HOCHSCHULE DER MEDIEN

Stuttgart Media University

Bachelor Thesis im Studiengang
Medieninformatik

Konzeption und Implementierung eines Monitoring Systems für Docker Swarm Cluster

Oliver Fesseler

Erstprüfer: Prof. Dr. Martin Goik
Zweitprüfer: Thomas Maier

13. Februar 2017

Erklärung der Urheberschaft

Hiermit versichere ich, Oliver Fesseler, an Eides Statt, dass ich die vorliegende Bachelorarbeit mit dem Titel: „Konzeption und Implementierung eines Monitoring Systems für Docker Swarm Cluster“ selbständig und ohne fremde Hilfe verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe.

Die Stellen der Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken entnommen wurden, sind in jedem Fall unter Angabe der Quelle kenntlich gemacht. Die Arbeit ist noch nicht veröffentlicht oder in anderer Form als Prüfungsleistung vorgelegt worden. Ich habe die Bedeutung der eidesstattlichen Versicherung und die prüfungsrechtlichen Folgen (§26 Abs. 2 Bachelor-SPO (6 Semester), § 23 Abs. 2 Bachelor-SPO (7 Semester) bzw. § 19 Abs. 2 Master-SPO der HdM) sowie die strafrechtlichen Folgen (gem. § 156 StGB) einer unrichtigen oder unvollständigen eidesstattlichen Versicherung zur Kenntnis genommen.

Ort, Datum Unterschrift

Danksagung

Danke: Laura Geßner, Thomas Maier, Prof. Martin Goik, Leomedia GmbH, Carsten Neumann

Zusammenfassung

Skalierbarkeit und Ausfallsicherheit sind bei der Entwicklung von Verkaufsplattformen relevante Größen, um Vertrauen und Verlässlichkeit gegenüber dem Kunden zu gewährleisten. Clusterlösungen sind dazu prädestiniert, da sie auf sich ändernde Anforderungen flexibel regieren können und bei teilweisem Ausfall funktionsfähig bleiben. Im Folgenden wird unter Cluster ein Zusammenschluss von Infrastrukturkomponenten verstanden, welche mit Docker, Docker Swarm, WeaveNet, Consul und GlusterFS betrieben werden.

Zur Gewährleistung eines unterbrechungsfreien Betriebes muss ein solches System überwacht werden. Dabei ist es wichtig einen Überblick über den aktuellen Status zu haben, damit im Fehlerfall entsprechend reagiert werden kann. Diese Bachelorarbeit beschäftigt sich mit der Evaluation von Metriken, welche den aktuellen Clusterstatus repräsentieren. Im Anschluss wird unter Einbezug dieser eine prototypische Implementierung eines Überwachungssystems ausgearbeitet.

Der besondere Fokus dieser Bachelorarbeit liegt dabei auf der Auswahl und Interpretation der einzelnen, aus den Komponenten gewonnenen Metriken.

Stichworte Monitoring, Cluster, Docker, Docker Swarm, Verteile Systeme, Prometheus, Consul

Abstract

And again in english.

Inhaltsverzeichnis

1. Einleitung

Das Internet wird immer mehr zum Dreh- und Angelpunkt unserer Gesellschaft. Um eine breite Masse zu erreichen, ist ein Internetauftritt unumgänglich geworden. Um jedoch nicht nur scheinbar einer breiten Öffentlichkeit zugänglich zu sein sondern tatsächlich leistungsfähige Infrastruktur bereitzustellen, die auch hohe Lasten sowie Zugriffszahlen verarbeiten kann, wird immer häufiger horizontal skaliert. Dies bedeutet einen zusätzlichen Verwaltungsaufwand, da mit jedem Skalierungsschritt auch neue zu betreuende Einheiten hinzukommen. An dieser Stelle soll diese Bachelorarbeit ansetzen und Lösungen für eine mögliche Überwachung sowie die Darstellung des aktuellen Status eines Clusters bieten. Die folgenden Fragen werden in dieser Bachelorarbeit *Konzeption und Implementierung eines Monitoring Systems für Docker Swarm Cluster* besprochen:

- Wie kann ein Cluster überwacht werden?
- Welche Metriken sind für den Betrieb wichtig?
- Wie kann eine Übersicht der wichtigsten Metriken erstellt werden?
- Welche Sicherheitsaspekte müssen beachtet werden und wie sieht eine beispielhafte Lösung aus?

Um Antworten auf diese Fragen zu erhalten, wurde ein auf Docker Swarm aufbauendes Testcluster eingerichtet. Dies wird mit Prometheus Überwacht und mit den aus Prometheus stammenden Daten wird prototypisch ein aktueller Zustand des Clusters dargestellt.

Im Kapitel 2 Grundlagen wird auf die grundsätzliche Funktion der Komponenten und deren Rolle innerhalb des Clusters eingegangen. Inhaltlich das spannendste Kapitel wird Kapitel 3, es werden Abhängigkeiten dargestellt und erörtert, erklärt wie der Status des Clusters bestimmt werden kann und es findet eine Evaluation der Metriken statt. Die praktische Umsetzung wird in Kapitel 4 beschrieben, aufgeteilt in die Bereiche: Installation des Testclusters (Abschnitt 4.1), Implementierung des Prototypen Panopticum (Abschnitt 4.3) und Eingehen auf die Sicherheit der Komponenten im Zusammenhang mit der vorgestellten Konfiguration (Abschnitt 4.4). Abschließend wird in Kapitel 5 ein Fazit gezogen und ein Ausblick auf die Weiterentwicklung sowie potentielle zukünftige Entwicklungen gegeben.

2. Grundlagen

Im Grundlagenkapitel wird der Aufbau des Testclusters beschrieben. Der Einsatzzweck der jeweiligen Komponente wird beschrieben um die Auswahl nachvollziehbar zu machen. Dabei wird auf die einzelnen Softwarekomponenten eingegangen und deren Besonderheit und Funktion für dieses Testcluster beschrieben. Auf eine ausführliche Erklärung der Komponenten wird verzichtet, da die Projekte sehr gut und aktuell dokumentiert sind.

Nach dem Lesen des Titels: *Konzeption und Implementierung eines Monitoring Systems für Docker Swarm Cluster* stellen sich einige Fragen:

- Was ist Monitoring?
- Was ist ein Cluster?
- Was ist Docker Swarm?

Diesen und weiteren Fragen geht Kapitel 2 nach. Im Folgenden werden die Gundlagen für den Aufbau und die Überwachung des Clusters beschrieben. Dabei richtet sich die Reihenfolge an die Installationshierarchie der einzelnen Komponenten.

2.1. Cluster

In dieser Bachelorarbeit wird unter einem Cluster ein Verbund von Computern verstanden, die sich nach außen als Einheit präsentieren. Diese besteht aus den Komponenten: Docker, Docker Swarm, Consul, Weavenet, GlusterFS. Dabei kommen die im folgenden beschrieben Produkte und Komponenten zum Einsatz.

2.1.1. Microservices Architektur

Microservice Architektur ist ein nicht definierter Begriff für eine Aufteilung von Software in kleine automome und funktionale Einheiten.
[FL15] [Fow14] [Ste15] [Bre00]

Die Microservice Architektur ist ein Msuter aus der informatik und Jeder Container bietet einen Service an. Es ist einfach Container auszutauschen. Es ist nicht wichtig auf welchem Host (Docker Host) ein Container läuft. Container können untereinander in einem gesondereten Netzwerk kommunizieren.

Container sollten möglichst nur Funktionalität für ihr Hauptziel beinhalten.[Far17d] Unterziele können von anderen Containern erfüllt werden. Dadurch ist eine hohe Austauschbarkeit und geringe kopplung gegeben.[Far17a]

2.1.2. Docker

Softwarelösung zur Container Virtualisierung. Sate-of-the-Art Microservices Software. Grundbaustein für das Cluster und Docker Swarm.

2.1.3. Docker Swarm

Docker Swarm ist eine systemeigene Clusterlösung für Docker. Es handhabt eine Ansammlung von Docker Hosts wie einen einzelnen, virtuellen Docker host und spricht diesen über die Docker API an. Weil Docker Swarm die Standard Docker API ausliefert, kann jedes Tool, das akutell bereits mit einem Docker Daemon kommuniziert den Swarm benutzen um transparent auf viele Hosts zu skalieren. Es unterstützt die folgenden Tools, ist jedoch nicht auf diese limitiert: - Dokku - Docker Compose - Docker Machtine - Jenkins

Daneben wird natürlich ebenfalls Docker selbst unterstützt und aufgrund der gleichen API kann Docker Swarm über die docker-eigene CLI administriert werden. [1]

Abgrenzung Swam mode von Docker Docker Swarm war/ist ein Produkt von Docker Inc. um einen Verbund von mehrere Docker Engines zu erstellen und verwalten. Bis zur Version 0.12 war dies nur mit Docker Swarm möglich, jetzt bringt Docker den Swarm Mode mit. Der Swarm Mode vereinfacht die Verwaltung von Services im Docker Cluster.

2.1.4. Consul

Consul ist ein verteilter *Key-Value Store*, eine *Service Discovery* und *Failure Detection* Software. In einem Docker Swarm Cluster stellt Consul eine Liste der registrierten Docker Swarm Instanzen bereit. Dabei wird ein *health*-Wert zu jeder Instanz hinzugefügt, der Aufschluss über den aktuellen Status gibt. Consul wählt mit Hilfe des Raft Konsensus Algorithmus die Instanz aus, die als Koordinator des Verbandes dient. Um diesen auszuwählen kommt der verteilte Konsensus Algorithmus *Raft*[OO14] zum Einsatz. Dieser wird in Exkurs 1 beschrieben.

In der Testinstallation dient Consul als reiner *Key-Value Store*. Service Discovery?! Docker Swarm legt darin gemeinsam genutzte Daten in ab. Dazu gehören wichtige Eigenschaften für das Cluster, wie z.B.: der Cluster Leader, die registrierten Nodes.

[1] Übersetzt aus der enlischen Dokumentation https://docs.docker.com/swarm/overview/

Exkurs 1: Raft Konsensus Algorithmus

Raft ist ein Konsensus Algorithmus, der ähnlich dem Paxos[Lam01] Algorithmus ist. Im Vergleich zu Paxos ist Raft einfacher zu verstehen und einfacher zu implementieren[OO14]. Dadurch versprechen sich die Entwickler geringere Fehleranfälligkeit und bessere Zuverlässigkeit als bei anderen Konsensus Algorithmen. Ein Merkmal von Raft ist, dass es immer einen *Leader* gibt, über den alle Aktionen ausgeführt werden. Dieser wurde von einer Mehrheit der beteiligten Knoten im Cluster gewählt. Ist dies nicht der Fall, bleibt/ist Consul funktionsunfähig.

2.1.5. WeaveNet

2.1.6. GlusterFS

GlusterFS ist ein verteiltes Dateisystem, dass auf FUSE[2] aufbaut. Im Testsystem replizieren alle GlusterFS Knoten alle Volumes. [16] Die Synchronisation läuft synchron. ¡-Überprüfen!!!

Das Besondere an GlusterFS ist, dass es mit Consumer Hardware betrieben werden kann. Dadurch ergeben sich wirtschaftlich erst einmal Vorteile.

2.2. Monitoring

Der Begriff Monitoring beschreibt die Protokollierung, Messung, Beobachtung oder Überwachung eines Vorgangs mit Hilfe von technischen Hilfsmitteln. Die regelmäßige Durchführung dieser ist ein zentrales Element um durch Vergleichen der Ergebnisse Schlussfolgerungen ziehen zu können. Vgl. [Wik16]

Damit wird der Prozess beschrieben, wie sichergestellt wird, dass ein System die gewünsche Funktion ausfüllt. Grundsätzlich wird dabei auf vorhandene Software zurückgegriffen. Da beim Monitoring Daten mit Zeitbezug anfallen, bieten sich Zeitserien Datenbanken auch TSDB genannt an. Für verteilte dynamische Systeme bietet sich Prometheus, das in Unterabschnitt 2.2.1 eingeführt wird an.

Um die gesammelten Daten darzustellen empfehlen sich die folgenden Anhaltspunkte: Eine zentrale Übersichtsseite auf der die wichtigsten Daten abgebildet werden, diese sollten zugunsten des Überblicks eine Bildschirmseite nicht überschreiten. Für detailiertere Informationen können weitere Übersichtsseiten nach Thema oder Aufgabe erstellt werden.

Als grobe Regel gilt, nicht mehr als 6 Graphen auf einem Dashboard und nicht mehr als 6 Plots in einem Graphen. Damit is laut [Far17b] eine gute Erkennbarkeit und Übersicht gewahrt. Jedes Team sollte ein eigenes Dashboard haben um den Zielen des Team gerecht zu werden, allerdings soltle nicht versucht werden jeder Person gerecht zu werden, damit die Teamziele gewahrt bleiben.

[2]FUSE https://www.kernel.org/doc/Documentation/filesystems/fuse.txt

Als Empfehlung gibt [Far17c] an, nicht auf anhieb zu versuchen das best mögliche Dashboard zu erstellen, sondern es als iterativen Prozess zu sehen in dem es immer weiter verbessert wird.

2.2.1. Prometheus

Prometheus ist ein Überwachungstool und eine Zeitseriendatenbank (Time Series Database) zur Überwachung von verteilten Systemen. Dieses wurde von Soundcloud entwickelt und erfreut sich einer wachsenden Benutzerbasis und sehr aktiven Community.

2.3. Sicherheit

Zur Absicherung wurden für alle Kommunikationswege eigene SSL-Zertifikate verwendet. Dadurch ist die CA in eigener Hand. Und es ist einfach weitere Zertifikate auszustellen oder für ungültig zu erklären. Die Absicherung erfolgt durch eine Überprüfung der Client-Zertifikate. Dadurch wird sichergestellt, dass nicht jeder eine Verbindung aufbauen kann, sondern nur jene mit einem gültigen, von der jeweiligen CA autorisierten Zertifikat. Der Datenverkehr findet verschlüsselt über SSL statt und dazu wird ebenfalls ein Zertifikat zur Authentifizierung benötigt um Daten aus dem System auszulesen.

Kritische Betrachtung Wenn ein Angreifer die CA unter Kontrolle hat, sind die Client-Zertifikate kein Schutz mehr. Allerdings ist es möglich die CA zu verschlüsseln, was einen Angriff deutlich erschwert.

3. Konzept

Dieses Kapitel soll eine Hinführung zur Problematik eines aggregierten Clusterstatuses sein. Um einen logischen Verbund von Hosts Cluster nennen zu können, kommt eine Vielzahl von verschiedener Software hinzu.

Zu Beginn wird ein verteilter Key-Value Store benötigt, der die Konfigurationen des Clusters speichert, aktualisiert und liest. Dabei ist wichtig, das auf diesen von allen aus Servern zugegriffen werden kann. Auf den Key-Value Store wird zugegriffen, wenn Verwaltungsaufgaben an Docker Swarm vorgenommen werden, so z.B.: starten eines Containers, hinzufügen eines weiteren Swarm Hosts. Teilweise ist es nicht möglich den Clusterstatus aus einzelnen unabhängigen Werten zu gewinnen, sondern nur aus im Zusammenhang verstandener Werte.

Dieser Zusammenhang muss beobachtet werden, denn der Ausfall von einigen Komponenten kann einen Ausfall des gesamten Clusters bedeuten, andere wiederum sind verkraftbar. Angenommen einer der drei Hosts fällt aus, so ist das ein verkraftbarer Verlust, auf den schnellst möglich reagiert werden sollte, das Cluster ist weiter voll funktionsfähig. Fällt zudem noch eine Instanz des Key-Value Stores aus, ist das gesamte Cluster manövrierunfähig.

3.1. Welche Clusterstatus gibt es?

Um deutlich und schnell zu erkennen, welchen Status das Cluster hat, werden drei Farben eingeführt: Rot, Orange, Grün. Deren Bedeutung sich intuitiv aus dem Straßenverkehr ableiten lässt.

🟢	**green**	Cluster ist **voll** Funktionsfähig.	*Keine Einschränkungen.*
🟠	**orange**	Cluster ist **teilweise** Funktionsfähig.	*Ein Eingriff wird empfohlen.*
🔴	**red**	Cluster ist **nicht** Funktionsfähig.	*Ein Eingriff ist notwendig.*

Tabelle 3.1.: Farbcodes Clusterstatus

3.2. Was bedeutet ein „grüner“ Clusterstatus?

Healthy oder grün ist der Status des Clusters, wenn alle Komponenten ihren Service anbieten, erreichbar sind und Anfragen in adäquater Zeit ausführen.

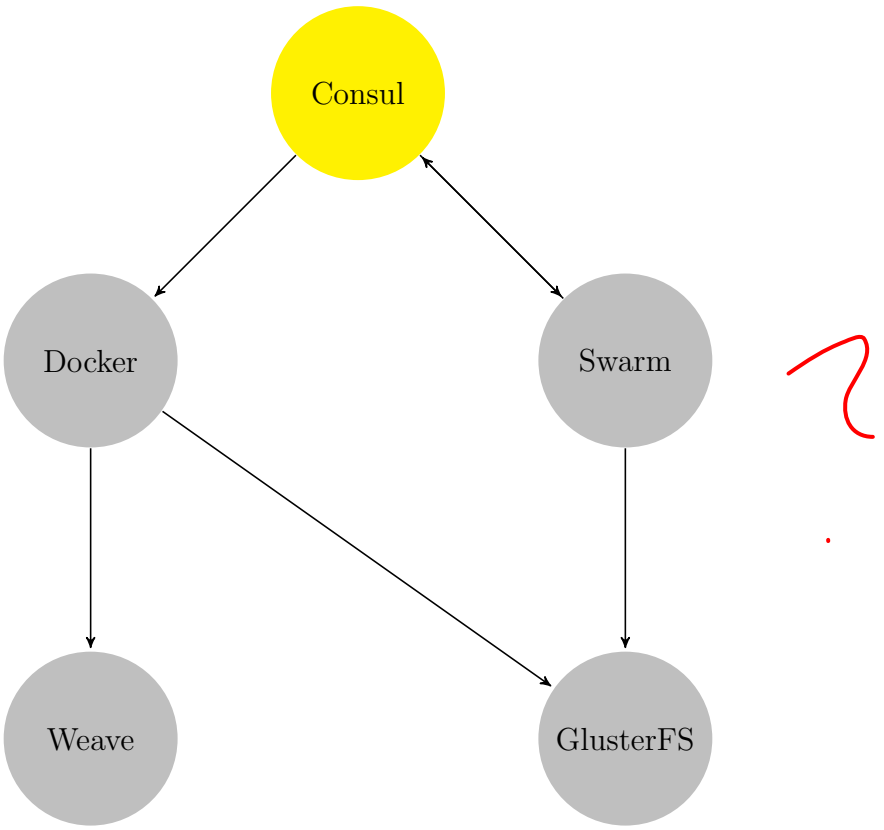

Abbildung 3.1.: Zusammenhänge der Services

So inkludiert dies, dass es einen Clusterleader gibt, dass es möglich ist Docker Container auf dem Swarm zu deployen, das verteilte Dateisystem GlusterFS beschreibbar und auslesbar ist und das Cluster von außen zu erreichen ist. Zudem sind sich alle Cluster-Knoten untereinander bekannt und können uneingeschränkt im Overlay-Netzwerk (Weave)(-¿ definieren) mit einander kommunizieren. Es können Dateien in den Mountpoints von Gluster verändert werden und diese werden synchronisiert.

Transitions

status uebergaenge

In Abbildung 3.2 ist der Übergang von einem in einen anderen Status dargestellt.
Wenn ein host ausfällt, dann warning state Wenn ein critical Service ausfällt dann critical state (Docker, Consul, Swarm, Gluster, Weave) Wenn ein non-critical service ausfällt dann warning state.

- Einzelne Komponenten und im Gesamten
- Wie hängen diese zusammen. Evtl. mit einer Hierarchie o.ä.

Ausfall Docker: Container laufen im schlimmsten Fall nicht mehr. Alles ist betroffen.

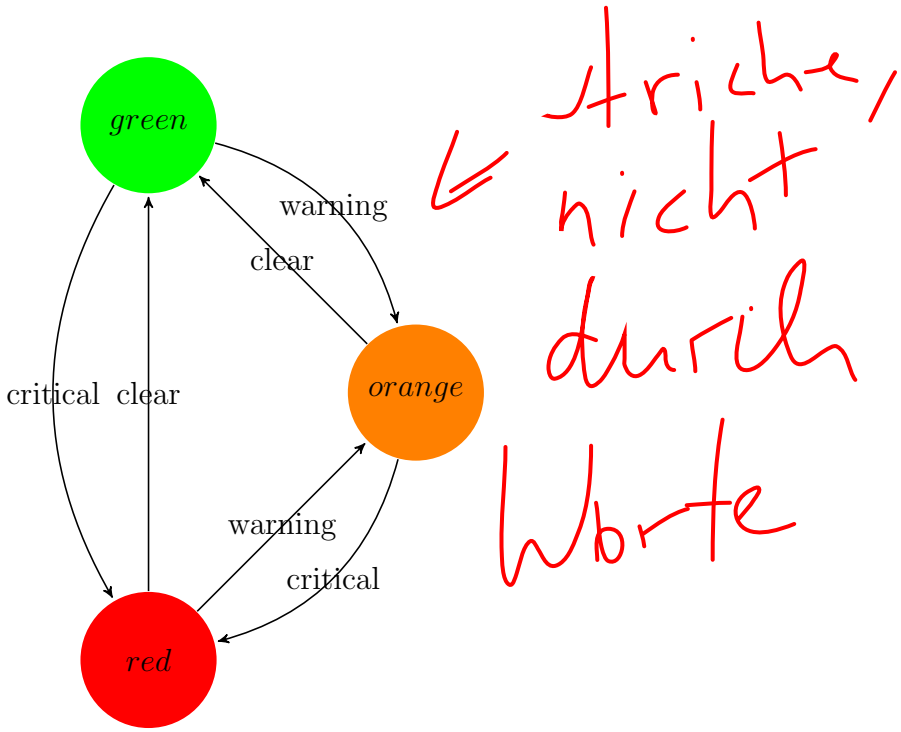

Abbildung 3.2.: State Machine of cluster states

Zusammenhänge der Services

Was bedeutet Ausfall von Docker?

- Container sind nicht mehr ausführbar
- Container lassen sich nicht mehr Verwalten?
- Es gibt Zombies.
- Swarm geht vermutlich nicht mehr

3.3. Welche Kenngrößen/Metriken sind wichtig und warum?

Woran erkenne ich, dass etwas nicht stimmt? Einer oder mehrere der folgenden Punkte werden nicht erfüllt: - Erreichbarkeit der einzelnen Knoten (critical, total ausfall) - Knoten kennen sich untereinander (critical, split brain möglich) - Knoten können untereinander kommunizieren (critical, split brain möglich) - Es gibt einen Clusterleader (critical?, es kann kein quorum gefunden werden und daher kann nicht aggiert werden.)

3.4. Was passiert, wenn einzelne Komponenten "degraden"?

Darstellen, welche Teile betroffen sind. Warum führt das zu Welchem Status. Beispiel: - Kein Leader: Docker Swarm kann nicht aggieren, daher ist das Cluster in einem kritischen Zustand.

3.5. Begründung für Tool

Metriken müssen in Zusammenhang gesehen werden. Dafür ist es nicht ausreichend die Metriken unabhängig darzustellen, sondern sie müssen im Zusammenhang gesehen werden. -¿ Dafür braucht es ein Tool, das diesen Part übernimmt und die Abhängigkeiten kennt und bewertet.

3.6. Ende:

Am Ende des Kapitels oder am Anfang der Umsetzung eine Liste mit Anforderungen/-Zielen der Thesis. Diese können im Fazit/Ergebnis beleuchtet werden.

4. Umsetzung

4.1. Testcluster mit Docker Swarm, Consul und GlusterFS

Im folgenden wird auf die Installation und Konfiguration des Testclusters eingegangen. Das Cluster umfasst vier Knoten, die durch physikalische Hosts repräsentiert werden. In Tabelle 4.1 werden die technischen Daten aufgelistet.

Name	Lenovo ThinkCenter
Prozessor	Intel® Core 2 Duo®CPU E6550 @ 2.33GHz
RAM	2 GB
HDD	74.5 GiB
Netzwerk	1 Gbit/s

Tabelle 4.1.: Hardwarekonfiguration der Nodes für das Testcluster

Debian Jessie[1] kommt als Betriebssystem zum Einsatz. Die weitere Software wurde sofern notwendig oder möglich als 3rd Party Quellen hinzugefügt. Docker ist in der Version 1.12.5 installiert. Um Docker im Cluster und mit Docker Swarm zu betrieben muss ein verteilter Key-Value Store verfügbar sein. Für das Testcluster wurde Consul ausgewählt. Consul ist in der Version 0.6.4 installiert und wurde als binary Release von HasiCorp heruntergeladen und für das Testcluster konfiguriert. Die Consul Nodes kommunizieren direkt mit einander, und wurden per IP auf der Firewall explizit für die Kommunikation freigegeben. Das Webinterface wurde so konfiguriert, dass es nur über einen Nginx Reverse Proxy auf Port 8501 zugänglich ist. Dieser ist nur für bestimmte vertrauenswürdige IP Adressen freigegeben. Zusätzlich muss sich ein Client mit einem TLS Zertifikat, ausgestellt von der Cluster-CA authentifizieren.

GlusterFS benutzt das Repository der Version 3.8, d.h. alle Aktuallisierungen in dieser Version werden bei einer Systemaktuallisierung ebenfalls mit berücksichtigt.

4.2. Evaluation Metriken

Um den Status für das Cluster bestimmen zu können, werden Kennzahlen benötigt die Aufschluss darüber geben, im weiteren Verlauf Metriken genannt. Zunächst eine

[1]Release Webseite von Debian Jessie https://www.debian.org/releases/jessie/

Auflistung über die verschiedenen Komponenten die solche Kennzahlen bereitstellen.
Linux

- Allgemeine Metriken
- Host / Node
  - CPU
  - Festplatten
  - Hauptspeicher und Swap
- GlusterFS
- Consul
- Docker Swarm / cAdvisor
- Weave
- Prometheus

In den jeweiligen Abschnitten werden die entsprechenden Metriken beschrieben inklusive eines Hinweises zu den Grenzwerten.

4.2.1. Allgemeine Metriken

Einige Kennzahlen sind für alle Exporter verfügbar und geben Aufschluss über die Funktionstüchtigkeit des jeweiligen Exporters.

up Zeigt an ob Prometheus den Exporter erreicht hat. Der Wert dieser Metrik ist 0 oder 1.

***_up** Der * steht für den Namen des jeweiligen Exporters. Zeigt an, ob dieser eine Abfrage an die Komponente ausführen konnte und eine Rückmeldung erfolgte. Der Wert dieser Metrik ist 0 oder 1.

4.2.2. Host / Node

4.3. Tool: Panopticon

Panopticon ist ein Tool, dass die einzelnen Komponenten des Clusters bewertet und einen aktuellen Status darstellt. Es lassen sich verschiedene Dienste Konfigurieren, die in die Bewertung mit aufgenommen werden. Dargestellt wird ein aggregierter Status aus den konfigurierten Services, der entweder „grün“, „orange“ oder „rot“ sein kann.

Der erste wichtige Schritt bestand darin, geeignete Metriken zu finden um eine Aussage über den Status des Clusters und der Komponeten zu geben.

4.3.1. gluster_exporter

GlusterFS hat zu beginn der Bachelorarbeit noch keine Instrumentierung für Prometheus gehabt. Daher habe ich mich entschieden einen Exporter selbst zu schreiben. Der gluster_exporter[2] ist in Go implementiert und instrumentiert die Informationen, die über das Gluster CLI abrufbar sind.

Ein Problem dabei ist, dass jede Anfrage an Gluster immer im gesamten Glusterverbund ausgeführt werden. Dadurch kommt es zu vergleichsweise hohen Latenzen ($\approx 1 - 5$ sec.). Das macht es notwendig, in Prometheus eine hohe Timeout- und Abfragezeit anzugeben. (> 30 sec.)

Prometheus liefert einige Bibliotheken mit um eigene Exporter zu implementieren unter anderen Java, Python und Go. Die Wahl fiel auf Go, da es leicht zu Deployen ist, das gesamte Programm als eine Binary ausgeliefert wird, zudem bietet Go mit seiner umfangreichen Standtardbibliothek ein sehr umfangreiches Toolset für Serverimplementierungen.

Die wichtigsten Metriken, die sich aus der Gluster CLI ableiten lassen sind `gluster_peers_connected` und `gluster_up`. Der Wert von *gluster_up* sagt aus, ob die Gluster CLI erreichbar war, die Abfrage erfolgreich war und ob das Ergebnis interfretiert werden konnte. Die mit dem abgefragten Peer verbundenen Peers werden in der Metrik *gluster_peers_connected* angegeben.

4.4. Sicherheit

[2]Github: gluster_exporter https://github.com/ofesseler/gluster_exporter

5. Fazit und Ausblick

Docker und Docker Swarm sind in den Kinderschuhen. Aus vielen Blogbeiträgen[1] wird klar, dass hier noch viel passiert. Es wird als unreif bezeichnet und die Verwendung in Produktionsumgebungen sollte gut überlegen werden.

Docker 1.13 bringt einige Veränderungen mit sich, die manche von den benutzten Applikationen überflüsssig macht. So ist Docker Swarm als Swarm Mode seit 1.12 in Docker integriert und Consul wird mit 1.13 als Key-Valuse Store und Service Discovery überflüssig.

ELK um die Logdaten an einem Ort zu sammeln, dann kann bei Fehlern schneller reagiert werden.

[1] Serverfault - Docker Swarm and Consul production setup recommendation http://serverfault.com/questions/748040/docker-swarm-and-consul-production-setup-recommendation

Literatur

[Bre00] Eric A. Brewer. „Towards Robust Distributed Systems“. In: *PODC*. Bd. 7. 2000. URL: http://awoc.wolski.fi/dlib/big-data/Brewer_podc_keynote_2000.pdf (besucht am 26. 01. 2017).

[Lam01] Leslie Lamport. „Paxos Made Simple“. In: *SIGACT News* 32.4 (Dez. 2001), S. 51–58. ISSN: 0163-5700. DOI: 10.1145/568425.568433. URL: http://research.microsoft.com/users/lamport/pubs/paxos-simple.pdf.

[Fow14] Martin Fowler. *Microservices - A Definition of This New Architectural Term*. 25. März 2014. URL: https://martinfowler.com/articles/microservices.html (besucht am 26. 01. 2017).

[OO14] Diego Ongaro und John Ousterhout. „In Search of an Understandable Consensus Algorithm“. In: *2014 USENIX Annual Technical Conference (USENIX ATC 14)*. 2014, S. 305–319. URL: https://www.usenix.org/conference/atc14/technical-sessions/presentation/ongaro (besucht am 10. 11. 2016).

[FL15] Martin Fowler und James Lewis. „Microservices: Nur Ein Weiteres Konzept in Der Softwarearchitektur Oder Mehr“. In: *Objektspektrum* 1.2015 (2015), S. 14–20. URL: https://www.sigs-datacom.de/uploads/tx_dmjournals/fowler_lewis_OTS_Architekturen_15.pdf (besucht am 28. 01. 2017).

[Ste15] Guido Steinacker. *Monolithen Und Microservices*. 2. Juni 2015. URL: https://www.informatik-aktuell.de/entwicklung/methoden/von-monolithen-und-microservices.html (besucht am 26. 01. 2017).

[16] *GlusterFS Architecture*. 2016. URL: https://github.com/gluster/glusterdocs/tree/8232306c91 (besucht am 12. 12. 2016).

[Wik16] Wikipedia. *Monitoring Begriffserklärung*. In: *Wikipedia*. Page Version ID: 160566642. 12. Dez. 2016. URL: https://de.wikipedia.org/w/index.php?title=Monitoring&oldid=160566642 (besucht am 23. 01. 2017).

[Far17a] Viktor Farcic. In: *The DevOps 2.1 Toolkit: Docker Swarm*. LeanPub, 1. Nov. 2017. Kap. Collecting Metrics and Monitoring The Cluster. URL: http://leanpub.com/the-devops-2-1-toolkit (besucht am 13. 01. 2017).

[Far17b] Viktor Farcic. In: *The DevOps 2.1 Toolkit: Docker Swarm*. LeanPub, 1. Nov. 2017. Kap. Monitoring Best Practices. URL: http://leanpub.com/the-devops-2-1-toolkit (besucht am 13. 01. 2017).

[Far17c] Viktor Farcic. In: *The DevOps 2.1 Toolkit: Docker Swarm*. LeanPub, 1. Nov. 2017. Kap. What Now? URL: http://leanpub.com/the-devops-2-1-toolkit (besucht am 13. 01. 2017).

[Far17d] Viktor Farcic. *The DevOps 2.1 Toolkit: Docker Swarm*. LeanPub, 1. Nov. 2017. 385 S. URL: http://leanpub.com/the-devops-2-1-toolkit (besucht am 13. 01. 2017).

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Glossar

Cluster ist eine Menge von Computern die miteinander agieren und sich als Gemeinschaft verstehen. Eine genauere Beschreibung befindet sich in Abschnitt 2.1. 3

Linux is a generic term referring to the family of Unix-like computer operating systems that use the Linux kernel. 13

Metrik ist eine Funktion, die eine Eigenschaft in einem Zahlenwert ausgedrück, darstellt. [1]. 10

[1] https://de.wikipedia.org/w/index.php?title=Softwaremetrik&oldid=160402025

Akronyme

API Application Programming Interface. 5

CLI Command Line Interface. 5

CPU Central Processing Unit. 13

TSDB Time Series Database. 6

A. Anhang