cphbusi

COPENHAGEN BUSINESS ACADEMY





DevOps part 2/4: Logging

Jens Egholm Pedersen
<jeep@cphbusiness.dk>

Learning how to learn

- A word on metacognition
  - What does that mean?
- Dunning-Kruger effect
  - Stupid people think they are smart
  - Why is that a bad thing?
- Continuous feedback
  - Where would you like to be when this course ends?
  - Keep evaluating yourself
  - … and be honest!

See also: Dunning-Kruger effect, Metacognition improves your grade!

A note on requirements

- Requirements are agreed upon – then set in stone
  - Mostly
- Example for this: HTTP header Content-Type for JSON
  - Yes it's a standard
  - When you agree on something, you have to live with it. Period.
- You are the boss of the client
  - Telling the client what to do is just not how it works
- Luckily we are not in the real world. So what to do now?

Recap

- Service-level agreement (SLA)
- Monitoring
  - Why do we need it?
- Pratical:
  - Installation of Prometheus and Grafana
  - Dashboarding
  - Alerts

SLA metrics

- Common metrics (for web)
  - Uptime/availability (usually percentage of all time)
  - Mean response time (average time before answer)
  - Mean time to recover (time to recover after outage)
  - Failure frequency (number of failures/timeouts over time)

- What metrics did you use and why?

See also: SLA on Wikipedia, classification of SLA metrics

What you should know

Goals of today:

- Alerts in Grafana
- Understand what logging is and why it's needed
- Understand what auditing is and how to employ it
- Gain practical knowledge on how to use and install logging software

Literature: DevOps introduction

A note on Prometheus syntax

- Grafana can connect to multiple backends
- When querying Prometheus you need to use their syntax
  - `http_requests_total`
  - `http_requests_total{job="nodejs-prometheus"}`
  - `rate(http_requests_total[5m])`
  - `sum(rate(http_requests_total[5m])) by (job)`

See also: Prometheus queries

Prometheus dashboards

- Just because you can include everything doesn't mean you should!

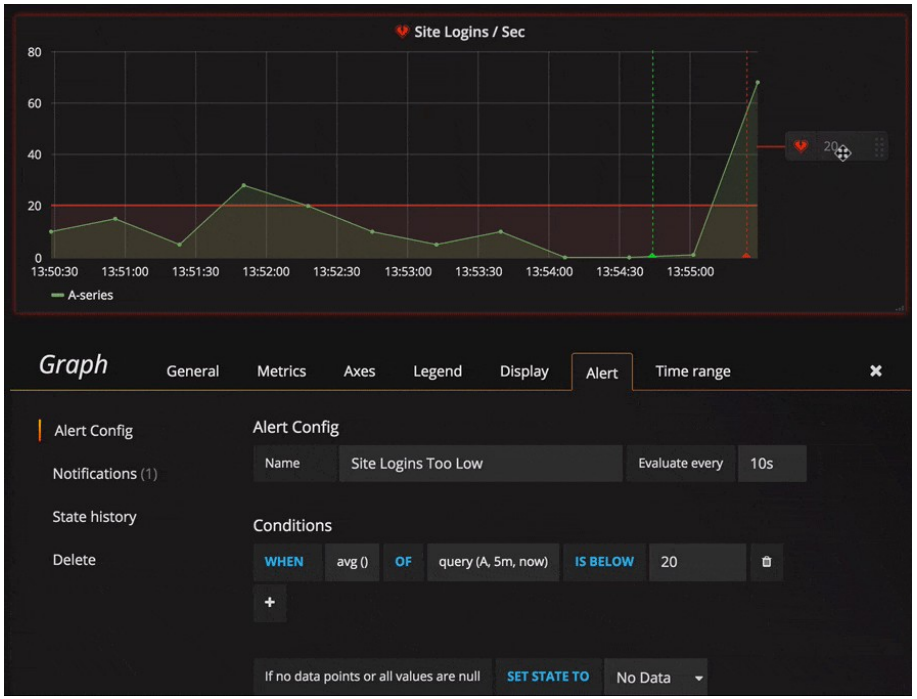
- `https://grafana.com/dashboards`

See also: Make DevOps Dashboard tell a Story

Alerts

- Active monitoring
  - Get notified when something goes wrong

- Alerts in Grafana:
  - On dasboards
  - Or via channel
    - Mail, PagerDuty, Telegram, Slack etc.



See also: Alerts in Grafana

Logging

- Recording of events that occur in software

- Purpose:
  - Understand activity (preliminary examinations)
  - Diagnosis (of an actual problem)
  - Audit trails

- Logs are essential to understanding activities in retrospect
  - Only source of information / proof

See also: Top 5 mistakes in logging

Logging standard

- What and how to log?
- Syslog standard
  - Developed around 1980

- Standard fields
  - Timestamp
  - Host
  - Application/process (id)
  - Facility level
  - Message

See also: Syslog on Wikipedia

Syslog

- What and how to log?
- Syslog standard

Facility level

Timestamp loghost Application/process

```
May 11 10:40:48 scrooge disk-health-nurse[26783]: [ID 702911 user.error]

  m:SY-mon-full-500 : partition health measures for /var did not suffice
```

Message id Message

See also: Syslog on Wikipedia

Log levels

- Syslog log levels (de facto standard)

Value	Severity	Keyword	Deprecated keywords	Description
0	Emergency	`emerg`	`panic` [8]	System is unusable. A panic condition.[9]
1	Alert	`alert`		Action must be taken immediately. A condition that should be corrected immediately, such as a corrupted system database.[9]
2	Critical	`crit`		Critical conditions, such as hard device errors.[9]
3	Error	`err`	`error` [8]	Error conditions.
4	Warning	`warning`	`warn` [8]	Warning conditions.
5	Notice	`notice`		Normal but significant conditions. Conditions that are not error conditions, but that may require special handling.[9]
6	Informational	`info`		Informational messages.
7	Debug	`debug`		Debug-level messages. Messages that contain information normally of use only when debugging a program.[9]

See also: Syslog on Wikipedia

Logging architecture

- Typically: servers funnel logs to a central server
- Typically: syslog

See also: Top 5 mistakes in logging

Logging laws

- Something you should think about
  - Just like with databases

1) Can you log the data that you are logging?

2) Can you retrace the decision process (EU law)?

See also: Top 5 mistakes in logging

Prometheus and logging

- Prometheus is a time-series key-values storage
- Stores metrics: “process_load 2.3”
- Not a logging system
- Link: Why Grafana is good at metrics and not logs

ELK stack

- Elasticsearch
  - Search engine (based on Lucene)
- Logstash
  - Logging parser
- Kibana
  - Logging frontend / dashboard

See also: elastic.co, Example: Spring boot logs

So… What can we use it for?

- Purpose:
  - Understand activity (preliminary examinations)
  - Diagnosis (of an actual problem)
  - Audit trails

- We now have the infrastructure to log
  - So we can get an overview of a system (understanding)
  - With Elasticsearch you can search for problems (diagnosis)
  - What about auditing?

Auditing

- An audit trail is a security-relevant chronological set of records that provide evidence of the sequence of activities that have affected a system at any time

- … So what?

- Example: You own a car manufacturing plant
  - Your operating system is hacked. You are now producing killer cars
  - Half of your staff is slaughtered. Bad business. How do you avoid this?

See also: Audit trail on Wikipedia

Auditing

- An audit trail is a security-relevant chronological set of records that provide evidence of the sequence of activities that have affected a system at any time
- Logs can be used retroactively as proof
- Same argument as with the SLA:
  - You need to document your critical assets
  - Example: root logins, bank transactions, permission changes etc.

See also: Audit trail on Wikipedia

Auditing

- Exists to give you evidence of actions
- In your system, you should log
  - authentications, privilege escalation
  - CRUD operations (transactions)
  - permission changes
- You should not be able to ‘disable’ audit logs!

See also: Audit trail on Wikipedia

Post-mortem analysis

- Eventually things do go wrong. Inevitably
  - But that's ok, just fail fast
- Only strategy: learn from your failures
  - If you do you'll learn and grow and be smarter
  - If you don't, you will have fewer clients to worry about
- Analyse the problem *after* it happened
  - Hence *post-mortem* analysis

Post-mortem analysis

- Summary
  - On Monday, 11 April, 2016, Google Compute Engine instances in all regions lost external connectivity for a total of 18 minutes, from 19:09 to 19:27 Pacific Time.
- Detailed description
  - … inbound internet traffic was not routed correctly …
- Root cause
  - </tech rant>
- The fix
  - … decided to revert the most recent configuration changes made…
- Lessons learned
  - … There are a number of lessons to be learned from this event …

See also: Google post mortem analysis

Post-mortem analysis

- Summary
  - Include scope/affected users, time-stamp and timezone.
- Detailed description
  - Be brutally honest
- Root cause
- The fix
  - This is where you need your backup strategy (in two weeks)
- Lessons learned
  - How can this be avoided?

See also: Yet another postmortem

Next hand-in

1) Implement logging in your system
2) Implement alarms in Grafana
3) Crash the system **at a random point in time**
    1) Be creative – you don't have to crash it all
4) Wait for your ops group to discover the outing and resolve the issue together with them

5) Hand-in: Post-mortem report (be brutally honest)