

Go: debugging and troubleshooting tips

Jack Lindamood
Engineer @ SignalFx
cep221@gmail.com

Target audience

- Thinking about deploying go
- Tend to write bugs
- Want to make troubleshooting easier



Logs

- Logging built in
- Stdout vs file
- Structured logging?
- Where do logs go?
 - Scribe
 - logstash
 - etc



Built in logging

Singleton optional pattern

```
import "log"
log.Printf("User %d", 123)
```

```
import "log"
var Logger *log.Logger
...
Logger.Printf("User %d",
123)
```

Singleton Optional Pattern

```
type Logger struct {  
    // ...  
}  
  
var std = New(...)  
  
func Println(s string) {  
    std.Println(s)  
}
```

Structured Logging

```
import "github.com/Sirupsen/logrus"  
logrus.WithField("userID", 123).Info("Request")
```

<https://github.com/inconshreveable/log15>
`log.Info("Request", "userID", 123)`

Why not both?

```
var Logger StdLogger = log.New(ioutil.Discard, "[Sarama] ", log.LstdFlags)

// StdLogger is used to log error messages.
type StdLogger interface {
    Print(v ...interface{})
    Printf(format string, v ...interface{})
    Println(v ...interface{})
}
```

Logging/defer pattern

```
func abc() {  
    log.Info("<- abc() ")  
    defer log.Info(">-> abc() ")  
    // ...  
}
```


PProf help

- Goroutines
- Heap
- CPU usage
- Running command line

PProf enabled

```
import _ "net/http/pprof"

go func() {
    log.Println(http.ListenAndServe("localhost:6060", nil))
}()

// ... OR ...

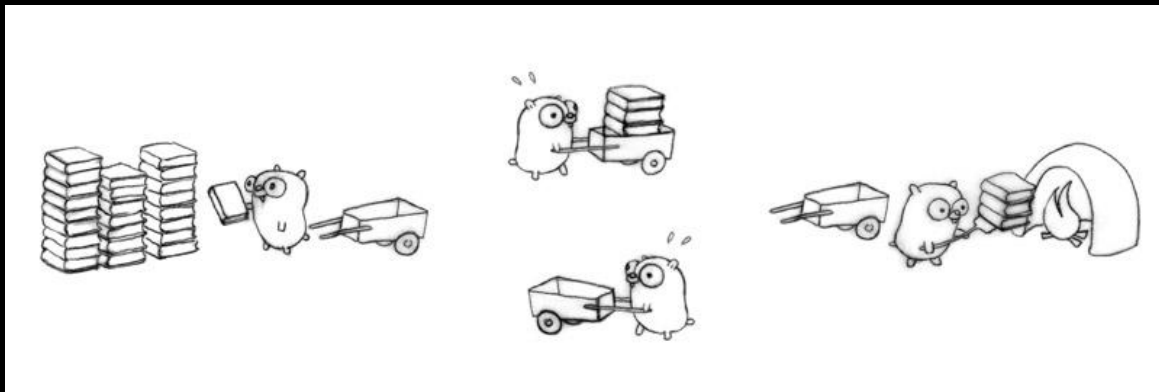
handler.PathPrefix("/debug/pprof/cmdline").HandlerFunc(pprof.Cmdline)
```

/debug/pprof/goroutine

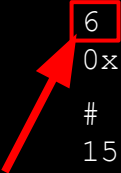
```
1 @ 0x96d3c8 0x96d1a3 0x968ae4 0x80543e 0x805650 0x4ef85a 0x5cca6e  
0x4f1cbe 0x4ef31e 0x46ee91
```

```
1 @ 0x43dfd3 0x44c6d4 0x44bc32 0x4034ac 0x402279 0x43dbf0 0x46ee91
```

...



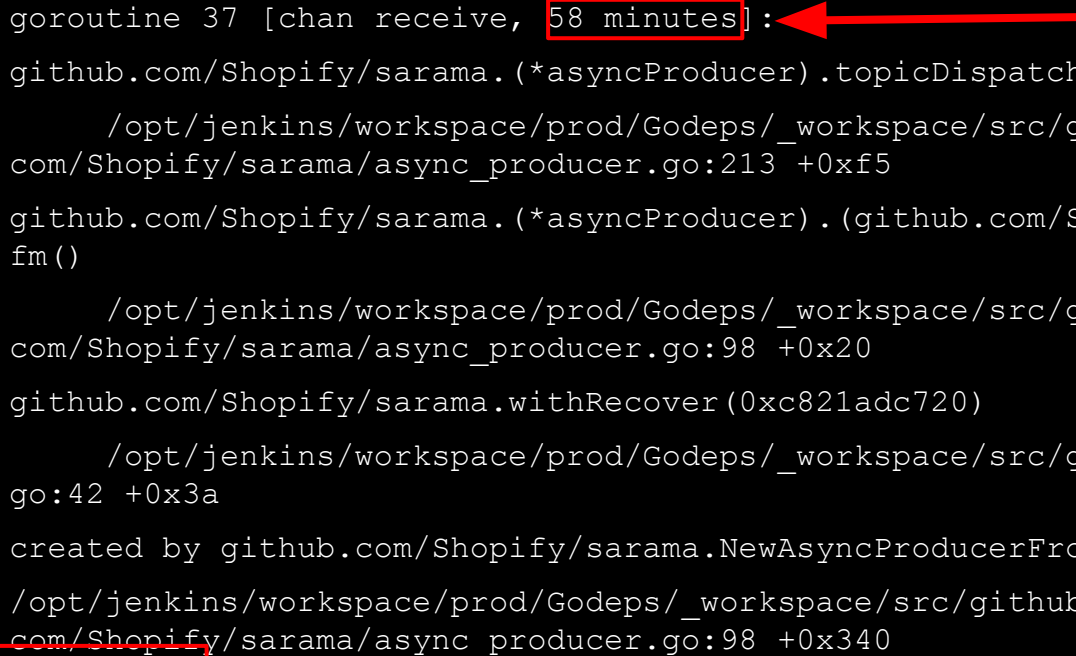
/debug/pprof/goroutine?debug=1



```
6 @ 0x43dfd3 0x4380fe 0x4375c0 0x499dda 0x499e46 0x49bb8a 0x4af6f4 0x502c07 0x50f110
0x6587a9 0x6589cc 0x4ffe07 0x46ee91
#      0x4375c0  net.runtime_pollWait+0x60      /usr/local/go/src/runtime/netpoll.go:
157
#      0x499dda  net.(*pollDesc).Wait+0x3a      /usr/local/go/src/net/fd_poll_runtime.
go:73
#      0x499e46  net.(*pollDesc).WaitRead+0x36  /usr/local/go/src/net/fd_poll_runtime.
go:78
#      0x49bb8a  net.(*netFD).Read+0x23a        /usr/local/go/src/net/fd_unix.go:232
#      0x4af6f4  net.(*conn).Read+0xe4          /usr/local/go/src/net/net.go:172
#      0x502c07  net/http.noteEOFReader.Read+0x67 /usr/local/go/src/net/http/transport.
go:1370
#      0x50f110  net/http.(*noteEOFReader).Read+0xd0      <autogenerated>:126
#      0x6587a9  bufio.(*Reader).fill+0x1e9        /usr/local/go/src/bufio/bufio.go:97
```

/debug/pprof/goroutine?debug=2

```
goroutine 37 [chan receive, 58 minutes]:  
    github.com/Shopify/sarama.(*asyncProducer).topicDispatcher(0xc820015140)  
        /opt/jenkins/workspace/prod/Godeps/_workspace/src/github.  
com/Shopify/sarama/async_producer.go:213 +0xf5  
    github.com/Shopify/sarama.(*asyncProducer).(github.com/Shopify/sarama.topicDispatcher)-  
fm()  
        /opt/jenkins/workspace/prod/Godeps/_workspace/src/github.  
com/Shopify/sarama/async_producer.go:98 +0x20  
    github.com/Shopify/sarama.withRecover(0xc821adc720)  
        /opt/jenkins/workspace/prod/Godeps/_workspace/src/github.com/Shopify/sarama/utills.  
go:42 +0x3a  
    created by github.com/Shopify/sarama.NewAsyncProducerFromClient  
        /opt/jenkins/workspace/prod/Godeps/_workspace/src/github.  
com/Shopify/sarama/async_producer.go:98 +0x340
```



Go heap usage debugging

- `alloc_objects` very useful
- go's CPU usage around GC strongly related to number of allocated objects



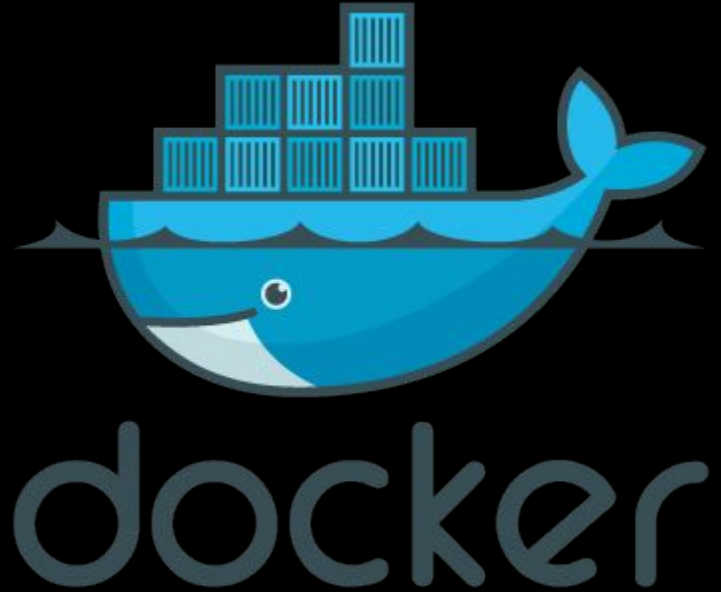
Example

```
go tool pprof -alloc_objects /tmp/server http://box:6060/debug/pprof/heap
```

```
(pprof) top
578225004112 of 902319909557 total (64.08%)
Dropped 756 nodes (cum <= 4511599547)
Showing top 10 nodes out of 112 (cum >= 75735754111)
      flat  flat%   sum%       cum   cum%
58755173926  6.51%  27.36% 117454393178 13.02%  github.
com/golang/protobuf/proto.(*Buffer).dec_slice_bool
... .
```

Docker

- Containers!
- Easy deployment!
- Static compile makes small docker images
- SignalFx container size
 - Go => 5MB
 - Java => 450 MB



Static go compiles for Docker

```
CGO_ENABLED=0 go build -v -installsuffix .
```

```
FROM scratch
```

```
COPY ./ingest /ingest
```

```
CMD ["/ingest"]
```

File explorer

- Just export / for docker containers
- `http.FileServer`
- Good for self download (pprof needed)



```
curl http://host.com:6060/debug/files/ingest >/tmp/ingest
```

Expvar

- Export JSON strings or numbers
- Visible in a browser
- Easy to use
- Best use in `func init()`
- Cmd line/env/build SHA/etc

Expvar the bad

Unable to unregister a variable

panic() on double register

What about non init state?

Global mutable state is an anti-pattern

Luck for us, the code is open
source!

github.com/signalfx/golib/expvar2

```
e := expvar2.ExpvarHandler{}
e.Exported["hello"] = expvar.Func(func() interface{} {
    return "world"
})
e.Exported["distconf"] = distconf.Var()
e.Exported["env"] = expvar2.EnvironmentalVariables()
handler.Path("/debug/vars").Handler(&e)

// http://localhost:6060/debug/vars?pretty=1&filter=env
```

Dynamic config

- Config for queue sizes, timeouts, etc
- We use zookeeper for config
- env > file > zookeeper
 - file/env allow easy testing and dev
- Turn debug levels on/off
- Can debug specific requests
- <https://github.com/signalfx/golib>



Dependencies

- Using master of dependencies is bad
- Need repeatable builds
- Vendoring is a requirement
- No pip, no cargo, no npm :(:(:(
- I blame “go get” for this mess
- <https://github.com/kardianos/govendor>
- <https://github.com/tools/godep>

Exported metrics

- Open source

- Graphite
- OpenTSDB
- InfluxDB

- SaaS

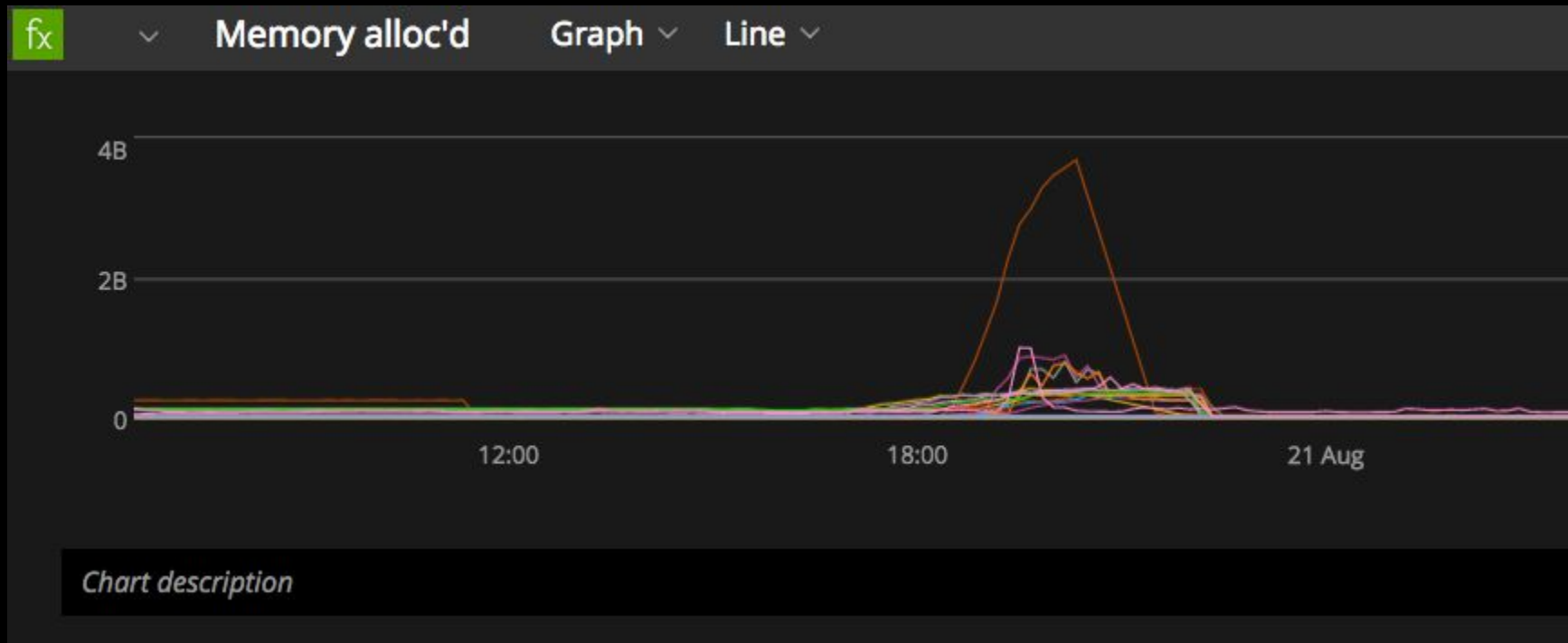
- SignalFx

- `runtime.NumGoroutine()`
- `runtime.ReadMemStats()`
 - `TotalAlloc`
 - `Alloc`
 - `PauseTotalNs`
- `time.Now().Sub(start)`
- `runtime.NumCgoCall()`

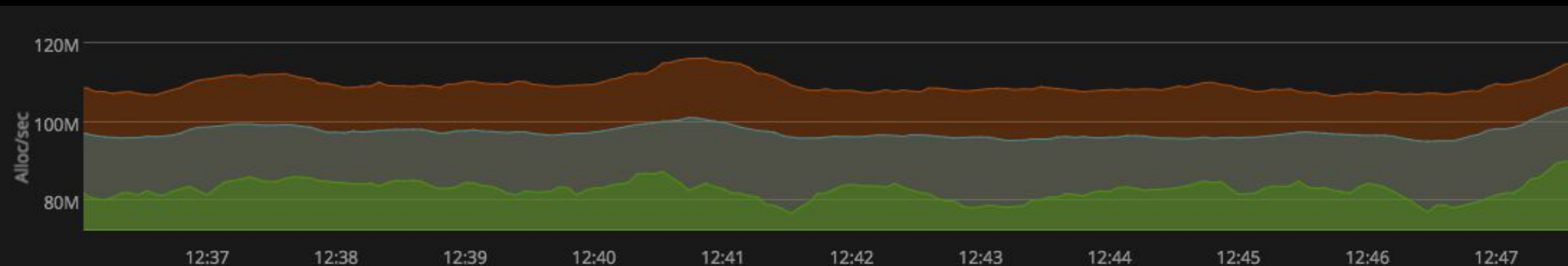
Count/defer pattern

```
atomic.AddInt64(&m.TotalConnections, 1)
atomic.AddInt64(&m.ActiveConnections, 1)
defer atomic.AddInt64(&m.ActiveConnections, -1)
start := time.Now()
next.ServeHTTP(rw, r)
reqTime := time.Since(start)
atomic.AddInt64(&m.TotalTimeNS, reqTime.Nanoseconds())
```

A single bad server



Tier Min/Median/Max Alloc/sec

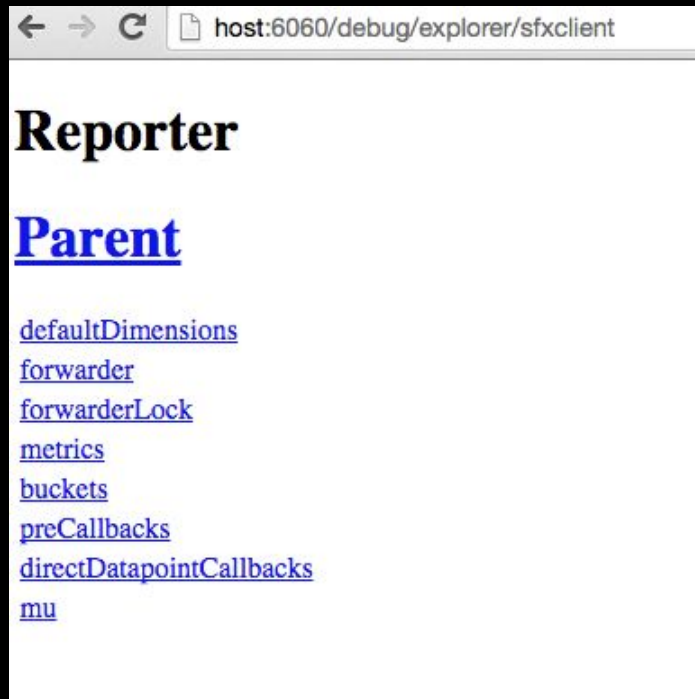


TotalAlloc min/median/max

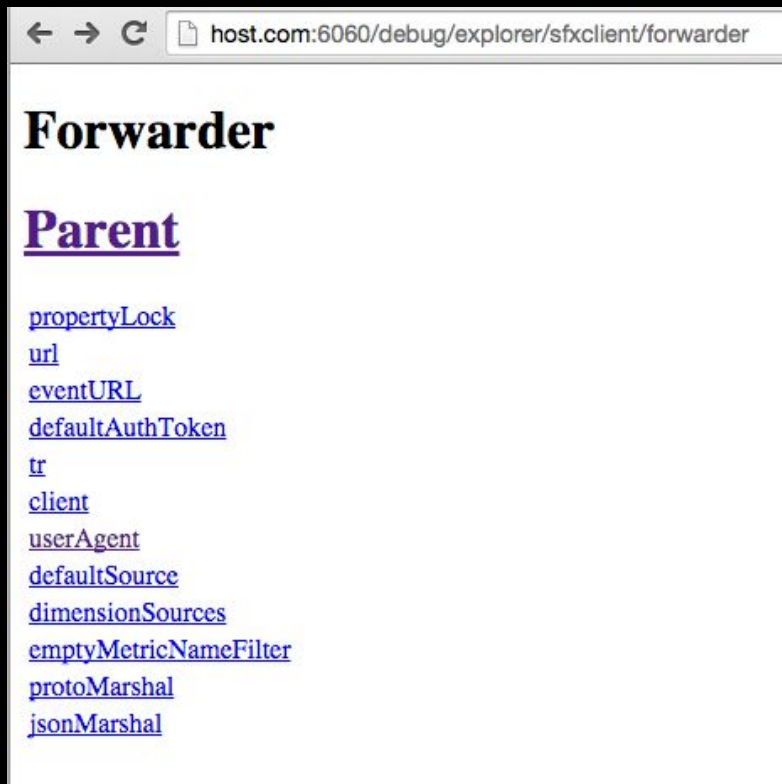
Object explorer

- What if you forgot to export a variable?
- Uses reflection for ad hoc exploration
- <https://github.com/signalfx/golib/tree/master/explorables>

Object explorer



Object explorer



Object explorer



Static checks

- golang.org/x/tools/cmd/goimports
- [gofmt](#)
- golang.org/x/tools/cmd/vet
- github.com/golang/lint/golint
 - Will need filters
- github.com/fzipp/gocyclo
- golang.org/x/tools/cmd/cover
- Wrapped into <https://github.com/cep21/goverify>



Questions?

- We're hiring
 - jobs@signalfx.com

The Signal FX logo is a green rectangle with a white border. Inside the rectangle, the words "signal fx" are written in a white, lowercase, sans-serif font. The "signal" part is in a regular weight, while the "fx" part is in a bolder weight.

signal fx