



Hochschule
Augsburg University of
Applied Sciences

Bachelorarbeit

Fakultät für
Informatik

Studienrichtung
Informatik

Bernd Fecht

**Integration von GPU-fähigen Docker-
Containern in eine virtuelle Umgebung
unter Einsatz von Citrix und NVIDIAs
vGPU**

Prüfer: Prof. Dr. Thorsten Schöler
Abgabe der Arbeit am: 13.12.2016

Hochschule für angewandte
Wissenschaften Augsburg
University of Applied Sciences
An der Hochschule 1
D-86161 Augsburg
Telefon +49 821 55 86-0
Fax +49 821 55 86-3222
www.hs-augsburg.de
info@hs-augsburg.de

Fakultät für Informatik
Telefon: +49 821 5586-3450
Fax: +49 821 5586-3499

Verfasser der Diplomarbeit:
Bernd Fecht
Rosenheimer Allee 1a
86399 Bobingen
b.fecht@gmx.de

Erstellungserklärung

Hiermit versichere ich, die eingereichte Abschlussarbeit selbstständig verfasst und keine andere als die von mir angegebenen Quellen und Hilfsmittel benutzt zu haben. Wörtlich oder inhaltlich verwendete Quellen wurden entsprechend den anerkannten Regeln wissenschaftlichen Arbeitens zitiert. Ich erkläre weiterhin, dass die vorliegende Arbeit noch nicht anderweitig als Abschlussarbeit eingereicht wurde.

Das Merkblatt zum Täuschungsverbot im Prüfungsverfahren der Hochschule Augsburg habe ich gelesen und zur Kenntnis genommen. Ich versichere, dass die von mir abgegebene Arbeit keinerlei Plagiate, Texte oder Bilder umfasst, die durch von mir beauftragte Dritte erstellt wurden.

Ort, Datum

Unterschrift des/der Studierenden

Inhaltsverzeichnis

| | |
|---|----|
| Abbildungsverzeichnis | 3 |
| 1. Deutsche Kurzfassung | 5 |
| 2. Englische Kurzfassung..... | 5 |
| 3. Einleitung | 6 |
| 3.1 Motivation | 6 |
| 3.2 Ausgangssituation | 7 |
| 3.3 Ziel der Arbeit | 7 |
| 3.4 Aufbau der Arbeit..... | 8 |
| 4. Stand der Technik..... | 9 |
| 4.1 Einführung in die Virtualisierung | 9 |
| 4.2 Grundlagen der Virtualisierung | 9 |
| 4.2.1 Plattformvirtualisierung | 10 |
| 4.2.2 Anwendungsvirtualisierung | 16 |
| 4.2.3 Desktopvirtualisierung | 17 |
| 4.3 Docker | 17 |
| 4.3.1 Docker Komponenten..... | 18 |
| 4.4 GPU-Virtualisierung mit NVIDIA..... | 23 |
| 4.4.1 NVIDIA GRID..... | 23 |
| 4.4.2 GRID vGPU Architektur | 24 |
| 4.4.3 GRID Profile | 24 |
| 5. Aufgabenstellung | 26 |
| 6. Beschreibung der Arbeit..... | 27 |
| 6.1 Aufsetzen der Testumgebung | 27 |
| 6.1.1 Anforderung an die Testumgebung..... | 27 |
| 6.1.2 Netzwerkarchitektur | 27 |
| 6.1.3 Komponenten | 32 |
| 6.1.4 Umsetzung | 42 |

| | | |
|-------|--|----|
| 6.1.5 | Funktionsweise..... | 45 |
| 6.1.6 | Testen der Xen-Umgebung..... | 46 |
| 6.2 | Bereitstellung von Containern auf dem XenServer..... | 48 |
| 6.3 | Docker-Container mit GPU-Unterstützung | 50 |
| 6.3.1 | NVIDIA Docker | 50 |
| 6.3.2 | Problemanalyse..... | 50 |
| 6.3.3 | Validieren der Lösung für NVIDIA GRID | 52 |
| 6.4 | Integration von Docker-Containern auf dem XenServer..... | 56 |
| 6.4.1 | Vorbereiten des XenServers..... | 56 |
| 6.4.2 | Einrichten des Container-Managements | 56 |
| 6.4.3 | Funktionsweise des Container-Managements | 58 |
| 6.4.4 | Fazit | 59 |
| 6.5 | Integration der Docker-Container in den StoreFront..... | 60 |
| 6.5.1 | Anforderung an die Integration | 60 |
| 6.5.2 | Integration der Container per SSH..... | 60 |
| 6.5.3 | Integration der Container per VNC..... | 72 |
| 7. | Ergebnisse..... | 76 |
| 8. | Ausblick | 77 |
| | Literaturverzeichnis..... | 78 |
| | Glossar | 82 |

Abbildungsverzeichnis

| | |
|--|----|
| Abbildung 1: Typ-1-Hypervisor..... | 11 |
| Abbildung 2: Typ-2-Hypervisor..... | 12 |
| Abbildung 3: Ringmodell für x86-Prozessoren (MEINEL 2010) | 13 |
| Abbildung 4: XenServer Architektur | 16 |
| Abbildung 5: Docker Architektur (DOCKER INC. 2016c) | 18 |
| Abbildung 6: Docker Client-Server Architektur (DOCKER INC. 2016a) | 19 |
| Abbildung 7: Ubuntu Filesystem Schichten (DOCKER INC. 2016b) | 20 |
| Abbildung 8: Vergleich der Architektur von virtuellen Maschinen und Containern | 22 |
| Abbildung 9: NVIDIA GRID Architektur | 24 |
| Abbildung 10: Richtig/Falsch-Beispiel einer Zuweisung der GRID K2 Profile | 25 |
| Abbildung 11: Architektur der Netzwerkumgebung | 28 |
| Abbildung 12: Die Testumgebung in AWS | 31 |
| Abbildung 13: Regionen und Availability Zonen Konzept (AMAZON WEB SERVICES, INC. 2016a) | 32 |
| Abbildung 14: Beispiel einer VPC mit privaten und öffentlichen Subnetzen (AMAZON WEB SERVICES, INC. 2016b) | 34 |
| Abbildung 15: Database-Mirroring in der Testumgebung | 37 |
| Abbildung 16: Ansicht im Citrix Studio nach erfolgreicher Installation | 38 |
| Abbildung 17: Anmeldemaske des StoreFronts | 39 |
| Abbildung 18: IP-Adressen des NetScalers (SLIDEShare.NET)..... | 40 |
| Abbildung 19: NetScaler-Gateway über die öffentliche IP-Adresse | 41 |
| Abbildung 20: AWS CloudFormation Designer..... | 42 |
| Abbildung 21: Erstellung eines Stacks über AWS CloudFormation | 44 |
| Abbildung 22: Stack-Ansicht über AWS CloudFormation | 44 |
| Abbildung 23: Ablauf eines Verbindungsaufbaus | 45 |
| Abbildung 24: Maschinenkatalog im Citrix Studio..... | 47 |
| Abbildung 25: Hinzufügen der XenApps über das Start-Menü des Host-Servers | 47 |
| Abbildung 26: Apps stehen nach erfolgreicher Einrichtung auf dem StoreFront bereit | 48 |
| Abbildung 27: Unterschied zwischen einer NVIDIA GRID GPU und einer „normalen“ NVIDIA GPU Architektur mit NVIDIA Docker | 51 |
| Abbildung 28: Informationen der vGPU im XenCenter | 53 |
| Abbildung 29: Docker Informationen einer VM im XenCenter | 58 |
| Abbildung 30: Ansicht des Docker-Containers im XenCenter..... | 59 |
| Abbildung 31: Konzept der Integration der Docker-Container als XenApps..... | 61 |
| Abbildung 32: "Docker XenApps" im StoreFront | 70 |
| Abbildung 33: Initiale Verbindung der "Docker-XenApp" zum Container | 71 |

| | |
|--|----|
| Abbildung 34: Das angelegte "Connection-File" im User-Profil | 71 |
| Abbildung 35: Ausgabe des CUDA-Programms..... | 71 |
| Abbildung 36: RealVNC Viewer im StoreFront..... | 75 |
| Abbildung 37: Der virtuelle Desktop innerhalb des Containers..... | 75 |

1. Deutsche Kurzfassung

Diese Arbeit beschäftigt sich mit der Integration von Docker-Containern in eine virtuelle Umgebung, die mit Produkten von Citrix Systems aufgesetzt wurde. Dabei soll diese Lösung für Software-Entwickler konzipiert werden, die Grafik-unterstützte Anwendungen mit CUDA, OpenCL, etc. programmieren. Deshalb müssen die Container fähig sein, GPU-Berechnungen durchführen zu können, um den Entwicklern eine Möglichkeit zu bieten, ihre grafikfähigen Anwendungen in den Containern laufen lassen zu können.

2. Englische Kurzfassung

This dissertation aims at developing an integration for docker-container in a virtual environment that is built with products from the Citrix Systems. It is meant to be used by software-developers, who are programming GPU-applications with CUDA, OpenCL, etc. Therefore the docker-containers should be able to perform calculations on the GPU to enable developers to run their GPU-applications in these containers.

3. Einleitung

3.1 Motivation

X86-Server wurden dafür ausgelegt, dass auf einem Server nur ein einziges Betriebssystem mit einer Anwendung läuft. Das stellt IT-Unternehmen vor ein großes Problem, da selbst kleine Rechenzentren eine Vielzahl an Servern benötigen, die dann nur zwischen etwa fünf bis fünfzehn Prozent ausgelastet sind.¹ Um eine höhere Effizienz der Hardware zu erreichen, setzen die meisten IT-Unternehmen, vor allem im Server-Bereich, Virtualisierung ein. Virtualisierung bietet die Möglichkeit, durch Abstraktion der Hard- und Softwareschicht, mehrere Betriebssysteme auf einer physischen Maschine laufen zu lassen. Das macht es IT-Abteilungen leichter, eine performante, ausfallsichere und effiziente IT-Infrastruktur möglichst kosteneffizient für ein Unternehmen bereit zu stellen.

Aber auch im Hinblick auf den Trend, dass immer weniger Unternehmen ihre eigene Infrastruktur betreiben wollen, sondern ihre IT-Anforderungen entweder vollständig (Public Cloud) oder teilweise (Hybrid Cloud) an externe Firmen outsourcen, die „IT as a service“, also Managed-Services anbieten, machen die Technologie der Virtualisierung umso bedeutender. Das liegt daran, dass diese Cloud-Lösungen kaum ohne die Technologie der Virtualisierung auskommen. Zumal viele Unternehmen nicht nur ihre Serverumgebung virtualisieren, sondern auch Applikation oder normale Desktops mit einem Windows- oder Linux-Betriebssystem in einer Cloud zur Verfügung stellen - oder von Dienstleistern stellen lassen. Man spricht im Allgemeinen davon, dass die Virtualisierung die Basis für dieses so genannte Cloud-Computing bereitstellt.²

Um die Performance dieser virtuellen Systeme sowohl im Serverbereich, als auch im Cloud-Computing zu steigern, setzen die meisten Unternehmen mittlerweile auf „virtual graphics processing units“ (vGPUs). Das sind Grafikkarten, die in einen Server eingebaut werden, um einen Teil der Rechenlast von der CPU parallelisiert auf die vGPU auszulagern.

Es stellt sich also einerseits die Frage, wie genau diese Technologie funktioniert. Andererseits aber auch, wie sie in den Einklang mit bereits bestehenden und neuen Lösungen im Virtualisierungsbereich, wie beispielsweise Container-Virtualisierung, gebracht werden kann. Eben dieser Frage soll in der folgenden Arbeit nachgegangen werden.

¹ VMWARE, Virtualisierung und Software für virtuelle Maschinen (2016).
<http://www.vmware.com/de/solutions/virtualization.html>. (30.11.2016)

² Virtualisierung als Basis für Cloud Computing.
<http://www.computerworld.ch/whitepapers/it-management/artikel/virtualisierung-als-basis-fuer-cloud-computing-65834/>. (07.07.2016)

3.2 Ausgangssituation

Derzeit bestehen kaum Lösungsansätze oder nützliche Integrationen von Docker-Containern in eine virtuelle Citrix-Umgebung. Die native Unterstützung von Citrix erlaubt es lediglich, Docker-Container als Service, dem sogenannten Container-as-a-Service (CaaS) Prinzip, zu betreiben -- und dies auch nur, wenn der Server, auf dem die Container laufen, über Citrix eigenen Hypervisor „XenServer“ verfügt. Dabei können Container mit laufenden Anwendungen zwar betrieben werden, aber sie dienen lediglich als leichtgewichtige, virtuelle Instanzen. Ein Beispiel dafür sind Webserver wie Microsoft IIS (Internet Information Services) oder CMS-Anwendungen (Content-Management-System) wie Wordpress, mit welchen Websites betrieben werden können. Das Container-Management von Citrix dient hierbei hauptsächlich der groben Administration der Container.

Um Software-Entwicklern ein Arbeiten an den Containern zu ermöglichen, also um Images und Container zu erstellen und zu bearbeiten, benötigen diese neben einen Zugriff auf die Host-VM auch eine Möglichkeit, die Container schnell zu erreichen.

In Anbetracht dessen, dass zunehmend Server mit einer Grafikkarte zur Performancesteigerung erweitert werden, wäre es zudem von Vorteil, diese Grafikkarte innerhalb der Docker-Container benutzen zu können. Damit könnten zum Beispiel Entwickler von grafikbeschleunigten Programmen, welche die Programmiersprachen CUDA, OpenCL, etc. verwenden, die Vorteile von Docker-Containern nutzen. Inwieweit bereits bestehende Lösungen für grafikbeschleunigte Container auf die Verwendung von Server-Grafikkarten reagieren, ist jedoch nicht bekannt und muss evaluiert werden.

3.3 Ziel der Arbeit

Ziel der Arbeit ist es, aufzuzeigen, wie eine virtuelle Citrix-Umgebung mit Docker-Containern erweitert werden kann. Des Weiteren soll eine Lösung erarbeitet werden, wie die Servergrafikkarte eines XenServers an die Docker-Container durchgereicht werden kann, ohne die Portabilität der Container einzuschränken.

Im Mittelpunkt der Arbeit steht dabei die Ausarbeitung einer Lösung für eine geeignete Integration der Docker-Container für Software-Entwickler.

3.4 Aufbau der Arbeit

Die vorliegende Arbeit thematisiert zunächst die Funktionsweise der Virtualisierung. Anschließend werden die Funktionsweise eines Container-Management-Systems namens Docker und dessen Bedeutung für die Virtualisierung aufgegriffen. Daraufhin wird auf das Prinzip der virtuellen Grafikkarten eingegangen.

In der Praxis wird zunächst eine virtuelle Umgebung geplant und aufgebaut. Dabei kommen Software-Produkte von Citrix Systems zur Verwendung.

Daraufhin wird untersucht, wie Container mit Unterstützung einer vGPU bereitgestellt werden können.

Anschließend werden die Container integriert. Dazu wird zunächst untersucht, wie die native Unterstützung der Docker-Container mit Citrix auf Administrationsebene funktioniert. Danach wird eine geeignete Integration der Container für Software-Entwickler entwickelt.

Eine Zusammenfassung der Ergebnisse und ein kurzer Ausblick auf mögliche Weiterentwicklungen der Integration schließen die Arbeit ab.

4. Stand der Technik

4.1 Einführung in die Virtualisierung

Den Begriff „Virtualisierung“ gibt es in der IT schon seit ca. 1960. Damals wurde die Virtualisierung benutzt, um Hardwareressourcen von Großrechnern, so genannten „Mainframe-Rechnern“, aufzuteilen. Die Idee hinter der Virtualisierung war, die Hardwareressourcen dieser Großrechner besser auszunutzen, indem auf einem physischen Mainframe mehrere virtuelle Mainframes betrieben wurden. Diese Technik hat sich in ihrer Grundidee bis heute kaum verändert. 1999 wurde der erste Hypervisor, damals noch Virtual Machine Monitor genannt, von VMware veröffentlicht und etabliert sich auch heute noch, nach stetiger Weiterentwicklung, immer noch am Markt.³

Mittlerweile ist die Virtualisierung im IT-Bereich kaum wegzudenken. Selbst große Firmen wie Amazon nutzen diese Technologie im Bereich des Cloud Hostings. Aber auch im privaten Umfeld wird diese Technologie beispielsweise mit der VirtualBox von Oracle immer mehr verwendet.

Mit der „Container Virtualisierung“, die vor allem durch die Unterstützung im neuen Server-Betriebssystem „Windows Server 2016“ im Jahr 2016 populär wurde, erhält diese Technologie auch gerade jetzt und in Zukunft einen Aufschwung und bleibt für die meisten IT-Firmen ein zentraler Bestandteil.

4.2 Grundlagen der Virtualisierung

Der Autor Henrik Mühe beschreibt die Virtualisierung in seiner Veröffentlichung „Virtualisierung – Geschichte, Techniken und Anwendungsfälle“ als „eine Methode zur Teilung der hardwaremäßigen Ressourcen eines Computers, die dafür sorgt, dass mehrere Ausführungsumgebungen auf einer Hardware entstehen“ (MÜHE 2007).

Diese Definition ist im Hauptanwendungsgebiet der Virtualisierung zwar nachvollziehbar formuliert, es muss aber zwischen den Möglichkeiten, Ressourcen unter Zuhilfenahme von Software zentral zusammenzufassen oder aufzuteilen, differenziert werden. Der Unterschied dabei liegt im Einsatzszenario der Virtualisierungslösung. Bei der Zusammenfassung werden mehrere Maschinen zu einer für den Anwender sichtbaren, virtuellen Maschine vereint. Ein Beispiel dafür sind Grids, bei welchen ein Cluster von mehreren verteilten Computern zu einem virtuellen „Supercomputer“ vereinigt wird, der in der Lage ist,

³ Henrik MÜHE, Virtualisierung - Geschichte, Techniken und Anwendungsfälle/ Henrik Mühe; Studienarbeit. Literaturverz. S. 26 (2007)

komplexe Rechenanforderungen unter den einzelnen Maschinen aufzuteilen und somit performant abarbeiten kann.

Bei der Aufteilung physischer Ressourcen werden die Hardwareressourcen eines Wirt-Systems (engl. Host) aufgeteilt und mehreren kleineren virtuellen Gast-Maschinen (engl. Guest) zugeteilt. Nach außen sieht es so aus, als habe jeder dieser aufgeteilten Maschinen seine eigene Hardware, obwohl sie sich eigentlich nur aus dem Hardwarepool des Hosts bedienen -- diese Maschinen und ihre Hardware erscheinen demnach als virtuell.

In dieser Arbeit liegt der Fokus hauptsächlich auf der Aufteilung der physischen Ressourcen.

Generell kann zwischen folgenden Virtualisierungsarten unterschieden werden:

- Plattformvirtualisierung
- Anwendungsvirtualisierung
- Desktopvirtualisierung

4.2.1 Plattformvirtualisierung

Unter der Plattformvirtualisierung versteht man die Simulation eines Rechnersystems (virtuelle Maschine), auf dem ein Gast-Betriebssystem (Guest-OS) bereitgestellt wird. Für diesen Vorgang wird auf dem Host-System eine Steuer-Software installiert, die für die Simulation der Hardwareressourcen, die Bereitstellung des Gast-Betriebssystems und die Befehlsverarbeitung der Gast-Maschinen zuständig ist.⁴ Diese Steuer-Software bezeichnet man als „Hypervisor“.

4.2.1.1 Hypervisoren

Der Hypervisor oder auch Virtual Machine Monitor (kurz VMM) ist ein Software-Layer, das sich zwischen der Hardware und den virtuellen Maschinen befindet.⁵ Ohne ihn würden Befehle direkt von den Betriebssystemen der virtuellen Maschine an die Hardware weitergeleitet werden. Damit würden viele gleichzeitige Anfragen auf Hardwareressourcen erfolgen, so dass sie ständig belegt wären. Das Ergebnis wäre eine chaotische Befehlsabarbeitung, die das System letztendlich ausbremsen und sogar zum Abstürzen bringen

⁴ JINGLI XU, Einsatz von Virtualisierungstechnologien in der IT-Infrastruktur eines Unternehmens: Vergleich und Einteilung mit Ausrichtung auf Desktopbereitstellung (2009)

⁵ KVM Best Practices. Virtualisierungslösungen für den Enterprise-Bereich (2012)

würde. Der Hypervisor regelt die Anfragen der virtuellen Betriebssysteme an die Hardwareressourcen, die von den virtuellen Maschinen geteilt werden.⁶

Man unterscheidet bei der Virtualisierung zwischen zwei Abstraktionsschichten:

- Typ-1-Hypervisor
- Typ-2-Hypervisor

Typ-1-Hypervisor

Typ-1-Hypvisoren laufen direkt auf dem Server oder der Gast-Hardware, ohne dass ein Betriebssystem als Basis für den Hypervisor nötig wäre. Da der Hypervisor direkt auf der physischen Hardware läuft, wird diese Lösung auch als bare-metal Implementation bezeichnet.⁷ Das hat den Vorteil, dass der Hypervisor direkt mit der Hardware kommunizieren kann, ohne den Umweg über ein Betriebssystem gehen zu müssen. Durch den kürzeren Kommunikationsweg ist eine performantere Befehlsabarbeitung möglich. Abbildung 1 zeigt, wie die Architektur eines solchen Typ-1-Hypervisors aussieht.

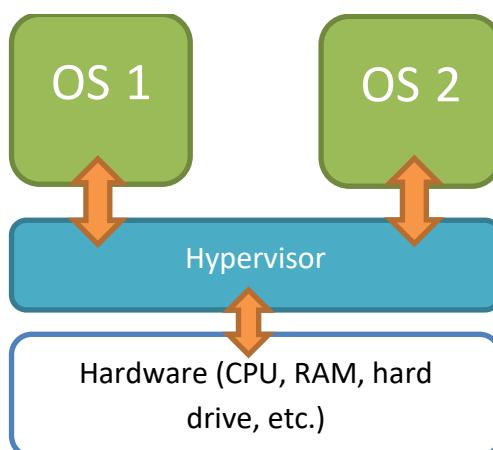


Abbildung 1: Typ-1-Hypervisor

Jeder der virtuellen Maschinen bekommt ihre eigenen virtuellen Hardwareressourcen vom Hypervisor zugeordnet. Es sieht demnach für jede Partition so aus, als wäre es ein eigener, vollständiger, Computer mit eigener Hardware.⁸

⁶ Matthew PORTNOY, Virtualization Essentials (Essentials2012)

⁷ PORTNOY, Virtualization Essentials (wie Anm. 6)

⁸ Robert VOGEL / Tarkan KOÇOGLU / Thomas BERGER, Desktop Virtualisierung. Definitionen - Architekturen - Business-Nutzen (2010)

Typ-2-Hypervisor

Der Typ-2-Hypervisor unterscheidet sich vom Typ-1-Hypervisor insofern, dass die Virtualisierungsschicht nicht direkt auf der Hardware liegt, sondern auf einem kompletten Betriebssystem. Deshalb werden Befehle nicht direkt von den Guest-Betriebssystemen über den Hypervisor an die Hardware weitergeleitet, sondern müssen aufwendig emuliert werden. Emulieren bedeutet, dass die komplette Hardwarearchitektur des Hosts auf dem Guest-Betriebssystem repliziert wird.⁹ Dadurch entsteht viel Overhead und macht diese Art der Virtualisierung aufgrund des großen Unterbaus instabil und somit weniger leistungsfest im Vergleich zu den Typ-1-Hypervisoren.¹⁰

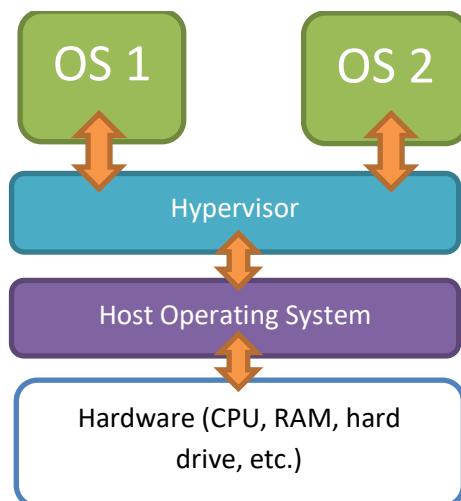


Abbildung 2: Typ-2-Hypervisor

4.2.1.2 Grundlagen der x86-Virtualisierung

Die x86-Prozessor-Architektur führt die Prozesse in einer bestimmten Privilegstufe aus. Abbildung 4 a) zeigt, wie die Privileg-Architektur der x86-Architektur aussieht. Wie man sieht, bestehen diese Privilegstufen aus insgesamt vier Level, die von innen nach außen als Ringe 0, 1, 2, und 3 bezeichnet werden. Während Benutzerprozesse im äußersten Ring (Ring 3) laufen, dem Benutzermodus (User-Mode), läuft der Betriebssystemkern im innersten Ring (Ring 0), dem Kernel-Modus (Kernel-Mode).¹¹

Der Grund für die Unterteilung sind die unterschiedlichen Privilegien, die bei der Ausführung von Befehlen zugeteilt werden. Generell werden bei der x86-Architektur die Befehle

⁹ Virtualisierung › Wiki › ubuntuusers.de.

<https://wiki.ubuntuusers.de/Virtualisierung/#Paravirtualisierung>. (08.07.2016)

¹⁰ VOGEL / KOÇOGLU / BERGER, Desktop Virtualisierung (wie Anm. 8)

¹¹ Seminar "Virtualisierung von Betriebssystemen: VMware Architektur" (2007).

<http://www.fh-wedel.de/~si/seminare/ws06/Ausarbeitung/02.VMware/vmware4.htm>. (14.07.2016)

in privilegierte und nicht privilegierte Befehle separiert. Privilegierte Befehle dürfen nur im Ring 0, also im Kernel-Mode, ausgeführt werden und haben Zugriff auf die Systemressourcen, wie dem CPU-Befehlssatz. Mit dem CPU-Befehlssatz können bestimmte Bereiche des Hauptspeichers direkt angesprochen werden. Diese Bereiche sind zum Beispiel Speicherbereiche für I/O-Geräte oder kritische funktionsrelevante Stellen für das Betriebssystem.¹²

In den Ringen 1 und 2 werden Befehle mit eingeschränkten Privilegien ausgeführt. Die Intention von Intel dahinter war es, dass dort Gerätetreiber angesiedelt werden können, die zwar privilegierten Zugriff haben, aber dennoch vom Kernel-Code im Ring 0 abgeschottet sind.¹³

Nicht privilegierte Befehle werden in dem Ring 3 ausgeführt.

Obgleich die x86-Architektur diesen Ring-Schutz bereitstellt, nutzen die meisten Betriebssysteme nicht alle Ringe. Tatsächlich bedient sich nur das Betriebssystem OS/2 (Operating System /2)¹⁴ aller Ringe, die meisten Betriebssysteme nutzen nur den Ring 0 und 3.¹⁵

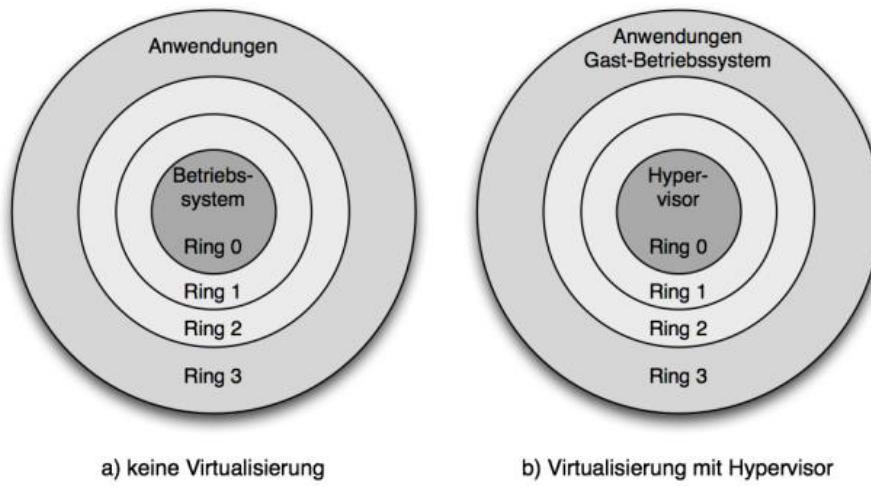


Abbildung 3: Ringmodell für x86-Prozessoren (MEINEL 2010)

¹² Christoph MEINEL, Virtualisierung und Cloud Computing : Konzepte, Technologiestudie, Marktübersicht (Technische Berichte des Hasso-Plattner-Instituts für Softwaresystemtechnik an der Universität Potsdam ; 44) (2010)

¹³ MEINEL, Virtualisierung und Cloud Computing : Konzepte, Technologiestudie, Marktübersicht (Technische Berichte des Hasso-Plattner-Instituts für Softwaresystemtechnik an der Universität Potsdam ; 44) (wie Anm. 12)

¹⁴ Dirk MAKOWSKI, Microsoft OS/2.

<http://www.winhistory.de/more/os2.htm>. (14.07.2016)

¹⁵ Udo HELMBRECHT / Gunnar TEEGE / Björn STELTE, Virtualisierung: Techniken und sicherheitsorientierte Anwendungen. Universität der Bundeswehr München - Institut für Technische Informatik (2010)

Eine Anwendung, die einen direkten Zugriff auf den Hauptspeicher oder den privilegierten CPU-Befehlssatz haben will, erhält keine Berechtigung zur Ausführung dieser Befehle im kernel-reservierten Ring 0. Es tritt eine Speicherschutzverletzung auf und das Programm stürzt mit hoher Wahrscheinlichkeit ab. Die Frage ist nun, was passiert, wenn eine virtuelle Maschine auf dem System laufen soll. Denn standardmäßig wird die virtuelle Maschine im Ring 3 ausgeführt und erhält somit keinen Zugriff auf die privilegierten Ressourcen. Auch die Kernel der Gast-Betriebssysteme der virtuellen Maschinen laufen nicht im Ring 0 sondern ebenfalls im Ring 3. Bei einem direkten Zugriff der VM auf die privilegierten Speicherbereiche wird demnach eine Speicherzugriffsverletzung ausgelöst und die VM stürzt ab.

Um die Virtualisierung auf x86-Systemen zu ermöglichen, muss also sichergestellt werden, dass virtuelle Maschinen zwar einen Zugriff auf den privilegierten Modus haben, aber trotzdem die Sicherheit und Stabilität des Host-Systems nicht gefährdet wird. Der Lösungsansatz dabei ist es, einen Vermittler, den Hypervisor, in Ring 0 einzuführen.¹⁶

4.2.1.3 Vollvirtualisierung

Bei der Vollvirtualisierung (oder auch vollständigen Virtualisierung) läuft das Guest-OS direkt auf einer Abstraktionsschicht und hat keinen direkten Zugriff auf die Hardware. Wie in Abbildung 4 b) zu sehen, wird im Ring 0 ein Hypervisor betrieben, der einer bzw. mehreren virtuellen Maschinen einen Teil der Hardware als virtuelle Hardware zuweist. Die VM sieht nach außen wie ein normaler Rechner aus. Das, bzw. die Guest-OS, werden dabei in einem höheren Ring als Ring 0 ausgeführt und müssen nicht modifiziert werden.

Versucht eine VM einen privilegierten Befehl auszuführen, wird eine Exception ausgelöst und das Gast-System unterbrochen. Daraufhin wird die Operation, die von der VM aus zur Exception geführt hat, abgefangen und zum Hypervisor weitergeleitet. Der Hypervisor merkt sich die Quelle des Befehls (also die VM) und überarbeitet den Befehl so, dass der Hypervisor die Direktive selbst im Ring 0 ausführen kann. Nach dem Ausführen des Befehls, nimmt der Hypervisor das Ergebnis entgegen und konvertiert die Rückgabe für die virtuelle Maschine.¹⁷

In der x86 Architektur gibt es jedoch auch privilegierte Befehle, die sich in höheren Ringen anders verhalten als in Ring 0, aber keine Exception auslösen. Diese Befehle würden

¹⁶ MEINEL, Virtualisierung und Cloud Computing : Konzepte, Technologiestudie, Marktübersicht (Technische Berichte des Hasso-Plattner-Instituts für Softwaresystemtechnik an der Universität Potsdam ; 44) (wie Anm. 12)

¹⁷ MEINEL, Virtualisierung und Cloud Computing : Konzepte, Technologiestudie, Marktübersicht (Technische Berichte des Hasso-Plattner-Instituts für Softwaresystemtechnik an der Universität Potsdam ; 44) (wie Anm. 12)

beim Ausführen zu einem Fehlverhalten des Gastbetriebssystems führen und müssen deshalb gefunden und abgeändert werden. Aufgrund dessen, dass diese Befehle keine Exception und somit keine Unterbrechung des Programms herbeiführen, ist die Bearbeitung der Befehle zur Laufzeit nur dann möglich, wenn der Programm-Code kurz vor der Ausführung dieser Direktiven analysiert und abgeändert wird. Diese Vorgehensweise wird „binary translation“ genannt. Dabei gibt es zwei Strategien. Entweder wird der Programm-Code an der kritischen Stelle so abgeändert, dass er eine Exception auslöst, die daraufhin an den Hypervisor weitergeleitet wird, oder der Befehl wird so übersetzt, dass er das gewünschte Ergebnis liefert und das Gastbetriebssystem so weiter arbeiten kann, wie es soll. Letztere Methode wird (obgleich der Implementationsaufwand höher ist) bevorzugt, da keine Unterbrechung des Programms und eine Weiterleitung an den Hypervisor stattfinden müssen.¹⁸

4.2.1.4 Paravirtualisierung

Bei der Paravirtualisierung wird der Kernel des Gastbetriebssystems angepasst, sodass die privilegierten, nicht-virtualisierbaren Befehle durch „Hypercalls“ ersetzt werden. Die Hypercalls werden dann direkt an den Hypervisor weitergeleitet. Der Hypervisor nimmt die Hypercalls entgegen, führt die gewünschten System Calls aus und liefert das Ergebnis zurück an das Gastbetriebssystem. Der Hypervisor verfügt zudem über ein Interface für kritische Systemaufrufe wie Speichermanagement und Interrupt Handling.¹⁹ Dadurch fallen die ständigen Exceptions und die Überprüfung des Programmcodes zur Laufzeit weg, was die Paravirtualisierung im Vergleich zur Vollvirtualisierung um einiges performanter macht. Da die Kernel des Gastbetriebssystems angepasst werden müssen, eignen sich open-source Betriebssysteme eher zur Paravirtualisierung, weil nicht-quelloffene Betriebssysteme von ihren Herstellern selbst bearbeitet werden müssen (z.B. Microsoft).²⁰

Einer der bekanntesten Vertreter der Paravirtualisierung ist Xen, ein angepasster Linux-Kernel von Citrix Systems, der unter anderem im praktischen Teil dieser Arbeit verwendet wird. Jede virtuelle Maschine erhält in der Typ2-Xen-Hypervisor-Architektur seine eigene Domäne (Dom), die von den anderen virtuellen Maschinen und dem Hypervisor selbst abgeschottet ist. Auf dem Xen-Hypervisor läuft neben den eigentlichen virtuellen

¹⁸ HELMBRECHT / TEEGE / STELTE, Helmbrecht, Teege et al. 2010 – Virtualisierung (wie Anm. 15)

¹⁹ Chris HORNE, Understanding Full Virtualization, Paravirtualization, and Hardware Assist. White Paper - vmWare

²⁰ MEINEL, Virtualisierung und Cloud Computing : Konzepte, Technologiestudie, Marktübersicht (Technische Berichte des Hasso-Plattner-Instituts für Softwaresystemtechnik an der Universität Potsdam ; 44) (wie Anm. 12)

Maschinen, den DomU, noch eine virtuelle Instanz namens Dom0.²¹ Diese virtuelle Maschine dient als Verwaltungsinstanz und regelt unter anderem den Zugriff auf CPU-, RAM-, Netzwerk- und Storageressourcen. Die DomUs kommunizieren also nicht direkt mit dem Hypervisor, sondern fragen die Ressourcen auf dem Dom0 an, der das Privileg hat, über den Hypervisor die Hardware direkt anzusprechen.²²

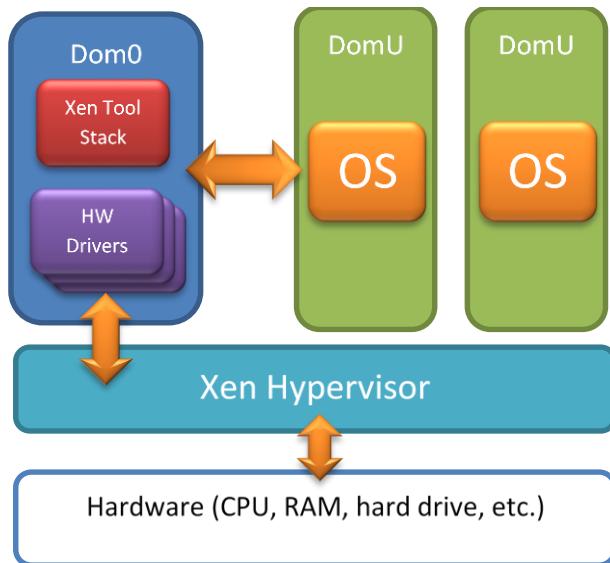


Abbildung 4: XenServer Architektur

4.2.2 Anwendungsvirtualisierung

Bei der Anwendungsvirtualisierung wird im Vergleich zu der Plattformvirtualisierung nicht ein Rechnersystem mit Betriebssystem, sondern die Ausführungsumgebung für eine Applikation virtualisiert.²³

Die Applikationen werden bei dieser Art der Virtualisierung direkt auf einem Server installiert und Benutzern über das Netzwerk bereitgestellt. Es wird nur die Bildschirmausgabe der Applikation an den Benutzer übertragen. Alle Interaktionen des Benutzers, wie Tastatur- und Mauseingaben, werden über das Netzwerk an die Applikation gesendet und dort verarbeitet. Die Vorteile sind einerseits, dass die Applikationen unabhängig von dem Betriebssystem des Benutzers genutzt werden können, andererseits, dass sie alle zentral

²¹ Xen ARM with Virtualization Extensions whitepaper - Xen (2016).

http://wiki.xen.org/wiki/Xen_ARM_with_Virtualization_Extensions_whitepaper. (08.07.2016)

²² Dom0 - Xen (2015).

<http://wiki.xenproject.org/wiki/Dom0>. (08.07.2016)

²³ Desktop, Client, Server, Anwendung: Ratgeber: Was ist was bei der Virtualisierung.

<http://www.tecchannel.de/a/ratgeber-was-ist-was-bei-der-virtualisierung,2037095,5>. (09.07.2016)

auf einem Serversystem betrieben werden und aus diesem Grund einfacher zu administrieren sind.²⁴

Ein Beispiel für die Virtualisierung von Anwendungen ist XenApp von Citrix Systems, dass auch in dieser Arbeit verwendet wird.²⁵

4.2.3 Desktopvirtualisierung

Die Desktopvirtualisierung stellt eine Weiterentwicklung im Bereich der Plattform- und Anwendungsvirtualisierung dar. Bei dieser Art der Virtualisierung wird ein kompletter Desktop, inklusive Anwendungen und Benutzerprofil, virtuell bereitgestellt und über einen Client erreicht.

Eine Breitstellung von mehreren virtuellen Desktops wird auch als Virtual Desktop Infrastructure (VDI) bezeichnet. Eine bekannte VDI-Lösung stellt Citrix Systems bereit, namens XenDesktop.²⁶

4.3 Docker

Entwickler stehen häufig vor dem Problem, dass Applikationen, vor allem im wissenschaftlichen Bereich (z.B. Deep-Learning) oder im Visualisierungsbereich (z.B. Echtzeitrendering, Simulation), eine Vielzahl von zusätzlichen Paketen benötigen. Die Aufgabe, diese Umgebung für jeden einzelnen Entwickler mit den benötigten Paketen aufzusetzen, und etwaige Änderungen vorzunehmen, gestaltet sich als sehr zeitintensiv. Eine Lösung für dieses Problem sind Docker-Container.

Docker wurde dafür konzipiert, Software mit all ihren Abhängigkeiten, wie beispielsweise Bibliotheken, schnell von einer zur anderen Umgebung zu verschicken, unabhängig davon, um was für eine Umgebung es sich handelt -- ähnlich einem Hafenarbeiter (engl. docker), der Container auf verschiedene Schiffe verfrachtet.

²⁴ VOGEL / KOCOGLU / BERGER, Desktop Virtualisierung (wie Anm. 8)

²⁵ Jonathan DESROCHER, Running Hyper-V VMware or Xen on an AWS EC2 Instance? - CloudStacking.com (2013).

[\(05.12.2016\)](http://cloudstacking.com/posts/running-hyper-v-vmware-or-xen-on-an-aws-ec2-instance.html)

²⁶ DESROCHER, Running Hyper-V VMware or Xen on an AWS EC2 Instance? - CloudStacking.com (wie Anm. 25)

4.3.1 Docker Komponenten

Die Softwarelösung Docker besteht insgesamt aus vier Komponenten (siehe Abbildung 5):²⁷

- Docker Client und Server
- Docker Images
- Registries
- Docker Container

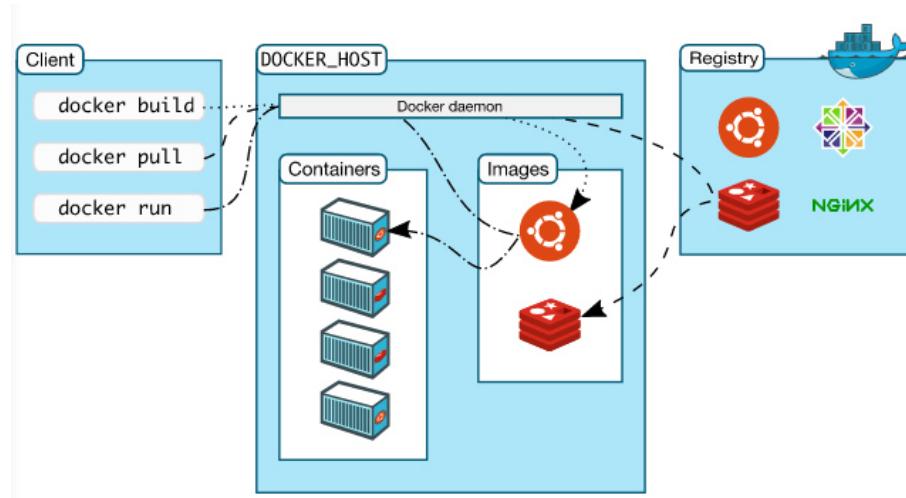


Abbildung 5: Docker Architektur (DOCKER INC. 2016c)

4.3.1.1 Docker Client und Server

Docker basiert auf der Client-Server Architektur. Dabei wird der Docker-Client verwendet, um mit dem Docker-Server, oder auch Daemon, zu kommunizieren. Der Docker-Client setzt die Befehle von der Erstellung bis hin zur Inbetriebnahme und Verwaltung der Container ab und der Docker-Daemon führt die Anweisungen anschließend aus. Sowohl der Daemon, als auch der Client, können beide auf demselben System laufen oder man verbindet den lokalen Client auf dem Host mit einem entfernten Daemon, der auf einem anderen Host läuft. Der Client bietet für die Kommunikation mit dem Daemon sowohl Kommandozeilen-Befehle als auch eine RESTful API an.^{28 29}

²⁷ James TURNBULL, The Docker book (2014)

²⁸ DOCKER INC., Remote API (2016).

https://docs.docker.com/engine/reference/api/docker_remote_api/. (09.09.2016)

²⁹ TURNBULL, The Docker book (wie Anm. 27)

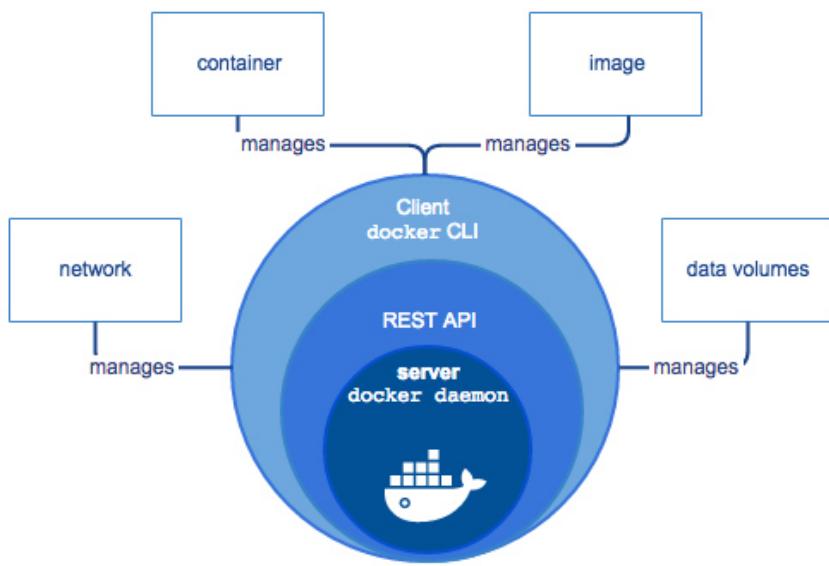
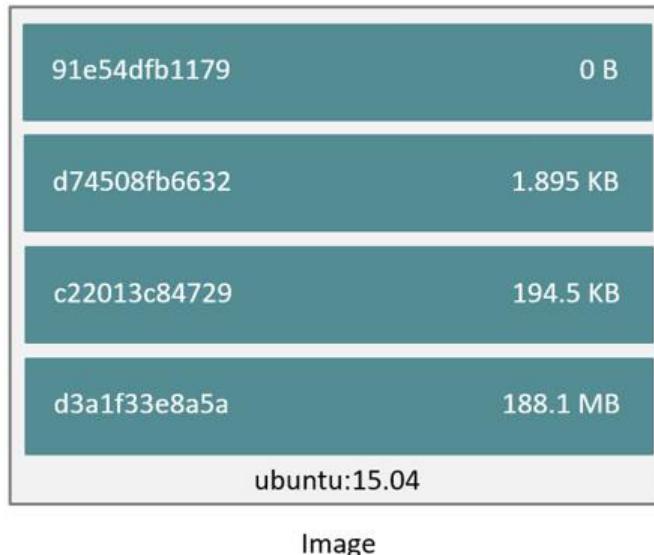


Abbildung 6: Docker Client-Server Architektur (DOCKER INC. 2016a)

4.3.1.2 Docker-Images

Als Basis für die Container dienen die Docker-Images. Docker-Images bestehen aus einem oder mehreren Filesystem-Schichten. Die Filesystem-Schichten sind Abbildungen der einzelnen „Bauschritte“, die benötigt wurden, um das Image zu erstellen. Jede Schicht repräsentiert dabei eine Änderung zum vorhergehenden Filesystem. Die Schichten werden dann übereinander „gestapelt“ und somit zu einem Image zusammengefügt. Abbildung 7 zeigt den Aufbau eines Ubuntu 15.04 Images mit vier Filesystem-Schichten.³⁰

³⁰ DOCKER INC., Understand images, containers, and storage drivers (2016b).
<https://docs.docker.com/engine/userguide/storagedriver/imagesandcontainers/>. (26.11.2016)



Image

Abbildung 7: Ubuntu Filesystem Schichten (Docker Inc. 2016b)

Das Image an sich wird über das Dockerfile, eine einfache Textdatei mit Anweisungen zum Erstellen des Images, generiert:³¹

```
# Base-Image
FROM ubuntu:14.04
MAINTAINER Max Mustermann

# Install nginx WebServer
RUN apt-get update
RUN apt-get install -y nginx

# Create index.html
RUN echo 'Hello World' > /usr/share/nginx/html/index.html

# Expose Port
EXPOSE 80
```

Listing 1: Einfaches Dockerfile für das Erstellen eines nginx-Webservers

Listing 1 zeigt ein Dockerfile, das einen nginx-Webserver installiert und eine Website (index.html) erstellt, die beim Aufrufen über die IP-Adresse des Host-Systems auf Port 80 „Hello World“ ausgibt.

4.3.1.3 Registries

Nachdem das Image erstellt wurde, ist der nächste Schritt das Veröffentlichen, bzw. Bereitstellen, des Images, sodass das Image von jedem Docker Host aus heruntergeladen werden kann und aus dem Image die Docker-Container erstellt werden können. Zu diesem Zweck können entweder öffentliche oder private Registries verwendet werden. Re-

³¹ Matthias KARL / Sean P. KANE, Docker: up and running (2016)

gistries sind zentralisierte „Sammelstellen“ für Docker Images, die über einfache Kommandozeilen-Befehle oder einer Benutzeroberfläche gesteuert werden können.

Docker bietet ein öffentliches Image-Registry an, auf dem die Entwickler ihre Images bereitstellen können. Dieses öffentliche Image-Registry heißt „Docker Hub“. In diesem öffentlichen Registry befinden sich neben den privat erstellten Images auch offizielle Images, wie zum Beispiel WordPress-, MySQL- und Ubuntu-Images.

Docker Hub verfügt über ein einfach zu bedienendes User-Interface sowie eine Benutzerverwaltung und bietet die Möglichkeit, die Zugriffsrechte für Teams aufzuteilen. Durch die große Community steht eine Vielzahl an verschiedenen Docker-Images bereit.

Für diejenigen, die ihre Images nicht in eine Online-Registry überführen wollen, sondern lieber eine Art „interne Docker-Image-Registry“ betreiben wollen, bietet sich docker-distribution³² an, das auf GitHub veröffentlicht wurde. Damit lassen sich lokale Registries betreiben, die über Kommandozeilen-Befehle bedient werden können.

4.3.1.4 Docker-Container

Aus den Docker-Images lassen sich die Docker-Container erstellen. Docker-Container basieren auf LinuxContainer (LXC)³³ und nutzen deren Fähigkeit, den Kernel des Host-Systems zu erweitern. Somit erzeugt ein Docker-Container eine leichtgewichtige virtuelle Laufzeitumgebung, die kein Betriebssystem beinhaltet, sondern den Kernel des Host-Systems benutzt. Daher sind Docker-Container sehr leichtgewichtig und auch portabel.

³² DOCKER INC., docker/distribution (2016).
<https://github.com/docker/distribution>. (10.09.2016)

³³ Linux Containers (2016).
<https://linuxcontainers.org/>. (04.09.2016)

Abbildung 8 zeigt, wie ein Docker-Container im Vergleich zu einer virtuellen Maschine aufgebaut ist.

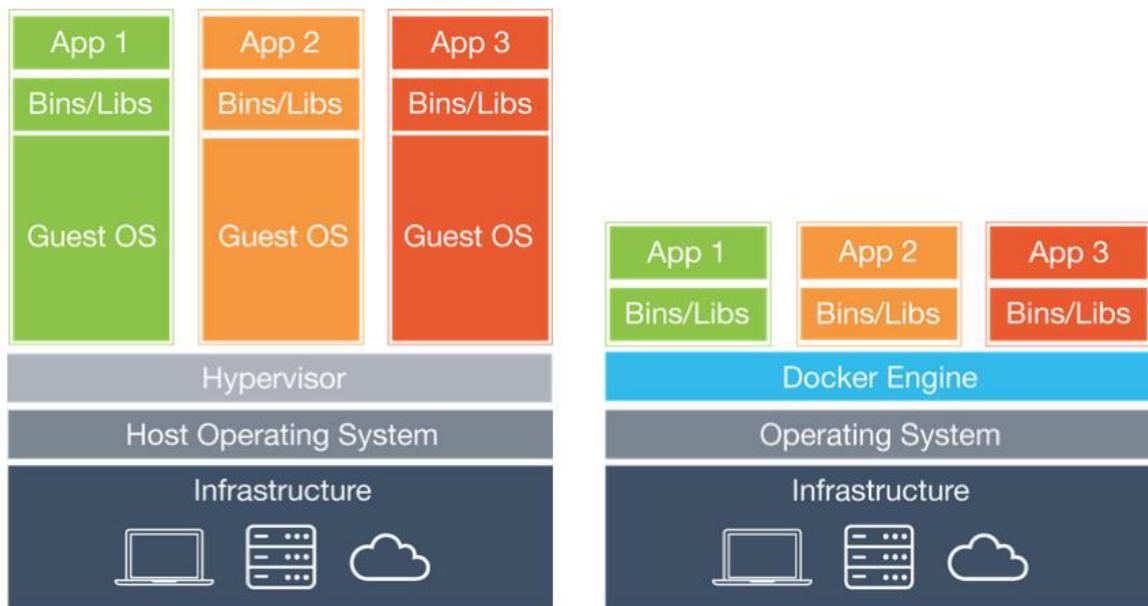


Abbildung 8: Vergleich der Architektur von virtuellen Maschinen und Containern

Bei einer virtuellen Maschine wird auf dem eigentlichen Betriebssystem des Host-Systems ein Hypervisor installiert, der die Kommunikation der virtuellen Instanz zur Hardware übernimmt. Darauf aufbauend wird zudem für jede virtuelle Maschine ein komplettes Betriebssystem, zusätzlich zu dem Host-Betriebssystem installiert, auf dem dann die eigentliche Anwendung mit allen benötigten Binärdateien und Bibliotheken läuft. Container benötigen diese zusätzlichen Betriebssysteme nicht, da sie sich den Kernel des Host-Systems teilen.

Docker-Container sind somit eine ressourcensparende Methode der Virtualisierung, da sie den Speicherplatz, der normalerweise von dem Guest-OS in Anspruch genommen werden würde, nicht benötigen. Zudem starten sie schneller als virtuelle Maschinen und verbrauchen weniger RAM.

Innerhalb der Docker-Container werden die Software selbst und ihre Abhängigkeiten, wie der Programm-Code, Laufzeitvariablen, System-Tools und Systembibliotheken verpackt. Alles, was auf einem Server installiert werden kann, kann auch in Container verpackt werden. Die Isolation der Software garantiert, dass die Software stets gleich läuft, unabhängig von ihrer Umgebung.³⁴

Ein weiterer Vorteil bei der Isolation von Prozessen besteht darin, dass bei einer Sicherheitslücke, im Gegensatz zu komplexen Systemen, nicht das ganze System kompromittiert wird, sondern lediglich ein Teilbereich.³⁵

4.4 GPU-Virtualisierung mit NVIDIA

Virtuelle Umgebungen leiden bei zunehmender Skalierung an Performanceeinbrüchen, da alle Berechnungen sequentiell auf der CPU ausgeführt werden. Vor allem bei grafikintensiven Anwendungen stößt die CPU schnell an ihre Grenzen.

Um die CPU zu entlasten und die Performance von virtuellen Servern, Desktops und Anwendungen zu steigern, werden mittlerweile auch Grafikkarten virtualisiert. Die virtuellen Grafikprozessoren (vGPUs) stehen dann den virtuellen Ressourcen zur Verfügung.

4.4.1 NVIDIA GRID

Die Firma NVIDIA ist einer der führenden Entwickler von Grafikprozessoren. Im Bereich der GPU-Virtualisierung hat NVIDIA eine Lösung namens GRID entwickelt. Mit NVIDIA GRID können mehrere virtuelle Maschinen gleichzeitig auf eine physische Grafikkarte zugreifen.

GRID steht dabei nicht nur für die Lösung selbst, sondern war auch die Bezeichnung der ersten zwei Servergrafikkarten im VDI-Bereich von NVIDIA, der GRID K1 und GRID K2. Mittlerweile bietet NVIDIA (neben den Cloud-Gaming-Grafikkarten NVIDIA GRID K340 und GRID K520) insgesamt fünf Servergrafikkarten an, die NVIDIA GRID K1, GRID K2, M6, M60 und die M10. Diese unterscheiden sich hauptsächlich in ihrer Prozessorarchitektur und Leistung.

³⁴ DOCKER INC., What is Docker? (2016).
<https://www.docker.com/what-docker>. (04.09.2016)

³⁵ DOCKER INC., What is Docker? (wie Anm. 34)

4.4.2 GRID vGPU Architektur

Abbildung 9 zeigt die high-level GRID-vGPU-Architektur.

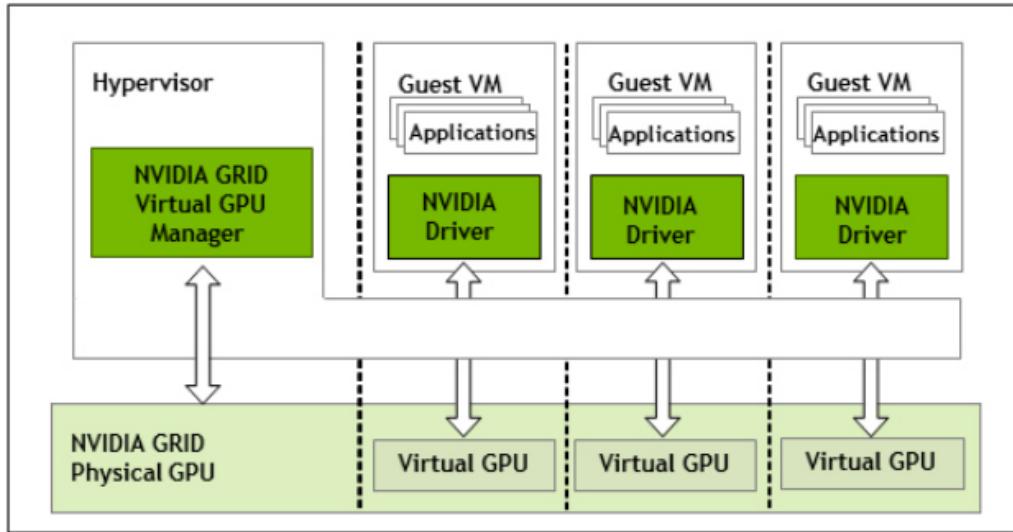


Abbildung 9: NVIDIA GRID Architektur

Neben dem Hypervisor, der die VMs bereitstellt, läuft der sogenannte NVIDIA GRID Virtu-alGPU Manager. Dieser regelt die Zuweisung der einzelnen vGPUs zu den Guest-VMs. Die Guest-VMs nutzen die virtuelle Grafikkarte genau gleich, wie sie es mit einer physi-schen Grafikkarte, die vom Hypervisor durchgereicht werden würde (engl. pass-trough), machen würden. Sie nutzen einen NVIDIA-Treiber, der auf der Host-VM installiert wird und der die Kommunikation zwischen der Guest-VM und der Grafikkarte übernimmt.

Alle GRID vGPUs besitzen einen fixen Framebuffer, auf den die einzelne vGPU exklusi-ven Zugriff hat. Um die Ressourcen der Grafikkarte optimal für mehrere vGPUs auszu-schöpfen, setzt NVIDIA auf Timeslicing. Dabei bekommt jede vGPU für eine gewisse Zeit exklusiven Zugriff auf die GPU-Engine und den gesamten CUDA-Cores. Somit steht der vGPU für eine kurze Zeit die komplette Grafikkarte für Berechnungen zur Verfügung.

4.4.3 GRID Profile

Die NVIDIA GRID vGPUs können entweder per Pass-through, also der direkten Zuwei-sung eines Grafikchips, oder über Profile, mit unterschiedlichen Framebuffer-Skalierungen, den virtuellen Maschinen zugewiesen werden. Des Weiteren unterscheidet sich mit den Profilen, wie viele Bildschirme (Display Heads) über eine Guest-VM ange-schlossen werden können und welche maximale Auflösung die Bildschirme haben kön-nen.

Während beim Pass-through nur eine virtuelle Maschine pro Grafikchip erstellt werden kann, können mittlerweile mit der M10 mit dem kleinsten Profil 64 vGPU Instanzen bereitgestellt werden.

Die folgende Tabelle zeigt die verschiedenen Profile, die für eine GRID K2 Grafikkarte verfügbar sind:

| Physical GPUs | GRID Virtual GPU Profile | Frame Buffer (Mbytes) | Virtual Display Heads | Maximum Resolution per Display Head | Maximum vGPUs per GPU | Maximum per Board |
|---------------|--------------------------|-----------------------|-----------------------|-------------------------------------|-----------------------|-------------------|
| 2 | K280Q | 4096 | 4 | 2560×1600 | 1 | 2 |
| 2 | K260Q | 2048 | 4 | 2560×1600 | 2 | 4 |
| 2 | K240Q | 1024 | 2 | 2560×1600 | 4 | 8 |
| 2 | K220Q | 512 | 2 | 2560×1600 | 8 | 16 |
| 2 | K200 | 256 | 2 | 1920×1200 | 8 | 16 |

Die GRID K2 verfügt über zwei Grafikchips. Die Profile können sich zwar pro Grafikchip unterscheiden, müssen aber auf einem Grafikchip vom selben Typ sein (siehe Abbildung 10).

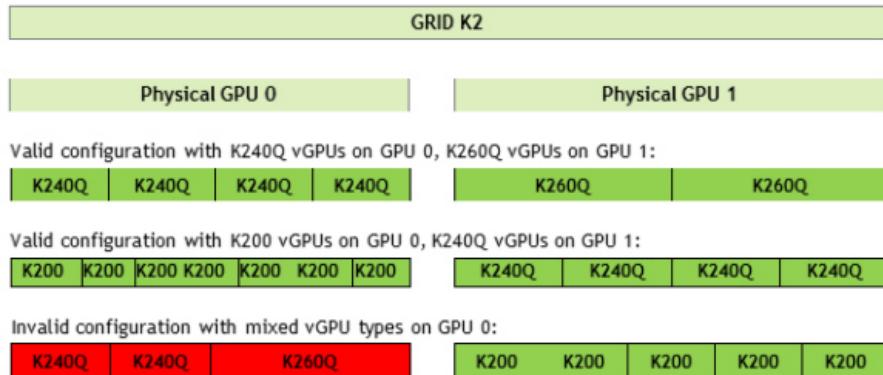


Abbildung 10: Richtig/Falsch-Beispiel einer Zuweisung der GRID K2 Profile

5. Aufgabenstellung

Die Aufgabe dieser Bachelorarbeit besteht darin, aufzuzeigen, wie Docker-Container in eine virtuelle Citrix-Umgebung integriert werden können. Eine weitere Intention ist es, herauszufinden, wie Docker-Container die virtuelle Grafikkarte von GPU-unterstützten Servern nutzen können. Dabei soll diese Arbeit einen Leitfaden darstellen, wie die virtuelle Umgebung in Hinsicht auf Ausfallsicherheit, Hochverfügbarkeit und der Integration der Docker-Container konzeptioniert werden könnte und wie ein Unternehmen diese Umgebung mit eigenen Ressourcen nachstellen könnte.

Hierzu soll eine virtuelle Netzwerkumgebung in Amazon AWS aufgezogen werden, um ein mögliches Netzwerkdesign zu demonstrieren und daraufhin die Citrix-Umgebung installiert werden. Diese virtuelle Umgebung dient als Basis für die Evaluierung der Integration der Docker-Container. Nach der Implementierung der virtuellen Umgebung soll aufgezeigt werden, wie Docker-Container in die bestehende virtuelle Umgebung integriert werden können und ein Konzept ausgearbeitet werden, wie diese grafikbeschleunigt werden können.

Anschließend soll aufgezeigt werden, wie die native Integration der Docker-Container auf dem XenServer funktioniert. Des Weiteren soll eine eigene, für Entwickler konzipierte, Integration entwickelt werden.

6. Beschreibung der Arbeit

6.1 Aufsetzen der Testumgebung

Zunächst muss die virtuelle Testumgebung aufgesetzt werden in der anschließend die Container integriert werden.

6.1.1 Anforderung an die Testumgebung

6.1.1.1 Authentizität

Die virtuelle Umgebung soll für die Validierung der Integration so entworfen werden, wie sie auch in einer produktiven Umgebung aufgesetzt werden würde.

6.1.1.2 Sicherheit

Da die virtuelle Umgebung von außen erreichbar sein soll, ist sie angreifbar. Deshalb müssen Sicherheitsvorkehrungen, wie beispielsweise das Einführen einer Firewall, getroffen werden, damit das System nicht kompromittiert wird.

6.1.1.3 Ausfallsicherheit

Im Bereich der Virtualisierung ist Ausfallsicherheit und Redundanz der kritischen Systeme einer der wichtigsten Faktoren, die beachtet werden müssen, da (in einem produktiven System) mehrere hundert Arbeitsplätze vom fehler- und unterbrechungsfreien Betrieb der virtuellen Umgebung abhängig sind. Sogenannte „Single-Points-of-Failure“, also nicht-redundante Server, Dienste, etc., von denen der komplette Betrieb der virtuellen Umgebung abhängt, sind somit unbedingt zu vermeiden, um die Produktivität eines Unternehmens nicht zu gefährden.

Die Testumgebung soll sich daher auch an das Prinzip der Ausfallsicherheit orientieren, um eine möglichst realistische Ausgangssituation für die Integration zu schaffen.

6.1.2 Netzwerkarchitektur

6.1.2.1 Konzept

Bevor mit dem Aufsetzen der Testumgebung angefangen wurde, war es zunächst erforderlich eine geeignete Architektur für die Umgebung zu entwerfen, die den Anforderungen an die Testumgebung entspricht.

Abbildung 11 zeigt den konzeptionellen Entwurf dieser Umgebung.

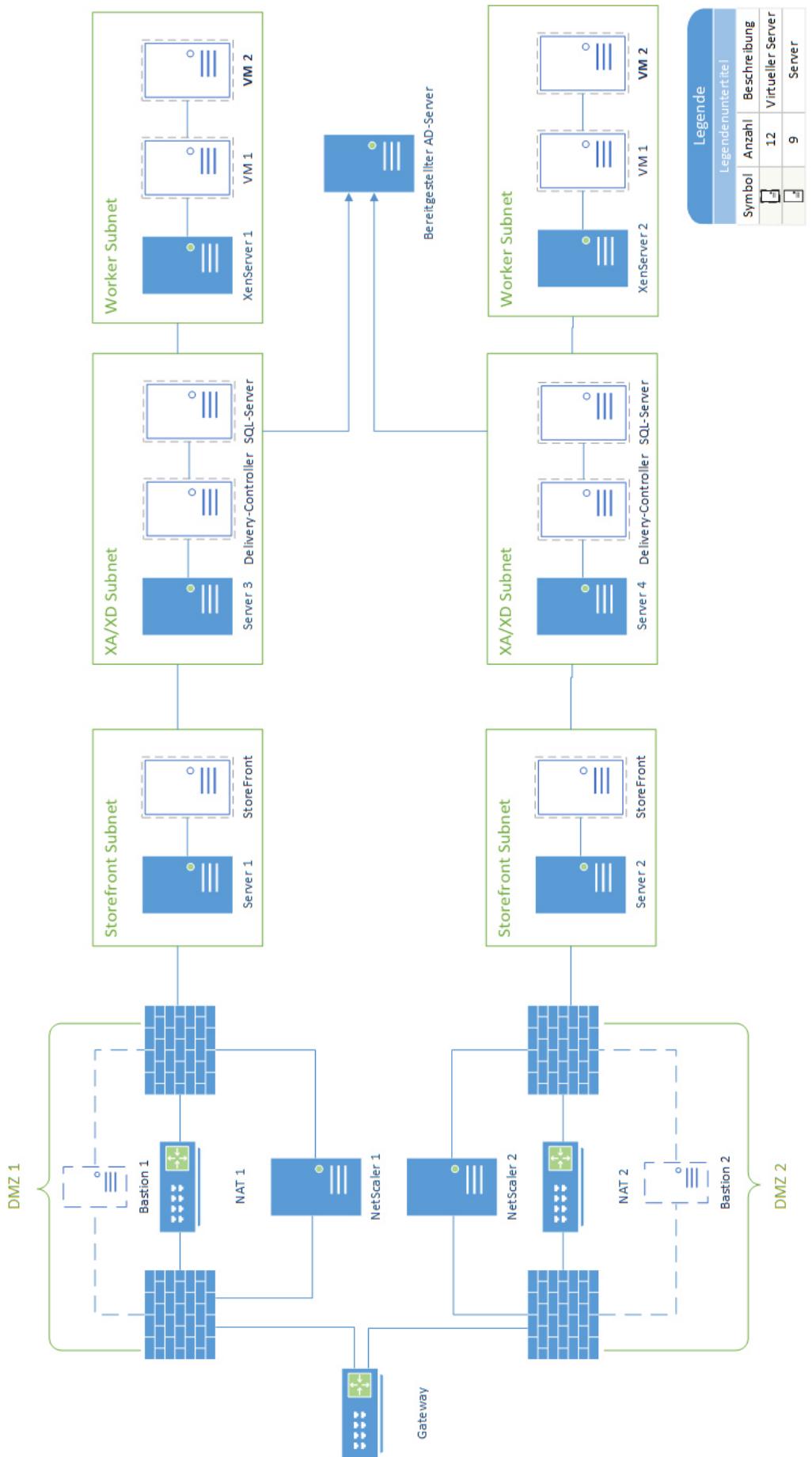


Abbildung 11: Architektur der Netzwerkumgebung

Um die Ausfallsicherheit der Umgebung zu gewährleisten, ist das Netzwerk deshalb so konzipiert, dass möglichst jede Komponente redundant vorhanden und von der Gegenkomponente abgeschottet ist. Das Netzwerk gliedert sich in zwei Subnetze auf, die exakt gleich aufgebaut sind. Die Subnetze sind über ein Gateway miteinander verbunden.³⁶ Das Gateway stellt zudem die Schnittstelle zum Internet dar.

Um die Umgebung vor Angreifen zu schützen, wird pro Subnetz eine DMZ (Demilitarisierte Zone) aufgebaut. Eine DMZ stellt eine Schnittstelle zwischen dem internen Netzwerk und dem Internet dar. In der DMZ siedeln sich alle Server an, die von außen zugänglich sein sollen (Webserver, öffentlicher FTP-Server, etc.). Die DMZ wird von zwei Firewalls abgesichert. Die eine Firewall trennt das Internet von der DMZ und die andere Firewall die DMZ von dem internen Netzwerk. Somit entsteht die Isolierung vom Internet zum internen Netzwerk.

Hinter der DMZ befinden sich die Server für die Xen-Umgebung, aufgeteilt in jeweils drei Subnetze: „Storefront Subnet“, „XA/XD Subnet“ und „Worker Subnet“.

Das Subnetz „Storefront Subnet“ enthält den StoreFront-Server. Der StoreFront ist zur Sicherheit durch ein Subnetz abgeschottet, da es sich um eine Webapplikation handelt.

Im „XA/XD Subnet“ Subnetz siedeln sich die Hauptkomponenten für die Xen-Umgebung an: der Delivery-Controller und der SQL-Server. Diese werden virtualisiert bereitgestellt. Hierfür kann ein beliebiger Hypervisor (wie z.B. HyperV von Microsoft) genutzt werden. Der Delivery-Controller erhält zudem den nötigen Zugriff auf das Active-Directory des Unternehmens.

Im „Worker Subnet“ befindet sich Citrix Hypervisor XenServer. Die XenServer sind mit einer NVIDIA GRID Karte ausgestattet. Auf ihnen laufen die virtuellen Desktops, Applikationen und die Host-VMs, die die Container bereitstellen.

In der DMZ sind zwei NetScaler auf jeweils einem physischen Server angesiedelt. Die NetScaler dienen als Gateway für die Anfragen an die Xen-Ressourcen, wie die Xen-Desktops und Xen-Apps. Des Weiteren teilt der NetScaler die Last der Anfragen an die zwei Netzwerke auf.

³⁶ Das Active-Directory wird in der Konzeptzeichnung nicht in Hinsicht auf Ausfallsicherheit berücksichtigt, da angenommen wird, dass bei einer praktischen Umsetzung ein Unternehmen bereits über ein hochverfügbares Active-Directory verfügt und dieses bereitstellt. In der eigentlichen Implementierung im nächsten Schritt wird es jedoch ausfallsicher bereitgestellt.

Zudem sind zwei Bastion-Host-Server in der DMZ angesiedelt. Diese dienen als zentraler Einstiegspunkt für die Administration der Umgebung vom Internet aus. Auf sie kann per SSH oder RDP zugegriffen werden. Von den Bastion-Hosts aus können wiederum alle Instanzen des lokalen Netzwerks per SSH oder RDP erreicht werden.

6.1.2.2 Umsetzung der Netzwerkarchitektur in AWS

Um die Netzwerkumgebung möglichst authentisch und ohne Hardwarekosten umsetzen zu können, wurde „Amazon Web Services“ (AWS) verwendet. Amazon Web Services ist eine Cloud-Computing Plattform die von Amazon.com angeboten wird. Neben „Microsoft Azure“ zählt AWS zu einer der größten Plattformen für Cloud-Computing. AWS stellt verschiedene Cloud-Services zur Verfügung, mit denen Server, Netzwerke, Datenbanken, etc. in einer Cloud abgebildet werden können.³⁷

Abbildung 12 zeigt, wie die Testumgebung in Anlehnung an die zuvor erstellte Netzwerkarchitektur mit Hilfe von AWS erstellt wurde. Da die VMs von AWS bereits auf einem Hypervisor laufen und Amazon das Aufsetzen eines weiteren Hypervisors auf deren Hypervisor (auch Cascading genannt) verbietet, wurde auf einen externen Server der Hypervisor „XenServer“ installiert und per VPN (Virtual Private Network) mit dem „Worker 1 Subnet“ Subnetz verbunden.³⁸ Der XenServer dient für die Bereitstellung der virtuellen Maschinen, also auch den Host-VMs auf denen die Docker-Container laufen. In den „Worker“ Subnetzen können zudem virtuelle Maschinen (AWS VMs) über den AWS-Hypervisor erstellt werden, falls dies nötig ist.

AWS verfügt über sogenannte „Security-Groups“, die für jede Instanz vergeben werden können. Die Security-Group ist im Endeffekt eine Sammlung an Firewall-Regeln, die den Traffic zu einer oder mehreren Instanzen regelt. Somit kann man die Security-Group auch als eine Art „virtuelle Firewall“ bezeichnen. Dabei können eine oder mehrere Security-Groups einer Instanz zugewiesen werden. Jede Instanz der Testumgebung wurde dabei mit diesen „Firewall-Regeln“ versehen. Demnach musste in die AWS-Umgebung keine eigene Firewall integriert werden.³⁹

³⁷ AMAZON WEB SERVICES, INC., Was ist AWS? - Amazon Web Services (2016). <https://aws.amazon.com/de/what-is-aws/>. (17.11.2016)

³⁸ DESROCHER, Running Hyper-V VMware or Xen on an AWS EC2 Instance? - CloudStacking.com (wie Anm. 25)

³⁹ AMAZON WEB SERVICES, INC., Security Groups for Your VPC - Amazon Virtual Private Cloud (2016).

http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_SecurityGroups.html. (07.12.2016)

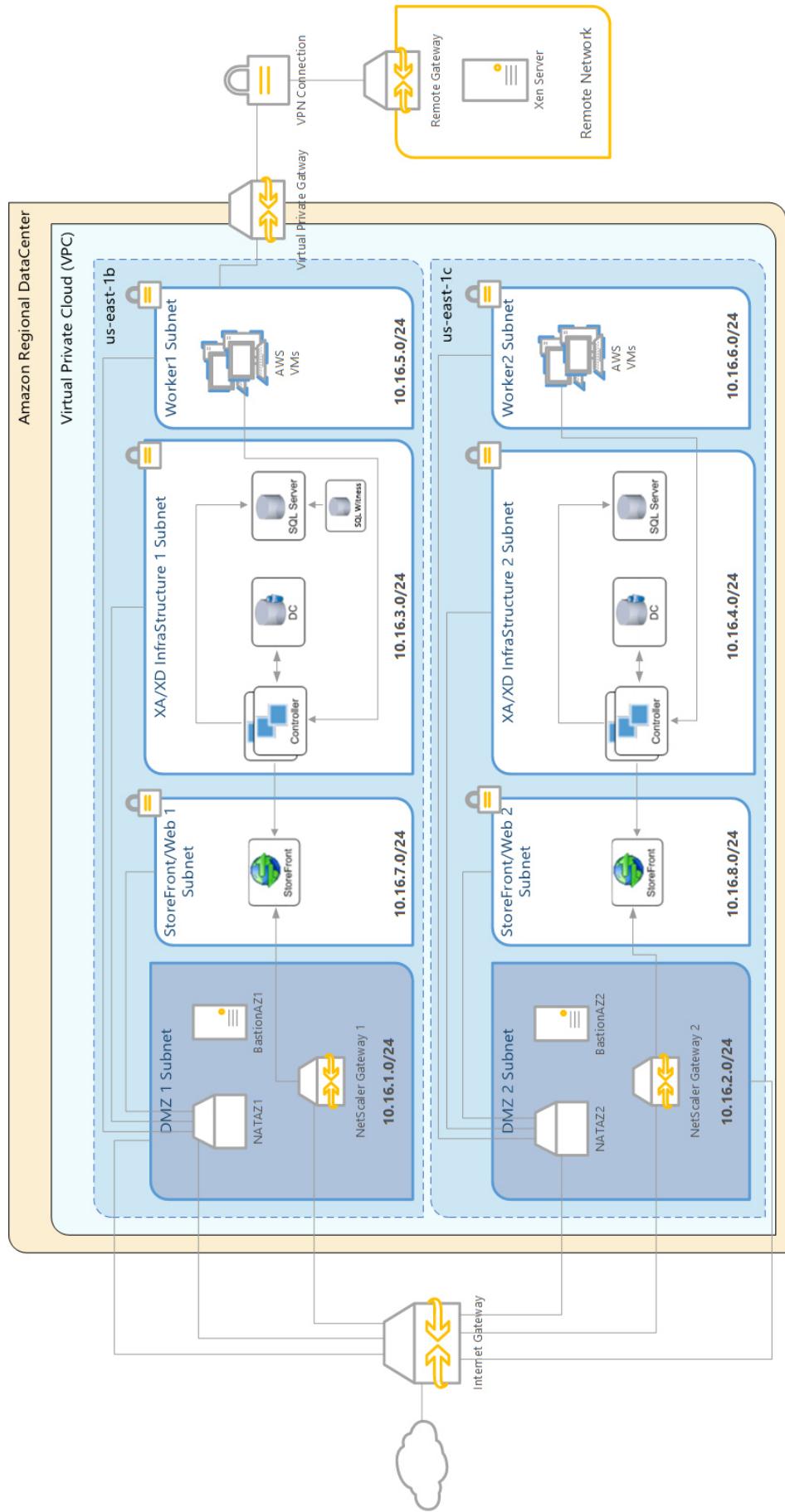


Abbildung 12: Die Testumgebung in AWS

6.1.3 Komponenten

6.1.3.1 Virtual Private Cloud

Die Amazon VPC ist ein logischer, isolierter, Netzwerkbereich, der mit einem eigenen IP-Adressbereich belegt werden kann.⁴⁰ Die VPC ist das Grundgerüst für die Testumgebung. Alle Ressourcen werden innerhalb dieser VPC angelegt.

Innerhalb der VPC wurden zwei Availability-Zonen (us-east-1b und us-east-1c) angelegt, in denen die verschiedenen Subnetze erstellt werden. Die zwei Availability-Zonen befinden sich in derselben Region wie die VPC (US East - North Virginia), sind aber voneinander isoliert und nur durch low-latency Links miteinander verbunden (siehe Abbildung 13).⁴¹

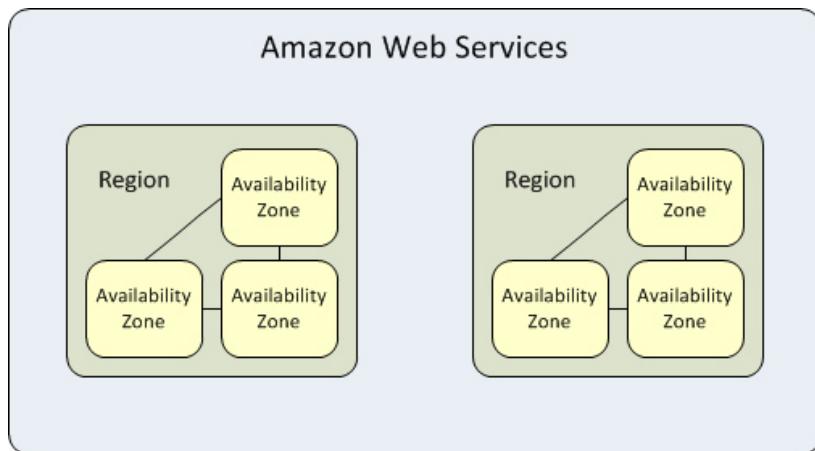


Abbildung 13: Regionen und Availability Zonen Konzept (AMAZON WEB SERVICES, INC. 2016a)

Durch die Duplikation der Komponenten in die zwei Availability-Zonen kann das Netzwerk ausfallsicher und somit redundant konzipiert werden. Fällt eine Komponente innerhalb einer Zone aus, übernimmt automatisch eine Zwillingsskomponente in der anderen Zone ihre Aufgaben. Im schlimmsten Fall übernehmen, bei einem Ausfall einer kompletten Zone, die Ressourcen der anderen Zone.

6.1.3.2 Internet-Gateway

Um die Subnetze und damit die Server, die nachher innerhalb der Subnetze angelegt werden, mit dem Internet zu verbinden, wird ein Internet-Gateway benötigt. Das Internet-Gateway stellt zum einen Route-Tabellen für den Internet-Traffic bereit und fungiert zum

⁴⁰ AMAZON WEB SERVICES, INC., AWS | Amazon Virtual Private Cloud (VPC) & VPS Hosting (2016). <https://aws.amazon.com/de/vpc/>. (28.09.2016)

⁴¹ AMAZON WEB SERVICES, INC., Regions and Availability Zones - Amazon Elastic Compute Cloud (2016a).

<http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.html>. (28.09.2016)

anderen als NAT und übersetzt die Adressen der Instanzen, denen eine öffentliche IP-Adresse zugewiesen wurde. Es stellt also eine Schnittstelle zum Internet bereit.

Um einen Server innerhalb eines Subnetzes mit dem Internet zu verbinden, benötigt dieser zusätzlich zu seiner privaten IP-Adresse, die sich im IP-Adressbereich des Subnetzes befindet, eine öffentliche IP-Adresse. Das Internet-Gateway führt dabei eine eins-zu-eins NAT durch, bei der der ausgehende Internet-Traffic der Instanz anstelle der privaten IP-Adresse, die öffentliche IP-Adresse als Adresse zugewiesen wird.

6.1.3.3 NAT-Gateway

Das Problem bei einer direkten Verbindung zum Internet-Gateway mit einer öffentlichen IP-Adresse ist, dass die Server von außen direkt erreicht werden können. Dies stellt eine große Sicherheitslücke dar. Deswegen werden nur Server, die auch von außen erreicht werden sollen, wie z.B. Webserver, mit einer öffentlichen IP-Adresse versehen.

Um dieses Problem zu beseitigen und den Servern, die von außen nicht erreichbar sein sollen, trotzdem eine Verbindung zum Internet zu ermöglichen, wird ein NAT-Gateway in der DMZ bereitgestellt.

Dadurch wird durch Routing ermöglicht, private (engl. private) und öffentliche (engl. public) Subnetze zu erstellen. Unter Routing versteht man das Festlegen des Weges, den ein Nachrichtenstrom eines Servers, PCs, etc. nehmen soll. Während das Routing des öffentlichen Subnetzes den Internet-Traffic direkt zum Internet-Gateway weiterleitet, wird der Internet-Traffic des privaten Subnetzes erst an das NAT-Gateway weitergeleitet. Abbildung 14 zeigt, wie das Routing für private und öffentliche Subnetze mit einem NAT-Gateway umgesetzt werden kann. Die „Elastic IP“ ist hier die öffentliche IP-Adresse.

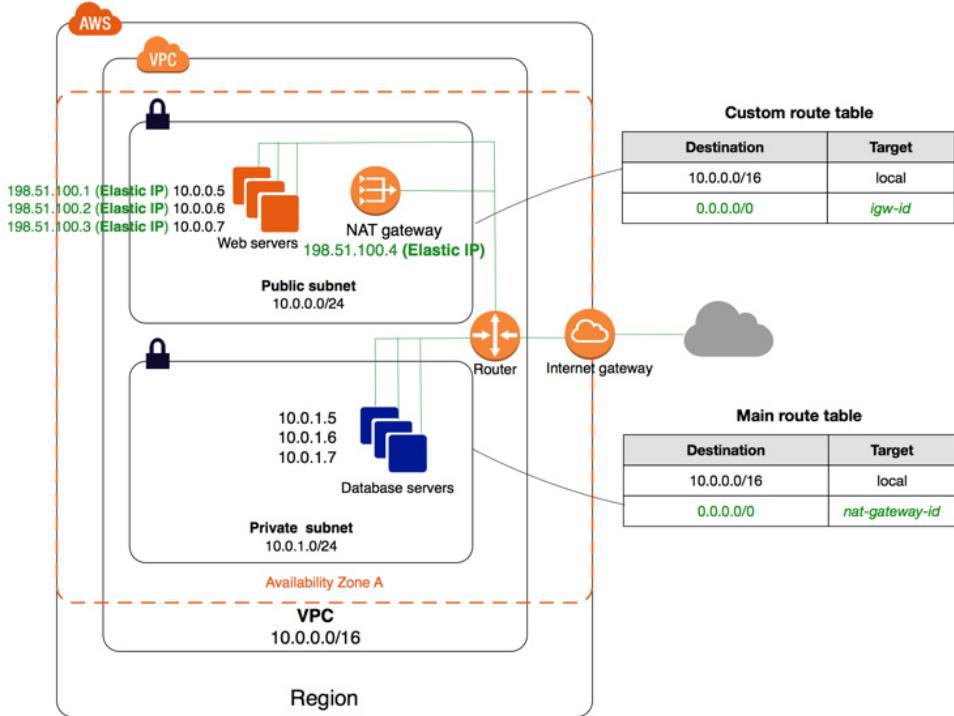


Abbildung 14: Beispiel einer VPC mit privaten und öffentlichen Subnetzen (AMAZON WEB SERVICES, INC. 2016b)

Das NAT-Gateway leitet den Traffic weiter an das Internet-Gateway. Die NAT-Instanz ermöglicht somit eine sichere Verbindung vom Internet zu den Instanzen im privaten Subnetz (zum Beispiel, um Windows Updates empfangen zu können) verhindert aber, dass eingehende Verbindungen über das Internet direkt an die Instanzen innerhalb des privaten Subnetzes weitergeleitet werden.

Bei der Testumgebung wurden insgesamt zwei öffentliche Subnetze eingerichtet:

- DMZ 1 Subnet
- DMZ 2 Subnet

Das Routing der anderen Subnetze wurde so eingerichtet, dass der Internet-Traffic zunächst zum NAT-Gateway weitergeleitet wird. Somit werden diese Subnetze zu privaten Subnetzen, die von außen nicht direkt erreicht werden können. In Abbildung 12 sind die privaten Subnetze mit einem Schloss gekennzeichnet

6.1.3.4 Domain-Controller

Für die Authentifizierung von Benutzern, Computern und Servern wird ein Domain-Controller benötigt (in Abbildung 12 mit „DC“ abgekürzt). Mit dem Domain-Controller können Computer und Server innerhalb einer Domain zusammengefasst werden und somit die Zugriffsrechte auf diese Computer verwaltet werden. Das Tool für diese Verwaltung

ist das Active-Directory (AD). Es enthält alle Einträge zu Benutzern, Computern, etc. Diese Einträgen können dann Rechte für die Authentifizierung vergeben werden.

Für die XenApp- und XenDesktop-Umgebung dient das Active-Directory als zentraler Punkt der Authentifizierung für den Zugriff auf die virtuellen Ressourcen und ist deshalb ein wichtiger Bestandteil der Xen-Umgebung.

6.1.3.5 Bastion-Host

Wie bereits erwähnt, dienen die Bastion-Hosts, zwei Windows 2012 R2 Server, für die Remote-Administration der Umgebung. Von ihnen aus kann jeder Server in den privaten Subnetzen erreicht werden. In jedem der öffentlichen Subnetze befindet sich jeweils einer dieser Server. Die Bastion-Server bekommen eine öffentliche IP-Adresse zugewiesen und die Ports für eine RDP-Verbindung werden freigeschaltet. Somit können die Bastion-Hosts vom Internet aus über eine RDP-Session erreicht werden.

6.1.3.6 SQL-Server

Citrix benötigt für die Xen-Site insgesamt drei SQL-Server-Datenbanken, die mit den Delivery-Controllern kommunizieren:

- **Site Datenbank:** Sie speichert die aktuelle Site-Konfiguration. Dazu gehören der Status der Sessions und Verbindungsinformationen.
- **Configuration-Logging Datenbank:** Sie loggt Änderungen und administrative Aktivitäten der Site mit.
- **Monitoring Datenbank:** Sie speichert Daten, die von dem CitrixDirector benutzt werden, wie zum Beispiel Session- und Verbindungsinformationen. Der CitrixDirector ist ein webbasiertes Tool, mit dem die Xen-Site und deren Ressourcen überwacht werden können. Er liefert unter anderem Informationen zu den Xen-Apps und XenDesktops sowie den User-Sessions. Über ihn lassen sich die virtuellen Ressourcen und sogar einzelne Dienste der Host-Betriebssysteme starten, stoppen, neustarten und noch vieles mehr.

Für eine Hochverfügbarkeitsumgebung mit automatischem Failover existieren mehrere Strategien:⁴²

- **SQL-Server Database-Mirroring:** Beim Mirroring wird eine Datenbank 1:1 auf eine andere Datenbank, die sich auf einem zweiten Server befindet, gespiegelt. Bei

⁴² CITRIX SYSTEMS INC., Databases (2016).

[\(02.10.2016\)](https://docs.citrix.com/en-us/xenapp-and-xendesktop/7-8/technical-overview/databases.html)

jeder Änderung der primären Datenbank, wird die zweite, sekundäre Datenbank auch angepasst. Bei einem Ausfall einer der Datenbanken übernimmt die andere Datenbank automatisch die eingehenden Anfragen an die Datenbank.

- **SQL-Clustering:** Microsoft SQL-Clustering funktioniert ähnlich wie das Database-Mirroring, bei dem ein Server automatisch die Aufgaben und Verantwortung des anderen Servers, der ausgefallen ist, übernimmt. Jedoch ist diese Methode aufwändiger und langsamer als das klassische Mirroring, da mehrere Storages und Server eingerichtet werden müssen.
- **AlwaysOn Availability-Groups:** Diese Strategie basiert auf einem Windows-Server Failover-Cluster (WSFC) mit mehreren Nodes und ist eine Alternative zum klassischen Database-Mirroring. Dabei werden einige primäre Datenbanken erstellt, die von bis zu acht weiteren, sekundären Datenbanken unterstützt werden.⁴³

Entgegen der offiziellen Server-Architektur Referenz von Citrix für Amazon AWS, bei der AlwaysOn Availability-Groups genutzt werden, wurde bei dieser Umgebung Database-Mirroring verwendet, da es ressourcensparender zu konfigurieren ist. In der Testumgebung wurde dazu in jeder Availability-Zone ein Microsoft Windows 2012 R2 („SQL1“ und „SQL2“) in das „XA/XD Infrastructure Subnet 1“ und das „XA/XD Infrastructure Subnet 2“ Subnetz eingebunden. Zusätzlich wurde im „XA/XD Infrastructure Subnet 1“ noch ein weiterer SQL-Server („SQL3“) erzeugt, der als „Witness“ dient (siehe Abbildung 15). Der Witness-Server entscheidet, ob ein automatisches Failover initiiert werden soll, oder nicht. Er verfügt jedoch nicht über eine Datenbank, sondern ist lediglich für das Failover verantwortlich.⁴⁴

⁴³ MICROSOFT, Always On Availability Groups (SQL Server) (2016).
<https://msdn.microsoft.com/en-us/library/hh510230>. (02.10.2016)

⁴⁴ MICROSOFT, Database Mirroring Witness (2016).
<https://msdn.microsoft.com/en-us/library/ms175191.aspx>. (03.10.2016)

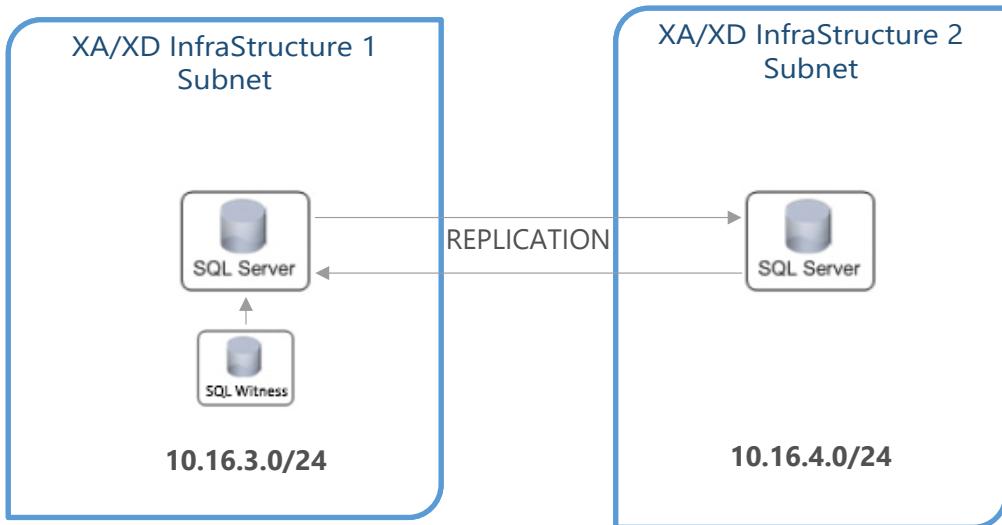


Abbildung 15: Database-Mirroring in der Testumgebung

6.1.3.7 Citrix Komponenten

6.1.3.7.1 Delivery-Controller

Der Delivery-Controller (in Abbildung 12 „Controller“ genannt) ist die zentrale Management-Komponente einer jeden XenApp oder XenDesktop Site. Der Delivery-Controller kommuniziert mit dem Hypervisor, um Applikationen und Desktops bereitzustellen, authentifiziert User über das Active Directory, dient als Broker zwischen den Usern und den virtuellen Ressourcen, optimiert Verbindungen und load-balances diese Verbindungen.

Insgesamt wurden zwei Delivery-Controller in jeweils einer der beiden Availability Zonen installiert, um die Redundanz und damit die Ausfallsicherheit zu gewährleisten. Neben dem Delivery-Controller werden auch das CitrixStudio, der CitrixDirector und der Lizenzserver auf dem Server installiert.

Der Lizenz-Server kommuniziert mit dem Delivery-Controller und verwaltet die Lizenzen, die für die Site gebraucht werden.

Das CitrixStudio ist eine Management-Konsole, mit dem die Site konfiguriert und gemanagt werden kann. Dabei ermöglicht die Konsole unter anderem das Hosten von Applikationen und Desktops und das Zuweisen dieser Ressourcen an User.

Im Citrix-Studio sind die beiden Delivery-Controller im Verbund sichtbar (siehe Abbildung 16).

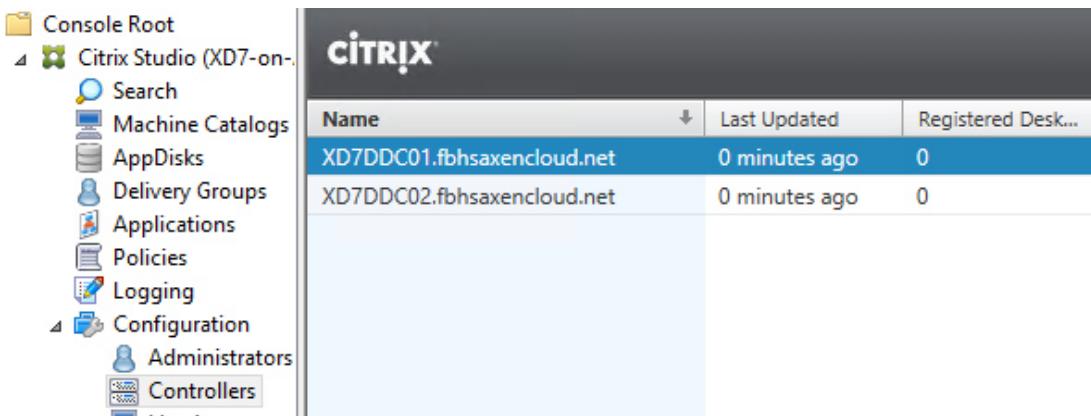


Abbildung 16: Ansicht im Citrix Studio nach erfolgreicher Installation

6.1.3.7.2 StoreFront-Server

Der StoreFront-Server authentifiziert die User zu ihrer zugehörigen Site und managet die virtuellen Desktops und Applikationen, auf die die User zugreifen. Er hostet eine Web-Applikation, namens StoreFront, die über den Browser aufgerufen werden kann und über dessen Web-Interface die User ihre zugewiesenen Xen-Ressourcen sehen und starten können.

In jeder Availability-Zone wird jeweils ein StoreFront-Server angelegt. Nach der Installation muss die StoreFront-Web-Applikation über die Citrix StoreFront-Management-Konsole, konfiguriert werden. Die Konsole wird mit dem StoreFront auf dem Server installiert. Da der StoreFront eine Webanwendung ist, benötigt er für die Einrichtung eine URL, über die der StoreFront erreichbar sein soll.

Nach dem Einrichten lässt sich der StoreFront dann über einen Browser oder mit dem Citrix-Receiver öffnen:

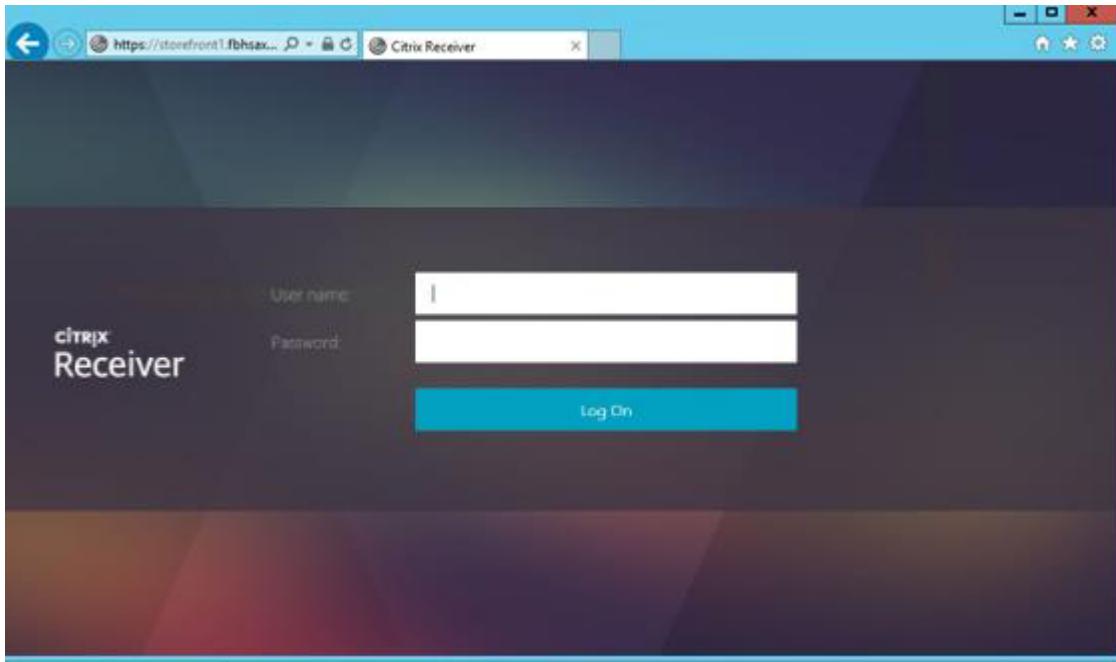


Abbildung 17: Anmeldemaske des StoreFronts

6.1.3.7.3 NetScaler

Um den Zugriff von außen zu ermöglichen, wird der NetScaler von Citrix benötigt. Der NetScaler wird im Allgemeinen als Web-Applikation-Delivery-Controller bezeichnet und hat mehrere Aufgaben:⁴⁵

1. **Load-Balancing:** Der NetScaler verteilt die einkommenden Anfragen an die zwei Availability-Zonen bzw. die zwei StoreFront-Server.
2. **Sicherer Remote-Zugriff:** Ermöglicht einen Virtual-Private-Network (VPN) Zugriff mittels Secure-Socket-Layers (SSL).
3. **Application-Layer Schutz:** Der NetScaler stellt eine Applikation-Firewall bereit, die die Web-Anwendungen vor Bedrohungen schützt.
4. **Erweiterte Zugriffs- und Aktionskontrolle:** Je nach Benutzeridentität und Endgerät können Regeln zur Weiterleitung festgelegt werden.
5. **NetScaler-Gateway:** Zentraler Punkt der Authentifizierung der User über das Internet. Bei erfolgreicher Authentifizierung wird der StoreFront geladen.

Für das Erstellen der NetScaler-Server stehen bereits vorgefertigte Installationen des NetScalers als sogenannte AMI (Amazon Machine Image) bereit. Um ein Failover zu ermöglichen, werden insgesamt zwei NetScaler installiert. Einer befindet sich in der DMZ

⁴⁵ CITRIX SYSTEMS INC., NetScaler 11.0 (2016).
<https://docs.citrix.com/en-us/netscaler/11.html>. (19.11.2016)

(DMZ 1 Subnet) der ersten Availability-Zone, der andere in der zweiten DMZ (DMZ 2 Subnet) der anderen Availability-Zone.

Nach dem Erstellen der NetScaler-Server sind noch einige Anpassungen nötig. Über die IP-Adresse des NetScaler kann die Admin-Konsole über einen Browser geöffnet werden. Mit Hilfe dieser Konsole können dann die Einstellungen zu SSL, dem Gateway, dem Load-Balancing, etc. vorgenommen werden.

Zudem müssen für jeden NetScaler noch zwei öffentliche IP-Adressen angelegt werden.

Der NetScaler verfügt dann über drei IP-Adressen:

- **NSIP** (NetScaler-IP): Die IP-Adresse des NetScalers (diese wurde während der Installation der NetScaler zugewiesen und befindet sich im Adressraum des Subnetzes).
- **SNIP** (Subnet-IP): Die IP-Adresse für die Server-Verbindungen. Die SNIP wird gebraucht, um Internet-Traffic von oder durch den NetScaler an das Subnetz zu senden, das mit dem NetScaler verbunden ist.
- **VIP** (Virtual-IP): Die öffentliche IP-Adresse, die für die Clients sichtbar ist. Mit dieser IP-Adresse erreichen die User das NetScaler-Gateway.

IP Addresses you *must* configure

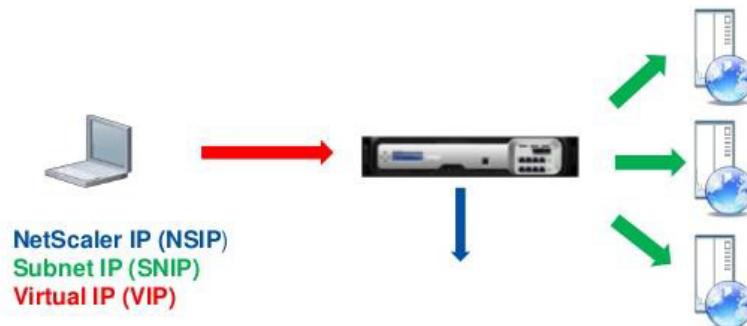


Abbildung 18: IP-Adressen des NetScalers (SLIDEShare.NET)

Nach einer erfolgreichen Konfiguration lässt sich der StoreFront nun über die Virtual-IP-Adresse außerhalb der Umgebung erreichen:

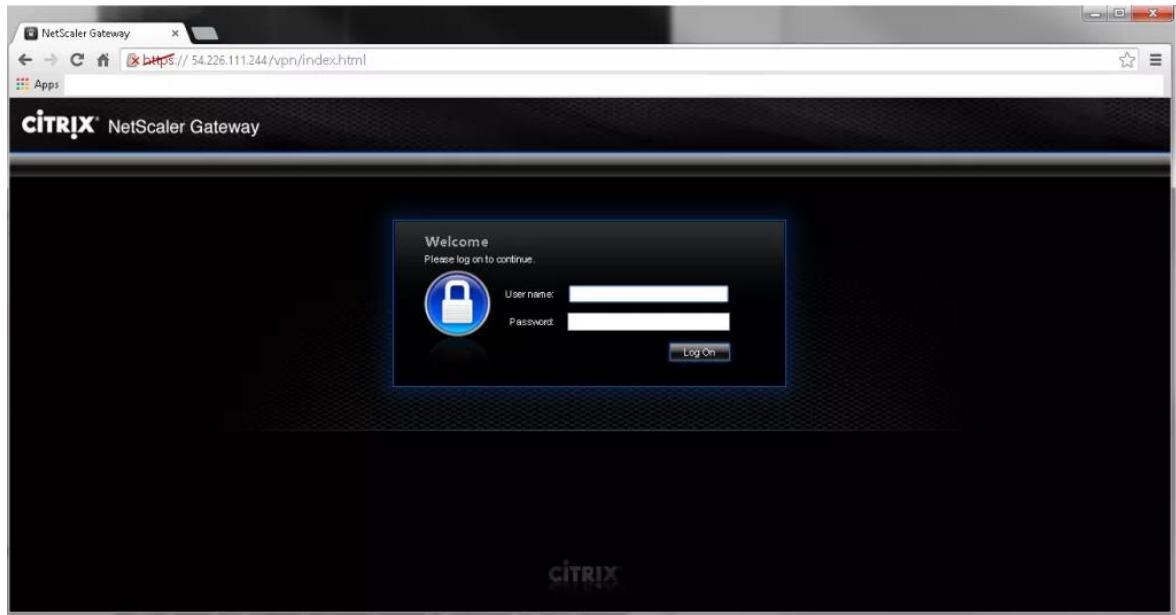


Abbildung 19: NetScaler-Gateway über die öffentliche IP-Adresse

Für den Aufruf des Gateways über eine URL, anstelle einer IP-Adresse würde man eine Domain benötigen und diese bei einer Zertifizierungsstelle registrieren lassen. Für die Testumgebung reicht ein Erreichen der virtuellen Ressourcen über eine öffentliche IP-Adresse.

6.1.4 Umsetzung

Die Umsetzung der Testumgebung erfolgte fast ausschließlich über AWS CloudFormation. Mit AWS CloudFormation können Entwickler einfach Templates für Server- und Netzwerkumgebungen erstellen, die automatisiert ausgerollt werden können. Für die Implementierung des Netzwerks wurde ein CloudFormation-Template in Anlehnung an die offizielle Anleitung von Citrix angepasst. (BATS 2015)

Die Templates werden im JSON-Format angelegt und können zudem durch den Amazon AWS CloudFormation Designer als Diagramm visualisiert werden (siehe Abbildung 20).⁴⁶

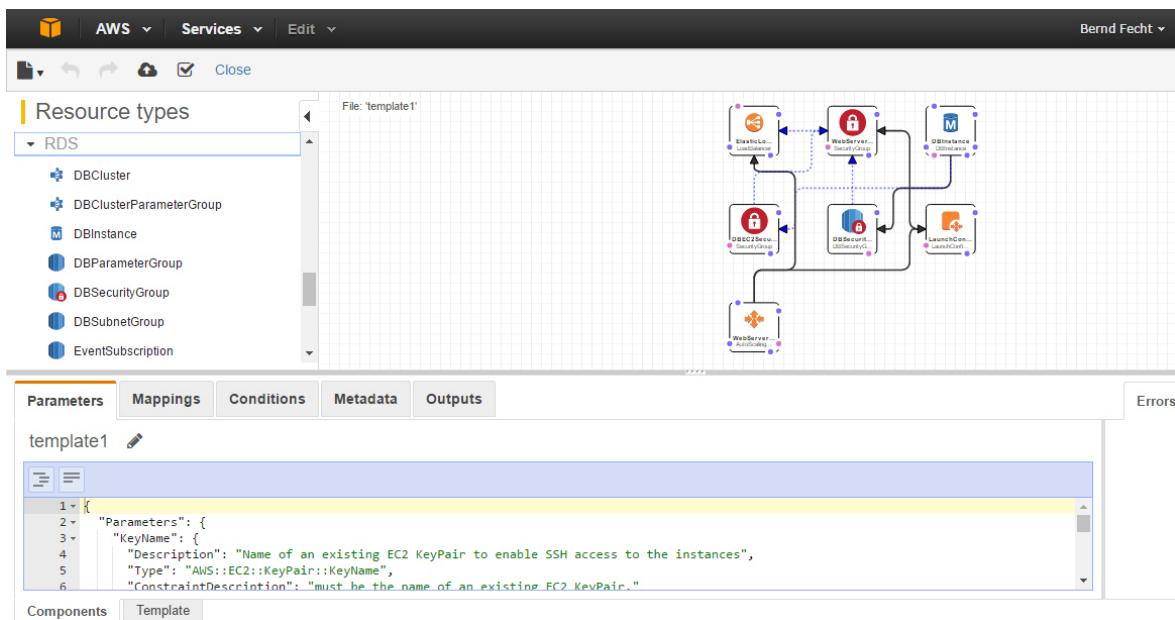


Abbildung 20: AWS CloudFormation Designer

Neben der Möglichkeit, AWS-Ressourcen zu erstellen und zu verwalten, bietet AWS CloudFormation auch die Möglichkeit, Anweisungen für die Installation und Konfiguration der Ressourcen innerhalb des Templates mit zu übergeben. Somit können neben der Erstellung einer virtuellen Maschine mit einem Betriebssystem auch die Installation von weiterer Software oder eine Anpassung der Einstellungen des Betriebssystems durch beispielsweise im Template definierte Power-Shell-Befehle automatisiert werden.

⁴⁶ AMAZON WEB SERVICES, INC., AWS CloudFormation – Infrastruktur als Code und AWS-Ressourcenbereitstellung (2016).

<https://aws.amazon.com/de/cloudformation/>. (17.11.2016)

Listing 2 enthält einen gekürzten Auszug aus der JSON-Datei, der aufzeigt, wie die JSON-Notation für die Erstellung der VPC aussieht.

```
{  
    "AWSTemplateFormatVersion": "2010-09-09",  
    "Description": "Abstract Template for VPC Creation",  
    "Parameters": {  
        "VPCCIDR": {  
            "Description": "CIDR Block for the VPC",  
            "Type": "String",  
            "Default": "10.16.0.0/16",  
            "AllowedPattern": "[a-zA-Z0-9]+\.\.+"  
        },  
        "VPCName": {  
            "Default": "XenApp - XenDesktop 7.x VPC",  
            "Description": "The name of the Virtual Private Cloud",  
            "Type": "String"  
        }  
    },  
    "Resources": {  
        "VPC": {  
            "Type": "AWS::EC2::VPC",  
            "Properties": {  
                "CidrBlock": {  
                    "Ref": "VPCCIDR"  
                },  
                "Tags": [  
                    {  
                        "Key": "Application",  
                        "Value": {  
                            "Ref": "AWS::StackName"  
                        }  
                    },  
                    {  
                        "Key": "Network",  
                        "Value": "Public"  
                    },  
                    {  
                        "Key": "Name",  
                        "Value": {  
                            "Ref": "VPCName"  
                        }  
                    }  
                ]  
            }  
        }  
    }  
}
```

Listing 2: JSON-Template für das Erstellen einer VPC

Für die VPC werden insgesamt zwei Parameter definiert, einerseits „VPCCIDR“, der den CIDR-Block der VPC als String einliest und andererseits „VPCName“, der den Namen für die VPC als String einliest. CIDR ist die Abkürzung für Classless Inter-Domain Routing und ist eine Teilung einer IP-Adresse mittels einer Subnetzmaske oder eines CIDR-Suffix.

Als Adressraum wird somit die 10.16.0.0/16 vergeben. Die eigentliche Erstellung der VPC erfolgt im „Ressources“-Teil des Templates.

Das fertige Template wird dann über das CloudFormation Tool der AWS Management Console importiert. Beim Aufruf des Wizards sieht man die im Template definierten Parameter, die an dieser Stelle noch bearbeitet bzw. vervollständigt werden können (siehe Abbildung 21).

Specify Details

Specify a stack name and parameter values. You can use or change the default parameter values, which are defined in the AWS CloudFormation template. [Learn more](#).

| | |
|---|--|
| Stack name | <input type="text"/> |
| Parameters | |
| KeyPairName | <input type="text"/> |
| Public/private key pairs allow you to securely connect to your instance after it launches | |
| VPCCIDR | <input type="text" value="10.16.0.0/16"/> |
| CIDR Block for the VPC | |
| VPCName | <input type="text" value="XenApp - XenDesktop 7.x VPC"/> |
| The name of the Virtual Private Cloud | |
| <input type="button" value="Cancel"/> <input type="button" value="Previous"/> <input type="button" value="Next"/> | |

Abbildung 21: Erstellung eines Stacks über AWS CloudFormation

Jegliche Ressourcen, die innerhalb eines Templates angegeben wurden, werden in einem sogenannten „Stack“ abgelegt und in der AWS Management Console angezeigt (siehe Abbildung 22). Auf diese Weise lassen sich die Ressourcen geordnet und automatisiert erstellen.

| CloudFormation ▾ Stacks | | | |
|--------------------------------|------------------------------|-------------------------|---|
| Create Stack | | Actions | Design template |
| Filter: Active ▾ By Stack Name | | Showing 4 stacks | |
| Stack Name | Created Time | Status | Description |
| XenSF | 2016-10-10 11:23:27 UTC+0200 | CREATE_COMPLETE | **Version 1.00, 10-10-2016** STEP-SF This |
| XenDCC | 2016-10-08 18:31:58 UTC+0200 | CREATE_COMPLETE | **Version 1.00, 08-10-2016** STEP-XD Thi |
| XenSQL | 2016-10-08 16:00:13 UTC+0200 | CREATE_COMPLETE | **Version 1.0, 08-10-2016** STEP-SQL Thi |
| XenCloud | 2016-10-08 13:03:29 UTC+0200 | CREATE_COMPLETE | **Version 1.0, 08-10-2016** STEP-AD Thi |

Abbildung 22: Stack-Ansicht über AWS CloudFormation

6.1.5 Funktionsweise

Um zu verstehen, wie die Komponenten der Xen-Site funktionieren, muss man sich den Vorgang der Authentifizierung und die Anfrage und Bereitstellung der virtuellen Ressourcen genauer ansehen. Abbildung 23 dient als Referenz für die einzelnen durchnummerierten Schritte.⁴⁷

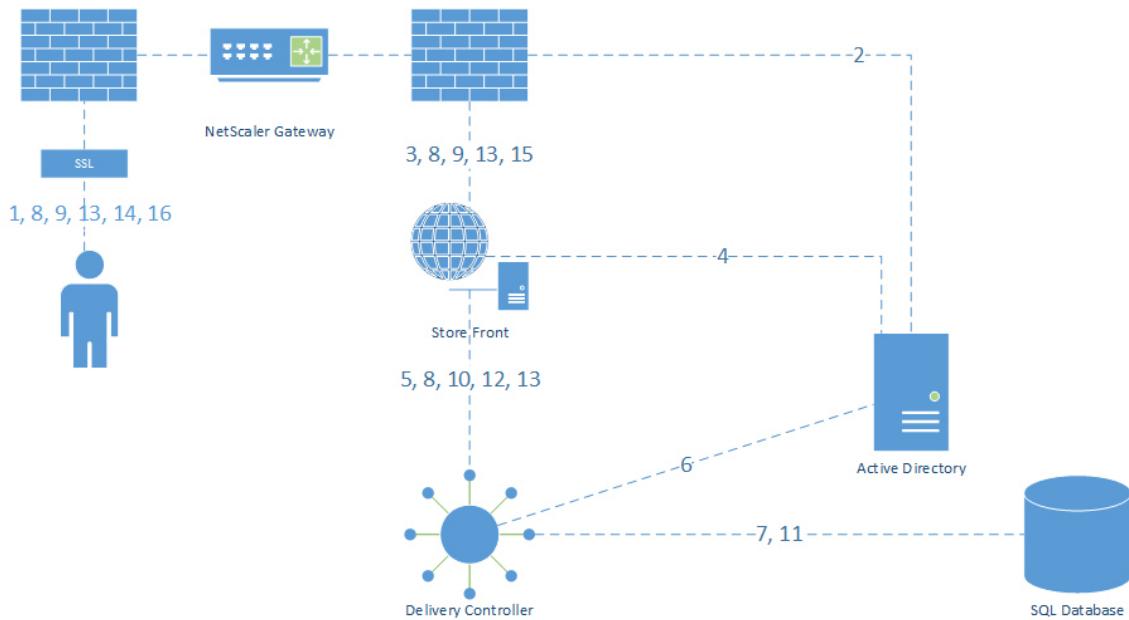


Abbildung 23: Ablauf eines Verbindungsbaus

Authentifizierungsvorgang

1. Ein User initialisiert eine Verbindung zum NetScaler Gateway mit seinen Login-Credentials. Hierfür kann er entweder den Citrix Receiver oder einen Browser verwenden. Der Receiver ist das Client-Programm, mit dem die Verbindung zu den virtuellen Ressourcen hergestellt werden kann.
2. Das NetScaler Gateway baut eine Verbindung zum Active-Directory auf und prüft die Credentials.
3. Das NetScaler Gateway leitet die validierten User-Credentials an den StoreFront weiter.

Bereitstellung

4. Der StoreFront authentifiziert den User noch einmal über das Active Directory, prüft, ob User-Einstellungen in der Datenbank des StoreFronts vorhanden sind, und lädt diese in den Speicher.

⁴⁷ Daniel FELLER, XenApp 7 Deployment Blueprint,Citrix XenDesktop 7 - Blueprint (2013)

5. Der StoreFront leitet die User-Credentials weiter zum Delivery-Controller.
6. Der Delivery-Controller validiert den User wiederholt über das Active Directory.
7. Nach der Validierung überprüft der Delivery-Controller, welche Ressourcen dem User zur Verfügung stehen, indem er die SQL-Datenbank abfragt.
8. Eine Liste der verfügbaren Ressourcen wird an den StoreFront gesendet, der dann die Ressourcen (nachdem sie über das NetScaler Gateway gesendet wurden) im Receiver oder Browser des Users darstellt.

Verbinden mit einer virtuellen Ressource

9. Wenn der User eine Ressource ausgewählt hat, die er starten will, wird die Anfrage über das NetScaler Gateway an den StoreFront gesendet.
10. Der Storefront leitet die Anfrage an den Delivery-Controller weiter.
11. Der Delivery-Controller sucht in der SQL-Datenbank nach dem passenden Host, der die Anfrage erfüllen kann.
12. Der Delivery-Controller sendet die Informationen des Hosts und der Verbindung an den StoreFront.
13. Der StoreFront fragt die Secure Ticket Authority (kurz STA) auf dem Delivery-Controller an, ein Ticket für den User auszustellen, das die angefragte Ressource, die Server-Adresse und die Port-Nummer enthält. Dieses Ticket wird dann in ein Launch-File verpackt und an den User über das NetScaler Gateway gesendet.
14. Der Citrix Receiver benutzt dieses Launch-File um die Verbindung zu der Ressource über das NetScaler Gateway aufzubauen.
15. Das NetScaler Gateway validiert die Gültigkeit des Tickets über die STA.
16. Das NetScaler Gateway stellt eine Verbindung zu der virtuellen Ressource her.

6.1.6 Testen der Xen-Umgebung

Nachdem die Xen-Umgebung fertig eingerichtet ist, muss überprüft werden, ob die Komponenten so zusammenarbeiten, wie sie sollen. Dazu wird ein beliebiger virtueller Windows-Server auf dem XenServer erstellt und in die Domain aufgenommen. Nach der Erstellung wird der Virtual-Delivery-Agent (VDA) von Citrix auf der VM installiert. Der VDA registriert sich bei den Delivery-Controllern und dient als Kommunikationsschnittstelle der VM zu der Xen-Infrastruktur.

Nach der Installation ist der Server bereit, seine Anwendungen und den Desktop virtualisiert für die User bereit zu stellen. Dazu muss im CitrixStudio zunächst ein Maschinenkatalog angelegt werden, der den Server über das Active-Directory in den Maschinenpool

aufnimmt (siehe Abbildung 24). Ein Maschinenkatalog ist eine Sammlung an virtuellen Maschinen, die einer bestimmten Benutzergruppe zugewiesen werden können.

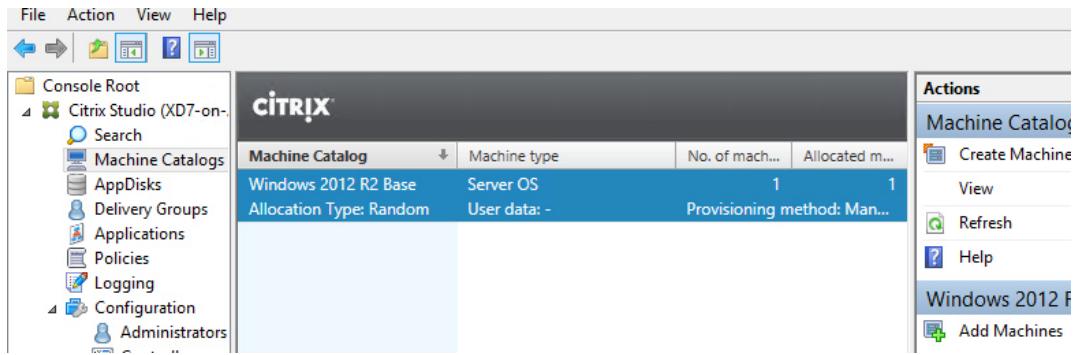


Abbildung 24: Maschinenkatalog im Citrix Studio

Um den Maschinenkatalog für die User verfügbar zu machen, benötigt man weiterhin eine sogenannte Delivery-Group. Innerhalb dieser Delivery-Group wird geregelt, welche User Zugriff auf welche Ressourcen des Servers haben. Die Ressourcen können dabei Applikationen oder der Desktop selbst sein. Ist alles richtig konfiguriert, können die Applikationen (XenApps) direkt aus dem Start-Menü des Servers zur Delivery-Group hinzugefügt werden (siehe Abbildung 25).

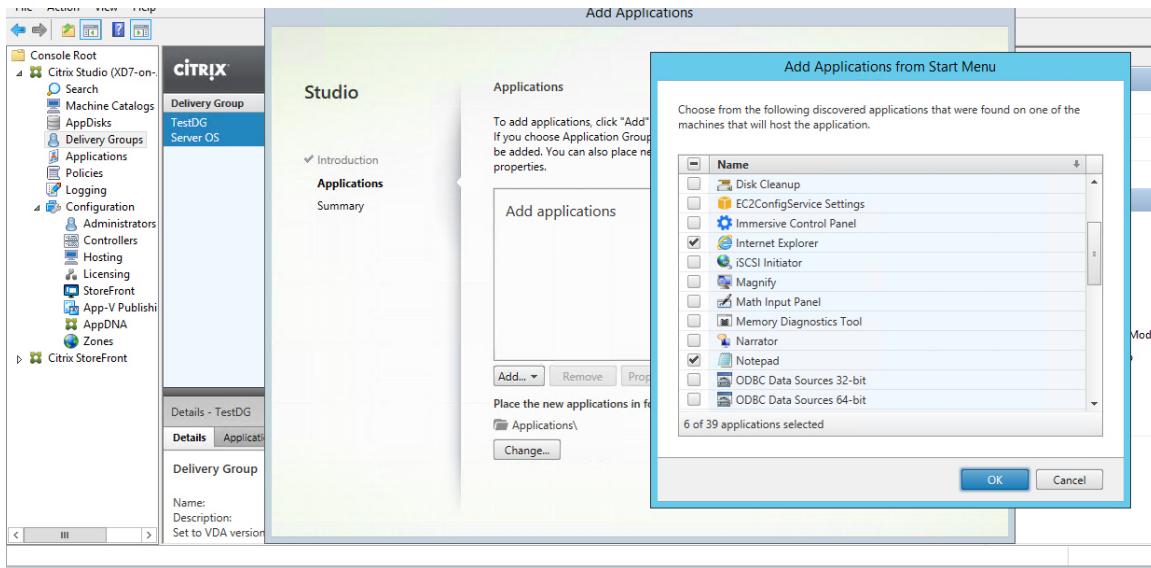


Abbildung 25: Hinzufügen der XenApps über das Start-Menü des Host-Servers

Loggt man sich nun auf den StoreFront der Xen-Site (entweder über die öffentliche IP-Adresse des NetScaler-Gateways oder über die interne Adresse) ein, sollten die Xen-Apps, die vorher aus dem Startmenü hinzugefügt wurden, sichtbar sein (siehe Abbildung 26). Durch einen einfachen Mausklick lassen diese sich dann starten.

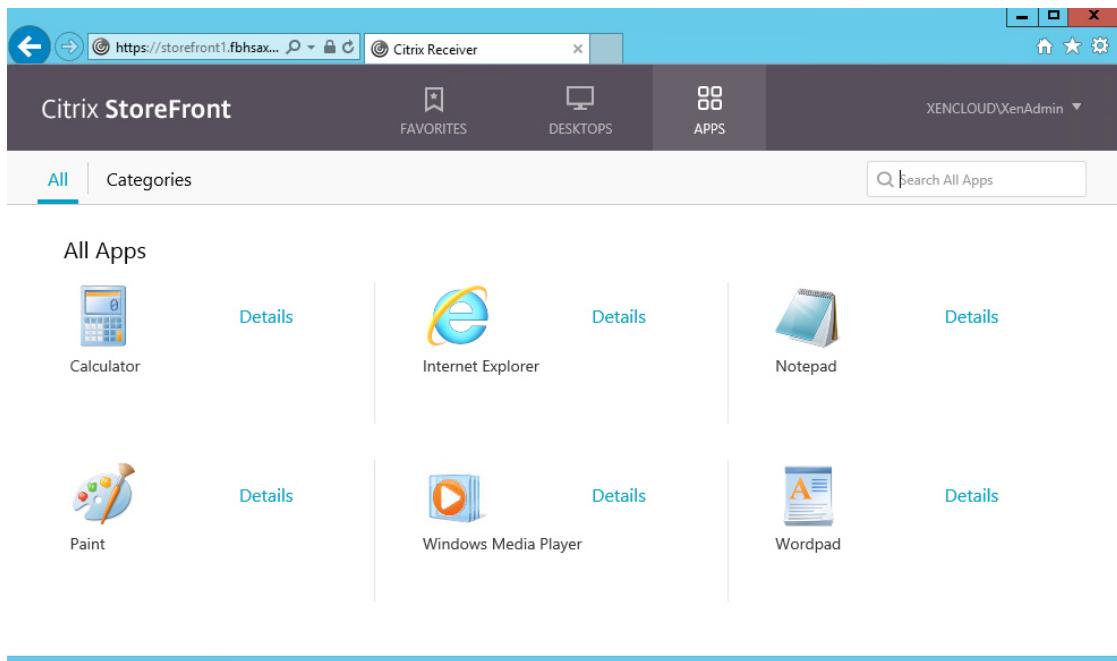


Abbildung 26: Apps stehen nach erfolgreicher Einrichtung auf dem StoreFront bereit

Die Xen-Umgebung ist somit vollständig aufgesetzt und für die Bereitstellung der Docker-Container fertig konfiguriert.

6.2 Bereitstellung von Containern auf dem XenServer

Nachdem die Testumgebung erstellt wurde, ist der nächste Schritt das Bereitstellen von Docker-Containern auf dem per VPN angebundenen XenServer, um das Durchreichen der vGPU zu testen und die Container in die virtuelle Umgebung integrieren zu können.

Die Docker-Container laufen auf einer Host-VM, die auf dem XenServer gehostet wird. Die Erstellung der VM erfolgt über einen Wizard des XenCenters. Das XenCenter ist die Verwaltungskonsole des XenServers. Als Host-Betriebssystem für die Docker-Container dient in diesem Fall ein Ubuntu 14.04. Dabei können für das Betriebssystem die Standardeinstellungen genutzt oder auch angepasst werden. Nach der Erstellung der VM über den Wizard startet die VM automatisch und das Betriebssystem kann installiert werden.

Die Installation von Docker erfolgt über einen Paketmanager. Wichtig ist es, nicht das Paket zu installieren, dass mit Ubuntu ausgeliefert wird, sondern dass offizielle Docker Paket. Dafür muss das Paketmanagement Tool APT unter Ubuntu so eingerichtet werden, dass es Pakete von der Docker-Repository verwendet. Erst dann kann über `apt-get install docker-engine` Docker auf der Ubuntu-VM installiert werden.

Um die Installation der Docker Komponenten zu testen, muss ein Docker-Container auf der Host-VM gestartet werden. Dazu wird eine SSH-Verbindung zu der VM aufgebaut, in der Docker läuft und mit folgendem Befehl ein Container erstellt:

```
docker run -d --name docker_hello_world ubuntu /bin/sh -c "while true; do echo Hello World; sleep 1; done"
```

Mit diesem Befehl wird ein Docker-Container mit dem Namen „docker_hello_world“ als Daemon gestartet. -d ist dabei das Flag, das den Container im Hintergrund laufen lässt (also als Daemon startet). Als Basis-Image dient Ubuntu. Docker erkennt automatisch, dass das Image nicht lokal auf dem Rechner verfügbar ist und lädt das Image aus dem Docker-Hub herunter. Über /bin/sh wird die Shell des Ubuntu-Images aufgerufen. -c "while true; do echo Hello World; sleep 1; done" ist das Kommando, das beim Starten des Containers in der Shell ausgeführt werden soll.

Im Endeffekt macht der Container nichts anderes, als ständig „Hello World“ auszugeben. Bei dem Absetzen des Befehls erhält man jedoch zunächst nicht die Ausgabe des Containers, sondern nur einen String, der den Container eindeutig identifiziert. Dieser String ist die Container-ID.

```
fechbern@DockerMaster:~$ docker run -d --name docker_hello_world ubuntu /bin/sh -c "while true; do echo Hello World; sleep 1; done"
f999d93d2a5debbbc9cdc2148cc622a5af08ffd0703c1c836e2b85b8817f24aa
```

Durch einen Aufruf von `docker logs docker_hello_world` wird die eigentlich Ausgabe des Containers angezeigt, in diesem Fall eben „Hello World“.

Der Container läuft jetzt auf der Host-VM, die auf dem XenServer aufgesetzt wurde und ist auch nur über diese VM zu steuern. Eine Integration in die virtuelle Umgebung ist zu diesem Zeitpunkt somit noch nicht erfolgt.

6.3 Docker-Container mit GPU-Unterstützung

6.3.1 NVIDIA Docker

Ein Vorteil von Docker-Containern ist, dass sie plattform- und hardwareunabhängig sind. Dadurch lassen sie sich leicht von einem System auf ein anderes System transferieren und laufen auf jedem System gleich. Dies wird jedoch genau dann zum Problem, wenn spezielle Hardware, wie zum Beispiel Grafikkarten, innerhalb der Container genutzt werden sollen.

Grafikkarten brauchen spezielle Kernel-Module und Bibliotheken auf Benutzerebene. Diese werden über einen Treiber installiert. Dabei hat jeder Grafikkartentyp seine eigene Treiberversion, die sich von den anderen Grafikkartentreibern unterscheiden. Daher unterstützt Docker nativ keine GPUs in Containern. Als Lösungsansatz für dieses Problem wurden bisher die Treiber im Container mitinstalliert und dann die entsprechende Grafikkarte beim Starten des Containers mitübergeben. Jedoch ist diese Lösung nicht mit dem Konzept der Portabilität von Docker-Containern vereinbar, denn die Treiber-Version innerhalb des Containers musste exakt der Treiber-Version auf dessen Host entsprechen. Als Konsequenz daraus ergab sich, dass die grafikfähigen Docker-Container nicht geteilt und portiert werden konnten.

Um das Problem der Portabilität von grafikfähigen Containern zu beseitigen, entwickelte NVIDIA „NVIDIA Docker“. NVIDIA Docker ist ein Wrapper, der auf Docker aufbaut und mit dem die „docker“ Kommandozeilenbefehle erweitert werden. Dabei werden die GPU und die nötigen Treiber-Daten, um Code auf der GPU auszuführen, beim Start des Containers in den Container gemountet, aber nicht installiert. Der NVIDIA Treiber befindet sich ausschließlich auf der Host-VM.

Dadurch, dass der Treiber nicht direkt in den Container geladen wird, bleibt die Portabilität, und damit einer der Grundsätze von Docker-Containern, bestehen.

6.3.2 Problemanalyse

Da in dieser Arbeit davon ausgegangen wird, dass ein XenServer mit einer NVIDIA GRID Karte die virtuellen Maschinen bereitstellt, muss zunächst verifiziert werden, ob NVIDIA Docker auch mit einer NVIDIA GRID Karte funktioniert.

NVIDIA Docker wurde für die Verwendung von normalen Grafikkarten konzipiert. Für virtuelle Grafikkarten, wie die NVIDIA GRID Grafikkarte gibt es noch keine Referenzen, ob diese Lösung funktioniert. Denn während normale GPUs nur einen Treiber für das Be-

triebssystem brauchen, braucht die NVIDIA GRID Lösung insgesamt zwei Treiber. Einmal einen Treiber, der auf dem Hypervisor, also dem XenServer läuft und einen anderen Treiber, der auf der Host-VM läuft (vgl. Abbildung 27). Der Treiber auf dem XenServer ermöglicht das Aufteilen der Grafikkarte in die unterschiedlichen vGPU-Profile. Der Treiber auf der Host-VM ist dann der eigentliche Grafikkartentreiber für die vGPU, der auch für die Docker-Container genutzt wird.

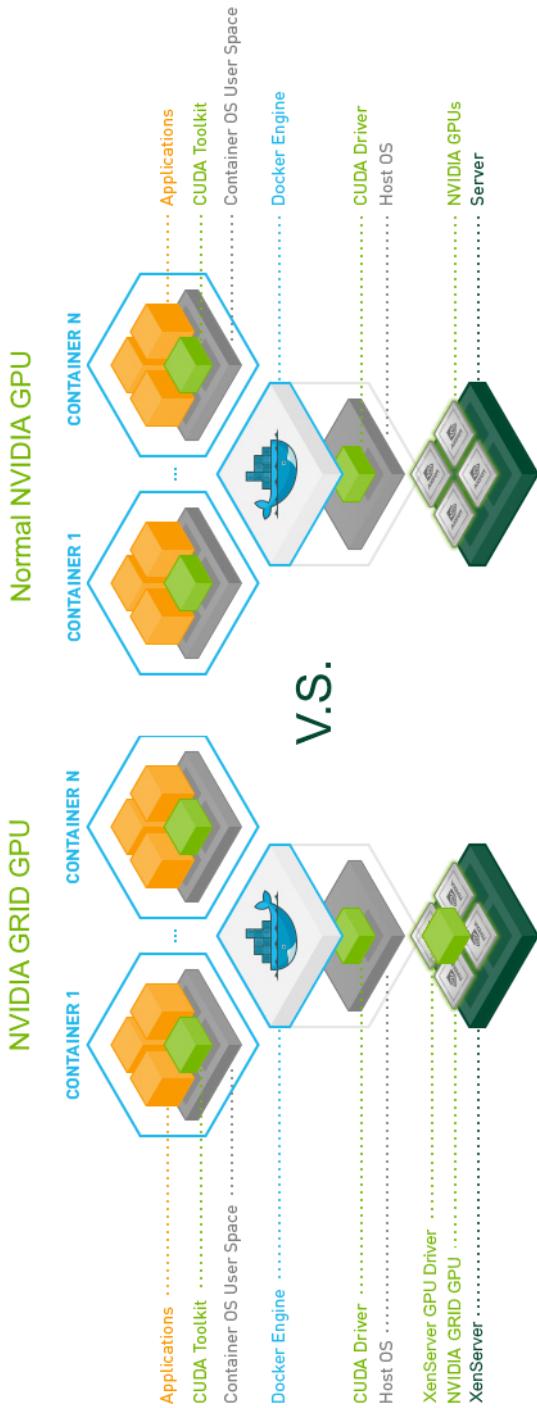


Abbildung 27: Unterschied zwischen einer NVIDIA GRID GPU und einer „normalen“ NVIDIA GPU Architektur mit NVIDIA Docker

Um die Funktionalität von NVIDIA Docker bei NVIDIA GRID-Karten im VDI-Bereich zu testen, stand für diese Arbeit zusätzlich ein physischer XenServer mit einer NVIDIA GRID K1 und GRID K2 Grafikkarte außerhalb der AWS-Umgebung bereit.

6.3.3 Validieren der Lösung für NVIDIA GRID

6.3.3.1 Durchreichen der GPU an die Host-VM

Als erstes wird NVIDIA Docker auf dem XenServer getestet. Dazu muss zunächst der Treiber für den XenServer installiert werden. Das Treiber-Paket für die NVIDIA GRID Karte kann über die Website von NVIDIA heruntergeladen werden. Der Treiber liegt als .rpm-Datei für die Installation auf dem XenServer bereit.

Nach einem Neustart sollte noch verifiziert werden, ob der Treiber richtig installiert wurde. Das erfolgt durch das Kommando „nvidia-smi“. War die Installation erfolgreich, werden Informationen der Grafikkarte und die darauf laufenden Prozesse über dieses Kommando angezeigt. Bei dem in dieser Arbeit zur Verfügung gestellten XenServer sind gleich zwei GRID Karten, die GRID K1 und GRID K2 verbaut.

```
[root@localhost ~]# nvidia-smi
Thu Nov  3 15:16:53 2016
+-----+
| NVIDIA-SMI 367.43          Driver Version: 367.43
+-----+
| GPU  Name      Persistence-M | Bus-Id      Disp.A  | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap| Memory-Usage | GPU-Util  Compute M. |
|=====+=====+=====+=====+=====+=====+=====+=====|
| 0  GRID K1           On           0000:86:00.0  Off    |          N/A |
| N/A  36C     P8    10W / 31W |   8MiB / 4095MiB | 0%       Default |
+-----+
| 1  GRID K1           On           0000:87:00.0  Off    |          N/A |
| N/A  34C     P8    10W / 31W |   8MiB / 4095MiB | 0%       Default |
+-----+
| 2  GRID K1           On           0000:88:00.0  Off    |          N/A |
| N/A  26C     P8    10W / 31W |   8MiB / 4095MiB | 0%       Default |
+-----+
| 3  GRID K1           On           0000:89:00.0  Off    |          N/A |
| N/A  29C     P8    10W / 31W |   8MiB / 4095MiB | 0%       Default |
+-----+
| 4  GRID K2           On           0000:8C:00.0  Off    |          Off |
| N/A  43C     P8    29W / 117W |   8MiB / 4095MiB | 0%       Default |
+-----+
+-----+
| Processes:                               GPU Memory |
| GPU  PID  Type  Process name             Usage      |
+-----+
| No running processes found               |
+-----+
```

Nachdem der Treiber auf dem XenServer installiert wurde, kann die virtuelle Grafikkarte im XenCenter über die Einstellungen der VM zugewiesen werden. Nach dem Durchreichen ist die zugewiesene virtuelle Grafikkarte im XenCenter sichtbar:

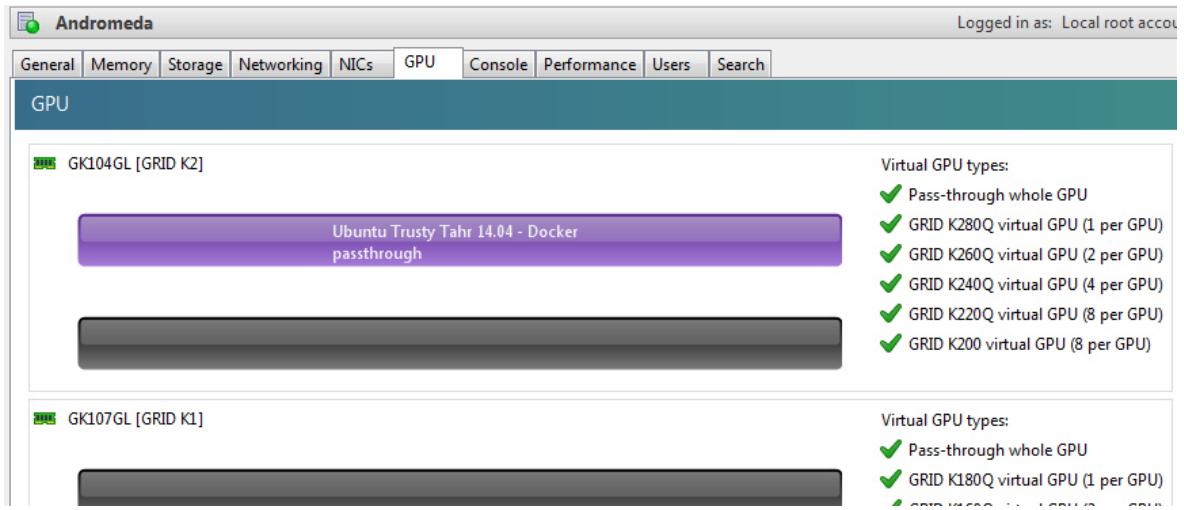


Abbildung 28: Informationen der vGPU im XenCenter

Noch kann die virtuelle Grafikkarte nicht in der Host-VM verwendet werden, da diese nicht über den vGPU-Treiber verfügt. Der nächste Schritt ist daher die Installation des Treibers. Der Linux-Treiber für die virtuelle NVIDIA GRID Karte kann ebenfalls über das NVIDIA Portal heruntergeladen werden und steht als .run-Datei zur Verfügung.

Bei der Installation müssen folgende Aspekte beachtet werden:

- Der Nouveau Treiber muss deaktiviert sein. Der Nouveau-Treiber ist ein freier Treiber für NVIDIA Grafikkarten unter Linux.⁴⁸ Die Installation des NVIDIA Treibers ermöglicht jedoch diese Deaktivierung, indem ein modprobe Konfigurationsfile erstellt wird. Modprobe ermöglicht es, Module des Linux-Kernels hinzuzufügen oder zu entfernen.⁴⁹ Nach der Erstellung muss die Host-VM neu gestartet und die .run-Datei neu ausgeführt werden.
- Die GNU Compiler Collection „gcc“ muss installiert sein⁵⁰
- Das Build-Management-Tool „make“ muss installiert sein⁵¹

Nach der Installation lässt sich der Erfolg ebenfalls mit dem Kommando nvidia-smi überprüfen. Die Host-VM verfügt jetzt über eine virtuelle Grafikkarte und steht somit bereit, die Docker-Container mit NVIDIA Docker zu testen.

⁴⁸ nouveau (2016).

<https://nouveau.freedesktop.org/wiki/>. (03.11.2016)

⁴⁹ modprobe(8): add/remove modules from Kernel - Linux man page (2016).

<https://linux.die.net/man/8/modprobe>. (03.11.2016)

⁵⁰ GCC, the GNU Compiler Collection - GNU Project - Free Software Foundation (FSF) (2016).

<https://gcc.gnu.org/>. (03.11.2016)

⁵¹ Make (2016).

<http://www.selflinux.org/selflinux/html/make01.html>. (03.11.2016)

6.3.3.2 Testen von grafikfähigen Docker-Containern mit NVIDIA Docker

Nachdem die NVIDIA-Treiber für den XenServer und die Host-VM installiert wurde, kann nun NVIDIA Docker installiert und getestet werden. Vorausgesetzt, der NVIDIA Treiber und Docker wurden richtig installiert, lässt sich NVIDIA Docker über wget herunterladen und installieren.

Wie bereits erwähnt, erweitert NVIDIA Docker die Docker Kommandos. Ein Container lässt sich jetzt anstelle von „docker run“ mit „nvidia-docker run“ starten. Durch einen Aufruf von nvidia-docker run -rm nvidia/cuda nvidia-smi wird ein auf Ubuntu basierter Container gestartet, dessen Image mit Anweisungen für die Installation von CUDA erweitert wurde.

CUDA (Compute Unified Device Architecture) wurde von NVIDIA entwickelt und stellt eine Programmietechnik für C und C++ bereit, mit der Teile eines Programmes parallelisiert auf der GPU ausgeführt werden können.⁵²

Das Kommando nvidia-smi gibt, wie bereits erwähnt, die verfügbaren Grafikkarten aus. In diesem Fall wird die zugewiesene NVIDIA GRID K2 Grafikkarte des XenServers ausgegeben. Damit wurde erfolgreich getestet, dass die Grafikkarte und der Grafikkartentreiber in den Container gemounted wurden.

Ob ein Programmcode auf der GPU vom Container aus ausgeführt werden kann wurde jedoch noch nicht überprüft. Um dies zu testen, werden einige Beispielprogramme aus dem CUDA SDK innerhalb eines NVIDIA Docker-Containers ausgeführt. Das erste Programm prüft, ob eine CUDA-fähige Grafikkarte vorhanden ist.

```
./deviceQuery Starting...

CUDA Device Query (Runtime API) version (CUDART static linking)

Detected 1 CUDA Capable device(s)

Device 0: "GRID K2"
[...]

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 7.5, CUDA Runtime
Version = 7.5, NumDevs = 1, Device0 = GRID K2
Result = PASS
```

Der Output des Programmes ergibt, dass die GRID K2 für CUDA Berechnungen bereitsteht.

⁵² NVIDIA, CUDA Toolkit.

<https://developer.nvidia.com/cuda-toolkit>. (22.11.2016)

Ein weiterer Test mit einem CUDA Programm, das eine Vektor-Addition auf der GPU mit 5000 Vektoren ausführt, validiert, ob Berechnungen auf der GPU ausgeführt werden:

```
./vectorAdd

[Vector addition of 50000 elements]
Copy input data from the host memory to the CUDA device
CUDA kernel launch with 196 blocks of 256 threads
Copy output data from the CUDA device to the host memory
Test PASSED
Done
```

Das Durchreichen der vGPU an den Container hat funktioniert. Somit wurde validiert, dass NVIDIA Docker auch für Systeme mit NVIDIA GRID Karten genutzt werden kann.

6.4 Integration von Docker-Containern auf dem XenServer

Nachdem erfolgreich GPU-fähige Container erstellt wurden, ist der nächste Schritt die Integration der Container in die virtuelle Umgebung. Seit XenServer 6.5 SP1 unterstützt XenServer auch die Integration von Docker-Containern und deren Administration über das XenCenter. Diese Integration, die von Citrix „Container-Management“ genannt wird, ist der erste Schritt der Integration der Docker-Container in die virtuelle Umgebung.

6.4.1 Vorbereiten des XenServers

Zunächst muss auf dem XenServer das „Supplemental-Pack“ für die Docker-Integration installiert werden. Supplemental-Packs sind einfache Patches, die die Funktionalität des XenServers modifizieren oder erweitern.⁵³ Dazu kann entweder per SSH auf den XenServer zugegriffen werden oder die Konsole über das XenCenter aufgerufen werden. Anschließend muss das Supplemental-Pack runtergeladen und über Xens Kommandozeilen-Interface namens „XAPI“ installiert werden.⁵⁴

```
wget http://downloadns.citrix.com.edgesuite.net/11621/XenServer-7.0.0-xsc
xe-install-supplemental-pack XenServer-7.0.0-xsc.iso
```

6.4.2 Einrichten des Container-Managements

Eine Voraussetzung für die Installation des Container-Managements von Citrix sind die XenServer-Tools. Die XenServer-Tools stellen einen Management-Agenten auf der VM bereit, mit dem unter anderem folgende Operationen über das XenCenter möglich sind:⁵⁵

- Sauberes Herunterfahren, Neustarten oder Anhalten der VM
- Anzeigen von Performance Daten
- Migration einer laufenden VM
- Erstellen und Wiederherstellen von Snapshots
- Ändern der Anzahl von vCPUs einer laufenden Linux VM (Windows benötigt einen Neustart)

⁵³ Citrix XenServer 7.0 Supplemental Packs & the DDK. Citrix Systems (2016)

⁵⁴ XEN PROJECT, XAPI Command Line Interface - Xen (2014).

https://wiki.xen.org/wiki/XAPI_Command_Line_Interface. (14.10.2016)

⁵⁵ CTRIX SYSTEMS INC., Installing XenServer Tools (2016).

<http://docs.citrix.com/fr-fr/xencenter/6-5/xs-xc-vms-configuring/xs-xc-vms-installtools.html>. (14.10.2016)

Ob die Installation der XenServer-Tools erfolgreich war, kann man im XenCenter an den zusätzlichen Start- und Stopoptionen und Performance-Graphen sehen.⁵⁶

Um das Container-Management zu aktivieren, braucht der XenServer zunächst einen Eintrag in der Datenbank, der beschreibt, welche VM über eine Docker-Integration verfügt. Dieser Eintrag wird von einem Daemon genutzt, der die Docker-Informationen von den definierten VMs anholt. Der Daemon benutzt SSH um mit den gemanagten VMs zu kommunizieren. Deshalb wird ein private-public Key ausgetauscht. Während der private Key nur dem Daemon bekannt ist, wird der public Key in jeder VM bereitgestellt, die für das Container-Management registriert werden soll. Den Schlüsselaustausch und die Registrierung in der Datenbank lassen sich über einen einzigen Befehl über die XAPI CLI steuern:

```
xscontainer-prepare-vm -v <vm-uuid> -u <username>
```

Dabei ist der Parameter <username> der Benutzername des Benutzers auf der Host-VM mit Docker-Zugriff und <vm-uuid> eine eindeutige Identifikationsnummer der Host-VM. Die UUID kann einfach über den Befehl xe vm-list abgefragt werden:

```
[root@Andromeda ~]# xe vm-list
uuid ( RO) : 87ed8873-b5e6-b17b-7aca-c6965e9daabd
    name-label ( RW): Ubuntu Trusty Tahr 14.04 - Docker Ready
    power-state ( RO): running
```

⁵⁶ CITRIX SYSTEMS INC., Installing XenServer Tools (2016).
[\(15.10.2016\)](http://docs.citrix.com/fr-fr/xencenter/6-5/xs-xc-vms-configuring/xs-xc-vms-installtools.html)

Nach einer erfolgreichen Installation befindet sich in den Eigenschaften der VM im XenCenter ein neuer Reiter mit dem Namen „Docker Information“:

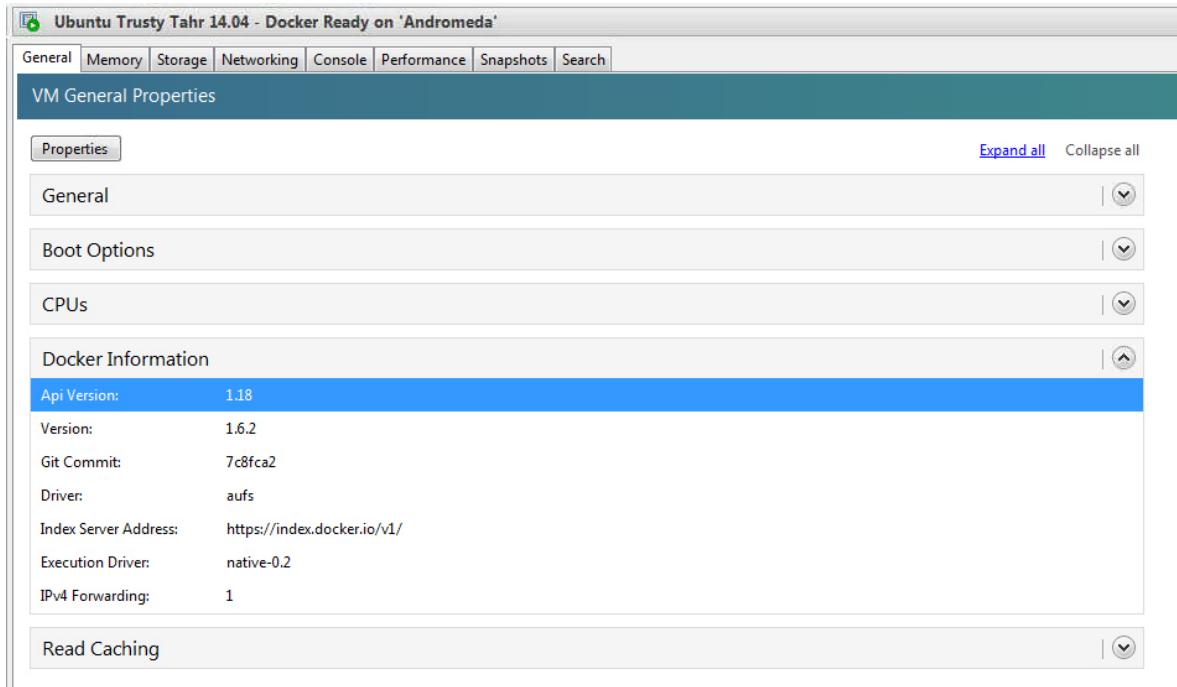


Abbildung 29: Docker Informationen einer VM im XenCenter

6.4.3 Funktionsweise des Container-Managements

Der Nächste Schritt ist, zu untersuchen, wie das Container-Management von Citrix funktioniert.

Im XenCenter sind die Container unter der Host-VM sichtbar und lassen sich starten, stoppen, pausieren und neustarten (siehe Abbildung 30). Zudem können zahlreiche Informationen zu dem Container angezeigt werden, wie laufende Prozesse, die Container-Nummer, Informationen zum Image, Details zum Docker-Container (Netzwerkkonfiguration, Hostname, etc.) und mehr.

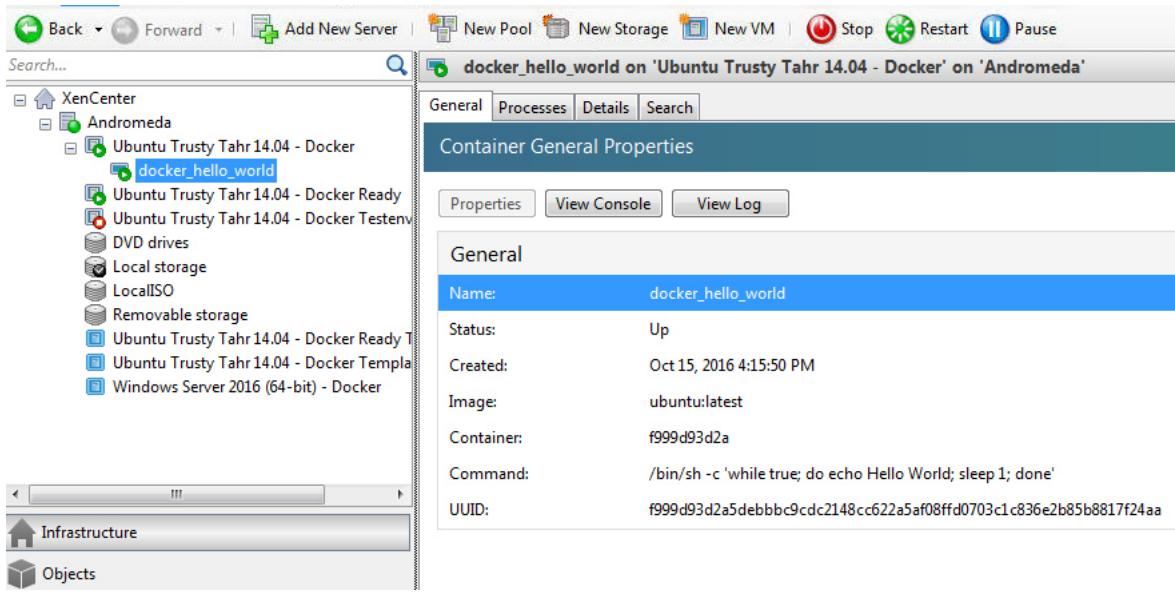


Abbildung 30: Ansicht des Docker-Containers im XenCenter

Der „View Console“ und „View Log“ Button mit denen augenscheinlich eine Konsole bzw. die Logfiles geöffnet werden sollen, hat in dieser Teststellung mit einer XenServer 7.0 Version mit allen Patches nicht funktioniert. Auch das Power-Management der Container sprach diese oft nicht an, so dass Container nicht über das XenCenter gestartet oder gestoppt werden konnten.

Das Container-Management ermöglicht also eine einfachere Wartung der Container und bietet folgende Vorteile:

- **Monitoring:** Es ist sofort ersichtlich, welche VMs die Docker-Integration verwenden und welche Container auf ihnen laufen.
- **Diagnosen:** Informationen über einen Container, wie die verwendeten Ports, der Name des Images usw. können einfach über ein Interface ausgelesen werden.
- **Runtime-Management:** Container können gestartet, beendet, neugestartet und pausiert werden.

Die Integration der Container erfolgt hierbei ausschließlich auf der Ebene des Hypervisors und die Container lassen sich nur über das XenCenter verwalten.

6.4.4 Fazit

Wie bereits erwähnt, ist die native Integration von Citrix eine komfortable Lösung für die Administration oder das Betreiben der Container als „Container-as-a-Service“. Für das Arbeiten mit Containern ist die Lösung jedoch nicht konzipiert.

Eine Möglichkeit, die Container für Entwickler verfügbar zu machen, wäre, das XenCenter als XenApp bereitzustellen, was jedoch zu einem großen Sicherheitsrisiko führen würde, da jeder Entwickler jede erdenkliche, auf dem Server laufende Maschine willkürlich löschen oder virtuelle Maschinen oder Storages erstellen könnte. Auch könnten sie sogar den XenServer (und damit auch alle anderen virtuellen Maschinen) komplett herunterfahren.

Die native Integration von Citrix stößt also hierbei an ihre Grenzen. Um Entwicklern eine komfortable Lösung anzubieten, mit der sie mit Docker-Containern arbeiten können, muss eine eigene Integration geschaffen werden.

6.5 Integration der Docker-Container in den StoreFront

6.5.1 Anforderung an die Integration

Die Integration in die virtuelle Umgebung soll für Software-Entwickler konzipiert werden, die virtuelle Maschinen innerhalb einer XenApp- und XenDesktop-Umgebung für das Programmieren von GPU-unterstützten Anwendungen nutzen. Die Container sollen zentral für die Entwickler über den StoreFront der virtuellen Umgebung bereitgestellt werden. Da die Ausgabe der Programme sowohl grafisch als auch textuell über eine Kommandozeile sein kann, muss für diese beiden Optionen eine geeignete Lösung gefunden werden.

6.5.2 Integration der Container per SSH

6.5.2.1 Architektur

Wie bereits erwähnt, sollen die Docker-Container als XenApps im StoreFront bereitstehen. Diese XenApps sind im Endeffekt eine SSH-Verbindung zu den Containern. Die Host-VM soll ebenfalls als SSH-Verbindung angelegt werden.

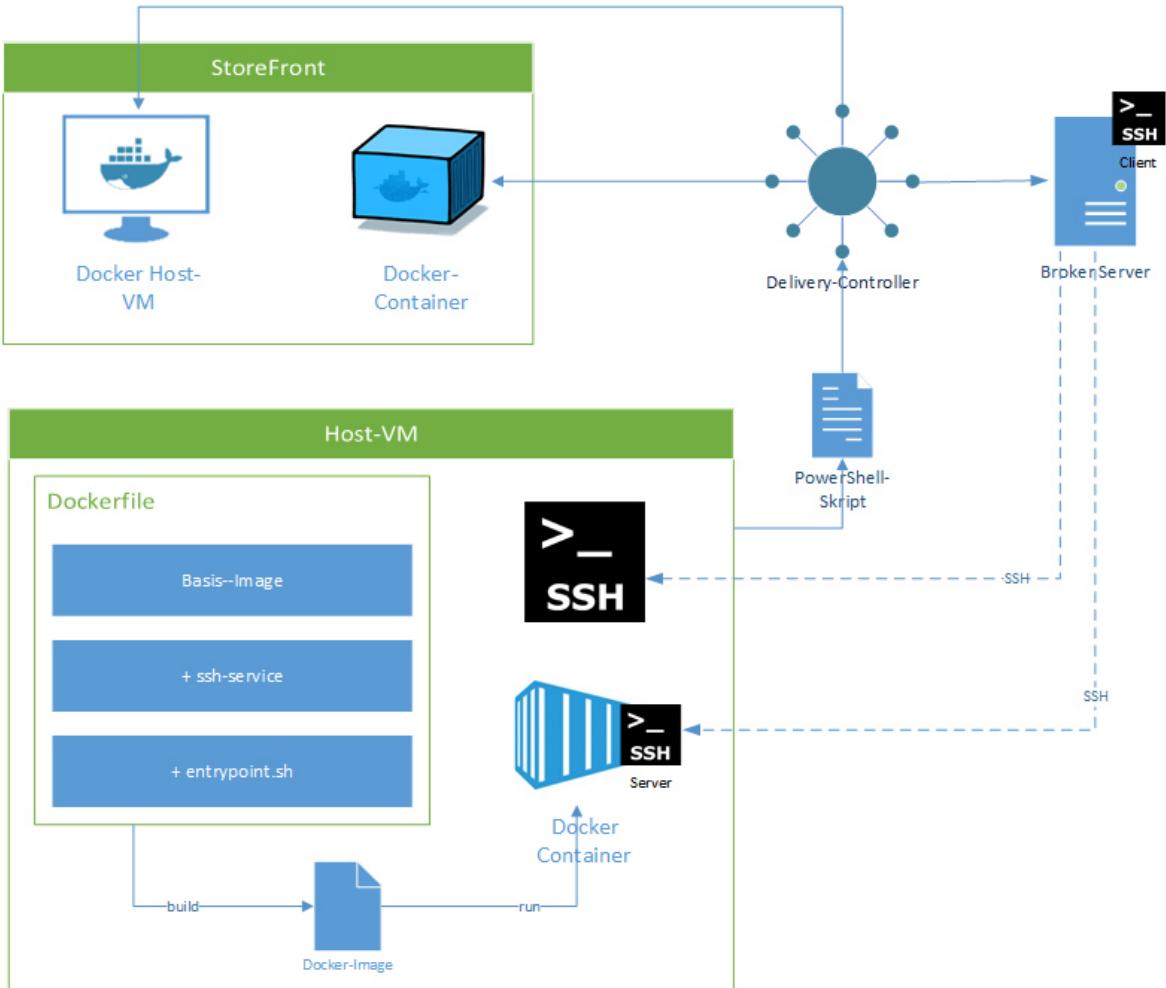


Abbildung 31: Konzept der Integration der Docker-Container als XenApps

Wie in Abbildung 31 zu sehen, muss zunächst ein Dockerfile erstellt werden, das das gewünschte Basis-Image mit der SSH-Komponente erweitert und den SSH-Dienst startet. Aus dem daraus resultierenden Docker-Image werden die Container erstellt. Diese agieren (neben der eigentlichen Aufgabe des Basis-Images) als SSH-Server.

Per Shell-Kommando soll dann eine SSH-Verbindung von der Host-VM zu dem Delivery-Controller aufgebaut werden. Anschließend wird dort ein Installationsskript mit den Parametern über den anzulegenden Docker-Container aufgerufen. Dieses Skript legt dann die XenApp für den Container an.

Da Citrix nur Windows Server Betriebssysteme für die Bereitstellung der XenApps unterstützt, kann deshalb auch nur ein Windows Server für die Integration verwendet werden. Deshalb wird über den XenServer ein Windows 2012 R2 Server erstellt, der als Broker-Server dient. Auf diesem Server steht dann ein SSH-Client (OpenSSH für Windows) bereit.

Die XenApp bedient sich bei dem Broker-Server an dessen SSH-Client, um die Verbindung zu den Containern (oder auch der Host-VM) aufzubauen.

6.5.2.2 Erweitern des Docker-Images mit SSH

Ziel der Integration ist es, jedes erdenkliche „Basis-Image“ für die Container verwenden zu können. Deshalb wird der SSH-Dienst einfach nur an ein bestehendes Image „angeknüpft“ und dann gestartet.

Das Erweitern des Basis-Images erfolgt immer nach dem gleichen Schema:

1. Angabe des Basis-Images
2. Instruktionen mit dem der Programmcode in den Container kopiert wird
3. Installation des SSH-Servers und –Client
4. Anpassen der SSH-Server Konfiguration
5. Freigeben des Ports
6. Starten des SSH-Dienstes

Je nachdem, welches Paketmanagement-Tool verwendet wird (APT, YUM, etc.), unterscheiden sich die Anweisungen für die Installation des SSH-Servers in dem Dockerfile.

Listing 3 zeigt, wie ein Dockerfile mit einem Ubuntu 14.04 Basis-Image mit dem SSH-Server erweitert wird.

```
FROM ubuntu:14.04
MAINTAINER Bernd Fecht "bernd.fecht@hs-augsburg.de"

[...]

# Install OpenSSH and set password

RUN \
    apt-get install -y openssh-client openssh-server && \
    mkdir /var/run/sshd && \
    echo 'root:root' |chpasswd

RUN sed -ri 's/^PermitRootLogin\s+.*$/PermitRootLogin yes/' \
/etc/ssh/sshd_config
RUN sed -ri 's/UsePAM yes/#UsePAM yes/g' /etc/ssh/sshd_config

# Expose Port for SSH usage

EXPOSE 22

[...]

COPY ./ssh.sh /
CMD ["/ssh.sh"]
```

Listing 3: Erweiterung des Dockerfiles mit SSH

Die verwendeten Kommandos sind dabei im Detail:⁵⁷

- **FROM** gibt dabei an, welches Basis-Image verwendet werden soll.
- **MAINTAINER** gibt den Ersteller des Dockerfiles an.
- Mit **RUN** werden die Kommandobefehle ausgeführt, mit denen SSH installiert und konfiguriert wird.
- **EXPOSE** gibt den Port für SSH frei (Port 22).
- **COPY** kopiert die Datei „ssh.sh“-Datei in das Image.
- **CMD** definiert eine Standardoperation, die beim Starten des Containers ausgeführt wird. In diesem Fall wird die „ssh.sh“-Datei aufgerufen

Die „ssh.sh“-Datei enthält Anweisungen, mit denen die SSH-Schlüssel erstellt werden und der SSH-Server gestartet wird:

```
#!/bin/sh

# generate host keys if not present
ssh-keygen -A

# do not detach (-D)
exec /usr/sbin/sshd -D
```

Listing 4: ssh.sh

Der Docker-Container enthält nun einen SSH-Server und ist somit für den Remote-Zugriff per SSH vorbereitet.

6.5.2.3 Registrierung der Container als XenApps

Der Delivery-Controller dient als Verwaltungsserver für die Xen-Site. Mit der Installation des Delivery-Controllers wird ein SDK für das Administrieren der Xen-Site über Powershell von Citrix bereitgestellt. Mit diesem SDK können XenApp und XenDesktop Ressourcen hinzugefügt werden. Dieses SDK wird genutzt, um die XenApps zu erstellen.

Der Programm-Ablauf ist dabei wie folgt:

1. Überprüfen, ob die nötigen Ressourcen für die Integration vorhanden sind. Falls nicht, wird versucht, diese zu erstellen. Überprüft werden folgende Ressourcen:
 - a. Maschinen-Katalog
 - b. Delivery-Group

⁵⁷ DOCKER INC., Dockerfile reference (2016).
<https://docs.docker.com/engine/reference/builder/>. (23.11.2016)

- c. Broker-Server
- 2. Es wird überprüft, ob die Host-VM als XenApp schon erstellt wurde. Falls nicht, wird diese erstellt.
- 3. Die eigentliche „Container-XenApp“ wird erstellt.

Ein wichtiger Aspekt bei dieser Arbeit ist es, dass sich alle Operationen der Integration zentral verwalten lassen. In diesem Fall ist der zentrale Punkt für die Erstellung und das Arbeiten mit den Containern die Host-VM auf dem XenServer, die die Docker-Container bereitstellt. Der Benutzer/Entwickler soll sich nicht jedes Mal, wenn er einen Container integrieren will, umständlich auf den Delivery-Controller einloggen müssen, um dort das Skript zu starten, das die Container als XenApps anlegt. Dafür gibt es zwei Möglichkeiten, zum einen PowerShell-Remoting und zum anderen SSH.

Falls die Container auf einer Windows-VM laufen, kann PowerShell-Remoting verwendet. PowerShell ist zwar mittlerweile auch für Linux-Betriebssysteme verfügbar, besitzt aber bis dato noch keine Unterstützung von PowerShell-Remote-Sessions von Linux zu Windows.⁵⁸⁵⁹ Mit PowerShell-Remoting lassen sich PowerShell-Befehle auf Remote-Computern ausführen. Für die Konfiguration von PowerShell-Remoting stellt Citrix ein Skript namens „Enable-XAPSRemoting“ bereit, mit dessen Hilfe man das PowerShell-Remoting auf dem Delivery-Controller einrichten kann. Dieses Skript basiert auf Windows eigenem Skript namens „Enable-PSRemoting“, ermöglicht jedoch, dass auch „nicht-lokale“ Administratoren eine Remote-Session aufbauen können.⁶⁰ Nachdem das Skript ausgeführt wurde, akzeptiert der Server PowerShell Verbindungen für die XenApp Kommandos.

Nach dem Einrichten von PowerShell-Remoting muss noch PowerShell auf der Host-VM installiert werden. Danach muss der Delivery-Controller noch als vertrauenswürdiger Server aufgenommen werden. Dies erfolgt über den Befehl „Set-TrustedHosts <server>“, der auf der Host-VM ausgeführt wird.

⁵⁸ GITHUB, PowerShell/PowerShell.

<https://github.com/PowerShell/PowerShell>. (12.12.2016)

⁵⁹ GITHUB, PSRP over WSMan/WinRM from Linux/OSX to Windows · Issue #1883 · PowerShell/PowerShell.

<https://github.com/PowerShell/PowerShell/blob/master/docs/KNOWNISSUES.md#remoting-support>. (13.12.2016)

⁶⁰ CITRIX SYSTEMS INC., XenApp 6 SDK – Remoting via PowerShell Remoting | Citrix Blogs (2016).

<https://www.citrix.com/blogs/2010/09/07/xenapp-6-sdk-remoting-via-powershell-remoting/>.

(23.11.2016)

Listing 5 zeigt wie die Powershell-Remoting Session aufgebaut wird.

```
#Define Remote Session
$session = New-XAPSSession -Computername $DCIP -SkipCertCheck

#Start Remote Session
Import-PSSession -Session $session
```

Listing 5: Verbindungsaufbau per Powershell-Remoting

Arbeitet man mit einer Linux-VM, empfiehlt es sich die Verbindung zu dem Delivery-Controller per SSH aufzubauen. Dazu kann ein beliebiger SSH-Server auf dem Delivery-Controller genutzt werden. Mit dem Befehl

```
ssh "$xenAdmin@$deliveryControllerIP" 'powershell
"C:\createXenContainer.ps1" [PARAMS]'
```

kann dann einfach das PowerShell-Skript mit den nötigen Parametern aufgerufen werden.

Listing 6 zeigt, wie der Container als XenApp mit dem Citrix SDK angelegt wird.

```
#
# Create Application 'Docker Container'
#

LogWrite "Creating Docker-XenApp Application..."
New-BrokerApplication -ApplicationType "HostedOnDesktop" ` 
    -CommandLineArguments "/c call %homedrive%\connectToContainer.bat $hostVMIP $p" ` 
    -CommandLineExecutable "%SystemRoot%\System32\cmd.exe" ` 
    -CpuPriorityLevel "Normal" -DesktopGroup $DesktopGroup -Enabled $True ` 
    -IconUid 10 -MaxPerUserInstances 0 -MaxTotalInstances 0 ` 
    -Name "Docker Container $containerName" -Priority 0 ` 
    -PublishedName "Docker Container $containerName" ` 
    -SecureCmdLineArgumentsEnabled $True -ShortcutAddedToDesktop $False ` 
    -ShortcutAddedToStartMenu $False -UserFilterEnabled $False ` 
    -Visible $True -WaitForPrinterCreation $False ` 
    -WorkingDirectory "%SystemRoot%\System32"
LogWrite "Successfully created Docker Container Application."
```

Listing 6: Erstellen der Container-XenApp mit dem Citrix SDK

Die XenApp für den Docker Container ist im Endeffekt nur ein Verweis auf die Windows-Konsole des Broker-Servers (-CommandLineExecutable "%SystemRoot%\System32\cmd.exe"), die dann startet und über den Befehl call %homedrive%\connectToContainer.bat \$hostVMIP \$p eine Batch-Datei auf dem Broker-Server

ausführt. Als Parameter werden die IP-Adresse der Host-VM und der freigegebene Port des Docker-Containers übergeben. Die Batch-Datei baut dann eine Verbindung zum Container mit SSH auf und kümmert sich um den Schlüsselaustausch.

Nahezu äquivalent verhält es sich bei der Erstellung der XenApp für die Verbindung vom Broker zur Host-VM. Die XenApp für die Host-VM erhält (neben einem anderen Namen) ein anderes Icon und bei dem Aufruf der Batch-Datei wird der Standard-Port 22, der für die SSH-Verbindung zur Host-VM reserviert ist, mitübergeben.

Die Container können jetzt als XenApps über eine Remote-Powershell Session vom Host aus erstellt werden. Diese sind zu diesem Zeitpunkt zwar im StoreFront sichtbar, besitzen aber noch keine Funktionalität, da die Batch-Datei, auf die die XenApp verweist, noch nicht erstellt wurde.

6.5.2.4 Aufbauen der SSH-Verbindung zum Container

Der nächste Schritt ist das Initialisieren des Verbindungsaufbaus per SSH vom Broker-Server zu dem Container mit Hilfe der Batch-Datei.

Nachdem OpenSSH auf dem Broker-Server installiert wurde, wird eine Batch-Datei zur Verfügung gestellt, die die eigentliche Verbindung zu dem Container herstellt. Diese wird über die Windows-Konsole des Broker-Servers von der XenApp des jeweiligen Containers aufgerufen (siehe Kapitel 6.5.2.3).

Listing 7 zeigt einen Auszug aus dem Batch-Skript das von der Container-XenApp aufgerufen wird.

```
[...]

:: Check for key-file
IF exist %keyFile% GOTO CHECKCONNECTION

call ssh-keygen -f %HOMEDRIVE%%HOME PATH%\ssh\id_rsa -t rsa -N ''
call icacls %HOMEDRIVE%%HOME PATH%\ssh\ /t /grant:r *S-1-3-0:(F)
/inheritance:r

:CHECKCONNECTION

:: Check ContainerConnection-File
IF exist %conConnectFile% GOTO SSH

call ssh %containerUserName%@%containerIP% -p %p% "umask 077; test -d .ssh
|| mkdir .ssh ; cat >> .ssh/authorized_keys || exit 1" < %keyFile%
echo DockerConnection: > %conConnectFile%
echo ContainerIP:      %containerIP% >> %conConnectFile%
echo Port:              %p% >> %conConnectFile%
echo UserName:          %containerUserName% >> %conConnectFile%

:SSH

call ssh %containerUserName%@%containerIP% -p %p%

[...]
```

Listing 7: Batch-Skript für den Verbindungsauftbau per SSH

Das Skript prüft, ob der Benutzer über die Schlüssel für die SSH-Verbindung verfügt und erstellt diese, falls sie nicht vorhanden sind. Für jede Verbindung wird ein „Connection-File“ angelegt, das die IP-Adresse, den Port und den Benutzernamen des Containers enthält. Falls diese Datei noch nicht angelegt wurde, weiß das Skript, dass noch nie eine Verbindung für den Benutzer, der die „Container XenApp“ gestartet hat, hergestellt wurde und kopiert deshalb den erstellten Public-Key des Benutzers in die „authorized_keys“-Datei des Containers. Damit weiß der SSH-Server des Containers, dass er dem Client und dessen Benutzer vertrauen kann und baut die SSH-Verbindung auch ohne Passwort-Eingabe auf. Zum Schluss stellt er die eigentliche SSH-Verbindung zu dem Container her.

6.5.2.5 Löschen der XenApp-Container

Um die XenApps der Container auch löschen zu können wird ein weiteres Skript bereitgestellt. Das Skript prüft, ob die XenApp für den Container vorhanden ist und löscht diese anschließend. Zusätzlich bereinigt es die Verbindungsdateien, die im Zuge der Integration angelegt werden.

```
$dc = Get-BrokerApplication -Name "Docker Container $containerName" -  
ErrorAction SilentlyContinue  
if ($dc) {  
    LogWrite "Deleting Docker-App 'Docker Container $containerName'."  
    Remove-BrokerApplication -InputObject $dc  
    LogWrite "Successfully deleted Docker-App."  
    LogWrite "Deleting .con Files..."  
    Get-ChildItem \\$brokerVMName\c$\users -recurse -Filter "$hostVMIP-  
$p*.con" | foreach ($_) {remove-item $_.fullname}  
}  
else {  
    LogWrite "No Docker-App 'Docker Container $containerName' found."  
}
```

Listing 8: Löschen einer Container-XenApp

6.5.2.6 Ablauf der Integration der Container

Um die Integration zu testen, wird ein offizielles CUDA-Image von NVIDIA verwendet:

```
FROM nvidia/cuda:8.0-devel
MAINTAINER Bernd Fecht "bernd.fecht@hs-augsburg.de"

# Install CUDA Samples

RUN apt-get update && apt-get install -y --no-install-recommends \
    cuda-samples-$CUDA_PKG_VERSION && \
    rm -rf /var/lib/apt/lists/*

WORKDIR /usr/local/cuda/samples/0_Simple/matrixMulCUBLAS
RUN make

# Install OpenSSH and set password

RUN \
    apt-get install -y openssh-client openssh-server && \
    mkdir /var/run/sshd && \
    echo 'root:root' |chpasswd

RUN sed -ri 's/^PermitRootLogin\s+.*$/PermitRootLogin yes/' \
/etc/ssh/sshd_config
RUN sed -ri 's/UsePAM yes/#UsePAM yes/g' /etc/ssh/sshd_config

# Expose Port for SSH usage

EXPOSE 22

# Add CUDA back to path during SSH

RUN echo "export PATH=$PATH:/usr/local/cuda/bin" >> /etc/profile

COPY ./ssh.sh /
CMD ["/ssh.sh"]
```

Listing 9: Dockerfile des Test-Containers

Das CUDA-Image basiert auf einem Ubuntu 14.04 Betriebssystem. Als Beispiel für ein Programm innerhalb des Containers, wird ein CUDA-Programm heruntergeladen, das eine einfache Matrixmultiplikation auf der Grafikkarte ausführt.

Nach dem „Builden“ des Images mit dem Befehl `nvidia-docker build -t <image-name> .` lässt sich der Container mit dem Befehl `nvidia-docker run -d -p 32790:22 -name <container-name> <image-name>` starten, wobei mit `-p` der Port (32780) auf der Host-VM für den Port 22 des Docker-Containers „gemappt“ wird.

Als nächstes wird das PowerShell-Skript für die Erstellung der Container-XenApp von der Host-VM aufgerufen. Nach dem Starten des Skripts muss man sich zunächst mit einem Administrator-Account für die Remote-Powershell Session authentifizieren. Anschließend prüft das Skript, ob die nötigen Ressourcen vorhanden sind. Da die Host-VM noch nicht als XenApp hinterlegt wurde, wird diese angelegt. Zum Schluss legt das Skript den Container als XenApp an.

Loggt man sich nun im StoreFront ein, sieht man die neu erstellten XenApps, die sich durch einen einfachen Mausklick starten lassen.

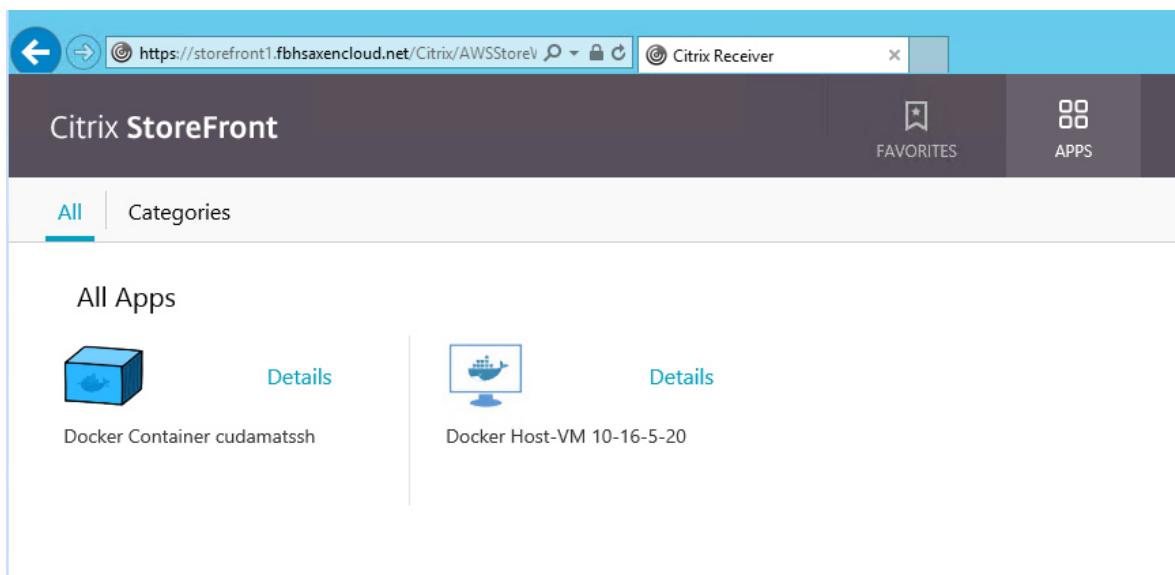


Abbildung 32: "Docker XenApps" im StoreFront

Beim Starten einer der XanApps wird dabei im Hintergrund die Batch-Datei auf dem Broker-Server gestartet, die die SSH-Verbindung zu dem Container aufbaut. Bei der initialen Verbindung wird erst einmal gefragt, ob der Verbindung vertraut werden soll, danach kopiert das Skript den Public-Key in den Docker-Container (siehe Abbildung 33).

 A screenshot of a terminal window titled 'Docker-XenApp Connector by b.fecht'. The window contains a black terminal session with white text. The text shows the following steps:


```
Enter username: root
The authenticity of host '[10.16.5.20]:32790 <[10.16.5.20]:32790>' can't be established.
ECDSA key fingerprint is SHA256:xbP+T6i/C3co7SJBN2qJffj3xE1LjQyK11k1MFKEcn4.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[10.16.5.20]:32790' (ECDSA) to the list of known hosts.
root@10.16.5.20's password:
```

 The terminal prompt 'root@10.16.5.20's password:' is visible at the bottom.

Abbildung 33: Initiale Verbindung der "Docker-XenApp" zum Container

Schaut man auf den Broker-Server, sieht man, dass nun die Verbindung als Datei hinterlegt wurde (siehe Abbildung 34). Ab jetzt weiß das Skript also, dass die Verbindung zum Container schon einmal hergestellt wurde und der Public-Key nicht mehr kopiert werden muss.

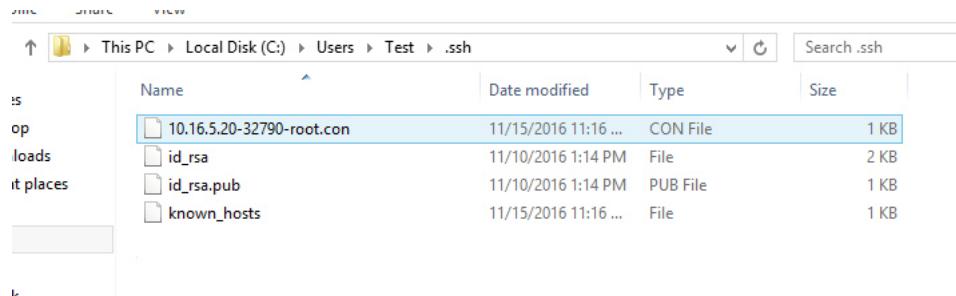


Abbildung 34: Das angelegte "Connection-File" im User-Profil

Von jetzt an kann die Verbindung auch ohne Passwort-Eingabe aufgebaut werden und das CUDA-Programm getestet bzw. ausgeführt werden:

A screenshot of a Citrix StoreFront interface. On the left, there's a sidebar with 'Citrix StoreFront' at the top, followed by 'All', 'Category', 'All Apps', and 'Docker Container'. The main area is a terminal window titled 'Docker-XenApp Connector by b.fecht'. The terminal output is as follows:

```
Enter username: root
Last login: Tue Nov 15 11:16:13 2016 from 10.16.5.40
root@963a53663c93:~# nvidia-smi
Tue Nov 15 11:25:52 2016
+-----+
| NVIDIA-SMI 367.57 | Driver Version: 367.57 |
+-----+
| GPU Name Persistence-M| Bus-Id Disp.A : Volatile Uncorr. ECC |
| Fan Temp Perf Pwr:Usage/Cap| Memory-Usage : GPU-Util Compute M. |
|-----+
| 0 GRID K520 Off : 0000:00:03.0 Off : N/A |
| N/A 31C P8 17W / 125W : 0MiB / 4036MiB : 0% Default |
+-----+
| Processes:                               GPU Memory |
| GPU PID Type Process name               Usage     |
|-----+
| No running processes found             |
+-----+
root@963a53663c93:~# cd /usr/local/cuda/samples/0_Simple/matrixMulCUBLAS
root@963a53663c93:/usr/local/cuda/samples/0_Simple/matrixMulCUBLAS# ./matrixMulCUBLAS -sizemult=10
[Matrix Multiply CUBLAS] - Starting...
GPU Device 0: "GRID K520" with compute capability 3.0
MatrixA<1280,960>, MatrixB<960,640>, MatrixC<1280,640>
Computing result using CUBLAS...done.
Performance= 1098.27 GFlop/s, Time= 1.432 msec, Size= 1572864000 Ops
Computing result using host CPU...done.
Comparing CUBLAS Matrix Multiply with CPU results: PASS
NOTE: The CUDA Samples are not meant for performance measurements. Results may vary when GPU Boost is enabled.
root@963a53663c93:/usr/local/cuda/samples/0_Simple/matrixMulCUBLAS# _
```

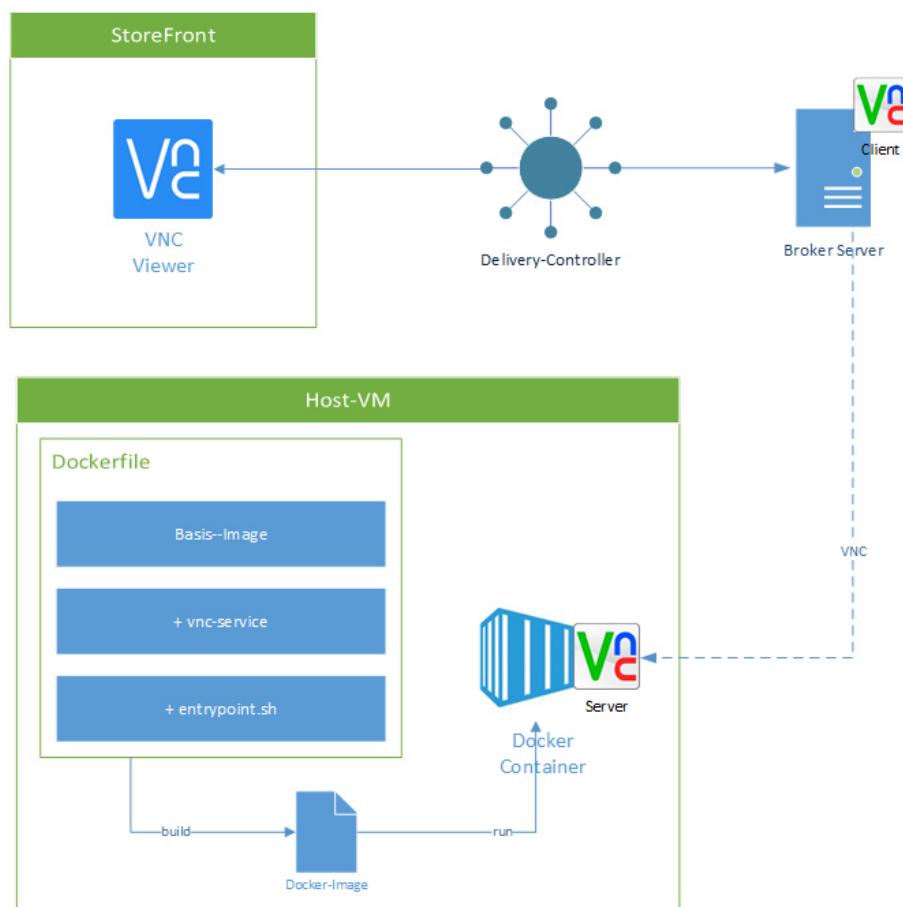
Abbildung 35: Ausgabe des CUDA-Programms

6.5.3 Integration der Container per VNC

6.5.3.1 Architektur

Falls das Programm des Entwicklers eine grafische Ausgabe verursacht, wäre es von Vorteil, diese Container ebenfalls über den StoreFront aus erreichen zu können. Deshalb wird für die Integration zusätzlich VNC genutzt.

VNC (Virtual Network Computing) ist ein Protokoll, mit dem der Bildschirminhalt eines Remote-Rechners von einem lokalen Rechner empfangen werden kann. Dabei können Benutzereingaben vom lokalen Rechner aus im Remote-Rechner ausgeführt werden. Auf dem Remote-Rechner läuft dazu ein VNC-Server und auf dem lokalen Rechner ein VNC-Client.⁶¹



Wie bei der Integration mit SSH wird zunächst das Dockerfile mit der Installation des VNC-Servers erweitert. Aus dem erstellten Image lassen sich dann die Container starten. Auf dem Broker-Server läuft ein VNC-Client namens „RealVNC Viewer“, mit dem die Verbindung zu dem Docker Container aufgebaut wird. Der Delivery-Controller stellt diesen

⁶¹ DATACOM, VNC :: virtual network computing :: ITWissen.info (2011).
<http://www.itwissen.info/definition/lexikon/virtual-network-computing-VNC.html>. (23.11.2016)

Client als XenApp zur Verfügung. Über die XenApp des RealVNC Viewers lassen sich dann unter Angabe der IP-Adresse und des Ports des Containers die Verbindung einrichten.

6.5.3.2 Erweiterung des Docker Image mit VNC

Die Erweiterung eines Basis-Images mit VNC erfolgt ähnlich wie auch schon bei der Erweiterung mit SSH. Nach der Angabe des Basis-Images wird der VNC-Server mit einem LXDE-Desktop und Firefox installiert, das Passwort für die Verbindung wird gesetzt und der Port wird freigegeben.

```
FROM nvidia/cuda
MAINTAINER Bernd Fecht "bernd.fecht@hs-augsburg.de"

# Install LXDE, VNC server and Firefox
RUN apt-get update && DEBIAN_FRONTEND=noninteractive apt-get install -y \
    firefox \
    lxde-core \
    lxterminal \
    tightvncserver

# Set user for VNC server (USER is only for build)
ENV USER root

RUN echo 'root:root' | chpasswd

EXPOSE 5901

COPY ./entrypoint.sh /
ENTRYPOINT ["/entrypoint.sh"]
```

Listing 10: Erweitern eines Dockerfiles mit VNC

Alternativ für den „CMD“ Befehl wird hier der Befehl „ENTRYPOINT“ verwendet. Der Unterschied zwischen CMD und ENTRYPOINT besteht lediglich darin, dass der CMD Aufruf beim Erstellen des Containers mit einem eigenen Argument überschrieben werden kann. Beispielsweise würde das Aufrufen des Containers mit dem Befehl docker run <myimage> <hostname> den CMD Aufruf und somit das Aufrufen der ssh.sh-Datei mit der Anweisung „hostname“ überschreiben und nur den Host-Name des Containers ausgegeben. Welche Variante gewählt wird, hängt von den Anforderungen des Entwicklers ab.

Die „entrypoint.sh“-Datei startet den VNC-Server:

```

#!/bin/bash

# Remove VNC lock (if process already killed)
rm /tmp/.X1-lock /tmp/.X11-unix/X1
# Run VNC server with tail in the foreground
vncserver :1 -geometry 1280x800 -depth 24 && tail -F /root/.vnc/*.log

```

Listing 11: entrypoint.sh

Das Builden und das Starten des Containers funktioniert genau gleich wie bei SSH (siehe Kapitel 6.5.2.2).

6.5.3.3 Bereitstellung des RealVNC Viewers als XenApp

Da die XenApp nur einmal für alle relevanten Benutzer bereitgestellt werden muss, lohnt sich ein Anlegen der Ressource über ein Skript nicht. Nach der Installation des RealVNC Viewers auf dem Broker-Server erstellt man die XenApp über das Studio, wie schon in Kapitel 6.1.6 beschrieben.

6.5.3.4 Verbindungsauflauf zum Docker-Container per VNC

Nachdem der Container gestartet wurde, kann über die zuvor bereitgestellte XenApp des RealVNC Viewers eine Verbindung auf den Desktop des Containers hergestellt werden. Dazu loggt man sich in den StoreFront über dessen URL-Adresse ein und startet die XenApp. Über den Reiter „File > New connection...“ gibt man die IP-Adresse und den Port zu dem VNC-Container an und speichert die Verbindung. Danach kann der virtuelle Desktop des Containers gestartet werden (siehe Abbildung 36).

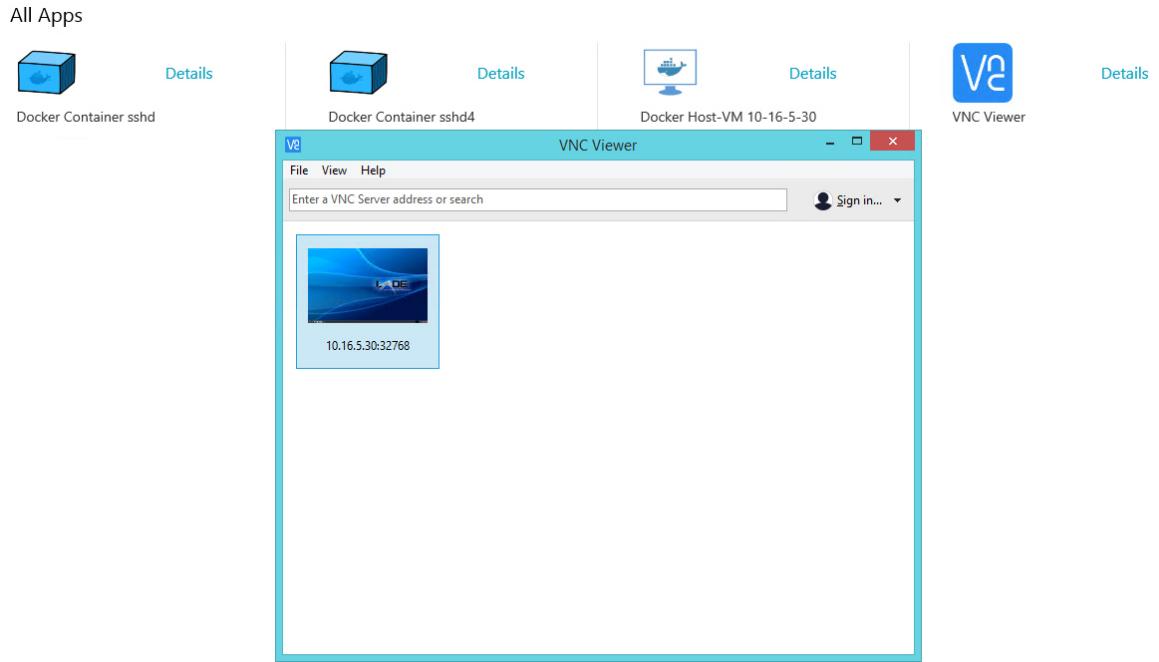


Abbildung 36: RealVNC Viewer im StoreFront

Ob die Grafikkarte erfolgreich durchgereicht wurde, lässt sich wieder mit dem Befehl `nvidia-smi` verifizieren. Wie in Abbildung 37 zu sehen, können zusätzlich Benchmark-Programme wie „glxgears“ ausgeführt werden, um eine visuelle Bestätigung für die Funktionsweise des grafikfähigen Docker-Containers zu erhalten.

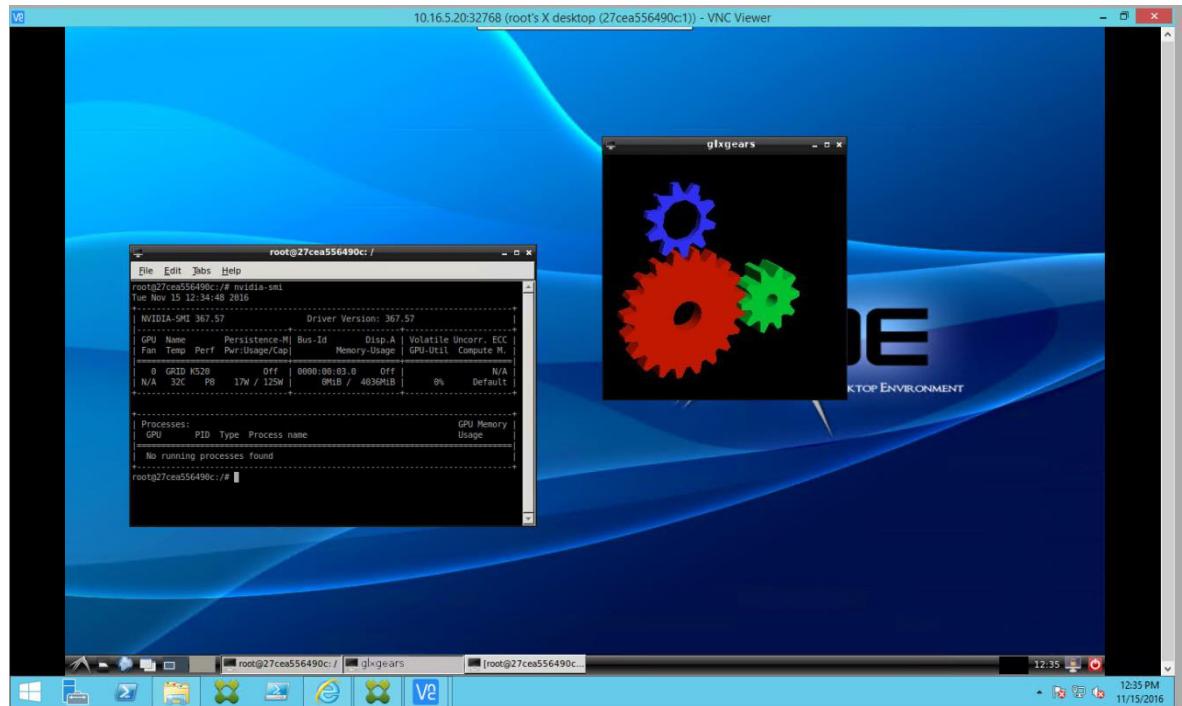


Abbildung 37: Der virtuelle Desktop innerhalb des Containers

7. Ergebnisse

Im Rahmen dieser Arbeit wurde zunächst untersucht, wie eine virtuelle Umgebung unter Verwendung von Citrix aufgesetzt und für Docker-Container konzipiert werden könnte.

Des Weiteren wurde mit NVIDIA-Docker eine Möglichkeit gefunden, dass Docker-Container, ohne Verlust ihrer Portabilität, Berechnungen auf der Grafikkarte durchführen können. Dabei wurde verifiziert, dass diese Lösung auch mit der Architektur der vGPUs funktioniert.

Anschließend wurde sowohl das Container-Management von Citrix auf dem XenServer als auch eine eigene Lösung für die Integration der Docker-Container in die virtuelle Xen-Umgebung genutzt.

Somit können Administratoren die Container auf dem XenServer administrieren und Entwickler nun ihre Grafik-beschleunigten Programme in Docker-Container „verpacken“ und diese dann zentral über die Host-VM aus als XenApps beim StoreFront registrieren. Die Container können dann auf dem StoreFront durch einen einfachen Mausklick gestartet werden.

8. Ausblick

Das in dieser Arbeit entwickelte Konzept der Integration ist vor allem eine wissenschaftliche Untersuchung, ob Entwickler von GPU-beschleunigten Anwendungen Docker-Container in einer Xen-Umgebung nutzen können und wie eine geeignete Integration dieser Docker-Container in die Xen-Umgebung funktioniert. Demnach lässt die erarbeitete Lösung noch viel Freiraum für Weiterentwicklungen.

Eine Weiterentwicklung wäre zum Beispiel, das Encoding der NVIDIA vGPU namens NVENC innerhalb der Container für die Übertragung des Bildschirms anstelle von VNC zu nutzen.⁶² Da VNC den Videostream über die CPU encodiert, könnte mit NVENC die dafür nötige CPU-Last auf die GPU ausgelagert werden. Zudem könnte jeder Docker-Container mit grafischer Ausgabe auch als eigene XenApp angelegt werden, anstatt über einen einzigen Client im StoreFront aufgerufen werden zu müssen.

Auch könnte ein Wrapper programmiert werden, mit dem die Docker-Container bei der Erstellung automatisch als XenApps angelegt werden und beim Löschen der Container auch die XenApp aus dem StoreFront mitlöscht.

Alternativ könnte das Container-Management auch mit der Integration der Container im StoreFront gekoppelt werden, indem die Daten der Container von den XenServer abgefragt werden.

⁶² NVIDIA VIDEO CODEC SDK.
<https://developer.nvidia.com/nvidia-video-codec-sdk>. (12.12.2016)

Literaturverzeichnis

- AMAZON WEB SERVICES, INC., AWS | Amazon Virtual Private Cloud (VPC) & VPS Hosting (2016).
<https://aws.amazon.com/de/vpc/>. (28.09.2016).
- AMAZON WEB SERVICES, INC., Regions and Availability Zones - Amazon Elastic Compute Cloud (2016a).
<http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.html>. (28.09.2016).
- AMAZON WEB SERVICES, INC., AWS CloudFormation – Infrastruktur als Code und AWS-Ressourcenbereitstellung (2016).
<https://aws.amazon.com/de/cloudformation/>. (17.11.2016).
- AMAZON WEB SERVICES, INC., Was ist AWS? - Amazon Web Services (2016).
<https://aws.amazon.com/de/what-is-aws/>. (17.11.2016).
- AMAZON WEB SERVICES, INC., Scenario 2: VPC with Public and Private Subnets (NAT) - Amazon Virtual Private Cloud (2016b).
http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_Scenario2.html. (18.11.2016).
- AMAZON WEB SERVICES, INC., Security Groups for Your VPC - Amazon Virtual Private Cloud (2016).
http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_SecurityGroups.html. (07.12.2016).
- Peter BATS, Citrix XenApp and XenDesktop 7.6 on Amazon Web Services Reference Architecture. White Paper - Citrix (2015).
- CITRIX SYSTEMS INC., Databases (2016).
<https://docs.citrix.com/en-us/xenapp-and-xendesktop/7-8/technical-overview/databases.html>. (02.10.2016).
- CITRIX SYSTEMS INC., Installing XenServer Tools (2016).
<http://docs.citrix.com/fr-fr/xencenter/6-5/xs-xc-vms-configuring/xs-xc-vms-installtools.html>. (14.10.2016).
- CITRIX SYSTEMS INC., Installing XenServer Tools (2016).
<http://docs.citrix.com/fr-fr/xencenter/6-5/xs-xc-vms-configuring/xs-xc-vms-installtools.html>. (15.10.2016).
- CITRIX SYSTEMS INC., NetScaler 11.0 (2016).
<https://docs.citrix.com/en-us/netscaler/11.html>. (19.11.2016).
- CITRIX SYSTEMS INC., XenApp 6 SDK – Remoting via PowerShell Remoting | Citrix Blogs (2016).

<https://www.citrix.com/blogs/2010/09/07/xenapp-6-sdk-remoting-via-powershell-remoting/>. (23.11.2016).

Citrix XenServer 7.0 Supplemental Packs & the DDK. Citrix Systems (2016).

DATACOM, VNC :: virtual network computing :: ITWissen.info (2011).
<http://www.itwissen.info/definition/lexikon/virtual-network-computing-VNC.html>. (23.11.2016).

Desktop, Client, Server, Anwendung: Ratgeber: Was ist was bei der Virtualisierung.
<http://www.tecchannel.de/a/ratgeber-was-ist-was-bei-der-virtualisierung,2037095,5>. (09.07.2016).

Jonathan DESROCHER, Running Hyper-V VMware or Xen on an AWS EC2 Instance? - CloudStacking.com (2013).
<http://cloudstacking.com/posts/running-hyper-v-vmware-or-xen-on-an-aws-ec2-instance.html>. (05.12.2016).

DOCKER INC., Docker Overview (2016a).
<https://docs.docker.com/engine/understanding-docker/>. (15.09.2016).

DOCKER INC., docker/distribution (2016).
<https://github.com/docker/distribution>. (10.09.2016).

DOCKER INC., Remote API (2016).
https://docs.docker.com/engine/reference/api/docker_remote_api/. (09.09.2016).

DOCKER INC., Understand images, containers, and storage drivers (2016b).
<https://docs.docker.com/engine/userguide/storagedriver/imagesandcontainers/>. (26.11.2016).

DOCKER INC., Understand the architecture (2016c).
<https://docs.docker.com/v1.8/introduction/understanding-docker/>. (09.09.2016).

DOCKER INC., What is Docker? (2016).
<https://www.docker.com/what-docker>. (04.09.2016).

DOCKER INC., Dockerfile reference (2016).
<https://docs.docker.com/engine/reference/builder/>. (23.11.2016).

Dom0 - Xen (2015).
<http://wiki.xenproject.org/wiki/Dom0>. (08.07.2016).

Daniel FELLER, XenApp 7 Deployment Blueprint,Citrix XenDesktop 7 - Blueprint (2013).

GCC, the GNU Compiler Collection - GNU Project - Free Software Foundation (FSF) (2016).
<https://gcc.gnu.org/>. (03.11.2016).

GITHUB, PowerShell/PowerShell.
<https://github.com/PowerShell/PowerShell>. (12.12.2016).

GITHUB, PSRP over WSMAN/WinRM from Linux/OSX to Windows · Issue #1883 · PowerShell/PowerShell.

<https://github.com/PowerShell/PowerShell/blob/master/docs/KNOWNISSUES.md#remoting-support>. (13.12.2016).

Udo HELMBRECHT / Gunnar TEEGE / Björn STELTE, Virtualisierung: Techniken und sicherheitsorientierte Anwendungen. Universität der Bundeswehr München - Institut für Technische Informatik (2010).

Chris HORNE, Understanding Full Virtualization, Paravirtualization, and Hardware Assist. White Paper - vmWare.

JINGLI XU, Einsatz von Virtualisierungstechnologien in der IT-Infrastruktur eines Unternehmens: Vergleich und Einteilung mit Ausrichtung auf Desktopbereitstellung (2009).

Matthias KARL / Sean P. KANE, Docker: up and running (2016).

KVM Best Practices. Virtualisierungslösungen für den Enterprise-Bereich (2012).

Linux Containers (2016).

<https://linuxcontainers.org/>. (04.09.2016).

Make (2016).

<http://www.selflinux.org/selflinux/html/make01.html>. (03.11.2016).

Dirk MAKOWSKI, Microsoft OS/2.

<http://www.winhistory.de/more/os2.htm>. (14.07.2016).

Christoph MEINEL, Virtualisierung und Cloud Computing : Konzepte, Technologiestudie, Marktübersicht (Technische Berichte des Hasso-Plattner-Instituts für Softwaresystemtechnik an der Universität Potsdam ; 44) (2010).

MICROSOFT, Always On Availability Groups (SQL Server) (2016).

<https://msdn.microsoft.com/en-us/library/hh510230>. (02.10.2016).

MICROSOFT, Database Mirroring Witness (2016).

<https://msdn.microsoft.com/en-us/library/ms175191.aspx>. (03.10.2016).

modprobe(8): add/remove modules from Kernel - Linux man page (2016).

<https://linux.die.net/man/8/modprobe>. (03.11.2016).

Henrik MÜHE, Virtualisierung - Geschichte, Techniken und Anwendungsfälle/ Henrik Mühe; Studienarbeit. Literaturverz. S. 26 (2007).

nouveau (2016).

<https://nouveau.freedesktop.org/wiki/>. (03.11.2016).

NVIDIA, CUDA Toolkit.

<https://developer.nvidia.com/cuda-toolkit>. (22.11.2016).

NVIDIA VIDEO CODEC SDK.

<https://developer.nvidia.com/nvidia-video-codec-sdk>. (12.12.2016).

Matthew PORTNOY, Virtualization Essentials (2012).

Seminar "Virtualisierung von Betriebssystemen: VMware Architektur" (2007).
<http://www.fh-wedel.de/~si/seminare/ws06/Ausarbeitung/02.VMware/vmware4.htm>.
(14.07.2016).

SLIDEShare.NET, Common Pitfalls when Setting up a NetScaler for the First Time.
<http://www.slideshare.net/davidmcg/common-pitfalls-when-setting-up-a-net-scaler-for-the-first-time>. (19.11.2016).

James TURNBULL, The Docker book (2014).

Virtualisierung > Wiki > [ubuntuusers.de](https://wiki.ubuntuusers.de/Virtualisierung/#Paravirtualisierung).
<https://wiki.ubuntuusers.de/Virtualisierung/#Paravirtualisierung>. (08.07.2016).

Virtualisierung als Basis für Cloud Computing.
<http://www.computerworld.ch/whitepapers/it-management/artikel/virtualisierung-als-basis-fuer-cloud-computing-65834/>. (07.07.2016).

VMWARE, Virtualisierung und Software für virtuelle Maschinen (2016).
<http://www.vmware.com/de/solutions/virtualization.html>. (30.11.2016).

Robert VOGEL / Tarkan KOCOGLU / Thomas BERGER, Desktop Virtualisierung. Definitionen - Architekturen - Business-Nutzen (2010).

Xen ARM with Virtualization Extensions whitepaper - Xen (2016).
http://wiki.xen.org/wiki/Xen_ARM_with_Virtualization_Extensions_whitepaper.
(08.07.2016).

XEN PROJECT, XAPI Command Line Interface - Xen (2014).
https://wiki.xen.org/wiki/XAPI_Command_Line_Interface. (14.10.2016).

Glossar

A

AMI *Amazon Machine Image*
AWS *Amazon Web Services*

C

CaaS *Container-as-a-Service*
CIDR *Classless Inter-Domain Routing*
CMS *Content-Management-System*
CUDA .. *Copmute Unified Device Architecture*

D

DMZ *Demilitarisierte Zone*

I

IIS *Internet Information Services*

N

NSIP *NetScaler-IP*

S

SNIP *Subnet-IP*
SSL *Secure Socket Layers*
STA *Secure Ticket Authority*

V

VDA *Virtual-Delivery-Agent*
vGPU *Virtual Graphics Processing Unit,
virtual graphics processing unit*
VIP *Virtual-IP*
VNC *Virtual Network Computing*
VPN ... *Virtual Private Network, Virtual Private
Network*