A Scalable, Distributed Service Mesh in Go

Brian Ketelsen

Brian Ketelsen @bketelsen

- Author of upcoming Pragmatic Bookshelf book on Go
- Executive Vice President at Clarity Services, Inc
- Created RailsLiveCD
- Responsible for some of the biggest Rails installations you've NEVER HEARD OF

Skynet is a direct response to the pain of upgrading, managing and scaling a monolithic application.*

*I'mma let you finish Rails, you're still awesome for web sites

Pain Points Addressed

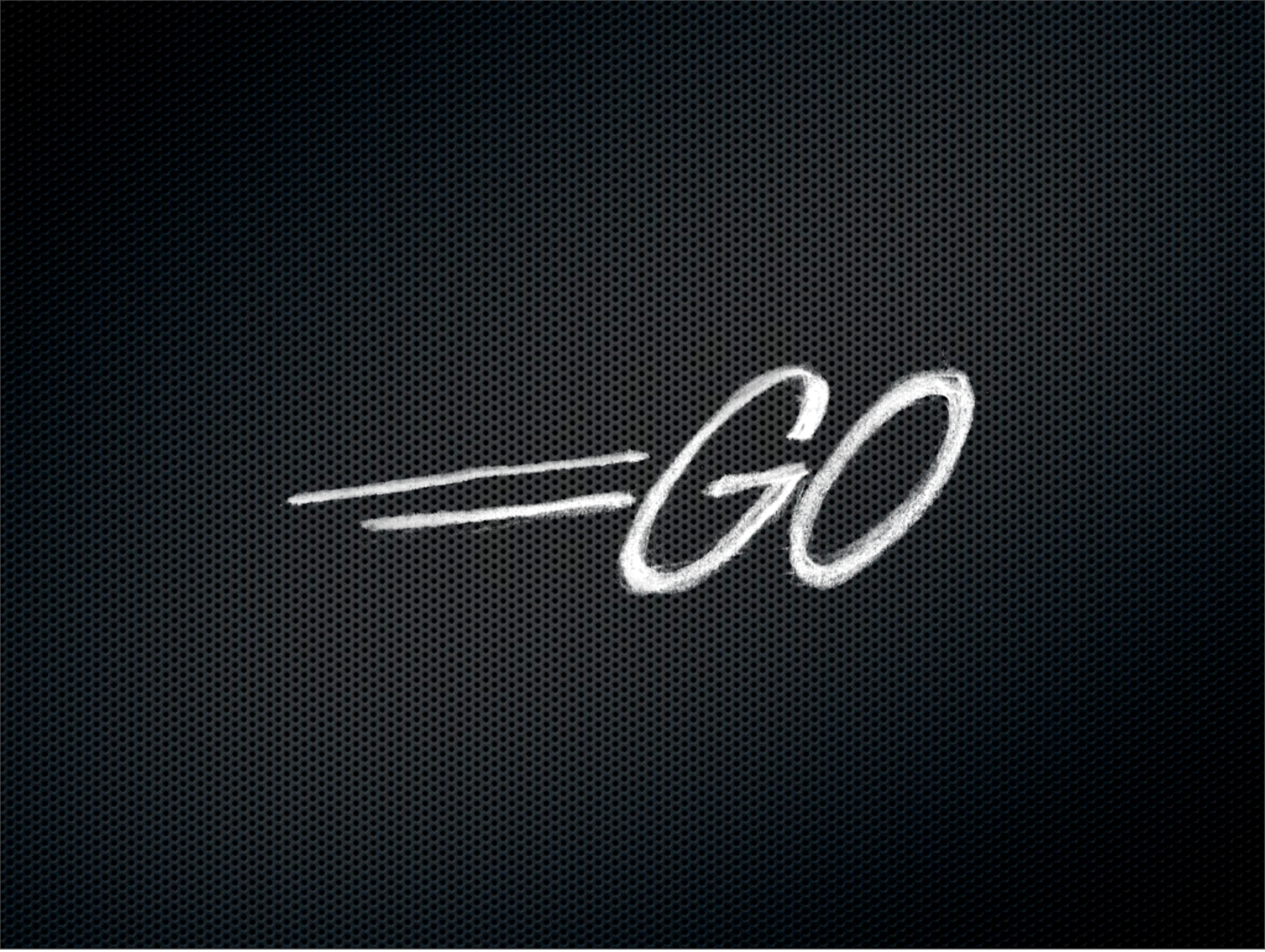
- Deploying entire app just to change small piece
- Dependency nightmare because of multiple developers working on monolithic codebase
- Monitoring pain - without New Relic you can’t see what your app is doing
- Deployment - JRuby helps a lot, but deploying Ruby still isn’t really great (Torquebox IS awesome, though)
- Configuration should be centralized

Skynet Is

- Distributed -- a mesh of processes. Upgrade a single component without redeploying the whole application
- Composable -- Each application is composed of smaller services linked by RPC calls operating on the same request/response
- Polyglot
- Self-Aware -- DNS-SD *soon*, Dead processes removed from configuration automatically by Reaper

Skynet is Not

- An Enterprise Service Bus
- The next Ruby on Rails
- Too Complicated
- Magic Scaling Sprinkles - you still have to optimize your application and hire smart people
- A new language that compiles to Javascript
- Too big for your problem



Skynet is written in Go

- Skynet was actually inspired by Go's features and solidified when Doozer appeared
- Compiled, type-safe, garbage collected
- Speed like C / Fun like Ruby
- Duck typing
- Unique "interface" system instead of inheritance

Go's Compiler is Fast

- Compile entire Go standard library in less than 20 seconds
- Typical projects compile in half a second
- Skynet compile:
  - real 0m0.157s
- Lightning fast compiles make compilation a non-issue

C inspired syntax

```
for k := 1; k < 100; k++ {
    fmt.Println("Iteration %d", k)
}
```

Type Inference

```
var count float64 // declared
count = 1.62 // then assigned
TOO MUCH TYPING
```

```
count := 1.62 // type inferred by compiler, declared
              // and assigned all at once
```

Awesome Concurrency

```
func pollTheServer(){
    for {
        if ServerIsBroken() {
            alertOnCallStaff()
        }
    }
}
go pollTheServer() // runs in a goroutine
```

Multiple Return Values

```
func getData(fk int)(data string, err os.Error){
    // get some data
return
}
if myData, err := getData(2); err != nil {
    // OH NO
}

// NO embedded secret error codes
// I'm looking at you C
```

Standard Library

- Internet Ready : websockets, http, rpc, json, xml, html, smtp, more...
- Good crypto support
- Hashes are first class data types
- ExpVar - my favorite little feature to export metadata from running application

External Libraries

- ‘goinstall’ retrieves and installs external libraries and commands
- Anything on Github, GoogleCode, Launchpad
- $> goinstall github.com/bketelsen/skynet/reaper


sky
net

Goal

- Create 1 to N atomic services that each do one thing
- Create “routes” that chain these services together to create an application
- Make some services run outside the response chain (asynchronously)
- Scale by running multiple nodes across multiple servers

More Goals

- Must be resilient
- Must be easy to monitor
- Must be friendly to multiple data centers or “cloud” computing

Skylib - the soul of Skynet

- Access to the registry
  - List of nodes and the services they provide
  - Add node
  - Remove node
  - Get RPC client for service

Skylib - Registry

- Doozer - original and still the best data store for Registry configuration...
- Didn’t run on OS X Lion, and building isn’t straightforward if you are running newer versions of Go

Skylib - Registry

- MongoDB
  - Not quite as resilient as Doozer
  - Still good clustering and fault tolerance
  - Good speed

Skylib - Registry

- Pluggable Registry is on the list of future enhancements to Skynet

Skylib

- Logging
  - Log levels for each Skynet node range from Debug to Error
  - Raise or lower the log level with USR1 and USR2 signals

Skylib

- Exported counters
- Default counters for requests, errors and goroutines
- available on http://server:port/debug/vars as JSON

Skynet Components

- Initiators
- Routers
- Services
- Data store (doozer, mongodb)
- Reaper - to take out the trash

Request/Responsse

- Created by initiator
- Sent to router, then to each service
- modifications along the way are passed along
- build your response sequentially

Smallest Skynet Application

Initiators

- Interface to the outside world
  - Web, file system, csv files, RPC, REST, STOMP
- Initiators are the “many” in a many to one relationship
  - Many ways to send the same message into Skynet

Initiators

FTP

Web

REST

JSON-RPC

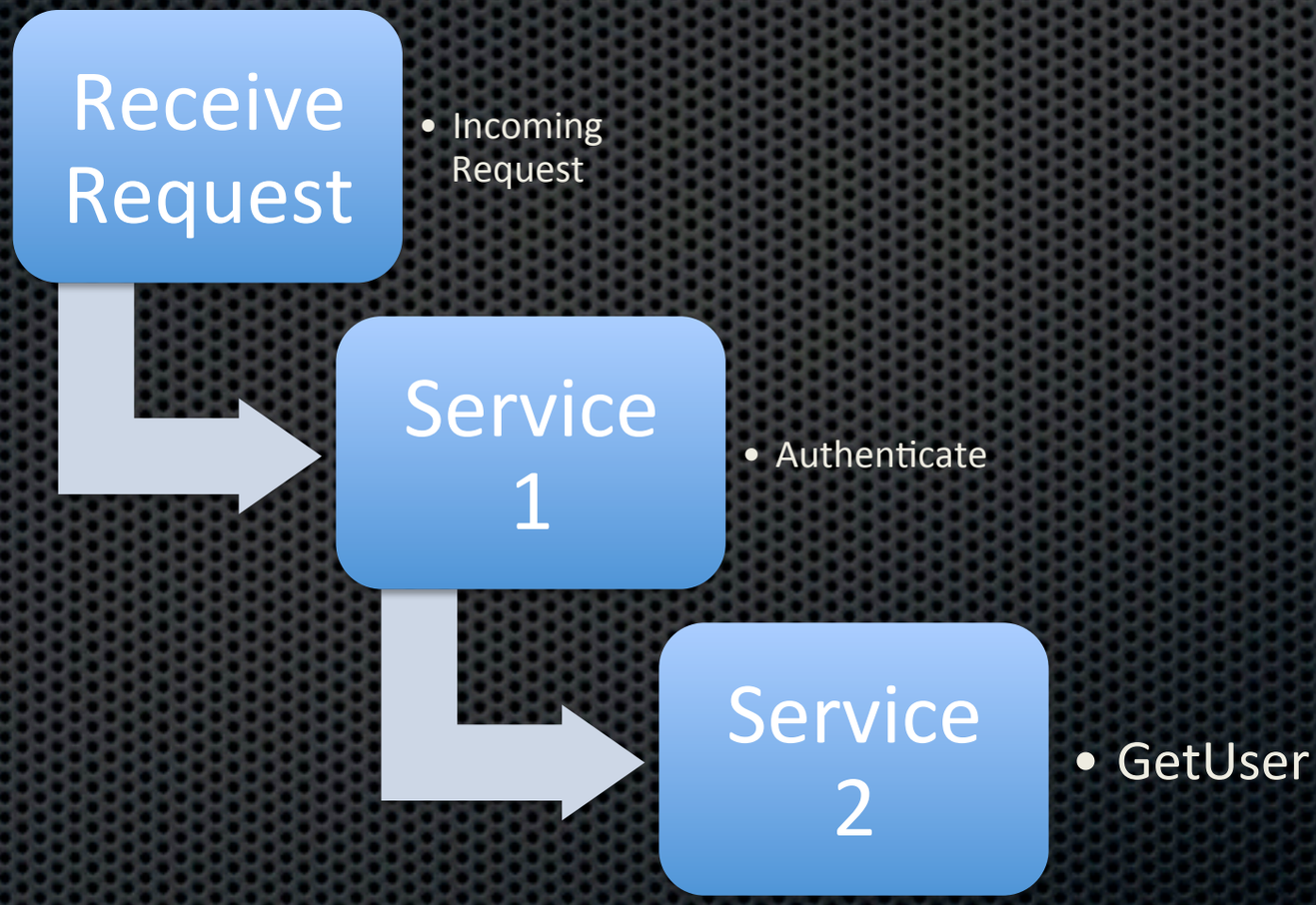
CSV

Router → Service(s)

Routers

- Receive requests
- Apply “route” - a list of services
- Call each service serially or out of band
  - serial calls are additive, data from each service available to next service
- Can retry services that fail
- Return response to initiator

Router


Receive Request
• Incoming Request
Service 1
• Authenticate
Service 2
• GetUser

Services

- Services for an application all have the same request/response signatures
- Can be run out of band (async)
  - Log requests
  - Notify other systems
  - Audit/Accounting
- Or in Series - assemble response piece by piece

Services


Authorization
Is User OK?
Retrieve Data
Get data from database
Assemble Response
Assemble data into required format

WHY?

- Why manage so many processes?

Services


Authorization
Is User Authorized?
Retrieve Data
Get data from database
Retrieve Data
Get MORE data from different database
Assemble Response
Assemble data into required format

Composition

- Add steps to your process without redeploying the entire application
- Deploy nodes without stopping or restarting the rest of the application

Composition

- To add another step to your application:
  - Create a new service
  - Deploy it
  - Change Router to add new service call
  - Deploy it

Learn from the past

- And borrow heavily from your friends

Skynet has:

- Rails-like generator
- Static request and response
- Metrics and monitoring built in
  - Go's expvars offer simple polling and json data output
- Send USR1 and USR2 to running process to raise or lower log level - get more info when you need it!

DEMO

SkyNe(x)t

What's ahead?

Future Additions

- Web tool to collect metrics from each node and display availability
- DNS-SD/zeroconf
- Polyglot tools - Ruby, Python, Java libs to allow Skynet initiators in other languages. Eventually support routers and initiators too
- Pluggable Datastores (doozer, MongoDB, Redis, Sql?)

Skynet is proudly Open

- Patches, Forks and Pull Requests are celebrated
- https://github.com/bketelsen/skynet
- skynet-dev@googlegroups.com
- http://www.brianketelsen.com
- @bketelsen

Questions?

Thank you