Logging

- syslog-ng
- rsyslog
  - Meldungen per Regex in eine andere Datei umleiten
  - Remote-Logging mit rsyslog
- 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

syslog-ng

Dieses Dokument beschreibt die Konfiguration und den Betrieb von Syslog-NG unter Fedora Core (5).

syslog-ng ist ein mächtiger syslogd-Ersatz.

Meine syslog-ng Konfiguration:

```
options {
    stats(3600);
    dir_perm(0755);
    perm(0644);
    chain_hostnames(no);
    keep_hostname(yes);
    time_reopen (10);
    log_fifo_size (1000);
    long_hostnames (off);
    use_dns (no);
    use_fqdn (no);
    create_dirs (no);
    keep_hostname (yes);
};

source s_sys {
    file ("/proc/kmsg" log_prefix("kernel: "));
    unix-stream("/dev/log");
    udp(ip(0.0.0.0) port(514));
    internal();
};

# /var/log/messages
filter f_messages { not facility(cron, mail, authpriv); };
filter f_nofirewall { not (facility(kern) and (match("RULE") or match("BLOCKLIST") or match("Act
destination d_messages { file("/var/log/messages"); };
log { source(s_sys); filter(f_messages); filter(f_nofirewall); destination(d_messages); };

# /var/log/firewall
filter f_firewall { match("RULE") or match("BLOCKLIST") or match("Activating firewall script");
destination d_firewall { file("/var/log/firewall"); };
log { source(s_sys); filter(f_firewall); destination(d_firewall); };

# /var/log/secure
filter f_authpriv { facility(authpriv); };
destination d_secure { file("/var/log/secure"); };
log { source(s_sys); filter(f_authpriv); destination(d_secure); };

# /var/log/maillog
filter f_maillog { facility(mail); };
destination d_maillog { file("/var/log/maillog" sync(10)); };
log { source(s_sys); filter(f_maillog); destination(d_maillog); };

# /var/log/cron
filter f_cron { facility(cron); };
destination d_cron { file("/var/log/cron"); };
log { source(s_sys); filter(f_cron); destination(d_cron); };

# consolenmeldung
filter f_emerg { level(emerg); };
destination d_console { usertty("*"); };
log { source(s_sys); filter(f_emerg); destination(d_console); };

# /var/log/spooler
filter f_spooler { facility(uucp,news); };
destination d_spooler { file("/var/log/spooler"); };
log { source(s_sys); filter(f_spooler); destination(d_spooler); };

# /var/log/boot.log
filter f_boot { facility(local7); };
destination d_boot { file("/var/log/boot.log"); };
log { source(s_sys); filter(f_boot); destination(d_boot); };

# alles
destination d_all { file("/var/log/all.log"); };
```

```
log { source(s_sys); destination(d_all); };
```

rsyslog

rsyslog

Meldungen per Regex in eine andere Datei umleiten

Mit diesem Beispiel lassen sich Meldungen, die von „apache2“ gesendet werden und nicht den String „PHP Notice“ enthalten in die Datei /var/log/apache2-global-error.log schreiben.

```
if $programname == 'apache2' and not ($msg contains 'PHP Notice') then /var/log/apache2-global-e
```

Das ist z.B. sehr nützlich, wenn man einen Cluster loadbalancter Server hat und diese zentral loggen läßt. Es gibt dann nur ein Error-Log, das man bequem durchsuchen kann und einen Überblick über die Fehler im gesamten Netz zeigt.

Remote-Logging mit rsyslog

Um den Empfang von Remote-Messages im rsyslog zu ermöglichen sind in /etc/rsyslog.conf zwei, bzw. 4 Zeilen einzukommentieren:

```
# provides UDP syslog reception
$ModLoad imudp
$UDPServerRun 514
```

Üblicherweise werden Syslog-Messages per UDP übertragen. Will man aber auch auf einem TCP-Port empfangen muss das entsprechende Modul geladen und konfiguriert werden:

```
# provides TCP syslog reception
$ModLoad imtcp
$InputTCPServerRun 514
```

Damit der entfernte Server nun auch Meldungen sendet ist folgendes zu tun:

```
*.* @syslog.meinserver.de
```

Damit werden sämtliche Syslog-Einträge per UDP an den entfernten Server geschickt. Sollen die Pakete über TCP laufen wird ein zweites @ vor den Server gepackt. Die Portangabe ist optional, default-Port ist 514. Sollen die Meldungen an einen anderen Port geschickt werden, ist das hierüber konfigurierbar.

```
*.* @@syslog.meinserver.de:1234
```

Die Meldungen lassen sich auch auf wichtige Sachen einschränken. Beispielsweise nur Emergencies und Alerts, etc.

```
*.emerg,*.alert     @syslog.meinserver.de
*.emerg,*.alert,*.crit,*.err,*.warning  @syslog.meinserver.de
```

Bei instabilen Verbindungen kann es helfen die Meldungen in einer Sendequeue vorzuhalten:

```
$ActionQueueType LinkedList
$ActionQueueFileName remote_queue
$ActionQueueMaxDiskSpace 1g
$ActionResumeRetryCount -1
$ActionQueueSaveOnShutdown on
```

Graylog

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

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
```

Graylog

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
  }
}
```

Graylog

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
```