GoLab Florence – November 2023

State of Go

Cameron Balahan

Google

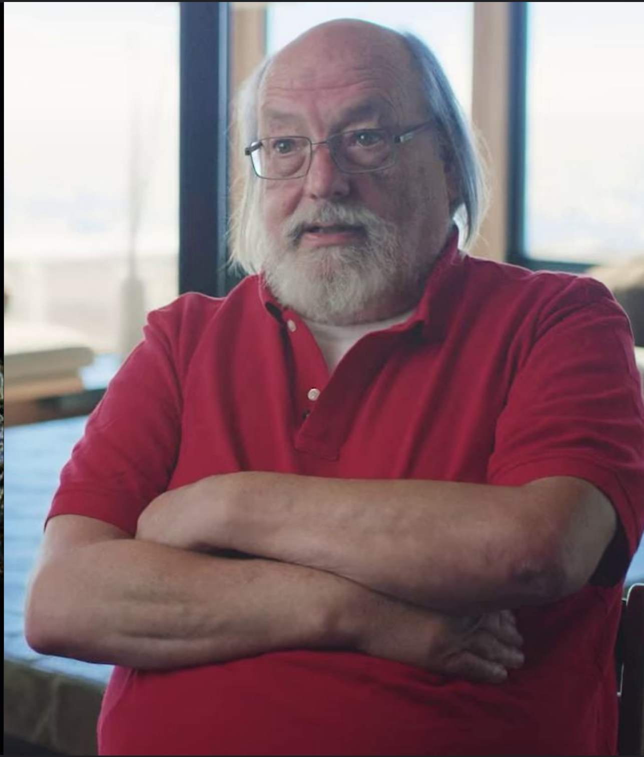
@cameronbalahan

SECTION ONE

Where we have been

Robert Griesemer

Rob Pike



Ken Thompson

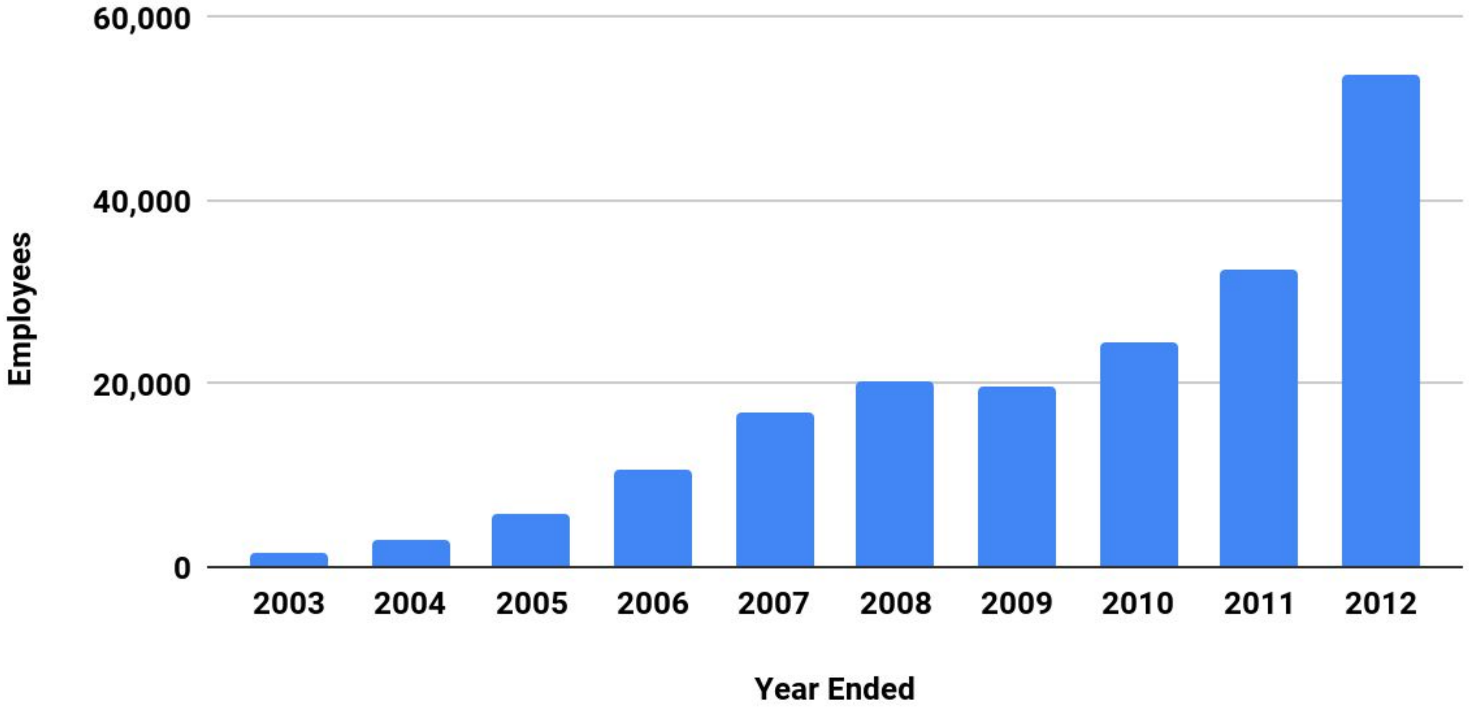
Did the C++ committee really believe what was wrong with C++ was that it didn't have enough features? **Surely... it would be a greater achievement to simplify the language rather than to add to it.**

“

Rob Pike
2007

Alphabet Headcount

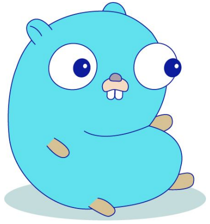
2003-2012



Source: Google

Five years from start to stability

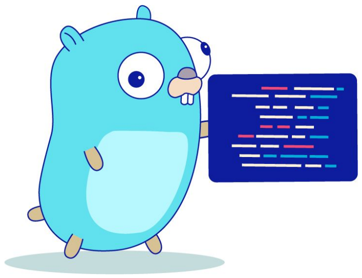
2007



Start

Go is started at Google as
a 20% project

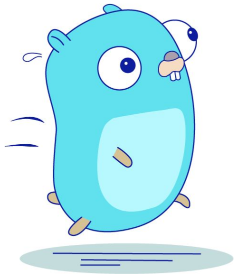
2009



Open Source

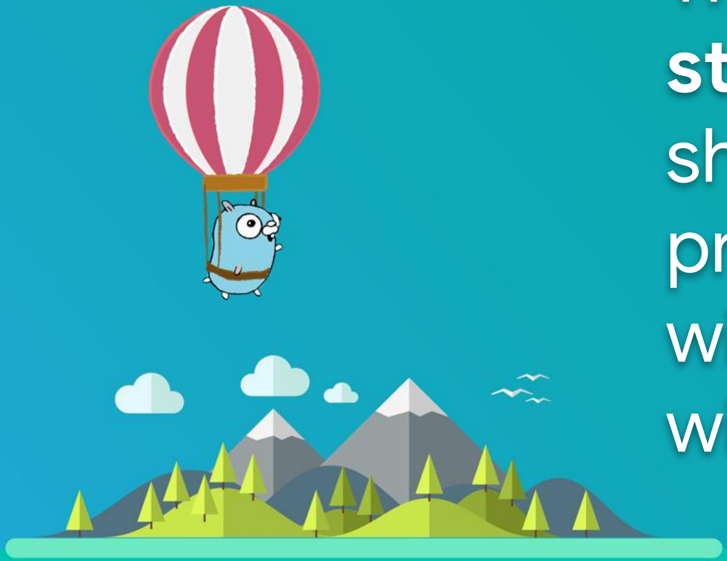
Go is open sourced

2012



Stability

Go 1.0 is released and
attention shifts to using Go



> “The driving motivation for Go 1 is **stability for its users.** People should be able to write Go programs and expect that they will continue to compile and run without change ...”

Go 1 Release Notes
March 28, 2012

Go 1 and the Future of Go Programs

> “**It is intended that programs written to the Go 1 specification will continue to compile and run correctly, unchanged, over the lifetime of that specification.** At some indefinite point, a Go 2 specification may arise, but until that time, **Go programs that work today should continue to work even as future ‘point’ releases of Go 1 arise** (Go 1.1, Go 1.2, etc.).”

go.dev/doc/go1compat

Go 1.0: A platform for software engineering

Go 1.0: A platform for software engineering

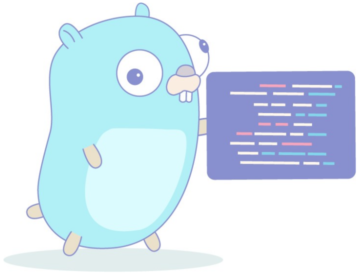
- Compatibility promise
- Testing
- Formatting
- Primitive dependency management
- Profiling

Go 1.0: A platform for software engineering

- Compatibility promise
- Testing
- Formatting
- Primitive dependency management
- Profiling
- **go command**

A few more years of refinement

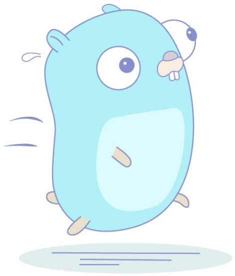
2009



Open Source

Go is open sourced

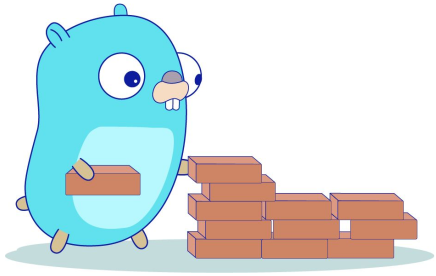
2012



Stability

Go 1.0 is released and attention shifts to using Go

2015



Refinement

Go is **refined significantly** through version 1.5

Go 1.5: A refined platform for software engineering

Go 1.5: A refined platform for software engineering

- Compiler & runtime rewritten entirely in Go
- Concurrent, low-latency garbage collection
- Semi-annual release cycle
- Hardening of the standard library
- Performance improvements throughout

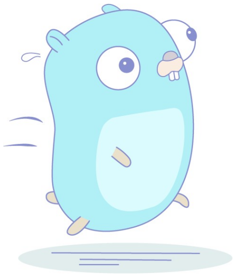
“

Go 1.5 Release Notes

Version 1.5 is a significant release, including major architectural changes to the implementation. Despite that, we expect almost all Go programs to continue to compile and run as before, **because the release still maintains the Go 1 promise of compatibility.**

Shifting focus to the broader ecosystem

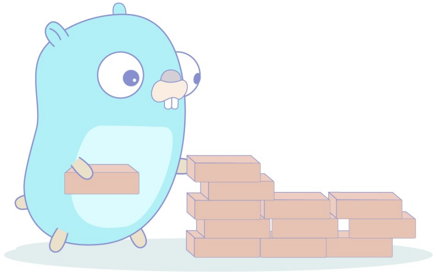
2012



Stability

Go 1.0 is released and attention shifts to using Go

2015



Refinement

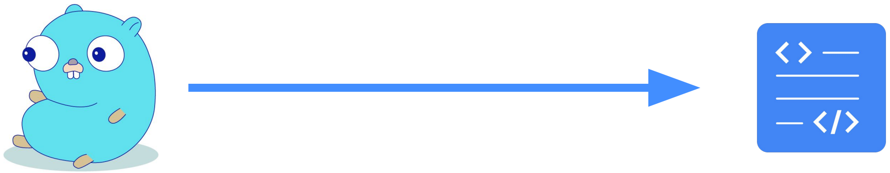
Go is refined significantly through version 1.5

2018

Ecosystem

Modules are introduced and **attention shifts to the broader Go ecosystem**

Go Modules

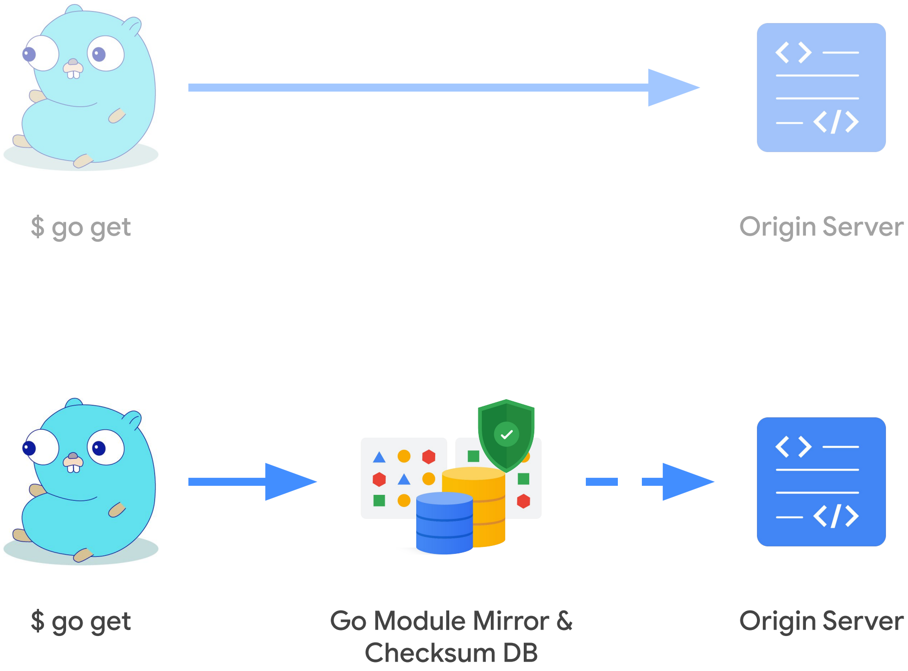


$ go get

Origin Server

Go Modules

Go’s **module mirror** and **checksum database** ensure that the bits you get the first time are the same bits you get every time.



Go 1.11+: A secure platform for software engineering

Go 1.11+: A secure platform for software engineering

- Modules
- Module mirror & checksum database

Go 1.11+: A secure platform for software engineering

- Modules
- Module mirror & checksum database
- **SBOM**

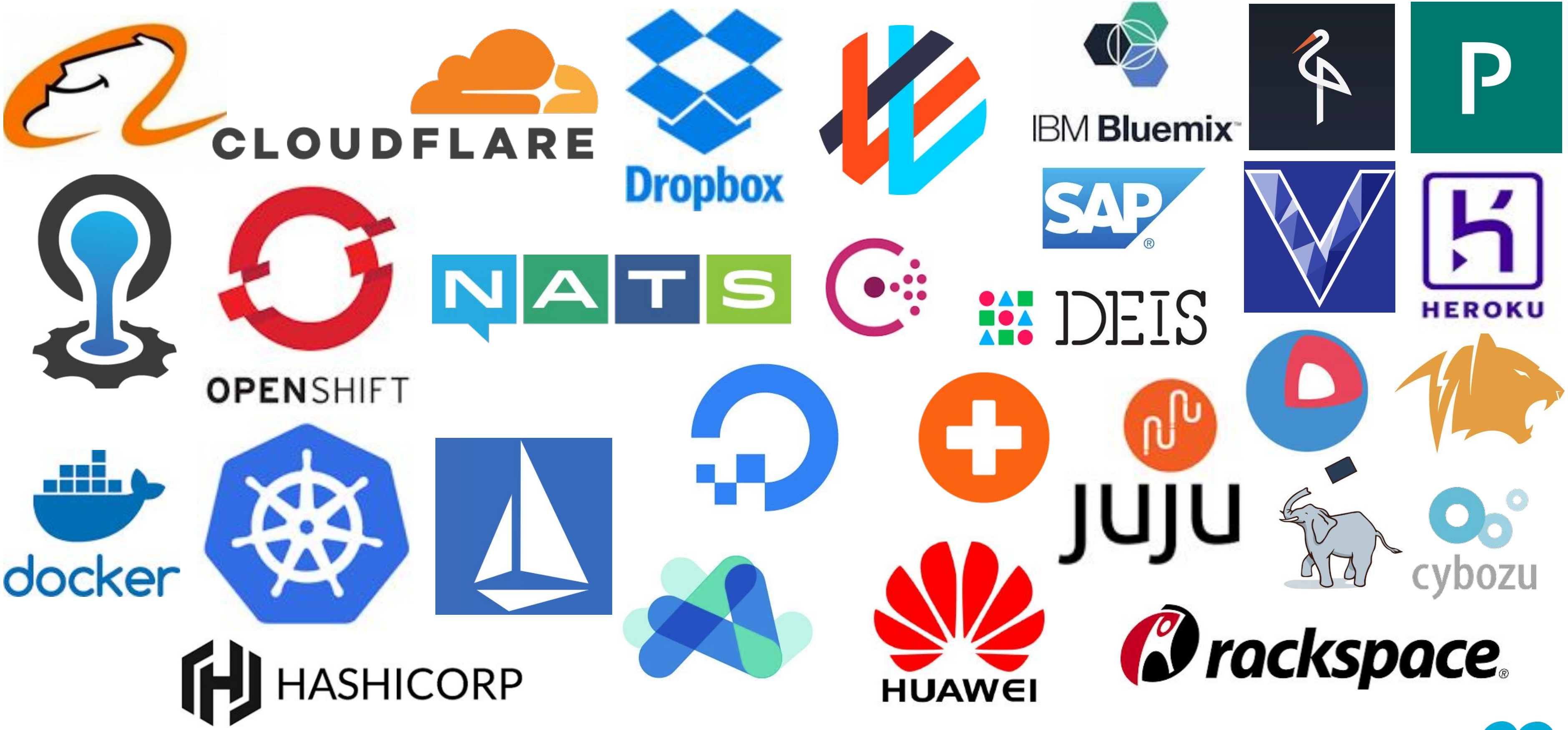
Go 1.11+: A secure platform for software engineering

- Modules
- Module mirror & checksum database
- SBOM
- **Fuzzing**

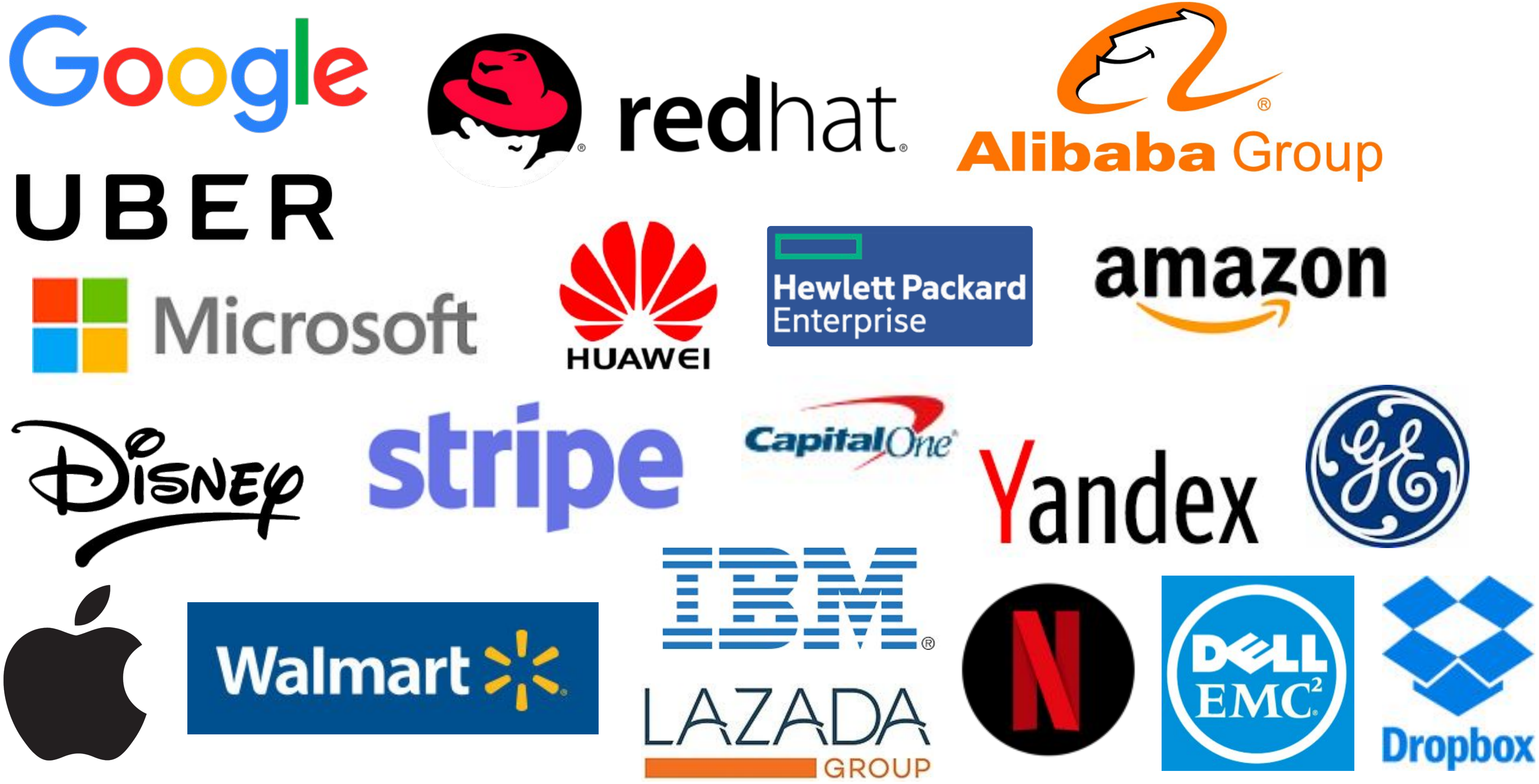
Go 1.11+: A secure platform for software engineering

- Modules
- Module mirror & checksum database
- SBOM
- Fuzzing
- **Vulnerability management**

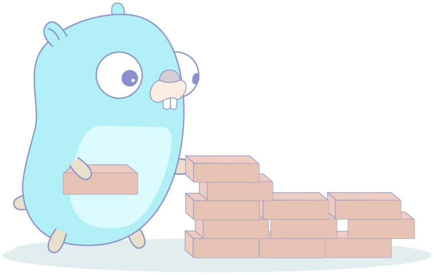
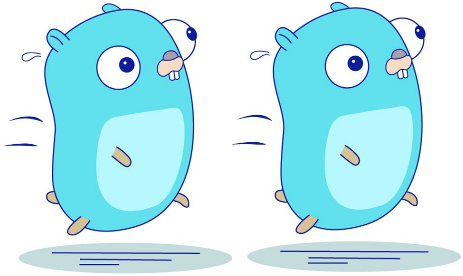
The cloud is built on Go



Enterprises adopt Go



Ten years of experience and evolution

2015	2018	2022
		
Refinement	**Ecosystem**	**Generics**
Go is refined significantly through version 1.5	Modules are introduced and attention shifts to the broader Go ecosystem	Go 1.18 is released, with the **most significant change ever made** to the language

Go 1 and the Future of Go Programs

“It is intended that programs written to the Go 1 specification will continue to compile and run correctly, unchanged, over the lifetime of that specification. **At some indefinite point, a Go 2 specification may arise,** but until that time, Go programs that work today should continue to work even as future ‘point’ releases of Go 1 arise (Go 1.1, Go 1.2, etc.).”

go.dev/doc/go1compat

When will Go 2 happen?

When will Go 2 happen?

Never. Compatibility is more valuable than any break with the past.

“

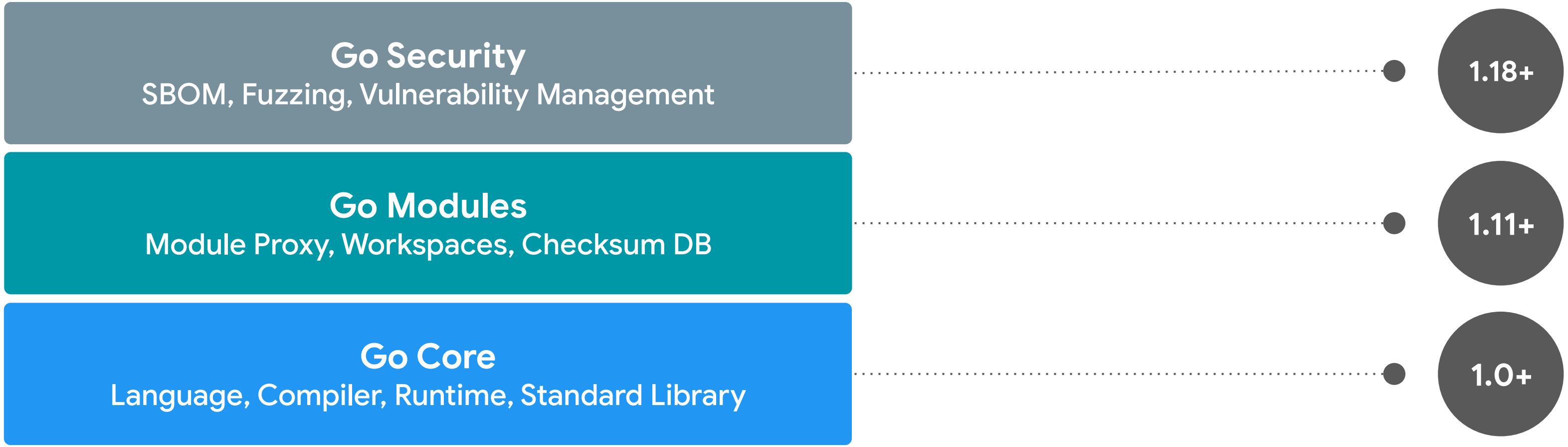
Go Release Notes

As always, the release **maintains the Go 1 promise of compatibility**. We expect almost all Go programs to continue to compile and run as before.

SECTION TWO

Where we are today

Go is a complete, secure platform for software engineering



Go Security
SBOM, Fuzzing, Vulnerability Management

1.18+

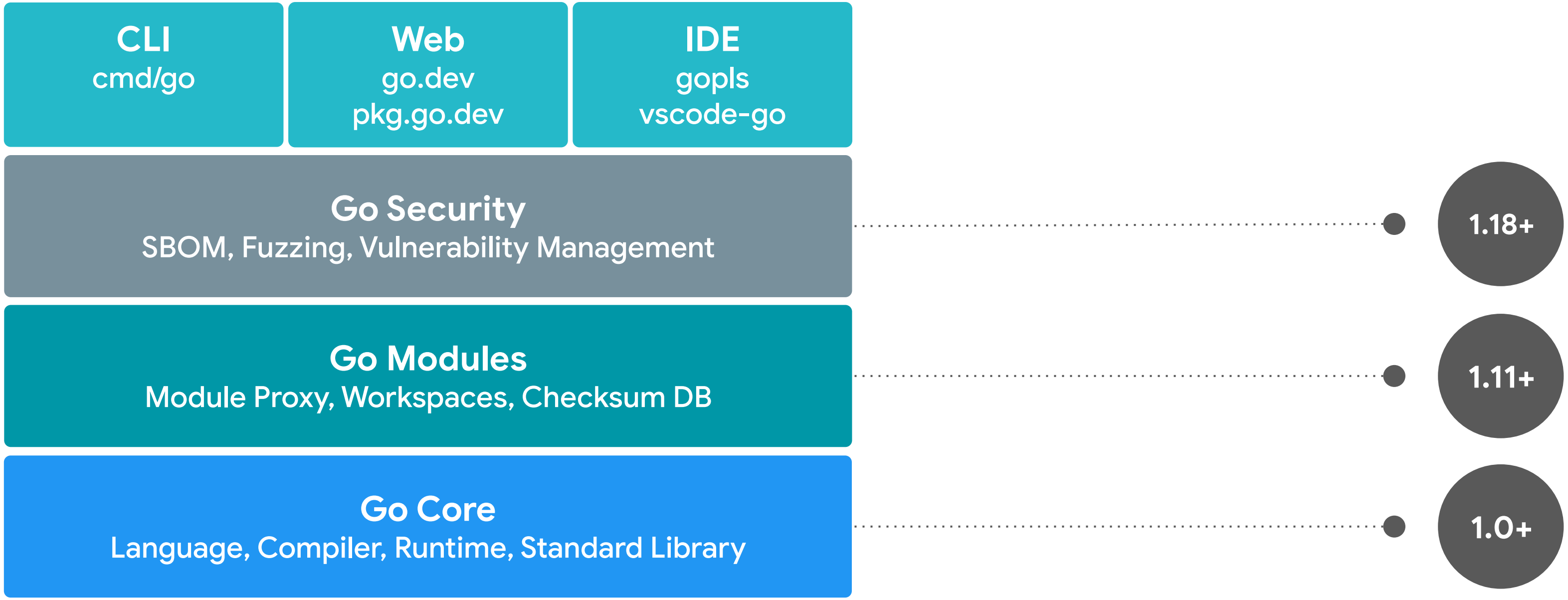
Go Modules
Module Proxy, Workspaces, Checksum DB

1.11+

Go Core
Language, Compiler, Runtime, Standard Library

1.0+

Go is a complete, secure platform for software engineering



CLI
cmd/go

Web
go.dev
pkg.go.dev

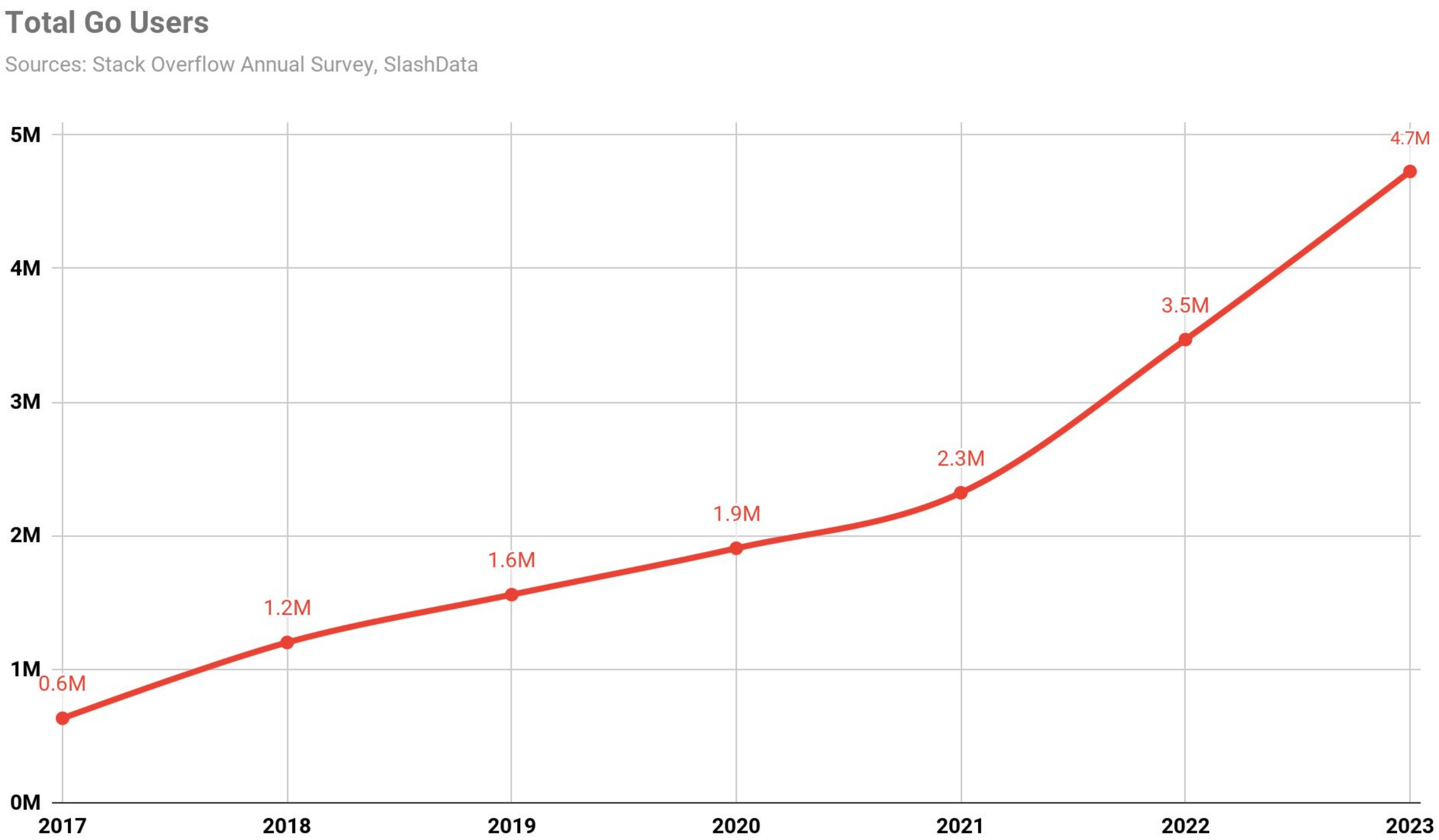
IDE
gopls
vscode-go

Go Security
SBOM, Fuzzing, Vulnerability Management

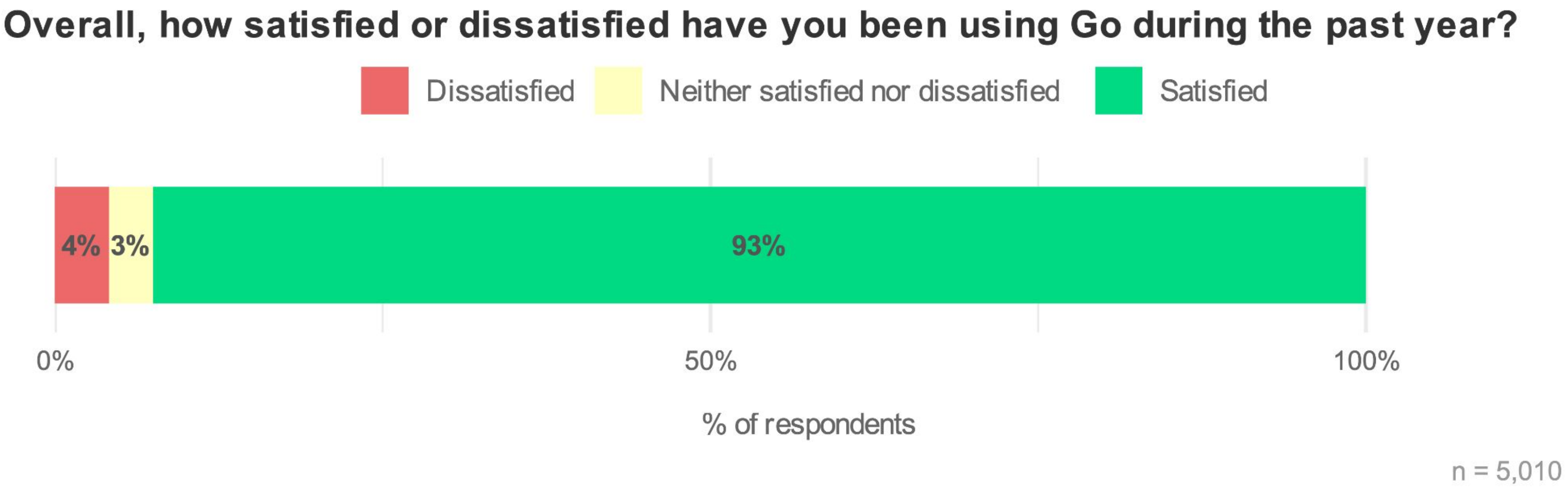
Go Modules
Module Proxy, Workspaces, Checksum DB

Go Core
Language, Compiler, Runtime, Standard Library

Go’s user base has grown 4x since 2018

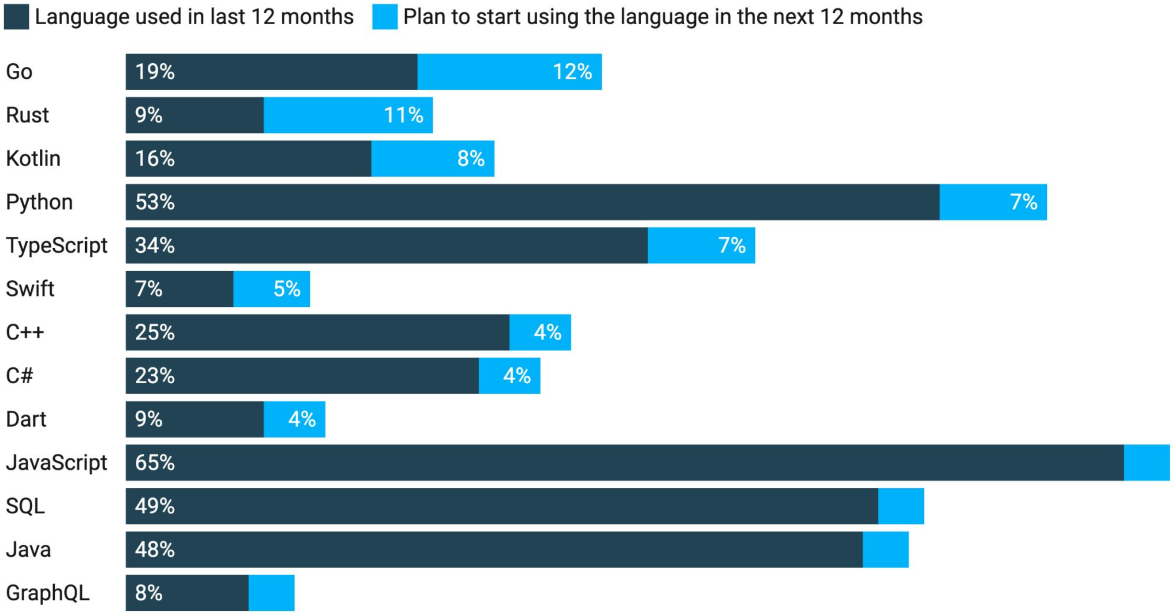


Go users are satisfied



THENEWSTACK

Go, Rust and Kotlin Expect the Largest Increase in New Users in 2023

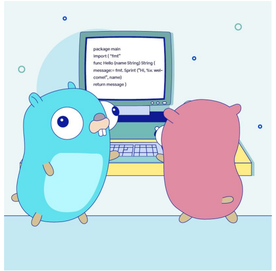


The only languages displayed are those with at least 3% planning to use it. Respondents that used a language in the last 12 months were not asked if they plan to start using that language.

Chart: Heather Joslyn and Lawrence Hecht • Source: "State of the Developer Ecosystem, 2022," JetBrains. • Get the data • Download image • Created with Datawrapper

Go provides the most **productive platform** for building **production systems**

Go provides the most productive platform for building production systems

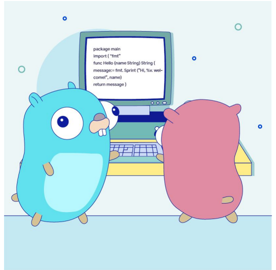


```
package main
import ( "fmt"
func Hello (name String) String {
message:= fmt. Sprint ("Hi, %v. wel-
come!", name)
return message }
```

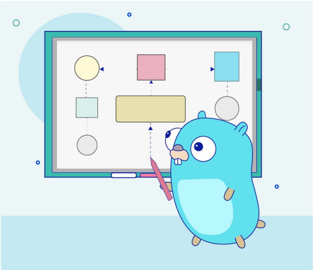
Go is Productive

Go is simple to **learn**, simple to **maintain**, **readable**, and **scales** well across teams, workloads, and use cases.

Go provides the most productive platform for building production systems



```
package main
import ( "fmt"
func Hello (name String) String {
message:= fmt. Sprint ("Hi, %v. wel-
come!", name)
return message }
```



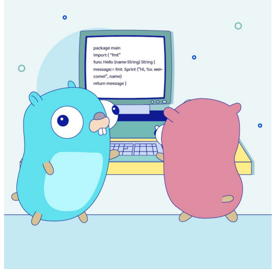
Go is Productive

Go is simple to **learn**, simple to **maintain**, **readable**, and **scales** well across teams, workloads, and use cases.

Go is a Platform

Go is not just a language, it is a **complete developer experience** across the development lifecycle.

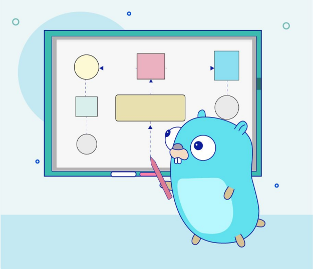
Go provides the most productive platform for building production systems



```
package main
import { "fmt"
func Hello (name String) String {
message:= fmt. Sprint ("Hi, %v. wel-
come!", name)
return message }
```

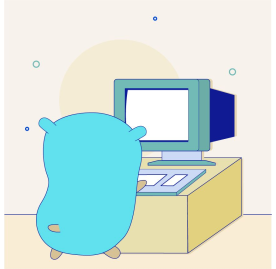
Go is Productive

Go is simple to **learn**, simple to **maintain**, **readable**, and **scales** well across teams, workloads, and use cases.



Go is a Platform

Go is not just a language, it is a **complete developer experience** across the development lifecycle.



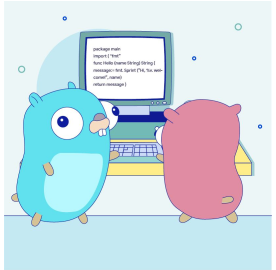
Go is Production Ready

Go is **reliable**, **efficient**, **stable**, and **secure**, making it well suited for critical business systems and infrastructure.

SECTION THREE

Where we are going next

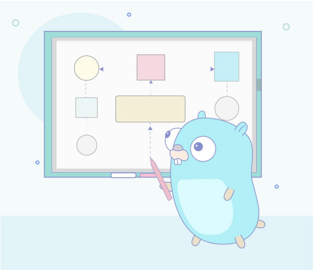
Go provides the most productive platform for building production systems



```
package main
import { "fmt"
func Hello (name String) String {
message:= fmt. Sprint ("Hi, %v. wel-
come!", name)
return message }
```

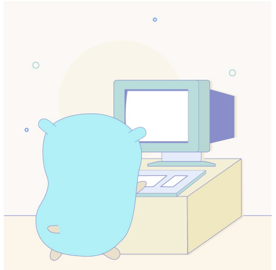
Go is Productive

Go is simple to **learn**, simple to **maintain**, **readable**, and **scales** well across teams, workloads, and use cases.



Go is a Platform

Go is not just a language, it is a **complete developer experience** across the development lifecycle.



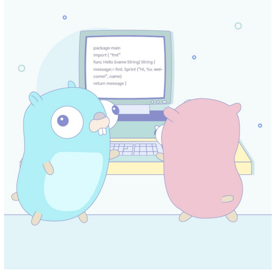
Go is Production Ready

Go is **reliable**, **efficient**, **stable**, and **secure**, making it well suited for critical business systems and infrastructure.

Improve Go developer productivity

- Generics
- Loop var fix
- Iterators
- Onboarding

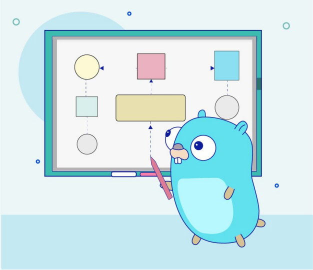
Go provides the most productive platform for building production systems



```
package main
import ( "fmt"
func Hello (name String) String {
message:= fmt. Sprint ("Hi, %v. wel-
come!", name)
return message }
```

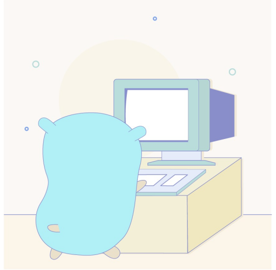
Go is Productive

Go is simple to **learn**, simple to **maintain**, **readable**, and **scales** well across teams, workloads, and use cases.



Go is a Platform

Go is not just a language, it is a **complete developer experience** across the development lifecycle.



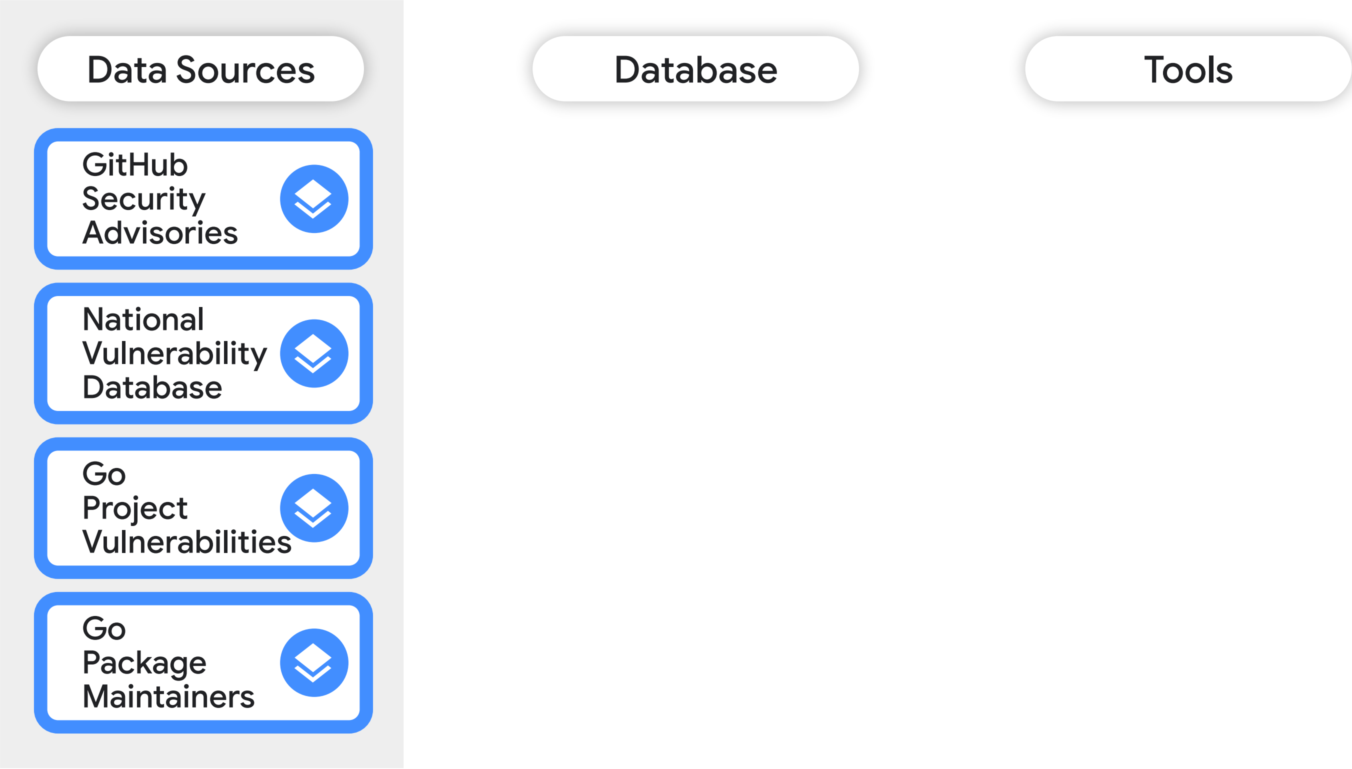
Go is Production Ready

Go is **reliable**, **efficient**, **stable**, and **secure**, making it well suited for critical business systems and infrastructure.

Enrich the Go platform

- Package ecosystem
- gopls
- Visual Studio Code Go
- Vulnerability management

Go Vulnerability Management System



Data Sources

GitHub Security Advisories

National Vulnerability Database

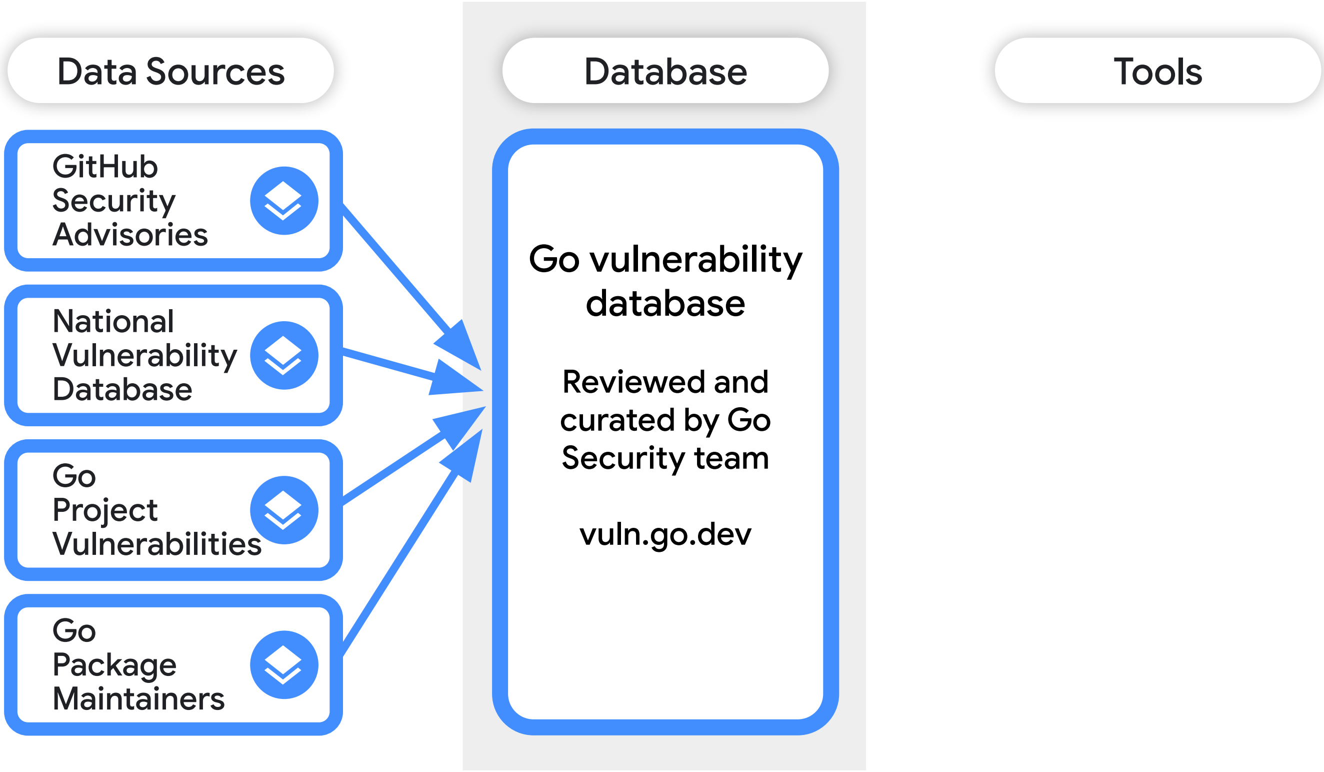
Go Project Vulnerabilities

Go Package Maintainers

Database

Tools

Go Vulnerability Management System



Data Sources

Database

Tools

GitHub Security Advisories

National Vulnerability Database

Go Project Vulnerabilities

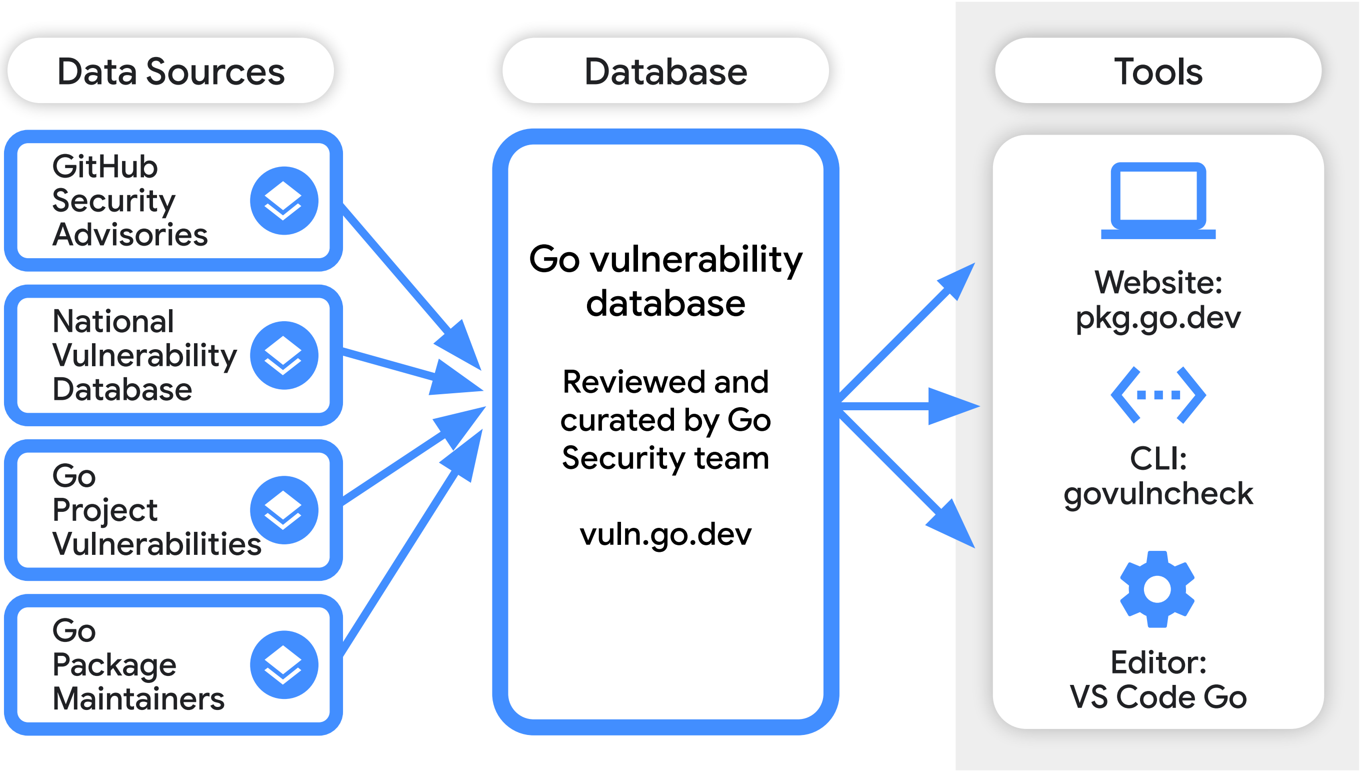
Go Package Maintainers

Go vulnerability database

Reviewed and curated by Go Security team

vuln.go.dev

Go Vulnerability Management System



Data Sources

- GitHub Security Advisories
- National Vulnerability Database
- Go Project Vulnerabilities
- Go Package Maintainers

Database

Go vulnerability database

Reviewed and curated by Go Security team

vuln.go.dev

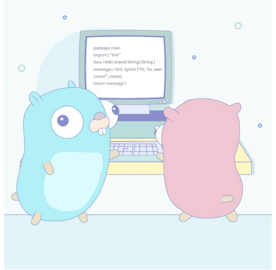
Tools

Website: pkg.go.dev

CLI: govulncheck

Editor: VS Code Go

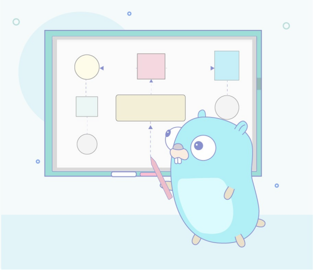
Go provides the most productive platform for building production systems



```
package main
import ( "fmt"
func Hello (name String) String {
message:= fmt. Sprint ("Hi, %v. wel-
come!", name)
return message }
```

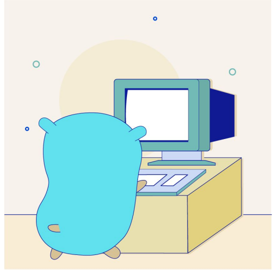
Go is Productive

Go is simple to **learn**, simple to **maintain**, **readable**, and **scales** well across teams, workloads, and use cases.



Go is a Platform

Go is not just a language, it is a **complete developer experience** across the development lifecycle.



Go is Production Ready

Go is **reliable**, **efficient**, **stable**, and **secure**, making it well suited for critical business systems and infrastructure.

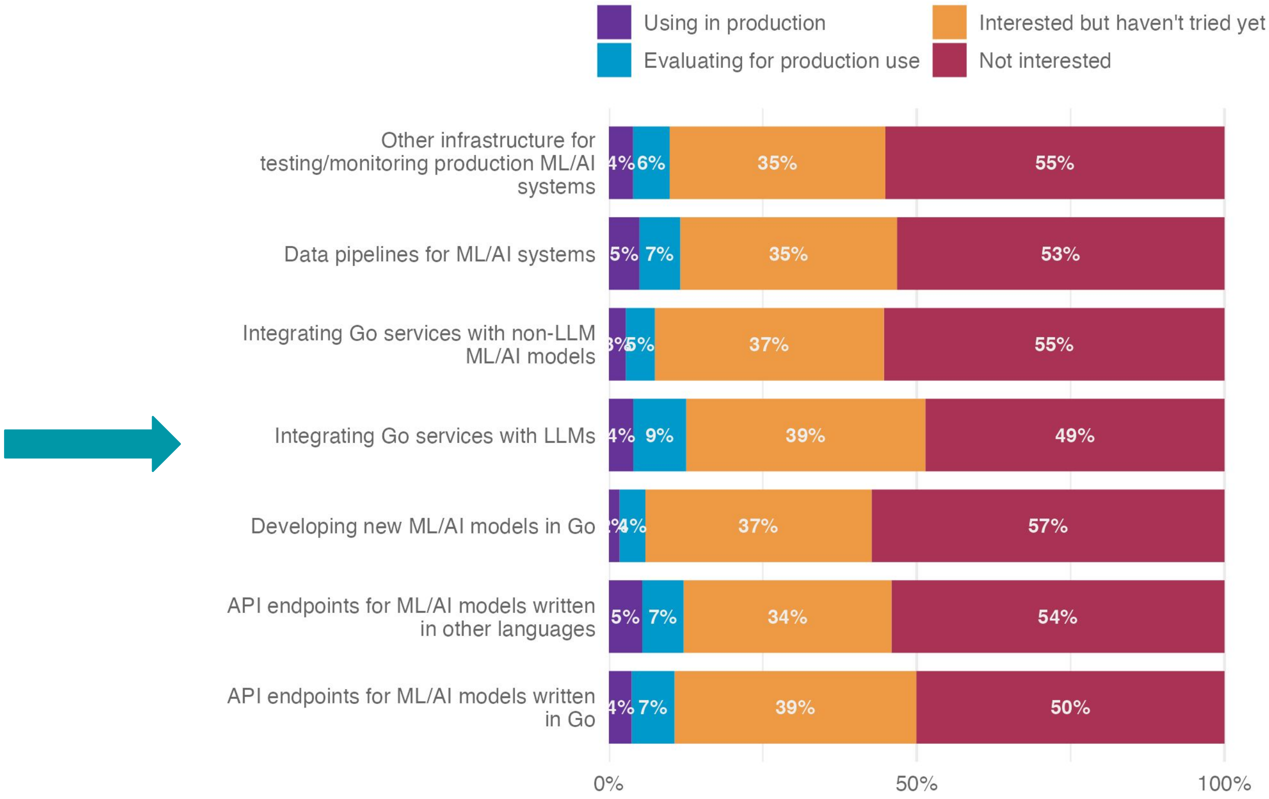
Enhance Go's production readiness

- Backward and forward compatibility
- Profile guided optimization (PGO)
- RAM efficiency
- Performance debugging
- Execution tracer
- Reproducible toolchain builds
- Standard library, e.g., slog, QUIC, HTTP mux

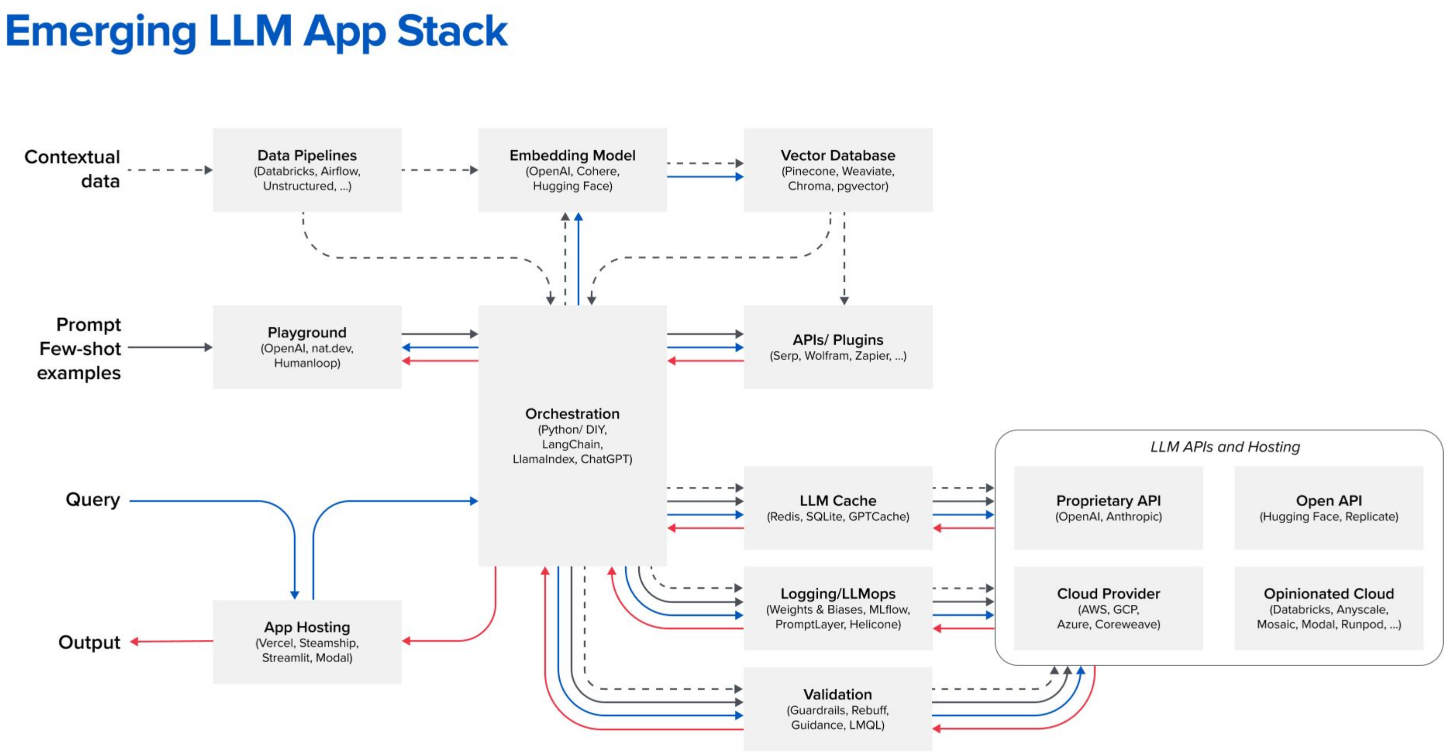
Enhance Go's **production readiness**

- Backward and forward compatibility
- Profile guided optimization (PGO)
- RAM efficiency
- Performance debugging
- Execution tracer
- Standard library, e.g., slog, QUIC, HTTP mux
- **AI-powered applications**

Most Go users are interested in LLM integrations



Go is ideal for productionizing AI-powered products



Emerging LLM App Stack

Contextual data

Data Pipelines
(Databricks, Airflow, Unstructured, ...)

Embedding Model
(OpenAI, Cohere, Hugging Face)

Vector Database
(Pinecone, Weaviate, Chroma, pgvector)

Prompt
Few-shot examples

Playground
(OpenAI, nat.dev, Humanloop)

Orchestration
(Python/ DIY, LangChain, LlamaIndex, ChatGPT)

APIs/ Plugins
(Serp, Wolfram, Zapier, ...)

Query

Output

App Hosting
(Vercel, Steamship, Streamlit, Modal)

LLM Cache
(Redis, SQLite, GPTCache)

Logging/LLMops
(Weights & Biases, MLflow, PromptLayer, Helicone)

Validation
(Guardrails, Rebuff, Guidance, LMQL)

LLM APIs and Hosting

Proprietary API
(OpenAI, Anthropic)

Open API
(Hugging Face, Replicate)

Cloud Provider
(AWS, GCP, Azure, Coreweave)

Opinionated Cloud
(Databricks, Anyscale, Mosaic, Modal, Runpod, ...)

https://a16z.com/emerging-architectures-for-llm-applications

Go’s community will pave the way

- Libraries
- Documentation
- Examples
- Tutorials
- Contributions
- Tools

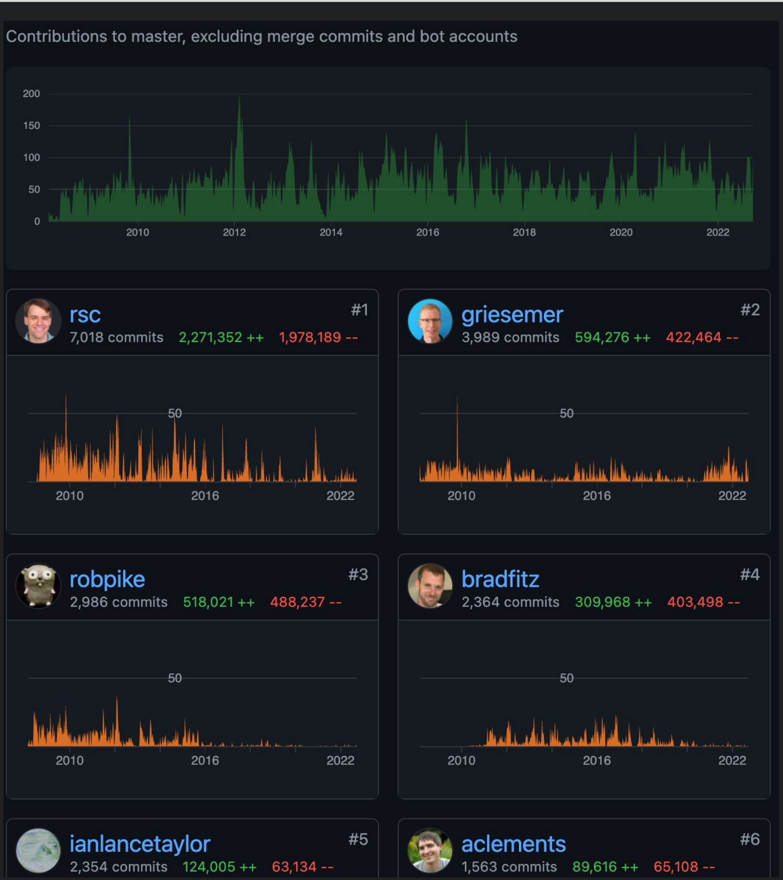
cbalahan@google.com

SECTION FOUR

How we will succeed

Go’s Community

Go’s contributor base grows



Contributions to master, excluding merge commits and bot accounts

rsc #1
7,018 commits 2,271,352 ++ 1,978,189 --

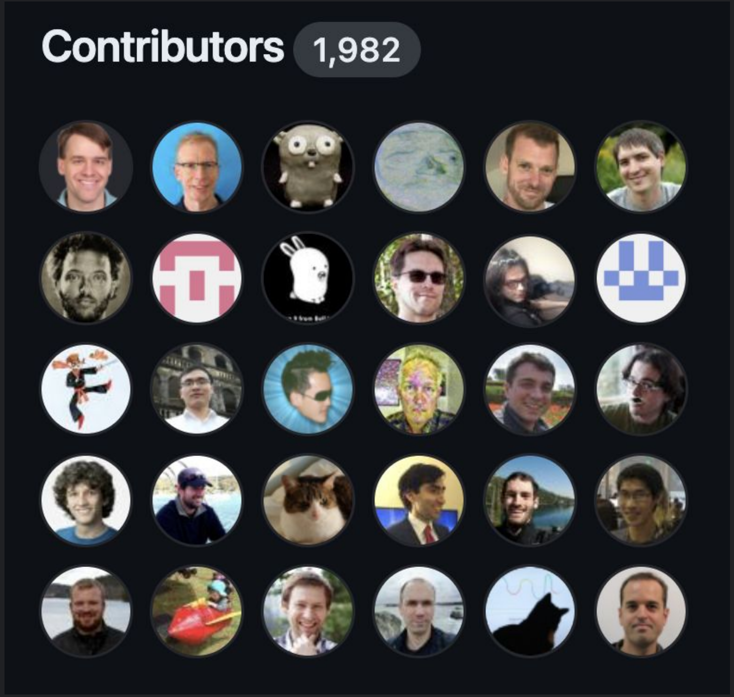
griesemer #2
3,989 commits 594,276 ++ 422,464 --

robpike #3
2,986 commits 518,021 ++ 488,237 --

bradfitz #4
2,364 commits 309,968 ++ 403,498 --

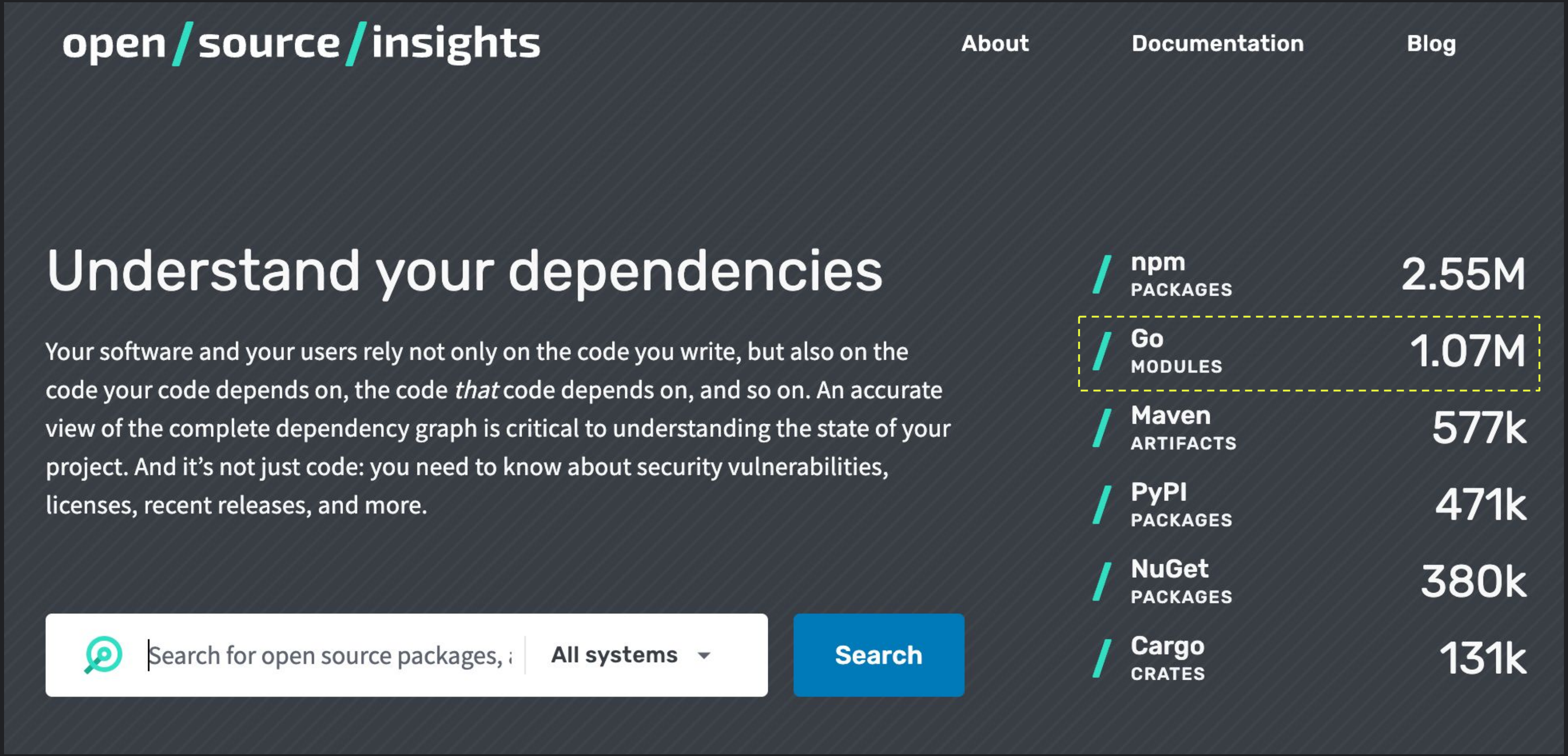
ianlancetaylor #5
2,354 commits 124,005 ++ 63,134 --

aclements #6
1,563 commits 89,616 ++ 65,108 --



Contributors 1,982

Go’s ecosystem grows

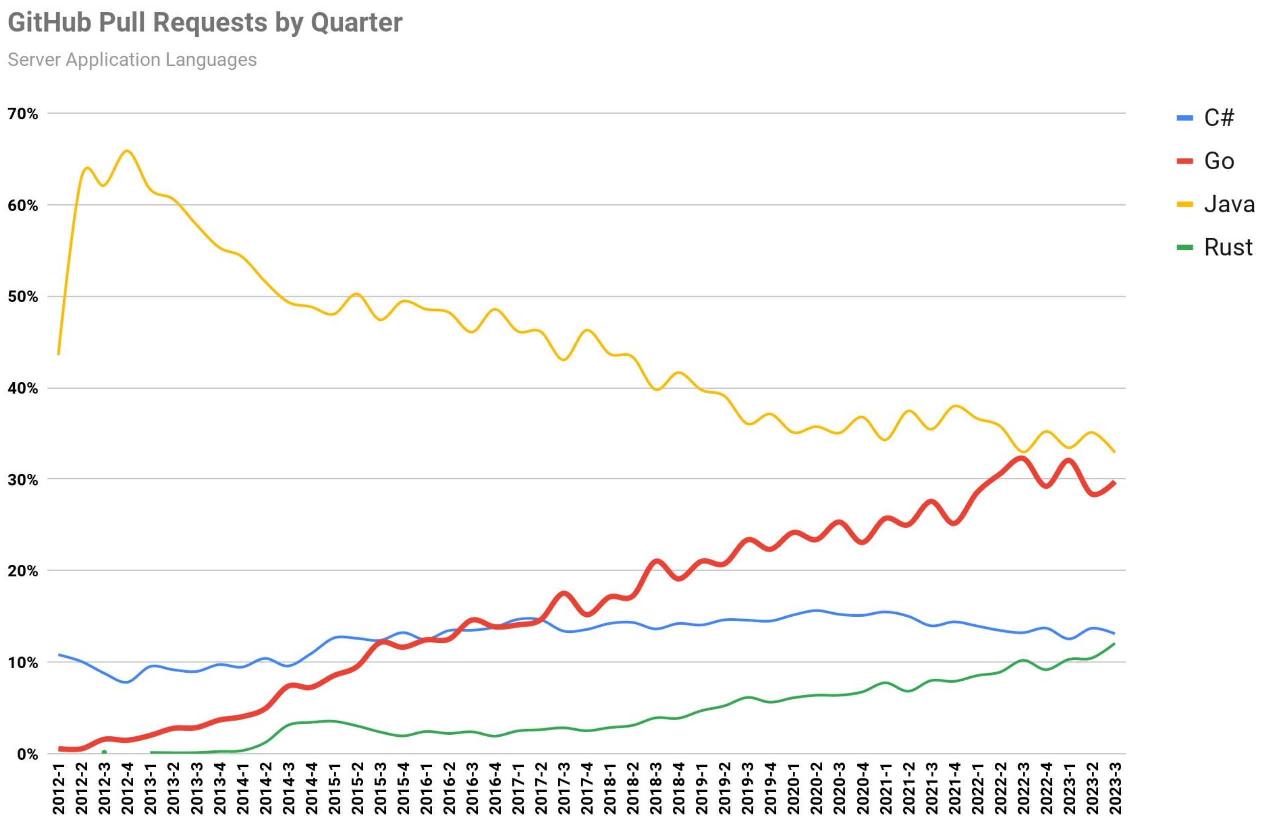


Understand your dependencies

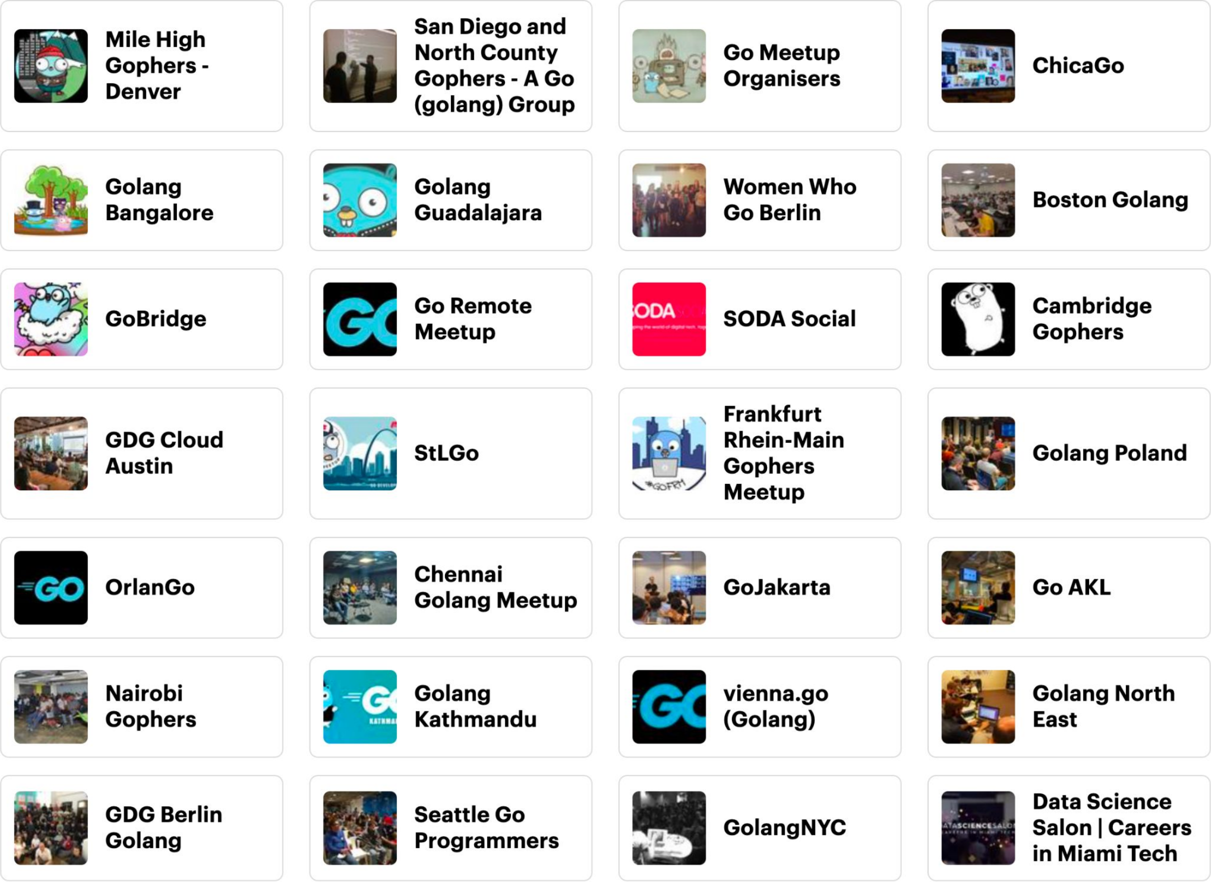
Your software and your users rely not only on the code you write, but also on the code your code depends on, the code *that* code depends on, and so on. An accurate view of the complete dependency graph is critical to understanding the state of your project. And it’s not just code: you need to know about security vulnerabilities, licenses, recent releases, and more.

npm PACKAGES	2.55M
Go MODULES	1.07M
Maven ARTIFACTS	577k
PyPI PACKAGES	471k
NuGet PACKAGES	380k
Cargo CRATES	131k

Go grows in open source



Go's community is you



- Mile High Gophers - Denver
- San Diego and North County Gophers - A Go (golang) Group
- Go Meetup Organisers
- ChicaGo
- Golang Bangalore
- Golang Guadalajara
- Women Who Go Berlin
- Boston Golang
- GoBridge
- Go Remote Meetup
- SODA Social
- Cambridge Gophers
- GDG Cloud Austin
- StLGo
- Frankfurt Rhein-Main Gophers Meetup
- Golang Poland
- OrlanGo
- Chennai Golang Meetup
- GoJakarta
- Go AKL
- Nairobi Gophers
- Golang Kathmandu
- vienna.go (Golang)
- Golang North East
- GDG Berlin Golang
- Seattle Go Programmers
- GolangNYC
- Data Science Salon | Careers in Miami Tech

GoLab Florence – November 2023

Thank You

Cameron Balahan

Google

@cameronbalahan