FLEX - Monitoring and Logging

Simon Härer, Michael Sperber

Created: 2024-06-10 Mon 07:49

@ctive group

Monitoring & Logging

What is going on with the system?

Stakeholder

- Operations
- Developer
- Business

Logging

Logging is the act of capturing events and translate them in a textual form so that a reader later can reconstruct a projection of history.

- ″received request: ...″
- ″sent response: ...″
- ″authentication succeeded″

Monitoring

As opposed to logging, monitoring is about the **state** of the system **right now**.

- "service alive"
- "5 instances of this service are up"
- "host is down"

Metrics Collection

A metric is aggregated quantifiable data that characterizes system operation over a period of time.

- ″67% system load″
- ″0.5s response time″
- ″handling 1000 requests/s″

Pre-Aggregation

- compute in system, rather than from events later
- recording all events, then aggregation often costly
- information loss

Probing vs. Instrumentation

- **probing**: collect metrics from *outside*, by sampling
- **instrumentation** calculate metrics from *inside*, by measurement

Taxonomy

service	purpose
Logging	**events**, post-mortem problem analysis
Monitoring	system **state** for live problem intervention
Metrics	collect performance characteristics

All three are transmitted as **messages**. Hence, substantial overlap in technical solutions.

Why are logging and monitoring important for microservices?

- complex communication patterns
- complex failure scenarios
- usually only accessible trace of system history
- many services involved

Operations in Architecture Concept

- Monitoring is important to maintain overall system uptime; otherwise failures may go unnoticed until the problems have cascaded.
- Logging is crucial to detect properties of and problems with the system that are only apparent over a greater period of time.

Centralized or Distributed

- Centralized logging is good for cross-microservice tracing or to detect data inconsistencies and error sources
- On the other hand, centralized logging introduces coupling. Teams *have to* use a logging interface and infrastructure. Changes in this interface require all teams to change their code.

Architectural Requirements

- The logging/monitoring service needs to be highly available, typically via replication.
- Logging/monitoring (in particular when operational problems with them) should not interfere with service operation.
- The logging/monitoring service needs to be network-accessible from the service.
- Alternative: Log events locally, transmit asynchronously to service service
- Metrics needs to be aggregated within the software.

System Resources

- Throughput needs to excess log/monitoring inflow.
- Storage needs to be large enough; data needs to be regularly thinned out and/or archived.
- This means the logging/monitoring service itself needs to be monitored.

Functional and Technical Data

Functional:

- "The item has left the building"
- "The tool has failed."
- "200 km/h"

Technical:

- "Host not found"
- "Storage exceeded"
- "Load average"

Data Selection is Essential

Dealing with log data requires sophisticated selection and indexing.

Business Metrics

- ″Conversion rate is 0.5″
- ″Average time spent on web site is 2 minutes″
- ″Volume sold today is $50.000″

Application Metrics

- ″5.000 requests per second″
- ″Average page build time is 0.5s″
- ″Average message transit time is 100ms″

System Metrics

- "Load average is 0.5"
- "Network throughput is 100MB/s"
- "Average # of open file descriptors is 100"
- "Disk is 68% full"

Metric Pipeline

- capture directly or sample & collect
- transmit to monitoring and/or logging system
- store in time-series database
- query
- visualize

Architectural Impact of Metrics Collection

- Performance and memory overhead
- Hidden non-functional dependencies in code
- Potential for additional errors and/or crashes
- Collection of metrics might affect the metrics

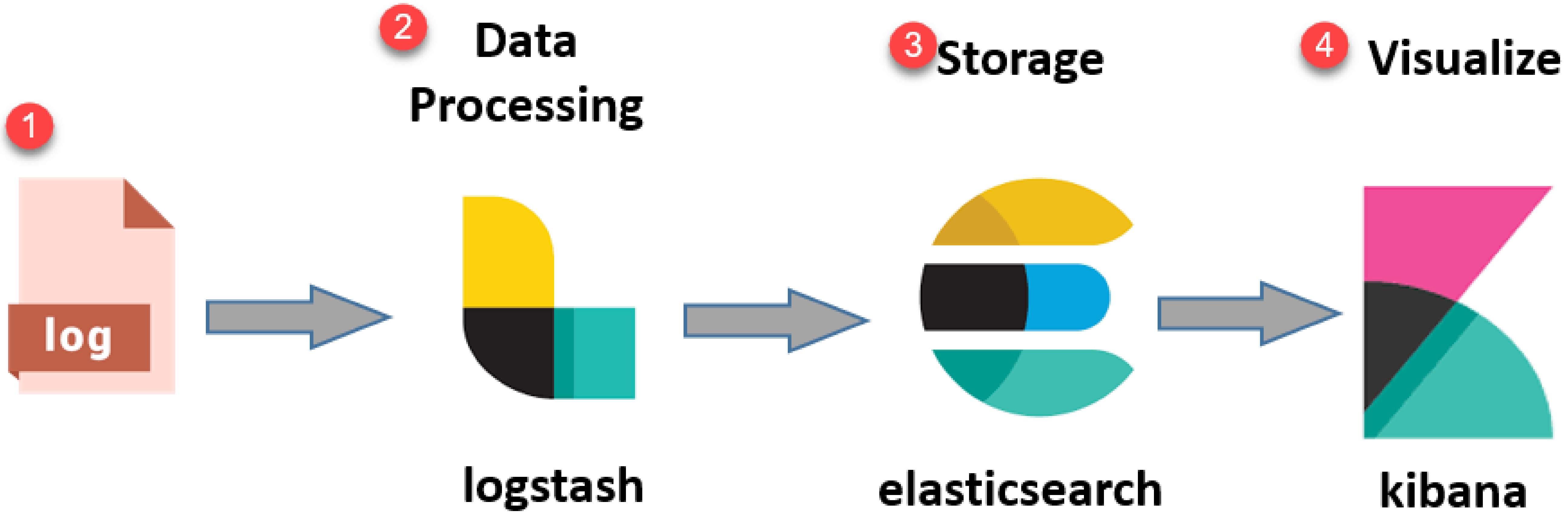
Correlation IDs

Often one request is executed by a chain of microservice calls. We have to identify the request and its successor-requests to debug the system. A solution for that are so called correlation IDs. That is, an id that is passed around with the requests and logged, every time something within the request chain happens.

There are different technologies that support correlation IDs:

- trace (Java) helps to implement the transmission containing a correlation id
- Multiple log framework implement support for a context with meta information that is logged with each message

ELK Stack


1
log
2
Data Processing
logstash
3
Storage
elasticsearch
4
Visualize
kibana

Logstash

Parses logfiles and makes them collectible from servers. Logstash can read, parse and transform logfiles from multiple sources and write them into sinks.

Elasticsearch / OpenSearch

Stores log files and makes them searchable. OpenSearch provides full-text search, and can search and filter fields in structured data.

Kibana / OpenSearch Dashboards

Web frontend for interactive data analysis, can store and run queries, and display visualizations.

Graylog

Graylog also uses Elasticsearch / OpenSearch to store and search log data. Metainformation is stored in a MongoDB database.

- Graylog uses the Graylog Extended Log Format (GELF) for log messages. GELF is meant to standardize web-transmitted data.
- Many libraries and languages support the GELF format
- Web frontend for log analysis

Time-Series Database

Specialized database for monitoring and metrics data:

- timestamp + key/value pairs
- horizontally scalable
- support downsampling of historical data
- allow live streaming
- alerting functionality

InfluxDB

- push-based
- custom query language
- analysis DSL

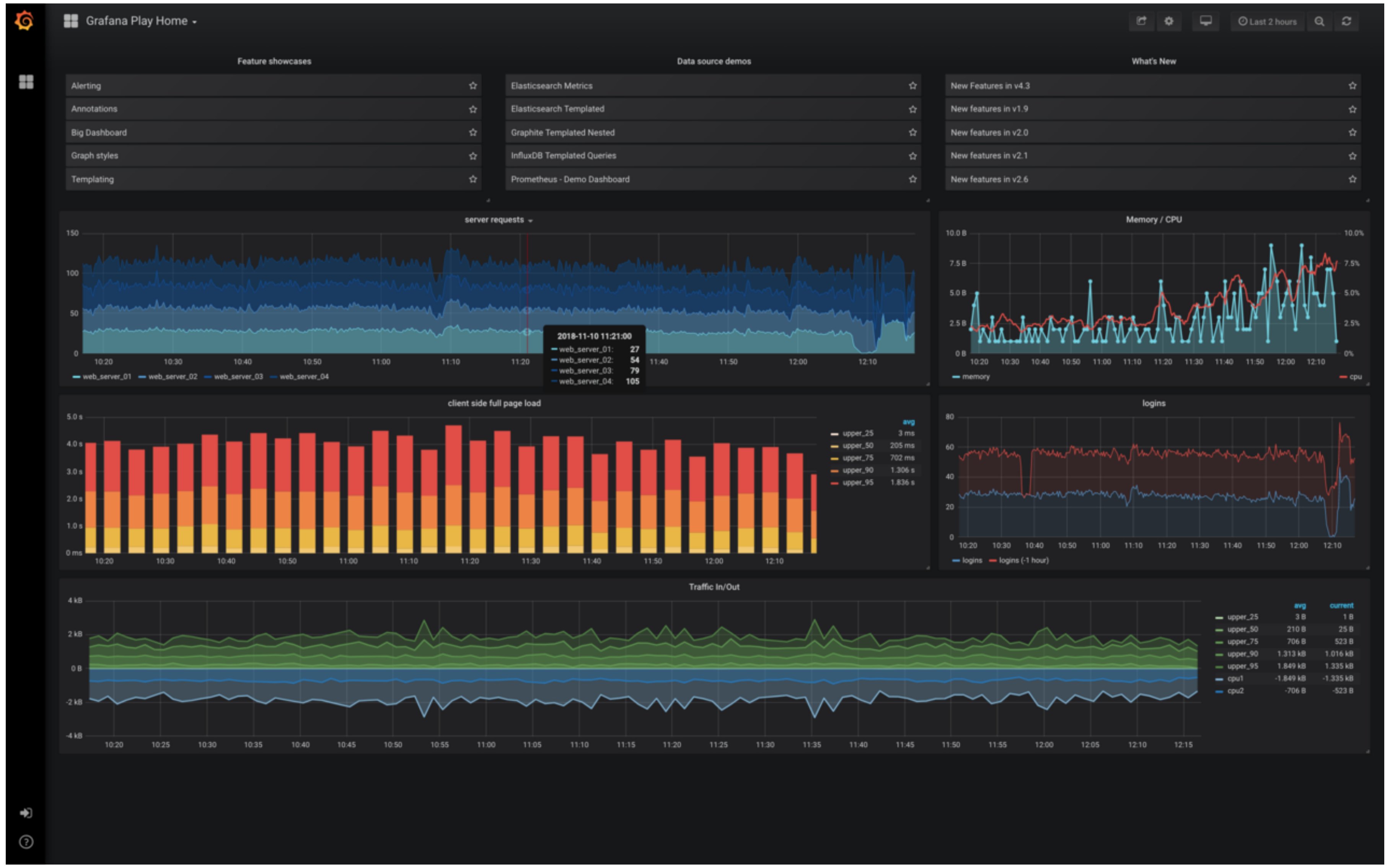
Prometheus

- pull-based
- custom query language

Grafana

- visualization of monitoring data and metrics
- can connect to different data sources
- configurable dashboards
- alerting functionality

Grafana


Grafana Play Home
Last 2 hours
Feature showcases
Alerting
Annotations
Big Dashboard
Graph styles
Templating
Data source demos
Elasticsearch Metrics
Elasticsearch Templated
Graphite Templated Nested
InfluxDB Templated Queries
Prometheus - Demo Dashboard
What's New
New Features in v4.3
New features in v1.9
New features in v2.0
New features in v2.1
New features in v2.6
server requests
150
100
50
0
10:20
10:30
10:40
10:50
11:00
11:10
11:20
11:40
11:50
12:00
12:10
web_server_01
web_server_02
web_server_03
web_server_04
2018-11-10 11:21:00
web_server_01: 27
web_server_02: 54
web_server_03: 79
web_server_04: 105
Memory / CPU
10.0 B
7.5 B
5.0 B
2.5 B
0 B
10.0%
7.5%
5.0%
2.5%
0%
memory
cpu
client side full page load
5.0 s
4.0 s
3.0 s
2.0 s
1.0 s
0 ms
avg
upper_25 3 ms
upper_50 205 ms
upper_75 702 ms
upper_90 1.306 s
upper_95 1.836 s
logins
80
60
40
20
0
logins
logins (-1 hour)
Traffic In/Out
4 kB
2 kB
0 B
-2 kB
-4 kB
10:20
10:25
10:30
10:35
10:40
10:45
10:50
10:55
11:00
11:05
11:10
11:15
11:20
11:25
11:30
11:35
11:40
11:45
11:50
11:55
12:00
12:05
12:10
12:15
avg current
upper_25 3 B 1 B
upper_50 210 B 25 B
upper_75 706 B 523 B
upper_90 1.313 kB 1.016 kB
upper_95 1.849 kB 1.335 kB
cpu1 -1.849 kB -1.335 kB
cpu2 -706 B -523 B

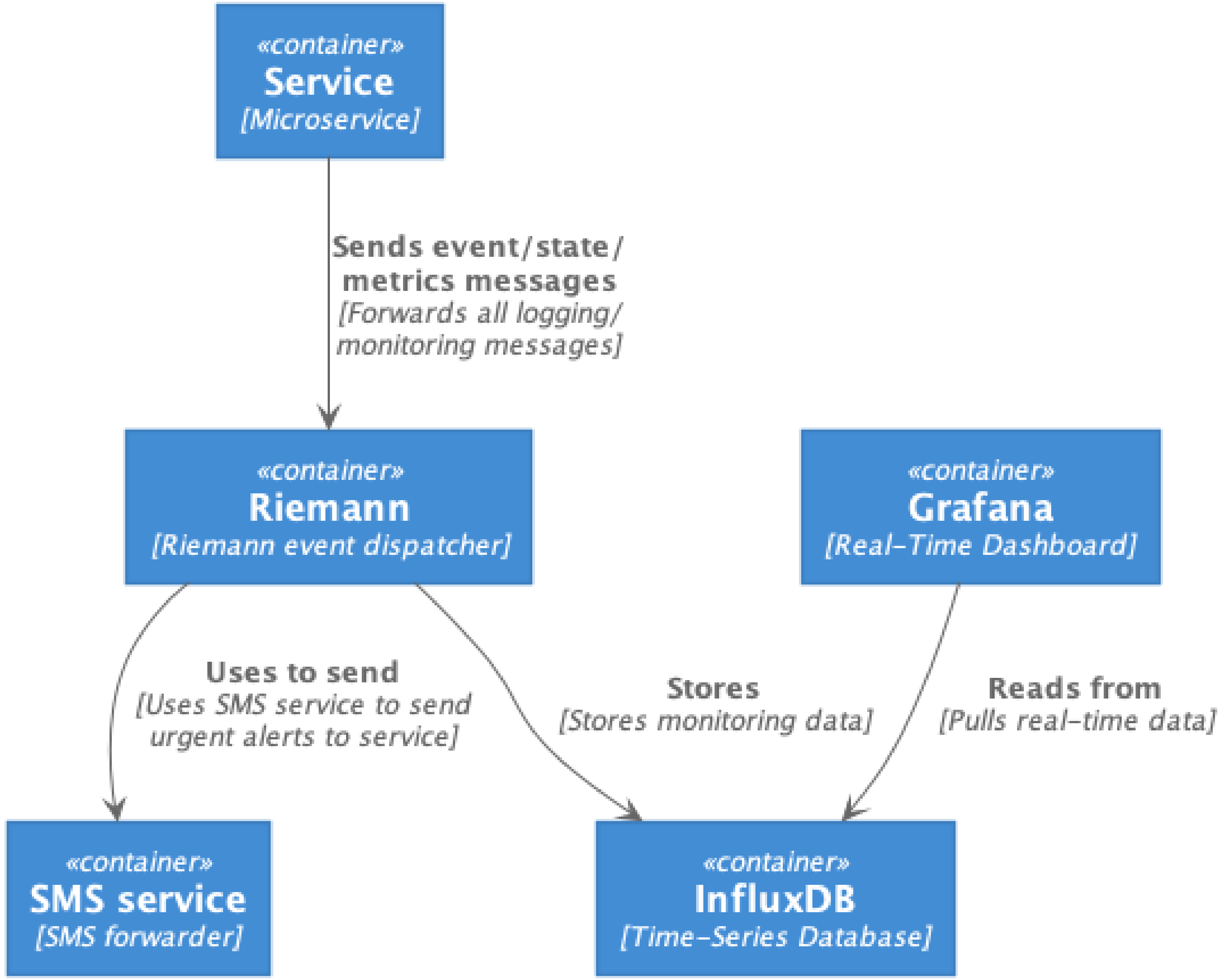
Graphite

- push-based
- graph rendering
- alterting dashboard: Seyren

Riemann

- receives and aggregates events
- tracks service state
- programmable alerts
- Clojure-based embedded DSL
- can forward to various time-series databases

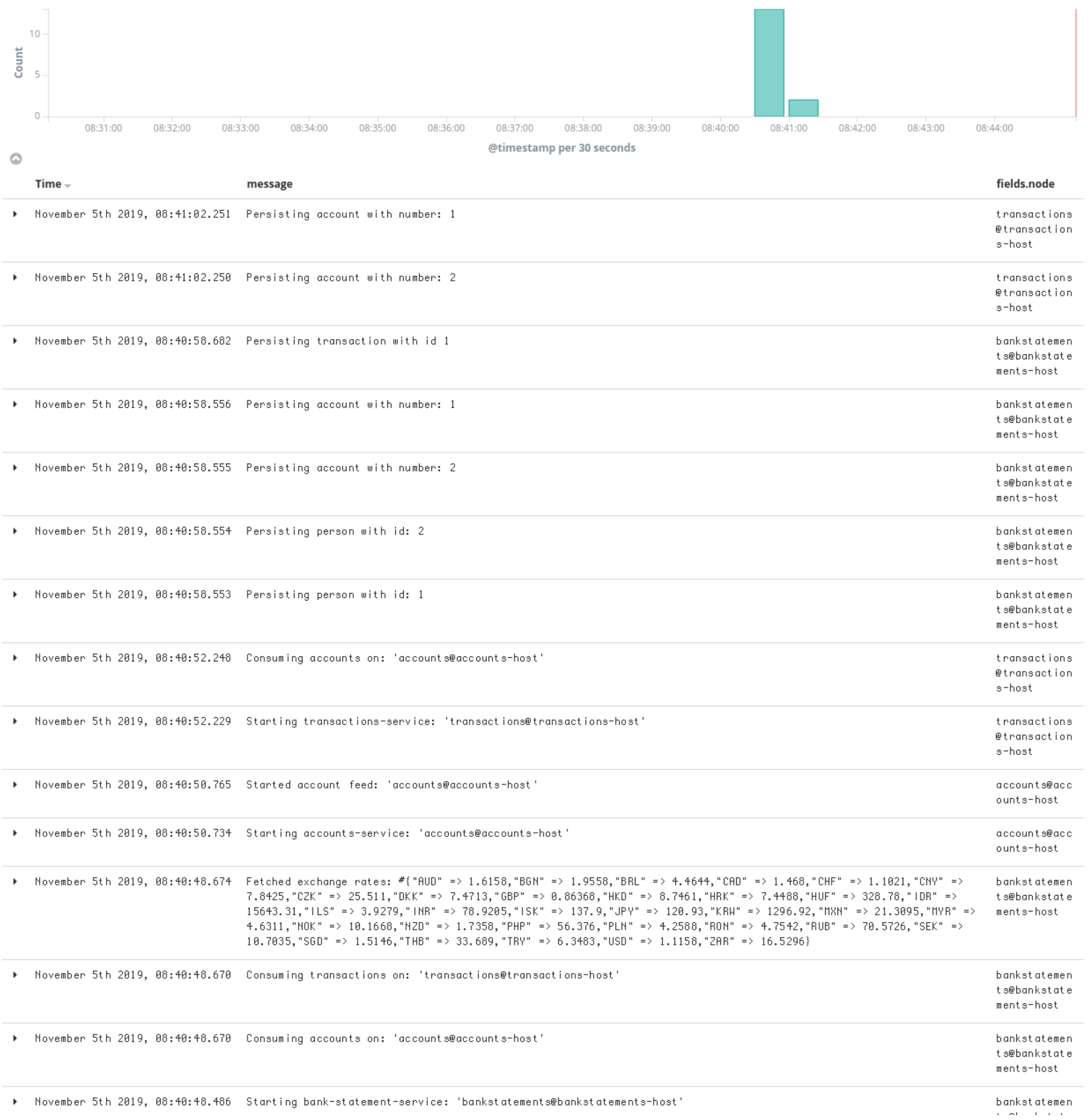
Example Monitoring Setup



Nagios

- mainly for systems monitoring
- various forks: checkmk, Icinga

Erlbank: Kibana / OpenSearch Dashboards



Time	message	fields.node
November 5th 2019, 08:41:02.251	Persisting account with number: 1	transactions@transactions-host
November 5th 2019, 08:41:02.250	Persisting account with number: 2	transactions@transactions-host
November 5th 2019, 08:40:58.682	Persisting transaction with id 1	bankstatements@bankstatements-host
November 5th 2019, 08:40:58.556	Persisting account with number: 1	bankstatements@bankstatements-host
November 5th 2019, 08:40:58.555	Persisting account with number: 2	bankstatements@bankstatements-host
November 5th 2019, 08:40:58.554	Persisting person with id: 2	bankstatements@bankstatements-host
November 5th 2019, 08:40:58.553	Persisting person with id: 1	bankstatements@bankstatements-host
November 5th 2019, 08:40:52.248	Consuming accounts on: 'accounts@accounts-host'	transactions@transactions-host
November 5th 2019, 08:40:52.229	Starting transactions-service: 'transactions@transactions-host'	transactions@transactions-host
November 5th 2019, 08:40:50.765	Started account feed: 'accounts@accounts-host'	accounts@accounts-host
November 5th 2019, 08:40:50.734	Starting accounts-service: 'accounts@accounts-host'	accounts@accounts-host
November 5th 2019, 08:40:48.674	Fetched exchange rates: #{"AUD" => 1.6158,"BGN" => 1.9558,"BRL" => 4.4644,"CAD" => 1.468,"CHF" => 1.1021,"CNY" => 7.8425,"CZK" => 25.511,"DKK" => 7.4713,"GBP" => 0.86368,"HKD" => 8.7461,"HRK" => 7.4488,"HUF" => 328.78,"IDR" => 15643.31,"ILS" => 3.9279,"INR" => 78.9205,"ISK" => 137.9,"JPY" => 120.93,"KRW" => 1296.92,"MXN" => 21.3095,"MYR" => 4.6311,"NOK" => 10.1668,"NZD" => 1.7358,"PHP" => 56.376,"PLN" => 4.2588,"RON" => 4.7542,"RUB" => 70.5726,"SEK" => 10.7035,"SGD" => 1.5146,"THB" => 33.689,"TRY" => 6.3483,"USD" => 1.1158,"ZAR" => 16.5296}	bankstatements@bankstatements-host
November 5th 2019, 08:40:48.670	Consuming transactions on: 'transactions@transactions-host'	bankstatements@bankstatements-host
November 5th 2019, 08:40:48.670	Consuming accounts on: 'accounts@accounts-host'	bankstatements@bankstatements-host
November 5th 2019, 08:40:48.486	Starting bank-statement-service: 'bankstatements@bankstatements-host'	bankstatemen