What all developers need to know about logging

Golang UK 2016

Slawosz Slawinski
slawosz@gmail.com

My name is Slawosz Slawinski

- I code mostly in Ruby and Go
- https://github.com/slawosz
- @slawosz
- I work at Stratajet, private jet online booking system

What is topic of presentation?

- Logging 101
- Why do we write logs?
- Logging infrastructure
- How to structure logs in microservices/SOA
- Golang log libraries
- Tips & Tricks

Logging 101

- Log is simple text message printed by application
- Logs may have different levels:
  - Debug - we display lot of information, mostly for debugging (development/staging env)
  - Info - information needed to understand application flow, what happens in application
- Printing log message or writing log library is relatively easy, maintaining logging infrastructure is complex task

Why do we log?

- We want to know what our application is doing
- There is a lot of hard/strange application bugs where good logging is the key to solve them
- Log can replace lot of third party monitoring services, or help choosing most suitable for us
- With proper log infrastructure, we can add new information about application behaviour
    - Info to debug strange behaviour/error
    - Performance measurement of certain action
    - A/B testing

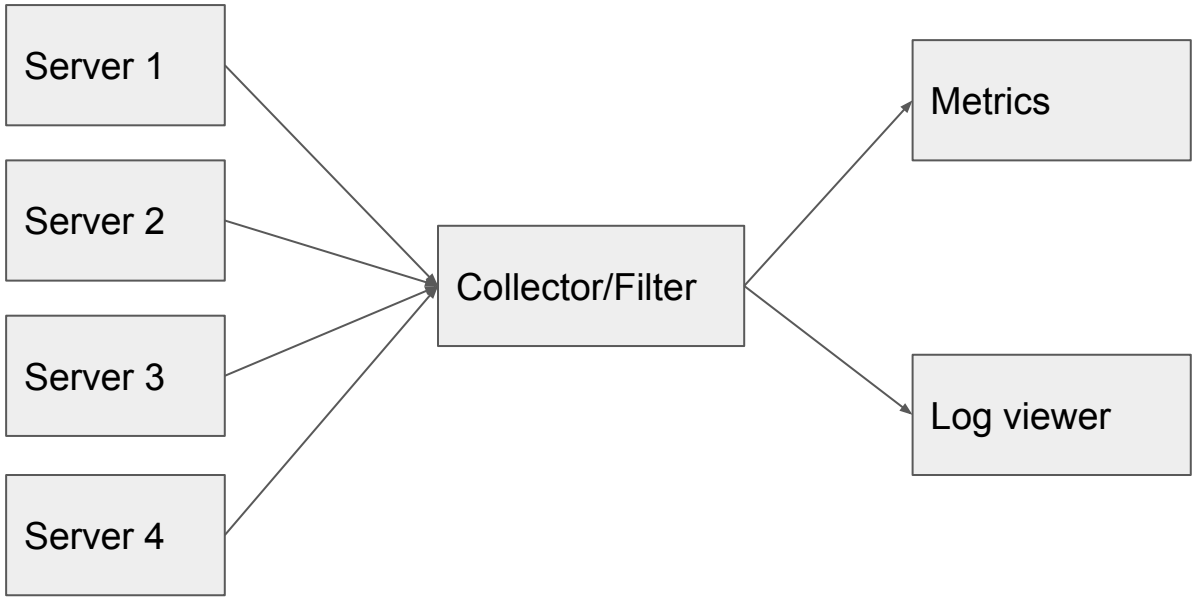
What do we log?

Everything that is essential to understand our application behaviour

Logging infrastructure

Proper logging infrastructure is not an easy task.

Logging infrastructure



Logging infrastructure

My favourite approach

- Application is logging to STDOUT
- STDOUT is redirected to file
- Log forwarders are moving logs to centralized log server
- Log server allows to browse/search/analyze application behaviours
- Its unix style, one simple step at the time
- Allows to use well known tools for simple log processing

Logging infrastructure

Make sure your infrastructure will not cause problems to your application

Logging infrastructure - applications

Open Source

- Logstash - log filtering and forwarding
- Kibana - analysis and visualization

Commercial solutions

- Summologic
- Splunk (expensive, but excellent)

Logs vs application metrics vs tracing

- Logs are general debug information
- Metrics measure application performance
- Tracing informs what action were performed
- We can use logs to save metrics and tracing
  - Easy to extract
  - Adds important context to log messages
- After some experience we can separate traces and metrics

Log message structure

```
[timestamp][RID=foobar][RD=0.003][LD=0.001]⏎
[service=API][search][results=432]
```

Log message structure

Timestamp

`[timestamp]` `[RID=foobar][RD=0.003][LD=0.001][service=API][search][results=432]`

Log message structure

Request ID/Unique ID/Trace ID

`[timestamp]` **`[RID=foobar]`** `[RD=0.003][LD=0.001][service=API][search][results=432]`

Log message structure

Request Delta - how much time has passed from the beginning of request?

`[timestamp][RID=foobar]` **`[RD=0.003]`** `[LD=0.001][service=API][search][results=432]`

Log message structure

Local delta - how much time has passed from the previous message?

`[timestamp][RID=foobar][RD=0.003]` **`[LD=0.001]`** `[service=API][search][results=432]`

Log message structure

Service name

`[timestamp][RID=foobar][RD=0.003][LD=0.001]` **`[service=API]`** `[search][results=432]`

Log message structure

Tag/keyword and key value pair

`[timestamp][RID=foobar][RD=0.003][LD=0.001][service=API]` **`[search][results=432]`**

Log message structure

Log message

`[timestamp][RID=foobar][RD=0.003][LD=0.001][service=API][search][results=432]` **Some msg**

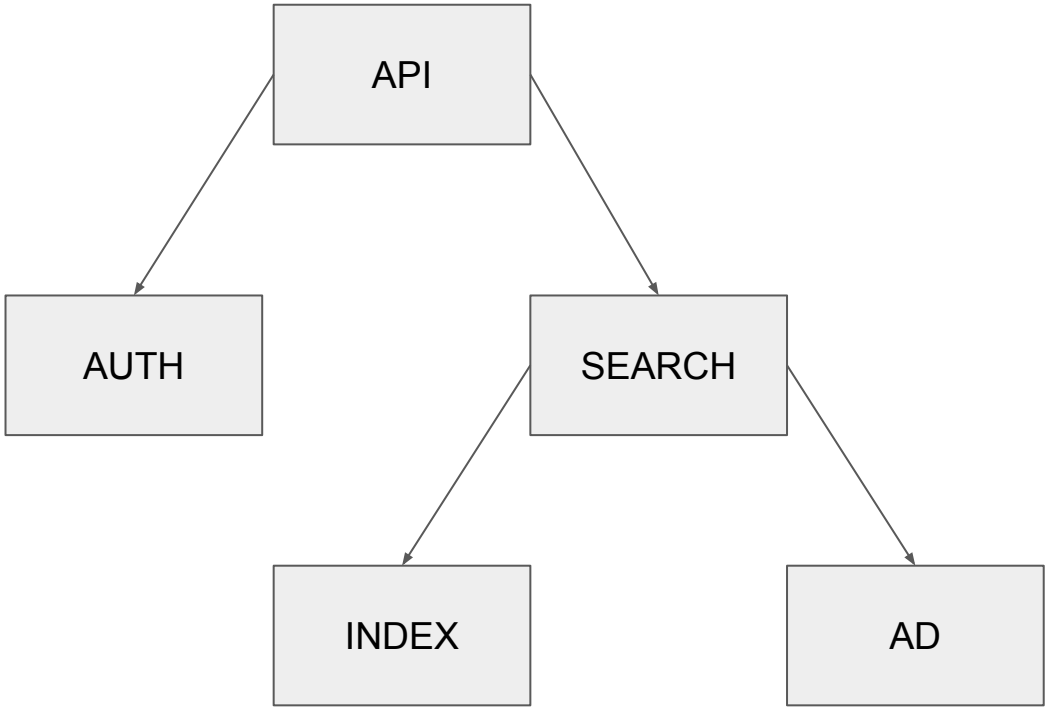
Log message structure

We should adapt log message structure to our application needs

Log message - customization examples:

- Process id
- Current memory consumption
- Server url
- Any information that would speed up your development process

Logging in microservices/SOA



Logging in microservices/SOA

```
[service=API][req="POST /search?q=golanguk"]

[service=AUTH][authenticate][user_id=123]*

[service=SEARCH][search_started]

[service=INDEX][idx_search][keyword="golanguk"]

[service=INDEX][idx_search][results=238]

[service=AD][ad_selected][id=2432]

[service=SEARCH][search_finished]

[service=API] [resp="200 OK"]
```

Golang log libraries

- Package log - simplest logger for golang
  - Suitable for simple projects, limited
- Glog - logging library from google, allows more control on log levels
- https://github.com/avelino/awesome-go#logging - lot of libraries to choose from, you will find inspiration for your app

Tips and tricks

Add option for your application to switch log level from info to debug - this will help debugging problem with application on production (how about control api query to do it so: **/_internal/toggle_log_level**)

Tips and tricks

Profile your logging system

Tips and tricks

Log sampling - when your systems are producing too many logs, collect only every Xth message, statistically you will get enough logs to understand your app behaviour

Tips and tricks

Security - very important. Never log user passwords, keys, name etc. Try to minimize possibility to recognize user personality from logs. Try to separate access to data which can disclosure user personality.

Dapper - tracing system from google

Dapper is Google tracing system, explained in very interesting paper (btw, Google published lot of very good papers about their architecture)
http://research.google.com/pubs/pub36356.html

Thank you!

Gopher by Renée French