

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

27. Januar 2021

Überblick

Kurzfassung

Diese Bachelorarbeit befasst sich mit der Einrichtung eines zentralisierten Logging-Servers und der Umsetzung von Logging-Richtlinien für das CFT Portale der Kassenärztlichen Vereinigung Westfalen-Lippe. Zu Beginn wird eine Evaluation erstellt, um ein passendes Tool für die Einrichtung eines zentralisierten Logging-Servers zu finden. Nach der Evaluation erfolgt die Installation des Logging-Servers. Anschließend werden die Logging-Richtlinien umgesetzt. Zum Abschluss werden die Ergebnisse der Arbeit vorgestellt.

Abstract

This bachelor thesis describes the installation of a centralized logging-server and the implementation of logging-guidelines for the „CFT Portale“ of the „Kassenärztlichen Vereinigung Westfalen-Lippe“. To locate a suitable tool for the centralized logging-server, at the beginning will be an evaluation. After the evaluation starts the installation of the centralized logging-server. Subsequently, perform the implementation of the logging-guidelines. At the end would be a description about the results of this bachelor thesis.

Inhaltsverzeichnis

1. Einführung	1
1.1. Motivation	1
1.2. Problemstellung	1
1.3. Zielsetzung	2
1.4. Vorgehensweise	2
2. Evaluation von Log-Management-Tools	4
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	8
2.2.3. Sematext	10
2.2.4. Fluentd	12
2.3. Vergleich der Tools	13
2.3.1. Graylog vs. Elastic Stack	13
2.3.2. Sematext vs. Elastic Stack	14
2.3.3. Fluentd vs. Logstash	15
2.4. Zusammenfassung	15
3. Einrichtung des zentralisierten Logging-Servers	17
3.1. RedHat Server	17
3.2. Elasticsearch	18
3.3. Kibana	20
3.3.1. Index Pattern	20
3.3.2. Dashboards	21
3.3.3. Kibana Query Language (KQL)	21
3.4. Logstash	22
3.5. Filebeat	23
3.6. Struktur der Logs	24
4. Umsetzung der Richtlinien	28
4.1. Vierteljahreserklärung	28
4.2. Konfiguration von NLog	31
4.3. Aufbau der Logs	33

4.4.	Was wird geloggt?	34
4.4.1.	Aufruf von externen Schnittstellen	35
4.4.2.	Auftreten von Fehlern	36
4.4.3.	Aufruf der REST-Schnittstellen	36
5.	Ergebnis	38
5.1.	Prozess vor der Bachelorarbeit	38
5.2.	Aktueller Prozess	40
5.3.	Erste praktische Erfahrungen	41
6.	Fazit und Ausblick	42
A.	Anhang	46
A.1.	Eidestattliche Erklärung	46

Abbildungsverzeichnis

2.1. Vergleich Graylog Open Source vs. Enterprise [Graa]	9
2.2. Fluentd vorher und nachher Vergleich [Pro]	13
3.1. CPU-Auslastung	22
3.2. Stages mit Logs	26
4.1. Vierteljahreserklärung	30
5.1. Prozess vor der Bachelorarbeit	39
5.2. Aktueller Prozess	41

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 in der redundanten Serveraufteilung 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 speichert ihre Logs auf dem jeweiligen Server. 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 Filtermöglichkeit 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? [Bol20]

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 Applikation „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 Stacks. 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 kostenlos sein, dann wird sich für das kostenlose 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 sich einige Anforderungen auf den Wartungsaufwand beziehen.

Da das Netzwerk der KVWL 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 KVWL läuft.

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.

Im CFT Portale werden keine eigenen Datenbanken betreut. Alle notwendigen Daten werden von anderen Teams zur Verfügung gestellt. Damit das CFT Portale keine weiteren Aufgaben zu sich zieht, fordert das CFT Portale, dass das Log-Management-Tool keine Logs in einer Datenbank speichert.

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 Updaten von neuen Versionen einfacher funktioniert.

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. Die Anforderungen werden nachfolgend kurz zusammenfassend aufgelistet:

- Selbsverwaltete Lösung (keine Cloud-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 wichtigsten Inhalte des Elastic Stacks 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 Kernprodukte sind Elasticsearch, Kibana, Logstash und Beat. Mit den Tools ist es möglich, zuverlässig Daten aus vielen unterschiedlichen Quellen zu erfassen und anschließend zu analysieren und visualisieren.

Mit der Komponente Elasticsearch ist das effiziente Suchen in großen Datenmengen möglich. [Elab] In einem Ranking von Suchmaschinen wurde Elasticsearch als erster Platz ausgezeichnet und beweist damit, dass Elasticsearch schnell und effizient mit großen Datenmengen umgehen kann. [DB-]

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 Stacks können auf den aktuell vorhanden Betriebssystemen installiert werden. Zusätzlich zu den Komponenten des Elastic Stacks 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 Nutzung des Elastic Stacks ist grundsätzlich kostenlos und ist zu einem großen Teil Open Source. Es ist möglich den Quellcode des Elastic Stack in öffentlichen Repositories betrachtet. Der Elastic Stack kann als Open Source Version oder als Basic Version installiert werden. Die Open Source Version ist in den Features jedoch stark eingeschränkt. Wichtige Features, wie Sicherheitsmechanismen sind nicht Nutzbar. Daher ist es ratsam die ebenfalls kostenlose Basic Version mit mehr Features zu nutzen die nicht Open Source ist. Zusätzlich zu den kostenlosen Versionen ist es möglich ein kostenpflichtiges Abonnement abzuschließen um zusätzliche Features zu erhalten. Die wichtigsten Features, die durch das kostenpflichtige Abonnement zugänglich werden, sind Machine Learning und der Support.

Elastic Stack bietet zu der selbsverwalteten Lösung zusätzlich eine Cloud-Lösung die ohne Installation genutzt werden kann. Der Vorteil der Cloud-lösung liegt im Wartungsaufwand, denn bei der Cloud Lösung ist kein Installieren und Upgraden notwendig. Da die Anforderungen des Teams eine Cloud-Lösung ausschließen, wird die Cloud-Lösung nicht weiter beachtet. [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 boolesche Operationen werden für die Suche genutzt. Die dazu benötigten Felder werden durch einfaches Klicken ausgewählt. Damit muss keine neue Syntax erlernt werden und kann von unausgebildetem Personal 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 2.1 betrachtet 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 in Betrieb genommen zu werden. Das bedeutet, dass diese Produkte zusätzlich auf der zu installierenden Maschine installiert werden müssen.

Die Graylog Enterprise Variante kann bis zu 5 GB/Tag kostenlos genutzt werden. Bei den 5 GB handelt es sich um Logs die gesendet werden. Für kleine Datenmengen ist es daher ratsam direkt auf die Enterprise Variante zu setzen. [Graa]

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

Abbildung 2.1.: Vergleich Graylog Open Source vs. Enterprise [Graa]

Graylog benötigt ein paar Systemvoraussetzungen, um installiert werden zu können. Zum Beispiel kann Graylog nur auf Linux-basierten Systemen installiert werden. Zusätzlich zu dem Betriebssystem muss noch Software installiert werden. Die zu installierende Software ist: [Grad]

- Elasticsearch 6.8+ oder 7
- MongoDB 3.6, 4.0 oder 4.2
- Oracle Java SE 8 (OpenJDK 8 funktioniert auch)

An der Auflistung wird deutlich, dass spezielle Versionen der Tools installiert sein müssen, damit Graylog funktionieren kann. Dies kann bei der Wartung zu Problemen führen: Es muss beim Update auf neuere Versionen darauf geachtet werden, dass alle Versionen miteinander kompatibel sind. Daher ist das Update von Graylog nicht so einfach möglich. Ein Problem ist auch die veraltete Java Version, die von Graylog genutzt wird. Aktuell ist es schon möglich Graylog mit Java 11 zu betreiben, jedoch ist es noch in der Testphase. Das Update wird dadurch nicht vereinfacht und erfordert zusätzliche Arbeit. Das sorgt dafür, dass der Wartungsaufwand sehr hoch ist.

Abschließend 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
 - Abhängigkeiten von externer Software
 - Installation nur auf Linux

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 Ausführen innerhalb der eigenen Infrastruktur ermöglicht. Das bedeutet, 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 auch diese Funktionen:

- Infrastructure monitoring
- Application Performance Management (APM)

- Log Management
- Real User Monitoring

Sematext nutzt für das Log-Management Elasticsearch und Kibana. Das bedeutet, 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 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 konfiguriert werden. [Semb]

Damit Sematext in der eigenen Infrastruktur betrieben werden kann, muss folgende Software installiert sein:

- Docker
- Kubernetes
- Helm

Sematext Enterprise wird als Helm Chart angeboten. Helm ist der Paketmanager für Kubernetes. Ein Chart ist eine Menge von Dateien, die zugehörige Kubernetes-Ressourcen beschreibt. 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]

Anschließend eine Auflistung der Vor- und Nachteile von Sematext:

- Vorteile:
 - Support und Wartung
 - Benachrichtigungen
- Nachteile:
 - Kostenpflichtig
 - Kein eigener Log-Shipper
 - Nur eine Installation mit Docker möglich

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 2.2 zu sehen. Die Datenströme im „Before Fluentd“ Teil sind chaotisch und unstrukturiert, 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 Plugin-Architektur von Fluentd ermöglicht. Fluentd ist in einer Kombination von C und Ruby entwickelt worden. Deswegen benötigt es nur wenig Systemressourcen. 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 sind und die Performance. Außerdem verbraucht Fluentd sehr wenig Speicher. Der größte Nachteil ist der Zwang zu strukturierten Daten. Dies bedeutet, dass es nicht so flexibel möglich ist, zu entscheiden, wie die Logs auszusehen haben. Fluentd formattiert die erhaltenen Daten immer im JSON-Format. Hier nochmal eine Auflistung der Vor- und Nachteile von Fluentd:

- Vorteile:
 - Open Source
 - Hohe Auswahl an Plugins
 - Performance
- Nachteile:
 - Zwang von strukturierten Daten

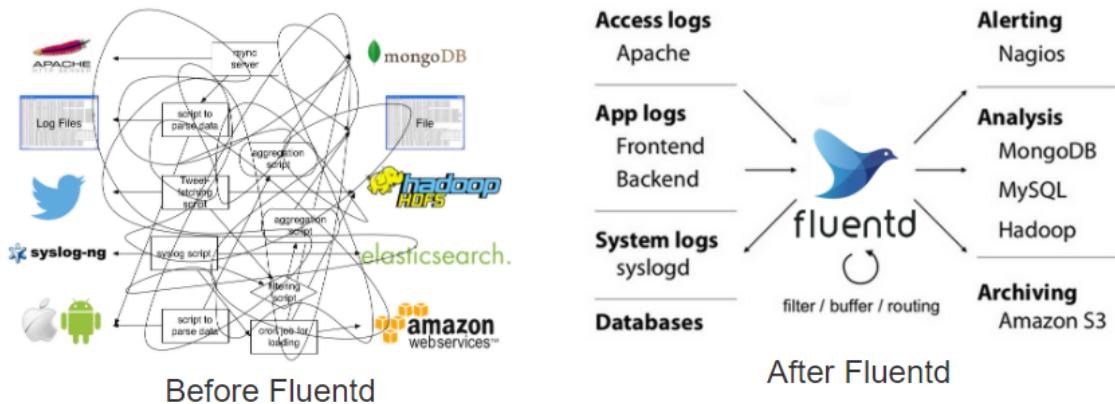


Abbildung 2.2.: Fluentd vorher und nachher Vergleich [Pro]

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.

Damit Graylog gestartet werden kann, müssen Java, Elasticsearch und MongoDB vorher installiert werden. Graylog ist somit auf Drittanbieter-Software angewiesen. Beim Updaten muss darauf geachtet werden, dass die neue Graylog Version mit den installierten Versionen der Drittanbieter-Software kompatibel ist. Das erzeugt einen geringfügigen Anstieg des Wartungsaufwandes. Jedoch ist das Updaten beim Elastic Stack ebenfalls aufwändig. Obwohl der Elastic Stack nur als Drittanbieter-Software

Java benötigt, ist die Installation dennoch aufwendig. Denn beim Update müssen alle Tools in einer bestimmten Reihenfolge installiert werden. [Elag]

1. Elasticsearch Hadoop
2. Elasticsearch
3. Kibana
4. Logstash
5. Beats
6. APM Server

Der Wartungsaufwand ist durch die bestimmte Reihenfolge der Installation eher Aufwendiger als bei Graylog.

Graylog bietet keinen eigenen Log-Agenten, der sich um das Senden von Logdateien kümmert. Graylog empfiehlt den Log-Agenten des Elastic Stacks zu nutzen, Filebeat. [Grac]

Beide Tools bieten ähnliche Funktionen an und erfüllen alle Anforderungen die vom CFT Portale definiert wurden. Daher ist es nicht möglich ein Tool für die Nutzung im Team auszuschließen. Da die Zeit für diese Bachelorarbeit begrenzt ist, wird in dieser Bachelorarbeit der Elastic Stack installiert und getestet. Sollte der Test nicht zufriedenstellend ausgehen, sollte in Anschluss der Graylog installiert und getestet werden.

2.3.2. Sematext vs. Elastic Stack

Im Gegensatz zum Elastic Stack ist Sematext kein Open-Source-Tool. Damit Sematext auf der eigenen Infrastruktur betrieben werden kann, muss Sematext Enterprise erworben werden. Durch den Erwerb der Sematext Enterprise Version, erhält man eine Kopie des Sematext Cloud Service, der auf der eigenen Infrastruktur installiert werden kann. Mit Sematext ist es anschließend möglich, *Infrastructure Monitoring*, *Application Performance Monitoring*, *Log Management* und *Real User Monitoring* durchzuführen. Im Hintergrund von Sematext läuft ein voll konfigurierter Elastic

Stack, der alle Funktionalitäten ermöglicht. Jedoch erhält man durch den Erwerb der Enterprise Version mehr Funktionen als die kostenlose Version des Elastic Stack. Der größte Vorteil von Sematext ist die einfache und schnellere Konfiguration des Systems. Bei der Installation von Sematext besteht jedoch ein Problem, denn die Installation erfolgt über Kubernetes und Docker Container. Kein Teammitglied des CFT Portale besitzt 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 zu den Anforderungen des CFT Portale am ehesten passten. Während der Evaluation wurden die Tools mit dem Elastic Stack verglichen. Bei der Evaluation ist aufgefallen, dass der Elastic Stack und Graylog die Anforderungen des CFT Portale am besten erfüllen. Jedoch ist es nicht möglich im Rahmen dieser Bachelorarbeit beide Tools zu installieren und zu testen. Daher wird im Rahmen dieser Bachelorarbeit der Elastic Stack installiert und getestet. In Tabelle 2.1 ist eine Auflistung der getesteten Tools mit relevanten Anforderungen.

	Elastic Stack	Graylog	Semantext	Fluentd
Selbstverwaltete Lösung	Ja	Ja	Ja	Ja
Einsammeln von Logs	Ja, Filebeat	Nein, Filebeat empfohlen	Nein	Nein
Speicherung ohne Datenbank	Ja	Ja	Ja	Kein Speichern
Anzeige und Analyse	Ja	Ja	Ja	Nein
Filtern und Durchsuchen	Ja	Ja	Ja	Nein
Linux-Installation	Ja	Ja	Nein	Ja
Open Source	Ja, eine Version	Ja	Nein	Ja
Drittanbieter Software	Java	MongoDB, Java, Elasticsearch	Docker, Kubernetes, Helm	Nein
Wartungsaufwand	Mittel	Mittel	Niedrig	Niedrig
Installation	Linux, Windows, MacOS	Linux, Docker, OVA	Docker, Kubernetes	Linux, Windows, MacOS

Tabelle 2.1.: Vergleich der Log-Management-Tools

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 Stacks 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 auch Linux-Server verwendet. In der Regel werden RedHat Server genutzt. So ist es auch bei dem Server für das zentrale Logging. Es ist ein RedHat Server mit der Version 7.9.

Damit die Kommunikation zum Server erfolgen kann, müssen einige Ports auf dem Server freigeschaltet werden. 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 musste 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 zufriedenstellend ist, 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. Sollte man die Open Source Version runtergeladen, dann ist es nicht möglich ein Upgrade auf die Basic Version zu machen. Damit man anschließend die Basic Version nutzen kann, muss man den Kompletten Stack neu installieren. Daher musste darauf geachtet werden, dass die Basic Version direkt installiert wird.

Aus Sicherheitsgründen hat der Server keinen Zugriff auf das Internet. Daher müssen die Installationspakete manuell vom Entwickler-Rechner runtergeladen werden und anschließend zum Server kopiert 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-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-7.10.0-x86_64.rpm <server-adresse>
```

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-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 läuft:

```
1 curl localhost:9200
```

Die zu erwartende Ausgabe sieht wie folgt aus:

```
1 {
2   "name" : "<server-name>",
3   "cluster_name" : "elasticsearch",
4   "cluster_uuid" : "5unW2cLtQoumj2z7P75pBA",
5   "version" : {
6     "number" : "7.10.0",
7     "build_flavor" : "default",
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 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 benötigt werden, 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* Paketes von Kibana erfolgt auf vergleichbare Weise wie die Installation des Packages von Elasticsearch. Daher werden die einzelnen Schritte nicht noch einmal 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. *Index pattern* sind Muster, nach denen in den Logs gesucht wird. Das Suchen erfolgt dabei auf ein bestimmtes Attribut der Logs, dem *index*. Ein Beispiel für ein *index pattern* ist: „*prod-**“. Jedes Log das ein *index* mit dem Beginn *prod* enthält, wird somit angezeigt.

Doch bevor ein *index pattern* erstellt werden kann, müssen Daten im Elasticsearch vorhanden sein. Das hat den Vorteil, dass keine *Index Pattern* erstellt werden kön-

nen, 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 vorhandenen 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 eine Java-Bibliothek, die Such-Features anbietet. Dazu gehören das Überprüfen der Syntax, Markieren der Ergebnisse und weitere Funktionen. [Apab] Die Syntax von Lucene sieht so aus: [Apa]

```
1 properties.LANR: "1234567" AND properties.level:Info
```

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=1234567; properties.level=Info
```

```
2 properties.LANR:1234567 or properties.level=Info
```

3.4. Logstash

Logstash ist eines der mächtigsten Tools aus dem Elastic Stack. Mit Logstash können die Logs gesammelt, geparsed und transformiert werden. 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. In Abbildung 3.1 ist ein Auschnitt aus der CPU-Auslastung des Servers, auf dem Logstash installiert wurde. Obwohl noch keine Logs gesendet wurden, gab es eine 100% Auslastung der CPU. Das zeigt deutlich, dass Logstash sehr viel Rechenleistung benötigt. Nach der deinstallation ist die CPU-Auslastung wieder auf den normal Bereich gesunken.

Außerdem ist während der Konfiguration aufgefallen, dass durch die neu geschaffenen Richtlinien, die Transformation und das Parsen der Logs nicht benötigt werden. Das Senden der Logs an Elasticsearch wird von Filebeat übernommen. Da die Logs durch NLog im JSON-Format gespeichert werden, gibt es keinen Nutzen mehr für Logstash. Daher wurde entschieden, dass Logstash in diesem Szenario nicht benötigt wird.

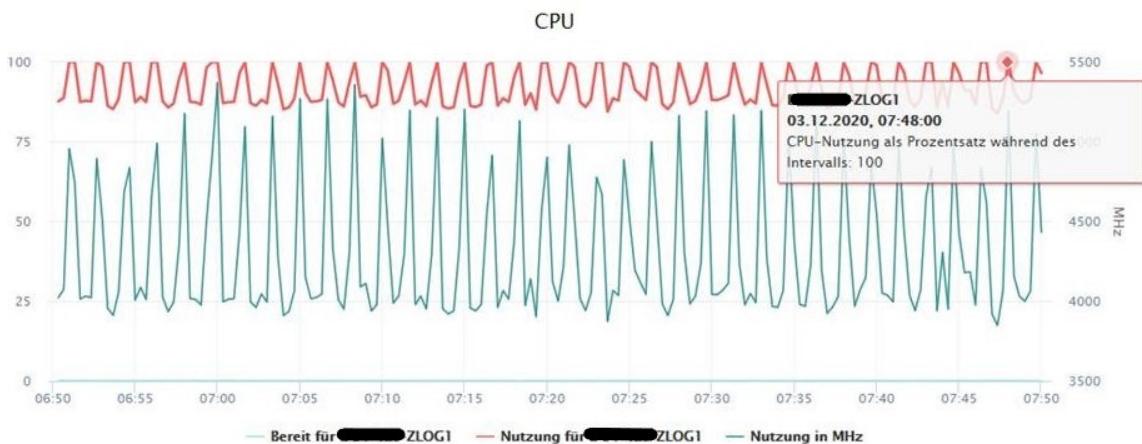


Abbildung 3.1.: CPU-Auslastung

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. Zuerst muss eine Verbindung zum Elasticsearch hergestellt werden. Das funktioniert mit folgender Zeile in der Konfigurationsdatei:

```
1 output.elasticsearch:  
2     hosts: ["http://<server-adresse>: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://server-adresse>: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 ersten *input*. Zeile 5 und 6 sind dabei wichtig, denn diese ermöglichen das Einlesen von strukturierten 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 ergänzt werden. In diesem Fall ist das der Name der Anwendung, der mitgeschickt wird. So kann in Kibana nach den einzelnen Anwendungen gefiltert werden.

3.6. Struktur der Logs

Das CFT Portale arbeitet mit drei 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*-Umgebung nur ein Server läuft, wird in der *Prod*- und *Test*-Umgebung auf zwei Server gesetzt die gespiegelt sind. Die zwei gespiegelten Server sollen dafür sorgen, dass Ausfälle abgefangen werden können.

Die Abbildung 3.2 zeigt die Serverstruktur des CFT Portale. Dabei kann man alle Server erkennen, auf denen die Provider hosted Apps des CFT Portale installiert sind (PHA-Server) und zusätzlich den 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 die *Test*-Server werden die Namen *Test1* und *Test2* verwendet. *Dev* hat einen Server mit dem gleichen Namen wie die Stage.

Auf jedem der in der Abbildung 3.2 zu sehenden Server der drei Stages, sind alle Provider hosted Apps des CFT Portale installiert. Das heißt, 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 fünf 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.

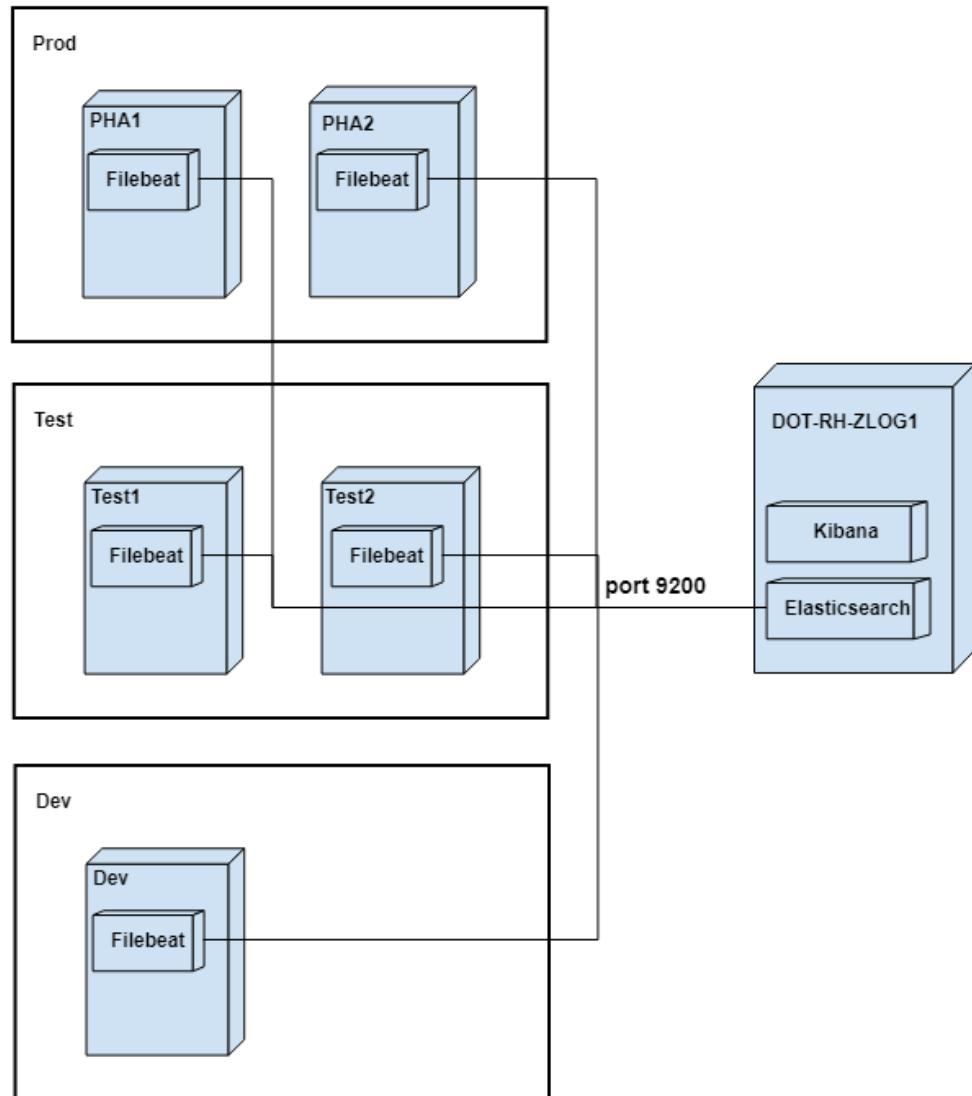


Abbildung 3.2.: Stages mit Logs

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 dies 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 einen 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 Konfigurationen der Anwendung. Um das Verständnis für die Anwendung zu erhalten, wird zu Beginn beschrieben, wofür die *Vierteljahreserklärung* überhaupt genutzt wird und wie die Anwendung entwickelt wurde.

4.1. Vierteljahreserklärung

Die Provider hosted App *Vierteljahreserklärung* ermöglicht die PDF-Generierung der *Erklärung zur Vierteljahresabrechnung*. Mit der *Erklärung zu Vierteljahresabrechnung* bestätigt der Vertragsarzt seine abgerechneten Leistungen. Dies ist in den Abrechnungsrichtlinien der KVWL festgelegt. Der Ausschnitt der Richtlinien, der die pflichten des Vertragsarztes bezüglich der *Erklärung der Vierteljahresabrechnung* beschreibt, ist Anschließend zitiert. Der Ausschnitt ist in §7 Absatz 1 zu finden:

„Der Vertragsarzt hat gegenüber der KVWL auf dem der Abrechnung beizufügenden Vordruck „Erklärung zur Vierteljahresabrechnung“ (§ 35 Abs. 2 BMV-Ä) schriftlich zu bestätigen, dass die zur Abrechnung gestellten Leistungen unter Beachtung der Vorgaben nach § 2 Abs. 1 tatsächlich erbracht worden sind und die Abrechnung sachlich-rechnerisch richtig ist. Die Vorlage des unterschriebenen Vordrucks „Erklärung zur Vierteljahresabrechnung“ ist Abrechnungsvoraussetzung.“ [Wes15]

Mithilfe der Anwendung *Vierteljahreserklärung* können die Angaben, die für die Vierteljahresabrechnung notwendig sind, eingetragen werden. Ein Screenshot der Anwendung ist in Abbildung 4.1 abgebildet. Bevor die PDF generiert werden kann, ist die Eingabe des Quartals und der Betriebstättennummer (BSNR) Voraussetzung.

Die BSNR ist eine neunstellige Zahl, die eine Zuordnung ärztlicher Leistungen zum Ort der Leistungserbringung ermöglicht. Die ersten beiden Ziffern stellen den KV-Landes- oder Bezirksstellenschlüssel dar. Die Ziffern drei bis neun werden von der Kassenärztlichen Vereinigung vergeben, in dessen Bereich die Betriebstätte liegt. Bei Abrechnungen muss die BSNR angegeben werden. Dies ist in den Richtlinien der Kassenärztlichen Bundesvereinigung (KBV) festgelegt worden. Eine BSNR bekommt jede Betriebstätte und Nebenbetriebstätte, sowie Einrichtungen die zur Teilnahme der vertragsärztlichen Versorgung ermächtigt wurden. [Bun19]

Nach Angabe des Quartals erfolgt eine Überprüfung der BSNRs, die in diesem Quartal für den Vertragsarzt gültig sind. Dabei wird geprüft, an welchen Betriebstätten der Vertragsarzt tätig war. Nach Erhalt der BSNRs, kann der Vertragsarzt eine BSNR aussuchen, für die diese Erklärung erstellt werden soll.

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 beide 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 HTTP-Anfragen zu reduzieren. Mit den DTOs können dann mehr Daten über einen Aufruf gesendet werden. Das Reduzieren der Daten ist für die Anwendungen im Mitgliederportal wichtig, denn nicht jedes Mitglied der KVWL besitzt eine ausreichend starke Netzanbindung, um große Datens Mengen zu erhalten. Daher ist es wichtig, alle Daten die über das Netzwerk gesendet werden, nur relevante Informationen enthalten.

4.2. Konfiguration von NLog

Mithilfe von NLog können drei Richtlinien der Projektarbeit umgesetzt werden. Die Richtlinien sind:

- Die Informationen die ein Log mitliefert
- Strukturiertes Logging
- Die Verwendung des Logging Frameworks NLog

Damit die Anforderungen aus den Richtlinien bezüglich NLog vollständig erfüllt werden können, muss NLog richtig konfiguriert werden. 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. Dies hat den Vorteil, dass die Konfigurationen für NLog schneller erreicht werden und Global von mehreren Anwendungen genutzt werden können.

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 Nutzen des *JsonLayout*. Ein Layout ist ein vordefiniertes Format, in dem die Logs ausgegeben werden. [Ver]

Das Layout kann dann mit attributen erweitert werden, um den Inhalt individuell zu gestalten, denn das Layout bezieht sich nur auf die Form der Daten. In diesem Fall ist es im JSON-Format. Das definierte Layout ist in Quellcode 4.1 zu sehen. In dem Layout werden alle Attribute definiert. Die Attribute 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*. Die *Global.asax* ist eine global Anwendungsdatei in ASP.NET Anwendungen. In der *Global.asax* werden seitenübergreifende Ereignisbehandlungs routinen definiert. Zum Beispiel der Start einer Sitzung. [Sch]

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: \"1234567\" 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: {
2     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. Was wird geloggt?

In diesem Kapitel werden Beispiele der durchgeführten Logging-Anweisungen gezeigt. Dabei werden die Logs in drei Kategorien unterteilt. Jede der in diesem kapitel vorgestellten Logs sind in der Vierteljahreserklärung durchgeführt worden. Die Logs werden zusätzlich in die Kategorien eingeteilt, die in der Projektarbeit vorgestellt wurden. [Bol20]

4.4.1. Aufruf von externen Schnittstellen

Ein Abschnitt der in der Vierteljahreserklärung geloggt werden muss, ist der Aufruf von externen Schnittstellen. In der Vierteljahreserklärung wird eine Schnittstelle mehrmals genutzt um Daten der Mitglieder zu erhalten. Die Schnittstelle ist die Arztregister-Schnittstelle. Die Arztregister-Schnittstelle wird vom CFT Sicherstellung & Versorgungsqualität verwaltet. Die Arztregister-Schnittstelle liefert alle relevanten Stammdaten über die Mitglieder der KVWL.

Das loggen dieser Aufrufe ist wichtig, um zu prüfen ob die Schnittstelle die Daten zurück liefert, die erwartet werden. So kann bei fehlerhaftem Aufruf überprüft werden, ob die Daten noch übereinstimmen oder durch veränderung der Struktur falsche Daten übermittelt werden. Die Aufrufe dieser Schnittstelle werden von einem *ArztregisterClient* durchgeführt, der vom CFT Portale entwickelt wurde. Durch diesen Client ist die Kommunikation mit dem Arztregister vereinfacht worden. Der Client sorgt dafür, dass beim Aufruf der Schnittstelle nur die für das CFT Portale relevanten Daten zurückgeliefert werden. Ohne Client werden von der Schnittstelle viel mehr Daten mitgeliefert, die für das CFT Portale nicht relevant sind. Quellcode 4.3 zeigt ein Log, dass nach dem Aufruf einer vom ArztregisterClient zur Verfügung gestellten Methode, geschrieben wird. Das Log-Level ist in diesem Fall *Debug*. Das Log-Level *Debug* wurde hier gewählt, weil die Informationen nur für die Entwicklung relevant ist. Im produktiven-Betrieb ist es nicht relevant, welche Informationen von dem Befehl zurückgeliefert werden.

Diese Log gehört zu der Kategorie *Semantic Description* die in der Projektarbeit vorgestellt wurde. Um präziser zu sein, gehört dieses Log zu der Unterkategorie *variable Description*. Das Log liefert den Inhalt einer Variable, um zu überprüfen ob die Variable den erwarteten Wert beinhaltet. [Bol20]

```
1 var leistungserbringerListe = await _arztregisterClient.  
    GetLeistungserbringerClusterAsync(lanr: LANR);  
2 log.Debug("Erhaltene Leistungserbringerliste:  
    leistungserbringerliste={leistungserbringerliste},  
    leistungserbringerListe);
```

Quellcode 4.3: Aufruf der Arztregister-Schnittstelle

4.4.2. Auftreten von Fehlern

Logs werden oft genutzt um Fehler anzuzeigen und analysieren zu können. Daher wurden in der Vierteljahreserklärung Error-Messages geloggt. Dafür wurde das Log-Level *Error* genutzt. Quellcode 4.4 zeigt die Methode *ValidatePflichtangaben*. Diese Methode startet die Validierung von *Pflichtangaben*. Wenn die Validierung fehlgeschlagen ist, wird ein error-log geschrieben und eine Exception wird geworfen.

Das Loggen dieser Aufrufe ist wichtig, um zu verstehen welche Fehler während der Laufzeit aufgetreten sind, damit diese anschließend analysiert werden können. Fehleranalyse ist ein wichtiger Teil des Log-Managements.

Die Art von Logs gehören zu der Kategorie *Error Message*. Die genaue Unterkategorie ist in diesem Fall *value-check*.

```
1 private void ValidatePflichtangaben(Pflichtangaben pflichtangaben)
2 {
3     var pflichtangabenValidator = new PflichtangabenValidation();
4     var validateResult = pflichtangabenValidator.Validate(
5         pflichtangaben);
6     if (!validateResult.IsValid) {
7         log.Error("Die BSNR {bsnr} ist nicht im richtigen Format. Die
8             BSNR muss 9-stellig sein und darf nur aus Ziffern bestehen.",
9             pflichtangaben.Bsnr);
10        throw new VierteljahreserklärungException("Die BSNR ist
11            nicht im richtigen Format. Die BSNR muss 9-stellig sein und darf
12            nur aus Ziffern bestehen.");
13    }
14 }
```

Quellcode 4.4: Auftreten von Fehlern

4.4.3. Aufruf der REST-Schnittstellen

Eine wichtige Stelle zum Loggen ist für das CFT Portale, der Aufruf von REST-Schnittstellen. Deswegen müssen diese Aufrufe geloggt werden, um prüfen zu können welche Mitglieder wann eine Anfrage gesendet haben. Das ist für das Bugtracking im Team wichtig. Die Software im CFT Portale ist in *Server* und *Client* aufgeteilt.

Der Client ruft die REST-Schnittstellen der Server-Anwendung auf, um Daten zu erhalten. Die Operationen die für die Anwendungen wichtig sind, werden von der Server Anwendung durchgeführt. Daher müssen Alle Anfragen in den Logs dokumentiert werden. Das hierfür genutzte Log-Level ist *Info*. Das Log-Level *Info* wird in diesem Fall genutzt, weil diese Logs in der Produktion benötigt werden.

Quellcode 4.5 zeigt die die Methode *GetBSNRsAsync*. Diese Methode wird über HTTP-Requests erreicht und liefert die BSNRs für ein bestimmtes Quartal. Das dazu geschriebene Log befindet sich zu Beginn dieser Operation, um zu definieren, dass der Aufruf der Methode geklappt hat.

Da das Log zu Beginn der Operation platziert wurde, gehört dieses Log zu der Kategorie *Program Operation* und zusätzlich zu der Unterkategorie *next operation*.

```
1 [Route("api/bsnrs")]
2 public async Task<IHttpActionResult> GetBSNRsAsync(int quarter, int
   year)
3 {
4     log.Info("Aufruf der REST-Schnittstelle: GetBSNRsAsync mit:
      quarter={quarter}, year={year}", quarter, year);
5     var bsnrList = await _arztregisterService.GetBSNR(quarter, year,
   LANR);
6     return Ok(bsnrList);
7 }
```

Quellcode 4.5: Aufruf der REST-Schnittstellen

5. Ergebnis

In diesem Kapitel wird das Ergebnis der durchgeführten Installationen und Anpassungen des Quellcodes vorgestellt. Dabei wird zu Beginn das Problem des vorherigen Prozesses näher erläutert, um genauer Festzustellen, was die Veränderungen hinsichtlich Fehlersuche gebracht haben. Damit das Ergebnis besser bewertet werden kann, wird der Ablauf der aktuellen Version anhand von einem Beispiel genauer erläutert und dargestellt.

5.1. Prozess vor der Bachelorarbeit

Das CFT Portale muss bei der fehleranalyse auf einen langwierig prozess setzen. Dieser Prozess ist in Abbildung 5.1 anhand eines Aktivitätsdiagramms dargestellt. Der Prozess beginnt mit dem Erhalt eines Tickets oder einer Email, dass einen Fehler in der Vierteljahreserklärung beschreibt. Nachdem der Entwickler sich das Ticket näher angeschaut hat, muss dieser nach dem dabei entstandenem Log suchen. Dafür muss sich der Entwickler an einem der Beiden PHA-Server anmelden und die Logs Manuell durchsuchen. Dabei wird eine Remotedesktopverbindung auf den ersten PHA1-Server aufgebaut. Anschließend muss die entsprechende Log-Datei für die Vierteljahreserklärung in der Ordnerstruktur des Servers gefunden werden. Nachdem die Log-Datei geöffnet wurde, muss die Textbasierte-Datei manuell nach dem entsprechenden Log durchsucht werden. Bei der Suche kann der Fall auftreten, dass der entsprechende Fehler nicht auf diesem PHA-Server entstanden ist. Das bedeutet, dass der Fehler noch auf dem zweiten PHA-Server durchsucht werden muss. Daher müssen die Schritte Suchen der Datei und das Durchsuchen der Logs wiederholt werden. Wenn der Fehler auf dem zweiten PHA-Server entdeckt wurde, kann der Fehler analysiert werden und Anschließend dokumentiert. Der Prozess ist mit der Dokumentation abgeschlossen.

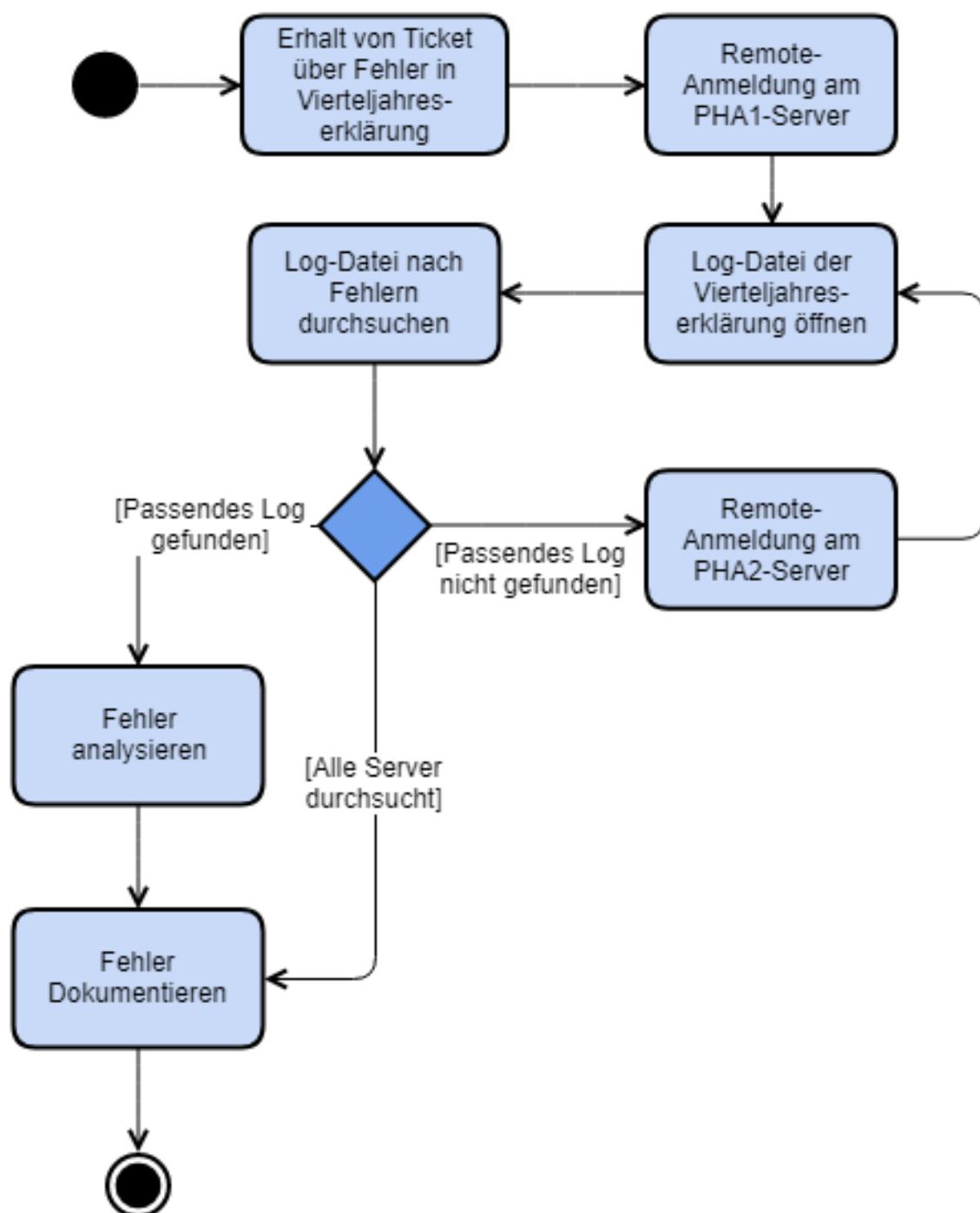


Abbildung 5.1.: Prozess vor der Bachelorarbeit

5.2. Aktueller Prozess

Der aktuelle Prozess, der durch das zentralisierte Logging und die Veränderungen des Logging-Codes ermöglicht wurde, fällt im Vergleich zum vorherigen Prozess deutlich einfacher aus. Denn schon anhand der beiden Abbildungen 5.1 und 5.2 ist erkennbar, dass der aktuelle Prozess mit weniger Aktionen auskommt. Jedoch ist der aktuelle Prozess ebenfalls vom Umfang her deutlich geringer. Die Aktionen im vorherigen Prozess benötigen mehr Aufwand für die Durchführung, als der aktuelle.

Beide Prozesse starten mit dem erhalt eines Tickets. Dann beginnt der Unterschied, denn beim vorherigen Prozess musste mit einer Remotedesktopverbindung auf den entsprechenden Server zugeschalten werden, beim aktuellen Prozess muss nur noch eine URL mit dem Browser aufgerufen werden. So wird dann Kibana aufgerufen. Der einzige Schritt der jetzt noch durchgeführt werden muss, ist die Filterung nach den Kriterien. Der entsprechende Log taucht dann in der Suche auf. Der Fehler kann nun analysiert und dokumentiert werden.

Der größte Unterschied der beiden Abbildungen, ist das Fehlen der Schleife in der Abbildung 5.2. Das heißt der aktuelle Prozess muss keine Aktionen mehrmals ausführen, um ans Ziel zu gelangen. Im vorherigen Prozess kann dies jedoch geschehen, sollten die gesuchten Logs nicht im *PHA1* vorhanden sein. Wenn dies der Fall ist, dann muss die Anmeldung am Server, das Öffnen der Log-Datei und das durchsuchen der Logs zweimal durchgeführt werden. Das erhöht den Aufwand beim Suchen nach bestimmten Logs.

Nicht nur das mehrfache Ausführen von Aktionen sind Unterschiede der beiden Prozesse, sondern der Aufwand je Aktion ist ebenfalls unterschiedlich. Denn schon das durchsuchen der Logs ist bei dem vorherigen Prozess deutlich Aufwändiger. Dies resultiert daraus, dass die Log-Datei aus dem vorherigen Prozess nicht nach bestimmten Attributen gefiltert werden kann und auch kein ausblenden von irrelevanten Informationen. Im Vergleich dazu ist es in Kibana möglich nach bestimmten Kriterien zu Suchen und nur die Informationen anzuzeigen, die relevant sind.

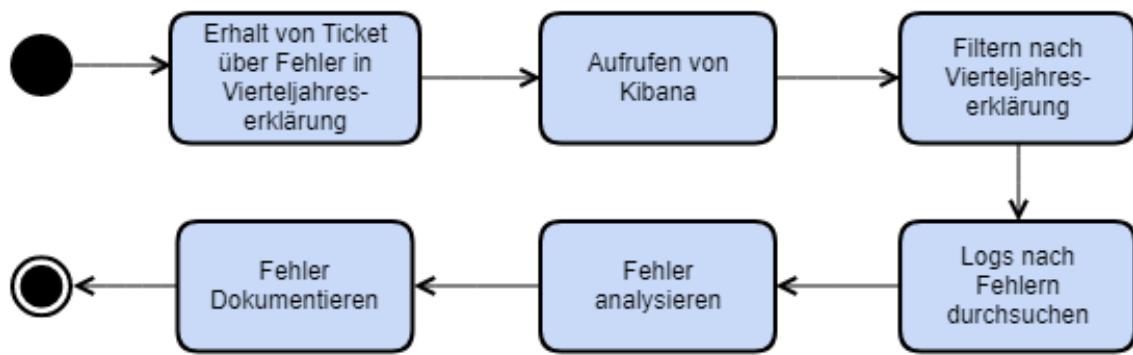


Abbildung 5.2.: Aktueller Prozess

5.3. Erste praktische Erfahrungen

Der Elastic Stack und die Umsetzung der Richtlinien wurden im CFT Portale den einzelnen Entwicklern vorgestellt. Bei der Vorstellung waren die Entwickler begeistert. Die Entwickler freuen sich einerseits darauf, dass sie sich nicht mehr an beiden PHA-Servern anmelden müssen und andererseits, dass die Logs nun genauere Details über die Fehler liefern. Denn bei der fehleranalyse werden öfter Daten wie die *LANR* und die *BSNR* benötigt, um Fehler besser zuordnen zu können. Durch das Strukturierte Logging und den Elastic Stack ist es nun möglich Mithilfe von Filterfunktion effektiv Daten in den Logs zu finden. Das Durchsuchen von Textdateien mithilfe der *STRG+F* Tastenkombination wird nicht mehr benötigt. Der Erste Eindruck ist somit sehr positiv vom CFT Portale.

6. Fazit und Ausblick

Das Ziel dieser Bachelorarbeit war es, einen zentralisierten Logging-Server für das CFT Portale einzurichten und an einer Anwendung die Logging-Richtlinien der Projektarbeit umzusetzen. Das Einrichten des zentralisierten Logging-Servers war nötig, weil der Prozess der fehleranalyse sehr aufwändig war. Hinzu kamen noch die schlecht lesbaren Logdateien, die in den Anwendungen geschrieben werden. Die enthaltenen Informationen helfen nur wenig bei der Suche nach einem Fehler.

Zu Beginn wurde eine Evaluation durchgeführt werden, um ein passendes Tool für das zentralisierte Logging zu finden. Das Ergebnis der Evaluation lieferte den Elastic Stack als beste Alternative für die Anforderungen des CFT Portale. Nach der Evaluation erfolgte die Installation des Elastic Stacks. Beim installieren der Komponenten des Elastic Stack ist aufgefallen, dass Logstash hohe CPU-Anforderungen mitbringt und im Falle des CFT Portale nicht gebraucht wird. Mit Abschluss der Installationen mussten die Logging-Richtlinien an einer Anwendung umgesetzt werden. Die Anwendung an der die Richtlinien umgesetzt worden sind, war die Vierteljahreserklärung. Damit die Richtlinien überhaupt umgesetzt werden können, musste die Anwendung verstanden werden. Dafür wurden die rechtlich relevanten Dinge erläutert und Anschließend die Architektur, um zu verstehen an welchen Stellen die wichtigen Anhaltspunkte für das Logging sind. Bevor die log-Nachrichten an den richtigen Stellen platziert werden konnten, musste das NLog Framework konfiguriert werden. Mit der Konfiguration wurde der Aufbau der Logs fest definiert. Mit Abschluss der Konfiguration konnten die logs an den richtigen Stellen in den Quellcode eingetragen werden. Zum Abschluss wurden beide Prozesse der Fehleranalyse miteinander verglichen. Dabei fiel auf, dass der neu entstandene Prozess deutlich weniger Aufwand erfordert, als der ursprüngliche.

Das Ergebnis der Bachelorarbeit war die erfolgreiche Einrichtung eines zentralisierten Logging-Servers, der den Arbeitsprozess der Fehlersuche des CFT Portale vereinfacht hat. Außerdem enthalten die geschriebenen Logs deutlich mehr Informationen als zuvor. Mithilfe des strukturierten Loggings ist es nun möglich, effektiv in den Logs nach bestimmten Attributen zu Filtern und zu Suchen.

Alle Forschungsfragen die zu Beginn der Bachelorarbeit aufgestellt wurden, konnten im Rahmen der Arbeit vollständig beantwortet werden. Die erste Forschungsfrage konnte mithilfe der Evaluation beantwortet werden. Das Ergebnis der Evaluation von Log-Management-Tools bestätigte die ursprüngliche Entscheidung, den Elastic Stack als zentralisiertes Logging-Tool einzusetzen. In der zweiten Forschungsfrage handelt sich um die Frage, wie ein zentralisierter Logging-Server eingerichtet werden kann. Das konnte mittels der Installation des Elastic Stacks beantwortet werden. Die genauen Schritte wurden in dem dazugehörigen Kapitel vollständig erklärt. In der abschließenden Forschungsfrage drehte es sich um die Logging-Richtlinien, die in dieser Arbeit praktisch umgesetzt werden sollten. Die Frage bezog sich dabei darauf, ob die Richtlinien umgesetzt werden können. Ja, die Logging-Richtlinien konnten praktisch an einem Projekt vollständig umgesetzt werden.

Im Anschluss an diese Bachelorarbeit sollten die aufgestellten Logging-Richtlinien an allen Applikationen des CFT Portale durchgeführt werden. Der aktuelle Stand bezieht sich nur auf die Vierteljahreserklärung. Dadurch ist bei der Fehlersuche in der Vierteljahreserklärung zwar ein optimaler Ablauf möglich, jedoch ist das Ziel, dass alle Applikationen des Teams über den Elastic Stack analysierbar sind. Nach erfolgreicher Integration des Elastic Stack wird das Ergebnis weiteren Teams der KVWL vorgestellt.

Literatur

- [Apaa] Apache. *Apache Lucene - Query Parser Syntax*. URL: https://lucene.apache.org/core/2_9_4/queryparsersyntax.html (besucht am 17.12.2020).
- [Apab] Apache. *Apache Lucene - Welcome to Apache Lucene*. URL: <https://lucene.apache.org/> (besucht am 12.01.2021).
- [Bol20] Kevin Bollich. „Erarbeitung von Logging-Richtlinien für das CFT Portale der Kassenärztlichen Vereinigung Westfalen-Lippe“. In: (31. Aug. 2020), S. 60.
- [Bun19] Kassenärztliche Bundesvereinigung. *Richtlinie Der Kassenärztlichen Bundesvereinigung Nach § 75 Absatz 7 SGB V Zur Vergabe Der Arzt-, Betriebsstätten- Sowie Der Praxisnetznummern*. 1. Mai 2019.
- [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).

- [Elag] Elastic. *Upgrading the Elastic Stack / Installation and Upgrade Guide [7.10]* / Elastic. URL: <https://www.elastic.co/guide/en/elasticsearch-stack/current/upgrading-elasticsearch.html> (besucht am 12.01.2021).
- [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).
- [Grac] Graylog. *Ingest from Files — Graylog 4.0.0 Documentation*. URL: <https://docs.graylog.org/en/4.0/pages/sending/files.html> (besucht am 12.01.2021).
- [Grad] Graylog. *Installing Graylog — Graylog 4.0.0 Documentation*. URL: <https://docs.graylog.org/en/4.0/pages/installation.html> (besucht am 07.01.2021).
- [Pro] Fluentd Project. *What Is Fluentd? / Fluentd*. URL: <https://www.fluentd.org/architecture> (besucht am 10.11.2020).
- [Sch] Dr. Holger Schwichtenberg. *Globale Ereignisse in ASP.NET-Webanwendungen*. URL: https://www.it-visions.de/dotnet/aspnet/ASPNET_global.aspx (besucht am 25.01.2021).
- [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).
- [Ver] Julian Verdurmen. *NLog/NLog*. GitHub. URL: <https://github.com/NLog/NLog> (besucht am 25.01.2021).
- [Wes15] Kassenärztliche Vereinigung Westfalen-Lippe. „Abrechnungsrichtlinien der Kassenärztlichen Vereinigung Westfalen-Lippe (KVWL)“. In: (1. Jan. 2015), S. 7.

A. Anhang

A.1. Eideststattliche Erklärung

Eideststattliche Erklärung

Ich versichere an Eides statt, dass ich die vorliegende Arbeit selbständig 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