

Untersuchung der Leistungsfähigkeit unterschiedlicher Cloud-Speicherdienste mit der Schnittstelle des Simple Storage Service (S3)

Masterthesis

zur Erlangung des akademischen Grades
Master of Science (M.Sc.)
am Fachbereich 2: Informatik und Ingenieurwissenschaften
der Frankfurt University of Applied Sciences

Eingereicht von

Marius Wernicke

Referent Prof. Dr. Christian Baun
Korreferent Prof. Dr. Thomas Gabel

September 2017

Eidesstattliche Erklärung

Ich erkläre an Eides statt, diese Masterthesis selbstständig und ohne Benutzung anderer als der von mir angegebenen Hilfsmittel angefertigt zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder unveröffentlichten Schriften entnommen sind, wurden als solche kenntlich gemacht. Die Arbeit wurde in gleicher oder ähnlicher Form oder auszugsweise im Rahmen einer anderen Prüfung noch nicht vorgelegt.

Frankfurt am Main, den 7. September 2017

Marius Wernicke

Danksagungen

Ich bedanke mich bei meiner Familie und meinen Freunden für das Verständnis, die viele Hilfe und Unterstützung während der Zeit des Schreibens, in welcher ich wenig Zeit für jene hatte. Besonderer Dank geht an alle, die diese Arbeit (mehrfach) gelesen und mir hilfreiche Verbesserungen und Tipps gegeben haben.

Ich danke meinem Referenten Herrn Prof. Dr. Christian Baun für das spannende Thema und die Unterstützung während der Arbeit. Herrn Prof. Dr. Thomas Gabel danke ich für die Betreuung als Korreferent.

Zusammenfassung

Cloud Computing ist eines der wichtigsten IT-Themen im Jahr 2017. Infrastrukturen, Dienste und Daten werden in die Cloud verschoben und sollen trotzdem so erreichbar wie im eigenen Netzwerk sein. Besonders die ausfallsichere Datenhaltung in der Cloud stellt eine Herausforderung dar.

Um den Dienst mit der besten Gesamtleistung zu bestimmen, werden im Rahmen dieser Masterthesis unterschiedliche Cloud-Speicherdienste untersucht und getestet. Virtualisierung und Cloud Computing bilden dabei die technologischen Grundlagen für die Untersuchung der Leistungsfähigkeit. Schwerpunkte sind die Speicherdienste der öffentlichen Clouds wie Amazon Web Services (AWS), Microsoft Azure (MSA) und die Google Cloud Plattform (GCP) sowie der privaten Open-Source-Clouds. Dabei werden die öffentlichen Clouds lediglich konfiguriert und die privaten Clouds zusätzlich vorher installiert.

Die Sicherheit und Integrität der Daten spielt eine wichtige Rolle. Zur Steigerung dieser Eigenschaften, werden verteilte Dateisysteme untersucht und innerhalb der Speicherdienste installiert und eingebunden, sodass die Daten an mehr als einem Ort vorgehalten werden. Diese werden anschließend auf ihre Leistungsfähigkeit hin untersucht und bewertet. Die Daten selbst werden mit einer Message-Digest Algorithm 5 (MD5)-Prüfsumme versehen, um zu untersuchen, ob diese nach der Datenübertragung fehlerfrei und unverändert sind.

Auf Basis eines Skripts werden automatisiert Tests durchgeführt, wobei unterschiedliche Aktionen ausgeführt und die Zeiten gemessen werden. Mit einer eigens entwickelten Auswertungstabelle und Bewertungsmatrix ist es möglich, die Dienste anhand ihrer Gesamtleistung zu bewerten und den idealen Cloud-Speicherdienst, auf Basis der Schnittstelle des Simple Storage Service (S3), zu ermitteln.

Abstract

Cloud computing is one of the most important IT topics in 2017. Infrastructures, services and data are moved into the cloud and should still be as accessible as in your own network. The fail-safe data storage in the cloud is a particular challenge.

In order to determine the service with the best overall performance, different cloud storage services are investigated and tested within the framework of this master thesis. Virtualization and cloud computing form the technological basis for the investigation of performance. The focus is on the storage services of public clouds such as Amazon Web Services (AWS), Microsoft Azure (MSA) and Google Cloud Plattform (GCP) as well as the private open source clouds. The public clouds are only configured and the private clouds are installed beforehand.

The security and integrity of the data plays an important role. To increase these features, distributed file systems are examined and installed and integrated within the storage services, so that the data is stored in more than one location. These are then examined and evaluated for their performance. The data itself are provided with a Message-Digest Algorithm 5 (MD5) checksum to check whether they are error-free and unchanged after the data transfer.

Tests are carried out automatically based on a script, whereby different actions are carried out and the times are measured. With a specially developed evaluation table and evaluation matrix, it is possible to evaluate the services on the basis of their overall performance and to determine the ideal cloud storage service based on the Simple Storage Service (S3) interface.

Inhaltsverzeichnis

Abbildungsverzeichnis	X
Tabellenverzeichnis	XII
Abkürzungsverzeichnis	XIII
1 Einführung und Motivation	1
1.1 Fragestellung und Herausforderungen	2
1.2 Lösungsansätze	3
1.3 Aufbau dieser Arbeit	4
2 Technische Grundlagen	6
2.1 Virtualisierung	6
2.1.1 Virtualisierung auf Betriebssystemebene	7
2.1.2 Emulation	8
2.1.3 Paravirtualisierung	8
2.1.4 Virtuelle Maschine	8
2.1.5 Virtualisierungslösung	9
2.1.6 Speichervirtualisierung	9
2.1.7 Netzwerkvirtualisierung	9
2.2 Virtualbox	10
2.3 Service-orientierte Architekturen	10
2.4 Web-Services	11
3 Cloud Computing	13
3.1 Bereitstellungsmodelle	13
3.1.1 Öffentliche (public) Cloud	13
3.1.2 Private (private) Cloud	14
3.1.3 Hybride (hybrid) Cloud	14
3.1.4 Gemeinschaftliche (Community) Cloud	14
3.2 Everything as a Service (XaaS)	15

3.2.1	Infrastructure as a Service (IaaS)	16
3.2.2	Plattform as a Service (PaaS)	16
3.2.3	Software as a Service (SaaS)	16
3.2.4	Weitere Ansätze	17
3.3	Amazon Web Services (AWS)	18
3.3.1	Elastic Compute Cloud (EC2)	19
3.3.2	Simple Storage Service (S3)	21
3.3.3	Virtual Private Cloud (VPC)	22
3.4	Microsoft Azure und Google Cloud Plattform	22
4	Möglichkeiten der Leistungsuntersuchung	24
4.1	Bibliotheken und Programme	24
4.2	Skripte	25
4.2.1	S3 Performance Test Tool	26
4.2.2	s3perf	26
4.2.3	Weitere Skripte	28
5	Ausgewählte Cloud-Speicherdienste	29
5.1	Öffentliche Cloud	29
5.1.1	Amazon Simple Storage Service (S3)	30
5.1.2	Microsoft Azure Blob Storage (ABS)	33
5.1.3	Google Cloud Storage (GCS)	37
5.2	Private Cloud	41
5.2.1	Eucalyptus Walrus	41
5.2.2	Fake S3	43
5.2.3	Minio	45
5.2.4	Nimbus Cumulus	47
5.2.5	OpenStack Swift	50
5.2.6	Riak Cloud Storage	54
5.2.7	S3 Ninja	58
5.2.8	S3rver	61
5.2.9	Scality CloudServer	63
6	Redundanz durch verteilte Dateisysteme	66
6.1	Verteilte Dateisysteme	66
6.1.1	GlusterFS	67
6.1.2	GlusterFS-Server installieren	67
6.1.3	GlusterFS-Client installieren	69
6.2	Cloud-Speicherdienste mit GlusterFS	69

6.2.1	Minio	69
6.2.2	Scality CloudServer	70
7	Untersuchung der Leistungsfähigkeit	72
7.1	Messung der Leistung	72
7.1.1	VirtualBox	73
7.1.2	Amazon S3	73
7.1.3	Azure BS	73
7.1.4	Google CS	73
7.1.5	Minio	74
7.1.6	Cumulus	74
7.1.7	Swift	74
7.1.8	Riak CS	74
7.1.9	S3 Ninja	75
7.1.10	S3rver	75
7.1.11	Scality CloudServer	75
7.2	Betrachtung der Datenintegrität	75
7.3	Ergebnisse	76
7.3.1	Bewertung	78
7.3.2	Bewertungsmatrix	81
7.3.3	Visualisierung der Bewertungen	82
8	Barrierefreie Cloud-Speicherdienste	83
8.1	Hochverfügbarkeit und Ausfallsicherheit	83
8.2	Implementierung von Cloud-Speicherdiensten	84
9	Fazit und Ausblick	86
9.1	Ausblick	90
	Literaturverzeichnis	91
A	Skript s3perf	i
B	AWS Instanzenübersicht	ii
C	VirtualBox VM Übersicht	iii
D	Einzelheiten der Tests	iv
D.1	Amazon S3	iv
D.2	Azure BS	v
D.3	Google CS	vii

D.4 Minio	viii
D.5 Cumulus	ix
D.6 Swift	xi
D.7 Riak CS	xii
D.8 S3 Ninja	xiii
D.9 S3rver	xv
D.10 Scalify CloudServer	xvi

Abbildungsverzeichnis

2.1	Die Schichten bis hin zu einer VM. Eine Virtualisierungslösung läuft auf physischer Hardware und einem Betriebssystem. Mit der Anwendung können VMs erzeugt werden, die wieder ein Betriebssystem und Anwendungen besitzen (in Anlehnung an [53]). Die untere Schicht ist die Host-Umgebung und die obere Schicht die Gast-Umgebung.	7
2.2	Simpler Aufbau einer Service-orientierten Architektur (in Anlehnung an [13])	11
3.1	Aufbau von Everything as a Service, der sehr gut aufzeigt, dass die Spitze, die Software, nicht ohne die anderen Unterschichten lauffähig ist (in Anlehnung an [78]).	15
5.1	Erstellung eines Buckets (s3-mt-testbucket) in der S3 Management Console. Die Einstellungen auf den folgenden Reitern wurden auf den Standardwerten belassen.	32
5.2	Übersicht über die Buckets, die sich im S3 befinden. Über einen Mausklick auf das Feld des Buckets, werden die Eigenschaften aufgerufen. Bei der Auswahl des Namens bewegt sich der Nutzer eine Ebene tiefer zum Inhalt des Buckets.	32
5.3	Eine Ebene tiefer wird eine Übersicht der im Bucket enthaltenen Daten angezeigt, wo diese bearbeitet werden können.	33
5.4	Das Speicherkonto (akurasol ist ein fiktiver Name) öffnet die Übersicht über die Container. Weitere Menüpunkte sind in der mittleren Spalte zu sehen.	34
5.5	Die Übersicht über die Dateien bzw. Blobs in den Containern wird an die Containerübersicht angedockt bzw. erweitert.	35
5.6	Azure-Dashboard mit geöffneter Cloud Shell.	36
5.7	Die Übersichtsseite der Buckets, wo der Nutzer animiert wird, einen Bucket zu erstellen, sollte noch keiner vorhanden sein.	38

5.8	Neben der Vergabe eines Namens für den Bucket muss festgelegt werden, welcher Klasse dieser zugewiesen wird und wie Daten darin genutzt werden sollen.	39
5.9	Die Seite mit dem Inhalt des Buckets gibt übersichtlich an, welche Aktionen durchgeführt werden können.	39
5.10	Die Google Cloud Shell innerhalb der Weboberfläche der GCP.	40
5.11	Über den Webclient können Daten auch visuell bearbeitet werden. Dieser ist vergleichbar zu den Weboberflächen der öffentlichen Clouds.	47
5.12	Das Webinterface von S3 Ninja zeigt die Zugangsschlüssel sowie Buckets und Dateien an, welche über die Oberfläche ebenfalls bearbeitet werden können.	60
7.1	Gestapeltes Balkendiagramm analog zu den Ergebnissen der Bewertungsmatrix mit Rang am Ende des jeweiligen Balkens.	82
B.1	Alle erstellten Instanzen innerhalb der AWS. Die IP von Swift ist blau, da dies eine Elastic IP ist, welche von Vagrant erzwungen wurde.	ii
C.1	Alle erstellten VMs in VirtualBox.	iii

Tabellenverzeichnis

3.1	Standorte der Rechenzentren der AWS [10].	18
3.2	Auswahl verfügbarer Instanz-Typen und welche Ressourcen sie zu bieten haben. Anhand der Ressourcenverteilung ist der vorgesehene Einsatzzweck ersichtlich. [3]	20
6.1	Leistungsvergleich der Gesamtzeiten in Sekunden von Minio ohne und mit GlusterFS.	70
6.2	Leistungsvergleich der Gesamtzeiten in Sekunden des Scalify CloudServers ohne und mit GlusterFS.	71
7.1	Die Ergebnisse (Gesamtzeiten in Sekunden) aus den Tests mit dem Skript s3perf.	77
7.2	Bewertung der einzelnen Speicherdienste anhand von festgelegten Kriterien. Die Kriterien haben eine Gewichtung und die Dienste werden von 0 bis 10 Punkte bewertet. Der Nutzwert errechnet sich wie in Kapitel 7.3.1 angegeben.	81

Abkürzungsverzeichnis

ABS	Azure Blob Storage
AMI	Amazon Machine Image
API	Application Programming Interface
AWS	Amazon Web Services
CE	Compute Engine
CPU	Central Processing Unit
EBS	Elastic Block Store
EC2	Elastic Compute Cloud
GB	Gigabyte
GCP	Google Cloud Plattform
GCS	Google Cloud Storage
GPL	GNU General Public License
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
IAM	Identity and Access Management
KVM	Kernel-based Virtual Machine
MD5	Message-Digest Algorithm 5
MSA	Microsoft Azure
OSS	Open-Source-Software
OVA	Open Virtual Appliance
QEMU	Quick EMUlator
RAM	Random Access Memory
REST	Representational State Transfer
RVM	Ruby Version Manager
S3	Simple Storage Service
SDK	Software Development Kit
SOAP	Simple Object Access Protocol
SSH	Secure Shell
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol

TTS	Text-to-Speech-System
UDP	User Datagram Protocol
URI	Uniform Resource Identifiers
vCPU	virtual CPU
VLAN	Virtual Local Area Network
VM	Virtual Machine
VMM	Virtual-Machine-Monitor
VPC	Virtual Private Cloud
WSDL	Web Services Description Language
WWW	World Wide Web
XML	Extensible Markup Language

1 Einführung und Motivation

Cloud Computing ist, zusammen mit IT-Sicherheit, eines der wichtigsten Themen für die Digitalwirtschaft im Jahr 2017 [22]. Beide Themen sind eng miteinander verknüpft, da der Anspruch an die Sicherheit mit der Weiterentwicklung und kontinuierlichen Nutzung von Cloud Computing immer größer wird.

Cloud Computing bezeichnet eine IT-Infrastruktur, welche Speicher, Rechenleistung, Software oder Plattformen über das Internet bereitstellt. Über Netzwerkprotokolle und Internetbrowser kann auf diese Dienste zugegriffen werden. Das Ziel ist die Auslagerung der Rechenleistung in das Internet bzw. zu einem externen Dienstleister. Neben dem Rechnen können auch Daten gesichert und verteilt verfügbar gemacht werden.

In den letzten Jahren entwickelte sich für die Sicherung und den Austausch von Daten der Simple Storage Service (S3)¹, welcher von Amazon bereitgestellt wird, zum Standard für Objekt Storage Dienste [12]. Das Spezielle an diesem Dienst bzw. dieser Schnittstelle ist die Ähnlichkeit zu Verzeichnissen und Daten, was mittlerweile von vielen Anbietern nachgeahmt wird. [5]

Mit der Zeit haben sich etliche Systeme etabliert und der Markt wächst weiter. Neben einfachen Diensten wie z.B. Nextcloud oder Dropbox, existieren auch Dienste für den professionellen Einsatz, wie für Unternehmen oder im Bereich der Hochverfügbarkeit. Bekannte Größen sind die Amazon Web Services (AWS), Microsoft Azure (MSA) sowie die Google Cloud Plattform (GCP) im Bereich der offenen Clouds. Bei den privaten Clouds gibt es sehr viele Open-Source²-Angebote, welche innerhalb einer eigenen Infrastruktur installiert werden können. Bekannte Systeme sind z.B. Minio, OpenStack und Scality CS.

1 S3 ist ein Objektspeicher und kann, je nach Infrastruktur, beliebig große Datenmengen speichern und verwalten [5].

2 Quelloffene Software, die von Dritten eingesehen, geändert und genutzt werden kann. Meistens ist diese kostenlos nutzbar [65].

1.1 Fragestellung und Herausforderungen

Bei der Fülle an Möglichkeiten stellt sich zwangsläufig die Frage, welches Cloud-System am besten für den jeweiligen Zweck geeignet ist. Um das herauszufinden, können unterschiedliche Leistungsmessungen, sogenannte Benchmarks, durchgeführt werden. Am einfachsten kann die Leistung eines Cloud-Speichers anhand seiner Geschwindigkeit, also den Zeiten des Hoch- und Herunterladens, bewertet werden.

Zusätzliche Eigenschaften, wie z.B. Latenz³, Verfügbarkeit, Sicherheit und Zuverlässigkeit, können Aufschluss über die Leistungsfähigkeit geben. Diese ist abhängig davon, wie Leistung von dem jeweiligen Nutzer definiert wird. Die genannten Eigenschaften sind meist von vielen weiteren Faktoren, wie beispielsweise dem Server, der Netzwerkanbindung und dem Betriebssystem, abhängig und können nicht ausschließlich anhand der Cloud-Speicherdienste festgestellt werden.

Solche und ähnliche Benchmarks wurden schon von Bjornson [23], Garfinkel [39], Land [55] und Palankar et al. [68] durchgeführt. Alle Arbeiten haben sich jedoch lediglich auf die AWS, MSA und GCP konzentriert. Andere Anbieter oder gar Open-Source-Speicherdienste wurden außen vor gelassen.

Bisher existieren keine Leistungsmessungen zu Open-Source-Clouds bzw. privaten Clouds, wovon es mittlerweile eine große Anzahl gibt, welche auch die Schnittstelle des S3 nutzen. Dabei ist es durchaus sinnvoll, einen Überblick über die Leistungsfähigkeit der verschiedenen Clouds zu haben, bspw. aus Gründen des Datenschutzes und der Sicherheit.

Eine Herausforderung stellt die Auswahl der Cloud-Speicherdienste dar. Da es sich hierbei um Speicherdienste mit der Schnittstelle des S3 handelt, ist die Auswahl bereits eingegrenzt. Mögliche Kandidaten wurden bereits auf der Github-Seite des Skripts, welches zur Bearbeitung der Fragestellung im Kapitel 1.2 erläutert wird, aufgelistet [20]. Eine Recherche ergab weitere Kandidaten (siehe Kapitel 5), die zum Testen eingesetzt und verglichen werden können.

Neben der Suche nach den passenden Speicherdiensten ist das Installieren und Konfigurieren der Auswahl eine weitere Herausforderung. Viele Produkte verfügen zwar über eine Dokumentation und geben Anweisungen zur Installation, jedoch nicht in der Art, wie sie für diese Arbeit benötigt werden. Wie in Kapitel 1.2 näher beschrieben wird, sollen sämtliche Clouds innerhalb der AWS installiert werden. Dies setzt voraus, dass

³ Auch Verzögerungszeit, ist die Zeit, die eine Nachricht von einem Ende des Netzwerks zu einem anderen benötigt [19]. Die Latenz kann über einen Ping bestimmt werden, was meist in Millisekunden (ms) angegeben wird.

die Infrastruktur der AWS diese Installationen auch zulässt bzw. ermöglicht. Sollten bei einer Installation Komplikationen auftreten, ist der Fehler zu analysieren und, wenn möglich, zu beheben. Eine Cloud alternativ bei einem anderen Anbieter zu installieren, stellt keine mögliche Lösung dar, da dadurch die Vergleichbarkeit der unterschiedlichen Systeme nicht mehr gewährleistet ist.

Eine weitere Möglichkeit, die den Leistungsvergleich erweitert, ist die parallele Installation innerhalb einer virtuellen Maschine (VM). Diese kann mit (virtueller) Hardware, ähnlich zu der in den AWS, eingerichtet werden, sodass auch hier eine Vergleichbarkeit ermöglicht wird. Diese ist nicht mit den Resultaten aus den Installationen der AWS gleichzusetzen, sondern vielmehr wird ein weiterer Leistungsvergleich unabhängig der AWS geschaffen. Besonders populär ist in diesem Bereich die Open-Source-Software VirtualBox von Oracle, welche, nach Möglichkeit, für die Testinstallationen genutzt wurde.

Das Besondere an zwei verschiedenen Infrastrukturen ist das Netzwerk. Die AWS sind extern und eine VM wird in den meisten Fällen intern im privaten Netzwerk betrieben. Dies ermöglicht die zusätzliche Bewertung der Netzwerkleistung bzw. es wird ersichtlich, wie groß der Unterschied der Übertragungszeiten ist.

Auch auf die Sicherheit der Daten und deren Kompromittierung⁴ wird eingegangen. Diese können beim Hoch- bzw. Herunterladen verändert werden und diese kompromittierten Daten stellen damit eine potentielle Gefahr dar. Aus diesem Grund sollte ein Ansatz erstellt werden, inwieweit mit solchen Daten weiter verfahren wird und wie die Daten innerhalb der Cloud sicher abgelegt werden können.

1.2 Lösungsansätze

Bisher wurden, wie in Kapitel 1.1 beschrieben, keine Leistungsmessungen mit privaten Clouds bzw. deren Open-Source-Diensten durchgeführt. Aus diesem Grund behandelt diese Arbeit einen Leistungsvergleich unterschiedlicher Cloud-Speicherdienste, welche mit der Schnittstelle des S3 arbeiten.

Um die Leistungsfähigkeit eines Dienstes bewerten zu können, werden Zufallsdaten mit einer spezifischen Größe generiert und auf den Dienst hoch- und davon heruntergeladen. Dabei wird gemessen, wie viel Zeit diese Aktion in Anspruch genommen hat. Zusätzlich

⁴ Wenn ein System kompromittiert wurde, ist dieses System nicht mehr vertrauenswürdig. Das heißt, dass alle Daten und Programme manipuliert sein könnten und dass alle Informationen, die auf dem System gespeichert waren oder verarbeitet worden sind, an Dritte weitergegeben worden sein könnten. [75]

wird mit einer Message-Digest Algorithm 5 (MD5)-Prüfsumme⁵ die Datenintegrität kontrolliert.

Die unterschiedlichen Dienste werden innerhalb der Infrastruktur der AWS installiert, um die Vergleichbarkeit gewährleisten zu können. Der Vorteil ist hierbei, dass die AWS gleichbleibende Leistung bieten und somit alle Dienste die gleichen Voraussetzungen besitzen. Eine Installation auf verschiedenen Diensten ist, wie zuvor erwähnt, auf Grund der unterschiedlichen Hardware nicht aussagekräftig.

Neben der Installation der privaten Clouds innerhalb der AWS werden der S3, der Google Cloud Storage (GCS) und der Azure Blob Storage (ABS) zum Vergleich herangezogen. Diese Dienste sind besonders auf das Speichern von Daten optimiert, welche damit sehr gut zum Vergleich herangezogen werden können. So kann auch untersucht werden, ob die privaten Clouds die gleiche oder eine ähnliche Leistung bieten können wie die öffentlichen Clouds oder umgekehrt.

Das Ergebnis dieser Arbeit ist eine strukturierte Liste und eine Bewertungsmatrix, welche die Leistungen sowie Vor- und Nachteile der einzelnen Dienste aufzeigen. Neben den Zeiten des Hoch- und Herunterladens, wird auch angezeigt, ob tatsächlich eine Kompromittierung während der Übertragung stattgefunden hat. Ist dies der Fall, kann diese Cloud nicht weiter zum Vergleich herangezogen werden, da dies ein großes Sicherheitsrisiko darstellt. Mit verschiedenen Kriterien werden die einzelnen Dienste bewertet.

Aus den Listen gehen einer oder mehrere Cloud-Speicherdienste hervor, welche neben der Leistung auch bei den anderen Kriterien eine sehr gute Bewertung erhalten haben. Damit können die Ergebnisse dieser Arbeit als Vergleich für zukünftige Entscheidungen genutzt und erweitert werden.

Besonders durch die Kosten für Infrastruktur, Flächen, Zeit und Personal ist es nur vorteilhaft, wenn bekannt ist, welches System für welchen Einsatzzweck die beste Leistung bietet.

1.3 Aufbau dieser Arbeit

Dieser erste Abschnitt zählt die verschiedenen Kapitel auf und gibt eine Übersicht über das Thema dieser Arbeit. Kapitel 2 und 3 beschäftigen sich mit der zugrunde

⁵ MD5 ist ein Hash-Algorithmus, der aus beliebigen Daten oder Nachrichten einen 128-bit Hashwert erzeugt [72].

liegenden Technik und dem Schwerpunkt Cloud Computing. In Kapitel 4 werden die Möglichkeiten der Leistungsuntersuchung aufgezählt. Dabei wird untersucht, welche Werkzeuge und Programmbibliotheken existieren.

Die einzelnen Cloud-Speicherdienste werden in Kapitel 5 näher beschrieben. Dazu zählen auch detaillierte Dokumentationen der Installationen und Konfigurationen sowie Probleme und Fehler. Im darauf folgenden Kapitel 6 wird auf die Lösungen der zu speichernden Daten eingegangen. Es wird untersucht, wie verteilte Dateisysteme in die Cloud-Speicherdienste eingebunden werden können und ob diese einen Einfluss auf die Leistungsfähigkeit haben.

Kapitel 7 führt die eigentliche Leistungsuntersuchung durch. Dazu wird eine Tabelle und Bewertungsmatrix aufgestellt, die die unterschiedlichen Cloud-Speicherdienste vergleichend darstellt und bewertet. Daneben wird die Datensicherheit bzw. die Integrität der Daten untersucht und bewertet.

Die vorliegende Arbeit wird im Masterstudiengang *Barrierefreie Systeme* der Frankfurt University of Applied Sciences (FRA-UAS) verfasst. Daher wird in Kapitel 8 auf die Barrierefreiheit und Interdisziplinarität des Themas dieser Arbeit eingegangen. Es wird die Frage geklärt, inwieweit diese Arbeit die Barrierefreiheit optimieren kann.

Die Arbeit schließt mit einem Fazit und einem Ausblick in Kapitel 9 ab. Zusätzlich werden die Ergebnisse und gewonnenen Erkenntnisse evaluiert.

Im Anhang werden die detaillierten Ergebnisse und zusätzliche Informationen aufgeführt.

2 Technische Grundlagen

Das Internet bildet eine Grundlage für die Arbeit mit der Cloud¹. Das bedeutet, dass keine physischen Infrastrukturen vorhanden sind, sondern diese bei einem Dienstleister stehen und ein Zugang zur Infrastruktur über das Internet zur Verfügung gestellt wird. Das hat den Vorteil, dass Kosten, Flächen, Zeit und Personal eingespart werden können. Anbieter großer Cloud-Infrastrukturen sind z.B. AWS, MSA oder GCP.

Neben der Notwendigkeit des Internets benötigt das Cloud Computing umfassende Möglichkeiten der Virtualisierung. Zur Umsetzung dieser existieren verschiedene Methoden, welche auf verschiedenen Technologien basieren.

2.1 Virtualisierung

Virtualisierung ist die Basis für das Cloud Computing. Dabei wird eine logische Schicht auf einem physischen Wirt abgebildet, worauf mehrere, voneinander isolierte, VMs laufen können, welche sich die physischen Ressourcen teilen (siehe Abbildung 2.1) [30]. Den verschiedenen VMs können feste Ressourcen zugewiesen werden, womit unterschiedliche Einsatzszenarien geschaffen werden. Das bedeutet, dass dynamisch auf eine Anfrage reagiert und diese umgesetzt werden kann. Ein Applikationsserver, welcher viele Berechnungen anstellt, benötigt weitaus mehr Ressourcen als z.B. ein Webserver, welcher lediglich eine Webseite mit wenigen Anfragen ausgibt. So können nicht nur dem Applikationsserver zeitnah die benötigten Ressourcen zugespield werden, sondern auch performante Lastverteilungen auf dem Host-System gewährleistet werden.

Ein gängiges Modell ist derzeit die stundenweise Abrechnung. Dabei werden einem Server für die entsprechende Laufzeit Ressourcen zugewiesen und berechnet. Wenn der Server abgeschaltet oder pausiert wird, werden die Ressourcen wieder freigegeben

¹ Das Internet wird oftmals als Wolke dargestellt, wodurch dieser Begriff entstand

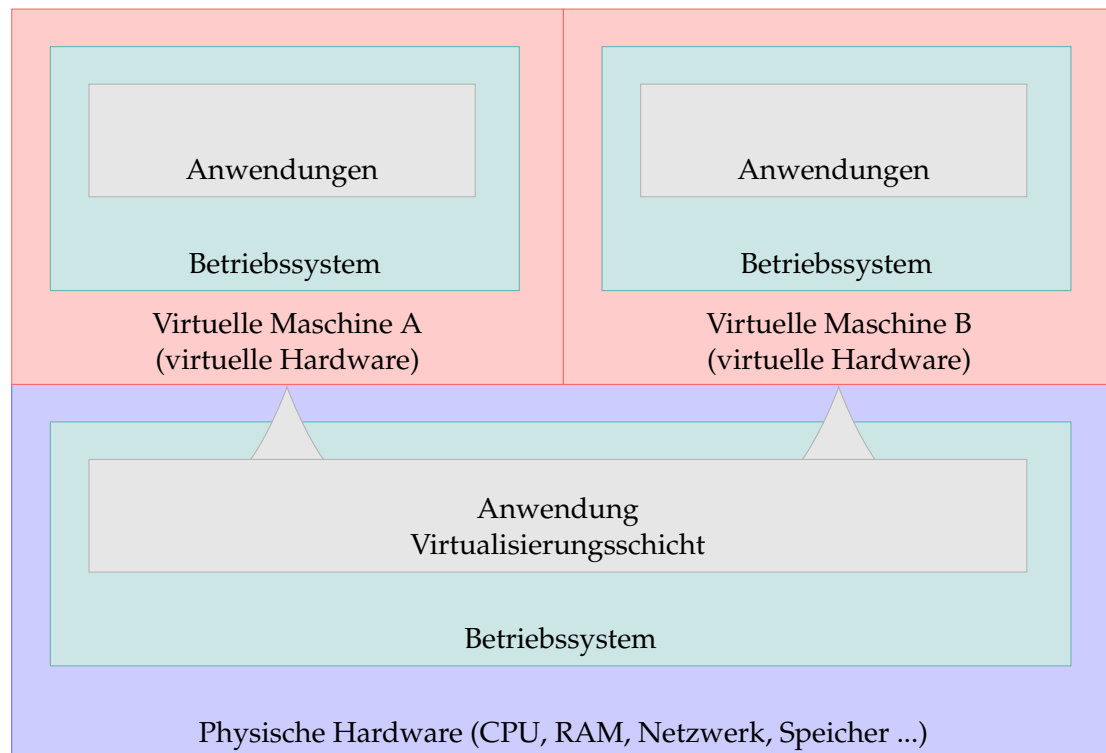


Abbildung 2.1: Die Schichten bis hin zu einer VM. Eine Virtualisierungslösung läuft auf physischer Hardware und einem Betriebssystem. Mit der Anwendung können VMs erzeugt werden, die wieder ein Betriebssystem und Anwendungen besitzen (in Anlehnung an [53]). Die untere Schicht ist die Host-Umgebung und die obere Schicht die Gast-Umgebung.

und für andere VMs zur Verfügung gestellt. So können die physischen Maschinen noch effizienter gestaltet und genutzt werden.

Neben der Virtualisierung ganzer Maschinen, können auch nur Anwendungen oder Netzwerke virtualisiert werden. Die unterschiedlichen Konzepte werden nachfolgend erklärt.

2.1.1 Virtualisierung auf Betriebssystemebene

Bei der Virtualisierung auf Betriebssystemebene können mehrere isolierte Container auf einem Kernel² gestartet werden. So können beispielsweise Programme isoliert ausgeführt werden. Allerdings kann nur mit einem Kernel gearbeitet werden, wodurch sich kein weiteres Betriebssystem starten lässt. Der Vorteil dieser Methode ist die beinahe

² Als Kernel wird der zentrale Bestandteil eines Betriebssystems bezeichnet, welcher zwischen Hard- und Software vermittelt [77].

native Performanz³ und Ressourcenverteilung. Parallels Virtuozzo Containers sind hier ein bekannter Name. [83]

2.1.2 Emulation

Emulatoren sind besonders im Bereich der Computerspiele und älterer Software sehr beliebt. Ein Emulator versucht die gesamte Architektur eines Systems, das heißt Prozessor, Speicher und BIOS⁴, nachzubilden. Die Entwicklung eines solchen Emulators ist aufwendig und kann oftmals nicht die Leistung des ursprünglichen Systems erreichen. Selten werden ganze Betriebssysteme virtualisiert, sondern vielmehr nur Anwendungen. Um eine Emulation umzusetzen kann auf Quick EMUlator (QEMU)⁵ zurückgegriffen werden, wobei dieser eine Ausnahme darstellt, da QEMU auch ganze Betriebssysteme virtualisieren kann. [70] [83]

2.1.3 Paravirtualisierung

Bei Paravirtualisierung handelt es sich um die Technik, die auch bei AWS und GCP angewendet wird. VMs teilen sich über eine abstrakte Verwaltungsschicht, Hypervisor oder Virtual-Machine-Monitor (VMM) genannt, die gemeinsamen Ressourcen des Wirts. Paravirtualisierung arbeitet hardwarenah und kann somit auf die vorhandene Hardware des Wirts zugreifen. Im besten Fall muss diese nicht virtualisiert werden. Es wird kein umfangreiches Wirtssystem benötigt und es ist insgesamt performanter. Jedoch sollte das Gastsystem an die Hardware angepasst werden. Oftmals wird bei der Paravirtualisierung Xen und Kernel-based Virtual Machine (KVM) genutzt. [83]

2.1.4 Virtuelle Maschine

Eine VM ist die Gesamtheit aus Hardware und Gastsystem bzw. Betriebssystem. Eine VM kann exportiert und auf einem neuen Wirt importiert werden. Damit kann plattformunabhängige Software betrieben werden, was unter anderem auch eine Anwendungsvirtualisierung möglich macht. VMs lassen sich in fast allen vorab genannten Diensten erstellen. [30]

3 Leistungsfähigkeit eines Rechners oder Systems.

4 Das Basic Input/Output System ist die Firmware bei x86-Rechnern. Diese Firmware enthält alle Grundfunktionen und leitet den Start des Betriebssystems ein.

5 QEMU ist eine freie Virtualisierungssoftware, die die gesamte Hardware eines Computers emuliert und durch die dynamische Übersetzung eine sehr gute Ausführungsgeschwindigkeit erreicht [70].

2.1.5 Virtualisierungslösung

Eine Virtualisierungslösung wird als eine Zusammenfassung aller Verfahren verstanden, um eine Maschine zu virtualisieren und zu betreiben. Moderne Prozessoren unterstützen beispielsweise AMD-V oder Intel VT⁶, welche für solche Produkte benötigt werden. VirtualBox nutzt beide Techniken gleichermaßen und lässt den Nutzer die Paravirtualisierung wählen. Das zeigt die Stärken von Virtualisierungslösungen, mit welchen alle Verfahren innerhalb einer Anwendung gebündelt und ausgewählt werden. [84]

Daneben existieren noch VMware und Parallels [83].

2.1.6 Speichervirtualisierung

Unter Speichervirtualisierung kann auch Datenspeicher verstanden werden, auf dem einfache Daten abgelegt werden können. Der Vorteil ist dabei, dass keine physischen Speicher mehr vorhanden sein müssen. Jedoch ist eine schnelle Internetverbindung bei größeren Daten unabdingbar. Für Datensicherungen und redundante Speicher kann eine Speichervirtualisierung ebenfalls verwendet werden. [17]

2.1.7 Netzwerkvirtualisierung

Unter Netzwerkvirtualisierung werden Virtual Local Area Networks (VLANs) verstanden, wobei zwischen virtualisierten Infrastrukturen ein virtuelles Netzwerk erstellt wird. So kann beispielsweise eine VM, welche als Server fungiert mit einem Speicher verbunden werden, worauf Backups durchgeführt werden. Ein VLAN ist nach außen hin isoliert und nur innerhalb des VLANs selbst sichtbar. Diese Isolation ermöglicht zwar eine hohe Sicherheit, da jedoch mehr Netzwerke zu verwalten sind, erhöht sich auch der Administrationsaufwand. [17]

Der Aufbau einer Netzwerkvirtualisierung ist vergleichbar mit der Virtualisierung in Abbildung 2.1, da dabei auch virtuelle Netzwerke erstellt werden (virtuelle Hardware).

6 Virtualisierungstechniken der Prozessoren der Hersteller.

2.2 Virtualbox

Neben der Installation in den AWS werden die privaten Cloud-Speicherdienste in einer VM betrieben und getestet. Dies soll den Leistungsvergleich um einen zusätzlichen Parameter erweitern, der auch Einblicke in die Performanz innerhalb eines privaten Netzwerks, z.B. in einem Unternehmen, gibt.

Virtualbox ist eine Virtualisierungslösung, die alle Komponenten einer IT-Infrastruktur zusammenfasst. Die Erzeugung einer neuen VM kann sowohl über ein Graphical User Interface (GUI)⁷ als auch über eine Konsole durchgeführt werden. Dabei wird die VM speziell für ein Linux-Derivat⁸ oder Windows erzeugt. Im weiteren Verlauf werden der Arbeits- und Massenspeicher definiert. Danach können durch die Einstellungen der VM die Prozessoranzahl, Hardware-Virtualisierung, der Grafikspeicher sowie Audio, Netzwerk und Peripherie bestimmt werden. Wenn eine Image-Datei in das virtuelle Laufwerk eingelegt wurde, kann von dieser gebootet und, wenn erwünscht, das gestartete Betriebssystem installiert werden.

Existiert ein Betriebssystem auf der erzeugten VM, können die verschiedenen Cloud-Speicherdienste darauf installiert und konfiguriert werden. Durch einen Host-only-Adapter wird ein Netzwerk erstellt, das zwischen dem Gast- und Wirt-System kommuniziert. Das erste Interface trägt meist den Namen vboxnet0 und konfiguriert die IP-Adresse, worüber der Gast angesprochen werden kann, sowie auch das DHCP⁹ innerhalb von Virtualbox. Damit ist es möglich, die VM auch auf dem Wirt oder sogar über das World Wide Web (WWW) anzusprechen, was auch notwendig ist, da die Cloud-Speicherdienste wie in den AWS getestet werden sollen. Das hat den Vorteil, dass auch Leistungsdaten zu den Cloud-Speicherdiensten innerhalb eines privaten Netzwerks vorhanden sind.

2.3 Service-orientierte Architekturen

Neben der Virtualisierung sind Service-orientierte Architekturen (SOA) und Web-Services wichtige Komponenten, um Cloud Computing zu betreiben.

⁷ Grafische Benutzeroberfläche oder auch Benutzerschnittstelle, welche die Bedienung durch grafische Symbole oder Steuerelemente ermöglicht.

⁸ Derivate sind Abstammungen von den ursprünglichen Linux-Distributionen Debian, Slackware, RedHat und Arch.

⁹ Das Dynamic Host Configuration Protocol (DHCP) ermöglicht die Zuweisung der Netzwerkkonfiguration an Clients durch einen Server [36].

Service-orientierte Architekturen bieten voneinander unabhängige Dienste (Services) an, was bedeutet, dass sie flexibel gebunden oder lose gekoppelt kommunizieren und Nachrichten austauschen können. Beim Cloud Computing handelt es sich um IT-Infrastrukturen, Plattformen und Anwendungen, welche als Dienste angeboten werden. Diese Dienste werden über Web-Protokolle und Schnittstellen angesprochen, insbesondere *Web Services* und *RESTful Services*. [21]

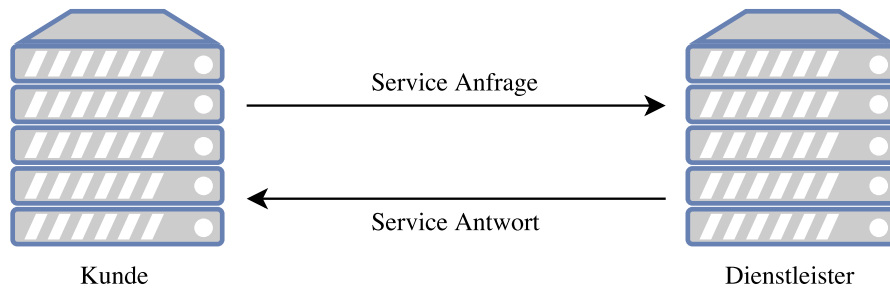


Abbildung 2.2: Simpler Aufbau einer Service-orientierten Architektur (in Anlehnung an [13])

In Abbildung 2.2 ist vereinfacht dargestellt, wie eine SOA funktioniert. Es gibt einen Endkunden, der eine Anfrage an einen Service bei einem Anbieter stellt. Der Anbieter verarbeitet diese Anfrage und sendet eine Antwort zurück an den Endkunden. Dieser kann mit der Antwort seinen eigenen Dienst betreiben bzw. eine entsprechende Ausgabe (dynamisch) generieren.

2.4 Web-Services

Im Gegensatz zu verteilten Ressourcen innerhalb eines lokalen Netzwerkes, gibt es im Cloud Computing einige Herausforderungen bei der Datenübertragung. Diese kann, in Abhängigkeit des Standorts des Antwort-Servers, langsam und/oder instabil sein. Da solche Probleme vermieden werden sollen, sollte hierbei auf eine schwach gekoppelte, asynchrone und nachrichtenbasierte Kommunikation über Web Services zurückgegriffen werden.

Web Services sind ein Standard, welcher genutzt wird, um die Nachrichten zu formatieren und zu bearbeiten. Am weitesten verbreitet dafür sind SOAP/WSDL-basierte Web Services und RESTful Services. Simple Object Access Protocol (SOAP)¹⁰ ist ein Messaging-Protokoll und Web Services Description Language (WSDL) ist eine Schnittstellenbeschreibungssprache. Representational State Transfer (REST) baut dagegen auf

¹⁰ SOAP ist seit Version 1.2 eine eigenständige Bezeichnung und kein Akronym mehr [85].

das Hypertext Transfer Protocol (HTTP) auf und beschreibt einen Architekturstil. RESTful Services werden lediglich über die uniforme HTTP-Schnittstelle angesprochen. Beide Ansätze identifizieren Dienste anhand von Uniform Resource Identifiers (URI)¹¹. [21]

REST wird, im Gegensatz zu SOAP, welches Extensible Markup Language (XML)-Dokumente zur Nachrichtenübertragung nutzt, über HTTP-Anfragemethoden angesprochen. Nach [63] existieren insgesamt neun Anfragemethoden. Die vier wichtigsten Methoden sind GET, POST, PUT und DELETE, um sämtliche Funktionen ausführen zu können [17].

¹¹ Eine Zeichenfolge, die zur Identifizierung einer abstrakten oder physischen Ressource dient.

3 Cloud Computing

Mit Cloud Computing werden jegliche Ressourcen, die eine IT-Infrastruktur bietet, durch das Internet abgebildet. Hard- und Software werden in einem Rechenzentrum verwaltet und zur Verfügung gestellt. Die Administration der zugrundeliegenden Infrastruktur obliegt dabei dem Anbieter, bei welchem Rechenkapazitäten angemietet werden. Der Nutzer kann über Web-Dienste (Web Services) mit einem Browser auf diese Infrastruktur zugreifen und beispielsweise eine VM oder ein VLAN erstellen. Mittlerweile haben sich zwei Abrechnungsmodelle etabliert, welche oftmals bereits beim Bestellprozess gewählt werden können. Einerseits existiert die monatliche Abrechnung, bei welcher zu Monatsbeginn ein fixer Betrag abgebucht wird. Auf der anderen Seite kann auch eine stündliche Abrechnung gebucht werden, die sich nach der Nutzung und dem Ressourcenverbrauch richtet. [18]

Nach [21] wird Cloud Computing folgendermaßen definiert:

„Unter Ausnutzung virtualisierter Rechen- und Speicherressourcen und moderner Web-Technologien stellt Cloud Computing skalierbare, netzwerk-zentrierte, abstrahierte IT-Infrastrukturen, Plattformen und Anwendungen als on-demand Dienste zur Verfügung. Die Abrechnung dieser Dienste erfolgt nutzungsabhängig.“

3.1 Bereitstellungsmodelle

Cloud Computing kann in verschiedene Arten der Bereitstellung unterteilt werden. Dabei wird zwischen der Verfügbarkeit und Nutzung unterschieden. Die öffentliche, private und hybride Cloud haben sich am Markt etabliert und werden nachfolgend erläutert.

3.1.1 Öffentliche (public) Cloud

Bei der public Cloud, auch öffentliche Cloud genannt, handelt es sich um ein Modell, welches für jeden frei zugänglich ist. Das bedeutet, dass dieser Dienst über das Web

genutzt wird und Interessenten diesen kostenfrei kennen lernen können. Bekannte Anbieter sind z.B. AWS, MSA und GCP, welche in Kapitel 5.1 genauer beschrieben werden. [57]

3.1.2 Private (private) Cloud

Die private Cloud ist, im Gegensatz zur öffentlichen Cloud, nicht jedem frei zugänglich. Hierbei handelt es sich um private Infrastrukturen, die z.B. zu einem Unternehmen gehören, worüber, aus Gründen des Datenschutzes und der Sicherheit, diverse Dienste laufen. Diese Dienste werden zumeist nur über das interne Netzwerk oder nur bestimmten Personengruppen verfügbar gemacht. [57]

Mit Hilfe einer privaten Cloud kann eine IT-Infrastruktur geschaffen werden, die ähnlich zu den AWS ist. Besonders populär sind Open Source Anbieter, wie z.B. Minio, Open-Stack und Scalify CS, welche in Kapitel 5.2 genauer erläutert werden.

3.1.3 Hybride (hybrid) Cloud

Hybrid bedeutet, dass sowohl eine offene (public) als auch private Cloud genutzt wird. Es ist also eine Kombination von beidem. So können z.B. Webapplikationen und bestimmte Dienste über die offene Cloud und datenschutzkritische Anwendungen über die private Cloud betrieben werden. Wichtig ist hier die konsequente Trennung der beiden Geschäftsprozesse. [57]

Mit einer hybriden Cloud können äußerst flexibel viele verschiedene Dienste betrieben werden. Außerdem können so die Kosten innerhalb der offenen Cloud reduziert werden, da nicht alle Dienste dort betrieben werden.

Lösungen für die Eingliederung der privaten Cloud in eine offene Cloud bieten Microsoft und Cisco an [59] [28].

3.1.4 Gemeinschaftliche (Community) Cloud

Die Community Cloud, oder auch gemeinschaftliche Cloud, ist das neuste Modell um eine Cloud bereitzustellen. Dahinter verbirgt sich ein Zusammenschluss mehrerer privater Clouds zu einer großen Cloud, die aber weiterhin nur den Benutzern zugänglich ist, die auch vorher einen Zugang hatten. Der Vorteil eines Zusammenschlusses ist die Bündelung der Ressourcen, Verfügbarkeit und Sicherheit. Wenn mehrere private Clouds,

z.B. von vielen Unternehmen, zusammengeschlossen werden, hat die resultierende Cloud die Ressourcen aller Rechenzentren zur Verfügung. Außerdem können die Daten verteilt gespeichert werden, was die Sicherheit erhöht und für mehr Redundanz sorgt. [57]

Besonders für kleine Unternehmen, die kein eigenes Rechenzentrum besitzen, kann sich die Teilnahme an einer Community Cloud lohnen [31].

3.2 Everything as a Service (XaaS)

Es existieren im Bereich Cloud Computing unzählige Begriffe. Meistens haben diese eine ähnliche Bedeutung und sorgen für Verwirrung. Unter *Everything as a Service*, kurz XaaS oder EaaS, wird die Bereitstellung von jeglichen Diensten in der Cloud als Service verstanden. Dies bedeutet, dass die Infrastruktur, Plattform und Software nicht mehr vor Ort betrieben werden, sondern all dies in die Cloud ausgelagert und überwiegend im Browser gearbeitet wird. Wie diese Schichten zusammenhängen, wird mit Abbildung 3.1 veranschaulicht. [54]

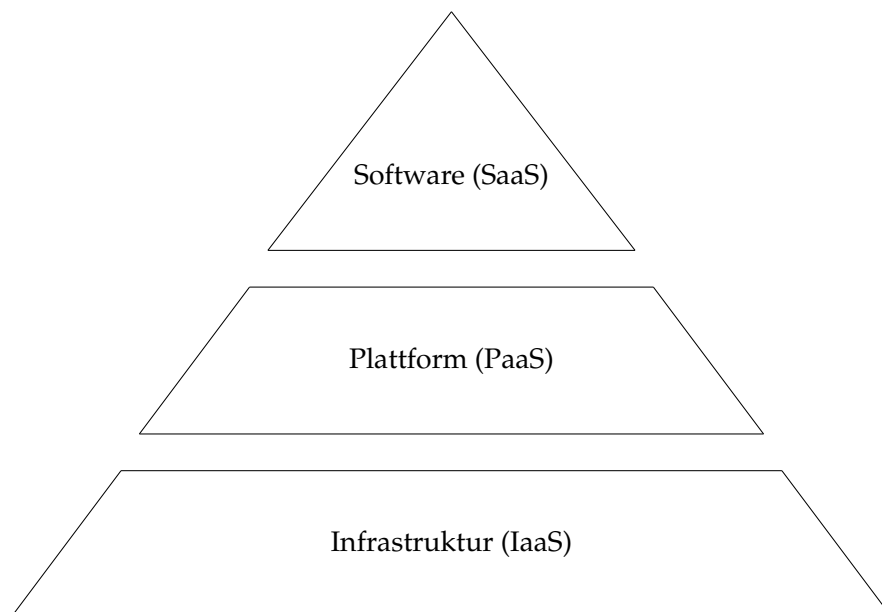


Abbildung 3.1: Aufbau von Everything as a Service, der sehr gut aufzeigt, dass die Spitze, die Software, nicht ohne die anderen Unterschichten lauffähig ist (in Anlehnung an [78]).

3.2.1 Infrastructure as a Service (IaaS)

Mit Infrastructure as a Service, kurz IaaS, wird das Grundgerüst für Cloud Computing zur Verfügung gestellt. Es schafft eine IT-Infrastruktur mit Netzwerken, Rechnern (virtuelle oder dedizierte) und Speicherlösungen. Diese Infrastrukturen bieten ein hohes Maß an Flexibilität und können ganz individuell die vorhandenen Ressourcen verwalten. Dabei sind sie von der Struktur so aufgebaut, wie Administratoren und Entwickler sie bereits kennen. [7]

Beispiele hierfür sind AWS, MSA und GCP.

3.2.2 Plattform as a Service (PaaS)

Durch Plattform as a Service, kurz PaaS, fällt keine Verwaltung der zugrundeliegenden Infrastruktur an [7]. Der Dienstleister der Infrastruktur ist für den Betrieb zuständig und stellt eine Plattform zur Administration der Systeme bereit. Im Gegensatz zu IaaS können hier keine VMs erstellt werden. Mit PaaS können diese z.B. gestartet und gestoppt sowie das Betriebssystem neu aufgesetzt werden.

SAP HANA Cloud Platform und die Google App Engine sind bekannte Beispiele.

3.2.3 Software as a Service (SaaS)

Software as a Service bietet ein vollständiges Produkt an, welches alle Schichten (Abbildung 3.1) enthält [7]. Im Mittelpunkt steht die Software und um alles andere kümmert sich in diesem Fall der Anbieter. Dieser stellt, meist automatisiert, eine (virtuelle) Maschine zur Verfügung und bietet die Software über eine Plattform an. Dies bedeutet auch, dass die Software nicht mehr lokal installiert wird und keine Kontrolle über die Daten besteht.

Derzeit ziehen immer mehr Anbieter ihre Software in die Cloud um. Mit Microsoft Office 365 [60] und Adobe Creative Cloud [1] als Beispiel existieren zwei Anbieter, die ein hybrides Modell nutzen. Neben einer lokalen Installation wird eine ständige Internetverbindung benötigt, um die Software mit einem Lizenzserver zu überprüfen. Die G Suite (vormals Google Apps) [41] ist beispielsweise ein reines SaaS.

3.2.4 Weitere Ansätze

Neben den etablierten as-a-service-Ansätzen existieren noch weitere, die jedoch noch nicht so bekannt sind, wie die zuvor genannten.

High Performance Computing as a Service (HPCaaS)

Mit diesem Ansatz sollen Hochleistungsrechner bzw. wissenschaftliche Anwendungen aus dem Grid Computing als Dienst zur Verfügung gestellt werden. [87]

Data Intensive Computing as a Service (DICaaS)

Ähnlich wie HPCaaS sollen auch mit DICaaS wissenschaftliche Aufgaben in der Cloud bewältigt werden. In diesem Fall handelt es sich jedoch mehr um die Verarbeitung und Speicherung von Daten im Petabyte-Bereich. [87]

Humans as a Service (HuaaS)

Mit HuaaS wird versucht, menschliche Intelligenz als Dienst abzubilden. Bei Anwendungen wie z.B. Bilderkennung sind Rechner oftmals langsam und die Erkennungsrate kann durch fehlende Ressourcen nicht in Echtzeit stattfinden. Deswegen wird darauf gesetzt, dass möglichst viele Menschen ihre Dienste anbieten und an solchen Problematiken teilnehmen. Für HuaaS gibt es Marktplätze, auf welchen Anbieter und Konsumenten zusammentreffen. [87]

Unbekanntere Ansätze

Es existieren noch einige weitere Ansätze, die weniger bekannt sind bzw. weniger genutzt werden. Dabei handelt es sich überwiegend um Serverdienste, wie z.B.

- Backup as a Service (BaaS),
- Desktop as a Service (DaaS),
- Load Balancer as a Service (LBaaS),
- Security as a Service,
- Storage as a Service und

- Database as a Service (DBaaS),

welche in die Cloud ausgelagert werden. [87]

3.3 Amazon Web Services (AWS)

Alle Cloud-Speicherdienste werden innerhalb der Infrastruktur der Amazon Web Services installiert und getestet. So kann für eine gleichbleibende Leistung und für gleiche Bedingungen gesorgt werden. Damit die Netzwerkzeiten möglichst gering bleiben, kann eine Region der Server gewählt werden. Die nächste Region liegt in Frankfurt und nennt sich `eu-central-1`. Es existieren Standorte überall auf der Welt verteilt (siehe Tabelle 3.1).

Tabelle 3.1: Standorte der Rechenzentren der AWS [10].

Code	Name
us-east-1	US East (N. Virginia)
us-east-2	US East (Ohio)
us-west-1	US West (N. California)
us-west-2	US West (Oregon)
ca-central-1	Canada (Central)
eu-west-1	EU (Ireland)
eu-central-1	EU (Frankfurt)
eu-west-2	EU (London)
ap-northeast-1	Asia Pacific (Tokyo)
ap-northeast-2	Asia Pacific (Seoul)
ap-southeast-1	Asia Pacific (Singapore)
ap-southeast-2	Asia Pacific (Sydney)
ap-south-1	Asia Pacific (Mumbai)
sa-east-1	South America (São Paulo)

Mit Hilfe der AWS werden hochverfügbare virtuelle Infrastrukturen erstellt und verwaltet. So ist es unter anderem möglich, ein vollständiges Netzwerk einzurichten und Rechte zu vergeben. Mit der Elastic Compute Cloud (EC2) ist es möglich, VMs nach eigenen Vorgaben zu erstellen. Der S3 dient zum Speichern von Objektdaten und ist skalierbar. Wenn mehr Speicher benötigt wird, kann dieser erweitert werden. Mit einer

Virtual Private Cloud (VPC) kann ein isolierter Bereich erstellt werden, welcher als VLAN fungiert.

Mit den AWS kann eine vollständige IT-Infrastruktur innerhalb des Internets abgebildet werden.

3.3.1 Elastic Compute Cloud (EC2)

Mit der EC2 können sichere, skalierbare Rechenkapazitäten, auch Instanzen genannt, in der Cloud erstellt werden. Durch eine einfache Weboberfläche können VMs eingerichtet und konfiguriert werden. Die Bereitstellung einer neuen Instanz nimmt nur wenige Minuten in Anspruch. Durch die Skalierbarkeit der EC2 ist es möglich, die Kapazitäten der Vorhaben entsprechend anzupassen. Kosten fallen nur dann an, wenn die Instanz läuft bzw. genutzt wird. Im ausgeschalteten Zustand werden die Ressourcen freigegeben und sie können anderweitig verwendet werden. [2]

Zu Beginn einer neuen Instanz wird ein Betriebssystem, welches sich Amazon Machine Image (AMI) nennt, ausgewählt. Dieses ist speziell an die AWS angepasst und besitzt oftmals auch Softwarequellen direkt von Amazons Servern. Verfügbar sind alle gängigen wie Debian, Ubuntu, SUSE Linux Enterprise Server (SLES), CentOS bzw. RedHat, Amazon Linux und Microsoft Windows Server 2003 R2, 2008, 2008 R2, 2012 und 2012 R2. Bei der Wahl des Betriebssystems sollte auf die Anforderungen und eventuell gegebene Vorgaben des zukünftigen Projekts geachtet werden.

Anschließend wird der Instanz-Typ bestimmt. Innerhalb der einjährigen Testphase der AWS können Instanzen vom Typen `t2.micro`, welche sich besonders für Testumgebungen eignen, kostenfrei betrieben werden. Diese Instanz besitzt einen virtuellen Prozessor und 1 Gigabyte (GB) Arbeitsspeicher. Diese Ressourcen sind für die meisten Installationen ausreichend, da diese nur zu Testzwecken betrieben werden und wenige Datenübertragungen entstehen. Im freien Kontingent sind bis zu 30 GB Elastic Block Store (EBS)-Speicher pro Instanz enthalten.

Neben dem genannten Instanz-Typ gibt es noch viele weitere (siehe Tabelle 3.2). Ein Spitzenmodell der T2-Generation hat 8 virtuelle Prozessoren, 32 GB Arbeitsspeicher und trägt die Bezeichnung `t2.2xlarge`. Es existieren Instanz-Typen für die unterschiedlichsten Anwendungsfälle, welche z.B. RAM-, speicher- oder hochleistungsoptimiert sind. Der EBS wird für die skalierbare Speicherung eingesetzt und bedeutet, dass die Speichergröße bis zu einem gewissen Wert selbst bestimmt werden kann. Die Netzwerkleistung gibt Aufschluss darüber, welche Priorität die Maschinen im Vergleich haben oder ob sie sogar eine eigene Gigabit-Leitung besitzen.

Tabelle 3.2: Auswahl verfügbarer Instanz-Typen und welche Ressourcen sie zu bieten haben. Anhand der Ressourcenverteilung ist der vorgesehene Einsatzzweck ersichtlich. [3]

Instanz-Typ	vCPU	Arbeitsspeicher [GB]	Speicherung	Netzwerkleistung
t2.nano	1	0.5	Nur EBS ¹	Niedrig
t2.micro	1	1	Nur EBS	Gering bis mittel
m4.10xlarge	40	160	Nur EBS	10 Gigabit
m3.medium	1	3.75	1 x 32 SSD	Mittel
c4.4xlarge	16	30	Nur EBS	Hoch
c3.large	2	3.75	2 x 16 SSD	Mittel
p2.8xlarge	32	488	Nur EBS	10 Gigabit
x1.32xlarge	128	1952	2 x 1920 SSD	20 Gigabit

Die Sicherheit der Daten und der Systeme ist insbesondere in der Cloud bzw. im WWW ein wichtiger Aspekt. Mit sogenannten Sicherheitsgruppen (engl. Security Groups) können Ports geöffnet oder geschlossen werden. Diese funktionieren, ähnlich einer Firewall, auf Betriebssystem-Ebene (z.B. iptables unter Linux).

Bei der Generierung einer neuen Instanz kann auch eine zur Instanz gehörende Sicherheitsgruppe erstellt werden, bei welcher vorerst nur auf dem geöffneten Port 22/tcp², dem Secure Shell (SSH)-Server, gelauscht wird³. Dies bedeutet im Umkehrschluss, dass Ports, auf welchen ein Dienst lauscht, explizit geöffnet werden müssen. Der Vorteil dieser Regelung ist die kleine Angriffsfläche für potentielle Gefahren, die über Ports in das System eindringen möchten.

Instanzen der AWS werden per Kommandozeile und SSH-Protokoll administriert. Das bedeutet, dass sich der Benutzer über eine verschlüsselte Netzwerkverbindung direkt mit dem Server verbindet, wobei anstelle eines Passworts ein SSH-Schlüssel zum Anmelden benutzt wird. Dieses Schlüsselpaar besteht aus einem privaten und einem öffentlichen Teil. Diese werden im letzten Schritt der Instanz-Initialisierung generiert, wobei der private Schlüssel lokal gesichert werden muss. Dies ist eine weitere Sicherheitsmaßnahme, wodurch Angreifern die Anmeldung am Server erschwert wird. Während der automatischen Installation des Betriebssystems wird der öffentliche SSH-Schlüssel auf der Instanz abgelegt. Nur wer im Besitz des privaten Schlüssels ist, kann sich mit

¹ Der Elastic Block Store wurde für die persistente Blockspeicherung entwickelt [4].

² Das Transmission Control Protocol (TCP) und User Datagram Protocol (UDP) sind Netzwerkprotokolle und besitzen jeweils 65535 solcher Ports [82].

³ Geöffnete Ports auf einem Server „lauschen“ auf eine Verbindung von Außen.

der Instanz verbinden und anmelden.

Zuletzt wird noch eine Zusammenfassung der Instanz angezeigt, welche anschließend gestartet wird. Wenn bereits eine Instanz erstellt wurde, kann eine ähnliche Instanz erstellt werden, ohne dass alle Schritte der Generierung erneut durchgeführt werden müssen.

3.3.2 Simple Storage Service (S3)

Neben der EC2 gibt es den S3 für die hochverfügbare und skalierbare Speicherung von Objektdaten. Dieser Dienst bietet die Möglichkeit per REST-API⁴ sowie SDK⁵ die Integration in andere Applikationen von Drittherstellern. In den vorhandenen Regionen können Sicherungen angelegt werden, wodurch diese an verteilten Orten vorgehalten werden. Der S3 ist so „simpel“, weil sich dieser an den Bedarf und die Aufgabe anpasst und der Speicherplatz beliebig erweitert oder verringert werden kann.

Mit dem Secure Sockets Layer (SSL) wird die Datenübertragung in den meisten Fällen geschützt, wobei viele Sicherheitsprodukte diesen Effekt aushebeln können [24]. Daneben können mit Bucket⁶-Richtlinien die Berechtigungen verwaltet und der Zugriff per Identity and Access Management (IAM)⁷ kontrolliert werden.

Um mit dem Dienst zu kommunizieren bzw. Daten zu übertragen werden verschiedene Werkzeuge, wie z.B. über die Kommandozeile (rsync, s3-sync und Amazon Glacier) oder mit Direct Connect bzw. Snowball von Amazon, angeboten. Erstere eignen sich für Daten, welche schnell synchronisiert werden sollen und sich im einstelligen GB-Bereich befinden. Zweitere dagegen stellen entweder eine direkte Verbindung zwischen dem aktuellen Standort und den AWS her oder sind für Daten im Petabyte-Bereich gedacht [8]. [5]

Der S3 wird ebenfalls konfiguriert und anschließend dessen Leistungsfähigkeit bewertet, was in Kapitel 5 genauer beschrieben wird.

4 Das Application Programming Interface, oder Programmierschnittstelle, dient der Anbindung von Programmen an ein anderes Programm oder System.

5 Ein Software Development Kit ist eine Zusammenstellung von Programmen, wodurch die Entwicklung vereinfacht wird.

6 Buckets, deutsch Eimer, sind die „Behälter“ für die Daten innerhalb des S3.

7 Mit IAM wird der Zugriff auf AWS-Dienste und -Ressourcen für teilnehmende Benutzer gesteuert.

3.3.3 Virtual Private Cloud (VPC)

Der Name Virtual Private Cloud ist an dieser Stelle etwas irreführend, da es sich dabei um die Bereitstellung eines Netzwerks bzw. eines isolierten Bereichs innerhalb der AWS handelt. Die einzelnen Instanzen können nicht untereinander kommunizieren oder Daten austauschen, da sie sich nicht in einem gemeinsamen Netzwerk befinden. Mit Hilfe der VPC kann ein Netzwerk erstellt werden, worüber Instanzen zusammengeschlossen, IP-Adressbereiche festgelegt, Subnetze erstellt und Berechtigungen vergeben werden können.

Beispielsweise kann ein Subnetz für Webserver und Applikationen erstellt werden, die aus dem Internet erreichbar sein sollen. Ein anderes Subnetz verwaltet die Datenbanken und sensible Daten, welches nur intern erreichbar ist und keinen Internetzugriff besitzt. Durch Sicherheitsgruppen und Netzwerk-Zugriffskontrolllisten können die Zugriffe und Berechtigungen innerhalb der Subnetze und des gesamten Netzwerks komplett gesteuert werden.

Es ist auch möglich, eine Hardware-VPN⁸-Verbindung zu einem VPC einzurichten, womit von außen auch auf das interne Subnetz zugegriffen werden kann. [6]

3.4 Microsoft Azure und Google Cloud Plattform

MSA und die GCP gelten als große Konkurrenten zu den AWS, was vor allem an den ähnlichen Diensten liegt, welche angeboten werden. Die drei Anbieter haben einen Dienst zum Erzeugen von VMs, einen oder mehrere Dienste zum Speichern von großen Datenmengen und einen VPC-Dienst, womit VLANs erstellt und verwaltet werden können. [42][58]

Unterschiede bestehen lediglich in der Einrichtung und Konfiguration der einzelnen Dienste. Die Compute Engine (CE) von Google, in der VMs erzeugt werden, führt alles in einem einzigen Schritt durch, wo die EC2 mehrere Schritte benötigt. MSA geht einen ganz anderen Weg und erweitert alle Optionen horizontal, wodurch die Oberfläche sehr oft verschoben werden muss. Der Leistungsumfang ist meist der gleiche. Alle Anbieter haben noch viele weitere Produkte im Angebot, welche jedoch nicht relevant für diese Arbeit sind.

⁸ Ein Virtual Private Network ist eine Technik, die es erlaubt, von überall auf der Welt auf Ressourcen innerhalb eines privaten Netzwerks zuzugreifen.

Der GCS und der ABS von Microsoft sind Objektspeicher und vergleichbar mit dem S3. Beide Dienste werden, wie der S3, bei dem Leistungsvergleich getestet und bewertet. Somit ist es möglich eine Aussage darüber zu treffen, welche öffentliche Cloud, zumindest beim Storage, am leistungsfähigsten ist.

4 Möglichkeiten der Leistungsuntersuchung

Es gibt verschiedene Möglichkeiten, um die Leistungsfähigkeit von Speicherdiensten in der Cloud zu messen. Die einfachste und aussagekräftigste Methode ist die Messung der Zeiten des Hoch- und Herunterladens sowie der Zeit, welche zur Erstellung eines Buckets bzw. Containers für die Daten benötigt wird. Zusätzlich wird, zur Überprüfung der Datenintegrität, eine MD5-Prüfsumme erstellt, über die feststellbar ist, ob die Daten fehlerfrei und unverändert sind.

Um all dies durchzuführen, ist es sinnvoll, auf diverse Bibliotheken oder Skripte zurückzugreifen, die genau diese Aktionen unterstützen. Dabei wurde der derzeitige Markt untersucht, um aufzuzeigen, womit Leistungstests durchgeführt werden können.

4.1 Bibliotheken und Programme

Mit Bibliotheken sind Module gemeint, die das Hauptprogramm um weitere Funktionen erweitern und auf die zurückgegriffen werden kann. Solche Bibliotheken gibt es für fast jede Programmiersprache, jedoch hat sich Python, durch den geringen Schwierigkeitsgrad, im wissenschaftlichen Bereich besonders etabliert [43]. Aus diesem Grund existieren eine große Anzahl an Python-Bibliotheken für den Bereich Data Science, womit Daten analysiert und ausgewertet werden können.

Auch für die Kommunikation mit Cloud-Diensten gibt es Bibliotheken, welche jedoch weniger zum Testen geeignet sind. Besonders prominent positioniert sich hier `boto3`, welches speziell für die Infrastrukturen von Amazon entwickelt wurde. Das heißt mit Hilfe dieser Bibliothek wird die Kommunikation mit der EC2 oder dem S3 erleichtert und kann in die unterschiedlichsten Programme eingebaut werden.

Eine andere Bibliothek, welche über 50 verschiedene Cloud-Anbieter unterstützt, ist `libcloud` von Apache. Damit könnte ein Großteil der Clouds angesprochen werden, wobei nur diese eine Bibliothek nötig wäre. Der Nachteil dabei ist jedoch, dass diese

Bibliothek nicht für S3-Dienste ausgelegt ist und nur solche unterstützt werden, die VMs erstellen, wie z.B. EC2 oder CE.

Ein bekanntes Kommandozeilenprogramm, welches stetig weiterentwickelt und wieder verwendet wird, nennt sich `s3cmd`. Ursprünglich geschaffen, um mit dem S3-Dienst von Amazon zu kommunizieren, wird es heute in vielen verschiedenen Projekten genutzt. Durch die Tatsache, dass der Quellcode frei verfügbar ist, kann es beliebig verändert werden, was auch die Möglichkeit offen lässt, weitere Anbieter hinzuzufügen bzw. zu unterstützen.

4.2 Skripte

Skripte dienen in erster Linie der Automatisierung von Befehlen, die auch manuell ausgeführt werden können. Sie basieren auf einer Skriptsprache, welche auch einen Interpreter benötigt, um sie ausführen zu können. Bekannte Interpreter sind in der Unix-Welt `bash` (`skript.sh`) und in der Windows-Welt `PowerShell` (`skript.ps`). Diese sind in den meisten Fällen beim Betriebssystem mit dabei und müssen nicht zusätzlich installiert werden.

Besonders unter Unix bzw. Linux werden solche Skripte gerne verwendet, da diese Betriebssysteme sehr oft über die Konsole bedient werden und damit viel Arbeit erleichtert werden kann. Zur Erzeugung eines Skripts wird eine Datei erstellt, welche mehrere Befehle enthält und diese nacheinander ausführt.

Ein einfaches Skript, um ein debianartiges Linux-Betriebssystem auf den aktuellen Stand zu bringen, könnte wie im Quellcode 4.1 aussehen.

Quellcode 4.1: Einfaches Update-Skript für debianartige Betriebssysteme. Der Eintrag in Zeile 1 gibt Auskunft darüber, mit welchem Interpreter dieses Skript gelesen werden soll.

```
1 #!/bin/bash
2
3 sudo apt update
4 sudo apt dist-upgrade
5 sudo apt autoremove
6 sudo apt autoclean
```

4.2.1 S3 Performance Test Tool

Das S3 Performance Test Tool ist ein in Java geschriebenes Test-Werkzeug womit der S3 von Amazon und ähnliche S3-Dienste getestet werden können. [48]

Um das Hochladen von n generierten zufälligen Daten mit der Größe von 2 kB auszuführen, ist nachfolgender Befehl notwendig:

```
$ java -jar target/s3pt.jar --accessKey <accessKey> --secretKey <secretKey> --bucketName <bucketName> -n <number of files to upload> -s 2048
```

Dadurch, dass das Skript auf Java basiert, ist auch ein Java-Interpreter auf dem System notwendig, wodurch vergleichsweise mehr Ressourcen benötigt werden, als das z.B. bei Systemwerkzeugen der Fall ist. Weiterhin wurde das Skript zuletzt im September 2016 (Stand: 14. August 2017) aktualisiert und unterstützt nur wenige Dienste. [48]

Weitere Dienste müssten manuell hinzugefügt werden, damit die Gesamtheit der geplanten S3-Dienste getestet werden kann.

4.2.2 s3perf

Ein Skript, welches automatisiert Daten hoch- und herunterlädt sowie diese auf Datenintegrität hin untersucht, wurde bereits von Baun [20] entwickelt und wird zur Messung der Leistungsfähigkeit in dieser Arbeit genutzt. Das Skript trägt den Namen *s3perf* und steht unter der Open-Source-Lizenz GNU General Public License (GPL) in Version 3 oder später. Bisher unterstützt dieses Skript zehn Cloud Anbieter (Stand: 14. August 2017), wobei einer der großen öffentlichen Cloud-Anbieter, Microsoft, noch fehlt. Dadurch, dass der Code frei verfügbar ist, kann der ABS von Microsoft hinzugefügt werden.

Um das Skript überhaupt nutzen zu können, müssen *s3cmd* und andere Abhängigkeiten, wie z.B. GNU Parallel, installiert sein. Das bedeutet, dass das Skript selbst auf andere Programme bzw. Funktionen zurückgreift und diese nur innerhalb dieser Datei bündelt.

Es ist sinnvoll die aktuellste Version von *s3cmd* zu installieren, da diese die meisten Funktionen unterstützt und mit den meisten Speicherdiensten funktioniert. Um *s3cmd* installieren zu können, sollten zuvor einige Abhängigkeiten installiert werden:

```
& sudo apt install python-setuptools unzip
```


Anschließend wird die aktuelle Version 2.0.0 heruntergeladen, entpackt und installiert.

```
$ wget https://downloads.sourceforge.net/project/s3tools/s3cmd/2.0.0/s3cmd-2.0.0.zip?r=&ts=1501428064&use_mirror=vorboss
$ unzip s3cmd-2.0.0.zip
$ cd s3cmd-2.0.0
$ sudo python setup.py install
```

Über die Paketquellen von Ubuntu kann nur die veraltete Version 1.6.x installiert werden. Über den Download des Quellcodes wird die aktuellste Version installiert.

Angenommen der S3 von Amazon soll getestet werden und die Zugangsdaten wurden bereits hinterlegt, kann mit dem Befehl

```
$ ./s3perf.sh -n 5 -s 1048576 -p
```

das Skript ausgeführt werden. Der Parameter *n* steht dabei für die Anzahl und *s* für die Größe der Dateien in Byte. Mit *p* wird das Skript angewiesen, alle Aktionen parallel auszuführen. Es werden insgesamt fünf Schritte durchgeführt.

1. Erzeugen von *n* zufällig gefüllten .txt-Dateien einer Größe zwischen 1 Byte und 16 Megabyte.
2. Erstellen eines Buckets oder Containers, je nachdem welcher Dienst genutzt wird.
3. Anschließend werden eine MD5-Prüfsumme erzeugt und die eben erstellten Daten in den Bucket bzw. Container hochgeladen. Alternativ können die Daten parallel hochgeladen werden.
4. Danach werden die Dateien wieder heruntergeladen, wobei mit der MD5-Prüfsumme die Daten auf Fehler und Veränderungen überprüft werden. Alternativ ist es möglich, die Daten parallel herunterzuladen.
5. Im letzten Schritt werden die erzeugten Dateien und der erstellte Bucket bzw. Container wieder gelöscht.

Bei allen Schritten wird gemessen, wie viel Zeit in Sekunden die jeweiligen Aktionen in Anspruch genommen haben. Am Ende wird eine Zusammenfassung ausgegeben, welche auch als Datei exportiert werden kann.

Wie die einzelnen Befehle im Detail aussehen und wie das Skript selbst aufgebaut ist, wird in Kapitel 7 und im Anhang A dargestellt.

4.2.3 Weitere Skripte

Es existieren, besonders auf Github (Suche nach „s3 performance“), viele weitere Skripte, welche die Leistungsfähigkeit von S3-Speicherdiensten testen. Jedoch sind diese entweder veraltet oder unterstützen nur Amazon und wenige weitere Dienste. Ein Werkzeug, dass wie s3perf arbeitet und die gesamte Leistungsfähigkeit untersucht, konnte für diese Arbeit nicht gefunden werden.

5 Ausgewählte Cloud-Speicherdienste

Cloud-Speicherdienste, sind Dienste, die speziell zum skalierten und hochverfügbaren Speichern von Objektdaten entwickelt wurden. Diese können meist über Weboberflächen oder Application Programming Interfaces (APIs) gesteuert und in Systeme eingebunden werden.

Speicherdienste passen sich der Verwendung an. Das bedeutet, sie bieten theoretisch unendlichen Speicherplatz und skalieren je nach Aufwand. Wenn mehr Speicherplatz benötigt wird, kann dieser angefragt und anschließend aufgestockt werden. Außerdem sollten Cloud-Speicherdienste hochverfügbar und ausfallsicher entwickelt werden, da dort Daten z.B. für Berechnungen oder Auswertungen vorgehalten und verwendet werden.

Cloud-Speicherdienste können prinzipiell in zwei Gruppen, offene und private Clouds, unterteilt werden. Diese unterscheiden sich vor allem in den Kosten, dem Standort der Infrastruktur und der Lizenz. Offene Clouds sind meist mit Kosten verbunden und die Infrastruktur steht beim Dienstleister, sodass kein physischer Einfluss auf diese genommen werden kann. Im Gegensatz dazu wird die private Cloud in den meisten Fällen auf privater Hardware, z.B. im eigenen Unternehmen, betrieben. Jedoch existieren auch Dienstleister, die ihre Infrastruktur mit einer privaten Cloud betreiben und diese ihren Kunden zur Verfügung stellen.

5.1 Öffentliche Cloud

Eine öffentliche Cloud, auch public Cloud genannt, steht allen Interessenten offen und ist meist kommerziell nutzbar. Das bedeutet, dass für diesen Dienst Kosten anfallen, welche sich in den meisten Fällen nach den verbrauchten Ressourcen richten.

Am Markt haben sich drei große Anbieter für offene Clouds positioniert, welche bereits jetzt das Internet kontrollieren. AWS ist hier, seit seiner Gründung im Jahr 2006, der Spitzenreiter, was vor allem an der vielfältigen Auswahl von Diensten und den Preisen,

welche in Amerika und Europa am geringsten sind, liegt. An zweiter Stelle positioniert sich Azure von Microsoft, welches der am schnellsten wachsende Cloud-Anbieter am Markt ist. Die GCP ist weit hinter den Kontrahenten, was vor allem an dem Gesamtkonzept und der Positionierung am Markt liegt. Nichtsdestotrotz zählt Google zu den Top-Visionären. [29]

Diese drei sind die Top IaaS-Anbieter am Markt und bieten allesamt einen Speicherdienst, der nachfolgend analysiert und getestet wird.

5.1.1 Amazon Simple Storage Service (S3)

Mit dem S3 können, wie auch bei den anderen offenen Clouds, Objekte bzw. Daten hochverfügbar gespeichert werden.

Um diesen Dienst nutzen zu können, ist ein Konto bei den AWS notwendig. Bei einer neuen Registrierung gewährt Amazon eine 12-monatige kostenlose Nutzung aller Dienste, wobei nicht alle Optionen nutzbar sind. Kostenlos können im S3 beispielsweise maximal 5 GB gespeichert und in der EC2 maximal 750 Stunden genutzt werden. Nach diesem Kontingent, spätestens jedoch nach 12 Monaten, fallen für mehr Speicher und Zeit entsprechende Kosten an. [9] Während dem Registrierungsprozess wird der Nutzer per Anruf authentifiziert und eine Kreditkarte muss angegeben werden.

Um den S3 anzusprechen gibt es, neben diversen grafischen Tools, eine API, die den Dienst direkt über die Kommandozeile ansteuert. Um über die Kommandozeile arbeiten zu können, müssen die Zugangsdaten in Form eines Zugangsschlüssels (ACCESS_KEY) und eines Sicherheitsschlüssels (SECRET_KEY) der aktuellen Shell¹ bekannt gemacht werden. Unter Linux kann dafür die Funktion `export` oder ein direkter Eintrag in die Datei `.bashrc`² genutzt werden. Solche Schlüssel können über das IAM erstellt werden. Mittlerweile wird versucht, auf IAM-Benutzer umzustellen, denen Berechtigungen und Schlüssel zugewiesen werden können. Da die AWS als IaaS aufgebaut sind, ist es sinnvoll, verschiedene Nutzer mit eigenen Berechtigungen auszustatten, wodurch jeder Benutzer seinen eigenen Zugangsschlüssel für den S3 besitzt.

Um nun Befehle an den S3 senden zu können, wird noch ein Kommandozeilenprogramm benötigt, was diese Befehle beherrscht. Dafür hat sich das Open-Source-Werkzeug `s3cmd` etabliert, welches mit einem einfachen Aufbau und logischen Befehlen arbeitet. `s3cmd`

¹ Traditionelle Benutzerschnittstelle unter Unix bzw. Linux.

² Die Datei `.bashrc` wird bei jedem Start eines Terminals geladen und enthält Befehle, wodurch z.B. das Aussehen verändert werden kann.

bietet eine eigene Konfiguration an, welche mit `s3cmd --configure` gestartet wird und die beiden Schlüssel sowie die Region abfragt. Seit Version 2.0.0 kann während der Konfiguration auch eine Domain bzw. IP angegeben werden, womit `s3cmd` für andere Dienste eingerichtet werden kann, die die Schnittstelle des S3 unterstützen.

Mit

```
$ s3cmd mb s3://BUCKET
```

kann ein neuer Bucket erstellt und mit Daten über

```
$ s3cmd put DIRECTORY/*.txt s3://BUCKET3
```

befüllt werden. Neben einem Befehl zum Herunterladen sowie zum Löschen gibt es noch weitere Befehle, die im Quellcode 5.1 aufgelistet sind.

Quellcode 5.1: Weitere Befehle die mit `s3cmd` genutzt werden können.

```
$ s3cmd ls s3://BUCKET
$ s3cmd get s3://BUCKET
$ s3cmd del s3://BUCKET/*
$ s3cmd rb s3://BUCKET
```

Eine andere Möglichkeit ist die direkte Arbeit mit dem S3 innerhalb der Weboberfläche, welche es bisher nur auf Englisch gibt. Dort kann auf einfache Art und Weise ein neuer Bucket, mit dem Button „Create Bucket“ (siehe Abbildung 5.2), erstellt werden. Daraufhin öffnet sich eine Abfrage, in der der Name und die Region angegeben werden (siehe Abbildung 5.1). Die Namen der Buckets im S3 sind einmalig und können von keinem anderen Kunden benutzt werden. Wenn gewünscht, können noch Optionen wie Versionierung oder Logging⁴ aktiviert werden. Anschließend sind die Berechtigungen zu vergeben und der Bucket wird erzeugt.

³ DIRECTORY soll an dieser Stelle einen Ordner darstellen, woraus Daten in den Bucket geladen werden. Dieser Befehl ist z.B. erweiterbar mit dem Präfix `/*.txt`, was nur Dateien des Typs `Text` hochlädt.

⁴ Logging wird das Mitschneiden von Ereignissen genannt, die im Hintergrund passieren.

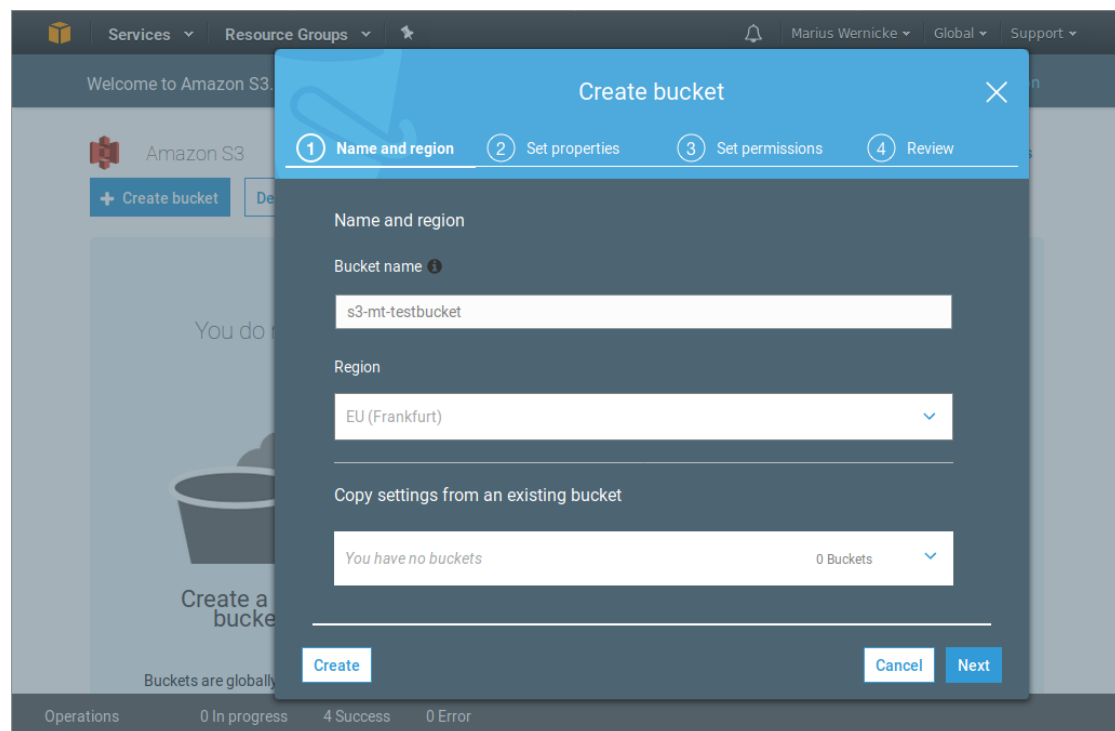


Abbildung 5.1: Erstellung eines Buckets (s3-mt-testbucket) in der S3 Management Console. Die Einstellungen auf den folgenden Reitern wurden auf den Standardwerten belassen.

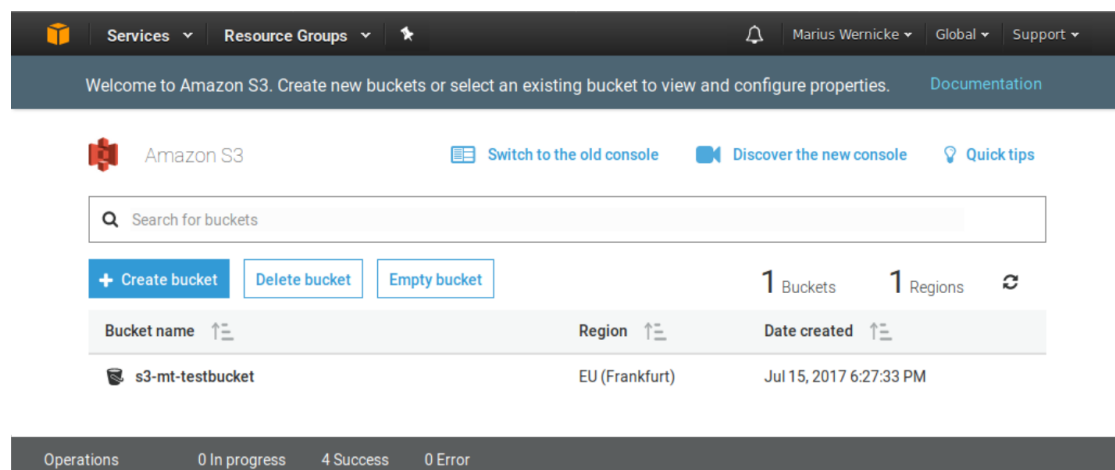


Abbildung 5.2: Übersicht über die Buckets, die sich im S3 befinden. Über einen Mausklick auf das Feld des Buckets, werden die Eigenschaften aufgerufen. Bei der Auswahl des Namens bewegt sich der Nutzer eine Ebene tiefer zum Inhalt des Buckets.

Nun können in dem erzeugten Bucket Ordner erstellt (Create folder) und neue Dateien hochgeladen (Upload) werden (siehe Abbildung 5.3). Diese in der Cloud vorgehaltenen Daten können, bei bestehender Internetverbindung, von überall auf der Welt

wieder heruntergeladen werden.

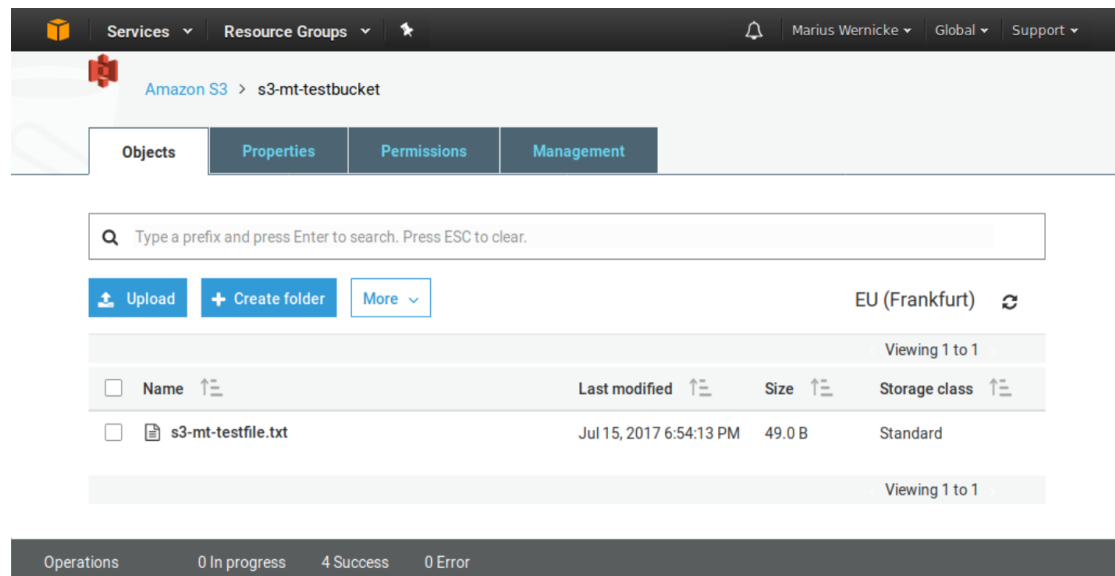


Abbildung 5.3: Eine Ebene tiefer wird eine Übersicht der im Bucket enthaltenen Daten angezeigt, wo diese bearbeitet werden können.

Wie in Abbildung 5.1 bis 5.3 zu sehen ist, befindet sich am unteren Rand ein Balken mit Informationen über die aktuellen Operationen. Dadurch kann der Überblick über den Status der einzelnen Aktionen behalten werden.

Diese ganzen Schritte werden mit `s3cmd` automatisiert, wobei alle Einstellungen, bis auf die Namen, auf den Standardwerten belassen werden.

Noch mehr Automatisierung wird mit dem Skript `s3perf` umgesetzt, mit welchem viele verschiedene Befehle des `s3cmd` durch einen einzigen Aufruf ausgeführt werden.

5.1.2 Microsoft Azure Blob Storage (ABS)

Microsoft ABS ist ein Speicherdienst für Objektdaten, welcher seine Daten *Blobs* und seine Buckets *Container* nennt. Der Dienst ist vergleichbar mit dem S3 und bietet etwa das gleiche Leistungsangebot. Daten können hoch- und heruntergeladen und Container können erstellt sowie gelöscht werden. Jedoch handelt es sich hierbei nicht um einen Speicherdienst, welcher die S3-Schnittstelle nutzt. Durch die Ähnlichkeit mit anderen Diensten wird die Azure-Schnittstelle in `s3perf` gepflegt und zur Verfügung gestellt (genauer in Kapitel 6 und 7).

Für die Nutzung des ABS wird ein Microsoft Konto benötigt und es ist eine Authentifizierung über SMS notwendig. Außerdem wird die Angabe einer Kreditkarte verlangt. Da Microsoft auch einen Mailedienst anbietet, besteht die Möglichkeit, sich eine komplett neue Mailadresse für die Nutzung von Azure anzulegen. Die Einrichtung des Kontos nimmt nur wenige Sekunden in Anspruch. Microsoft gewährt eine Nutzung seiner Dienste im Gesamtwert von 170 Euro und verspricht, dass das kostenlose Konto nicht automatisch in ein Abo umgewandelt wird.

Die Weboberfläche von Azure ist ähnlich wie Windows 10 aufgebaut, was vor allem für Windows-Nutzer den Einstieg erleichtert. Azure nutzt für sein Interface eine adaptive Menüführung, welche horizontal erweitert wird (siehe Abbildung 5.4). Das bedeutet, dass neue Unterpunkte an das bestehende Menü angedockt werden. Dadurch ergibt sich eine Ansicht, welche horizontal, anstatt vertikal, verschoben werden muss.

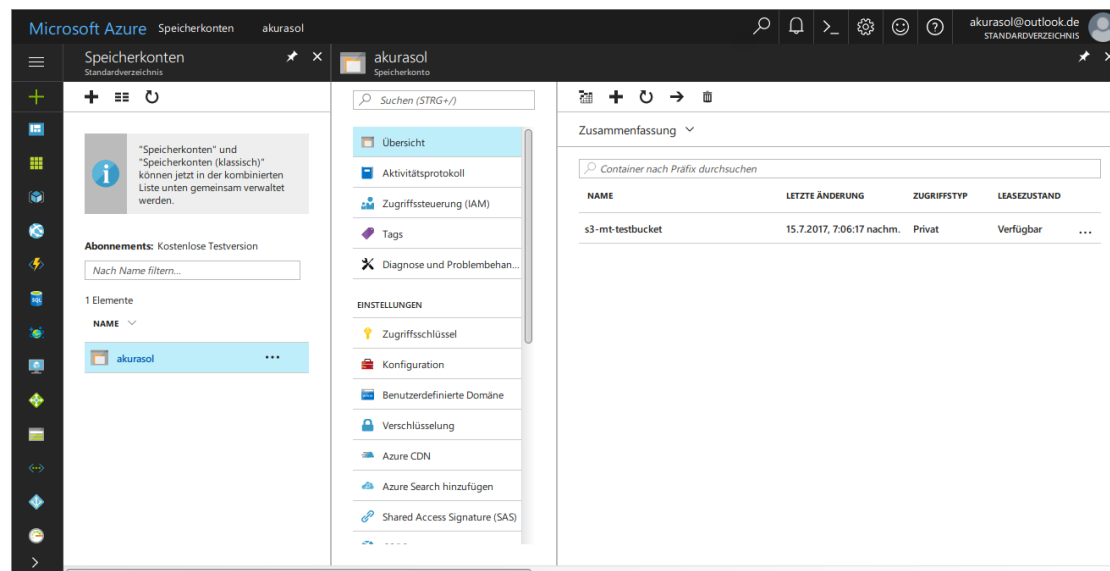


Abbildung 5.4: Das Speicherkonto (akurasol ist ein fiktiver Name) öffnet die Übersicht über die Container. Weitere Menüpunkte sind in der mittleren Spalte zu sehen.

Nach der Anmeldung stehen dem Nutzer, in einer Seitenleiste (siehe Abbildung 5.4), sämtliche Dienste zur Verfügung, die Azure zu bieten hat. Für den Speicherdienst ABS muss der Menüpunkt *Speicherkonten* gewählt werden, worüber ein Konto mit einem einmaligen Namen, in Azure, eingerichtet wird. Anschließend können Container für Blobs erzeugt werden, die jedoch nicht, wie beim S3, einmalig sind, was die Namensfindung erleichtert (siehe Abbildung 5.4). Wenn Container gelöscht werden, sind die Namen für kurze Zeit reserviert und können nicht sofort mit dem gleichen Namen neu erstellt werden. Innerhalb dieses Containers können wieder über verschiedene

Buttons bzw. Symbole Daten hochgeladen und bearbeitet werden (siehe Abbildung 5.5). Außerdem können die Zugriffsrichtlinien auf *Privat* (kein öffentlicher Zugriff), *Blob* (öffentlicher Lesezugriff) oder *Container* (öffentlicher Lesezugriff) eingestellt werden.

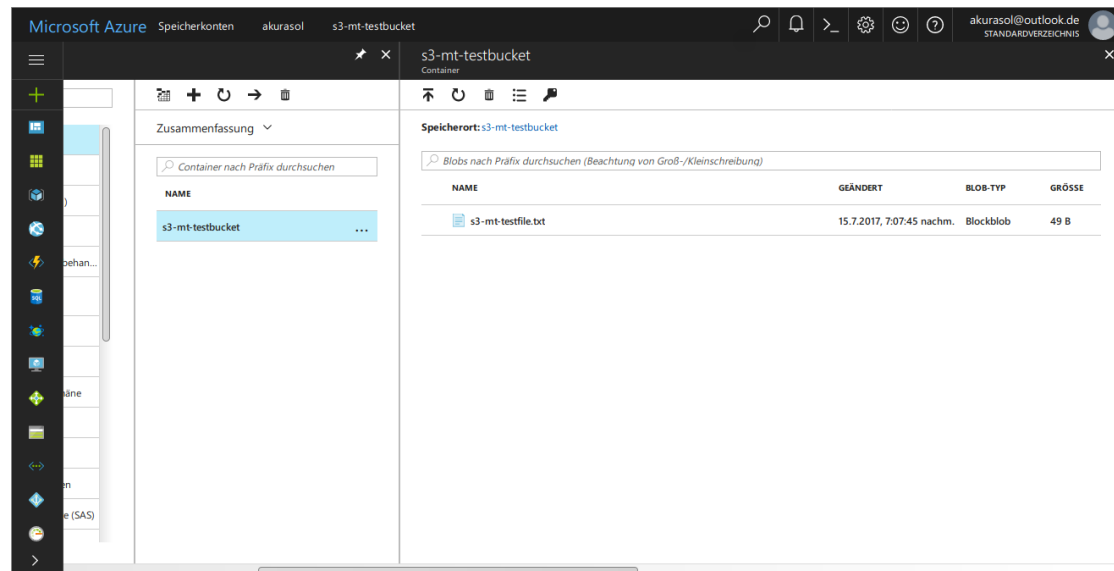


Abbildung 5.5: Die Übersicht über die Dateien bzw. Blobs in den Containern wird an die Containerübersicht angedockt bzw. erweitert.

Azure lässt sich auch über die Konsole mit Hilfe des Azure CLI⁵ bedienen. Um Azure ansprechen zu können, müssen die Zugangsdaten, ähnlich zu `s3cmd`, der aktuellen Shell mitgeteilt werden, was mit

```
$ export AZURE_STORAGE_ACCOUNT
$ export AZURE_STORAGE_ACCESS_KEY
```

für den Nutzernamen und den Zugangsschlüssel durchgeführt werden kann. Die Azure CLI hat jedoch die Einschränkung, dass immer nur mit einer einzigen Datei gearbeitet werden kann.

Nachdem mit

```
$ az storage container create --name CONTAINER
```

ein Container erstellt wurde, kann dieser mit Dateien befüllt werden. Für das Hoch- bzw. Herunterladen existiert ein Batch-Befehl, womit mehrere Daten auf einmal verarbeitet werden können. Damit wird lediglich die Quelle, also der Ordner mit den Daten und das Ziel, der Container, angegeben. Beim Herunterladen ist der Befehl genau umgekehrt, um Daten aus dem Container in den lokalen Ordner zu laden.

⁵ Command Line Interface, das Kommandozeileninterface, worüber Befehle gesendet werden können.

```
$ az storage blob upload-batch --destination CONTAINER --source DIRECTORY/
$ az storage blob download-batch --destination DIRECTORY/ --source CONTAINER
```

Leider existiert bisher kein Batch-Befehl, welcher mehrere Daten auf einmal aus einem Container löscht. Um diesen Umstand zu umgehen, kann eine for-Schleife geschrieben werden, die für alle Daten innerhalb eines Containers das Löschskript aufruft und so den gesamten Container leert.

```
1 for i in `az storage blob list --container-name CONTAINER --output table | awk '{print $1}' | sed '1,2d' | sed '/^$/d'`; do
2   if az storage blob delete --name $i --container-name CONTAINER >/dev/null ; then
3     echo "Files $i inside the CONTAINER have been erased"
4   else
5     echo "Unable to erase the files $i inside the CONTAINER." && exit 1
6   fi
7 done
```

Im ersten Schritt wird der Inhalt des Containers aufgelistet und mit awk und sed auf die Dateinamen gekürzt. Anschließend wird diese Schleife für jede Datei innerhalb des Containers durchlaufen und gelöscht. Durch eine Meldung kann festgestellt werden, ob das Löschen erfolgreich war.

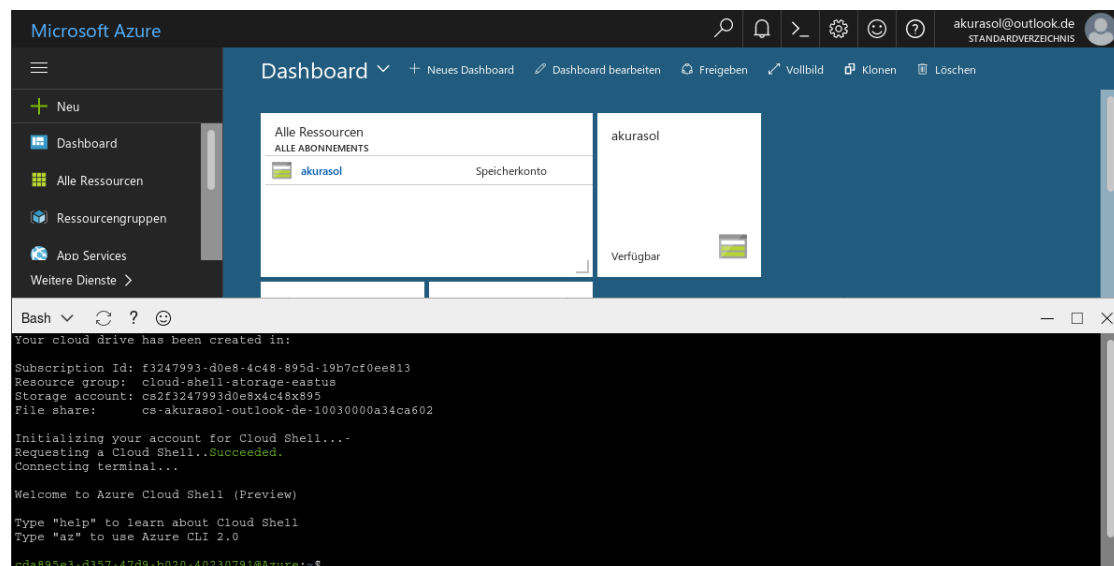


Abbildung 5.6: Azure-Dashboard mit geöffneter Cloud Shell.

Alle weiteren Befehle werden von dem Azure CLI direkt unterstützt und müssen nicht speziell angepasst werden. Die Befehle können auch alle innerhalb von Azure selbst mit

der Cloud Shell (siehe Abbildung 5.6) ausgeführt werden, welche in die Weboberfläche integriert ist.

5.1.3 Google Cloud Storage (GCS)

Der GCS ist der Speicherdienst von Google und vergleichbar mit dem S3 von Amazon. Im Gegensatz zu Microsoft lässt sich der GCS zusammen mit `s3cmd` nutzen, da die Schnittstelle sehr ähnlich ist. Daneben gibt es eine Weboberfläche, welche sich am Material Design, welches Google bei all seinen Diensten durchsetzt, orientiert.

Die Registrierung gestaltet sich ähnlich zu den anderen offenen Cloud-Speicherdiensten und bedarf nur wenigen Eingaben. Es wird auch hier die Angabe einer Kreditkarte benötigt, jedoch stellt Google für einen Zeitraum von 12 Monaten 300 (virtuelle) Dollar zur Nutzung aller GCP-Dienste zur Verfügung. Nach dieser Zeit gibt es Dienste, welche weiterhin kostenlos genutzt werden können. So ist es z.B. möglich, eine `f1-micro` Instanz mit 30 GB Festplattenspeicher oder 5 GB Speicherplatz pro Monat in dem GCS zu nutzen.

Um die Dienste nutzen zu können, muss ein Projekt erstellt werden, worin die Dienste, APIs, Kosten und Schlüssel gebündelt werden. Bei der Projekterstellung wird ein Name benötigt, welcher frei gewählt werden kann. Dazu wird eine Projekt-ID zugewiesen, welche nach dem Schema `projektname-123456` benannt wird. Wenn dies geschehen ist, stehen alle Dienste zur Verfügung und werden eindeutig dem Projekt zugewiesen.

Über das Menü (\equiv) in der oberen linken Ecke, kann aus den verschiedenen Diensten gewählt werden. Unter *Storage* gibt es den Unterpunkt *Storage*, welcher zum GCS führt und die Bucketübersicht anzeigt. Wenn bisher keine Buckets erstellt wurden, wird hier ein kleines Overlay angezeigt, welches dem Nutzer signalisiert einen Bucket zu erstellen (siehe Abbildung 5.7). Es muss lediglich ein Name und die Storage-Klasse des Buckets bei der Erstellung angegeben werden (siehe Abbildung 5.8). Die Klasse des Buckets bestimmt, wie oft die Daten abgerufen und wie diese genutzt werden:

- Multi-Regional: Zum Streamen von Videos und Hosten von Webinhalten. Redundant innerhalb einer Region, wie z.B. EU.
- Regional: Zum Speichern von Daten und Ausführen von Datenanalysen. In einer einzelnen Region redundant, wie z.B. `eu-west-1`.

- Nearline: Zum Speichern von Dokumenten, auf die selten zugegriffen wird. In einer beliebigen Region innerhalb einer Unterregion redundant, wie z.B. EU (beliebige Region)⁶.
- Coldline: Zum Speichern von Dokumenten, auf die sehr selten zugegriffen wird. In einer beliebigen Region innerhalb einer Region redundant, wie z.B. EU (beliebige Region).

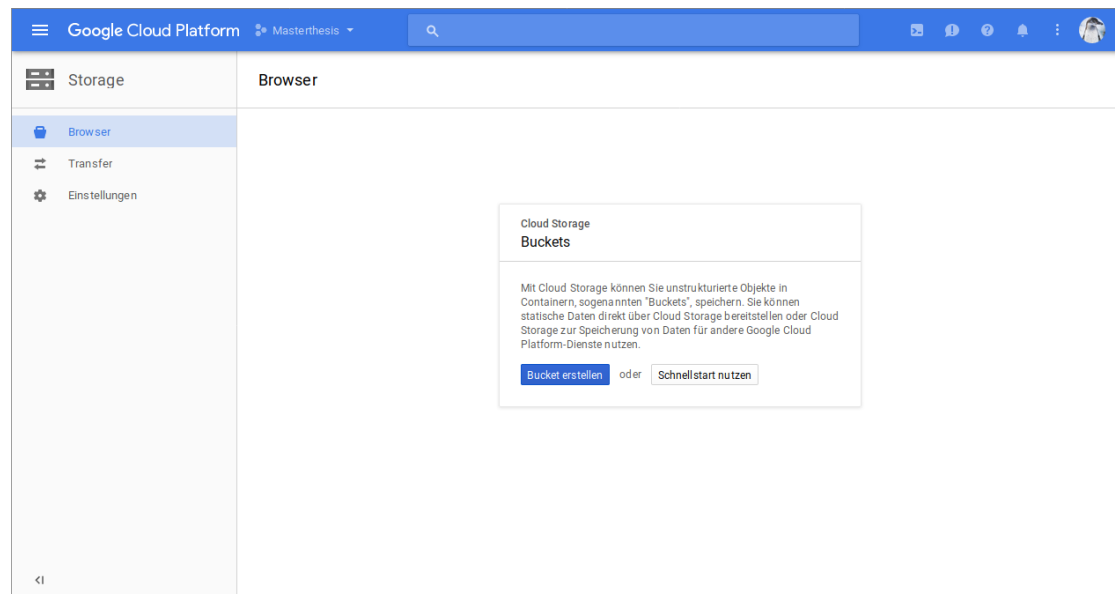


Abbildung 5.7: Die Übersichtsseite der Buckets, wo der Nutzer animiert wird, einen Bucket zu erstellen, sollte noch keiner vorhanden sein.

Wenn der Bucket erfolgreich erstellt wurde, wird der Nutzer anschließend auf den Inhalt des Buckets weitergeleitet. Darin ist es möglich, einzelne Dateien oder ganze Ordner hochzuladen oder zu erstellen (siehe Abbildung 5.9).

Über den Menüpunkt *Transfer* auf der linken Seite, können diverse Übertragungen von z.B. Amazons S3 oder anderen Diensten, die Buckets nutzen, eingerichtet werden. So können bspw. Backups von einem Bucket in einen anderen getätigt werden, falls einer ausfällt oder versehentlich gelöscht wird.

Die GCP bietet für alle seine Dienste diverse APIs für sehr viele Programmiersprachen an. Damit kann der GCS auf einfache Art und Weise in ein bestehendes Programm eingebunden werden. Ein Kommandozeilenprogramm, wie es eines bei Amazon und Microsoft gibt, existiert ebenfalls bei Google und nennt sich `gsutil`. Das Programm ist Bestandteil des Google Cloud Software Development Kit (SDK). Dieses Werkzeug

⁶ Beliebige Region bedeutet, dass innerhalb der Region EU eine „Unterregion“ beliebig ausgesucht wird.

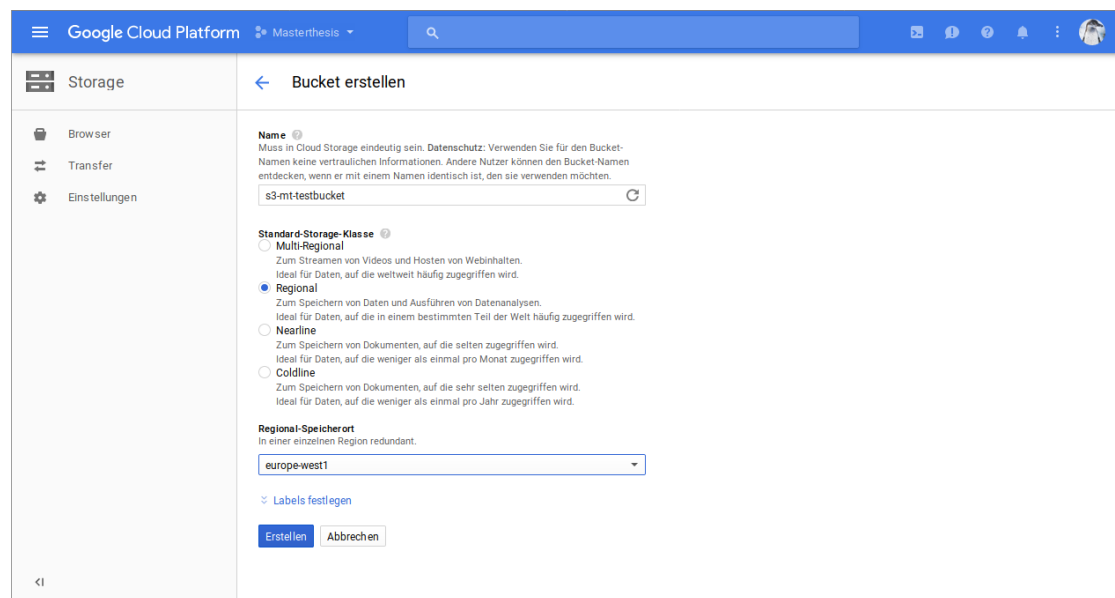


Abbildung 5.8: Neben der Vergabe eines Namens für den Bucket muss festgelegt werden, welcher Klasse dieser zugewiesen wird und wie Daten darin genutzt werden sollen.

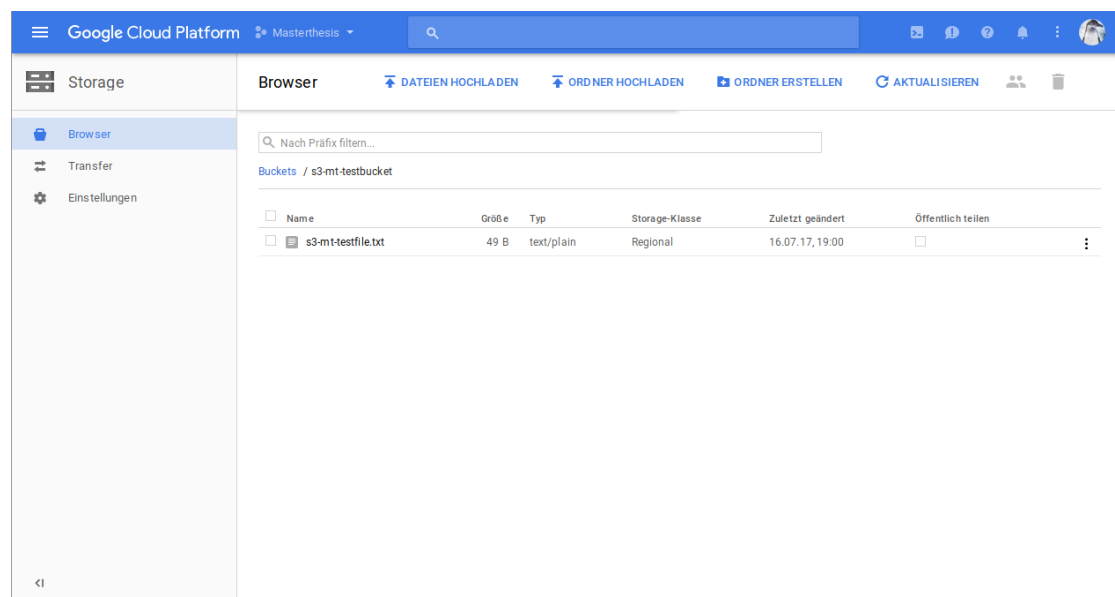


Abbildung 5.9: Die Seite mit dem Inhalt des Buckets gibt übersichtlich an, welche Aktionen durchgeführt werden können.

wurde auf Grund von Fehlermeldungen mit `s3cmd` in `s3perf` integriert (siehe Kapitel 7).

Das Kommandozeilenprogramm `gsutil` benutzt zum Interagieren Parameter, wie sie auch unter Linux und von `s3cmd` verwendet werden. Mit dem Befehl

5 Ausgewählte Cloud-Speicherdienste

```
$ gsutil mb gs://BUCKET/
```

wird ein neuer Bucket erstellt und mit

```
$ gsutil cp DIRECTORY/File.txt gs://BUCKET
```

wird eine Datei hochgeladen. Durch den Befehl

```
$ gsutil ls gs://BUCKET
```

wird der Inhalt des Buckets aufgelistet. Um den Bucket wieder zu löschen, kann

```
$ gsutil rm -r gs://BUCKET
```

ausgeführt werden und der Bucket und dessen Inhalt werden gelöscht.

Google bietet auch innerhalb seiner Weboberfläche eine Shell an, mit der sich die Befehle aus dem Google Cloud SDK direkt dort eingeben lassen (siehe Abbildung 5.10). Somit wird nicht extra eine Konsole benötigt, wenn direkt mit der Weboberfläche interagiert wird.

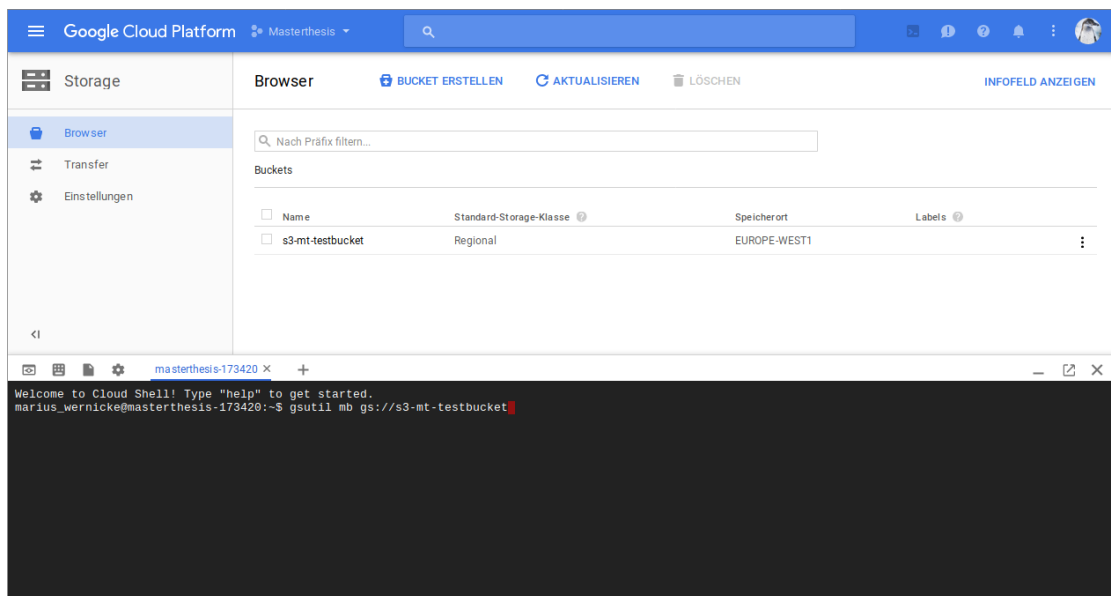


Abbildung 5.10: Die Google Cloud Shell innerhalb der Weboberfläche der GCP.

Wie anhand der Parameter zu sehen ist, orientieren sich diese stark an den Parametern, welche auch unter Linux genutzt werden. So ist vor allem der Einstieg für Linux-Nutzer einfacher, als z.B mit Azure.

5.2 Private Cloud

Eine private Cloud ist eine Cloud, welche nicht für die Öffentlichkeit zugänglich ist. Zumeist handelt es sich dabei um unternehmensweite oder persönliche Installationen, die nur für eine bestimmte Nutzergruppe zur Verfügung steht. Außerdem sind die Cloud-Speicherdienste, welche nachfolgend erklärt werden, in den meisten Fällen Open-Source-Software (OSS) und damit kostenlos nutzbar und an die eigenen Bedürfnisse anpassbar.

Insgesamt wurden neun Cloud-Speicherdienste ausgewählt, analysiert und installiert. Um Vergleichbarkeit zu gewährleisten war Grundlage für die Installationen jeweils eine t2.micro-Instanz der AWS mit, wenn nicht anders angegeben, Ubuntu 16.04 und eine VirtualBox VM mit einer virtual CPU (vCPU) und 1 GB Arbeitsspeicher, ebenfalls Ubuntu 16.04. Vor Beginn der Cloud-Installationen wurde das System immer mit

```
$ sudo apt update  
$ sudo apt dist-upgrade
```

auf den aktuellen Stand gebracht.

Die Installationen, z.B. mit Vagrant, wurden von einem Lenovo E460 mit einem i3-6100U und 8 GB Arbeitsspeicher, sowie Xubuntu 17.04 als Betriebssystem durchgeführt.

5.2.1 Eucalyptus Walrus

Bei Eucalyptus⁷ handelt es sich um eine IaaS, welche nach den AWS modelliert wurde und mit diesen kompatibel ist [38]. Mit Eucalyptus lassen sich VMs sowie Schlüssel zur Authentifizierung erzeugen und Daten mit dem Speicherdienst *Walrus*, welcher die S3-API nutzt, verwalten.

Da es sich bei Eucalyptus um eine vollständige Infrastruktur handelt, kann diese auch automatisiert VMs erzeugen und verwalten. Um dies zu bewerkstelligen, muss der Prozessor Virtualisierungsfunktionen, wie z.B. Intel-VT oder AMD-V, unterstützen. Das bedeutet aber auch von Anfang an, dass Eucalyptus nicht innerhalb der

⁷ Eucalyptus ist ein Akronym für *Elastic Utility Computing Architecture for Linking Your Programs To Useful Systems*.

AWS installiert werden kann, da dort keine verschachtelte Virtualisierung⁸ unterstützt wird.

Ersichtlich wird dieser Umstand, sobald das Faststart-Skript von Eucalyptus in einer AWS-Instanz ausgeführt wird. Dieses Skript vereinfacht die Installation von Eucalyptus enorm und automatisiert diese, setzt allerdings CentOS⁹ als Betriebssystem voraus. Sobald eine Instanz mit CentOS erzeugt wurde, kann mit dem Befehl

```
$ bash <(curl -s https://raw.githubusercontent.com/eucalyptus/eucalyptus-cookbook/master/faststart/cloud-in-a-box.sh)
```

das Skript heruntergeladen und gestartet werden. Wichtig ist dabei, dass der aktuelle Benutzer root ist, da sonst das Skript nicht ausgeführt wird. Im Laufe des Skripts wird abgefragt, ob die Central Processing Unit (CPU) Virtualisierung unterstützt. Da dies in den AWS nicht der Fall ist, bricht das Skript an dieser Stelle ab. Und damit ist es nicht möglich, Eucalyptus innerhalb einer AWS-Instanz zu installieren. Eine Möglichkeit bietet ein externer Dienst namens *Ravello Systems*, welcher mit einem eigenen Hypervisor arbeitet, an.

Ravello Systems

Ravello Systems bietet die Möglichkeit, spezielle EC2-Instanzen zu erzeugen, die eine verschachtelte Virtualisierung erlauben. Dabei wird eine Instanz erzeugt, worüber ein sogenannter HVX, der Hypervisor oder auch VMM von Ravello, installiert wird. Dieser simuliert dem System die Funktion zur Virtualisierung, womit CPUs, Random Access Memory (RAM) und Netzwerke virtualisiert werden können. [71]

Bei der Erzeugung einer Instanz über Ravello wird auch die Region der Instanz erfragt. Diese kann jedoch nur in der kostenpflichtigen Version bestimmt werden. In der Testversion wird die Instanz in einer beliebigen Region (Amazon oder Google) erzeugt. Das ist dahingehend ein Nachteil, da alle anderen Systeme in der Region eu-central-1 (Frankfurt) erzeugt wurden und eine andere Region somit keine Vergleichbarkeit für spätere Tests bietet, da vor allem die Übertragungszeiten stark unterschiedlich sind.

Die Installation des Faststart-Skripts bricht nach kurzer Zeit mit der Fehlermeldung ab, dass keine Verbindung zu einem anderen Dienst initiiert werden konnte. Das be-

⁸ Verschachtelte Virtualisierung bedeutet, dass eine Virtualisierung innerhalb einer virtualisierten Umgebung stattfindet. Also eine VM in einer VM.

⁹ CentOS, aktuell in Version 7, ist eine Community Version des RedHat Enterprise Linux.

deutet, dass sich Ravello nicht für die Installation einer kompletten Infrastruktur eignet.

Alternativen

Die Installation in einer lokalen VM mit VirtualBox oder VMware Workstation Player blieb ebenfalls erfolglos, da VirtualBox einerseits keine Virtualisierung unterstützt und VMware Workstation Player die gleiche Fehlermeldung wie unter Ravello anzeigt.

Insgesamt sollte Eucalyptus auf einem „Bare-Metal“-Server¹⁰ und nicht in einer virtualisierten Umgebung installiert werden, da dadurch viel Leistung und Skalierung verloren gehen und es letztendlich nur dort lauffähig ist.

5.2.2 Fake S3

Fake S3 ist ein leichtgewichtiger Server, welcher die gleiche API wie Amazons S3 benutzt. Dieser Server kann für das Testen von S3 in einer Sandbox¹¹ sinnvoll sein, ohne kostenpflichtige Anfragen bei den AWS zu tätigen. [50]

Fake S3 basiert auf Ruby, welches ebenfalls auf dem Server installiert sein muss, um Fake S3 zu betreiben. Um Ruby unter Ubuntu zu installieren ist es sinnvoll, den Ruby Version Manager (RVM) zu nutzen, um eine möglichst aktuelle Version und bestmögliche Leistung zu erhalten¹². In den Paketquellen von Ubuntu befindet sich aktuell die Version 2.3.1, welche zuletzt im April 2016 aktualisiert wurde.

Ruby installieren

Mit der Installation von Ruby über RVM existiert auch ein Werkzeug, welches die Version von Ruby verwaltet. Damit kann ständig die aktuelle Version von Ruby installiert werden, ohne z.B. die Quellen jedes mal neu kompilieren zu müssen, was sehr viel Zeit in Anspruch nehmen kann. Für Ubuntu existiert zwar eine Paketquelle für RVM, jedoch sollte auch RVM in der aktuellsten Version installiert werden.

¹⁰ Ein Server, der auf physischer Hardware und nicht virtualisiert betrieben wird.

¹¹ Eine Sandbox ist ein System oder ein Programm, worin Programme, abgeschottet vom restlichen System, ausgeführt werden können.

¹² Selbst in der Ruby-Community wird vehement dazu geraten, Drittanbieter-Werkzeuge anstelle des Paketverwaltungssystems zu nutzen [73].

Um Ruby mit Hilfe von RVM zu installieren, sind mehrere Schritte notwendig. Zuerst wird mit

```
$ gpg --keyserver hkp://keys.gnupg.net --recv-keys 4092  
B6B1796C275462A1703113804BB82D39DC0E3 72  
D2BAF1CF37B13E2069D6956105BD0E739499BDB  
$ \curl -sSL https://get.rvm.io | bash -s stable
```

der öffentliche Schlüssel zum Verifizieren des Installationspakets installiert, um die Sicherheit zu erhöhen. Der zweite Befehl lädt RVM herunter und installiert es in der aktuellen stabilen Version 1.29.2. Mit dem Befehl

```
$ rvm install ruby
```

wird Ruby in Version 2.4.0 installiert.

Fake S3 installieren

Nachdem Ruby fertig installiert und eingerichtet ist, wird mit

```
$ gem install fakes3
```

Fake S3 über einen `gem`¹³ auf dem System installiert. Bei der Installation von Fake S3 werden auch `thor`¹⁴ und `builder`¹⁵ als Abhängigkeiten durch `gem` mitinstalliert.

Bevor Fake S3 gestartet wird, ist es sinnvoll, zumindest innerhalb der AWS, die Sicherheitsgruppe der Instanz so anzupassen, dass der Port der Servers auch auf der Instanz offen ist, sodass mit Fake S3 kommuniziert werden kann.

Um den Server zu starten, genügt es den Befehl

```
$ fakes3 -r ~/s3folder -p 8080
```

auszuführen. Unter `~/s3folder` befindet sich das `root`-Verzeichnis für die Buckets und Fake S3 ist über den Port 8080 erreichbar. Ein Aufruf über den Browser gibt eine XML-Datei aus, was bedeutet, dass der Server ordnungsgemäß läuft (siehe Quellcode 5.2).

¹³ `Gems` ist das Paketsystem, ähnlich zu `apt`, der Programmiersprache Ruby, womit es sehr einfach ist, neue Programmbibliotheken oder auch ganze Programme zu installieren, zu verwalten und auch wieder zu entfernen [69].

¹⁴ `thor` ist zum einfachen und effizienten Bauen von Kommandozeilen-Werkzeugen [37].

¹⁵ `builder` erstellt auf einfache Art und Weise XML-Markups und -Datenstrukturen [49].

Quellcode 5.2: XML-Datei beim Aufruf des Fake S3 Servers über einen Webbrowser.

```
1 <ListAllMyBucketsResult>
2   <Owner>
3     <ID>123</ID>
4     <DisplayName>FakeS3</DisplayName>
5   </Owner>
6   <Buckets/>
7 </ListAllMyBucketsResult>
```

Fake S3 kann mit einer großen Anzahl an Clients in unterschiedlichen Programmiersprachen interagieren [51]. Dazu ist es meistens nur nötig, die IP sowie den Port und die Zugangsschlüssel zu ändern bzw. einzutragen.

s3cmd einrichten

Fake S3 kann, wie Amazon und Google, mit dem Werkzeug s3cmd kommunizieren. Für Fake S3 muss jedoch die Konfigurationsdatei .s3cfg so angepasst werden, dass s3cmd mit Fake S3 zusammenarbeitet (siehe Quellcode 5.3).

Quellcode 5.3: Diese Zeilen müssen in der .s3cfg angepasst werden, damit s3cmd ordnungsgemäß mit Fake S3 kommuniziert.

```
1 access_key = 123
2 host_base = <ip_of_vm>:8080
3 host_bucket = <ip_of_vm>:8080
4 secret_key = abc
5 use_https = False
```

Mit diesen Einstellungen können fast alle Aktionen mit Fake S3 ausgeführt werden. Buckets können erstellt, Daten können hoch- und heruntergeladen sowie wieder gelöscht werden. Jedoch lassen sich die Buckets in der aktuellen Version von Fake S3, mit einer Fehlermeldung, dass der Bucket nicht leer sei, nicht löschen [79]. Damit ist es nicht möglich, diesen Speicherdienst im späteren Verlauf zu testen, da das Löschen des Buckets mit zum Test zählt.

5.2.3 Minio

Minio ist ein Server für Objektdaten, der in Go geschrieben wurde, und ist kompatibel mit dem S3 von Amazon. Minio ist am besten zum Speichern von unstrukturierten Daten wie Fotos, Videos, Log-Dateien, Backups und Containern bzw. VM-Abbildern geeignet. Die Dateigröße kann von wenigen kB bis zu 5 TB gehen. [61]

Minio installieren

Um Minio unter Linux zu installieren, muss mit

```
$ wget https://dl.minio.io/server/minio/release/linux-amd64/minio
```

der Server (Version zum Zeitpunkt des Schreibens dieser Arbeit: 2017-07-24T18-27-35Z) heruntergeladen werden. Dabei handelt es sich um eine einzige Datei, welche ausgeführt wird, um den Server zu starten. Zuvor wird die Datei mit

```
$ chmod +x minio
```

ausführbar gemacht und mit

```
$ ./minio server --address ":8080" ~/s3folder
```

auf Port 8080 gestartet. Der Minio Server gibt nach erfolgreichem Start eine Meldung aus, dass dieser jetzt mit den Parametern aus Quellcode 5.4 läuft. Die Adressen von Endpoint und Browser Access sind die internen bzw. lokalen IP-Adressen der EC2-Instanz und zeigen nicht nach draußen. Außerdem hat Minio einen Webclient, mit welchem die Daten grafisch verwaltet werden können (siehe Abbildung 5.11).

Quellcode 5.4: Ausgabe des Minio Servers nachdem dieser erfolgreich gestartet wurde.

```
Created minio configuration file successfully at /home/ubuntu/.minio

Endpoint:  http://172.31.32.116:8080  http://127.0.0.1:8080
AccessKey:  XQT3R4HENLKZSOP2JDLK
SecretKey:  M1xlQtZQM0sqYU2/5L/9Hm4qwrfrRNKkZTjKdydDw

Browser Access:
http://172.31.32.116:8080  http://127.0.0.1:8080

...

Drive Capacity: 6.5 GiB Free, 7.7 GiB Total
```

s3cmd einrichten

Um auch hier über s3cmd mit dem Cloud-Speicherdienst kommunizieren zu können, sind nur wenige Anpassungen der .s3cfg notwendig [62]:

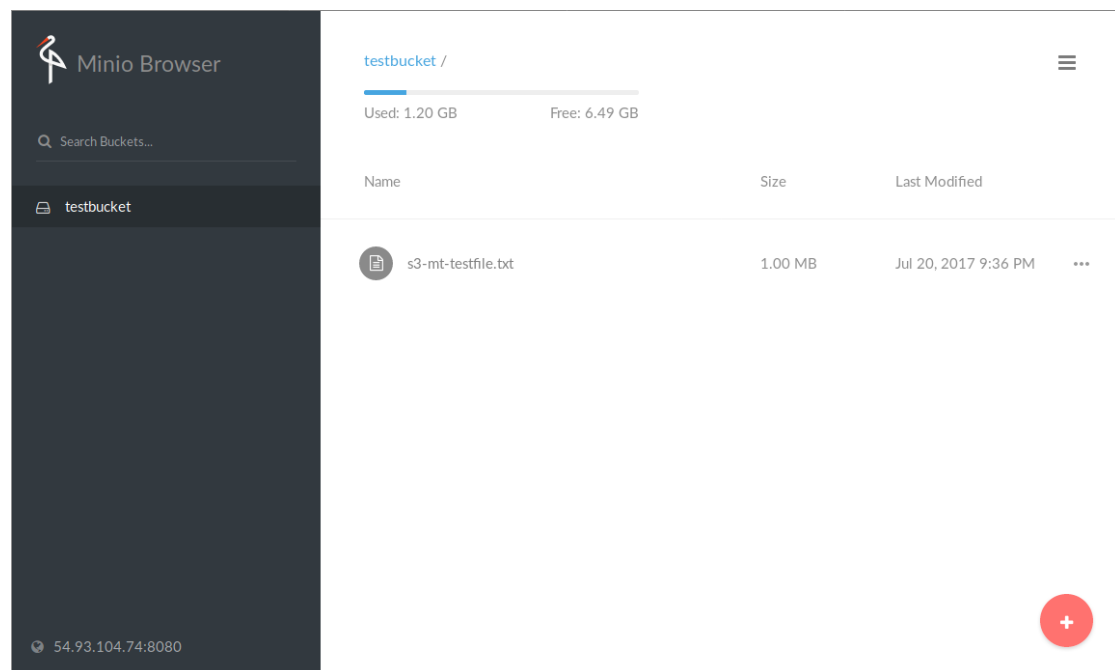


Abbildung 5.11: Über den Webclient können Daten auch visuell bearbeitet werden. Dieser ist vergleichbar zu den Weboberflächen der öffentlichen Clouds.

```
1 # Setup endpoint
2 host_base = <ip_of_vm>:8080
3 host_bucket = <ip_of_vm>:8080
4 bucket_location = us-east-1
5 use_https = False
6
7 # Setup access keys
8 access_key = XQT3R4HENLKZSOP2JDLK
9 secret_key = Mlx1QtZQM0sqYU2/5L/9Hm4qwrfrNKkZTjKdydDw
10
11 # Enable S3 v4 signature APIs
12 signature_v2 = False
```

Wenn alles richtig konfiguriert wurde, kann mit Hilfe von `$ s3cmd mb s3://bucket` ein neuer Bucket erstellt werden, welcher auch direkt im Webclient sichtbar sein sollte. Anschließend ist es möglich, Daten in den Bucket hoch- und herunterzuladen sowie den Inhalt aufzulisten.

5.2.4 Nimbus Cumulus

Bei Nimbus handelt es sich um eine Open-Source-IaaS speziell für wissenschaftliche Projekte. Der S3-konforme Speicherdienst von Nimbus nennt sich Cumulus und kann

auch ohne die gesamte Infrastruktur alleine installiert werden. Dies ist auch sinnvoll, da die gesamte Funktionalität für diese Arbeit nicht benötigt wird.

Pakete installieren

Da Nimbus Cumulus auf Python basiert und SQLite als Datenbank benötigt, müssen zu Beginn diese Abhängigkeiten installiert werden. Unter Ubuntu kann dies mit

```
$ sudo apt update
$ sudo apt install virtualenv sqlite3 python-dateutil python-pysqlite2 \
    build-essential libssl-dev libffi-dev ant-optional python-dev \
    openjdk-9-jdk
```

durchgeführt werden, wodurch alle notwendigen Pakete und Abhängigkeiten¹⁶ für Cumulus installiert werden.

In der genutzten Ubuntu-Version muss ein fehlender Link manuell hinzugefügt werden, da sonst die Installation fehlschlägt [26]:

```
$ sudo ln -s /usr/lib/python2.7/plat-*/_sysconfigdata_nd.py /usr/lib/\
python2.7/
```

Anschließend ist es sinnvoll, mit Hilfe von

```
$ sudo adduser nimbus
$ su nimbus
$ cd ~
```

einen Nutzer für das Ausführen von Nimbus Cumulus zu erstellen und auf diesen Nutzer sowie in dessen Heimatverzeichnis zu wechseln.

Nimbus Cumulus installieren

Um die Quelldateien herunterzuladen und zu entpacken, wird

```
$ curl -O http://www.nimbusproject.org/downloads/nimbus-iaas-2.10.1-src\
.tar.gz
$ tar xvzf nimbus-iaas-2.10.1-src.tar.gz
```

¹⁶ virtualenv erstellt eine virtuelle Entwicklungsumgebung, sqlite3 und python-* installieren sowohl die Hauptpakete als auch Entwicklungspakete, womit Quelldateien kompiliert werden, build-essential beinhaltet diverse Compiler, um Quelltext zu kompilieren, libssl-dev und libffi-dev sind Entwicklungspakete für die entsprechenden Bibliotheken, ant-optional ist eine Java-Bibliothek und openjdk-9-jdk installiert das Entwicklungspaket für Java.

ausgeführt. Hier wurde zwar das komplette Infrastruktur-Paket heruntergeladen, jedoch kann Cumulus aus diesem gesondert installiert werden.

Vor Beginn der Installation sollte die Version von PyOpenSSL auf den aktuellen Stand gebracht werden, da die mitgelieferte Version 0.13 nicht ordnungsgemäß mit der genutzten Ubuntu-Version arbeitet [26]. Dazu muss mit

```
$ cd ~/nimbus-iaas-2.10.1-src/cumulus/deps/  
$ wget https://pypi.python.org/packages/3a/c1/65822bca6b910848d36556e47e00408c28c53df2c846d76f04cb033718f37/pyOpenSSL_0.15.tar.gz#md5=661ddf97b75320d6004a56160a4a8578
```

in das entsprechende Verzeichnis gewechselt und PyOpenSSL-0.15 per wget heruntergeladen werden. Anschließend muss in der Datei

```
$ nano ~/nimbus-iaas-2.10.1-src/cumulus/reqs.txt
```

der Eintrag `deps/pyOpenSSL-0.13.tar.gz` zu `deps/pyOpenSSL-0.15.tar.gz` geändert werden.

Ein neuer Ordner wird mit

```
$ mkdir ~/cumulus  
$ ./nimbus-iaas-2.10.1-src/cumulus/cumulus-install.sh ~/cumulus
```

erstellt und anschließend Cumulus in diesen Ordner installiert.

Ist das Skript ohne Fehler durchgelaufen, sollte der Shell der Pfad von Cumulus bekannt gemacht werden, sodass von überall die Befehle ausgeführt werden können:

```
$ echo "export CUMULUS_HOME=~/.cumulus" > ~/.bashrc  
$ bash
```

Zu guter Letzt wird Cumulus mit

```
$ ~/cumulus/bin/cumulus &
```

gestartet¹⁷. Der Cumulus-Server läuft jetzt auf Port 8888, worüber dieser, wieder per `s3cmd`, angesprochen werden kann. Doch bevor das möglich ist, wird ein Nutzer erstellt, der anschließend die Zugangsschlüssel für `s3cmd` erhält:

```
$ ~/cumulus/bin/cumulus-add-user s3test@thesis.de  
ID          : 95i2AHInRXpflIdGyGmrV  
password    : ICTIwzV8NFkxyYXTsNdHpUWvIJr4WQn4pv5upV0pLG  
quota       : None  
canonical id : 2b5d74ca-6e22-11e7-a9ae-068ad1aada5b
```

¹⁷ Das `&`-Zeichen am Ende des Befehls sorgt dafür, dass der Befehl nicht im Vordergrund läuft bzw. die Konsole weiter genutzt werden kann.

s3cmd einrichten

In der `.s3cfg` werden anschließend folgende Zeilen bearbeitet, damit der Client korrekt mit Cumulus kommuniziert:

```
1 access_key = 95i2AHInRXpf1IdGyGmrV
2 host_base = <ip_of_vm>:8888
3 host_bucket = <ip_of_vm>:8888
4 secret_key = ICTIwzV8NFkxyYXTsNdHpUWvIJr4WQn4pv5upV0pLG
5 use_https = False
```

Beim Erstellen eines neuen Buckets ist zu beachten, dass dessen Name mit einem Großbuchstaben beginnt, da eine Fehlermeldung ausgegeben wird, wenn der Name des Buckets nur aus Kleinbuchstaben besteht:

```
$ s3cmd mb s3://s3-mt-testbucket
WARNING: Retrying failed request: /
WARNING: 500 (InternalServerError): We encountered an internal error. Please try again.
WARNING: Waiting 3 sec...
```

Besteht der Name z.B. aus `S3-MT-TESTBUCKET` wird der Bucket erstellt und kann mit Daten befüllt sowie heruntergeladen und gelöscht werden.

5.2.5 OpenStack Swift

OpenStack ist, ähnlich zu Eucalyptus oder Nimbus, eine IaaS, welche z.B. von der Deutschen Telekom, Wikimedia und dem CERN genutzt wird. OpenStack setzt sich aus vielen einzelnen Komponenten zusammen, wie z.B. dem Erzeugen von VMs und dem Bereitstellen von Speicher.

Das Bereitstellen von Speicher übernimmt die Komponente *Swift*, der Objektspeicher von OpenStack, womit über eine einfache API viele unterschiedliche Daten verwaltet werden können. Swift bietet eine sehr gute Skalierung und wurde auf Haltbarkeit, Erreichbarkeit und Parallelität des gesamten Datensatzes hin optimiert. Damit ist Swift ideal für die Speicherung von unstrukturierten Daten geeignet. [67]

Ein aktuelles Thema ist die Automatisierung von Prozessen im Bereich der IT-Infrastruktur. Mit der Ruby-Anwendung *Vagrant* können VMs erstellt und anschließend automatisch verteilt und installiert werden. Geschaffen wurde dieses Programm ursprünglich für die verschiedenen Virtualisierungsprogramme, jedoch funktioniert Vagrant auch mit den AWS.

Eine besonders zuverlässige und einfache Methode ist die Installation mit Hilfe von SAIO – *Swift All In One* [66]. Mit dem Projekt `vagrant-swift-all-in-one` [80] gibt es eine vorgefertigte VM, welche nur wenige Anpassungen benötigt. Vagrant muss einerseits lokal auf dem System installiert werden und besteht andererseits meist aus einer Konfigurationsdatei, dem `Vagrantfile`, das an Vagrant übergeben wird und Anweisungen zum Ausführen enthält. Das Besondere an „Vagrant-SAIO“ ist, dass es noch eine zusätzliche Datei namens `localrc` enthält, welche die individuelle Konfiguration an das `Vagrantfile` übergibt, wodurch es nicht nötig ist, das `Vagrantfile` selbst zu bearbeiten.

Vagrant

Durch die bereits implementierte Unterstützung für die AWS, kann die `localrc` so angepasst werden, dass Vagrant die VM direkt in eine Instanz der EC2 verteilt und installiert (siehe Listing 5.5).

Quellcode 5.5: Die `localrc.aws` übergibt die Parameter an das `Vagrantfile` und kann so direkt eine VM und alle Installation innerhalb der AWS ausführen.

```
1 export AWS_ACCESS_KEY_ID=<access_key>
2 export AWS_SECRET_ACCESS_KEY=<secret_key>
3 export AWS_REGION=eu-central-1
4 export AWS_INSTANCE_TYPE=t2.micro
5 export AWS_AMI=ami-1c45e273
6 export AWS_KEYPAIR_NAME=<key-pair>
7 export SSH_PRIVATE_KEY_PATH=/path/to/key.pem
8 export AWS_ELASTIC_IP=false
9 export AWS_SECURITY_GROUPS=ssh-only
10 export VAGRANT_BOX=dummy
```

Zeilen 1 und 2 sind die Zugangsschlüssel für AWS, Zeilen 3 bis 5 geben an, in welcher Region welcher Instanzen-Typ mit welchem Betriebssystem-Abbild (in diesem Fall Ubuntu 16.04) erzeugt wird. Zeilen 6 und 7 konfigurieren den SSH-Schlüssel, 8 und 9 geben an, dass keine Elastic-IP und die Sicherheitsgruppe „swift“ verwendet werden soll. Die letzte Zeile gibt die Box¹⁸ an, die Vagrant als Grundlage nutzt.

Durch den Inhalt der `localrc.aws` weiß Vagrant sofort, dass eine Instanz auf Amazon erstellt werden soll. Bei einer anderen Konfiguration wird die VM z.B. für Virtualbox erstellt.

¹⁸ Boxen sind bei Vagrant vorkonfigurierte VMs, um den Prozess der Verteilung und Entwicklung zu beschleunigen.

Für Vagrant existiert unter Ubuntu ein Paket, welches mit

```
$ sudo apt update
$ sudo apt install vagrant
```

auf dem lokalen System installiert werden kann.

SAIO installieren

Anschließend wird das Projekt von Github geklont und in den Ordner gewechselt:

```
$ git clone https://github.com/swiftstack/vagrant-swift-all-in-one.git
$ cd ~/vagrant-swift-all-in-one
```

Mit

```
$ cp localrc-template localrc.aws
$ nano localrc.aws
$ source localrc.aws
```

wird die Datei `localrc-template` als `localrc.aws` kopiert, mit `nano`¹⁹ bearbeitet und der Inhalt wird mit `source` an die aktuelle Shell übergeben. In diesem Fall wird dort der Inhalt aus Listing 5.5 eingefügt, sodass Vagrant den Prozess auf Amazon ausführt. Sind alle Daten korrekt, kann mit dem Befehl

```
$ vagrant up
```

innerhalb des Projektordners der Prozess gestartet werden. Dabei wird eine Instanz mit dem angegebenen Betriebssystem aufgesetzt und anschließend mit `Chef Solo` die Cookbooks²⁰ des Projekts installiert. Bei `Chef Solo` handelt es sich um einen weiteren Automatismus, der Skripte (Recipes²¹) innerhalb des Cookbooks, ausführt. [25]

`Chef` installiert das Recipe „swift“, welches nichts anderes tut, als die Schritte aus der Anleitung zu SAIO [66] durchzuführen und Swift auf dem System zu installieren. Dieser Vorgang nimmt nur wenige Minuten in Anspruch und bestätigt die erfolgreiche Installation mit der Meldung:

```
...
==> default: [2017-07-22T16:45:52+00:00] INFO: Chef Run complete in 2
      209.162444768 seconds
==> default:
==> default: Running handlers:
```

¹⁹ `nano` ist ein einfacher Texteditor für die Kommandozeile.

²⁰ Ein Cookbook kann ein oder mehrere Recipes enthalten.

²¹ Ein Recipe beinhaltet mehrere Skripte, die z.B. ein Programm automatisch installieren.

```
==> default: [2017-07-22T16:45:52+00:00] INFO: Running report handlers
==> default: Running handlers complete
==> default:
==> default: [2017-07-22T16:45:52+00:00] INFO: Report handlers complete
==> default: Chef Client finished, 173/187 resources updated in 03
minutes 30 seconds
```

Es ist möglich, direkt durch Vagrant auf die Instanz zuzugreifen. Mit

```
$ vagrant ssh
```

wird eine SSH-Verbindung aufgebaut, um mit dem System zu interagieren. Sollte Swift z.B. nicht mehr ordnungsgemäß laufen, kann mit `$ reinstallswift` ein Skript gestartet werden, welches Swift neu installiert. In diesem Projekt existieren mehrere solcher Skripte, welche bei Problemen manche Schritte vereinfachen.

Swift-Client einrichten

Swift kann nicht mit `s3cmd` zusammenarbeiten und stellt eine eigene API zur Verfügung. Im Ökosystem von Swift werden die Buckets, wie bei Azure, Container genannt. Doch zuvor müssen der aktuellen Shell die Zugangsdaten für den Client mitgeteilt werden:

```
$ export ST_AUTH=http://<vm_ip>:8080/auth/v1.0
$ export ST_USER=test:tester
$ export ST_KEY=testing
```

Damit weiß der Swift-Client, wohin und mit welchen Daten sich dieser verbinden soll. Um mit diesem zu interagieren, können folgende Befehle ausgeführt werden:

```
$ swift post testcontainer
$ swift upload testcontainer testfile.txt
$ swift list testcontainer
$ swift download testcontainer testfile.txt
$ swift delete testcontainer testfile.txt
$ swift delete testcontainer
```

Die API ist ähnlich zu `s3cmd`, lediglich die Befehle sind etwas anders. Mit `$ swift stat` kann der aktuelle Status von Swift bzw. dessen Inhalt überprüft werden.

5.2.6 Riak Cloud Storage

Riak CS ist ein einfach zu nutzender Cloud-Speicher, welcher auf Riak KV, die verteilte NoSQL-Datenbank von Riak, aufsetzt. Riak CS wurde entworfen um einfache, verfügbare und verteilte Cloud-Speicher zu bieten, die skalierbar sind und für ganze Cloud-Architekturen oder Speicher-Infrastrukturen für hochleistungsfähige Anwendungen und Dienste genutzt werden können. Die API von Riak CS ist ebenfalls S3-kompatibel. [15]

Pakete installieren

Riak CS benötigt für den Betrieb die Datenbank Riak KV sowie Stanchion zur Serialisierung²² von Anfragen. Um die Installation aller Komponenten zu vereinen, kann Docker genutzt werden. Docker ist die meistgenutzte Container-Software der Welt und kann Applikationen in isolierten Containern ausführen. Dadurch ist es auch möglich z.B. die gleiche Applikation mehrmals in unterschiedlichen Containern, im Fall von Riak auch mit verschiedenen Ports, auszuführen. [35]

Für Riak CS existiert ein Github-Projekt [45], bei dem Riak KV, Stanchion und Riak CS in einem Docker-Container zusammengefasst wurden und für die Installation nur noch dieser Container auf dem System installiert und ausgeführt werden muss.

Docker selbst und Riak besitzen Abhängigkeiten, welche installiert werden müssen:

```
$ sudo apt update
$ sudo apt install build-essential autoconf libncurses5-dev openssl \
  libssl-dev fop xsltproc unixodbc-dev libpam0g-dev git
$ sudo apt install apt-transport-https ca-certificates curl software-properties-common
```

Um die Paketquelle für Docker einzupflegen, ist es notwendig den offiziellen GPG²³-Schlüssel mit

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

hinzuzufügen. Anschließend kann die Paketquelle mit

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

²² Umwandlung eines Objekts in einen Bytestrom [88].

²³ GPG ist der GNU Privacy Guard und ein freies Kryptographiesystem.

als Paketzweig `stable` hinzugefügt werden.

Um nun Docker zu installieren, sollten zuvor die Paketquellen auf den aktuellen Stand gebracht werden:

```
$ sudo apt update
$ sudo apt install docker-ce
```

Danach kann die Community Edition von Docker (`docker-ce`) installiert werden. Mit

```
$ sudo docker run hello-world

Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
b04784fba78d: Pull complete
Digest: sha256:2f3b3b28a45160805bb16542c9531888519430e9e6d6ffc09d72261b0d26ff74f
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.
...
```

kann überprüft werden, ob Docker ordnungsgemäß installiert wurde. Dabei wird ein Container heruntergeladen, gestartet, ein Text ausgegeben und der Container wieder beendet. [34]

Für Riak ist es sinnvoll, das Limit für offene Dateien anzuheben, da sonst beim Starten der Dienste Warnungen angezeigt werden, dass dies geändert werden muss [14]. Dazu muss die Datei

```
$ sudo nano /etc/security/limits.conf
```

bearbeitet und am Ende mit

```
ubuntu soft nofile 65536
ubuntu hard nofile 65536
```

erweitert werden, womit das Limit für den Benutzer `ubuntu` auf 65536 Dateien angehoben wird.

Docker Riak CS installieren

Docker Riak CS ist ein Projekt auf Github, was zu Beginn auf das System geklont werden muss:

```
git clone https://github.com/ianbytchek/docker-riak-cs.git
```

Anschließend wird der Container neu gebaut, damit dieser die aktuellste Version der Komponenten enthält. Durchgeführt werden kann dies mit:

```
$ cd ~/docker-riak-cs
$ mkdir src
$ sudo docker build --tag 'ianbytchek/riak-cs' './src'
```

Dabei werden die Abhängigkeiten, die für diesen Container benötigt werden, zusammen mit den Riak-Komponenten heruntergeladen und zu einem Container gebaut. Sollten während der Prozedur Fehler auftreten oder Abhängigkeiten fehlen, kann mit der Option `--no-cache` das Bauen des Containers neu gestartet werden, da sonst die Daten aus dem Cache²⁴ benutzt werden und die gleichen Fehler erneut auftreten.

Docker Riak CS starten

Wenn der Container fehlerfrei gebaut wurde, kann dieser durch

```
$ docker run --detach --env 'RIAK_CS_BUCKETS=testbucket' --publish 8080:8080 --name 'riak-cs' ianbytchek/riak-cs
```

gestartet werden. Die Option `--detach` sorgt dafür, dass der Container im Hintergrund ausgeführt wird. Mit `--env 'RIAK_CS_BUCKETS=testbucket'` wird bereits während dem Start des Containers ein Bucket mit übergeben. Durch `--publish '8080:8080'` wird der Container angewiesen, auf Port 8080 zu laufen. Der Name des Containers und die Quelle werden mit `--name 'riak-cs' ianbytchek/riak-cs` festgelegt.

Beim Starten des Containers werden automatisiert die Zugangsschlüssel für `s3cmd` generiert, welche jedoch nicht angezeigt werden. Diese können über das Log²⁵ vom Start des Containers eingesehen werden:

```
$ sudo docker logs -f 'riak-cs'

Update data permissions in case it's mounted as volume... OK!
Starting Riak... OK!
```

²⁴ Ein Cache ist ein schneller Puffer-Speicher, um Neuberechnungen zu vermeiden.

²⁵ Protokoll von Ereignissen eines Programms.

```
Waiting for riak kv service to startup... OK!
Starting Stanchion... OK!
Starting Riak CS... OK!

#####

Riak admin credentials, make note of them, otherwise you
will not be able to access your files and data. Riak
services will be restarted to take effect.

Access key: SI3HATL0K6HI38PJQNKT
Secret key: HMrHDS3zSr9RsUfz9M76wxI6hpQsphhASlroiQ==

#####

Restarting Riak... OK!
Waiting for riak kv service to startup... OK!
Restarting Stanchion... OK!
Restarting Riak CS... OK!
Creating Riak CS buckets.
testbucket... OK!
...
```

Über dieses Log ist außerdem ersichtlich, ob alle Dienste fehlerfrei gestartet und der angegebene Bucket erstellt wurde.

Mit dem Aufruf von `nmmap`²⁶

```
$ nmap localhost
...
PORT      STATE SERVICE
22/tcp    open  ssh
8080/tcp   open  http-proxy
```

kann verifiziert werden, dass Riak CS tatsächlich auf seinem Port läuft, was an dieser Stelle hilfreich ist, da es keine Meldung gibt, ob die Dienste ordnungsgemäß arbeiten. Um zu prüfen, wie viel Arbeitsspeicher die Dienste belegen, kann `$ free -h` ausgeführt werden:

```
$ free -h
              total used    free   shared  buff/cache   available
Mem:    990M   500M   77M    4.6M    413M         288M
Swap:    0B      0B      0B
```

²⁶ Mit `nmmap` kann nach offenen Ports auf einem System gescannt werden.

s3cmd einrichten

Wie bei den vorherigen Cloud-Speicherdiensten gestaltet sich die Konfiguration von s3cmd ebenfalls ähnlich. Wichtig ist die Option `signature_v2 = True`, da Riak CS nicht mit Version 3 der S3-Signatur zurechtkommt [16]:

```
1 access_key = SI3HATL0K6HI38PJQNKT
2 host_base = <ip_of_vm>:8080
3 host_bucket = <ip_of_vm>:8080
4 secret_key = HMrHDS3zSr9RsUfz9M76wxI6hpQsphhASlroiQ==
5 use_https = False
6 signature_v2 = True
```

Anschließend kann mit

```
$ s3cmd ls
s3://testbucket
```

der Inhalt des Speichers angezeigt werden. Da ein Bucket erstellt wurde, wird dieser angezeigt und s3cmd wurde korrekt konfiguriert. Weitere Buckets können erstellt, Daten können hoch- und heruntergeladen sowie gelöscht werden. Mit Hilfe von Docker kann Riak CS sehr schnell und ressourcenschonend verteilt und installiert werden.

5.2.7 S3 Ninja

S3 Ninja ist ein Speicherdienst, der auf Java basiert und die S3-API für Entwicklungs- und Testzwecke emuliert [76].

Pakete installieren

Da S3 Ninja auf Java basiert, muss dies zuvor auf dem System installiert werden. Dazu kann unter Ubuntu `openjdk`²⁷ in Version 9 genutzt werden. Außerdem ist es möglich, dass das Paket `unzip` nicht auf dem System existiert und ebenfalls installiert werden muss.

```
$ sudo apt install unzip openjdk-9-jre
```

²⁷ OpenJDK ist die Open Source Variante von Java.

S3 Ninja installieren

Für den Speicherdienst wird ein Ordner erstellt, das Paket heruntergeladen und entpackt sowie anschließend gelöscht:

```
$ mkdir ~/s3ninja && cd ~/s3ninja
$ wget https://oss.sonatype.org/content/groups/public/com/scireum/
  s3ninja/2.7/s3ninja-2.7-zip.zip
$ unzip s3ninja-2.7-zip.zip
$ rm -rf s3ninja-2.7-zip.zip
```

Damit S3 Ninja Daten speichern kann, sollte ein Ordner, wie z.B. s3, dafür erstellt werden:

```
$ mkdir ~/s3ninja/data/s3
```

S3 Ninja starten

Im Anschluss an die Vorbereitungen, kann der Server gestartet werden. Dies kann durch die Eingabe von

```
$ ./sirius.sh start

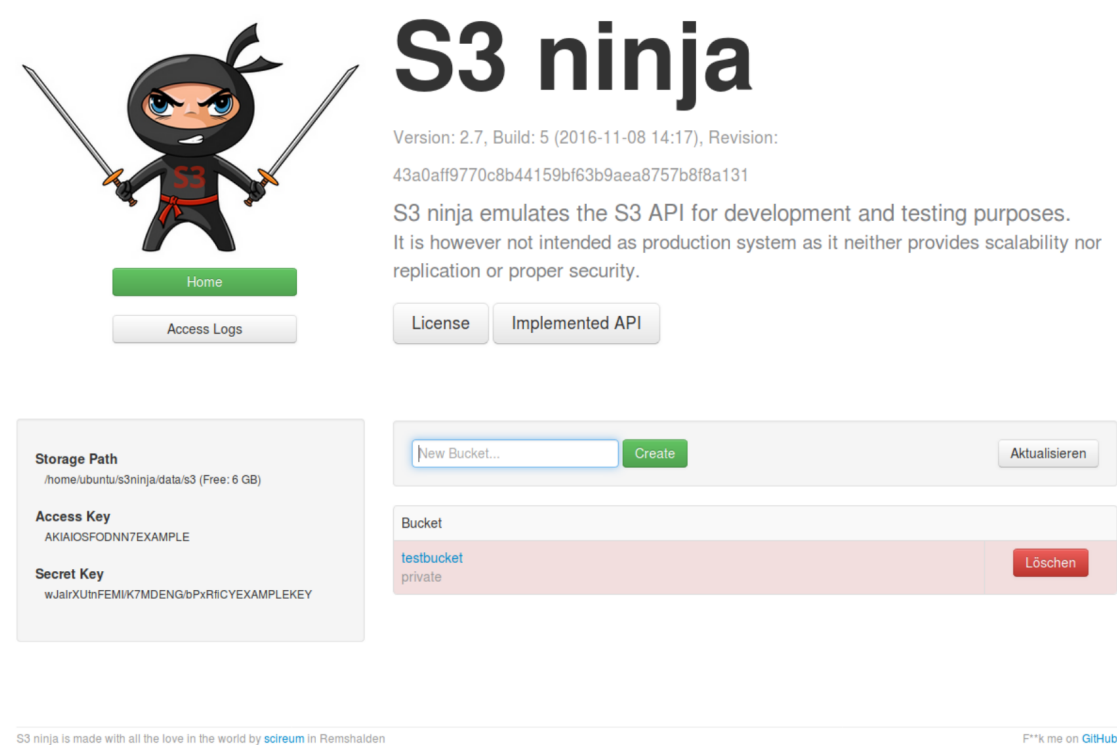
SIRIUS Launch Utility
=====

Use a custom config.sh to override the settings listed below

SERVICE:
SIRIUS_HOME:    /home/ubuntu/s3ninja
JAVA_CMD:       java
JAVA_OPTS:      -server -Xmx1024m -Djava.net.preferIPv4Stack=true
SHUTDOWN_PORT:  9191
STDOUT:         logs/stdout.txt
USER_ID:

Starting Application...
```

durchgeführt werden. Das Skript gibt beim Start diverse Informationen aus, die anzeigen, wie der Server momentan konfiguriert ist. Mit der `config.sh` können diese Werte angepasst werden. Des Weiteren bietet S3 Ninja ein Webinterface, worüber die Buckets und Daten verwaltet werden können (siehe Abbildung 5.12).



Version: 2.7, Build: 5 (2016-11-08 14:17), Revision: 43a0aff9770c8b44159bf63b9aea8757b8f8a131

S3 ninja emulates the S3 API for development and testing purposes. It is however not intended as production system as it neither provides scalability nor replication or proper security.

License Implemented API

Storage Path
/home/ubuntu/s3ninja/data/s3 (Free: 6 GB)

Access Key
AKIAIOSFODNN7EXAMPLE

Secret Key
wJalrXUtnFEMIK7MDENG/bPxRiCYEXAMPLEKEY

New Bucket... Create Aktualisieren

Bucket

testbucket	private	Löschen
------------	---------	---------

S3 ninja is made with all the love in the world by [scireum](#) in Remshalden F**k me on [GitHub](#)

Abbildung 5.12: Das Webinterface von S3 Ninja zeigt die Zugangsschlüssel sowie Buckets und Dateien an, welche über die Oberfläche ebenfalls bearbeitet werden können.

s3cmd einrichten

Um zum Datenaustausch s3cmd nutzen zu können, sollte ein Webserver wie nginx als Proxy²⁸ genutzt werden. Das ist notwendig, da die API nur den Zugriff auf das Wurzelverzeichnis / erlaubt, was S3 Ninja jedoch nicht zulässt. Als Lösung sollte die Adresse mit /s3 erweitert werden. [32]

nginx wird durch

```
$ sudo apt install nginx
```

installiert. Anschließend muss die Konfiguration mit `sudo nano /etc/nginx/sites-available/default` angepasst werden, sodass nginx als Proxy fungiert und die Adresse von S3 Ninja weiterleitet:

```
1 server {  
2     listen 8080;  
3     listen [::]:8080;  
4 }
```

²⁸ Ein Proxy ist eine Netzwerkschnittstelle und fungiert als Vermittler zwischen zwei Systemen.

```
5  server_name s3.eu-central-1.amazonaws.com;
6
7  location / {
8      proxy_pass http://172.31.38.76:9444/s3/;
9  }
10 }
```

Ein Neustart von nginx ist nach der Änderung der Konfiguration notwendig:

```
$ sudo /etc/init.d/nginx restart
```

Nach der Fertigstellung von nginx, kann s3cmd mit den Daten von S3 Ninja konfiguriert werden. Die Zugangsschlüssel sind über die Datei `~/s3ninja/app/application.conf` und über das Webinterface einsehbar.

```
1  access_key = AKIAIOSFODNN7EXAMPLE
2  host_base = <ip_of_vm>:8080
3  host_bucket = <ip_of_vm>:8080
4  secret_key = wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
5  use_https = False
```

5.2.8 S3rver

S3rver ist ein leichtgewichtiger Speicherserver, ähnlich zu Fake S3, der Befehle der S3-API beherrscht. S3rver ist sehr nützlich, um einen S3-Dienst in einer Sandbox zu testen, ohne aktiv mit Amazon zu kommunizieren. [47]

Pakete installieren

S3rver basiert auf node.js, welches auch auf dem System vorhanden sein sollte, damit S3rver betrieben werden kann. Über npm²⁹ wird ein S3rver-Paket offiziell zum Installieren angeboten, sodass npm installiert werden muss:

```
$ sudo apt install npm
```

Weitere Abhängigkeiten werden für den Betrieb von S3rver nicht benötigt.

²⁹ Der Node Package Manager (npm) ist ein Paketmanager für node.js.

S3rver installieren

Um anschließend S3rver zu installieren, sollte

```
$ sudo npm install s3rver -g
```

ausgeführt werden, da npm auch Zugriff auf Verzeichnisse benötigt, auf welche nur root Zugriff hat. Die Option `-g` bedeutet, dass das Paket als globales Paket, welches überall verfügbar ist, installiert wird. Als Ausgabe wird angezeigt, welche zusätzlichen Bibliotheken und Pakete installiert wurden, damit S3rver betrieben werden kann.

Um nach der Installation von S3rver auf den Befehl zuzugreifen und Fehlermeldungen wie

```
$ s3rver --help
/usr/bin/env: 'node': No such file or directory
```

zu vermeiden, sollte ein Symlink³⁰ hinzugefügt werden:

```
$ ln -s /usr/bin/nodejs /usr/bin/node
```

Nach der Verlinkung funktionieren die Befehle:

```
$ s3rver --help

Usage: s3rver [options]

Options:

--version                output the version number
-h, --hostname [value]  Set the host name or ip for the server
-p, --port <n>          Set the port of the http server
-s, --silent            Suppress log messages
-i, --indexDocument [path] Index Document for Static Web Hosting
-e, --errorDocument [path] Custom Error Document for Static Web Hosting
-d, --directory [path]  Data directory
-c, --cors              Enable CORS
-h, --help              output usage information
```

³⁰ Eine symbolische Verknüpfung verweist auf eine andere Datei oder ein anderes Verzeichnis im Dateisystem [81].

S3rver starten

S3rver benötigt noch ein Verzeichnis, worin Buckets und Daten gespeichert werden können:

```
$ mkdir ~/s3rver
```

Anschließend kann S3rver mit der IP der VM (bei AWS die interne IP auf eth0) und einem Port über den Befehl `$ s3rver -h <ip_of_vm> -p <port> -d ~/<dir>/` gestartet werden:

```
$ s3rver -h 172.31.33.239 -p 8080 -d ~/s3rver/  
now listening on host 172.31.33.239 and port 8080  
...
```

s3cmd einrichten

Bei S3rver ist es wichtig, dass die Namen der Buckets komplett kleingeschrieben sind, da sonst eine Fehlermeldung angezeigt wird. Die Zugangsschlüssel sind, ähnlich zu Fake S3, auf 123 und abc festgelegt und bisher ist keine Möglichkeit bekannt, diese zu ändern.

```
1 access_key = 123  
2 host_base = <ip_of_vm>:8080  
3 host_bucket = <ip_of_vm>:8080  
4 secret_key = abc  
5 use_https = False
```

Anschließend ist es möglich, wie bei allen anderen Diensten, über s3cmd mit S3rver zu kommunizieren.

5.2.9 Scalify CloudServer

Der Scalify CloudServer (vormals S3-Server) ist ein Amazon S3-kompatibler Objektspeicher, mit dem Applikationen schneller erstellt, integriert und Daten überall gespeichert werden können [74].

Pakete installieren

Für die Arbeit mit dem CloudServer werden bestimmte Pakete benötigt:

```
$ sudo apt install python build-essential
```

Scality CloudServer basiert auf node.js Version 6, wobei diesmal eine andere Quelle [64] für die Installation genutzt wird, da dies in der Dokumentation [74] ausdrücklich empfohlen wird.

```
$ curl -sL https://deb.nodesource.com/setup_6.x | sudo -E bash -  
$ sudo apt install nodejs
```

Mit diesen Befehlen wird node.js 6.x den Paketquellen zusammen mit dem Sicherheits-schlüssel hinzugefügt. Danach kann das Paket über den Paketmanager von Ubuntu installiert werden. Mit

```
$ nodejs --version  
v6.11.1  
$ npm -version  
3.10.10
```

kann überprüft werden, ob es erfolgreich installiert wurde und welche Versionen sich von node.js und npm auf dem System befinden.

Der CloudServer benötigt mehr als 1 GB Arbeitsspeicher, um problemlos laufen zu können [27]. Um dieses Problem auf einer Maschine mit nur 1 GB Arbeitsspeicher zu beheben, kann eine 2 GB große Swap³¹-Datei erstellt werden:

```
$ sudo mkdir -p /var/cache/swap  
$ sudo fallocate -l 2G /var/cache/swap/swap0  
$ sudo chmod 0600 /var/cache/swap/swap0  
$ sudo mkswap /var/cache/swap/swap0  
$ sudo swapon /var/cache/swap/swap0
```

Damit der Swap-Speicher auch aktiv genutzt werden kann, muss ein Eintrag in der `/etc/fstab` hinzugefügt werden:

```
/var/cache/swap/swap0    none    swap    sw    0 0
```

³¹ Swap ist ein Bereich auf einem Speichermedium, welcher zum virtuellen Speicher gehört und dem Arbeitsspeicher hinzugefügt werden kann.

Scality CloudServer installieren und starten

Um den CloudServer zu installieren, muss zuvor der Quellcode über Github geklont

```
$ git clone https://github.com/scality/S3.git
$ cd ~/S3/
```

und anschließend mit npm kompiliert und installiert werden:

```
$ npm install
```

Im Anschluss daran kann der CloudServer mit

```
$ npm start
```

auf Port 8000 gestartet werden. Dies dauert einen kurzen Augenblick, wobei viele Daten über den Startvorgang ausgegeben werden.

s3cmd einrichten

Die Zugangsschlüssel für den CloudServer befinden sich in der Konfigurationsdatei `~/S3/conf/authdata.json`. Mit

```
$ nano ~/S3/conf/authdata.json
```

kann die Datei eingesehen und die Daten können in die `.s3cfg`, zur Kommunikation über s3cmd, eingetragen werden:

```
access_key = accessKey1
host_base = <ip_of_vm>:8000
host_bucket = <ip_of_vm>:8000
secret_key = verySecretKey1
use_https = False
```

6 Redundanz durch verteilte Dateisysteme

Genauso wichtig wie die Datenintegrität ist die Mehrfachhaltung von Daten innerhalb der Cloud. Dies verringert den Datenverlust bei Ausfall eines Speichers, da die Daten weiterhin an einem anderen Ort zur Verfügung stehen. Dies ist heutzutage, in Zeiten von Verschlüsselungstrojanern und anderen Angriffen von Außen sehr wichtig, denn Datenverluste können erhebliche finanzielle und zeitliche Einbußen zur Folge haben.

Um die Daten, welche über einen S3-Speicherdienst hoch- und heruntergeladen werden, auf mehrere Orte zu verteilen, wurde ein verteiltes Dateisystem eingeführt, das für mehr Redundanz sorgt. Da es sich hierbei um ein reines Testszenario handelt, wurde nur ein entfernter Speicher eingeführt, worauf alle Speicherdienste gleichermaßen zugreifen.

6.1 Verteilte Dateisysteme

Verteilte Dateisysteme sind Dateisysteme, die z.B. Daten über mehrere Datenträger bzw. Speichersysteme verteilen. Dies bedeutet, dass ein Speicherdienst seine Daten lokal in einem Unterordner ablegt und dort verwaltet. Befindet sich dieser Ordner auf einem Laufwerk, das mit einem verteilten Dateisystem betrieben wird, werden die Daten gleichzeitig auch auf dem entfernten Speicher gesichert und sind so doppelt vorgehalten, sollte der lokale Speicher ausfallen. In produktiven Systemen werden oftmals noch mehrere entfernte Speicher installiert, damit die Daten n-fach vorhanden sind. Auf diese Weise geht die Wahrscheinlichkeit eines Datenverlusts gegen Null.

Wie bereits angedeutet, ist eine n-fache Datenhaltung im Fall dieser Arbeit nicht notwendig, da ein verteiltes Dateisystem nur zu Testzwecken installiert wird, um den Unterschied der Leistungsfähigkeit mit und ohne verteiltem Dateisystem zu betrachten.

6.1.1 GlusterFS

Es existieren unterschiedliche Implementierungen mit unterschiedlichem Leistungsumfang, die als verteiltes Dateisystem genutzt werden können. Als besonders leistungsfähig und einfach zu handhaben hat sich *GlusterFS* herausgestellt. GlusterFS ist ein skalierbares verteiltes Dateisystem, womit große verteilte Speicherlösungen für das Streamen von Medien, Datenanalysen und anderen datenintensiven Aufgaben erstellt werden können. GlusterFS ist eine freie Open-Source-Software, wodurch der Quellcode eingesehen und verändert werden kann. [40]

Damit das System als verteilt bezeichnet werden kann, existieren ein oder mehrere Server und verschiedene Clients. Der Server selbst sollte auf einem System betrieben werden, welches sich im besten Fall an einem anderen Standort als die Clients befindet.

6.1.2 GlusterFS-Server installieren

GlusterFS ist aufgrund der einfachen Installation und Konfiguration sehr beliebt. Zu Beginn wird GlusterFS unter Ubuntu durch

```
$ sudo add-apt-repository ppa:gluster/glusterfs-3.8
$ sudo apt update
$ sudo apt install software-properties-common glusterfs-server
```

den Paketquellen hinzugefügt und installiert. Dem GlusterFS-Server muss noch ein weiterer Datenträger hinzugefügt werden, welcher als Speicherort für das Dateisystem fungiert. Der Datenträger, auf den GlusterFS zugreift, muss ein anderer als der des Betriebssystems sein, da der Datenträger in jedem Fall formatiert wird. Wenn sich der Server innerhalb der AWS befindet, wird mit

```
$ sudo fdisk /dev/xvdb
```

der Datenträger entsprechend der Eingaben formatiert¹. In den meisten Fällen können hierbei die Standardvorgaben akzeptiert werden. Im Anschluss daran wird die erste Partition, die zuvor erstellt wurde, formatiert:

```
$ sudo mkfs.xfs -i size=512 /dev/xvdb1
```

1 Innerhalb anderer Dienste kann die Bezeichnung der Datenträger abweichen.

Damit der Datenträger in das System eingebunden werden kann, wird ein zusätzlicher Eintrag in die `/etc/fstab` eingefügt:

```
$ sudo echo "/dev/xvdb1 /export/xvdb1 xfs defaults 0 0" >> /etc/fstab
```

Nun können die Ordner erstellt und der Datenträger eingebunden werden:

```
$ sudo mkdir -p /export/xvdb1 && mount -a && mkdir -p /export/xvdb1/brick
```

In Produktivsystemen ist es sinnvoll, mehrere GlusterFS-Server zu betreiben. In dem nächsten Befehl zum Erzeugen eines replizierenden Datenträgers können auch mehrere Server mit der Option `replica 2`² zusammengefasst werden. Zu Testzwecken wird jedoch nur ein Server zur Sicherung der Daten verwendet:

```
$ sudo gluster volume create gv0 ip-172-31-47-142:/export/xvdb1/brick
```

Damit wird ein Datenträger mit dem Namen `gv0` erstellt, welcher unter dem Hostname `ip-172-31-47-142` unter `/export/xvdb1/brick` zu finden ist. Alle Informationen zum neu erstellten Datenträger können mit

```
$ sudo gluster volume info

Volume Name: gv0
Type: Distribute
Volume ID: f521ae2e-507f-446d-89c9-0e7831472a19
Status: Started
Snapshot Count: 0
Number of Bricks: 1
Transport-type: tcp
Bricks:
Brick1: ip-172-31-47-142:/export/xvdb1/brick
Options Reconfigured:
transport.address-family: inet
performance.readdir-ahead: on
nfs.disable: on
```

abgerufen werden. Um den Datenträger mit GlusterFS zu starten, wird der Befehl

```
$ sudo gluster volume start gv0
```

ausgeführt. Damit läuft ein GlusterFS-Server, dessen Laufwerk über die öffentliche IP und den Namen des Datenträgers auf Servern und Clients eingebunden werden kann.

² Die 2 gibt hier die Anzahl der Server an.

6.1.3 GlusterFS-Client installieren

Der GlusterFS-Client kann nun z.B. auf einem Server mit einem Cloud-Speicherdienst installiert werden, wo dieser das entfernte Laufwerk als Ordner für die Daten einbindet.

Zu Beginn sollte sichergestellt werden, dass der Kernel das benötigte Modul fuse³ unterstützt:

```
$ modprobe fuse
$ dmesg | grep -i fuse
[    0.729675] fuse init (API version 7.26)
```

Der Ablauf der Installation des Clients ist ähnlich zu der Installation des Servers.

```
$ sudo add-apt-repository ppa:gluster/glusterfs-3.8
$ sudo apt update
$ sudo apt install glusterfs-client
```

Nach der Installation kann das entfernte Laufwerk mit

```
$ sudo mkdir /mnt/glusterfs
$ sudo mount -t glusterfs 54.93.235.26:/gv0 /mnt/glusterfs
```

eingebunden werden. Der Ordner, der als Einbindepunkt dienen soll, sollte vor Ausführen des Befehls erstellt werden.

6.2 Cloud-Speicherdienste mit GlusterFS

Generell können alle Dienste mit einem entfernten Dateisystem betrieben werden, da der entfernte Datenträger auf dem System lediglich einen Ordner darstellt. Der Unterschied ist jedoch, dass sich dieser Ordner mit dem entfernten Server(n) synchronisiert.

6.2.1 Minio

Am Beispiel von Minio wird GlusterFS eingebunden. Der Client wird, wie in Kapitel 6.1.3 beschrieben, installiert und anschließend wird das entfernte Laufwerk eingebunden.

³ FUSE ist ein Akronym und bedeutet Filesystem in USErspace.

Um Minio mit dem eingebundenen Laufwerk von GlusterFS zu starten, wird der Startbefehl ein wenig verändert:

```
$ ./minio server --address ":8080" /mnt/glusterfs
```

Somit wird der Ordner benutzt, in welchen der GlusterFS-Server eingebunden wurde. Alle Daten werden dadurch mit dem entfernten Server synchronisiert.

Nach diesem Schema können die meisten Speicherdienste mit einem verteilten Dateisystem betrieben werden. Unter Umständen ist es notwendig, eine Konfigurationsdatei anzupassen.

Wie in Tabelle 6.1 aufgezeigt wird, hat die Umstellung auf ein verteiltes Dateisystem so gut wie keinen Einfluss auf die Leistungsfähigkeit des Speicherdienstes. Dies liegt vor allem daran, dass das eingebundene Laufwerk einen normalen Ordner darstellt und die Daten im Hintergrund synchronisiert, wovon der Speicherdienst nicht betroffen ist.

Tabelle 6.1: Leistungsvergleich der Gesamtzeiten in Sekunden von Minio ohne und mit GlusterFS.

Gesamtzeiten [s]	
Minio Standard	Minio GlusterFS
8.905	9.325
15.078	14.748
25.718	26.217
49.106	47.108
93.172	91.206

6.2.2 Scality CloudServer

Neben Minio wird auch der Scality CloudServer zusammen mit GlusterFS getestet. Die Installation des GlusterFS-Client gestaltet sich genauso einfach wie bei Minio.

Damit der CloudServer das eingebundene Laufwerk verwendet, muss dies dem Speicherdienst mitgeteilt werden. Am einfachsten geht das über den export der Variablen für den Speicherort:

```
$ sudo mkdir -p -m 700 /mnt/glusterfs/cloudserver/data
$ sudo mkdir -p -m 700 /mnt/glusterfs/cloudserver/metadata
$ export S3DATAPATH="/mnt/glusterfs/cloudserver/data"
```

```
$ export S3METADATAPATH="/mnt/glusterfs/cloudserver/metadata"  
$ npm start
```

Anschließend speichert der CloudServer sämtliche Daten direkt in das eingebundene Laufwerk. Tests mit s3perf ergaben ein ähnliches Ergebnis wie bei Minio (siehe Tabelle 6.2).

Tabelle 6.2: Leistungsvergleich der Gesamtzeiten in Sekunden des Scalify CloudServers ohne und mit GlusterFS.

Gesamtzeiten [s]	
CloudServer Standard	CloudServer GlusterFS
10.441	10.223
15.145	15.221
25.727	26.059
48.146	48.155
92.433	92.020

Da sich die Zeiten kaum unterscheiden und die Daten weiterhin nicht kompromittiert werden, wurde darauf verzichtet, alle Speicherdienste zusätzlich mit GlusterFS zu testen. Die Installation und das Einbinden des Laufwerks gestaltet sich bei allen Diensten gleich. Meist kann durch eine Konfiguration der Speicherort bestimmt oder ein Parameter kann beim Starten mitgegeben werden.

Mit den Ergebnissen dieser Untersuchung lohnt es sich umso mehr, ein redundantes verteiltes Dateisystem einzurichten, mit mehreren Servern an verschiedenen Standorten, um die Daten in der Cloud zusätzlich abzusichern.

7 Untersuchung der Leistungsfähigkeit

Bei der Untersuchung der Leistungsfähigkeit der einzelnen Dienste wurde das Hauptaugenmerk auf die Zeit und die Datenintegrität gelegt. Die Zeit, die ein Dienst benötigt, um bestimmte Aufgaben zu bewältigen, gibt Auskunft darüber, wie schnell dieser Daten verarbeiten und bereitstellen kann. Dies ist besonders wichtig, wenn diese Daten oft abgerufen werden und hochverfügbar abgelegt werden. Die Datenintegrität gibt Auskunft über den vollständigen und fehlerfreien Zustand der Daten.

Alle in Kapitel 5 installierten und konfigurierten S3-Speicherdienste werden mit Hilfe des Skripts `s3perf` getestet, welches am Ende eines Durchlaufs eine Gesamtzeit ausgibt, die es benötigt hat, alle S3-spezifischen Operationen durchzuführen. Je kleiner dieser Wert ist, desto leistungsfähiger, in Bezug auf Schnelligkeit, ist der entsprechende Dienst.

Beim Hoch- und Herunterladen der Daten wird eine MD5-Prüfsumme generiert, welche beim Herunterladen der Daten deren Echtheit bzw. Integrität überprüft. Damit kann sichergestellt werden, dass die Daten nicht kompromittiert wurden und denen entsprechen, welche zuvor auch hochgeladen wurden.

7.1 Messung der Leistung

Die Tests der einzelnen Dienste wurden von einem Lenovo Thinkpad E460 mit einem Core i3 und 8 GB RAM durchgeführt. Am zeitaufwändigsten war das Hochladen der Daten. Dies wurde mit DSL 50 durchgeführt, was 50 Mbps im Down- und 10 Mbps im Upstream besitzt.

Die Dienste wurden mit jeweils fünf 1, 2, 4, 8 und 16 MB großen Dateien auf der Infrastruktur der AWS und in VMs von VirtualBox getestet. Das heißt, zu Beginn wurden fünf Dateien mit zufälligem Inhalt und der angegebenen Größe erstellt und auf den Dienst hoch- und heruntergeladen. Die Zeiten der einzelnen Dateigrößen können übersichtlich in der Tabelle 7.1 in Kapitel 7.3 eingesehen werden. Die Einzelheiten der individuellen Operationen können im Anhang D detailliert erfasst werden.

Bei den öffentlichen Clouds von Amazon, Microsoft und Google bestand nicht die Möglichkeit, sie innerhalb von VirtualBox zu testen.

7.1.1 VirtualBox

Für VirtualBox wurde eine VM mit 1 vCPU und 1 GB RAM mit dem Ubuntu Server 16.04 erzeugt, die anschließend als Open Virtual Appliance (OVA) exportiert wurde. Dies hat den Vorteil, dass nicht für jede Cloud erst eine VM erzeugt, das Betriebssystem und Updates installiert werden müssen, sondern dies alles schon durchgeführt wurde, ähnlich eines AMI innerhalb der EC2. Wenn die OVA importiert wurde, kann direkt mit der Installation und Einrichtung der Cloud-Speicherdienste begonnen werden, was viel Zeit spart.

7.1.2 Amazon S3

Der S3 von Amazon kann ohne weitere Probleme mit dem Skript `s3perf` getestet werden.

7.1.3 Azure BS

Für Azure war es notwendig das Kommandozeilenprogramm `az` der Azure CLI zu implementieren, da es nicht möglich ist, durch `s3cmd` mit Azure zu kommunizieren. Weiterhin kann die Azure CLI nicht von Haus aus mehrere Dateien auf einmal aus einem Container löschen, wodurch der Befehl zum Löschen innerhalb einer `for`-Schleife ausgeführt werden muss [11]. Dies hat zur Folge, dass das Löschen insgesamt mehr Zeit als bei anderen Diensten in Anspruch nimmt.

Außerdem ist der Befehl nur bis Ubuntu 16.04 oder unter anderen Distributionen nutzbar, da ab einer aktuelleren Version Probleme mit Python auftreten und der Befehl nicht korrekt ausgeführt wird [11][86].

7.1.4 Google CS

Google CS sollte prinzipiell mit `s3cmd` kommunizieren. Jedoch wird beim Löschen des Buckets immer die Meldung

```
ERROR: S3 error: 403 (SignatureDoesNotMatch): The request signature we calculated does not match the signature you provided. Check your Google secret key and signing method.
```

ausgegeben, wodurch der Test erfolglos abbricht. Abhilfe schafft die Implementierung der Google API in Form des Kommandozeilenprogramms `gsutil`, was ohne Fehler den Test durchläuft.

7.1.5 Minio

Minio kann ohne weitere Probleme mit dem Skript `s3perf` getestet werden.

7.1.6 Cumulus

Cumulus konnte nicht im parallelen Modus getestet werden und hat den Zugriff mit der Meldung

```
ERROR: S3 error: 403 (AccessDenied): Access Denied
```

bei einer Dateigröße von 16 MB verweigert. Außerdem müssen die Buckets in Großbuchstaben sein, sodass die Option `-u` notwendig ist.

7.1.7 Swift

Swift nutzt die eigene Swift API und kann ohne weitere Probleme mit dem Skript `s3perf` getestet werden.

7.1.8 Riak CS

Riak CS kann nur bis zu 8 MB Daten verarbeiten, 16 MB konnten nicht hochgeladen werden. Generell ist Riak CS langsamer als andere Dienste, was den Upload angeht. Die Ursache konnte nicht ausfindig gemacht werden, da Riak CS in einer lokalen VM sehr performant ist.

7.1.9 S3 Ninja

Da S3 Ninja mit nginx läuft, muss der Wert der Größe von Daten, die hochgeladen werden, angepasst werden. Dazu muss in der `/etc/nginx/sites-available/default` zusätzlich `client_max_body_size 100M;` vor der letzten `}` hinzugefügt werden. Außerdem müssen die Buckets in Großbuchstaben sein, sodass die Option `-u` notwendig ist.

7.1.10 S3rver

S3rver kann keine 16 MB Dateien verarbeiten. Bei dem Versuch diese hochzuladen, erscheint die Meldung:

```
ERROR: S3 error: 404 (Not Found)
Unable to upload the files.
```

7.1.11 Scalify CloudServer

Scalify S3 kann ohne weitere Probleme mit dem Skript `s3perf` getestet werden.

7.2 Betrachtung der Datenintegrität

Während der Messung der Leistungsfähigkeit, also die Zeit die ein Dienst benötigt, um Daten zu verarbeiten, wurde auch die Datenintegrität betrachtet, welche einen noch höheren Stellenwert einnimmt als die Leistung selbst.

Die Datenintegrität trifft eine Aussage über den Erhalt der Daten, die über ein Netzwerkprotokoll ausgetauscht werden. Es gibt mehrere Möglichkeiten diese zu kompromittieren und zu beschädigen oder zu verändern.

Einerseits kann der Datenverkehr direkt abgegriffen und verändert werden. Daten können in wenigen Sekunden mit einem Schädling (Virus, Trojaner etc.) versehen werden und zurück in den Datenstream geladen werden. Der Nutzer bekommt davon nur wenig mit und sieht z.B. nur, dass die Übertragung etwas mehr Zeit in Anspruch nimmt, was auf den ersten Blick nicht verdächtig wirkt, da das Netz mal mehr und mal weniger ausgelastet sein kann.

Andererseits können die Daten am Bestimmungsort, also dem Server mit welchem die Daten ausgetauscht werden, korrumpiert werden. Diese können ebenfalls abgegriffen

und verändert wieder in den Speicher geladen werden, was nur schwer erkannt werden kann. [33][44]

Aus diesem Grund ist es sinnvoll, wie auch bei `s3perf`, einen Hashwert für die Daten zu erzeugen, der die Daten als jene identifiziert, die zuvor hochgeladen wurden. Sollten Daten korumpiert sein, enthalten diese einen anderen Hashwert und können nicht gegengeprüft werden, wodurch ersichtlich wird, dass die Daten fehlerhaft sind.

MD5 wird im Fall dieser Arbeit für die Überprüfung genutzt. MD5 ist zwar schon älter, aber um Daten mit einem Hashwert zu versehen, reicht es durchaus und ist gleichzeitig performant, durch die Berechnung mit 128-bit. Alternativ könnte SHA265 genutzt werden, was zwar moderner ist und neuere Standards nutzt, jedoch aufgrund der Berechnung mit 265-bit mehr Zeit in Anspruch nimmt [46].

In allen Tests stimmten die Hashwerte der heruntergeladenen Daten mit den hochgeladenen überein.

7.3 Ergebnisse

Die Ergebnisse aus der Leistungsmessung werden in Tabelle 7.1 strukturiert aufgelistet. Die Tabelle ist so aufgebaut, dass auf der linken Seite die einzelnen Cloud-Speicherdienste, deren Installation und Konfiguration in Kapitel 5 beschrieben wurde, aufgereiht sind.

Anschließend wird die Tabelle in zwei Teile unterteilt. Zum einen die Messungen der öffentlichen Clouds und der privaten Clouds auf der Infrastruktur der AWS. Zum anderen die Installation der privaten Clouds innerhalb von VirtualBox.

Diese Aufteilung wird ein weiteres Mal in die einzelnen Dateigrößen (in MB) unterteilt, sodass insgesamt bis zu 10 Zeilen je Cloud abgebildet werden.

Ein „x“ in einer Zelle der öffentlichen Clouds bedeutet, dass es nicht möglich war, den Dienst innerhalb von VirtualBox zu installieren und zu testen. Das liegt vor allem daran, dass die öffentlichen Clouds nicht auf eigener Hardware installiert werden können, da sie nur auf der Infrastruktur des Anbieters betrieben werden. Ein „x“ in einer Zelle eines privaten Cloud-Speicherdiensts bedeutet, dass bei dieser Dateigröße kein Messergebnis erzeugt werden konnte. Die Ursachen dafür wurden in Kapitel 7.1 erläutert.

Tabelle 7.1: Die Ergebnisse (Gesamtzeiten in Sekunden) aus den Tests mit dem Skript s3perf.

Cloud	Öffentliche Cloud / AWS					VirtualBox				
	Größe der Daten [MB]									
	1	2	4	8	16	1	2	4	8	16
Gesamtzeiten aller Aktionen [s]	Amazon S3	24.600	26.262	45.468	69.829	104.568	x	x	x	x
	Azure BS	37.356	44.155	57.342	80.072	122.108	x	x	x	x
	Google CS	33.189	40.562	53.495	74.356	126.014	x	x	x	x
	Minio	8.905	15.078	25.718	49.106	93.172	4.309	5.120	6.382	11.573
	Cumulus	8.249	13.808	24.857	47.720	x	2.647	3.335	4.252	6.326
	Swift	13.329	18.960	29.382	51.451	94.428	7.890	8.217	8.801	10.394
	Riak CS	13.567	23.977	42.482	79.864	x	5.007	5.880	5.751	7.442
	S3 Ninja	9.537	14.767	25.565	48.443	93.141	5.707	7.475	11.421	18.809
	S3rver	9.429	14.767	25.906	47.791	x	5.832	8.124	10.956	23.021
	Scality CS	10.441	15.145	25.727	48.146	92.433	6.760	13.525	12.270	20.500

Die einzelnen Werte in der Tabelle sind die Gesamtzeiten in Sekunden aus den einzelnen Messungen. Wie zu sehen ist, sind die Zeiten innerhalb der VirtualBox um ein vielfaches geringer. Was daran liegt, dass die VMs im selben Netzwerk wie das Skript, dass die Messungen durchgeführt hat, waren.

Alle öffentlichen Cloud-Speicherdienste waren langsamer als die privaten Dienste. Dabei muss jedoch davon ausgegangen werden, dass dies an der Auslastung der öffentlichen Dienste liegt. Dort kommen jede Sekunde tausende von Anfragen an, wodurch diese Dienste eine viel höhere Auslastung verbuchen müssen, als ein privater Dienst, welcher in der Regel nur wenige Anfragen bekommt.

Bis zum 8-MB-Test ist Cumulus der schnellste Dienst, was sich auch innerhalb der VirtualBox widerspiegelt. Danach war es nicht möglich, die 16 MB zu verarbeiten und Scality CS machte den schnellsten 16-MB-Test. Minio schaffte in VirtualBox den 16-MB-Test in einer sehr schnellen Zeit, was nur ca. ein Drittel der Zeiten von S3 Ninja und Scality CS ist.

Vergleichsweise hoch waren die Zeiten von Riak CS ab einer Dateigröße von 2 MB. Wie im Anhang D eingesehen werden kann, waren die Zeiten des Hochladens besonders langsam. Auch durch Tests an verschiedenen Tagen und zu unterschiedlichen Zeiten brachten keine Besserung. Der Dienst ist lediglich innerhalb der AWS langsam, da dieser in VirtualBox schneller agierte.

Insgesamt ermöglicht die übersichtliche Darstellung einen schnellen Überblick über die Gesamtzeiten der einzelnen Cloud-Speicherdienste.

7.3.1 Bewertung

Die Bewertung der Speicherdienste ist eine essentielle Aufgabe, um herauszufinden welcher das beste Gesamtpaket im Vergleich bietet. Hierzu wurden die fünf Kriterien Datenintegrität, Leistung, Installation / Einrichtung, Dokumentation und Weboberfläche festgelegt, die von einem S3-Speicherdienst erwartet werden. Die Kriterien wurden mit einer Gewichtung von 0 bis 10 versehen, wobei 0 unwichtig und 10 sehr wichtig ist (siehe Tabelle 7.2).

Die einzelnen Bewertungen bewegen sich in einem Bewertungsbereich von 0 bis 10 Punkte, wobei 0 sehr schlecht bzw. nicht vorhanden und 10 sehr gut ist. Der Nutzwert errechnet sich, indem die Bewertungen mit den jeweiligen Gewichtungen multipliziert werden und diese Summe durch die Summe der Gewichtung dividiert wird:

$$\text{Nutzwert} = \sum (\text{Bewertung} * \text{Gewichtung}) / \sum \text{Gewichtung}$$

Umso höher der resultierende Wert ist, desto besser ist das Gesamtpaket bzw. desto leistungsfähiger ist der Dienst.

Zusätzlich zur Matrix visualisiert ein Balkendiagramm (Abbildung 7.1) die Ergebnisse aus dieser, um einen besseren Eindruck über die Bewertungen zu bekommen. Die Balken sind gestapelt angeordnet und beinhalten alle Bewertungskriterien. Je länger ein Balken ist, desto besser ist dieser Dienst insgesamt. Durch die farbliche Unterteilung kann die Bewertung der einzelnen Kriterien visualisiert werden.

Datenintegrität

Der Punkt Datenintegrität ist der wichtigste, da jede Cloud nutzlos ist, wenn Daten korrumpiert werden. Von daher wurde diesem Punkt die höchste Gewichtung von 10 zugewiesen.

Leistung

Im Anschluss daran kann die Leistung der Cloud betrachtet werden. Da es ausschlaggebend für die Leistung ist, wie schnell ein Dienst agiert, hat dieses Kriterium eine Gewichtung von 8.

Installation / Einrichtung

Die Installation und Einrichtung von Diensten spielt eine essentielle Rolle, da der Dienst möglichst einfach und nachvollziehbar zu installieren sein sollte. Die öffentlichen Clouds können nicht installiert, sondern nur eingerichtet werden. Bei der Bewertung muss sowohl die Installation auf einer EC2-Instanz als auch innerhalb einer VM in VirtualBox betrachtet werden. Dieses Kriterium wurde mit 6 gewichtet.

Dokumentation

Die Dokumentation stellt bei jeder Art von Diensten die erste Anlaufstelle bei der Benutzung dar. Die Dokumentation gibt Aufschluss über die Installation und Konfiguration sowie Anleitungen für Entwickler, die einen Open-Source-Dienst weiterentwickeln möchten. Neben der Dokumentation existieren oftmals zahlreiche Anleitungen, wie ein Dienst betrieben werden kann. Dieses Kriterium wurde mit 4 gewichtet.

Weboberfläche

Durch eine Weboberfläche können die Speicherdienste unterstützend betrieben werden. Diese sind mehr oder weniger hilfreich, um sich einen Überblick über die aktuellen Daten zu machen oder sogar gänzlich über die Weboberfläche mit dem Dienst zu interagieren. Besonders bei den öffentlichen Clouds ist dies meist die erste Anlaufstelle für neue Kunden. Bei den privaten Clouds sind Weboberflächen eher selten oder, wenn vorhanden, vergleichsweise schlecht umgesetzt. Da für diese Arbeit eine Weboberfläche nur eine nebensächliche Rolle spielt, wurde hier mit 1 gewichtet.

7.3.2 Bewertungsmatrix

Tabelle 7.2: Bewertung der einzelnen Speicherdienste anhand von festgelegten Kriterien. Die Kriterien haben eine Gewichtung und die Dienste werden von 0 bis 10 Punkte bewertet. Der Nutzwert errechnet sich wie in Kapitel 7.3.1 angegeben.

Kriterium	Gewichtung	Amazon S3	Azure BS	Google CS	Minio	Cumulus	Swift	Riak CS	S3 Ninja	S3rver	Scality CS
Datenintegrität	10	10	10	10	10	10	10	10	10	10	10
Leistung	8	5	3	4	7	2	6	2	10	2	9
Installation/ Einrichtung	6	8	7	7	10	7	6	5	8	9	9
Dokumentation	4	7	7	6	9	6	9	8	6	7	9
Weboberfläche	1	7	5	9	10	0	0	0	8	0	0
Summe	29	223	199	207	262	182	220	178	260	198	262
Nutzwert		7.69	6.86	7.14	9.03	6.28	7.59	6.14	8.97	6.83	9.03
Rang		3	6	5	1	8	4	9	2	7	1

Beispielrechnung für Amazon S3

$$\text{Nutzwert} = (10 * 10 + 5 * 8 + 8 * 6 + 7 * 4 + 7 * 1) / 29 \approx 7.69$$

7.3.3 Visualisierung der Bewertungen

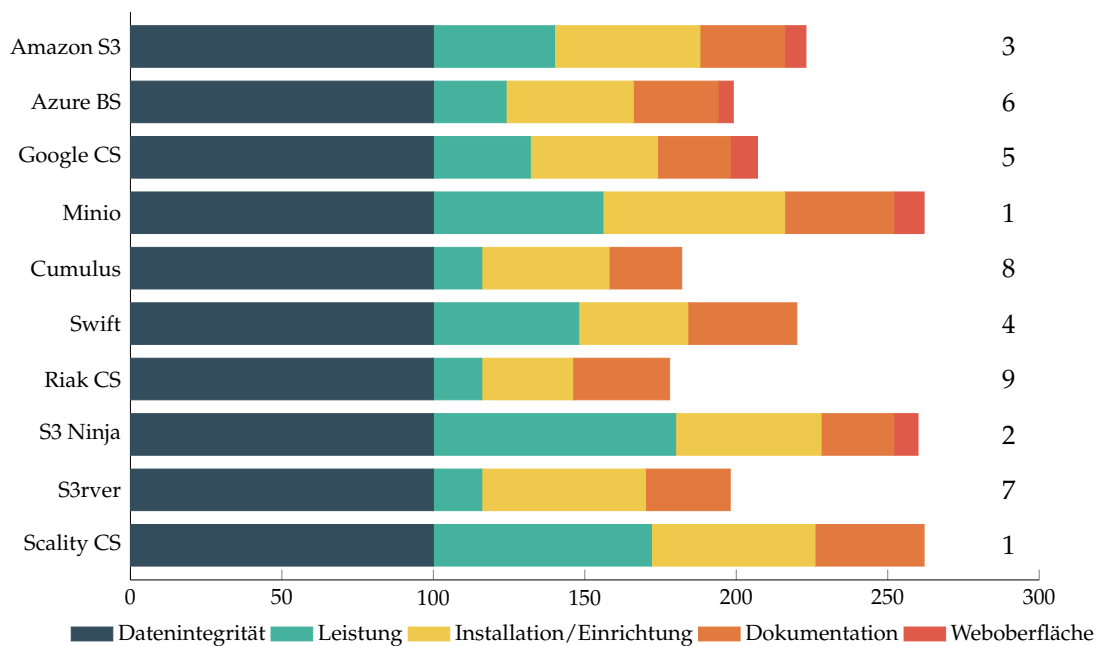


Abbildung 7.1: Gestapeltes Balkendiagramm analog zu den Ergebnissen der Bewertungsmatrix mit Rang am Ende des jeweiligen Balkens.

Das Balkendiagramm visualisiert anschaulich die Ergebnisse aus der Bewertungsmatrix. Gut zu erkennen ist, dass alle Dienste 10 Punkte bei der Datenintegrität erhalten haben. Erst danach unterscheiden sich die Dienste deutlich, vor allem bei der Leistung. Die folgenden Bewertungen bewegen sich in ähnlichen Bereichen. Wie zu sehen ist, hat das letzte Kriterium *Weboberfläche* kaum Auswirkungen auf die Gesamtwertung. Scalify CS und Minio sind auf Platz 1, wobei Scalify CS keine Weboberfläche besitzt.

An der rechten Seite lassen sich die Ränge der Cloud-Speicherdienste, wie sie in der Bewertungsmatrix ermittelt wurden, ablesen.

8 Barrierefreie Cloud-Speicherdienste

Der Studiengang *Barrierefreie Systeme* beschäftigt sich mit Themen der Barrierefreiheit und dessen Umsetzung. Der Schwerpunkt *Intelligente Systeme* betrachtet die informationstechnische Entwicklung und Gestaltung von barrierefreien Systemen. Durch die Zusammenarbeit von drei unterschiedlichen Fachdisziplinen (Architektur, Informatik und Pflege/Soziale Arbeit) ist dieser Studiengang zusätzlich interdisziplinär aufgebaut.

Besonders die Barrierefreiheit ist ein wichtiges Thema, welches konsequent umgesetzt werden muss, um alle Personengruppen zu erreichen und zu unterstützen. Vor allem Personen mit visuellen und mobilen Einschränkungen haben Schwierigkeiten, PC-Technik und deren Software sowie Dienste zu nutzen. Um dem entgegenzuwirken, muss PC-Technik barrierefrei und einfach handhabbar gemacht werden. Mit Speicherdiensten kann PC-Technik nicht nur vereinfacht und unterstützt werden, sondern auch so umgesetzt werden, dass diese barrierefrei genutzt werden kann.

8.1 Hochverfügbarkeit und Ausfallsicherheit

S3-Speicherdienste werden unter anderem mit dem Ziel der Hochverfügbarkeit und Ausfallsicherheit konstruiert. Dadurch können diese Dienste als Grundlage für kritische Anwendungen und Systeme genutzt werden, da nur eine sehr geringe Wahrscheinlichkeit besteht, dass diese Dienste nicht verfügbar sind. So können beispielsweise kontinuierlich Daten an einen Speicherdienst gesendet werden, welche in Echtzeit ausgewertet werden und auf welche bei Schwierigkeiten direkt reagiert werden kann. Bei medizinischen Instrumenten wie z.B. einer Insulinpumpe können die Werte ständig von einem Arzt (automatisiert) überwacht werden und dieser kann bei zu niedrigen Werten darauf reagieren und Kontakt mit dem Patienten aufnehmen.

Würde in einem solchen Szenario ein beliebiger Speicherdienst genutzt, kann es vorkommen, dass dieser nicht erreichbar ist, ohne dass die Parteien etwas davon mitbekommen. Durch die Ausfallsicherheit der S3-Speicherdienste können Vorhaben realisiert werden,

die Menschenleben betreffen und kurze Reaktionszeiten benötigen. Zu beachten ist dabei, dass keine öffentlichen Anbieter gewählt werden, sondern Lösungen auf privater Hardware umgesetzt werden. Im Falle von medizinischen Anwendungen handelt es sich um sehr sensible Daten, die nicht an Dritte weitergegeben werden dürfen und vertraulich sind.

Durch die vorangegangenen Tests kann ausgewählt werden, welcher Dienst am besten für ein solches Vorhaben geeignet ist. Dabei wäre besonderes Augenmerk auf die Leistung zu legen, da die Daten möglichst schnell vom Patienten zum Arzt gelangen müssen.

8.2 Implementierung von Cloud-Speicherdiensten

S3-Speicherdienste besitzen in den meisten Fällen eine API, womit diese in andere Anwendungen eingebunden oder umprogrammiert werden können. Damit ist es auf einfache Art und Weise möglich, einen Dienst an Personengruppen anzupassen, die Schwierigkeiten haben, einen PC zu bedienen.

Es existieren Möglichkeiten, wie z.B. ein Text-to-Speech-System (TTS)¹, Steuerung mit den Augen oder der Stimme, um einen PC und Software zu bedienen. Mit Hilfe der einfachen API und den verschiedenen unterstützenden Maßnahmen können Anwendungen geschaffen werden, die von allen Personengruppen genutzt werden können.

Einerseits kann eine grafische Oberfläche geschaffen werden, die leicht zu erkennen sowie abzulesen ist. Andererseits kann diese Oberfläche so konstruiert werden, dass ein TTS keine Probleme bei der Erkennung von Zeichen hat und den Nutzer fehlerfrei durch die Navigation geleiten kann.

Umgekehrt ist dies auch möglich, sodass per Sprache Befehle gegeben werden, die der PC anschließend umsetzt. Innerhalb der Anwendung können Befehle und Schlüsselwörter definiert werden, auf die ein Mikrofon reagiert und bei korrekter Erkennung eine Aktion ausführt. Der Nutzer könnte beispielsweise „Erstelle einen neuen Bucket“ sagen, woraufhin das Programm über die API den Speicherdienst anweist, einen neuen Bucket zu erstellen.

¹ Ein TTS ist eine Software, die Fließtext auf einem Bildschirm in akustische Sprache umwandelt und diese ausgibt.

Mit einer einfachen API können Anwendungen geschaffen werden, die viele körperliche Einschränkungen kompensieren und damit das Leben des Nutzers erleichtern.

Durch die Hochverfügbarkeit und Ausfallsicherheit können S3-Speicherdienste für kritische Anwendungen genutzt und optimiert werden.

9 Fazit und Ausblick

Die Themen Cloud Computing und Internet sind allgegenwärtig und aktueller denn je. Immer mehr Anbieter lagern ihre Plattformen und Anwendungen in die Cloud aus. Ganze Infrastrukturen werden nicht mehr Inhouse sondern extern und in VMs betrieben.

Über eine Cloud ist es möglich, Daten überall verfügbar zu machen. Dafür sind Dienste notwendig, die diese Daten immer bereitstellen können und gegen Ausfälle gesichert sind. Wer nicht bei den großen drei Diensten Amazon, Microsoft und Google Dienste einkaufen möchte, muss einen Speicherdienst auf eigener Hardware betreiben. Die Daten liegen im privaten Umfeld und Außenstehende haben in der Regel keinen Zugriff auf diese, jedoch muss für den reibungslosen Betrieb Sorge getragen werden. Besonders im Hinblick auf die „Datensammelwut“ vieler Anbieter, kann es nur vorteilhaft sein, sämtliche (Open-Source-)Dienste auf eigener Hardware zu betreiben und zur Verfügung zu stellen. Denn nur so kann gewährleistet werden, dass die Daten nicht unberechtigt an Dritte weitergegeben und im schlimmsten Fall vermarktet werden.

Um der „Datensammelwut“ und der Weitergabe von Daten entgegenzuwirken, kann es hilfreich sein, eine Übersicht über aktuelle Speicherdienste, welche die Schnittstelle des S3 nutzen, zu haben. Neben den drei öffentlichen Clouds wurden neun private Open-Source-Clouds installiert und analysiert. Die erste Hürde dieser Arbeit war das Zusammentragen passender Speicherdienste und deren Auswertung. Viele Skripte und Anwendungen boten nicht das passende Gesamtpaket für eine Auswertung und die Benutzung mehrerer Skripte kam nicht in Frage, da dies die Vergleichbarkeit beeinträchtigt hätte. Letztendlich erschien das Skript `s3perf` am geeignetsten, da es viele S3-Speicherdienste aufzählt, die auf eigener Hardware installiert werden können. Insgesamt wurden 12 Speicherdienste zusammengetragen, 9 davon sind private Clouds.

Die meiste Zeit wurde in die Installation und Einrichtung der einzelnen Dienste investiert. Die öffentlichen Clouds mussten lediglich eingerichtet werden und liefen danach ohne Probleme. Die privaten Speicherdienste mussten auf passender (virtueller) Hardware installiert werden. Eine der Vorgaben dieser Arbeit ist die Nutzung der AWS und die

Installation der Dienste innerhalb dieser. Anfangs schien es, als sei diese Aufgabe gut zu bewältigen, bei der Installation von Eucalyptus Walrus stellten sich allerdings erste Probleme heraus. Dieser Dienst schied gleich zu Anfang aus der Auswahl aus, da es nicht möglich war, diesen innerhalb der AWS zu installieren. Das Problem war hierbei die Notwendigkeit einer verschachtelten Virtualisierung, was in einer VM, zumindest bei Amazon, nicht möglich ist.

Aber auch bei den restlichen Speicherdiensten war eine reibungslose Installation nur bedingt möglich. Der Dienst Fake S3 ließ sich zwar installieren, quittierte jedoch später die Arbeit beim Löschen eines Buckets. Minio und Nimbus Cumulus konnten ohne größere Schwierigkeiten installiert werden. Openstack Swift und Riak CS stellten eine weitere Herausforderung dar, da diese Dienste nicht manuell installiert werden konnten bzw. nach der Installation nicht erreichbar waren. Swift wurde mit Vagrant installiert, was eine Automatisierung darstellt und gleichzeitig interessant zu nutzen war, da immer mehr Dienste automatisiert werden. Ebenso wurde Riak CS mit Hilfe von Docker installiert, was auch vermehrt genutzt wird. Mit den beiden genannten Diensten wurden damit zwei aktuelle Techniken in der Arbeit verwendet, die generell immer mehr genutzt werden. Die letzten drei Speicherdienste S3 Ninja, S3rver und Scalify CloudServer ließen sich ohne größere Komplikationen installieren und betreiben.

Neben der Installation in den AWS wurden alle privaten Cloud-Speicherdienste in einer VM unter VirtualBox installiert. Dies hat den Vorteil, dass die Leistung auch innerhalb eines privaten Netzwerks gemessen werden kann.

Zusätzlich zu der hohen Ausfallsicherheit sollte untersucht werden, wie sich ein verteiltes Dateisystem auf die Speicherdienste auswirkt. Nach umfassender Recherche in das Thema stellte sich heraus, dass sich GlusterFS am besten für das Vorhaben eignet, da es ein Dateisystem ist, welches leicht zu installieren und zu nutzen ist. Dafür muss nur ein Server, worauf der GlusterFS-Server läuft, erzeugt und anschließend Clients zu diesem verbunden werden. Nach der Installation des GlusterFS-Clients unter Minio und Scalify CS stand schnell fest, dass ein verteiltes Dateisystem kaum einen Einfluss auf die Leistung eines S3-Speicherdienstes hat. Durch diese Ergebnisse wurde nicht jeder Dienst mit GlusterFS getestet, da davon auszugehen ist, dass sich alle Dienste ähnlich verhalten werden. Die Einbindung eines verteilten Dateisystems ist für den Speicherdienst auch nur ein Ordner auf dem System, der im Hintergrund synchronisiert wird. Aus diesem Grund ist dieses Verhalten nur logisch und besonders deswegen sollte immer ein verteiltes Dateisystem für Cloud-Speicherdienste genutzt werden, um zusätzliche Redundanz und Sicherheit zu besitzen.

Bei der anschließenden Untersuchung der Leistungsfähigkeit wurden alle Dienste der

Reihe nach mit `s3perf` getestet und bewertet. Dabei wurde nochmals auf alle Dienste eingegangen und es wurde aufgezeigt, wo Probleme entstanden sind und welche Dienste problemlos getestet werden konnten. Hierbei fiel auf, dass sich der GCS nicht mit `s3cmd` testen ließ, egal wie es eingestellt wurde. Aus diesem Grund musste das Werkzeug `gsutil` installiert und in das Skript eingepflegt werden. Damit war es problemlos möglich, Google zu testen. Um auch Microsoft mit in den Test einzubeziehen, musste die Azure CLI `az` zusätzlich in das Skript eingebunden werden. Eine größere Hürde stellte das Löschen des Containerinhalts dar, da bisher keine Funktion vorgesehen ist, die mehrere Blobs in einem Durchlauf löscht. Aus diesem Grund wurde eine `for`-Schleife geschrieben, die für alle Dateien innerhalb des Containers den Löschbefehl ausführt, um so den Container zu leeren. Zusätzlich lässt sich die Schleife nicht auf jedem beliebigen Betriebssystem ausführen. Erst durch lange Recherchen und Fragen auf *Stack Overflow* kam heraus, dass Python unter Ubuntu 17.04 nicht mit Azure kompatibel ist. Auffällig waren auch die drei Speicherdienste, die nicht in der Lage waren, die 16MB-Daten zu verarbeiten. Leider konnte nicht festgestellt werden, aus welchem Grund dieser Test fehlschlug.

Die Datenintegrität ist einer der wichtigsten Punkte bei einem Cloud-Speicherdienst. Denn einem Dienst nützt auch die beste Leistung nichts, wenn dieser kompromittiert werden kann. Erfreulicherweise konnte kein Dienst kompromittierte Daten vorweisen. Getestet wurde die Datenintegrität mit Hilfe einer MD5-Prüfsumme, die aus den Daten generiert wurde. Beim Herunterladen der Daten hätte sich, bei einer Veränderung, dieser Wert geändert.

Die Aktionen, welche die einzelnen Dienste durchgeführt haben, wurden in Sekunden gemessen. Um diese ganzen Zeiten übersichtlich darzustellen, wurde eine Tabelle mit allen Werten erstellt. Daraus konnte relativ schnell abgeleitet werden, dass die öffentlichen Cloud-Anbieter nicht die beste Performanz bieten und im Vergleich zu den privaten Cloud-Speicherdiensten langsamer sind. Dies kann natürlich daran liegen, dass die Dienste wesentlich frequenter sind, als jene, die nur wenige Nutzer bewältigen müssen. Dafür sorgen die Anbieter für einen reibungslosen Betrieb und für die Verfügbarkeit ihrer Dienste.

Die anschließende Bewertung der Dienste war mitunter das Spannendste an der gesamten Arbeit, was vor allem daran liegt, dass die Bewertung das endgültige Ergebnis der Arbeit darstellt. Das Verfassen der Bewertungskriterien und die anschließende Einschätzung der Dienste nahm wieder etwas mehr Zeit in Anspruch. Dabei wurde letztlich auf alle Schwerpunkte eingegangen, die während der gesamten Arbeit sehr präsent waren. Den fünf Kriterien musste nun eine Gewichtung zugeteilt werden, wodurch festgelegt wurde, wie wichtig die einzelnen Kriterien sind.

Von 0 bis 10 sollten die Dienste bewertet werden, wobei 0 „nicht vorhanden“ und 10 „sehr gut“ bedeutet. Die Bewertung wurde objektiv durchgeführt, ohne eine eigene Präferenz für einen Speicherdienst einfließen zu lassen. Dies gelang gut, indem z.B. die Leistung von der höchsten Punktzahl bis zur niedrigsten absteigend bewertet wurde. Die anderen Kriterien spiegeln den Aufwand für die Installation wieder. Umso besser die Dokumentation war, desto besser konnte auch die Installation durchgeführt werden bzw. der Dienst wurde besser erläutert. Die Weboberfläche wurde, wenn eine vorhanden war, bewertet und wie intuitiv diese bedient werden konnte. Die sich daraus ergebenden Werte konnten in eine Rangfolge umgewandelt werden, sodass feststand, welche Dienste, in dieser Arbeit, die beste und die schlechteste Gesamtleistung boten. Minio und Scality CS konnten mit einer sehr guten Gesamtleistung mit 262 von 290 Punkten als beste Dienste aus der Untersuchung hervorgehen. Am schlechtesten schnitt Riak CS mit 178 Punkten ab.

Durch ein gestapeltes Balkendiagramm wurden die Ergebnisse visualisiert. So konnte farblich dargestellt werden, wie die einzelnen Dienste in den Kategorien aufgestellt sind und wie die Gesamtverteilung aussieht.

Diese Masterthesis wurde im Studiengang *Barrierefreie Systeme* an der FRA-UAS verfasst, weswegen ein weiterer wichtiger Punkt der Zusammenhang des Themas dieser Arbeit mit der Barrierefreiheit darstellt. Gut vorstellbar ist es, dass sich ein Cloud-Speicherdienst, vor allem durch die Ausfallsicherheit und Hochverfügbarkeit, für medizinische Daten eignet. So kann z.B. ein medizinisches Gerät, das ein Patient benutzen muss, Daten an die Cloud senden und ein Arzt kann diese Werte ständig überwachen und im Notfall eingreifen. Wobei dann hier streng private Dienste genutzt werden müssen, damit die sensiblen Daten nicht an Unberechtigte gelangen.

Durch die einfache API, die von den meisten Diensten und der S3-Schnittstelle geboten wird, können barrierefreie Oberflächen geschaffen werden, die von Menschen mit Einschränkungen, wie z.B. Einschränkungen des Bewegungsapparats oder visuelle Einschränkungen, genutzt werden können. Weiterhin können die Dienste so einfach gestaltet werden, dass sie verständlich und nutzbar sind.

Insgesamt kann gesagt werden, dass das Thema „Untersuchung der Leistungsfähigkeit unterschiedlicher Cloud-Speicherdienste mit der Schnittstelle des Simple Storage Service (S3)“ in vollem Umfang bearbeitet werden konnte. Aktuelle Dienste wurden getestet, bewertet und daraus gingen schlussendlich zwei „Gewinner“ hervor. Durch die Installation und Arbeit mit den AWS wurden viele neue Erfahrungen gesammelt, die anschließend auch wieder zur Problembewältigung beitrugen. Cloud Computing ist ein faszinierender Bereich, der immer weiter wächst und in Zukunft einen noch höheren

Stellenwert einnehmen wird. Das liegt vor allem an der Kosteneinsparung und der Hochverfügbarkeit von Daten. Nicht umsonst ist es eines der wichtigsten Themen für die Digitalwirtschaft im Jahr 2017.

9.1 Ausblick

Für zukünftige Tests wäre es sinnvoll physische Server zur Verfügung zu haben, wodurch sich mehr Dienste installieren und testen lassen. Durch die Einschränkung auf VMs gehen viel Leistung und viele Möglichkeiten verloren. In dieser Arbeit war der Betrieb von physischen Servern auf Grund der Kosten nicht möglich. Als Forschungsprojekt mit ausreichend finanziellen Mitteln wäre es möglich, noch mehr Dienste in den Testpool aufzunehmen.

Denkbar wäre ein Dienst mit dem allgemein Cloud-Dienste getestet und bewertet werden können. Dadurch würde sich ein gesunder Wettbewerb entwickeln und es gäbe nicht nur die drei öffentlichen Clouds, sondern viele weitere Anbieter.

Interessant wäre auch, wie sich die Leistungsfähigkeit mit einer leistungstärkeren Instanz innerhalb der AWS verhält.

Das Skript könnte so weiterentwickelt werden, dass es tatsächlich barrierefrei bedienbar wird und damit viele weitere Personengruppen anspricht.

Literaturverzeichnis

- [1] ADOBE: *Adobe Creative Cloud*. 2017. – <http://www.adobe.com/de/creativecloud.html>, abgerufen am 12. Juni 2017
- [2] AMAZON: *Amazon EC2*. 2017. – <https://aws.amazon.com/de/ec2/>, abgerufen am 18. Juni 2017
- [3] AMAZON: *Amazon EC2-Instance-Typen*. 2017. – <https://aws.amazon.com/de/ec2/instance-types/>, abgerufen am 20. Juni 2017
- [4] AMAZON: *Amazon Elastic Block Store*. 2017. – <https://aws.amazon.com/de/ebs/>, abgerufen am 20. Juni 2017
- [5] AMAZON: *Amazon S3*. 2017. – <https://aws.amazon.com/de/s3/>, abgerufen am 12. Juni 2017
- [6] AMAZON: *Amazon Virtual Private Cloud VPC*. 2017. – <https://aws.amazon.com/de/vpc/>, abgerufen am 27. Juni 2017
- [7] AMAZON: *Arten von Cloud Computing*. 2017. – <https://aws.amazon.com/de/types-of-cloud-computing/>, abgerufen am 12. Juni 2017
- [8] AMAZON: *Datenmigration in die Cloud*. 2017. – <https://aws.amazon.com/de/cloud-data-migration/#optimized>, abgerufen am 25. Juni 2017
- [9] AMAZON: *Kostenloses Kontingent für AWS*. 2017. – <https://aws.amazon.com/de/free/>, abgerufen am 13. Juli 2017
- [10] AMAZON: *Regions and Availability Zones*. 2017. – <http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.html>, abgerufen am 20. Juni 2017
- [11] ARUKA ; JASON YE: *Azure-CLI Blob Storage remove multiple blobs*. 2017. – <https://stackoverflow.com/questions/45424688/azure-cli-blob-storage-remove-multiple-blobs>, abgerufen am 12. August 2017

- [12] BARDY, C.: *Amazon Web Services S3 wird zum Standardzugang für Objekt Storage*. 2017. – <http://www.searchstorage.de/lernprogramm/Amazon-Web-Services-S3-wird-zum-Standardzugang-fuer-Objekt-Storage>, abgerufen am 06. August 2017
- [13] BARRY & ASSOCIATES, INC.: *Service-Oriented Architecture (SOA) Definition*. 2017. – http://www.service-architecture.com/articles/web-services/service-oriented_architecture_soa_definition.html, abgerufen am 17. Juni 2017
- [14] BASHO: *Building a Local Test Environment*. 2017. – <http://docs.basho.com/riak/cs/2.1.1/tutorials/fast-track/local-testing-environment/>, abgerufen am 22. Juli 2017
- [15] BASHO: *Riak Cloud Storage*. 2017. – <http://docs.basho.com/riak/cs/2.1.1/>, abgerufen am 22. Juli 2017
- [16] BASHO: *Testing the Riak CS Installation*. 2017. – <http://docs.basho.com/riak/cs/2.1.1/tutorials/fast-track/test-installation/>, abgerufen am 22. Juli 2017
- [17] BAUN, C.: *Untersuchung und Entwicklung von Cloud Computing-Diensten als Grundlage zur Schaffung eines Marktplatzes*, Frankfurt University of Applied Sciences, Diss., 2011
- [18] BAUN, C.: *Cloud Computing – Eine kompakte Einführung*. 2012. – http://baun-vorlesungen.appspot.com/Netzwerke12/Skript/Was_ist_Cloud_Computing_auf_2_Seiten_v1.1.pdf, abgerufen am 18. Juni 2017
- [19] BAUN, C.: *Computernetze kompakt*. Berlin, Heidelberg : Springer, 2015. – ISBN 978-3-662-46931-6
- [20] BAUN, C.: *s3perf*. 2017. – <https://github.com/christianbaun/s3perf>, abgerufen am 12. Juni 2017
- [21] BAUN, C. ; KUNZE, M. ; NIMIS, J. ; TAI, S.: *Cloud Computing - Web-basierte dynamische IT-Services*. Berlin, Heidelberg : Springer, 2011. – ISBN 978-3-642-18435-2
- [22] BITKOM: *IT-Sicherheit, Cloud Computing und Internet of Things sind Top-Themen des Jahres in der Digitalwirtschaft*. 2017. – <https://www.bitkom.org/Presse/Presseinformation/IT-Sicherheit-Cloud-Computing-und-Internet-of-Things-sind-Top-Themen-des-Jahres-in-der-Digitalwirtschaft.html>, abgerufen am 12. Juni 2017

- [23] BJORNSON, Z.: *AWS S3 vs Google Cloud vs Azure: Cloud Storage Performance*. 2015.
– <http://blog.zachbjornson.com/2015/12/29/cloud-storage-performance.html>, abgerufen am 18. Juni 2017
- [24] BÖCK, H.: *Sicherheitsprodukte gefährden HTTPS*. 2017. – <https://www.golem.de/news/https-interception-sicherheitsprodukte-gefaehrden-https-1702-126098.html>, abgerufen am 06. August 2017
- [25] CHEF: *chef-solo*. 2017. – https://docs.chef.io/chef_solo.html, abgerufen am 22. Juli 2017
- [26] CHRISTIANBAUN: *Nimbus Cumulus on an Odroid U3 with Ubuntu 16.04 LTS*. 2017.
– <https://github.com/christianbaun/s3perf/wiki/Nimbus-Cumulus-on-an-Odroid-U3-with-Ubuntu-16.04-LTS#install-nimbus-cumulus>, abgerufen am 20. Juni 2017
- [27] CHRISTIANBAUN: *Unable to deploy the S3 server as docker container on a Raspberry Pi 3 #701*. 2017. – <https://github.com/scality/S3/issues/701>, abgerufen am 29. Juli 2017
- [28] CISCO: *Mehr Auswahl, größere Flexibilität: Lösungen für Private- und Hybrid-Cloud*. 2015. – http://www.cisco.com/c/de_de/solutions/cloud/private-hybrid-solutions.html, abgerufen am 18. Juni 2017
- [29] CLOUDYN: *AWS & Azure & GCP: Determining Your Optimal Mix of Clouds*. 2016.
– <https://www.cloudyn.com/wp-content/uploads/2016/05/AWS-GCP-Azure-May-2016-final.pdf>, abgerufen am 13. Juli 2017
- [30] CREASY, R.: *The Origin of the VM/370 Time-sharing System*. In: *IBM Journal of Research & Development* 25 (1981)
- [31] CRISP RESEARCH: *Der Stellenwert der Community Cloud wird sich erhöhen*. 2017.
– <https://www.crisp-research.com/der-stellenwert-der-community-cloud-wird-sich-erhoehen/>, abgerufen am 18. Juni 2017
- [32] DANIEL R.: *S3Cmd doesn't work with S3 Ninja*. 2016. – <https://stackoverflow.com/questions/38434905/s3cmd-doesnt-work-with-s3-ninja>, abgerufen am 26. Juli 2017
- [33] DFN-CERT: *Suche nach Hinweisen auf Kompromittierung am Beispiel von UNIX / Linux*. 2009. – <https://web.archive.org/web/20100513023734/http://www.dfn-cert.de/informationen/themen/incident-response-informationen/nachsehen-linux.html>, abgerufen am 05. September 2017

- [34] DOCKER INC.: *Get Docker CE for Ubuntu*. 2017. – <https://docs.docker.com/engine/installation/linux/docker-ce/ubuntu/>, abgerufen am 25. Juli 2017
- [35] DOCKER INC.: *What is Docker*. 2017. – <https://www.docker.com/what-docker>, abgerufen am 25. Juli 2017
- [36] DROMS, R.: *Dynamic Host Configuration Protocol*. 1997. – <https://tools.ietf.org/html/rfc2131>, abgerufen am 11. Juli 2017
- [37] ERIKHUDA: *thor*. 2017. – <https://github.com/erikhuda/thor>, abgerufen am 19. Juli 2017
- [38] EUCALYPTUS: *Eucalyptus Wiki Github*. 2017. – <https://github.com/eucalyptus/eucalyptus/wiki>, abgerufen am 17. Juli 2017
- [39] GARFINKEL, S. L.: An Evaluation of Amazon's Grid Computing Services: EC2, S3, and SQS. (2007). <http://nrs.harvard.edu/urn-3:HUL.InstRepos:24829568>
- [40] GLUSTER COMMUNITY: *Storage for your Cloud*. 2017. – <https://www.gluster.org/>, abgerufen am 30. Juli 2017
- [41] GOOGLE: *G Suite*. 2017. – <https://gsuite.google.com/intl/de/>, abgerufen am 12. Juni 2017
- [42] GOOGLE: *Products & Services*. 2017. – <https://cloud.google.com/products/>, abgerufen am 11. Juli 2017
- [43] HEISE DEVELOPER: *Zehn Programmiersprachen, die Entwickler 2017 lernen sollten*. 2017. – <https://www.heise.de/developer/artikel/Zehn-Programmier-sprachen-die-Entwickler-2017-lernen-sollten-3636059.html?seite=2>, abgerufen am 11. Juli 2017
- [44] HERTZOG, R. ; MAS, R.: *Umgang mit einem kompromittierten Rechner*. 2015. – <https://debian-handbook.info/browse/de-DE/stable/sect.dealing-with-compromised-machine.html>, abgerufen am 05. September 2017
- [45] IANBYTCHEK: *Docker Riak CS*. 2017. – <https://github.com/ianbytchek/docker-riak-cs>, abgerufen am 25. Juli 2017
- [46] INFORMATION TECHNOLOGY LABORATORY: *Secure Hash Standard (SHS)*. 2012. – <http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf>, abgerufen am 05. September 2017
- [47] JAMHALL: *S3rver*. 2017. – <https://github.com/jamhall/s3rver>, abgerufen am 28. Juli 2017

- [48] JENSHADLICH: *S3 Performance Test Tool*. 2016. – <https://github.com/jenshadlich/S3-Performance-Test>, abgerufen am 14. August 2017
- [49] JIMWEIRICH: *builder*. 2013. – <https://github.com/jimweirich/builder>, abgerufen am 19. Juli 2017
- [50] JUBOS: *Fake S3*. 2017. – <https://github.com/jubos/fake-s3>, abgerufen am 19. Juli 2017
- [51] JUBOS: *Fake S3 Supported Clients*. 2017. – <https://github.com/jubos/fake-s3/wiki/Supported-Clients>, abgerufen am 19. Juli 2017
- [52] KOLYSHKIN, K.: *Virtualization in Linux*. (2006). – <https://download.openvz.org/doc/openvz-intro.pdf>
- [53] KRESALEK, T.: *Virtualisierung von Betriebssystemen*. 2007. – <http://www.fh-wedel.de/~si/seminare/ws06/Ausarbeitung/02.VMware/vmware2.htm>, abgerufen am 06. August 2017
- [54] KUBICA, T.: *Proseminar - Everything as a Service: Aktueller Stand und Bewertung*. (2013). – http://www.rn.inf.tu-dresden.de/hara/arbeiten/SCC_PS_SS2013_Kubica_-_Everything_as_a_Service.pdf
- [55] LAND, L.: *Real-world benchmarking of cloud storage providers: Amazon S3, Google Cloud Storage, and Azure Blob Storage*. 2015. – <https://lg.io/2015/10/25/real-world-benchmarking-of-s3-azure-google-cloud-storage.html>, abgerufen am 18. Juni 2017
- [56] MACHADO, G. S. ; BOCEK, T. ; AMMANN, M. ; STILLER, B.: *A Cloud Storage overlay to aggregate heterogeneous Cloud services*. In: *38th Annual IEEE Conference on Local Computer Networks*, 2013, S. 597–605
- [57] MELL, P. ; GRANCE, T.: *The NIST Definition of Cloud Computing*. Special Publication, 2011. – <http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf>, abgerufen am 13. Juni 2017
- [58] MICROSOFT: *Azure-Produkte*. 2017. – <https://azure.microsoft.com/de-de/services/>, abgerufen am 11. Juli 2017
- [59] MICROSOFT: *Hybrid-Cloud*. 2017. – <https://www.microsoft.com/de-de/cloud-platform/hybrid-cloud>, abgerufen am 18. Juni 2017
- [60] MICROSOFT: *Was ist Office 365?* 2017. – <https://products.office.com/de-de/business/explore-office-365-for-business>, abgerufen am 12. Juni 2017

- [61] MINIO: *Minio Quickstart Guide*. 2017. – <https://github.com/minio/minio>, abgerufen am 20. Juli 2017
- [62] MINIO: *S3cmd with Minio Server*. 2017. – <https://docs.minio.io/docs/s3cmd-with-minio>, abgerufen am 20. Juli 2017
- [63] MOZILLA DEVELOPER NETWORK: *HTTP request methods*. 2017. – <https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods>, abgerufen am 18. Juni 2017
- [64] NODESOURCE: *NodeSource Node.js Binary Distributions*. 2017. – <https://github.com/nodesource/distributions>, abgerufen am 29. Juli 2017
- [65] OPEN SOURCE INITIATIVE: *What is "Open Source" software?* 2017. – <https://opensource.org/faq#osd>, abgerufen am 12. Juni 2017
- [66] OPENSTACK: *SAIO - Swift All In One*. 2017. – https://docs.openstack.org/swift/latest/development_saio.html, abgerufen am 21. Juli 2017
- [67] OPENSTACK: *Swift*. 2017. – <https://wiki.openstack.org/wiki/Swift>, abgerufen am 21. Juli 2017
- [68] PALANKAR, M. R. ; IAMNITCHI, A. ; RIPEANU, M. ; GARFINKEL, S.: Amazon S3 for Science Grids: A Viable Solution? In: *Proceedings of the 2008 International Workshop on Data-aware Distributed Computing*. New York, NY, USA : ACM, 2008 (DADC '08). – ISBN 978-1-60558-154-5, 55–64
- [69] PORTAL.DE ruby: *RubyGems*. 2017. – <http://wiki.ruby-portal.de/RubyGems>, abgerufen am 19. Juli 2017
- [70] QEMU: *What is QEMU?* 2017. – <http://www.qemu.org/index.html>, abgerufen am 12. Juni 2017
- [71] RAVELLO SYSTEMS: *HVX: High performance nested virtualization*. 2017. – <https://www.ravelliosystems.com/technology/nested-virtualization>, abgerufen am 12. Juni 2017
- [72] RIVEST, R.: *The MD5 Message-Digest Algorithm*. 1992. – <https://tools.ietf.org/html/rfc1321>, abgerufen am 17. August 2017
- [73] RUBY: *Ruby installieren*. 2017. – <https://www.ruby-lang.org/de/documentation/installation/#package-management-systems>, abgerufen am 19. Juli 2017
- [74] SCALITY: *S3*. 2017. – <https://github.com/scality/S3>, abgerufen am 29. Juli 2017

- [75] SCHAD, O.: *Kompromittierung*. 2017. – <http://www.oschad.de/wiki/Kompromittierung>, abgerufen am 24. Juni 2017
- [76] SCIREUM: *S3 ninja*. 2016. – <https://s3ninja.net/>, abgerufen am 26. Juli 2017
- [77] SEARCHDATACENTER: *kernel*. 2017. – <http://searchdatacenter.techtarget.com/definition/kernel>, abgerufen am 21. August 2017
- [78] SEBBL2GO: *Architektur Cloud Computing*. 2009. – https://commons.wikimedia.org/wiki/File:Architektur_Cloud_Computing.svg, abgerufen am 12. Juni 2017
- [79] SOUVIKDUTTACHOUDHURY: *Unable to Delete Buckets*. 2016. – <https://github.com/jubos/fake-s3/issues/172>, abgerufen am 20. Juli 2017
- [80] SWIFTSTACK: *vagrant-swift-all-in-one*. 2017. – <https://github.com/swiftstack/vagrant-swift-all-in-one>, abgerufen am 21. Juli 2017
- [81] UBUNTUUSERS: *ln*. 2017. – <https://wiki.ubuntuusers.de/ln/>, abgerufen am 22. August 2017
- [82] UBUNTUUSERS: *Offene Ports*. 2017. – https://wiki.ubuntuusers.de/Offene_Ports/, abgerufen am 20. Juni 2017
- [83] UBUNTUUSERS: *Virtualisierung*. 2017. – <https://wiki.ubuntuusers.de/Virtualisierung/>, abgerufen am 12. Juni 2017
- [84] VIRTUALBOX: *About VirtualBox*. 2017. – <https://www.virtualbox.org/manual/ch01.html>, abgerufen am 12. Juni 2017
- [85] W3C: *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)*. 2007. – <https://www.w3.org/TR/soap12/#intro>, abgerufen am 18. Juni 2017
- [86] WARSAW, B. A.: *ModuleNotFoundError when using literal string interpolation with invalid format specifier*. 2017. – <https://bugs.python.org/issue29307#msg285730>, abgerufen am 12. August 2017
- [87] WIKIPEDIA: *Everything as a Service*. 2017. – https://de.wikipedia.org/wiki/Everything_as_a_Service#Weitere_Ans.C3.A4tze, abgerufen am 21. Juni 2017
- [88] WIKTIONARY: *Serialisierung*. 2017. – <https://de.wiktionary.org/wiki/Serialisierung>, abgerufen am 02. September 2017

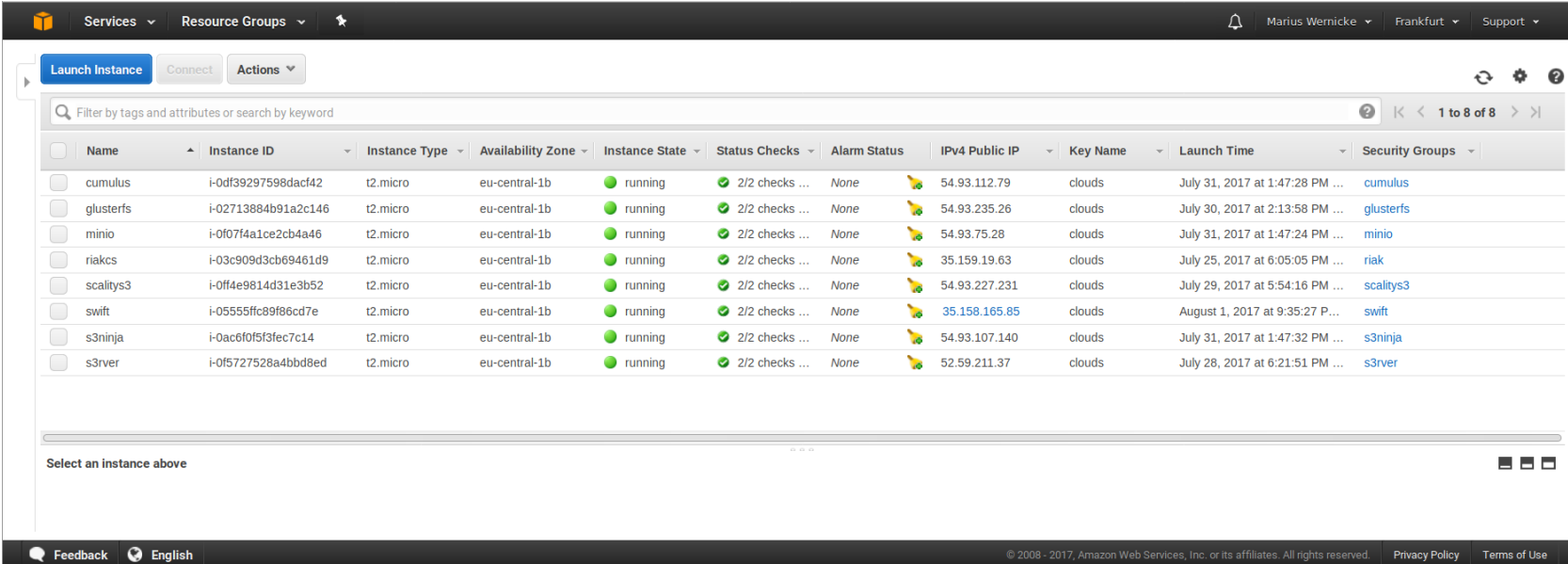
A Skript s3perf

Auf Grund der Tatsache, dass dieses Skript viele Zeilen enthält aber nur wenige geändert wurden, wurde dieses nicht als Anhang, sondern als Link auf Github eingebunden.

<https://github.com/mwrnck/s3perf/blob/master/s3perf.sh>

Dort können das Skript und die Änderungen eingesehen werden. Es wurde um den Azure Blob Storage und dessen API (Azure-CLI) sowie um die API (gsutil) des Google Cloud Storage erweitert. Außerdem wurden aktuelle Änderungen, welche mit der Version 2.0.0 von s3cmd kamen, in das Skript eingepflegt.

B AWS Instanzenübersicht



	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	IPv4 Public IP	Key Name	Launch Time	Security Groups
<input type="checkbox"/>	cumulus	i-0df39297598dacf42	t2.micro	eu-central-1b	running	2/2 checks ...	None	54.93.112.79	clouds	July 31, 2017 at 1:47:28 PM ...	cumulus
<input type="checkbox"/>	glusterfs	i-02713884b91a2c146	t2.micro	eu-central-1b	running	2/2 checks ...	None	54.93.235.26	clouds	July 30, 2017 at 2:13:58 PM ...	glusterfs
<input type="checkbox"/>	minio	i-0f07f4a1ce2cb4a46	t2.micro	eu-central-1b	running	2/2 checks ...	None	54.93.75.28	clouds	July 31, 2017 at 1:47:24 PM ...	minio
<input type="checkbox"/>	riakcs	i-03c909d3cb69461d9	t2.micro	eu-central-1b	running	2/2 checks ...	None	35.159.19.63	clouds	July 25, 2017 at 6:05:05 PM ...	riak
<input type="checkbox"/>	scalitys3	i-0ff4e9814d31e3b52	t2.micro	eu-central-1b	running	2/2 checks ...	None	54.93.227.231	clouds	July 29, 2017 at 5:54:16 PM ...	scalitys3
<input type="checkbox"/>	swift	i-05555ffc89f86cd7e	t2.micro	eu-central-1b	running	2/2 checks ...	None	35.158.165.85	clouds	August 1, 2017 at 9:35:27 P...	swift
<input type="checkbox"/>	s3ninja	i-0ac6f0f5f3fec7c14	t2.micro	eu-central-1b	running	2/2 checks ...	None	54.93.107.140	clouds	July 31, 2017 at 1:47:32 PM ...	s3ninja
<input type="checkbox"/>	s3rver	i-0f5727528a4bdd8ed	t2.micro	eu-central-1b	running	2/2 checks ...	None	52.59.211.37	clouds	July 28, 2017 at 6:21:51 PM ...	s3rver

Abbildung B.1: Alle erstellten Instanzen innerhalb der AWS. Die IP von Swift ist blau, da dies eine Elastic IP ist, welche von Vagrant erzwungen wurde.

C VirtualBox VM Übersicht

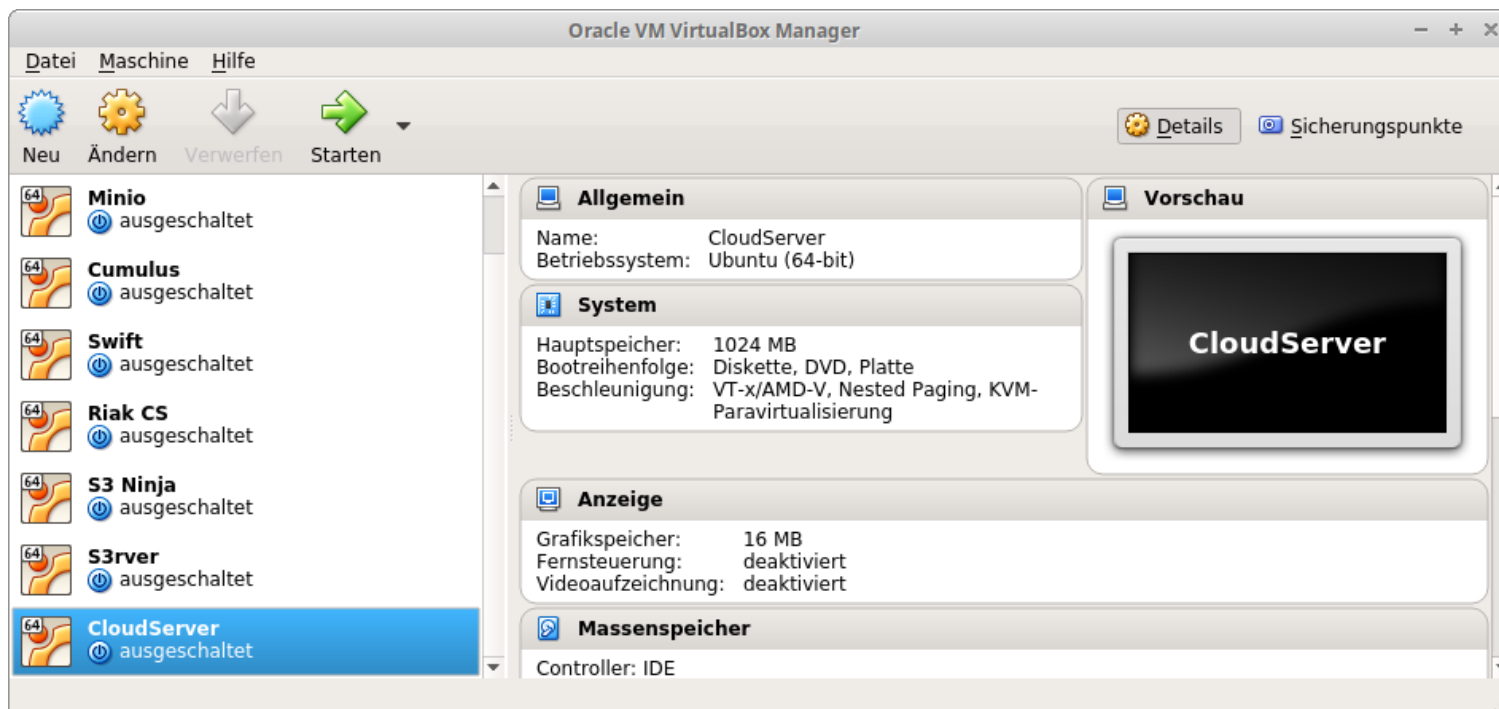


Abbildung C.1: Alle erstellten VMs in VirtualBox.

D Einzelheiten der Tests

D.1 Amazon S3

1 MB

```
$ ./s3perf.sh -n 5 -s 1048576 -p
...
Required time to create the bucket:          2.307s
Required time to upload the files:          11.479s
Required time to fetch a list of files:      1.163s
Required time to download the files:         4.342s
Required time to erase the objects:          2.517s
Required time to erase the bucket:           2.792s
Required time to perform all S3-related operations: 24.600s
```

2 MB

```
$ ./s3perf.sh -n 5 -s 2097152 -p
...
Required time to create the bucket:          2.115s
Required time to upload the files:          13.108s
Required time to fetch a list of files:      1.477s
Required time to download the files:         4.412s
Required time to erase the objects:          2.755s
Required time to erase the bucket:           2.395s
Required time to perform all S3-related operations: 26.262s
```

4 MB

```
$ ./s3perf.sh -n 5 -s 4194304 -p
...
Required time to create the bucket:          2.606s
Required time to upload the files:          25.707s
Required time to fetch a list of files:      3.187s
Required time to download the files:         6.994s
Required time to erase the objects:          2.480s
Required time to erase the bucket:           4.494s
```

```
Required time to perform all S3-related operations: 45.468s
```

8 MB

```
$ ./s3perf.sh -n 5 -s 8388608 -p
...
Required time to create the bucket:                2.426s
Required time to upload the files:                 50.424s
Required time to fetch a list of files:            1.578s
Required time to download the files:               10.067s
Required time to erase the objects:                2.706s
Required time to erase the bucket:                 2.628s
Required time to perform all S3-related operations: 69.829s
```

16 MB

```
$ ./s3perf.sh -n 5 -s 16777216 -p
...
Required time to create the bucket:                2.083s
Required time to upload the files:                 76.304s
Required time to fetch a list of files:            2.134s
Required time to download the files:               18.414s
Required time to erase the objects:                2.998s
Required time to erase the bucket:                 2.635s
Required time to perform all S3-related operations: 104.568s
```

D.2 Azure BS

1 MB

```
$ ./s3perf.sh -n 5 -s 1048576 -z -p
...
Required time to create the bucket:                2.652s
Required time to upload the files:                 7.577s
Required time to fetch a list of files:            2.411s
Required time to download the files:               4.226s
Required time to erase the objects:                17.950s
Required time to erase the bucket:                 2.540s
Required time to perform all S3-related operations: 37,356s
```

2 MB

```
$ ./s3perf.sh -n 5 -s 2097152 -z -p
...
Required time to create the bucket:          2.771s
Required time to upload the files:          12.408s
Required time to fetch a list of files:      2.435s
Required time to download the files:         6.204s
Required time to erase the objects:          17.756s
Required time to erase the bucket:           2.581s
Required time to perform all S3-related operations: 44,155s
```

4 MB

```
$ ./s3perf.sh -n 5 -s 4194304 -z -p
...
Required time to create the bucket:          2.776s
Required time to upload the files:          21.544s
Required time to fetch a list of files:      2.705s
Required time to download the files:         8.689s
Required time to erase the objects:          19.491s
Required time to erase the bucket:           2.137s
Required time to perform all S3-related operations: 57,342s
```

8 MB

```
$ ./s3perf.sh -n 5 -s 8388608 -z -p
...
Required time to create the bucket:          2.543s
Required time to upload the files:          39.407s
Required time to fetch a list of files:      2.725s
Required time to download the files:        13.112s
Required time to erase the objects:          19.208s
Required time to erase the bucket:           3.077s
Required time to perform all S3-related operations: 80,072s
```

16 MB

```
$ ./s3perf.sh -n 5 -s 16777216 -z -p
...
Required time to create the bucket:          2.876s
Required time to upload the files:          75.905s
Required time to fetch a list of files:      2.640s
Required time to download the files:        22.210s
Required time to erase the objects:          16.213s
Required time to erase the bucket:           2.264s
Required time to perform all S3-related operations: 122,108s
```

D.3 Google CS

1 MB

```
$ ./s3perf.sh -n 5 -s 1048576 -g -p
...
Required time to create the bucket:          10.060s
Required time to upload the files:           9.296s
Required time to fetch a list of files:      1.933s
Required time to download the files:         6.208s
Required time to erase the objects:          3.251s
Required time to erase the bucket:           2.441s
Required time to perform all S3-related operations: 33.189s
```

2 MB

```
$ ./s3perf.sh -n 5 -s 2097152 -g -p
...
Required time to create the bucket:          10.676s
Required time to upload the files:           13.674s
Required time to fetch a list of files:      2.039s
Required time to download the files:         7.910s
Required time to erase the objects:          3.121s
Required time to erase the bucket:           3.142s
Required time to perform all S3-related operations: 40.562s
```

4 MB

```
$ ./s3perf.sh -n 5 -s 4194304 -g -p
...
Required time to create the bucket:          10.153s
Required time to upload the files:           24.900s
Required time to fetch a list of files:      1.955s
Required time to download the files:         10.248s
Required time to erase the objects:          3.187s
Required time to erase the bucket:           3.052s
Required time to perform all S3-related operations: 53.495s
```

8 MB

```
$ ./s3perf.sh -n 5 -s 8388608 -g -p
...
Required time to create the bucket:          10.357s
Required time to upload the files:           41.755s
Required time to fetch a list of files:      2.038s
```

```
Required time to download the files:      14.427s
Required time to erase the objects:       3.006s
Required time to erase the bucket:        2.773s
Required time to perform all S3-related operations: 74.356s
```

16 MB

```
$ ./s3perf.sh -n 5 -s 16777216 -g -p
...
Required time to create the bucket:        10.626s
Required time to upload the files:         82.176s
Required time to fetch a list of files:    1.837s
Required time to download the files:       24.943s
Required time to erase the objects:        3.464s
Required time to erase the bucket:         2.968s
Required time to perform all S3-related operations: 126.014s
```

D.4 Minio

1 MB

```
$ ./s3perf.sh -n 5 -s 1048576 -p
...
Required time to create the bucket:        0.293s
Required time to upload the files:         5.138s
Required time to fetch a list of files:    0.325s
Required time to download the files:       1.813s
Required time to erase the objects:        0.926s
Required time to erase the bucket:         0.410s
Required time to perform all S3-related operations: 8.905s
```

2 MB

```
$ ./s3perf.sh -n 5 -s 2097152 -p
...
Required time to create the bucket:        0.289s
Required time to upload the files:         9.629s
Required time to fetch a list of files:    0.308s
Required time to download the files:       3.390s
Required time to erase the objects:        1.031s
Required time to erase the bucket:         0.431s
Required time to perform all S3-related operations: 15.078s
```

4 MB

```
$ ./s3perf.sh -n 5 -s 4194304 -p
...
Required time to create the bucket:          0.292s
Required time to upload the files:          18.627s
Required time to fetch a list of files:      0.302s
Required time to download the files:         5.111s
Required time to erase the objects:          0.989s
Required time to erase the bucket:           0.397s
Required time to perform all S3-related operations: 25.718s
```

8 MB

```
$ ./s3perf.sh -n 5 -s 8388608 -p
...
Required time to create the bucket:          0.305s
Required time to upload the files:          36.652s
Required time to fetch a list of files:      0.345s
Required time to download the files:         10.481s
Required time to erase the objects:          0.934s
Required time to erase the bucket:           0.389s
Required time to perform all S3-related operations: 49.106s
```

16 MB

```
$ ./s3perf.sh -n 5 -s 16777216 -p
...
Required time to create the bucket:          0.300s
Required time to upload the files:          74.765s
Required time to fetch a list of files:      0.332s
Required time to download the files:         16.248s
Required time to erase the objects:          0.918s
Required time to erase the bucket:           0.609s
Required time to perform all S3-related operations: 93.172s
```

D.5 Cumulus

1 MB

```
$ ./s3perf.sh -n 5 -s 1048576 -u
...
Required time to create the bucket:          0.306s
Required time to upload the files:           5.152s
Required time to fetch a list of files:      0.315s
```



```
Required time to download the files:      1.573s
Required time to erase the objects:       0.596s
Required time to erase the bucket:        0.307s
Required time to perform all S3-related operations: 8.249s
```

2 MB

```
$ ./s3perf.sh -n 5 -s 2097152 -u
...
Required time to create the bucket:        0.333s
Required time to upload the files:         9.699s
Required time to fetch a list of files:    0.334s
Required time to download the files:       2.543s
Required time to erase the objects:        0.594s
Required time to erase the bucket:         0.305s
Required time to perform all S3-related operations: 13.808s
```

4 MB

```
$ ./s3perf.sh -n 5 -s 4194304 -u
...
Required time to create the bucket:        0.322s
Required time to upload the files:        18.762s
Required time to fetch a list of files:    0.335s
Required time to download the files:       4.503s
Required time to erase the objects:        0.588s
Required time to erase the bucket:         0.347s
Required time to perform all S3-related operations: 24.857s
```

8 MB

```
$ ./s3perf.sh -n 5 -s 8388608 -u
...
Required time to create the bucket:        0.305s
Required time to upload the files:        37.497s
Required time to fetch a list of files:    0.376s
Required time to download the files:       8.563s
Required time to erase the objects:        0.670s
Required time to erase the bucket:         0.309s
Required time to perform all S3-related operations: 47.720s
```

16 MB

```
$ ./s3perf.sh -n 5 -s 16777216 -u
...
Bucket S3PERF-TESTBUCKET has been created.
```

```
ERROR: S3 error: 403 (AccessDenied): Access Denied
Unable to upload the files.
```

D.6 Swift

1 MB

```
$ ./s3perf.sh -n 5 -s 1048576 -a -p
...
Required time to create the bucket:          0.733s
Required time to upload the files:          6.323s
Required time to fetch a list of files:      0.679s
Required time to download the files:         2.845s
Required time to erase the objects:          2.049s
Required time to erase the bucket:           0.700s
Required time to perform all S3-related operations: 13.329s
```

2 MB

```
$ ./s3perf.sh -n 5 -s 2097152 -a -p
...
Required time to create the bucket:          0.675s
Required time to upload the files:          10.618s
Required time to fetch a list of files:      0.649s
Required time to download the files:         4.133s
Required time to erase the objects:          2.051s
Required time to erase the bucket:           0.834s
Required time to perform all S3-related operations: 18.960s
```

4 MB

```
$ ./s3perf.sh -n 5 -s 4194304 -a -p
...
Required time to create the bucket:          0.675s
Required time to upload the files:          19.520s
Required time to fetch a list of files:      0.666s
Required time to download the files:         5.754s
Required time to erase the objects:          2.074s
Required time to erase the bucket:           0.693s
Required time to perform all S3-related operations: 29.382s
```

8 MB

```
$ ./s3perf.sh -n 5 -s 8388608 -a -p
...
Required time to create the bucket:          0.681s
Required time to upload the files:          38.072s
Required time to fetch a list of files:      0.719s
Required time to download the files:         8.962s
Required time to erase the objects:          2.226s
Required time to erase the bucket:           0.791s
Required time to perform all S3-related operations: 51.451s
```

16 MB

```
$ ./s3perf.sh -n 5 -s 16777216 -a -p
...
Required time to create the bucket:          0.748s
Required time to upload the files:          73.826s
Required time to fetch a list of files:      0.689s
Required time to download the files:         16.507s
Required time to erase the objects:          1.957s
Required time to erase the bucket:           0.701s
Required time to perform all S3-related operations: 94.428s
```

D.7 Riak CS

1 MB

```
$ ./s3perf.sh -n 5 -s 1048576 -p
...
Required time to create the bucket:          0.349s
Required time to upload the files:           9.789s
Required time to fetch a list of files:      0.315s
Required time to download the files:         1.682s
Required time to erase the objects:           0.924s
Required time to erase the bucket:           0.508s
Required time to perform all S3-related operations: 13.567s
```

2 MB

```
$ ./s3perf.sh -n 5 -s 2097152 -p
...
Required time to create the bucket:          0.333s
Required time to upload the files:          19.250s
Required time to fetch a list of files:      0.321s
```

```
Required time to download the files:          2.587s
Required time to erase the objects:           0.980s
Required time to erase the bucket:            0.506s
Required time to perform all S3-related operations: 23.977s
```

4 MB

```
$ ./s3perf.sh -n 5 -s 4194304 -p
...
Required time to create the bucket:           0.343s
Required time to upload the files:            35.277s
Required time to fetch a list of files:       0.318s
Required time to download the files:          5.071s
Required time to erase the objects:           0.928s
Required time to erase the bucket:            0.545s
Required time to perform all S3-related operations: 42.482s
```

8 MB

```
$ ./s3perf.sh -n 5 -s 8388608 -p
...
Required time to create the bucket:           0.332s
Required time to upload the files:            69.058s
Required time to fetch a list of files:       0.340s
Required time to download the files:          8.677s
Required time to erase the objects:           0.923s
Required time to erase the bucket:            0.534s
Required time to perform all S3-related operations: 79.864s
```

16 MB

```
$ ./s3perf.sh -n 5 -s 16777216 -p
...
ERROR: S3 error: 403 (AccessDenied): Access Denied
Unable to upload the files.
```

D.8 S3 Ninja

1 MB

```
$ ./s3perf.sh -n 5 -s 1048576 -u -p
...
Required time to create the bucket:           0.304s
Required time to upload the files:            5.511s
Required time to fetch a list of files:       0.464s
```

```
Required time to download the files:          1.977s
Required time to erase the objects:           0.964s
Required time to erase the bucket:            0.317s
Required time to perform all S3-related operations: 9.537s
```

2 MB

```
$ ./s3perf.sh -n 5 -s 2097152 -u -p
...
Required time to create the bucket:           0.301s
Required time to upload the files:            9.647s
Required time to fetch a list of files:       0.530s
Required time to download the files:          2.930s
Required time to erase the objects:           1.013s
Required time to erase the bucket:            0.346s
Required time to perform all S3-related operations: 14.767s
```

4 MB

```
$ ./s3perf.sh -n 5 -s 4194304 -u -p
...
Required time to create the bucket:           0.310s
Required time to upload the files:            18.646s
Required time to fetch a list of files:       0.465s
Required time to download the files:          4.849s
Required time to erase the objects:           0.987s
Required time to erase the bucket:            0.308s
Required time to perform all S3-related operations: 25.565s
```

8 MB

```
$ ./s3perf.sh -n 5 -s 8388608 -u -p
...
Required time to create the bucket:           0.296s
Required time to upload the files:            37.591s
Required time to fetch a list of files:       0.580s
Required time to download the files:          8.702s
Required time to erase the objects:           0.942s
Required time to erase the bucket:            0.332s
Required time to perform all S3-related operations: 48.443s
```

16 MB

```
$ ./s3perf.sh -n 5 -s 16777216 -u -p
...
Required time to create the bucket:           0.296s
```

```
Required time to upload the files:          74.612s
Required time to fetch a list of files:     0.785s
Required time to download the files:        16.143s
Required time to erase the objects:         0.991s
Required time to erase the bucket:          0.314s
Required time to perform all S3-related operations: 93.141s
```

D.9 S3rver

1 MB

```
$ ./s3perf.sh -n 5 -s 1048576 -p
...
Required time to create the bucket:          0.371s
Required time to upload the files:          5.269s
Required time to fetch a list of files:     0.365s
Required time to download the files:        1.934s
Required time to erase the objects:         0.999s
Required time to erase the bucket:          0.491s
Required time to perform all S3-related operations: 9.429s
```

2 MB

```
$ ./s3perf.sh -n 5 -s 2097152 -p
...
Required time to create the bucket:          0.319s
Required time to upload the files:          9.779s
Required time to fetch a list of files:     0.359s
Required time to download the files:        2.731s
Required time to erase the objects:         1.119s
Required time to erase the bucket:          0.460s
Required time to perform all S3-related operations: 14.767s
```

4 MB

```
$ ./s3perf.sh -n 5 -s 4194304 -p
...
Required time to create the bucket:          0.336s
Required time to upload the files:          18.637s
Required time to fetch a list of files:     0.363s
Required time to download the files:        5.130s
Required time to erase the objects:         1.007s
Required time to erase the bucket:          0.433s
Required time to perform all S3-related operations: 25.906s
```

8 MB

```
$ ./s3perf.sh -n 5 -s 8388608 -p
...
Required time to create the bucket:          0.336s
Required time to upload the files:          37.300s
Required time to fetch a list of files:      0.357s
Required time to download the files:         8.277s
Required time to erase the objects:          1.059s
Required time to erase the bucket:           0.462s
Required time to perform all S3-related operations: 47.791s
```

16 MB

```
$ ./s3perf.sh -n 5 -s 16777216 -p
...
ERROR: S3 error: 404 (Not Found)
Unable to upload the files.
```

D.10 Scalify CloudServer

1 MB

```
$ ./s3perf.sh -n 5 -s 1048576 -p
...
Required time to create the bucket:          0.431s
Required time to upload the files:           5.371s
Required time to fetch a list of files:      0.434s
Required time to download the files:         2.278s
Required time to erase the objects:          1.224s
Required time to erase the bucket:           0.703s
Required time to perform all S3-related operations: 10.441s
```

2 MB

```
$ ./s3perf.sh -n 5 -s 2097152 -p
...
Required time to create the bucket:          0.393s
Required time to upload the files:           9.692s
Required time to fetch a list of files:      0.411s
Required time to download the files:         2.909s
Required time to erase the objects:          1.110s
Required time to erase the bucket:           0.630s
Required time to perform all S3-related operations: 15.145s
```

4 MB

```
$ ./s3perf.sh -n 5 -s 4194304 -p
...
Required time to create the bucket:          0.391s
Required time to upload the files:          18.712s
Required time to fetch a list of files:      0.378s
Required time to download the files:         4.601s
Required time to erase the objects:          1.057s
Required time to erase the bucket:           0.588s
Required time to perform all S3-related operations: 25.727s
```

8 MB

```
$ ./s3perf.sh -n 5 -s 8388608 -p
...
Required time to create the bucket:          0.382s
Required time to upload the files:          37.405s
Required time to fetch a list of files:      0.372s
Required time to download the files:         8.295s
Required time to erase the objects:          1.104s
Required time to erase the bucket:           0.588s
Required time to perform all S3-related operations: 48.146s
```

16 MB

```
$ ./s3perf.sh -n 5 -s 16777216 -p
...
Required time to create the bucket:          0.389s
Required time to upload the files:          73.560s
Required time to fetch a list of files:      0.367s
Required time to download the files:        16.146s
Required time to erase the objects:          1.124s
Required time to erase the bucket:           0.847s
Required time to perform all S3-related operations: 92.433s
```