

# TechTalk #3

What do people say  
when they switch to Go?

Vũ Nguyễn  
vu.nguyen@will.vn

Go is an optional language released in 2012

( It does not force you to use it like  
Java for Android or Obj-C for iOS )

Why is it popular today?

## Search

 **Repositories**

22,463

We've found 22,463 repository

 **Code**

 **Issues**

187,602

[ethereum/go-ethereum](#)

 **Users**

03/2015

## Languages

JavaScript 371,865

Ruby 245,139

Java 213,398

Python 204,142

PHP 167,570

C 97,485

C++ 88,020

Objective-C 62,205

Shell 58,670

C# 57,874

[wcong/ants-go](#)

ants in go

Updated 5 minutes ago

[hkwi/gopenflow](#)

golang openflow implementation

Updated 10 minutes ago

## Search

 **Repositories**

118,113

We've found 118,113 repository results

 **Code**

 **Issues**

523,309

[docker/docker](#)

 **Users**

04/2016

## Languages

JavaScript 2,132,721

Java 1,868,868

Python 1,031,472

Ruby 997,530

PHP 842,018

HTML 762,206

CSS 669,298

C++ 555,223

C 476,413

C# 470,911

[golang/go](#)

The Go programming language

Updated an hour ago

[kubernetes/kubernetes](#)

Container Cluster Manager from Google

Updated 3 hours ago

[Advanced search](#) [Cheat sheet](#)

[gogits/gogs](#)

## Who are using Go?

Google, Facebook, Microsoft, Amazon, Mozilla,  
Yahoo, eBay, GitHub, Twitter, Dropbox, Docker,  
MongoDB, Couchbase, Disqus, Basecamp,  
CoreOS, CloudFlare, ...



Google has big softwares and big problems.

The hardware is **big** and the software is **big**.

There are **many millions of lines** of software,  
with servers **mostly in C++** and lots of Java and Python  
for the other pieces.

**Thousands of engineers** work on the code, at the "head"  
of a single tree comprising all the software.



The goals of the Go project were to **eliminate the slowness and clumsiness** of software development at Google, and thereby to make the process **more productive and scalable**. The language was designed by and for people who write—and read and debug and maintain—**large software systems**.

2011 We started with Ruby on Rails, quickly build first version.  
At the end of 2012, we had 200 API servers which serve 3000 requests per second for 60,000 mobile apps.

When our API traffic started growing faster, we started having to rapidly spin up more database machines.

The “one process per request” started to fall apart.



After rewriting the EventMachine push backend to Go we went from 250k connections per node to 1.5 million connections per node.

The time it takes to run our full integration test suite dropped from 25 minutes to 2 minutes.

The time to do a full API server deploy with rolling restarts dropped from 30 minutes to 3 minutes.

“The hardest part of the rewrite was dealing with all the undocumented behaviors and magical mystery bits that you get with Rails middleware.”

“We love Go. We've found it really fast to deploy, really easy to instrument, really lightweight and inexpensive in terms of resources.”

We need to scale a company  
from the early days of 5 engineers  
to 200+ engineers as the business grows.



## [Scala at Gravity]

Several of us started investigating and were able to track the source of the issue.

The only problem was we **had no idea what the code was doing** at first.

We came across a strange symbol we hadn't seen in our projects before. The spaceship operator `<|*|>`. Someone said out loud “**what the hell is that?**”.



# CROWDSTRIKE 2015

```
1 import scalaz._
2 import scalaz.std.list._
3 import scalaz.syntax.monad._
4 import scalaz.syntax.monoid._
5 import scalaz.syntax.traverse.{ToFunctorOps => _, _}
6
7 class Foo[F[+_] : Monad, A, B](val execute: Foo.Request[A] => F[B], val joins: Foo.Re
8
9 def bar: Foo[({type l[+a]=WriterT[F, Log[A, B], a]})#l, A, B] = {
10     type TraceW[FF[+_], +AA] = WriterT[FF, Log[A, B], AA]
11     def execute(request: Request[A]): WriterT[F, Log[A, B], B] =
12         self.execute(request).liftM[TraceW] :++>> (repr => List(request -> request.resp
13     ----- REDACTED -----
```



While you can have very high performing small teams going with Scala, **trying to grow and engineering organization > 50** is an uphill battle.

One of Go's reasons for existence is to **make developers more productive**. [...] New developers we've hired are ramped up in weeks vs months.



“Wow, I read through that [Go] library once and I knew **exactly what it was doing**,  
I’ve read the **Scala** version of that library four times and I still **have no idea what it does**,  
I can see why you guys like it so much”



About a year ago, we decided to **migrate our performance-critical backends** from Python to Go to leverage better concurrency support and faster execution speed.

This was a massive effort—around 200,000 lines of Go code—undertaken by a small team of engineers. At this point, we have **successfully moved major parts of our infrastructure to Go.**

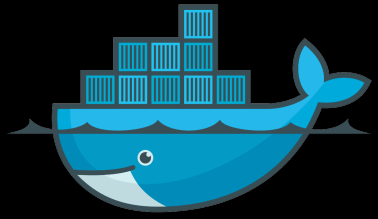


CLOUDFLARE®

2012

Go's library is **extensive** and **easy** to work with.

Go generates a **single executable** that can be distributed to our clients. There's no complex dependency chain or layout of shared libraries to worry about.



2013

1. Static compilation
2. It's not C++, not Python, not Ruby, not Java
3. It's has good async, low-level interface, extensive library
4. “go doc”, “go get”, “go fmt”, “go test”, “go run”
5. Multi-arch build



Basecamp 2015

“Go feels perfect for Ops work.

The error handling seems to fit so naturally into the way I want to write systems software.

Deployment is really simple too, where I'd have to think about how to package up deps and configure Ruby versions I can now just push an updated binary.”

When we launched the CoreOS project we knew from the very beginning that **everything we built would be written in Go**.

This was not to make a fashion statement, but rather Go happened to be **the perfect platform for reaching our goals** – to build products that make distributed computing as easy as installing a Linux distro.

Go is amazingly stable and awesome.  
I can't begin to list why everything about it is just great.

# TechTalk #3

Thank you

Vũ Nguyễn  
vu.nguyen@will.vn