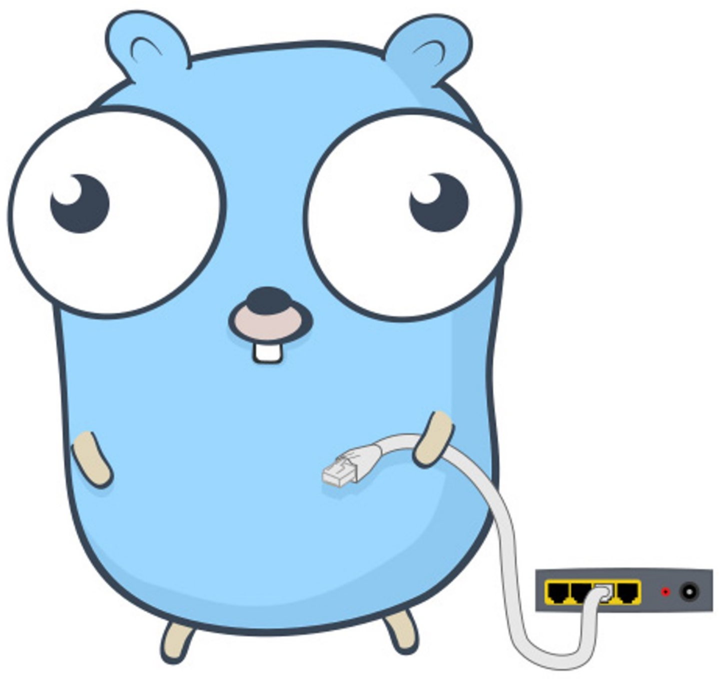
Golang for network engineers

NANOG 90

2-13-2024

Agenda

- Intro
- Go language
- Go ecosystem
- Demos

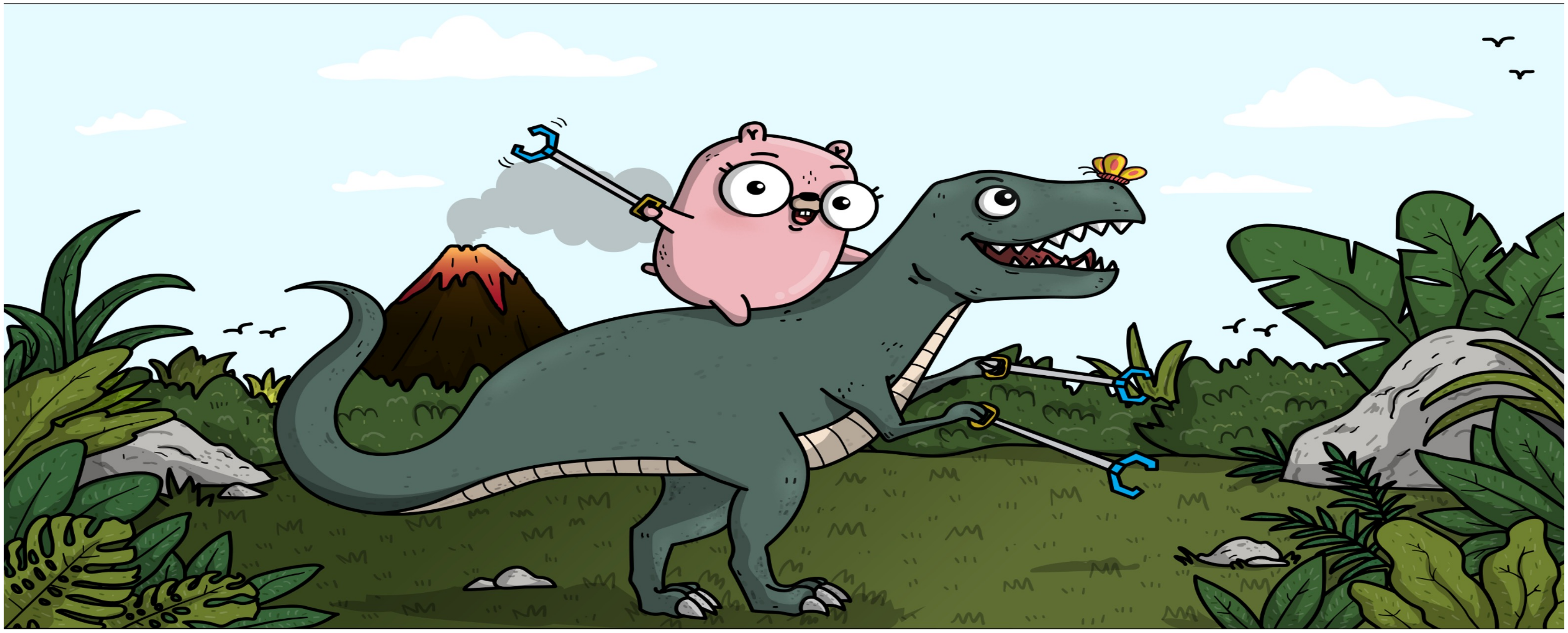


Demo Repo
https://github.com/burnyd/nanog-go-intro

Daniel Hertzberg
Arista Networks

NANOG™

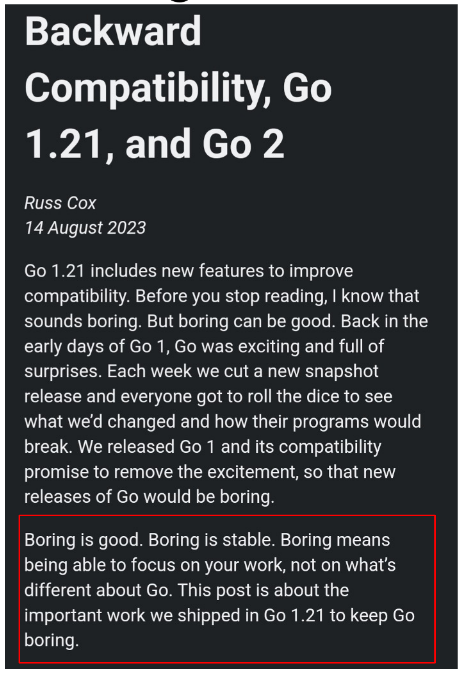
My Go Story



NANOG™

Go what is it?

- A **SIMPLE**, typed, **compiled** programming language with **easy readability**.
- **Speed** && **Efficient**.
- **Multiple Architectures(arm,x86,darwin)**
- Developed in 2009 first main release by Google. Mainly because C was too difficult.
- **Concurrency** as a first class citizen.
- Honestly, it's boring but just works! The founders said this not me!



Backward Compatibility, Go 1.21, and Go 2

Russ Cox
14 August 2023

Go 1.21 includes new features to improve compatibility. Before you stop reading, I know that sounds boring. But boring can be good. Back in the early days of Go 1, Go was exciting and full of surprises. Each week we cut a new snapshot release and everyone got to roll the dice to see what we'd changed and how their programs would break. We released Go 1 and its compatibility promise to remove the excitement, so that new releases of Go would be boring.

Boring is good. Boring is stable. Boring means being able to focus on your work, not on what's different about Go. This post is about the important work we shipped in Go 1.21 to keep Go boring.

Go Features

- Better package management **go.mod/go.sum**
- Concurrency && **go routines**.
- Formatting && white spacing.
- Built in unit testing.
- **pkg.go.dev**.
- **Error values** are values.(Also super simple)
- **Generics** support.
- vendoring packages.

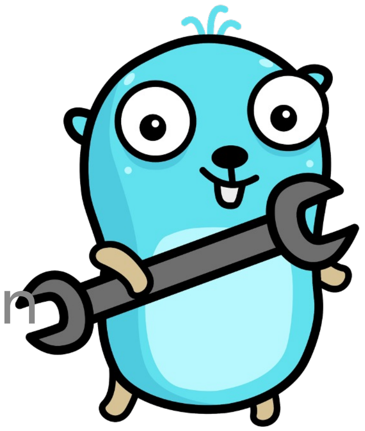
NANOG™

Go use cases in networking

Streaming telemetry use cases - There is a shift currently to remove SNMP from networks. Go is well suited given its relation to gRPC and the gNMI service to stream to different telemetry stacks.

Infrastructure provisioning - Most infrastructure provisioning tooling is written in Go (Terraform, Pulumi and crossplane) due to static typing and cloud API's.

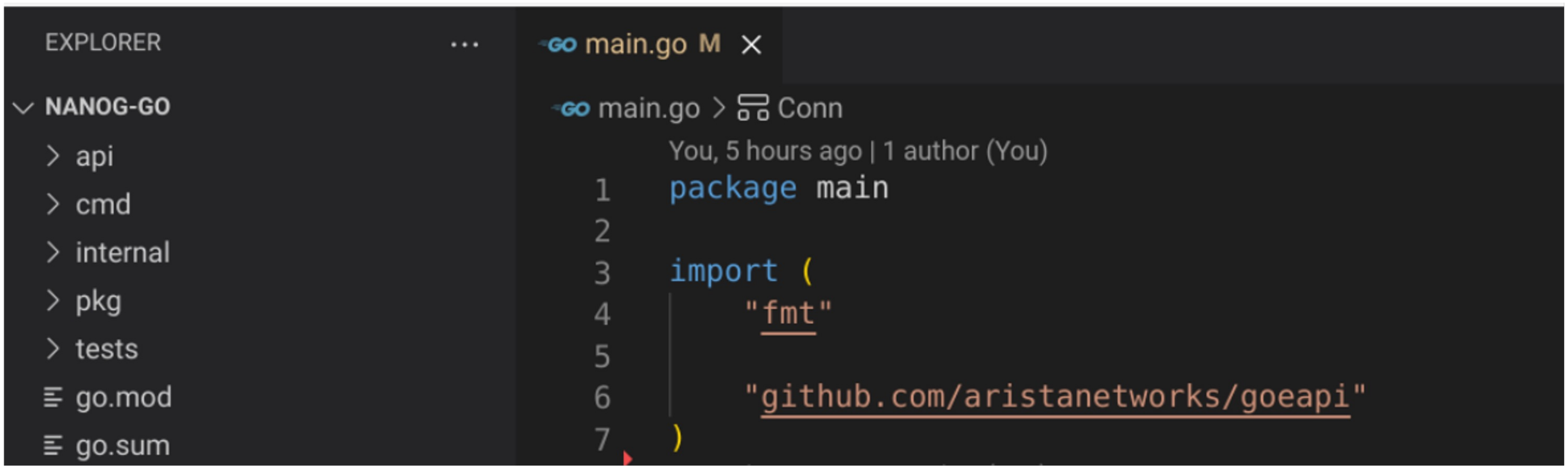
Network Config Management/Generation - Not seeing many use cases here. There are a few frameworks out there but not too popular. As openconfig projects grow it is expected this will gain traction.



NANOG™

Go File structure

go mod init github.com/burnyd/nanog-90-golang &&
mkdir -p cmd pkg internal api tests

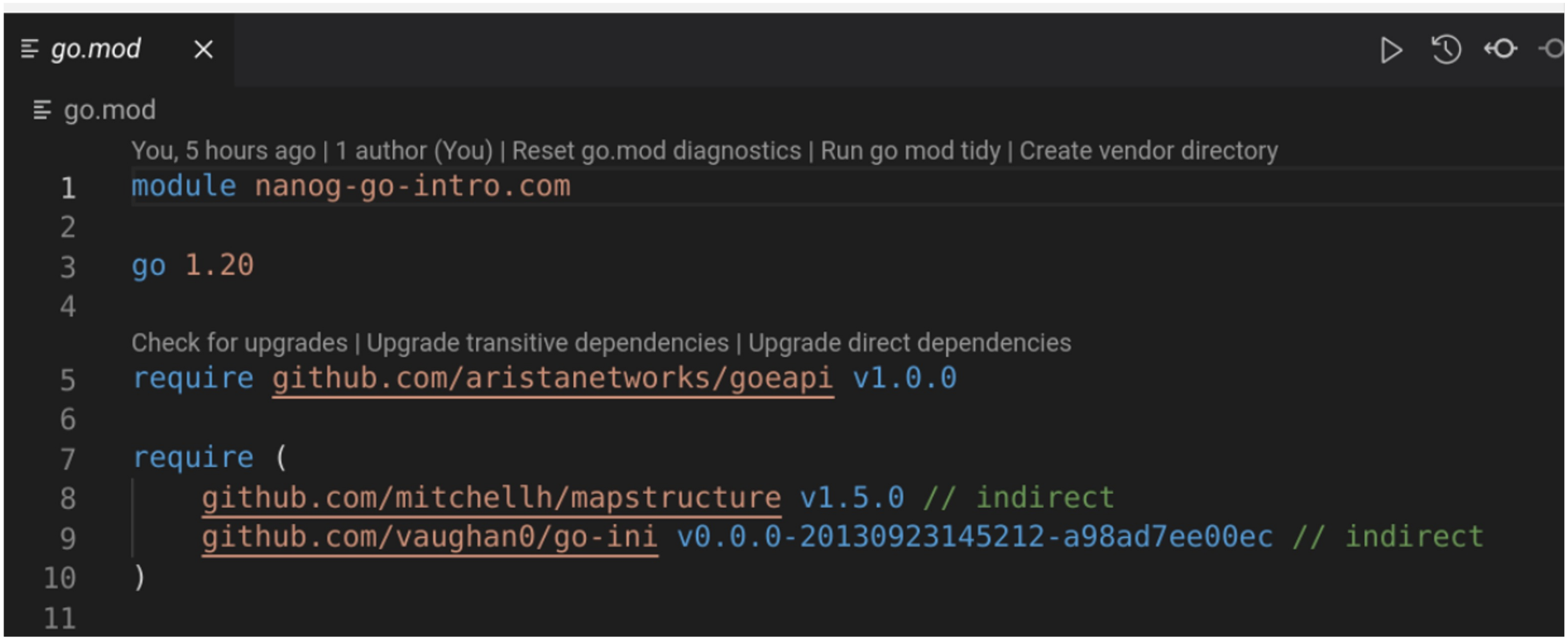


```
api
cmd
internal
pkg
tests
go.mod
go.sum
```

```go
package main

import (
	"fmt"

	"github.com/aristanetworks/goeapi"
)
```

NANOG™

Package management


go.mod
go.mod
You, 5 hours ago | 1 author (You) | Reset go.mod diagnostics | Run go mod tidy | Create vendor directory
Check for upgrades | Upgrade transitive dependencies | Upgrade direct dependencies

```
module nanog-go-intro.com

go 1.20

require github.com/aristanetworks/goeapi v1.0.0

require (
	github.com/mitchellh/mapstructure v1.5.0 // indirect
	github.com/vaughan0/go-ini v0.0.0-20130923145212-a98ad7ee00ec // indirect
)

```

NANOG™

Building with Go

Compiling with go

```
GOOS=linux GOARCH=amd64 go build -o myapp-linux-amd64
GOOS=windows GOARCH=amd64 go build -o myapp-windows.exe
GOOS=linux GOARCH=arm go build -o myapp-linux-arm
GOOS=linux GOARCH=arm64 go build -o myapp-onm2
GOOS=darwin GOARCH=amd64 go build -o myapp-linux-darwin-mac
```

```
-rwxrwxr-x 1 burnyd burnyd 7.4M Jan  4 08:34 myapp-linux-amd64
-rwxrwxr-x 1 burnyd burnyd 7.2M Jan  4 08:35 myapp-linux-arm
-rwxrwxr-x 1 burnyd burnyd 7.2M Jan  4 08:36 myapp-linux-darwin-mac
-rwxrwxr-x 1 burnyd burnyd 7.0M Jan  4 08:35 myapp-onm2
-rwxrwxr-x 1 burnyd burnyd 7.4M Jan  4 08:35 myapp-windows.exe
```

NANOG™

Running && installing with Go

Go allows for a program to be ran on a system with go installed symply with the **go run** syntax. Mainly meant for testing.

Running a go program

```
go run main.go
```

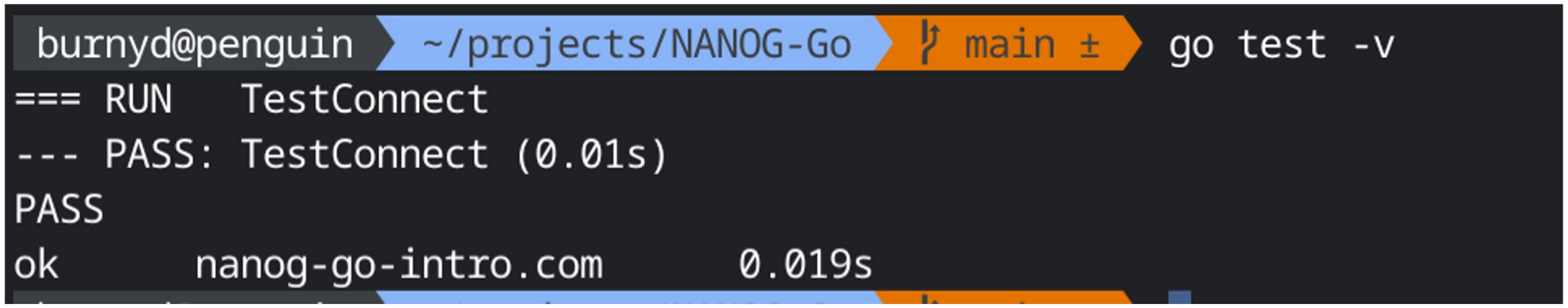
Go also allows for a program to be installed on a system into what's called the **GOBIN**. As long as the **$GOBIN** env var is set a package can be installed with the **go install flag**. To then run on the system.

Installing a go program

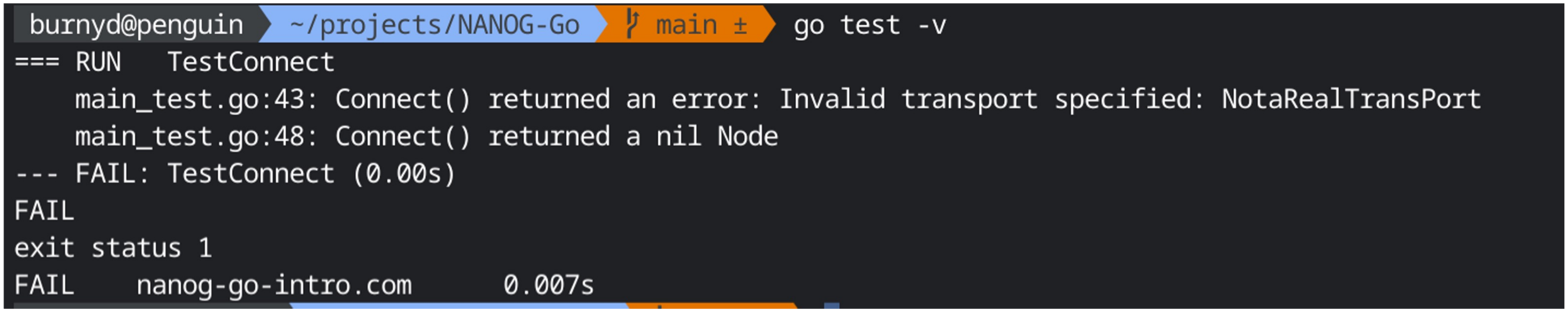
```
go install github.com/repo/awesomepackage
```

NANOG™

Unit tests



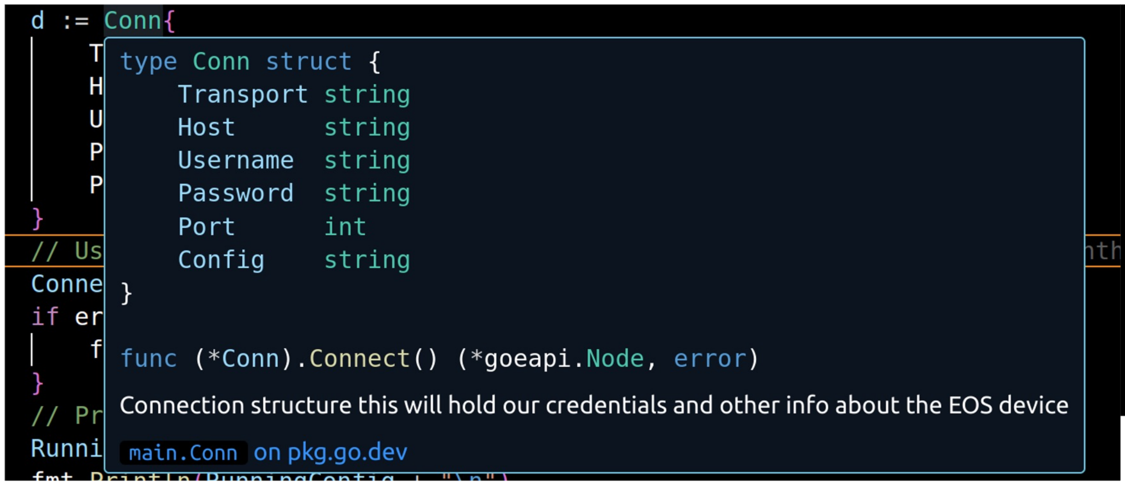
```
burnyd@penguin  ~/projects/NANOG-Go  ᚠ main ±  go test -v
=== RUN   TestConnect
--- PASS: TestConnect (0.01s)
PASS
ok      nanog-go-intro.com      0.019s
```



```
burnyd@penguin  ~/projects/NANOG-Go  ᚠ main ±  go test -v
=== RUN   TestConnect
    main_test.go:43: Connect() returned an error: Invalid transport specified: NotaRealTransPort
    main_test.go:48: Connect() returned a nil Node
--- FAIL: TestConnect (0.00s)
FAIL
exit status 1
FAIL    nanog-go-intro.com      0.007s
```

NANOG™

Go Typing

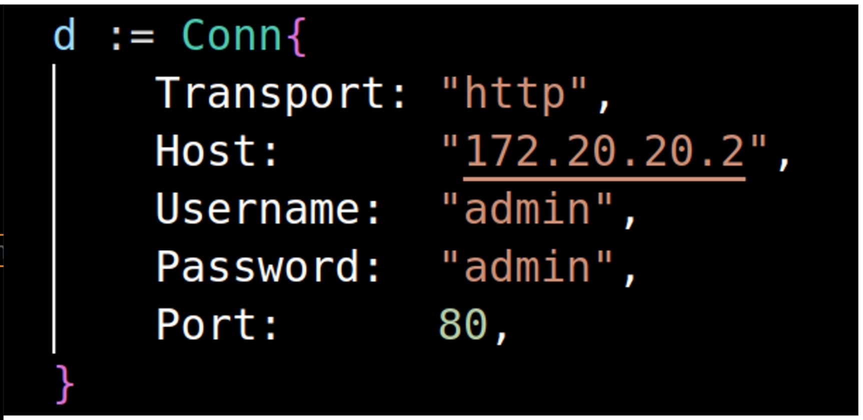


```
d := Conn{
    T
    H
    U
    P
    P
}
// Us
Conne
if er
    f
}
// Pr
Runni
```

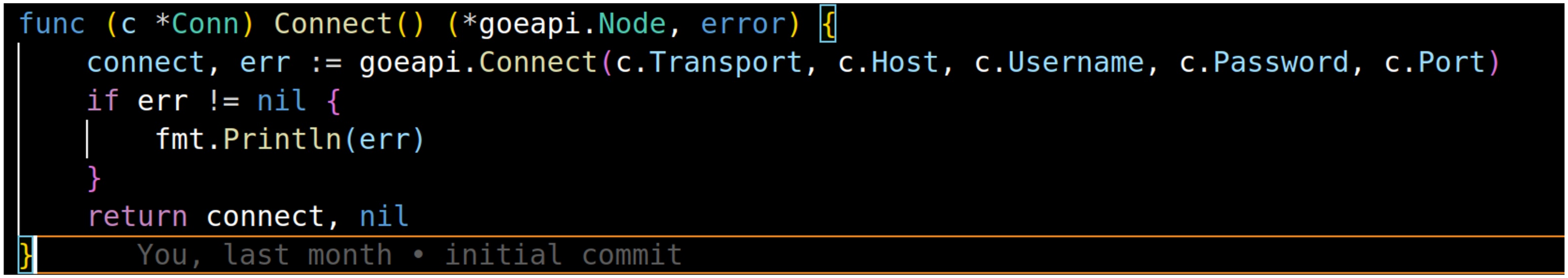
```
type Conn struct {
    Transport string
    Host      string
    Username  string
    Password  string
    Port      int
    Config    string
}

func (*Conn).Connect() (*goeapi.Node, error)
```

Connection structure this will hold our credentials and other info about the EOS device

`main.Conn` on pkg.go.dev



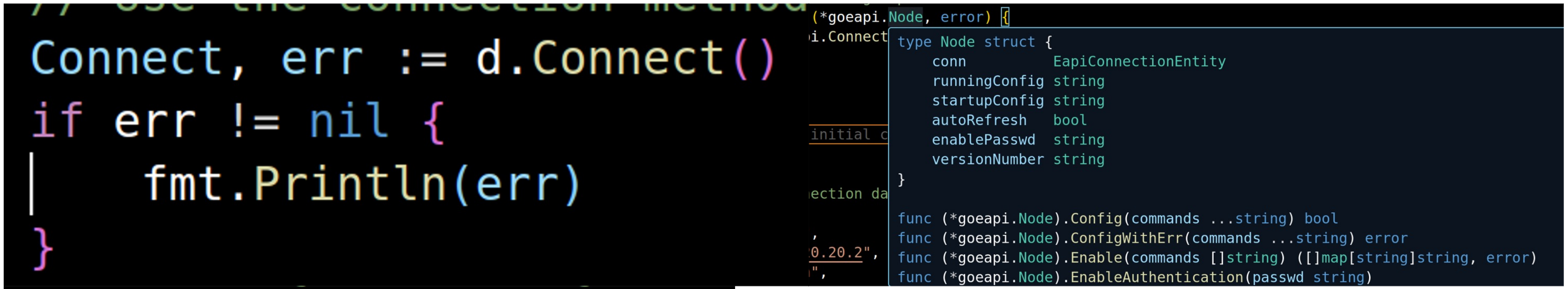
```
d := Conn{
    Transport: "http",
    Host:      "172.20.20.2",
    Username:  "admin",
    Password:  "admin",
    Port:      80,
}
```



```
func (c *Conn) Connect() (*goeapi.Node, error) {
    connect, err := goeapi.Connect(c.Transport, c.Host, c.Username, c.Password, c.Port)
    if err != nil {
        fmt.Println(err)
    }
    return connect, nil
}     You, last month • initial commit
```

NANOG™

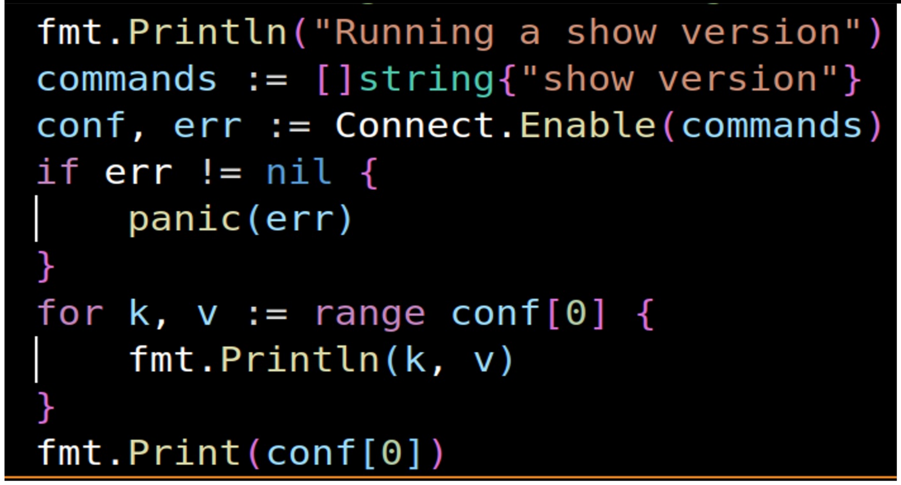
Go Typing



```
Connect, err := d.Connect()
if err != nil {
    fmt.Println(err)
}
```

```
(*goeapi.Node, error) {
i.Connect
initial c
ection da
0.20.2",
",
```

```
type Node struct {
    conn           EapiConnectionEntity
    runningConfig  string
    startupConfig  string
    autoRefresh    bool
    enablePasswd   string
    versionNumber  string
}

func (*goeapi.Node).Config(commands ...string) bool
func (*goeapi.Node).ConfigWithErr(commands ...string) error
func (*goeapi.Node).Enable(commands []string) ([]map[string]string, error)
func (*goeapi.Node).EnableAuthentication(passwd string)
```



```
fmt.Println("Running a show version")
commands := []string{"show version"}
conf, err := Connect.Enable(commands)
if err != nil {
    panic(err)
}
for k, v := range conf[0] {
    fmt.Println(k, v)
}
fmt.Print(conf[0])
```

NANOG™

error handling

NANOG™

Projects that use Go

HashiCorp
Terraform

CLOUD NATIVE
COMPUTING FOUNDATION

Grafana

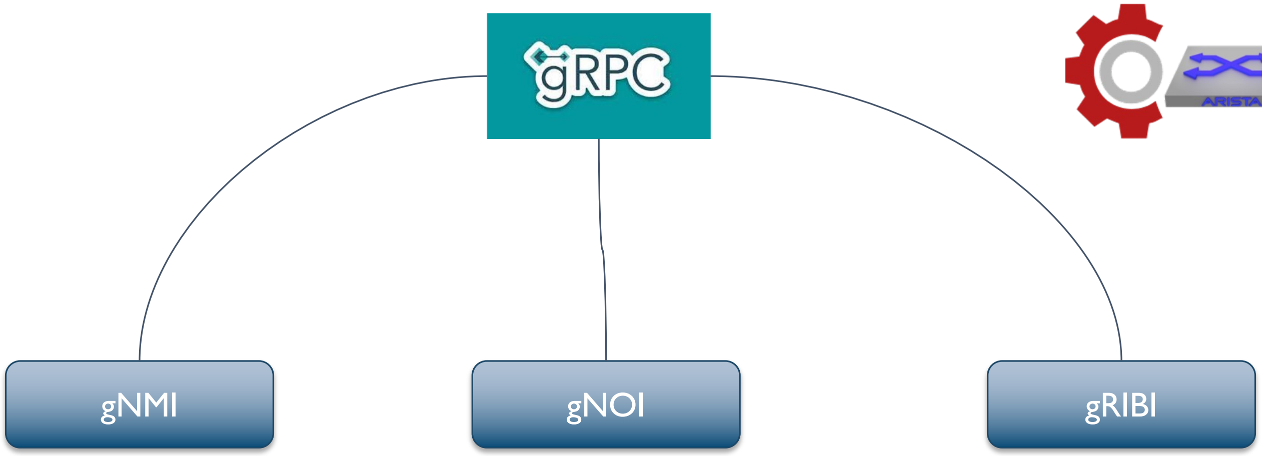
CONTAINERlab

gNMIc

NANOG™

OPENCONFIG

Go and Openconfig



Methods

- Capabilities
- Get
- Set
- Subscribe

Services

- bgp
- cert
- diag
- os
- ping
- traceroute
- healthz

Methods

- Modify
- Get
- Flush

Popular go network modules

Default	Openconfig	misc
net/http	ygot	goBGP
crypto/tls	ygNMI	netaddr
	gRIBI	prometheus
		gonetbox
		gopacket

NANOG™

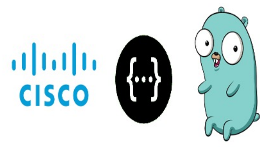
Go network integrations

Arista - Cloudvision resource gRPC apis

Equinix - Infrastructure go modules.

Juniper - Apstra go sdk

Cisco - ACI go client



Interact with a network device with go

NANOG™

Stream Network data with gNMI and go

NANOG™

Suggested go content

GO A Tour of Go

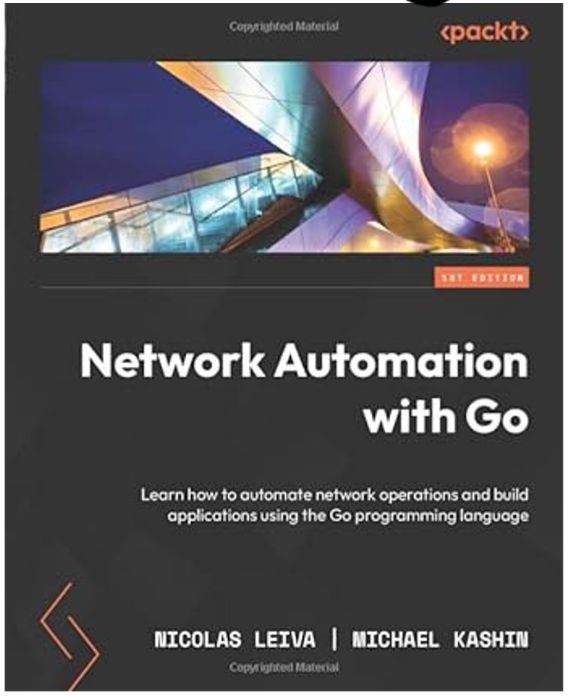
Hello, 世界

Welcome to a tour of the Go programming language.

Go by Example

Golang Weekly

A weekly newsletter about the Go programming language



Copyrighted Material

‹packt›

1ST EDITION

Network Automation with Go

Learn how to automate network operations and build applications using the Go programming language

NICOLAS LEIVA | MICHAEL KASHIN

Copyrighted Material

NANOG™

Thank you

NANOG™