/evrone.

FROM DEVELOPMENT TO DISTRIBUTION

CLI IN GO

mgrachev

mgrachev

BACKEND, TEAMLEAD, DEVOPS

Grachev Mikhail

/evrone.

PLAN

- Development CLI
  - Libraries
  - Problems
- Distribution CLI
  - Instruments
  - Release

CLI

Libraries

spf13/cobra

Star 10,741

- Code generation
- Nested commands
- Flags and Arguments
- Intelligent suggestions
- Bash/Zsh-completion
- Hooks

spf13/cobra

CODE GENERATION

```
$ cobra init github.com/<USER>/<APP_NAME>
Your Cobra application is ready at
<GO_PATH>/src/github.com/<USER>/<APP_NAME>

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

$ cobra add <COMMAND_NAME>
<COMMAND_NAME> created at .../<USER>/<APP_NAME>/cmd/<COMMAND_NAME>.go
```

https://github.com/spf13/cobra/pull/817

CODE GENERATION

```
$ tree github.com/<USER>/<APP_NAME>
├── LICENSE
├── cmd
│   └── root.go
└── main.go

1 directory, 3 files
```

COMMANDS

```go
package cmd

var rootCmd = &cobra.Command{
    Use:   "hugo",
    Short: "Hugo is a very fast static site generator",
    Run: func(cmd *cobra.Command, args []string) {
        // Do Stuff Here
    },
}

func Execute() {
    if err := rootCmd.Execute(); err != nil {
        fmt.Println(err)
        os.Exit(1)
    }
}
```

COMMANDS

```go
package cmd

func init() {
    rootCmd.AddCommand(versionCmd)
}

var versionCmd = &cobra.Command{
    Use:   "version",
    Short: "Print the version number of Hugo",
    Long:  `All software has versions. This is Hugo's`,
    Run: func(cmd *cobra.Command, args []string) {
        fmt.Println("Hugo Static Site Generator v0.9 -- HEAD")
    },
}
```

FLAGS

```
$ go run main.go --help
...

Usage:
  <APP_NAME> [flags]

Flags:
      --config string   config file ($HOME/.config.yaml)
  -h, --help            help for <APP_NAME>
  -t, --toggle          Help message for toggle
```

```go
// Persistent Flag
rootCmd.PersistentFlags().StringVar(&cfgFile, "config", "", "config file ($HOME/.config.yaml)")

// Local Flag
rootCmd.Flags().BoolP("toggle", "t", false, "Help message for toggle")
```

ARGUMENTS

```
$ go run main.go
Usage:
  <APP_NAME> [flags]

Flags:
  -h, --help            help for <APP_NAME>

requires at least 1 arg(s), only received 0
exit status 1
```

```go
var printCmd = &cobra.Command{
    // ...
    Args: cobra.MinimumNArgs(1),
    Run: func(cmd *cobra.Command, args []string) {
        fmt.Println("Print: " + strings.Join(args, " "))
    }
}
```

INTELLIGENT SUGGESTIONS

```
$ hugo srever
Error: unknown command "srever" for "hugo"

Did you mean this?
        server

Run 'hugo --help' for usage.
```

INTELLIGENT SUGGESTIONS

```
$ kubectl remove
Error: unknown command "remove" for "kubectl"

Did you mean this?
        delete

Run 'kubectl help' for usage.
```

HOOKS

```go
var rootCmd = &cobra.Command{
    // ...
    PersistentPreRun: func(cmd *cobra.Command, args []string) {
        fmt.Printf("Inside rootCmd PersistentPreRun with args: %v\n", args)
    },
    PreRun: func(cmd *cobra.Command, args []string) {
        fmt.Printf("Inside rootCmd PreRun with args: %v\n", args)
    },
    Run: func(cmd *cobra.Command, args []string) {
        fmt.Printf("Inside rootCmd Run with args: %v\n", args)
    },
    PostRun: func(cmd *cobra.Command, args []string) {
        fmt.Printf("Inside rootCmd PostRun with args: %v\n", args)
    },
    PersistentPostRun: func(cmd *cobra.Command, args []string) {
        fmt.Printf("Inside rootCmd PersistentPostRun with args: %v\n", args)
    },
}
```

spf13/viper

Star 7,654

- JSON, TOML, YAML, HCL, Java properties
- ENV
- Etcd и Consul
- Live watching

EXAMPLE

```
viper.SetConfigType("toml")
viper.SetConfigName("config")
viper.AddConfigPath("$HOME/.<APP_NAME>")
viper.AddConfigPath(".")
viper.ReadInConfig()

viper.SetDefault("verbose", false)
viper.Set("verbose", true)
viper.Set("path", "/etc")

viper.IsSet("path") // => true
viper.GetBool("verbose")
viper.GetString("path")

viper.WriteConfig()
```

ENV

```
viper.SetEnvPrefix("go")
viper.BindEnv("path")

os.Setenv("GO_PATH", "/root")
viper.Get("path")

// Or
viper.SetEnvPrefix("go")
viper.AutomaticEnv()

viper.Get("os")
viper.Get("path")
```

AlecAivazis/survey

Star 1,066

```
$ go run main.go
? What is your name? Mikhail
? Choose your favourite conference: [Use arrows to move, type to filter]
  DevConf
  RubyRussia
> GopherCon Russia ❤️
```

PASSWORD

```go
password := ""
prompt := &survey.Password{
    Message: "Please type your password",
}
survey.AskOne(prompt, &password, nil)
```

```
$ go run main.go
? Please type your password ********
```

CONFIRM

```go
result := false
prompt := &survey.Confirm{
    Message: "Do you want to delete it?",
}
survey.AskOne(prompt, &result, nil)
```

```
$ go run main.go
? Do you want to delete it? (y/N)
```

SELECT

```go
lang := ""
prompt := &survey.Select{
    Message: "Choose your favourite language:",
    Options: []string{"Ruby", "Rust", "Go 💪 "},
}
survey.AskOne(prompt, &lang, nil)
```

```
$ go run main.go
? Choose your favourite language:  [Use arrows to move, type to filter]
  Ruby
  Rust
> Go 💪
```

avelino/awesome-go#standard-cli

avelino/awesome-go#configuration

avelino/awesome-go#advanced-console-uis

Problems

CROSS-PLATFORM PATHS

```go
func ConfigPath() string {
    if runtime.GOOS == "windows" {
        configPath := homeDir + "\\" + configName
    } else {
        configPath := homeDir + "/" + configName
    }
    return configPath
}
```

CROSS-PLATFORM PATHS

```go
// Option 1
import "os"
func ConfigPath() string {
    return homeDir + string(os.PathSeparator) + configName
}

// Option 2
import "path/filepath"
func ConfigPath() string {
    return filepath.FromSlash(homeDir + "/" + configName)
    // Or
    return filepath.Join(homeDir, configName)
}
```

HOME DIRECTORY

```go
package main

import (
    "os/user"
    "fmt"
    "log"
)

func main() {
    usr, err := user.Current()
    if err != nil {
        log.Fatal(err)
    }
    fmt.Println(usr.HomeDir)
}
```

HOME DIRECTORY

GO 1.12

```go
homeDir, err := os.UserHomeDir()
// unix - HOME
// windows - USERPROFILE
```

mitchellh/go-homedir

Star 633

```
homedir.Dir()
// => /Users/mgrachev

homedir.Expand("~/dev")
// => /Users/mgrachev/dev
```

FILE SYSTEM

```go
func NewConfig() *Config {
    fullConfigPath := configPath(AppName)
    return &Config{path: fullConfigPath}
}

func (c *Config) CreateFile() error {
    _, err := os.Create(c.path)
    return err
}

func (c *Config) RemoveFile() error {
    return os.Remove(c.path)
}
```

spf13/afero

Star 1,948

AFERO

EXAMPLE

```go
var AppFs = afero.NewMemMapFs()
// Or
var AppFs = afero.NewOsFs()

AppFs.Open('file.txt')
AppFs.MkdirAll("tmp/foo", 0755))

bp := afero.NewBasePathFs(afero.NewOsFs(), "/base/path")
bp.Create("/file.txt") // => /base/path/file.txt

fs := afero.NewReadOnlyFs(afero.NewOsFs())
fs.Create("/file.txt") // => syscall.EPERM
```

TESTING

```go
func TestExist(t *testing.T) {
    appFS := afero.NewMemMapFs()
    appFS.MkdirAll("src/a", 0755)
    afero.WriteFile(appFS, "src/a/b", []byte("file b"), 0644)
    afero.WriteFile(appFS, "src/c", []byte("file c"), 0644)
    name := "src/c"
    _, err := appFS.Stat(name)
    if os.IsNotExist(err) {
        t.Errorf("file \"%s\" does not exist.\n", name)
    }
}
```

Distribution

HOMEBREW

https://brew.sh

```
$ brew install <APP_NAME>
$ brew update
$ brew upgrade <APP_NAME>
$ brew uninstall <APP_NAME>
```

REPOSITORY

- https://github.com/<USER>/homebrew-<APP_NAME>
- https://github.com/<USER>/<APP_NAME>

```
$ tree homebrew-test
└── Formula
    └── <APP_NAME>.rb

1 directory, 1 file
```

FORMULA

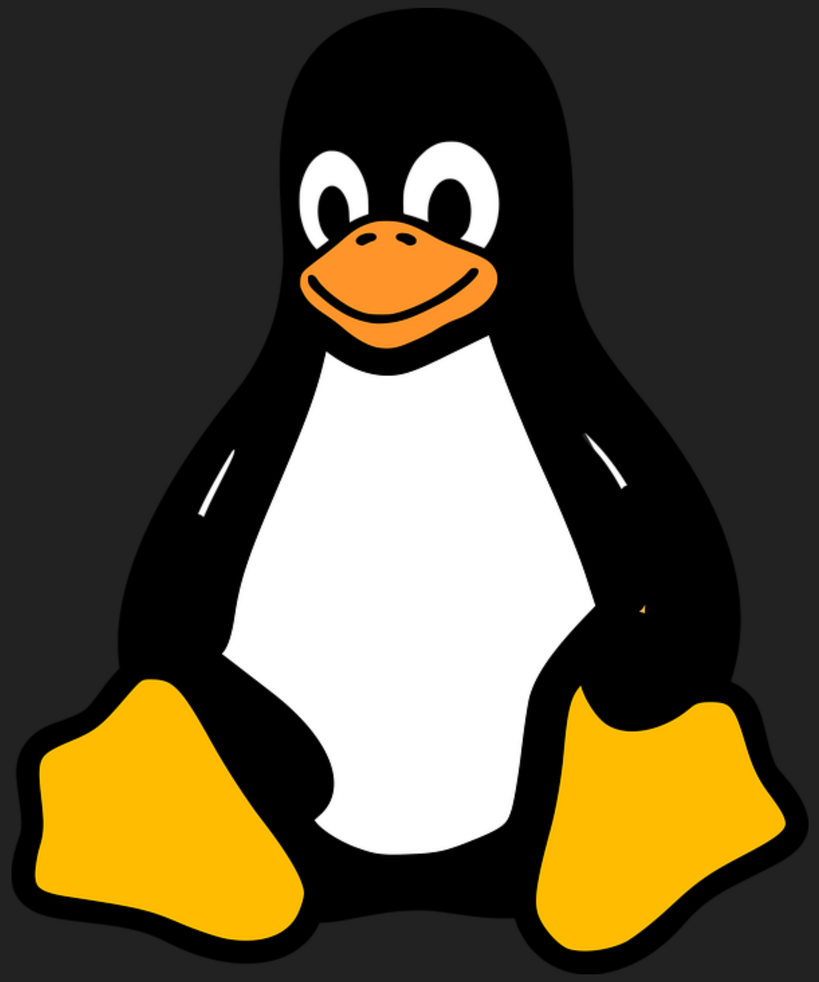
```
1 class TestApp < Formula
2   desc "Test application"
3   homepage "https://github.com/mgrachev/homebrew-test"
4   url "https://github.com/mgrachev/test/releases/download/v1.0.0.tar.gz"
5   version "1.0.0"
6   sha256 "0514d0e54151922a76388e9753077b6265df9edabd48f583eba3b990d0d82c92"
7
8   def install
9     bin.install "test_app"
10  end
11 end
```

https://brew.sh

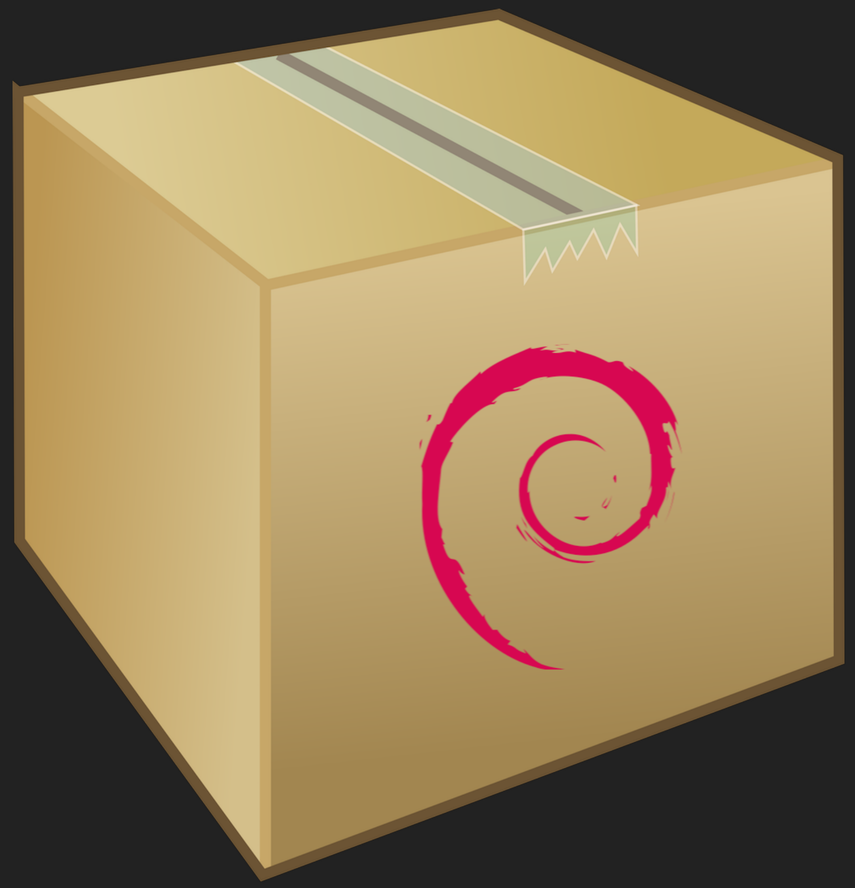
INSTALL

```
$ brew tap <USER>/<APP_NAME>
# Or
$ brew tap <USER>/<APP_NAME> URL

# Install/Update
$ brew install <APP_NAME>
$ brew update
$ brew upgrade <APP_NAME>

# Uninstall
$ brew uninstall <APP_NAME>
$ brew untap <USER>/<APP_NAME>
```

LINUX



DEB / RPM



SNAPCRAFT

https://snapcraft.io

https://snapcraft.io

SNAPCRAFT

- Support Linux distributions
- Isolation
- No conflicts

SNAPCRAFT

- snap
- snapd
- snapcraft
- https://snapcraft.io/store

SNAP

```
$ mount | grep simple
squashfuse on /snap/simple/x1 type fuse.squashfuse (ro)
```

SNAP

```
$ tree /snap/<SNAP_NAME>
├── current -> x1
└── x1
    ├── meta
    │   └── snap.yaml
    └── <SNAP_NAME>

3 directories, 2 files
```

https://snapcraft.io

SNAP

```yaml
# snap.yaml
name: simple
version: 1.0.0
summary: Simple application
description: Simple test application
confinement: strict

architectures:
- amd64

apps:
  simple:
    command: simple
    plugs:
    - home
```

RESTRICTIONS

- Strict
- Classic
- Devmode

https://snapcraft.io

SNAP

```
$ snap list
$ snap info vlc
$ snap find "media player"

$ sudo snap install vlc
$ sudo snap install --channel=edge vlc

$ sudo snap refresh vlc
$ sudo snap revert vlc

$ sudo snap disable vlc
$ sudo snap enable vlc
```

SNAPCRAFT

```
$ sudo snap install snapcraft --classic

$ snapcraft init
Created snap/snapcraft.yaml.
Go to https://docs.snapcraft.io/the-snapcraft-format/8337
for more information about the snapcraft.yaml format.
```

SNAPCRAFT

```yaml
name: my-snap-name
base: core18
version: "0.1"
summary: Single-line elevator pitch for your amazing snap
description: "This is my-snap's description"
grade: devel
confinement: devmode

parts:
  my-part:
    plugin: nil

apps:
  my-snap-name:
    command: my-snap-name
    plugs:
    - home
```

SNAPCRAFT

```
$ snapcraft --debug
Launching a VM.
Launched: snapcraft-my-snap-name [...]
Pulling my-part
Building my-part
Staging my-part
Priming my-part
Snapping 'my-snap-name' |
Snapped my-snap-name_0.1_amd64.snap

$ sudo snap install my-snap-name_0.1_amd64.snap --dangerous --devmode
my-snap-name 0.1 installed
```

SNAPCRAFT

```
$ snapcraft login
$ snapcraft push --release=stable my-snap-name_0.1_amd64.snap

$ sudo snap install my-snap-name
```

SCOOP

https://scoop.sh

```
$ scoop search ssh
$ scoop status curl

$ scoop install git curl ssh

$ scoop list
$ scoop update

$ scoop uninstall curl
```

MANIFEST

```
// <APP_NAME>.json
{
    "version": "1.0.0",
    "architecture": {
        "64bit": {
            "url": "https://github.com/mgrachev/test/releases/download/v1.0.0.tar.gz",
            "bin": "test_app.exe",
            "hash": "85bf7dabadf2e3803197905ed2612ecbb4ed276650f4c31e8498942e17f92209"
        }
    },
    "homepage": "https://github.com/mgrachev/test",
    "description": "Test application",
    "persist": [
        ".config.toml"
    ]
}
```

https://scoop.sh

MANIFEST

```
$ scoop bucket add <APP_NAME> <APP_URL>
$ scoop install <APP_NAME>
```

Release

goreleaser/goreleaser

Star 3,813

https://goreleaser.com

GORELEASER.YML

```
$ goreleaser init

   • Generating .goreleaser.yml file
   • config created; please edit accordingly to your needs file=.goreleaser.yml
```

COMPILATION

```yaml
# .goreleaser.yml
before:
  hooks:
    - go mod download

builds:
  -
    binary: program
    ldflags:
      - -s -w -X main.build={{.Version}}
    env:
      - CGO_ENABLED=0
    goos:
      - linux
      - windows
    goarch:
      - amd64
```

ARCHIVING

```yaml
# .goreleaser.yml
archive:
  name_template: "{{ .ProjectName }}_{{ .Version }}_{{ .Os }}_{{ .Arch }}"

  replacements:
    amd64: 64-bit
    darwin: macOS

  # Formats: tar.gz, gz, zip and binary
  format: zip

  format_overrides:
    - goos: windows
      format: zip

  files:
    - README.md
    - CHANGELOG.md
```

S3

```yaml
# .goreleaser.yml
s3:
  - bucket: my-bucket
    region: us-east-1
    folder: "foo/bar/{{.Version}}"
    profile: my-profile
    endpoint: "http://minio.foo.com"
    acl: public-read
```

HOMEBREW

```yaml
# .goreleaser.yml
brew:
  url_template: "http://site.com/releases/{{ .Tag }}/{{ .ArtifactName }}"
  homepage: "https://example.com/"
  description: "Software to create fast and easy drum rolls."

  github:
    owner: user
    name: homebrew-tap
  commit_author:
    name: goreleaserbot
    email: goreleaser@carlosbecker.com

  dependencies:
    - git
  install: 'bin.install "program"'
```

SCOOP

```yaml
# .goreleaser.yml
scoop:
  url_template: "http://site.com/releases/{{ .Tag }}/{{ .ArtifactName }}"

  bucket:
    owner: user
    name: scoop-bucket
  commit_author:
    name: goreleaserbot
    email: goreleaser@carlosbecker.com

  homepage: "https://example.com/"
  description: "Software to create fast and easy drum rolls."

  persist:
  - "config.toml"
```

goreleaser/nfpm

Star 490

```yaml
# .goreleaser.yml
nfpm:
  name_template: "{{ .ProjectName }}_{{ .Version }}_{{ .Os }}_{{ .Arch }}"
  homepage: https://example.com/
  description: Software to create fast and easy drum rolls.

  formats:
    - deb
    - rpm
  dependencies:
    - git

  files:
    "scripts/etc/init.d/": "/etc/init.d"
  scripts:
    postinstall: "scripts/postinstall.sh"
```

SNAPCRAFT

```yaml
# .goreleaser.yml
snapcraft:
  name_template: "{{ .ProjectName }}_{{ .Version }}_{{ .Os }}_{{ .Arch }}"

  name: drumroll
  publish: true
  summary: Software to create fast and easy drum rolls.
  grade: stable
  confinement: strict

  apps:
    drumroll:
      plugs: ["home", "network"]
      daemon: simple
      args: --foo
```

RELEASE

```
$ goreleaser --snapshot --skip-publish --rm-dist

$ git tag -a v1.0.0 -m "First release"
$ git push origin v1.0.0

$ goreleaser
```

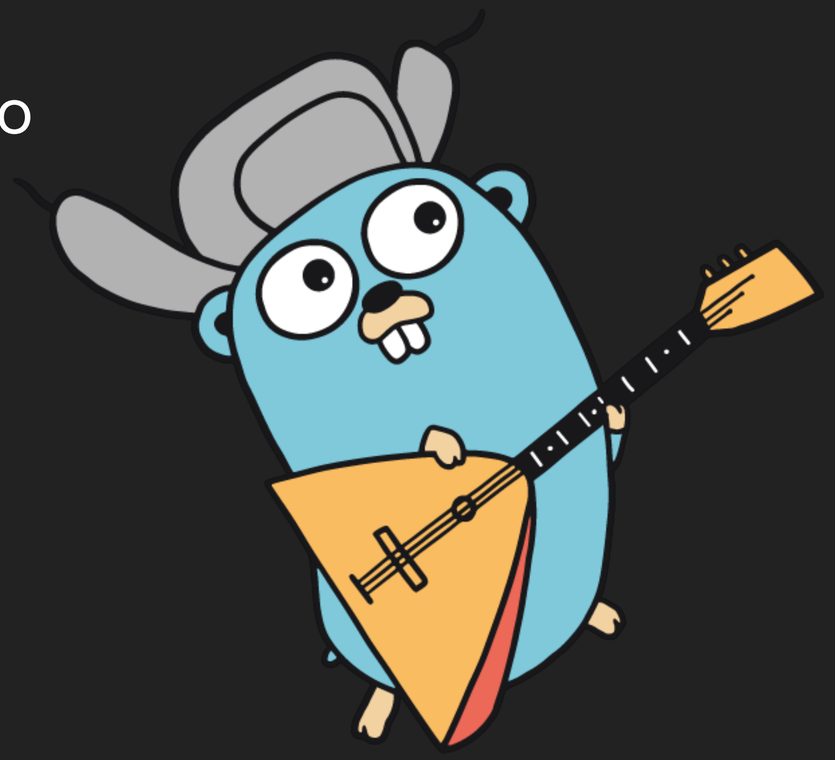
goreleaser/godownloader

Star 192

https://install.goreleaser.com

SUMMARY

- Development: cobra, viper, survey, go-homedir, afero
- Distribution: homebrew, snapcraft, scoop
- Release: goreleaser, nfpm, godownloader



mgrachev

mgrachev

i@mgrachev.com

https://github.com/mgrachev/gophercon-russia-2019