

REFACTORING TO MODULES:

All you need to know
in less than an hour

LET'S GO BACK IN
TIME



POLL TIME!

 Pre 1.0 (2012)

 1.2 (2013)

 1.5 (2015)

 1.8 (2017)

 1.11 (2018)

 1.13 (2019)

BARUCH SADOGURSKY

CHIEF STICKER OFFICER

(ALSO 🎩 OF DEVELOPER ADVOCACY)



JBARUCH@JFROG.COM

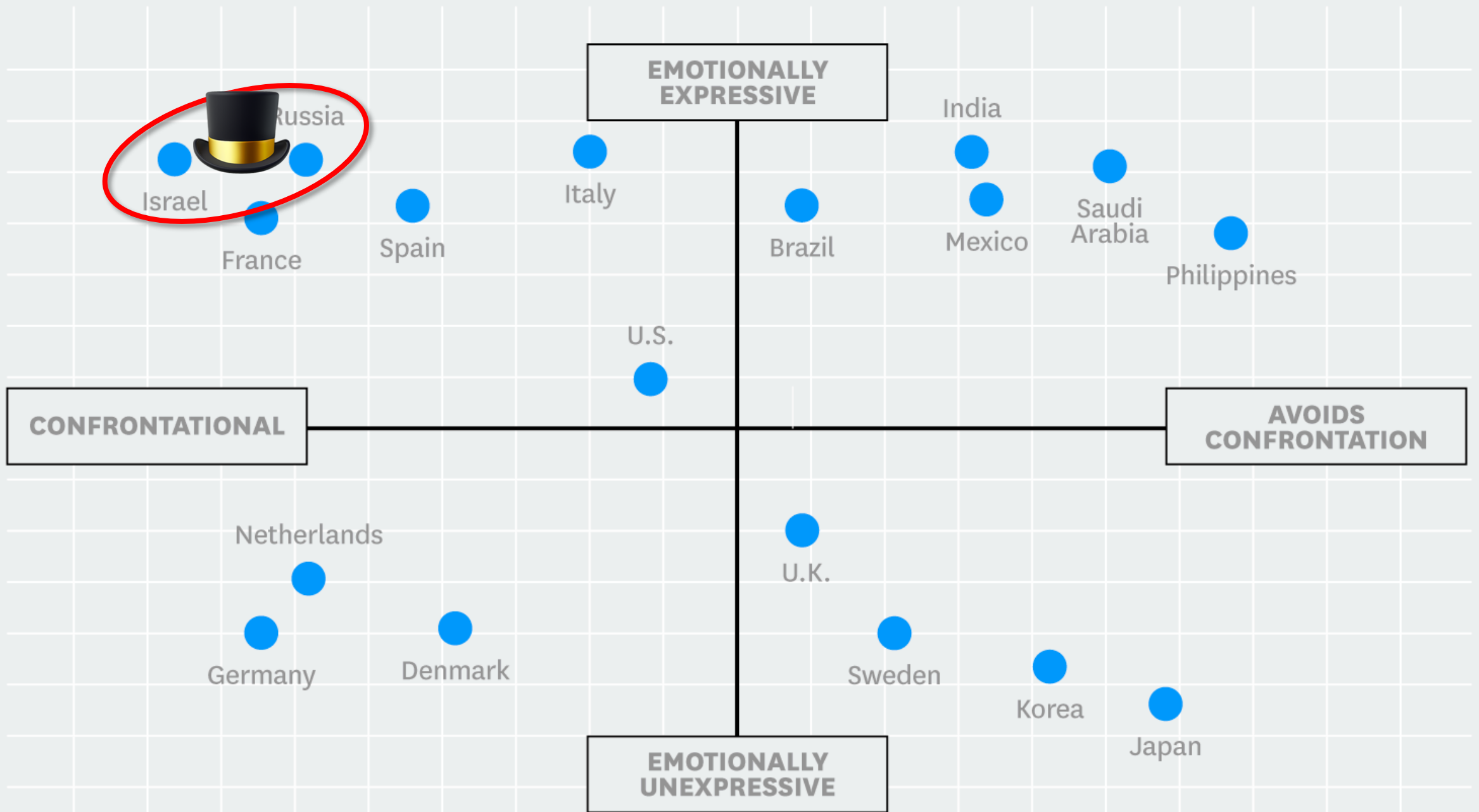


@JBARUCH



+1(408)890-9281





SHOWNOTES



<http://jfrog.com/shownotes>



Slides



Video



Links



Comments, Ratings



Raffle

WHY WE HAVE A PROBLEM?

History

Design began in late 2007.

Key players:

- Robert Griesemer, Rob Pike, Ken Thompson
- Later: Ian Lance Taylor, Russ Cox

WHY WE HAVE A PROBLEM?

A personal history of dependencies at Google

Plan 9 demo: a story

Early Google: one Makefile

2003: Makefile generated from per-directory BUILD files

- explicit dependencies
- 40% smaller binaries

Dependencies still not checkable!

SIMPLE SOLUTION!


- 🐸 Dependencies are sources
- 🐸 Remote import is a VCS path
- 🐸 Dump everything together into one source tree (GOPATH)
- 🐸 Compile
- 🐸 Profit




@jbaruch #golang #GoSG #GoCenter <http://jfrog.com/shownotes>

BUT... HOW DO I...

 Know which dependencies do I use?

 Know which dependencies did you use?

 Know which dependencies should I use?

 Know is it our code that I am editing right now?

 WTF is going on?!

YEAH...

“ To date, we’ve resorted to an email semaphore whenever someone fixes a bug a package, imploring everyone else to run `go get -u`. You can probably imagine how successful this is, and how much time is being spent chasing bugs that were already fixed.

Dave Cheney

DUPLICATE YOUR DEPENDENCIES

“ Check your dependencies to your own VCS.

Brad Fitzpatrick

BUILD YOUR OWN DEPENDENCY MANAGER

“ It's not the role of the tooling provided by the language to dictate how you manage your code in the production sense.

Andrew Gerrand

WE EXPECT YOU TO ALREADY HAVE A HOMEGROWN DEPENDENCY MANAGER

“ If you need to build any tooling around what Go uses (Git, Mercurial, Bazaar), you already understand those tools, so it should be straightforward to integrate with whatever system you have.

Andrew Gerrand

DON'T TRUST WHAT WE'VE BUILT

“ go-get is nice for playing around, but if you do something serious, like deploying to production, your deploy script now involves fetching some random dude's stuff on GitHub.

Brad Fitzpatrick

NEXT THING YOU KNOW...

THERE ARE 19 DEPENDENCY MANAGERS



QUIZ TIME!



godeps.json



dependencies.tsv



govendor, govend,
goven, gv



trash, garbage, rubbish



Weapons manufacturer

GOPATH + VENDORING = 

Google Stores Billions of Lines of Code in a Single Repository ...

<https://www.youtube.com/watch?v=W71BTkUbdqE>



Bazel How to build at Google scale? - YouTube

<https://www.youtube.com/watch?v=OUwu0myF8M>



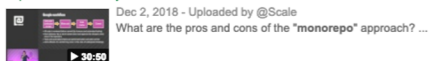
Billions of lines of code in a single repository, SRSLY? by Guillaume ...

<https://www.youtube.com/watch?v=yM0GQw1zgrA>



Very interesting talk about how Google handles a monorepo at ...

<https://dev.to/matteojoliveau/comment/7a34>



Git at Google: Making Big Projects (and Everyone Else) Happy, Dave ...

<https://www.youtube.com/watch?v=cY34mr71ky8>



The Curious Case of Monorepos, Johannes Stein - React London ...

<https://www.youtube.com/watch?v=DFCtAupKbEk>



Uber Technology Day: Monorepo to Multirepo and Back Again ...

<https://www.youtube.com/watch?v=IV8-1S28ycM>



The Challenge of Monorepos: Strategies from git-core and Open ...

<https://www.youtube.com/watch?v=FSYBask5ao>



Monorepos in the Wild - Markus Oberlehner @ WeAreDevelopers ...

<https://www.youtube.com/watch?v=kwhOl4mmqNM>



BazelCon 2018 Day 1: Virtual Mono-Repo & Bazel - YouTube

<https://www.youtube.com/watch?v=2gNITegwQD4>



GOPATH, THE PROUD SON OF THE MONOREPO

TWO HUGE PROBLEMS WITH GOPATH



It only allows a single version of any given package to exist at once (per GOPATH)








We cannot programmatically differentiate between code the user is working on and code they merely depend on

VENDORING – THE WORST KIND OF FORKING

- “ Copy all of the files at some version from one version control repository and paste them into a different version control repository

WHAT'S WRONG WITH IT (WELL, WHAT'S NOT)

-  History, branch, and tag information is lost
-  Pulling updates is impossible
-  It invites modification, divergence, and bad fork
-  It wastes space
-  Good luck finding which version of the code you forked

BBC

WAIT A MINUTE...

WE USE SUBMODULES!

STILL WRONG!



You still have no idea
what version are you
using



You have to connect each
dependency as a
submodule manually



Switching branches and
forks LOL



Working on modules
with other teams ROFL



THE GO DEP EXPERIMENT



So you want to write a package manager

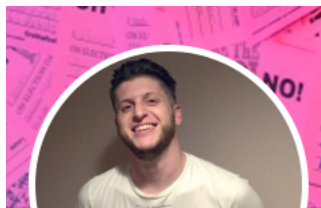
You woke up this morning, rolled out of bed, and thought, “Y’know what? I don’t have enough misery and suffering in my life. I know what to do—I’ll write a language package manager!”

...

Package management is awful, you should quit right now

Package management is a nasty domain. Really nasty. On the surface, it *seems* like a purely technical problem, amenable to purely technical solutions. And so, quite reasonably, people approach it that way. Over time, these folks move inexorably towards the conclusion that:

1. software is terrible
2. people are terrible
3. there are too many different scenarios
4. nothing will really work for sure
5. it’s provable that nothing will really work for sure
6. our lives are meaningless perturbations in a swirling vortex of chaos and entropy



sam boyer

@sdboyer

systems | people

📍 Detroit metro

📅 Joined December 2008

PROPER DEPENDENCY MANAGEMENT?



Working in project
directories



Local cache for
dependencies



Version declarations



Conflict resolution

**CONFLICT ON THE
CONFLICT
RESOLUTION**



SAT/SMT
vs
MVS/SIV

**ENTER GO
MODULES**



ENTER GO MODULES

BACKWARDS COMPATIBILITY AND MIGRATION



`go mod init`



`go.mod` file is created



The rest is the same:
imports in code just work

**THAT'S SOME
SERIOUS MAGIC...**

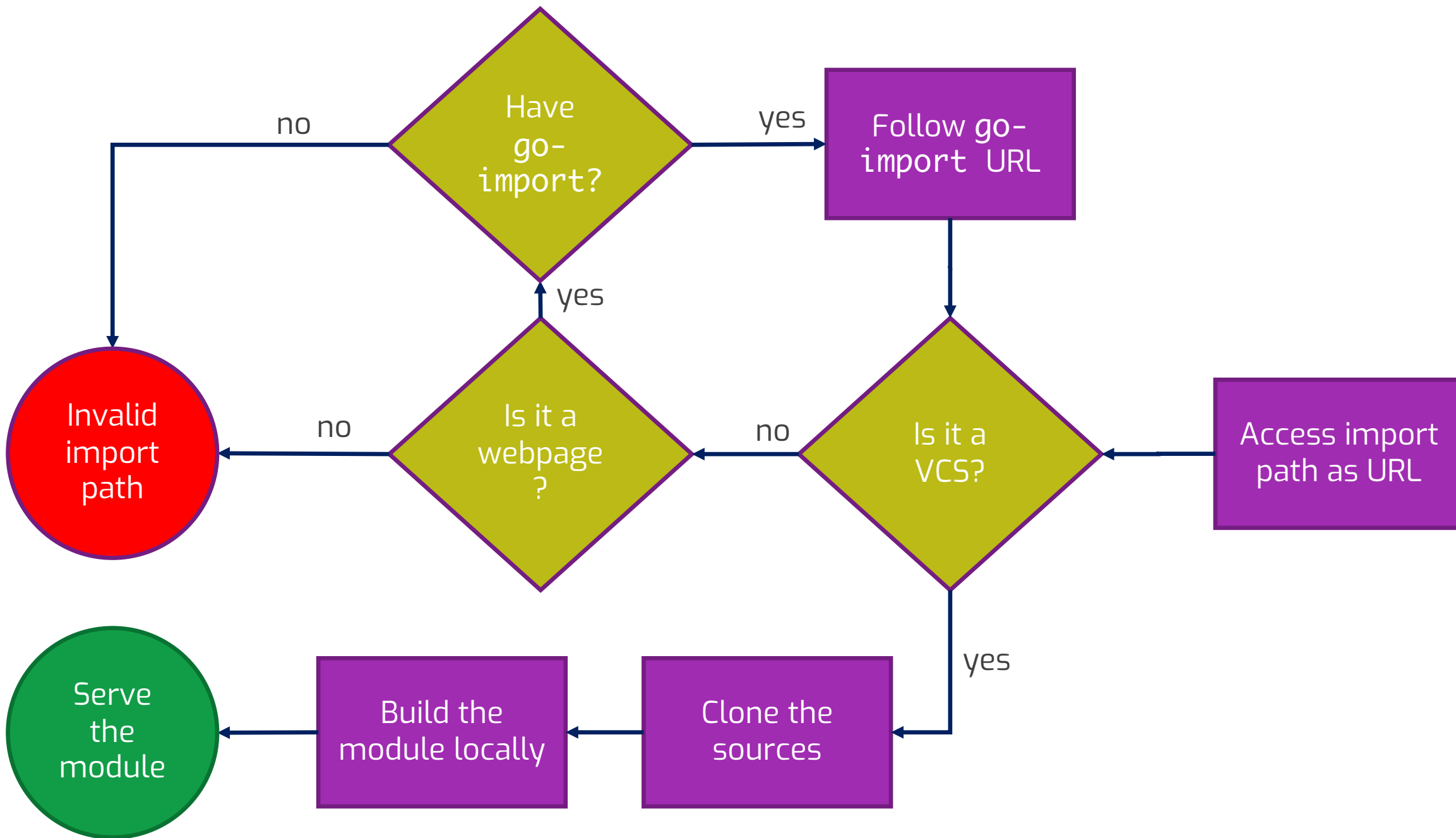


GO MODULES CONVERT EVERYTHING (ALMOST?)

```
var Converters = map[string]func(string, []byte) (*modfile.File, error){
    "GLOCKFILE":          ParseGLOCKFILE,
    "Godeps/Godeps.json": ParseGodepsJSON,
    "Gopkg.lock":          ParseGopkgLock,
    "dependencies.tsv":    ParseDependenciesTSV,
    "glide.lock":          ParseGlideLock,
    "vendor.conf":         ParseVendorConf,
    "vendor.yml":          ParseVendorYML,
    "vendor/manifest":     ParseVendorManifest,
    "vendor/vendor.json":  ParseVendorJSON,
}
```



**WHAT HAPPENS TO GO.MOD WHEN
YOU ADD IMPORT
(AND RUN GO GET/GO BUILD)**



SELECTING A VERSION?

Latest
compatible
version tag

EASY.



@jbaruch #golang #GoSG #GoCenter <http://jfrog.com/shownotes>

COMPATIBLE?!



Let's assume SemVer works (LOL)



The latest version of v1.x.x is compatible with v1.0.0 and up



Premise: import path string should always be backwards compatible

WHAT ABOUT VERSION 2?!



Incompatible code can't
use the same import path



Add **/v2/** to the module
path



Use **/v2/** in the import
path

```
import "github.com/my/module/v2/mypkg"
```


**WHAT IF IT DOESN'T
HAVE ANY SEMVER
TAGS?!**



Pseudo version

v0.0.0-yyyymmddhhmmss-abcdefabcdef

WHAT IF (WHEN) I WANT TO BAN A VERSION?!



You can specify “version
X or later”: `>= x.y.z`



You can use `exclude` or
`replace` for better control

BUT WHICH VERSION
IS REALLY USED?

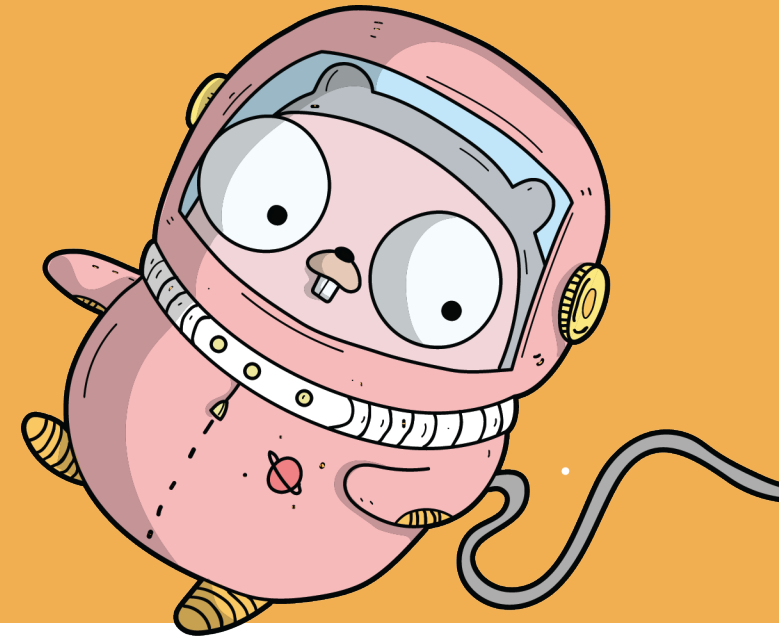


The version in use is
recorded in `go.sum` file



Used to verify the
checksums of
downloaded modules

HOUSTON... I THINK I LOST MY MODULE?



FROM VENDORING
TO HIERARCHY OF
MODULE
REPOSITORIES



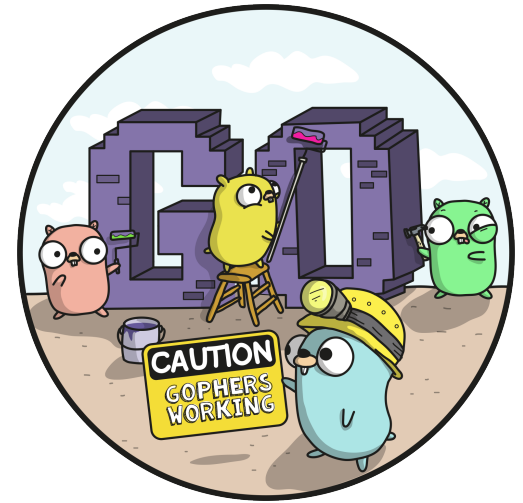
MODULES SHOULD BE IMMUTABLE

The `<module>@v<version>` construct should be immutable

That means that

`github.com/myuser/myrepo/@v/v1.0.0`

Should forever be the same...



BUT ARE THEY REALLY?

“Friends don't let friends do git push -f”

Aaron Schlesinger



CHECKSUMS FOR THE RESCUE!

CHECKSUMS FOR THE RESCUE!

KEEPING MODULES



Local cache (\$GOPATH/pkg/mod)

Immediate access, not shared, can be wiped...



Organizational cache (private proxy)

Fast access, requires infra, shared across devs



Public cache (public proxy)

Highly available, CDN, no infra, free



JFrog
ARTIFACTORY

JFrog
GoCenter

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

Wait a second, my
private modules don't
have an entry in
sum.golang.org!



HOW DO I EXCLUDE MODULES FROM CHECKSUM VERIFICATION?



Use GONOSUMDB to list
packages to exclude from
checksum checks

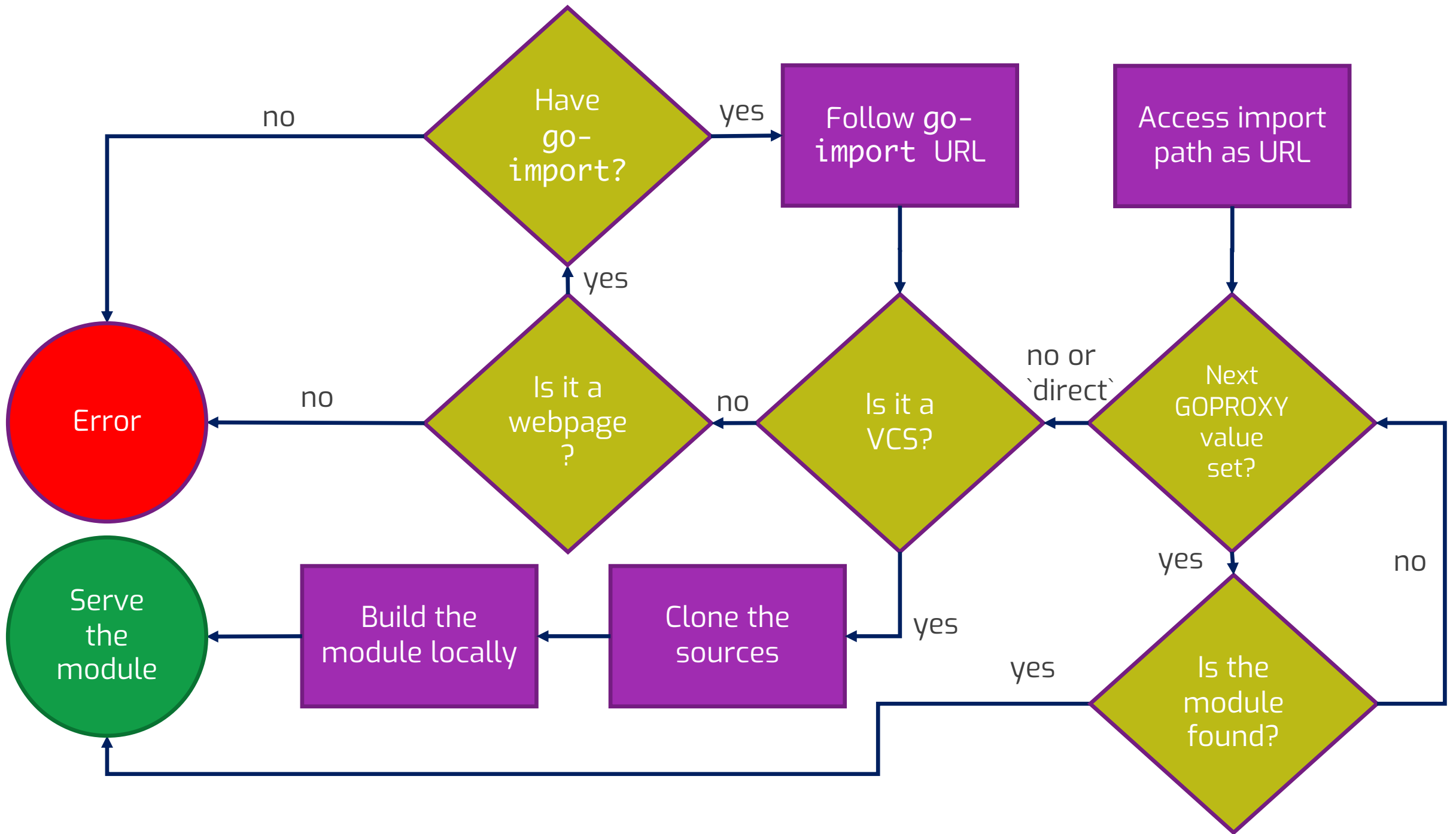
GONOSUMDB=*.corp.example.com

USING THE GOPROXY VARIABLE

```
export GOPROXY=https://myawesomeproxy.com,https://proxy.golang.org,direct
```

OR

```
export GOPROXY= https://myartifactory.jfrog.io/go,direct
```



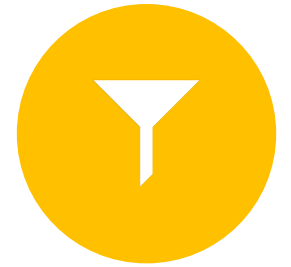
ARTIFACTORY GO SUPPORT FTW



REMOTE
PROXY



LOCAL
REPOSITORY
FOR PRIVATE
MODULES



VIRTUAL
REPOSITORIES
FOR FASTER
RESOLUTION

IMMUTABLE AND REPEATABLE BUILDS



Immutable dependencies

The best way to guarantee issues is force push



Lost Dependencies

Who doesn't remember left-pad with Node.js?



Internet Issues





Even build when GitHub is down!?



AND ALSO FASTER BUILDS...



PROXY.GOLANG.ORG OR GOCENTER?







Feature	GoCenter	proxy.golang.org
Serves modules		
Supports checksum DB		
Search		
Add module to index		
Stats		
Vulnerabilities		

**BUT
CRYPTOGRAPHIC
HASH SHOULD
PROVIDE SECURITY!**



Hash protects only
against supply chain
attacks

PROXY.GOLANG.ORG OR GOCENTER?

Feature	GoCenter	proxy.golang.org
Serves modules		
Supports checksum DB		
Search		
Add module to index		
Stats		
Vulnerabilities		
Metrics		
Default in Go		



Go Module Mirror, Index, and Checksum Database

The Go team is providing the following services run by Google: a module mirror for accelerating Go module downloads, an index for discovering new modules, and a global go.sum database for authenticating module content.

As of Go 1.13, the go command by default downloads and authenticates modules using the Go module mirror and Go checksum database. See proxy.golang.org/privacy for privacy information about these services and the [go command documentation](#) for configuration details including how to disable the use of these servers or use different ones. If you depend on non-public modules, see the [documentation for configuring your environment](#).

Services

proxy.golang.org - a module mirror which meets the spec provided at `go help goproxy`.



Go Module Mirror, Index, and Checksum Database

The Go team is providing the following services run by Google: a module mirror for accelerating Go module downloads, an index for discovering new modules, and a global go.sum database for authenticating module content.

As of Go 1.13, the `go` command by default downloads and authenticates modules using the Go module mirror and Go checksum database. See proxy.golang.org/privacy for privacy information about these services and the [go command documentation](#) for configuration details including how to disable the use of these servers or use different ones. If you depend on non-public modules, see the [documentation for configuring your environment](#).

Services

proxy.golang.org - a module mirror which meets the sp

JFrogGoCenterThe Central Go Modules RepositoryBlogReadMeJFrog

To use GoCenter:

```
export GOPROXY=https://gocenter.io
```

SET ME UP

github.com/aerogo/http

☆
9
Stars

NOASSERTION
License

①
1,186
Downloads

April 19th 2020
Last Modified

Version: v1.1.3

ReadMeMod FileSecurityDependencies (2)Used By (2)MetricsVersions

http

godocreferencego reportA+build successcodecov75%github donate

Provides HTTP utilities. Currently it offers a fast and easy-to-use HTTP client.

Installation

```
go get github.com/aerogo/http/client
```

Request

Basic GET request

```
response, err := client.Get("https://example.com").End()
```

Other HTTP methods

Powered by JFrog © Copyright 2020 JFrog Ltd.Contact UsPrivacy PolicyTerms Of ServiceStatus

@jbaruch #golang #GoSG #GoCenter <http://jfrog.com/shownotes>

Go Module Mirror, Index, and Checksum Database

The Go team is providing the following services run by Google: a module mirror for accelerating Go module downloads, an index for discovering new modules, and a global go.sum database for authenticating module content.

As of Go 1.13, the go command by default downloads and authenticates modules using the Go module mirror and Go checksum database. See proxy.golang.org/privacy for privacy information about these services and the [go command documentation](#) for configuration details including how to disable the use of these servers or use different ones. If you depend on non-public modules, see the [documentation for configuring your environment](#).

Services

proxy.golang.org - a module mirror which meets the sp

JFrogGoCenterThe Central Go Modules Repository

To use GoCenter:export GOPROXY=https://gogetter.io

BlogReadMe

github.com/moby/moby

56,887Stars

Apache-2.0License

4,315Downloads

April 22nd 2020Last Modified

Version: v17.12...

ReadMeMod FileSecurityDependencies (109)Used By (0)MetricsVersions

Total number of vulnerabilities: 1

POWERED BY JFrog Xray

CVE	Severity	Description
CVE-2018-20699	Low	Docker Engine before 18.09 allows attackers to cause a denial of service (dockerd memory consumption) via a large integer in a --cpuset-mems or --cpuset-cpus value, related to daemon/daemon_unix.go, pkg/parsers/parsers.go, and pkg/sysinfo/sysinfo.go.

This table captures only vulnerabilities for this specific module and version and not all of its dependencies. By navigating to the dependencies tab - you can walk through the module's dependencies and find out vulnerability info about those specific versions.

The vulnerabilities found here are a result of running a limited version of JFrog Xray, for more information [click here](#)

JFrogGoCenterThe Central Go Modules Repository

To use GoCenter:export

github.com/aerogo/http

9Stars

NOASSERTIOLicense

ReadMeMod FileSecurityDependencies (2)Used By (2)MetricsVersions

http

godocreferencego reportA+build successcodecov75%github donate

Provides HTTP utilities. Currently it offers a fast and easy-to-use HTTP client.

Installation

go get github.com/aerogo/http/client

Request

Basic GET request

response, err := client.Get("https://example.com").End()

Other HTTP methods

Powered by JFrog © Copyright 2020 JFrog Ltd.

Contact UsPrivacy PolicyTerms Of ServiceStatus



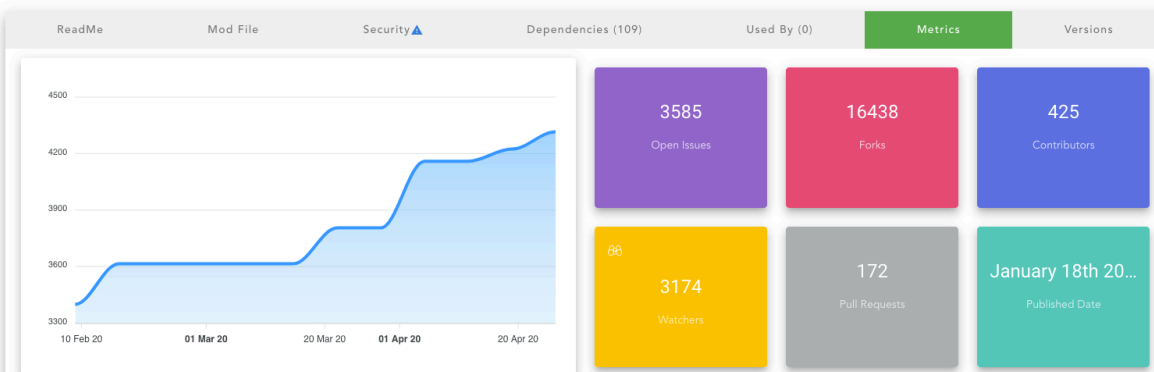
Go Module Mirror, Index, and Checksum



To use GoCenter: `export GOPROXY=https://gocenter.io` SET ME UP

github.com/moby/moby

github.com/moby/moby 56,887 Stars Apache-2.0 License 4,315 Downloads April 22nd 2020 Last Modified Version: v17.12...



godoc reference go report A+ build success codecov 75% github donate

Provides HTTP utilities. Currently it offers a fast and easy-to-use HTTP client.

Installation

```
go get github.com/aerogo/http/client
```

Request

Basic GET request

```
response, err := client.Get("https://example.com").End()
```

Other HTTP methods

56,887 Stars Apache-2.0 License 4,315 Downloads April 22nd 2020 Last Modified Version: v17.12...

Mod File Security Dependencies (109) Used By (0) Metrics Versions

vulnerabilities: 1

POWERED BY JFrog Xray

ity Description

Docker Engine before 18.09 allows attackers to cause a denial of service (dockerd memory consumption) via a large integer in a --cpuset-mems or --cpuset-cpus value, related to daemon/daemon_unix.go, pkg/parsers/parsers.go, and pkg/sysinfo/sysinfo.go.

ilities for this specific module and version and not all of its dependencies. By navigating to the dependencies tab - you can walk through the module's dependencies and find out ecific versions.

e a result of running a limited version of JFrog Xray, for more information [click here](#)

Feedback

TWITTER ADS AND Q&A



jfrog.com/shownotes



[@jbaruch](https://twitter.com/jbaruch)



[#GoSG](https://twitter.com/GoSG)



gocenter.io