Log management with Logstash and Elasticsearch

Matteo Dessalvi

GSI hpc department

HEPiX 2013

HEPiX

Outline

- Centralized logging.
- Logstash: what you can do with it.
- Logstash + Redis + Elasticsearch.
- Grok filtering.
- Elasticsearch for indexing/searching the logs.
- Elasticsearch plugins and web interface.
- Pros and cons of this solution.
- References.

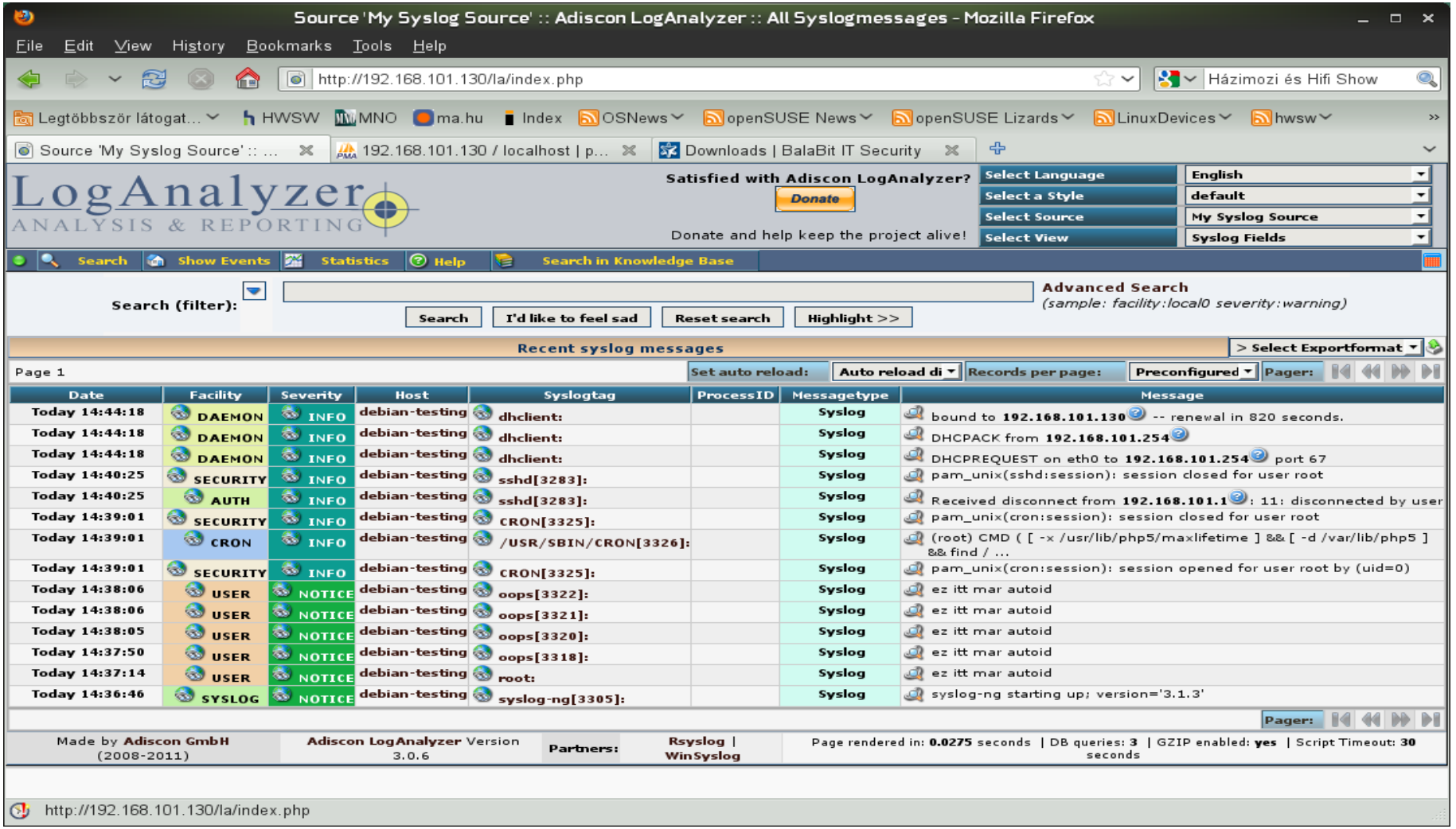
Centralized logging solutions

Logs are usually collected throughout software agents (rsyslog / syslog-ng) in one central location (usually on a relational DB).

But:

- The only thing available is the history of the logs. It's difficult to extract other statistics.
- This configuration is not flexible: [r]syslog clients to [r] syslog server only.

Loganalyzer web interface



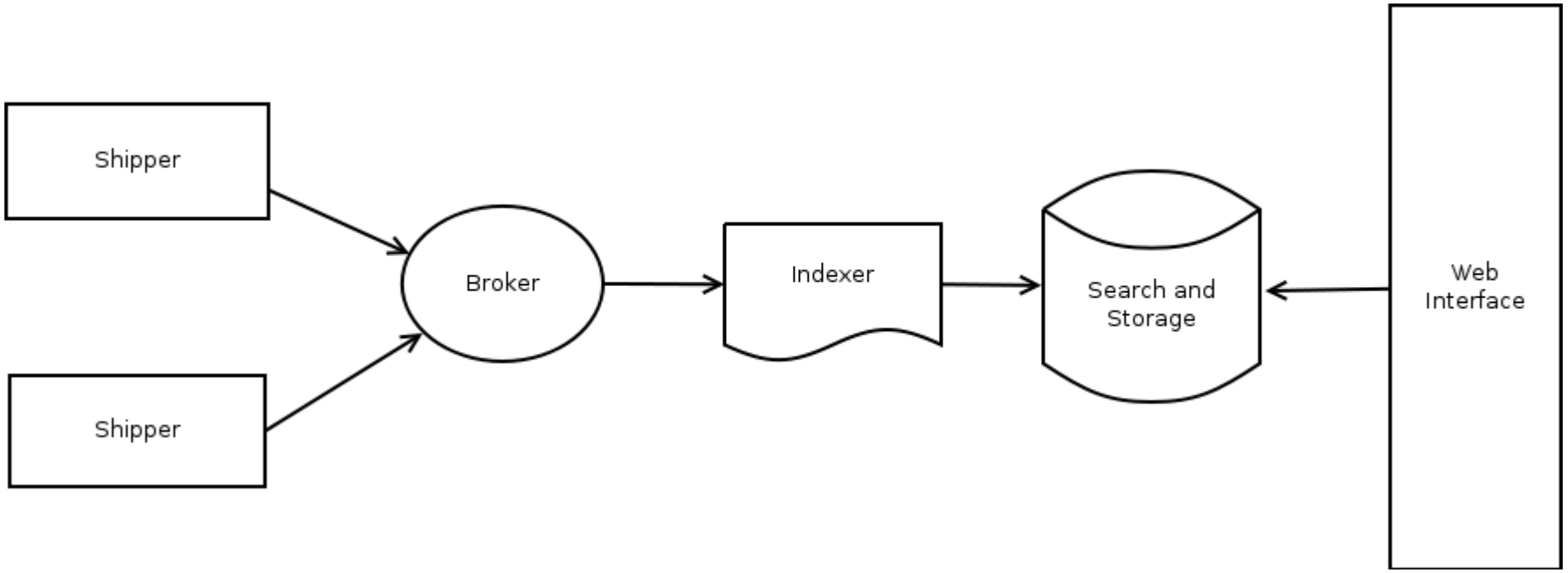
What is Logstash?

- Logstash is a tool for managing events and logs. You can use it to collect logs, parse them, and store them for later use.
- You can think of it as an event pipeline, divided in three parts: *inputs*, *outputs* and *filters*.
- It is written in *JRuby*, a Java implementation of Ruby.
- Easy to deploy: a single JAR file,it can be started directly from the cmd line (no Tomcat is needed).
- Depending on the configuration file a Logstash agent can act with different roles: *Shipper, Indexer, Broker, Searching/Storage, Web interface.*

Logstash functions

- *Shipper*: send the collected events to another Logstash instance or another software.
- *Broker and Indexer*: receives and indexes the events.
- *Search and Storage*: searching and storing the events.
- *Web Interface*: different options available: native one, based on Elasticsearch.

Logstash architecture



The system we are testing now is using Logstash only to ship and indexing the events. The *broker*, *search/storage* and *web interface* parts are replaced with other open source software.

Logstash configuration

The configuration file is mainly composed of two blocks, one called *input* and the other one called *output*. A third block, which is optional, is called *filter*.

- *Inputs*: how events gets into Logstash.
- *Filters*: how you can manipulate events in Logstash.
- *Outputs*: how you can output events from Logstash.

Logstash plugins

Input	*Filters*	*Output*
Amqp / Zeromq	CSV	Elasticsearch
Eventlog (Windows)	JSON	Ganglia
Ganglia	Grok	Graphite
Zenoss	XML	Nagios
log4j	Syslog_pri	OpenTSDB
Syslog	Multiline	MongoDB
TCP/UDP	Split	Zabbix
(...)	(...)	(...)

Complete list of plugins: http://logstash.net/docs/latest/

A small example

The most simple configuration file:

```
input { stdin
        { type => "stdin-type"}
      }
output {stdout
        { debug => true
          debug_format => "json"}
       }
```

Start a Logstash instance like this:

```
java -jar logstash-1.1.9-monolithic.jar agent -f config.
file
```

After that you can start to type something on the terminal.

Output in JSON format


```
{
  "@source":"stdin://localhost/",
  "@tags":[],
  "@fields":{},
  "@timestamp":"2013-04-08T08:07:08.282Z",
  "@source_host":"localhost",
  "@source_path":"/",
  "@message":"test",
  "@type":"stdin-type"
}
```

Fields description

@*source*: The source of the event which includes the plugin that generated it and the hostname that produced it.
@*tags*: An array of tags on the event.
@*fields*: A set of fields, for example "user": "james" for the event.
@*timestamp*: An ISO8601 timestamp.
@*source_host*: The source host of the event.
@*source_path*: The path, if any, of a source, for example /var/log/messages.
@*message*: The event's message. In our case it is what we put into STDIN.
@*type*: The value of the type configuration option we set.

Logstash syslog plugin

```
input {
        syslog {
                type => syslog
                port => 5000
              }}
output {
        elasticsearch {
                        host => "10.1.1.18"
                        }}
```

Note: the 'syslog' plugin status is experimental, which means it is essentially untested.

Chef client run event


```
{
"@source":"syslog",
"@tags":[],"@fields":{},
"@timestamp":"2013-03-26T06:40:36.692Z",
"@source_host":"lxb009",
"@source_path":"/var/log/rsyslog.d/lxb007/messages",
"@message":"Mar 26 07:40:35 lxb007 chef: [2013-03-26T07:
40:32+01:00] INFO: Starting Chef Run for
        lxb007.devops.test","@type":"linux-syslog"
}
```

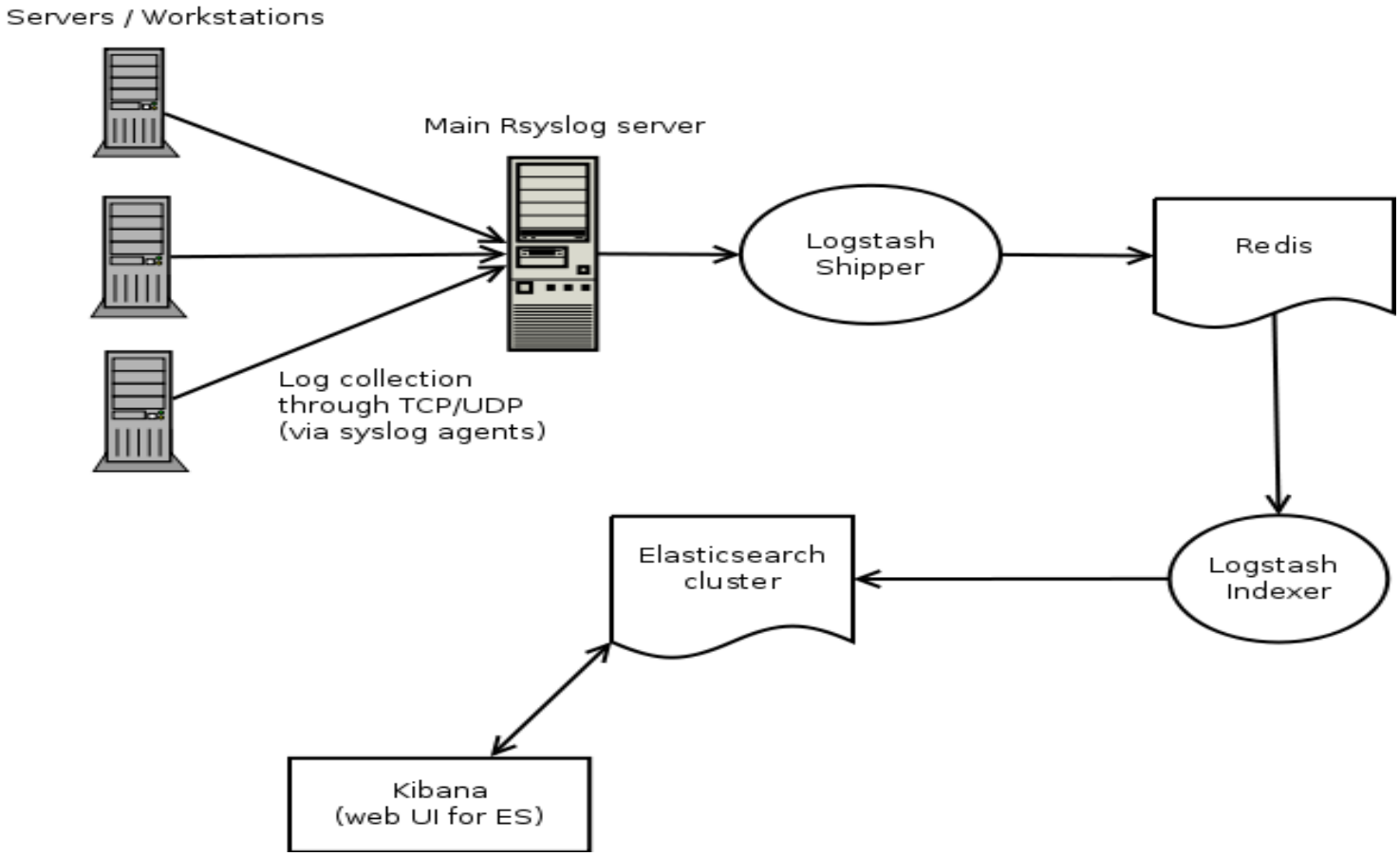
Some problems

The previous configuration may be good to test the basic capabilities of Logstash and Elasticsearch but it has some drawbacks:

- Logstash buffering capabilities are quite low: as the number of the events to be processed keep increasing the internal buffer may be filled up quite easily.
- Too tight interaction between Logstash and Elasticsearch: it makes not possible to update one of the software without breaking the flow of the logs.

Log farm diagram

To overcome the previous limitations we will split the roles of the various components using multiple instances of Logstash.



Components description

- *Logstash shipper*: this instance of Logstash will read the logs directly from the files saved on the central Rsyslog server.
- *Redis*: it act as a temporary broker.
- *Logstash indexer*: this instance will read the logs stored on Redis and it will redirect them directly to the Elasticsearch cluster.
- *Elasticsearch* will then index the logs and make it possible to run full text search on them.
- *Kibana*: it's a web interface for Logstash and Elasticsearch.

Why are we using Redis?

- Redis will give us an efficient system to buffering the logs collected through the shipper instance of Logstash.
- It will also makes easy to upgrade the Logstash indexing instances, without breaking the flow of the logs.

Redis is an in-memory persistent key-value store. Keys can contain *strings*, *hashes*, *lists*, *sets* and *sorted sets*.

Logstash shipping agent

```
input {
    file {
      type => "linux-syslog"
      path => [ "/var/log/*.log", "/var/log/messages",
                "/var/log/rsyslog.d/lxb*/*" ]
      exclude => [ "*.gz" ]
    }
}

output {
        stdout { debug => true debug_format => "json"}
        redis  {
                 host => "127.0.0.1"
                 data_type => "list"
                 key => "syslog"
                }
```

Config file directives

Input plugin:

@path: the path to the files that will be used use as an input.

@type: it populates the *type* of our event and it is used to help identify what events are .

@exclude: files excluded by Logstash.

Output plugin:

@host: where the Redis instance is located (default port is 6379).

@data_type: it specify the Redis data type to be used (*list*).

@key: the name of the list.

The *stdout* config is useful only for debugging or during the test phase.

Logstash indexer agent

```
input {
  redis {
          host => "127.0.0.1"
          type => "redis-input"
          data_type => "list"
          key => "syslog"
          format => "json_event"
        }
}

output {
  elasticsearch {
                  cluster => "lxb009"
                  node_name => "Logmaster"
                  host => "10.1.1.18"
                }
}
```

Config file directives

Input plugin:

@host: the IP address of the running Redis instance.

@type: it populates the *type* of our event and it is used to help identify what kind of events the filter is managing.

@data_type: depending on the value (list or channel) we will use different Redis operations on the data.

@key: the name of the Redis list or channel.

@format: the format of input data (plain, json, json_event).

Output plugin:

@cluster: the name of the ES cluster (useful for discovery).

@node_name: the node name ES will use when joining a cluster.

@host: the name/address of the host to use for ElasticSearch unicast discovery.

GROK: filtering the logs

Using the *grok* filter you'll be able to parse arbitrary text and structure it.

Grok works by using combining text patterns into something that matches your logs.

A grok pattern is: '%{`SYNTAX:SEMANTIC`}'

- '`SYNTAX`' is the name of the pattern that will match your text.
- '`SEMANTIC`' is the identifier you give to the piece of text being matched.

Logstash is shipped with about 120 patterns by default.

Postfix log filtering

```
filter  {
  grok {
        type => "postfix"
        pattern => [ "{%SYSLOGBASE}" ]
        add_tag => [ "postfix" ]
}
```

Based on the pattern already available in Logstash this filter will parse only logs from Postfix and it will just add the tag 'postfix' into it.

Grok filtering for bounced emails

```
grok {
    patterns_dir => "/etc/logstash/patterns"
    tags    => "postfix/bounce"
    pattern =>
     "%{TIMESTAMP_ISO8601} %{HOST}
      %{SYSLOGPROG}: %{QUEUEID}: to=<%{EMAILADDRESS:to}>,
      relay=%{RELAY},
      delay=%{POSREAL:delay}, delays=%{DELAYS},
      dsn=%{DSN}, status=%{STATUS} %{GREEDYDATA:reason}"
      add_tag => "BOUNCED"
      named_captures_only => true
}
```

Get metrics from the logs

Using the metrics plugin in Logstash you can extract useful numeric information. Using the generator plugin we will generate a stream a random event:

```
input {
  generator {
    type => "generated"
  }
}
```

The logs generated in this way will be catched by a proper grok filter and counted by the *metrics* plugin.

Grok and counting events

```
filter {
  metrics {
    type => "generated"
    meter => "events"
    add_tag => "metric"
  }
}

output {
  stdout {
    # only emit events with the 'metric' tag
    tags => "metric"
    message => "rate: %{events.rate_1m}"
  }}
```

A sample of the output

The stream of the events are counted every minute:

```
rate: %{events.rate_1m}
rate: 0.0
rate: 619.8
rate: 714.1315996203243
rate: 838.931746259273
rate: 1243.784314492951
```

Elasticsearch for indexing logs

- Elasticsearch is a REST based, distributed search engine built on top of the Apache Lucene library.
- JSON + HTTP API built in support.
- Elasticsearch can scale horizontally: you can add more ES instances and they will be automatically added to the cluster.
- ElasticSearch is able to achieve fast search responses because, instead of searching the text directly, it searches an index.

Elasticsearch terminology

Node: an Elasticsearch instance.

Cluster: a set of nodes (it's called cluster even if there's only one ES instance).

Document: a JSON document stored in ES. It's like a row in a table of a relational DB.

Index: it consists of one or more Documents.

Shard: they contain the data from the index.

Replica shard: used to increase search performance and for fail-over.

A terminology comparison

Relational database	*Elasticsearch*
Database	Index
Table	Type
Row	Document
Column	Field
Schema	Mapping
Index	Everything is indexed
SQL	Query DSL
SELECT * FROM table...	GET http://...
UPDATE table SET	PUT http://...

Search through ES with the cmd line

```
curl 'http://lxb009:9200/logstash-2013.03.26/
   linux-syslog/_search?q=lxb009&pretty=true'
```

Output:


```
{
 "_index" : "logstash-2013.03.26",
 "_type" : "linux-syslog",
 "_id" : "ySFmhBadQOaZME2A3VSZcw",
 "_score" : 0.34786326, "_source" :
{ "@source":"file://lxb009/var/log/rsyslog.d/lxb009/messages",
"@tags":[],"@fields":{},"@timestamp":"2013-03-26T02:12:48.559
Z",
"@source_host":"lxb009","@source_path":"/var/log/rsyslog.
d/lxb009/messages",
"@message":"Mar 26 03:12:47 lxb009 dhclient: bound to 10.1.1.18
          -- renewal in 1460 seconds.","@type":"linux-syslog"
}
```

Access ES through command line

Using the es2unix tool it is possible to obtain various informations from the Elasticsearch cluster.

es health

```
23:37:58 lxb009  yellow     1    1  30      30     0     0   30
```

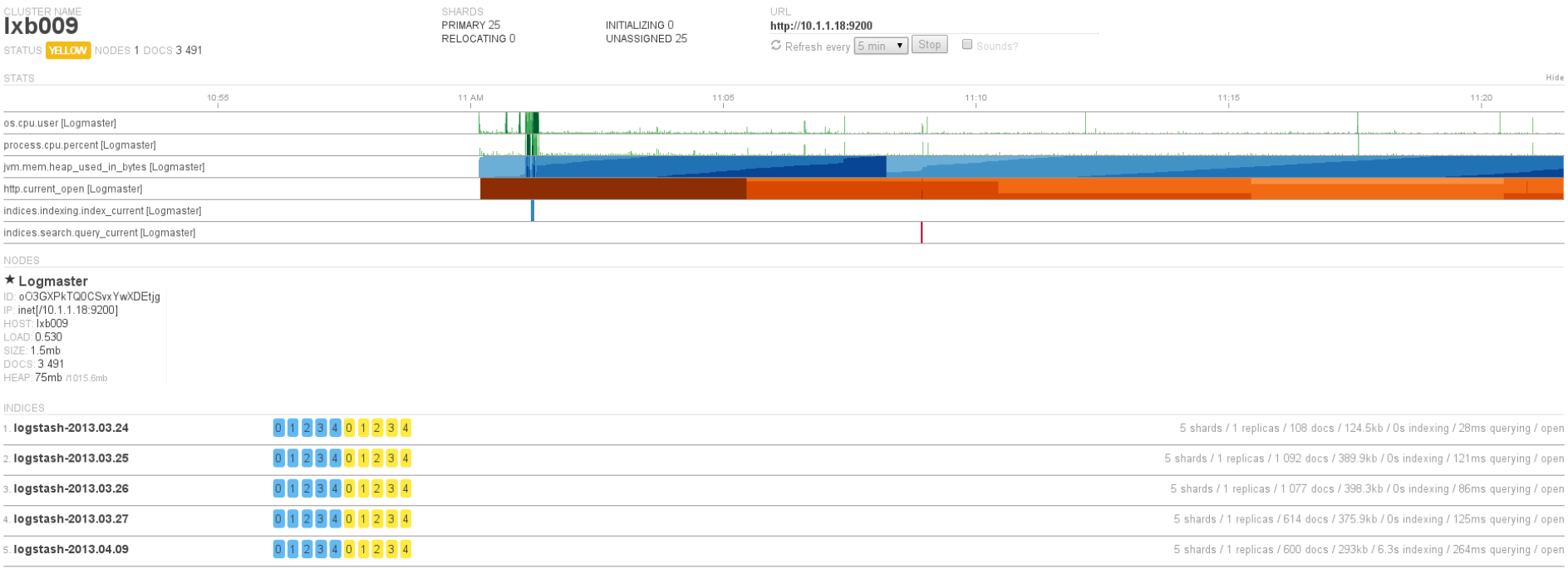
es search

```
1.0     logstash-2013.03.24 linux-syslog nDzGwOyMTSehKv-4lnuVcw
1.0     logstash-2013.03.24 linux-syslog OzUANuFpTaKKdzmHSMdGrw
1.0     logstash-2013.03.24 linux-syslog hifUg4O-THyo_97nqN9M_A
(...)
   Total: 3791
```

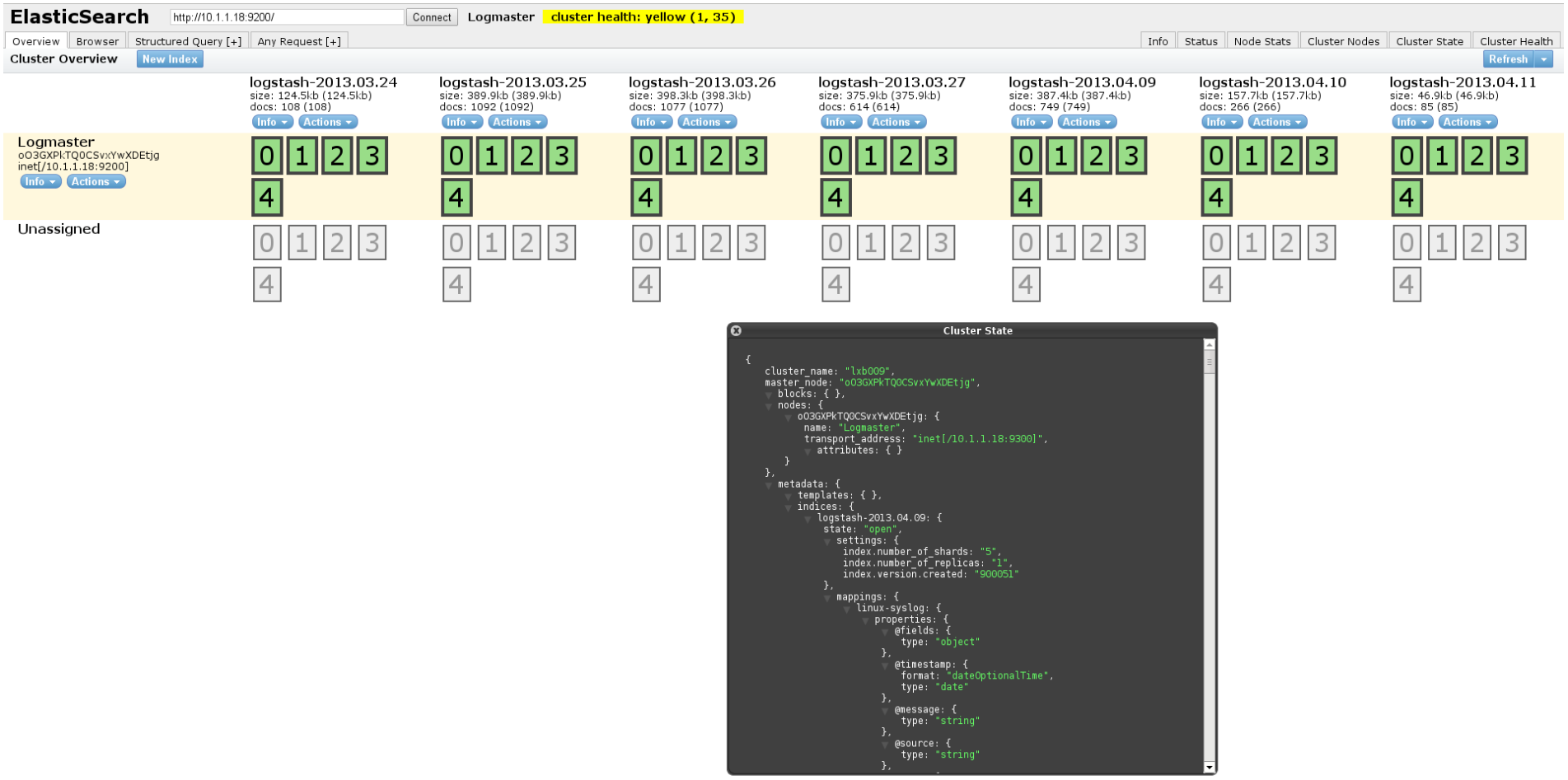
You can also use the '-v' parameter to print the column headings.

Monitoring the Elasticsearch cluster

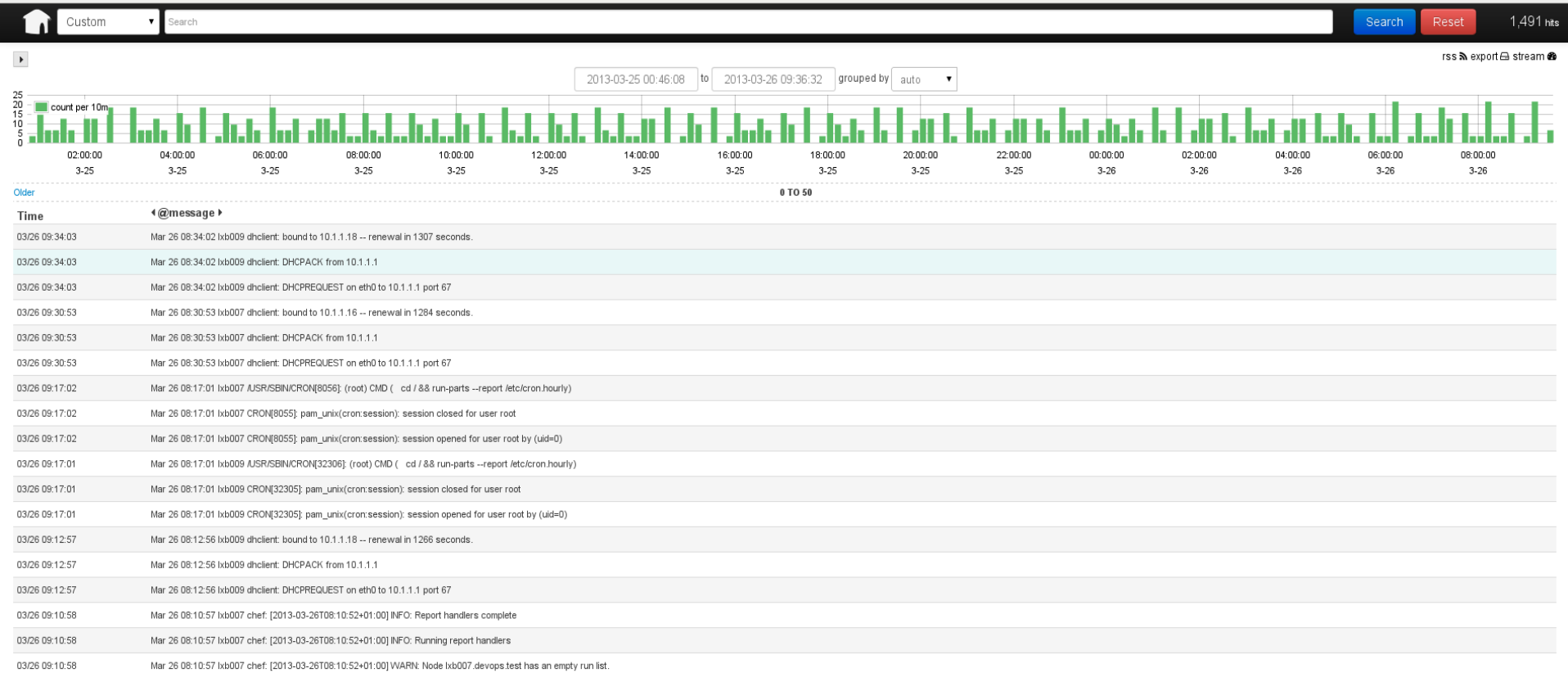
Web view of the ES status through the paramedic plugin.



ES: status of the index using the head plugin



Kibana dashboard



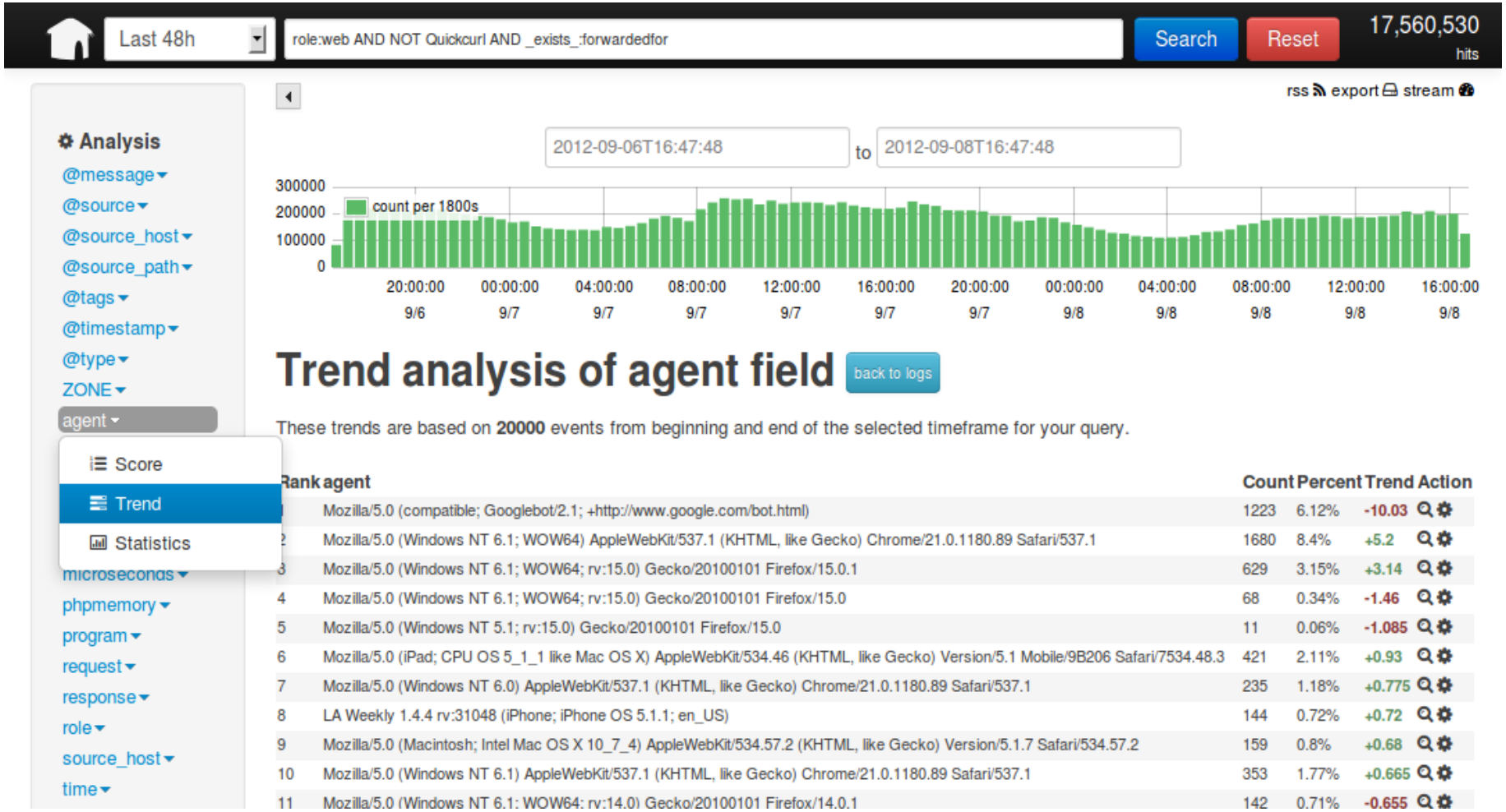
Time	@message
03/26 09:34:03	Mar 26 08:34:02 lxb009 dhclient: bound to 10.1.1.18 -- renewal in 1307 seconds.
03/26 09:34:03	Mar 26 08:34:02 lxb009 dhclient: DHCPACK from 10.1.1.1
03/26 09:34:03	Mar 26 08:34:02 lxb009 dhclient: DHCPREQUEST on eth0 to 10.1.1.1 port 67
03/26 09:30:53	Mar 26 08:30:53 lxb007 dhclient: bound to 10.1.1.16 -- renewal in 1284 seconds.
03/26 09:30:53	Mar 26 08:30:53 lxb007 dhclient: DHCPACK from 10.1.1.1
03/26 09:30:53	Mar 26 08:30:53 lxb007 dhclient: DHCPREQUEST on eth0 to 10.1.1.1 port 67
03/26 09:17:02	Mar 26 08:17:01 lxb007 /USR/SBIN/CRON[8056]: (root) CMD (cd / && run-parts --report /etc/cron.hourly)
03/26 09:17:02	Mar 26 08:17:01 lxb007 CRON[8055]: pam_unix(cron:session): session closed for user root
03/26 09:17:02	Mar 26 08:17:01 lxb007 CRON[8055]: pam_unix(cron:session): session opened for user root by (uid=0)
03/26 09:17:01	Mar 26 08:17:01 lxb009 /USR/SBIN/CRON[32306]: (root) CMD (cd / && run-parts --report /etc/cron.hourly)
03/26 09:17:01	Mar 26 08:17:01 lxb009 CRON[32305]: pam_unix(cron:session): session closed for user root
03/26 09:17:01	Mar 26 08:17:01 lxb009 CRON[32305]: pam_unix(cron:session): session opened for user root by (uid=0)
03/26 09:12:57	Mar 26 08:12:56 lxb009 dhclient: bound to 10.1.1.18 -- renewal in 1266 seconds.
03/26 09:12:57	Mar 26 08:12:56 lxb009 dhclient: DHCPACK from 10.1.1.1
03/26 09:12:57	Mar 26 08:12:56 lxb009 dhclient: DHCPREQUEST on eth0 to 10.1.1.1 port 67
03/26 09:10:58	Mar 26 08:10:57 lxb007 chef: [2013-03-26T08:10:52+01:00] INFO: Report handlers complete
03/26 09:10:58	Mar 26 08:10:57 lxb007 chef: [2013-03-26T08:10:52+01:00] INFO: Running report handlers
03/26 09:10:58	Mar 26 08:10:57 lxb007 chef: [2013-03-26T08:10:52+01:00] WARN: Node lxb007.devops.test has an empty run list.
03/26 09:10:58	Mar 26 08:10:57 lxb007 chef: [2013-03-26T08:10:52+01:00] INFO: Chef Run complete in 0.269177615 seconds
03/26 09:10:58	Mar 26 08:10:57 lxb007 chef: [2013-03-26T08:10:52+01:00] INFO: Loading cookbooks []
03/26 09:10:58	Mar 26 08:10:57 lxb007 chef: [2013-03-26T08:10:52+01:00] INFO: Start handlers complete.
03/26 09:10:58	Mar 26 08:10:57 lxb007 chef: [2013-03-26T08:10:52+01:00] INFO: Running start handlers
03/26 09:10:58	Mar 26 08:10:57 lxb007 chef: [2013-03-26T08:10:52+01:00] INFO: Starting Chef Run for lxb007.devops.test
03/26 09:10:58	Mar 26 08:10:57 lxb007 chef: [2013-03-26T08:10:52+01:00] INFO: HTTP Request Returned 404 Not Found: No routes match the request: /reports/nodes/lxb007.devops.test/runs
03/26 09:10:58	Mar 26 08:10:57 lxb007 chef: [2013-03-26T08:10:52+01:00] INFO: Run List expands to []

Detailed view of a log entry

Time	@message
04/10 15:54:58	Apr 10 15:54:57 lxb009 dhclient: bound to 10.1.1.18 -- renewal in 1454 seconds.

Field	Action	Value
@message		Apr 10 15:54:57 lxb009 dhclient: bound to 10.1.1.18 -- renewal in 1454 seconds.
@source		file://lxb009/var/log/rsyslog.d/lxb009/messages
@source_host		lxb009
@source_path		/var/log/rsyslog.d/lxb009/messages
@tags		
@timestamp		2013-04-10T13:54:58.711Z
@type		linux-syslog

Kibana trends analysis



Pros and Cons

- Extreme flexible: with Logstash and ES you can collect and index logs/events coming from barely any kind of input sources.
- Grok filtering capabilities help to properly manage different kind of logs and change their structure according to your needs.

- Hardware demands: Logstash and Elasticsearch can be quite hungry in terms of RAM usage. JVM options needs to be properly tuned.
- No user access control: once Kibana is up and running there's no mechanism to control who is accessing the service.

References

http://logstash.net
http://www.elasticsearch.org
http://redis.io/
A web interface for managing Redis instances:
https://github.com/steelThread/redmon

Elasticsearch API consumable by the command line:
https://github.com/elasticsearch/es2unix

http://www.elasticsearch.org/tutorials/using-elasticsearch-for-logs/
http://logstash.net/docs/1.1.9/tutorials/metrics-from-logs

Thank you!

Questions?