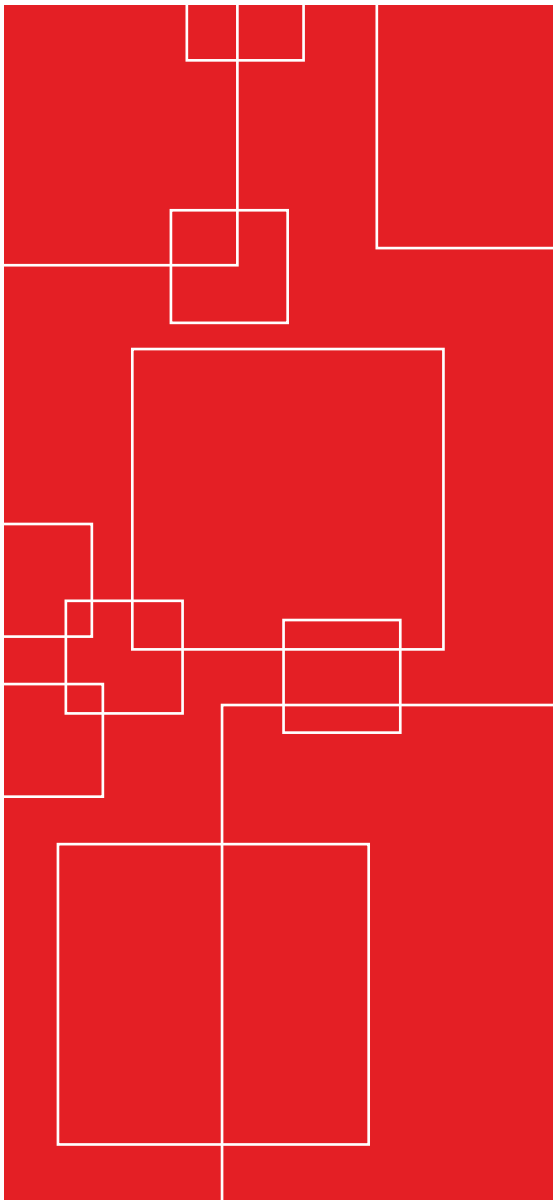
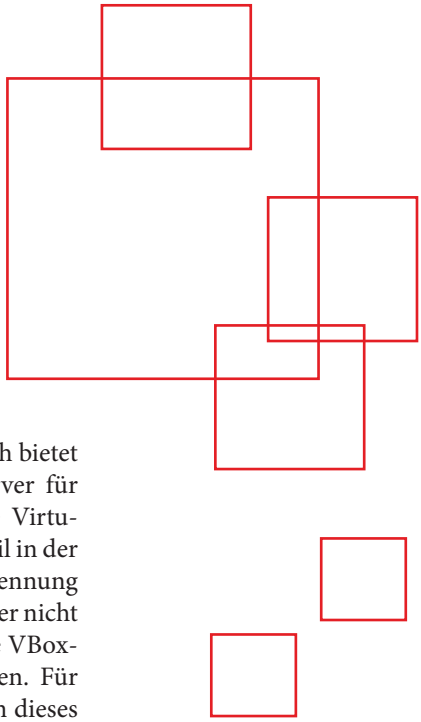
Virtualbox als Private Cloud



Virtualbox taugt als Hypervisor nicht nur für die Virtualisierung einzelner Hosts auf einem Desktop-Rechner, sondern kann sogar als Herzstück einer cloud-ähnlichen Umgebung fungieren. Dieser Beitrag zeigt, wie auf der Basis von VirtualBox unter openSUSE exemplarisch eine Private Cloud mit zwei Hosts aufzubauen ist. Michael Kromer



Der Begriff Cloud Computing erfährt momentan einen Hype. Inzwischen gibt es zahlreiche, teils kontrovers diskutierte Definitionen (1). In der Praxis erklärt man den Begriff oft mit der lokationsunabhängigen Bereitstellung von IT-Ressourcen in einer gesicherten Umgebung. Dazu bedarf es solcher Voraussetzungen wie Sicherheit, Verfügbarkeit, Verwaltbarkeit, Skalierbarkeit sowie Geschwindigkeit. Aus der Provider-Perspektive gehört auch die automatische Verrechenbarkeit der erbrachten Leistungen dazu.
Im Interesse der Skalierbarkeit trennt man den Storage in der Praxis häufig von den Virtualisierungshost und benutzt ein SAN. Virtualbox selbst bringt einen eigenen iSCSI-Initiator mit, so dass Gäste direkt von iSCSI booten können. Die hier vorgestellte Umgebung benutzt einen NFS-Mount, da so ziemlich jedes Speichernetz oder NAS das NFS-Protokoll anbietet.

API und Management

Verwaltbarkeit ist letztlich nur über die API des jeweiligen Hypervisors erreichbar. Es gibt bereits Produkte wie Abiquo (2), die sich auf Basis der APIs (3) diverser Hypervisor dem Management einer heterogenen Umgebung verschrieben haben. Abiquo unterstützt auch Virtualbox.
Daneben hat sich phpVirtualbox (4) in der Praxis als gute, clientunabhängige Konfigurationsoberfläche für einen Host erwiesen, auch wenn Sie bei weitem noch nicht alle Funktionen bietet, die VBoxManage (CLI) zur Verfügung stellt. Per ».htaccess«-Direktiven ist es möglich, die Anwendung auf definierbare Benutzer/Admins zu beschränken. phpVirtualbox kann man als Pedant zu der webbasierten Konsole eines VMWare ESX-Hosts verstehen.
Ein cloud-ähnliches Setup benötigt folgende Schritte:
1. Installation der notwendigen Komponenten
2. Vorbereitung des zentralen Storage und Anlage des VirtualBox-Benutzers
3. Erstellen von TAP-Devices zur Verwendung durch VirtualBox und OpenVPN
4. Erzeugen einer Bridge und Anschluß der TAP-Devices
5. Zusammenschluß der Bridges beider Hosts der private Cloud via OpenVPN
6. Konfiguration von VMs
7. Installation und Konfiguration von phpVirtualbox für das Management
8. optional: Definition von iptables-Regeln
9. optional: Konfiguration eines DHCP-Servers

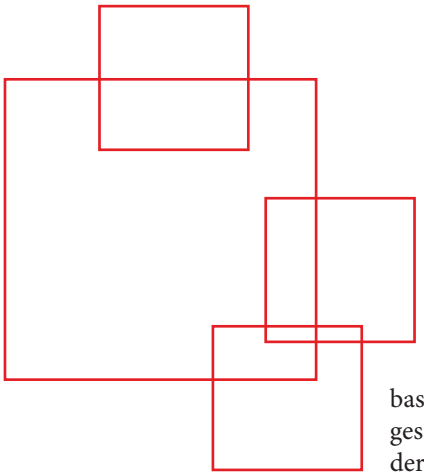
Nach Implementation dieser „Todo-List“ steht eine Umgebung zur Verfügung, die alle virtuelle Maschinen über beide Hosts verfügbar macht. Außerem lassen sich dann die Gäste zwischen beiden Hosts „teleportieren“.

Installation

Virtualbox-Instanzen laufen immer mit den Rechten desjenigen Benutzers, der die Instanzen erzeugt hat. Theoretisch bietet sich so die Möglichkeit, auf einem Server für verschiedene Benutzer unterschiedliche Virtualbox-Konfigurationen zu verwalten. Weil in der beispielhaften Cloud-Umgebung eine Trennung der Instanzen in unterschiedliche Benutzer nicht erwünscht ist, empfiehlt es sich dort, alle VBox-Befehle als Benutzer »vbox« auszuführen. Für die Softwareinstallation und das Anlegen dieses Users reichen dann zwei Zeilen:

```
zypper in http://download.virtualbox.org/➥
virtualbox/3.2.10/VirtualBox-3.2-3.2.10➥
_66523_openSUSE113-1.86_64.rpm ➥
brigde-utils openvpn
useradd -c "Virtualbox User" -gvboxusers ➥
-m -k /dev/null -d /vbox vbox
```

Die Nutzung eines öffentlichen IP-Pools ist bei einer Cloud oft unnötig, weil viele Dienste lediglich cloud-intern arbeiten sollen. Das Netzwerk der virtuellen Maschinen aller Hosts ähnelt damit einem normalen internen LAN. Damit jedoch alle virtuellen Maschinen in in diesem Subnetz über beide Hosts erreichbar sind, müssen sie durch den Admin per VPN (oder Routing) zusammengeschlossen werden. Nun gilt es TAP-Devices für VirtualBox als auch für OpenVPN zu erzeugen. Alle virtuellen Maschinen teilen sich dann die Schnittstelle »vbox0«, während OpenVPN mit »vpn0« eine Bridge zum anderen Virtualisierungshost aufbaut. OpenVPN empfiehlt sich hier, weil damit auch über unsichere Links die Sicherheit der Netzwerkkommunikation gewährleistet ist. Durch dieses Zusammenspiel von Bridges und TAP-Devices lässt sich eine Umgebung realisieren, in der jede virtuelle Instanz gezielt mit den gewünschten Systemen kommunizieren kann.
Die folgende Bridge-Konfiguration ist für den Einsatz auf einer der beiden Maschinen gedacht, die zweite ist mit der IP-Addresse »10.10.0.2/16« zu versehen. Da die Bridge per OpenVPN-Bridge zusammengeschlossen wird, ist der Ein-



satz einer eindeutigen IP-Addresse pro Host angeraten, auch wenn deren Angabe an sich technisch nicht erforderlich ist. Der Vorteil dieser auf OpenVPN-basierten Lösung ist, dass die gesamte Kommunikation der Gäste komplett transparent (samt Broadcast) auf beiden Host-Systemen verfügbar ist. Durch die Nutzung der Bridge lassen sich damit auch physikalische Interfaces zusätzlich einbinden, damit entsteht die eine universelle Netzwerk-Brücke für die physikalisch-virtuellen Bindungen dieser Cloud. Das Setup führt zu einem ganz ähnlichen Ergebnis wie man es in einer vSphere-Umgebung mit vSwitches und der Verbindung von vSwitches und Host-Interfaces erhält.

Zusammenschluss per OpenVPN

Das hier erzeugte Interface »vpn0« (**Listing 1**, **Abbildung 1**) stellt über die VPN-Bridge die Verbindung mit dem zweiten Host her. Wollte man noch mehr Hosts einbeziehen, böte sich der Einsatz der »server«-Direktive sowie der erweiterten Konfiguration eines IP-Pools an, denn die Anzahl der notwendigen VPN-Konfigurationsdateien mit [Anzahl Hosts]^2-[Anzahl Hosts], beliefe sich mit vier dezentralen Hosts bereits auf 12 verschiedene VPN-Konfigurationsdateien. Die minimale OpenVPN-Konfiguration (**Listing 2**) komprimiert per LZO den Datenverkehr, und belastet dank Fast-IO die CPU um 5 bis 10 Prozent weniger durch Schreibope-

Listing 1: Netzwerkkonfigurationen

```
01 »/etc/sysconfig/network/ifcfg-vbox0:«
02 STARTMODE='onboot'
03 BOOTPROTO='static'
04 TUNNEL='tap'
05 TUNNEL_SET_OWNER='vbox'
06
07 »/etc/sysconfig/network/ifcfg-vpn0:«
08 STARTMODE='onboot'
09 BOOTPROTO='static'
10 TUNNEL='tap'
11
12 »/etc/sysconfig/network/ifcfg-br0:«
13 BOOTPROTO='static'
14 BRIDGE='yes'
15 BRIDGE_AGEINGTIME='15'
16 BRIDGE_PORTS='vbox0 vpn0'
17 BRIDGE_STP='off'
18 STARTMODE='onboot'
19 USERCONTROL='no'
20 IPADDR='10.10.0.1/16'
```


rationen. Das in der VPN-Konfigurationsdatei verwendete OpenVPN-Secret wird einmalig mittels »openvpn --genkey --secret /etc/openvpn/bridgekey« erzeugt und danach auf den anderen Host übertragen.

Nach den Änderungen aus **Listing 2** aktiviert man den gesamten Bridge/TAP-Netzwerk-Stack:

```
#/etc/init.d/network start vpn0 && /etc/➥
init.d/network start vbox0 && /etc/init.d/➥
network start br0
#/etc/init.d/openvpn start
```

Indem man die interne Bridge-Addresse des anderen Hosts per ICMP (»ping«) testet, lässt sich sicherstellen, dass die Kommunikation der beiden Bridges »br0« über die »vpn0«-Interfaces korrekt funktioniert.

Erzeugen eines Gast-Systems

Mit Virtualbox lässt sich der Gast vollkommen per Kommandozeile (oder via API) erzeugen. Virtualbox bietet seinen Gästen durch sein Feature „Teleport“ die Möglichkeit, virtuelle Maschinen von einem Host zu einem anderen zu übertragen. Dieses Verhalten ähnelt der Funktionalität von VMotion in der vSphere-Welt, und erlaubt es, virtuelle Maschinen auch im eingeschalteten Zustand zu bewegen (zu „teleportieren“).

Listing 3 erzeugt eine eine virtuelle Machine mit einer Festplatte auf einem gemounteten NFS-Filesystem (»/nfs«) und bereitet sie für den Start vor. Bei der Nutzung von Teleport ist eine nahezu identische Konfiguration der virtuellen Maschine notwendig, auch wenn sich die Hosts in Ausstattung oder gar Betriebssystem unterscheiden. Also sind bis auf das Erzeugen der Festplatte selbst (»createhd«) alle Befehle auf

allen Hosts durchzuführen. Es empfiehlt sich der Einsatz möglichst identischer Prozessoren, damit ein Teleport erfolgreich und ohne Performance-Einbußen ausgeführt werden kann.
Mit der option »--synthcpu« lässt sich für den Gast ein „synthetischer" Prozessor emulieren, der notwendig wird, wenn sich die eingesetzten Host-Prozessoren zu stark unterscheiden (wenn sie etwa drei Prozessorgenerationen oder mehr trennen). Mittels »VBoxManage modifyvm *GAST* --cpuidset« lässt sich gezieltes CPU-Masking realisieren, um eine durchgängige CPU-Unterstützung aller Hosts trotz unterschiedlicher Prozessoren zu ermöglichen. Die vorliegende Konfiguration bootet bevorzugt per Netzwerk (PXE), alternativ ließe sich dafür auch ein ISO-Image zur Installation verwenden (mit »--boot1 dvd«).

Teleporting

Sind alle virtuellen Maschinen auf allen Hosts erzeugt und ist überall dieselbe Konfiguration verfügbar, ist ein Teleporting möglich. Nun lässt sich eine virtuelle Instanz mittels »VBoxHeadless -s guest1« auf einem beliebigen Host starten. Das Teleporting funktioniert mittels »VBoxManage modifyvm "guest1" --teleporter on --teleporterport 6000« und dem darauffolgendem Start per »VBoxHeadless -s guest1 &« . Nun kann der Admin im Bedarfsfall die virtuelle Instanz per »VBoxManage controlvm "guest1" teleport --host *IP des Host* --port 6000« auf den Host transferrieren, der auf dem Teleporter-Port lauscht.

Monitoring und VDI

Virtualbox bietet mit den Metrics die Möglichkeit, genaue Last- und Nutzungsdaten einer Instanz abzurufen. Nach der Installation muss der Admin sie mittels »VBoxManage metrics setup« aktivieren. Danach lassen sich Metrics per »VBoxManage metrics query« abrufen. So lassen sich die genutzten Ressourcen weiterberechnen. Ganz nebenbei lassen sich diese Nutzdaten auch in beliebige Monitoring-Systeme wie etwa Nagios oder Munin einbinden.
Mit der erweiterten RDP-Implementation von Virtualbox gibt es die Möglichkeit USB-Geräte per RDP weiterzuleiten. Mit dem mitgelieferten »rdesktop-vrdp« kann damit jeder Linux-Client auf Aforderung die angesteckten USB-Geräte direkt an den Gast der virtualisierten Umgebung exportieren. Mit »rdesktop-vrdp -xb -a16 -r usb -r sound:local« ist ein voll ausgestatteter Desktop auch über schmalere Verbindungen mit Sound und USB nutzbar.

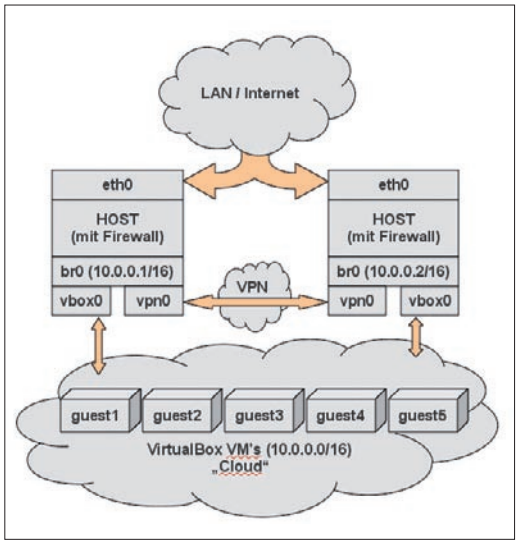


Abbildung 2: Virtualbox als Private Cloud – ein Überblick über die Architektur der Lösung.

TAP-Devices und Kernel 2.6.26

Mit der Kernel-Version 2.6.26 hat sich das Handling von TAP-Devices sich aus Applikationssicht erheblich geändert. Bis zur Virtualbox-Version 3.2.10 war ein Patch (5) notwendig, der dafür sorgte, dass ein TAP-Device von Virtualbox zunächst geöffnet wurde, um dann erst Pakete darüber zu schicken. Ohne diesen Patch war das Bridging mit dem hier verwendeten TAP-Interface »vbox0« nicht möglich, da der Kernel alle Schreiboperationen auf das Interface schlichtweg verwarf.

Listing 2: »vpn0.conf«

```
01 ;IP des gegensätzlichen Hosts
02 remote 1.2.3.4
03 proto udp
04 port 1194
05 dev vpn0
06 secret /etc/openvpn/bridgekey
07 comp-lzo
08 comp-noadapt
09 persist-key
10 persist-tun
11 fast-io
```

Der Autor

Michael Kromer ist Senior Consultant und Linux Engineer bei der Millenux GmbH in München. Er entwickelt an diversen Open-Source-Projekten mit, zum Beispiel dem Linux-Kernel, OpenNX, Asterisk, VirtualBox und ISC Bind. Seine privaten Leidenschaften sind openSUSE, wo er „Member" ist, und Virtualisierungstechnologien.

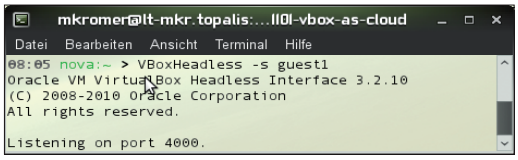


Abbildung 3: Ein auf RDP-Port 4000 lauschender VBox-Prozess, lauschend auf Port 6000 (Teleport-Vorbereitung).

phpVirtualbox und Best Practice

Damit phpVirtualbox die Möglichkeit hat, mit Virtualbox zu sprechen, muss der mitgelieferte Dienst »vboxwebsrv« mit den Rechten des Benutzers laufen, der die virtuellen Instanzen steuert (»vboxwebsrv -b«). phpVirtualbox liefert auf seiner Download-Seite ein Beispiel-Skript. Für die Nutzung von phpVirtualbox wird Apache2 vorausgesetzt. Mit »zypper in apache2 apache2-mod_php5 php5-json php5-soap« sind die notwendigen Abhängigkeiten erfüllt. Die Authentifikation der Web Services kann die bestehende Authentifikation von VRDP mitnutzen, also wahlweise PAM oder LDAP (vrdp-ldap). Mit »VBoxManage setproperty websrvauthlibrary null« lässt sich diese gänzlich deaktivieren. Praktischerweise liefert phpVirtualbox einen flash-basierten RDP-Viewer mit, der es ermöglicht die Konsole der virtuellen Instanz plattformunabhängig zu betrachten.

In diversen Benchmarks hat sich in den aktuellen Virtualbox 3.x-Versionen »virtio-net« als Performance-Bremse erwiesen. Da die Virtio-Implementation erst in der kommenden Virtualbox 4.0 komplett erneuert sein wird, ist bis dahin der Einsatz der E1000-basierten Netzwerkkarten im Gast anzuraten. Darüber hinaus empfiehlt es sich, das Management der virtuellen Instanzen mit einem DHCP-Server zu vereinfachen, der auf dem Bridge-Interface (br0) aktiv ist.

Listing 3: Einen Gast erzeugen (Beispiel)

```
01 #su - vbox
02 ##Erstellung und Registrierung des Gasts
03 #VBoxManage createvm --name "guest1" --ostype OpenSUSE
   --register
04 ##Einstellungen des Gastes (RAM, CPU, Netz, etc.)
05 #VBoxManage modifyvm "guest1" --cpus 2 --memory 1024
   --nestedpaging on --acpi on
06 --ioapic on --hpet on --hwvirtex on --hwvirtexexcl on
   --nic1 bridged --nictype1
07 82545EM --bridgeadapter1 vbox0 --boot1 net --boot2 disk
   --rtcuseutc on --usb off
08 --audio none --clipboard disabled --keyboard ps2 --mouse
   ps2 --vrdp on
09 --vrdpport 4000 --vrdpmulticon on --cableconnected1 on
   --pagefusion on
10 ##Erstellen eines Festplatten-Images auf zentralem NFS
11 #VBoxManage createhd --filename "/nfs/guest1.vdi" --size
   10000 --remember
12 ##Erstellen eines SATA-Controllers für den Gast
13 #VBoxManage storagectl "guest1" --name "SATA Controller"
   --add sata --controller
14 IntelAHCI
15 ##Zuweisung des Festplatten-Images zum Gast
16 #VBoxManage storageattach "guest1" --storagectl "SATA
   Controller" --port 0
17 --device 0 --type hdd --medium /nfs/guest1.vdi
18 ##(Optionales) Einhängen eines ISO-Images
19 #VBoxManage storagectl "guest1" --name "IDE Controller"
   --add ide --controller
20 PIIX4
21 #VBoxManage openmedium dvd /nfs/image.iso
22 #VBoxManage storageattach "guest1" --storagectl "IDE
   Controller" --port 0
23 --device 0 --type dvddrive --medium /nfs/image.iso
```

Fazit

Virtualbox als Hypervisor lässt sich mit den momentan verfügbaren Mitteln zu einer Cloud beliebiger Größe ausbauen, da es eine vollständige API zur Kontrolle von Gast-Instanzen mitbringt. Die notwendigen Komponenten zum Betrieb einer eigenen Cloud lassen sich beliebig erweitern, so ist der Einsatz von Pacemaker eine durchaus valide Lösung zur Verwaltung von Instanzen, um auch im Fehlerfalle Gäste am laufen zu halten. Ein Resource Agent für Virtualbox befindet sich bereits in der Entwicklung um die VBoxManage-Events wie etwa »teleport« oder »savestate« im Cluster zu unterstützen. ■■■

Infos

(1) Larry Ellison über Clouds: (http://www.youtube.com/watch?v=UOEFXaWHppE)

(2) Cloud-Manager Abiquo: (http://www.abiquo.com/products/abiquo-overview.php?lang=de)

(3) VirtualBox API: (http://www.virtualbox.org/sdkref/index.html)

(4) PHPVirtualbox: (http://code.google.com/p/phpvirtualbox)

(5) Ticket zu TAP-Devices: (http://www.virtualbox.org/ticket/7649)