Abschlussbericht

IAI - Institut für Angewandte Informatik

Ansätze zur Software-Virtualisierung für KMU - eine empirische Bestandsaufnahme

Herbert Kuchen, Vincent von Hof, Andreas Fuchs

Inhaltsverzeichnis

Abbildungsverzeichnis

Executive Summary

Dieser Bericht stellt die Ergebnisse einer Befragung von verschiedenen kleinen und mittleren Unternehmen hinsichtlich ihrer bisherigen Erfahrungen und Nutzungen von Software-Virtualisierungen dar. Insgesamt wurden sieben Kleine und mittlere Unternehmen (KMU) aus verschiedenen Branchen wie beispielsweise der Finanzdienstleistung, Infrastruktur-Anbieter und Software-Dienstleister befragt. Es hat sich gezeigt, dass der Einsatz von Software-Virtualisierung in allen KMU ein wichtiges Thema ist und dementsprechend viele Software-Services der KMU bereits virtualisiert sind. Neuerdings werden immer häufiger Container-basierte Virtualisierungen – wie beispielsweise mit Hilfe von *Docker* – mit dem Ziel eingesetzt, die Verfügbarkeit von Software-Services und dadurch auch die Robustheit der angebotenen Software im Unternehmen zu erhöhen.

1 Einführung

Die Schlagworte *Cloud Computing* und *DevOps* sind aktuelle Themen, mit denen sich viele Unternehmen bei IT-Entscheidungen beschäftigen. Unter Cloud-Computing wird die Bereitstellung und Nutzung von einer IT-Infrastruktur über das Internet verstanden. Diese Infrastruktur kann beispielsweise Speicherplatz, Rechenleistung oder Software-Services bereitstellen. Die grundlegende Idee dabei ist, dass diese IT-Infrastruktur nicht auf lokalen Rechnern installiert werden muss, sondern über ein Rechnernetz zur Verfügung gestellt wird. Anwendungen, die in der Cloud laufen, versprechen beispielsweise eine höhere Ausfallsicherheit und damit einen reibungsloseren Ablauf.

Die Nutzung von Cloud-Diensten hat für Unternehmen unter anderem einen Kostenvorteil, welcher zudem gut kalkulierbar ist. Teilweise erhebliche Investitionen für den Betrieb von Software-Services müssen nicht mehr selbst vorgenommen werden, sondern können von einem externen Dienstleister flexibel angemietet werden. Zudem sind die Kosten scheinbar gut kalkulierbar, da die Nutzungstarife bekannt sind. Zuletzt stellten Analysten von Gartner heraus, dass “Viele Unternehmen [...] entsetzt (sind), wenn sie ihre ersten Cloud-Rechnungen bekommen, da diese weit höher sind als veranschlagt” [6]; daher schlussfolgern die Analysten, dass der Begriff Cloud nunmehr im “Tal der Ernüchterung” bezüglich des Hype-Cycles angelangt sei. Dennoch versprechen private oder hybride Clouds weiterhin Vorteile im Hinblick auf skalierbare IT-Infrastruktur und installations- und wartungsfreie IT-Anwendungen [4].

Um ein Programm in der Cloud lauffähig zu machen, bedarf es einer abstrakten Schicht zwischen der Hard- und Software, welche man mit der Virtualisierung des Programmes erreichen kann. In der Informatik versteht man unter dem Begriff *Virtualisierung* die Erzeugung eines virtuellen Objekts – wie beispielsweise einer emulierten Hardware, eines Betriebssystems (BS) oder einer Netzwerkressource – so dass auf einer abstrakten Ebene eine Entkoppelung von der physischen Hardware stattfindet. Damit ist es möglich, die vorhanden IT-Ressourcen besser auszulasten sowie die Flexibilität zu erhöhen. Unter anderem haben diese großen Vorteile der Virtualisierung dazu geführt, dass Virtualisierungstechniken heutzutage in nahezu jedem Unternehmen eingesetzt—beziehungsweise indirekt genutzt—werden, welches eine beliebige Form von IT-Systemen selbst betreibt oder nutzt.

In jüngster Zeit hat eine neue Form der Virtualisierung sehr starke Aufmerksamkeit bekommen, welche auf die Virtualisierung von Betriebssystemen und Anwendungen zielt: die *Container-basierte* Virtualisierung. Das Ziel dieser Virtualisierungs-Technik ist es, dass mehrere Instanzen eines Betriebssystems isoliert voneinander auf einem Hostsystem betrieben werden können. Das besondere dabei ist, dass diese Instanzen einen gemeinsamen Kernel im Betriebssystem nutzen. Die Container-basierte Virtualisierung

hat den großen Vorteil, dass dadurch Rechen-Ressourcen gespart werden können, die normalerweise für das Betriebssystem jeder einzelnen Anwendung aufgebracht werden müssen, während gleichzeitig nicht komplett auf Isolierung verzichten werden muss. Eine sehr verbreitete Container-basierte Virtualisierungs-Engine ist *Docker* [9]. Zur horizontalen Skalierung (durch Replikation von Komponenten), zur Lastverteilung und zum Konfigurationsmanagement hat sich Kubernetes [10] hervorgetan.

Nicht zuletzt durch die weite Verbreitung dieser Technologien und Methoden, hat der Begriff “DevOps” in den letzten Jahren an Bedeutung gewonnen. DevOps bezeichnet, grob gesagt, die schnelle und reibungslose Integration von Entwicklung (Development), Betrieb (Operations) und Qualitätssicherung, die im Idealfall dazu führt, dass neu enwickelte oder geänderte Komponenten automatisch getestet und, falls keine Fehler auftraten, in das lauffähige System übernommen werden (Deployment). Nach aktuellen Umfragen konnte DevOps in vielen Unternehmen zuvor nur schwer umgesetzt werden und befindet sich nunmehr im Tal der Desillusionierung bezüglich des Gartner Hype Cycles.

Im Rahmen des vorliegenden Projektes wird untersucht, welche Erfahrungen, Hoffnungen und Erwartungen KMU an Virtualisierung generell, und die Container-basierte Virtualisierung im Speziellen, haben.

2 Hintergrund: Virtualisierung

Sobald Anwendungen auf einer einzelnen physischen Maschine ohne Beeinflussung von anderen Anwendungen laufen sollen, müssen sie gegeneinander isoliert werden. Diese Isolierung betrifft Systemressourcen und muss durch die ausführende Umgebung unterstützt werden. Ein Software-Artefakt hierbei mit einem kontrollierten Anteil der Systemressourcen versorgt werden [5].

Lösungsansätze für das Problem der Software-Isolierung finden sich sowohl in der Hardware- als auch in der Software-Virtualisierung. In den folgenden Kapiteln werden nun in Kapitel 2.1 diese Ansätze erläutert und kontrastiert.

2.1 Software-Container

Die meisten Betriebssysteme gewährleisten Isolierung durch Hardware-Schutz-Mechanismen wie Speicherunterteilung, Speicherseiten-Mapping und unterscheidbare Benutzer- und Kernel-Instruktionen. Ein softwareisolierter Prozess (SIP) ist ein Prozess, dessen Grenzen durch Sprachsicherheitsregeln festgelegt und durch statische Typprüfung erzwungen werden. AIKEN ET AL. kommen zu dem Schluss, dass SIP Rechenleistungs-günstige Isolierungsmechanismen bieten, die Fehlerisolierung ermöglichen ohne Interprozesskommunikation einzuschränken [1]. Es stellt sich heraus, dass hardwarebasierte Isolierung die Performance der Software um bis zu 35% reduzieren kann. Die Software-Isolierung hingegen kommt auf weniger als 5% Overhead. Die geringeren Laufzeiterhöhungen von SIPs ermöglichen den Einsatz in einer feineren Granularität als bei herkömmlichen Verfahren. Allerdings bleibt die Hardware-Isolation als Abwehrmaßnahme gegen mögliche Ausfälle von Software-Isolierungsmechanismen wertvoll. Die Fähigkeit, eine Abwägung von Kosten und Nutzen der einzelnen Isolationstechniken durchführen zu können, wird proportional zur Ausweitung der Software-Isolierung an Relevanz gewinnen.

2.1.1 Historie der Software-Container

Das Konzept des Software-Isolierung ist mitnichten eine neue Erscheinung. Schon 1979 wurde mit `chroot` in Linux ein Kommando eingeführt, um System-Ressourcen zu isolieren. 1988 folgte mit *software jails* eine Erweiterung des Ansatzes [2, p. 82]. In der zweiten Dekade des 21. Jahrhunderts schlossen sich Lösungen wie *Rocket*, LXD und *Docker* an, die allesamt darauf abzielten wieder-benutzbare Software-Container bereitzustellen und ein Management der Container-Konstellationen zu ermöglichen. Von diesen Lösungen kristallisierte sich schnell Docker als besonders beliebt heraus. Anfang 2016

tauchte das Quellcode Verzeichnis zum Kern von Docker auf dem 20. Platz der Quellcode Management Plattform *Github.com* “Sternchen” Liste auf, die Projekte nach Anzahl der Empfehlungen für ein Projekt sortiert — damit rangierte das Projekt nur 4 Plätze hinter dem Linux-Kernel-Quellcode-Verzeichnis [7]. Docker selbst wird mit einer Reihe von Dienstprogrammen ausgeliefert, die die Hauptfunktionalität der Software-Virtualisierung um Funktionen erweitern, die Nutzer beim Einrichten, Bereitstellen und Verwalten einer verteilten Umgebung unterstützen.

2.1.2 Abgrenzung von Hard- und Software-Virtualisierung

Sollen mehrere Anwendungen auf einer einzigen physischen Maschine ohne gegenseitige Beeinflussung laufen, so sind sie hinsichtlich der Systemressourcen und der Ausführungsumgebung zu isolieren. Idealerweise können sie mit einem kontrollierbaren Anteil an Systemressourcen ausgestattet werden [5]. Diese Probleme können, wie besprochen, sowohl von virtuellen Maschinen als auch von Software-Containern gelöst werden. *Virtualisierung* zielt darauf ab, das Vorhandensein mehrerer Maschinen auf einer einzigen Maschine zu simulieren. Für die Virtualisierung auf der Hardware-Ebene läuft ein *Hypervisor*-Dienst auf dem BS des Hosts oder direkt auf seiner Hardware und abstrahiert ihn durch die Bereitstellung virtueller Schnittstellen. Bei Virtualisierung auf der BS-Ebene — diejenige, die von Software-Containern verwendet wird — wird die Abstraktion durch die Offenlegung der Host-Kernelfunktionen über Linux LXC bereitgestellt, welche die Linux-Steuerungsgruppen *cgroup* und *namespaces* verwendet, um den Zugriff auf die Ressourcen des Hosts zu beschränken und so die Isolierung zu gewährleisten.

Im Gegensatz zur traditionellen Virtualisierung reduziert die Containerisierung sowohl den Platzbedarf von Anwendungsumgebungen als auch die Zeit, die zum Starten einer Anwendung und zum Bereitstellen des zugehörigen Speichers benötigt wird, falls mehrere ähnliche Anwendungen auf der gleichen Maschine eingesetzt werden. Dies lässt sich mit Hilfe von Figur 1 erläutern.

Eine **virtuelle Maschine** in der Hardware-Virtualisierung besteht aus drei Elementen: dem Betriebssystem, den Bibliotheken, welche für die Anwendung benötigt werden, und trivialerweise der Anwendung selbst. Im Gegensatz dazu besteht ein **Container** in der Software-Virtualisierung nur aus zwei Elementen: Den für die Anwendung benötigten Bibliotheken und der Anwendung selbst.

Da also der BS-Kernel von Containern gemeinsam genutzt wird, ist es nicht mehr notwendig, mehrere Kernel (BS-Level) im Speicher zu halten, wodurch entsprechend der Speicherbedarf reduziert wird. Trivialerweise wird Speicherplatz gespart, da nicht mehrere BS-Instanzen vorgehalten werden müssen.

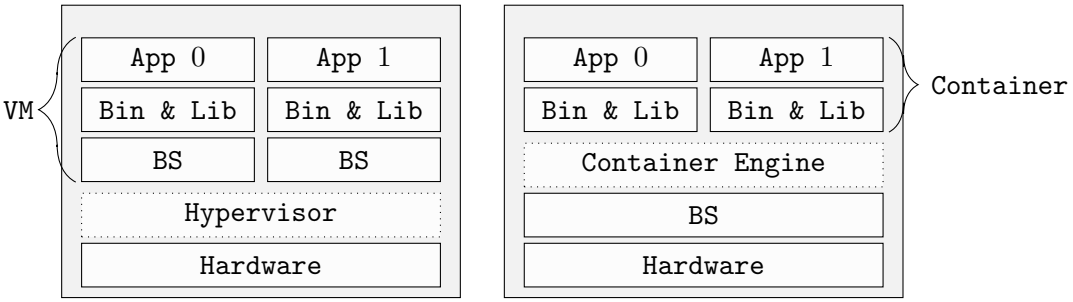


Abbildung 1: Hard- and BS-level Virtualisierung.

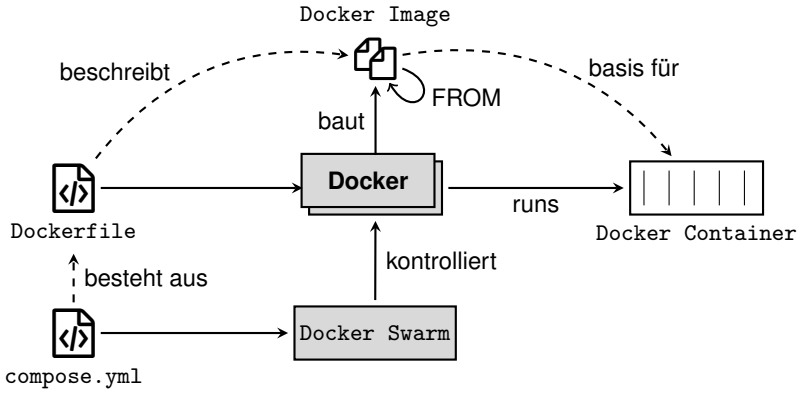


Abbildung 2: Zusammenhang der Docker-Begrifflichkeiten.

Da zweitens beim Starten einer Anwendung das Starten des Gast-BS entfällt, ist die Gesamtzeit, die bis zum Starten der Anwendung vergeht, im Falle der Software-Virtualisierung geringer. Schließlich wird die Isolierung des Dateisystems durch einen Mechanismus namens *Copy-on-Write (CoW)* gewährleistet, der die Notwendigkeit eliminiert, Kopien des gesamten Dateisystems für einzelne Container zu erstellen. Stattdessen teilen sich Container den Zugriff auf Dateien für den Lesezugriff. Erst wenn ein Container eine Datei ändern muss, wird eine Kopie erstellt. Um weiteren Platz zu sparen, können nur die Inkremente gespeichert werden. Die anderen Container sehen die genannten Änderungen nicht, so dass die erforderliche Isolierung gewährleistet ist.

Nennenswerte Softwarelösungen sind *chroot*, von 1979, und *software jails* 1998 [2]. Seit 2003 entstanden Lösungen wie rkt, LXD und Docker, die standardisierte wiederverwendbare Software-Container, Container-Management-Tools und sogar ein auf Software-Container zugeschnittenes BS bieten.

Die Kernbegriffe, die sich auf Docker beziehen, sind in Abbildung 2 dargestellt. Ein **Dockerfile**—wie es in Abbildung 3 aufgeführt ist—besteht aus einem Satz von Anweisungen, die, nachdem sie einmal ausgeführt wurden, zu einem CoW-basierten Dateisystem namens **Docker Image** führen. Die erste Anweisung im Dockerfile ist immer eine *Vererbungs*anweisung FROM, die auf ein anderes Docker-Image verweist. Die Images wer-

den in privaten oder öffentlichen Repositorien zugänglich gemacht. Schließlich wird ein Docker-Image in Form eines Docker-**Container**s instanziiert und dieses durch die Docker-Engine gestartet.

Das Image, das aus dem Dockerfile erzeugt werden soll, besteht im Kern aus einem Dateisystemzustand und einer Deklaration mehrerer Konnektoren, welche die Integration eines gestarteten Containers in das zugrundeliegende System ermöglichen, z.B. die Netzwerkverbindung über Ports oder der Zugriff auf die interne Hardware über gemountete Festplatten-Volumes.

Man stelle sich beispielhaft ein Image vor, das laut Dockerfile-Definition intern Port 8080 exponieren soll. Nehmen wir weiterhin an, dass aus diesem Image zwei Docker-Container *container1* und *container2* durch Instanziierung entstanden sin. *container1* kann dann beispielsweise Systemport 80 an den *container1*-internen Port 8080 binden, während *container2* den Systemport 81 an den *container2*-internen Port 8080 bindet, da Systemport 80 nun schon besetzt ist. Zusammengefasst lässt sich also sagen, dass Konfigurationen innerhalb des Containers fix gesetzt werden können, ohne Rücksicht auf Einstellungen nehmen zu müssen, die erst bei der Auslieferung relevant werden. Diese Abstraktion ermöglicht bei der Containerisierung Skalierbarkeit, da so viele Instanzen des gleichen Images, wie benötigt, bereitgestellt werden können. Die intern verwendeten Port-Nummern stören hierbei nicht.

Eine containerisierte Software kann praktisch überall eingesetzt werden, da sie nur wenige Annahmen über die zur Ausführzeit vorhandene Hardware oder BS treffen muss. Gleichzeitig kann durch den Container sichergestellt werden, dass Software-Abhängigkeiten der auszuliefernden Software vorhanden sind. Somit entfällt ein Großteil der Umgebungskonfigurationsarbeit bei der Installation.

2.2 Verknüpfung mehrerer Container zu einem Service von Servicen

```
version: '2'
```

```
# Ein anderes Image mit dem Namen "base-jdk" in Version 8, vom Nutzer "jboss" dient als Basis.
FROM jboss/base-jdk:8

# Die WILDFLY_VERSION wird als Umgebungsvariable im Container wie folgt gesetzt sein:
ENV JBOSS_HOME /opt/jboss/wildfly

# Lade und installiere die vorbestimmte Wildfly Version.
RUN curl -O https://download.jboss.org/wildfly/$WILDFLY_VERSION/wildfly-$WILDFLY_VERSION.tar.gz
... # unpack and install

# Definiere die vorhandene Netzwerkschnittstelle (an Hand eines Ports)
EXPOSE 8080

# Spezifiere das Start Kommando
CMD ["/opt/jboss/wildfly/bin/standalone.sh", "-b", "0.0.0.0"]
```

Abbildung 3: Beispielhaftes Dockerfile für eine Wildfly Installation.

```
2  services:
3    web:
4      build: ./Dockerfiles/wildfly/.
5      ports: - 8080
6      depends_on: -  db
7    db:
8      build: ./Dockerfiles/mysql/.
9
10   loadbalancer:
11     image: dockercloud/haproxy
12     ports:
13      - 80:80
14     links:
15      - web
16     volumes:
17      - /var/run/docker.sock:/var/run/docker.sock
```

Listing 1: Ausschnitt aus einer Compose-Datei.

Hierauf aufbauend ist eine Vielzahl von sich ergänzenden Tools entstanden, die die Docker-Engine um Funktionen erweitern, die helfen, Umgebungen zu verteilen, einzurichten, bereitzustellen und zu verwalten.

Erstens gibt es *Docker Swarm*, eine Abstraktionsschicht, die eine Reihe von *Docker Engines* in eine “virtuelle” Engine bündelt, welche *Swarm Manager* genannt wird. Der *Swarm Manager* wird verwendet, um mehrere Docker Engines wie eine zu steuern. Hinsichtlich der Zuordnung von Containern zu bestimmten Engines führt Docker Swarm Filter und Strategien ein. Anhand einer *Strategie* entscheidet Swarm beispielsweise, die Container gleichmäßig auf die Engines zu verteilen oder zuerst eine Engine aufzufüllen, bis sie ihr Containerlimit erreicht hat, bevor die nächsten Engine angesprochen wird. *Filter* hingegen können verwendet werden, um explizite Container-zu-Engine-Knoten-Zuweisungen zu setzen, basierend auf dem Knotennamen oder der Konfiguration von lokal eingesetzten Containern.

Schließlich erlaubt *Docker Compose* die Definition eines kompletten Dienstes, der aus mehreren Containern besteht, indem die impliziten Abhängigkeiten zwischen den benötigten einzelnen Containern in einer so genannten *compose.yml* Datei erklärt werden. Dies ist in

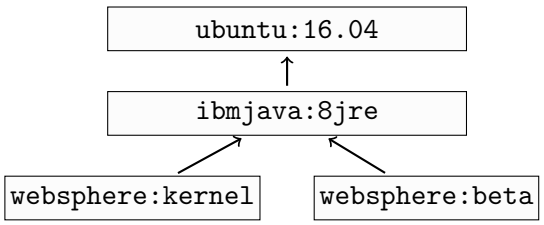


Abbildung 4: Beispiel des hierarchischen Aufbaus eines Docker-Images.

Listing 1 demonstriert. Eine breite Palette von Schlüsselwörtern wird unterstützt. Neben der Auflistung der einzelnen Container unterhalb von *(*Zeile 7) ist es möglich, zu definieren, welche Ports aus dem einzelnen Container exponiert werden sollen. Darüber hinaus kann man definieren, welche Container *verlinkt* sind, was zur Bildung eines Brücken-Netzwerkes zwischen ihnen führen wird.

Ein Image kann durch so viele Containern instanziiert werden, wie es die Auslastung des hierdurch bereitgestellten Dienstes erfordert. Weiterhin ist das Erweitern der Hardware zur Laufzeit durch Hinzufügen neuer Hosts zum Swarm einfach zu bewerkstelligen. Wenn ein Container z.B. aufgrund eines Hardware- oder BS-Fehlers verloren geht, erstellt *Swarm* automatisch als Ersatz einen neuen Container aus dem betreffenden Image.

3 Untersuchungsmethode der Studie

Im Folgenden wird erörtert, wie das Potential von Software-Virtualisierung in KMUs untersucht wurde. Dazu wurden Interviews mit den IT-Verantwortlichen der jeweiligen KMUs durchgeführt. Interviews sind eine Form der qualitativen Forschung, welche von zwei wesentlichen Prinzipien geprägt ist [3]: dem *Prinzip der Offenheit* und dem *Prinzip der Kommunikation*.

Das erste Prinzip besagt, dass eine theoretische Strukturierung des Forschungsgegenstandes erst dann vorgenommen wird, wenn dessen Strukturierung durch die beforschten Subjekte erhoben und analysiert wurde [3]. Im Rahmen dieses Projektes wurden in den Interviews verschiedene Fragen gestellt, welche auf unterschiedliche Dimensionen der übergeordnete Forschungsfrage zielten. Der Aufbau des Fragebogens wird in Unterkapitel 3.2.2 näher erläutert.

Das zweite Prinzip besagt, dass eine Erhebung von “bedeutungsstrukturierten Daten” nur durch eine Kommunikationsbeziehung mit dem Forschungsobjekt erfolgen kann, welche den Kommunikationsregeln des Beforschten folgt – und eben nicht denen der wissenschaftlichen Forschung [3]. Die Interviews wurden daher immer in persönlichen Gesprächen durchgeführt, welche im Unterkapitel 3.2.3 erläutert werden.

Im Rahmen dieses Projektes werden basierend auf diesen zwei Prinzipien der qualitativen Forschung die folgenden Forschungsfragen (FF) beantwortet:

FF1: Wie ist der aktuelle Stand der Einführung von Software-Virtualisierungen in KMUs?

FF2: Was sind generelle Hindernisse bei der Einführung von Container-basierten Virtualisierungslösungen?

3.1 Datenerhebung bei IAI Mitgliedern

Um die Forschungsfragen zu beantworten, wurde eine Einladung zu einem Gespräch an alle 30 Mitgliedsfirmen des Fördervereins des Instituts für Angwandte Informatik der Uni Münster (IAI) versendet. Aus diesem Kreis hatten sich sieben Mitglieder bereit erklärt, an diesem Projekt teilzunehmen. Dies entspricht zwar nur einer Teilnahmequote von 23%, jedoch waren die Projektteilnehmer eine sehr heterogene Gruppe. Wir konnten daher Informationen ermitteln, wie sehr unterschiedliche Unternehmen mit dem Thema Software-Virtualisierung vertraut sind, in welchem Umfang sie sie einsetzten und ob Container-basierte Virtualisierungslösungen momentan – oder in Zukunft geplant – eine Rolle spielen.

- 14

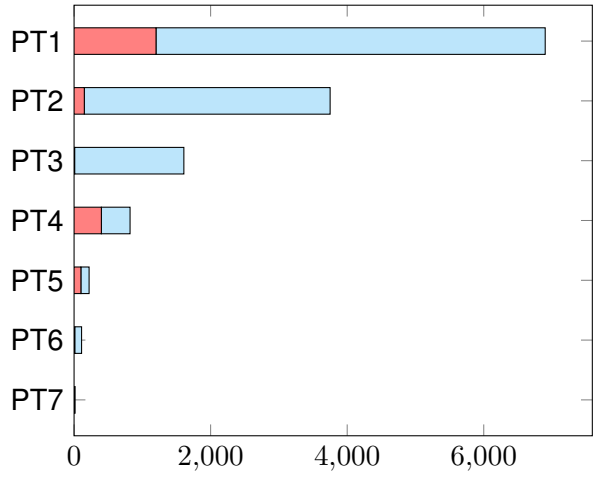


Abbildung 5: Anzahl der Mitarbeiter in den Unternehmen der Projektteilnehmer.

In Abbildung 5 ist die Anzahl der Mitarbeiter der sieben Projektteilnehmer (PT1, . . . , PT7) dargestellt. Der Projektteilnehmer PT1 hat mit ca. 5700 die meisten Mitarbeiter (blauer Balken), wovon etwa 1200 in der IT tätig sind (roter Balken). Auf der anderen Seite hat der Projektteilnehmer PT7 mit etwa 10 Mitarbeitern – davon 5 in der IT tätig – die wenigsten Mitarbeiter im Unternehmen beschäftigt.

Weiterhin sind die Unternehmen der Projektteilnehmer in vier verschiedenen Branchen tätig. In Abbildung 6 sind die Verteilungen der Branchen von den Unternehmen der Projektteilnehmer dargestellt. Aus der Abbildung ist ersichtlich, dass die meisten Projektteilnehmer hauptsächlich in der IT-Beratung tätig sind.

Alle Unternehmen haben gemein, dass sie IT-Systeme betreiben, welche durch den Ein-

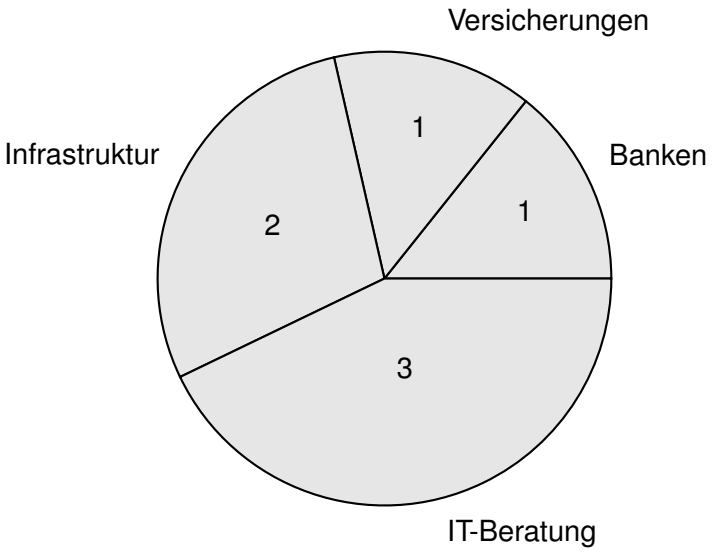


Abbildung 6: Verteilung der Branchen der Projektteilnehmer.

satz von Virtualisierungs-Techniken unterstützt werden können. Um dieses Unterstützungs-Potenzial aufzudecken, wurden mit den IT-Verantwortlichen der jeweiligen Unternehmen Interviews durchgeführt. Ein zentraler Bestandteil von Interviews ist der Wunsch, dass die befragte Zielgruppe möglichst selbst zu Wort kommt, um auch subjektive Sichtweisen erfassen zu können. Um die verschiedenen Interviews nach einem einheitlichen Muster durchführen zu können, wurde im Rahmen dieses Projektes ein Gesprächsleitfaden entwickelt. Der Entwurf dieses Interviews wird im folgenden Kapitel vorgestellt.

3.2 Problemzentriertes Interview

Interviews erlauben eine offene Interaktion in einem direkten Gespräch zwischen dem Interviewer und der bzw. den interviewten Personen. Das *problemzentrierte Interview* wird auf der Grundlage eines *Leitfandens* durchgeführt, in welchem offene Fragen gestellt werden [11]. Ein Interview-Leitfaden ist ein strukturiertes Dokument, das als Instrument der Vorbereitung auf das Gespräch dient. Im Rahmen dieses Projektes hatte der Leitfaden eine zentrale Bedeutung. Er diente unter anderem dazu, die einzelnen Interviews mit den Projektteilnehmern vergleichbar zu gestalten. Die Ergebnisse dieser Umfragen werden in Kapitel 4 beschrieben. In den folgenden beiden Unterkapiteln werden die Interview-Methodik (3.2.1) und der Interview-Leitfaden (3.2.2) genauer beschrieben.

3.2.1 Methodik

Die Methodik des problemzentrierten Interviews ist auf eine bestimmte Problemstellung zentriert [8]. Im Rahmen dieses Projektes stehen die Forschungsfragen FF1 und FF2 (s. Seite 13) im Zentrum. Das Verfahren des problemzentrierten Interviews geht auf WITZEL zurück [11], der diese als eine Methodenkombination aus Interview, biographischer Methode, Gruppendiskussion und Fallanalyse im Rahmen eines problemzentrierten Forschungsprojekts entwickelte [8]. Es handelt sich dabei um eine offene, halb-strukturierte Befragung, in welcher die Befragten möglichst frei zu Wort kommen [8].

Die Interviews wurden paarweise durchgeführt. Eine Person hat gesprächsführend die Fragen an den Interviewten gestellt und die andere Person die Schriftführung übernommen.

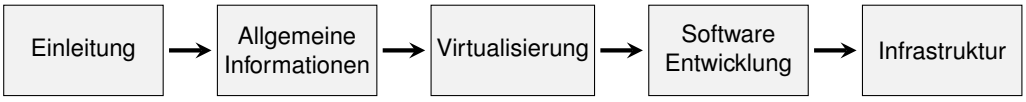


Abbildung 7: Struktur des Interview-Leitfadens.

3.2.2 Interview-Leitfaden

Abbildung 7 stellt den Aufbau des Leitfadens dar. Jedes Interview wurde mit einer Einleitung begonnen, in der der Interviewte über die Ziele des Interviews informiert wurde. Die Interviews wurden jeweils mit Einverständnis des Interviewten aufgezeichnet und in einer Nachbereitung transkribiert, um die Antworten auszuwerten und mit anderen Interview-Ergebnisse vergleichen zu können.

Nach der Einleitung sollten allgemeine Informationen zum Unternehmen und der angebotenen IT-Dienstleistungen identifiziert werden. Das Ziel dieses Schrittes ist es, einen Überblick über das Umfeld des Projektpartners zu bekommen und zu verstehen, in welchem Umfang das Unternehmen bereits Ressourcen (wie beispielsweise Mitarbeiter, s. Abbildung 5) für den Betrieb von Software-Virtualisierungen im Einsatz hat.

In einem dritten Schritt soll festgestellt werden, welches Wissen die Projektpartner bereits im Bereich Virtualisierung haben. Zusätzlich soll ermittelt werden, in welchen Bereichen im Unternehmen bereits Virtualisierungen von Software-Artefakten stattfinden und welche Technologien dafür verwendet werden. Von einem besonderen Interesse ist außerdem, welche Herausforderungen, Probleme und Chancen die IT-Verantwortlichen durch den Einsatz von Software-Virtualisierung – und im Speziellen von Container-basierter Virtualisierung – sehen.

In einem vierten Schritt sollte ermittelt werden, inwieweit Virtualisierung bei der Software-Entwicklung im Unternehmen eine Rolle spielt. Eine Software-Virtualisierung kann nicht nur genutzt werden, um Software-Services zu betreiben, sondern auch um einen Software-Entwickler bei seiner Arbeit zu unterstützen, indem beispielsweise benötigte Entwicklungs-Umgebungen virtualisiert werden. Ziel dieser Phase war es, diesen alternativen Blick auf Software-Virtualisierung im Unternehmen aufzuzeigen und zu identifizieren, inwieweit Virtualisierungstechniken – und dabei insbesondere Container-basierte Virtualisierung – im Unternehmen bei der Software-Entwicklung im Einsatz ist.

In einem letzten Schritt soll ermittelt werden, wie Container-basierte Virtualisierung bei der Wahl einer geeigneten Service-Infrastruktur unterstützend wirken kann. Außerdem soll in diesem Schritt identifiziert werden, wie das Unternehmen bestimmte Services virtualisiert – und ob dabei eine bestimmtes Transformationsschema verwendet wird. Ziel dieses Schrittes ist es, herauszufinden, ob das Unternehmen die Chancen von Container-basierter Virtualisierung so groß einschätzt, dass es eine solche in dem Unternehmen

einsetzen würde. Zudem soll herausgefunden werden, welche abstrakten Services in welchem Umfang von einer Transformation betroffen wären.

Zusammenfassend hat der Gesprächsleitfaden die folgenden drei Kernbereiche:

- Allgemeine Informationen zum Unternehmen der Projektpartner identifizieren.
- Kenntnisstand der Projektpartner im Bereich Software-Virtualisierung ermitteln.
- Lösungsansätze und Vorgehensmodelle zur Einführung von Software-Virtualisierung im Unternehmen aufzeigen.

3.2.3 Durchführung der Interviews

Die einzelnen Interviews wurden auf drei unterschiedliche Arten mit den IT-Verantwortlichen der KMUs durchgeführt.

- persönlich beim Teilnehmer vor Ort
- persönlich in den Räumlichkeiten der Universität
- persönlich in einem Telefon-Gespräch

Der Grund für ein nicht einheitliches Vorgehen war die Berücksichtigung von Bedürfnissen der befragten Personen. So waren beispielsweise nicht alle IT-Verantwortlichen im Raum Münster zugegen, um hier ein persönliches Treffen vor Ort durchführen zu können. Mit diesen Personen wurde ein Telefon-Interview durchgeführt.

Mit dem Einverständnis der befragten Personen wurden alle Interviews aufgezeichnet und in einer Nachbearbeitung transkribiert. Alle Interviews hatten den gleichen grundlegenden Aufbau, welcher im Fragebogen als Leitfaden festgelegt wurde. Dennoch erlaubte die angewandte Form des Interviews auch eine freie Beantwortung der Fragen durch den bzw. die interviewten Personen. Dadurch konnten teilweise Sachverhalte und Zusammenhänge aufgedeckt werden, welche im Vorfeld bei der Erstellung des Fragebogens nicht berücksichtigt wurden. Im folgenden Unterkapitel werden diese Ergebnisse der Interviews detailliert erläutert.

4 Ergebnisse der Studie

In diesem Kapitel werden die empirischen Umfrage- und Interview-Ergebnisse nach den zuvor vorgestellten organisatorischen und technischen Dimensionen von Containervirtualisierung strukturiert und analysiert. Nachdem alle Interviews durchgeführt und transkribiert wurden, wurden die Ergebnisse kodiert und in einer Datenbank gesammelt. Aus dieser Tabelle wurden insgesamt sechs Themenblöcke identifiziert, welche in den folgenden Unterkapiteln näher beschrieben werden.

4.1 Status Quo

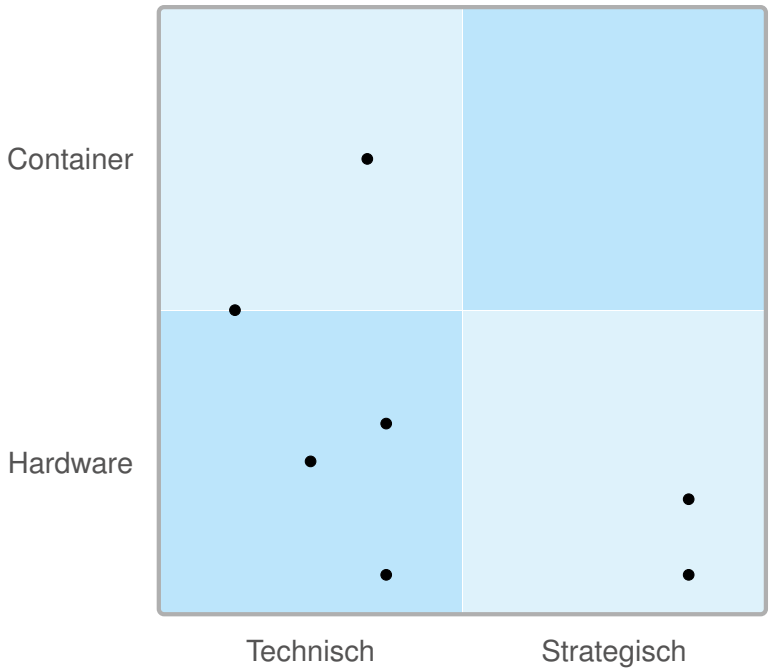
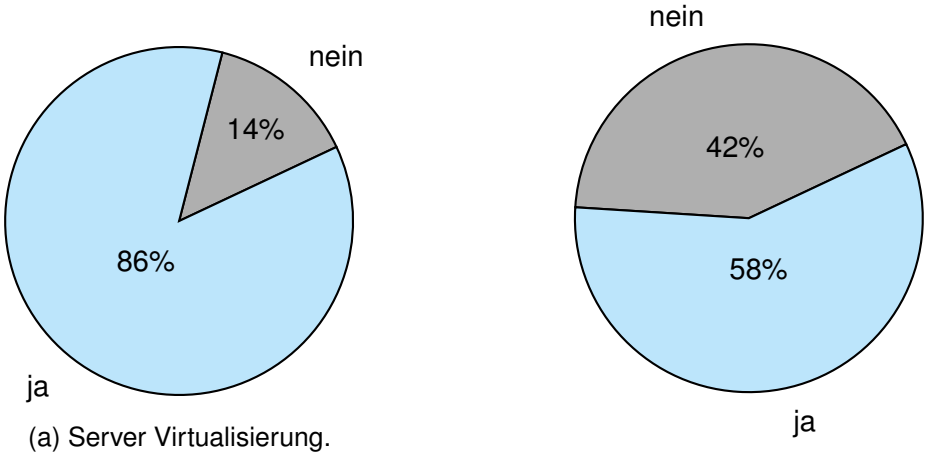


Abbildung 8: Ausrichtung der Interviewpartner.

Nachdem generelle Informationen zu den Unternehmen gesammelt wurden (s. Kapitel 3.1), wurde die Positionierung der Teilnehmer ermittelt. Aus der Abbildung 8 lässt sich ablesen, dass der Großteil der Teilnehmer aus technischer Sicht in das Thema Virtualisierung involviert ist. Dabei bezeichnet sich aktuell nur ein Teilnehmer schon als Experte im Bereich der Container-basierten Virtualisierung. Ein Teilnehmer ist schon bimodal aufgestellt, d.h. er ist sowohl im besser vorhersehbaren und verstandenem Feld der Hardware-Virtualisierung fest verankert, experimentiert aber schon im Bereich der Container-basierten Virtualisierung. Zwei Teilnehmer sind aus strategischer Sicht mit der Thematik betraut und ein Umfrageteilnehmer verfolgt bereits eine auf mehrere Jahre ausgelegte Strategie zur Einführung von Container-basierter Virtualisierung.



(a) Server Virtualisierung.

(b) Client Virtualisierung.

Abbildung 9: Virtualisierungsquote.

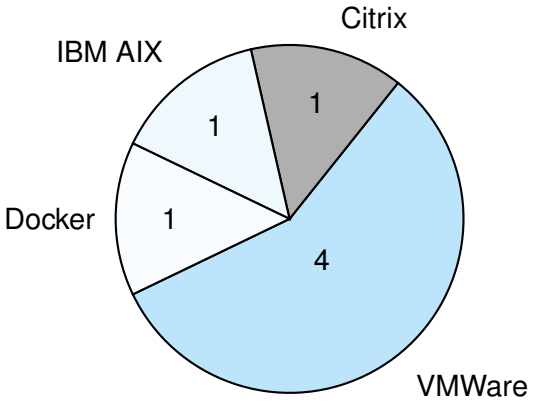


Abbildung 10: Eingesetze Virtualisierungs-Technologien.

Bezüglich der Verbreitung von Virtualisierung mit Blick auf den Einsatzort im Unternehmen, ergeben sich Virtualisierungsquoten wie in Abbildung 9 dargestellt: 85% der Unternehmen setzen bereits auf Virtualisierung im Umfeld ihrer Server. 68% der Unternehmen virtualisieren die Benutzer(Client)-Umgebungen.

Abbildung 10 gibt einen Überblick über die primäre Virtualisierungs-Technologie der befragten Unternehmen. Bei den befragten Firmen wird größtenteils VMWare zur Virtualisierung eingesetzt ($\sim 57\%$). Citrix und IBM AIX sind bei jeweils einem Teilnehmer dominant; ein Teilnehmer setzt auf Docker. Dabei bleibt festzuhalten, dass das Unternehmen, welches aktuell IBM AIX einsetzt, schon einen Transformationsprozess gestartet hat, der IBM AIX durch VM Ware ersetzen wird.

Bezüglich der Transformation besitzt nur eines der befragten Unternehmen einen Transformationsplan, der sich auch mit Container-basierter Virtualisierung beschäftigt.

Bezüglich des technischen Wissensstandes lässt sich eine starke Polarisierung erken-

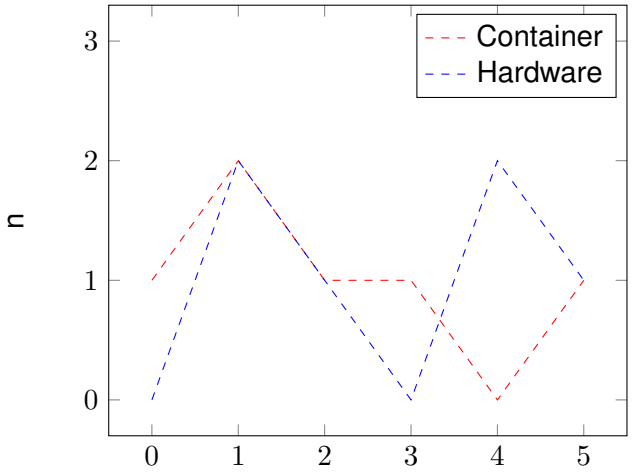


Abbildung 11: Technisches Wissen bezüglich Hardware- und Container-basierter Virtualisierung auf der Skala von 0 (wenig Wissen) bis 5 (ausführliches Wissen).

nen, die in Abbildung 11 dargestellt ist. Trivialerweise sind die Strategie-Verantwortlichen nicht mit jedem neuen Trend direkt im Detail vertraut, da technisches Detailwissen für ihr Alltagsgeschäft von geringer Relevanz ist. Aufgrund der relativen Neuheit der Thematik gibt es eine große Lücke bezüglich Wissen zu Container-basierter Virtualisierung.

Alle der angesprochenen Unternehmen besitzen eine IT-Abteilung, entweder in-house oder ausgelagert in eine Tochtergesellschaft. Das Thema Infrastruktur ist dabei für Firmen, die primär IT-Dienstleistungen im Auftrag erledigen, weniger relevant als für Unternehmen, die kritische Infrastruktur in-house betreiben. Dementsprechend sind auftragnehmende Unternehmen tendenziell offensiver in Bezug auf den Einsatz neuer Technologien.

4.2 Wahrgenommene Vor- und Nachteile von Virtualisierung

Von einem besonderen Interesse ist, welche Herausforderungen, Probleme und Chancen die IT-Verantwortlichen durch den Einsatz von Software-Virtualisierung – und im Speziellen von Container-basierter Virtualisierung – sehen. Alle befragten Unternehmen setzen Virtualisierung entweder Hardware- oder Container-basiert ein. Für einige ist dies schlicht eine alltägliche Notwendigkeit, für andere ist es eine Missions-kritische Aufgabe, die auch aus Compliance Gründen auf keinen Fall an andere Firmen abgetreten werden kann. ??? TODO: Erläuterung

Als Vorteile werden Ausfallsicherheit, Portabilität und Skalierbarkeit genannt. Wie in Abbildung 13 erkennbar, ist Ausfallsicherheit der mit Abstand wichtigste Vorteil, den sich die Teilnehmer von Virtualisierung erhoffen. Portabilität ist für die meisten Teilnehmer relevant, da die Fähigkeit logische Systeme zwischen physischen Systemen zu bewegen die

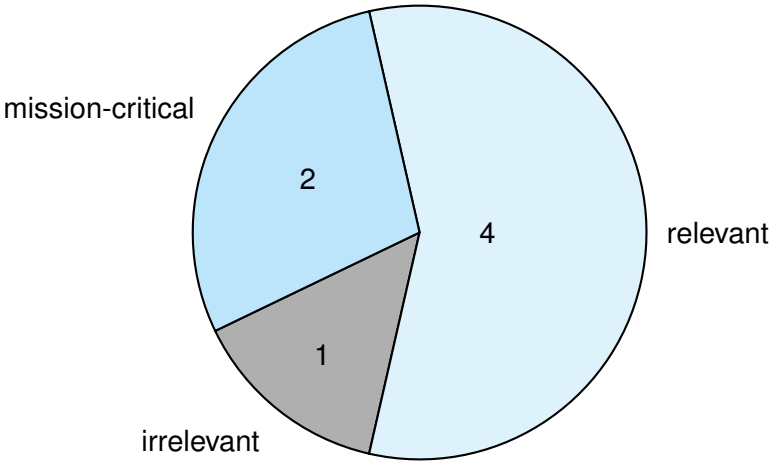


Abbildung 12: Wichtigkeit von Virtualisierung in den befragten Unternehmen.

Wartung von Hardware einfacher gestaltet. Außerdem ist es eine Voraussetzung um Ausfallsicherheit zu erlangen. Skalierbarkeit hingegen spielt für die meisten Projektteilnehmer eine tendenziell untergeordnete Rolle. Dies lässt sich damit erklären, dass die meisten Teilnehmer schon einen sehr guten Überblick über ihre Auslastungsprofile besitzen; d.h. sie werden selten von einem Kundenansturm überrascht. Da die Softwaresysteme bei einigen Unternehmen Missions-kritisch sind, sind die Kapazitäten von vornherein oft so konzipiert, dass genügend Last-Spielraum vorhanden ist. Natürlich ist in Anbetracht von Wachstumszielen auch hier in bestimmten Zeiträumen ein Ausbau dieser Kapazitäten erforderlich. Abschließend ist zu sagen, dass Unternehmen, die der Virtualisierung für ihren Geschäftserfolg eine untergeordnete Rolle zuordnen, auch die genannten Themen Ausfallsicherheit, Portabilität und Skalierbarekeit als tendenziell unwichtiger einstufen (2/7).

Bezüglich der Probleme, die Teilnehmer mit Virtualisierung haben, lässt sich Folgendes zusammenfassen: In kleinen und mittelständischen Unternehmen sind die Entwickler ausschließlich auf Software-Projekte aufgeteilt. Es gibt häufig keine Mitarbeiter, die dediziert für die Thematik der Container-basierten Virtualisierung verantwortlich sind. Stattdessen verbringen Mitarbeiter Zeit in allen Phasen eines Software-Projekts: bei der Anfor-

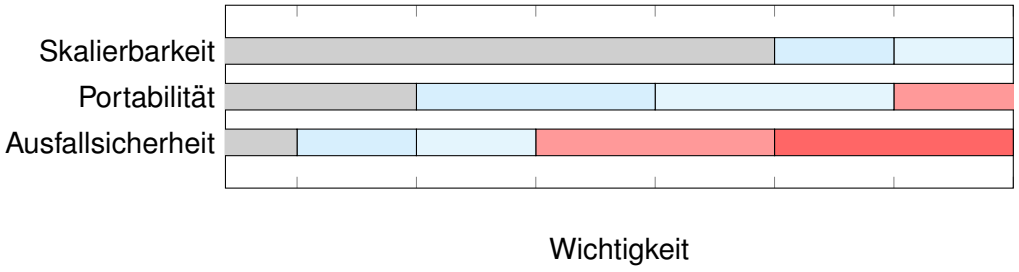


Abbildung 13: Einschätzung der Vorteile von Virtualisierung.
Unwichtig (grau), über niedrige (blau) bis hohe (rot) Wichtigkeit.

derungsanalyse, Entwicklung, Deployment und Wartung. Für komplexe Fragestellungen im Bereich Virtualisierung wird Support in Form externer Berater angefordert. Mit fortschreitender Zeit betreut ein Mitarbeiter entsprechend viele Systeme in der Wartung. Hier wird von allen das konstante Monitoring und, sobald ein Systemupgrade aufgrund von unzureichender Systemressourcen ansteht, der Upgrade-Prozess als belastend empfunden — gerade da dies zu Zeiten auftreten kann, in der sich andere Projekte gerade in einer heißen Phase befinden.

Skalierbarkeit. Skalierbarkeit lässt sich generell durch vertikale oder horizontale Skalierung realisieren. Vertikale Skalierung bezieht sich dabei auf die Erhöhung der Performance bestehender Systemen, z.B. durch ein Upgrade der Prozessoren oder des Arbeitsspeichers. Horizontale Skalierung erreicht höhere Performance durch das Hinzufügen weiterer Maschinen, sodass mehr Aufgaben zugleich abgearbeitet werden können. Gerade die horizontale Skalierung wird zur Zeit in der IT-Welt breit diskutiert. Im Startup-Bereich, wo teils rein digitale Produkte mit kleinen Teams auch auf einen globalen Zielmarkt ausgerichtet werden, wird die Möglichkeit der horizontalen Skalierung hoch geschätzt, da im Falle eines plötzlichen Popularitätssprunges des Produktes innerhalb kurzer Zeit die Kapazität vervielfacht werden kann. Wider der Erwartung ergab die Umfrage, dass sich alle KMU wenig für diesen Nutzen der horizontalen Skalierung interessieren. Die wahrgenommenen Potentiale im Bereich Skalierbarkeit beschränken sich auf Einsparungen, die erreicht werden können, wenn Server-Ressourcen nur zu Lastzeiten angemietet werden würden, statt sie durchgängig zu betreiben.

Andererseits sehen die Befragten durchaus Gefahren, die von dem “Technologie-Zoo” ausgehen, d.h. der Vielzahl von Tools, die benötigt werden, um horizontale Skalierbarkeit umzusetzen. Eine große Anzahl von extern entwickelten Tools müsste evaluiert, die Weiterentwicklung überwacht und die Integration zwischen den Tools sichergestellt werden. Auch aus Personalsicht würden weitere Kosten anfallen, da direkt mehrere Experten für diese einzelnen Teilbereiche eingestellt werden müssten.

Portabilität. Bezüglich der Portabilität, d.h. des Grad der Transferierbarkeit einer Anwendungen zwischen verschiedenen Systemen, sehen die Befragten noch keinen Handlungsbedarf. Der Vorteil wird als “nicht ganz so präsent” oder “notwendig” angesehen. Allerdings werden die Lock-in Effekte der aktuellen Virtualisierungs-Lösungen als unkritisch angesehen, sodass potentiell eine Veränderung durchaus machbar wäre.

4.3 Erwartungen an Container-basierte Virtualisierung

Simplizität. Die meisten Befragten interessieren sich für Container-basierte Virtualisierung aus anderen Gründen als der genannten horizontalen Skalierbarkeit und Portabilität. Gerade für die kleineren Unternehmen innerhalb der befragten KMU liegt die Attraktivität dieser Form der Virtualisierung in ihrer Simplizität für einfache Fälle. Wohingegen traditionelle Virtualisierung direkt viele Probleme auf einmal zu lösen versucht, lassen sich durch die clevere Auswahl und Beschränkung auf nur wenige Tools aus dem "Technologie-Zoo" einfache Anforderungen potentiell einfacher umsetzen. Die Einarbeitungszeit in Konzepte rund um Container wird als einfacher eingeschätzt, da für einige Szenarien das Wissen, welches in Kapitel 2.1.2 aufgebaut wurde, bereits ausreicht, um eine Anwendung in einfachen Szenerien zu virtualisieren. Die Wissensbarriere ist entsprechend niedrig, es werden lediglich Kenntnisse über die Kommandozeilen-Befehle vorausgesetzt, die nicht komplizierter sind als jene, die von der Anwendung schon normalerweise vorausgesetzt werden bzw. in deren Dokumentation erläutert sind. Auch die Bereitstellung der eigenen Hardware ist simpel: Aktuelle Versionen der weit weitestverbreiteten Linux Distributionen werden mit allen notwendigen Pakete vorinstalliert ausgeliefert. Am Ende erhoffen sich zwei Befragte, dass diese niedrigere Barriere den Wissenstransfer vereinfachen wird.

Vorteile für Lastspitzen-Szenarios. Abseits dieses Aspekts interessieren sich KMU für Kostenoptimierungen bei Lastspitzen, d.h. Einsparungen, die erreicht werden können, wenn Server-Ressourcen nur zu Lastzeiten angemietet werden, statt sie durchgängig zu betreiben. Bezüglich dieser Verheißung stellten Analysten von Gartner zuletzt jedoch heraus, dass "Viele Unternehmen [...] entsetzt (sind), wenn sie ihre ersten Cloud-Rechnungen bekommen, da diese weit höher sind als veranschlagt" [6]. Entsprechend sollte eine Einführung von Container-basierter Virtualisierung nur zum Erreichen dieses Ziels sehr kritisch betrachtet werden. Für ein Unternehmen, dass die Lastzeiten sehr gut abschätzen kann, z.B. weil es morgens beim Login der Mitarbeiter immer zu einer Lastspitze kommt, bietet der Ansatz durchaus Potential.

Schnelles Feedback. Ein weiterer genannter Vorteil betrifft eine erwartete Beschleunigung bei der Anwendungsauslieferung, die es ermöglicht, während der Entwicklung schnelleres Feedback zu Änderungen zu bekommen. Hier erhoffen sich die Befragten eine höhere Personaleffizienz und eine damit verbundene Optimierung im Hinblick auf die Personalkosten.

Entwicklung. Aus Sicht der Entwickler gibt es verschiedene Vorteile. Zum einen bietet ein Container-basierter "Azubibaukasten" die Möglichkeit zum Ausprobieren von neuen

Technologien, ohne dass durch Fehler laufende Systeme beeinträchtigt werden können. Bezüglich Integrationstests wird eine bessere Parallelisierung der Tests und Ausweitung von Konfigurationskonstellationstests erwartet.

4.4 Vorgehensmodell bei Service-Virtualisierung

In Abbildung 14 sind die drei Reifegrad-Levels der Service-Virtualisierung in KMUs dargestellt, welche aus den Interview-Ergebnissen abgeleitet wurden. Im Reifegrade Level 1 (RGL-1) wird ein Prototyp als Entwicklungsprojekt verwendet, in welchem man den Prototypen mit Hilfe von Container-basierter Software-Virtualisierung erstellt. In diesem Projekt wird Erfahrung mit den Virtualisierungstechniken gesammelt, die in zukünftige Entscheidungen einfließen kann. Diese Entscheidungen können mehrere Dimensionen haben. Auf der einen Seite kann die Erfahrung in der Entwicklung eines Prototypen dazu dienen, eine generelle Aussage zur Einführung von Virtualisierungs-Software im Unternehmen zu treffen. Ein Ergebnis könnte beispielsweise sein, dass die Einarbeitung der Mitarbeiter in neue Virtualisierungstechniken zeitaufwendiger ist als erhofft. Eine Konsequenz daraus könnte sein, dass eine Virtualisierungsstrategie im Unternehmen nicht weiter verfolgt wird oder dass die Mitarbeiter gezielter in der neuen Technologie geschult werden.

In Reifegrade Level 2 (RGL-2) wird ein neues Projekt mit Hilfe von Container-basierter Software-Virtualisierung erstellt. Im Gegensatz zu RGL-1 wird bei RGL-2 ein gesamtes Software-Projekt mit Hilfe von Virtualisierungstechnik entwickelt.

Das letzte Reifegrade Level (RGL-3) beinhaltet die Konvertierung von bestehenden Projekten auf Virtualisierungstechniken. Das konvertierte Software-Projekt sollte dabei die gleichen Services und Funktionalitäten bereithalten wie das bisherige Software-System.

Abbildung 15 zeigt die Verteilung der Projektteilnehmer in die drei Reifegrade Levels. Es ist zu erkennen, dass 4 der 7 Projektteilnehmer bisher einen Prototypen entwickelt haben,

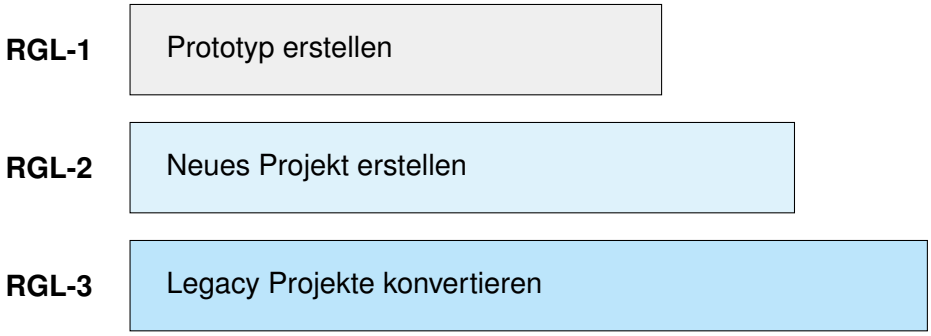


Abbildung 14: Reifegrade Level (RGL) der Service-Virtualisierung in KMUs.

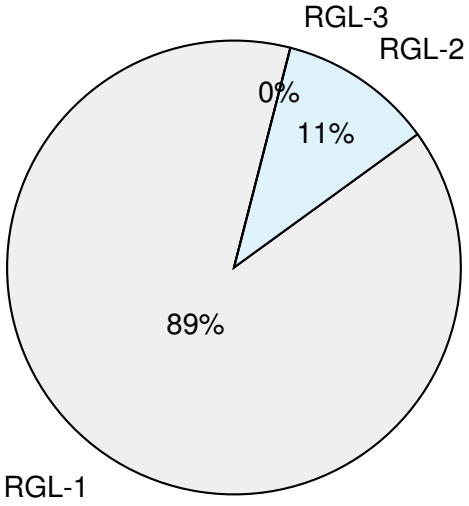


Abbildung 15: Verteilung der Reifegrade Level (RGL) der Projektteilnehmer.

in welchem Sie Container-basierte Virtualisierungstechniken angewendet haben. Zudem haben 2 der 7 Projektteilnehmer ein komplett neues Projekt auf Basis von Container-basierter Virtualisierung entwickelt. Zu erkennen ist auch, dass bisher kein einziger Projekteilnehmer ein bestehendes IT-System auf Container-basierter Virtualisierung konvertiert hat. Als Gründe dafür wurden genannt, dass bestehende IT-Systeme, die langjährig im Einsatz sind, oftmals sehr zuverlässig laufen und die Gefahr gesehen wird, bei einer Konvertierung auf Container-basierte Virtualisierung neue Fehler in das System einzubauen.

Das zentrales Ergebnis der Interviews zu diesem Themenblock ist, dass bisher keiner der Projektteilnehmer ein dokumentiertes Vorgehen im Unternehmen hat, mit dem Software-Artefakte virtualisiert werden. Daraus lässt sich ableiten, dass bisher die Notwendigkeit eines solchen Dokumentes nicht hoch genug war, um ein solches zu erstellen. Dennoch haben alle Projektteilnehmer zugestimmt, dass ein solches Vorgehen sinnvoll wäre. Im Rahmen dieses Projektes wurde ein solch abstraktes Vorgehensmodell entwickelt. In Abbildung 16 ist das abstraktes Vorgehensmodell zur Einführung von Virtualisierungen in einem KMU dargestellt. In einem ersten Schritt werden IT-verantwortliche Personen innerhalb des Unternehmens identifiziert, um einen klaren Ansprechpartner für die IT-Transformation zu ermitteln. Ein weiteres Ziel dabei ist, dass es ein Commitment der IT-Verantwortlichen gibt, an dem Virtualisierungsprozess teilzunehmen. Sofern es im Unternehmen keinen direkten Ansprechpartner dafür gibt, sollte eine Rolle dafür geschaffen werden.

In einem nächsten Schritt werden die relevanten IT-Services mit Berücksichtigung des Virtualisierungsvorteils identifiziert. Es gibt dabei drei Virtualisierungsvorteile, auf die geprüft werden sollte: *Abstraktion*, *Skalierbarkeit* und *Ausfallsicherheit* (s. Unterkapitel 4.2).

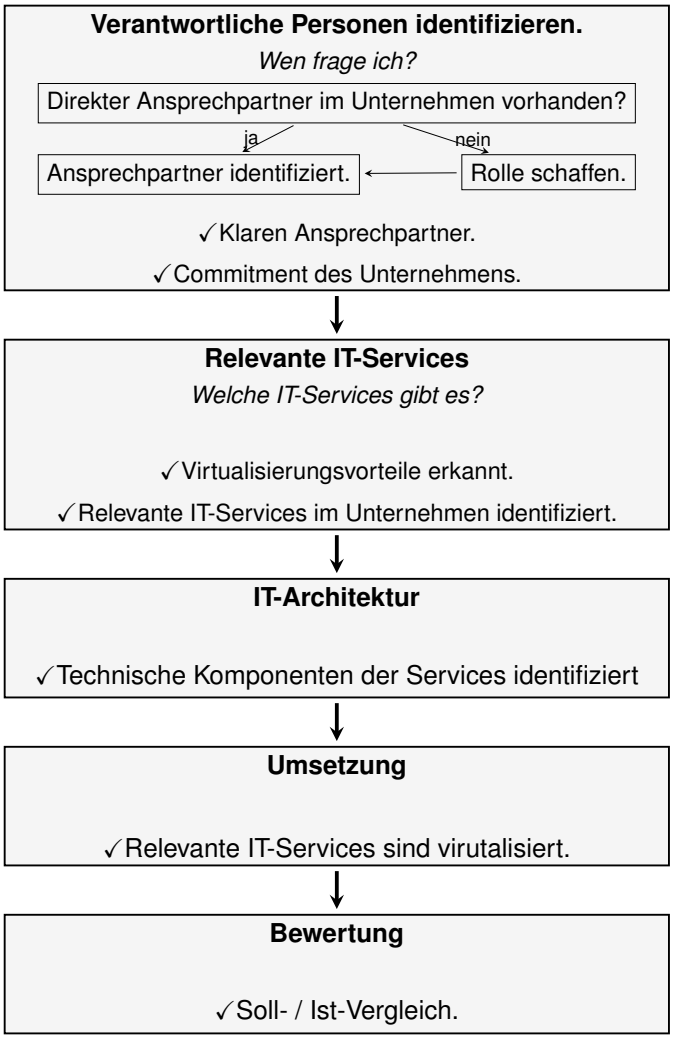


Abbildung 16: Vorgehensmodell zur Einführung von Virtualisierungen.

In einem dritten Schritt wird die IT-Architektur der identifizierten IT-Services genauer beleuchtet. Ein Ziel dabei ist es herauszufinden, welche technischen Komponenten von den zu virtualisierenden Services betroffen sind.

Anschließend kann mit Unterstützung geeigneter Virtualisierungstools der Service (teil-) virtualisiert werden. Für den Virtualisierungsvorteil *Abstraktion* bietet sich beispielsweise *Docker* an, für *Ausfallsicherheit* Frameworks wie *Ceph* und *Docker-Swarm* und für *Skalierbarkeit* unter anderem *OpenStack*.

Zum Schluss soll zur Bewertung des tatsächlichen Virtualisierungsvorteils ein Soll-/Ist-Vergleich vorgenommen werden.

5 Handlungsempfehlung zur Einführung und Einordnung der Technologien

Der im folgenden vorgestellte Prozess wird von der Cloud Native Container Foundation (CNCF) zur Einführung von Container-basierten Infrastrukturen empfohlen. Hierbei sind alle Schritte nach Schritt 3 als **optional** zu betrachten, wobei in jedem der optionalen Schritte die Notwendigkeit der Durchführung vorab zu prüfen ist.

1. CONTAINERIZATION
   - Meist mit **Docker** Container realisiert.
   - Eine Anwendung beliebiger Größe und mit beliebiger Anzahl von Abhängigkeiten lässt sich in einem Container zusammenfassen.
   - Im Laufe der Zeit sollten Sie danach streben, geeignete Anwendungen aufzuteilen und zukünftig Anwendungen direkt container-basiert zu konzipieren.
2. CI/CD (Continuos Integration and Delivery)
   - Einrichtung kontinuierliche Integration/kontinuierliche Auslieferung (CI/CD), so dass Änderungen an Ihrem Quellcode automatisch zu einem neuen Container führen. D.h. der Container wird für den Quellcode neu gebaut, getestet und auf die *staging*- und ggf. sogar *produktiv*-Server ausgeliefert.
3. ORCHESTRATION
   - **Kubernetes** ist die marktführende Orchestrierungslösung.
   - Sie sollten eine zertifizierte Kubernetes-Distribution wählen, eine gehostete Plattform evaluieren oder eine Installation in die eigene Systemlandschaft in Betracht ziehen.
4. OBSERVABILITY & ANALYSIS
   - Wählen Sie eine Lösung zum Monitoring, Logging und Tracing.
   - Ziehen Sie dabei in Betracht, die CNCF-Projekte **Prometheus** für die Überwachung, **Fluentd** für Logging und **Jaeger** für Tracing einzusetzen.
5. SERVICE MESH AND DISCOVERY
   - `CoreDNS` ist ein schnelles und flexibles Werkzeug, das nützlich für die Serviceentdeckung ist.
   - Sowohl **Envoy**, als auch **Linkerd** erlauben das Erstellen von *service mesh architectures* . . .
   - . . . die Health-Checks, Routing, und Lastverteilung ermöglichen.

6. NETWORKING
   - Um eine flexiblere Vernetzung zu ermöglichen, verwenden Sie CNI-konforme Netzwerkprojekte wie **Calico**, **Flannel** oder **Weave Net**.
7. DISTRIBUTED DATABASE
   - Wenn Sie mehr Ausfallsicherheit und Skalierbarkeit benötigen, als man mit einer einzigen Datenbank erreichen kann, ist **Vitess** eine gute Option für den Betrieb von MySQL, was Skalierung durch sharding (Datenbankpartitionierungen) ermöglicht.
8. MESSAGING
   - Wenn Sie mehr Leistung benötigen, als ein RESTful Webservice (mit JSON) bieten kann, erwägen Sie die Verwendung von **gRPC**. **NATS** ist eine nachrichtenorientierte Middleware.
9. CONTAINER RUNTIME
   - Sie können alternative Container-Laufzeitumgebungen verwenden. Die gängigsten Lösungen, die alle OCI-konform sind, sind **containerd**, **rkt** und **CRI-O**.
10. SOFTWARE DISTRIBUTION
    - Wenn Sie eine sichere Softwareverteilung benötigen, evaluieren Sie **Notar**, eine Implementierung des *The Update-Framework*.

5.1 Sicherheitsüberlegungen

Generell muss festgehalten werden, dass Container im Hinblick auf Sicherheit für einen Angreifer keine unüberwindbare Hürde darstellen. Zwar wird eine Zugriffssteuerung für die (geteilten) Systemressourcen geboten, diese ist aber keine großes Hindernis. Wie besprochen kapseln sowohl virtuelle Maschinen, als auch Container Anwendungen. Im Gegensatz zum Container, der Grenzen für Anwendungen definiert, bilden virtuelle Maschine auch Grenzen zu ihren Ressourcen[1].

- Container sind gut darin, Anwendungen zu *kapseln*.
  - Vorteile: Skalierbarkeit, Zugriffsregelung auf Ressourcen.
  - Vorsicht: **Sicherheit** nicht Fokus von Container.

- **Gefährlichkeit** von Software wird meist nicht binär bewertet, sondern existiert auf ein Spektrum.

[1] Die sich ggf. auf Hardware Level wieder umgehen lassen (s. aktuell `https://meltdownattack.com/`)

Ist also der Einsatz einer bestimmten Software geplant, so gilt es vor der Entscheidung zu einer speziellen Virtualisierungslösung festzustellen, um welche Art von Software es sich handelt. Handelt es sich zum Beispiel um Malware, also schädlicher Software, die z.b. zu einem Erprobungszwecke eingesetzt wird, so ist die logische Schlussfolgerung jene Technologie auszuwählen, welche das höchste Maß an Isolierung bietet. Handelt es sich um eine Anwendung Dritter, so sollte die Entscheidung davon abhängig sein, ob es möglich ist einen eigenen Sicherheits-Audit für die Software durchzuführen. Ist dies nicht der Fall, so ist auch diese Software ein Kandidat für erhöhten Isolierungsbedarf. In allen anderen Fällen, sollte der Sicherheits-bezogene Isolierungsbedarf gering sein; eine Ausschöpfung der Vorteile Container-basierter Virtualisierung steht ergo nach diesem Gesichtspunkt nichts im Wege.

6 Einfluss von Docker auf die Software-Entwicklung

Container-basierte Virtualisierung kann nicht nur bei der Bereitstellung von produktionsreifen IT-Services unterstützten, sondern kann auch in der Software-Entwicklungsphase von einem IT-Entwickler genutzt werden, um bessere Software zu entwickeln.

Ein Entwickler kann Docker nutzen, um auf dem Entwicklungs-Rechner eine Umgebung zu erstellen, in der Anwendungen in Containern lokal entwickelt und getestet werden.

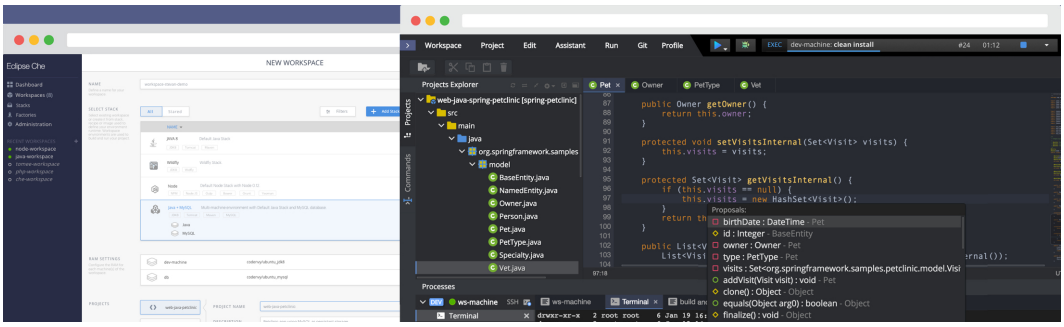


Abbildung 17: Ein Screenshot von der Eclipse Che IDE.

DevOps ist gerade für kleinere Entwickler-Teams nützlich, wenn quasi “jeder Entwickler alles kann”. Damit gibt es auch kein spezielles Inselwissen, falls mal ein Mitarbeiter das Unternehmen verlässt.

Ein Beispiel für eine Integrated Development Environment (IDE)—die in der Cloud läuft—ist *Eclipse Che*[2].

Wie in Abbildung 6 dargestellt, sind die Unternehmen der Projektteilnehmer in unterschiedlichen Branchen tätig. Daraus resultiert, dass die Unternehmen auch in einem unterschiedlichen Umfang eine IT-Abteilung betreiben (s. auch Abbildung 5). Daher ist das Thema *Software Entwicklung mit Docker* auf den ersten Blick nicht für jeden der Projektteilnehmer unmittelbar von großer Relevanz.

7 Diskussion

Im Rahmen dieses Projektes wurden zwei wesentliche Gegenstände untersucht. Zum einen wurde untersucht, in welchem Umfang Container-basierte Virtualisierung in lokalen Unternehmen im Raum Münster momentan genutzt wird. Zum anderen wurde untersucht, welche Chancen und Risiken IT-Verantwortliche dieser Unternehmen in der Nutzung dieser Virtualisierungstechnik sehen.

[2] `https://www.eclipse.org/che/`

Einen der zentralen Schlüsse, die wir aus den Gesprächen mit den verschiedenen Projektpartnern ziehen können, ist:

Container-basierte Virtualisierung sollte bei IT-Entscheidern Berücksichtigung finden.

Diese IT-Entscheidungen können in einem Unternehmen sowohl von CTOs und IT-Architekten getroffen, als auch von den Entwicklern mit beeinflusst werden.

Container-Virtualisierung ist ein wichtiges Thema für IT-Entscheider. Die zentralen Vorteile von Container-Virtualisierung sind die gesteigerte Ausfallsicherheit (oft in Kombinatition mit einer Microservice-Architektur), die gute Portabilität und die Skalierbarkeit. Nachdem die IT-Industrie über Jahre hinweg Container in Form von virtuellen Maschinen genutzt hat, um eine Abstraktionsebene zu den physikalischen Plattformen zu schaffen, ermöglichen es Technologien wie Docker nun, Container zwischen Plattformen zu verschieben. Genauer gesagt: Linux-Applikationen und Workloads zwischen mehreren Clouds zu bewegen, um damit die Portabilität zu verbessern. Im ersten Moment erscheint Docker damit als ein typisches Tool für Entwickler. Aus dem Blickwinkel eines CIOs handelt es sich allerdings klar um ein strategisches Werkzeug, um die angestrebte hohe Verfügbarkeit der Unternehmens-IT zu erreichen.

Außerdem interessieren sich KMUs für Kostenoptimierungen bei Lastspitzen, d.h. Einsparungen, die erreicht werden können, wenn Server-Ressourcen nur zu Lastzeiten angemietet werden, statt sie durchgängig zu betreiben. Bezüglich dieser Verheißung stellten Analysten von Gartner zuletzt jedoch heraus, dass viele Unternehmen hier mit unerwartet hohen Kosten konfrontiert wurden. Entsprechend sollte eine Einführung von Container-basierter Virtualisierung rein zum Erreichen dieses Ziels mit einer gewissen Skepsis betrachtet werden.

Von der Portabilität bei Container-Virtualisierung können beispielsweise auch Cloud-Marktplätze und Cloud-Brokerage-Services wie Deutsche Börse Cloud Exchange (DBCE) profitieren und ihren Kunden die Möglichkeit bieten, Applikationen zwischen den unterschiedlichen Anbietern je nach Bedarf und Kosten zu verschieben.

Literatur

[1] Mark Aiken, Manuel Fahndrich, Chris Hawblitzel, Galen Hunt, and Jim Larus. Deconstructing process isolation. In *ACM SIGPLAN Workshop on Memory Systems Performance and Correctness*, pages 1–10, San Jose, CA, October 2006. ACM.

[2] David Bernstein. Containers and cloud: From lxc to docker to kubernetes. 1(3):81–84.

[3] Uwe Flick, Ernst von Kardorff, and Ines Steinke. Qualitative forschung. *Ein Handbuch*, 6:14, 2008.

[4] Fraunhofer-Gesellschaft. Public, Private und Hybrid Cloud? `https://www.cloud.fraunhofer.de/de/faq/publicprivatehybrid.html`. Accessed Jun. 2018.

[5] Tal Garfinkel, Ben Pfaff, Jim Chow, Mendel Rosenblum, and Dan Boneh. Terra: A Virtual Machine-based Platform for Trusted Computing. In *ACM SIGOPS Operating Systems Review*, volume 37, pages 193–206. ACM.

[6] Gartner, Inc. Umfrage: Cloud kann teuer werden. `https://www.gartner.com/events/emea/infrastructure-operations-management#section_exhibitors`. Accessed Jun. 2018.

[7] GitHub, Inc. Repositories by stars.

[8] Andrea Kurz, Constanze Stockhammer, Susanne Fuchs, and Dieter Meinhard. Das problemzentrierte interview. In *Qualitative Marktforschung*, pages 463–475. Springer, 2007.

[9] Dirk Merkel. Docker: Lightweight linux containers for consistent development and deployment. *Linux J.*, 2014(239), March 2014.

[10] The Linux Foundation. Kubernetes - Production-Grade Container Orchestration. `https://kubernetes.io/`. Accessed Jun. 2018.

[11] Andreas Witzel. *Verfahren der qualitativen Sozialforschung: Überblick und Alternativen*. Campus-Verlag, 1982.