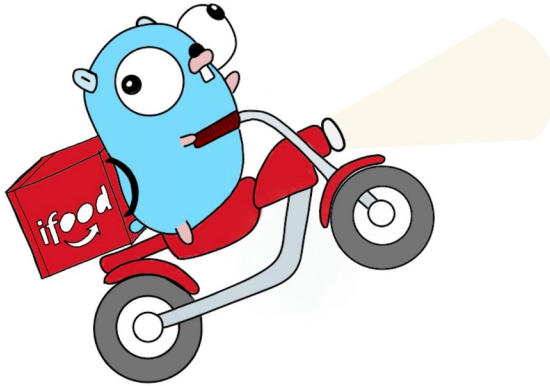
Por que criar bibliotecas internas?

Um pouco sobre como bibliotecas internas fazem a diferença no dia-a-dia do iFood

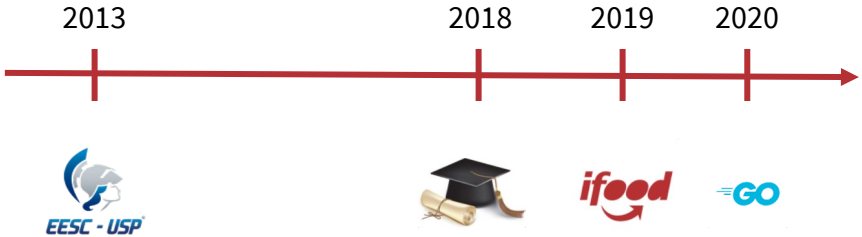


GUSTAVO OKUYAMA

Gustavo Okuyama

Sobre

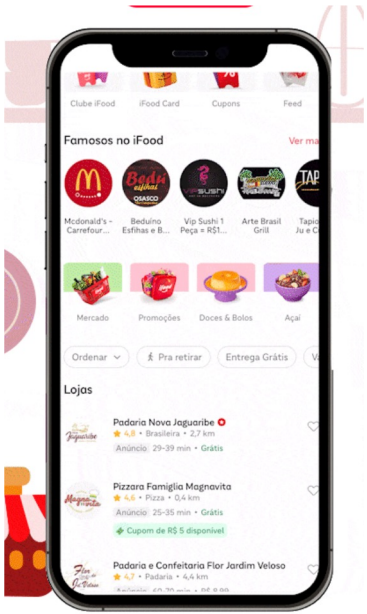
- Engenheiro de Software
- iFood - Conteúdo
- 28 anos
- São Carlos - SP
- 5 anos TI



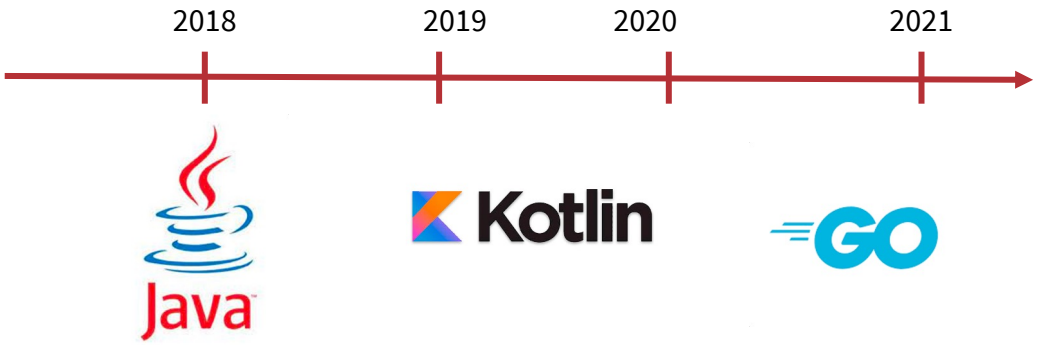
O que é o iFood?

SOMOS
uma empresa
BRASILEIRA
de tecnologia
que CONECTA

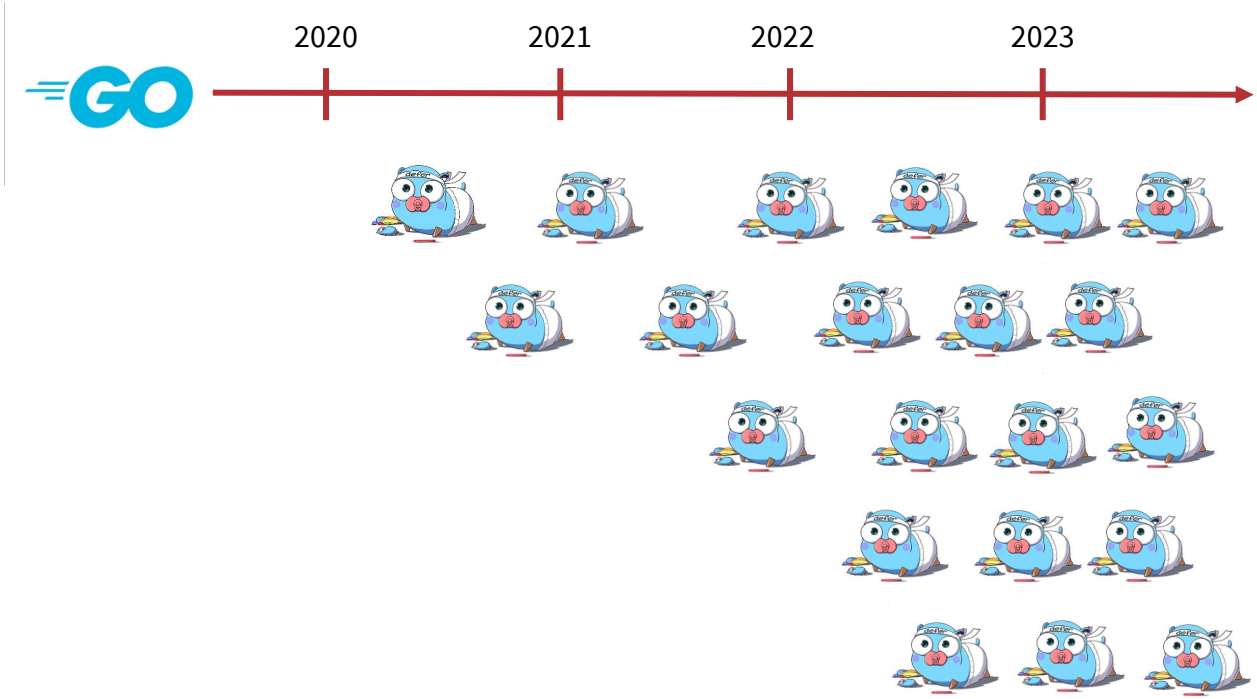
Consumer Discovery



Linha do Tempo

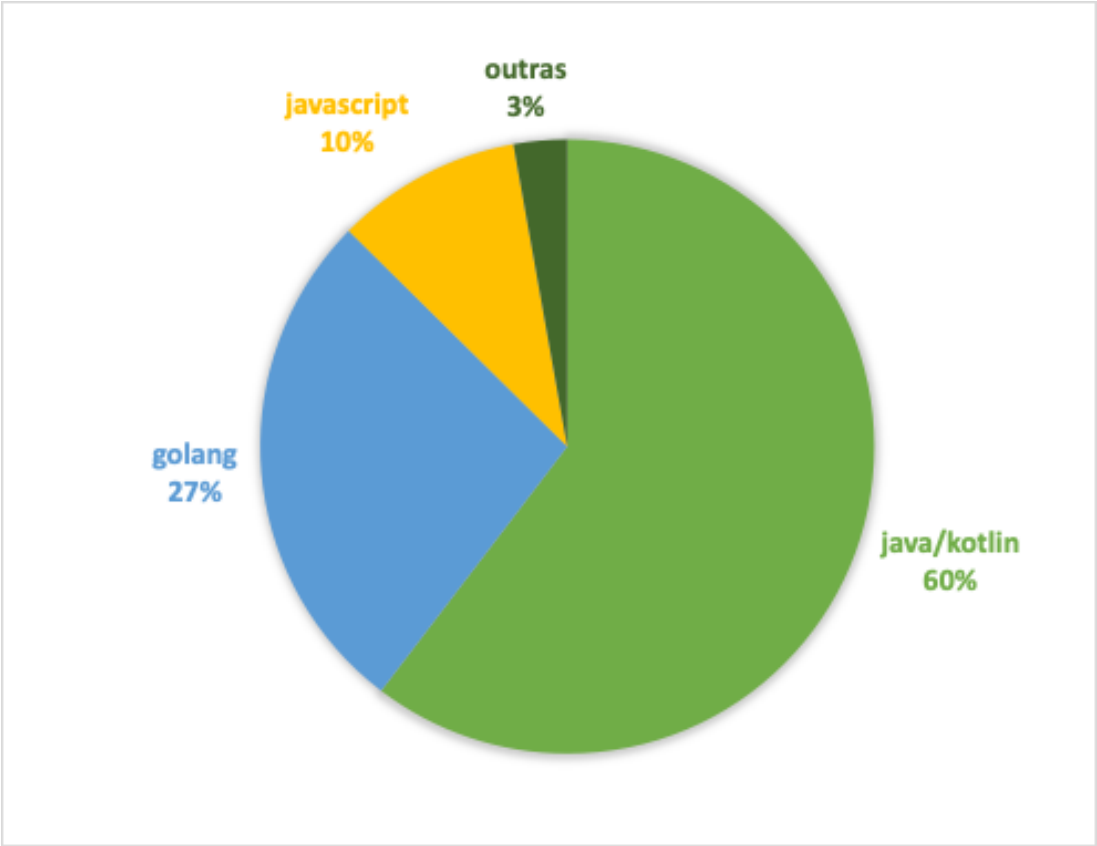


Linha do Tempo



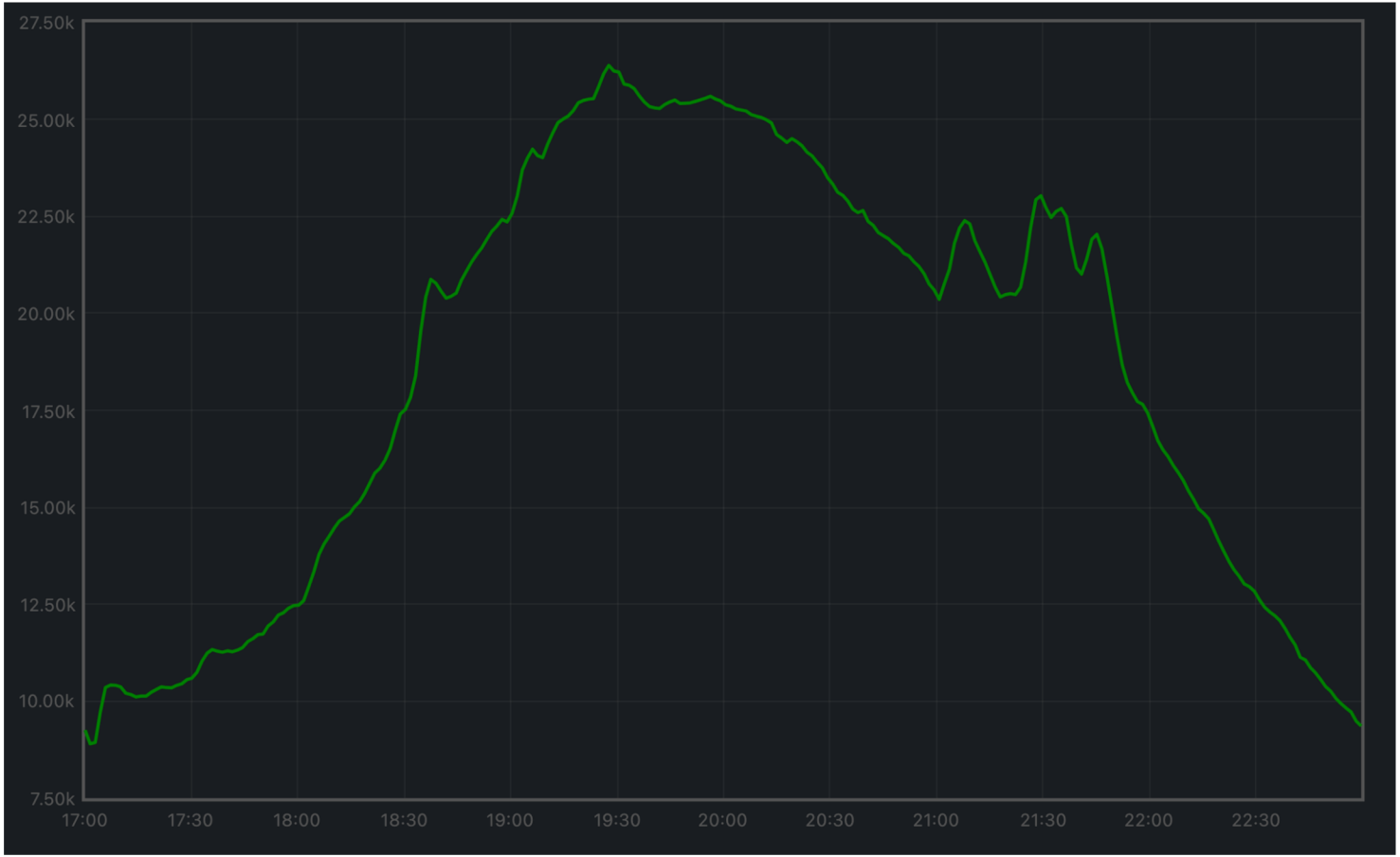
Golang no iFood

COMPARATIVO ENTRE LINGUAGENS UTILIZADAS NO IFOOD (SET./2023)



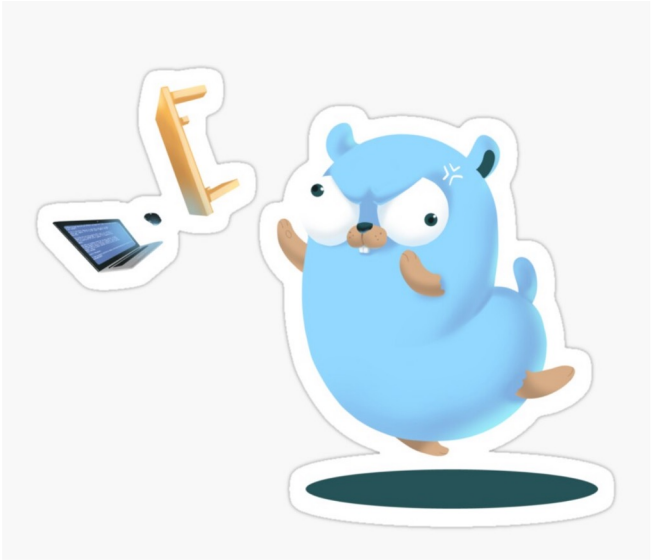
REQ/S EM UM ÚNICO SERVIÇO GOLANG EM UM JANTAR

Requests/s



Dores

1. Muito Código Copiado
2. Evolução do código copiado
3. Standards (padrões)



Logs

APM (Application Performance Monitoring)

Métricas

HTTP Clients

Circuit Breakers

Config

Kafka

Headers e Metadata

gRPC

Feature Flags

Primeiros Passos

- Entender como é feita a criação de uma biblioteca

- Criação de uma biblioteca pública

Como criar uma biblioteca

É necessário:

- repositório git
- 1 arquivo go.mod
- 1 ou mais arquivos .go
- 1 arquivo go.sum (recomendado)

Como começar:

- criar o repositório no github
- rodar o comando go mod init (com o path do github)
- criar um arquivo go na mesma pasta
- rodar go mod tidy
- commitar e fazer push

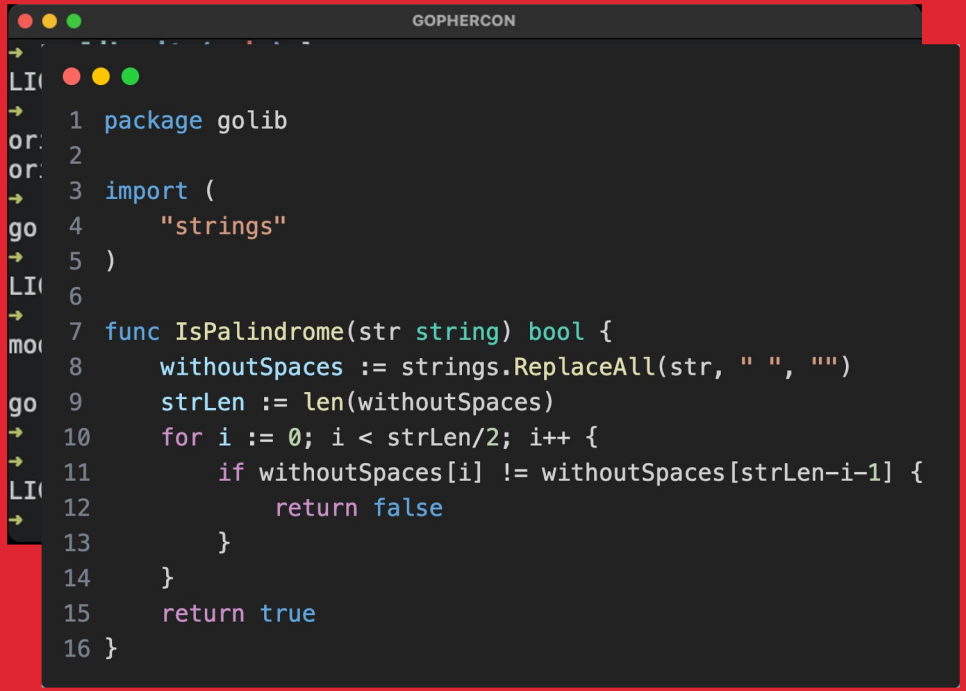
Como criar uma biblioteca

É necessário:

- repositório git
- 1 arquivo go.mod
- 1 ou mais arquivos .go
- 1 arquivo go.sum (recomendado)

Como começar:

- criar o repositório no github
- rodar o comando go mod init (com o path do github)
- criar um arquivo go na mesma pasta
- rodar go mod tidy
- commitar e fazer push



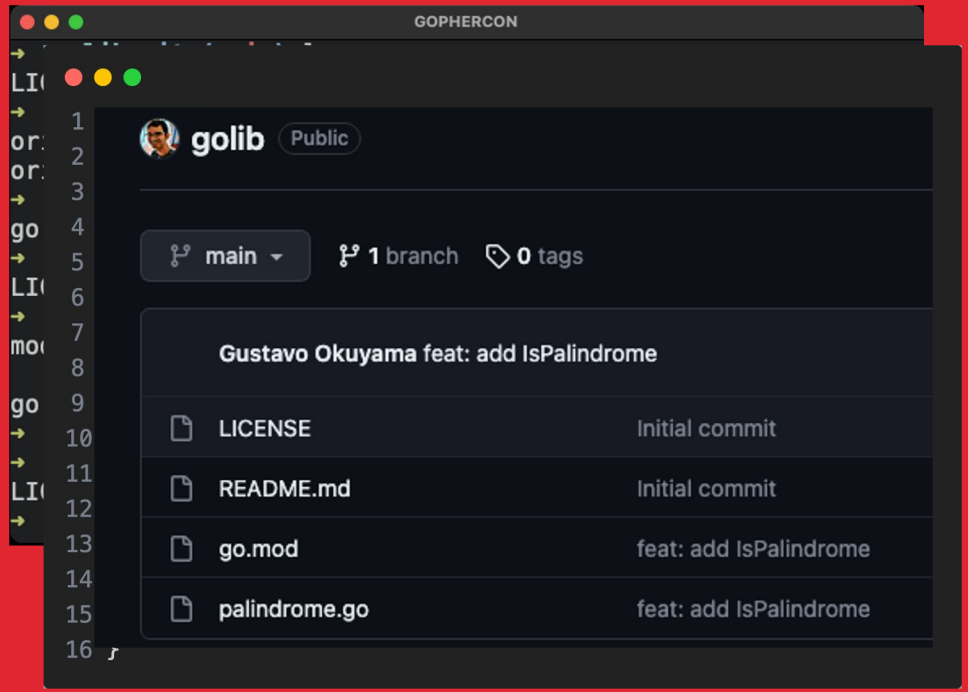
Como criar uma biblioteca

É necessário:

- repositório git
- 1 arquivo go.mod
- 1 ou mais arquivos .go
- 1 arquivo go.sum (recomendado)

Como começar:

- criar o repositório no github
- rodar o comando go mod init (com o path do github)
- criar um arquivo go na mesma pasta
- rodar go mod tidy
- commitar e fazer push



```
GOPHERCON
→  usinggolib go get -u github.com/gubtos/golib
go: downloading github.com/gubtos/golib v0.0.0-20230903165350-5461b29fc013
go: added github.com/gubtos/golib v0.0.0-20230903165350-5461b29fc013
→  usinggolib tree
.
├── go.mod
├── go.sum
└── main.go

1 directory, 3 files
```

```go
package main

import (
    "fmt"

    "github.com/gubtos/golib"
)

func main() {
    str := "a sacada da casa"
    if isPalidrome := golib.IsPalindrome(str); isPalidrome {
        fmt.Println("É palíndromo")
    } else {
        fmt.Println("Não é palíndromo")
    }
}
```

Usando a biblioteca

1. Rode o comando go get
2. Importe a biblioteca no seu arquivo
3. Execute o código

```
GOPHERCON
→  usinggolib go run .
É palíndromo
→  usinggolib
```

COMO IMPORTAR UMA BIBLIOTECA GOLANG RECÉM CRIADA

Adicionando uma versão semântica

GOPHERCON

```
→  golib git:(main) git --no-pager tag
→  golib git:(main) git tag v0.0.1
→  golib git:(main) git --no-pager tag
v0.0.1
→  golib git:(main) git push origin v0.0.1
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:gubtos/golib.git
 * [new tag]         v0.0.1 -> v0.0.1
```

GOPHERCON

```
→  usinggolib cat go.mod
module github.com/gubtos/usinggolib

go 1.21.0

require github.com/gubtos/golib v0.0.0-20230903165350-5461b29fc013
→  usinggolib go get -u github.com/gubtos/golib
go: upgraded github.com/gubtos/golib v0.0.0-20230903165350-5461b29fc013 => v0.0.1
→  usinggolib cat go.mod
module github.com/gubtos/usinggolib

go 1.21.0

require github.com/gubtos/golib v0.0.1
→  usinggolib
```

1. Adicione a tag vX.Y.Z no seu repositório.
   Se X >= 2 (major update) são necessários outros passos

Criando versão executável

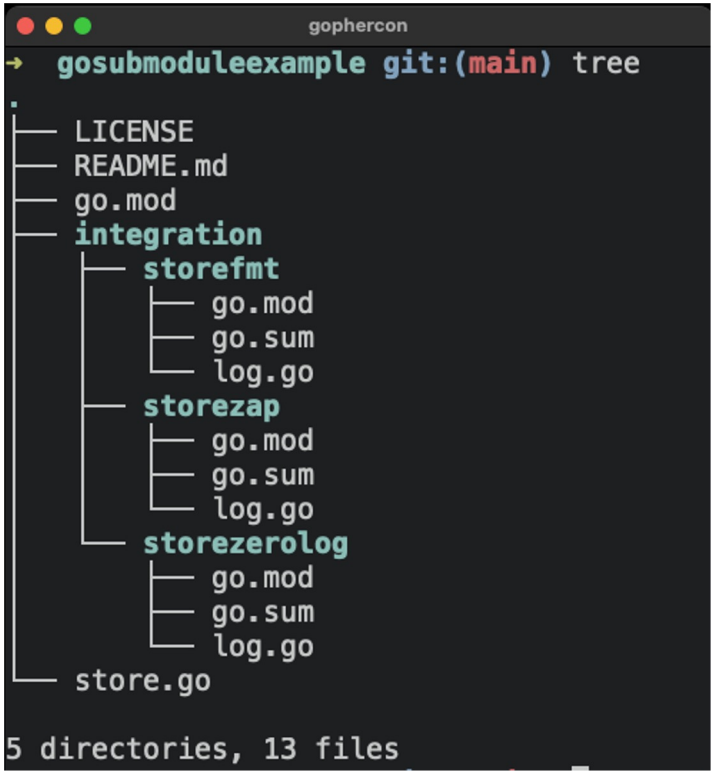
```go
package main

import (
    "fmt"
    "os"

    "github.com/gubtos/golib"
)

func main() {
    if len(os.Args) != 2 {
        fmt.Println("execute da seguinte forma:\npalindrome \"a sacada da casa\"")

        return
    }
    str := os.Args[1]
    if isPalidrome := golib.IsPalindrome(str); isPalidrome {
        fmt.Println("É palíndromo")
    } else {
        fmt.Println("Não é palíndromo")
    }
}
```

```
module github.com/gubtos/palindrome

go 1.21.0

require github.com/gubtos/golib v0.0.1

```

```
GOPHERCON
→  ~ palindrome
zsh: command not found: palindrome
→  ~ go install github.com/gubtos/palindrome@v0.0.3
go: downloading github.com/gubtos/palindrome v0.0.3
go: downloading github.com/gubtos/golib v0.0.1
→  ~ palindrome
execute da seguinte forma:
palindrome "a sacada da casa"
→  ~ palindrome "anotaram a data da maratona"
É palíndromo
```

1. O código precisa estar definido dentro do package main
2. A função a ser executada deve estar definida dentro da função main

Submódulos


gophercon
→ gosubmoduleexample git:(main) tree
.
├── LICENSE
├── README.md
├── go.mod
├── integration
│ ├── storefmt
│ │ ├── go.mod
│ │ ├── go.sum
│ │ └── log.go
│ ├── storezap
│ │ ├── go.mod
│ │ ├── go.sum
│ │ └── log.go
│ └── storezerolog
│ ├── go.mod
│ ├── go.sum
│ └── log.go
└── store.go
5 directories, 13 files


gophercon
→ gosubmoduleexample git:(main) git --no-pager tag
integration/storefmt/v0.0.1
integration/storezap/v0.0.1
integration/storezerolog/v0.0.1
v0.0.1

Submódulos

gophercon

```
→ gosubmoduleexample git:(main) bat -P store.go
```

```go
package gosubmoduleexample

import "errors"

var ErrNotFound = errors.New("not found")

type Data struct {
    ValA string
    ValB string
}

type Logger func(Data)

type Store struct {
    logger  Logger
    dataMap map[string]Data
}

func NewStore(logger Logger) *Store {
    return &Store{
        logger:  logger,
        dataMap: make(map[string]Data),
    }
}
```

Submódulos

gophercon

```
→ gosubmoduleexample git:(main) bat integration/storezerolog/log.go
```

```
   1   package storezerolog
   2
   3   import (
   4       "github.com/gubtos/gosubmoduleexample"
   5       "github.com/rs/zerolog/log"
   6   )
   7
   8   func Log(data gosubmoduleexample.Data) {
   9       log.Printf("data A: %s\ndata B: %s", data.ValA, data.ValB)
  10   }
```

gophercon

```
→ gosubmoduleexample git:(main) bat -P store.go
```

```
    dataMap map[string]Data
}

func NewStore(logger Logger) *Store {
    return &Store{
        logger:  logger,
        dataMap: make(map[string]Data),
    }
}
```

Submódulos

IMPORTANDO A BIBLIOTECA E USANDO UM DOS SUBMÓDULOS

gophercon

```
→ gosubmoduleexample git:(main) bat -P store.go
```

gophercon

```
→ gosubmoduleexample git:(main) bat integration/storezerolog/log.go
```

gophercon

```
→ gosubmoduleusageexample git:(main) bat main.go
```

```go
package main

import (
    "github.com/gubtos/gosubmoduleexample"
    "github.com/gubtos/gosubmoduleexample/integration/storezerolog"
)

func main() {
    logger := storezerolog.Log
    store := gosubmoduleexample.NewStore(logger)

    store.StoreData("my str a", "my str b")
}
```

Variadic

USANDO VARIADIC

```go
package main

import "fmt"

func receivingArray(values []int) {
    for i := range values {
        fmt.Println(i)
    }
}

func receivingVariadic(values ...int) {
    for i := range values {
        fmt.Println(i)
    }
}

func variadicComparison() {
    myArray := []int{1, 2, 3}

    receivingArray(myArray)

    receivingVariadic(1, 2, 3)
    receivingVariadic(myArray...)
}

```

Primeiras libs

```go
package object

import (
    "errors"
    "math/rand"
)

type BoilerFunc func(temp int, state bool)

type Config struct {
    TurnOnTemp              *int
    TurnOffTemp             *int
    TemperatureMeasureHooks []BoilerFunc
}

type Boiler struct {
    turnOnTemp              int
    turnOffTemp             int
    state                   bool
    temperatureMeasureHooks []BoilerFunc
}
```

```go
func NewBoiler(cfg Config) (*Boiler, error) {
    boiler := &Boiler{
        turnOnTemp:              20,
        turnOffTemp:             30,
        temperatureMeasureHooks: nil,
    }
    if cfg.TurnOnTemp != nil {
        boiler.turnOnTemp = *cfg.TurnOnTemp
    }
    if cfg.TurnOffTemp != nil {
        boiler.turnOffTemp = *cfg.TurnOffTemp
    }
    if boiler.turnOffTemp < boiler.turnOnTemp {
        return nil, errors.New("turn off temp should be greather than turn on temp")
    }
    if len(cfg.TemperatureMeasureHooks) > 0 {
        boiler.temperatureMeasureHooks = cfg.TemperatureMeasureHooks
    }

    return boiler, nil
}
```

Paradigma funcional

```go
type BoilerFunc func(temp int, state bool)

type Boiler struct {
    turnOnTemp              int
    turnOffTemp             int
    state                   bool
    temperatureMeasureHooks []BoilerFunc
}

type options struct {
    turnOnTemp              int
    turnOffTemp             int
    temperatureMeasureHooks []BoilerFunc
}

type Option func(*options)

func TurnOnTemp(temp int) Option {
    return func(o *options) {
        o.turnOnTemp = temp
    }
}

func TurnOffTemp(temp int) Option {
    return func(o *options) {
        o.turnOffTemp = temp
    }
}

func TemperatureMeasureHooks(hooks ...BoilerFunc) Option {
    return func(o *options) {
        o.temperatureMeasureHooks = hooks
    }
}
```

```go
func NewBoiler(opts ...Option) (*Boiler, error) {
    opt := &options{
        turnOnTemp:              20,
        turnOffTemp:             30,
        temperatureMeasureHooks: nil,
    }
    for i := range opts {
        opts[i](opt)
    }
    if opt.turnOffTemp < opt.turnOnTemp {
        return nil, errors.New("turn off temp should be greather than turn on temp")
    }
    return &Boiler{
        turnOnTemp:              opt.turnOnTemp,
        turnOffTemp:             opt.turnOffTemp,
        temperatureMeasureHooks: opt.temperatureMeasureHooks,
        state:                   false,
    }, nil
}
```

Comparação

```go
func ObjectPattern() {
    _, _ = object.NewBoiler(object.Config{})

    turnOnTemp := 20
    turnOffTemp := 80
    f1 := func(temp int, state bool) { fmt.Println(temp) }
    f2 := func(temp int, state bool) { fmt.Println(state) }
    _, _ = object.NewBoiler(object.Config{
        TurnOnTemp:              &turnOnTemp,
        TurnOffTemp:             &turnOffTemp,
        TemperatureMeasureHooks: []object.BoilerFunc{f1, f2},
    })
}
```

```go
func FunctionalPattern() {
    _, _ = functional.NewBoiler()

    f1 := func(temp int, state bool) { fmt.Println(temp) }
    f2 := func(temp int, state bool) { fmt.Println(state) }
    _, _ = functional.NewBoiler(
        functional.TurnOnTemp(20),
        functional.TurnOffTemp(80),
        functional.TemperatureMeasureHooks(f1, f2),
    )
}
```

Struct tags

```
10 type ClientConfig struct {
11     URL     string        `json:"url"`
12     Timeout time.Duration `json:"timeout"`
13     Retry   int           `json:"retry"`
14 }
```

```
10 type ClientConfig struct {
11     URL                string        `key:"url" default:""`
12     Timeout            time.Duration `key:"timeout" default:"2s"`
13     Retry              int           `key:"retry" default:"0"`
14     MaxIdleConnsPerHost int          `key:"maxIdleConsPerHost" default:"20"`
15     ExampleFloat       float64       `key:"exampleFloat" default:"20.2"`
16 }
```

Struct tags

```go
func FromDefaultTags[T any]() T {
    var defaultClientConfig T

    cfgPtr := reflect.ValueOf(&defaultClientConfig)
    cfgElem := cfgPtr.Elem()
    cfgElemType := cfgElem.Type()

    for i := 0; i < cfgElem.NumField(); i++ {
        f := cfgElemType.Field(i)
        if defaultValue, ok := f.Tag.Lookup("default"); ok {
            elemType := cfgElem.Field(i).Type().String()
            field := cfgElem.Field(i)
            switch elemType {
            case "string":
                field.SetString(defaultValue)
            case "int32", "int", "int64":
                v, err := strconv.ParseInt(defaultValue, 10, 0)
                if err != nil {
                    panic(err)
                }
                field.SetInt(v)
            case "time.Duration":
                v, err := time.ParseDuration(defaultValue)
                if err != nil {
                    panic(err)
                }
                field.SetInt(int64(v.Abs()))
            case "float32", "float64":
                v, err := strconv.ParseFloat(defaultValue, 64)
                if err != nil {
                    panic(err)
                }
                field.SetFloat(v)
            default:
                panic(errUnimplementedType)
            }
        }
    }

    return defaultClientConfig
}
```

Configs

```go
func Config(clientConfig ClientConfig) Option {
    return func(o *options) {
        o.clientConfig = clientConfig
    }
}

func ConfigFromEnv() Option {
    return func(o *options) {
        o.clientConfig = FromEnv[ClientConfig]()
    }
}

func DefaultConfig() ClientConfig {
    return FromDefaultTags[ClientConfig]()
}
```

```go
func StructTagsComparison() {
    // mode 1
    structtags.NewClient(structtags.ConfigFromEnv())

    // mode 2
    cfg := structtags.DefaultClientConfig()
    cfg.URL = "https://gopherconbr.org/"
    structtags.NewClient(structtags.Config(cfg))

    //mode 3
    serviceConfig := FromExternalConfigLib()
    structtags.NewClient(structtags.Config(serviceConfig.ClientConfig))
}

type ServiceConfig struct {
    structtags.ClientConfig `key:"clientConfig"`
}
```

Evolução das bibliotecas

Antes

- "estimulava" o preenchimento da config
- fazia tudo em um só lugar
- pouco extensível ou personalizável
- sem submódulos
- difícil evolução

Depois

- paradigma funcional
- código default é bem limpo
- com submódulos (hooks, middlewares)
- maior personalização
- fácil evolução
- suporte a ENV e default values usando struct-tags (integração com a lib de config interna)

COMPARATIVO PARA AS BIBLIOTECAS MAIS NOVAS

Exemplos de bibliotecas

27 bibliotecas

config: lib lê de um arquivo (yaml, json) e ENV. Integra com outras bibliotecas através das struct tags (default values)

logs: lib de logs que implementa os standards (formato de logs e campos obrigatórios). Também possui middlewares(submódulos) que injetam os valores dos headers/metadata nos logs

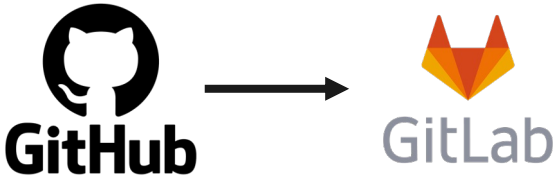
feature-flags: integração para a solução interna de feature flags

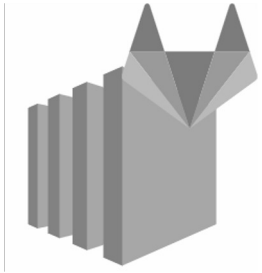
http-client: client http que implementa os standards (integração com APM, métricas prometheus, circuit breaker)

pg-client, redis-client, aws-client, kafka-client: client que integra com config, possui configurações recomendadas e middlewares

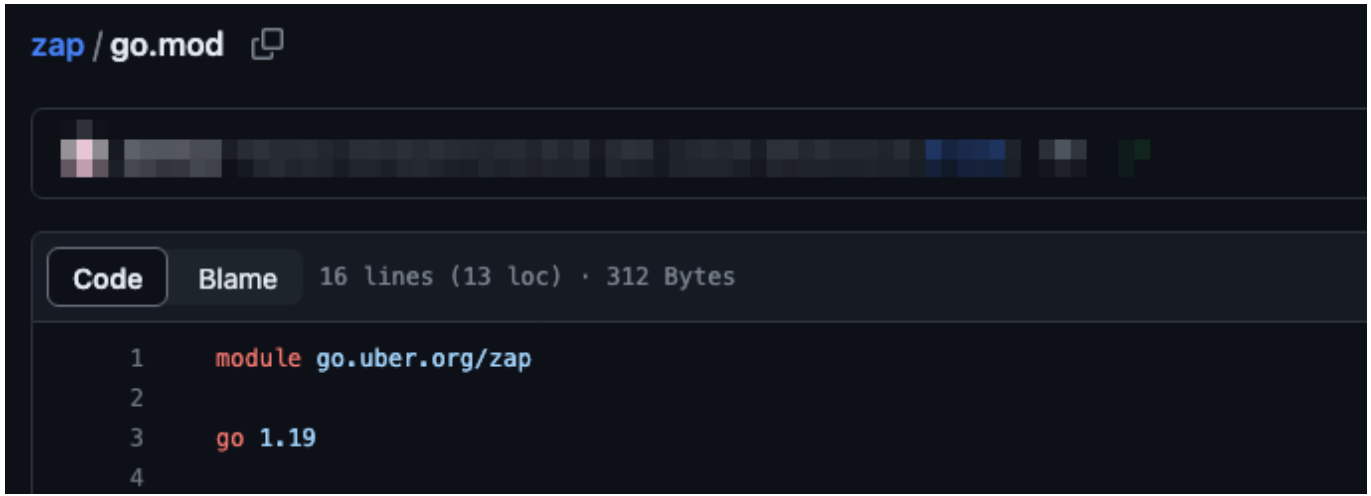
grpc-interceptors: conjunto de interceptors (circuit breaker, repassar metadata, debug, tracing, metrics)

Problemas levantados





Vanity URL


zap / go.mod
Code
Blame
16 lines (13 loc) · 312 Bytes
1 module go.uber.org/zap
2
3 go 1.19
4

Vanity URL

gophercon

```
→  usinggolib export GO111MODULE=off
→  usinggolib go get -u -x go.uber.org/zap
# get https://go.uber.org/zap?go-get=1
# get https://go.uber.org/zap?go-get=1: 200 OK (1.198s)
```

gophercon

```
→  usinggolib curl --silent --request GET --url 'http://go.uber.org/zap?go-get=1' | bat --language html
```

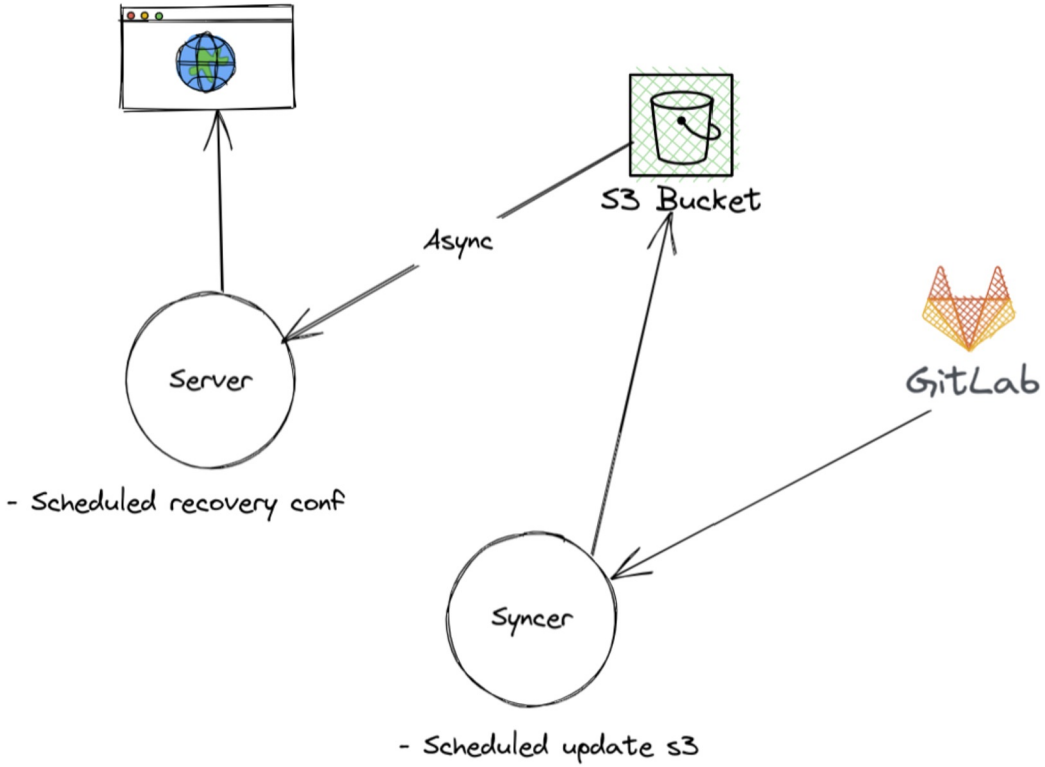
STDIN

```
<!DOCTYPE html>
<html>
    <head>
        <meta name="go-import" content="go.uber.org/zap git https://github.com/uber-go/zap">
        <meta name="go-source" content="go.uber.org/zap https://github.com/uber-go/zap https://gi
thub.com/uber
    -go/zap/tree/master{/dir} https://github.com/uber-go/zap/tree/master{/dir}/{file}#L{line}">
        <meta http-equiv="refresh" content="0; url=https://pkg.go.dev/go.uber.org/zap">
    </head>
    <body>
        Nothing to see here. Please <a href="https://pkg.go.dev/go.uber.org/zap">move along</a>.
    </body>
</html>
```

Implementação

GitHub

github.com/uber-go/sally



Disaster Recovery (Antes)

- Configuração do job que executa replace no estágio inicial da pipeline (regex go.mod + go mod replace)

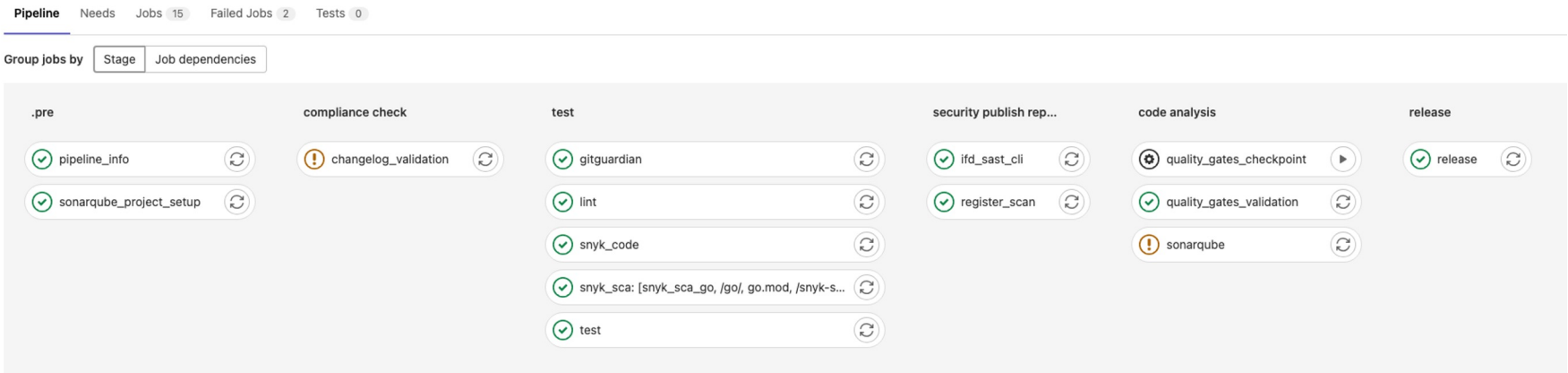
- URL da lib (import) muda
- maior complexidade pipeline
- mudanças descentralizadas
- limitação ao gitlab

Disaster Recovery (Depois)

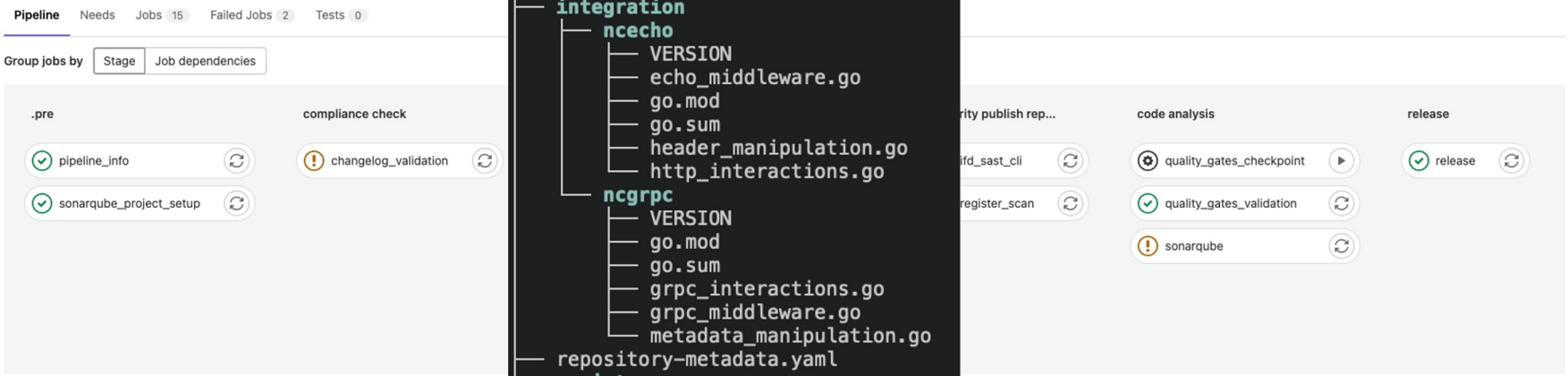
- Configuração das credenciais do syncer
- Configuração do server para o novo host e fonte de dados
- Deploy do server
- Configuração do DNS

- URL da lib (import) não muda
- Não precisa mudar pipeline
- Mudanças centralizadas

Pipeline


Pipeline
Needs
Jobs 15
Failed Jobs 2
Tests 0
Group jobs by
Stage
Job dependencies
.pre
pipeline_info
sonarqube_project_setup
compliance check
changelog_validation
test
gitguardian
lint
snyk_code
snyk_sca: [snyk_sca_go, /go/, go.mod, /snyk-s...
test
security publish rep...
ifd_sast_cli
register_scan
code analysis
quality_gates_checkpoint
quality_gates_validation
sonarqube
release
release

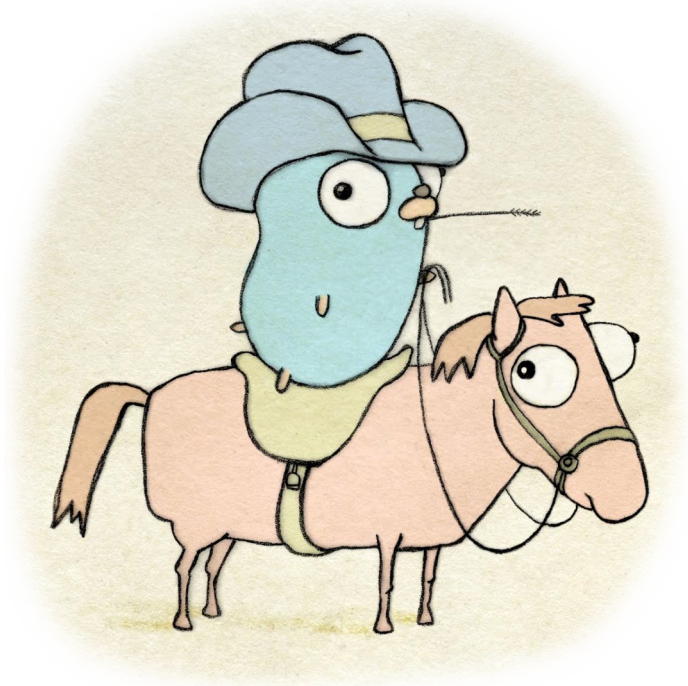
Pipeline


Pipeline
Needs
Jobs 15
Failed Jobs 2
Tests 0
Group jobs by
Stage
Job dependencies
.pre
pipeline_info
sonarqube_project_setup
compliance check
changelog_validation
code analysis
quality_gates_checkpoint
quality_gates_validation
sonarqube
release
release


gophercon
nocontent git:(main) ✗ tree
.
├── Makefile
├── README.md
├── VERSION
├── engine.go
├── go.mod
├── go.sum
├── integration
│   ├── ncecho
│   │   ├── VERSION
│   │   ├── echo_middleware.go
│   │   ├── go.mod
│   │   ├── go.sum
│   │   ├── header_manipulation.go
│   │   └── http_interactions.go
│   └── ncgrpc
│   ├── VERSION
│   ├── go.mod
│   ├── go.sum
│   ├── grpc_interactions.go
│   ├── grpc_middleware.go
│   └── metadata_manipulation.go
├── repository-metadata.yaml
└── scripts
    └── bash
        ├── lint.sh
        ├── test.sh
        ├── tidy.sh
        ├── update.sh
        └── vet.sh
6 directories, 24 files

Próximos passos

1. vanity libs em qualquer grupo do gitlab
2. atualização instantânea do vanity server
3. godocs
4. adicionar proxy (ex. gomods/athens)



LINK PARA PALESTRA:
GITHUB.COM/GUBTOS/GOPHERCON23