Сетевой мост для QEMU в Linux.

Часть 1. Создание конфигурационных файлов.

Во первых необходимо включить маршрутизацию между интерфейсами.

$ sudo sysctl net.ipv4.ip_forward=1

Для постоянного включения маршрутизации создайте файл.

$ echo "net.ipv4.ip_forward = 1" **>** /etc/sysctl.d/99-sysctl.conf

Теперь придумайте имя вашего будущего сетевого адаптера для режима **«Мост»**. Пусть у меня будет **«br0»**. Мы с вами будет пользоваться встроенными в систему средствами без каких-либо дополнительных утилит.

С одной стороны — это слегка усложняет настройку, с другой стороны убирает необходимость устанавливать что-либо. Усложняет в том, смысле что, например, графический интерфейс **Network-Manager** не поддерживает создание адаптера для режима мост, только консольные средства.

Возможно для последнего я просто не нашел необходимых настроек, однако, вам всё равно также придётся искать. Команды же намного проще, чем кажутся. Ну да хватит демагогии, переходим к делу.

Теперь необходимо создать файл **«/etc/qemu/bridge.conf»** и записать в него одну строку с названием придуманного вами адаптера режима **«Мост»**. Собственно говоря, адаптер может быть не один. Просто запишите на каждой строке свой адаптер.

$ sudo mkdir -p /etc/qemu/

$ sudo touch /etc/qemu/bridge.conf

$ sudo nano /etc/qemu/bridge.conf

allow br0

Если вам сетевой мост необходим на постоянной основе, то стоит также сделать ручную настройку адресов данного адаптера, **но только после его создания**. Однако, настройку всё равно приведу, чтобы не листать вверх-вниз.

$ sudo nano /etc/network/interfaces

allow-hotplug br0

auto br0

iface br0 inet static

address 192.168.0.120

netmask 255.255.255.0

broadcast 192.168.0.255

gateway 192.168.0.1

dns-nameservers 192.168.0.1 8.8.8.8

pre-up ifconfig br0 hw ether xx:xx:xx:xx:xx:xx

Последняя строка, которая изменяет MAK-адрес не обязательна. Широковещательный адрес рекомендую всё-таки указывать, хоть он и не обязателен. Шлюз указывать обязательно. А вот DNS сервер не обязателен.

Однако, именно эту часть настроек вы можете внести уже графически через **«Network-Manager»**, т.к. после создания адаптера вручную с указанием его типа он автоматически появляется в списках адаптеров для всех менеджеров сетей. Так что не обязательно делать всё вручную. Вручную можно сделать только часть настроек.

Часть 2. Создаем адаптер и настраиваем. Машрутизация.

Создаем адаптер с типом **«Мост»**.

$ sudo ip link add name br0 type bridge

Включаем его.

$ sudo ip link set br0 up

Далее не обязательно вносить все настройки вручную, через терминал, т.к. после создания адаптера вручную с указанием его типа он автоматически появляется в списках адаптеров для всех менеджеров сетей.

Указываем главный адаптер. У меня это **«enp3s0»**, проводной адаптер. У вас может другой, например, **«eth0»**, или **«wlp4s0»** беспроводной. Просто наберите в терминале команду **«ip link show»**, чтобы узнать названия всех ваших адаптеров и активных соединений (если отсутствует ifconfig — из net-tools утилиты). Если используете WiFi — то название должно начинаться с **«wl»**, а у проводного с **«en»** или с **«et»**.

$ sudo ip link set enp3s0 master br0

Подключаем сетевой мост.

$ sudo bridge link

Теперь настраиваем адреса и не забываем про широковещательный адрес. Я для вычисления широковещательного адреса использую самый простой калькулятор — консольная утилита **«ipcalc»**.

$ ipcalc 192.168.0.120/24

$ sudo ip addr add 192.168.0.120/24 dev br0

$ sudo ifconfig br0 broadcast 192.168.0.255

$ sudo ifconfig br0 mtu 1000

И самое главное маршрут для этого моста по умолчанию, иначе интернета в **Qemu** не будет. Кстати, эту команду прям непосредственно можно вставить в ручные настройки адресов «**/etc/network/interfaces**», разумеется без «**sudo**».

$ sudo ip route add 192.168.0.120 via 192.168.0.1

И не забудьте указать шлюз по умолчанию, а то интернет на основном адаптере пропадет.

$ sudo route add default gw 192.168.0.1 br0

Для просмотра маршрутов по умолчанию:

$ ip route

Часть 3. QEMU

Адаптер сетевого моста указывается так:

```
-netdev bridge,br=br0,id=net0 \

-device virtio-net-pci,netdev=net0
```

Чтобы меньше заморачиваться я для себя просто создал скрипт рядом с необходимым образом системы. И в этот скрипт вношу все необходимые команды. Некоторые записываю в комментарий, чтобы не забыть — просто для удобства, чтобы ничего не искать.

Базовый файл запуска виртуальной машины, например, Archlinux выглядит так — **«qemu.sh»**:

```
#!/usr/bin/env bash

qemu-system-x86_64 \

-enable-kvm \

-cpu host \

-smp cores=1 \

-m 1024 \

-machine q35 \

-device intel-iommu \

-vga virtio \
```

-netdev bridge,br=br0,id=net0 \

-device virtio-net-pci,netdev=net0 \

-boot menu=on \

-cdrom archlinux-2021.11.01-x86_64.iso \

-hda arch.qcow

А далее меняйте в этом скрипте необходимые параметры по своему усмотрению.

Часть 4. Завершение работы с сетевым мостом.

По окончании работы с сетевым мостом необходимо его правильно удалить из системы.

Если сетевой мост в QEMU больше не нужен, то смело удаляйте конфигурацию.

$ sudo rm -rf /etc/qemu/bridge.conf

Отсоединяем основной адаптер от сетевого моста.

$ sudo ip link set enp3s0 nomaster

Отключаем сетевой мост.

$ sudo ip link set br0 down

На всякий случай очищаем маршруты, если таковые остались. Обычно **«Network-Manager»** делает всё за нас в автоматическом режиме. При проблемах — команды ниже.

$ sudo route del default gw 192.168.0.1 br0

$ sudo ip route del 192.168.0.120 via 192.168.0.1

И только после этого, можем его безвредно удалить.

$ sudo ip link delete br0 type bridge

P.S.:

Тест всех соединений в приложенных скриншотах.

Ну а сегодня на этом всё. Надеюсь я хоть немного вас заинтересовал.

Спасибо за внимание. Всем Удачи, до новых встреч, Пока-Пока!

`/etc/systemd/network/50-dummy0.netdev`:

```
[NetDev]
Name=dummy0
Kind=dummy
Description=Dummy network for dnscrypt-proxy
```

`/etc/systemd/network/50-dummy0.network`:

```
[Match]
Name=dummy0

[Network]
DHCP=no
Address=10.0.197.1/24
```

```
DefaultRouteOnDevice=false
```

Bridge-Utils.

$ sudo pacman -S bridge-utils --noconfirm

# echo 'deb http://ftp.de.debian.org/debian sid main' > /etc/apt/sources.list.d/ftp.de.debian.org.list
$ sudo apt install bridge-utils –y

With bridge-utils

Create a new bridge:

brctl addbr bridge_name

Add a device to a bridge, for example eth0:

brctl addif bridge_name eth0

Show current bridges and what interfaces they are connected to:

$ brctl show

Set the bridge device up:

ip link set dev bridge_name up

Delete a bridge, you need to first set it to down:

ip link set dev bridge_name down
brctl delbr bridge_name

modprobe br_netfilter

Configuring bridging in /etc/network/interfaces

```
 # Bridge setup
 iface br0 inet static
    bridge_ports eth0 eth1
```

```
    address 192.168.1.2
    broadcast 192.168.1.255
    netmask 255.255.255.0
    gateway 192.168.1.1
    bridge_ports eth1
    bridge_stp off
    bridge_fd 0
    bridge_maxwait 0
```

```
bridge_stp off       # disable Spanning Tree Protocol
bridge_waitport 0    # no delay before a port becomes available
bridge_fd 0          # no forwarding delay
bridge_ports none    # if you do not want to bind to any ports
bridge_ports regex eth* # use a regular expression to define ports
```

```
auto br0
iface br0 inet static
        address 10.18.44.26
        netmask 255.255.255.192
        broadcast 10.18.44.63
        dns-nameservers 10.0.80.11 10.0.80.12
        # set static route for LAN
        post-up route add -net 10.0.0.0 netmask 255.0.0.0 gw 10.18.44.1
        post-up route add -net 161.26.0.0 netmask 255.255.0.0 gw 10.18.44.1
        bridge_ports eth0
        bridge_stp off
        bridge_fd 0
        bridge_maxwait 0
```