A somewhat gentle introduction to

Jason W. Eckert

The what and why

- Relatively new VPN technology (i.e. not as mature as OpenVPN)
  - Cross platform: Linux, BSD, Windows, Android, iOS, macOS
  - Implemented in kernel space on Linux
- High performance & strong crypto
  - ChaCha20 & Curve25519
  - Built-in protection for impersonation & crypto attacks
- Simple structure
  - No built-in authentication overhead (↓ bloat, ↑ flexibility)
  - Functions only when traffic is sent (↓ bandwidth)

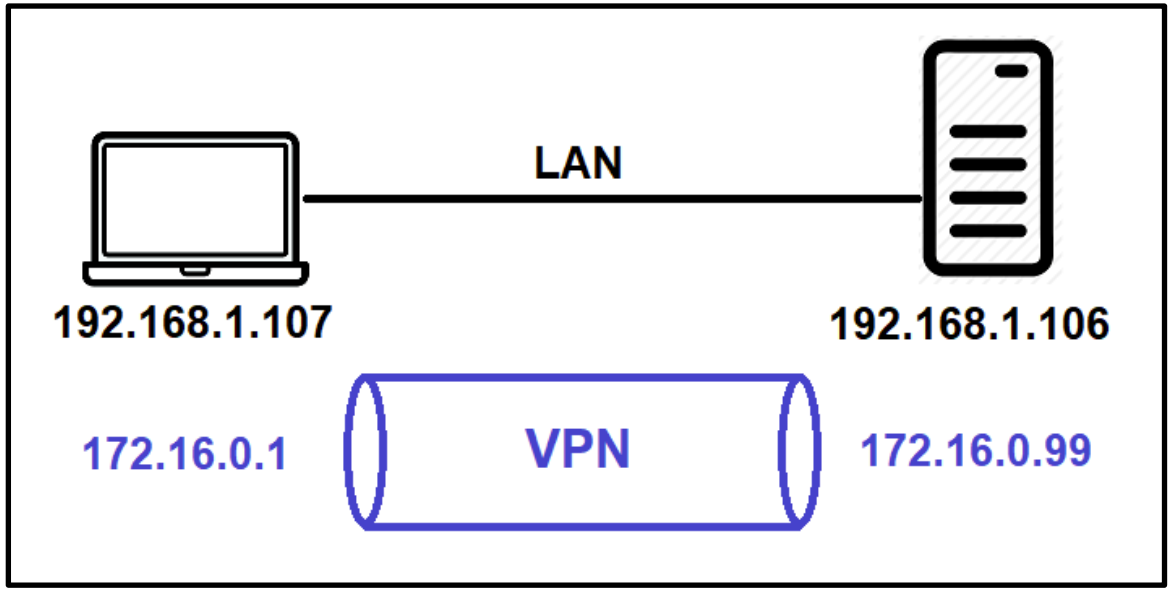
Other stuff

- You can get it from your Linux/BSD software repo or via an app
  - Details @ https://www.wireguard.com/install/
  - Avoid the macOS app right now and install via Homebrew or MacPorts
  - I'm using Fedora, so I'll use `sudo dnf install wireguard-tools`

- Approach:
  - To start we'll implement WireGuard from scratch between 2 Linux systems
  - Next, we'll make a persistent WireGuard configuration
  - Finally we'll apply this to other use cases

Basic VPN implementation (2 peers)

- No server or client (but easier to visualize)
- Virtual Private Network (VPN) = overlay network that auto-encrypts traffic
- We'll keep to IPv4 for simplicity (although IPv6 works too)



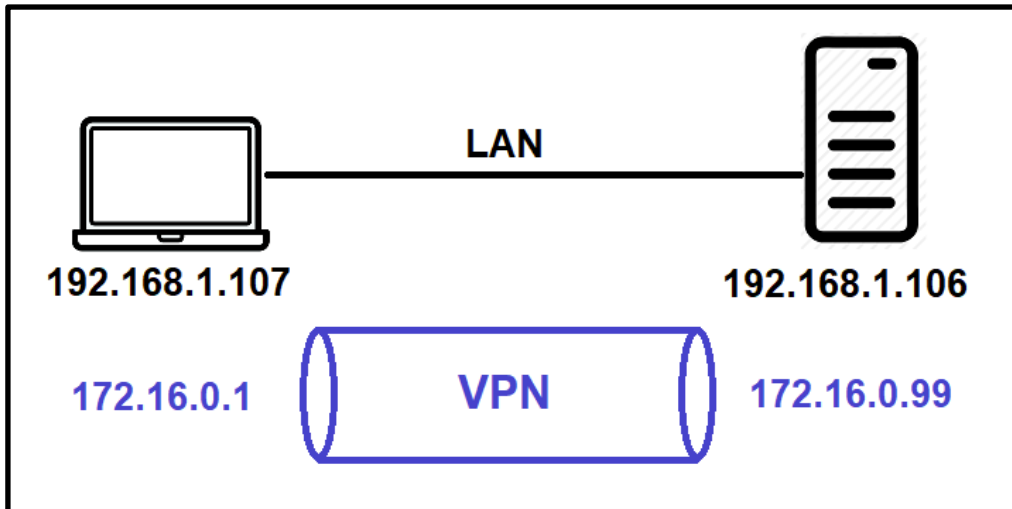
1. Generate asymmetric public/private key pair on each peer:

```
[root@server ~]# wg genkey | tee privatekey | wg pubkey > publickey
[root@server ~]# cat privatekey
+FqYdSx+rIg2gwwyd3hCfap/1Vz3z2UuRZCPKKwMaXw=
[root@server ~]# cat publickey
cjmyZf4c+6U3pD2QT+6Bxkjj9qzU8EePjc8dSeuXvWs=
```

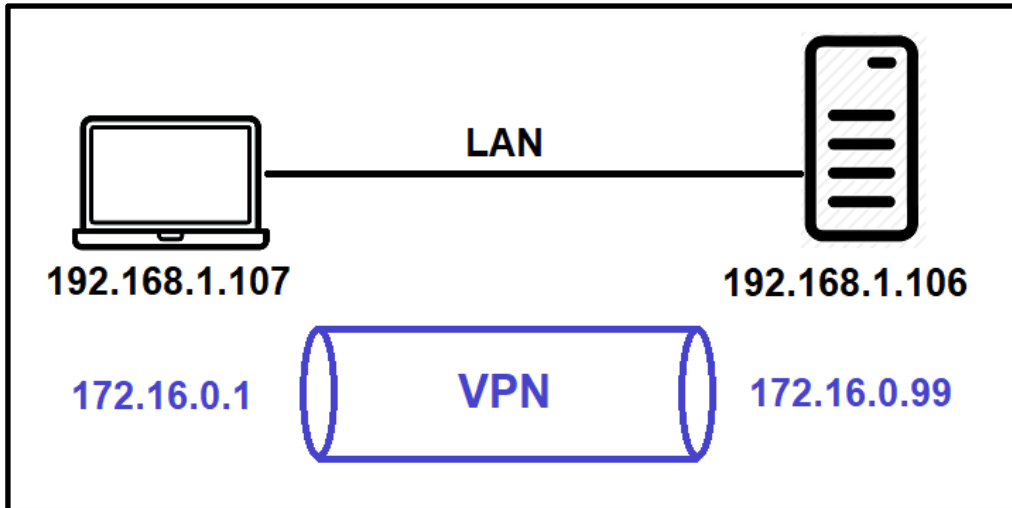
```
[root@client ~]# wg genkey | tee privatekey | wg pubkey > publickey
[root@client ~]# cat privatekey
0EQpGsSfGwVRdxcCywG2ymnLG7mjmv+rB02UodcH10k=
[root@client ~]# cat publickey
8pfWwwPK8R+Qe/fuN5FZ0P2ddngWd8s79sOQw5Q7SXE=
```

2. Construct a WireGuard interface on each peer (virtual IP & port of your choice). The first interface is called wg0, and should use the private key you generated (public key is auto generated from private key):

```
[root@server ~]# ip link add wg0 type wireguard
[root@server ~]# ip addr add 172.16.0.99/24 dev wg0
[root@server ~]# wg set wg0 private-key ./privatekey listen-port 55234
[root@server ~]# ip link set wg0 up
[root@server ~]# wg
interface: wg0
  public key: cjmyZf4c+6U3pD2QT+6Bxkjj9qzU8EePjc8dSeuXvWs=
  private key: (hidden)
  listening port: 55234
```



```
[root@client ~]# ip link add wg0 type wireguard
[root@client ~]# ip addr add 172.16.0.1/24 dev wg0
[root@client ~]# wg set wg0 private-key ./privatekey listen-port 55123
[root@client ~]# ip link set wg0 up
[root@client ~]# wg
interface: wg0
  public key: 8pfWwwPK8R+Qe/fuN5FZ0P2ddngWd8s79sOQw5Q7SXE=
  private key: (hidden)
  listening port: 55123
```

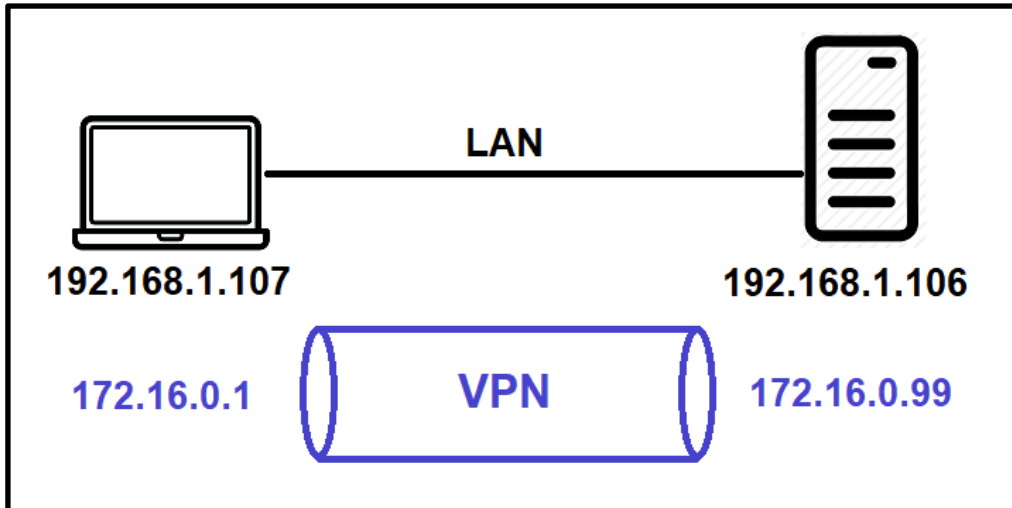


3. Add the client's public key, IP & port to the server, as well as add the server's public key, IP & port to the client. You must also specify the IP addresses or networks you will allow over WireGuard:

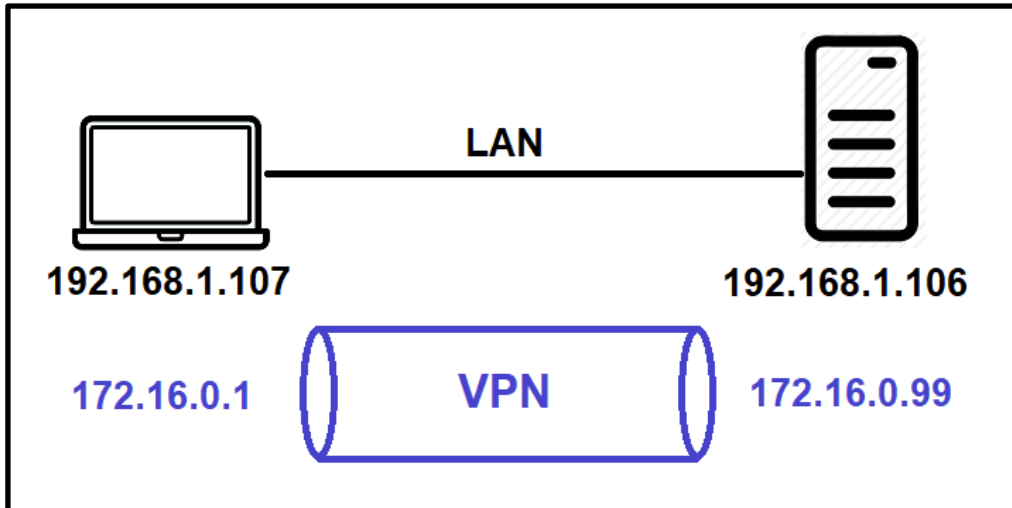
```
[root@server ~]# wg set wg0 peer 8pfWwwPK8R+Qe/fuN5FZ0P2ddngWd8s79sOQw5Q7SXE= \
allowed-ips 172.16.0.0/16 endpoint 192.168.1.107:55123

[root@client ~]# wg set wg0 peer cjmyZf4c+6U3pD2QT+6Bxkjj9qzU8EePjc8dSeuXvWs= \
allowed-ips 172.16.0.0/16 endpoint 192.168.1.106:55234
```

When sending data to a peer, 172.16.0.0/16 is treated like a target route.

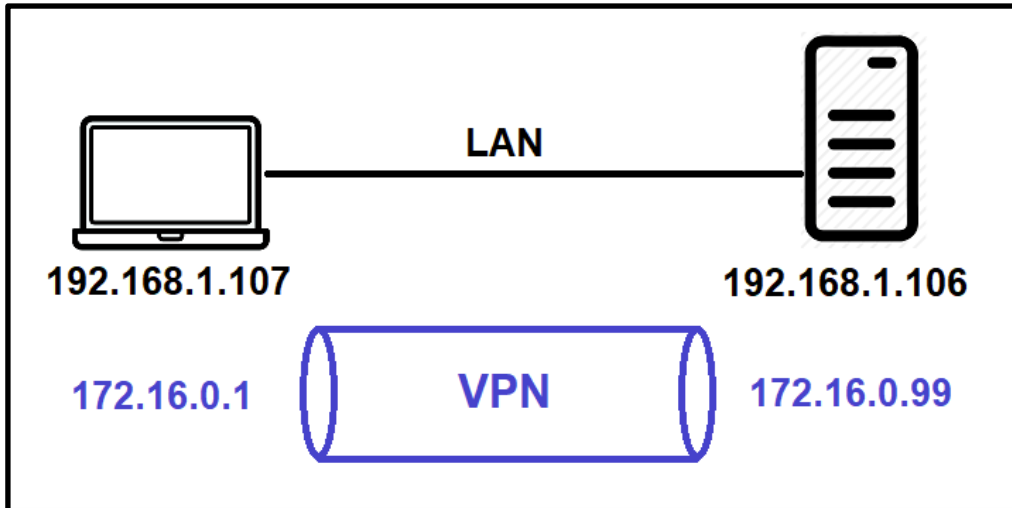
When receiving data from a peer, 172.16.0.0/16 is treated like an access control list.



```
[root@client ~]# ping 172.16.0.99
<output omitted>
[root@client ~]# wg
interface: wg0
   public key: 8pfWwwPK8R+Qe/fuN5FZ0P2ddngWd8s79sOQw5Q7SXE=
   private key: (hidden)
   listening port: 55123
peer: cjmyZf4c+6U3pD2QT+6Bxkjj9qzU8EePjc8dSeuXvWs=
   endpoint: 192.168.1.106:55234
   allowed ips: 172.16.0.0/16
   latest handshake: 18 seconds ago
   transfer: 8.21 KiB received, 15.10 KiB sent
```



```
[root@server ~]# wg
interface: wg0
   public key: cjmyZf4c+6U3pD2QT+6Bxkjj9qzU8EePjc8dSeuXvWs=
   private key: (hidden)
   listening port: 55192
peer: 8pfWwwPK8R+Qe/fuN5FZ0P2ddngWd8s79sOQw5Q7SXE=
   endpoint: 192.168.1.107:51423
   allowed ips: 172.16.0.0/16
   latest handshake: 27 seconds ago
   transfer: 7.01 KiB received, 11.39 KiB sent
```



Making WireGuard configuration persistent

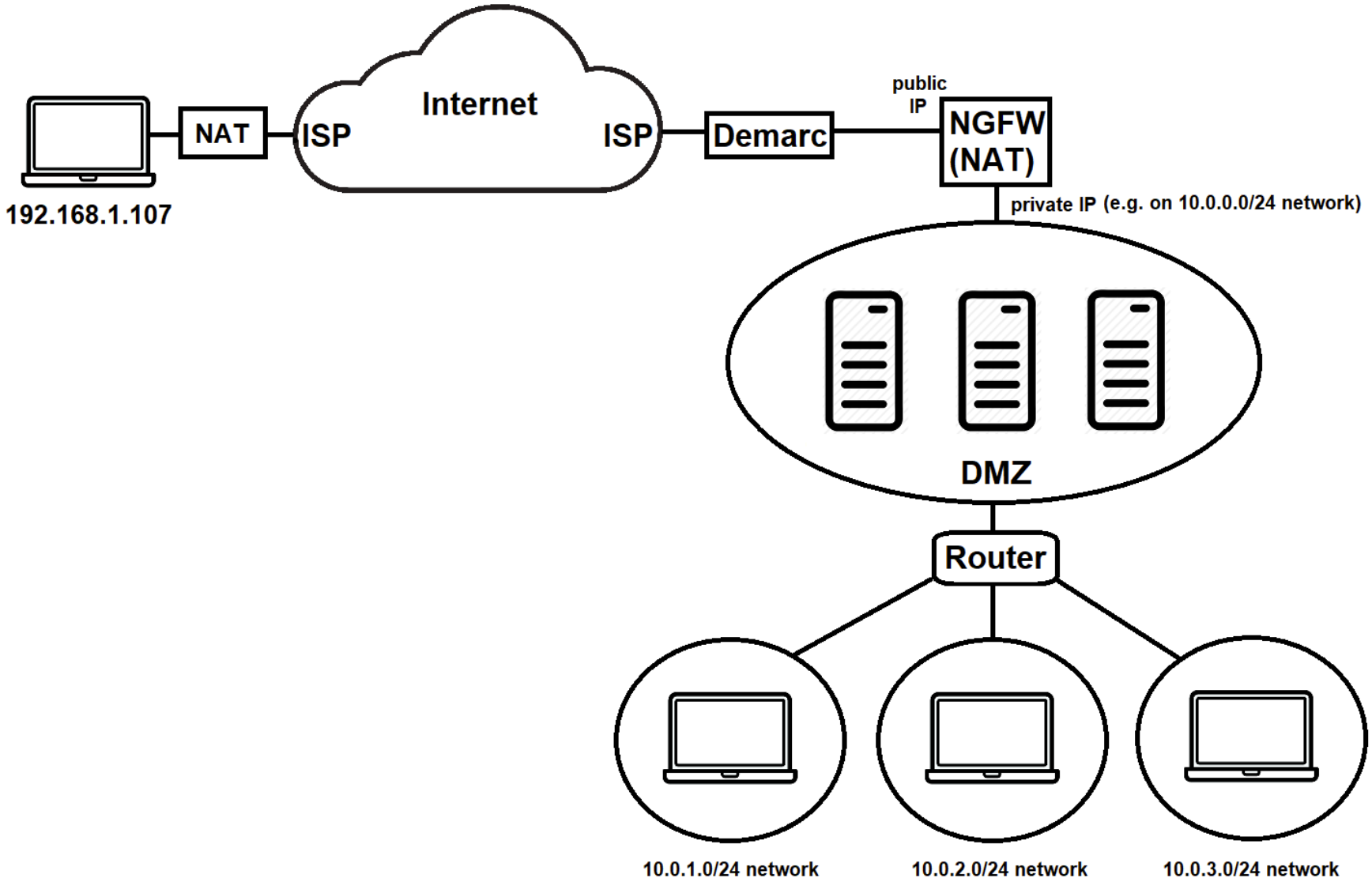
- Method 1: Write a shell script:

```
[root@client ~]# cat wireguard-client.sh
#!/bin/bash
ip link add wg0 type wireguard
ip addr add 172.16.0.1/16 dev wg0
wg set wg0 private-key ./privatekey listen-port 55123
ip link set wg0 up
wg set wg0 peer cjmyZf4c+6U3pD2QT+6Bxkjj9qzU8EePjc8dSeuXvWs= allowed-ips
172.16.0.0/16 endpoint 192.168.1.106:55234
```

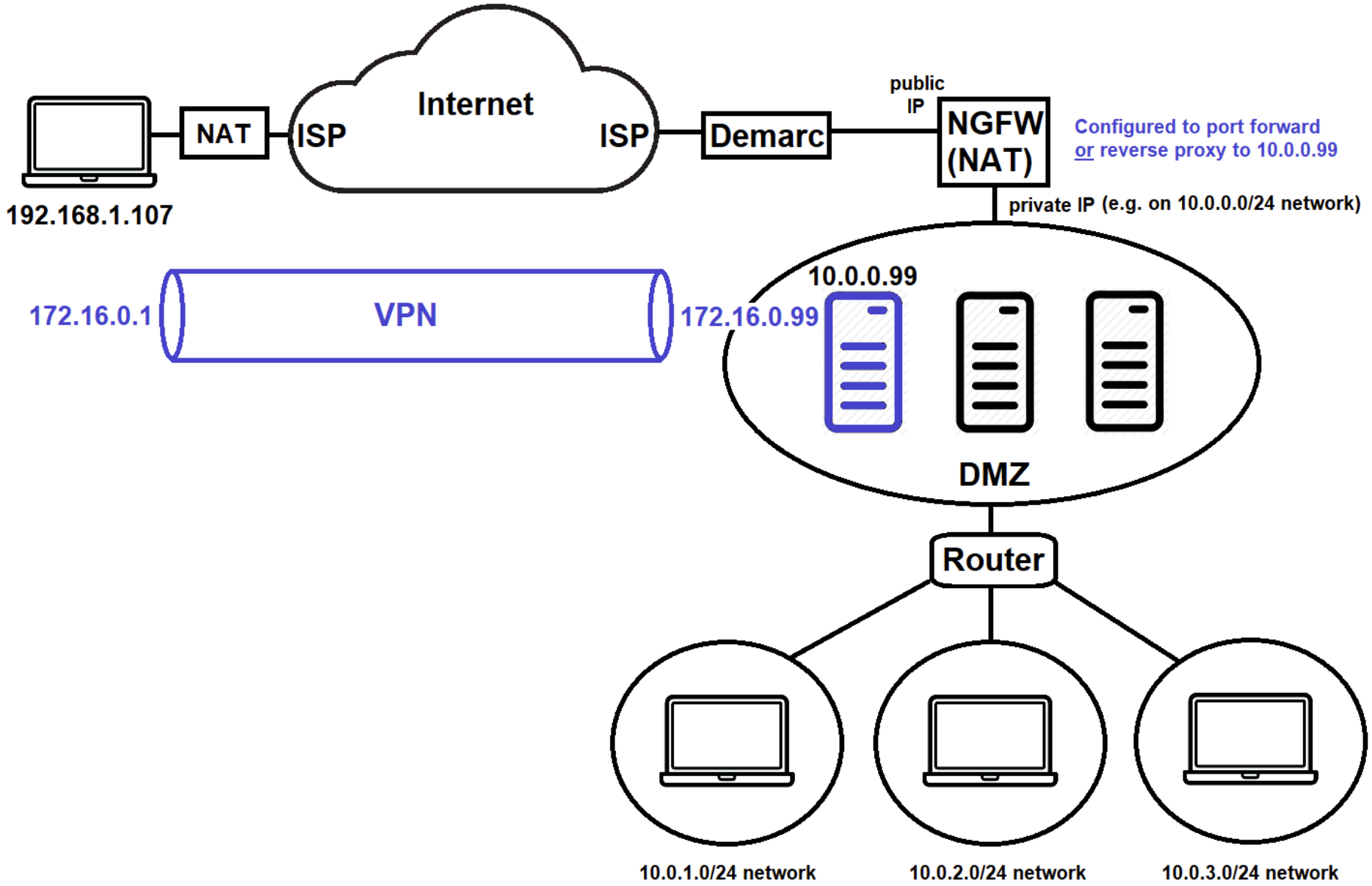
- Method 2: Use /etc/wireguard/wg0.conf:

```
[root@client ~]# wg showconf wg0 > /etc/wireguard/wg0.conf
[root@client ~]# chmod 600 /etc/wireguard/wg0.conf
[root@client ~]# cat /etc/wireguard/wg0.conf
  [Interface]
  Address = 172.16.0.1/16
  ListenPort = 55123
  PrivateKey = 0EQpGsSfGwVRdxcCywG2ymnLG7mjmv+rB02UodcH10k=

  [Peer]
  PublicKey = cjmyZf4c+6U3pD2QT+6Bxkjj9qzU8EePjc8dSeuXvWs=
  AllowedIPs = 172.16.0.0/16
  Endpoint = 192.168.1.106:55234

[root@client ~]# wg-quick up wg0
[root@client ~]# systemctl enable wg-quick@wg0.service
```

Using WireGuard to access a remote network



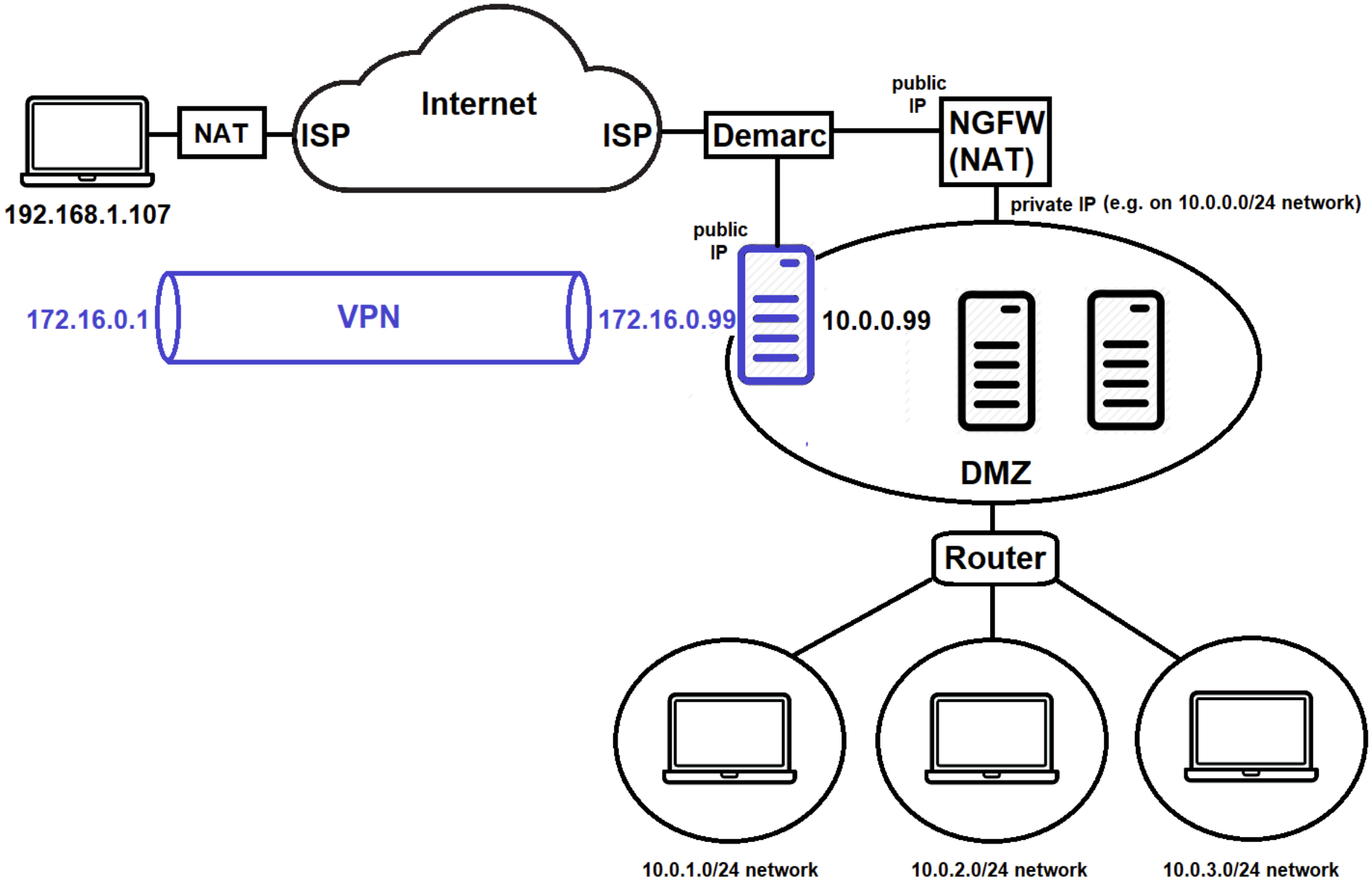
- Method 1: WireGuard server behind a NGFW



```
[root@client ~]# cat /etc/wireguard/wg0.conf
  [Interface]
  PrivateKey = 0EQpGsSfGwVRdxcCywG2ymnLG7mjmv+rB02UodcH10k=
  Address = 172.16.0.1/16
  DNS = 1.1.1.1   #By specifying DNS here, name resolution is faster
  #No ListenPort needed!

  [Peer]
  PublicKey = cjmyZf4c+6U3pD2QT+6Bxkjj9qzU8EePjc8dSeuXvWs=
  AllowedIPs = 0.0.0.0/0   #This forwards all traffic to the WireGuard server
  Endpoint = publicIP:55234
```

```
[root@server ~]# sysctl -w net.ipv4.ip_forward=1 >> /etc/sysctl.conf

[root@server ~]# cat /etc/wireguard/wg0.conf
  [Interface]
  PrivateKey = +FqYdSx+rIg2gwwyd3hCfap/1Vz3z2UuRZCPKKwMaXw=
  Address = 172.16.0.99/16
  ListenPort = 55234
  PostUp = iptables -A FORWARD -i wg0 -j ACCEPT; iptables -t nat -A POSTROUTING \
           -o eth0 -j MASQUERADE
  PostDown = iptables -D FORWARD -i wg0 -j ACCEPT; iptables -t nat -D POSTROUTING \
           -o eth0 -j MASQUERADE

  [Peer]
  PublicKey = 8pfWwwPK8R+Qe/fuN5FZ0P2ddngWd8s79sOQw5Q7SXE=
  AllowedIPs = 172.16.0.0/16
  #No Endpoint needed!
```

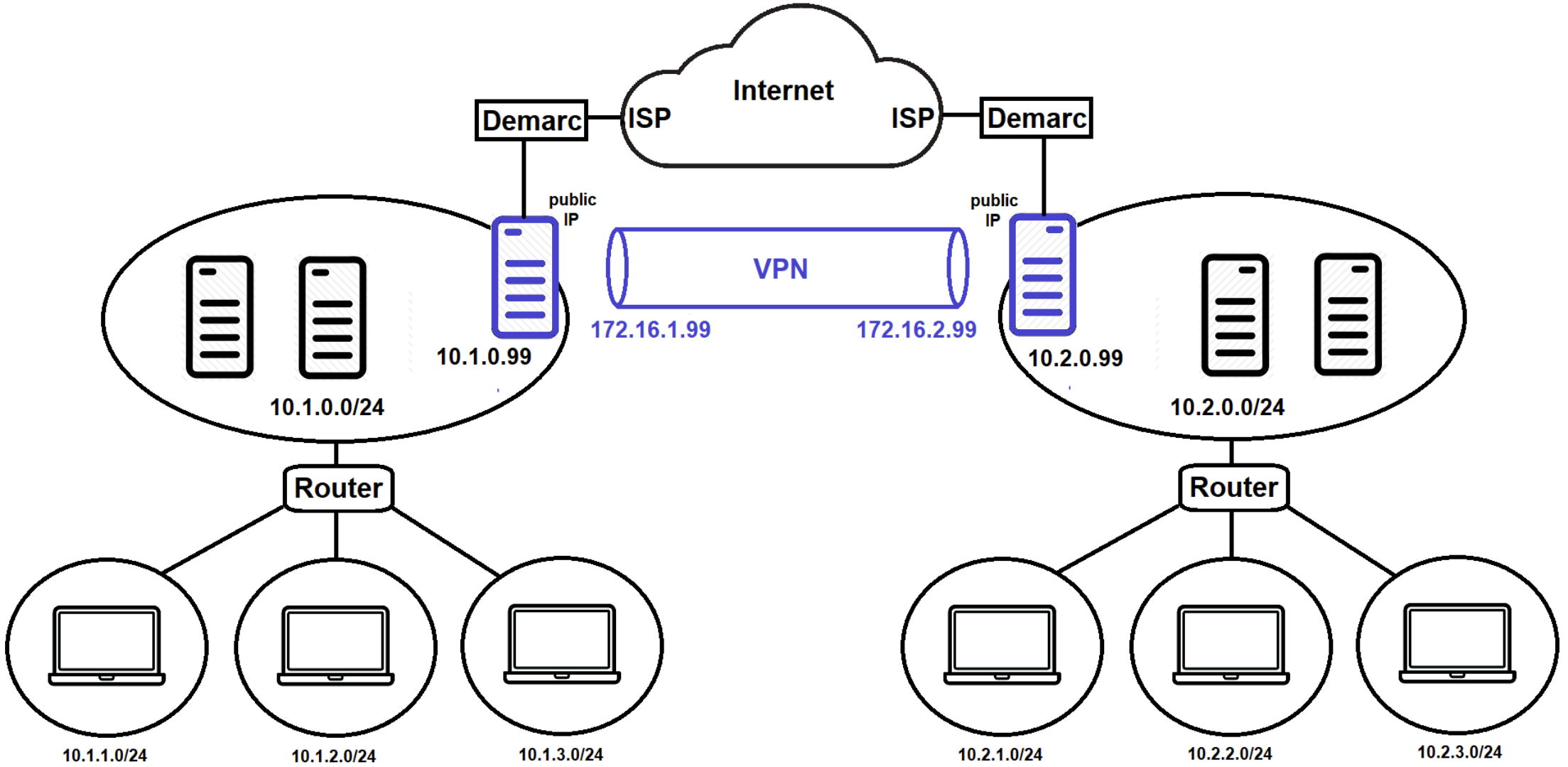
- Method 2: WireGuard server connected to the demarc



- Nearly all the configuration is identical, except:
  - The client specifies the endpoint of the actual WireGuard server connected to the demarc (no need for port forwarding or reverse proxy on NGFW)
  - You must specify the network interface that is connected to the DMZ when configuring IP masquerading on the server. If this network is eth1 (since eth0 is connected to the demarc), then:

```
[root@server ~]# grep Post /etc/wireguard/wg0.conf
PostUp = iptables -A FORWARD -i wg0 -j ACCEPT; iptables -t nat -A POSTROUTING \
         -o eth1 -j MASQUERADE
PostDown = iptables -D FORWARD -i wg0 -j ACCEPT; iptables -t nat -D POSTROUTING \
         -o eth1 -j MASQUERADE
```

Using WireGuard to secure traffic between 2+ networks



```
[root@server1 ~]# cat /etc/wireguard/wg0.conf
  [Interface]
  PrivateKey = 0EQpGsSfGwVRdxcCywG2ymnLG7mjmv+rB02UodcH10k=
  Address = 172.16.1.99/16
  ListenPort = 55123
  PostUp = iptables -A FORWARD -i wg0 -j ACCEPT; iptables -t nat -A POSTROUTING \
           -o eth1 -j MASQUERADE
  PostDown = iptables -D FORWARD -i wg0 -j ACCEPT; iptables -t nat -D POSTROUTING \
           -o eth1 -j MASQUERADE

  [Peer]
  PublicKey = cjmyZf4c+6U3pD2QT+6Bxkjj9qzU8EePjc8dSeuXvWs=
  AllowedIPs = 10.2.0.0/16   #Forwards traffic for 10.2.x.0/24 networks to server2
  Endpoint = server2publicIP:55234
```

```
[root@server2 ~]# cat /etc/wireguard/wg0.conf
  [Interface]
  PrivateKey = +FqYdSx+rIg2gwwyd3hCfap/1Vz3z2UuRZCPKKwMaXw=
  Address = 172.16.2.99/16
  ListenPort = 55234
  PostUp = iptables -A FORWARD -i wg0 -j ACCEPT; iptables -t nat -A POSTROUTING \
           -o eth1 -j MASQUERADE
  PostDown = iptables -D FORWARD -i wg0 -j ACCEPT; iptables -t nat -D POSTROUTING \
           -o eth1 -j MASQUERADE

  [Peer]
  PublicKey = 8pfWwwPK8R+Qe/fuN5FZ0P2ddngWd8s79sOQw5Q7SXE=
  AllowedIPs = 10.1.0.0/16   #Forwards traffic for 10.1.x.0/24 networks to server1
  Endpoint = server1publicIP:55123
```

One more thing…

- Some NGFWs and NAT routers don't like that WireGuard only sends traffic when it needs to. To make WireGuard a bit more "chatty", add the following to your configuration:

```
[root@client ~]# grep -i keep /etc/wireguard/wg0.conf
KEEPALIVE_PERIOD = 25
```

- Easy masquerading & client isolation:

```
[root@server ~]# egrep -i "(internet|client)" /etc/wireguard/wg0.conf
ALLOW_INTERNET_ACCESS = yes
CLIENT_TO_CLIENT = no
```

- You can also get this presentation in blog format @ jasoneckert.net

Using the apps