Institut für Parallele und Verteilte Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Masterarbeit

Cloud Service Monitoring mit TOSCA

Manuel Wiedenhöfer

Studiengang:	Informatik
Prüfer/in:	Prof. Dr.-Ing. habil. Bernhard Mitschang
Betreuer/in:	Dipl.-Inf. Mathias Mormul
Beginn am:	1. Dezember 2017
Beendet am:	1. Juni 2018

Kurzfassung

Aktuelle Begriffe und Konstrukte wie Predictive Maintenance, Self-Healing oder Machine Learning versprechen viele Vorteile für Unternehmen aus allen Wirtschaftsbereichen. Grundlegend dafür ist das vollumfängliche Sammeln und Auswerten enormer Datenmengen, dass unter dem Sammelbegriff Big Data zusammengefasst wird. Neben der Überwachung der Komponenten, muss dabei gleichzeitig die immer komplexer werdende IT-Infrastruktur flexibel gestaltet werden.

Die von der Universität Stuttgart entwickelte Laufzeitumgebung OpenTOSCA, erlaubt das automatisierte Provisionieren von Applikationen auf Basis des TOSCA Standards. Dies ermöglicht eine Flexibilisierung und Automatisierung der IT-Infrastruktur. Durch die automatisierte Integration der Monitoring-Komponente in die provisionierten Applikationen, wird das Sammeln der Daten ohne zusätzlichen Aufwand gesichert.

Die vorliegende Arbeit beschreibt die Integration eines Monitoringsystems in das bestehende OpenTOSCA Ökosystem. Hierbei werden vier Monitoringsysteme (Splunk, ELK-Stack, TICK-Stack, Prometheus) auf die Integration von OpenTOSCA, anhand aufgestellter Anforderungen, untersucht. Es folgt die konzeptionelle Beschreibung und Integration des Monitoringsystems (TICK-Stack). In Kooperation mit der Robert Bosch GmbH, wurde der TICK-Stack in einen bestehenden Projekt-Testaufbau erfolgreich implementiert.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Listings

1 Einleitung

„Daten sind das neue Gold“ hat sich in den letzten Jahren zu einem festen Satz etabliert und wird häufig in Zusammenhang mit Plattformen wie Facebook, Google sowie weiteren Unternehmen genannt[Weia]. Er soll unter anderem die Erhebung personenspezifischer Daten wie, Alter, Geschlecht, Interessen und politische Einstellungen beschreiben, welche eine gezielte und individualisierte Werbung ermöglichen. Ebenso profitieren Unternehmen von der Sammlung anfallender Systemdaten, um Ausfälle der IT-Infrastruktur zu vermeiden oder präventiv auf diese reagieren zu können[its].

Doch nicht nur für die Unternehmens-IT ist die Sammlung von Daten relevant. Auch im Produktionsbereich wird vermehrt der Nutzen einer durchgehenden Sammlung der Daten wahrgenommen, wie das Schlagwort Industrie 4.0 veranschaulicht[Lie]. Die Verkürzung der Produktionszyklen und die immer kleiner werden Stückzahlen, sind nur eines der Ziele. Selbstständige, intelligente Produktionsabläufe optimieren hierbei Produktionsprozesse, um Ausfälle der Produktion zu vermeiden. Vergleichsweise unrentable Kleinserien und Produktionen bis zur Losgröße Eins, werden durch diese Flexibilisierung ermöglicht[Gne14]. Für die Realisierung sogenannter intelligenter Fabriken ist es allerdings notwendig, anfallende Daten, wie Messungen der Roboter oder die Statuserfassung der Systeme, zu sammeln, auszuwerten und bei Komplikationen reagieren zu können. Dies erfordert ein Monitoringsystem, welches die anfallenden Daten erfasst und für eine weitere Verarbeitung zur Verfügung stellt.

Ziel dieser Arbeit ist das Implementieren eines Monitoringsystems in das bestehende OpenTOSCA Ökosystem. OpenTOSCA ist eine von der Universität Stuttgart entwickelte Laufzeitumgebung zur automatischen Provisionierung und Verwaltung von Cloud-Applikationen. Die Modellierung der Anwendungen erfolgt über das auf dem OASIS[1] Standard basierenden TOSCA-Archiv. Dies ermöglicht das Beschreiben von Cloud-Applikationen in einem portablen, standardisierten und maschinenlesbaren Format[BBKL14]. Cloud-Applikationen die mit OpenTOSCA provisioniert werden, sammeln durch die Integration des Monitoringsystems somit automatisch anfallende Daten und ermöglichen eine direkte Weiterverarbeitung. Zusätzlich profitieren die immer komplexer werdenden Systeme von der Automatisierung und Flexibilisierung, die mit der Nutzung von OpenTOSCA einhergeht[BBKL14].

Die Integration eines Monitoringsystems in das OpenTOSCA Ökosystem setzt bestimmte Anforderungen voraus. Im Rahmen dieser Arbeit werden hierzu vier aktuelle Monitoringsysteme (Splunk, ELK-Stack, TICK-Stack und Prometheus) anhand fest definierter Anforderungen evaluiert. Die anschließende Implementierung des TICK-Stacks in eine reale Produktionsumgebung, verdeutlicht die Vorteile eines Monitoringsystems. In Kooperation mit der Robert Bosch GmbH wurde der TICK-Stack in den bestehenden Projekt-Testaufbau der Arena 2036 implementiert.

[1] OASIS: Organization for the Advancement of Structured Information Standards

Die Arena 2036[2] ist laut eigenen Angaben die größte und führende Forschungsplattform für Mobilität in Deutschland. Sie fungiert als Schnittstelle zwischen Wissenschaft und Industrie und dient als Impulsgeber für den nachhaltigen Automobilbau[AREa]. Finanziert wird das Projekt durch Forschungseinrichtungen wie das Deutsche Zentrum für Luft- und Raumfahrt oder das Fraunhofer-Institut für Produktionstechnik und Automatisierung, aber auch durch Industrieunternehmen wie die Robert Bosch GmbH, Daimler AG und Siemens AG.

Die Robert Bosch GmbH arbeitet derzeit an zwei Projekten in der Arena 2036. Das Forschungsprojekt „Leichtbau durch Funktionsintegration (LeiFu)“ mit dem Ziel Leichtbaustrukturen für hochintegrierte Fahrzeugkomponenten zu erforschen[AREb]. Das zweite Projekt „Forschungsfabrik: Die wandlungsfähige (Fahrzeug-) Forschungsproduktion (ForschFab)“ hat das Ziel, ein wandlungsfähiges Produktionskonzept zu entwickeln. Dafür werden unter anderem Lösungsansätze für zukünftige Anwendungsszenarien mit dem schutzzaunlosen Produktionsassistent APAS[3] gesucht[AREb].

Die vorliegende Arbeit entspricht folgender Gliederung:

Kapitel 2 – Stand der Technik: In diesem Kapitel werden die Grundlagen beschrieben, die für das Verständnis der weiteren Arbeit erforderlich sind.

Kapitel 3 – Anforderungen an das Monitoringsystem: Ausgangspunkt der Evaluation der Monitoringsysteme bilden bestimmte funktionale und nicht-funktionale Anforderungen, welche dieses Kapitel beschreibt. Zusätzlich werden Anforderungen für eine Integration in den Projekt-Testaufbau in der Arena 2036 definiert.

Kapitel 4 – Evaluation Monitoringsysteme: Nachdem die Anforderungen an die Monitoringsysteme aufgestellt sind, werden die vier Monitoringsysteme (Splunk, ELK-Stack, TICK-Stack und Prometheus) anhand der aufgestellten Anforderungen evaluiert.

Kapitel 5 – Konzept: In diesem Kapitel werden verschiedene Konzepte zur Integration des Monitoringsystems in das OpenTOSCA Ökosystem vorgestellt und die jeweiligen Vor- und Nachteile beschrieben.

Kapitel 6 – Implementierung: Die Implementierung der in Kapitel 5 aufgezeigten Konzepte wird in diesem Kapitel vorgestellt.

Kapitel 7 – Evaluation: Dieses Kapitel unterzieht sich der Prüfung des zuvor implementierten Monitoringsystems in OpenTOSCA. Zusätzlich wird der in der Arena 2036 eingesetzte TICK-Stack evaluiert.

Kapitel 8 – Zusammenfassung und Ausblick: Das abschließende Kapitel fasst die Ergebnisse der Arbeit zusammen und stellt einen Ausblick dar.

[2] https://www.arena2036.de
[3] https://www.bosch-apas.com

2 Stand der Technik

In dieser Arbeit wird die Integration eines Monitoringsystems in das bestehende OpenTOSCA Ökosystem beschrieben. Zum besseren Verständnis wird in Kapitel 2.1 der OASIS Standard TOSCA vorgestellt und in Kapitel 2.2 die, von der Universität Stuttgart entwickelte, TOSCA Laufzeitumgebung OpenTOSCA beschrieben. Kapitel 2.3 beschreibt das allgemeine Vorgehen bei der Einführung eines Monitoringsystems, die Unterscheidung zwischen agent-based und agentless Monitoringsystemen und den Vergleich von zentralen und dezentralen Monitoringsystemen.

2.1 TOSCA

Enterprise-Applikationen sind komplexe Anwendungen, die in der Regel aus verschiedenen Komponenten bestehen, die durch verschiedene Abhängigkeiten in Verbindung stehen. Beispielsweise läuft ein Webserver auf einem Betriebssystem oder eine Anwendung greift auf eine externe Datenbank zu. Enterprise-Applikationen sind außerdem häufig modular entwickelt, wodurch sie von den Vorteilen von Cloudtechnologien (Elastizität, Skalierbarkeit, hohe Verfügbarkeit) profitieren. Gleichzeitig steigt aber auch der Wartungsaufwand und die Fehleranfälligkeit, da die einzelnen Komponenten oft nur schlecht dokumentiert und die Verwaltung über Bash-Skripte oder komplett manuell abläuft[BBKL14]. Um den Enterprise-Applikationen weiterhin die Vorteile des Cloud Computing zu ermöglichen und gleichzeitig den Verwaltungsaufwand zu verringern, wurde eine Modellierungssprache entwickelt. Diese soll Anwendungen portabel, standardisiert und in einem maschinenlesbaren Format beschreiben[BBKL14].

Die Organization for the Advancement of Structured Information Standards (OASIS)[1] hat in diesem Zusammenhang Topology and Orchestration Specification for Cloud Applications (TOSCA) entwickelt. TOSCA ist eine XML-basierte Modellierungssprache mit dem Ziel, Anwendungsstrukturen formalisiert darzustellen (Application Topology) und die Verwaltungsaufgaben in sogenannten Plänen zu beschreiben (Management Plans). Dadurch gewährleistet TOSCA das automatisierte Bereitstellen und Löschen von Applikationen oder das Sichern kompletter Applikationen. Ermöglicht wird dies durch die beiden Strukturelemente Application Topology und Management Plans[BBKL14].

Application Topology beschreibt alle Komponenten und wie diese zueinander in Beziehung stehen. Jeder Node (Knoten) besitzt dabei eine Liste von Operatoren, um sich selbst zu verwalten. Dadurch ist die Topologie nicht nur eine Beschreibung der Komponenten und deren Beziehung, sondern auch eine Beschreibung der Verwaltungsmöglichkeiten[BBKL14]. Das Strukturelement Management Plans bezieht sich auf diese Verwaltungsmöglichkeiten, um ein automatisiertes Bereitstellen, Konfigurieren und Verwalten der Anwendungen zu ermöglichen[BBKL14]. Abbildung

[1] https://www.oasis-open.org

2.1 visualisiert dieses Grundkonzept. Die Application Topology enthält sogenannte Nodes, die wiederum Bezug auf andere Nodes nehmen können. Management Plans werden dabei durch sogenannte Messages gestartet und rufen die Operatoren der Nodes auf[BBKL14].

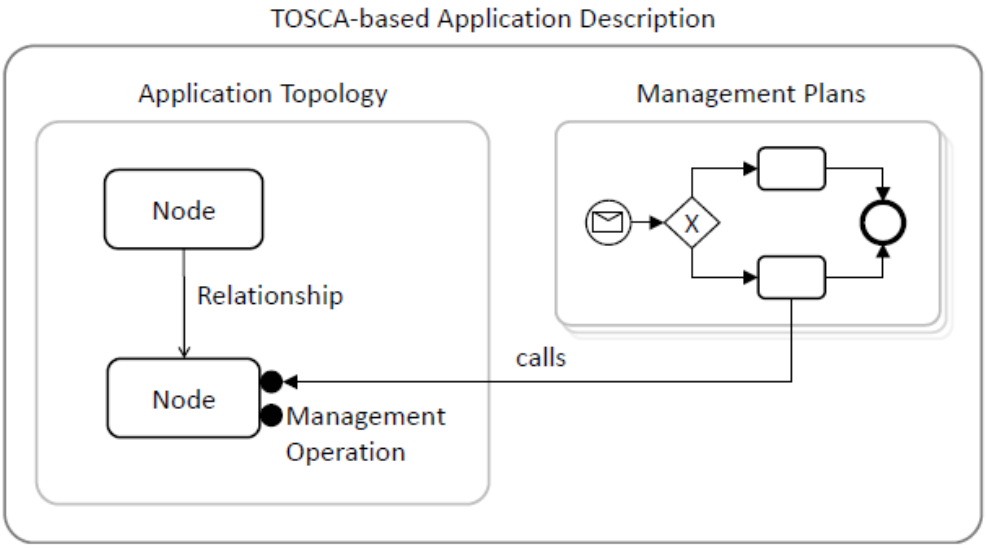

Abbildung 2.1: Aufbau TOSCA Quelle: Binz et al.[BBKL14]

2.1.1 Cloud Service Archive (CSAR)

Zur Beschreibung komplexer Cloud-Applikationen verwendet TOSCA das Cloud Service Archive, kurz CSAR. Eine CSAR-Datei beinhaltet ein Metamodell zur Beschreibung der Applikation und enthält gleichzeitig benötigte Daten wie Images oder Bash-Skripte. Die CSAR-Datei besteht aus vier Komponenten, siehe Abbildung 2.2. Das Topology Model ist ein gerichteter Graph, welcher aus sogenannten Node Types und Relationship Types besteht. Node Types dienen zur Beschreibung der Komponenten. Relationship Types beschreiben dagegen die Beziehungen zwischen den Nodes und mögliche Abhängigkeiten[BCK+16]. Zur Verwaltung und Provisionierung der Applikationen ist die Komponente Artifacts notwendig. Diese können beispielsweise ein Bash-Skript oder eine SQL-Datei beinhalten. Mithilfe sogenannter Management Plans können Management-Funktionalitäten abgebildet werden. Beispielsweise kann das Skalieren einer Applikation oder das Migrieren auf einen anderen Provider beschrieben werden[BCK+16].

Die oben beschriebenen Komponenten werden alle in ein portables Format (CSAR-Datei) gebündelt und können von jeder TOSCA Laufzeitumgebung, wie zum Beispiel OpenTOSCA, interpretiert und in Form neuer Instanzen bereitgestellt werden[BCK+16].

2.1.2 TOSCA-Konzept

Abbildung 2.3 beschreibt das grundlegende TOSCA-Konzept. Ausgehend von einer Metabetrachtung, wird zwischen Type, Template und Instance unterschieden. Dabei kann Type mit einer

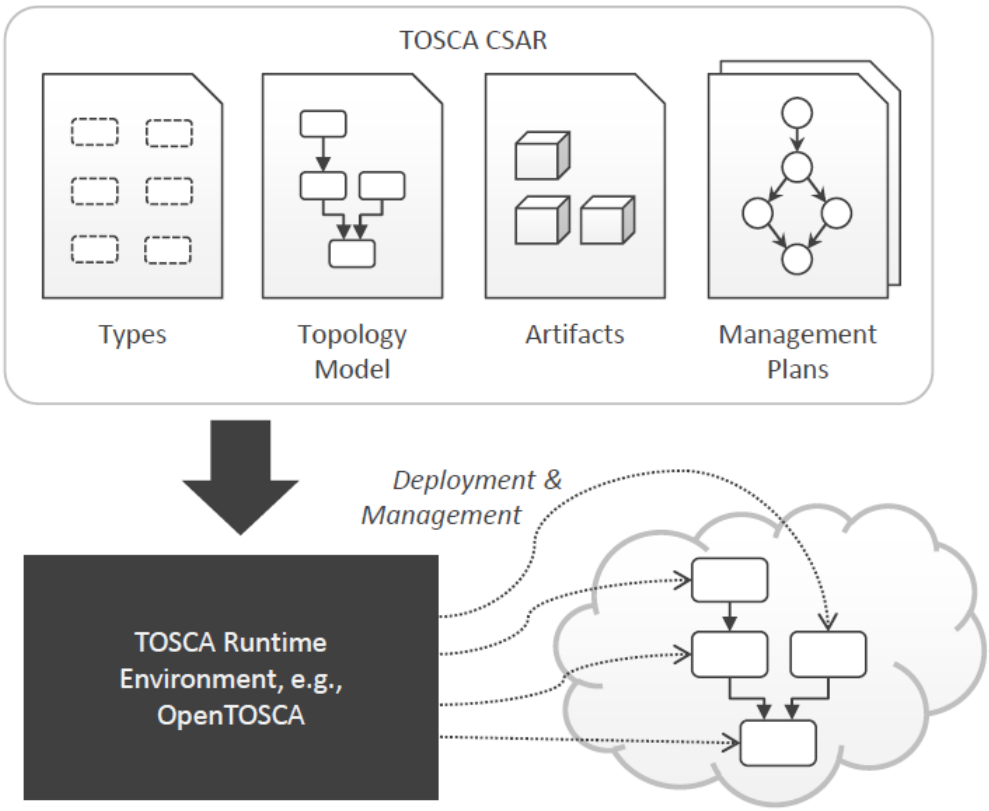

Abbildung 2.2: Aufbau CSAR-Datei Quelle: Breitenbücher et al.[BCK+16]

abstrakten Klasse in der Programmiersprache Java verglichen werden. Das Template entspräche dabei einer konkreten Klasse, welche wiederum von der abstrakten Klasse erbt. Auf der Meta-Ebene der Types, werden jeweils die Semantik, Eigenschaften und Schnittstellen spezifiziert. Instances entsprechen einer konkreten Instanz, wie beispielsweise eine virtuelle Maschine[BCK+16].

Zur Verdeutlichung folgt ein Beispiel mit zwei NodeTemplates. Es existiert ein Node Template mit dem Node Type PHP und ein NodeTemplate mit dem Node Type MySQL Datenbank. Diese beiden NodeTemplates sind über ein RelationshipTemplate miteinander verbunden. Das RelationshipTemplate wiederum ist vom Relationship Type [BCK+16]. Node Types und Relationship Types können Eigenschaften definieren. So kann beispielsweise der Node Type MySQL Datenbank die Eigenschaften *Benutzername, Passwort* und *Port* definieren. Ebenfalls können Types Schnittstellen spezifizieren. Der Node Type Apache Webserver kann beispielsweise die Schnittstelle deployApplication spezifizieren, welche über den Management Plan oder die TOSCA Laufzeitumgebung aufgerufen werden kann, um eine neue Instanz zu generieren[BCK+16].

Node Types und Relationship Types können über den Type Implementation sogenannte Artifacts integrieren. Artifacts können beispielsweise Installationsskripte oder komplette Systemimages beinhalten. In TOSCA werden Artifacts als ArtifactTemplate bezeichnet und in zwei Kategorien (DeploymentArtifact und ImplementationArtifact) unterteilt. DeploymentArtifacts implementieren Geschäftslogik, wie beispielsweise die Binärdateien eines Apache Webservers, die für die Installation notwendig ist. ImplementationArtifacts implementieren dagegen Management-

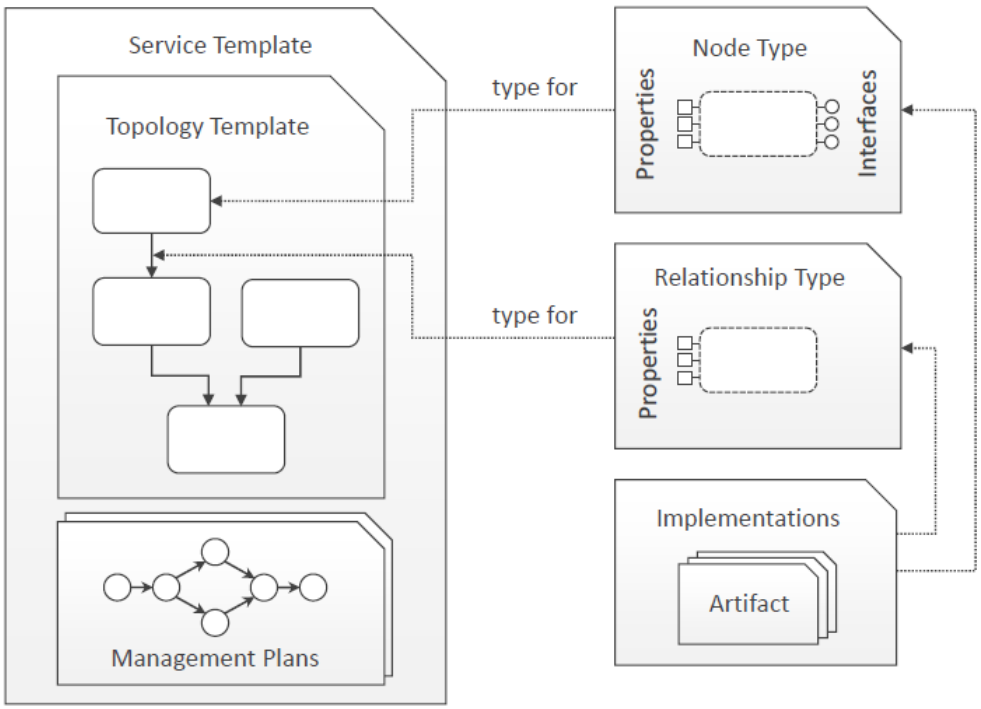

Abbildung 2.3: TOSCA Modell Quelle: Breitenbücher et al.[BCK+16]

Funktionalitäten, die für eine Managementoperation notwendig sind. Beispielsweise wäre das ImplementationArtifact eines Webservers ein Installationsskript, welches den Webserver installiert[BCK+16].

Unter dem Oberbegriff Service Template, werden neben oben beschriebenen Topology Templates, sogenannte Management Plans gefasst. Diese implementieren Mangagement-Funktionalitäten der Applikation, wie beispielsweise das Skalieren von Komponenten oder die Migration der kompletten Applikation zu einem anderen Cloud Provider[BCK+16].

2.2 OpenTOSCA

OpenTOSCA[2] ist eine, von der Universität Stuttgart[3] entwickelte, Laufzeitumgebung für TOSCA-basierende Cloud-Anwendungen. Abbildung 2.4 beschreibt die drei Hauptkomponenten, aus denen das OpenTOSCA Ökosystem besteht. Das TOSCA Modellierungstool Winery wird zum Modellieren der Applikation verwendet. Diese kann dann über die sogenannte TOSCA Runtime beispielsweise auf einem Amazon Web Services[4] Server bereitgestellt werden. Zum Instanziieren neuer Anwendungen wird das Self-Service Portal Vinothek verwendet[BCK+16].

[2] https://www.opentosca.org
[3] https://www.uni-stuttgart.de
[4] https://aws.amazon.com

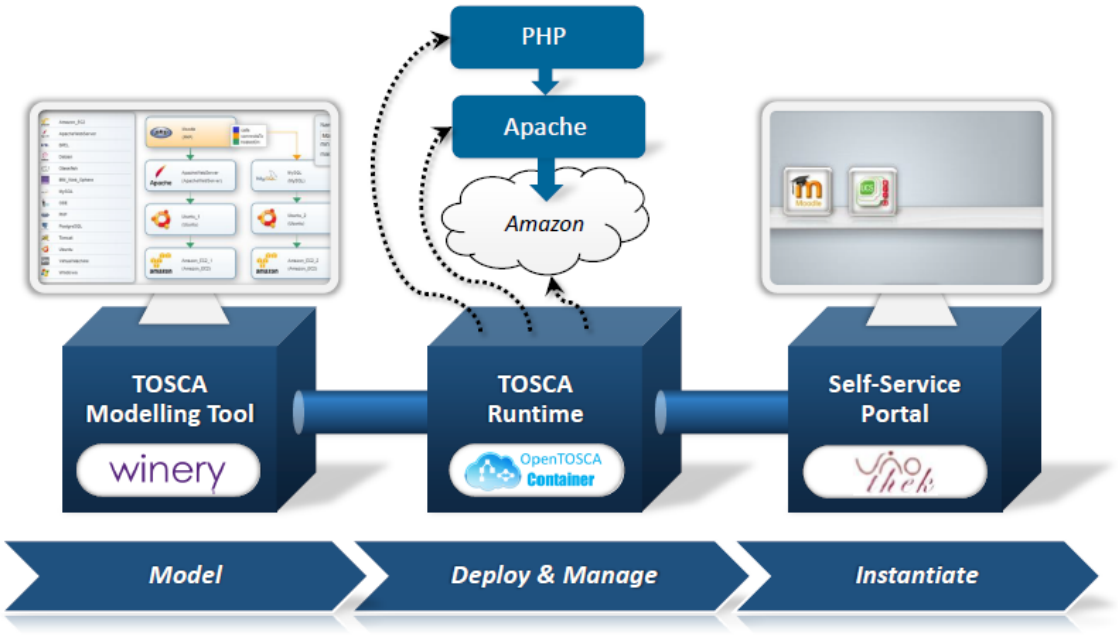

Abbildung 2.4: Aufbau OpenTOSCA Ökosystem Quelle: Breitenbücher et al.[BCK+16]

Eclipse Winery

Winery ist ein von der Universität Stuttgart entwickeltes grafisches Modellierungstool für TOSCA Topologien. Das Projekt wurde an die Eclipse Foundation[5] übergeben, wird aber von der Universität Stuttgart weiterentwickelt. In Winery können komplette Topologien modelliert und zusammen mit den dazugehörigen Artefakten in ein Archiv (CSAR-Datei) gepackt werden. Die generierte CSAR-Datei kann im nächsten Schritt von einer TOSCA Laufzeitumgebung, wie OpenTOSCA, ausgelesen und instanziiert werden[BCK+16].

OpenTOSCA Container

Der OpenTOSCA Container ist die eigentliche TOSCA Laufzeitumgebung des OpenTOSCA Ökosystems. Im OpenTOSCA Container werden die in Winery modellierten CSAR-Dateien provisioniert. Abbildung 2.4 visualisiert beispielsweise das Bereitstellen eines Apache HTTP Servers mit einem PHP Modul auf einem Amazon Web Service Server[BCK+16].

Vinothek

Die Verwaltung der Instanzen läuft über das Self-Service Portal Vinothek. Diese bietet eine Übersicht über die aktuell installierten CSAR-Dateien. Neben einer detaillierten Beschreibung, können die bestehenden Instanzen verwaltet werden. Die in HTML5 und Java Server Pages entwickelte Web-GUI ermöglicht auch das Provisionieren neuer Applikationen. Nach Eingabe der benötigten Parameter, wird über eine REST-API die Provisionierung der Instanz in dem OpenTOSCA Container gestartet[BCK+16].

OpenTOSCA bietet aktuell noch keine Möglichkeit den Status der Instanzen auszulesen bzw. darzustellen. Daher muss der Status jeder Instanz manuell geprüft werden. Ziel dieser Arbeit ist

[5] https://www.eclipse.org

es daher, durch die Integration eines Monitoringsystems in das OpenTOSCA Ökosystem, den Status der Instanz direkt durch OpenTOSCA zu visualisieren.

2.3 Monitoring

Vor der Integration eines Monitoringsystems, muss das zu überwachende System betrachtet und das weitere Vorgehen geplant werden. In dieser Arbeit wird dies als gegeben vorausgesetzt und daher in Kapitel 2.3.1 dargestellt. Das Sammeln von Daten kann grundsätzlich über zwei verschiedene Methoden erfolgen. In Kapitel 2.3.2 und 2.3.3 wird daher der Unterschied zwischen agent-based und agentless Monitoringsystemen beschrieben und die jeweiligen Vor- und Nachteile aufgezeigt. Kapitel 2.3.4 beschreibt abschließend den Unterschied zwischen einem zentralisierten und dezentralisierten Monitoringsystem.

2.3.1 Vorgehen

Focke et al.[FHRS07] beschreiben ein in fünf Phasen gegliedertes Vorgehensmodell, um ein Monitoringsystem in ein bestehendes Softwaresystem zu integrieren. Ziel des Modells ist der ganzheitliche Aufbau einer effektiven und gut wartbaren Monitoring-Lösung. Auch wenn sich das Vorgehensmodell von Focke et al.[FHRS07] primär auf das Überwachen von Softwarekomponenten bezieht, kann das Modell auch auf das Überwachen von Hardwarekomponenten übertragen werden.

Performance-Ziele definieren

Bevor die Daten erfasst werden ist es wichtig, ein klar spezifiziertes Ziel auszugeben. Außerdem muss das Ziel nachvollziehbar sein und zu späteren Zeitpunkten regelmäßig überprüft werden. Monitoring-Ziele können beispielsweise anhand von Unternehmenszielen, Systemanforderungen oder Architekturvorgaben definiert werden. Eine Dokumentation der Ziele ist ebenfalls vorzunehmen[FHRS07].

Performance-Metriken herleiten

Anschließend werden die Performance-Metriken bestimmt. Focke et al.[FHRS07] schlagen dafür die Goal-Question-Metric-Methode (GQM)[RCR94] vor. Anhand von Zielen (Goals) und Fragen (Questions), leitet die GQM bestimmte Metriken (Metric) ab. Ein sorgfältiges Auswählen der Metriken, beugt der Erhebung unnötiger oder falscher Daten vor[FHRS07].

Messpunkte identifizieren

Im nächsten Schritt werden die Messpunkte, die zum Abfragen der Leistungsmetriken nötig sind, identifiziert. Messpunkte sind Lokalisationen in der Softwarearchitektur, an denen Focke et al.[FHRS07] den Programmiercode für die Datenerfassung ergänzen. Ebenso können die Messpunkte aber auch andere Lokalisationen in der Systemarchitektur einschließen. Dies kann beispielsweise der Embedded PC sein, der einen Roboterarm in der Fertigungshalle steuert. Da jeder zusätzliche Messpunkt gleichzeitig eine zusätzliche Belastung für das System darstellt, sollte die Anzahl der Messpunkte ein Kompromiss zwischen Datenqualität und Systemauslastung bilden[FHRS07].

Messcode einsetzen

Nachdem die Messpunkte ermittelt wurden, muss an den jeweiligen Messpunkten der Quellcode für die Datenabfrage implementiert werden. Auf Software-Ebene kann dies über das Erweitern des Quellcodes der Applikation oder über eine Middleware Interception erfolgen. Diese überwachen die Nachrichten zwischen Applikation und Middleware[FHRS07]. Auf Hardwarekomponenten-Ebene kann die Überwachung über einen sogenannten Agenten ablaufen. Dieser wird auf der identifizierten Komponente installiert und sammelt alle definierten Daten.

Monitoring integrieren

Im letzten Schritt erfolgt die eigentliche Integration in das Monitoringsystem. Das Monitoringsystem speichert alle gesammelten Daten zentralisiert. Außerdem können die gespeicherten Daten für eine Weiterverarbeitung durch Dritte zugänglich gemacht werden.

2.3.2 Agent-based

In der Softwaretechnik ist der Begriff Agent sehr unterschiedlich besetzt. Beispielsweise ist der Agent im Bereich der künstlichen Intelligenz ein Assistent, der selbständig handelt und Entscheidungen trifft und so den Nutzer unterstützt[PW01]. In dieser Arbeit wird auf die Definition von Plösch et al.[PW01] verwiesen. Ein Agent ist demnach eine Softwarekomponente, welche den eigenen Systemzustand zu anderen Computern übertragen kann. Agenten sind ein zentraler Bestandteil vieler Monitoringsysteme (Splunk, ELK-Stack, TICK-Stack, etc.) und werden genutzt, um Daten direkt auf den Hostsystemen zu sammeln, eventuell weiterzuverarbeiten und an die zentrale Datenbank zu transferieren. Folgend eine Übersicht über die Vor- und Nachteile eines agent-based Monitoringsystems[Avi].

Vorteil

- Agenten sind direkt auf dem Hostsystem installiert und haben somit direkten Zugriff auf Daten wie Logdateien.

- Agenten können Daten in Echtzeit abfragen.

- Agenten benötigen keine speziellen Firewallregeln um auf das System zugreifen zu können, da sie direkt auf dem System installiert sind.

Nachteil

- Vor Erfassung des Systemzustandes muss der Agent auf dem Hostsystem installiert werden.

- Agenten sind betriebssystemspezifische Programme. Somit muss für jedes Betriebssystem ein separater Agent entwickelt bzw. portiert werden.

- Agenten müssen regelmäßig aktualisiert werden, um eventuelle Sicherheitslücken auszuschließen.

- Agenten benötigen Systemressourcen und können unter Umständen das Hostsystem beeinträchtigen (Monitoring Overhead).

2.3.3 Agentless

Alternativ können Monitoringsysteme auch ohne Agenten Daten sammeln. Dies wird als agentless Monitoring bezeichnet. Ein agentless Monitoringsystem kann mithilfe verschiedener Techniken realisiert werden. Drei Techniken und die jeweiligen Vor- und Nachteile werden folgend vorgestellt[Avi].

Wire Data Monitoring

Das Wire Data Monitoring betrachtet nicht die Endpunkte[6] eines Netzwerkes, sondern wertet einzelne Pakete aus, die über das Netzwerk übertragen werden. Dieses Verfahren ist passiv und benötigt keinen Agenten der auf dem Zielsystem installiert ist. Das Monitoringsystem Splunk Stream[7] verwendet diese Technik der passiven Überwachung des Netzwerkverkehrs.

Ein gewichtiger Vorteil des Wire Data Monitoring ist der Ausschluss eines Overhead, der durch den Einsatz eines Agenten auf dem Hostsystem erzeugt werden könnte. Außerdem wird jede Plattform unterstützt, da keine zusätzliche Softwarekomponente installiert werden muss. Das Wire Data Monitoring kann keine umfassende Abbildung der Aktivitäten des Netzwerkes darstellen. Abhängig von der Netzwerkinfrastruktur und des verwendeten Monitoringsystems, können VLANs (Virtual Local Area Network)[8] und verschlüsselte Nachrichten möglicherweise nicht ausgelesen werden[Avi].

Remote Query Monitoring

Remote Query Monitoring verwendet Protokolle um Querys auf dem Zielsystem auszuführen. Durch den Einsatz von Querys, wird der Status des Zielsystems abgerufen und zurück an das Monitoringsystem gesendet. Hierfür werden beispielsweise RPC (Remote Procedure Call), WMI (Windows Management Instrumentation) oder andere Protokolle verwendet.

Der Vorteil von Remote Query Monitoring ist, neben den bereits erwähnten Vorteilen von Wire Data Monitoring, dass betriebssystemspezifische Daten, wie die Prozessorauslastung, direkt abgefragt werden können. Da Querys erst auf das Zielsystem übertragen und dort ausgeführt werden, ist die Belastung des Zielsystems und des Netzwerks, im Vergleich zum Wire Data Monitoring, erhöht. Zusätzlich müssen unter Umständen Firewall-Regeln erstellt und ein Service Account auf dem Zielsystem eingerichtet werden. Ebenfalls ist eine Echtzeitanalyse nicht möglich, da das Ausführen der Querys Verzögerungen nach sich zieht[Avi].

Hypervisor Monitoring

Die dritte Monitoring-Technologie ist das Hypervisor Monitoring. Dieses greift auf Metriken zurück, die bereits von den Hypervisoren wie Microsoft Hyper-V[9] und VMware vSphere[10] bereitgestellt wurden. Dadurch sind Hypervisor Monitoring-Tools wie Turbonomic[11] oder Solarwinds[12] in der Lage, detaillierte Informationen wie die Prozessorauslastung oder

[6] Als Endpunkt in einem Netzwerk wird beispielsweise ein Computer oder Embedded PC bezeichnet.
[7] https://splunkbase.splunk.com/app/1809
[8] VLANs sind logische Teilnetze innerhalb eines Switches.
[9] https://microsoft.com
[10] https://vmware.com
[11] https://turbonomic.com
[12] https://solarwinds.com

Arbeitsspeicherbelegung über jede individuelle virtuelle Instanz auszulesen. Ein direkter Zugriff auf die virtuelle Instanz ist beim Hypervisor Monitoring dagegen nicht möglich. So kann nicht direkt ausgelesen werden, welcher Nutzer aktuell eingeloggt ist oder welcher Prozess gestartet wurde[Avi].

Alle drei beschriebenen Technologien zum Überwachen von Systemen ohne Agenten haben Vor-und Nachteile. Das gleiche gilt auch für Monitoringsysteme, die auf Agenten zurückgreifen. Der Einsatz eines Monitoringsystems ist daher immer situationsabhängig zu prüfen.

2.3.4 Zentralisiert / Dezentralisiert

Monitoringsysteme können zentral und dezentral aufgebaut sein. Bei einem zentralisierten Monitoringsystem sind alle Komponenten, wie die Datenbank und die Webschnittstelle zur Abfrage der Daten, zentral auf einem System installiert. Ein Ausfall der Netzwerkverbindung an einzelnen Standorten, welche das zentrale Monitoringsystem zum Speichern und Weiterverarbeiten der Daten nutzt, würde zu einem kompletten Ausfall sämtlicher Monitoring-Funktionalitäten führen[Lit].

Alternativ können Monitoringsysteme dezentral aufgebaut sein. Dabei werden die Komponenten jeweils dezentralisiert an den einzelnen Standorten installiert. Dies sorgt für eine Entlastung der Netzwerkinfrastruktur und erhöht gleichzeitig durch die Redundanz der Komponenten die Ausfallsicherheit[Lit]. So kann beispielsweise pro Standort eine Datenbank installiert werden, welche die Daten der jeweiligen Systeme direkt speichert.

3 Anforderungen an das Monitoringsystem

Um verschiedene Monitoringsysteme miteinander vergleichen zu können, müssen zunächst grundsätzliche Anforderungen an ein potentielles System aufgestellt werden. Die Anforderungen richten sich dabei primär an die Integration in das OpenTOSCA Ökosystem. Zusätzlich soll der tatsächliche Nutzen eines Monitoringsystems in einer Produktivumgebung evaluiert werden. Hierbei wird, in Kooperation mit der Robert Bosch GmbH, das laufende Forschungprojekt „Forschungsfabrik: Die wandlungsfähige (Fahrzeug-) Forschungsproduktion (ForschFab)“ aus der Arena 2036 um ein Monitoringsystem erweitert.

Basierend auf der internationalen Norm ISO/IEC 25010[Nor14] wird bezüglich der Anforderungen an ein Monitoringsystem im folgenden Kapitel zwischen funktionalen und nicht-funktionalen Anforderungen unterschieden. Funktionale Anforderungen beschreiben Leistungen, die das Monitoringsystem erbringen muss. Hierzu zählen beispielsweise das Vorhandensein einer grafischen Oberfläche oder andere Funktionen. Nicht-funktionale Anforderungen beschreiben dagegen qualitätsrelevante Eigenschaften. Beispiele hierfür wären die Zuverlässigkeit oder die Skalierbarkeit.

3.1 Funktionale Anforderungen

Um das OpenTOSCA Ökosystem optimal überwachen zu können und dadurch einen größtmöglichen Nutzen zu erbringen, werden im Folgenden drei funktionale Anforderungen an das Monitoringsystem gestellt. Neben der Verarbeitung von numerischen und alphanumerischen Metriken, muss das Monitoringsystem über ein Benachrichtigungssystem verfügen, welches beispielsweise bei der Überschreitung definierter Grenzwerte ein Ereignis ausführt.

3.1.1 Numerische Metriken

Mit OpenTOSCA können verschiedene Anwendungen wie Moodle[1] oder ein Tomcat Server[2] automatisch provisioniert werden. Diese Anwendungen benötigen alle ein Betriebssystem, welches vorher ebenfalls installiert werden muss. Die Abhängigkeit zu dem verwendeten Betriebssystem erfordert die permanente Überwachung der Funktionalität. Aus diesem Grund sollte ein potentielles Monitoringsystem in der Lage sein, numerische Werte, wie die Prozessorauslastung oder den freien Arbeitsspeicher, zu überwachen. Die gesammelten Informationen über das Hostsystem ermöglichen bei Anwendungsproblemen eine erste Analyse, um beispielsweise mögliche Überlastungen erkennen oder vorhersagen zu können.

[1] https://moodle.org
[2] https://tomcat.apache.org

3.1.2 Alphanumerische Metriken

Zusätzlich zu den rein numerischen Werten, sollte das Monitoringsystem ebenfalls alphanumerische Werte verarbeiten können. Das Auswerten von Logdateien ist Teil der funktionalen Anforderungen, da diese notwendig sind, um eine Überwachung von Betriebssystem und Anwendung zu gewährleisten.

Neben den Logdateien des Betriebssystems, generieren Anwendungen oftmals ebenfalls Logdateien. Der Apache HTTP Server schreibt beispielsweise verschiedene Logdateien, die bei Problemen zur Suche nach möglichen Fehlerursachen ausgewertet werden können[Theb].

3.1.3 Benachrichtigungssystem

Die dritte Anforderung fokussiert nicht das Sammeln der Daten, sondern die anschließende Auswertung bzw. Weiterverarbeitung. Gängige Monitoringsysteme wie Splunk oder der ELK-Stack bieten hierbei die Möglichkeit, die Daten anhand von Tabellen oder Grafiken zu visualisieren. Trotzdem werden Probleme oftmals übersehen oder zu spät bemerkt. Das Vorhandensein eines Benachrichtigungssystems wird im Folgenden daher als funktionale Anforderung aufgestellt. Dieses sollte so konfigurierbar sein, dass bei der Überschreitung eines definierten Grenzwertes eine Benachrichtigung, der sogenannte Alert, versendet wird. Dieser Alert erfolgt beispielsweise in Form einer E-Mail oder einer HTTP-Anfrage. So könnte direkt eine E-Mail an den Benutzer versendet werden, wenn der Arbeitsspeicher einen gewissen Grenzwert überschreitet oder ein Roboterarm mit einem Gegenstand kollidiert.

Neben einer visuellen Benachrichtigung sind auch weitere Aktionen, wie das Ausführen eines Bash-Skriptes, denkbar. So könnte das Monitoringsystem genutzt werden, um automatisch auf auftretende Probleme zu reagieren. Diese Aktionen lassen sich unter dem Begriff des Self-Healing zusammenfassen und bieten einen großen Mehrwert. Ein Beispiel dafür wäre der Roboterarm in der Produktionshalle. Fällt dieser aufgrund eins Softwarefehlers aus, erfolgt der Neustart durch ein Bash-Skript. Dieses wird durch das Monitoringsystem gestartet und erfordert somit kein Eingreifen eines Mitarbeiters. Der Themenkomplex Self-Healing würde den Rahmen dieser Arbeit sprengen und wird daher nicht weiter behandelt. Zukünftige Arbeiten können dies als Grundlage nutzen, da ein funktionierendes Monitoringsystem die Basis für ein Self-Healing-System darstellt.

3.2 Nicht-Funktionale Anforderungen

Nicht-funktionale Anforderungen beschreiben qualitätsrelevante Eigenschaften und müssen im Gegensatz zu den funktionalen Anforderungen nicht zwangsläufig erfüllt werden. Die folgende Aufstellung orientiert sich dabei an Aceto et al.[ABDP13] und deren Veröffentlichung zu nicht-funktionalen Anforderungen an ein Cloud Monitoringsystem.

3.2.1 Skalierbarkeit

Ein Monitoringsystem wird als skalierbar bezeichnet, wenn es eine ansteigende Menge an Agenten und somit gesammelten Daten verarbeiten kann. Durch die gesteigerte Verwendung virtueller Instanzen, steigt automatisch die Anzahl der verwendeten Agenten. Dadurch steigt ebenfalls die Anzahl der gesammelten Daten, obwohl nur eine physikalische Maschine verwendet wird. Daher ist es wichtig, dass ein Monitoringssystem, auch bei steigenden Datenmengen, diese weiterhin effizient sammeln, übertragen und analysieren kann. Das zu überwachende System sollte dabei nicht beeinflusst werden[ABDP13]. Eine ansteigende Datenmenge sorgt ebenfalls für eine erhöhte Belastung der Netzwerkinfrastruktur und der Datenbank. Beispielsweise erzeugt eine Boeing 787 pro Flug etwa ein halbes Terabyte an Daten[Weib]. Daher hat sich in den letzten Jahren der Begriff Edge Computing etabliert. Dabei werden große Datenmengen durch Aggregation stark verkleinert. Um die Netzwerkinfrastruktur zu entlasten, wird die Aggregation der Daten direkt auf dem zu überwachenden System vollzogen[SCZ+16].

3.2.2 Elastizität

Ein Monitoringsystem wird als elastisch bezeichnet, wenn es dynamische Änderungen, wie das Generieren oder Löschen virtueller Instanzen, verarbeiten kann. Dies kann mithilfe eines agent-based Monitoringsystems realisiert werden. Die Agenten werden dabei direkt auf dem zu überwachenden System installiert und senden die Daten an die Monitoring Datenbank. Der Nutzer kann im nächsten Schritt die gesammelten Daten auf der Datenbank abfragen[ABDP13].

3.2.3 Aktualität

Ein Monitoringsystem sollte gesammelte Daten zeitnah erfassen und dem Nutzer zur Verfügung stellen, damit dieser rechtzeitig darauf reagieren kann. Nähert sich beispielsweise die Belegung des Arbeitsspeichers einer virtuellen Instanz dem Maximum, sollte das Monitoringsystem automatisch einen Prozess starten, um neue Instanzen zu provisionieren oder der Instanz zusätzlichen Arbeitsspeicher zuzuweisen. Wird die hohe Arbeitsspeicherbelegung nicht schnell genug erkannt, kann dies zu einem Leistungseinbruch oder Komplettabsturz der Insatz führen[ABDP13]. Eine hohe Erfassungsrate sorgt für eine zeitnahe Erfassung der Daten. Gleichzeitig erhöht dies auch die Belastung auf dem zu überwachenden System und dem Monitoringsystem. Daher ist es wichtig, einen Ausgleich zwischen der zusätzlichen Belastung und der benötigten Erfassungsrate der Daten zu finden[ABDP13].

3.2.4 Vollständigkeit, Erweiterbarkeit, Beeinträchtigend

Ein Monitoringsystem wird als vollständig bezeichnet, wenn sowohl physikalische als auch virtuelle Instanzen und verschiedene Arten von Daten gesammelt werden können. Von Erweiterbarkeit spricht man, wenn das Monitoringsystem durch Plugins oder Module erweitert werden kann. Sind diese Erweiterungen ohne große Auswirkung auf das Gesamtsystem implementierbar, ist das Monitoringsystem nicht beeinträchtigend[ABDP13].

3.2.5 Belastbarkeit, Zuverlässigkeit, Verfügbarkeit

Ein Monitoringsystem ist belastbar, wenn es trotz Ausfall einiger Komponenten immer noch zuverlässig Daten sammelt und verarbeitet. Es wird als zuverlässig bezeichnet, wenn es unter diesen Bedingungen für einen gewissen Zeitraum die Funktionalität gewährleisten kann. Verfügbarkeit ist gegeben, wenn der Benutzer jederzeit die angebotene Funktionalität nutzen kann[ABDP13].

3.2.6 Lizenzierung

Da das Monitoringsystem im Rahmen dieser Arbeit ein Bestandteil des OpenTOSCA Ökosystems werden soll, müssen auch die Lizenzen beachtet und auf Kompatibilität geprüft werden. OpenTOSCA wird unter der Apache License 2.0[3] veröffentlicht und ermöglicht so das Verwenden, Modifizieren und Verteilen der Software unter der Bedingung, dass dies ebenfalls unter der Apache License 2.0 veröffentlicht wird[The04]. Ab der dritten Version der GNU General Public License[4] ist diese mit der Apache License 2.0 kompatibel und kann ebenfalls in Betracht gezogen werden[Fre17].

Um einer möglichst großen Gruppe von Nutzern Zugriff zu ermöglichen, muss das Monitoringsystem, zumindest in der Basisvariante, kostenlos verfügbar sein. Die durch die Nutzung einer kostenlosen Variante einhergehenden Einschränkungen, sollten dabei weitgehend andere aufgestellte Anforderungen nicht oder nur minimal beeinträchtigen.

3.2.7 Automatisierbarkeit

Automatisierbarkeit bezeichnet die Möglichkeit, einen Installationsprozess ohne den Eingriff eines Nutzers auszuführen. Somit wäre eine automatisierte Installation und Konfiguration des Agenten gewährleistet.

[3] https://www.apache.org/licenses/LICENSE-2.0.html
[4] https://www.gnu.org/licenses/gpl.html

3.3 Anforderungen der Arena 2036

Abbildung 3.1 zeigt die Architektur eines laufenden Projekt-Testaufbaus in der Arena 2036. Der Aufbau besteht aus drei Komponenten. Dem mobilen Roboterarm Bosch APAS[5], der aufgrund seiner Sensorhaut Menschen erkennt und daher für den Betrieb ohne Schutzzaun zertifiziert ist[Rob]. Die zweite Komponente ist das Bosch Werkerassistenzsystem ActiveAssist. Hierbei handelt es sich um ein modular erweiterbares Montageassistenzsystem, welches die Arbeitsschritte mithilfe von Sensoren, wie Kameras und Ultraschallsensoren, überwacht und über Beamer, Touchscreens und weitere Schaltflächen mit dem Benutzer interagieren kann[Bosb]. Die dritte Komponente ist der Bosch WLAN-Akkuschrauber Nexo, der automatisch das Drehmoment und den Drehwinkel erfasst und an ein übergeordnetes System überträgt[Bosa]. Der Bohrer wird sowohl im Active Assist, als auch als eigenständiges Gerät verwendet.

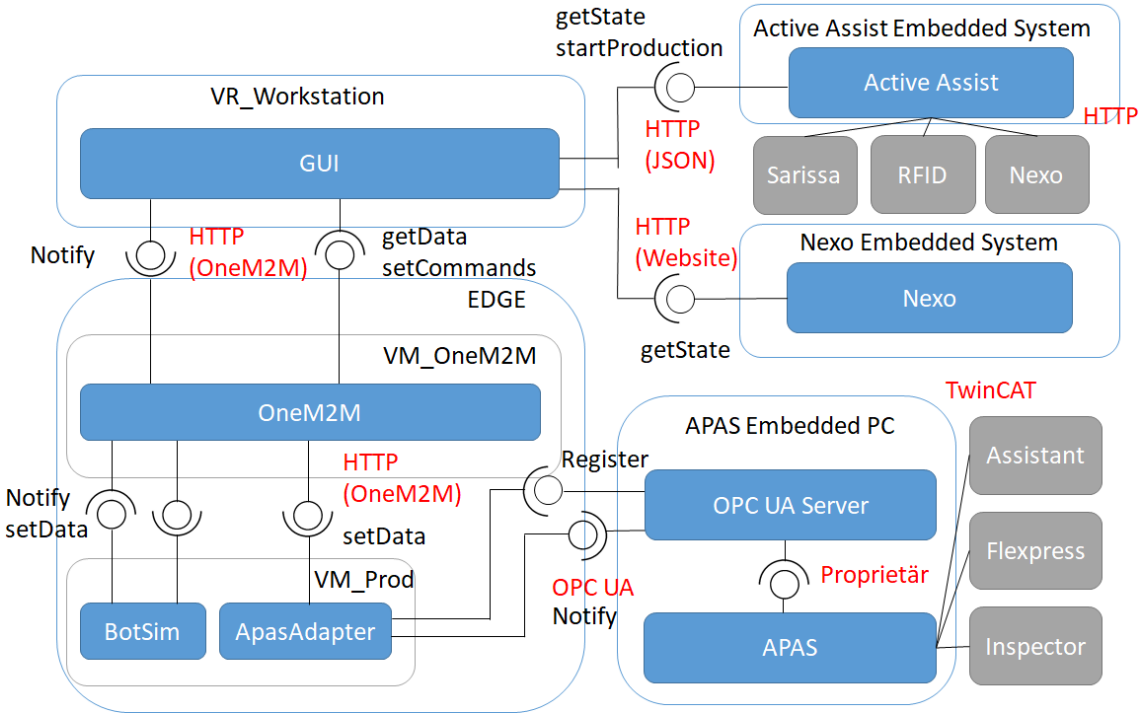

Abbildung 3.1: Architektur Arena 2036 ohne Monitoring

Für die Konfiguration und Visualisierung werden zwei Server (VR_Workstation und EDGE) verwendet. Der Server VR_Workstation bietet eine Web-GUI zur Darstellung des aktuellen Zustands der Komponenten. Der Status der Komponenten Active Assist und Nexo, wird jeweils über ein separates embedded system gesammelt und über eine HTTP-Schnittstelle zur Verfügung gestellt. Das APAS embedded system bietet keine direkte HTTP-Schnittstelle. Um die Daten trotzdem visualisieren zu können, wurden auf dem Server EDGE zwei virtuelle Maschinen (VM_OneM2M und VM_Prod) eingerichtet. Auf der virtuellen Maschine VM_Prod registriert sich der ApasAd-

[5]https://www.bosch-apas.com

apter auf dem OPC UA Server[6] der auf dem APAS Embedded PC installiert ist und erhält so den aktuellen Systemstatus. Diese Daten werden an den OneM2M Service[7] auf der virtuellen Maschine VM_OneM2M gesendet, welcher wiederum eine HTTP-Schnittstelle anbietet. Diese Schnittstelle nutzt die Web-GUI, um den aktuellen Status des APAS zu erhalten.

Anhand der oben dargestellten Rahmenbedingungen, ergeben sich für das zukünftig eingesetzte Monitoring drei Zielvorstellungen: Zustandsanzeige der Komponenten, Überwachung der Maschinen und Benachrichtigung bei Ausfall.

3.3.1 Zustandsanzeige der Komponenten

Der Projekt-Testaufbau besteht aus unterschiedlichen Komponenten, die in der Arena 2036 verteilt sind. Derzeit gibt es keine Möglichkeit den Status aller Systeme und Maschinen zentral abzufragen. Daher soll das Monitoringsystem genutzt werden, um zentralisiert den Status aller Geräte zu sammeln und direkt zu visualisieren. Dabei geht es nicht um eine detaillierte Analyse, sondern lediglich um die Erreichbarkeit und Auslastung der Systeme. Die Erreichbarkeit könnte beispielsweise anhand des Ping-Protokolls überprüft werden. Die Auslastung der Systeme dagegen, könnte über das Abfragen der Prozessorauslastung und des verfügbaren Arbeitsspeichers geprüft werden.

3.3.2 Überwachung der Maschinen

Neben der Prüfung der Erreichbarkeit und Auslastung der Systeme, sollten bestimmte Komponenten detailliert analysiert werden können. Ein Beispiel dafür wäre der Service BotSim, welcher für die Steuerung des APAS Roboters verantwortlich ist. Dieser schreibt den aktuellen Zustand des Roboters alle 50 Millisekunden in die Syslog Datei. Wie in folgendem Auszug der Logdatei erkennbar ist, werden Daten über den Status und der Position des Roboterarms als JSON-String gespeichert. Für eine Auswertung müsste das Monitoringsystem also nach bestimmten Begriffen, wie botsim, DEBUG und State bzw. Pose, filtern und die Werte, wie joblist, x oder y, im nächsten Schritt auslesen und speichern können.

```
Jan 12 13:33:30 cr-ubuntu python3.5[30063]: 2018-01-12 13:33:30,533 - botsim - DEBUG
    - State: {"current_job": "WP12", "state": "Auto", "joblist": ["WP0", "WP1",
    "WP2"]}
Jan 12 13:33:30 cr-ubuntu python3.5[30063]: 2018-01-12 13:33:30,588 - botsim - DEBUG
    - Pose: {"theta": 0, "x": 5.994543918483426, "y": 2.4973198614233305}
Jan 12 13:33:30 cr-ubuntu python3.5[30063]: 2018-01-12 13:33:30,645 - botsim - DEBUG
    - State: {"current_job": "WP12", "state": "Auto", "joblist": ["WP0", "WP1",
    "WP2"]}
Jan 12 13:33:30 cr-ubuntu python3.5[30063]: 2018-01-12 13:33:30,688 - botsim - DEBUG
    - Pose: {"theta": 0, "x": 5.976655489125701, "y": 2.488375360592448}
```

Listing 3.1: Auszug Logdatei Arena 2036

[6] OPC Unified Architecture, kurz OPC UA, ist ein Hersteller- und Plattform-unabhängiger Standard für die machine-to-machine Kommunikation[OPC14]

[7] OneM2M ist ein Kommunikationsstandard für die machine-to-machine Kommunikation[Deu17]

Alternativ können bestimmte Daten auch über die REST-Schnittstelle von der virtuellen Maschine OneM2M abgefragt werden. Daher ist es zwingend notwendig, dass das Monitoringsystem ebenfalls die Daten über die REST-Schnittstelle abfragen und weiterverarbeiten kann. Die gesammelten Daten sollen über ein Dashboard visualisiert werden. Dadurch kann der Benutzer direkt ablesen, an welcher Position sich der Roboterarm derzeit befindet und welche Jobs noch ausgeführt werden sollen.

3.3.3 Benachrichtigung bei Ausfall

Die Darstellung der aktuellen Position des Roboterarmes oder der aktuellen Prozessorauslastung des Hostsystems, ist über ein Dashboard möglich. Bestimmte Ereignisse, wie der Absturz des für die Steuerung des APAS verantwortlichen Services, besitzen eine erhöhte Dringlichkeit und erfordern ein zeitnahes Eingreifen. Daher wird ein Benachrichtigungssystem benötigt, welches bei Auftreten eines bestimmten Logeintrags oder der Überschreitung bestimmter Grenzwerte direkt reagieren kann. Das Benachrichtigungssystem sollte beispielsweise direkt E-Mails versenden können, sodass der verantwortliche Mitarbeiter zeitnah über einen möglichen Ausfall informiert wird. Zusätzlich sollte das Benachrichtigungssystem Bash-Skripte oder HTTP-Anfragen ausführen können, um automatisiert auf bestimmte Probleme reagieren zu können.

4 Evaluation Monitoringsysteme

Im folgenden Kapitel werden die vier Monitoringsysteme anhand der aufgestellten Anforderungen evaluiert. Während die funktionalen Anforderungen zwangsläufig erfüllt sein müssen, sind die nicht-funktionalen Anforderungen optional, da diese nur qualitätsrelevante Eigenschaften wie Zuverlässigkeit oder Skalierbarkeit beschreiben. Zur besseren Verständlichkeit wird im Folgenden zunächst das jeweilige Monitoringsystem vorgestellt und die einzelnen Komponenten betrachtet, die zur Erfüllung der funktionalen Anforderungen erforderlich sind. Anschließend werden die nicht-funktionalen Anforderungen behandelt. Sollten die funktionalen Anforderungen nicht erfüllt werden, wird auf die Evaluation weiterer Anforderungen verzichtet. Da sich die Anforderungen der Arena 2036, siehe Kapitel 3.3, mit den funktionalen Anforderungen überschneiden, werden diese nicht gesondert evaluiert.

4.1 Splunk

Die Splunk Inc. wurde im Jahr 2003 gegründet und ist laut dem Portal db-engines.com[1] nach Elasticsearch die beliebteste Search Engine[Sola]. Das Monitoringsystem besteht, wie in Abbildung 4.1 ersichtlich, aus drei Komponenten. Dem Forwarder, der die Daten sammelt, dem Indexer, der sie parst und indiziert und dem Search Head, der als Schnittstelle zwischen Nutzer und dem Indexer fungiert[Splr]. Die Komponente Forwarder wird direkt auf dem zu überwachenden System installiert und sammelt Daten, welche anschließend an die Komponente Indexer weitergeleitet werden. Der Indexer parst und indiziert die empfangenen Daten. Dies erfolgt in zwei Schritten. Zunächst werden die Rohdaten ausgelesen und den jeweiligen Feldern zugeordnet (parsen). Zusätzlich wird ein Zeitstempel ausgelesen. Kann kein Zeitstempel in den Rohdaten gefunden werden, wird der Indexierungszeitpunkt verwendet. Im zweiten Schritt werden die Rohdaten und die generierten Indexdateien auf der Festplatte gespeichert (indizieren)[Spl18a]. Die Komponete Search Head ermöglicht dem Nutzer anschließend den Zugriff auf die gesammelten Daten. Die Daten werden dabei über die von Splunk entwickelte Abfragesprache Search Processing Language (SPL) abgerufen[Splc].

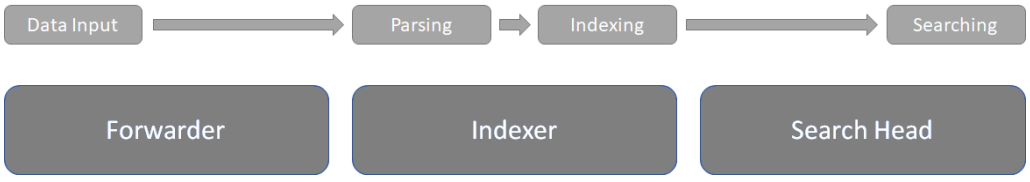

Abbildung 4.1: Splunk Komponenten angelehnt an edureka.co[Var]

[1] https://db-engines.com

Funktionale Anforderungen

4.1.1 Numerische Metriken

Das Monitoringsystem Splunk verwendet zum Sammeln der Daten Agenten, die direkt auf dem Zielsystem installiert werden. Diese Agenten werden bei Splunk Forwarder genannt und sind in verschiedenen Ausführungen verfügbar. In Kapitel 4.1.4 werden die drei Forwarder (Universal Forwarder, Heavy Forwarder und Light Forwarder) detailliert beschrieben. Die Forwarder werden über eine Datei (inputs.conf) konfiguriert. Die Überwachung numerischer Metriken, wie die Prozessorauslastung oder die Belegung des Arbeitsspeichers, erfolgt über vorkonfigurierte Forwarder[Splq]. Folgende Daten können beispielsweise über das Windows Hostsystem direkt gesammelt werden[Splq].

General computer Hostname, Active Directory Domain

Operating system Installiertes Betriebssystem und verfügbarer Arbeitsspeicher

Processor Anzahl der verfügbaren Prozessoren und deren Auslastung

Disk Übersicht über die Festplatten und den verfügbaren Speicher

Network Adapter Übersicht über installierte Netzwerkkarten

Service Status und Informationen über installierte Services

Process Status und Informationen über laufende Prozesse

Dies ist nur ein Ausschnitt über die Daten, die direkt über den Forwarder ausgelesen werden. Weitere Daten können über dritte Programme wie NetIQ[2] ausgewertet und über dem Splunk Forwarder ausgelesen werden[Mic]. Ein direktes Auslesen von Systemdaten wie bei Windows Hostsytemen wird für das Betriebssystem Linux ebenfalls angeboten. So kann beispielsweise folgender Befehl in der inputs.conf[3] die Prozessorauslastung eines Linux-basierten Hostsystems auslesen.

```
[script://./bin/cpu.sh]
sourcetype = cpu
source = cpu
...
```

Listing 4.1: Splunk Konfiguration: Prozessorauslastung (Auszug)

[2] https://www.netiq.com
[3] http://docs.splunk.com/Documentation/Splunk/latest/Admin/Inputsconf

4.1.2 Alphanumerische Metriken

Alphanumerische Werte wie Logdateien können ebenfalls über die Komponente Forwarder gesammelt werden. Dabei muss die Konfigurationsdatei (input.conf) auf dem jeweiligen Forwarder angepasst werden. Neben dem Pfadname der Logdateien, können noch weitere Attribute wie der Suchindex[4] oder reguläre Ausdrücke festgelegt werden. Durch die Nutzung von Platzhaltern, können komplette Verzeichnisse oder Dateien mit bestimmter Endung verarbeitet werden[Splp]. Die folgende Konfiguration liest beispielsweise alle Daten im Verzeichnis C:\Users\AppData\inin_tracing mit dem Datentyp .ininlog aus.

```
[monitor://C:\Users\AppData\inin_tracing\*.ininlog]
disabled = 0
index = main
sourcetype = log4j
```

Listing 4.2: Splunk Konfiguration: Logdatei

Sourcetype beschreibt dabei das Format der gesammelten Daten, beispielsweise log4j, apache error oder cisco syslog[Splz] und kann bei der Suche über den Search Head zum Gruppieren oder Filtern verwendet werden. Main ist der Standard-Index für alle gesammelten Daten[Splj]. Zur besseren Verwaltung können neue Indizes erstellt werden. Beispielsweise kann bei einem parallelen Betrieb einer Testumgebung und einer Produktivumgebung das Monitoring auf zwei Indizes verteilt werden. Die gesammelten Daten der Testumgebung werden unter dem Index Test und die Daten der Produktivumgebung unter dem Index Prod gespeichert.

4.1.3 Benachrichtigungssystem

Splunk hat mit der zugehörigen Search Processing Language (SPL) eine eigene Abfragesprache[Splc]. Mithilfe der SPL können Anfragen gestellt werden, die zur Suche, aber auch zum Generieren von Tabellen, Grafiken und Alerts, verwendet werden können. Das Auslösen der Alerts kann mithilfe sogenannter Crontabs zu festgelegten Zeiten geprüft werden. Crontabs bestehen aus fünf Zahlen[5], die den Abfrageintervall definieren. Beispielsweise wiederholt der Crontab folgende Abfrage alle 5 Minuten (*/5 * * * *), während folgender Crontab die Abfrage an jedem ersten Montag im Monat um 9 Uhr morgens ausführt (0 9 1-7 * 1)[Sply]. Alternativ ist eine Echtzeitüberwachung über die sogenannte real-time Suche möglich. Diese sorgt aber gleichzeitig für eine erhöhte Belastung des Search Heads[Splg]. Wird ein Alert ausgelöst, kann eine E-Mail versendet, eine HTTP-Anfrage gestellt oder ein Bash-Skript gestartet werden[Splt].

Splunk erfüllt alle drei funktionalen Anforderungen. Somit ist Splunk grundsätzlich als Monitoringsystem für OpenTOSCA geeignet. Im nächsten Schritt werden die nicht-funktionalen Anforderungen evaluiert.

[4] Der Index kann bei der Suche über den Search Head zum Gruppieren verwendet werden.
[5] 1. Zahl: Minute, 2. Zahl: Stunde, 3. Zahl: Tag, 4. Zahl: Monat, 5. Zahl: Wochentag

Nicht-Funktionale Anforderungen

4.1.4 Skalierbarkeit

Ein skalierbares Monitoringsystem kann mit einer großen und steigenden Menge von Agenten und Daten umgehen. Splunk nutzt zum Sammeln der Daten Forwarder, die auf das Zielsystem installiert werden und direkt die Daten sammeln bzw. verarbeiten und anschließend weiterleiten. Zur Abdeckung möglichst vieler Szenarien, werden drei verschiedene Forwarder angeboten[Spli].

Universal Forwarder
: Der Universal Forwarder beinhaltet nur die nötigsten Komponenten und belastet das Hostsystem nur minimal. Die Rohdaten werden direkt an den Indexer weitergeleitet und anschließende geparst und indiziert. Da alle gesammelten Daten ungefiltert an den Indexer weitergeleitet werden, führt dies zu einer starken Netzwerkauslastung[Splx].

Heavy Forwarder
: Der Heavy Forwarder ermöglicht, ähnlich wie der Indexer, das Parsen und Indizieren von Daten. Diese erste Aggregation auf dem Hostsystem entlastet durch Verringerung der Datenmenge nicht nur die Netzwerkinfrastruktur, sondern auch den Indexer, da dort der Parsing-Prozess übersprungen werden kann. Außerdem können direkt auf dem Heavy Forwarder Alerts definiert werden. Dies ermöglicht eine zeitnahe Reaktion, da die Daten direkt auf dem Hostsystem ausgewertet werden und nicht erst an den Indexer weitergeleitet werden müssen. Durch den zusätzlichen Parsing-Prozess auf dem Forwarder, wird die Systembelastung auf dem Hostsystem erhöht[Spli].

Light Forwarder
: Der Light Forwarder wurde ursprünglich als eingeschränkte Version des Heavy Forwarders entwickelt, um den Overhead auf dem Zielsystem zu verringern. Seit Splunk Enterprise Version 6.0 wurde dieser zugunsten des Universal Forwarders eingestellt[Spln].

Die Anzahl der Forwarder in einem System ist durch das Monitoringsystem Splunk nicht begrenzt. Somit kann das System unbegrenzt skaliert werden. Allerdings setzt dies die gleichzeitige Erhöhung der Indexer voraus, da diese sonst mit dem Auswerten der Daten überlastet wären. Durch die unterschiedlichen Forwarder ist die Skalierbarkeit ebenfalls gewährleistet, da eine Nutzung von Heavy Forwardern die Indexer entlasten. Außerdem wird durch Aggregation die Datenmenge verringert und somit die Netzwerkinfrastruktur entlastet. Auf Systemen mit wenig verfügbaren Systemressourcen, kann der Universal Forwarder installiert werden. Dieser belastet das Hostsystem nur minimal und verlagert das rechenintensive Parsing auf die Indexer.

4.1.5 Elastizität

Um neue virtuelle Maschinen zu instanziieren, muss der Forwarder installiert, die Adresse des Indexer eingetragen und die zu sammelnden Daten festgelegt werden. Eine Änderung am Indexer oder Search Head muss nicht vorgenommen werden. Das Entfernen von Instanzen erfolgt ebenfalls ohne Einfluss auf das Gesamtsystem[Splf]. Alternativ können die Forwarder auch über einen Deploymentserver konfiguriert werden. Dabei wird ein zusätzlicher Deploymentserver benötigt, um die Konfiguration der Forwarder zu verwalten. Dies verringert den Konfigurationsaufwand,

da nach der Installation des Forwarders nur die Adresse des Deploymentservers übergeben werden muss. Die weitere Konfiguration und mögliche nachträgliche Anpassungen werden zentral über den Deploymentserver vorgenommen[Splk]. Somit ist Splunk sowohl mit als auch ohne Deploymentserver ein elastisches Monitoringsystem, da durch den Einsatz von Forwardern nur eine Lose Kopplung zwischen Datenbank und dem zu überwachenden System besteht.

4.1.6 Aktualität

Die Splunk Forwarder versenden die gesammelten Daten standardmäßig über das verbindungsorientiert TCP-Protokoll, welches aufgrund der Architektur den Fokus auf sichere Übertragung und weniger auf geringe Latenz setzt[spi]. Zusätzlich kann die Komponente Indexer die erhaltenen Daten bestätigen und so die Vollständigkeit, bei gleichzeitiger Erhöhung der Verzögerung, der Daten garantieren[Spls]. Das Monitoringsystem Splunk bietet sowohl eine Echtzeitsuche, als auch ein Benachrichtigungssystem an. Bei der Echtzeitsuche werden keine Angaben zu einer maximalen Verzögerungszeit gegeben[Spla][6]. Das Benachrichtigungssystem erfüllt ebenfalls die aufgestellten Anforderungen und wurde bereits in Kapitel 4.1.3 beschrieben.

4.1.7 Vollständigkeit, Erweiterbarkeit, Beeinträchtigend

Wie in den vorherigen Unterkapiteln schon beschrieben, unterstützen die Splunk Forwarder eine große Anzahl an Plattformen und Datentypen. Für virtuelle Maschinen die über die Virtualisierungssoftware VMware instanziiert wurden, gibt es beispielsweise zusätzlich das Splunk Add-on for VMware[7]. Das Add-on erweitert Splunk um vorgefertigte Reports und Dashboards und vereinfacht das Sammeln und Visualisieren von vCenter Logdateien, ESXi Logdateien und Leistungsdaten der einzelnen virtuellen Maschinen[Splu] Neben dem Sammeln und Indizieren der Daten, gehört das Auswerten und Visualisieren der Daten zu den Hauptaufgaben eines Monitoringsystems. Werden beispielsweise die Daten des Monitoringsystems vom Benutzer falsch interpretiert, kann dies zu neuen Problemen führen. Um dies zu vermeiden, bietet die Splunk Inc. eine Vielzahl von Apps und Add-ons an, die kostenlos auf der Splunkbase Seite[8] heruntergeladen werden können.

Die angebotenen Apps stellen eine Sammlung von vorbereiteten Dashboards, Berichten und Benachrichtigungen dar, die relevante Daten für bestimmte Szenarien übersichtlich visualisieren und bei bekannten Problemen direkt Benachrichtigungen versenden können. So gibt es Apps für den Cloud-Computing-Anbieter Amazon Web Services (AWS) zur Darstellung der aktuellen und voraussichtlichen Kosten, Nutzungszeiten der jeweiligen Instanzen und eine Übersicht über die Netztopologie[9]. Mit Add-Ons können neue Datenquellen hinzugefügt werden[Sple]. Ein Beispiel dafür wäre das Add-On Splunk DB Connect[10]. Mithilfe dieser Erweiterung können gängige Datenbanken wie DB2, MySQL, MSSQL, PostgreSQL und viele weitere ausgelesen und in Splunk

[6] Wie sich in Testläufen herausstellte, bewegte sich der durchschnittliche Zeitraum zwischen Erfassung der Daten und der Verfügbarkeit im Searchead im Sekundenbereich.
[7] https://splunkbase.splunk.com/app/3215/
[8] https://splunkbase.splunk.com
[9] https://splunkbase.splunk.com/app/1274/
[10] https://splunkbase.splunk.com/app/2686/

importiert werden[Splv]. Da die Apps den Funktionsumfang und Nutzen für den Anwender signifikant steigern, werden folgend zwei Apps detailliert betrachtet.

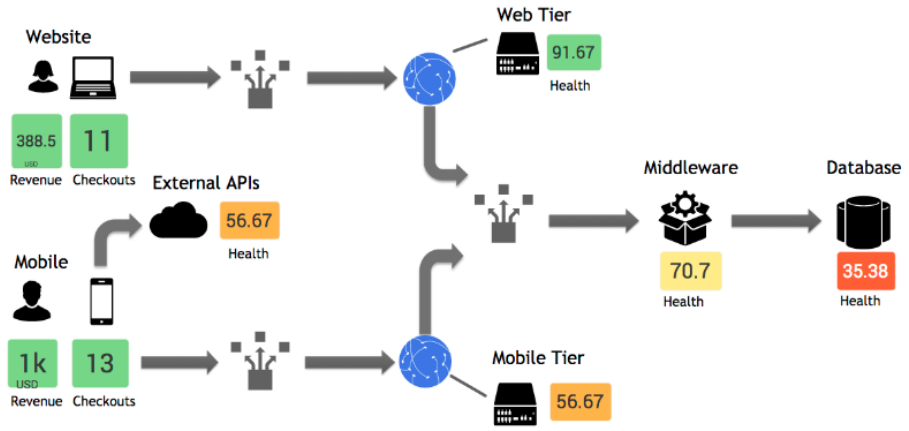

Abbildung 4.2: Splunk IT Service Intelligence KPI Dashboard Quelle: splunk.com[Spl18b]

Die Splunk IT Service Intelligence App (ITSI)[11] ermöglicht das Überwachen der Infrastruktur und der Performance von geschäftskritischen Services. Durch die Aggregation der Daten und dem automatischen Korrelieren von Events, wird die Komplexität verringert und sorgt so für eine übersichtliche Darstellung der gesammelten Informationen[Spl18b]. Abbildung 4.2 visualisiert eine Beispielinfrastruktur, die durch dynamisch generierte Daten angereichert wurde und den Gesundheitszustand der einzelnen Komponenten darstellt. Dies unterstützt den Benutzer dahingehend, dass direkt ersichtlich ist, welche Komponente möglicherweise zu stark ausgelastet ist und genauer überprüft werden sollte. Nicht nur die Infrastruktur kann visualisiert werden. Mithilfe von definierten Key Performance Indicator, kurz KPI, kann die Performance von Services dargestellt werden. KPIs beschreiben Leistungskennzahlen, wie Kosten, Qualität und Zeit, die genutzt werden, um einen Service bewerten zu können[CC04]. Abbildung 4.3 zeigt die Darstellung der Services durch die Splunk IT Service Intelligence App.

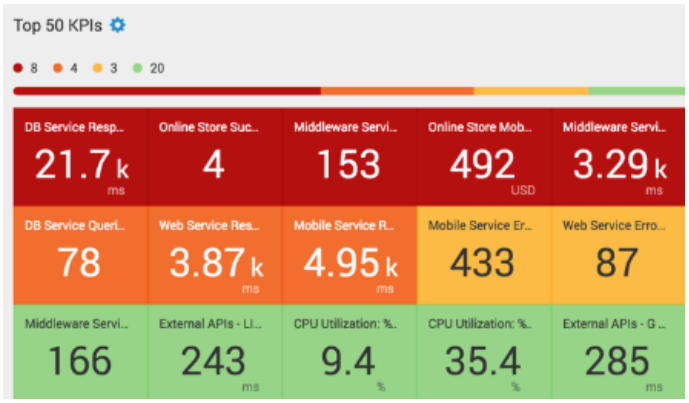

Abbildung 4.3: Splunk IT Service Intelligence Infrastruktur Dashboard Quelle: splunk.com[Spl18b]

[11] https://splunkbase.splunk.com/app/1841

Während die Splunk ITSI App hauptsächlich zur Überwachung von Business Services und der Infrastruktur genutzt wird, kann das Splunk Machine Learning Toolkit[12] zur verbesserten Auswertung jeglicher Art von Daten genutzt werden. So ist die Überwachung vieler Services relativ problematisch, da natürliche Schwankungen nicht berücksichtigt werden. Betrachtet man beispielsweise die Bestellung von unternehmensinternen Services, wie die Bestellung von WLAN Zugängen für externe Mitarbeiter oder Kunden, zeigt sich zwangsläufig, dass am Wochenende weniger bestellt wird als unter der Woche. Ein starrer Grenzwert wird daher voraussichtlich am

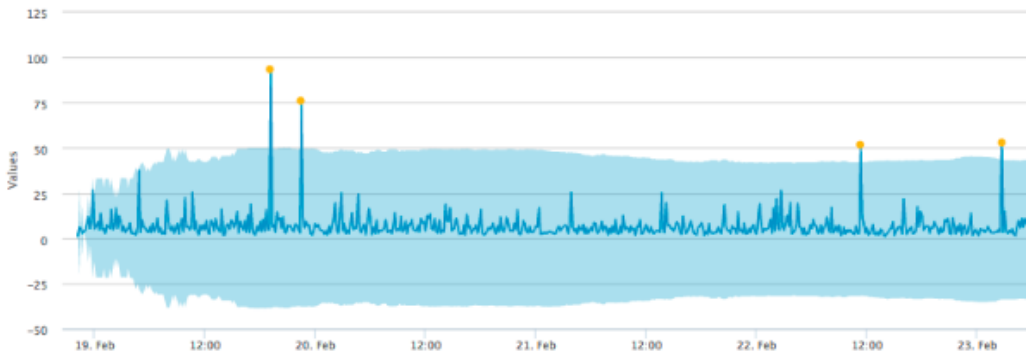

Abbildung 4.4: Splunk Machine Learning Toolkit Quelle: splunk.com[Spl18c]

Wochenende überschritten, da die Bestellungen den Durchschnitt stark unterschreiten. Ähnlich verhält es sich voraussichtlich am Wochenanfang, da hier mit sehr vielen Anträgen zu rechnen ist. Daher ist es notwendig starre Grenzwerte durch flexible zu ersetzen, um Fehlalarme durch starke Ausreißer wie in Abbildung 4.4 zu vermeiden. Abbildung 4.5 verdeutlicht nochmals wie mithilfe von Machine Learning flexible Ober- und Untergrenzen ermittelt und bei der Auswertung behilflich sein können. Dabei werden sowohl kurzfristige Schwankungen, als auch langfristige Trends berücksichtigt.

Die große Anzahl an unterstützten Plattformen und Datenquellen und die Möglichkeit das Monitoringsystem durch Add-ons zu erweitern, sorgen für eine hohe Vollständigkeit und Erweiterbarkeit. Die Trennung zwischen Forwarder, Indexer und Search Head ermöglicht das Erweitern des Search Heads mit Add-ons, ohne die beiden anderen Komponenten (Forwarder und Indexer) zu beeinträchtigen.

4.1.8 Belastbarkeit, Zuverlässigkeit, Verfügbarkeit

Splunk bietet zur Erhöhung der Belastbarkeit, Zuverlässigkeit und Verfügbarkeit die Möglichkeit des Clusterns an. Dabei wird zwischen Search Head Clustering und Indexer Clustering unterschieden.

Index-Cluster

Für einen Index-Cluster werden drei Komponenten benötigt. Einen Master Node, der den Cluster verwaltet. Dieser koordiniert die Replikationen der Peer Nodes und teilt dem

[12] https://splunkbase.splunk.com/app/2890/

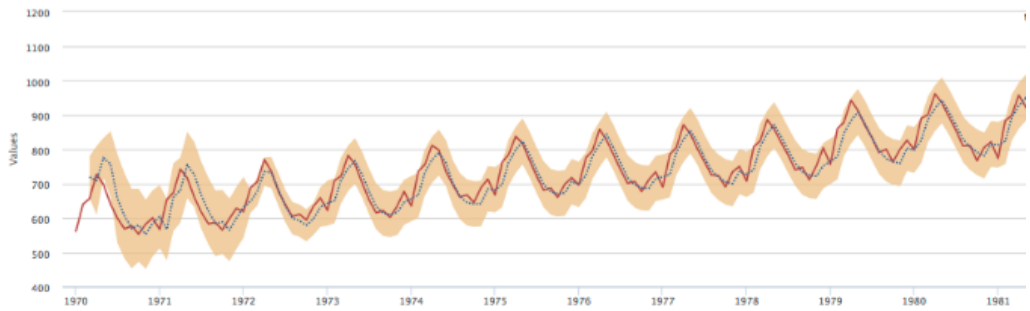

Abbildung 4.5: Splunk Machine Learning Toolkit Quelle: splunk.com[Spl18c]

Search Head mit, wo die Daten gespeichert sind[Splw]. Die zweite Komponente sind die Peer Nodes. Diese empfangen die Daten der Forwarder und indizieren diese. Peer Nodes können gleichzeitig neue Daten indizieren und Replikate empfangen oder senden. Über den Replikationsfaktor wird die Anzahl der Kopien bestimmt. Bei dem Replikationsfaktor drei werden mindestens drei Peer Nodes benötigt, da alle Daten dreifach gespeichert werden. Eine höhere Anzahl an Peer Nodes, bei gleichem Replikationsfaktor, erhöht die Leistung bzw. entlastet die Peer Nodes[Splw]. Die dritte Komponente ist der Search Head. Jeder Index-Cluster benötigt mindestens ein Search Head. Werden weitere Search Heads dem Cluster hinzugefügt, wird dies Search Head Cluster genannt[Splw].

Search Head Cluster

Der Search Head ist die zentrale Komponente für den Benutzer und dient als Schnittstelle zwischen Benutzer und Indexer. Fällt der Server mit dem Search Head aus, ist es für den Benutzer nicht möglich auf die gesammelten Daten zuzugreifen, selbst wenn die Indexer noch erreichbar sind. Um diesen single point of failure zu vermeiden, können weitere Search Heads hinzugefügt werden. In Kombination mit einem Load Balancer erhöhen diese nicht nur die Belastbarkeit, Zuverlässigkeit und Verfügbarkeit, sondern können auch Lastspitzen kompensieren, da sich die Anfragen auf mehrere Search Heads verteilen[Splb].

Um eine hohe Belastbarkeit, Zuverlässigkeit, Verfügbarkeit des Monitoringsystems zu garantieren, ist sowohl das Clustern der Indexer, als auch des Search Heads notwendig. Ein weiteres Szenario kann ein Netzwerkausfall sein, bei dem der Forwarder den Indexer nicht mehr erreichen kann. Zur Vermeidung von Datenverlust erkennen Splunk Forwarder automatisch Netzausfälle und leiten die gesammelten Daten an einen anderen Indexer weiter. Alternativ können die Daten auf dem Forwarder zwischengespeichert werden bis der Indexer wieder erreichbar ist. Zusätzlich kann der Indexer den Empfang der Daten quittieren, um den Datenverlust zu vermeiden[Spls].

4.1.9 Lizenzierung

Für die Nutzung von Splunk gibt es drei verschiedene Lizenzen[Splh].

Splunk Free

Die kostenlose Lizenz von Splunk erlaubt ein maximales Indizierungsvolumen von 500 Megabyte pro Tag. Es fehlen außerdem Features wie die verteilte Suche, das Versenden von Benachrichtigungen und das Machine Learning.

Splunk Enterprise
Splunk Enterprise hat keinerlei Einschränkung und bietet den kompletten Funktionsumfang von Verteilte Suche, Benachrichtigungen und einer zentralen Managementkonsole.

Splunk Cloud
Splunk Cloud ermöglicht das Überwachen von Cloud-Services von Drittanbietern wie AWS[13], Akamai[14] oder ServiceNow[15]. So kann mit Splunk Cloud die Betriebszeit und Verfügbarkeit der Cloud-Services überwacht und die Sicherheit und Compliance[16] sichergestellt werden[Spld].

Die Lizenzkosten orientieren sich immer an der Datenmenge die pro Tag indiziert werden. Das Unternehmen Splunk inc. gewährt bei großen Datenmengen in der Regel einen hohen Rabatt, eine transparente Auflistung der Kosten ist allerdings nicht öffentlich einsehbar[Splaa]. Die kostenlose Lizenz ist stark limitiert, so ist beispielsweise das Versenden von Benachrichtigungen in dieser Variante nicht enthalten. Außerdem ist die Kompatibilität mit der Apache License 2.0 oder GPL Version 3 nicht gegeben.

4.1.10 Automatisierbarkeit

Ein Monitoringsystem welches die Automatisierbarkeit vollständig erfüllt, würde den Agenten automatisch auf jeder neuen Instanz bereitstellen. Anschließend würde der Agent automatisch die Konfiguration generieren oder diese von einem Deploymentserver abrufen. Splunk bietet hierbei verschiedene Möglichkeiten, um einen Agenten (Forwarder) auf dem zu überwachenden System zu installieren.

Installer
Der Splunk Forwarder kann über den Windows Installer installiert und konfiguriert werden[Spll]. Ein Automatisierung ist in diesem Szenario nicht oder nur schwer möglich.

Kommandozeile
Als zweite Möglichkeit bietet Splunk die Installation über die Kommandozeile an. Die Konfiguration kann als Parameter übergeben werden. Bei der Verwendung eines Deploymentservers, müsste nur die Adresse des Deploymentservers übergeben werden. Die Konfiguration erfolgt anschließend über den Deploymentserver[Splk]. Wird zusätzlich noch der Parameter AGREETOLICENSE=yes /quiet hinzugefügt, werden automatisch die Lizenzbedingungen akzeptiert und die Installation läuft komplett im Hintergrund ab[Splm]. Somit wäre eine automatisierte Installation über ein Skript möglich. Zukünftige Änderungen an der Konfiguration können zentral über den Deploymentserver verwaltet werden.

Host Image
Die dritte Möglichkeit der automatisierten Installation wäre das einmalige Erstellen und Konfigurieren eines Images mit installierten Splunk Forwarder. Mit dem Befehl ./splunk clone-prep-clear-config können alle instanz-spezifischen Konfigurationen, wie Hostname

[13] https://aws.amazon.com
[14] https://www.akamai.com
[15] https://www.servicenow.com
[16] Compliance beschreibt die Einhaltung Unternehmensinterner und vertraglich festgelegter Regeln[Ges]

und die Globally Unique Identifier (GUID)[17], gelöscht werden. Das Image kann anschließend gespeichert und wiederverwendet werden. Der Forwarder auf den duplizierten Instanzen initialisiert sich automatisch und verwendet den neuen Hostname und die entsprechende GUID[Splo]. Eine Automatisierung wäre in diesem Fall ebenfalls möglich. Dabei müsste im Voraus für jedes Betriebssystem ein Image erstellt und der Forwarder darauf installiert werden.

Eine vollständige Automatisierung ist nativ mit Splunk nicht möglich. Die Installation per Kommandozeile und anschließender Verteilung der Konfiguration über den zentralen Deploymentserver, minimiert den Installationsaufwand und kann mithilfe von weiteren Tools komplett automatisiert werden.

4.2 ELK-Stack

Elasticsearch wurde im Jahr 2004 unter dem Namen Compass von Shay Banon entwickelt und ist laut dem Portal db-engines.com[18] derzeit die beliebteste Search Engine[Ban10][Sola]. Abbildung 4.6 visualisiert den Aufbau der Hauptkomponenten des ELK-Stacks[19] und gleichzeitig den Ablauf des Monitoringsystems.

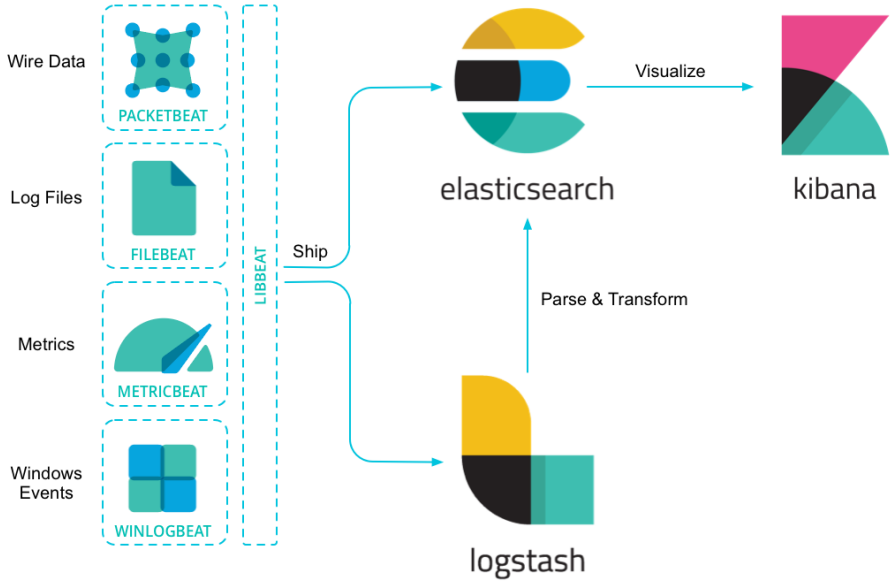

Abbildung 4.6: Aufbau ELK-Stack Quelle: elastic.co[Elab]

[17] Eindeutiger Schlüssel zur Identifizierung von Systemen unter Windows[Tec]
[18] https://db-engines.com
[19] Der ELK-Stack wurde nach den Anfangsbuchstaben der drei Hauptkomponenten (Elasticsearch, Logstash und Kibana) benannt.

Beats bezeichnen Agenten, die direkt auf dem zu überwachenden System installiert werden und gesammelte Daten über die Libbeat API entweder direkt an die Datenbank oder zur weiteren Verarbeitung an Logstash weiterleiten. Für unterschiedliche Einsatzszenarien stehen verschiedene Varianten des Agenten zur Verfügung. Der sogenannte Packetbeat analysiert den Datenverkehr, während ein Filebeat Logdateien sammelt[Elac]. Die Komponente Logstash ist eine data collection engine mit dem Ziel, die von den Agenten (Beats) gesammelten Daten zu parsen und zu transformieren. Im Anschluss werden die Daten an die Datenbank Elasticsearch weitergeleitet[Elao]. Elasticsearch ist eine Suchengine, die auf der Apache Lucene[20] Bibliothek basiert[Roh]. Die von den Agenten und Logstash übertragenen Daten, werden in einem NoSQL-Format gespeichert und per REST-Schnittstelle bereitgestellt[Elae]. Die Komponente Kibana nutzt die von Elasticsearch bereitgestellte REST-Schnittstelle, um die gesammelten Daten abzufragen und zu visualisieren. Das Webinterface von Kibana dient als Schnittstelle zwischen dem Benutzer und der Elasticsearch Datenbank[Elan]. Eine weitere Komponente ist der im X-Pack enthaltene Watcher. Dieser stellt das Benachrichtigungssystem des ELK-Stacks dar und kann verwendet werden, um bei Überschreitung festgelegter Grenzwerte in Form einer E-Mail oder HTTP-Anfrage reagieren zu können[Elaae]. In der Erweiterung X-Pack sind außerdem Komponenten für die reports oder das Machine Learning enthalten[Elaac].

Funktionale Anforderungen

4.2.1 Numerische Metriken

Zum Sammeln der Daten nutzt der ELK-Stack den Agenten Beat. Für Metriken, wie die Prozessorauslastung durch einzelne Services, kann der Agent Metricbeat verwendet werden. Metricbeat bietet Module, um Metriken von unterschiedlichen Services auszulesen[Elau]. Das Modul System liest beispielsweise Metriken wie die Prozessorauslastung, verwendeter Arbeitsspeicher, Netzwerkauslastung und weitere aus. Ausgelesen werden die Betriebssysteme FreeBSD, Linux, macOS und Windows[Elaw]. Neben Informationen wie der Prozessorauslastung können auch installierte Services ausgelesen werden. In der Metricbeat Dokumentation[21] gibt es eine Übersicht über die verfügbaren Module. Darunter sind Services wie Apache, Docker, MySQL, PostgreSQL oder vSphere[Elav].

4.2.2 Alphanumerische Metriken

Neben der oben vorgestellten Version des Agenten Beat, gibt es eine weitere Version die zum Auslesen von Logfiles verwendet werden kann. Konfiguriert wird Filebeat über die filebeat.yml[Elal]. Folgende Konfiguration liest alle Dateien mit der Dateierweiterung .log im Verzeichnis /var/log aus und sendet den Inhalt direkt an Elasticsearch.

```
filebeat.prospectors:
- type: log
  enabled: true
  paths:
```

[20] https://lucene.apache.org
[21] https://www.elastic.co/guide/en/beats/metricbeat/current/metricbeat-modules.html

```
   - /var/log/*.log

output.elasticsearch:
hosts: ["192.168.1.42:9200"]
```

Listing 4.3: ELK-Stack Konfiguration: Filebeat

Alternativ können die Daten zur Weiterverarbeitung an Logstash gesendet werden[Elal]. Logstash parst die Logeinträge mit dem Grok Filter Plugin und weist so den Logeinträgen Felder zu. Folgende Logstash Konfiguration empfängt die Daten von Beats und wendet das Grok Pattern COMBINEDAPACHELOG[22] an. Dieses trennt die Logeinträge auf und weist den einzelnen Einträgen feste Felder zu. Anschließend werden die gefilterten Einträge an Elasticsearch weitergeleitet[Elas].

```
input {
   beats {
      port => "5044"
   }
}
filter {
   grok {
      match => { "message" => "%{COMBINEDAPACHELOG}"}
   }
}
output {
   stdout { ... }
}
```

Listing 4.4: ELK-Stack Konfiguration: Logstash (Auszug)

Wie das Beispiel verdeutlicht, können auch alphanumerische Werte wie Logeinträge vom ELK-Stack verarbeitet werden.

4.2.3 Benachrichtigungssystem

Benachrichtigungen oder im englischen alerts können über den im X-Pack enthaltenen Watcher versendet werden. Dafür muss im Watcher ein definierter Input und ein fester Zeitintervall festgelegt werden. Folgende Konfiguration überprüft beispielsweise alle 10 Sekunden den Index logs ob Einträge mit dem Stichwort Error hinzugefügt wurden[Elaae]. Wird ein Logeintrag gefunden, kann der Watcher eine definierte Aktion ausführen. Dies kann entweder das Versenden einer E-Mail oder auch das Ausführen einer HTTP-Anfrage sein[Elaad].

```
PUT \_xpack/watcher/watch/log\_error\_watch
{
  "trigger" : {
    "schedule" : { "interval" : "10s" }
  },
```

[22] Siehe Grok Standardpattern GitHub Repository: https://github.com/elastic/logstash/blob/v1.4.2/patterns/grok-patterns

```
 "input" : {
  "search" : {
   "request" : {
    "indices" : [ "logs" ],
    "body" : {
     "query" : {
      "match" : { "message": "error" }
     }
    }
   }
  }
 }
}
```

Listing 4.5: ELK-Stack Konfiguration: Benachrichtigungssystem

Das Monitoringsystem ELK-Stack erfüllt alle drei funktionalen Anforderungen. Daher ist der ELK-Stack grundsätzlich für die Integration in das OpenTOSCA Ökosystem geeignet und kann weiter evaluiert werden.

Nicht-Funktionale Anforderungen

4.2.4 Skalierbarkeit

In Kapitel 4.2 wurden die beiden Beat Varianten Filebeat und Metricbeat vorgestellt, welche als Agenten auf den zu überwachenden Systemen agieren. Diese werden verwendet um Logfiles, bzw. Metriken wie die Prozessorauslastung oder den Netzwerkverkehr zu überwachen. Neben den beiden Beat Varianten gibt es noch vier weitere Agenten.

Packetbeat
: Packetbeat ist ein Netzwerk-Analyse-Tool, welches den Netzwerkverkehr zwischen den Servern überwacht und die Anfragen und Antworten korreliert. Es werden verschiedene Protokolle wie ICMP, DNS oder HTTP unterstützt[Elaz].

Winlogbeat
: Mit Winlogbeat können die event logs von windows-basierte Systemen direkt ausgelesen werden. Darunter fallen Events wie application events, hardware events, security events und system events. Dies ermöglicht beispielsweise das Auswerten sicherheitsrelevanter Vorgänge, wie der wiederholten Eingabe falscher Anmeldedaten[Elaab].

Auditbeat
: Auditbeat überwacht die Aktivitäten der Benutzer und Prozesse auf Linux Systemen. Events aus dem Linux Audit Framework können direkt abgerufen werden. Ebenso können Änderungen an kritischen Dateien überwacht werden[Elaa].

Heartbeat
: Heartbeat kann genutzt werden um periodisch zu prüfen, ob ein Service erreichbar ist. Die Prüfung kann per ICMP, TCP oder HTTP erfolgen[Elam].

Die von den Agenten gesammelten Daten können entweder direkt an Elasticsearch übermittelt und dort gespeichert, oder zur Weiterverarbeitung an Logstash gesendet werden. Eine Aggregation der Daten in Logstash ist über das Aggregate filter Plugin möglich. So können auch Logdateien aggregiert werden, indem mehrfach auftretende Nachrichten nur einmalig, dafür mit der Anzahl der Nachrichten, an Elasticsearch weitergeleitet werden[Elar].

Mit den verschiedenen Agenten und Logstash, ist der ELK-Stack nicht nur in der Lage verschiedene Arten von Daten auf unterschiedlichen Plattformen (FreeBSD, Linux, macOS und Windows[Elaw]) zu sammeln, sondern kann durch Aggregation die Menge der Daten verkleinern und so die Netzwerkinfrastruktur entlasten. Genauso ist das System theoretisch unendlich skalierbar, da die Anzahl der Agenten nicht begrenzt ist.

4.2.5 Elastizität

Der ELK-Stack ist ein agent-based Monitoringsystem. Die in Kapitel 4.2.4 vorgestellten Agenten werden direkt auf dem zu überwachenden System installiert und senden die Daten an die Datenbank Elasticsearch oder zur Weiterverarbeitung an Logstash[Elad]. Wird eine neue Maschine instanziiert, benötigt diese nur den Agenten Beat. Eine Änderung in Elasticsearch ist nicht notwendig. Zur Erhöhung der Übersichtlichkeit, können zusätzliche Indizes angelegt werden. Dies ist optional, da Elasticsearch sonst die gesammelten Daten in dem Standardindex main speichert[Splj]. Durch den Einsatz von Agenten ist der ELK-Stack elastisch, da das Hinzufügen und Entfernen neuer zu überwachender Systeme keinen Einfluss auf das Monitoringsystem hat.

4.2.6 Aktualität

Das Unternehmen Elastic bezeichnet die Suchgeschwindigkeit von Elasticsearch es als Near Realtime und definiert dabei ein Zeitfenster, zwischen Indizierung und Sichtbarkeit der Daten, von etwa einer Sekunde[Elaf]. Wie in Kapitel 4.2 schon beschrieben, verfügt der ELK-Stack über ein integriertes Benachrichtigungssystem. Dies kann durch Ausführung eines Skriptes, beispielsweise einer virtuellen Instanz bei Überlast automatisiert mehr Arbeitsspeicher zuweisen oder weitere Instanzen bereitstellen[Elaad]. Die Agenten besitzen zudem einen internen Puffer. Dieser kann genutzt werden, um die gesammelten Daten zwischenzuspeichern. So können beispielsweise mehrere Events gesammelt und anschließend gebündelt an Logstash oder Elasticsearch gesendet werden. Dies sorgt für eine Entlastung der Netzwerkinfrastruktur, da einzelne Anfragen den Overhead erhöhen[Elay]. Aufgrund des beschriebenen Zeitfensters von etwa einer Sekunde zwischen Indizierung und Sichtbarkeit der gesammelten Daten, erfüllt das Monitoringsystem die Anforderung der Aktualität.

4.2.7 Vollständigkeit, Erweiterbarkeit, Beeinträchtigend

In Kapitel 4.2 wurde der Agent Metricbeat beschrieben. Neben den bereits vorgestellten Modulen zur Überwachung der physikalischen Instanz, gibt es ein Modul zum Auslesen der virtuellen Maschinen. Das Modul befindet sich derzeit noch im Beta-Status, kann aber bereits Systemdaten wie die Prozessorauslastung oder den verfügbaren Arbeitsspeicher auslesen[Elax]. Wie das beschriebene Modul zum Auslesen der virtuellen Maschinen gezeigt hat, ist der ELK-Stack modular

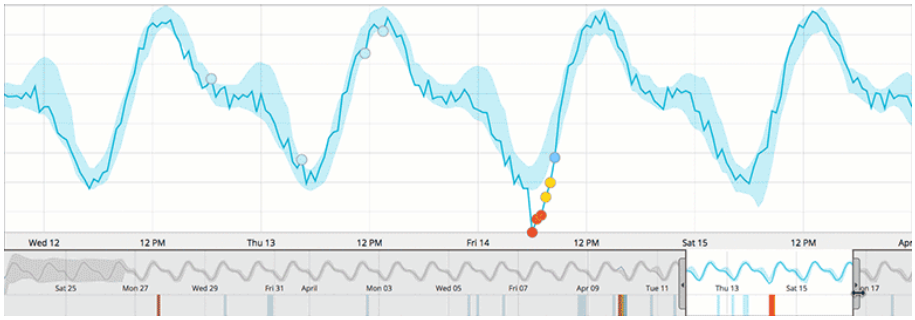

Abbildung 4.7: Elastic Machine Learning Dashboard Quelle: elastic.co[Elat]

erweiterbar. So können die Agenten mit neuen Modulen erweitert werden, ohne die anderen Komponenten wie Elasticsearch oder Kibana zu beeinflussen[Elav]. Ein weiteres Beispiel ist das in X-Pack enthaltene maschine learning[23]. Wie in Abbildung 4.7 ersichtlich, kann das Machine Learning genutzt werden, um regelmäßige Schwankungen zu kompensieren und tatsächliche Ausreißer aufzuspüren.

4.2.8 Belastbarkeit, Zuverlässigkeit, Verfügbarkeit

Die Belastbarkeit, Zuverlässigkeit und Verfügbarkeit kann durch das sogenannte Clustern der einzelnen Komponenten erhöht werden. Ab der Gold Lizenz kann Clustering im ELK-Stack verwendet werden[Elaaa]. Logstash kann dafür horizontal skaliert und in Gruppen zusammengefasst werden[Elaq]. Die Agenten können die Daten verteilt an alle Nodes senden[Elak]. Elasticsearch kann ebenfalls horizontal skaliert werden. Ein Cluster besteht aus einem oder mehreren Instanzen auf denen Elasticsearch installiert ist. Jedes Cluster bestimmt automatisch einen Master Node, welcher Änderungen, wie das Erstellen und Löschen von Indizes, verwaltet[Elai]. Ein Index wird benötigt um Daten auf Elasticsearch speichern zu können. Indizes sind logische Namensräume, die auf einen oder mehrere physische shard zeigen. Die sogennanten shards werden verwendet um die Daten verteilt speichern zu können. Daher können ein oder mehrere Shards auf verschiedenen Node liegen[Elag]. Elasticsearch unterscheidet zwischen Primary Shards und Replica Shards. Jedes Dokument im Index gehört einem Primary Shards. Zur Erhöhung der Verfügbarkeit, wird das Dokument auf den Replica Shards ebenfalls gespeichert. Dies verhindert den Datenverlust bei Ausfall eines Nodes[Elag]. Abbildung 4.8 zeigt ein Cluster bestehend aus zwei Nodes. Node 1 ist der Master Node und beinhaltet drei Primary Shards (P0, P1, P2). Node 2 beinhaltet drei Replica Shards (R0, R1, R2). Fällt in diesem Beispiel ein kompletter Node aus, würden keine Daten verloren gehen. Werden neue Dokumente hinzugefügt, werden diese zuerst im primary shard gespeichert und parallel auf die Replica Shards kopiert[Elah].

4.2.9 Lizenzierung

Die drei Hauptkomponenten des ELK-Stacks (Elasticsearch, Logstash und Kibana) wurden von Elasticsearch unter der Apache Lizense 2.0 veröffentlicht[Elaj]. Der Quelltext ist über das Elastic

[23] https://www.elastic.co/de/products/x-pack/machine-learning

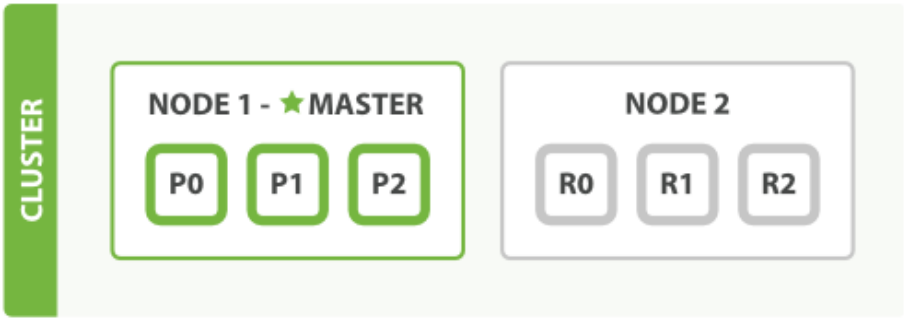

Abbildung 4.8: ELK-Stack: Zwei Node Cluster Quelle: elastic.co[Elah]

GitHub Repository[24] einsehbar. Somit ist die Kompatibilität mit der OpenTOSCA Lizenz gegeben. Auch wenn die drei Hauptkomponenten des ELK-Stacks unter der Apache Lizense 2.0 veröffentlicht wurden, gibt es neben einem kostenpflichtigen Support weitere Komponenten, die nur in einer kostenpflichtigen Lizenz enthalten sind[Elaaa]. Beispielsweise ist das X-Pack erst ab der Platinum Lizenz komplett enthalten[Elaaa]. Das Bilden von Clustern, um die Verfügbarkeit und Skalierbarkeit zu erhöhen, ist erst ab der Enterprise Lizenz möglich[Elaaa]. Da das Benachrichtigungssystem eine funktionale Anforderung ist und dieses nicht in der kostenlosen Version enthalten ist, zählt diese Anforderung als nicht erfüllt.

4.2.10 Automatisierbarkeit

Der ELK-Stack nutzt verschiedene Varianten von Agenten. Installiert werden alle ähnlich, daher wird die Installation nur anhand von Filebeat beschrieben. Filebeat kann über die Elastic Homepage für Windows[25] und verschiedene Linux Distributionen[26] heruntergeladen und anschließend installiert werden. Anschließend muss die Konfiguration angepasst werden[Elal]. Die Konfiguration erfolgt ausschließlich über die Konfigurationsdatei. Ein zentrale Konfiguration über einen Deploymentserver ähnlich wie bei dem Monitoringsystem Splunk ist nicht möglich.

[24] https://github.com/elastic
[25] https://www.elastic.co/downloads/beats/filebeat
[26] Beispiel Debian: https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-6.2.3-amd64.deb0

4.3 TICK-Stack

Das Monitoringsystem TICK-Stack wurde 2013 von dem amerikanischen Unternehmen InfluxData entwickelt. TICK-Stack wurde nach den vier Hauptkomponenten (Telegraf, InfluxDB, Chronograf und Kapacitor) benannt. Abbildung 4.9 zeigt, in welcher Beziehung die einzelnen Komponenten zueinander stehen[Infj].

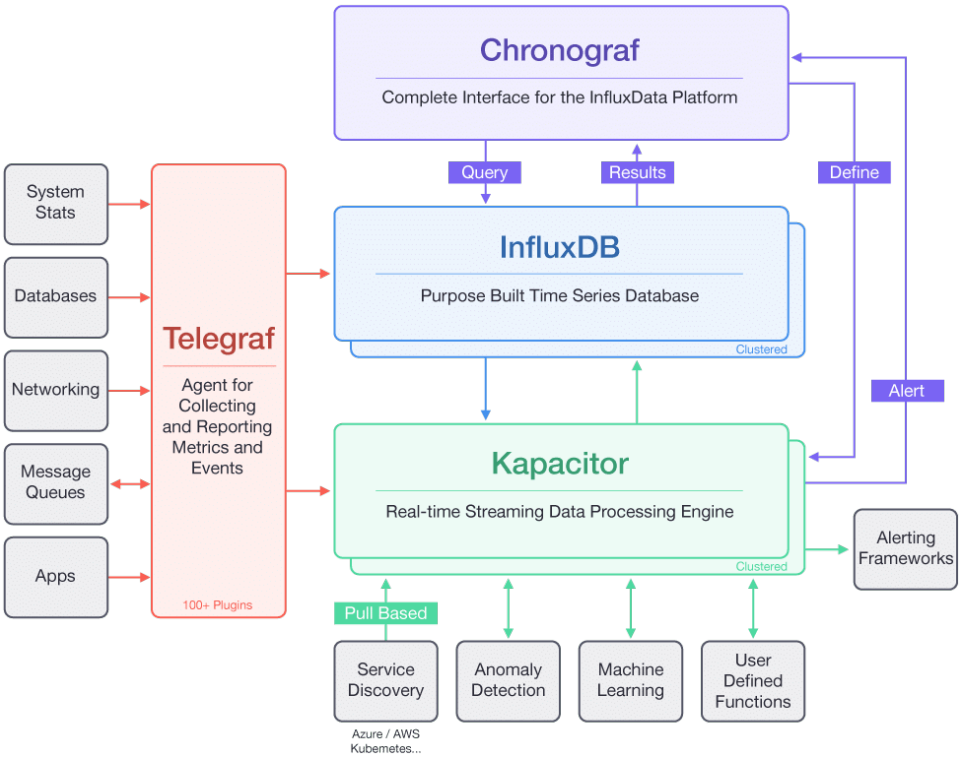

Abbildung 4.9: Architektur TICK-Stack Quelle: influxdata.com[Infj]

Kernstück des TICK-Stacks ist die Time Series Datenbank InfluxDB, welche laut dem Portal db-engines[27] die beliebteste Time Series Datenbank ist[Solb]. Entwickelt wurde InfluxDB in der Programmiersprache Go[28], mit dem Ziel möglichst große Mengen an Daten mit einer möglichst hohen Schreibrate und minimaler Verzögerung zu verarbeiten. Daher wurde InfluxDB speziell für die Bereiche DevOps Monitoring, Applikationsüberwachung und Echtzeitüberwachung entwickelt[Infp]. Gesammelt werden die Daten über den Plugin-basierten Agenten Telegraf, der die Daten an InfluxDB weiterleitet. Die Komponente Chronograf dient als Schnittstelle zwischen dem Benutzer und der Datenbank. Über die Chronograf Web-GUI kann der Benutzer Abfragen an die Datenbank senden. Die Ergebnisse werden grafisch oder tabellarisch dargestellt. Die vierte Komponente ist Kapacitor. Dieser fungiert als Benachrichtigungssystem und kann bei Überschreitung definierter Grenzwerte den Benutzer über diverse Kommunikationsdienste benachrichtigen oder direkt Bash-Skripte ausführen. Konfiguriert werden die sogenannten alerts über die Web-GUI des Chronografs.

[27] https://db-engines.com
[28] https://golang.org/

Funktionale Anforderungen

4.3.1 Numerische Metriken

Das Sammeln der Daten erfolgt bei TICK-Stack über den Plugin-basierten Agenten Telegraf. In dem Telegraf GitHub Repository[29] der InfluxData Inc. gibt es eine große Anzahl an Input Plugins, die numerische Metriken, wie die Prozessorauslastung, Arbeitsspeichernutzung oder Festplattenkapazität, auslesen. Zur besseren Übersicht folgt eine Aufzählung der Input Plugins, die zur Erfassung des Zustands des Hostsystems nützlich sind[Infl].

inputs.cpu Auslastung der einzelnen CPUs.

inputs.mem Details über die Belegung des Arbeitsspeichers.

inputs.netstat Übersicht über aktive TCP-Verbindungen und UDP-Sockets.

inputs.tomcat Erzeugte Systemauslastung durch den installierten Tomcat Service.

inputs.apache Erzeugte Systemauslastung durch den installierten Apache Service.

inputs.docker Erzeugte Systemauslastung durch die Docker Container.

In dem GitHub Repository gibt es noch weitere Plugins[30]. Mit den vorgestellten Input Plugins ist das Monitoringsystem in der Lage, den Zustand eines Systems, bezogen auf die numerischen Metriken, zu überwachen.

4.3.2 Alphanumerische Metriken

Alphanumerische Werte wie Logeinträge können ebenfalls über den Agenten Telegraf ausgelesen und weiterverarbeitet werden. Dafür kann das Input Plugin Logparser[31] verwendet werden. Dieses liest die Logdatei aus und ordnet den Logeinträgen die entsprechenden Felder in der Datenbank zu. Die Zuordnung erfolgt über den Grok Parser, der auch modifiziert in der ELK-Stack Komponente Logstash zum Einsatz kommt[Elap]. Grok pattern sind folgendermaßen aufgebaut:

```
{<capture_syntax>[:<semantic_name>][:<modifier>]}
```

Listing 4.6: TICK-Stack Aufbau Grok Pattern

Capture_syntax beschreibt den Inhalt der Logdatei und wie dieser weiterverarbeitet werden soll. Telegraf hat dafür diverse pattern integriert. In folgenden Beispielen werden reguläre Ausdrücke verwendet, um die Pattern zu beschreiben[Infq].

USERNAME `[a-zA-Z0-9._-]+`

WORD `\b\w+\b`

MONTHDAY `(?:(?:0[1-9])|(?:[12][0-9])|(?:3[01])|[1-9])`

[29] https://github.com/influxdata/telegraf/tree/master/plugins
[30] Stand 01.05.2018: 111 Input Plugins
[31] https://github.com/influxdata/telegraf/tree/master/plugins/inputs/logparser

Semantic_name weist dabei dem durch das Pattern festgelegten Bereich ein Feld in der Datenbank zu. Der Tag Modifier kann den Datentyp anpassen, wobei standardmäßig alle Felder als String interpretiert werden[Infi]. Um den Aufbau des Input Plugins nochmals zu verdeutlichen, folgt ein Auszug aus der Telegraf Konfigurationsdatei der Arena 2036 zum Auslesen der System-Log Einträge.

```
[[inputs.logparser]]
 files = ["/var/log/syslog"]
  [inputs.logparser.grok]
  ...
  patterns = ['\%{SYSLOGTIMESTAMP:timestamp} (?:\%{SYSLOGFACILITY}
      )?\%{SYSLOGHOST:logsource} \%{SYSLOGPROG}:\%{GREEDYDATA:log}']
```

Listing 4.7: TICK-Stack Telegraf Konfiguration: Input Plugin Logparser (Auszug)

Files gibt den Speicherort der Logdatei an. Wurde die Datei nie eingelesen, wird immer die komplette Logdatei eingelesen. Patterns bestimmt den Aufbau der Logdatei und welche Bereiche welchem Feld zugeordnet werden. Das pattern SYSLOGTIMESTAMP löst beispielsweise die Kombination aus Monat, Tag und Uhrzeit auf, welches in Linux Logdateien häufig verwendet wird[32][Infq]. Somit ist der TICK-Stack in der Lage, auch komplette Logeinträge auszulesen und weiterzubearbeiten.

4.3.3 Benachrichtigungssystem

Das Versenden von Alerts über ein Benachrichtigungssystem ist ebenfalls möglich. Dafür kann die real-time streaming data processessing engine Kapacitor verwendet werden. Folgendes Beispiel beschreibt eine Konfiguration des Kapacitor.

```
  stream
        ...
      |alert()
        .id('kapacitor/{{ index .Tags "service" }}')
        .message('{{ .ID }} is {{ .Level }} value:{{ index .Fields "value" }}')
        .info(lambda: "value" > 10)
        .warn(lambda: "value" > 20)
        .crit(lambda: "value" > 30)
        .post("http://example.com/api/alert")
        .tcp("exampleendpoint.com:5678")
        .email('oncall@example.com')
```

Listing 4.8: TICK-Stack Konfiguration: Benachrichtigungssystem (Auszug)

Die Zeilen .id und .message bestimmen den Inhalt der ID bzw. der Message. Inhalte die in geschweifter Klammer geschrieben werden, wie .Level , sind Felder aus der Datenbank und werden dynamisch generiert. Steigt ein Wert (value) über 10 Prozent, wird ein alert der Stufe info ausgelöst. Bei einem Anstieg über 20 Prozent wird der alert der Stufe warning und bei 30 Prozent

[32] Beispiel: Jan 12 08:27:52

der Stufe critical ausgelöst. Beim Auslösen einer dieser alerts, wird in diesem Beispiel eine HTTP POST-Anfrage, eine TCP-Anfrage und eine E-Mail versendet[Infa]. Wie das Beispiel gezeigt hat, bietet der TICK-Stack mit dem Kapacitor ein vollwertiges Benachrichtigungssystem.

Der TICK-Stack erfüllt alle drei funktionalen Anforderungen. Daher ist der TICK-Stack grundsätzlich für dieses Szenario als Monitoringsystem geeignet und kann weiter evaluiert werden.

Nicht-Funktionale Anforderungen

4.3.4 Skalierbarkeit

Das Monitoringsystem TICK-Stack verwendet, wie auch Splunk und der ELK-Stack, zum Sammeln der Daten Agenten. Die Komponente Telegraf ist ein Plugin-basierter Agent um Daten zu sammeln. Die gesammelten Daten können anschließend aggregiert und an die Datenbank InfluxDB weitergeleitet werden. Über die Konfigurationsdatei[33] können die Plugins beliebig hinzugefügt und konfiguriert werden[Info]. Außerdem ermöglicht Telegraf das Aufteilen der Konfiguration in mehrere Dateien[Infc]. Dies kann genutzt werden, um Plugins zu trennen und auf mehrere Konfigurationsdateien zu verteilen. So kann in der Hauptkonfiguration die Verbindung zur Datenbank über das Output Plugin definiert werden und über das Input Plugin der Systemzustand (Prozessorauslastung und Arbeitsspeicherbelegung) überwacht werden. Es gibt vier Arten von Plugins[Infr]. Eine detaillierte Übersicht über die aktuellen Plugins gibt es in dem Telegraf GitHub Repository[34] der InfluxData Inc. Im Folgenden werden vier Arten der Plugins (Input, Output, Aggregator und Processor) vorgestellt.

Input

Input Plugins ermöglichen das Sammeln verschiedener Daten von unterschiedlichen Quellen. So werden die gängigen Datenbanken wie MySQL, MSSQL oder PostgreSQL unterstützt. Genauso können Services wie Tomcat oder Apache überwacht werden. Außerdem kann durch das Auslesen von Logdateien fast jeder Service, wenn diese Log-Einträge schreiben, zumindest teilweise überwacht werden.

Output

Die durch die Input Plugins gesammelten Daten müssen nach eventueller Aggregation an eine Datenbank gesendet werden. Obwohl Telegraf eine Teilkomponente des TICK-Stacks ist, werden neben der InfluxDB auch andere Zielsysteme unterstützt. So können die gesammelten Daten per Advanced Message Queuing Protocol (AMQP) oder auf ein UDP, TCP oder Unix Socket geschrieben werden. Es werden aber auch Datenbanken wie die CrateDB unterstützt.

Aggregator

Neben dem Auslesen und direkten Weiterleiten der Daten an die Datenbank, kann Telegraf auch mithilfe von aggregator Plugins Daten gezielt verdichten, bevor diese gespeichert werden. Das minmax Plugin sammelt Daten über einen vordefinierten Zeitraum und ermittelt daraus den minimalen und maximalen Wert.

[33] Unter Debian: /etc/telegraf/telegraf.conf
[34] https://github.com/influxdata/telegraf/tree/master/plugins

Processor

Processor Plugins können verwendet werden, um die gesammelten Daten zu transformieren oder zu filtern. Derzeit gibt es allerdings nur ein Plugin, welches seit über einem Jahr nicht mehr gepflegt und nicht dokumentiert wurde.

Um ein Plugin zu verwenden, muss lediglich die Konfiguration angepasst werden. Folgende Konfiguration reicht aus, um Informationen über den Prozessor, den Arbeitsspeicher und die Syslog Datei auszulesen und an InfluxDB[35] zu senden.

```
[global_tags]
  department = "Testlabor"

[agent]
  interval = "10s"
  ...
  hostname = "vmtest01"

# Send metrics to the monitoring instance
[[outputs.influxdb]]
  urls = ["192.168.0.5"]
  ...

[[inputs.cpu]]
   ...

[[inputs.mem]]
```

Listing 4.9: TICK-Stack Telegraf Ausschnitt der Konfiguration

Zusätzlich werden den gesammelten Daten weitere Informationen wie der hostname (vmtest01) und das department (Testlabor) zugeordet. Dies ermöglicht bei der Auswertung das Filtern nach bestimmten hostnames oder das Gruppieren nach bestimmten Departments. Telegraf unterstützt nicht nur viele unterschiedliche Arten von Daten, sondern ist für diverse Plattformen wie OS X, Ubuntu, Debian oder Windows verfügbar und kann auch direkt als Docker Container installiert werden[Infm]. Eine Einschränkung in Bezug auf die Maximalanzahl an Agenten gibt es nicht, wodurch die Anforderung der Skalierbarkeit erfüllt ist.

4.3.5 Elastizität

Anpassungen am Monitoringsystem wie das Hinzufügen neuer Instanzen erzeugt keine Probleme, da der TICK-Stack die Daten mithilfe von Agenten direkt auf dem Hostsystem sammelt. Der Agent, beim TICK-Stack Telegraf genannt, kann ohne Beeinträchtigung des Gesamtsystems auf einer neuen Instanz installiert werden und sendet die Daten an die Datenbank. Eine Anpassung der Datenbank ist nicht notwendig. Ebenfalls beeinträchtigt das Löschen einer Instanz das System nicht[Info]. Einen Deploymentserver der die Konfiguration zentral Verwaltet, ähnlich wie bei Splunk, gibt es bei TICK-Stack nicht.

[35] In diesem Beispiel ist InfluxDB unter folgender IP erreichbar: 192.168.0.5

4.3.6 Aktualität

Telegraf ist der Agent, der die Daten direkt auf dem Hostsystem sammelt. Über die Konfigurationsdatei kann das Intervall bestimmt werden, wie oft die Daten an die Datenbank weitergeleitet werden. Zusätzlich kann für jedes Input Plugin ein individueller Intervall bestimmt werden. So können weniger kritische Werte seltener an die Datenbank gesendet werden, um das Netzwerk und das Hostsystem zu entlasten. Bei kritischen Werten kann Telegraf die Daten sekündlich abfragen und weiterleiten[Infd]. Um auf mögliche Ereignisse direkt hingewiesen zu werden bzw. eventuell sogar automatisch reagieren zu können, bietet der TICK-Stack den Kapacitor an. Kapacitor kann bei Überschreitung bestimmter Grenzwerte oder bei Auftreten eines bestimmten Events reagieren[Infg]. Als Reaktion kann Kapacitor beispielsweise eine E-Mail mit zusätzlichen Details über das Event versenden. Genauso können Skripte ausgeführt oder TCP-Anfragen versendet werden, um möglicherweise automatisch das Problem zu beheben[Infa].

4.3.7 Vollständigkeit, Erweiterbarkeit, Beeinträchtigend

Der TICK-Stack ist ein vollwertiges Monitoringsystem und unterstützt sowohl diverse Plattformen, als auch verschiedene Datentypen. Wie in Kapitel 4.3.4 beschrieben, kann Telegraf durch Plugins erweitert werden. Neben dem GitHub Repository für Telgraf Plugins[36], gibt es keine offizielle Seite für Erweiterungen der anderen TICK-Stack Komponenten. Zur Überwachung virtueller Maschinen der Virtualisierungssoftware VMware gibt es beispielsweise nur eine inoffizielle Erweiterung der französischen Firma Oxalide[37], die sie auf ihrem GitHub Repoistory[38] zur Verfügung stellen. Ob diese Erweiterung den Ansprüchen einer Nutzung im Produktivumfeld entspricht, ist allerdings unklar.

Eine sogennante anomalie detection kann in den Kapacitor ebenfalls integriert werden. Anders als bei dem Machine Learning Toolkit von Splunk, muss die anomalie detection per user-defined function selbst umgesetzt werden und bietet keine GUI zur einfachen Konfiguration. In der Kapacitor Dokumentation[39] der InfluxData gibt es zwar ein Beispiel für die Überwachung der Temperatur bei einem 3D-Drucker, für andere Szenarien muss der Code aber angepasst werden[Infe]. Libarys wie die Machine Learning libary scikit-learn[40] können über die user-defined function implementiert werden. Dadurch, dass externe libarys unabhängig vom TICK-Stack entwickelt werden können, ist der TICK-Stack auch zukünftig durch das Einbinden von libarys erweiterbar.

Die Komponente Grafana[41] kann als Alternative zum Chronograf verwendet werden. Anders als Chronograf ist Grafana nicht an den TICK-Stack gebunden[Arb17]. Neben der InfluxDB und der Elasticsearch Datenbank werden derzeit 37 weitere Datenquellen unterstützt[Graa]. Zur Darstellung der Daten unterstützt Grafana neben den klassischen Darstellungen wie Tabellen und Diagrammen, weitere Darstellungsformen die durch Plugins erweitert werden können. Auf der Grafana Plugin Website[42] können kostenlos Plugins heruntergeladen werden. Diese erweitern

[36] https://github.com/influxdata/telegraf/tree/master/plugins
[37] https://www.oxalide.com
[38] https://github.com/Oxalide/vsphere-influxdb-go
[39] https://docs.influxdata.com/kapacitor
[40] http://scikit-learn.org
[41] https://grafana.com
[42] https://grafana.com/plugins

Grafana um neue Darstellungsformen wie zum Beispiel einer heatmap, einem gauge panel oder komplette dashboards zum Verwalten weltweiter pingtests. Da der TICK-Stack aus verschiedenen Komponenten (Telegraf, InfluxDB, Kapacitor, Chronograf) besteht, können einzelne Komponenten erweitert werden, ohne das Gesamtsystem zu beeinträchtigen. Außerdem können zur Visualisierung der gesammelten Daten weitere Visualisierungssysteme, wie Grafana, eingesetzt werden, welches ebenfalls durch Plugins erweitert werden kann. Somit erfüllt der TICK-Stack alle drei aufgestellten Anforderungen.

4.3.8 Belastbarkeit, Zuverlässigkeit, Verfügbarkeit

Das Monitoringsystem TICK-Stack ermöglicht bei Verwendung der kostenpflichtigen InfluxEnterprise und InfluxCloud Lizenz das Bilden von Clustern. Dies erhöht durch die Verteilung der Anfragen und dem Entfernen des single points of failure die Belastbarkeit, Zuverlässigkeit und Verfügbarkeit. Somit kann, trotz Ausfall einiger Komponenten, das Monitoringsystem noch zuverlässig funktionieren. Zur Umsetzung des Clusters, werden folgende Komponenten angeboten.

InfluxDB Enterprise

Die Enterprise Version von InfluxDB unterscheidet zwischen Meta Nodes und Data Nodes. Meta Nodes speichern keine Messdaten, sondern ausschließlich Daten, die für die Organisation des Clusters notwendig sind. Unter anderem werden Informationen über alle Nodes im Cluster gespeichert und welche Benutzer berechtigt sind auf diese zuzugreifen. Außerdem werden alle laufenden Abfragen gespeichert. Die Meta Nodes kommunizieren untereinander über das TCP-Protokoll und tauschen so Änderungen aus[Infb]. Da zum Austausch der Daten das Konsens-Protokoll verwendet wird, empfiehlt es sich immer eine ungerade Anzahl an Meta Nodes zu verwenden. Das Konsens-Protokoll benötigt bei Änderungen immer ein Quorum. Selbst wenn ein node dauerhaft ausfällt, gibt es durch die zwei verbleibenden Meta Nodes immer noch ein Quorum und Änderungen können so vorgenommen werden [DKC05].

Data Nodes speichern dagegen ausschließlich Messdaten. Über den Replikationsfaktor wird die Anzahl der Kopien und die so benötigten Data Nodes bestimmt. Zusätzlich können sogenannte shared groups gebildet werden, um die einzelnen nodes besser auszulasten. Dabei werden die Messdaten aus bestimmten Zeitbereichen nur in bestimmten nodes gespeichert[Infb]. So werden bei vier Data Nodes beispielsweise alle Werte aus dem Jahr 2017 nur in Node A und B (Shared Group 1) und alle Werte aus dem Jahr 2018 nur in Node C und D (Shared Group 2) gespeichert.

Kapacitor Enterprise

Kapacitor Enterprise erweitert die open source Komponente Kapacitor um die Möglichkeit Cluster zu erstellen. Dabei werden mehrere Instanzen von Kapacitor zu einem Cluster zusammengeschlossen. Jede Kapacitor Instanz kennt jeweils die anderen Kapacitor Instanzen in dem Cluster. Zum Austausch der Informationen wird das Gossip-Protokoll verwendet[Infh]. Durch die Nutzung des Protokolls, wird der Overhead verringert und gleichzeitig die Verfügbarkeit erhöht, da eine Veränderung nicht sofort an alle anderen Nodes verteilt wird. Dafür besteht nur eine eventual consistency und erzeugt dadurch den Effekt, dass zwei nodes zur selben Zeit verschiedene Werte zurückgeben[DGH+87]. Um diesen Effekt zu minimieren, kann in der Konfigurationsdatei der gossip-interval verringert werden[Infh].

Somit bietet der TICK-Stack, zumindest in der kostenpflichtigen Variante, Möglichkeiten um die Belastbarkeit, Zuverlässigkeit und Verfügbarkeit zu verbessern.

4.3.9 Lizenzierung

Ähnlich wie auch beim ELK-Stack ist der TICK-Stack nicht vollständig kostenlos. Während bei dem ELK-Stack unter anderem das Benachrichtigungssystem nur bei der kommerziellen Version nutzbar ist, können alle Funktionen des TICK-Stacks auch in der kostenlosen Version genutzt werden. Wird eine höhere Verfügbarkeit oder Skalierbarkeit benötigt, muss die kommerzielle Version des TICK-Stacks[43] InfluxEnterprise oder InfluxCloud verwendet werden. Bei der InfluxCloud werden die Komponenten auf einer Amazon Web Services (AWS) Instanz installiert und von der InfluxData Inc. verwaltet. InfluxEnterprise ermöglicht den Aufbau einer geclusterten TICK-Stack Infrastruktur im internen Netzwerk oder bei einem Cloud-Anbieter wie Google Cloud[44] oder Microsoft Azure[45].

Die Komponenten Telegraf, InfluxDB und Kapacitor wurden von der InfluxData Inc. unter der MIT Lizenz veröffentlicht[Infs], welche kompatibel zur Apache License 2.0 ist[Thec]. Der Quellcode ist über das GitHub Repository der InfluxData Inc.[46] abrufbar. Die Komponente Chronograf wurde unter der GNU GPLv3 Lizenz veröffentlicht[Infn], welche ebenfalls kompatibel zur Apache License 2.0 ist[Fre17]. Da die kostenlose Variante des TICK-Stacks alle funktionalen Anforderungen erfüllt und der Quellcode unter einer kompatiblen Lizenz veröffentlicht wurde, zählt die Anforderung als erfüllt.

4.3.10 Automatisierbarkeit

Der TICK-Stack bietet den Telegraf als Agenten an. Die Installation erfolgt unter dem Betriebssystem Ubuntu über die Paketverwaltung APT und unter Windows über den Windows installer[Inff]. Die Konfiguration erfolgt ausschließlich über die Konfigurationsdatei. Ein zentrale Konfiguration über einen Deploymentserver, wie Splunk es anbietet, ist nicht möglich.

[43] https://www.influxdata.com/products/editions/
[44] https://cloud.google.com/
[45] https://azure.microsoft.com
[46] https://github.com/influxdata

4.4 Prometheus

Prometheus wurde ursprünglich als interne Monitoringlösung von dem deutschen Unternehmen SoundCloud Limited entwickelt. Seit 2016 wird Prometheus als eigenständiges Projekt von der Cloud Native Computing Foundation[47] verwaltet, die wiederum eine Tochter der Linux Foundation[48] ist[Los]. Abbildung 4.10 beschreibt die Architektur von Prometheus. Die Kernkomponente ist der Prometheus Server. Dieser speichert nicht nur die gesammelten Daten, sondern bietet eine API zum Auslesen der Daten an. Zusätzlich werden über den Prometheus Server die Metriken direkt von den zu überwachenden Systemen abgerufen[Los].

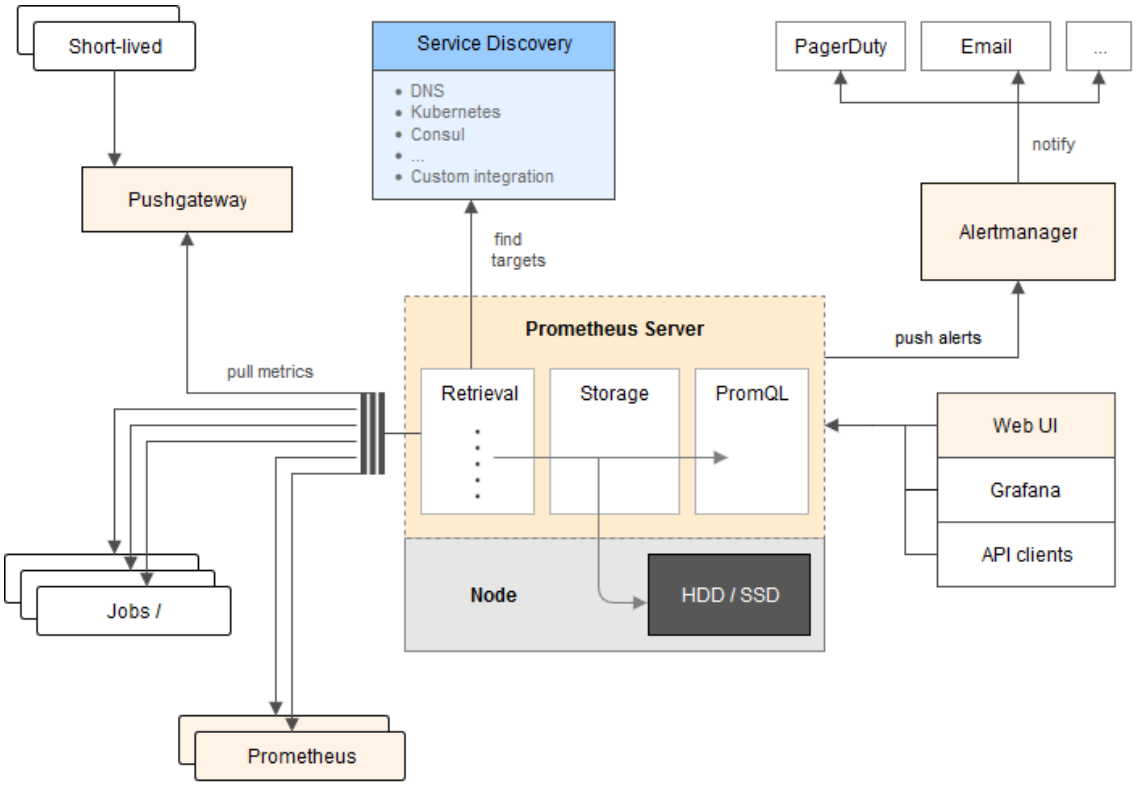

Abbildung 4.10: Architektur Prometheus Quelle: prometheus.io[Proc]

Die drei bisher vorgestellten Monitoringsysteme waren alle push-basiert. Das heißt, auf den zu überwachenden Systemen wird ein Agent installiert, der in regelmäßigen Abständen die gesammelten Daten an die Datenbank sendet. Dies hat den Vorteil, dass dynamisch neue Instanzen hinzugefügt und entfernt werden können, ohne das Monitoringsystem neu konfigurieren zu müssen. Pull-basierte Systeme wie Prometheus nutzen ebenfalls Agenten. Diese haben aber nur die Aufgabe Daten zu sammeln. Der zentrale Server ruft in regelmäßigen Abständen die Messdaten von den jeweiligen Agenten ab. Dies hat wiederum den Vorteil, dass zentral auf dem Server das Abfrageintervall bestimmt wird[Los]. Gespeichert werden die Daten ebenfalls direkt auf dem

[47] https://www.cncf.io
[48] https://www.linuxfoundation.org

Prometheus Server. Dafür verwendet Prometheus die von Google entwickelte NoSQL Datenbank LevelDB[49][Los]. Die Abfrage der Daten erfolgt über eine REST-Schnittstelle und der, an SQL angelehnten, Query-Sprache PromQL. Tools wie Grafana können diese Schnittstelle nutzen, um die gesammelten Daten zu visualisieren[Los]. Um neue zu überwachende Systeme zu lokalisieren, verwendet Prometheus bestehende Dienste wie DNS, Kubernetes[50] oder Consul[51]. Zum Sammeln der Daten verwendet Prometheus sogenannte Exporter, welche direkt auf den jeweiligen Systemen installiert sind. Diese Sammeln die Daten und bieten sie dem Prometheus Server zum Download an. Für temporäre Systeme bietet Prometheues den Pushgateway an. Dieser kann für Server oder Skripte zur kurzzeitigen Nutzung bereitgestellt werden. Der Pushgateway erlaubt es, die gesammelten Daten per Push an den Gateway zu senden. Dieser sammelt die Daten und bietet sie wiederum den Prometheus Server zum Download an[Prod]. Die letzte Komponente ist der Alertmanager. Ähnlich wie bei den bereits vorgestellten Benachrichtigungssystemen, können auch bei dem Monitoringsystem Prometheus Regeln definiert werden. Bei Überschreitung eines Grenzwertes, sendet der Prometheus Server einen Alert an den Alertmanager. Der Alertmanager leitet die Benachrichtigung an Services wie PagerDuty weiter oder sendet alternativ eine E-Mail[Prob].

Funktionale Anforderungen

Prometheus wurde als System zum Sammeln von Prozess-Metriken entwickelt. Somit können Buchstaben oder ganze Sätze nicht ausgewertet bzw. gespeichert werden[Proa]. Da die Auswertung und Speicherung von Log-Dateien Teil der funktionalen Anforderung ist, kommt das Monitoringsystem zur weiteren Verwendung nicht in Frage und wird daher nicht weiter evaluiert.

[49] http://leveldb.org
[50] https://kubernetes.io
[51] https://www.consul.io

4.5 Vergleich

Zur besseren Übersicht wurde die Evaluation der Monitoringsysteme in Tabelle 4.1 zusammengefasst.

Anforderung	Splunk	ELK-Stack	TICK-Stack	Prometheus
Funktional				
Numerische Metriken	√	√	√	√
Alphanumerische Metriken	√	√	√	x
Alerting	√	√	√	-
Nicht-Funktional				
Skalierbarkeit	√	√	√	-
Elastizität	√	√	√	-
Aktualität	√	√	√	-
Vollständigkeit, Erweiterbarkeit, Beeinträchtigend	√	√	√	-
Belastbarkeit, Zuverlässigkeit, Verfügbarkeit	√	√	√	-
Lizenzierung	x	x	√	-
Automatisierbarkeit	√	x	x	-
Beispielszenario	√	√	√	x

Tabelle 4.1: Vergleich der Anforderungen

Wie die Evaluation von Splunk aufzeigt, erfüllt das Monitoringsystem alle funktionalen Anforderungen und somit auch die Anforderungen durch das Beispielszeanario Arena 2036. Ebenso werden die nicht-funktionalen Anforderungen, abgesehen von der Lizenzierung, erfüllt. Die Lizenz der Splunk Komponenten ist dabei nicht kompatibel zu der Lizenz des OpenTOSCA Ökosystems.

Der ELK-Stack erfüllt alle funktionalen Anforderungen und somit auch die Anforderungen durch das Beispielszeanario Arena 2036. Die nicht-funktionalen Anforderungen werden mit Ausnahme der Lizenzierung ebenfalls erfüllt. Da das Benachrichtigungssystem nicht in der kostenlosen Version enthalten ist, zählt diese Anforderung als nicht erfüllt. Außerdem ist die Möglichkeit zur automatisierten Installation und Konfiguration des Agenten nicht direkt im ELK-Stack implementiert.

Das Monitoringsystem Prometheus unterstützt keine alphanumerischen Metriken. Da dies Teil der funktionalen Anforderung ist, kommt das System für eine Implementierung nicht in Frage.

Der TICK-Stack ist das einzige Monitoringsystem der Evaluation, das die Anforderung der Lizenzierung erfüllt. Abgesehen von der automatisierten Installation und Konfiguration des Agenten, werden alle anderen Anforderungen erfüllt. Da die Installation des Agenten mithilfe von OpenTOSCA automatisiert werden kann, wird das Monitoringsystem für die Implementierung in das OpenTOSCA Ökosystem und die Integration in das Beispielszenario in der Arena 2036 genutzt.

5 Konzept

Nachdem in dem vorherigen Kapitel verschiedene Monitoringsysteme evaluiert wurden, soll in diesem Kapitel ein Konzept zur Integration eines Monitoringsystems vorgestellt werden. Das Monitoringsystem TICK-Stack hat sich dabei als geeignet herausgestellt. Abgesehen von der Automatisierbarkeit, erfüllt der TICK-Stack alle aufgestellten Anforderungen. Anders als bei anderen Monitoringsystemen mit Deploymentservern, erfolgt die Konfiguration der Agenten ausschließlich über die Konfigurationsdatei. Mithilfe des OpenTOSCA Ökosystems kann die Installation und Konfiguration der TICK-Stack Agenten allerdings problemlos automatisiert werden.

In der aktuellen Version des OpenTOSCA Ökosystems gibt es bisher keine Möglichkeit provisionierte Instanzen zu überwachen. Beispielsweise könnten, ohne das Wissen des Nutzers, provisionierte Instanzen beendet werden. Die folgende Integration des Monitoringsystems TICK-Stacks soll dies ermöglichen. So kann in der OpenTOSCA GUI Vinothek der aktuelle Status einer provisionierten Instanz direkt dargestellt und mögliche Probleme erkannt werden.

In Kapitel 4.3 wurde der Aufbau des Monitoringsystems mit den vier TICK-Stack Komponenten (Telegraf, InfluxDB, Chronograf und Kapacitor) detailliert beschrieben. Diese können in Monitoring-Service-Komponenten und Agent unterteilt werden. Monitoring-Service-Komponenten haben die Aufgaben Daten zu speichern (InfluxDB), zu visualisieren (Chronograf) und auf Ereignisse zu reagieren (Kapacitor). Diese drei Komponenten können auf einen eigenen Server ausgelagert werden. Eine Installation auf dem zu überwachenden System ist nicht notwendig, da der Agent (Telegraf) gesammelte Daten an die Datenbank (InfluxDB) sendet. Die drei Monitoring-Service-Komponenten müssen dabei nicht auf dem gleichen System installiert sein wie der Agent. Anders als die Monitoring-Service-Komponenten wird der Agent Telegraf direkt auf dem zu überwachenden System installiert. Dort überwacht dieser, abhängig von den jeweils installierten Input Plugins, das Hostsystem und sendet die gesammelten Daten an die Datenbank. Monitoring-Service-Komponenten und Agent werden im Folgenden getrennt voneinander betrachtet.

Kapitel 5.1 beschreibt zwei Methoden wie die Monitoring-Service-Komponenten (InfluxDB, Chronograf und Kapacitor) in das OpenTOSCA Ökosystem integriert werden können. Kapitel 5.2 betrachtet wie der Agent Telegraf auf den zu überwachenden Systemen bereitgestellt werden kann. Dafür werden ebenfalls zwei verschiedene Methoden beschrieben. Das Visualisieren gesammelter Daten gehört ebenfalls zu den Aufgaben eines Monitoringsystems. In Kapitel 5.3 werden Konzepte vorgestellt, wie die gesammelten Daten dargestellt werden können.

5.1 Monitoring-Service

Die drei Monitoring-Service-Komponenten (InfluxDB, Chronograf und Kapacitor) können durch verschiedene Methoden in das bestehende OpenTOSCA Ökosystem integriert werden. Als Ansatz dient die Designstruktur von Stevens et al.[SMC74]. Diese beschreiben das Aufteilen einzelner Programmmodule, um die Komplexität und Abhängigkeit zu verringern. Das Aufteilen bzw. Entkoppeln einzelner Komponenten sorgt außerdem dafür, dass bei Fehlern in einzelnen Komponenten nicht das gesamte System betroffen ist. Dieser Ansatz der lose gekoppelten Integration wird in Kapitel 5.1.1 beschrieben. Alternativ kann der TICK-Stack zusammen mit dem OpenTOSCA Ökosystem über ein gemeinsames Installationsskript auf dem selben System installiert werden. Dies wird als eng gekoppelte Integration bezeichnet und in Kapitel 5.1.2 beschrieben.

5.1.1 Lose Kopplung

Bei einer losen Kopplung zwischen dem TICK-Stack und dem OpenTOSCA Ökosystem, werden die drei Monitoring-Service-Komponenten (InfluxDB, Chronograf und Kapacitor) unabhängig von OpenTOSCA installiert. Die drei Monitoring-Service-Komponenten werden über das OpenTOSCA Ökosystem instanziiert. Dies kann als Monitoring-as-a-Service bezeichnet werden, da die Komponenten wie jede andere Applikation instanziiert oder auch gelöscht werden kann. Dies ermöglicht den problemlosen Austausch des Monitoringsystems.

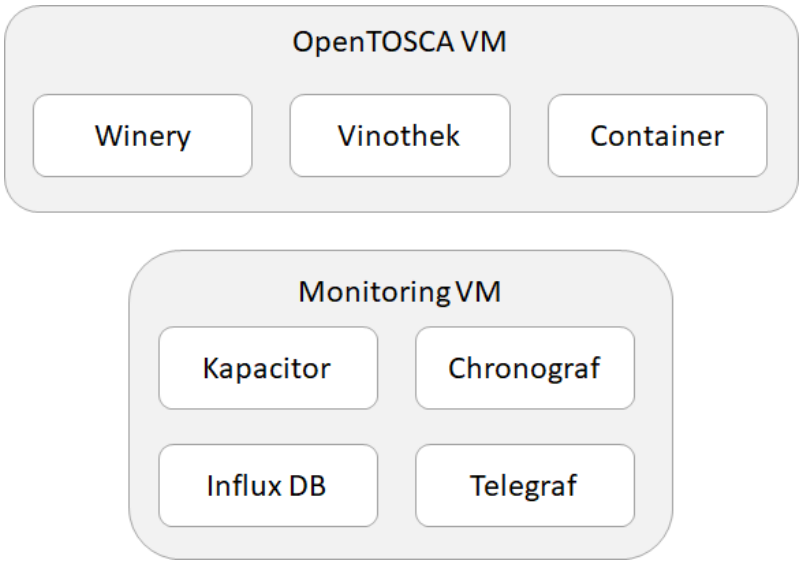

Abbildung 5.1: Architektur OpenTOSCA Monitoring-as-a-Service (Auszug)

Abbildung 5.1 visualisiert die Architektur des OpenTOSCA Ökosystems mit lose gekoppelten Monitoring-Service-Komponenten. Neben einer virtuellen Maschine bestehend aus den OpenTOSCA Komponenten (Winery, Vinothek und Container), ist die Monitoring-as-a-Service Instanz abgebildet. Diese wird über das OpenTOSCA Ökosystem provisioniert und besteht aus den drei Monitoring-Service-Komponenten InfluxDB, Chronograf, Kapacitor und dem Agenten Telegraf. Der Agent überwacht dabei die Daten der Instanz selbst. Dies ermöglicht die Überwachung der

Monitoring-as-a-Service Instanz. Folgend werden die Vor- und Nachteile einer losen Kopplung der Monitoring-Service-Komponenten und des OpenTOSCA Ökosystems beschrieben.

Vorteil

- Die Monitoring-Service-Komponenten können unabhängig von dem OpenTOSCA Ökosystem aktualisiert oder neu installiert werden.

- Die Monitoring-Service-Komponenten können unabhängig von den OpenTOSCA Komponenten ausgetauscht werden. Dies ist beispielsweise bei einem Wechsel des Monitoringsystem sinnvoll, falls dieses nicht weiterentwickelt wird.

Nachteil

- Die von den Agenten gesammelten Daten werden erst nach der Installation der Monitoring-Service-Komponenten verarbeitet und gespeichert. Somit besteht zwischen Installation von OpenTOSCA und Instanziierung der Monitoring-Service-Komponenten ein Zeitfenster, in dem keine gesammelten Daten gespeichert werden können.

- Unerfahrene Nutzer können versehentlich die Monitoring-Service-Komponenten und damit auch alle gesammelten Daten entfernen.

- Die Adresse der Monitoring-Service-Komponenten muss ausgelesen und an die Agenten übermittelt werden.

5.1.2 Enge Kopplung

Bei einer engen Kopplung der TICK-Stack-Komponenten und dem OpenTOSCA Ökosystem, werden die drei Monitoring-Service-Komponenten (InfluxDB, Chronograf und Kapacitor) fest in das OpenTOSCA Ökosystem integriert. Bei der Installation von OpenTOSCA, werden die drei Komponenten automatisch installiert und konfiguriert.

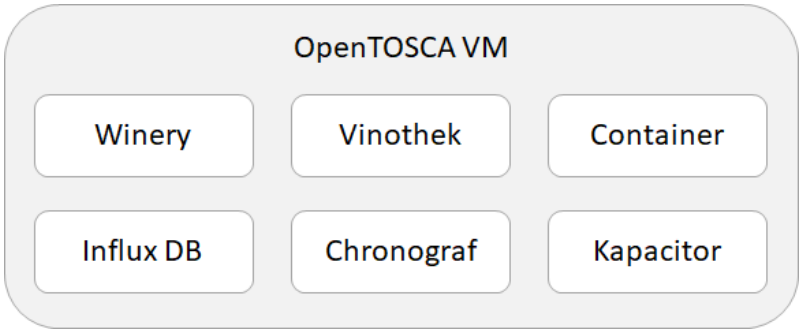

Abbildung 5.2: Architektur OpenTOSCA: Enge Kopplung (Auszug)

Abbildung 5.2 zeigt ausschnittsweise Komponenten des OpenTOSCA Ökosystems und Monitoring-Service-Komponenten. Dabei werden sowohl die drei OpenTOSCA Komponenten (Winery, Vinothek und Container), als auch die drei Monitoring-Service-Komponenten (InfluxDB, Chronograf und Kapacitor) über ein gemeinsames Skript installiert. Die Komponenten können dabei wie in Abbildung 5.2 auf einer gemeinsamen oder auf getrennten virtuellen Maschinen installiert werden. Eine Aufteilung der Komponenten auf getrennte virtuelle Maschinen kann dabei verhindern, dass

die Belegung des Arbeitsspeichers durch eine Komponente, das ganze OpenTOSCA Ökosystem behindert.

Vorteil

- Die Monitoring-Service-Komponenten benötigen kein eigenes Installationsskript, da diese zusammen mit den OpenTOSCA Komponenten installiert werden.

- Monitoring-Service-Komponenten sind nach Installation von OpenTOSCA direkt verfügbar und können sofort Daten der Agenten empfangen und verarbeiten.

- Die Monitoring-Service-Komponenten sind über eine feste Adresse erreichbar.

Nachteil

- Die Monitoring-Service-Komponenten können nicht unabhängig von OpenTOSCA aktualisiert oder neu installiert werden.

- Die Monitoring-Service-Komponenten können nicht unabhängig von den OpenTOSCA Komponenten ausgetauscht werden.

Eine lose Kopplung in Form eines Monitoring-as-a-Service sorgt für einen hohen Grad an Flexibilität, da das Monitoringsystem beliebig modifiziert oder ausgetauscht werden kann, ohne Einfluss auf die anderen OpenTOSCA Komponenten zu nehmen. Bei einer engen Kopplung dagegen sind die Monitoring-Service-Komponenten ab der Installation des OpenTOSCA Ökosystems verfügbar und können Daten der Agenten speichern und verarbeiten.

Wie in Kapitel 4.3.4 beschrieben, verwendet der TICK-Stack den Plugin-basierten Agenten Telegraf um Daten auf zu überwachenden Systemen zu sammeln. Die Implementierung des Agenten auf der über OpenTOSCA provisionierten Instanz, ist ebenfalls über zwei Methoden möglich. Dies wird in Kapitel 5.2 detailliert erläutert.

5.2 Agent

Anders als die Monitoring-Service-Komponenten muss der Agent Telegraf auf dem zu überwachenden System installiert werden. Wie bereits in Kapitel 4.3.10 beschrieben verfügt der Agent über keine Möglichkeit, automatisch auf einem neuen System installiert zu werden. Mit der Verwendung von OpenTOSCA ist eine automatisierte Installation und Konfiguration des Agenten auf eine neu provisionierte Instanz möglich.

Abbildung 5.3 visualisiert den Ablauf der einzelnen Provisionierung-Schritte in Form eines Ablaufdiagramms. Der rot markierte Bereich zeigt den im Rahmen dieser Arbeit implementierten Agenten. Der Ablauf besteht dabei aus drei Phasen. In der Pre-Phase werden zuerst die ImplementationArtifacts und anschließend die DeploymentArtifacts bereitgestellt. Das Installationsskript, sowie die verschiedenen Konfigurationsdateien für den Agenten, wird in dieser Phase ebenfalls auf den jeweiligen Node bereitgestellt[Kép13]. Im Anschluss folgt die Provisioning-Phase. Hier wird eine Reihe von Management-Operationen ausgeführt. Dies kann beispielsweise die Installation des Betriebssystems Ubuntu sein[Kép13]. In der Post-Phase wird mithilfe von Consistency Activities die Konsistenz der CSAR-Datei wiederhergestellt, die in der Pre- und Provisioning-Phase verletzt wurde. Mithilfe der Consistency Activities werden der Instanz aktuelle Eigenschaften (engl. properties) übergeben[Kép13]. Diese werden verwendet, um in der Telegraf Konfigurationsdatei die

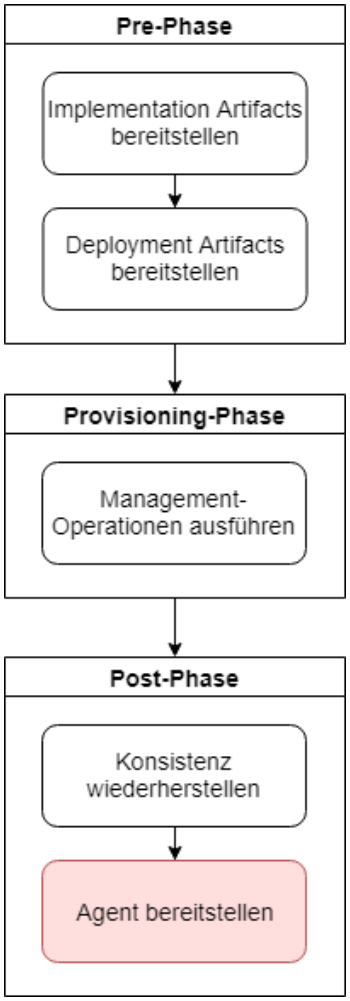

Abbildung 5.3: Ablaufdiagramm OpenTOSCA angelehnt an Képes[Kép13]

Variablen zu definieren. Zusätzlich wird in dieser Phase der Agent Telegraf installiert und die Konfigurationsdateien in das passende Verzeichnis kopiert.

Die Integration des Agenten kann unterschiedlich umgesetzt werden. Eine Möglichkeit zur Integration des Agenten ist die Entwicklung eines zusätzlichen Node Types (Monitoring Node Type). Alternativ kann der Agent über eine separate Monitoringmethode umgesetzt werden. Folgend werden die beiden Varianten beschrieben.

5.2.1 Monitoring Node Type

Die Installation und Konfiguration des Agenten kann über einen zusätzlichen Node Type erfolgen. Dieser wird auf Topologieebene mit den jeweiligen zu überwachenden Node Types verbunden. Eine Anpassung im Plan Builder ist nicht notwendig. Dies sorgt für eine Erhöhung der Komplexität auf Topologieebene. Eine separate Entscheidung für die Überwachung einzelner Nodes ist über den Plan Builder nicht möglich.

5.2.2 Monitoringmethode

Alternativ kann der Agent auch über eine Monitoringmethode bereitgestellt werden. Dabei erhält der Agent keinen eigenen Node Type, sondern erweitert die bestehenden Node Types um eine Monitoringmethode. Die Auswahl der zu überwachenden Komponenten wird über den Plan Builder getroffen. Dies ermöglicht die separate Überwachung jedes Nodes, ohne Änderung auf Topologieebene vorzunehmen. Gerade in Bereichen mit sensiblen Daten, kann somit die Erfassung der Daten eingeschränkt werden. Zusätzlich wird durch den Einsatz der Monitoringmethode die Komplexität auf Topologieebene verringert, da die Konfiguration über den Plan Builder erfolgt.

5.3 Visualisierung

Die Visualisierung der Daten erfolgt entweder über die TICK-Stack-Komponente Chronograf oder Grafana. Alternativ kann die bestehende OpenTOSCA GUI Vinothek durch Informationen über die Systemauslastung der provisionierten Instanzen erweitert werden. Die OpenTOSCA Komponente Vinothek dient als Schnittstelle zwischen dem Benutzer und dem OpenTOSCA Container. Mithilfe einer Web-GUI können Benutzer neue Instanzen erstellen und bestehende löschen. Diese kann genutzt werden, um die gesammelten Monitoringdaten zu visualisieren. Das Monitoring kann dabei lose oder eng mit der Vinothek gekoppelt werden. Bei einer losen Kopplung werden bestehende Schnittstellen genutzt, um die gesammelten Monitoringdaten in der Vinothek zu visualisieren. Eine enge Kopplung der Komponenten entsteht dagegen bei einem Zugriff auf die gesammelten Daten in der Datenbank über statische Adressen.

5.3.1 Lose Kopplung

Eine lose Kopplung der beiden Komponenten entsteht durch die Nutzung des bereits integrierten Logsystems des OpenTOSCA Ökosystems. Über die OpenTOSCA Container REST API[1] können neue Logeinträge angelegt werden. So kann das im TICK-Stack enthaltene Benachrichtigungssystem Kapacitor verwendet werden, um Monitoringdaten wie die Prozessorauslastung des Hostsystems regelmäßig über die REST API an die OpenTOSCA GUI Vinothek zu senden. Abbildung 5.4 zeigt die bereits in der Vinothek enthaltene Tabelle mit Logeinträgen einer erzeugten Instanz. Die Tabelle wird verwendet, um regelmäßig die Systemauslastung (CPU und RAM) der

Timestamp	Status	Message
2018-05-10 14:08	INFO	Executing operation waitForAvailability of NodeTemplate Ubuntu-14_04-VM
2018-05-10 14:08	INFO	Executing operation createVM of NodeTemplate Ubuntu-14_04-VM
2018-05-14 09:11	INFO	CPU: 23.5% - RAM: 11.9%
2018-05-14 09:12	INFO	CPU: 12.1% - RAM: 12.7%

Abbildung 5.4: Screenshot OpenTOSCA: Systemauslastung in Logsystem (lose Kopplung)

[1] https://github.com/OpenTOSCA/container/tree/master/org.opentosca.container.api

Instanz darzustellen. Für detaillierte Informationen kann zusätzlich noch ein Link zum Chronograf hinzugefügt werden. Dieser erlaubt das Übergeben von Suchparametern per HTTP-Anfrage.

Alternativ kann auch die bereits in der Web-GUI der Vinothek vorhande Tabelle über die provisionierten Instanzen erweitert werden. Abbildung 5.5 zeigt ein Mockup der Tabelle in der Vinothek. Diese wird um die Spalte Monitored erweitert und bildet den Status der provisionierten Instanz ab. Ist der Status grün, werden Daten von dem Agenten Telegraf an die Datenbank InfluxDB gesendet. Der Status rot dagegen zeigt, dass die Datenbank keine aktuellen Monitoringdaten der Instanz erhält. Dies kann als Indikator für eine Fehlfunktion der Instanz gewertet werden. Der Status kann über die OpenTOSCA Container REST API übergeben werden. Diese müsste in diesem Fall aber erweitert werden.

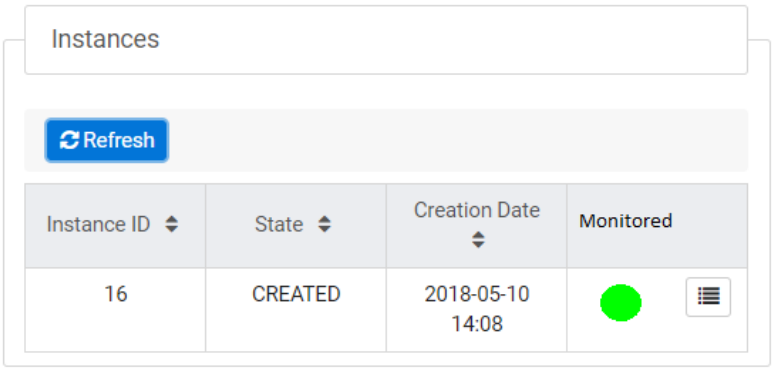

Abbildung 5.5: Mockup OpenTOSCA: Tabelle mit Statusanzeige des Hostsystems

5.3.2 Enge Kopplung

Wird bei der Integration auf keine bereits vorhanden Schnittstellen zurückgegriffen, sondern erfolgt die Abfrage beispielsweise über statische Adressen, wird dies als enge Kopplung der Komponenten bezeichnet. So kann das Monitoring mithilfe von Javascript Bibliotheken wie Chart.js[2] oder Highcharts[3] in die Vinothek integriert werden. Abbildung 5.6 zeigt wie die bestehende Website der Vinothek um einen neuen Reiter erweitert werden könnte. Dieser neue Reiter könnte mithilfe von Diagrammen und Tabellen die aktuelle Auslastung der Instanzen über einen längeren Zeitraum darstellen. Zusätzlich könnten aber auch Einträge aus den Logdateien angezeigt werden. Die Abfrage der Daten würde über die InfluxDB API erfolgen[Infk]. Bei einer Änderung der Adresse oder einem Austausch der verwendeten Datenbank, müsste die Anfrage in der OpenTOSCA Vinothek angepasst werden.

[2] https://www.chartjs.org
[3] https://www.highcharts.com

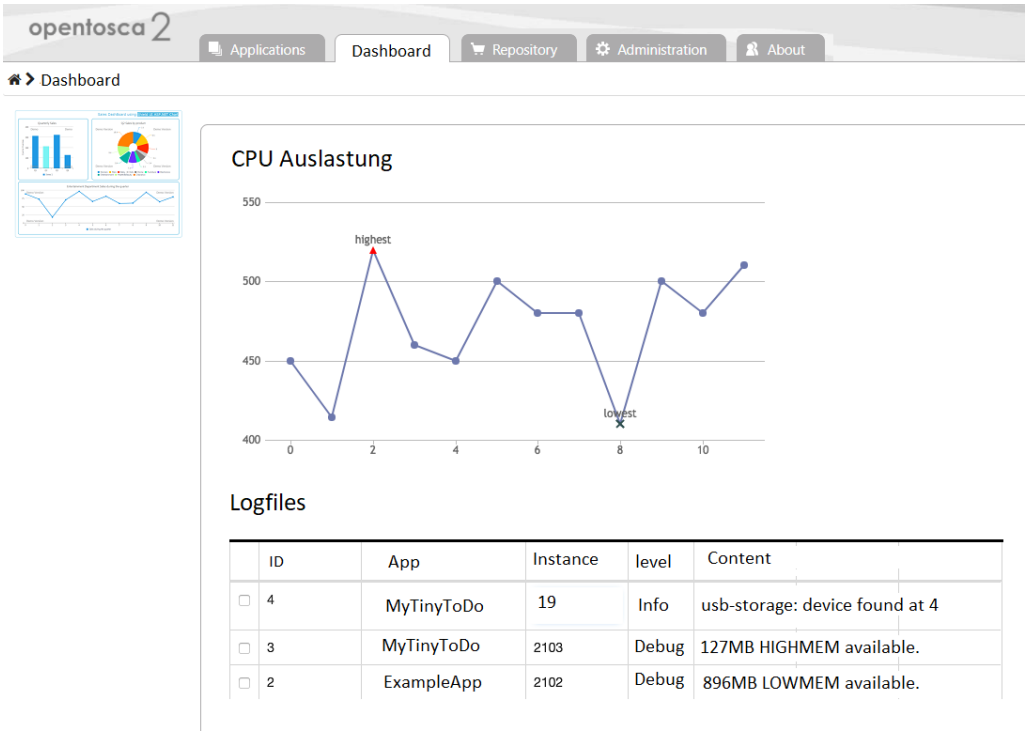

Abbildung 5.6: Mockup OpenTOSCA: Tabelle und Diagramm per Javascript Bibliothek

6 Implementierung

In diesem Kapitel sollen die in Kapitel 5 beschriebenen Konzepte implementiert werden. Der Fokus liegt dabei darauf, die Komponenten möglichst lose zu koppeln und vorhandene Schnittstellen zu verwenden. So können die Komponenten auch weiterhin getrennt weiterentwickelt und das Monitoringsystem gegebenenfalls komplett ausgetauscht werden. In Kapitel 6.1 wird das Konzept der lose gekoppelten Monitoring-Service-Komponenten aus Kapitel 5.1.1 implementiert. Die Implementierung des Agenten in Kapitel 6.2 erfolgt über die, in Kapitel 5.2.2 vorgestellte, neu erstellte Monitoringmethode. Für die Visualisierung der gesammelten Monitoringdaten in Kapitel 6.3 wird das bereits in OpenTOSCA vorhanden Logsystem verwendet. Dies entspricht einer losen Kopplung der Komponenten und wird in Kapitel 5.3.1 konzeptionell beschrieben.

6.1 Monitoring-Service

Die beiden Methoden der Integration der Monitoring-Service-Komponenten in OpenTOSCA (enge und lose Kopplung) wurden in Kapitel 5.1 konzeptionell beschrieben. Da bei einer losen Kopplung die einzelnen Komponenten (TICK-Stack und OpenTOSCA) unabhängig voneinander weiterentwickelt werden können, wird folgend eine lose gekoppelte Implementierung der Monitoring-Service-Komponenten beschrieben.

Für die Installation der Monitoring-Service-Komponenten werden Docker Container verwendet. Folgendes Bash-Skript installiert über das Paketverwaltungssystem Advanced Packaging Tool (APT) den Service Docker[1]. Anschließend wird das Docker Netzwerk tick-network erstellt und die vier TICK-Stack-Komponenten (Telegraf, InfluxDB, Chronograf und Kapacitor) werden als Docker Container gestartet.

```
#!/bin/bash

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu
    $(lsb_release -cs) stable"

sudo apt-get update
sudo apt-get install -y docker-ce

docker network create tick-network

docker run -d -it --name influxdb -p 8086:8086 --network tick-network influxdb
```

[1] https://www.docker.com

```
docker run -it -p 8888:8888 -d --name chronograf --network tick-network chronograf

docker run -it -d --network tick-network --name telegraf --hostname docker_tick_test
    -v {HOME}/telegraf.conf:/etc/telegraf/telegraf.conf:ro telegraf

docker run -d --network tick-network -p 9092:9092 --hostname kapacitor -e
    KAPACITOR_INFLUXDB_0_URLS_0=http://influxdb:8086 --name kapacitor kapacitor
```

Listing 6.1: Bash-Skript: Installation der Monitoring-Service-Komponenten als Docker Container

Abbildung 6.1 erweitert die, in Kapitel 5.1.2 vorgestellte, Architektur um die virtuelle Maschine Test VM, welche über OpenTOSCA provisioniert wurde. Auf der virtuellen Maschine Test VM wurden der Service Tomcat und der Agent Telegraf installiert. Der Agent sammelt Daten über den Systemzustand der virtuellen Maschine und des Tomcat Services und sendet diese an die Datenbank InfluxDB auf der virtuellen Maschine MonitoringVM.

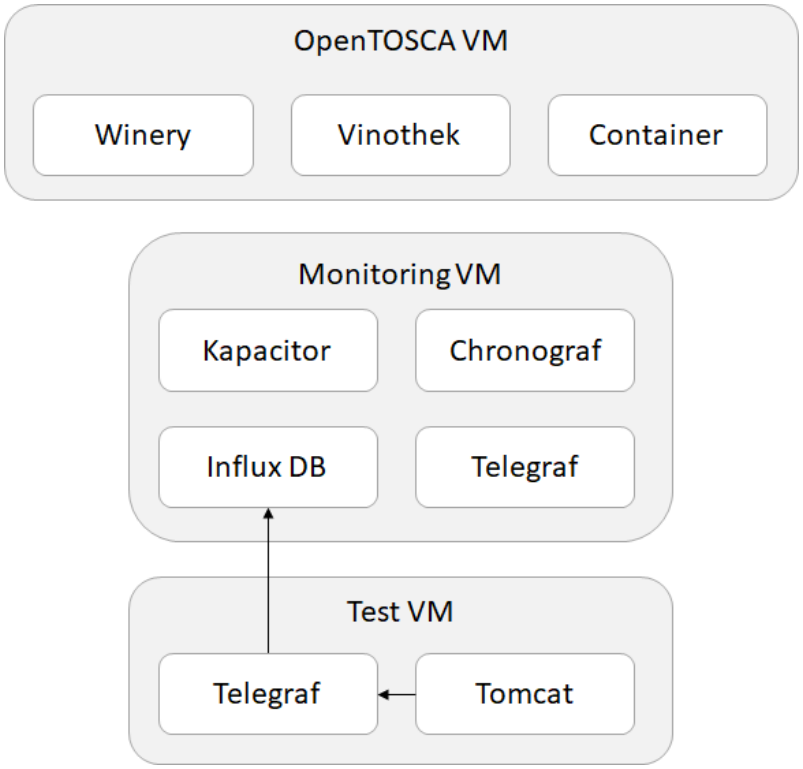

Abbildung 6.1: Architektur OpenTOSCA: Monitoring-as-a-Service mit Testmaschine (Auszug)

6.2 Agent

Für die Installation des Agenten auf dem Zielsystem wird die in Kapitel 5.2 vorgestellte Implementierung über eine Monitoringmethode beschrieben. Die Konfiguration erfolgt dabei nicht auf Topologieebene, sondern ausschließlich über den Plan Builder. Dies ermöglicht eine gesonderte Auswahl der zu überwachenden Nodes.

Der in Kapitel 4.3.4 vorgestellte Plugin-basierte Agent Telegraf muss, anders als die Monitoring-Service-Komponenten, direkt auf den zu überwachenden Systemen installiert sein. Eine Ausnahme bildet dabei das Input Plugin Ping[2]. Dieses kann genutzt werden, um die Verfügbarkeit eines Systems zu überwachen. Dabei muss sich der Agent mit dem Input Plugin Ping nur auf einem System befinden, welche das zu überwachende System über das Netzwerk erreichen kann. Zum Auslesen bestimmter Services wie Docker oder Tomcat, muss der Agent dagegen direkt auf dem System installiert sein.

Der Vorteil eines Plugin-basierten Agenten wie Telegraf besteht darin, dass für jeden Anwendungszweck spezifiziert werden kann, welche Plugins benötigt werden. Dies entlastet das Hostsystem und die Datenbank, da keine unnötigen Daten gesammelt werden. Die Auswahl der Plugins erfolgt über Konfigurationsdateien. Da Telegraf das Aufteilen der Konfigurationen ermöglicht, kann jedes Plugin über eine einzelnen Konfigurationsdatei verwaltet werden[Infc]. So können je nach Bedarf die Konfigurationsdateien einzelner Plugins hinzugefügt und entfernt werden. Wird beispielsweise auf einem Ubuntu System ein Tomcat installiert, kann die Konfigurationsdatei mit dem Input Plugin Tomcat[3] hinzugefügt werden. Die Konfigurationsdatei mit dem Input Plugin Docker[4] wird dagegen nicht hinzugefügt, da dieses in diesem Fall nicht benötigt wird. Das Speichern der gesammelten Daten erfordert ebenfalls ein Plugin. Das sogenannte Output Plugin InfluxDB[5] sendet die gesammelten Daten an die Datenbank InfluxDB. Das Output Plugin wird ebenfalls in eine gesonderte Konfigurationsdatei ausgelagert. Zusätzlich werden Input Plugins zur Überwachung des Hostsystems (Prozessorauslastung oder Arbeitsspeicherbelegung) hinzugefügt. Dies entspricht einer Minimalkonfiguration und wird auf jeder zur überwachenden Instanz installiert.

Für die Installation und Konfiguration des Agenten Telegraf werden über die OpenTOSCA Winery drei neue ArtifactTemplates erzeugt. Das ImplementationArtifact Ubuntu-14.04-VM-impl-deployAgent-ImplementationArtifact beinhaltet folgendes Bash-Skript:

```
#!/bin/bash

if ! [ -d "/etc/telegraf" ];then
   sudo curl -sL https://repos.influxdata.com/influxdb.key | sudo apt-key add -
      source /etc/lsb-release
      echo "deb https://repos.influxdata.com/${DISTRIB_ID,,} ${DISTRIB_CODENAME}
          stable" | sudo tee /etc/apt/sources.list.d/influxdb.list
      sudo apt-get update
      sudo apt-get install telegraf
```

[2] https://github.com/influxdata/telegraf/tree/master/plugins/inputs/ping
[3] https://github.com/influxdata/telegraf/tree/master/plugins/inputs/tomcat
[4] https://github.com/influxdata/telegraf/tree/master/plugins/inputs/docker
[5] https://github.com/influxdata/telegraf/tree/master/plugins/outputs/influxdb

```
fi

masterconfigfile=$(find ~ -maxdepth 20 -depth -iname "*telegraf*.conf*");
sudo rm /etc/telegraf/telegraf.conf
sudo cp $masterconfigfile /etc/telegraf

configfiles=$(find ~ -maxdepth 20 -depth -iname "*input.conf");
sudo cp $configfiles /etc/telegraf/telegraf.d

sudo service telegraf restart
```

Listing 6.2: Bash-Skript: Installation und Konfiguration des Agenten Telegraf

Das obige Bash-Skript prüft, ob der Agent Telegraf bereits installiert ist. Falls nicht, wird dieser über das Paketverwaltungssystem Advanced Packaging Tool (APT) installiert. Anschließend wird die Haupt-Konfigurationsdatei in der CSAR-Datei gesucht. Die Haupt-Konfigurationsdatei telegraf.conf beinhaltet die Minimalkonfiguration[6] und wird durch das DeploymentArtifact Ubuntu-14.04-VM-impl-agentconfig-DeploymentArtifact, siehe unten, hinzugefügt. Diese Datei wird anschließend in das Verzeichnis /etc/telegraf kopiert. Dateien mit dem Dateinamen *input.conf beinhalten weitere Input Plugins und werden in das Verzeichnis /etc/telegraf/telegraf.d kopiert. Im letzten Schritt wird Telegraf neu gestartet, um die neuen Konfigurationen zu übernehmen[Infc].

Das zweite ArtifactTemplate ist das DeploymentArtifact Ubuntu-14.04-VM-impl-agentconfig-DeploymentArtifact. Dieses beinhaltet die folgende Haupt-Konfigurationsdatei input.conf und dient als Haupt-Konfiguration für den Telegrafen:

```
[global_tags]
  department = "tosca_test"

[agent]
  ...
  hostname = "${VMIP}"

# Send metrics to the monitoring instance
[[outputs.influxdb]]
  urls = ["${InfluxDBIP}"]
  database = "telegraf"
  ...

[[inputs.cpu]]
  ...

[[inputs.diskio]]
[[inputs.mem]]
[[inputs.netstat]]
```

Listing 6.3: OpenTOSCA Telegraf Haupt-Konfiguration (Auszug)

[6] Die Minimalkonfiguration besteht aus einem Output Plugin um die gesammelten Daten speichern zu können und Input Plugins um den Systemzustand wie Prozessorauslastung zu erfassen

Die Konfigurationsdatei beinhaltet das Output Plugin Influxdb. Das Plugin sendet die gesammelten Monitoringdaten an die hinterlegte Adresse der Datenbank InfluxDB. Die Variable ["${InfluxDBIP}"] wird beim Provisionieren der Instanz durch den Plan Builder durch die passende Adresse der InfluxDB ersetzt. Weitere Parameter wie der Timeout oder der Name der Datenbank können später ebenfalls durch den Plan Builder festgelegt werden. Im Rahmen dieser Arbeit werden diese Werte statisch eingetragen. Neben dem Output Plugin Influxdb beinhaltet die Konfigurationsdatei Input Plugins zur Überwachung der Prozessorauslastung (inputs.cpu), des Arbeitsspeichers (inputs.mem), der Festplatten (inputs.diskio) und der Netzwerkkarten (inputs.netstat). Weitere Einstellungen wie Abfrageintervall oder Hostname können unter [agent] und [global_tags] konfiguriert werden.

Das dritte ArtifactTemplate ist das DeploymentArtifact Tomcat-impl-agentconfig-DeploymentArtifact. Dieses beinhaltet die Datei tomcat_input.conf und folgendes Input Plugin zur Überwachung des Service Tomcat:

```
[[inputs.tomcat]]
## URL of the Tomcat server status
 url = ["${TOMCAT_URL}"]

## HTTP Basic Auth Credentials
 username = ["${TOMCAT_USER}"]
 password = ["${TOMCAT_PW}"]

## Request timeout
 timeout = 5s

## Optional SSL Config
# ssl_ca = /etc/telegraf/ca.pem
# ssl_cert = /etc/telegraf/cert.pem
# ssl_key = /etc/telegraf/key.pem
## Use SSL but skip chain & host verification
# insecure_skip_verify = false
```

Listing 6.4: OpenTOSCA Telegraf Konfiguration Tomcat (Auszug)

Die Konfigurationsdatei beinhaltet ausschließlich das Input Plugin Tomcat. Über die Variablen ["${TOMCAT_URL}"], ["${TOMCAT_USER}"] und ["${TOMCAT_PW}"] werden die Adresse, der Benutzername und das Passwort konfiguriert[Thea]. Zusätzlich können ein Timeout und Verzeichnisse mit optionalen SSL Zertifikaten angegeben werden. Bei der Implementierung des Agenten über eine zusätzliche Monitoringmethode wird kein neuer Node Type erstellt. Es werden nur bestehende Node Types um eine zusätzliche Monitoringmethode erweitert. Beispielsweise wurde das Service Template Ubuntu_On_OpenStack aus dem internen GitHub Repository[7] der Universität Stuttgart erweitert.

[7] https://github.com/OpenTOSCA

6.3 Visualisierung

In Kapitel 5.3 wurden zwei verschiedene Methoden (lose und enge Kopplung) zur Integration der Visualisierung in OpenTOSCA vorgestellt. Da bei einer losen Kopplung definierte Schnittstellen verwendet werden, können beide Komponenten (OpenTOSCA und TICK-Stack) unabhängig voneinander weiterentwickelt und ausgetauscht werden. Aus diesem Grund wird im Folgenden die lose gekoppelte Integration implementiert. Die OpenTOSCA Container REST API[8] wird verwendet, um dem Logsystem weitere Einträge hinzuzufügen. Folgende Zeile beschreibt den Aufbau der Adresse für die HTTP-Anfrage um neue Logeinträge hinzuzufügen.

```
/csars/{csar}/servicetemplates/{servicetemplate}/buildplans/{plan}/instances/
{instance}/logs
```

Listing 6.5: OpenTOSCA Container REST API Adressaufbau

Die in geschweiften Klammern enthaltenen Parameter müssen dabei durch die jeweiligen Werte ersetzt werden. Beispielsweise erzeugt folgender cURL Befehl einen neuen Log-Eintrag mit dem Wert CPU: 23.5% - RAM: 11.9%.

```
curl -X POST -H 'Content-Type: application/json' -i
    'http://192.168.209.131:1337/csars/Ubuntu_On_OpenStack.csar/servicetemplates/
\%257Bhttp:\%252F\%252Fopentosca.org\%252Fservicetemplates\%
257DUbuntu_On_OpenStack/buildplans/Ubuntu_On_OpenStack_buildPlan/
instances/1525954091448-0/logs' --data '{ "log_entry": "CPU: 23.5\% - RAM: 11.9\%" }'
```

Listing 6.6: OpenTOSCA Container REST API curl Befehl

Für den Aufruf der OpenTOSCA Container REST API wird die TICK-Komponente Kapacitor verwendet. Dieser ruft wahlweise, in festen Intervallen oder nur bei Überschreitung bestimmter Grenzwerte, die API auf und schreibt so die aktuellen Monitoringdaten in das Logsystem des OpenTOSCA Containers. Alternativ kann das Logsystem auch verwendet werden, um eine direkte Verlinkung des Chronografen zu erreichen. Dies ist möglich, da der Chronograf die Übergabe von Suchparametern per HTTP GET-Anfrage unterstützt. Folgende URL ermöglicht beispielsweise eine Suche nach der Prozesorauslastung der Instanz vm_test innerhalb der letzten sieben Tage.

```
http://192.168.209.131:8888/sources/1/chronograf/data-explorer?query=SELECT
    mean("usage_guest") AS "mean_usage_guest", mean("usage_system") AS
    "mean_usage_system" FROM "telegraf"."autogen"."cpu" WHERE time > now() - 7d AND
    "host"='vm_test' GROUP BY time(10s) FILL(null)
```

Listing 6.7: Chronograf Beispieladresse mit Parameterübergabe

Somit muss nur einmalig der Link zur TICK-Komponente Chronograf generiert und anschließend mithilfe der OpenTOSCA Container REST API der Log-Tabelle hinzugefügt werden.

[8] https://github.com/OpenTOSCA/container/tree/master/org.opentosca.container.api

7 Evaluation

In Kapitel 6 wurden die Integration des TICK-Stacks in das OpenTOSCA Ökosystem beschrieben. Kapitel 7.1 wird diese Implementierung evaluieren. Kapitel 7.2 beschreibt und evaluiert die Implementierung des TICK-Stacks in den bestehenden Projekt-Testaufbau der Arena 2036.

7.1 OpenTOSCA

Für eine abschließende Evaluation des in OpenTOSCA implementierten TICK-Stacks, wurden die OpenTOSCA Komponenten als Docker Container installiert. Dafür wurde die CSAR-Datei Ubuntu_On_OpenStack aus dem internen GitHub Repository[1] der Universität Stuttgart in OpenTOSCA importiert. Über das Modellierungstool Winery wurde dem Service Template, wie in Kapitel 6.2 beschrieben, die Monitoringmethode Agent hinzugefügt und anschließend der Plan Builder angepasst. Die Monitoring-Service-Komponenten wurden, wie in Kapitel 6.1 beschrieben, als Monitoring-as-a-Service provisioniert. Im letzten Schritt wurde über die OpenTOSCA GUI Vinothek die überarbeitete CSAR-Datei provisioniert. Nachdem das Betriebssystem Ubuntu 16.04 auf der neu provisionierten virtuellen Maschine installiert wurde, erfolgte die Installation und Konfiguration über das Bash-Skript. Telegraf sammelt anschließend Informationen über die Systemauslastung und sendet die gesammelten Daten an die Monitoring-Service-Komponente InfluxDB. Der Zeitraum zwischen dem Provisionieren der neuen virtuellen Maschine und dem Empfang erster Daten zur Systemauslastung in der Monitoring-Service-Komponenten Chronograf betrug dabei weniger als eine Minute. Somit wurde das Monitoringsystem TICK-Stack erfolgreich in das OpenTOSCA Ökosystem implementiert.

7.2 Arena 2036

Abbildung 7.1 zeigt die Architektur des Projekt-Testaufbaus der Arena 2036. Die in Kapitel 3.3 vorgestellte Architektur wurde um den TICK-Stack als Monitoringsystem erweitert. Um eine Trennung zwischen Produktivsystem und Monitoringssystem zu gewährleisten, wurde auf dem Server Edge eine weitere virtuelle Maschine (VM_Monitor) erstellt und darauf die TICK-Stack-Komponenten installiert. Zur Prüfung der Erreichbarkeit der Komponenten, wird das Input Plugin Ping des Agenten Telegraf verwendet. Der Agent auf der virtuellen Maschine VM_Monitor ruft regelmäßig (Intervall: zehn Sekunden) die IP-Adresse des Active Assist Embedded Systems, Nexo Embedded Systems und APAS Embedded PC auf und leitet die Daten an die Datenbank InfluxDB. Außerdem wird regelmäßig (Intervall: zehn Sekunden) die Website www.bosch.com aufgerufen, um die Verfügbarkeit der Internetverbindung zu prüfen.

[1] https://github.com/OpenTOSCA/tosca-definitions-internal

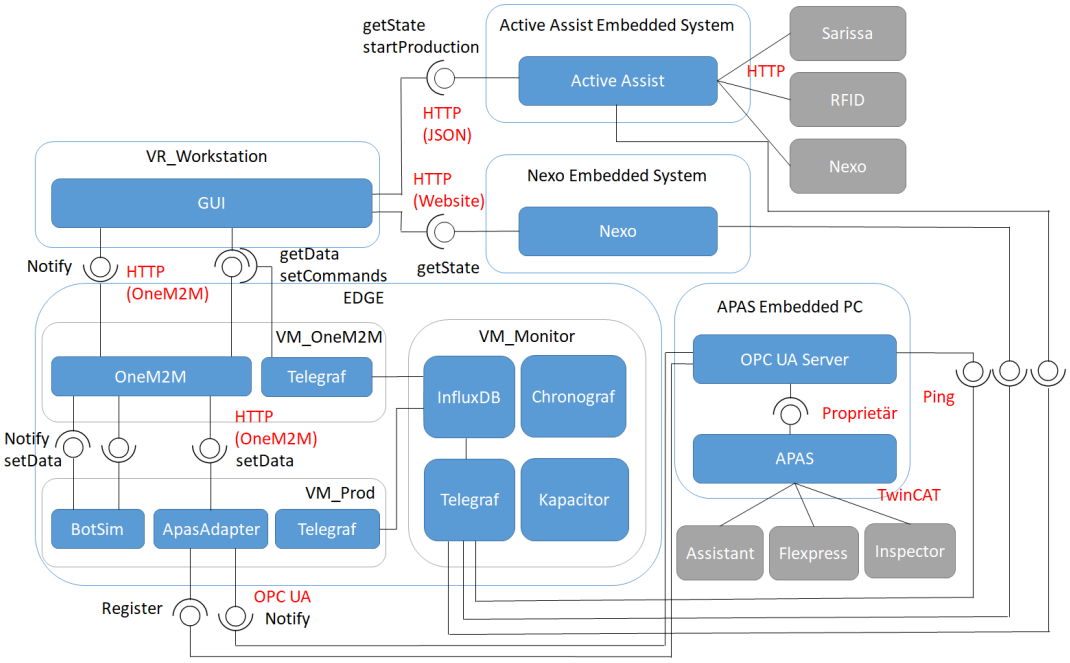

Abbildung 7.1: Architektur Arena 2036 mit Monitoring

Zur Erfassung der Logdateien und Prüfung des Service BotSim und ApasAdapter wurde auf beiden virtuellen Maschinen (VM_OneM2M und VM_Prod) ebenfalls der Agent Telegraf installiert. Auf der virtuellen Maschine VM_Prod überwacht der Agent regelmäßig die Services BotSim und ApasAdapter und liest die Logdateien aus. Die gesammelten Daten werden an die Datenbank InfluxDB gesendet. Der Agent auf der virtuellen Maschine VM_OneM2M ruft die HTTP-Schnittstelle des Services OneM2M auf, um Informationen, wie die aktuelle Position des Roboterarmes, auszulesen. Außerdem werden ebenfalls die Logdateien ausgelesen und an die Datenbank weitergeleitet. Die TICK-Stack-Komponente Chronograf visualisiert die gesammelten Daten und stellt diese dem Benutzer zur Verfügung. Das Benachrichtigungssystem Kapacitor wird genutzt, um bei Änderungen direkt den betreffenden Mitarbeiter zu informieren. Im Folgenden wird das Monitoringsystem anhand der verwendeten Komponenten für das Beispielszenario in der Arena 2036 beschrieben.

7.2.1 Telegraf

Zur Datenerfassung wurde der Agent Telegraf auf drei virtuellen Maschinen (VM_Monitor, VM_OneM2M und VM_Prod) installiert. Der Agent auf der virtuellen Maschine VM_Monitor nutzt das Input Plugin Ping, um die Verfügbarkeit der vier Komponenten (Active Assist Embedded System, Nexo Embedded System, APAS Embedded PC und bosch.com) zu überwachen. Die Konfiguration erfolgt durch folgende Codezeilen. Die Zeile URL definiert dabei die zu überprüfende Adresse, in diesem Fall die IP-Adresse des Active Assist Embedded Systems.

```
[[inputs.ping]]
 urls = ["192.168.100.160"]
 ...
```

Listing 7.1: Arena 2036 Konfiguration Input Plugin Ping (Auszug)

Der Agent Telegraf auf der virtuellen Maschine VM_Prod erfasst über das Input Plugin Procstat[2] die Prozessorauslastung der Services Botsim und Apasadapter. Folgende Codezeilen konfigurieren das Input Plugin.

```
[[inputs.procstat]]
 exe = "botsim|.*"
 fieldpass = ["cpu_usage"]

[[inputs.procstat]]
 exe = "apasadapter|.*"
 fieldpass = ["cpu_usage"]
```

Listing 7.2: Arena 2036 Konfiguration Input Plugin Procstat

Die aktuelle Position des mobilen Roboterarm Bosch APAS kann über zwei Methoden erfasst werden. Entweder wird regelmäßig die Datei Syslog auf der virtuellen Maschine VM_Prod ausgelesen oder die Daten werden über die HTTP-Schnittstelle des Services OneM2M abgerufen. Der Service ApasAdapter auf der virtuellen Maschine VM_Prod schreibt Informationen, wie die aktuelle Position, in festen Intervallen in die Syslog Datei. Der Agent auf der virtuellen Maschine VM_Prod nutzt das Input Plugin Logparser [3] um diese Daten auszulesen. Das Filtern und Zuordnen der gesammelten Daten zu den vorgesehenen Feldern in InfluxDB erfolgt über den Grok Parser. Folgendes Input Plugin liest die Syslog Datei regelmäßig aus und weist den Feldern (date, level, typ, theta[4], x und y) in der Datenbank die ausgelesenen Werte zu. Diese Werte werden für die Visualisierung der aktuellen Position des APAS benötigt.

```
[[inputs.logparser]]
   files = ["/var/log/syslog"]
   ...
   [inputs.logparser.grok]
      patterns = ['%{DATESTAMP:date} - botsim - %{LOGLEVEL:level} -
          %{WORD:typ}:{"theta":%{BASE10NUM:theta}, "x": %{BASE10NUM:x}, "y":
          %{BASE10NUM:y}']
```

Listing 7.3: Arena 2036 Konfiguration Input Plugin Logparser (Auszug)

Alternativ kann die Postion des Roboterarms auch per HTTP-Schnittstelle abgerufen werden. Die virtuelle Maschine VM_OneM2M bietet dafür die OneM2M HTTP-Schnittstelle an. Mithilfe des Telegraf Input Plugin httpjson[5] wird die HTTP-Schnittstelle aufgerufen. Folgender Ausschnitt zeigt den Rückgabewert der OneM2M HTTP-Schnittstelle.

[2] https://github.com/influxdata/telegraf/tree/master/plugins/inputs/procstat
[3] https://github.com/influxdata/telegraf/tree/master/plugins/inputs/logparser
[4] Verdrehwinkel des Roboterarms
[5] https://github.com/influxdata/telegraf/tree/master/plugins/inputs/httpjson

```
{
  "m2m:cin" : {
    "rn" : "pose_17358",
    "ty" : 4,
    "ri" : "/in-cse/cin-986034404",
    "pi" : "/in-cse/cnt-536420046",
    "ct" : "20180502T094239",
    "lt" : "20180502T094239",
    "st" : 0,
    "cnf" : "text/plain:0",
    "cs" : 28,
    "con" : "{\"theta\": 0, \"x\": 7, \"y\": 2}"
  }
}
```

Listing 7.4: Arena 2036 APAS JSON-Objekt

Strings können durch das Input Plugin httpjson aber nur geparst werden, wenn sich diese direkt im Objekt befinden und nicht weiter verschachtelt sind. Für eine Auswertung der Position sind vor allem X, Y und der Theta Wert wichtig. Da diese Werte als String zurückgegeben werden und der Agent Telegraf keinen geeigneten Parser bietet, muss hierfür eine Middleware eingesetzt werden. Im Rahmen dieser Arbeit wurde ein in NodeJS[6] entwickelter Parser erstellt. Dieser liest die Positionsdaten des mobilen Roboterarms APAS über die HTTP-Schnittstelle des Service OneM2M aus. Anschließend wird der empfangene JSON-String geparst und über einen Webserver zur Verfügung gestellt. Den durch die Middleware transformierten JSON-String kann der Agent Telegraf anschließend auslesen und die Positionsdaten an die Datenbank senden. Folgende Codezeilen beschreiben die in NodeJS entwickelte Middelware:

```
#!/usr/bin/env nodejs

var regex = /\\"theta\\": .*[0-9]+.[0-9]*, \\"y\\": .*[0-9]+.[0-9]*, \\"x\\":
    .*[0-9]+.[0-9]*/g
var result = '-';

var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});

const request = require('request');
const options = {
   url: 'http://192.168.100.69:8080/ARENA_CSE/ActiveShuttle/
   active-shuttle/pose/la',
   method: 'GET',
   headers: {
      'Content-Type': 'application/vnd.onem2m-res+json',
      'X-M2M-Origin': 'admin:admin'
   }
};
```

[6] https://nodejs.org

```
request(options, function(err, res, body) {
console.log(body);
let jsonneu = body.match(regex);
result = jsonneu[0].replace(/\\/gi, "");
console.log('{' + result);

});

  res.write('{' + result);
  res.end();
}).listen(8889);
```

Listing 7.5: Arena 2036 NodeJS JSON Parser

7.2.2 Chronograf

Die TICK-Stack-Komponente Chronograf fungiert als Schnittstelle zwischen dem Benutzer und der Datenbank InfluxDB. Die Visualisierung der gesammelten Daten des Arena 2036 Projekt-Testaufbaus erfolgte über zwei Dashboards (Infrastruktur-Dashboard und APAS-Dashboard). Abbildung 7.2 zeigt das Infrastruktur-Dashboard der Arena 2036. Das Dashboard visualisiert die Prozessorauslastung der drei Hostsysteme (Diagramm: CPU; Zeile 1 Spalte 1) und die Prozessorauslastung der Services Botsim und Apasadapter auf der virtuellen Maschine VM_Prod (Diagramm: CPU (Botsim, Apasadpater); Zeile 1 Spalte 2). Zusätzlich wird die Belegung des Arbeitsspeichers der drei Hostsysteme (Diagramm: RAM; Zeile 2 Spalte 1) und die Antwortzeit der Internetseite bosch.com (Diagramm: Ping: bosch.com; Zeile 3 Spalte 1) und der drei anderen Komponenten (Nexo, Active Assist und APAS) in Millisekunden dargestellt.

Abbildung 7.2: Chronograf Infrastruktur-Dashboard Arena 2036 (30 Tage)

Abbildung 7.3 zeigt das APAS-Dashboard. Dieses visualisiert die aktuelle Position (X, Y und Theta) des mobilen Roboterarms Bosch APAS. Die Positionsangaben werden alle fünf Sekunden aus der Datenbank ausgelesen und aktualisiert. In einer zukünftigen Arbeit könnten weitere Darstellungsformen, wie zum Beispiel in Form einer Heatmap, hinzugefügt werden. Eine Heatmap könnte auch die Position über einen längeren Zeitbereich visualisieren.

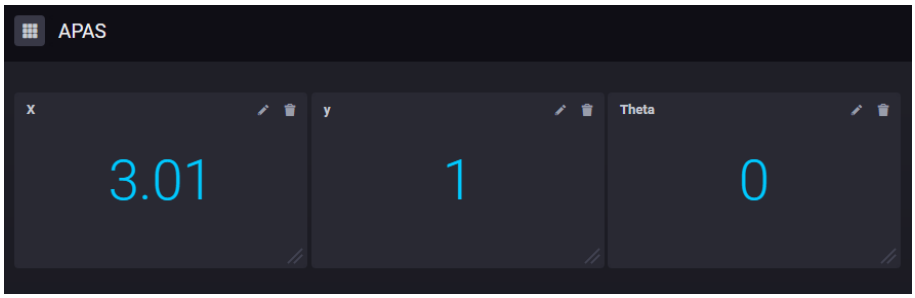

Abbildung 7.3: Chronograf APAS-Dashboard Arena 2036

7.2.3 InfluxDB

Die Datenbank InfluxDB wird zur Speicherung der gesammelten Daten verwendet. Während der Implementierung des TICK-Stacks in den Projekt-Testaufbau, wurde ein ungewöhnliches Verhalten der Datenbank beobachtet. Das Infrastuktur-Dashboard in Abbildung 7.2 zeigt, dass der Arbeitsspeicher der virtuellen Maschine VM_Monitor im Laufe von etwa drei Tagen bis zu 80 Prozent belegt und anschließend wieder freigegeben wurde. Eine detaillierte Analyse ergab, dass die Datenbank InfluxDB für den hohen Arbeitsspeicherverbrauch verantwortlich war. In Anbetracht der Tatsache, dass die virtuelle Maschine VM_Monitor 32 Gigabyte Arbeitsspeicher zur Verfügung hat und insgesamt nur drei Systeme überwacht wurden, ist dies eine enorm hohe Belegung. Da der Arbeitsspeicher ohne Eingreifen wieder freigegeben wurde, beeinflusste dies den Normalbetrieb nicht. In einer weiterführenden Arbeit könnte allerdings analysiert werden, wie sich dies bei einer Verringerung des verfügbaren Arbeitsspeichers oder einer Erhöhung der zu überwachenden Systeme auswirkt.

7.2.4 Grafana

Als Alternative zur TICK-Stack-Komponente Chronograf kann das Tool Grafana verwendet werden. Anders als Chronograf, können in Grafana auch Tabellen mit Logeinträgen dargestellt werden. Zusätzlich können die Tabellen mithilfe von Filterfunktionen dynamisch angepasst werden. Dies ermöglicht eine Darstellung bestimmter Einträge. Abbildung 7.4 fasst beispielsweise Logeinträge von drei verschiedenen virtuellen Maschinen (cr-ubuntu, cr-ubuntu-copy und om2m) in einer Tabelle zusammen. Über das rot markierte Auswahlfeld Host, können einzelne virtuelle Maschinen ausgewählt werden. Alternativ kann die Auswahl der jeweiligen Filter auch als Parameter per HTTP GET-Anfrage erfolgen[Grab].

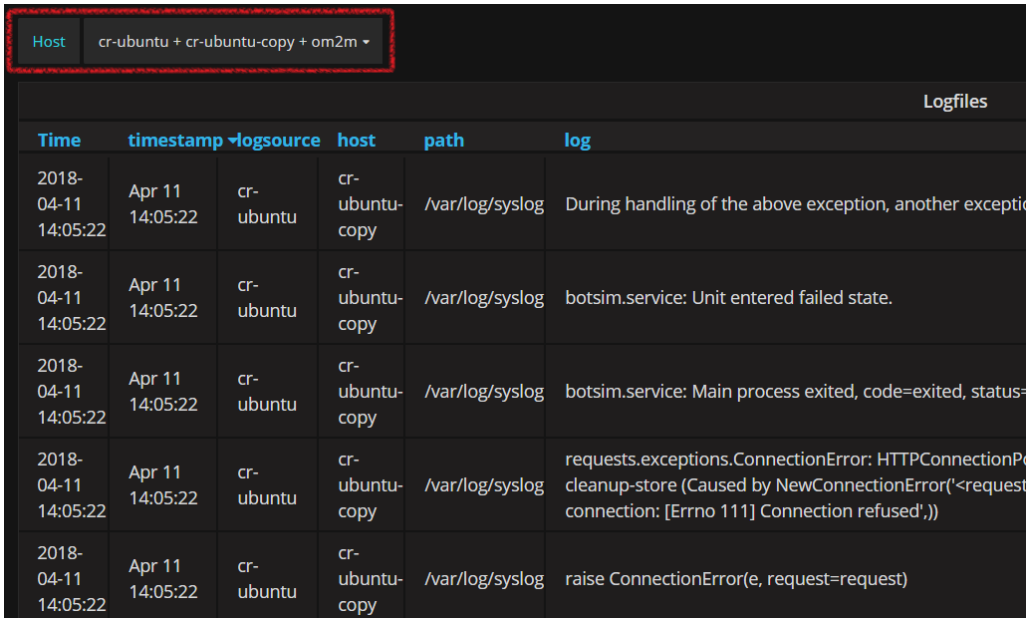

Abbildung 7.4: Grafana Dashboard mit Auswahlfeld

7.2.5 Kapacitor

Das Benachrichtigungssystem Kapacitor überwacht die gesammelten Daten anhand festgelegter Grenzwerte und startet ein Event bei Überschreitung. Abbildung 7.5 zeigt ein mögliches Einsatzgebiet für das Benachrichtigungssystem. Die Arbeitsspeicherbelegung der virtuellen Maschine in der Arena 2036 mit den Monitoring-Service-Komponenten steigt konstant an. Bei einer Arbeitsspeicherbelegung von etwa 80 Prozent, aktiviert sich automatisch ein Garbage Collector, der den belegten Arbeitsspeicher wieder freigibt. In diesem Beispiel könnte der Grenzwert für einen Alarm auf 90 Prozent gesetzt werden. Falls das automatische Freigeben des Arbeitsspeichers ausfällt, wird ein betreuender Mitarbeiter per E-Mail informiert. Dieser kann so einen möglichen Ausfall des kompletten Systems verhindern. Zukünftig kann das Benachrichtigungssystem weiter ausgebaut werden, um bestimmte Probleme automatisiert zu lösen. Eine Möglichkeit wäre die Überwachung des Bosch APAS. Hat dieser längere Zeit keinen aktiven Job, obwohl die JobList Einträge hat, könnte der ApasAdapter Service auf der virtuellen Maschine VM_Prod per Skript neugestartet werden. Dies fällt in den Bereich des Self-Healing und könnte in einer zukünftigen Arbeit thematisiert werden.

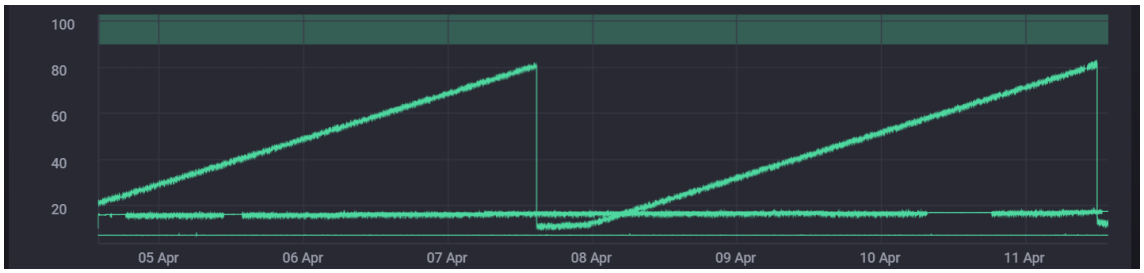

Abbildung 7.5: Kapacitor Alert Grenzwert konfigurieren

8 Zusammenfassung und Ausblick

Mit der vorliegenden Arbeit wurde ein geeignetes Monitoringsystem in das bestehende OpenTOSCA Ökosystem integriert. Um den Nutzen eines Monitoringsystems zu demonstrieren, ist, in Kooperation mit der Robert Bosch GmbH, ein bestehender Projekt-Testaufbau in der Arena 2036 um ein zusätzliches Monitoringsystem erweitert worden.

Als Grundlage führt Kapitel 2 den OASIS Standard TOSCA und die von der Universität Stuttgart entwickelte Laufzeitumgebung für TOSCA-basierende Cloud-Anwendungen auf. Zudem wurde das Vorgehen für die Einführung eines Monitoringsystems und der Unterschied zwischen agentbased und agentless, sowie zentralen und dezentralen Monitoringsystemen beschrieben.

Daraufhin behandelt Kapitel 3 die Anforderungen an ein Monitoringsystem, um dieses erfolgreich in das OpenTOSCA Ökosystem integrieren zu können. Zusätzlich wurden weitere Anforderungen aufgestellt, die sich speziell an ein Monitoringsystem für den Projekt-Testaufbau in der Arena 2036 richten.

Die Evaluation der Monitoringsysteme hat anhand der aufgestellten Anforderungen in Kapitel 4 ergeben, dass drei der vier Monitoringsysteme (Splunk, ELK-Stack und TICK-Stack) alle funktionalen Anforderungen erfüllen. Nach zusätzlicher Betrachtung der nicht-funktionalen Anforderungen, sowie der Anforderungen des Arena 2036 Projekt-Testaufbaus, stellte sich der TICK-Stack als geeignetes Monitoringsystem heraus.

Im Weiteren wurden zur Ermittlung eines geeigneten Monitoringsystems in Kapitel 5 verschiedene Konzepte für die Integration des TICK-Stacks in das OpenTOSCA Ökosystem vorgestellt. Dabei werden die Vor- und Nachteile einer engen und losen Kopplung der einzelnen Monitoring-Service-Komponenten aufgezeigt. Außerdem wird der Unterschied zwischen der Implementierung des Agenten als Node Type und als Monitoringmethode verdeutlicht. Die Integration der Visualisierung wurde anhand der Konzepte einer engen und losen Kopplung beschrieben.

Kapitel 6 erläutert im Anschluss die Implementierung der in Kapitel 5 vorgestellten Konzepte. Dabei wurden bereits vorhandene Schnittstellen verwendet, um eine lose Kopplung der Komponenten zu erzielen.

Wie die abschließende Evaluation des OpenTOSCA Ökosystems in Kapitel 6 aufzeigt, ist der TICK-Stack erfolgreich in das OpenTOSCA Ökosystem integrierbar. Monitoringdaten, wie die Systemauslastung, der über OpenTOSCA provisionierten Instanz, werden über den Agenten gesammelt und in der InfluxDB gespeichert. Eine Visualisierung der gesammelten Daten ist über die TICK-Stack Komponente Chronograf und über das integrierte Logsystem der OpenTOSCA Vinothek möglich.

Die Integration des Monitoringsystems in einen laufenden Projekt-Testaufbau der Robert Bosch GmbH offenbarte dagegen Schwierigkeiten bei der Umsetzung. Neben dem Problem des Parsens von verschachtelten JSON Strings, zeigte sich das Phänomen des regelmäßigen Anstiegs und

anschließender Freigabe des belegten Arbeitsspeichers durch die InfluxDB. Um die JSON Strings in ein lesbares Format zu konvertieren, wurde eine Middleware entwickelt. Bezüglich des Anstiegs des belegten Arbeitsspeichers konnte aufgezeigt werden, dass dieser bei etwa 80 Prozent wieder freigegeben wurde und dadurch keinen Einfluss auf den laufenden Betrieb nimmt. Abgesehen von den oben beschriebenen Schwierigkeiten, wurde der TICK-Stack erfolgreich in den bestehenden Projekt-Testaufbau der Robert Bosch GmbH integriert.

Ausblick

Ausgehend von der Integration eines Monitoringssystems in das OpenTOSCA Ökosystem, eröffnen sich Möglichkeiten zur weiteren Verwertung der gesammelten Daten in allen Wirtschaftssektoren. Beispielhaft ließe sich hierbei auf das Konzept AIOps verweisen. Der Begriff AIOps steht als Abkürzung für Artificial Intelligence for IT Operations und beschreibt das Automatisieren und Optimieren von Abläufen im IT-Betrieb durch den Einsatz von Machine Learning. Dabei können große Datenmengen aus unterschiedlichen Quellen kontinuierlich analysiert und gegebenenfalls in Echtzeit optimiert werden[JM92].

Ein weiterer produktiver Einsatz betrifft das Konzept des Self-Healing aus dem Industrie 4.0 Kontext. Self-Healing-Systeme verfügen über Mechanismen der automatisierten Fehlererkennung und -behebung, ohne den laufenden Betrieb zu beeinflussen[GSRU07]. Dies könnte ausgehend von dem in Kapitel 4.3.3 vorgestellten Benachrichtigungssystem entwickelt werden. Bei der Überschreitung eines festgelegten Grenzwertes, beispielsweise die Arbeitsspeicherbelegung eines Systems, wird durch das Benachrichtigungssystem automatisch ein Skript ausgeführt, welches das System neu startet. Dies könnte einen Systemausfall verhindern, der durch die hohe Arbeitsspeicherbelegung ausgelöst wird.

In Kapitel 2.3 wurden zwei Verfahren (agent-based und agentless) zur Datenerfassung vorgestellt. In einer weiterführenden Arbeit könnte das Thema Hybrid-Monitoring als Kombination beider Verfahren evaluiert werden[Avi].

Literaturverzeichnis

[ABDP13] G. Aceto, A. Botta, W. de Donato, A. Pescapè. „Cloud monitoring: A survey“. In: *Computer Networks* 57.9 (2013), S. 2093–2115. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2013.04.001. URL: http://www.sciencedirect.com/science/article/pii/S1389128613001084 (zitiert auf S. 27, 28).

[Arb17] G. Arbezzano. „Infrastructure Monitoring with TICK Stack“. In: (2017). URL: https://blog.codeship.com/infrastructure-monitoring-with-tick-stack/ (zitiert auf S. 54).

[AREa] ARENA2036 e.V. *Forschungscampus*. Besucht: 04.04.2018. URL: https://www.arena2036.de/de/arena2036/forschungscampus (zitiert auf S. 14).

[AREb] ARENA2036 e.V. *Über Bosch*. Besucht: 04.04.2018. URL: https://www.arena2036.de/de/134-deutsch/partner/partnersteckbrief/315-steckbrief-bosch-de (zitiert auf S. 14).

[Avi] Y. Avital. *Which Way to Go – Agent, Agentless or Hybrid?* Besucht: 08.04.2018. URL: https://www.controlup.com/blog/agent-agentless/ (zitiert auf S. 21–23, 84).

[Ban10] S. Banon. *The Future of Compass Elasticsearch*. Besucht: 31.03.2018. 2010. URL: http://thedudeabides.com/articles/the_future_of_compass (zitiert auf S. 42).

[BBKL14] T. Binz, U. Breitenbücher, O. Kopp, F. Leymann. In: *Advanced Web Services*. New York: Springer, Jan. 2014. Kap. TOSCA: Portable Automated Deployment and Management of Cloud Applications, S. 527–549. ISBN: 978-1-4614-7534-7. DOI: 10.1007/978-1-4614-7535-4_22 (zitiert auf S. 13, 15, 16).

[BCK+16] U. Breitenbuecher, K. K. Christian Endres, O. Kopp, F. Leymann, S. Wagner, J. Wettinger, M. Zimmermann. *The OpenTOSCA Ecosystem – Concepts Tools*. 2016 (zitiert auf S. 16–19).

[Bosa] Bosch Rexroth AG. *Akkuschrauber Nexo*. Besucht: 25.03.2018. URL: https://www.boschrexroth.com/de/de/produkte/produktneuheiten/elektrische-antriebe-und-steuerungen/nexo-1 (zitiert auf S. 29).

[Bosb] Bosch Rexroth AG. *Werkerassistenzsystem ActiveAssist*. Besucht: 25.03.2018. URL: https://www.boschrexroth.com/de/de/produkte/produktneuheiten/montagetechnik/activeassist (zitiert auf S. 29).

[CC04] A. P. Chan, A. P. Chan. „Key performance indicators for measuring construction success“. In: *Benchmarking: An International Journal* 11.2 (2004), S. 203–221. DOI: 10.1108/14635770410532624. URL: https://doi.org/10.1108/14635770410532624 (zitiert auf S. 38).

[Deu17] Deutsche Telekom AG. „oneM2M – standardisiert den Machine-to-Machine-Markt“. In: (2017). URL: https://m2m.telekom.com/de/m2m-blog/article/onem2m-standardizes-the-machine-to-machine-market/ (zitiert auf S. 30).

[DGH+87] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, D. Terry. „Epidemic Algorithms for Replicated Database Maintenance“. In: *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing*. PODC '87. Vancouver, British Columbia, Canada: ACM, 1987, S. 1–12. ISBN: 0-89791-239-X. DOI: 10.1145/41840.41841. URL: http://doi.acm.org/10.1145/41840.41841 (zitiert auf S. 55).

[DKC05] J. Dollimore, T. Kindberg, G. Coulouris. *Distributed Systems: Concepts and Design (4th Edition) (International Computer Science Series)*. Addison Wesley, may 2005. ISBN: 0321263545. URL: http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20&path=ASIN/0321263545 (zitiert auf S. 55).

[Elaa] Elastic. *Auditbeat Reference [6.2] » Auditbeat overview*. Besucht: 02.04.2018. URL: https://www.elastic.co/guide/en/beats/auditbeat/current/auditbeat-overview.html (zitiert auf S. 45).

[Elab] Elastic. *Beats Platform Reference*. Besucht: 29.03.2018. URL: https://www.elastic.co/guide/en/beats/libbeat/current/beats-reference.html (zitiert auf S. 42).

[Elac] Elastic. *Beats Platform Reference [6.2]*. Besucht: 31.03.2018. URL: https://www.elastic.co/guide/en/beats/libbeat/current/beats-reference.html (zitiert auf S. 43).

[Elad] Elastic. *Beats Platform Reference [6.2] » Getting started with Beats and the Elastic Stack*. Besucht: 03.04.2018. URL: https://www.elastic.co/guide/en/beats/libbeat/current/getting-started.html (zitiert auf S. 46).

[Elae] Elastic. *Elasticsearch Reference [6.2]*. Besucht: 31.03.2018. URL: https://www.elastic.co/guide/en/elasticsearch/reference/current/getting-started.html (zitiert auf S. 43).

[Elaf] Elastic. *Elasticsearch Reference [6.2] » Getting Started » Basic Concepts*. Besucht: 03.04.2018. URL: https://www.elastic.co/guide/en/elasticsearch/reference/current/_basic_concepts.html (zitiert auf S. 46).

[Elag] Elastic. *Elasticsearch: The Definitive Guide [master] » Getting Started » Life Inside a Cluster » Add an Index*. Besucht: 07.04.2018. URL: https://www.elastic.co/guide/en/elasticsearch/guide/master/_add_an_index.html (zitiert auf S. 47).

[Elah] Elastic. *Elasticsearch: The Definitive Guide [master] » Getting Started » Life Inside a Cluster » Add Failover*. Besucht: 07.04.2018. URL: https://www.elastic.co/guide/en/elasticsearch/guide/master/_add_failover.html (zitiert auf S. 47, 48).

[Elai] Elastic. *Elasticsearch: The Definitive Guide [master] » Getting Started » Life Inside a Cluster » An Empty Cluster*. Besucht: 07.04.2018. URL: https://www.elastic.co/guide/en/elasticsearch/guide/master/_an_empty_cluster.html (zitiert auf S. 47).

[Elaj] Elastic. *elasticsearch/LICENSE.txt*. Besucht: 31.03.2018. URL: https://github.com/elastic/elasticsearch/blob/master/LICENSE.txt (zitiert auf S. 47).

[Elak] Elastic. *Filebeat Reference [6.2] » Configuring Filebeat » Load balance the output hosts*. Besucht: 07.04.2018. URL: https://www.elastic.co/guide/en/beats/filebeat/6.2/load-balancing.html (zitiert auf S. 47).

[Elal] Elastic. *Filebeat Reference [6.2] » Getting Started With Filebeat*. Besucht: 02.04.2018. URL: https://www.elastic.co/guide/en/beats/filebeat/6.2/filebeat-installation.html (zitiert auf S. 43, 44, 48).

[Elam] Elastic. *Heartbeat Reference [6.2] » Heartbeat overview*. Besucht: 02.04.2018. URL: https://www.elastic.co/guide/en/beats/heartbeat/current/heartbeat-overview.html (zitiert auf S. 45).

[Elan] Elastic. *Kibana User Guide [6.2]*. Besucht: 31.03.2018. URL: https://www.elastic.co/guide/en/kibana/current/introduction.html (zitiert auf S. 43).

[Elao] Elastic. *Logstash Reference [6.2]*. Besucht: 31.03.2018. URL: https://www.elastic.co/guide/en/logstash/current/introduction.html (zitiert auf S. 43).

[Elap] Elastic. *Logstash Reference [6.2]*. Besucht: 01.04.2018. URL: https://www.elastic.co/guide/en/logstash/current/plugins-filters-grok.html (zitiert auf S. 50).

[Elaq] Elastic. *Logstash Reference [6.2] » Deploying and Scaling Logstash*. Besucht: 07.04.2018. URL: https://www.elastic.co/guide/en/logstash/current/deploying-and-scaling.html (zitiert auf S. 47).

[Elar] Elastic. *Logstash Reference [6.2] » Filter plugins » Aggregate filter plugin*. Besucht: 03.04.2018. URL: https://www.elastic.co/guide/en/logstash/6.2/plugins-filters-aggregate.html (zitiert auf S. 46).

[Elas] Elastic. *Logstash Reference [6.2] » Getting Started with Logstash*. Besucht: 02.04.2018. URL: https://www.elastic.co/guide/en/logstash/current/advanced-pipeline.html (zitiert auf S. 44).

[Elat] Elastic. *Machine Learning*. Besucht: 12.03.2018. URL: https://www.elastic.co/de/products/x-pack/machine-learning (zitiert auf S. 47).

[Elau] Elastic. *Metricbeat Reference [6.2] » Metricbeat overview*. Besucht: 02.04.2018. URL: https://www.elastic.co/guide/en/beats/metricbeat/current/metricbeat-overview.html (zitiert auf S. 43).

[Elav] Elastic. *Metricbeat Reference [6.2] » Modules*. Besucht: 02.04.2018. URL: https://www.elastic.co/guide/en/beats/metricbeat/current/metricbeat-modules.html (zitiert auf S. 43, 47).

[Elaw] Elastic. *Metricbeat Reference [6.2] » Modules » System module » System network metricset*. Besucht: 02.04.2018. URL: https://www.elastic.co/guide/en/beats/metricbeat/current/metricbeat-metricset-system-network.html (zitiert auf S. 43, 46).

[Elax] Elastic. *Metricbeat Reference [6.2] » Modules » vSphere module » vSphere host metricset*. Besucht: 03.04.2018. URL: https://www.elastic.co/guide/en/beats/metricbeat/current/metricbeat-metricset-vsphere-host.html (zitiert auf S. 46).

[Elay] Elastic. *Packetbeat Reference [6.2] » Configuring Packetbeat » Configure the internal queue*. Besucht: 03.04.2018. URL: https://www.elastic.co/guide/en/beats/packetbeat/6.2/configuring-internal-queue.html#_literal_events_literal (zitiert auf S. 46).

[Elaz] Elastic. *Packetbeat Reference [6.2] » Packetbeat overview*. Besucht: 02.04.2018. URL: https://www.elastic.co/guide/en/beats/packetbeat/current/packetbeat-overview.html (zitiert auf S. 45).

[Elaaa] Elastic. *Subskriptionen*. Besucht: 12.03.2018. URL: https://www.elastic.co/de/subscriptions (zitiert auf S. 47, 48).

[Elaab] Elastic. *Winlogbeat Reference [6.2] » Winlogbeat Overview*. Besucht: 02.04.2018. URL: https://www.elastic.co/guide/en/beats/winlogbeat/current/_winlogbeat_overview.html (zitiert auf S. 45).

[Elaac] Elastic. *X-Pack*. Besucht: 31.03.2018. URL: https://www.elastic.co/de/products/x-pack (zitiert auf S. 43).

[Elaad] Elastic. *X-Pack for the Elastic Stack [6.2] » Alerting on Cluster and Index Events » Actions*. Besucht: 02.04.2018. URL: https://www.elastic.co/guide/en/x-pack/6.2/actions.html (zitiert auf S. 44, 46).

[Elaae] Elastic. *X-Pack for the Elastic Stack [6.2] » Alerting on Cluster and Index Events » Getting Started with Watcher*. Besucht: 02.04.2018. URL: https://www.elastic.co/guide/en/x-pack/current/watcher-getting-started.html (zitiert auf S. 43, 44).

[FHRS07] T. Focke, W. Hasselbring, M. Rohr, J.-G. Schute. „Ein Vorgehensmodell für Performance-Monitoring von Informationssystemlandschaften." In: 27 (Jan. 2007), S. 26–31 (zitiert auf S. 20, 21).

[Fre17] Free Software Foundation, Inc. *Freie Softwarelizenzen (vereinbar mit GNU GPL)*. 2017. URL: http://www.gnu.org/licenses/license-list.html#apache2 (zitiert auf S. 28, 56).

[Ges] Gesellschaft für Informatik e.V. *IT-Compliance*. Besucht: 22.05.2018. URL: https://gi.de/informatiklexikon/it-compliance/ (zitiert auf S. 41).

[Gne14] M. Gneuss. *Industrie 4.0 - Die vierte industrielle Revolution*. http://www.gfos.com/fileadmin/news/PDF/Ref_Industrie40_HB.pdf. 2014 (zitiert auf S. 13).

[Graa] Grafana Labs. *Official community built plugins*. Besucht: 17.04.2018. URL: https://grafana.com/plugins?type=datasource (zitiert auf S. 54).

[Grab] Grafana Labs. *Variables*. Besucht: 19.05.2018. URL: http://docs.grafana.org/reference/templating/ (zitiert auf S. 80).

[GSRU07] D. Ghosh, R. Sharman, H. R. Rao, S. Upadhyaya. „Self-healing systems — survey and synthesis". In: *Decision Support Systems* 42.4 (2007). Decision Support Systems in Emerging Economies, S. 2164–2185. ISSN: 0167-9236. DOI: https://doi.org/10.1016/j.dss.2006.06.011. URL: http://www.sciencedirect.com/science/article/pii/S0167923606000807 (zitiert auf S. 84).

[Infa] InfluxData. *AlertNode*. Besucht: 29.03.2018. URL: https://docs.influxdata.com/kapacitor/v1.4/nodes/alert_node/ (zitiert auf S. 52, 54).

[Infb] InfluxData. *Clustering in InfluxDB Enterprise*. Besucht: 29.03.2018. URL: https://docs.influxdata.com/enterprise_influxdb/v1.5/concepts/clustering/ (zitiert auf S. 55).

[Infc] InfluxData. *Configuring Telegraf*. Besucht: 23.04.2018. URL: https://docs.influxdata.com/telegraf/v1.6/administration/configuration/ (zitiert auf S. 52, 71, 72).

[Infd] InfluxData. *Configuring Telegraf*. Besucht: 29.03.2018. URL: https://docs.influxdata.com/telegraf/v1.5/administration/configuration/ (zitiert auf S. 54).

[Infe] InfluxData. *Custom anomaly detection using Kapacitor*. Besucht: 29.03.2018. URL: https://docs.influxdata.com/kapacitor/v1.4/guides/anomaly_detection/ (zitiert auf S. 54).

[Inff] InfluxData. *Installing Telegraf*. Besucht: 02.04.2018. URL: https://docs.influxdata.com/telegraf/v1.5/introduction/installation/ (zitiert auf S. 56).

[Infg] InfluxData. *Kapacitor alerts overview*. Besucht: 29.03.2018. URL: https://docs.influxdata.com/kapacitor/v1.4/working/alerts/ (zitiert auf S. 54).

[Infh] InfluxData. *Kapacitor Enterprise 1.4 documentation*. Besucht: 29.03.2018. URL: https://docs.influxdata.com/enterprise_kapacitor/v1.4/ (zitiert auf S. 55).

[Infi] InfluxData. *Logparser Input Plugin*. Besucht: 01.04.2018. URL: https://github.com/influxdata/telegraf/tree/master/plugins/inputs/logparser (zitiert auf S. 51).

[Infj] InfluxData. *Open Source Time Series Platform*. Besucht: 12.03.2018. URL: https://www.influxdata.com/time-series-platform/ (zitiert auf S. 49).

[Infk] InfluxData. *Querying data with the HTTP API*. Besucht: 22.04.2018. URL: https://docs.influxdata.com/influxdb/v1.5/guides/querying_data/ (zitiert auf S. 67).

[Infl] InfluxData. *telegraf*. Besucht: 01.04.2018. URL: https://github.com/influxdata/telegraf/tree/master/plugins/inputs/ (zitiert auf S. 50).

[Infm] InfluxData. *Telegraf v1.5.3*. Besucht: 29.03.2018. URL: https://portal.influxdata.com/downloads (zitiert auf S. 53).

[Infn] InfluxData Inc. *chronograf/LICENSE*. Besucht: 13.03.2018. URL: https://github.com/influxdata/chronograf/blob/master/LICENSE (zitiert auf S. 56).

[Info] InfluxData Inc. *Getting started with Telegraf*. Besucht: 13.03.2018. URL: https://docs.influxdata.com/telegraf/v1.5/introduction/getting_started/ (zitiert auf S. 52, 53).

[Infp] InfluxData Inc. *InfluxDB 1.5 documentation*. Besucht: 14.03.2018. URL: https://docs.influxdata.com/influxdb/v1.5/ (zitiert auf S. 49).

[Infq] InfluxData Inc. *logstash-plugins/logstash-patterns-core*. Besucht: 13.03.2018. URL: https://github.com/logstash-plugins/logstash-patterns-core/blob/master/patterns/grok-patterns (zitiert auf S. 50, 51).

[Infr] InfluxData Inc. *Telegraf plugins*. Besucht: 13.03.2018. URL: https://docs.influxdata.com/telegraf/v1.5/plugins/ (zitiert auf S. 52).

[Infs] InfluxData Inc. *telegraf/LICENSE*. Besucht: 13.03.2018. URL: https://github.com/influxdata/telegraf/blob/master/LICENSE (zitiert auf S. 56).

[its] it-service.network. *IT-Monitoring für Unternehmen*. Besucht: 24.05.2018. URL: https://it-service.network/blog/2016/07/08/it-monitoring-fuer-unternehmen/ (zitiert auf S. 13).

[JM92] P. K. JAIN, C. T. MOSIER. „Artificial intelligence in flexible manufacturing systems“. In: *International Journal of Computer Integrated Manufacturing* 5.6 (1992), S. 378–384. DOI: 10.1080/09511929208944545. eprint: https://doi.org/10.1080/09511929208944545. URL: https://doi.org/10.1080/09511929208944545 (zitiert auf S. 84).

[Kép13] K. Képes. „Konzept und Implementierung einer Java-Komponente zur Generierung von WS-BPEL 2.0 BuildPlans für OpenTOSCA“. 2013. DOI: http://dx.doi.org/10.18419/opus-3114 (zitiert auf S. 64, 65).

[Lie] D. Liebhart. *Mit Big Data auf dem Weg ins Industrie-4.0-Zeitalter*. Besucht: 04.05.2018. URL: https://www.bigdata-insider.de/mit-big-data-auf-dem-weg-ins-industrie-40-zeitalter-a-550209/ (zitiert auf S. 13).

[Lit] S. Litke. *Dezentrale Überwachung*. Besucht: 12.05.2018. URL: https://www.crn.de/netzwerke-storage/artikel-84371.html (zitiert auf S. 23).

[Los] M. Loschwitz. *Skalierbares Monitoring mit Prometheus*. Besucht: 30.03.2018. URL: http://www.admin-magazin.de/Das-Heft/2017/06/Skalierbares-Monitoring-mit-Prometheus (zitiert auf S. 57, 58).

[Mic] Micro Focus. *Configuring Splunk for Integration*. Besucht: 02.04.2018. URL: https://www.netiq.com/documentation/secure-configuration-manager-7/user_guide_scm/data/t456bwg6ygou.html (zitiert auf S. 34).

[Nor14] I. O. für Normung. *ISO/IEC 25000:2014 Systems and software engineering – Systems and software Quality Requirements and Evaluation*. https://www.iso.org/standard/64764.html. Besucht: 03.02.2018. 14.03.2014 (zitiert auf S. 25).

[OPC14] OPC Foundation. „OPC UA als Wegbereiter der Industrie 4.0." In: (2014). URL: https://opcfoundation.org/wp-content/uploads/2014/03/OPC_UA_I_4.0_Wegbereiter_DE_v2.pdf (zitiert auf S. 30).

[Proa] Prometheus Authors. *Frequently Asked Questions*. Besucht: 30.03.2018. URL: https://prometheus.io/docs/introduction/faq/ (zitiert auf S. 58).

[Prob] Prometheus Authors. *Notification Template Reference*. Besucht: 31.03.2018. URL: https://prometheus.io/docs/alerting/notifications/ (zitiert auf S. 58).

[Proc] Prometheus Authors. *OVERVIEW - What is Prometheus?* Besucht: 18.03.2018. URL: https://prometheus.io/docs/introduction/overview/ (zitiert auf S. 57).

[Prod] Prometheus Authors. *Prometheus Pushgateway*. Besucht: 31.03.2018. URL: https://github.com/prometheus/pushgateway (zitiert auf S. 58).

[PW01] R. Plösch, R. Weinreich. „Ein agentenbasierter Ansatz für die Ferndiagnose und-überwachung von Automatisierungssystemen". In: *Wirtschaftsinformatik* 43.2 (Apr. 2001), S. 167–173. ISSN: 1861-8936. DOI: 10.1007/BF03250793. URL: https://doi.org/10.1007/BF03250793 (zitiert auf S. 21).

[RCR94] V. R. Basili, G. Caldiera, D. Rombach. „The goal question metric approach". In: 1 (Jan. 1994) (zitiert auf S. 20).

[Rob] Robert Bosch Manufacturing Solutions GmbH. *APAS Intelligente Systeme für die Mensch-Maschine-Kollaboration*. Besucht: 25.03.2018. URL: https://www.bosch-apas.com/media/de/apas/downloads/20170324_web_apas_broschure_de.pdf (zitiert auf S. 29).

[Roh] C. Rohmann. *Elasticsearch, Logstash Kibana*. Besucht: 31.03.2018. URL: http://www.linux-magazin.de/ausgaben/2016/02/elk-stack/ (zitiert auf S. 43).

[SCZ+16] W. Shi, J. Cao, Q. Zhang, Y. Li, L. Xu. „Edge Computing: Vision and Challenges". In: *IEEE Internet of Things Journal* 3.5 (Okt. 2016), S. 637–646. ISSN: 2327-4662. DOI: 10.1109/JIOT.2016.2579198 (zitiert auf S. 27).

[SMC74] W. P. Stevens, G. J. Myers, L. L. Constantine. „Structured design". In: *IBM Systems Journal* 13.2 (1974), S. 115–139. ISSN: 0018-8670. DOI: 10.1147/sj.132.0115 (zitiert auf S. 62).

[Sola] Solid IT GmbH. *DB-Engines Ranking von Suchmaschinen*. Besucht: 26.03.2018. URL: https://db-engines.com/de/ranking/suchmaschine (zitiert auf S. 33, 42).

[Solb] Solid IT GmbH. *DB-Engines Ranking von Time Series DBMS*. Besucht: 26.03.2018. URL: https://db-engines.com/de/ranking/time+series+dbms (zitiert auf S. 49).

[spi] spirent.com. *TCP Network Latency and Throughput*. Besucht: 27.03.2018. URL: https://www.spirentfederal.com/documents/tcp_network-latency-and-throughput_whitepaper.pdf (zitiert auf S. 37).

[Spla] Splunk Inc. *About real-time searches and reports*. Besucht: 27.03.2018. URL: http://docs.splunk.com/Documentation/Splunk/latest/Search/Aboutrealtimesearches (zitiert auf S. 37).

[Splb] Splunk Inc. *About search head clustering*. Besucht: 28.03.2018. URL: http://docs.splunk.com/Documentation/Splunk/7.0.3/DistSearch/AboutSHC (zitiert auf S. 40).

[Splc] Splunk Inc. *About the search language*. Besucht: 01.04.2018. URL: http://docs.splunk.com/Documentation/Splunk/7.0.3/Search/Aboutthesearchlanguage (zitiert auf S. 33, 35).

[Spld] Splunk Inc. *Apps für Cloud-Services*. Besucht: 15.03.2018. URL: https://www.splunk.com/de_de/cloud/apps-for-cloud-services.html (zitiert auf S. 41).

[Sple] Splunk Inc. *Apps und Add-Ons*. Besucht: 13.03.2018. URL: https://www.splunk.com/blog/2012/08/27/do-you-hadoop-how-splunk-can-help.html (zitiert auf S. 37).

[Splf] Splunk Inc. *Configure the universal forwarder*. Besucht: 26.03.2018. URL: http://docs.splunk.com/Documentation/Forwarder/7.0.2/Forwarder/Configuretheuniversalforwarder (zitiert auf S. 36).

[Splg] Splunk Inc. *Create real-time alerts*. Besucht: 01.04.2018. URL: http://docs.splunk.com/Documentation/Splunk/7.0.2/Alert/DefineRealTimeAlerts (zitiert auf S. 35).

[Splh] Splunk Inc. *Free- und Enterprise-Versionen im Vergleich*. Besucht: 15.03.2018. URL: https://www.splunk.com/de_de/products/splunk-enterprise/free-vs-enterprise.html (zitiert auf S. 40).

[Spli] Splunk Inc. *heavy forwarder*. Besucht: 15.03.2018. URL: http://docs.splunk.com/Splexicon:Heavyforwarder (zitiert auf S. 36).

[Splj] Splunk Inc. *How indexing works*. Besucht: 14.04.2018. URL: http://docs.splunk.com/Documentation/Splunk/7.0.3/Indexer/Howindexingworks (zitiert auf S. 35, 46).

[Splk] Splunk Inc. *How to forward data to Splunk Enterprise*. Besucht: 26.03.2018. URL: http://docs.splunk.com/Documentation/Forwarder/7.0.2/Forwarder/HowtoforwarddatatoSplunkEnterprise (zitiert auf S. 37, 41).

[Spll] Splunk Inc. *Install a Windows universal forwarder from an installer*. Besucht: 02.04.2018. URL: http://docs.splunk.com/Documentation/Forwarder/7.0.2/Forwarder/InstallaWindowsuniversalforwarderfromaninstaller (zitiert auf S. 41).

[Splm] Splunk Inc. *Install a Windows universal forwarder from the command line*. Besucht: 02.04.2018. URL: http://docs.splunk.com/Documentation/Forwarder/7.0.2/Forwarder/InstallaWindowsuniversalforwarderfromthecommandline (zitiert auf S. 41).

[Spln] Splunk Inc. *light forwarder*. Besucht: 15.03.2018. URL: http://docs.splunk.com/Splexicon:Lightforwarder (zitiert auf S. 36).

[Splo] Splunk Inc. *Make a universal forwarder part of a host image*. Besucht: 02.04.2018. URL: http://docs.splunk.com/Documentation/Forwarder/7.0.2/Forwarder/Makeauniversalforwarderpartofahostimage (zitiert auf S. 42).

[Splp] Splunk Inc. *Monitor files and directories with inputs.conf*. Besucht: 02.04.2018. URL: http://docs.splunk.com/Documentation/Splunk/7.0.3/Data/Monitorfilesanddirectorieswithinputs.conf (zitiert auf S. 35).

[Splq] Splunk Inc. *Monitor Windows host information*. Besucht: 02.04.2018. URL: http://docs.splunk.com/Documentation/Splunk/7.0.3/Data/MonitorWindowshostinformation (zitiert auf S. 34).

[Splr] Splunk Inc. *Scale your deployment with Splunk Enterprise components*. Besucht: 15.03.2018. URL: http://docs.splunk.com/Documentation/Splunk/7.0.2/Deploy/Distributedoverview (zitiert auf S. 33).

[Spls] Splunk Inc. *Senden automatisch Daten zur Indizierung*. Besucht: 26.03.2018. URL: https://www.splunk.com/de_de/products/splunk-enterprise/features/forwarders.html (zitiert auf S. 37, 40).

[Splt] Splunk Inc. *Set up alert actions*. Besucht: 01.04.2018. URL: http://docs.splunk.com/Documentation/Splunk/7.0.2/Alert/Setupalertactions (zitiert auf S. 35).

[Splu] Splunk Inc. *Splunk Add-on for VMware*. Besucht: 29.03.2018. URL: http://docs.splunk.com/Documentation/AddOns/released/VMW/Collectionconfiguration (zitiert auf S. 37).

[Splv] Splunk Inc. *Splunk DB Connect*. Besucht: 13.03.2018. URL: https://splunkbase.splunk.com/app/2686/ (zitiert auf S. 38).

[Splw] Splunk Inc. *The basics of indexer cluster architecture*. Besucht: 28.03.2018. URL: http://docs.splunk.com/Documentation/Splunk/7.0.2/Indexer/Basicclusterarchitecture (zitiert auf S. 40).

[Splx] Splunk Inc. *The universal forwarder*. Besucht: 15.03.2018. URL: http://docs.splunk.com/Documentation/Forwarder/7.0.2/Forwarder/Abouttheuniversalforwarder (zitiert auf S. 36).

[Sply] Splunk Inc. *Use cron expressions for scheduling*. Besucht: 14.04.2018. URL: http://docs.splunk.com/Documentation/Splunk/7.0.2/Alert/CronExpressions (zitiert auf S. 35).

[Splz] Splunk Inc. *Why source types matter*. Besucht: 13.04.2018. URL: https://docs.splunk.com/Documentation/Splunk/7.0.2/Data/Whysourcetypesmatter (zitiert auf S. 35).

[Splaa] Splunk Inc. *You Have Pricing Questions, We Have Answers!* Besucht: 15.03.2018. URL: https://www.splunk.com/blog/2017/08/11/you-have-pricing-questions-we-have-answers.html (zitiert auf S. 41).

[Spl18a] Splunk Inc. *Managing Indexers and Clusters of Indexers - How indexing works*. Besucht: 12.03.2018. 2018. URL: http://docs.splunk.com/Documentation/Splunk/7.0.2/Indexer/Howindexingworks (zitiert auf S. 33).

[Spl18b] Splunk Inc. *Splunk IT Service Intelligence*. Besucht: 12.03.2018. 2018. URL: https://splunkbase.splunk.com/app/1841/ (zitiert auf S. 38).

[Spl18c] Splunk Inc. *Splunk Machine Learning Toolkit*. Besucht: 12.03.2018. 2018. URL: https://splunkbase.splunk.com/app/2890/ (zitiert auf S. 39, 40).

[Tec] TechTarget. *GUID (global unique identifier)*. Besucht: 22.05.2018. URL: https://searchwindowsserver.techtarget.com/definition/GUID-global-unique-identifier (zitiert auf S. 42).

[Thea] The Apache Software Foundation. *Apache Tomcat 9*. Besucht: 19.05.2018. URL: https://tomcat.apache.org/tomcat-9.0-doc/manager-howto.html#Server_Status (zitiert auf S. 73).

[Theb] The Apache Software Foundation. *FAQ/Logging*. Besucht: 31.03.2018. URL: https://wiki.apache.org/tomcat/FAQ/Logging (zitiert auf S. 26).

[Thec] The Apache Software Foundation. *Legal*. Besucht: 29.03.2018. URL: https://www.apache.org/legal/resolved.html (zitiert auf S. 56).

[The04] The Apache Software Foundation. *Apache License - Apache License, Version 2.0*. 2004. URL: https://www.apache.org/licenses/LICENSE-2.0.html (zitiert auf S. 28).

[Var] Vardhan NS. *Splunk Architecture: Tutorial On Forwarder, Indexer And Search Head*. Besucht: 26.03.2018. URL: https://www.edureka.co/blog/splunk-architecture/ (zitiert auf S. 33).

[Weia] I. Weidner. *Daten sind das neue Gold*. Besucht: 09.05.2018. URL: https://www.computerwoche.de/a/daten-sind-das-neue-gold,3330102 (zitiert auf S. 13).

[Weib] H. Weiss. *Big Data weiter im Höhenflug*. Besucht: 04.05.2018. URL: https://www.vdi-nachrichten.com/Technik-Wirtschaft/Big-Data-im-Hoehenflug (zitiert auf S. 27).

Alle URLs wurden zuletzt am 28. 05. 2018 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift