

AWS VPN Project Documentation

Collaborators: Nicolas Portilla Gomez, Joshua Shapiro, Andrew Enright

Date: 08/05/2025

Introduction: In this project we used the EC2 feature of AWS and the OpenVPN open-source protocol to route traffic securely to the AWS VPN we set up.

Project Details:

Before setting up our VPN, we needed to create a virtual server in AWS using the EC2 instances feature. We did this by going to the EC2 console and clicking “Launch Instances.”

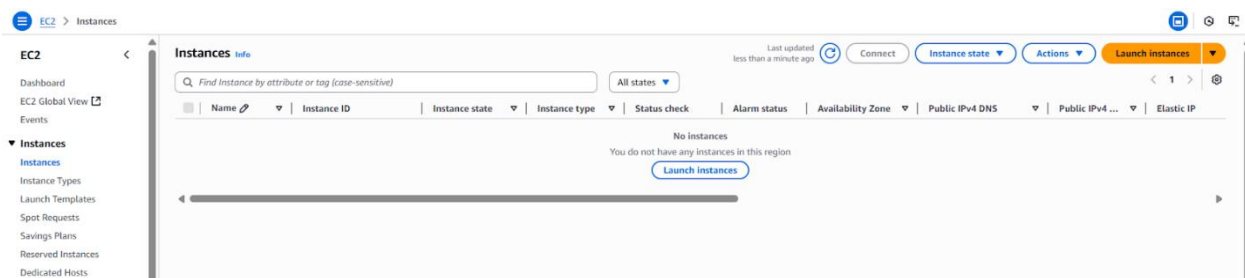


Figure 1: AWS EC2 Instances Dashboard

We were then asked to name our virtual server and give it an OS to run on. We named it “OpenVPN-Server” and chose Ubuntu as its stable and lightweight and works well with OpenVPN.

Name and tags [Info](#)

Name

OpenVPN-Server

Add additional tags

▼ Application and OS Images (Amazon Machine Image) [Info](#)

An AMI contains the operating system, application server, and applications for your instance. If you don't see a suitable AMI below, use the search field or choose [Browse more AMIs](#).

Q Search our full catalog including 1000s of application and OS images

Recents

Quick Start

Amazon Linux

aws

macOS

Mac

Ubuntu

ubuntu

Windows

Microsoft

Red Hat

Red Hat

SUSE Linux

SUSE

Debian

debian

Q

Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Ubuntu Server 22.04 LTS (HVM), SSD Volume Type

ami-0b05d988257befb8e (64-bit (x86)) / ami-0c96d7d095577f8de (64-bit (Arm))

Virtualization: hvm ENA enabled: true Root device type: ebs

Free tier eligible

Figure 2: Name and OS selection settings

We also made a key pair to securely log onto our EC2 server and chose the t2.micro instance type as it provides just enough CPU and memory for OpenVPN.

▼ Instance type [Info](#) | [Get advice](#)

Instance type

t2.micro

Free tier eligible

Family: t2 1 vCPU 1 GiB Memory Current generation: true

On-Demand Ubuntu Pro base pricing: 0.0134 USD per Hour On-Demand Linux base pricing: 0.0116 USD per Hour
On-Demand SUSE base pricing: 0.0116 USD per Hour On-Demand Windows base pricing: 0.0162 USD per Hour
On-Demand RHEL base pricing: 0.026 USD per Hour

All generations

Compare instance types

Additional costs apply for AMIs with pre-installed software

▼ Key pair (login) [Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - required

OpenVPNTest2

Create new key pair

Figure 3: Instance type and key pair selection

Network settings were also required so we used the default TCP protocol with port 22 which allowed us to securely connect to our server, and we added a custom UDP protocol with the port range 1194 which OpenVPN uses to make it so VPN clients can connect. Also since this was a personal project, our source was 0.0.0.0/0 which meant any traffic from any IP address in the world could access our VPN.

2

Inbound Security Group Rules

▼ Security group rule 1 (TCP, 22, 0.0.0.0/0) Remove

Type Info	Protocol Info	Port range Info
ssh	TCP	22
Source type Info	Source Info	Description - optional Info
Custom	<input type="text" value="Add CIDR, prefix list or security group"/> 0.0.0.0/0 X	e.g. SSH for admin desktop

▼ Security group rule 2 (UDP, 1194, 0.0.0.0/0) Remove

Type Info	Protocol Info	Port range Info
Custom UDP	UDP	1194
Source type Info	Source Info	Description - optional Info
Custom	<input type="text" value="Add CIDR, prefix list or security group"/> 0.0.0.0/0 X	e.g. SSH for admin desktop

Add security group rule

Figure 4: Network security group rules configuration

After configuring our EC2, we were ready to connect using a public IP address. A private IP address was not needed as that only works for connections inside AWS's private network (such as between two EC2s).

EC2 Instance Connect | Session Manager | SSH client | EC2 serial console

Instance ID: i-0dffa8d5eeb65371 (OpenVPN-Server)

☒ Connect using a Public IP
 Connect using a public IPv4 or IPv6 address

☐ Connect using a Private IP
 Connect using a private IP address and a VPC endpoint

☒ Public IPv4 address
 3.129.61.152

☐ IPv6 address
 -

Username
 Enter the username defined in the AMI used to launch the instance. If you didn't define a custom username, use the default username, ubuntu.

ⓘ Note: In most cases, the default username, ubuntu, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

Cancel Connect

Figure 5: EC2 instance connect screen

Using the curl -O bash command, we download a ready to use OpenVPN install script.

```
ubuntu@ip-172-31-30-161:~$ curl -O https://raw.githubusercontent.com/angristan/openvpn-install/master/openvpn-install.sh
% Total    % Received % Xferd Average Speed   Time    Time     Time  Current
           Dload  Upload   Total   Spent    Left   Speed
100 42167  100 42167    0     0  199k      0  --:--:-- --:--:-- --:--:-- 200k
```

Figure 6: OpenVPN install script command

We then gave the script permission to run and ran the script which walked us through a setup wizard.

```
ubuntu@ip-172-31-30-161:~$ chmod +x openvpn-install.sh
ubuntu@ip-172-31-30-161:~$ sudo ./openvpn-install.sh
Welcome to the OpenVPN installer!
```

Figure 7: permissions given and script ran

Most settings were left as defaults during the installation process and the client's name given was "client1."

```
The configuration file has been written to /home/ubuntu/client1.ovpn.
Download the .ovpn file and import it in your OpenVPN client.
```

Figure 8: completion of setup wizard for OpenVPN

When we connected to the EC2 instance, we connected to a browser-based terminal which meant direct file download wasn't supported, meaning we couldn't directly download our .ovpn file, so we worked around that by first viewing the contents of the file with the "cat" command. We then selected the entire output of the file and created a new notepad file on our local machine and then pasted the contents of the EC2 file onto the notepad.

```
ubuntu@ip-172-31-30-161:~$ cat client1.ovpn
```

Figure 9: Using the "cat" command to open the file

We didn't want to save the file as a .txt file however, so we changed the file type to a .ovpn file instead of a .txt file.



 OpenVPNTTest2.pem	8/5/2025 4:22 PM	PEM File	2 KB
 client1.ovpn	8/5/2025 4:45 PM	OVPN Profile	0 KB

Figure 10: .ovpn file in local directory

Once our .ovpn file was ready, we needed to download the client which would make connecting to the VPN possible, and that is exactly what we did.



Figure 11: OpenVPN client download screen

Once the client was downloaded, we uploaded our .ovpn file directly into the client.

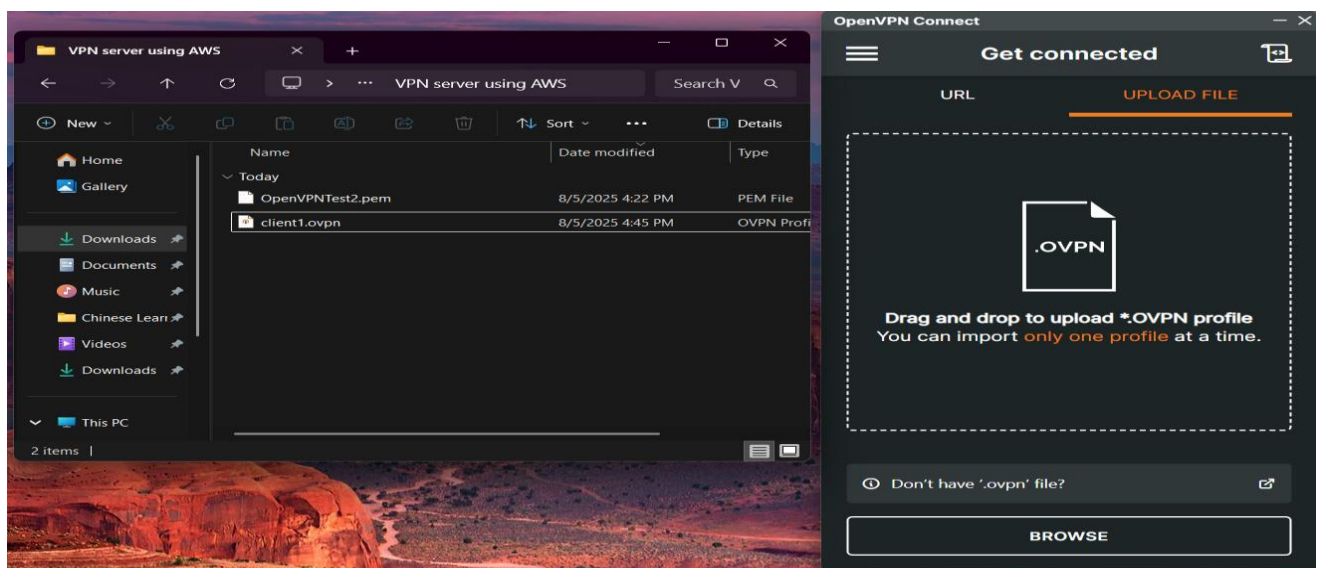


Figure 12: Importing .ovpn file from local machine to client

We were successfully able to connect, which meant the file worked, but we still needed to verify that the security settings we had put down were working properly and that we were able to properly route traffic.

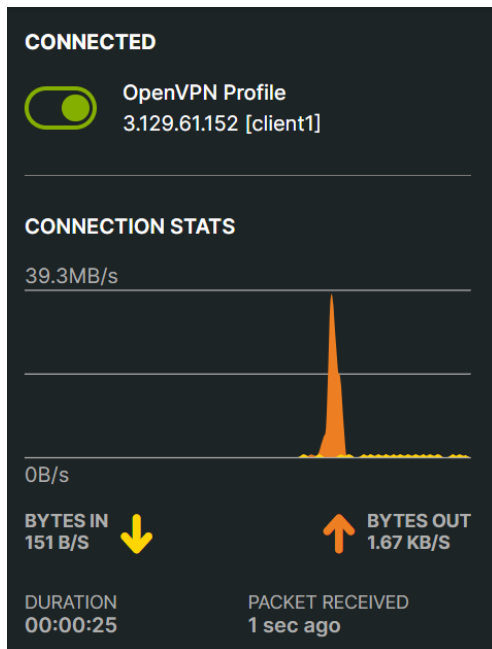


Figure 13: Successful connection to OpenVPN client using .ovpn file

The final step involved us going onto whatismyip.com and comparing what it showed on the website to what we had on our client, which we verified were the exact same values.

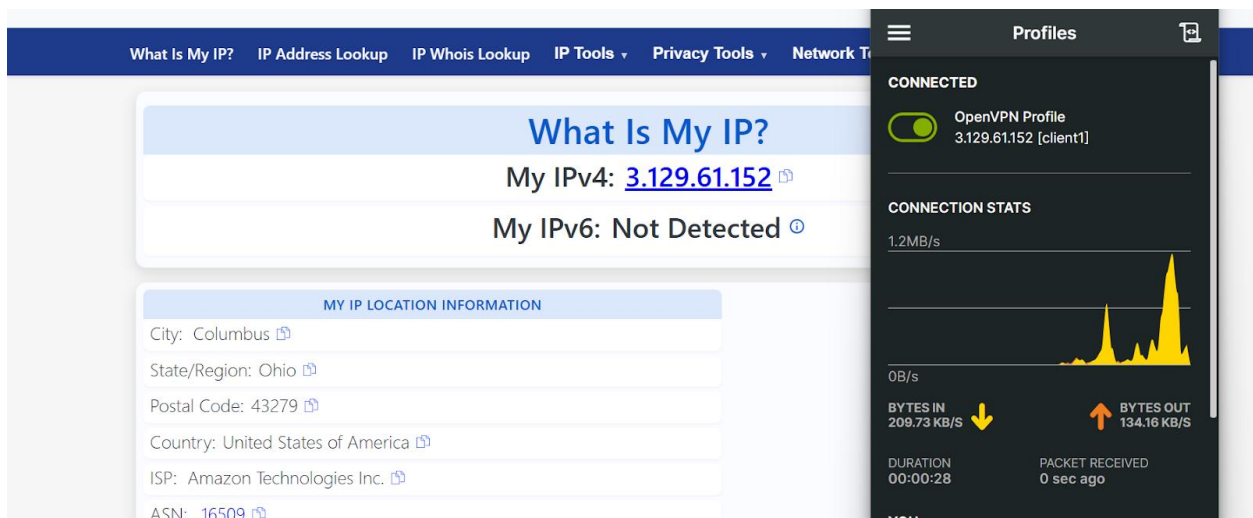
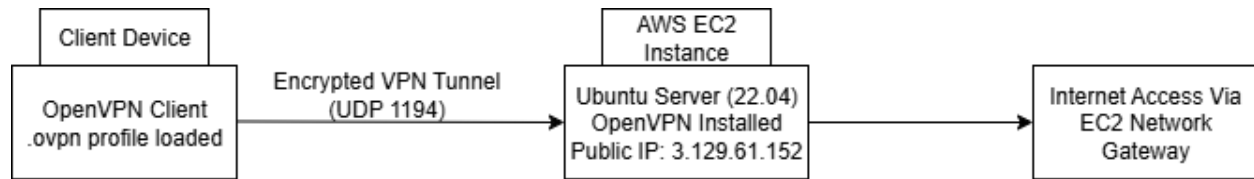


Figure 14: Comparison of OpenVPN IP address to what is displayed on whatismyip.com

This diagram displays the whole process of connecting to our VPN from our local device.



Conclusion:

This project successfully demonstrated the process of setting up a secure, encrypted VPN server on AWS using OpenVPN and Ubuntu. From the configuration of EC2 instances and security groups to OpenVPN setup, we got an up-and-running VPN tunnel that provided secure remote access and internet forwarding. In doing so, we acquired real hands-on experience in network security and cloud infrastructure. This project showcased a solid background understanding of cloud networking and is a steppingstone towards more advanced cloud security implementations.