Building a Cloud-Hosted VPN

with **WireGuard** and **AWS**

Jeffrey Obi
May 2026

Contents

Building a Cloud-Hosted VPN with WireGuard and AWS

Introduction

This project involves setting up a fast, secure, and modern self-managed Virtual Private Network (VPN) using WireGuard on an Amazon Web Services (AWS) EC2 instance. This project demonstrates practical knowledge of cloud infrastructure, networking, Linux system administration, and modern security protocols.

Why This Project Matters

This project demonstrates the following capabilities:

1. **Navigating Cloud Environments:** Provisioning and managing resources in AWS (the industry leader).
2. **Understand Networking Concepts:** Applying concepts like IP addressing, routing, firewalls (Security Groups), and network address translation (NAT).
3. **Perform Linux System Administration:** Implementing the command line, install packages, edit configuration files, and manage system services.
4. **Implement Security First:** Deploying WireGuard, which is known for its lean, auditable codebase and state-of-the-art cryptography.

Project Prerequisites

- An active AWS account.
- An SSH client (PowerShell).

Technologies Used

- AWS (EC2, VPC, Security Groups), WireGuard, Linux (Ubuntu), Bash Networking (iptables).

Objective

- To establish a secure, encrypted tunnel for remote internet access, bypassing local network restrictions and securing traffic on untrusted Wi-Fi networks.

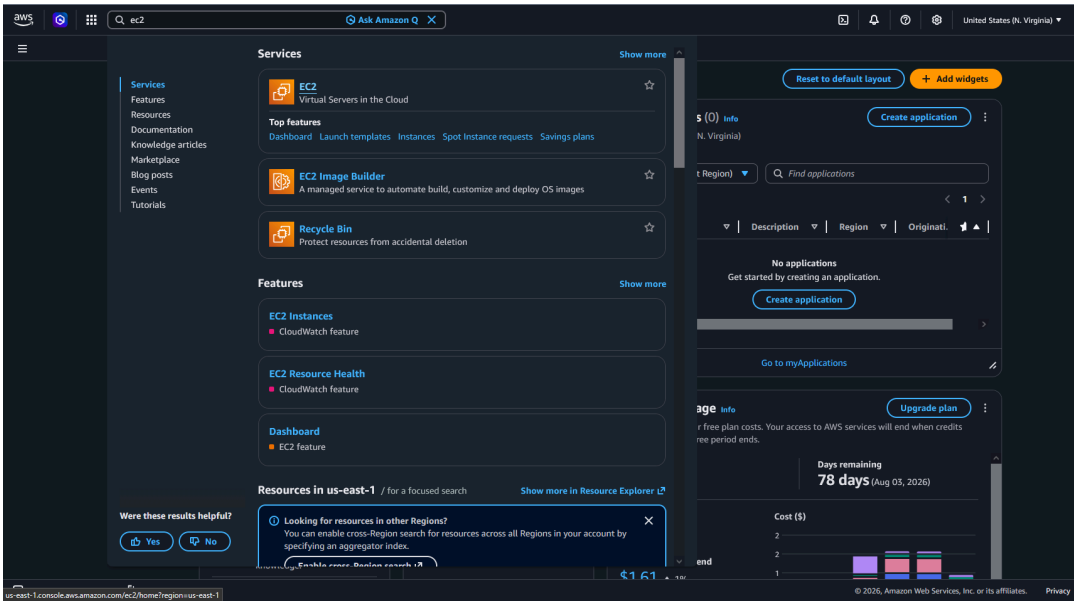
Procedure

Phase 1: Provisioning the AWS Infrastructure

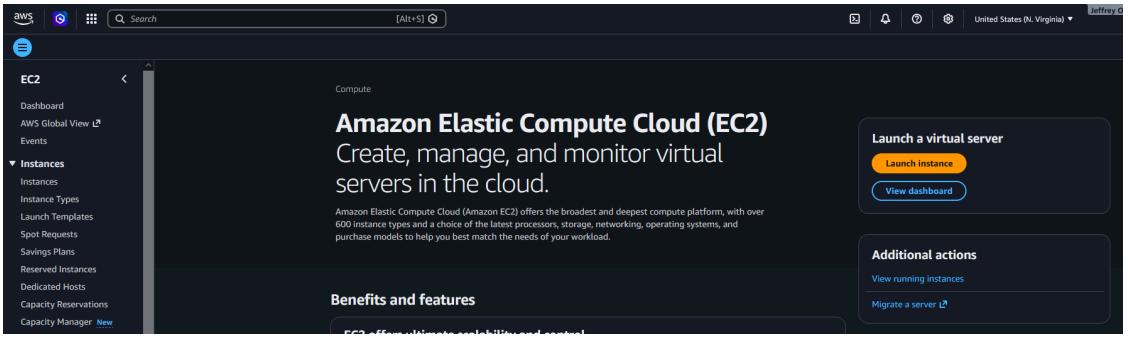
This phase involves setting up the "server" part of our VPN in the cloud.

Step 1.1: Launch an EC2 Instance

1. Log in to your AWS Management Console.
2. In the search bar at the top, type EC2 and select it.

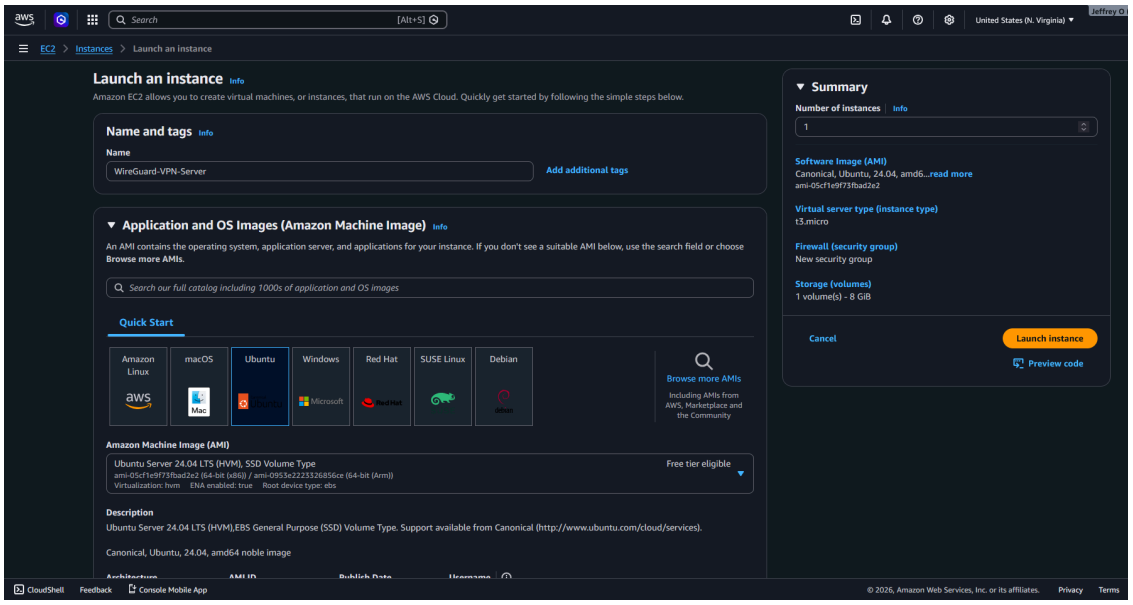


3. Ensure you are in a region geographically close to you (e.g., `us-east-1` for US East Coast) to minimize latency. You can change this in the top right corner.
4. Click the **Launch instance** button.

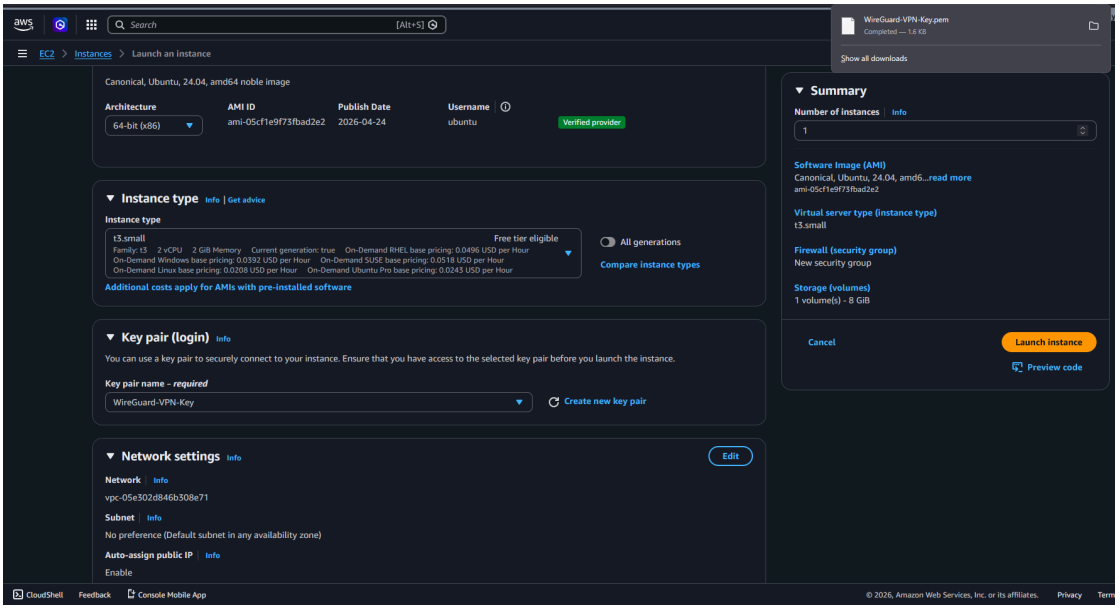


5. **Name and tags:** Give it a descriptive name (e.g., `WireGuard-VPN-Server`).

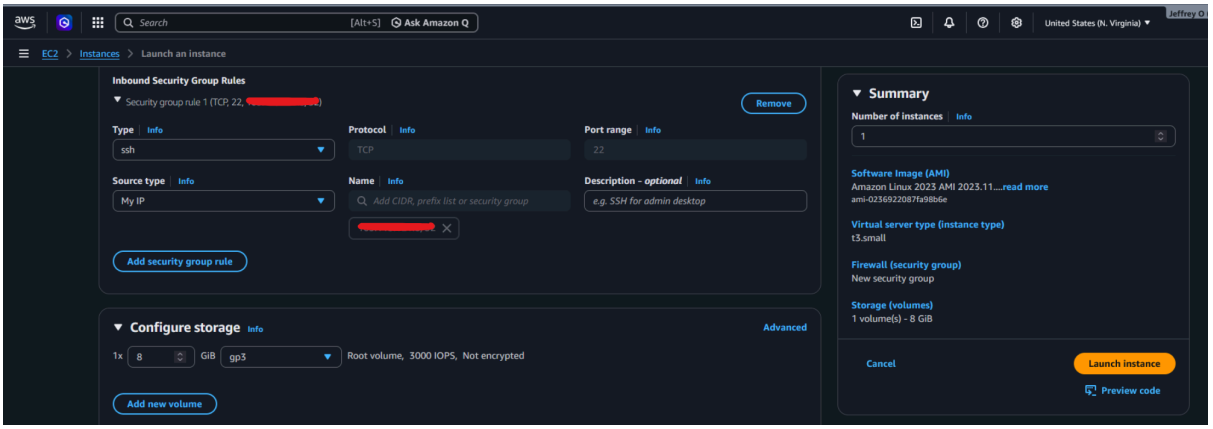
6. **Application and OS Images (Amazon Machine Image - AMI):**
    - Select **Ubuntu**.
    - Choose the **Ubuntu Server 24.04 LTS (HVM), SSD Volume Type**.

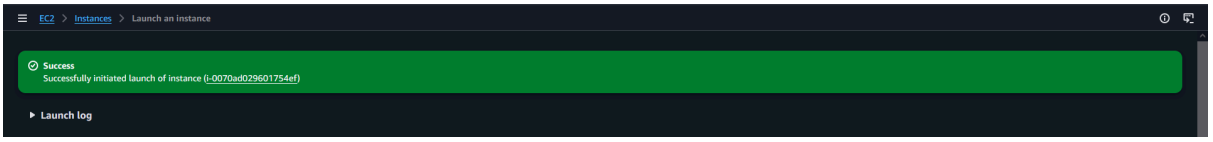


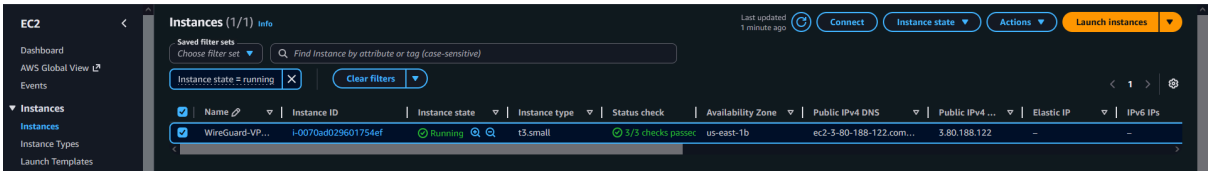
7. **Instance type:** Select `t3.small`.
8. **Key pair (login):**
    - Click **Create new key pair**.
    - Name it (e.g., `WireGuard-VPN-Key`).
    - Key pair type: `RSA`.
    - Private key file format: `.pem` (for OpenSSH/Mac/Linux).
    - Click **Create key pair**. *Save this file securely; you will need it to log in.*



9. **Network settings:**
    - Leave the VPC and Subnet as default.
    - Ensure "Auto-assign public IP" is **Enabled**.
    - Under "Firewall (security groups)", select **Create security group**.
    - Check "**Allow SSH traffic from**" and select **My IP** (my public IP & CIDR Notation: “xxx.xxx.xxx.xxx/xx”), applying the principle of least privilege.
10. **Configure storage: 8 GB (gp3)**.
11. Click **Launch instance** on the right side.



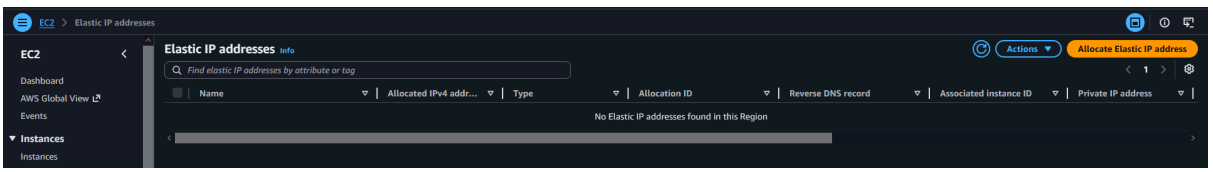




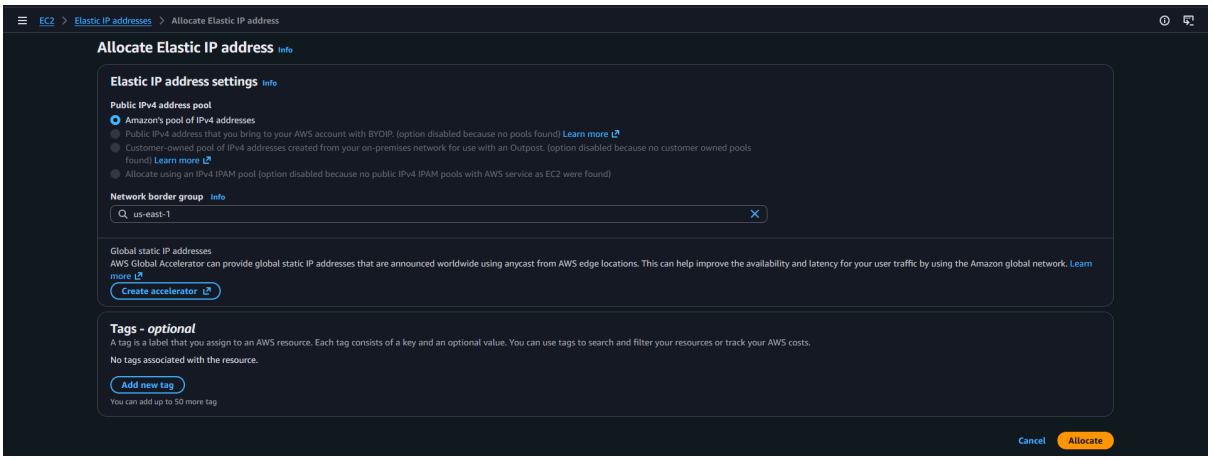
Step 1.2: Allocate an Elastic IP

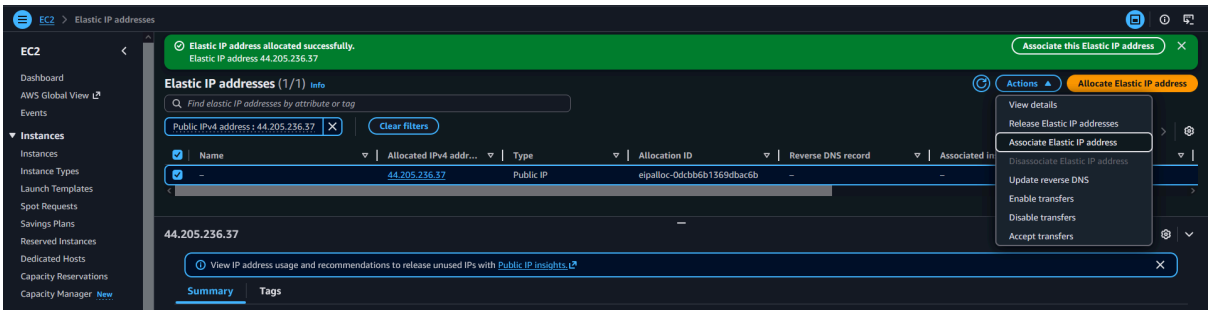
EC2 instances often change their public IP addresses when stopped and restarted. For a VPN server, you need a static address so your devices can always find it.

1. In the left sidebar of the EC2 dashboard, scroll down to **Network & Security** and click **Elastic IPs**.
2. Click the orange **Allocate Elastic IP address** button.

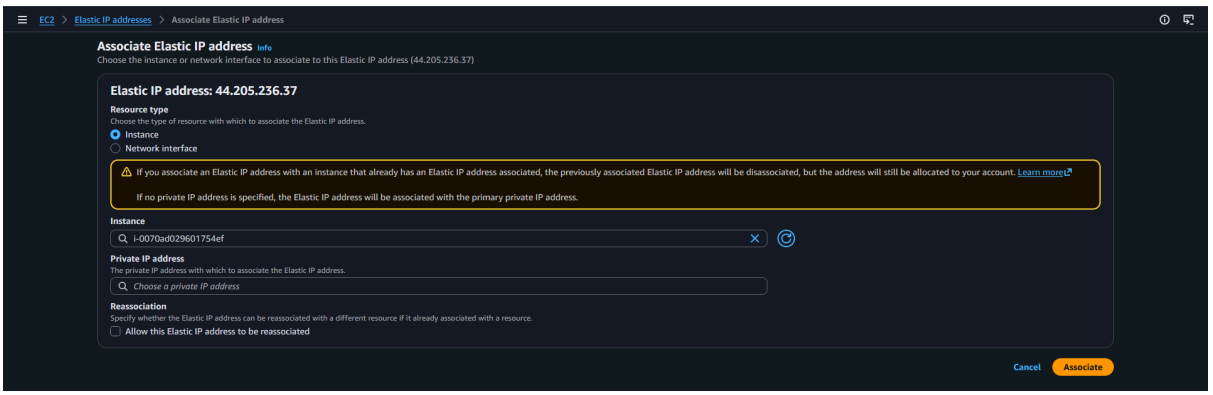


3. Leave settings default and click **Allocate**.
4. Select the newly created IP address from the list.
5. Click the **Actions** dropdown menu and select **Associate Elastic IP address**.





6. Resource type: `Instance`.
7. Instance: Click the search box and select your `WireGuard-VPN-Server` instance.
8. Click **Associate**.

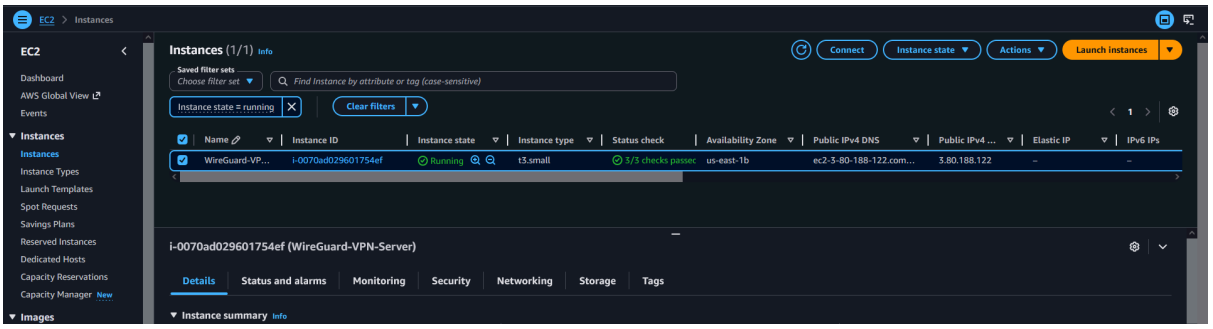


9. **Important Note:** Note down this Elastic IP address. This is the public address of your new VPN server.

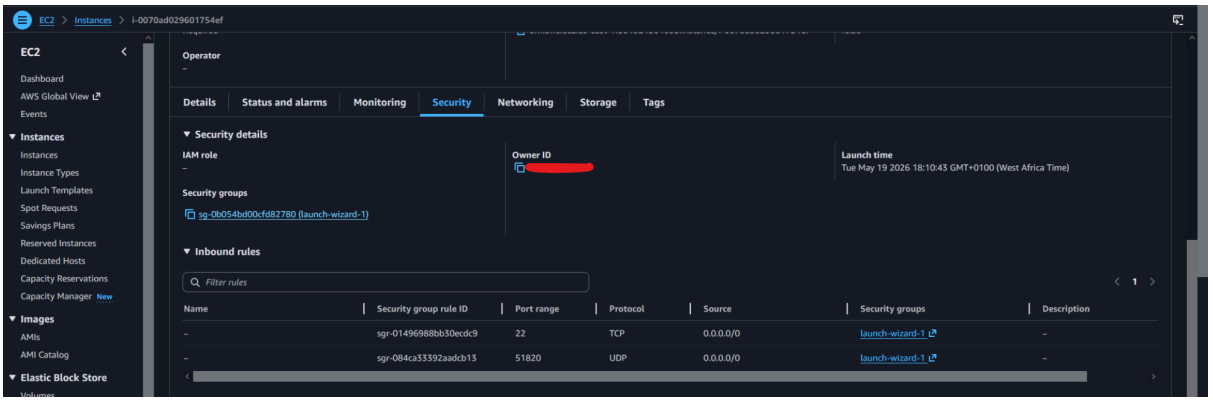
Step 1.3: Configure the Security Group (Firewall)

WireGuard needs a specific port open to receive incoming connections. The default port is **`51820 UDP`**.

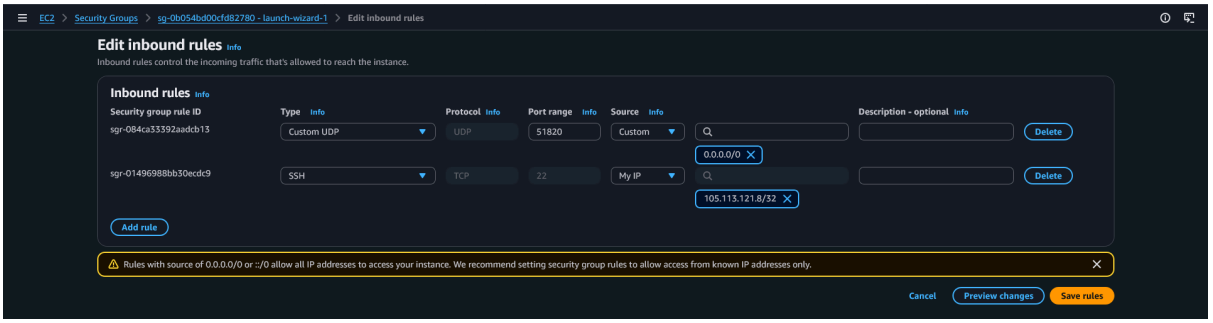
1. Go back to your `WireGuard-VPN-Server` in the EC2 dashboard (**Instances**).
2. Click on the instance ID to open its details.

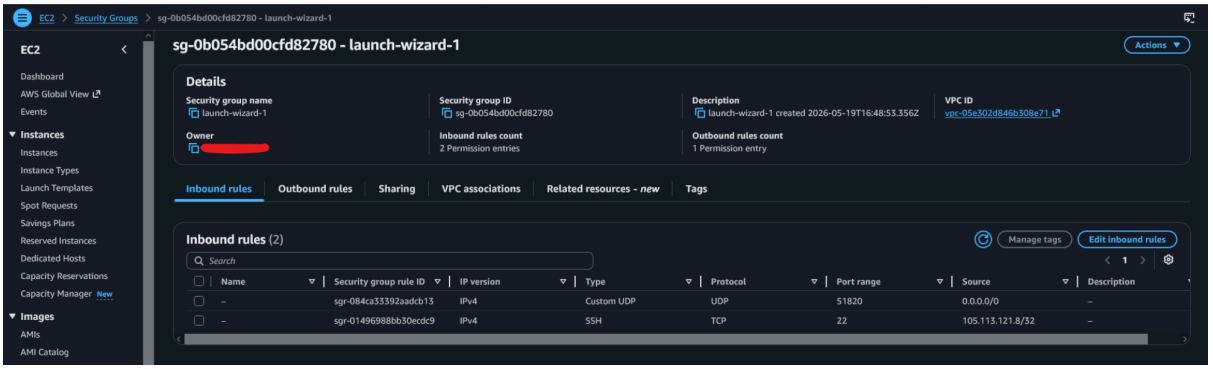


3. Click on the **Security** tab in the middle of the screen.



4. Click on the link under **Security groups** (it will look like `sg-0abc123...`).
5. Click the **Inbound rules** tab, then the **Edit inbound rules** button.
6. Click **Add rule**.
7. Type: `Custom UDP`
8. Port range: `51820`
9. Source: `Anywhere-IPv4` (0.0.0.0/0). *This allows you to connect from any network.*
10. Click **Save rules**.





Phase 2: Installing and Configuring WireGuard on the Server

Now we connect to our new server (via SSH) and set up the software.

Step 2.1: Connect via SSH

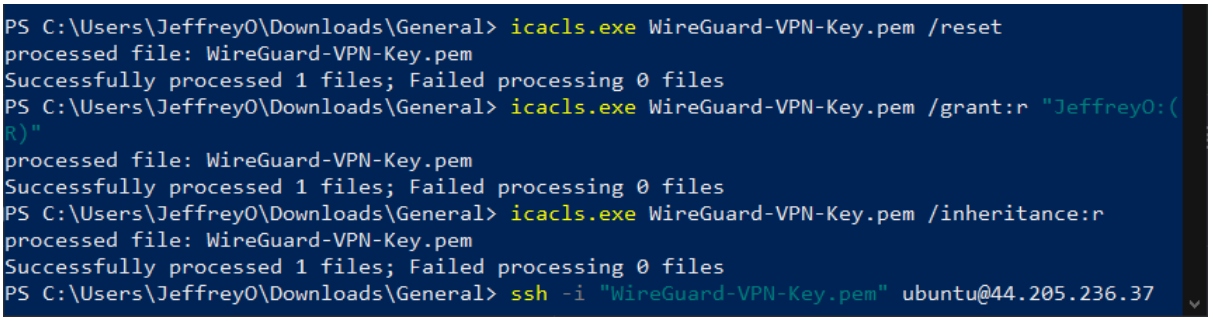
Open your terminal (**Windows PowerShell**). Use the SSH key you downloaded earlier.

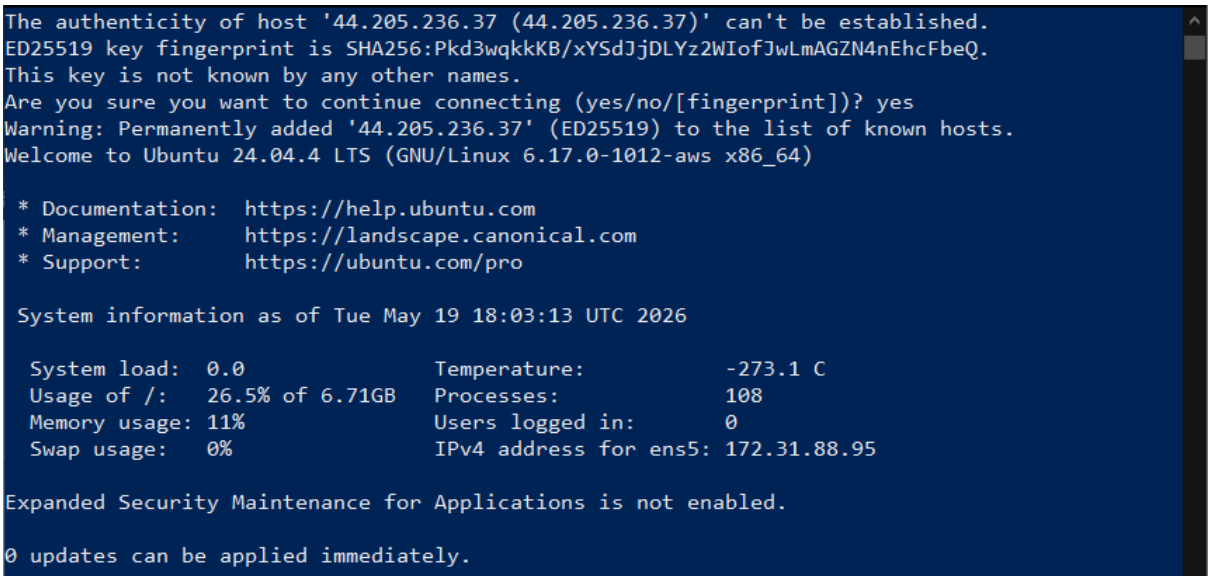
First ensure your key has the correct permissions (not accessible by others).

```
None
# Navigate to where you downloaded the key
cd Downloads

# Restrict permissions (Windows only)

# 1. Reset permissions and remove inherited access
icacls.exe WireGuard-VPN-Key.pem /reset

# 2. Grant only the current Windows user read-only access
icacls.exe WireGuard-VPN-Key.pem /grant:r "JeffreyO:(R)"
```

```
# 3. Disable all inherited permissions
icacls.exe WireGuard-VPN-Key.pem /inheritance:r

# Connect to the server
# Use the Elastic IP which was allocated in Step 1.2
ssh -i "WireGuard-VPN-Key.pem" ubuntu@44.205.236.37
```

Note: If prompted with "Are you sure you want to continue connecting?", type `yes` and hit enter.



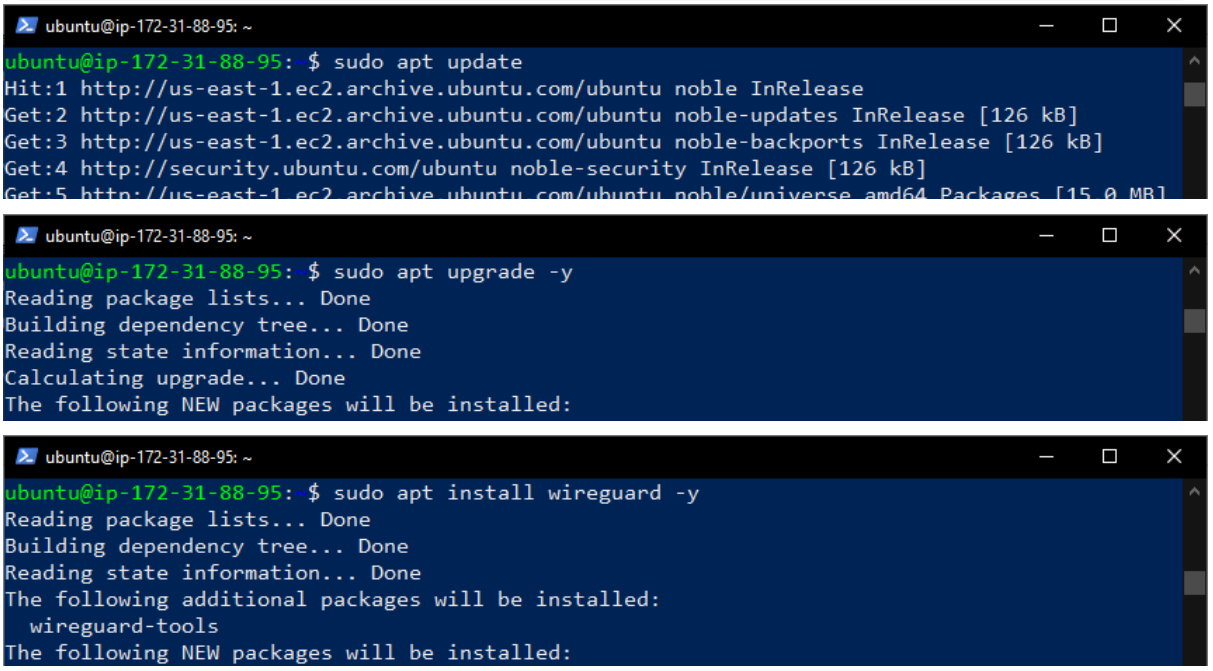


Step 2.2: Update the System and Install WireGuard

Once logged in, run these commands to update your server's package list and install WireGuard:

```
None
sudo apt update
sudo apt upgrade -y
```

```
sudo apt install wireguard -y
```



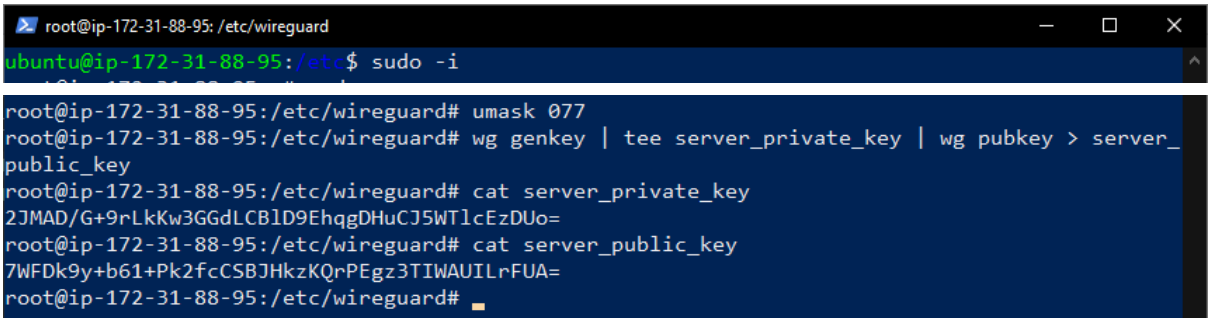
Step 2.3: Generate Server Keys

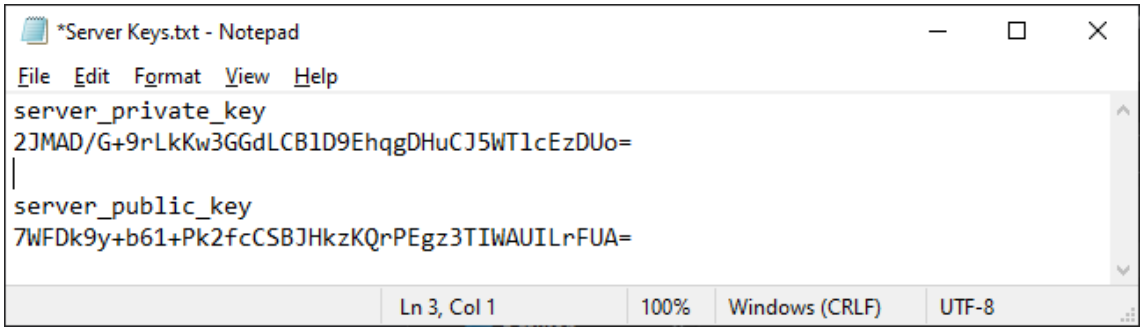
WireGuard uses public-key cryptography. Every device (peer) needs a private and a public key.

None

```
# Optional: Enable root access for seamless configuration
sudo -i

# Navigate to the WireGuard directory
cd /etc/wireguard

# Set secure permissions for the directory
umask 077

# Generate the server's private and public keys
wg genkey | tee server_private_key | wg pubkey >
server_public_key

# View the keys (you will need the private key for the config
file)
cat server_private_key
cat server_public_key
```

Note down the `server_private_key` value.

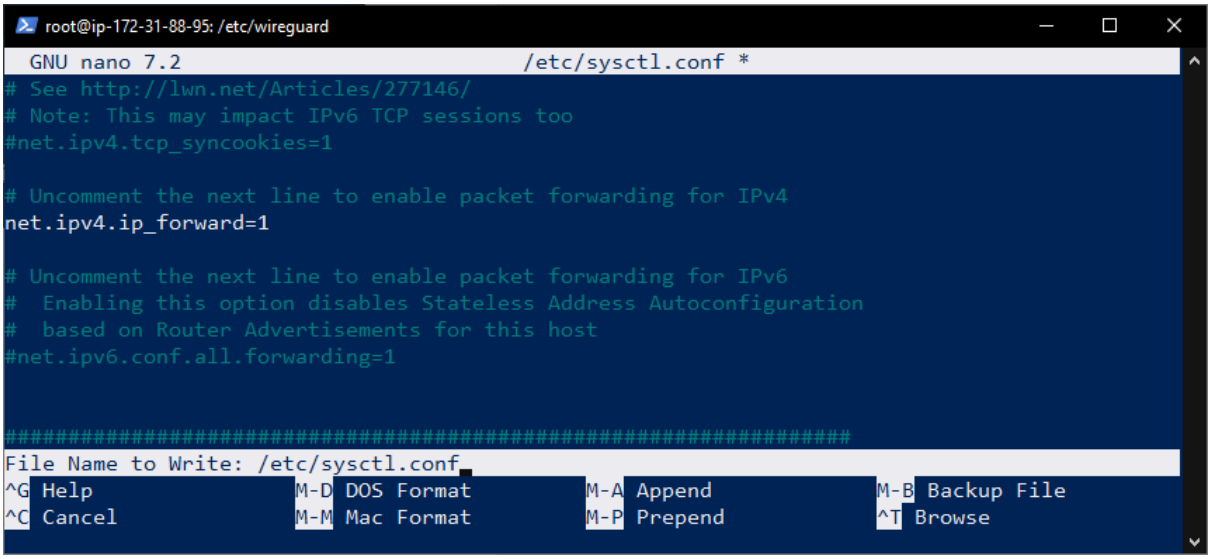




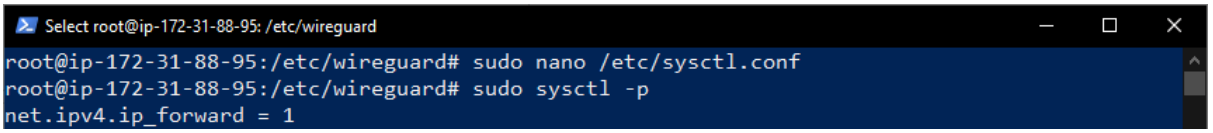
Step 2.4: Enable IP Forwarding

To allow the VPN server to route traffic from your connected devices out to the internet, we must enable IP forwarding in the Linux kernel.

1. Open the sysctl configuration file:
   `sudo nano /etc/sysctl.conf`
2. Scroll down to find the line `#net.ipv4.ip_forward=1`. Remove the `#` at the beginning to uncomment it:
   `net.ipv4.ip_forward=`



3. Save and exit (Ctrl+O, Enter, Ctrl+X).
4. Apply the changes immediately:
   `sudo sysctl -p`

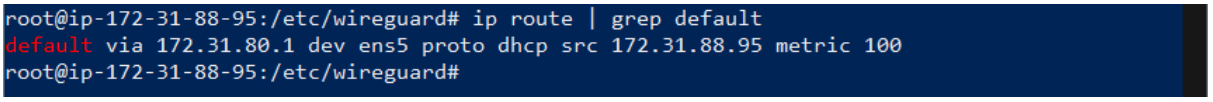


Step 2.5: Find Your Network Interface

We need to know the name of the main network interface connected to the internet to set up the firewall rules correctly.

```
None
ip route | grep default
```

Look for the text after `dev`. It's usually `eth0` or `ens5`. Note this down (let's assume it's `eth0` for the next steps).

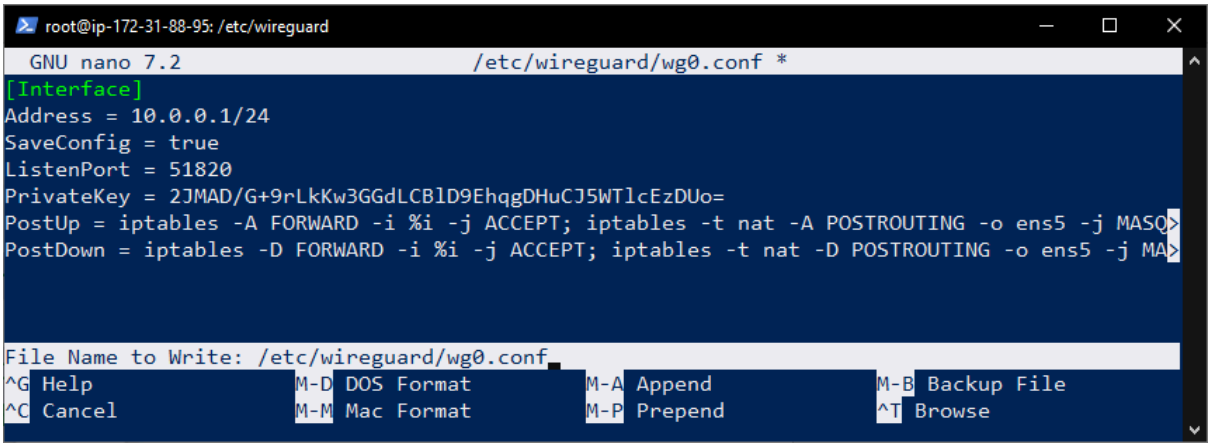


Step 2.6: Create the Server Configuration File

Now we build the `wg0.conf` file, which defines how the WireGuard interface operates.

1. Create the file:
   `sudo nano /etc/wireguard/wg0.conf`
2. Paste the following configuration, replacing the placeholders (including the server private key) carefully:

```
None
[Interface]
Address = 10.0.0.1/24
SaveConfig = true
ListenPort = 51820
PrivateKey = 2JMAD/G+9rLkKw3GGdLCBlD9EhqgDHuCJ5WTlcEzDUo=
PostUp = iptables -A FORWARD -i %i -j ACCEPT; iptables -t nat -A
POSTROUTING -o ens5 -j MASQUERADE
PostDown = iptables -D FORWARD -i %i -j ACCEPT; iptables -t nat
-D POSTROUTING -o ens5 -j MASQUERADE
```



3. Save and exit (Ctrl+O, Enter, Ctrl+X).

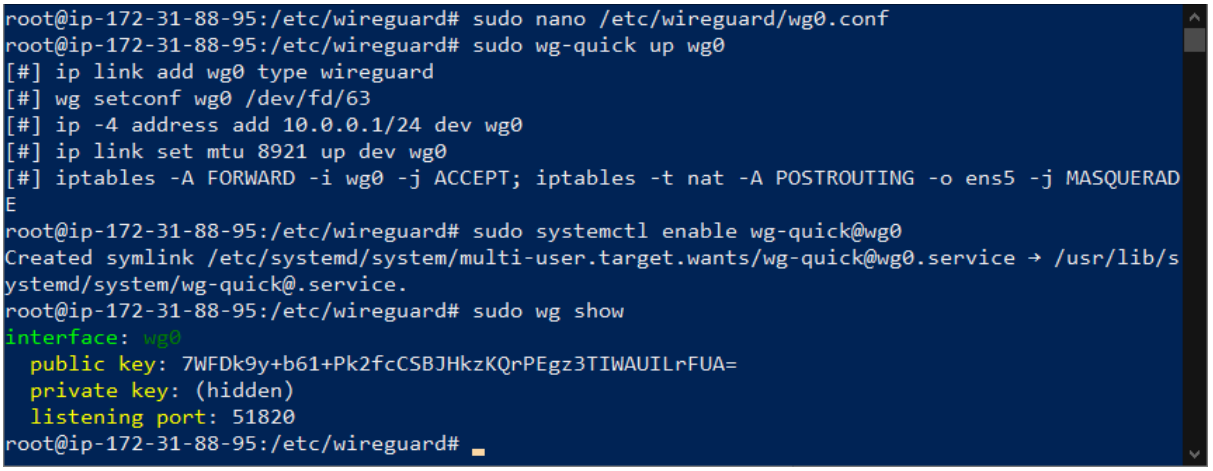
Step 2.7: Start the WireGuard Service

Let's bring up the interface and ensure it starts automatically when the server reboots.

None

```
# Start the interface now
sudo wg-quick up wg0

# Enable it to start on boot
sudo systemctl enable wg-quick@wg0

# Check the status
sudo wg show
```

If everything is correct, the **`sudo wg show`** command will display the active interface **`wg0`** and its public key.



Phase 3: Configuring the Windows Client Device

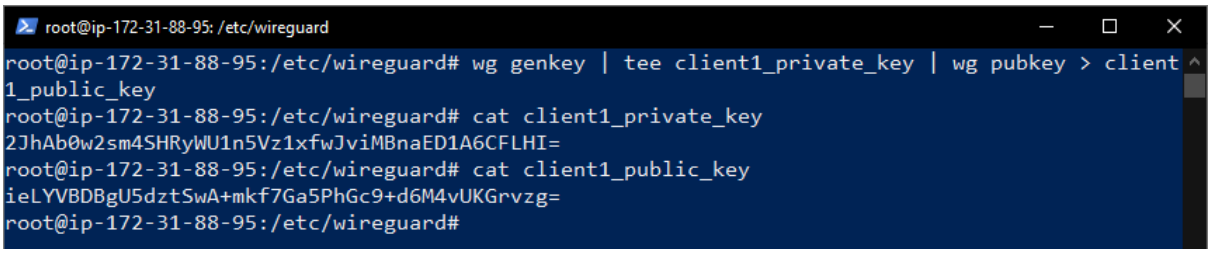
Create a configuration for the Windows laptop to connect *to* the VPN. We'll generate the keys on the server and create a config file for your device.

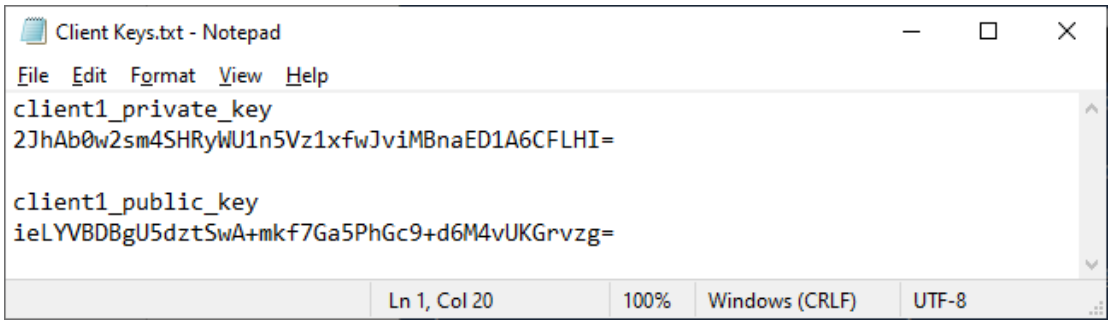
Step 3.1: Generate Client Keys (On the Server)

Stay logged into your AWS EC2 instance.

```
None
# Generate keys for "client1"
wg genkey | tee client1_private_key | wg pubkey >
client1_public_key

# View the keys
cat client1_private_key
cat client1_public_key
```

Note down BOTH the `client1_private_key` and `client1_public_key` values.





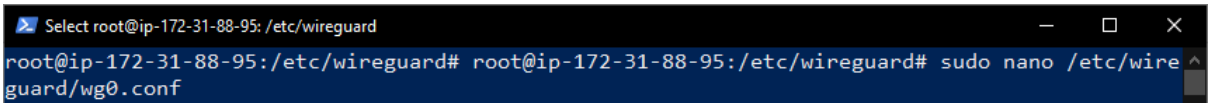
Step 3.2: Add the Client to the Server Configuration

We need to tell the server about this new authorized device.

1. Open the server config:
   `sudo nano /etc/wireguard/wg0.conf`
2. Add a `[Peer]` block at the very bottom containing the client public key:

None

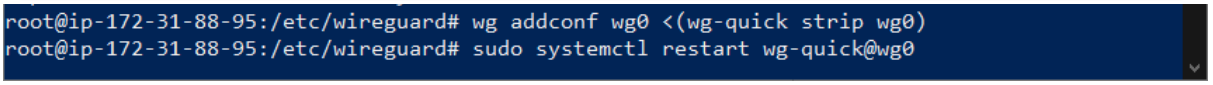
```
[Interface]
Address = 10.0.0.1/24
SaveConfig = true
ListenPort = 51820
PrivateKey = 2JMAD/G+9rLkKw3GGdLCBlD9EhqgDHuCJ5WTlcEzDUo=
PostUp = iptables -A FORWARD -i %i -j ACCEPT; iptables -t nat -A
POSTROUTING -o ens5 -j MASQUERADE
PostDown = iptables -D FORWARD -i %i -j ACCEPT; iptables -t nat
-D POSTROUTING -o ens5 -j MASQUERADE

[Peer]
PublicKey = ieLYVBDBgU5dztSwA+mkf7Ga5PhGc9+d6M4vUKGrvzg=
AllowedIPs = 10.0.0.2/32
```



3. Save and exit (Ctrl+O, Enter, Ctrl+X).

None

```
# Apply the changes to the running server:
sudo wg addconf wg0 <(wg-quick strip wg0)

# Or simply restart the service:
sudo systemctl restart wg-quick@wg0
```

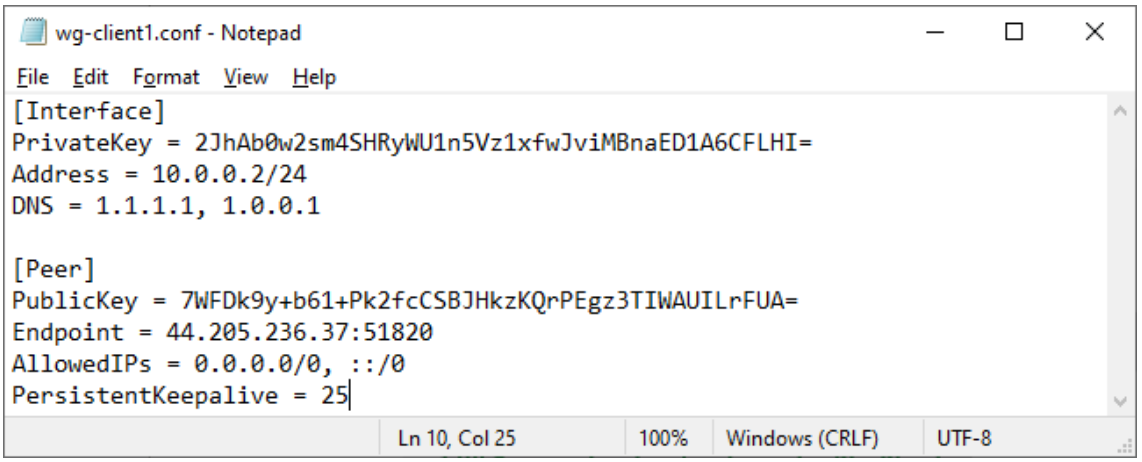


Step 3.3: Create the Client Configuration File

Now, we create the configuration file that will be imported into the WireGuard app on the Windows client. Create this file directly on the client.

1. On the Windows client, open a text editor (Notepad).
2. Create a file named **`wg-client1.conf`**
3. Paste the following, including the client private key and server public key:

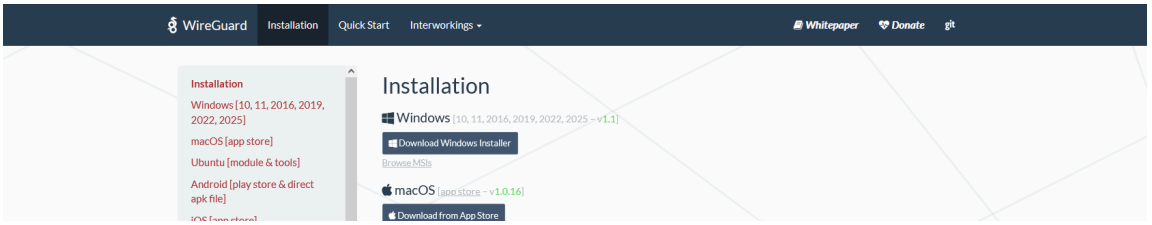
```
None
[Interface]
PrivateKey = 2JhAb0w2sm4SHRyWU1n5Vz1xfwJviMBnaED1A6CFLHI=
Address = 10.0.0.2/24
DNS = 1.1.1.1, 1.0.0.1

[Peer]
PublicKey = 7WFDk9y+b61+Pk2fcCSBJHkzKQrPEgz3TIWAUILrFUA=
Endpoint = 44.205.236.37:51820
AllowedIPs = 0.0.0.0/0, ::/0
PersistentKeepalive = 25
```



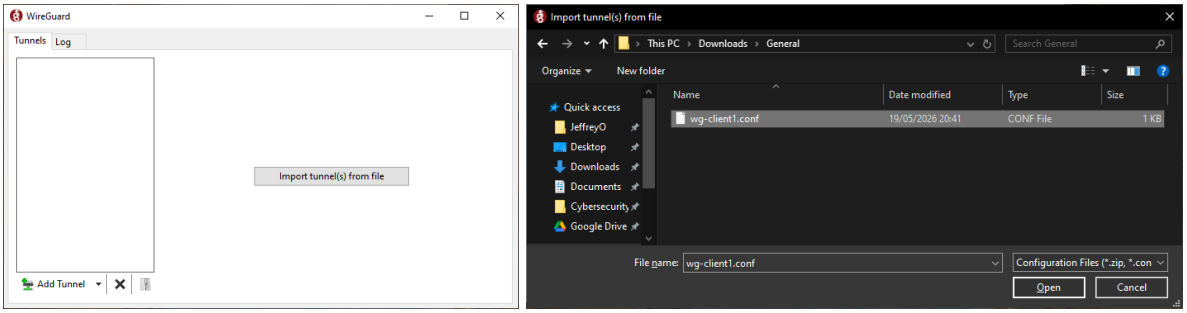
4. Save the file (Ctrl+O, Enter, Ctrl+X).

Step 3.4: Connect!

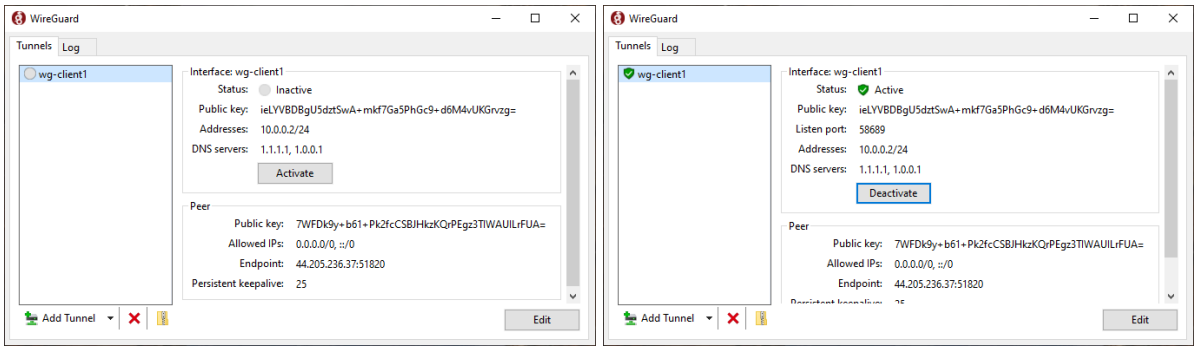
1. Download the official WireGuard client for Windows from **`wireguard.com`**.



2. Open the application.
3. Choose **Add Tunnel** or **Import tunnel(s) from file**.
4. Select the **`wg-client1.conf`** file you just created.

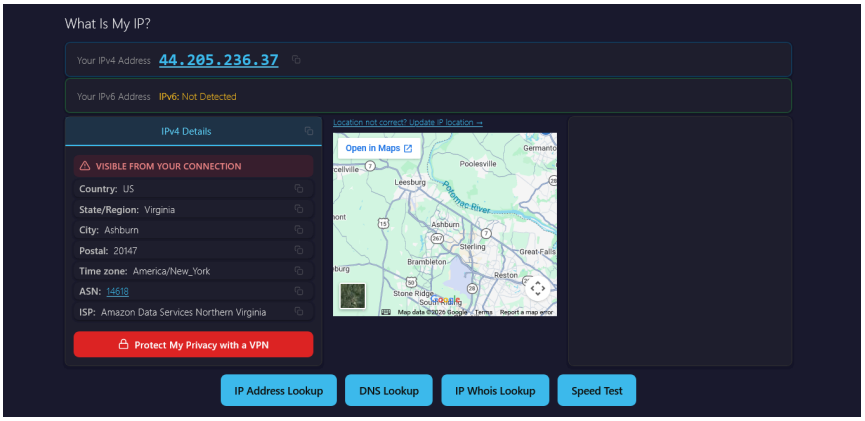


5. Click **Activate**.



Step 3.5: Test Connection!

1. **Web Test:** Connection established! Visit the site **`whatismyip.com`** on any browser on the Windows client, and it shows the Elastic IP of your AWS server, confirming the traffic is securely routed through the cloud.



2. **Terminal Test**: On the terminal (Powershell), run **`sudo wg show`** to confirm the handshake and data transfer details from this peer/client.

```
ubuntu@ip-172-31-88-95:~$ sudo -i
root@ip-172-31-88-95:~# sudo wg show
interface: wg0
  public key: 7WFDk9y+b61+Pk2fcCSBJHkzKQrPEgz3TIWAUILrFUA=
  private key: (hidden)
  listening port: 51820

peer: 8cwLkJ5K+AY7Ccp/hfNV+apjL16z/5vCkzQJDgcikW4=
  endpoint: 105.113.121.8:50297
  allowed ips: 10.0.0.2/32
  latest handshake: 51 seconds ago
  transfer: 1.51 MiB received, 6.59 MiB sent
```

Phase 4: Configuring the Android Client Device (Smartphone)

Create a configuration for the device you want to connect *to* the VPN. We'll generate the keys on the server, and create a config file on the Windows Client and transfer it to the Android device.

Step 4.1: Generate Client Keys (On the Server)

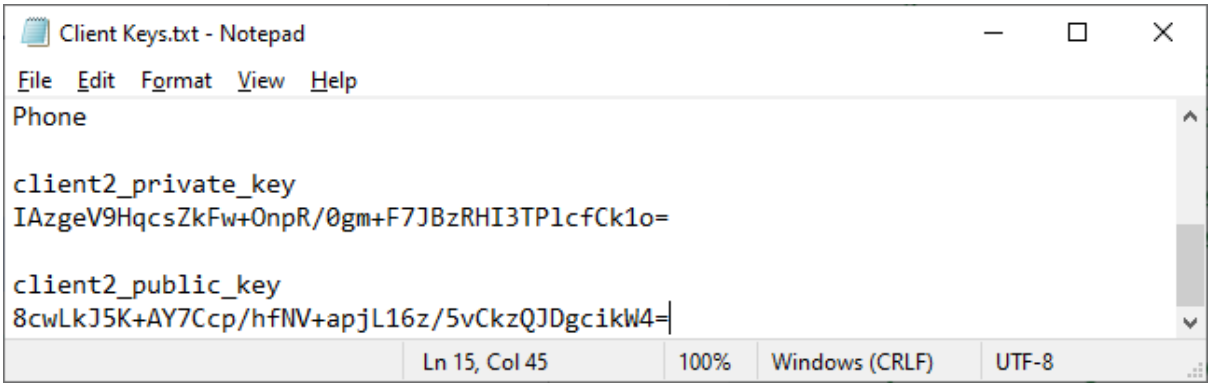
Stay logged into your AWS EC2 instance.

None

```
# Generate keys for "client1"
wg genkey | tee client2_private_key | wg pubkey >
client2_public_key

# View the keys
cat client2_private_key
cat client2_public_key
```

Note down BOTH the `client2_private_key` and `client2_public_key` values.

```
root@ip-172-31-88-95: /etc/wireguard
ubuntu@ip-172-31-88-95:~$ sudo -i
root@ip-172-31-88-95:~# cd /etc/wireguard
root@ip-172-31-88-95:/etc/wireguard# wg genkey | tee client1_private_key | wg pubkey > client
1_public_key
root@ip-172-31-88-95:/etc/wireguard# cat client1_private_key
IAzgeV9HqcsZkFw+OnpR/0gm+F7JBzRHI3TPlcfCk1o=
root@ip-172-31-88-95:/etc/wireguard# cat client1_public_key
8cwLkJ5K+AY7Ccp/hfNV+apjL16z/5vCkzQJDgcikW4=
root@ip-172-31-88-95:/etc/wireguard#
```



Step 4.2: Add the Client to the Server Configuration

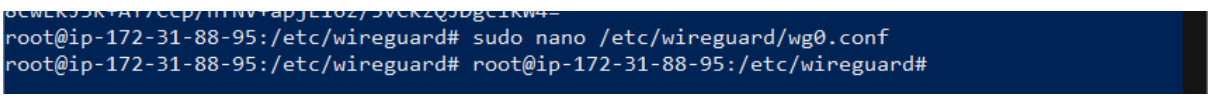
We need to tell the server about this new authorized device.

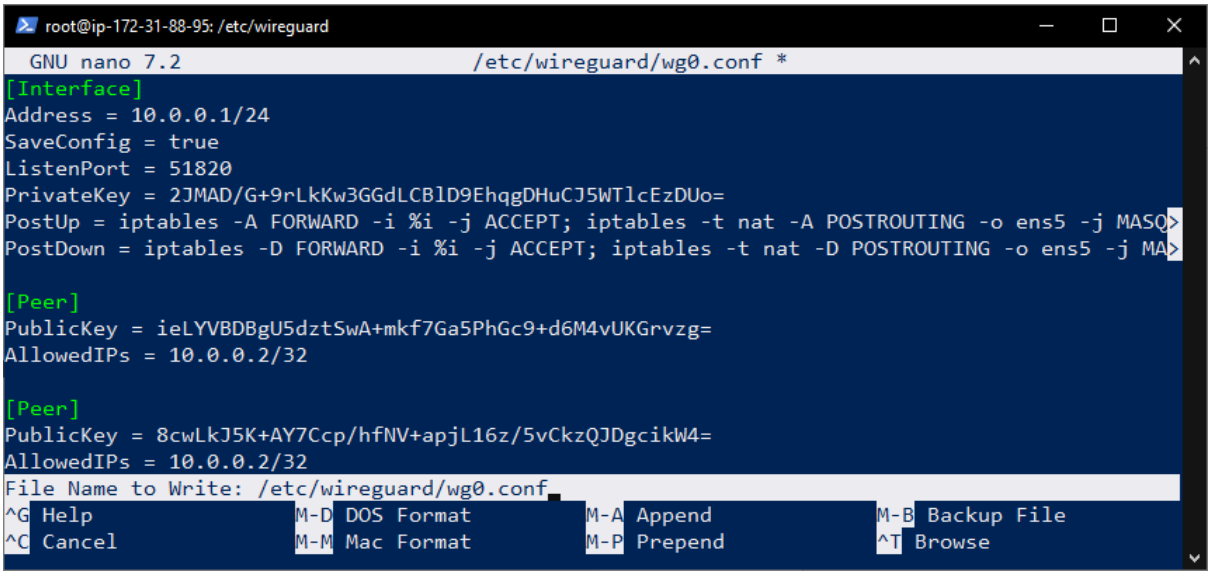
1. Open the server config:

```
None
sudo nano /etc/wireguard/wg0.conf
```

2. Add a **[Peer]** block at the very bottom containing the client public key:

```
None
[Interface]
Address = 10.0.0.1/24
SaveConfig = true
ListenPort = 51820
PrivateKey = 2JMAD/G+9rLkKw3GGdLCBlD9EhqgDHuCJ5WTlcEzDUo=
PostUp = iptables -A FORWARD -i %i -j ACCEPT; iptables -t nat -A
POSTROUTING -o ens5 -j MASQUERADE
PostDown = iptables -D FORWARD -i %i -j ACCEPT; iptables -t nat
-D POSTROUTING -o ens5 -j MASQUERADE

[Peer]
PublicKey = ieLYVBDBgU5dztSwA+mkf7Ga5PhGc9+d6M4vUKGrvzg=
AllowedIPs = 10.0.0.2/32

[Peer]
PublicKey = 8cwLkJ5K+AY7Ccp/hfNV+apjL16z/5vCkzQJDgcikW4=
AllowedIPs = 10.0.0.2/32
```





3. Save and exit (Ctrl+O, Enter, Ctrl+X).

```
None
# Apply the changes to the running server:
sudo wg addconf wg0 <(wg-quick strip wg0)

# Or simply restart the service:
sudo systemctl restart wg-quick@wg0
```

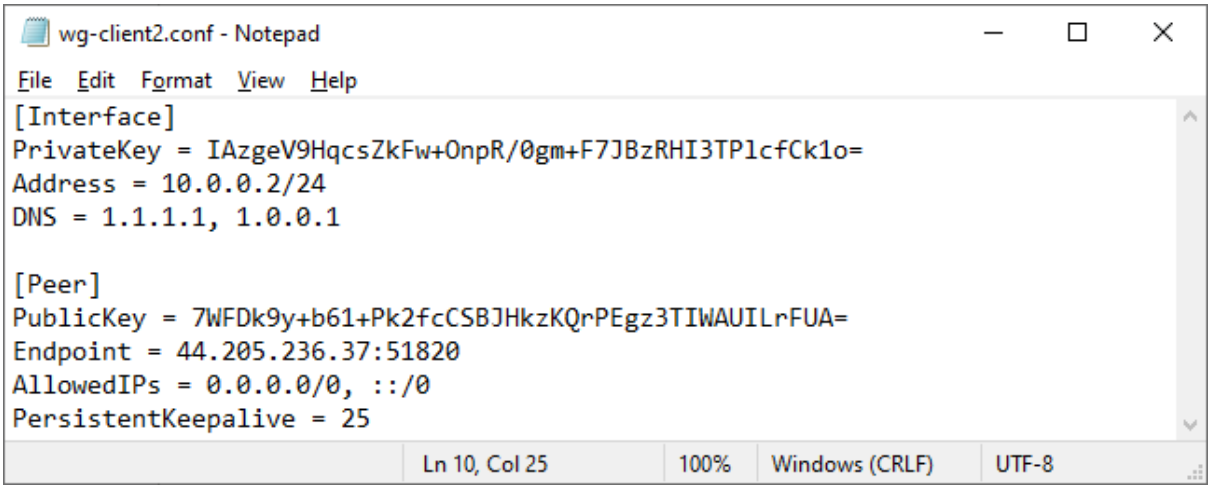
Step 4.3: Create the Client Configuration File

Now, we create the configuration file that will be imported into the WireGuard app on the Android device. Create this file on the Windows client to be later transferred to the smartphone..

1. On the Windows client, open a text editor (Notepad).
2. Create a file named **`wg-client2.conf`**.
3. Paste the following, including the client private key and server public key:

```
None
[Interface]
PrivateKey = IAzgeV9HqcsZkFw+OnpR/0gm+F7JBzRHI3TPlcfCk1o=
Address = 10.0.0.2/24
DNS = 1.1.1.1, 1.0.0.1

[Peer]
PublicKey = 7WFDk9y+b61+Pk2fcCSBJHkzKQrPEgz3TIWAUILrFUA=
Endpoint = 44.205.236.37:51820
```

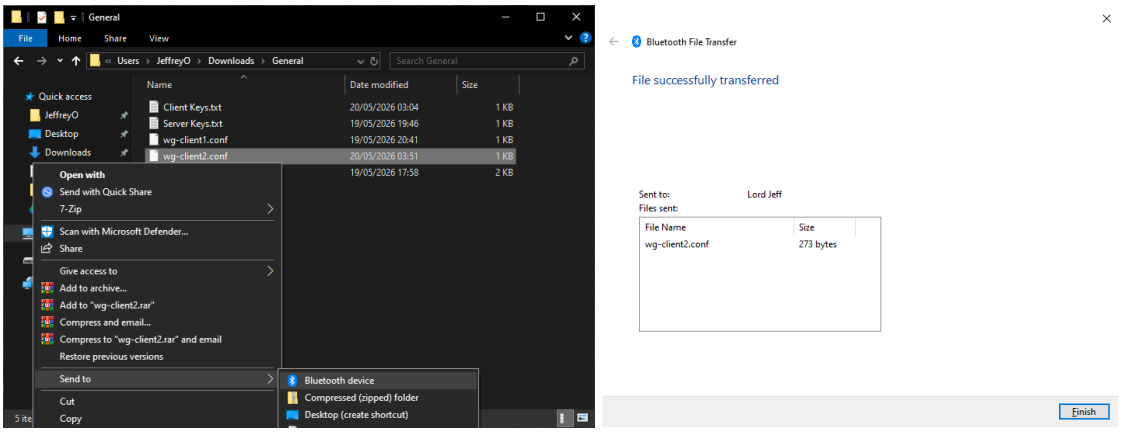
```
AllowedIPs = 0.0.0.0/0, ::/0
PersistentKeepalive = 25
```

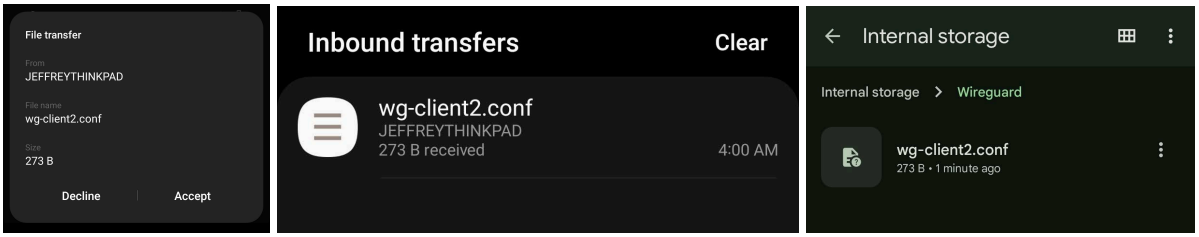


4. Save the file (Ctrl+O, Enter, Ctrl+X).

Step 4.4: Connect!

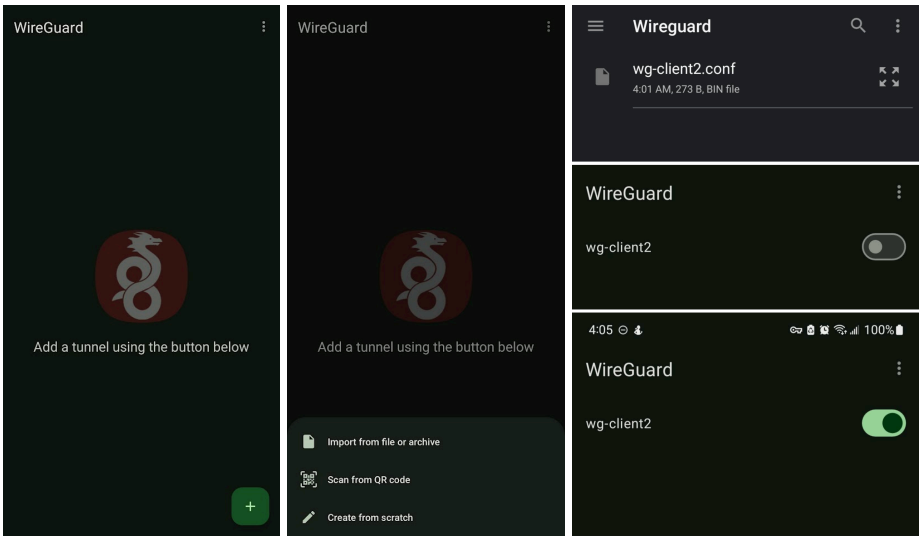
1. Transfer the file (`wg-client2.conf`) to the Android smartphone.





2. Download and install the official **WireGuard** app for **Android** from the **Play Store**.

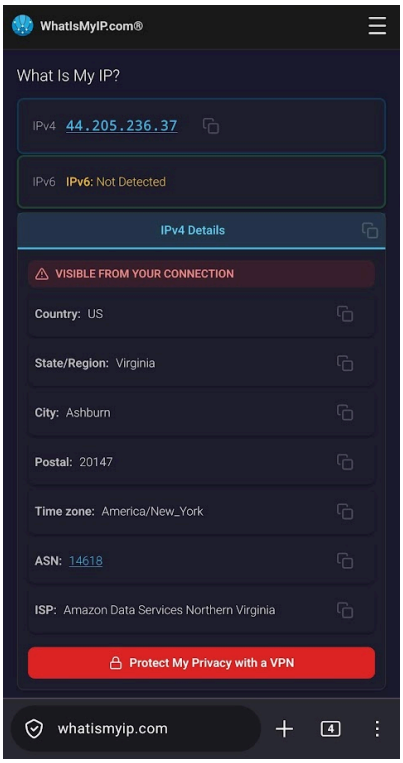
3. Open the application.
4. Tap the **+** button on the lower right of the screen to import the client configuration file.

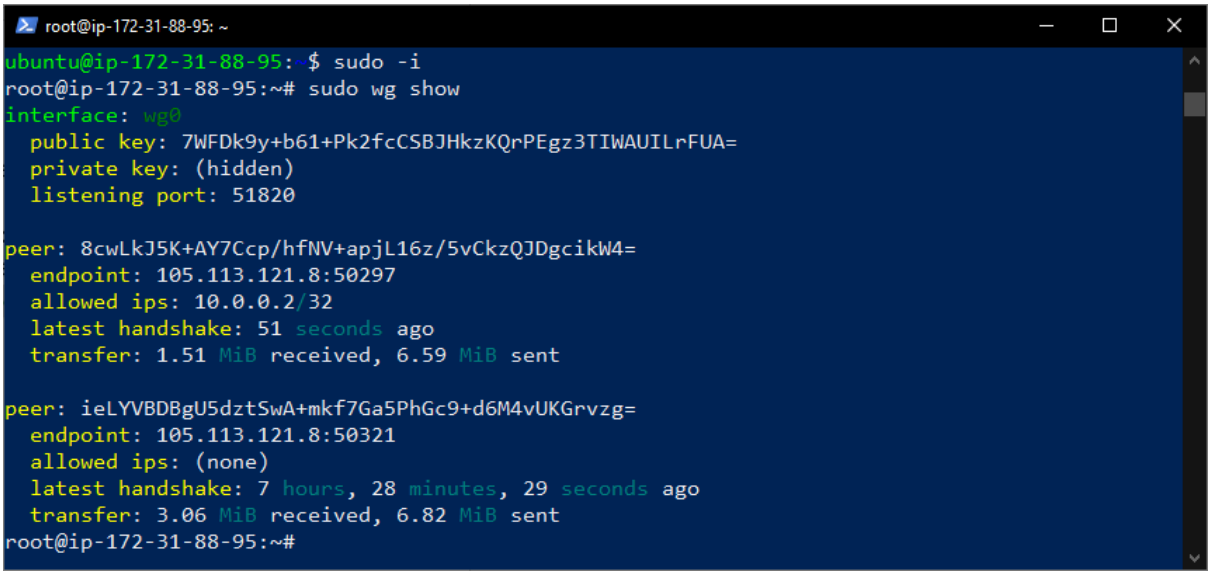


5. Select the `wg-client2.conf` file you just created.
6. The file is added to your main interface with a toggle next to it.
7. Tap the toggle to turn it on. This activates the connection.

Step 4.5: Test Connection!

1. **Web Test:** Connection established! Visit the site `whatismyip.com` on any browser on the Android client, and it shows the Elastic IP of your AWS server, confirming the traffic is securely routed through the cloud.
2. **Terminal Test**: On the Powershell terminal (or on **Termux** for Android, after establishing a secure shell connection via `ssh`), run `sudo wg show` to confirm the handshake and data transfer details from this peer/client.





Challenges Overcome

- Configured *iptables* to ensure proper NAT routing between the WireGuard interface and the public internet.
- Applied the principle of least privilege by limiting VPN server access and administration privileges to my machine.

Future Improvements for VPN Hardening:

- **DNS Ad-Blocking:** Running a DNS filter (e.g., Pi-hole) on the *same* EC2 instance to block ads and trackers at the network level for all devices connected to the VPN.
- **Adding a Preshared Key (PSK):** Add an additional symmetric encryption key (a Preshared Key) on top of the standard public/private key pairs.
    - It demonstrates advanced knowledge of cryptography. PSKs are designed to offer post-quantum resistance, ensuring that even if an attacker records your traffic today and quantum computers break the standard public-key cryptography in the future, your traffic remains encrypted.
- **Change the Default SSH Port:** Move SSH from port 22 to a non-standard port (like 2222). While this is "security through obscurity," it drastically reduces the amount of background noise and automated bot scans hitting your server's logs.
- **Enforce IMDSv2 (Instance Metadata Service v2):** By default, AWS EC2 instances often allow IMDSv1, which can be vulnerable to Server-Side

Request Forgery (SSRF) attacks. Forcing your instance to use IMDSv2 shows a deep, specialized understanding of AWS cloud security.

- **Set up CloudWatch Billing Alarms:** While not strictly "hacking" security, setting a billing alarm protects against financial attacks. If someone discovers your VPN's public IP and launches a massive DDoS attack against it, AWS could bill you for the outbound bandwidth. An alarm shows you understand cloud cost management and risk mitigation.
- **Host-Based Firewall (UFW):** Right now, the AWS Security Group acts as your firewall. Installing UFW (Uncomplicated Firewall) on the Ubuntu machine itself ensures that even if someone misconfigures the AWS Security Group, the server OS still drops unauthorized traffic.