

# How Static Code Analysis Prevents You From Waking Up at 3AM With Production on Fire


Xe Iaso

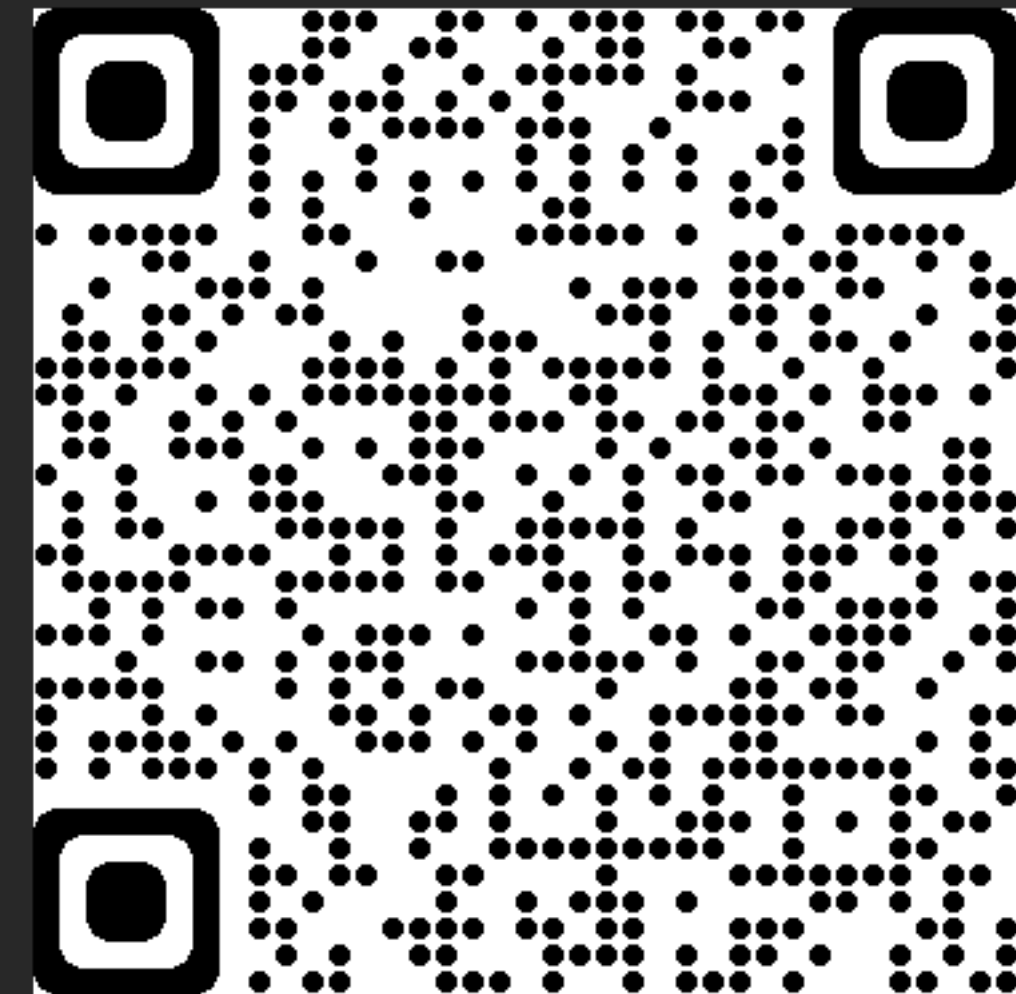
<https://christine.website>

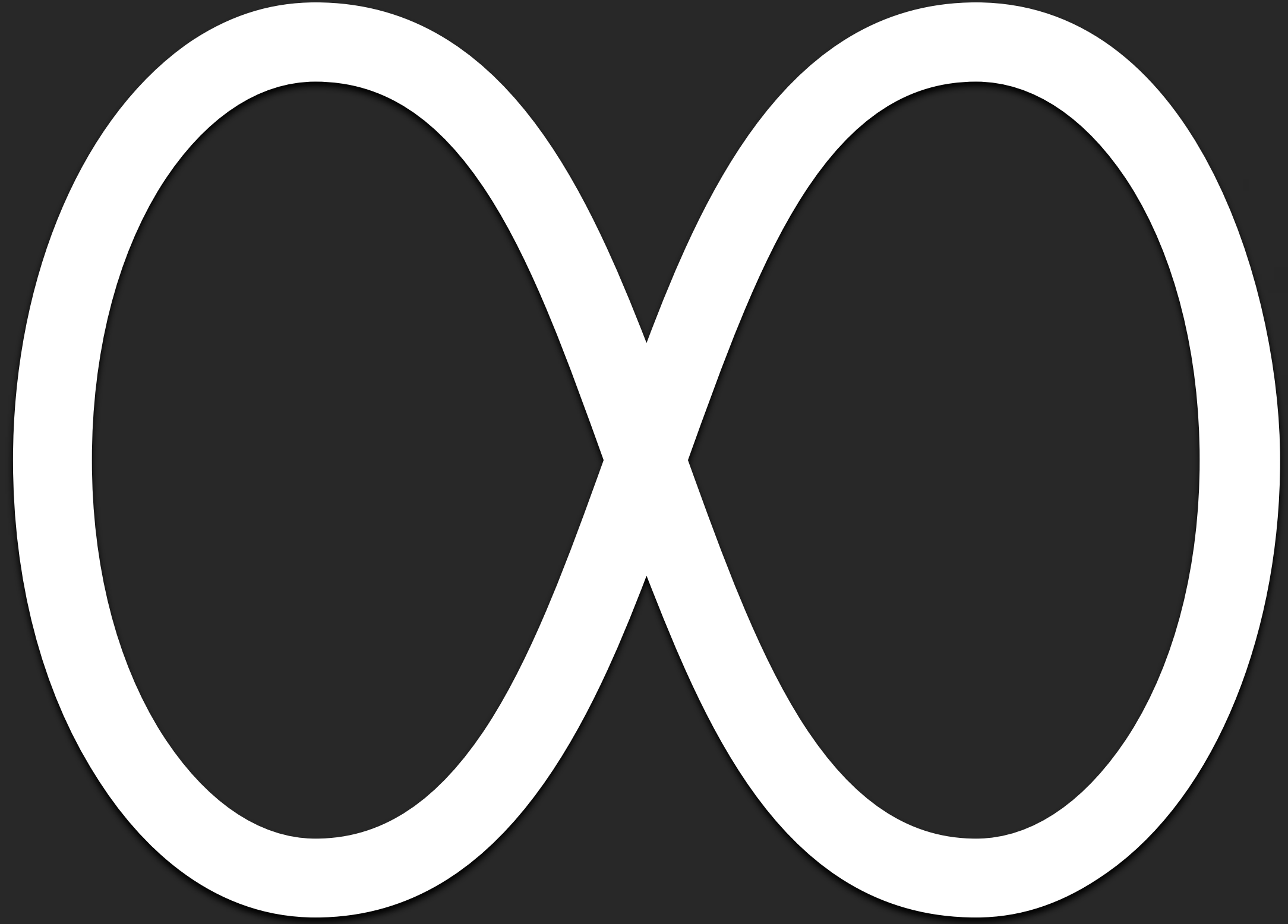
Conf42 SRE 2022

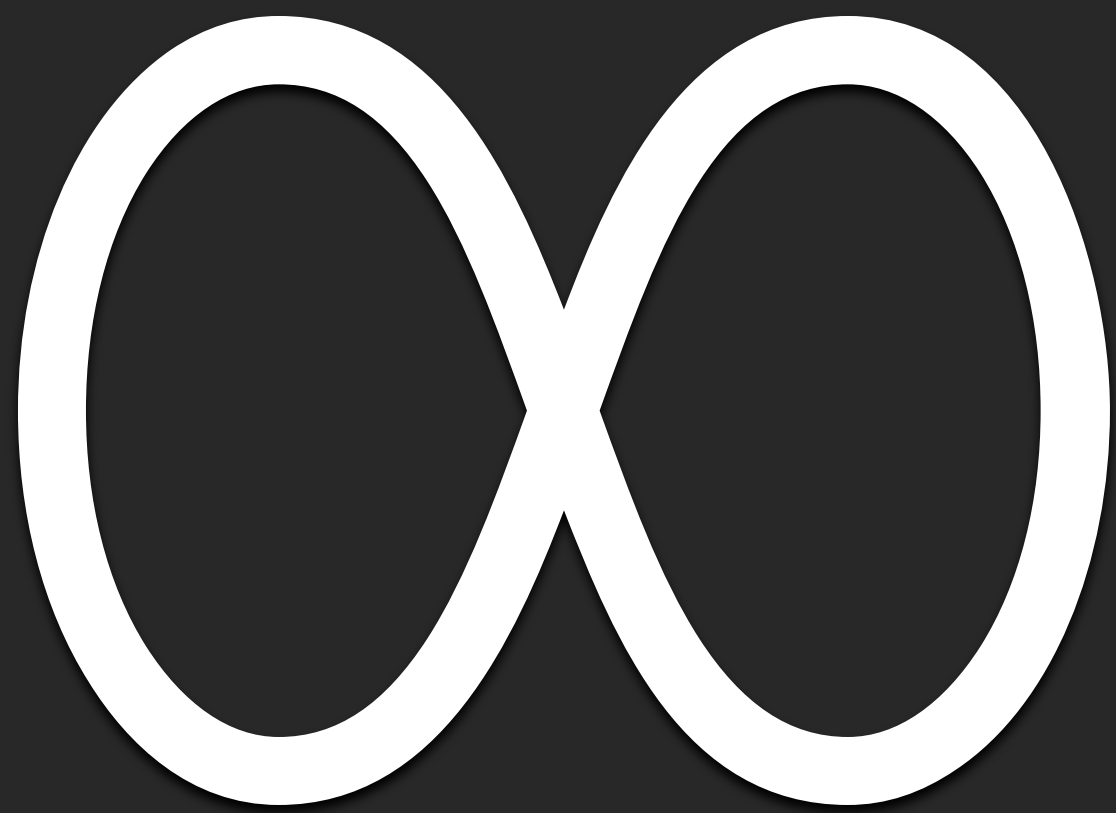


# Speaker Introduction

- ✦ Archmage of Infrastructure at Tailscale 
- ✦ SRE for half of my career
- ✦ Moving into DevRel
- ✦ This talk may contain opinions, these opinions are my own and not always the opinions of my employer







Compiler

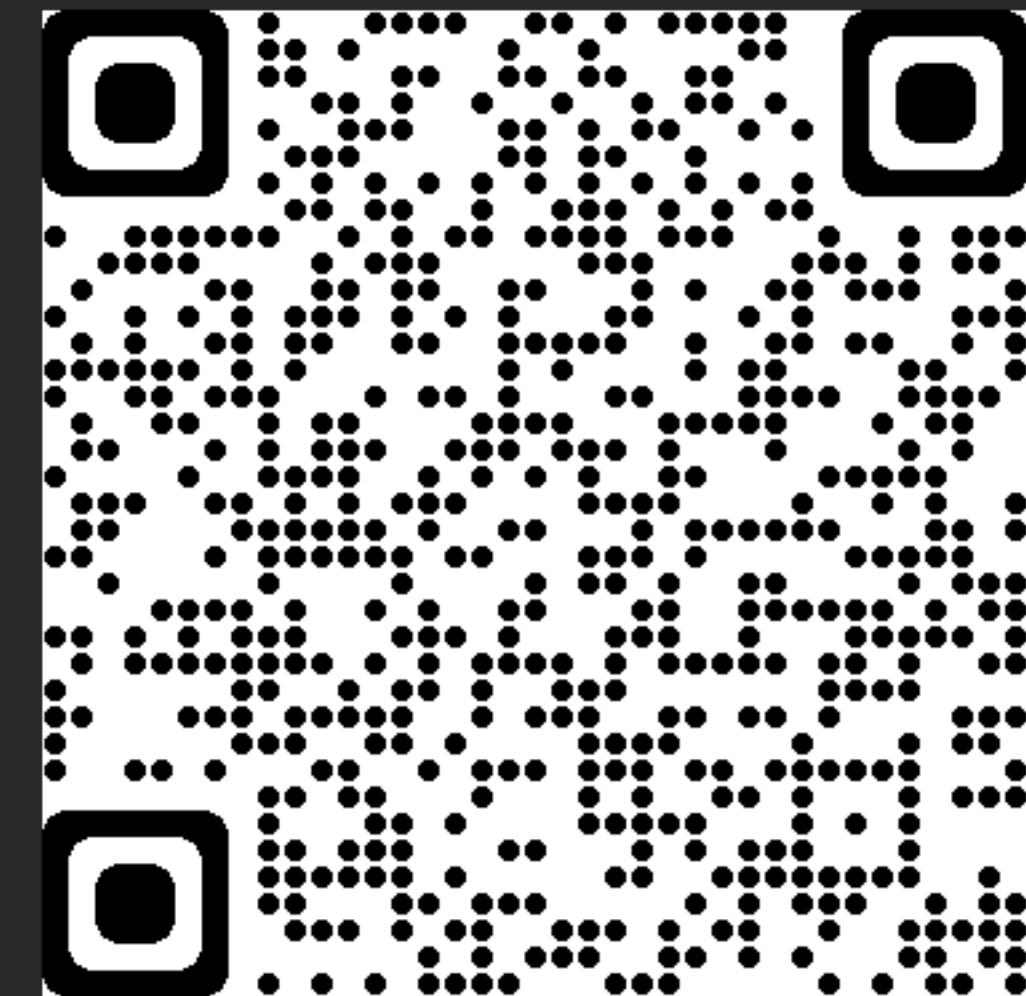


# The Go Programming Language Specification

Version of March 10, 2022

## Table of Contents

Introduction	Index expressions
Notation	Slice expressions
Source code representation	Type assertions
Characters	Calls
Letters and digits	Passing arguments to ... parameters
Lexical elements	Instantiations
Comments	Type inference
Tokens	Operators
Semicolons	Arithmetic operators
Identifiers	Comparison operators
Keywords	Logical operators
Operators and punctuation	Address operators
Integer literals	Receive operator
Floating-point literals	Conversions
Imaginary literals	Constant expressions
Rune literals	Order of evaluation
String literals	Statements
Constants	Terminating statements
Variables	Empty statements
Types	Labeled statements
Boolean types	Expression statements
Numeric types	Send statements
String types	IncDec statements
Array types	Assignments
Slice types	If statements
Struct types	Switch statements
Pointer types	For statements
Function types	Go statements
Interface types	Select statements
Map types	Return statements
Channel types	Break statements



<https://go.dev/ref/spec>

```
package main
```

```
func main() {  
    var x string = "hi"  
    x = 4  
}
```

```
main.go:5:6: cannot use 4 (untyped int  
constant) as string value in assignment
```













# What Static Analysis Can Prove

- ✦ Not releasing some resources
- ✦ Making typos the compiler can't prove
- ✦ Invalid constants (time format, URL, regexes)
- ✦ Prevent a wide range of preventable crashes



Gopher image by Renee French,  
licensed under Creative Commons  
3.0 Attributions license.



```
func doHTTP() error {
    resp, err := http.Get("https://christine.website/.within/health")
    if err != nil {
        return err
    }

    io.Copy(os.Stdout, resp.Body)
    return nil
}
```

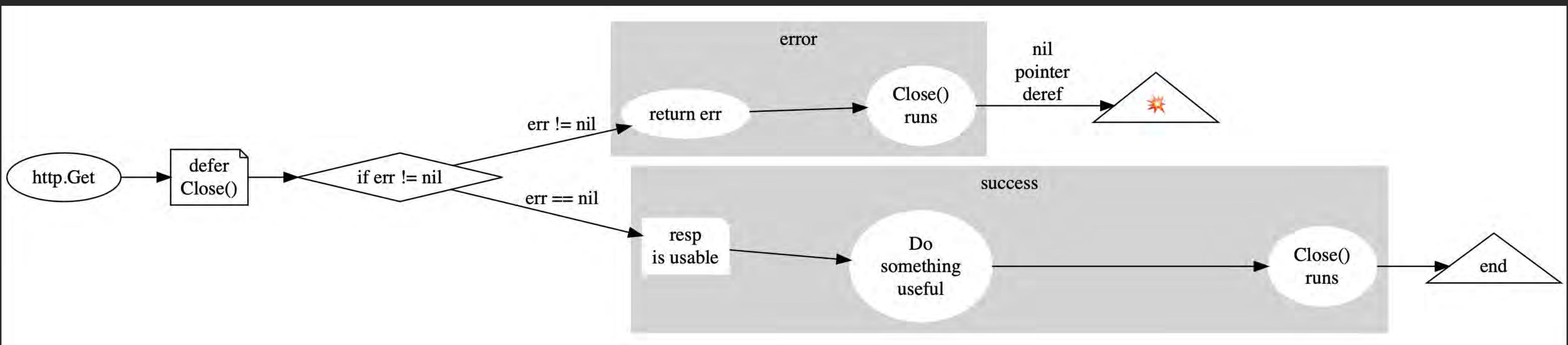
```
func doHTTP() error {
    resp, err := http.Get("https://christine.website/.within/health")
    if err != nil {
        return err
    }

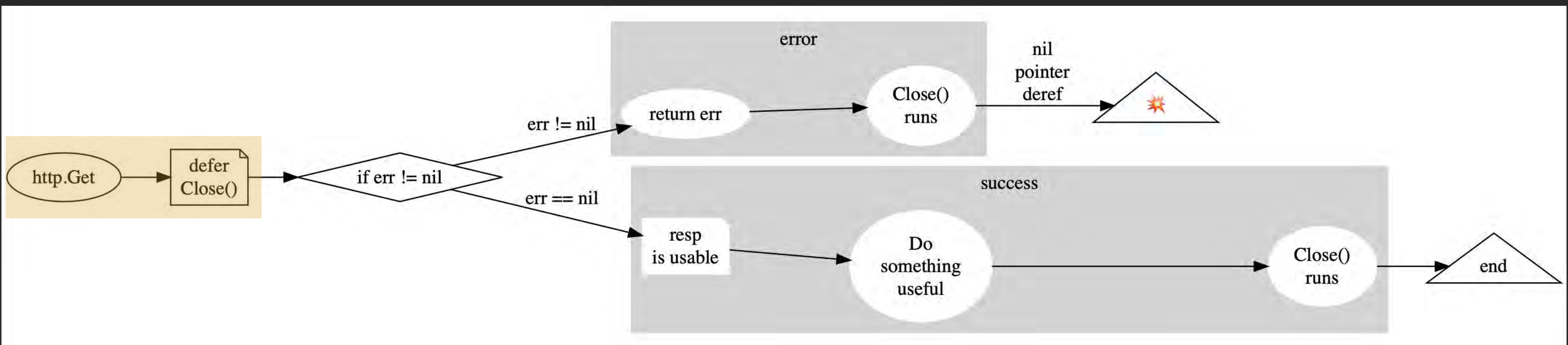
    io.Copy(os.Stdout, resp.Body)
    return nil
}
```

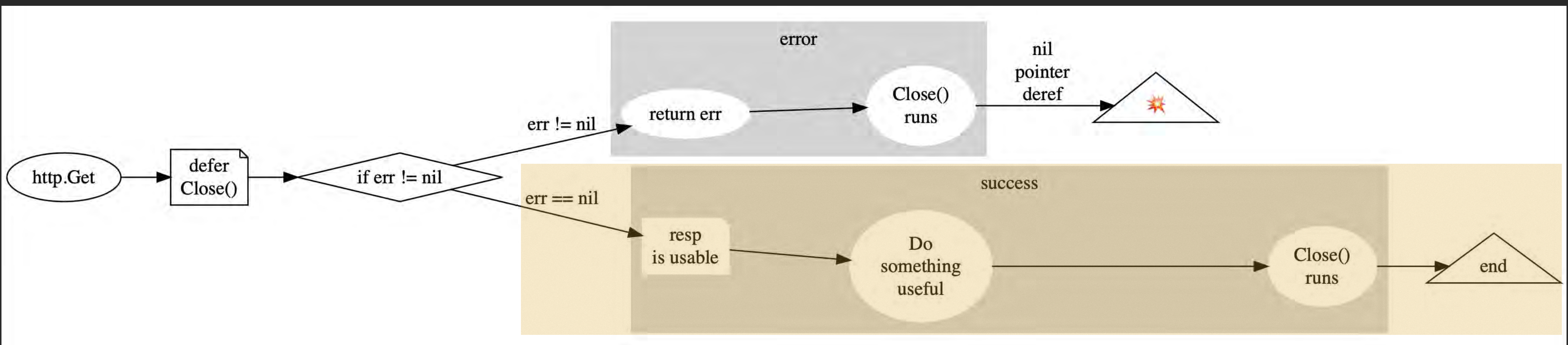
```
func doHTTP() error {
    resp, err := http.Get("https://christine.website/.within/health")
    defer resp.Body.Close()
    if err != nil {
        return err
    }

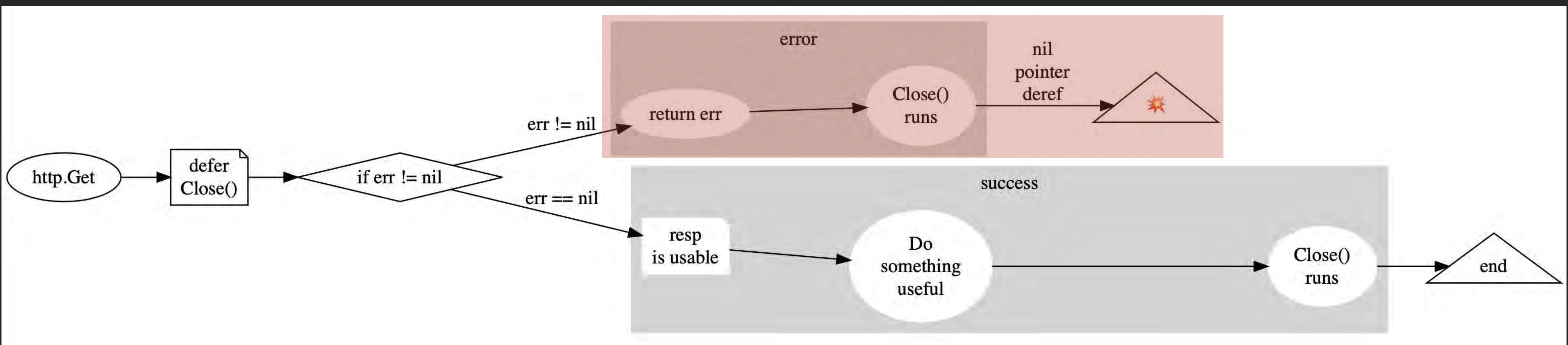
    io.Copy(os.Stdout, resp.Body)
    return nil
}
```

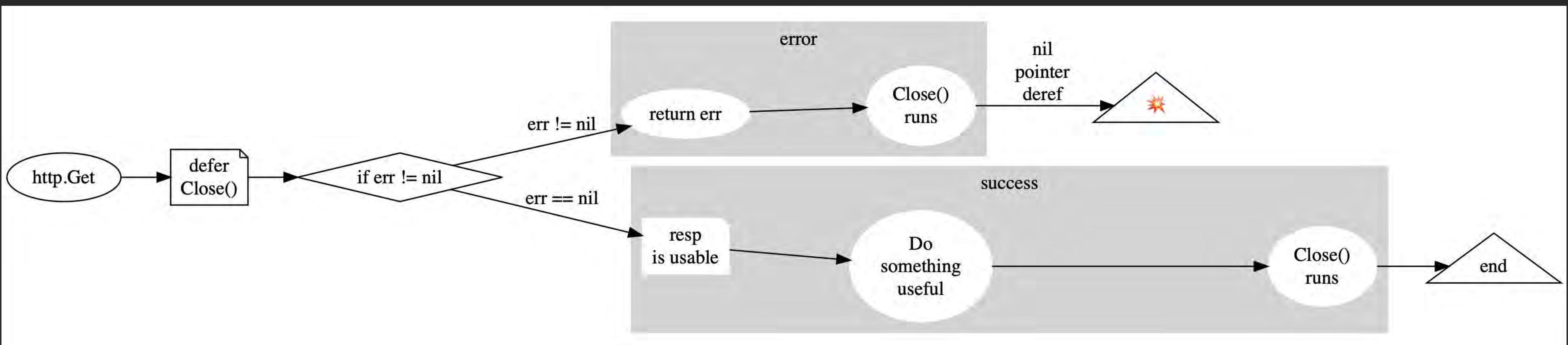












```
$ go vet httpget.go
# command-line-arguments
./httpget.go:16:8: using resp before checking for errors
```

httpget.go — httpget\_fixed.go

---

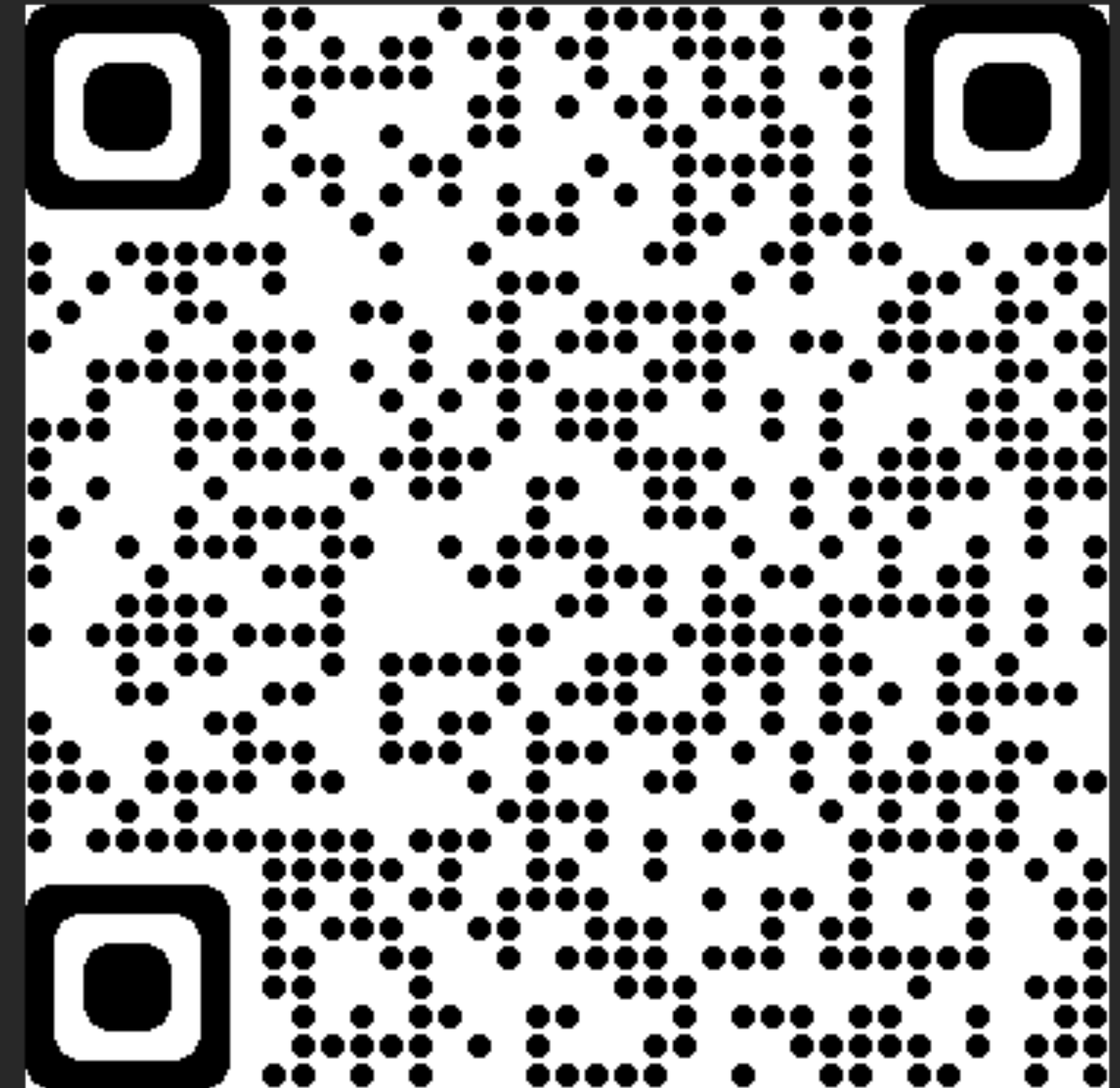
```
13: func main() {
```

```
func doHTTP() error {  
    resp, err := http.Get("https://christine.website/.within/health")  
    defer resp.Body.Close()  
    if err != nil {  
        return err  
    }  
    defer resp.Body.Close()  
  
    // do something useful  
    return nil  
}
```



To list the available checks, run "go tool vet help":

```
asmdecl    report mismatches between assembly files and Go declarations
assign     check for useless assignments
atomic     check for common mistakes using the sync/atomic package
bools      check for common mistakes involving boolean operators
buildtag   check that +build tags are well-formed and correctly located
cgocall    detect some violations of the cgo pointer passing rules
composites check for unkeyed composite literals
copylocks  check for locks erroneously passed by value
httpresponse check for mistakes using HTTP responses
loopclosure check references to loop variables from within nested functions
lostcancel check cancel func returned by context.WithCancel is called
nilfunc    check for useless comparisons between functions and nil
printf     check consistency of Printf format strings and arguments
shift      check for shifts that equal or exceed the width of the integer
stdmethods check signature of methods of well-known interfaces
structtag  check that struct field tags conform to reflect.StructTag.Get
tests      check for common mistaken usages of tests and examples
unmarshal  report passing non-pointer or non-interface values to unmarshal
unreachable check for unreachable code
unsafeptr  check for invalid conversions of uintptr to unsafe.Pointer
unusedresult check for unused results of calls to some functions
```



<https://pkg.go.dev/cmd/vet>







Gopher image by Renee French,  
licensed under Creative Commons  
3.0 Attributions license.

# GitHub Actions

Running Staticcheck in GitHub Actions

We publish [our own action](#) for [GitHub Actions](#), which makes it very simple to run Staticcheck in CI on GitHub.

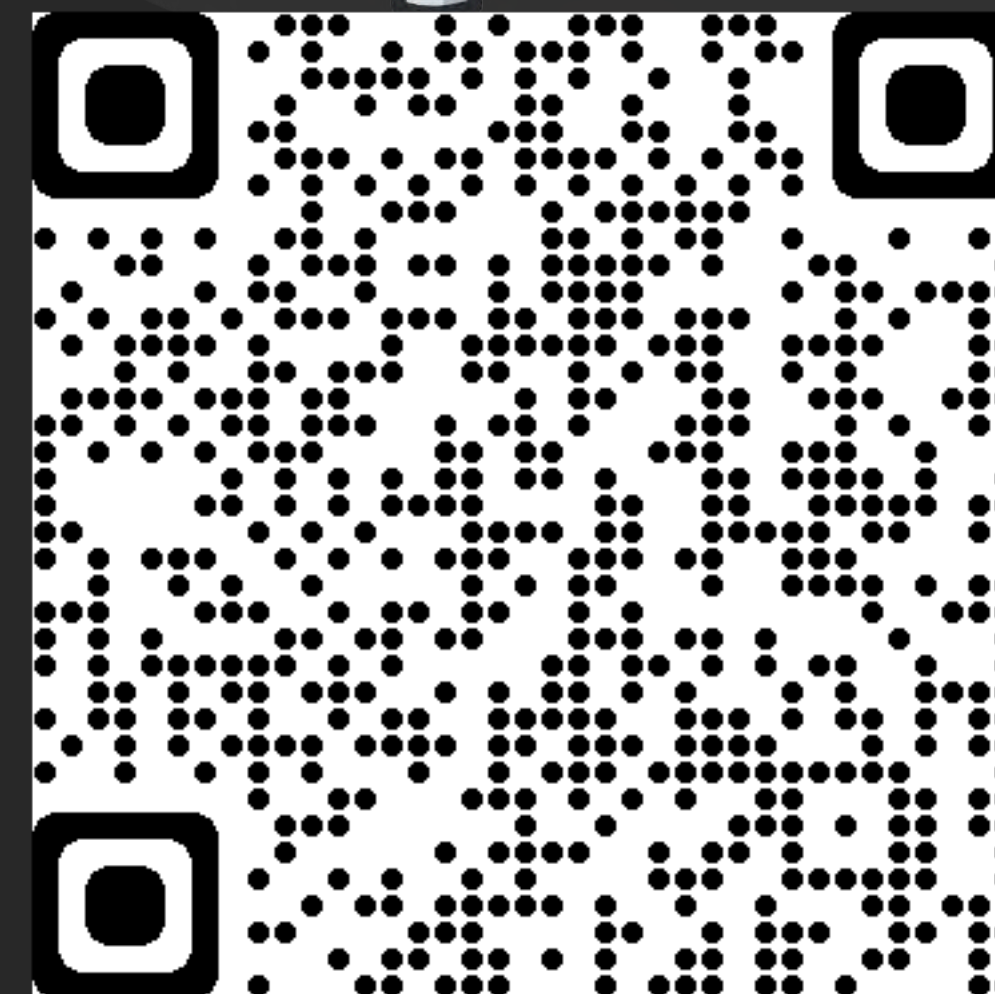
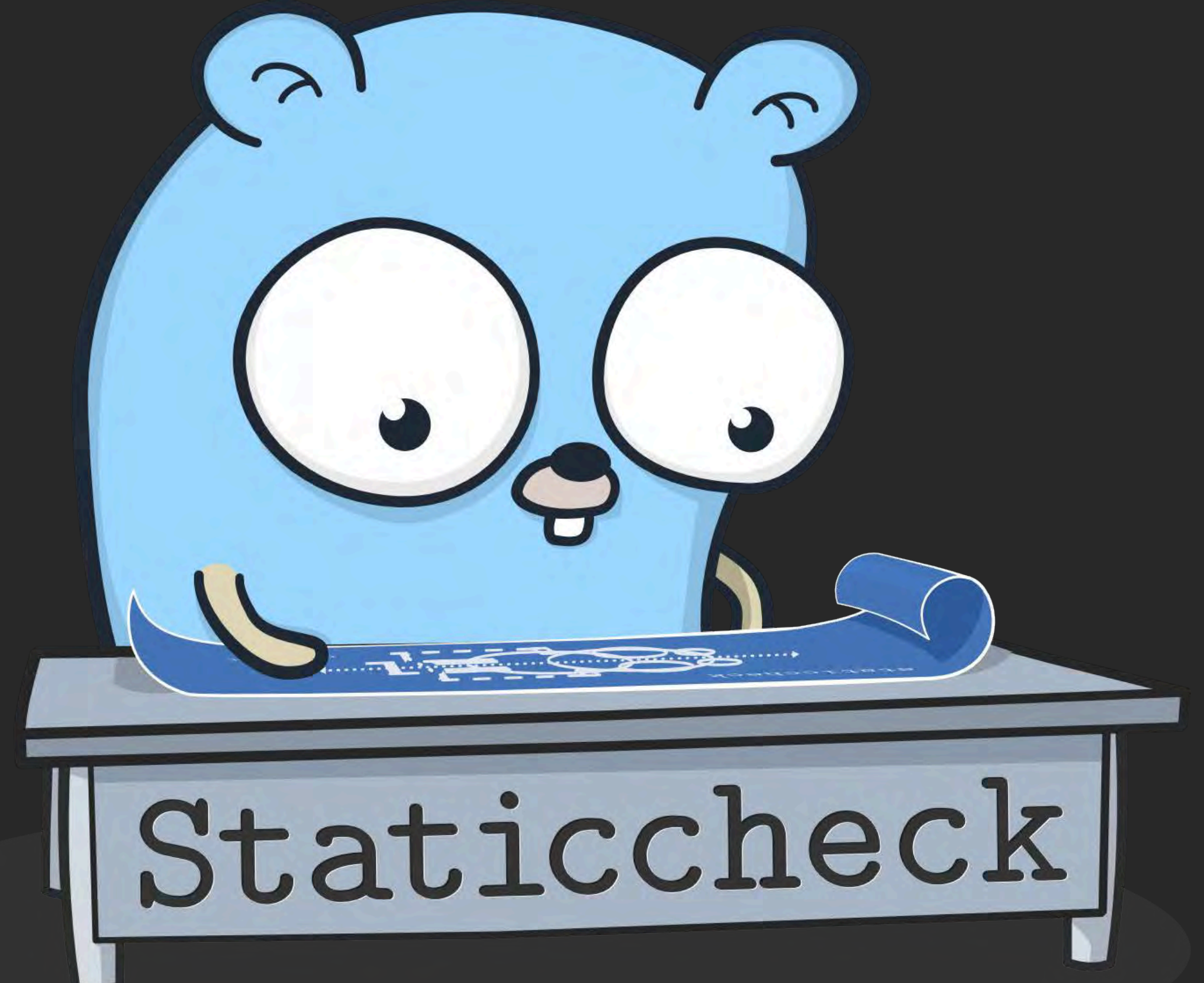
## Examples

At its simplest, just add `dominikh/staticcheck-action` as a step in your existing workflow. A minimal workflow might look like this:

```
name: "CI"
on: ["push", "pull_request"]

jobs:
  ci:
    name: "Run CI"
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v1
        with:
          fetch-depth: 1
      - uses: dominikh/staticcheck-action@v1.2.0
        with:
          version: "2022.1.1"
```

<https://staticcheck.io/docs/running-staticcheck/ci/github-actions/>



Gopher image by Renee French,  
licensed under Creative Commons  
3.0 Attributions license.

```
if x, ok := x.(int); ok {  
    // ...  
} else {  
    fmt.Printf("unexpected type %T\n", x)  
}
```

```
if err := doSomething(); err != nil {  
    log.Fatal(err)  
}  
// err isn't valid here
```

```
if err := doSomething(); err != nil {  
    log.Fatal(err)  
}  
// err isn't valid here
```



```
{  
    err := doSomething()  
    if err != nil {  
        log.Fatal(err)  
    }  
}  
// err isn't valid here
```

```
if x, ok := x.(int); ok {  
    // ...  
} else {  
    fmt.Printf("unexpected type %T\n", x)  
}
```

```
if x, ok := x.(int); ok {  
    // ...  
} else {  
    fmt.Printf("unexpected type %T\n", x)  
}
```



unexpected type int

```
if xInt, ok := x.(int); ok {  
    // ...  
} else {  
    fmt.Printf("unexpected type %T\n", x)  
}
```

```
type Failure struct{ Reason string }

func (err *Failure) Error() string { return err.Reason }

func failed() bool { return false }

func doWork() error {
    var err *Failure
    if failed() {
        err = &Failure{"oh no"}
    }
    return err
}

func TestDoWork(t *testing.T) {
    if err := doWork(); err != nil {
        t.Log(err.Error())
    }
}
```

```
type Failure struct{ Reason string }
```

```
func (err *Failure) Error() string { return err.Reason }
```

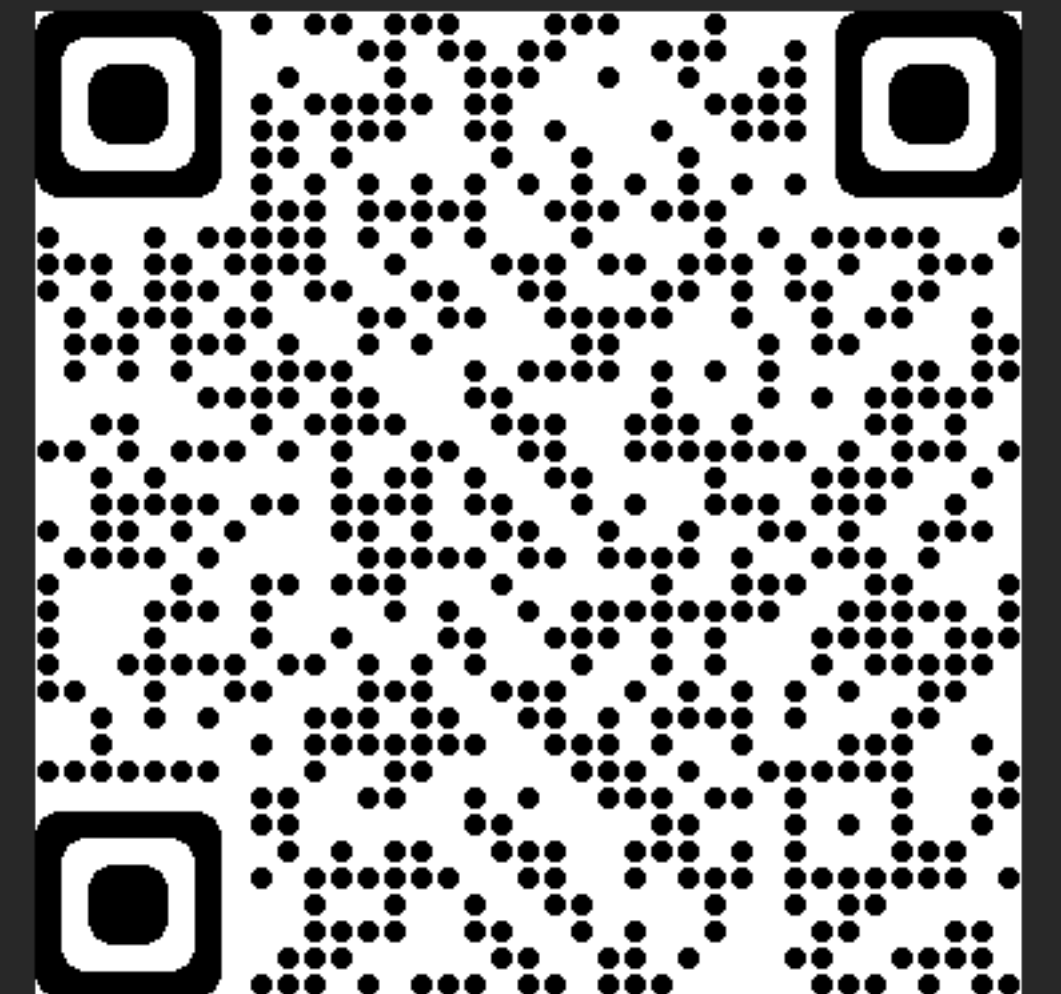
```
func failed() bool { return false }
```

```
func doWork() error {  
    var err *Failure  
    if failed() {  
        err = &Failure{"oh no"}  
    }  
    return err  
}
```

```
func TestDoWork(t *testing.T) {  
    if err := doWork(); err != nil {  
        t.Log(err.Error())  
    }  
}
```

```
type error interface {  
    Error() string  
}
```

<https://pkg.go.dev/builtin#error>



```
type Failure struct{ Reason string }

func (err *Failure) Error() string { return err.Reason }

func failed() bool { return false }

func doWork() error {
    var err *Failure
    if failed() {
        err = &Failure{"oh no"}
    }
    return err
}

func TestDoWork(t *testing.T) {
    if err := doWork(); err != nil {
        t.Log(err.Error())
    }
}
```

```
type Failure struct{ Reason string }
```

```
func (err *Failure) Error() string { return err.Reason }
```

```
func failed() bool { return false }
```

```
func doWork() error {  
    var err *Failure  
    if failed() {  
        err = &Failure{"oh no"}  
    }  
    return err  
}
```

```
func TestDoWork(t *testing.T) {  
    if err := doWork(); err != nil {  
        t.Log(err.Error())  
    }  
}
```

```
panic: runtime error: invalid memory address or nil pointer dereference [recovered]
  panic: runtime error: invalid memory address or nil pointer dereference
[signal SIGSEGV: segmentation violation code=0x1 addr=0x0 pc=0x4ac960]
```





interface error value



```
type Failure struct{ Reason string }

func (err *Failure) Error() string { return err.Reason }

func failed() bool { return false }

func doWork() error {
    var err *Failure
    if failed() {
        err = &Failure{"oh no"}
    }
    return err
}

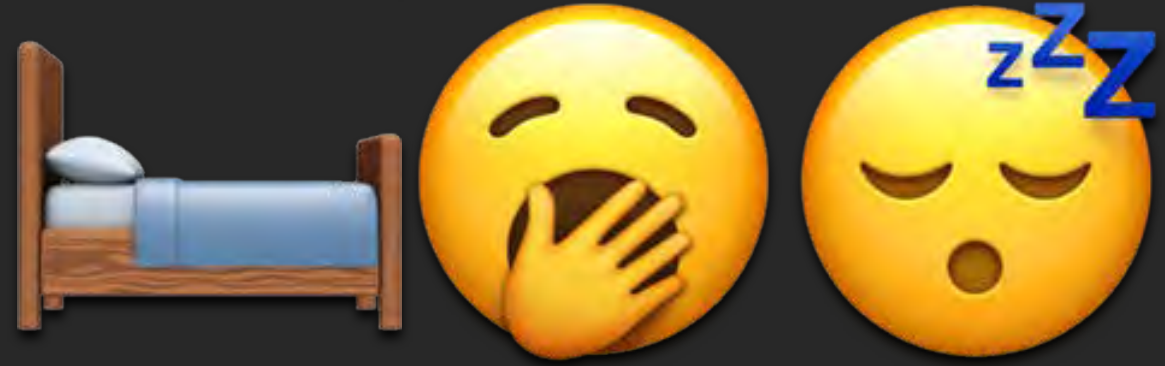
func TestDoWork(t *testing.T) {
    if err := doWork(); err != nil {
        t.Log(err.Error())
    }
}
```

```
errorbomb.go:11:6: doWork never returns a nil  
interface value
```

```
func doWork() error {  
    if failed() {  
        return &Failure{"oh no"}  
    }  
    return nil  
}
```



```
func doWork() error {  
    if failed() {  
        return &Failure{"oh no"}  
    }  
    return nil  
}
```











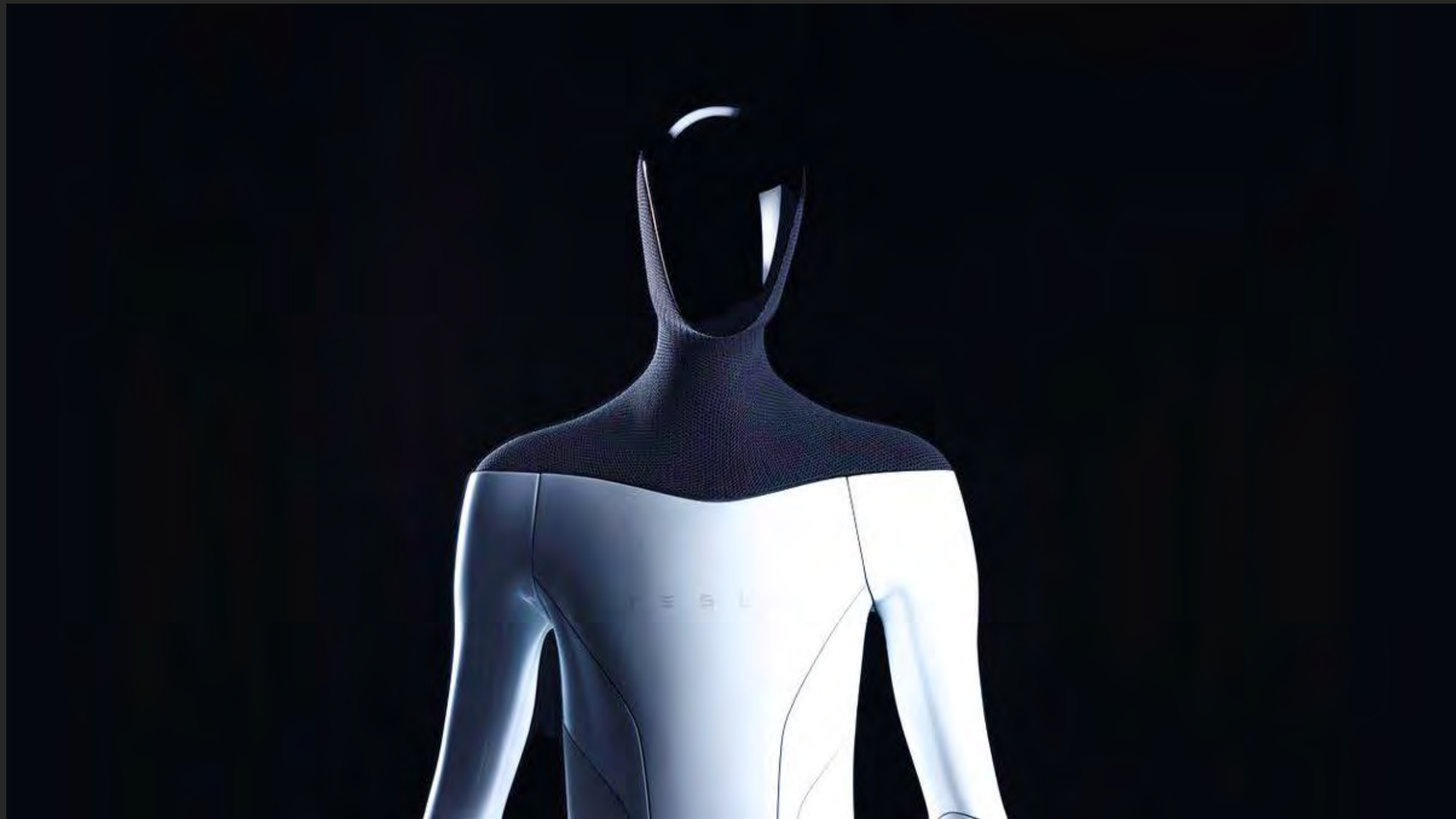


**WISHING YOU REST AND  
RELAXATION**













**Dominik Honnef**

dominikh

Germany

Hi,

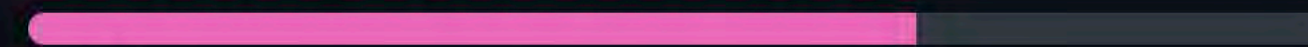
I'm Dominik and I have dedicated a large part of my life to open source. In particular, I love writing code that helps other programmers do their job.

While I focused on writing open source Ruby libraries in my teenage years, I have shifted to Go development many years ago. I am the author of [staticcheck](#) – one of the most widely used bug finding tools for Go – and [go-mode](#) – the Go mode for Emacs. GopherJS users may know me from the `js/dom` package, the original GopherJS bindings for the DOM.

In addition to my main projects I have also worked on dozens of smaller libraries and tools, as well as contributed changes to many open source projects that aren't my own, including the Go project itself.

It is my hope to one day make open source be sustainable for me so that I don't have to pick up paid jobs that take time away from my projects. You will have my eternal gratitude if you decide to help me with that goal.

68% towards \$5,000 per month goal



You and 75 others sponsor this goal

Sponsoring as Xe

\$1 a month

Manage

For \$1 I will offer you my undying gratitude. Any amount counts and brings us closer to more sustainable open source!

Select a new tier

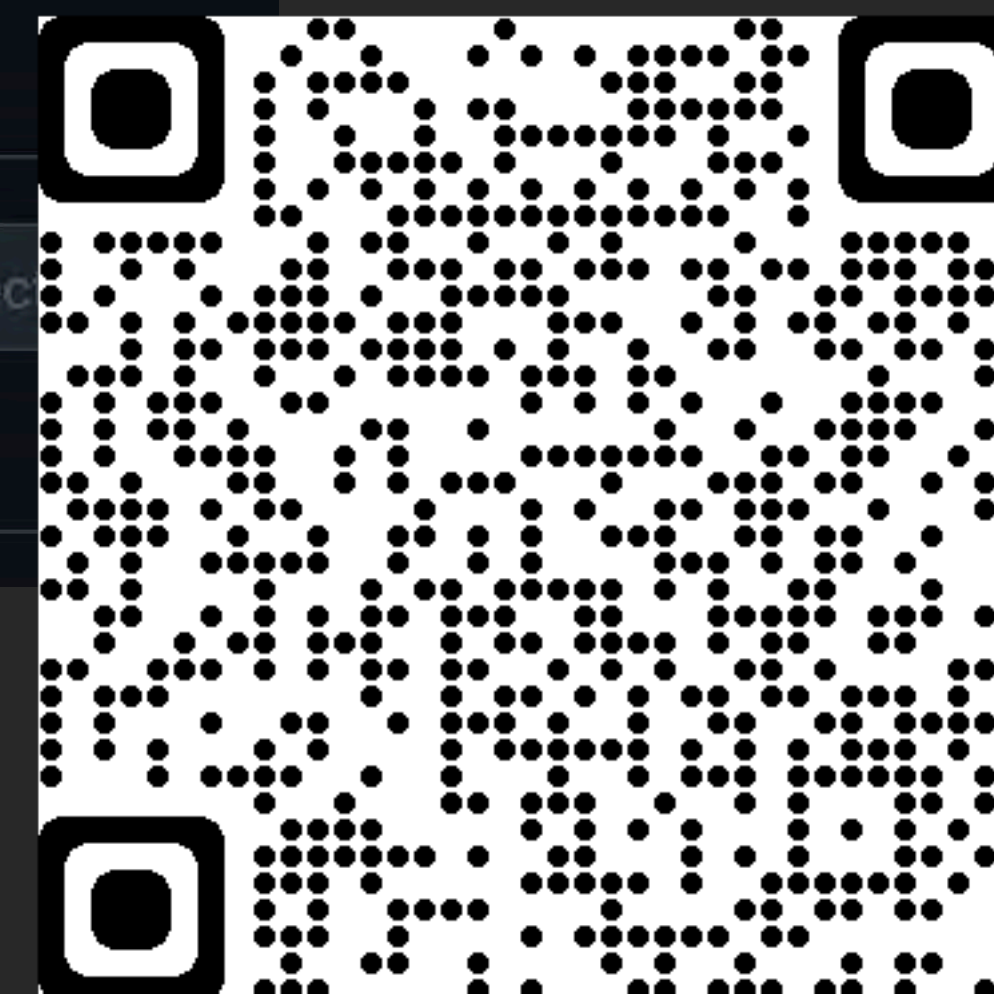
Add a one-time payment

\$

one time

Select

Choose a custom amount.



<https://github.com/sponsors/dominikh>





# GReeTZ



- ✦ Selicre
- ✦ Artzora
- ✦ dominikh
- ✦ catzkorn
- ✦ dgentry
- ✦ apenwarr
- ✦ jbrandhorst

- ✦ HakerTeam
- ✦ Scootaloose
- ✦ crawshaw
- ✦ lauratellsjokes
- ✦ markogilbee
- ✦ sailorfrag
- ✦ artemis



[code42sre2022@xeserv.us](mailto:code42sre2022@xeserv.us)



<https://christine.website>



<https://twitter.com/theprincessxena>