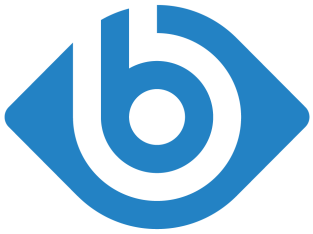
How to parse data with syslog-ng, store in Elasticsearch and analyze with Kibana

January 06, 2016



BALABIT

CONTEXTUAL SECURITY INTELLIGENCE

Table of Contents

1. Introduction

Anytime a new language binding is introduced to *syslog-ng*, somebody immediately implements an *Elasticsearch* destination. There is one in Lua, Perl and Python, meaning that there is a very strong interest in getting data from syslog-ng into Elasticsearch. Recently, an official Elasticsearch destination has been developed by the syslog-ng team in Java.

Why do so many people want to send their logs to Elasticsearch? There are many reasons:

- It is an easy-to-scale and easy-to-search data store
- It is NoSQL: any number of name-value pairs can be stored
- *KIBANA*: an easy-to-use data explorer and visualization solution for Elasticsearch

And why to use syslog-ng on the sending side? There are also very good reasons for that:

- A single, high-performance and reliable log collector for all of your logs, no matter if they are coming from network devices, local system or applications. Therefore, it can greatly simplify your logging architecture.
- High speed data processor, parsing both structured (*JSON, CSV/CLICK STREAM*) and unstructured log messages (*PATTERNDB*). It can also anonymize log messages if required by policies or regulations, and reformat them to be easily digested by analyzers.
- Complex *FILTERING*, to make sure that only important messages get through and they reach the right destination.

The next screenshot shows a Kibana dashboard, which displays logs collected by syslog-ng, parsed by PatternDB and stored into Elasticsearch by our brand new Java-based driver:

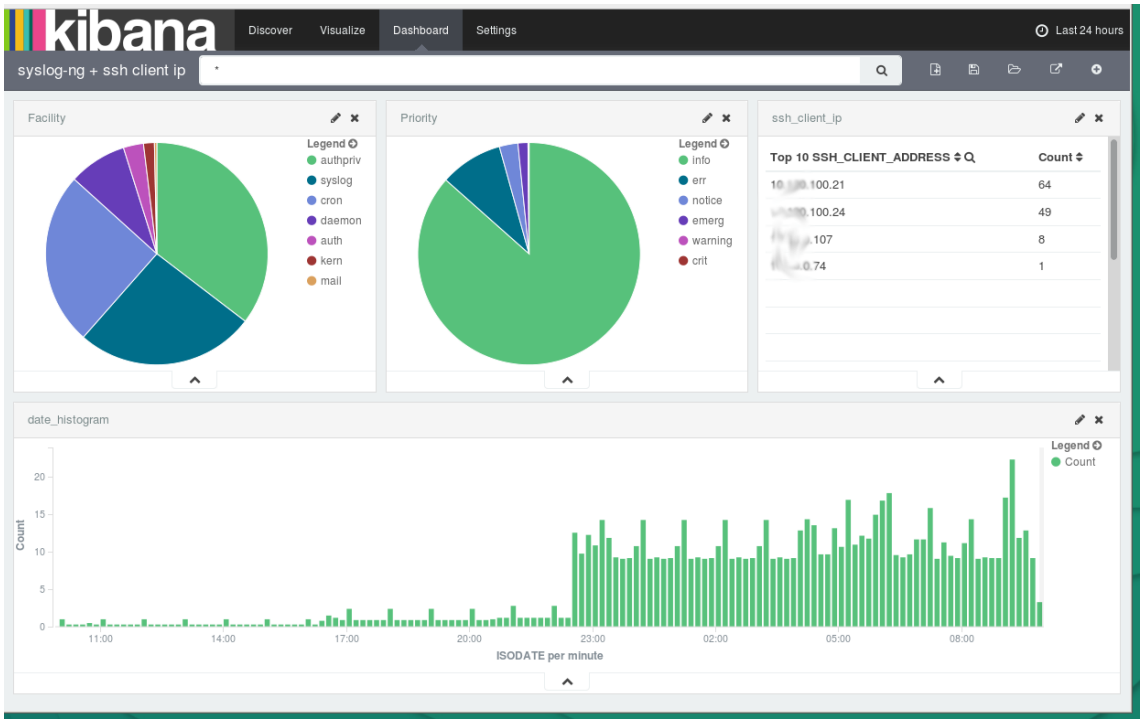


Figure 1. Logging directly to the central syslog-ng server

You might ask, why Java, when it is usually not as resource-efficient as something developed in C. The simple answer is that while there are client libraries implemented in dozens of languages, there is not one written in C. The official Elasticsearch client is implemented in Java and developed together with the server component. The core of syslog-ng remains in C, it can efficiently collect, process and filter logs just as until now. There is a small, syslog-ng-specific Java code that can utilize the official Elasticsearch client JAR files to connect to Elasticsearch clusters. This ensures full compatibility even with a quickly changing Elasticsearch now and also in the future.

The simplicity that syslog-ng brings to the logging architecture, canot be over-emphasized. Logstash, or its dedicated file reader component, Filebeat, requires syslog to be installed on machines to be able to collect system logs. In comparison, syslog-ng can be used to collect both system and application logs. There is no need for a separate file reader or a separate system log collector. The syslog-ng application can also perform all the log processing-related tasks, which means that a single application can be used instead of having to configure three different software (syslog + filebeat + Logstash) for the same tasks. Or rather four, as queuing is missing from Logstash, so it utilizes external queuing solutions to level out peaks or survive network outages. The syslog-ng application can use disk-based buffering for the same purpose, avoiding the need for an external software for it.

2. Getting started

If you would like to use the Elasticsearch destination, you will need to install the latest syslog-ng version. For syslog-ng Premium Edition, the latest version is 5 F5 and all the information to get started is included in the documentation at *Sending messages directly to Elasticsearch*. For the Open Source Editon, 3.7.2 is the current version. Java support is not yet included in the syslog-ng packages available as part of the different Linux distributions due to lack of required JAR files and a build tool (Gradle). Fortunately there are 3rd party repositories for several Linux distributions (Debian, Fedora, openSUSE and many of their derivatives), which have Java support enabled, together with instructions on how to get started: *Getting started with syslog-ng and Elasticsearch / Hadoop / Kafka* In either case you need to make sure that syslog-ng can load `libjvm.so` and that the Elasticsearch JAR files are available.

3. Configuring syslog-ng

Once all the prerequisites are met, you can continue with configuring syslog-ng. The next example is the relevant part of a syslog-ng.conf. You might need to change directory names and other parameters to better fit your system:

```
######
# network sources
source s_net {
  udp();
  tcp();
  syslog();
};

######
# patterndb parser
parser pattern_db {
  db-parser(
    file("/opt/syslog-ng/etc/patterndb.xml")
  );
};

######
# Elasticsearch destination
destination d_es {
  elasticsearch(
    index("syslog-ng_${YEAR}.${MONTH}.${DAY}")
    type("test")
    cluster("syslog-ng")
    template("$(format-json --scope rfc3164 --scope nv-pairs --exclude R_DATE --key
 ISODATE)\n");
  );
};

######
# sending logs to ES destination
log {
  source(s_net);
  source(s_local);
  parser(pattern_db);
  destination(d_es);
  flags(flow-control);
};
```

Unless you want to store logs from a single machine in Elasticsearch, you will need to declare a network source, where you can collect logs from remote machines. Source “s_net” collects legacy syslog messages on port 514 through both UDP and TCP, and RFC5424 syslog messages on port 601. If you want to use other ports, encryption or any other parameters, consult the “Collecting log messages” section of the syslog-ng admin guide.

I have already mentioned in the introduction, that Elasticsearch is a NoSQL database and can store any number of different name-value pairs. Basic syslog data can easily be stored in SQL databases, but syslog-ng can parse messages and create name-value pairs based on message content. There are parsers for JSON-formatted messages

and columnar data, like CSV files or Apache access logs, but the most interesting one is PatternDB, a radix tree-based parser in syslog-ng, which can parse unstructured logs at extreme speed, without the performance penalties of RegExp-based parsers. On the above screenshot you can see results from parsing SSH login messages: PatternDB-created name-value pairs, which contain the source IP address of the logged in user.

There are some sample *patterns available on GitHub* and you can read more about creating new patterns in the "Processing message content with a pattern database" chapter of the syslog-ng admin guide. For this example, I have used *sshd.pdb*.

The next section in the cited configuration is about configuring the Elasticsearch destination. The only required parameters are index and type, the rest have default value and syslog-ng connects to Elasticsearch in transport mode. Here we also modify the cluster name and the message template. You might also need to modify directory names for JAR files using the `client-lib-dir()` parameter.

The latest release also enabled the possibility to send concurrent requests to Elasticsearch servers with the `concurrent-requests()` option, which can increase performance considerably. Defaults were also changed to provide better out-of-the-box performance, by sending messages in batches instead of one by one. Note that there is trade-off between batch size, message throughput and risk of losing messages in case of failure. Read the documentation about the differences between connection modes and a detailed description of parameters.

Finally, all of the previously mentioned components are glued together by a log statement. There are two extra parameters: `source(s_local)` stands for the logs collected on the local machine and might be called something completely different in your base `syslog-ng.conf`. The other one is about flow control. This is an optional mechanism in syslog-ng, which slows down reading from sources (not possible for UDP) when the destination is not fast enough. It is described in the "Managing incoming and outgoing messages with flow-control" section of the documentation.

4. Configuring Kibana

Once you configured syslog-ng to store logs into Elasticsearch, it is time to configure Kibana. Starting with version 4.0 it is a standalone server application. You might have to configure the elasticsearch_url variable, if Kibana is installed to a separate machine (by default it searches for Elasticsearch on localhost). When you first access it, the web interface requires you to configure an index pattern. The index name should be “syslog-ng_*”, and the time field name “ISODATE”, if you use the above configuration.

The “Discover” is the first screen to appear after configuration. You should be able to see the messages sent by syslog-ng.

Even if it takes a few minutes and a few megabytes of extra storage space, I would recommend you to follow the “Getting started” tutorial from the Kibana documentation now. There is some sample data to download and load into your Elasticsearch cluster and a few exercises that demonstrate the basic functionality of Kibana in the *Kibana User Guide*.

5. Next steps

We hope you find this information useful to get started. We have already provided several pointers in the documentation, which should cover most aspects on the syslog-ng side. Getting started with Elasticsearch is as easy as installing a package on a single machine and starting it. On the other hand, if you want to use it in production, you should definitely read the Elasticsearch documentation as well about clustering, removing old logs, performance tuning and so on.

If you have any questions, or would like to report a problem, you can reach us through the following:

- *The syslog-ng mailing list*
- The *source code of syslog-ng is on GitHub* where you can also report issues
- On IRC the #syslog-ng channel on Freenode
- For commercial support check *the Premium Edition website*