



http://ovm-kassel.de Lernjob	
Lernjob IT-BS-VM-LJ-3.2 Netzwerke mit Virtualbox	
Code	IT-BS-VM-LJ-3.2
Autor	André Bauer <a(dot)bauer(at)ovm-kassel(dot)de>
Datum	22. Mai 2018
Links	<ul style="list-style-type: none"> • Netzwerkkonfiguration in VirtualBox • Oracle VM VirtualBox User Manual • Oracle VM VirtualBox User Manual — Chapter 6. Virtual Networking • VirtualBox: Netzwerkkonfiguration • ip command cheat sheet • Buch: Everything curl • Deprecated Linux networking commands and their replacements • IP Route Management
Verwandte Literatur	<ul style="list-style-type: none"> • IT-BS-VM-LJ-3.1 Dokumentieren mit Script und AsciiDoc • IT-BS-VM-LJ-3.3 SSH-Client und -Server • IT-BS-VM-LS-3.1 Netzwerke mit Virtualbox
Lizenz	 Dieses Werk ist lizenziert unter einer Creative Commons Namensnennung - Weitergabe unter gleichen Bedingungen 4.0 International Lizenz .

Netzwerke mit Virtualbox

Wie können virtuelle Netzwerke zwischen Virtuellen Maschinen (VMs) aufgebaut und mithilfe von Shell-Befehlen getestet werden?

Virtualbox bietet sechs verschiedene Modi an, wie ein Netzwerkadapter verbunden werden kann, die bis auf den sechsten im Folgenden vorgestellt und beispielhaft verwendet werden:

1. NAT
2. Internes Netzwerk

3. NAT-Netzwerk
4. Netzwerkbrücke
5. Host-only-Adapter
6. Nicht angeschlossen



Im Fachgespräch wird von Ihnen erwartet, dass Sie die Konfiguration der verschiedenen Netzwerk-Modi und den Einsatz der Test-Werkzeuge auch vorführen können.

1. NAT mit Port-Weiterleitung

Bei der Einstellung NAT erhält der Adapter des Gasts per NAT (NAT steht für Network Address Translation) Zugriff auf das Netzwerk, an das der Host angeschlossen ist, der Gast kann aber umgekehrt von außen oder vom Hostsystem nicht erreicht werden.

In den VM-Einstellungen richtet man unter Netzwerk bei einem Adapter mit der Einstellung NAT unter „Erweitert“ eine Portweiterleitung ein. Zum Beispiel eine Weiterleitung des Host-Ports 8080 auf den Gast-Port 8000, womit der Gast auf diesem Port erreichbar wird.

Mit Ruby kann man ohne vorherige Konfiguration einen Webserver (WEBrick-HTTP-Server) in einer Virtuellen Maschine (VM) betreiben. Falls Ruby noch nicht installiert sein sollte, kann dies unter debian-basierten Systemen wie Ubuntu mit den folgenden beiden Befehlen eingerichtet werden:

Quellcode 1. Auf dem Gastsystem

```
$ sudo apt update  
$ sudo apt install ruby
```

Nun kann ein Verzeichnis `srv` mit einer Datei `index.html` angelegt werden und mit Ruby ein Webserver gestartet werden.

Quellcode 2. Auf dem Gastsystem

```
$ sudo hostname saturn ①  
$ mkdir srv  
$ echo "Hello from `hostname`." > srv/index.html ②  
$ ruby -run -e httpd srv -p 8000 ③
```

① Setzt den Hostnamen auf `saturn`.

② Erstellt eine einfache Textdatei im Ordner `srv`.

③ Startet WEBrick auf Port 8000 mit `srv` als Wurzelverzeichnis (Documentroot).

Die Portweiterleitung kann mit `curl` auf dem Hostsystem getestet werden. Das Programm ist für **alle gängigen Betriebssysteme** verfügbar. In dem Buch „Everything curl“ ist eine umfassende Dokumentation des Befehls `curl` enthalten. Unter debian-basierten Systemen kann curl mit

```
$ sudo apt update
$ sudo apt install curl
```

installiert werden.

Quellcode 3. Auf dem Hostsystem

```
$ curl localhost:8080
Hello from saturn.
```

Eine zweite Möglichkeit ist, die Adresse <http://localhost:8080> in einem Browser auf dem Hostsystem aufzurufen.

Die Portweiterleitung ist aber nicht nur lokal über die Loopback-Schnittstelle gültig, sondern auch über die Netzwerkschnittstelle des Hosts, die mit dem physischen Netzwerkadapter verknüpft ist. Diese kann mit `ipconfig` (Windows), `ifconfig` (Mac OS X) oder `ip -4 addr` (Linux) ermittelt werden.

Quellcode 4. Auf dem Hostsystem

```
$ ip -4 addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
2: enp0s25: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    inet 192.168.178.21/24 brd 192.168.178.255 scope global enp0s25
        valid_lft forever preferred_lft forever

$ curl 192.168.178.21:8080
Hello from saturn.
```

Quellcode 5. Auf einer zweiten Linux-VM oder dem Hostsystem

```
$ curl 192.168.178.21:8080
Hello from saturn.
```

Aufgabe 1



Dokumentieren Sie Ihr Vorgehen mit Script und AsciiDoc und stellen Sie das Ergebnis als PDF-Datei in Ihrem E-Portfolio zur Verfügung.

- a. Richten Sie bei einer Linux-VM NAT mit einer Port-Weiterleitung ein und erstellen Sie wie in [Quellcode 2](#) eine Datei, die sie mit WEBrick zugänglich machen. Wählen Sie Port-Nummern, die von dem Beispiel und anderen Quellen abweichen.
- b. Testen Sie mit curl die Port-Weiterleitung von
 - Ihrem Hostsystem und
 - einer zweiten Linux-VM.

2. Internes Netzwerk

Ein internes Netzwerk ist ein virtuelles Netzwerk, an dem ausschließlich Virtuelle Maschinen angebunden werden können. Im VirtualBox Manager wird dazu zwei oder mehreren *ausgeschalteten* VMs ein zusätzlicher Netzwerk-Adapter mit demselben internen Netzwerknamen hinzugefügt. Im Gegensatz zu der Standardoption NAT erhalten die Schnittstellen zunächst keine IP-Adressen, so dass diese entweder manuell vergeben werden müssen (siehe unten) oder ein [DHCP-Server](#) für dieses interne Netz eingerichtet werden muss. Insofern eignet sich dieser Modus besonders, um z. B. die Einrichtung eines DHCP-Servers oder eines Domänencontrollers in einem virtuellen Netzwerk auszuprobieren und zu testen.

Im folgenden werden zwei Linux-VMs benötigt, dazu kann z. B. eine bestehende, ausgeschaltete VM im Virtualbox Manager geklont werden.

Die Systeme haben nun eine zusätzliche Netzwerkschnittstelle, der noch keine IPv4-Adresse zugeordnet wurde. Um Informationen über Netzwerkschnittstellen zu erhalten und diese zu konfigurieren, gibt es unter unix-artigen Systeme u. a. den Befehl `ip`. Unter Windows gibt es den Befehl `ipconfig`, der eine ähnliche Funktionalität hat. Im Beispiel hat die neue Schnittstelle den Namen `enp0s8`.

Beispiel 1. Auf Gast 1

```
$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo ①
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
group default qlen 1000
    link/ether 08:00:27:07:74:c5 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic enp0s3 ②
        valid_lft 86186sec preferred_lft 86186sec
    inet6 fe80::3bbb:2d8e:1e5:dea6/64 scope link
        valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
group default qlen 1000
    link/ether 08:00:27:f3:87:36 brd ff:ff:ff:ff:ff:ff ③
```

- ① Die Loopback-Schnittstelle erhält per Konvention die IPv4-Adresse **127.0.0.1/8** sowie die IPv6-Adresse **::1/128**.
- ② Der Schnittstelle **enp0s3** ist die IP-Adresse **10.0.2.15** zugeordnet.
- ③ Der Schnittstelle **enp0s8** ist bislang keine IP-Adresse zugeordnet.

Nun wird beiden Gästen den Schnittstellen, die am internen Netzwerk angeschlossen sind, jeweils mit **ip** eine IP-Adresse zugeordnet. Das verwendete Netzwerk darf noch nicht vergeben sein. Da die Schnittstelle **enp0s3** per NAT bereits mit dem Netzwerk **10.0.2.0/24** verbunden ist, wird mit der IP-Adresse **10.0.3.10/24** ein Netzwerk gewählt, dass noch frei ist.

Beispiel 2. Auf Gast 1

```
$ sudo ip addr add 10.0.3.10/24 dev enp0s8 valid_lft forever ①
$ ip addr show enp0s8 ②
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
group default qlen 1000
    link/ether 08:00:27:f3:87:36 brd ff:ff:ff:ff:ff:ff
    inet 10.0.3.10/24 scope global enp0s8 ③
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fef3:8736/64 scope link
        valid_lft forever preferred_lft forever

$ ip route ④
default via 10.0.2.2 dev enp0s3 proto static metric 100
10.0.2.0/24 dev enp0s3 proto kernel scope link src 10.0.2.15 metric 100
10.0.3.0/24 dev enp0s8 proto kernel scope link src 10.0.3.10 ⑤
169.254.0.0/16 dev enp0s3 scope link metric 1000
```

- ① Der Schnittstelle `enp0s8` wird die IP-Adresse `10.0.3.10` in dem Subnetz `10.0.3.0/24` (CIDR-Notation) zugeordnet. Die Gültigkeit der IP-Adresse wird mit `valid_lft forever` als unbegrenzt gesetzt. Ohne diese Angabe wird die IP-Adresse nach einiger Zeit automatisch wieder verworfen, was in diesem Zusammenhang ein unerwünschtes Systemverhalten wäre. Falls nötig, kann mit `sudo ip addr del 10.0.3.10/24 dev enp0s8` die Zuordnung der IP-Adresse wieder entfernt werden.
- ② Durch Angabe des Schnittstellennamens gibt `ip addr` nur die Informationen für `enp0s8` aus.
- ③ In der Ausgabe kann man nachvollziehen, dass die Netzwerkschnittstelle `enp0s8` nun die IP-Adresse `10.0.3.10/24` erhalten hat.
- ④ Der Befehl `ip route` gibt die **Routingtabelle** des Systems aus.
- ⑤ Diese Zeile legt fest, dass alle Ziele aus dem Subnetz `10.0.3.0/24` über die Schnittstelle `enp0s8` weitergeleitet werden.

Die zweite VM erhält die IP-Adresse `10.0.3.11/24`. Mit dem Befehl `ping` kann schließlich überprüft werden, ob die erste VM unter der IP-Adresse `10.0.3.10/24` erreichbar ist.

Beispiel 3. Auf Gast 2

```
$ sudo ip addr add 10.0.3.11/24 dev enp0s8 valid_lft forever ①
$ ping 10.0.3.10 ②
PING 10.0.3.10 (10.0.3.10) 56(84) bytes of data.
64 bytes from 10.0.3.10: icmp_seq=1 ttl=64 time=0.972 ms
64 bytes from 10.0.3.10: icmp_seq=2 ttl=64 time=0.649 ms
64 bytes from 10.0.3.10: icmp_seq=3 ttl=64 time=0.584 ms
64 bytes from 10.0.3.10: icmp_seq=4 ttl=64 time=0.630 ms
^C ③
--- 10.0.3.10 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3052ms
rtt min/avg/max/mdev = 0.584/0.708/0.972/0.157 ms
```

- ① Das zweite System erhält mit **10.0.3.11/24** auch eine IP-Adresse aus dem Subnetz **10.0.3.0/24**.
- ② Mit **Ping** wird auf der Ebene des TCP-Protokolls, dem **Internet Control Message Protocol (ICMP)**, geprüft, ob das System mit der angegebenen IP-Adresse erreichbar ist.
- ③ Mit der Tastenkombination **Strg+C** wird **ping** abgebrochen.

Aufgabe 2



Dokumentieren Sie Ihr Vorgehen mit Script und AsciiDoc und stellen Sie das Ergebnis als PDF-Datei in Ihrem E-Portfolio zur Verfügung.

- a. Erstellen Sie ein internes Netzwerk mit zwei Linux-VMs. Wählen Sie dazu IP-Adressen, die von den Beispielen und anderen Quellen abweichen.
- b. Testen Sie mit ping und der Kombination aus curl und WEBrick wie in **Aufgabe 1** die Netzwerk-Verbindung zwischen den Linux-VMs.

3. NAT-Netzwerk

Zunächst wird im VirtualBox Manager unter Datei → Einstellungen → Netzwerk ein NAT-Netzwerk eingerichtet, z. B. mit dem Subnetz **10.0.5.0/24** und DHCP. Bei den VMs muss nun jeweils an einem Netzwerk-Adapter die Einstellung NAT-Netzwerk gesetzt werden. Aufgrund der DHCP-Unterstützung werden den Schnittstellen automatisch IP-Adressen zugeordnet. Die angeschlossenen Gäste können über das NAT-Netzwerk sowohl untereinander kommunizieren als auch Systeme außerhalb des Netzwerkes erreichen, die über die Netzwerkadapter des Hosts-Systems erreichbar sind. Die Gäste sind aber umgekehrt von außen oder dem Hostsystem ohne Port-Weiterleitung nicht erreichbar.

Beispiel 4. Auf Gast 1

```
$ ip -4 addr ①
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
qlen 1000
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
group default qlen 1000
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic enp0s3
        valid_lft 86116sec preferred_lft 86116sec
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
group default qlen 1000
    inet 10.0.5.4/24 brd 10.0.5.255 scope global dynamic enp0s8 ②
        valid_lft 916sec preferred_lft 916sec
```

① Mit der Option `-4` zeigt `ip addr` nur IPv4-Adressen an.

② Die am NAT-Netzwerk angeschlossene Schnittstelle `enp0s8` hat per DHCP die IP-Adresse `10.0.5.4/24` erhalten.

Beispiel 5. Auf Gast 2

```

$ ip -4 addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic enp0s3
        valid_lft 85808sec preferred_lft 85808sec
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    inet 10.0.5.5/24 brd 10.0.5.255 scope global dynamic enp0s8 ①
        valid_lft 1196sec preferred_lft 1196sec

$ ping 10.0.5.4 ②
PING 10.0.5.4 (10.0.5.4) 56(84) bytes of data.
64 bytes from 10.0.5.4: icmp_seq=1 ttl=64 time=0.272 ms ③
64 bytes from 10.0.5.4: icmp_seq=2 ttl=64 time=0.322 ms
64 bytes from 10.0.5.4: icmp_seq=3 ttl=64 time=0.647 ms
^C
--- 10.0.5.4 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2044ms
rtt min/avg/max/mdev = 0.272/0.413/0.647/0.167 ms

```

- ① Die Schnittstelle `enp0s8` hat per DHCP die IP-Adresse `10.0.5.5/24` erhalten und befindet sich im gleichen Subnetz wie `enp0s8` von Gast 1.
- ② Gast 1 erhält ICMP-Anfragen mit ping.
- ③ Antworten von Gast 1 auf ICMP-Anfragen werden fortlaufend ausgegeben.

Aufgabe 3

Dokumentieren Sie Ihr Vorgehen mit Script und AsciiDoc und stellen Sie das Ergebnis als PDF-Datei in Ihrem E-Portfolio zur Verfügung.

- a. Erstellen Sie ein NAT-Netzwerk mit zwei Linux-VMs. Wählen Sie dazu IP-Adressen, die von den Beispielen und anderen Quellen abweichen.
- b. Testen Sie mit ping und der Kombination aus curl und WEBrick wie in den [Aufgaben 1](#) und [2](#) die Netzwerk-Verbindungen zwischen
 - den beiden Linux-VMs sowie
 - dem Hostsystem und einer Linux-VM.

4. Netzwerkbrücke

Bei der Netzwerkbrücke erhält der Gast direkten Zugriff auf das Host-Netzwerk, so als wäre er dort direkt angeschlossen. Falls auf dem Host-Netzwerk ein DHCP-Server die IP-Adressen verwaltet, so erhält der Gast auch eine eigenständige IP-Adresse, die vom Hostsystem abweicht. Aufgrund der direkten Anbindung an das Host-Netzwerk ist die Netzwerkbrücke für einige Experimente, insbesondere mit DHCP, *nicht* geeignet.

Aufgabe 4



Dokumentieren Sie Ihr Vorgehen mit Script und AsciiDoc und stellen Sie das Ergebnis als PDF-Datei in Ihrem E-Portfolio zur Verfügung.

- a. Erstellen Sie eine Netzwerkbrücke mit einer Linux-VM.
- b. Testen Sie mit ping und der Kombination aus curl und WEBrick wie in den vorherigen Aufgaben die Netzwerk-Verbindung zwischen
 - dem Hostsystem und der Linux-VM sowie
 - der Linux-VM und einer zweiten VM.

5. Host-only-Adapter

Ähnlich wie beim internen Netzwerk muss im Virtualbox Manager unter „Globale Tools“ zunächst ein Host-only-Netzwerk (optional mit DHCP-Server) erstellt werden. Dann kann in den Einstellungen der *abgeschalteten* VMs jeweils ein Adapter als Host-only-Adapter konfiguriert werden.

Auch das Hostsystem erhält automatisch eine virtuelle Netzwerk-Schnittstelle und, falls der DHCP-Server für das Host-only-Netzwerk aktiviert ist, auch automatisch eine IP-Adresse für diese Schnittstelle.

Die angeschlossenen Gäste und das Hostsystem haben nun ein gemeinsames internes Netzwerk, das mit anderen Netzen nicht verbunden ist. Dieser Modus eignet sich wie auch das interne Netzwerk oder das NAT-Netzwerk sehr gut für Experimente. Das Host-only-Netzwerk ermöglicht es, dass man auch das Hostsystem ohne Port-Weiterleitung zum Testen und zur Diagnose einsetzen kann.

Aufgabe 5



Dokumentieren Sie Ihr Vorgehen mit Script und AsciiDoc und stellen Sie das Ergebnis als PDF-Datei in Ihrem E-Portfolio zur Verfügung.

- a. Erstellen Sie ein Host-only-Adapter mit zwei Linux-VM. Wählen Sie dazu IP-Adressen, die von den Beispielen und anderen Quellen abweichen.
- b. Testen Sie mit ping und der Kombination aus curl und WEBrick wie in [Aufgabe 3](#) die Netzwerk-Verbindung zwischen
 - den beiden Linux-VMs sowie
 - dem Hostsystem und einer Linux-VM.