

# Nagios Log Server - Full Architecture Overview

Article Number: 98 | Rating: Unrated | Last Updated: Thu, Jul 28, 2016 at 4:29 PM

## High-Level Overview

Nagios Log Server is an application that provides organizations a central location to send their machine generated event data (e.g., Windows Event Logs, Linux syslogs, mail server logs) to store the data for later retrieval, querying, and analysis in near real-time.

Once log data has been indexed (indexing usually happens within 5 seconds from arrival) it can be easily analyzed using the graphical query and filtering tools on the dashboard. Log data can be queried using Google, Bing, or Stack Overflow. Alerts can be created based on a query, and are able to be sent via email to users of your choice. Alerts can also be sent to Nagios XI/Nagios Core via NRD

Field	Action	Value	Search
@timestamp	Q 🔍	2015-06-04T21:01:01.000Z	Q ▼
@version	Q 🔍	1	Q ▼
_id	Q 🔍	3yw-kkhSwysMtael1BzWQ	Q ▼
_index	Q 🔍	logstash-2015.06.04	Q ▼
_type	Q 🔍	syslog	Q ▼
facility	Q 🔍	9	Q ▼
facility_label	Q 🔍	clock	Q ▼
host	Q 🔍	localhost.localdomain	Q ▼
logsource	Q 🔍	localhost	Q ▼
message	Q 🔍	(nagios) CMD (/usr/bin/php -q /var/www/html/nagioslogserver/www/index.php poller > /usr/local/nagioslogserver/var/poller.log 2>&1)	Q ▼
pid	Q 🔍	21934	Q ▼
priority	Q 🔍	78	Q ▼
program	Q 🔍	CROND	Q ▼
severity	Q 🔍	6	Q ▼
severity_label	Q 🔍	Informational	Q ▼
tags	Q 🔍	dns	Q ▼
timestamp	Q 🔍	Jun 4 16:01:01	Q ▼
type	Q 🔍	syslog	Q ▼

The data that is sent to Nagios Log Server can be automatically archived to a shared network drive. The archived data can be restored and re-analyzed at any point in the future.

What that means in plain English is that it can be used to record any log events that are happening across all of the machines and network devices organization-wide. Users of Log Server can query the data in one location has the benefit of being able to compare or correlate log data from multiple devices. Also, the automated archiving of the log data will assist in maintaining large amounts of time.

### Most Common Use Case

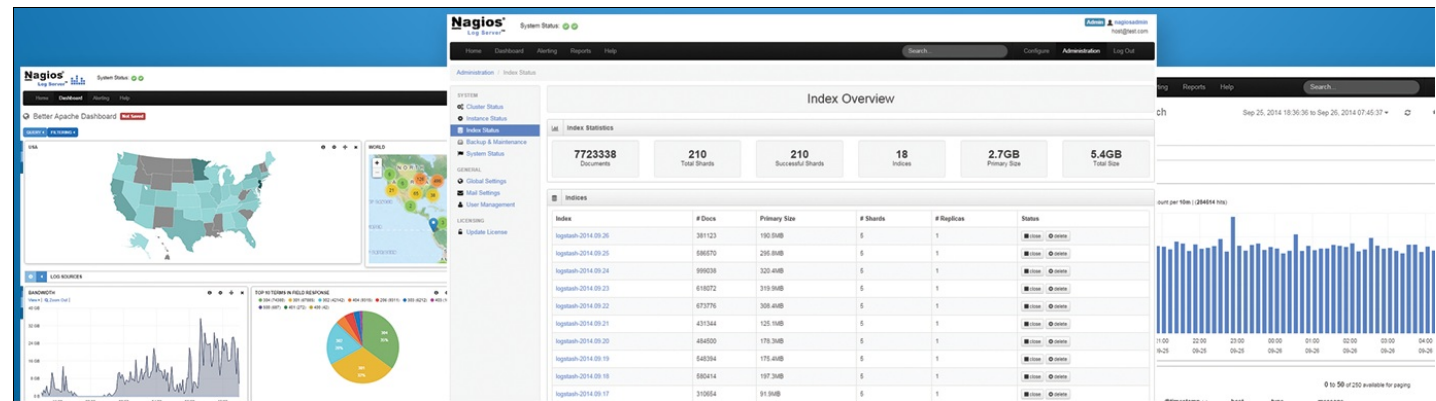
- An obvious application where Log Server could be used would be as an advanced system to analyze the received logged events and send important items (e.g. Critical Errors) to a central location for analysis.

### Less Common Use Cases

- Developers can send debug logs to Log Server, and easily filter out the information that isn't important leaving just the key items of interest.
- Organizations can utilize the graphical and analytic capabilities of Log Server to analyze web server logs, not only for errors, but to determine what are the most requested pages.
- With a small script, users could archive Nagios XI or Core check results including performance data, and have the ability to set up custom dashboards visualizing the data how they want.
- Log Server could be used to index and archive messages from an IMAP mailbox, for security or historical reference.
- Log Server can also receive SNMP traps, again allowing all of the previous functionality on the traps received.

### The Benefits of Nagios Log Server Over Text-Based Systems

Nagios Log Server allows all of your organization's machine generated data to be stored and indexed in one central location, allowing for queries to be performed on all of the log data. The data can be presented to the user running the query in customized views called dashboards, including a table of results, bar charts, pie charts, line graphs, etc. for any of the fields of data. Also, when creating/using the graphing/table functionality to provide data like total, min, max, mean, etc.



## Important Terms and Vocabulary

Nagios Log Server is a combination of three different open-source components: Elasticsearch, Logstash, Kibana. (ELK)

**Elasticsearch:** The scalable and redundant datastore used by Log Server.

**Logstash:** The log receiver for Log Server – Logstash outputs logs to the Elasticsearch database.

**Kibana:** The visualization component of the ELK stack – it is used to produce dashboards, made up of tables, graphs, and other elements.

Elasticsearch, Logstash, and Kibana work together to ultimately produce log visualizations.

**Cluster:** A Cluster is composed of two or more instances. "Cluster" is the term used to refer to all of the instances at once, working together. (e.g. The cluster is operating at wonderland)

**Instance:** A single installation of Nagios Log Server. Several connected instances make up a cluster. Typically instances are installed on separate virtual or physical machines.

**Index:** You can think of an index like a typical relational database. Indices are responsible for mapping to primary and replica shards. An index must map to one or more primary shards.

**Shard:** A shard is a low-level 'worker' which is managed by Elasticsearch. A shard can be primary or a shard can be a replica. By default, ten shards make up every index – five primary and five replicas.

**Primary Shard:** Every log entry in Nagios Log Server is stored in a primary shard. After the shard has been indexed on the primary shard, it is duplicated to a replica shard. In this way, data is replicated across multiple shards.

**Replica Shard:** By default, each primary shard (5) has a replica shard (5), making up a total of (10) shards. A replica shard is simply a copy of a primary shard, and is always available for availability. If a primary shard goes down, a replica shard will exist and be capable of taking the role of the missing primary shard. Replica shards are also suitable for performance. Adding more replica shards to the Nagios Log Server cluster improves performance.

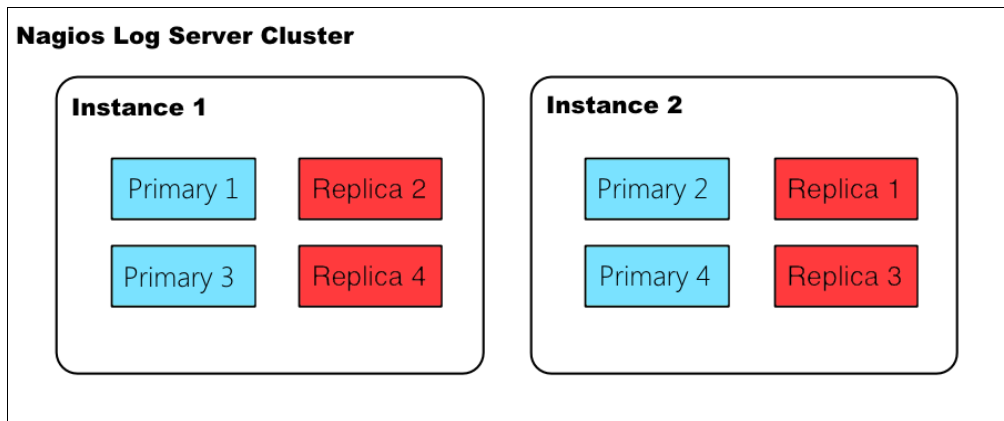
## How Nagios Log Server differs from the ELK stack

- Nagios Log Server (NLS) comes pre-configured for optimal performance, so system administrators can spend more time producing log visualizations, and less time tuning their Elasticsearch cluster.
- NLS is fully supported by Nagios support staff - by phone, email, or forum.
- NLS comes with authentication and security built-in. Typically, the ELK stack is open to whoever wants to query it – meaning that system administrators need to spend time developing authentication and security.
- NLS has an alert system built in - you are capable of alerting based on log queries via email. It can also tie in with several other Nagios products, including Nagios XI and Nagios Core.
- In general, NLS reduces management overhead and complexity of the ELK stack, and includes built-in security and alerting.

## Understanding Elasticsearch

Elasticsearch is a transparent component – it does not need a lot of tuning, especially if your cluster is small.

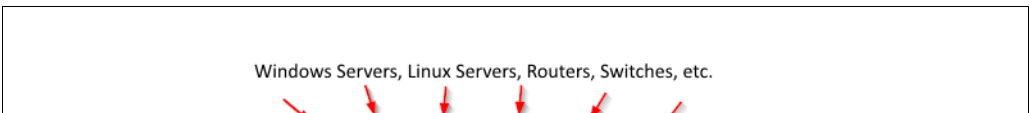
Elasticsearch is the database of choice because it is distributed and redundant by default. Every time an instance is added to your cluster, Elasticsearch ensures that its database is replicated across multiple shards, which increases the resiliency of the data.

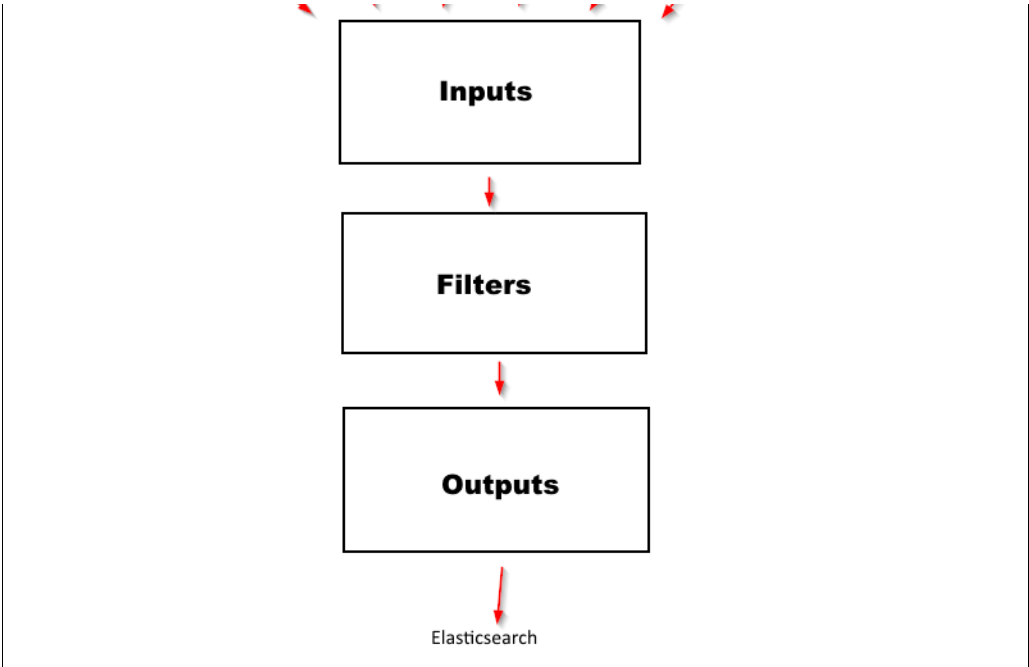


In the above picture, there are two instances joined in a single cluster. Each instance contains two primary shards (colored blue) and two replica shards (colored red). Note that Elasticsearch is distributed across multiple instances, which is how high availability is achieved.

## Understanding Logstash

Logstash is the most complex component in Nagios Log Server that an administrator will have to deal with. It is imperative that a good understanding of logstash is developed. The Logstash receiver receives incoming logs, and pass those logs to the filter chain, filters modify them, outputs ship them elsewhere - in our case, to the elasticsearch database.





Inputs

Inputs listen for incoming logs. The three most common inputs by far are TCP, UDP, and syslog. The tcp input will listen on a specified TCP port, and accept any logs coming in on that port. Things get a little more complex - every log that comes into the syslog input will automatically have the 'syslog' filter applied. For reference, the syslog filter is a simple grok filter

```
"match" => { "message" => "<{%POSINT:priority}>{%SYSLOGLINE}"
```

For additional information on inputs, the elasticsearch documentation is valuable: <https://www.elastic.co/guide/en/logstash/current/plugins-inputs-elasticsearch.html>

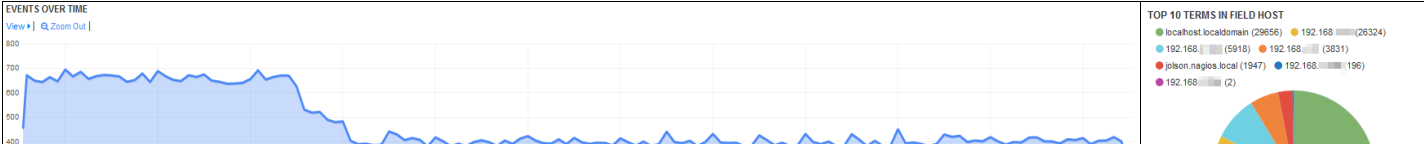
Filters

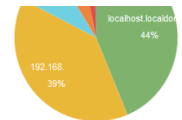
Filters are the most complex and important part of the Logstash chain. If you spend time learning anything, learn filters. It will help if you have a background in regular expression. Regular expressions are used to generate your own filters. There are many free tutorials online to learn from.

Logstash filters parse logs passed down from the input chain to 'filters' that you define. Before filters are applied, your logs are likely unstructured and have no 'fields' applied to them.

Field	Action	Value
@timestamp	Q [icon]	2015-06-05T18:05:01.000Z
@version	Q [icon]	1
_id	Q [icon]	G-NWfrn-SISGAST5v6yYDQ
_index	Q [icon]	logstash-2015.06.05
_type	Q [icon]	syslog
facility	Q [icon]	9
facility_label	Q [icon]	clock
host	Q [icon]	localhost.localdomain
logsource	Q [icon]	localhost
message	Q [icon]	(nagios) CMD (/usr/bin/php -q /var
pid	Q [icon]	18472
priority	Q [icon]	78
program	Q [icon]	CROND
severity	Q [icon]	6
severity_label	Q [icon]	Informational
tags	Q [icon]	dns
timestamp	Q [icon]	Jun 5 13:05:01
type	Q [icon]	syslog

Fields are important because they enable the creation of graphs and visualizations:





Outputs

By default, our output is responsible for exporting data to elasticsearch. You can also define [custom outputs](#) to do many other things - from sending alerts to Nagios via the [Nagios o](#)

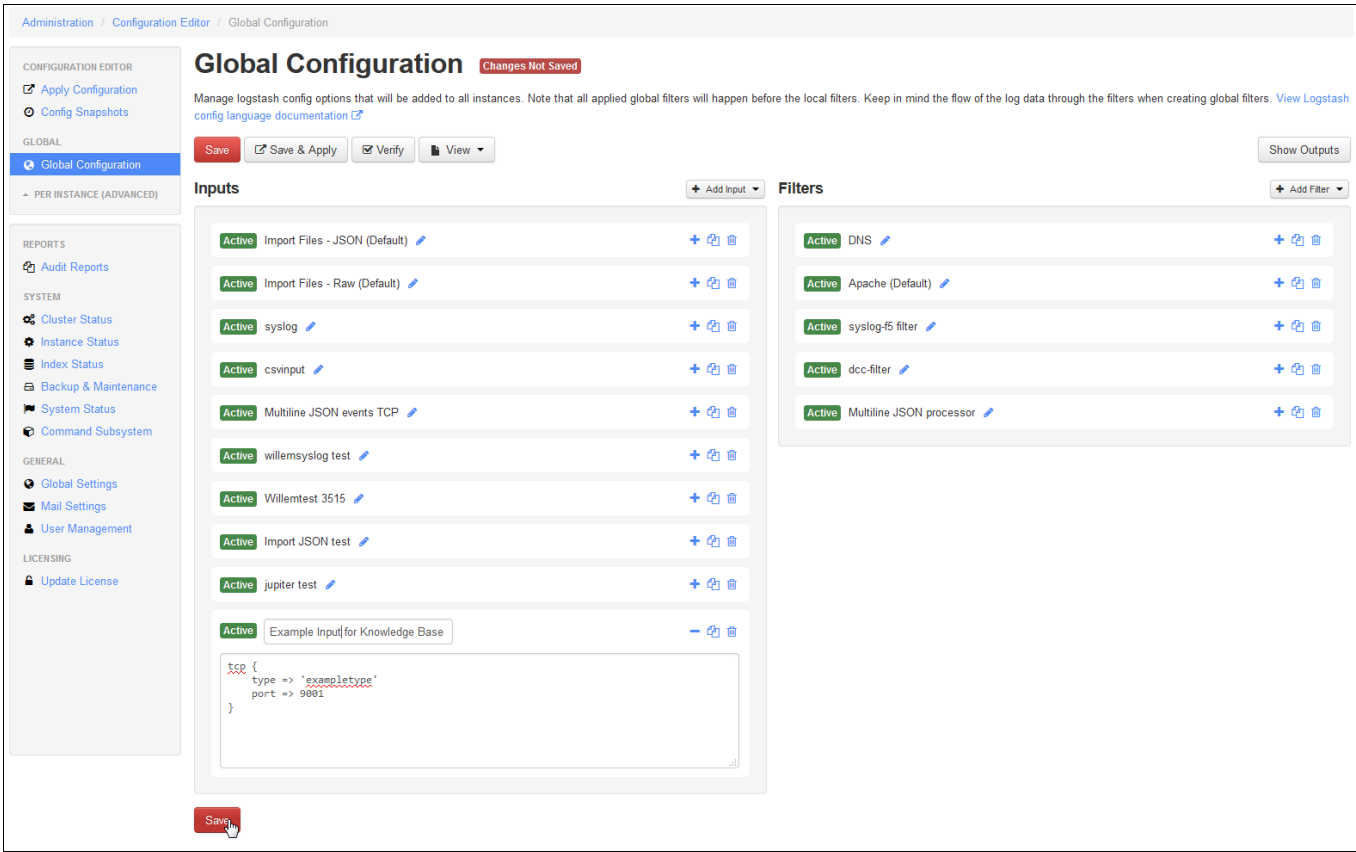
Logstash Example Configuration

The best way to explain Logstash is by example. In this example, we will be using a basic 'tcp' input and the 'grok' filter - this is a widely-used filter. Grok is a filter that matches regex here: <https://www.elastic.co/guide/en/logstash/current/plugins-filters-grok.html>

First, we define an input to take logs in. For the sake of simplicity, the following input will be used:

```
tcp {
  type => 'exampletype'
  port => 9001
}
```

You can add this configuration to the Web GUI quite easily:



Be sure to 'Save & Apply' when you are finished. You will have to open the appropriate firewall port on Nagios Log Server.

Now, we'll start sending logs to Logstash from our remote host. Once a log arrives, you will see it displayed in the Web GUI as unstructured information, like so:



0 to 2 of 2 available for paging

@timestamp ▼	host	type	message	Actions
2015-06-05T13:34:14.037-05:00	192.168.1.1	exampletype	[Sun May 31 19:00:08 2015] [error] [client 192.168.1.1] File does not exist: /usr/local/nagiosxi/html/includes/js/jquery/jquery-1.8.2.min.js, referer: http://192.168.1.2/	Q

View: Table / JSON / Raw

Field	Action	Value	Search
@timestamp	Q	2015-06-05T13:34:14.037Z	Q
@version	Q	1	Q
_id	Q	zEmHTodDRMuAXXOb14ZuwQ	Q
_index	Q	logstash-2015.06.05	Q
_type	Q	exampletype	Q
highlight	Q	[object Object]	Q
host	Q	192.168.1.1	Q
message	Q	[Sun May 31 19:00:08 2015] [error] [client 192.168.1.1] File does not exist: /usr/local/nagiosxi/html/includes/js/jquery/jquery-1.8.2.min.js, referer: http://192.168.1.2/	Q
tags	Q	dns	Q
type	Q	exampletype	Q

Logstash will attach certain fields by default - @timestamp, @version, \_id, \_index, \_type, etc. were all generated automatically. The bulk of the sent information is located in the 'message' field.

At this point, the question to ask is what information in the 'message' field is relevant to you. Since I don't know this, I will define this filter by my standards - you are of course free to put any relevant information into appropriate fields.

The [grok debug](#) utility makes our work rather simple. I also like to keep the [grok patterns](#) at hand when defining a filter.

Grok Debugger    Debugger    Discover    Patterns

[Sun May 31 19:00:08 2015] [error] [client 192.168.1.1] File does not exist: /usr/local/nagiosxi/html/includes/js/jquery/jquery-1.8.2.min.js, referer: http://192.168.1.2/

\[%{DAY:day} %{MONTH:month} %{MONTHDAY:monthday} %{TIME:time} %{YEAR:year}\} \[%{WORD:status}\} \[client %{IP:client}\} \[%{GREEDYDATA:info}\}

☐ Add custom patterns   
 ☐ Keep Empty Captures   
 ☐ Named Captures Only   
 ☐ Singles   
 ☐ Autocomplete

```

{
  "day": [
    [
      "Sun"
    ]
  ],
  "month": [
    [
      "May"
    ]
  ],
  "monthday": [
    [
      "31"
    ]
  ],
  "time": [
    [
      "19:00:08"
    ]
  ],
  "hour": [
    [
      "19"
    ]
  ],
  "minute": [
    [
      "00"
    ]
  ]
}

```

After debugging your log message appropriately, you will wind up with a grok pattern. A very basic grok pattern (developed in a few minutes) is as follows:

```
\[%{DAY:day} %{MONTH:month} %{MONTHDAY:monthday} %{TIME:time} %{YEAR:year}\} \[%{WORD:status}\} \[client %{IP:client}\} \[%{GREEDYDATA:info}\}
```

In this pattern, I have tagged what is important with fields. Note that anything in CAPITAL letters is simply a pre-defined regex pattern that grok provides. After the colon comes the field name that I want to graph, I need to use the :int suffix on. An int pattern might look something like:

```
%{NUMBER:loginretries:int}
```

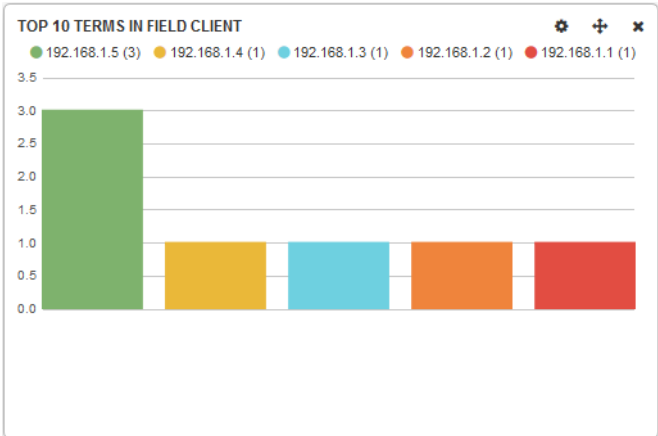
Now we need to put this pattern into a filter. Let's quickly define a grok filter:

```
if [type] == "exampletype" {
  grok {
    match => [ "message", "\[%{DAY:day} %{MONTH:month} %{MONTHDAY:monthday} %{TIME:time} %{YEAR:year}\} \[%{WORD:status}\} \[client %{IP:client}\} \[%{GREEDYDATA:info}\}"
  ]
}
```

Once implemented, anything matching 'exampletype' will be passed through this filter. In our case, the input automatically tags everything coming in as 'exampletype'. The end result could be something like:

@timestamp ▾	host	type	message	Actions
2015-06-08T14:57:52.050-05:00	192.168.1.1	exampletype	[Sun May 31 19:00:08 2015] [error] [client 192.168.1.1] File does not exist: /usr/local/nagiosxi/html/includes/js/jquery/jquery-1.8.2.min.js, referer: http://192.168.1.2/	<div>Q ▾</div>
View: <a href="#">Table</a> / <a href="#">JSON</a> / <a href="#">Raw</a>				
Field	Action	Value	Search	
@timestamp	<div>Q ▾</div>	2015-06-08T19:57:52.050Z	<div>Q ▾</div>	
@version	<div>Q ▾</div>	1	<div>Q ▾</div>	
_id	<div>Q ▾</div>	qQ5nqXizSXWfyu1Xlr2CRA	<div>Q ▾</div>	
_index	<div>Q ▾</div>	logstash-2015.06.08	<div>Q ▾</div>	
_type	<div>Q ▾</div>	exampletype	<div>Q ▾</div>	
client	<div>Q ▾</div>	192.168.1.1	<div>Q ▾</div>	
day	<div>Q ▾</div>	Sun	<div>Q ▾</div>	
highlight	<div>Q ▾</div>	[object Object]	<div>Q ▾</div>	
host	<div>Q ▾</div>	192.168.1.1	<div>Q ▾</div>	
information	<div>Q ▾</div>	File does not exist: /usr/local/nagiosxi/html/includes/js/jquery/jquery-1.8.2.min.js	<div>Q ▾</div>	
message	<div>Q ▾</div>	[Sun May 31 19:00:08 2015] [error] [client 192.168.1.1] File does not exist: /usr/local/nagiosxi/html/includes/js/jquery/jquery-1.8.2.min.js, referer: http://192.168.1.2/	<div>Q ▾</div>	
month	<div>Q ▾</div>	May	<div>Q ▾</div>	
monthday	<div>Q ▾</div>	31	<div>Q ▾</div>	
status	<div>Q ▾</div>	error	<div>Q ▾</div>	
tags	<div>Q ▾</div>	dns	<div>Q ▾</div>	
time	<div>Q ▾</div>	19:00:08	<div>Q ▾</div>	
type	<div>Q ▾</div>	exampletype	<div>Q ▾</div>	
url	<div>Q ▾</div>	http://192.168.1.2/	<div>Q ▾</div>	
year	<div>Q ▾</div>	2015	<div>Q ▾</div>	

Ultimately, we can generate beautiful Kibana visualizations using these new fields.



Additional filters can be found here: <https://www.elastic.co/guide/en/logstash/current/filter-plugins.html>

## Troubleshooting Commands

These are the commands that are most often used to troubleshoot misbehaving clusters:

### Generic Information Gathering

Master:

```
curl 'localhost:9200/_cat/master?v'
```

#### Nodes:

```
curl 'localhost:9200/_cat/nodes?v'
```

#### Pending tasks:

```
curl 'localhost:9200/_cat/pending_tasks?v'
```

#### Recovery:

```
curl -XGET 'localhost:9200/_cat/recovery?v'
```

#### Running plugins:

```
curl 'localhost:9200/_cat/plugins?v'
```

#### Cluster basic health:

```
curl -XGET 'http://localhost:9200/_cluster/health?pretty=true'
```

#### Check jvm settings:

```
curl -XGET localhost:9200/_nodes/jvm?pretty
```

### Hail Mary Information Gathering

#### Shard status (use index status instead for clarify, in most cases):

```
curl -XGET http://localhost:9200/_cat/shards
```

#### Cluster state:

```
curl -XGET 'http://localhost:9200/_cluster/state?pretty'
```

#### Shard health (index health works better in most circumstances):

```
curl -XGET 'http://localhost:9200/_cluster/health/*?level=shards'
```

### Red Cluster Recovery

#### Check index health:

```
curl 'localhost:9200/_cluster/health?level=indices&pretty'
```

#### Shard list:

```
curl -s localhost:9200/_cat/shards
```

#### Unassigned shards:

```
curl -s localhost:9200/_cat/shards | grep UNASS
```

#### Delete bad index:

```
curl -XDELETE 'http://localhost:9200/twitter/'
```

### Active Troubleshooting Commands

#### Tail maintenance logs:

```
tail -f /usr/local/nagioslogserver/var/jobs.log
```

#### Tail poller logs:

```
tail -f /usr/local/nagioslogserver/var/poller.log
```

#### Shut down all nodes:

```
curl -XPOST 'http://localhost:9200/_shutdown'
```

#### Check users:

```
curl -XGET 'http://localhost:9200/nagioslogserver/_search?type=user&pretty'
```

### Backup Troubleshooting Commands

#### Check running knapsack export:

```
curl -XPOST 'http://localhost:9200/_export/state'
```

Typically jobs.log should be tailed and the backup+maintenance command should be forced from the subsys - jobs.log should complain about a partial snapshot.

#### Delete a snapshot (versions <= 1.3.0):

```
curator delete --older-than 1 --prefix logstash-2015.05.19 (replace with snapshot name)
```

#### (versions >= 1.3.0):

```
curator delete snapshots --repository nlsbackup --prefix (replace with snapshot prefix)
```

You can also delete snapshots from the Web GUI under the backup page, or directly from the repo.

### Purpose-Driven Commands

Check min master nodes:

```
curl localhost:9200/_cluster/settings?pretty
```

Set min master nodes:

```
curl http://localhost:9200/_cluster/settings -XPUT -d '{ "persistent": { "discovery.zen.minimum_master_nodes": 2 } }'
```

Check users:

```
curl -XGET 'http://localhost:9200/nagioslogserver/_search?type=user&pretty'
```

## Important Files and Directories

---

elasticsearch.yml = /usr/local/nagioslogserver/elasticsearch/config/elasticsearch.yml

This file is responsible for low-level Elasticsearch configuration.

logging.yml = /usr/local/nagioslogserver/elasticsearch/config/logging.yml

This file can be used to set logging levels in the case that Elasticsearch needs to be debugged.

elasticsearch = /etc/sysconfig/elasticsearch

This file is responsible for high-level Elasticsearch configuration.

logstash = /etc/sysconfig/logstash

This file is responsible for high-level Logstash configuration.

cluster\_hosts = /usr/local/nagioslogserver/var/cluster\_hosts

This file contains the IP addresses of all known instances in the cluster, in addition to 'localhost'. If you are having issues with node visibility, this is a good file to check.

cluster\_uuid = /usr/local/nagioslogserver/var/cluster\_uuid

This file contains a unique string used to identify the cluster. This file should be the same across all instances in the cluster.

## Final Thoughts

---

For any support related questions please visit the [Nagios Support Forums](#) at:

<http://support.nagios.com/forum/>

Posted by: **jolson** - Thu, Jun 4, 2015 at 4:01 PM. This article has been viewed 2547 times.

Online URL: <https://support.nagios.com/kb/article.php?id=98>