Advanced Use Cases for Observability

Piotr Gaczkowski
IT CONSULTANT

@doomhammerng doomhammer.info

Overview

- Advanced use of the Unified Logging Layer
- Combining Fluentd with external services
- Enriching events with additional data
- Event Sourcing

Advanced Use Cases for Logging

Processing logs inside the Unified Logging Layer

Filtering log data

Enriching log data

Writing logs to different destinations

Creating actionable audit trails (Event Sourcing)

Processing Logs

Log formats vary between applications

We can store them in original format...

...or we can parse them storing data as separate fields

Unparsed Logs

```
127.0.0.1 192.168.0.1 - [28/Feb/2013:12:00:00 +0900] "GET / HTTP/1.1" 200
777 "-" "Opera/12.0" -
```

Parsed Logs

```
127.0.0.1 192.168.0.1 - [28/Feb/2013:12:00:00 +0900] "GET / HTTP/1.1" 200
777 "-" "Opera/12.0" -
```

```
{
  "remote": "127.0.0.1",
  "host": "192.168.0.1",
  "user": "-",
  "method": "GET",
  "path": "/",
  "code": "200",
  "size": "777",
  "referer": "-",
  "agent" : "Opera/12.0",
  "http_x_forwarded_for": "-"
}
```

Formats Fluentd Understands Natively

- Apache 2
- Apache error log
- Nginx
- Syslog
- LTSV/CSV/TSV
- JSON
- Regular expressions

Parsing JSON Logs

```
<filter kubernetes.var.log.containers.workout-gateway**workout-gateway**>
  @type parser
  key_name log
  reserve_data true
  <parse>
    @type json
  </parse>
</filter>
```

Filtering Logs

Not every destination needs to contain all the logs

We may want to exclude some noise

Removing duplicates and redundant data

Filtering Logs

```
<filter kubernetes.var.log.containers.run-controller**>
  @type grep
  <regexp>
    key log
    pattern Persisted workout
  </regexp>
</filter>
```

Enriching Logs

Adding GeoIP data

Including performance metrics

Performing lookups or correlations during processing

Demo

Enrich logs with geographical location

Writing Logs to Different Destinations

Satisfying compliance

Archival

Integrating external services

Federation

Demo

Deploy Datadog Agent

Configure Fluentd to send logs also to Datadog

Command Query Separation

Commands perform an action

Queries return data

No method should do both

Command Query Responsibility Segregation

Separate models for commands and queries

Command and query interfaces implemented as different objects

Fits with event-based programming models

CQRS is often tied with domain-driven design

Event Sourcing

Allows to replay the system state

Events are stored in a database (Event Store)

Ability to trace errors from the past

Actionable audit trail

Summary

Unified Logging Layer can be used to decouple log collection from log aggregation

To get a better picture of each operation we can modify and enrich our logs after they are emitted

Advanced forms of observability may require design changes

Thank you!

@doomhammerng

doomhammer.info