

# Application Note

*Analyze ALOHA's HAProxy logs with halog*

**Document version:** v1.1

**Last update:** 3rd September 2013



## Purpose

Being able to analyze logs generated by the ALOHA Load-Balancer stored in a third party syslog server.

## Complexity



## Versions concerned

- Aloha 4.2 and above


## Synopsis

The ALOHA Load-Balancer, when used in layer 7 mode, generates very verbose log lines. It can store them in a circular buffer in memory. Unfortunately, the memory allocated to store the logs is limited and can only be used for instant troubleshooting.

To analyze an issue which occurred a few minutes ago, or to get tendencies on server or application response time, it is not enough.

To analyze logs, HAProxy comes with a small tool called **halog** that we can install and use on the server which stores the logs generated by the ALOHA.

## ALOHA remote syslog configuration

In the WUI do the following, go in the **Services** tab, then click on the **Edit** icon  from the **syslog traffic** line

Add a line **server** in the configuration, like in the example below:

```
server 192.168.10.26:514
```

Then click on the **restart** icon .

The ALOHA Load-Balancer emits two types of logs:

- traffic log: connection log, very verbose.  
They are sent with **local0** facility and **info** severity.

- event log: events occurring on frontends, backends, servers, etc...  
They are sent with **local0** facility and **notice** severity.

We may want to log them differently in the file system because each type of log can be used for different purpose.

## Linux syslog server configuration

### syslog-ng

**Syslog-ng** is one of the most powerful syslog server.

For **syslog-ng**, you have to define a source, a filter and a destination, like in the example below:

```
# tells syslog-ng to listen on its external IP
source s_net { udp(ip("192.168.10.26") port(514)); };

# where to write the logs
# traffic logs
destination d_aloha_traffic { file("/var/log/aloha/traffic.log"
    create_dirs(yes)); };
# event logs
destination d_aloha_events { file("/var/log/aloha/events.log"
    create_dirs(yes)); };

# ALOHA traffic logs are emitted with facility local0 and level info
filter f_aloha_traffic { facility(local0) and level(info); };
# ALOHA event logs are emitted with facility local0 and level notice
filter f_aloha_events { facility(local0) and level(notice); };

# traffic logging
log { source ( s_net ); filter(f_aloha_traffic);
    destination ( d_aloha_traffic ); };
# events logging
log { source ( s_net ); filter(f_aloha_events);
    destination ( d_aloha_events ); };
```

### rsyslog

**rsyslog** is one of the most used syslog server, since it's installed by default on the main Linux distribution.

For **rsyslog**, we have to enable the network socket and route ALOHA log lines too.

In order to make **rsyslog** listen on the network, uncomment the two line below in the file `/etc/rsyslog.conf`:

```
$ModLoad imudp
$UDPServerRun 514
```

In order to route syslog messages to different files, add the two line below to the end of the `/etc/rsyslog.conf` file:

```
# ALOHA logs traffic with facility local0 and severity info
local0.info                                -/var/log/aloha/traffic.log
# ALOHA logs events with facility local0 and severity notice
local0.notice                             -/var/log/aloha/events.log
```

## logrotate

When adding new log files, it's a good idea to rotate them as well as to delete oldest files. This is the role of **logrotate**.

Create a new file called **aloha** in logrotate's configuration directory `/etc/logrotate.d`:

```
/var/log/aloha/*.log
{
    rotate 31
    daily
    missingok
    notifempty
    delaycompress
    compress
    sharedscripts
    postrotate
        invoke-rc.d rsyslog reload > /dev/null
    endscript
}
```



In the example above, replace **rsyslog** by **syslog-ng**, depending which syslog server you're running

## HALog installation

**HALog** is a small and very powerful tool to analyze **ALOHA**'s log lines.

Installation is pretty simple, as described bellow:

```
cd /usr/src
wget http://haproxy.1wt.eu/download/1.5/src/devel/haproxy-1.5-dev11.tar.gz
tar xzf haproxy-1.5-dev11.tar.gz
cd haproxy-1.5-dev11/contrib/halog
make
cp halog /usr/bin/
```

## Analyzing ALOHA's logs

Now we have ALOHA's log and halog in the same server we can run some analyze on them.

## List servers by number of requests treated

The command below lists the servers by the number of requests they treated. The **server** name is prefixed by the **backend** name.

The eighth columns "**tot\_req**" gives the number of requests treated by the **server**.

```
cat traffic.log | halog -srv -H -q | awk 'NR==1; NR > 1 {print $0 | "sort -n -r -k 9"}' | column -t
#srv_name      1xx  2xx  3xx  4xx  5xx  other  tot_req req_ok pct_ok avg_ct avg_rt
dynamic/server1 0    3510 0    7    0    0      3517   3517  100.0 1495  1747
dynamic/server2 0    3516 0    0    0    0      3516   3516  100.0 1372  1776
```

## List servers by response time

The command below lists the servers by response time. The **server** name is prefixed by the **backend** name.

The response time is in milliseconds and the latest columns "**avg\_rt**" gives the average response time for all the URLs forwarded to this **server** in this **backend**.

```
cat traffic.log | halog -srv -H -q | awk 'NR==1; NR > 1 {print $0 | "sort -n -r -k 12"}' | column -t
#srv_name      1xx  2xx  3xx  4xx  5xx  other  tot_req req_ok pct_ok avg_ct avg_rt
dynamic/server2 0    3516 0    0    0    0      3516   3516  100.0 1372  1776
dynamic/server1 0    3510 0    7    0    0      3517   3517  100.0 1495  1747
```



It is a best practice to split dynamic and static traffic: you would see the server response time for each type of traffic

## List servers by application errors: HTTP status code 5xx

The command below lists the servers by number of application errors. The **server** name is prefixed by the **backend** name.

The sixth column "**5xx**" gives the number of application errors generated by the **server**.

```
cat traffic.log | halog -srv -H -q | awk 'NR==1; NR > 1 {print $0 | "sort -n -r -k 6"}' | column -t
#srv_name      1xx  2xx  3xx  4xx  5xx  other  tot_req req_ok pct_ok avg_ct avg_rt
dynamic/server2 0    3516 0    0    0    0      3516   3516  100.0 1372  1776
dynamic/server1 0    3510 0    7    0    0      3517   3517  100.0 1495  1747
```



It is a best practice to split your applications per backend, that way you will see whose application generates errors on which server

## List servers by errors

The command below lists the servers by number of errors not related to the application. The **server** name is prefixed by the **backend** name.

```
cat traffic.log | halog -srv -H -q | awk 'NR==1; NR > 1 {print $0 | "sort -n -r -k 5"}' | column -t
#srv_name      1xx  2xx  3xx  4xx  5xx  other  tot_req req_ok pct_ok avg_ct avg_rt
dynamic/server1 0    3510 0    7    0    0      3517   3517  100.0 1495  1747
dynamic/server2 0    3516 0    0    0    0      3516   3516  100.0 1372  1776
```

## List URLs by server computation time

The command below lists the URLs by the average computation time, whatever the server which treated it.

The sixth column "okavg" provides the URL average computation time in milliseconds.

```
cat traffic.log | halog -ut -H -q | column -t
#req  err  ttot   tavg  oktot  okavg  url
1004  0    6609819  6583  6609819  6583  /3s.php
2006  0    2771766  1381  2771766  1381  /health.php
2008  0    1601026  797   1601026  797   /fast.php
1004  0    1003335  999   1003335  999   /mega.php
1004  0    406830   405   406830   405   /health.html
7     0     19       2     19       2     /favicon.ico
```

## List URLs by errors

The command below lists the URLs by the number of errors they have generated, whatever the server which treated it or the type of error.

The second column "err" provides the number of errors generated by the given URL (latest column).

```
cat traffic.log | halog -ue -H -q | column -t
#req  err  ttot   tavg  oktot  okavg  url
1004  0    1003335  999   1003335  999   /mega.php
2006  0    2771766  1381  2771766  1381  /health.php
1004  0    406830   405   406830   405   /health.html
7     0     19       2     19       2     /favicon.ico
2008  0    1601026  797   1601026  797   /fast.php
1004  0    6609819  6583  6609819  6583  /3s.php
```

## List URLs by missing files: HTTP status code 404

The command below lists the URLs by the number of missing files error they have generated, whatever the server which treated it.

The first column "req" provides the number of 404 returned for the given URL (latest column).

```
cat traffic.log | halog -u -H -q -hs 404 | column -t
#req  err  ttot   tavg  oktot  okavg  url
7     0     19       2     19       2     /favicon.ico
```

## List URLs by number of request

The command below lists the URLs by the number of time they have been requested on the platform. The first column "req" provides the number of time the URLs was called.

```
cat aloha.log | halog -u -H -q | awk 'NR==1; NR > 1 {print $0 | "sort -n -r -k 1"}' | column -t
#req  err  ttot   tavg  oktot  okavg  url
2008  0    1601026  797   1601026  797   /fast.php
2006  0    2771766 1381   2771766 1381   /health.php
1004  0    6609819 6583   6609819 6583   /3s.php
1004  0    406830  405   406830  405   /health.html
1004  0    1003335 999    1003335 999    /mega.php
7     0     19      2     19      2     /favicon.ico
```