TechTalk #3

What do people say when they switch to Go?

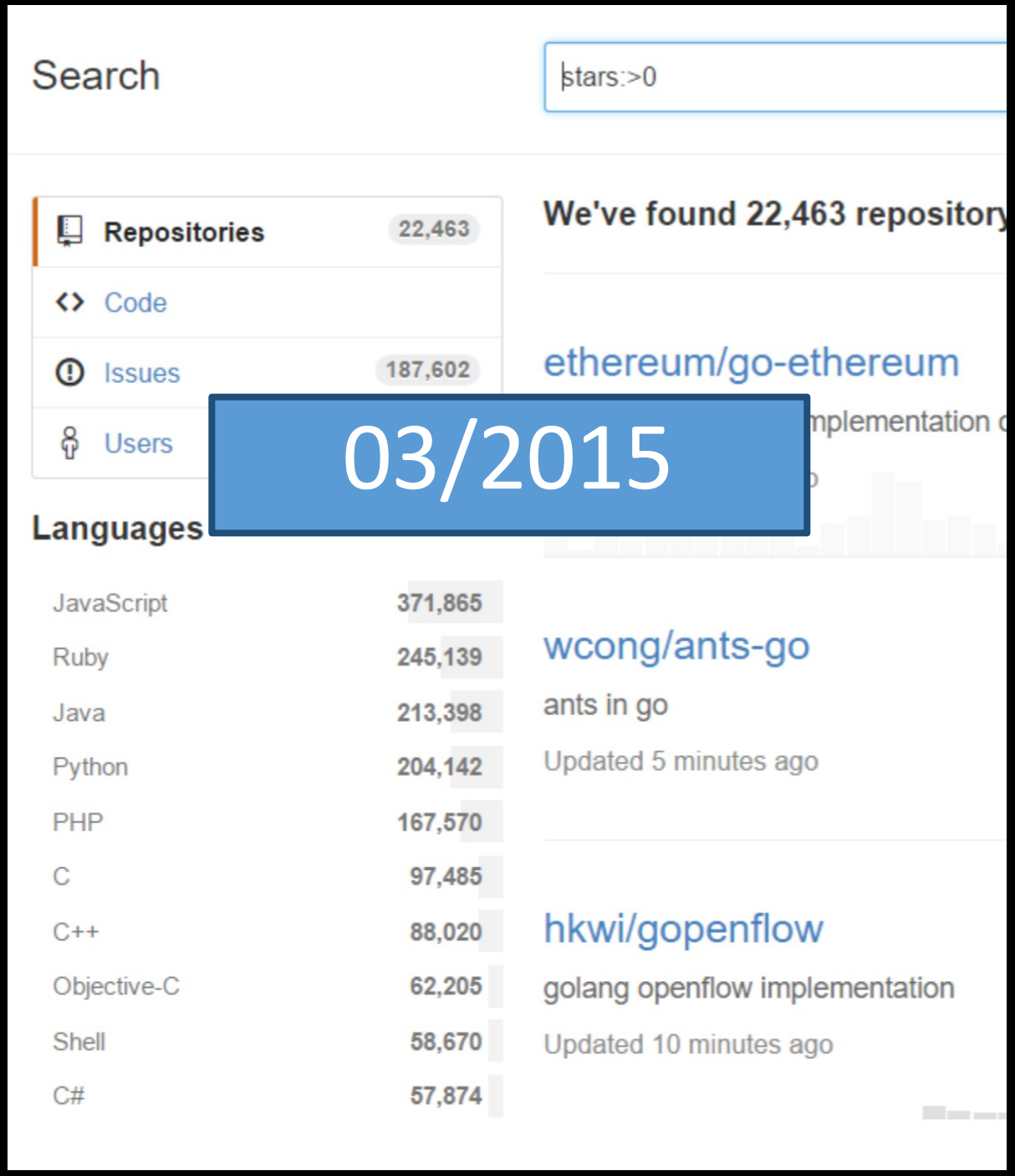
Oliver N.

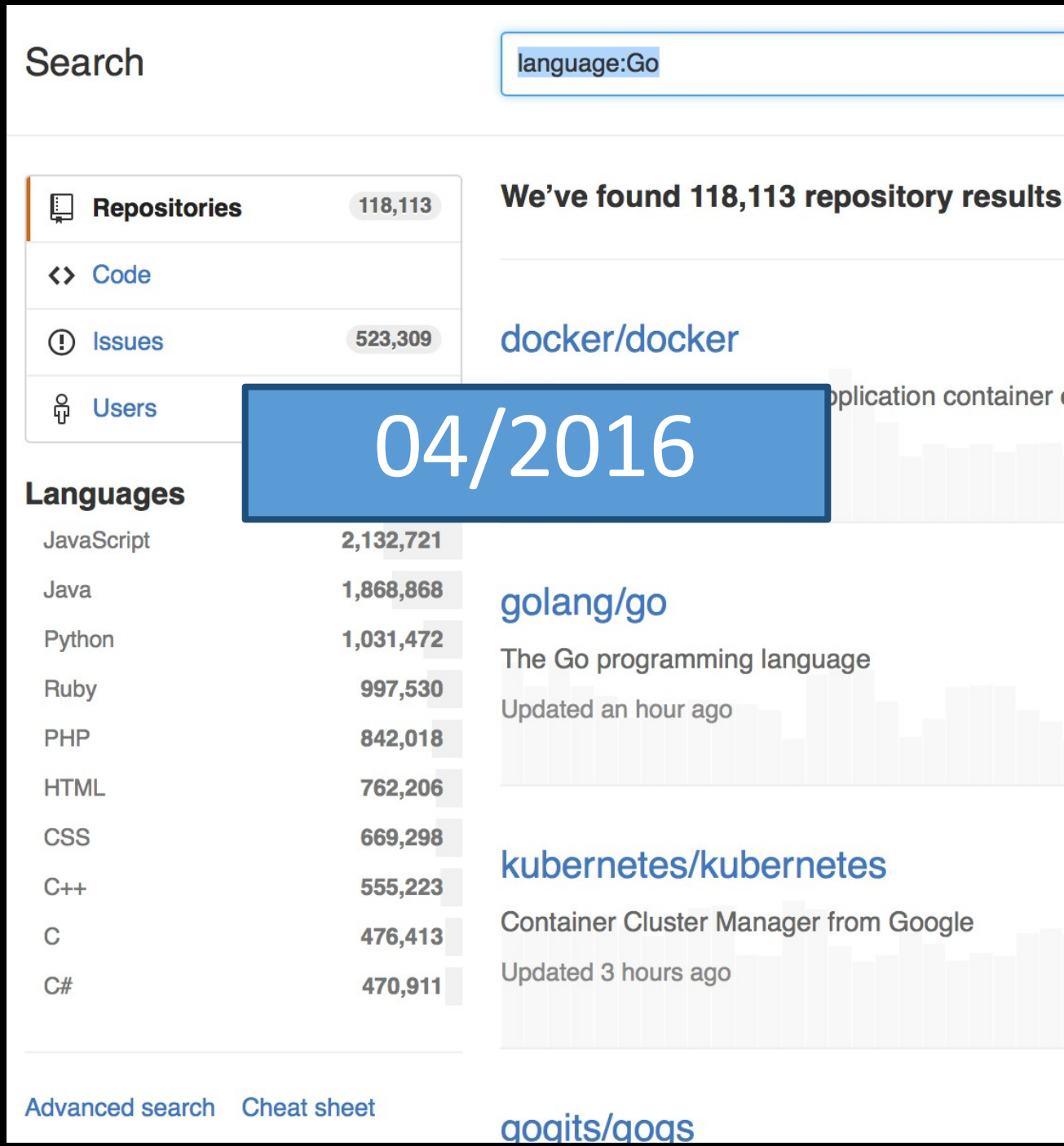
Software Engineer

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
stars:>0
Repositories 22,463
Code
Issues 187,602
Users
03/2015
We've found 22,463 repository
ethereum/go-ethereum
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
language:Go
Repositories 118,113
Code
Issues 523,309
Users
04/2016
We've found 118,113 repository results
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
Advanced search
Cheat sheet
docker/docker
golang/go
The Go programming language
Updated an hour ago
kubernetes/kubernetes
Container Cluster Manager from Google
Updated 3 hours ago
gogits/gogs

Who are using Go?

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

https://github.com/golang/go/wiki/GoUsers

Google 2012

Google has big softwares and big problems.

https://talks.golang.org/2012/splash.article

Google 2012

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.

https://talks.golang.org/2012/splash.article

Google 2012

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.

https://talks.golang.org/2012/splash.article

Parse 2015

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.

http://blog.parse.com/learn/how-we-moved-our-api-from-ruby-to-go-and-saved-our-sanity/

Parse 2015

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.

http://blog.parse.com/learn/how-we-moved-our-api-from-ruby-to-go-and-saved-our-sanity/

Parse 2015

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

http://blog.parse.com/learn/how-we-moved-our-api-from-ruby-to-go-and-saved-our-sanity/

2015

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

http://blog.parse.com/learn/how-we-moved-our-api-from-ruby-to-go-and-saved-our-sanity/

CROWDSTRIKE 2015

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

http://jimplush.com/talk/2015/12/19/moving-a-team-from-scala-to-golang/

CROWDSTRIKE 2015

[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?”.

http://jimplush.com/talk/2015/12/19/moving-a-team-from-scala-to-golang/

CROWDSTRIKE 2015

```scala
import scalaz._
import scalaz.std.list._
import scalaz.syntax.monad._
import scalaz.syntax.monoid._
import scalaz.syntax.traverse.{ToFunctorOps => _, _}

class Foo[F[+_] : Monad, A, B](val execute: Foo.Request[A] => F[B], val joins: Foo.Re

def bar: Foo[({type l[+a]=WriterT[F, Log[A, B], a]})#l, A, B] = {
    type TraceW[FF[+_], +AA] = WriterT[FF, Log[A, B], AA]
    def execute(request: Request[A]): WriterT[F, Log[A, B], B] =
      self.execute(request).liftM[TraceW] :++>> (repr => List(request -> request.resp
  ----- REDACTED -------
```

http://jimplush.com/talk/2015/12/19/moving-a-team-from-scala-to-golang/

CROWDSTRIKE 2015

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.

http://jimplush.com/talk/2015/12/19/moving-a-team-from-scala-to-golang/

CROWDSTRIKE 2015

"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"

http://jimplush.com/talk/2015/12/19/moving-a-team-from-scala-to-golang/

2014

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.

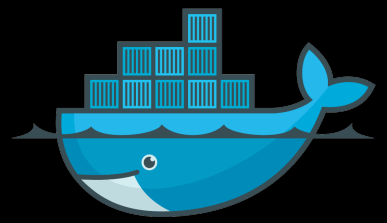
https://blogs.dropbox.com/tech/2014/07/open-sourcing-our-go-libraries/

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.

https://blog.cloudflare.com/go-at-cloudflare/



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

http://www.slideshare.net/jpetazzo/docker-and-go-why-did-we-decide-to-write-docker-in-go

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."

https://signalvnoise.com/posts/3897-go-at-basecamp

CoreOS 2014

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.

https://blog.gopheracademy.com/birthday-bash-2014/go-at-coreos/

2012

Go is amazingly stable and awesome.

I can't begin to list why everything about it is just great.

https://www.quora.com/Why-did-Koding-switch-from-Node-js-to-Go

TechTalk #3

Thank you!

Oliver N.

Software Engineer