

FLEX - Monitoring and Logging

Simon Härer, Michael Sperber

Created: 2023-04-25 Tue 08:07

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 of 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

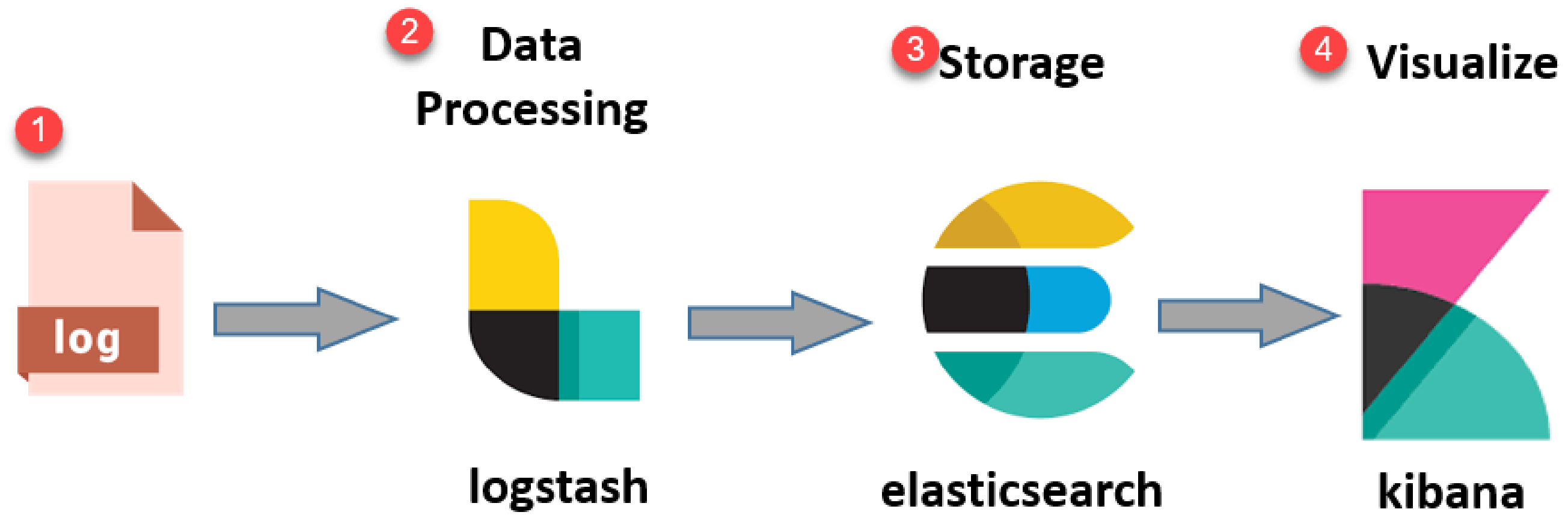
Correlation IDs

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



Logstash

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

Elasticsearch

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

Kibana

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

Graylog

Graylog also uses Elasticsearch 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



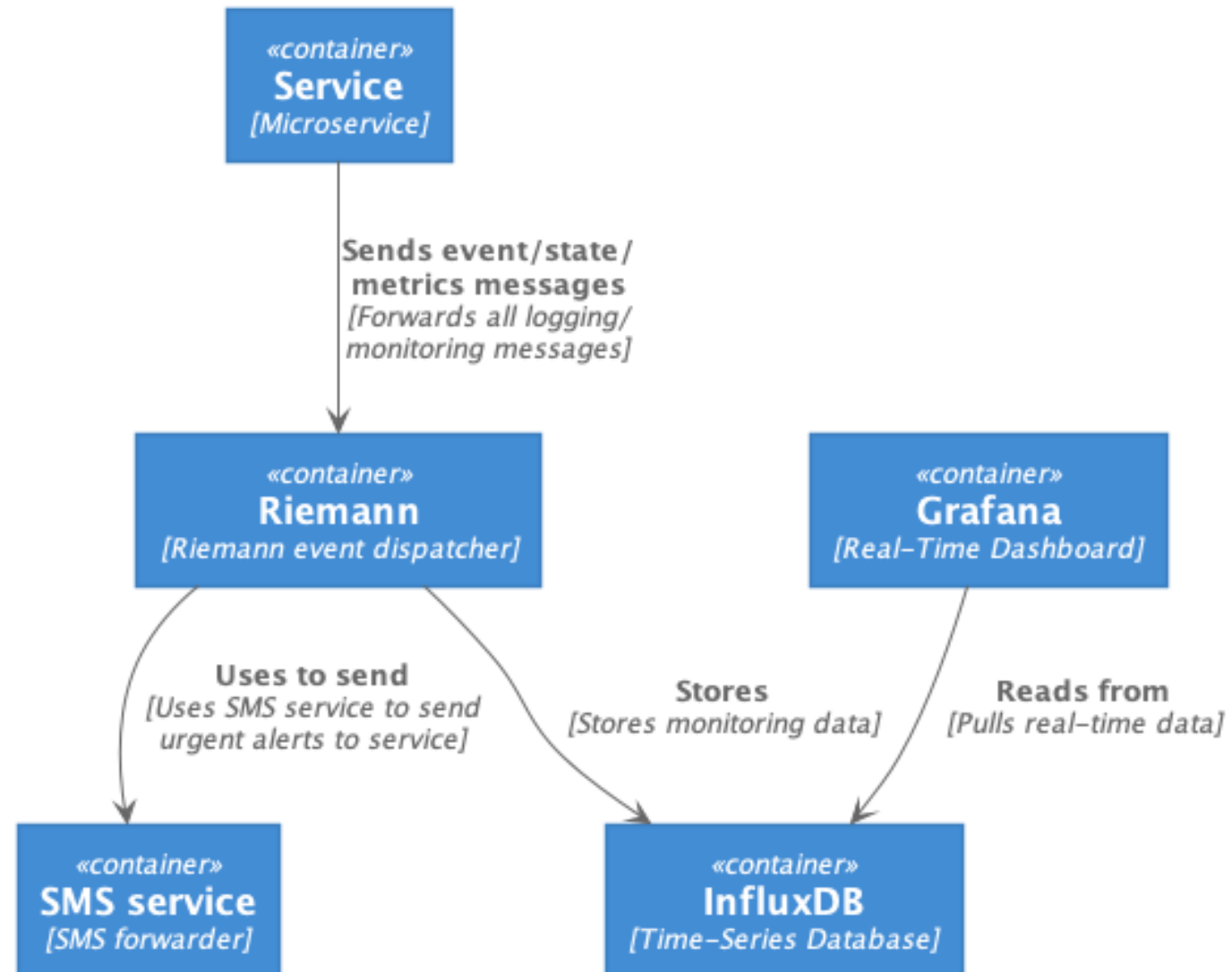
Graphite

- push-based
- graph rendering
- alerting 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

