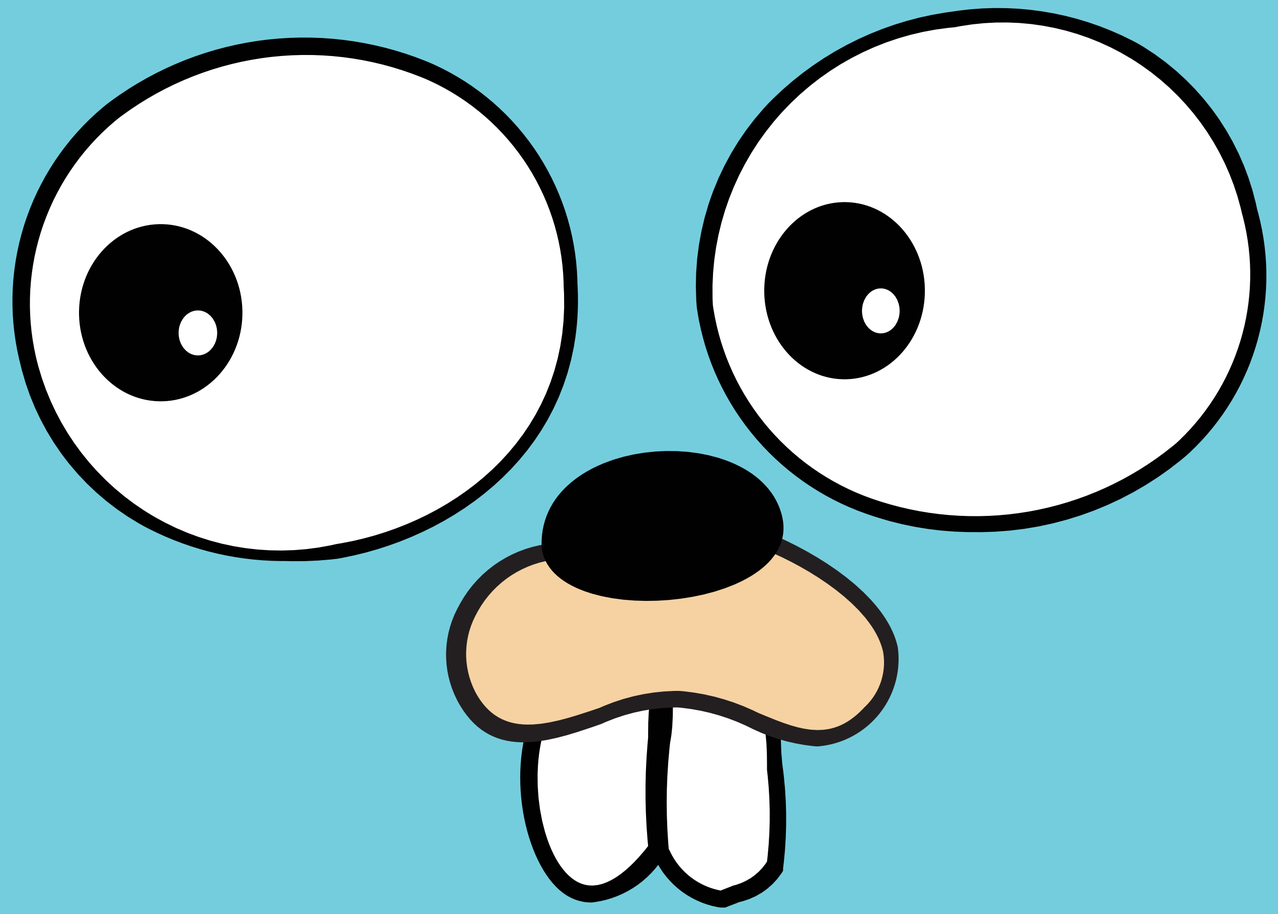


we are frank!

open source integration specialists

Presentation

Go & a little bit of NLX

Stijn van der Kolk

Time for a little history lesson

Robert Griesemer, Rob Pike, Ken Thompson - creators (Google)

- 2007
- Wanted to improve programming productivity
- Addressing criticism of other languages used at Google but keeping useful stuff:
  - Static typing and run-time efficiency (C)
  - Readability and usability (JS)
  - High-performance networking and multiprocessing
- C--
- Publicly 2009
- 1.0 - March 2012

https://go.dev/talks/2012/splash.article

Why Go?

- It tries to eliminate slowness
- It tries to eliminate clumsiness
- It tries to improve productiveness
- It should be more maintainable in scale
- Simplicity

"In short, development at Google is **big**, can be **slow**, and is often **clumsy**. But it is **effective**."

Designed **by** and **for** people who write, read, debug and, maintain large software systems

– Rob Pike, Creator of Golang –

https://go.dev/talks/2012/splash.article

Why Go at WeAreFrank?

- NLX, more on that later!

What is Go?

Go is a compiled, concurrent, garbage-collected, statically typed language developed at Google

But Go is also...

A tool for managing the Go source code, with tools like build, run, test, install and more!

```
Usage:

        go <command> [arguments]

The main tools are:

        build         compile packages and dependencies
        clean         remove object files
        env           print Go environment information
        run           compile and run Go program
        test          test packages

Other tools:
        fix, fmt, get, install, list, tool, version, vet
```

Is it widely used?

Yes, to name some big companies:

- Google
- Dropbox
- YouTube
- Cloudflare
- Apple
- Shopify
- DigitalOcean
- Uber

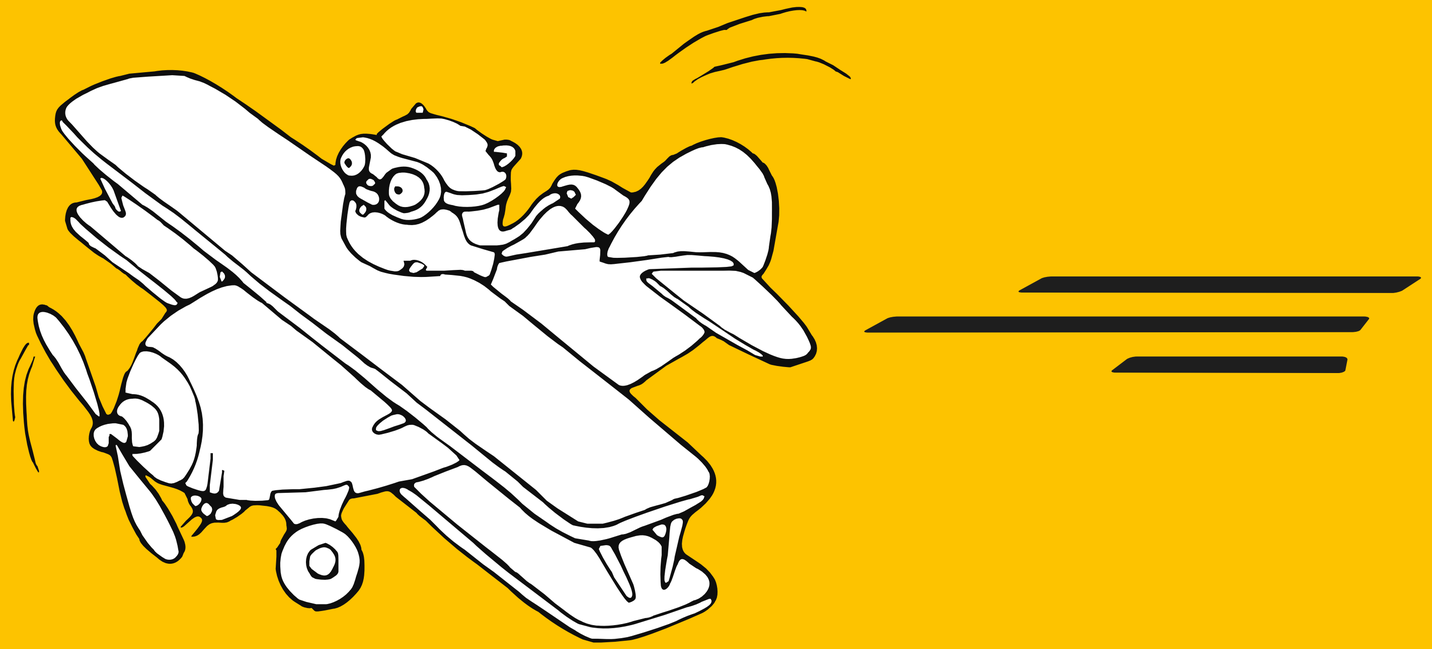
What will you see in Go?

- Object Oriented without inheritance
- Statically typed
- Cross platform
- Great standard library
- Garbage collection
- Points, structures, interfaces
- Concurrent (goroutines)

~~What will you see in Go?~~

- Exception Handling
- Inheritance
- Generics
- Ternary Operators
- Method overloading

Let’s look into some simple code



The famous Hello World!

```go
package hello_world

// HelloWorld greets the world.
func HelloWorld() string {
    return "Hello, World!"
}
```

Let's step through it

```go
package hello_world

import (
  "fmt"
)

func HelloWorld() {
  return fmt.Println("Hello, World!")
}
```

Time for a little comparison

```go
package hello_world

import (
  "fmt"
)

func HelloWorld() {
  return fmt.Println("Hello, World!")
}
```

```java
class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

```python
def hello_world():
  print("Hello, World!")
```

Speeeed

Benchmarks Game

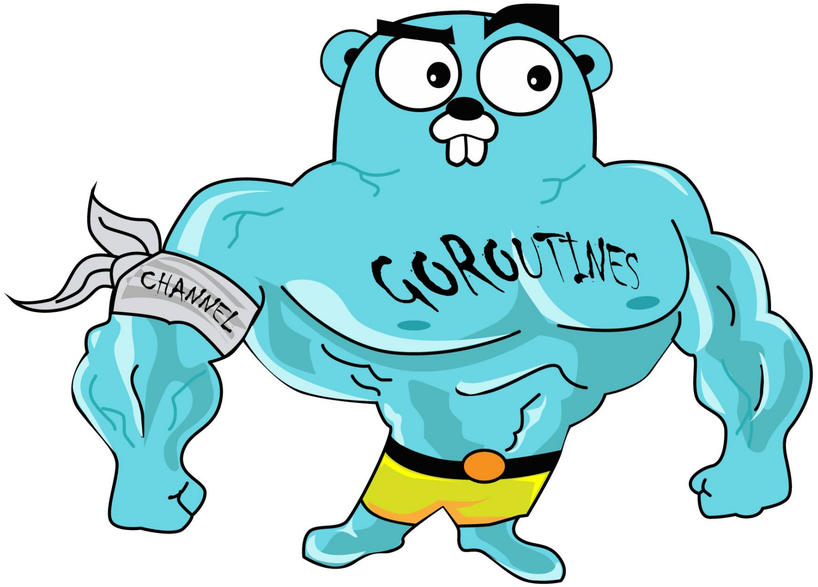
- Go vs Java
- Go vs Python

https://salsa.debian.org/benchmarksgame-team/benchmarksgame
https://benchmarksgame-team.pages.debian.net/benchmarksgame/fastest/go-python3
https://benchmarksgame-team.pages.debian.net/benchmarksgame/fastest/go

Let's do a quick dive in some more things Go

Goroutines!

"go"
Lightweight thread managed by Go runtime.



A simple goroutine

```go
package main

import (
    "fmt"
    "time"
)
func say(s string) {
  for i := 0; i < 5; i++ {
    time. Sleep(100 * time.Millisecond)
    fmt.Println(s)
  }
}

func main() {
    go say("world")
    say("hello")
}
```

```
$ go run goroutines

hello
world
hello
world
hello
world
hello
world
hello
world
```

NLX

What is NLX?

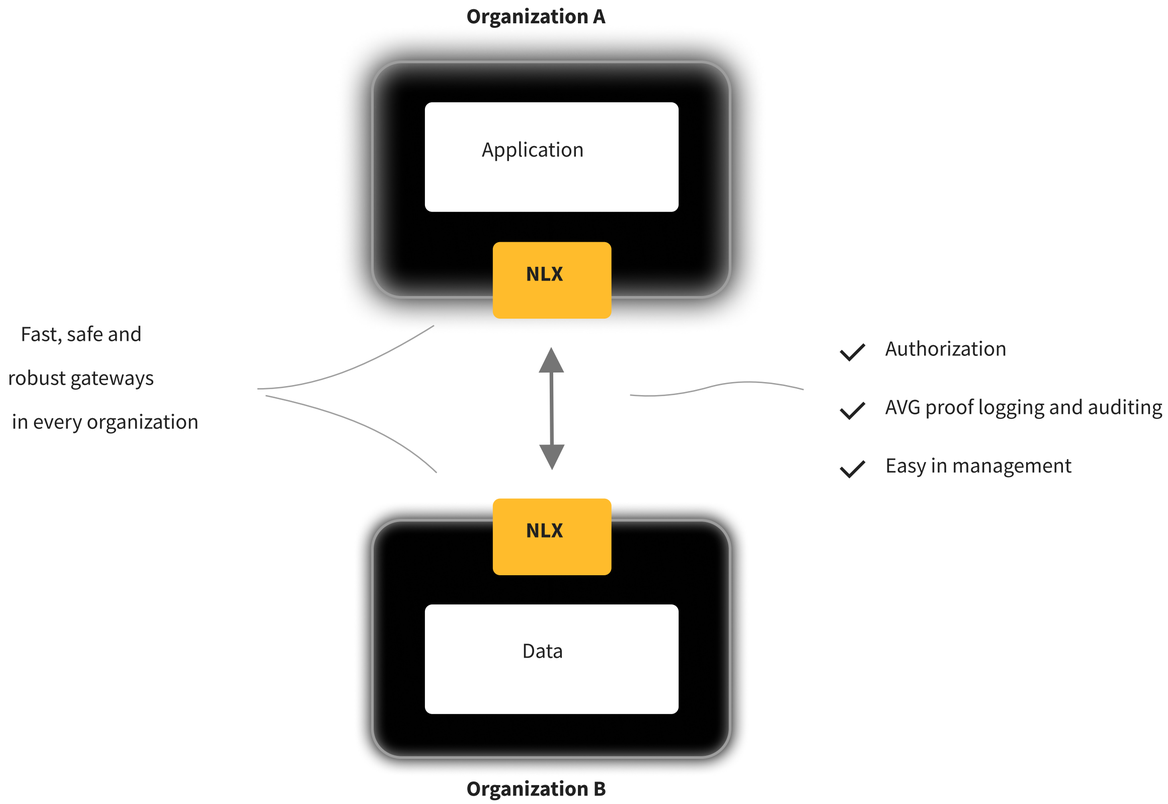
AVG-proof way to transfer data between and inside of (government) organizations

Why NLX?

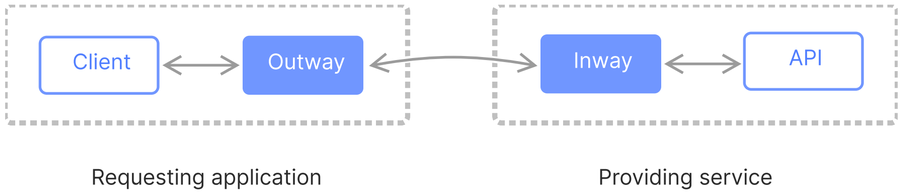
- It's secure!
- It's fast!
- It's robust
- Easier to comply with privacy rules
- More efficient
- Contributes to an open market

How does NLX work?

Every NLX participant has their own NLX gateway within their organization. This gateway makes a direct connection to the NLX gateway of the other organization as soon as necessary. Over this connection, data is retrieved securely as if the data were local.



- Authorization
- AVG proof logging and auditing
- Easy in management



Questions?