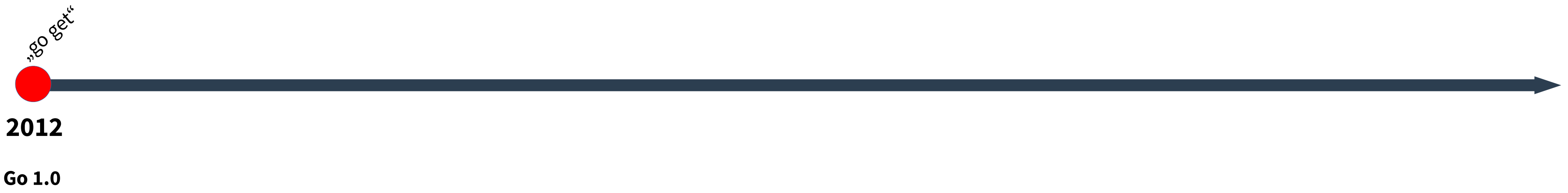
How to manage Go dependencies

From „Go get“ to „Modules“

Rouven Hernier

History: A long time ago in a galaxy far, far...

Timeline



go get

Lets get a new dependency

```
$ go get github.com/volkswagen/letsrock
```

- Download location is the (one) system wide $GOPATH

- Always „origin/master“

- go get also uses flag instruction such as -u (update), -insecure (http) etc.

go get

Lets use the dependency

```
import "github.com/volkswagen/letsrock"
```

go get

Demo

go get

Problems

- No reproducable builds because a specific dependency version can´t be used
- All projects share $GOPATH so all will use the same (already downloaded) dependency version

App1 uses dependency A1.0 → OK!

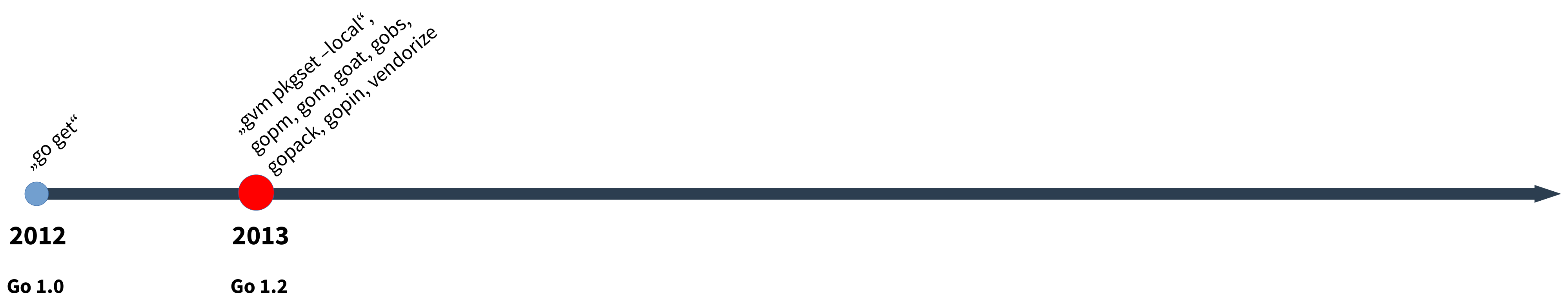
App2 needs dependency A1.5 → broken! (too old!) / Dependency ‚A‘ already downloaded as Version 1.0

„go get -u A“ → But fetches A2.0 now → broken again! (now it is too new!)

→ „go get“ is just a tool for acquire a dependency and not to manage it

History: 2013

Timeline

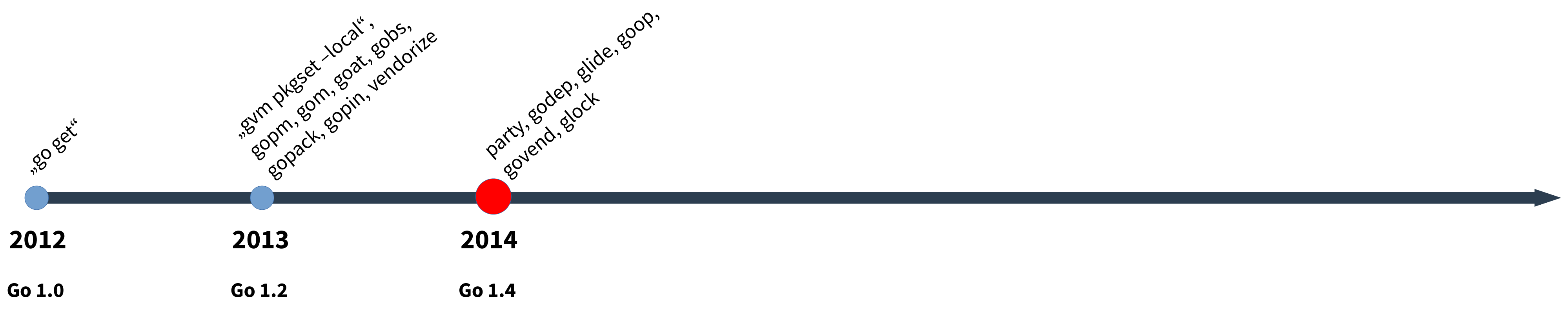


Current total: 8 dependency management tools

(„go get“ not counted)

History: 2014

Timeline

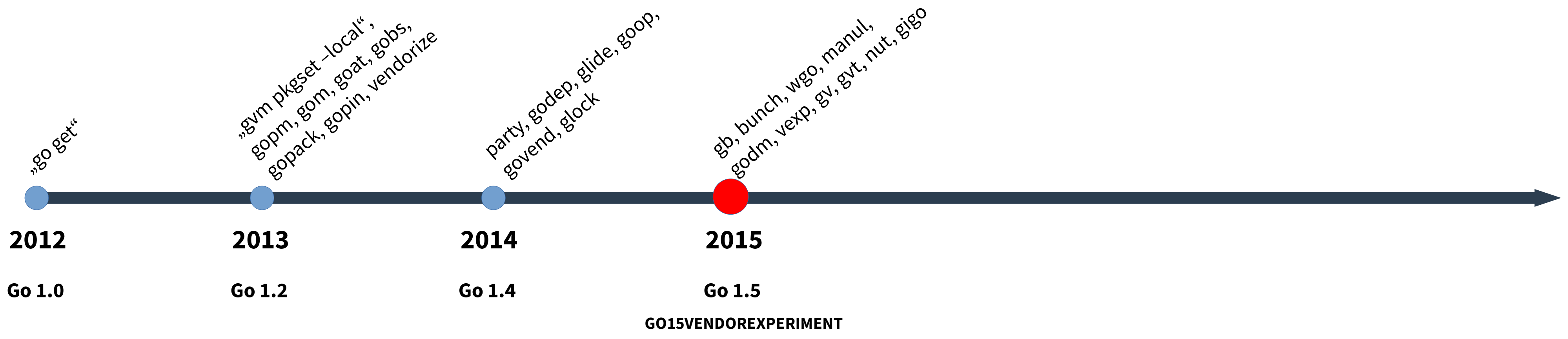

„go get“
„gvm pkgset –local“, gopm, gom, goat, gobs, gopack, gopin, vendorize
party, godep, glide, goop, govend, glock
2012
Go 1.0
2013
Go 1.2
2014
Go 1.4

Current total: 14 dependency management tools

(„go get“ not counted)

History: 2015

Timeline

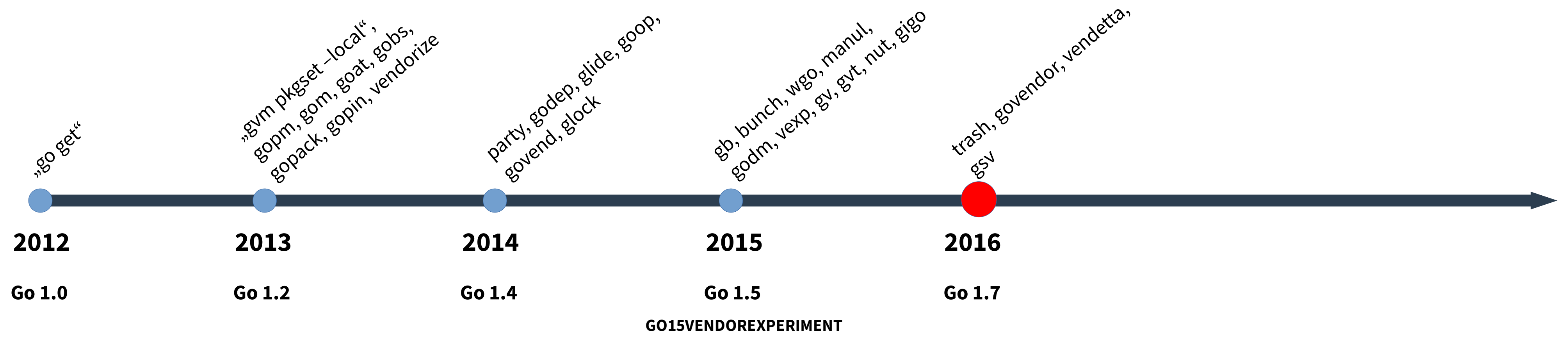

„go get“
„gvm pkgset –local“, gopm, gom, goat, gobs, gopack, gopin, vendorize
party, godep, glide, goop, govend, glock
gb, bunch, wgo, manul, godm, vexp, gv, gvt, nut, gigo
2012
Go 1.0
2013
Go 1.2
2014
Go 1.4
2015
Go 1.5
GO15VENDOREXPERIMENT

Current total: 24 dependency management tools

(„go get“ not counted)

History: 2016

Timeline

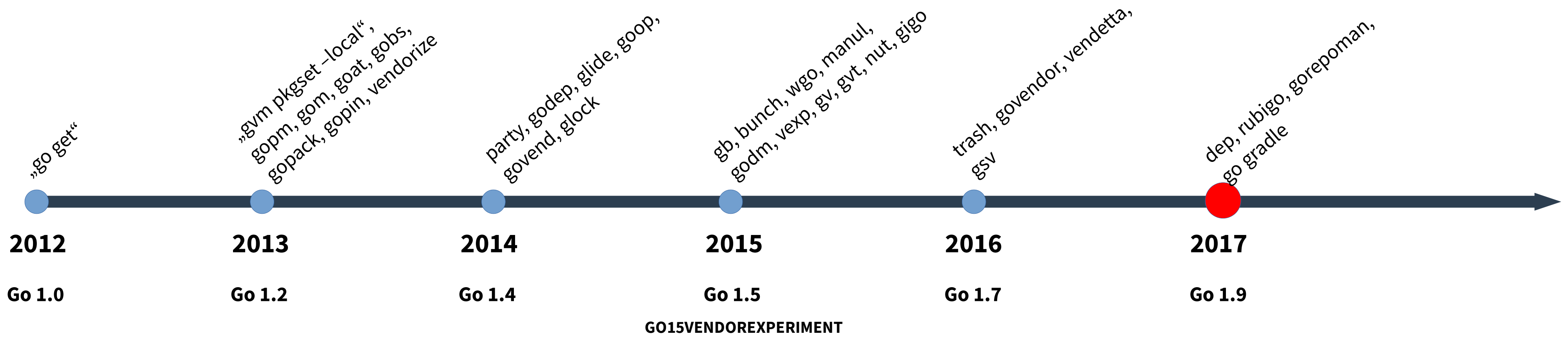

„go get“
„gvm pkgset –local“, gopm, gom, goat, gobs, gopack, gopin, vendorize
party, godep, glide, goop, govend, glock
gb, bunch, wgo, manul, godm, vexp, gv, gvt, nut, gigo
trash, govendor, vendetta, gsv
2012
Go 1.0
2013
Go 1.2
2014
Go 1.4
2015
Go 1.5
GO15VENDOREXPERIMENT
2016
Go 1.7

Current total: 28 dependency management tools

(„go get“ not counted)

History: 2017

Timeline

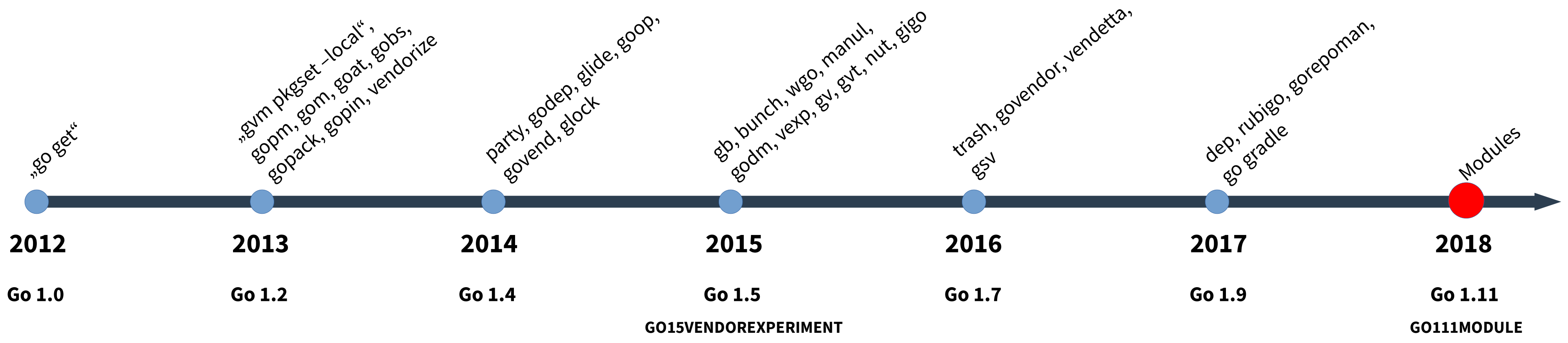

„go get“
„gvm pkgset –local“, gopm, gom, goat, gobs, gopack, gopin, vendorize
party, godep, glide, goop, govend, glock
gb, bunch, wgo, manul, godm, vexp, gv, gvt, nut, gigo
trash, govendor, vendetta, gsv
dep, rubigo, gorepoman, go gradle
2012
2013
2014
2015
2016
2017
Go 1.0
Go 1.2
Go 1.4
Go 1.5
Go 1.7
Go 1.9
GO15VENDOREXPERIMENT

Current total: 32 dependency management tools

(„go get“ not counted)

History: 2018

Timeline

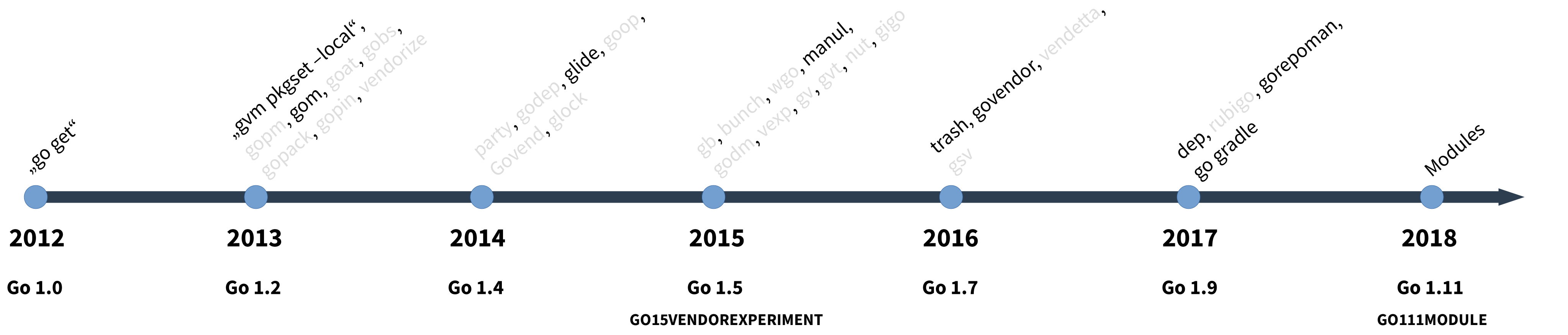

„go get“
„gvm pkgset –local“, gopm, gom, goat, gobs, gopack, gopin, vendorize
party, godep, glide, goop, govend, glock
gb, bunch, wgo, manul, godm, vexp, gv, gvt, nut, gigo
trash, govendor, vendetta, gsv
dep, rubigo, gorepoman, go gradle
Modules
2012
Go 1.0
2013
Go 1.2
2014
Go 1.4
2015
Go 1.5
GO15VENDOREXPERIMENT
2016
Go 1.7
2017
Go 1.9
2018
Go 1.11
GO111MODULE

Current total: 33 dependency management tools

(„go get“ not counted)

History: Active in development

Timeline

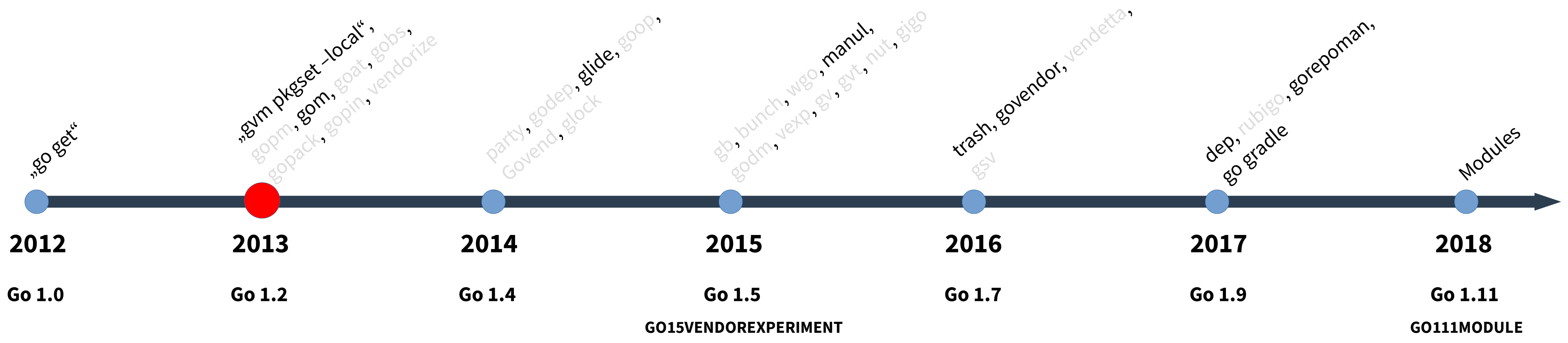

„go get“
„gvm pkgset –local“, gopm, gom, goat, gobs, gopack, gopin, vendorize
party, godep, glide, goop, Govend, glock
gb, bunch, wgo, manul, godm, vexp, gv, gvt, nut, gigo
trash, govendor, vendetta, gsv
dep, rubigo, gorepoman, go gradle
Modules
2012
Go 1.0
2013
Go 1.2
2014
Go 1.4
2015
Go 1.5
GO15VENDOREXPERIMENT
2016
Go 1.7
2017
Go 1.9
2018
Go 1.11
GO111MODULE

Active total: 10 dependency management tools

(„go get“ not counted)

History: Back to 2013 (gvm)

Timeline


„go get“
„gvm pkgset –local“, gopm, gom, goat, gobs, gopack, gopin, vendorize
party, godep, glide, goop, Govend, glock
gb, bunch, wgo, manul, godm, vexp, gv, gvt, nut, gigo
trash, govendor, vendetta, gsv
dep, rubigo, gorepoman, go gradle
Modules
2012
Go 1.0
2013
Go 1.2
2014
Go 1.4
2015
Go 1.5
GO15VENDOREXPERIMENT
2016
Go 1.7
2017
Go 1.9
2018
Go 1.11
GO111MODULE

2013: gvm

Dependency tool: gvm

```
$ gvm pkgset create vwletsrock_v2      # create global package set
$ gvm pkgset use vwletsrock_v2         # choose a global package set
$ gvm pkgset create --local            # create local package set
$ gvm pkgset use --local               # choose local package set
```

- GVM is a tool to switch between multiple Go versions

- In addition GVM provides „package sets“ to change between multiple $GOPATH sets

- Local package set can be shared in in the project repository

2013: gvm

Demo

2013: gvm

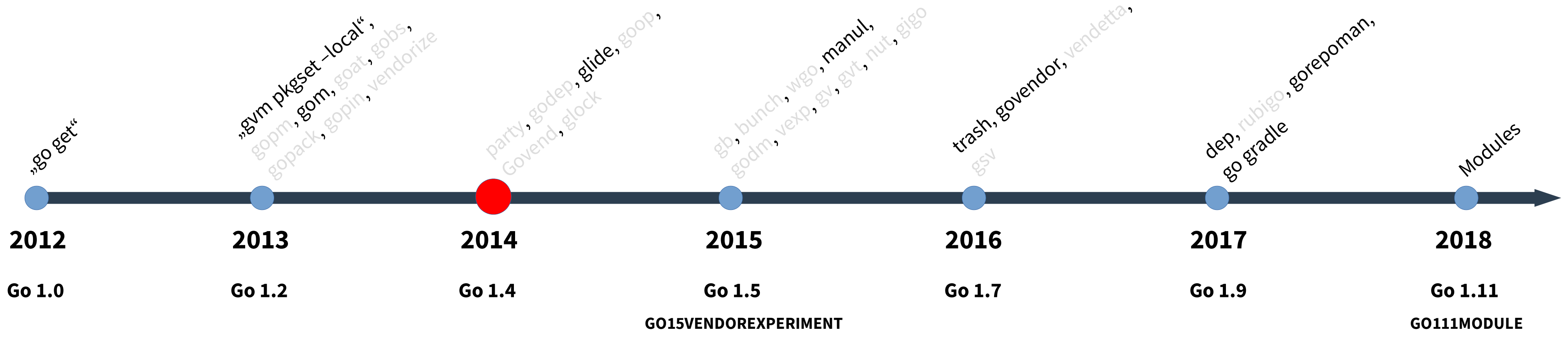
Dependency tool: gvm

→ Type: GOPATH modification without specific dependency version selection

- Package sets don´t provide easy dependency version configuration / pinning
  - Manual selection / download of version to a package set

History: Back to 2014 (glide)

Timeline


„go get"
„gvm pkgset –local", gopm, gom, goat, gobs, gopack, gopin, vendorize
party, godep, glide, goop, Govend, glock
gb, bunch, wgo, manul, godm, vexp, gv, gvt, nut, gigo
trash, govendor, vendetta, gsv
dep, rubigo, gorepoman, go gradle
Modules
2012
Go 1.0
2013
Go 1.2
2014
Go 1.4
2015
Go 1.5
GO15VENDOREXPERIMENT
2016
Go 1.7
2017
Go 1.9
2018
Go 1.11
GO111MODULE

2014: glide

Dependency tool: glide

```
$ mkdir <project>              # create project src dir
$ glide create                 # create a new glide.yaml file
$ vi glide.yaml                # add dependencies (with rev) or ‚glide get <dep>‘
$ glide install                # downloads deps (‚glide update‘ for updating)
$ glide in                     # configure GOPATH
```

- „glide in“ was for $GOPATH trickery/switching so that the „go tools“ still work

- „glide install“ will also fetch all addional required dependencies

2014: glide

Dependency tool: glide

```
$ less glide.yaml
package: github.com/volkswagen/app
import:
- package: github.com/example/dependency
  version: ^1.2.0
```

- Version configuration like >= 1.2.0 && < 2.0.0 is possible
- Dependencies were downloaded in a local / per project „_vendor“ folder

2014: glide

Demo

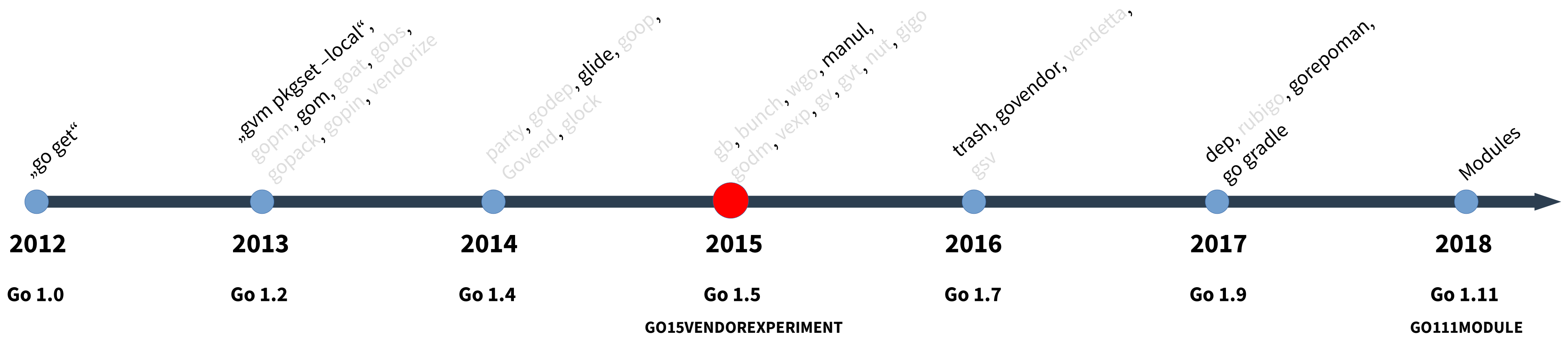
2014: glide

Dependency tool: glide

- → 2014 type: GOPATH modification with specific dependency version selection
- → Current type: Vendor packages with specific dependency version selection

History: Back to 2015 (manul)

Timeline



2015: manul

Dependency tool: manul

```
$ manul -Q                   # show all dependencies
$ manul -I <dep>=34a235h1    # d/l dep (version) into proj. vendor folder (git submodules)
$ manul -U <dep>=TAG         # update dep to specific tag (optional)
```

- No extra dependencies config file. Just GIT submodules

2015: manul

Demo

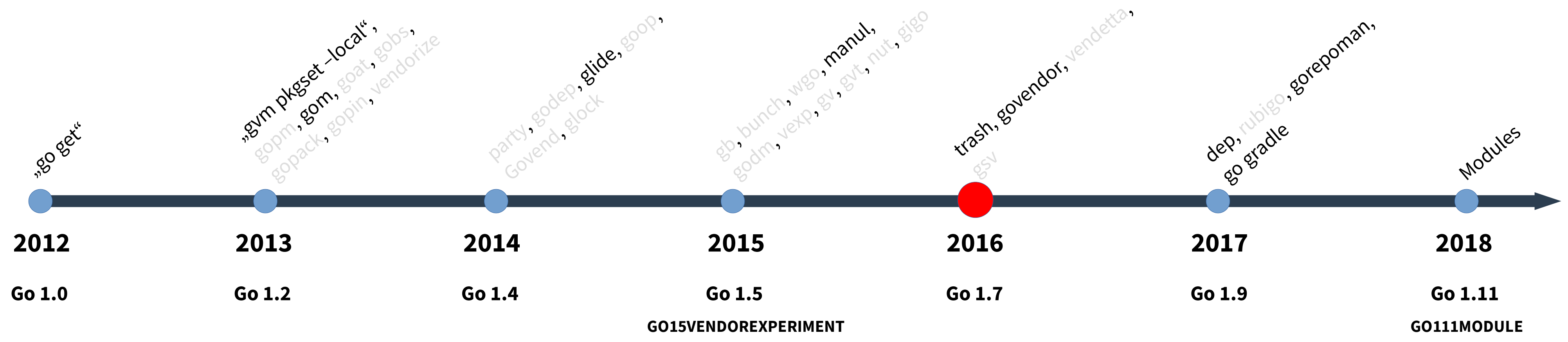
2015: manul

Dependency tool: manul

→ Type: Vendor packages with specific dependency version selection

History: Back to 2016 (govendor)

Timeline



„go get“
„gvm pkgset –local“, gopm, gom, goat, gobs, gopack, gopin, vendorize
party, godep, glide, goop, Govend, glock
gb, bunch, wgo, manul, godm, vexp, gv, gvt, nut, gigo
trash, govendor, vendetta, gsv
dep, rubigo, gorepoman, go gradle
Modules
2012
Go 1.0
2013
Go 1.2
2014
Go 1.4
2015
Go 1.5
GO15VENDOREXPERIMENT
2016
Go 1.7
2017
Go 1.9
2018
Go 1.11
GO111MODULE

2016: govendor

Dependency tool: govendor

```
$ govendor init               # create vendor folder with vendor.json
$ govendor fetch <dep>@v2     # d/l dep (version) into proj. vendor folder
$ govendor sync               # d/l all configured dependencies
```

2016: govendor

Demo

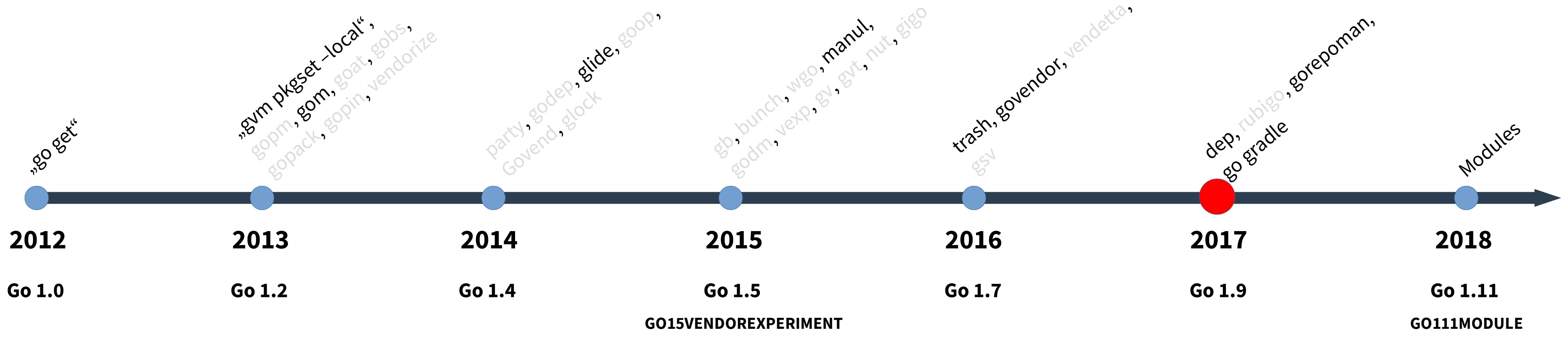
2016: govendor

Dependency tool: govendor

→ Type: Vendor packages with specific dependency version selection

History: Back to 2017 (dep)

Timeline



„go get“
„gvm pkgset –local“, gopm, gom, goat, gobs, gopack, gopin, vendorize
party, godep, glide, goop, Govend, glock
gb, bunch, wgo, manul, godm, vexp, gv, gvt, nut, gigo
trash, govendor, vendetta, gsv
dep, rubigo, gorepoman, go gradle
Modules
2012
Go 1.0
2013
Go 1.2
2014
Go 1.4
2015
Go 1.5
GO15VENDOREXPERIMENT
2016
Go 1.7
2017
Go 1.9
2018
Go 1.11
GO111MODULE

2017: dep

Dependency tool: dep

```
$ dep init                     # create vendor folder, Gopg.lock & Gopkg.toml

$ dep ensure -add <dep>        # add dependency to project

$ dep ensure                   # check deps status/vendor and d/l if needed
```

2017: dep

Dependency tool: dep

```
$ less Gopkg.toml
[[constraint]]
  name = "github.com/volkswagen/letsrock"
  version = "=1.3.2"
[prune]
  go-tests = true                                     # dont add dep tests to vendor
  unused-packages = true                              # dont add not used packs to vendor
```

– Be aware of the version format (e.g. version = „=1.2.0“)

2017: dep

Demo

2017: dep

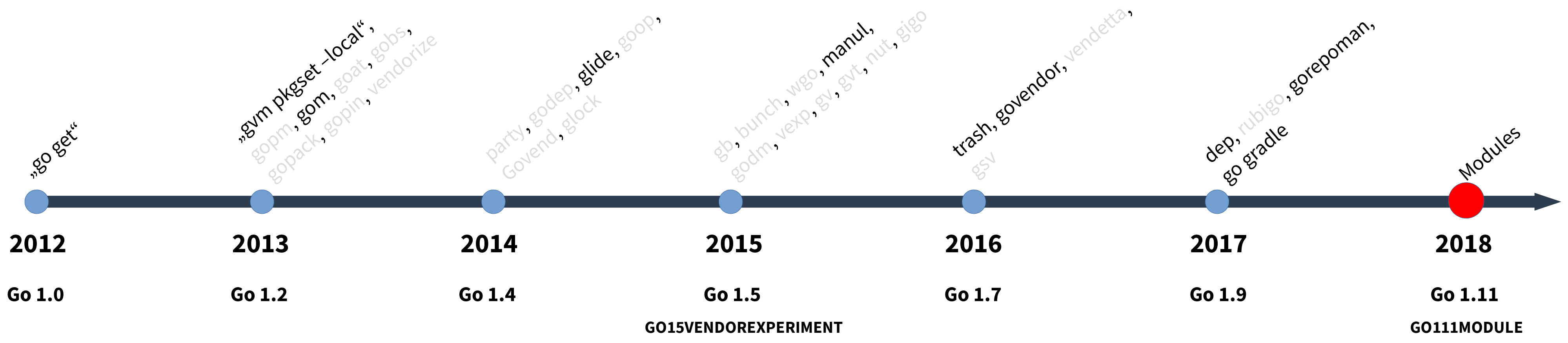
Dependency tool: dep

→ Type: Vendor packages with specific dependency version selection

- Was till Go version 1.11 the „official experiment“ tool for dependency management

History: 2018 (Modules)

Timeline



2018: Modules

Dependency tool: Modules

```
$ mkdir <project>             # create project folder OUTSIDE $GOPATH
$ go mod init <module-name>   # creates go.mod for module e.g. ‚github.com/vw/letsrock‘
$ go run/build <app>           # creates go.sum and downloads all deps in the latest version
$ go get <dep>@v1.2.4         # get specific dep version, rev or tag
$ go get -u                   # update to the latest minor or patch release
$ go get -u=patch             # update to latest patch release
$ go mod tidy                 # remove unused modules & add missing modules
```

- Dependencies will be downloaded to the $GOPATH/pkg/mod folder

2018: Modules

Dependency tool: Modules

```
$ less go.mod

module github.com/vw/letsrock

require (

    github.com/some/dependency v1.2.3

    github.com/another/dependency/v4 v4.0.0

)
```

- If a new module version has the same import path it must be backward compatible

  - Breaking new version needs new import path. E.g: github.com/vw/letsrock/v2

2018: Modules

Demo

2018: Modules

Dependency tool: Modules

→ Type: GO111MODULE

- Plan to finalize Go Modules with Go 1.12

- IDE support for GoLand, beta for VS Code exists

- Vendoring can be used together with Modules

    - „go mod vendor“ → create vendor directory again
    - „go build“ ignore vendor directory when in module mode
        - „go build -mod=vendor“

- ‚vgo‘ as standalone implementation for a Go 1.10 toolchain

Thats it!? :)

One more thing…
(Teaser)

2018: Github.com as dependency repo

Problems

- Developer pulls off github repository
    - See / search for Javascript left-pad problem :)

- Github.com is down (forever)

→ All the shown dependency managing tools will not solve these problems.

2018: athens

In 2018 was the project athens started

- Go module data store
    - Keep the source code @Github
    - But store a never changeable and always online dependency @ a trusted location
        - E.g: Google, Microsoft or Amazon
        - Corperate On-Prem

- Dependency proxy (with caching)

→ More infos in one of the next Wolfsburg Gophers Meetups! :)