Table of Contents

1. Basic Installation of the ELK stack

This book is a short intro to the ELK stack. ELK is an acronym for Elasticsearch,Logstash and Kibana. At the core of the ELK stack is Elasticsearch, the most important and mature component on which the ELK stack is built on. Elasticsearch is going to be a very important opensource project in the time we live in as more and more software is built around using Elasticsearch. Elasticsearch allows users to make sense of large amounts of data and to find novelty in the overflow of data that seems unimportant. Instead of allowing data to stay dormant in large data warehouses , elasticsearch allows users to find important data and make use of it. In the current time we live in one of the biggest problems facing the world today is not the ability to accumulate data, but the ability to make sense of all of the data that we accumulate. Both yahoo and amazon for example have large open data sets that are freely available to the public on request, these datasets contain all kinds of interesting information such as information about the competitors etc. The only problem is to make sense of the data requires exploring and large amounts of effort and processing power. If you take the time to learn to use the ELK stack, you will be able make very interesting findings out of seeminly boring or mundane data.

The Story of Elasticsearch

You can read more about the story of how the open source project Elasticsearch started in "Elasticsearch The Definite Guide": Chapter 1: The Mists of Time. To give a quick summary of the story, Elasticsearch started out as a project that, the then unemployed developer: Shay Banon, started while his wife was studying to become a chef. While playing with one of the early versions of Lucene, Shay come up with the idea of creating a recipe search engine for his wife. This recipe search engine apparently never came into excistance, but an open source project by the name of Compass came out of Shay’s thoughtful efforts to build something for his wife. He later went on to rewrite the Compass libraries as a standlone server called Elasticsearch.

To quote from Elasticsearch The Definite Guide: "Shay’s wife is still waiting for the recipe search…".

Lucene. You might see the name Lucene mentioned several places. Lucene is an open source project also written in Java, Lucene can best described as a set of Java libraries that can be used for full text search. Elasticsearch was initially an API of its own that made use of Lucene,back when it

was called "Compass". Elasticsearch was then created by making compass into a standalone server that could be used directly without writing any code. One of the strongpoints of Lucene is that it is a very mature open source project with a very mature code base. Lucene has been around for years, but thanks to Elasticsearch , Lucene has no become more well known to the rest of the world.

Logstash. Logstash is a tool used to process logs and event related data from variety of sources and systems. Logstash is written in Ruby. Logstash can process almost any kind of data and normalize it. Logstash can help you normalize data to a common format before sending the data on forward to a data store, in the case of the ELK stack the data store is Elasticsearch. Logstash is used to normalize data before sending it off to Elasticsearch. Logstash is a very powerful tool to have in your toolbox, Logstash can really be very useful if data is not in a format that is easy to index into Elasticsearch.

Elasticsearch. Some people consider Elasticsearch as a data store with many similarities to a relational database, but also with many other features that are not at all similiar to the features of a relational database. Elasticsearch is written in Java. Elasticsearch can search through gigantic amounts of data in real time. Elasticsearch is a standalone server based on Lucene, it is meant to be used to conduct full text search on text. If you think back of the story of how Elasticsearch started, Shay was looking to build a tool that could search through many different recipe books for specific keywords and return a result to the query, such as the full recipe or details such as where to find the recipe. Elasticsearch has a rest api that returns responses in JSON.

Kibana: Kibana is a Javascript based web application and front end for data stored in an Elasticsearch cluster. Kibana can be used run Elasticsearch queries and to create visual representations of the results of complex Elasticsearch queries. The ELK stack is made up of logstash receiving data,processing it and sending it on to an Elasticsearch cluster. Kibana then queries the Elasticsearch cluster and displays a the results of the query on a webinterface. As of Kibana 4.2, Kibana also has the capability to install plugins, thus you can now install plugins for Kibana and for Elasticsearch. This is a very interesting feature which will increase the popularity of Elasticsearch immensely. The aim of this book is to give you a push in the right direction in terms of using the ELK stack,especially in the context of using the ELK stack for operational usage.This book is aimed at devops enthusiasts, system administrators, network administrators and people working in security related operational jobs.

In the time we live in it is becoming increasingly difficult to manage systems in an enterprise environment due to constant increase in the complexity of systems and also the size of networks. Monitoring systems for uptime an for security related events is also becoming more important as the attack surface grows for nation state hackers and organized crime organizations dedicated completely to hacking corporate and government networks. There are very few solutions to help system administrators for example point out suspicious behavior in log files. There is the old school tools such as Logwatch which can point out specific anomolies in log files and alert the system admin via email.

There are quite a few security related open source projects emerging that are based on the ELK stack. One of these is mozdef: Mozilla’s MozDef [https://mozdef.readthedocs.org/en/latest/] This project uses the moto:"You’ve collected your security logs, now what?". This raises an interesting question also when it comes to logs related even to day to day operations. Why are we collecting so many logs on so many devices for so many services if nobody is reviewing these logs. What would the benefit be of reviewing our log files?

Back To Elasticsearch. Elasticsearch is being used by many of the largest sites on the internet today: Netflix,Github,Yelp and wikipedia, just to name a few. Github uses Elasticsearch to search over billions of lines of code. Wikipedia uses Elasticsearch to search over millions of articles, but also

uses the ELK stack for monitoring of their infrastructure and application logs. I can only speculate that organizations such as the NSA and GCHQ make use of Elasticsearch extensively to make sense of the gigantic amount of data that they have to process on a daily basis.

Wikipedia or the wikimedia organization is a very interesting case to learn from, because they publish a large amount of information on their implementation of the ELK stack on the internet for anyone to read and learn from. You can see more here: Logstash at Wikipedia [https://wikitech.wikimedia.org/wiki/Logstash] Also has a section on Elasticsearch specifically which is very well documented. Elasticsearch at wikipedia [https://wikitech.wikimedia.org/wiki/Search] Elasticsearch is being used by many startups in their new products. Elasticsearch is also used at big companies such as Yelp and Netflix.

If you compare Elasticsearch to a relational database, an index is the equivalent of a database in elasticsearch. What is an Elasticsearch Index [https://www.elastic.co/blog/what-is-an-elasticsearch-index]

Basic Installation of the ELK stack. An important concept to understand is that if you are running elasticsearch, you are effectively running an Elasticsearch cluster, even if your cluster only has one node. Before you install the ELK stack you will need to do some thinking about how you plan on deploying your Elasticsearch cluster and what it will be used for. Would you want to host the ELK stack on a server in a data center or would you pay a third party provider to host the ELK stack for you in the cloud? If you plan on logging sensitive data to elasticsearch, then you should consider running your own instance on your own hardware. You should also consider the network layout if you are planning on putting Elasticsearch on your network. It would be ideal to put Elasticsearch on a DMZ and only allow certain machines to access Elasticsearch on the DMZ.You also need to make some calculations in terms of how much data you plan on logging to Elasticsearch relative to the hardrive space that you have available on your servers.

Installing the different components that make up the ELK stack is very easy. Just for the sake of security, it would be good to install Elasticsearch first on your local machine or local network bechind a firewall and to first get comfortable working with it before you start logging sensitive data to it.

Installation. The devlopment on the ELK stack moves very fast. To make sure you have the most up to date instructions refer to:

Quick Deployment Solutions

Installing ELK Stack Instructions: Installing the ELK Stack Instructions [https://github.com/elastic/examples/blob/master/Installation%20and%20Setup.md]

ELK Docker Setup [https://github.com/elastic/examples/tree/master/ELK_docker_setup]

Deploying Elasticsearch in Production: There is an officially supported puppet module [https://github.com/elastic/puppet-elasticsearch] There is also an officially supported Chef cookbook [https://github.com/elastic/cookbook-elasticsearch]

ELK Stack Docker Images. The official github examples repository for Elasticsearch has a very thorough guide on setting up the ELK stack using Docker. You can have a look at it here.

ELK Docker Setup [https://github.com/elastic/examples/tree/master/ELK_docker_setup]

Installing Elasticsearch [https://www.elastic.co/guide/en/elasticsearch/guide/current/_installing_elasticsearch.html]

You will need Java installed to run Elasticsearch and Logstash. The official Elasticsearch documentation recommends using the latest Oracle JDK to run Elasticsearch, although OpenJDK will also work. Java is the only requirement for being able to install Elasticsearch. The official documentation recommends always using the latest version of Java.

```
$ java -version
```

Development on the ELK stack moves very fast so you should install the .deb or zip files for logstash and elasticsearch. For the duration of this book we are assuming that you are using either Debian Or Ubuntu Server LTS. You can find the latest version of Elasticsearch here: http://www.elasticsearch.org/download/

```
$ mkdir elk

$ cd elk
```

You will need to install curl

```
$ sudo apt-get install curl
```

Now download the Elasticsearch .deb

```
$ curl -L -O http://download.elasticsearch.org/PATH/TO/ELASTICSEARCH_VERSION.deb
```

Now Install the package

```
$ sudo dpkg -i ELASTICSEARCH_VERSION.deb
```

Start Elasticsearch and interact with it.

```
$ sudo service elasticsearch start
```

List the indexes:

```
$ curl 'localhost:9200/_cat/indices'
```

You can create an index with the PUT http verb. The PUT verb is usually used in REST to specify a record that is being updated. You can think of it in this context as updating your Elasticsearch cluster with a new index.

```
$ curl -XPUT 'localhost:9200/index_name'

$ curl -XPUT 'localhost:9200/foo'
```

To make the output returned from running the requests to Elasticsearch more readable, you can add the pretty argument to the REST API. You can do it like:

```
$ curl -XPUT 'localhost:9200/foo?pretty=true'
```

To delete an index. You can delete an index using the DELETE HTTP verb.

```
$ curl -XDELETE 'localhost:9200/foo'
```

```
or

$ curl -XDELETE 'localhost:9200/foo?pretty=true'
```

> **Interacting with Elasticsearch**
>
> Most interaction with Elasticsearch happens via the REST API. The REST API also has wrappers for many common programming languages such as: Java and Python just to name two.
>
> See Elasticsearch article "Talking to Elasticsearch".https://www.elastic.co/guide/en/elasticsearch/guide/current/_talking_to_elasticsearch.html
>
> You can use any http client to interact with Elasticsearch. Many of the examples you will see online are shown using the http client curl. The Elasticsearch plugin Sense which has recently been opensourced is very useful for interacting with Elasticsearch. Sense uses a short hand REST notation to describe interaction with Elasticsearch, I will cover that more in detail later on.
>
> Curl Reference from Conquering The Commandline [http://conqueringthecommandline.com/book/curl]
>
> HTTP From The Commandline [https://web.archive.org/web/20150411002413/http://httpkit.com/resources/HTTP-from-the-Command-Line/]
>
> Take note of the shorthand notation that is often used when describing interactions with the Elasticsearch API, this notation is also used in the Sense opensource plugin which is an alternative to interacting with Elasticsearch as opposed to using curl.

Installing Elasticsearch Plugins: There are various plugins for Elasticsearch. This is the generic method to install them. If you have to install the plugins from behind a proxy that requires authentication, it would be best to download the plugins manually with curl and install them from file. If you are working on a corporate network at work you will most probably have to follow this method to install plugins as most corporate networks these days make use of an HTTP proxy that requires authentication. The plugin tool can install plugins from file, you just need to specify where they are located on your system.

```
$ cd /usr/share/elasticsearch

$ ./bin/plugin -i somedir/plugin_name
```

If you are on a network where you need to connect to the internet using a proxy, then you should use this syntax

```
# ./bin/plugin -DproxyPort=port_number -DproxyHost=proxy_ip -i somedir/plugin_name

# ./bin/plugin -DproxyPort=8080 -DproxyHost=196.168.1.4 -i somedir/plugin_name
```

Installing Sense: Sense is a web application for interacting with Elasticsearch. It can be very helpful if you are doing complex queries to use Sense as opposed to using curl on the commandline. The plugin "Sense" used to be one of the paid plugins for Elasticsearch, but was packaged as part of the Marvel plugin. Sense was recently Open sourced. In the Elasticsearch documentation the

shorthand notation is usually used to show the equivalent of a curl query, but in Sense. Sense uses the shorthand notation and make sure to know how the normal REST notation can be rewritten using the shorthand REST notation. The sense github repository is located here: Sense on Github [https://github.com/elastic/sense]

Installing Logstash: Just take note that Logstash and Logstash forwarder are two different things. Make sure to install the Logstash stable version and not the Logtash Beta version. Also take note that logstash forwarder will soon be replaced with "filebeat", which is also written in Golang.

```
$ wget https://download.elastic.co/PATH/TO/VERSION.deb

$ sudo dpkg -i logstash_VERSION.deb

$ sudo dpkg -i elasticsearch-VERSION.deb
```

You can start the elasticsearch and logstash services on Ubuntu/Debian

```
$ sudo service elasticsearch start
```

Start logstash. Make sure if it is not already running

```
$ ps aux | grep logstash


$ sudo service logstash start
```

Now install Kibana: Download and decompress.

```
$ wget https://download.elastic.co/kibana/kibana/kibana-VERSION-linux-x64.tar.gz


$ tar -zxvf kibana-VERSION-linux-x64.tar.gz


$ cd kibana-*


$ cd bin


$ ./kibana&
```

This will run Kibana in the babckground. It will be listening on Kibana Web Interface [http://yourip:5601] You should now see some JSON output. If you get the error:

```
"Error: No Living connections"
```

Then you will need to modify your Kibana configuration file.

```
$ cd ..

$ vi config/kibana.yml
```

Open your elasticsearch config:

```
$ sudo vi /etc/elasticsearch/elasticsearch.yml
```

Your config should look like this:

```
#
#network.bind_host: 0.0.0.0

# Set the address other nodes will use to communicate with this node. If not
# set, it is automatically derived. It must point to an actual IP address.
#
network.publish_host: 127.0.0.1

# Set both 'bind_host' and 'publish_host':
#
network.host: 127.0.0.1

# Set a custom port for the node to node communication (9300 by default):
#
#transport.tcp.port: 9300
```

To check if Elasticsearch is running:

```
$ curl -X GET "http://localhost:9200"
```

You can also check your cluster health with:

```
$ curl -X GET 'http://localhost:9200/_cluster/health'


$ cd bin
```

Start Kibana

```
$ ./kibana&
```

Running Kibana won't help much if you aren't sending any data to logstash.Lets send a log file to logstash so we can also create indices for Kibana. Some tips if you have trouble getting Kibana working:1. Try rebooting the box.2. Open Kibana in an incognito tab in your browser.3. Make sure Elasticsearch is running.

Now to log something. Let me explain how the ELK stack works conceptually:

```
logfile-logstash--parsed by logstash--elasticsearch-Kibana
```

REST Notation used: The usual notation used when describing interaction with Elasticsearch's REST api via curl is the following:

```
$ curl -XHTTP VERB  'protocol://host:port on which elasticsearch is running on'

or

$ curl -XGET 'http://127.0.0.1:9200'
```

The default port that Elasticsearch is running on is port 9200. Lets look more at the notation in depth:

```
$ curl -XHTTP VERB  'protocol://host:port on which elasticsearch is running on/PATH_TO_S

or

$ curl -XPUT 'http://127.0.0.1:9200/foo?pretty=true'
```

Query string with no Value:

```
/foo?pretty
```

The parameter pretty defaults to true if no value is passed to it. You can also send it multiple arguments, by combining them with the & operator:

/foo?pretty=true&someotherparameter=blah

You can also pass data in your interaction with Elasticsearch. Use the optional -d or --data switch.

```
$ curl -XPOST 'protocol://host:port on which elasticsearch is running on/PATH_TO_SOMETHI
```

POST /PATH?QUERY_STRING' -d *BODY*

You can also send data from a file using the -d or --data switch, this is very useful, because you can just copy and paste the json body into a file and just refer to the file every time instead of having to paste the json string on the commandline with your query.

```
$ curl -XPOST 'protocol://host:port on which elasticsearch is running on/PATH_TO_SOMETHI
```

Shorthand Notation. The shorthand notation used sometimes to describe interaction with the Elasticsearch rest API. The elasticsearch usually shows examples in both the normal and short hand notation. This the notation you will sometimes see in the Elasticsearch documentation when you click on "View in Sense".

```
$ curl -X GET'http://localhost:9200/_cat/indices'

...
$ curl -XGET 'localhost:9200/_cat/indices'
```

Uses the shorthand:

```
GET /_cat/indices
```

To view your clusterhealth for example:

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

Uses the shorthand:

```
GET /_cluster/health?pretty=true
```

This is the long version of a query:

```
$ curl -XGET 'localhost:9200/_count?pretty' -d '
{
"query": {
"match_all": {}
}
}'
```

This is the same query, but just in short hand notation. You should get used to the short hand version, which this often is used in the Elasticsearch documentation and by the Sense console.

```
GET /_count
{
"query": {
;"match_all": {}}
```

```
}
```

Important operations:

There are some important operations to memorize, try memorize them as they will come handy often. You can think of these operations as some of the most basic elasticsearch operations that you will reuse over and over again.

List indices:

```
$ curl 'localhost:9200/_cat/indices'
```

Create Index:

```
$ curl -XPUT 'localhost:9200/index_name'
```

For example:

```
$ curl -XPUT 'localhost:9200/foo'
```

The PUT http verb is used to create.

Delete Index: The DELETE HTTP verb is used to remove.

```
$ curl -XDELETE 'localhost:9200/foo'
```

2. Basic Configuration:

Choosing the right hardware and paying attention to configuration of your system can have a major impact on the performance of the service that you plan on running. In the case of Elasticsearch and Logstash, paying attention to configuration can really save you lots of headache or even your reputation. Imagine if you installed the ELK stack at a client, but you forgot to spend some time to ensure that you configured it properly to handle a large amount of data, and the Elasticsearch cluster crashed. This could really rune the trust relationship between you and your client, but also the client's trust in your technical abilities.

When it comes to configuration of a service on a server, it really pays to spend as much time as possible to read the documentation and available forum posts, blog posts and anything you can find to help you understand what is the best and most efficient way to configure the service. It is very important to understand why this is the best way to configure the service. It also pays to test your configurations and to test the different scenarios that you anticipate your server could find difficult to handle. For example if you are setting up Elasticsearch with multiple nodes in the cluster, it would be useful to test what happens if one of the nodes in the cluster is shut down while the others are running, by actually shutting the nodes down one by one while the rest of the cluster is still running. There are thousands of different ways to configure the ELK stack and for so many different use cases. Trying to configure the ELK stack for your specific use case can be confusing. If you get stuck one of the best resources for consulting others about the ELK stack is the official elastic.co forum.

Official Elasticsearch Forum [https://discuss.elastic.co]

Taming a Wild Elasticsearch Cluster [http://www.wilfred.me.uk/blog/2015/01/31/taming-a-wild-elasticsearch-cluster]

Important parts of configuration. The Elasticsearch config file is called elasticsearch.yml. On Ubuntu Server LTS it is located at

```
/etc/elasticsearch/elasticsearch.yml
```

Cluster Name: Elasticsearch clusters can be named, this is how Elasticsearch knows which cluster a node should be associated with.

Note

Elasticsearch had an interesting feature prior to Elasticsearch 2.0 where it autodiscovers other Elasticsearch nodes on your network and joins them to your cluster if they have the same cluster name. If you don't set a cluster name in your config then it will assume the default value. You don't need to worry about this in Elasticsearch 2.0 as you need to install a plugin to enable the autodiscovery of nodes. If you are working with old software that makes use of an Elasticsearch version prior to 2.0, then it would be helpful to keep this feature in mind. You can read more about the removal of auto discovery of nodes here Multicast Removed Elasticsearch 2.0 [https://www.elastic.co/blog/elasticsearch-unplugged]

Make sure to set a cluster name for all the nodes that you want to be joined to a given cluster. Set the name in elasticsearch.yml. By default cluster.name is commented out.

```
cluster.name: es11
```

Node Names: Nodes have autogenerated names if they aren't set in your config. Ideally you should come up with a naming convention for your nodes and name each node accordingly. The author of the "Taming A Wild Elasticsearch Cluster" suggests that each node is named based on its physical location. The example he uses:

```
# ELASTICSEARCH.YML
node.name: "data-node-1-london"</pre>
```

Taming a Wild Elasticsearch Cluster [http://www.wilfred.me.uk/blog/2015/01/31/taming-a-wild-elasticsearch-cluster/]

Heap Size: This is one of the single most important settings for performance when it comes to Elasticsearch. Allocating Memory for Elasticsearch [http://www.elasticsearch.org/guide/en/elasticsearch/guide/current/heap-sizing.html#_give_half_your_memory_to_lucene] Don't pay attention to this if you have not worked with Elasticsearch much, it might just confuse you. If you have worked with Elasticsearch and you have had to try and tune Elasticsearch for performance, than just remember that it is a good idea to give Elasticsearch up to half of RAM available on your machine.

This setting should be set in your Elasticsearch startup file. Mine can be set by changing the value in this manner:

```
$ sudo vim /etc/init.d/elasticsearch
```

You just need to set the value:

```
export ES_HEAP_SIZE=8G</pre>
```

If you are using upstart to start Elasticsearch on Linux the file you need to edit will be:

```
$ sudo vim /etc/init/elasticsearch
```

Filebeat. In a realistic scenario you would also have to configure shippers for transporting your log files from machines where log data is being collected to the server which is running the ELK stack.

Data can be collected from hardware devices such as network switches, Web Application firewall devices or from servers running different operating systems such as Windows Server, Linux or some other unix variants.

Filebeat is a replacement for the tool "Logstash-forwarder". Both Filebeat and Logstash-forwarder are written in Golang or what is known as the Go programming language.

Filebeat

Filebeat can be found on github here: Filebeat github page [https://github.com/elastic/filebeat]

Filebeat is one of the component of a set of tools refered to as "Beats". These are a set of tools made from the libbeats framework, these tools are meant to ship operational data from all kinds of devices servers and operating systems.

"Beats" are developed by the "Beats" team, which is led by Tudor and Monica who are from Packetbeat which was recently accquired by the company behind Elasticsearch. The idea behind "Beats" was to create a framework that could be used to ship operational data that could either be used to ship data directly to Elasticsearch or to Logstash.

"The Beats are lightweight processes, written in Go, that you install on your servers to capture all sorts of operational data like logs, operating system metrics or network packet data, and to send it to Elasticsearch, either directly or via Logstash, so it can be visualized with Kibana." From Beats github page [https://github.com/elastic/beats]

The idea behind "Beats" was to create a framework that could be used to create lightweight shippers of operational data that could be used to ship data to Logstash or in some cases directly to Elasticsearch.

The "Beats" make use of Libbeat which is a Golang library which was created for developers who would like to create programs that consume operational data and ships it to Elasticsearch or Logstash.

If you would like to create your own "Beats" program, make sure to have Golang installed and the libbeat library. The Elasticsearch official website has an excellent guide on creating New Beats.

Developer Guide: Creating A New Beat [https://www.elastic.co/guide/en/beats/libbeat/current/new-beat.html]

Note on below content: Info mostly from Filebeat: Getting Started Guide on the Elasticsearch Official website.

You could install Filebeat from source and by compiling it from one of the golang compilers or you could just install it from the .deb or .rpm file for your given Linux distro.

Filebeat configuration. Filebeat uses the file filebeat.yml to manage its configuration.

There are several tutorials online that you can use to get started with Filebeat, I found The Logstash Book to be very good you can learn more about it here: The Logstash Book [http://www.logstashbook.com/] "The Logstash Book:Filebeat"

Filebeat has some interesting features:

1. Tail Logs
2. Manage Log rotation
3. Send log data to Logstash
4. Send log data to Elasticsearch

If you are not new to the ELK stack, then you might be familiar with logstash-forwarder and not with filebeat. The Elasticsearch official site has a very good guide to help you migrate from logstash-forwarder to filebeat. Have a look at Migrating from Logstash Forwarder to Filebeat [https://www.elastic.co/guide/en/beats/filebeat/current/_migrating_from_logstash_forwarder_to_filebeat.html]

2.1. Logstash

Logstash was specifically mode with consuming Log and event data in mind. In most cases Logstash has a very simple architecture, which consists of:

1. One or more input.
2. A filter.
3. Output. Logstash has several plugins for output. It can output to Elasticsearch,file, to certain databases, to csv and even to jira and hipchat.

Logstash has two required elements input and output, the filter element is optional. Logstash can either read configuration from a file or it can be defined on the commandline.

Running Logstash with a config file

This is how logstash can be run from the commandline by supplying it a configuration file.

```
$ cd logstash/location/depends/where/it/is/installed
$ bin/logstash -f some_config.conf --configtest
```

Test your config before actually running it. Now run logstash using the config and without testing it.

```
$  /opt/logstash/bin/logstash -f some_config.conf
```

Running Logstash from the commandline

```
$ cd logstash/location/depends/where/it/is/installed

$ bin/logstash -e 'input { stdin { } } output { stdout {} }'
```

What ever is typed in the commandline will be echoed by Logstash. Press ctrl+z to stop Logstash and get back to your terminal.

Logstash sincedb

When Logstash parses data from a file then it can "remember" where it was busy parsing a file the last time Logstash was shutdown or crashed. This feature is useful for if something goes wrong and Logstash crashes or if you want to stop Logstash and restart it and you would like to continue parsing a file from where you left off previously. Perhaps you parsed a directory containing several log files and you would like to make a change to the config that you were using to parse the files. Using this feature of the file plugin of Logstash you can stop Logstash, make a change to the config, start Logstash and run Logstash on all those log files from scratch as if you had not parsed them before.

To use the Logstash file plugin effectively, you need to be able to know how the sincedb mechanism works. The sincdb file is usually located in $HOME/.sincedb. The since database is written to the sincedb file in a specific format. If you need to parse a file that has already been proccessed by Logstash, then you need to rename the sincedb file or you need to remove it. Then you can stop logstash restart it and process the file that you want to parse again.

3. Locking Down Logstash and Elasticsearch

Getting the ELK stack setup and working can be very straight forward, but getting it working while still being secure can be challenging. Security is a tradeoff of usability.

> **Quote** "Elasticsearch does not perform authentication or authorization, leaving that as an exercise for the developer."
>
> Use: Elasticsearch official blog

Due to the fast pace of development life cycles in the time we live in,leaving the responsibility of security to the developer makes as much sense as trusting a teen son with the car keys of your sports car.

Elasticsearch Blog:Elasticsearch Security [https://www.elastic.co/blog/found-elasticsearch-security]

Both Elasticsearch and Logstash have a great deal of features and as a system administrator you want to make sure that you lock down those features only to be used by users of your system. One of Elasticsearch’s obvious features is the ability to query the Elasticsearch REST interface for data. You want to make sure that you lock down Elasticsearch otherwise this could lead to your data being accessed by malicious hackers or even have hackers take over your Elasticsearch cluster and use it to send out spam. (For example if you have the Elasticsearch watcher plugin installed, but allow public access to your Elasticsearch cluster.)

To give you an idea of what malicious hackers will be looking for you should take a look at the shodan search engine. With Shodan you can find an open Elasticsearch cluster with very little effort. The best thing you can do as to take up the attacker mindset, in order for you to know how attackers find elasticsearch instances on the public internet. This will help you from falling victim to hack, by knowing how a hack will by carried out and also by taking time to research how to lock down an Elasticsearch cluster. Have a look at the results of:Searching for Elasticsearch Clusters on Shodan [https://www.shodan.io/search?query=elasticsearch]

Elasticsearh is can be used as a data store. Just as a database can be compromised and used to run system command on its host, compromising elasticsearch could also give an attacker the ability to run system commands on the host. Relational databases such as postgresql and mysql have the concept of user access: where databases,tables and database functions can be restricted on a per user bases. Elasticsearch has no such concept at the time of writing.

From the elasticsearch website: Elasticsearch Website [https://www.elastic.co/blog/found-elasticsearch-security]

> **Quote** "Elasticsearch has no concept of a user. Essentially, anyone that can send arbitrary requests to your cluster is a 'super user'"
>
> Use: https://www.elastic.co/blog/found-elasticsearch-security

Thus it is important to restrict access so that not just anybody can send arbitrary requests to your cluster. Elasticsearch has an API, and just the way that most modern API's restrict access to the API by using some means of authentication such as an API key, you should also restrict access to Elasticsearch somehow, there are various ways to do this.

One of the most common ways to restrict access is to use Nginx as a reverse proxy and add http authentication. Thus a user needs a username and password in order to query eleasticsearch.

> **Quote** "If money could be said to exist anywhere in a network,it exists on a database server."
>
> Use: The Database Hacker's Handbook: Defending Database Servers by David Litchfield et al. John Wiley

The same could soon be said for Elasticsearch. You could very possibly find medical records,employment records and even transaction records on an elasticsearch cluster somewhere in the world today. The world today is very reliant on databases, but also on the ability to query large amounts of data and get results from that query instantly.

The Elasticsearch blog gives some tips on securing an Elasticsearch cluster. Here is a brief overview of those tips:

1. Disable users ability to run arbitrary code. The dynamic scripting feature in Elasticsearch allow users to run arbitrary code. Scripts do not run inside a sandbox. Users who can run scripts on your system have the same abilities as someone with a shell on your system. You should take the same precautions you would take in terms of hardening the host that you would take when handing over shell access to a user when you consider giving a user scripting abilities on an Elasticsearch host. Consider hardening the kernel and installing regular updates.

2. User Access Control is extremely important. You cannot give all users on your system access to all data and functionality of your system. Imagine you had an accounting Web Application that contained data of several different clients, you would need to restrict user's from seeing each other's data, to prevent clients from seeing each other's data. The same concept applies to running Elasticsearch and logging client data from multiple clients to your Elasticsearch cluster.

3. Disable requests that can trigger a denial of service. This can be any request that affects the availibility of your cluster. For example:

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

> "Go with an absolute least privilege approach. Do not expose any operations that are not needed. Do not expose any data that is not required. "
>
> — Greg Patton

This request shuts down all the nodes in your cluster. You certainly want to disable this by disabling the Node Shutdown API: you can do this by changing the line in your config:

```
action.disable_shutdown
```

3.1. Security Checklist

It is important to treat your Elasticsearch clusters different that are used in production compared to those used in the testing phase of your environment or platform.

Before deploying Elasticsearch clusters in production, make sure to have a security checklist that can be used to make sure you have done everything possible to secure and to test the security of your cluster’s configuration.

Here are some articles that can help you come up with a security checklist:

Elasticsearch Security Checklist [https://www.elastic.co/guide/en/found/current/security-checklist.html]

Elasticsearch Preflight Checklist [https://www.elastic.co/guide/en/found/current/preflight-checklist.html]

Elasticsearch in Production Security [https://www.elastic.co/blog/found-elasticsearch-in-production?q=elasticsearch%20in#security]

As part of your security policy you could consider deploying honeypot Elasticsearch clusters to observe the latest trends in terms of attacking Elasticsearch clusters. There is a Elasticsearch honeypot project which is opensource and can be found on github:Elastichoney Project Github page [https://github.com/jordan-wright/elastichoney]. This Elasticsearch honeypot was used to gather statistics on real life attacks that were making use of CVE-2015-1427. You can read more here Elasticsearch Attacks in the Wild Analysis Using ElasticHoney Honeypot [https://threatpost.com/elasticsearch-honeypot-snares-8000-attacks-against-rce-vulnerability/112715/]

Security CheckList Example:

Whether you are using the ELK stack directly in your operations environment or if you are using Elasticsearch in a web application that you are writing, you should think of considering Elasticsearch the same way that you would think about securing an API.

I use the following: "High Level overview of doing an API assessment:" which is based on a presentation by the security researcher Greg patton as my checklist.

> "Analyzing any given API is likely to yield significant vulnerabilities"
>
> — Greg Patton

Example 1. Elasticsearch API:

NOTE Important

Security: These vulnerabilities might not even be in the API itself, it could be just in the configuration of the API or how authentication is implemented in the given API.

Based on presentation by Greg Patton: "API Assessment Primer - Video" [https://www.youtube.com/watch?v=oPrrFNEasgE]

"API Assessment Primer Slides" [https://2015.appsec.eu/wp-content/uploads/2015/09/owasp-appseceu2015-haddix.pdf]

This can be used for your checklist for securing public facing Elasticsearch clusters:

1. Get all documentation: For API and for administration of API. The documentation will tell you what the correct way is to install and lockdown your cluster.

2. Get runtime traffic:See what requests and responses look like. Are they sent in clear text?

3. MAP API:Map out the API, map out documented and undocumented functionality of the given API.

4. Manual Testing: Review the official documentation and review your configurations to ensure you are doing things in the most secure manner possible.

5. Automated Tools.

6. Various automated tools can be used to test the security of your configuration externally, but also to check if you

7. have the latest patches installed. Both Nessus and Openvas can be very helpful for doing your automated testing.

8. These tools also check the SSL configuration of your hosts. SSLscan,ssldigger can be used to verify your ssl configuration, if your cluster is public facing there are various online tools you can use too.

> "APIs often lack sufficient protection of confidentiality and integrity of data in transit"
> — Greg Patton

4. What can the ELK stack be used for?

What the ELK stack can be used for? In an operations setting it can be very boring to look at logs. Logs are just data with timestamps attached to them. In the time we live in we are consuming more data than ever, but also producing more data than ever. For some log files only seem to be useful when a problem occurs in a development or operations setting.

The current time we live there is an increasing trend to move towards making decisinos based on data. In the blog post:

> "Storing, processing and querying logs effectively is helping businesses succeed."
> — http://jamesthom.as/blog/2015/07/08/making-logs-awesome-with-elasticsearch-and-docker/

A business needs to know why their current strategy is either working to make their business profitable, but also the opposite. If a business can pinpoint the reasons for their success, then it can keep working on improving aspects that could make the business adapt to current trends and upcoming trends and either become more profitable or become more sustainable.

The search engine duckduckgo was operating for quite a few years without grabbing much attention, until out of nowhere it suddenly became extremely popular. They couldn't pinpoint anything that they had done different that could be causing the sudden spike in popularity. After much review the search engine's staff realised that the increase in popularity was due to the PRISM revelations of the system administrator Eric Snowden. It is very important for a business to determine whether a sudden increase in business is caused by external factors or not and whether a spike in profit will be sustainable.

Collecting information is not the only important technical advantage that a business or government can have, but storing ,querying and analyzing information. Having exabytes of data available is completely worthless if you don't have the tools to gather information from your information. During the dot com era the perl programming language(Practical Extraction and Reporting Language) became very popular because it allowed businesses to parse files of different formats and extract useful information from it into useful forms such as reports.

Being able to represent data in different forms and explain how a report was produced is very important as reports and metrics are often put under scrutiny for accuracy in terms of how data was collected and how it is represented in the current business world. Kibana solves this problem.

What can the ELK stack be used for? Lets have a look at how and what it is being used for at some big organizations and companies:

1. NOC (Network Operations Center)

2. Log Analysis

3. Application Logging

4. Search Engine

This book is more focused on using the ELK stack in an operations context so lets look at a high level at some of the operations related use cases for the ELK stack.

4.1. Network Operations Center

A network operations center is responsible for monitoring and maintaining uptime of critical
 infrastructure it can either be servers,network devices or anything in the data center. Monitoring what it is like to use the network from a user's point of view is also important. Sometimes when one of the user's on your network is doing something illegal or that they are not allowed to is also important.

Important. Always make sure when you do network monitoring of any network even your own network, that you are acting within the laws of your country, but also make sure that you respect the privacy of your users.</p>

The ELK stack is a very good pick for a NOC as it logstash can parse so many different types of data. The ELK stack can be used for netflow analytics,with the netflow codec for Logstash.

The University of California Los Angeles makes use of the ELK stack in an operations context to view security related events in their Log files. They have documented how it is used in their environment extensively. You can have a look at the following resources:

UCLA IT Security [https://www.itsecurity.ucla.edu/elk]

UCLA IT Security ELK Stack on Github [https://github.com/ucla-it-security/iso-elk-stack]

This implementation is also meant to be used in a network forensics context.

4.2. Network Forensics

There aren't many open source solutions for working with PCAP files. Analyzing PCAP files can be very challenging especially when working with large PCAP files. Molock, an opensource project by AOL ,was created to make a tool that would enable users to almost instantly process and run text searches on packet captures of IPV4 traffic. The tool makes use of Elasticsearch and Kibana, and also has its own custom front end. The project has very mature documentation and even comes with sample PCAP files.

Molock is extremely fast, because it makes use of Elasticsearch. The project documentation even shows how to Scale your Elasticsearch install with very good explanations of the configuration files. It will certainly be worth your time to read over the documentation if you would like to find out more about scaling Elasticsearch.

The existing paid solutions for indexing PCAP files are extremely expensive and this project is the first in its kind.

Network Forensics at AOL powered by Elasticsearch [https://www.elastic.co/videos/moloch-elasticsearch-powering-network-forensics-aol]

PCAP Capture and Import [https://github.com/aol/moloch]

Finding a Needle in a Hay Stack with the ELK Stack [https://digital-forensics.sans.org/summit-archives/dfirprague14/Finding_the_Needle_in_the_Haystack_with_FLK_Christophe_Vandeplas.pdf]

There are also many other projects that use the ELK stack to create something useful for network forensics. The project "SELKS" SELKS Github page [https://github.com/StamusNetworks/SELKS] is a project that combines the ELK stack with Suricata. The latest version uses Kibana 3, but is very useful if you are trying to find malware traffic on your network by looking at DNS queries for example. This project is very interesting, there is an ISO that you can download from the github page which you can install on a device or in a virtual machine. Their Kibana interface alrady has quite a few very interesting dashboards to give info on DNS requests made and also has an alert dashboard which gives info on potentially malicious activity.

4.3. Application Logging

Application logging can help you see what is going in your application while you are busy testing it. Finding useful information from custom application debug data is usually a tedious process. Application logging will not only help you find issues in your application, but also help you find ways to improve your code.

The ELK stack can be used to replace traditional application logging tools used by developers and system administrators. There are many examples online of using Monolog with the ELK stack to debug PHP applications. See the following resources for more information.

Application Logging with ELK Stack [http://www.slideshare.net/benwaine/application-logging-with-the-elk-stack]

More on Application Logging

If you would like to learn more about using the ELK stack for appliction logging I would highly recommend that you read the book "Using Docker: Developing and Deploying Software with Containers by Adrian Mouat" Using Docker Developing and Deploying Software with Containers [http://shop.oreilly.com/product/0636920035671.do]

The book is very interesting and has an example of using the the ELK stack inside docker containers to do application logging of an application running in one of the containers and also to monitor the docker container logs.

5. Use Cases:With Step by Step Examples

Working with the ELK can be daunting at first, but practice makes perfect. This chapter looks at some very real life hands on examples that the ELK stack can be used for in an operations context.

Going over some basic example tasks with the ELK stack is a good way to get familiar with it and to understand how all three components work together.

Elasticsearch has a repo on github full of examples of using the ELK stack. There are quite a few examples that are related to using the ELK stack in a operations context. These examples are handy because the include test data in the form of sample log files that can be used to practice the examples. It is worth it to have a look at the examples repo, there is even an example of integrating Elasticsearch into the search funtionality of a PHP web application.

Apache Logs: Plotting hits by IP on a Map Using GeoIP. The first example is based on the example on the official examples repo of Elasticsearch on github:

Elasticsearch Official Examples [https://github.com/elastic/examples/tree/master/ELK_apache]] My examples are based on the examples from the Elasticsearch official examples repository, but were modified to work with the latest version of Elasticsearch,Logstash and Kibana at the time of writing.

The ELK stack makes it very easy to map physical location to an IP found in a log file. This can be handy for several reasons such as: to see where most of your website traffic is coming from, to see where malicious traffic is coming from etc.

Logstash can be used to map IP addresses to physical addresses using a GeoIP database, then in return elasticsearch can create geo_point fields and geohash strings which can be used by Kibana to ploit IP’s on an actual map visulization.

This is very useful in the context of web server logs to see the diversity in traffic coming from different regions of the world. This is a good way to find out more about the readers of your blog or visitors of your website for example.

For this example you will need to have Kibana and Elasticsearch already running.

You can start elasticsearch on Ubuntu Server with:

```
$ sudo service elasticsearch start
```

You can verify that Elasticsearch is running with:

```
$ ps aux | grep elasticsearch
```

Now you need to start Kibana and make sure it is running:

My Kibana is installed in a directory named elk in my home directory, but yours could be installed somewhere else. You do not need root privileges to start Kibana. Elasticsearch and Kibana should be able to run together without having to run Logstash, remember Logstash is used to parse your data and put it in a format that Elasticsearch can digest.

```
$ cd /home/timo/elk/kibana-4.*

$ cd bin/
```

Now lets start Kibana and run it in the background. I usually have screen or tmux open and then I start Kibana in one of the windows while I have htop open in another windows to watch performance. To start Kibana in the background:

```
$ ./kibana&
```

If everything went as planned you should see the following output. You can verify that Kibana is running by looking at the output that Kibana generates.

```
...
log   [11:29:36.272] [info][listening] Server running at http://0.0.0.0:5601
log   [11:29:36.339] [info][status][plugin:elasticsearch] Status changed from yellow to
...
```

When you list the indexes in Elasticsearch, you should now see a index called ".kibana":

```
$ curl 'http://localhost:9200/_cat/indices?pretty=true'
yellow open .kibana 1 1 1 0 2.8kb 2.8kb
```

Kibana and Plugins

From the latest version of Kibana , which at the time of writing is Kibana 4.2, you can now install Kibana plugins which are plugins specifically written for Kibana. Installation is very similiar to how Elasticsearch plugins are installed, but don’t confuse the two.

There will be many interesting Kibana plugins in the near future. Have a look at Timelion the Kibana Time Series Composer [https://www.elastic.co/blog/timelion-timeline]

We will be using logstash in this example, but we are going to run it directly on the config file being used. You don’t need to start logstash as a service. When you run logstash as a service it reads all the config files automatically in the directory /etc/logstash/conf.d/. It is better to first debug your logstash config files, by running logstash manually on each config file one, by one.

Some Tips on Working With Logstash

See if you can get a hold of "The Logstash Book". At the time of writing you could download "The Logstash Book" legally in pdf form for free.

Debug your grok filters.

Split your configurations accross config file. One config file should only be for one kind of data.

Use the config test flag.

Create a directory to keep the files we will be working with for this example:

```
$ mkdir Apache_ELK_Example


$ cd Apache_ELK_Example
```

Now you need to download the following files. If you are running the ELK stack in a vm or remote server then download all the files to your machine which is running the ELK stack except the apache_kibana.json file, this file is for your dashboards. THe apach_kibana.json file you can download to your desktop computer, because you will need to upload it to Kibana via your browser.

Download the following files on your server which is running the ELK stack:

This file is your configuration which is used by logstash, this file tells logstash where to get input, what to do with input and where to write the output. You can place this file anywhere , by convention if you are running logstash as a service and you are constantly receiving log files , then it is best to put the configuration file for those log files in the directory

```
/etc/logstash/conf.d/
```

…

```
$ wget https://raw.githubusercontent.com/timogoosen/examples/master/ELK_apache/apache_lo
```

This isn’t documented in the official examples, but you also need to download a file to your /etc/logstash/conf.d/ directory in order to be able to map geo location. The file is called the "GeoLite City database". A new version of this file is released every month so make sure to always have the latest version installed.

```
$ cd /tmp/
```

Download the file:

```
$ wget "http://geolite.maxmind.com/download/geoip/database/GeoLiteCity.dat.gz"


$ sudo cp /tmp/GeoLiteCity.dat.gz /etc/logstash/

$ sudo gunzip /etc/logstash/GeoLiteCity.dat.gz
```

You should now have the file GeoLiteCity.dat in your /etc/logstash directory:

```
$ sudo ls /etc/logstash/

conf.d  GeoLiteCity.dat
```

The next file that you will need is your elasticsearch template. As this very good article states:

> **Quote** "Templates define settings and mappings that will be used when a new index is created. They are especially useful if you create daily indexes (e.g. from logstash) or you have dynamic field names."
>
> Use: http://svops.com/blog/elasticsearch-mappings-and-templates/

Elasticsearch Mappings and Templates [http://svops.com/blog/elasticsearch-mappings-and-templates/]

```
$ wget https://raw.githubusercontent.com/timogoosen/examples/master/ELK_apache/apache_te
```

You need to download your sample log file. This is the file that will be parsed by logstash.

```
$ wget https://raw.githubusercontent.com/timogoosen/examples/master/ELK_apache/apache_lo
```

On your desktop machine download the file that will be used for creating your Kibana visualizations that make up the dashboard.

```
$  wget https://raw.githubusercontent.com/timogoosen/examples/master/ELK_apache/apache_k
```

Find out where logstash is installed:

```
$ whereis logstash
logstash: /etc/logstash /opt/logstash/bin/logstash.bat /opt/logstash/bin/logstash
```

On my machine logstash is installed at:

```
/opt/logstash/bin/logstash
```

First test your config file:

```
$ cat apache_logs | /opt/logstash/bin/logstash -f ./apache_logstash.conf --configtest
```

Now import your data into logstash:

```
$ cat apache_logs | /opt/logstash/bin/logstash -f ./apache_logstash.conf
```

Now list the indexes on your Elasticsearch cluster:

```
$ curl 'http://localhost:9200/_cat/indices?pretty=true'
yellow open .kibana            1 1     1 0  2.9kb  2.9kb
yellow open apache_elk_example 5 1 10161 0 11.7mb 11.7mb
```

Open up Kibana in your browser so you can configure an index pattern:

```
http://your_ip_here:5601/

If you are running everything on your local machine.
http://127.0.0.1:5601/
```

Now go to Settings - Indices- Configure an Index Pattern.

Index name or pattern: apache_elk_example

Time-field name: @timestamp.

Click on create.

Our config accepts input from stdin which means that if we concatenate the contents of the apache_logs file and pipe it to logstash, we are effectively sending stdin to logstash. This is where we specified input for logstash:

```
input {
 stdin { }
}
```

We could have also loaded the file using the file keyword. Logstash accepts various forms of input.

I had trouble for some reason with this example and had to slightly modify my config file to be able to get this example working on the latest version of Ubuntu Server LTS at the time of writing. I changed the codec from dots to plain and I specified the charset, this even if the file contained funny characters,we would still be able to parse it using logstash. The original config was slightly different: Take note of the output section in the original config.

```
...
output {
stdout {
codec => dots
}
```

The input section of this config is important as it shows how flexible logstash is at taking input. Logstash is very flexible and can take data from streams, files or even stdin.

Nginx Logs: GeoIP Example: If you followed the previous example, you would have Elasticsearch and Kibana running. This example takes access logs from the popular web server Nginx and visualizes it in Kibana. This makes use of the default Nginx logging format.

Create a directory for the example files:

```
$ mkdir nginx_ELK_Example</pre>


$ cd nginx_ELK_Example
```

Download the files needed for the example to your machine running the ELK stack. The file that will be used by Kibana to create the dashboard can be downloaded and saved to your local machine, because you will need to upload it using your browser, to Kibana.

```
$ wget https://raw.githubusercontent.com/timogoosen/examples/master/ELK_nginx/nginx_logs

$ wget https://raw.githubusercontent.com/timogoosen/examples/master/ELK_nginx/nginx_temp
```

This file is the file containing the test data:

```
$ wget https://raw.githubusercontent.com/timogoosen/examples/master/ELK_nginx/nginx_logs
```

Download this file to your local machine, this will be uploaded to Kibana using your browser:

```
$ wget https://raw.githubusercontent.com/timogoosen/examples/master/ELK_nginx/nginx_kiba
```

Go ahead and run the example:

```
$ cat nginx_logs | /opt/logstash/bin/logstash -f ./nginx_logstash.conf
```

Have a look at the indices you have created by running the two examples, and take note of the size of each indice.

```
$ curl 'localhost:9200/_cat/indices?pretty'
yellow open apache_elk_example  5 1 40644 0 28.7mb 28.7mb
yellow open nginx_elk_example 5 1 110159 0 68.9mb 68.9mb
yellow open .kibana 1 1 30 6 82.5kb 82.5kb
```

Now navigate to your Kibana it will be on

Kibana Running on Localhost [http://127.0.0.1:5601] If you are running it on your local computer.

Go ahead to Settings, click on the Indices tab, go to "Configure an index pattern". Make the sure you select "Index contains time-based events". Index name or pattern needs to be:"nginx_elk_example". At time-field name select @timestamp and click on "create".

Click on Discover. Just below the search bar is a dropdown to select the current index to search on.

Index name or pattern: nginx_elk_example Time-field name: @timestamp

Have a look at some of the data. If everything went well , then all the fields will be filled with values and you won’t see any fields showing "_grokparsefailure". If you can’t see any data yet, make sure you have the correct index selected in discover and also adjust the time frame that is displayed in the "Discovery" page.

Now you need to import the dashboard that will be used to visualize your Nginx logs. A Kibana dashboard is made up of several visualizations. Dashboards can be exported and imported in json format, which is a great feature of Kibana as it allows people to share there Dashboards with each other.

Click on "Settings" -"Objects" and click on "Import". Select the file containing your dashboard that you saved to your local machine.

Now Click on "Dashboard". Next to the search bar click on "Load Saved Dashboard". Then click on "Sample Dashboard for Nginx Logs".

You should now see dashboard showing a map reflecting regions where the IP’s should be physically located that were in the sample Nginx log.

Going Forward. Make sure to do many examples of parsing different kinds of log files in order to get familiar with the ELK stack. The Elasticsearch community is very beginner friendly and easy to engage with. Feel free to ask questions on their forum or Freenode IRC channel. I predict that the ELK stack will be widely adopted in the corporate environment over the next few years, by both developers and operations staff. It might feel difficult at first to adopt the ELK stack into your list of favourite tools, but it is certainly worth the time and effort. To be one of the early adopters of a new technology is always a good way to improve your skills, but also a good way to learn how to teach others. All the examples in this short book were on the Linux platform, but the ELK stack can also run on other platforms such as FreeBSD and Windows.

The best way to stay up to date with what is going on in the Elasticsearch world is to regularly have a look at the Elasticsearch official blog and to engage with other users of the ELK stack on the elasticsearch forum.