

# Dokumentation OpenWRT-Test-Setup IP-Assignment

Das Test-Setup soll komplett virtualisiert auf einem Linux-Host laufen. Daher wird hier auf KVM + QEMU zur Virtualisierung gesetzt. Da die einzelnen VMs nur wenig Ressourcen benötigen und auch nur wenig benötigen sollen, werden hier OpenWRT-Guests genutzt.

Zuallererst müssen die erforderlichen Pakete installiert werden (hier auf Debian-basiertem Linux mit dem Paketmanager **apt**):

```
root@buster:~# apt install qemu qemu-system-x86
```

Zeichenstile Ohne

Dabei handelt es sich um die Pakete **qemu** und **qemu-system-x86**. Mithilfe des Pakets **qemu-system-x86** wird dann gezielt x86-Hardware emuliert. Zum Nutzen anderer Architekturen können die entsprechenden Pakete **qemu-system-XXXX** + ein entsprechendes OpenWRT-Image mit der richtigen Architektur genutzt werden.

Nach der Installation wird das gewünschte Netzwerksetup vorbereitet. QEMU kann dabei je nach Einstellungen mehrere Guests an einer virtuellen Bridge verbinden sodass diese dann ein eigenständiges, isoliertes Netzwerk bilden.

Diese virtuelle Bridge kann man mit folgenden beiden Befehlen temporär (bleibt nach einem Systemneustart nicht erhalten) erstellen und aktivieren:

```
root@buster:~# ip link add name br0 type bridge && ip link set dev br0 up
```

*(In der Linux Shell können zwei Kommandos mittels des && Operators kombiniert werden. Dabei handelt es sich um eine bedingte Verbindung, die Befehle werden nacheinander ausgeführt, der zweite jedoch nur wenn der erste erfolgreich war. Alternativ kann mit dem Operator ; erreicht werden, dass zwei Befehle einfach nacheinander ausgeführt werden)*

Der erste Befehl fügt ein Netzwerkinterface vom Typ **bridge** mit dem Namen **br0** zur Netzwerkkonfiguration hinzu. Der zweite Befehl aktiviert dann dieses eben erstellte Interface.

Damit QEMU dieses Bridge-Interface dann auch nutzen kann, muss es in der QEMU-Konfigurationsdatei explizit erlaubt werden. Da diese Datei und auch der entsprechende Ordner im Regelfall noch nicht existiert, werden diese zuerst erstellt:

```
root@buster:~# mkdir /etc/qemu
```

Zeichenstile Ohne

Mit einem Editor (hier nano) wird dann die Datei erstellt und gleichzeitig zum Bearbeiten geöffnet:

```
root@buster:~# nano /etc/qemu/bridge.conf
```

Dort wird dann folgendes eingetragen:

```
GNU nano 3.2 /etc/qemu/bridge.conf
allow br0
```

Das schaltet das eben erstellte Interface **br0** explizit für die Verwendung als QEMU-Bridge frei. Oft kommt es auch noch vor, dass der qemu-bridge-helper im Hintergrund nicht richtig auf die Datei zugreifen kann. Vorsorglich sollte mit folgendem Befehl das SGID-Bit dieses Programms gesetzt werden. Damit erhält es je nach Konfiguration weitere benötigte Rechte:

```
root@buster:~# chmod u+s /usr/lib/qemu/qemu-bridge-helper
```

Damit ist dann die hostseitige Netzwerkkonfiguration abgeschlossen.

Als nächstes muss ein bootfähiges OpenWRT-Image besorgt werden. Dazu begibt man sich entweder auf die offizielle OpenWRT-Download-Seite:

<https://downloads.openwrt.org/>

oder baut sich sein Image für die Architektur x86 selbst.

Download in der Kommandozeile:

```
root@buster:~# wget https://downloads.openwrt.org/snapshots/targets/x86/64/openwrt-x86-64-generic-ext4-combined.img.gz
```

Damit wird das Image unter dem Link in das aktuelle Verzeichnis heruntergeladen. Hier ist es das Home-Verzeichnis des Benutzers **root**.

Herunterladbare wie auch selbst erstellte Images liegen meist gepackt als GZ-Archiv (Dateiendung .gz) vor. Um dieses zu entpacken, kann folgender Befehl genutzt werden:

```
root@buster:~# gzip -d openwrt-x86-64-generic-ext4-combined.img.gz
```

Anschließend liegt das Image entpackt als .img-File im selben Ordner, das GZ-Archiv wurde dabei vom Kommando gzip entfernt.

Je nachdem, wieviele Guests das Testsetup schließlich beinhaltet soll, müssen von diesem Image nun entsprechend viele Kopien erstellt werden. Dies ist notwendig, da QEMU für jede VM das Image sperrt um darin Daten verändern zu können. Eine Kopie erstellt man mit:

```
root@buster:~# cp openwrt-x86-64-generic-ext4-combined.img openwrt-x86-64-generic-ext4-combined-2.img
```

Anschließend hat man, je nach dem wie viele man erstellt hat, mehrere Images im Ordner:

```
root@buster:~# ls
openwrt-x86-64-generic-ext4-combined-2.img
openwrt-x86-64-generic-ext4-combined-3.img
openwrt-x86-64-generic-ext4-combined-4.img
openwrt-x86-64-generic-ext4-combined.img
root@buster:~#
```

Anschließend kann nun mit dem Befehl **qemu-system-x86\_64** jeweils die VM für den Client gestartet werden.

```
root@buster:~# qemu-system-x86_64 openwrt-x86-64-generic-ext4-combined.img -nographic -net
dev bridge,id=hn1 -device virtio-net,netdev=hn1,mac=52:54:00:12:34:50
```

Dabei müssen einige Parameter übergeben werden damit QEMU das Image mit den richtigen Einstellungen bootet:

- nographic Weist QEMU an, die VM ohne Grafikunterstützung zu starten. Standardmäßig bindet QEMU an die VM einen virtuellen Grafikadapter an die VM und zeigt die Ausgabe in einem separaten Fenster. Dazu nutzt QEMU einen Window Server, der bei der Nutzung einer GUI auf dem Host sowieso schon vorhanden ist. Da wir hier nur in der Kommandozeile arbeiten, demnach kein Window-Server existiert und auch die Mehrfensterunterstützung fehlt, würde der Befehl ohne diesen Parameter fehlschlagen.
- netdev Dieser Parameter definiert für diese VM ein Netzwerkbackend. Dieses ist auf dem Host notwendig um mit dem virtuellen Netzwerkinterface der VM zu kommunizieren. Dieses Backend abstrahiert das Interface am Host und stellt die Kommunikation der Netzwerkteilnehmer sicher. Das Backend wird von QEMU mit dem vorhin erstellten Bridge-Interface gekoppelt.
  - bridge legt fest, dass das Backend vom Typ Bridge ist und an eine Netzwerkbridge am Host gebunden wird.
  - id legt die ID des Backends fest. Anhand dieser ID unterscheidet QEMU letztendlich, welche VMs gemeinsam an einem Backend und damit an der Bridge hängen.
- device Definiert für die entsprechende VM dann einen virtuellen Netzwerkadapter, über den die VM kommunizieren kann.
  - virtio-net ist dabei der interne Typ des Adapters.
  - netdev stellt dann die Referenz zum Backend her und muss der ID entsprechen, die beim Parameter -netdev angegeben wurde
  - mac legt die MAC-Adresse des Interfaces fest. Diese muss sich zwischen den VMs unterscheiden.

Beim Ausführen dieses Kommandos sollte dann die Boot-Ausgabe und die serielle Konsole der OpenWRT-VM angezeigt werden.

Dieses Vorgehen wird dann für alle gewünschten Knoten im Setup entsprechend wiederholt. Da QEMU in der Konstellation nicht im Hintergrund als Daemon ausführbar ist, muss damit für jeden Knoten eine neues Terminal verwendet werden.

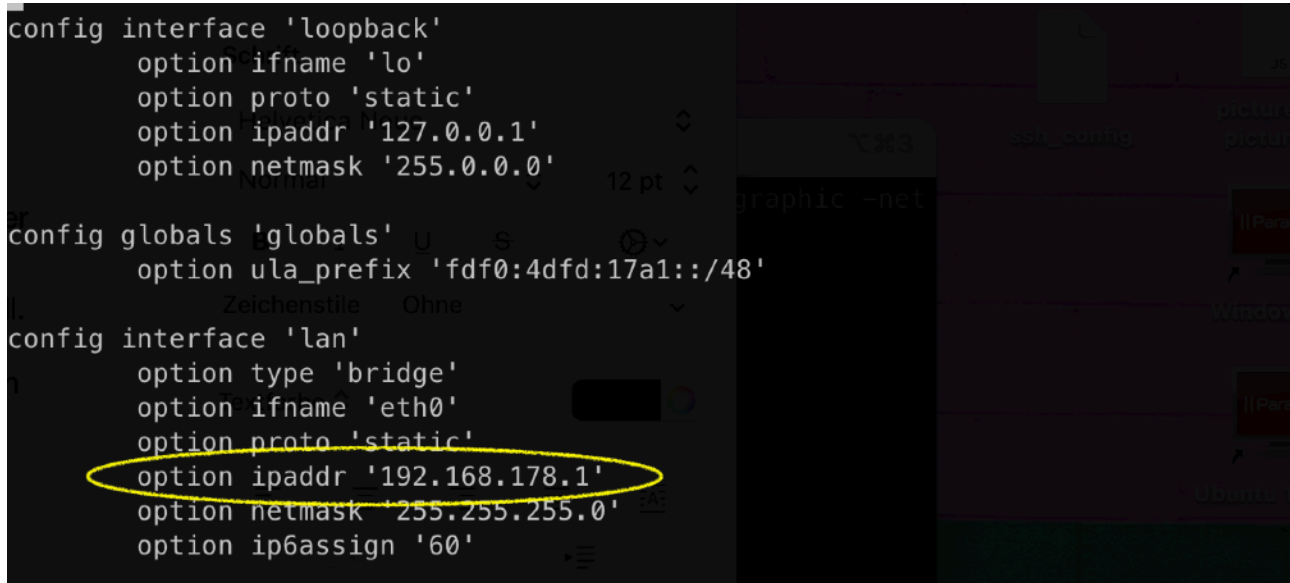
Weiterhin muss dann entsprechend eine Kopie des Images gewählt werden und die MAC-Adresse alterniert werden. Die restlichen Parameter bleiben gleich.

## Konfiguration in den VMs

Damit sind alle Voraussetzungen für das IP-Assignment-Setup erfüllt und es kann mit der Konfiguration innerhalb der VMs begonnen werden.

Zuerst wird eine VM gewählt, welche als DHCP-Server in dem isolierten Netz dienen soll. Als DHCP-Server-Anwendung wird hier **dnsmasq** verwendet, welcher auf OpenWRT optimalerweise schon vorinstalliert ist und nur noch konfiguriert werden muss.

Auf dieser VM wird dann zuerst mittels eines Editors die Datei **/etc/config/network** bearbeitet (hier mit vim da andere Editoren standardmäßig nicht in OpenWRT vorhanden sind und erst nachinstalliert werden müssen):



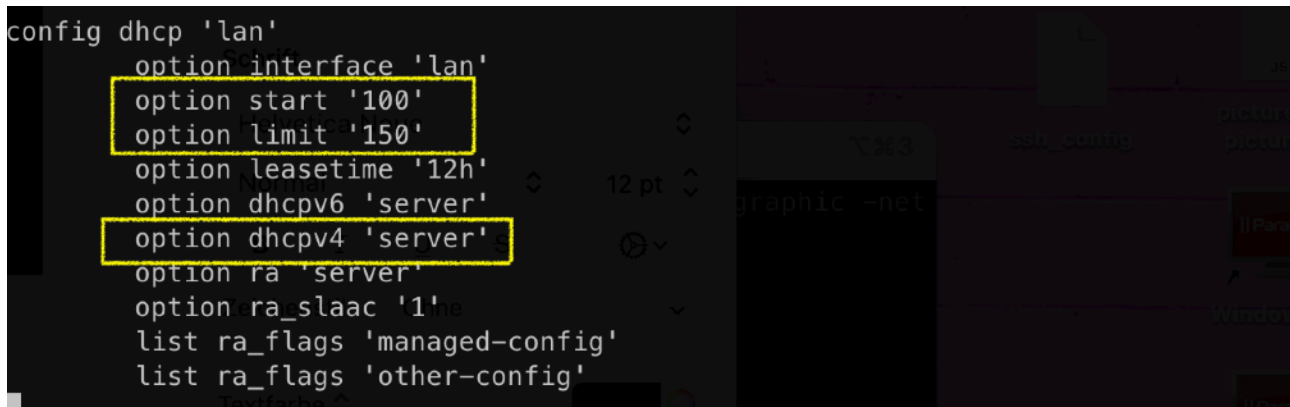
```
config interface 'loopback'
    option ifname 'lo'
    option proto 'static'
    option ipaddr '127.0.0.1'
    option netmask '255.0.0.0'

config globals 'globals'
    option ula_prefix 'fdf0:4dfd:17a1::/48'

config interface 'lan'
    option type 'bridge'
    option ifname 'eth0'
    option proto 'static'
    option ipaddr '192.168.178.1'
    option netmask '255.255.255.0'
    option ip6assign '60'
```

Das meiste der Standardkonfiguration kann belassen werden, lediglich die statische IP sollte geändert werden, hier wird das Netz **192.168.178.0/24** verwendet.

Anschließend wird die Datei gespeichert. Als nächstes wird mit entsprechend auch die Datei **/etc/config/dhcp** editiert. Hier bis zum Config-Block für das Interface lan scrollen. Im allgemeinen Block für dnsmasq müssen keine Änderungen vorgenommen werden.



```
config dhcp 'lan'
    option interface 'lan'
    option start '100'
    option limit '150'
    option leasetime '12h'
    option dhcpv6 'server'
    option dhcpv4 'server'
    option ra 'server'
    option ra_slaac '1'
    list ra_flags 'managed-config'
    list ra_flags 'other-config'
```

Die Anweisung **option dhcpv4 ,server'** ist standardmäßig nicht vorhanden, muss also noch eingefügt werden im den DHCPv4 Server zu aktivieren.

Weiterhin kann mit den Anweisungen **option start** und **option limit** der IP-Adressbereich festgelegt werden. **Start** gibt hierbei den Offset des IP-Bereichs von der Basisadresse (hier 192.168.178.0) an und **Limit** die Größe des Adressbereichs. Dementsprechend kann der Server mit dieser Konfiguration nun IP-Adressen von 192.168.178.100 bis 192.168.178.249 vergeben.

Anschließend wird die Datei gespeichert und sowohl der Netzwerkdienst, als auch dnsmasq mit folgenden Befehlen neu geladen:

```
root@OpenWrt:/# service network reload
root@OpenWrt:/# service dnsmasq reload
```

Die Konfiguration des Servers ist nun abgeschlossen, nun müssen nur noch die Clients entsprechend angepasst werden. Das folgende Vorgehen muss dann für jede Client-VM wiederholt werden.

Die einzige Anpassung die hier nötig ist, dass das Umstellen der statischen IP-Konfiguration auf DHCP. Das wird wiederum in der Datei **/etc/config/network** getan:

```
config interface 'lan'
    option type 'bridge'
    option ifname 'eth0'
    option proto 'dhcp'
    #option ipaddr '192.168.1.1'
    #option netmask '255.255.255.0'
    #option ip6assign '60'
```

Dort wird im Abschnitt für das Interface 'lan' die Anweisung **option proto ,static'** in **option proto ,dhcp'** geändert. Die Anweisungen **ipaddr**, **netmask**, **ip6assign** werden mit **#** auskommentiert.

Anschließend wird auch hier der Netzwerkdienst mit **service network reload** neu geladen und der Client sollte kurze Zeit später eine IP-Adresse vom DHCP-Server erhalten haben.