

# Using NXLog with Elasticsearch and Kibana

## Contents

<b>1</b>	<b>Setting up Elasticsearch and Kibana</b>	<b>1</b>
1.1	Installing Elasticsearch . . . . .	1
1.2	Installing Kibana . . . . .	1
<b>2</b>	<b>Loading data into Elasticsearch with NXLog</b>	<b>2</b>
2.1	Loading data with om_elasticsearch . . . . .	2
2.2	Loading data with om_http . . . . .	4
2.3	Using Logstash . . . . .	5

---

Elasticsearch coupled with the Kibana frontend has become quite popular recently as a low-cost centralized log monitoring solution. This is commonly referred to as the ELK stack comprised of Elasticsearch, Logstash and Kibana.

While Logstash is a great piece of software it has some disadvantages compared to NXLog:

- Logstash is written in ruby and requires Java to run. Besides being a lot more hungry on system resources, many system administrators would rather not take the hassle of deploying the Java runtime onto their production servers and needing take care of the Java security updates.
- The Eventlog plugin in Logstash pulls the eventlog data through the Windows WMI interface which incurs a significant performance penalty. NXLog hooks directly into the Windows EventLog API natively and can collect logs from our highly loaded Domain Controllers also.
- It's just one more piece of software to take care about.

NXLog is a small and efficient log collector that can be set up to securely and reliably centralize event data from Windows and Unix platforms. As such, NXLog is recommended by many ELK users as the log collector of choice for Windows and Linux.

While most users run Logstash in front of Elasticsearch, contrary to popular belief Logstash is not necessarily needed to load data since NXLog can do this as well. This is the KEN stack: Kibana, Elasticsearch and NXLog.

In this this guide we will show you how to set up and configure NXLog, Elasticsearch and the Kibana web interface. We are going use Debian Wheezy as the base OS.

## 1 Setting up Elasticsearch and Kibana

This guide is not intended to provide all the details and it just shows the bare minimum required to get Elasticsearch and Kibana installed and running so that we can point NXLog at it.

There are many guides apart from the official installation instructions on the web on how to set up and configure Elasticsearch and Kibana, for example the following two tutorials at DigitalOcean can be quite useful:

- [How To Install Elasticsearch, Logstash, and Kibana 4 on Ubuntu 14.04](#)
- [How To Use Kibana Dashboards and Visualizations](#)

### 1.1 Installing Elasticsearch

You can download the latest version from <http://elastic.co>

```
wget https://download.elastic.co/elasticsearch/elasticsearch/elasticsearch-1.5.1.deb
dpkg -i elasticsearch-1.5.1.deb
```

Then start Elasticsearch with:

```
/etc/init.d/elasticsearch start
```

### 1.2 Installing Kibana

Kibana is a javascript-based web frontend for Elasticsearch offering an attractive user interface.

Download and extract the latest version of Kibana:

```
wget http://download.elastic.co/kibana/kibana/kibana-4.0.2-linux-x64.tar.gz
tar xvf kibana-*.tar.gz
mv kibana-4.0.2-linux-x64 /opt/kibana
```

Now start Kibana:

```
/opt/kibana/bin/kibana
```

With everything in place we should be able to access Kibana from the browser at <http://localhost:5601>.

## 2 Loading data into Elasticsearch with NXLog

### 2.1 Loading data with `om_elasticsearch`

The NXLog Enterprise Edition comes with a module that can load data natively into Elasticsearch. The advantage of this module over `om_http` is the support for bulk data operations and dynamic indexing. Event data is sent in batches, this greatly reduces the latency caused by the HTTP responses and the elasticsearch server can also process bulk data faster. The `om_elasticsearch` module can insert data at a rate of 10,000EPS or more on low end hardware.

For Kibana's time filters to work properly we will need to apply a template. This can be pushed to Elasticsearch with the following command:

```
curl -XPUT localhost:9200/_template/nxlog -d '{ "template" : "nxlog*", "mappings" : { " ←
  _default_ : { "properties": { "EventTime": { "type": "date", "format": "YYYY-MM-dd HH: ←
    mm:ss" } } } } }'
```

This will tell Elasticsearch to treat our `EventTime` field as date and parse it in the given format. This will be applied to all records in indexes beginning with `nxlog`. Further mappings can be added in the properties section.

The following configuration shows how to use the `om_elasticsearch` module to load the log data into Elasticsearch.

```
<Extension _json>
  Module xm_json
</Extension>

<Input in>
  Module im_tcp
  Host 0.0.0.0
  Port 1514
  InputType Binary
</Input>

<Output es>
  Module om_elasticsearch
  URL http://localhost:9200/_bulk
  FlushInterval 2
  FlushLimit 100
  # Create an index daily
  Index strftime($EventTime, "nxlog-%Y%m%d")
  IndexType "My logs"

  # Use the following if you don't have $EventTime set
  #Index strftime(now(), "nxlog-%Y%m%d")
</Output>

<Route r>
  Path in => es
</Route>
```

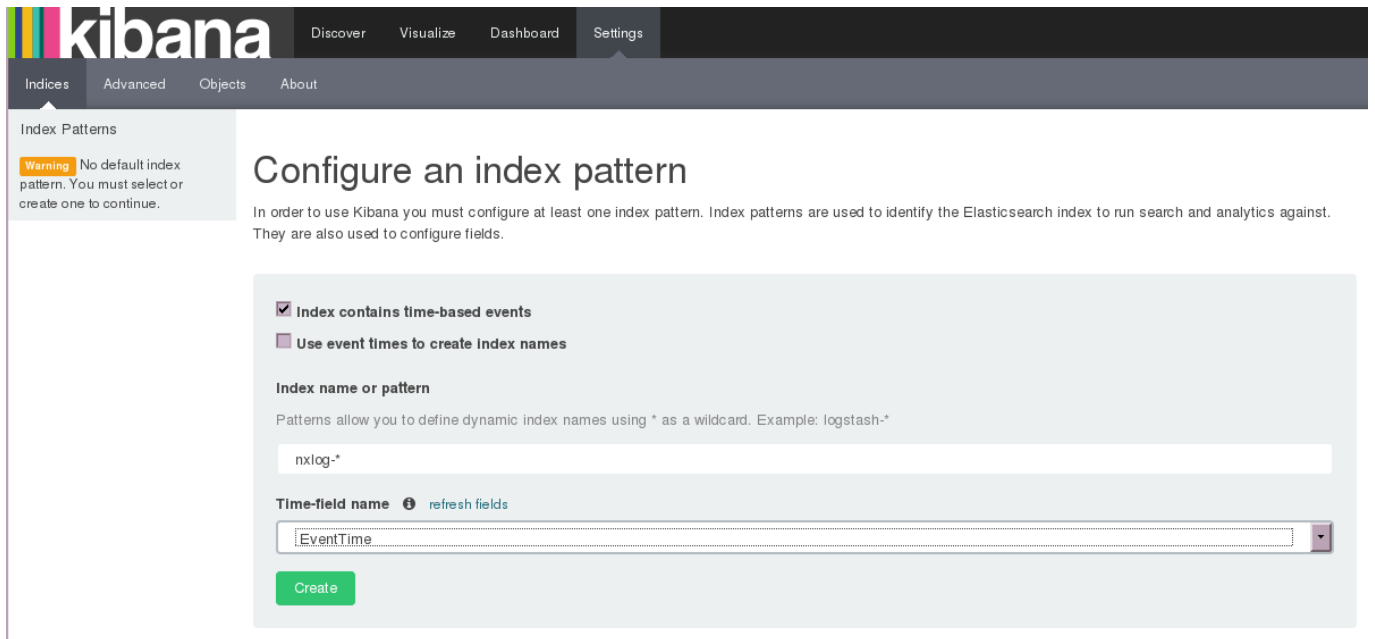
The `IndexType` parameter, although not mandatory, can help sorting our logs on the dashboard. It expects a string type expression and is reevaluated for each event record. Refer to the NXLog Enterprise Edition Reference Manual for further details.

The input section in the configuration above uses `im_tcp` with NXLog's Binary data format as this can preserve structured data across the network better than JSON. The client (agent) side configuration is not the scope of this document.

After saving the configuration we should restart NXLog to apply the changes:

```
/etc/init.d/nxlog restart
```

Logs should now be sent to and indexed by Elasticsearch. To verify that the data is getting ingested, first we will need to configure the Kibana indexing as shown in the screenshot below.



The screenshot shows the Kibana interface for configuring an index pattern. The top navigation bar includes 'Discover', 'Visualize', 'Dashboard', and 'Settings'. The left sidebar shows 'Indices', 'Advanced', 'Objects', and 'About'. The main content area is titled 'Configure an index pattern' and includes a warning: 'Warning No default index pattern. You must select or create one to continue.' Below this, there are two checkboxes: 'Index contains time-based events' (checked) and 'Use event times to create index names' (unchecked). The 'Index name or pattern' section has a text input field containing 'nxlog-\*'. The 'Time-field name' section has a dropdown menu with 'EventTime' selected and a 'refresh fields' link. A green 'Create' button is at the bottom.

Figure 1: Kibana index configuration

Now we should be able to search and analyze event data on the Kibana interface.

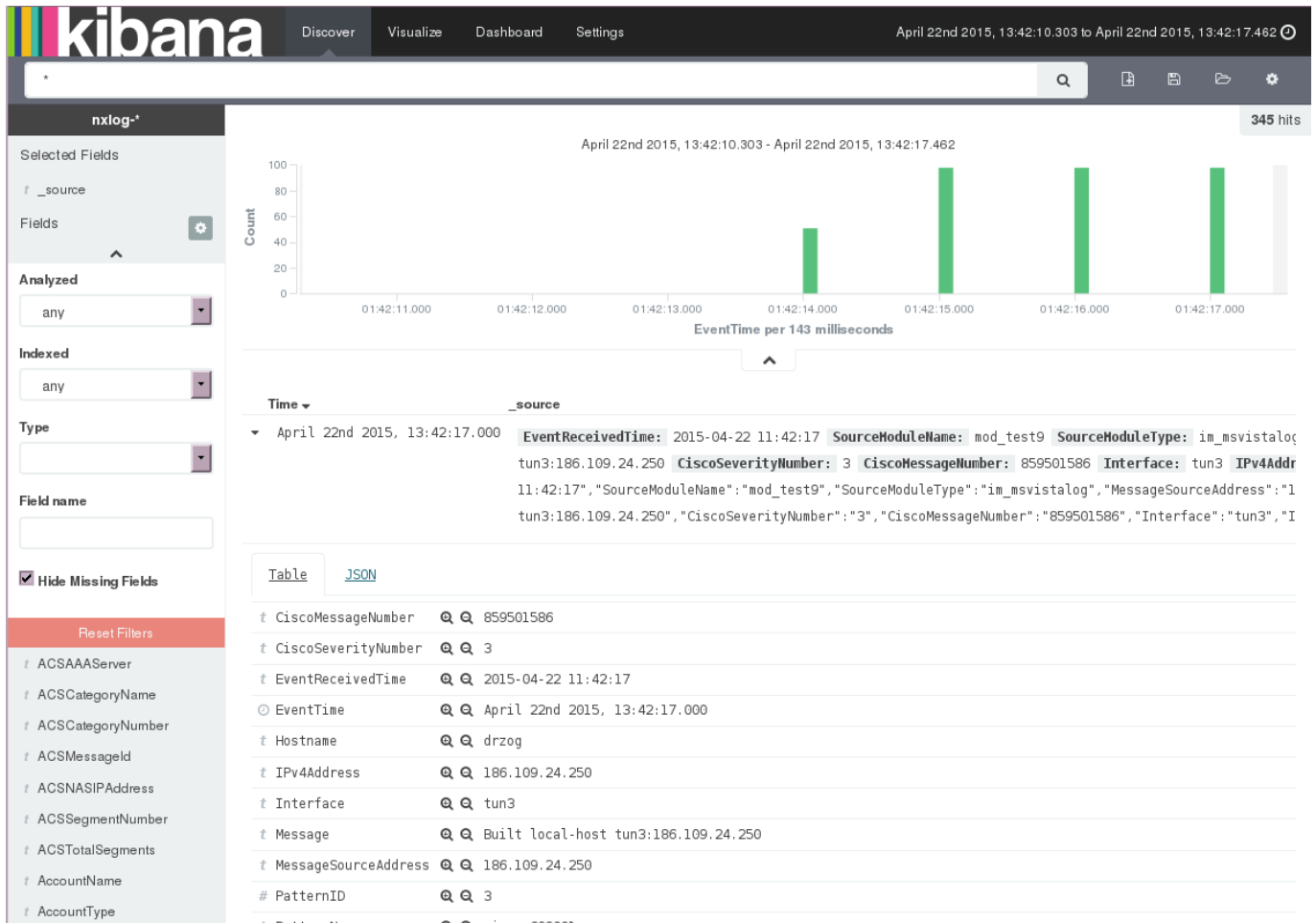


Figure 2: Kibana search results

## 2.2 Loading data with om\_http

Data can be also submitted to Elasticsearch via its HTTP REST API using the `om_http` module available in both NXLog editions. This will send one event per HTTP request to ES in JSON format. The throughput is limited by the HTTP request-response latency, regardless, this may be still suitable for low-volume environments.

The nxlog configuration below is for a Windows client that collects log data from a file and the Windows EventLog and sends it to the Elasticsearch server directly.

```
define ROOT C:\Program Files (x86)\nxlog

Moduledir %ROOT%\modules
CacheDir %ROOT%\data
Pidfile %ROOT%\data\nxlog.pid
SpoolDir %ROOT%\data
LogFile %ROOT%\data\nxlog.log

<Extension json>
  Module xm_json
</Extension>

<Input internal>
  Module im_internal
</Input>
```

```

<Input myapp>
  Module im_file
  File "C:\\MyApp\\Logs\\mylog.json"
  Exec parse_json();
  Exec $EventTime = parsedate($timestamp);
</Input>

<Input eventlog>
  Module im_msvistalog
</Input>

<Output elasticsearch>
  Module om_http
  URL http://elasticsearch:9200
  ContentType application/json
  Exec set_http_request_path(strftime($EventTime, "/nxlog-%Y%m%d/" + ↵
    $SourceModuleName)); rename_field("timestamp","@timestamp"); to_json();
</Output>

<Route es>
  Path internal, eventlog, myapp => elasticsearch
</Route>

```

## 2.3 Using Logstash

Logstash is a great piece of software and may offer some additional features that can be worth the extra hassle of installing it. On the other hand most users have Logstash running in their stack simply because they don't know that it's not mandatory.

Below is a client side NXLog configuration to collect file and Windows Eventlog data. The data is sent to Logstash over a TCP connection in JSON format.

```

define ROOT C:\Program Files (x86)\nxlog

Moduledir %ROOT%\modules
CacheDir %ROOT%\data
Pidfile %ROOT%\data\nxlog.pid
SpoolDir %ROOT%\data
LogFile %ROOT%\data\nxlog.log

<Extension json>
  Module xm_json
</Extension>

<Input internal>
  Module im_internal
</Input>

<Input file>
  Module im_file
  File "C:\\MyApp\\Logs\\mylog.json"
</Input>

<Input eventlog>
  Module im_msvistalog
# Uncomment if you want only specific logs
# Query <QueryList>\
#     <Query Id="0">\
#         <Select Path="Application">*</Select>\
#         <Select Path="System">*</Select>\
#         <Select Path="Security">*</Select>\

```

```
#           </Query>\
#           </QueryList>
</Input>

<Output logstash>
  Module      om_tcp
  Host        10.1.1.1
  Port        3515
  Exec        to_json();
</Output>

<Route 66>
  Path        file, eventlog, internal => logstash
</Route>
```

The following configuration is for the server-side Logstash process accepting event data from the NXLog clients.

```
input {
  tcp {
    codec => json_lines { charset => CP1252 }
    port => "3515"
    tags => [ "tcpjson" ]
  }
}
filter {
  date {
    locale => "en"
    timezone => "Etc/GMT"
    match => [ "EventTime", "YYYY-MM-dd HH:mm:ss" ]
  }
}
output {
  elasticsearch {
    host => localhost
  }
  stdout { codec => rubydebug }
}
```

---

**Note**

The `json` codec in Logstash sometimes fails to properly parse JSON and it will concatenate more than one JSON record into one event. Make sure to use the `json_lines` codec.

---

**Note**

Although the `im_msvistalog` module converts data to UTF-8, Logstash seems to have issues parsing that data. Apparently the `charset => CP1252` can help in such situations.

---

In this tutorial we have tried to show how to set up the ELK (or KEN) stack and send data to it. The tools are very flexible, both Logstash and NXLog come with a lot of extras such as message rewrite, parsing and field extraction, log rotation, correlation, TLS support and much more. While the configurations shown here mostly focused on using NXLog on Windows as a client side collector, NXLog can be also configured to collect file, syslog and kernel logs on Linux systems and forward that to Logstash, to an NXLog server or directly to Elasticsearch the same way.

Feel free to [drop us a line](#) if you have any questions and comments or are interested in the enterprise offering and support.

---