jamk.fi

Centralized log management

Miikka Ruokola

Bachelor's thesis
May 2017
School of Technology
Degree Programme in Information Technology

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Description

Author(s)	Type of publication	Date
Ruokola, Miikka	Bachelor's thesis	May 2017
		Language: English
	Number of pages: 70	Permission for web publication: X

Title
Centralized log management

Degree programme
Information Technology

Supervisor(s)
Rantonen, Mika; Häkkinen, Antti

Assigned by
Solteq Oyj; Hasanen, Kimmo

Abstract

The project was assigned by Solteq Oyj, a Finnish software company centralized around digital marketing with offices in multiple countries. The goal of the project was to replace an existing proof of concept environment running the Elastic Stack by recreating everything from scratch. The objectives were to achieve the goal by using automation and containerization while learning the use of every component involved in the process.

Because the tools used were determined beforehand both by company policy and from previous internal experience with the stack, there were no comparisons done between these tools and other similar ones.

The implementation of the Elastic Stack was completely carried out with automation by utilizing Ansible and its features. As the result, with little previous experience of using many of the software components involved in the project, much was learned. The outcome of the project was taken into use and it has benefited the company in many ways, including easier resolution to problem situations.

Keywords/tags
log, log management, Elastic Stack, Redis, Ansible, Docker

Miscellaneous

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Ruokola, Miikka	Opinnäytetyö, AMK	Toukokuu 2017
		Julkaisun kieli Englanti
	Sivumäärä 70	Verkkojulkaisulupa myönnetty: X

Työn nimi
Keskitetty lokienhallinta

Tukinto-ohjelma
Tietotekniikka

Työn ohjaaja(t)
Rantonen, Mika; Häkkinen, Antti

Toimeksiantaja(t)
Solteq Oyj; Hasanen, Kimmo

Tiivistelmä

Opinnäytetyön antoi Solteq Oyj, joka on suomalainen digitaaliseen markkinointiin keskittyvä ohjelmistoyritys jolla on toimistoja useissa maissa. Projektin päämääränä oli korvata olemassaoleva Elastic Stackia pyörittävä konseptiympäristö tekemällä kaikki alusta asti uudestaan. Tarkoituksena päämäärään pääsemisessä oli käyttää automaatio- ja kontitustekniikoita samalla oppien kaikkien työhön liittyvien ohjelmistojen käyttöä.

Koska käytetyt työkalut oli päätetty etukäteen sekä yrityksen käytäntöjen että aiemman sisäisen kokemuksen perusteella, ei tässä työssä käsitelty käytettyjen ohjelmistojen vertailuja muihin vastaaviin.

Toteutusosiossa Elastic Stackin käyttöönotto tehtiin täysin automatisoidusti Ansiblea ja sen ominaisuuksia käyttäen. Tuloksena saavutettiin paljon kokemusta ja opittuja asioita työssä käytettyjen ohjelmistojen osalta, kun aiempaa kokemusta näistä oli vain vähän. Työssä tehtyjen töiden tulokset otettiin käyttöön ja ne ovat auttaneet monilla tavoin esimerkiksi ongelmatilanteiden ratkaisemisia.

Avainsanat
loki, lokienhallinta, Elastic Stack, Redis, Ansible, Docker

Muut tiedot

Contents

Figures

Tables

Terms

AIX	Advanced Interactive eXecutive - Properietary Unix operating system developed by IBM
API	Application Programming Interface - A set of definitions for communicating between software components
CGI	Common Gateway Interface - A standardized scripting protocol created for web servers that generates pages dynamically
EPEL	Extra Packages for Enterprise Linux - Repository that provides additional packages for CentOS and some other Linux distributions
HTTP	Hypertext Transfer Protocol - Protocol to transfer text on the World Wide Web with visual presentation
HTTPS	HTTP Secure - HTTP over an encrypted connection
HTTP/2	HTTP version 2 - A newer, somewhat backwards compatible, version of the HTTP protocol with improvements in performance and efficiency
IP address	Internet Protocol address - Computer identification system used in telecommunication networks
JSON	JavaScript Object Notation - Human readable text format for data structure representation
REST	Representational state transfer - RESTful web services provide unified interoperability between computers
SSH	Secure Shell - Telecommunications protocol for secure, encrypted and confidential communication between two computers

SSL/TLS	Secure Sockets Layer / Transport Layer Security - Protocols providing cryptographic security in communications over computer networks
TCP	Transmission Control Protocol - One of the main protocols used in computer networks to transport data between applications

1 Premise

1.1 Assigner

This project was assigned by Solteq Oyj, a company providing its clients with services focusing on digital commerce. Solteq expertises in digital customer engagement. It has the capabilities to provide comprehensive service to its customers by being able to cover technological solutions, continuous services and business support. The company's revenue in 2016 was over 63 million euros and during that year it employed a staff of over 450 employees in average. (Solteq Oyj 2016)

1.2 Assignment and its goals

The goal of this assignment was to replace a proof of concept environment by setting up the Elastic Stack as a tool to gather server logs from different customer environments to a centralized location for easy near-real time data inspection, visualization and reporting. The software stack was to be set up and configured with the automation tool Ansible for both easier and faster deployment of the stack itself and any configuration changes that may occur later in the environment's lifecycle.

Instead of the traditional way of installing the Elastic Stack directly on top of the target server's operating system, the Docker containerization technology was utilized. As the tools used were chosen before the project began, it was left in the scope of the assignment to learn the usage of these tools, mainly Ansible and Docker, and use them together in a way, which would allow for the outcome to be utilized in other projects within the company.

2 Log

Log data is a document of an event that has happened at a certain moment and exists for a pre-defined purpose and amount of time. Logs and log processing are needed in both normal and abnormal situations, in the former for monitoring operations, statistics and performing analysis. In abnormal situations logs are

needed for normalizing the situation and situational investigation such as cause analysis as to why the situation deviated from normal. (Government Information Security Management Board 2009, 13)

3 Software components

3.1 Elastic Stack

The Elastic Stack (previously called ELK Stack) is a software stack that is mainly comprised of three different open source server software components: Elasticsearch, Logstash and Kibana. These three components were created to make a powerful combination of software that allows its users to easily and efficiently conduct centralized log management, inspection, analysis and visualization. (Chhajed 2015, 5)

3.1.1 Elasticsearch

Elasticsearch is the backbone of the Elastic Stack. It is a very scalable open source engine for text search and analysis enabling its users to quickly store, search and analyze data in near-real time and is usually used as the backend technology for applications that have complex features and requirements for searching. (Elasticsearch BV 2017)

An Elasticsearch index is a collection of documents or data that is similar or otherwise related. It is defined by its unique name, which can be used as the identification to determine the target for queries such as searching or updating the data in it. An index is categorized with one or more types, usually containing their own set of data fields with specific type of values. (Elasticsearch BV 2017)

Elasticsearch supports distributed computing, which means it is possible to combine multiple nodes into a cluster that provides redundancy, stores data and provides the capability to index and search through it across all the cluster's nodes. Elasticsearch can form clusters automatically, which it does by communicating with other nodes in or out of a cluster and groups up with the nodes that have the same cluster name configured. Despite these features,

Elasticsearch is able to operate as a lone single node system as well. (Elasticsearch BV 2017)

3.1.2 Logstash

Logstash is an open source engine for data collection and processing. It can read and normalize data from various sources and then send it off to the desired destinations for archival or later processing. Logstash was originally created for log gathering and enrichment; however, thanks to its modular core, it is now capable of much more with its wide selection of input, output, codec and filter plugins. (Elasticsearch BV 2017)

The capabilities of Logstash are visualized in Figure 1.

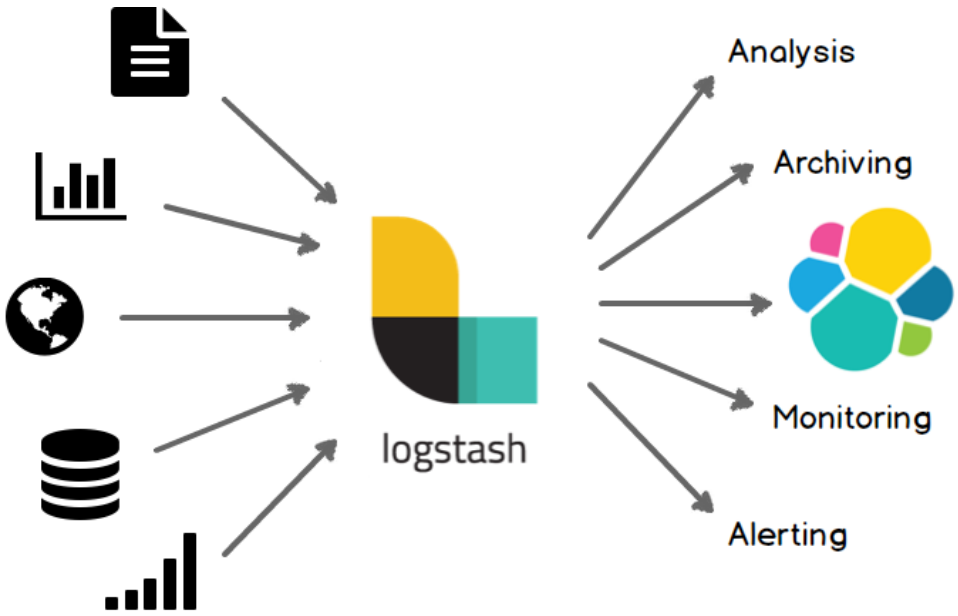

Figure 1: Visualization of Logstash's capabilities. (Elasticsearch BV 2017)

As mentioned in the previous paragraph, Logstash has a vast selection of plugins divided into four categories: input, output, codec and filter plugins. Of these the first two are responsible for reading from and writing to outside data sources, such as Elasticsearch, Redis or even a simple file on the filesystem. Codec plugins are used in conjunction with input and ouput plugins and are responsible for reading data in certain formats, such as JSON. Filter plugins are the types of plugins that Logstash uses to process and transform the data in an event, for example, the Grok filter can parse information from text strings using regular

expression. In total, there are over 150 officially maintained plugins. (Elasticsearch BV 2017)

3.1.3 Kibana

Kibana is an open source visualization and analytics platform for Elasticsearch. It is used with Elasticsearch indices to search, view and interact with their data and can be used to analyze and visualize data as charts, tables and maps in an intuitive browser-based user interface. (Elasticsearch BV 2017)

An example of a Kibana dashboard presenting website statistics can be seen in Figure 2.

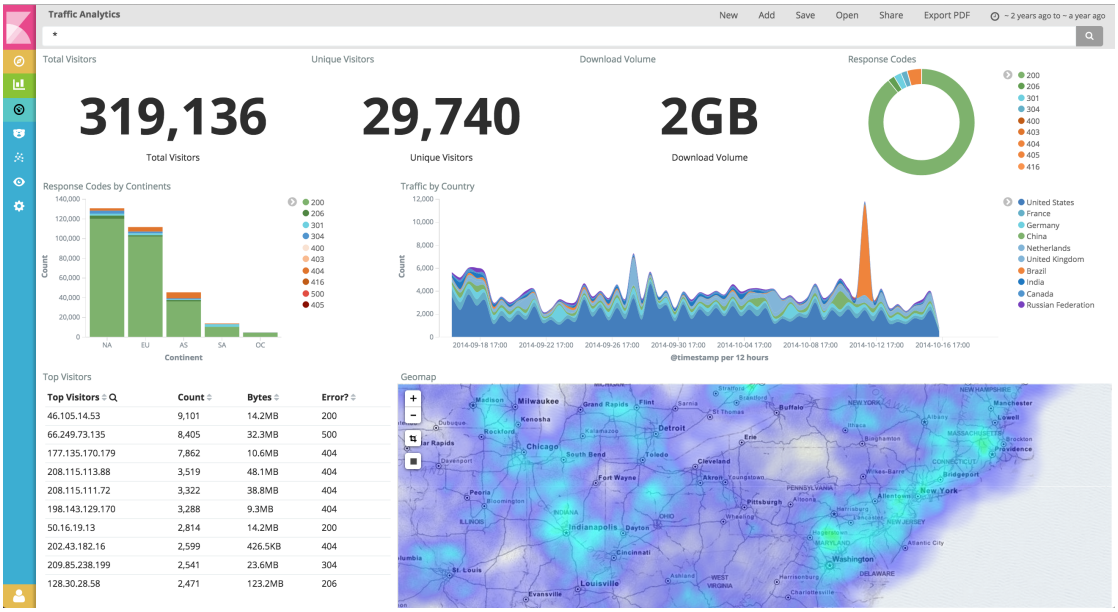

Figure 2: An example Kibana dashboard. (Ewing 2016)

Kibana can also be used to interactively explore the data in Elasticsearch from the Discover page. Document data in every index that matches the selected index pattern can be viewed and executed queries against, for a specified time period. For a search query along with the data a histogram is shown, visualizing the distribution of occurrences the search yielded over time. Search queries can also be saved for later use and visualization purposes. (Elasticsearch BV 2017)

The different sections of the Discover page are explained in Figure 3.

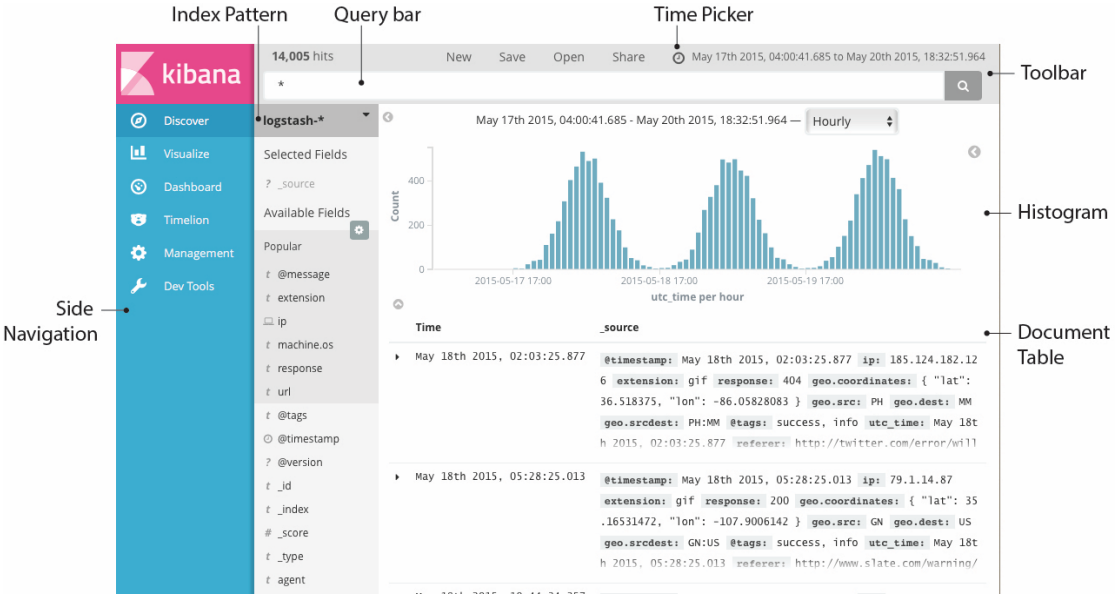

Figure 3: Kibana's Discover page and its sections explained. (Elasticsearch BV 2017)

3.2 Logstash-forwarder-java

Logstash-forwarder-java is a Java port of the original logstash-forwarder which was written in the Go programming language by the creator of Logstash, Jordan Sissel. (didfet 2016)

The purpose of logstash-forwarder-java is to read a given source, either text files or an input stream piped to the program, for new log lines and send those lines off as events to a Logstash server using the Lumberjack protocol. It originated from the need of a lightweight, portable and viable Lumberjack-compatible log shipper for platforms that the Go language does not support, such as IBM AIX. (didfet 2016)

3.3 Redis

Redis is a data store for structured data such as strings, key-value pairs and lists. It has built-in support for transactions, Lua scripting, different levels of database on disk persistence and more. Redis can perform very well and achieves its performance by working with an in-memory dataset, which can be made persistent by either periodically saving it to disk or by writing each command executed to a log. It also supports asynchronous data replication with the

master-slave model, with fast non-blocking synchronization, reconnection and partial resynchronization. (Redis 2017)

3.4 Nginx

Nginx is an open source web server that was originally designed to be a stable and high performance web server, however, since then it has incorporated many more features such as the ability to act as a reverse proxy or a load balancer. As its central goal, it has always been one of the fastest web servers available, and has been able to maintain that status despite web technologies evolving and becoming much more complicated than what they originally were when Nginx was created. (NGINX Inc. 2017)

3.5 Docker

Docker is an open source platform that automates the deployment of software in Docker containers. A container is both a lightweight and standalone package that includes everything required to run a specific software, while isolating it from the underlying infrastructure it is run in. (Docker Inc. 2017)

Containers differ from virtual machines in that they do not contain an entire operating system, but only the programs that run on top of the operating system. This reduces overhead making them lightweight, reducing the usage of system memory, disk and other resources (Docker Inc. 2017). The difference is visualized in Figure 4.

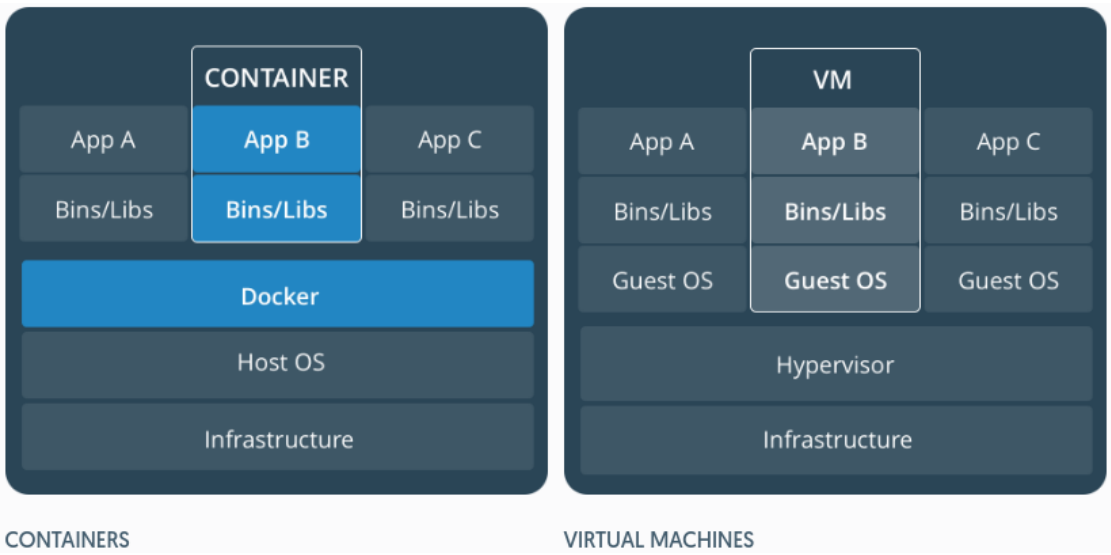

Figure 4: Comparison of containers and virtual machines. (Docker Inc. 2017)

3.6 Ansible

Ansible is an open source automation engine that enables the automation of configuration management, application deployment and much more. It is an agentless system that takes advantage of the very readable and human friendly YAML markup language to abstract functionality, making it very simple and easy to use. By being agentless, Ansible works by connecting to the target machines and uploading small programs consisting of Ansible modules, which are written to target the desired state described to them, and removing the modules when finished. (Red Hat Inc. 2017)

In Ansible, playbooks are used to declare the tasks to be executed. A task is a single operation which will call a module and can reside in either a playbook, a role or a separate task file. A single role can consist of multiple tasks, and they are usually used to group up multiple tasks aiming for the same end goal. Ansible modules are idempotent, meaning that they will try to achieve the desired state and only that. If the same task or module is run multiple times with the same parameters, it will not change anything on the target system when the state it is trying to achieve has already been achieved. (Red Hat Inc. 2017)

4 Log sources

4.1 IBM HTTP Server

IBM HTTP Server is an Apache HTTP Server based web server developed by IBM. Apache HTTP Server is an open source web server project that's goal is to create a stable, rich-featured and openly available HTTP server implementation that is suitable for business use. (Apache Software Foundation 2017)

The biggest differences in IBM HTTP Server compared to Apache HTTP Server are the added support for the WebSphere Administrative Console and suEXEC which allows running CGI scripts as other users. (IBM Inc. 2016)

4.2 IBM WebSphere Application Server

IBM WebSphere Application Server provides flexible and secure Java application server runtime environments ranging from lightweight web, cloud-based applications to large-scale mission critical applications offering near continuous availability. It supports a variety of features including fast installation and deployment, web tier clustering across multiple application server instances, web server load balancing, centralized management and more. (IBM Inc. 2017)

4.3 IBM DB2

IBM DB2 is an SQL-compatible relational database for the enterprise, offering extreme performance, flexibility, scalability and reliability regardless of the size of the organization. It is used for transactional and analytical workloads and provides high availability of data, storage optimizations to transparently compress data and save disk space, storage requirements in general and to improve application performance. (IBM Inc. 2017)

5 Implementation

The implementation steps described in this chapter were done in an environment consisting of five different virtual machines, four running CentOS Linux 7.3 and

one running Ubuntu 16.04 (Xenial). Two of the CentOS servers were used for deploying the Elastic Stack onto, running on top of the Community Edition of the Docker Engine, and the other two for shipping log entries from. The Ubuntu server was purposed as the Ansible control machine.

There was no base configuration done for Redis as it was not deemed necessary due to it working well enough with an out of the box setup.

The final directory structure for the implementation can be found in Appendix 8.

5.1 Environment

The target environments, both test and production, each consist of a single server that were to be running the software components in their own isolated Docker containers. While this may not be ideal considering the possible future workloads, it is practical for learning all the software components used in the assignment.

The architecture chosen was a simple buffered version of the basic Logstash to Elasticsearch pipeline, meaning that instead of log entries being sent directly to the Logstash parser, in front of it there was a dedicated Logstash instance for receiving logs that sent them to a cache server for later on-demand parsing. This architecture and Redis as the caching server were chosen internally beforehand and so these choices were not part of the assignment.

The user frontend was implemented with Nginx acting as an SSL reverse proxy with HTTP basic authentication for both the Kibana frontend and the Elasticsearch HTTP API. The architecture is visualized in Figure 5.

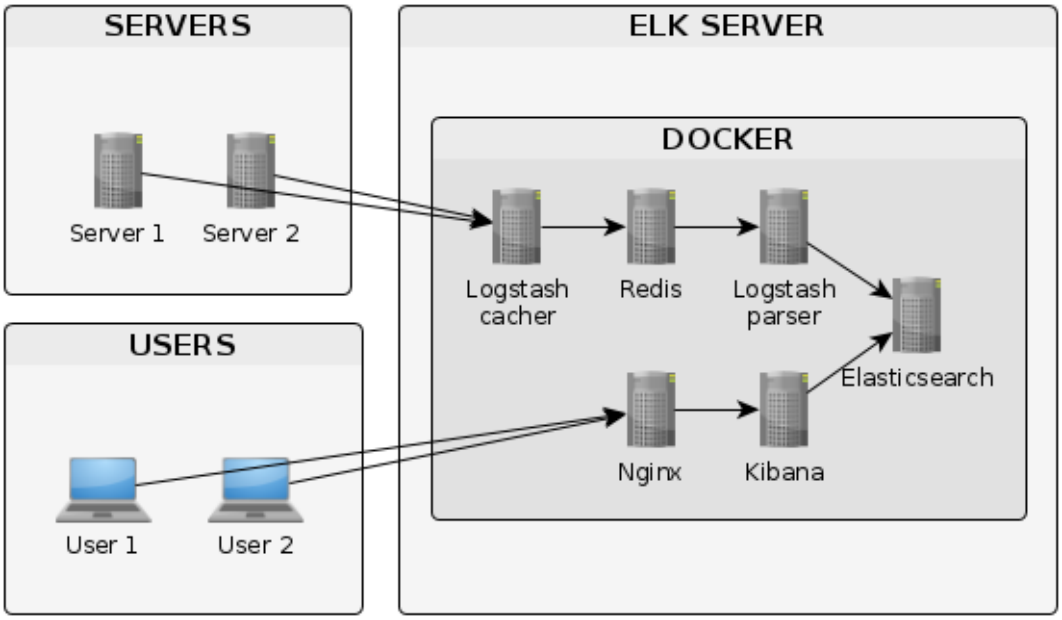

Figure 5: ELK architecture.

The virtual servers used in the assignment and their purposes are listed in Table 1.

Table 1: Virtual servers used.

Hostname	Purpose
elk-prod.example.com	Elastic Stack production server
elk-test.example.com	Elastic Stack test server
server-prod.example.com	Production server where logs are shipped from
server-test.example.com	Test server where logs are shipped from
elk-control.example.com	Ansible control server

5.2 Base configuration

5.2.1 Ansible

As an agentless configuration management software, Ansible does not require much from the hosts it controls. At bare minimum it can use programs over connections like SSH to do work, however for more complicated tasks require a basic Python installation. In this assignment, on top of the basic Python installation some additional third party modules were installed due to the requirements set by some of the Ansible modules used. On the control machine however, some initial setting up is required.

In this assignment an Ubuntu Xenial installation was chosen as the control machine. The easiest way to install Ansible would be to use the operating system's package manager, however, because Xenial is a long term support version of Ubuntu its package repositories do not have the latest version of Ansible as can be seen below:

```
$ apt update
$ apt show ansible | grep ^Version
Version: 2.0.0.2-2ubuntu1
```

From the output it can be determined that the Xenial repositories only have Ansible version 2.0, which is almost a year and a half old at the time of this project. So instead of installing it from the system repositories, Python's `pip` package manager was used instead. Because Ansible was installed this way, there were some system library requirements as well. The installation commands were as follows:

```
$ apt install -y python python-pip libssl-dev libffi-dev sshpass
$ pip install ansible
$ ansible --version
ansible 2.3.0.0
```

Here the OpenSSL and libffi development packages, `libssl-dev` and `libffi-dev` respectively, were also installed, because Ansible depends on the `cryptography` module that in turn depends on the native bindings for these libraries that `cryptography`'s setup script builds when installed with `pip`. Finally looking at the output of `ansible --version` it can be seen that doing the extra steps for the installation was worth it since it is now at version 2.3 instead of the previously would-be 2.0. The `sshpass` program was also installed because in this assignment password authentication was used instead of key-based authentication.

After the installation, there was some project-specific configuration to be performed. While Ansible's default configuration is usually sufficient, there were some parameters that needed to be changed for this use case and these changes were done by creating a new file called `ansible.cfg` in the project's working directory. In addition to ansible.cfg in the current directory where Ansible commands are run it will read configuration from an environment variable called `ANSIBLE_CONFIG`, which it prioritizes over ansible.cfg, and the default location of

the configuration file at `/etc/ansible/ansible.cfg`, which is the last place it will read the configuration from. Creating a custom configuration file for each Ansible project can be useful in the sense that when the project files are shared through version control for example, it will run the same way for everyone, which is why this approach was chosen for this assignment as well.

```
1  [defaults]
2  ask_pass = True
3  host_key_checking = False
4  inventory = ./hosts
5  log_path = ./ansible.log
6  callback_whitelist = timer,profile_task
7
8  [privilege_escalation]
9  become = True
10 become_ask_pass = True
```

In this project the custom configurations were rather simple: Ansible is told to always ask for passwords when starting, disable SSH host key checking and always log all output with timestamps into a logfile also. The name of the default inventory file was also given.

The next step was to create the aforementioned inventory file, called `hosts`. The inventory file holds all the information about Ansible's target hosts and is mandatory when working with remote hosts like this. Each target server is given a friendly name optionally pointing to a specific address and port that is then grouped into environment-specific groups, enabling easy environment or task-specific control:

```
1  [elk_prod]
2  elk_prod_node1  ansible_host=192.168.1.201  ansible_port=22
3
4  [elk_test]
5  elk_test_node1  ansible_host=192.168.1.202  ansible_port=22
6
7  [elk:children]
8  elk_prod
9  elk_test
10
11 [servers_prod]
12 server_prod_node1  ansible_host=192.168.1.203  ansible_port=22
13
14 [servers_test]
15 server_test_node1  ansible_host=192.168.1.204  ansible_port=22
16
17 [servers:children]
18 servers_prod
19 servers_test
```

Normally groups cannot be referred to from inside other groups, however on lines 7-9 a special `:children` keyword is used which tells Ansible that the `elk` group

will actually be referring to the `elk_prod` and `elk_test` groups instead of single hosts. The same was done for the servers where the logs originated from.

The last step in preparing the configuration was to create the variable files for the inventory groups `elk_test` and `elk_prod`. These files were created under the directory `group_vars` which, as the name suggests, is meant for holding the optional variable files for each defined group in files called `groupname` either with or without the YAML file extension. As an example, here is the file for the production environment, `elk_prod.yml`:

```
---

env: prod
http_hostname: elk-{{ env }}.example.com

users:
- user: produser
  pass: pass456
- user: admin
  pass: adminerino456

xpack_settings:
  security: "false"
  monitoring: "false"
  graph: "false"
  watcher: "false"
  reporting: "false"
  ml: "false"

kibana_config:
  users: "{{ users }}"
  server_name: "{{ env }}-kibana1"
  xpack: "{{ xpack_settings }}"

elasticsearch_config:
  users: "{{ users }}"
  cluster_name: "{{ env }}-cluster1"
  node_name: "{{ env }}-node1"
  xpack: "{{ xpack_settings }}"
```

There are quite a few variables defined here, however, between the two environments the most relevant ones are `env` and `users`, where the former holds the environment's name, `prod` in this case, and the latter defines a list of key-value pairs. These two are referenced by the some of the variables under `kibana_config` and `elasticsearch_config` with two curly brackets, so for example `elk-{{ env }}.example.com` becomes `elk-prod.example.com` when Ansible is reading the file at runtime. The translation is possible thanks to the Jinja2 templating engine that Ansible uses.

The variable `xpack_settings` is used to switch off all X-Pack related features, due to X-Pack being a paid add-on to the Elastic Stack.

In Elasticsearch's case, both the `cluster_name` and `node_name` are already defined at this level, which will help future-proof the environment by handily separating the production and test environments from joining into each other's clusters, as described in Chapter 3.1.1.

The commands and files referenced in this chapter can also be found in Appendices 8, 8, 8 and 8.

5.2.2 Elasticsearch

The first actual component part of the Elastic Stack configured was Elasticsearch. While the configuration itself is very simple and straightforward, there are some things to take note of. First, the file was created as follows:

```
$ mkdir -p files/template/elasticsearch
$ vim files/template/elasticsearch/elasticsearch.yml.j2
```

The contents of the configuration file are:

```
1  network.host: 0.0.0.0
2
3  cluster.name: {{ elasticsearch_config.cluster_name }}
4  node.name: {{ elasticsearch_config.node_name }}
5
6  xpack.security.enabled: {{ elasticsearch_config.xpack.security }}
7  xpack.monitoring.enabled: {{ elasticsearch_config.xpack.monitoring }}
8  xpack.ml.enabled: {{ elasticsearch_config.xpack.ml }}
```

In the Elasticsearch configuration file, there are multiple references to variables defined in the group-specific variable files created in the previous chapter. Here the variables are used to help configuration management by putting all the relevant configuration in the same centralized place.

By giving `network.host` the value `0.0.0.0`, Elasticsearch is told to bind itself to every network interface, this is desired with containers, as the actual container layer will handle the networking aspect anyway. Also, the features of the X-Pack add-on are disabled as per the previous chapter.

The commands and the file referenced here are also located in Appendices 8 and 8.

5.2.3 Logstash cacher

The Logstash instance performing the caching of incoming log data was configured before the parser. The Jinja2 configuration template was created:

```
$ mkdir -p files/template/ls-cacher
$ vim files/template/ls-cacher/cacher.conf.j2
```

Logstash configuration files are usually made of three different configuration blocks: input, filter and output. However, as the purpose of this Logstash instance was only to receive and cache the incoming log data, the configuration only has the input and output blocks.

The beginning of the file looks like this:

```
1  input {
2    lumberjack {
3      port => 5000
4      ssl_certificate => "/ssl/{{ http_hostname }}.crt"
5      ssl_key => "/ssl/{{ http_hostname }}.key"
6    }
7  }
```

In a logical manner, the input block is configured first. In the snippet above the input plugin for the Lumberjack protocol is configured to listen the port `5000` and to use the specific SSL key and certificate for authenticating the incoming connections. These are the only mandatory fields for the Lumberjack plugin, so nothing else was configured. By default the address it will bind the port to is `0.0.0.0`.

Ending the cacher configuration file is the output section:

```
9  output {
10   redis {
11     host => "redis:6379"
12     data_type => "list"
13     key => "logstash"
14     codec => "json"
15   }
16 }
```

As the Redis cache server is the only desired destination for the events coming in this Logstash instance, the plugin for it is the only plugin configured in the output block. The `host` setting is set to `redis:6379`, which will be an entry that Docker will automatically put in the container's `/etc/hosts` file on creation.

`data_type` is set to `list`, which is a queue-like data type in Redis, it allows for data to be put at the beginning or in the end of the queue, although the

Logstash plugin will only push events in the end of the queue. The other possible value for `data_type` would be `channel`, however, that was not used because then the relationship of Redis and the parsing Logstash instance would be that Redis would be pushing new events to the Logstash pipeline which in theory could degrade performance as Logstash might start choking due to the amount of the incoming data.

`key` is the list identifier under which all the events will be put for the parser. In this case, the `json` plugin was chosen as the codec because it seemed logical as Logstash events contain much more information than just the log entry. It will create a JSON formatted string of the log event that is then pushed to Redis.

The SSL certificate and key used on lines 4-5 were generated with the following commands:

```
$ cd files/ls-cacher/
$ NAME=elk-prod.example.com
$ openssl req -newkey rsa:4096 -keyout $NAME.key \
    -new -x509 -out $NAME.crt -days 3650 \
    -nodes -subj "/C=FI/CN=$NAME"
```

The files were created with OpenSSL and were generated as a 4096-bit RSA key/certificate pair. The certificate was self-signed and valid for 10 years, until 2027.

Java KeyStore (JKS) files for both environments were also created, as these are needed by the chosen log shipping software, logstash-forwarder-java. The JKS files for production were created with the following commands:

```
$ cd files/
$ NAME=elk-prod.example.com
$ keytool -importcert -trustcacerts \
    -file ls-cacher/$NAME.crt -alias ca \
    -keystore logstash-forwarder-java/$NAME.jks \
    -storepass changeit
```

The same commands were repeated with `NAME`'s value being `elk-test.example.com`.

The commands and the full configuration file can also be found in Appendices 8 and 8.

5.2.4 Logstash parser

In the configuration, the parser utilizes a pattern file to help with the `grok` filter plugin. Both of the files were created with this structure:

```
$ mkdir -p files/ls-parser/patterns files/template/ls-parser
$ vim files/ls-parser/patterns/ibm
$ vim files/template/ls-parser/parser.conf.j2
```

The pattern file called `ibm` is, as mentioned, a helper file for the `grok` plugin. It contained named regular expression (regexp) patterns that make using grok easier:

```
1 WAS_DATETIME %{MONTHNUM}[./]%{MONTHDAY}[./]%{YEAR} %{TIME}:%{INT} \S{,4}
2 DB2DIAG_TIMESTAMP %{YEAR}-%{MONTHNUM}-%{MONTHDAY}-%{HOUR}\.%{MINUTE}\.%{S
  ↪ ECOND}\.%{INT}%{INT:timezone}
```

Two named patterns are defined here: `WAS_DATETIME` and `DB2DIAG_TIMESTAMP`. They both heavily utilize Logstash's built-in named regexp patterns, which are similarly defined in Logstash's internal configurations and will recursively resolve to actual regexp patterns.

The `WAS_DATETIME` pattern will match WebSphere Application Server's log timestamps formatted as
`MONTH.DAY.YEAR HOUR:MINUTE:SECOND:MILLISECOND TZ` with month, day and year being separated by either a dot or a forward slash and where `TZ` is a time zone name, like UTC, for example.

`DB2DIAG_TIMESTAMP` will match DB2's diagnostic log timestamps where the format is `YEAR-MONTH-DAY-HOUR.MINUTE.SECOND.MILLISECONDS+TZ` where all the other fields are the same as with `WAS_DATETIME` with the exception of `TZ` which is actually represented as minutes instead of hours or a time zone's name. The time zone in this pattern is parsed into a separate field called `timezone`, to help with later processing as Logstash's `date` plugin does not support time zones represented in minutes.

Configuration for the Logstash instance doing all the actual data processing and parsing is somewhat more complicated, however, the same principles as in the previous chapter still apply. The configuration consists of all three types of plugins: input, filter and output, out of which the filter block is the most complicated one.

Starting with the input block:

```
1  input {
2    redis {
3      host => "redis"
4      data_type => "list"
5      key => "logstash"
6      codec => "json"
7    }
8  }
```

This is practically the same as the output block in Chapter 5.2.3, but instead of sending data out with the output block, the input block is used instead to read from Redis, with the same settings used in sending the data from the cacher.

```
10 filter {
11   mutate {
12     strip => "message"
13     remove_field => ["offset", "line"]
14     split => { "tags" => "," }
15   }
```

Here the filter block starts, with three actions using the mutate filter plugin used to do mutations on the fields of the Logstash event. On line 12 the `message` field containing the full line of log from the log shipper, is stripped of all leading and trailing whitespaces, removing empty lines and unnecessary spaces before and after the actual log. Line 13 removes the `offset` and `line` fields from the event, which were deemed to be useless information sent by logstash-forwarder-java. Logstash-forwarder-java does not actually support the sending of fields as arrays, so the `tags` field is actually a string that needs to be split into an array here, using a comma as the separator. The field needs to be an array to allow for more fine-grained searching of events in Kibana.

```
17   if ! [message] {
18     drop {}
19   }
```

Continuing with the theme of stripping the `message` field, this event will be dropped entirely if the field is empty.

```
21   grok {
22     patterns_dir => ["/patterns"]
23     match => { "message" => [
24       "^%{COMMONAPACHELOG}",
```

Grok is the filter plugin used to parse arbitrary strings with regexp patterns. The setting `patterns_dir` is used to refer to the directory where the `ibm` pattern file containing the custom named patterns resides. On line 23 the plugin is initialized to match for any of the defined patterns in the `message` field.

On line 24 another Logstash's internal named pattern, `COMMONAPACHELOG` is referenced. This named defined pattern is defined as an Apache HTTP Server-specific pattern and looks for lines formatted as the default CommonLog format for access logs. Here is an example line from a log file that it would be able to parse:

```
::1 - - [23/May/2017:13:43:42 +0300] "GET / HTTP/1.1" 200 3493
```

This line would be broken into fields with values described in Table 2.

Table 2: Example of parsed fields from HTTP server access log.

FIELD	VALUE
clientip	::1
ident	-
auth	-
timestamp	23/May/2017:13:43:42 +0300
verb	GET
request	/
httpversion	1.1
response	200
bytes	3493

```
25          "(?m)\[%{WAS_DATETIME:timestamp}\]%{SPACE}%{BASE16NUM:thread}%{SPAC
             ↪  E}%{WORD:shortname}%{SPACE}%{WORD:loglevel}%{SPACE}%{GREEDYDAT
             ↪  A:message2}",
```

On line 25 is the parser for WebSphere Application Server's (WAS) SystemOut.log file, which is where all the messages going to stdout are written. The pattern starts with `(?m)` which indicates that this can be a multi line event that spans multiple lines, in case of Java exception stack traces and such. The end of the string is parsed into the `message2` field because `grok` does not overwrite existing fields. An example from a log file that the pattern would be able to parse into several fields follows:

```
[5/23/17 13:22:33:157 EEST] 00000001 WsServerImpl  A   WSVR0001I: Server
server1 open for e-business
```

This line would be broken into the fields and values described by Table 3.

Table 3: Example of parsed fields from WebSphere Application Server log.

FIELD	VALUE
timestamp	5/23/17 13:22:33:157 EEST
thread	00000001
shortname	WsServerImpl
loglevel	A
message2	WSVR0001I: Server server1 open for e-business

```
26          "(?m)%{DB2DIAG_TIMESTAMP:timestamp}%{SPACE}%{NOTSPACE:id}%{SPACE}LE
                VEL:%{SPACE}%{DATA:loglevel}%{SPACE}PID%{SPACE}:%{SPACE}%{INT:
             ↪  pid}%{SPACE}TID%{SPACE}:%{SPACE}%{INT:tid}%{SPACE}PROC%{SPACE}
             ↪  :%{SPACE}%{DATA:proc}%{SPACE}INSTANCE%{SPACE}:%{SPACE}%{NOTSPA
             ↪  CE:instance}%{SPACE}NODE%{SPACE}:%{SPACE}%{INT:node}%{SPACE}HO
             ↪  STNAME%{SPACE}:%{SPACE}%{NOTSPACE:hostname}%{SPACE}%{GREEDYDAT
             ↪  A:message2}"
27        ]}
28    }
```

Line 26 contains the most complicated of the three, the pattern for DB2's diagnostic log entries. The log file itself is very elaborate and informative which is of course a good thing, however, when trying to parse those entries it is often impossible to avoid the mental gymnastics involved. Again the pattern starts with the multi line identifier `(?m)` same as the WAS log pattern and it uses the custom named pattern `DB2DIAG_TIMESTAMP` to find the timestamp, and several of Logstash's internally defined named patterns such as `SPACE` which matches zero or more spaces in a row. The `DATA` pattern is a non-greedy regexp matcher that matches everything until the next character specified in the pattern is found. Here as well the end of the string is parsed into the `message2` field because `grok` does not overwrite existing fields. An example block from the DB2 diagnostic log file might look something like the following:

```
2017-05-23-09.53.38.354591+180 I1736E334            LEVEL: Event
PID     : 8261                 TID : 140448931723136 PROC : db2flacc
INSTANCE: db2inst1             NODE : 000
HOSTNAME: localhost.localdomain
FUNCTION: DB2 UDB, config/install, sqlfLogUpdateCfgParam, probe:30
CHANGE  : CFG DBM: "Discover_comm" From: ""  To: "TCPIP"
```

This would then be broken by the pattern into the fields and values described in Table 4.

Table 4: Example of parsed fields from DB2 diagnostic log.

FIELD	VALUE
timestamp	2017-05-23-09.53.38.354591
timezone	+180
id	I1736E334
loglevel	Event
pid	8261
tid	140448931723136
proc	db2flacc
instance	db2inst1
node	000
hostname	localhost.localdomain
message2	FUNCTION: DB2 UDB, config/install, sqlfLogUpdate...

```
30    if [type] == "db2" {
31      ruby {
32        code => "event.set('timestamp', event.get('timestamp')[0..-5] + '
          ↪  +%02d00' % (event.get('timezone').to_i / 60))"
33      }
34      mutate {
35        remove_field => [ "timezone" ]
36      }
37    }
```

Because DB2's log format prints the time zone as minutes instead of hours, this condition block is required. It will check if the event's `type` is `db2`, and if it is the `ruby` filter plugin on lines 31-33 will be run to convert the time zone to hours and concatenate it with the `timestamp` field. When the `ruby` filter is done, the `timezone` field is removed from the event.

```
39    if [message2] {
40      mutate {
41        replace => { "message" => "%{message2}" }
42        remove_field => [ "message2" ]
43      }
44    }
```

Because the `grok` filter will not replace the values of existing fields, the `message2` field that stores main message of the event is removed after having replaced the original `message` field.

```
46    mutate {
47      gsub => ["timestamp", "EE(S|D)T", "+0300"]
48    }
```

Here the mutate plugin is used again, this time it needs to be processed after the grok parsing is done because WAS uses either `EEST` or `EEDT` as its time zone

name when run on a machine using the Europe/Helsinki time zone. The `gsub` option is used to search the field `timestamp` for all matches of either `EEST` or `EEDT` and those matches are then replaced with `+0300`. This is done because the parsing engine Logstash uses to parse datetime strings, Joda-Time, does not support those named time zones, but it does support +0300.

```
50    date {
51      match => ["timestamp",
52        "dd/MMM/yyyy:HH:mm:ss Z",
53        "M/d/yy HH:mm:ss:SSS z",
54        "M.d.yy HH:mm:ss:SSS z",
55        "M/d/yy HH:mm:ss:SSS Z",
56        "M.d.yy HH:mm:ss:SSS Z",
57        "yyyy-MM-dd-HH.mm.ss.SSSSSS Z"
58      ]
59    }
60  }
```

This is where the Joda-Time parsing is carried out in Logstash. In this configuration, the `date` plugin uses the `timestamp` field to match all the defined datetime formats. On line 52 is the HTTP Server time format, on lines 53-56 are the possible WebSphere Application Server formats and on line 57 is the DB2 time format. The last curly bracket here signifies the end of the filtering block.

```
62  output {
63    elasticsearch {
64      hosts => "elasticsearch:9200"
65      index => "{{ env }}-logs-%{+YYYY.w}"
66    }
67
68    if "_grokparsefailure" in [tags] or "_rubyexception" in [tags] or
      ↪  "_dateparsefailure" in [tags] {
69      stdout { codec => rubydebug }
70    }
71  }
```

The output block is configured to deliver the event out to the Elasticsearch instance running at `elasticsearch:9200` and to store the event as a document in the index `{{ env }}-logs-%{+YYYY.w}`. Here again the Ansible variable `env` that was defined in Chapter 5.2.1 is used. Logstash will parse the datetime pattern `%{+YYYY.w}` into a format of `YEAR.WEEK`. So for example, an event might be stored in an index called `prod-logs-2017.21`.

Lines 68-70 make Logstash print the event as a Ruby debug message to stdout using the `stdout` plugin combined with the `rubydebug` codec, in the case that any of the three conditions are met: the grok parser failed to find a match for the log line of the current event, the `ruby` plugin threw an exception, or if the `date` plugin fails to parse the timestamp.

The commands and the full configuration file can be found in Appendices 8 and 8.

5.2.5 Kibana

Configuring Kibana, like Elasticsearch, was rather simple:

```
$ mkdir -p files/template/kibana
$ vim files/template/kibana/kibana.yml.j2
```

The contents of `kibana.yml.j2` were:

```
1  server.host: "0.0.0.0"
2  server.name: "{{ kibana_config.server_name }}"
3  server.basePath: "/kibana"
4
5  elasticsearch.url: "http://elasticsearch:9200"
6  elasticsearch.requestTimeout: 90000
7
8  xpack.security.enabled: {{ kibana_config.xpack.security }}
9  xpack.monitoring.enabled: {{ kibana_config.xpack.monitoring }}
10 xpack.graph.enabled: {{ kibana_config.xpack.graph }}
11 xpack.watcher.enabled: {{ kibana_config.xpack.watcher }}
12 xpack.reporting.enabled: {{ kibana_config.xpack.reporting }}
13 xpack.ml.enabled: {{ kibana_config.xpack.ml }}
```

As with Elasticsearch in Chapter 5.2.2, Kibana is also made to listen for connections in all available network interfaces by giving `server.host` the value of `0.0.0.0`. On the second line `server.name` being set to `{{ env }}-kibana1` makes this Kibana instance uniquely identifiable. For the production environment the third value would for example be `prod-kibana1`. On the following two lines of the file the connection to Elasticsearch's HTTP REST API is configured to use plain unencrypted HTTP with a request timeout of 90 seconds, after which Kibana's state will change red and it will be unable to operate until it can again establish a connection to Elasticsearch. Using the container's name `elasticsearch` here means that the Kibana container will be linked to Elasticsearch and Docker will automatically do the name resolution to the proper IP address. Finally on lines 8-13 the X-Pack add-ons are disabled as configured in Chapter 5.2.1.

The commands and the configuration file can be found in Appendices 8 and 8.

5.2.6 Nginx

The file and directory structure for the Nginx files were created like so:

```
$ mkdir -p files/nginx
$ mkdir -p files/template/nginx
$ vim files/template/nginx/default.conf.j2
$ vim files/template/nginx/index.html.j2
```

The file `default.conf.j2` contains the actual web server configuration needed for this assignment, and is meant to overwrite the `/etc/nginx/conf.d/default.conf` file in the Nginx container. `index.html.j2` is the Jinja2 template file for the website default landing page that was used to link to the Kibana and Elasticsearch reverse proxies.

```
1  server {
2    listen  80   default_server;
3    server_name  {{ http_hostname }};
4    return  301  https://$server_name$request_uri;
5  }
```

In this `server` configuration block is defined a simple plain and unencrypted HTTP server behind port 80 the only job of which is to serve a permanent redirect to the HTTPS encrypted version of the website. As `server_name` was set to `{{ http_hostname }}`, it was for example in the production environment's case resolved to `elk-prod.example.com`, as defined by the variable set in Chapter 5.2.1.

```
7  server {
8    listen  443    default_server  ssl  http2;
9    server_name    {{ http_hostname }};
10
11   ssl_certificate      /ssl/{{ http_hostname }}.crt;
12   ssl_certificate_key  /ssl/{{ http_hostname }}.key;
```

The second server configuration block is set to listen to the HTTPS port 443 for SSL only, optionally for HTTP/2 when the client supports it. The difference in this block to the HTTP server block is that no redirect instruction is given and that the SSL key and certificate are used.

```
14   location  /  {
15     root   /usr/share/nginx/html;
16     index  index.html;
17   }
```

This is the default location block that the users will hit when accessing the website's index. Its document root is defined as `/usr/share/nginx/html` where the `index.html` index file resides.

```
19    location  ~  ^/(kibana|elasticsearch)$ {
20      return  301  https://$server_name$request_uri/;
21    }
```

This location block is a simple helper that will permanently redirect users who try to access Kibana without the trailing forward slash, to the proper location, for example, from `https://elk-prod.example.com/kibana` to `https://elk-prod.example.com/kibana/`. This is necessary for the reverse proxy to properly work, because if the trailing forward slash is missing Nginx would send an improper request to the backend application.

```
23    location  ~  ^/kibana/.*$ {
24      rewrite           /kibana/(.*)  /$1  break;
25      proxy_pass        http://kibana:5601;
26      proxy_set_header  X-Forwarded-For  $proxy_add_x_forwarded_for;
27      proxy_buffering   off;
28
29      auth_basic            "Kibana";
30      auth_basic_user_file  /auth/kibana.htpasswd;
31    }
```

Here is the first of the two reverse proxy blocks that Nginx was serving. In this block everything located under the `/kibana/` path is forwarded to the Kibana instance running in its container at `http://kibana:5601`. The first thing done in this block on line 24 is that a part of the request the user sent to Nginx will be stripped off, more specifically the `kibana/` portion, after which the request is passed on to Kibana. On line 26 the X-Forwarded-For header is set, a common method of providing applications behind proxies the actual IP address of the user, because by normal means the application behind the proxy will only see the IP address of the proxy itself. On line 27 proxy buffering is disabled. Proxy buffering makes Nginx to first completely load the requested content from the backend application into memory and only then send it off to the user; by disabling this functionality the content from the application is streamed directly to the user immediately when Nginx receives it.

On lines 29-30 basic HTTP authentication is enabled for this location block. When the user accesses it with their browser, the browser will give them a popup asking for authorization to the realm `Kibana` and Nginx will authenticate the users against the credential store `/auth/kibana.htpasswd`.

```
33    location  ~  ^/elasticsearch/.*$ {
34      rewrite           /elasticsearch/(.*)  /$1  break;
35      proxy_pass        http://elasticsearch:9200;
```

```
36      proxy_set_header  X-Forwarded-For  $proxy_add_x_forwarded_for;
37      proxy_buffering   off;
38
39      auth_basic            "Elasticsearch";
40      auth_basic_user_file  /auth/elasticsearch.htpasswd;
41    }
```

The `/elasticsearch/` location block is the same as the `/kibana/` block, except the connections are passed through to Elasticsearch and a different credential store is used instead.

```
1  <h2>ELK - {{ env }}</h2>
2  <li> <a href="/kibana">Kibana</a>
3  <li> <a href="/elasticsearch">Elasticsearch</a>
```

The `index.html.j2` file is simple and its only purpose is to serve as a portal of sorts for the users to easily be able to differentiate between the two environments and to provide clickable links to the applications. As with some of the other configuration files, the `env` variable was used here as well to tell the user which environment they are currently using.

The SSL certificate and key used on lines 11-12 were generated with the following commands:

```
$ cd files/nginx/
$ NAME=elk-prod.example.com
$ openssl req -newkey rsa:4096 -keyout $NAME.key \
    -new -x509 -out $NAME.crt -days 3650 \
    -nodes -subj "/C=FI/CN=$NAME" \
    -extensions sAN -config <(awk -v name=$NAME \
      '{print $0}
       END{
       print "[sAN]"
       print "subjectAltName=DNS:"name
      }' /etc/pki/tls/openssl.cnf)
```

The files were generated with OpenSSL as a 4096-bit RSA key/certificate pair. The certificate will be self-signed and valid for 10 years, until 2027.

Along with the normal procedure of assigning the certificate to a common name (CN), it was also assigned to a subject alternate name (sAN) because in Google Chrome, as of version 58 (released on 19th of April 2017), the field is mandatory. (Sleevi 2017)

The same commands were repeated for the value of `elk-test.example.com` for the `NAME` variable.

All the commands and files in this chapter can be found in Appendices 8, 8 and 8.

5.2.7 Logstash-forwarder-java

Logstash-forwarder-java was chosen as the program to ship the logs with because there are quite a few server environments running IBM AIX.
Logstash-forwarder-java being written in Java, is very portable and can run on these servers with no problems. While Linux servers were used for the implementation description instead, the principle of the methods applied remain the same for AIX.

Unfortunately the official version of logstash-forwarder-java does not support multiline events, which was very much a requirement in the project, so a fork was used instead and compiled from source:

```
$ mkdir files/bin/logstash-forwarder-java
$ cd files/bin/logstash-forwarder-java
$ wget
 ↪  https://github.com/Sentido-Labs/logstash-forwarder-java/archive/master.zip
$ unzip master.zip
$ cd logstash-forwarder-java-master/
$ apt install unzip maven openjdk-8-jdk
$ mvn package
$ unzip target/logstash-forwarder-java-0.2.4-bin.zip
$ cp -r logstash-forwarder-java-0.2.4/* ../
$ cd ..
```

Here the required packages for building the fork's source code were installed and the fork was built with Apache Maven.

Java Service Wrapper from Tanuki Software Ltd. was used on top of logstash-forwarder-java. The wrapper makes Java programs available as system-like services. The wrapper was downloaded and extracted in the same directory and cleanup was performed:

```
$ wget https://wrapper.tanukisoftware.com/download/3.5.32/wrapper-linux-x86-64
 ↪  -3.5.32.tar.gz
$ tar -xzf wrapper-linux-x86-64-3.5.32.tar.gz
$ cp -r wrapper-linux-x86-64-3.5.32/* .
$ rm -r logstash-forwarder-java-master master.zip wrapper-linux-x86-64-3.5.32
 ↪  wrapper-linux-x86-64-3.5.32.tar.gz
```

After which the wrapper's configuration file was created:

```
$ cp src/conf/wrapper.conf.in conf/wrapper.conf
$ vim wrapper.conf
```

The wrapper's configuration is a simple INI-style file, consisting of lines with `key=value` properties. The modified and added properties can be seen in Table 5.

Table 5: Wrapper configuration modified parameters.

KEY	VALUE
wrapper.java.command	../jre/bin/java
wrapper.java.classpath.2	../logstash-forwarder-java-0.2.4.jar
wrapper.app.parameter.1	info.fetter.logstashforwarder.Forwarder
wrapper.app.parameter.2	-config
wrapper.app.parameter.3	../config.json
wrapper.app.parameter.4	-sincedb
wrapper.app.parameter.5	../sincedb.json
wrapper.app.parameter.6	-tail
wrapper.console.title	logshipper

Then the script to start and stop the service was created and modified:

```
$ cp src/bin/sh.script.in bin/wrapper.sh
$ vim bin/wrapper.sh
$ perl -pi -e 's/\@app\.(long\.)?name\@/logshipper/' bin/wrapper.sh
$ perl -pi -e 's/^#RUN_AS_USER=/RUN_AS_USER=logshipper/' bin/wrapper.sh
```

Here Perl was used to find and replace all occurrences of `@app.name@` and `@app.long.name@` with `logshipper`, and changing the only occurrence of a line starting with `#RUN_AS_USER=` to `RUN_AS_USER=logshipper`. This was done to make the wrapper identify itself as `logshipper` and to have the script start the program as the user `logshipper`.

When the wrapper was configured, the configuration file template for the forwarder was created:

```
$ mkdir -p files/template/logstash-forwarder-java
$ vim files/template/logstash-forwarder-java/config.json.j2
```

It is a JSON-formatted file that specifies the connection options and the files watched.

```
1  { "network": {
2       "servers": ["{{ logstash_address }}:{{ logstash_port }}"],
3       "timeout": 30,
4       "ssl ca": "/opt/logstash-forwarder-java/{{ logstash_address }}.jks"
```

In the `network` part of the configuration the target server is specified to `{{ logstash_address }}:{{ logstash_port }}` that resolved, for example, to `elk-prod.example.com:5000`. A thirty second connection timeout is set and the SSL CA Java KeyStore file generated in Chapter 5.2.3 is given. While the program accepts multiple servers, since there was only one Logstash cacher instance per environment used in this project, only one was specified.

```
6       { "paths": [ "/opt/IBM/WebSphere/AppServer/profiels/AppSrv01/server1/
         ↪  SystemOut.log"
         ↪  ],
7        "fields": { "type": "was",
8                    "tags": "AppSrv01,server1,systemout" },
9     "multiline": { "pattern": "(^\\[)",
10                   "negate": "true", "what": "Previous" } },
11      { "paths": [ "/opt/IBM/HTTPServer/logs/access_log" ],
12       "fields": { "type": "ihs",
13                   "tags": "accesslog" } },
14      { "paths": [ "/home/db2inst1/sqllib/db2dump/db2diag*log" ],
15       "fields": { "type": "db2",
16                   "tags": "diaglog" },
17    "multiline": { "pattern": "^(?>\\d\\d){1,2}-(?:0?[1-9]|1[0-9])-",
18                   "negate": "true", "what": "Previous" } } ] }
```

Then the `files` part which is an array of objects specifying the files being watched and the field options for those files. The file paths support file glob patterns, which the DB2 file path is using. All the three files and their parameters are listed in Table 6.

Table 6: Logstash-forwarder-java watched files.

FILE	TYPE	TAGS
SystemOut.log	was	AppSrv01,server1,systemout
access_log	ihs	accesslog
db2diag*log	db2	db2diag

Looking at the table above, as an example logs in the `SystemOut.log` file is sent with additional fields, where the field `type` is set to `was`, and the field `tags` is set to `AppSrv01,server1,systemout`. Both of these fields were set to make searching for the correct type of data easier. The type being set made the logs more categorizable and when coupled with the tags, it makes it easy for the user browsing the logs with Kibana to know exactly where the log originated from. The `tags` field is a string since configuring fields as arrays is not possible with logstash-forwarder-java, so the field is split into an array at parsing time as configured in Chapter 5.2.4.

For DB2 logs the file glob pattern `db2diag*log` was used, because the log files start from `db2diag.log` but when the maximum log file size is reached, log rotation makes the next log files become `db2diag.0.log`, `db2diag.1.log` and so on. The pattern is used because it matches all the variations.

Both of the WAS and DB2 watchers use the multiline option, which with the configured values will collect lines for the event and send it only after the pattern

specified is seen. The patterns are Java-compatible regexp and look for lines starting with the WAS and DB2 log event hallmarks.

The commands and files can be found in Appendices 8, 8.

5.3 Ansible role and playbook configurations

5.3.1 Dependencies

The dependencies role's purpose was to install the Docker Engine and all the packages required to control it. All it needed was the main task file:

```
$ mkdir -p roles/elk_deps/tasks
$ vim roles/elk_deps/tasks/main.yml
```

The task file itself had only seven tasks in total.

```
1  ---
2
3  - name: Enable the EPEL repository and install Python SELinux support
4    yum: name={{ item }}
5    with_items:
6    - epel-release
7    - libselinux-python
8
9  - name: Install PIP
10   yum: name=python-pip
```

The first task uses the `yum` module to install the `epel-release` and `libselinux-python` packages. `libselinux-python` is a package allowing Python to interact with SELinux and is required by Ansible when the target host has SELinux enabled. `epel-release` installs and enables the EPEL repository where `python-pip` is located.

The task utilizes the `with_items` option which makes the task run `yum` in a loop, replacing `{{ item }}` with the value of the item in each iteration, so for example for the first loop, `{{ item }}` would be replaced by `epel-release`, effectively making the task run `yum: name=epel-release`. However, `python-pip` needs to be installed separately in its own task or the task would fail with the message "`No package matching 'python-pip' found available, installed or updated`".

```
12 - name: Install Python docker-py and passlib modules
13   pip: name={{ item }}
14   with_items:
15   - docker-py
16   - passlib
```

Using `pip` to install two Python modules: `docker-py`, a requirement for Ansible's `docker_container` module and `passlib` which is required by the `htpasswd` module.

```
18  - name: Install Docker CE dependencies
19    yum: name={{ item }}
20    with_items:
21    - device-mapper-persistent-data
22    - lvm2
23
24  - name: Enable Docker CE repository
25    get_url:
26      url: https://download.docker.com/linux/centos/docker-ce.repo
27      dest: /etc/yum.repos.d/docker-ce.repo
28
29  - name: Install latest Docker CE
30    yum: name=docker-ce
```

Then some Docker-specific packages are checked that they are installed, which they usually are in a default CentOS installation. Because the Docker packages are hosted by Docker Inc. themselves, the `get_url` module is used to download the repository configuration into a file called `docker-ce.repo`. When that is done the engine itself is installed with `yum`.

```
32  - name: Enable and start docker
33    systemd: name=docker state=started enabled=yes daemon_reload=yes
```

Finally, the last dependency task uses the `systemd` module to start the Docker Engine and make it start automatically on boot.

The commands and files referenced can be found in Appendices 8 and 8.

5.3.2 Elasticsearch

Compared to the previous role, here a new file was created for handlers. Ansible handlers are special tasks that are run at the end of a play and only when notified by a task that has changed something. Here are the file creations:

```
$ mkdir -p roles/elk_elasticsearch/{tasks,handlers}
$ vim roles/elk_elasticsearch/tasks/main.yml
$ vim roles/elk_elasticsearch/handlers/main.yml
```

First the task file:

```
1  ---
2
3  - name: Set mmap count kernel parameter for Elasticsearch
4    sysctl:
5      name: vm.max_map_count
6      value: 262144
7      sysctl_set: yes
```

Here the `sysctl` module is used to set the kernel parameter `vm.max.map_count` to `262144` as recommended by the Elasticsearch documentation (Elasticsearch BV 2017). This parameter is usually too low for Elasticsearch and might result in out of memory errors.

```
9  - name: Ensure Elasticsearch directory paths exist
10   file:
11     path: "{{ item }}"
12     state: directory
13     recurse: yes
14   with_items:
15   - /srv/conf/elasticsearch
16   - /srv/data/elasticsearch
```

This task uses `with_items` for the two directory paths and uses the `file` module to create them. The `file` module's parameters specify that the paths should be directories, created recursively (equivalent to `mkdir -p`).

```
18 - name: Give Elasticsearch data directory uid/gid 1000 ownership
19   file:
20     path: /srv/data/elasticsearch
21     state: directory
22     recurse: yes
23     owner: 1000
24     group: 1000
```

This task works in the same kind of way except its only job is to make the data directory owned by a user and group with the id 1000. The ownership is required because inside the container Elasticsearch is run as a user with the same user and group ids.

```
26 - name: Render Elasticsearch config
27   template:
28     src: files/template/elasticsearch/elasticsearch.yml.j2
29     dest: /srv/conf/elasticsearch/elasticsearch.yml
30   notify: Restart Elasticsearch
```

In this task the `template` module is used. This will render Jinja2 template files, processing any Jinja2-specific logic and replacing variable references with their corresponding values. `notify` is used to call the handler named `Restart Elasticsearch`, which will restart the container if this task resulted in a change.

```
32 - name: Run Elasticsearch container
33   docker_container:
34     name: elasticsearch
35     image: docker.elastic.co/elasticsearch/elasticsearch:5.4.0
36     env:
37       ES_JAVA_OPTS: "-Xms1g -Xmx1g"
38     log_driver: json-file
```

```
39      log_options:
40        max-size: 10m
41        max-file: "5"
42      ports:
43      - 127.0.0.1:9200:9200/tcp
44      volumes:
45      - /srv/conf/elasticsearch/elasticsearch.yml:/usr/share/elasticsearch/
         ↪  config/elasticsearch.yml:ro
46      - /srv/data/elasticsearch:/usr/share/elasticsearch/data:rw
47      - /etc/localtime:/etc/localtime:ro
```

Breaking this task down, it uses the `docker_container` module to create and start a named container called `elasticsearch`. It uses the official Elasticsearch version 5.4.0 image provided by the company's official Docker Registry. The `env` option takes key-value parameters that are passed as global environment variables to the created container, and here the `ES_JAVA_OPTS` environment variable is set to limit Elasticsearch's Java heap size to one gigabyte.

`log-driver` and `log-options` define how the container's logging is done, in this case `json-file` is used to make the Docker Engine write logs into a JSON file while limiting the maximum number of said log files to `5` with each file being limited to being at most `10` megabytes in size. The value of `max-file` is in quotes because of the way Ansible processes YAML, without them the value would be sent to the Docker API as an integer, when the API excepts a string. This would result in an error when the task is run.

The `ports` parameter specifies a list of ports that can be bound to the host. In this case the TCP port `9200` within the container is bound to `127.0.0.1:9200` on the host which makes the port accessible from outside the container, but only locally within the host machine itself.

Files, directories, etc. can be passed through to a container with the `volumes` option, which takes a list of paths to mount in the container. For example with the first entry, the Elasticsearch configuration file which was rendered with the `template` module above is located at `/srv/conf/elasticsearch/elasticsearch.yml` on the host machine and is mounted at `/usr/share/elasticsearch/config/elasticsearch.yml` inside the container as `read only`. The file is mounted as read only because the container does not need to modify it, unlike with the second entry which has the parameter `rw` to signify that. As the last volume the `/etc/localtime` file is mounted to set the container's time settings to match the host's settings.

The handler file is rather simple:

```
1  ---
2
3  - name: Restart Elasticsearch
4    docker_container:
5      name: elasticsearch
6      restart: yes
```

When run, it uses the `docker_container` module to restart the Elasticsearch container.

The commands and files referenced can be found in Appendices 8 and 8.

5.3.3 Redis

Since there was no special configuration created for Redis, a handler is not needed. Only the task file was created:

```
$ mkdir -p roles/elk_redis/tasks
$ vim roles/elk_redis/tasks/main.yml
```

The file is broken into three tasks:

```
1  ---
2
3  - name: Set overcommit memory kernel parameter for Redis
4    sysctl:
5      name: vm.overcommit_memory
6      value: 1
7      sysctl_set: yes
```

Like with Elasticsearch, a kernel parameter is set for Redis as well. The `vm.overcommit_memory` parameter is enabled as recommended and if it is not, Redis' background saving may not work due to the way it forks the process and dumps the database (Redis 2017).

```
9  - name: Ensure Redis directory paths exist
10   file:
11     path: /srv/data/redis
12     state: directory
13     recurse: yes
```

Here the data directory for Redis' background saving is created.

```
15 - name: Run Redis container
16   docker_container:
17     name: redis
18     image: redis:3.2-alpine
19     log_driver: json-file
20     log_options:
```

```
21        max-size: 10m
22        max-file: "5"
23      ports:
24      - 127.0.0.1:6379:6379/tcp
25      volumes:
26      - /srv/data/redis:/data:rw
27      - /etc/localtime:/etc/localtime:ro
```

And the container itself is created and started. This time a third party Registry is not specified, so Docker will pull the image from the Docker Hub. The image is an official Redis version 3.2 image that's based on Alpine Linux. Logging options are the same as with Elasticsearch but this time the port `6379`, which is the default Redis port, is bound to the host.

The commands and files referenced can be found in Appendices 8 and 8.

5.3.4 Logstash

Both Logstash instances are deployed from the same role, so only one role was needed to do the job:

```
$ mkdir -p roles/elk_ls/{tasks,handlers}
$ vim roles/elk_ls/tasks/main.yml
$ vim roles/elk_ls/handlers/main.yml
```

Following is the task file:

```
1   ---
2
3   - name: Ensure Logstash directory paths exist
4     file:
5       path: "{{ item }}"
6       state: directory
7       recurse: yes
8     with_items:
9     - /srv/ssl/ls-cacher
10    - /srv/conf/ls-cacher
11    - /srv/conf/ls-parser
12    - /srv/conf/ls-parser-patterns
```

A simple loop to prepare all the directories for the configuration files for both Logstash instances.

```
14  - name: Render Logstash cacher config
15    template:
16      src: "{{ item }}"
17      dest: "/srv/conf/ls-cacher/{{ item | basename |
          ↪  regex_replace('\\.j2$', '') }}"
18    with_fileglob: files/template/ls-cacher/*.conf.j2
19    notify: Restart Logstash cacher
20
21  - name: Render Logstash parser config
22    template:
```

```
23      src: "{{ item }}"
24      dest: "/srv/conf/ls-parser/{{ item | basename |
         ↪  regex_replace('\\.j2$', '') }}"
25    with_fileglob: files/template/ls-parser/*.conf.j2
26    notify: Restart Logstash parser
```

These two tasks are essentially the same, just with different paths. `with_fileglob` acts much like `with_items` except that it can generate the list of items to loop through itself, when given a file glob pattern.

Instead of the usual `{{ item }}`, the `dest` parameter is a bit more complicated. `basename` and `regex_replace` are both Jinja2 filters in Ansible's core and can be used within Jinja2 syntax. For example when `with_fileglob` matched `files/template/ls-cacher/cacher.conf.j2`, it passed it on as a string to the Jinja2 filter. The string was then processed by `basename`, transforming it to `cacher.conf.j2` and passed to `regex_replace`, which replaced `.j2` from the end of the string with nothing, if found, so the end result in this case was `cacher.conf` and the contents of the `dest` parameter were `/srv/conf/ls-cacher/cacher.conf`.

```
28  - name: Copy Logstash cacher SSL files
29    copy: src={{ item }} dest=/srv/ssl/ls-cacher/
30    with_items:
31    - "files/ls-cacher/{{ http_hostname }}.key"
32    - "files/ls-cacher/{{ http_hostname }}.crt"
33    notify: Restart Logstash cacher
```

Fourth on the task list is the copying of the Logstash cacher SSL key and certificate, with a notification being sent to `Restart Logstash cacher`.

```
35  - name: Copy Logstash parser patterns
36    copy: src={{ item }} dest=/srv/conf/ls-parser-patterns/
37    with_fileglob:
38    - files/ls-parser/patterns/*
39    notify: Restart Logstash parser
```

Here `with_fileglob` and `notify` are used again, this time to copy the custom named patterns that were created in Chapter 5.2.4.

```
41  - name: Run Logstash containers
42    docker_container:
43      name: "{{ item.name }}"
44      image: docker.elastic.co/logstash/logstash:5.4.0
45      env:
46        LS_JAVA_OPTS: "-Xms256m -Xmx256m"
47        XPACK_MONITORING_ENABLED: "false"
48      log_driver: json-file
49      log_options:
50        max-size: 10m
51        max-file: "5"
52      links: "{{ item.links | default([]) }}"
53      ports: "{{ item.ports | default([]) }}"
54      volumes: "{{ item.volumes |
         ↪  default(['/etc/localtime:/etc/localtime:ro']) }}"
```

The creation of the actual containers, only this time with a loop which is given a list of dictionaries. Because the `links` and `ports` options only accept a list as their parameter, a Jinja2 filter is used to default to an empty list (`[]`) in the case `item.links` or `item.ports` are empty or not defined. Same applies to `volumes`, except the default value will be a list with the `/etc/localtime` file being mounted. This makes the `item.name` the only required parameter, and without it the task would fail.

With the `env` option the Logstash instance's Java heap size is set to `256` megabytes and to save from having to create a separate configuration file for a single option, the X-Pack add-on was disabled here with the `XPACK_MONITORING_ENABLED` environment variables set to `"false"`. Much like with `max-file`, the quotes are important here as well, because Logstash will case sensitively only accept `true` or `false` and the YAML parser Ansible uses would give it the value of `False` instead as that is the string representation of a Python boolean set to false.

```
55    with_items:
56    - name: ls-cacher
57      links:
58      - redis
59      ports:
60      - 0.0.0.0:5000:5000/tcp
61      volumes:
62      - /srv/conf/ls-cacher:/usr/share/logstash/pipeline:ro
63      - /srv/ssl/ls-cacher:/ssl:ro
64      - /etc/localtime:/etc/localtime:ro
65    - name: ls-parser
66      links:
67      - redis
68      - elasticsearch
69      volumes:
70      - /srv/conf/ls-parser:/usr/share/logstash/pipeline:ro
71      - /srv/conf/ls-parser-patterns:/patterns:ro
72      - /etc/localtime:/etc/localtime:ro
```

The actual parameters for the `docker_container` task are defined here. Both dictionaries have the `name` parameter set, so the task will not fail. As can be seen, the `ls-parser` parameters are missing `ports` definitions but this would not cause the task to fail because of the `default([])` filter from above.

Because these containers have links to other containers, namely `elasticsearch` and `redis`, those containers need to already exist and be running when these containers are created, or else the task would fail.

```
1  ---
2
```

```
3  - name: Restart Logstash cacher
4    docker_container:
5      name: ls-cacher
6      restart: yes
7
8  - name: Restart Logstash parser
9    docker_container:
10     name: ls-parser
11     restart: yes
```

Both instances have their own `notify` handlers because was no need to restart both if only the other one's configuration had changed.

The commands and files referenced can be found in Appendices 8 and 8.

5.3.5 Kibana

Kibana's file structure was created with the following commands:

```
$ mkdir -p roles/elk_kibana/{tasks,handlers}
$ vim roles/elk_kibana/tasks/main.yml
$ vim roles/elk_kibana/handlers/main.yml
```

The task file as follows:

```
1  ---
2
3  - name: Ensure Kibana directory paths exist
4    file:
5      path: /srv/conf/kibana
6      state: directory
7      recurse: yes
```

Only one directory is created this time, so there is no need for a loop.

```
9  - name: Render Kibana config
10   template:
11     src: files/template/kibana/kibana.yml.j2
12     dest: /srv/conf/kibana/kibana.yml
13   notify: Restart Kibana
```

Only one configuration file as well, and the `Restart Kibana` handler is notified with `notify`.

```
15 - name: Run Kibana container
16   docker_container:
17     name: kibana
18     image: docker.elastic.co/kibana/kibana:5.4.0
19     log_driver: json-file
20     log_options:
21       max-size: 10m
22       max-file: "5"
23     links:
24     - elasticsearch
25     ports:
26     - 127.0.0.1:5601:5601/tcp
27     volumes:
28     - /srv/conf/kibana/kibana.yml:/usr/share/kibana/config/kibana.yml:ro
29     - /etc/localtime:/etc/localtime:ro
```

Most of this is the same as what the previously configured containers had, with the exception of this container being linked to `elasticsearch`, which means this container needs to be created after the Elasticsearch one. Also the port `5601` is bound for local access only.

```
---

- name: Restart Kibana
  docker_container:
    name: kibana
    restart: yes
```

And the handler for restarting the `kibana` container when needed.

The commands and files referenced can be found in Appendices 8 and 8.

5.3.6 Nginx

Nginx role structure was created with these commands:

```
$ mkdir -p roles/elk_nginx/{tasks,handlers}
$ vim roles/elk_nginx/tasks/main.yml
$ vim roles/elk_nginx/handlers/main.yml
$ mkdir -p roles/logstash-forwarder-java/{tasks,handlers}
```

The role configuration with the following:

```
---

- name: Ensure Nginx directory paths exist
  file:
    path: "{{ item }}"
    state: directory
    recurse: yes
  with_items:
  - /srv/conf/nginx
  - /srv/ssl/nginx
  - /srv/auth/nginx
  - /srv/html
```

Using a loop to create the directories that are mounted in the Nginx container.

```
- name: Render Nginx config
  template:
    src: files/template/nginx/default.conf.j2
    dest: /srv/conf/nginx/default.conf
  notify: Restart Nginx

- name: Render Nginx index.html
  template:
    src: files/template/nginx/index.html.j2
    dest: /srv/html/index.html
```

Since the web server config and the landing page HTML file are both Jinja2 templates, they are rendered using the `template` module.

```
25  - name: Copy Nginx SSL files
26    copy:
27      src: "{{ item }}"
28      dest: /srv/ssl/nginx/
29    with_fileglob:
30    - "files/nginx/{{ http_hostname }}.key"
31    - "files/nginx/{{ http_hostname }}.crt"
32    notify: Restart Nginx
```

Here the SSL key and certificate are copied over.

```
34  - name: Create Nginx Kibana htpasswd file
35    htpasswd:
36      name: "{{ item.user }}"
37      password: "{{ item.pass }}"
38      path: /srv/auth/nginx/kibana.htpasswd
39    with_items: "{{ kibana_config.users }}"
40    notify: Restart Nginx
41
42  - name: Create Nginx Elasticsearch htpasswd file
43    htpasswd:
44      name: "{{ item.user }}"
45      password: "{{ item.pass }}"
46      path: /srv/auth/nginx/elasticsearch.htpasswd
47    with_items: "{{ elasticsearch_config.users }}"
48    notify: Restart Nginx
```

These two tasks use the `htpasswd` module to generate the credential stores for HTTP basic authentication. A list of dictionaries is passed to `with_items` which loops through them, enabling the use of `item.user` and `item.pass`. These were defined in the `group_vars` files in Chapter 5.2.1. By default the module uses the `apr_md5_crypt` encryption scheme from Python's `passlib` module (Red Hat Inc. 2017).

```
50  - name: Run Nginx container
51    docker_container:
52      name: nginx
53      image: nginx:1.13-alpine
54      log_driver: json-file
55      log_options:
56        max-size: 10m
57        max-file: "5"
58      links:
59      - elasticsearch
60      - kibana
61      ports:
62      - 0.0.0.0:80:80/tcp
63      - 0.0.0.0:443:443/tcp
64      volumes:
65      - /srv/conf/nginx:/etc/nginx/conf.d:ro
66      - /srv/ssl/nginx:/ssl:ro
67      - /srv/auth/nginx:/auth:ro
68      - /srv/html:/usr/share/nginx/html:ro
69      - /etc/localtime:/etc/localtime:ro
```

Again when compared to the settings of the previously defined containers, the image used is from Docker Hub and is the official Nginx image for version 1.13 based on Alpine Linux. As this container is linked to both `elasticsearch` and

`kibana`, it is necessary to create and start those containers before this one. The default HTTP and HTTPS ports, `80` and `443` respectively, are exposed to every network interface on the host, so that this container can be accessed from outside the host machine. Finally, all the necessary files and directories are mounted into various locations inside the container as read only, because there is no need for the container to modify the contents of any of the files and directories.

```
1  ---
2
3  - name: Restart Nginx
4    docker_container:
5      name: nginx
6      restart: yes
```

Finally the handler for restarting the Nginx container when configuration changes are deployed.

The commands and files referenced can be found in Appendices 8 and 8.

5.3.7 Logstash-forwarder-java

The role for logstash-forwarder-java was created with the following structure:

```
$ mkdir -p roles/logstash-forwarder-java/{tasks,handlers}
$ vim roles/logstash-forwarder-java/tasks/main.yml
$ vim roles/logstash-forwarder-java/handlers/main.yml
```

The task file creates the user for the wrapper to run as, copies all the relevant files and installs the IBM JRE:

```
1  ---
2
3  - name: Create logshipper user
4    user: name=logshipper
```

Here the `user` module is used to create the user `logshipper`.

```
6  - name: Copy logstash-forwarder-java files
7    copy:
8      src: files/bin/logstash-forwarder-java/
9      dest: /opt/logstash-forwarder-java/
10     owner: logshipper
11     group: logshipper
12   notify: Restart logstash-forwarder-java
13
14 - name: Make wrapper.sh executable
15   file:
16     path: /opt/logstash-forwarder-java/bin/wrapper.sh
17     mode: "ug+x"
18
```

In these tasks the `copy` module is used to recursively copy the contents of `files/bin/logstash-forwarder-java/` to `/opt/logstash-forwarder-java/` on the server, with `logshipper` as the owning user and group. The `Restart logstash-forwarder-java` handler is notified in case this task causes any changes on the system. After the files have been copied, the `wrapper.sh` script is made executable.

```
19  - name: Render config.json
20    template:
21      src: files/template/logstash-forwarder-java/config.json.j2
22      dest: /opt/logstash-forwarder-java/config.json
23      owner: logshipper
24      group: logshipper
25    notify: Restart logstash-forwarder-java
```

The program's configuration is rendered with this task, notifying `Restart logstash-forwarder-java` if the configuration was updated.

```
27  - name: Create empty sincedb
28    copy:
29      dest: /opt/logstash-forwarder-java/sincedb.json
30      content: "[]"
31      force: no
32      owner: logshipper
33      group: logshipper
34    notify: Restart logstash-forwarder-java
```

An empty sincedb file is created because otherwise the forwarder would fail to initialize properly, causing an exception being thrown when started.

```
36  - name: Copy Java CA KeyStore
37    copy:
38      src: "files/logstash-forwarder-java/{{ logstash_address }}.jks"
39      dest: /opt/logstash-forwarder-java/
40      owner: logshipper
41      group: logshipper
42    notify: Restart logstash-forwarder-java
```

This task copies over the Java KeyStore file created in Chapter 5.2.3.

```
44  - name: Check is IBM JRE java binary already exists
45    stat: path=/opt/logstash-forwarder-java/jre/bin/java
46    register: stat_java
47    changed_when: False
```

This task checks if the IBM JRE is already installed, by using the `stat` module to check is the `java` binary file exists. It then uses `register` to create a variable called `stat_java` which can be used in the later tasks in this role. Because `stat` will always result in a change, `changed_when` is set to `False` which makes this task to always return `ok`.

```
49  - block:
50    - name: Copy IBM JRE installer
51      copy:
52        dest: /opt/logstash-forwarder-java/
53        src: files/bin/ibm-java-jre-8.0-4.5-x86_64-archive.bin
54        owner: logshipper
55        group: logshipper
56        mode: "ug+x"
```

Here a `block` section is started, which means the tasks inside the block will only be executed if the specified conditions are met. This is to save the trouble of having to specify the wanted condition in every single task where it is needed.

Again, the `copy` module is used, but this time to copy the IBM JRE installer.

```
58    - name: Create IBM JRE installer response file
59      copy:
60        dest: /opt/logstash-forwarder-java/ibm_jre.properties
61        owner: logshipper
62        group: logshipper
63        content: |+
64          INSTALLER_UI=silent
65          LICENSE_ACCEPTED=TRUE
66          USER_INSTALL_DIR=/opt/logstash-forwarder-java/
```

The properties file for the installer is created, by using the `blockinfile` module to write text into the `ibm_jre.properties` file instead of using a template. The syntax `block: |+` means that the text for the option is written properly with each line being on their own lines in the output file, with leading and trailing whitespace removed.

```
68    - name: Install IBM JRE
69      shell: ./ibm-java-jre-8.0-4.5-x86_64-archive.bin -f ibm_jre.properties
70      args:
71        chdir: /opt/logstash-forwarder-java/
72        creates: /opt/logstash-forwarder-java/jre/bin/java
73      notify: Restart logstash-forwarder-java
74
75    when: not stat_java.stat.exists
```

Finally as the last task the `shell` module is used to run the JRE installer, with module `args` `chdir` and `create`, former of which will change to the `/opt/logstash-forwarder-java/` directory before running the command, and the latter signifies that this shell command should create the file `/opt/logstash-forwarder-java/jre/bin/java` and if it does not, the task should fail.

To end the `block` section, a conditional using `when` checks if the variable `stat_java.stat.exists` is set to `False`. This uses the registered `stat_java`

from above, and the `stat.exists` is output from the `stat` module itself. If the `stat_java.stat.exists` variable is not `False`, this block will not be run which means that the IBM JRE will not be installed because it already is.

```
1  ---
2
3  - name: Restart logstash-forwarder-java
4    shell: /opt/logstash-forwarder-java/bin/wrapper.sh restart
```

The handler called by one of the tasks will use the `wrapper.sh` script to restart the program.

The command and files can be found in Appendices 8, 8 and 8.

5.3.8 Elastic Stack playbooks

As the method to separate different tasks with was chosen to be the usage of roles, playbooks were required. To accommodate for both the production and test environments, they both were to have their own separate playbooks:

```
$ vim deploy_elk_test.yml
$ vim deploy_elk_prod.yml
```

Following is the production playbook for calling the roles:

```
1  ---
2
3  - hosts: elk_prod
4    roles:
5    - { name: elk_deps,          tags: deps }
6    - { name: elk_elasticsearch, tags: elasticsearch }
7    - { name: elk_redis,         tags: redis }
8    - { name: elk_ls,            tags: logstash }
9    - { name: elk_kibana,        tags: kibana }
10   - { name: elk_nginx,         tags: nginx }
```

Here with the specification of `hosts: elk_prod`, this playbook will only run against the production group that was defined in the `hosts` file in Chapter 5.2.1.

In the playbook the roles are listed in the order they are executed. There is little room in this configuration to change the order in which the containers are deployed, due to dependencies between containers as described in Table 7.

Table 7: Ansible role deploy dependencies.

↓ deps on →	elk_deps	elk_elastics...	elk_redis	elk_ls	elk_kibana	elk_nginx
elk_deps						
elk_elastics...	X					
elk_redis	X					
elk_ls	X	X	X			
elk_kibana	X	X				
elk_nginx	X	X			X	

The main dependency being `elk_deps`, of course when its packages have already been installed it is not needed any more. Same applies for dependencies to linked containers, if the linked containers are running then there is no need to deploy them along with the container that depends on them.

Both the commands and the playbook files can be found in Appendices 8, 8 and 8.

5.3.9 Logstash-forwarder-java playbooks

This playbook files were created as follows:

```
$ vim deploy_logshipper_test.yml
$ vim deploy_logshipper_prod.yml
```

The playbooks required for the deployment of logstash-forwarder-java are much simpler than the ones for the Elastic Stack, calling only one role. Following is the playbook for production:

```
1  ---
2
3  - hosts: servers_prod
4    roles:
5    - logstash-forwarder-java
```

The playbook will run the `logstash-forwarder-java` role on the servers belonging in the `server_prod` group, which was defined in Chapter 5.2.1.

The commands and files for both environments can be found in Appendices 8, 8 and 8.

5.4 Deployment

5.4.1 Elastic Stack

The production environment was deployed with the following command:

```
$ ansible-playbook deploy_elk_prod.yml -u vagrant
```

And as can be seen on this truncated output from the command, the deployment was successful:

```
SSH password:
SUDO password[defaults to SSH password]:

PLAY [elk_prod]

TASK [Gathering Facts]
ok: [elk_prod_node1]
# ...
# ...
# ...
RUNNING HANDLER [elk_nginx : Restart Nginx]
changed: [elk_prod_node1]

PLAY RECAP
elk_prod_node1             : ok=37   changed=35   unreachable=0    failed=0

Playbook run took 0 days, 0 hours, 2 minutes, 57 seconds
```

The commands used and the full output from the deployment for both production and test can be found in Appendices 8, 8 and 8.

5.4.2 Logstash-forwarder-java

Logstash-forwarder-java was deployed on the production server with the following command:

```
$ ansible-playbook deploy_logshipper_prod.yml -u vagrant
```

With the following output:

```
SSH password:
SUDO password[defaults to SSH password]:

PLAY [servers_prod]

TASK [Gathering Facts]
ok: [server_prod_node1]
# ...
# ...
# ...
RUNNING HANDLER [logstash-forwarder-java : Restart logstash-forwarder-java]
changed: [server_prod_node1]

PLAY RECAP
server_prod_node1          : ok=12   changed=10   unreachable=0    failed=0

Playbook run took 0 days, 0 hours, 0 minutes, 51 seconds
```

The deployment was successful, as seen in the above output.

The commands and full output for both production and test can be found in Appendices 8, 8 and 8.

6 Verification

Because of the fact that both production and test environments are identical, verification was done only on the production environment.

6.1 Redeployment with changed configuration

To verify that the `notify` calls defined in some of the role tasks are working, a new user was added to the production users list:

```
$ vim group_vars/elk_prod.yml
users:
# ...
- user: newuser
  pass: newpass098
```

And by running the playbook again, the only tasks that resulted in a change were `Create Nginx Kibana htpasswd file` and `Create Nginx Elasticsearch htpasswd file`, which notified the handler `Restart Nginx` to restart the container:

```
SSH password:
SUDO password[defaults to SSH password]:

PLAY [elk_prod]
# ...
# ...
# ...
TASK [elk_nginx : Create Nginx Kibana htpasswd file]
ok: [elk_prod_node1] => (item={u'user': u'produser', u'pass': u'pass456'})
ok: [elk_prod_node1] => (item={u'user': u'admin', u'pass': u'adminerino456'})
changed: [elk_prod_node1] => (item={u'user': u'newuser', u'pass':
 ↪  u'newpass098'})

TASK [elk_nginx : Create Nginx Elasticsearch htpasswd file]
ok: [elk_prod_node1] => (item={u'user': u'produser', u'pass': u'pass456'})
ok: [elk_prod_node1] => (item={u'user': u'admin', u'pass': u'adminerino456'})
changed: [elk_prod_node1] => (item={u'user': u'newuser', u'pass':
 ↪  u'newpass098'})

TASK [elk_nginx : Run Nginx container]
ok: [elk_prod_node1]

RUNNING HANDLER [elk_nginx : Restart Nginx]
changed: [elk_prod_node1]

PLAY RECAP
```

```
elk_prod_node1             : ok=33   changed=3    unreachable=0    failed=0

Playbook run took 0 days, 0 hours, 0 minutes, 19 seconds
```

The full Ansible output can be found in Appendix 8.

6.2 Operating system

The target host's operating system was verified with the following commands:

```
$ yum info epel-release libselinux-python python2-pip | grep -E "Name|Repo"
Name        : docker-ce
Repo        : installed
Name        : epel-release
Repo        : installed
Name        : libselinux-python
Repo        : installed
Name        : python2-pip
Repo        : installed
$ pip list | grep -E "(docker-py|passlib) "
docker-py (1.10.6)
passlib (1.7.1)
```

These are the `yum` and `pip` packages installed with the `elk_deps` role, configured in Chapter 5.3.1.

```
$ sysctl vm.max_map_count
vm.max_map_count = 262144
$ sysctl vm.overcommit_memory
vm.overcommit_memory = 1
```

These kernel parameters were set in the Elasticsearch and Redis deployment tasks configured in Chapters 5.3.2 and 5.3.3.

As can be seen from the above outputs, the container and software dependencies were successfully applied.

6.3 Containers

Ansible does not keep track of whether the container actually stays up (as it is not supposed to) so it might be a good idea to check that the containers are still running after the deployment has completed. This was done by logging in to the server running the containers, in this case `elk-prod.example.com`, and running the following command:

```
$ docker ps -a --format '{{.Names}} - {{.Status}}'
nginx - Up 9 minutes
kibana - Up 9 minutes
ls-parser - Up 9 minutes
ls-cacher - Up 9 minutes
redis - Up 9 minutes
elasticsearch - Up 9 minutes
```

And since all the containers had been up for over nine minutes, it was safe to assume that there were no silent errors from the deployment.

By checking for ports instead of status:

```
$ docker ps -a --format '{{.Names}} - {{.Ports}}'
nginx - 0.0.0.0:80->80/tcp, 0.0.0.0:443->443/tcp
kibana - 127.0.0.1:5601->5601/tcp
ls-parser - 5044/tcp, 9600/tcp
ls-cacher - 0.0.0.0:5000->5000/tcp, 5044/tcp, 9600/tcp
redis - 127.0.0.1:6379->6379/tcp
elasticsearch - 127.0.0.1:9200->9200/tcp, 9300/tcp
```

As it can be seen in the output, the configured ports `80`, `443` and `5000` were bound to the host's network interfaces, as indicated by `0.0.0.0:80->80/tcp`, for example. Also as configured in Chapters 5.3.5 and 5.3.2 the Kibana, Elasticsearch and Redis containers bound their `5601`, `9200` and `6379` ports respectively for local access.

With these port configurations, the web services were checked right in the terminal as well:

```
$ curl -kL localhost
<h2>ELK - prod</h2>
<li> <a href="/kibana">Kibana</a>
<li> <a href="/elasticsearch">Elasticsearch</a>
```

```
$ curl -kL localhost:5601
<script>var hashRoute = '/kibana/app/kibana';
var defaultRoute = '/kibana/app/kibana';

var hash = window.location.hash;
if (hash.length) {
  window.location = hashRoute + hash;
} else {
  window.location = defaultRoute;
}</script>
```

```
$ curl -kL localhost:9200
{
  "name" : "prod-node1",
  "cluster_name" : "prod-cluster1",
  "cluster_uuid" : "Wv0MR_wcROKDV3ut36OEFA",
  "version" : {
    "number" : "5.4.0",
    "build_hash" : "780f8c4",
    "build_date" : "2017-04-28T17:43:27.229Z",
    "build_snapshot" : false,
    "lucene_version" : "6.5.0"
  },
  "tagline" : "You Know, for Search"
}
```

Redis was checked by using the `redis-cli` program inside the container with the parameters `info server`:

```
$ docker exec redis redis-cli info server
# Server
redis_version:3.2.9
redis_git_sha1:00000000
redis_git_dirty:0
redis_build_id:a185302eca11aed0
redis_mode:standalone
os:Linux 3.10.0-514.16.1.el7.x86_64 x86_64
arch_bits:64
multiplexing_api:epoll
gcc_version:6.2.1
process_id:1
run_id:e90529b01533f61a267193cf4962f5fbf9eb2a59
tcp_port:6379
uptime_in_seconds:1575
uptime_in_days:0
hz:10
lru_clock:2501064
executable:/data/redis-server
config_file:
```

6.4 Kibana frontend

The web frontend was verified by visiting the production environment at `http://elk-prod.example.com` with a web browser. Figure 6 shows the landing page (index.html) for the environment and it can be seen that the deployment of the Nginx configuration was successful, as signified by the title of the page "ELK - prod".

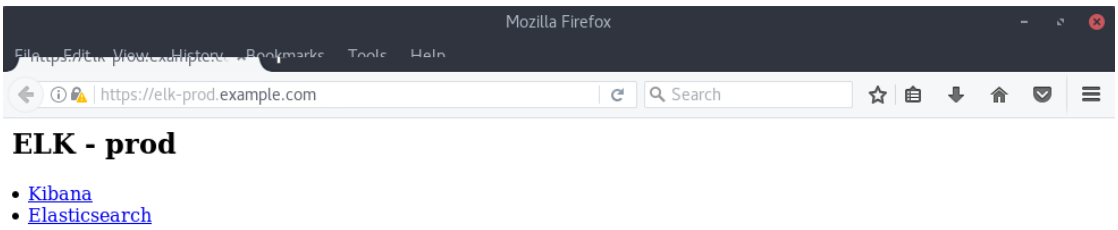

Figure 6: Nginx landing page.

By clicking the link to Kibana an authentication window asking for credentials to the realm ”Kibana” appeared, as seen in Figure 7.

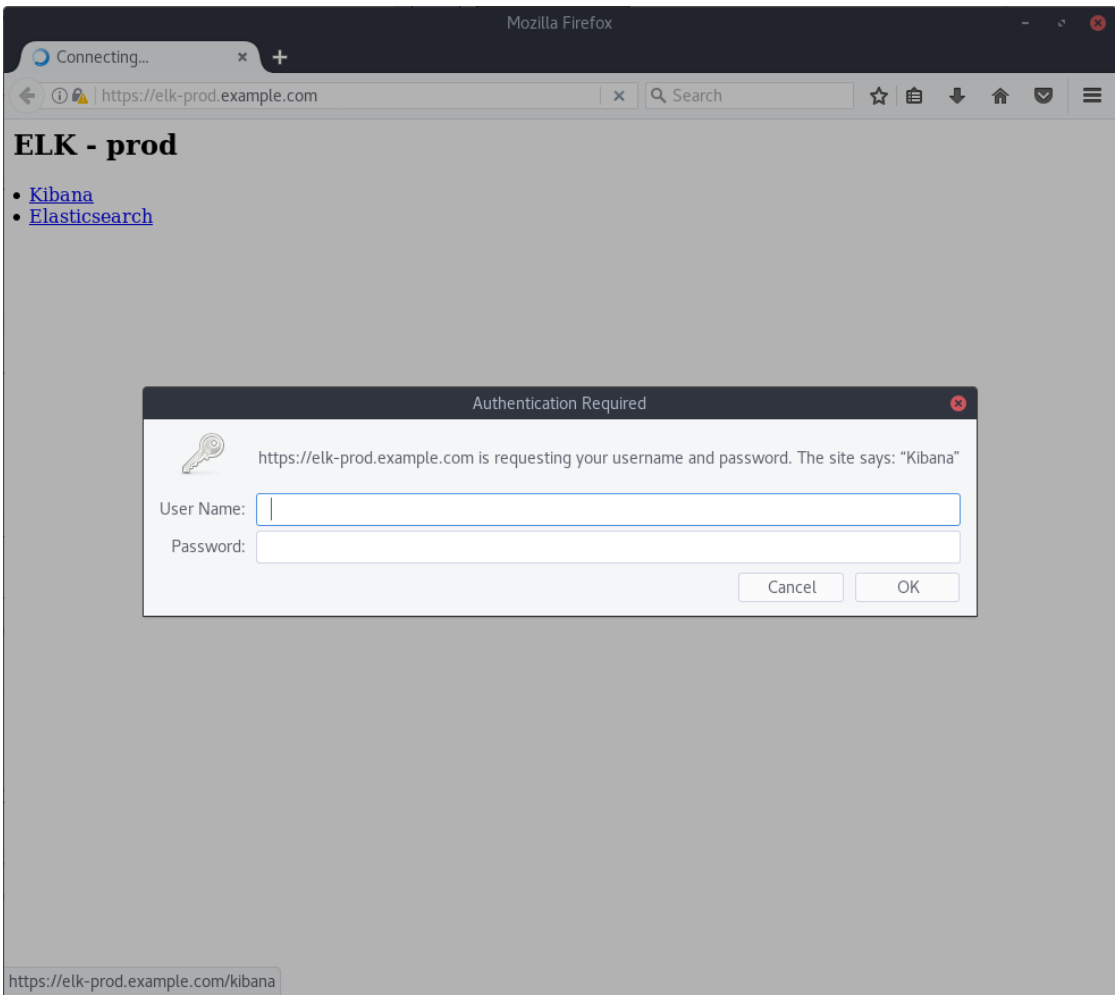

Figure 7: Kibana authentication window.

When the proper credentials were given, for example the username `admin` and the password `adminerino123`, access to the Kibana frontend was granted, verifying that the Kibana reverse proxy was working. Seen in Figure 8.

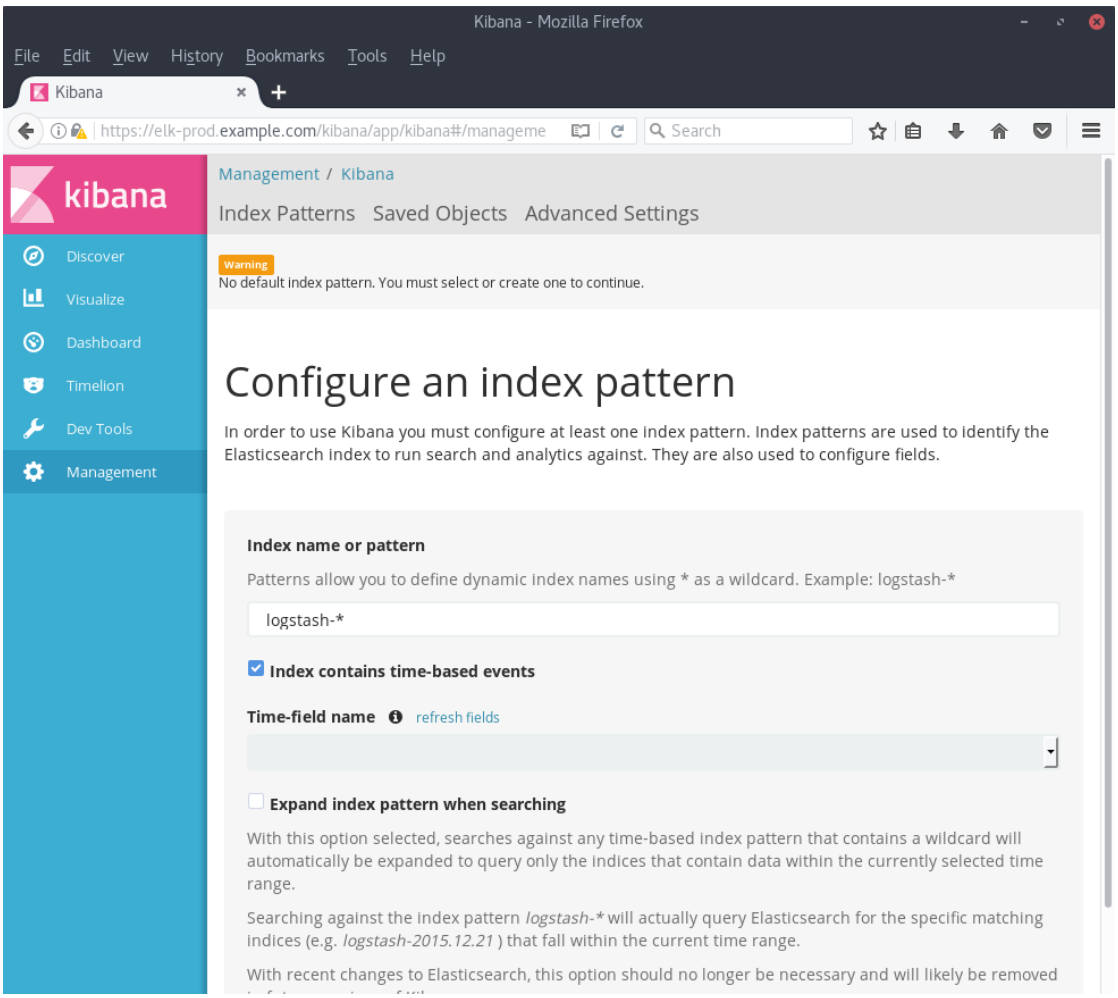

Figure 8: Kibana frontend.

However, to test the authentication properly, wrong credentials were also given when trying to access Kibana. See Figure 9.

Figure 9: Kibana access forbidden for the wrong credentials.

Elasticsearch was accessed by going back to the frontpage and clicking the Elasticsearch link. This verified that the Elasticsearch reverse proxy was working as well, as can be seen in Figure 10.

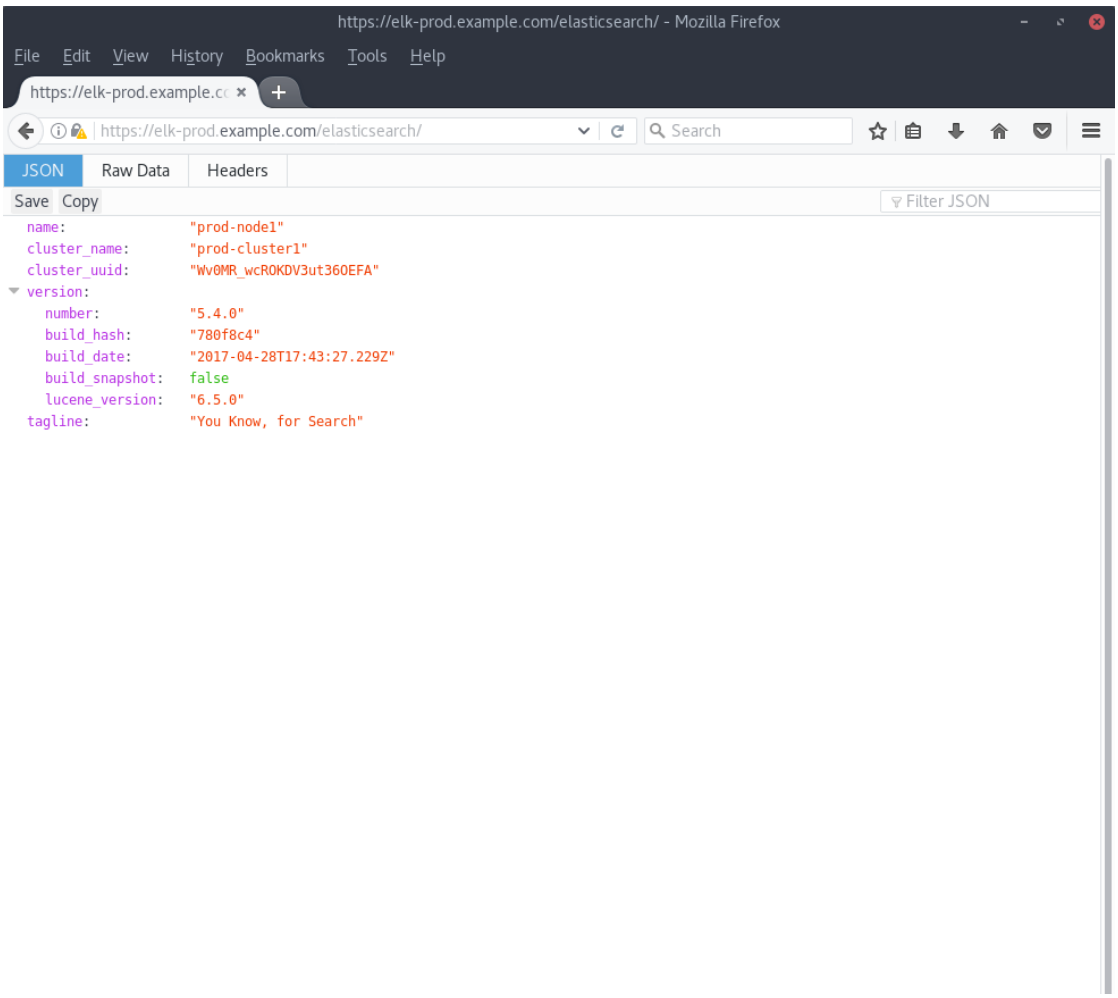

Figure 10: Elasticsearch REST API index.

Elasticsearch's landing page gave basic information about the node accessed, such as the node and cluster names which as seen above are `prod-node1` and `prod-cluster1` respectively, as configured in Chapter 5.2.2.

The proxy allowed direct access to Elasticsearch's REST API and made it possible to do simple queries in the browser, such as viewing the cluster's health as seen in Figure 11.

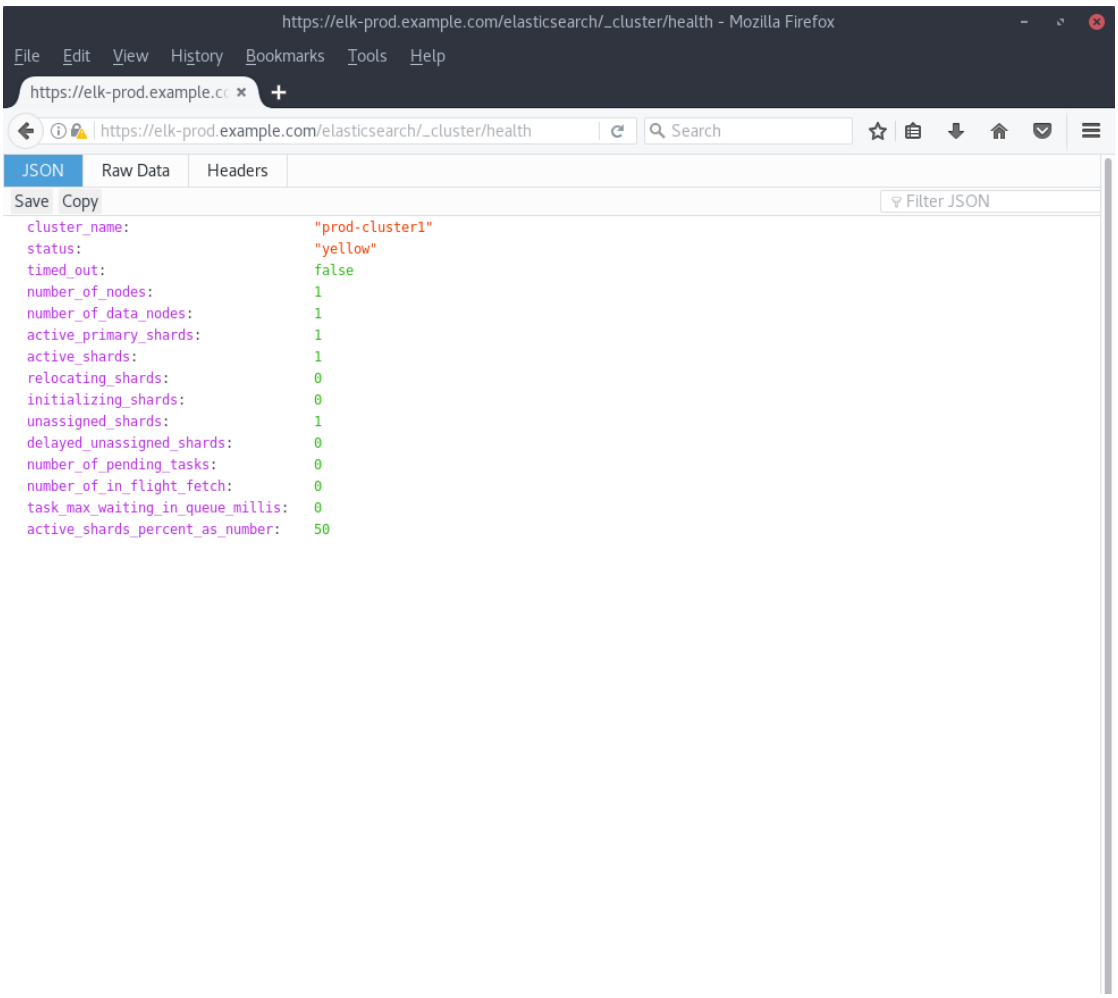

Figure 11: Elasticsearch cluster health.

6.5 Log pipeline

The log shipping and processing pipeline was verified by checking each component separately:

```
$ tail -5 /opt/logstash-forwarder-java/logs/wrapper.log | cut -b 67-
INFO Forwarder - Trying to connect to elk-prod.example.com:5000
INFO LumberjackClient - Connected to elk-prod.example.com:5000
INFO LumberjackClient - Sending 218 events
INFO LumberjackClient - Sending 28 events
INFO LumberjackClient - Sending 481 events
```

Here the log file of logstash-forwarder-java was tailed, and as can be seen it was sending events successfully. `cut` was used here to leave out the timestamp information.

```
$ docker logs --tail=2 ls-cacher | cut -b 50-
[INFO ][logstash.pipeline        ] Pipeline main started
[INFO ][logstash.agent           ] Successfully started Logstash API endpoint
 ↪  {:port=>9600}
```

With the `docker logs` command the log file of the `ls-cacher` Logstash instance was tailed to check if there were any errors. As the output does not have any errors, it was safe to assume the events passed through. Again, `cut` was used to leave out the timestamp.

```
$ docker exec redis redis-cli keys "*"
```

Checking the Redis queue was done by executing `redis-cli` within the container with `docker exec`. Passing the `keys` command with the parameter `"*"` usually makes Redis list all the keys that exist, such as `logstash` in this case, which would mean that they have items in them. As the output for this command was empty, there were no keys present which meant that the queue was empty and the `ls-parser` Logstash instance pulled all the events successfully.

```
$ docker logs --tail=2 ls-parser | cut -b 26-
[INFO ][logstash.pipeline        ] Pipeline main started
[INFO ][logstash.agent           ] Successfully started Logstash API endpoint
 ↪  {:port=>9600}
```

Finally the last part of checking in things in the terminal was done by again running the `docker logs` command, only this time for the `ls-parser` container. Since there were no errors in the output, every event was parsed successfully.

Next to check the Elasticsearch's end, Kibana was configured to use the `prod-logs-*` index pattern to find the parsed events. This pattern comes from the `{{ env }}-logs-%{+YYYY.w}` index pattern configured for Logstash in Chapter 5.2.4. Configuring the index pattern for Kibana is shown in Figure 12.

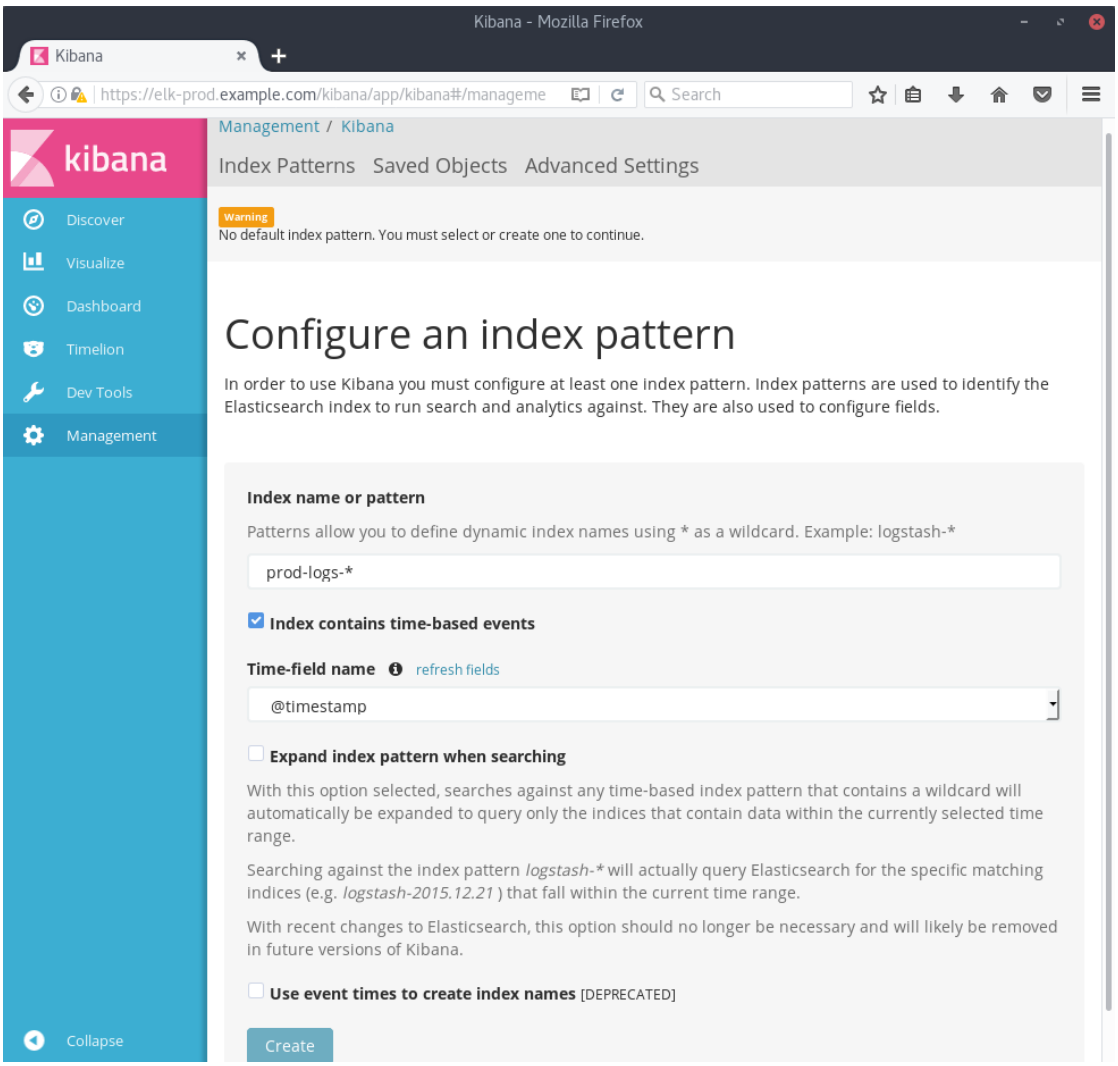

Figure 12: Configuring Kibana index pattern.

After creating the index pattern, Kibana redirected to the overview page showing all the fields and their options found for the pattern. This is show in Figure 13.

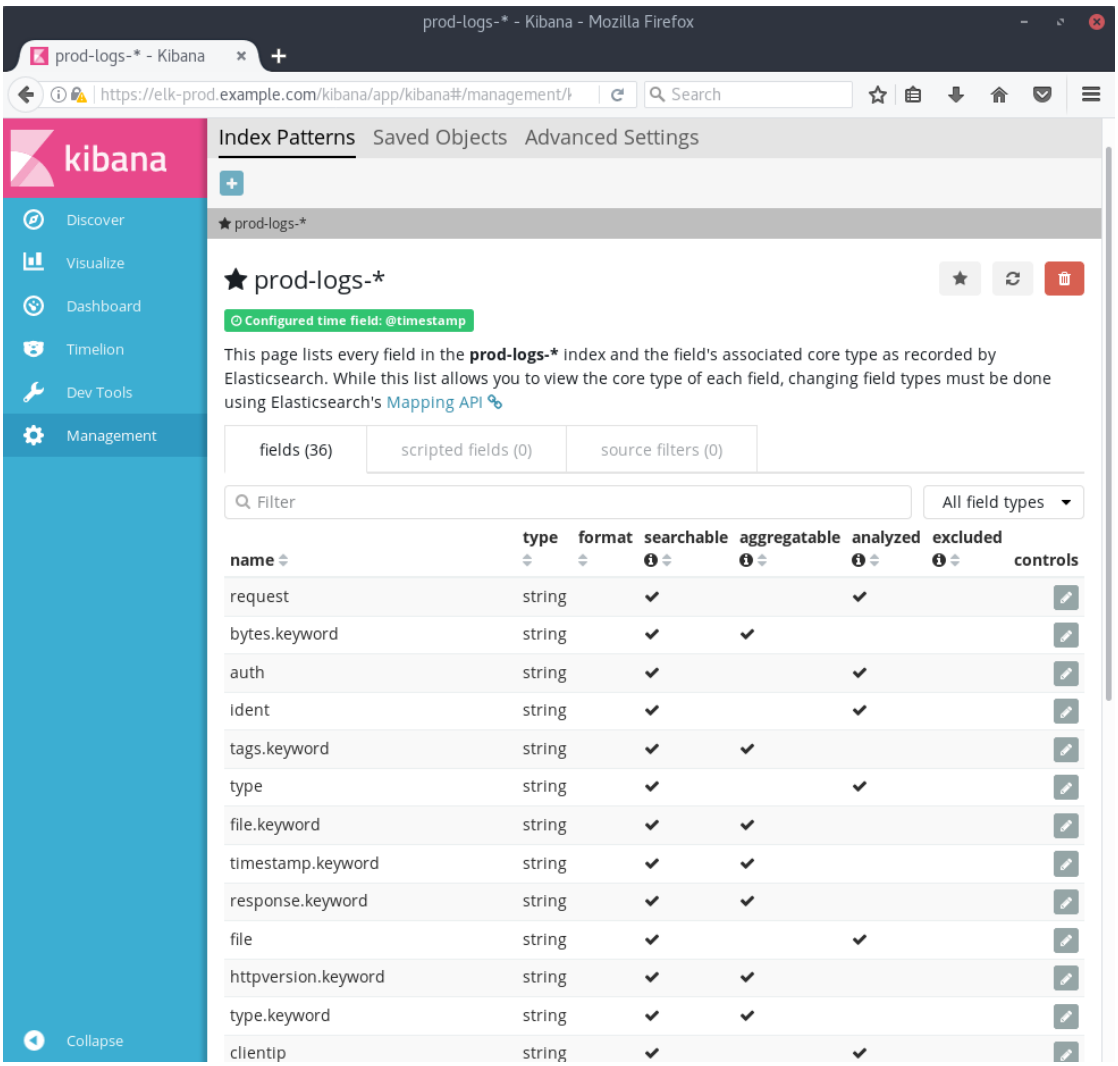

Figure 13: Viewing Kibana index pattern.

Kibana being able to display all the fields shows that the parsed data did indeed exist in Elasticsearch, but the Discovery page was also checked, as show in Figure 14.

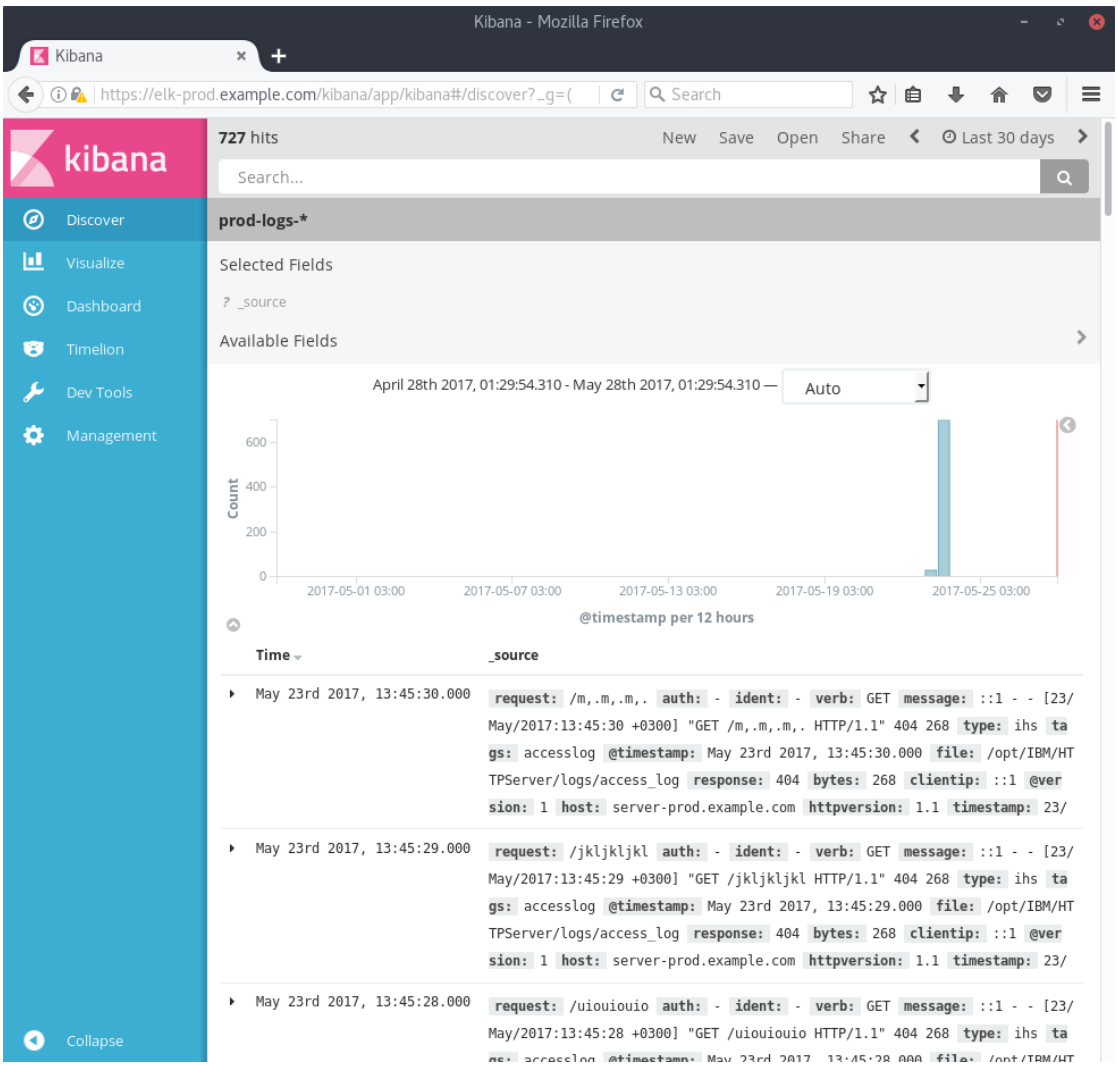

Figure 14: Kibana Discover page.

To test searching, a simple search query was made:

`type: ihs AND bytes: [5000 TO *]` which searched for all the HTTP server documents where the size of the response was over 5000 bytes. This is shown in Figure 15.

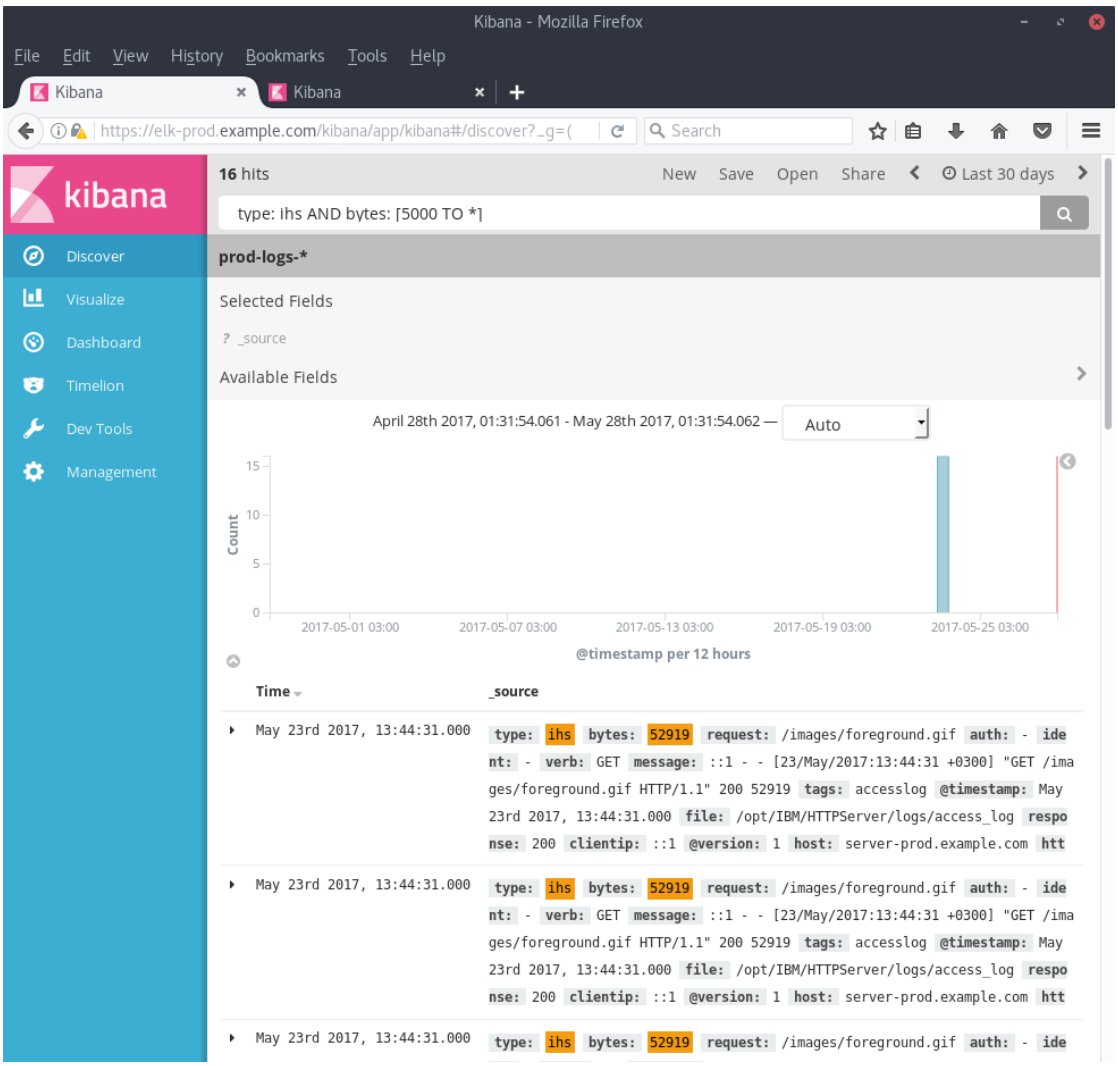

Figure 15: Searching for IHS response size in Kibana.

7 Kibana visualizations

For demonstrating the visualization side of Kibana, three very simple metric visualizations were created to show the number of results for a given query. These visualizations were labelled for easy recognition in the dashboard. The queries were as follows:

- DB2 warning count: `type: db2 AND loglevel: Warning`
- HTTP request count (not images):
  `type: ihs AND NOT request: (*.gif, *.jpg, *.png)`
- Number of WAS events from AppSrv01 SystemOut.log:
  `type: type: was AND tags: (AppSrv01, systemout)`

Creation of the HTTP request count visualization is shown in Figure 16.

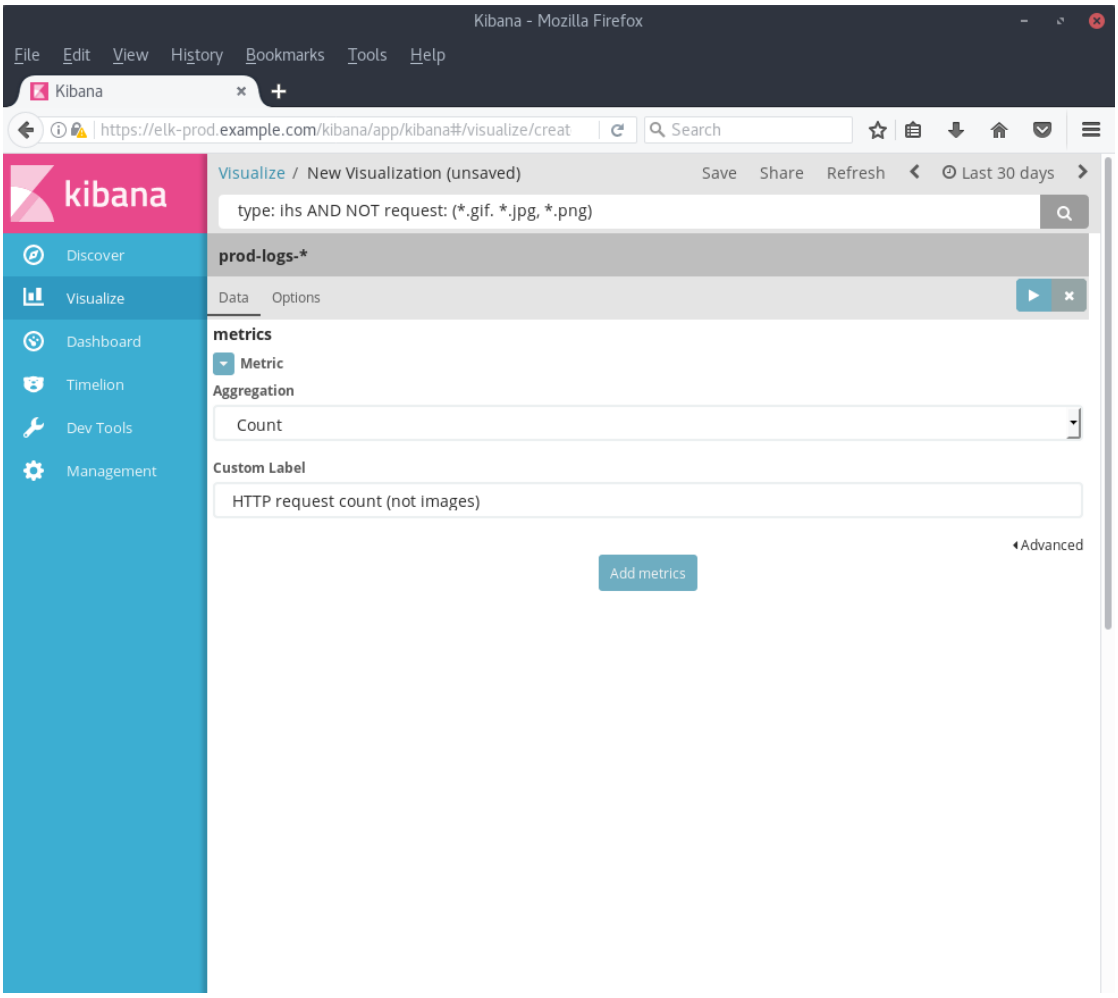

Figure 16: Creating a Kibana visualization for HTTP requests.

The dashboard showing all the three visualizations can be seen in Figure 17.

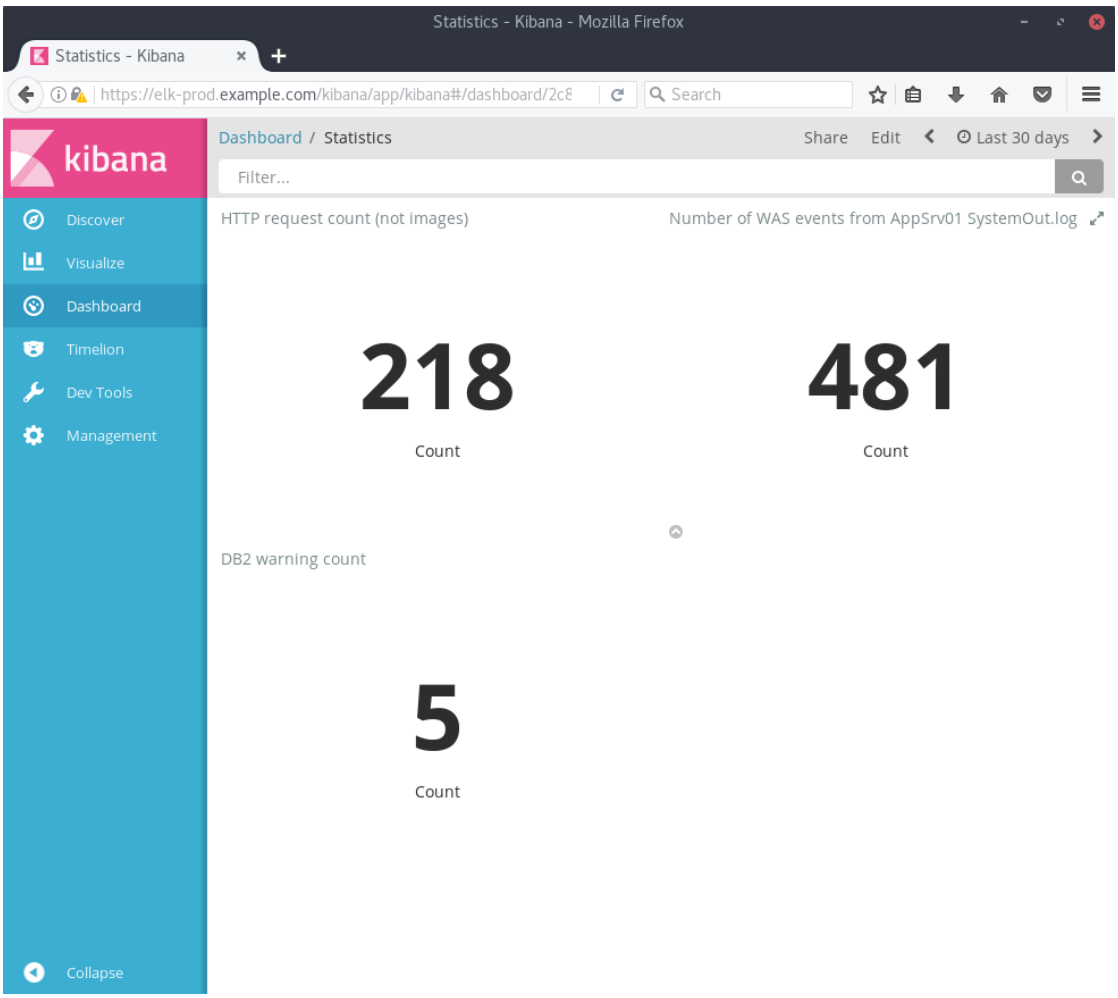

Figure 17: Kibana dashboard for statistics.

8 Discussion

While the result produced in the assignment is working, looking back there are several improvement points that come to mind, e.g. with Ansible, instead of storing the website credentials in plaintext, Ansible Vault could be used. Vault is a tool that allows any file to be encrypted into a text file that can be then put in version control, for example. Ansible recognizes the files encrypted with Vault automatically, so the files would not even need to be decrypted on the disk but would be decrypted by Ansible in runtime.

Another issue that in this implementation could be improved is parametrization, almost every hardcoded value could and should be replaced by a reference to a variable defined somewhere else. This would make the roles more flexible in the sense that the roles themselves would not need to be modified in the event that values such as the Java heap sizes, for example, would need to be increased.

Proper parametrization would also allow for the base role files to be put in a generic version control repository for easy tracking and the variable files would reside elsewhere.

Regarding Elasticsearch's data types, in this implementation none of the number fields were not actually stored as numbers in the index, including fields such as `bytes` from the HTTP server access log parser. While these string fields could be converted to numbers in Logstash's configuration with the `mutate` plugin's `convert` option, an Elasticsearch index templates could be used instead to simplify the process.

In the end the original objectives of the assignment were indeed reached. The Elastic Stack and the deployment process produced were taken into use and improved upon later and the stack is now processing several gigabytes of logs in a day. The project was also a great learning experience of the benefits and quirks of all the components, especially Ansible and Docker and how they can be made to work together.

References

Apache Software Foundation. 2017. 'About the Apache HTTP Server Project'. Accessed 1/1/2017. Retrieved from https://httpd.apache.org/ABOUT_APACHE.html.

Chhajed, Saurabh. 2015. Learning ELK Stack.

didfet. 2016. 'logstash-forwarder-java'. Accessed 27/5/2017. Retrieved from https://github.com/didfet/logstash-forwarder-java/blob/master/README.md.

Docker Inc. 2017. 'What is a Container'. Accessed 27/5/2017. Retrieved from https://www.docker.com/what-container.

Elasticsearch BV. 2017. 'Basic Concepts - Elasticsearch Reference'. Accessed 23/5/2017. Retrieved from https://www.elastic.co/guide/en/elasticsearch/reference/5.4/_basic_concepts.html.

Elasticsearch BV. 2017. 'Discover - Kibana User Guide'. Accessed 26/5/2017. Retrieved from https://www.elastic.co/guide/en/kibana/5.4/discover.html.

Elasticsearch BV. 2017. 'Getting Started - Elasticsearch Reference'. Accessed 23/5/2017. Retrieved from https://www.elastic.co/guide/en/elasticsearch/reference/5.4/getting-started.html.

Elasticsearch BV. 2017. 'Introduction - Kibana User Guide'. Accessed 23/5/2017. Retrieved from https://www.elastic.co/guide/en/kibana/5.4/introduction.html.

Elasticsearch BV. 2017. 'Logstash Introduction - Logstash Reference'. Accessed 23/5/2017. Retrieved from https://www.elastic.co/guide/en/logstash/5.4/introduction.html.

Elasticsearch BV. 2017. 'Logstash Reference'. Accessed 23/5/2017. Retrieved from https://www.elastic.co/guide/en/logstash/5.4/index.html.

Elasticsearch BV. 2017. ‘Virtual memory - Elasticsearch Reference’. Accessed 26/5/2017. Retrieved from https://www.elastic.co/guide/en/elasticsearch/reference/5.4/vm-max-map-count.html.

Ewing, Court. 2016. ‘Kibana 5.0.0-alpha1 released’. Accessed 5/4/2016. Retrieved from https://www.elastic.co/blog/kibana-5-0-0-alpha1.

Government Information Security Management Board. 2009. Lokiohje. Retrieved from https://www.vahtiohje.fi/web/guest/3/2009-lokiohje.

IBM Inc. 2016. ‘Key differences from the Apache HTTP Server’. Accessed 18/12/2016. Retrieved from https://www.ibm.com/support/knowledgecenter/SSEQTJ_8.5.5/com.ibm.websphere.ihs.doc/ihs/cihs_changes.html.

IBM Inc. 2017. ‘IBM DB2 - Database software’. Accessed 27/5/2017. Retrieved from https://www.ibm.com/analytics/us/en/technology/db2/.

IBM Inc. 2017. ‘WebSphere Application Server’. Accessed 27/5/2017. Retrieved from http://www-03.ibm.com/software/products/en/appserv-was.

NGINX Inc. 2017. ‘What is NGINX?’ Accessed 23/5/2017. Retrieved from https://www.nginx.com/resources/glossary/nginx/.

Red Hat Inc. 2017. ‘How Ansible Works’. Accessed 27/5/2017. Retrieved from https://www.ansible.com/how-ansible-works.

Red Hat Inc. 2017. ‘htpasswd - Ansible Documentation’. Accessed 25/5/2017. Retrieved from http://docs.ansible.com/ansible/htpasswd_module.html.

Red Hat Inc. 2017. ‘Intro to Playbooks’. Accessed 27/5/2017. Retrieved from http://docs.ansible.com/ansible/playbooks_intro.html.

Redis. 2017. ‘FAQ - Redis’. Accessed 25/5/2017. Retrieved from https://redis.io/topics/faq.

Redis. 2017. ‘Introduction to Redis’. Accessed 28/5/2017. Retrieved from https://redis.io/topics/introduction.

Sleevi, Ryan. 2017. ‘Support for commonName matching in Certificates (removed)’. Accessed 6/3/2017. Retrieved from `https://www.chromestatus.com/feature/4981025180483584`.

Solteq Oyj. 2016. ‘Strategy and focus areas’. Accessed 23/5/2017. Retrieved from `https://www.solteq.com/en/investors/`.

Appendices

Appendix 1 Ansible project directory structure

```
.
|-- ansible.cfg
|-- deploy_elk_prod.yml
|-- deploy_elk_test.yml
|-- deploy_logshipper_prod.yml
|-- deploy_logshipper_test.yml
|-- files
|   |-- bin
|   |   |-- ibm-java-jre-8.0-4.5-x86_64-archive.bin
|   |   `-- logstash-forwarder-java
|   |       |-- bin
|   |       |   |-- demoapp
|   |       |   |-- testwrapper
|   |       |   |-- wrapper
|   |       |   `-- wrapper.sh
|   |       |-- conf
|   |       |   |-- demoapp.conf
|   |       |   `-- wrapper.conf
|   |       |-- doc
|   |       |   |-- index.html
|   |       |   |-- revisions.txt
|   |       |   `-- wrapper-community-license-1.3.txt
|   |       |-- lib
|   |       |   |-- commons-cli-1.2.jar
|   |       |   |-- commons-io-2.2.jar
|   |       |   |-- commons-lang-2.6.jar
|   |       |   |-- hamcrest-core-1.3.jar
|   |       |   |-- jackson-annotations-2.1.5.jar
|   |       |   |-- jackson-core-2.1.5.jar
|   |       |   |-- jackson-databind-2.1.5.jar
|   |       |   |-- junit-4.11.jar
|   |       |   |-- libwrapper.so
|   |       |   |-- log4j-1.2.17.jar
|   |       |   |-- wrapperdemo.jar
|   |       |   |-- wrapper.jar
|   |       |   `-- wrappertest.jar
|   |       |-- LICENSE.md
|   |       |-- logs
|   |       |   `-- wrapper.log
|   |       |-- logstash-forwarder-java-0.2.4.jar
|   |       |-- README_de.txt
|   |       |-- README_en.txt
|   |       |-- README_es.txt
|   |       |-- README_ja.txt
|   |       |-- README.md
|   |       `-- src
|   |           |-- bin
|   |           |   `-- sh.script.in
|   |           `-- conf
|   |               `-- wrapper.conf.in
|   |-- logstash-forwarder-java
|   |   |-- elk-prod.example.com.jks
|   |   `-- elk-test.example.com.jks
|   |-- ls-cacher
|   |   |-- elk-prod.example.com.crt
|   |   |-- elk-prod.example.com.key
|   |   |-- elk-test.example.com.crt
|   |   `-- elk-test.example.com.key
|   |-- ls-parser
|   |   `-- patterns
|   |       `-- ibm
|   |-- nginx
|   |   |-- elk-prod.example.com.crt
|   |   |-- elk-prod.example.com.key
```

```
|   |   |-- elk-test.example.com.crt
|   |   `-- elk-test.example.com.key
|   `-- template
|       |-- elasticsearch
|       |   `-- elasticsearch.yml.j2
|       |-- kibana
|       |   `-- kibana.yml.j2
|       |-- logstash-forwarder-java
|       |   `-- config.json.j2
|       |-- ls-cacher
|       |   `-- cacher.conf.j2
|       |-- ls-parser
|       |   `-- parser.conf.j2
|       `-- nginx
|           |-- default.conf.j2
|           `-- index.html.j2
|-- group_vars
|   |-- elk_prod.yml
|   |-- elk_test.yml
|   |-- servers_prod.yml
|   `-- servers_test.yml
|-- hosts
`-- roles
    |-- elk_deps
    |   `-- tasks
    |       `-- main.yml
    |-- elk_elasticsearch
    |   |-- handlers
    |   |   `-- main.yml
    |   `-- tasks
    |       `-- main.yml
    |-- elk_kibana
    |   |-- handlers
    |   |   `-- main.yml
    |   `-- tasks
    |       `-- main.yml
    |-- elk_ls
    |   |-- handlers
    |   |   `-- main.yml
    |   `-- tasks
    |       `-- main.yml
    |-- elk_nginx
    |   |-- handlers
    |   |   `-- main.yml
    |   `-- tasks
    |       `-- main.yml
    |-- elk_redis
    |   `-- tasks
    |       `-- main.yml
    `-- logstash-forwarder-java
        |-- handlers
        |   `-- main.yml
        `-- tasks
            `-- main.yml

44 directories, 73 files
```

Appendix 2 Ansible installation and configuration commands

Ansible installation:

```
$ apt update
$ apt show ansible | grep ^Version
Version: 2.0.0.2-2ubuntu1
$ apt install -y python python-pip libssl-dev libffi-dev sshpass
$ pip install ansible
$ ansible --version
ansible 2.3.0.0
```

Ansible and group configurations:

```
$ vim ansible.cfg
$ vim hosts
$ mkdir -p group_vars
$ vim group_vars/elk_prod.yml
$ vim group_vars/elk_test.yml
$ vim group_vars/elk.yml
```

Elasticsearch configuration:

```
$ mkdir -p files/template/elasticsearch
$ vim files/template/elasticsearch/elasticsearch.yml.j2
```

Logstash cacher configuration:

```
$ mkdir -p files/template/ls-cacher
$ vim files/template/ls-cacher/cacher.conf.j2
```

Logstash cacher SSL certificate generation for the production environment:

```
$ cd files/ls-cacher/
$ NAME=elk-prod.example.com
$ openssl req -newkey rsa:4096 -keyout $NAME.key \
    -new -x509 -out $NAME.crt -days 3650 \
    -nodes -subj "/C=FI/CN=$NAME"
```

Logstash cacher SSL certificate generation for the test environment:

```
$ cd files/ls-cacher/
$ NAME=elk-test.example.com
$ openssl req -newkey rsa:4096 -keyout $NAME.key \
    -new -x509 -out $NAME.crt -days 3650 \
    -nodes -subj "/C=FI/CN=$NAME"
```

Logstash cacher Java KeyStore generation for production:

```
$ cd files/
$ NAME=elk-prod.example.com
$ keytool -importcert -trustcacerts \
    -file ls-cacher/$NAME.crt -alias ca \
    -keystore logstash-forwarder-java/$NAME.jks \
    -storepass changeit
```

Logstash cacher Java KeyStore generation for test:

```
$ cd files/
$ NAME=elk-test.example.com
$ keytool -importcert -trustcacerts \
    -file ls-cacher/$NAME.crt -alias ca \
    -keystore logstash-forwarder-java/$NAME.jks \
    -storepass changeit
```

Logstash parser configuration:

```
$ mkdir -p files/ls-parser/patterns files/template/ls-parser
$ vim files/ls-parser/patterns/ibm
$ vim files/template/ls-parser/parser.conf.j2
```

Kibana configuration:

```
$ mkdir -p files/template/kibana
$ vim files/template/kibana/kibana.yml.j2
```

Nginx configuration:

```
$ mkdir -p files/nginx
$ mkdir -p files/template/nginx
$ vim files/template/nginx/default.conf.j2
$ vim files/template/nginx/index.html.j2
```

Nginx SSL certificate generation for the production environment:

```
$ cd files/nginx/
$ NAME=elk-prod.example.com
$ openssl req -newkey rsa:4096 -keyout $NAME.key \
    -new -x509 -out $NAME.crt -days 3650 \
    -nodes -subj "/C=FI/CN=$NAME" \
    -extensions sAN -config <(awk -v name=$NAME \
      '{print $0}
       END{
       print "[sAN]"
       print "subjectAltName=DNS:"name
      }' /etc/pki/tls/openssl.cnf)
```

Nginx SSL certificate generation for the test environment:

```
$ cd files/nginx/
$ NAME=elk-test.example.com
$ openssl req -newkey rsa:4096 -keyout $NAME.key \
    -new -x509 -out $NAME.crt -days 3650 \
    -nodes -subj "/C=FI/CN=$NAME" \
    -extensions sAN -config <(awk -v name=$NAME \
      '{print $0}
       END{
       print "[sAN]"
       print "subjectAltName=DNS:"name
      }' /etc/pki/tls/openssl.cnf)
```

Logstash-forwarder-java preparation:

```
$ mkdir files/bin/logstash-forwarder-java
$ cd files/bin/logstash-forwarder-java
$ wget
 ↪  https://github.com/Sentido-Labs/logstash-forwarder-java/archive/master.zip
$ unzip master.zip
$ cd logstash-forwarder-java-master/
$ apt install unzip maven openjdk-8-jdk
$ mvn package
$ unzip target/logstash-forwarder-java-0.2.4-bin.zip
$ cp -r logstash-forwarder-java-0.2.4/* ../
$ cd ..
$ wget https://wrapper.tanukisoftware.com/download/3.5.32/wrapper-linux-x86-64
 ↪  -3.5.32.tar.gz
$ tar -xzf wrapper-linux-x86-64-3.5.32.tar.gz
$ cp -r wrapper-linux-x86-64-3.5.32/* .
$ rm -r logstash-forwarder-java-master master.zip wrapper-linux-x86-64-3.5.32
 ↪  wrapper-linux-x86-64-3.5.32.tar.gz
$ cp src/conf/wrapper.conf.in conf/wrapper.conf
$ vim wrapper.conf
$ cp src/bin/sh.script.in bin/wrapper.sh
$ vim bin/wrapper.sh
$ perl -pi -e 's/\@app\.(long\.)?name\@/logshipper/' bin/wrapper.sh
$ perl -pi -e 's/^#RUN_AS_USER=/RUN_AS_USER=logshipper/' bin/wrapper.sh
$ mkdir -p files/template/logstash-forwarder-java
$ vim files/template/logstash-forwarder-java/config.json.j2
```

Ansible tasks configuration:

```
$ vim deploy_elk_test.yml
$ vim deploy_elk_prod.yml
$ mkdir -p roles/elk_deps/tasks
$ vim roles/elk_deps/tasks/main.yml
$ mkdir -p roles/elk_elasticsearch/{tasks,handlers}
$ vim roles/elk_elasticsearch/tasks/main.yml
$ vim roles/elk_elasticsearch/handlers/main.yml
$ mkdir -p roles/elk_redis/tasks
$ vim roles/elk_redis/tasks/main.yml
$ mkdir -p roles/elk_ls/{tasks,handlers}
$ vim roles/elk_ls/tasks/main.yml
$ vim roles/elk_ls/handlers/main.yml
$ mkdir -p roles/elk_kibana/{tasks,handlers}
$ vim roles/elk_kibana/tasks/main.yml
$ vim roles/elk_kibana/handlers/main.yml
$ mkdir -p roles/elk_nginx/{tasks,handlers}
$ vim roles/elk_nginx/tasks/main.yml
$ vim roles/elk_nginx/handlers/main.yml
$ mkdir -p roles/logstash-forwarder-java/{tasks,handlers}
$ vim roles/logstash-forwarder-java/tasks/main.yml
$ vim roles/logstash-forwarder-java/handlers/main.yml
$ vim deploy_logshipper_test.yml
$ vim deploy_logshipper_prod.yml
```

Ansible production deployment of Elastic Stack:

```
$ ansible-playbook deploy_elk_prod.yml -u vagrant
```

Ansible test deployment of Elastic Stack:

```
$ ansible-playbook deploy_elk_test.yml -u vagrant
```

Ansible production deployment of logstash-forwarder-java:

```
$ ansible-playbook deploy_logshipper_prod.yml -u vagrant
```

Ansible test deployment of logstash-forwarder-java:

```
$ ansible-playbook deploy_logshipper_test.yml -u vagrant
```

Appendix 3 File ansible.cfg

```
[defaults]
ask_pass = True
host_key_checking = False
inventory = ./hosts
log_path = ./ansible.log
callback_whitelist = timer,profile_task

[privilege_escalation]
become = True
become_ask_pass = True
```

Appendix 4 File hosts

```
[elk_prod]
elk_prod_node1  ansible_host=192.168.1.201  ansible_port=22

[elk_test]
elk_test_node1  ansible_host=192.168.1.202  ansible_port=22

[elk:children]
elk_prod
elk_test

[servers_prod]
server_prod_node1  ansible_host=192.168.1.203  ansible_port=22

[servers_test]
server_test_node1  ansible_host=192.168.1.204  ansible_port=22

[servers:children]
servers_prod
servers_test
```

Appendix 5 Files group_vars/elk_prod.yml and group_vars/elk_test.yml

elk_prod.yml:

```
---

env: prod
http_hostname: elk-{{ env }}.example.com

users:
- user: produser
  pass: pass456
- user: admin
  pass: adminerino456

xpack_settings:
  security: "false"
  monitoring: "false"
  graph: "false"
  watcher: "false"
  reporting: "false"
  ml: "false"

kibana_config:
  users: "{{ users }}"
  server_name: "{{ env }}-kibana1"
  xpack: "{{ xpack_settings }}"

elasticsearch_config:
  users: "{{ users }}"
  cluster_name: "{{ env }}-cluster1"
  node_name: "{{ env }}-node1"
  xpack: "{{ xpack_settings }}"
```

elk_test.yml:

```
---

env: test
http_hostname: elk-{{ env }}.example.com

users:
- user: testuser
  pass: pass123
- user: admin
  pass: admin123

xpack_settings:
  security: "false"
  monitoring: "false"
  graph: "false"
  watcher: "false"
  reporting: "false"
  ml: "false"

kibana_config:
  users: "{{ users }}"
  server_name: "{{ env }}-kibana1"
  xpack: "{{ xpack_settings }}"

elasticsearch_config:
  users: "{{ users }}"
  cluster_name: "{{ env }}-cluster1"
  node_name: "{{ env }}-node1"
  xpack: "{{ xpack_settings }}"
```

Appendix 6 File files/template/elasticsearch/elasticsearch.yml.j2

```
network.host: 0.0.0.0

cluster.name: {{ elasticsearch_config.cluster_name }}
node.name: {{ elasticsearch_config.node_name }}

xpack.security.enabled: {{ elasticsearch_config.xpack.security }}
xpack.monitoring.enabled: {{ elasticsearch_config.xpack.monitoring }}
xpack.ml.enabled: {{ elasticsearch_config.xpack.ml }}
```

Appendix 7 File files/template/ls-cacher/cacher.conf.j2

```
input {
  lumberjack {
    port => 5000
    ssl_certificate => "/ssl/{{ http_hostname }}.crt"
    ssl_key => "/ssl/{{ http_hostname }}.key"
  }
}

output {
  redis {
    host => "redis:6379"
    data_type => "list"
    key => "logstash"
    codec => "json"
  }
}
```

Appendix 8 File files/template/ls-parser/parser.conf.j2

```
input {
  redis {
    host => "redis"
    data_type => "list"
    key => "logstash"
    codec => "json"
  }
}

filter {
  mutate {
    strip => "message"
    remove_field => ["offset", "line"]
    split => { "tags" => "," }
  }

  if ! [message] {
    drop {}
  }

  grok {
    patterns_dir => ["/patterns"]
    match => { "message" => [
      "^%{COMMONAPACHELOG}",
      "(?m)\[%{WAS_DATETIME:timestamp}\]%{SPACE}%{BASE16NUM:thread}%{SPACE}%{WORD:shortname}%{SPACE}%{WORD:loglevel}%{SPACE}%{GREEDYDATA:message2}",
      "(?m)%{DB2DIAG_TIMESTAMP:timestamp}%{SPACE}%{NOTSPACE:id}%{SPACE}LEVEL:%{SPACE}%{DATA:loglevel}%{SPACE}PID%{SPACE}:%{SPACE}%{INT:pid}%{SPACE}TID%{SPACE}:%{SPACE}%{INT:tid}%{SPACE}PROC%{SPACE}:%{SPACE}%{DATA:proc}%{SPACE}INSTANCE%{SPACE}:%{SPACE}%{NOTSPACE:instance}%{SPACE}NODE%{SPACE}:%{SPACE}%{INT:node}%{SPACE}HOSTNAME%{SPACE}:%{SPACE}%{NOTSPACE:hostname}%{SPACE}%{GREEDYDATA:message2}"
    ]}
  }

  if [type] == "db2" {
    ruby {
      code => "event.set('timestamp', event.get('timestamp')[0..-5] + '+%02d00' % (event.get('timezone').to_i / 60))"
    }
    mutate {
      remove_field => [ "timezone" ]
    }
  }

  if [message2] {
    mutate {
      replace => { "message" => "%{message2}" }
      remove_field => [ "message2" ]
    }
  }

  mutate {
    gsub => ["timestamp", "EE(S|D)T", "+0300"]
  }

  date {
    match => ["timestamp",
      "dd/MMM/yyyy:HH:mm:ss Z",
      "M/d/yy HH:mm:ss:SSS z",
      "M.d.yy HH:mm:ss:SSS z",
      "M/d/yy HH:mm:ss:SSS Z",
      "M.d.yy HH:mm:ss:SSS Z",
      "yyyy-MM-dd-HH.mm.ss.SSSSSS Z"
    ]
```

```
59    }
60  }
61
62  output {
63    elasticsearch {
64      hosts => "elasticsearch:9200"
65      index => "{{ env }}-logs-%{+YYYY.w}"
66    }
67
68    if "_grokparsefailure" in [tags] or "_rubyexception" in [tags] or
      ↪  "_dateparsefailure" in [tags] {
69      stdout { codec => rubydebug }
70    }
71  }
```

Appendix 9 File files/template/kibana/kibana.yml.j2

```
server.host: "0.0.0.0"
server.name: "{{ kibana_config.server_name }}"
server.basePath: "/kibana"

elasticsearch.url: "http://elasticsearch:9200"
elasticsearch.requestTimeout: 90000

xpack.security.enabled: {{ kibana_config.xpack.security }}
xpack.monitoring.enabled: {{ kibana_config.xpack.monitoring }}
xpack.graph.enabled: {{ kibana_config.xpack.graph }}
xpack.watcher.enabled: {{ kibana_config.xpack.watcher }}
xpack.reporting.enabled: {{ kibana_config.xpack.reporting }}
xpack.ml.enabled: {{ kibana_config.xpack.ml }}
```

Appendix 10 File files/template/nginx/default.conf.j2

```
server {
  listen  80   default_server;
  server_name  {{ http_hostname }};
  return  301  https://$server_name$request_uri;
}

server {
  listen  443    default_server  ssl  http2;
  server_name    {{ http_hostname }};

  ssl_certificate      /ssl/{{ http_hostname }}.crt;
  ssl_certificate_key  /ssl/{{ http_hostname }}.key;

  location  /  {
    root   /usr/share/nginx/html;
    index  index.html;
  }

  location  ~  ^/(kibana|elasticsearch)$ {
    return  301  https://$server_name$request_uri/;
  }

  location  ~  ^/kibana/.*$ {
    rewrite           /kibana/(.*)  /$1  break;
    proxy_pass        http://kibana:5601;
    proxy_set_header  X-Forwarded-For  $proxy_add_x_forwarded_for;
    proxy_buffering   off;

    auth_basic            "Kibana";
    auth_basic_user_file  /auth/kibana.htpasswd;
  }

  location  ~  ^/elasticsearch/.*$ {
    rewrite           /elasticsearch/(.*)  /$1  break;
    proxy_pass        http://elasticsearch:9200;
    proxy_set_header  X-Forwarded-For  $proxy_add_x_forwarded_for;
    proxy_buffering   off;

    auth_basic            "Elasticsearch";
    auth_basic_user_file  /auth/elasticsearch.htpasswd;
  }
}
```

Appendix 11 File files/template/nginx/index.html.j2

```
<h2>ELK - {{ env }}</h2>
<li> <a href="/kibana">Kibana</a>
<li> <a href="/elasticsearch">Elasticsearch</a>
```

Appendix 12 File files/template/logstash-forwarder-java/config.json.j2

```
{ "network": {
    "servers": ["{{ logstash_address }}:{{ logstash_port }}"],
    "timeout": 30,
    "ssl ca": "/opt/logstash-forwarder-java/{{ logstash_address }}.jks"
  }, "files": [
    { "paths": [ "/opt/IBM/WebSphere/AppServer/profiels/AppSrv01/server1/SystemOut.log" ],
     "fields": { "type": "was",
                 "tags": "AppSrv01,server1,systemout" },
  "multiline": { "pattern": "(^\\[)",
                 "negate": "true", "what": "Previous" } },
    { "paths": [ "/opt/IBM/HTTPServer/logs/access_log" ],
     "fields": { "type": "ihs",
                 "tags": "accesslog" } },
    { "paths": [ "/home/db2inst1/sqllib/db2dump/db2diag*log" ],
     "fields": { "type": "db2",
                 "tags": "diaglog" },
  "multiline": { "pattern": "^(?>\\d\\d){1,2}-(?:0?[1-9]|1[0-9])-",
                 "negate": "true", "what": "Previous" } } ] }
```

Appendix 13 File roles/elk_deps/tasks/main.yml

```
---

- name: Enable the EPEL repository and install Python SELinux support
  yum: name={{ item }}
  with_items:
  - epel-release
  - libselinux-python

- name: Install PIP
  yum: name=python-pip

- name: Install Python docker-py and passlib modules
  pip: name={{ item }}
  with_items:
  - docker-py
  - passlib

- name: Install Docker CE dependencies
  yum: name={{ item }}
  with_items:
  - device-mapper-persistent-data
  - lvm2

- name: Enable Docker CE repository
  get_url:
    url: https://download.docker.com/linux/centos/docker-ce.repo
    dest: /etc/yum.repos.d/docker-ce.repo

- name: Install latest Docker CE
  yum: name=docker-ce

- name: Enable and start docker
  systemd: name=docker state=started enabled=yes daemon_reload=yes
```

Appendix 14 File roles/elk_elasticsearch/tasks/main.yml

```
---

- name: Set mmap count kernel parameter for Elasticsearch
  sysctl:
    name: vm.max_map_count
    value: 262144
    sysctl_set: yes

- name: Ensure Elasticsearch directory paths exist
  file:
    path: "{{ item }}"
    state: directory
    recurse: yes
  with_items:
  - /srv/conf/elasticsearch
  - /srv/data/elasticsearch

- name: Give Elasticsearch data directory uid/gid 1000 ownership
  file:
    path: /srv/data/elasticsearch
    state: directory
    recurse: yes
    owner: 1000
    group: 1000

- name: Render Elasticsearch config
  template:
    src: files/template/elasticsearch/elasticsearch.yml.j2
    dest: /srv/conf/elasticsearch/elasticsearch.yml
  notify: Restart Elasticsearch

- name: Run Elasticsearch container
  docker_container:
    name: elasticsearch
    image: docker.elastic.co/elasticsearch/elasticsearch:5.4.0
    env:
      ES_JAVA_OPTS: "-Xms1g -Xmx1g"
    log_driver: json-file
    log_options:
      max-size: 10m
      max-file: "5"
    ports:
    - 127.0.0.1:9200:9200/tcp
    volumes:
    - /srv/conf/elasticsearch/elasticsearch.yml:/usr/share/elasticsearch/
     ↪  config/elasticsearch.yml:ro
    - /srv/data/elasticsearch:/usr/share/elasticsearch/data:rw
    - /etc/localtime:/etc/localtime:ro
```

Appendix 15 File roles/elk_elasticsearch/handlers/main.yml

```
1  ---
2
3  - name: Restart Elasticsearch
4    docker_container:
5      name: elasticsearch
6      restart: yes
```

Appendix 16 File roles/elk_redis/tasks/main.yml

```
---

- name: Set overcommit memory kernel parameter for Redis
  sysctl:
    name: vm.overcommit_memory
    value: 1
    sysctl_set: yes

- name: Ensure Redis directory paths exist
  file:
    path: /srv/data/redis
    state: directory
    recurse: yes

- name: Run Redis container
  docker_container:
    name: redis
    image: redis:3.2-alpine
    log_driver: json-file
    log_options:
      max-size: 10m
      max-file: "5"
    ports:
    - 127.0.0.1:6379:6379/tcp
    volumes:
    - /srv/data/redis:/data:rw
    - /etc/localtime:/etc/localtime:ro
```

Appendix 17 File roles/elk_ls/tasks/main.yml

```
---

- name: Ensure Logstash directory paths exist
  file:
    path: "{{ item }}"
    state: directory
    recurse: yes
  with_items:
  - /srv/ssl/ls-cacher
  - /srv/conf/ls-cacher
  - /srv/conf/ls-parser
  - /srv/conf/ls-parser-patterns

- name: Render Logstash cacher config
  template:
    src: "{{ item }}"
    dest: "/srv/conf/ls-cacher/{{ item | basename | regex_replace('\\.j2$', '') }}"
  with_fileglob: files/template/ls-cacher/*.conf.j2
  notify: Restart Logstash cacher

- name: Render Logstash parser config
  template:
    src: "{{ item }}"
    dest: "/srv/conf/ls-parser/{{ item | basename | regex_replace('\\.j2$', '') }}"
  with_fileglob: files/template/ls-parser/*.conf.j2
  notify: Restart Logstash parser

- name: Copy Logstash cacher SSL files
  copy: src={{ item }} dest=/srv/ssl/ls-cacher/
  with_items:
  - "files/ls-cacher/{{ http_hostname }}.key"
  - "files/ls-cacher/{{ http_hostname }}.crt"
  notify: Restart Logstash cacher

- name: Copy Logstash parser patterns
  copy: src={{ item }} dest=/srv/conf/ls-parser-patterns/
  with_fileglob:
  - files/ls-parser/patterns/*
  notify: Restart Logstash parser

- name: Run Logstash containers
  docker_container:
    name: "{{ item.name }}"
    image: docker.elastic.co/logstash/logstash:5.4.0
    env:
      LS_JAVA_OPTS: "-Xms256m -Xmx256m"
      XPACK_MONITORING_ENABLED: "false"
    log_driver: json-file
    log_options:
      max-size: 10m
      max-file: "5"
    links: "{{ item.links | default([]) }}"
    ports: "{{ item.ports | default([]) }}"
    volumes: "{{ item.volumes | default(['/etc/localtime:/etc/localtime:ro']) }}"
  with_items:
  - name: ls-cacher
    links:
    - redis
    ports:
    - 0.0.0.0:5000:5000/tcp
    volumes:
    - /srv/conf/ls-cacher:/usr/share/logstash/pipeline:ro
    - /srv/ssl/ls-cacher:/ssl:ro
    - /etc/localtime:/etc/localtime:ro
  - name: ls-parser
    links:
```

```
67      - redis
68      - elasticsearch
69      volumes:
70      - /srv/conf/ls-parser:/usr/share/logstash/pipeline:ro
71      - /srv/conf/ls-parser-patterns:/patterns:ro
72      - /etc/localtime:/etc/localtime:ro
```

Appendix 18 File roles/elk_ls/handlers/main.yml

```
1  ---
2
3  - name: Restart Logstash cacher
4    docker_container:
5      name: ls-cacher
6      restart: yes
7
8  - name: Restart Logstash parser
9    docker_container:
10     name: ls-parser
11     restart: yes
```

Appendix 19 File roles/elk_kibana/tasks/main.yml

```
---

- name: Ensure Kibana directory paths exist
  file:
    path: /srv/conf/kibana
    state: directory
    recurse: yes

- name: Render Kibana config
  template:
    src: files/template/kibana/kibana.yml.j2
    dest: /srv/conf/kibana/kibana.yml
  notify: Restart Kibana

- name: Run Kibana container
  docker_container:
    name: kibana
    image: docker.elastic.co/kibana/kibana:5.4.0
    log_driver: json-file
    log_options:
      max-size: 10m
      max-file: "5"
    links:
    - elasticsearch
    ports:
    - 127.0.0.1:5601:5601/tcp
    volumes:
    - /srv/conf/kibana/kibana.yml:/usr/share/kibana/config/kibana.yml:ro
    - /etc/localtime:/etc/localtime:ro
```

Appendix 20 File roles/elk_kibana/handlers/main.yml

```
---

- name: Restart Kibana
  docker_container:
    name: kibana
    restart: yes
```

Appendix 21 File roles/elk_nginx/tasks/main.yml

```
---

- name: Ensure Nginx directory paths exist
  file:
    path: "{{ item }}"
    state: directory
    recurse: yes
  with_items:
  - /srv/conf/nginx
  - /srv/ssl/nginx
  - /srv/auth/nginx
  - /srv/html

- name: Render Nginx config
  template:
    src: files/template/nginx/default.conf.j2
    dest: /srv/conf/nginx/default.conf
  notify: Restart Nginx

- name: Render Nginx index.html
  template:
    src: files/template/nginx/index.html.j2
    dest: /srv/html/index.html

- name: Copy Nginx SSL files
  copy:
    src: "{{ item }}"
    dest: /srv/ssl/nginx/
  with_fileglob:
  - "files/nginx/{{ http_hostname }}.key"
  - "files/nginx/{{ http_hostname }}.crt"
  notify: Restart Nginx

- name: Create Nginx Kibana htpasswd file
  htpasswd:
    name: "{{ item.user }}"
    password: "{{ item.pass }}"
    path: /srv/auth/nginx/kibana.htpasswd
  with_items: "{{ kibana_config.users }}"
  notify: Restart Nginx

- name: Create Nginx Elasticsearch htpasswd file
  htpasswd:
    name: "{{ item.user }}"
    password: "{{ item.pass }}"
    path: /srv/auth/nginx/elasticsearch.htpasswd
  with_items: "{{ elasticsearch_config.users }}"
  notify: Restart Nginx

- name: Run Nginx container
  docker_container:
    name: nginx
    image: nginx:1.13-alpine
    log_driver: json-file
    log_options:
      max-size: 10m
      max-file: "5"
    links:
    - elasticsearch
    - kibana
    ports:
    - 0.0.0.0:80:80/tcp
    - 0.0.0.0:443:443/tcp
    volumes:
    - /srv/conf/nginx:/etc/nginx/conf.d:ro
    - /srv/ssl/nginx:/ssl:ro
    - /srv/auth/nginx:/auth:ro
    - /srv/html:/usr/share/nginx/html:ro
    - /etc/localtime:/etc/localtime:ro
```

Appendix 22 File roles/elk_nginx/handlers/main.yml

```
1  ---
2
3  - name: Restart Nginx
4    docker_container:
5      name: nginx
6      restart: yes
```

Appendix 23 File roles/logstash-forwarder-java/tasks/main.yml

```
---

- name: Create logshipper user
  user: name=logshipper

- name: Copy logstash-forwarder-java files
  copy:
    src: files/bin/logstash-forwarder-java/
    dest: /opt/logstash-forwarder-java/
    owner: logshipper
    group: logshipper
  notify: Restart logstash-forwarder-java

- name: Make wrapper.sh executable
  file:
    path: /opt/logstash-forwarder-java/bin/wrapper.sh
    mode: "ug+x"

- name: Render config.json
  template:
    src: files/template/logstash-forwarder-java/config.json.j2
    dest: /opt/logstash-forwarder-java/config.json
    owner: logshipper
    group: logshipper
  notify: Restart logstash-forwarder-java

- name: Create empty sincedb
  copy:
    dest: /opt/logstash-forwarder-java/sincedb.json
    content: "[]"
    force: no
    owner: logshipper
    group: logshipper
  notify: Restart logstash-forwarder-java

- name: Copy Java CA KeyStore
  copy:
    src: "files/logstash-forwarder-java/{{ logstash_address }}.jks"
    dest: /opt/logstash-forwarder-java/
    owner: logshipper
    group: logshipper
  notify: Restart logstash-forwarder-java

- name: Check is IBM JRE java binary already exists
  stat: path=/opt/logstash-forwarder-java/jre/bin/java
  register: stat_java
  changed_when: False

- block:
  - name: Copy IBM JRE installer
    copy:
      dest: /opt/logstash-forwarder-java/
      src: files/bin/ibm-java-jre-8.0-4.5-x86_64-archive.bin
      owner: logshipper
      group: logshipper
      mode: "ug+x"

  - name: Create IBM JRE installer response file
    copy:
      dest: /opt/logstash-forwarder-java/ibm_jre.properties
      owner: logshipper
      group: logshipper
      content: |+
        INSTALLER_UI=silent
        LICENSE_ACCEPTED=TRUE
        USER_INSTALL_DIR=/opt/logstash-forwarder-java/
```

```
67
68    - name: Install IBM JRE
69      shell: ./ibm-java-jre-8.0-4.5-x86_64-archive.bin -f ibm_jre.properties
70      args:
71        chdir: /opt/logstash-forwarder-java/
72        creates: /opt/logstash-forwarder-java/jre/bin/java
73      notify: Restart logstash-forwarder-java
74
75    when: not stat_java.stat.exists
```

Appendix 24 File roles/logstash-forwarder-java/handlers/main.yml

```
1  ---
2
3  - name: Restart logstash-forwarder-java
4    shell: /opt/logstash-forwarder-java/bin/wrapper.sh restart
```

Appendix 25 File deploy_elk_prod.yml

```
1  ---
2
3  - hosts: elk_prod
4    roles:
5    - { name: elk_deps,          tags: deps }
6    - { name: elk_elasticsearch, tags: elasticsearch }
7    - { name: elk_redis,         tags: redis }
8    - { name: elk_ls,            tags: logstash }
9    - { name: elk_kibana,        tags: kibana }
10   - { name: elk_nginx,         tags: nginx }
```

Appendix 26 File deploy_elk_test.yml

```
1  ---
2
3  - hosts: elk_test
4    roles:
5    - { name: elk_deps,          tags: deps }
6    - { name: elk_elasticsearch, tags: elasticsearch }
7    - { name: elk_redis,         tags: redis }
8    - { name: elk_ls,            tags: logstash }
9    - { name: elk_kibana,        tags: kibana }
10   - { name: elk_nginx,         tags: nginx }
```

Appendix 27 File deploy_logshipper_prod.yml

```
---

- hosts: servers_prod
  roles:
  - logstash-forwarder-java
```

Appendix 28 File deploy_logshipper_test.yml

```
---

- hosts: servers_test
  roles:
  - logstash-forwarder-java
```

Appendix 29 Ansible output for production environment deployment

The output has been stripped from all the asterisks which were making lines too long.

```
SSH password:
SUDO password[defaults to SSH password]:

PLAY [elk_prod]

TASK [Gathering Facts]
ok: [elk_prod_node1]

TASK [elk_deps : Enable the EPEL repository and install Python SELinux support]
changed: [elk_prod_node1] => (item=[u'epel-release', u'libselinux-python'])

TASK [elk_deps : Install PIP]
changed: [elk_prod_node1]

TASK [elk_deps : Install Python docker-py and passlib modules]
changed: [elk_prod_node1] => (item=docker-py)
changed: [elk_prod_node1] => (item=passlib)

TASK [elk_deps : Install Docker CE dependencies]
ok: [elk_prod_node1] => (item=[u'device-mapper-persistent-data', u'lvm2'])

TASK [elk_deps : Enable Docker CE repository]
changed: [elk_prod_node1]

TASK [elk_deps : Install latest Docker CE]
changed: [elk_prod_node1]

TASK [elk_deps : Enable and start docker]
changed: [elk_prod_node1]

TASK [elk_elasticsearch : Set mmap count kernel parameter for Elasticsearch]
changed: [elk_prod_node1]

TASK [elk_elasticsearch : Ensure Elasticsearch directory paths exist]
changed: [elk_prod_node1] => (item=/srv/conf/elasticsearch)
changed: [elk_prod_node1] => (item=/srv/data/elasticsearch)

TASK [elk_elasticsearch : Give Elasticsearch data directory uid/gid 1000
↪  ownership]
changed: [elk_prod_node1]

TASK [elk_elasticsearch : Render Elasticsearch config]
changed: [elk_prod_node1]

TASK [elk_elasticsearch : Run Elasticsearch container]
changed: [elk_prod_node1]

TASK [elk_redis : Set overcommit memory kernel parameter for Redis]
changed: [elk_prod_node1]

TASK [elk_redis : Ensure Redis directory paths exist]
changed: [elk_prod_node1]

TASK [elk_redis : Run Redis container]
changed: [elk_prod_node1]

TASK [elk_ls : Ensure Logstash directory paths exist]
changed: [elk_prod_node1] => (item=/srv/ssl/ls-cacher)
changed: [elk_prod_node1] => (item=/srv/conf/ls-cacher)
changed: [elk_prod_node1] => (item=/srv/conf/ls-parser)
changed: [elk_prod_node1] => (item=/srv/conf/ls-parser-patterns)
```

```
TASK [elk_ls : Render Logstash cacher config]
changed: [elk_prod_node1] =>
 ↪  (item=/home/ubuntu/elk/files/template/ls-cacher/cacher.conf.j2)

TASK [elk_ls : Render Logstash parser config]
changed: [elk_prod_node1] =>
 ↪  (item=/home/ubuntu/elk/files/template/ls-parser/parser.conf.j2)

TASK [elk_ls : Copy Logstash cacher SSL files]
changed: [elk_prod_node1] => (item=files/ls-cacher/elk-prod.example.com.key)
changed: [elk_prod_node1] => (item=files/ls-cacher/elk-prod.example.com.crt)

TASK [elk_ls : Copy Logstash parser patterns]
changed: [elk_prod_node1] =>
 ↪  (item=/home/ubuntu/elk/files/ls-parser/patterns/ibm)

TASK [elk_ls : Run Logstash containers]
changed: [elk_prod_node1] => (item={u'volumes': [u'/srv/conf/ls-cacher:/usr/sh
 ↪  are/logstash/pipeline:ro', u'/srv/ssl/ls-cacher:/ssl:ro',
 ↪  u'/etc/localtime:/etc/localtime:ro'], u'name': u'ls-cacher', u'links':
 ↪  [u'redis'], u'ports': [u'0.0.0.0:5000:5000/tcp']})
changed: [elk_prod_node1] => (item={u'volumes': [u'/srv/conf/ls-parser:/usr/sh
 ↪  are/logstash/pipeline:ro', u'/srv/conf/ls-parser-patterns:/patterns:ro',
 ↪  u'/etc/localtime:/etc/localtime:ro'], u'name': u'ls-parser', u'links':
 ↪  [u'redis', u'elasticsearch']})

TASK [elk_kibana : Ensure Kibana directory paths exist]
changed: [elk_prod_node1]

TASK [elk_kibana : Render Kibana config]
changed: [elk_prod_node1]

TASK [elk_kibana : Run Kibana container]
changed: [elk_prod_node1]

TASK [elk_nginx : Ensure Nginx directory paths exist]
changed: [elk_prod_node1] => (item=/srv/conf/nginx)
changed: [elk_prod_node1] => (item=/srv/ssl/nginx)
changed: [elk_prod_node1] => (item=/srv/auth/nginx)
changed: [elk_prod_node1] => (item=/srv/html)

TASK [elk_nginx : Render Nginx config]
changed: [elk_prod_node1]

TASK [elk_nginx : Render Nginx index.html]
changed: [elk_prod_node1]

TASK [elk_nginx : Copy Nginx SSL files]
changed: [elk_prod_node1] =>
 ↪  (item=/home/ubuntu/elk/files/nginx/elk-prod.example.com.key)
changed: [elk_prod_node1] =>
 ↪  (item=/home/ubuntu/elk/files/nginx/elk-prod.example.com.crt)

TASK [elk_nginx : Create Nginx Kibana htpasswd file]
changed: [elk_prod_node1] => (item={u'user': u'produser', u'pass': u'pass456'})
changed: [elk_prod_node1] => (item={u'user': u'admin', u'pass':
 ↪  u'adminerino456'})

TASK [elk_nginx : Create Nginx Elasticsearch htpasswd file]
changed: [elk_prod_node1] => (item={u'user': u'produser', u'pass': u'pass456'})
changed: [elk_prod_node1] => (item={u'user': u'admin', u'pass':
 ↪  u'adminerino456'})

TASK [elk_nginx : Run Nginx container]
changed: [elk_prod_node1]

RUNNING HANDLER [elk_elasticsearch : Restart Elasticsearch]
changed: [elk_prod_node1]

RUNNING HANDLER [elk_ls : Restart Logstash cacher]
changed: [elk_prod_node1]
```

```
RUNNING HANDLER [elk_ls : Restart Logstash parser]
changed: [elk_prod_node1]

RUNNING HANDLER [elk_kibana : Restart Kibana]
changed: [elk_prod_node1]

RUNNING HANDLER [elk_nginx : Restart Nginx]
changed: [elk_prod_node1]

PLAY RECAP
elk_prod_node1             : ok=37   changed=35   unreachable=0    failed=0

Playbook run took 0 days, 0 hours, 2 minutes, 57 seconds
```

Appendix 30 Ansible output for test environment deployment

The output has been stripped from all the asterisks which were making lines too long.

```
SSH password:
SUDO password[defaults to SSH password]:

PLAY [elk_test]

TASK [Gathering Facts]
ok: [elk_test_node1]

TASK [elk_deps : Enable the EPEL repository and install Python SELinux support]
changed: [elk_test_node1] => (item=[u'epel-release', u'libselinux-python'])

TASK [elk_deps : Install PIP]
changed: [elk_test_node1]

TASK [elk_deps : Install Python docker-py and passlib modules]
changed: [elk_test_node1] => (item=docker-py)
changed: [elk_test_node1] => (item=passlib)

TASK [elk_deps : Install Docker CE dependencies]
ok: [elk_test_node1] => (item=[u'device-mapper-persistent-data', u'lvm2'])

TASK [elk_deps : Enable Docker CE repository]
changed: [elk_test_node1]

TASK [elk_deps : Install latest Docker CE]
changed: [elk_test_node1]

TASK [elk_deps : Enable and start docker]
changed: [elk_test_node1]

TASK [elk_elasticsearch : Set mmap count kernel parameter for Elasticsearch]
changed: [elk_test_node1]

TASK [elk_elasticsearch : Ensure Elasticsearch directory paths exist]
changed: [elk_test_node1] => (item=/srv/conf/elasticsearch)
changed: [elk_test_node1] => (item=/srv/data/elasticsearch)

TASK [elk_elasticsearch : Give Elasticsearch data directory uid/gid 1000
↪  ownership]
changed: [elk_test_node1]

TASK [elk_elasticsearch : Render Elasticsearch config]
changed: [elk_test_node1]

TASK [elk_elasticsearch : Run Elasticsearch container]
changed: [elk_test_node1]

TASK [elk_redis : Set overcommit memory kernel parameter for Redis]
changed: [elk_test_node1]

TASK [elk_redis : Ensure Redis directory paths exist]
changed: [elk_test_node1]

TASK [elk_redis : Run Redis container]
changed: [elk_test_node1]

TASK [elk_ls : Ensure Logstash directory paths exist]
changed: [elk_test_node1] => (item=/srv/ssl/ls-cacher)
changed: [elk_test_node1] => (item=/srv/conf/ls-cacher)
changed: [elk_test_node1] => (item=/srv/conf/ls-parser)
changed: [elk_test_node1] => (item=/srv/conf/ls-parser-patterns)
```

```
TASK [elk_ls : Render Logstash cacher config]
changed: [elk_test_node1] =>
 ↪  (item=/home/ubuntu/elk/files/template/ls-cacher/cacher.conf.j2)

TASK [elk_ls : Render Logstash parser config]
changed: [elk_test_node1] =>
 ↪  (item=/home/ubuntu/elk/files/template/ls-parser/parser.conf.j2)

TASK [elk_ls : Copy Logstash cacher SSL files]
changed: [elk_test_node1] => (item=files/ls-cacher/elk-test.example.com.key)
changed: [elk_test_node1] => (item=files/ls-cacher/elk-test.example.com.crt)

TASK [elk_ls : Copy Logstash parser patterns]
changed: [elk_test_node1] =>
 ↪  (item=/home/ubuntu/elk/files/ls-parser/patterns/ibm)

TASK [elk_ls : Run Logstash containers]
changed: [elk_test_node1] => (item={u'volumes': [u'/srv/conf/ls-cacher:/usr/sh
 ↪  are/logstash/pipeline:ro', u'/srv/ssl/ls-cacher:/ssl:ro',
 ↪  u'/etc/localtime:/etc/localtime:ro'], u'name': u'ls-cacher', u'links':
 ↪  [u'redis'], u'ports': [u'0.0.0.0:5000:5000/tcp']})
changed: [elk_test_node1] => (item={u'volumes': [u'/srv/conf/ls-parser:/usr/sh
 ↪  are/logstash/pipeline:ro', u'/srv/conf/ls-parser-patterns:/patterns:ro',
 ↪  u'/etc/localtime:/etc/localtime:ro'], u'name': u'ls-parser', u'links':
 ↪  [u'redis', u'elasticsearch']})

TASK [elk_kibana : Ensure Kibana directory paths exist]
changed: [elk_test_node1]

TASK [elk_kibana : Render Kibana config]
changed: [elk_test_node1]

TASK [elk_kibana : Run Kibana container]
changed: [elk_test_node1]

TASK [elk_nginx : Ensure Nginx directory paths exist]
changed: [elk_test_node1] => (item=/srv/conf/nginx)
changed: [elk_test_node1] => (item=/srv/ssl/nginx)
changed: [elk_test_node1] => (item=/srv/auth/nginx)
changed: [elk_test_node1] => (item=/srv/html)

TASK [elk_nginx : Render Nginx config]
changed: [elk_test_node1]

TASK [elk_nginx : Render Nginx index.html]
changed: [elk_test_node1]

TASK [elk_nginx : Copy Nginx SSL files]
changed: [elk_test_node1] =>
 ↪  (item=/home/ubuntu/elk/files/nginx/elk-test.example.com.key)
changed: [elk_test_node1] =>
 ↪  (item=/home/ubuntu/elk/files/nginx/elk-test.example.com.crt)

TASK [elk_nginx : Create Nginx Kibana htpasswd file]
changed: [elk_test_node1] => (item={u'user': u'testuser', u'pass': u'pass123'})
changed: [elk_test_node1] => (item={u'user': u'admin', u'pass': u'admin123'})

TASK [elk_nginx : Create Nginx Elasticsearch htpasswd file]
changed: [elk_test_node1] => (item={u'user': u'testuser', u'pass': u'pass123'})
changed: [elk_test_node1] => (item={u'user': u'admin', u'pass': u'admin123'})

TASK [elk_nginx : Run Nginx container]
changed: [elk_test_node1]

RUNNING HANDLER [elk_elasticsearch : Restart Elasticsearch]
changed: [elk_test_node1]

RUNNING HANDLER [elk_ls : Restart Logstash cacher]
changed: [elk_test_node1]

RUNNING HANDLER [elk_ls : Restart Logstash parser]
```

```
changed: [elk_test_node1]

RUNNING HANDLER [elk_kibana : Restart Kibana]
changed: [elk_test_node1]

RUNNING HANDLER [elk_nginx : Restart Nginx]
changed: [elk_test_node1]

PLAY RECAP
elk_test_node1             : ok=37   changed=35   unreachable=0    failed=0

Playbook run took 0 days, 0 hours, 3 minutes, 9 seconds
```

Appendix 31 Ansible output for production log shipper deployment

The output has been stripped from all the asterisks which were making lines too long.

```
SSH password:
SUDO password[defaults to SSH password]:

PLAY [servers_prod]

TASK [Gathering Facts]
ok: [server_prod_node1]

TASK [logstash-forwarder-java : Create logshipper user]
changed: [server_prod_node1]

TASK [logstash-forwarder-java : Copy logstash-forwarder-java files]
changed: [server_prod_node1]

TASK [logstash-forwarder-java : Make wrapper.sh executable]
changed: [server_prod_node1]

TASK [logstash-forwarder-java : Render config.json]
changed: [server_prod_node1]

TASK [logstash-forwarder-java : Create empty sincedb]
changed: [server_prod_node1]

TASK [logstash-forwarder-java : Copy Java CA KeyStore]
changed: [server_prod_node1]

TASK [logstash-forwarder-java : Check is IBM JRE java binary already exists]
ok: [server_prod_node1]

TASK [logstash-forwarder-java : Copy IBM JRE installer]
changed: [server_prod_node1]

TASK [logstash-forwarder-java : Create IBM JRE installer response file]
changed: [server_prod_node1]

TASK [logstash-forwarder-java : Install IBM JRE]
changed: [server_prod_node1]

RUNNING HANDLER [logstash-forwarder-java : Restart logstash-forwarder-java]
changed: [server_prod_node1]

PLAY RECAP
server_prod_node1          : ok=12   changed=10   unreachable=0    failed=0

Playbook run took 0 days, 0 hours, 0 minutes, 51 seconds
```

Appendix 32 Ansible output for test log shipper deployment

The output has been stripped from all the asterisks which were making lines too long.

```
SSH password:
SUDO password[defaults to SSH password]:

PLAY [servers_test]

TASK [Gathering Facts]
ok: [server_test_node1]

TASK [logstash-forwarder-java : Create logshipper user]
changed: [server_test_node1]

TASK [logstash-forwarder-java : Copy logstash-forwarder-java files]
changed: [server_test_node1]

TASK [logstash-forwarder-java : Make wrapper.sh executable]
changed: [server_test_node1]

TASK [logstash-forwarder-java : Render config.json]
changed: [server_test_node1]

TASK [logstash-forwarder-java : Create empty sincedb]
changed: [server_test_node1]

TASK [logstash-forwarder-java : Copy Java CA KeyStore]
changed: [server_test_node1]

TASK [logstash-forwarder-java : Check is IBM JRE java binary already exists]
ok: [server_test_node1]

TASK [logstash-forwarder-java : Copy IBM JRE installer]
changed: [server_test_node1]

TASK [logstash-forwarder-java : Create IBM JRE installer response file]
changed: [server_test_node1]

TASK [logstash-forwarder-java : Install IBM JRE]
changed: [server_test_node1]

RUNNING HANDLER [logstash-forwarder-java : Restart logstash-forwarder-java]
changed: [server_test_node1]

PLAY RECAP
server_test_node1          : ok=12   changed=10   unreachable=0    failed=0

Playbook run took 0 days, 0 hours, 0 minutes, 52 seconds
```

Appendix 33 Ansible output for production redeployment

The output has been stripped from all the asterisks which were making lines too long.

```
SSH password:
SUDO password[defaults to SSH password]:

PLAY [elk_prod]

TASK [Gathering Facts]
ok: [elk_prod_node1]

TASK [elk_deps : Enable the EPEL repository and install Python SELinux support]
ok: [elk_prod_node1] => (item=[u'epel-release', u'libselinux-python'])

TASK [elk_deps : Install PIP]
ok: [elk_prod_node1]

TASK [elk_deps : Install Python docker-py and passlib modules]
ok: [elk_prod_node1] => (item=docker-py)
ok: [elk_prod_node1] => (item=passlib)

TASK [elk_deps : Install Docker CE dependencies]
ok: [elk_prod_node1] => (item=[u'device-mapper-persistent-data', u'lvm2'])

TASK [elk_deps : Enable Docker CE repository]
ok: [elk_prod_node1]

TASK [elk_deps : Install latest Docker CE]
ok: [elk_prod_node1]

TASK [elk_deps : Enable and start docker]
ok: [elk_prod_node1]

TASK [elk_elasticsearch : Set mmap count kernel parameter for Elasticsearch]
ok: [elk_prod_node1]

TASK [elk_elasticsearch : Ensure Elasticsearch directory paths exist]
ok: [elk_prod_node1] => (item=/srv/conf/elasticsearch)
ok: [elk_prod_node1] => (item=/srv/data/elasticsearch)

TASK [elk_elasticsearch : Give Elasticsearch data directory uid/gid 1000
 ↪  ownership]
ok: [elk_prod_node1]

TASK [elk_elasticsearch : Render Elasticsearch config]
ok: [elk_prod_node1]

TASK [elk_elasticsearch : Run Elasticsearch container]
ok: [elk_prod_node1]

TASK [elk_redis : Set overcommit memory kernel parameter for Redis]
ok: [elk_prod_node1]

TASK [elk_redis : Ensure Redis directory paths exist]
ok: [elk_prod_node1]

TASK [elk_redis : Run Redis container]
ok: [elk_prod_node1]

TASK [elk_ls : Ensure Logstash directory paths exist]
ok: [elk_prod_node1] => (item=/srv/ssl/ls-cacher)
ok: [elk_prod_node1] => (item=/srv/conf/ls-cacher)
ok: [elk_prod_node1] => (item=/srv/conf/ls-parser)
ok: [elk_prod_node1] => (item=/srv/conf/ls-parser-patterns)
```

```
TASK [elk_ls : Render Logstash cacher config]
ok: [elk_prod_node1] =>
 ↪  (item=/home/ubuntu/elk/files/template/ls-cacher/cacher.conf.j2)

TASK [elk_ls : Render Logstash parser config]
ok: [elk_prod_node1] =>
 ↪  (item=/home/ubuntu/elk/files/template/ls-parser/parser.conf.j2)

TASK [elk_ls : Copy Logstash cacher SSL files]
ok: [elk_prod_node1] => (item=files/ls-cacher/elk-prod.example.com.key)
ok: [elk_prod_node1] => (item=files/ls-cacher/elk-prod.example.com.crt)

TASK [elk_ls : Copy Logstash parser patterns]
ok: [elk_prod_node1] => (item=/home/ubuntu/elk/files/ls-parser/patterns/ibm)

TASK [elk_ls : Run Logstash containers]
ok: [elk_prod_node1] => (item={u'volumes': [u'/srv/conf/ls-cacher:/usr/share/l
 ↪  ogstash/pipeline:ro', u'/srv/ssl/ls-cacher:/ssl:ro',
 ↪  u'/etc/localtime:/etc/localtime:ro'], u'name': u'ls-cacher', u'links':
 ↪  [u'redis'], u'ports': [u'0.0.0.0:5000:5000/tcp']})
ok: [elk_prod_node1] => (item={u'volumes': [u'/srv/conf/ls-parser:/usr/share/l
 ↪  ogstash/pipeline:ro', u'/srv/conf/ls-parser-patterns:/patterns:ro',
 ↪  u'/etc/localtime:/etc/localtime:ro'], u'name': u'ls-parser', u'links':
 ↪  [u'redis', u'elasticsearch']})

TASK [elk_kibana : Ensure Kibana directory paths exist]
ok: [elk_prod_node1]

TASK [elk_kibana : Render Kibana config]
ok: [elk_prod_node1]

TASK [elk_kibana : Run Kibana container]
ok: [elk_prod_node1]

TASK [elk_nginx : Ensure Nginx directory paths exist]
ok: [elk_prod_node1] => (item=/srv/conf/nginx)
ok: [elk_prod_node1] => (item=/srv/ssl/nginx)
ok: [elk_prod_node1] => (item=/srv/auth/nginx)
ok: [elk_prod_node1] => (item=/srv/html)

TASK [elk_nginx : Render Nginx config]
ok: [elk_prod_node1]

TASK [elk_nginx : Render Nginx index.html]
ok: [elk_prod_node1]

TASK [elk_nginx : Copy Nginx SSL files]
ok: [elk_prod_node1] =>
 ↪  (item=/home/ubuntu/elk/files/nginx/elk-prod.example.com.key)
ok: [elk_prod_node1] =>
 ↪  (item=/home/ubuntu/elk/files/nginx/elk-prod.example.com.crt)

TASK [elk_nginx : Create Nginx Kibana htpasswd file]
ok: [elk_prod_node1] => (item={u'user': u'produser', u'pass': u'pass456'})
ok: [elk_prod_node1] => (item={u'user': u'admin', u'pass': u'adminerino456'})
changed: [elk_prod_node1] => (item={u'user': u'newuser', u'pass':
 ↪  u'newpass098'})

TASK [elk_nginx : Create Nginx Elasticsearch htpasswd file]
ok: [elk_prod_node1] => (item={u'user': u'produser', u'pass': u'pass456'})
ok: [elk_prod_node1] => (item={u'user': u'admin', u'pass': u'adminerino456'})
changed: [elk_prod_node1] => (item={u'user': u'newuser', u'pass':
 ↪  u'newpass098'})

TASK [elk_nginx : Run Nginx container]
ok: [elk_prod_node1]

RUNNING HANDLER [elk_nginx : Restart Nginx]
changed: [elk_prod_node1]

PLAY RECAP
```

```
elk_prod_node1             : ok=33   changed=3    unreachable=0    failed=0

Playbook run took 0 days, 0 hours, 0 minutes, 19 seconds
```