Používání KVM a na vpsFree.cz

Tato info stránka obsahuje návody pro zprovoznění KVM na Alpine Linux 3.4+, CentOS 7 a Debian 8.

Nejprve je potřeba ve vpsAdminu, v detailu vaší VPS, zapnout následující vlastnosti:

- Bridge – povolí vytvoření síťového bridge, do kterého následně připojíte KVM virtuály,
- TUN/TAP - povolí vytváření virtuálních interface, které jsou pak bridgovány,
- iptables – povolí použití iptables, které jsou potřeba pro nastavení IP maškarády,
- KVM – povolí použití KVM (pro HW podporu virtualizace).

Na vpsFree není jiná možnost než mít disky pro KVM virtuály v souborech (images). Doporučujeme k tomu vytvořit nový (sub)dataset, **vypnout v něm kompresi** a nastavit velikost bloku (Record size) na 65536 B. Vypnutí komprese je velmi důležité, jinak vám bude image bobtnat a bobtnat, přes deklarovanou velikost image, dokud narazí na kvótu datasetu! Pro optimální výkon vytvářejte „raw“ image.

KVM na Alpine Linuxu

Přípravu hypervizoru lze provést automatizovaně pomocí Ansible.

Na GitHubu jsou dle níže uvedeného návodu automatizovány zatím tyto sekce:

- Instalace balíčků a konfigurace hypervizoru - tag `instalace`
- Nastavení iptables - tag `nastaveni-iptables`

Instalace balíčků a konfigurace hypervizoru

Nainstalujte potřebné balíčky (ip6tables je volitelný):

```
apk update
apk add qemu-system-x86_64 qemu-openrc qemu-img bridge iptables ip6tables
```

Nastavte potřebná práva pro přístup ke KVM a TUN (do skupin *kvm* a *netdev* by měl patřit uživatel *qemu*, pod kterým bude běžet vytvořený virtuál):

```
chown :kvm /dev/kvm
chmod g+rw /dev/kvm
chown :netdev /dev/net/tun
chmod g+rw /dev/net/tun
```

Nakonfigurujte bridge pro Qemu/KVM virtuály; vytvořte soubor */etc/network/interfaces.tail* (IP adresu si samozřejmě můžete zvolit libovolnou z privátního rozsahu):

```
auto br0
iface br0 inet static
     pre-up brctl addbr br0
```

```
    address 172.17.1.1
    netmask 255.255.255.0
    post-down brctl delbr br0
```

Jelikož OpenVZ poněkud neohrabaně přepisuje soubor */etc/network/interfaces* (proto také konfiguraci bridge přidáváme do *interfaces.tail*, tak nejjistější je v tuto chvíli kontejner restartovat. :/

Povolte uživateli ve skupině *qemu* spravovat vytvořený bridge:

```
echo "allow br0" > /etc/qemu/bridge.conf
chown root:qemu /etc/qemu/bridge.conf
chmod 0640 /etc/qemu/bridge.conf
```

Nastavte IP maškarádu pro přístup Qemu/KVM virtuálů ven do veřejného Internetu.

Pokud už máte nakonfigurované iptables, tak jen přidejte pravidlo:

```
iptables -t nat -A POSTROUTING -s 172.17.1.0/24 ! -o br0 -j MASQUERADE
```

V opačném případě můžete postupovat např. podle odstavce Nastavení iptables.

Vytvoření a spuštění virtuálu

V tomto návodu předpokládám použití qemu-openrc – OpenRC init skript pro startování Qemu/KVM. Na Alpine samozřejmě můžete používat i libvirt, ale opravdu chcete a potřebujete jeho neohrabané XML konfiguráky a/nebo klikátka nad ním…? ;) Qemu-openrc je výrazně jednodušší a transparentnější řešení. Každý virtuál je reprezentovaný init skriptem, stejně jako ostatní programy, můžete deklarativně nastavovat závislosti mezi virtuály atd.

Vytvoření nového virtálu spočívá pouze v přípravě image disku, vytvoření symlinku pro init skript a úpravě jednoduchého konfiguračního skriptu. Nechť se nový virtuál jmenuje třeba „myvirt.“

Připravte raw image pro myvirt o potřebné velikosti:

```
mkdir -p /var/lib/qemu/myvirt/
qemu-img create -f raw /var/lib/qemu/myvirt/disk0.raw 5G
chown qemu:qemu /var/lib/qemu/myvirt/disk0.raw
chmod 0600 /var/lib/qemu/myvirt/disk0.raw
```

Zkopírujte výchozí konfigurák /etc/conf.d/qemu do /etc/conf.d/qemu.myvirt a upravte dle potřeby:

```
cd /etc/conf.d
cp qemu qemu.myvirt
vi qemu.myvirt  # read comments and edit
```

Zejména tedy přidejte připravený image:

```
disk1_file="/var/lib/qemu/myvirt/disk0.raw"
disk1_format="raw"
```

A zřejmě také instalační CD některé distribuce, které jste si předem stáhli:

```
cdrom0_file="/var/lib/qemu/alpine-virt-3.4.1-x86_64.iso"
```

Vytvořte symlink pro init skript a spusťte myvirt:

```
cd /etc/init.d
ln -s qemu qemu.myvirt

rc-service qemu.myvirt start
```

Pokud máte problémy s tím, že při větším provozu na síťové kartě spadne ve virtuálu sít, skuste použít driver e1000 místo výchozího virtio-net-pci:

```
net0_device="e1000"
```

stabilitu je možno otestovat iperf-em, na testovanem stroji:

```
iperf -s
```

na hostiteli pak:

```
iperf -c 172.17.1.X -P 20
```

Nastavení iptables

Pokud nepoužíváte žádný nástroj pro generování iptables pravidel (jako např. ferm), doporučuji vyjít z připravené šablony pravidel. Tyto jsou přímo ve formátu iptables, který je možný načíst pomocí iptables-restore.

Stáhněte upravenou šablonu pravidel s přidanou maškarádou pro náš bridge do /etc/iptables:

```
rmdir /etc/iptables
wget -O /etc/iptables
https://gist.githubusercontent.com/jirutka/3742890/raw/c9f6bdbfcf597578e562c
92ea1e256a9ebcf3a2c/rules-both.iptables
```

Upravte konfigurák /etc/conf.d/iptables (IPv4):

```
# /etc/conf.d/iptables

IPTABLES_SAVE="/etc/iptables"
#SAVE_RESTORE_OPTIONS="-c"
SAVE_ON_STOP="no"
IPFORWARD="yes"
```

...a /etc/conf.d/ip6tables (IPv6):

```
# /etc/conf.d/ip6tables

IP6TABLES_SAVE="/etc/iptables"
SAVE_RESTORE_OPTIONS="-T filter"
SAVE_ON_STOP="no"
IPFORWARD="yes"
```

Spusťte iptables a ip6tables a přidejte je do runlevel boot:

```
rc-service iptables start
rc-service ip6tables start
rc-update add iptables boot
rc-update add ip6tables boot
```

KVM na CentOS 7

Tento návod platí pouze pro CentOS 7.1. Na CentOS 7.2 aktuálně nefunguje interní síť mezi VPS a VM. Použijte buď CentOS 7.1 nebo Debian 8, než bude tento problém vyřešen.

2018-03-26 (phatina) Pripojenie k libvirtd zlyhava: An error occurred, but the cause is unknown.

KVM používám prostřednictvím libvirt na aktualizovaném CentOS 7.

Doporučuju CentOS 7 plně aktualizovat, nakonfigurovat a nainstalovat potřebný software. Z důvodu povolení iptables je potřeba nakonfigurovat nebo vypnout firewalld.

```
yum group install virtualization-host-environment
yum install virt-manager xauth
systemctl enable libvirtd
systemctl disable firewalld
reboot
```

Vytvoření virtuálu pomocí virt-manager na straně serveru

Motivace: Když pracujete na pomalém připojení (což O2 ADSL na vsi bezpochyby je), potřebujete minimalizovat datové toky přes váš pracovní počítač. Lokální virt-manager by stahoval minimálně kernel a initramdisk po relativně pomalém downloadu a typicky po ukrutně pomalém uploadu nahrával na hostitelský kontejner.

```
ssh root@your-host-name -Y virt-manager
```

Na vzdáleně spuštěné instanci lze zahájit instalaci, ale nefunguje mi zobrazení instalátoru pomocí výchozího spice. Protože mi přijde krajně nepraktické přepínat libvirt na VNC, takže doporučuju pro

instalaci a další používání spustit virt-manager lokálně a přidat cestu k serveru.

KVM na Debian 8

Tento návod je prováděn na Debianu 8. Měl by být plně funkční také na CentOS (s jinými příkazy viz. výše). Pokoušel jsem se KVM rozchodit i na Ubuntu 14.04, ale tam bohužel marně.

Pokud používáte Windows, je potřeba nainstalovat a zapnout XMing (pokud budete chtít minimálně virtuál nainstalovat a nastavit skrz grafické rozhraní jako já). Poté zapneme putty (nezapomeňte si zapnout XMing a zaškrtnout „Enable X11 forwarding“) a můžeme kouzlit. Stačí také přes „apt install xrdp“ na server nainstalovat podporu remote desktop a od té doby už se z Windows připojovat nativním klientem Remote desktop (mstsc).

Začneme klasikou:

```
apt-get update
apt-get upgrade
```

Poté nainstalujeme libvirt knihovnu:

```
apt-get install qemu-kvm libvirt-bin
apt-get install virt-manager
```

Celkově mají tyto dva balíčky asi 320 MB. Dále je potřeba si někam na disk stáhnout image distribuce, kterou chcete do virtuálu nainstalovat. Já jsem instaloval Ubuntu server 14.04. Vyberte si adresář, do kterého chcete image stáhnout, a pomocí `wget` stáhněte ISO image.

```
cd /home
wget http://releases.ubuntu.com/14.04.3/ubuntu-14.04.3-server-amd64.iso
```

Zapněte virt-managera.

```
virt-manager
```

Díky XMingu můžete teď na dálku z Windows nastavit a spustit instalaci virtuálního serveru uvnitř své VPS. Upozorňuji, že pod X11 nefunguje spice displej, takže pokud chcete grafické rozhraní při instalaci virtuálu, je potřeba to v nastavení přepnout na „Display VNC“ a restartnout virtuál. Pokud to uděláte a server restartnete, může ztratit informaci o tom, že byl připojen .iso image pro instalaci, takže musíte opět do nastavení do záložky „CD-ROM“ a tam v „Source-path“ nastavit znova .iso image pro instalaci. Pokud vytvoření KVM mašiny selže s hláškou „Could not start virtual network „default“: Unable to set bridge virbr0 forward_delay: Operation not permitted“, pak si ve VPS adminu povolte „TUN/TAP“. Poté již jen nainstalujte zvolený Linux. Nakonec už zbývá jen na vaší VPS udělat správný portforwarding, aby jste se k virtuálu skrz VPS dostali z venku. Pokud budete chtít používat příkazy virsh, např. pro start a autostart sítě „default“, používejte přepínač -c, např.

```
virsh -c qemu:///system net-start default
virsh -c qemu:///system net-autostart default
virsh -c qemu:///system net-list --all
```

```
virsh -c qemu:///system list
```

Pokud nechcete pokaždé zadávat přepínač -c pro příkazy virsh a používáte pouze lokální hypervisor, je potřeba v tomto souboru /etc/libvirt/libvirt.conf odkomentovat následující příkaz:

```
uri_default = "qemu:///system"
```

Poté příkazy virsh dále používáme již bez přepínače -c tedy následně:

```
virsh net-start default
virsh net-autostart default
virsh net-list --all
virsh list
```

Konfigurace sítě KVM na Debian 8

* Routování veřejné IPv4 přes lokální rozsah do KVM virtuálu, lokální IPv4 u VPS

Tento návod byl otestován na VPS Debian 8 OpenVZ kontejneru, jako KVM virtuál byl použit systém Ubuntu 16.04.1 LTS x86_64.

Máme pouze jednu veřejnou IPv4 adresu `185.8.164.43` a tu přidělíme KVM virtuálu a síťový provoz plně směrujeme pro veřejnou IPv4 adresu `185.8.164.43` z VPS přes privátní / lokální rozsah `172.16.20.0/24` / `192.168.122.0/24` na veřejnou IPv4 adresu `185.8.164.43` KVM virtuálu. Tedy kromě portu 1022, ten rezervujeme pro SSH na VPS, aby byla VPS na IPv4 adrese `185.8.164.43` přes SSH i nadále dostupná. KVM virtuál má plnou konektivitu z / do internetu přes privátní / lokální IPv4 adresu `172.16.20.13` / `192.168.122.2` (kromě portu 1022 na IPv4 adrese `185.8.164.43`).

VPS má pouze privátní / lokální IPv4 adresu `172.16.20.12` / `192.168.122.1` a je z pohledu internetu za NATem v rámci sítě vpsfree.cz a není z internetu dostupná, kromě výše zmíněného portu 1022 (na veřejné IPv4 adrese `185.8.164.43` KVM virtuálu) pro přístup na SSH dané VPS. Pokud má VPS privátní IPv4 adresu `172.16.20.12` (rozsah v rámci sítě vpsfree.cz), má VPS IPv4 konektivitu do internetu. Pokud má VPS lokální IPv4 adresu `192.168.122.1` (rozsah v rámci sítě VPS ⇔ KVM virtuál), nemá VPS IPv4 konektivitu do internetu a není možné například stahovat a instalovat aktualizace přes IPv4 (pouze případně jen přes IPv6).

Nejprve je potřeba ve vpsAdminu (v detailu nastavení vaší VPS), zapnout / zkontrolovat vlastnosti: Bridge, iptables, KVM a TUN/TAP.
Dále nakonfigurujte bridge `br0` pro KVM virtuál na straně VPS. Jelikož OpenVZ přepisuje soubor `/etc/network/interfaces`, konfiguraci bridge `br0` přidáváme do `interfaces.tail`, OpenVZ ji pak sám po restartu vloží do `/etc/network/interfaces`.

Vytvořte tedy soubor `/etc/network/interfaces.tail` a vložte do něj následující kód (privátní IPv4 adresu VPS zaměníte za správnou IPv4 adresu z aktuální nastavené konfigurace ve vpsAdminu, lokální IPv4 adresu VPS si samozřejmě můžete zvolit libovolnou z lokálního rozsahu. Já jsem dále použil privátní IPv4 adresu pro VPS v rámci sítě vpsfree.cz:

```
auto br0
iface br0 inet static
    address 172.16.20.12
    netmask 255.255.255.255
    bridge_ports venet0:0
    bridge_stp on
    bridge_fd 0.0
    dad-attempts 0
    accept_ra 0
```

Vzhledem k tomu že OpenVZ již po této úpravě automaticky nevloží do `/etc/network/interfaces` konfiguraci pro veřejnou IPv4 adresu, vložíme ještě na konec souboru `/etc/network/interfaces.tail` dočasně konfiguraci pro veřejnou IPv4 adresu VPS. Správnou veřejnou IPv4 adresu zvolíme dle aktuální nastavené konfigurace ve vpsAdminu (tento následující kód budeme v průběhu návodu zase mazat, takže na to pamatovat):

```
auto venet0:0
iface venet0:0 inet static
    address 185.8.164.43
    netmask 255.255.255.255
```

Soubor `/etc/network/interfaces.tail` uložíme a v tuto chvíli je nutné VPS tedy OpenVZ kontejner restartovat aby se daná nastavení v souboru `interfaces.tail` uplatnila do `interfaces` a vytvořil se bridge `br0`.

Například pomocí `virt-manager` vytvoříme KVM virtuál a nakonfigurujeme síťové rozhraní směřující na bridge `br0` VPS (důležité vybrat bridge `br0`). Nastartujeme OS KVM virtuálu a v systému nakonfigurujeme jeho síť dle vzoru se správnými parametry (opět privátní IPv4 adresu KVM virtuálu zvolíte správnou z aktuální konfigurace ve vpsAdminu, lokální IPv4 adresu KVM virtuálu si zvolíme dle použitého lokálního rozsahu v předchozích krocích. Já jsem zde opět použil privátní IPv4 adresu pro KVM virtuál v rámci sítě vpsfree.cz, kde brána IPv4 je již zvolená IPv4 adresa pro bridge `br0`, tedy privátní / lokální IPv4 adresa pro VPS OpenVZ kontejner):

```
IP adresa:  172.16.20.13
Maska sítě: 255.255.255.0  (maska pro privátní IPv4, pro lokální IPv4 je
nutné pro tuto konfiguraci použít masku 255.255.255.255)
Brána:      172.16.20.12
DNS servery:    37.205.9.100,  37.205.10.88
```

A k danému rozhraní ještě přidáme veřejnou IPv4 adresu:

```
IP adresa:  185.8.164.43 (zvolíte správnou z aktuální konfigurace ve
vpsAdminu)
Maska sítě: 255.255.255.0 (masku zvolíme stejnou jako pro privátní / lokální
IPv4 adresu)
```

Vrátíme se ke konfiguraci souboru `/etc/network/interfaces.tail` a odebereme dočasnou konfiguraci pro veřejnou IPv4 adresu, tedy vymažeme následující řádky kódu na konci souboru:

```
auto venet0:0
```

```
iface venet0:0 inet static
    address 185.8.164.43
    netmask 255.255.255.255
```

Místo toho do souboru `/etc/network/interfaces.tail` přidáme routování privátního / lokálního rozsahu na rozhraní VPS tedy `venet0`.

Správný tvar pro privátní rozsah IPv4 nalezneme ve vpsAdminu zde. Správný tvar rozsahu pro lokální rozsah IPv4 je dle zvolené lokální IPv4 adresy. Do souboru `/etc/network/interfaces.tail` přidáme řádky se správným rozsahem za předchozí kód:

```
    up ip route add 172.16.20.0/24 dev venet0
    down ip route del 172.16.20.0/24 dev venet0
```

Přidáme routování pro privátní / lokální IPv4 adresu KVM virtuálu na daný bridge tedy `br0` a routování pro veřejnou IPv4 adresu KVM virtuálu na daný bridge `br0`. Do souboru `/etc/network/interfaces.tail` přidáme řádky se správnou privátní / lokální IPv4 adresou a správnou veřejnou IPv4 adresou za výše uvedený kód:

```
    up ip route add 172.16.20.13 dev br0
    down ip route del 172.16.20.13 dev br0
    up ip route add 185.8.164.43 dev br0
    down ip route del 185.8.164.43 dev br0
```

Povolíme předávání datagramů z jednoho síťového rozhraní na druhé (ip_forward/ forwarding) a nastavíme DNS servery IPv4 sítě vpsfree.cz. Do souboru `/etc/network/interfaces.tail` přidáme řádky za výše uvedený kód:

```
    post-up echo 1 > /proc/sys/net/ipv4/ip_forward
    post-up echo 1 > /proc/sys/net/ipv4/conf/all/forwarding
    dns-nameservers  37.205.9.100  37.205.10.88
```

Nastavíme směrování portu 1022 pro SSH VPS (nezapomeňte nastavit také port 1022 u samotné služby SSH na VPS), abychom neztratili přístup k SSH u VPS, vhledem k tomu že síťový provoz bude zcela směrován jen na veřejnou IPv4 adresu KVM virtuálu. Do souboru `/etc/network/interfaces.tail` přidáme řádky za výše uvedený kód:

```
    up iptables -t nat -A PREROUTING -p tcp --dport 1022 -d 185.8.164.43 -j
DNAT --to 172.16.20.12
    up iptables -t nat -A POSTROUTING -p tcp --sport 1022 -d 172.16.20.12 -j
SNAT --to 185.8.164.43
    down iptables -t nat -D PREROUTING -p tcp --dport 1022 -d 185.8.164.43 -
j DNAT --to 172.16.20.12
    down iptables -t nat -D POSTROUTING -p tcp --sport 1022 -d 172.16.20.12
-j SNAT --to 185.8.164.43
```

Soubor `/etc/network/interfaces.tail` uložíme a v tuto chvíli je nutné VPS tedy OpenVZ kontejner restartovat aby se daná nastavení v souboru `interfaces.tail` uplatnila a je hotovo.

*** Lokální IPv4 adresa KVM virtuálu za NATem, veřejná IPv4 u VPS**

Tento návod byl otestován na VPS Debian 8 OpenVZ kontejneru, jako KVM virtuál byl použit systém Ubuntu 16.04.1 LTS x86_64.

Máme lokální IPv4 adresu `192.168.122.2` (rozsah v rámci sítě VPS ⇔ KVM virtuál) za NATem dané VPS a na tuto lokální IPv4 adresu `192.168.122.2` směrujeme jednotlivé porty pro jednotlivé dané služby běžící na KVM virtuálu. KVM virtuál má plnou konektivitu pouze do internetu přes lokální IPv4 adresu `192.168.122.2`, směrem z internetu jsou dostupné pouze předem dané porty služeb běžících na KVM virtuálu a to přes veřejnou IPv4 adresu `185.8.164.43` VPS.

VPS má vlastní veřejnou IPv4 adresu `185.8.164.43` a má plnou konektivitu z / do internetu přes veřejnou IPv4 adresu `185.8.164.43` (kromě portů směrovaných na služby KVM virtuálu). VPS má také lokální IPv4 adresu `192.168.122.1` (rozsah v rámci sítě VPS ⇔ KVM virtuál) která slouží jako brána pro NAT VPS.

Nejprve je potřeba ve vpsAdminu (v detailu nastavení vaší VPS), zapnout / zkontrolovat vlastnosti: Bridge, iptables, KVM a TUN/TAP.

Dále nakonfigurujte bridge `br0` pro KVM virtuál na straně VPS. Jelikož OpenVZ přepisuje soubor `/etc/network/interfaces`, konfiguraci bridge `br0` přidáváme do `interfaces.tail`, OpenVZ ji pak sám po restartu vloží do `/etc/network/interfaces`.

Vytvořte tedy soubor `/etc/network/interfaces.tail` a vložte do něj následující kód (lokální IPv4 adresu VPS si samozřejmě můžete zvolit libovolnou z lokálního rozsahu. Já jsem použil lokální IPv4 adresu v rámci rozsahu `192.168.122.0`:

```
auto br0
iface br0 inet static
    address 192.168.122.1
    netmask 255.255.255.255
    bridge_ports venet0:0
    bridge_stp on
    bridge_fd 0.0
```

Do souboru `/etc/network/interfaces.tail` přidáme routování lokálního rozsahu na rozhraní bridge `br0`. Správný tvar rozsahu pro lokální rozsah IPv4 je dle zvolené lokální IPv4 adresy v předchozím kroku. Do souboru `/etc/network/interfaces.tail` přidáme řádky se správným rozsahem za předchozí kód:

```
    up ip route add 192.168.122.0/24 dev br0
    down ip route del 192.168.122.0/24 dev br0
```

Povolíme předávání datagramů z jednoho síťového rozhraní na druhé (ip_forward) a nastavíme NAT a IP maškarádu mezi bridge `br0` a rozhraní VPS tedy `venet0`. Do souboru `/etc/network/interfaces.tail` přidáme řádky za výše uvedený kód:

```
    post-up echo 1 > /proc/sys/net/ipv4/ip_forward
    up iptables --table nat --append POSTROUTING --out-interface venet0 -j
```

```
MASQUERADE
    up iptables --append FORWARD --in-interface br0 -j ACCEPT
```

Vzhledem k tomu že OpenVZ již po této úpravě automaticky nevloží do `/etc/network/interfaces` konfiguraci pro veřejnou IPv4 adresu, vložíme ještě na konec souboru `/etc/network/interfaces.tail` konfiguraci pro veřejnou IPv4 adresu VPS dle aktuální nastavené konfigurace ve vpsAdminu a nastavíme DNS servery IPv4 sítě vpsfree.cz:

```
auto venet0:0
iface venet0:0 inet static
    address 185.8.164.43
    netmask 255.255.255.255
    dns-nameservers  37.205.9.100  37.205.10.88
```

Soubor `/etc/network/interfaces.tail` uložíme a v tuto chvíli je nutné VPS tedy OpenVZ kontejner restartovat aby se daná nastavení v souboru `interfaces.tail` uplatnila do `interfaces` a vytvořil se bridge `br0`.

Například pomocí `virt-manager` vytvoříme KVM virtuál a nakonfigurujeme síťové rozhraní směřující na bridge `br0` VPS (důležité vybrat bridge `br0`). Nastartujeme OS KVM virtuálu a v systému nakonfigurujeme jeho síť dle vzoru se správnými parametry (lokální IPv4 adresu KVM virtuálu si zvolíme dle použitého lokálního rozsahu v předchozích krocích, kde brána IPv4 je již zvolená IPv4 adresa pro bridge `br0`, tedy lokální IPv4 adresa pro VPS OpenVZ kontejner):

```
IP adresa:  192.168.122.2
Maska sítě: 255.255.255.0
Brána:      192.168.122.1
DNS servery:    37.205.9.100,  37.205.10.88
```

Nyní je čas na nastavení směrování portů (port forwarding) pro dané služby na KVM virtuálu, které boudou na daných portech dostupné na veřejné IPv4 adrese VPS. Toto nastavení provedeme pomocí skriptu (tento skript je převzat a upraven dle oficiálního návodu k libvirt, zdroj zde) umístěného v souboru `/etc/libvirt/hooks/qemu`

Vytvoříme tedy soubor `/etc/libvirt/hooks/qemu` a vložíme následující kód:

```
#!/bin/bash
# used some from advanced script to have multiple ports: use an equal number
of guest and host ports

# Update the following variables to fit your setup
Guest_name='kvm-virtual'
Guest_ipaddr='192.168.122.2'
Guest_port=( '80' '443' )
Host_ipaddr='185.8.164.43'
Host_port=(  '80' '443' )

length=$(( ${#Host_port[@]} - 1 ))
if [ "${1}" = "${Guest_name}" ]; then
   if [ "${2}" = "stopped" ] || [ "${2}" = "reconnect" ]; then
       for i in `seq 0 $length`; do
```

```
               iptables -t nat -D PREROUTING -d ${Host_ipaddr} -p tcp --
dport ${Host_port[$i]} -j DNAT --to ${Guest_ipaddr}:${Guest_port[$i]}
               iptables -D FORWARD -d ${Guest_ipaddr}/32 -p tcp -m state --
state NEW -m tcp --dport ${Guest_port[$i]} -j ACCEPT
               iptables -t nat -D PREROUTING -d ${Host_ipaddr} -p udp --
dport ${Host_port[$i]} -j DNAT --to ${Guest_ipaddr}:${Guest_port[$i]}
               iptables -D FORWARD -d ${Guest_ipaddr}/32 -p udp -m state --
state NEW -m udp --dport ${Guest_port[$i]} -j ACCEPT
       done
   fi
   if [ "${2}" = "start" ] || [ "${2}" = "reconnect" ]; then
       for i in `seq 0 $length`; do
               iptables -t nat -A PREROUTING -d ${Host_ipaddr} -p tcp --
dport ${Host_port[$i]} -j DNAT --to ${Guest_ipaddr}:${Guest_port[$i]}
               iptables -I FORWARD -d ${Guest_ipaddr}/32 -p tcp -m state --
state NEW -m tcp --dport ${Guest_port[$i]} -j ACCEPT
               iptables -t nat -A PREROUTING -d ${Host_ipaddr} -p udp --
dport ${Host_port[$i]} -j DNAT --to ${Guest_ipaddr}:${Guest_port[$i]}
               iptables -I FORWARD -d ${Guest_ipaddr}/32 -p udp -m state --
state NEW -m udp --dport ${Guest_port[$i]} -j ACCEPT
       done
   fi
fi
```

V souboru ještě před uložením skriptu `/etc/libvirt/hooks/qemu` musíme nastavit správné konstanty ve skriptu a to následovně:

```
Guest_name='kvm-virtual'    (vyplňte název virtuálu v libvirt, seznam
virtuálů a jejich správné názvy se dají zobrazit příkazem ‚virsh list‘)
Guest_ipaddr='192.168.122.2'    (je IPv4 lokální adresa již samotného KVM
virtuálu)
Guest_port=( '80' '443' )   (definice portů běžících služeb na KVM virtuálu)
Host_ipaddr='185.8.164.43'  (je veřejná IPv4 adresa samotné VPS, na které
spustíme virtuál)
Host_port=(  '80' '443' )   (definice portů na veřejné IPv4 adrese,
směrované na KVM virtuál)
```

V příkladu výše, se směruje port 80 a 443 pro webserver na KVM virtuálu, z veřejné IPv4 adresy VPS (port 80 a 443) na lokální IPv4 adresu (port 80 a 443) KVM virtuálu. Pokud chceme mít také přístup na SSH u KVM virtuálu, je třeba doplnit také dané porty pro SSH na KVM virtuálu (pozor na to , aby port pro SSH nebyl stejný jako je aktuálně SSH u VPS). Vstupní a výstupní port samozřejmě nemusí být stejný.

Soubor `/etc/libvirt/hooks/qemu` uložíme a nastavíme správné atributy souboru skriptu ke spouštění skriptu, příkazem:

```
chmod +x /etc/libvirt/hooks/qemu
```

V tuto chvíli je nejjistější VPS tedy OpenVZ kontejner restartovat aby se všechna daná nastavení uplatnila a je hotovo.

*** Routování veřejné IPv4 do KVM virtuálu, veřejná IPv4 u VPS**

Tento návod byl otestován na VPS Debian 8 OpenVZ kontejneru, jako KVM virtuál byl použit systém Ubuntu 16.04.1 LTS x86_64.

Máme veřejnou IPv4 adresu `185.8.164.43` a tu přidělíme KVM virtuálu a síťový provoz pro danou veřejnou IPv4 adresu `185.8.164.43` KVM virtuálu směrujeme přímo přes veřejnou IPv4 adresu `185.8.164.36` VPS na veřejnou IPv4 adresu `185.8.164.43` KVM virtuálu. KVM virtuál má plnou konektivitu z / do internetu přes veřejnou IPv4 adresu `185.8.164.43`.

VPS má vlastní veřejnou IPv4 adresu `185.8.164.36`a má plnou konektivitu z / do internetu přes veřejnou IPv4 adresu `185.8.164.36`.

Nejprve je potřeba ve vpsAdminu (v detailu nastavení vaší VPS), zapnout / zkontrolovat vlastnosti: Bridge, iptables, KVM a TUN/TAP.

Dále nakonfigurujte bridge `br0` pro KVM virtuál na straně VPS. Jelikož OpenVZ přepisuje soubor `/etc/network/interfaces`, konfiguraci bridge `br0` přidáváme do `interfaces.tail`, OpenVZ ji pak sám po restartu vloží do `/etc/network/interfaces`.

Vytvořte tedy soubor `/etc/network/interfaces.tail` a vložte do něj následující kód (veřejnou IPv4 adresu VPS zaměníte za správnou IPv4 adresu z aktuální nastavené konfigurace ve vpsAdminu):

```
auto br0
iface br0 inet static
    address 185.8.164.36
    netmask 255.255.255.255
    bridge_ports venet0:0
    bridge_stp on
    bridge_fd 0.0
    dad-attempts 0
    accept_ra 0
```

Do souboru `/etc/network/interfaces.tail` přidáme routování veřejného rozsahu na rozhraní VPS tedy `venet0`.

Správný tvar pro veřejný rozsah k dané veřejné IPv4 adrese nalezneme ve vpsAdminu zde. Do souboru `/etc/network/interfaces.tail` přidáme řádky se správným rozsahem za předchozí kód:

```
    up ip route add 185.8.164.0/32 dev venet0
    down ip route del 185.8.164.0/32 dev venet0
```

Přidáme routování pro veřejnou IPv4 adresu KVM virtuálu na daný bridge tedy `br0`. Do souboru `/etc/network/interfaces.tail` přidáme řádky se správnou veřejnou IPv4 adresou za výše uvedený kód:

```
    up ip route add 185.8.164.43 dev br0
    down ip route del 185.8.164.43 dev br0
```

Povolíme předávání datagramů z jednoho síťového rozhraní na druhé (ip_forward/ forwarding) a nastavíme DNS servery IPv4 sítě vpsfree.cz. Do souboru `/etc/network/interfaces.tail` přidáme řádky za výše uvedený kód:

```
    post-up echo 1 > /proc/sys/net/ipv4/ip_forward
    post-up echo 1 > /proc/sys/net/ipv4/conf/all/forwarding
    dns-nameservers  37.205.9.100  37.205.10.88
```

Soubor `/etc/network/interfaces.tail` uložíme a v tuto chvíli je nutné VPS tedy OpenVZ kontejner restartovat aby se daná nastavení v souboru `interfaces.tail` uplatnila a vytvořil se bridge `br0`.

V systému KVM virtuálu nakonfigurujeme jeho síť dle vzoru se správnými parametry (veřejnou IPv4 adresu pro KVM virtuál jsme si již vybrali v předchozím kroku, zadáme stejnou a brána je IPv4 adresa VPS již zvolená pro bridge `br0`, tedy veřejná IPv4 adresa pro VPS OpenVZ kontejner):

```
IP adresa:  185.8.164.43
Maska sítě: 255.255.255.255
Brána:      185.8.164.36
DNS servery:    37.205.9.100  37.205.10.88
```

A je hotovo.

*** Routování veřejné IPv6 do KVM virtuálu, veřejná IPv6 u VPS**

Tento návod byl otestován na VPS Debian 8 OpenVZ kontejneru, jako KVM virtuál byl použit systém Ubuntu 16.04.1 LTS x86_64.

Máme veřejnou IPv6 adresu `2a03:3b40:3::52` a tu přidělíme KVM virtuálu a síťový provoz pro danou veřejnou IPv6 adresu `2a03:3b40:3::52` KVM virtuálu směrujeme přímo přes veřejnou IPv6 adresu `2a03:3b40:3::47` VPS na veřejnou IPv6 adresu `2a03:3b40:3::52` KVM virtuálu. KVM virtuál má plnou konektivitu z / do internetu přes veřejnou IPv6 adresu `2a03:3b40:3::52`.

VPS má vlastní veřejnou IPv6 adresu `2a03:3b40:3::47` a má plnou konektivitu z / do internetu přes veřejnou IPv6 adresu `2a03:3b40:3::47`.

Následující návod předpokládá již provedení kroků konfigurace IPv4 pomocí jednoho z následujících postup v těchto odstavcích:

* „Routování veřejné IPv4 přes lokální rozsah do KVM virtuálu, lokální IPv4 u VPS“,

* „Lokální IPv4 adresa KVM virtuálu za NATem, veřejná IPv4 u VPS“ či

* „Routování veřejné IPv4 do KVM virtuálu, veřejná IPv4 u VPS“

tohoto návodu výše.

Bez těchto předchozích kroků nelze následující návod pro IPv6 použít v této formě!

Do existujícího souboru `/etc/network/interfaces.tail` vložte následující kód za existující kódu pro definování IPv4 bridge `br0` (veřejnou IPv6 adresu VPS zvolíte správnou z aktuální nastavené konfigurace ve vpsAdminu):

```
iface br0 inet6 static
    address 2a03:3b40:3::47
    netmask 128
    autoconf 0
    dad-attempts 0
    accept_ra 0
```

Do souboru `/etc/network/interfaces.tail` přidáme routování IPv6 rozsahu na rozhraní VPS tedy `venet0` a routování pro veřejnou IPv6 adresu KVM virtuálu na daný bridge `br0`.

Správný tvar pro rozsah IPv6 nalezneme ve vpsAdminu zde a danou veřejnou IPv6 adresu pro KVM virtuálu nastavíme správnou dle aktuální nastavené konfigurace ve vpsAdminu.

Do souboru `/etc/network/interfaces.tail` přidáme řádky se správným rozsahem a správnou veřejnou IPv6 adresou pro KVM virtuál za předchozí kód:

```
    up route -A inet6 add 2a03:3b40:3::/128 dev venet0
    down route -A inet6 del 2a03:3b40:3::/128 dev venet0
    up route -A inet6 add 2a03:3b40:3::52 dev br0
    down route -A inet6 del 2a03:3b40:3::52 dev br0
```

Povolíme předávání datagramů z jednoho síťového rozhraní na druhé (forwarding) a nastavíme DNS servery IPv6 sítě vpsfree.cz. Do souboru `/etc/network/interfaces.tail` přidáme řádky za výše uvedený kód:

```
    post-up echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
    dns-nameservers 2a01:430:17:1::ffff:666 2a01:430:17:1::ffff:588
```

V systému KVM virtuálu nakonfigurujeme jeho síť dle vzoru se správnými parametry (veřejnou IPv6 adresu pro KVM virtuál jsme si již vybrali v předchozím kroku, zadáme stejnou a brána je IPv6 adresa již zvolená pro bridge `br0`, tedy veřejná IPv6 adresa pro VPS OpenVZ kontejner):

```
IP adresa:  2a03:3b40:3::52
Maska sítě: 128
Brána:      2a03:3b40:3::47
DNS servery:    2a01:430:17:1::ffff:666, 2a01:430:17:1::ffff:588
```

Soubor `/etc/network/interfaces.tail` uložíme a v tuto chvíli je nutné VPS tedy OpenVZ kontejner restartovat aby se daná nastavení v souboru `interfaces.tail` uplatnila a je hotovo.

KVM na Debian 9

Nefunguje více info v https://lists.vpsfree.cz/pipermail/community-list/2017-July/004532.html

Kontakty

- pavlix (CentOS, Debian)
- jirutka (Alpine Linux)