Logging for golang microservices? EFK?

fluentd elasticsearch kibana

Wojciech Barczyński - Hypatos.ai (SMACC.io) | LI | Twitter | 13 March 2019

ABOUT ME

- Head of Engineering - Hypatos.ai/smacc.io (FinTech/AI)
- Before:
  System Engineer && Developer Lyke (RocketInternet)
- Looking:
  Tools for efficent teams

POINT OF VIEW

- Software Developer
- startups && fast-moving environment
- Infra and platform should just work
- Infra and platform ~ invisible

MY GOAL

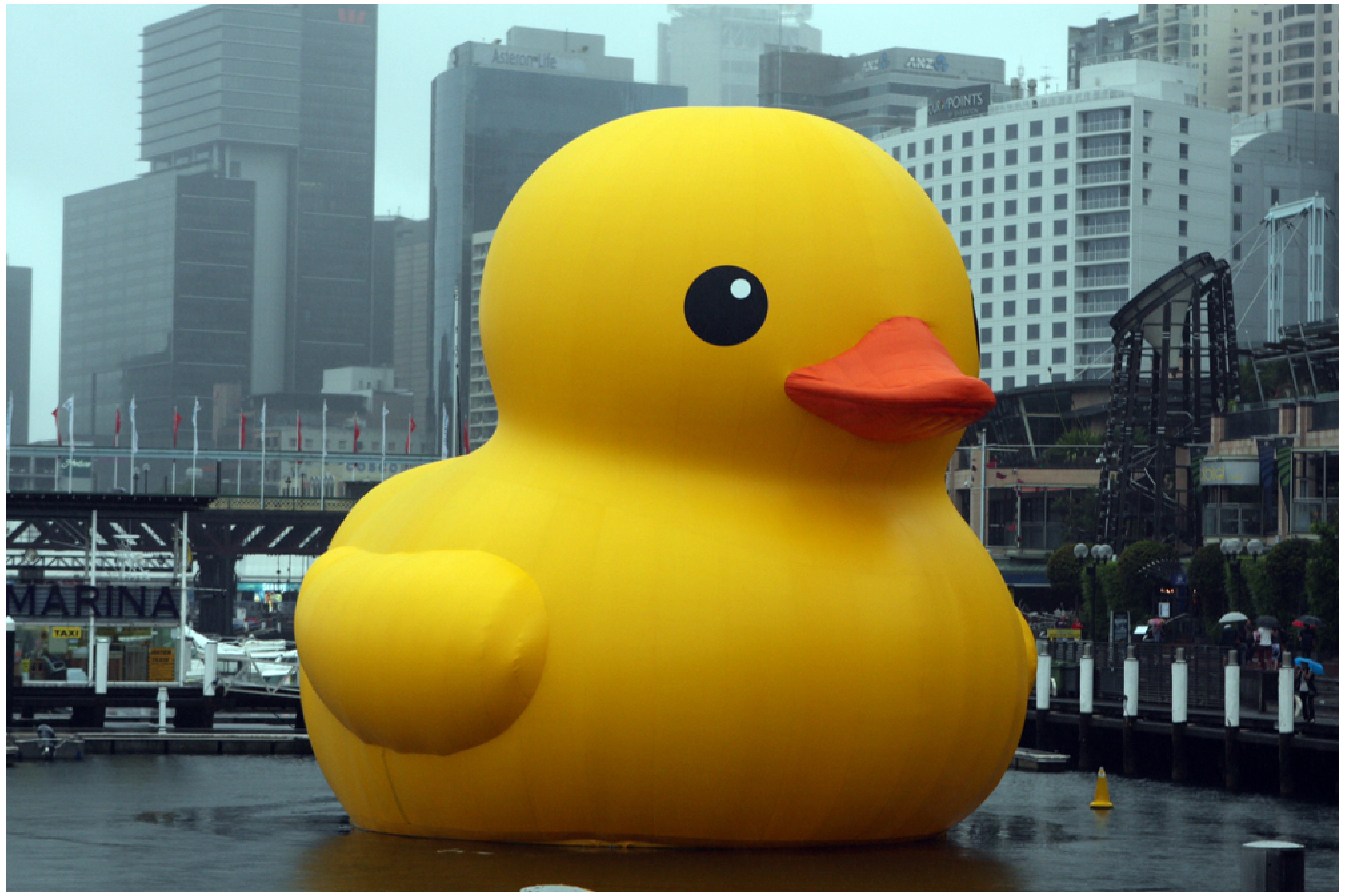
- Do not forget to start with monitoring
- What are the logging frameworks for Golang?
- What is the best strategy?

START WITH MONITORING

- Check my previous talks wojciech12/talk_monitoring_with_prometheus
- Peter Bourgon's talk on Go for Industrial Programming

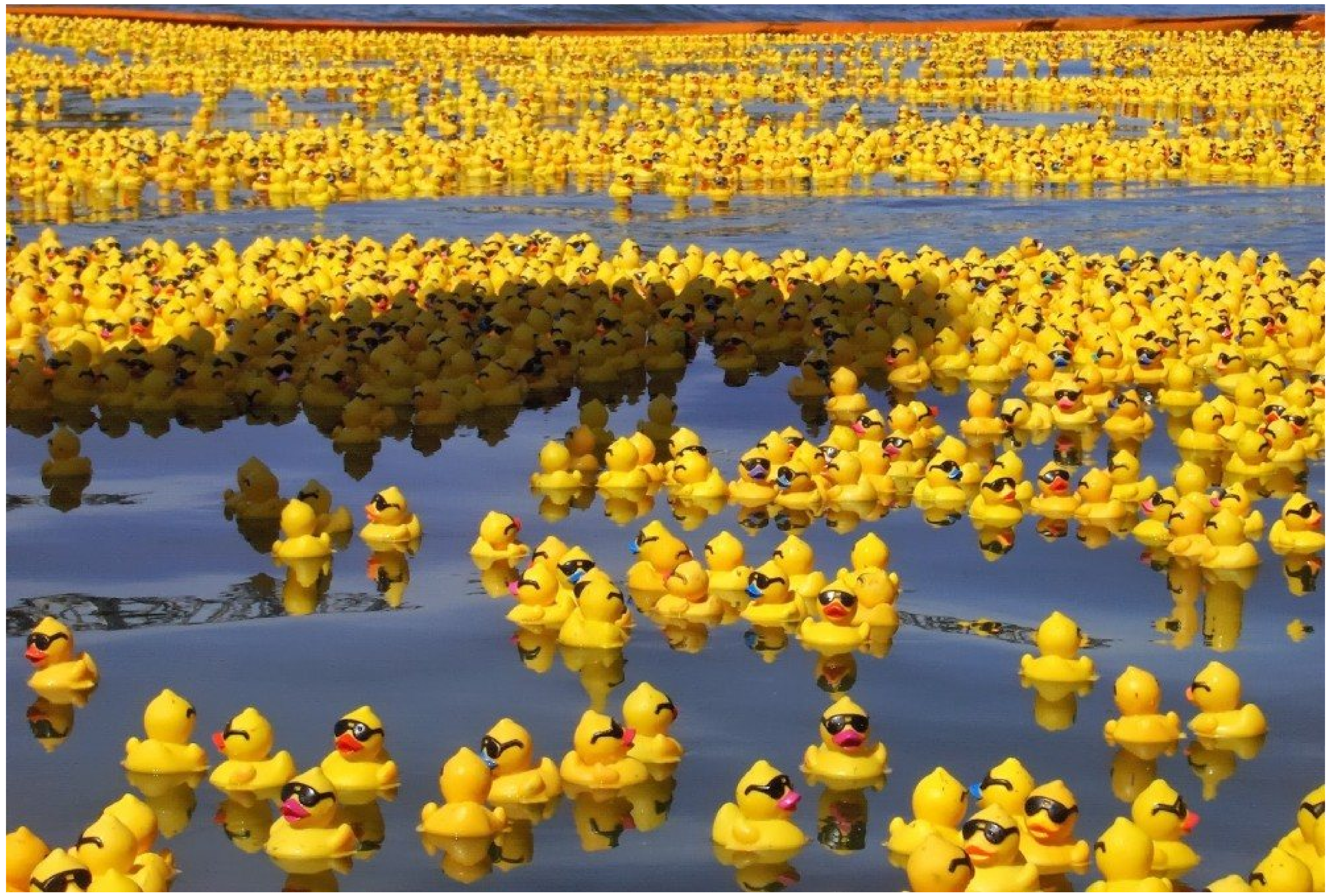
WHY?

MONOLIT ;)



WHY?

MICROSERVICES ;)



OBSERVABILITY

- Monitoring
- Logging
- Tracing

OBSERVABILITY

	Metrics	Logging	Tracing
CapEx	Medium	Low	High
OpEx	Low	High	Medium
Reaction	High	Medium	Low
Investigation	Low	Medium	High

Go for Industrial Programming by Peter Bourgon

CENTRALIZED LOGGING

- Debugging tool
- Post-mortem
- Finding the needle
- ! High TCO

Notice: you can get a long way with `grep`

LOGGING

- Structured
- Unstructured

LOGGING

- Stream of discrete events
- Best: structured at the caller site
- 12factorapps: push on stdout

LOGGING

Golang library:

- pkg/log
- logrus
- uber-go/zap

pkg/log

```go
package main

import (
        "log"
        "fmt"
)

func main() {
        log.Println("Hello World!")
        log.Fatal("Buum!")
        fmt.Println("You will not see me!")
}
```

```
2009/11/10 23:00:00 Hello World!
2009/11/10 23:00:00 Buum!
Program exited: status 1.
```

pkg/log

- very minimalistic
- you print log and move on
- if an error, handle it XOR bubble up!
- XOR die `Fatalf`, `Errorf`!

See Dave Cheney's blog post on logging

pkg/log

```
err := somethingHard()
if err != nil {
        log.Error("oops, something was too hard", err)
        return err // what is this, Java ?
}
```

From Dave Cheney's blog post on logging

pkg/log

```
err := somethingHard()
if err != nil {
        log.Error("oops, something was too hard", err)
        return err // what is this, Java ?
}
```

From Dave Cheney's blog post on logging

pkg/log

```
if err := planA(); err != nil {
        log.Infof("could't open the foo file, continuing with
            err)
        planB()
}
```

From Dave Cheney's blog post on logging

pkg/log

```
const (
        Ldate         = 1 << iota     // the date in the local
        Ltime                         // the time in the local
        Lmicroseconds                 // microsecond resolutio
        Llongfile                     // full file name and li
        Lshortfile                    // final file name eleme
        LUTC                          // if Ldate or Ltime is
        LstdFlags     = Ldate | Ltime // initial values for th
)
```

pkg/log

```
package main
import (
        "log"
        "os"
)
func main() {
        f, err := os.OpenFile("filename", os.O_WRONLY|os.O_CRE
        if err != nil {
                log.Fatal(err)
        }
        defer f.Close()
        log.SetOutput(f)
        log.Println("Hellow World!")
}
```

Error levels?

```go
package main
import (
        "log"
        "os"
)
func main() {
        f, err := os.OpenFile("filename",
            os.O_WRONLY|os.O_CREATE|os.O_APPEND, 0644)
        if err != nil {
                log.Fatal(err)
        }
        defer f.Close()
        log.SetOutput(f)
        log.Println("Hellow World!")
}
```

pkg/log

```go
package main
import (
    "log"
    "os"
)

func main() {
    var logInfo *log.Logger
    logInfo = log.New(os.Stdout,
        "INFO: ",
        log.Ltime|log.Lshortfile)

    logInfo.Println("Good to know!")
}
```

sirupsen/logrus

- formatters
- log level: `Trace,Debug,Info,Error`
- log and die: `Fatal, Panic`
- fields for the structed json logging

and more, e.g., hooks, support for tests

sirupsen/logrus

```go
package main

import (
  "os"
  log "github.com/sirupsen/logrus"
)

func init() {
  // Log as JSON instead of the default ASCII formatter.
  log.SetFormatter(&log.JSONFormatter{})

  // Output to stdout instead of the default stderr
  // Can be any io.Writer, see below for File example
  log.SetOutput(os.Stdout)
```

https://github.com/sirupsen/logrus

sirupsen/logrus

```go
package main

import (
  "os"
  log "github.com/sirupsen/logrus"
)

func init() {
  // Log as JSON instead of the default ASCII formatter.
  log.SetFormatter(&log.JSONFormatter{})

  // Output to stdout instead of the default stderr
  // Can be any io.Writer, see below for File example
  log.SetOutput(os.Stdout)
```

https://github.com/sirupsen/logrus

sirupsen/logrus

```
func main() {
  log.WithFields(log.Fields{
    "animal": "walrus",
    "size":   10,
  }).Info("A group of walrus emerges from the ocean")

  log.WithFields(log.Fields{
    "omg":    true,
    "number": 122,
  }).Warn("The group's number increased tremendously!")

  log.WithFields(log.Fields{
    "omg":    true,
    "number": 100,
  }).Fatal("The ice breaks!")
```

https://github.com/sirupsen/logrus

sirupsen/logrus

```go
package main

import (
  "os"
  "github.com/sirupsen/logrus"
)

var log = logrus.New()

func main() {
  log.Out = os.Stdout

  log.WithFields(logrus.Fields{
    "animal": "walrus",
```

https://github.com/sirupsen/logrus

uber-go/zap

- fast speed - less CPU nad less allocations
- less handy

uber-go/zap

```go
logger, _ := zap.NewProduction()
defer logger.Sync() // flushes buffer, if any
sugar := logger.Sugar()
sugar.Infow("failed to fetch URL",
  // Structured context as loosely typed key-value pairs.
  "url", url,
  "attempt", 3,
  "backoff", time.Second,
)
sugar.Infof("Failed to fetch URL: %s", url)
```

uber-go/zap

```go
logger, _ := zap.NewProduction()
defer logger.Sync()
logger.Info("failed to fetch URL",
  // Structured context as strongly typed Field values.
  zap.String("url", url),
  zap.Int("attempt", 3),
  zap.Duration("backoff", time.Second),
)
```

RECOMMENDATION

- logurus

BEST PRACTISES

- Do not log or create a Log in Gorountine.
- Use asynchronous libraries.
- Use all possible severity levels with caution.
- Stdout

BEST PRACTISES

Logging hints as an opportunity
to add more metrics for monitoring

LOGGING EVENTS != BUSINESS EVENTS

- Logging frameworks are OK if you lose data
- if you go for ES, ES is not designed to be your primary storage

ELK STACK

- Fluentd - collect
- Elasticsearch - store
- Kibana - visualize

Fluentd

- lightweight
- configuration for tranforming and routing logs
- out-of-the-box integration with Kubernetes

Elasticsearch

- scalable storage
- search engine
- easy to scale
- quite robust, but it might lose data

Kibana

- good tool
- still always feel slow if not on a big machine [1]

[1] TODO: measure it

Kibana and alerts.. no

For alerting: run elastcsearch queries from prometheus-elasticsearch exporter.

See a blog post from kuther.net

DEMO

DEMO: SIMPLE REST SERVICE

```
                                          ---------------
                                         |    Kibana     |
                                          ---------------
                                                 |
     -----------         ---------        ---------------
    |    App    | ----> | Fluentd | ---> | ElasticSearch |
    |           |       |         |      |               |
     -----------         ---------        ---------------
```

DEMO:

- http://127.0.0.1:8080/hello - service
- http://127.0.0.1:5601 - Kibana
- http://127.0.0.1:9200 - ElasticSearch

DEMO

```
☁  demo ⚡ make start
☁  demo ⚡ docker ps
CONTAINER ID        IMAGE                              PORTS
74021b1bb310        httpd                              0.0.0.0:80
4a461f90d5c5        talk-observability-log_fluentd     5140/tcp,
4be4dbba931b        kibana:6.6.1                       0.0.0.0:56
342e290e1afd        elasticsearch:6.6.1                0.0.0.0:92
```

DEMO: GENERATE CALLS

```
☁  demo      ⚡ make srv_wrk_random
```

With error injection

DEMO

Everything Included:

- One command start with docker-compose
- Fluentd config
- Sample webservice

SUMMARY

- Monitoring saves your time
- Debugging helps when things go south
- Checking logs == debuging vs having tests

THANK YOU

ps. We are hiring: TESTER/QA, FRONT-DEV, PM

```
def distance_matrix(regions):
    """ Computes a distance matrix against a region list """
    tuples = [r.as_tuple() for r in regions]
    return cdist(tuples, tuples, region_distance)


def clusterize(words, **kwargs):
    # TODO: write a cool docstring here
    db = DBSCAN(metric="precomputed", **kwargs)
    X = distance_matrix([Region.from_word(w) for w in words])
    labels = [int(l) for l in db.fit_predict(X)]
```

BACKUP SLIDES

```
def distance_matrix(regions):
    """ Computes a distance matrix against a region list """
    tuples = [r.as_tuple() for r in regions]
    return cdist(tuples, tuples, region_distance)


def clusterize(words, **kwargs):
    # TODO: write a cool docstring here
    db = DBSCAN(metric="precomputed", **kwargs)
    X = distance_matrix([Region.from_word(w) for w in words])
    labels = [int(l) for l in db.fit_predict(X)]
```

FLUENTD + K8S = <3

More in one of the next talks