

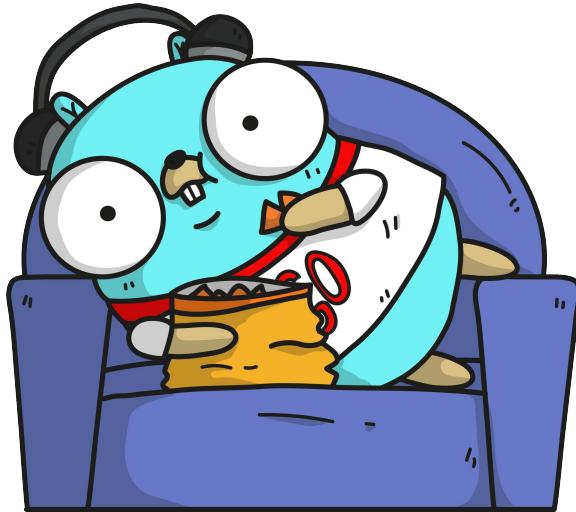
Golang



Catherin Cruz
@UserTwoGG



Conceptos Básicos



- ❑ Lenguaje Imperativo
- ❑ Fuertemente tipado
- ❑ Sintaxis similar a C, menos (), no ;
- ❑ No es necesaria una máquina virtual
- ❑ No hay clases, pero sí estructuras con métodos
- ❑ Funciones pueden retornar varios parámetros
- ❑ Diseñado para concurrencia



Hello World



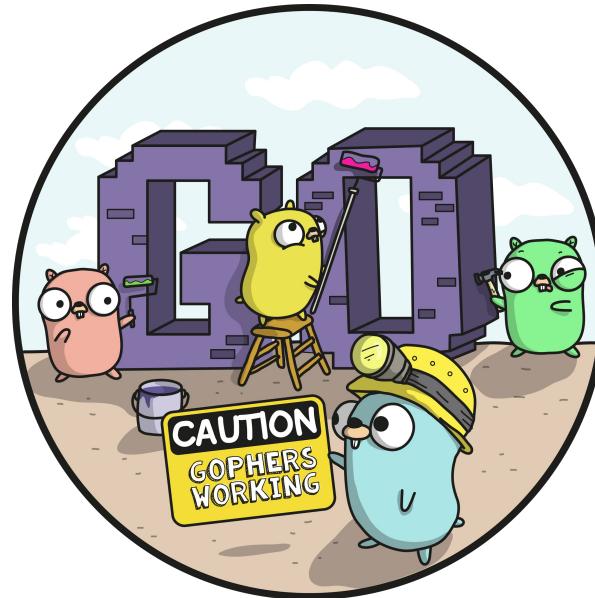
File: hello.go

```
package main

import (
    "fmt"
    "time"
)

func main() {
    fmt.Println("Hello Go")
    fmt.Println("The time is", time.Now())
}
```

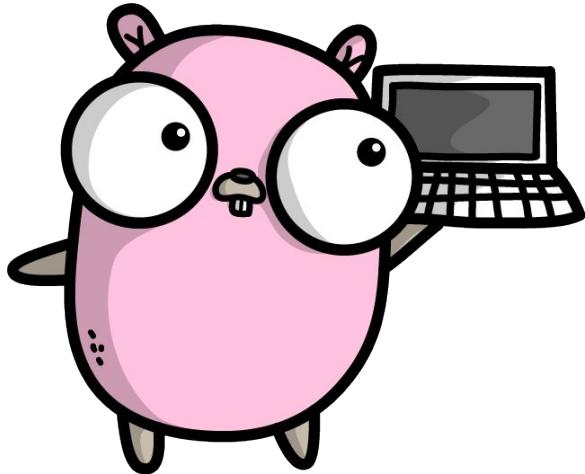
\$ go run hello.go



Catherin Cruz



Declaración & Variables



```
var foo int // sin inicializar  
var foo int = 42 // con inicialización  
var foo, bar int = 42, 1302 // declaración múltiple  
var foo = 42 // tipo omitido se infiere  
foo := 42 // tipo omitido, sólo dentro de funciones  
const constant = "This is a constant"
```

```
bool, string  
int int8 int16 int32 int64  
uint uint8 uint16 uint32 uint64 uintptr  
byte // alias for uint8  
float32 float64  
complex64 complex128
```



Funciones

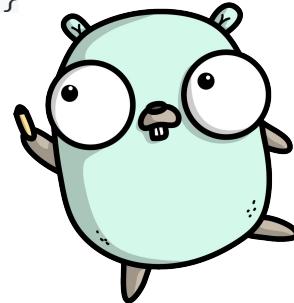


```
// función simple  
func functionName() {}
```

```
// función con parámetros (nombre tipo)  
func functionName(param1 string, param2 int) {}
```

```
// múltiples parámetros mismo tipo  
func functionName(param1, param2 int) {}
```

```
// tipo del retorno inferido  
func functionName() int {  
    return 42  
}
```

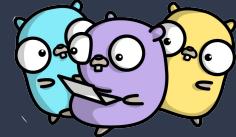


```
// retorno con múltiples valores  
func returnMulti() (int, string) {  
    return 42, "foobar"  
}  
var x, str = returnMulti()
```

```
// retorno múltiple inferido  
func returnMulti2() (n int, s string) {  
    n = 42  
    s = "foobar"  
    // n and s will be returned  
    return  
}  
var x, str = returnMulti2()
```



Funciones 2



Golang

```
func main() {
    fmt.Println(adder(1, 2, 3)) // 6
    fmt.Println(adder(9, 9))    // 18
    nums := []int{10, 20, 30}
    fmt.Println(adder(nums...)) // 60
}
```

// Usando ... antes del tipo se puede indicar que se aceptan 0 o más parámetros del mismo tipo

```
func adder(args ...int) int {
    total := 0
    for _, v := range args { // Intera según tantos argumentos reciba
        total += v
    }
    return total
}
```



Catherin Cruz



Paquetes



Declaración siempre al inicio de la archivo
Ejecutables están en el paquete main
Importar con mayúsculas: público
Importar con minúsculas: privado

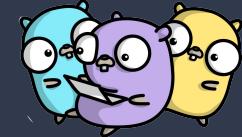
```
import (
    "fmt"
    "time"
    "http/net"
    "google.golang.org/grpc"
    "github.com/jtolds/gls"
    "github.com/smartystreets/assertions"
)
```



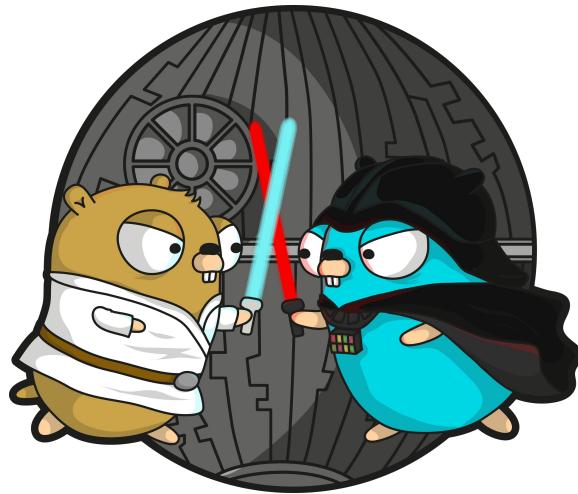
Catherin Cruz



Estructuras: If



Golang



```
func main() {  
    // Basic one  
    if x > 0 {  
        return x  
    } else {  
        return -x  
    }  
  
    if a := b + c; a < 42 {  
        return a  
    } else {  
        return a - 42  
    }  
}
```



Catherin Cruz





Estructuras: For

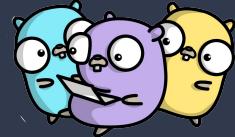
```
// Solo existe el for  
for i := 1; i < 10; i++ {  
}
```

```
for ; i < 10; { // ejemplo similar a while  
}
```

```
for i < 10 { // se pueden omitir ; si es solo una condición  
}
```

```
for { // se puede omitir la condición mientras sea verdadero (true)  
}
```





Golang

Estructuras: Switch

```
// switch statement
switch operatingSystem {
    case "darwin":
        fmt.Println("Mac OS Hipster")
        // cierra el case automáticamente
    case "linux":
        fmt.Println("Linux Geek")
    default:
        // Windows, BSD, ...
        fmt.Println("Other")
}
```

```
// también se pueden hacer comparaciones
number := 42
switch {
    case number < 42:
        fmt.Println("Smaller")
    case number == 42:
        fmt.Println("Equal")
    case number > 42:
        fmt.Println("Greater")
}
```



Catherin Cruz



Estructuras: Arrays



```
var a [10]int // array tamaño 10 tipo enteros  
a[3] = 42    // define elemento  
i := a[3]    // leer el elemento  
  
// declaración e inicialización  
var a = [2]int{1, 2}  
a := [2]int{1, 2} //tamaño definido  
a := [...]int{1, 2} // elipsis -> Compilador define el tamaño
```



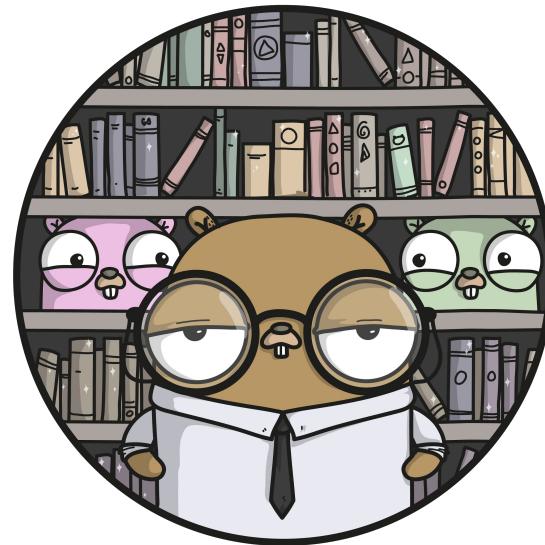
Estructuras: Slice



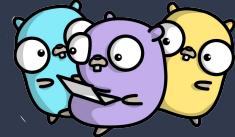
```
var a []int // declaración similar al array, tamaño no especificado
var a = []int {1, 2, 3, 4} // declaración con inicialización
a := []int{1, 2, 3, 4} // versión corta
chars := []string{0:"a", 2:"c", 1: "b"} // ["a", "b", "c"]

// creación con función make
a = make([]byte, 5, 5) // primero tamaño, luego capacidad
a = make([]byte, 5) // capacidad es opcional

// creación con base en array
x := [3]string{"Лайка", "Белка", "Стрелка"}
s := x[:] // a slice referencing the storage of x
```



Estructuras: Maps



Golang



```
var m map[string]int  
m = make(map[string]int)  
m["key"] = 42  
fmt.Println(m["key"])
```

```
delete(m, "key")
```

```
elem, ok := m["key"] // evalúa si key está presente y la retorna
```

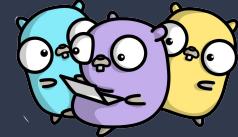
```
// map literal  
var m = map[string]Vertex{  
    "Bell Labs": {40.68433, -74.39967},  
    "Google": {37.42202, -122.08408},  
}
```



Catherin Cruz



Estructuras: Structs



Golang



```
package main

import (
    "fmt"
)

type Employee struct {
    firstName, lastName string
    age, salary int
}
```

```
func main() {
    //creación de estructuras usando el nombre
    emp1 := Employee{
        firstName: "Sam",
        age: 25,
        salary: 500,
        lastName: "Anderson",
    }
    // creación de estructuras sin usar nombres
    emp2 := Employee{"Thomas", "Paul", 29, 800}
    fmt.Println("Employee 1", emp1)
    fmt.Println("Employee 2", emp2)
}
```



Catherin Cruz



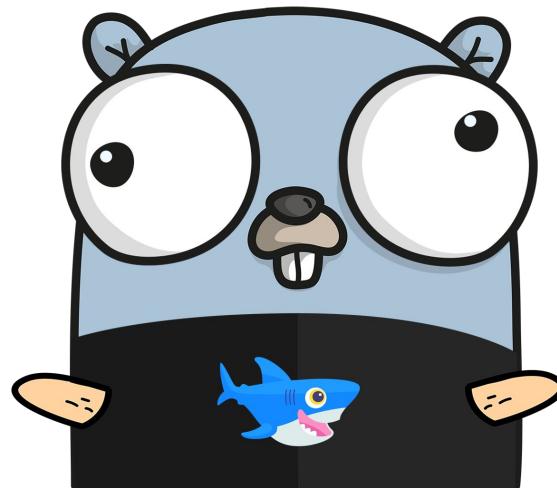
Docker



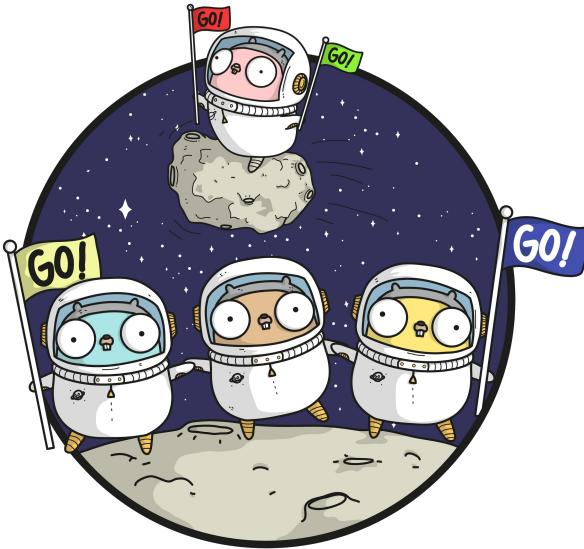
- Un único archivo de configuración
- Ejecución de contenedores en una sola instancia
- Agnóstico al contenido en ejecución
- Conexión directa a recursos de la máquina

<http://training.play-with-docker.com/>

<http://labs.play-with-docker.com/>



Go + Docker + Heroku



Servicios Rest

Templates y formularios

Despliegue gratuito en Heroku

<https://github.com/twogg-git/go-getting-started>

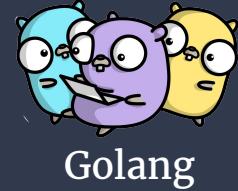
<https://twogg-go-getting-started.herokuapp.com/>



Catherin Cruz



gopherize.me



Golang

<https://gopherize.me/>

<https://github.com/matryer/gopherize.me>

<https://github.com/ashleymcnamara/>



Catherin Cruz

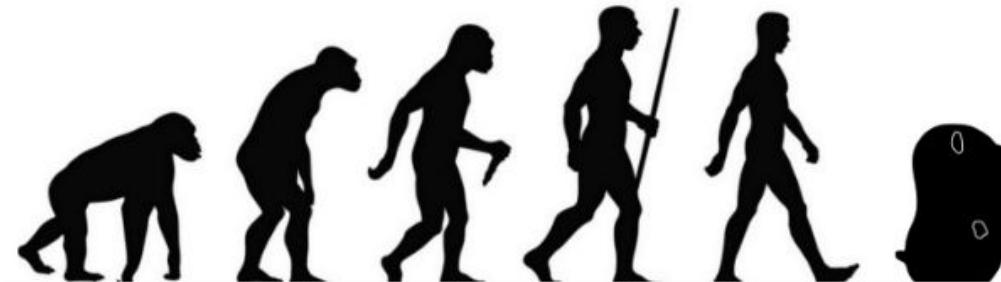


Just For Func



<https://youtube.com/c/justforfunc>

<https://github.com/campoy/justforfunc>



Catherin Cruz





Quick Start

Create a Hugo site using the beautiful Ananke theme.

This quick start uses `macos` in the examples. For instructions about how to install Hugo on other operating systems, see [install](#).

You also need [Git installed](#) to run this tutorial.

Step 1: Install Hugo [»](#)

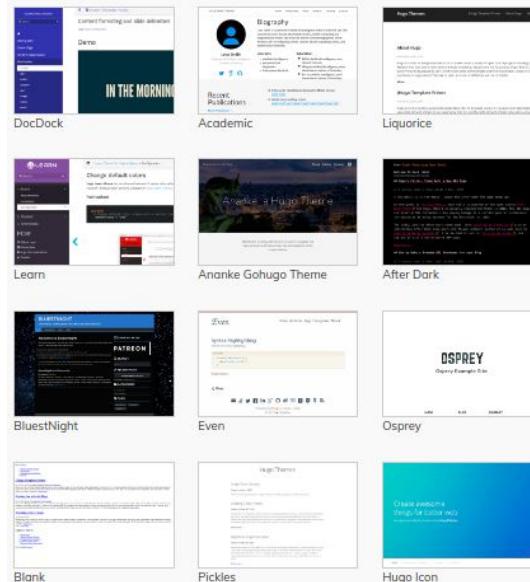
Homebrew, a package manager for `macos`, can be installed from `brew.sh`. See [install](#) if you are running Windows etc.

```
brew install hugo
```

To verify your new install:

```
hugo version
```

Hugo Themes



Hugo Documentation

Hugo is the [world's fastest static website engine](#). It's written in Go (aka Golang) and developed by [bep](#), [spf13](#) and [friends](#). Below you will find some of the most common and helpful pages from our documentation.

Base Templates and Blocks

The base and block constructs allow you to define the outer shell of your master templates (i.e., the chrome of the page).

[Read More »](#)

Content Organization

Hugo assumes that the same structure that works to organize your source content is used to organize the rendered site.

[Read More »](#)

Hugo's Lookup Order

The lookup order is a prioritized list used by Hugo as it traverses your files looking for the appropriate corresponding file to render your content.

[Read More »](#)

Introduction to Hugo Templating

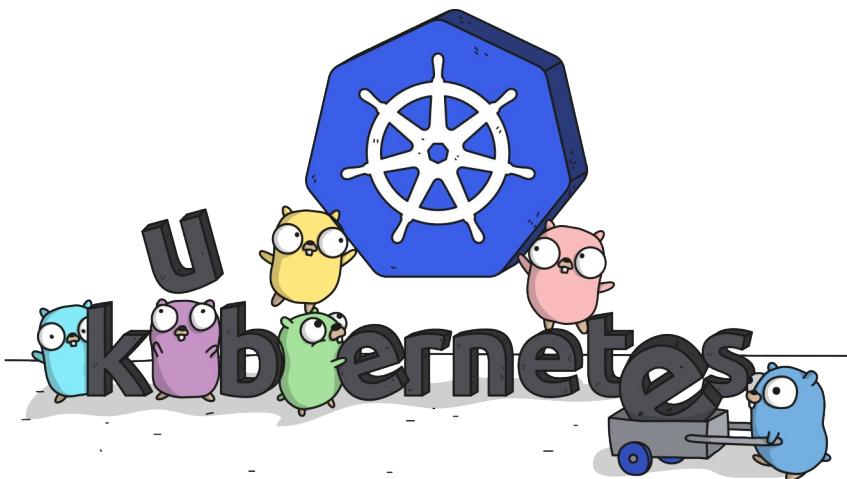
Hugo uses Go's `html/template` and `text/template` libraries as the basis for



Kubernetes



- Conexión directa con los recursos
- Una configuración muchos ambientes
- Despliegue múltiple en paralelo!
- Control de todos los ambientes
- Escala de servicios en tiempo real





Servicio bancario hecho en Go

Orientado a microservicios

Control de fallos y estado de los servicios

Integración y despliegue continuo

Matt Heath

<https://speakerdeck.com/mattheath>

<https://www.youtube.com/watch?v=dVnMLtdJzn4>

Referencias & Links



<https://blog.golang.org/>

https://golang.org/doc/effective_go.html#

<https://gobyexample.com/>

<https://github.com/ashleymcnamara/gophers>

<https://golangbot.com/learn-golang-series/>

<https://themes.gohugo.io/>

<http://www.womenwhogo.org/>

<https://gophers.slack.com>

