Setting up Wireguard for multi VPS system

Aadniz

August 16, 2023

Contents

1 Server Tutorial

Here we will explain how to setup wireguard on the VPS servers

1.1 Install wireguard

The first thing we need to do is to install wireguard

```
sudo apt update
sudo apt upgrade
sudo apt install wireguard
```

1.2 Setting up the keys

Generate the public and private keys by running the following commands

```
wg genkey | sudo tee /etc/wireguard/private.key
sudo chmod go= /etc/wireguard/private.key
sudo cat /etc/wireguard/private.key | wg pubkey | sudo tee /etc/wireguard/public.key
```

1.3 Configuring wg0.conf

When writing the configuration file, we need to figure out what ip address we want the server to have. We will follow the pattern of 10.0.x.1/16, where x is the VPS server count. We also need to write down the ethernet interface, being *eth0*, *ens3*, etc... Use this in the PostUp and PreDown.

For all clients connecting to the VPS servers, we will give them the ip 10.0.0.x/32, where we want the ending number to match the local ip number on the home network. So having 192.168.1.205, set the wireguard ip to 10.0.0.205.

The configuration file is located at */etc/wireguard/wg0.conf*

```
[Interface]
Address = 10.0.2.1/16
SaveConfig = true
PostUp = ufw route allow in on wg0 out on eth0
PostUp = iptables -t nat -I POSTROUTING -o eth0 -j MASQUERADE
PreDown = ufw route delete allow in on wg0 out on eth0
PreDown = iptables -t nat -D POSTROUTING -o eth0 -j MASQUERADE
ListenPort = 51820
PrivateKey = zJO4N....ZwiV=
# Client 1
[Peer]
PublicKey = lnDdwz....25XD=
AllowedIPs = 10.0.0.105/32
PersistentKeepalive = 25

# Client 2
[Peer]
PublicKey = KUvyJS....ovWW=
AllowedIPs = 10.0.0.205/32
PersistentKeepalive = 25
```

You can't set the private key to link to the private.key file. We have to copy paste the value of the file into the configuration.

1.4 Configuring the WireGuard Server's firewall

Open the firewall for SSH and port 51820 (Allow more ports if needed, like 80 and 443)

```
sudo ufw allow 51820/udp
sudo ufw allow OpenSSH
```

Then restart the firewall

```
sudo ufw disable
sudo ufw enable
sudo ufw status
```

1.5 Allow the server to forward the traffic

To allow the server to forward the traffic, we need to add the following in the */etc/sysctl.conf*:

```
net.ipv4.ip_forward=1
net.ipv6.conf.all.forwarding=1
```

And then run this command for it to take effect:

```
sudo sysctl -p
```

1.6 Start WireGuard

Now start wireguard using the following command:

```
systemctl start wg-quick@wg0.service
```

1.7 Finishing up

Confirm that everything is working correctly. When everything looks OK, finally run this command:

```
systemctl enable wg-quick@wg0.service
```

2 Client Tutorial

We are assuming that the client is running a Debian 12 instance. We will ensure this setup is possible to achieve over SSH throughout the steps.

2.1 Install wireguard

```
sudo apt update
sudo apt upgrade
sudo apt install wireguard
```

2.2 Setting up the keys

Again, as we did on the server, we need to run the following commands:

```
wg genkey | sudo tee /etc/wireguard/private.key
sudo chmod go= /etc/wireguard/private.key
sudo cat /etc/wireguard/private.key | wg pubkey | sudo tee /etc/wireguard/public.key
```

2.3 Configuring wg0.conf

The configuration file is located at */etc/wireguard/wg0.conf*

```
[Interface]
Address = 10.0.0.205/16
ListenPort = 51820
PrivateKey = hZ46R....tsNy=

[Peer]
PublicKey = 9vxKz...IWns=
AllowedIPs = 0.0.0.0/0
Endpoint = 18.52.125.212:51820
PersistentKeepalive = 25

[Peer]
PublicKey = 6PsUS...wbxBY=
AllowedIPs = 10.0.1.0/24
Endpoint = 93.122.227.117:51820
PersistentKeepalive = 25
```

```
[Peer]
PublicKey = 4vVA4...4u0as=
AllowedIPs = 10.0.2.0/24
Endpoint = 103.12.207.17:51820
PersistentKeepalive = 25
```

In the configuration we want to set the ip address within the subnet of the wireguard network, with the same ending numbers. So having 192.168.1.205, set the wireguard ip of 10.0.0.205.

You can't set the private key to link to the private.key file. We have to copy paste the value of the file into the configuration.

The peer that has *AllowedIPs* is the peer that gets the gateway priority.

2.4 Adding peer to the server(s)

Copy the content of */etc/wireguard/public.key* from the client, and paste in the following command on the server

```
sudo wg set wg0 peer PeUm3...29hg= allowed-ips 10.0.0.204 persistent-keepalive 25
```

2.5 Starting the service

```
sudo systemctl start wg-quick@wg0.service
sudo journalctl -fu wg-quick@wg0.service
```

2.6 Switching from one to another VPS

When switching from one wireguard VPS to another, it is important to never overlap the AllowedIPs.

Given the active VPS1 public key being *9vxKz...IWns=* and the other VPS2 public key being *6PsUS...wbxBY*, the order of switching should look like this:

```
# Cut off gateway to VPS1
sudo wg set wg0 peer 9vxKz...IWns= allowed-ips 10.0.0.0/24
# Start gateway to VPS2
sudo wg set wg0 peer 6PsUS...wbxBY allowed-ips 0.0.0.0/0
```

2.7 Automate the VPS switching

For the last couple of days, I have made a python script that should automatically switch to the working VPS server during interruptions. Privacy VPS services such as Njalla or 1984 Hosting are bound to be unreliable at times, therefore requires such configuration.

Github link: https://github.com/D3faIt/vps-wireguard-switcher

2.7.1 settings.json

Looking at the github link, we need to set our settings.json configuration inside the project folder. When tuning the settings, we can test it's working by running the following commands to run:

```
python3 -m venv venv
source venv/bin/activate
./venv/bin/pip3 install -r requirements.txt
sudo venv/bin/python3 main.py
```

2.7.2 Systemd service

Create a systemd serivce that runs this script under */etc/systemd/system/*. Paste this content

```
[Unit]
Description=VPS manager
After=network.target

[Service]
WorkingDirectory=/home/user/Documents/vps-manager
ExecStart=/home/user/Documents/vps-manager/venv/bin/python3 /home/user/Documents/vps-manager/ma
RestartSec=5
Restart=always
Environment=PYTHONUNBUFFERED=1

[Install]
WantedBy=multi-user.target
```

Now run the systemd service, and monitor it. Assuming the systemd serivce is called *vps-manager.service*, we will run this command:

```
sudo systemctl start vps-manager && sudo journalctl -fu vps-manager --all
```

If everything are working as expected, we can enable it:

```
sudo systemctl enable vps-manager
```