Provide outgoing IPv4 connectivity to Docker Container on a IPv6-only Host via clatd (464xlat)

Date: 2022-04-10
modified: 2022-04-10
tags: Linux, Docker, Container, IPv6, NAT
description: Docker Container on IPv6-Only Host with clatd (464xlat)
category: Linux
slug: provide_outgoing_ipv4_connectivity_for_docker_container_on_a_ipv6-only_host_via_clatd_464xlat

Author: Dominik Wombacher

lang: en
transid: provide_outgoing_ipv4_connectivity_for_docker_container_on_a_ipv6-only_host_via_clatd_464xlat
Status: published

My experiences are based on a recent Harbor setup on Rocky Linux, but should be easily transferable to any other Application and Linux Distribution.

The journey began after Harbor was up and running and the vulnerability Scanner Trivy, used to Scan pushed Images, failed to get it's updates from a IPv4-only address.

So the goal was to get outgoing IPv4 working, incoming traffic is handled by a HAProxy and not further described.

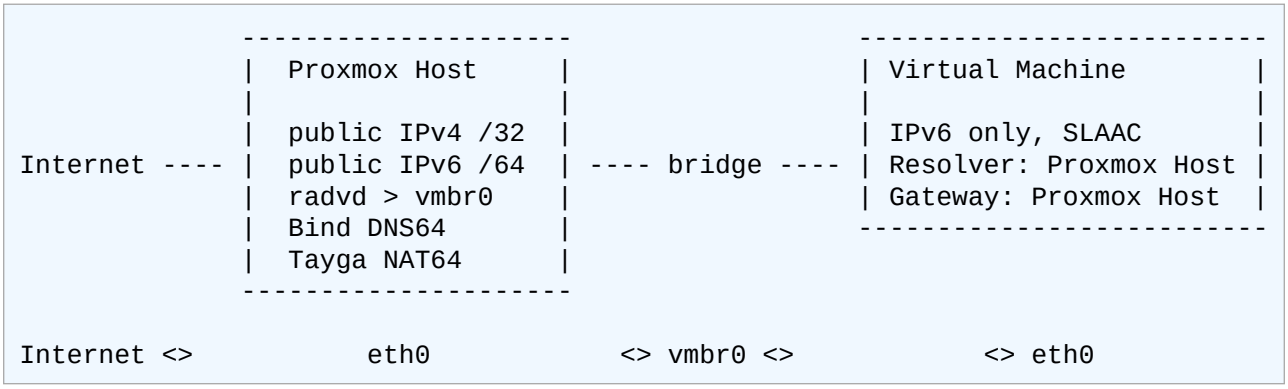
Some background about the Infrastructure: The Proxmox VE Server that host the VM has public IPv4 and IPv6 connectivity, is running radvd to advertise a /64 prefix to a bridge interface, Bind / Named to provide DNS64 and Tayga for NAT64.

This Setup allows the VM to automatically configure IPv6 via SLAAC and also talk to IPv4 targets, e.g. github.com, out of the box. DNS Resolution is done by Bind with enabled DNS64 option on the Proxmox Host, which also act as Gateway for the VM.

If no AAAA Record is available for a site, Bind will return one where the IPv4 address is embedded in a IPv6 address. In my setup I defined `64:ff9b::/96` for that purpose, example result for github.com (IPv4: 140.82.121.4):

```
dig github.com aaaa +short
64:ff9b::8c52:7903
```

Tayga, running on the Proxmox Host, will receive this IPv6 traffic from the VM and perform NAT64, viola github.com is accessible from the IPv6-only VM. More about that setup in a earlier Post.



If you put Docker into the mix, things getting complicated due to the fact that everything was build with IPv4 in mind.

Even though there is some sort of IPv6 Support in the meantime, it's not really usable in my opinion. You can achieve that Docker assigns a IPv6 from a given Prefix (recommend: /80) to each Container. But that need to be configured in `/etc/docker/daemon.json` for the default network and separately via cli or docker-compose for every custom network.

It's a random public reachable IP address and all ports will exposed, independent of the Container configuration. Also managing DNS Records for those random generated, not predictable Container IPs, is nearly impossible and SLAAC or at least DHCPv6 is not supported at all.

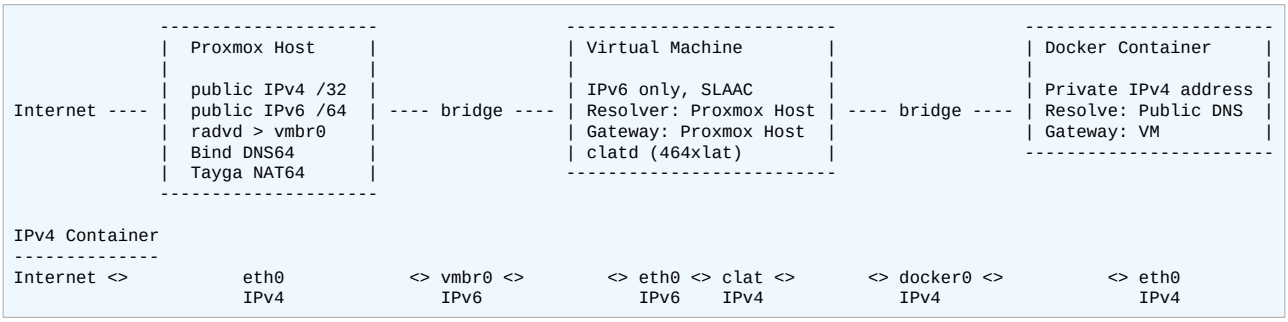
Apropos DNS, Docker embedded DNS, which is enabled on Custom Networks and can't be disabled, only support IPv4. So as soon you work with custom networks, which is quite common with Docker Compose, you have to set it globally via `/etc/docker/daemon.json`, which is only a good idea if all you container going to use IPv6 or alternatively manually overwrite `/etc/resolv.conf` inside the specific Container, e.g. by read-only mount a file, to get name resolution working.

Also you can't disable IPv4, so your Container will always get a (private) IPv4 in addition to the public IPv6 address. That means on a IPv6 only Host, NAT64 as described above doesn't work and connecting to IPv4 only targets like github.com will fail.

You can use ipv6nat to get rid of some of the earlier mentioned limitations, but this doesn't solve the IPv4 connectivity problem and still requires to setup IPv6 prefixes manually for every docket network.

After investing quite some time in IPv6 and Docker I came to the conclusion to stay away from it, at least until embedded DNS support IPv6 and disabling IPv4 is possible.

I decided to run Docker just as ever, with NAT and private IP4 addresses assigned to each container. To provide the necessary Translation from IPv4 > IPv6 I use clatd, which use Tayga under the hood and performs the heavy lifting for your.



A quick look on the setup, focused on the clatd related parts, assumption that docker is already up and running. Probably the most important is to allow masquerade in the public firewalld zone, otherwise the packages will silently dropped.

```
# Allow masquerade to avoid silent package drop
sudo firewall-cmd --zone=public --add-masquerade --permanent
sudo firewall-cmd --reload

# Install EPEL Repositories, required for clatd and tayga
sudo nf config-manager --set-enabled powertools
sudo dnf install epel-release

sudo dnf install make git tar

# Download and install clatd
cd ~
git clone https://github.com/toreanderson/clatd
sudo make -C clatd install installdeps
```

```
sudo systemctl enable --now clatd

# Configure global IPv4 DNS Server, used by all Container
# Privacy friendly Server by Freifunk München https://ffmuc.net/wiki/doku.php?id=knb:dohdot
/etc/docker/daemon.json
```
```
{
  "dns" : [ "5.1.66.255", "185.150.99.255" ]
}
```

```
sudo systemctl restart docker
```

Drawback: That way Container will _not_ be able to reach IPv6-only Systems, the target need to be reachable via IPv4.

At least in my case that's acceptable, main goal is to ensure Trivy can download updates, if you need outgoing IPv6, you have to go either with ipv6nat or the docker build-in IPv6 feature.

There seem to be no perfect one-size-fits-all Solution, so as often in IT, you have to pick the right tool for the job.