

KVM

für die Server- Virtualisierung

Michael Kofler, Ralf Spenneberg

Von Konfiguration
und Administration
bis Clustering und Cloud

ebooks.kofler

Michael Kofler, Ralf Spenneberg

KVM für die Server-Virtualisierung

Von Konfiguration und Administration
bis Clustering und Cloud

KVM für die Server-Virtualisierung

Von der Konfiguration und Administration bis Clustering und Cloud.

© Michael Kofler, Ralf Spenneberg und ebooks.kofler 2012, 2014

Autoren: Michael Kofler, Ralf Spenneberg
Korrekturat: Friederike Daenecke
ISBN PDF: 978-3-902643-22-3
Verlag: ebooks.kofler, Schönbrunngrasse 54c, 8010 Graz, Austria

Die PDF-Ausgabe dieses eBooks ist hier erhältlich:

<https://kofler.info/ebooks/kvm-2012/>

Viele in diesem eBook genannten Hard- und Software-Bezeichnungen sind geschützte Markennamen.

Dieses eBook wurde mit großer Sorgfalt verfasst. Dennoch sind Fehler nicht ganz auszuschließen. Für allfällige Fehler kann keine Verantwortung oder Haftung übernommen werden. Verbesserungsvorschläge oder Korrekturen sind selbstverständlich willkommen (ebooks@kofler.info). Vielen Dank dafür!

Dieses eBook ist durch das österreichische Urheberrecht geschützt. Sie dürfen das eBook für den persönlichen Gebrauch kopieren und ausdrucken, aber nicht an andere Personen weitergeben, weder in elektronischer noch in anderer Form.

Hinweis: Dieses eBook basiert auf dem gleichnamigen, ursprünglich 2012 bei Addison-Wesley erschienenen Buch. Für diese Ausgabe wurden alle bis zum November 2014 bekannten Fehler korrigiert. Davon abgesehen wurde der Text aber nicht aktualisiert. KVM wird in diesem Buch also in den im Frühjahr 2012 verfügbaren Funktionen dargestellt.

Inhaltsverzeichnis

Vorwort	9
1. Grundlagen	11
1.1 Virtualisierungsgrundlagen	11
1.2 QEMU, KVM und libvirt	14
1.3 Virtuelle Hardware-Komponenten	20
2. Hello World!	25
2.1 Voraussetzungen	25
2.2 Der Virtual Machine Manager	27
2.3 Linux-Gast einrichten	33
2.4 Konfigurationstipps für Fedora- und RHEL-Gäste	38
2.5 Konfigurationstipps für Ubuntu-Gäste	42
2.6 Windows-Gast einrichten	45
2.7 Fertige Appliances und vereinfachte Gast-Installation	49
3. KVM per Kommandozeile steuern	53
3.1 KVM-Kommando	53
3.2 virsh	59
3.3 libvirt-Kommandos	61
3.4 Migration und Konvertierung von virtuellen Maschinen	67
3.5 libvirt-Konfiguration und -Interns	69
3.6 Verhalten beim Neustart des Hostsystems	75
3.7 BoxGrinder (Fedora, RHEL)	76
3.8 VMBuilder (Ubuntu)	82
4. Virtuelle Datenträger	89
4.1 Grundlagen	89
4.2 Image-Dateien und Speicherpools	96
4.3 Logical Volumes als virtuelle Datenträger	104
4.4 Festplatten und Partitionen	107

4.5	iSCSI-Anbindung	108
4.6	Snapshots	118
4.7	Direkter Zugriff auf virtuelle Datenträger	122
4.8	libguestfs-Werkzeuge	125
4.9	Virtuelle Datenträger nachträglich vergrößern	130
4.10	Benchmark-Tests	136
5.	Netzwerkconfiguration	147
5.1	Netzwerkanbindung des Gastes	147
5.2	Nutzung einer Bridge	149
5.3	Routing von Netzen	161
5.4	Verwaltung der Netze mit libvirt	164
5.5	OpenvSwitch	169
5.6	libvirt-Absicherung mit TLS und Zertifikaten	177
5.7	Fortgeschrittene Netzwerkfunktionen	182
6.	Grafik	197
6.1	Grafikadapter	197
6.2	VNC	199
6.3	Spice	202
6.4	SDL	207
6.5	Maus	208
6.6	Arbeiten in der Textkonsole	208
7.	CPU- und Speicherverwaltung	211
7.1	CPU-Eigenschaften	211
7.2	NUMA	214
7.3	Speicherverwaltung (RAM)	220
8.	Andere Hardware-Komponenten	223
8.1	USB-Passthrough	223
8.2	PCI-Passthrough	226
8.3	Audio	229

8.4	Uhrzeit	230
8.5	Ressourcensteuerung mit Cgroups	231
9.	Sicherheit	247
9.1	Secure Hypervisor sVirt	247
9.2	SELinux	250
9.3	AppArmor	253
9.4	Monitoring mit Nagios	260
9.5	Backups	262
9.6	Live-Migration	269
10.	Hochverfügbarkeit und Cloud	275
10.1	Clustering mit Ganeti	276
10.2	DRBD und Pacemaker	292
10.3	Cloud	303
11.	Kommandoreferenz	307
	Stichwortverzeichnis	335

Vorwort

Vor einigen Jahren war ein virtualisierter Server im Vergleich zu einem Server, der auf echter, physikalischer Hardware lief, die zweite Wahl. Das lag auch daran, dass viele Hosting-Unternehmer virtuelle Server mit unzureichender Hardware-Unterstützung anboten. Frustrierte Kunden, die über starke Leistungsschwankungen klagten, waren die Folge. Mittlerweile hat sich die Situation komplett verändert.



Virtuelle Maschinen bieten weit mehr Flexibilität als herkömmliche Installationen auf realer Hardware. Es ist bei Bedarf problemlos möglich, eine virtuelle Maschine mit zusätzlicher Hardware auszustatten (z. B. mit mehr RAM oder zusätzlichen CPU-Cores). Sollte ein Virtualisierungs-Host ausfallen, ist es vergleichsweise unkompliziert, die virtuellen Maschinen auf einem anderen Host auszuführen.

Gerade in großen Rechenzentren sind virtuelle Maschinen wesentlich besser zu administrieren als traditionelle Installationen (ein Rechner – ein Betriebssystem). Gleichzeitig ermöglicht die parallele Ausführung mehrerer virtueller Maschinen auf einem Rechner eine bessere Nutzung der vorhandenen Hardware.

Mit der Virtualisierung gehen oft auch Sicherheitsgewinne einher: Während früher versucht wurde, möglichst viele Funktionen auf einer Installation zusammenzufassen, ist es jetzt oft zweckmäßiger, einzelne Funktionen (z. B. den Web-Server, den Mail-Server, den Datenbank-Server) in getrennten virtuellen Maschinen auszuführen. Das minimiert die Nebenwirkungen und begrenzt bei Ausfällen oder einer Kompromittierung den Schaden.

Warum KVM?

Virtualisierungssysteme gibt es viele – was spricht also gerade für *Kernel-based Virtual Machines* oder kurz KVM?

- » KVM hat sich in den vergangenen Jahren zum populärsten Virtualisierungssystem für Linux entwickelt. Alle gängigen Enterprise- und Server-Distributionen setzen mittlerweile auf KVM als primäres Virtualisierungssystem.

- » Im Vergleich zu anderen Virtualisierungssystemen ist KVM besonders gut in den Linux-Kernel integriert. KVM war 2007 das erste Virtualisierungssystem, das in den offiziellen Kernel-Code aufgenommen wurde.
- » Schließlich ist KVM ein echtes Open-Source-Produkt: Sowohl der Hypervisor selbst als auch seine Administrationswerkzeuge sind kostenlos erhältlich und Bestandteil der meisten Linux-Distributionen.

KVM ist speziell für die Server-Virtualisierung optimiert, also für die Ausführung mehrerer Linux- oder Windows-Server-Installationen auf einem einzigen Rechner. Diese Einsatzvariante steht in diesem Buch im Vordergrund. KVM kann aber auch zur Desktop-Virtualisierung eingesetzt werden. Im Vergleich zu VirtualBox oder VMware Workstation ist KVM bzw. der Virtual Machine Manager aber weniger intuitiv zu bedienen.

Distributionen

Als Grundlage für dieses Buch dienten die folgenden Distributionen:

- » Red Hat Enterprise Linux 6.*n* sowie dazu kompatible Linux-Distributionen (z. B. Scientific Linux, CentOS und Oracle Linux)
- » Fedora
- » Ubuntu

Red Hat Enterprise Linux wird in diesem Buch generell mit RHEL abgekürzt. Wenn wir uns auf RHEL beziehen, schließt dies automatisch alle dazu kompatiblen Klone ein. Mit anderen Worten: »verfügbar ab RHEL 6.2« bedeutet, dass diese Funktion auch in CentOS 6.2, Scientific Linux 6.2 etc. genutzt werden kann.

Fedora ist für den Unternehmenseinsatz nicht zu empfehlen. Aus KVM-Sicht ist diese Distribution dennoch von großer Bedeutung: Neue KVM-Features sind fast immer zuerst in Fedora verfügbar, bevor sie in Enterprise-Distributionen aufgenommen werden.

Michael Kofler und Ralf Spenneberg (Mai 2012)

<http://kofler.info/> — <http://www.os-t.de/>

1. Grundlagen

Dieses Kapitel fasst die wichtigsten Grundlagen rund um Virtualisierung, KVM und libvirt zusammen. Sie erfahren hier, wie QEMU, KVM und libvirt zusammenhängen, welche Hardware-Komponenten wie durch die Virtualisierung abgebildet werden etc. Das Kapitel dient damit gleichzeitig als Glossar.

Falls Sie schon Virtualisierungserfahrung haben, können Sie dieses Kapitel überspringen und gleich im nächsten Kapitel weiterlesen: Dort führen wir Sie in den praktischen Umgang mit KVM ein und zeigen Ihnen, wie Sie Ihre ersten virtuellen KVM-Maschinen einrichten.



1.1 Virtualisierungsgrundlagen

Virtualisierung bedeutet, dass auf einem physikalischen Rechner mehrere Betriebssysteme parallel installiert und ausgeführt werden. Die Betriebssysteme laufen in sogenannten »virtuellen Maschinen«.

Bei der Beschreibung von Virtualisierungssystemen hat es sich eingebürgert, das Grundsystem als Wirt (Host) und die darauf laufenden virtuellen Maschinen als Gäste zu bezeichnen. Mitunter ist anstelle von Gästen auch von Domains die Rede.

Bei KVM kommt als Host-System ausschließlich Linux infrage. Wesentlich flexibler ist KVM bei den Gästen: Außer Linux können innerhalb der virtuellen KVM-Maschinen auch diverse Windows-, BSD- und Solaris-Varianten ausgeführt werden.

Virtualisierungstechniken und -programme

KVM ist ein Virtualisierungssystem mit Paravirtualisierung, Hardware-Unterstützung und einem Typ-2-Hypervisor. Damit Sie die Informationen verstehen und besser einordnen können, haben wir in der folgenden Liste die gängigsten Virtualisierungstechniken zusammengefasst und Beispiele für Vertreter der jeweiligen Gattung angegeben. Dabei

tauchen manche Virtualisierungssysteme mehrfach auf, weil darin je nach Hardware- und Treiberunterstützung unterschiedliche Techniken zum Einsatz kommen.

- » **Vollvirtualisierung (virtuelle Maschinen, Emulation):** Hier simuliert ein Programm virtuelle Hardware, also einen Rechner, der aus CPU, RAM, Festplatte, Netzwerkkarte etc. besteht. Für die Gastsysteme sieht es so aus, als würde die virtuelle Hardware real existieren. Damit das funktioniert, muss das Virtualisierungsprogramm des Wirts den Code des Gasts überwachen und bestimmte Anweisungen durch anderen Code ersetzen. Diese Aufgabe übernimmt der sogenannte Hypervisor (auch *Virtual Machines Monitor* oder kurz VMM). Der Hypervisor ist auch für die Speicher- und Prozessverwaltung und andere hardware-nahe Funktionen verantwortlich.

Vorteile: Nahezu jedes Betriebssystem kann innerhalb der virtuellen Maschine ausgeführt werden. Das Betriebssystem muss dazu nicht verändert werden.

Nachteile: Relativ langsam.

Beispiele: VMware Workstation, QEMU, VirtualBox, Microsoft Virtual PC

- » **Paravirtualisierung:** Auch hier stellt der Wirt virtuelle Maschinen zur Verfügung, in denen die Gäste laufen. Der Unterschied zur Vollvirtualisierung besteht darin, dass das Gastbetriebssystem für die Virtualisierung modifiziert sein muss und dank geeigneter Treiber direkt mit dem Hypervisor kommunizieren kann.

Vorteile: Effizient.

Nachteile: Erfordert speziell für das Virtualisierungssystem modifizierte Betriebssysteme bzw. Treiber.

Beispiele: Xen, UML (User-mode Linux), teilweise VirtualBox (virtio-net)

- » **(Para-)Virtualisierung mit Hardware-Unterstützung:** Moderne CPUs von Intel und AMD enthalten Funktionen zur Vereinfachung von Virtualisierungstechniken. Intel nennt diese Technik *Intel-VT* (ehemals Vanderpool), AMD taufte seine Funktionen *AMD-V* (ehemals Pacifica).

Vorteile: Effizient, je nach Implementierung ist keine Modifikation im Gastbetriebssystem erforderlich.

Nachteile: Erfordert spezielle Prozessoren.

Beispiele: VMware ESX, Hyper-V, KVM, Xen

- » **Virtualisierung auf Betriebssystemebene (Containers):** Dieses Verfahren verzichtet auf richtige virtuelle Maschinen. Die Gastsysteme nutzen vielmehr den gemeinsamen Kernel und Teile des Dateisystems des Wirts. Zu den wichtigsten Aufgaben des Virtualisierungssystems zählt es, den Wirt von seinen Gästen zu isolieren, um jede Art von Sicherheitsrisiken zu vermeiden.

Vorteile: Sehr effizient, spart Ressourcen (RAM, Festplatte etc.).

Nachteile: Nur geeignet, wenn der Wirt und seine Gäste jeweils exakt dasselbe Betriebssystem bzw. exakt dieselbe Kernelversion nutzen. Das Betriebssystem muss entsprechend modifiziert werden.

Beispiele: OpenVZ, Virtuozzo, Linux-VServer

Um Linux in einer virtuellen Maschine (also als Gast) auszuführen, sind bei allen Verfahren außer dem ersten in den Kernel integrierte Treiber bzw. Erweiterungen erforderlich. Der entsprechende Code ist für die Virtualisierungssysteme KVM, Xen, Hyper-V und UML offizieller Bestandteil des Kernels. (Die Hyper-V-Funktionen für den Linux-Kernel hat Microsoft sogar als Open-Source-Quellcode bereitgestellt, was durchaus bemerkenswert ist!) Bei anderen Virtualisierungssystemen müssen im Gastsystem hingegen zusätzliche Treiber bzw. Kernelmodule kompiliert und eingerichtet werden, was umständlich ist und bei Kernel-Updates häufig zu Problemen führt.

Weitere Informationen zu verschiedenen Virtualisierungstechniken finden Sie in der Wikipedia sowie auf den folgenden Seiten:

<http://virt.kernelnewbies.org/TechOverview>

http://wiki.openvz.org/Introduction_to_virtualization

Hypervisor-Typen

Je nach Virtualisierungssystem gibt es zwei grundsätzliche Möglichkeiten, den Hypervisor zu implementieren (also das Programm, das die Ausführung der virtuellen Maschinen überwacht):

- » **Typ 1 (Native bzw. Bare Metal):** In diesem Fall läuft der Hypervisor direkt auf dem physikalischen Rechner, ohne die Hilfe eines Betriebssystems in Anspruch zu nehmen. Populäre Bare-Metal-Hypervisors sind Microsoft Hyper-V, VMware ESXi oder Xen. Die Administration des Hypervisors erfolgt in der Regel durch einen privilegierten Gast (z. B. durch eine Linux-Installation in der sogenannten Domain 0 bei Xen oder durch eine Windows-Server-Installation innerhalb von Hyper-V).
- » **Typ 2 (Hosted):** Bei dieser Variante wird der Hypervisor in einem Betriebssystem ausgeführt. Das Virtualisierungssystem kann damit alle Betriebssystemfunktionen nutzen und so wesentlich schlanker ausgeführt werden. Zu den Typ-2-Hypervisors zählen neben KVM auch VirtualBox, VMware Workstation und Server sowie Microsoft Virtual PC.

KVM versus Xen

Vor wenigen Jahren war Xen der Star in der Virtualisierungsszene. Sowohl Red Hat als auch SUSE (damals Novell) setzten in ihren Enterprise-Distributionen voll auf Xen, und selbst Microsoft bemühte sich um Kompatibilität zu Xen. Infolge der Übernahme durch Citrix hat Xen in der Open-Source-Szene stark an Popularität eingebüßt. Über mehrere Jahre hakte es zudem bei der Integration des aktuellen Xen-Codes in den Kernel. Zumindest dieses Problem scheint seit Kernelversion 3.0 gelöst.

Aus technischer Sicht gibt es einerseits fundamentale Unterschiede zwischen KVM und Xen, andererseits aber auch große Gemeinsamkeiten bei der Administration. Xen ist im Gegensatz zu KVM ein Typ-1-Hypervisor. Was in KVM das Hostsystem ist, läuft unter Xen als privilegiertes Gastsystem und wird als *Domäne 0*, kurz *Dom0*, bezeichnet. Alle weiteren virtuellen Maschinen werden *unprivilegierte Domänen*, kurz *DomU*, genannt. Zur Verwaltung der virtuellen Maschinen können wie bei KVM die libvirt-Werkzeuge verwendet werden.

Auch wenn Virtualisierung unter Linux ganz klar in Richtung KVM geht, bietet Xen bei manchen Virtualisierungsaspekten mehr Performance als KVM und ist nach wie vor sehr weit verbreitet. Viele Cloud-Systeme basieren auf Xen. Eine gute Zusammenfassung der Unterschiede zwischen Xen und KVM bietet dieser Artikel:

<http://www.windowspro.de/andrej-radonic/kvm-und-xen-opensource-virtualisierung-im-vergleich>

Red Hat Enterprise Linux unterstützt Xen ab Version 6 allerdings nur noch als Gastsystem, nicht aber als Hostsystem. Wenn Sie virtuelle Maschinen von Xen zu KVM migrieren möchten, finden Sie hier weitere Informationen:

http://fedoraproject.org/wiki/Features/Xen_to_KVM_migration

http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Virtualization_for_Servers/2.2/html/Administration_Guide/virt-v2v-scripts.html

http://michael.stapelberg.de/Artikel/xen_to_kvm

<http://www.gloudemans.info/migrate-paravirtualized-xen-to-kvm-under-rhel/>

1.2 QEMU, KVM und libvirt

KVM (*Kernel-based Virtual Machine*) ist genau genommen nur eine Kernelerweiterung. Erst in Kombination mit diversen anderen Komponenten wird aus KVM ein richtiges Virtualisierungssystem. In diesem Abschnitt stellen wir Ihnen diese Komponenten näher vor. Vorweg fasst die folgende Liste die wichtigsten Eigenschaften von KVM zusammen:

- » Typ-2-Hypervisor (Hosted)
- » Hardware-Virtualisierung (setzt AMD-V bzw. Intel-VT voraus)
- » Paravirtualisierung, sofern im Gastsystem virtio-Treiber zur Verfügung stehen
- » Verschachtelte Virtualisierung
- » Unterstützte Host-Systeme: nur Linux
- » Unterstützte Gastssysteme: Linux, Windows, BSD, Solaris etc.

KVM unterstützte ursprünglich nur Linux-Versionen mit x86- bzw. x86-64-CPU's. Mittlerweile gibt es Portierungen für S/390, Power-PC und IA-64. Eine weitere Portierung für ARM-CPU's ist in Arbeit. In diesem Buch setzen wir allerdings voraus, dass Sie einen Rechner mit einer x86-64-CPU verwenden.

QEMU

Die Basis für KVM ist das Emulationsprogramm QEMU (Kommandoname `qemu`). Es emuliert verschiedene CPU's und elementare Hardware-Komponenten eines typischen Rechners (Netzwerkkarte, CD-Laufwerk etc.). Die größte Stärke von QEMU besteht darin, dass das Programm auch fremde CPU's emulieren kann. Sie können also auf einem Rechner mit einer Intel- oder AMD-CPU ein Programm testen, das für eine ARM- oder Sparc-CPU kompiliert wurde. Darunter leidet aber naturgemäß die Geschwindigkeit. Eine ausführlichere Beschreibung der Funktionen und Grenzen von QEMU finden Sie hier:

<http://wiki.qemu.org/>

Im Prinzip kann QEMU alleine dazu verwendet werden, um ein anderes Betriebssystem unter Linux auszuführen. Das Problem besteht aber darin, dass die resultierende Geschwindigkeit vergleichsweise gering ist.

KVM

KVM ist ein relativ kleines Kernelmodul, das seine Wirkung erst in Kombination mit der Emulationssoftware QEMU entfaltet. KVM setzt eine CPU mit Funktionen zur Hardware-Virtualisierung voraus und macht aus dem Emulator QEMU ein Hardware-Virtualisierungssystem.

KVM wurde ursprünglich von der Firma Qumranet entwickelt. Red Hat kaufte Qumranet 2008 und entwickelt KVM seither weiter. KVM ist seit Version 2.6.20 integraler Bestandteil des Linux-Kernels. Wie der gesamte Kernel kommt auch für die KVM-Funktionen die Open-Source-Lizenz GPL zur Anwendung.

KVM ist ein Typ-2-Hypervisor. Grundlegende Funktionen wie die Speicherverwaltung, Prozessverwaltung etc. sind daher nicht direkt im Virtualisierungssystem implementiert. Vielmehr greift KVM auf die im Linux-Kernel vorhandenen Funktionen zurück. Das macht KVM zu einem besonders schlanken Virtualisierungssystem, das nahtlos auf den über viele Jahre optimierten Linux-Kernel aufsetzt. In vielerlei Hinsicht verhält sich eine KVM-Maschine also ganz ähnlich wie ein gewöhnlicher Linux-Prozess.

KVM führt standardmäßig eine vollständige Virtualisierung durch, d. h., die Gäste brauchen keine KVM-spezifischen Treiber. KVM unterstützt aber auch einige paravirtualisierte Treiber, insbesondere den virtio-Blocktreiber für virtuelle Festplattenzugriffe, den virtio-Netzwerktreiber sowie QXL-Grafiktreiber zur effizienten Darstellung des Grafiksystems im Zusammenspiel mit Spice. Die Verwendung dieser Treiber ist optional, aber aus Geschwindigkeitsgründen empfehlenswert.

Die Nutzung der KVM-Funktionen erfolgt über die Device-Datei `/dev/kvm`. Grundsätzlich können Sie das KVM-Kommando ohne root-Rechte ausführen, Ihre Benutzerrechte müssen aber ausreichen, um die Device-Datei `/dev/kvm` zu lesen und zu verändern. Bei manchen Distributionen (z. B. Ubuntu) müssen Sie dazu Mitglied der Gruppe `kvm` sein.

Auch wenn in diesem Buch nur von KVM die Rede ist – in Wirklichkeit ist die Kombination von QEMU und KVM gemeint. Der hohe Grad der Abhängigkeit von KVM und QEMU drückt sich bei manchen Distributionen auch im Paketnamen `qemu-kvm` ab. Damit werden sowohl QEMU als auch die darauf aufsetzenden KVM-Erweiterungen installiert. (Weiterer KVM-Code befindet sich im Kernel. Diese KVM-Teile müssen nicht extra installiert werden, sondern stehen bei allen aktuellen Distributionen standardmäßig zur Verfügung.)

QEMU ist also trotz KVM keineswegs obsolet. Abgesehen davon, dass es als Fundament für KVM dient, bietet es noch zwei Vorteile gegenüber KVM:

- » QEMU funktioniert im Gegensatz zu KVM selbst dann, wenn die CPU keine Virtualisierungsfunktionen zur Verfügung stellt.
- » QEMU ist in der Lage, andere Prozessoren als jenen des Wirtssystems zu emulieren.

Native Linux KVM Tool

Einige Kernel-Entwickler möchten QEMU durch das schlankere *Native Linux KVM Tool* ersetzen (Kommando `1kvm`). Damit ist es möglich, virtuelle KVM-Maschinen ohne die Hilfe von QEMU auszuführen. Der Grund für diese Initiative besteht darin, dass QEMU eine Menge Code enthält, der für KVM nicht relevant ist. Zudem ist die Steuerung von QEMU durch unzählige, oft inkonsistent benannte Optionen ausgesprochen mühsam.

Zurzeit (im Frühjahr 2012) bietet das Native Linux KVM Tool allerdings nur einen Bruchteil der Funktionen von QEMU. Es ist unklar, ob bzw. wann das Native Linux KVM Tool in den Kernel-Code integriert wird. In diesem Buch gehen wir auf jeden Fall davon aus, dass Sie KVM in Kombination mit QEMU einsetzen.

Weitere Informationen können Sie hier nachlesen:

<http://lwn.net/Articles/438182/>

<http://lwn.net/Articles/447556/>

<https://github.com/penberg/linux-kvm/tree/master/tools/kvm#readme>

libvirt

libvirt ist eine Schnittstelle (Application Programming Interface, kurz API) zur Verwaltung von virtuellen Maschinen und der dazugehörigen virtuellen Netzwerk- und Festplatten-Devices. Eine Voraussetzung für die Nutzung der libvirt-Werkzeuge besteht darin, dass auf dem Hostsystem der Dämon libvirtd läuft. Dieses Programm wird beim Hochfahren des Host-Rechners durch das Init-System gestartet.

Die Steuerung der virtuellen Maschinen kann nun wahlweise durch die Shell virsh, den Virtual Machine Manager oder durch andere libvirt-Kommandos erfolgen. Sie können damit neue virtuelle Maschinen erzeugen, kopieren (klonen) etc. Mit den libvirt-Werkzeugen gestartete virtuelle Maschinen laufen weiter, wenn Sie sich ab- und neu anmelden.

Auch wenn wir uns in diesem Buch ausschließlich auf KVM beziehen, können die libvirt-Werkzeuge auch zur Steuerung anderer Virtualisierungssysteme verwendet werden (z. B. Xen). Weil die libvirt-Werkzeuge versuchen, die Verwaltung virtueller Maschinen möglichst unabhängig vom Virtualisierungssystem zu machen, passen manche Begriffe der libvirt-Nomenklatur schlecht zu KVM. Beispielsweise werden virtuelle Maschinen in der libvirt-Welt generell als Domänen bezeichnet, was eigentlich nur unter Xen üblich ist.

TIPP

Sehr oft, wenn im Internet oder in anderen Dokumenten von KVM die Rede ist, werden in Wirklichkeit libvirt-Werkzeuge zur Verwaltung von virtuellen KVM-Maschinen beschrieben. Natürlich ist es richtig, dass Sie alles, was libvirt & Co. bewerkstelligen, ebenso gut auch durch die direkte Steuerung von KVM erledigen können. Die libvirt-Werkzeuge vereinfachen aber viele grundlegende Aufgaben und machen den effizienten Betrieb mehrerer virtueller Maschinen erst möglich.

Benutzer- versus Systemebene

KVM-Maschinen können via libvirt auf zwei verschiedenen Ebenen ausgeführt werden:

- » **Benutzerebene** (`qemu:///session`): Diese Variante ist vor allem für die Desktop-Virtualisierung gedacht und gibt den virtuellen Maschinen weniger Zugriffsmöglichkeiten auf die Hardware des Hostrechners. Intern wird beim ersten Aufruf eines libvirt-Werkzeugs auf Benutzerebene ein eigener libvirtd-Prozess gestartet, dem nur die Rechte des aktuellen Benutzers zukommen. KVM-Maschinen auf Benutzerebene minimieren also die Sicherheitsrisiken durch die Virtualisierung.
- » **Systemebene** (`qemu:///system`): Virtuelle Maschinen auf Systemebene sind besser für die Server-Virtualisierung geeignet, weil sie direkt auf Hardware-Komponenten des Hostrechners zugreifen können. Die libvirt-Prozesse kommunizieren dabei mit dem Dämon libvirtd, der mit root-Rechten läuft. Zur Server-Virtualisierung müssen Sie zumeist auf der Systemebene (root-Ebene) arbeiten.

Bei der Kommunikation zwischen den libvirt-Werkzeugen und dem Dämon libvirtd bestehen starke Konfigurationsunterschiede zwischen den Distributionen. Ganz einfach ist es bei **Fedora und RHEL**: Wenn Sie mit libvirtd auf Systemebene kommunizieren möchten, benötigen Sie root-Rechte. Der Virtual Machine Manager kann zwar mit Benutzerrechten gestartet werden, das Programm erwartet aber unmittelbar nach dem Start die Angabe des root-Passworts.

Beachten Sie dabei, dass zwar die libvirt-Werkzeuge mit root-Rechten ausgeführt werden, nicht aber das eigentliche Virtualisierungskommando! Vielmehr starten die libvirt-Werkzeuge das Kommando `qemu-kvm` unter dem Benutzeraccount `qemu`. Auf diese Feinheit müssen Sie vor allem bei der richtigen Einstellung der Zugriffsrechte für Image- oder ISO-Dateien achten!

Auch unter **Ubuntu** kommunizieren libvirt-Werkzeuge, die mit root-Rechten ausgeführt werden, mit libvirtd auf Systemebene. Aber auch libvirt-Kommandos, die nur mit Benutzerrechten ausgeführt werden, dürfen mit libvirtd auf Systemebene kommunizieren, sofern der Benutzer der Gruppe libvirtd angehört! (Genau genommen ist entscheidend, ob der Benutzer auf die Datei `/var/run/libvirt/libvirt-sock` zugreifen darf. Diese Datei gehört root und der Gruppe libvirtd.)

Die Zuordnung zur Gruppe libvirtd wird bei der Installation des Pakets libvirt-bin automatisch für den Benutzer hergestellt, der die Installation durchführt. Weitere Benutzer können mit dem folgenden Kommando der libvirtd-Gruppe hinzugefügt werden:

```
root# usermod -a -G libvirtd benutzername (Ubuntu)
```

Ein verwirrender Aspekt bei der Verwendung der libvirt-Werkzeuge mit Benutzerrechten und libvirtd-Gruppenzugehörigkeit unter Ubuntu besteht darin, dass `virsh` und `virt-manager` standardmäßig auf der Systemebene arbeiten, einzelne Kommandos wie `virt-install` aber auf der Benutzerebene:

```
user$ virsh ...      (kommuniziert standardmäßig auf Systemebene)
user$ virt-manager ... (kommuniziert standardmäßig auf Systemebene)
user$ virt-install ... (erzeugt virtuelle Maschinen auf Benutzerebene)
```

Wenn Sie `virt-install` auf Systemebene ausführen möchten, müssen Sie es mit `sudo` bzw. als `root` ausführen.

HINWEIS

Im weiteren Verlauf dieses Buchs werden wir nur noch fallweise zwischen Benutzer- und Systemebene differenzieren! Wir gehen davon aus, dass Sie libvirt-Programme mit `root`-Rechten ausführen bzw. mit `libvirtd` auf Systemebene kommunizieren. Für die Server-Virtualisierung ist dies im Regelfall unumgänglich.

libvirt-Werkzeuge

In diesem Buch werden Sie diverse libvirt-Kommandos und -Benutzeroberflächen im Detail kennenlernen. Die folgende Liste gibt vorweg einen ersten Überblick:

- » `virsh`: Mit der Ausführung des Kommandos `virsh` gelangen Sie in eine Shell. Dort können Sie dann eine Vielzahl von Befehlen ausführen, um damit virtuelle Maschinen zu starten, zu stoppen etc.
- » Virtual Machine Manager: Hierbei handelt es sich um eine grafische Benutzeroberfläche, die ähnliche Funktionen wie die `virsh` bietet.
- » `virt-viewer` ist ein VNC-Client zur Darstellung des Bildschirminhalts sowie zur Kommunikation mit einer virtuellen Maschine. Statt `virt-viewer` können Sie auch jeden beliebigen VNC-Client einsetzen. In diesem Fall müssen Sie vorweg mit dem `virsh`-Kommando `vncdisplay` die VNC-Verbindungsdaten ermitteln.
- » `virt-install` richtet eine neue virtuelle Maschine ein. Unter Ubuntu oder Debian können Sie neue virtuelle Maschinen noch effizienter mit dem `vm-builder` erzeugen.
- » `virt-clone` dupliziert eine vorhandene virtuelle Maschine und gibt dem Netzwerkadapter eine neue, zufällige MAC-Adresse. Die zugrunde liegende virtuelle Maschine muss vor dem Klonen heruntergefahren werden.

libvirt im Netzwerk

Die libvirt-Werkzeuge können nicht nur auf dem lokalen Rechner genutzt werden, sondern auch im Netzwerk. So können Sie beispielsweise mit der Shell `virsh` die virtuellen Maschinen eines anderen KVM-Hosts steuern oder mit dem Virtual Machine Manager externe KVM-Hosts verwalten. Grundsätzlich stehen die folgenden Kommunikationsmechanismen zur Auswahl:

- » unverschlüsselt über das TCP/IP-Protokoll (unsicher!)
- » getunnelt via SSH
- » verschlüsselt mit SASL/Kerberos
- » verschlüsselt mit TLS/x509

Die Datei `/etc/libvirt/libvirtd.conf` steuert, welche Verfahren zulässig sind und wo sich gegebenenfalls die Schlüsseldateien befinden. Aus Sicherheitsgründen scheidet die erste Variante aus. Die Varianten drei und vier erfordern einen relativ hohen Konfigurationsaufwand. Insofern ist SSH für eine einfache Installation die beste Wahl. Die einzige Voraussetzung besteht darin, dass auf dem KVM-Host auch ein SSH-Server läuft. Aus Sicherheitsgründen ist es empfehlenswert, die SSH-Authentifizierung nicht per Passwort, sondern über Schlüsseldateien durchzuführen.

Detaillierte Anleitungen, wie Sie TLS für mehrere libvirt-Maschinen einrichten, finden Sie in Abschnitt 5.6.

1.3 Virtuelle Hardware-Komponenten

Dieser Abschnitt gibt einen Überblick, aus welchen Hardware-Komponenten eine virtuelle Maschine auf der Basis von KVM/QEMU zusammengesetzt ist:

- » **CPU:** Virtuelle Maschinen sehen im Wesentlichen dieselbe CPU wie das Hostsystem. Allerdings stehen den Gästen nicht alle Eigenschaften bzw. Spezialfunktionen der CPU zur Verfügung. Außerdem wird im Regelfall die Anzahl der CPUs und Cores limitiert, die ein Gast nutzen darf.
- » **Arbeitsspeicher:** Der gesamte Arbeitsspeicher des Hostrechners muss auf das Hostbetriebssystem und die laufenden Gäste verteilt werden. Normalerweise wird der für eine virtuelle Maschine beanspruchte Speicher dem Hostsystem sofort zur Gänze abgezwickelt. Es gibt aber Möglichkeiten, die Speicherzuweisung dynamisch durchzuführen (Ballooning, erfordert im Gast den `virtio-balloon`-Treiber) sowie Speicherseiten, die in mehreren virtuellen Maschinen denselben Inhalt haben, zu teilen (Kernel Samepage Merging).

- » **Datenträger/Festplatte:** Der Gast sieht seine virtuelle Festplatte wahlweise als IDE- oder SCSI-Festplatte. Im Hostsystem wird diese Festplatte in der Regel durch eine Image-Datei oder durch ein Logical Volume abgebildet. Aber auch Netzwerkverzeichnisse oder iSCSI-Geräte können als Datenspeicher für die virtuellen Maschinen konfiguriert werden.
- » **CD/DVD-Laufwerk:** Das CD- oder DVD-Laufwerk des Hostrechners werden Sie in der Regel nur während der Installation einer virtuellen Maschine nutzen. Noch praktischer und effizienter ist es, stattdessen direkt die entsprechende ISO-Datei als virtuelles CD/DVD-Laufwerk zu verwenden. KVM-Gäste können CDs/DVDs übrigens nur lesen, aber nicht schreiben (brennen).
- » **Netzwerkadapter:** Beim Einrichten einer virtuellen Maschinen können Sie sich zwischen mehreren virtuellen Netzwerkadaptern entscheiden. Für den Gast sieht es dann so aus, als würde ihm beispielsweise ein RTL-8139- oder ein Intel-E1000-Netzwerkadapter zur Verfügung stehen. Wesentlich effizienter als die Emulation eines echten Netzwerkadapters ist die Verwendung des virtio-Treibers (siehe unten).

Neben der Auswahl des Netzwerkadapters müssen Sie als KVM-Administrator auch festlegen, wie die virtuelle Maschine mit dem Netzwerk des Hostrechners verbunden ist: z. B. über ein privates Netzwerk per NAT oder über eine Netzwerkbrücke mit dem lokalen Netzwerk des Hostrechners.
- » **Grafiksystem:** Zumindest während der Installation müssen Sie die Ausgaben der virtuellen Maschine sehen. Der Gast braucht also ein eigenes Grafiksystem. Dazu wird eine VGA-kompatible Grafikkarte emuliert, deren Ausgaben dann via VNC, SDL (Simple DirectMedia Library) oder Spice in einem Fenster angezeigt werden. Für den 2D-Einsatz funktioniert dies selbst in hoher Auflösung gut. KVM bietet zurzeit aber keine Unterstützung für 3D-Funktionen.
- » **Audio:** Wenn KVM zur Desktop-Virtualisierung verwendet wird, soll die virtuelle Maschine zumeist auch mit einer virtuellen Sound-Karte ausgestattet werden. In der Praxis bereitet das aber oft Probleme. Auch hier zeigt sich, dass KVM primär für den Server-Einsatz optimiert ist.
- » **Tastatur:** Wenn die virtuelle Maschine via VNC oder Spice gesteuert wird, werden die internen Tastaturcodes direkt an die virtuelle Maschine weitergeleitet. Deswegen ist auch innerhalb der virtuellen Maschine eine korrekte Konfiguration der Tastatur erforderlich (also z. B. für die deutsche Tastaturbelegung).
- » **Maus:** Zur Bedienung einer grafischen Benutzeroberfläche im Gast ist eine Maus erforderlich. Damit Mausbewegungen an den Gast weitergeleitet werden, wird innerhalb der virtuellen Maschine eine PS/2-Maus oder ein USB-Tablet-Gerät emuliert.

- » **Serielle Schnittstelle, Konsole:** Zur Steuerung der virtuellen Maschine im Textmodus (in einer Konsole) besteht die Möglichkeit, Ein- und Ausgaben über eine virtuelle serielle Schnittstelle (ein `pty-Device`) zu leiten.
- » **USB-Geräte:** Damit eine virtuelle Maschine mit einem USB-Gerät kommunizieren kann, muss dieses direkt an den Gast weitergeleitet werden (*pass-through*). Jedes USB-Gerät kann immer nur von einem Gast genutzt werden. Auch das Hostsystem muss die Kontrolle über das USB-Gerät abgeben.
- » **PCI-Geräte:** Die Nutzung von PCI-Geräten durch Gäste ist in der Regel nur indirekt möglich: Beispielsweise kann ein Gast über eine virtuelle Netzwerkkarte von QEMU oder mittels des `virtio-net`-Treibers (siehe unten) auf einen PCI-Netzwerkadapter des Hostrechners zugreifen.

Es gibt allerdings Sonderfälle: So ermöglicht SR-IOV (*Single Root I/O Virtualization*) die gemeinsame direkte Nutzung bestimmter Netzwerkkarten durch Gäste. Sofern die CPU und das Mainboard es unterstützen (Intel VT-D bzw. AMD IOMMU), kann ein PCI-Gerät auch dezidiert an einen Gast weitergereicht werden.

- » **Uhr:** Normalerweise ist erwünscht, dass die Uhrzeit des Hostsystems und seiner Gäste synchron ist. Das klingt nach einem trivialen Problem – es ist aber nicht trivial, weil die Taktfrequenz der Host-CPU variiert, das Gastsystem davon aber nicht informiert wird. Bei Linux-Gästen ab Kernelversion 2.6.27 kümmert sich der in den Kernel integrierte `kvm-clock`-Treiber um die Synchronisation; bei älteren Linux-Distributionen sowie bei Windows-Gästen ist aber vielfach eine Konfiguration des Timing-Verfahrens erforderlich.

Bei der Konfiguration der virtuellen Hardware gibt es unzählige Möglichkeiten und Varianten, auf die wir im weiteren Verlauf dieses Buchs noch detailliert eingehen (siehe die Kapitel 4 bis 8).

virtio-Treiber

Für den effizienten Hardware-Zugriff spielen die `virtio`-Treiber eine große Rolle. Sie ermöglichen es Linux- und zum Teil auch Windows-Gästen, direkt mit dem KVM-Hypervisor zu kommunizieren (Paravirtualisierung). Unter Linux stehen diese Treiber standardmäßig zur Verfügung (es sei denn, Sie virtualisieren Distributionen mit uralten Kernel-Versionen vor 2.6.25). Auch für Windows gibt es `virtio`-Treiber, diese müssen aber extra installiert werden (siehe Seite 47).

Die folgende Liste gibt eine Kurzbeschreibung der wichtigsten virtio-Treiber für Linux:

- » virtio-net: Zugriff auf die Netzwerkfunktionen des Hostsystems
- » virtio-blk: Zugriff auf virtuelle Festplatten
- » virtio-pci: Zugriff auf PCI-Geräte
- » virtio-balloon: dynamische Speicherzuordnung

QXL-Treiber und Spice

Ein Sonderfall ist der für Linux und Windows verfügbare QXL-Treiber, der in Kombination mit Spice (*Simple Protocol for Independent Computing Environments*) eine besonders effiziente Emulation des Grafiksystems ermöglicht (siehe Kapitel 6). Anders als Xen ermöglichen KVM und QEMU momentan allerdings keinen direkten Zugriff auf die Grafikkarte.

2. Hello World!

Genug der grauen Theorie! In diesem Kapitel lernen Sie den Virtual Machine Manager kennen und richten damit Ihre ersten virtuellen Maschinen ein, wobei hier sowohl Linux- als auch Windows-Gäste behandelt werden. Dabei werden auch einige Sonderfälle erörtert, etwa die Nutzung des Virtual Machine Managers im Netzwerk oder die Installation einer virtuellen Maschine auf einem Root Server.

Wenn Sie Ihre virtuellen Maschinen lieber im Terminal administrieren, finden Sie im nächsten Kapitel eine detaillierte Beschreibung aller KVM- und libvirt-Kommandos.



2.1 Voraussetzungen

Damit Sie KVM einsetzen können, müssen zwei Voraussetzungen erfüllt sein: Zum einen muss die CPU Ihres Rechners Virtualisierungsfunktionen anbieten, zum anderen müssen Sie KVM, die libvirt-Werkzeuge und den Virtual Machine Manager installieren.

CPU-Unterstützung

Um festzustellen, ob Ihre CPU die für KVM notwendigen Virtualisierungsfunktionen anbietet (also Intel-VT oder AMD-V), führen Sie das folgende egrep-Kommando aus:

```
user$ egrep '^flags.*(vmx|svm)' /proc/cpuinfo
flags : fpu vme de pse tsc msr pae mce cx8 apic mtrr pge mca cmov pat pse36
      clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx rdtscp lm
      constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc
      aperfmperf pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 cx16
      xtpr pdcm sse4_1 sse4_2 popcnt aes xsave avx lahf_lm ida arat epb xsaveopt
      pln pts dts tpr_shadow vnmi flexpriority ept vpid
```

...

Das hier gezeigte Ergebnis stammt von einer Intel-i5-CPU. Wenn das `egrep`-Kommando keine Ausgabe liefert, unterstützt Ihre CPU keine Virtualisierung, oder die Funktion wurde im BIOS bzw. EFI deaktiviert.

HINWEIS

RHEL 6 unterstützt KVM nur, wenn auf dem Hostrechner eine 64-Bit-Version dieser Distribution installiert ist. Generell gehen wir in diesem Buch davon aus, dass Sie auf dem Hostrechner eine 64-Bit-Version von Linux installiert haben.

KVM- und libvirt-Pakete auf RHEL- und Fedora-Systemen installieren

Um virtuelle KVM-Maschinen auszuführen und mit den libvirt-Werkzeugen steuern zu können, müssen Sie die folgenden Pakete installieren:

```
root# yum install qemu-kvm libvirt python-virtinst
```

Der für die libvirt-Werkzeuge erforderliche Dämon `libvirtd` wird erst beim nächsten Rechnerneustart automatisch ausgeführt. Um den Dienst sofort zu starten, führen Sie das folgende Kommando aus:

```
root# service libvirtd start
```

Nur wenn auf Ihrem Hostsystem eine grafische Benutzeroberfläche installiert ist, ist es zweckmäßig, auch die grafischen Administrationswerkzeuge zu installieren (insbesondere also den Virtual Machine Manager). Bei einem extern laufenden (Root-)Server ist das folgende Kommando hingegen nicht zweckmäßig. Vielmehr installieren Sie den Virtual Machine Manager in diesem Fall auf Ihrem lokalen Rechner, von dem aus Sie Ihren Server administrieren.

```
user$ yum install virt-viewer virt-manager
```

KVM-Pakete unter Ubuntu installieren

Unter Ubuntu führen Sie zur Installation der KVM- und libvirt-Basispakete das folgende Kommando aus:

```
root# apt-get install qemu-kvm libvirt-bin virtinst
```

`libvirtd` wird damit automatisch gestartet. Mit dem Kommando `kvm-ok` aus dem Paket `cpu-checker` prüfen Sie, ob alle Voraussetzungen zur Nutzung von KVM erfüllt sind:

```
user$ kvm-ok
INFO: Your CPU supports KVM extensions
INFO: /dev/kvm exists
KVM acceleration can be used
```

Nur wenn auf Ihrem Hostsystem eine grafische Benutzeroberfläche installiert ist, ist es zweckmäßig, auch die grafischen Administrationswerkzeuge zu installieren:

```
root# apt-get install virt-viewer virt-manager
```

Die libvirt-Werkzeuge können nur Benutzer verwenden, die der Gruppe `libvirtd` angehören. In vergangenen Ubuntu-Versionen wurde der aktuelle Benutzer bei der Installation automatisch dieser Gruppe hinzugefügt. In Ubuntu 12.04 ist das aber nicht der Fall. Abhilfe schafft das folgende Kommando. Es muss für alle Benutzer ausgeführt werden, die den Virtual Machine Manager oder andere libvirt-Werkzeuge einsetzen dürfen:

```
root# usermod -a -G libvirtd benutzername
```

Die neue Gruppenzuordnung wird erst nach einem neuerlichen Login wirksam!

2.2 Der Virtual Machine Manager

Der Virtual Machine Manager (Programm- und Paketname `virt-manager`) ist eine grafische Benutzeroberfläche, die beim Erzeugen und Ausführen virtueller Maschinen hilft. Die Bedienung des Virtual Machine Managers ist allerdings nicht in allen Punkten intuitiv und kann nicht mit den Benutzeroberflächen von VMware oder VirtualBox mithalten.

Unter RHEL starten Sie den Virtual Machine Manager mit `ANWENDUNGEN|SYSTEMWERKZEUGE|VIRTUAL MACHINE MANAGER`. Damit Sie das Programm benutzen können, müssen Sie eine Verbindung zum libvirt-Dämon herstellen. Dazu reicht ein Doppelklick auf den bereits vorgesehenen Eintrag `LOCALHOST (QEMU)`. Der Verbindungsaufbau erfordert die Eingabe des `root`-Passworts. Ein weiterer Doppelklick auf den Verbindungsnamen führt in einen Dialog, der die Eckdaten der Verbindung zusammenfasst (siehe Abbildung 2.1).

Unter Ubuntu ist beim Start des Virtual Machine Managers keine Passwortangabe erforderlich. Aktuelle Ubuntu-Versionen stellen wie unter RHEL und Fedora eine Verbindung auf Systemebene her. Das setzt voraus, dass der Benutzer zur Gruppe `libvirtd` gehört. (Bei älteren Ubuntu-Versionen erfolgte die Verbindung zu einer lokalen Instanz von `libvirtd` auf Benutzerebene. Eine Verbindung auf Systemebene konnte aber problemlos zur Liste der Verbindungen hinzugefügt werden.)

Wenn der lokale Rechner nur zur Administration von virtuellen Maschinen auf anderen Rechnern dient, kann der Virtual Machine Manager auch eine Netzwerkverbindung zu einer `libvirtd`-Instanz eines anderen Rechners herstellen. Details zur Verwendung des Virtual Machine Managers im Netzwerk folgen auf Seite 31.

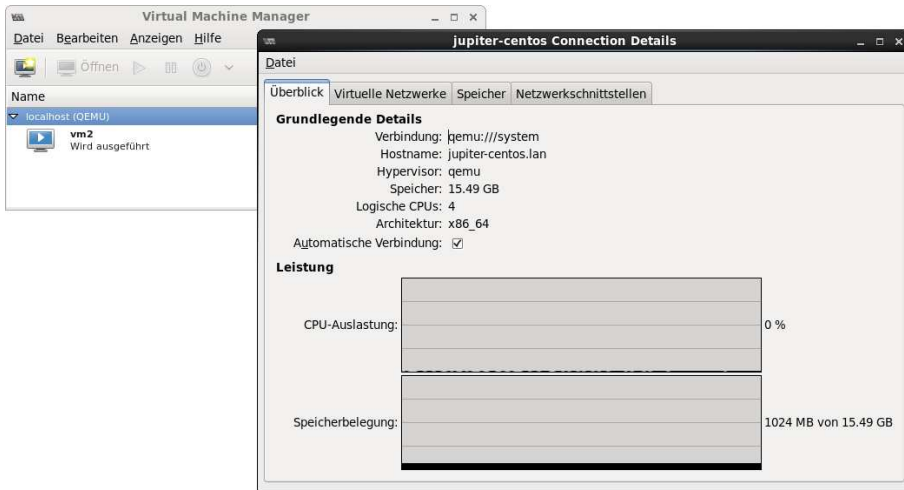


Abbildung 2.1: Eckdaten der Verbindung zwischen dem Virtual Machine Manager und dem libvirt-Dämon

Virtuelle Maschinen ausführen und bedienen

Das Hauptfenster des Virtual Machine Managers enthält eine Liste aller libvirt-Verbindungen. Standardmäßig besteht diese Liste nur aus einem Eintrag, nämlich LOCALHOST (QEMU). Sie können aber mit DATEI|VERBINDUNG HINZUFÜGEN die Eckdaten weiterer KVM-Hosts angeben.

Ein Doppelklick auf einen Eintrag dieser Liste stellt die Verbindung zum KVM-Host her und zeigt dann alle virtuellen Maschinen dieses Hosts an. Bei jeder virtuellen Maschine zeigt ein Icon, ob die Maschine heruntergefahren ist, läuft oder pausiert ist.

Um eine virtuelle Maschine zu starten, klicken Sie deren Eintrag mit der rechten Maustaste an und führen AUSFÜHREN aus. Ein Doppelklick auf den Eintrag öffnet ein neues Fenster, das in zwei Ansichten den Zustand der virtuellen Maschine zeigt:

- » **Die Konsolenansicht** zeigt das Grafiksystem der virtuellen Maschine. Hier sehen Sie die Ausgaben der virtuellen Maschine und können per Tastatur und Maus Eingaben durchführen. Bei virtuellen Maschinen, die im Textmodus laufen, wird der Mauscursor durch einen Klick in der virtuellen Maschine gleichsam eingefangen. `[Strg] + [Alt]` löst den Cursor wieder.
- » **Die Detailansicht** zeigt die Eckdaten der virtuellen Maschine an. Hier können Sie die Hardware-Ausstattung der virtuellen Maschine verändern. Die meisten Änderungen können allerdings nur durchgeführt werden, wenn die virtuelle Maschine gerade nicht läuft.

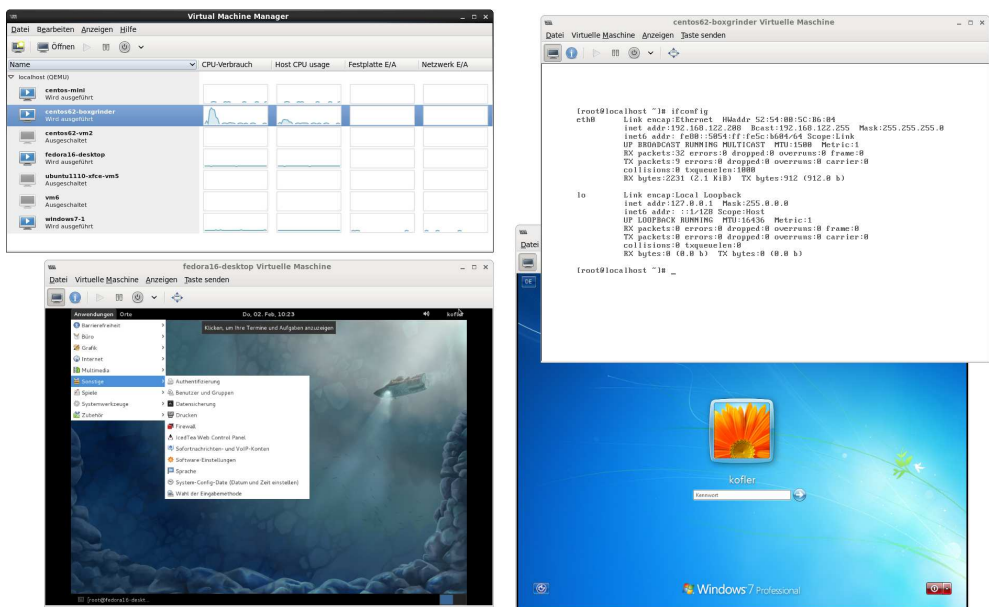


Abbildung 2.2: Der Virtual Machine Manager mit drei aktiven virtuellen Maschinen

Um zwischen den beiden Ansichten umzuschalten, führen Sie ANZEIGEN|KONSOLE bzw. ANZEIGEN|DETAILS aus bzw. klicken in der Symbolleiste auf die entsprechenden Buttons.

TIPP

Unter RHEL und Fedora werden die Fenster zur Darstellung virtueller Maschinen standardmäßig ohne Symbolleiste angezeigt. ANZEIGEN|WERKZEUGLEISTE behebt dieses Manko und ermöglicht dann ein wesentlich komfortableres Wechseln zwischen der Konsolen- und Detailsansicht.

Virtuelle Maschinen stoppen

Die virtuellen Maschinen laufen vollkommen unabhängig vom Virtual Machine Manager! Sie können also die Fenster des Virtual Machine Managers schließen und später wieder öffnen – die virtuellen Maschinen laufen in der Zwischenzeit weiter. Sie können sich sogar aus- und neu einloggen, ohne die Ausführung der virtuellen Maschinen zu beeinträchtigen.

Es gibt vier Möglichkeiten, eine virtuelle Maschine zu stoppen:

- » HERUNTERFAHREN|NEUSTART sendet ein entsprechendes ACPI-Ereignis an die virtuelle Maschine. Wenn die virtuelle Maschine ACPI-Ereignisse verarbeitet (unter Linux ist dazu die Installation des Pakets acpid erforderlich), leitet sie einen Shutdown und anschließend einen Neustart ein.

- » HERUNTERFAHREN|HERUNTERFAHREN leitet via ACPI einen Shutdown ein.
- » HERUNTERFAHREN|FORCIERTES AUSSCHALTEN beendet die Ausführung der virtuellen Maschine sofort – so, als würden Sie bei einem realen Rechner das Stromkabel ziehen. Naturgemäß sollten Sie versuchen, diese Variante des Ausschaltens zu vermeiden, da sie mit Datenverlusten verbunden sein kann.
- » HERUNTERFAHREN|SPEICHERN speichert den Inhalt des virtuellen RAMs der Maschine in einer Datei und beendet dann die Ausführung. Wird die virtuelle Maschine später wieder gestartet, befindet sie sich exakt im selben Zustand wie beim Herunterfahren.

Grundeinstellungen

Im Dialog BEARBEITEN|EINSTELLUNGEN können Sie einige grundlegende Optionen des Virtual Machine Managers einstellen, unter anderem einige Defaulteigenschaften neuer virtueller Maschinen, das Verhalten der Maus, die gewünschte Tastenkombination, um den Tastaturfokus aus einer virtuellen Maschine zu lösen etc. Weiters können Sie nicht nur die CPU-Auslastung, sondern auch I/O- und Netzwerktransfers aufzeichnen. Die Anzeige dieser Daten für jede virtuelle Maschine aktivieren Sie mit ANZEIGEN |GRAPH.

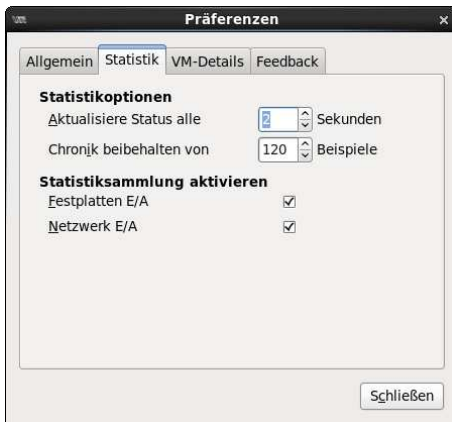


Abbildung 2.3: Einstellungen des Virtual Machine Managers

Speicherort der Image-Dateien

Beim Einrichten virtueller Maschinen ist es am einfachsten, so genannte Image-Dateien als virtuelle Festplatten für die Gäste zu verwenden. Der Virtual Machine Manager speichert solche Image-Dateien standardmäßig im Verzeichnis `/var/lib/libvirt/images`. Dieses Verzeichnis kann im Virtual Machine Manager nicht geändert werden. Es ist aber möglich, im Dialogblatt BEARBEITEN|VERBINDUNGSDetails|SPEICHER mit dem Plus-

Button einen neuen Datenträgerpool hinzuzufügen, wobei Sie den Typ DIR: FILESYSTEM DIRECTORY wählen und ein beliebiges Verzeichnis angeben.

Beim Einrichten der virtuellen Maschine müssen Sie im vierten Schritt des Assistenten NEUE VM die Option VERWALTETEN ODER ANDEREN SPEICHER aktivieren und können dann mit DURCHSUCHEN eine Image-Datei im neuen Speicherpool erzeugen und auswählen. Umfassende Informationen zum Umgang mit Image-Dateien und Speicherpools folgen dann in Kapitel 4.

Administration von virtuellen Maschinen im Netzwerk via SSH

Sie können den Virtual Machine Manager auch verwenden, um einen via SSH erreichbaren externen KVM-Host zu administrieren. Dazu definieren Sie zuerst mit DATEI|VERBINDUNG HINZUFÜGEN eine Verbindung zu diesem Server, wobei Sie als Verbindungsmethode SSH auswählen. Beim Verbindungsaufbau müssen Sie zudem das entsprechende Login-Passwort angeben (siehe Abbildung 2.4).

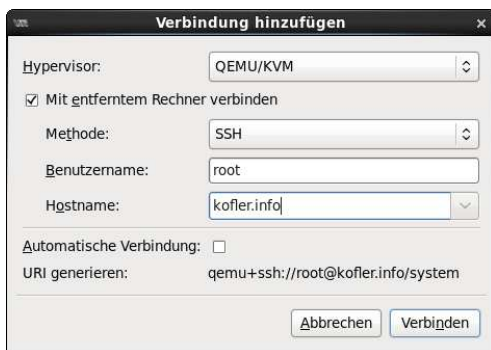


Abbildung 2.4: Verbindungsaufbau via SSH

Wenn der externe KVM-Host unter RHEL oder Fedora läuft, erfordert die libvirt-Administration root-Rechte und somit einen root-Login via SSH. Aus Sicherheitsgründen sind SSH-Server aber häufig so konfiguriert, dass ein direkter root-Login unmöglich ist.


Ein Kompromiss kann so aussehen, dass Sie den SSH-Server so konfigurieren, dass ein root-Login nur bei einer Authentifizierung durch einen Schlüssel akzeptiert wird, nicht aber per Passwort. Dazu erzeugen Sie zuerst auf Ihrem lokalen Rechner mit `ssh-keygen` ein Schlüsselpaar. Diesen Schlüssel sollten Sie durch eine Passphrase selbst verschlüsseln. (Eine Passphrase ist ein aus mehreren Wörtern bestehendes Passwort.)

```
user@localmachine$ ssh-keygen
```

```
Generating public/private rsa key pair.
```

```
Enter file in which to save the key (/home/user/.ssh/id_rsa): <Return>
```

```
Enter passphrase (empty for no passphrase):  *****
Enter same passphrase again:  *****
Your identification has been saved in /home/user/.ssh/id_rsa.
Your public key has been saved in /home/user/.ssh/id_rsa.pub.
```

Wenn Sie die Passphrase-Frage einfach mit  oder mit der Eingabe empty beantworten, verzichtet ssh-keygen auf die Verschlüsselung. Das ist bequem, weil es eine SSH-Nutzung ohne Passwort-Rückfrage ermöglicht. Sie gehen damit aber ein Sicherheitsrisiko ein: Wem immer Ihr Schlüssel auf dem lokalen Rechner in die Hände gerät, der kann sich ohne Weiteres auf allen Rechnern anmelden, auf denen Sie den öffentlichen Teil des Schlüssels installiert haben!

Anschließend fügen Sie mit `ssh-copy-id` den öffentlichen Teil Ihres Schlüssels in die Datei `.ssh/authorized_keys` auf dem Server ein:

```
user@localmachine$ ssh-copy-id -i root@kvmhost:key
root@kvmhost's password:  *****
```

Nachdem Sie den SSH-Login ohne Schlüssel getestet haben, ändern Sie auf dem KVM-Host die Datei `/etc/ssh/sshd_config` und fügen dort die folgende `PermitRootLogin`-Anweisung ein. Sie bewirkt, dass ein root-Login nur möglich ist, wenn die Authentifizierung mit einem Schlüssel oder einem anderen Verfahren erfolgt, das sicherer ist als die simple Passwortangabe.

```
# in /etc/ssh/sshd_config
...
PermitRootLogin without-password
```

Die geänderten Einstellungen werden mit `service sshd reload` wirksam und gelten dann für jedes Programm, das sich via SSH authentifiziert, somit also auch für den Virtual Machine Manager.

Authentifizierung per SASL/Kerberos und TLS/x509

Der Virtual Machine Manager bzw. die zugrunde liegenden libvirt-Bibliotheken unterstützen außer der gerade beschriebenen SSH-Variante auch eine Authentifizierung und Verschlüsselung der Netzwerkkommunikation durch SASL/Kerberos sowie durch TLS/x509. Die Datei `/etc/libvirt/libvirtd.conf` steuert, welche Verfahren zulässig sind und wo sich gegebenenfalls die Schlüsseldateien befinden. Detaillierte Anleitungen und Konfigurationstipps finden Sie in Abschnitt 5.6.

2.3 Linux-Gast einrichten

Grundsätzlich eignen sich alle aktuellen Linux-Distributionen zur Installation als KVM-Gast. Vorweg sollten Sie sich aber einige Gedanken zum richtigen Desktop-System machen.

- » **Gnome 3 und Unity:** KVM kann keine 3D-Funktionen an den Gast weitergeben. Deswegen funktionieren in einer virtuellen Maschine weder Gnome 3 noch das von Ubuntu favorisierte Desktop-System Unity. Natürlich können Sie dennoch auf Gnome 3 oder Unity basierende Distributionen installieren – zur Desktop-Bedienung müssen Sie dann aber auf die jeweiligen Fallback-Lösungen zurückgreifen: Bei Gnome 3 sieht der Fallback-Modus ähnlich aus wie bei Gnome 2. Bei Ubuntu wird seit Version 11.10 Unity-2D mitgeliefert. Dabei handelt es sich um eine Unity-Variante, die ganz ähnlich wie das originale Unity aussieht, aber keine 3D-Funktionen voraussetzt und in virtuellen Maschinen gut funktioniert.
- » **XFCE, LXDE:** Gut geeignet für den Einsatz in virtuellen Maschinen sind die Desktop-Systeme XFCE und LXDE. Ihre Installation kostet weniger Platz auf der (virtuellen) Festplatte. Im Betrieb ist der Ressourcenbedarf geringer als bei Gnome oder Unity. 3D-Funktionen sind nicht erforderlich.
- » **Server-Installation ohne Desktop:** Wenn eine virtuelle Maschine nur Server-Funktionen erfüllen soll, ist eine Installation ganz ohne Desktop-System naheliegend. Die virtuelle Maschine kann dann allerdings nur im Textmodus bzw. via SSH gesteuert und konfiguriert werden. Tipps zur Durchführung einer Minimalinstallation von RHEL/Fedora und Ubuntu folgen auf Seite 38).

Neue virtuelle Maschine erstellen

Wir setzen jetzt voraus, dass der Virtual Machine Manager läuft und Sie eine Verbindung zu einem QEMU/KVM-Host hergestellt haben. Das Einrichten einer neuen virtuellen Maschine beginnt mit dem Button **NEUE VIRTUELLE MASCHINE ERSTELLEN**. Bei der Einstellung der Eckdaten hilft ein Assistent in fünf Schritten:

- » Im **ersten Schritt** geben Sie den Namen der virtuellen Maschine und deren Installationsquelle an. Bei einer Linux-Installation handelt es sich üblicherweise um eine ISO-Datei. Es ist aber auch möglich, die Installationsdaten vom DVD-Laufwerk des Host-Rechners zu lesen oder über ein entsprechend eingerichtetes Netzwerk zu beziehen. Die Option **VORHANDENES FESTPLATTEN-ABBILD IMPORTIEREN** erzeugt eine Kopie einer bereits vorhandenen Image-Datei mit einer virtuellen Maschine.

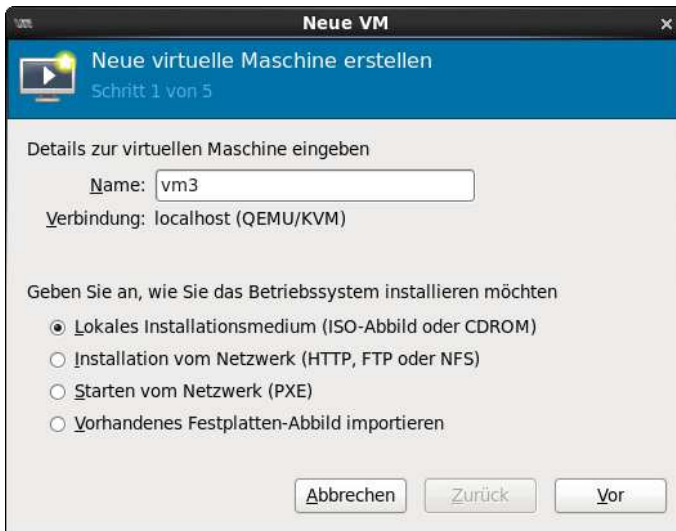


Abbildung 2.5: Der Assistent zum Einrichten einer neuen virtuellen Maschine

- » Wenn Sie im ersten Schritt ein ISO-Abbild oder eine CD/DVD als Installationsquelle ausgewählt haben, können Sie im **zweiten Schritt** den Dateinamen einer ISO-Datei oder das CD/DVD-Laufwerk angeben. Der Button DURCHSUCHEN führt in einen Dialog, der vorerst nur die dem Virtual Machine Manager bekannten STORAGE POOLS anzeigt. Um eine ISO-Datei direkt auszuwählen, müssen Sie in diesem Dialog den Button LOKAL DURCHSUCHEN anklicken.

Außerdem stellen Sie im zweiten Schritt des Assistenten den Typ des Betriebssystems (z. B. WINDOWS oder LINUX) und die Version des Gastsystems ein (z. B. RED HAT ENTERPRISE LINUX 6). Diese Einstellungen haben Einfluss darauf, wie der Assistent in der Folge die (virtuellen) Hardware-Komponenten einrichtet. Beispielsweise verwendet der Virtual Machine Manager die effizienten paravirtualisierten virtio-Treiber für den Festplatten- und Netzwerkzugriff nur bei aktuellen Linux-Distributionen.

- » Im **dritten Schritt** geben Sie an, wie viel Speicher (RAM) und wie viele CPU-Cores Sie der virtuellen Maschine zuweisen möchten.
- » Im **vierten Schritt** richten Sie die virtuelle Festplatte ein. Normalerweise werden Sie die bereits vorselektierte Option PLATTENABBILD AUF FESTPLATTE DES SYSTEMS ERSTELLEN nutzen. Die neue Image-Datei wird standardmäßig im Verzeichnis /var/lib/libvirt/images angelegt. Beachten Sie, dass eine nachträgliche Vergrößerung der virtuellen Festplatte mit großem Aufwand verbunden ist.

Neben der Größe können Sie auch auswählen, ob der virtuelle Festplattenspeicher sofort zugewiesen wird oder ob die virtuelle Festplatte erst bei Bedarf wachsen soll.

Ersteres ist effizienter und schließt aus, dass zu einem späteren Zeitpunkt vielleicht nicht genug Platz auf der Festplatte ist, um dem steigenden Platzbedarf der virtuellen Maschine gerecht zu werden.

Die Alternative VERWALTETEN ODER ANDEREN SPEICHER WÄHLEN ist nur dann von Relevanz, wenn Sie vorher mit BEARBEITEN|VERBINDUNGSDetails im Dialogblatt SPEICHER weitere Speicherpools eingerichtet haben (siehe Kapitel 4).

- » Im **fünften Schritt** können Sie schließlich in den ERWEITERTEN OPTIONEN die Netzwerkschnittstelle konfigurieren. Der Assistent weist dem Netzwerkadapter jeder virtuellen Maschine eine eindeutige MAC-Adresse der Form 52:54:00:nn:nn:nn zu. Standardmäßig wird die virtuelle Maschine mittels NAT (Network Address Translation) mit dem Hostsystem verbunden. Damit kann die virtuelle Maschine den Internetzugang des Host-Rechners nutzen, aber keine Verbindungen zu anderen Rechnern in Ihrem lokalen Netzwerk herstellen. Andere Formen der Netzwerkanbindung sind in Kapitel 5 beschrieben.

Mit dem Button ABSCHLIESSEN wird die Konfiguration beendet und die neue virtuelle Maschine sofort gestartet. Wenn Sie das nicht wünschen, aktivieren Sie im letzten Dialogblatt des Assistenten die Option KONFIGURATION BEARBEITEN VOR DER INSTALLATION. Damit gelangen Sie nach dem Ende des Assistenten in einen Dialog, der die Hardware-Komponenten der virtuellen Maschine zusammenfasst (siehe Abbildung 2.6).

Mit dem Abschluss der Konfiguration der virtuellen Maschine wird diese gestartet. Die Ausgaben der virtuellen Maschine sehen Sie in einem neuen Fenster des Virtual Machine Managers. (Hinter den Kulissen agiert dieses Fenster als VNC- oder Spice-Client.)

Solange die virtuelle Maschine ein Linux-System im Textmodus ausführt, ist die Maus nach einem Klick in das Fenster gewissermaßen »gefangen«. Alle Tastatureingaben werden jetzt an die virtuelle Maschine weitergeleitet. Um den Eingabefokus zu lösen und die Maus wieder zu befreien, drücken Sie gleichzeitig die linken **Strg**- und **Alt**-Tasten. Sobald in der virtuellen Maschine ein Grafiksystem gestartet wird, kann die Maus einfach aus dem Fenster hinausbewegt werden.

Die eigentliche Linux-Installation unterscheidet sich nicht von einer Installation auf einem realen Rechner (siehe Abbildung 2.7). Einzig die (virtuelle) Festplatte ist vermutlich kleiner, als Sie es sonst gewohnt sind.

Das Fenster einer virtuellen Maschine kann jederzeit geschlossen werden, ohne die Ausführung einer virtuellen Maschine zu beeinträchtigen. Sie können sich auch aus- und neu einloggen, den Virtual Machine Manager neu starten und dann die Verwaltung der weiterhin laufenden virtuellen Maschinen fortsetzen.

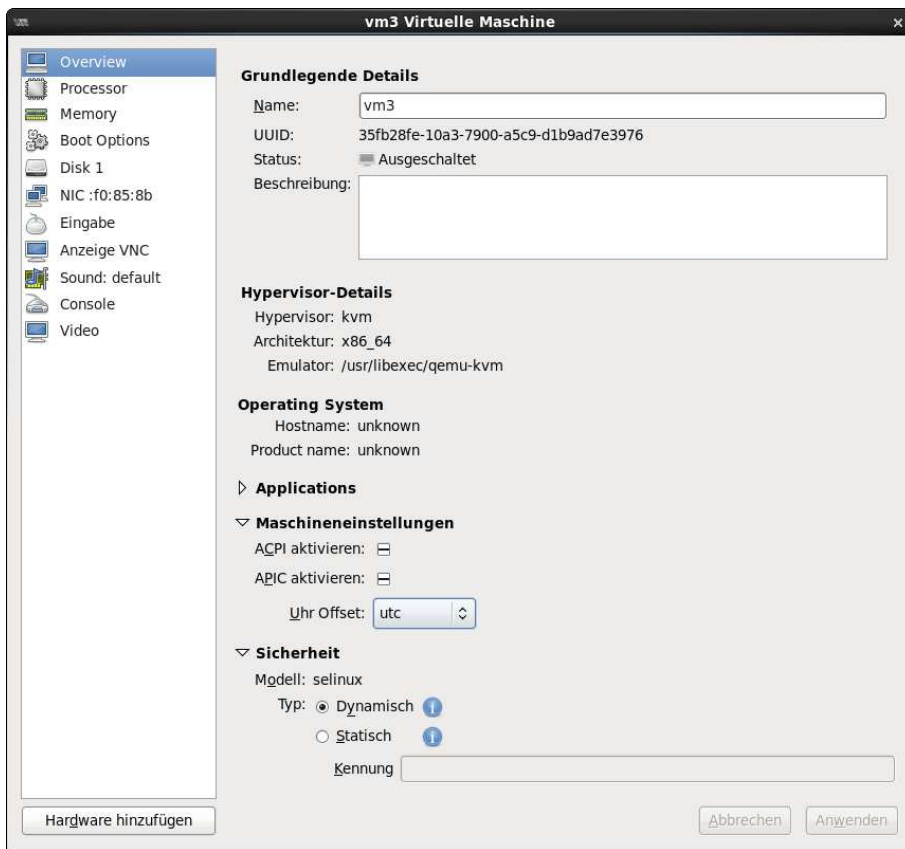


Abbildung 2.6: Hardware-Verwaltung im Virtual Machine Manager

Die Hardware-Komponenten der virtuellen Maschine

Wenn Sie eine virtuelle Linux-Maschine einrichten und dabei die Defaulteinstellungen beibehalten, gelten unter anderem die folgenden Hardware-Einstellungen:

- » eine CPU mit einem Core
- » einen virtio-Netzwerkadapter, NAT-Netzwerk
- » eine virtio-Festplatte (ist im Gast als `/dev/vda` sichtbar)
- » eine Cirrus-kompatible VGA-Karte, Auflösung maximal 1024*768 Pixel

Innerhalb der virtuellen Maschine können Sie mit `lsmod` überprüfen, welche virtio-Treiber geladen sind:

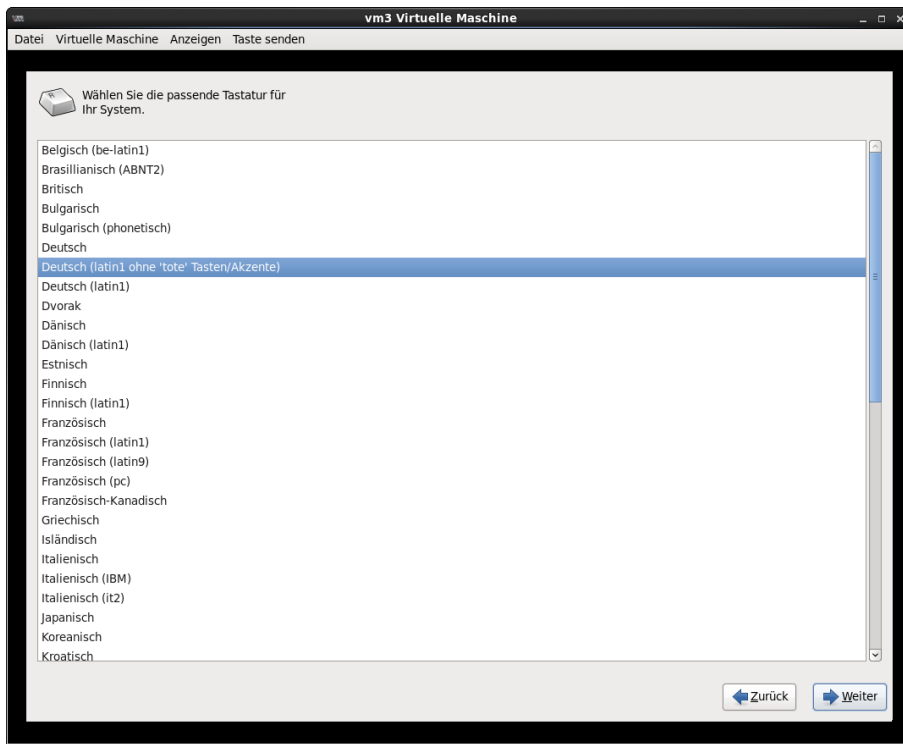


Abbildung 2.7: Installation eines Linux-Gasts in einer virtuellen Maschine

```
user@gast$ lsmod | grep virtio
virtio_balloon    4347  0
virtio_net        15839 0
virtio_blk        6671  3
virtio_pci        6687  0
virtio_ring       7729  4 virtio_balloon,virtio_net,virtio_blk,virtio_pci
virtio            4890  4 virtio_balloon,virtio_net,virtio_blk,virtio_pci
```

Wenn Sie die Parameter einer virtuellen Maschine verändern möchten, öffnen Sie per Doppelklick das Fenster der virtuellen Maschine und wechseln dann in die sogenannte Detailansicht (ANZEIGEN|DETAILS, die wie Abbildung 2.6 auf Seite 36 aussieht).

Für die meisten Modifikationen muss die virtuelle Maschine vorher heruntergefahren werden. Die Benutzeroberfläche des Virtual Machine Managers ist mitunter nicht in der Lage, einzelne Optionen einer Hardware-Komponente zu verändern. Vielmehr müssen Sie zunächst die ganze Komponente löschen und dann wieder neu anlegen. Details zu den möglichen Hardware-Einstellungen folgen in den Kapitel 4 bis 8.

Achten Sie darauf, dass Sie geänderte Einstellungen für jedes Dialogblatt zuerst explizit mit ANWENDEN bestätigen, bevor Sie in ein anderes Dialogblatt wechseln – andernfalls werden die Änderungen nicht gespeichert.

TIPP

Wenn Sie einer virtuellen Maschine einen neuen Namen geben möchten, müssen Sie diese zuerst herunterfahren. Anschließend geben Sie im Dialogblatt OVERVIEW der Hardware-Einstellungen den neuen Namen an und klicken auf ANWENDEN.

2.4 Konfigurationstipps für Fedora- und RHEL-Gäste

In diesem Abschnitt erfahren Sie, wie Sie eine minimale Fedora- oder RHEL-Installation durchführen und wie Sie die Basiskonfiguration im Textmodus durchführen. Minimalinstallationen sind dann zweckmäßig, wenn Sie möchten, dass Ihre virtuelle Maschine möglichst wenige Ressourcen beansprucht. Im Server-Einsatz ist das zumeist zweckmäßig. Eine Minimalinstallation erschwert freilich auch die Konfiguration, weil die aus dem Desktop-Betrieb gewohnten Administrationswerkzeuge nicht zur Verfügung stehen.

Die Tipps aus diesem Abschnitt werden Ihnen auch dann weiterhelfen, wenn Sie aus dem Internet eine fertige virtuelle Maschine (eine sogenannte *Appliance*) herunterladen oder wenn Sie mit dem im nächsten Kapitel beschriebenen Werkzeug BoxGrinder eine automatisierte Installation durchführen.

Minimalinstallation

Um eine Minimalinstallation von Fedora durchzuführen, stellen Sie bei der Software-Auswahl anstelle der vorgegebenen Option GRAFISCHE OBERFLÄCHE die Option MINIMAL ein. Bei RHEL-Distributionen ist diese Option bereits voreingestellt.

Außer root werden keine Benutzer eingerichtet. Immerhin wird standardmäßig ein SSH-Server installiert und eine Firewall eingerichtet. Auch SELinux ist aktiv. Die gesamte weitere Administration muss nun mit textbasierten Werkzeugen erfolgen und setzt daher gute Fedora- oder RHEL-Grundlagenkenntnisse voraus. Das betrifft auch die Netzwerkkonfiguration – standardmäßig ist nur die Loopback-Schnittstelle aktiv.

Tastatur

Wenn Sie die virtuelle Maschine nicht selbst installiert haben, sondern auf eine fertige virtuelle Maschine zurückgreifen oder eine BoxGrinder-Installation erstmalig starten, gilt das US-Tastaturlayout. Unmittelbar Abhilfe schafft das folgende Kommando:

```
root# loadkeys de
```

Das neue Tastaturlayout gilt damit allerdings nur bis zum nächsten Neustart der virtuellen Maschine. Um das Tastenlayout dauerhaft zu ändern, ändern Sie die Datei `/etc/sysconfig/keyboard` wie folgt:

```
# Datei /etc/sysconfig/keyboard
KEYTABLE="de-latin1-nodeadkeys"
MODEL="pc105"
LAYOUT="de"
KEYBOARDTYPE="pc"
VARIANT="nodeadkeys"
```

Damit die neuen Einstellungen unmittelbar beim Start berücksichtigt werden, müssen sie in die `initrd`-Datei übernommen werden. Diese Datei mit diversen Modulen und Einstellungen wird beim Systemstart an den Kernel übergeben. Um die zum aktuellen Kernel passende `initrd`-Datei neu zu erzeugen, führen Sie dieses Kommando aus:

```
root# dracut -f
```

Zeitzone

Bei virtuellen Maschinen, die Sie vorkonfiguriert aus dem Internet heruntergeladen haben oder mit BoxGrinder erstellt haben, ist die Zeitzone häufig falsch eingestellt. Abhilfe schafft das folgende Kommando, wobei Sie `Europe/Berlin` gegebenenfalls durch einen anderen Ort ersetzen müssen:

```
root# cp /usr/share/zoneinfo/Europe/Berlin /etc/localtime
```

Netzwerkconfiguration

Wenn Sie selbst eine Minimalinstallation durchgeführt haben, ist die Netzwerkschnittstelle nicht konfiguriert und Sie haben somit im Gast noch keinen Netzwerk- und Internetzugang. Wenn die virtuelle Maschine ihre Netzwerkparameter via DHCP bezieht, richten Sie einfach die Datei `/etc/sysconfig/network-scripts/ifcfg-eth0` wie folgt ein:

```
# Datei /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
HWADDR=52:54:00:xx:xx:xx      (eigene MAC-Adresse)
NM_CONTROLLED=no
ONBOOT=yes
BOOTPROTO=dhcp
TYPE=Ethernet
USERCTL=no
PEERDNS=yes
IPV6INIT=no
```

Die MAC-Adresse des Netzwerkadapters der virtuellen Maschine können Sie entweder der XML-Datei mit der Beschreibung der virtuellen Maschine entnehmen [/etc/libvirt/qemu/name.xml] oder in der virtuellen Maschine mit `ifconfig -a` ermitteln.

Bei einer statischen Konfiguration muss die Datei dem folgenden Muster entsprechen, wobei Sie die IP-Adressen und -Masken durch eigene Werte ersetzen müssen:

```
# Datei /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
HWADDR=52:54:00:xx:xx:xx      (eigene MAC-Adresse)
NM_CONTROLLED=no
ONBOOT=yes
BOOTPROTO=none
TYPE=Ethernet
USERCTL=no
IPV6INIT=no
IPADDR=10.0.17.33
NETMASK=255.255.255.0
NETWORK=10.0.17.0
BROADCAST=10.0.17.255
GATEWAY=10.0.17.1
```

Die Gateway-Adresse können Sie auch in `/etc/sysconfig/network` einstellen. Das ist dann zweckmäßig, wenn es ein zentrales Gateway für alle Netzwerkschnittstellen gibt. Den oder die Nameserver tragen Sie in `/etc/resolv.conf` ein:

```
# /etc/resolv.conf
nameserver 10.0.17.1    # erster DNS
nameserver 10.0.17.2    # zweiter DNS
```

Falls Sie den Hostnamen verändern möchten, finden Sie den entsprechenden Parameter in der Datei `/etc/sysconfig/network`. Im Regelfall ist es zweckmäßig, die Zuordnung der IP-Adresse des Rechners zu seinem Hostnamen auch in `/etc/hosts` einzutragen:

```
# /etc/hosts
...
10.0.17.33      myhostname.mydomainname myhostname
```

Das folgende Kommando aktiviert die Netzwerkeinstellungen:

```
root# service network restart
```

Während dieser Konfigurationsarbeiten steht Ihnen als einziger Editor `vi` zur Verfügung. Wenn Sie einen anderen Editor vorziehen, können Sie die Netzwerkschnittstelle `eth0` vorweg durch das Kommando `dhclient eth0` aktivieren. (Das setzt voraus, dass die virtuelle Maschine in einem Netzwerk mit DHCP-Server läuft.) Anschließend können Sie mit `yum` einen anderen Editor installieren, z. B. `nano`.

Netzwerkconfiguration bei einer Desktop-Installation

Haben Sie Fedora oder RHEL samt Desktop installiert, ersparen Sie sich die manuelle Bearbeitung von Konfigurationsdateien. Falls Sie eine statische Netzwerkconfiguration durchführen möchten, ist es aber vielfach zweckmäßig, den vorinstallierten Network-Manager zu deinstallieren.

```
root# yum remove NetworkManager
```

Anschließend führen Sie die Netzwerkconfiguration wie oben beschrieben in der Datei `/etc/sysconfig/network-scripts/ifcfg-eth0` durch. Entscheidend ist dabei die Einstellung `NM_CONTROLLED=no` – der NetworkManager hat also keine Kontrolle mehr über die Schnittstelle. Zuletzt starten Sie das Init-Script, das für die manuelle Netzwerkconfiguration verantwortlich ist:

```
root# service network start           (gilt sofort)
root# chkconfig network on           (gilt ab dem nächsten Neustart)
```

ACPI-Dämon installieren

Damit Sie virtuelle Maschinen via `virsh shutdown` oder über den Virtual Machine Manager herunterfahren können, muss der ACPI-Dämon laufen. Bei Minimalinstallationen ist das nicht der Fall. Abhilfe schaffen die folgenden Kommandos:

```
root# yum install acpid
root# service acpid start
```

Serielle Schnittstelle aktivieren

In der Regel werden Sie Ihre virtuelle Maschine via VNC oder SSH steuern. Alternativ können Sie stattdessen auch die Konsole verwenden, in der Sie das KVM-Kommando oder `virsh` ausführen (siehe Seite 210). Damit ein Getty-Prozess auf der seriellen Schnittstelle arbeitet, richten Sie unter RHEL die folgende Upstart-Konfigurationsdatei ein:

```
# Datei /etc/init/ttyS0.conf      (RHEL)
start on stopped rc RUNLEVEL=[12345]
stop on runlevel [!12345]
respawn
exec /sbin/mingetty /dev/ttyS0
```

Fedora verwendet seit Version 15 Systemd als Init-System (also nicht mehr Upstart). Damit auch hier ein Getty-Prozess für die serielle Schnittstelle gestartet wird, müssen Sie den folgenden Link einrichten:

```
root# ln -s /lib/systemd/systemd/serial-getty@.service \  
      /etc/systemd/system/getty.target.wants/serial-getty@ttyS0.service
```

Wenn Sie sich über die serielle Konsole als root anmelden möchten (nicht als gewöhnlicher Benutzer), dann müssen Sie unabhängig vom Init-System an das Ende der Datei `/etc/securetty` eine Zeile mit dem Eintrag `ttyS0` anfügen! Andernfalls ist der root-Login aus Sicherheitsgründen blockiert.

GRUB-Wartezeit minimieren

Beim Start der virtuellen Maschine wird für fünf Sekunden das GRUB-Menü angezeigt. Da virtuelle Maschinen in der Regel ohne interaktiven Eingriff gestartet werden, ist diese Wartezeit unnötig. Außerdem ist diese Zeitspanne auf einem performanten KVM-Host größer als die anschließende Boot-Zeit!

Unter RHEL kommt bis einschließlich Version 6.2 GRUB 0.97 zum Einsatz. (Ob bzw. wann RHEL auf GRUB 2 umsteigen wird, ist momentan nicht bekannt.) Die GRUB-Wartezeit wird durch den Parameter `timeout` in `/etc/grub.conf` gesteuert. In dieser Datei durchgeführte Änderungen werden beim nächsten Bootprozess wirksam.

Fedora verwendet seit Version 16 GRUB 2. In diesem Fall bestimmt der Parameter `GRUB_TIMEOUT` in der Datei `/etc/default/grub` die Zeit, während der das GRUB-Menü angezeigt wird. Hier durchgeführte Änderungen werden allerdings erst wirksam, wenn Sie danach das folgende Kommando ausführen:

```
root# grub2-mkconfig -o /boot/grub2/grub.cfg    (nur bei GRUB 2!)
```

2.5 Konfigurationstipps für Ubuntu-Gäste

Dieser Abschnitt fasst einige Tipps zusammen, wie Sie eine minimale Ubuntu-Installation durchführen und wie Sie die grundlegende Konfiguration im Textmodus bewältigen (Tastatur, Zeitzone, Netzwerk etc.). Die hier gesammelten Ratschläge sind auch dann wertvoll, wenn Sie aus dem Internet eine fertig konfigurierte Ubuntu-Maschine herunterladen oder wenn Sie mit dem im nächsten Kapitel beschriebenen Script VMBuilder eine automatisierte Installation durchführen.

Minimalinstallation

Wenn Sie in einer virtuellen Maschine eine minimale Ubuntu-Installation durchführen möchten, sollten Sie als Installationsquelle das ISO-Image der Alternate- oder Server-CD verwenden (nicht die ISO-Datei zur Desktop-Installation). Sollten Sie später doch eine Desktop-Umgebung wünschen, können Sie deren Installation jederzeit mit `apt-get install ubuntu-desktop` (Gnome) bzw. `yum install xubuntu-desktop` (XFCE) nachholen.

Am Beginn einer Ubuntu-Server-Installation können Sie mit **[F4]** die Installationsvariante **EINE MINIMALE VIRTUELLE MASCHINE INSTALLIEREN** auswählen. Der Vorteil gegenüber einer herkömmlichen Server-Installation besteht darin, dass ein spezieller Kernel eingesetzt wird, der für den Einsatz in virtuellen Maschinen optimiert ist und mit wenig zusätzlichem Ballast auskommt.

Tastatur und Konsole

Nach einer Minimalinstallation oder bei der Verwendung einer fertigen virtuellen Maschine gilt in der Regel das US-Tastaturlayout. Abhilfe schafft das folgende Kommando, dessen Eingabe Sie trotz US-Tastaturlayout schaffen sollten. (Y und Z sind vertauscht, den Bindestrich geben Sie auf einer deutschen Tastatur mit **[B]** ein.)

```
root# dpkg-reconfigure keyboard-configuration
```

Während der Ausführung des Kommandos können Sie dann interaktiv mit den Cursor-tasten Ihr Tastaturmodell, das Layout sowie einige weitere Optionen eingeben. Die Einstellungen werden in `/etc/default/keyboard` gespeichert. Außerdem wird im Anschluss an die Konfiguration die Datei `/etc/console-setup/cached.kmap.gz` erzeugt und `update-initramfs` ausgeführt.

Nur in seltenen Fällen muss auch der Zeichensatz bzw. die Schrift für die Konsole richtig eingestellt werden:

```
root# dpkg-reconfigure console-setup
```

Zeitzone

Die virtuelle Maschine erhält vom KVM-Host üblicherweise die koordinierte Weltzeit (UTC-Zeit). Welche lokale Zeit in der virtuellen Maschine gilt, hängt davon ab, wie die Zeitzone konfiguriert ist. Wenn `date` die falsche Zeit liefert, stellen Sie mit dem folgenden Kommando die Zeitzone richtig ein. (Intern wird dadurch die Datei `/etc/localtime` mit dem Inhalt einer Zeitzonendatei aus `/usr/share/zoneinfo` überschrieben.)

```
root# dpkg-reconfigure tzdata
```

Netzwerkkonfiguration

Das Ubuntu-Installationsprogramm hilft Ihnen auch bei einer Minimalinstallation bei der Netzwerkkonfiguration. Wenn Sie die Konfiguration später ändern möchten, ist der zentrale Dreh- und Angelpunkt die Datei `/etc/network/interfaces`. Bezieht die virtuelle Maschine ihre Netzwerkparameter von einem DHCP-Server, muss diese Datei so aussehen:

```
# /etc/network/interfaces
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet dhcp
```

Bei einer statischen Konfiguration können Sie sich am folgenden Muster orientieren:

```
# /etc/network/interfaces
auto lo
iface lo inet loopback

auto eth0 inet static
    address 10.0.17.33
    netmask 255.255.255.0
    gateway 10.0.17.1
    dns-nameservers 10.0.17.1
```

Falls Sie den Hostnamen neu einstellen möchten, führen Sie die Änderungen in `/etc/hostname` und `/etc/hosts` durch. Bei einer Ubuntu-Desktop-Installation ist es zudem ratsam, den Network Manager zu deinstallieren:

```
root# apt-get remove libnm* network-manager*
```

Damit Änderungen an der Konfiguration wirksam werden, führen Sie das folgende Kommando aus:

```
root# /etc/init.d/networking restart
```

HINWEIS

Das obige Kommando liefert die Warnung */etc/init.d/networking restart is deprecated*, funktioniert aber in der Regel. Eine vernünftige Alternative zu dem Kommando existiert momentan nicht, wie Sie hier nachlesen können:

<http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=565187>
<http://tinyurl.com/6uqdbpt>

ACPI-Dämon installieren

Um virtuelle Ubuntu-Maschinen via `virsh shutdown` oder über den Virtual Machine Manager herunterfahren zu können, muss der ACPI-Dämon laufen. Virtuelle Maschinen mit Ubuntu 12.04 werden durch `halt` zwar angehalten, aber nicht ausgeschaltet. Abhilfe: führen Sie `halt -p` aus!

Serielle Schnittstelle aktivieren

Wenn Sie die virtuelle Maschine in einer Konsole im Textmodus steuern möchten (siehe Seite 210), muss während des Systemstarts ein Getty-Prozess für die serielle Schnittstelle gestartet werden. Dazu richten Sie die folgende neue Upstart-Konfigurationsdatei ein:

```
# Datei /etc/init/ttyS0.conf (Ubuntu)
start on stopped rc RUNLEVEL=[12345]
stop on runlevel [!12345]
respawn
exec /sbin/getty -L -8 38400 ttyS0 vt102
```

Wenn Sie möchten, dass auch die Kernelmeldungen über die serielle Schnittstelle ausgegeben werden, ändern Sie `/etc/default/grub` wie folgt und führen dann `update-grub` aus:

```
# Datei /etc/default/grub (Ubuntu)
...
GRUB_CMDLINE_LINUX_DEFAULT="console=ttyS0"
GRUB_TERMINAL=console
...
```

2.6 Windows-Gast einrichten

KVM ist Windows-kompatibel, und prinzipiell unterscheidet sich eine Windows-Installation nur unwesentlich von einer Linux-Installation. Sie beginnen abermals damit, dass Sie eine neue virtuelle Maschine einrichten. Achten Sie darauf, dass Sie im zweiten Schritt des Assistenten den Betriebssystemtyp `WINDOWS` und die entsprechende Windows-Version auswählen! Nur dann verwendet der Virtual Machine Manager für Windows geeignete virtuelle Hardware-Komponenten.

Selbstverständlich erfordert eine Windows-Installation eine entsprechende Lizenz. Warten Sie mit der Online-Registrierung aber so lange, bis Sie mit der Leistung zufrieden sind. Wenn Sie später in den Einstellungen der virtuellen Maschine das RAM vergrößern oder andere virtuelle Hardware-Parameter ändern, müssen Sie die Registrierung unter Umständen wiederholen!

Uhrzeit

Große Probleme gab es bei unseren Tests mit der Uhr, die im Windows-Gast viel zu schnell lief: Alle fünf Sekunden (real) sprang die Uhrzeit um eine Minute voraus. In der

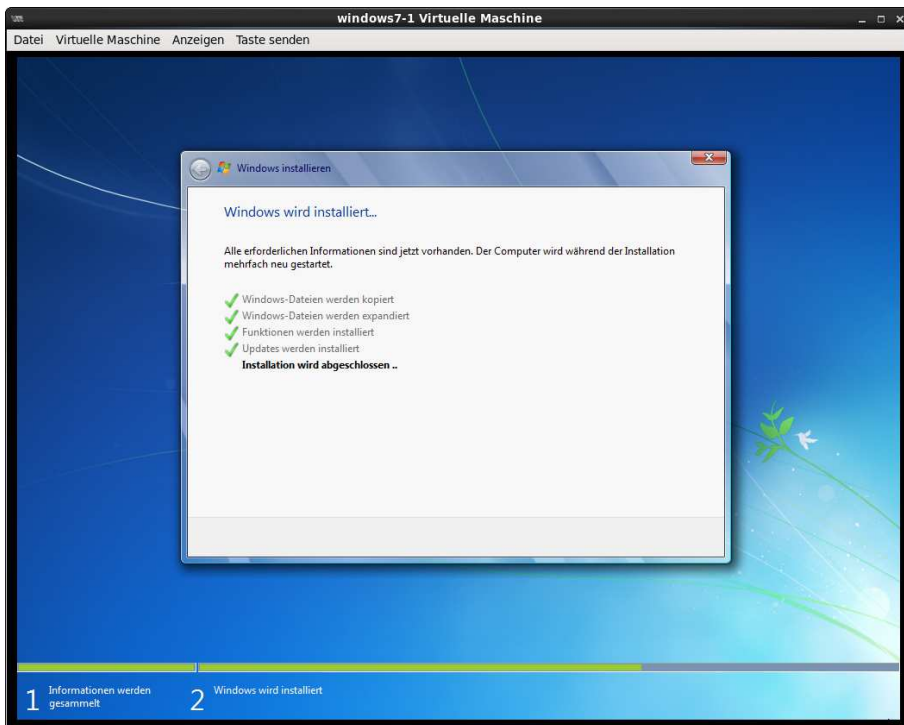


Abbildung 2.8: Installation von Windows 7

analogen Uhrendarstellung sieht der rasende Sekundenzeiger aus wie in einem Science-Fiction-Film, in dem der Effekt einer Zeitmaschine demonstriert werden soll.

Unter Windows 7 bzw. Windows Server 2008 beheben Sie dieses Problem, indem Sie die Uhr von Time Stamp Counter (TSC) auf Real-Time Clock (RTC) umstellen. Dazu öffnen Sie im Windows-Gast das Startmenü, klicken den Eintrag ALLE PROGRAMME|ZUBEHÖR|EINGABEAUFFORDERUNG mit der rechten Maustaste an und wählen den Kontextmenüeintrag ALS ADMINISTRATOR AUSFÜHREN aus. Im Kommandofenster führen Sie die folgende Anweisung aus:

> **bcdedit /set default USEPLATFORMCLOCK on**

Falls Sie Windows XP oder den Windows Server 2003 virtualisieren, stellen Sie das gewünschte Timing-Verfahren in der Datei boot.ini ein. Dazu fügen Sie am Ende dieser Datei die folgende Zeile ein:

```
/usepmtimer
```

Unabhängig von der Windows-Version gilt: Die geänderte Uhr wird erst mit einem Neustart aktiv. Hintergründe zu den hier beschriebenen Timing-Problemen, die auch bei der

Virtualisierung alter Linux-Distributionen auftreten können, sind in Abschnitt 8.4 ab Seite 232 beschrieben.

virtio-Treiber

Standardmäßig kommen unter Windows 7 die folgenden virtuellen Hardware-Komponenten zum Einsatz:

- » eine CPU mit einem Core
- » ein RTL-8139-Netzwerkadapter im NAT-Netzwerk
- » eine IDE-Festplatte
- » eine Cirrus-kompatible VGA-Karte mit nahezu beliebiger Auflösung (bei unseren Tests bis zu 2560*1200 Pixel), wobei Windows aber anfänglich nur 800*600 Pixel nutzt

Um Netzwerk- und Festplattenzugriffe effizienter zu gestalten, sollten Sie nun unbedingt virtio-Treiber unter Windows installieren und anschließend die Hardware-Einstellungen der virtuellen Maschine entsprechend ändern.

Auf der folgenden Website finden Sie Download-Links für eine ISO-Datei mit signierten Treibern für alle gängigen Windows-Versionen von Windows XP bis Windows 7:

http://www.linux-kvm.org/page/WindowsGuestDrivers/Download_Drivers

Nachdem Sie die ISO-Datei auf das Hostsystem heruntergeladen haben, fahren Sie Ihren Windows-Gast herunter. Mit ANZEIGEN|DETAILS wechseln Sie in die Hardware-Ansicht der virtuellen Maschine. Dort geben Sie die ISO-Datei als Quelle für das virtuelle CD-Laufwerk an. Außerdem fügen Sie der virtuellen Maschine zusätzlich zu den vorhandenen Netzwerk- und Festplattenadaptern eine neue virtio-Netzwerkkarte und eine virtio-Festplatte hinzu. Die Image-Datei für die neue Festplatte muss nicht groß sein – es geht nur darum, dass Windows beim nächsten Start die neuen Hardware-Komponenten bemerkt.

Anschließend starten Sie den Windows-Gast neu und öffnen den Geräte-Manager (SYSTEMSTEUERUNG|HARDWARE UND SOUND|GERÄTE-MANAGER). Dort erscheinen die noch unbekannten Hardware-Komponenten als ETHERNET- und SCSI-CONTROLLER. Bei beiden Komponenten öffnen Sie nun per Doppelklick den Eigenschaftendialog, klicken auf EINSTELLUNGEN ÄNDERN, dann auf TREIBER AKTUALISIEREN und schließlich auf AUF DEM COMPUTER NACH TREIBERSOFTWARE SUCHEN. Bei dieser Suche müssen Sie mithelfen und geben als Ort der Treibersoftware das DVD-Laufwerk an (also üblicherweise D:).

Nun fahren Sie Windows herunter und entfernen in der Hardware-Übersicht des Virtual Machine Managers die IDE-Festplatte, die virtio-Festplatte und den RTL-8139-Netzwerkadapter. Außerdem richten Sie eine neue virtio-Festplatte ein, wobei Sie die

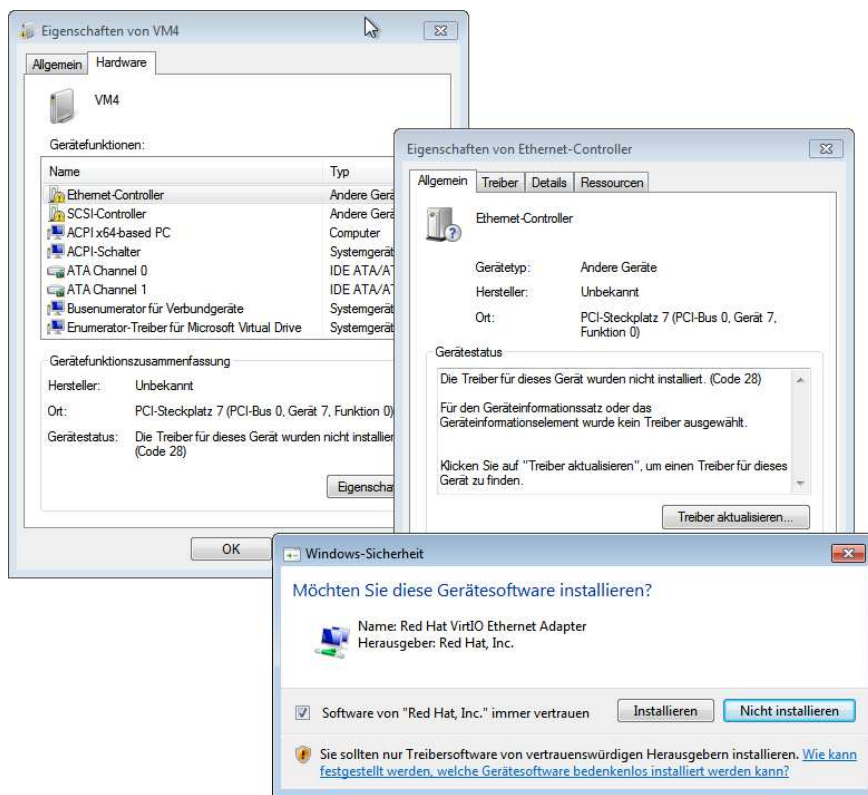


Abbildung 2.9: virtio-Treiberinstallation unter Windows 7

ursprüngliche Image-Datei auswählen (also die, die bisher mit der IDE-Festplatte verbunden war). Das Ergebnis ist eine virtuelle Maschine mit einem virtio-Netzwerkadapter und einer virtio-Festplatte. Unter Windows sollten Sie nochmals einen Blick in den Geräte-Manager werfen, der so wie in Abbildung 2.10 aussehen sollte.

Spice

Standardmäßig kommt in der virtuellen Maschine eine Cirrus-kompatible VGA-Grafikkarte zum Einsatz, deren Ausgaben via VNC im Virtual Machine Manager angezeigt werden. Das funktioniert auch in hohen Auflösungen zufriedenstellend. Lediglich auf die Aero-Effekte müssen Sie verzichten.

Wenn Sie auf eine höhere Grafikleistung Wert legen, empfiehlt sich der Einsatz des effizienteren Spice-Systems. Dazu müssen Sie zuerst unter Windows den Spice-QXL-Treiber installieren und dann im Virtual Machine Manager das Grafiksystem auf Spice umstellen. Im Detail ist der Umgang mit Spice in Kapitel 6 beschrieben. Erwarten Sie sich von

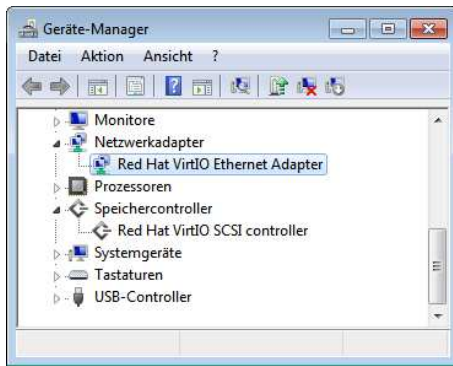


Abbildung 2.10: Die virtio-Geräte im Windows-Geräte-Manager

Spice aber keine Wunder! 3D-Funktionen oder Aero-Effekte stehen auch damit nicht zur Verfügung.

2.7 Fertige Appliances und vereinfachte Gast-Installation

Die manuelle Installation einer virtuellen Maschine geht zwar mit etwas Routine rasch von der Hand, aber bis das virtuelle System vollständig konfiguriert ist, alle Updates eingespielt sind etc., vergehen dennoch rasch ein, zwei Stunden. Diese Zeit können Sie sich mitunter sparen, wenn Sie einen der in diesem Abschnitt beschriebenen Wege beschreiben.

Virtuelle Maschine klonen

Im Virtual Machine Manager können Sie virtuelle Maschinen per Kontextmenü KLONEN. Die virtuelle Maschine muss dazu pausiert oder (noch besser) heruntergefahren werden. »Klonen« bedeutet, dass die Image-Datei mit dem Inhalt der virtuellen Festplatte dupliziert wird. Den Ort und Namen der neuen Image-Datei können Sie über das Dropdown-Menü DIESE FESTPLATTE KLONEN|DETAILS einstellen.

Außerdem werden die meisten Parameter der virtuellen Maschine übernommen und in einer neuen Konfigurationsdatei gespeichert. Ein Sonderfall ist der Netzwerkadapter: Er erhält beim Klonen eine neue MAC-Adresse, um Adresskonflikte im Netzwerk zu vermeiden.

Nach dem Klonen müssen Sie unter Umständen innerhalb des Gasts noch die Netzwerkkonfiguration anpassen (insbesondere dann, wenn die IP-Konfiguration statisch ist). Je

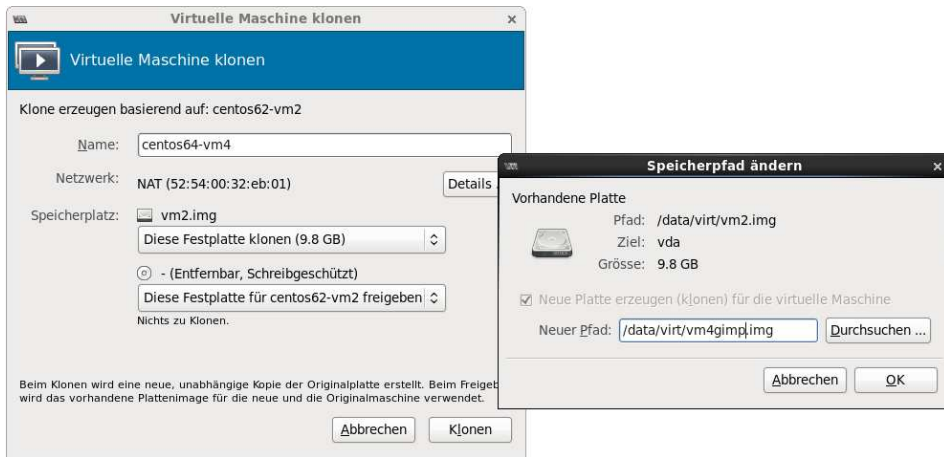


Abbildung 2.11: Eine virtuelle Maschine klonen

nach Distribution kann es auch sein, dass sich die virtuelle Maschine darüber beklagt, dass die Schnittstelle `eth0` fehlt. Der neue Netzwerk-Adapter erhält dann in der Regel den Device-Namen `eth1`.

Falls Sie im Gast einen SSH-Server laufen haben, müssen Sie aus Sicherheitsgründen dessen Schlüssel neu erzeugen. Wenn im Gast Fedora oder RHEL läuft, erreichen Sie das mit diesen Kommandos:

```
root# service sshd stop
root# rm /etc/ssh/ssh_host_*
root# service sshd start
```

Unter Debian oder Ubuntu führen Sie dazu diese Kommandos aus:

```
root# service ssh stop
root# rm /etc/ssh/ssh_host_*
root# dpkg-reconfigure openssh-server
```

Fertige Appliances herunterladen

Im Internet gibt es mehrere Sites, von denen Sie kostenlos oder gegen eine kleine Gebühr virtuelle Maschinen mit einer vorkonfigurierten Linux-Distribution herunterladen können. Zum Teil handelt es sich dabei einfach um eine Minimalinstallation für den Server-Einsatz, zum Teil sind auch Anwendungen vorinstalliert (z. B. ein Datenbank- oder Mail-Server). Es hat sich eingebürgert, derartige vorkonfigurierte virtuelle Maschinen als *Appliances* zu bezeichnen.

Die bekannteste derartige Seite ist <http://stacklet.com>. Auch auf den Homepages der wichtigsten Linux-Distributionen sowie anderer Open-Source-Produkte finden Sie Image-