LXC container as a 44Net VPN Router/Firewall

Mark Phillips, NI2O 07/2024 V1.3

Contents

Abstract

This document aims to walk the reader through the setting up of an LXC container for use as a 44Net VPN Router and Firewall. While this document speaks to ProxMox and LXC systems it may be useful for other Linux/Mac implementations as the steps and commands are almost identical.

Assumptions

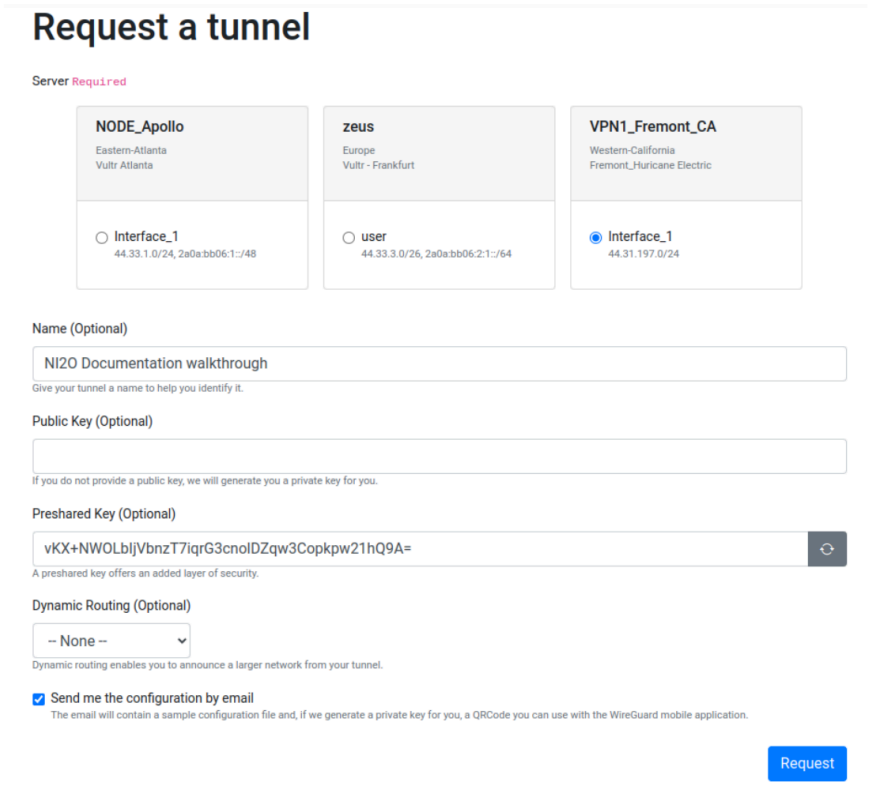
Familiarity with ProxMox Virtual Environment
Familiarity with Linux (Debian) networking
Networking basics

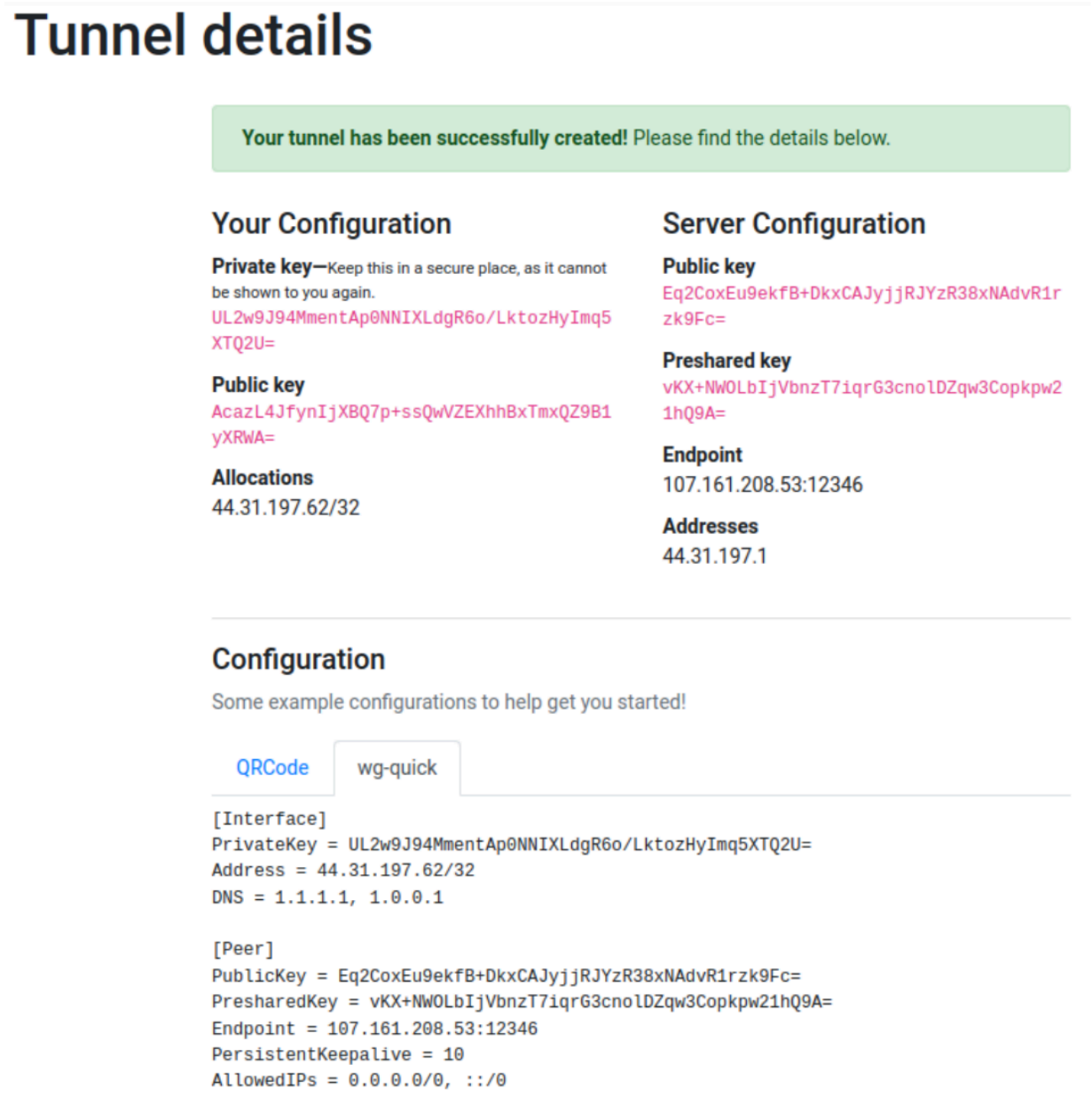
Requirements

Functioning Proxmox LXC environment
Functioning LAN with Internet connectivity

Obtain a POP account

Create a POP access account at https://pop.44net.cloud. Then request a tunnel. Ensure to click the refresh button at the end of the “Preshared Key (optional)” box. A Preshared Key will ensure even greater security (Wireguard is very secure in itself).





Ensure to copy the “wg-quick” information to a text file somewhere for safe keeping. It will NEVER be shown again. Keep this file for later use.

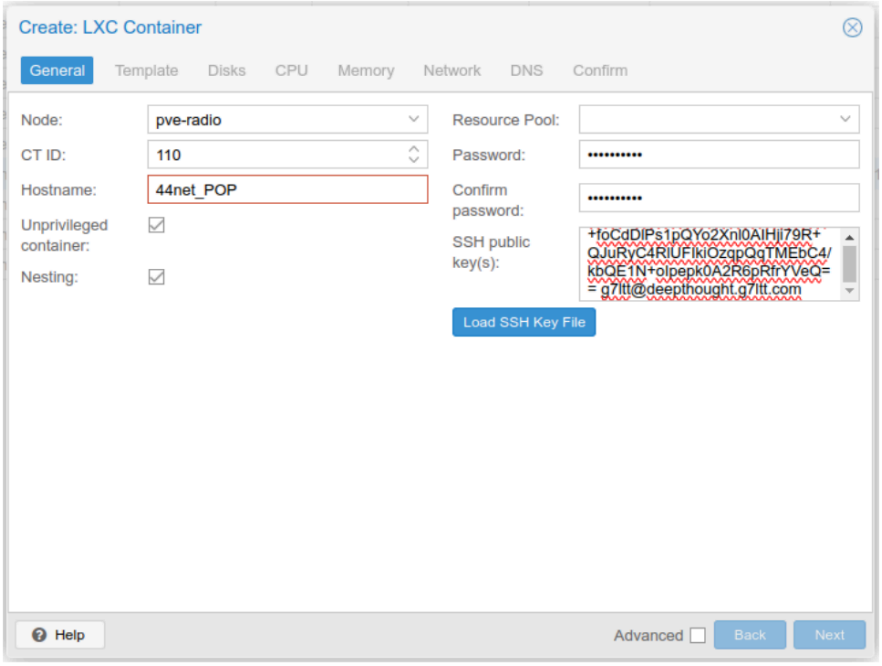
Create an LXC container

On your ProxMox host create a container with the following specifications

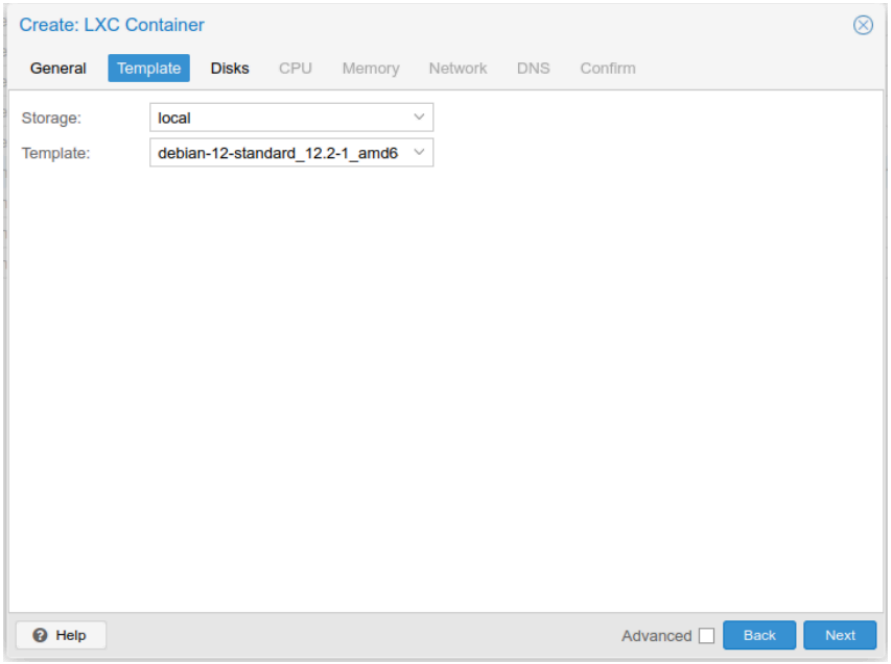
- 512MB memory
- 512MB swap
- 1 CPU
- 1 network adapter

4GB disk space (routers don't need much)
Unprivileged
Nesting

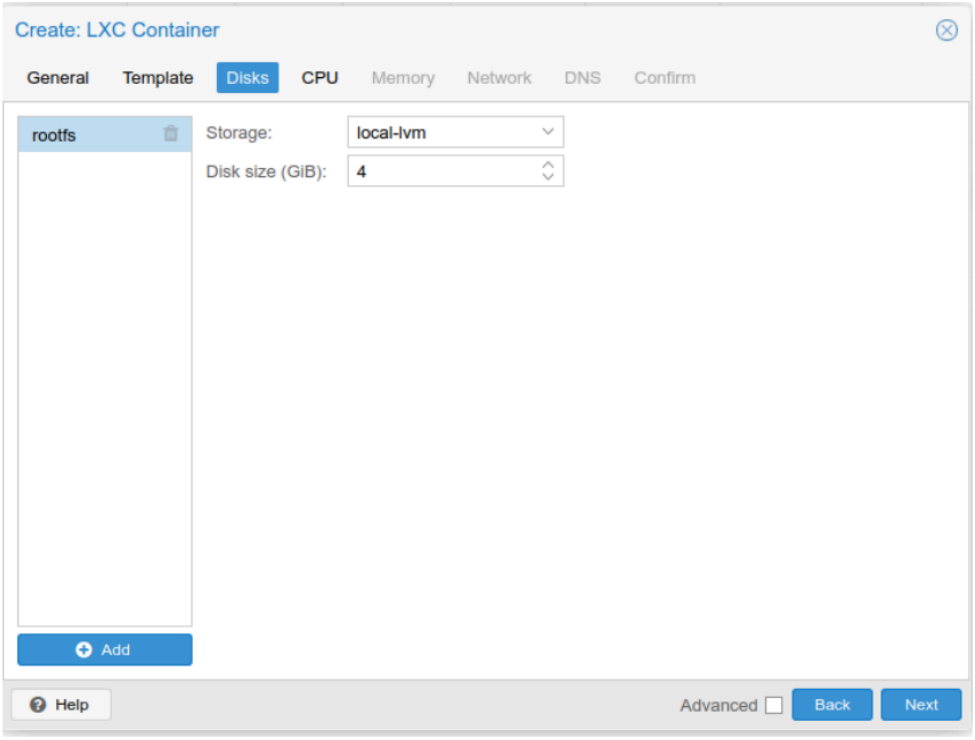
Create the container. Label it in a useful manner. Add both a root password as well as SSH keys to ensure secure remote access if required.



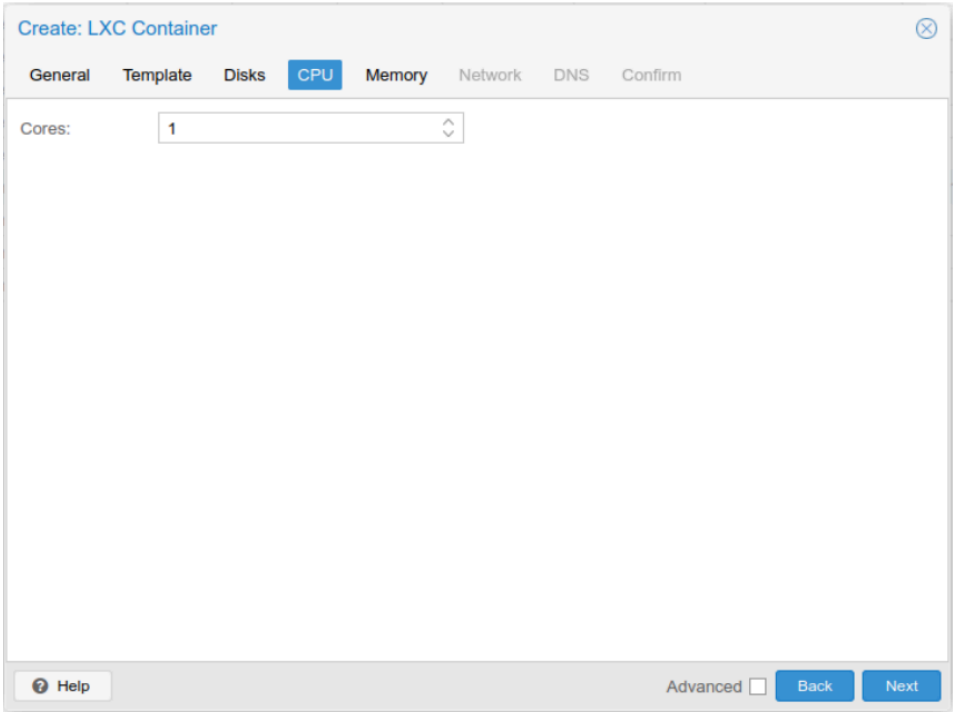
Use your favorite linux distribution as the basis for the container (we use Debian)



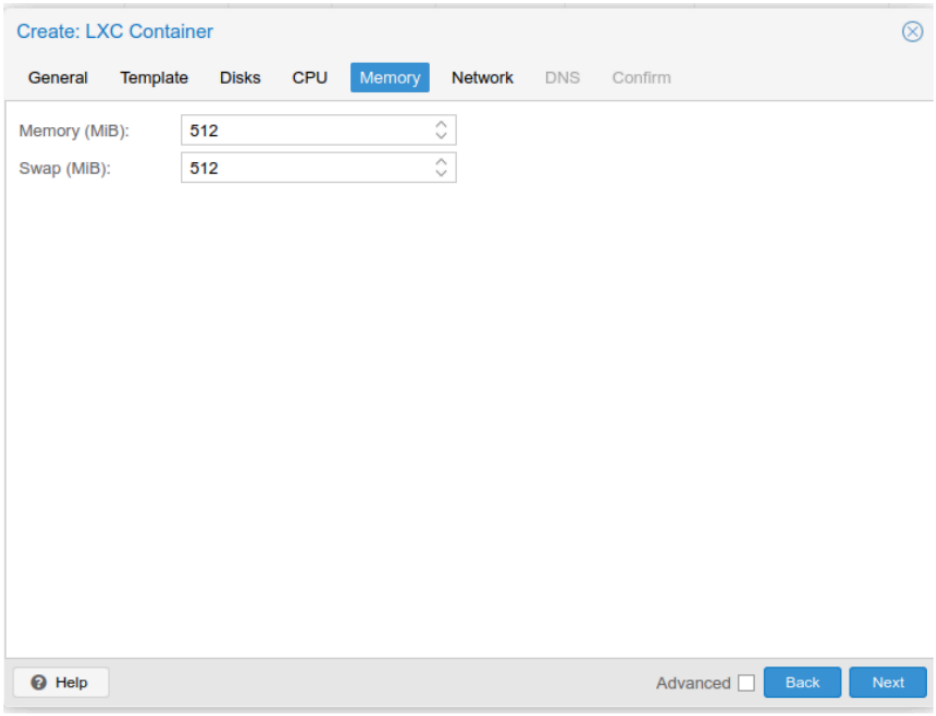
Create a disk with 4GB of space for the operating system



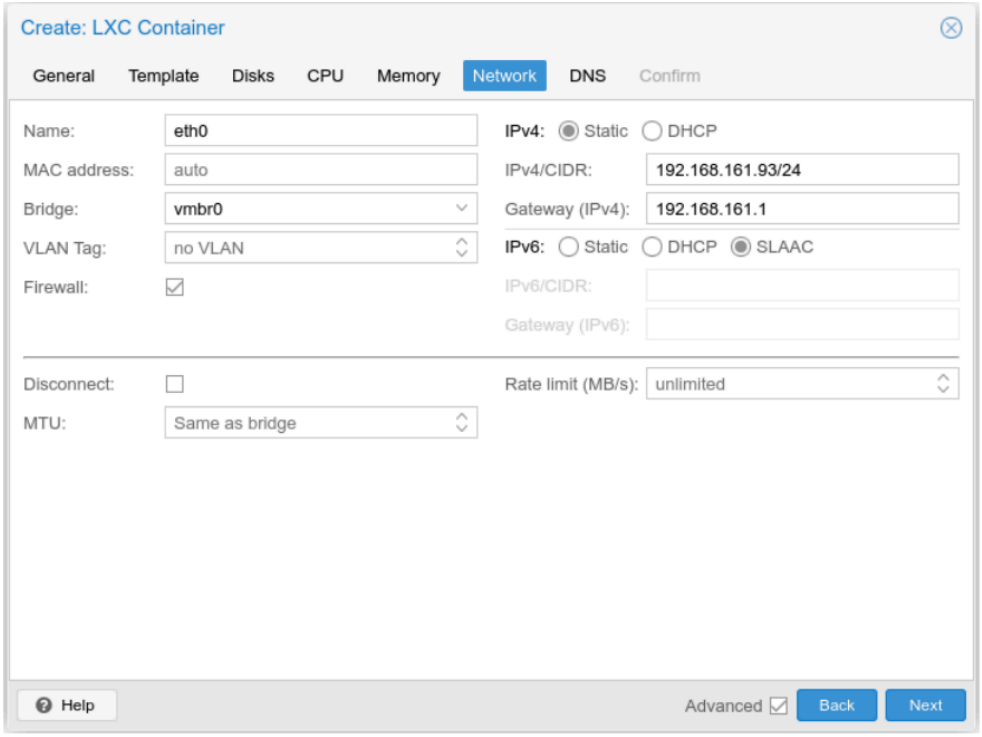
Add just 1 CPU core



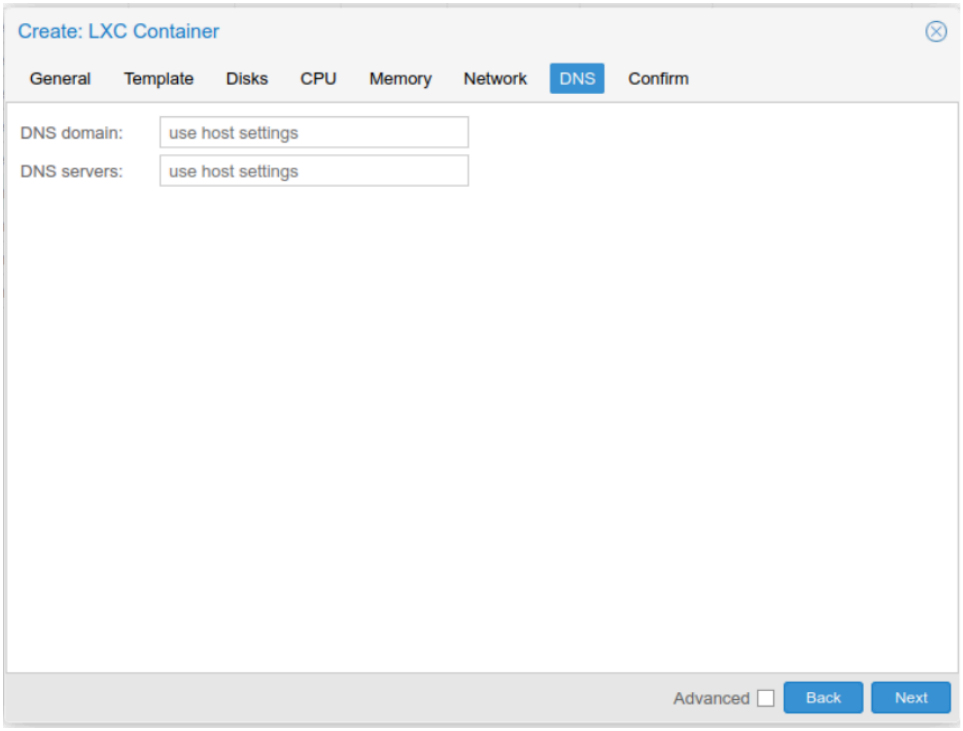
512MB of RAM and SWAP



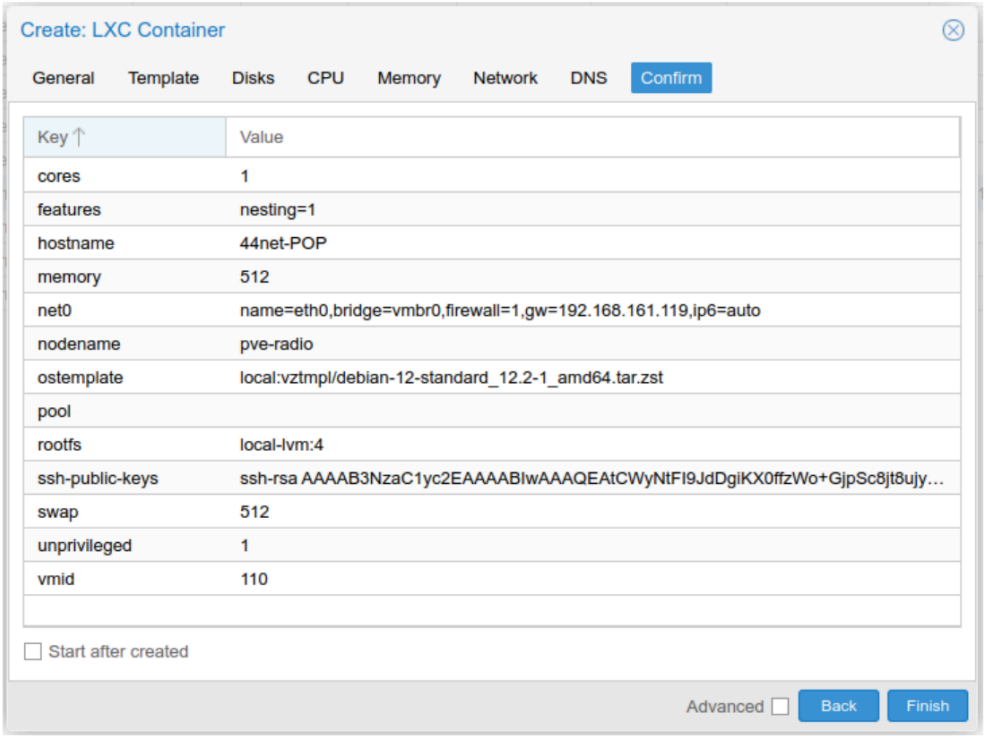
Add a static internal IPv4 address to the network interface



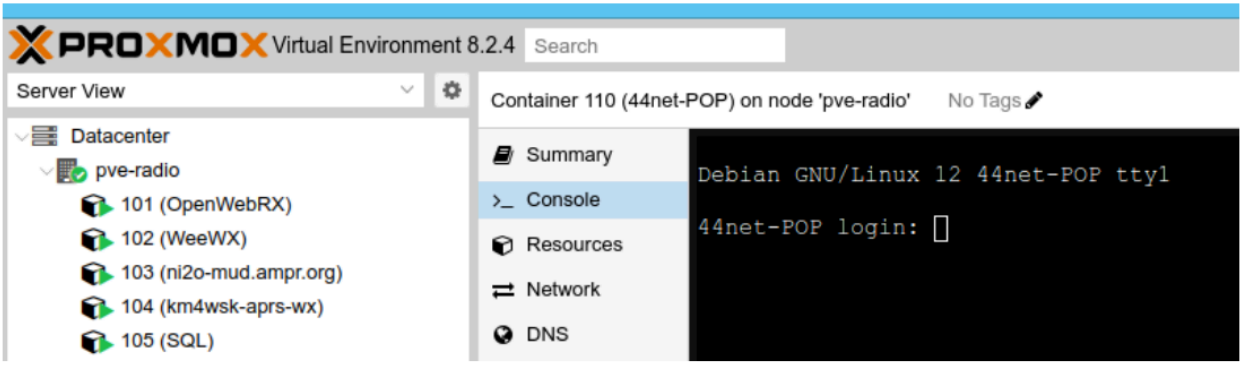
Keep the host DNS settings



Confirm everything is correct



Start the container!



Perform system updates

Log in as root to the container via the console and perform the system updates and also install some applications and tools(Debian assumed)

```
apt update && apt upgrade -y
apt install net-tools wireguard ufw resolvconf mtr
```

Configure the VPN

Now that we have the Wireguard software installed create a VPN config file

```
nano /etc/wireguard/44net-pop.conf
```

Then paste in your wg-quick config file saved from earlier

```
[Interface]
PrivateKey = 8H8ey27QlQq68/H5NOydICzDQbxlE2fWo6t1mo4nfnU=
Address = 44.33.1.32/32, 2a0a:bb06:1::d/128
DNS = 1.1.1.1, 1.0.0.1

[Peer]
PublicKey = CCK2lhIIo1BpAzqfQVUjPhVjFeslZ/q9Vh0AU1LJ218=
PresharedKey = WY8OUQRiErfenglOHzwon8oXGj56vSFj1J9wt0Zfdpo=
Endpoint = 45.32.220.92:12345
PersistentKeepalive = 10
AllowedIPs = 0.0.0.0/0, ::/0
```

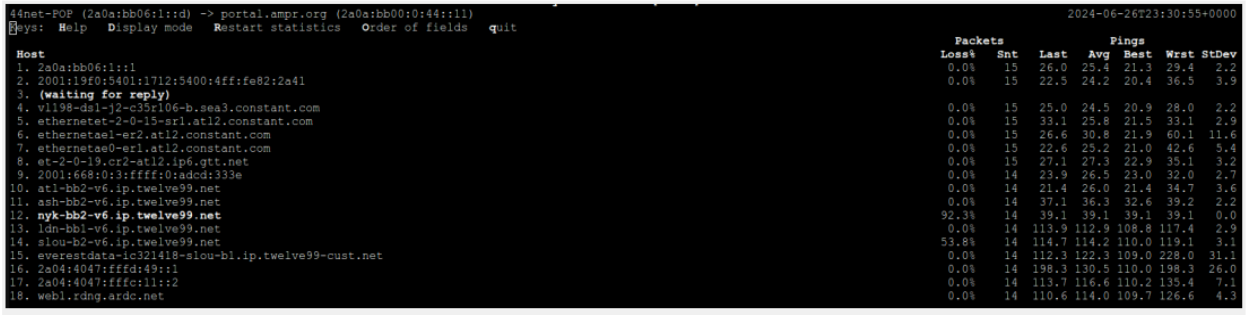
Test the connection

```
wg-quick up 44net-pop
```

```
root@44net-POP:~# wg-quick up 44net-pop
[#] ip link add 44net-pop type wireguard
[#] wg setconf 44net-pop /dev/fd/63
[#] ip -4 address add 44.33.1.32/32 dev 44net-pop
[#] ip -6 address add 2a0a:bb06:1::d/128 dev 44net-pop
[#] ip link set mtu 1420 up dev 44net-pop
[#] resolvconf -a tun.44net-pop -m 0 -x
[#] wg set 44net-pop fwmark 51820
[#] ip -6 route add ::/0 dev 44net-pop table 51820
[#] ip -6 rule add not fwmark 51820 table 51820
[#] ip -6 rule add table main suppress_prefixlength 0
[#] nft -f /dev/fd/63
[#] ip -4 route add 0.0.0.0/0 dev 44net-pop table 51820
[#] ip -4 rule add not fwmark 51820 table 51820
[#] ip -4 rule add table main suppress_prefixlength 0
[#] sysctl -q net.ipv4.conf.all.src_valid_mark=1
[#] nft -f /dev/fd/63
root@44net-POP:~#
```

Try a traceroute to the ARDC portal

```
mtr portal.ampr.org
```



Survive a reboot

We must make the VPN setup survive a reboot so that it all comes back up again when restarted. Use the below commands to create a system “service” that will restart the VPN every time the container is rebooted. The final command reboots the container.

```
systemctl enable wg-quick@44net-pop.service
systemctl daemon-reload
```

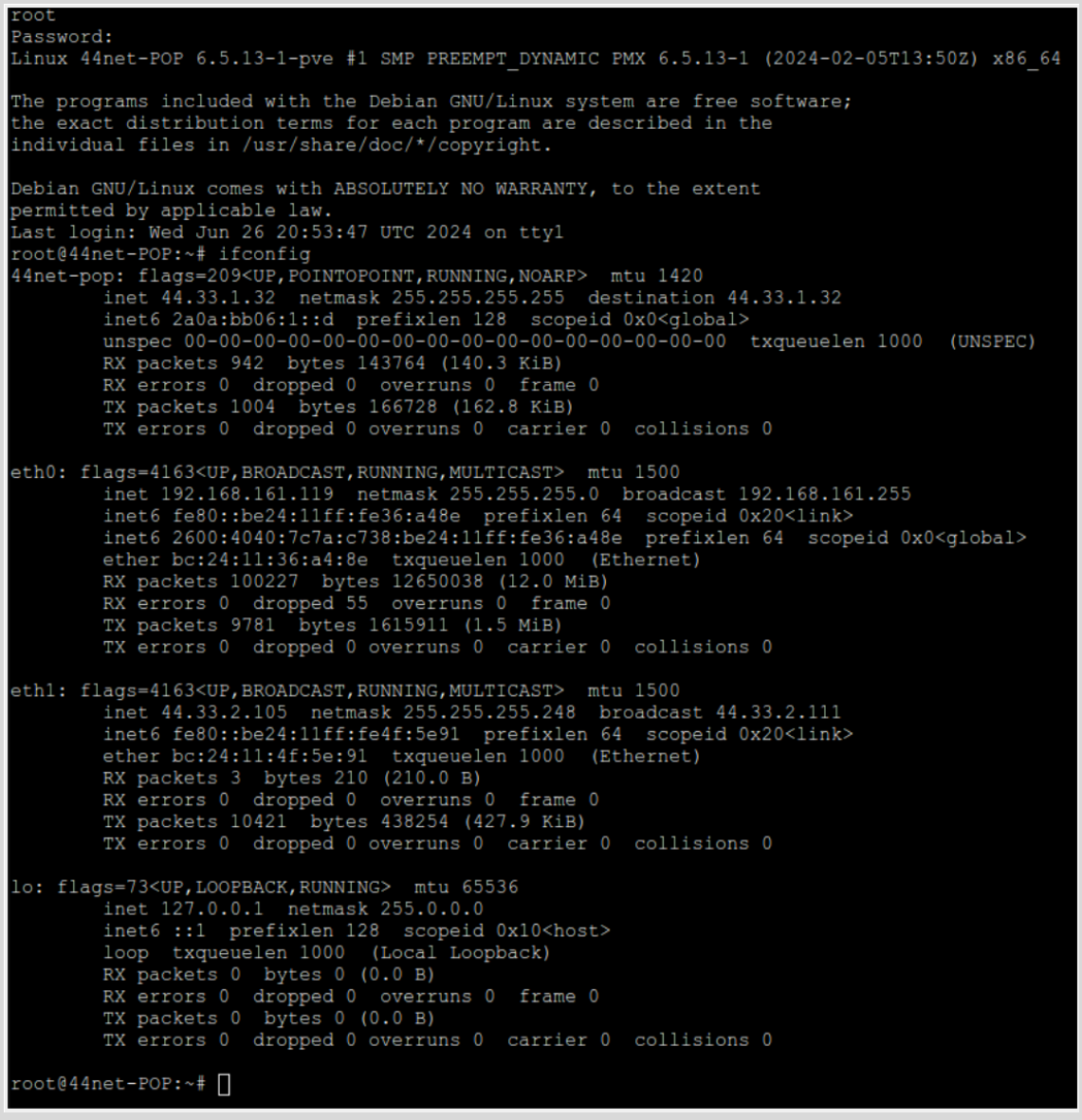
```
systemctl status wg-quick@44net-pop
```

Finally, reboot the system

```
reboot
```

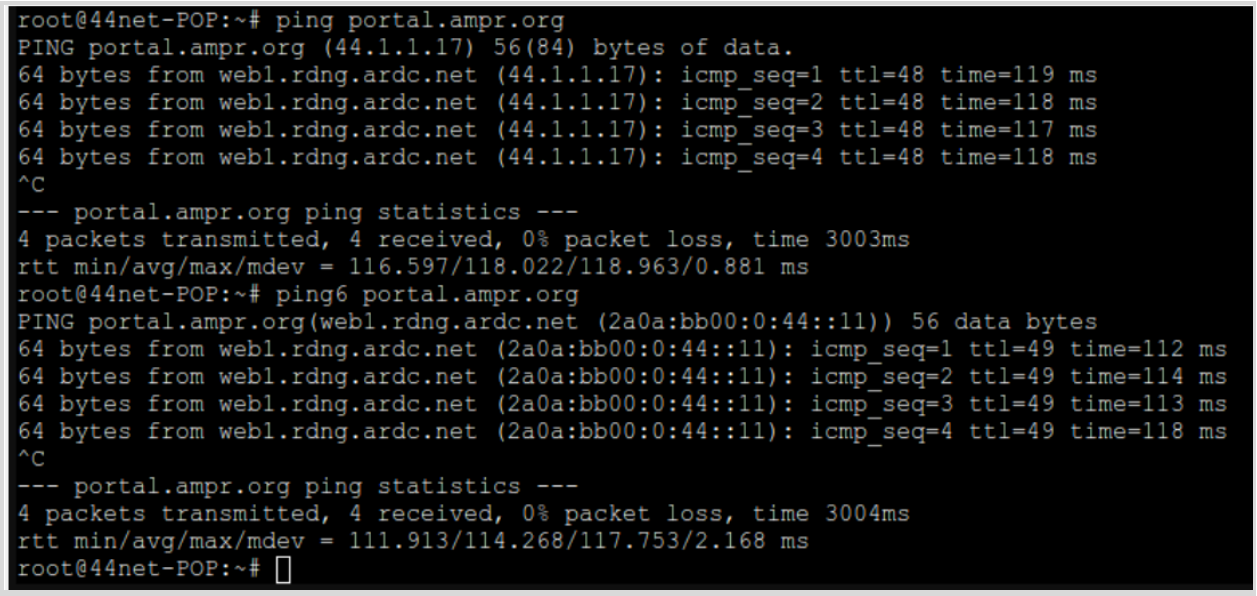
When the container returns log in and then check for the VPN interface

```
ifconfig
```



Ensure both IPv4 and IPv6 connectivity

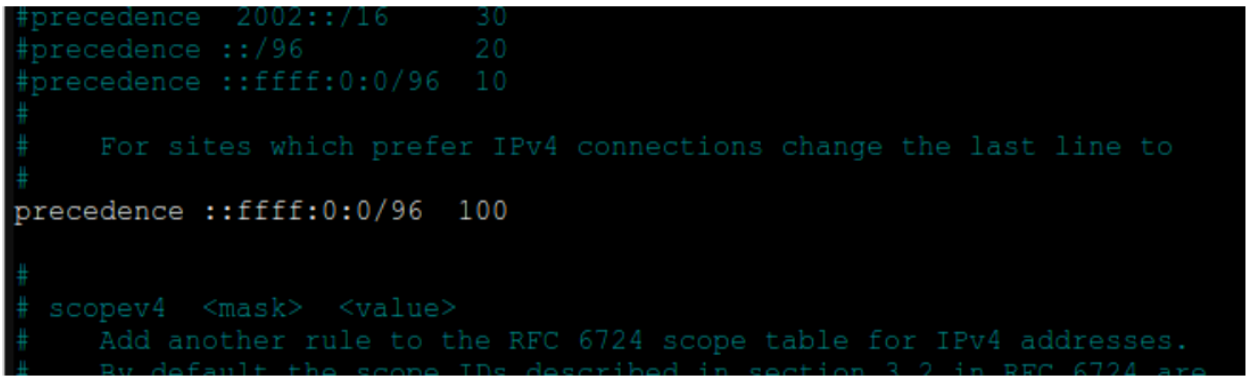
```
ping portal.ampr.org
ping6 portal.ampr.org
```



Force IPv4 default rather than IPv6 (optional)

By default Linux will prefer IPv6 routes rather than the traditional IPv4. This is not an issue as both IPv4 and IPv6 data will pass over the VPN. For your own personal reasons you may prefer to use IPv4 as the default. Edit the /etc/gai.conf file by removing the hash mark from the beginning of the line shown in the example below;

```
nano /etc/gai.conf
```



Restart the VPN tunnel

```
wg-quick down 44net-pop
wg-quick up 44net-pop
```

Then test the tunnel again noting the IPv4 addresses used this time

```
mtr portal.ampr.org
```

Firewall incoming connections (optional but highly recommended!!!!!)

In this document we have created a VPN tunnel from your home to the Great Unwashed Internet and have allowed all that is nasty there to traverse your link. If Freedom of Speech is your thing, you are finished. If you want to protect yourself, others and your equipment I would recommend you install a firewall to stop the miscreants on the Net from hacking your systems.

Earlier in this document we installed a few applications including a firewall tool called Uncomplicated Firewall (UFW). To turn on the firewall simply type

```
ufw enable
```

This will turn on the firewall and protect your system from incoming threats. By default UFW will allow your outgoing connections but disallow any incoming connections just like your home router/firewall. It can be configured to allow port forwarding and other firewall services as you see fit. It is not within the scope of this document to explain the workings of the Uncomplicated Firewall. Further information can be found at
https://manpages.ubuntu.com/manpages/trusty/man8/ufw.8.html

NAT outgoing connections

If you are using your new VPN container as a home style router you will need to NAT the outgoing data so that it appears to emanate from the router itself rather than your desktop machine.

To enable this feature it is necessary to further edit the VPN config file we created earlier;

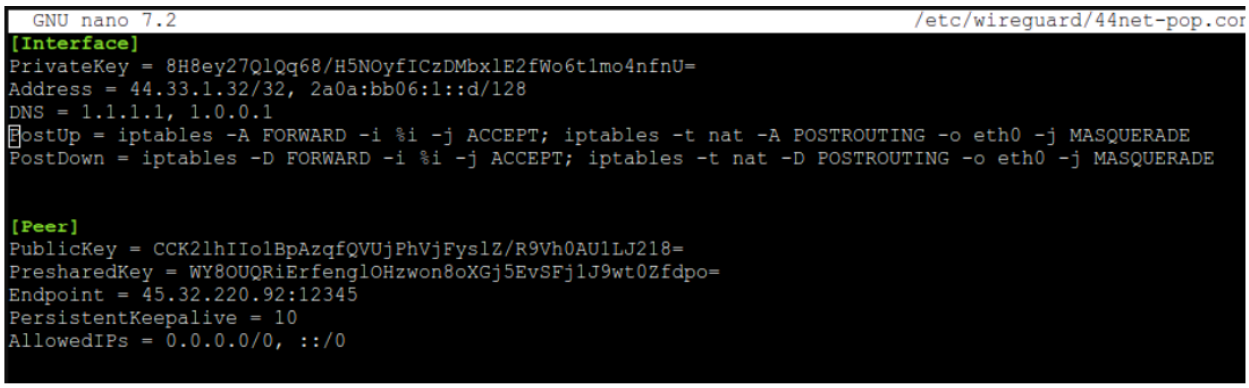
```
nano /etc/wireguard/44net-pop.conf
```

Add the following lines below the last line in the [Interface] section;

```
PostUp = iptables -A FORWARD -i eth0 -j ACCEPT; iptables -t nat
-A POSTROUTING -o 44net-pop -j MASQUERADE
```

```
PostDown = iptables -D FORWARD -i eth0 -j ACCEPT; iptables -t
nat -D POSTROUTING -o 44net-pop -j MASQUERADE
```

As shown in the example below;



Then restart the VPN

```
wg-quick down 44net-pop
wg-quick up 44net-pop
```

Now we need to tell linux that it should forward traffic. By default it is not a router. To enable forwarding type

```
sysctl -w net.ipv4.ip_forward=1
```

All your outgoing traffic will not only be sent to the Internet over the VPN but will be NAT'd or Masqueraded as if coming from the VPN container itself just like your home router does. Do not forget to add a route into your home router to send all 44.0.0.0/9 and 44.128.0.0/10 data to this container!

Route incoming connections

By default, all incoming connections will be stopped by the UFW firewall application. To allow incoming connections for any subnet you may have please read the appropriate documentation.