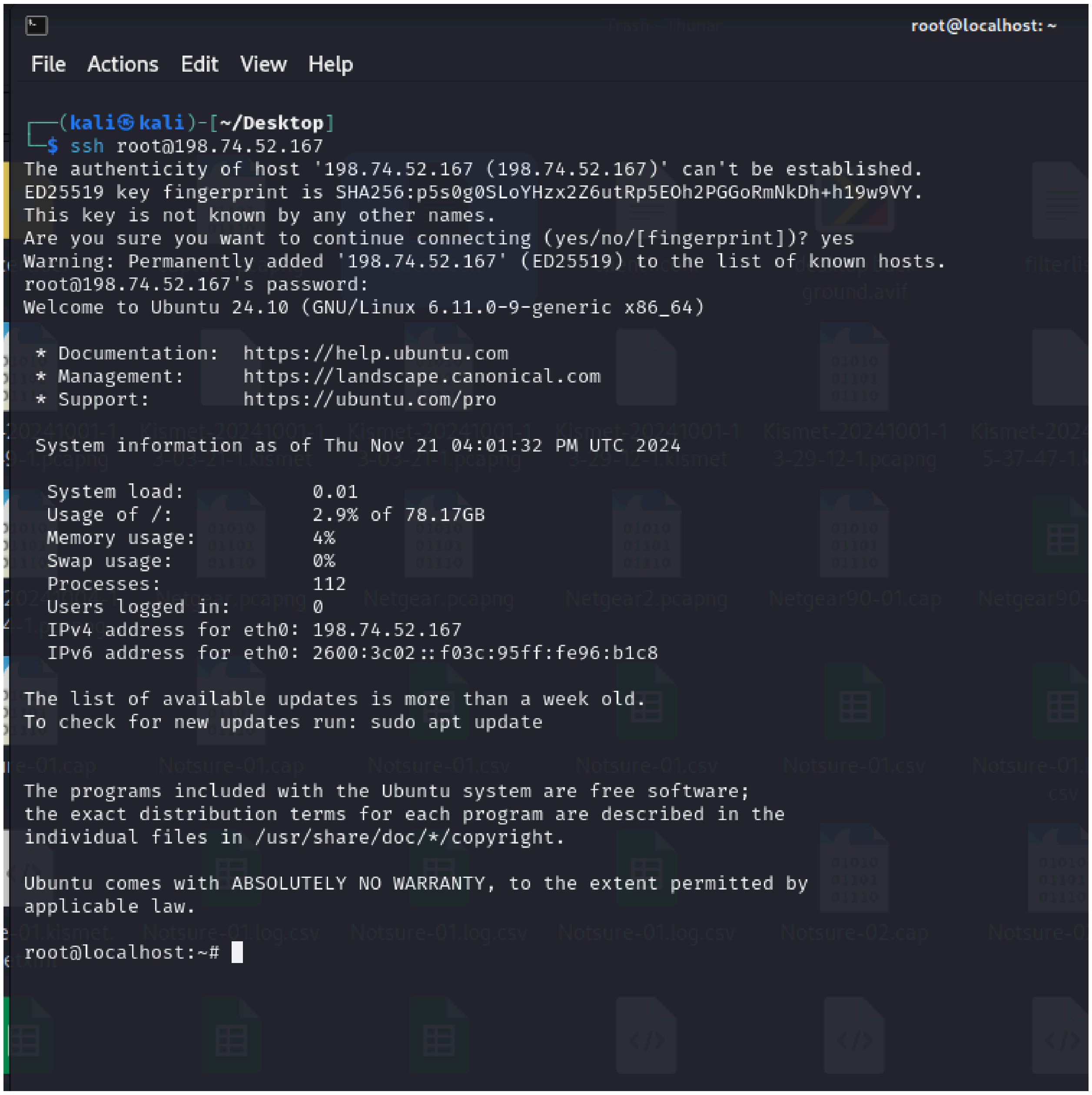
WireGuard Sever:

Step 1:

- Create a sever, I uses Linode running Ubuntu.
- https://cloud.linode.com/linodes/67114829
- On my VM, I SSH’d into my sever.



```
┌──(kali㉿kali)-[~/Desktop]
└─$ ssh root@198.74.52.167
The authenticity of host '198.74.52.167 (198.74.52.167)' can't be established.
ED25519 key fingerprint is SHA256:p5s0g0SLoYHzx2Z6utRp5EOh2PGGoRmNkDh+h19w9VY.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '198.74.52.167' (ED25519) to the list of known hosts.
root@198.74.52.167's password:
Welcome to Ubuntu 24.10 (GNU/Linux 6.11.0-9-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Thu Nov 21 04:01:32 PM UTC 2024

  System load:           0.01
  Usage of /:            2.9% of 78.17GB
  Memory usage:          4%
  Swap usage:            0%
  Processes:             112
  Users logged in:       0
  IPv4 address for eth0: 198.74.52.167
  IPv6 address for eth0: 2600:3c02::f03c:95ff:fe96:b1c8

The list of available updates is more than a week old.
To check for new updates run: sudo apt update


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

root@localhost:~#
```

1. Install WireGuard Tools

```
sudo apt install wireguard-tools
```

- This command installs the **WireGuard** command-line tools on your cloud server. The `wireguard-tools` package provides the necessary utilities (`wg`, `wg-quick`, etc.) for managing and configuring WireGuard VPN connections.

- The `sudo` part ensures the command runs with administrative privileges, necessary for installing software on the system.

2. **Switch to the Root User**

```
sudo su #if you are not already in root
```

- This command switches you to the root user. While you can run commands with `sudo`, switching to the root user can make it easier to execute multiple commands without needing to prefix each one with `sudo`.
- **Caution**: Running as `root` can be risky if not done carefully because you have unrestricted access to the system.

3. **Navigate to the WireGuard Configuration Directory**

```
cd /etc/wireguard/
```

- WireGuard configuration files are stored in `/etc/wireguard/`. This command changes the working directory to that location, where you will create the configuration files for WireGuard.

4. **Generate the Private Key (Keep a copy, you will need to enter this in later.)**

```
wg genkey > privatekey

#root@localhost:/etc/wireguard# cat privatekey
sFpRbh9rG48fHZ+gz6Ps7u7ce4oKA9A0HEZSGOmh3GE=
```

- This command uses the `wg` (WireGuard) tool to generate a **private key** for your server.
- The `> privatekey` part saves the generated private key to a file named `privatekey`.
- **Important**: The private key must be kept secure and should never be shared with anyone else. It's used to authenticate the server.

5. **Set Correct Permissions on the Private Key**

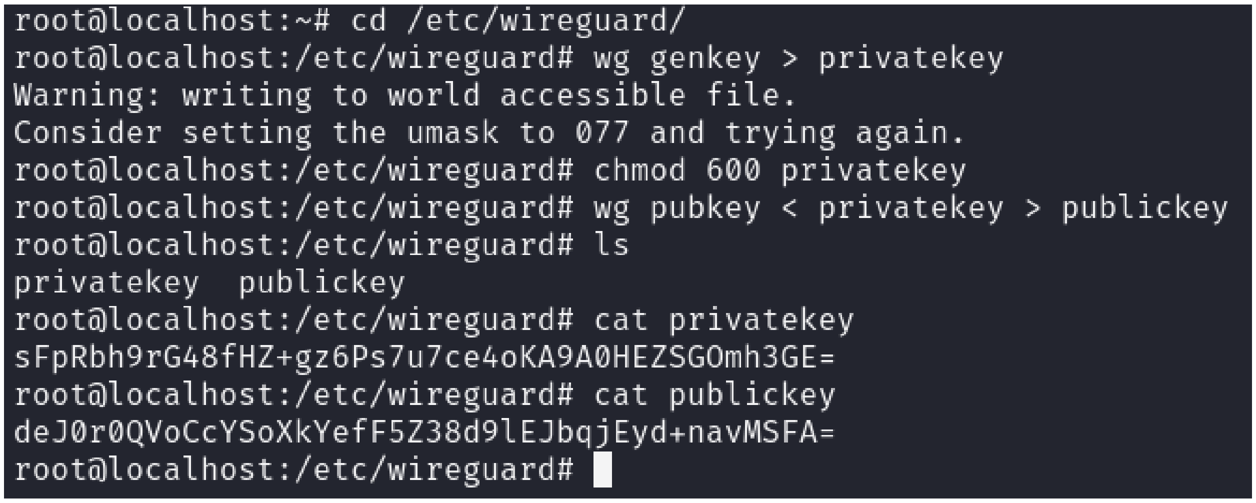
```
chmod 600 privatekey
```

- This command sets the file permissions for `privatekey` to `600`, which means:
    - Only the file owner (root in this case) has read and write permissions.
    - No other user or group can access the file.
- This is important to ensure the private key remains secure.

6. **Generate the Public Key from the Private Key**

```
wg pubkey < privatekey > publickey

#root@localhost:/etc/wireguard# cat publickey
deJ0r0QVoCcYSoXkYefF5Z38d9lEJbqjEyd+navMSFA=
```

- This command generates the **public key** from the previously generated private key.
- The `< privatekey` means the input is the private key file, and `> publickey` saves the generated public key into the `publickey` file.
- The public key is safe to share with peers (other devices connecting to the VPN).



```
root@localhost:~# cd /etc/wireguard/
root@localhost:/etc/wireguard# wg genkey > privatekey
Warning: writing to world accessible file.
Consider setting the umask to 077 and trying again.
root@localhost:/etc/wireguard# chmod 600 privatekey
root@localhost:/etc/wireguard# wg pubkey < privatekey > publickey
root@localhost:/etc/wireguard# ls
privatekey  publickey
root@localhost:/etc/wireguard# cat privatekey
sFpRbh9rG48fHZ+gz6Ps7u7ce4oKA9A0HEZSGOmh3GE=
root@localhost:/etc/wireguard# cat publickey
deJ0r0QVoCcYSoXkYefF5Z38d9lEJbqjEyd+navMSFA=
root@localhost:/etc/wireguard# 
```

7. Create the WireGuard Configuration File

```
nano wg0.conf
```

- This command opens the `wg0.conf` file using the `vim` text editor. This file will hold the configuration settings for the WireGuard VPN interface.
- If `vim` is not installed on your server, you can use other editors like `nano` (`sudo apt install nano`), but `vim` is widely available.

Here's a quick reference for how to **save, exit, and edit** in both **Vim** and **Nano** editors:

Vim Commands

1. **Enter Command Mode**:
   - When you first open Vim, you're in **Normal mode**. To enter **Command mode** (if you're in Insert or Visual mode), press `Esc`.
2. **Save and Exit**:
   - To **save and exit** Vim, type the following in Command mode:

     ```
     :wq
     ```

     - `w` stands for "write" (save), and `q` stands for "quit."

    - Then press `Enter`.

3. **Exit Without Saving**:
   - If you want to **exit without saving**:

     ```
     :q!
     ```

     - `q` stands for "quit," and `!` forces it to quit even if there are unsaved changes.

4. **Save Without Exiting**:
   - If you want to **save the file without exiting**:

     ```
     :w
     ```

     - `w` writes the changes to the file.

5. **Edit Text**:
   - To **start editing** (insert mode), press `i` (Insert mode) after pressing `Esc` to make sure you're in Normal mode.
   - After you're done editing, press `Esc` again to return to Normal mode.

Nano Commands

1. **Save and Exit**:
   - To **save** your work in Nano, press:

     ```
     Ctrl + O
     ```

     - This opens a prompt at the bottom asking for the file name. Just press `Enter` to confirm the current file name.
   - To **exit** Nano after saving:

     ```
     Ctrl + X
     ```

2. **Exit Without Saving**:
    - If you **want to exit without saving**, press:

      ```
      Ctrl + X
      ```

        - If you've made changes, Nano will ask if you want to save. Type `N` to discard changes and exit.
3. **Save Without Exiting**:
    - To **save** the file but stay in the editor:

      ```
      Ctrl + O
      ```

        - Confirm the file name and press `Enter`.
4. **Edit Text**:
    - Nano is always in **Edit mode**, so you can start typing right away when you open the file.

Summary:

Command	Vim	Nano
Save	`:w`	`Ctrl + O`
Save & Exit	`:wq`	`Ctrl + O`, then `Ctrl + X`
Exit (without saving)	`:q!`	`Ctrl + X`, then `N`
Exit	`:q`	`Ctrl + X`

Let me know if you need more details on Vim or Nano commands!

8. Add Configuration to the `wg0.conf` File

Inside the `wg0.conf` file, you need to define several parameters for the WireGuard VPN interface. Below is what you would typically add to the file:

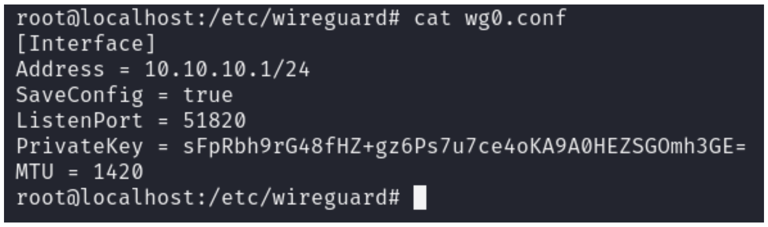
Make sure you have your private and public key cat out and saved before going on.

```
[Interface]
Address = 10.10.10.1/24
SaveConfig = true
ListenPort = 51820
PrivateKey = "Insert your PRIVATEKEY here"
MTU = 1420
```

Explanation of each parameter:

- **[Interface]**: This section defines the local interface settings for WireGuard.
- **Address**: The IP address range assigned to this server's WireGuard interface. `10.10.10.1/24` means the server's WireGuard IP is `10.10.10.1` with a subnet mask of `/24`. The subnet `/24` allows up to 254 devices (IPs) in this VPN network.
- **SaveConfig**: If set to `true`, it allows the configuration to be automatically saved when the interface is brought down (stopped). This is useful for automatically saving dynamic configurations.
- **ListenPort**: This is the port on which WireGuard will listen for incoming VPN connections. Here, the server will listen on port `51820` (which is the default WireGuard port).
- **PrivateKey**: This is where you input the private key generated earlier. You need to copy the contents of the `privatekey` file here. You would replace `PRIVATEKEY` with the actual key (e.g., `PrivateKey = XyZABCD1234567...`).
- **MTU**: The **Maximum Transmission Unit** defines the maximum size of a packet that can be transmitted. When traffic is sent, there are bits on the beginning of the traffic.`MTU = 1420` is a safe value to avoid fragmentation issues, but you can adjust this depending on your network requirements. A value that is too high might cause fragmentation.

```
cat wg0.conf
```


```
root@localhost:/etc/wireguard# cat wg0.conf
[Interface]
Address = 10.10.10.1/24
SaveConfig = true
ListenPort = 51820
PrivateKey = sFpRbh9rG48fHZ+gz6Ps7u7ce4oKA9A0HEZSGOmh3GE=
MTU = 1420
root@localhost:/etc/wireguard# 
```

9. **Enable the WireGuard Interface to Start on Boot**

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

- This command enables **systemd** to automatically start the WireGuard interface at boot using the `wg0` configuration. `wg-quick@wg0` refers to the WireGuard interface defined in the `wg0.conf` file.
- Enabling it ensures the VPN is active when the server is restarted.

10. **Start the WireGuard Interface**

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

- This command starts the WireGuard VPN interface immediately using the `wg0.conf` configuration.
- After starting the interface, the server will begin listening on port `51820` for incoming WireGuard connections, using the keys and settings defined in the configuration file.

IP Forwarding (Routing Between Networks)

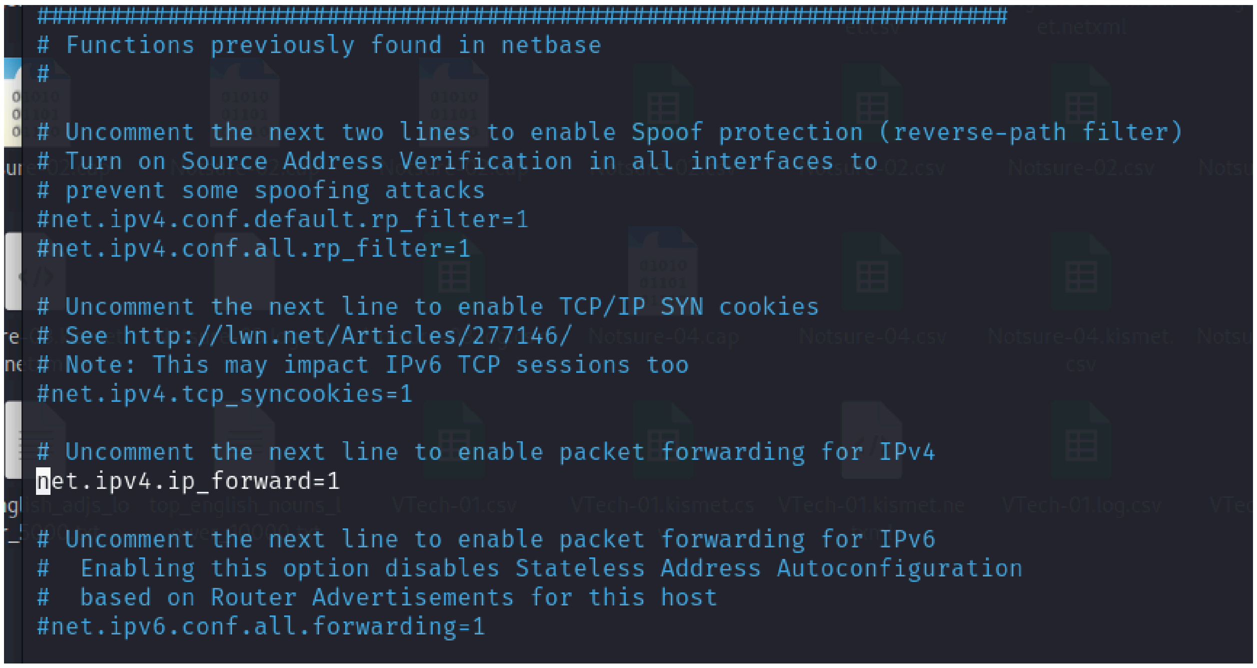
1. **Enable IP Forwarding (Routing Between Networks)**

IP forwarding allows your cloud server to act as a router by forwarding network traffic between different network interfaces (for example, between your VPN and the internet).

Command 1: Edit `sysctl.conf` to Enable IP Forwarding

```
sudo nano /etc/sysctl.conf
```

- This command opens the system configuration file `/etc/sysctl.conf` in `vim` to enable or modify system parameters.
- In this file, you will enable IP forwarding by uncommenting (removing the `#` symbol) the line that specifies IP forwarding for IPv4 traffic.



```
###################################################################
# Functions previously found in netbase
#

# Uncomment the next two lines to enable Spoof protection (reverse-path filter)
# Turn on Source Address Verification in all interfaces to
# prevent some spoofing attacks
#net.ipv4.conf.default.rp_filter=1
#net.ipv4.conf.all.rp_filter=1

# Uncomment the next line to enable TCP/IP SYN cookies
# See http://lwn.net/Articles/277146/
# Note: This may impact IPv6 TCP sessions too
#net.ipv4.tcp_syncookies=1

# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1

# Uncomment the next line to enable packet forwarding for IPv6
#  Enabling this option disables Stateless Address Autoconfiguration
#  based on Router Advertisements for this host
#net.ipv6.conf.all.forwarding=1
```

- This setting enables IPv4 forwarding on your system, allowing the server to route packets between different network interfaces.
- We get rid of the # because if we dont the program wont read the script. If the script sees a # it skips that line.

Command 3: Apply the Configuration Changes

```
sudo sysctl -p /etc/sysctl.conf
```

- This command applies the changes made in the `sysctl.conf` file. It tells the system to reload the settings, enabling IP forwarding.

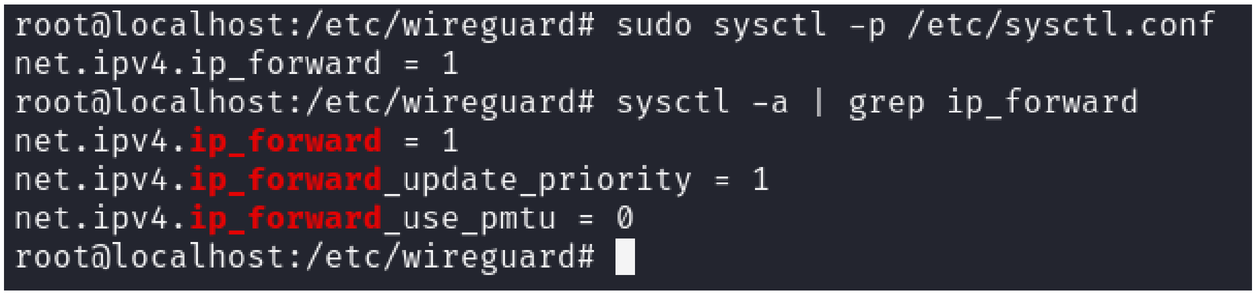
Command 4: Verify IP Forwarding

```
sysctl -a | grep ip_forward
```

- This command checks whether IP forwarding has been successfully enabled.
- You should see an output similar to:

  ```
  net.ipv4.ip_forward = 1
  net.ipv4.ip_forward_update_priority = 1
  net.ipv4.ip_forward_use_pmtu = 0
  ```

  - `net.ipv4.ip_forward = 1` confirms that IPv4 forwarding is enabled.



```
root@localhost:/etc/wireguard# sudo sysctl -p /etc/sysctl.conf
net.ipv4.ip_forward = 1
root@localhost:/etc/wireguard# sysctl -a | grep ip_forward
net.ipv4.ip_forward = 1
net.ipv4.ip_forward_update_priority = 1
net.ipv4.ip_forward_use_pmtu = 0
root@localhost:/etc/wireguard#
```

---

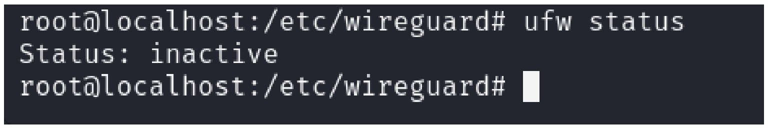
2. **Set Up UFW (Uncomplicated Firewall) and iptables Rules for Port Forwarding**

Now, we need to configure the firewall (`UFW`) and network address translation (`iptables`) to handle the forwarding of traffic between the VPN interface (`wg0`) and external networks (such as the internet).

Command 1: Check UFW Status

```
ufw status
```

- This command shows the status of the firewall and its current active rules. You can see whether it's enabled and which ports are allowed or blocked.



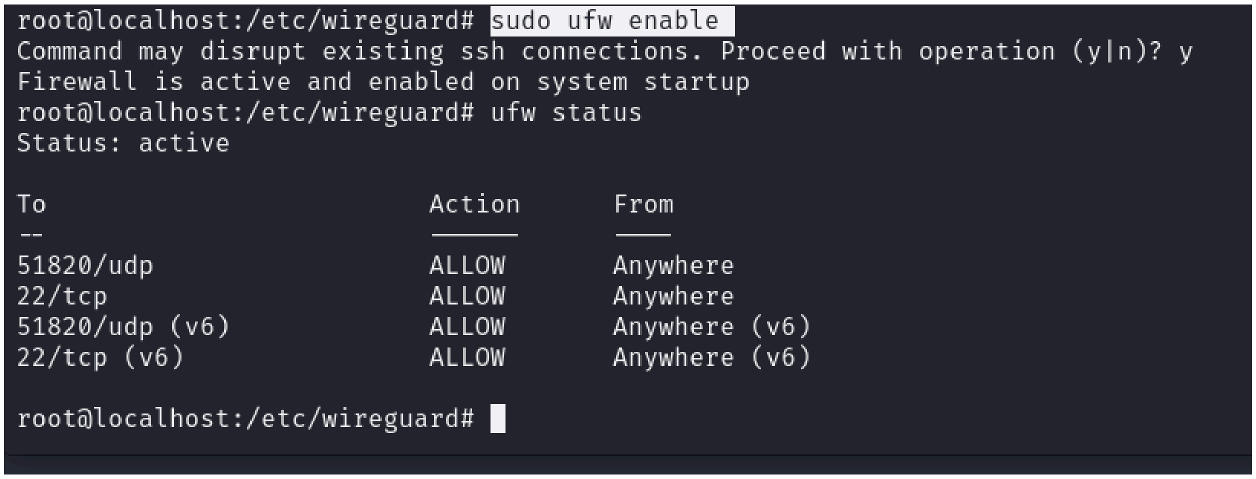
```
root@localhost:/etc/wireguard# ufw status
Status: inactive
root@localhost:/etc/wireguard#
```

Command 2: Allow UDP Traffic on Port 51820

```
sudo ufw allow 51820/udp
sudo ufw allow 22/tcp #so you can still SSH into your device
sudo ufw enable #click yes afterwards
```

- This command creates a firewall rule to allow incoming UDP traffic on port `51820`, which is the default WireGuard port. This ensures that your WireGuard VPN can receive connections.

```
ufw status #check to see if everything is working
```



```
root@localhost:/etc/wireguard# sudo ufw enable
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
Firewall is active and enabled on system startup
root@localhost:/etc/wireguard# ufw status
Status: active

To                         Action      From
--                         ------      ----
51820/udp                  ALLOW       Anywhere
22/tcp                     ALLOW       Anywhere
51820/udp (v6)             ALLOW       Anywhere (v6)
22/tcp (v6)                ALLOW       Anywhere (v6)

root@localhost:/etc/wireguard#
```

Command 3: Set Default Rule to Allow Forwarding

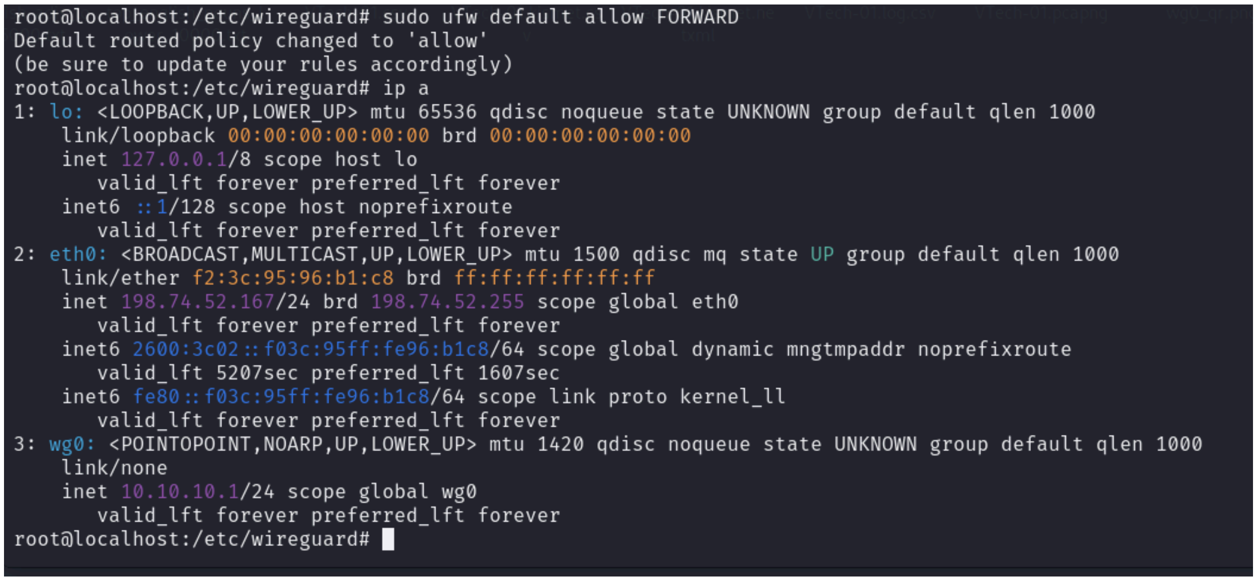
```
sudo ufw default allow FORWARD
```

- This command sets the default policy to allow forwarding of traffic between interfaces. Without this, UFW might block the routing of traffic between different interfaces, which could break the VPN functionality.

Command 4: Check Network Interfaces

```
ip -br a
```

- This command shows the network interfaces on your server. You'll use this to see which interface is connected to the VPN (`wg0` in this case) and to check the IP addresses of your network interfaces.



```
root@localhost:/etc/wireguard# sudo ufw default allow FORWARD
Default routed policy changed to 'allow'
(be sure to update your rules accordingly)
root@localhost:/etc/wireguard# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether f2:3c:95:96:b1:c8 brd ff:ff:ff:ff:ff:ff
    inet 198.74.52.167/24 brd 198.74.52.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 2600:3c02::f03c:95ff:fe96:b1c8/64 scope global dynamic mngtmpaddr noprefixroute
       valid_lft 5207sec preferred_lft 1607sec
    inet6 fe80::f03c:95ff:fe96:b1c8/64 scope link proto kernel_ll
       valid_lft forever preferred_lft forever
3: wg0: <POINTOPOINT,NOARP,UP,LOWER_UP> mtu 1420 qdisc noqueue state UNKNOWN group default qlen 1000
    link/none
    inet 10.10.10.1/24 scope global wg0
       valid_lft forever preferred_lft forever
root@localhost:/etc/wireguard# █
```

Command 5: Edit `before.rules` for NAT (Network Address Translation) Configuration

```
sudo nano /etc/ufw/before.rules
```

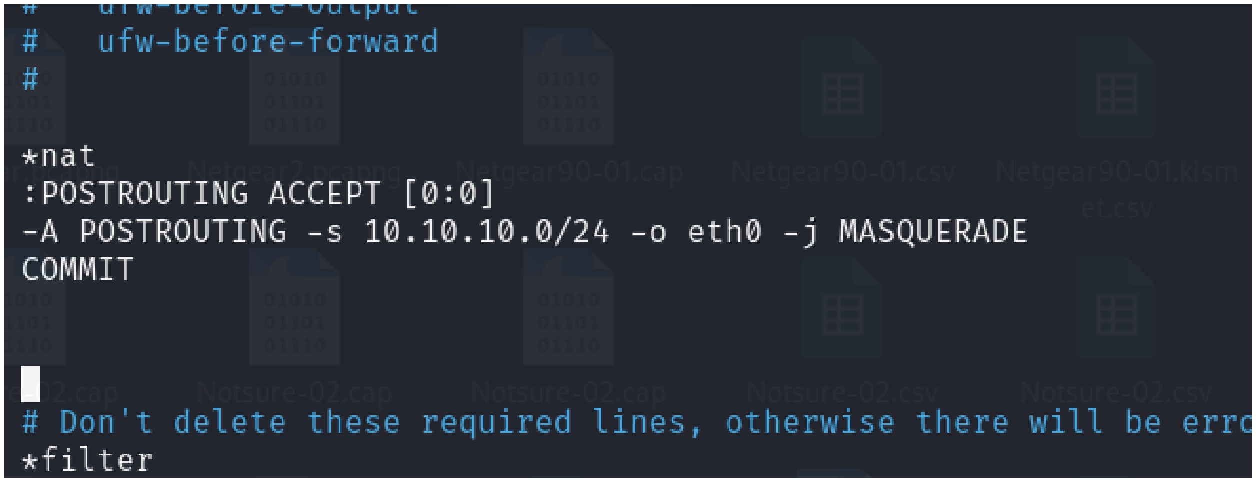
- This command opens the `before.rules` file, which is used to define custom firewall rules that are applied before the default UFW rules.

Command 6: Add NAT Rules to `before.rules`

Inside the `/etc/ufw/before.rules` file, add the following rules to set up **Network Address Translation (NAT)**. These rules ensure that traffic originating from the VPN network is properly routed to the internet.

```
*nat
:POSTROUTING ACCEPT [0:0]
-A POSTROUTING -s {VPN_NETWORK/SUBNET} -o {EXTERNAL_INTERFACE_I
ike eth0} -j MASQUERADE
COMMIT

#might need to enable ufw if it doesnt take the command
```

- Replace `{VPN_NETWORK/SUBNET}` with the actual subnet of your VPN network (e.g., `10.10.10.0/24`).
- Replace `{EXTERNAL_INTERFACE}` with the name of the network interface that connects to the internet (e.g., `eth0` or `ens3`).
- These NAT rules enable the server to masquerade (perform source NAT) so that VPN clients can use the server's IP address when accessing external networks.



```
#   ufw-before-forward
#

*nat
:POSTROUTING ACCEPT [0:0]
-A POSTROUTING -s 10.10.10.0/24 -o eth0 -j MASQUERADE
COMMIT


# Don't delete these required lines, otherwise there will be erro
*filter
```

Command 7: Save and Quit

- In `vim`, after adding the necessary lines, press `Esc`, type `:wq`, and press `Enter` to save and quit the editor.

---

3. Allow Routing on WireGuard Interface (`wg0`)

Command 1: Allow Routing on `wg0` Interface

```
sudo ufw route allow in on wg0
```

- This command allows traffic to be routed through the `wg0` interface, which is the WireGuard VPN interface. It enables clients on the VPN to access the server and the internet.

Command 2: Reload UFW to Apply Changes

```
sudo ufw reload
```

- This command reloads UFW to apply the new firewall rules that you have configured.

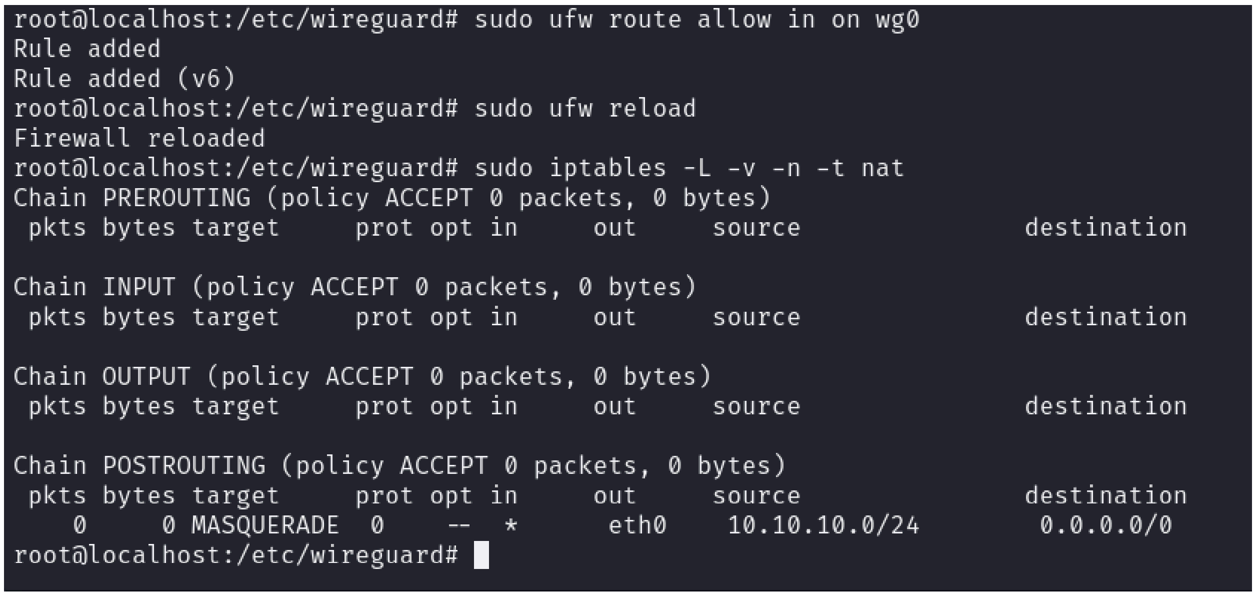
---

4. **Verify iptables Rules**

Command 1: List iptables Rules for NAT Table

```
sudo iptables -L -v -n -t nat
```

- This command lists all **NAT** table rules in **iptables**, with detailed information (`v`) and numeric output (`n`).
- You should see the NAT rule you added earlier in the `POSTROUTING` chain, ensuring that the server can properly handle and forward traffic from the VPN to the internet.



```
root@localhost:/etc/wireguard# sudo ufw route allow in on wg0
Rule added
Rule added (v6)
root@localhost:/etc/wireguard# sudo ufw reload
Firewall reloaded
root@localhost:/etc/wireguard# sudo iptables -L -v -n -t nat
Chain PREROUTING (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 MASQUERADE  0    --  *      eth0    10.10.10.0/24        0.0.0.0/0
root@localhost:/etc/wireguard#
```

The command `sudo iptables -L -v -n -t nat` is used to list (display) the current firewall rules in the `nat` (Network Address Translation) table of `iptables`, with additional details.

Let's break it down into parts:

Command Breakdown:

1. **`iptables`**:
   - `iptables` is a command-line tool used for configuring and managing the Linux kernel firewall. It can be used to control incoming and outgoing traffic on a network interface.
2. **`L`**:
   - This option stands for "list". It tells `iptables` to display the rules currently applied to the firewall.
   - This will list all the current rules in the specified chain or table.
3. **`v`**:
   - `v` stands for "verbose". When you add `v`, `iptables` provides more detailed information about the rules, such as packet and byte counts (how many packets and how much data have been processed by each rule).
   - This option gives you a more detailed output compared to the basic `iptables -L` command.
4. **`n`**:
   - `n` prevents `iptables` from resolving IP addresses or port names to their symbolic names (like translating IPs to hostnames or ports to service names).
   - By using `n`, you get numeric output (e.g., IP addresses and port numbers), which is usually faster because `iptables` does not have to perform DNS or service lookups.
5. **`t nat`**:
   - `t` specifies the table you want to interact with. In this case, `t nat` tells `iptables` to show rules from the **NAT (Network Address Translation)** table.

- The `nat` table is used for network address translation (such as port forwarding, IP masquerading, etc.). It's primarily used for manipulating the source and destination addresses of packets.
- Other common tables include `filter` (default table for filtering traffic) and `mangle` (for specialized packet modifications).

Putting it all together:

- `sudo iptables -L -v -n -t nat` will list all the rules in the **NAT table**, with detailed information and numerical IP addresses/ports, so you can see how your network traffic is being modified or forwarded.
- For example, this could show port forwarding rules, IP masquerading (e.g., for sharing a network connection), or other network address translations.

Example Output:

Here's an example of what the command's output might look like:

```
Chain PREROUTING (policy ACCEPT 12345 packets, 123456 bytes)
 pkts bytes target     prot opt in     out     source               destination
  123  45678 DNAT       tcp  --  *      *       0.0.0.0/0            192.168.1.100        tcp
dpt:8080 to:192.168.1.200:80

Chain POSTROUTING (policy ACCEPT 12345 packets, 123456 bytes)
 pkts bytes target     prot opt in     out     source               destination
   45  23456 MASQUERADE  all  --  *      *       192.168.1.0/24       0.0.0.0/0

Chain OUTPUT (policy ACCEPT 12345 packets, 123456 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

Explanation of the Output:

- **Chain PREROUTING**: This chain handles traffic before routing occurs. It's often used for DNAT (Destination Network Address Translation), which is used for things like port forwarding.

    - The `DNAT` rule is forwarding TCP traffic that is destined for port 8080 to a different IP (192.168.1.200) on port 80.
- **Chain POSTROUTING**: This chain handles traffic after routing occurs. It's used for SNAT (Source Network Address Translation) or MASQUERADE (commonly used for NAT gateway setups).
    - The `MASQUERADE` rule is modifying the source address of traffic coming from a local network (192.168.1.0/24) to appear as if it's coming from the server's external IP. This is typical in scenarios like internet sharing.
- **Chain OUTPUT**: This chain deals with traffic generated from the local system itself.

Summary:

- **`sudo iptables -L -v -n -t nat`** displays detailed (verbose) rules in the **NAT table** (used for address translation) of `iptables`, showing packet statistics and numeric addresses.
- This is useful for diagnosing or configuring network address translation rules such as port forwarding or masquerading.

Let me know if you need more clarification!

WireGuard Client Configuration

1. **Create a Directory for WireGuard Client Configuration**

```
sudo mkdir /etc/wireguard/clients
cd clients
```

- This command creates a new directory `/etc/wireguard/clients` where you'll store the client configuration file and key. It's standard practice to keep client configurations organized in a separate directory for security and management.

2. **Switch to Root Privileges**

```
sudo bash
```

or

```
sudo -i
```

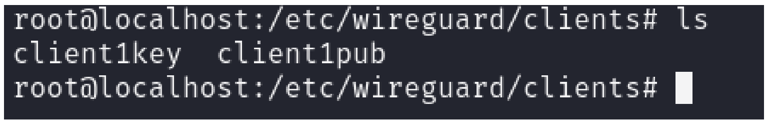
- These commands allow you to switch to the **root user**. You need root privileges to manage WireGuard configurations and generate keys. The difference is:
  - `sudo bash` opens a new shell with root privileges.
  - `sudo -i` starts an interactive root shell, providing a root environment.

3. Generate the Client's Private and Public Keys (put a number with the clientkey so you can create multiple clients and not write over other keys)

```
#CD into your clients folder
wg genkey > client1key

#root@localhost:/etc/wireguard/clients# cat client1key
eKnFzOnM/2NiRjGNuwcBjxIW5jCN6xIkoURzFfwbR2Q=

wg pubkey < client1key > client1pub

#root@localhost:/etc/wireguard/clients# cat client1pub
2aZZsGR437P2yuiMfF0cHT2pVTvsnUAW2z5JOrFpbSM=

chmod 600 client1key
```

- `wg genkey > client1key`: Generates a new **private key** for the client and saves it to the file `client1key`.
- `wg pubkey < client1key > client1pub`: Uses the **private key** from the file `client1key` to generate the corresponding **public key** and saves it to `client1pub`.

- These keys are required for authenticating the client on the server.



```
root@localhost:/etc/wireguard/clients# ls
client1key  client1pub
root@localhost:/etc/wireguard/clients# 
```

4. Client Configuration File: `client1.conf`

Next, you need to create a configuration file for the client to allow it to connect to the WireGuard server.

```
nano client1.conf
```

- This command opens the `vim` text editor to create and edit the WireGuard configuration file for the client. Inside this file, you will paste the following configuration.

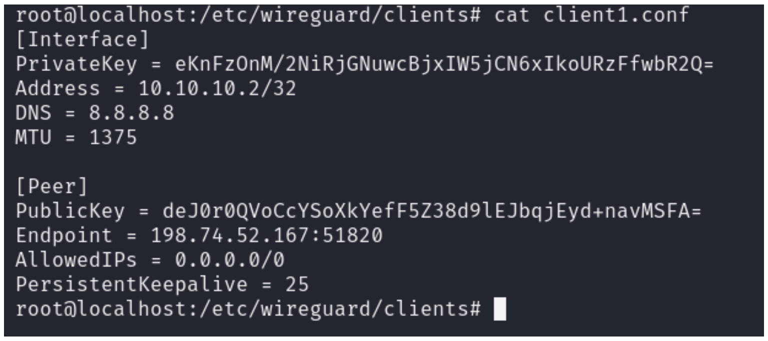
Client Configuration File (`client1.conf`):

```
[Interface]
PrivateKey = "INSERT_CLIENTS_PRIVATE_KEY"
Address = 10.10.10.2/32 #take note, more clients you make, the number chang
es 2,3,4
DNS = 8.8.8.8
MTU = 1375

[Peer]
PublicKey = "INSERT_SERVERS_PUBLIC_KEY"
Endpoint = "INSERT_IP_ADDRESS_OF_SERVER":51820 #the one you SSH to, n
ot 10.10.10.1
AllowedIPs = 0.0.0.0/0
PersistentKeepalive = 25
```

Let's break down what each section means:

- **[Interface] Section**: This defines the client's settings.

    - `PrivateKey = CLIENTS_PRIVATE_KEY`: Replace `CLIENTS_PRIVATE_KEY` with the private key you generated earlier (`client1key`).
    - `Address = 10.10.10.2/32`: This assigns the client an IP address in the VPN network (use the next available IP in your VPN subnet, `10.10.10.2/32` here).
    - `DNS = 8.8.8.8`: This configures Google's DNS server for the client.
    - `MTU = 1375`: This is the **Maximum Transmission Unit**. A value of `1375` can help prevent fragmentation, especially in some network environments.
- **[Peer] Section**: This defines the connection to the **server**.
    - `PublicKey = SERVERS_PUBLIC_KEY`: Replace `SERVERS_PUBLIC_KEY` with the public key of your WireGuard server.
    - `Endpoint = IP_ADDRESS_OF_SERVER:51820`: This is the IP address and port where the server is reachable (replace `IP_ADDRESS_OF_SERVER` with the actual server IP).
    - `AllowedIPs = 0.0.0.0/0`: This routes all traffic through the VPN. `0.0.0.0/0` means all IPv4 addresses.
    - `PersistentKeepalive = 25`: This keeps the connection alive by sending a packet every 25 seconds, which helps maintain the VPN connection through NAT or firewalls.



```
root@localhost:/etc/wireguard/clients# cat client1.conf
[Interface]
PrivateKey = eKnFzOnM/2NiRjGNuwcBjxIW5jCN6xIkoURzFfwbR2Q=
Address = 10.10.10.2/32
DNS = 8.8.8.8
MTU = 1375

[Peer]
PublicKey = deJ0r0QVoCcYSoXkYefF5Z38d9lEJbqjEyd+navMSFA=
Endpoint = 198.74.52.167:51820
AllowedIPs = 0.0.0.0/0
PersistentKeepalive = 25
root@localhost:/etc/wireguard/clients# 
```

5. Save and Quit `vim` Editor

- Press `Esc` to leave editing mode.
- Type `:wq` (write and quit) and press `Enter` to save the file and exit the editor.

6. Configure the Server to Recognize the Client

```
#while in your client directory
wg set wg0 peer $(cat client1pub) allowed-ips 10.10.10.2/32
```

- This command configures the WireGuard server to accept the client (`client1pub`) and allow its IP (`10.10.10.2/32`) for routing.
- `wg set wg0`: This is the command to modify the configuration of WireGuard interface `wg0`.
- `peer $(cat client1pub)`: This part of the command adds the client's **public key** (`client1pub`) as a peer for the WireGuard interface.
- `allowed-ips 10.10.10.2/32`: This ensures that the server allows traffic destined for the client's IP (`10.10.10.2/32`).

7. Restart WireGuard Interface

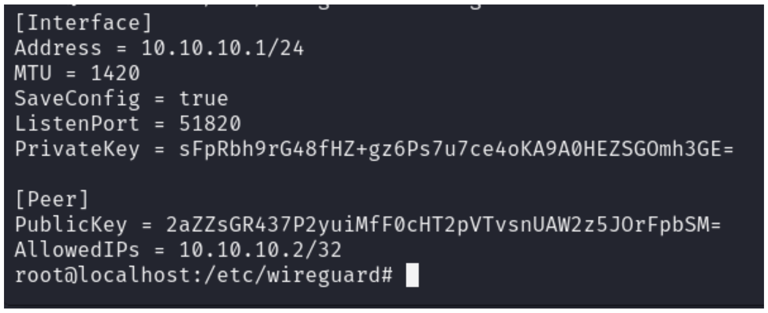
```
systemctl restart wg-quick@wg0
```

- This command restarts the WireGuard interface (`wg0`) to apply the new settings (including the new client configuration).

8. Verify Server Configuration

```
cat ../wg0.conf
```

- This command outputs the content of the WireGuard server's configuration file (`wg0.conf`), so you can verify that the settings are correct and that the client has been added properly as a peer.



```
[Interface]
Address = 10.10.10.1/24
MTU = 1420
SaveConfig = true
ListenPort = 51820
PrivateKey = sFpRbh9rG48fHZ+gz6Ps7u7ce4oKA9A0HEZSGOmh3GE=

[Peer]
PublicKey = 2aZZsGR437P2yuiMfF0cHT2pVTvsnUAW2z5JOrFpbSM=
AllowedIPs = 10.10.10.2/32
root@localhost:/etc/wireguard# 
```

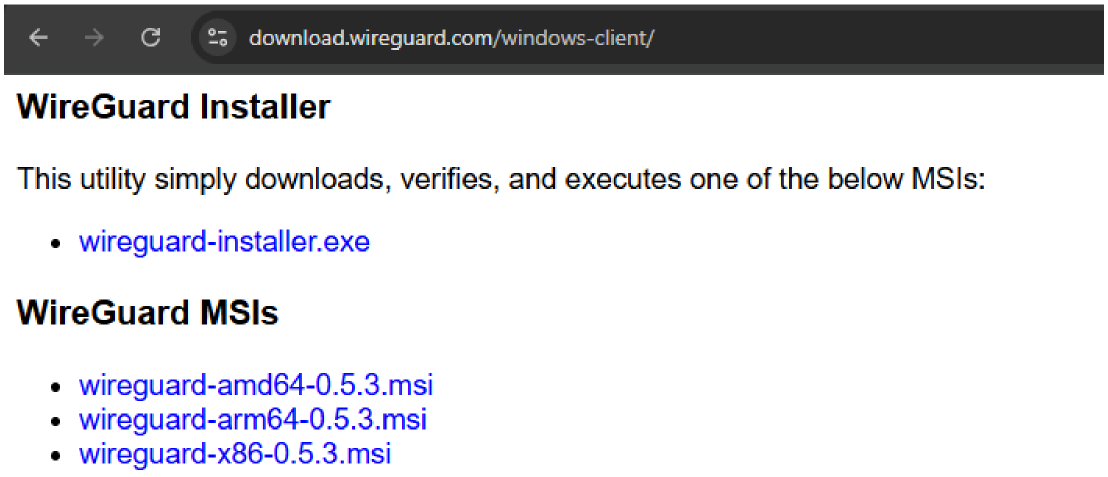
!!!STOP!!!

To set up more clients, do it the same way you did it for the first client, but make sure you change the IP.

9. Adding Client configuration to Linux and Windows Computers.

Windows Computers

1. Download and Install WireGuard
Visit the official WireGuard website.
Download the Windows installer and run it.
Follow the prompts to complete the installation.



download.wireguard.com/windows-client/

WireGuard Installer

This utility simply downloads, verifies, and executes one of the below MSIs:

- wireguard-installer.exe

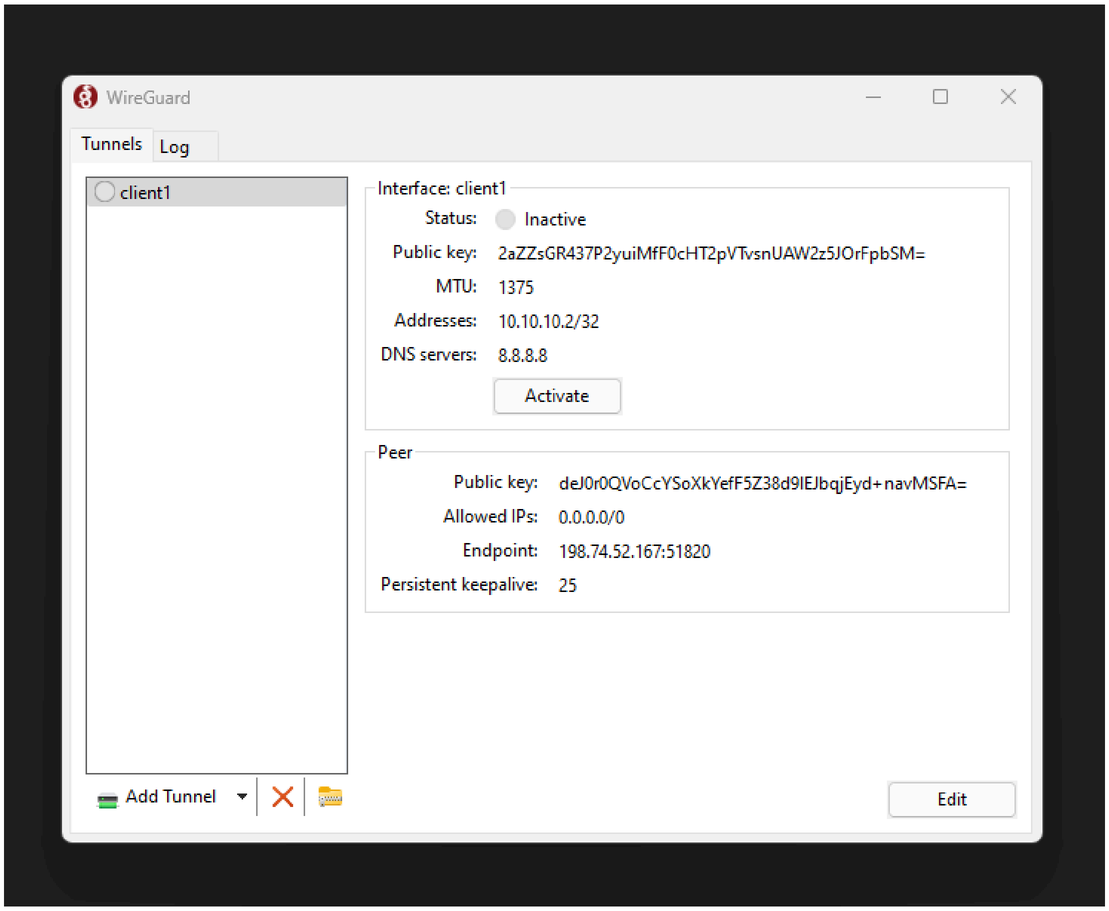
WireGuard MSIs

- wireguard-amd64-0.5.3.msi
- wireguard-arm64-0.5.3.msi
- wireguard-x86-0.5.3.msi

2. Obtain the Configuration File
Get the .conf file from your VPN provider or generate it manually if you're setting

up your own WireGuard server.
The file typically contains your private key, server address, and other network settings.

3. Import the Configuration File
Open the WireGuard app on your Windows PC.
Click “Import Tunnel(s) from File” or drag and drop the .conf file into the WireGuard window.
Give the tunnel a name if prompted.



Interface: client1

Status:	Inactive
Public key:	2aZZsGR437P2yuiMfF0cHT2pVTvsnUAW2z5JOrFpbSM=
MTU:	1375
Addresses:	10.10.10.2/32
DNS servers:	8.8.8.8

Peer

Public key:	deJ0r0QVoCcYSoXkYefF5Z38d9IEJbqjEyd+navMSFA=
Allowed IPs:	0.0.0.0/0
Endpoint:	198.74.52.167:51820
Persistent keepalive:	25

4. Activate the Tunnel
Click the toggle switch to activate the tunnel.
The status will change to Active once connected.

5. Test the Connection
Verify your connection by visiting a site like https://whatismyip.com to check your new IP address.
Ensure it's different from your regular IP, indicating that you're using the VPN.

6. Optional: Configure Advanced Settings
Click on the tunnel name to view or manually edit the configuration details if needed.

9. **Move the Client Configuration to the User’s Directory**

```
cp client1.conf /home/student
```

- This command copies the `client1.conf` configuration file to the home directory of the user `student`.

10. **Change Ownership of the Configuration File**

```
chown student:student client1.conf
```

- This command changes the ownership of the `client1.conf` file to the user `student` and the group `student`. This ensures that the user `student` has the necessary permissions to access and modify the file.

Certainly! Here’s a detailed rewrite of the instructions, with added explanations for each step and some fixes where necessary:

---

1. **Install WireGuard on Your Linux Machine**

To get WireGuard up and running, you first need to install it. Depending on your Linux distribution, the installation process varies slightly:

For Debian-based systems (like Ubuntu):

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

- **Explanation**:
  - `sudo apt update`: Updates the local package database to ensure you install the latest version available.
  - `sudo apt install wireguard`: Installs the WireGuard package.

For Red Hat-based systems (like Fedora or CentOS):

```
sudo dnf install wireguard-tools
```

- **Explanation**:
  - `sudo dnf install wireguard-tools`: Installs the required WireGuard tools (including the `wg` and `wg-quick` commands) on Fedora or CentOS. This command uses `dnf` package manager for Red Hat-based distributions.

For Arch-based systems:

```
sudo pacman -S wireguard-tools
```

- **Explanation**:
  - `sudo pacman -S wireguard-tools`: Installs the WireGuard tools using the `pacman` package manager on Arch Linux or any Arch-based distribution.

2. Set Up the WireGuard Client Configuration

Once you have WireGuard installed, the next step is to configure your WireGuard client.

1. **Place the Configuration File**:
   You should have received a configuration file with a `.conf` extension (e.g., `client.conf`) from your VPN provider or network administrator. This file contains the necessary details like server IPs, keys, and protocols to connect to a WireGuard server.

2. **Move the Configuration File**:
   It's important to move the configuration file to a secure location, typically the `/etc/wireguard/` directory. You can also place it in your home directory (`~/`), but `/etc/wireguard/` is the conventional location.

   Example command to move the configuration file:

   ```
   sudo cp /path/to/your/client.conf /etc/wireguard/
   ```

   - **Explanation**:
     - `sudo cp /path/to/your/client.conf /etc/wireguard/`: Copies the `client.conf` file to `/etc/wireguard/`. Ensure to replace `/path/to/your/client.conf` with the actual file path.

3. **Set the Correct Permissions**:
   For security reasons, you should make sure that only the root user can read the configuration file. This ensures that unauthorized users do not access sensitive VPN configuration data.

   Command to set permissions:

   ```
   sudo chmod 600 /etc/wireguard/client.conf
   ```

   - **Explanation**:
     - `sudo chmod 600 /etc/wireguard/client.conf`: This command ensures that only the root user has read and write access to the file (`600` means `rw-------`), which enhances security.

3. Enable and Start the WireGuard Client

Now that your configuration is in place, you can bring up the WireGuard interface.

1. **Start the WireGuard Interface**:
   Use the `wg-quick` command to bring up the interface based on your configuration file. The name of the interface is derived from the configuration file name, excluding the `.conf` extension.

   Example:

```
sudo wg-quick up client
```

- **Explanation**:
    - `sudo wg-quick up client`: This command activates the WireGuard interface using the `client.conf` file. The `client` here refers to the configuration file name (without the `.conf` extension).

2. **Enable Auto-Start at Boot**:
   If you want WireGuard to automatically start when your system boots, enable the service using `systemctl`. This will make sure the WireGuard connection is established on boot.

   Example:

```
sudo systemctl enable wg-quick@client
```

- **Explanation**:
    - `sudo systemctl enable wg-quick@client`: This command configures `wg-quick` to start automatically at boot for the `client` configuration.

4. Check the Connection

After starting the WireGuard interface, you can verify if it's working correctly by checking its status. The `wg` command provides detailed information about the current WireGuard connection.

To check the connection:

```
sudo wg
```

- **Explanation**:
    - `sudo wg`: This command will display information about your WireGuard interface, including the peer connection details, data transfer statistics, and current status of the interface.

5. (Optional) Bring Down the Connection

If you ever need to disconnect the WireGuard VPN connection, you can bring down the interface with the following command:

```
sudo wg-quick down client
```

- **Explanation**:
    - `sudo wg-quick down client`: This command shuts down the WireGuard interface for the `client` configuration, effectively disconnecting the VPN.

Troubleshooting Notes:

- **Kernel Module**: Some systems may require the installation of the WireGuard kernel module. On modern Linux distributions, WireGuard is included in the kernel, but on others (e.g., older versions of Ubuntu or custom kernels), you may need to install the module manually. You can check if it's loaded with `lsmod | grep wireguard`.
- **Firewall Rules**: Make sure your firewall allows UDP traffic on the default WireGuard port (51820) or the port specified in your configuration file. If you have a restrictive firewall, this could block your connection.
- **Log Files**: If things aren't working as expected, checking the system logs can provide insights into potential issues. For example, `journalctl -xe` can show detailed logs about your WireGuard service.

Adding WireGuard to My Phone

To change the config to a QR code IOT use your phone

---

1. Install `qrencode`

To generate a QR code from your WireGuard configuration file, you need a tool called `qrencode`. This tool can take any text input (in this case, your `.conf` file) and convert it into a QR code.

Run this command to install `qrencode` on your machine:

```
sudo apt install qrencode
```

- **Explanation**:
  - `sudo apt install qrencode`: This command installs the `qrencode` package using your package manager (`apt`) on a Debian-based system (like Ubuntu). `qrencode` is a command-line tool that generates QR codes from plain text or files.
  - `sudo` ensures you have the necessary administrative privileges to install software.
  - After installation, you can use `qrencode` to generate QR codes from text files or strings.

---

2. **Navigate to the directory containing your WireGuard configuration file**

Next, you need to navigate to the directory where your WireGuard configuration file (`client1.conf`) is stored.

```
cd client
```

- **Explanation**:
  - `cd client`: This changes the current directory to the folder called `client`. Make sure to replace `client` with the actual directory name if it's different, or just navigate to where your `client1.conf` file is located.
  - This step ensures that you're in the correct directory where your `.conf` file resides, so the next command can work correctly.

---

3. **Generate the QR Code**

Now that you're in the correct directory, you can generate the QR code from your WireGuard configuration file. The `qrencode` command will read the configuration file and output a PNG image containing the QR code.

```
qrencode -t PNG -o output.png < client1.conf
```

- **Explanation**:
    - `qrencode`: This is the command used to generate the QR code.
    - `t PNG`: This option tells `qrencode` to output the QR code as a PNG image.
    - `o output.png`: Specifies the name of the output file where the QR code image will be saved. In this case, it will be saved as `output.png`.
    - `< client1.conf`: This redirects the contents of the `client1.conf` file as input to the `qrencode` command. Essentially, `qrencode` takes the text inside the configuration file (the VPN settings) and encodes it into a QR code.

---

4. Transfer the QR Code to Your Phone

Once you've generated the QR code (saved as `output.png`), you'll want to transfer it to your phone so you can scan it.

To transfer the file from your Linux machine to your phone, you can use `scp` (secure copy). `scp` is a command-line tool that allows you to securely transfer files between a local machine and a remote one (such as your phone if it's set up to accept file transfers).

```
scp username@remote_host:/path/to/remote/file /path/to/local/destination
```

- **Explanation**:
    - `scp`: This command is used to securely copy files between two machines over SSH (Secure Shell).
    - `username@remote_host`: Replace `username` with your login username on the remote machine (in this case, it could be your phone, if it has SSH access set up) and `remote_host` with the IP address or hostname of that machine.
    - `/path/to/remote/file`: This specifies the full path to the file on the remote machine (the QR code you generated, `output.png` in this case).
    - `/path/to/local/destination`: This specifies where on your local machine (the one running the `scp` command) you want to save the file. Usually, it's a directory where you want the file to be stored temporarily before transferring it to your phone.

Note:

- To use `scp` to transfer the file to your phone, your phone must have an SSH server running (which is uncommon but possible with tools like Termux on Android), or you can use a file transfer protocol like `adb` (Android Debug Bridge) for Android or AirDrop/another method for iOS.
- If you don't have SSH access to your phone, another simpler way would be to transfer the QR code via USB cable, Bluetooth, or a cloud storage service.

Recap of Process:

1. Install `qrencode` using `sudo apt install qrencode`.
2. Navigate to the directory containing your WireGuard configuration file (`client1.conf`).
3. Run the `qrencode` command to generate a PNG image (`output.png`) that represents your configuration file as a QR code.
4. Use `scp` to transfer the generated QR code to your phone.

Additional Tip:

Once you have the QR code on your phone, you can use WireGuard's mobile app (available for iOS and Android) to scan the code. Open the WireGuard app on your phone, select "Add a Tunnel", and then scan the QR code to automatically populate the connection settings.

```
sudo apt install qrencode
cd client
qrencode -t PNG -o output.png < client1.conf
scp username@remote_host:/path/to/remote/file /path/to/local/destination #do this on your machine and not the remote machine
```

How do you know its working:

To ensure that your WireGuard server is working properly and troubleshoot any issues, you can follow a few key steps. Here's a breakdown of how to verify that everything is functioning correctly and what troubleshooting steps to follow if problems arise:

---

1. Check the Status of the WireGuard Server

On the Server:

To check if the WireGuard server is running, use the following command:

```
sudo systemctl status wg-quick@server
```

- **Explanation**:
    - `sudo systemctl status wg-quick@server`: This will show you the status of the WireGuard service. If it's active and running, you should see "active (running)" in the output.
    - Make sure that "server" matches the name of your configuration file, without the `.conf` extension (e.g., if your config is `server.conf`, the name would be `server`).

Checking the WireGuard Interface:

You can also check if the WireGuard interface itself is up and running with:

```
sudo wg
```

- **Explanation**:
    - `sudo wg`: This command shows the status of the WireGuard interface and the connections. It will show details like peers, transferred data, and whether it is connected.

---

2. Verify the Server's Listening Port

Ensure that the WireGuard server is listening on the correct port (usually UDP 51820 by default). You can check this by running:

```
sudo netstat -uln | grep 51820
```

- **Explanation**:
  - `sudo netstat -uln`: This shows all UDP ports that are currently listening on your server.
  - `grep 51820`: Filters the output to check if port 51820 (or whichever port you've configured for WireGuard) is listening.

If the server is not listening on the expected port, check the server's WireGuard configuration file (`/etc/wireguard/server.conf`) for the correct port setting and make sure it's configured to listen on the proper interface.

---

3. Check Firewall Rules

Firewall misconfigurations are one of the most common causes of connectivity issues with WireGuard. Verify that your server's firewall allows traffic on the WireGuard port (default is UDP 51820).

- **Check Firewall Rules (UFW example)**:
  If you're using UFW (Uncomplicated Firewall), you can check the status with:

  ```
  sudo ufw status
  ```

  If needed, allow the WireGuard port:

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

- **Check iptables (if applicable)**:
  If you're using `iptables` for firewall management, ensure the appropriate rules are in place to allow UDP traffic on the WireGuard port.

  Example:

  ```
  sudo iptables -A INPUT -p udp --dport 51820 -j ACCEPT
  sudo iptables -A FORWARD -i wg0 -j ACCEPT
  sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
  ```

4. Verify Network and Routing Configuration

WireGuard relies on correct routing and NAT (Network Address Translation) to work properly, especially if it's functioning as a VPN. Ensure that:

- **IP forwarding is enabled**: This allows traffic to pass between different network interfaces.
    - To enable IP forwarding, run:

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

    - To make it permanent, edit `/etc/sysctl.conf` and add:

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

- **Check the routes**: Make sure your WireGuard server is properly routing traffic for the VPN. You can check the routing table with:

  ```
  ip route
  ```

  Ensure the WireGuard interface has a route for the VPN subnet.

5. Test the WireGuard Server's Connectivity

Once you've verified the status of the WireGuard server, you should check if clients can connect.

From a Client Machine:

- **Ping the Server's WireGuard IP**:
  Ensure that the client can reach the server's WireGuard interface IP address (usually something like `10.0.0.1` if you're using a private IP range).
    - Example:

      ```
      ping 10.0.0.1
      ```

- **Check for NAT or DNS issues**:
    - If you're trying to access external services through the VPN and not just the server, ensure that DNS is configured correctly on the client side. You can try pinging external IPs or websites to rule out DNS problems.

From the Server:

- **Ping the Client's WireGuard IP**:
  Test if the server can ping the client's WireGuard IP address (e.g., `10.0.0.2`). This helps ensure that the server can properly route traffic to the client.

---

6. Review Log Files for Errors

If things aren't working as expected, checking the logs can help pinpoint issues.

On the Server:

- **WireGuard logs**:
  Use `journalctl` to view logs related to the WireGuard service:

  ```
  sudo journalctl -u wg-quick@server
  ```

- **System logs**:
  You can also review the system logs for errors related to networking or firewall:

  ```
  sudo journalctl -xe
  ```

---

7. Verify Configuration on the Client Side

If the server looks fine but clients can't connect, check the following on the client:

- **Correct Client Configuration**: Ensure that the client's `client.conf` matches the server's configuration in terms of:
    - Correct server endpoint (IP/port)
    - Correct public key

- Proper routing (e.g., `AllowedIPs`)

- **Client Firewall**: Ensure that the client firewall is not blocking the WireGuard connection.
- **Client Logs**: Check the client-side logs for any errors or messages related to the WireGuard interface:

  ```
  sudo journalctl -u wg-quick@client
  ```

---

8. Restart Services

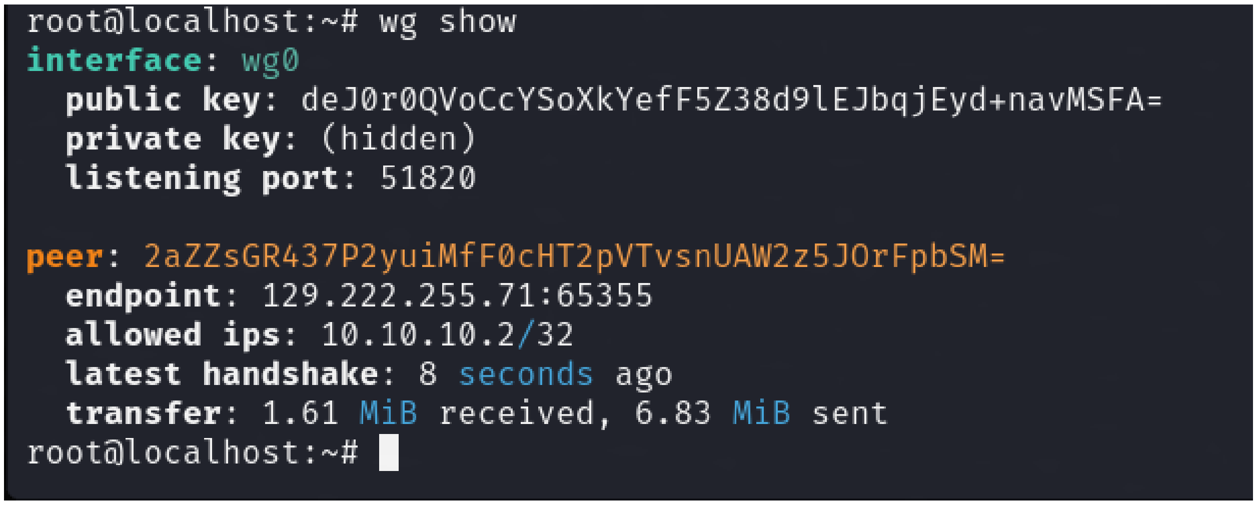
Sometimes, simply restarting the WireGuard service can resolve connection issues.

- **Restart the WireGuard service** on the server:

  ```
  sudo systemctl restart wg-quick@server
  ```

- **Restart the WireGuard client** on the client machine:

  ```
  sudo systemctl restart wg-quick@client
  ```



```
root@localhost:~# wg show
interface: wg0
  public key: deJ0r0QVoCcYSoXkYefF5Z38d9lEJbqjEyd+navMSFA=
  private key: (hidden)
  listening port: 51820

peer: 2aZZsGR437P2yuiMfF0cHT2pVTvsnUAW2z5JOrFpbSM=
  endpoint: 129.222.255.71:65355
  allowed ips: 10.10.10.2/32
  latest handshake: 8 seconds ago
  transfer: 1.61 MiB received, 6.83 MiB sent
root@localhost:~# 
```

This shows that another device is connected.

All done

Untitled

🛡️ WireGuard VPN Setup with Ansible (Step-by-Step Guide)