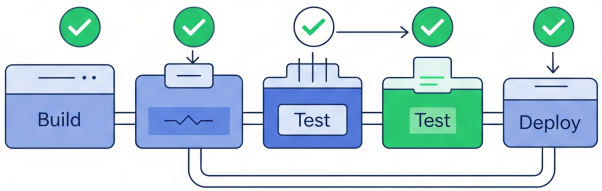
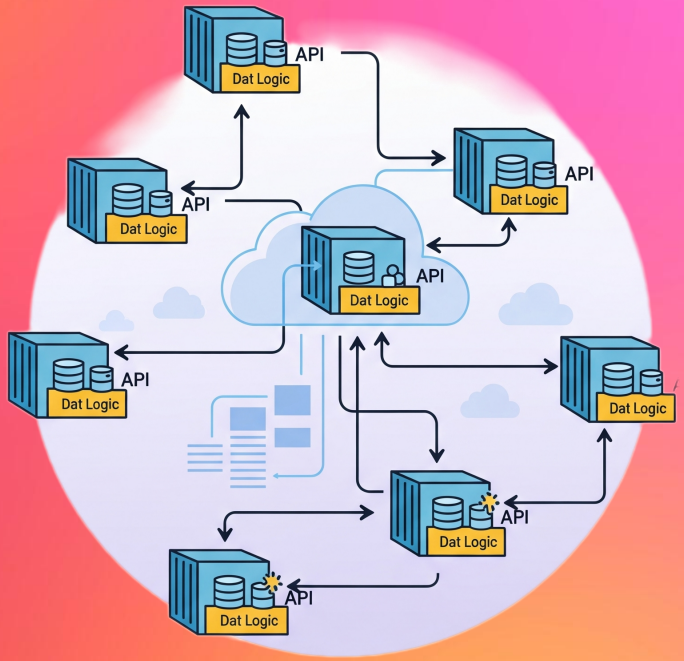
Nil today, outage tomorrow

Mukul Mantosh
Developer Advocate

JETBRAINS

Logos by Maria Letta | https://github.com/MariaLetta/free-gophers-pack


With All checks passing
Build
Test
Test
Deploy


API
Dat Logic
API
Dat Logic
API
Dat Logic
API
Dat Logic
API
Dat Logic
Dat Logic
API
Dat Logic
API
Dat Logic

Staging Lied. Production Died.

Manager “Why is the app down?”
Developer “Something was nil.
Manager “What was nil?”
Developer “Exactly, line number 27”

Looks familiar?

```
> <4 go setup calls>
  panic: runtime error: invalid memory address or nil pointer dereference
  [signal SIGSEGV: segmentation violation code=0x2 addr=0x8 pc=0x104210c88]

  goroutine 1 [running]:
  main.SafeCreateUser(0x14000098f38?, {0x104211a50?, 0x0?})
      /Users/mukulmantosh/GolandProjects/nil_dereference/main.go:12 +0x78
  main.main()
      /Users/mukulmantosh/GolandProjects/nil_dereference/main.go:17 +0x2c

  Process finished with the exit code 2
```

Something doesn’t seem quite right here...

```
func RegisterUser(name string) {  1 usage  new *
    user := CreateUser(name)
    fmt.Printf( format: "Name: %s, ID: %d\n", user.Name, user.ID)
}
```

Time to investigate...

```
func CreateUser(name string) *User {  1 usage  new *
    if !validName(name) {
        // ❌ Returns nil without signaling failure
        return nil
    }
    return &User{ID: GenerateUniqueIntID(), Name: name}
}
```

Time to refactor

```go
func RegisterUser(name string, email string) (*User, error) {  1 usage
    user, err := CreateUser(name, email)

    if err != nil {
        return nil, fmt.Errorf( format: "create user: %w", err)
    }
    fmt.Printf( format: "User ID: %d, Name: %s\n", user.ID, user.Name)
    return user, nil
}
```

```go
func CreateUser(name string, email string) (*User, error) {
    if !IsValidEmail(email) {
        return nil, errors.New( text: "the email is invalid")
    }
    if !IsValidName(name) {
        return nil, nil
    }
    return &User{ID: GenerateUniqueIntID(), Name: name}, nil
}
```

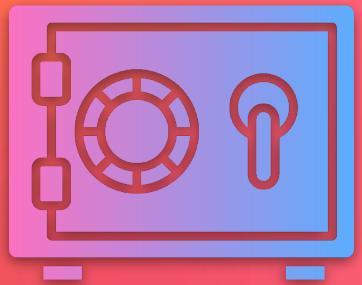
```
if !IsValidName(name) {
    return nil, errors.New( text: "invalid name")
}
```

the silence of nil is deceptive.

Here I come...

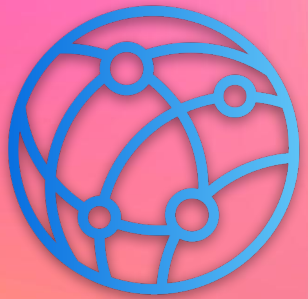
Global

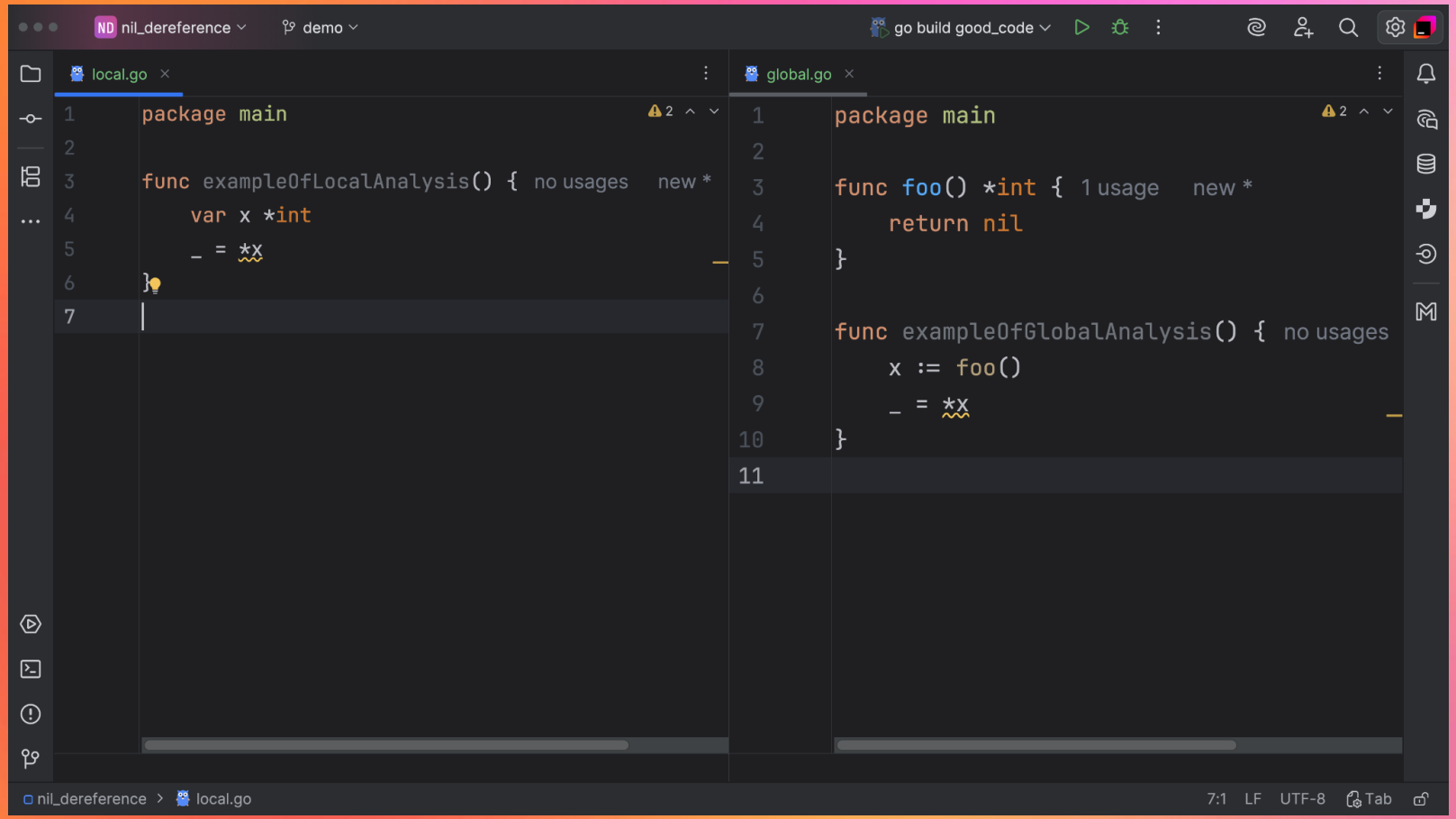
Intraprocedural



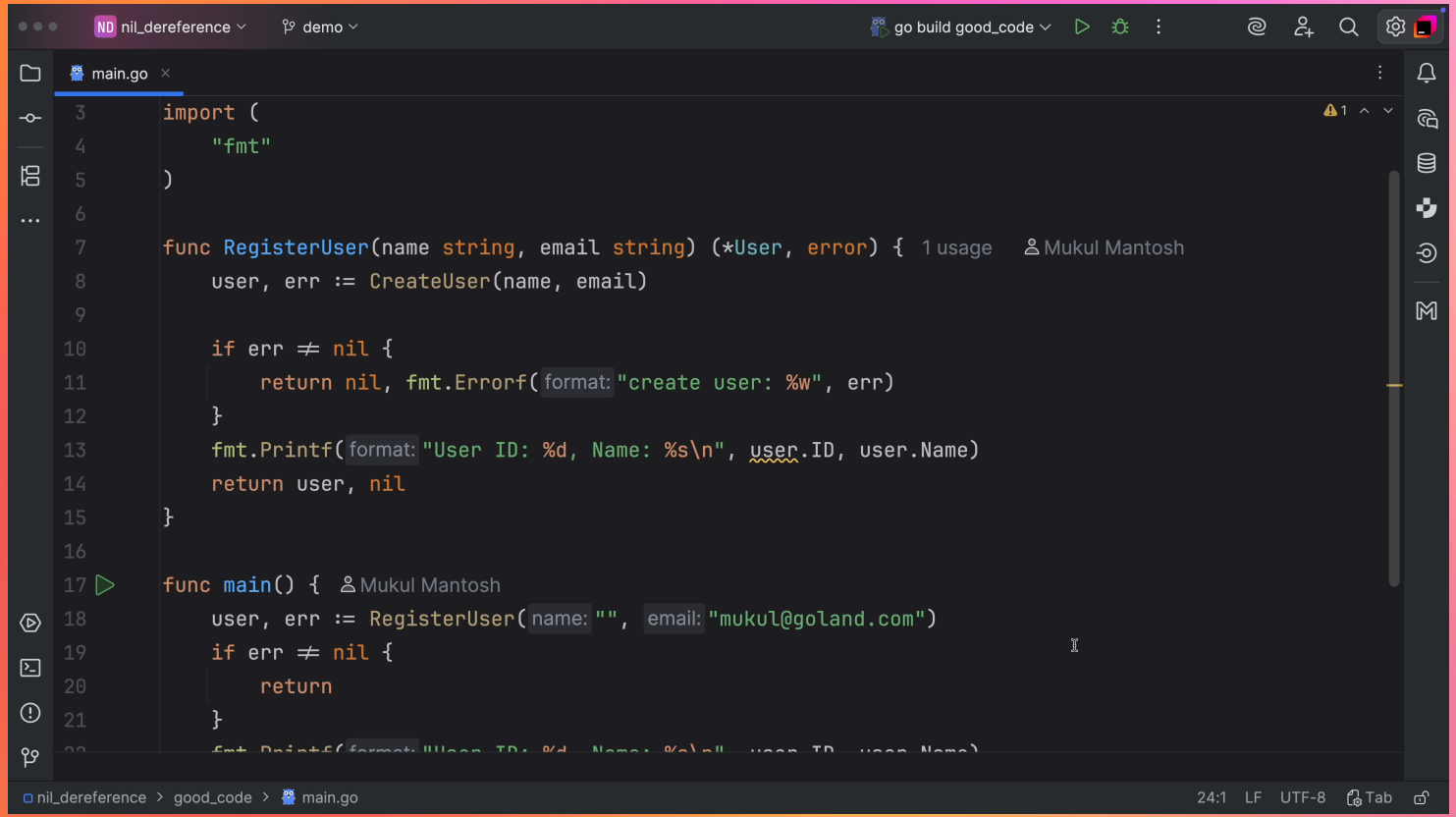
Local

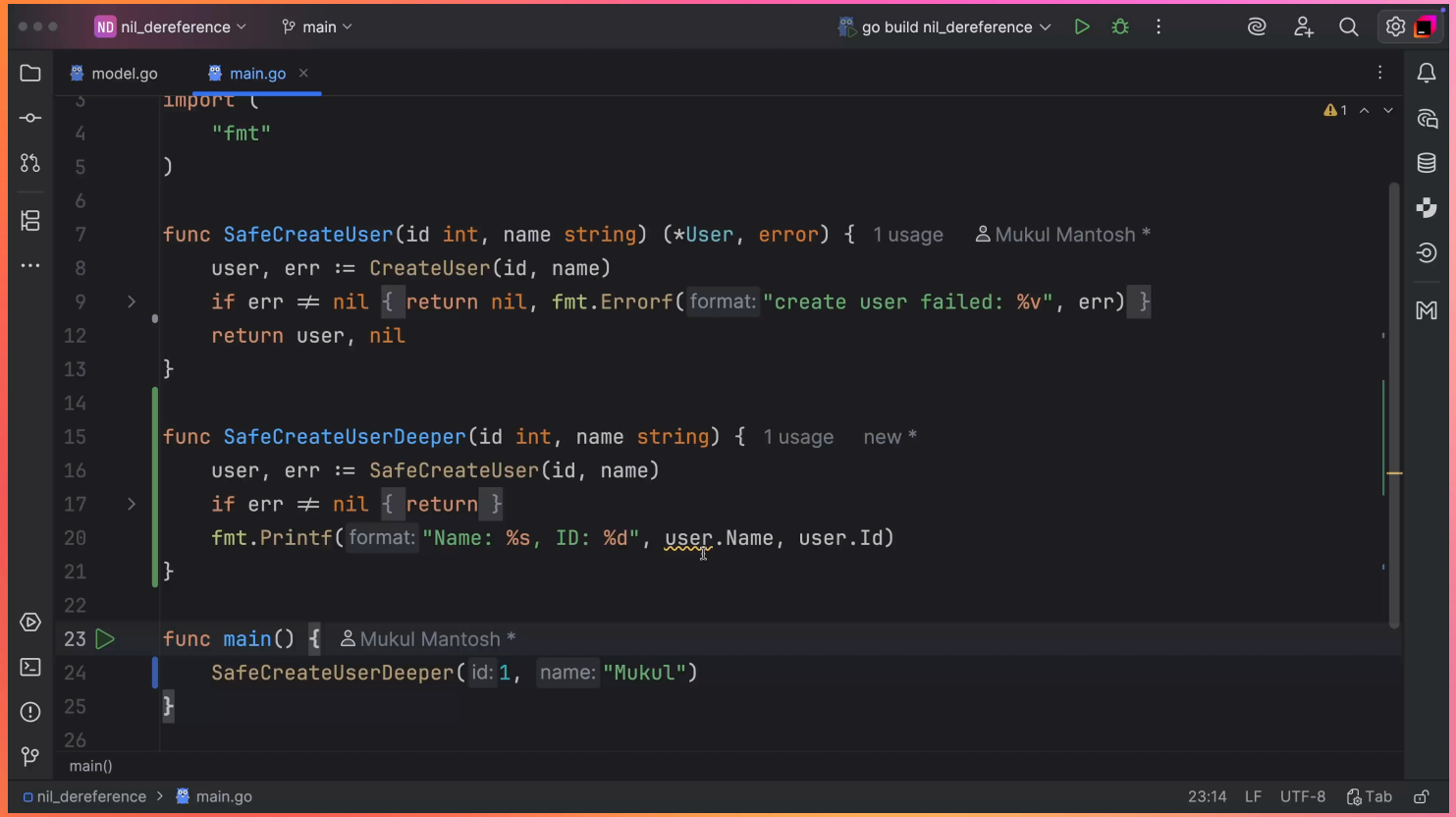
Interprocedural




nil_dereference
demo
go build good_code
local.go
package main
func exampleOfLocalAnalysis() { no usages new *
var x *int
_ = *x
}
global.go
package main
func foo() *int { 1 usage new *
return nil
}
func exampleOfGlobalAnalysis() { no usages
x := foo()
_ = *x
}
nil_dereference
local.go
7:1
LF
UTF-8
Tab

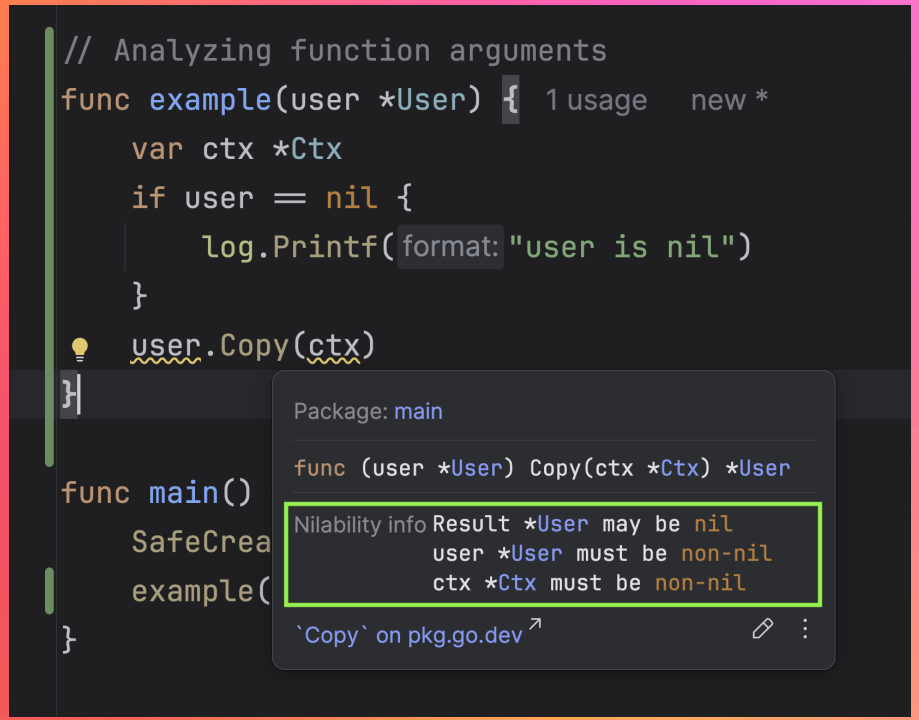

You didn’t show any demo, right?
Programming
FWP


nil_dereference
demo
go build good_code
main.go
import (
"fmt"
)
func RegisterUser(name string, email string) (*User, error) { 1 usage Mukul Mantosh
user, err := CreateUser(name, email)
if err != nil {
return nil, fmt.Errorf(format: "create user: %w", err)
}
fmt.Printf(format: "User ID: %d, Name: %s\n", user.ID, user.Name)
return user, nil
}
func main() { Mukul Mantosh
user, err := RegisterUser(name: "", email: "mukul@goland.com")
if err != nil {
return
}
nil_dereference
good_code
main.go
24:1
LF
UTF-8
Tab


nil_dereference
main
go build nil_dereference
model.go
main.go
import (
"fmt"
)
func SafeCreateUser(id int, name string) (*User, error) { 1 usage Mukul Mantosh *
user, err := CreateUser(id, name)
if err != nil { return nil, fmt.Errorf(format: "create user failed: %v", err) }
return user, nil
}
func SafeCreateUserDeeper(id int, name string) { 1 usage new *
user, err := SafeCreateUser(id, name)
if err != nil { return }
fmt.Printf(format: "Name: %s, ID: %d", user.Name, user.Id)
}
func main() { Mukul Mantosh *
SafeCreateUserDeeper(id: 1, name: "Mukul")
}
main()
nil_dereference
main.go
23:14
LF
UTF-8
Tab


8 func process(user *common.User) { 1 usage Arseniy Terekhov
9 if user == nil {
10 log.Println(v...: "user is nil")
11 }
12 copiedUser := user.Copy(ctx: nil)
13 handleUser(copiedUser)
14 }
Potential nil dereference when passing 'nil' to 'Copy'
Line 12: 'nil' is passed to 'user.Copy'

Quick Docs


// Analyzing function arguments
func example(user *User) { 1 usage new *
var ctx *Ctx
if user == nil {
log.Printf(format: "user is nil")
}
user.Copy(ctx)
}
func main()
SafeCrea
example(
}
Package: main
func (user *User) Copy(ctx *Ctx) *User
Nilability info Result *User may be nil
user *User must be non-nil
ctx *Ctx must be non-nil
`Copy` on pkg.go.dev

Doesn't it resemble NilAway?

Avengers... assemble

Thank you!

mukulmantosh

MantoshMukul

mukul-mantosh