Logging Best practices

Geshan Manandhar
Senior Software Engineer
THE ICONIC

@geshan

whoami

Geshan Manandhar

- Senior Software Engineer
- Microservices are good, agile is better :)


I work for THE ICONIC (Tech)
THE ICONIC
TECH

If dog is a man's best friend, logs are software engineer's best friend.

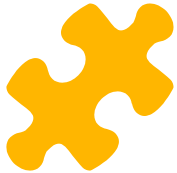
@geshan


We start from this, a pile of logs (if any) -- probably sorted
@geshan


Hopefully, end up in this. Following best practices :)
@geshan

This feature we deployed last week was working fine till yesterday now I have no idea why is it not working on production!

@geshan



Logging from application level

- If errors should be reported, normal operation also need to be logged
- Applications should log actions to provide visibility and observability
- This allows the software engineers to debug and pinpoint the problems faster in case of any issue

How does logging help you?

- If you have logs are the right places, you will find out where is the program not behaving as expected
- It helps you find things on production you were not sure of
- Be careful to not log secrets like passwords though

Having logs is like having a torch light in a dark place

@geshan

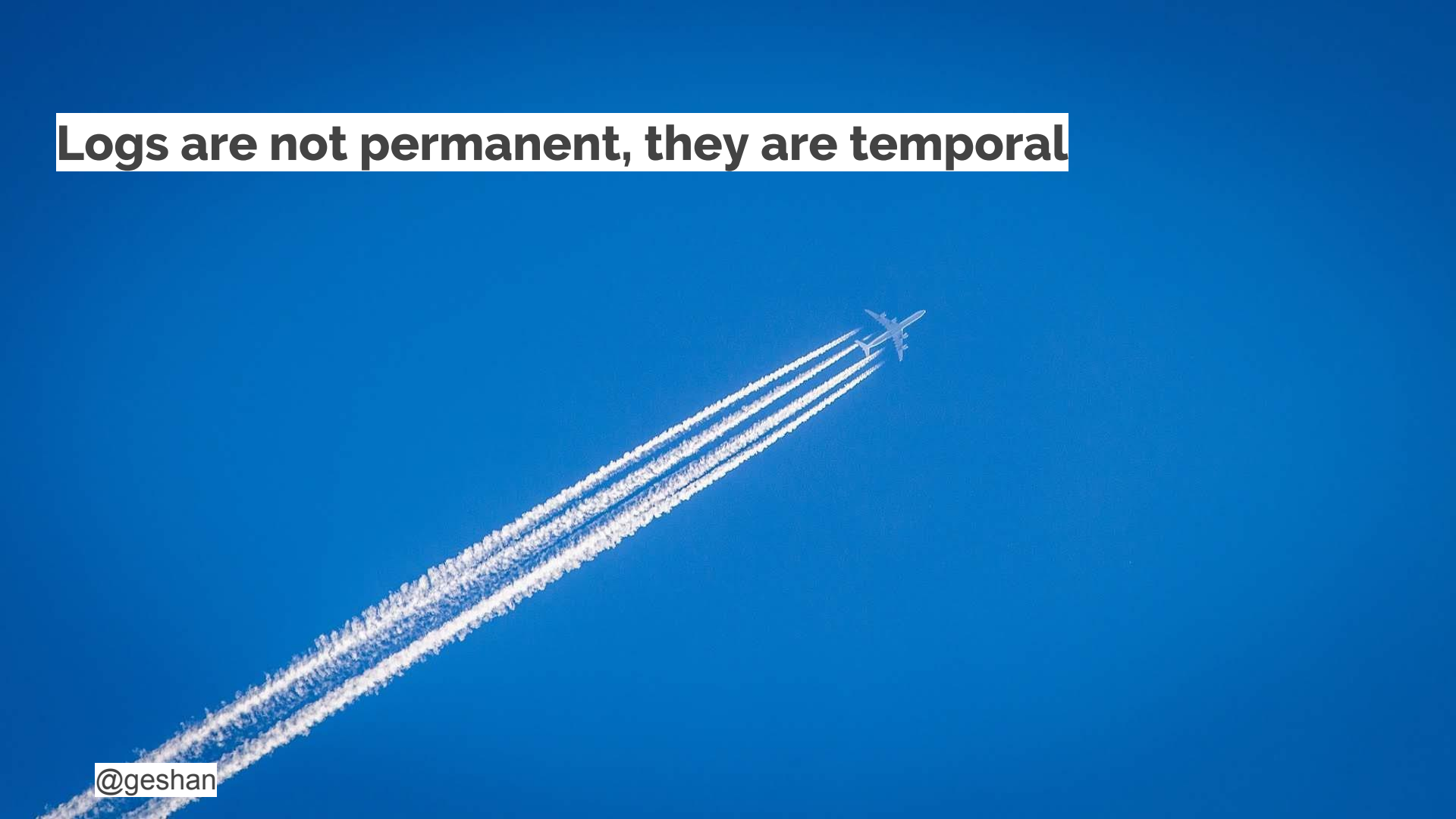
Log information optimally

Too much logs = noise, too less = inadequate information

Logging in microservices

- Same request ID travels through multiple apps/services
  - Like create shipment request travelled through 3 apps
  - Request ID 112Ac120 -> App A -> MS B -> Service C
- This also helps in distributed tracing between apps/microservices
- Istio telemetry is a good read (distributed tracing, visualizing…)

Logs are not permanent, they are temporal



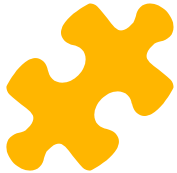
@geshan

Logging severity standards

How is alert different from notice for instance

@geshan

Logs severity levels



- Standard RFC-5425
  - 0 Emergency: system is unusable
  - 1 Alert: action must be taken immediately
  - 2 Critical: critical conditions
  - 3 Error: error
  - 4 Warning: warning
  - 5 Notice: normal but significant
  - 6 Informational: informational
  - 7 Debug: debug-level messages

Always follow severity standards

Emergency means your on-call phone rings at 2 AM.
Having agreed upon logging standards helps everyone.

Have structure in logs

Structured logs go a long way as it is easier to parse

@geshan

Structure your logs

- Define a log format like date is required, log title needs to be less than 255 characters
- Always add contextual information like request id, id of the subject in context like order id/order nr
- JSON can be used to structure and parse logs better
- Think of how to make searching ultra easy

Always provide context with structured logs

Follow a structure and format for logs.

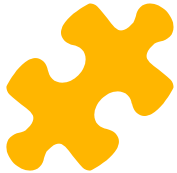
Context always helps, JSON is your friend.

Write logs carefully

Don’t add more milliseconds to your app performance because of logging

@geshan

Write logs async as far as possible



- If you start calling a 3rd party https API to write your logs it will add milliseconds to your app
  - Writing it locally then shipping it some other way (ELK)
  - Queues for logs can also be a good option
- With non sequential executing languages like **javascript** you can make it async easily

Use a trusted logging library

- Depending on the language your can choose one that suits your needs
- Some languages also come with built in support like Go Lang

Some Logging libraries

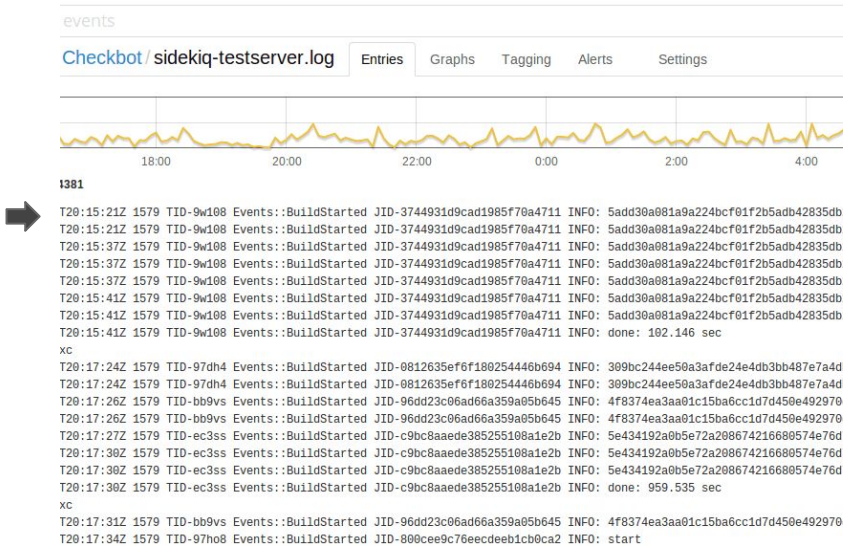
Language	Library	Github stars
PHP	**Monolog**	**~13.5k**
TypeScript/JS	**Winston**	**~12.5k**
Python	**Native**	**N/A**

Note: Don't forget monolog handlers and formatters :)

Monolog to logentries

```
public function [redacted]
{
    $this->logger->info( message: 'getting[redacted]

    return $this->[redacted]
        $this->[redacted]
        $this[redacted]
    );
}
```



Write logs asynchronously

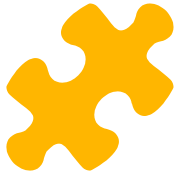
Non blocking logs are the best.

Be careful with console.log in JS/TS.

Log shipping is intelligent and efficient.

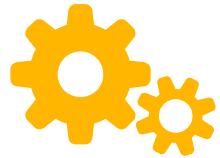
Tools we are using

Logentries.com aggregates most of our logs

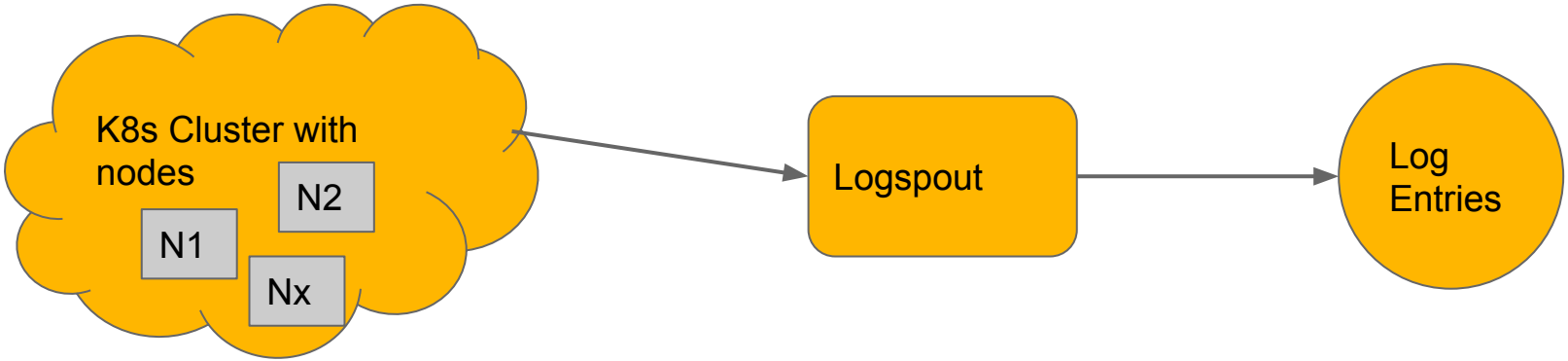


Log aggregators and viewers

- Logs can be aggregated, shipped and viewed multiple ways
- Primary choice might be between self hosted/managed or SaaS
  - Graylog, ELK stack are self hosted, self managed solution
  - Logentries, loggly, Sematext Logsense, Scaylr are some good SaaS options

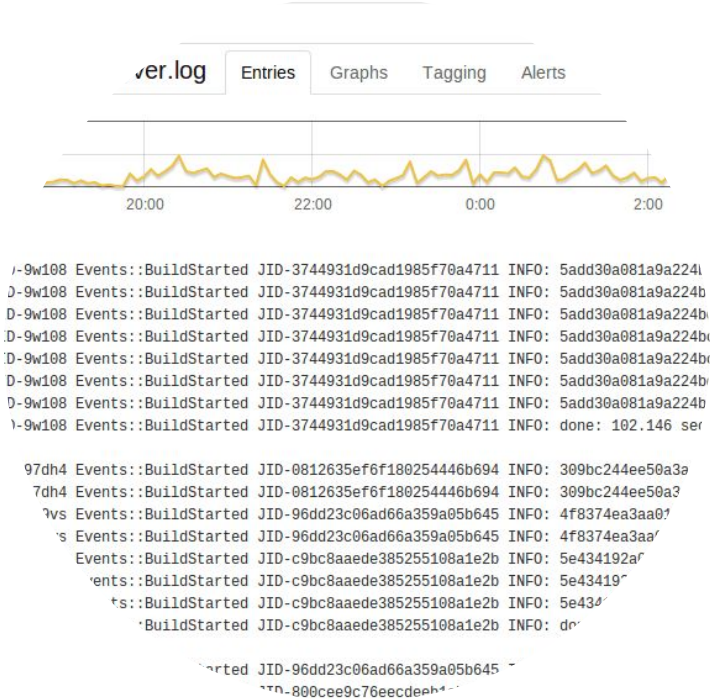


K8s container logs to LE



LogEntries

- Currently we are using logentries to view and search all our logs
- You can also create alerts with logs



Alerts with logs

- Searching and viewing logs are the primary requirements of a log management system
- Alerts add that extra zing
  - If I get “these” logs more than 80 times in 5 minutes send me an email or slack message is kind of an alert based on logs

Use the tools on disposal efficiently

Know how to search your logs, add dashboards if needed.
You can even set up alerts if some logs are consistent over time.

Logging -> Instrumentation -> Observability

- Instrument every meaningful number available for capture - source
  - tends to be things like incoming request counts, request durations, and error counts
  - No. of order per minute, no. of stuck payments
- Both logging and instrumentation are ultimately just methods to achieve system observability.

Thanks!

Any questions?

Credits/references

- https://blog.scalyr.com/2018/08/microservices-logging-best-practices/
- https://www.loggly.com/blog/30-best-practices-logging-scale/
- https://peter.bourgon.org/blog/2016/02/07/logging-v-instrumentation.html
- https://news.ycombinator.com/item?id=11054973
- https://surfingthe.cloud/dont-fear-node-js-console-log/
- https://tools.ietf.org/html/rfc5424
- https://en.wikipedia.org/wiki/Instrumentation_(computer_programming)
- https://www.loomsystems.com/blog/single-post/2017/01/26/9-logging-best-practices-based-on-hands-on-experience
- https://geshan.com.np/blog/2015/08/importance-of-logging-in-your-applications/
- https://blog.scalyr.com/2018/06/go-logging/
- https://blog.codeship.com/how-to-understand-logs-with-logentries/