

VPN Server setup with AWS, Pihole, and PiVPN

Kayvon Karimi

July 21st, 2025

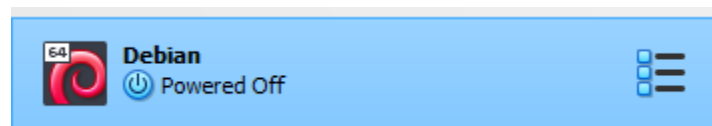
Introduction

This lab involved setting up a VPN server on Amazon Web Services (AWS) EC2, integrating Pi-hole for ad and tracker blocking, and PiVPN with WireGuard for secure connections. The purpose was to gain hands-on experience with cloud infrastructure, VPN protocols, and network privacy tools. I used a Debian VM in VirtualBox as the client instead of a Windows VM, adapting the instructions accordingly. This setup prepares for the final project by demonstrating VPN monitoring and privacy enhancement.

Step 1: Setting Up the Environment

VirtualBox and Debian were already installed prior to building the environment. I created a free AWS account at aws.amazon.com.

Debian VM Installed in VirtualBox



Signing up for AWS



Try AWS at no cost for up to 6 months

Start with USD \$100 in AWS credits, plus earn up to USD \$100 by completing various activities.



Sign up for AWS

Root user email address

Used for account recovery and as described in the [AWS Privacy Notice](#)

AWS account name

Choose a name for your account. You can change this name in your account settings after you sign up.

Verify email address

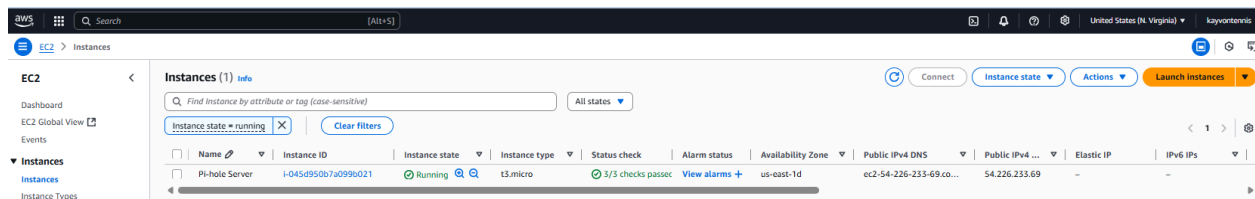
OR

Sign in to an existing AWS account

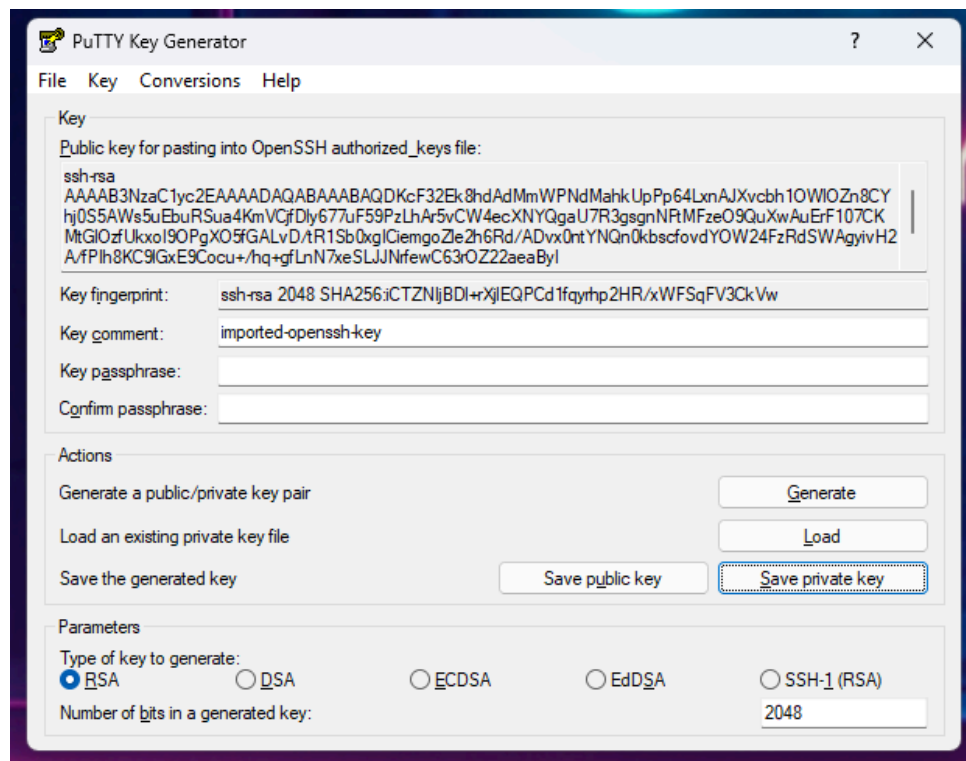
Step 2: Configuring AWS EC2 Instance

- Selected and launched a free tier Ubuntu Server AMI instance.
- created and downloaded a key pair (vpn-key.pem).
- converted it to .ppk for PuTTY using PuTTYgen.
- Configured Security Rules
- Connected via SSH from Windows using PuTTY.
- Created a Debian Client

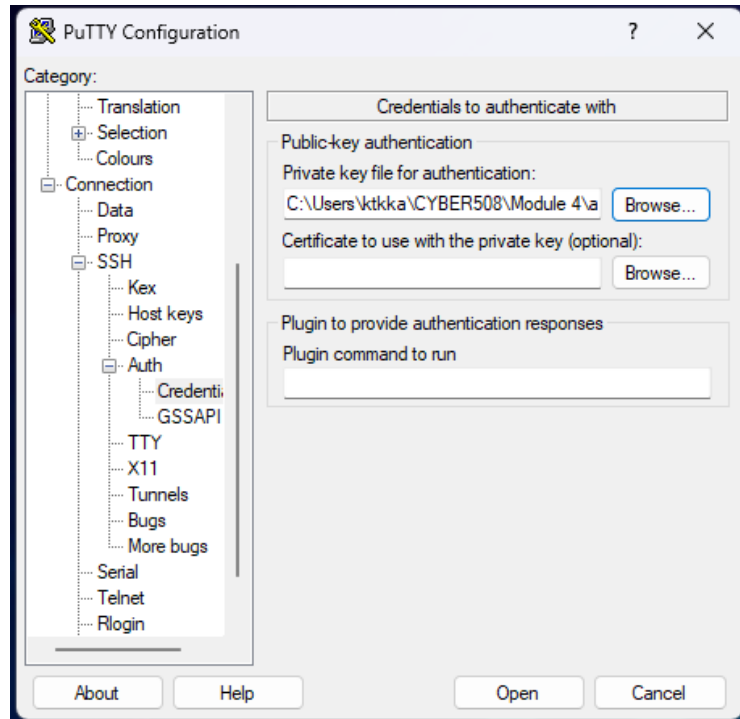
AWS instance launch (annotated: "EC2 instance running with public IP 54.226.233.69").



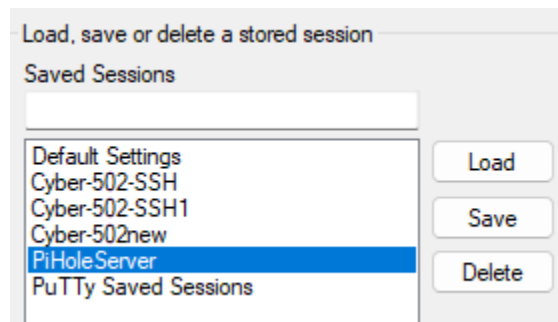
PuTTYgen used to generate a public/private key pair. I selected RSA with 2048 bits, generated the keys, and saved the private key for authentication



PuTTYgen configuration window loading the private key (converted to .ppk format) under the “Auth” section.



Saved “PiHole Server and opened SSH



Configuring Security Group Rules

sg-06166efdbacaafd03 - PiHole-SecurityGroup

Actions

Details

Security group name
PiHole-SecurityGroup

Security group ID
sg-06166efdbacaafd03

Description
Allows SSH, HTTP, HTTPS, and DNS access for Pi-hole server

VPC ID
vpc-09f83b55662b3ad404

Owner
863979974637

Inbound rules count
7 Permission entries

Outbound rules count
1 Permission entry

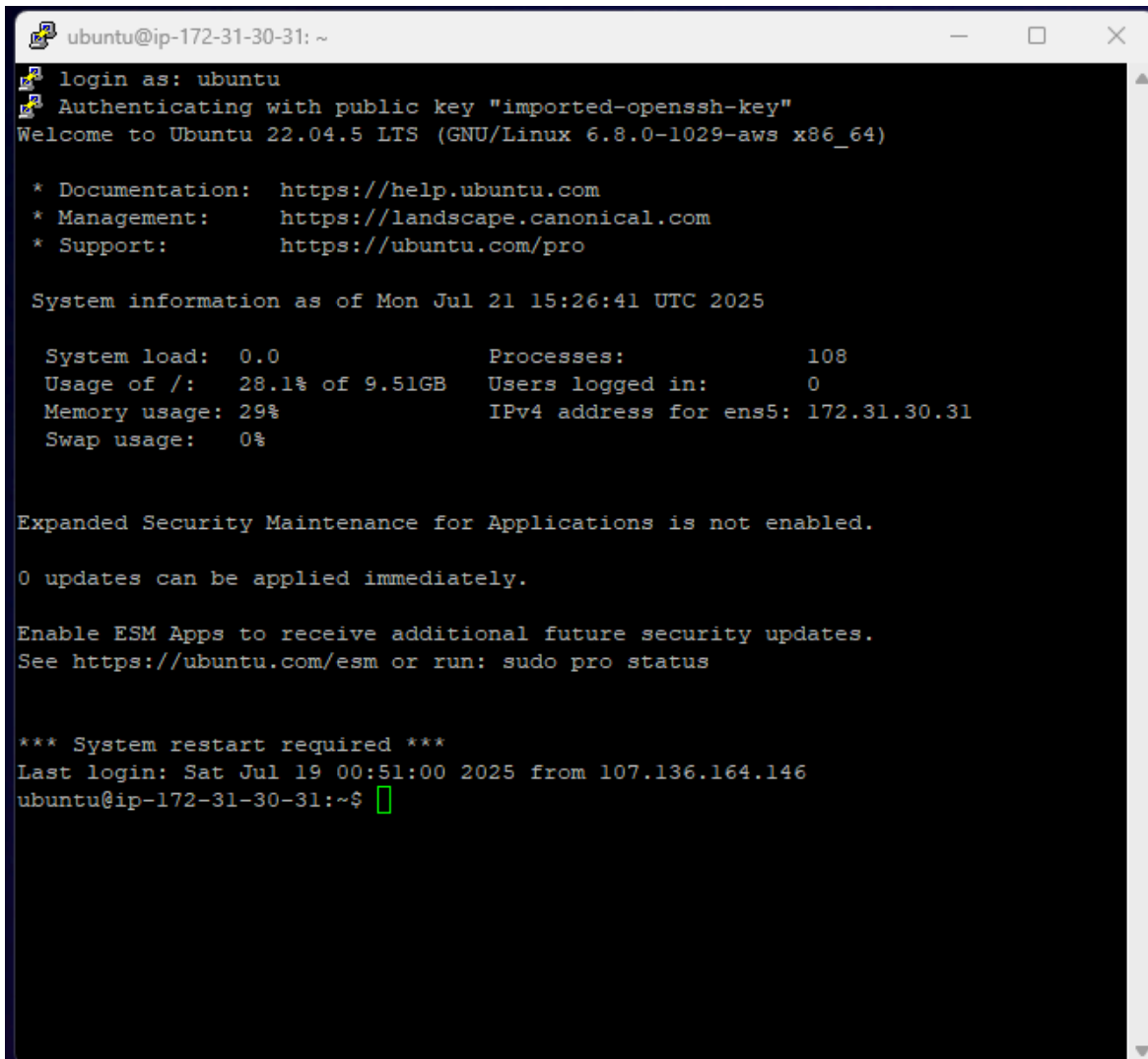
Inbound rules | Outbound rules | Sharing - new | VPC associations - new | Tags

Inbound rules (7)

Manage tags | Edit inbound rules

	Name	Security group rule ID	IP version	Type	Protocol	Port range	Source	Description
<input type="checkbox"/>	-	sgr-00a6aff2864838e59	IPv4	Custom UDP	UDP	51820	0.0.0.0/0	testing
<input type="checkbox"/>	-	sgr-034077d222adcac74c	IPv4	HTTP	TCP	80	172.65.32.248/32	LetsEncrypt renewal se...
<input type="checkbox"/>	-	sgr-0ba1466f79714334e	IPv4	SSH	TCP	22	107.136.164.146/32	Home LAN SSH access
<input type="checkbox"/>	-	sgr-0f01d83e2abd1b351	IPv4	DNS (TCP)	TCP	53	107.136.164.146/32	TCP DNS resolution
<input type="checkbox"/>	-	sgr-0a22875c647f346b5	IPv4	HTTPS	TCP	443	107.136.164.146/32	Home LAN HTTP access
<input type="checkbox"/>	-	sgr-08be26c08eb6afc77	IPv4	HTTP	TCP	80	107.136.164.146/32	Home LAN HTTP access
<input type="checkbox"/>	-	sgr-0bc70821caf5e403f	IPv4	DNS (UDP)	UDP	53	107.136.164.146/32	UDP DNS resolution

Entering SSH with login credential “ubuntu” in Putty SSH on Windows



```
ubuntu@ip-172-31-30-31: ~
login as: ubuntu
Authenticating with public key "imported-openssh-key"
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.8.0-1029-aws x86_64)

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

System information as of Mon Jul 21 15:26:41 UTC 2025

System load:  0.0           Processes:            108
Usage of /:   28.1% of 9.51GB Users logged in:          0
Memory usage: 29%          IPv4 address for ens5: 172.31.30.31
Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

*** System restart required ***
Last login: Sat Jul 19 00:51:00 2025 from 107.136.164.146
ubuntu@ip-172-31-30-31:~$
```

Adding Debian Client

```
::: Done! DebianClient.conf successfully created!
::: DebianClient.conf was copied to /home/ubuntu/configs for easytransfer.
::: Please use this profile only on one device and create additional
::: profiles for other devices. You can also use pivpn -qr
::: to generate a QR Code you can scan with the mobile app.
```

Step 3: Install Pi-hole and PiVPN with WireGuard

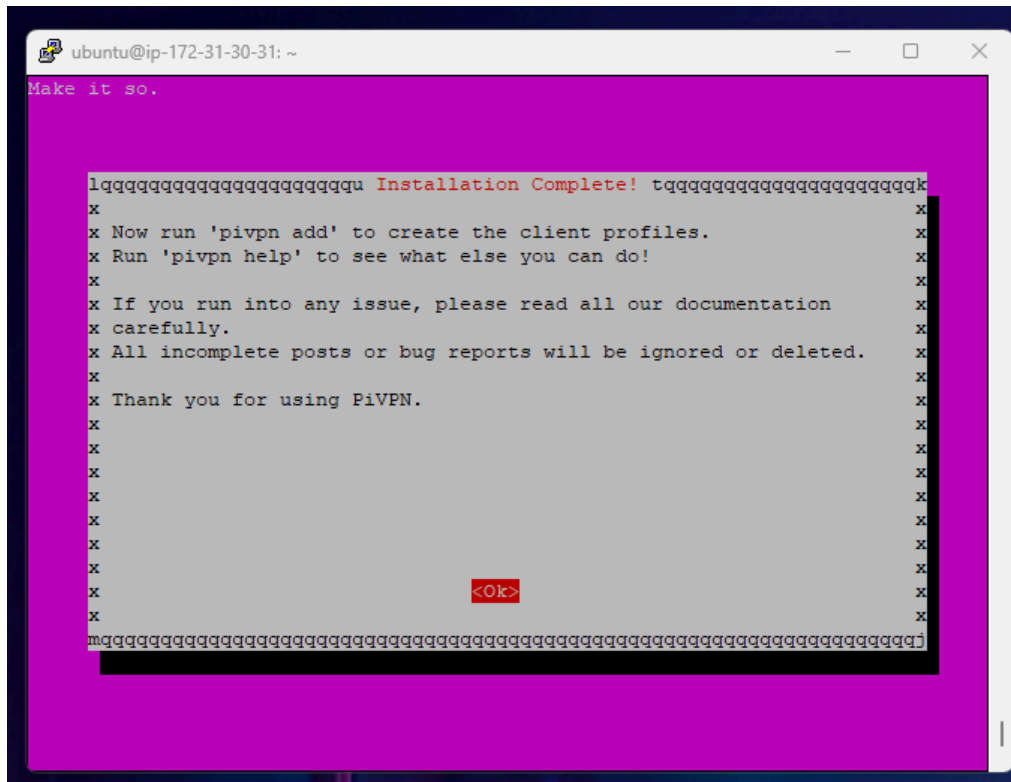
Updated the system (sudo apt update && sudo apt upgrade -y), installed lighttpd, and ran the Pi-hole installer (curl -sSL https://install.pi-hole.net | bash), selecting defaults and lighttpd. Then

installed PiVPN (curl -L https://install.pivpn.io | bash), choosing WireGuard, endpoint 54.226.233.69, and generated a client profile (pivpn add named DebianClient).

Pi-hole installation complete (annotated: "Pi-hole admin password and IP shown")

[illegible]

PiVPN setup (annotated: "WireGuard selected, client generated")

A screenshot of a terminal window titled 'ubuntu@ip-172-31-30-31: ~'. The terminal has a pink background. It displays the message 'Make it so.' followed by a large block of text enclosed in a grey box. The text inside the box says 'Installation Complete!' and provides instructions for using PiVPN, including running 'pivpn add' to create client profiles and 'pivpn help' for more options. It also mentions documentation and bug reports. At the bottom of the grey box is a red button labeled '<Ok>'. The terminal window has standard Ubuntu window controls (minimize, maximize, close) in the top right corner.

Step 4: Configure Debian VM to Connect to VPN

Installed WireGuard on Debian (sudo apt install wireguard -y). Copied DebianClient.conf from AWS, moved to /etc/wireguard/wg0.conf, added PersistentKeepalive = 25, and activated (sudo wg-quick up wg0). Verified with sudo wg show (showed latest handshake).

Ran cat /home/ubuntu/configs/DebianClient.conf on the AWS instance to display the file contents

```
ubuntu@ip-172-31-30-31:~$ cat DebianClient.conf
cat: DebianClient.conf: No such file or directory
ubuntu@ip-172-31-30-31:~$ cat /home/ubuntu/configs/DebianClient.conf
[Interface]
PrivateKey = uMsZkEVm0XY+dmCaOzePhWmASLMQSWenNzBsmf+1OV0=
Address = 10.108.167.2/24
DNS = 10.108.167.1

[Peer]
PublicKey = WDSGge/Yo4r5msRFcyqrQ02wiMfTwZIycpbuiE/Rzm0=
PresharedKey = 8SSDndCL4FieG+QdtecaPPZugxZ3KXTvbb1aKYA2Q1s=
Endpoint = 54.226.233.69:51820
AllowedIPs = 0.0.0.0/0, ::0/0
```


Copied the output, and pasted it into a new file on the Debian VM using nano
~/DebianClient.conf

```
Terminal - kayvon@DebianK: ~
File Edit View Terminal Tabs Help
GNU nano 7.2 /home/kayvon/DebianClient.conf *
[Interface]
PrivateKey = uMsZkEVm0XY+dmCa0zePhWmASLMQSWenNzBsmf+10Vo=
Address = 10.108.167.2/24
DNS = 10.108.167.1

[Peer]
PublicKey = WDSGge/Yo4r5msRFcyqrQ02wiMfTwZIycpbuiE/Rzm0=
PresharedKey = 8SSDndCL4FieG+QdtecaPPZuqxZ3KXTvbb1aKYA2Q1s=
Endpoint = 54.226.233.69:51820
AllowedIPs = 0.0.0.0/0, ::0/0
```

Ran 'sudo wg show' and see successful connection with data transfer")

```
kayvon@DebianK:~$ sudo wg show
interface: wg0
  public key: pkE+xyjMhzEY4CogNK9YI0pq0uY5G5kqeXiCBc8SwX4=
  private key: (hidden)
  listening port: 52400
  fwmark: 0xca6c

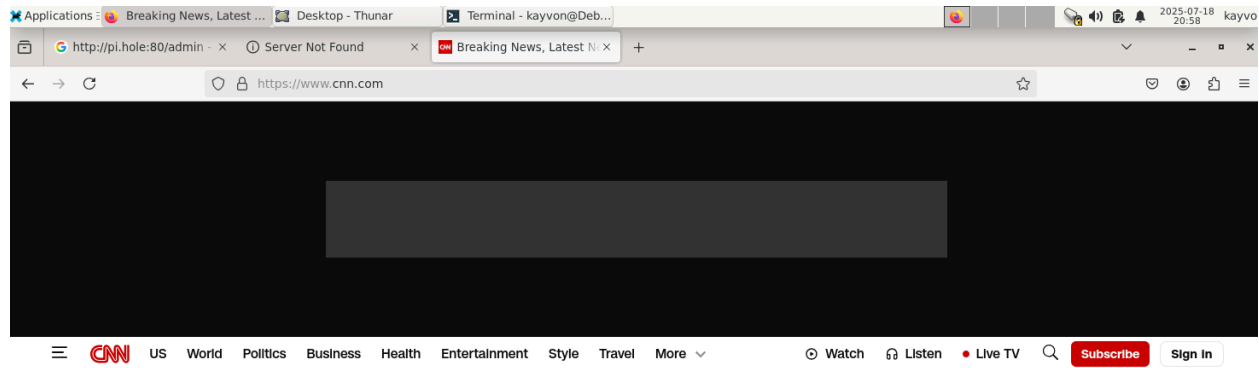
peer: WDSGge/Yo4r5msRFcyqrQ02wiMfTwZIycpbuiE/Rzm0=
  preshared key: (hidden)
  endpoint: 54.226.233.69:51820
  allowed ips: 0.0.0.0/0, ::/0
  latest handshake: 7 seconds ago
  transfer: 92 B received, 180 B sent
  persistent keepalive: every 25 seconds
kayvon@DebianK:~$ ping -c 4 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=116 time=80.2 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=116 time=77.7 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=116 time=79.9 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=116 time=78.7 ms

--- 8.8.8.8 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 77.730/79.118/80.186/0.984 ms
```

Ran 'curl ifconfig.me' and returned 54.226.233.69, my public IPv4 address

```
kayvon@DebianK:~$ curl ifconfig.me
54.226.233.69kayvon@DebianK:~$
```

Browsed [cnn.com](https://www.cnn.com) in Debian and ads were blocked by Pi-hole



Analysis

The VPN setup is effective for privacy, encrypting traffic via WireGuard's efficient protocol and routing through AWS (full tunnel). Pi-hole enhances this by blocking ads/trackers at DNS level, reducing data collection—e.g., [cnn.com](https://www.cnn.com) loaded without ads. Challenges included missing security group rules (UDP 51820) causing no handshake, resolved by adding rules. Learning: VPN tunneling secures public Wi-Fi, Pi-hole saves bandwidth/privacy. Overall, a robust setup for network security.

Challenges encountered included configuring AWS security groups for WireGuard (UDP port 51820), troubleshooting SSH key conversions for PuTTY, and ensuring the WireGuard handshake completed for a stable connection. Learning outcomes include understanding tunneling (full tunnel mode routes all traffic through the VPN for privacy), Pi-hole's role in DNS-based blocking, and AWS basics like EC2 instances and key pairs.