

Ultimate Guide to Golang Development for Beginners



Huseyin BABAL

Software Development Team Lead, Hazelcast Cloud

Outline

- Golang Warm-up
- Development Environment Setup
- Go by Examples
- Testing
- CI/CD
- Demo Time

#1 Golang Warm-up

...

What is Golang?

- Statically typed & compiled language
- Designed by Robert Griesemer, Rob Pike, Ken Thompson at Google
- Similar to C, but with **memory safety**, **garbage collection**
- Landing page is <https://golang.org/>

#2 Development Environment Setup

...

Installation

You can download and install Golang based on your distribution here

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

Featured downloads

Microsoft Windows

Windows 7 or later, Intel 64-bit processor

[go1.16.5.windows-amd64.msi](#) (119MB)

Apple macOS

macOS 10.12 or later, Intel 64-bit processor

[go1.16.5.darwin-amd64.pkg](#) (125MB)

Linux

Linux 2.6.23 or later, Intel 64-bit processor

[go1.16.5.linux-amd64.tar.gz](#) (123MB)

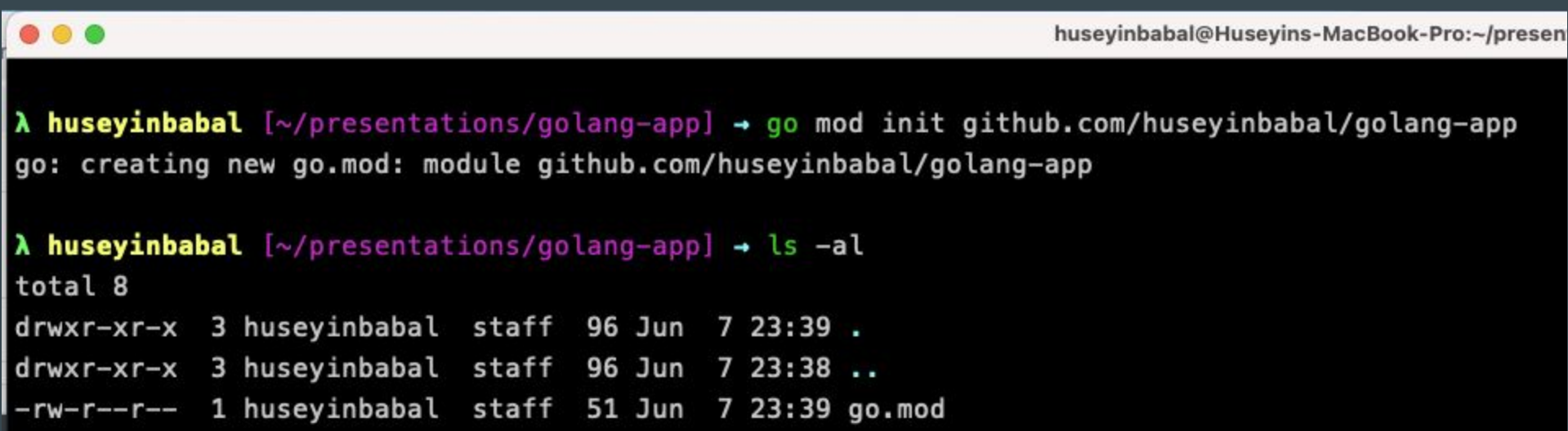
Source

[go1.16.5.src.tar.gz](#) (20MB)

Stable versions

Go Modules

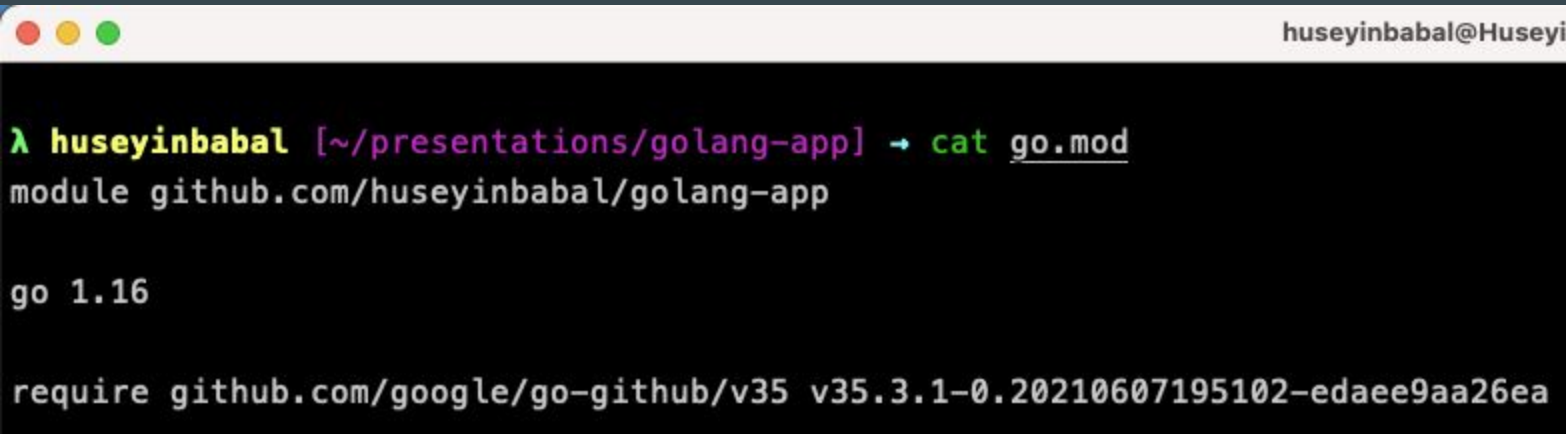
Dependency management system for Golang.



```
huseyinbabal@Huseyins-MacBook-Pro:~/presen  
  
λ huseyinbabal [~/presentations/golang-app] → go mod init github.com/huseyinbabal/golang-app  
go: creating new go.mod: module github.com/huseyinbabal/golang-app  
  
λ huseyinbabal [~/presentations/golang-app] → ls -al  
total 8  
drwxr-xr-x  3 huseyinbabal  staff  96 Jun  7 23:39 .  
drwxr-xr-x  3 huseyinbabal  staff  96 Jun  7 23:38 ..  
-rw-r--r--  1 huseyinbabal  staff  51 Jun  7 23:39 go.mod
```

Go Modules

All the dependencies are stored inside go.mod file with their VCS urls and versions.



```
huseyinbabal@Huseyi
λ huseyinbabal [~/presentations/golang-app] → cat go.mod
module github.com/huseyinbabal/golang-app

go 1.16

require github.com/google/go-github/v35 v35.3.1-0.20210607195102-edaae9aa26ea
```


Go Modules

You can also see all the dependencies for your application

```
λ huseyinbabal [~/presentations/golang-app] → go list -m all
github.com/huseyinbabal/golang-app
github.com/golang/protobuf v1.3.2
github.com/google/go-cmp v0.5.6
github.com/google/go-github/v35 v35.3.1-0.20210607195102-edae9aa26ea
github.com/google/go-querystring v1.0.0
golang.org/x/crypto v0.0.0-20190308221718-c2843e01d9a2
golang.org/x/net v0.0.0-20190311183353-d8887717615a
golang.org/x/oauth2 v0.0.0-20180821212333-d2e6202438be
golang.org/x/sys v0.0.0-20190215142949-d0b11bdaac8a
golang.org/x/text v0.3.0
golang.org/x/xerrors v0.0.0-20191204190536-9bdfabe68543
google.golang.org/appengine v1.1.0
```

Coding

- You can use **vim-go**, **Goland**, **VSCo**de to start writing Go application.
- You can run your application with `:GoRun` within vim if you are using vim-go plugin
- In Goland you can just run your application after you apply Run Configuration
- In VSCo

#3 Go by Examples

...

Package and Import

- Every resource file starts with a package
- You can state your imports within `import` clause
- If package name is `main`, then it means this is executable rather than a library

```
go main.go > ...
```

```
1 package main
2
3 import (
4     "context"
5     "fmt"
6
7     "github.com/google/go-github/v35/github"
8 )
```

Variable Declaration

```
var count int // Declare
count = 5     // Assign

limit := 15 // Declare and Assign

numbers := []int{1, 2, 3} // Slice with initialization

nums := make([]int, 3) // Allocate slice with 3 ints
nums[0] = 1           // Add to slice

var twodim [][]float64 // 2D slice

students := map[int]string{1: "John", 2: "Doe"}
```

Function Declaration

```
func Sum(a int, b int) int { return a + b }
```

```
func GetCoordinates() (float64, float64) { return 34.565666, 56.123123 }
```

```
func doSomething() {} // this is private
```

```
result := functions.Sum( a: 2, b: 3)
```

```
x, y := functions.GetCoordinates()
```

Control Structures

```
func If(category string) {  
    if category == "shoes" {  
        fmt.Println(a...: "Shoe!")  
    } else {  
        fmt.Println(a...: "Not a shoe!")  
    }  
}  
  
func Switch(device string) {  
    switch device {  
    case "PC":  
        fmt.Println(a...: "Stay at home!")  
    case "LAPTOP":  
        fmt.Println(a...: "You can go outside!")  
    default:  
    }  
}
```



Control Structures

```
func For() {  
    for i := 0; i < 10; i++ {  
        if i%2 == 0 {  
            continue  
        }  
        fmt.Println(i)  
    }  
  
    counter := 0  
    for {  
        if counter == 10 {  
            break  
        }  
        counter++  
    }  
}
```


Control Structures

```
func ForEach() {  
    numbers := []int{2, 4, 6, 8}  
    for _, number := range numbers {  
        fmt.Println(number)  
    }  
  
    countries := map[string]string{"TR": "Turkey", "GB": "London"}  
    for code, country := range countries {  
        fmt.Println(code + "=" + country)  
    }  
}
```

#4 Testing

...

Testing Library

- You can use built-in `testing` package or third-party something like `testify` if you like assertion statements
- If you need to mock dependent components, you can use `mockery` or `golang/mock`

Unit Test

```
func TestFibonacci(t *testing.T) {
    result := Fibonacci( number: 15)
    if result != 610 {
        t.Errorf( format: "Want 610, got %d", result)
    }
}

func TestFibonacciInBatch(t *testing.T) {
    var parameters = []struct{
        input int
        expected int
    } {
        { input: 1,  expected: 1},
        { input: 5,  expected: 5},
        { input: 13, expected: 233},
        { input: 7,  expected: 13},
        { input: 8,  expected: 21},
    }

    for _, parameter := range parameters{
        actual := Fibonacci(parameter.input)
        if actual != parameter.expected {
            t.Errorf( format: "Wanted %d, got %d", parameter.expected, actual)
        }
    }
}
```

Coverage

```
λ huseyinbabal [~/presentations/golang-app] → go test ./... -cover -coverprofile=coverage.out
?      github.com/huseyinbabal/golang-app      [no test files]
?      github.com/huseyinbabal/golang-app/conditionals [no test files]
ok     github.com/huseyinbabal/golang-app/fibonacci    0.348s coverage: 100.0% of statements
?      github.com/huseyinbabal/golang-app/functions    [no test files]
?      github.com/huseyinbabal/golang-app/variable     [no test files]
```

#5 CI/CD

...

Building Artifacts

Golang has built-in tools to generate artifacts based on distributions. To see supported platforms;

```
1 go tool dist list
```

aix/ppc64	freebsd/arm	linux/mips64le	openbsd/386
android/386	freebsd/arm64	linux/mipsle	openbsd/amd64
android/amd64	illumos/amd64	linux/ppc64	openbsd/arm
android/arm	js/wasm	linux/ppc64le	openbsd/arm64
android/arm64	linux/386	linux/riscv64	plan9/386
darwin/amd64	linux/amd64	linux/s390x	plan9/amd64
darwin/arm64	linux/arm	netbsd/386	plan9/arm
dragonfly/amd64	linux/arm64	netbsd/amd64	solaris/amd64
freebsd/386	linux/mips	netbsd/arm	windows/386
freebsd/amd64	linux/mips64	netbsd/arm64	windows/amd64
			windows/arm

Building Artifacts

```
1 GOOS=darwin GOARCH=amd64 go build
```


Containerization

```
1 touch Dockerfile
```

```
1 FROM golang:1.13-alpine3.11 as builder
2 RUN mkdir -p /usr/src/app
3 WORKDIR /usr/src/app
4 COPY . .
5 RUN go mod download
6 RUN CGO_ENABLED=0 GOOS=linux go build -o hello
7
8 FROM scratch
9 COPY --from=builder /usr/src/app/hello /bin/hello
10 ENTRYPOINT ["/bin/hello"]
11 EXPOSE 8080
```

Demo Time...

Any Questions?

<http://github.com/huseyinbabal>

