Graylog

- Graylog-Index in Elasticsearch anzeigen und löschen
- Graylog-Services per Supervisord starten und überwachen
- ISPConfig nginx Logfiles an Graylog-Server schicken
- ISPConfig Apache2 Logfiles an Graylog-Server schicken
- rsyslog-Messages an Graylog-Server schicken

Graylog-Index in Elasticsearch anzeigen und löschen

Für Testzwecke läßt sich der Index von Graylog in Elasticsearch löschen (und damit alle gespeicherten Nachrichten entfernen). Das harte Löschen kann durchaus zu Problemen führen und sollte auf produktiven Servern vermieden werden!

Elasticsearch Indizes anzeigen:

```
# curl http://localhost:9200/_aliases
{"graylog2_0":{"aliases":{"graylog2_deflector":{}}}}
```

Graylog Index löschen (VORSICHT):

```
# curl -XDELETE localhost:9200/graylog2_0
{"ok":true,"acknowledged":true}root@loki:~#
```

Danach den Graylogserver neu starten.

Graylog-Services per Supervisord starten und überwachen

Um Graylog ohne eigenes Initscript starten und überwachen zu können setze ich das in Python geschriebene Tool Supervisor ein. Ähnlich wie monit kann es Dienste überwachen und bei Problemen neu starten.

Die Konfiguration für Graylog sieht so aus:

/etc/supervisor/conf.d/graylog2-server.conf

```
[program:graylog2server]
command=/usr/bin/java -jar graylog2-server.jar -f /etc/graylog2.conf -p /tmp/graylog2.pid
directory=/opt/graylog2-server
startsecs=60
user=graylog2
```

/etc/supervisor/conf.d/graylog2-web-interface.conf

```
[program:graylog2webinterface]
command=/opt/graylog2-web-interface/bin/graylog2-web-interface
directory=/opt/graylog2-web-interface
startsecs=60
user=graylog2
```

In der supervisord.conf waren keine weiteren Anpassungen notwendig.

Supervisor bietet eine Shell, über die man konfigurierte Dienste starten, stoppen, neustarten, usw. kann.

```
# supervisorctl
graylog2server                   RUNNING    pid 3264, uptime 23 days, 0:31:19
graylog2webinterface             RUNNING    pid 2238, uptime 23 days, 0:32:21

supervisor> stop graylog2webinterface
graylog2webinterface: stopped

supervisor> stop graylog2server
graylog2server: stopped

supervisor> status
graylog2server                   STOPPED    Aug 27 10:22 AM
graylog2webinterface             STOPPED    Aug 27 10:22 AM
```

ISPConfig nginx Logfiles an Graylog-Server schicken

Diese Konfiguration schickt nginx Access- und Errorlogs über GELF an einen Graylog-Server. Der GELF-Input im Graylog sollte natürlich aktiviert sein. Das Pattern-Matching funktioniert leider noch nicht exakt.

/etc/logstash/patterns.d/nginx-access.conf

```
NGINX_WEBSITE /[^/]+/[^/]+/[^/]+/[^/]+/(?<website>[^/]+)/
```

/etc/logstash/patterns.d/nginx-error.conf

```
HTTPERRORDATE %{DAY} %{MONTH} %{MONTHDAY} %{TIME} %{YEAR}
NGINXERRORLOG \[%{HTTPERRORDATE:timestamp}\] \[%{WORD:severity}\] \[client %{IPORHOST:clie
```

/etc/logstash/conf.d/nginx.conf

```
# nginx log input
input {
  file {
    type => "nginx-access"
    path => ["/var/log/ispconfig/httpd/*/access.log"]
  }
  file {
    type => "nginx-error"
    path => ["/var/log/ispconfig/httpd/*/error.log"]
  }
}

# filters
filter {
  if [type] == "nginx-access" {
      grok {
        match => { "message" => "%{COMBINEDAPACHELOG}" }
      }
      grok {
        patterns_dir => [ "/etc/logstash/patterns.d" ]
        match => [ "path", "%{NGINX_WEBSITE}" ]
      }
  }

  if [type] == "nginx-error" {
      grok {
        match => { "message" => "%{NGINXERRORLOG}" }
        patterns_dir => ["/etc/logstash/patterns.d"]
      }

    if !("_grokparsefailure" in [tags]) {

      mutate {
        remove_field => [ "message" ]
        add_field =>  ["timestamp_submitted", "%{@timestamp}"]
      }

      date {
        match => [ "timestamp", "EEE MMM dd HH:mm:ss yyyy" ]
        remove_field => [ "timestamp" ]
      }

      geoip {
        source => "clientip"
      }
    }
  }
}
```

```
# output
output {
  #stdout {
  #  #codec => "plain"
  #  codec => "rubydebug"
  #}
  gelf {
    host => "log.myserver.de"
    port => 12201
  }
}
```

ISPConfig Apache2 Logfiles an Graylog-Server schicken

Diese Konfiguration schickt nginx Access- und Errorlogs über GELF an einen Graylog-Server. Der GELF-Input im Graylog sollte natürlich aktiviert sein. Das Pattern-Matching funktioniert leider noch nicht exakt.

/etc/logstash/patterns.d/apache.conf

```
# get hostname from access.log path
APACHE_WEBSITE /[^/]+/[^/]+/[^/]+/[^/]+/(?<website>[^/]+)/

# error
APACHE_ERROR_TIME %{DAY} %{MONTH} %{MONTHDAY} %{TIME} %{YEAR}
APACHE_ERROR_LOG \[%{APACHE_ERROR_TIME:timestamp}\] \[%{LOGLEVEL:loglevel}\] (?:\[client %
```

/etc/logstash/conf.d/apache.conf

```
# apache log input
input {
  file {
    type => "apache-access"
    path => ["/var/log/ispconfig/httpd/*/access.log"]
  }
  file {
    type => "apache-error"
    path => ["/var/log/ispconfig/httpd/*/error.log"]
  }
}

# filters
filter {
  if [type] == "apache-access" {
      grok {
        match => { "message" => "%{COMBINEDAPACHELOG}" }
      }
      grok {
        patterns_dir => [ "/etc/logstash/patterns.d" ]
        match => [ "path", "%{APACHE_WEBSITE}" ]
      }
  }

  if [type] == "apache-error" {
    grok {
      patterns_dir => [ "/etc/logstash/patterns.d" ]
      match => [ "message", "%{APACHE_ERROR_LOG}" ]
    }

    if !("_grokparsefailure" in [tags]) {

      mutate {
        remove_field => [ "message" ]
        add_field =>  ["timestamp_submitted", "%{@timestamp}"]
      }

      date {
        match => [ "timestamp", "EEE MMM dd HH:mm:ss yyyy" ]
        remove_field => [ "timestamp" ]
      }

      geoip {
        source => "clientip"
      }
    }
  }
}

# output
```

```
output {
  #stdout {
  #  #codec => "plain"
  #  codec => "rubydebug"
  #}
  gelf {
    host => "log.myserver.de"
    port => 12201
  }
}
```

rsyslog-Messages an Graylog-Server schicken

Diese Konfiguration ermöglicht es rsyslog-Nachrichten an einen externen Graylog-Server weiterzuleiten:

/etc/rsyslog.d/graylog.conf

```
# keep full qualified domain names
$PreserveFQDN on

# graylog-server on log.myserver.de UDP Port 5140
$template GRAYLOGRFC5424,"<%pri%>%protocol-version% %timestamp:::date-rfc3339% %HOSTNAME% %
*.* @log.myserver.de:5140;GRAYLOGRFC5424

# graylog-server on log.myserver.de TCP Port 5140
#$template GRAYLOGRFC5424,"<%pri%>%protocol-version% %timestamp:::date-rfc3339% %HOSTNAME%
#*.* @@log.myserver.de:5140;GRAYLOGRFC5424
```