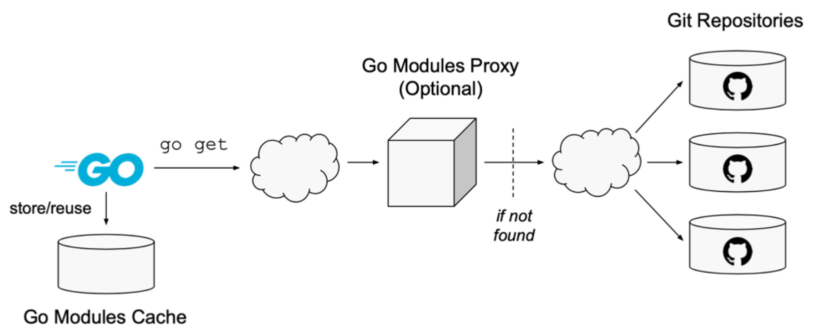
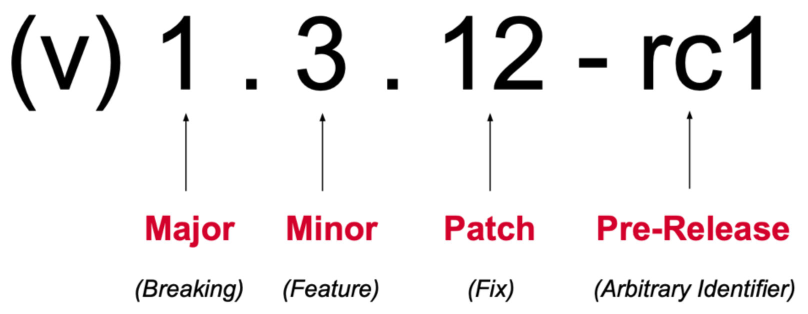
Twitter: @sakthis02

Golang Modules Cheat Sheet

Workflow of Go Modules

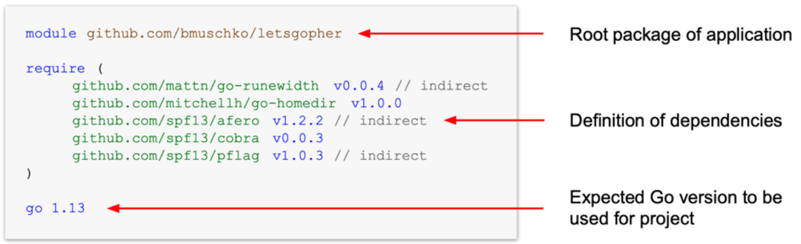


Selects packages [or] dependencies based on semantic versioning



Anatomy of Go Modules

go.mod file



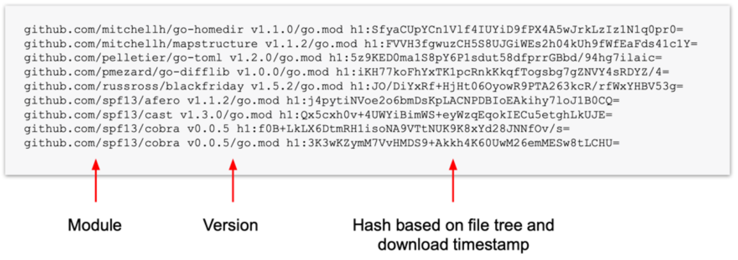
```
module github.com/bmuschko/letsgopher

require (
        github.com/mattn/go-runewidth  v0.0.4 // indirect
        github.com/mitchellh/go-homedir  v1.0.0
        github.com/spf13/afero  v1.2.2 // indirect
        github.com/spf13/cobra  v0.0.3
        github.com/spf13/pflag  v1.0.3 // indirect
)

go 1.13
```

Root package of application
Definition of dependencies
Expected Go version to be used for project

go.sum

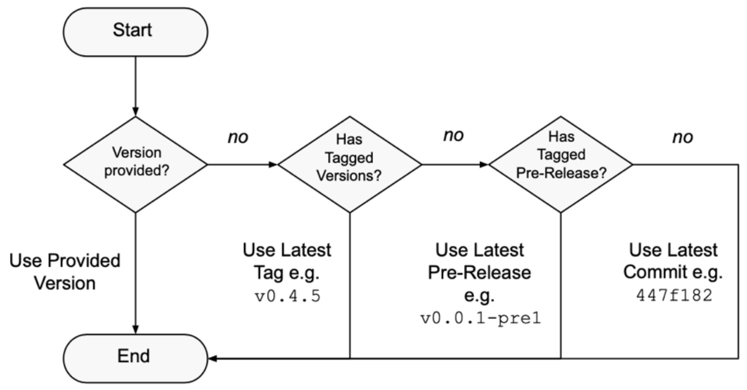


```
github.com/mitchellh/go-homedir v1.1.0/go.mod h1:SfyaCUpYCn1Vlf4IUYiD9fPX4A5wJrkLzIz1N1q0pr0=
github.com/mitchellh/mapstructure v1.1.2/go.mod h1:FVVH3fgwuzCH5S8UJGiWEs2h04kUh9fWfEaFds41c1Y=
github.com/pelletier/go-toml v1.2.0/go.mod h1:5z9KED0ma1S8pY6P1sdut58dfprrGBbd/94hg7ilaic=
github.com/pmezard/go-difflib v1.0.0/go.mod h1:iKH77koFhYxTK1pcRnkKkqfTogsbg7gZNVY4sRDYZ/4=
github.com/russross/blackfriday v1.5.2/go.mod h1:JO/DiYxRf+HjHt06OyowR9PTA263kcR/rfWxYHBV53g=
github.com/spf13/afero v1.1.2/go.mod h1:j4pytiNVoe2o6bmDsKpLACNPDBIoEAkihy7loJ1B0CQ=
github.com/spf13/cast v1.3.0/go.mod h1:Qx5cxh0v+4UWYiBimWS+eyWzqEqokIECu5etghLkUJE=
github.com/spf13/cobra v0.0.5 h1:f0B+LkLX6DtmRH1isoNA9VTtNUK9K8xYd28JNNfOv/s=
github.com/spf13/cobra v0.0.5/go.mod h1:3K3wKZymM7VvHMDS9+Akkh4K60UwM26emMESw8tLCHU=
```

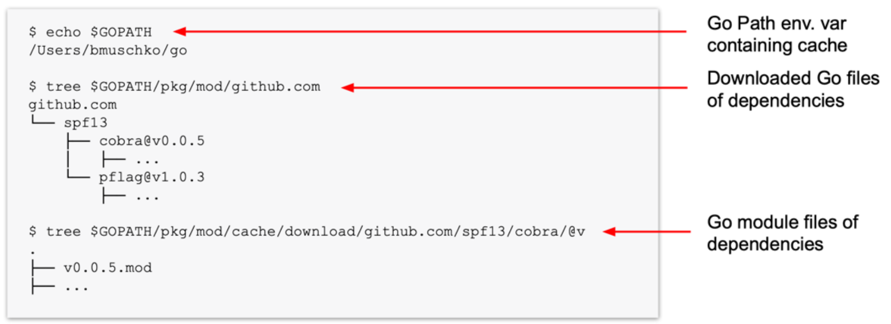
- **go.mod =>** would contain dependency requirements that is needed for our project [or] package.
- **go.sum =>** would contain the expected cryptographic hashes for the dependencies that would ensure
  the same version of dependencies gets downloaded every time [reproducible builds]
- **To initialize a go module =>**
  command => go mod init <module-path-of-current-project>
  e.g. =>. go mod init github.com/sakthishanmugam02/helloworld

SAKTHI SARAVANAN SHANMUGAM – twitter: @sakthis02 - reference/courtesy: @bmuschko

Version Selection Strategy of 'go get'


Start
Version provided?
no
Has Tagged Versions?
no
Has Tagged Pre-Release?
no
Use Provided Version
Use Latest Tag e.g. v0.4.5
Use Latest Pre-Release e.g. v0.0.1-pre1
Use Latest Commit e.g. 447f182
End

Storage location of downloaded dependencies



```
$ echo $GOPATH
/Users/bmuschko/go

$ tree $GOPATH/pkg/mod/github.com
github.com
└── spf13
    ├── cobra@v0.0.5
    │   ├── ...
    └── pflag@v1.0.3
        ├── ...

$ tree $GOPATH/pkg/mod/cache/download/github.com/spf13/cobra/@v
.
├── v0.0.5.mod
├── ...
```

Go Path env. var containing cache
Downloaded Go files of dependencies
Go module files of dependencies

- **To add [or] resolve a dependency =>**
  command => go get <module-path-of-the-dependency>
  e.g. =>. go get github.com/sakthishanmugam02/rpc

- **To list down available releases for a particular dependency =>**
  command => go list -m -versions -json <module-path-of-the-dependency>
  e.g. =>. go list -m -versions -json github.com/sakthishanmugam02/rpc

- **To download a specific version of a dependency/major version/commit =>**
  command => go get <module-path-of-the-dependency>@<version/commit-id> [use '@' symbol]
  e.g. =>. go get github.com/sakthishanmugam02/rpc@v0.0.4
  [note: use the same command while upgrading a dependency]

- **To list down the selected version of dependencies =>**
  command => go list -m all

Go module graph to list down all the transitive dependencies

```
$ go mod graph
github.com/bmuschko/hello-world github.com/spf13/cobra@v0.0.5
github.com/spf13/cobra@v0.0.5 github.com/BurntSushi/toml@v0.3.1
github.com/spf13/cobra@v0.0.5 github.com/cpuguy83/go-md2man@v1.0.10
...
```

aka

```
github.com/bmuschko/hello-world
 └─ github.com/spf13/cobra@v0.0.5
     ├─ github.com/BurntSushi/toml@v0.3.1
     └─ github.com/cpuguy83/go-md2man@v1.0.10
```

To replace with a specific version

"even if go picks latest version" (in go.mod file)
[you can point to local/version control location]

```
module github.com/bmuschko/hello-world

go 1.13

require (
    github.com/spf13/cobra v0.0.3
    github.com/spf13/pflag v1.0.5 // indirect
)

replace github.com/spf13/pflag => github.com/spf13/pflag v1.0.3
```

To exclude a specific version (in go.mod file)

"to tell 'GO' not to pick a specific version of dependency that would not be compatible with your project"

```
module github.com/bmuschko/hello-world

go 1.13

require (
    github.com/spf13/cobra >v0.0.3
    github.com/spf13/pflag v1.0.5 // indirect
)

exclude github.com/spf13/cobra v0.0.4
```

- **To verify all the downloaded dependencies are not corrupted [or] tampered =>**
  command => go mod verify
  explanation. =>. this would help in identifying malicious changes in the downloaded dependencies if any

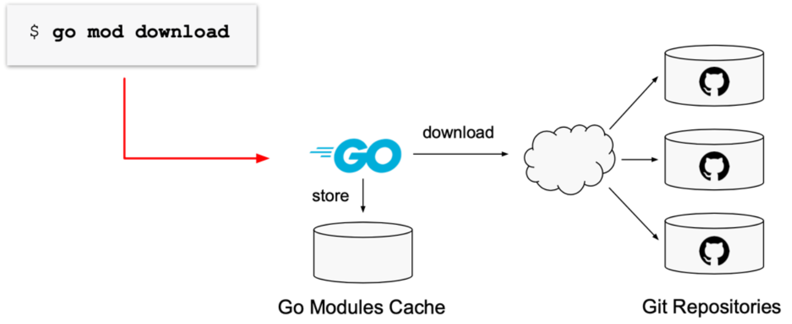
- **To check why a particular module gets downloaded =>**
  command => go mod why <path-of-the-package-in-question>
  explanation. =>. this would point to package that requires the specified dependency

- **To identify and remove dependencies that are no longer required =>**
  command => go mod tidy
  explanation. =>. this would remove dependencies that are no longer referenced in your project

- **To clean the local cache =>**
  command => go clean --modcache
  explanation. =>. this would clean up the downloaded cache folder

To download dependencies!!!

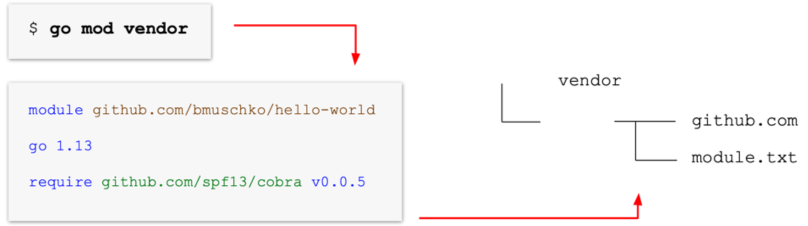
Use: 'go mod download' command



```
$ go mod download
```

To download dependencies as vendor

Use: 'go mod vendor' command
along with go build -mod vendor



```
$ go mod vendor
```

```
module github.com/bmuschko/hello-world
go 1.13
require github.com/spf13/cobra v0.0.5
```

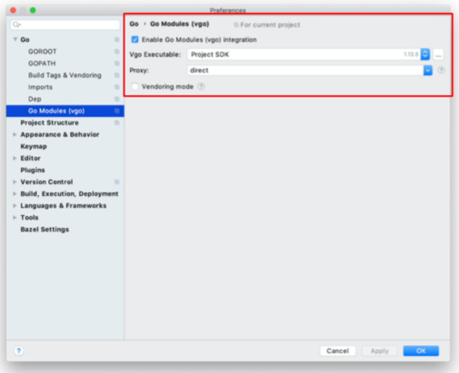
Useful Environmental Variables (related to GO Modules)

- GOMODULE111 => to turn on [or] off the module behavior [for backward compatibility]
  - possible values: on, off, auto
  - note: auto would select module version of go, if it detects go.mod file, else fall back to GOPATH way of vendor directory
- GOPROXY => to configure proxy endpoints [useful, if proxy is integrated]
  - E.g. export GOPROXY=https://gocenter.io

- GOPRIVATE => to configure corporate proxy [useful, if proxy integrated]

To enable GO Modules in IDE

`"go.useLanguageServer": true`



Preferences

Go › Go Modules (vgo) — For current project

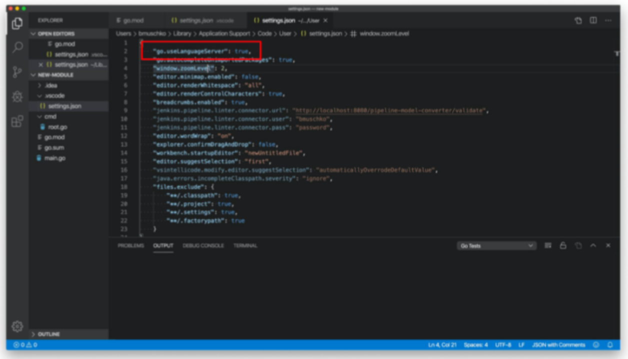
[✓] Enable Go Modules (vgo) integration

Vgo Executable: Project SDK 1.13.5

Proxy: direct

[ ] Vendoring mode

GoLand



Users › bmuschko › Library › Application Support › Code › User › settings.json › window.zoomLevel

```
{
    "go.useLanguageServer": true,
    "go.autocompleteUnimportedPackages": true,
    "window.zoomLevel": 2,
    "editor.minimap.enabled": false,
    "editor.renderWhitespace": "all",
    "editor.renderControlCharacters": true,
    "breadcrumbs.enabled": true,
    "jenkins.pipeline.linter.connector.url": "http://localhost:8080/pipeline-model-converter/validate",
    "jenkins.pipeline.linter.connector.user": "bmuschko",
    "jenkins.pipeline.linter.connector.pass": "password",
    "editor.wordWrap": "on",
    "explorer.confirmDragAndDrop": false,
    "workbench.startupEditor": "newUntitledFile",
    "editor.suggestSelection": "first",
    "vsintellicode.modify.editor.suggestSelection": "automaticallyOverrodeDefaultValue",
    "java.errors.incompleteClasspath.severity": "ignore",
    "files.exclude": {
        "**/.classpath": true,
        "**/.project": true,
        "**/.settings": true,
        "**/.factorypath": true
    }
}
```

VSCode

SAKTHI SARAVANAN SHANMUGAM – twitter: @sakthis02 - reference/courtesy: @bmuschko