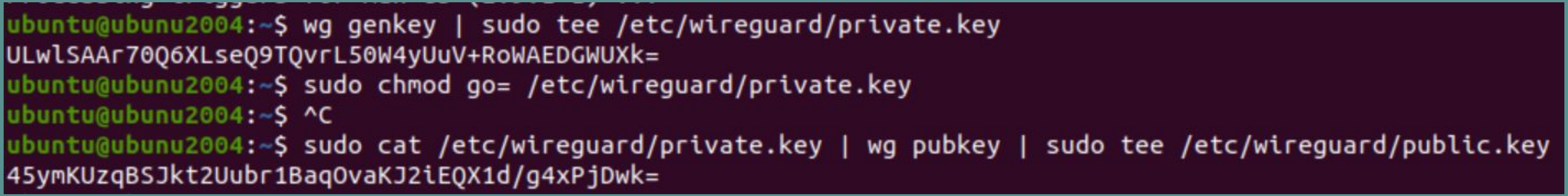
ITIS 3246 Final Project: VPN Configuration

Aadya Maurya and Esha Patel

We started off by updating and installing wireguard. Once that was complete, we generated the public and private key pairs for the server using the following commands.



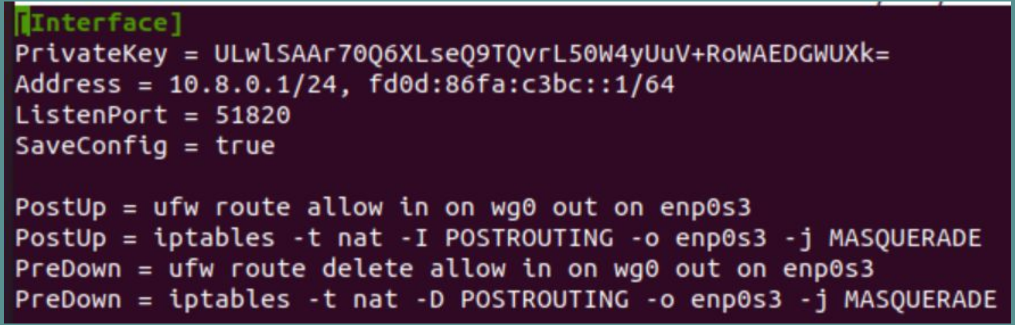
```
ubuntu@ubunu2004:~$ wg genkey | sudo tee /etc/wireguard/private.key
ULwlSAAr70Q6XLseQ9TQvrL50W4yUuV+RoWAEDGWUXk=
ubuntu@ubunu2004:~$ sudo chmod go= /etc/wireguard/private.key
ubuntu@ubunu2004:~$ ^C
ubuntu@ubunu2004:~$ sudo cat /etc/wireguard/private.key | wg pubkey | sudo tee /etc/wireguard/public.key
45ymKUzqBSJkt2Uubr1BaqOvaKJ2iEQX1d/g4xPjDwk=
```

Next, we changed the permissions, allowing only the root user to access the private key, using this command:

```
sudo chmod go= /etc/wireguard/private.key
```

After that, we chose the IPv4 range that we wanted to use, which is 10.8.0.0/24

Then, we created the wireguard server configuration file, named wg0.conf. The contents of the file are displayed below:

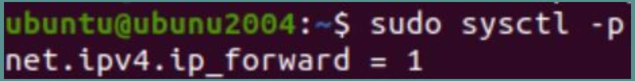


```
[Interface]
PrivateKey = ULwlSAAr70Q6XLseQ9TQvrL50W4yUuV+RoWAEDGWUXk=
Address = 10.8.0.1/24, fd0d:86fa:c3bc::1/64
ListenPort = 51820
SaveConfig = true

PostUp = ufw route allow in on wg0 out on enp0s3
PostUp = iptables -t nat -I POSTROUTING -o enp0s3 -j MASQUERADE
PreDown = ufw route delete allow in on wg0 out on enp0s3
PreDown = iptables -t nat -D POSTROUTING -o enp0s3 -j MASQUERADE
```

We made sure to add in our private key to the file.

Our next step was to configure the IP forwarding. We opened sysctl.conf and added the following line:

```
net.ipv4.ip_forward=1
```



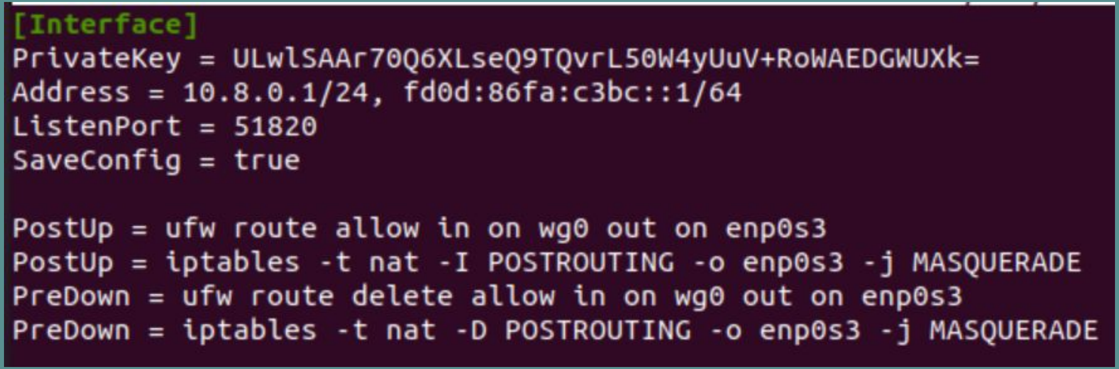
```
ubuntu@ubunu2004:~$ sudo sysctl -p
net.ipv4.ip_forward = 1
```

This allows the wireguard server to forward incoming traffic from the virtual VPN device to others on the server and then the public internet. Using this configuration will allow the clients public IP address to be properly hidden.

Now we configured wiregards server to add firewall rules, so that traffic to and from the server and clients is routed correctly. We ran the command below to find out the name of the public network interface of the wireguard server:

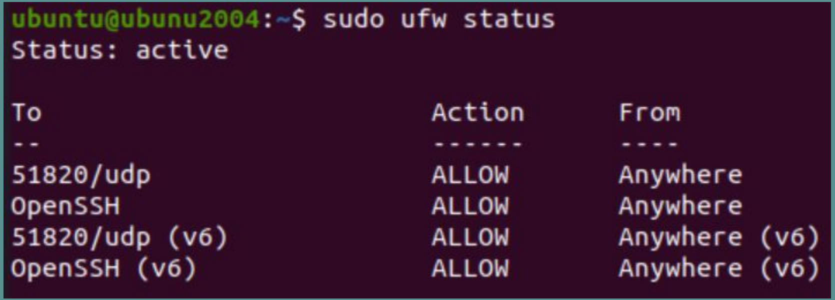
ip route list default

Once we found that out, we opened wg0.conf and added the following lines to enable masquerading.



```
[Interface]
PrivateKey = ULwlSAAr70Q6XLseQ9TQvrL50W4yUuV+RoWAEDGWUXk=
Address = 10.8.0.1/24, fd0d:86fa:c3bc::1/64
ListenPort = 51820
SaveConfig = true

PostUp = ufw route allow in on wg0 out on enp0s3
PostUp = iptables -t nat -I POSTROUTING -o enp0s3 -j MASQUERADE
PreDown = ufw route delete allow in on wg0 out on enp0s3
PreDown = iptables -t nat -D POSTROUTING -o enp0s3 -j MASQUERADE
```

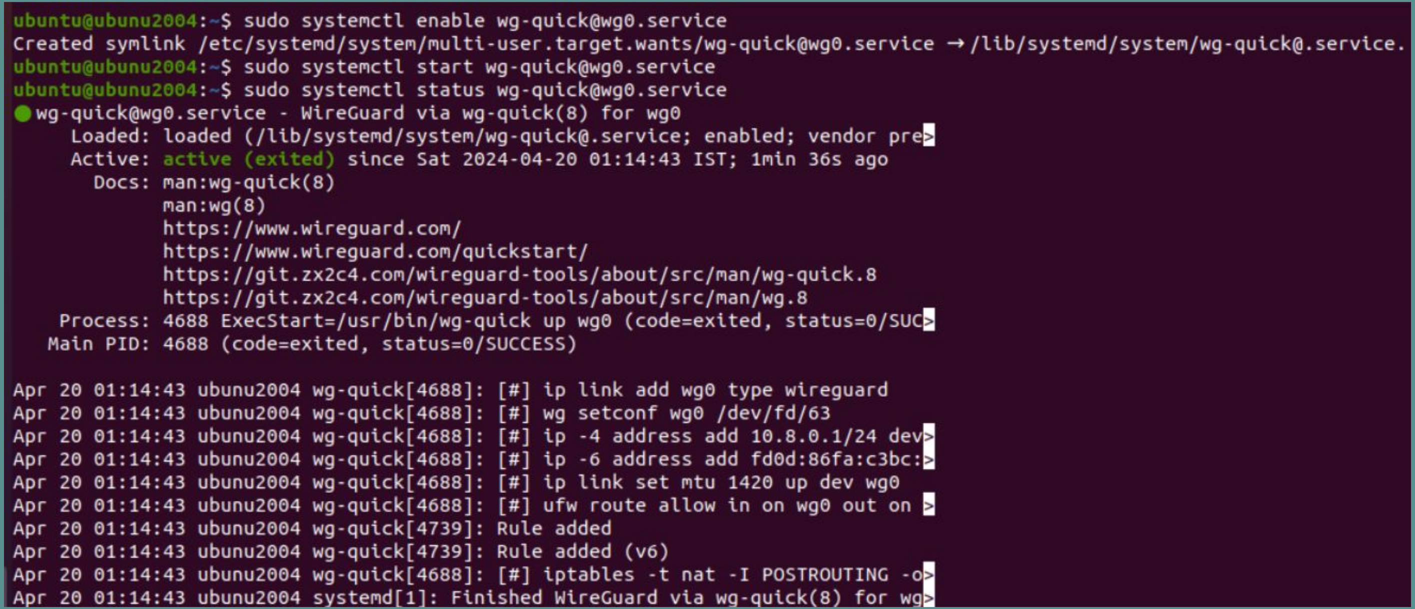
Then we ran the sudo ufw status command to ensure that the rules were in place.



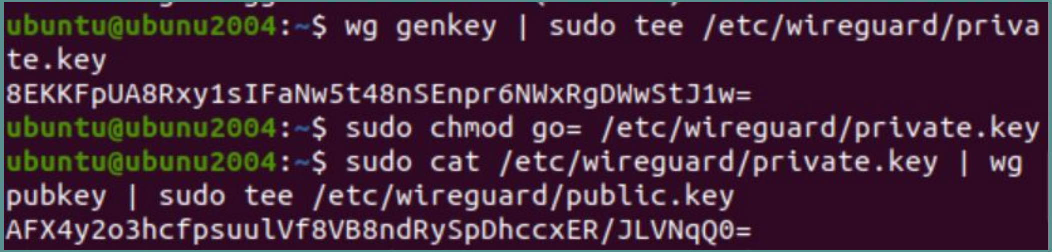
```
ubuntu@ubunu2004:~$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
51820/udp                  ALLOW       Anywhere
OpenSSH                    ALLOW       Anywhere
51820/udp (v6)             ALLOW       Anywhere (v6)
OpenSSH (v6)               ALLOW       Anywhere (v6)
```

We started the wireguard server to make sure it was running properly. Before that, we enabled wg-quick service, and then we started and checked the status.

The output displays the ip commands that are used to create the wg0 device and assign it to the ip addresses that we added to the configuration file.

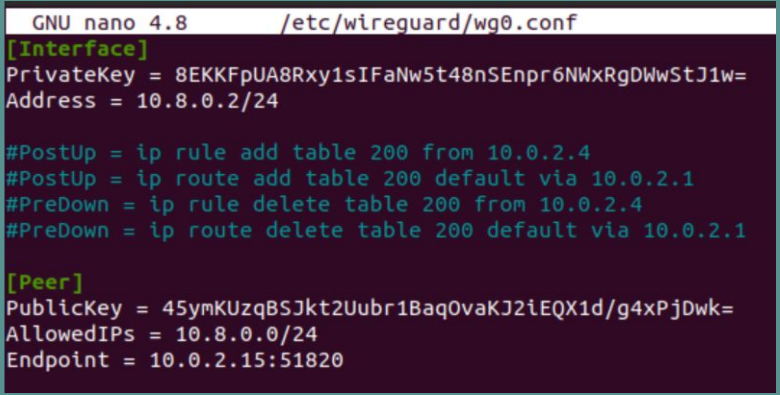

```
ubuntu@ubunu2004:~$ sudo systemctl enable wg-quick@wg0.service
Created symlink /etc/systemd/system/multi-user.target.wants/wg-quick@wg0.service → /lib/systemd/system/wg-quick@.service.
ubuntu@ubunu2004:~$ sudo systemctl start wg-quick@wg0.service
ubuntu@ubunu2004:~$ sudo systemctl status wg-quick@wg0.service
● wg-quick@wg0.service - WireGuard via wg-quick(8) for wg0
     Loaded: loaded (/lib/systemd/system/wg-quick@.service; enabled; vendor pre>
     Active: active (exited) since Sat 2024-04-20 01:14:43 IST; 1min 36s ago
       Docs: man:wg-quick(8)
             man:wg(8)
             https://www.wireguard.com/
             https://www.wireguard.com/quickstart/
             https://git.zx2c4.com/wireguard-tools/about/src/man/wg-quick.8
             https://git.zx2c4.com/wireguard-tools/about/src/man/wg.8
    Process: 4688 ExecStart=/usr/bin/wg-quick up wg0 (code=exited, status=0/SUC>
   Main PID: 4688 (code=exited, status=0/SUCCESS)

Apr 20 01:14:43 ubunu2004 wg-quick[4688]: [#] ip link add wg0 type wireguard
Apr 20 01:14:43 ubunu2004 wg-quick[4688]: [#] wg setconf wg0 /dev/fd/63
Apr 20 01:14:43 ubunu2004 wg-quick[4688]: [#] ip -4 address add 10.8.0.1/24 dev>
Apr 20 01:14:43 ubunu2004 wg-quick[4688]: [#] ip -6 address add fd0d:86fa:c3bc:>
Apr 20 01:14:43 ubunu2004 wg-quick[4688]: [#] ip link set mtu 1420 up dev wg0
Apr 20 01:14:43 ubunu2004 wg-quick[4688]: [#] ufw route allow in on wg0 out on >
Apr 20 01:14:43 ubunu2004 wg-quick[4739]: Rule added
Apr 20 01:14:43 ubunu2004 wg-quick[4739]: Rule added (v6)
Apr 20 01:14:43 ubunu2004 wg-quick[4688]: [#] iptables -t nat -I POSTROUTING -o>
Apr 20 01:14:43 ubunu2004 systemd[1]: Finished WireGuard via wg-quick(8) for wg>
```

On our peer 1 VM, we updated and installed wiregard. Then, we generated the public and private key pairs for the peer VM as well.



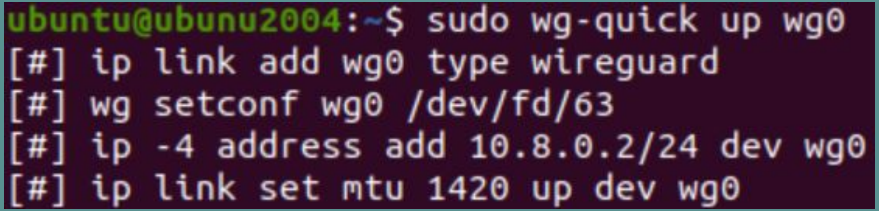
```
ubuntu@ubunu2004:~$ wg genkey | sudo tee /etc/wireguard/priva
te.key
8EKKFpUA8Rxy1sIFaNw5t48nSEnpr6NWxRgDWwStJ1w=
ubuntu@ubunu2004:~$ sudo chmod go= /etc/wireguard/private.key
ubuntu@ubunu2004:~$ sudo cat /etc/wireguard/private.key | wg
pubkey | sudo tee /etc/wireguard/public.key
AFX4y2o3hcfpsuulVf8VB8ndRySpDhccxER/JLVNqQ0=
```

We created the configuration file wg0.conf for peer 1 and added the private key and allowed IPs inside of it.



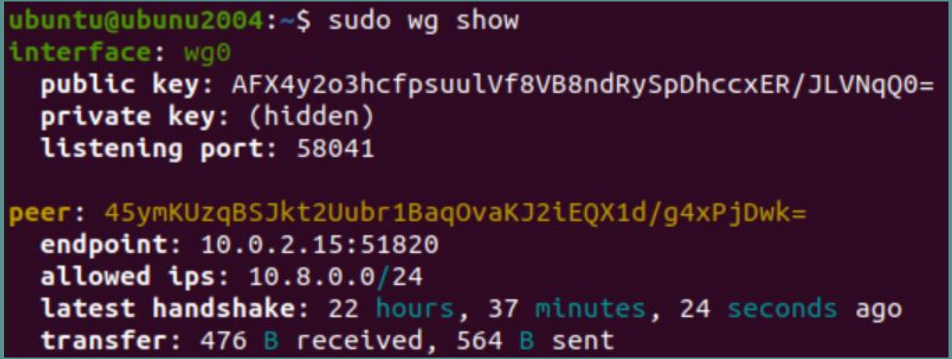
```
  GNU nano 4.8        /etc/wireguard/wg0.conf
[Interface]
PrivateKey = 8EKKFpUA8Rxy1sIFaNw5t48nSEnpr6NWxRgDWwStJ1w=
Address = 10.8.0.2/24

#PostUp = ip rule add table 200 from 10.0.2.4
#PostUp = ip route add table 200 default via 10.0.2.1
#PreDown = ip rule delete table 200 from 10.0.2.4
#PreDown = ip route delete table 200 default via 10.0.2.1

[Peer]
PublicKey = 45ymKUzqBSJkt2Uubr1BaqOvaKJ2iEQX1d/g4xPjDwk=
AllowedIPs = 10.8.0.0/24
Endpoint = 10.0.2.15:51820
```

We added the peer's public key to the wireguard server so that we can connect and route traffic over the VPN. We used the sudo wg set wg0 peer command to set the peer 1 up. Then we connected peer to the VPN tunnel and installed resolvconf. To start the tunnel we ran sudo wg-quick up wg0



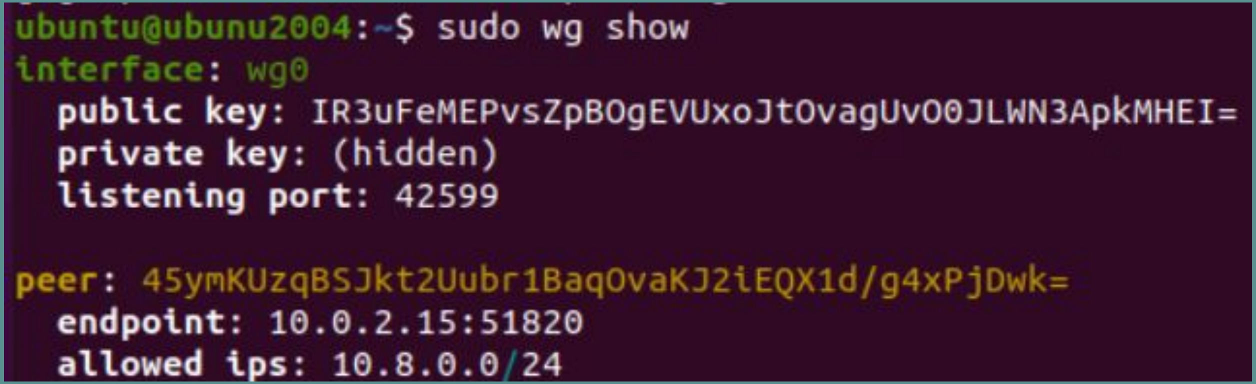
```
ubuntu@ubunu2004:~$ sudo wg-quick up wg0
[#] ip link add wg0 type wireguard
[#] wg setconf wg0 /dev/fd/63
[#] ip -4 address add 10.8.0.2/24 dev wg0
[#] ip link set mtu 1420 up dev wg0
```

Using the wg show command, we checked the status of the tunnel on the peer 1 VM.



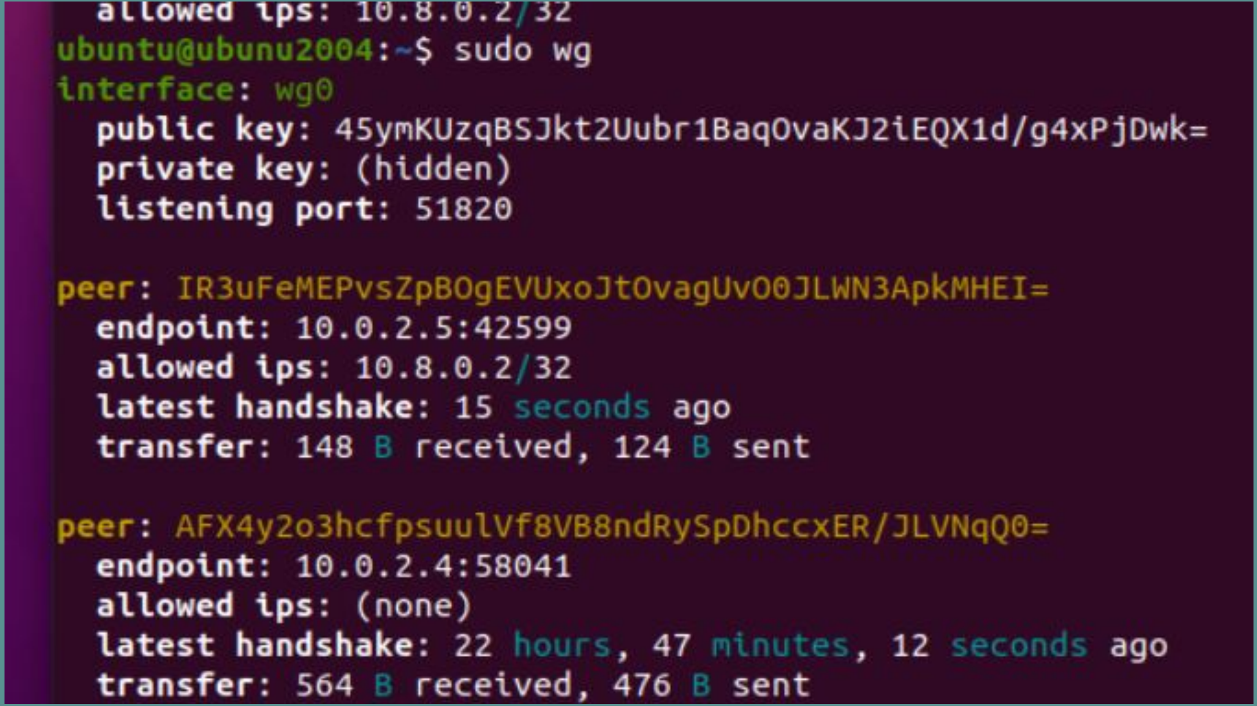
```
ubuntu@ubunu2004:~$ sudo wg show
interface: wg0
  public key: AFX4y2o3hcfpsuulVf8VB8ndRySpDhccxER/JLVNqQ0=
  private key: (hidden)
  listening port: 58041

peer: 45ymKUzqBSJkt2Uubr1BaqOvaKJ2iEQX1d/g4xPjDwk=
  endpoint: 10.0.2.15:51820
  allowed ips: 10.8.0.0/24
  latest handshake: 22 hours, 37 minutes, 24 seconds ago
  transfer: 476 B received, 564 B sent
```

We repeated the process in peer 2’s VM. The result of the wg show command is displayed below:



```
ubuntu@ubunu2004:~$ sudo wg show
interface: wg0
  public key: IR3uFeMEPvsZpBOgEVUxoJtOvagUvO0JLWN3ApkMHEI=
  private key: (hidden)
  listening port: 42599

peer: 45ymKUzqBSJkt2Uubr1BaqOvaKJ2iEQX1d/g4xPjDwk=
  endpoint: 10.0.2.15:51820
  allowed ips: 10.8.0.0/24
```

Finally, here is the result of the sudo wg command when run on the wireguard server, showing both the peer's public keys:



```
  allowed ips: 10.8.0.2/32
ubuntu@ubunu2004:~$ sudo wg
interface: wg0
  public key: 45ymKUzqBSJkt2Uubr1BaqOvaKJ2iEQX1d/g4xPjDwk=
  private key: (hidden)
  listening port: 51820

peer: IR3uFeMEPvsZpBOgEVUxoJtOvagUvO0JLWN3ApkMHEI=
  endpoint: 10.0.2.5:42599
  allowed ips: 10.8.0.2/32
  latest handshake: 15 seconds ago
  transfer: 148 B received, 124 B sent

peer: AFX4y2o3hcfpsuulVf8VB8ndRySpDhccxER/JLVNqQ0=
  endpoint: 10.0.2.4:58041
  allowed ips: (none)
  latest handshake: 22 hours, 47 minutes, 12 seconds ago
  transfer: 564 B received, 476 B sent
```