01 Go Lang

Go Is Fast
Go is a really fast language. Because Go is compiled to machine code, it will naturally outperform languages that are interpreted or have virtual runtimes. Go programs also compile extremely fast, and the resulting binary is very small. Our API compiles in seconds and produces an executable file that is 11.5 MB. You can't complain about that.

Static Typing
Go is a strongly, statically typed language. There are primitive types like int, byte, and string. There are also structs. Like any strongly typed language, the type system allows the compiler helps catch entire classes of bugs. Go also has built-in types for lists and maps, and they are easy to use.

Interface Types
Go has interfaces, and any struct can satisfy an interface simply by implementing its methods. This allows you to decouple the dependencies in your code. You can then mock your dependencies in tests. By using interfaces, you can write more modular, testable code. Go also has first-class functions, which open up the possibility to write your code in a more functional style.

Standard Library
Go has a nice standard library. It provides handy built-in functions for working with primitive types. There are packages that make it easy to stand up a web server, handle I/O, work with cryptography, and manipulate raw bytes. JSON serialization and deserialization provided by the standard library is trivial. With the use of "tags", you can specify JSON field names right next to struct fields.

Testing Support
Testing support is built into the standard library. There is no need for

an extra dependency. If you have a file called thing.go, write your tests in another file called thing_test.go, and run “go test”. Go will execute these tests fast.

Static Analysis Tools
Go static analysis tools are numerous and robust. One in particular is gofmt, which formats your code according to Go’s suggested style. This can normalize a lot of opinions on a project and free your team to focus on what the code is doing. We run gofmt, golint, and vet on every build, and if any warnings are found, the build fails.

Garbage Collection
Memory management in Go was intentionally made easier than in C and C++. Dynamically allocated objects are garbage collected. Go makes using pointers much safer because it doesn’t allow pointer arithmetic. It also gives you the option of using value types.

Easier Concurrency Model
While concurrent programming is never easy, Go makes it easier than in other languages. It is almost trivial to create a lightweight thread, called a “goroutine”, and communicate with it via a “channel.” More complex patterns are possible.