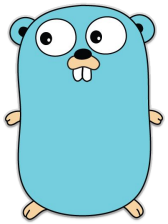


Go Lang

Diego Pacheco

About Me

- Cat's Father
- Principal Software Architect
- Agile Coach
- SOA/Microservices Expert
- DevOps Practitioner
- Speaker
- Author

diegopacheco

@diego_pacheco

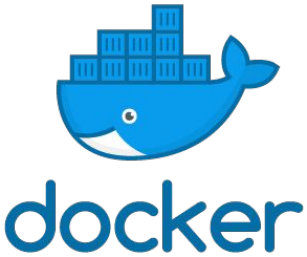
http://diego-pacheco.blogspot.com.br/

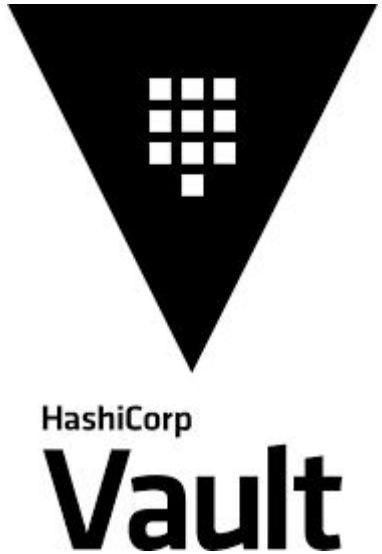
Go Lang

Robert Griesemer, Rob Pike, Ken Thompson

- 2009
- By Google
- Typed and Awesome for Concurrency programing
- Similar to C
- Compiled, Fast
- Very Opinionated(no Ternary, No Exceptions, No Generics)
- Single Binary programs
- Great for DevOps

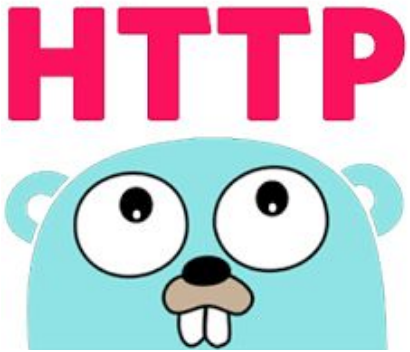
Software Written in Go





https://github.com/golang/go/wiki/GoUsers

Go Lang
Http Service

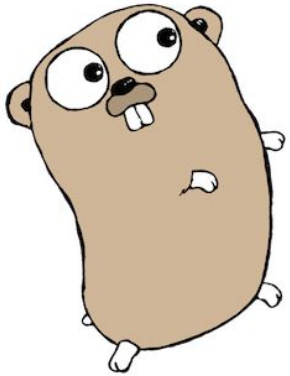


web-sample.go

```go
package main

import (
    "fmt"
    "net/http"
)

func handler(w http.ResponseWriter, r *http.Request) {
    fmt.Fprint(w, "Go HTTP Service running like a charm.")
}

func main() {
    fmt.Print("Serving at http://0.0.0.0:8080")
    http.HandleFunc("/", handler)
    http.ListenAndServe(":8080", nil)
}
```

```
go run web-sample.go
Serving at http://0.0.0.0:8080
```

Go Lang Logging



log-sample.go

```go
package main

import (
    "log"
    "os"
)

func main() {
    l := log.New(os.Stdout, "", log.Ldate|log.Lmicroseconds|log.Lshortfile)
    l.Println("hello log from Go. ")
}
```

```
go run log-sample.go
2018/12/26 19:04:40.381173 log-sample.go:10: hello log from Go.
```

Go Lang Yaml Part 1

yaml-sample.go

```go
package main

import (
    "fmt"
    "log"

    "gopkg.in/yaml.v2"
)

var data = `
a: Easy!
b:
  c: 2
  d: [3, 4]
`

type T struct {
    A string
    B struct {
        RenamedC int   `yaml:"c"`
        D        []int `yaml:",flow"`
    }
}
```

Go Lang Yaml Part 2

```go
func main() {
    t := T{}

    err := yaml.Unmarshal([]byte(data), &t)
    if err != nil {
        log.Fatalf("error: %v", err)
    }
    fmt.Printf("--- t:\n%v\n\n", t)

    d, err := yaml.Marshal(&t)
    if err != nil {
        log.Fatalf("error: %v", err)
    }
    fmt.Printf("--- t dump:\n%s\n\n", string(d))

    m := make(map[interface{}]interface{})

    err = yaml.Unmarshal([]byte(data), &m)
    if err != nil {
        log.Fatalf("error: %v", err)
    }
    fmt.Printf("--- m:\n%v\n\n", m)
```

Go Lang OS Env Vars

envvars-sample.go

```go
package main

import "os"
import "strings"
import "fmt"

func main() {
    os.Setenv("FOO", "1")
    fmt.Println("FOO:", os.Getenv("FOO"))
    fmt.Println("BAR:", os.Getenv("BAR"))

    fmt.Println()
    for _, e := range os.Environ() {
        pair := strings.Split(e, "=")
        fmt.Println(pair[0])
    }
}
```

Go Lang JSON Part 1

json-sample.go

```go
package main

import (
    "encoding/json"
    "fmt"
    "net/http"
)

type User struct {
    Firstname string `json:"firstname"`
    Lastname  string `json:"lastname"`
    Age       int    `json:"age"`
}

```

Go Lang JSON Part 2

```go
func main() {
    http.HandleFunc("/decode", func(w http.ResponseWriter, r *http.Request) {
        var user User
        json.NewDecoder(r.Body).Decode(&user)
        fmt.Fprintf(w, "%s %s is %d years old!", user.Firstname, user.Lastname, user.Age)
    })

    http.HandleFunc("/encode", func(w http.ResponseWriter, r *http.Request) {
        peter := User{
            Firstname: "John",
            Lastname:  "Doe",
            Age:       25,
        }
        json.NewEncoder(w).Encode(peter)
    })

    http.ListenAndServe(":8080", nil)
}
```

```
curl -s -XPOST -d'{"firstname":"Donald","lastname":"Trump","age":70}' http://localhost:8080/decode
Donald Trump is 70 years old!%

curl -s http://localhost:8080/encode
{"firstname":"John","lastname":"Doe","age":25}
```

Go Lang Make your REPL

repl-sample.go

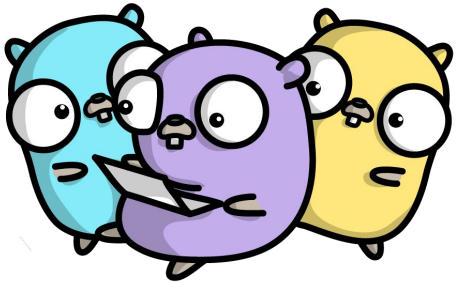
```go
package main

import "strings"
import "github.com/abiosoft/ishell"

func main() {
    // create new shell.
    // by default, new shell includes 'exit', 'help' and 'clear' commands.
    shell := ishell.New()

    // display welcome info.
    shell.Println("Sample Interactive Shell")

    // register a function for "greet" command.
    shell.AddCmd(&ishell.Cmd{
        Name: "greet",
        Help: "greet user",
        Func: func(c *ishell.Context) {
            c.Println("Hello", strings.Join(c.Args, " "))
        },
    })

    // run shell
    shell.Run()
}
```

```
diego@4winds ~/github/diegopacheco/DevOpsEngineerExpress/source/golang master go run repl-sample.go
Sample Interactive Shell
>>> help

Commands:
  clear      clear the screen
  exit       exit the program
  greet      greet user
  help       display help

>>>
```

REPL(s) are better than ALIAS because they are OUTSIDE of the OS. They can be inside too, however they are way easier to update and provide much better troubleshooting experience. Take look at CMSH use case:

http://diego-pacheco.blogspot.com/2018/07/experiences-building-cassandra.html

Go Lang Goroutines Part 1

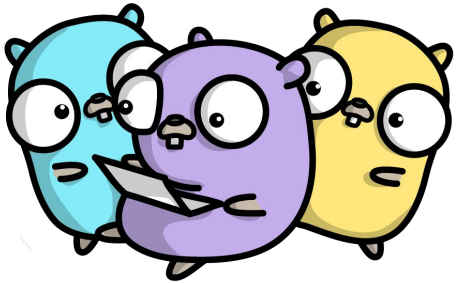


goroutine-sample.go

```go
package main

import "fmt"

func main() {
    jobs := make(chan int, 100)
    results := make(chan int, 100)

    go worker(jobs, results)
    go worker(jobs, results)
    go worker(jobs, results)

    for i := 0; i < 100; i++ {
        jobs <- i
    }
    close(jobs)

    for i := 0; i < 100; i++ {
        fmt.Println(<-results)
    }
}
```

Go Lang Goroutines Part 2

```
23  /* Sender */
24  func worker(jobs <-chan int, results chan<- int) {
25      for n := range jobs {
26          results <- fib(n)
27      }
28  }
29
30  func fib(n int) int {
31      if n <= 1 {
32          return n
33      }
34      return fib(n-1) + fib(n-2)
35  }
```



Go Lang Etcd Client Part 1

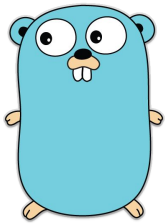
etcd-client-sample.go

```go
package main

import (
    "context"
    "fmt"
    "time"

    "github.com/coreos/etcd/clientv3"
)

func main() {
    cli, err := clientv3.New(clientv3.Config{
        Endpoints:   []string{"localhost:2379", "localhost:22379", "localhost:32379"},
        DialTimeout: 5 * time.Second,
    })
    if err != nil {
        fmt.Println(err)
    } else {
        fmt.Println("Etcd Client connected")
    }
```

Go Lang Etcd Client Part 2

```
22      ctx, cancel := context.WithTimeout(context.Background(), 10*time.Second)
23
24      resPut, errPut := cli.Put(ctx, "x", "10")
25      if errPut != nil {
26          fmt.Println(errPut)
27      } else {
28          fmt.Printf("Put on ETCD : %s \n", resPut.OpResponse().Put())
29      }
30
31      x, _ := cli.Get(ctx, "x")
32      fmt.Printf("Get x from ETCD: %s \n", string(x.Kvs[0].Value))
33
34      cancel()
35      cli.Close()
36  }
```


```
39  diego@4winds Ξ ~/bin/etcd-v3.3.6-linux-amd64 Ξ ./etcd
40  2018-12-26 19:58:15.156871 I | etcdmain: etcd Version: 3.3.6
41  2018-12-26 19:58:15.156926 I | etcdmain: Git SHA: 932c3c01f
42  2018-12-26 19:58:15.156933 I | etcdmain: Go Version: go1.9.6
43  2018-12-26 19:58:15.156939 I | etcdmain: Go OS/Arch: linux/amd64
44  2018-12-26 19:58:15.156946 I | etcdmain: setting maximum number of CPUs to 8, total number of available CPUs
45  ...
46
47  go run etcd-client-sample.go
48  Etcd Client connected
49  Put on ETCD : &{cluster_id:14841639068965178418 member_id:10276657743932975437 revision:29 raft_term:5  <nil>
50  Get x from ETCD: 10
```



Go Lang

Diego Pacheco