Mailtracing mit ELK

Was ist Mailtracing?

- Suche nach einem Absender und/oder Empfänger
- Wurde eine bestimmte Mail auf dem eigenen System zugestellt?
- Wurde eine bestimmte Mail versendet und dem Empfänger-System übergeben?
- Mail-ID, Timestamp

Themen Vortrag

- Kurze Beschreibung ELK
- Sicherer Aufbau (Netzwerk)
- Voraussetzungen
- Anbindung Logs
- Logstash Übergabe/Verarbeitung
- Grok-Patterns (Aggregierung!)
- ELK-RestAPI
- Zugriff über Python, Beispiel Script
- Ergebnisse extrahieren mit Kibana
- ~~Kein Thema: Open-Source-Lizenz Elasticsearch~~

Wenn noch Zeit:

- Sizing: Anzahl Shards in den Index-Templates
- Index-Templates generell
- Life-Cycle-Policy

Was ist ELK?

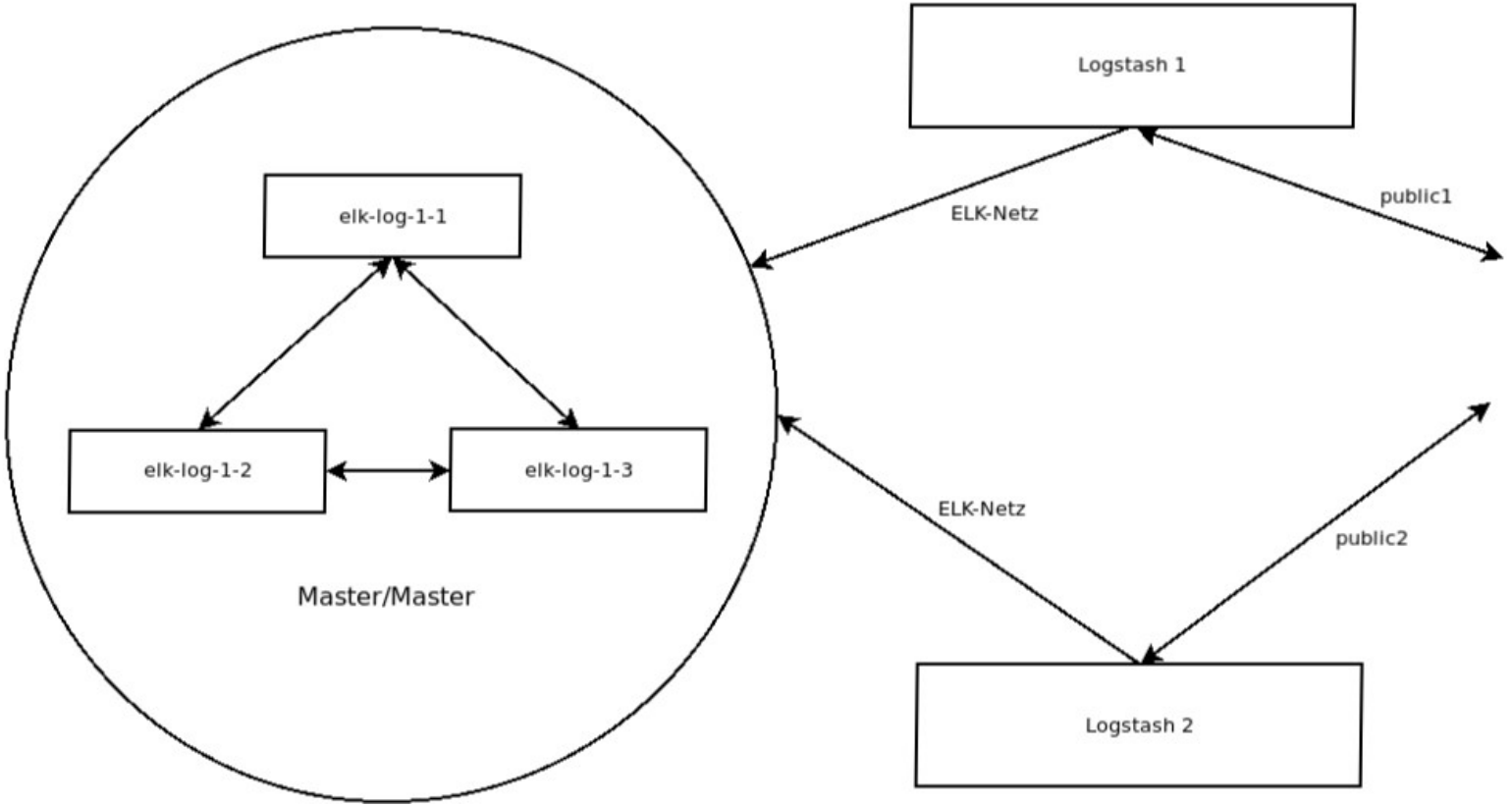
- Indizierende DB für „weniger“ strukturierte Daten (Dokumente)
- REST-API (auch Kibana)
- Skalierbarkeit für Performance und HA
- Flexible Suchfunktionen
- Grok-Filter Logstash
- (Sicheres) System für zentrales Logging möglich (begrenzte Zeit speichern und durchsuchen)

Anwendungen ELK
Der ELK-Stack

- Elasticsearch (auf mehreren Nodes)

  Aufteilung der Last und HA durch Sharding
- Logstash (Filebeat, Pipelines)
- Kibana (besser auf einem eigenen Node)

Systemskizze



Voraussetzungen ELK

- Java: https://www.elastic.co/de/support/matrix#matrix_jvm
- Elasticsearch 7.10: OpenJDK-11 / AdoptOpenJDK 1.8.0
- **Elastic supports some OpenJDK-derived distributions: 1. builds by the IcedTea Project; 2. those produced by OS vendors in the “Product and Operating System” matrix which have passed the TCK tests; 3.
- *Manueller Symlink in Gentoo: /usr/share/elasticsearch/jdk/bin → /opt/openjdk-bin-11.0.10_p9/bin/java*

Logstash

- Was macht Logstash?
- Serielles Abarbeiten von Konfiguartions-Files
- Darüber wird eine Pipeline definiert
- Mehrere Pipelines möglich
- Input → Verarbeitung → Output

Anschluss Logstash

- Logs gehen über Logstash
- UDP-Port
- Markierung mit Type: hier *mail*

/etc/logstash/conf.d/10-input-mail.conf

```
input {
  udp {
      port => 5000
      type => mail
      codec => plain {
          charset => "ISO-8859-1"
          }
      }
}
```

Anschluss Postfix-Logs

- Rsyslog-Konfiguration auf Postfix-Host

  /etc/rsyslog.d/logstash.conf
- mail.* @logstash-host:5000
- *To forward it via plain tcp, prepend two at signs ("@@").*

- R in Rsyslog?

  https://rainer.gerhards.net/

Was ist Grok?

„When you claim to "grok" some knowledge or technique, you are asserting that you have not merely learned it in a detached instrumental way but that it has become part of you, part of your identity. For example, to say that you "know" Lisp is simply to assert that you can code in it if necessary — but to say you "grok" LISP is to claim that you have deeply entered the world-view and spirit of the language, with the implication that it has transformed your view of programming.“

(https://en.wikipedia.org/wiki/Grok#Adoption_and_modern_usage)

Logstash Grok-Patterns

Grok-Patterns sind Regexes. Auch bekannte Regexes sind möglich wie:

.* oder [0-9] etc.

- POSTFIX_QUEUEID ([0-9A-F]{6,}|[0-9a-zA-Z]{15,}|NOQUEUE)
- POSTFIX_ACTION (reject|defer|accept|header-redirect)
- POSTFIX_SMTP %{POSTFIX_SMTP_DELIVERY}|%{POSTFIX_SMTP_CONNERR}|…}

Vordefinierte Grok-Patterns Beispiele:

- https://www.elastic.co/guide/en/logstash/master/plugins-filters-grok.html
- %{IP:client} %{WORD:method} %{URIPATHPARAM:request} %{NUMBER:bytes} %{NUMBER:duration}
- Verschachtelter Aufbau
- Test von Grok-Patterns möglich unter z.B.: grokconstructor.appspot.com

Logstash Patterns anwenden

- Prefilter: Wesentliche Struktur festlegen
- logstash/conf.d/50-filter-postfix.conf

```
... else if [program] =~ /^postfix.*\/smtpd$/ {
        grok {
            patterns_dir   => "/etc/logstash/patterns.d"
            match          => [ "message", "%{POSTFIX_SMTPD}" ]
            tag_on_failure => [ "_grok_postfix_smtpd_nomatch" ]
            add_tag        => [ "_grok_postfix_success" ]
        }
```

Aggregierung

Mappen der Felder aus: postfix/qmgr

```
else if [program] == "postfix/qmgr" and [postfix_from]{
   aggregate {
     task_id => "%{postfix_queueid}"
     code => "
       map['postfix_from'] = event.get('postfix_from')
       map['postfix_size'] = event.get('postfix_size')
       map['postfix_nrcpt'] = event.get('postfix_nrcpt')
     "
   }
```

Alle gemappten Felder nach postfix/smtp schreiben

```
else if [program] == "postfix/smtp" {
   aggregate {
            task_id => "%{postfix_queueid}"
            code => "map.each do |key, value| event.set(key, value) end"
            }
   }
```

Ausgabe Logstash → ELK

```
output {
        elasticsearch {
                hosts => ["https://192.168.10.172:9200"]
                index => "%{[type]}-%{+YYYY.MM.dd}"
                ssl => true
                cacert => '/etc/ssl/certs/x_intermediate_ca.pem'
                user => "logstash_internal"
                password => "xxxxxx"
                }
}
```

Anbindung REST-API Beispiel

- Python-Beispiel für die Suche und Anbindung an REST-API
- Pythonmodul Elasticsearch
- Anbindung Kibana

Beispiel Python

```
from elasticsearch import Elasticsearch
es = Elasticsearch(
    '192.168.64.2',
    http_auth=('zsolt', 'xxxxx'),
    scheme="http",
    port=9200,)


res = es.search(index="mail-*", body={"query": {"bool": {"must": [{"match_phrase": {"postfix_to": {"query": args.toaddr}}}, {"range" : {"@timestamp" : {"gte" : time_str, "lt" : "now"}}}]}}}, size=50)
```

Verbindung mit TLS

```
sslSettings = ssl.SSLContext()
sslSettings.verify_mode = ssl.CERT_REQUIRED
sslSettings.load_verify_locations("/usr/local/share/ca-certificates/ca.crt")

es = Elasticsearch(
    '192.168.10.174',
    http_auth=('zsolt', xxxx),
    scheme="https",
    port=9200,
    ssl_context=sslSettings,)
```

Jetzt tracen!

- Beispielsuche mit dem Script
- Verwenden der Ergebnisse (QueueIDs)
- Suche mit den QueueIDs in Kibana
- Suche nach Absender/Empfänger in Kibana

Sizing

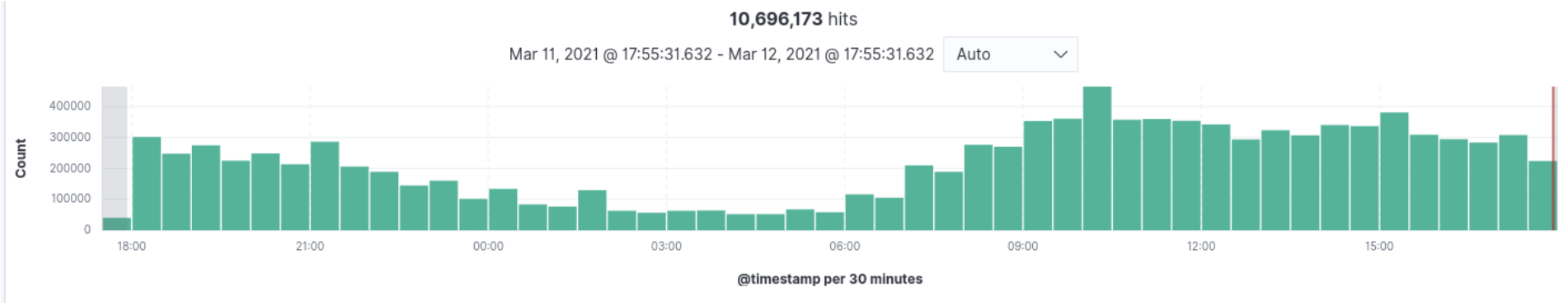
- Sizing: Nicht einfach zu beantworten: abhängig von Größe der Indexe, Anzahl der Nodes, verfügbarer JVM-Heap
- Einstellung über Index-Templates
- Default **number_of_shards = 1** ist für kleine Indexe gut (Hängt auch von der Gesamtanzahl der Shards ab)
- Problem des „Oversharding“:

  „Most searches hit multiple shards. Each shard runs the search on a single CPU thread. While a shard can run multiple concurrent searches, searches across a large number of shards can deplete a node’s search thread pool. This can result in low throughput and slow search speeds.“
- 20 Shards or fewer per GB of heap memory
- Replica für Such-Performance: max(max_failures, ceil(num_nodes / num_primaries) – 1)
- Setting Replica dynamisch, number_of_shards per Index
- Grösse eines Index 10GB - 50GB

Typischer Index

Name ↓	Health	Status	Primaries	Replicas	Docs count	Storage size
mbox-2021.03.12	green	open	1	2	7809684	11.2gb
mbox-2021.03.11	green	open	1	2	10888548	13.2gb
mbox-2021.03.10	green	open	1	2	10663461	13.1gb
mbox-2021.03.09	green	open	1	2	11760243	14.2gb
mbox-2021.03.08	green	open	1	2	11577725	13.9gb

Index-Größe


10,696,173 hits
Mar 11, 2021 @ 17:55:31.632 - Mar 12, 2021 @ 17:55:31.632
Auto
Count
400000
300000
200000
100000
0
18:00
21:00
00:00
03:00
06:00
09:00
12:00
15:00
@timestamp per 30 minutes

Lifecycle Policy

Index-Template:

```
{
  "index": {
    "number_of_shards": "1",
    "refresh_interval": "5s"
  }
}
```