

Deployment von Open-Source-Webanwendungen in virtualisierten Linux-Umgebungen

WOLFGANG KERSCHBAUMER

DIPLOMARBEIT

eingereicht am
Fachhochschul-Masterstudiengang

INTERACTIVE MEDIA

in Hagenberg

im Juli 2011

© Copyright 2011 Wolfgang Kerschbaumer

Diese Arbeit wird unter den Bedingungen der *Creative Commons Lizenz Namensnennung – Nicht-kommerziell – Keine Bearbeitung Österreich* (CC BY-NC-ND) veröffentlicht – siehe <http://creativecommons.org/licenses/by-nc-nd/3.0/at/>.

Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die aus anderen Quellen entnommenen Stellen als solche gekennzeichnet habe.

Hagenberg, am 23. Juni 2011

Wolfgang Kerschbaumer

Inhaltsverzeichnis

Erklärung	iii
Kurzfassung	vi
Abstract	vii
1 Einleitung	1
2 Deployment	2
2.1 Software-Deployment unter Linux	2
2.1.1 Advanced Packaging Tool (APT) unter Debian	4
2.1.2 LAMPP	4
2.1.3 Deployment von LAMPP-Anwendungen	8
3 Virtualisierung	11
3.1 Virtual Machine Monitor	12
3.2 Paravirtualisierung	16
3.3 Hardwareerweiterungen	16
3.4 Virtualisierungstechnologien	17
3.4.1 Systemvirtualisierung	17
3.5 libvirt, virsh und Virtual Machine Manager	23
3.5.1 Betriebssystemvirtualisierung	24
3.5.2 Virtual Appliances	26
3.6 Vorteile der Virtualisierung	28
3.7 Nachteile der Virtualisierung	29
3.8 Virtual Private Server (VPS)	31
4 Cloud-Computing	33
4.1 IAAS, PAAS und SAAS	34
4.1.1 Infrastructure as a Service (IaaS)	34
4.1.2 Platform as a Service (PaaS)	35
4.1.3 Software as a Service (SaaS)	36

4.2	Amazon Web Services (AWS)	36
4.2.1	Amazon Elastic Compute Cloud (Amazon EC2)	36
5	Deploymentlösungen	38
5.1	Modelle	38
5.1.1	Ausfallssicherheit	41
5.2	Konfigurationswerkzeuge	43
5.2.1	WD2-Tools	43
5.2.2	Funktionen	45
5.2.3	Fabric	52
5.2.4	Puppet	54
5.2.5	Cfengine	55
5.3	Betriebssystemkomponenten	57
5.3.1	ezjail	57
5.3.2	LAMPP-Stack	57
5.4	Cloud-Computing	58
5.4.1	JiffyBox	58
5.4.2	Bitnami und JumpBox	59
5.4.3	Beanstalk	60
6	Fazit	65
A	Glossar	67
	Literaturverzeichnis	75

Kurzfassung

Der Begriff Deployment umfasst das Verteilen, Konfigurieren und die Inbetriebnahme von Software. Das Deployment stellt einen wesentlichen Teil in der Entwicklung von Webanwendungen dar. Die Arbeit konzentriert sich auf Webanwendungen, die auf dem LAMPP-Stack basieren. LAMPP ist eine Abkürzung und steht für Linux, das freie und quelloffene Betriebssystem, Apache als Webserver-Software, das relationale Datenbankmanagementsystem MySQL sowie für die Skriptsprachen PHP, Perl oder Python.

Je nach Aufbau des zum Einsatz kommenden Stacks von Anwendungen kann so der Umzug vom Development- auf den Produktivserver mit einem erheblichen Aufwand verbunden sein. Um diesen Prozess zu vereinfachen wurde vom Autor ein Prototyp zum Deployment mittels der Secure Shell (SSH) geschaffen. Dessen Eigenschaften werden unter dem Aspekt virtueller Systeme diskutiert.

Zudem werden Deploymentlösungen, die Virtualisierungstechnologien voraussetzen, analysiert und auf ihre Tauglichkeit im Umgang mit LAMPP-basierten Anwendungen untersucht. Dazu werden sowohl mathematische Modelle vorgestellt als auch die Praxistauglichkeit für kleinere bis mittlere Webprojekte eruiert.

Abschließend werden die Ergebnisse diskutiert und es wird begründet, weshalb Virtualisierung in Zukunft eine wichtige Rolle in Deploymentprozessen von LAMPP-Anwendungen einnehmen wird.

Abstract

The term deployment involves the distribution, configuration and activation of software. The deployment process is an essential part of web application development. This work focuses on LAMPP applications. LAMPP is an abbreviation and stands for Linux, the free and open source operating system, the Apache web server software, the relational database management system MySQL and the scripting languages PHP, Perl or Python.

Depending on the stack of applications, the migration from the development to the production server can take quite an effort. To simplify this process the author implemented a prototype for the deployment via the secure shell (SSH). Its features are discussed in terms of virtualization technologies.

In addition to that, virtualization-based deployment solutions are analyzed and rated regarding their usefulness with LAMPP applications. To do so, mathematical models are presented and the suitability for web projects of small to medium size is determined.

Finally the results are discussed and reasons for the emerging importance of virtualization in the deployment of LAMPP applications are given.

Kapitel 1

Einleitung

Die vorliegende Arbeit befasst sich mit dem Deployment von Serverdiensten. Im Speziellen geht es dabei um das Einrichten von LAMPP, einem gängigen Stack für Webanwendungen, welcher in Abschnitt 2.1.2 vorgestellt wird. Deploymentlösungen für andere Webtechnologien wie dem Spring-Framework oder Microsofts *Internet Information Services* (IIS) werden nicht berücksichtigt. Die in Kapitel 5 vorgestellten Deploymentlösungen sind zudem für eine geringe bis mittlere Anzahl an zu wartenden Systemen ausgelegt, Lösungen für Unternehmen mit vielen tausend Installationen, wie es bei Datenzentren oder Internet Service Providern oft der Fall ist, sind ebenfalls nicht Teil des Themas.

Die Betriebssysteme, welche den folgenden Betrachtungen zugrunde liegen sind *Free and Open Source Software* (FOSS), da Vorgänge unter solchen Systemen einfacher nachzuvollziehen sind und der gesamte Programmcode frei zur Verfügung steht. Im Gegensatz zu proprietären Lösungen kann dieser bei Bedarf auch angepasst werden, wie das bei manchen Virtualisierungslösungen Voraussetzung ist. Das ist auch der Grund, wieso Linux am Virtualisierungs- und Cloud-Computing-Markt bevorzugt zum Einsatz kommt.

Das LAMPP-Deployment ist aufwendig und ohne Einsatz von Hilfsprogrammen nur manuell durchzuführen. Entsprechend hoch ist die Gefahr, dass sich Fehler einschleichen.

Wie diese Vorgänge automatisiert werden können und welche Vor- und Nachteile Virtualisierung im Deployment mit sich bringt wird in dieser Arbeit erläutert. Dabei wird das Deployment-Werkzeug des Autors mit weiteren Lösungen verglichen, außerdem wird beurteilt, welche Rolle Virtualisierung im jeweiligen Deployment-Vorgang spielt.

Kapitel 2

Deployment

2.1 Software-Deployment unter Linux

Software-Deployment, oder auch „Softwareverteilung“ ist ein breit gefasster Begriff. Im Rahmen der vorliegenden Arbeit ist mit Deployment das Installieren und Konfigurieren von Software gemeint. Die einfachste Methode Software unter UNIX- und Linuxdistributionen zu installieren und einzurichten ist, ein Paketverwaltungssystem (Package Management System) zu verwenden. Fast alle modernen Linux-Derivate stellen diese Installationsmethode zur Verfügung.

Diese Pakete beinhalten alle für die Integration in das jeweilige System wesentlichen Informationen und stellen die Software in Form von Quellcode oder vorkompilierten Binärpaketen zur Verfügung. Während bereits ausführbare Programme den schnelleren und komfortableren Installationsweg darstellen, erlauben Quellcodeinstallationen die Anpassung von Softwareeigenschaften vor dem Kompilieren. Binärpakete hingegen sind mit für ein möglichst breites Anwendungsspektrum sinnvollen Eigenschaften vorkompiliert.

Das verwendete Paketverwaltungssystem lässt sich nicht beliebig wechseln sondern muss speziell auf eine Linux-Variante angepasst werden und Kenntnis über System- und Verzeichnisaufbau des jeweiligen Betriebssystems haben. Bei der Installation muss lediglich das gewünschte Paket angegeben werden und die Paketverwaltung kümmert sich um den Rest. Wird das Paket nicht in den der Paketverwaltung bekannten Quellen gefunden, kann das mehrere Gründe haben: Da es keine einheitliche Namenskonvention für Pakete unter den Betriebssystemen gibt, ist es möglich, dass das Paket unter anderem Namen zur Verfügung steht. Bei weniger stark verbreiteten Programmen oder Bibliotheken kann es natürlich auch vorkommen, dass sich noch kein Betreuer (*Package Maintainer*) der gewünschten Software angenommen hat, um daraus ein Paket für die betreffende Plattform zu erstellen. Da auch

Tabelle 2.1: Paketmanagementsystem verschiedener Linux und UNIX-Betriebssysteme.

Betriebssystem	Paketmanager	Kommandozeilenbefehle
Debian und Ubuntu	APT	<code>dpkg</code> , <code>apt-get</code> , <code>aptitude</code>
Fedora	RPM	<code>yum</code>
RHEL	RPM	<code>yum</code>
openSUSE	RPM	<code>zypper</code>
SUSE Linux Enterprise	RPM	<code>zypper</code>
FreeBSD	FreeBSD-Ports	<code>make</code> , <code>pkg_add</code>
Mac OS X	MacPorts	<code>port</code>

Paketverwaltungssysteme quelloffen zur Verfügung stehen, steht es jedem frei, ein Verzeichnis (*Repository*) für kompatible Pakete anzubieten. Der Administrator muss beim Hinzufügen von Drittquellen entscheiden, ob er diesen ausreichendes Vertrauen entgegenbringt, denn sie stellen eine Möglichkeit dar, Schadsoftware zu verbreiten. Die Möglichkeit der Verwendung unterschiedlicher Paketquellen wird von Linux-Distributoren häufig dazu genutzt um Software nach Kriterien zu trennen. Debian-basierte Systeme nehmen die Lizenzbedingungen der Programme als Kategorisierungseigenschaft. Da die Verwendung proprietärer Software das Hinzufügen eines Repositories erfordert, kann der Administrator bei Bedarf sicherstellen, dass ausschließlich freie Software auf dem System installiert werden kann.

Tabelle 2.1 zeigt Paketmanagementsysteme einiger UNIX- und Linux-Betriebssysteme. Die Liste ist nicht vollständig, sondern zeigt Paketmanagementsysteme populärer Betriebssysteme. Vorherrschend sind *Advanced Packaging Tool* (APT) bei *Debian*-basierten Systemen und *RPM Package Manager, rekursives Akronym* (RPM) bei Red Hat basierten Systemen wie z. B. *Red Hat Enterprise Linux* (RHEL). Zudem stehen auf einigen Linux und UNIX-Varianten mehrere Paketverwaltungsprogramme zur Verfügung, hier wurde das, nach Meinung des Autors, populärste System genannt.

In den letzten Jahren hat das Konzept einer zentralen Paketverwaltung erneut an Bedeutung auf für Endanwender gewonnen. Nennenswerte Implementierungen sind der Mac App Store und der App Store für iOS-Geräte sowie der Android Market auf dem Google-Betriebssystem für mobile Geräte.

Zu einer der wichtigsten Eigenschaften eines Paketmanagementsystems zählt seine Fähigkeit, Abhängigkeiten zu weiteren Paketen sowie Konflikte mit bereits bestehender Software erkennen zu können und Unterstützung beim Auflösen solcher Probleme zu bieten. Auf das Paketmanagementsystem von *Debian* wird in Abschnitt 2.1.1 genauer eingegangen. Hierbei handelt

es sich um das erste weit verbreitete System mit einer sehr guten Abhängigkeitsauflösung.

2.1.1 Advanced Packaging Tool (APT) unter Debian

APT unterstützt Quellpakete und Binärpakete. In den Quellpaketen befindet sich sowohl der Quellcode der jeweiligen Software, als auch Anweisungen, wie dieser zu übersetzen und das übersetzte Programm ins System zu integrieren ist. Binärpakete, die stets die Endung `.deb` tragen, beinhalten neben Kontrolldateien bereits kompilierte, ausführbare Dateien. Das Kommandozeilenprogramm zum Arbeiten mit Debian-Paketen ist `dpkg` (siehe Abbildung 2.1). Darauf aufbauend wurden `apt-get` (siehe Abbildung 2.2) und später `aptitude` (siehe Abbildung 2.3) entwickelt um den Umgang mit Paketen weiter zu vereinfachen und zu automatisieren. `apt-get` nimmt Befehle direkt entgegen wogegen `aptitude` zusätzlich zu einzeiligen Befehlen auch eine `ncurses`-Oberfläche zur Bedienung bereitstellt. Diese ist in Abbildung 2.4 dargestellt.

Zur Verwaltung der Pakete werden Textdateien verwendet, Informationen über alle dem System bekannten Pakete sind in `/var/lib/dpkg/available` untergebracht [1].

Auch für grafische Bedienoberflächen wurden Werkzeuge zum Umgang mit Debian-Paketen entwickelt, das bekannteste darunter ist der auf `GTK+` basierende Paketmanager *Synaptic*, wie in Abbildung 2.5 gezeigt. Die Bildschirmfotos stammen aus Ubuntu 11.04, einer weit verbreiteten, auf *Debian* basierenden Distribution. Als weiteres Front-End für APT dient dort das *Software-Center*, wie es in Abbildung 2.6 zu sehen ist.

2.1.2 LAMP

Um Webanwendungen zu erstellen gibt es viele Möglichkeiten. Für klassische Anwendungen hat sich dabei die Verwendung von XAMPP etabliert. Diese Arbeit beschränkt sich auf das LAMP-Modell und geht auf andere im Einsatz befindliche Webserver- und Webentwicklungstechniken nicht ein.

Im Folgenden werden die Komponenten von XAMPP erläutert:

X steht für die Plattformunabhängigkeit (*cross platform*), der Stack ist unter den meisten Linux-basierten und unixoiden System sowie Microsoft Windows verfügbar. Da sich die Betrachtungen in dieser Arbeit auf Linux-Server beschränken wird im Folgenden ausschließlich vom LAMP-Stack die Rede sein. Dies ist gleichbedeutend mit der ebenfalls verwendeten Bezeichnung XAMPP für *Linux*.

```
wk42@roxy: ~/Downloads
wk42@roxy:~/Downloads$ sudo apt-get download php5-mysql
wk42@roxy:~/Downloads$ sudo dpkg -i php5-mysql_5.3.5-1ubuntu5_i386.deb
Selecting previously deselected package php5-mysql.
(Reading database ... 230469 files and directories currently installed.)
Unpacking php5-mysql (from php5-mysql_5.3.5-1ubuntu5_i386.deb) ...
dpkg: dependency problems prevent configuration of php5-mysql:
 php5-mysql depends on phpapi-20090626+lfs; however:
  Package phpapi-20090626+lfs is not installed.
 php5-mysql depends on php5-common (= 5.3.5-1ubuntu5); however:
  Package php5-common is not installed.
dpkg: error processing php5-mysql (--install):
 dependency problems - leaving unconfigured
Errors were encountered while processing:
 php5-mysql
wk42@roxy:~/Downloads$
```

Abbildung 2.1: Installation mit dpkg.

```
wk42@roxy: /
wk42@roxy:/$ sudo apt-get install php5-mysql
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
 apache2-mpm-prefork apache2-utils apache2.2-bin apache2.2-common
 libapache2-mod-php5 libapr1 libaprutil1 libaprutil1-dbd-sqlite3
 libaprutil1-ldap php5-cli php5-common
Suggested packages:
 apache2-doc apache2-suexec apache2-suexec-custom php-pear php5-suhosin
The following NEW packages will be installed:
 apache2-mpm-prefork apache2-utils apache2.2-bin apache2.2-common
 libapache2-mod-php5 libapr1 libaprutil1 libaprutil1-dbd-sqlite3
 libaprutil1-ldap php5-cli php5-common php5-mysql
0 upgraded, 12 newly installed, 0 to remove and 0 not upgraded.
Need to get 9,496 kB of archives.
After this operation, 27,5 MB of additional disk space will be used.
Do you want to continue [Y/n]?
```

Abbildung 2.2: Installation mit apt-get.

A bezieht sich auf den Apache Server für das *Hypertext Transfer Protocol* (HTTP)

M bedeutet hier das Datenbanksystem MySQL

P ist der Anfangsbuchstabe der Skriptsprache PHP

P steht für weitere Skriptsprachen wie *Perl* oder *Python*

```
wk42@roxy: ~/Downloads
wk42@roxy:~/Downloads$ sudo aptitude install php5-mysql
The following NEW packages will be installed:
  apache2-mpm-prefork(a) apache2-utils(a) apache2.2-bin(a)
  apache2.2-common(a) libapache2-mod-php5(a) libapr1(a) libaprutil1(a)
  libaprutil1-dbd-sqlite3(a) libaprutil1-ldap(a) php5-cli(a) php5-common(a)
  php5-mysql
0 packages upgraded, 12 newly installed, 0 to remove and 0 not upgraded.
Need to get 9,496 kB of archives. After unpacking 27.5 MB will be used.
Do you want to continue? [Y/n/?]
```

Abbildung 2.3: Installation mit aptitude install.

```
wk42@roxy: /
Actions Undo Package Resolver Search Options Views Help
C-T: Menu ?: Help q: Quit u: Update g: Download/Install/Remove Pkgs
aptitude 0.6.3
p php5-mysql <none> 5.3.5-1ubu
p php5-odbc <none> 5.3.5-1ubu
p php5-pgsql <none> 5.3.5-1ubu
p php5-pspell <none> 5.3.5-1ubu
p php5-recode <none> 5.3.5-1ubu
p php5-snmp <none> 5.3.5-1ubu
p php5-sqlite <none> 5.3.5-1ubu
p php5-tidy <none> 5.3.5-1ubu
p php5-xmldrpc <none> 5.3.5-1ubu
MySQL module for php5
This package provides modules for MySQL database connections directly from PHP #
scripts. It includes the generic "mysql" module which can be used to connect
to all versions of MySQL, an improved "mysqli" module for MySQL version 4.1 or
later, and the pdo_mysql module for use with the PHP Data Object extension.
PHP5 is a widely-used general-purpose scripting language that is especially
suited for Web development and can be embedded into HTML. The goal of the
language is to allow web developers to write dynamically generated pages
quickly. This version of PHP5 was built with the Suhosin patch.
Homepage: http://www.php.net/
```

Abbildung 2.4: Installation im aptitude ncurses-Interface.

Im Folgenden werden die Komponenten vorgestellt:

Apache

Um eine Seite an anfragende Webbrowser auszuliefern, sind mehrere Komponenten erforderlich. Eine zentrale Rolle spielt dabei der Webserver. Dieser Begriff wird auch für den virtuellen oder physischen Rechner verwendet, meint hier aber das ebenfalls „Webserver“ genannte Programm, welches eingehende Anfragen von Webbrowsern abarbeitet. Der Webserver mit der größten Verbreitung ist der Apache HTTP Server der *Apache Software Foundation*. Version 2.2.17 ist die aktuelle stabile Version der unter der Open-Source-kompatiblen Apache Lizenz frei verfügbar ist. Der Apache Webserver ist Dank seines modularen Auf-

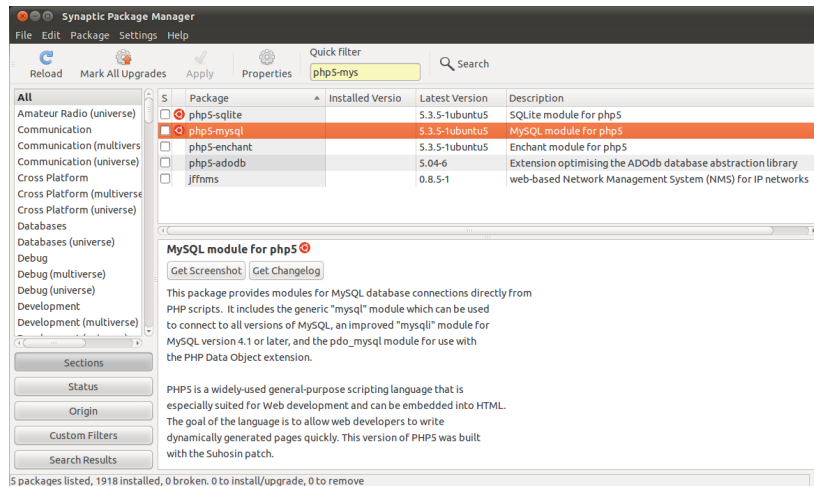


Abbildung 2.5: Installation über den Synaptic Package Manager.

baus einfach zu konfigurieren. Durch seine hohe Verbreitung sind seine Konfigurationsmöglichkeiten und Arbeitsweisen gut dokumentiert. Ein

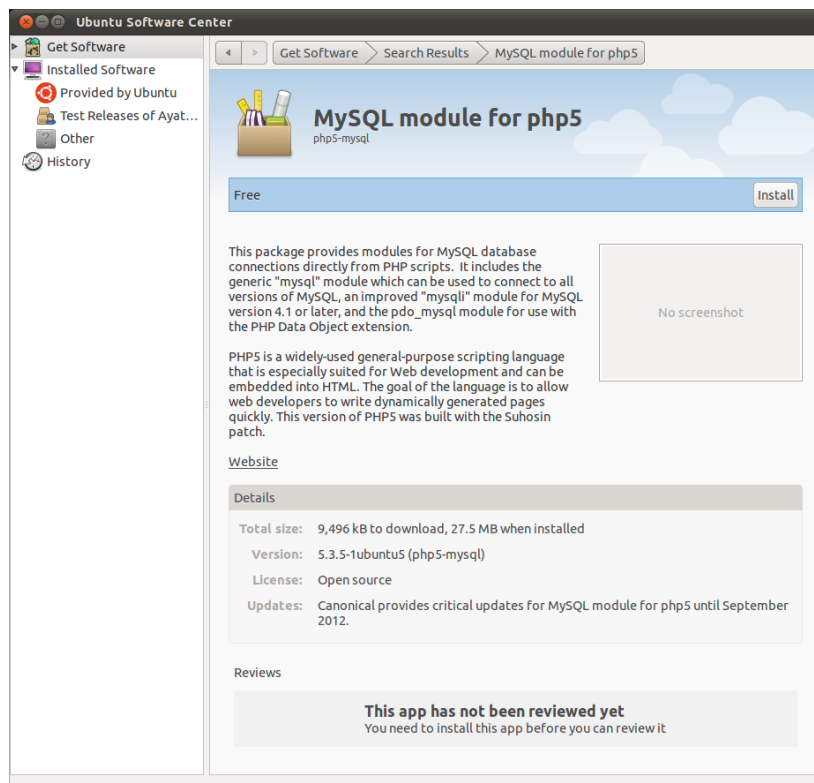


Abbildung 2.6: Installation über das Software-Center.

Nachteil von Apache ist, dass er in Bezug auf die Leistungsfähigkeit von neueren, sehr schlanken Webserverimplementierungen, wie beispielsweise *nginx* übertroffen wird. Durch die Verwendung eines Proxies, der Anfragen zwischenspeichert (*caching proxy*), kann dieser Nachteil meist ausgeglichen werden.

MySQL Datenbank

MySQL ist ein relationales Datenbankverwaltungssystem, auch *Relational Database Management System* (RDBMS) genannt. Für die Verwendung von MySQL fallen keine Gebühren an, das Projekt steht zudem unter der GNU *General Public License* (GPL). Der Eigentümer *Oracle Corporation* bietet aber auch kostenpflichtige MySQL-Versionen mit erweitertem Funktionsumfang für den kommerziellen Gebrauch an.

PHP, Perl oder Python

Die beiden “P” im Ausdruck LAMPP bezeichnen die Skriptsprachen PHP und je nach Konfiguration Perl oder Python. Gemeinsam mit MySQL erlauben sie die Erstellung dynamischer Webprojekte.

2.1.3 Deployment von LAMPP-Anwendungen

LAMPP Stack installieren

Nachfolgend sind die Möglichkeiten, LAMPP einzurichten, beschrieben:

Über den Paketmanager

Die einfachste und für Produktivumgebungen empfohlene Methode ist es, zur Installation den Paketmanager der Distribution zu verwenden. Dieser ist in der Lage, alle Abhängigkeiten korrekt aufzulösen und Konfigurationsdateien automatisiert anzupassen. Unter *Debian* werden Apache, PHP und MySQL mit dem Befehl `sudo apt-get install apache2 php5 libapache2-mod-php5` installiert.

Fertiges LAMPP-Paket

Besonders für Testumgebungen geeignet ist ein fertiges LAMPP-Paket, das alle notwendigen Programme enthält und eigenständig neben den bestehenden Systemkomponenten installiert werden kann. Der Hauptanbieter eines solchen Bundles ist das Apache-Friends-Projekt¹.

Fertiges Image

Die bequemste Variante, Webanwendungen zu installieren stellen Komplettpakete dar, bei welchen lediglich die gewünschte Anwendung gewählt werden muss und ein herunterladbarer Installer bereitgestellt wird.

¹<http://www.apachefriends.org/en/xampp-linux.html>

Tabelle 2.2: Paketbezeichnungen für den Apache Webserver in unterschiedlichen Systemen [2].

Betriebssystem	Paketbezeichnung
Debian	apache2
Fedora	httpd
FreeBSD	apache2
Gentoo	apache
NetBSD	apache2
OpenBSD	apache-httpd
Red Hat	httpd
SuSE	apache2
Ubuntu	apache2

Ein bekannter Anbieter dieser Variante ist Bitnami, das als Deployment-Komplettlösung verwendbar ist (siehe Abschnitt 5.4.2).

Selbst übersetzen

Da der Quellcode zu den Komponenten des LAMPP-Stacks frei zur Verfügung steht, besteht die Möglichkeit, die benötigten Komponenten von Hand zu übersetzen. Da hier alle Kompiliereinstellungen beeinflusst werden können ist das mit Abstand die flexibelste, aber auch zeitaufwendigste Methode.

Herausforderungen

Auf dem LAMPP-Stack basierende Webanwendungen bringen einige Abhängigkeiten mit sich. Eine grundlegende Voraussetzung ist, dass die von der Anwendung benötigten Komponenten installiert sind.

Wenn es sich um einen Server mit LAMPP-Diensten handelt müssen diese unter Umständen erst installiert werden. Falls zur Installation auf das vom System bereitgestellte Paketmanagement zurückgegriffen wird, muss beachtet werden, dass sich Paketnamen für die gleiche Softwarekomponenten zum Teil unterscheiden. Während sich das bei zentralen Komponenten wie dem Apache Webserver in Grenzen hält, kommen selbst dort von System zu System unterschiedliche Paketbezeichnungen zum Einsatz, wie Tabelle 2.2 zeigt.

Die Artenvielfalt unter den unixoiden Systemen bringt es mit sich, dass weder von einheitlicher Namensgebung ausgegangen werden kann, noch davon, dass der Ort für das Ablegen von Konfigurationsdateien standardisiert ist. Eine tabellarische Gegenüberstellung der Verzeichnisstrukturen von Linux, FreeBSD und Mac OS X wie sie in [3, S. 22] durchgeführt wurde, verdeutlicht, dass die selbe Art von Information in unixoiden Systemen je nach Betriebssystem an unterschiedlichen Stellen abgelegt wird. Dabei ist noch

nicht berücksichtigt, dass selbst unter den einzelnen Linuxdistributionen die Verzeichnisstruktur zum Teil unterschiedlich ist.

Bei Server-Distributionen gehören Apache und PHP häufig zur Grundinstallation, was bei MySQL, Perl oder Python nicht unbedingt der Fall ist. Sind diese grundlegenden Voraussetzungen gegeben, muss überprüft werden, ob in der Webserver-Software die von der Anwendung benötigten Module aktiviert wurden und generell so konfiguriert sind, dass es nicht zu Problemen bei der Ausführung der Webanwendungen kommt. Einer von vielen Fallstricken ist hier z. B. ein falsch konfiguriertes oder nicht vorhandenes Modul `mod_rewrite`.

Zudem besteht jede Webanwendung, die auf dem klassischen LAMPP-Stack basiert, stets aus ausführbaren und nicht ausführbaren Dateien sowie einem Datenbankteil, der mit diesen zusammenarbeitet. Die Datenbank erhöht die Komplexität beim Deployment. Ist die benötigte Software vorhanden, muss noch darauf geachtet werden, dass sie in der gleichen bzw. zumindest einer kompatiblen Version vorliegt. Zudem muss Rücksicht auf die korrekte Übernahme der Dateirechte sowie die richtige Zeichensatzkodierung genommen werden. Außerdem muss sichergestellt werden, dass Pfade und der Inhalt von Konfigurationsdateien an die Gegebenheiten des neuen Systems angepasst werden.

Mit zumindest einem Teil dieser Schwierigkeiten ist bei jedem Deployment einer Webanwendung zu rechnen, wenn Entwicklungsrechner und Produktivserver nicht exakt gleich aufgebaut sind. Unangenehm ist, dass gewisse Fehler selbst bei sorgfältigem Vorgehen auftreten, wobei sie sich oft erst bemerkbar machen, wenn sich die Anwendung im Produktivbetrieb befindet.

Kapitel 3

Virtualisierung

Der Begriff Virtualisierung ist in der Informatik weit verbreitet und wird für viele unterschiedliche Methoden der Ressourcenteilung verwendet, sodass eine eindeutige Definition des Begriffs nicht möglich ist. Das liegt zum Teil daran, dass Virtualisierung kein neues Konzept ist, sondern bei IBM bereits seit den späten Sechzigerjahren des vorigen Jahrhunderts erforscht und eingesetzt wird [4]. Bei den sich damals im Einsatz befindenden Mainframes wurde das Virtualisieren – Aufteilen der Ressourcen unter mehreren Benutzern – *Partitionieren* genannt. In der vorliegenden Arbeit ist mit Virtualisierung der Bereich der Virtualisierung unter x86 sowie jener der Software-Virtualisierung gemeint. Dieser beschäftigt sich mit dem Zur-Verfügung-Stellen virtueller Betriebssysteme oder Teilen des Betriebssystems. Bei der Virtualisierung wird ein Großteil der Befehle auf dem nativen Prozessor ausgeführt und nicht wie bei der Emulation die Hardware gänzlich in Software nachgebildet. Die meisten Virtualisierungstechnologien, so auch die in Abschnitt 3.4.1 auf Seite 20 vorgestellte *Kernel Virtual Machine* (KVM), kombinieren Verfahren der Virtualisierung und Emulation von Hardwarekomponenten.

Den Grundstein der Virtualisierung wie sie heute betrieben wird, haben G. Popek und R. Goldberg in ihrem 1974 veröffentlichtem Paper „Formal Requirements for Virtualizable Third Generation Architectures“ [5] gelegt. Darin wird festgelegt, dass virtualisierte Betriebssysteme im User-Mode (*Ring 3*) laufen und Befehle, die höhere Privilegien erfordern, an den *Virtual Machine Monitor*, auch *Hypervisor* (VMM) weitergegeben werden. Ebenfalls von großer Bedeutung ist eine Veröffentlichung von K. Adams und O. Agesen aus dem Jahre 2006 [6], in der sie einen Vergleich der damals neu eingeführten Hardware-gestützten Virtualisierung von AMD sowie Intel und Methoden der Software-Virtualisierung anstellen. Dass dabei Virtualisierung mittels Hardwareerweiterungen nicht sonderlich gut abschneidet, ist allerdings mit

der Tatsache, dass die beiden Autoren zum Zeitpunkt des Verfassens der Arbeit VMware-Mitarbeiter waren, in Kontext zu setzen.

Der VMM entscheidet, was mit dem übergebenen Befehl geschieht, führt ihn bei Bedarf auch aus und gibt das Resultat wieder an die virtuelle Maschine zurück. Dieser Prozess muss dabei so ablaufen, dass die virtuelle Maschine keinen Unterschied zwischen nativer Ausführung und der Ausführung über den Hypervisor merkt.

Zudem werden im Paper von Popek und Goldberg [5] drei wesentliche Eigenschaften des VMM festgelegt, die auch heute noch von Bedeutung sind:

1. *Exaktheit*

Abgesehen von zeitlichen Unterschieden darf sich die Ausführung eines Programms unter dem VMM nicht von der direkten Ausführung unterscheiden.

2. *Leistung*

Ein Großteil der Befehle des Gasts werden direkt von der Hardware ohne das Eingreifen des VMM ausgeführt.

3. *Sicherheit*

Der VMM regelt sämtliche Zugriffe auf die Hardwareressourcen.

Besonders zwei Konzepte tragen dazu bei, Virtualisierung effizienter zu gestalten. Zum einen ist das die unter Abschnitt 3.2 vorgestellte Paravirtualisierung, zum anderen die unter Abschnitt 3.3 erläuterten Hardwareerweiterungen.

3.1 Typ 1 und Typ 2 Virtual Machine Monitor

Durch sein bereits 1973 veröffentlichtes Paper „Architecture of Virtual Machines“ prägte Golberg auch einen weiteren Aspekt der Virtualisierung: Die Unterteilung von VMM in Typ 1 und Typ 2 [7]:

- Ein *Typ-1-VMM* läuft direkt auf der Hardware des Hosts und wird deshalb auch *nativer VMM* oder *bare-metal-VMM* genannt. Ein Gastsystem läuft direkt eine Ebene über dem Hypervisor.
- Der *Typ-2-VMM* hingegen läuft innerhalb eines üblichen Betriebssystems. Neben Betriebssystem und Hypervisor ist das Gastsystem damit die dritte Ebene über der Hardware.

Der Vorgang, kritische Prozessoraufrufe aus Gästen abzufangen und zu emulieren, wird *Trap and Emulate* genannt. Dieser Prozess kann sowohl in Software als auch in Hardware ablaufen, wird jedoch auf älterer x86-Hardware nicht unterstützt. Der Grund, wieso ältere Hardware nicht unterstützt wird, sind 17 der insgesamt 250 Befehle des Prozessors, welche eine der Grund-

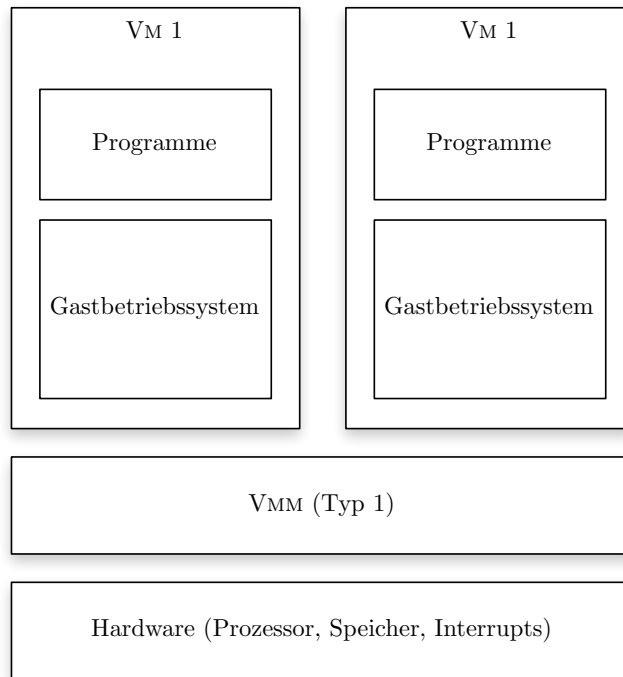


Abbildung 3.1: Typ-1-VMM, in Anlehnung an [8].

voraussetzungen für VMM-Hardware nicht erfüllen. Es geht dabei um die Bedingung, den VMM zu informieren, sobald eine virtuelle Maschine vorhat, kritische Befehle auszuführen. Diese 17 Befehle sind kritisch in der Ausführung, werden jedoch im unprivilegierten Modus ausgeführt und erzeugen daher keine Unterbrechung oder Ausnahme [10].

Im Falle eines Typ-1-Hypervisors ruft ein Befehl, der den privilegierten Modus erfordert, den Hypervisor auf, welcher den Grund des Aufrufs ermittelt und den Trap-Handler des Gastsystems aufruft. Nach der Ausnahme wird wieder in den Hypervisor gewechselt, um den Status des Programms, das den Aufruf aus dem Gast heraus getätigt hat, wiederherzustellen. Wie in Abbildung 3.3 gezeigt, wird bei einem solchen Systemaufruf vier mal der Kontext gewechselt.

Die Kontextwechsel zusammengefasst:

1. Programm in der *virtuellen Maschine* (VM) – VMM
2. VMM – VM
3. VM – VMM
4. VMM – Programm in der VM

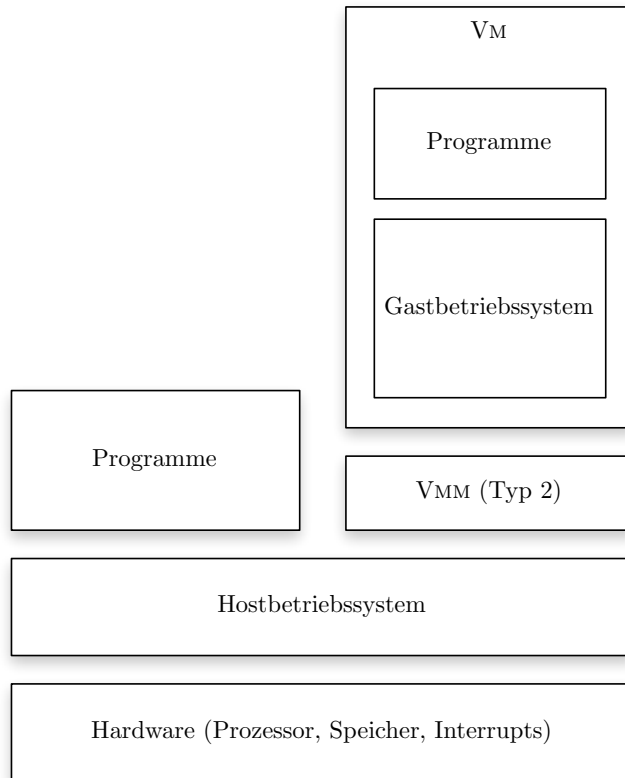


Abbildung 3.2: Typ-2-VMM, in Anlehnung an [8].

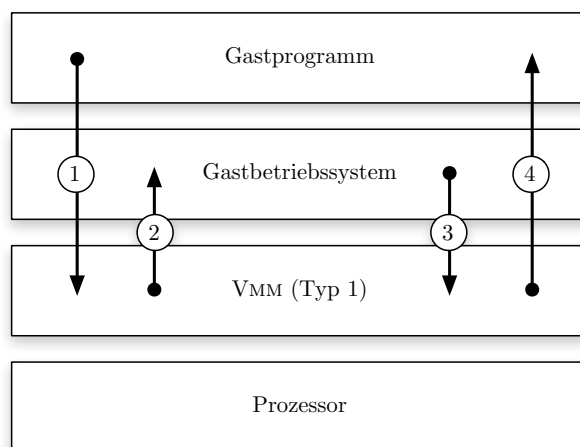


Abbildung 3.3: Ausführen privilegierter Befehle in Typ 1 VMM, in Anlehnung an [9].

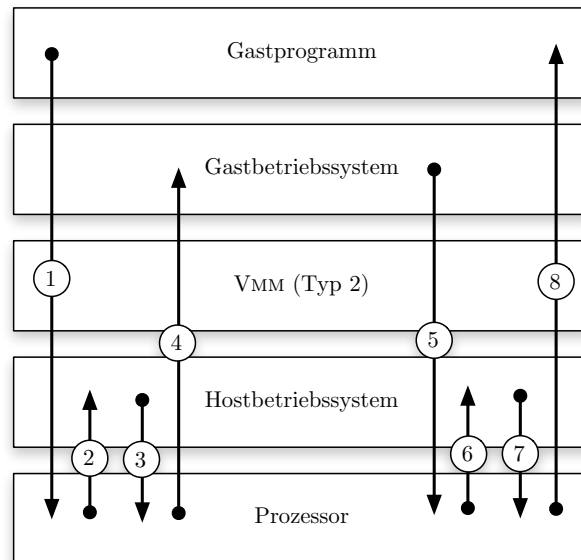


Abbildung 3.4: Ausführen privilegierter Befehle in Typ 2 VMM, in Anlehnung an [9].

Kommt hingegen ein Typ-2-Hypervisor zum Einsatz, registriert das Hostsystem den Aufruf. Das Hostsystem weiß, dass es für den erfolgten Aufruf der VMM zuständig ist und ruft diesen auf. Dieser wiederum weiß, dass es sich um einen Aufruf der VM handelt und fragt das Hostsystem um Erlaubnis, die Kontrolle der VM zu übergeben. Diese führt den Aufruf aus und versucht zur betreffenden Anwendung zurückzuwechseln. Dadurch wird erneut ein Kontextwechsel in das Hostsystem ausgelöst. Der Kontextwechsel wird vom Hypervisor übernommen, welcher den Host aufruft um zur Anwendung zurückzukehren. Dieser Vorgang wird in Abbildung 3.4 dargestellt.

Die Kontextwechsel eines Typ-2-VMM:

1. Programm in der VM – Hostsystem
2. Hostsystem – VMM
3. VMM – Hostsystem
4. Hostsystem – VM
5. VM – Hostsystem
6. Hostsystem – VMM
7. VMM – Hostsystem
8. Hostsystem – Programm in der VM

Aktuelle Prozessoren mit den in Absatz 3.3 vorgestellten Hardwareerweiterungen unterstützen diesen *Trap-and-Emulate* Ansatz.

Tabelle 3.1: Schlüsselwörter für Hardwareerweiterungen in `/proc/cpuinfo`.

Schlüsselwort	CPU
<code>vms</code>	Intel
<code>vmx</code>	AMD

In der Literatur wird der *Trap-and-Emulate*-Ansatz häufig auch als *klassische Virtualisierung* bezeichnet.

3.2 Paravirtualisierung

Paravirtualisierung stellt eine Schnittstelle bereit, die der physikalischen Hardware ähnlich ist. Bei der Paravirtualisierung wird das Betriebssystem modifiziert und für die jeweilige Virtualisierungslösung angepasst. Dadurch besitzt das virtualisierte Betriebssystem Kenntnis über die Befehle, deren Ausführung in der virtualisierten Domäne aufwendig oder gar kritisch ist. Diese Befehle können vom VMM sehr einfach verarbeitet und anschließend entsprechend weitergegeben werden. Dadurch ergibt sich eine erhöhte Leistungsfähigkeit, da das explizite Abfangen von privilegierten Befehlen nicht durchzuführen ist.

Die erforderliche Portierung ist bei proprietären Betriebssystemen wie Microsoft Windows oder Mac OS X ausgeschlossen, da der Quellcode nicht zur Verfügung steht. Selbst wenn dieser zur Verfügung stünde, wäre es aus rechtlichen Gründen nicht erlaubt, diesen zu verändern. Aus diesem Grund sind ausschließlich quelloffene Betriebssysteme wie Linux und BSD-Derivate für Paravirtualisierung geeignet.

3.3 Hardwareerweiterungen

Hardwareerweiterungen unterstützen die Virtualisierung, indem sie das gleichzeitige Benutzen von Ressourcen auf CPU-Ebene ermöglichen.

Die Bezeichnungen für diese Hardwareerweiterungen lauten AMD-V oder *Pazifca* bei AMD und VT-x oder *Vanderpool* bei Intel. In Bezug auf die bei der Virtualisierung besonders wichtigen Sicherheitsaspekte ist diese in Hardware gegessene Lösung einem Software-Hypervisor vorzuziehen.

In der Praxis lässt sich unter Linux mit einem Blick in die Datei `/proc/cpuinfo` feststellen, ob Hardwareunterstützung verfügbar ist. Im Abschnitt „`flags`“ muss eines der in Tabelle 3.1 angeführten Schlüsselwörter vermerkt sein [11].

3.4 Virtualisierungstechnologien

An dieser Stelle werden die für diese Arbeit relevanten Virtualisierungstechnologien vorgestellt. Zusammengefasst ist dabei festzustellen, dass die Effizienz der jeweiligen Lösung von Systemvirtualisierung über Paravirtualisierung bis hin zu KVM steigt.

3.4.1 Systemvirtualisierung

Bei dieser Art der Virtualisierung ist der Hypervisor, auch Virtual Machine Monitor genannt, für die Ressourcenverteilung unter den virtuellen Hosts zuständig. Er agiert dabei als eine Art übergeordnetes Betriebssystem und erlaubt so den parallelen Betrieb mehrerer virtueller Gastsysteme.

VMware Workstation und VirtualBox

VMware Workstation sowie *VirtualBox* emulieren Standard-PC-Hardware im virtuellen Gast und setzen Hardwarezugriffe in Systemaufrufe auf dem Host um. VMware waren die ersten, deren Produkte am Markt der x86-Virtualisierung weite Verbreitung fanden. Prinzipiell kann ein VMware-Betriebssystem ohne jegliche Veränderungen von jedem gängigen Installationsmedium installiert werden. Sämtlicher Code wird dabei vom VMM kontrolliert und bei Bedarf an die physikalische Hardware weitergereicht. Das Verfahren wird von VMware optimiert, indem auf charakteristische Eigenschaften des jeweiligen Systems eingegangen wird. Zusätzlich besteht die Möglichkeit, paravirtualisierte Treiber, die sogenannten *VMware Tools*, am virtualisierten Rechner zu installieren.

Eine neuere Entwicklung nach dem gleichen Prinzip ist *VirtualBox*, das von *Sun Microsystems* ins Leben gerufen wurde und jetzt im Besitz der *Oracle Corporation* ist. Das Prinzip der Virtualisierung ist gleich wie bei den Produkten von VMware.

Xen

Im Gegensatz zu den in Abschnitt 3.4.1 vorgestellten Virtualisierungslösungen läuft Xen direkt auf der Hardware (*bare metal*) und wendet das unter Abschnitt 3.2 erläuterte Prinzip der Paravirtualisierung an. Das dass damit nur modifizierte Betriebssysteme verwendet werden können. Xen nutzt einen speziellen Gast, der im *Ring 0* des Ringsystems läuft und gewissermaßen auch als Teil des VMM betrachtet werden kann.

Diese privilegierte Domäne wird als *Domäne 0*, kurz *Dom0*, bezeichnet. Die anderen virtualisierten Betriebssysteme laufen als unprivilegierte Domäne,

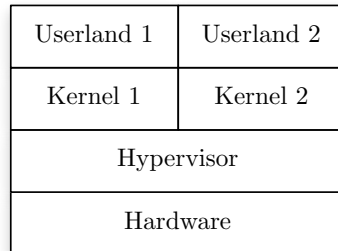


Abbildung 3.7: Schematischer Aufbau von Xen (nach [12]).

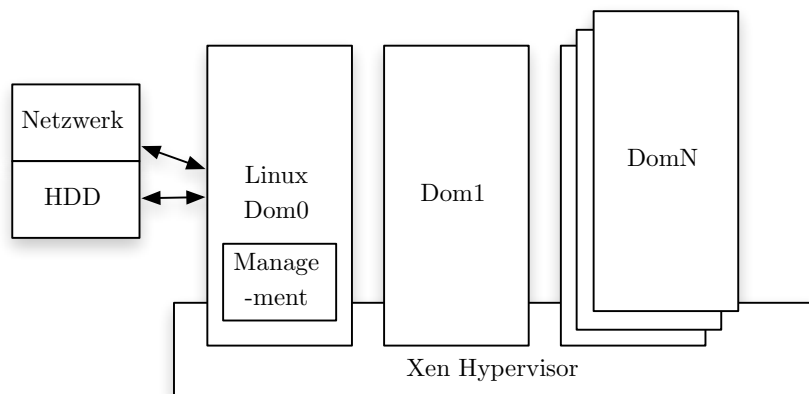


Abbildung 3.8: Domänen in Xen (nach [12]).

Xen erlaubt aber auch das Ausführen von unmodifizierten Betriebssystemen. Im Xen-Jargon wird dies *Hardware Virtual Machine* (HVM) genannt und setzt eine der in Absatz 3.3 vorgestellten Hardwareerweiterungen voraus.

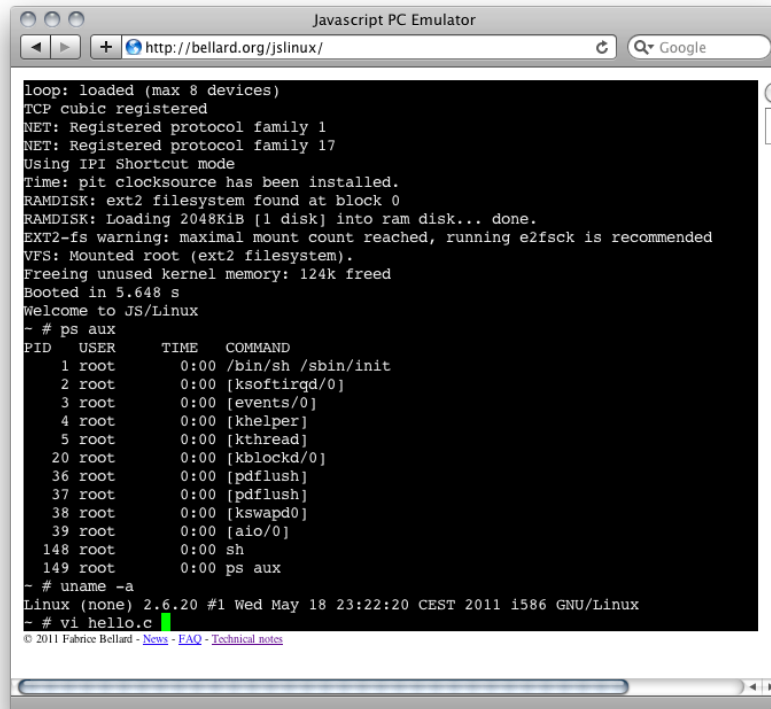
Wie bei den meisten erfolgreichen Projekten aus dem Umfeld von FOSS steht auch hinter Xen ein professionelles Entwicklerteam. Geleitet wird die Entwicklung von *Citrix Systems*.

Xen ist in der Entwicklung problematisch, da der Linux-Kernel angepasst werden muss. Die Arbeit der Xen-Entwickler basiert dabei auf der über drei Jahre alten Kernel-Version 2.6.18. Wird Xen mit einer Versionsnummer unter vier mit einem moderneren Kernel betrieben, so wurden sämtliche Patches von den Entwicklern der jeweiligen Distribution geschrieben. Dieser hohe Anpassungsaufwand ist der Hauptgrund dafür, dass Kernel-Programmierer ihrerseits Xen-Funktionalität nur zögerlich in den Hauptzweig der Kernelentwicklung einpflegen. Das im folgenden Abschnitt 3.4.1 beschriebene KVM hingegen konnte dank ihres schlanken, modularen Konzepts umgehend in den Kernel übernommen werden.

Eine Verbesserung der Xen-Situation wird Linux ab Version 3.0 mit sich bringen, da ab dieser Version sämtliche für Xen benötigte Funktionen in die Kernel-Hauptlinie eingepflegt werden [13]. Der Versionsnummernsprung auf 3.0 geht nicht mit einer Fülle neuer Funktionen einher, sondern ist eine kosmetische Maßnahme, um die Revisionsnummer des Kernels nicht weiter ansteigen zu lassen. Dass diese Versionsnummernänderung zeitlich mit der Integration der Xen-Funktionen zusammenfällt ist Zufall.

KVM

KVM ist jene Virtualisierungslösung die den stärksten Benutzerzuwachs verzeichnen kann. Dieses Bild ergibt sich nicht nur durch Beobachtung der Marktsituation, sondern wird auch in der Literatur, wie z. B. [14, S. 518] widergespiegelt. Dieser Benutzerzuwachs geht fast ausschließlich auf Kosten von Xen, dem bisherigen Marktführer im Segment der Servervirtualisierung. Namhafte Anbieter im Bereich der Virtualisierung unter Linux, wie z. B. Red Hat, haben bereits begonnen, den Umstieg von Xen auf KVM einzuleiten. So wird im aktuellen RHEL der Version 6 nur noch KVM und kein Xen mehr unterstützt. Das liegt vor allem am schlanken Konzept der KVM. Entsprechend der UNIX-Philosophie wird hier ein Werkzeug zur Verfügung gestellt, das nur eine kleine Aufgabe erledigt und für alle weiteren Aufgaben bereits bestehende Programme einsetzt. Im Wesentlichen ist KVM ein Treiber für die Hardwarevirtualisierung der in Form eines Gerätes `/dev/kvm` zur Verfügung gestellt wird. Dementsprechend sind Hardwareerweiterungen Grundvoraussetzung bei der Verwendung von KVM. Für die beiden Hardwareerweiterungen VT-x und AMD-V (siehe Abschnitt 3.3) gibt es jeweils eine angepasste Version des Kernelmoduls `kvm.ko`, das mittels `modprobe` einzubinden ist. Damit ist durch KVM nur der Prozessor virtualisiert, die restliche Hardware wird emuliert. Zur Emulation kommt meist eine angepasste Version von QEMU zum Einsatz. Der Unterschied zwischen Virtualisierung und Emulation ist, dass ein Emulator den Prozessor gänzlich in Software nachbildet, während Virtualisierer auf die Hardware zugreifen. QEMU lässt sich ohne KVM als reiner Emulator verwenden, die Leistungsfähigkeit ist dann allerdings nicht vergleichbar mit jener von KVM in Verbindung mit QEMU. Ein neuer Emulator des Hauptautors von QEMU heißt JS/Linux und ist ein in JavaScript verfasster Linuxemulator. Bei QEMU und dem in Abbildung 3.9 dargestellten JS/Linux handelt es sich aber nicht um Virtualisierung sondern um Emulation, die häufig in Kombination mit Virtualisierung verwendet wird. Damit stellt QEMU nur eine Komponente der Virtualisierung dar, die dank des modularen Aufbaus dieser Virtualisierungslösung auch durch andere



```
loop: loaded (max 8 devices)
TCP cubic registered
NET: Registered protocol family 1
NET: Registered protocol family 17
Using IPI Shortcut mode
Time: pit clocksource has been installed.
RAMDISK: ext2 filesystem found at block 0
RAMDISK: Loading 2048KiB [1 disk] into ram disk... done.
EXT2-fs warning: maximal mount count reached, running e2fsck is recommended
VFS: Mounted root (ext2 filesystem).
Freeing unused kernel memory: 124k freed
Booted in 5.648 s
Welcome to JS/Linux
- # ps aux
PID  USER    TIME  COMMAND
  1  root      0:00  /bin/sh /sbin/init
  2  root      0:00  [ksoftirqd/0]
  3  root      0:00  [events/0]
  4  root      0:00  [khelper]
  5  root      0:00  [kthread]
 20  root      0:00  [kblockd/0]
 36  root      0:00  [pdflush]
 37  root      0:00  [pdflush]
 38  root      0:00  [kswapd0]
 39  root      0:00  [aio/0]
148  root      0:00  sh
149  root      0:00  ps aux
- # uname -a
Linux (none) 2.6.20 #1 Wed May 18 23:22:20 CEST 2011 i586 GNU/Linux
- # vi hello.c
© 2011 Fabrice Bellard - News - FAQ - Technical notes
```

Abbildung 3.9: Js/Linux.

ersetzt werden kann. Nachdem einige Entwickler mit der Leistungsfähigkeit von QEMU nicht zufrieden waren, sind Alternativen bereits in Planung [15].

Der Ansatz von KVM ist am ehesten mit der Verwendung einer HVM unter Xen zu vergleichen.

Kvm Einrichten

Während das Einrichten von KVM auf der Kommandozeile gut dokumentiert ist, steht Servern mit grafischer Oberfläche auch das Programm *Virtual Machine Manager* zur Verfügung, mit dem sich virtuelle Maschinen besonders einfach verwalten lassen (siehe Abbildung 3.11).

Um ein Netzwerkgerät, das *Preboot Execution Environment* (PXE) unterstützt in der virtuellen Maschine zu verwenden, wird wie in Listing 3.1 ein Bridge-Interface in `/etc/network/interfaces` angelegt. Wird das netzwerkbasierte Booten nicht benötigt, kann das Netzwerk für die virtuelle Betriebssysteminstanz auch über *Network Address Translation* (NAT) zur Verfügung gestellt werden.

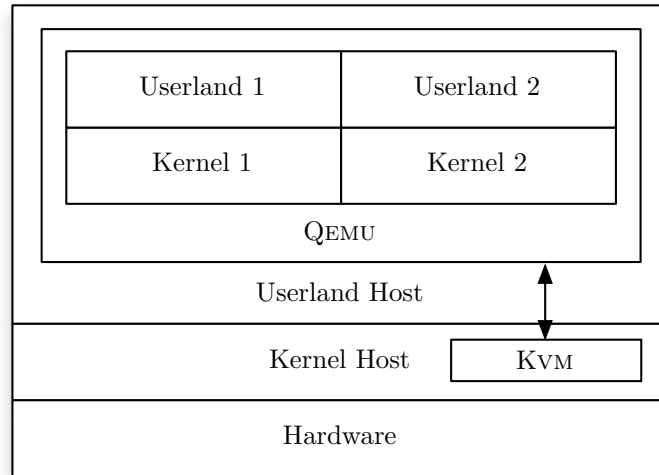


Abbildung 3.10: Schematischer Aufbau von KVM (nach [12]).

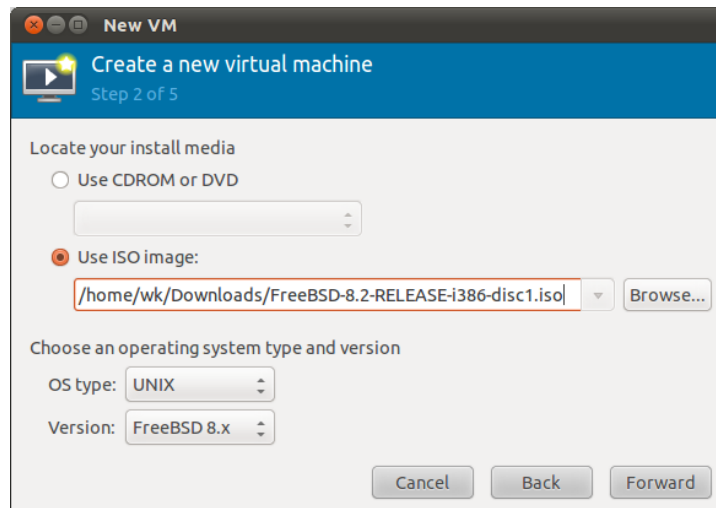


Abbildung 3.11: Hinzufügen einer Instanz mit dem Virtual Machine Manager.

Listing 3.1: Bridge-Interface in /etc/network/interfaces

```

1 auto br0
2 iface br0 inet static
3     address 192.168.1.173
4     network 192.168.1.0
5     netmask 255.255.255.0
6     broadcast 192.168.1.255
7     gateway 192.168.1.1
8     bridge_ports eth0
9     bridge_fd 9
10    bridge_hello 2
11    bridge_maxage 12
12    bridge_stp off

```

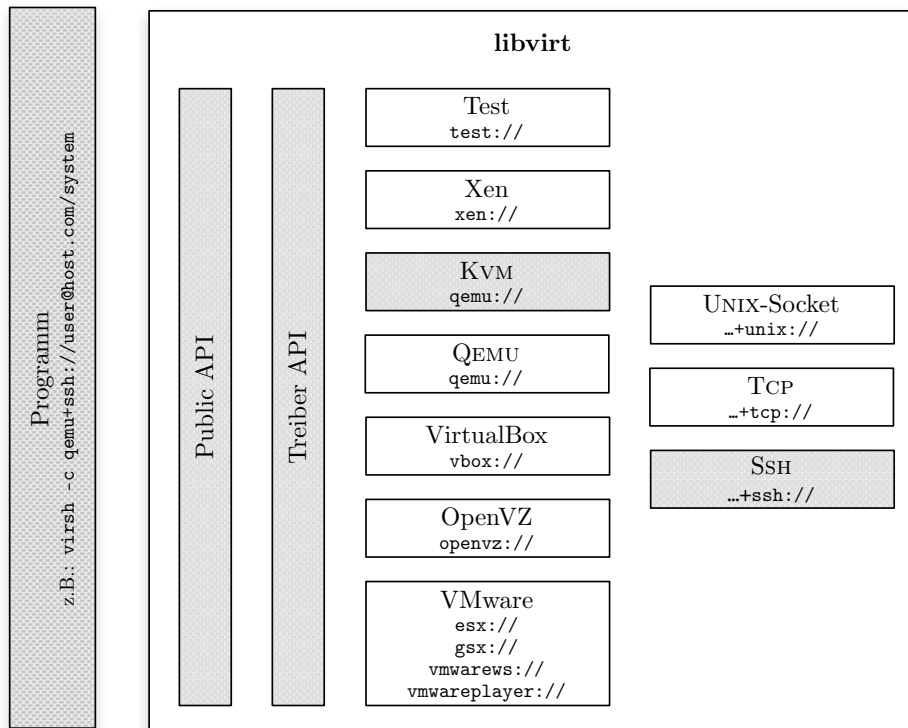


Abbildung 3.12: Funktionsweise von libvirt.

3.5 libvirt, virsh und Virtual Machine Manager

Die hier beschriebenen Werkzeuge stellen keine eigenständige Virtualisierungstechnologie dar, sondern bieten Werkzeuge und Bibliotheken, um unterschiedliche Virtualisierungslösungen einheitlich verwalten zu können. Bei **libvirt** handelt es sich um eine Bibliothek, die den Zugriff auf unterschiedliche Virtualisierungslösungen abstrahiert. Zur Bedienung von **libvirt** eignet sich vor allem das Kommandozeilenprogramm **virsh** oder das in Python geschriebene und in GIMP *Toolkit* (GTK+) verfasste, grafische Frontend *Virtual Machine Manager*.

Angesprochen wird **libvirt** über den Aufruf eines *Uniform Resource Identifier* (URI) von einem Programm wie z. B. **virsh** aus. Die unterstützten Virtualisierungstechnologien, sowie der generelle Aufbau von **libvirt** ist Abbildung 3.12 zu entnehmen. Der verwendete URI beinhaltet die Art der Verbindung. Diese setzt sich aus einer Kurzbezeichnung und optionalen, mit einem `+`-Symbol verbundenen Verbindungsparametern zusammen. Ein gültiger URI ist z. B. `virsh -c qemu+ssh://user@host.com/system`. Eine Sonderstellung nimmt der Testtreiber ein, der wie der Name schon sagt,

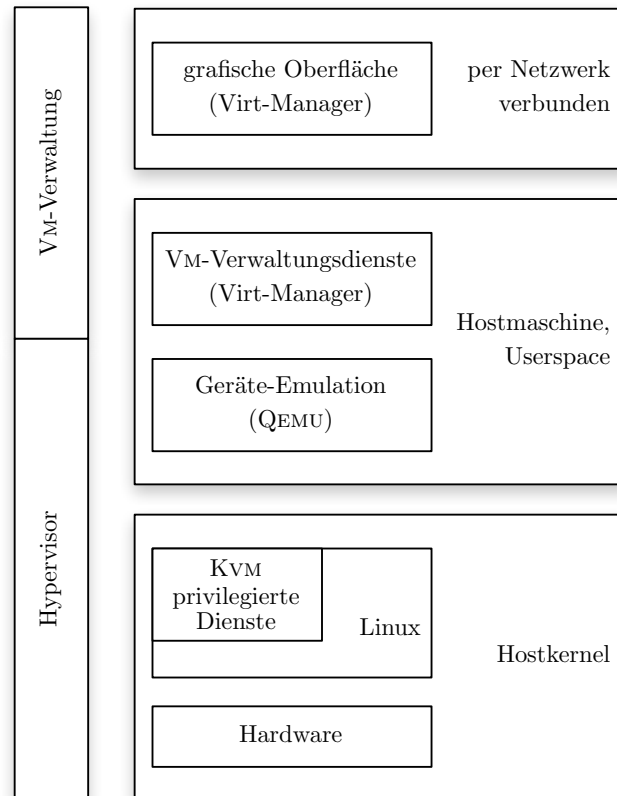


Abbildung 3.13: Detaillierter Aufbau von KVM (nach [12]).

für Testzwecke einen Hypervisor vortäuscht. Ebenfalls erwähnenswert ist das Aufbauen einer KVM-Verbindung. Hierfür wird, wie bei einer direkten QEMU-Verbindung `qemu://` verwendet und `libvirt` baut automatisch eine KVM-Verbindung auf, wenn die notwendigen KVM-Komponenten aktiv sind. Zu den Verbindungsoptionen gehören, wie in Abbildung 3.12 zu sehen, eine Verbindung über das *Transmission Control Protocol* (TCP), UNIX-Sockets oder SSH.

3.5.1 Betriebssystemvirtualisierung

Die Betriebssystemvirtualisierung stellt eine abgeschlossene Umgebung für die Virtualisierung zur Verfügung. Die Gastprozesse sind für andere Gastssysteme dabei nicht sichtbar und können auch nur auf den ihnen zugewiesenen Bereich des Dateisystems zugreifen. Wesentlicher Unterschied zur Systemvirtualisierung mittels Hypervisor ist, dass sich die virtuellen Systeme sowie der Host einen gemeinsamen Kernel teilen. Das bedeutet, dass die virtuellen

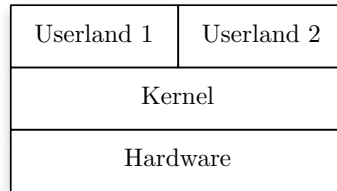


Abbildung 3.14: Schematischer Aufbau Betriebssystemvirtualisierung (nach [12]).

Betriebssysteme vom gleichen Typ sein müssen wie das Hostsystem. Das kann je nach Anwendungsfall eine Einschränkung darstellen, selbst Linuxsysteme lassen sich nicht beliebig kombinieren. Die Betriebssystemvirtualisierung eignet sich gut für die Virtualisierung von Servern, da sie eine sehr schlanke Virtualisierungsschicht mit geringem Virtualisierungs-overhead erlaubt.

User-mode Linux (UML)

User-mode Linux (UML) erlaubt die Ausführung von Gästen innerhalb von Linux Systemen. Während UML ursprünglich als Patch für den Linux-Kernel erhältlich war, ist es seit der Kernelversion 2.6 bereits integriert. Betriebssysteme mit einem solchen Kernel lassen sich als UML-Prozesse in einem Linuxhost ausführen. UML erlaubt die Ausführung unterschiedlicher Linuxderivate und Kernelversionen und wird daher gerne zum Testen oder Debuggen verwendet. Auch Honeypots können damit realisiert werden. Ein Nachteil von UML gegenüber Virtualisierungstechniken wie Xen oder KVM ist, dass es nicht so leistungsstark ist.

chroot, BSD Jails und Solaris Containers

Eine **chroot**-Umgebung unter unixoiden Systemen ermöglicht es, bestimmten Prozessen und Benutzern ein „virtuelles“ Rootverzeichnis im tatsächlichen Verzeichnisbaum zur Verfügung zu stellen. Prozessen ist es nicht möglich, eine solche **chroot**-Umgebung zu verlassen („auszubrechen“), woraus sich auch der Name *chroot jail* ergeben hat. Besonders für kritische Dienste finden **chroot**-Umgebungen nach wie vor Anwendung, häufig wird z. B. empfohlen, *Berkeley Internet Name Domain* (BIND), einen Server für das *Domain Name System* (DNS), in einer solchen zu betreiben. Ein entscheidender Nachteil von **chroot**-Umgebungen ist jedoch, dass **chroot**, abgesehen von den Beschränkungen für das Dateisystem, Vollzugriff auf die Systemressourcen hat. Damit kann durch Zugriff auf Ressourcen des Rechners von einer **chroot**-Umgebung aus, mutwillig oder aufgrund eines Fehlers, das gesamte System blockiert werden.

Die beiden Virtualisierungstechnologien BSD Jails und Solaris Containers greifen das Konzept der **chroot**-Umgebungen auf, erweitern dieses aber um Aspekte der Virtualisierung. Konzeptionell handelt es sich bei den BSD Jails und den Containers unter Solaris um den gleichen Ansatz, weshalb sie hier gemeinsam behandelt werden.

Ein Nachteil der BSD-Jails ist, dass Funktionen explizit implementiert werden müssen. Ein Beispiel dafür ist die Unterstützung von *Internet Protocol Version 6* (IPv6), welche in FreeBSD schon lange zur Verfügung stand, für Jails aber erst mit Version 7.3 eingeführt wurde.

OpenVZ/Virtuozzo

OpenVZ ist eine quelloffene Lösung für Betriebssystemvirtualisierung. Es erlaubt die Verwendung mehrerer Betriebssysteminstanzen unter demselben physikalischen System und Kernel. Eine solche virtuelle Instanz wird als *Container* oder *Virtual Private Server* (VPS) bezeichnet.

Parallels, ein auf Virtualisierung spezialisiertes Unternehmen ist maßgeblich an der Entwicklung von OpenVZ beteiligt und hat darauf basierend das proprietäre Format der Virtuozzo-Container entwickelt.

Abbildung 3.15 zeigt die Ressourcenverwaltung eines solchen Systems während in Abbildung 3.16 eine Ausgabe von Systemprozessen, ähnlich der eines `top`-Befehles auf der Kommandozeile für dasselbe System, dargestellt ist. Wird für das gezeigte System `uname -r`, aufgerufen um die Kernelversion zu erhalten, so handelt es sich wenig überraschend, aus den in Abschnitt 3.4.1 erläuterten Gründen, um eine angepasste 2.6.18-er-Version (2.6.18-028stab070.14).

Parallels bietet auch weitere Produkte an, beispielsweise ist *Parallels Desktop* neben *VMware Fusion* und *VirtualBox* eines der meist verbreitetsten Desktopvirtualisierungsprogramme für Mac OS X. Neben dieser Consumer-Variante gibt es noch *Parallels Server* für Mac OS X.

3.5.2 Virtual Appliances

Software-Komplettlösungen, die für die Ausführung auf bestimmten virtualisierten Plattformen vorbereitet wurden, werden auch *Virtual Appliance* genannt. Sie zeichnen sich dadurch aus, dass die später zu verwendende Software bereits vorinstalliert und -konfiguriert zur Ausführung bereitsteht. Das erleichtert das Deployment von Anwendungssoftware erheblich. Die Kostenersparnis ist umso größer, je öfter eine *Virtual Appliance* zum Einsatz kommt. Für ein Unternehmen mit sehr heterogenen Software-Stacks sind sie daher weniger interessant als für *Internet Service Provider* (ISP), welche

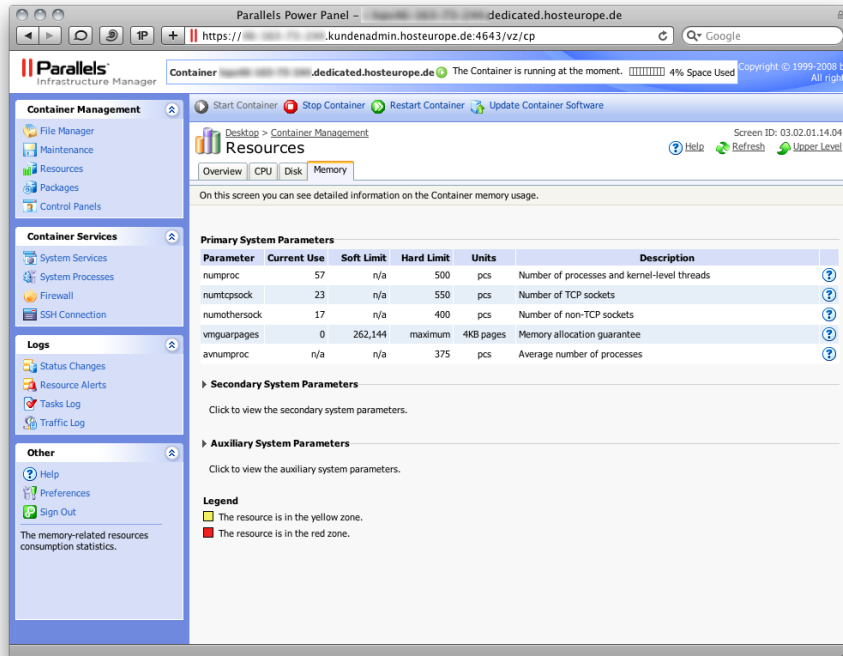


Abbildung 3.15: Ressourcenverwaltung in Parallels Virtuozzo.

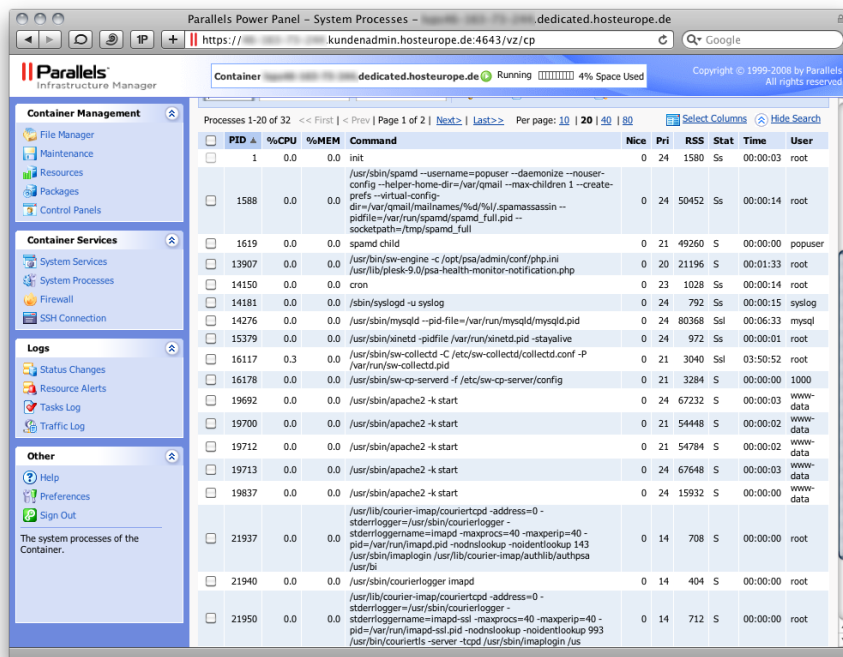


Abbildung 3.16: Systemprozesse einer Instanz in Parallels Virtuozzo.

Tausende gleich aufgebaute VPS einsetzt. Viele der virtuellen Appliances bieten ein eigenes Webinterface zur Konfiguration.

3.6 Vorteile der Virtualisierung

Serverkonsolidierung

Bei der Leistung heutiger Server kann von effizienter Nutzung oft nicht die Rede sein, wenn ihnen nur eine Aufgabe zugeteilt wird. Daher werden einem physikalischen Rechner häufig mehrere Aufgaben überlassen. Dies wird mit virtuellen Maschinen realisiert.

Sicherheit

Ein wesentlicher Grund für die Verwendung von Virtualisierung ist die erhöhte Sicherheit durch die Zuweisung einzelner Systemaufgaben zu einzelnen virtuellen Maschinen, im Vergleich zu einem physischen Rechner, der diese Aufgaben gemeinsam im selben Kontext verwaltet. Mittels Virtualisierung können Aufgaben getrennt werden. Wird beispielsweise der Webserver kompromittiert, so hält sich der Schaden in Grenzen, da die Angreifer nur Zugang zur virtuellen Maschine erhalten können, in welcher ausschließlich der Webserver läuft. Dies gilt für den Idealfall, theoretisch besteht die Möglichkeit, von einem virtuellen System auszubrechen, um auf ein anderes virtuelles System oder gar den Host zuzugreifen, wie das bei den Nachteilen auch angeführt wird. Zu bedenken ist daher, dass physisch getrennte Systeme noch sicherer sind als zwei virtuelle Maschinen auf demselben Rechner.

Lastbalancierung

Da ein virtuelles System nicht direkt an die Hardware gebunden ist, kann eine Lastbalancierung durch einfaches Verschieben der virtuellen Maschine auf einen anderen Rechner realisiert werden.

Portabilität und Migration

Nicht selten findet sich Produktivsoftware im Einsatz, welche unter modernen Betriebssystemen und aktueller Hardware nicht mehr eingesetzt werden kann. Bei virtuellen Systemen stellt sich dieses Problem nicht in diesem Ausmaß, da virtuelle Maschinen problemlos auf aktuelle Systeme migriert werden können. Gerade in Hinblick auf die rasche Entwicklung im Softwarebereich kann es sich rechnen, gut funktionierende aber nicht mehr erhältliche und nur mehr schwer zu wartende Systeme in virtuelle Maschinen zu übernehmen. Bei alten Systemen – virtualisiert oder nicht – muss natürlich besonders auf den Aspekt der Sicherheit geachtet werden, denn der Nachteil solcher Systeme ist, dass keine aktuellen Sicherheitspatches bereitgestellt werden.

Virtual Appliances

Die Möglichkeit, eine Virtual Appliance, wie sie in Abschnitt 3.5.2 vorgestellt wird, zu verwenden, ist ein weiterer Vorteil von virtualisierten Systemen. Die vorkonfigurierten, einsatzbereiten virtuellen Maschinen helfen, die Deploymentkosten zu senken.

Hardwareänderungen

Wachsen die Anforderungen an den Server oder fallen physische Hardwarekomponenten aus, kann problemlos auf ein anderes System gewechselt werden und die virtuelle Maschine kann dort weiterbetrieben werden.

Sicherung/Wiederherstellung

Vergleichsweise einfach gestaltet sich auch das Backup und die Wiederherstellung einer virtuellen Maschine. Dadurch, dass die virtuelle Maschine im Regelfall nicht an einen bestimmten physischen Rechner gebunden ist, lässt sich die Sicherung bei Hardwareproblemen auch auf anderer Hardware einspielen. In [16] werden Möglichkeiten aufgezeigt, Backups in virtualisierten Umgebungen effizient durchzuführen.

Vermeidung von Versionskonflikten

Die Zuweisung einzelner Serverdienste zu virtuellen Maschinen bringt den Vorteil, dass der für den Dienst benötigte Stack von Anwendungen und Bibliotheken isoliert vorliegt. Bringt ein weiterer Serverdienst andere Anforderungen an Software mit, so lassen sich diese ohne Rücksicht auf Kompatibilität zu den in anderen virtuellen Maschinen installierten Abhängigkeiten einrichten.

Testumgebung

Virtualisierung ist bestens dafür geeignet, kostengünstige Testsysteme zu erstellen und Konfigurationen zu überprüfen, bevor sie in den Produktivbetrieb übernommen werden.

Umweltfreundlichkeit

Durch die zusätzlich eingeführte Abstraktion zwischen Hardware und Betriebssystem lassen sich Server effizienter betreiben und Ressourcen dynamisch ab- oder anschalten, was sich in einem geringeren Energiebedarf widerspiegelt.

3.7 Nachteile der Virtualisierung

Sicherheit

Sicherheit wurde bereits bei den Vorteilen von Virtualisierung genannt. Während die Aufteilung von Serverdiensten in einzelne virtuelle Einheiten tatsächlich einen Zugewinn an Sicherheit bedeutet, sind mit der Virtualisierung auch eine Reihe von Gefahren verbunden. Das Problem

dabei ist, dass ein Angreifer auf sämtliche Hosts zugreifen kann, wenn es ihm gelingt, von einer übernommenen virtuellen Maschine aus den Hypervisor und das Hauptbetriebssystem zu kompromittieren. Wird von größeren Unternehmen mit einigen tausend virtuellen Hosts ausgegangen, viele VPS-Anbieter fallen z. B. in diese Kategorie, hätte das weitreichende Folgen.

Von besonderer Bedeutung ist dabei die Arbeit von J. Rutkowska. Das von ihr entwickelte Angriffsszenario für hardwarevirtualisierte Systeme wurde *Blue Pill* [17] genannt, eine Referenz an den Film *The Matrix*. Dieses Konzept für Malware wurde für die AMD-V-Technologie entwickelt, später erfolgte die Portierung auf Intels VT-x.

In seinem, das erste Mal 2007, im Rahmen der *DeepSec-Konferenz* vorge-tragenem Ansatz, virtuelle Maschinen zu manipulieren, zeigt N. Quynh anhand von Xen, eine weitere Möglichkeit, virtuelle Maschinen zu manipu-lieren auf und betont, dass die vorgestellte Vorgehensweise keineswegs nur auf den VMM von Xen angewandt werden kann. Dabei wird unter anderem demonstriert, wie es möglich ist, mit einem Programm in der *Dom0* SSH-Zugangsdaten einer Linux-VM in der Userdomäne *DomU* aufzuzeichnen. Dafür ist es notwendig, Kontrolle über die *Dom0* zu erlangen, ist dies aber gegeben, können Daten ohne dass es auffällt „on the fly“ mitprotokolliert werden¹.

Einen Ausweg daraus kann die vollständige Verschlüsselung bieten, sodass im Idealfall selbst der Anbieter nicht auf die Daten des Kunden zugreifen kann [18, siehe S. 69ff.].

Hardwareausfälle

Neben mutwilligen Systemangriffen müssen zwei weitere Gefahren in Betracht gezogen werden: Zum einen kann ein Programmabsturz im Hypervisor oder Hostbetriebssystem dazu führen, dass alle virtuellen Hosts mit zum Absturz gebracht werden. Zum anderen können auch bei Hardwareausfällen mehrere bis alle virtuellen Hosts betroffen sein.

Kontrolle

Zwar bedingt weder Virtualisierung das Cloud-Computing (siehe Kapitel 4), noch ist das umgekehrt der Fall. Trotzdem werden sie meist gemeinsam eingesetzt, denn für kleinere bis mittlere Projekte rechnet sich Virtualisierung vor allem dann, wenn sie ausgelagert wird und von darauf spezialisierten Anbietern übernommen wird. Neben vielen Vorteilen wird damit aber auch das Maß der Kontrolle reduziert. Selbst bei großen, etablierten Anbieter wie Amazon mit *Elastic Compute Cloud*

¹<http://vimeo.com/24229185>

(EC2) (siehe Abschnitt 4.2.1) können Ausfälle mit Datenverlust nicht ausgeschlossen werden. Ein solcher Ausfall fand z. B. am 21. April 2011 statt und führte zur Nichterreichbarkeit vielen Internetdienste, darunter *Foursquare*, *Springpad*, *Quoara* und *Reddit*².

Neben technischen Gründen kann der Anbieter sich auch dazu entschließen aus politischen Gründen Services von Kunden von einem Tag auf den anderen zu deaktivieren. Als Beispiel hierfür sei ebenfalls Amazons EC2 genannt, von deren Benutzung *WikiLeaks* im Dezember 2010 ausgeschlossen wurde [19].

Kostenüberblick

Viele auf Virtualisierung basierende Dienste, insbesondere die in Kapitel 4 vorgestellten Cloud-Computing-Services werden nach Rechenzeit abgerechnet. Das bedeutet aber auch, dass die anfallenden Kosten schwerer zu berechnen sind. Außerdem ist diese leistungsbezogene Verrechnung nicht für jeden Anwendungsfall günstiger, wie es die Autoren von [20] bestätigen.

Migration

Ebenfalls hauptsächlich für Clouddienste gilt, dass die Migration und Integration bestehender Systeme aufwendig sein kann (siehe auch [21]).

3.8 Virtual Private Server (VPS)

VPS sind virtualisierte Instanzen von Servern. Für VPS kommt meist Betriebssystemvirtualisierung zum Einsatz. Weit verbreitet ist dabei Parallels Virtuozzo, eine proprietäre, auf OpenVZ basierende Virtualisierungslösung (siehe Abschnitt 3.5.1). Die Verwaltung der VPS-Instanzen erfolgt meist über ein Webinterface.

Ein VPS stellt eine günstige Alternative zu einem Rootserver dar, bei dem ein Server einen physischen Rechner beansprucht. Durch das Zusammenlegen vieler Instanzen auf ein und der selben Hardware kann der Anbieter einen VPS deutlich günstiger zur Verfügung stellen. Dem Kunden wird dabei ein Maß an Ressourcen zugesichert, das garantiert nicht unterschritten wird. Da meist nicht alle auf einem physischen System zusammengefassten Kundeninstanzen unter Volllast laufen, kann der Anbieter einzelnen Kunden dynamisch mehr Ressourcen zuweisen.

Abgesehen davon, dass die in Abschnitt 3.6 und 3.7 aufgelisteten Vor- und Nachteile zum Tragen kommen gibt es kaum Unterschiede zu Rootservern. So können beliebige Dienste wie SSH, FTP und Webserver vom Benutzer aktiviert

²Eintrag für 21. April 2011 auf <http://status.aws.amazon.com/>

und eingerichtet werden. Bei hardwarenahen Anpassungen kann es jedoch zu Einschränkungen durch die Virtualisierung kommen. Ebenso problematisch ist der Einsatz beliebiger Kernel, da Virtualisierungstechnologien wie Xen bestimmte Kernelversionen voraussetzen.

Kapitel 4

Cloud-Computing

Der Begriff Cloud-Computing ist zur Zeit allgegenwärtig und aus dem IT-Umfeld nicht mehr wegzudenken. Das liegt vor allem an der inflationären Verwendung des Begriffs, sodass es sich kein Internetservice erlauben kann, nicht darauf hinzuweisen, auch an die berühmte „Wolke“ angebunden zu sein. Von der Wirksamkeit des Marketings abgesehen lässt sich aber nur schwer klären was eigentlich hinter dem Begriff steckt und was Clouddienste gegen schon viel länger existierende Online-Services abgrenzt. Mittlerweile wird „in der Cloud“ gerne synonym für „online“ verwendet.

Da viele der später vorgestellten Lösungen aber tatsächlich als Clouddienste bezeichnet werden, muss die Bedeutung klar sein. Die vorliegende Arbeit orientiert sich an folgender, aus [21, S. 44] stammenden Definition von „Cloud-Services“:

Cloud-Services bezeichnen den bedarfsgerechten Einsatz eines Dienstes (zum Beispiel Infrastruktur, Plattform oder Anwendung), der sich schnell an neue Bedingungen anpassen lässt und der basierend auf dem tatsächlichen Gebrauch der bereitgestellten Ressourcen abgerechnet wird.

Eine sinnvolle Ergänzung dazu liefern die in [22] angeführten grundlegenden Eigenschaften:

- Der Benutzer kann, ausreichende Rechte vorausgesetzt, Ressourcen verwalten ohne dass dafür die Interaktion mit einer Person auf Seiten des Dienstleisters notwendig ist.
- Die Ressourcen werden über das Netzwerk und mittels standardisierter Mechanismen für unterschiedliche Endgeräte angeboten.
- Der Anbieter bündelt seine Dienste, die auf unterschiedlicher physikalischer Hardware basieren können, dynamisch. Die Services sind zudem nicht ortsgebunden, der Kunde weiß normalerweise nicht, wo sich die

Hardware des Anbieters exakt befindet, ihm ist meist nur die entsprechende Region oder das entsprechende Land bekannt.

- Ressourcen können rasch den Anforderungen angepasst werden, unter Umständen erfolgt dies sogar automatisch.
- Cloud-Systeme protokollieren den Einsatz der Ressourcen und stellen ihn in übersichtlicher Form zur Auswertung und Kontrolle bereit. Dadurch wird Transparenz für den Dienste-Anbieter sowie den Benutzer gewährleistet.

Auch in [23] wird beschrieben, dass es beim Cloud-Computing darum geht, nach Bedarf und dynamisch Serverleistungen zur Verfügung zu stellen und zu konfigurieren. Die Autoren untersuchen in [23] vor allem die Zusammenhänge zwischen Cloud-Computing und Virtualisierung. Von Bedeutung ist ihr Schluss, dass Virtualisierung bei Cloud-Computing-Architekturen die Schlüsselrolle spielt.

Auf dem Gebiet des Cloud-Computing findet zur Zeit viel Forschung statt. Umsetzungen von Konzepten, wie die in [24] vorgestellten *Lazy Update Propagation* werden es in Zukunft noch attraktiver machen. Dabei steht die Minimierung der durch die Datenreplikation in der Cloud auftretenden Schwierigkeiten beim Abgleich mehrerer Instanzen im Vordergrund. Um den Überblick zu bewahren beschränken sich die Betrachtungen in dieser Arbeit auf die eigentlichen Clouddienste, auch *öffentliche Cloud* genannt und geht nicht näher auf Konzepte der *gemeinschaftlich genutzten*, *privaten* und *hybriden* Clouds ein [25].

4.1 IAAS, PAAS und SAAS

Im Zusammenhang mit Cloud-Computing haben sich drei Begriffe etabliert: *Infrastructure as a Service* (IAAS), *Platform as a Service* (PAAS) und *Software as a Service* (SAAS). Einfach unterscheiden lassen sich diese Kategorien, wenn die Abhängigkeit gegenüber dem Anbieter herangezogen wird. Diese ist am geringsten bei IAAS, nimmt bei PAAS zu und ist bei SAAS am größten. Die detaillierten Eigenschaften werden im Folgenden erläutert. Zudem veranschaulicht Abbildung 4.1, nach [26], die Abgrenzung der Begriffe zueinander.

4.1.1 Infrastructure as a Service (IaaS)

Unter dem Begriff IAAS wird die Bereitstellung von Computerinfrastruktur zusammengefasst.

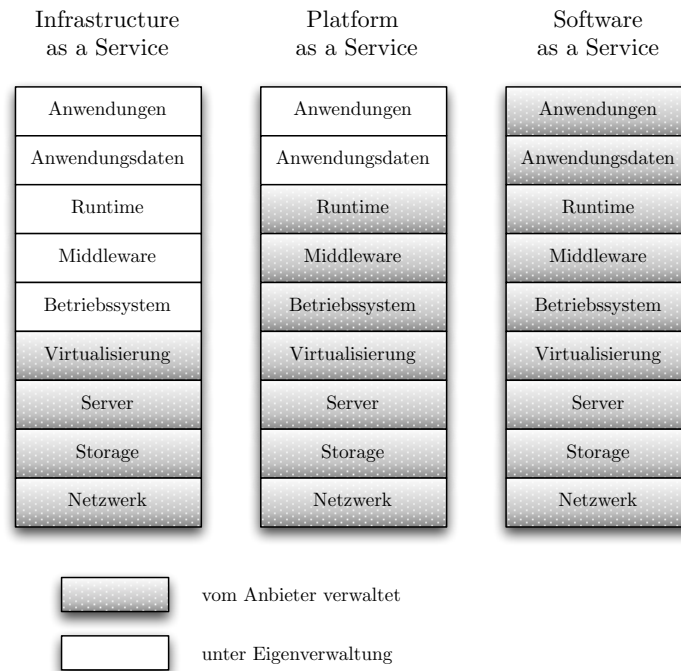


Abbildung 4.1: Abgrenzung IAAS, PAAS und SAAS.

Während auch die unter Abschnitt 3.8 VPS mit IAAS-Diensten vieles gemeinsam haben, werden sie durch folgende Merkmale abgegrenzt: IAAS-Dienste skalieren einfacher, da sämtliche Ressourcen verteilt verwaltet werden. Erweiterung und Ausbau bestehender Infrastruktur lassen sich bei IAAS einfacher realisieren. Auch moderne VPS lassen sich anpassen, das ist aber meist mit einer Neukonfiguration und Ausfallszeiten verbunden. Entscheidend ist, dass dem Kunden die Kosten bei den meisten IAAS-Anbietern nicht pauschal, sondern nach tatsächlich in Anspruch genommenen Ressourcen verrechnet werden. Dies macht diese Art von Services auch für kleinere Projekte interessant. Wenn diese wenig Speicherplatz beanspruchen, kaum Rechenzeit benötigen und wenig Traffic generieren, fallen entsprechend niedrigere Kosten an.

4.1.2 Platform as a Service (Paas)

Bei PAAS steht im Gegensatz zu IAAS-Diensten nur noch die Anwendung sowie die Datenverwaltung unter der Kontrolle des Benutzers (siehe Abbildung 4.1). PAAS stellt also einen Anwendungsdienst oder ein Framework zur Verfügung mit dessen Hilfe Anwendungen entwickelt, verteilt und betrieben werden

können. Zur einfachen Verwaltung wird dabei vom Anbieter die Möglichkeit geboten, den PaaS-Service über eine Weboberfläche zu bedienen.

4.1.3 Software as a Service (SaaS)

SAAS-Dienste weisen die wenigsten Freiheitsgrade für den Benutzer auf. Diesem wird lediglich das vereinbarte Programm zur Ausführung überlassen. Sämtliche Konfigurationen, Virtualisierung- und Deploymentaufgaben werden vom Anbieter übernommen. SAAS bezeichnet demnach nicht das Betreiben beliebiger Programme in der Cloud, sondern einen Umstieg auf die bestehende Onlinelösung des Anbieters. Ein Beispiel für einen SAAS-Dienste ist *Gmail* von Google, das im deutschsprachigen Raum aus lizenzrechtlichen Gründen als *Google Mail* angeboten wird. Solange die Nutzungsbedingungen eingehalten werden, können Benutzer alle Funktionen der Software verwenden. Alles was über die reine Verwendung der Software hinausgeht, kann nicht von Kunden modifiziert werden. Eine Ausnahme stellt in diesem konkreten Fall die Verwendung von Googles Maildienst unter eigenem Domainnamen dar. Dabei müssen zumindest die DNS-Einträge durch den Benutzer angepasst werden.

4.2 Amazon Web Services (AWS)

Amazon, das Unternehmen, welches als Onlineversandhaus für Bücher Bekanntheit erlangte, bietet seit 2002 auch Dienste rund um das Thema Remote-Computing an. Amazon nimmt hier einen besonderen Stellenwert ein, da viele der in Kapitel 5 vorgestellten Deploymentlösungen auf *Amazon Web Services* (AWS) basieren oder damit kompatibel sind.

4.2.1 Amazon Elastic Compute Cloud (Amazon EC2)

Ein wesentlicher Bestandteil der AWS ist EC2. Diese erlaubt es Kunden, virtualisierte Betriebssysteme zu mieten und darauf eigene Anwendungen auszuführen. Als Virtualisierungstechnologie kommt Xen (siehe Abschnitt 3.4.1) zum Einsatz. Der Zustand einer Instanz lässt sich als sogenanntes *Amazon Machine Image* (AMI) persistieren und als Ausgangspunkt für neue Instanzen nutzen.

Amazon EC2 ist als Einzelprodukt zum Deployment von LAMP-Anwendungen geeignet. Zudem basieren virtuelle Instanzen vieler weiterer Anbieter auf Amazons Cloud-Infrastruktur und versuchen, die Verwaltung der virtualisierten Instanzen weiter zu vereinfachen.

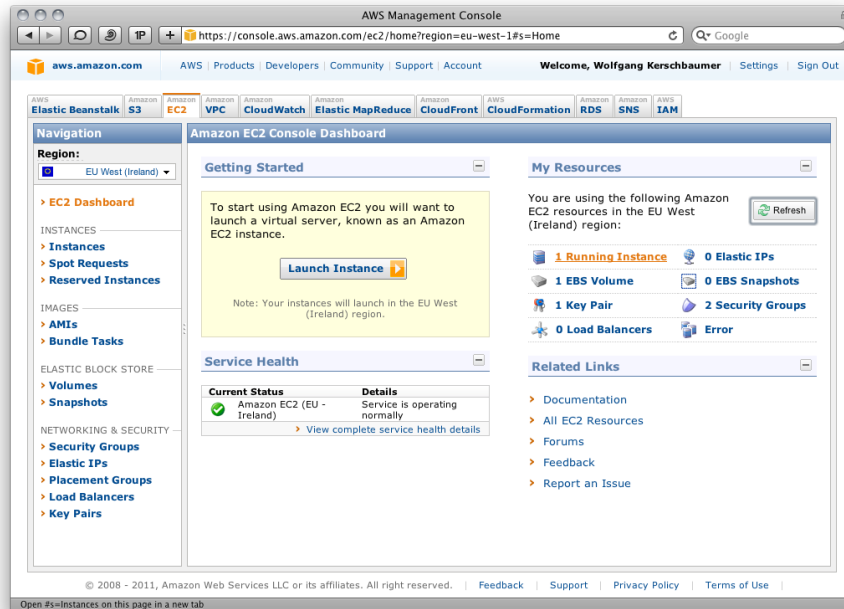


Abbildung 4.2: Die Amazon EC2-Managementkonsole.

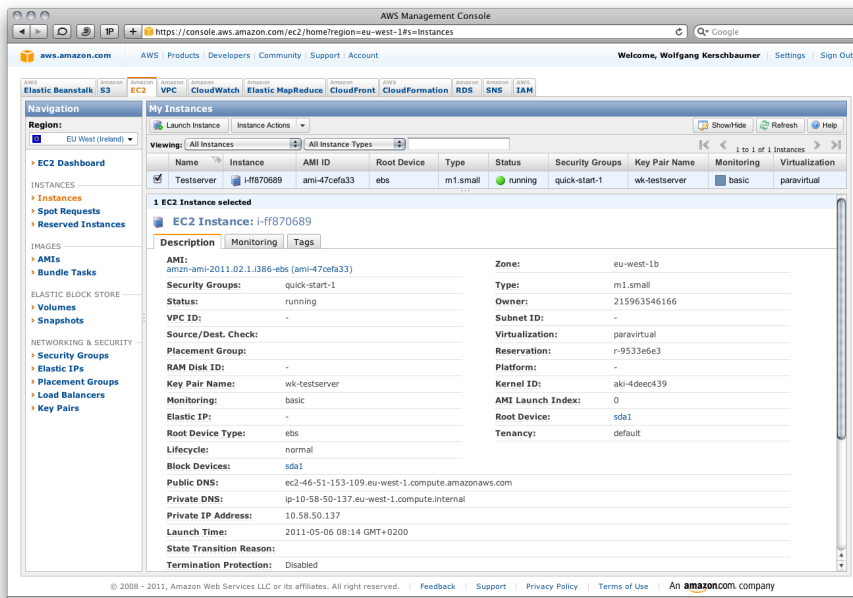


Abbildung 4.3: Eigenschaften einer Amazon Ec2-Instanz.

Kapitel 5

Deploymentlösungen

In diesem Kapitel werden zunächst Modelle zum Deploymentvorgang sowie Möglichkeiten zu dessen Bewertung vorgestellt. Danach werden Implementierungen von Deploymentlösungen beschrieben. Dabei ist es weder sinnvoll noch möglich, sämtliche Implementierungen und Lösungen vorzustellen, vielmehr werden aus jeder Kategorie an Werkzeugen einige wenige vorgestellt. Die Auswahl der Lösungen richtete sich dabei in erster Linie danach, wie einfach das Deployment mit der jeweiligen Lösung umzusetzen ist.

Zuerst werden exemplarisch Konfigurations- und Deploymentprogramme vorgestellt, die keine virtualisierte Umgebung erfordern, den Umgang mit einer solchen aber mehr oder weniger stark vereinfachen.

Danach werden Deploymentlösungen in Form von Diensten vorgestellt, welche allesamt auf Virtualisierung basieren.

5.1 Modelle

Ein automatisierter Deploymentprozess wird von G. Kecskemeti in [27] detailliert beschrieben. Der empfohlene Deploymentprozess basiert dort auf Xen (siehe Abschnitt 3.4.1) und umfasst zwei Phasen, die in Abbildung 5.1 bildlich dargestellt werden.

- *Phase 1:*

Erzeugen des Virtual Machine Images:

1. Eine Deploymentanfrage wird an den *Automated Virtual Appliance Creation Service* gestellt. Es soll ein VM-Image der *Xen-Domäne M* von *Grid A* in einer *Application Archive* gespeichert werden.
2. Die *Base Virtual Appliance* wird erzeugt und optimiert.
3. Der *Automated Virtual Appliance Creation Service* speichert die *Base Virtual Appliance* inklusive Deltapaketen in einer Application-

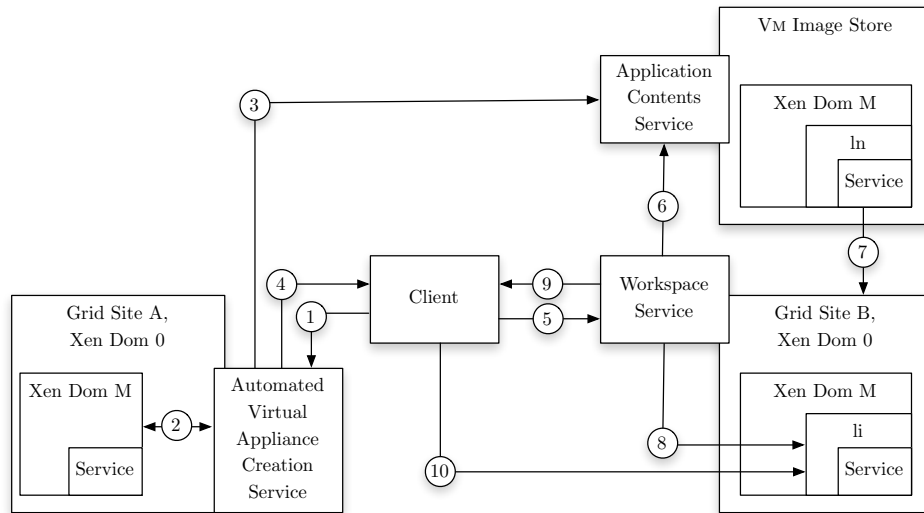


Abbildung 5.1: Automatic Service Deployment nach [27].

Archive-Instanz und fügt die Konfigurationen und alle Abhängigkeiten, die notwendig sind, um die Deploymentanfrage zu erfüllen, hinzu.

4. Dann bestätigt der *Automated Virtual Appliance Creation Service* die Virtual Appliance, indem er dem Client deren Endpunktreferenz vom *Application Contents Service* schickt.

- *Phase 2:*

Service Deployment von einem Verzeichnis aus:

5. Der Benutzer beauftragt den *Workspace-Service*, die *Virtual Machine* unter Verwendung der Endpunktreferenz des *Application Contents Services* auszurollen.
6. Die *Workspace-Service-Factory* erfragt die *Application-Archive-Instanz* und *Workspace-Service-Konfiguration* beim *Application Contents Service*.
7. Das Image wird auf Grid B übertragen.
8. Der Ort für das Virtual-Machine-Image von der *Application-Archive-Instanz* wird vorbereitet.
9. Die *Workspace-Service-Factory* gibt dem Benutzer die Endpunktreferenz des *Workspace-Services* zurück.

Ein anderer Ansatz wird in [28] aufgezeigt. Dabei werden sogenannte *Basis-Virtual-Disks* verwendet, welche die grundlegenden Betriebssystemkomponenten sowie gemeinsam genutzte Bibliotheken enthalten. Von einer

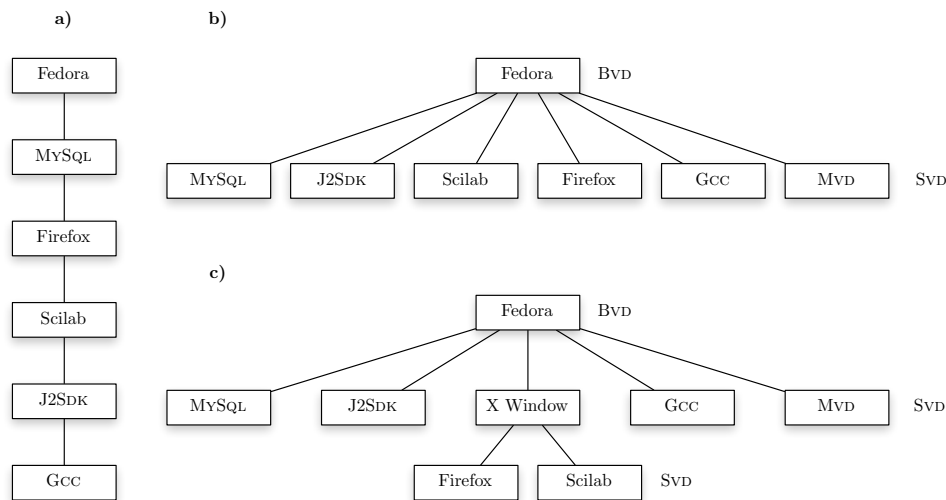


Abbildung 5.2: Virtual-Disk-Image Hierarchien nach [28].

solchen *Basis-Virtual-Disk* wird ein Kind-Image, auch *Software-Virtual-Disk* genannt, für weitere Komponenten wie MySQL, erzeugt. Der Aufbau einer solchen Deploymentlösung ist in Abbildung 5.2 dargestellt, wobei (a) die ursprüngliche Struktur zeigt und (b) die Aufteilung in eine Basis-Virtual-Disk und eine *Software-Virtual-Disk*. Zu beachten ist, dass es, wie in (c) gezeigt, zu Abhängigkeiten unter den *Software-Virtual-Disks* kommen kann, wie das im gezeigten Beispiel bei dem Webbrowser *Firefox* und der numerischen Kalkulationssoftware *Scilab* der Fall ist, welche eine grafische Oberfläche voraussetzen und somit beide vom X-Window-System abhängig sind.

Gegenstand der Untersuchung in [29] war eine Architektur für das Clouddeployment. Der Ablauf ist auf Amazons EC2-Dienst, sowie die Xen-Umgebung der Autoren abgestimmt, und beansprucht für sich, die folgenden Probleme zu adressieren:

- Eigenschaften wie Verfügbarkeit, Sicherheit, Leistung und Kosten wurden in das Design der Lösung miteinbezogen.
- Konfiguration des Softwarestack über unterschiedliche virtuelle Maschinen hinweg.
- Konfiguration des Cloudumfelds, sodass das Verhältnis zwischen den Anforderungen des Dienstes und den Ressourcen der Cloud feststeht.
- Die Deployment-Automatisierung erfordert eine komplexe Abfolge an Konfigurationsaufrufen und unterscheidet sich je nach Dienst.

In Abbildung 5.3 ist der Deploymentprozess für die Cloud grafisch dargestellt.

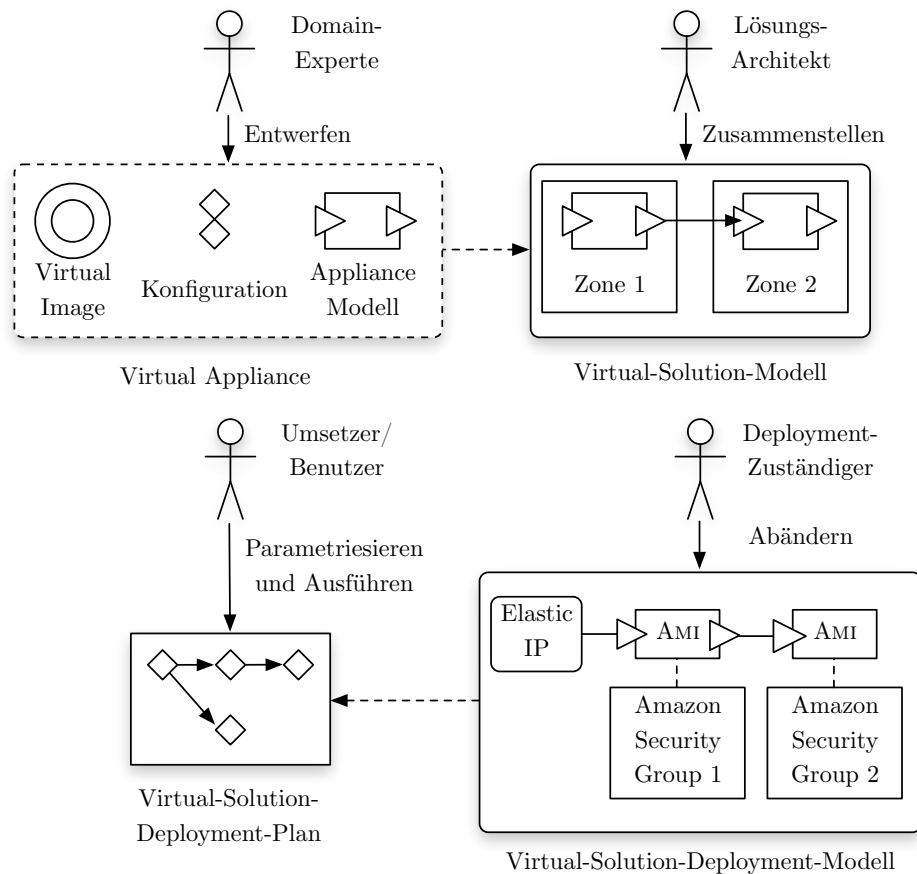


Abbildung 5.3: Virtual-Solution-Design und -Deployment nach [29].

5.1.1 Ausfallssicherheit

Der Erfolg des Deployments virtueller Appliances lässt sich, wie in [30] gezeigt, anhand von Wahrscheinlichkeiten ausdrücken. Dabei wird von einem Aufbau wie in Abbildung 5.4 ausgegangen.

Die Anzahl der Vorgänge, die notwendig sind um ein Service auszurollen, wird mit m bezeichnet. Jeder Vorgang i weist eine Fehlerwahrscheinlichkeit von a_i ($i = 1, \dots, m$) auf. Die Wahrscheinlichkeit R_1 , einen Dienst erfolgreich einzusetzen, errechnet sich aus

$$R_1 = \prod_{i=1}^m \prod_{i=1}^m (1 - a_i). \quad (5.1)$$

Es ist davon auszugehen, dass jeder Vorgang i eine bestimmte Anzahl an Konfigurationsparametern m_i erfordert. Dabei ist die Fehlerwahrscheinlichkeit

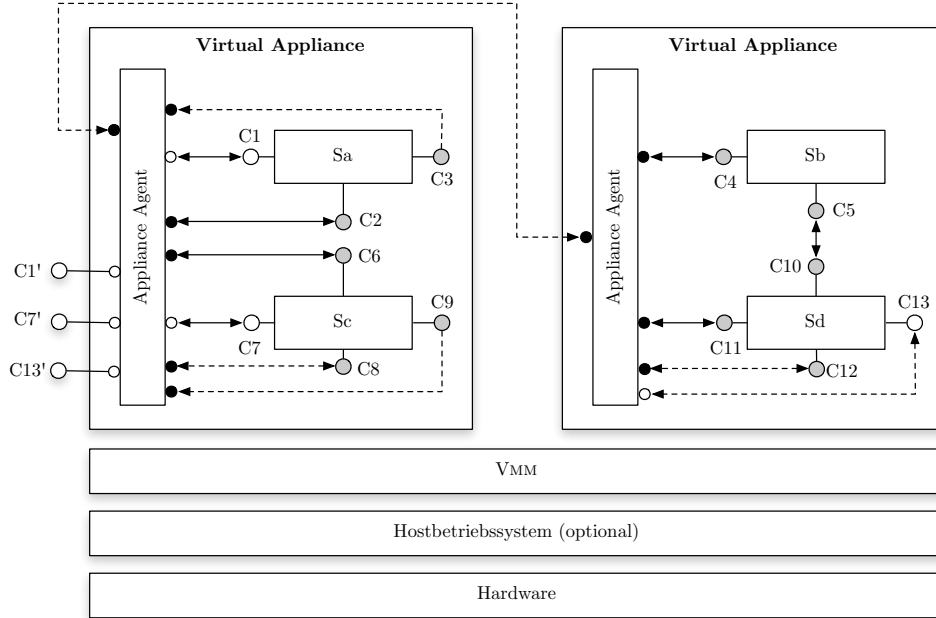


Abbildung 5.4: Aufbau der virtuellen Appliance nach [30].

des Konfigurationsparameters j mit p_j festgelegt, sodass sich die Fehlerwahrscheinlichkeit a_i aus

$$a_i = 1 - \prod_{j=1}^{m_i} (1 - p_j) \quad (5.2)$$

zusammensetzt. Aus Gleichung 5.1 und 5.2 lässt sich die Gesamtwahrscheinlichkeit für ein erfolgreiches Deployment R_1 durch

$$R_1 = \prod_{i=1}^m \prod_{j=1}^{m_i} (1 - p_j) \quad (5.3)$$

berechnen. Ein anderer Ansatz geht von n notwendigen Vorgängen aus und einer Fehlerwahrscheinlichkeit eines Vorgangs i von b_i ($i = 1, \dots, n$). Ähnlich wie zuvor ist davon auszugehen, dass Vorgang i die Anzahl von n_i Konfigurationsparametern benötigt und der Konfigurationsparameter j eine Fehlerwahrscheinlichkeit von q_j aufweist. In diesem Fall ergibt sich die Gesamtwahrscheinlichkeit für ein erfolgreiches Deployment R_2 durch

$$R_2 = \prod_{i=1}^n (1 - b_i) = \prod_{i=1}^n \prod_{j=1}^{n_i} (1 - q_j). \quad (5.4)$$

In [30] werden die beiden Ansätze verglichen, indem R_1 und R_2 gegenübergestellt werden. Die Schlussfolgerung lautet, dass mit der Reduktion der Deploymentschritte und deren Fehlerwahrscheinlichkeiten die Erfolgsrate beim Deployment steigt. Nachdem jeder Schritt von der Konfiguration seiner Parameter abhängt, steigt die Wahrscheinlichkeit für ein erfolgreiches Deployment auch an, wenn die Anzahl der Parameter, und somit die Komplexität des Deployments, reduziert werden kann.

5.2 Konfigurationswerkzeuge mit Unterstützung für Virtualisierung

Die vorgestellten Programme stehen stellvertretend für die Kategorie der Konfigurationswerkzeuge. Übersichten der unterschiedlichen Werkzeuge lassen sich im Internet aufrufen ¹. Jedes dieser Werkzeuge wurde für bestimmte Aufgaben entwickelt, in ihrer Auswirkung beim Deployment von LAMPP-Anwendungen in virtualisierten Umgebungen unterscheiden sie sich jedoch nicht grundlegend.

5.2.1 WD2-Tools

Der Name steht für *Web Development und Deployment Tools*. Bei WD2-Tools handelt es sich um das Konzept des Autors, Deploymentschritte für den LAMPP-Stack (siehe Abschnitt 2.1.2) unter Linux zu automatisieren. Es ist ein Kommandozeilenprogramm, dessen Parameter bestimmen, welche Funktion ausgeführt wird. Der Aufbau von WD2-Tools ist in Abbildung 5.5 dargestellt.

Als Programmiersprache wurde Python gewählt. Python weist alle Eigenschaften einer modernen Programmiersprache auf und erfreute sich in den letzten Jahren einer wachsenden Beliebtheit. Es ist universell einsetzbar, sodass es sowohl von Systemadministratoren, als auch Webentwicklern verwendet wird. Da viele grundlegende Programme in Python geschrieben werden, ist es fester Bestandteil vieler Linuxdistributionen. Auch finden sich in aktuellen Linuxbetriebssystemen unzählige Programme für Endanwender, die auf einem der Python-Frameworks oder Toolkits für ein *Graphical User Interface* (GUI) basieren. Deshalb ist Python unter aktuellen Linuxsystemen bereits vorinstalliert. Diese weite Verbreitung war ein wichtiger Grund für die Verwendung von Python.

¹http://en.wikipedia.org/wiki/Comparison_of_open_source_configuration_management_software

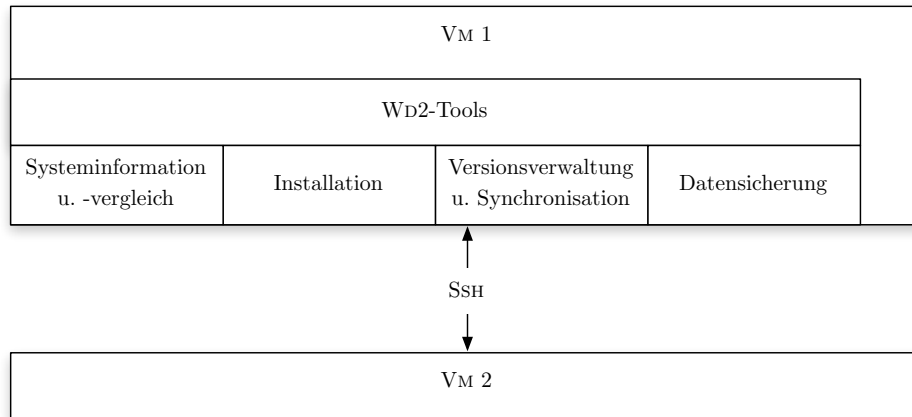


Abbildung 5.5: Aufbau der WD2-Tools.

Aufgerufen wird WD2-Tools über `wd2tools.py`. Die verwendeten Parameter bestimmen, welche Funktion ausgeführt wird. In der Datei `config.ini` werden projektspezifische Daten festgelegt. Diese Datei kann entweder manuell oder mit Hilfe eines `wd2tools.py --config`-Aufrufes erstellt werden. Dabei werden die notwendigen Informationen abgefragt und in der Konfigurationsdatei abgelegt.

Für den Betrieb der WD2-Tools gibt es folgende Voraussetzungen:

- Eine auf Debian basierende Distribution mit Kernel 2.6.x aufwärts.
- Python Version 2.6 aufwärts.
- Python SSH-Bibliothek *paramiko*.
- Rootrechte auf Host `local` sowie `remote`.
- SSH-Zugang auf Host `remote`.
- Kommandozeilenprogramm `sudo`.

Das Programm wurde unter *Ubuntu*, Version 10.10 Desktop Edition entwickelt und dafür sowie *Debian* in den Versionen 5 und 6 optimiert. Zur Verwendung unter anderen Linuxdistributionen können Anpassungen erforderlich sein.

Das Programm muss nicht installiert werden, es muss nur darauf geachtet werden, dass es ausführbar ist (`chmod +x wd2tools.py`).

Falls nicht anders konfiguriert, muss die zu installierende Webanwendung dabei im Ordner `wd2tools-webapplications`, auf selber Ebene wie das Programm selbst, liegen.

Zur Kommunikation zwischen zwei Rechnern wird ausschließlich SSH verwendet. Für notwendiges Hintergrundwissen zum Arbeiten mit SSH wurde [31] herangezogen. Auch im Umgang mit SSH gilt, dass ein Kompromiss

zwischen Benutzerfreundlichkeit und Sicherheit zu schließen ist. Empfohlen wird, mit öffentlichen Schlüsseln zu arbeiten. Diese sollten aber unbedingt mit einem Passwort versehen sein. Um den Umgang mit Schlüsselpasswörtern zu erleichtern sollte das Open-SSH-Hilfsprogramm `ssh-agent` benutzt werden.

Um die Verbindung aufzubauen gibt es zwei Möglichkeiten:

- SSH Public-Key-Authentifizierung (*empfohlen*)

Die Authentifizierung wird über die Public-Key Kryptographie von SSH durchgeführt. Für die Automatisierung ist es von Vorteil wenn nicht zusätzlich noch ein Passwort eingegeben werden muss. Das Passwort für den Schlüssel kann daher leer bleiben (*nicht empfohlen*) oder `ssh-agent` wird zur Verwaltung des Passworts verwendet.

- Authentifizierung über Passwordeingabe

Dabei wird das Passwort beim ersten SSH-Aufbau abgefragt und temporär gespeichert.

Für einige WD2-Tools-Befehle sind Rootrechte erforderlich. Da das Programm aber automatisiert ausgeführt werden soll, wird am besten ein eigener Benutzer eingerichtet, der nur dieses Programm ausführen darf und für die Verwendung von WD2-Tools keine Passwordeingabe benötigt. Dazu wird mit `visudo` am Ende der `sudo`-Konfigurationsdatei (meist `/etc/sudoers`) folgendes hinzugefügt:

```
wd2tools-user host = NOPASSWD:/home/wd2tools-user/bin/wd2tools
```

Wird WD2-Tools mit den dafür vorbereiteten Archiven verwendet, so sind in den enthaltenen Webanwendungen die nachfolgenden Daten für den Administrationsaccount vorgegeben. Aus Sicherheitsgründen müssen diese Anmeldedaten nach der Installation geändert werden.

- Username: `wd2tools`
- Password: `wd2tools`

5.2.2 Funktionen

WD2-Tools nimmt Befehle über die Kommandozeile und einer Konfigurationsdatei `config.ini` entgegen. Indem das Argument `--config` verwendet wird, kann die Konfigurationsdatei erstellt oder überschrieben werden.

Mit WD2-Tools können Webanwendungen in Form eines speziell dafür aufbereiteten `tar.bz2`-Archivs installiert werden. Um wichtige Informationen über das System zu erhalten, wird `wd2tools` mit `--sysinfo` als Argument ausgeführt. Ein Vergleich dieser Systeminformationen zwischen dem lokalen und dem Remotehost wird beim Aufruf mit `--compare` angestellt.

Nach der Installation ist es möglich, diese auf den unter der Datei `config.ini` eingetragenen Remotehost zu übertragen. Dies, sowie die spätere Übernahme von Änderungen geschieht mittels `--sync`. Um Änderungen unter der lokalen Installation zu “speichern” und später mit `--sync` auf den Remotehost übertragen zu können, ist der Befehl `--commit` notwendig.

Um sämtliche Daten der Webanwendung, verpackt in einem `tar.bz2`-Archiv, zu exportieren, wird `--backup` verwendet.

Abgesehen von `--host` und `--verbose` sind die Argumente *mutually exclusive*, d. h., es kann jeweils nur ein Hauptargument angegeben werden. Kurzinformationen zu den Optionen werden, wie in den meisten UNIX- und Linux-Programmen üblich, entweder mit `-h` oder der Langversion `--help` aufgerufen.

Im Folgenden werden die einzelnen Argumente im Detail erläutert:

- Webanwendung installieren (`--install`)

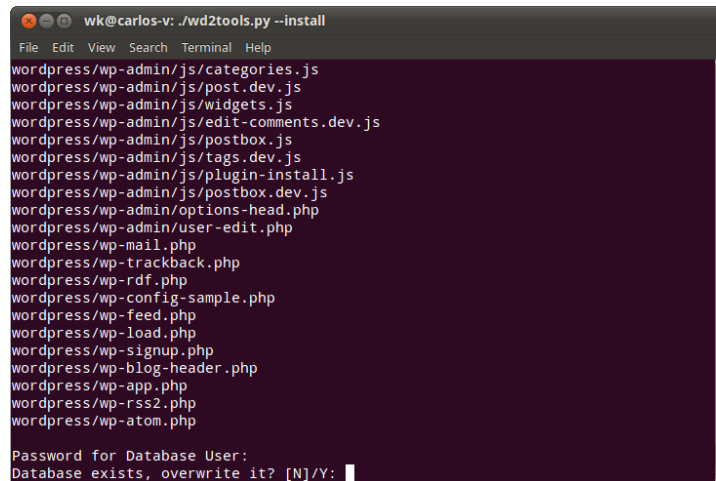
Mit dieser Option lässt sich ein als `tar.bz2`-Datei bereitgestelltes Paket installieren. Dabei muss die Archivdatei sämtliche für die Installation notwendige Dateien enthalten. Dazu gehört, neben den Dateien der Webanwendung, auch eine Datenbankdatei. Dieses initiale Paket muss für die erste Verwendung manuell eingerichtet werden. Zu der üblichen Ordnerstruktur kommen dabei die folgenden Verzeichnisse hinzu:

- `./wd2tools-backup`
- `./wd2tools-data/db`

Wenn die Option ausgeführt wird, werden alle Verzeichnisse und Dateien entsprechend der Konfiguration eingerichtet. Auch die Datenbank wird gemäß der Konfiguration eingerichtet und Datenbankname und Name des Datenbankusers bei Bedarf erstellt. Gibt es die Datenbank schon, wird darauf hingewiesen und man hat die Wahl den Vorgang abzurechnen oder die bestehende Datenbank zu überschreiben. Die zur Installation herangezogene Datenbankdatei muss im Verzeichnis `./wd2tools-data/db` liegen. Der Dateiname ist nicht relevant, da stets die neueste Datei gewählt wird. Abbildung 5.6 zeigt die Installation einer Webanwendung.

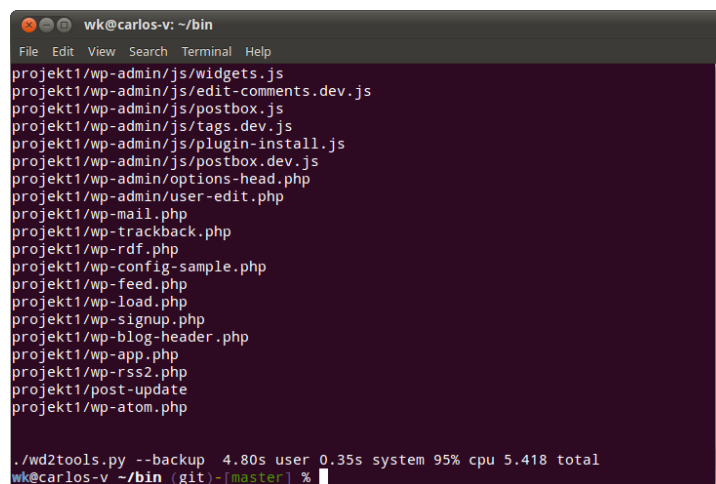
- Backup erstellen (`--backup`)

Mit dem Argument `--backup` erzeugt WD2-Tools eine Archivdatei mit sämtlichen Dateien und einer Sicherung der Datenbank, wie es in Abbildung 5.7 dargestellt ist. Diese wird nach Erstellen in den Ordner `./wd2tools-backup` verschoben. Nicht mitgesichert werden der Git-Hilfsordner (`.git`) sowie der Backupordner selbst (`wd2tools-backup`).



```
wk@carlos-v: ~/wd2tools.py --install
File Edit View Search Terminal Help
wordpress/wp-admin/js/categories.js
wordpress/wp-admin/js/post.dev.js
wordpress/wp-admin/js/widgets.js
wordpress/wp-admin/js/edit-comments.dev.js
wordpress/wp-admin/js/postbox.js
wordpress/wp-admin/js/tags.dev.js
wordpress/wp-admin/js/plugin-install.js
wordpress/wp-admin/js/postbox.dev.js
wordpress/wp-admin/options-head.php
wordpress/wp-admin/user-edit.php
wordpress/wp-mail.php
wordpress/wp-trackback.php
wordpress/wp-rdf.php
wordpress/wp-config-sample.php
wordpress/wp-feed.php
wordpress/wp-load.php
wordpress/wp-signup.php
wordpress/wp-blog-header.php
wordpress/wp-app.php
wordpress/wp-rss2.php
wordpress/wp-atom.php
Password for Database User:
Database exists, overwrite it? [N]/Y: █
```

Abbildung 5.6: Installation einer Webanwendung.

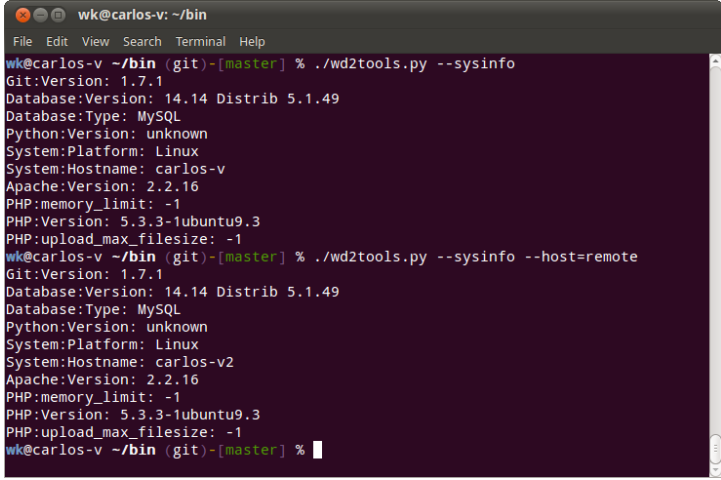


```
wk@carlos-v: ~/bin
File Edit View Search Terminal Help
projekt1/wp-admin/js/widgets.js
projekt1/wp-admin/js/edit-comments.dev.js
projekt1/wp-admin/js/postbox.js
projekt1/wp-admin/js/tags.dev.js
projekt1/wp-admin/js/plugin-install.js
projekt1/wp-admin/js/postbox.dev.js
projekt1/wp-admin/options-head.php
projekt1/wp-admin/user-edit.php
projekt1/wp-mail.php
projekt1/wp-trackback.php
projekt1/wp-rdf.php
projekt1/wp-config-sample.php
projekt1/wp-feed.php
projekt1/wp-load.php
projekt1/wp-signup.php
projekt1/wp-blog-header.php
projekt1/wp-app.php
projekt1/wp-rss2.php
projekt1/post-update
projekt1/wp-atom.php
./wd2tools.py --backup 4.80s user 0.35s system 95% cpu 5.418 total
wk@carlos-v ~/bin (git)-[master] % █
```

Abbildung 5.7: Backup einer Webanwendung.

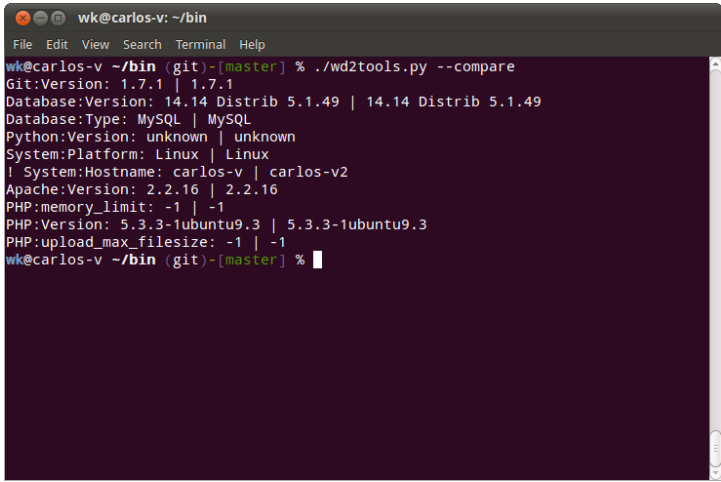
Die erstellte Backupdatei lässt sich mit `--install` auf ein neues System installieren.

- Systeminformation abfragen (`--sysinfo`)
Diese Option wird gewählt, um nähere Informationen über die Installationsumgebung zu erhalten. Informationen über Webserver-, PHP-, Python-Version werden dabei gesammelt und ausgegeben. Ohne zusätzliche Option `--host remote` oder `--host r` beziehen sich die Angaben immer auf den lokalen Rechner. Der Aufruf von `--sysinfo` wird in Abbildung 5.8 gezeigt.



```
wk@carlos-v: ~/bin
File Edit View Search Terminal Help
wk@carlos-v ~/bin (git)-[master] % ./wd2tools.py --sysinfo
Git:Version: 1.7.1
Database:Version: 14.14 Distrib 5.1.49
Database:Type: MySQL
Python:Version: unknown
System:Platform: Linux
System:Hostname: carlos-v
Apache:Version: 2.2.16
PHP:memory_limit: -1
PHP:Version: 5.3.3-1ubuntu9.3
PHP:upload_max_filesize: -1
wk@carlos-v ~/bin (git)-[master] % ./wd2tools.py --sysinfo --host=remote
Git:Version: 1.7.1
Database:Version: 14.14 Distrib 5.1.49
Database:Type: MySQL
Python:Version: unknown
System:Platform: Linux
System:Hostname: carlos-v2
Apache:Version: 2.2.16
PHP:memory_limit: -1
PHP:Version: 5.3.3-1ubuntu9.3
PHP:upload_max_filesize: -1
wk@carlos-v ~/bin (git)-[master] %
```

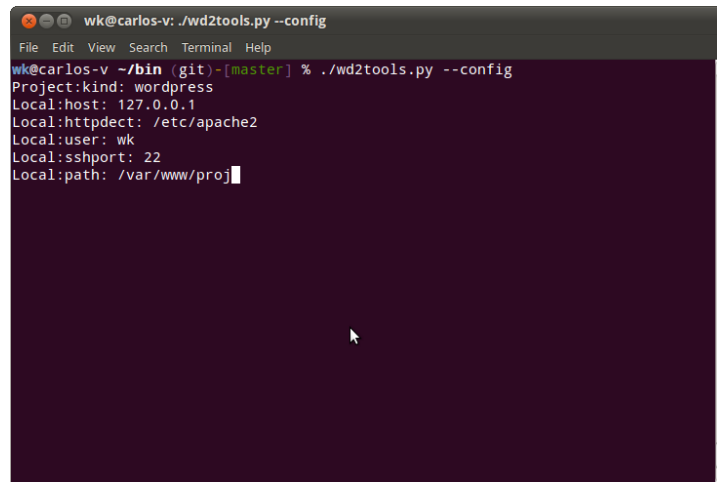
Abbildung 5.8: Sammeln von Systeminformationen.



```
wk@carlos-v: ~/bin
File Edit View Search Terminal Help
wk@carlos-v ~/bin (git)-[master] % ./wd2tools.py --compare
Git:Version: 1.7.1 | 1.7.1
Database:Version: 14.14 Distrib 5.1.49 | 14.14 Distrib 5.1.49
Database:Type: MySQL | MySQL
Python:Version: unknown | unknown
System:Platform: Linux | Linux
! System:Hostname: carlos-v | carlos-v2
Apache:Version: 2.2.16 | 2.2.16
PHP:memory_limit: -1 | -1
PHP:Version: 5.3.3-1ubuntu9.3 | 5.3.3-1ubuntu9.3
PHP:upload_max_filesize: -1 | -1
wk@carlos-v ~/bin (git)-[master] %
```

Abbildung 5.9: Vergleichen von Systeminformationen.

- Systeminformation vergleichen (`--compare`)
Bei `--compare` werden die gleichen Informationen gesammelt wie bei `--sysinfo`, nur mit dem Unterschied, dass sie für beide Hosts gegenübergestellt werden. Nicht übereinstimmenden Eigenschaften wird dabei ein `!` (Ausrufezeichen) vorangestellt, damit der Benutzer Unterschiede in der Konfiguration sofort erkennt (siehe Abbildung 5.9).
- Konfiguration erstellen (`--config`)
Gibt es noch keine Konfigurationsdatei `config.ini` oder soll diese überschrieben werden, kann dies manuell oder durch Verwendung der Option `--config` geschehen. Dabei werden der Reihe nach die Optionen



```
wk@carlos-v: ~/wd2tools.py --config
File Edit View Search Terminal Help
wk@carlos-v ~/bin (git)-[master] % ./wd2tools.py --config
Project:kind: wordpress
Local:host: 127.0.0.1
Local:httpdport: /etc/apache2
Local:user: wk
Local:sshport: 22
Local:path: /var/www/proj
```

Abbildung 5.10: Erzeugen der Konfigurationsdatei.

abgefragt und anschließend in die Datei `config.ini` geschrieben. Ein Teil der Abfrage ist in Abbildung 5.10 zu sehen.

- Änderungen übernehmen (`--commit`)
Um Änderungen komfortabel auf den Produktivserver übertragen zu können kommt *Git* zum Einsatz. Die Vorgehensweise wird dabei auf zwei Schritte reduziert:
 - `wd2tools.py --commit`
um Änderungen lokal zu bestätigen und zu übernehmen.
 - `wd2tools.py --sync`
um den aktuellen Stand des lokalen Repositories auf den Remotehost zu übertragen.

Dieser Vorgang ist in Abbildung 5.11 gezeigt. Weitere notwendige Schritte führt das Programm automatisch aus. Als Commitnachricht wird `Automated commit by wd2tools.` verwendet. Ist ein Zugriff auf ältere Versionen erforderlich, kann dieser wie gewohnt über das `git`-Kommandozeilenprogramm oder einer darauf basierenden grafischen Oberfläche, wie sie z. B. `gitk` bereitstellt, erfolgen.

- Änderungen synchronisieren (`--sync`)
Wie unter dem Punkt „Änderungen übernehmen (`-commit`)“ angeführt, ist die Option `--sync` der zweite Teil der Versionskontrollmechanismen in *WD2-Tools*. Dabei werden alle bereits mit `commit` übernommenen Änderungen auf den Remotehost übertragen. Das betrifft auch die Datenbank, die dann neu eingespielt wird. Existiert sie schon, gibt es einen Warnhinweis mit Möglichkeit zum Abbruch der Synchronisation


```
wk@carlos-v: ~/bin
File Edit View Search Terminal Help
You can suppress this message by setting them explicitly:
git config --global user.name "Your Name"
git config --global user.email you@example.com

If the identity used for this commit is wrong, you can fix it with:
git commit --amend --author='Your Name <you@example.com>'

12 files changed, 3132 insertions(+), 1 deletions(-)
create mode 100644 wd2tools-backup/backup-2011-02-09-14-22-46.tar.bz2
create mode 100644 wd2tools-backup/backup-2011-02-09-14-25-08.tar.bz2
create mode 100644 wd2tools-backup/backup-2011-02-09-14-25-19.tar.bz2
create mode 100644 wd2tools-data/db/wordpress-2011-02-09-14-09-53.sql
create mode 100644 wd2tools-data/db/wordpress-2011-02-09-14-19-24.sql
create mode 100644 wd2tools-data/db/wordpress-2011-02-09-14-19-53.sql
create mode 100644 wd2tools-data/db/wordpress-2011-02-09-14-20-34.sql
create mode 100644 wd2tools-data/db/wordpress-2011-02-09-14-22-45.sql
create mode 100644 wd2tools-data/db/wordpress-2011-02-09-14-25-02.sql
create mode 100644 wd2tools-data/db/wordpress-2011-02-09-14-25-08.sql
create mode 100644 wd2tools-data/db/wordpress-2011-02-09-14-25-19.sql

./wd2tools.py --commit 3.55s user 3.96s system 88% cpu 8.481 total
wk@carlos-v ~/bin (git)-[master] %
```

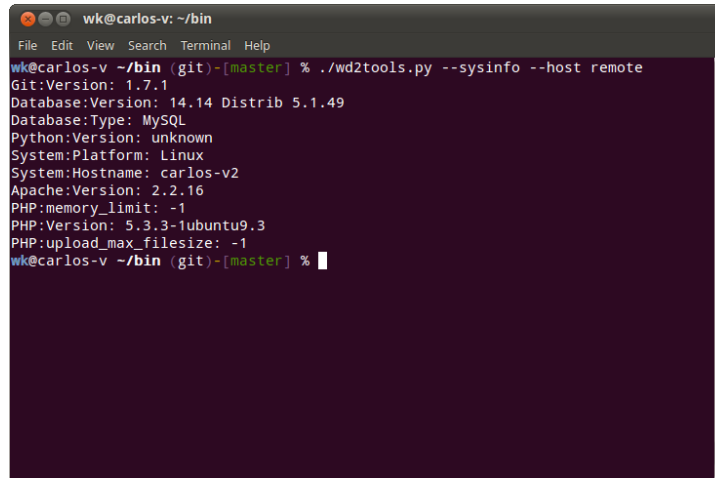
Abbildung 5.11: Versionsverwaltung mit WD2-Tools.

```
wk@carlos-v: ~/bin
File Edit View Search Terminal Help
wk@carlos-v ~/bin (git)-[master] % sudo vim /var/www/projekt1/index.php
1 wk@carlos-v ~/bin (git)-[master] % ./wd2tools.py --sync
Password for Database User:
Database exists, overwrite it? [N]/Y: N
./wd2tools.py --sync 2.70s user 35.35s system 30% cpu 2:03.86 total
wk@carlos-v ~/bin (git)-[master] %
```

Abbildung 5.12: Synchronisation der Instanzen.

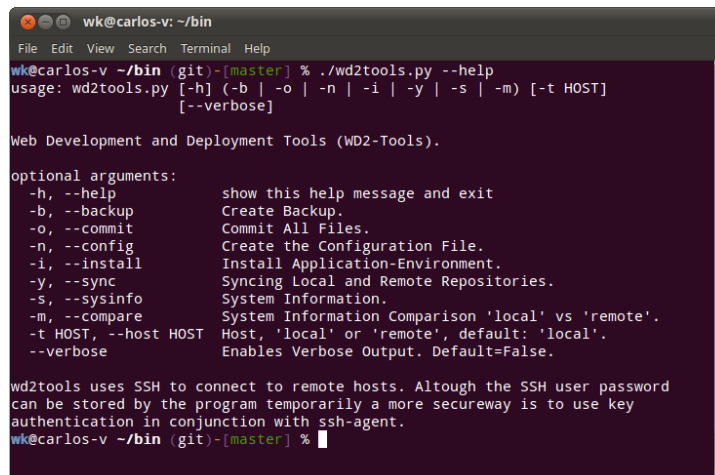
(siehe Abbildung 5.12). `--sync` kommt auch zum Einsatz, wenn die Webanwendung das erste Mal auf den Remotehost übertragen wird.

- Host angeben (`--host`)
Erlaubt ein Befehl die Auswahl des Hosts, wie beispielsweise `--sysinfo` z.B., kann dieser mit `--host` angegeben werden. Hierbei wird bei `--host remote` oder `--host r` der Remotehost aus der Konfigurationsdatei gewählt, bei `--host local` oder `--host l` der lokale Rechner. Systeminformationen des Remotehosts werden z. B. mit dem in Abbildung 5.13 dargestellten Befehl eingeholt.



```
wk@carlos-v: ~/bin
File Edit View Search Terminal Help
wk@carlos-v ~/bin (git)-[master] % ./wd2tools.py --sysinfo --host remote
Git:Version: 1.7.1
Database:Version: 14.14 Distrib 5.1.49
Database:Type: MySQL
Python:Version: unknown
System:Platform: Linux
System:Hostname: carlos-v2
Apache:Version: 2.2.16
PHP:memory_limit: -1
PHP:Version: 5.3.3-1ubuntu9.3
PHP:upload_max_filesize: -1
wk@carlos-v ~/bin (git)-[master] %
```

Abbildung 5.13: Informationen des Remotehosts.



```
wk@carlos-v: ~/bin
File Edit View Search Terminal Help
wk@carlos-v ~/bin (git)-[master] % ./wd2tools.py --help
usage: wd2tools.py [-h] (-b | -o | -n | -i | -y | -s | -m) [-t HOST]
               [--verbose]

Web Development and Deployment Tools (WD2-Tools).

optional arguments:
  -h, --help            show this help message and exit
  -b, --backup          Create Backup.
  -o, --commit          Commit All Files.
  -n, --config          Create the Configuration File.
  -i, --install         Install Application-Environment.
  -y, --sync            Syncing Local and Remote Repositories.
  -s, --sysinfo        System Information.
  -m, --compare        System Information Comparison 'local' vs 'remote'.
  -t HOST, --host HOST Host, 'local' or 'remote', default: 'local'.
  --verbose            Enables Verbose Output. Default=False.

wd2tools uses SSH to connect to remote hosts. Although the SSH user password
can be stored by the program temporarily a more secure way is to use key
authentication in conjunction with ssh-agent.
wk@carlos-v ~/bin (git)-[master] %
```

Abbildung 5.14: Hilfestellungen zu WD2-Tools.

- Warnmeldungen ausgeben (`--verbose`)
Gemeinsam mit allen Befehlen lässt sich mit `--verbose` eine ausführliche Ausgabe von Hinweisen und Warnungen erzeugen. Log-Meldungen werden dann nicht nur in die Logdatei `wd2tools.log`, sondern auch auf `stdout` ausgegeben.
- Hilfe (`--help`)
Zeigt kurze Informationen und Hilfestellungen zum Programm und dessen Optionen an (siehe Abbildung 5.14).

Erkenntnisse

Sicherheit vs. Komfort

Deutlich erkennbar war, dass eine Abhängigkeit zwischen dem Maß der Benutzerinteraktion und der Sicherheit des Programms besteht. Soll die Software möglichst autonom arbeiten, sind entsprechende Rechte erforderlich. Wird die Rechtevergabe restriktiver gehandhabt, müssen entsprechende Anmeldedaten bei jeder privilegierten Aktion eingegeben werden.

Systemabhängigkeit

Anfänglich war das Werkzeug zum Deployment unter allen unixoiden Systemen gedacht. Die Schwierigkeit hierbei ist aber, dass diese Systeme stark unterschiedlich aufgebaut sind. Auch wenn die gleichen Softwarekomponenten zum Einsatz kommen so unterscheiden sich die Konfigurationsdateien und -verzeichnisse doch beträchtlich. Dabei sind nicht nur Programme auf UNIX-Systemen wie Mac OS X oder FreeBSD anders konfiguriert als jene unter Linux, sondern auch bei linuxbasierten Systemen variiert die Konfiguration von Distribution zu Distribution. Selbst bei Distributionen, die die gleiche Basis aufweisen, wie es bei *Debian* und *Ubuntu* der Fall ist, sind Unterschiede noch ausreichend groß um ein automatisiertes Skript aus dem Konzept zu bringen. Da es sich bei WD2-Tools um einen Prototypen und kein Programm für den Produktiveinsatz handelt, wurde der Fokus auf debianbasierte Systeme gelegt. Es wurde hauptsächlich *Debian* in der Version 6.0 sowie *Ubuntu* in den Versionen 10.10 sowie 11.04 (beta) verwendet.

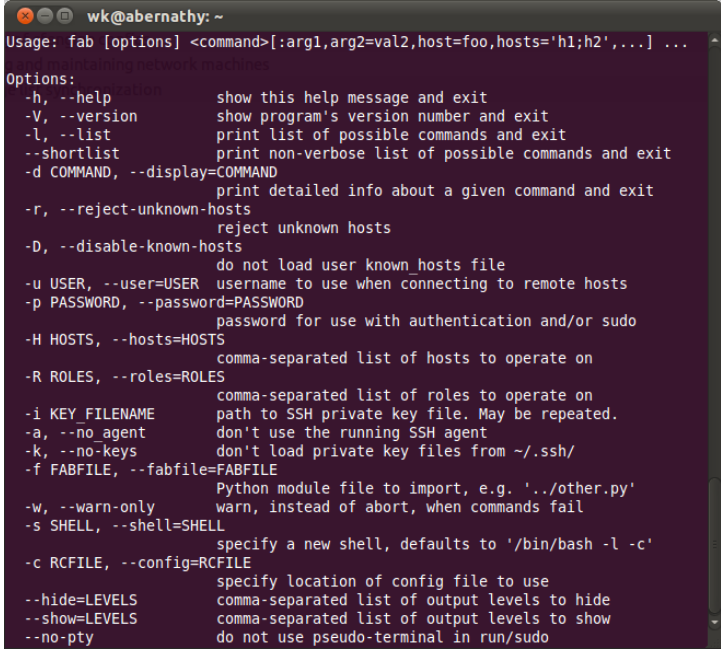
Virtualisierung

In Hinblick auf Virtualisierung sind keine Anpassungen notwendig, die VM muss lediglich über SSH zugänglich gemacht werden.

5.2.3 Fabric

Fabric ist eine in Python geschriebene Bibliothek sowie ein Kommandozeilenwerkzeug, das auf das Deployment von Anwendungen ausgelegt ist. Die Kommunikation erfolgt wie bei den WD2-Tools über SSH. *Fabric* erlaubt die Ausführung von Python-Befehlen auf unterschiedlichen Hosts. Zudem bringt es für das Deployment von Software nützliche Programmfunktionen mit. Damit lassen sich unzählige Szenarien zur Interaktion mit entfernten Rechnern umsetzen. Der einfachste Fall ist die Ausführung von `fab task`, wobei die Funktion `task` zuvor definiert wurde.

Installiert wird *Fabric* am einfachsten über `pip`. Zu beachten ist, dass hier auch die Header-Dateien von Python benötigt werden, und dass zusätzliche



```

wk@abernathy: ~
Usage: fab [options] <command>[:arg1,arg2=val2,host=foo,hosts='h1;h2',...] ...

Options:
-h, --help            show this help message and exit
-V, --version         show program's version number and exit
-l, --list            print list of possible commands and exit
--shortlist          print non-verbose list of possible commands and exit
-d COMMAND, --display=COMMAND
                    print detailed info about a given command and exit
-r, --reject-unknown-hosts
                    reject unknown hosts
-D, --disable-known-hosts
                    do not load user known hosts file
-u USER, --user=USER
                    username to use when connecting to remote hosts
-p PASSWORD, --password=PASSWORD
                    password for use with authentication and/or sudo
-H HOSTS, --hosts=HOSTS
                    comma-separated list of hosts to operate on
-R ROLES, --roles=ROLES
                    comma-separated list of roles to operate on
-i KEY FILENAME      path to SSH private key file. May be repeated.
-a, --no_agent       don't use the running SSH agent
-k, --no-keys        don't load private key files from ~/.ssh/
-f FABFILE, --fabfile=FABFILE
                    Python module file to import, e.g. '../other.py'
-w, --warn-only      warn, instead of abort, when commands fail
-s SHELL, --shell=SHELL
                    specify a new shell, defaults to '/bin/bash -l -c'
-c RCFILE, --config=RCFILE
                    specify location of config file to use
--hide=LEVELS        comma-separated list of output levels to hide
--show=LEVELS        comma-separated list of output levels to show
--no-pty             do not use pseudo-terminal in run/sudo

```

Abbildung 5.15: Argumente in *Fabric*.

Abhängigkeiten zu Python-Bibliotheken bestehen, die gegebenenfalls erst installiert werden müssen. Nicht zu empfehlen ist die Installation über das Paketmanagement der Linuxdistribution, da die dort enthaltene Version meist älter ist.

Nach der Installation wird *Fabric* auf dem Terminalprogramm mit **fab** gestartet. Dabei stehen dem Benutzer zur Interaktion eine Reihe von Möglichkeiten offen. Abbildung 5.15 zeigt eine Übersicht der Optionen.

Die Funktionen werden in eine Datei namens `fabfile.py` geschrieben und mit **fab funktionsname** ausgeführt. Dazu wird `fabfile.py` einfach von *Fabric* importiert, es stehen also sämtliche Pythonfunktionen zur Verfügung.

An die so erstellten Funktionen können auch benutzerdefinierte Argumente übergeben werden. Listing 5.1 zeigt eine einfache Deploymentfunktion in *Fabric*. Die Funktion wird in `fabfile.py` abgelegt und mit **fab deploy** aufgerufen. Es wird versucht, den Quellcode am Server zu aktualisieren (`git pull`). Wenn das entsprechende Verzeichnis jedoch nicht existiert wird eine Version des Repositories im Dateisystem des Servers mittels `git clone` erzeugt. Ein Rechner, auf dem diese Befehle auszuführen sind, wird in SSH-Schreibweise (`user@host:port`) übergeben. Da das hier nicht der Fall war wird interaktiv danach gefragt. Hostnamen werden dabei im global für *Fabric* gültigen Dictionary mit Umgebungsvariablen, `env` abgelegt. Ein Host wird mittels `env.hosts = ['user@host:port']` hinzugefügt. Die Angaben von

Listing 5.1: Einfacher Deploymentschritt mit *Fabric*

```
1 from __future__ import with_statement
2 from fabric.api import local, settings, abort, run, cd
3 from fabric.contrib.console import confirm
4
5 def deploy():
6     code_dir = '/srv/django/myproject'
7     with settings(warn_only=True):
8         if run("test -d %s" % code_dir).failed:
9             run("git clone user@vcshost:/path/to/repo/.git %s" %
10                code_dir)
11         with cd(code_dir):
12             run("git pull")
13             run("touch app.wsgi")
```

`user` und `port` sind optional, der momentane lokale Benutzername und der SSH-Standardport 22 werden beim Weglassen dieser Angaben herangezogen.

Im Vergleich zu WD2-Tools ist *Fabric* weniger auf einzelne Aufgaben spezialisiert, sondern lässt sich universell einsetzen. Ein Nachteil ist, dass die Deploymentschritte erst erstellt werden müssen. *Fabric* ist somit ein Framework für Administrations- und Konfigurationsaufgaben, nicht aber ein ohne Vorarbeit einsetzbares Werkzeug.

Fabric bringt keinerlei spezielle Funktionen für virtualisierte Umgebungen mit, eignet sich aufgrund seines modularen Aufbaus aber sehr gut zum Deployment unter solchen. Ein Vorteil ist es, dass sich ein und derselbe Befehl gleichzeitig auf mehreren VM ausführen lässt.

5.2.4 Puppet

Puppet ist eine in Ruby geschriebene Konfigurations- und Deployment-Werkzeugsammlung. Während sie meist in Netzen mit vielen Rechnern eingesetzt wird ist sie auch für den Fall des Deployments auf einigen wenigen Geräten interessant. Viele Universitäten, aber auch große Unternehmen wie Google setzen zur Serververwaltung auf *Puppet*².

Puppet bietet zwar einen serverless-Modus, wird aber meist im Client-Server-Modus verwendet. Durch die zusätzliche Abstraktionsebene lassen sich mit *Puppet* Rechner auch plattformübergreifend konfigurieren, zumindest, wenn es sich dabei um unixoide Systeme handelt.

In *Puppet* werden sogenannte Ressourcen deklariert und in Klassen zusammengefasst, die dann ausgeführt werden können. Eine sehr einfache Klasse ist in Listing 5.2 gezeigt. Der *Resource Abstraction Layer (RAL)* ist für die Aufteilung der Ressourcen in High-Level-Modelle und plattformabhängige Ressourcen zuständig.

²<http://www.puppetlabs.com/customers/companies/>

Listing 5.2: Klasse *apache* in *Puppet*

```
1 class apache {
2   package {httpd: ensure => installed }
3
4   service { "httpd":
5     ensure => running,
6     require => Package["httpd"],
7   }
8 }
```

Listing 5.3: Bedingte Selektoren in *Puppet* [2]

```
1 service { "apache":
2   name => $operatingsystem ? {
3     debian => "apache2",
4     redhat => "httpd",
5     default => "apache",
6   },
7   ensure => running,
8 }
```

Das in Tabelle 2.2 in Abschnitt 2.1.3 aufgezeigte Problem der unterschiedlichen Paketbezeichnungen lässt sich mit Hilfe bedingter Selektoren in *Puppet* lösen (siehe Listing 5.3).

Puppet bietet außerdem die Möglichkeit der Versionsverwaltung.

Der Vorteil von *Puppet* liegt darin, dass sich virtuelle Instanzen bequem von einer einzigen lokalen Installation von *Puppet* administrieren lassen.

5.2.5 Cfengine

CFEngine ist ein Konfigurationsmanagementsystem, das aus Forschungsarbeiten der Universität von Oslo hervorging. In einer eigenen, plattformunabhängigen Sprache werden Regeln festgelegt, welchen den Idealzustand des Systems beschreiben. Stellt CFEngine eine Diskrepanz zwischen dem Ideal- und Istzustand fest, versucht es, diese wieder in Einklang zu bringen [32]. CFEngine weist ein vergleichsweise komplexe Konfigurationsschnittstelle auf, zeichnet sich jedoch durch einen niedrigen Speicherbedarf aus [33].

Das Listing 5.4 zeigt, wie ein Apache-Webserver eingerichtet wird.

Über eine Schnittstelle zu *libvirt* unterstützt das CFEngine-Framework die meisten Virtualisierungstechnologien [34].

Listing 5.4: Webserver einrichten mit CFEngine

```

1 bundle agent web_server(state)
2 {
3 vars:
4   "document_root" string => "/";
5   "site_http_conf" string => "/home/mark/cfengine-inputs/httpd.conf";
6   "match_package" slist => {
7     "apache2",
8     "apache2-mod_php5",
9     "apache2-prefork",
10    "php5"
11  };
12 processes:
13   web_ok.on::
14     "apache2"
15     restart_class => "start_apache";
16   off::
17     "apache2"
18     process_stop => "/etc/init.d/apache2 stop";
19 commands:
20   start_apache::
21     "/etc/init.d/apache2 start"; # or startssl
22 packages:
23   "$(match_package)"
24     package_policy => "add",
25     package_method => zypper,
26     classes => if_ok("software_ok");
27 files:
28   software_ok::
29     "/etc/sysconfig/apache2"
30     edit_line => fixapache,
31     classes => if_ok("web_ok");
32 reports:
33   !software_ok.on::
34     "The web server software could not be installed";
35 classes:
36   "on" expression => strcmp("$(state)","on");
37   "off" expression => strcmp("$(state)","off");
38 }
39 bundle edit_line fixapache
40 {
41 vars:
42   "add_modules" slist => {
43     "ssl",
44     "php5"
45   };
46   "del_modules" slist => {
47     "php3",
48     "php4",
49     "jk"
50   };
51 insert_lines:
52   "APACHE_CONF_INCLUDE_FILES=$(web_server.site_http_conf)";
53 field_edits:
54   "APACHE_MODULES=.*"
55     edit_field => quotedvar("$(add_modules)","append");
56   "APACHE_MODULES=.*"
57     edit_field => quotedvar("$(del_modules)","delete");
58 }

```

5.3 Betriebssystemkomponenten

5.3.1 ezjail

Im Gegensatz zu den zuvor vorgestellten Lösungen ist *Ezjail* kein Werkzeug zum Verwalten von Deployments sondern es ermöglicht die einfache Administration der in Abschnitt 3.5.1 vorgestellten FreeBSD Jails. Es besteht aus zwei Teilen, dem Programm zum Anlegen, Aktualisieren und Löschen von Jails (`ezjail-admin`) und dem Programm zum Starten und Stoppen von Jails.

Ezjail ist nur eines von mehreren etablierten Programmen zur Verwaltung von Jails, eine Übersicht ist in [35, S. 722] zu finden.

Ezjails setzt die BSD Betriebssystemvirtualisierung Jails voraus und ist nur für diese Virtualisierungstechnologie verwendbar. Es stellt gegenüber der manuellen Einrichtung und Verwaltung von Jails eine deutliche Erleichterung dar.

In Bezug auf das LAMPP-Deployment ist FreeBSD gegenüber dem Deployment auf Linuxdistributionen im Vorteil, da sowohl die Konfiguration als auch die Softwareversionen standardisiert sind und nicht wie bei den Linux-Derivaten zum Teil sehr stark variieren.

5.3.2 LAMPP-Stack

Die einfachste Variante des LAMPP-Deployments ist es, direkt auf XAMPP für Linux zurückzugreifen. Dieses wird während der Entwicklung lokal, üblicherweise unter `/opt/lampp`, eingerichtet und für den Betrieb dann auf den Produktivserver übertragen.

XAMPP weist im Zustand, wie es von der Webseite der *Apache Friends* zu beziehen ist, schwerwiegende Sicherheitsmängel auf. Hier haben sich die XAMPP-Entwickler für den raschen Einstieg in die XAMPP-Verwendung zu Lasten der Berücksichtigung von Sicherheitseinstellungen entschieden. Für den Produktivbetrieb müssen diese Einstellungen deshalb unbedingt geändert werden.

Auch andere Konfigurationen, wie beispielsweise die Portzuweisung, sollten vor dem Produktiveinsatz geändert werden. Außerdem muss darauf geachtet werden, dass sich XAMPP-Programme und eventuell vorhandene native Software nicht gegenseitig behindern. Aus all diesen Gründen ist XAMPP, der Meinung des Autors nach, für den Produktivbetrieb nur in Ausnahmefällen geeignet.

Die verwendete Virtualisierungstechnik ist beim Deployment des XAMPP-Fertigpaketes nicht ausschlaggebend. Solange das Betriebssystem am Entwick-

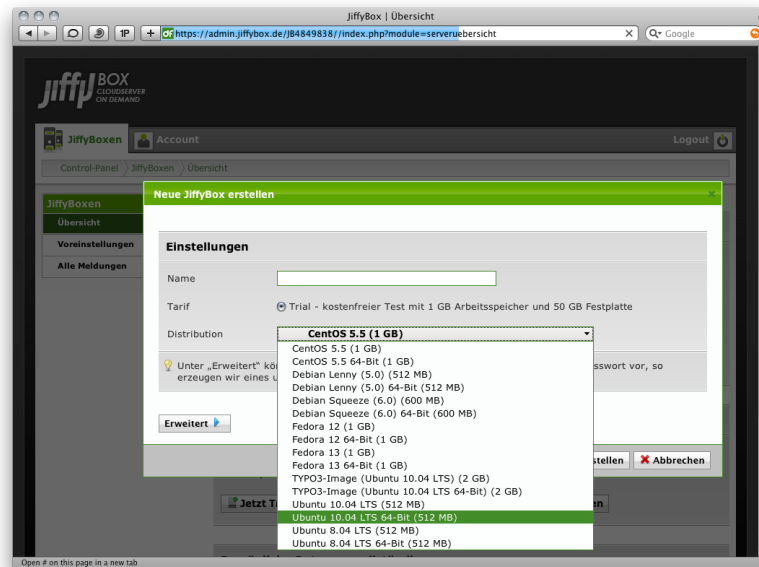


Abbildung 5.16: *Jiffybox* Serverauswahl.

lunsrechner binärkompatibel zu jenem in der virtuellen Instanz ist, sollte das Deployment mit dem Verschieben des XAMPP-Ordners abgeschlossen sein.

5.4 Cloud-Computing

5.4.1 JiffyBox

Jiffybox steht stellvertretend für einen Serveranbieter mit nutzungsbasierter Abrechnung.

Dienste mit derselben Funktionsweise wie *Jiffybox* können als Verbindungselement zwischen VPS zu IAAS-Diensten betrachtet werden. Eine Serverinstanz ist binnen Sekunden angelegt, wie das in Abbildung 5.16 für die aktuelle *Long-Term-Support*-Version von Ubuntu gezeigt wird. Das Deployment der LAMP-Anwendungen obliegt dem Benutzer, Kopien von Instanzen können per Klick angefertigt und eingesetzt werden. Die Benutzeroberfläche zum Duplizieren einer Instanz wird in Abbildung 5.17 dargestellt.

Weitere Vertreter dieser Kategorie sind u. a. *Linode*³, *Rackspace*⁴ und *ElasticHosts*⁵.

³<http://www.linode.com/>

⁴<http://www.rackspace.com/cloud/>

⁵<http://www.elastichosts.com/>

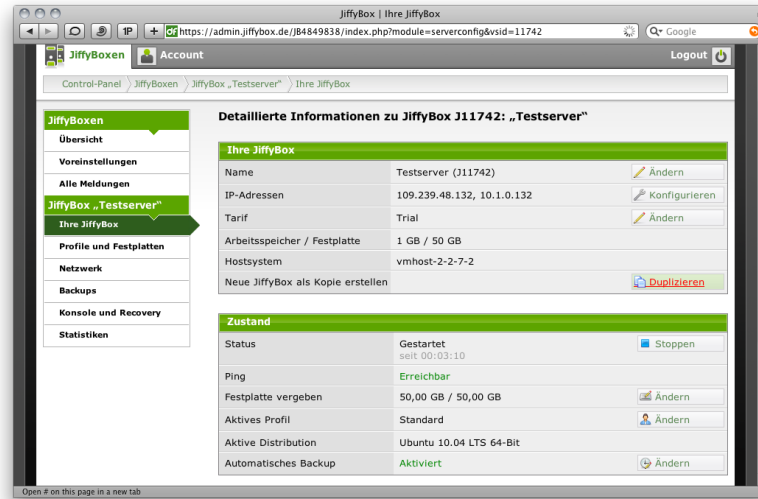


Abbildung 5.17: Jiffybox Serverauswahl.

5.4.2 Bitnami und JumpBox

Das von der spanischen Firma *BitRock* verwaltete Open-Source-Projekt *Bitnami* stellt drei Produkte zur Verfügung:

Nativer Installer

Ein ausführbarer Stack von wählbaren Programmen, erhältlich für Windows, Mac OS X und Linux.

VM-Image

Ein Stack wählbarer Programme, bereitgestellt als virtuelles Image.

Cloudservice

Ein Stack wählbarer Programme, welcher als AMI bereitgestellt wird und sich zur Verwendung mit Amazon EC2 eignet (siehe Abschnitt 4.2.1) oder Cloud Hosting direkt von *Bitnami*, das ebenfalls auf Amazon EC2 aufsetzt.

Die nativen Installer benötigen keinerlei Virtualisierungstechnologien. Es hat auch keine positiven Auswirkungen auf den Deploymentprozess, sie in virtualisierten Umgebungen anstatt physischen Instanzen auszuführen.

Die virtuellen Images hingegen eignen sich besonders gut für das Deployment. Sie ermöglichen es, die Entwicklung unter demselben System durchzuführen, das danach für den Produktivbetrieb verwendet wird.

Diese Variante des Deployments setzt natürlich Vertrauen in den Anbieter der fertigen Images voraus. Zum einen, weil nicht ausgeschlossen werden kann, dass über den Anbieter selbst oder einen unerlaubten Zugriff Schadsoftware

auf die virtuelle Maschine gelangt. Zum anderen ist zu überprüfen, ob die vorgenommene Konfiguration im Allgemeinen und besonders in Hinblick auf Sicherheitseinstellungen den eigenen Anforderungen genügt. In der Standardeinstellung gibt der Browser beim Aufrufen jedenfalls keine Informationen über den Serverpreis und auch vorinstallierte Konfigurationsdienste, wie *phpMyAdmin*, sind nur von der *localhost*-IP-Adresse aus erreichbar.

*Jumpbox*⁶ ist ebenfalls ein Anbieter von Virtual-Machine-Images. Die Einrichtung wird bei *Jumpbox* vor allem durch die webbasierte Oberfläche vereinfacht (siehe Abbildung 5.18).

Gegen monatliches Entgelt wird eine Bibliothek mit Images von über 50 Webanwendungen in jeweils aktueller Version geboten, wovon die meisten auf dem klassischen LAMPP-Stack basieren. Damit ist der größte Teil an prominenten LAMPP-Anwendungen abgedeckt. Wird die benötigte Anwendung nicht angeboten, gibt es die Möglichkeit, auf die generische LAMPP-*Jumpbox* zurückzugreifen. Eingerichtet wird die *Jumpbox* über einen Textinstaller (siehe Abbildung 5.21), administriert werden die Instanzen danach über eine Weboberfläche, wie sie in Abbildung 5.19 zu sehen ist. Eine Anleitung zum Deployment von Webanwendungen wird mitgeliefert (siehe Abbildung 5.20).

Das selbsternannte Ziel von *Jumpbox* ist es, überall zu funktionieren, wo Virtualisierungsinfrastruktur vorhanden ist. Die von *Jumpbox* unterstützten Lösungen werden in Gruppen nach der Güte der Unterstützung sortiert. Am besten unterstützt werden sämtliche Virtualisierungs-Technologien von VMware sowie Amazons Clouddienst EC2. Mit kleineren Einschränkungen unterstützt werden die Desktop- und Servervirtualisierungsprodukte von Parallels, nicht aber deren Virtuozzo-Container. *VirtualBox* funktioniert ebenfalls, sofern *Physical Address Extension* (PAE) aktiviert wird. Nicht offiziell unterstützt werden hingegen Xen, KVM und QEMU. Zur Gruppe der Virtualisierungslösungen auf welchen *Jumpbox* nicht lauffähig ist, zählen Virtualisierungstechnologien von Microsoft, OpenVZ- und die darauf basierenden Virtuozzo-Containerformate.

Neben *Jumpbox* gibt es weitere Deploymentlösungen die auf demselben Konzept basieren, uns sich nur im Detail unterscheiden, weshalb sie an dieser Stelle nicht vorgestellt werden.

5.4.3 Beanstalk

*Beanstalk*⁷ bietet einen anderen Ansatz zum Deployment von Webanwendungen. Ein bestehendes oder neu angelegtes, lokales Verzeichnis, das unter einem *Version Control System* (VCS) steht, ist der Ausgangspunkt für die

⁶<http://www.jumpbox.com/>

⁷<http://beanstalkapp.com/>

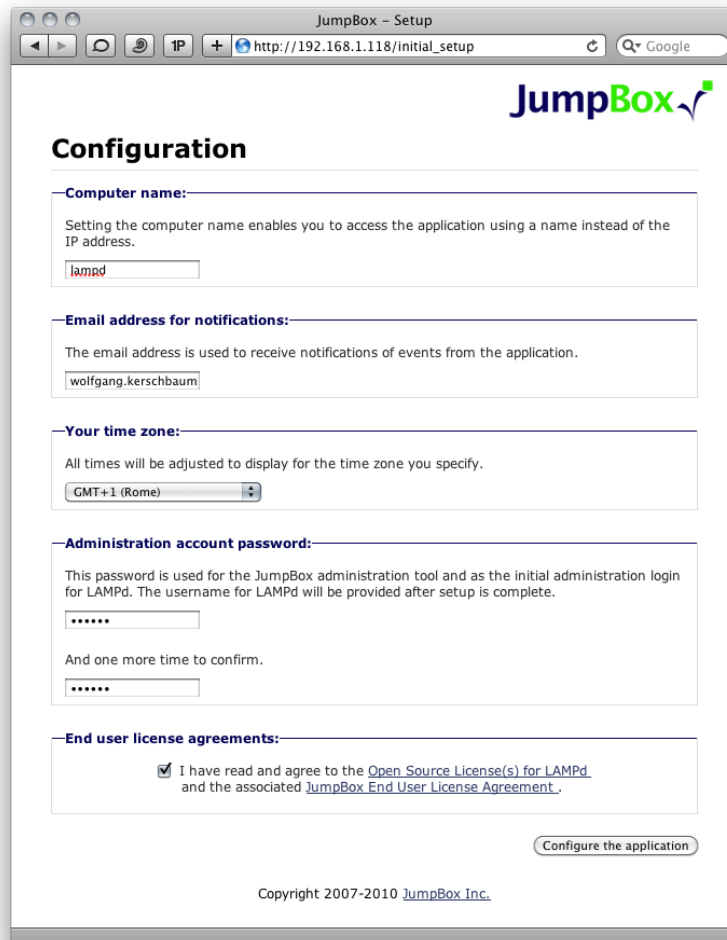
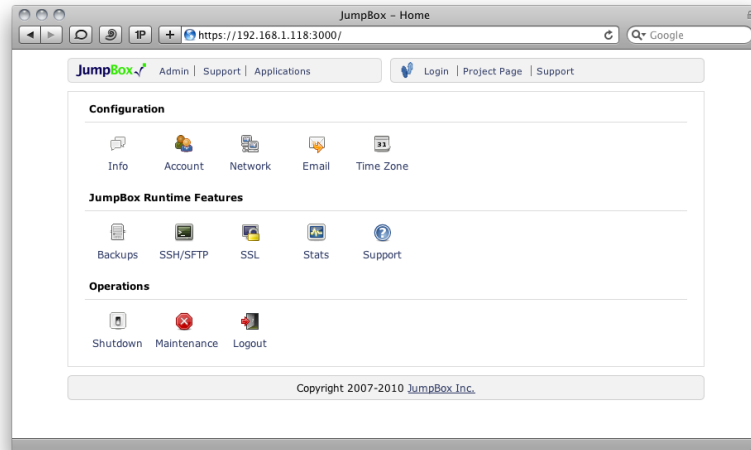
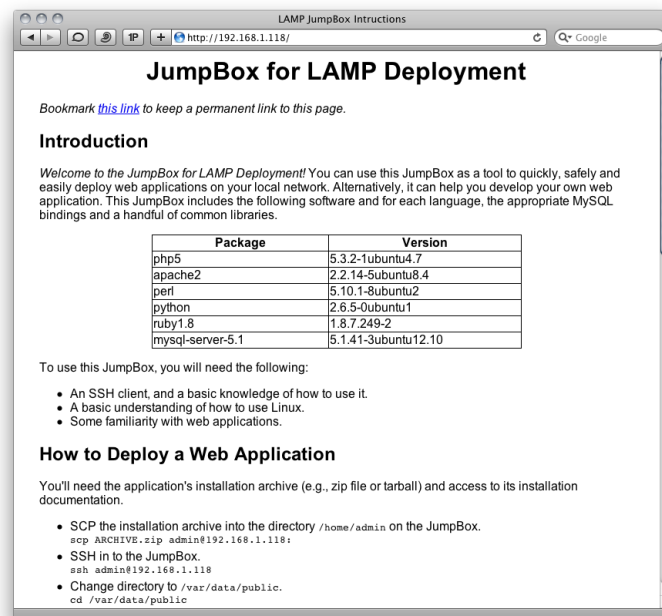


Abbildung 5.18: Der *Jumpbox* Web-Konfigurationsdialog.

Verwendung von *Beanstalk*. Zur Auswahl standen zum Testzeitpunkt *Git* und *SVN*. Das lokale Verzeichnis wird mit dem zuvor eingerichteten *Beanstalk*-Account verknüpft. Zur Kommunikation wird *SSH* verwendet, dazu müssen die öffentlichen Schlüssel *Beanstalk* bekanntgegeben werden.

Bei *Beanstalk* handelt es sich in erster Linie um einen Online-Anbieter zum Hosten von Repositories. Durch das Hinzufügen von Servern kann ein bei *Beanstalk* gehostetes Projekt über *FTP* oder *SSH File Transfer Protocol (SFTP)* zum Deployment auf diesen verwendet werden. Es gibt verschiedene Möglichkeiten, das Deployment auszulösen:

- automatisch mit jedem Commit.
- manuell über das Webinterface.

Abbildung 5.19: Die *Jumpbox* Admin-Oberfläche.Abbildung 5.20: *Jumpbox* für LAMP-Entwicklung Benutzungsanleitung.

- durch Schlüsselwörter in der Commit-Nachricht.

Da bei den meisten komplexen Projekten das Deployment neben der Übernahme der aktuellen Daten weitere Arbeitsschritte notwendig sind, bietet *Beanstalk* an, „Web Hooks“ zu definieren, die vorbereitende oder abschließende Aufgaben durchführen. Das Aktualisieren der Datenbank, erneutes Kompilieren

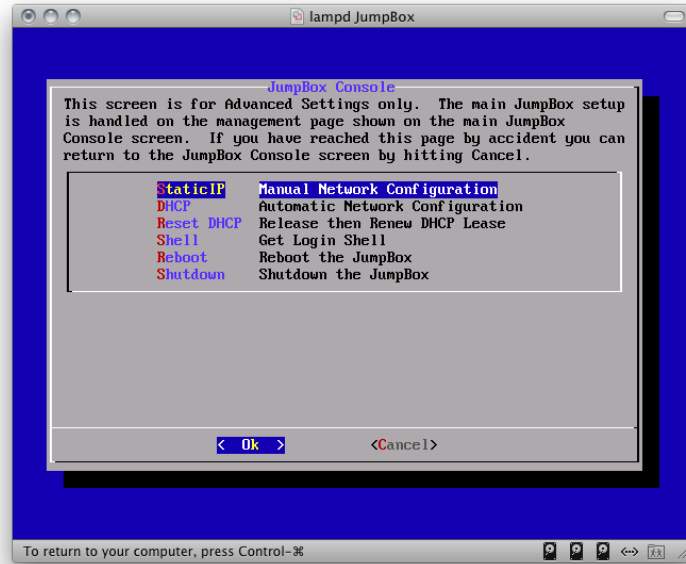


Abbildung 5.21: LAMPP-Version der *Jumpbox* in VMware Fusion.

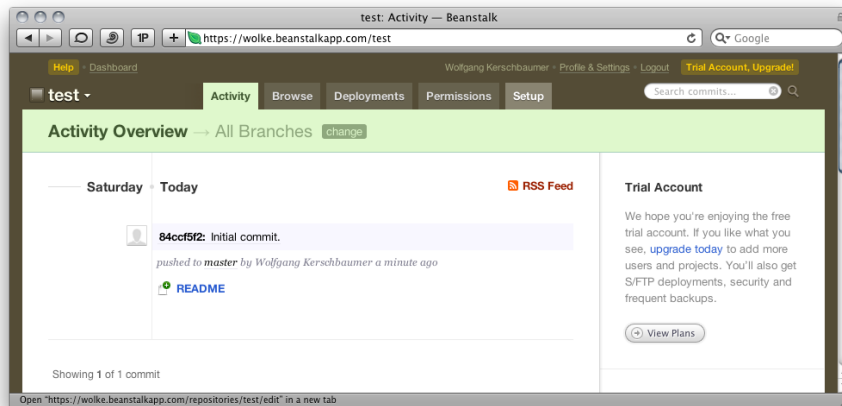


Abbildung 5.22: Aktivitätsübersicht *Beanstalk*.

ren oder das Setzen von Dateirechten sind Beispiele für solche zusätzlichen Aktionen.

Neben der Verwendung von FTP oder SFTP-Servern können auch unterstützte Clouddienste wie z. B. Rackspace eingesetzt werden.

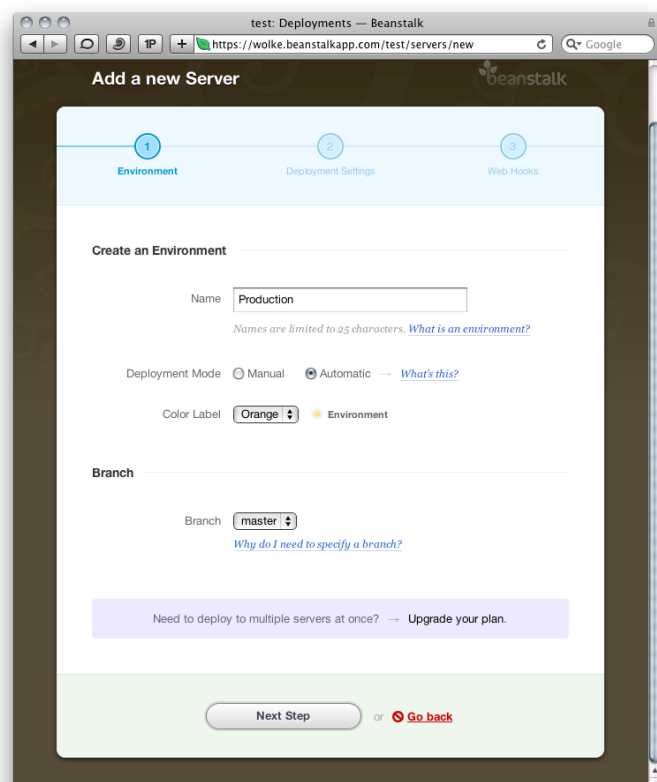


Abbildung 5.23: Umgebung einrichten mit *Beanstalk*.

Kapitel 6

Fazit und Schlussbemerkung

Während sich Virtualisierung in großen Unternehmen bereits durchgesetzt hat, ist der Verwendung von virtuellen LAMPP-Instanzen für kleinere bis mittlere Anwendungen bis vor Kurzem vor allem das Fehlen kostengünstiger Infrastruktur im Wege gestanden. Mit einer stetig wachsenden Anzahl an cloudbasierten Entwicklungs- und Deploymentlösungen kann der Deploymentprozess im Vergleich zur konventionellen Vorgehensweise schon jetzt deutlich schneller, weniger fehleranfällig und kostengünstiger durchgeführt werden. Im Idealfall wird mit vorkonfigurierten, virtuellen Images gearbeitet und das Deployment besteht darin, die auf dem Testserver verwendete LAMPP-Instanz auf den Produktivserver zu übertragen und aktiv zu schalten.

Generell lässt sich feststellen, dass sich virtuelle Images besonders bei LAMPP-Stacks mit aufwendiger Konfiguration und besonderem Setup rechnen. Je komplexer der Stack und die Serveranpassungen sind, desto sinnvoller ist es, diesen Stack zu virtualisieren.

Durch die Verwendung unklar definierter Modewörter wie „die Cloud“ ist es nicht einfach, sich einen Überblick der Deploymentkonzepte und -produkte zu verschaffen. Die Mühe lohnt sich aber definitiv, da die Aufwandsersparnis je nach Anwendungsfall beträchtlich sein kann.

Außer Frage stehen die Vorteile virtualisierter Instanzen in Verbindung mit Clouddiensten, wenn rasche Skalierbarkeit von hoher Bedeutung für den Betreiber ist. Das ist vor allem für Services relevant, die kurzzeitig hohe Lastspitzen aufweisen. Ein physischer Server ohne Anbindung an ein Cluster müsste dabei so ausgelegt werden, dass er diese Lastspitzen abfangen kann, während in Normalzeiten diese Ressourcen gar nicht benötigt werden. Auch das Cracken von Passwörtern¹ oder gar Angriffe auf Netzwerke, wie beispielsweise auf Sonys PlayStation-Netzwerk im Mai 2011² stellen solche

¹<http://news.electricalchemistry.net/2009/10/cracking-passwords-in-cloud.html>

²<http://futurezone.at/digitallife/3125-sony-skandal-taeter-nuetzten-amazons-cloud.php>

Lastspitzen dar und werden daher immer öfter unter Zuhilfenahme von Clouddiensten bewerkstelligt.

Konfigurationswerkzeuge wie CFEngine (siehe Abschnitt 5.2.5) oder *Puppet* (siehe Abschnitt 5.2.4) vereinfachen das Deployment unter virtuellen Instanzen. Die Vorgehensweise unterscheidet sich dabei meist nicht wesentlich von jener bei physikalischen Hosts, die meisten Lösungen bringen dennoch Unterstützung für das Deployment in virtualisierten Umgebungen mit. Zu bedenken ist aber, dass dies eine, je nachdem, wie detailliert Deploymentschritte abgedeckt werden, eine zum Teil erhebliche Einarbeitungszeit in das jeweilige Framework erfordert, die sich am ehesten bei vielen ähnlichen Installationen rechnet.

Die Tatsache, dass durch die Verwendung virtualisierter Instanzen zwar der Deploymentprozess vereinfacht wird, intern die Komplexität der Systeme aber zunimmt, spricht stark für die Verwendung einer automatisierten Deploymentlösung, dabei muss aber bei komplexen zusätzlichen Werkzeugen wie CFEngine oder *Puppet* stets eine gewisse Einarbeitungszeit mit eingerechnet werden. Ansonsten besteht die Gefahr, auch Fehler, die schwerwiegend sein können und unter Umständen erst im Produktivbetrieb auffallen, zu automatisieren. Abhilfe könnten hier standardisierte Verfahren zur Administration von virtuellen Maschinen schaffen, wie sie sich derzeit in Planung befinden [36].

Da sich der Aufbau einer eigenen Virtualisierungsinfrastruktur für kleine bis mittlere Betriebe nur selten lohnt, ist es üblich, Dienste von darauf spezialisierten Anbietern in Anspruch zu nehmen. Neben dem Aspekt, dass die Daten diesem Provider anvertraut werden müssen kommt hinzu, dass im Falle eines Ausfalles die Einflussnahme des Kunden auf die Wiederaufnahme des Betriebs gering ist. Auch darf nicht davon ausgegangen werden, dass Clouddienste einen besseren Schutz gegen Datenverlust darstellen, das Anlegen regelmäßiger lokaler Datensicherungen ist auch bei dieser Hostingvariante Pflicht.

Bei der Cloud-Computing-Lösung muss ein gewisser Kontrollverlust in Kauf genommen werden, der gesamte Entwicklungs- und Deploymentprozess lässt sich nicht weiter vereinfachen, als eine schon vorkonfigurierte virtuelle Maschine bei Bedarf mit einem Klick vom Testserver auf den Produktivserver zu übertragen. Konfigurationsframeworks oder eigens entwickelte Programme wie WD2-Tools sind im Vergleich dazu aufwendig in der Erstellung und Wartung.

Anhang A

Glossar

AMD-V

Virtualisierungs-Erweiterung des Prozessorherstellers *Advanced Micro Devices* (AMD).

AMI

Amazon Machine Image.

Apache

Apache HTTP Server.

APT

Advanced Packaging Tool.

Ausnahme

Eine *emph*Ausnahme oder *Exception* ist ein Programmzustand, meist Fehler, der Informationen an eine andere Ausführungsebene zur Behandlung weiterreicht.

AWS

Amazon Web Services.

Beanstalk

*Beanstalk*¹ LAMPP-Deploymentdienst.

BIND

Berkeley Internet Name Domain.

Bitnami

*Bitnami*² bietet native Installer, VM und Cloudhosting-Lösungen an.

BSD

Berkeley Software Distribution, Abzweigung des originalen AT&T UNIX-Systems.

¹<http://beanstalkapp.com/>

²<http://bitnami.org/>

CFEngine

CFEngine³ ist ein in C geschriebenes Konfigurationsmanagementsystem.

chroot

chroot ist der Name des Programms und Systemaufrufs um das Rootverzeichnis eines Prozesses zu ändern.

Citrix Systems

Citrix Systems ist ein 1990 gegründetes, auf Virtualisierung und SAAS spezialisiertes Unternehmen mit Sitz in Florida. Seit der Übernahme von *XenSource* in 2007 zeichnet es für die Leitung der freien Virtualisierungslösung Xen verantwortlich.

Cloud-Computing

ein vom Marketing geprägter, nicht klar abgegrenzter Begriff für sich in einem Netzwerk befindliche Computerressourcen und -dienste.

Debian

Debian ist eine besonders auf Servern häufig eingesetzte Linuxdistribution. Sie wird ausschließlich mit quelloffener, GPL-basierter Software ausgeliefert. Die vollständige Bezeichnung lautet Debian GNU/Linux, wird in dieser Arbeit aber mit *Debian* abgekürzt.

DNS

Domain Name System.

Dom0

Die *Domäne 0* oder kurz *Dom0*, ist das erste Gastbetriebssystem einer Xen-Virtualisierungsumgebung, übernimmt wichtige Managementfunktionen auch für weitere Xen-Gäste.

DomU

Domäne U oder *DomU* ist die Bezeichnung für unprivilegierte Gastsysteme bei Xen.

EC2

Elastic Compute Cloud.

Ezjail

*Ezjail*⁴ ist ein Administrationsprogramm für BSD-Jails.

Fabric

Bei *Fabric* handelt es sich um eine Pythonbibliothek und ein Deploymentframework.

³<http://cfengine.com>

⁴<http://erdgeist.org/arts/software/ezjail/>

Foss

Free and Open Source Software.

FreeBSD

FreeBSD ist eine Nachfolger von BSD UNIX und heute neben OpenBSD und NetBSD eine der am weitesten verbreiteten BSD-Varianten.

FTP

FTP steht für *File Transfer Protocol*, nach Meinung des Autors ein nicht mehr zeitgemäßes Protokoll zur Datenübertragung in Netzwerken. Findet dennoch auch heute noch breite Anwendung.

Git

*Git*⁵ ist ein verteiltes, von Linus Torvalds ins Leben gerufenes VCS. Git wurde zum Verwalten des Linux-Kernel-Quellcodes entwickelt und wird seit seiner Einführung im Jahre 2005 für zahlreiche weitere Softwareprojekte verwendet, darunter auch für einige in dieser Arbeit besprochene Projekte, wie z. B. KVM oder Puppet.

GPL

GNU *General Public License*.

GTK+

GIMP *Toolkit*.

GUI

Graphical User Interface.

HTTP

Hypertext Transfer Protocol.

HVM

Hardware Virtual Machine.

IAAS

Infrastructure as a Service.

IBM

International Business Machines, eine der größten 20 amerikanischen Firmen, tätig im Technologiebereich mit Hardware- und Softwareprodukten, gegründet 1911 als *Computing Tabulating Recording Corporation*.

IIS

Internet Information Services.

IPv6

Internet Protocol Version 6.

⁵<http://git-scm.com/>

ISP

Internet Service Provider.

Jiffybox

*JiffyBox*⁶ ist ein Anbieter von Cloudservern.

Js/Linux

Js/Linux⁷ ist eine Emulation eines Linux-PC in JavaScript.

Jumpbox

*Jumpbox*⁸ ist ein Anbieter von VM für Cloudhosting.

KVM

Kernel Virtual Machine.

LAMP

Stack für Webanwendungen dessen Bezeichnung sich aus den Anfangsbuchstaben der verwendeten Komponenten zusammensetzt: Linux, Apache, MySQL, PHP, Perl oder Python.

libvirt

libvirt ist eine freie Bibliothek zum Verwalten unterschiedlicher Virtualisierungslösungen.

Linux

Ein vom finnischen Studenten Linus Torvalds 1991 ins Leben gerufener, quelloffene und kostenlose Betriebssystemkernel. Der heute üblichen Verwendung nach bezeichnet der Begriff „Linux“ in dieser Arbeit aber den Betriebssystemkernel in Verbindung mit einer Reihe von Programmen, die für den Erfolg von Linux ebenfalls maßgebend waren, allen voran die GNU-Programme. Viele Systemeigenschaften von Linux sind jenen von UNIX nachempfunden, basieren aber nicht auf demselben Quellcode.

Mac OS x

UNIX-basiertes, proprietäres Desktop-Betriebssystem von Apple Inc.

MySQL

MySQL ist ein weit verbreitetes, unter GPL stehendes RDBMS von *Oracle Corporation*.

NAT

Network Address Translation.

⁶<https://www.jiffybox.de/>

⁷<http://bellard.org/jslinux/>

⁸<http://www.jumpbox.com/>

OpenVz

freie Betriebssystemvirtualisierungslösung, Basis von *Parallels Virtuozzo*⁹.

Oracle Corporation

Oracle Corporation ist ein Hard- und Softwareunternehmen mit 105000 Mitarbeitern (Stand 2010).

PAAS

Platform as a Service.

PAE

Physical Address Extension.

paramiko

*paramiko*¹⁰ ist ein SSH-Modul für Python.

PHP

PHP: Hypertext Preprocessor, rekursives Akronym, Skriptsprache.

pip

*pip*¹¹ ist ein Paketinstallationsprogramm für Python.

Puppet

*Puppet*¹² ist ein in Ruby geschriebenes Konfigurationsframework.

PXE

Preboot Execution Environment.

QEMU

QEMU ist ein von F. Bellard ins Leben gerufener Prozessor-Emulator.

RDBMS

Relational Database Management System.

RHEL

Red Hat Enterprise Linux.

Ringsystem

Historisch gewachsenes Sicherheitskonzept, besteht aus mehreren Ringen, *Ring 0* ist der Kernelmode, *Ring 3* der User-Mode. *Ring 1* und *Ring 2* werden aus Kompatibilitätsgründen in modernen Betriebssystemen nicht verwendet. In Abbildung A.1 ist das Ringsystem grafisch dargestellt .

⁹<http://wiki.openvz.org/>

¹⁰<http://www.lag.net/paramiko>

¹¹<http://www.pip-installer.org/>

¹²<http://www.puppetlabs.com/>

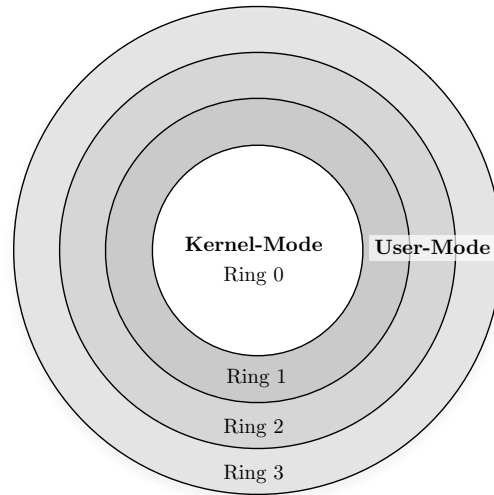


Abbildung A.1: Ringsystem.

RPM

RPM *Package Manager*, rekursives Akronym.

SAAS

Software as a Service.

SFTP

SSH *File Transfer Protocol*.

SSH

Secure Shell. Netzwerkprotokoll, das verschlüsselte Datenübertragung von Rechnern über Netzwerke ermöglicht. SSH ist der Standard unter den sicheren Kommunikationswegen. Diese Arbeit bezieht sich mit SSH stets auf OpenSSH, die Implementierung des OpenBSD-Teams.

SVN

Apache Subversion oder auch SVN ist ein weit verbreitetes VCS. Weist Nachteile gegenüber verteilten Versionskontrollsystemen wie *Git* auf.

TCP

Transmission Control Protocol.

Ubuntu

Ubuntu ist ein auf *Debian* basierendes Linuxbetriebssystem. Aktuell die am weitesten verbreitete Linuxdistribution¹³. *Ubuntu* wird vorwiegend für den Desktopbetrieb eingesetzt, eine angepasste Server-Version ist

¹³<http://distrowatch.com/>

ebenfalls erhältlich. Die Firma, die die Ubuntu-Entwicklung leitet, wurde von M. Shuttleworth gegründet und heißt *Canonical*.

UML

User-mode Linux.

UNIX

UNIX ist ein von AT&T-Mitarbeitern an den *Bell Labs* entwickeltes Mehrbenutzersystem. Die erste Version stammt aus 1969. UNIX-Betriebssysteme, die noch heute im Einsatz sind, gehören HP-UX, AIX, (*Open*) *Solaris* und sämtliche BSD-Varianten von UNIX, darunter auch Mac OS x. Einen kompakten Abriss über die Geschichte von UNIX mit weiterführenden Verweisen bietet [37, S. 985].

URI

Uniform Resource Identifier.

VCS

Version Control System.

virsh

virsh ist eine Kommandozeilenschnittstelle für *libvirt*.

Virtual Appliance

Eine *Virtual Alliance* ist ein vorgefertigter, betriebsfertiger Softwarestack für VM.

VirtualBox

*VirtualBox*¹⁴ ist eine Virtualisierungslösung von *Oracle Corporation*.

Virtuozzo

proprietäre Betriebssystemvirtualisierungslösung von *Parallels*, basierend auf OpenVz.

VM

virtuelle Maschine.

VMM

Virtual Machine Monitor, auch *Hypervisor*.

Vmware

Vmware ist ein kalifornisches Unternehmen, das sich auf Desktop- sowie Server-Virtualisierung spezialisiert hat.

VPS

Virtual Private Server.

VT-x

Virtualisierungserweiterung des Prozessorherstellers Intel.

¹⁴<http://www.virtualbox.org/>

WD2-Tools

Kommandozeilenframework für Deployment über SSH.

x86

Auf dem Intel 8086 Prozessor basierendes Befehlsset, heute in der Virtualisierung übliche ist die 64-Bit-Variante (x86-64).

XAMPP

XAMPP ist gleichbedeutend mit LAMPP, mit der Ausnahmer, dass diese Bezeichnung sich nicht auf Linux als Betriebssystem beschränkt sondern das „X“ die Plattformunabhängigkeit ausdrückt („x“ für eng. „crossplatform“).

Xen

Ein von Citrix Systems verwalteter, freier und quelloffener VMM.

X-Window-System

Standard-Prorokoll zur Anzeige von Fenstern auf unixoiden Systemen.

Literaturverzeichnis

- [1] AMBERG, E.: *Linux Server mit Debian GNU/Linux*. mitp, Heidelberg, 2. Aufl., 2009.
- [2] TURNBULL, J.: *Pulling Strings with Puppet: Automated System Administration Done Right*. Apress, New York City, 2008.
- [3] STEIDLER-DENNISON, T.: *Mac for Linux Geeks*. Apress, New York City, 2009.
- [4] RUEST, D. und N. RUEST: *Virtualization: A Beginner's Guide*. McGraw Hill, New York City, 2009.
- [5] POPEK, G. und R. GOLDBERG: *Formal Requirements for Virtualizable Third Generation Architectures*. Commun. ACM 17:412–421, Juli 1974.
- [6] ADAMS, K. und O. AGESEN: *A Comparison of Software and Hardware Techniques for x86 Virtualization*. SIGPLAN, 41:2–13, Okt. 2006.
- [7] GOLDBERG, R.: *Architecture of Virtual Machines*. In: *Proceedings of the Workshop on Virtual Computer Systems*, S. 74–112, 1973.
- [8] HEISER, G.: *Much Ado About Type-2*, Okt. 2010. <http://www.ok-labs.com/blog/entry/much-ado-about-a-type-2/>
- [9] HEISER, G.: *Virtualization: Some get it, some don't*, März 2009. <http://www.ok-labs.com/blog/entry/some-get-it-some-dont/>
- [10] ROBIN, J. und C. Irvine: *Analysis of the Intel Pentium's Ability to Support a Secure Virtual Machine Monitor*. In: *Proceedings of the Ninth Conference on USENIX Security Symposium - Volume 9*, S. 10, Denver, Colorado, 2000, USENIX Association.
- [11] RODEWIG, K.: *Webserver einrichten und administrieren* Galileo Computing, Bonn, 2009.
- [12] DIEDRICH O.: *Linux virtuell*. In: *c't Server-Praxis*, 2011.

- [13] COEKAERTS W.: *Linux Mainline Contains All the Xen Code Bits for Dom0 and DomU Support*, Mai 2011. http://blogs.oracle.com/wim/entry/linux_mainline_contains_all_the
- [14] KANIA, S. und D. DEIMEKE: *Linux-Server* Galileo Computing, Bonn, 2011.
- [15] ENBERG, P.: *Announcement of Native Linux KVM Tool.*, März 2011. <https://lkml.org/lkml/2011/3/31/406>
- [16] LORD, D.: *Five Tips for Effective Backup and Recovery in Virtual Environments*. März 2011, Quest Software.
- [17] RUTKOWSKA J.: *Introducing Blue Pill*, Juni 2006. <http://visiblethings.blogspot.com/2006/06/introducing-blue-pill.html>
- [18] HUGOS M. UND D. HULITZKY: *Business in the Cloud: What Every Business Needs to Know About Cloud Computing*. John Wiley & Sons, 2010.
- [19] GROSS, D.: *WikiLeaks Cut Off from Amazon Servers*, Dez. 2011. <http://edition.cnn.com/2010/US/12/01/wikileaks.amazon/>
- [20] FALKNER J. und C. CHRISTMANN: ix - Magazin für professionelle Informationstechnik 5:38–43, Mai 2011.
- [21] KOPP M.: *Trotz Hype: Nicht alles geht in der Cloud*. ix - Magazin für professionelle Informationstechnik 5:38–43, Mai 2011.
- [22] SERENA, D.: *Implementing and Developing Cloud Computing Applications*. Auerbach Publications, Boston, 2010.
- [23] GURAV, U. und R. SHAIKH: *Virtualization – A Key Feature of Cloud Computing*. In: *Proceedings of the International Conference and Workshop on Emerging Trends in Technology*. ICWET '10, S. 227–229, Mumbai, Feb. 2010.
- [24] GAO A. und DIAO L.: *Lazy update propagation for data replication in Cloud Computing*. In: *Pervasive Computing and Applications (ICPCA)*, S. 250–254, Maribor, Jan. 2010.
- [25] WILLIAMS, M.: *A Quick Start Guide to Cloud Computing: Moving Your Business into the Cloud*. Kogan Page Ltd., 2010.
- [26] CZERNICKI, B.: *IaaS, PaaS and SaaS Terms Clearly Explained and Defined*, Feb. 2011. <http://www.silverlighthack.com/post/2011/02/27/laaS-PaaS-and-SaaS-Terms-Explained-and-Defined.aspx>

- [27] KECSKEMETI, G.: *Automatic Service Deployment Using Virtualisation*. In: *Parallel, Distributed and Network-Based Processing*, S. 628–635, Toulouse, Feb. 2008.
- [28] CHEN, B.: *Fast, On-Demand Software Deployment with Light-weight, Independent Virtual Disk Images*. In: *Grid and Cooperative Computing, Eight International Conference on*, S. 16–23, Lanzhou, Apr. 2009.
- [29] MÜLLER, T. UND A. KNOLL *Virtualization Techniques for Cross Platform Automated Software Builds, Tests and Deployment*. In: *Software Engineering Advances, ICSEA '09. Fourth International Conference on*, S. 73–77, Porto, Sept. 2009.
- [30] SUN, C.: *Simplifying Service Deployment with Virtual Appliances*. In: *Services Computing, 2008. SCC '08, IEEE International Conference on*, S. 265–272, Honolulu, Juli 2008.
- [31] LÜTTICKE, T. und DOTZAUER T.: *Das SSH-Buch*. Nicolaus Millin Verlag, Lohmar, 2007.
- [32] NEMETH, E., G. SNYDER UND T. HEIN: *Linux Administration Handbook*. Prentice Hall, New Jersey, 2007.
- [33] CLARKE, J.: *Why We Use Cfengine: Memory Footprint*, Feb. 2011. <http://blog.normation.com/2011/02/23/why-we-use-cfengine-memory-footprint/>
- [34] CFEngine: *CFEngine Special Topic Virtualization*, Dez. 2010.
- [35] DANKOWEIT, J.: *FreeBSD - Installieren, Konfigurieren, Vernetzen*. C&L Computer und Literaturverlag, Böblingen, 2009.
- [36] HANTTELMANN, F.: *Virtuelle Umgebungen ins Systemmanagement integrieren*. ix Special Cloud, Grid Virtualisierung, 2:132–136, Juni 2010.
- [37] WILLEMER, A.: *UNIX – Das umfassende Handbuch*. Glileo Computing, Bonn, 2008.