**Go** or **Golang** is a programming language created at Google, by Google developers and other programmers. This programming language is free and open source and currently being maintained by Google. One of the founding member of Go is [**Ken Thompson**](https://en.wikipedia.org/wiki/Ken_Thompson) who is best known for his work in Unix operating system development. Go compiler was initially written in C but now, it is [written in Go itself](https://www.quora.com/How-is-it-possible-that-the-Go-language-is-written-in-the-Go-language), making it self hosted.

### **Why was Go needed**

The authors mention that the primary motive for designing a new language was to solve software engineering issues at Google. They also mention that Go was actually developed as an alternative to C++.

**Motivation**

Rob Pike mentions the purpose for the Go programming language:

“Go’s purpose is therefore not to do research into programming language design; it is to improve the working environment for its designers and their coworkers. Go is more about software engineering than programming language research. Or to rephrase, it is about language design in the service of software engineering.”

**Issues that were plaguing the software engineering horizon at Google were**(taken from [https://talks.golang.org/2012/splash.article](https://talks.golang.org/2012/splash.article%29:)):

a) slow builds — builds would sometime take as long as an hour to complete  
 b) uncontrolled dependencies  
 c) each programmer using a different subset of the language  
 d) poor program understanding (code hard to read, poorly documented, and so on)  
 e) duplication of effort  
 f) cost of updates  
 g) version skew  
 h) difficulty of writing automatic tools  
 i) cross-language builds

### Target Audience

Go is a systems programming language. Go really shines for stuff such as cloud systems (web servers, caches), microservices, distributed systems (due to concurrency support).

### **Strengths**

1. **Statically typed:** Go is statically typed. This means that you need to declare types for all your variables and your function arguments (and return variables) at compile time. Although this may sound inconvenient, this is a great advantage since a lot of errors will be found at compile time itself. This factor plays a very big role when your team size increases, since declared types make functions and libraries more readable and more easier to understand.
2. **Compilation Speed:** Go code compiles **really fast**, so you don’t need to keep waiting for your code to compile. :) In fact, the ‘go run’ command fires up your Go program so quickly so that you don’t even get a feeling that your code got compiled first. It feels like an interpreted language.
3. **Execution Speed:** Go code gets directly compiled to machine code, depending upon the OS (Linux/Windows/Mac) and the CPU instruction set architecture (x86, x86–64, arm etc) of the machine the code is being compiled upon. So, it runs really fast.
4. **Portable:** Since the code gets directly compiled to machine code, therefore, the binaries become portable. Portability here means that you can pick up the binary from your machine (let’s say Linux, x86–64) and directly run that on your server (if your server is also running Linux on a x86–64 architecture).This becomes possible since Go binaries are statically linked, meaning that any shared operating system libraries your program needs are included in the binary at the time of the compilation. They are not dynamically linked at the time of running the program.

This has a huge benefit for deployment of your programs on multiple machines in a data center. If you have 100 machines in your data center, you can simply ‘scp’ your program binary to all of them, as long as the binary is compiled for the same OS and instruction set architecture your machines run on. You don’t need to care about which version of Linux they are running. There is no need for checking/managing dependencies. The binaries simply run and all your services are up :)

1. **Concurrency:** Go has first class support for concurrency. Concurrency is one of the major selling points of Go. The language designers have designed the concurrency model around the ‘[Communicating Sequential Processes](http://www.cs.cmu.edu/~crary/819-f09/Hoare78.pdf)’ paper by Tony Hoare.**The Go runtime allows you to run hundreds of thousands of concurrent goroutines on a machine**. A Goroutine is a lightweight thread of execution. The Go runtime multiplexes those goroutines over operating system threads. That means that multiple goroutines can run concurrently on a single OS thread. The Go runtime has a scheduler whose job is to schedule these goroutines for execution.

There are two benefits of this approach:

i) A Goroutine when initialized has a stack of 4 KB. This is really tiny as compared to a stack of an OS thread, which is generally 1 MB. This number matters when you need to have hundreds of thousands of different goroutines running concurrently. If you would run more than thousands of OS threads in parallel, the RAM obviously will become a bottleneck.

ii) Go could have followed the same model as other languages like Java, which support the same concept of threads as OS threads. But in that case, the cost of a context switch between OS threads is much larger than the cost of a context switch between different goroutines.

**f) Garbage collection:** Unlike C, you don’t need to remember to free up pointers or worry about dangling pointers in Go. The garbage collector automatically does this job.

**g) No exceptions, handle errors yourself:** I love the fact that Go doesn’t have the standard exception logic that other languages have. Go forces developers to handle basic errors like ‘couldn’t open file’ etc rather than letting them wrap up all of their code in a try catch block. This also puts pressure on developers to actually think about what needs to be done to handle these failure scenarios.

**h) Amazing tooling:** One of the best aspects about Go is its tooling. It has tools like:

i) [Gofmt:](https://blog.golang.org/go-fmt-your-code" \t "_blank) It automatically formats and indents your code so that your code looks like the same as every Go developer on the planet. This has a huge effect on code readability.

ii) [Go run:](https://golang.org/cmd/go/#hdr-Compile_and_run_Go_program) This compiles your code and runs it, both :). So even though Go needs to be compiled, this tool makes you feel like it’s an interpreted language since it just compiles your code so fast that you don’t even feel when the code got compiled.

iii) [Go get:](https://golang.org/cmd/go/#hdr-Download_and_install_packages_and_dependencies) This downloads the library from GitHub and copies it to your GoPath so that you can import the library in your project

iv) [Godoc:](https://godoc.org/golang.org/x/tools/cmd/godoc" \t "_blank) Godoc parses your Go source code — including comments — and produces its documentation in HTML or plain text format. Through godoc’s web interface, you can then see documentation tightly coupled with the code it documents. You can navigate from a function’s documentation to its implementation with one click.

You can check more [tools](https://golang.org/cmd/go/) here.

**j) Great built-in libraries:** Go has great built-in libraries to aid modern development. Some of them are:

a) [net/http](https://golang.org/pkg/net/http/) — Provides HTTP client and server implementations

b) [database/sql](https://golang.org/pkg/database/sql/) — For interaction with SQL databases

c) [encoding/json](https://golang.org/pkg/encoding/json/) — JSON is treated as a first class member of the standard language :)

d) [html/templates](https://golang.org/pkg/html/template/) — HTML templating library

e) [io/ioutil](https://golang.org/pkg/io/ioutil/" \t "_blank) — Implements I/O utility functions

There is a lot of development going on in the Go horizon. You can find all Go libraries and frameworks for all sorts of tools and use cases [here](https://github.com/avelino/awesome-go).

**So what’s so special about Go?**

* Go is statically typed compiled language.
* It supports built in concurrency model with the help of **GoRoutines and channels**.
* It has built in [garbage collector](https://en.wikipedia.org/wiki/Garbage_collection_(computer_science)) and [memory safety](https://en.wikipedia.org/wiki/Memory_safety)(!).
* Strings in Go are [UTF-8](https://en.wikipedia.org/wiki/UTF-8) encoded by default.
* It has simpler syntax compared to other compiled programming languages.