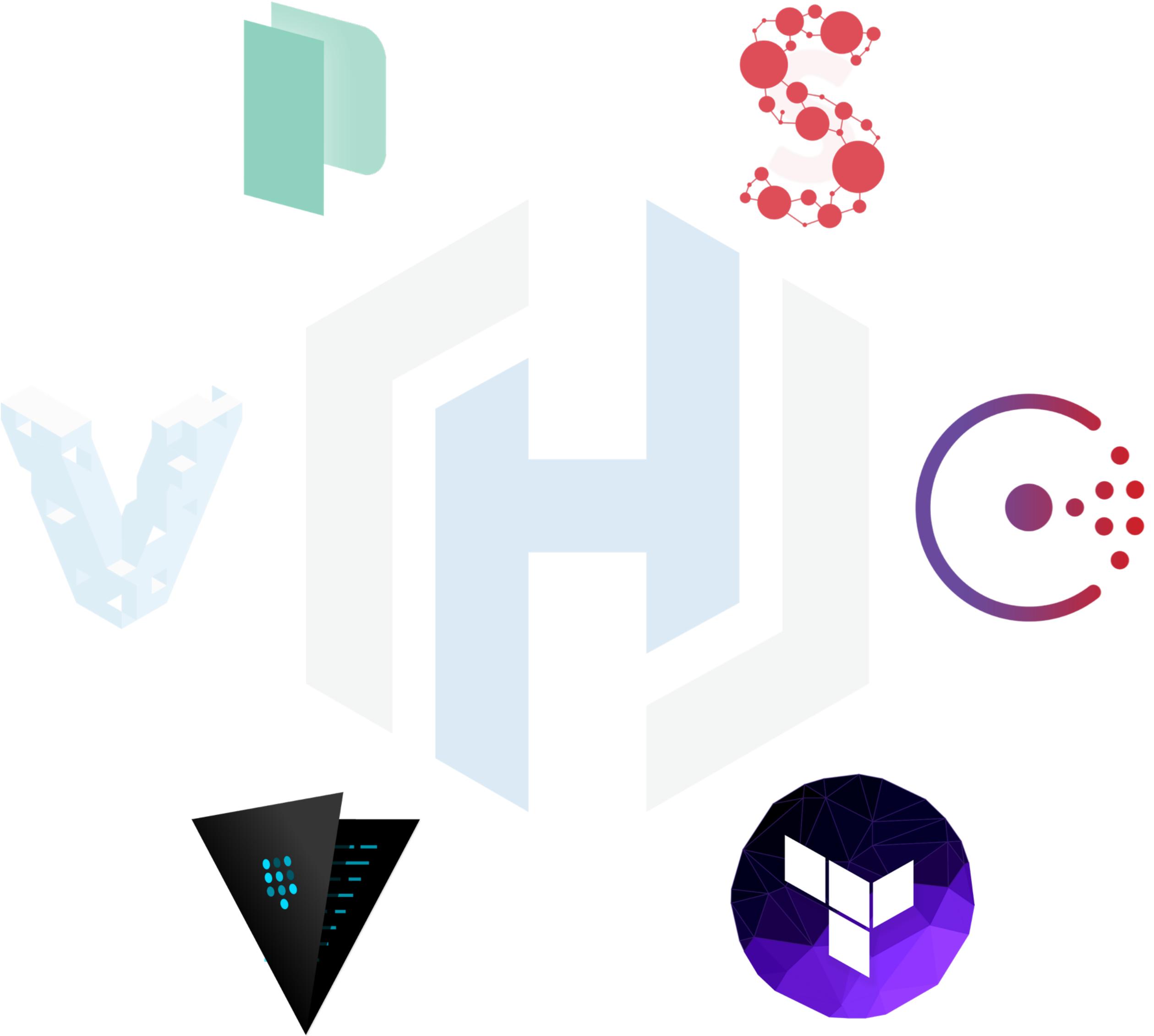


SETH VARGO

@sethvargo







HISTORY

○

2007



○

2007



"MODERN"

C and Java

Ruby and Python

NodeJS

Scala

JavaScript

Objective-C/Swift



O

2007

"MODERN"

C and Java
Ruby and Python
NodeJS
Scala
JavaScript
Objective-C/Swift

PROBLEMS



Pick one:
- efficient compilation
- efficient execution
- ease of development

Type safety
Modern



2007

2008

"MODERN"
C and Java
Ruby and Python
NodeJS
Scala
JavaScript
Objective-C/Swift

PROBLEMS
Pick one:
- efficient compilation
- efficient execution
- ease of development
Type safety
Modern

REQUIREMENTS
Compile FAST
Execute FAST
Concurrency
Built-in GC
Support for dependencies
Good enough for Google



.....

2007 2008



"MODERN"

C and Java
Ruby and Python
NodeJS
Scala
JavaScript
Objective-C/Swift

PROBLEMS

Pick one:
- efficient compilation
- efficient execution
- ease of development

Type safety
Modern

REQUIREMENTS

Compile FAST
Execute FAST
Concurrency
Built-in GC
Support for dependencies
Good enough for Google



2007



2008



2009

GO!



"MODERN"

C and Java
Ruby and Python
NodeJS
Scala
JavaScript
Objective-C/Swift

PROBLEMS

Pick one:
- efficient compilation
- efficient execution
- ease of development

Type safety
Modern

REQUIREMENTS

Compile FAST
Execute FAST
Concurrency
Built-in GC
Support for dependencies
Good enough for Google



2007



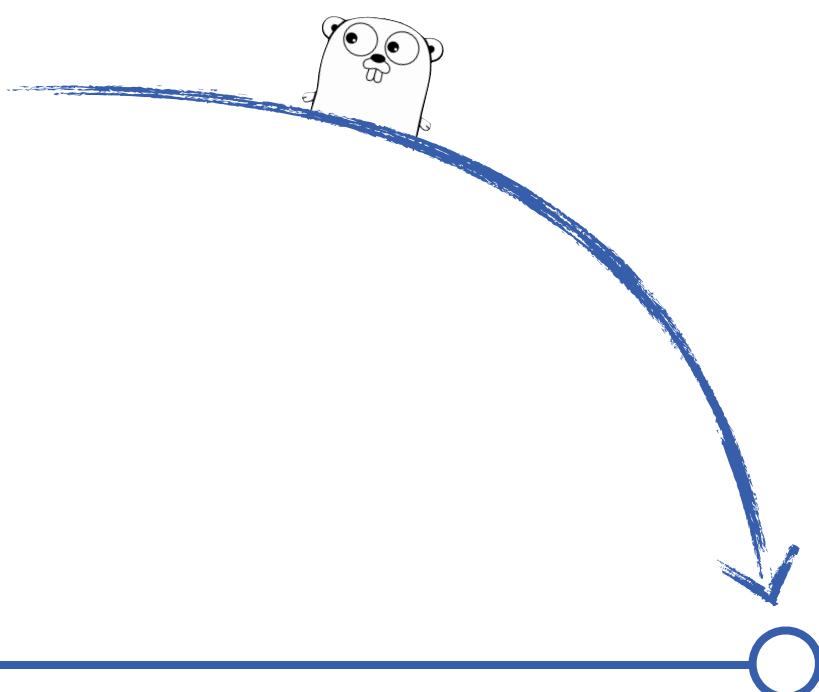
2008



2009

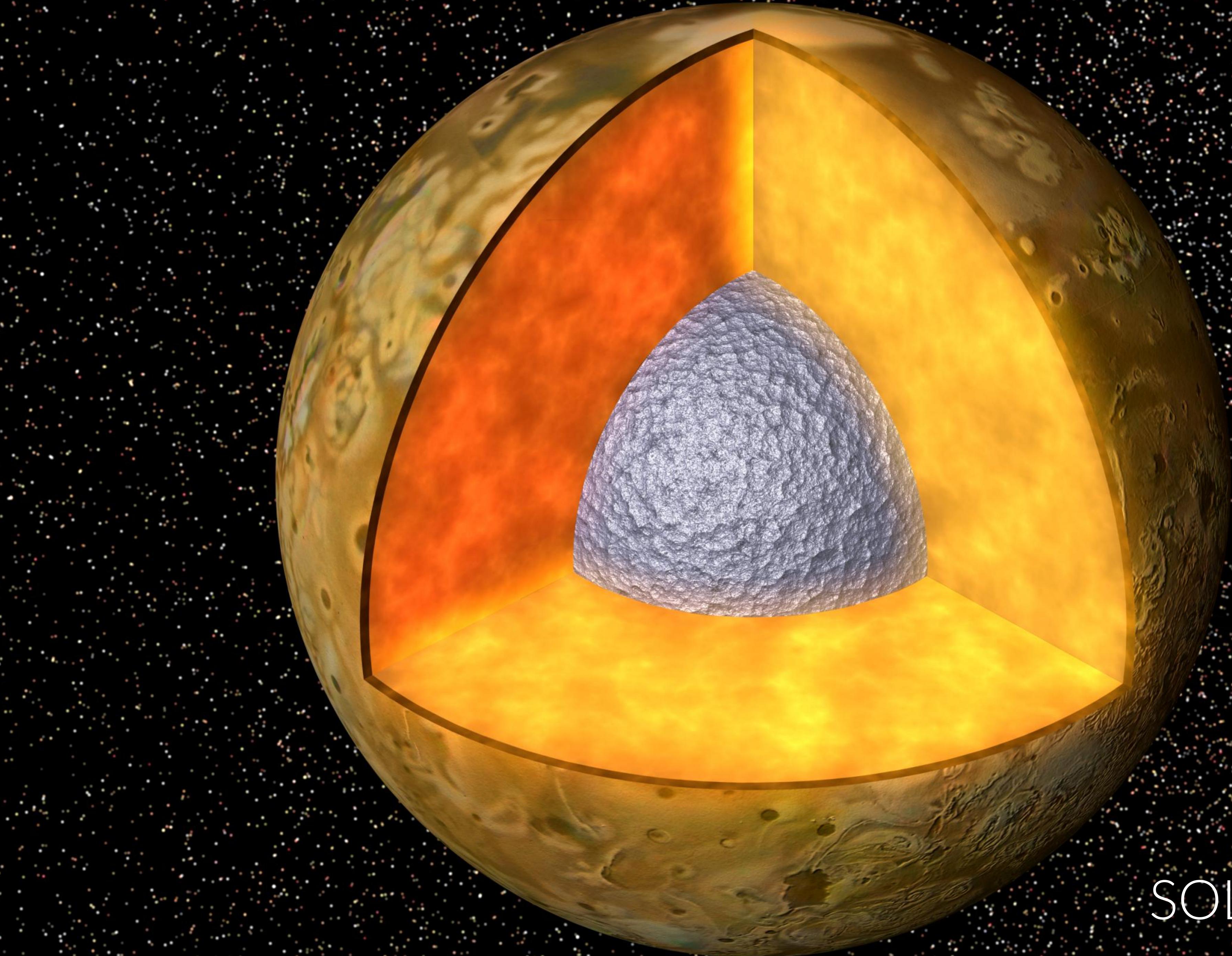
2015

today



OVERVIEW

GO HAS



SOLID CORE



GO HAS



A ROBUST COMMUNITY

GO HAS



INCREDIBLE PERFORMANCE



GO HAS

```
package main

import (
    "bytes"
    "fmt"
    "io/ioutil"
    "path/filepath"
    "sync"
    "text/template"
    dep "github.com/hashicorp/consul-template/dependency"
)

type Template struct {
    sync.Mutex
    // Path is the path to this template on disk.
    Path string
    // dependencies is the internal list of dependencies.
    dependencies []dep.Dependency
    // contents is string contents for this file when read from disk.
    contents string
}

// NewTemplate creates and parses a new Consul Template template at the given
// path. If the template does not exist, an error is returned. During
// initialization, the template is read and is parsed for dependencies. Any
// errors that occur are returned.
func NewTemplate(path string) (*Template, error) {
    template := &Template{Path: path}
    if err := template.init(); err != nil {
        return nil, err
    }

    return template, nil
}

// Execute evaluates this template in the context of the given brain. The
// return value is a slice of missing dependencies that were encountered
// during evaluation. If there are any missing dependencies found, it is
// considered unsafe to write the resulting contents to disk. This value
// is the contents of the template as-rendered. This value may not be
// final resulting template if there are any missing dependencies.
// return value is any error that occurred while evaluating the template.
func (t *Template) Execute(brain *Brain) ([]dep.Dependency, []byte, error) {
    usedMap := make(map[string]dep.Dependency)
    brain.UsedDependencies = usedMap
```

```
1 package main
2
3 import (
4     "bytes"
5     "fmt"
6     "io/ioutil"
7     "path/filepath"
8     "sync"
9     "text/template"
10
11    dep "github.com/hashicorp/consul-template/dependency"
12 )

13 type Template struct {
14     sync.Mutex
15     // Path is the path to this template on disk.
16     Path string
17     // dependencies is the internal list of dependencies.
18     dependencies []dep.Dependency
19     // contents is string contents for this file when read from disk.
20     contents string
21 }

22
23 // NewTemplate creates and parses a new Consul Template template at the given
24 // path. If the template does not exist, an error is returned. During
25 // initialization, the template is read and is parsed for dependencies. Any
26 // errors that occur are returned.
27 func NewTemplate(path string) (*Template, error) {
28     template := &Template{Path: path}
29     if err := template.init(); err != nil {
30         return nil, err
31     }

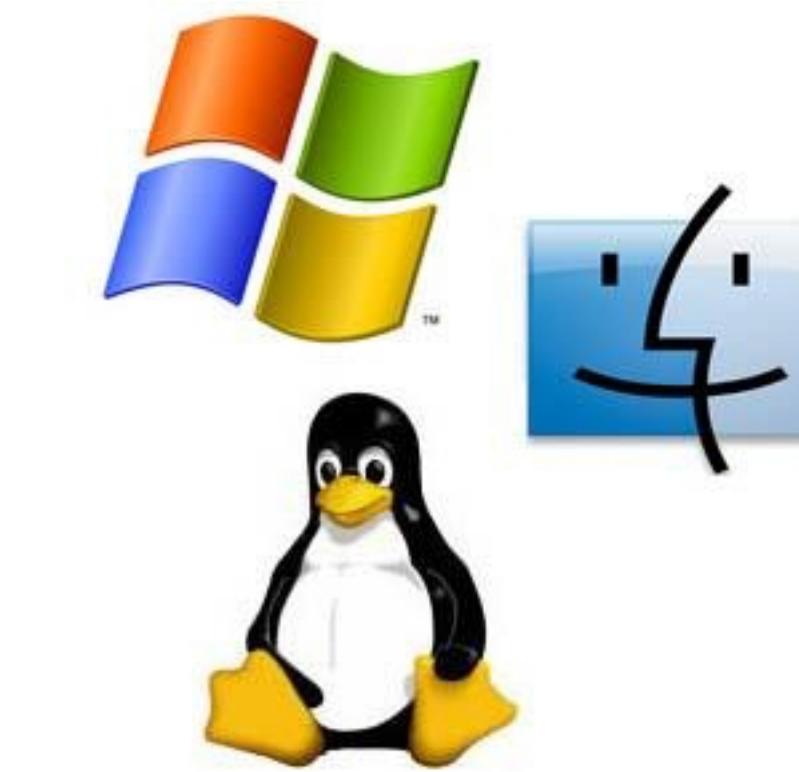
32     return template, nil
33 }

34 // Execute evaluates this template in the context of the given brain. The
35 // return value is a slice of missing dependencies that were encountered
36 // during evaluation. If there are any missing dependencies found, it is
37 // considered unsafe to write the resulting contents to disk. The second
38 // value is the contents of the template as-rendered. This value may not be
39 // final resulting template if there are any missing dependencies. The last
40 // value is any error that occurred while evaluating the template.
41 func (t *Template) Execute(brain *Brain) ([]dep.Dependency, []byte, error) {
42     usedMap := make(map[string]dep.Dependency)
43     brain.UsedDependencies = usedMap
```

BUILT-IN LINTING AND FORMATTING



GO HAS



EASY CROSS-PLATFORM BINARIES



GO HAS



INTERNAL GC



GO DOES NOT HAVE

The Go Programming Language

Try Go

```
// You can edit this code!
// Click here and start typing.
package main

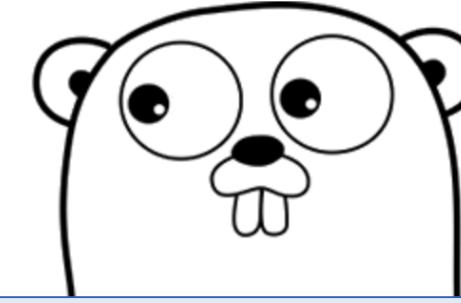
import "fmt"

func main() {
    fmt.Println("Hello, 世界")
}
```

Pop-out ↗

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, Mac OS X, Windows, and more.

Featured video

A Tour of Go



0:00 / 34:41

Featured articles

Package names

Go code is organized into packages. Within a package, code can refer to any identifier (name) defined within, while clients of the package may only reference the package's exported types, functions, constants, and variables. Such references always include the package name as a prefix: `foo.Bar` refers to the exported name `Bar` in the imported package named `foo`.

Published 4 February 2015

Errors are values

A common point of discussion among Go programmers, especially those new to the language, is how to handle errors. The conversation often turns into a lament at the number of times the sequence

Published 12 January 2015

Read more

Build version go1.4.2.
Except as noted, the content of this page is licensed under the Creative Commons Attribution 3.0 License, and code is licensed under a [BSD license](#).
[Terms of Service](#) | [Privacy Policy](#)

A PRETTY WEBSITE



GO DOES NOT HAVE



GENERICS

GO DOES NOT HAVE GO

Google search results for "go":

- Go (game) - Wikipedia, the free encyclopedia**
[en.wikipedia.org/wiki/Go_\(game\)](https://en.wikipedia.org/wiki/Go_(game)) ▾ Wikipedia ▾
Go (simplified Chinese: 围棋; traditional Chinese: 圍棋; pinyin: wéiqí, Japanese: 囲碁 igo, literal meaning: "encircling game", Korean: 바둑 baduk) is a board ...
Four Arts - Rules of Go - List of go games - Go ranks and ratings
- Go (1999) - IMDb**
www.imdb.com/title/tt0139239/ ▾ Internet Movie Database ▾
★★★☆☆ Rating: 7.3/10 - 56,477 votes
Directed by Doug Liman. With Sarah Polley, Jay Mohr, Scott Wolf, Taye Diggs. **Go!** tells the story of the events after a drug deal, told from three different points of ...
- What Is Go? | American Go Association**
www.usgo.org/what-go ▾ American Go Association ▾
Go is . . . ancient board game which takes simple elements: line and circle, black and white, stone and wood, combines them with simple rules and generates ...
- Go.com | The Walt Disney Company**
go.com/ ▾ Go.com ▾
Go.com is the top-level home on the Internet to the online properties of The Walt Disney Company.
- go - The Go Programming Language - Google Project Hosting**
code.google.com/p/go/ ▾
Go is an expressive, concurrent, garbage-collected programming language. The **Go** home page is the primary source of information about **Go**. It contains ...
- golang/go · GitHub**
<https://github.com/golang/go> ▾
The **Go** programming language. Contribute to **go** development by creating an account on GitHub. ... The **Go** programming language. <https://golang.org>.
- The Interactive Way To Go - playGO.to**
playgo.to/wtg/en/ ▾
Java-based interactive tutorial in several languages, in 33 brief lessons.
- Go at Sensei's Library**
senseis.xmp.net/?Go ▾
Jul 14, 2014 - **Go** is an ancient game which, from its forgotten origins in China, spread first to the rest of East Asia, and then to the entire world. How ancient?

See results about

Go (Programming Language)
Go, also commonly referred to as golang, is a programming language initially developed at ...

Go (Game)
Go is a board game involving two players that originated in ancient China more than 2,500 years ...

Go (1999 Film)
Initial release: April 7, 1999
Director: Doug Liman

SEO

INSTALL

The screenshot shows a web browser window with the URL golang.org/dl/ in the address bar. The page is titled "The Go Programming Language" and features a navigation menu with links to "Documents", "Packages", "The Project", "Help", "Blog", and "Search". Below the menu, a section titled "Downloads" is visible. A sub-section titled "Stable versions" contains a dropdown menu with the option "go1.4.2 ▾" selected. The main content area displays a table of download links for various Go versions across different platforms and architectures. The table includes columns for "File name", "Kind", "OS", "Arch", and "SHA1 Checksum". The "File name" column lists files such as "go1.4.2.src.tar.gz", "go1.4.2.darwin-386-osx10.6.tar.gz", and "go1.4.2.windows-amd64.msi". The "Kind" column indicates the type of file (Source, Archive, Installer). The "OS" and "Arch" columns specify the operating system and architecture. The "SHA1 Checksum" column provides a unique identifier for each file. The table rows alternate in background color for readability.

File name	Kind	OS	Arch	SHA1 Checksum
go1.4.2.src.tar.gz	Source			460caac03379f746c473814a65223397e9c9a2f6
go1.4.2.darwin-386-osx10.6.tar.gz	Archive	OS X 10.6+	32-bit	fb3e6b30f4e1b1be47bbb98d79dd53da8dec24ec
go1.4.2.darwin-386-osx10.8.tar.gz	Archive	OS X 10.8+	32-bit	65f5610fdb38febd869aeffbd426c83b650bb408
go1.4.2.darwin-386-osx10.6.pkg	Installer	OS X 10.6+	32-bit	3ed569ce33616d5d36f963e5d7cef55727c8621
go1.4.2.darwin-386-osx10.8.pkg	Installer	OS X 10.8+	32-bit	7f3fb2438fa0212febef13749d8d144934bb1c80
go1.4.2.darwin-amd64-osx10.6.tar.gz	Archive	OS X 10.6+	64-bit	00c3f9a03daff818b2132ac31d57f054925c60e7
go1.4.2.darwin-amd64-osx10.8.tar.gz	Archive	OS X 10.8+	64-bit	58a04b3eb9853c75319d9076df6f3ac8b7430f7f
go1.4.2.darwin-amd64-osx10.6.pkg	Installer	OS X 10.6+	64-bit	3fa5455e211a70c0a920abd53cb3093269c5149c
go1.4.2.darwin-amd64-osx10.8.pkg	Installer	OS X 10.8+	64-bit	8fde619d48864cb1c77ddc2a1aec0b7b20406b38
go1.4.2.linux-386.tar.gz	Archive	Linux	32-bit	50557248e89b6e38d395fda93b2f96b2b860a26a
go1.4.2.linux-amd64.tar.gz	Archive	Linux	64-bit	5020af94b52b65cc9b6f11d50a67e4bae07b0aff
go1.4.2.windows-386.zip	Archive	Windows	32-bit	0e074e66a7816561d7947ff5c3514be96f347dc4
go1.4.2.windows-386.msi	Installer	Windows	32-bit	e8bd3d87cb52441b2c9aee7c2c5f5ce7ffccc832
go1.4.2.windows-amd64.zip	Archive	Windows	64-bit	91b229a3ff0a1ce6e791c832b0b4670bfc5457b5
go1.4.2.windows-amd64.msi	Installer	Windows	64-bit	a914f3dad5521a8f658dce3e1575f3b6792975f0

[go1.4.1 ▾](#)

[go1.4 ▾](#)

\$GOPATH

```
~ ➤ go
Go is a tool for managing Go source code.

Usage:
  go command [arguments]

The commands are:

  build      compile packages and dependencies
  clean      remove object files
  env        print Go environment information
  fix        run go tool fix on packages
  fmt        run gofmt on package sources
  generate   generate Go files by processing source
  get        download and install packages and dependencies
  install    compile and install packages and dependencies
  list       list packages
  run        compile and run Go program
  test       test packages
  tool       run specified go tool
  version   print Go version
  vet        run go tool vet on packages

Use "go help [command]" for more information about a command.
```

```
package main

import "fmt"

// this is a comment

func main() {
    fmt.Println("Hello World!")
}
```

example.go



package main



import "fmt"

// this is a comment

```
func main() {  
    fmt.Println("Hello World!")  
}
```

example.go



```
package main

import "fmt" ←

// this is a comment

func main() {
    fmt.Println("Hello World!")
}
```

example.go



```
package main

import "fmt"

// this is a comment ←

func main() {
    fmt.Println("Hello World!")
}
```

example.go



```
package main

import "fmt"

// this is a comment

func main() {
    fmt.Println("Hello World!")
}
```

example.go



```
package main

import "fmt"

// this is a comment

func main() {
    fmt.Println("Hello World!") ←
}
```

example.go



2. sethvargo@Seth's MBP: ~ (zsh)

```
~ go build example.go
~ ./example
Hello World!
~
```

```
~ ➤ time ./example
Hello World!
./example 0.00s user 0.00s system 54% cpu 0.007 total
~ ➤
```



```
package main

import (
    "bufio"
    "fmt"
    "os"
    "strings"
)

func main() {
    reader := bufio.NewReader(os.Stdin)

    fmt.Print("Your name: ")
    text, err := reader.ReadString('\n')
    if err != nil {
        fmt.Println("error:", err.Error())
        return
    }

    fmt.Println(fmt.Sprintf("Your name is %q!", strings.TrimSpace(text)))
}
```

example.go



```
package main

import (
    "bufio"
    "fmt"
    "os"
    "strings"
)

func main() {
    reader := bufio.NewReader(os.Stdin)

    fmt.Print("Your name: ")
    text, err := reader.ReadString('\n')
    if err != nil {
        fmt.Println("error:", err.Error())
        return
    }

    fmt.Println(fmt.Sprintf("Your name is %q!", strings.TrimSpace(text)))
}
```

example.go



```
package main

import (
    "bufio"
    "fmt"
    "os"
    "strings"
)

func main() {
    reader := bufio.NewReader(os.Stdin) ←
        fmt.Print("Your name: ")
    text, err := reader.ReadString('\n')
    if err != nil {
        fmt.Println("error:", err.Error())
        return
    }

    fmt.Println(fmt.Sprintf("Your name is %q!", strings.TrimSpace(text)))
}
```

example.go



```
package main

import (
    "bufio"
    "fmt"
    "os"
    "strings"
)

func main() {
    reader := bufio.NewReader(os.Stdin)

    fmt.Print("Your name: ")
    text, err := reader.ReadString('\n') ←
    if err != nil {
        fmt.Println("error:", err.Error())
        return
    }

    fmt.Println(fmt.Sprintf("Your name is %q!", strings.TrimSpace(text)))
}
```

example.go



```
package main

import (
    "bufio"
    "fmt"
    "os"
    "strings"
)

func main() {
    reader := bufio.NewReader(os.Stdin)

    fmt.Print("Your name: ")
    text, err := reader.ReadString('\n')
    if err != nil {
        fmt.Println("error:", err.Error())
        return
    }

    fmt.Println(fmt.Sprintf("Your name is %q!", strings.TrimSpace(text)))
}
```

example.go



```
package main

import (
    "bufio"
    "fmt"
    "os"
    "strings"
)

func main() {
    reader := bufio.NewReader(os.Stdin)

    fmt.Print("Your name: ")
    text, err := reader.ReadString('\n')
    if err != nil {
        fmt.Println("error:", err.Error())
        return
    }

    fmt.Println(fmt.Sprintf("Your name is %q!", strings.TrimSpace(text)))
}
```

example.go



```
package main

import (
    "bufio"
    "fmt"
    "os"
    "strings"
)

func main() {
    reader := bufio.NewReader(os.Stdin)

    fmt.Print("Your name: ")
    text, err := reader.ReadString('\n')
    if err != nil {
        fmt.Println("error:", err.Error())
        return
    }

    fmt.Println(fmt.Sprintf("Your name is %q!", strings.TrimSpace(text)))
}
```

example.go



```
package main

import (
    "bufio"
    "fmt"
    "os"
    "strings"
)

func main() {
    reader := bufio.NewReader(os.Stdin)

    fmt.Print("Your name: ")
    text, err := reader.ReadString('\n')
    if err != nil {
        fmt.Println("error:", err.Error())
        return
    }

    fmt.Println(fmt.Sprintf("Your name is %q!", strings.TrimSpace(text)))
}
```

example.go



```
package main

func main() {
    reader := bufio.NewReader(os.Stdin)

    fmt.Print("Your name: ")
    text, err := reader.ReadString('\n')
    if err != nil {
        fmt.Println("error:", err.Error())
        return
    }

    fmt.Println(fmt.Sprintf("Your name is %q!", strings.TrimSpace(text)))
}
```

example.go



```
package main

func main() {
    reader := bufio.NewReader(os.Stdin)

    text, err := reader.ReadString('\n')
    if err != nil {
        fmt.Println("error:", err.Error())
        return
    }

    fmt.Println(fmt.Sprintf("Your name is %q!", strings.TrimSpace(text)))
}
```

example.go



```
package main

func main() {
    reader := bufio.NewReader(os.Stdin)

    text, err := reader.ReadString('\n')

    fmt.Println(fmt.Sprintf("Your name is %q!", strings.TrimSpace(text)))
}
```

example.go



```
package main

func main() {
    text, err := reader.ReadString('\n')
    fmt.Println(fmt.Sprintf("Your name is %q!", strings.TrimSpace(text)))
}
```

example.go



```
package main

func main() {
    text,      := "Hi\n"
    fmt.Println(fmt.Sprintf("Your name is %q!", strings.TrimSpace(text)))
}
```

example.go



```
package main

func main() {
    text,      := "Hi\n"
    fmt.Println(fmt.Sprintf("Your name is %q!", strings.TrimSpace("Hi\n")))
}
```

example.go



```
package main

func main() {
    text,      := "Hi\n"
    fmt.Println(fmt.Sprintf("Your name is %q!", "Hi"))
}
```

example.go



```
package main

func main() {
    text,      := "Hi\n"
    fmt.Println(fmt.Sprintf("Your name is \"Hi\"!"))
}
```

example.go



```
package main

func main() {
    text,      := "Hi\n"
    fmt.Println("Your name is \"Hi\"!")
}
```

example.go



2. sethvargo@Seth's MBP: ~ (zsh)

```
~/Documents/Code/Python/HelloWorld> ./example
Your name: Hi
Your name is "Hi"!
~/Documents/Code/Python/HelloWorld>
```



```
package main

import (
    "bufio"
    "fmt"
    "os"
    "strings"
)

func main() {
    reader := bufio.NewReader(os.Stdin)

    fmt.Print("Your name: ")
    text, err := reader.ReadString('\n')
    if err != nil {
        fmt.Println("error:", err.Error())
        return
    }

    fmt.Println(fmt.Sprintf("Your name is %q!", strings.TrimSpace(text)))
}
```

example.go



```
package main

import (
    "bufio"
    "fmt"
    "os"
    "strings"
)

func main() {
    reader := bufio.NewReader(os.Stdin)

    fmt.Print("Your name: ")
    text, err := reader.ReadString('\n')
    if err != nil {
        fmt.Println("error:", err.Error())
        return
    }

    fmt.Println(fmt.Sprintf("Your name is %q!", strings.TrimSpace(text)))
}
```

example.go



```
package main

import (
    "bufio"
    "fmt"
    "os"
    "strings"
)

func main() {
    reader := bufio.NewReader(os.Stdin)

    fmt.Print("Your name: ")
    text, err := reader.ReadString('\n')
    if err != nil {
        fmt.Println("error:", err.Error())
        return
    }

    fmt.Println(fmt.Sprintf("Your name is %q!", strings.TrimSpace(text)))
}
```

example.go



```
package main

import (
    "bufio"
    "fmt"
    "os"
    "strings"
)

func main() {
    reader := bufio.NewReader(os.Stdin)

    fmt.Print("Your name: ")
    text, err := reader.ReadString('\n')
    if err != nil {
        fmt.Println("error:", err.Error())
        return
    }

    fmt.Println(fmt.Sprintf("Your name is %q!", strings.TrimSpace(text)))
}
```

example.go



```
package main

import (
    "bufio"
    "fmt"
    "os"
    "strings"
    "io/ioutil" ←
)

func main() {
    reader := bufio.NewReader(os.Stdin)

    fmt.Print("Your name: ")
    text, err := reader.ReadString('\n')
    if err != nil {
        fmt.Println("error:", err.Error())
        return
    }

    fmt.Println(fmt.Sprintf("Your name is %q!", strings.TrimSpace(text)))
}
```

example.go



2. sethvargo@Seth's MBP: ~ (zsh)

```
~ ➤ go build example.go
# command-line-arguments
./example.go:8: imported and not used: "io/ioutil"
✖ ➤ ~
```

FIZZBUZZ



```
def fizzbuzz(max)
  1.upto(max) do |i|
    if i % 5 == 0 and i % 3 == 0
      puts "FizzBuzz"
    elsif i % 3== 0
      puts "Fizz"
    elsif i % 5 == 0
      puts "Buzz"
    else
      puts i
    end
  end
end
```

fizzbuzz.rb



```
func fizzbuzz(max int) {
    for i := 1; i <= max; i++ {
        if i %3 == 0 && i %5 == 0 {
            fmt.Println("Fizzbuzz")
        } else if i%3 == 0 {
            fmt.Println("Fizz")
        } else if i%5 == 0 {
            fmt.Println("Buzz")
        } else {
            fmt.Println(i)
        }
    }
}
```

fizzbuzz.go



```
func fizzbuzz(max int) {  
    for i := 1; i <= max; i++ {  
        if i %3 == 0 && i %5 == 0 {  
            fmt.Println("Fizzbuzz")  
        } else if i%3 == 0 {  
            fmt.Println("Fizz")  
        } else if i%5 == 0 {  
            fmt.Println("Buzz")  
        } else {  
            fmt.Println(i)  
        }  
    }  
}
```

fizzbuzz.go

```
def fizzbuzz(max)  
    1.upto(max) do |i|  
        if i % 5 == 0 and i % 3 == 0  
            puts "FizzBuzz"  
        elsif i % 3== 0  
            puts "Fizz"  
        elsif i % 5 == 0  
            puts "Buzz"  
        else  
            puts i  
        end  
    end  
end
```

fizzbuzz.rb



```
func fizzbuzz(max int) {
    for i := 1; i <= max; i++ {
        if i %3 == 0 && i %5 == 0 {
            fmt.Println("Fizzbuzz")
            continue
        }

        if i%3 == 0 {
            fmt.Println("Fizz")
            continue
        }

        if i%5 == 0 {
            fmt.Println("Buzz")
            continue
        }

        fmt.Println(i)
    }
}
```

```
def fizzbuzz(max)
  1.upto(max) do |i|
    if i % 5 == 0 and i % 3 == 0
      puts "FizzBuzz"
      next
    end

    if i % 3== 0
      puts "Fizz"
      next
    end

    if i % 5 == 0
      puts "Buzz"
      next
    end

    puts i
  end
end
```

fizzbuzz.go

fizzbuzz.rb

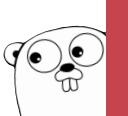


FIBONACCI



```
def fib(max)
  if max < 2
    return max
  else
    return fib(max-1) + fib(max-2)
  end
end
```

fib.rb



```
func fib(max int) int {  
    if max < 2 {  
        return max  
    } else {  
        return fib(max-1) + fib(max-2)  
    }  
}
```

fib.go



```
func fib(max int) int {  
    if max < 2 {  
        return max  
    } else {  
        return fib(max-1) + fib(max-2)  
    }  
}
```

```
def fib(max)  
    if max < 2  
        return max  
    else  
        return fib(max-1) + fib(max-2)  
    end  
end
```

fib.go

fib.rb



```
func fib(max int) int {  
    if max < 2 {  
        return max  
    } else {  
        return fib(max-1) + fib(max-2)  
    }  
}
```

fib(40)

fib.go



```
def fib(max)  
    if max < 2  
        return max  
    else  
        return fib(max-1) + fib(max-2)  
    end  
end
```

fib(40)

fib.rb

```
func fib(max int) int {  
    if max < 2 {  
        return max  
    } else {  
        return fib(max-1) + fib(max-2)  
    }  
}
```

```
fib(40)  
1.02s user  
0.01s system  
99% cpu  
1.027 total
```

```
def fib(max)  
    if max < 2  
        return max  
    else  
        return fib(max-1) + fib(max-2)  
    end  
end
```

```
fib(40)
```

fib.rb

fib.go



```
func fib(max int) int {  
    if max < 2 {  
        return max  
    } else {  
        return fib(max-1) + fib(max-2)  
    }  
}
```

```
fib(40)  
1.02s user  
0.01s system  
99% cpu  
1.027 total
```

```
def fib(max)  
    if max < 2  
        return max  
    else  
        return fib(max-1) + fib(max-2)  
    end  
end
```

```
fib(40)  
17.37s user  
0.04s system  
99% cpu  
17.467 total
```

fib.go

fib.rb



```
func fib(max int) int {  
    i, j := 0, 1  
    for c := 0; c < max; c++ {  
        i, j = i+j, i  
    }  
    return i  
}
```

```
fib(40)
```

fib.go



```
func fib(max int) int {  
    i, j := 0, 1  
    for c := 0; c < max; c++ {  
        i, j = i+j, i  
    }  
    return i  
}
```

```
fib(40)  
0.00s user  
0.00s system  
74% cpu  
0.004 total
```

fib.go



```
func fib(max int) int {  
    i, j := 0, 1  
    for c := 0; c < max; c++ {  
        i, j = i+j, i  
    }  
    return i  
}
```

```
fib(1000)
```

fib.go



```
func fib(max int) int {  
    i, j := 0, 1  
    for c := 0; c < max; c++ {  
        i, j = i+j, i  
    }  
    return i  
}
```

```
fib(1000)  
0.00s user  
0.00s system  
72% cpu  
0.004 total
```

fib.go



```
func fib(max int) int {  
    i, j := 0, 1  
    for c := 0; c < max; c++ {  
        i, j = i+j, i  
    }  
    return i  
}
```

```
fib(1000)  
0.00s user  
0.00s system  
72% cpu  
0.004 total
```

```
43466557686937456435688527675040625802564660517371780402481729089536555417  
94905189040387984007925516929592259308032263477520968962323987332247116164  
2996440906533187938298969649928516003704476137795166849228875
```

fib.go



```
func fibGenerator() ? {  
  
    return ?  
}
```

fib.go



```
func fibGenerator() ? {  
    (some closureyish thing)  
  
    return ?  
}
```

fib.go



```
func fibGenerator() ? {  
  
    for i, j := 0, 1; ; i, j = i+j, i {  
        // do something with i  
    }  
  
    return ?  
}
```

fib.go



```
func fibGenerator() ? {  
  
    func() {  
        for i, j := 0, 1; ; i, j = i+j, i {  
            // do something with i  
        }  
    }()  
  
    return ?  
}
```

fib.go



```
func fibGenerator() ? {  
  
    go func() {  
        for i, j := 0, 1; ; i, j = i+j, i {  
            // do something with i  
        }  
    }()  
  
    return ?  
}
```

fib.go



```
func fibGenerator()      ?      {
    c := someMagicalGenerator()

    go func() {
        for i, j := 0, 1; i, j = i+j, i {
            // do something with i
        }
    }()
}

return ?
}
```

fib.go



```
func fibGenerator()      ?      {
    c := someMagicalGenerator()

    go func() {
        for i, j := 0, 1; i, j = i+j, i {
            c.send(i)
        }
    }()
}

return ?
}
```

fib.go



```
func fibGenerator()      ?      {
    c := someMagicalGenerator()

    go func() {
        for i, j := 0, 1; i, j = i+j, i {
            c.send(i)
        }
    }()
}

return c
}
```

fib.go



fib.go

someMagicalGenerator() ???



CH 22



```
func fibGenerator()      ?      {
    c := someMagicalGenerator()

    go func() {
        for i, j := 0, 1; i, j = i+j, i {
            c.send(i)
        }
    }()
}

return c
}
```

fib.go



```
func fibGenerator() ? {
    c := make(chan int)

    go func() {
        for i, j := 0, 1; i, j = i+j, i {
            c.send(i)
        }
    }()
}

return c
}
```

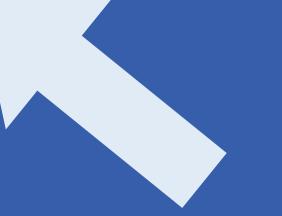
fib.go



```
func fibGenerator() <-chan int {  
    c := make(chan int)  
  
    go func() {  
        for i, j := 0, 1; ; i, j = i+j, i {  
            c.send(i)  
        }  
    }()  
  
    return c  
}
```

fib.go



```
func fibGenerator() <-chan int {  
    c := make(chan int)   
    go func() {  
        for i, j := 0, 1; ; i, j = i+j, i {  
            c.send(i)  
        }  
    }()  
  
    return c  
}
```

fib.go



```
func fibGenerator() <-chan int {  
    c := make(chan int)  
  
    go func() {  
        for i, j := 0, 1; ; i, j = i+j, i {  
            c <- i  
        }  
    }()  
  
    return c  
}
```

fib.go



```
func fibGenerator() <-chan int {
    c := make(chan int)

    go func() {
        for i, j := 0, 1; ; i, j = i+j, i {
            c <- i
        }
    }()
}

return c
}

func main() {
    fib := fibGenerator()
    for i := 0; i < 25; i++ {
        fmt.Println( ? )
    }
}
```

fib.go



```
func fibGenerator() <-chan int {
    c := make(chan int)

    go func() {
        for i, j := 0, 1; i, j = i+j, i {
            c <- i
        }
    }()
}

return c
}

func main() {
    fib := fibGenerator()
    for i := 0; i < 25; i++ {
        fmt.Println(<-fib)
    }
}
```

fib.go



fib.go

```
0  
1  
1  
2  
3  
5  
8  
13  
21  
34  
55  
89  
144  
233  
377  
610  
987  
1597  
2584  
4181  
6765  
10946  
17711
```



```
func fibGenerator() <-chan int {
    c := make(chan int)

    go func() {
        for i, j := 0, 1; ; i, j = i+j, i {
            c <- i
        }
    }()
}

return c
}

func main() {
    fib := fibGenerator()
    for i := 0; i < 25; i++ {
        fmt.Println(<-fib)
    }
}
```

fib.go



```
func fibGenerator() <-chan int {
    c := make(chan int)

    go func() {
        for i, j := 0, 1; i, j = i+j, i {
            c <- i
        }
    }()
}

return c
}

func main() {
    fib1 := fibGenerator()
    fib2 := fibGenerator()

    for i := 0; i < 5; i++ {
        fmt.Println(<-fib1)
    }

    for i := 0; i < 5; i++ {
        fmt.Println(<-fib2)
    }
}
```

fib.go



```
func fibGenerator() <-chan int {
    c := make(chan int)

    go func() {
        for i, j := 0, 1; i, j = i+j, i {
            c <- i
        }
    }()
}

return c
}

func main() {
    fib1 := fibGenerator()
    fib2 := fibGenerator()

    for i := 0; i < 5; i++ {
        fmt.Println(<-fib1)
    }

    for i := 0; i < 5; i++ {
        fmt.Println(<-fib2)
    }
}
```

fib.go



```
go func() {
    for i, j := 0, 1; i, j = i+j, i {
        c <- i
    }
}()
```

fib.go



fib.go

```
for i, j := 0, 1; ; i, j = i+j, i {  
    c <- i  
}
```



fib.go

```
for i, j := 0, 1; ; i, j = i+j, i {  
    c <- i  
}
```



fib.go

```
for {  
    c <- i  
}
```



```
func fibGenerator() <-chan int {  
    c := make(chan int)  
  
    go func() {  
        for i, j := 0, 1; ; i, j = i+j, i {  
            c <- i  
        }  
    }()  
  
    return c  
}
```

fib.go



```
func fibGenerator() <-chan int {  
    c := make(chan int, 50)  
  
    go func() {  
        for i, j := 0, 1; ; i, j = i+j, i {  
            c <- i  
        }  
    }()  
  
    return c  
}
```

fib.go



```
func fibGenerator() <-chan int {  
    c := make(chan int, 50)  
  
    go func() {  
        for i, j := 0, 1; ; i, j = i+j, i {  
            c <- i  
        }  
    }()  
  
    return c  
}
```

fib.go



```
func fibGenerator() <-chan int {  
    c := make(chan int, 50)  
  
    go func() {  
        for i, j := 0, 1; ; i, j = i+j, i {  
            c <- i  
        }  
    }()  
  
    return c  
}
```

fib.go



STRUCTS



```
type fib struct {
    // buffer is the size of the channel buffer.
    buffer int

    // ch is the internal channel where numbers are written.
    ch chan int
}
```

structs.go



```
type Fib struct {
    // Buffer is the size of the channel buffer.
    Buffer int

    // Ch is the internal channel where numbers are written.
    Ch chan int
}
```

structs.go



```
type Fib struct {
    // Buffer is the size of the channel buffer.
    Buffer int

    // Ch is the internal channel where numbers are written.
    Ch chan int
}

func PublicFunc() {}

func privateFunc() {}
```

structs.go



```
type fib struct {
    // buffer is the size of the channel buffer.
    buffer int

    // ch is the internal channel where numbers are written.
    ch chan int
}
```

structs.go



```
type fib struct {
    // buffer is the size of the channel buffer.
    buffer int

    // ch is the internal channel where numbers are written.
    ch chan int
}

func NewFib(b int) (*fib, error) {
    f := new(fib)
    f.buffer = b
    f.ch = make(chan int, b)

    go func(f *Fib) {
        for i, j := 0, 1; ; i, j = i+j, i {
            f.ch <- i
        }
    }(f)

    return f, nil
}
```

structs.go



```
func main() {
    f, err := NewFib(10)
    if err != nil {
        panic(err)
    }

    for i := 0; i < 10; i++ {
        fmt.Println(<-f.ch)
    }
}
```

structs.go



```
func main() {  
    f, err := NewFib(10)  
    if err != nil {  
        panic(err)  
    }  
  
    for i := 0; i < 10; i++ {  
        fmt.Println(<-f.ch)  
    }  
}
```



structs.go



```
type fib struct {  
    // buffer is the size of the channel buffer.  
    buffer int  
  
    // ch is the internal channel where numbers are written.  
    ch chan int  
}
```

structs.go



```
type fib struct {
    // buffer is the size of the channel buffer.
    buffer int

    // ch is the internal channel where numbers are written.
    ch chan int
}

// Next gets the next number in the fib sequence.
func (f *fib) Next() int {

}
```

structs.go



```
type Fib struct {
    // buffer is the size of the channel buffer.
    buffer int

    // ch is the internal channel where numbers are written.
    ch chan int
}

// Next gets the next number in the fib sequence.
func (f *fib) Next() int {
    return <-f.ch
}
```

structs.go



```
func main() {
    f, err := NewFib(10)
    if err != nil {
        panic(err)
    }

    for i := 0; i < 10; i++ {
        fmt.Println(<-f.ch)
    }
}
```

structs.go



```
func main() {
    f, err := NewFib(10)
    if err != nil {
        panic(err)
    }

    for i := 0; i < 10; i++ {
        fmt.Println(f.Next())
    }
}
```

structs.go



```
func main() {
    f, err := NewFib(10)
    if err != nil {
        panic(err)
    }

    for {
        fmt.Println(f.Next())
        time.Sleep(100 * time.Millisecond)
    }
}
```

structs.go



structs.go

```
0  
1  
1  
2  
3  
5  
8  
13  
21  
34  
55  
89  
144  
233  
377  
610  
987  
1597  
2584  
4181  
6765  
10946  
17711
```

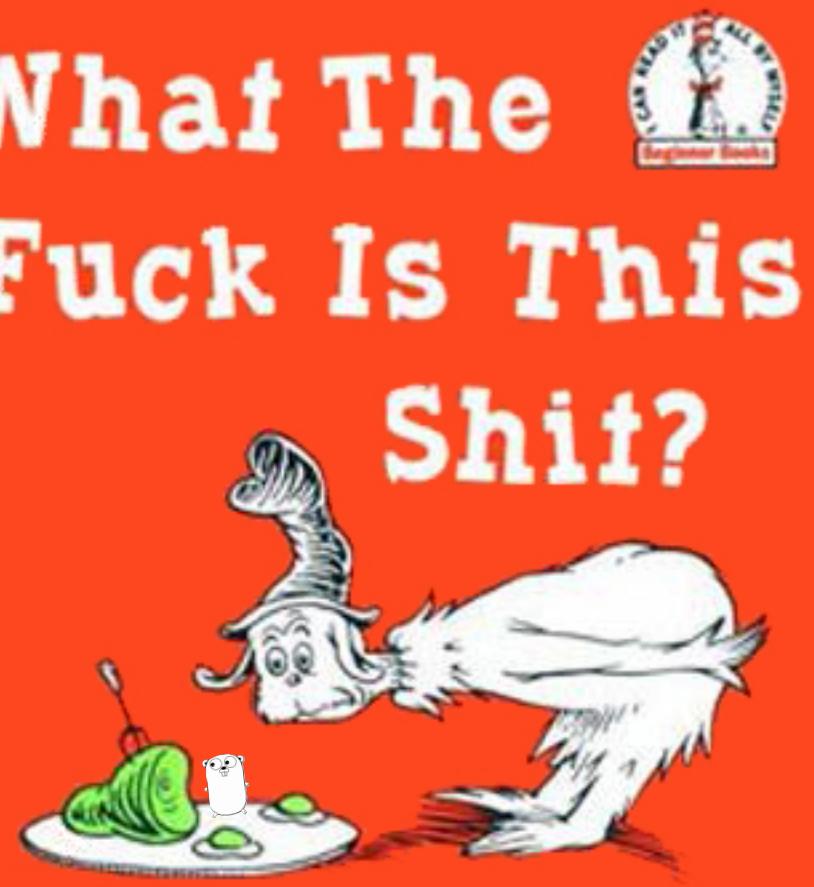


1779979416004714189
2880067194370816120
4660046610375530309
7540113804746346429
-6246583658587674878
1293530146158671551
-4953053512429003327
-3659523366270331776
-8612576878699335103
6174643828739884737
-2437933049959450366
3736710778780434371
1298777728820984005
5035488507601418376
6334266236422402381
-7076989329685730859
-742723093263328478
-7819712422949059337
-8562435516212387815
2064596134548104464
-6497839381664283351
-4433243247116178887
7515661444929089378

structs.go



What The
Fuck Is This
Shit?



By Dr. Seuss

`uint64: range: 0 through 18446744073709551615.`



The screenshot shows a web browser window displaying the Go Programming Language documentation for the `big` package. The URL in the address bar is `golang.org/pkg/math/big/#Int.Add`. The page title is "The Go Programming Language". The top navigation bar includes links for "Documents", "Packages", "The Project", "Help", "Blog", "Play", and "Search". Below the navigation bar, the section title "Package big" is displayed in blue. A code snippet at the top shows the import statement `import "math/big"`. To the right of the code are three links: "Overview", "Index", and "Examples". The "Overview" tab is currently selected, indicated by a blue background. The content under the "Overview" tab states: "Package big implements multi-precision arithmetic (big numbers). The following numeric types are supported:" followed by a list: "- Int signed integers" and "- Rat rational numbers". Below this, it says: "Methods are typically of the form:" followed by the signature `func (z *Int) Op(x, y *Int) *Int (similar for *Rat)`. Further down, it explains: "and implement operations $z = x \text{ Op } y$ with the result as receiver; if it is one of the operands it may be overwritten (and its memory reused). To enable chaining of operations, the result is also returned. Methods returning a result other than `*Int` or `*Rat` take one of the operands as the receiver." The "Index" tab is also present. At the bottom left, there is a small cartoon character icon.

The Go Programming Language

golang.org/pkg/math/big/#Int.Add

Documents Packages The Project Help Blog Play Search

Package big

```
import "math/big"
```

Overview Index Examples

Overview ▾

Package big implements multi-precision arithmetic (big numbers). The following numeric types are supported:

- Int signed integers
- Rat rational numbers

Methods are typically of the form:

```
func (z *Int) Op(x, y *Int) *Int (similar for *Rat)
```

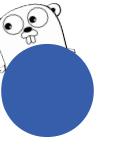
and implement operations $z = x \text{ Op } y$ with the result as receiver; if it is one of the operands it may be overwritten (and its memory reused). To enable chaining of operations, the result is also returned. Methods returning a result other than `*Int` or `*Rat` take one of the operands as the receiver.

Index ▾

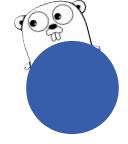
Constants

```
type Int
func NewInt(x int64) *Int
func (z *Int) Abs(x *Int) *Int
func (z *Int) Add(x, y *Int) *Int
func (z *Int) And(x, y *Int) *Int
func (z *Int) AndNot(x, y *Int) *Int
func (z *Int) Binomial(n, k int64) *Int
func (x *Int) Bit(i int) uint
func (x *Int) BitLen() int
func (x *Int) Bits() []Word
func (x *Int) Bytes() []byte
func (x *Int) Cmp(y *Int) (r int)
func (z *Int) Div(x, y *Int) *Int
func (z *Int) DivMod(x, y, m *Int) (*Int, *Int)
func (z *Int) Exp(x, y, m *Int) *Int
```

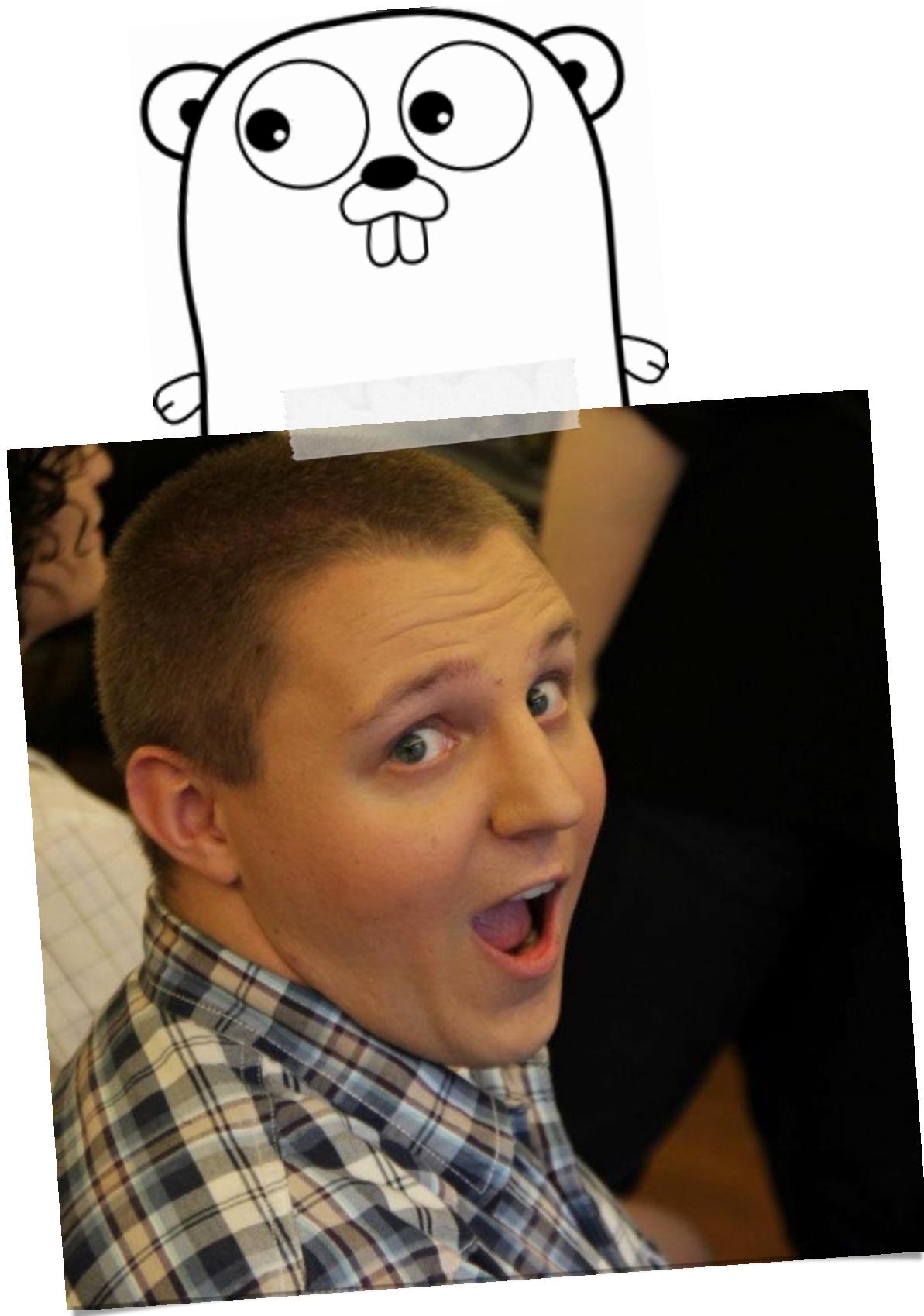
MORE?



MORE?



tour.golang.org



SETH VARGO
@sethvargo

THANK YOU!

