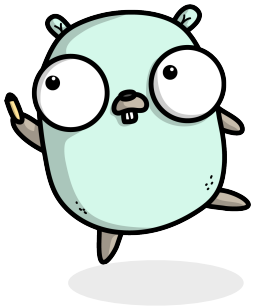
Lionel LONKAP — @lktslionel
Devops Engineer & AWS Cloud Solution Architect

ZenikaLille



ZenikaLille

01

Context

```
$ find $MYPROJ -type f -name "*.go" | grep -Ev '(vendor|test)' | xargs -IA wc -l A | sort -nr | head -3

    142 /Users/lktslionel/Workspaces/personal/golang/src/github.com/lktslionel/envctl/env/repository.go
    110 /Users/lktslionel/Workspaces/personal/golang/src/github.com/lktslionel/awssel/env/ssm_store.go
     83 /Users/lktslionel/Workspaces/personal/golang/src/github.com/lktslionel/awssel/cmd/load.go
```

Command

Line

Interface

Command

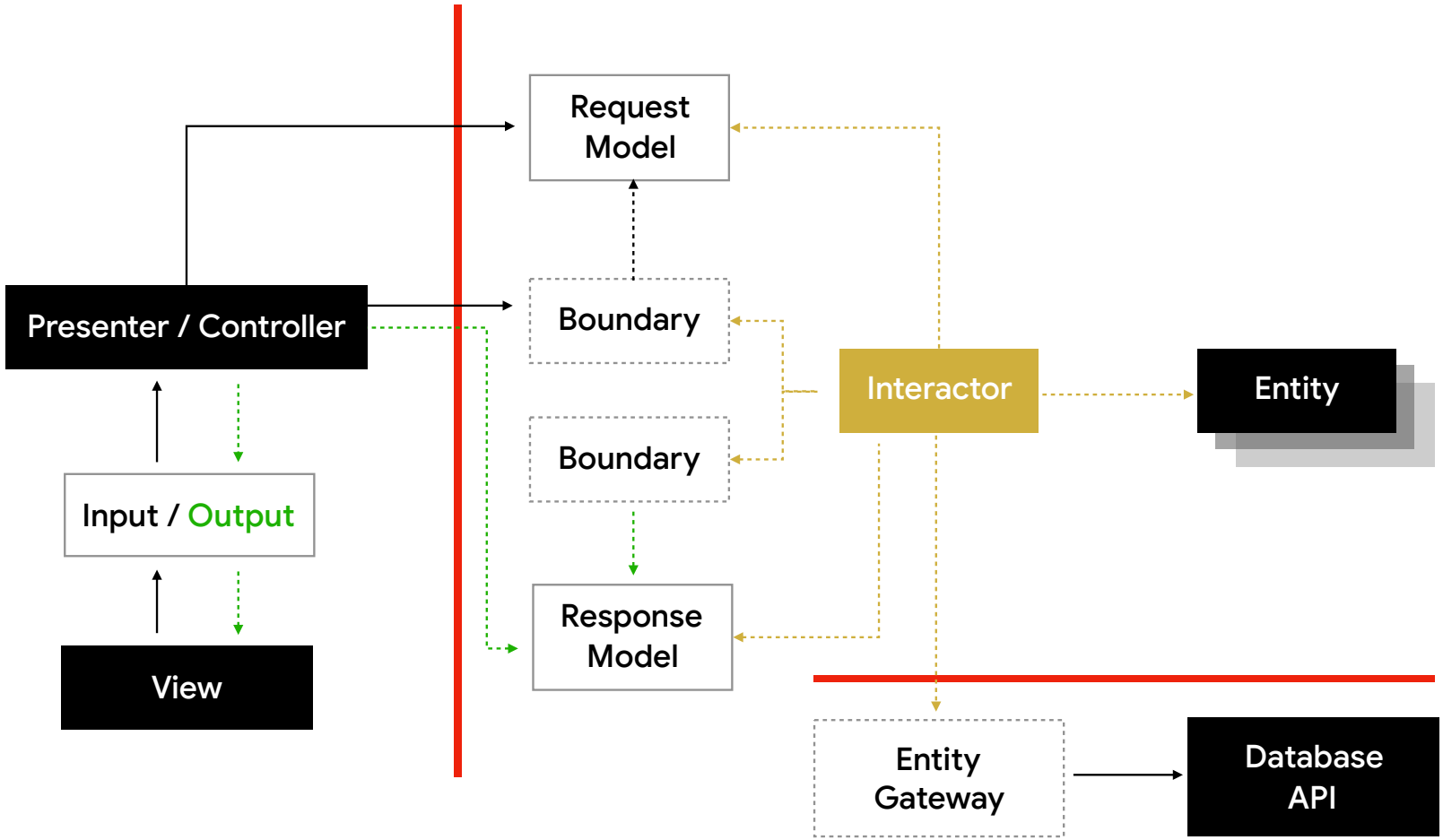
Language

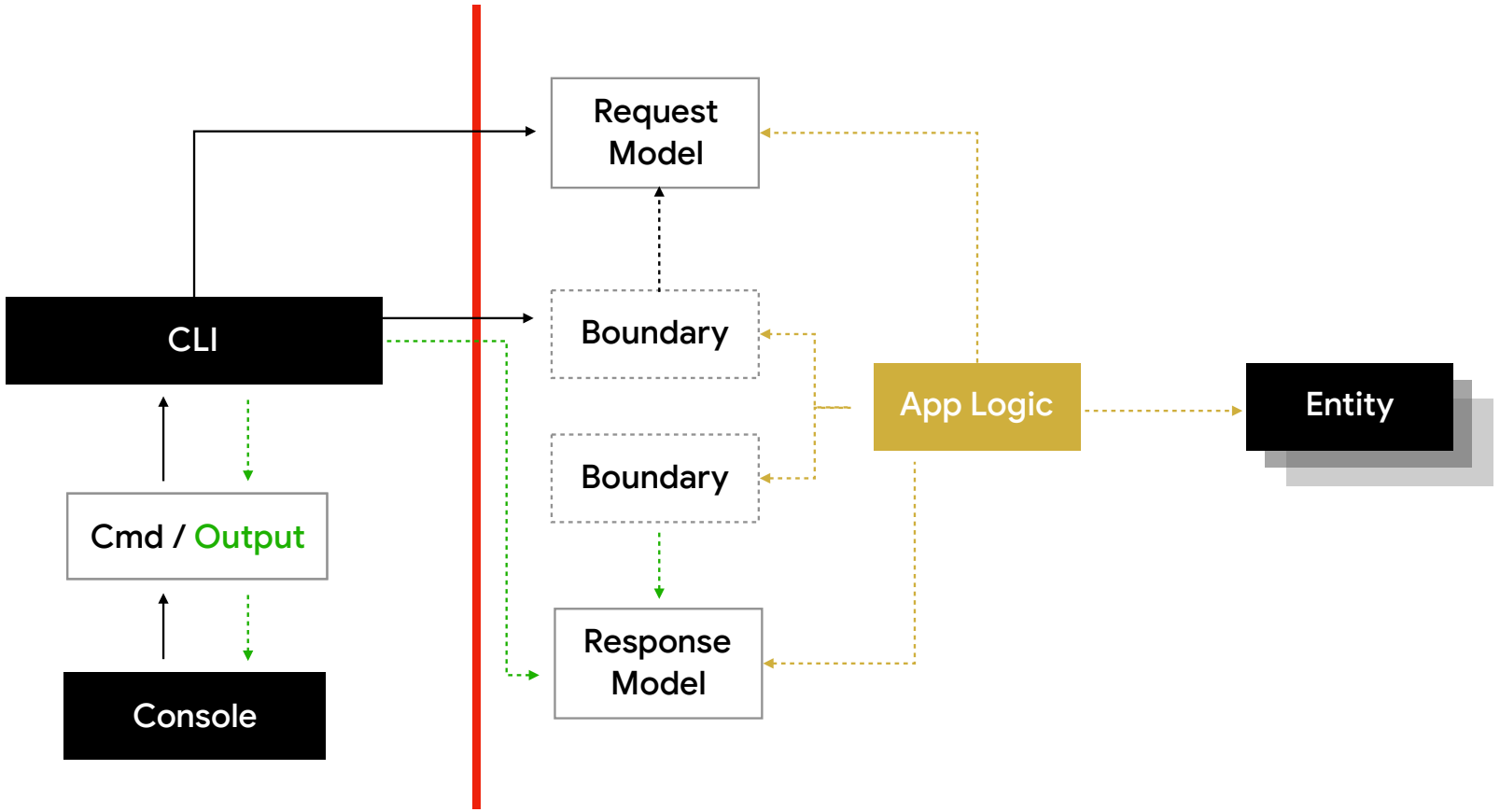
Interpreter

ZenikaLille

> **CLI** is a means of interacting with a program where the user issues commands to the program in the form of successive lines of text ...
>
> — Wikipedia

> **CLI** is a means of **interacting** with a **program** where the user issues **commands** to the program in the form of successive **lines** of text ...
>
> — Wikipedia


Request Model
Presenter / Controller
Boundary
Interactor
Entity
Boundary
Input / Output
Response Model
View
Entity Gateway
Database API


Request Model
CLI
Boundary
App Logic
Entity
Boundary
Cmd / Output
Response Model
Console

CONTEXT

CLI Types

Simple tool

```
$ cli-app   [FLAGS]  <ARGUMENT> ...
```

Example: *rm, ls, find*

CONTEXT

CLI Types

Command suite

```
$ cli-app   [sub-command]
```

Example *: kubectl, docker*

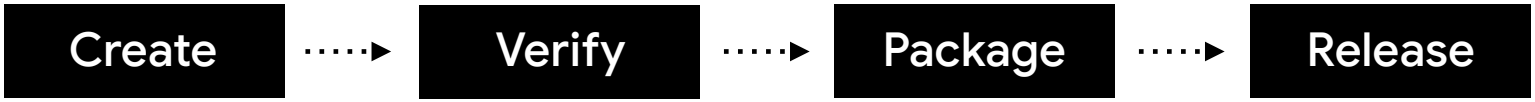
02

Manifesto

1. First Class Application
2. Helpful
3. Play well with Others

Rule #1 • First Class Application

Software development workflow



Arguments & Options (Flags)

```
$ cli-app   [FLAGS]  <ARGUMENT> ...
```

Arguments & Options (Flags)

Use Short Options

```
-h      For help
-o      For Output
-c      For Config file
```

Arguments & Options (Flags)

Use Long Options

```
--help              For help
--verbose           For Log verbosity
--long-option       For Any other opt name
```

Configurable

- Env vars
- Arguments & Options
- Config file

Source : github.com/spf13/pflag

Rule #2 • Helpful

Usage text

```
$ go --help || go -h

Go is a tool for managing Go source code.
Usage:
    go command [arguments]
The commands are:
    build       compile packages and dependencies
    clean       remove object files and cached files
    doc         show documentation for package or symbol
    env         print Go environment information
. . .
```

Suggestions

```
$ git tg

  git: 'tg' is not a git command. See 'git --help'.

  The most similar command is
     tag
```

Rule #3 • Play well with Others

Redirections

- Stdin
- Stdout
- Stderr

Source : github.com/spf13/pflag

Exit codes

- Success (0)
- Error (!0)

Source : github.com/spf13/pflag

Grep-able

Log one line record

```
$ cli-app   [FLAGS]  <ARGUMENT> ... | grep [opts] <args>
```

Source : github.com/spf13/pflag

Cut-able

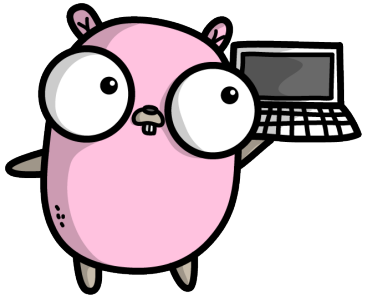
Fields delimiter

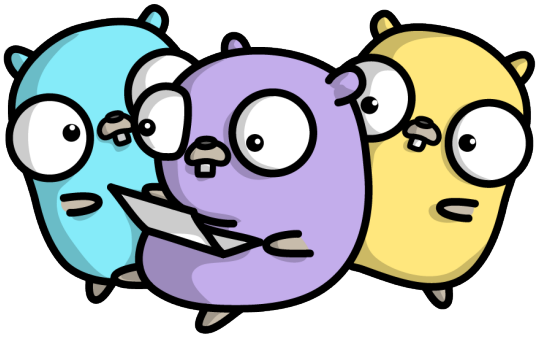
```
$ cli-app [FLAGS] <ARGUMENT> ... | cut -d<DELIMITER> -f<NUM_COL>
```

Source : github.com/spf13/pflag

03

Why Go ?





Source : github.com/ashleymcnamara/gophers

ZenikaLille

Because ...

- **Easy & Fun**
- **Cross Compilation**
- **Standard Library**

Cross compilation

Windows

```
$ GOARCH=amd64 GOOS=windows go build -v
```

MacOS

```
$ GOARCH=amd64 GOOS=darwin go build -v
```

Linux

```
$ GOARCH=amd64 GOOS=linux go build -v
```

04 Cobra

Source : github.com/ashleymcnamara/gophers

Library for creating powerful modern CLI applications

Source : github.com/spf13/cobra

ZenikaLille

COBRA

Delve

OpenShift

Hugo

Kubernetes

Docker

rkt

etcd

Source : github.com/spf13/cobra

ZenikaLille

Install

```
$ go get -u github.com/spf13/cobra/cobra
```

COBRA

Create a project

```
$ cobra init cli-app

  Your Cobra application is ready at
  /Users/lktslionel/Workspaces/personal/golang/src/cli-app.

  Give it a try by going there and running `go run main.go`.
  Add commands to it by running `cobra add [cmdname]`.
```

Root command

```
var rootCmd = &cobra.Command{
   Use:   “cli-app",
   Short: "A brief description of your application",
   Long: `A longer description …`,

   Run: func(cmd *cobra.Command, args []string) { },
}
```

COBRA

Flags

```
var globalVar <TYPE>

func init() {
   . . .
   rootCmd.PersistentFlags().<TYPE>Var(&globalVar, “<long>“, “<default>“, "<desc>")
   rootCmd.Flags().<TYPE>(“<long>“, <default>, “<desc>")
   rootCmd.Flags().<TYPE>P(“<long>“, “<short>“, <default>, “<desc>")
}
```

Source : github.com/spf13/pflag

COBRA

Usage text

```
$ cli-app   -h|—help
```

Provided Automatically

COBRA

Suggestions

```
$ cli-app remove

Error: unknown command "remove" for “cli-app"

Did you mean this?
        delete

Run ‘cli-app help' for usage.
```

COBRA

Suggestions

```
command.DisableSuggestions = true | false
command.SuggestFor = []String{}
```

Provided Automatically

COBRA

Run

```
package main

import “cli-app/cmd"

func main() {
  cmd.Execute()
}
```

```
.
├── LICENSE
├── cmd
|   └── root.go
└── main.go
```

COBRA

Run

```
$ go run main.go

# Or

$ go install .
$ cli-app
```

```
.
├── LICENSE
├── cmd
|   └── root.go
└── main.go
```

Source : github.com/spf13/pflag

ZenikaLille

COBRA

Add a sub-command

```
$ cobra add <sub-command-name>
```

```
.
├── LICENSE
├── cmd
|   ├── root.go
|   └── subCommandName.go
└── main.go
```

Source : github.com/spf13/pflag

ZenikaLille

COBRA

Add a sub-command

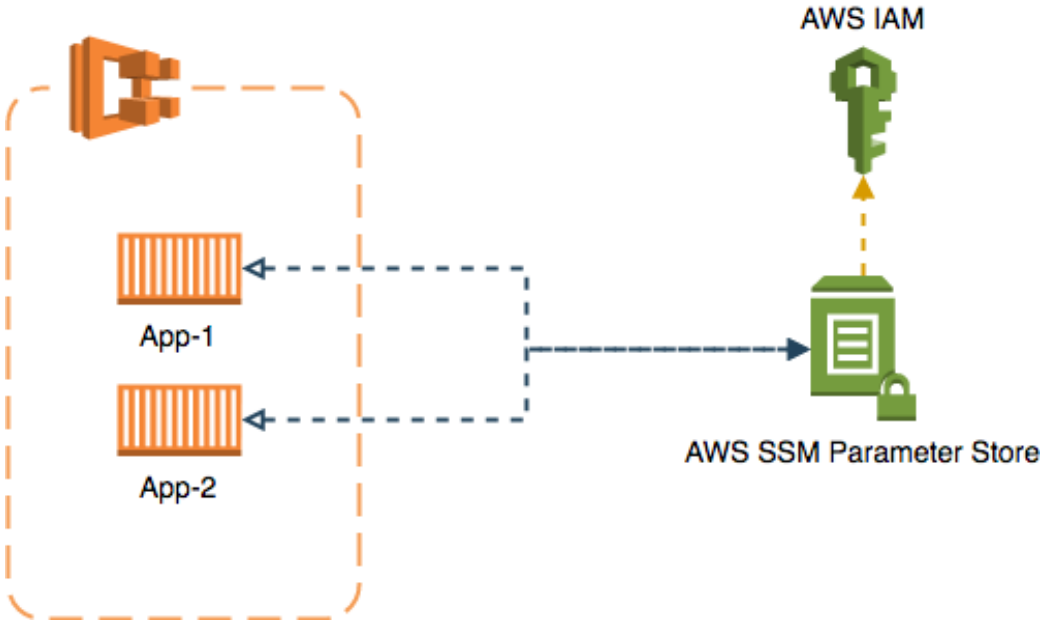
```
package cmd
. . .
var subCommandNameCmd = &cobra.Command{
  Use:   "subCommandName",
  . . .
  Run: func(cmd *cobra.Command, args []string) {
      fmt.Println("subCommandName called")
  },
}

func init() {
  rootCmd.AddCommand(subCommandNameCmd)
  . . .
```

Source : github.com/spf13/pflag

05

Case Study

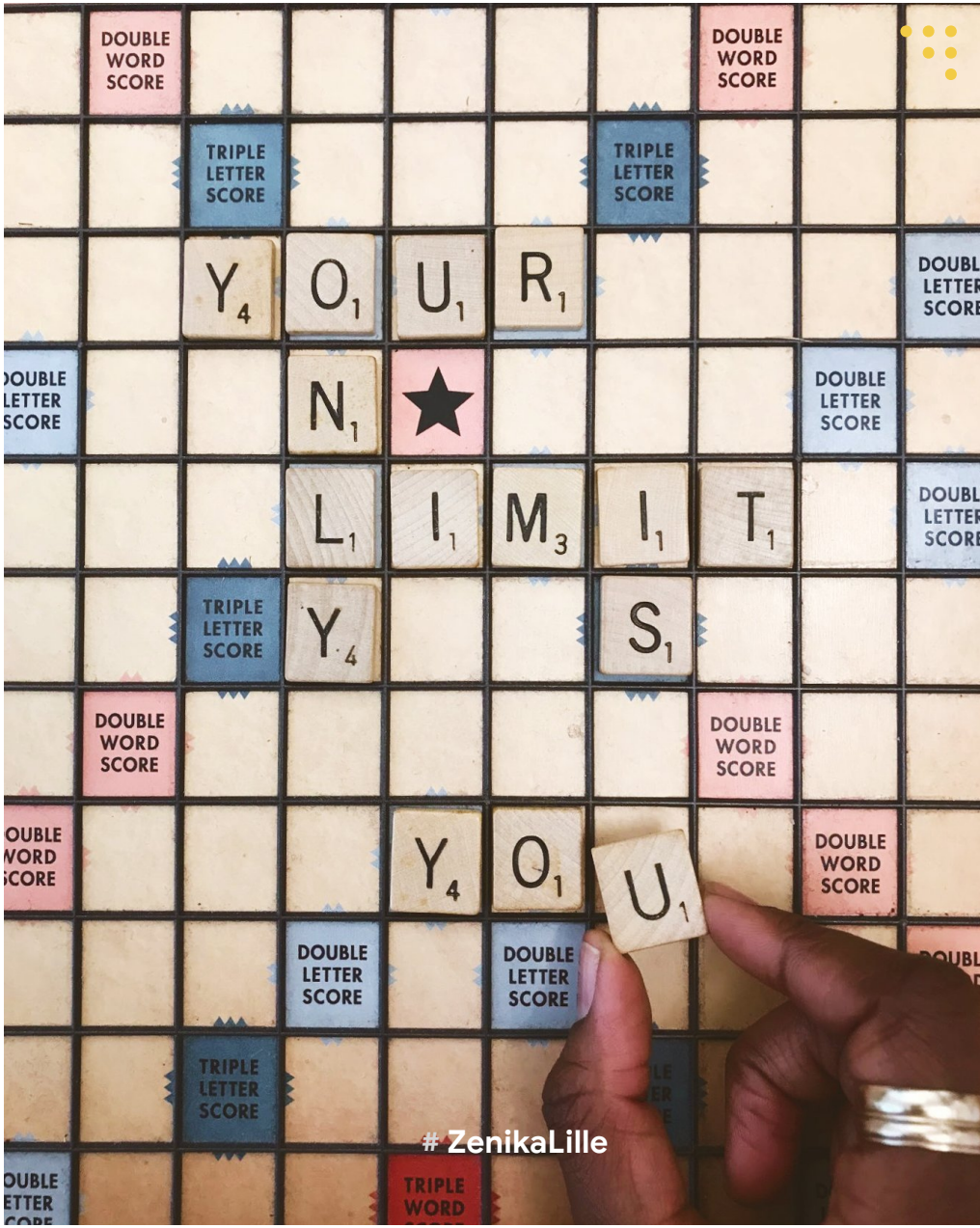


Source : github.com/lktslionel/awssel

06

Demo

Thanks !


DOUBLE WORD SCORE
TRIPLE LETTER SCORE
DOUBLE LETTER SCORE
TRIPLE WORD SCORE
YOUR
NOT
LIMIT
YOU
ZenikaLille