SV Informatik
GEMEINSAM. ZUKUNFT. ENTWICKELN.

DHBW
Duale Hochschule
Baden-Württemberg
Mannheim

Duale Hochschule Baden-Württemberg
Mannheim

Bachelorthesis

Integration einer Container-Umgebung in einen automatisierten Deployment-Prozess und die Untersuchung ihrer Effekte auf diesen

Studiengang Wirtschaftsinformatik

Studienrichtung Software Engineering

Sperrvermerk

Verfasser/in:	Yves Torsten Staudenmaier
Matrikelnummer:	7146590
Firma:	SV Informatik GmbH
Abteilung:	IE2 – Deployment
Kurs:	WWI17SEC
Studiengangsleiter:	Prof. Dr.-Ing. habil. Dennis Pfisterer
Wissenschaftlicher Betreuer:	Marius Ebel info@mariusebel.net +49 176 / 473 45452
Firmenbetreuer:	Thomas Teske thomas.teske@sv-informatik.de +49 621 / 454 44096
Bearbeitungszeitraum:	17.02.—08.05.2020

Sperrvermerk

Der Inhalt dieser Arbeit darf weder als Ganzes noch in Auszügen Personen außerhalb des Prüfungsprozesses und des Evaluationsverfahrens zugänglich gemacht werden, sofern keine anders lautende Genehmigung der Ausbildungsstätte vorliegt. Die Bachelorarbeit enthält unternehmensinterne Architektur- und Prozessmodellierung und deren Dokumentation. Es ist zum Zeitpunkt der Anmeldung nicht sicher, ob interne Schnittstellen in der Anwendungslandschaft offen gelegt werden.

Mannheim, 05.05.2020

Nadja Haumbach, Ausbildungsverantwortliche

Lesehinweise

Die folgenden Hinweise sollen das Lesen dieser Projektarbeit erleichtern und spezielle Formatierung definieren:

- Im Sinne der Gleichberechtigung wird in dieser Arbeit entweder die Form *„die Entwickler*in“* oder die grammatikalisch korrekte Form *„die/der Entwickler/-in“* verwendet werden. Bei der Kurzform mit der Sternnotation wird auf Grund der Lesbarkeit der weibliche Artikel benutzt.
- Abbildungen, die mit dem Vermerk *unternehmensintern* gekennzeichnet sind, unterliegen folgendem rechtlichen Hinweis: „Alle Rechte, einschließlich der Vervielfältigung, Veröffentlichung, Bearbeitung und Übersetzung bleiben der SV Informatik GmbH vorbehalten.“
- Produkt- oder Eigennamen werden in Kapitälchen gesetzt, wie beispielsweise Node.js.
- Hochgestellte Ziffern weisen auf Fußnoten am Seitenende hin.
- Einen Überblick über die verwendeten Begriffsdefinitionen ist dieser Arbeit**??** auf Seite ?? beigefügt.

Kurzfassung

Titel	Integration einer Container-Umgebung in einen automatisierten Deployment-Prozess und die Untersuchung ihrer Effekte auf diesen
Verfasser/in:	Yves Torsten Staudenmaier
Kurs:	WWI17SEC
Ausbildungsstätte:	SV Informatik GmbH

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Quelltextverzeichnis

Abkürzungsverzeichnis

AWL	Anwendungslandschaft
AWS	Amazon Web Services
BaFin	Bundesanstalt für Finanzdienstleistungsaufsicht
CAB	„Change Advisory Board“
Cloud-C	Cloud-Computing
i.d.R.	in der Regel
IaaS	Infrastructure-as-a-Service
IE	IE – Entwicklungs- und Betriebsunterstützung
IE2	IE2 – Deployment
ITIL	Information Technology Infrastructure Library
IU11	IT-Einkauf/-Recht
K8s	Kubernetes
LXC	Linux Container
NIST	United States National Institute of Standards and Technology
OS	Betriebssystem
PaaS	Platform-as-a-Service
SaaS	Software-as-a-Service
SOA	service-orientierte Architektur
SV	SV SparkassenVersicherung
SVI	SV Informatik GmbH
TTM	Time to Market
VAIT	Versicherungsaufsichtliche Anforderungen an die IT
VM	virtuelle Maschine

1 Einleitung

Motivation der Arbeit irgendwas Orginelles...

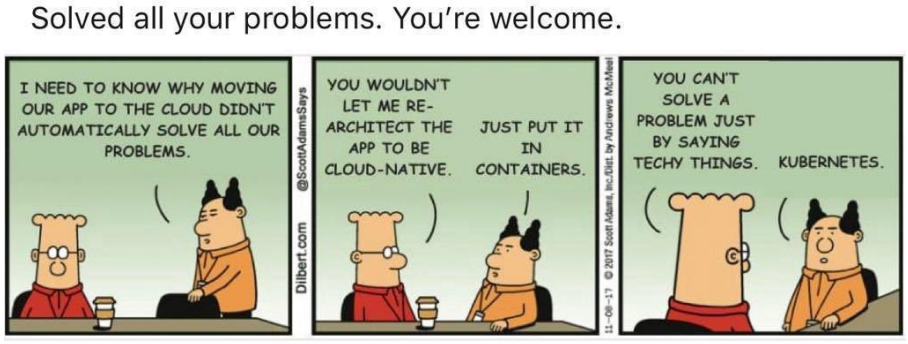

Abbildung 1.1: Dilbert Comic zu Kubernetes
Quelle: *Dilbert on Kubernetes* 2017
Redaktionelle Anmerkung: Abbildung nur als komprimiertes Format verfügbar (Qualitätseinbuße)

Problemstellung/-abgrenzung

Zielstellung der Arbeit

Forschungsfragen/-design Die Forschungsfragen mit der sich diese Bachelorarbeit beschäftigen wird, sind eine direkte Konsequenz aus der Zielstellung und aus den unternehmensinternen Anforderungen an einen möglichen automatisierten Prozess. Dabei liegt der Fokus auf der Betrachtung beider Teildisziplinen der Wirtschaftsinformatik, nämlich der Informatik und der Wirtschaft – jedoch wird der größere Teil dieser Arbeit einen informationstechnischen Fokus besitzen. Die folgende Aufzählung nennt die einzelnen Forschungsfragen, die im weiteren Verlauf ein gemeinsames Ergebnis erbringen werden. Dieses ist in Kapitel 5 auf Seite 19 zu finden.

1. Wie können Container-Anwendungen den Prozess des automatisierten „Deployments“[1] unterstützen?

[1] „Software deployment may be considered to be a process consisting of a number of inter-related activities including the release of software at the end of the development cycle; the configuration of the software, the installation of software into the execution environment, and the activation of the software. It also includes post installation activities including the monitoring, deactivation, updating, reconfiguration, adaptation, redeploying and undeploying of the software.“ (Dearle 2007)

2. Welche wirtschaftlichen Vorteile hat der Einsatz von Container auf den Prozess des automatisierten „Deployments“?
3. Welche besonderen sicherheitstechnischen Aspekte muss ein solcher Prozess im Bereich der Versicherung erfüllen?

Die Forschungsfrage eins wird einen Ist-Zustand analysieren. Dieser enthält eine Prozessanalyse, eine identifizierte Technologie-Wertkette[2] sowie einen Anforderungskatalog der Entwicklungsabteilungen an den zu konzeptionierenden „Deployment“-Prozess für die Container-Anwendungen. Danach wird ein Konzept eines container-basierten, automatisierten „Deployment“-Prozesses erstellt, dabei wird die Methodologie und das eigentliche Konzept erläutert. Die Forschungsfrage eins schließt mit einem Teilergebnis ab.

Die Forschungsfrage zwei beschäftigt sich mit den wirtschaftlichen Vorteilen eines Einsatzes der Container auf den Prozess des automatisierten „Deployment“-Prozesses. Dabei werden die Erstellung eines „Business Case[3]“, die Prüfung der Übereinstimmung der Ziele dieser Arbeit mit der Geschäftsstrategie der SV Informatik GmbH (SVI) und mögliche Disharmonien dieser identifiziert. Außerdem entsteht eine Konzeption eines verbesserten Geschäftsszenarios, das die Kosteneinsparpotentiale und die Zielharmonisierung enthalten wird. Ein Ausblick schließt die Forschungsfrage zwei ab.

Die Forschungsfrage drei identifiziert sicherheitsrelevante Anforderungen, die nicht nur die funktionalen/nicht-funktionalen Anforderungen einer Anwendung betreffen, sondern auch die komplette Anwendungslandschaft (AWL). Dabei beeinflusst die Bundesanstalt für Finanzdienstleistungsaufsicht (BaFin) und auch verschiedene DIN/ISO-Normen diese Anforderungen. Außerdem soll analysiert werden, wie bei der Beschaffung von „open source“- bzw. „closed source“-Anwendungen mögliche Schwachstellen identifiziert werden, die potentielle Angriffsvektoren in der AWL eröffnen würden, und wie mit diesen verfahren wird. Dabei soll versucht werden Rückschlüsse auf die Anwendung OpenShift[4] von Red Hat[5] zu ziehen. Auch hier wird ein Teilergebnis diese Forschungsfrage abschließen

Einordnung der Abteilung in den Geschäftsprozess Die Abteilung IE2 – Deployment (IE2), die sich im Bereich der Organisationseinheit IE – Entwicklungs- und

[2] Definition: <Defintion/>

[3] engl. Geschäftsszenario

[4] „OpenShift is an open source container application platform by Red Hat based on the Kubernetes container orchestrator for enterprise app development and deployment.“ Quelle: Red Hat, Inc. 2020a

[5] „Red Hat ist der weltweit führender Anbieter von Open Source-Lösungen, die auf verlässlichen und leistungsstarken Technologien in den Bereichen Cloud, Virtualisierung, Storage, Linux, Mobile und Middleware basieren. Darüber hinaus bieten wir Support-, Trainings- und Consulting-Services an, die mehrfach prämiert wurden.“ Quelle: Red Hat, Inc. 2020b

Betriebsunterstützung (IE) befindet, befasst sich in erster Linie mit dem Transport („Deployment“) von Software-Artefakten der einzelnen Software-Produkte der SVI. Diese werden für die SV SparkassenVersicherung (SV) entwickelt, betrieben und gewartet. Zu den zentralen Aufgaben der Abteilung gehören die Planung, Durchführung und Überwachung der „Build/Deployment“-Prozesse auf den verschiedenen Serverumgebungen. Des weiteren stellt IE2 die Einspielung von datenbank-relevanten Objekten sicher. Auch entwickelt sie die Bau- und Transportprozesse kontinuierlich weiter und passt diese an die sich ständig veränderten Anforderung der Entwicklungsabteilungen an. Von zentraler Bedeutung ist die Planung und Durchführung der Veröffentlichungen der neuen Versionen einer zu betreuenden Anwendung. Zu dieser Aufgabe gehören auch Aufbau und Bereitstellung der Systemtest-, Releasetest- und Produktions-Umgebungen. Eine weitere zentrale Aufgabe, die nach der Organisationsumstrukturierung am 01.01.2020 in der Abteilung IE2 angesiedelt wurde, ist das Umgebungsmanagement. Die Aufgaben dieses Teilbereichs befasst sich mit folgenden Inhalten: Planung von Aktivitäten in der Produktionsumgebung, Planung und Koordination der Infrastruktur und Notfall-„Fix“ der Produktion, der allgemeinen „Patch“-Planung; Beratung zur Erweiterung, Koordination und Planung von verschiedenen Testumgebungen. Außerdem ist das Umgebungsmanagement Teil des „Change Advisory Board“ (CAB), das ein Gremium nach der Sammlung Information Technology Infrastructure Library (ITIL) darstellt. Dieses ist für die Freigabe von „Changes“ verantwortlich und hat ständige, wie auch der Situation angepasste, Mitglieder.

Aufbau der Arbeit In Kapitel 2 auf der nächsten Seite

In Kapitel 3 auf Seite 14

In Kapitel 4 auf Seite 15

In Kapitel 5 auf Seite 19

2 Wie können Container-Anwendungen den Prozess des automatisierten „Deployments“ unterstützen?

Dieses Kapitel ...

2.1 Grundlagen: Definieren der Begrifflichkeiten zur Forschungsfrage eins

Dieses Teilkapitel soll grundlegende Begrifflichkeiten, die im weiteren Verlauf dieser Arbeit verwendet werden, definieren, um so eine einheitliche Terminologie der Begriffe zu entwickeln. Dadurch wird ein gemeinsames Verständnis erzeugt.

2.1.1 Methodik der Anforderungsanalyse

Die Anforderungsanalyse leitet sich aus dem thematischen Komplex des „Requirements-Engineering“ ab, die verschiedene Bedeutungsvarianten besitzt – dabei „[...] steht [es] einmal für alle konkreten Aktivitäten am Beginn einer Systementwicklung, die auf eine Präzisierung der Problemstellung abzielen. Ebenso steht es aber auch für eine ganze Teildisziplin im Grenzbereich zwischen Systems-Engineering, Informatik und Anwendungswissenschaften.“[6] Diese Analyse soll, laut der herrschenden Meinung der Wissenschaft, am Anfang jeder Systementwicklung stehen, um so bestimmte Vorgehensweise anzuwenden. Dabei entstehen, wenn der später weiter definierte Prozess verfolgt wird, viele systematisch verbundene Dokumente, die Anforderungen enthalten. So ist jede Anforderung wieder ein Cluster von kleineren Anforderungen, die miteinander verbunden sind. Diese werden durch den IEEE-Standard 1220 definiert als „a statement that identifies a product or process operational, functional, or design characteristic or constraint, which is unambiguous, testable or measurable, and necessary for product or process acceptability (by consumers or internal quality assurance guidelines).“[7] Dieser

[6] Partsch 2010, S.19.
[7] IEEE 2005, S.9.

Standard legt mit höchster Priorität den Fokus auf die Formulierung einer Anforderung als elementar wichtig für das Produkt bzw. für das Erreichen der Akzeptanz des Produktes. Ziel der Analyse ist es, funktionale und nicht-funktionale Anforderungen zu identifizieren und diese testbar zu dokumentieren. Funktionale Anforderungen definieren genau, was ein System später erfüllen muss, sie ergeben sich aus der Fragestellung „Was tut das System?/Was soll es aufgrund der Aufgabenstellung können?“[8] Nichtfunktionale Anforderungen konkretisieren die Qualitätsansprüche an das System, die Forderung an das zu implementierende System als Ganzes, sowie Randbedingungen, die aus Projekt-/Prozess-/Unternehmensbedingungen resultieren können.[9]

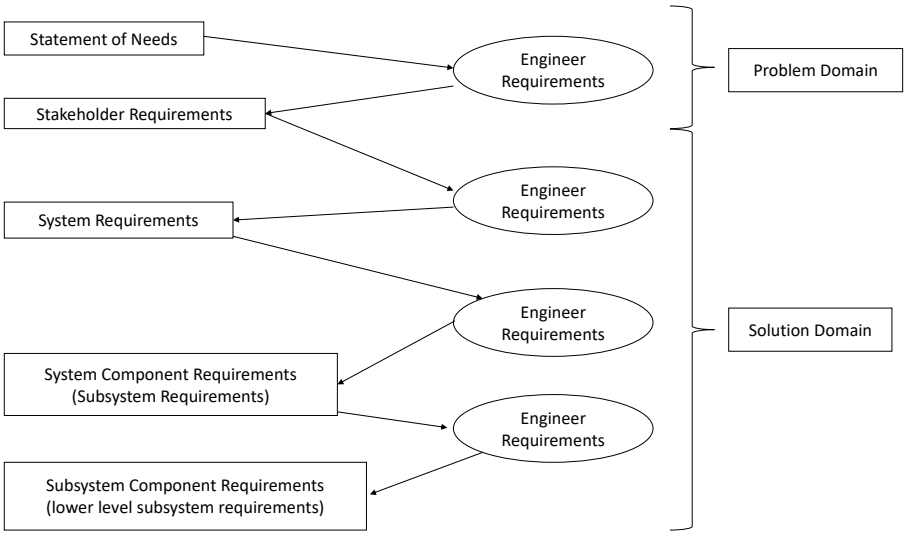

Abbildung 2.1: Entwicklungsprozess der Anforderungen
Quelle: in Anlehnung an Hull, Jackson und Dick 2011, S.28

Das „statement of needs“ ist der Startpunkt für die Entwicklung einer Anforderung die am Ende des Prozesses, der in Abbildung 2.1 dargestellt ist, präzise dokumentiert sein wird. Dieses ist am Anfang immer ein Ausdruck eines Anspruchs oder Wunsches an das zu entwerfende System; dabei bildet das „statement“ und die „stakeholder requirements“ die „problem domain“. Diese definiert grundständige Methodik, wie auch eine nicht-technische Herangehensweise, die auf die Projektbeteiligten („stakeholder“) angepasst ist. Nachfolgend werden die Projektbeteiligen als „stakeholder“ bezeichnen, dabei ist die Rolle beschrieben als „(Stakeholder) sind Personen oder Organisationen, die ein potenzielles Interesse an einem zukünftigen System haben und somit in der Regel auch Anforderungen an das System stellen.“[10] Später definiert die „problem domain“ den Zweck des Systems – dadurch ist bei der Ermittlung der Anforderungen

[8] Partsch 2010, S.27.
[9] vgl. Partsch 2010, S.27-29.
[10] Partsch 2010, S.8.

die Frage „Was ist der Zweck des Systems?“ anstelle „Was soll das System ihrer Meinung nach tun?“. Dies soll die „stakeholder“ extrinsisch motivieren über den Zweck des zu entwerfenden Systems und nicht über einen möglichen Lösungsweg (das Wie) nachzudenken. Durch diesen Ansatz folgen Antworten nach dem Muster „Ich möchte etwas tun können ...“ – wissenschaftlich bzw. literarisch betrachtet sind diese Form der Anforderungen als „capability requirement(s)“[11] bekannt. Sie stellen die wichtigsten Erkenntnisse in der „problem domain“ dar. Nun wird im weiteren Verlauf ein Modell konstruiert, das den Projektbeteiligten, den „stakeholder“, präsentiert wird. Dies unterliegt der Einschränkung, dass es jede/jedem Projektbeteiligte/n versteht. Denn sie validieren das konstruiert Modell in jedem weiteren Schritt, der in Abbildung 2.1 auf der vorherigen Seite, ersichtlich ist. Die Anforderungen an das Modell sind quantitativ gering: es muss nicht-technisch sein und es muss geeignet sein die Anforderungen an das Systems abzubilden. Eine solche Darstellung ist dann geeignet, wenn sie den gewünschten Zweck an das System abbildet, das heißt, dass sie keine technischen Details zeigt, sondern einen Überblick bietet. Ein „use scenario“[12] wird meist verwendet, da es sich eignet menschliche Aktionen bzw. Ziele darzustellen. Abschließend müssen die „stakeholder“-Anforderungen folgende Kriterien erfüllen:

- kurz und prägnant formulierte Beschreibung, jedoch einfach zu verstehen und
- gleichzeitig sollten sie nicht-technisch aber realistisch formuliert sein.

Die „solutions domain“, die auf Abbildung 2.1 auf der vorherigen Seite zu sehen ist, ist die Nachfolgerin von der „problem domain“. Der Hauptunterschied zwischen den beiden Bereichen ist, dass die „solution domain“ idealtypisch qualitativ hochwertig beschriebene Anforderungen als „Input“ bekommt. Dazu konträr erhält die „problem domain“ vage formulierte Wunschliste oder einem nicht klar definierten Ziel als initialen „Input“. Ausgehend von der Aussage von E. Hull, „in an ideal world, all the requirements would be clearly articulated, individual test able requirements“,[13] ist zu deduzieren, dass viele Ebenen zu erforschen gibt, um dieser Aufforderung zu entsprechen. So muss iterativ in jeder Ebene eine neue Analyse des „Inputs“ erfolgen, um einen Ausgangspunkt für das weitere Vorgehen zu initialisieren. Die Komplexität diese Ebenen ist anhängig von dem Grad der Innovation sowie vom Kontext des zu entwickelnden Systems. Jede Entscheidung während des Prozess kann mögliche Entscheidungspfade in einer anderen Ebene verhindern. Ziel des Prozesses ist es, ein Anforderungsdokument/-katalog zu entwerfen, das laut der gesichteten Literatur in verschiedenen Repräsentationen vorliegen kann. Dennoch sollten primäre Bestandteile, wie die Rahmenbedingungen, die Projektbeteiligten, die Projektaspekte und die funktionale/nicht-funktionale Anforderungen, enthalten sein. Ein Beispiel dieses Katalogs ist im Anhang A.1 auf Seite XIV zur Ansicht enthalten. Außerdem gibt es im

[11] vgl. Hull, Jackson und Dick 2011, S.94.
[12] vgl. Hull, Jackson und Dick 2011, S.94.
[13] Hull, Jackson und Dick 2011, S.115.

Bereich der Cloud besondere architektonische Anforderungen. Diese sind im Anhang als Abbildung A.2 auf Seite XVIII einzusehen.

2.1.2 Cloud-Computing (Cloud-C)

Cloud-C ist ein neuartiger und disruptiver Ansatz in der Informationstechnologie, der seit mehreren Jahren Führungskräfte und IT-Abteilungen beschäftigt. Dieser Ansatz verspricht die Lösung für sämtliche Herausforderungen der Kapazitäts- und Leistungsengpässe moderner IT-Infrastruktur zu sein.[14] Auch diskutiert die Bevölkerung stark und meist auch sehr kontrovers über dieses Thema – Themen wie Datenschutz und Privatsphäre; Risiko eines Datendiebstahls und die rechtlichen Fragen sind auch nach 20 Jahren Diskussion immer noch allgegenwärtig. Ein Grund dafür ist die hohe Dynamik dieser Technologie, sowie die ständigen Weiterentwicklung, die von großen Unternehmen, wie Microsoft, Google, Amazon und IBM, voran getrieben werden. Momentan haben Microsoft und Amazon die meisten Marktanteile am Umsatz im Bereich des Cloud-C.[15] Des weiteren prognostiziert Gartner 2019 einen exponentiell wachsenden weltweiten Umsatz bis 2022 auf ungefähr 354,6 Milliarden US-Dollar. Damit würde dieser in den nächsten zwei Jahren um circa 100 Milliarden US-Dollar steigen. Für eine ausführliche Umsatzprognose ist auf die Abbildung A.4 auf Seite XX zu verweisen. Diese verdeutlicht auch, dass in den folgenden Jahren nach 2022 weiterhin mit einer exponentiellen Umsatzsteigerung zu rechnen ist, wenn das mathematische Modell der exponentiellen Regression weiterhin bestand hat.

Historisch betrachtet leitet sich Cloud-C an verschiedenen Konzepten anderer „Computing“-Bereiche und Architekturmustern ab: So spielte zur Entwicklung des heutigen Verständnis „Utility Computing“, „Service Orientation“ und „Grid Computing“ eine große Rolle.[16] John McCarthy hat in den 1960er-Jahren das erste Konzept im Bereich des „Utility Computing“ entwickelt.[17] Später wurde es durch Douglas Parkhill verfeinert und durch die folgenden Schlüsselkomponenten beschrieben: „Parkhill examined the nature of utilities such as water, natural gas and electricity in the way they are provided to create an understanding of the characteristics that computing would require if it was truly a utility. When we consider electricity supply, for example, in the developed world, we tend to take it for granted that the actual electrical power will be available in our dwellings. To access it, we plug our devices into wall sockets and draw the power we need. Every so often we are billed by the electricity supply company, and we pay for what we have used“.[18] Dieses Konzept leitete er auch auf

[14] vgl. Reinheimer und Springer Fachmedien Wiesbaden GmbH 2018, S.4.
[15] siehe dazu Abbildung A.3 auf Seite XIX
[16] vgl. Hill 2013, S.3-5.
[17] vgl. McCarthy 1983.
[18] vgl. Parkhill 1966.

eine technologische Ressource im Bereich des Computers ab.[19] Der Gedanke der Serviceorientierung beschreibt eine klare Begrenzung einer Funktion, die zur Erfüllung eines bestimmten Ziels verwendet wird. Services werden meist durch die Konzepte der Objektorientierung und der Abstraktion in einer Organisation definiert. Aus dem Grundgedanken und den genannten Konzepten entwickelt sich die service-orientierte Architektur (SOA), die diese Prinzipien in ein technologiebasiertes Modell abbildet. Die Leitgedanken der SOA spielen auch im Cloud-C eine wichtige Rolle, denn, wie später noch näher definiert, ist der Servicegedanke ein elementarer Bestandteil der Cloud, der deutlich das Geschäftsmodell prägt. „Grid Computing“ ist ein Konzept aus den 1990er-Jahren und fand seine Anwendung im Bereich der elektrischen Netze.[20] Ziel dieses Konzeptes war es, die Einfachheit und Zuverlässigkeit der Stromnetze zu gewährleisten über einen standardisierten Adapter Zugriff auf dieses zu erhalten ohne sich um die technische Realisierung kümmern zu müssen. Dabei stellten die Pioniere dieses Konzeptes folgende Eigenschaften[21] an das System:

- Dezentrale Ressourcenkontrolle, d. h. ein Grid besteht aus geografisch verteilten Ressourcen, die administrativ unabhängig von Organisationen betreut werden.
- Standardisierte, offene Protokolle und Schnittstellen, d. h. die Grid-Middleware ist nicht anwendungsspezifisch und kann zu verschiedenen Zwecken eingesetzt werden.
- Nichttriviale Eigenschaften des Dienstes, z. B. in Bezug auf Antwortzeitverhalten, Verfügbarkeit oder Durchsatz.

Diese Prinzipien haben eine Ähnlichkeit zu denen des Cloud-C, jedoch sind die wirtschaftlichen Aspekte durch die Gedanken des „Grid Computing“ beschrieben. Des weiteren werden die Aspekte des „Grid Computings“ im Bereich des dezentralen Managements und der verteilten Ressourcen beim Cloud-C nicht weiterverfolgt. Vielmehr bietet die Zentralisierung die ökonomischen Vorteile, die eine zentrale Rolle des Geschäftsmodells darstellen.

Da es mehrere Definitionen von Cloud-C gibt, beschränkt sich diese Arbeit auf folgende: „Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models.“[22] Das United States National Institute of Standards and Technology (NIST)

[19] vgl. Hill 2013, S.4.
[20] vgl. Weinhardt et al. 2009.
[21] vgl. Foster und Kesselman 1999.
[22] Mell und Grance 2011, S.2.

beschreibt in der Publikation Mell und Grance 2011 folgende essentielle Charakteristika[23]:

- on-demand self-service
- broad network access
- resource pooling
- rapid elasticity
- measured service

Des weiteren beschreibt die NIST drei Servicemodelle, wie sich Unternehmen die Cloud zunutze machen können: Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS) und Infrastructure-as-a-Service (IaaS). Dabei wird SaaS definiert als: „The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. [...] The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings."[24] Cloud-Infrastruktur ist eine Sammlung von Hard-/Software des Cloud-Anbieters, die die fünf essentiellen Charakteristika des Cloud-C unterstützt bzw. erfüllt. Beispiele hierfür sind Google Docs und Office 365. PaaS wird beschrieben durch: „The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment."[25] Bei der später in der Konzeptionierung verwendeten Software, OpenShift, handelt es sich um eine PaaS-Lösung. Weitere Beispiele sind Google App Engine, Windows Azure und Heroku.[26] IaaS wird durch folgende Definition abgebildet: „The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications;and possibly limited control of select networking

[23] Jedoch werden diese Charakteristika in anderen wissenschaftlichen Ausarbeitungen um „multitenancy", „service oriented" und „utility-based pricing" ergänzt.(vgl. Institute of Electrical and Electronics Engineers 2011, S.1)
[24] Mell und Grance 2011, S.2.
[25] Mell und Grance 2011, S.2.
[26] vgl. Kumar und Vidhyalakshmi 2018, S.8.

components (e.g. host firewalls).“[27] Hierzu zählen die Produkte Amazon EC2, OpenStack und VMware. Nun sind die Bereitstellungsmodelle der Cloud noch von Bedeutung – die NIST sowie weitere, schon für diesen Abschnitt verwendete, Literatur definiert vier Modelle: „private, community, public and hybrid cloud“ Die „private cloud“ ist in exklusiver Nutzung eines Unternehmens, dass mehrere interne Konsumenten bedient. Es kann entscheiden, ob alle Management-/Betriebsoperationen intern oder extern von einem Anbieter durchgeführt werden. Die Cloud kann intern oder extern gehostet sein. Die „community cloud“ ist eine „private cloud“, jedoch unterscheiden sich die beiden durch die Benutzergruppen. Bei der „community“-Variante ist es nicht auf Organisation sondern auf Gruppe mit gleichen Angelegenheiten beschränkt. Die „public cloud“ ist offen für die Öffentlichkeit natürlich beschränkt durch die Regel des Cloud-Anbieter. Die hybride Variante wird folgendermaßen beschrieben: „The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load balancing between clouds).“[28]

2.1.3 Container(-isierung) und Orchestrierung

„Historically, virtualization technologies have developed out of the need for scheduling processes as manageable container units. The processes and resources in question are the file system, memory, network, and system information.“[29] Aus dieser Notwendigkeit heraus entstanden verschiedene Lösungsansätze: die Virtualisierung in einer virtuellen Maschine (virtuelle Maschine (VM)) und etwas später die Container-Lösungen. Virtuelle Maschinen konnten einige Herausforderungen, wie „scheduling, packaging and resource access“, durch ihre technologischen Ansätze lösen. Dabei wurde der architektonische Ansatz des sogenannten „Gast Systems“ entwickelt, d. h. die virtuelle Maschine ist ein vollwertiges Betriebssystem (OS) mit kompletten Dateisystem und eigenem Prozess auf dem „Host System“.[30] Im Vergleich dazu können Container die gleichen Anforderung abbilden, jedoch unterscheidet sich die Architektur dieser (vgl. Abbildung 2.2 auf der nächsten Seite). Ein Container enthält alle notwendigen, für die App relevanten, Bibliotheken beziehungsweise Abhängigkeiten und kann so, ohne ein komplettes OS, lauffähige Applikationen beinhalten. Diese Abstraktion ist im Cloud-Umfeld (bspw. in einer PaaS-Ausprägung) nützlich, da die Container leichtgewichtiger sind und weniger Speicherauslastung (persistenter Speicher) dadurch benötigen. Auch später für die Orchestrierung der Container in einem Cluster-Umfeld ist dies von Nutzen.

[27] Mell und Grance 2011, S.3.
[28] Mell und Grance 2011, S.3.
[29] Pahl 2015, S.25.
[30] bietet Services für die Gastsysteme an

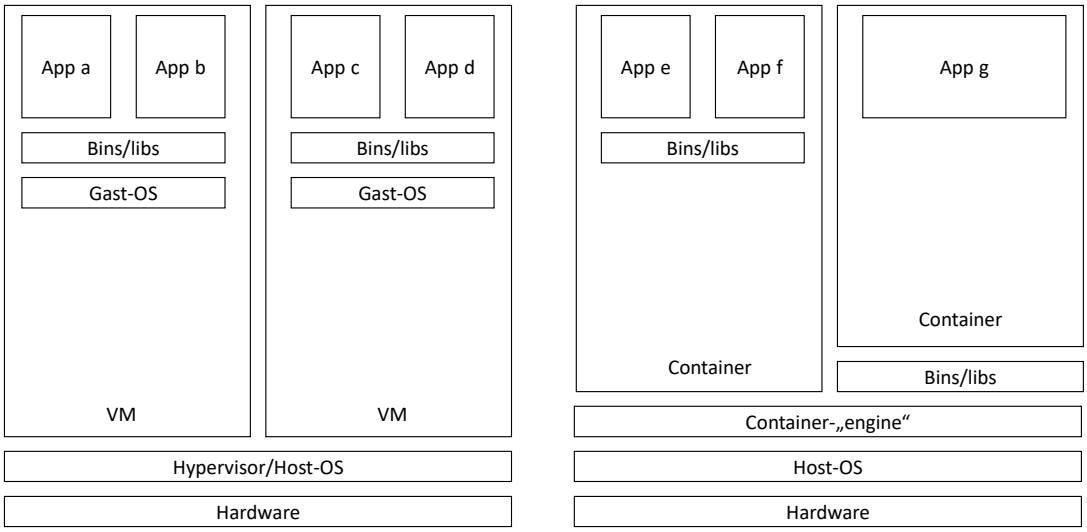

Abbildung 2.2: Architektur der Virtualisierungsmodelle: VM vs. Container
Quelle: in Anlehnung an Pahl 2015

Die gesichtete Literatur (Pahl 2015, Bernstein 2014, Kharb 2016; Combe, Martin und Di Pietro 2016 und weitere siehe Literaturverzeichnis) definieren Container immer anhand ihrer charakteristischen Merkmalen und im Vergleich zur VM. Google Ireland Limited 2020 folgt auch diesem Muster, dennoch eher auf Makroebene: „Container bieten einen logischen Mechanismus der Paketerstellung, der darauf beruht, dass Anwendungen von ihrer Ausführungsumgebung abstrahiert werden. Mit dieser Entkopplung können containerbasierte Anwendungen einfach und konsistent bereitgestellt werden, unabhängig davon, ob es sich bei der Zielumgebung um ein privates Rechenzentrum, die öffentliche Cloud oder auch um den persönlichen Laptop eines Entwicklers handelt.“[31] Ziel der Containerisierung ist es, Entwicklerinnnen die Möglihckeit zu bieten, sich nur auf die Anwendungslogik und -abhängigkeiten zu konzentrieren. Gleichzeitig können andere IT-Teams, wie IE2, sich um die Bereitstellung und Verwaltung dieser Container kümmern. Diese Teams können den Container als geschlossene Verpackung sehen, bei der sie keine Kenntnis über das Innenleben (die Anwendungsdetails) für ihre Arbeit benötigen.[32] Dies ist ein Bestandteil der Grundlagen für schnellere und qualitativ hochwertigere Deployments.[33]

Initial entwickelte Canonical Ltd 2020 die Linux Container (LXC); Docker Inc. ist ein „open source“-Projekt, dass sich die LXC-Technologie zunutze macht und ei-

[31] Google Ireland Limited 2020.
[32] vgl. Google Ireland Limited 2020.
[33] vgl. Kharb 2016, S.1.

ne Container-„engine“ gebaut hat, um diese Container nutzbar zu machen: „Basically, Docker extends LXC with a kernel- and application-level API that together run processes in isolation: CPU, memory, I/O, network, and so on. Docker also uses namespaces to completely isolate an application's view of the underlying operating environment, including process trees, network, user IDs, and file systems.“[34] Docker-Container nutzen eine „Image“-Struktur. So lassen sich durch Kombination verschiedener „Images“ Applikationen abbilden, die durch Programmierlogik ergänzt werden (vgl. Abbildung A.5 auf Seite XXI). In der Industrie ist Docker als de-facto Standard[35] angesehen.[36] Außerdem bietet es viele Vorteile, wie die Lichtgewichtigkeit, „open source“, Sicherheit, Kollaboration zwischen verschiedenen IT-Teams, die Applikation kann überall (wo Docker installiert ist) ausgeführt werden und Docker passt sich an die Unternehmensanforderungen ständig neu an.[37] Durch diese Vorteile entstehen unmittelbare Konsequenzen, die Auswirkungen auf das Arbeiten haben, so wird das Einlernen eines neuen Mitarbeitenden beschleunigt, die Kreativität der Entwicklerinnen verstärkt, die Entwicklungsumgebung vereinheitlicht,[38] die Zusammenarbeit zwischen verschiedenen Teams wird vereinfacht und eine schnelle „Time to Market (TTM)“[39] wird erreicht.[40]

Um die Stärken und Vorteile der Docker-Container, brauchen diese eine Netzwerkanbindung. Nur mit dieser können Container in der Produktion eingesetzt und ein Orchestrierung[41] möglich gemacht werden. Für die Orchestrierung von Container-Anwendungen wird eine weitere Technologie benötigt, die von Google LLC 2020 entwickelt und als „open source“ veröffentlicht wurde: Kubernetes (K8s).[42] Die Semantik des Wortes Kubernetes bedeutet auf griechisch „Loste/Steuermann“. Diese Metapher beschreibt die Hauptaufgaben von K8s zu treffend; es „verdeckt die Hardwareinfrastruktur und stellt ihr gesamtes Rechenzentrum als eine einzige, enorme Rechenressource dar. Dadurch können Sie ihre Softwarekomponenten bereitstellen und ausführen, ohne sich darum zu kümmern, welche Server konkret unterhalb dieser Schicht laufen. Bei der Bereitstellung von Anwendungen mit mehreren Komponenten wählt Kubernetes für jede dieser Komponenten einen Server aus, stellt sie bereit und ermöglicht es ihr, die anderen Komponenten zu finden und mit ihnen zu kommunizie-

[34] Bernstein 2014, S.82.
[35] vgl. Pahl 2015, S.30.
[36] vgl. Kharb 2016, S.1.
[37] vgl. Kharb 2016, S.1.
[38] Dies wirkt sich direkt auf die Code-/Produktqualität aus
[39] „TTM is the strategy of focusing on reducing the time to introduce new products to market.“ (Pawar, Menon und Riedel 1994)
[40] vgl. Kharb 2016, S.2.
[41] Der Begriff ist aus der Musik abgeleitet: flexibles Kombinieren mehrerer Services oder Dienste zu einer sinnvollen Konzeption (Komposition), die einen Geschäftsprozess beschreibt. Quelle: Duden.de
[42] „is an open-source system for automating deployment, scaling, and management of containerized applications.“ (Google LLC 2020)

ren.“[43] Der Nutzen von K8s wird bei einer großen Cloudanbieterin, wie Amazon Web Services (AWS) u. a., maximiert, da es den Entwicklerinnen ermöglicht die Ausführung und Bereitstellung von Anwendungen entkoppelt von den Systemadministratorinnen zu betreiben.[44] Eine grundlegende Übersicht einer KUBERNETES-Architektur ist im Anhang A.6 auf Seite XXII zu finden.

2.2 Ist-Analyse des jetzigen „Deployment“-Prozesses

2.3 Konzeption eines container-basierten, automatisierten „Deployments“

2.4 Ergebnis der Forschungsfrage eins

[43] Lukša 2018, S.4.
[44] vgl. Lukša 2018, S.4.

3 Welche wirtschaftlichen Vorteile hat der Einsatz von Container auf den Prozess des automatisierten „Deployments“?

4 Welche besonderen sicherheitstechnischen Aspekte muss ein solcher Prozess im Bereich der Versicherung erfüllen?

Diese Kapitel ...

Informations- und Kommunikationssysteme sind in der heutigen Gesellschaft von elementarer Bedeutung – sie spielen eine immer größer werdende Rolle. Der Innovationsgrad in der Informationstechnik ist konstant hoch und deswegen sind folgende Bereiche ständiger Weiterentwicklung unterlegen: steigende Vernetzung der Bevölkerung, IT-Verbreitung und Durchdringung, verschwinden der Netzgrenzen, kürze Angriffszyklen auf wichtige Infrastruktur, höhere Interaktivität von Anwendungen und die Verantwortung der Benutzer eines IT-Systems.[45]

4.1 Sicherheitstechnische Anforderungen an den Betrieb einer Anwendung

Informationen sind elementarer Bestandteil der heutigen Welt – diese sind von sehr hohem Wert für Unternehmen und Behörden. Die meisten Geschäftsprozesse, die im heutigen Prozessablauf einer Organisation verankert sind, funktionieren nicht ohne IT-Unterstützung. Somit ist die Informationstechnologie elementarer Bestandteil jedes Unternehmens. Deswegen ist ein zuverlässiges System mit entsprechender Soft- und Hardware unerlässlich. Es muss darauf geachtet werden, dass die Informationen, die auf diesen System verteilt sind, ausreichend gut geschützt sind, damit es nicht zu einer Bedrohungslage kommt. Unzureichend geschützte Systeme stellen ein sehr hohes Risiko dar. „Dabei ist ein vernünftiger Informationsschutz ebenso wie eine Grundsicherung der IT schon mit verhältnismäßig geringen Mitteln zu erreichen. Die verarbeiteten Daten und Informationen müssen adäquat geschützt, Sicherheitsmaßnahmen sorgfältig geplant, umgesetzt und kontrolliert werden. Hierbei ist es aber wichtig, sich nicht nur auf die Sicherheit von IT-Systemen zu konzentrieren, da Informationssicherheit ganzheitlich betrachtet werden muss. Sie hängt auch stark von infrastrukturellen,

[45] vgl. Bundesamt für Sicherheit in der Informationstechnik (BSI) 2020, S.2f.

organisatorischen und personellen Rahmenbedingungen ab. “[46] Die Mängel in der IT-Sicherheit führen meist zu folgenden drei Kategorien von Problemen[47]:

- Verlust der Verfügbarkeit
- Verlust der Vertraulichkeit
- Verlust der Integrität

Der Verlust der Verfügbarkeit eines IT-Systems fällt in der Regel (i.d.R.) sofort auf, da meist Aufgaben ohne diese Informationen nicht weitergeführt werden können. Meist fällt dies in dem Verlust der Funktionen eines Systems auf. Die Vertraulichkeit von personenbezogenen Daten ist ein bestehendes Grundrecht jedes Bürgers beziehungsweise jedes Kunden. Dies ist in verschiedenen Gesetzen wie auch Verordnung geregelt. Diese Daten müssen geschützt werden, da jedes Konkurrenzunternehmen Interesse an den Daten des Unternehmens hat. „Gefälschte oder verfälschte Daten können beispielsweise zu Fehlbuchungen, falschen Lieferungen oder fehlerhaften Produkten führen. Auch der Verlust der Authentizität (Echtheit und Überprüfbarkeit) hat, als ein Teilbereich der Integrität, eine hohe Bedeutung: Daten werden beispielsweise einer falschen Person zugeordnet. So können Zahlungsanweisungen oder Bestellungen zulasten einer dritten Person verarbeitet werden, ungesicherte digitale Willenserklärungen können falschen Personen zugerechnet werden, die digitale Identität wird gefälscht.“[48]

4.1.1 IT-Sicherheit: Grundnorm ISO 27001

4.1.2 IT-Grundschutz-Katalog

4.1.3 Versicherungsaufsichtliche Anforderungen an die IT (VAIT)

4.2 Beschaffung von „open source“-Software

In der SVI gibt es, wie in den meisten anderen Unternehmen, eine prozessorientierte Vorgehensweise, um Software zu beschaffen. Die Beschaffung von Software orientiert sich an ITIL Version 4 – formal ist die Beschaffung von Software mit Hilfe eines

[46] Bundesamt für Sicherheit in der Informationstechnik (BSI) 2020, S.1.
[47] vgl. Bundesamt für Sicherheit in der Informationstechnik (BSI) 2020, S.1ff.
[48] Bundesamt für Sicherheit in der Informationstechnik (BSI) 2020, S.1.

„service requests“[49] zu beantragen. Für die Verteilung der Anwendung müssen danach mehrere „changes“ eingereicht werden. Im weiteren Verlauf wird die „open source“-Variante beleuchtet, da es sich bei den verwendeten Containern, die von Docker Inc. angeboten werden, um diese Variante handelt. Definitionsgemäß muss „open source“-Software laut Opensource.org 2020 folgende Kriterien erfüllen: „free redistribution, source code, derived works, integrity of the author's source Code, no discrimination against persons or groups, no discrimination against fields of endeavor, distribution of license, license must not be specific to a product, license must not restrict other software, license must be technology-neutral“.

Es gibt in der SVI drei Prozesse, die sich in zwei Aspekten unterscheiden: die Kosten und die Anforderungen, die an einen Prozess gestellt werden. Folgende Anfragen gibt es: die Beschaffungsanfrage, die „freeware“-Beschaffung und die juristische Prüfung von Vertragsdokumenten oder Sachverhalten. Die Beschaffungsanfrage wird bei kostenpflichtiger Software beantragt. Da es in diesem Kapitel um die kostenlose Software geht, wird auf die weitere Ausführung dieser Anfrage verzichtet. Der Prozess „freeware“-Beschaffung wird laut den Juristen der Abteilung IT-Einkauf/-Recht (IU11) kaum[50] verwendet, denn die Fachbereiche[51] (die IT-Abteilungen) arbeiten zum jetzigen Zeitpunkt an dem Prozess vorbei – sie übergehen wissentlich diesen. Folgende Probleme haben sich bei der Befragung der Fachbereiche herausgestellt: die Anforderungen, die dieser Prozess an sie stellt, sind „nicht verhältnismäßig“ gegen über dem Nutzen; die Fachbereiche wissen nicht, dass es einen solchen Prozess gibt oder ignorieren diesen. Die Anforderungen/Kriterien, die die Abteilung IU11 festgelegt hat, sind folgende: es muss eine Produktverantwortliche definiert werden, es muss eine Architekturfreigabe von den zuständigen „Entreprise“-Architekten beantragt werden und es muss der genaue, angedachte Verwendungszweck der einzukaufenden „freeware/open source“-Software definiert werden. Diese Hürden, aus Sicht der IT-Abteilung, erfüllen nicht die Kosten-Nutzen-Konformität. Aus rechtlicher Sicht ist das ein sehr hoch zu bewertendes Risiko, da es zu unmittelbaren juristischen Konsequenzen führen kann. Deswegen nutzt die IT-Abteilung meist den rechtlichen Prozess (juristische Prüfung von Vertragsdokumenten oder Sachverhalten) da dieser nicht die oben genannten Hürden enthält. Bei diesem wird der Verwendungszweck der Software erfragt und die Lizenz dieser durch IU11 geprüft. Jedoch ist davon auszugehen, dass eine offizielle Beschaffungsanfrage bei „open source“-Software in wenigen[52] Fällen gestellt wird. Begründet durch die Administrator-Berechtigung, die es Benutzern erlaubt ohne Restriktionen alles auf ihrem Computer zu installieren, kann keine numerische Aus-

[49] a request from a user or a user's authorized representative that initiates a service action which has been agreed as a normal part of service delivery. Quelle: AXELOS Limited und Stationery Office (Great Britain) 2019, S.195

[50] $n \leq 5, n \in \mathbb{N}_0$, gemessen p. a.

[51] aus Sicht von IU11

[52] $n \leq 10, n \in \mathbb{N}_0$, gemessen p. a.

sage über die Dunkelziffer getroffen werden. Es bleibt nur die Hypothese der Juristen der Abteilung IU11, die weder falsifizierbar noch validierbar ist.

Ist die Software in der AWL implementiert, gibt es noch eine Anwendung, Nexus Lifecycle von sonatype, die auf eventuelle Schwachstellen dieser benutzten Software prüft. Nexus Lifecycle ist eine Hilfsanwendung, die u. a. auch von Creditreform verwendet wird. Das Ziel dieses Produktes ist es, die gesamte Software-„Supply Chain“ kontinuierlich zu bereinigen und sicher zu halten.[53] Aus dem Prüfbericht werden dann entsprechende Maßnahmen abgeleitet. Die erste ist die Software, in der die Schwachstelle gefunden wurde, als unsicher zu markieren und danach zu sperren. Nun müssen die Entwicklungsabteilung versuchen die Schwachstellen zu beseitigen. Problematisch ist es, wenn diese ignoriert werden. In letzter Konsequenz wird der Betrieb und die Verteilung der Anwendung gestoppt. Dies führt zu massiven Problem in der Produktion und somit verringert sich die vertragliche, mit dem Kunden vereinbarte, Verfügbarkeit der Systeme.

4.3 Konzept zur Implementierung der Sicherheitsanforderungen

4.4 Ergebnis der Forschungsfrage drei

[53] vgl. Sonatype Inc. 2020.

5 kritische Betrachtung

5.1 Zusammenfassung der Erkenntnisse

5.2 Fazit

5.3 Ausblick

Literaturverzeichnis

Atomic Requirement Download (19. Aug. 2019). URL: https://www.volere.org/atomic-requirement-download/.

AXELOS Limited und Stationery Office (Great Britain) (2019). *ITIL® Foundation, ITIL 4 edition*. v4. OCLC: 1122856407. Norwich: TSO (The Stationery Office). ISBN: 978-0-11-331607-6. URL: https://www.axelos.com/getmedia/5896d51f-ab6c-4843-992b-4f045eab0875/ITIL-4-Foundation-glossary_v0_22.aspx (besucht am 10. 03. 2020).

Bernstein, David (Sep. 2014). „Containers and Cloud: From LXC to Docker to Kubernetes“. In: *IEEE Cloud Computing* 1.3, S. 81–84. ISSN: 2372-2568. DOI: 10.1109/MCC.2014.51.

Bundesamt für Sicherheit in der Informationstechnik (BSI) (2020). *IT-Grundschutz-Kompendium*. 2020. Aufl. OCLC: 1027470677. Bundesanzeiger Verlag GmbH. 816 S. ISBN: 978-3-8462-0906-6. URL: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Grundschutz/Kompendium/IT_Grundschutz_Kompendium_Edition2020.pdf?__blob=publicationFile&v=6 (besucht am 09. 03. 2020).

Canonical Ltd (2020). *Linux Containers*. Linux Container. URL: https://linuxcontainers.org/ (besucht am 17. 03. 2020).

Combe, Theo, Antony Martin und Roberto Di Pietro (Sep. 2016). „To Docker or Not to Docker: A Security Perspective“. In: *IEEE Cloud Computing* 3.5, S. 54–62. ISSN: 2372-2568. DOI: 10.1109/MCC.2016.100.

Dearle, Alan (Mai 2007). „Software Deployment, Past, Present and Future“. In: *Future of Software Engineering (FOSE '07)*. Future of Software Engineering (FOSE '07). Minneapolis, MN: IEEE, S. 269–284. ISBN: 978-0-7695-2829-8. DOI: 10.1109/FOSE.2007.20. URL: https://ieeexplore.ieee.org/document/4221626/ (besucht am 18. 03. 2020).

Dilbert on Kubernetes (11. Aug. 2017). URL: https://miro.medium.com/max/1024/1*R0DEnf_7sjswuBHouioQFg.jpeg.

Foster, Ian und Carl Kesselman, Hrsg. (1999). *The grid: blueprint for a new computing infrastructure*. San Francisco: Morgan Kaufmann Publishers. 677 S. ISBN: 978-1-55860-475-9.

Gartner (2019). *Cloud Computing - Umsatz bis 2022*. Statista. Library Catalog: de.statista.com. URL: https://de.statista.com/statistik/daten/studie/195760/umfrage/umsatz-mit-cloud-computing-weltweit/ (besucht am 13. 03. 2020).

Google Ireland Limited (2020). *Container und ihre Vorteile*. Google Cloud. Library Catalog: cloud.google.com. URL: https://cloud.google.com/containers?hl=de (besucht am 17. 03. 2020).

Google LLC (2020). *Production-Grade Container Orchestration with K8s*. Library Catalog: kubernetes.io. URL: https://kubernetes.io/ (besucht am 18. 03. 2020).

Hill, Richard, Hrsg. (2013). *Guide to cloud computing: principles and practice*. Computer communications and networks. OCLC: ocn807043821. London ; New York: Springer. 278 S. ISBN: 978-1-4471-4602-5 978-1-4471-4603-2.

Hull, Elizabeth, Ken Jackson und Jeremy Dick (2011). *Requirements engineering*. 3rd ed. London ; New York: Springer. 207 S. ISBN: 978-1-84996-404-3 978-1-84996-405-0.

IEEE (2005). „IEEE Standard for Application and Management of the Systems Engineering Process". In: *IEEE Std 1220-2005 (Revision of IEEE Std 1220-1998)*, S. 1–96. DOI: 10.1109/IEEESTD.2005.96469.

„Cloud Computing: Deployment Models, Delivery Models, Risks and Research Challenges" (2011). In: 2011 International Conference on Computer and Management (CAMAN). Hrsg. von Institute of Electrical and Electronics Engineers. OCLC: 838686677. Wuhan, China: IEEE, S. 1–4. ISBN: 978-1-4244-9283-1 978-1-4244-9282-4. DOI: 10.1109/CAMAN.2011.5778816. URL: https://ieeexplore.ieee.org/abstract/document/5778816.

ITCandor (2019). *Cloud Computing - Marktanteile der führenden Unternehmen 2019*. Statista. Library Catalog: de.statista.com. URL: https://de.statista.com/statistik/daten/studie/150979/umfrage/marktanteile-der-fuehrenden-unternehmen-im-bereich-cloud-computing/ (besucht am 13. 03. 2020).

Kharb, Dr Latika (2016). „Automated Deployment of Software Containers Using Dockers". In: 4.10, S. 3.

Kumar, Vikas und R. Vidhyalakshmi (2018). *Reliability aspect of cloud computing environment*. Singapore: Springer. 170 S. ISBN: 9789811330230 9789811330223. URL: https://doi.org/10.1007/978-981-13-3023-0.

Lukša, Marko (2018). *Kubernetes in Action: Anwendungen in Kubernetes-Clustern bereitstellen und verwalten*. OCLC: 1017485179. München: Hanser. 642 S. ISBN: 978-3-446-45510-8 978-3-446-45602-0.

McCarthy, John (1983). *REMINISCENCES ON THE HISTORY OF TIME SHARING*. REMINISCENCES ON THE HISTORY OF TIME SHARING. URL: http://www-formal.stanford.edu/jmc/history/timesharing/timesharing.html (besucht am 13. 03. 2020).

Mell, Peter und Timothy Grance (28. Sep. 2011). *The NIST Definition of Cloud Computing*. Special Publication (NIST SP) 800-145. Gaithersburg, MD 20899-8930: National Institute of Standards und Technology. 7 S. URL: https://doi.org/10.6028/NIST.SP.800-145 (besucht am 13. 03. 2020).

Opensource.org (2020). *The Open Source Definition*. The Open Source Definition. URL: https://opensource.org/osd (besucht am 10. 03. 2020).

Pahl, Claus (Mai 2015). „Containerization and the PaaS Cloud". In: *IEEE Cloud Computing* 2.3, S. 24–31. ISSN: 2372-2568. DOI: 10.1109/MCC.2015.51.

Parkhill, Douglas Freeman (1966). *The Challenge of the computer utility: D.F. Parkhill.* OCLC: 460679364. Reading: Mass., London : Addison-Wesley Publishing C°.

Partsch, Helmuth (2010). *Requirements-Engineering systematisch: Modellbildung für softwaregestützte Systeme*. 2., überarb. und erw. Aufl. eXamen.press. OCLC: 845656932. Berlin: Springer. 394 S. ISBN: 978-3-642-05358-0 978-3-642-05357-3. URL: http://dx.doi.org/10.1007/978-3-642-05358-0.

Pawar, Kulwant S., Unny Menon und Johann C.K.H. Riedel (1. Jan. 1994). „Time to Market". In: *Integrated Manufacturing Systems* 5.1. Publisher: MCB UP Ltd, S. 14–22. ISSN: 0957-6061. DOI: 10.1108/09576069410815765. URL: https://doi.org/10.1108/09576069410815765 (besucht am 17. 03. 2020).

Red Hat, Inc. (2020a). *OpenShift Container Platform by Red Hat, Built on Kubernetes*. OpenShift Container Platform by Red Hat, Built on Kubernetes. URL: https://www.openshift.com/ (besucht am 11. 03. 2020).

Red Hat, Inc. (2020b). *Red Hat – Wir entwickeln Open Source-Technologien für Unternehmen*. Red Hat – Wir entwickeln Open Source-Technologien für Unternehmen. URL: https://www.redhat.com/de (besucht am 11. 03. 2020).

Reinheimer, Stefan und Springer Fachmedien Wiesbaden GmbH, Hrsg. (2018). *Cloud Computing: die Infrastruktur der Digitalisierung*. Edition HMD. OCLC: 1038769740. Wiesbaden: Springer Vieweg. 216 S. ISBN: 978-3-658-20966-7 978-3-658-20967-4. URL: https://doi.org/10.1007/978-3-658-20967-4.

Rimal, Bhaskar Prasad et al. (März 2011). „Architectural Requirements for Cloud Computing Systems: An Enterprise Cloud Approach". In: *J Grid Computing* 9.1, S. 3–26. ISSN: 1570-7873, 1572-9184. DOI: 10.1007/s10723-010-9171-y. URL: http://link.springer.com/10.1007/s10723-010-9171-y (besucht am 13. 03. 2020).

Sonatype Inc. (2020). *Nexus Lifecycle Product*. Nexus Lifecycle Product. URL: https://de.sonatype.com/product-nexus-lifecycle?utm_campaign=NVS&utm_source=ppc&utm_medium=adwords&ahcsource=paid&utm_term=%2Bnexus%20%2Blifecycle&hsa_tgt=kwd-437257894053&hsa_grp=90875397990&hsa_src=s&hsa_net=adwords&hsa_mt=b&hsa_ver=3&hsa_ad=406628330148&hsa_acc=2665806879&hsa_kw=%2Bnexus%20%2Blifecycle&hsa_cam=8625747087&gclid=EAIaIQobChMIgsvQt8mP6AIVh-h3Ch29zQJJEAAYASAAEgK02fD_BwE (besucht am 10. 03. 2020).

Volere Requirements Specification Template (19. Aug. 2019). URL: https://www.volere.org/templates/volere-requirements-specification-template/.

Weinhardt, Christof et al. (Okt. 2009). „Cloud Computing – A Classification, Business Models, and Research Directions“. In: *Bus. Inf. Syst. Eng.* 1.5, S. 391–399. ISSN: 1867-0202. DOI: 10.1007/s12599-009-0071-2. URL: http://link.springer.com/10.1007/s12599-009-0071-2 (besucht am 13. 03. 2020).

A Ergänzungen zur Forschungsfrage eins

In diesem Teil des Anhangs sind Ergänzungen zur Forschungsfrage zwei des Kapitels 3 auf Seite 14 beschrieben.

A.1 Anforderungsdokument

Ein Anforderungskatalog hat bestimmte Anforderungen, die an den Katalog gestellt werden. Neben der Forderung nach Einhaltung der Qualitätskriterien, definiert nach dem ISO-Standard 9000/9001, sind noch folgende Forderungen in der Literatur beschrieben:[54]

- vollständig (inhaltlich – d. h., alle Anforderungen sind erfasst –, formal, Normkonform)
- konsistent (keine Widersprüche zwischen den Bestandteilen des Dokuments, insbesondere keine Konflikte zwischen verschiedenen Anforderungen)
- lokal änderbar (Änderungen an einer Stelle sollten keine Einflüsse auf Konsistenz und Vollständigkeit des Gesamtdokuments haben)
- verfolgbar (ursprüngliche Stakeholderwünsche und Zusammenhänge zwischen Anforderungen sind leicht zu finden)
- klar strukturiert
- umfangsmäßig angemessen
- sortierbar/projezierbar (nach verschiedenen Kriterien, für verschiedene Stakeholder).

Die folgende Aufzählung beschreibt eine Vorlage für das Anforderungsdokument nach Quelle: Sie nutzt die Hilfsmittelsammlung „Volere“. Diese bietet im Themenbereich „requirements engineering“ kostenpflichtig Dokumentenvorlagen an. Die beiden Bekanntesten sind die hier gezeigte „Volere Requirements Specification Template“ und das kostenlose „Volere Atomic Requirement Template“, das umgangssprachlich „Snow Card“ genannt wird. Die „Snow Card“ (A.1 auf Seite XVII) ist eine Karteikarte, die

[54] sig. Partsch 2010, S.34.

benutzt wird, um eine vollständige Aufnahme aller Informationen einer einzelnen Anforderung zu gewährleisten.[55] Die folgende Liste wurde in Anlehnung an die Quelle *Volere Requirements Specification Template* 2019 erstellt.

[55] vgl. *Atomic Requirement Download* 2019.

- Projekt-Treiber
  1. Zweck des Projekts
  2. Auftraggeber, Kunde und andere Stakeholder
  3. Nutzer des Produkts
- Projekt-Randbedingungen
  1. Einschränkungen
  2. Namenskonventionen und Definitionen
  3. Relevante Fakten und Annahmen
- Funktionale Anforderungen
  1. Arbeitsrahmen
  2. Systemgrenzen
  3. Funktionale und Daten-Anforderungen
- Nicht-funktionale Anforderungen
  1. Look-and-Feel-Anforderungen
  2. Usability-Anforderungen
  3. Performanz-Anforderungen
  4. Operationale und Umfeld-Anforderungen
  5. Wartungs- und Unterstützungsanforderungen
  6. Sicherheitsanforderungen
  7. Kulturelle und politische Anforderungen
  8. Rechtliche Anforderungen
- Projekt-Aspekte
  1. Offene Punkte
  2. Standardlösungen
  3. Neu aufgetretene Probleme
  4. Installationsaufgaben
  5. Migrationstätigkeiten
  6. Risiken
  7. Kosten
  8. Nutzerdokumentation
  9. Zurückgestellte Anforderungen
  10. Lösungsideen

Abbildung A.1: Volere Snow Card
Quelle: *Atomic Requirement Download* 2019

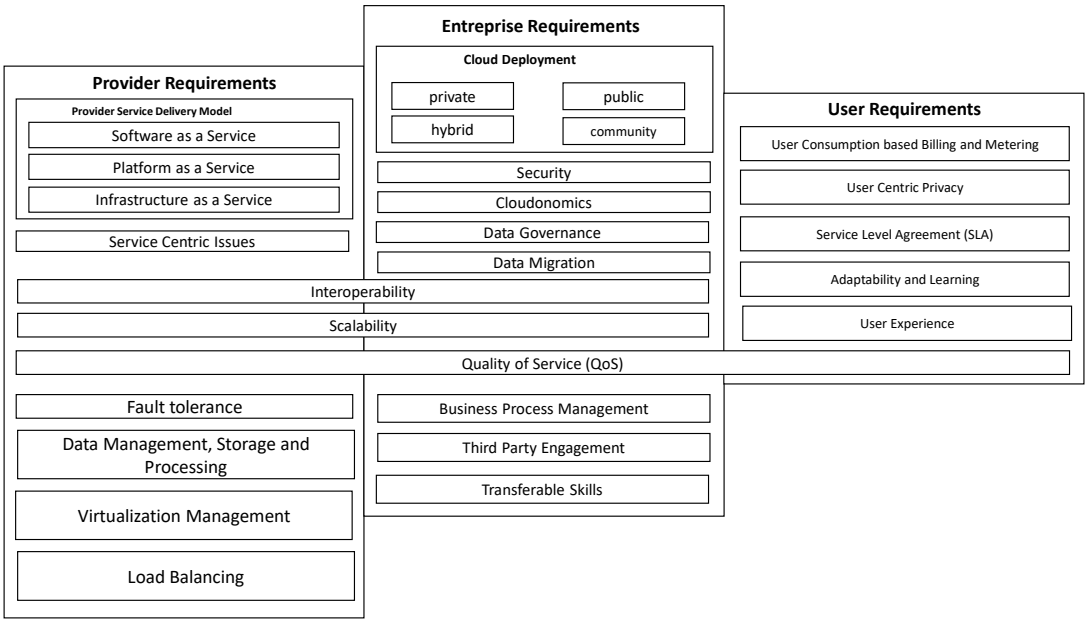

Abbildung A.2: Volere Snow Card
Quelle: in Anlehnung an Rimal et al. 2011

A.2 Statistiken zum Themengebiet Cloud-C

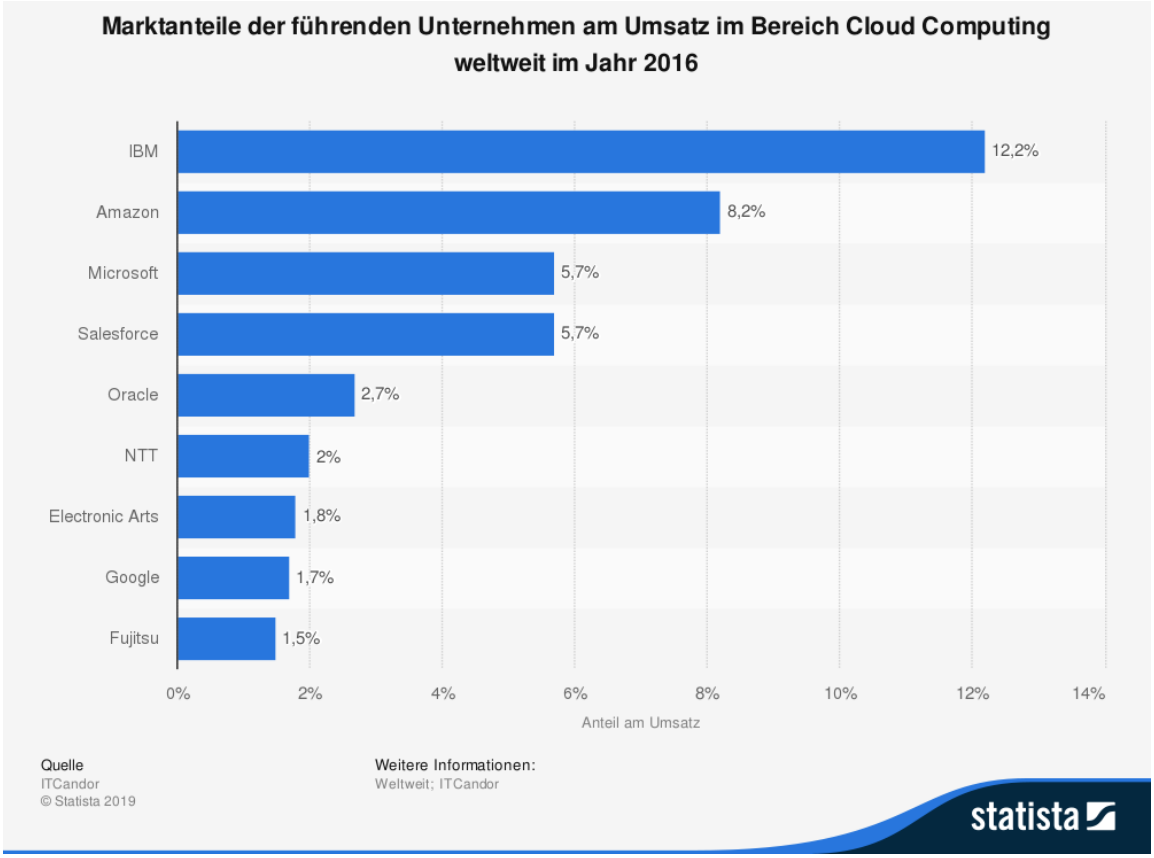

Abbildung A.3: Marktanteile der führenden Unternehmen am Umsatz im Bereich Cloud Computing weltweit von Juli 2018 bis Juni 2019
Quelle: ITCandor 2019

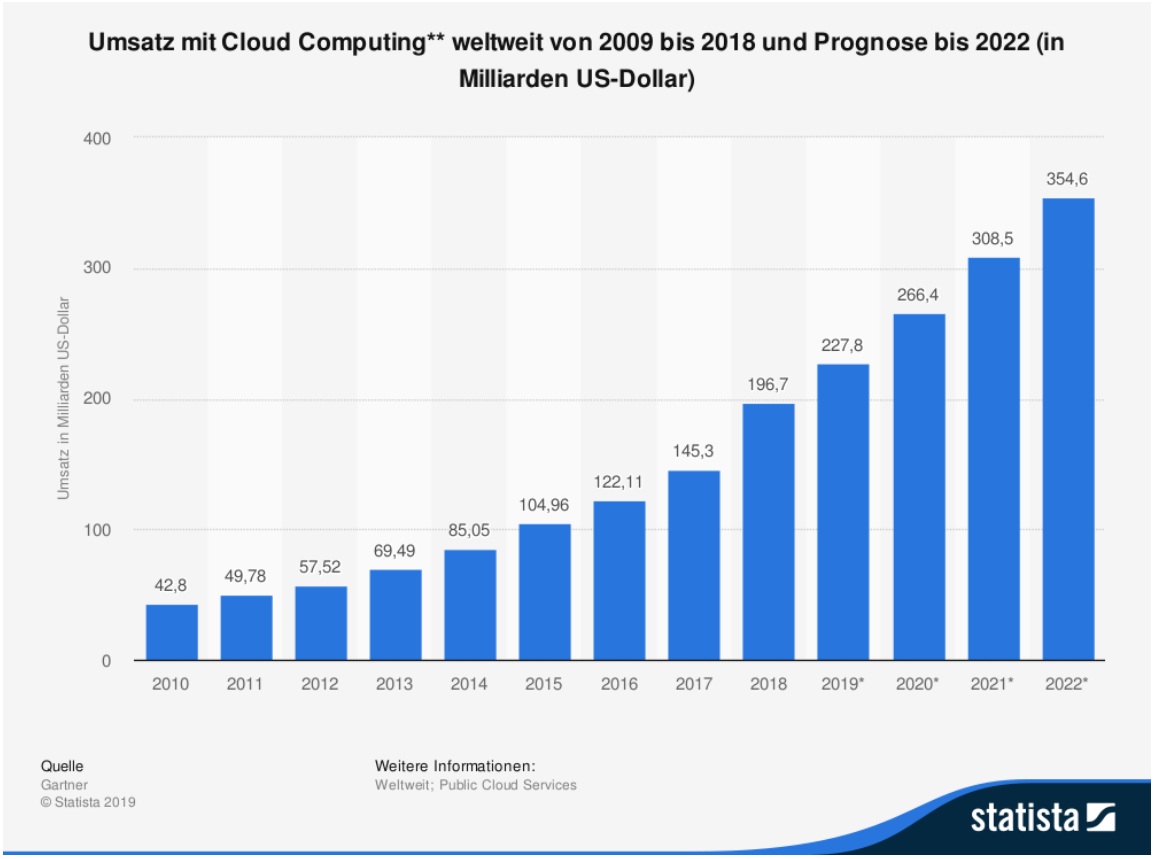

Abbildung A.4: Umsatz mit Cloud-Computing weltweit von 2009 bis 2018 und Prognose bis 2022

Quelle: Gartner 2019

A.3 Ergänzungen zum Kapitel Container(-isierung) und Orchestrierung

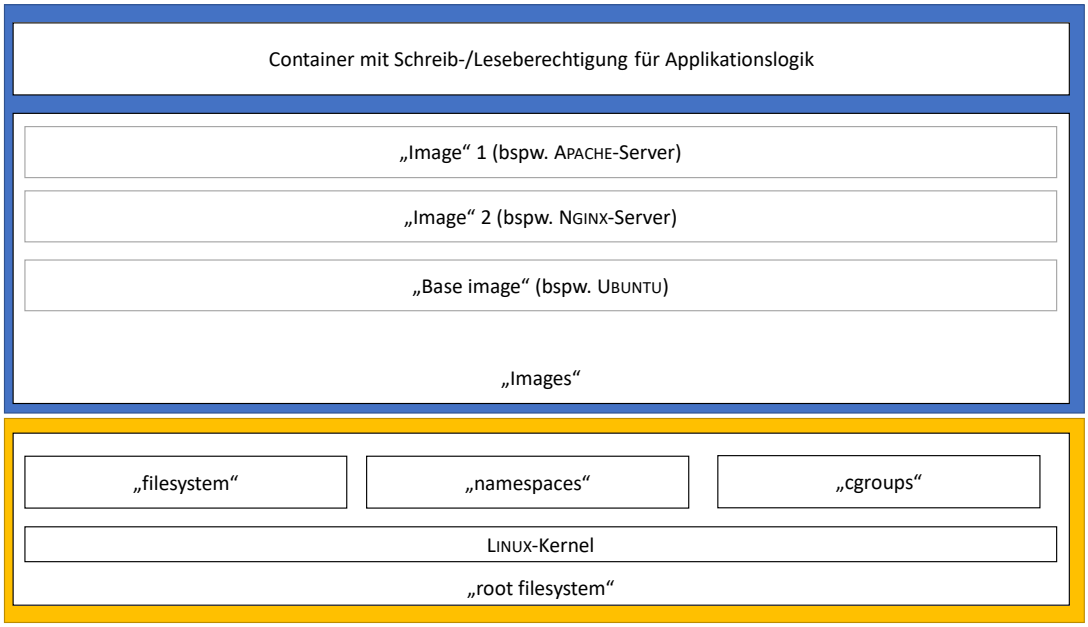

Abbildung A.5: Architektur des Container-„Images“
Quelle: in Anlehnung an Pahl 2015

Hierbei ist zu beachten, dass das orange gefärbte die Funktionalitäten der DOCKER-„Engine“ und das blau gefärbte die möglichen Bestandteile eines DOCKER-„Images“ darstellen soll.

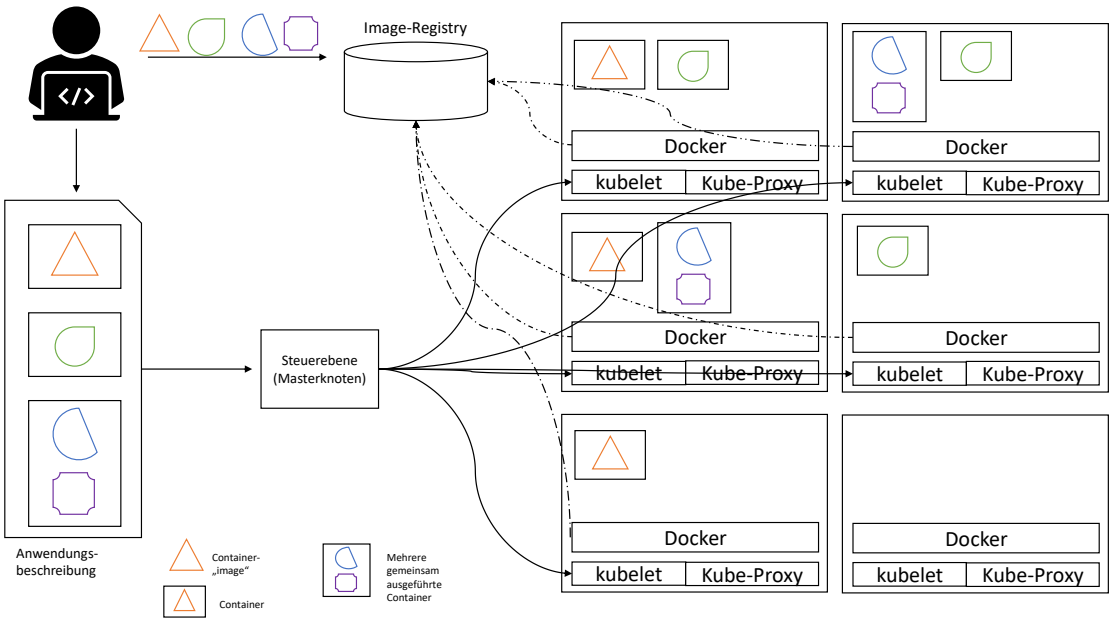

Abbildung A.6: Überblick über eine KUBERNETES-Architektur
Quelle: in Anlehnung an Lukša 2018, S.23

B Ergänzungen zur Forschungsfrage zwei

C Ergänzungen zur Forschungsfrage drei

Ehrenwörtliche Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit mit dem Thema: *Integration einer Container-Umgebung in einen automatisierten Deployment-Prozess und die Untersuchung ihrer Effekte auf diesen* selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Ort, Datum	Yves Torsten Staudenmaier