Installation eines Linux–Basissystems in einer Virtual Machine

Ideen, Aspekte, Anregungen, Diskussion

Dirk Geschke

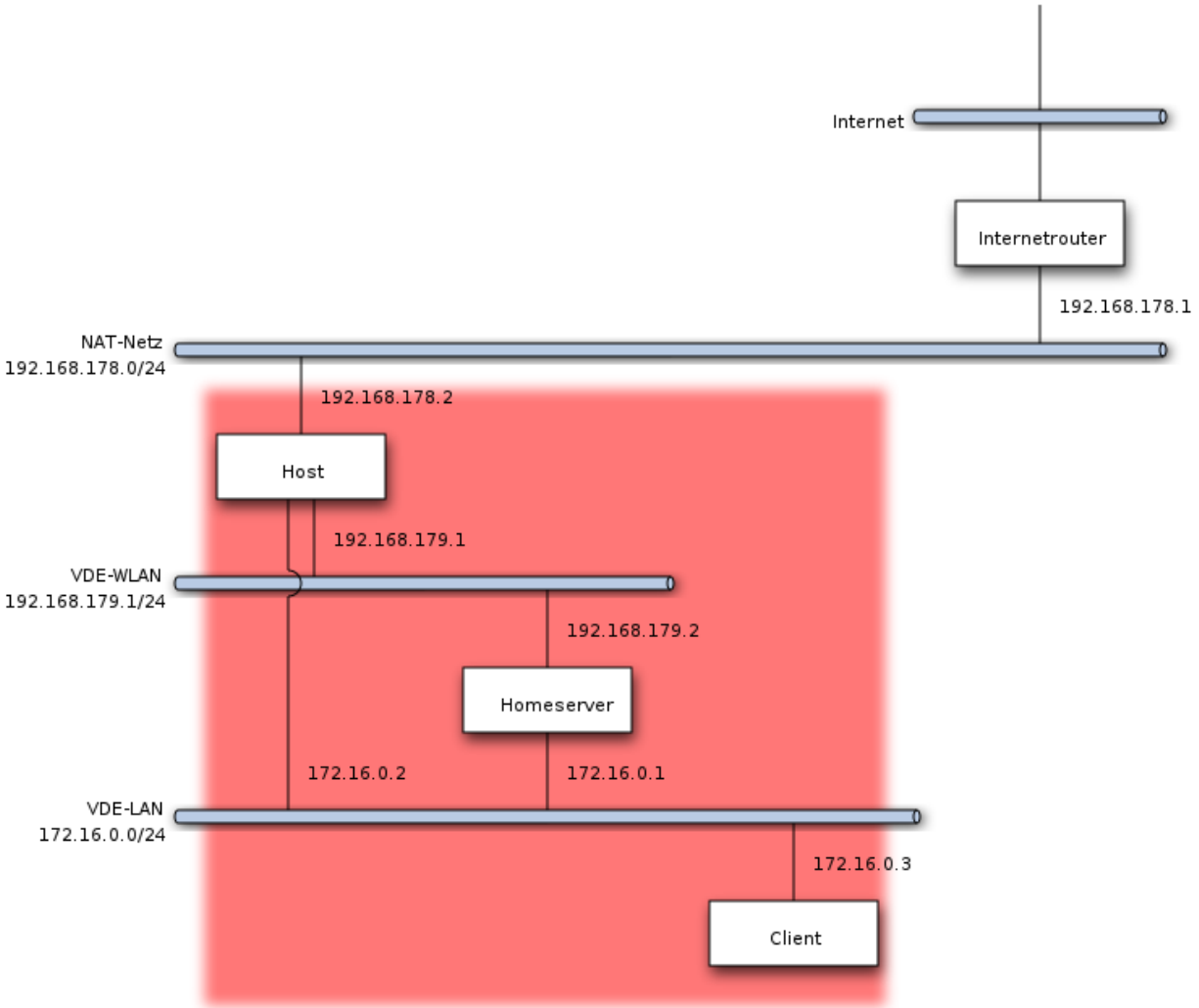
Linux User Group Erding

24. April 2013

Gliederung

Allgemeines

- erst einmal Minimalinstallation eines Debian–Servers (wheezy)
- nach und nach Ausbau der Serverdienste
- Installation in einer VM $\longrightarrow$ KVM.
- Verwendung von Snaphots für Zwischenstände
- virtuelle Vernetzung per **VDE** (optional)
- gleich zwei Netzwerkkarten einplanen


Internet
Internetrouter
192.168.178.1
NAT-Netz
192.168.178.0/24
192.168.178.2
Host
192.168.179.1
VDE-WLAN
192.168.179.1/24
192.168.179.2
Homeserver
172.16.0.2
172.16.0.1
VDE-LAN
172.16.0.0/24
172.16.0.3
Client

Zu den Namen

Host Das ist der PC/Laptop auf dem die KVM-Instanz läuft.

VM Virtuelle Instanz (*Virtual Machine*), sie läuft unter `kvm` auf dem Host.

tap Über das TAP-Interface kann eine VM mit dem Host vernetzt werden

VDE Virtual Distributed Ethernet: Das ist ein virtueller Switch zum vernetzen von VMs und auch dem Host.

Vorbereitungen: Installation kvm & Co.

- `apt-get install qemu-kvm`
- bei Verwendung von VDE: `apt-get install vde2`
- optional: `apt-get install dnsmasq`
- sicherstellen, dass man in der Gruppe `kvm` ist:
  `$ grep kvm /etc/group`
- wenn nicht, einfach als `root` hinzufügen, z.B.:
  `# adduser geschke kvm`

Verwendung von VDE

- Start der zwei VDE-Switche:

```
# vde_switch -sock /tmp/vde-wan -mgmt \
/tmp/vdecfg-wan -mod 770 -group kvm \
-daemon -tap tap0
# vde_switch -sock /tmp/vde-lan -mgmt \
/tmp/vdecfg-lan -mod 770 -group kvm \
-daemon -tap tap1
```

- Konfiguration der TAP-Interfaces

```
# ifconfig tap0 192.168.179.1 \
netmask 255.255.255.0 up
# ifconfig tap1 172.16.0.2 \
netmask 255.255.255.0 up
```

Verwendung von VDE

- Weiterleitung der IP-Pakete aktivieren

```
echo "1" > /proc/sys/net/ipv4/ip_forward
```

- Bei Verwendung einer Firewall das Weiterleiten erlauben:

```
# iptables -A FORWARD -j ACCEPT
```

- Optionales Masquerading aktivieren `eth0/wlan0`

```
# iptables -t nat -A POSTROUTING \
-o eth0 -j MASQUERADE
# iptables -t nat -A POSTROUTING \
-o wlan0 -j MASQUERADE
```

Verwendung von VDE

- Bei Verwendung einer Firwall mit IPtables: Input erlauben

```
# iptables -A INPUT -i tap0 -j ACCEPT
# iptables -A INPUT -i tap1 -j ACCEPT
```

- Die IPtables-Regeln sind alles andere als optimal!
- gewöhnlich ist ausgehender Traffic erlaubt!
- Adressen:

WAN	192.168.179.2
Gateway	192.168.179.1
LAN	172.16.0.1

DNSmasq

- Verwendung ist optional, erleichtert aber ein paar Einstellungen
- Konfigurationsdatei: `/etc/dnsmasq.d/tap.conf`

```
interface=tap0
listen-address=192.168.179.1
dhcp-range=192.168.179.2,192.168.179.24,12h
dhcp-host=52:54:00:12:34:56,homix,192.168.179.2
dhcp-option=option:router,192.168.179.1
dhcp-option=option:ntp-server,192.168.179.1
```

Vorgehen zur Vorbereitung

1. Installieren von KVM, VDE und dnsmasq
2. User der Gruppe `kvm` hinzufügen
3. Abspeichern der Datei `tap.conf` in `/etc/dnsmasq.d/`
4. Ausführen des Skriptes `root-homeserver.sh` als root

Alternative ohne VDE

- `tap`-Interfaces über `kvm`–Aufruf anlegen lassen
- `kvm`-Aufruf muss vorher erfolgen. Konfiguration der `tap`–Interfaces über KVM-Skripte: `tap0.sh` und `tap1.sh`
- Konfiguration von IPtables wie oben, wird mit `tap`–Skripte ausgeführt
- `kvm`–Aufruf muss nun aber als `root` erfolgen!

Erste Schritte

- eigenes Verzeichnis: `mkdir homeserver`
- Wechsel dahin: `cd homeserver`
- Download des Install-ISO-Images von Debian-wheezy:
  `debian-testing-amd64-netinst.iso`
- Anlegen einer KVM-Datei:
  `qemu-img create -f qcow2 homeserver.kvm 8G`
- Format `qcow2` erlaubt Snapshots
- Keine Sparse–Datei!

Installation mit VDE

- Der `kvm`–Aufruf sieht etwas kryptisch aus:

```
$ kvm -enable-kvm -m 256m -k de \
-net vde,vlan=0,sock=/tmp/vde-wan \
-net nic,vlan=0,model=virtio \
-net vde,vlan=1,sock=/tmp/vde-lan \
-net nic,vlan=1,model=virtio \
-drive file=homeserver.kvm,if=virtio \
-cdrom debian-testing-amd64-netinst.iso \
-boot once=d
```

Details

-enable-kvm Aktiviere die Hardwareunterstützung

-m 256m Stelle 256 MB der VM zur Verfügung

-k de Verwende eine deutsche Tastatur

-net vde,vlan=0,sock=/tmp/vde-wan Verlinke das `vlan 0` mit dem VDE–Switch via dem Socket `/tmp/vde-wan`

-net nic,vlan=0,model=virtio Die erste Netzwerkkarte der VM wird mit `vlan 0`, also dem WAN–Switch verbunden. Als Kartentyp wird `virtio` verwendet.

-drive file=homeserver.kvm,if=virtio Hier wird unsere angelegte KVM–Datei als `virtio`–Laufwerk festgelegt.

-cdrom debian-testing-amd64-netinst.iso Dies bindet das ISO-Image als CD ein

-boot once=d Boote nur einmal von der CD, danach von HD

Installation ohne VDE

- Der `kvm`–Aufruf sieht ebenfalls kryptisch aus:

```
# kvm -enable-kvm -m 256m -k de \
-net tap,vlan=0,ifname=tap0,script=tap0.sh \
-net nic,vlan=0,model=virtio \
-net tap,vlan=1,ifname=tap1,script=tap1.sh \
-net nic,vlan=1,model=virtio \
-drive file=homeserver.kvm,if=virtio \
-cdrom debian-testing-amd64-netinst.iso \
-boot once=d
```

Abweichende Details

-net tap,vlan=0,ifname=tap0,script=tap0.sh Dieser Aufruf legt das TAP–Interface `tap0` an. Anschließend wird für diese Konfiguration das Skript tap0.sh aufgerufen. Dieses konfiguriert das `tap0`–Interface des Hosts und aktiviert das Forwarding sowie das Masquerading.

-net tap,vlan=1,ifname=tap1,script=tap1.sh Analoges Vorgehen für das zweite TAP–Interface, Forwarding und Masquerading müssen aber hier nicht mehr aktiiviert werden.

tap0.sh–Skript

```
#/bin/sh
ifconfig tap0 192.168.179.1 \
netmask 255.255.255.0 up
iptables -A INPUT -i tap0 -j ACCEPT
echo "1" > /proc/sys/net/ipv4/ip_forward
iptables -t nat -A POSTROUTING -o eth0 \
-j MASQUERADE
iptables -t nat -A POSTROUTING -o wlan0 \
-j MASQUERADE
iptables -A FORWARD -j ACCEPT
test -x /etc/init.d/dnsmasq && \
/etc/init.d/dnsmasq restart
```

Minimalinstallation

- Install auswählen
- Language: German
- Land : Deutschland
- Tastatur: Deutsch
- eth0 auswählen
- ohne DHCP auf Timeout warten oder abbrechen

Minimalinstallation ohne DHCP

- Netzwerk manuell einrichten
  - IP-Adresse 192.168.179.2
  - Netzmaske 255.255.255.0
  - Gateway 192.168.179.1
  - DNS-Server Wert aus /etc/resolv.conf vom Host
  - Rechnername homix
  - Domain-Name netzwerk.daheim
- Mit DNSmasq werden die meisten Werte automatisch gesetzt!

Fortsetzung Minimalinstallation

- Root-Passwort: ich nehme zum Testen meistens toor"
- Benutzer: lug
- Benutername für Ihr Konto: lug
- Passwort: toor
- Partitionierung
  - Geführt - vollständige Festplatte verwenden
  - SCSI1 (0,0,0) (sda) auswählen
  - Alle Dateien auf einer Partition
  - Partitionierung beenden und Änderungen Übernehmen
  - Änderungen auf Festplatten schreiben? <Ja> (Default ist <Nein>!!!)

Fortsetzung Minimalinstallation

- Installieren des Grundsystems
- Land des Debin-Archiv-Spiegelservers: Deutschland
- Debian-Archiv-Spiegelserver: `ftp.de.debian.org`
- HTTP-Proxy-Daten, sofern benötigt: `http://`*proxy*`:`*port*`/`
- Paketverwendungserfasung <Nein>
- Softwareauswahl:
  [*] `SSH server`
  [*] `Standard Systemwerkzeuge`
- Rest kommt später einmal $\longrightarrow$ erst nur minimale Basis.
- Boot und gut!

Weiteres Vorgehen

- entfernen der CD beim KVM-Aufruf
- Umstellung `grub` auf `serial`
- Umstellung `console` auf `serial`
- Umlenkung von Monitor- und Serial-Port auf `localhost`-Ports/TCP
- Starten von KVM ohne Monitor. Damit brauchen wir dann kein X11-Fenster mehr.
- Verwendung von `-daemonize` um im Hintergrund zu werkeln

Umstellung auf serielle Konsole

- Editieren von `/etc/default/grub`
- Hinzufügen von 2 Zeilen:

```
GRUB_TERMINAL=serial
GRUB_SERIAL_COMMAND="serial -speed=9600 \
-unit=0 -word=8 -parity=no -stop=1"
```

- Ändern von 2 Zeilen in:

```
GRUB_CMDLINE_LINUX_DEFAULT="console=tty0 \
console=ttyS0,9600n8"
GRUB_CMDLINE_LINUX="console=tty0 \
console=ttyS0,9600n8"
```

- Damit bleibt auch der Montior `tty0` noch verwendbar

Umstellung auf serielle Konsole

- Aktuelle `grub`–Konfiguration installieren: `update-grub`
- Anpassung `/etc/inittab`:

  ```
  T0:23:respawn:/sbin/getty -L ttyS0 9600 vt100
  ```

- Anhalten des Systems: `halt`

Anpassung KVM, hier mit VDE

```
$ kvm -enable-kvm -m 256m -k de \
-serial telnet:127.0.0.1:10000,server \
-monitor telnet:127.0.0.1:10001,server,nowait \
-net vde,vlan=0,sock=/tmp/vde-wan \
-net nic,vlan=0,model=virtio \
-net vde,vlan=1,sock=/tmp/vde-lan \
-net nic,vlan=1,model=virtio \
-drive file=homeserver.kvm,if=virtio \
-display none -daemonize
```

Details

-serial telnet:127.0.0.1:10000,server Umlenkung der ersten seriellen Schnittstelle auf den TCP-Port 10000. Dieser ist vom Host über 127.0.0.1 erreichbar, es wird das `telnet`–Protokoll gesprochen. Es wird mit dem booten gewartet bis eine Verbindung auf diesem Port eingegangen ist

-monitor telnet:127.0.0.1:10001,server,nowait Analog wird die Qemu-Shell auf den `telnet`–Port 10001 umgelenkt. Option `nowait` bewirkt, dass hier zum Starten keine Verbindung notwenig ist.

-display none Es wird kein X11-Fenster geöffnet

-daemonize Die VM läuft als Daemon im Hintergrund

Snaphots

- Snapshots können nun über die Qemu-Shell erstellt werden:

```
# telnet 127.0.0.1 10001
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
QEMU 1.1.2 monitor - type 'help' for more inform
(qemu) savevm fresh_installed
```

- Ansehen der Snapshots:

```
(qemu) info snapshots
ID TAG VM SIZE DATE ...
1 fresh_installed 109M ...
```

Testen des Snaphots

- Anlegen einer Datei:

```
root@homix:~# ls
root@homix:~# touch Eine_Datei
root@homix:~# ls
Eine_Datei
```

- Restoren des Snapshots

```
(qemu) loadvm fresh_installed
```

- und nachsehen:

```
root@homix:~# ls
root@homix:~#
```

- scheint zu funktionieren!

Praxis!