



Collecting Logs from Docker Containers

Lauri Moilanen

Bachelor's thesis

May 2020

School of Technology

Degree Programme in Information Technology

Author(s) Moilanen, Lauri Ville Juhani	Type of publication Bachelor's thesis	Date May 2020 Language of publication: English
	Number of pages: 55	Permission for web publication: X
Title Collecting logs from Docker containers		
Degree programme Information and Communication Technology		
Supervisor(s) Rintamäki, Marko; Rantala, Ari		
Assigned by Qvantel Oy; Värre, Tommy		
Abstract <p>The thesis was assigned by Qvantel Finland Oy as a part of internal development with the intent to improve the logging solutions provided by Qvantel. The purpose of this project was to develop an automated logging solution that could be used in customer environments and Qvantel internal environments. The project had a heavy focus on the Elastic Stack which was used for the log collection, storage, formatting and visualization.</p> <p>Qvantel had a need for an automated logging solution due to new technologies being taken into use and due to getting new customers. Docker was taken into use company wide so the logging solution would also have to be in containers. Having new multiple customer environments meant that the solution had to be automated.</p> <p>The solution was implemented during this project in an internal testing environment, where it was showcased to stakeholders, and further changes were made based on stakeholder feedback. The installation was fully automated, so the implementation was fast and simple when the development work had been finished.</p> <p>Qvantel approved of the solution, and it was taken into use in customer and internal environments. The solution achieved all its goals and was considered a success. Future development was discussed as well and plans to develop the solution further were made when feedback is received from customer environments and the actual users.</p>		
Keywords/tags logs, Elastic Stack, Ansible, Docker, Mesosphere, visualization, log collection, automation, automated systems, microservices, ELK, Logstash, Elasticsearch, Kibana, Qvantel		
Miscellaneous (Confidential information)		

Description

Tekijä(t) Moilanen, Lauri Ville Juhani	Julkaisun laji Opinnäytetyö, AMK	Päivämäärä Toukokuu 2020
		Julkaisun kieli Englanti
	Sivumäärä: 55	Verkojulkaisulupa myönnetty: x
Työn nimi Lokien kerääminen Docker-konteista		
Tutkinto-ohjelma Insinööri (AMK), tieto- ja viestintätekniikka		
Työn ohjaaja(t) Marko Rintamäki, Ari Rantala		
Toimeksiantaja(t) Qvantel Oy; Värre, Tommy		
<p>Tiivistelmä</p> <p>Opinnäytetyön toimeksiantona oli parantaa lokitusratkaisuja osana Qvantelin sisäistä kehittämistyötä. Projektin tarkoituksena oli kehittää automaattinen lokitusratkaisu, jota voitaisiin käyttää sekä asiakasympäristöissä että Qvantelin sisäisissä ympäristöissä. Projekti keskittyi vahvasti Elastic Stackiin, jota käytettiin lokien keräämiseen, säilyttämiseen, formatointiin ja visualisointiin.</p> <p>Qvantelilla oli tarve automaattiselle lokitusratkaisulle uusien teknologiaratkaisujen ja uusien asiakkuuksien myötä. Docker otettiin käyttöön yhtiön laajuisesti, joka tarkoitti sitä että lokitusratkaisun täytyisi myös olla kontitettu. Useiden eri asiakasympäristöjen takia ratkaisun piti olla automatisoitu.</p> <p>Syntynyt ratkaisu otettiin projektin aikana käyttöön sisäisessä testausympäristössä. Se myös esiteltiin sidosryhmille, joiden antaman palautteen perusteella tehtiin vielä muutoksia. Asennus oli täysin automatisoitu, joten sen toteutus oli helppoa ja nopeaa kehitystyön päätyttyä.</p> <p>Qvantel hyväksyi ratkaisun, ja se otettiin käyttöön sekä yrityksen sisäisissä ympäristöissä että asiakasympäristöissä. Ratkaisu saavutti kaikki tavoitteensa ja oli menestys. Tulevaa kehitystyötä ja ratkaisun jatkokehittämistä aiotaan jatkaa, kun palautetta saadaan asiakasympäristöistä ja käyttäjiltä.</p>		
<p>Avainsanat</p> <p>lokkit, Elastic Stack, Ansible, Docker, Mesosphere, visualisaatio, lokien keräys, automaatio, automatisoidut järjestelmät, mikropalvelut, ELK, Logstash, Elasticsearch, Kibana, Qvantel</p>		
Muut tiedot		

Contents

Terms	5
1 Introduction	6
1.1 Motivation	6
1.2 Assigner	7
1.3 Requirements and goals set by Qvantel.....	7
2 Theory.....	8
2.1 On-prem, IaaS, PaaS & SaaS	8
2.2 Logs.....	10
2.3 IT automation	12
2.4 Docker containers.....	13
2.5 Elastic Stack	14
3 Solution architecture and used technologies	15
3.1 Application orchestration overview	15
3.2 Logging solution overview.....	16
3.3 Libvirt and Virsh.....	17
3.4 Oracle Linux.....	17
3.5 Ansible	18
3.6 Ansible Playbooks.....	19
3.7 Docker.....	20
3.8 Elasticsearch	22
3.9 Filebeat and Auditbeat	24
3.10 Kibana	25
3.11 Logstash.....	25
3.12 Marathon.....	26
3.13 Mesos	26

	2
3.14 Apache ZooKeeper	27
3.15 Consul and Registrator	27
4 Development and automation	28
4.1 Overview.....	28
4.2 Configuration management	29
4.3 Playbook configurations.....	29
4.4 Component configurations.....	32
4.4.1 Auditbeat	33
4.4.2 Filebeat	34
4.4.3 Logstash	35
4.4.4 Elasticsearch	36
4.4.5 Kibana	37
5 Implementation.....	37
5.1 Virtual machine creation	37
5.2 Environment	38
5.3 Log Sources.....	43
5.4 Collection of the logs.....	44
5.5 Testing	45
5.6 Searching logs from Kibana	45
6 Results	46
7 Conclusions	47
References.....	48
Appendices	51
Appendix 1. Truncated log of the Ansible playbook run of orchestration node installation	51
Appendix 2. Truncated log of the application node installation	52

Appendix 3. Truncated log of the log node installation.....	54
---	----

Figures

Figure 1. Example log	11
Figure 2. Docker container builds	13
Figure 3. Container architecture	14
Figure 4. The Elastic Stack	15
Figure 5. Orchestration overview.....	16
Figure 6. Logging system architecture	17
Figure 7. Ansible popularity	19
Figure 8. Example Ansible playbook	20
Figure 9. Docker engine.....	21
Figure 10. Docker demand	22
Figure 11. Elasticsearch popularity	24
Figure 12. Mesos components	27
Figure 13. Orchestration node installation playbook	30
Figure 14. Application node installation playbook	31
Figure 15. Logging node installation playbook	32
Figure 16. Auditbeat Auditd module.....	33
Figure 17. Auditbeat Auditd module continued	34
Figure 18. Auditbeat file_integrity module.....	34
Figure 19. Filebeat Logstash output.....	35
Figure 20. Logstash input	35
Figure 21. Logstash filters	36
Figure 22. Elasticsearch output.....	36
Figure 23. Elasticsearch configuration	37
Figure 24. Kibana configuration	37
Figure 25. List of VMs	38
Figure 26. Ansible installation	38
Figure 27. Sshpass installation issue	38

Figure 28. Sshpass installation workaround	39
Figure 29. Storage allocation.....	39
Figure 30. RAM allocation	39
Figure 31. vCPU allocation	40
Figure 32. Kernel version and operating system version.....	40
Figure 33. Docker containers running on orchestration nodes	41
Figure 34. Orchestrated container "trump".....	41
Figure 35. Mesos-slave running on the node.....	41
Figure 36. No orchestrated containers running.....	41
Figure 37. Logging components running.....	42
Figure 38. Kibana UI and logs	42
Figure 39. Elasticsearch cluster information.....	43
Figure 40. "Trump" container configuration.....	44
Figure 41. Wall mentions	46

Terms

AWS – Amazon Web Services

B2B – Business to Business

B2C – Business to Consumer

BSS – Business Support System

CLI - Command-Line Interface

Elastic Stack - Group of services: Elasticsearch, Logstash, Kibana

EPEL - Extra Packages for Enterprise Linux

GUI – Graphical User Interface

KVM - Kernel-based Virtual Machine

OL – Oracle Linux

RAM – Random Access Memory

RHEL – RedHat Enterprise Linux

SSH - Secure Shell

URL – Uniform Resource Locator

vCPU – Virtual Processing Unit

YAML - YAML Ain't Markup Language

1 Introduction

1.1 Motivation

Without an automated logging solution logs only exist on the machines themselves and can be hard to browse efficiently. The proposed solution had to have centralized log storage and convenient log discovery. Log formatting is important as well when handling different types of data.

Manual deployments are not enough when working in a large scale with multiple different environments. Automated deployments provide extra value compared to manual deployments as they are very easily replicable and multicable. Creating a single automation script once can easily replace hundreds of manual steps down the line, for example in different customer deliveries.

Log events generated by software are often very valuable, as they can contain important data about application behavior and audit events. Log collection is extremely important in multiple industries such as banking for legal reasons. Phones, computers and other electronic devices collect logs to verify events, fix errors and detect security breaches. The information derived from log events is crucial in debugging issues during development in internal environments or during production in customer environments.

Using Docker makes it possible to deploy this solution to most Linux-based environments without any additional configuration. Having the dependencies included in the container images themselves lessens the amount of needed dependencies that must be installed during the software installation.

The Elastic Stack is a very popular collection of open-source software, commonly used in similar projects. The choice to use the Elastic Stack was already made for the author, so no research was done towards other solution as the Elastic Stack provided all the necessary features.

1.2 Assigner

The thesis was assigned by Qvantel Finland Oy during the author's practical training. Qvantel is a privately held, global IT company with over 850 employees with offices in Finland, Sweden, Estonia, India, Italy, Spain and USA, and its headquarters in Helsinki, Finland. Qvantel is listed on London Stock Exchange's 1,000 Companies to Inspire Europe 2017. Qvantel was founded in 1995, with roots in the pre-internet era in Finland. It was founded as Starnet Systems, and it focused on Public Switch Telephone Network (PSTN) billing software for Scandinavian teleoperators. The CEO as of 2 April 2020 is Jean-Marc Lazzari. (Qvantel n.d.)

Qvantel provides its customers a full Business Support System (BSS) stack, which supports their business and makes their life easier via automated fulfillment process and catalogue driven order orchestration. Qvantel provides solutions to both B2C and B2B clients across the world. (Qvantel n.d.)

1.3 Requirements and goals set by Qvantel

The goal for this assignment was to develop a logging solution for Qvantel customer environments and Qvantel internal environments. All the used software components were already in use by Qvantel; hence no research was conducted towards other software solutions. The following features were required from the solution:

1. collect application logs from Docker containers and audit logs from the operating system
2. fully automated installation of the entire solution using Ansible
3. important configuration must be modifiable with environment files or automatically generated

The most important goal was to collect logs from Qvantel applications, which run in Docker containers. Audit logs from the operating system also had to be collected due to security requirements. The extra value comes from less time spent looking for the logs and having redundancy due to the database setup.

The Elastic Stack had to be used due to Qvantel already having it in use in some environments prior to this project and they had decided on its usage across the company. The Elastic Stack met all the requirements that were set for this project, so using it

was not an issue. Having the whole stack in Docker containers meant that the installation was straightforward across multiple different environments and managing them was easier.

The solution would have to be automated, fast and easy to install and easily customizable due to the number of different environments this solution would have to be installed in. Fast and simple configuration was important, as the environments would differ from each other. All the automation was done with Ansible, as it was chosen as the automation software for this project by Qvantel.

Dockerizing components were adopted in Qvantel as a goal that all software solutions should aim at due to their easier management, disconnection from operating systems and other dependencies. The solution would also have to be production ready, and ready to be used in real customer environments as Qvantel already had made agreements about providing this kind of solution to customers.

Lastly, it was required that no confidential information was to be included in this thesis from Qvantel's internal sources or external sources i.e. customers. Confidential information such as personal details, phone numbers and addresses were omitted from the data if they were present.

2 Theory

2.1 On-prem, IaaS, PaaS & SaaS

Different models of cloud computing consist of Infrastructure as a Service, Platform as a Service and Software as a Service. In addition to those, on-premise solutions are

also included. Each of those terms explained in simpler terms can be found from Figure 1.

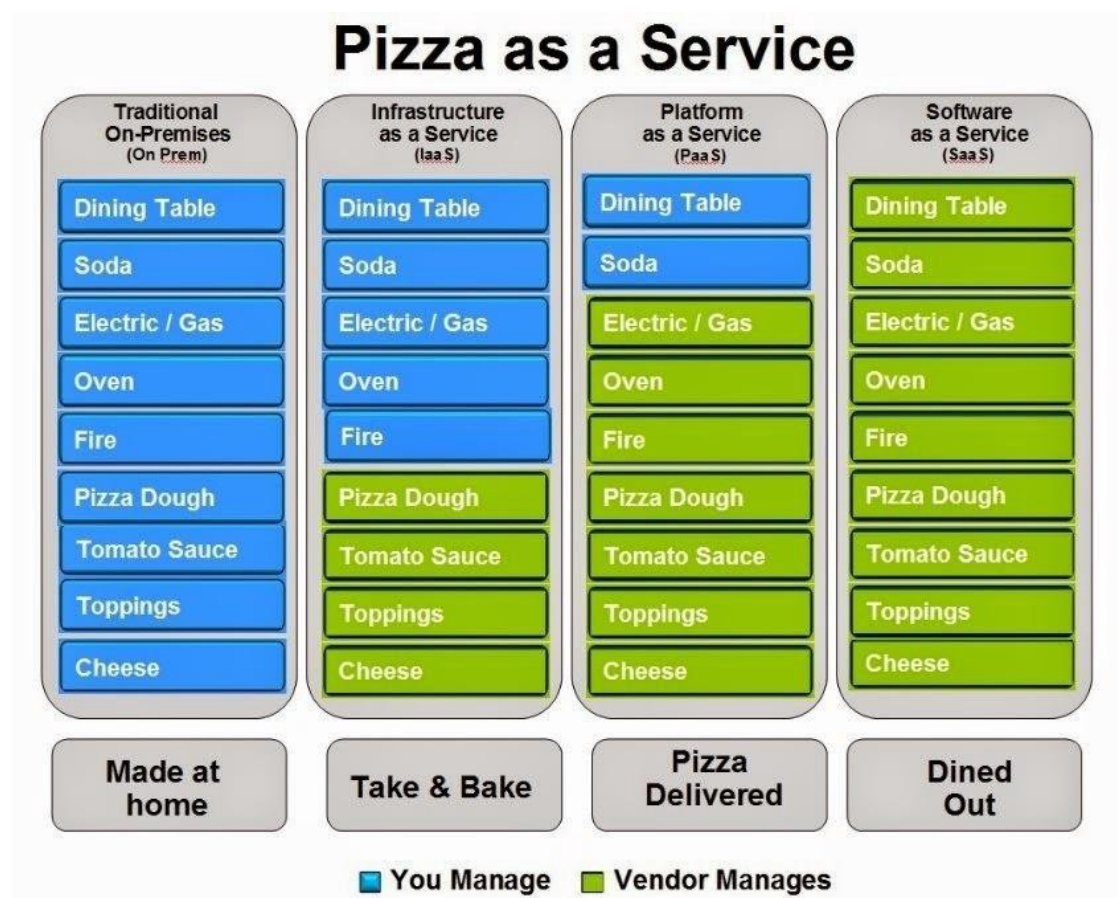


Figure 1. Pizza as a Service (Kerrison 2017)

The on-premises solution does not include any services from outside sources, meaning that everything from hardware to software is managed independently. The expertise and cost required to set up an on-prem solution due to reasons mentioned earlier is very high. The on-premises solution is the least approachable and has the highest requirements to set up. Compared to solutions that offer some of the parts as a service, on-premises solutions are less flexible and scalable.

IaaS (Infrastructure as a Service) offers cloud-based alternatives to on-premise infrastructure. IaaS platforms are flexible and scalable, as well as accessible by multiple users. IaaS platforms can also help cut down costs, as maintaining on-site IT infrastructure can be costly and labor-intensive. (Hou 2020)

Platforms as a Service (PaaS) are built on virtualization technology. PaaS vendors provide hardware and software tools to their users, who are often developers. PaaS platforms cut down a lot of time, as developers do not need to start from scratch every time. PaaS platforms are scalable and do not require extensive system administration knowledge. (Hou 2020)

Software as a Service (SaaS) platforms provide software online, removing the need to install and run applications on the user's computer. SaaS platforms often operate on monthly subscription fees, and usually include maintenance, compliance and security services. Running these supporting services using on-premise software can be time-consuming and expensive. SaaS providers also offer ready-made solutions ranging from simple to more complex, depending on the organization's needs. (Hou 2020)

2.2 Logs

Log events are created by defined actions that take place on the machine. Each log event contains fields and values for the event that took place. Log events document when an event or a change occurred, what occurred, why it occurred and who caused the occurrence. Without appropriate logging, root causes of issues can be harder to track down and fixing the errors can be harder. In most cases, verifying authenticity of information is an important part of a functioning logging system, especially when dealing with sensitive data such as personal information. Logs are retained for predefined time, in some cases due to legal reasons. (Lokien keräys ja käyttö: ohje lokitietojen tallentamiseen ja hyödyntämiseen [Log collection and usage: guide on storing and using logs] 2016, 2-3).

There are multiple different types of logs, with each of them with their own format, use case and need. Logs can be classified into six different categories based on their format, use case and utilization: maintenance log, event log, change log, error log, communication log and owner's log. (Lokien keräys ja käyttö: ohje lokitietojen tallentamiseen ja hyödyntämiseen [Log collection and usage: guide on storing and using logs] 2016, 2-3).

The logs this thesis mainly focuses on are event logs and error logs. The event logs are the most common and most needed type of log. Event logs gather information

about normal events that occur in the system, such as logins and logouts. The error logs provide information related to defects or issues that might occur in the system. They are crucial in finding and debugging issues faster and more reliably. (Näin keräät ja käytät lokitietoja [How to collect and use logs]).

The handling of the logs must be based on law. The laws regulate the contents of the logs, the amount of time they are stored, the integrity of the contents, as well as how the logs are used. The different log types have different sets of requirements that they must follow. (Näin keräät ja käytät lokitietoja [How to collect and use logs]).

Figure 2 shows see an example log file of Crontab being edited by root. The log event starts with the timestamp that records the time the event occurred, then information about what process was called, next about who made the call and finally a description of the event so all the necessary information is present.

```
Sep 11 09:46:33 sys1 crontab[20601]: (root) BEGIN EDIT (root)
Sep 11 09:46:39 sys1 crontab[20601]: (root) REPLACE (root)
Sep 11 09:46:39 sys1 crontab[20601]: (root) END EDIT (root)
```

Figure 2. Example log (What are Linux Logs? Code Examples, Tutorials & More 2017)

These types of logs can be cross-referenced with events from other sources, such as applications and draw conclusions based on those. For example, if there are any issues with the software that is being executed from the crontab file, this log event provides extremely beneficial information.

It is possible to create visualizations out of these types of logs to make monitoring the system more convenient. A simple graph showing the number of logs over a predetermined amount of time can show critical errors in the system, for example if the number of logs suddenly doubles or the logs stop coming to the system completely.

The most common and easiest way to store logs is to save them into a file. Most logs are written to log files by default; hence, no additional setup is needed. Saving the logs into a database can provide additional security, as providing redundancy can be important in some cases.

2.3 IT automation

IT automation can be defined as use of software to create repeatable instructions and processes. Automation software works within the confines of those predetermined instructions, reducing the amount of human interaction necessary or even replacing it completely. IT automation is considered vital in the modern digital transformation. (What's IT automation? N.d.)

IT automation can be a powerful tool for scaling a business and cutting costs significantly. Automation also allows the IT staff to focus more on the strategic work. Benefits of automation include savings on both cost and time required to manage the operations. (What is IT Automation? N.d.)

However, automation also has its own challenges and risks. Although it can cut down the costs in the long run, it can also require a significant upfront investment in the purchase of the software necessary and in time to get the automation up and running. It is therefore important to evaluate which tasks are worth automating and which are not. It is also crucial to note that while automation does lower the risk of human error, the speed and power of automation also means that any error occurring in the automated system can cause more significant damage. Consequently, the importance of proper setup and testing increases. Lastly, automated processes tend to be built for a specific, narrow purpose, which makes them very inflexible. (What is IT Automation? N.d.)

At least some level of automation can be applied to most IT tasks. Different use cases include network automation, infrastructure, cloud provisioning and standard operating environments (SOEs), as well as containers, methodologies, security, testing and monitoring. A fully efficient set of automated, essential tasks and capabilities is critical in successful application deployments, especially during the testing phases. Automation helps the user to move from commit and build to testing and deployment faster and more reliably. This improves efficiency and also reduces the chance of human errors. (What's IT automation? N.d.)

2.4 Docker containers

According to Docker (What is a container? N.d.),

a container is a standard unit of software that packages up code and all its dependencies, so the application runs quickly and reliably from one computing environment to another. A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings. Container images become containers at runtime and in the case of Docker containers - images become containers when they run on Docker Engine.

Flow of Docker container builds can be seen in Figure 3.

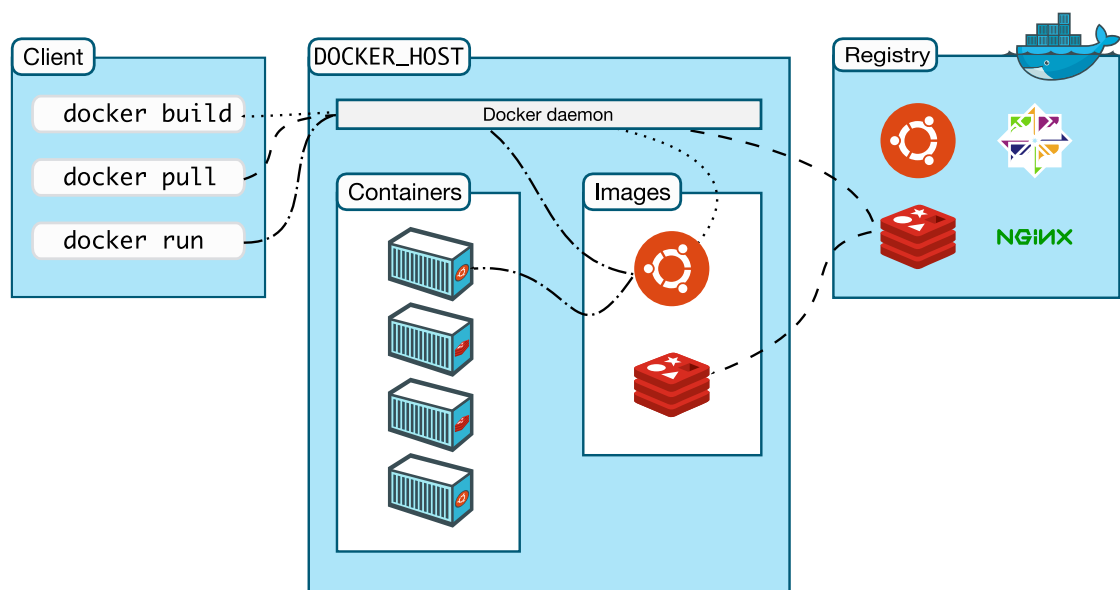


Figure 3. Docker container builds (Docker overview, n.d.)

The container architecture can be seen in Figure 4.

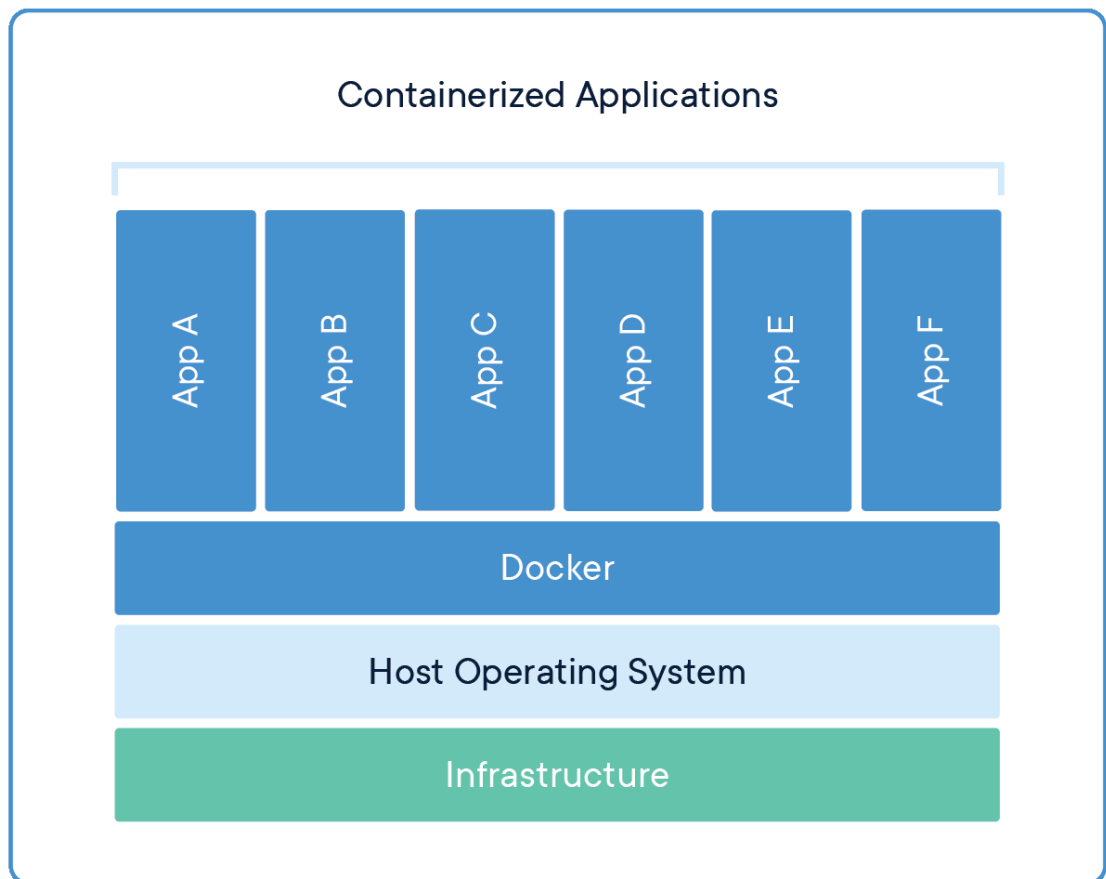


Figure 4. Container architecture (What is a container? N.d.)

2.5 Elastic Stack

Elastic Stack is a software stack provided by Elastic. In this project, it is used for log collection, storage, formatting, filtering and visualization. Filebeat and Auditbeat are the specific Beats used in this project, although other Beats are also available for tasks such as collecting metric data and network data. The structure of Elastic Stack can be seen in Figure 5.

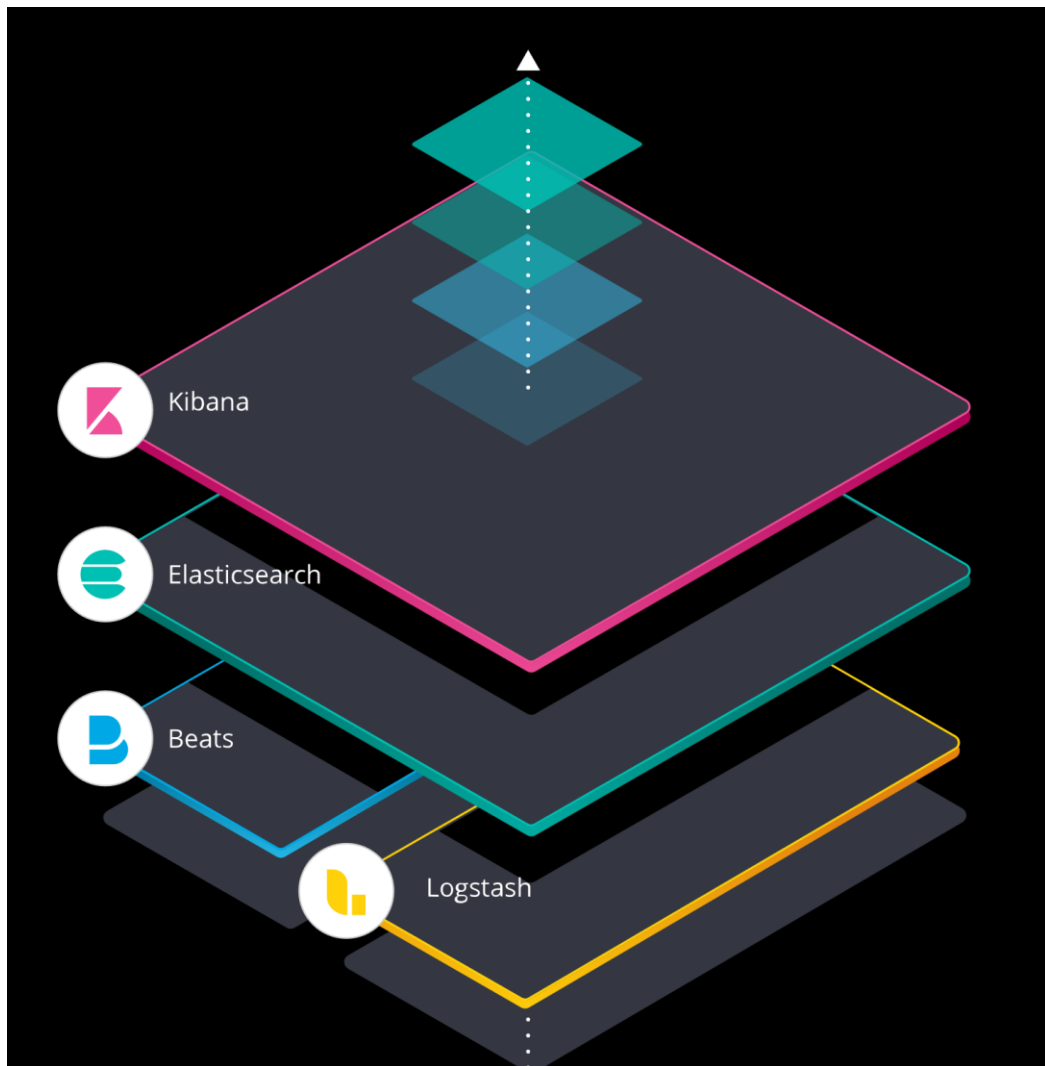


Figure 5. The Elastic Stack (Elastic Stack: Elasticsearch, Logstash, Kibana. n.d.)

3 Solution architecture and used technologies

3.1 Application orchestration overview

The solution deployed to orchestrate the application which would generate the logs is described in Figure 6. The user deploys an application to Marathon via an API or the GUI, which then sends a message to the Mesos master to start the application with the specifications that the user has defined. Mesos master then checks the mesos-slave nodes for available resources and deploys the application to a node with the available resources. The application is then registered to Consul via Registrator, and it can be reached with a hostname instead of the IP address of the node. The

hostname of each application is `{application_name}.service.consul`. The specific components and their setups are described in more detail in the next chapters.

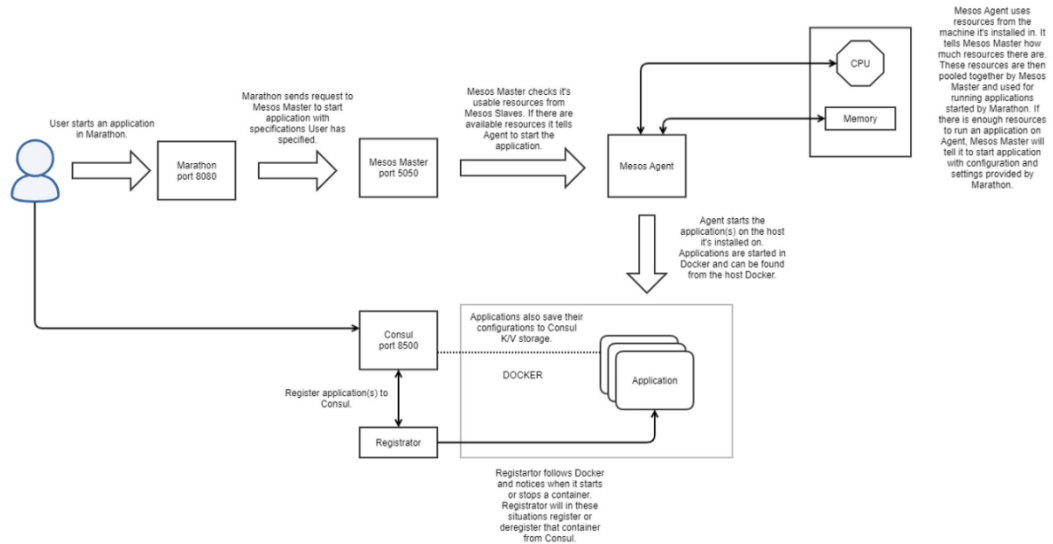


Figure 6. Orchestration overview

3.2 Logging solution overview

Figure 7. describes the whole pipeline that a single log event goes through is described. The log event can come from application, or from an operating system event. Auditbeat writes the audit logs into a file, which Filebeat then sends to Logstash for filtering and formatting. The applications running on top of the orchestration automatically write the logs into files, which Filebeat is prospecting. The logs are then sent to Logstash for application specific filtering and formatting, after which Logstash then sends the logs to Elasticsearch and checks the tags to specify the index where they should be sent to. The logs can then be viewed from Kibana, which is the GUI for Elasticsearch.

Using the Elastic Stack was set as a requirement to this project; hence it was used without doing any research towards other solutions that could have been used. Having the Elastic Stack in containers was also a requirement; hence they were set up in

containers. The official Elastic open-source images are used for this solution. The specific components and their setups are described in more detail in the next chapters.

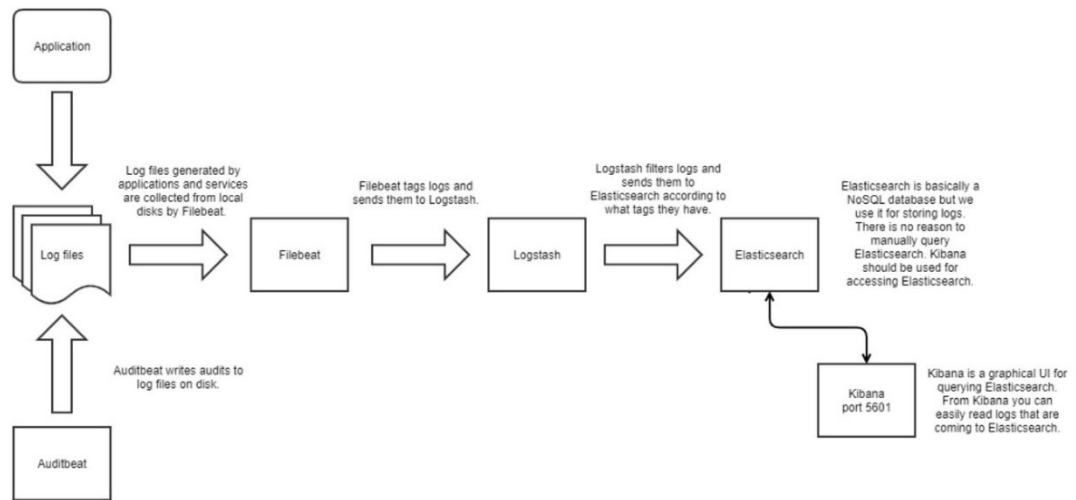


Figure 7. Logging system architecture

3.3 Libvirt and Virsh

Libvirt is collection of software for managing virtual machines and other virtualization functionality. These software pieces include a long-term stable C API, a daemon (libvirtd), and a command line utility (virsh). Libvirt is free software available under the GNU Lesser General Public License. (Libvirt n.d.)

Virsh guest domains are virtual machines managed by the main interface of Virsh program. It can be used to create, pause, and shutdown domains, as well as to list current domains. Virsh(1): Management User Interface n.d.)

In this project, Virsh is used as the management tool for the virtual machines. The virtual machines used for developing and testing this project are created and managed with Virsh. The ease of recreating the virtual machines allowed for simple testing of the deployments.

3.4 Oracle Linux

Oracle Linux is an enterprise-class Linux distribution supported by Oracle and built from source packages for Red Hat Enterprise Linux (RHEL). Oracle Linux includes a

Linux kernel called "Oracle Unbreakable Kernel", as well as a tight integration with Oracle's hardware and software products including most database applications. The "zero downtime patching" feature enables administrators to update the kernel without a reboot. (Distribution Release: Oracle Linux 7.5 2018)

Oracle Linux was chosen as the operating system running on the virtual machines, as it is very commonly used in internal and customer environments. Any other RedHat-based Linux could be used as well, such as RHEL, CentOS, or Amazon Linux AMI.

3.5 Ansible

Ansible is an IT automation engine used for automation, cloud provisioning, configuration management, application deployment among other things. There are no limitations from Ansible regarding what can be automated. The engine connects to the user's nodes and then pushes out small programs, called "Ansible modules", to them. These Ansible modules are written to be resource models of the desired state of the system. The engine then executes these modules over SSH by default and removes them when finished. The library of modules can reside on any machine. The user will typically work with the terminal program of their choice, a text editor, and a version control system for tracking the changes to their content. A compatible Ansible version is required on the machine. (How Ansible Works n.d.).

Ansible was chosen as the automation tool for this project as it was already in use at Qvantel and it was found suitable for this project. Ansible is a very popular tool among developers as seen in Figure 8. All the configuration and customization to the components are carried out in the Ansible playbooks and configuration files, so the whole deployment is automated.

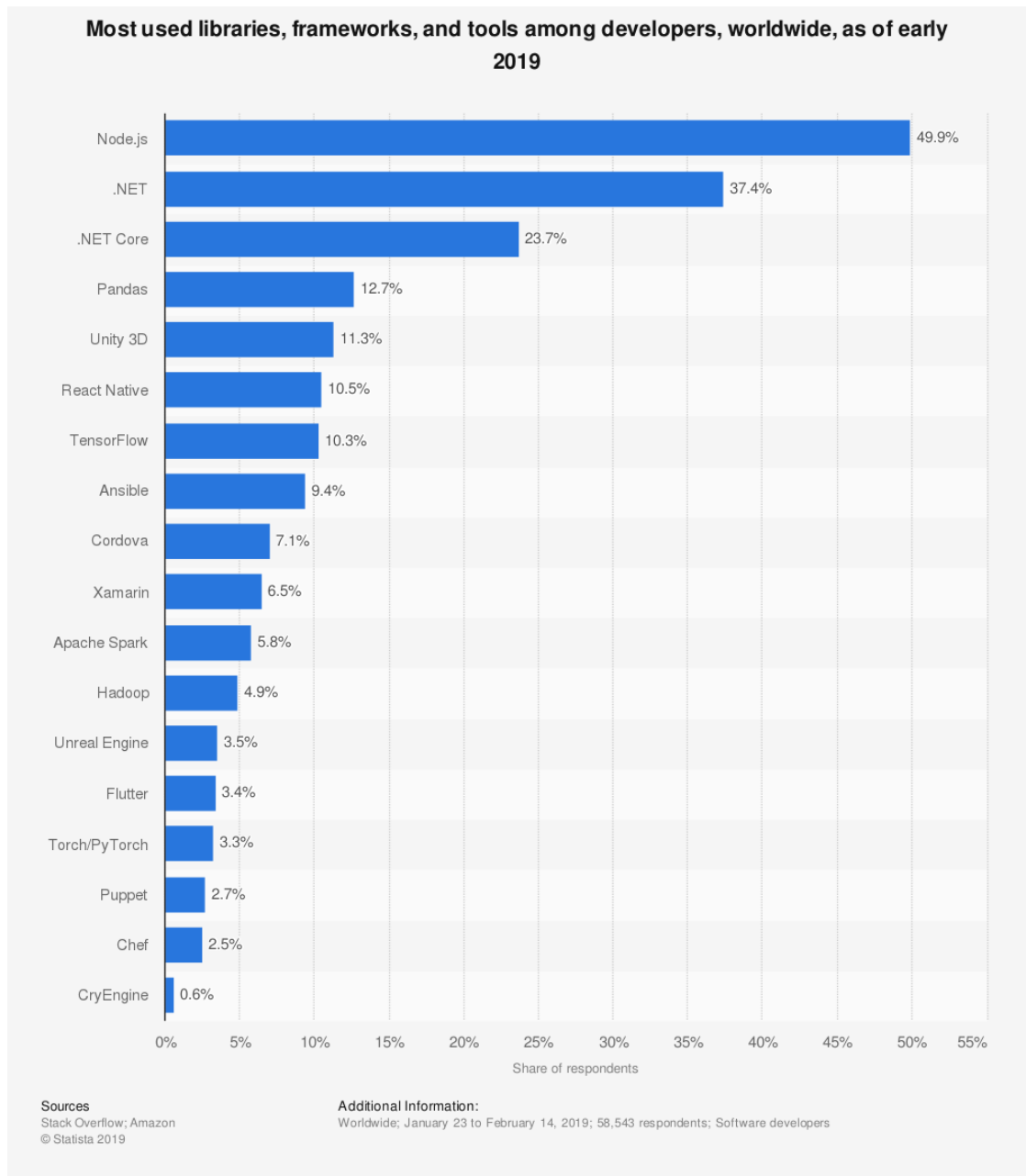


Figure 8. Ansible popularity (Liu 2019c)

3.6 Ansible Playbooks

All the deployments are carried out with Ansible playbooks. A collection of playbooks is used to deploy specific nodes. Each component has its own installation playbook, and those playbooks are then included into a larger playbook, which is used to install the node itself.

Figure 9 displays what a playbook can look like; in this example Dnsmasq is installed and some configuration is done, such as adding node.consul to search domains. All the variables are pulled from environment files, where they can be defined and managed without modifying the playbooks themselves.

```

---
- name: Install dnsmasq
  yum:
    name: dnsmasq
    state: latest
    tags: install
    notify: restart dnsmasq

- name: Apply Consul configuration to dnsmasq
  template:
    src: templates/10-consul.j2
    dest: /etc/dnsmasq.d/10-consul
    notify: restart dnsmasq
    tags: configure

- name: Add Consul to resolv.conf
  lineinfile:
    dest: /etc/resolv.conf
    state: present
    insertbefore: "nameserver {{ ansible_dns['nameservers'][0] | default('') }}"
    regexp: '^nameserver {{ dnsmasq_listen_address }}$'
    line: 'nameserver {{ dnsmasq_listen_address }}'
    tags: configure

- name: Add node.consul to search domains
  lineinfile:
    dest: /etc/resolv.conf
    regexp: '^search.*$'
    line: "search {{ ansible_dns['search'] | default([]) | union(['node.consul']) | unique | join(' ') }}"
    state: present
    tags: configure

- name: Start and enable Dnsmasq
  systemd:
    name: dnsmasq
    state: started
    enabled: yes

```

Figure 9. Example Ansible playbook

3.7 Docker

Docker is used to create, deploy, and run applications by using containers. These containers allow a developer to package up an application with all the necessary parts, for example libraries and other dependencies, and ship them all out as one package. By using the container, the application will be able to run on any other Linux machine that has Docker installed, regardless of any customized setting differences between users. Docker engine structure is described at a high level in Figure 10. (What is Docker? N.d)

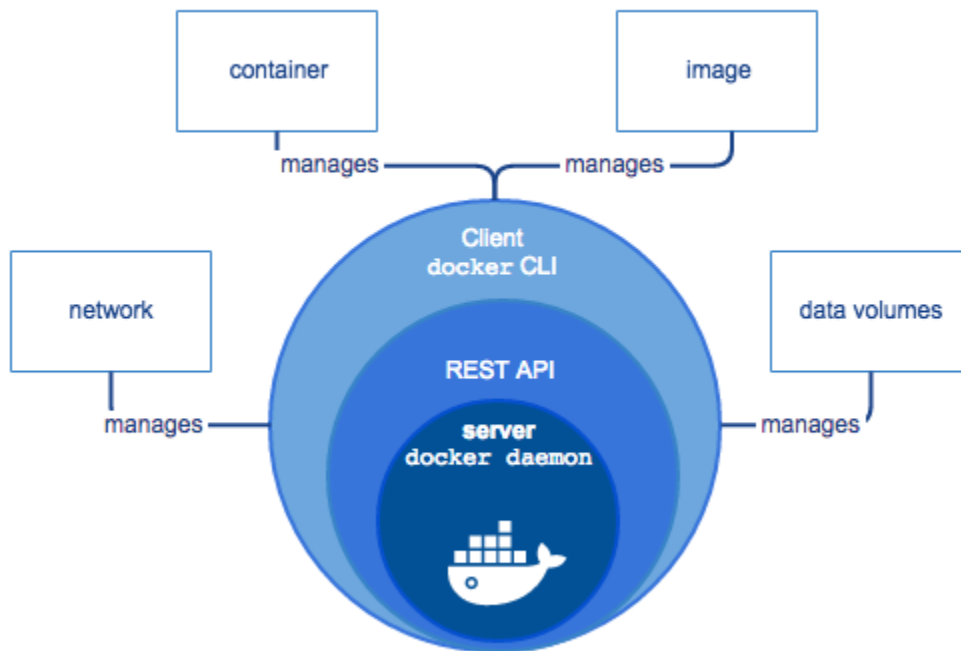


Figure 10. Docker engine (Docker overview, n.d.)

Most components that are used in this project are running in containers. This provides easy upgradeability, maintenance and scalability. The components that are not in containers are recommended that they should not be ran in Docker containers. One of the reasons why Docker is used in this project is its global popularity and demand as illustrated in Figure 11.

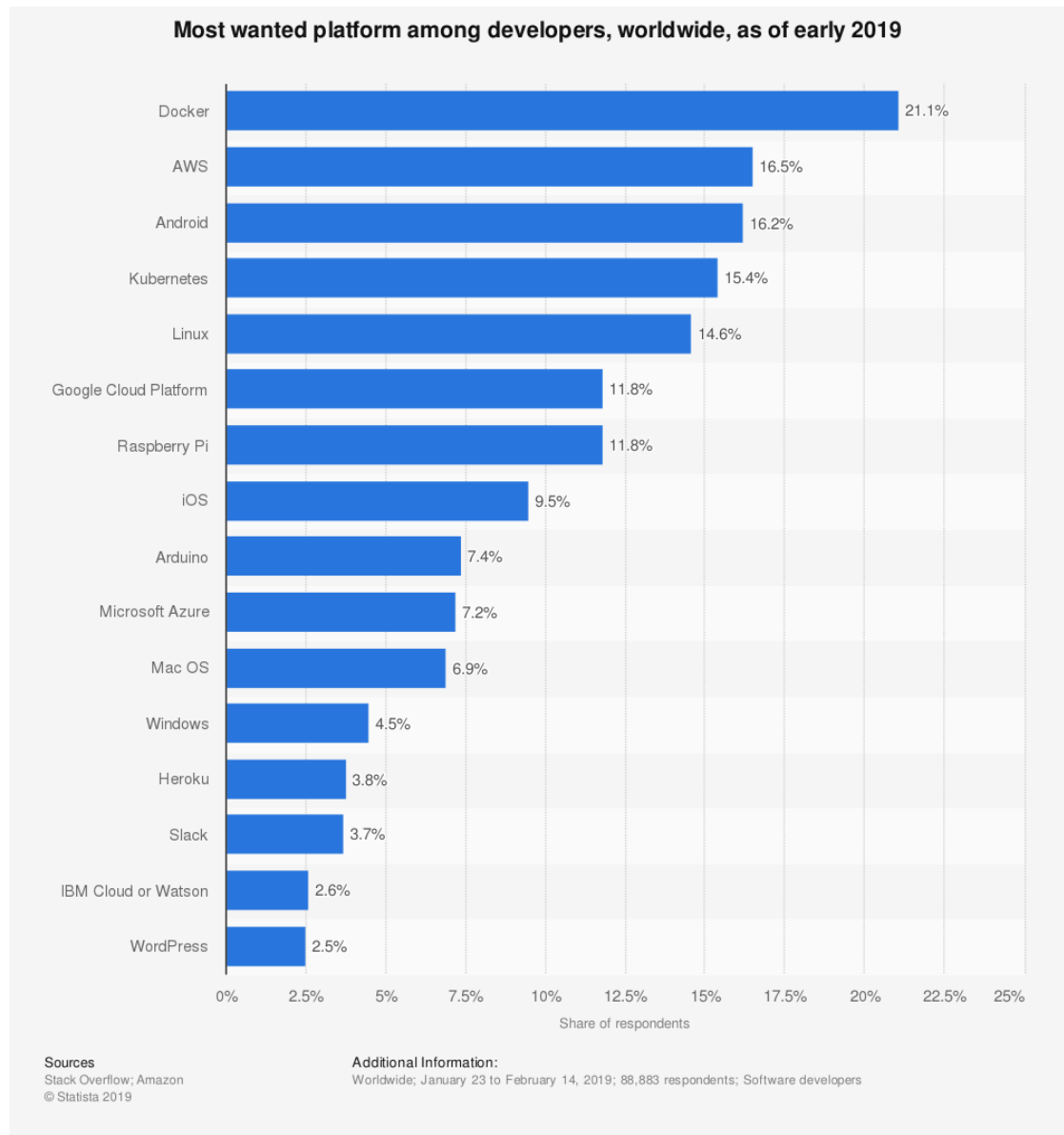


Figure 11. Docker demand (Liu 2019b)

3.8 Elasticsearch

Elasticsearch is a distributed document store that uses a data structure called an inverted index. An inverted index supports fast full-text searches, and it can identify all the documents a certain word occurs in. This is possible due to the inverted index listing every unique word appearing in any document, while also identifying all the documents each word occurs in. Elasticsearch stores complex data structures serialized as JSON documents, instead of storing the information as rows of columnar

data. Elasticsearch has a REST API that can be used to manage the cluster and its settings, as well as searching and indexing data. (Data in: documents and indices n.d.)

Elasticsearch can also be schema-less, meaning that indexing documents is possible without individually specifying how to handle each of the fields occurring in a document. Dynamic mapping allows Elasticsearch to automatically detect and add new fields to the index. (Data in: documents and indices n.d.)

It is possible to add nodes to a cluster to increase its capacity. Elasticsearch will automatically distribute the user's data and query load across all the available nodes. The data is stored in shards, and a shard is a standalone Lucene instance. Each document in an index belongs to one primary shard. In addition to primary shards, Elasticsearch also has replica shards, which are copies of the primary shards. These replicas provide redundant copies of the user's data to protect against hardware failure. They also increase the cluster's capacity to serve read requests, such as searching or retrieving documents. (Scalability and resilience: clusters, nodes, and shards n.d.)

All the logs collected in this thesis are stored in Elasticsearch. They are indexed into daily indices and separated to audit logs and application logs. Elasticsearch was in the top 3 of most wanted database skills among developers in 2019, as can be seen in

Figure 12.

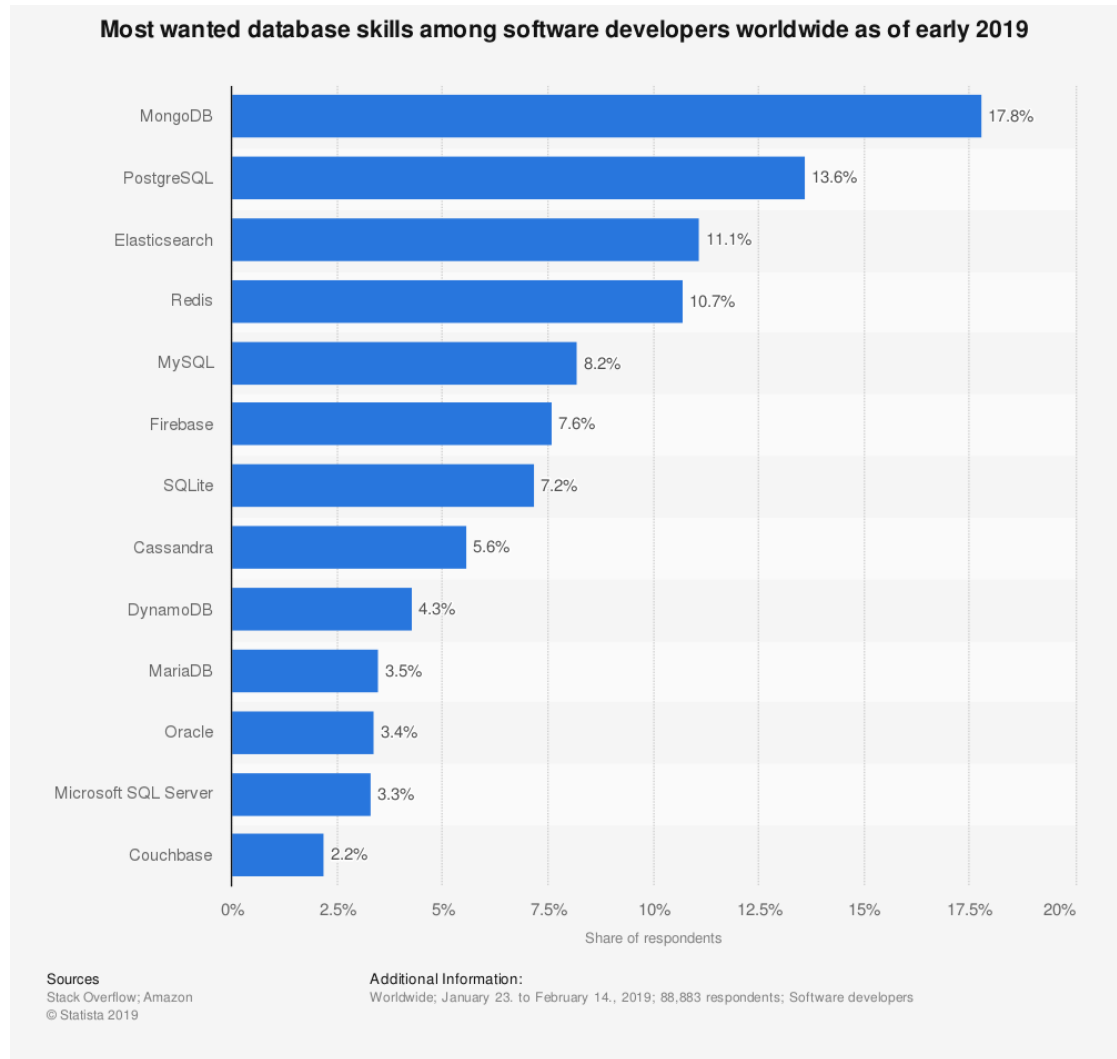


Figure 12. Elasticsearch popularity (Liu 2019a)

3.9 Filebeat and Auditbeat

The official documentation describes Filebeat as “a lightweight log shipper for forwarding and centralizing log data”. It is installed as an agent on the user’s server, and then it monitors the specified log files or locations, forwarding the data either to Elasticsearch or Logstash for indexing. (Filebeat overview n.d.)

Auditbeat is a similar lightweight log shipper that can be installed on servers to audit the system users’ activities and system processes. Potential use cases include detecting changes to critical files, such as binaries and configuration files, as well as identifying potential security policy violations. (Auditbeat overview n.d.)

Filebeat and Auditbeat are used for this thesis as the log prospectors. They are the only components not running in containers apart from Docker and Mesos-slaves. Filebeat is assigned specific folders to prospect logs, and applications are configured to send the logs to that specific folder. The logs prospected are then sent to Logstash for filtering and formatting. Auditbeat is used for collecting audit logs, which are then also sent to Logstash for filtering and formatting.

3.10 Kibana

Kibana is an open-source software used for analytics and visualization, which can be used to explore data stored in Elasticsearch. It is designed to only use Elasticsearch as its data source. Kibana allows the user to manage the Elastic Stack, for example by managing security settings or assign user roles. (Kibana – your window to Elastic Stack n.d.)

Kibana provides a user-friendly interface, where the logs can be easily accessed and searched. Different indices can be selected, in this case Auditbeat and Filebeat have separate indices and those can then be filtered further with different queries. Graphs and other kinds of visualizations can be created to visualize the incoming log data. In this project, Kibana is used as the main tool for the end user. Developers and operational specialists can inspect the logs easily from Kibana.

3.11 Logstash

The official documentation defines Logstash as “an open source data collection engine with real-time pipelining capabilities”. It enables the user to enrich and transform events with different inputs, filters and output plugins. Grok is the cornerstone of Logstash filters, and it is the most commonly used Logstash filter for deriving structure out of raw data. (Logstash Introduction n.d.)

Logstash takes in the logs sent from Filebeat and Auditbeat. It then checks for the tag that the line of log has and mutates it accordingly. Filebeat is currently sending JSON-type logs, so Logstash is configured to deal with those.

3.12 Marathon

Marathon is a container orchestration platform used with Mesosphere's Datacenter Operating System (DC/OS) and Apache Mesos. In this project, it is used in tandem with Apache Mesos and not with DC/OS. (Marathon: A container orchestration platform for Mesos and DC/OS. n.d.)

The container that we are collecting the logs from is deployed from the Marathon user interface. Marathon is used for Docker container orchestration and configuration management. The instances are also upscaled and downscaled from the Marathon user interface in this project. In production use, all of these changes will be made through the Marathon API.

3.13 Mesos

Apache Mesos is a distributed systems kernel providing applications with API's for managing and scheduling resources. This can be done across entire datacenters and cloud environments. According to Apache Mesos' website, it also "abstracts CPU, memory, storage, and other compute resources away from machines (physical or virtual), enabling fault-tolerant and elastic distributed systems to easily be built and run effectively". The inner workings of Mesos can be seen in Figure 13. (Apache Mesos

n.d.)

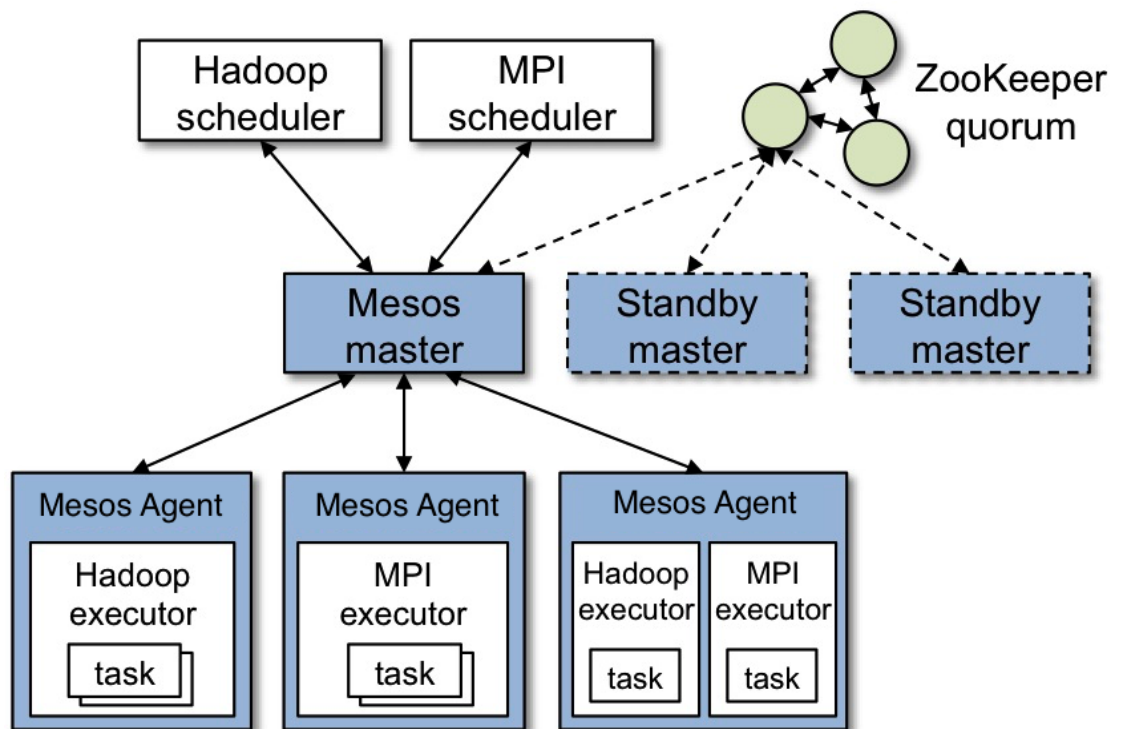


Figure 13. Mesos components

Mesos takes the payloads from Marathon and assigns them to nodes that have resources available on them. The application nodes have Mesos-slaves installed, and they will run the Docker containers assigned by Marathon.

3.14 Apache ZooKeeper

ZooKeeper is a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services. All of these kinds of services are used in some form or another by distributed applications. Apache ZooKeeper is used in this thesis for the leader election of the Mesos and Marathon masters. (Welcome to Apache ZooKeeper™. n.d.)

3.15 Consul and Registrator

Consul is used for service discovery, which allows the use of hostnames instead of static IP-addresses, e.g. elasticsearch.service.consul. This makes the platform more

flexible and decreases the amount of changes that one would have to do to the configuration, if the environment configuration would change. (Consul. n.d.)

Registrar is provided by Gliderlabs and is used to register and deregister containers. Registrar scans for the containers and automatically registers or deregisters them from Consul. It is a Docker container running on all nodes in the project environment.

4 Development and automation

4.1 Overview

The main requirement for this project was to collect application logs from Docker containers and audit logs from the operating system. Due to that, the development began with becoming familiar with the software stack that Qvantel already had in use for their logging purposes. The first order of business was to get a high-level understanding of the software stack that was going to be used in this project.

As a fully automated installation of the entire solution using Ansible was a requirement set by the stakeholders, knowledge of Ansible was essential since it was the tool used to configure and deploy all the components. Fortunately, Ansible had great documentation that helped significantly when going through the playbooks and initial architecture. Qvantel already had some Ansible playbooks written, so they were used as a template when creating playbooks for the logging components.

Knowledge of Docker was the second essential technology that the author had to familiarize himself with, although he already had previous experience with Docker from Wimma Lab, a summer program organized by JAMK University of Applied Sciences. The main workload in this project consisted of configuring mount points, modifying configuration files, modifying Ansible playbooks, Ansible configuration templates and environment variables.

General knowledge of all current and future environments was also crucial in this project, as the configuration of all the components varied depending on the environ-

ment. For example, the file paths of the logs may vary from one environment to another. Testing also had to be carried out against multiple different operating systems and kernel versions to ensure compatibility across all customers.

Lastly and most importantly, the author had to familiarize himself with the Elastic Stack and the inner working of each of the components included in the stack. Auditbeat and Filebeat configurations are mostly default configurations that were recommended by Elastic, with some changes to the file paths and such.

Consul, Zookeeper, Mesos and Marathon are used for orchestrating the log creation Docker container; therefore, only very basic knowledge of them was required for this specific project. Installation and configuration were carried out with readymade Ansible playbooks, with slight configuration changes such as configuring the correct IP addresses and file paths.

As one of the requirements were that important configuration must be modifiable with environment files or automatically generated, all relevant configurations would have to be modifiable with variables for ease of use purposes. The plan was to develop a fully automated logging system that can be deployed to customer environments. Previously the logging system was not containerized; hence Elasticsearch, Logstash and Kibana installations were modified to support Docker containers.

4.2 Configuration management

All the configuration such as Ansible playbooks and configuration file templates were stored in Qvantel's version control repository. Git was used to keep all the configurations up to date with the latest changes. All the software images, container images and operating system images were found in Qvantel's internal repository manager.

4.3 Playbook configurations

All the nodes had Docker, Beats, Registrator and Consul installed on them. Those services were needed for any of the components to function as intended.

First, the orchestration nodes were installed to add the supporting services that the other nodes need, such as service discovery from Consul. The orchestration nodes

contain all the master-role containers used in this project: Marathon, Consul, Zookeeper and Mesos. The playbook that was used to deploy the nodes can be seen in Figure 14.

```

---
- name: get facts from log hosts
  hosts: log
  tags: configure

- name: set up orchestration nodes
  hosts: orchestration
  vars_files:
    - vars/versions.yml
  become: yes

pre_tasks:
- name: include instance-specific extra variables
  include_vars: "{{ item }}"
  with_fileglob: extra_vars/*
  tags: configure

- name: add mesos_masters, zookeeper_servers, consul_servers
  add_host:
    groups: zookeeper_servers,consul_servers,mesos_masters
    name: "{{ item }}"
  with_items:
    - "{{ groups['orchestration'] }}"
  changed_when: false
  tags: configure

- name: ensure firewalld is installed
  yum:
    name: firewalld
    state: present

- include_role:
  name: qvcp-common
  when: package_install == 'online'

- include: "{{ item }}"
  when: package_install == 'offline'
  with_items:
    - rpm_packages.yml
    - pypl.yml

- include: users/orchestration_users.yml

- include_role:
  name: docker

- include: container_images.yml
  when: package_install == 'offline'

roles:
- { role: beats }
- { role: docker-consul, consul_role: 'server' }
- { role: docker }
- { role: docker-registrator }
- { role: docker-zookeeper }
- { role: docker-mesos, mesos_role: 'master' }
- { role: docker-marathon }

tasks:
- include: package_cleanup.yml
  when: package_install == 'offline'

```

Figure 14. Orchestration node installation playbook

After the orchestration nodes were installed, the nodes that the applications would run on were installed. The application nodes contain all the slave-role nodes used in this project: Consul and Mesos. The playbook that was ran can be seen in Figure 15.

```

---
- name: get facts from orchestration nodes
  hosts: orchestration
  tags: configure

- name: get facts from log hosts
  hosts: log
  tags: configure

- name: set up application nodes
  hosts: application
  vars_files:
    - vars/versions.yml
  become: yes

pre_tasks:
  - name: include instance-specific extra variables
    include_vars: "{{ item }}"
    with_fileglob: extra_vars/*
    tags: configure

  - name: add mesos_masters, zookeeper_servers, consul_server
    add_host:
      groups: zookeeper_servers,consul_servers,mesos_masters
      name: "{{ item }}"
    with_items:
      - "{{ groups['orchestration'] }}"
    changed_when: false
    tags: configure

  - name: ensure firewalld is installed
    yum:
      name: firewalld
      state: present

  - include_role:
      name: qvcp-common
      when: package_install == 'online'

  - include: "{{ item }}"
      when: package_install == 'offline'
    with_items:
      - rpm_packages.yml
      - pypl.yml

  - include: users/application_users.yml

  - include_role:
      name: docker

  - include: container_images.yml
      when: package_install == 'offline'

roles:
  - { role: beats }
  - { role: docker-consul, consul_role: 'agent' }
  - { role: docker-mesos, mesos_role: 'slave' }
  - { role: docker-registrator }
  - { role: docker-cadvisor }

tasks:
  - include: package_cleanup.yml
      when: package_install == 'offline'

  - name: set platform version number
    lineinfile:
      create: yes
      insertbefore: BOF
      regexp: '.*'
      path: /etc/qvcp_version
      line: "{{ platform_version }}"

```

Figure 15. Application node installation playbook

Finally, the logging node was installed. The playbook that was run can be seen in Figure 16. The Elastic Stack was installed on this node to collect the logs, filter and format them in Logstash and visualize them in Kibana.

```

---
- name: get facts from orchestration nodes
  hosts: orchestration
  tags: configure

- name: set up logging nodes
  hosts: log
  vars_files:
    - vars/versions.yml
  become: yes

  pre_tasks:

    - name: add all extra_vars
      include_vars: "{{ item }}"
      with_fileglob: extra_vars/*
      tags: configure

    - name: add zookeeper_servers & consul-masters
      add_host:
        groups: zookeeper_servers,consul_servers,mesos_masters
        name: "{{ item }}"
      with_items:
        - "{{ groups['orchestration'] }}"
      changed_when: false
      tags: configure

    - name: ensure firewalld is installed
      yum:
        name: firewalld
        state: present

    - include_role:
        name: qvcp-common
        when: package_install == 'online'

    - include: "{{ item }}"
      when: package_install == 'offline'
      with_items:
        - rpm_packages.yml
        - pypi.yml

    - include: users/log_users.yml

    - include_role:
        name: docker

    - include: container_images.yml
      when: package_install == 'offline'

  roles:
    - { role: beats }
    - { role: docker-consul, consul_role: 'agent' }
    - { role: docker-registrator }
    - { role: docker-elasticsearch }
    - { role: docker-kibana }
    - { role: docker-logstash }

  tasks:
    - include: package_cleanup.yml
      when: package_install == 'offline'

```

Figure 16. Logging node installation playbook

4.4 Component configurations

No manual changes were made to these files; the files were created by the Ansible playbooks and the values were placed from the environment files.

The most crucial components for this project were included in the Elastic Stack. Those configurations will be inspected more closely in the next sections.

4.4.1 Auditbeat

Auditbeat is using only a single configuration file in this setup, found at `/etc/auditbeat/auditbeat.yml`. The most important part of this configuration can be seen in Figures 17-19, where the different Auditbeat modules are configured. The modules determine what logs are collected from the operating system. In this case, the Auditd and `file_integrity` modules are enabled. The different file paths and syscall audits are defined in these modules.

```
auditbeat.modules:
- module: auditd
  audit_rules: |
    ## Define audit rules here.
    ## Create file watches (-w) or syscall audits (-a or -A). Uncomment these
    ## examples or add your own rules.

    ## If you are on a 64 bit platform, everything should be running
    ## in 64 bit mode. This rule will detect any use of the 32 bit syscalls
    ## because this might be a sign of someone exploiting a hole in the 32
    ## bit API.
    -a always,exit -F arch=b32 -S all -F key=32bit-abi

    ## Executions.
    -a always,exit -F arch=b64 -S execve,execveat -k exec

    ## External access (warning: these can be expensive to audit).
    #-a always,exit -F arch=b64 -S accept,bind,connect -F key=external-access

    ## Identity changes.
    -w /etc/group -p wa -k identity
    -w /etc/passwd -p wa -k identity
    -w /etc/gshadow -p wa -k identity

    ## Security changes
    -w /etc/security/ -p wa -k security
    -w /etc/selinux/ -p wa -k selinux_changes

    ## Configuration changes
    -w /etc/sysconfig/ -p wa -k CFG_sysconfig

    ## Logging
    -w /var/log/lastlog -p wa -k logins
    -w /var/log/faillog -p wa -k logins

    # Watch syslog configuration
    -w /etc/syslog.conf -k syslog

    ## Watch PAM and authentication configuration
    -w /etc/pam.d/ -k PAM_config
    -w /etc/nsswitch.conf -k PAN_config

    ## Watch system log files
    -w /var/log/messages -k LOG_messages
    -w /var/log/audit/audit.log -k LOG_audit
    -w /var/log/audit/audit[1-4].log -k LOG_audit

    ## Unauthorized access attempts.
    -a always,exit -F arch=b64 -S open,creat,truncate,ftruncate,openat,open_by_handle_at -F exit=-EACCES -k access
    -a always,exit -F arch=b64 -S open,creat,truncate,ftruncate,openat,open_by_handle_at -F exit=-EPERM -k access

    ## SYSTEM events
    -a always,exit -F arch=b64 -S acct,reboot,sched_setparam,sched_setscheduler,setrlimit,swapon -k SYS_event

    ## Systems Network Environment
    -a always,exit -F arch=b64 -S sethostname,setdomainname -k audit_network_modifications
    -w /etc/issue -p wa -k audit_network_modifications
    -w /etc/issue.net -p wa -k audit_network_modifications
    -w /etc/hosts -p wa -k audit_network_modifications
    -w /etc/sysconfig/network -p wa -k audit_network_modifications

    ## Process and Session Initiation Information
    -w /var/run/utmp -p wa -k session
    -w /var/log/btmp -p wa -k session
    -w /var/log/wtmp -p wa -k session
```

Figure 17. Auditbeat Auditd module

```

## Discretionary Access Control Permission Modification Events
-a always,exit -F arch=b32 -S chmod,fchmod,fchmodat -k perm_file_mod
-a always,exit -F arch=b32 -S chown,fchown,fchownat,lchown -k perm_file_mod
-a always,exit -F arch=b32 -S setxattr,lsetxattr,fsetxattr,removexattr,lremovexattr,removexattr -k perm_file_mod
-a always,exit -F arch=b64 -S chmod,fchmod,fchmodat -k perm_file_mod
-a always,exit -F arch=b64 -S chown,fchown,fchownat,lchown -k perm_file_mod
-a always,exit -F arch=b64 -S setxattr,lsetxattr,fsetxattr,removexattr,lremovexattr,removexattr -k perm_file_mod

##Information on the Use of Privileged Commands
-a always,exit -F path=/bin/ping -F perm=x -k privileged
-a always,exit -F path=/bin/umount -F perm=x -k privileged
-a always,exit -F path=/bin/mount -F perm=x -k privileged
-a always,exit -F path=/bin/su -F perm=x -k privileged
-a always,exit -F path=/bin/chgrp -F perm=x -k privileged
-a always,exit -F path=/bin/ping6 -F perm=x -k privileged
-a always,exit -F path=/sbin/pam_timestamp_check -F perm=x -k privileged
-a always,exit -F path=/sbin/unix_chkpwd -F perm=x -k privileged
-a always,exit -F path=/sbin/pwck -F perm=x -k privileged
-a always,exit -F path=/usr/sbin/suexec -F perm=x -k privileged
-a always,exit -F path=/usr/sbin/useradd -F perm=x -k privileged
-a always,exit -F path=/usr/sbin/userdel -F perm=x -k privileged
-a always,exit -F path=/usr/sbin/usermod -F perm=x -k privileged
-a always,exit -F path=/usr/sbin/newusers -F perm=x -k privileged
-a always,exit -F path=/usr/sbin/groupadd -F perm=x -k privileged
-a always,exit -F path=/usr/sbin/groupdel -F perm=x -k privileged
-a always,exit -F path=/usr/sbin/groupmod -F perm=x -k privileged
-a always,exit -F path=/usr/sbin/semanage -F perm=x -k privileged
-a always,exit -F path=/usr/sbin/usernetctl -F perm=x -k privileged
-a always,exit -F path=/usr/sbin/userhelper -F perm=x -k privileged
-a always,exit -F path=/usr/bin/Xorg -F perm=x -k privileged
-a always,exit -F path=/usr/bin/rlogin -F perm=x -k privileged
-a always,exit -F path=/usr/bin/sudoedit -F perm=x -k privileged
-a always,exit -F path=/usr/bin/sudoedit -F perm=x -k privileged
-a always,exit -F path=/usr/bin/rsh -F perm=x -k privileged
-a always,exit -F path=/usr/bin/gpasswd -F perm=x -k privileged
-a always,exit -F path=/usr/bin/crontab -F perm=x -k privileged
-a always,exit -F path=/usr/bin/sudo -F perm=x -k privileged
-a always,exit -F path=/usr/bin/staprun -F perm=x -k privileged
-a always,exit -F path=/usr/bin/rcp -F perm=x -k privileged
-a always,exit -F path=/usr/bin/passwd -F perm=x -k privileged
-a always,exit -F path=/usr/bin/chsh -F perm=x -k privileged
-a always,exit -F path=/usr/bin/chfn -F perm=x -k privileged
-a always,exit -F path=/usr/bin/chage -F perm=x -k privileged
-a always,exit -F path=/usr/bin/setfacl -F perm=x -k privileged
-a always,exit -F path=/usr/bin/chacl -F perm=x -k privileged
-a always,exit -F path=/usr/bin/chcon -F perm=x -k privileged
-a always,exit -F path=/usr/bin/newgrp -F perm=x -k privileged
-a always,exit -F path=/usr/bin/kpac_dhcp_helper -F perm=x -k privileged

## Files Deletion Events by User (successful and unsuccessful)
-a always,exit -F arch=b64 -S unlink,rmdir,unlinkat,rename,renameat -k delete
-a always,exit -F arch=b64 -S rmdir,unlink,unlinkat,rename,renameat -F auid=0 -k delete

```

Figure 18. Auditbeat Audited module continued

```

- module: file_integrity
  paths:
  - /bin
  - /usr/bin
  - /sbin
  - /usr/sbin
  - /etc

```

Figure 19. Auditbeat file_integrity module

4.4.2 Filebeat

Filebeat has two different configuration file locations defined for it in this setup. The prospectors and the main configuration file. The prospectors can be found in `/etc/filebeat/prospectors/`. Those files contain the file paths that Filebeat monitors

and the tags needed for Logstash filtering. In the filebeat.yml the filebeat is configured to send all logs to the Logstash host, as can be seen in Figure 20.

```
output.logstash:
  # The Logstash hosts
  hosts:
    - 192.168.81.76:5044
  # Optional SSL. By default is off.
  # List of root certificates for HTTPS server verifications
  #ssl.certificate_authorities: ["/etc/pki/root/ca.pem"]

  # Certificate for SSL client authentication
  #ssl.certificate: "/etc/pki/client/cert.pem"

  # Client Certificate Key
  #ssl.key: "/etc/pki/client/cert.key"
```

Figure 20. Filebeat Logstash output

4.4.3 Logstash

Logstash is configured to receive events from Beats, which is shown in Figure 21.

The default Logstash port is 5044.

```
input {
  beats {
    port => 5044
  }
}
```

Figure 20. Logstash input

In the filter section, there are three different filter options based on the tag that the event has: “container”, “mesos” and “nginx”. All these logs have different formats, so they are parsed differently. In this project, there are no Nginx logs included; hence, that filter is not used. If the log does not match any of those tags, the event is dropped. The filter is shown in Figure 22.

```

filter {
  if ("container" in [tags]) {
    json {
      source => "message"
    }

    mutate {
      update => { "message" => "%{log}" }
      remove_field => [ "log" ]
    }

    date {
      match => [ time, "ISO8601" ]
      remove_field => [ "time" ]
    }
  }

  else if ("mesos" in [tags]) {
    json {
      source => "message"
    }
  }

  else if ("nginx" in [tags]) {
    grok {
      match => { "message" => "%{IPORHOST:clientip} %{USER:ident} %{USER:auth} \[%{HTTPDATE:timestamp}\] \"(?:%{WORD:verb} %{URIPATHPARAM:request})?(?: HTTP/%{NUMBER:httpversion})?%{(GREEDYDATA:rawrequest)}\" %{NUMBER:response} (?:%{(NUMBER:bytes)}|-) \"(?:%{(URI:referrer)}|%{(GREEDYDATA:referrer)})\" %{QS:agent} %{NUMBER:request_time}" }
    }

    date {
      match => [ timestamp, "dd/MMM/yyyy:HH:mm:ss Z" ]
      remove_field => [ "timestamp" ]
    }
  }

  else {
    drop {}
  }
}

```

Figure 21. Logstash filters

The output is configured to send the data to Elasticsearch, as shown in Figure 23.

```

output {
  elasticsearch {
    hosts => ["192.168.81.76:9200"]
    index => "%{[@metadata][beat]}-%{[@metadata][version]}-%{+YYYY.MM.dd}"
  }
}

```

Figure 22. Elasticsearch output

The Elasticsearch IP is automatically generated from the Ansible environment file.

The index is defined by the metadata and Beat. The version number of the Beat and the current date are included in the index name, which in practice means that the daily indices are created for Auditbeat and Filebeat.

4.4.4 Elasticsearch

The Elasticsearch configuration is rather simple, as most of the configuration needed is done already in the Beats and Logstash. In the Elasticsearch configuration, the node name is defined, the IP that is published is defined and the logging level is defined. All these configurations can be seen in Figure 24.

```
node.name: cp16

network.publish_host: 192.168.81.76

logger._root: warn
```

Figure 23. Elasticsearch configuration

4.4.5 Kibana

In the Kibana configuration, the server name, Elasticsearch's URL and the logging level are defined. The configuration is shown in Figure 25. The user can choose to make their own settings such as saved queries and filters from the UI.

```
server.name: "Kibana"
elasticsearch.url: "http://192.168.81.76:9200"
logging.quiet: true
```

Figure 24. Kibana configuration

5 Implementation

5.1 Virtual machine creation

The virtual machines were created using Virsh. The Virsh is installed on a hypervisor, which has resources available for virtual machines. The machines were installed using a readymade script located on the hypervisor. In Figure 26 the list of virtual machines deployed can be seen.


```
[root@c02h001i lmoil]# virsh list --all
setlocale: No such file or directory
 Id      Name      State
-----
 453     cp09      running
 454     cp11      running
 455     cp13      running
 456     cp15      running
 457     cp16      running
```

Figure 25. List of VMs

5.2 Environment

The so-called launch machine where the Ansible playbooks were deployed from was the author's MacBook, provided by Qvantel. The prerequisites for running the playbooks were installing Ansible and Sshpass. Ansible is used for deploying the playbooks, and to deploy over SSH Sshpass needs to be installed. There was an issue with the default Sshpass installation, so a workaround was implemented as can be seen in Figures 28-29. The installation was carried out with brew, which is a packet manager for MacOS, as can be seen in Figures 27-29.

```
QFINM234:thesis lmoilanen$ brew install ansible
```

Figure 26. Ansible installation

```
QFINM234:thesis lmoilanen$ brew install sshpass
Updating Homebrew...
=> Auto-updated Homebrew!
Updated 1 tap (homebrew/core).
=> Updated Formulae
buku

Error: No available formula with the name "sshpass"
We won't add sshpass because it makes it too easy for novice SSH users to
ruin SSH's security.
```

Figure 27. Sshpass installation issue

```

QFINM234:thesis lmoilanen$ brew install http://git.io/sshpas.rb
##### 100.0%
Warning: A newer Command Line Tools release is available.
Update them from Software Update in System Preferences or
https://developer.apple.com/download/more/.

=> Downloading http://sourceforge.net/projects/sshpas/files/sshpas/1.05/sshpas-1.05.tar.gz
=> Downloading from https://netcologne.dl.sourceforge.net/project/sshpas/sshpas/1.05/sshpas-1.05.tar.gz
##### 100.0%
=> ./configure --prefix=/usr/local/Cellar/sshpas/1.05
=> make install
📦 /usr/local/Cellar/Cellar/sshpas/1.05: 9 files, 44.8KB, built in 28 seconds

```

Figure 28. Sshpass installation workaround

The environment consists of five virtual machines, all with identical resource configurations. Each of the nodes had 9.8GB of storage allocated to them, shown in Figure 30.

```

[root@cp09 lmoilanen]# df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        4.8G   0  4.8G   0% /dev
tmpfs           4.8G   0  4.8G   0% /dev/shm
tmpfs           4.8G  25M  4.8G   1% /run
tmpfs           4.8G   0  4.8G   0% /sys/fs/cgroup
/dev/mapper/main-root 9.8G  3.0G  6.3G  32% /
/dev/vda1       575M  247M  287M  47% /boot
/dev/mapper/main-var 9.8G  4.1G  5.2G  45% /var
tmpfs           975M   0  975M   0% /run/user/1640

```

Figure 29. Storage allocation

Each node had 9.5GB of RAM allocated to them, shown in Figure 31.

```

[root@cp09 lmoilanen]# free -h
              total        used         free       shared  buff/cache   available
Mem:           9.5G         533M         1.2G           24M           7.8G         8.7G
Swap:          0B             0B             0B

```

Figure 30. RAM allocation

And two virtual processing cores, as shown in Figure 32.

```
[root@cp09 lmoilanen]# lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                2
On-line CPU(s) list:   0,1
Thread(s) per core:    1
Core(s) per socket:    1
Socket(s):             2
NUMA node(s):          1
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 44
Model name:            Westmere E56xx/L56xx/X56xx (IBRS update)
Stepping:              1
CPU MHz:               2925.841
BogoMIPS:              5851.99
Hypervisor vendor:     KVM
Virtualization type:   full
L1d cache:             32K
L1i cache:             32K
L2 cache:              4096K
NUMA node0 CPU(s):    0,1
Flags:                 fpu vme de pse tsc msr pae mce cx8 apic sep mtr
                        d sse4_1 sse4_2 x2apic popcnt aes hypervisor lahf_lm pti
```

Figure 31. vCPU allocation

All nodes are running Oracle Linux 7.8, with kernel version 4.1.12-124.37.1.el7uek.x86_64, as can be seen in Figure 33.

```
[root@cp15 lmoilanen]# uname -a
Linux cp15 4.1.12-124.37.1.el7uek.x86_64 #2 SMP Wed Mar 4 16:28:22 PST 2020 x86_64 x86_64 x86_64 GNU/Linux
[root@cp15 lmoilanen]# cat /etc/os-release
NAME="Oracle Linux Server"
VERSION="7.8"
ID="ol"
ID_LIKE="Fedora"
VARIANT="Server"
VARIANT_ID="server"
VERSION_ID="7.8"
PRETTY_NAME="Oracle Linux Server 7.8"
ANSI_COLOR="0;31"
CPE_NAME="cpe:/o:oracle:linux:7.8:server"
HOME_URL="https://linux.oracle.com/"
BUG_REPORT_URL="https://bugzilla.oracle.com/"

ORACLE_BUGZILLA_PRODUCT="Oracle Linux 7"
ORACLE_BUGZILLA_PRODUCT_VERSION=7.8
ORACLE_SUPPORT_PRODUCT="Oracle Linux"
ORACLE_SUPPORT_PRODUCT_VERSION=7.8
```

Figure 32. Kernel version and operating system version

In the project environment, there were two orchestration nodes, which act as the coordinating nodes. The server deployments were identical, with both running Marathon, Mesos-master, Zookeeper, Registrator and Consul-Server. All these components were in Docker containers, as can be seen in Figure 34. A truncated log of the Ansible playbook run is presented in Appendix 1.

```
[root@cp09 lmoilanen]# docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
1434c6d079b6	artifactory.qvante.net/cp-marathon:v1.5.8	"bin/marathon"	38 minutes ago	Up 35 minutes		marathon
f4c9418ef435	artifactory.qvante.net/cp-mesos-master:1.5.0	"mesos-master --regi..."	About an hour ago	Up 35 minutes		mesos-master
796033422d67	artifactory.qvante.net/cp-zookeeper:3.4.12	"/docker-entrypoint..."	2 hours ago	Up 2 hours		zookeeper
58f9aacc7c2	artifactory.qvante.net/cp-registrator:master	"/bin/registrator -i..."	2 hours ago	Up 2 hours		registrator
fc94b74bf940	artifactory.qvante.net/cp-consul:1.1.0	"docker-entrypoint.s..."	2 hours ago	Up About an hour		consul-server

Figure 33. Docker containers running on orchestration nodes

Two of the five machines were dedicated as application nodes, which are responsible for running the payloads Marathon assigns. As was the case with the orchestration nodes, the deployments between these two nodes were also identical. In Figure 35 it can be seen that the payload from Marathon has been assigned to this specific node as it does have a Mesos-slave running on it (Figure 36). Looking at the other node in Figure 37, it can be seen that it does not currently have any provisioned containers running on it. A truncated log of the Ansible playbook run is presented in Appendix 2.

```
[root@cp13 lmoilanen]# docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
385f7fc3b6	artifactory.qvante.net/trump	"/bin/sh -c /bin/tru..."	25 minutes ago	Up 25 minutes		mesos-d4ca13e8-33ff-4898-932d-ce9ad6d4580b
3bc7d215910	artifactory.qvante.net/cp-registrator:master	"/bin/registrator -i..."	26 minutes ago	Up 26 minutes		registrator
86c9494262b	artifactory.qvante.net/cp-consul:1.1.0	"docker-entrypoint.s..."	28 minutes ago	Up 28 minutes		consul-agent

Figure 34. Orchestrated container "trump"

```
[root@cp13 lmoilanen]# systemctl status mesos-slave -l
```

```
mesos-slave.service - Mesos Slave
Loaded: loaded (/usr/lib/systemd/system/mesos-slave.service; enabled; vendor preset: disabled)
Active: active (running) since Sat 2020-04-11 16:45:49 EEST; 32min ago
Main PID: 19965 (mesos-slave)
Memory: 26.1M
Group: /system.slice/mesos-slave.service
├─19965 /usr/sbin/mesos-slave --master-zk://192.168.81.69:2181,192.168.81.71:2181/mesos --containerizers=docker_mesos --executor_registration_timeout=10mins --work_dir=/var/lib/mesos
├─19977 logger -p user.info -t mesos-slave[19965]
└─19978 logger -p user.err -t mesos-slave[19965]
```

```
Apr 11 17:08:49 cp13 mesos-slave[19978]: 10411 17:08:49.668887 19981 slave.cpp:6360] Current disk usage 23.39%. Max allowed age: 15.985317514057501hrs
Apr 11 17:09:49 cp13 mesos-slave[19978]: 10411 17:09:49.669639 19985 slave.cpp:6360] Current disk usage 23.39%. Max allowed age: 15.985270409515000hrs
Apr 11 17:10:49 cp13 mesos-slave[19978]: 10411 17:10:49.670732 19981 slave.cpp:6360] Current disk usage 23.39%. Max allowed age: 15.98523272580833hrs
Apr 11 17:11:49 cp13 mesos-slave[19978]: 10411 17:11:49.671296 19985 slave.cpp:6360] Current disk usage 23.39%. Max allowed age: 15.985204463155277hrs
Apr 11 17:12:49 cp13 mesos-slave[19978]: 10411 17:12:49.672139 19980 slave.cpp:6360] Current disk usage 23.40%. Max allowed age: 15.985166779521110hrs
Apr 11 17:13:49 cp13 mesos-slave[19978]: 10411 17:13:49.673213 19982 slave.cpp:6360] Current disk usage 23.40%. Max allowed age: 15.985138516795553hrs
Apr 11 17:14:49 cp13 mesos-slave[19978]: 10411 17:14:49.674042 19981 slave.cpp:6360] Current disk usage 23.40%. Max allowed age: 15.985110254069999hrs
Apr 11 17:15:49 cp13 mesos-slave[19978]: 10411 17:15:49.674706 19982 slave.cpp:6360] Current disk usage 23.40%. Max allowed age: 15.985072570435833hrs
Apr 11 17:16:49 cp13 mesos-slave[19978]: 10411 17:16:49.675582 19981 slave.cpp:6360] Current disk usage 23.40%. Max allowed age: 15.985044307710277hrs
Apr 11 17:17:49 cp13 mesos-slave[19978]: 10411 17:17:49.676601 19982 slave.cpp:6360] Current disk usage 23.40%. Max allowed age: 15.985016044984722hrs
```

Figure 35. Mesos-slave running on the node

```
[root@cp15 lmoilanen]# docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
57d2b9494c6f	artifactory.qvante.net/cp-registrator:master	"/bin/registrator -i..."	27 minutes ago	Up 27 minutes		registrator
f159100fd0a	artifactory.qvante.net/cp-consul:1.1.0	"docker-entrypoint.s..."	28 minutes ago	Up 28 minutes		consul-agent

Figure 36. No orchestrated containers running

As for the last node, all the logging related components were deployed there. Elasticsearch, Logstash and Kibana are running on this node (Figure 38). As can be seen, Kibana has an exposed port; 5601. One can access the Kibana user interface from that port and browse the logs (Figure 39). Elasticsearch can be accessed from port

9200, but only returns basic information about the database (Figure 40). A truncated log of the Ansible playbook run is presented in Appendix 3.

```

root@cp16-lmo1lanen# docker ps -a
CONTAINER ID        IMAGE                                     COMMAND                  CREATED              STATUS              PORTS
56d65575e091       artifactory.qvante.net/cp-logstash:6.2.3  "/logstash-entrypoint..."  32 minutes ago      Up 32 minutes      192.168.81.76:5601->5601/tcp
899a1a75302b       artifactory.qvante.net/cp-kibana:6.2.3   "/docker-entrypoint..."  32 minutes ago      Up 32 minutes      0.0.0.0:9200->9200/tcp, 0.0.0.0:9300->9300/tcp
b2f6ec314577       artifactory.qvante.net/cp-elasticsearh:6.2.3  "/elastic-entrypoint..."  About an hour ago   Up About an hour   0.0.0.0:9200->9200/tcp, 0.0.0.0:9300->9300/tcp
164731e4bc52       artifactory.qvante.net/cp-registry:master  "/bin/registry -L..."    About an hour ago   Up About an hour
27758f92d97c       artifactory.qvante.net/cp-consul:1.1.0     "docker-entrypoint.s..."  About an hour ago   Up About an hour
    
```

Figure 37. Logging components running

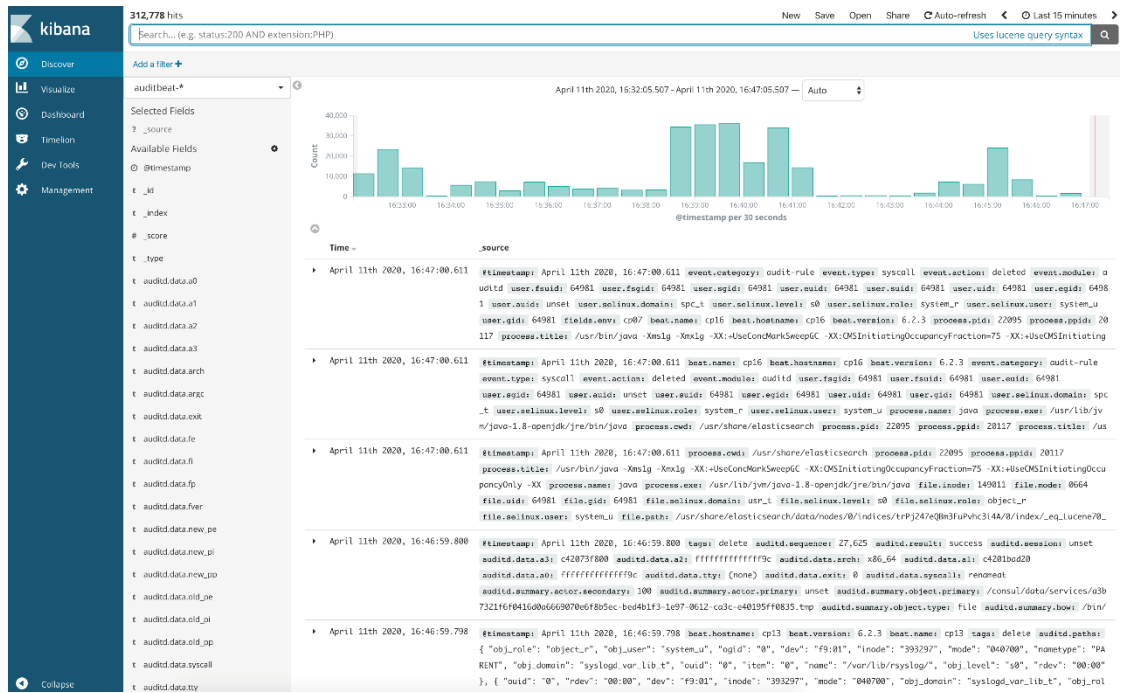
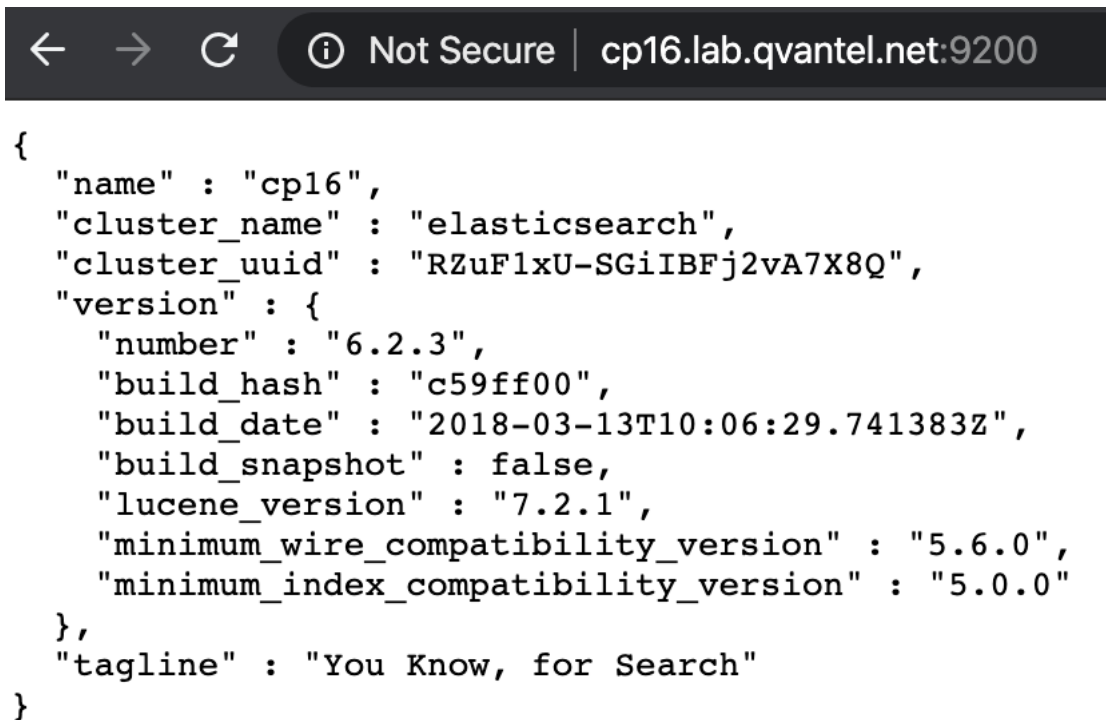


Figure 38. Kibana UI and logs



```
{
  "name" : "cp16",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "RZuFlxU-SGiIBFj2vA7X8Q",
  "version" : {
    "number" : "6.2.3",
    "build_hash" : "c59ff00",
    "build_date" : "2018-03-13T10:06:29.741383Z",
    "build_snapshot" : false,
    "lucene_version" : "7.2.1",
    "minimum_wire_compatibility_version" : "5.6.0",
    "minimum_index_compatibility_version" : "5.0.0"
  },
  "tagline" : "You Know, for Search"
}
```

Figure 39. Elasticsearch cluster information

5.3 Log Sources

Docker containers running on top of Mesos/Marathon produce logs by default to stdout and stderr. In this project, a dummy container was used that had the sole purpose of producing log events, called trump. It would randomly generate a log event which contained a random message from Donald Trump's twitter account. The configuration assigned to it from Marathon can be seen in Figure 41.

The screenshot shows a code editor titled "Edit Application" in "JSON Mode". The JSON configuration is as follows:

```

1  {
2    "id": "/trump",
3    "cmd": null,
4    "cpus": 1,
5    "mem": 128,
6    "disk": 0,
7    "instances": 1,
8    "acceptedResourceRoles": [],
9    "container": {
10   "type": "DOCKER",
11   "docker": {
12     "forcePullImage": false,
13     "image": "artifactory.qvantel.net/trump",
14     "parameters": [],
15     "privileged": false
16   },
17   "volumes": []
18 },
19 "portDefinitions": [
20 {
21   "port": 10000,
22   "name": "default",
23   "protocol": "tcp"
24 }
25 ]
26 }

```

At the bottom of the editor, there are two buttons: "Cancel" and "Change and deploy configuration".

Figure 40. "Trump" container configuration

5.4 Collection of the logs

The main focus of this project was to collect the logs from the orchestrated applications, in this case "trump". These logs are collected from `/var/lib/Mesos/slaves/*/executors/*/runs/latest/*` on application nodes by Filebeat. Each container has its unique ID included in the path of the log file. The Filebeat configuration takes this into account by using wildcards(*). Some metadata is also added to the event, which then directs it to the correct Logstash pipeline.

Logs were also collected from audit events, all the different components of the logging system and operating system events. These are also useful for debugging purposes, but they were not the main focus of this project.

5.5 Testing

Automated testing was off scoped for this project; hence only manual testing was carried out. After the installation, all nodes were checked for errors and all user interfaces were checked for error messages as well.

The logs from specific Docker containers were also checked to reveal underlying issues that could possibly occur. Systemctl-commands to Docker, Mesos, Auditbeat and Filebeat were also run done to make sure they were working as intended.

5.6 Searching logs from Kibana

Kibana uses KQL, Kibana Query Language for searching information from Elasticsearch. The most common use for Kibana is searching for log traces about issues that might have occurred during testing. For this kind of purpose, using single terms is enough to find the relevant information. Figure 42 shows many times Trump has mentioned “Wall” in his tweets in the past 15 minutes. Kibana highlights the queried

terms and provides a graph showing the occurrences in a specified time period.

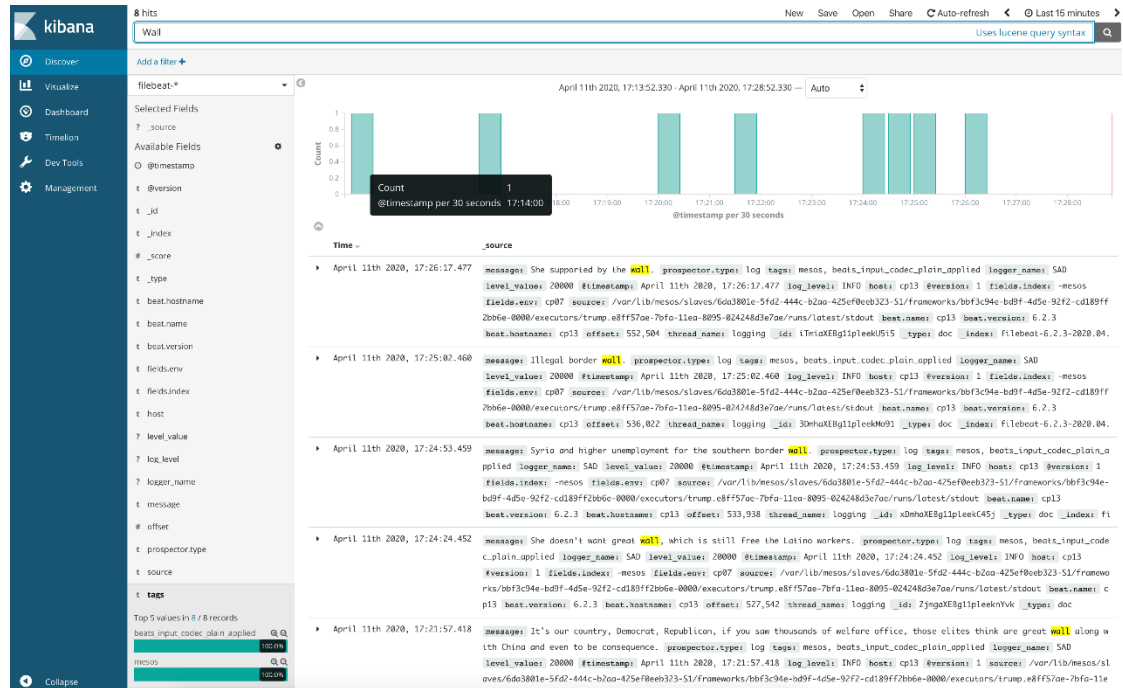


Figure 41. Wall mentions

6 Results

The goal for the thesis was to create and implement an automated logging system, which would be included in the platform provided by Qvantel. The goal was achieved by creating Dockerized versions of the main Elastic Stack components and automating their installations via Ansible playbooks. The components were fully configurable via the Ansible environment files, so the installation to practically any environment was possible. The environment used for testing purposes also served as a demonstration environment, which showcased the behavior from the components, and they were accepted. The logs were coming through from the demo application, and the data was stored successfully in the correct format in Elasticsearch. Audit logs from the operating systems were also found in Elasticsearch. It was concluded that the solution was working and functional enough so that it can be delivered as a part of Qvantel's platform.

7 Conclusions

To summarize the results of this project: it was accepted as a part of Qvantel's platform, and it met all the requirements presented and set by the stakeholders at the time. The solution that was developed is currently used in over 90 different customer environments across the world.

Some of the software components used in this project are out of date, such as the Elastic Stack. The components used in this project are using version 6.2.3, whereas the latest version is 7.6.0. After the initial project was finished, more development towards a more self-maintaining solution has been done. Changes were made while moving from the open source version of the Elastic Stack, to the Basic version of the stack. Some of these changes include: Automatic index lifecycle management, index templating, improving the Logstash filtering to provide metrics for monitoring, collecting more audit events, version upgrades to 7.1.0 and other minor changes.

No research was done towards other solutions that could be used for similar purposes, as Qvantel had already committed to using the software provided by Elastic. Other logging solutions that could have been used include Sentry, which is a platform for managing errors from your apps and software, through logs. With the Elastic Stack being free to use with the Basic license, its usage was easily justifiable.

The development with this solution will most likely never end, as new features are often being released, and customer needs increase the longer such systems like this are in place. The development has continued from the moment this project ended until this day.

References

- Apache Mesos*. N.d. Page on Apache's website. Accessed on 19 April 2020. Retrieved from <http://mesos.apache.org/>
- Auditbeat overview*. N.d. Documentation on Elastic's website. Accessed on 19 April 2020. Retrieved from <https://www.elastic.co/guide/en/beats/auditbeat/current/auditbeat-overview.html>
- Company*. N.d. Page on Qvantel Corporation's website. Accessed on 19 April 2020. Retrieved from <https://www.qvantel.com/company/>
- Consul*. N.d. *Service discovery made easy*. Page on Consul's website. Accessed on 19 April 2020. Retrieved from <https://www.consul.io/discovery.html>
- Data in: documents and indices*. N.d. Documentation on Elastic's website. Accessed on 19 April 2020. Retrieved from <https://www.elastic.co/guide/en/elasticsearch/reference/current/documents-indices.html>
- Distribution Release: Oracle Linux 7.5*. 2018. Article on DistroWatch website. Accessed on 19 April. Retrieved from <https://distrowatch.com/?newsid=10170>
- Docker overview*. N.d. Page on Docker's website. Accessed on 19 April 2020. Retrieved from <https://docs.docker.com/get-started/overview/>
- Elastic Stack: Elasticsearch, Logstash, Kibana*. N.d. Page on Elastic's website. Accessed on 19 April 2020. Retrieved from <https://www.elastic.co/what-is/elk-stack>
- Elasticsearch introduction*. N.d. Documentation on Elastic's website. Accessed on 19 April 2020. Retrieved from <https://www.elastic.co/guide/en/elasticsearch/reference/current/elasticsearch-intro.html>
- Filebeat overview*. N.d. Documentation on Elastic's website. Accessed on 19 April 2020. Retrieved from <https://www.elastic.co/guide/en/beats/filebeat/current/filebeat-overview.html>
- Getting Started*. N.d. Documentation on Elastic's website. Accessed on 19 April 2020. Retrieved from <https://www.elastic.co/guide/en/elasticsearch/reference/6.2/getting-started.html>
- Hou, T. 2020. *IaaS vs PaaS vs SaaS: Examples and How to Differentiate*. Blog post on BigCommerce's website. Accessed on 5.5.2020. Retrieved from <https://www.bigcommerce.com/blog/saas-vs-paas-vs-iaas>
- How Ansible Works*. N.d. Page on Ansible's website. Accessed on 19 April 2020. Retrieved from <https://www.ansible.com/overview/how-ansible-works>
- Information out: search and analyze*. N.d. Documentation on Elastic's website. Accessed on 19 April 2020. Retrieved from <https://www.elastic.co/guide/en/elasticsearch/reference/current/search-analyze.html>

Kerrison, P. 2017. *Pizza as a Service 2.0*. Article on Medium. Accessed on 5.5.2020. Retrieved from <https://medium.com/@pkerrison/pizza-as-a-service-2-0-5085cd4c365e>

Kibana – your window into the Elastic Stack. N.d. Documentation on Elastic's website. Accessed on 19 April 2020. Retrieved from <https://www.elastic.co/guide/en/kibana/7.6/introduction.html>

Libvirt. N.d. Wiki article on Arch Linux's website. Accessed on 19 April 2020. Retrieved from <https://wiki.archlinux.org/index.php/libvirt>

Liu, S. 2019. *Global sought-after database skills for developers 2019*. Graph displaying results of an online survey. Accessed on 19 April 2020. Retrieved from <https://www.statista.com/statistics/793854/worldwide-developer-survey-most-wanted-database/>

Liu, S. 2019. *Global sought-after platform among developers 2019*. Graph displaying results of an online survey. Accessed on 19 April 2020. Retrieved from <https://www.statista.com/statistics/793884/worldwide-developer-survey-most-wanted-platform/>

Liu, S. 2019. *Most utilized frameworks among developers worldwide 2019*. Graph displaying results of an online survey. Accessed on 19 April 2020. Retrieved from <https://www.statista.com/statistics/793840/worldwide-developer-survey-most-used-frameworks/>

Logstash Introduction. N.d. Documentation on Elastic's website. Accessed on 19 April 2020. Retrieved from <https://www.elastic.co/guide/en/logstash/7.6/introduction.html>

Lokien keräys ja käyttö: ohje lokitietojen tallentamiseen ja hyödyntämiseen [Log collection and usage: guide on storing and using logs]. 2016. PDF document on National Cyber Security Centre's website. Accessed on 19 April 2020. Retrieved from <https://www.kyberturvallisuuskeskus.fi/sites/default/files/media/file/Lokitusohje.pdf>

Marathon: A container orchestration platform for Mesos and DC/OS. N.d. Page on GitHub. Accessed on 19 April 2020. Retrieved from <https://mesosphere.github.io/marathon/>

Näin keräät ja käytät lokitietoja [How to collect and use logs]. 2019. Page on National Cyber Security Centre's website. Accessed on 19 April 2020. Retrieved from <https://www.kyberturvallisuuskeskus.fi/fi/ajankohtaista/ohjeet-ja-oppaat/nain-keraat-ja-kaytat-lokitietoja>

Scalability and resilience: clusters, nodes, and shards. N.d. Documentation on Elastic's website. Accessed on 19 April 2020. Retrieved from <https://www.elastic.co/guide/en/elasticsearch/reference/current/scalability.html>

Virsh(1): Management User Interface. N.d. Linux man page. Accessed on 19 April 2020. Retrieved from <https://linux.die.net/man/1/virsh>

Welcome to Apache ZooKeeper™. N.d. Page on Apache Zookeeper website. Accessed on 19 April 2020. Retrieved from <https://zookeeper.apache.org/>

What are Linux Logs? Code Examples, Tutorials & More. 2017. Page on Stackify's website. Accessed on 19 April 2020. Retrieved from <https://stackify.com/linux-logs/>

What is a Container? N.d. Page on Docker's website. Accessed on 19 April 2020. Retrieved from <https://www.docker.com/resources/what-container>

What is Docker? N.d. Page on Opensource.com's website. Accessed on 19 April 2020. Retrieved from <https://opensource.com/resources/what-docker>

What is IT Automation? N.d. Page on VMware's website. Accessed on 26 April 2020. Retrieved from <https://www.vmware.com/topics/glossary/content/it-automation>

What's IT automation? N.d. Page on Red Hat's website. Accessed on 26 April 2020. Retrieved from <https://www.redhat.com/en/topics/automation/whats-it-automation>

Appendices

Appendix 1. Truncated log of the Ansible playbook run of orchestration node installation

```
QFINM234:thesis lmoilanen$ Ansible-playbook -u lmoilanen -i inventory/hosts -k orchestration.yml
SSH password:
```

```
=====
=====
Docker-consul : Install dnsmasq -----
-----
----- 26.73s
Docker-Mesos : start Mesos-master -----
-----
----- 11.82s
Docker-consul : Enable firewall rules for service -----
-----
----- 2.93s
Docker-consul : Apply firewalld service configuration file -----
-----
----- 2.76s
Docker : Create Docker networks -----
-----
----- 2.46s
Gathering Facts -----
-----
----- 2.40s
Docker : Create Docker networks -----
-----
----- 2.35s
Docker-Marathon : start Marathon -----
-----
----- 2.34s
beats : enable and start beats -----
-----
----- 2.23s
Gathering Facts -----
-----
----- 1.97s
Docker : Install Docker python library -----
-----
----- 1.88s
Docker-consul : apply consul configuration -----
-----
----- 1.79s
```

```

ensure firewalld is installed -----
-----
----- 1.77s
Docker-consul : start consul server -----
-----
----- 1.77s
Docker-zookeeper : Apply Zookeeper configuration -----
-----
----- 1.76s
Docker : Install Docker python library -----
-----
----- 1.75s
Docker-Mesos : Apply Mesos configuration -----
-----
----- 1.74s
beats : auditbeat configuration -----
-----
----- 1.74s
Docker : Uninstall Docker -----
-----
----- 1.74s
Docker-Mesos : Apply firewalld service configuration file -----
-----
----- 1.72s

```

Appendix 2. Truncated log of the application node installation

```

QFINM234:thesis lmoilanen$ Ansible-playbook -u lmoilanen -i in-
ventory/hosts -k application.yml

```

```

PLAY RECAP *
cp09.lab.qvantel.net      : ok=1    changed=0    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
cp11.lab.qvantel.net      : ok=1    changed=0    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
cp13.lab.qvantel.net      : ok=63   changed=2    unreachable=0
failed=0    skipped=9    rescued=0    ignored=0
cp15.lab.qvantel.net      : ok=62   changed=2    unreachable=0
failed=0    skipped=9    rescued=0    ignored=0
cp16.lab.qvantel.net      : ok=1    changed=0    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0

```

```

Saturday 11 April 2020 17:52:40 +0300 (0:00:00.477)
0:01:58.165 *****
=====
=====

```

```

Docker-consul : Install dnsmasq -----
-----
----- 20.60s
Docker : Apply Docker systemd overrides -----
-----

```

```
-----  
----- 3.35s  
Docker-consul : Enable firewall rules for service -----  
-----  
----- 2.89s  
Docker : Create Docker networks -----  
-----  
----- 2.66s  
Docker-consul : Apply firewalld service configuration file -----  
-----  
----- 2.65s  
Docker : Start Docker -----  
-----  
----- 2.22s  
Gathering Facts -----  
-----  
----- 2.21s  
Gathering Facts -----  
-----  
----- 2.06s  
beats : enable and start beats -----  
-----  
----- 2.02s  
Docker-Mesos : Install Mesos -----  
-----  
----- 1.92s  
Docker-Mesos : Apply firewalld service configuration file -----  
-----  
----- 1.91s  
Docker : Install Docker python library -----  
-----  
----- 1.86s  
Docker-consul : consul agent ip in file -----  
-----  
----- 1.76s  
Docker-consul : apply consul configuration -----  
-----  
----- 1.75s  
Docker-consul : start consul agent -----  
-----  
----- 1.74s  
beats : auditbeat configuration -----  
-----  
----- 1.74s  
Docker-Mesos : Apply Zookeeper configuration -----  
-----
```



```

-----
----- 1.72s
Docker-Mesos : Apply Mesos environment variables -----
-----
----- 1.71s
Docker-Mesos : Apply Mesos modules configuration -----
-----
----- 1.71s
beats : Install Auditbeat -----
-----
----- 1.71s

```

Appendix 3. Truncated log of the log node installation

```

QFINM234:thesis lmoilanen$ Ansible-playbook -u lmoilanen -i in-
ventory/hosts -k log.yml

```

```

PLAY RECAP *

```

```

cp09.lab.qvantel.net      : ok=1    changed=0    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
cp11.lab.qvantel.net      : ok=1    changed=0    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
cp16.lab.qvantel.net      : ok=72   changed=2    unreachable=0
failed=0    skipped=11   rescued=0    ignored=0

```

```

Saturday 11 April 2020 17:50:05 +0300 (0:00:00.476)
0:02:01.260 *****

```

```

=====
=====
Docker-consul : Install dnsmasq -----
-----
----- 19.25s
Docker-consul : Enable firewall rules for service -----
-----
----- 2.80s
Docker-consul : Apply firewall service configuration file -----
-----
----- 2.66s
Docker : Create Docker networks -----
-----
----- 2.47s
Gathering Facts -----
-----
----- 2.28s
beats : enable and start beats -----
-----
----- 2.08s
Docker-kibana : Apply firewall service configuration file -----
-----

```

```
-----  
----- 1.94s  
Gathering Facts -----  
-----  
----- 1.82s  
Docker : Install Docker python library -----  
-----  
----- 1.78s  
Docker-kibana : Enable firewall rules for service -----  
-----  
----- 1.78s  
Docker-elasticsearch : Apply Elasticsearch configuration -----  
-----  
----- 1.74s  
Docker-elasticsearch : Apply firewalld service configuration file  
-----  
----- 1.74s  
Docker-kibana : create configuration -----  
-----  
----- 1.72s  
Docker-logstash : Apply firewalld service configuration file -----  
-----  
----- 1.66s  
beats : filebeat for Mesos stderr -----  
-----  
----- 1.64s  
Docker-consul : start consul agent -----  
-----  
----- 1.63s  
ensure firewalld is installed -----  
-----  
----- 1.63s  
beats : filebeat for containers -----  
-----  
----- 1.62s  
Docker-logstash : Apply Logstash pipeline config files -----  
-----  
----- 1.61s  
Docker-consul : apply consul configuration -----  
-----  
----- 1.61s  
QFINM234:thesis lmoilanen$
```