Logmanagement mit Graylog

Wissen sammeln

Es ist noch gar nicht lange her, da interessierte sich in den meisten Unternehmen allerhöchstens der IT-Betrieb für ein systematisches Logmanagement. Im Zuge von aktuellen technologischen und methodischen Entwicklungen wie Cloud Computing, Continuous Delivery und Continuous Deployment wird es für Unternehmen immer wichtiger, den Zustand eines Systems genau zu kennen und schnelle Fehlerdiagnosen zu erstellen.

von Richard Attermeyer

Je verteilter die Anwendungen, umso wichtiger ist die zentrale Logspeicherung. Immer mehr Anwendungen werden in privaten oder öffentlichen Clouds betrieben [1]. Die Anwendungen laufen also nicht mehr auf realen Servern, sondern in virtuellen Maschinen oder Containern. Mit diesem Trend zu leichtgewichtigeren Umgebungen verändert sich auch ihre Lebensdauer. Anstatt einen Server immer wieder manuell zu hegen und zu pflegen, werden Container oder virtuelle Maschinen direkt neu erstellt und alte Versionen gelöscht. Ohne ein zentrales Logmanagement gehen die Logs der Anwendungen also verloren und damit auch eine wichtige Informationsquelle, die gebraucht wird, um Development und Operations bei der Untersuchung von Fehlern zu unterstützen.

Gleichzeitig beobachten wir eine Entwicklung, die von Applikationsservern und monolithischen Applikationen weggeht hin zu verteilten Anwendungen, bei denen jede Business-Capability von einem eigenen Microservice umgesetzt wird. Gerade in verteilten Umgebungen ist es für den Betrieb und die Softwarewartung schwierig, Anfragen über verschiedene Knoten zu verfolgen und in den einzelnen Applikationslogs nach Hinweisen zu suchen. Traditionell müssen sich die Administratoren dazu an jedem Knoten anmelden und dann die Logs durchforsten oder sie manuell zusammentragen. Das ist zeitaufwändig und skaliert nicht.

Mittlerweile gibt es viele verschiedene Logmanagementlösungen sowohl im Open-Source-Bereich als auch von kommerziellen Anbietern. Dabei kann der Betrieb in der eigenen Infrastruktur oder auch in der Cloud erfolgen. Graylog ist ein Open-Source-Werkzeug, das eine integrierte Plattform für das Sammeln, Indizieren und Analysieren von Logdaten anbietet. Im Wesentlichen besteht das System aus dem Graylog-Web-Interface, den Graylog-Servern, den Elasticsearch-Knoten und einer Mongo-Datenbank (**Abb. 1**). Die Knoten lassen sich nach Bedarf skalieren. Zum Testen reicht ein System aus, bei dem alles in einem Knoten vereint ist. Der Graylog-Server ist das zentrale Element der Architektur, der sich um das Management der Elasticsearch-Indizes kümmert

und einen Abstraktionslayer bildet. Es wäre daher möglich, Elasticsearch gegen ein anderes System zu tauschen, das für die Analyse der Logdaten besonders geeignet ist.

Wie kommen meine Daten in den Graylog-Server?

Graylog unterstützt verschiedene Inputmechanismen. Standardmäßig werden vier verschiedene Formate oder Protokolle unterstützt: syslog, GELF, JSON/REST-URLs und RAW. syslog ist ein Standard zur Übermittlung von Logmeldungen und wird häufig von Systemkomponenten verwendet. Daher eignet sich dieser Transport recht gut, um systemnahe Logmeldungen an Graylog zu senden, z. B. HAProxy-Lognachrichten. Setzt man unter Graylog eine syslog-Eingabeschnittstelle auf, die auf Host und Port 10.20.30.6:11002 lauscht, kann HAProxy dorthin loggen, wenn man HAProxy wie folgt konfiguriert:

```
global
  log 10.20.30.6:11002 local0 info
```

GELF ist ein offenes von Graylog spezifiziertes Format, um strukturierte Lognachrichten zu unterstützen. Es ist für eigene Applikationen sehr zu empfehlen. Wenn der JSON-Input benutzt wird, lässt sich die JSON-Antwort einer REST-Ressource einfach auswerten. Diese Funktion kann dafür benutzt werden, JMX Beans abzufragen, die über Jolokia [2] exportiert werden. Die RAW-Eingangsschnittstelle ermöglicht es, Text über TCP/UDP entgegenzunehmen und mittels eigener Extraktoren zu parsen. So wird jedes beliebige textbasierte Logformat unterstützt.

Daten strukturieren

Logs sind häufig unstrukturiert. Die wirklichen Vorteile von Logmanagmentwerkzeugen erzielt man aber nur, wenn man mehr Struktur in die Logs bringt. Unstrukturierte Lognachrichten müssen daher häufig geparst und in einzelne Bestandteile zerlegt werden. Dies ist auch in Graylog über die Definition von Extraktoren möglich. Einfacher wird es, wenn die Logdaten schon in einem strukturierten Format ankommen. Graylog propagiert daher das Graylog Extended Log Format (GELF). Mit diesem ist es möglich, Logevents mit strukturierten Informationen anzureichern. Dazu gibt es einige vordefinierte Felder, wie *host*, *timestamp*, *level*, und zusätzliche Felder, die per Konvention mit einem Unterstrich beginnen. Das GELF-Format wird von einigen Applikationen schon nativ unterstützt. So bietet Docker einen GELF-Logtreiber an. Ansonsten gibt es für die gängigen Java-Logframeworks entsprechende GELF Appender. Diese lassen sich dann entweder direkt in der Anwendung oder im Applikationsserver konfigurieren.

Neben der eigentlichen Lognachricht übertragen diese Appender die Werte aus dem Mapped Diagnostic Context (MDC) in die GELF-Nachricht. Im MDC lassen sich zusätzliche Metainformationen zu einem Logeintrag ablegen. Wer Log4j 2 nutzt, kann dazu auch den Threadcontext verwenden. Besonders interessante Daten für den MDC sind jene, die es ermöglichen, verschiedene Nachrichten von unterschiedlichen Systemen zu korrelieren, etwa Logausgaben vom Load Balancer und Applikationsserver, oder Daten, die in den Kontext einer Logausgabe gesetzt werden können, etwa URLs von REST-Ressourcen.

Nutzung von Unique IDs

Unique IDs können für unterschiedliche Zwecke zum Einsatz kommen. Man kann beispielsweise mit ihnen eine Anfrage über verschiedene Systeme hin verfolgen. Das folgende Skript zeigt, wie man mittels HAProxy eine *UniqueId* injiziert:

```
frontend www-https
  # adding unique id for all requests that can be used for logging
  acl existing-unique-id req.hdr(X-Unique-ID) -m found
  http-request set-header X-Unique-ID %{+X}o\
                %ci:%cp_%fi:%fp_%Ts_%rt:%pid unless existing-unique-id
```

In diesem Beispiel wird ein HTTP-Header *X-Unique-ID* erzeugt, falls er noch nicht existiert. Der letzte Punkt ist wichtig, wenn ein erster Request von außen Requests zwischen Subsystemen auslöst. Da eine Unique ID nur einmalig generiert wird, kann man innerhalb der Logmanagementsoftware alle Logausgaben nach ihr filtern. Dies setzt allerdings voraus, dass die eigene Anwendung diese Information in einem Message Diagnostic Context setzt und dann bei weiteren Aufrufen propagiert. Das kann man selbst implementieren oder auch mithilfe einer Library wie TracEE [3]. Neben einer selbst generierten Unique ID eignen sich auch andere IDs, wie eine JMS Message ID. Wer Lösungen für größere Umgebungen benötigt, sollte sich in diesem Zusammenhang Zipkin [4] anschauen. Je nach Anwendungsfall kann auch der Einsatz von Tools für Application Performance Management sinnvoll sein, um herauszufinden, wo genau sich die Engstellen bei der Anfragebearbeitung befinden.

Mit Extraktoren, Konverter und Konventionen vereinheitlichen

Eine Aufgabe beim Logmanagement ist die Normalisierung. Typische Tasks sind dabei das Parsen und Konvertieren von Datums- und Zeitfeldern und das Vereinheitlichen von Feldnamen, die semantisch gleichen Inhalt enthalten. Wir haben schon gesehen, wie man Streams für HTTP-Codes erstellt und Korrelation über

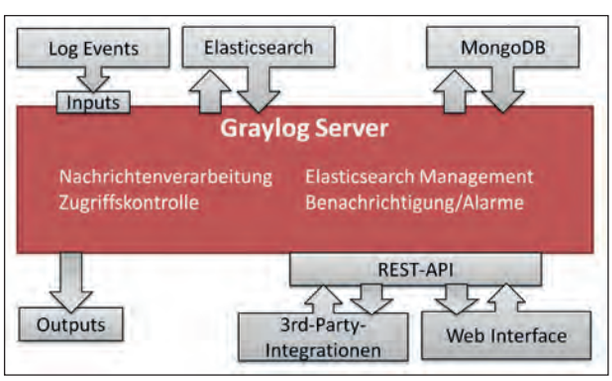

Abb. 1: Die Graylog-Architektur: Der zentrale Bestandteil ist der Graylog-Server. Er kümmert sich um die Suchindizes und bildet eine Abstraktionsebene

eindeutige IDs herstellen kann. Leider ist es so, dass jede Anwendung das Logging autonom realisiert. So sind HTTP-Codes bei dem einen System im Feld *http_response_code* enthalten, bei einem anderen System im Feld *HTTP_CODE* und beim dritten im Feld *status_code*. Bei Graylog lässt sich das Logging z. B. mittels Drools realisieren. Dies kann aktuell zwar noch nicht über die Oberfläche eingestellt werden, mit dem in Graylog Version 2 verfügbaren Message-Processing-Pipeline-Feature soll sich das aber ändern. Hat man die einzelnen Systeme unter Kontrolle, sollte man Konventionen für die Benennung von entsprechenden Feldern vereinbaren, um die Nachbearbeitung so einfach wie möglich zu halten.

Ebenso wichtig ist es, Lognachrichten zeitlich richtig zuordnen zu können. Es gibt sehr unterschiedliche String-Repräsentationen für Datums- und Zeitwerte. Über einen Konverter ist es möglich, diese extrahierten Werte in ein einheitliches Timestamp-Format zu konvertieren.

Ein weiterer wichtiger Punkt beim Logmanagement ist der Datenschutz. Wenn immer mehr Menschen auf Logs zugreifen können, müssen personenbezogene Daten anonymisiert werden. Dazu zählt z. B. die IP-Adresse. Auch das lässt sich über einen entsprechenden Konverter in Graylog realisieren.

Streams zum Routen, Alarmieren und zur Zugriffskontrolle

Mit Streams lassen sich Nachrichten in Echtzeit in Kategorien einordnen. Dabei verwendet man für die Definition des Streams die gleichen Abfragen, die man für die Suche von Nachrichten verwendet. Ein solcher Stream kann im Anschluss für zwei wichtige Anwendungsfälle genutzt werden: Alarmierung und Zugriffskontrolle. Eine Alarmierungsfunktion sieht dann wie folgt aus: Man definiert einen Stream für HTTP-Fehler (400er- und 500er-Fehlercodes). Daraufhin kann man auf diesem Stream eine Bedingung definieren, die eine Benachrichtigung auslöst, wenn z. B. innerhalb von fünf Minuten mehr als zehn Nachrichten in den Stream einsortiert werden. Der Zugriff auf Logs wird bei Graylog 2 über Rollen geregelt. Eine Rolle gewährt Rechte auf bestimmte Streams. So kann man unterschiedliche Streams und Rollen für Entwickler und Administratoren definieren.

Features von Graylog 2: Dashboards und Erweiterungen

Graylog 2 besitzt auch die Möglichkeit, Dashboards zu erzeugen. Damit sind grafische Auswertungen zu Abfragen auf den Logdaten möglich, wie etwa Antwortzeiten oder die Anzahl an Ausnahmen in einer Anwendung pro Umgebung. Die Funktionalität von Graylog reicht häufig für einfache Dashboards aus. Allerdings sind spezialisierte Werkzeuge wie Grafana für komplexere und ansprechendere Darstellungen besser geeignet.

Außerdem lässt sich Graylog 2 einfach über Plug-ins oder Content Packs erweitern. Content Packs sind Zusammenfassungen von Inputs, Extractors, Streams, Dashboards und Ausgabekonfigurationen. Für eine Anzahl an Infrastrukturkomponenten stehen schon entsprechende Content Packs auf dem Graylog Marketplace [5] zur Verfügung. Content Packs sind aber auch eine Möglichkeit, aktuelle Konfigurationseinstellungen eines Servers zu exportieren, zu sichern und zu versionieren. Bei einem Content Pack handelt es sich um eine einfache JSON-Datei. Es eignet sich auch, um die Konfiguration von Graylog zwischen verschiedenen Installationen zu propagieren.

Indexfehler vermeiden

Graylog nutzt zum Speichern von Lognachrichten Elasticsearch. Der entsprechende Elasticsearch-Index wird dynamisch erzeugt. In der Dokumentation von Graylog empfehlen die Autoren deshalb, sich konsequent für einen Typ pro Feld zu entscheiden und dabei zu bleiben. Leider hat man über Graylog keinen direkten Einfluss darauf. Elastisearch versucht den Typ eines Felds in der Nachricht nämlich automatisch zu bestimmen. Hier liegt ein kleiner Stolperstein. Wenn der erste Wert numerisch ist, setzt Elasticsearch den Typ des Felds auf „numerisch“. Wenn die Werte aber eigentlich Strings sind, kommt es zu Indexfehlern, wenn nachfolgend Nachrichten mit beliebigen Zeichen für das entsprechende Feld gespeichert werden sollen. Dieses Problem kann man umgehen, indem man in der Elasticsearch-Konfiguration Dynamic Templates [6] definiert und den Typ des problematischen Felds direkt auf „String“ einstellt.

Fazit

Zu wissen, ob Applikationen rund laufen, und schnelle Fehlerdiagnosen sind für Unternehmen essenziell wichtig. Logs sind eine wichtige Quelle für diese Informationen. Einerseits ist eine Datenzentrale heute wichtiger denn je. Andererseits ist aber auch der Einstieg ins Logmanagement viel einfacher geworden. Für Betreiber von verteilten oder Cloud-basierten Anwendungslandschaften ist daher jetzt ein guter Zeitpunkt, um Logmanagement als wichtigen Baustein zu etablieren. Mit Graylog 2 steht ein einfach zu administrierendes System bereit.

Richard Attermeyer arbeitet als Senior Solution Architect bei der OPITZ CONSULTING Deutschland GmbH. Er beschäftigt sich seit vielen Jahren mit der Architektur und Implementierung von Anwendungen im agilen Umfeld und fokussiert sich dabei aktuell auf moderne Architekturansätze rund um Microservices, Cloud, DevOps und Continuous Delivery.

Links & Literatur

[1] Bitkom Cloud Monitor 2015: https://www.bitkom.org/Publikationen/2015/Studien/Cloud-Monitor-2015/Cloud-Monitor-2015-KPMG-Bitkom-Research.pdf

[2] Jolokia: https://jolokia.org/

[3] TracEE: http://tracee.io/

[4] Zipkin: https://github.com/openzipkin/zipkin

[5] Graylog Marketplace: https://marketplace.graylog.org/

[6] Elasticsearch, Customizing Dynamic Mapping: https://www.elastic.co/guide/en/elasticsearch/guide/current/custom-dynamic-mapping.html