

## BACHELORARBEIT

# Umsetzung von Logging-Richtlinien und Einrichtung eines zentralisierten Logging-Servers für das CFT Portale der Kassenärztlichen Vereinigung Westfalen-Lippe

*Kevin Bollich*  
geboren am 19.04.1997  
Matr.-Nr.: 7102160

An der Fachhochschule Dortmund im Fachbereich Informatik erstellte  
Bachelorarbeit  
im Studiengang Software- und Systemtechnik Dual - Vertiefungsrichtung  
Softwaretechnik

zur Erlangung des akademischen Grades  
Bachelor of Science  
B. Sc.

**Betreuung durch:**  
Prof. Dr. Martin Hirsch

4. Januar 2021

# Inhaltsverzeichnis

<b>1. Einführung</b>	<b>1</b>
1.1. Motivation . . . . .	1
1.2. Problemstellung . . . . .	1
1.3. Zielsetzung . . . . .	2
1.4. Vorgehensweise . . . . .	2
<b>2. Evaluation von Log-Management Tools</b>	<b>4</b>
2.1. Anforderungen an die Log-Management Tools . . . . .	4
2.2. Log-Management Tools . . . . .	6
2.2.1. Elastic Stack . . . . .	6
2.2.2. Graylog . . . . .	7
2.2.3. Sematext . . . . .	9
2.2.4. Fluentd . . . . .	10
2.3. Vergleich der Tools . . . . .	11
2.3.1. Graylog vs. Elastic Stack . . . . .	12
2.3.2. Sematext vs. Elastic Stack . . . . .	12
2.3.3. Fluentd vs. Logstash . . . . .	12
2.4. Zusammenfassung . . . . .	13
<b>3. Einrichtung des zentralisierten Logging Servers</b>	<b>14</b>
3.1. RedHat Server . . . . .	14
3.2. Elasticsearch . . . . .	15
3.3. Kibana . . . . .	17
3.3.1. Index Pattern . . . . .	17
3.3.2. Dashboards . . . . .	18
3.3.3. Kibana Query Language (KQL) . . . . .	18
3.4. Logstash . . . . .	19
3.5. Filebeat . . . . .	19
3.6. Struktur der Logs . . . . .	20
<b>4. Umsetzung der Richtlinien</b>	<b>24</b>
4.1. Vierteljahreserklärung . . . . .	24
4.2. Konfiguration von NLog . . . . .	26
4.3. Aufbau der Logs . . . . .	28

4.4. Abgleich der Richtlinien . . . . .	29
<b>A. Anhang</b>	<b>32</b>
A.1. Vergleich Graylog Open Source vs. Enterprise . . . . .	32
A.2. Eidestattliche Erklärung . . . . .	32

# **Abbildungsverzeichnis**

2.1. Fluentd vorher und nachher Vergleich [Pro] . . . . .	11
3.1. Stages mit Logs . . . . .	22
4.1. Vierteljahreserklärung . . . . .	25
A.1. Vergleich Graylog Open Source vs. Enterprise . . . . .	32

# **1. Einführung**

## **1.1. Motivation**

Das CFT Portale der Kassenärztlichen Vereinigung Westfalen-Lippe (KVWL) verwaltet eine hohe Anzahl an Applikationen, bei denen regelmäßig neue Funktionen hinzukommen. Bei der stetigen Weiterentwicklung können während der Laufzeit Fehler auftreten, deren Herkunft nicht immer eindeutig ist. Damit die Herkunft solcher Fehler erkannt werden kann, sollten bestimmte Laufzeitinformationen geloggt werden. Da derzeit keine klare Struktur im Logging erkennbar ist, ist das Bugtracking im CFT Portale sehr zeitaufwendig. Der Grund dafür liegt hauptsächlich an der redundanten Serververteilung und dem unstrukturierten Logging.

## **1.2. Problemstellung**

Im CFT Portale müssen die Entwickler regelmäßig die Ursachen von aufgetretenen Fehlern analysieren. Dabei sieht der Prozess folgendermaßen aus:

Jede Anwendung ist auf zwei redundanten Servern installiert und schreibt auf dem jeweiligen Server ihre Logs. Damit die Entwickler herausfinden können, wo das entsprechende Log geschrieben wurde, muss auf beiden Servern manuell nach dem Fehler gesucht werden. Da ein Fehler nicht immer sofort nach Auftreten gemeldet wird und die Software trotz des Fehlers weiter läuft, steigt die Menge an geschriebenen Logs. Den Fehler in den Logdateien zu finden, kann durch die fehlende Möglichkeit des Filterns sehr zeitaufwendig werden.

Eine weitere Herausforderung bei der Fehlersuche liegt in den unstrukturierten Informationen in den Logdateien.

Daraus leiten sich folgende Forschungsfragen ab:

- Mit welchem Log-Management-Tool ist ein an die Probleme des CFT Portale angepasstes zentralisiertes Logging möglich?
- Wie kann ein zentralisierter Logging-Server eingerichtet werden?
- Können die Logging-Richtlinien aus der Projektarbeit in der Praxis umgesetzt werden? [Bol]

### **1.3. Zielsetzung**

Ziel dieser Bachelorarbeit ist es, die in der Projektarbeit definierten Logging-Richtlinien anhand der Software „Vierteljahreserklärung“ durchzuführen. Außerdem soll eine Evaluierung von Log-Management Tools erfolgen, damit herausgefunden werden kann, ob der in der Projektarbeit erwähnte Elastic-Stack die beste Lösung für das CFT Portale ist, um ein zentralisiertes Logging einzurichten. Wenn die Entscheidung über das Log-Management Tool getroffen wurde, soll ein zentralisiertes Logging mit dem Log-Management Tool umgesetzt werden.

### **1.4. Vorgehensweise**

Zu Beginn der Bachelorarbeit erfolgt eine Evaluation von Log-Management-Tools. Mithilfe der Evaluation soll ein passendes Tool identifiziert werden, dass eine effiziente zentralisierte Lösung für das CFT Portale ermöglicht. Bevor dies geschieht, müssen noch die Anforderungen an das Tool aufgestellt werden. Dies geschieht in Absprache mit dem CFT Portale. Nachdem ein Tool identifiziert wurde, soll ein zentralisierter Logging-Server eingerichtet werden. Dieser soll in Zukunft die erstellten Logs sammeln, anzeigen und analysieren. Anschließend sollen in der Applikationen „Vierteljahreserklärung“ alle in der Projektarbeit definierten Richtlinien umgesetzt werden. Zum Schluss wird ein Fazit zum Verlauf der Bachelorarbeit gezogen. Dabei werden die Ergebnisse der Arbeit noch einmal vorgestellt und bewertet.

Vorläufige Gliederung:

- Einführung
- Evaluation von Log-Management Tools
- Einrichten eines zentralisierten Logging-Server
- Umsetzen der Richtlinien
- Fazit

## **2. Evaluation von Log-Management Tools**

In der vorherigen Projektarbeit wurden für das CFT Portale Richtlinien definiert, die in dieser Bachelorarbeit praktisch umgesetzt werden sollen. Eine dieser Richtlinien war die Nutzung von einem zentralisierten Logging-Server mithilfe des Elastic Stack. Jedoch wurden in der Projektarbeit keine weiteren Tools herangezogen, um zu prüfen, ob der Elastic Stack die beste Alternative ist.

In diesem Kapitel werden unterschiedliche Tools, die für das Log-Management genutzt werden können, evaluiert. Das Ziel dieser Evaluation ist es, zu prüfen, ob es eine bessere Alternative für eine zentralisierte Logging Lösung gibt als den Elastic Stack. Dafür werden Tools evaluiert, die den kompletten Elastic Stack ersetzen können, aber auch Tools, die einzelne Komponenten austauschen können.

Das CFT Portale wäre in der Lage, weitere Kosten für ein Tool auf sich zu nehmen, sollte es dem Team die Arbeit erleichtern können. Daher werden Open-Source und lizenzpflchtige Tools in dieser Evaluation betrachtet. Sollten jedoch zwei Tools gleichermaßen die Anforderungen erfüllen und eines der Tools Open-Source sein, dann wird sich für das Open-Source-Tool entschieden, um Kosten zu sparen.

Damit eine Evaluation erfolgen kann, müssen Anforderungen aufgestellt werden. Die Anforderungen sollen dabei helfen, eine Entscheidung bezüglich der Tools treffen zu können. Tools, die diese Anforderungen nicht erfüllen können, werden nicht weiter betrachtet. Im nächsten Abschnitt werden die Anforderungen definiert.

### **2.1. Anforderungen an die Log-Management Tools**

In diesem Abschnitt werden Anforderungen für die zentralisierten Logging Tools definiert. Die Anforderungen helfen bei der Entscheidung, ein passendes Tool für

das CFT Portale auszuwählen. Daher wurden in Absprache mit dem Team einige Anforderungen definiert, die das zukünftige Tool haben sollte. Für das CFT Portale spielt Wartung eine wichtige Rolle, daher werden einige Anforderungen sich auf den Wartungsaufwand beziehen.

Da das KV-Netz sicherheitstechnisch stark abgeschirmt ist, kommt eine Cloud-Lösung nicht in Frage. Das bedeutet, dass das Tool eine Lösung bieten muss, die auf den Servern der KV laufen sollen.

Durch die Menge an Anwendungen, die das CFT Portale betreuen muss, ist es wichtig, dass das Tool die einzelnen Logs entweder von den unterschiedlichen Maschinen selbst einsammeln kann oder das Senden der Logs möglich ist. Das Installieren weiterer Log-Agenten, die für das Senden der Logs zuständig sind, sollten vermieden werden, da sonst weiterer Wartungsaufwand entstehen würde.

Da das CFT Portale keine Datenbanken betreuen muss, ist der aktuelle Wissensstand des Teams daher eher allgemein vorhanden. Denn das Team übernimmt in der Regel die Entwicklung von Provider-hosed Apps im SharePoint Umfeld. Damit das Team also erfolgreich mit den Datenbanken arbeiten kann, müssen Schulungen absolviert werden. Daher ist es wichtig, dass das ausgewählte Tool eine Möglichkeit bietet, die Logs zu speichern.

Ein wichtiger Punkt ist die Analyse und Anzeige von Logs. Das heißt, es soll eine Oberfläche vorhanden sein, die intuitiv benutzt werden kann, um die Logs anzusehen und zu analysieren. Jedoch sollte in dem Tool auch die Möglichkeit bestehen, Logs zu filtern und zu durchsuchen.

Im CFT Portale sind Linux- und Windows Server im Betrieb. Jedoch möchte das Team das Tool gerne auf einem Linux-Server installieren, da dort das Update von neuen Versionen einfacher funktioniert und der Wissensstand des Teams da komplett gegeben ist.

Diese Vorgaben wurden im gemeinsamen Gespräch mit den Entwicklern des CFT Portale definiert. Mithilfe dieser Vorgaben soll ein passendes Log-Management Tool für das CFT Portale ausgesucht werden. Zur Veranschaulichung der Vorgaben folgt eine Aufzählung:

- Selbstorganisierte Lösung
- Einsammeln von Logs aus unterschiedlichen Anwendungen
- Speicherung von Logs ohne externe Datenbank
- Anzeige und Analyse von Logs
- Filtern und durchsuchen von Logs
- Installation auf Linux Server

## 2.2. Log-Management Tools

In der Projektarbeit wurde der Elastic Stack in seiner Funktionsweise und dessen Möglichkeiten detailliert vorgestellt. Jedoch wird in diesem Kapitel auf die wichtigen Inhalte des Elastic Stack eingegangen, um eine Bewertung der Tools zu ermöglichen. Alle Tools werden hier mit all ihren Vor- und Nachteilen vorgestellt. Eine Bewertung der Tools wird in Kapitel 2.3 durchgeführt. Bevor die Tools evaluiert werden können, musste eine Vorauswahl getroffen werden. Dafür wurden viele Tools betrachtet und nach den Anforderungen in Kapitel 2.1 ausgewählt.

### 2.2.1. Elastic Stack

Der Elastic Stack ist eine Sammlung von Tools, die unterschiedliche Aufgaben erledigen. Die Tools sind Elasticsearch, Kibana, Logstash und Beat. Mit diesen Tools können unterschiedliche Aufgaben gelöst werden. Im Rahmen dieser Bachelorarbeit ist jedoch nur das Log-Management interessant. Daher wird hier nur auf das Log-Management eingegangen. Die Komponente Elasticsearch aus dem Elastic Stack ist laut dem Ranking der Searchengines von **DB-Engines** auf Platz 1. [DB-] Das bedeutet, dass die Effizienz der Suche sichergestellt sein kann. Die Aufgaben der einzelnen Tools sind:

- Elasticsearch - Speichern und Suchen
- Kibana - Analyse und Anzeige

- Logstash - Parsen und Weiterleiten
- Beat - Senden von Daten

Die Voraussetzungen für die Nutzung des Elastic Stack sind niedrig. Denn die einzelnen Komponenten des Elastic Stack können auf den aktuell vorhanden Betriebssystemen installiert werden. Zusätzlich zu den Komponenten des Elastic Stack wird nur noch eine Installation von Java benötigt. Java wird jedoch automatisch bei der Installation von Elasticsearch mitinstalliert. Elasticsearch wurde in Java geschrieben und muss daher in einer JVM (Java virtual machine) laufen. [Elab]

Die Kosten für den Elastic Stack können unterschiedlich ausfallen, denn die Komponenten selber sind Open Source und damit dann auch kostenlos. Es ist jedoch möglich fertig konfigurierte Lösungen zu kaufen. Einerseits gibt es eine Cloud Lösung, die nach dem Kauf sofort genutzt werden kann ohne jegliche Installation, oder auch vorgefertigte Lösungen die dann noch auf den eigenen Systemen installiert werden müssen. Der Vorteil von der gekauften Version, ist der niedrigere Wartungsaufwand der gefordert ist und der Support der mitangeboten wird. Die Cloud Lösung kommt jedoch nicht infrage, wie im Kapitel 2.1 beschrieben wurde. Das günstigste Modell des Elastic Stack kostet 16 \$ im Monat. Jedoch bietet dieser nur Support für den Elastic Cloud dienst. Damit der Support richtig genutzt werden kann muss mindestens die Gold Version für 19\$ im Monat gekauft werden. Mit dem Kauf steigen auch die Funktionalitäten, die mit dem Elastic Stack möglich sind. Zum Beispiel würde das Reporting bei der Gold-Stufe hinzukommen. [Elaf]

### 2.2.2. Graylog

Graylog ist ein Open-Source Log-Management Tool. Dessen Motto ist:

„less cost, more performance“[Grab]

Das Tool setzt auf Performance. Die Funktionalitäten des Tools beziehen sich auf das Sammeln, verbessern, Speichern und der Analyse von Logs. In Graylog kann man eigene Dashboards erstellen und individuell anpassen. Das Dashboard wird mithilfe von Suchabfragen definiert. Damit nicht jeder Mitarbeiter sich mit

den Suchabfragen beschäftigen muss, können die Dashboards untereinander geteilt werden. Graylog bietet zusätzlich vordefinierte Dashboards an, die genutzt werden können.

Mithilfe von Graylog können unmengen an Logs gespeichert werden. Daher ist die Suche in großen Datenmengen essenziell. In Graylog werden die Logs beim Speichern indiziert, um eine effiziente Suche zu ermöglichen. Die Daten werden beim Speichern geprüft. Bei der Prüfung wird die Struktur genauer untersucht, um festzustellen, ob die Struktur in Ordnung ist. Wenn die Struktur nicht in Ordnung ist, wird sie verbessert.

Die Architektur von Graylog ermöglicht eine multi-threaded Suche. Jede Suche nutzt dabei mehrere Prozessoren, um möglichst effizient zu sein. Die Suche in Graylog ist dabei einfach aufgebaut. Einfache Boolean Operationen werden für die Suche genutzt. Die dazu benötigten Felder werden durch einfaches Klicken ausgewählt. Damit muss keine neue Suchsprache erlernt werden und kann von nicht Fachpersonal genutzt werden. [Grab]

Wenn mehr benötigt wird, als die Open Source Version anbietet, dann kann Graylog als Enterprise Variante gekauft werden. Bei der Enterprise Variante werden Support und zusätzliche Funktionen angeboten. Der genaue Vergleich der beiden Varianten kann in Abbildung A.1 im Anhang A.1 gesehen werden. Zu dem Support gehört Hilfe zu allen Graylog bezogenen Fragen, jedoch bietet Graylog zusätzlich Support für Elasticsearch, MongoDB und Oracle Java SE 8 (oder OpenJDK 8). Sie bieten diesen Support an, weil Graylog diese Produkte benötigt, um laufen zu können. Das bedeutet, dass diese Produkte zusätzlich auf der zu installierenden Maschine installiert werden. Zum Thema der zu installierenden Maschine ist wichtig zu beachten, dass Graylog nur auf Linux-basierten Betriebssystemen installiert werden kann.

Die Graylog Enterprise variante kann bis zu 5 GB/Tag kostenlos erworben werden. Für so kleine Datenmengen ist es daher ratsam direkt auf die Enterprise Variante zu setzen. [Graa]

Damit Graylog funktionieren kann, muss weitere Software auf der Maschine installiert werden. Das bewirkt, dass der Aufwand für das Updaten des Tools zu Pro-

blemen führen kann. Beim Betrieb von Graylog muss darauf geachtet werden, dass die Versionen der Tools passen. Das sorgt dafür, dass der Wartungsaufwand sehr hoch ist.

Abließend können die folgenden Vor- und Nachteile für Graylog zusammengefasst werden:

- Vorteile:
  - Open Source (Enterprise auch möglich)
  - Sehr performant durch multi-threaded Suche und indizierung
  - Speichern großer Datenmengen möglich
- Nachteile:
  - Notwendige Installation von weiterer Software
  - Durch extra Software Updates Problematisch
  - Installation nur auf Linux Maschinen

### 2.2.3. Sematext

Sematext ist ein kostenpflichtiges Log-Management und Infrastructure Monitoring Tool. Das Tool wird als SaaS (Software as a Service) angeboten, jedoch gibt es zusätzlich eine Enterprise Variante, die das Laufen innerhalb der eigenen Infrastruktur ermöglicht. Das heißt, dass eine Kopie der Cloud Version von Sematext auf der eigenen Infrastruktur läuft. Sematext unterstützt im Vergleich zu anderen Tools nicht nur das Log-Management, sondern diese Funktionen:

- Infrastructure Monitoring
- Application Performance Management (APM)
- Log Management
- Real User Monitoring

Sematext nutzt für das Log-Management Elasticsearch und Kibana. Das heißt, beim Kauf von Sematext erhält man eine fertige Lösung des Elastic Stack. Der Vorteil besteht hier in der Wartung, die von Sematext selbst übernommen wird. Ein Vor- und Nachteil besteht beim Log-SHIPPER, der zusätzlich noch installiert werden muss. Dieser wird nicht von Sematext bestimmt. Daher kann da flexibel entschieden werden, welcher Log Shipper am besten geeignet ist. Jedoch kann dadurch auch nicht sichergestellt werden, dass der Log-Shipper mit dem Tool gut funktioniert. mithilfe der Filter Funktion können mit Sematext Benachrichtigungen definiert werden, die beim Auftreten einer Nachricht senden. [Semb]

Damit Sematext in der eigenen Infrastruktur laufen kann muss folgende Software installiert sein:

- Docker
- Kubernetes
- Helm

Sematext Enterprise wird als Helm Chart angeboten. Helm Chart ist ein Paket für Kubernetes. Das Helm Chart beinhaltet alles, was nötig ist, um Sematext Enterprise nutzen zu können. Da Sematext Enterprise, nur als Helm Chart angeboten wird, muss die Installation über Kubernetes erfolgen. [Sema]

#### **2.2.4. Fluentd**

Fluentd ist ein Open Source data Collector, der es Anwendern ermöglichen soll, alles zu loggen. Das heißt Fluentd sammelt Daten und leitet sie in gewünschter Form weiter zum Ziel. Also ist Fluentd kein Tool, um zentralisiertes Logging zu ermöglichen, sondern nur ein Tool, das beim zentralisierten Logging behilflich ist. Mithilfe von Fluentd können Datenströme einheitlich und verständlich ablaufen. Ein Beispiel dafür ist in Abbildung 2.1 zu sehen. Die Datenströme im „Before Fluentd“ Teil sind nicht zu erkennen, wobei mit der Nutzung von Fluentd die Datenströme einheitlich über Fluentd laufen. [Flu]

Fluentd versucht, soweit es möglich ist, die erhaltenen Daten zu strukturieren. Dabei werden die Daten im JSON-Format gespeichert. Somit kann Fluentd die Daten einheitlich verarbeiten. Zum Verarbeiten gehören das Sammeln, Filtern, Puffern und die Weitergabe der Logs über verschiedene Quellen und Ziele hinweg. Für weitere Funktionalitäten können Lösungen von der Community genutzt werden. Das wird durch die steckbare Architektur von Fluentd ermöglicht. Fluentd ist in einer Kombination von C und Ruby entwickelt worden. Deswegen benötigt es nur wenig System Ressourcen, um zu funktionieren. Die Standard Version von Fluentd ohne weitere Komponenten benötigt ungefähr 30-40 MB Speicher. [Pro]

Die größten Vorteile von Fluentd sind die Menge an Plugins die verfügbar und die Performance. Außerdem verbraucht Fluentd sehr wenig Speicher. Der größte Nachteil ist der Zwang nach strukturierten Daten, damit ist gemeint, dass es nicht so flexibel möglich ist, zu entscheiden, wie die Logs auszusehen haben. Fluentd formatiert die erhaltenen Daten immer in JSON-Format.

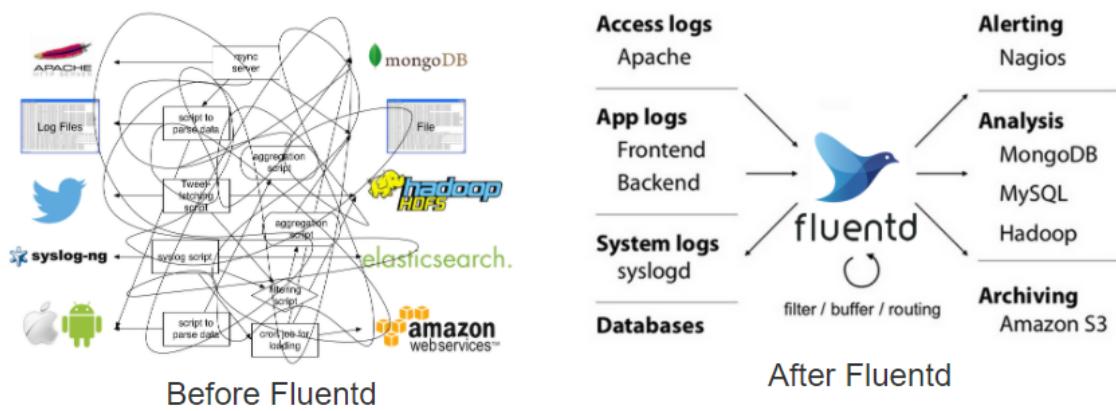


Abbildung 2.1.: Fluentd vorher und nachher Vergleich [Pro]

## 2.3. Vergleich der Tools

In diesem Kapitel erfolgt ein Vergleich der einzeln vorgestellten Tools mit dem Elastic Stack. So soll geprüft werden, welche Vor- und Nachteile die Tools im Vergleich

zum Elastic Stack mitbringen. Anschließend soll mit Hilfe der in Kapitel 2.1 definierten Anforderungen entschieden werden, ob ein Tool für das CFT Portale besser geeignet ist als der Elastic Stack.

### **2.3.1. Graylog vs. Elastic Stack**

Graylog ist wie der Elastic Stack ein Open Source Tool, um zentralisiertes Logging zu ermöglichen. Dabei setzt Graylog bei der Suche auf Elasticsearch. Dadurch ist die Suche bei beiden Produkten identisch. Im Vergleich zum Elastic Stack bietet Graylog keinen weiteren Vorteil, den Elastic Stack nicht hat. Jedoch bringt Graylog einen Nachteil mit sich, die der Elastic Stack nicht hat. Das ist die Installation weiterer Software, die Probleme beim zukünftigen Updaten bringen können. Der Elastic Stack benötigt zum Laufen nur die Tools vom Elastic Stack.

Beim Vergleich von Graylog und dem Elastic Stack fällt auf, dass der Elastic Stack insgesamt die bessere Entscheidung darstellt.

### **2.3.2. Sematext vs. Elastic Stack**

Sematext bietet eine vollständig installierte Version vom Elastic Stack an. Das heißt, die Vor- und Nachteile sind sehr identisch, da sie an Funktionalitäten das Gleiche bieten. Jedoch ist der Vorteil von Sematext, dass der Support mit angeboten wird und das konfigurieren und einrichten nicht so komplex fällt wie beim Elastick Stack selber. Ein großes Problem von Sematext stellt sich bei der Installation der Enterprise Variante auf. Denn die Installation erfolgt über Kubernetes und Docker Container. Im CFT Portale besitzt kein Teammitglied das nötige Wissen, um die Installation und Wartung zu übernehmen. Daher kommt die Nutzung von Sematext nicht infrage.

### **2.3.3. Fluentd vs. Logstash**

Da Fluentd kein Tool ist, um zentralisiertes Logging zu ermöglichen, sondern nur eine Middleware, die beim Senden der Logs behilflich sein kann, wird das Tool mit Logstash alleine verglichen. Im Vergleich zu Logstash ist Fluentd sehr viel performanter und verbraucht weniger Speicherplatz. Jedoch sind die zwei Vorteile nicht

relevant für den Einsatz im CFT Portale. Ressourcen sind genügend vorhanden. Da Logstash mit den Komponenten des Elastic Stack zusammen Entwickelt wurde, ist der Wechsel zu Fluentd im Szenario des CFT Portale nicht sehr sinnvoll.

## 2.4. Zusammenfassung

Im Rahmen dieser Evaluation wurden drei Tools zusätzlich zum Elastic Stack genauer untersucht. Das Ziel der Evaluation war die Antwort auf die Frage: „Ist der Elastic Stack die beste Wahl für das CFT Portale?“ Anhand der Anforderungen wurden eine Vielzahl an Tools untersucht. Dabei wurden viele der Tools durch die Anforderungen direkt ausgeschlossen. Es wurden drei Tools gefunden, die den Anforderungen des CFT Portale am ehesten passten. Während der Evaluation wurden die Tools mit dem Elastic Stack verglichen. Dabei ist aufgefallen, dass der Elastic Stack die beste Wahl für einen zentralisierten Logging Server für das CFT Portale bietet.

# **3. Einrichtung des zentralisierten Logging Servers**

Die im vorherigen Kapitel durchgeführte Evaluation kam zum entschluss, dass der Elastic Stack die Anforderungen des CFT Portale am besten erfüllen kann. Daher wird in diesem Kapitel ein zentralisierter Logging-Server mit dem Elastic Stack installiert und konfiguriert. Dafür wird zu Beginn ein Server benötigt, auf dem die einzelnen Komponenten installiert werden können. Anschließend müssen die einzelnen Tools des Elastic Stack installiert und konfiguriert werden.

Dieses Kapitel ist wie folgt aufgebaut:

- RedHat Server
- Elasticsearch
- Kibana
- Logstash
- Filebeat

## **3.1. RedHat Server**

Damit die einzelnen Komponenten des Elastic Stack installiert werden können, musste ein Server beantragt werden. Dafür wurde ein Auftrag an das CFT Infrastruktur der KVWL gesendet, um so einen Server zu erhalten. Wie schon in Kapitel 2.3 geschrieben wurde, werden im CFT Portale Linux Server verwendet. In der Regel werden RedHat Server genutzt. So ist es auch bei dem Server für das zentralisierte Logging. Es ist ein RedHat Server mit der Version 7.9.

Damit die Kommunikation zum Server erfolgen kann, müssen einige Ports freigeschaltet werden. Denn Elasticsearch und Kibana nutzen die Ports 9200 und 5601. Ohne Freischalten der Ports würde die Kommunikation von anderen Systemen nicht funktionieren. Die Portfreischaltung erfolgte mit diesen Befehlen:

```
1 $ sudo firewall-cmd --zone=public --add-port=9200/tcp --permanent  
2 $ sudo firewall-cmd --zone=public --add-port=5601/tcp --permanent
```

Bevor die Installation der Elastic Stack Komponenten erfolgen konnte, musste festgelegt werden wie die Installation durchgeführt werden soll. Denn die KVWL nutzt *Satellite* um die installierten Packages zu verwalten. Aktuell sind die Komponenten des Elastic Stack nicht im Satellite eingebunden. Daher muss geklärt werden wie die Komponenten installiert werden dürfen. Um den Organisatorischen Aufwand möglichst gering zu halten, wurde in Absprache mit den Administratoren festgelegt, dass eine Manuelle Installation derzeit durchgeführt werden soll. Im Anschluss an diese Bachelorarbeit sollen die Pakete, sofern das Ergebnis zufriedenstellende Ergebnisse liefert, ins Satellite eingebunden werden, um den zukünftigen Wartungsaufwand gering zu halten.

## 3.2. Elasticsearch

Die erste zu installierende Komponente des Elastic Stack ist Elasticsearch. Bei der Installation wurde auf die Anleitung vom Hersteller zurückgegriffen. [Elac] Bei der Installation war es wichtig zu beachten die richtigen Packages runterzuladen. Denn in der Anleitung wird von der Kommerziellen Lösung gesprochen, die kostenlos getestet werden kann. Die richtigen Package Versionen beinhalten ein *oss* im Namen.

Aus Sicherheitsgründen hat der Server keinen Zugriff auf das Internet. Daher müssen die Installationspakete Manuell vom Entwickler Rechner runtergeladen werden und anschließend zum Server überkopiert werden. Da es sich um einen RedHat Server handelt, werden *rpm* Pakete runtergeladen und installiert. Der Befehl der genutzt wurde um das Package runterzuladen ist:

```
1 wget https://artifacts.elastic.co/downloads/elasticsearch/  
      elasticsearch-oss-7.10.0-x86_64.rpm
```

Nachdem der Download fertiggestellt wurde, musste das *rpm* package auf den Red-Hat Server kopiert werden:

```
1 pscp -P 22 elasticsearch-oss-7.10.0-x86_64.rpm bollrich@DOT-RH-ZLOG1  
     .doms.kwvl.de:/home/bollrich
```

Anschließend musste das *rpm* Paket installiert werden. Nach der Installation musste eingestellt werden, dass der Elasticsearch Service automatisch startet wenn der Server hochfährt.

```
1 sudo rpm --install elasticsearch-oss-7.10.0-x86_64.rpm  
2 sudo /bin/systemctl daemon-reload  
3 sudo /bin/systemctl enable elasticsearch.service
```

Nun kann der Service gestartet werden. Zum Starten und Stoppen werden folgende Befehle genutzt:

```
1 sudo systemctl start elasticsearch.service  
2 sudo systemctl stop elasticsearch.service
```

Mit dem folgenden Befehl kann geprüft werden, ob der Elasticsearch Service am laufen ist:

```
1 curl localhost:9200
```

Die zu erwartende Ausgabe sieht wie folgt aus:

```
1 {  
2   "name" : "DOT-RH-ZLOG1",  
3   "cluster_name" : "elasticsearch",  
4   "cluster_uuid" : "5unW2cLtQoumj2z7P75pBA",  
5   "version" : {  
6     "number" : "7.10.0",  
7     "build_flavor" : "oss",  
8     "build_type" : "rpm",  
9     "build_hash" : "51e9d6f22758d0374a0f3f5c6e8f3a7997850f96",  
10    "build_date" : "2020-11-09T21:30:33.964949Z",  
11    "build_snapshot" : false,  
12    "lucene_version" : "8.7.0",  
13    "minimum_wire_compatibility_version" : "6.8.0",  
14    "minimum_index_compatibility_version" : "6.0.0-beta1"  
15  },  
16  "tagline" : "You Know, for Search"  
17}
```

Hiermit ist die Installation nach der Anleitung abgeschlossen. Damit der Elasticsearch Service von Außerhalb erreicht werden kann, muss der Service entsprechend konfiguriert werden. Dafür müssen die IP-Adressen der Server eingetragen werden, die Daten an Elasticsearch senden dürfen. Denn nach Installation ist es nur erlaubt Daten von Localhost zu senden. Die Konfiguration muss in der *elasticsearch.yml* Datei eingetragen werden. Diese ist im Pfad „*/etc/elasticsearch/elasticsearch.yml*“ zu finden. Für den Testfall ist es möglich zu definieren, dass von überall Daten an Elasticsearch gesendet werden können. Dafür muss die IP-Adresse *0.0.0.0* eingetragen werden. Das Eintragen der IP-Adressen muss in *network.host* eingetragen werden. Wenn mehrere IP-Adressen eingetragen werden müssen, dann erfolgen die Einträge in *network.hosts*. Elasticsearch ist mit diesen Konfigurationen nun bereit, daten zu erhalten.

### 3.3. Kibana

Die nächste zu installierende Komponente ist Kibana. Wie bei der Installation von Elasticsearch wird hier auch auf die Anleitung des Herstellers zugegriffen, um die Installation erfolgreich durchzuführen. [Elad] Die Installation des *rpm* Packages von Kibana erfolgt vergleichbar wie die Installation des Packages von Elasticsearch. Daher werden die einzelnen Schritte nicht nocheinmal vorgeführt.

#### 3.3.1. Index Pattern

Kibana ist im Vergleich zu Elasticsearch eine Webanwendung die von außerhalb erreicht werden kann und von Nutzern in Zukunft genutzt wird. Daher müssen wichtige Bestandteile der Oberfläche von Kibana näher erläutert werden. Zum einen müssen *Index Pattern* erstellt werden, um die Logs aus dem Elasticsearch visualisieren zu können. Doch bevor ein *Index Pattern* erstellt werden kann, müssen Daten im Elasticsearch vorhanden sein. Das hat den Vorteil, dass keine *Index Pattern* erstellt werden können, die niemals genutzt werden können, weil keine Daten zu dem Pattern passen. Wenn Daten vorhanden sind können im Reiter *Stack Management -> Index Patterns* Index Patterns verwaltet werden. Dabei muss ein *Index Pattern* als

default definiert werden. Das als default definierte *Index Pattern* wird beim aufrufen von Kibana angezeigt. Die Logs die hinter diesem Pattern stehen, werden in *Discover* angezeigt. [Elaa]

### 3.3.2. Dashboards

Die Oberfläche von Kibana ermöglicht es individuell angepasste Dashboards zu erstellen. Dabei können mehrere Diagramme mit unterschiedlichen Index Pattern angezeigt werden, um die benötigten Metriken zu veranschaulichen. So ist es möglich schnell bestimmte Informationen anzuzeigen. Für die Visualisierung können alle bekannten klassischen Diagrammtypen genutzt werden. [Elae]

### 3.3.3. Kibana Query Language (KQL)

Kibana Query Language (KQL) ist eine Abfragesprache (engl. query language), mit der das Suchen von Daten in großen Datenmengen ermöglicht wird. In Kibana wird KQL dazu genutzt, die vorhanden Logs nach bestimmten Kriterien zu Filtern und die gewünschten Daten zu erhalten. KQL setzt dabei auf eine leicht verständliche Syntax, damit keine große einarbeitung nötig ist. Beim eintippen in der Suchleiste werden Vorschläge gegeben, um schnelles Filtern zu ermöglichen. Jedoch wird dafür mindestens eine *Basic License* benötigt. Sollte KQL nicht gewünscht sein, kann man diese ausschalten und *Lucene* nutzen. Lucene ist sowie KQL eine Abfragesprache, die jedoch Open Source ist. Die Syntax von Lucene sieht so aus: [Apa]

```
1 title: "The Right Way" AND text:go
```

Um beispielhaft eine KQL Abfrage vorzuführen, wurde eine Suche in Kibana durchgeführt. Dabei können zwei unterschiedliche Methoden genutzt werden. Beide Varianten der gleich Abfrage werden hier gezeigt:

```
1 properties.LANR=8303571; properties.username=TestPraesentation  
2 properties.LANR:8303571 or properties.username=TestPraesentation
```

## 3.4. Logstash

Logstash ist eine der Komponenten aus dem Elastic Stack. Dabei ist Logstash eines der mächtigsten Tools aus dem Elastic Stack. Mit Logstash können die Logs Manipuliert werden und noch vieles mehr. Jedoch ist ein großes Problem von Logstash, dass es sehr viel Leistung benötigt betrieben zu werden und sehr viel Wissen um es richtig zu konfigurieren. Daher kann Logstash Probleme mit sich bringen. Im Laufe der Durchführung ist dabei aufgefallen, dass Logstash im Falle des CFT Portale nicht benötigt wird. Denn Filebeat übernimmt das Senden der Logs an Elasticsearch und ist dabei leichter zu konfigurieren und benötigt nicht viel Leistung.

## 3.5. Filebeat

Filebeat ist die letzte zu installierende Komponente des Elastic Stack. Die Installation von Filebeat muss etwas anders ablaufen. Denn Filebeat wird auf den Windows Servern der produktiven und test Systemen installiert. Daher muss eine Windows Installation durchgeführt werden. Zu Beginn muss ein Zip File runtergeladen werden, dass die Open Source Version von Filebeat enthält. Nach der Extraktion des Archivs, muss ein PowerShell-Skript als Administrator ausgeführt werden. Das PowerShell-Skript *install-service-filebeat.ps1* führt die komplette installation durch.

Nachdem die Installation abgeschlossen ist, muss die Verbindung zu Elasticsearch und Kibana hergestellt werden. Dafür muss die Konfigurationsdatei *Filebeat.yml* angepasst werden. Zu erst muss eine Verbindung zum Elasticsearch hergestellt werden. Das funktioniert mit folgender Zeile in der Konfigurationsdatei:

```
1 output.elasticsearch:  
2     hosts: ["http://DOT-RH-ZLOG1.doms.kvwl.de:9200"]
```

Sollte es mehrere Elasticsearch Services geben, können diese in das Interval der Hosts mit eingetragen werden. So kann bei hoher Last zu mehreren Services gesendet wendet.

Damit vordefinierte Index Pattern an Kibana gesendet werden können, muss Kibana auch in der Konfigurationsdatei eingetragen werden. Der Befehl ist dabei ein bisschen anders:

```
1 setup.kibana:
```

```
2     host: "http://DOT-RH-ZLOG1.doms.kvwl.de:5601"
```

Damit der Filebeat Service weiß welche Logs zu Elasticsearch gesendet werden sollen, müssen noch *inputs* definiert werden. Dabei können mehrere Inputs definiert werden. Die Inputs werden dabei getrennt mit „;“ angegeben. Für die Inputs können verschiedene Konfigurationen definiert werden. Damit die Logs der Vierteljahreserklärung gesammelt und gesendet werden, wird diese Konfiguration benötigt:

```
1 filebeat.inputs:
2 - type: log
3   enabled: true
4   paths: D:\logs\KVWL.PortalVEAddIn\*
5   json.keys_under_root: true
6   json.add_error_key: true
7   fields:
8     app_name: VEAddIn
```

Zeile 1 definiert, dass jetzt *inputs* folgen. Zeile 2 bis 8 sind die Konfigurationen von dem einen *input*. Zeile 5 und 6 sind dabei wichtig, denn diese ermöglichen das Einlesen von strukturierter Log-Daten im JSON-Format. Zeile 7 und 8 beschreiben selbst definierte Felder in den gesendeten Logs. Da können Informationen für alle Logs aus diesem Input mitgesendet werden. In diesem Fall ist das der Name der Anwendung der mitgeschickt wird. So kann in Kibana nach den einzelnen Apps gefiltert werden.

## 3.6. Struktur der Logs

Das CFT Portale arbeitet mit 3 Stages: Prod, Test und Dev. Dabei beschreibt Prod das produktive System auf denen die Apps für die Mitglieder installiert werden. Test ist das Testsystem auf denen die Software getestet wird bevor sie ins produktive System deployed wird. Dabei ist das Testsystem identisch zum produktiven System aufgebaut. So können Konfigurationsfehler ausgeschlossen werden. Dev spiegelt eine Umgebung wieder, in der das Team neue Features testen kann. Die Dev Umgebung kann sich von der Test- und der Prod Umgebung unterscheiden.

Während in der Dev und Test Umgebung nur ein Server läuft, wird in der Prod Umgebung auf zwei Server gesetzt die gespiegelt sind. Die zwei gespiegelten Server

sollen dafür sorgen, dass ausfälle abgefangen werden können.

Die Abbildung 3.1 zeigt die Serverstruktur des CFT Portale. Dabei kann man alle Server erkennen, auf denen die Provider hosted Apps des CFT Portale installiert sind und zusätzlich des zentralisierten Log-Server. Der zentralisierte Log-Server hat den Namen: *DOT-RH-ZLOG1*. Die anderen Server sind je nach Stage benannt. Dabei ist die erste Stage Prod. Auf Prod sind zwei Server vorhanden, die gespiegelt sind. Dort sind die Apps installiert in der produktiven Umgebung. Die Servernamen sind: *PHA1* und *PHA2*. Für Test und Dev gibt es jeweils einen Server mit dem gleichen Namen wie die Stage.

Auf jedem der in der Abbildung 3.1 zu sehenden Server der drei Stages, sind alle Provider hosted Apps des CFT Portale installiert. Das heißtt, dass für einen Server jeweils ein Filebeat Agent installiert wurde. Dieser Agent kümmert sich um die Logs der Apps auf diesem einen Server. Das bedeutet, dass jede App vier mal installiert ist und auch genau so viele Logs von unterschiedlichen Servern schreiben wird. Damit die Logs lesbar und strukturiert bleiben, musste eine durchsuchbare Lösung erarbeitet werden.

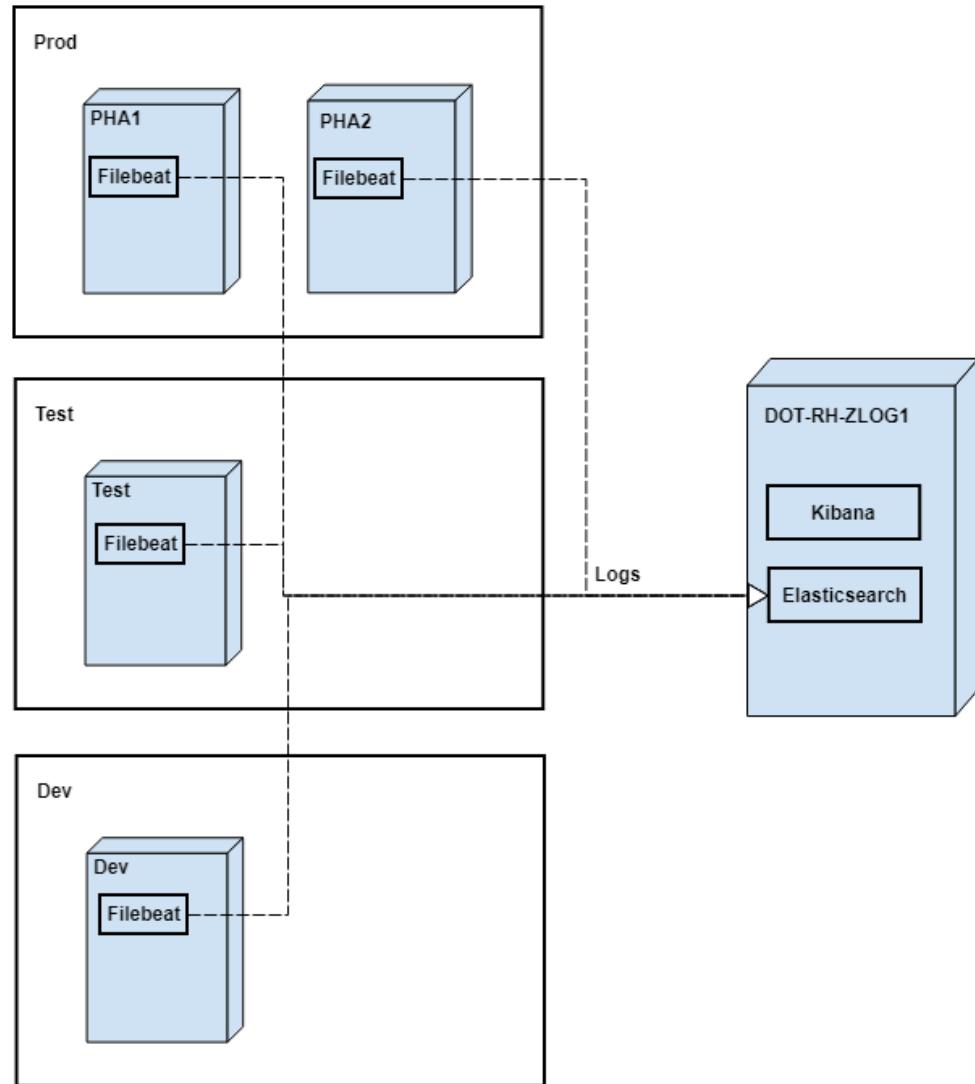


Abbildung 3.1.: Stages mit Logs

Damit die Suche in Kibana effektiv durchgeführt werden kann, werden drei Index Pattern erstellt. Jeweils für eine Stage. Also sind die drei Patterns dann: *Prod*, *Test* und *Dev*. So kann eine erste Schnelle Trennung der Zuständigkeiten erreicht werden. Da natürlich pro Stage immer noch eine hohe Anzahl an unterschiedlichen

Apps vorhanden ist, wird es schwierig darin die erforderlichen Daten zu finden. Die Patterns sollen als erste Filterung genutzt werden um den Anwendungsfall zu finden. Anschließend soll nach den Apps gefiltert werden können. Damit diese geschehen kann, muss in der Filebeat Konfiguration für jedes Input ein extra Feld definiert werden in dem der Appname definiert ist. Die extra Zeilen für ein Input sehen so aus:

```
1 fields:  
2   app_name: VEAddIn
```

Das Beispiel zeigt die Konfigurationen für die Vierteljahreserklärung. Mit diesen zusätzlichen Feldern ist eine effektive Filterung nach Stages und Anschließend nach den Apps möglich. Für das Team ist ein häufig auftretender Fall in der produktiven Umgebung, dass es notwendig ist zu wissen auf welchem der beiden produktiven Server (PHA1 und PHA2) die Logs geschrieben wurden. Filebeat sendet dafür automatisch die Informationen über den Hostname mit. Somit muss das Team zusätzlich zu den zwei Filterungsoptionen noch nach dem *agent.hostname* Filtern.

# **4. Umsetzung der Richtlinien**

In diesem Kapitel werden die Richtlinien der vorher geschriebenen Projektarbeit praktisch an der Anwendung *Vierteljahreserklärung* durchgeführt. Dafür muss der Quellcode angepasst werden, sowie einzelne Konfiguration der Anwendung. Um das Verständnis für die Anwendung zu erhalten, wird zu Beginn erstmal beschrieben wofür die *Vierteljahreserklärung* überhaupt genutzt wird und wie die Anwendung entwickelt wurde.

## **4.1. Vierteljahreserklärung**

Die Provider hosted App *Vierteljahreserklärung* ermöglicht die Generierung einer PDF für die Vierteljahreserklärung. Dabei ist die Vierteljahreserklärung eine Erklärung, die von den Mitgliedern der Kassenärztlichen Vereinigung Westfalen-Lippe an die KVWL gesendet werden muss. Die Mitglieder müssen dabei unterschiedliche Angaben zu dem Quartal abgeben. Am Ende muss ein Quartal angegeben werden, für das die Vierteljahreserklärung erstellt werden soll. Nach der Angabe des Quartals erfolgt eine Überprüfung der Software, für welche BSNR das Mitglied berechtigt ist eine Vierteljahreserklärung abzugeben. Nach Auswahl der BSNR kann das PDF Dokument heruntergeladen werden. Die Provider hosted App kann in Abbildung 4.1 genauer betrachtet werden.

// BILD NOCHMAL AUF DER ARBEIT SCREENSHOTEN (GRÖßERER MONITOR UND HÖHERE AUFLÖSUNG)



Die App *Vierteljahreserklärung* besteht aus einer C# Anwendung und einer Angular Anwendung. Dabei ist die Angular Anwendung für die Darstellung der Daten zuständig. Die C# Anwendung sorgt dafür, dass alle Abfragen durchgeführt werden und die notwendigen Daten für die Darstellung zur Verfügung stehen. Damit beiden Anwendungen miteinander kommunizieren können, wird in der C# Anwendung eine REST-Schnittstelle zur Verfügung gestellt. Die Schnittstelle bietet zwei GET-Anfragen: *GetBSNRsAsync* und *GetPDFAsync*. *GetBSNRsAsync* liefert die gültigen BSNRs für ein bestimmtes Quartal. Das Quartal und das zugehörige Jahr werden als Parameter übergeben. In der Methode wird die *GetBSNR* Methode aus der *ArztregisterService* Klasse aufgerufen. Die Methode kümmert sich um das beschaffen der notwendigen Daten der Arztregister Schnittstelle. Die Arztregister Schnittstelle stellt alle notwendigen Daten der Mitglieder zur Verfügung. Die Schnittstelle wird von dem CFT Sicherstellung der KVWL gewartet und weiterentwickelt. Die aus der Arztregister Schnittstelle erhaltenen Daten sind sehr stark verschachtelt und enthalten viele Informationen die für die Vierteljahreserklärung irrelevant sind. Daher müssen die Daten in DTOs (Data transfer Objects) geladen werden, um die Menge an Daten zu reduzieren. Das Reduzieren der Daten ist für die Anwendungen im Mitgliederportal wichtig, denn nicht jedes Mitglied der KVWL besitzt eine ausreichend starke Netzanbindung, um große Datenmengen zu erhalten. Daher ist es wichtig, alle Daten die über das Netzwerk gesendet werden, nur relevante Informationen enthalten.

## 4.2. Konfiguration von NLog

Damit die Anforderungen aus den Richtlinien bezüglich NLog vollständig erfüllt werden können, muss NLog richtig konfiguriert werden. Mithilfe von NLog können drei Richtlinien der Projektarbeit umgesetzt werden. Die Richtlinien sind:

- Die Struktur eines Logs
- Die Verwendung von strukturiertem Logging
- Die Verwendung des Logging Frameworks NLog

Die Konfiguration von NLog kann entweder durch eine eigene *Nlog.config* Datei realisiert werden oder durch das einbinden der Konfigurationen in die *Web.config*. Um den Zugriff auf die Konfigurationen von NLog zu vereinfachen, wurden die Konfigurationen in einer eigenen *Nlog.config* Datei ausgelagert.

Damit die gewünschte Log-Struktur und strukturiertes Logging verwendet werden können, muss ein *target* definiert werden. Dieses *target* beschreibt den Ort an den die Logs geschrieben werden sollen und welche Informationen mitgeliefert werden. Die gewünschte Struktur erfordert das erstellen eines eigenen Layouts für NLog. Das definierte Layout ist in Quellcode 4.1 zu sehen. In dem Layout werden alle attribute definiert. Die Attributen werden dann in allen Logs mitgeliefert. Im Layout wird außerdem definiert, dass strukturiertes Logging verwendet werden soll. Durch die Zeile 3, in der *includeAllProperties* auf *true* gesetzt wird, wird das erweitern der Logs durch eigens erstellte Attribute im Quellcode ermöglicht. So ist es dann möglich, für spezielle Fälle die Logs zu erweitern.

```
1 <layout type="JsonLayout">
2   <attribute name="properties" encode="false">
3     <layout type="JsonLayout" includeAllProperties="true"
4       maxRecursionLimit="2">
5       <attribute name="time" layout="${longdate}"/>
6       <attribute name="correlationId" layout="${activityid}"/>
7       <attribute name="method" layout="${callsite}"/>
8       <attribute name="level" layout="${level}"/>
9       <attribute name="message" layout="${message}"/>
10      </layout>
11    </attribute>
12  </layout>
```

Quellcode 4.1: Konfiguration NLog

Um die gewünschten Daten im Log zu erhalten, werden vordefinierte Variablen genutzt. Jedoch reichen die vordefinierten Variablen nicht aus. Damit die *correlationId* genutzt werden kann, muss die *\${activityid}* bei dem Beginn eines Requests neu gesetzt werden. Das geschieht in der *Global.asax*. Der Quellcode dafür ist in Quellcode 4.2 zu finden. Die genutzte Methode wird von der Oberklasse der *Global.asax* vererbt und sorgt dafür, dass die Methode beim starten eines neuen Requests aufgerufen wird. Die Oberklasse ist *HttpApplication*.

```

1 protected void Application_BeginRequest(object sender, EventArgs e)
2 {
3     Trace.CorrelationManager.ActivityId = Guid.NewGuid();
4 }

```

Quellcode 4.2: Setzen der CorrelationID

### 4.3. Aufbau der Logs

Das Ziel der Richtlinien ist das Produzieren von Logs, die lesbarer und effizienter zu durchsuchen sind. Wichtig beim Lesen und Suchen ist dabei der Aufbau der Logs. In den Richtlinien wurde definiert, dass ein Log folgende Informationen liefern soll:

- Timestamp
- LogLevel
- Correlation ID
- System Komponente
- String, um den Fehler oder das Ereignis zu beschreiben

Durch die Konfiguration von NLog werden alle Informationen die benötigt werden auch mitgeliefert. Ein Log, dass jetzt geschrieben wird, sieht wie folgt aus:

```

1 {
2     "properties": {
3         "time": "2021-01-04 11:47:18.7972",
4         "correlationId": "4f80c77d-30d4-453a-933a-392e502a29e6",
5         "method": "KVWL.PortalVEAddIn.BLL.Services.
6             ArztregisterService.GetBSNR",
7         "level": "Info",
8         "message": "LANR: \"8303571\" in dem Jahr: 2020 Mit diesem
9             Quartal: 4",
10        "LANR": "8303571",
11        "year": 2020,
12        "quarter": 4
13    }
14 }

```

Der Dafür genutzte Log Befehl ist:

```
1 log.Info("LANR: {LANR} in dem Jahr: {year} Mit diesem Quartal: {  
    quarter}", LANR, year, quarter);
```

Die Nachricht die beim Log-Befehl eingegeben wird, ist im Log anschließend im Attribut *message* hinterlegt. Im Log werden auch die drei definierten Attribute *LANR*, *quarter* und *year* als eigene Attribute gespeichert. Alle Attribute die von NLog geschrieben werden, sind unter dem Attribut *properties* gespeichert. Der Grund dafür ist der Elastic Stack. Denn durch das extra Attribut können die Logs besser in Kibana visualisiert werden. Filebeat sendet zusätzlich zu den Logs eigene Daten. Durch das zusätzliche Attribut *properties* können die Anwendungsspezifischen Daten schneller von den Filebeat spezifischen Daten differenziert werden.

#### 4.4. Abgleich der Richtlinien

# Literatur

- [Apa] Apache. *Apache Lucene - Query Parser Syntax*. URL: [https://lucene.apache.org/core/2\\_9\\_4/queryparsersyntax.html](https://lucene.apache.org/core/2_9_4/queryparsersyntax.html) (besucht am 17.12.2020).
- [Bol] Kevin Bollich. „Erarbeitung von Logging-Richtlinien für das CFT Portale der Kassenärztlichen Vereinigung Westfalen-Lippe“. In: (), S. 60.
- [DB-] DB-Engines. *DB-Engines Ranking*. DB-Engines. URL: <https://db-engines.com/en/ranking/search+engine> (besucht am 29.09.2020).
- [Elaa] Elastic. *Create an Index Pattern | Kibana Guide [7.10] | Elastic*. URL: <https://www.elastic.co/guide/en/kibana/current/index-patterns.html> (besucht am 14.12.2020).
- [Elab] Elastic. *Elastic Stack: Elasticsearch, Kibana, Beats und Logstash*. Elastic. URL: <https://www.elastic.co/de/elastic-stack> (besucht am 16.11.2020).
- [Elac] Elastic. *Install Elasticsearch with RPM | Elasticsearch Reference [7.10] | Elastic*. URL: <https://www.elastic.co/guide/en/elasticsearch/reference/current/rpm.html#rpm> (besucht am 09.12.2020).
- [Elad] Elastic. *Install Kibana with RPM | Kibana Guide [7.10] | Elastic*. URL: <https://www.elastic.co/guide/en/kibana/current/rpm.html> (besucht am 14.12.2020).
- [Elae] Elastic. *Kibana: Visualisieren, Analysieren und Erkunden von Daten*. Elastic. URL: <https://www.elastic.co/de/kibana> (besucht am 15.12.2020).
- [Elaf] Elastic. *Offizielle Preisinformationen zu Elasticsearch* | Elastic. URL: <https://www.elastic.co/de/pricing/> (besucht am 04.01.2021).
- [Flu] Fluentd. *Fluentd | Open Source Data Collector | Unified Logging Layer*. URL: <https://www.fluentd.org/> (besucht am 10.11.2020).
- [Graa] Graylog. *Graylog | Open Source vs. Enterprise*. URL: <https://www.graylog.org/products/open-source-vs-enterprise> (besucht am 04.11.2020).
- [Grab] Graylog. *Industry Leading Log Management | Graylog*. URL: <https://www.graylog.org/> (besucht am 02.11.2020).
- [Pro] Fluentd Project. *What Is Fluentd? | Fluentd*. URL: <https://www.fluentd.org/architecture> (besucht am 10.11.2020).

- [Sema] Sematext. *Sematext Enterprise Overview*. URL: <https://sematext.com/docs/sematext-enterprise/> (besucht am 12.11.2020).
- [Semb] Sematext. *Sematext Enterprise: Log Management & Infrastructure Monitoring Solution*. Sematext. URL: <https://sematext.com/enterprise/> (besucht am 12.11.2020).

# A. Anhang

## A.1. Vergleich Graylog Open Source vs. Enterprise

	OPEN SOURCE <small>Contact sales</small>	GRAYLOG ENTERPRISE <small>Contact sales</small>
<a href="#">Extended log collection using Sidecar</a>	✓	✓
Scalable log collection	✓	✓
Log enrichment data	✓	✓
Simple UI for administration	✓	✓
Graphical log analysis	✓	✓
<a href="#">Content Packs</a>	✓	✓
<a href="#">Alerts &amp; Triggers</a>	✓	✓
<a href="#">REST API</a>	✓	✓
Free marketplace of extensions	✓	✓
LDAP integration	✓	✓
<a href="#">Correlation Engine</a>		✓
<a href="#">Scheduled Reports</a>		✓
<a href="#">Data Forwarder</a>		✓
<a href="#">Offline log Archiving</a>		✓
<a href="#">User Audit Logs</a>		✓
<a href="#">Search Parameters</a>		✓
<a href="#">Technical Support</a>		✓
<a href="#">Search Workflows</a>		✓

Abbildung A.1.: Vergleich Graylog Open Source vs. Enterprise

## A.2. Eideststattliche Erklärung

### Eideststattliche Erklärung

Ich versichere an Eides statt, dass ich die vorliegende Arbeit selbstständig angefertigt und mich keiner fremden Hilfe bedient sowie keine anderen als die angegebenen

Quellen und Hilfsmittel benutzt habe. Alle Stellen, die wörtlich oder sinngemäß veröffentlichten oder nicht veröffentlichten Schriften und anderen Quellen entnommen sind, habe ich als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Dortmund, den 31.08.2020

Kevin Bollich

**Erklärung**

Mir ist bekannt, dass nach § 156 StGB bzw. § 163 StGB eine falsche Versicherung an Eides Statt bzw. eine fahrlässige falsche Versicherung an Eides Statt mit Freiheitsstrafe bis zu drei Jahren bzw. bis zu einem Jahr oder mit Geldstrafe bestraft werden kann.

Dortmund, den 31.08.2020

Kevin Bollich