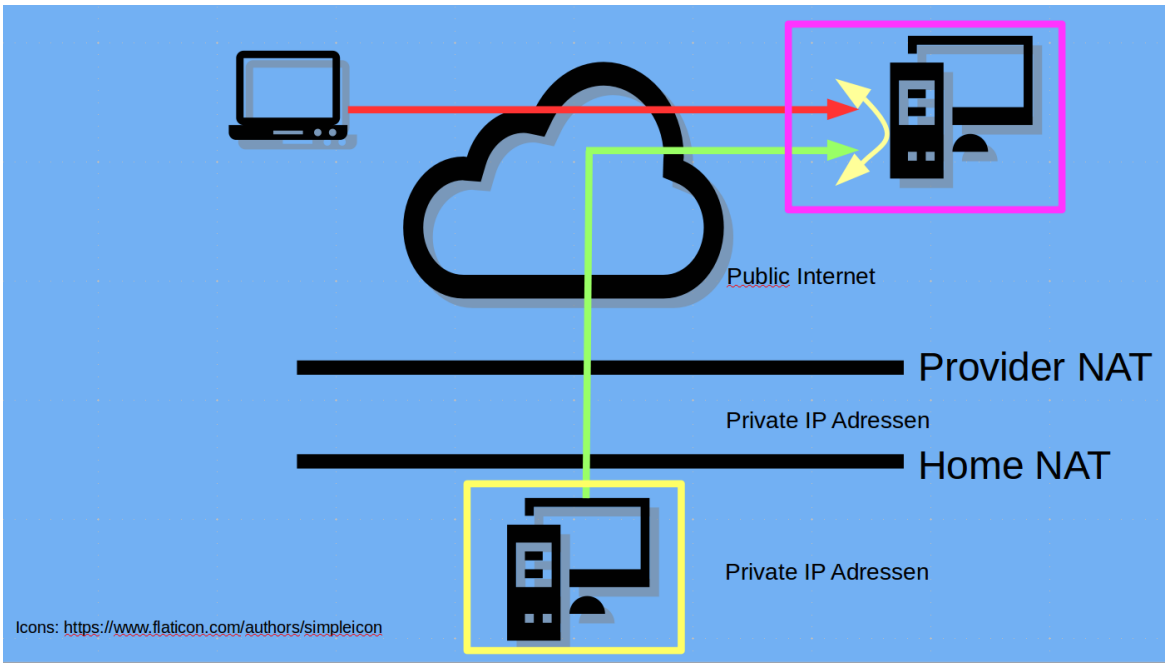
SSH-Tunnel

How To Run A Server At Home Without An IPv4 Address



Once upon a time the Internet was bidirectional and everyone could run a server at their end. Unfortunately, these days are long gone and many ISPs today, especially cable providers, do not assign a public IPv4 address to their customers. Not even when you ask them nicely. Not even for money, unless you are a business customer who is willing to pay through the nose for the privilege. Fortunately, there is a way to run servers at home and make them accessible to the outside world and an easy one at that. The following text and shell commands are from a talk I gave at GPN19 (in German).

The program that makes this straight forward is ssh, or secure shell. Despite its name, ssh is the Swiss army knife when it comes establishing tunnels between different parts of the Internet through which TCP packets can be forwarded.

So making a server at home available despite a provider double NAT can be done by renting a server with a public IP address, preferably from a local cloud provider and then use it for forwarding

packets. In my case, I pay €3 a month for such a server, that, by the way, I also use to run this blog on!

The image on the left shows the principle and the following commands show how the tunnel is configured. In the example, the server at home uses TCP port 7324 but should be accessible from the Internet on TCP port 8080. One would usually use the same ports but I thought an example with different ports is better to show if the local or the remote port is meant in my example below. In addition I use TCP port 39122 for ssh on the remote server instead of port 22 as my access log would otherwise be full with connection requests from bots on a mission.

Once things are working, autossh and crontab can be used to put the tunnel process into the background, to restore the tunnel automatically when it fails and to even survive reboots without administrative action. Have fun!

```
##### Test Apache, local works, remote does not, no tunnel yet.
http://localhost:7324/apollo.jpg
http://remote-server.gpn-demo.de:8080/apollo.jpg



#home-srv: Generate ssh keys for normal user.
#These are used to authenticate on the remote server
#
ssh-keygen -t rsa -b 4096 -c "user@clienthost"



#home-srv: Display ssh public key and copy it
#
cat ~/.ssh/id_rsa.pub



#remote-srv: Put public key into root account's
#'authorized_keys'
#
sudo nano /root/.ssh/authorized_keys



#remote-srv: Configure SSH daemon to:
# - allow tunnel ports to be used by incoming requests
#   from the Internet (Gateway)
# - timeouts for stale connections

sudo nano /etc/ssh/sshd_config
```

```
GatewayPorts yes
ClientAliveInterval 60
ClientAliveCountMax=2

sudo service sshd restart

#remote-server: Open second shell and show open tcp ports
#
watch -n 0.5 "netstat -tulpn"


#home-server: Test establishment of tunnel from
#home-srv (local) port 7324 to remote-srv 8080. First time
#around, fingerprint needs to be confirmed!
#
ssh -p 39122 -N -R 8080:localhost:7324 root@remote-server.gpn-demo.de

##### Test Apache, local works and remote works too now!

#CTRL-C ssh command

##### Test Apache, local works, remote does NOT work because tunnel is gone!

#home-server: Once working, use same command with '-f' option to put to the background
#
autossh -M 0 -f -o ConnectTimeout=10 -o ServerAliveInterval=60 -o ServerAliveCountMax=2 \
-p 39122 -N -R 8080:localhost:7324 root@remote-server.gpn-demo.de


#Simulate ERROR scenario - kill ssh connection on remote side
#remote-server: Terminate process that handles port
#8080 (see pid in 'watch')
#
kill XXXX

##### Test Apache, local works, remote still works!

###### Finally
```

```
#home-server: Run autossh command on startup
#
crontab -e

### IMPORTANT: make this ONE line, crontab doesn't
### like the backslash!
@reboot autossh -M 0 -f -o ConnectTimeout=10 -o ServerAliveInterval=60 -o
ServerAliveCountMax=2 ......
-p 39122 -N -R 8080:localhost:7324 root@remote-server.gpn-demo.de

DONE!!!
```

https://blog.wirelessmoves.com/2019/06/how-to-run-a-server-at-home-without-an-ipv4-address.html

Rapimc:

SSH-Tunnel raspimc: (root crontab)

@reboot sleep 20 &&/usr/bin/mount_all
@reboot autossh -M 0 -f -o ConnectTimeout=10 -o ServerAliveInterval=60 -o ServerAliveCountMax=2 -p 48153 -N -R 8081:localhost:22 root@meet.wkmimnl.de
@reboot autossh -M 0 -f -o ConnectTimeout=10 -o ServerAliveInterval=60 -o ServerAliveCountMax=2 -p 48153 -N -R 8082:localhost:8080 root@meet.wkmimnl.de
@reboot autossh -M 0 -f -o ConnectTimeout=10 -o ServerAliveInterval=60 -o ServerAliveCountMax=2 -p 48153 -N -R 8083:localhost:6875 root@meet.wkmimnl.de

---

Revision #5
Created 2023-01-07 12:10:22 UTC by willi
Updated 2025-03-04 00:01:58 UTC by willi