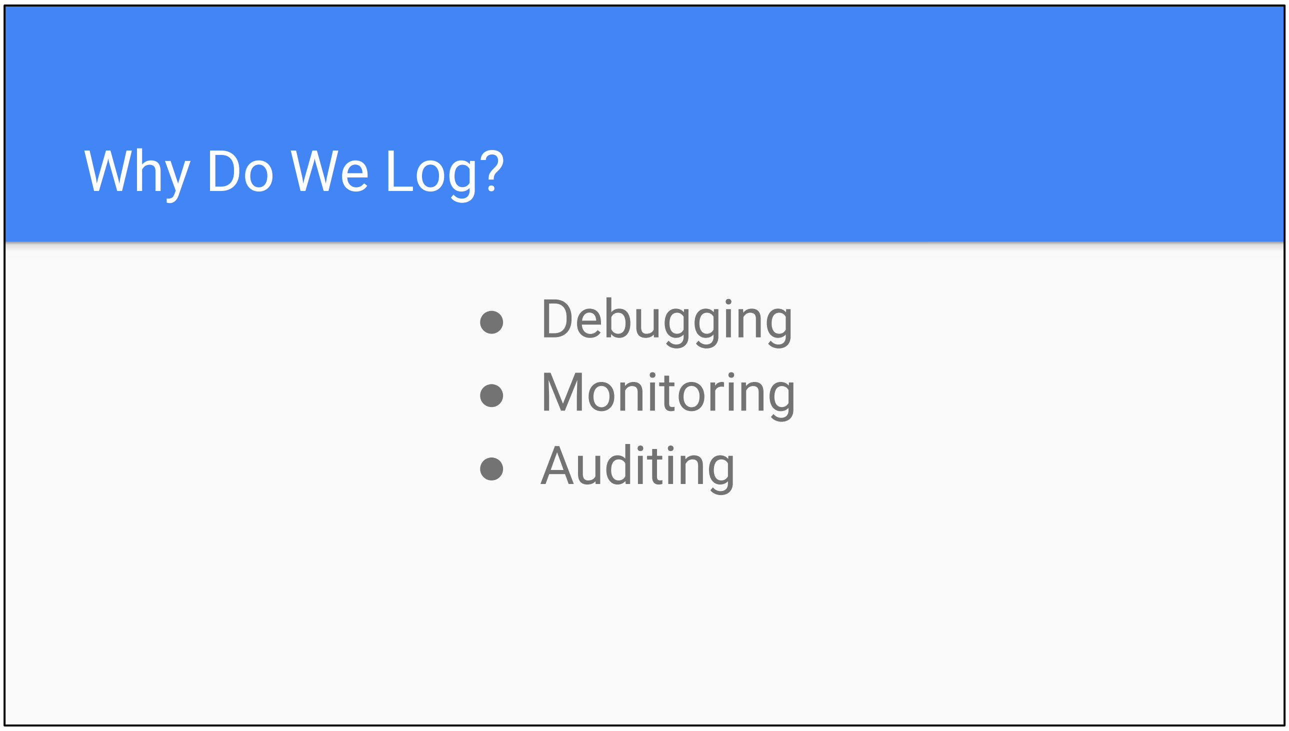
Structured Logging

Adam Wolfe Gordon
Edmonton Go Meetup
2017-06-26



Why Do We Log?

- Debugging
- Monitoring
- Auditing

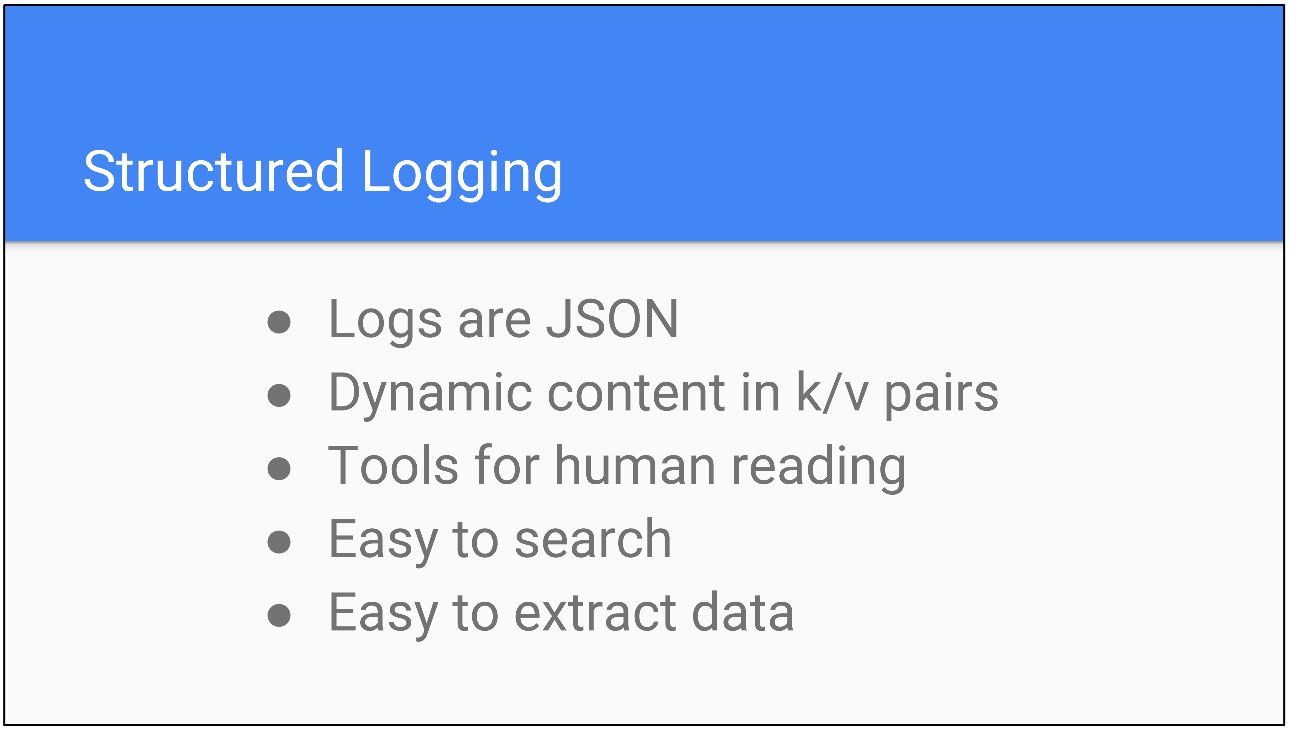
We all know what logging is. But a more fundamental question: why?

- Debugging during development or when we hit a problem.
- Monitoring or observing and understanding the behavior of a system. (Metrics are better for some use cases, but logging is great for others.)
- Audit trail. (For things that aren't too important - real audit log in a DB is better.)

How often do you read a log from beginning to end?

Almost never!
Usually searching through a log e.g. with grep, maybe reading a few lines up/down for context, extracting information e.g. with awk.

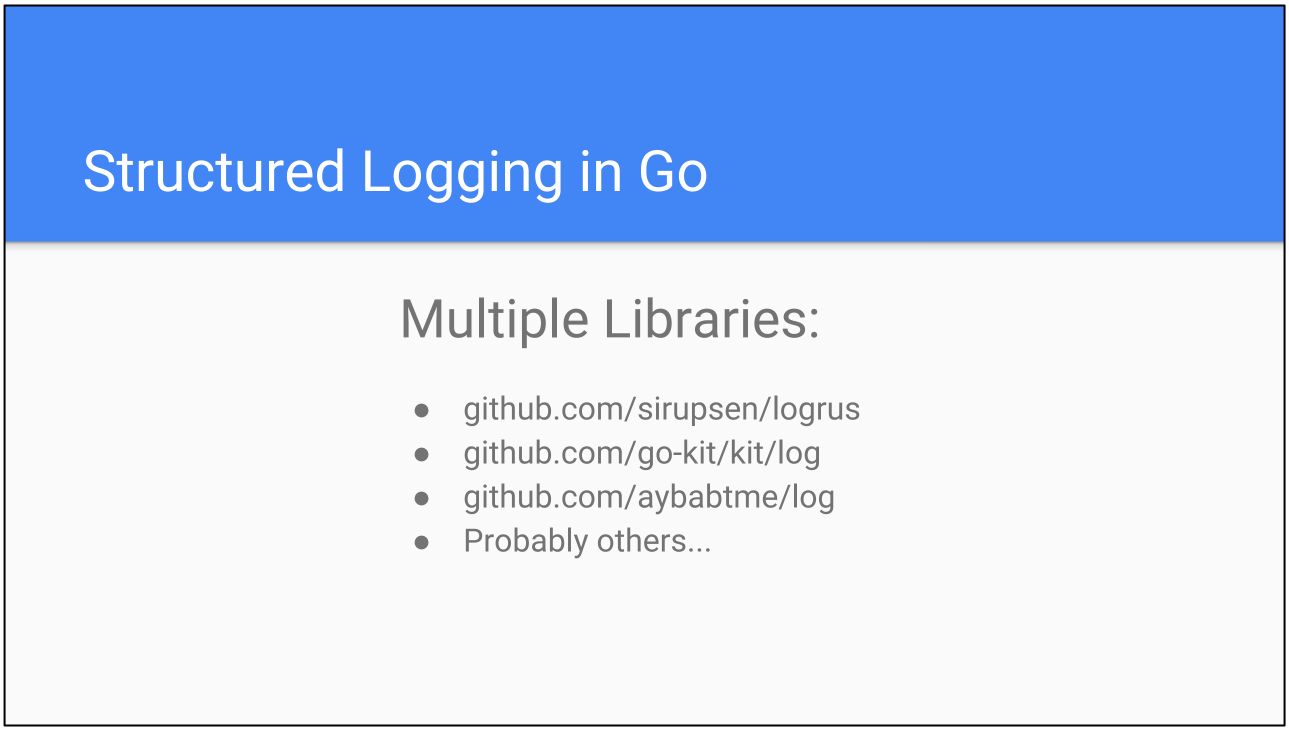
Why optimize your logs for human reading if you almost never read them? Why not optimize them for searching and extraction?



Structured Logging

- Logs are JSON
- Dynamic content in k/v pairs
- Tools for human reading
- Easy to search
- Easy to extract data

Structured logging means making logs machine-readable - e.g. JSON.
Preferably keep messages constant and represent dynamic values in k/v pairs (for searchability, extraction).
Use tools to transform JSON logs into human-readable logs.
Easy to search and extract data using e.g. jq.

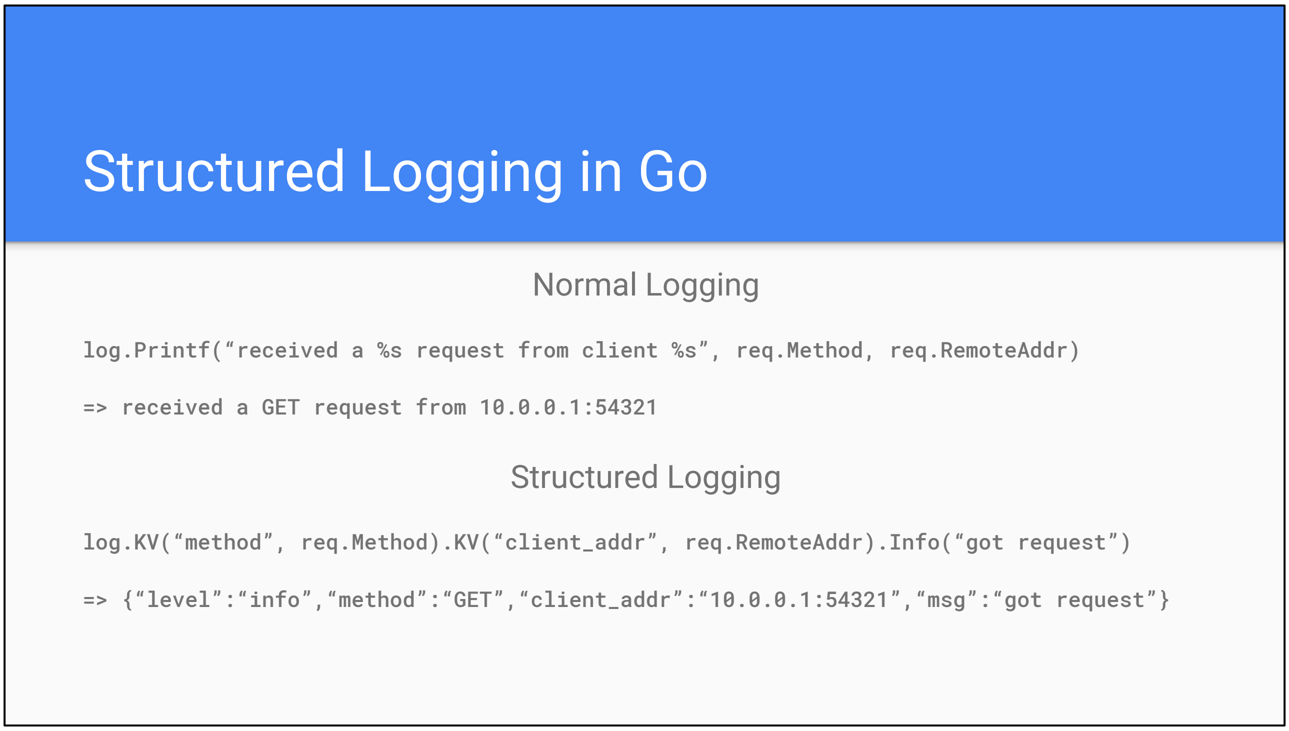


Structured Logging in Go

Multiple Libraries:

- github.com/sirupsen/logrus
- github.com/go-kit/kit/log
- github.com/aybabtme/log
- Probably others...

There are a bunch of Go libraries for structured logging.
For this talk I'll use aybabtme/log, because my friend Antoine wrote it and it's simple (very small API, wraps go-kit log).



Structured Logging in Go

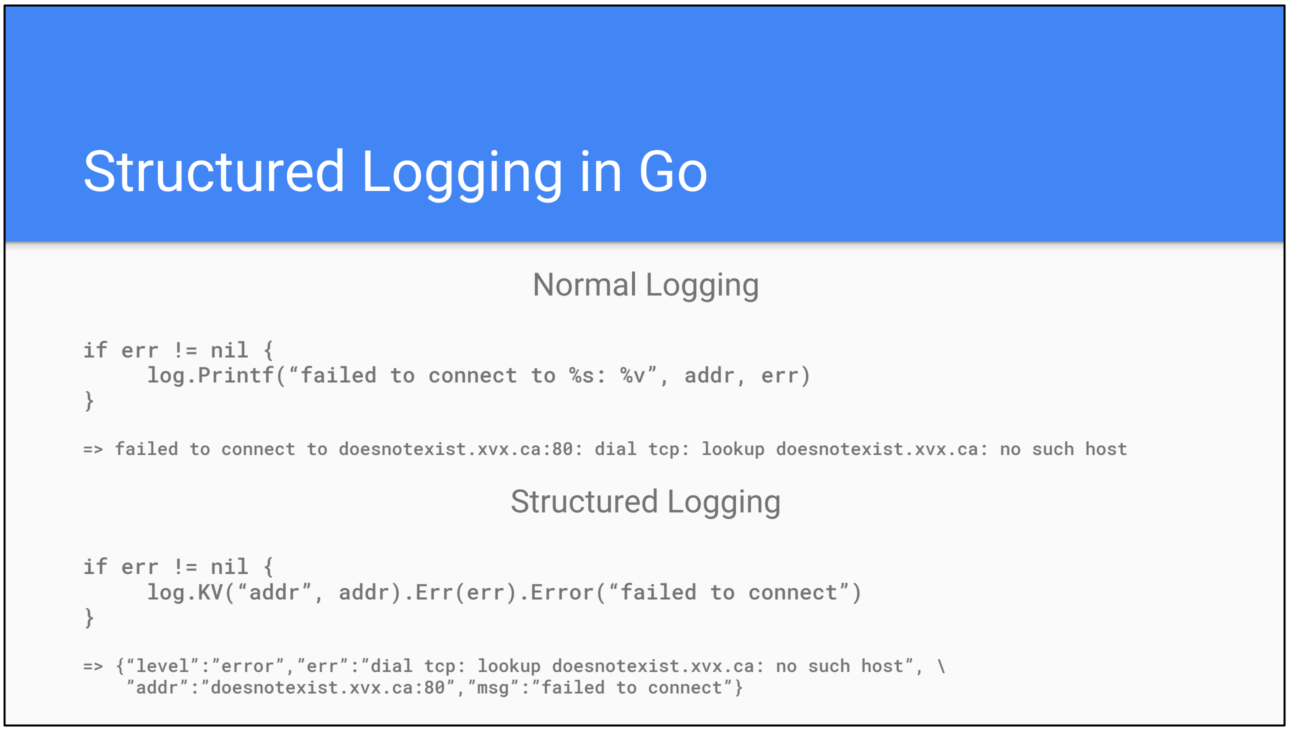
Normal Logging

```
log.Printf("received a %s request from client %s", req.Method, req.RemoteAddr)

=> received a GET request from 10.0.0.1:54321
```

Structured Logging

```
log.KV("method", req.Method).KV("client_addr", req.RemoteAddr).Info("got request")

=> {"level":"info","method":"GET","client_addr":"10.0.0.1:54321","msg":"got request"}
```

String formatting turns into k/v pairs. The message is a constant.
Leaving a couple of things out for brevity, e.g. time, line number.



Structured Logging in Go

Normal Logging

```
if err != nil {
     log.Printf(“failed to connect to %s: %v”, addr, err)
}
```

```
=> failed to connect to doesnotexist.xvx.ca:80: dial tcp: lookup doesnotexist.xvx.ca: no such host
```

Structured Logging

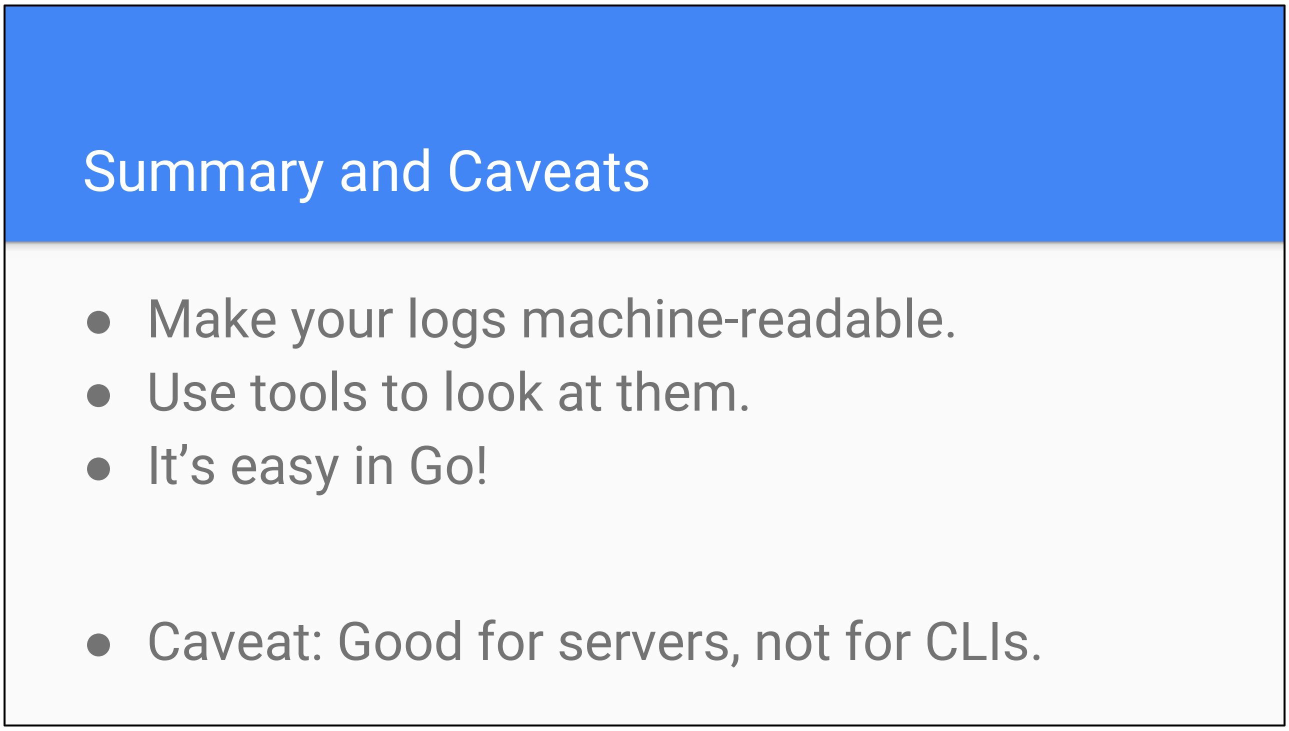
```
if err != nil {
     log.KV(“addr”, addr).Err(err).Error(“failed to connect”)
}
```

```
=> {“level”:”error”,”err”:”dial tcp: lookup doesnotexist.xvx.ca: no such host”, \
    ”addr”:”doesnotexist.xvx.ca:80”,”msg”:”failed to connect”}
```

Example of logging in an error case. The error message becomes a k/v; log message is still constant.

Live Example Time!

- Walk through https://github.com/adamwg/structured-logging-example
- Show raw output
- Log to a file
    - Show output with humanlog
    - Filter output with jq
    - ELK; Show Kibana searches



Summary and Caveats

- Make your logs machine-readable.
- Use tools to look at them.
- It’s easy in Go!

- Caveat: Good for servers, not for CLIs.

Using structured logging lets you get more value from your logs.
Make them machine-readable. Use tools to parse them, look at them, explore them, etc.
There are plenty of Go libraries for structured logging, with varying features.

Note that this is mostly applicable to servers. If you’re building a CLI your logs should probably be human-oriented.

Questions?
Comments?

Adam Wolfe Gordon
@maybeawg
awg@xvx.ca

Shameless Plug:
DigitalOcean is always hiring Go developers.
We are very remote-friendly!
http://l.xvx.ca/do-jobs

References and Links

- Antoine Grondin's GopherCon 2016 lightning talk: https://github.com/gophercon/2016-talks/tree/master/AntoineGrondin-StructuredLoggingAndYou
- GoDoc for the library used in the example: http://godoc.org/github.com/aybabtme/log
- The humanlog tool: https://github.com/aybabtme/humanlog
- Source for the example: https://github.com/adamwg/structured-logging-example
- How to set up a simple ELK stack, as in the example: https://www.digitalocean.com/community/tutorials/how-to-use-the-digitalocean-elk-stack-one-click-application