Grafana Loki

How to make sense of your logs

whoami

- Alexander Knipping, IT Systems Engineer @ noris network
  - Obeservability
  - Cloud Native Adoption
- ❤️ Open Source
  - Contributor to Kubernetes (wg-component-standard) and M3DB
  - https://github.com/obitech
- Co-Host of this Meetup

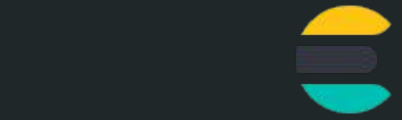
KCD 2020

- June/July 2020 in Munich
- Would you like to be part...
  - ... as a sponsor?
  - ... as a speaker?

Agenda

Grafana Loki: “Prometheus but for Logs”

1. Overview ElasticSearch
2. Overview Prometheus
3. Overview Loki
4. Demo



Elasticsearch

Elasticsearch: Overview

- OpenCore, distributed search- and analytics engine based on Apache Lucene
- Released in 02/2009 by Elastic
- Used to save & analyse text documents
- HTTP REST API
- Often deployed as part of the Elastic- or ELK-Stack

Elasticsearch: Inverted Index

Three documents:

1 ➜ {“hello”, “alex”}

2➜ {“my”, “name”}

3➜ {“is”, “alex”}

Inverted index:

“hello” ➜ [1:0]

“alex” ➜ [1:0, 3:1]

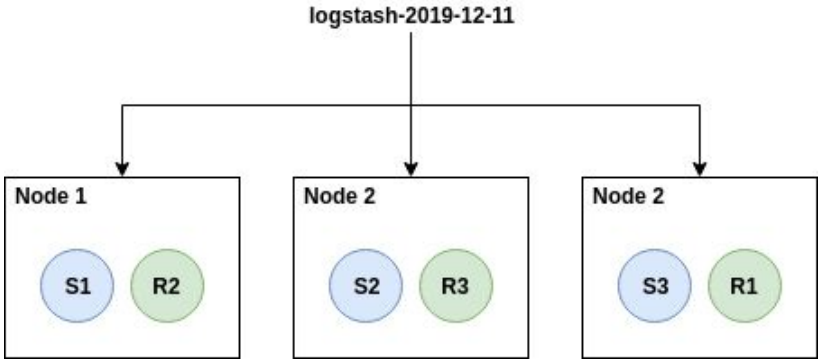
“my” ➜ [2:0]

...

- Stemming: rainfall ➜ rain
- Synonyms: Pen, Pens ➜ Pen

Elasticsearch: Sharding

- Split up index over several machines (shards)
- Logs: usually one index per day: **logstash-2019-12-11**
- Given **logstash-2019-12-11** has 150 log entries, 3 shards and replication factor 2:



Elasticsearch: Summary

What we get:

- Ability to index…
  - … big amount of documents (logs)
  - … in a fine grained way
- Ability to ask elaborate questions
  - Show me the Top 10 Users by amount paid aggregated across countries during the past two months

We pay for it with:

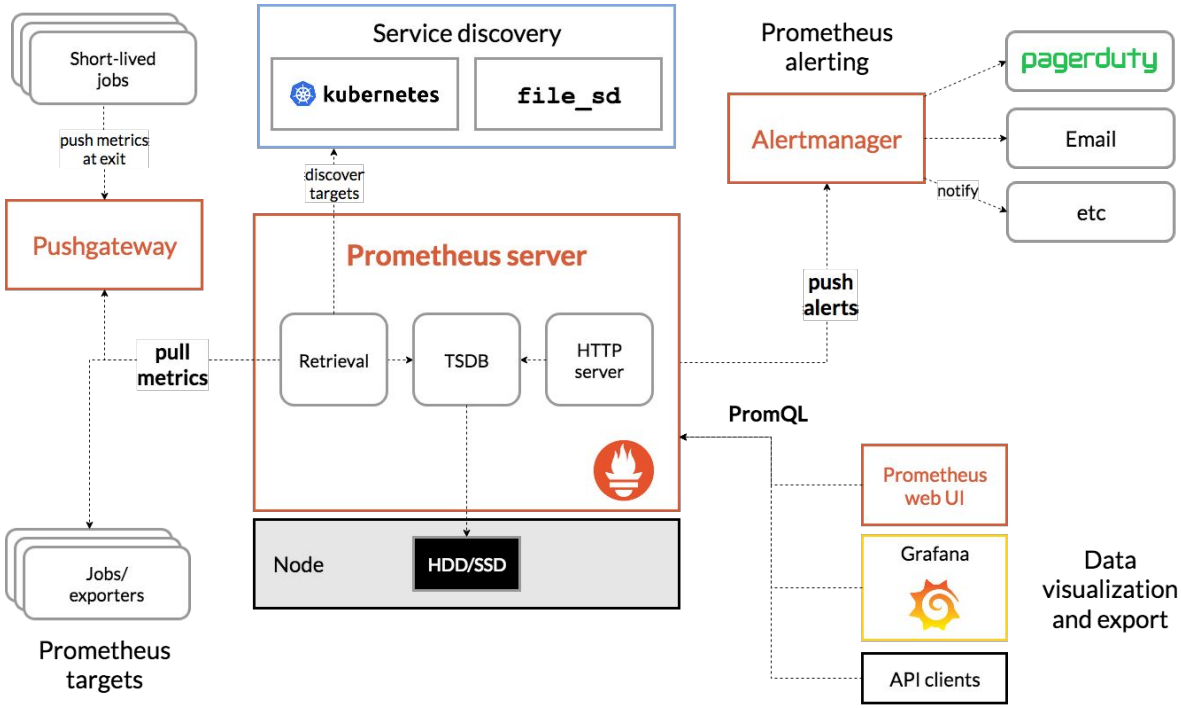
- Resource intensity
- Operational complexity

Prometheus

Prometheus: Overview

- Metric-base white-box monitoring & alerting system
- CNCF Graduated project, started at SoundCloud in 2012
- Multidimensional data model
- Simple to operate
- Optimized for dynamic (cloud) environments

Prometheus: Architecture

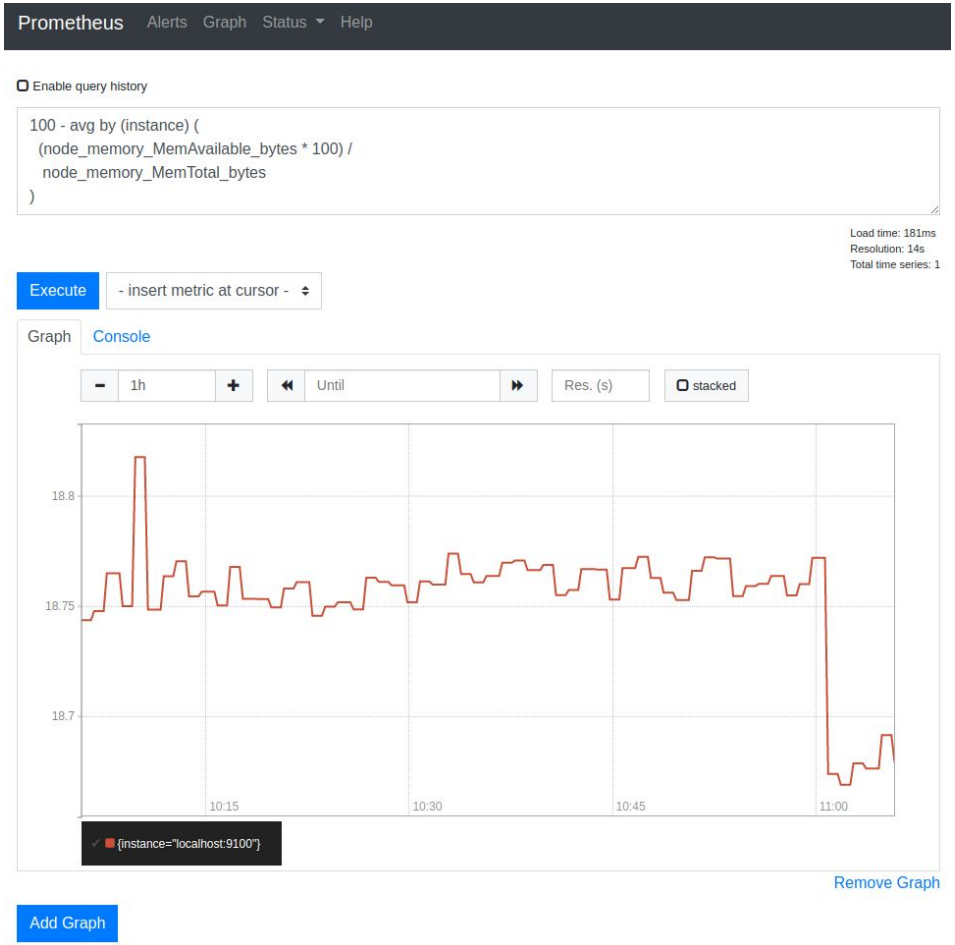


Prometheus: TSDB

```
prometheus_sd_discovered_targets{config="grafana",name="scrape"} 1
prometheus_sd_discovered_targets{config="node-exporter",name="scrape"} 1
prometheus_sd_discovered_targets{config="prometheus",name="scrape"} 1
prometheus_sd_discovered_targets{config="telegraf",name="scrape"} 1
```

- config=”grafana”
  - ➔ [000001]
- config=”node-exporter”
  - ➔ [000002]
- name=”scrape”
  - ➔ [000001, 000002]

```
./data
├── b-000001
│   ├── chunks
│   │   ├── 000001
│   │   └── 000002
│   ├── index
│   └── meta.json
└── b-000002
    ├── meta.json
    └── wal
        ├── 000001
        ├── 000002
        └── 000003
```


Prometheus
Alerts
Graph
Status
Help
Enable query history
100 - avg by (instance) (
(node_memory_MemAvailable_bytes * 100) /
node_memory_MemTotal_bytes
)
Load time: 181ms
Resolution: 14s
Total time series: 1
Execute
- insert metric at cursor -
Graph
Console
1h
Until
Res. (s)
stacked
18.8
18.75
18.7
10:15
10:30
10:45
11:00
{instance="localhost:9100"}
Remove Graph
Add Graph

Prometheus: Cardinality Explosion

- Cardinality is multiplicative
- Total number of time series = metric names * cardinality of labels * targets
- Prometheus can handle a couple 100k time series

Multiplicative Cardinality

You have a histogram, *http_request_duration_seconds_bucket*

100 instances * 10 buckets -> 1000 series
* 10 endpoints -> 10,000 series
* 10 response codes -> 100,000 series (maximum recommended cardinality)
* 4 http methods -> 400,000 series
* 100 tenants -> 40,000,000 series ☠☠☠

splunk> turn data into doing

https://promcon.io/2019-munich/slides/containing-your-cardinality.pdf

Prometheus: Summary

What we get:

- Low operational cost
  - Simplicity
  - Robustness
- High throughput
  - ~1,5m samples/s
  - ~500k time series
- Multidimensional data model
- Rich query language

We pay for it with:

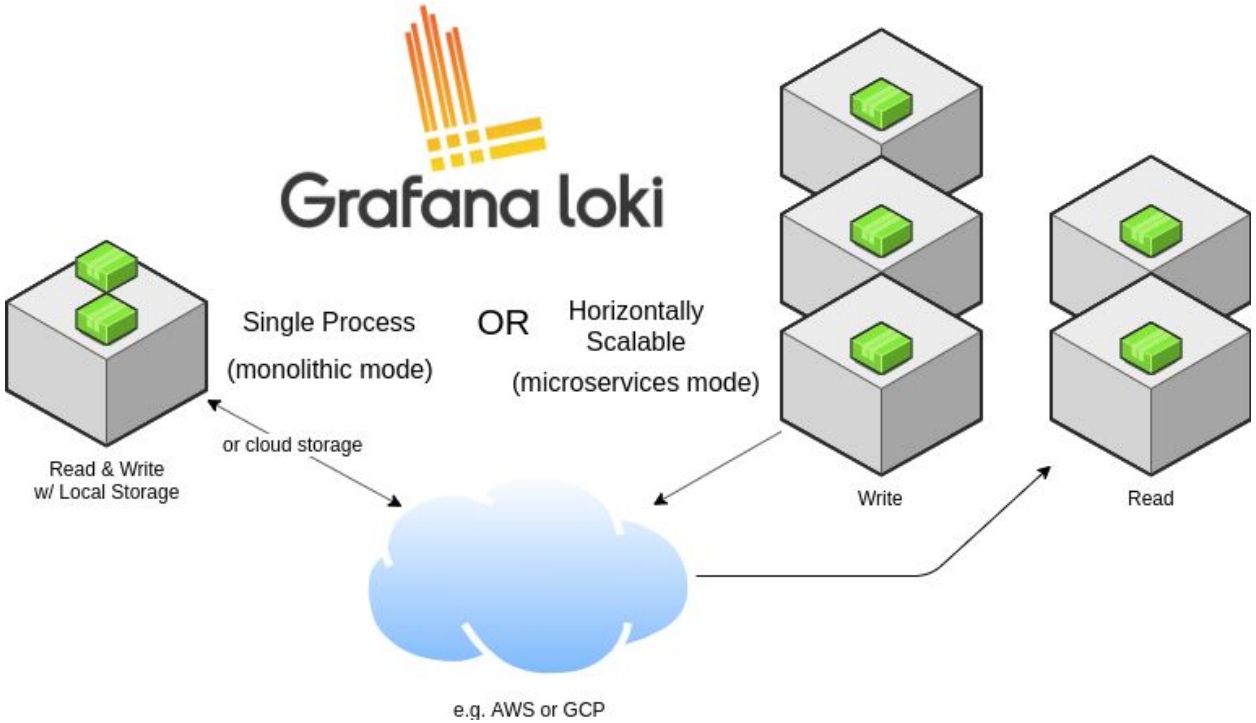
- Single Server only
  - Label cardinality is limiting factor
- No built-in long term storage

Grafana Loki

Grafana Loki: Overview

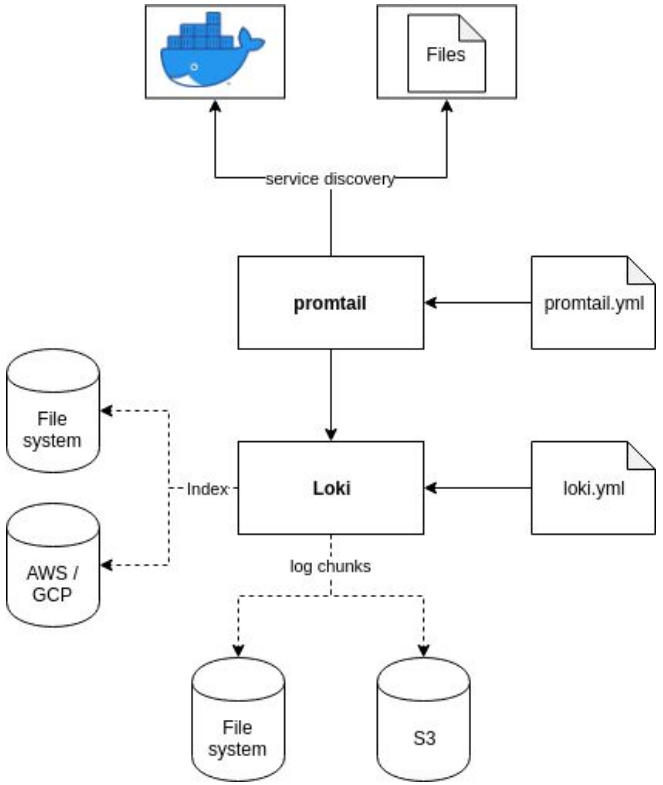
- Started 2017, v1.0 since 12/2019
- “Prometheus but for Logs”
  - Complexity of Elasticsearch often not needed for operational log aggregation
  - Index only metadata (labels) of logs
- Column store for indices, S3 for chunks
- Operationally simple
  - Single process & microservices mode
- First-class Grafana integration
  - Less context switches

Grafana Loki: Architecture



Grafana Loki

- Promtail discovers logs, adds labels and sends them to Loki
- Usage of Prometheus libraries for service discovery and label relabeling
- In Loki, log streams are batched up into chunks, compressed, then sent to chunk store
- Indices (labels) are sent to column store
- On read, chunks are streamed out via gRPC, deduplicated and sent out via HTTP/1



Demo

https://github.com/noris-network/loki-demo

Grafana Loki: Summary

What we get:

- Simplicity
- Scalability
- Cost effectiveness
- Less context switches between metrics and logs

We pay for it with:

- Restricted by label cardinality
- At some point we need to use cloud services...
  - ... or run our own Cassandra & S3

Thank You!

Demo: https://github.com/noris-network/loki-demo

Loki: https://github.com/grafana/loki

Loki Design Doc: on Google docs