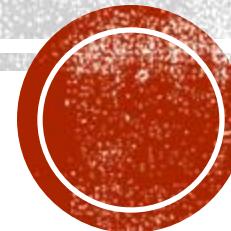


CROSS-PLATFORM VPN DEPLOYMENT ON AWS USING MACOS

Implementing OpenVPN with Secure Certificate Management
via Cyberduck and Tunnelblick



Project Description

This project focuses on establishing a secure Virtual Private Network (VPN) connection between a client machine and a Virtual Private Cloud (VPC) hosted on Amazon Web Services. The objective is to ensure confidentiality, integrity, and secure remote access to cloud resources using OpenVPN and Public Key Infrastructure (PKI)-based authentication.

Unlike traditional Windows-based configurations, this project demonstrates a cross-platform approach using macOS as the client operating system, with Cyberduck for secure certificate storage and Tunnelblick as the VPN client.

Goal

To set up secure Openvpn tunnel between a macos client and an AWS-hosted EC2 instance running Ubuntu 22.04 LTS



MAIN TASKS (3 STEPS):

1. Create Server and Client Certificates

Using Easy-RSA to generate a certificate authority (CA), server, and client keys for encrypted communication.

2. Configure OpenVPN Server on Ubuntu EC2

Install and configure OpenVPN on the AWS instance, enabling secure VPN access.

3. Connect and Verify VPN Connectivity

Import client configuration into Tunnelblick on macOS and verify successful connection using websites like *WhatIsMyIP.com*.



Tools and Technologies

Category	Tools / Technologies
Cloud Platform	AWS EC2
Operating System	Ubuntu 22.04 LTS (Server), macOS (Client)
VPN Software	OpenVPN
PKI Tool	Easy-RSA
File Management	Cyberduck
Remote Access	AWS EC2 Instance Connect
VPN Client	Tunnelblick
Verification Tool	WhatIsMyIP.com

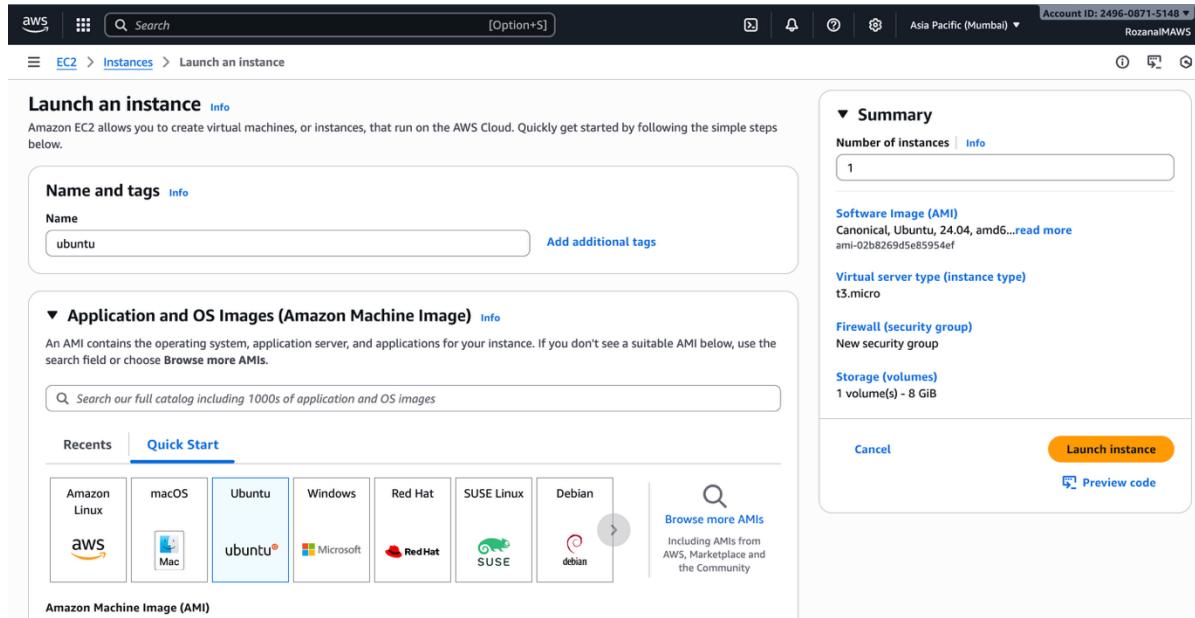


Step 1: Launch an EC2 Instance on AWS

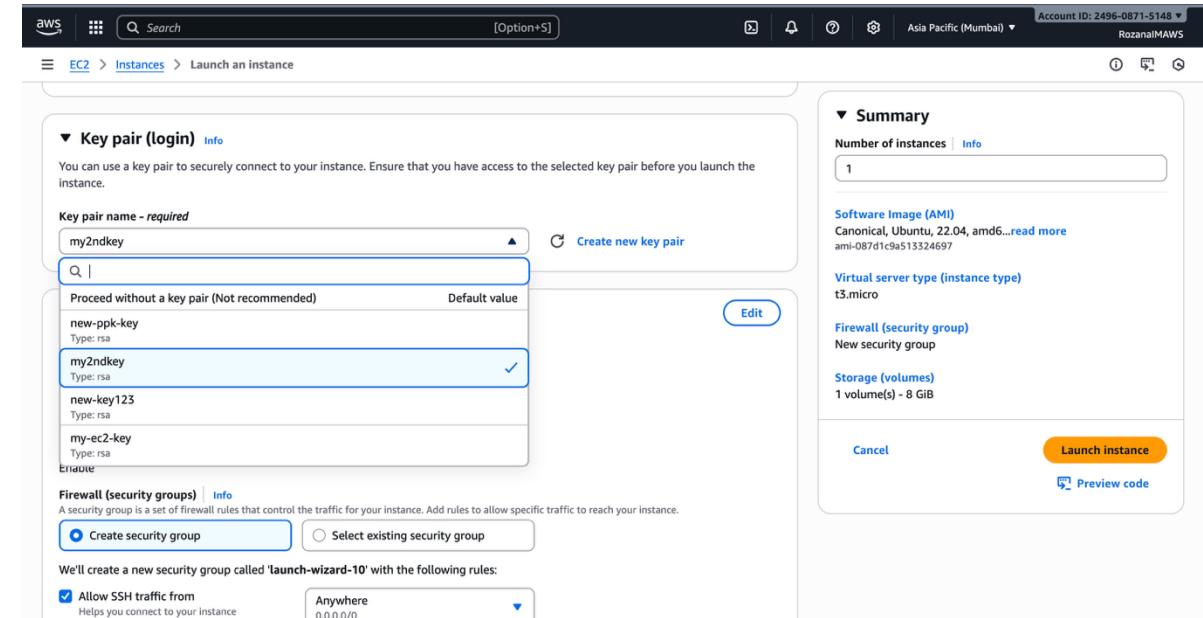
- Login to the AWS Management Console and navigate to the EC2 Dashboard.
- Click on “Launch Instance” and select Ubuntu 22.04 LTS as the operating system.
- Choose an instance type — t3.micro (eligible for free tier).
- Either select an existing Key Pair or create a new .Pem key (for SSH authentication).
- Finally, click “Launch Instance” to create and start your Ubuntu server on AWS.



LAUNCHING AN UBUNTU INSTANCE



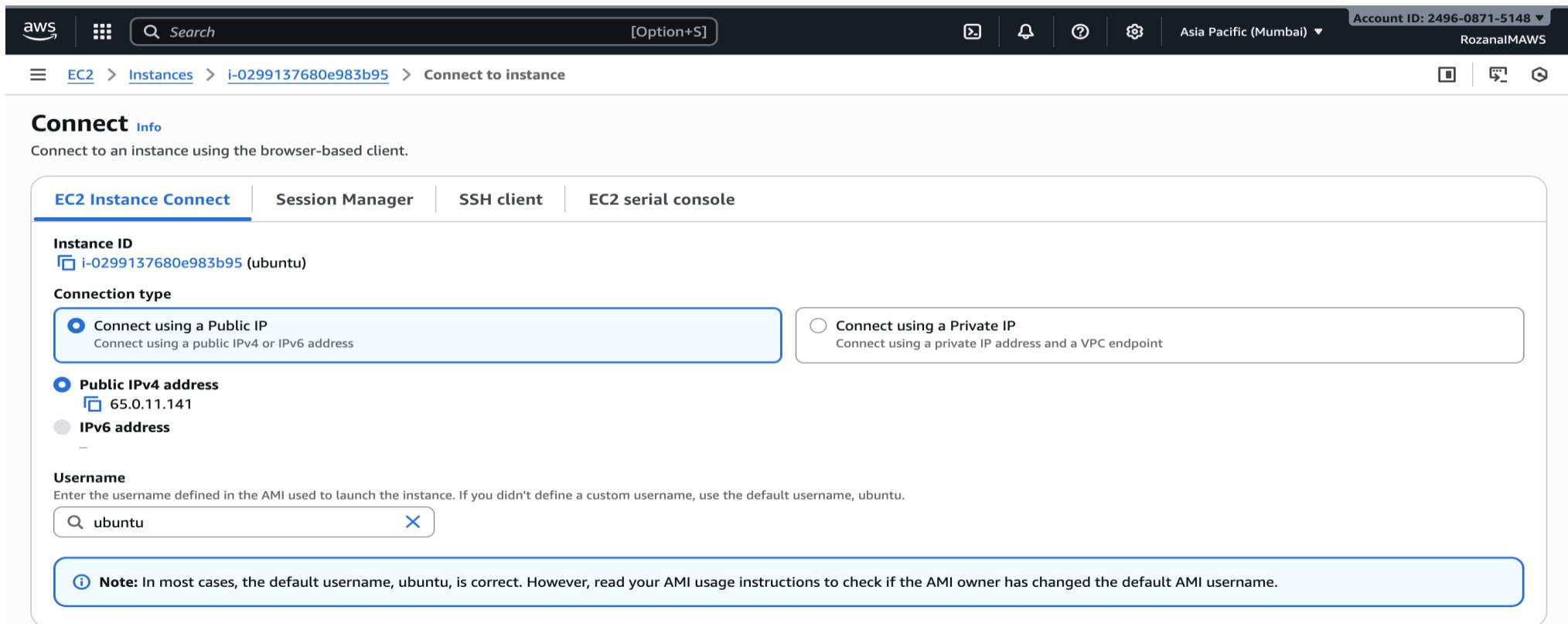
This screenshot shows the first step of the AWS EC2 'Launch an instance' wizard. The page title is 'Launch an instance'. It includes a summary section with 'Number of instances' set to 1, a 'Software Image (AMI)' section showing 'Canonical, Ubuntu, 24.04, amd64...', a 'Virtual server type (instance type)' section set to 't3.micro', a 'Firewall (security group)' section for 'New security group', and a 'Storage (volumes)' section indicating '1 volume(s) - 8 GiB'. At the bottom are 'Cancel', 'Launch instance', and 'Preview code' buttons.



This screenshot shows the second step of the AWS EC2 'Launch an instance' wizard. It's titled 'Key pair (login)'. It lists available key pairs: 'my2ndkey' (selected), 'new-ppkkey', 'my2ndkey', 'new-key123', 'my-ec2-key', and 'trnave'. A 'Create new key pair' button is available. Below this is a 'Firewall (security groups)' section with options to 'Create security group' or 'Select existing security group'. A note says 'We'll create a new security group called "launch-wizard-10" with the following rules:'. At the bottom are 'Cancel', 'Launch instance', and 'Preview code' buttons.

Step 2: Connect to the running Instance

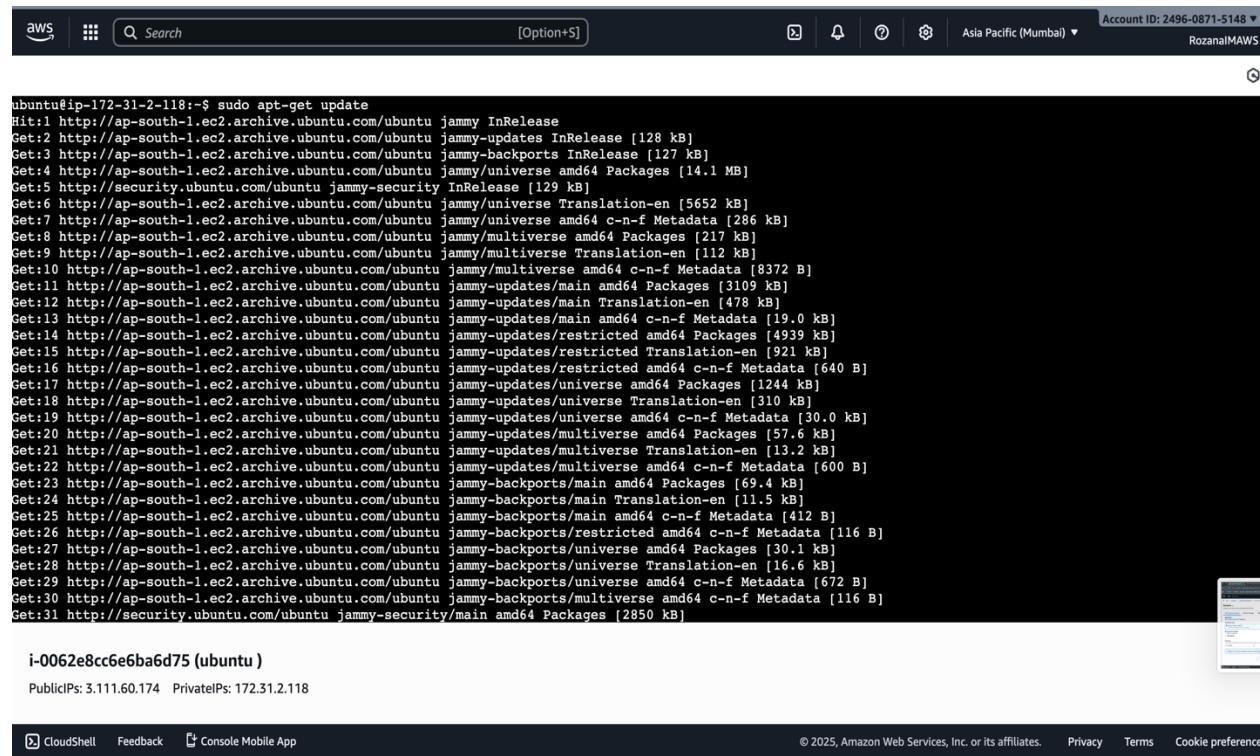
- Navigate to your EC2 instance in the AWS Console.
- Click on “Connect” and open the EC2 Instance Connect tab.
- Use the default username ubuntu and click “Connect” again.
- This opens a browser-based terminal, providing direct access to your Ubuntu server.



Step 3: Setting up and Configuring openvpn

3.1 Update Packages and Install OpenVPN & Easy-RSA

- Once the terminal session is open, update the package list using the command:
sudo apt-get update
- This ensures your server retrieves the latest package information from the repositories.



```
ubuntu@ip-172-31-2-118:~$ sudo apt-get update
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:4 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [14.1 MB]
Get:5 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:6 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/universe Translation-en [5652 kB]
Get:7 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 c-n-f Metadata [286 kB]
Get:8 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [217 kB]
Get:9 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse Translation-en [112 kB]
Get:10 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 c-n-f Metadata [8372 B]
Get:11 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [3109 kB]
Get:12 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [478 kB]
Get:13 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 c-n-f Metadata [19.0 kB]
Get:14 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [4939 kB]
Get:15 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted Translation-en [921 kB]
Get:16 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 c-n-f Metadata [640 B]
Get:17 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1244 kB]
Get:18 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe Translation-en [310 kB]
Get:19 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 c-n-f Metadata [30.0 kB]
Get:20 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 Packages [57.6 kB]
Get:21 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse Translation-en [13.2 kB]
Get:22 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 c-n-f Metadata [600 B]
Get:23 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/main amd64 Packages [69.4 kB]
Get:24 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/main Translation-en [11.5 kB]
Get:25 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/main amd64 c-n-f Metadata [412 B]
Get:26 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/restricted amd64 c-n-f Metadata [116 B]
Get:27 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 Packages [30.1 kB]
Get:28 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe Translation-en [16.6 kB]
Get:29 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 c-n-f Metadata [672 B]
Get:30 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/multiverse amd64 c-n-f Metadata [116 B]
Get:31 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [2850 kB]
```

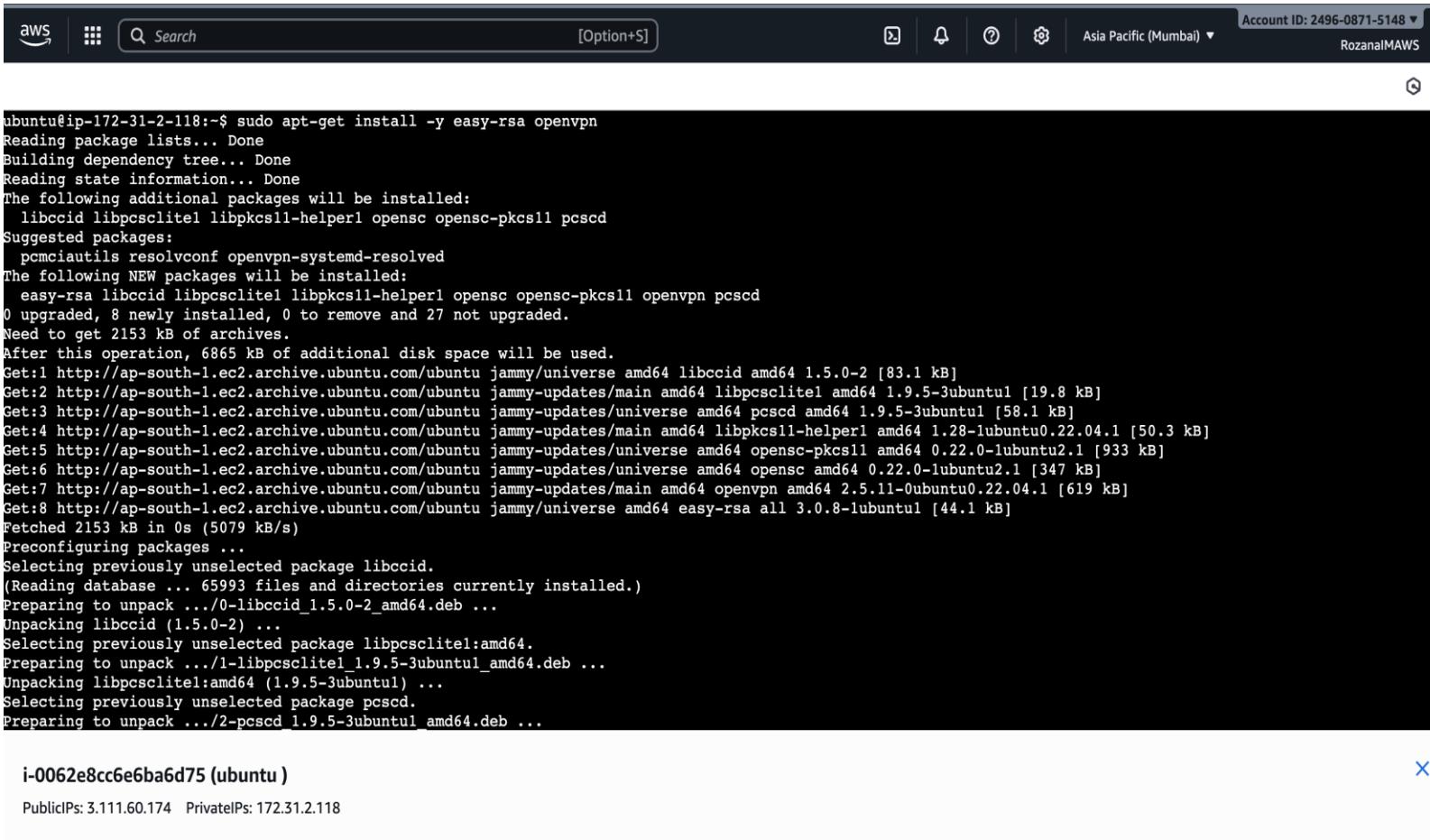
i-0062e8cc6e6ba6d75 (ubuntu)
Public IPs: 3.111.60.174 Private IPs: 172.31.2.118

CloudShell Feedback Console Mobile App © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

- Next, install the required tools with the command:

`sudo apt-get install -y easy-rsa openvpn`

- This installs OpenVPN for VPN setup and Easy-RSA for certificate and key management.
- Wait for the process to complete before moving to certificate creation and server configuration.



The screenshot shows a terminal window within an AWS CloudShell interface. The terminal displays the output of the command `sudo apt-get install -y easy-rsa openvpn`. The output includes package dependency resolution, download details, and the unpacking of several packages including libccid, libpcsc-lite, and pcscd. The terminal has a dark background with white text. At the bottom, it shows the user's session ID (i-0062e8cc6e6ba6d75), their Public IP (3.111.60.174), and Private IP (172.31.2.118). The AWS logo and navigation icons are visible at the top of the CloudShell interface.

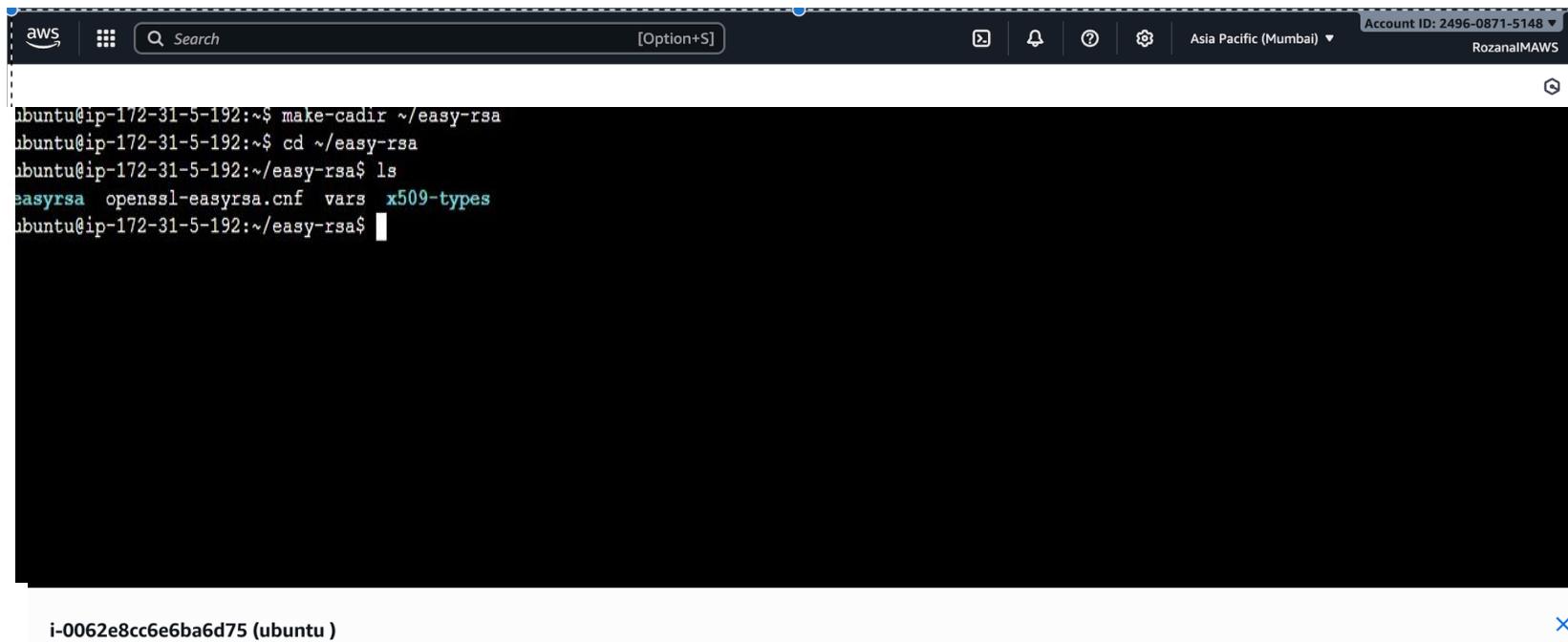
```
ubuntu@ip-172-31-2-118:~$ sudo apt-get install -y easy-rsa openvpn
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libccid libpcsc-lite1 libpkcs11-helper1 opensc opensc-pkcs11 pcscd
Suggested packages:
  pcmciautils resolvconf openvpn-systemd-resolved
The following NEW packages will be installed:
  easy-rsa libccid libpcsc-lite1 libpkcs11-helper1 opensc opensc-pkcs11 openvpn pcscd
0 upgraded, 8 newly installed, 0 to remove and 27 not upgraded.
Need to get 2153 kB of archives.
After this operation, 6865 kB of additional disk space will be used.
Get:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 libccid amd64 1.5.0-2 [83.1 kB]
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libpcsc-lite1 amd64 1.9.5-3ubuntul [19.8 kB]
Get:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 pcscd amd64 1.9.5-3ubuntul [58.1 kB]
Get:4 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libpkcs11-helper1 amd64 1.28-1ubuntu0.22.04.1 [50.3 kB]
Get:5 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 opensc-pkcs11 amd64 0.22.0-1ubuntu2.1 [933 kB]
Get:6 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 opensc amd64 0.22.0-1ubuntu2.1 [347 kB]
Get:7 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 openvpn amd64 2.5.11-0ubuntu0.22.04.1 [619 kB]
Get:8 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 easy-rsa all 3.0.8-1ubuntul [44.1 kB]
Fetched 2153 kB in 0s (5079 kB/s)
Preconfiguring packages ...
Selecting previously unselected package libccid.
(Reading database ... 65993 files and directories currently installed.)
Preparing to unpack .../0-libccid_1.5.0-2_amd64.deb ...
Unpacking libccid (1.5.0-2) ...
Selecting previously unselected package libpcsc-lite1:amd64.
Preparing to unpack .../1-libpcsc-lite1_1.9.5-3ubuntul_amd64.deb ...
Unpacking libpcsc-lite1:amd64 (1.9.5-3ubuntul) ...
Selecting previously unselected package pcscd.
Preparing to unpack .../2-pcscd_1.9.5-3ubuntul_amd64.deb ...
```

i-0062e8cc6e6ba6d75 (ubuntu)

PublicIPs: 3.111.60.174 PrivateIPs: 172.31.2.118

3.2 Set up the Easy-RSA Directory

- Create a new directory for Easy-RSA using: **make-cadir ~easy-rsa**
- Navigate into the newly created directory: **cd ~easy-rsa**
- Verify the directory contents by listing files: **ls**
- This prepares the working environment for certificate creation and management.

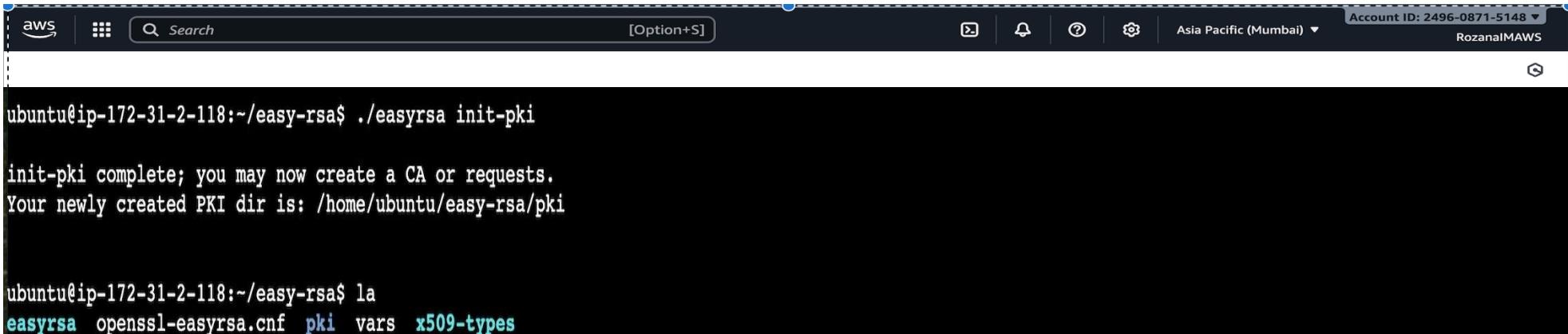


The screenshot shows a terminal window within the AWS CloudShell interface. The terminal is running on an Ubuntu instance (i-0062e8cc6e6ba6d75). The user has run the following commands:

```
ubuntu@ip-172-31-5-192:~$ make-cadir ~/easy-rsa
ubuntu@ip-172-31-5-192:~$ cd ~/easy-rsa
ubuntu@ip-172-31-5-192:~/easy-rsa$ ls
easyrsa  openssl-easyrsa.cnf  vars  x509-types
ubuntu@ip-172-31-5-192:~/easy-rsa$
```

3.3 Initialize PKI and Build the Certificate Authority (CA)

- Initialize the Easy-RSA environment by running: **./easyrsa init-pki**
- This command creates the essential Public Key Infrastructure (PKI) directory.



```
aws | [Option+S] | Asia Pacific (Mumbai) | Account ID: 2496-0871-5148 | RozanaMAWS

ubuntu@ip-172-31-2-118:~/easy-rsa$ ./easyrsa init-pki
init-pki complete; you may now create a CA or requests.
Your newly created PKI dir is: /home/ubuntu/easy-rsa/pki

ubuntu@ip-172-31-2-118:~/easy-rsa$ la
easyrsa openssl-easyrsa.cnf pki vars x509-types
```

- Next, build the Certificate Authority using: **./easyrsa build-ca**
- When prompted, provide the CA passphrase and enter a Common Name (e.g., “ubuntu / OpenVPN-CA etc ”) to generate the root certificate.

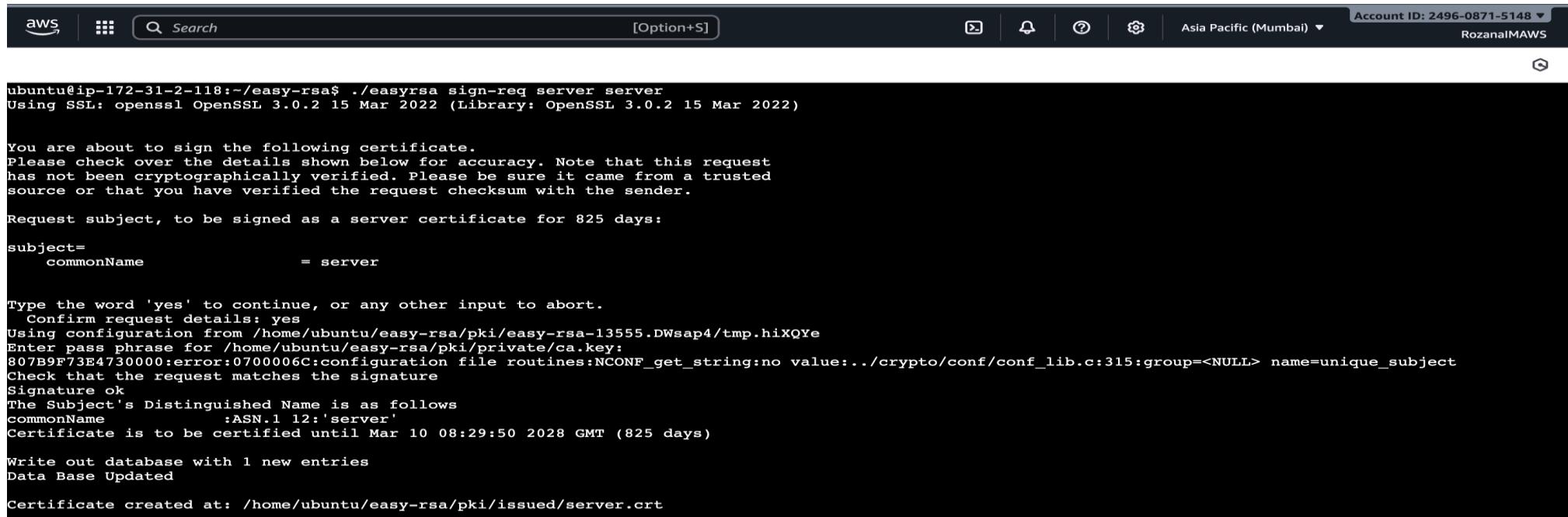
- After entering the CA passphrase and common name, the Certificate Authority (CA) creation process completes successfully.
 - Easy-RSA confirms the completion and indicates that the public CA certificate has been generated.
 - The CA certificate is now saved at the following path: **/home/ubuntu/easy-rsa/pki/ca.crt**
 - This file will be used later to sign both server and client certificates.

3.4 Generate the Server Certificate Request

- Generate a server certificate request by running: **./easysrsa gen-req server nopass**
 - This command creates the server's public and private key pair.
 - The “nopass” option allows the key to be used without requiring a password during service startup.

3.5 Sign the Server Certificate Request

- Sign the server certificate request with the existing Certificate Authority (CA) using:
./easyrsa sign-req server server
- Enter the CA passphrase when prompted to authorize the signing process.
- The server certificate is successfully generated, valid for 825 days, and saved at:
/home/ubuntu/easy-rsa/pki/issued/server.crt



```
ubuntu@ip-172-31-2-118:~/easy-rsa$ ./easyrsa sign-req server server
Using SSL: openssl OpenSSL 3.0.2 15 Mar 2022 (Library: OpenSSL 3.0.2 15 Mar 2022)

You are about to sign the following certificate.
Please check over the details shown below for accuracy. Note that this request
has not been cryptographically verified. Please be sure it came from a trusted
source or that you have verified the request checksum with the sender.

Request subject, to be signed as a server certificate for 825 days:
subject=
    commonName          = server

Type the word 'yes' to continue, or any other input to abort.
Confirm request details: yes
Using configuration from /home/ubuntu/easy-rsa/pki/easy-rsa-13555.DWsap4/tmp.hiXQYc
Enter pass phrase for /home/ubuntu/easy-rsa/pki/private/ca.key:
807B9F73E4730000:error:0700006C:configuration file routines:NCONF_get_string:no value:../crypto/conf/conf_lib.c:315:group=<NULL> name=unique_subject
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName          :ASN.1 12:'server'
Certificate is to be certified until Mar 10 08:29:50 2028 GMT (825 days)

Write out database with 1 new entries
Data Base Updated

Certificate created at: /home/ubuntu/easy-rsa/pki/issued/server.crt
```

i-0062e8cc6e6ba6d75 (ubuntu)

PublicIPs: 3.111.60.174 PrivateIPs: 172.31.2.118

Step 3.6 : Generate Diffie-Hellman (DH) Parameters

- Execute the following command to generate Diffie-Hellman parameters: **./easyrsa gen-dh**
 - The DH parameters are essential for secure key exchange between the VPN server and clients.
 - This process enhances the cryptographic strength of the OpenVPN connection.

i-0062e8cc6e6ba6d75 (ubuntu)

Public IPs: 3.111.60.174 Private IPs: 172.31.2.11

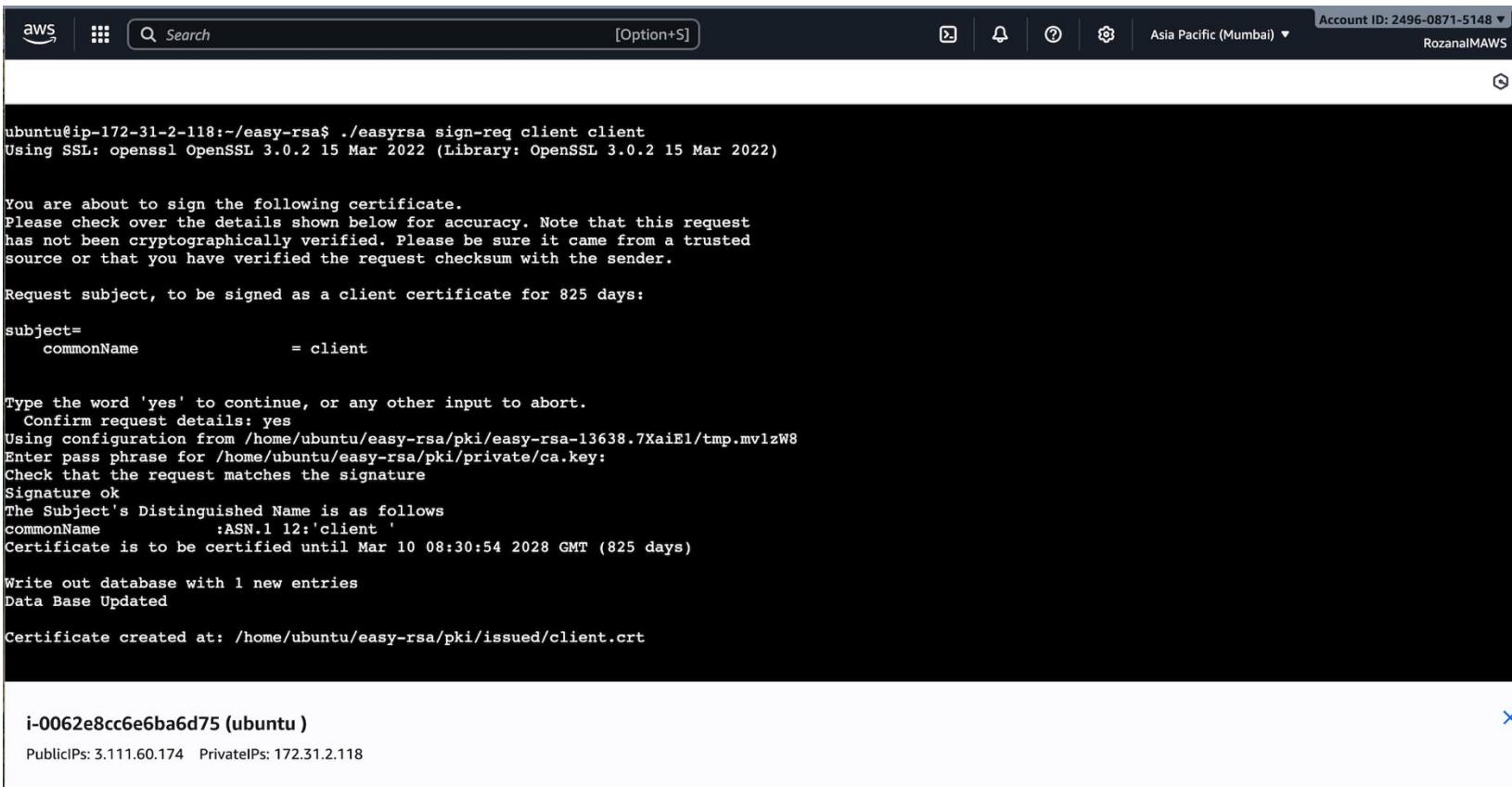
Step 3.7 : Generate and Sign the Client Certificate

- Generate a client certificate request using: **./easyrsa gen-req client nopass**
 - This command creates the client's public and private key pair without a password.
 - The request file will later be signed by the CA to make it valid.

i-0062e8cc6e6ba6d75 (ubuntu)

Public IPs: 3.111.60.174 Private IPs: 172.31.2.118

- Sign the client certificate request with the established Certificate Authority (CA):
./easyrsa sign-req client client
- Enter the CA passphrase when prompted to authorize the signing.
- The signed client certificate is valid for 825 days and is saved at:
/home/ubuntu/easy-rsa/pki/issued/client.crt



The screenshot shows a terminal window within the AWS CloudWatch interface. The terminal output is as follows:

```

ubuntu@ip-172-31-2-118:~/easy-rsa$ ./easyrsa sign-req client client
Using SSL: openssl OpenSSL 3.0.2 15 Mar 2022 (Library: OpenSSL 3.0.2 15 Mar 2022)

You are about to sign the following certificate.
Please check over the details shown below for accuracy. Note that this request
has not been cryptographically verified. Please be sure it came from a trusted
source or that you have verified the request checksum with the sender.

Request subject, to be signed as a client certificate for 825 days:
subject=
    commonName      = client

Type the word 'yes' to continue, or any other input to abort.
Confirm request details: yes
Using configuration from /home/ubuntu/easy-rsa/pki/easy-rsa-13638.7XaiEl/tmp.mvlzW8
Enter pass phrase for /home/ubuntu/easy-rsa/pki/private/ca.key:
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName :ASN.1 12:'client'
Certificate is to be certified until Mar 10 08:30:54 2028 GMT (825 days)

Write out database with 1 new entries
Data Base Updated

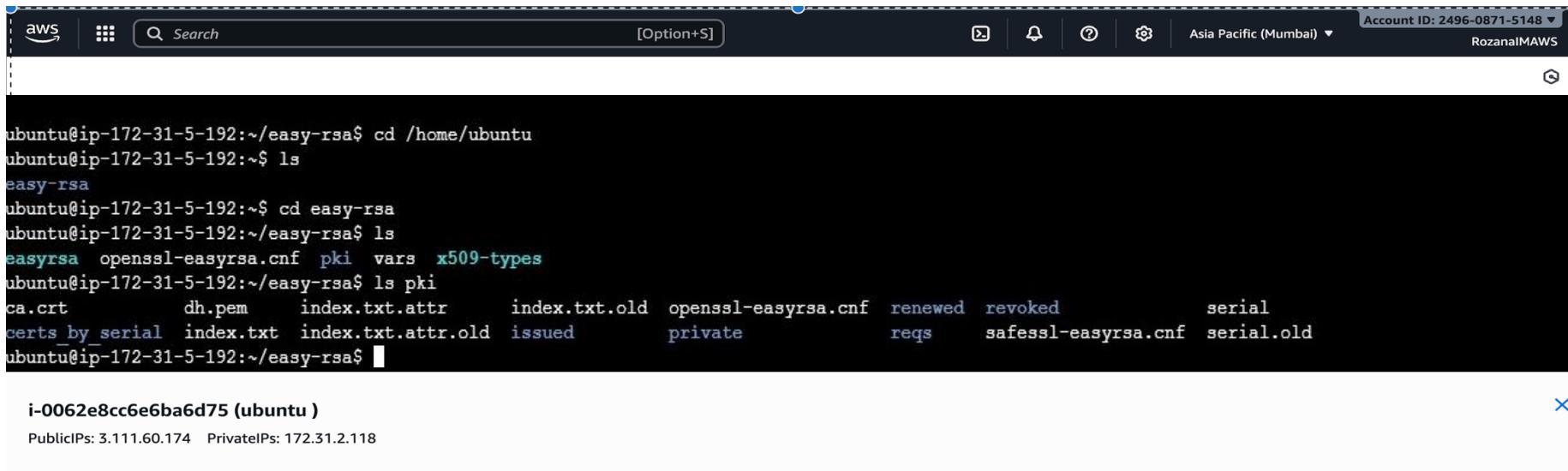
Certificate created at: /home/ubuntu/easy-rsa/pki/issued/client.crt

```

At the bottom of the terminal window, there is a status bar with the text "i-0062e8cc6e6ba6d75 (ubuntu)" and "PublicIPs: 3.111.60.174 PrivateIPs: 172.31.2.118".

Step 3.8 : Verify PKI File Creation and Directory Structure

- List the contents of the easy-rsa directory using: ls
- This confirms that all required files — including the CA certificate, server certificate, client certificate, and Diffie-Hellman parameters — have been generated successfully.
- The presence of these files in the appropriate folders (e.g., /pki/issued/, /pki/private/) verifies a complete and properly organized PKI setup.



The screenshot shows a terminal window within the AWS CloudShell interface. The terminal output displays the following command sequence and file listing:

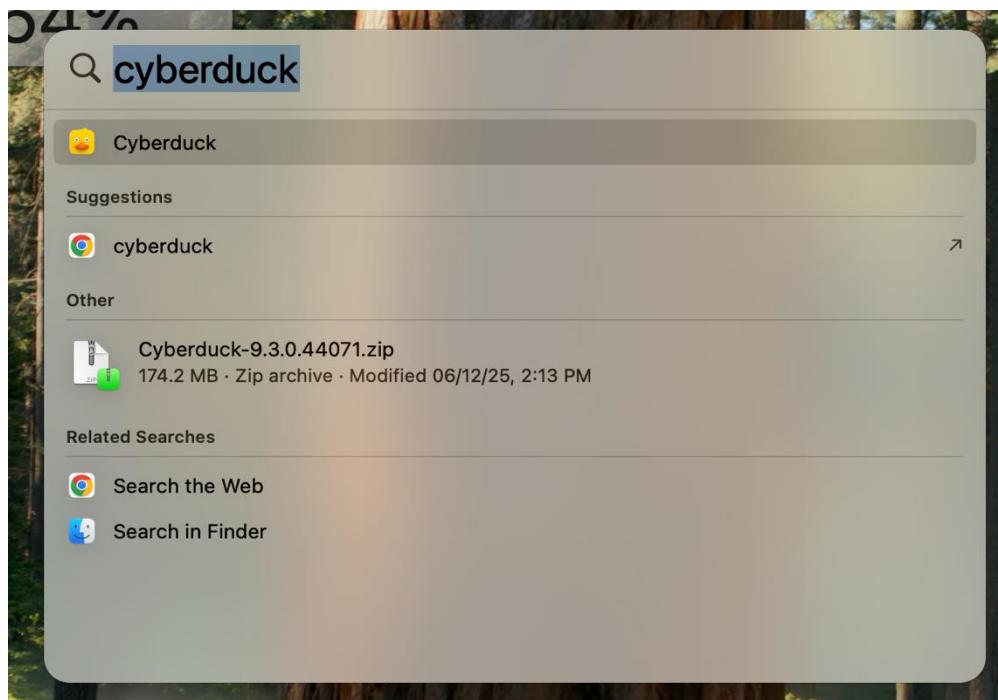
```
ubuntu@ip-172-31-5-192:~/easy-rsa$ cd /home/ubuntu
ubuntu@ip-172-31-5-192:~$ ls
easy-rsa
ubuntu@ip-172-31-5-192:~$ cd easy-rsa
ubuntu@ip-172-31-5-192:~/easy-rsa$ ls
easyrsa  openssl-easyrsa.cnf  pki  vars  x509-types
ubuntu@ip-172-31-5-192:~/easy-rsa$ ls pki
ca.crt      dh.pem      index.txt.attr      index.txt.old  openssl-easyrsa.cnf  renewed  revoked      serial
certs_by_serial  index.txt  index.txt.attr.old  issued      private      reqs  safessl-easyrsa.cnf  serial.old
ubuntu@ip-172-31-5-192:~/easy-rsa$
```

Below the terminal window, the CloudShell interface shows the instance identifier and public IP address:

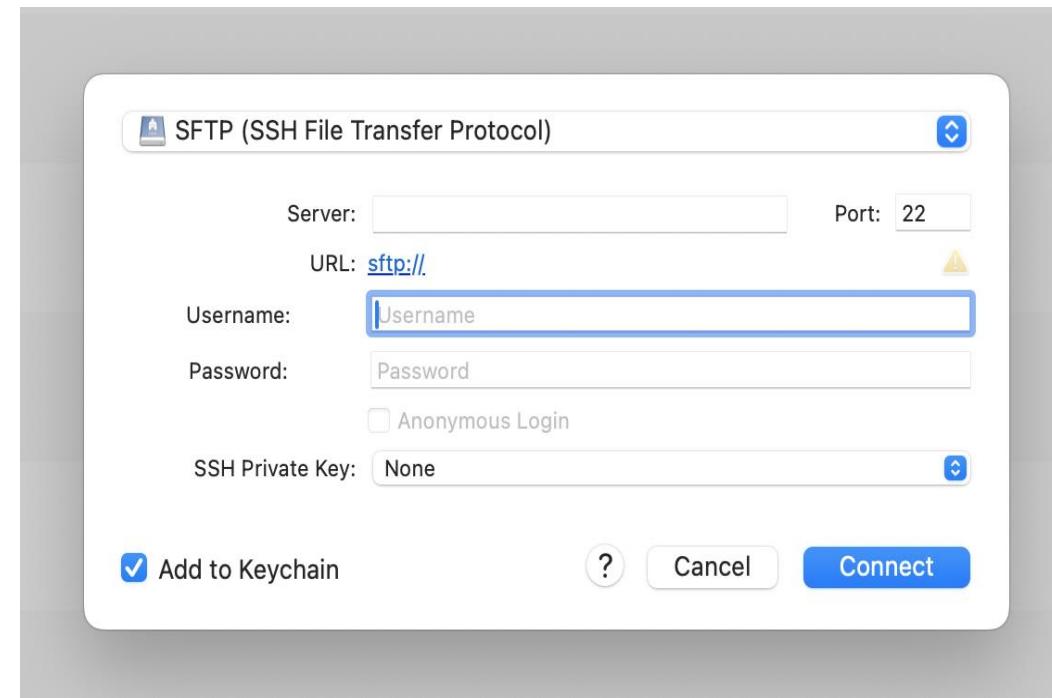
i-0062e8cc6e6ba6d75 (ubuntu)
PublicIPs: 3.111.60.174 PrivateIPs: 172.31.2.118

Step 4: Use Cyberduck to transfer certificates and keys via SFTP

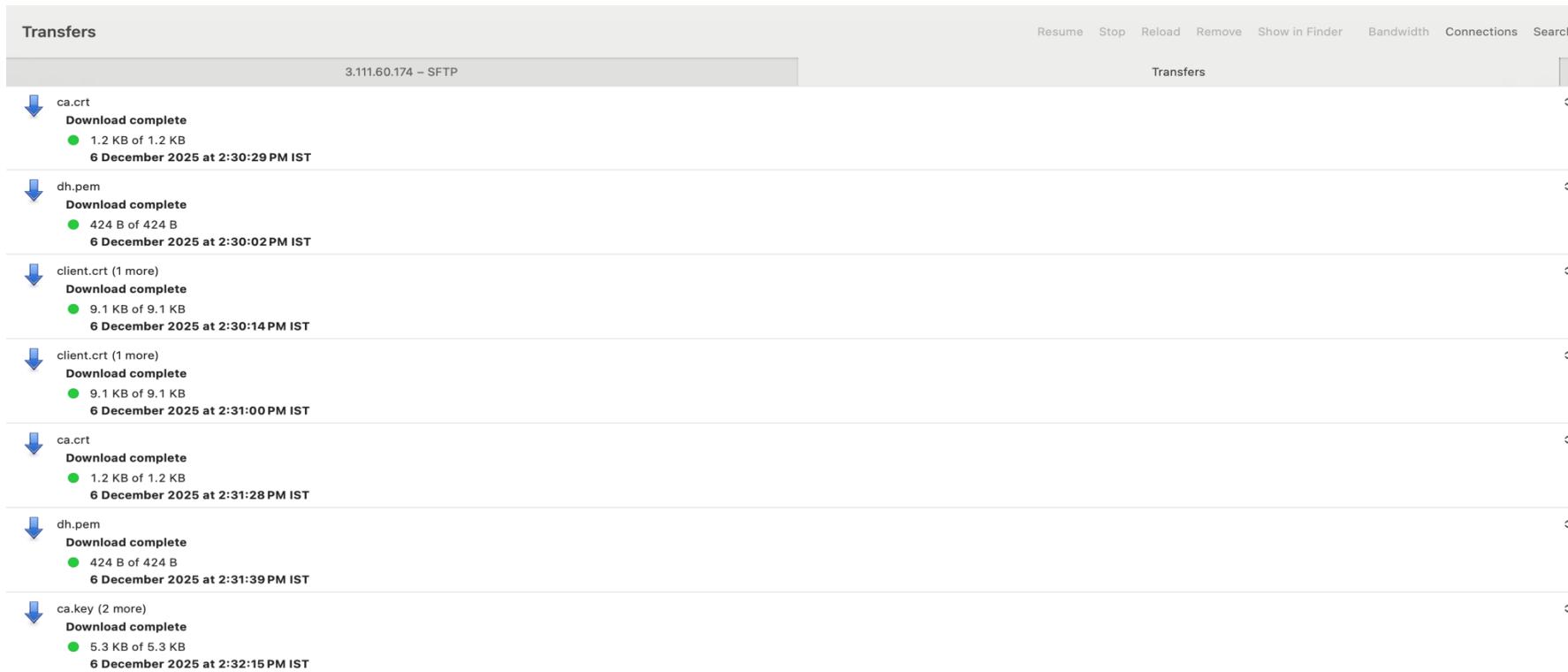
- Open Cyberduck on your macOS for Secure File Transfer.
- Some applications like Cyberduck or PuTTY require a .ppk key instead of a .pem file.
- Convert the PEM key to PPK format using PuTTYgen with the following command(in terminal):
`puttygen /Users/apple/Downloads/my-ec2-key.pem -o /Users/apple/Downloads/my-ec2-key.ppk`
- The converted .ppk key can now be used in Cyberduck for secure file transfer via SFTP.



- Click on “Open Connection.”
- From the dropdown, select SFTP (SSH File Transfer Protocol) as the connection type.
- In the Server (Hostname) field, enter your instance’s Public IPv4 address from the AWS EC2 dashboard.
- Under SSH Private Key, browse and select your instance’s .ppk key file (downloaded when the EC2 instance was created).
- Click Connect to establish a secure file transfer session between your local system and the EC2 instance.



- The screenshot below shows the successful file transfer completed using Cyberduck.
- These files are essential for configuring the OpenVPN client on your macOS system.

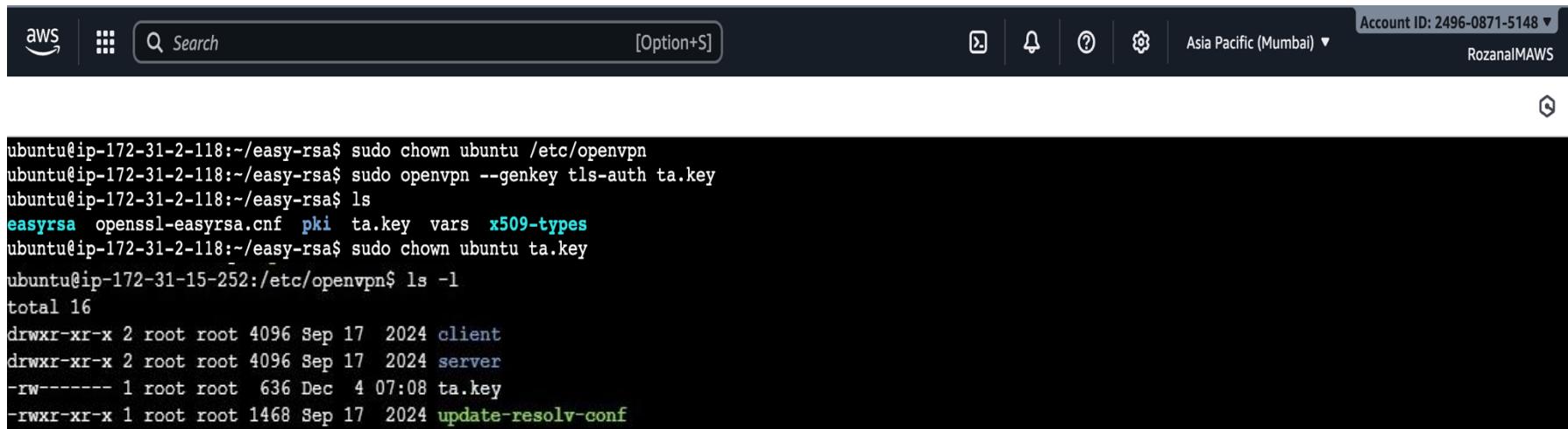


OpenVPN Server Configuration and Network Setup



Step 1: Generate the TLS Authentication key(ta.key)

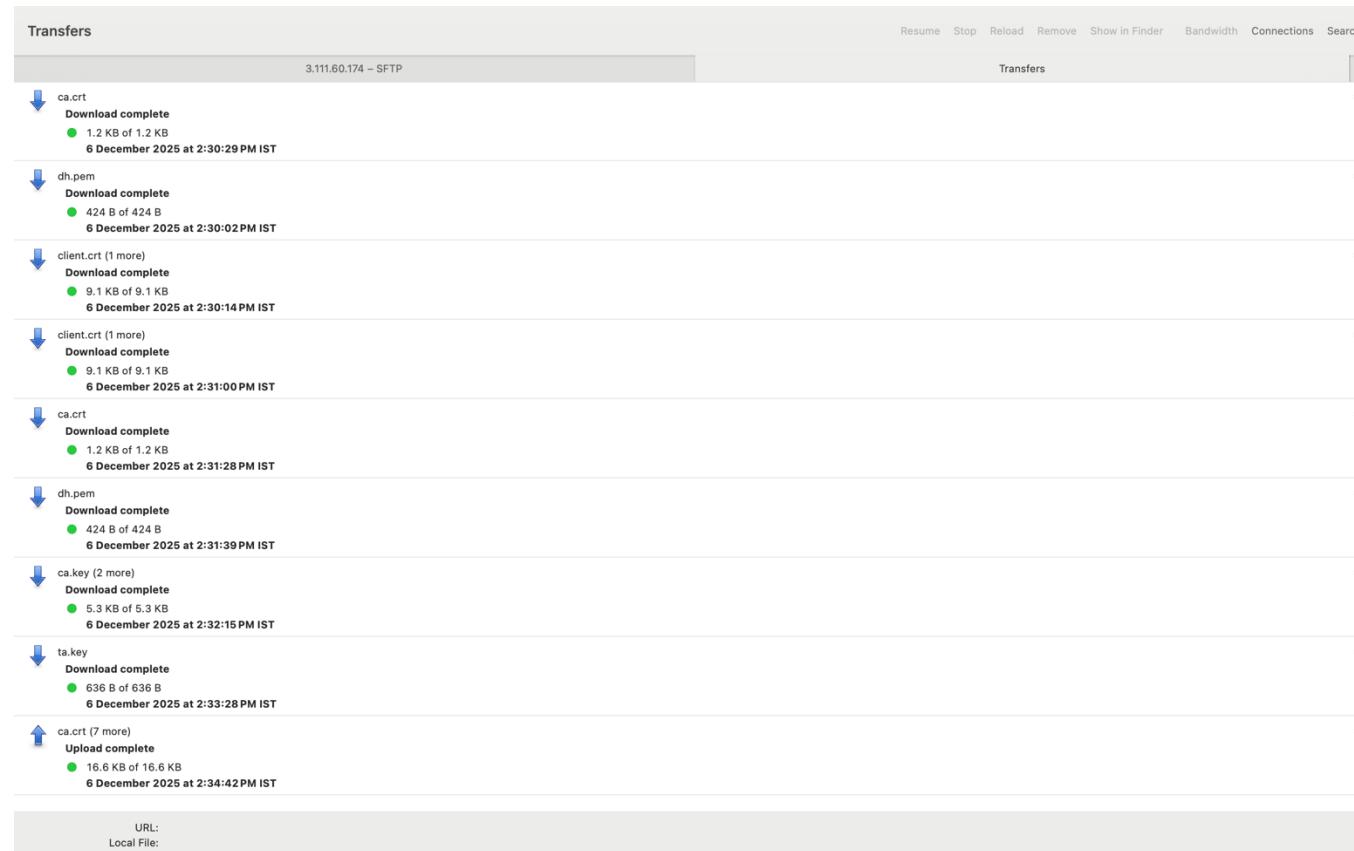
- As OpenVPN is already installed, proceed to create the TLS authentication key by running:
openvpn --genkey tls-auth ta.key
- This command generates a file named ta.key, which adds an extra layer of security to the VPN connection.
- The ta.key ensures that only clients with the correct shared key can initiate communication with the VPN server, protecting against unauthorized access and attacks.



The screenshot shows a AWS CloudShell interface with a dark theme. At the top, there's a navigation bar with the AWS logo, a search bar, and account information (Account ID: 2496-0871-5148, Region: Asia Pacific (Mumbai), User: RozanaMAWS). Below the bar is a toolbar with icons for copy, paste, refresh, and settings. The main area is a terminal window displaying a terminal session:

```
ubuntu@ip-172-31-2-118:~/easy-rsa$ sudo chown ubuntu /etc/openvpn
ubuntu@ip-172-31-2-118:~/easy-rsa$ sudo openvpn --genkey tls-auth ta.key
ubuntu@ip-172-31-2-118:~/easy-rsa$ ls
easyrsa openssl-easyrsa.cnf pki ta.key vars x509-types
ubuntu@ip-172-31-2-118:~/easy-rsa$ sudo chown ubuntu ta.key
ubuntu@ip-172-31-15-252:/etc/openvpn$ ls -l
total 16
drwxr-xr-x 2 root root 4096 Sep 17 2024 client
drwxr-xr-x 2 root root 4096 Sep 17 2024 server
-rw----- 1 root root 636 Dec 4 07:08 ta.key
-rwxr-xr-x 1 root root 1468 Sep 17 2024 update-resolv-conf
```

- After generating **ta.key**, open Cyberduck and connect to your EC2 instance via SFTP.
- Locate the **ta.key** file and download it to your local system.
- This key will be used later in the OpenVPN client configuration for secure access.



Step 2: Copy Certificates and keys to openvpn Directory

- Copy all the required VPN certificates and keys to the OpenVPN configuration directory using:

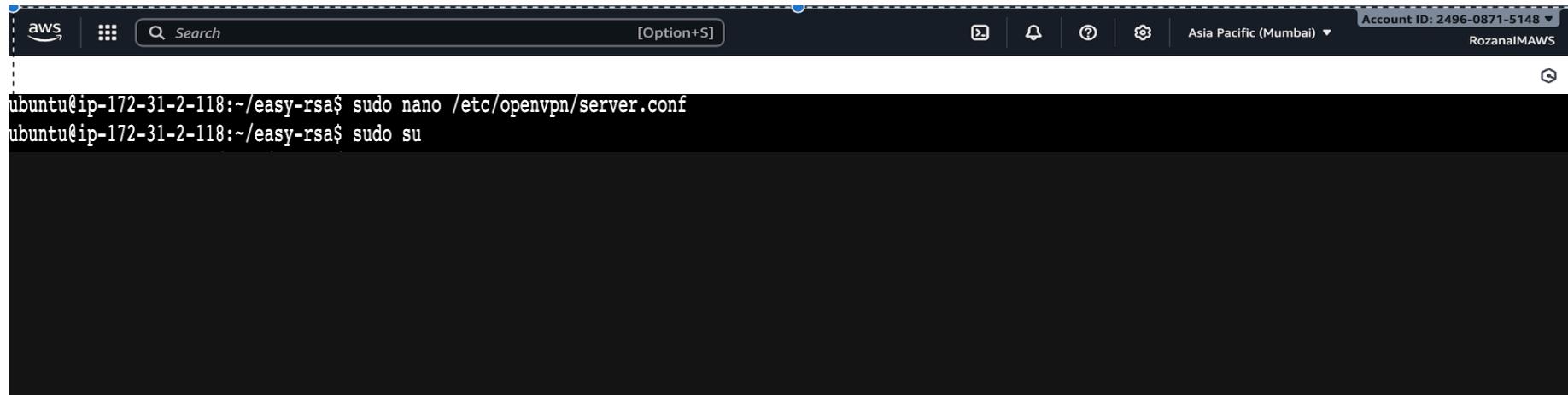
```
sudo cp ~/easy-rsa/pki/ca.crt /etc/openvpn/  
sudo cp ~/easy-rsa/pki/issued/server.crt /etc/openvpn/  
sudo cp ~/easy-rsa/pki/private/server.key /etc/openvpn/  
sudo cp ~/easy-rsa/pki/dh.pem /etc/openvpn/  
sudo cp ~/easy-rsa/ta.key /etc/openvpn/
```

- This ensures that the OpenVPN server can access all necessary files during startup.
- These files include the CA certificate, server certificate, private key, Diffie-Hellman parameters, and TLS key.



Step 3: Configure the OpenVPN server

- Create and open the configuration file:
`sudo nano /etc/openvpn/server.conf`
- Add the following configuration parameters to define the VPN server:
 1. Port: 1194 (UDP protocol)
 2. Certificates & keys: ca.crt, server.crt, server.key, dh.pem, ta.key
 3. Network: server 10.8.0.0 255.255.255.0
 4. DNS: 8.8.8.8
 5. Encryption: AES-256-CBC



The screenshot shows a terminal window within the AWS CloudShell interface. The terminal is running on an Ubuntu system (version 18.04 LTS). The user has run two commands:

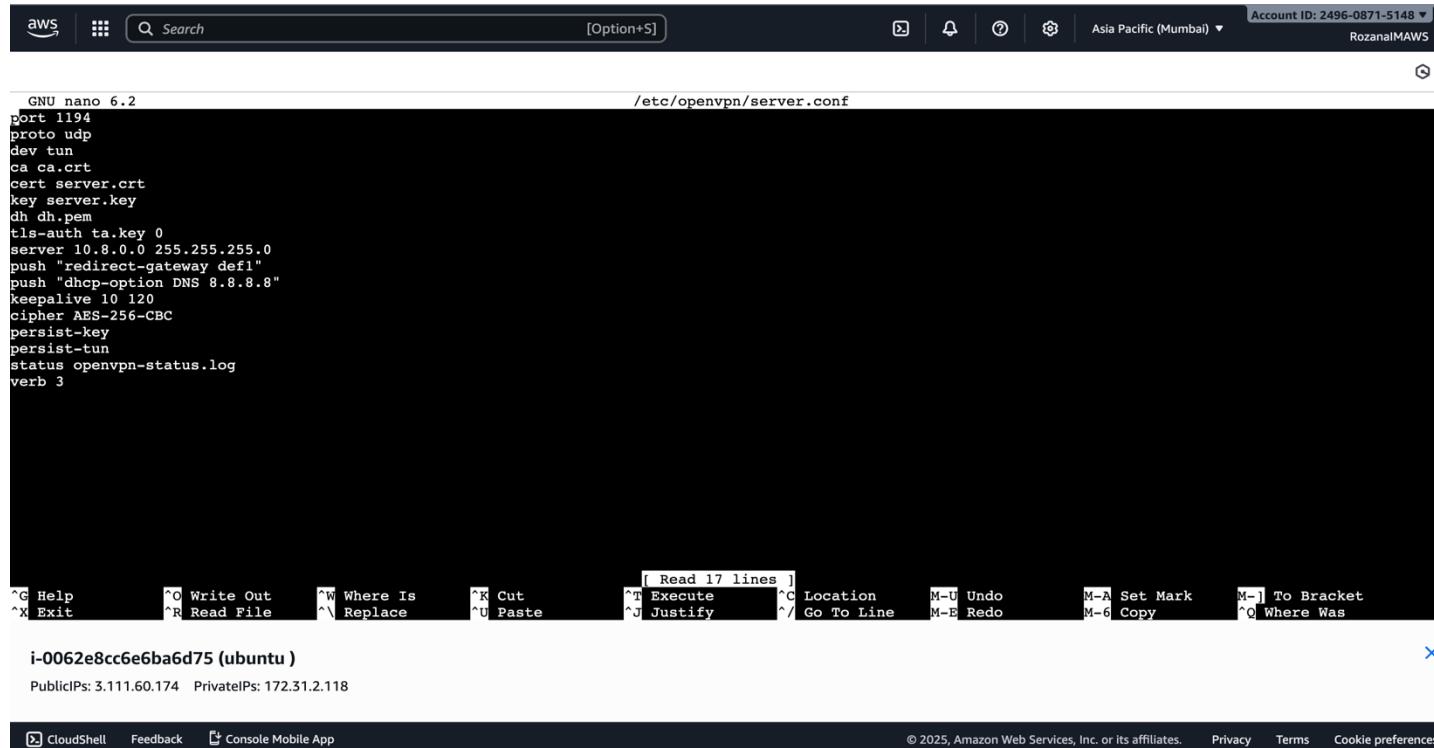
```
ubuntu@ip-172-31-2-118:~/easy-rsa$ sudo nano /etc/openvpn/server.conf
ubuntu@ip-172-31-2-118:~/easy-rsa$ sudo su
```

The terminal window has a dark background and light-colored text. The AWS navigation bar is visible at the top, showing the account ID (2496-0871-5148), region (Asia Pacific (Mumbai)), and the user's name (RozanalMAWS).



3.1 Update the VPN CIDR in Configuration

- In the server.conf file, locate the line: **server 10.8.0.0 255.255.255.0**
- Replace this with your VPC's CIDR block to align the VPN subnet with your AWS network.
Example: **server 172.31.0.0 255.255.255.0**
- This ensures that VPN-connected clients can securely access resources inside the AWS private network.



```
GNU nano 6.2
port 1194
proto udp
dev tun
ca ca.crt
cert server.crt
key server.key
dh dh.pem
tls-auth ta.key 0
server 10.8.0.0 255.255.255.0
push "redirect-gateway def1"
push "dhcp-option DNS 8.8.8.8"
keepalive 10 120
cipher AES-256-CBC
persist-key
persist-tun
status openvpn-status.log
verb 3
```

The terminal shows the configuration file `/etc/openvpn/server.conf`. The configuration includes settings for port (1194), protocol (udp), device (tun), certificate authority (ca), certificate (server.crt), key (server.key), Diffie-Hellman parameters (dh), TLS authentication (tls-auth), and a server address (10.8.0.0). It also includes push commands for redirecting traffic and setting DNS to 8.8.8.8, and specifies cipher (AES-256-CBC), persistency, and a log file (openvpn-status.log). The verb level is set to 3.

At the bottom of the terminal window, there is a menu bar with various keyboard shortcuts:

- Help (^G)
- Write Out (^O)
- Where Is (^W)
- Cut (^X)
- Paste (^V)
- Execute (^T)
- Location (^C)
- Undo (M-U)
- Redo (M-E)
- Set Mark (M-A)
- Copy (M-C)
- To Bracket (M-B)
- Where Was (M-W)

Below the terminal window, the session information is displayed:

i-0062e8cc6e6ba6d75 (ubuntu)
Public IPs: 3.111.60.174 Private IPs: 172.31.2.118

CloudShell Feedback Console Mobile App © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

- Enable IP forwarding to allow routing between clients and the internet:
`sudo sysctl -w net.ipv4.ip_forward=1`
`sudo echo "net.ipv4.ip_forward = 1" >> /etc/sysctl.conf`
- Set up firewall (UFW) and NAT rules to allow VPN traffic and secure communication:
`sudo iptables -t nat -A POSTROUTING -s 172.31.0.0/24 -o ens5 -j MASQUERADE`
`sudo ufw allow 1194/udp`
`sudo ufw allow OpenSSH`
`sudo ufw enable`

```
root@ip-172-31-2-118:/home/ubuntu/easy-rsa# sudo sysctl -w net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
root@ip-172-31-2-118:/home/ubuntu/easy-rsa# sudo echo "net.ipv4.ip_forward = 1" >> /etc/sysctl.conf
root@ip-172-31-2-118:/home/ubuntu/easy-rsa# sudo iptables -t nat -A POSTROUTING -s 172.31.0.0/16 -o ens5 -j MASQUERADE
root@ip-172-31-2-118:/home/ubuntu/easy-rsa# sudo ufw allow 1194/udp
Rules updated
Rules updated (v6)
root@ip-172-31-2-118:/home/ubuntu/easy-rsa# sudo ufw allow OpenSSH
Rules updated
Rules updated (v6)
root@ip-172-31-2-118:/home/ubuntu/easy-rsa# sudo ufw enable
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
Firewall is active and enabled on system startup
root@ip-172-31-2-118:/home/ubuntu/easy-rsa# sudo ufw status
Status: active

          To                         Action    From
          --                         ----    ---
1194/udp           ALLOW      Anywhere
OpenSSH            ALLOW      Anywhere
1194/udp (v6)     ALLOW      Anywhere (v6)
OpenSSH (v6)       ALLOW      Anywhere (v6)

root@ip-172-31-2-118:/home/ubuntu/easy-rsa# nano /etc/ufw/before.rule
```

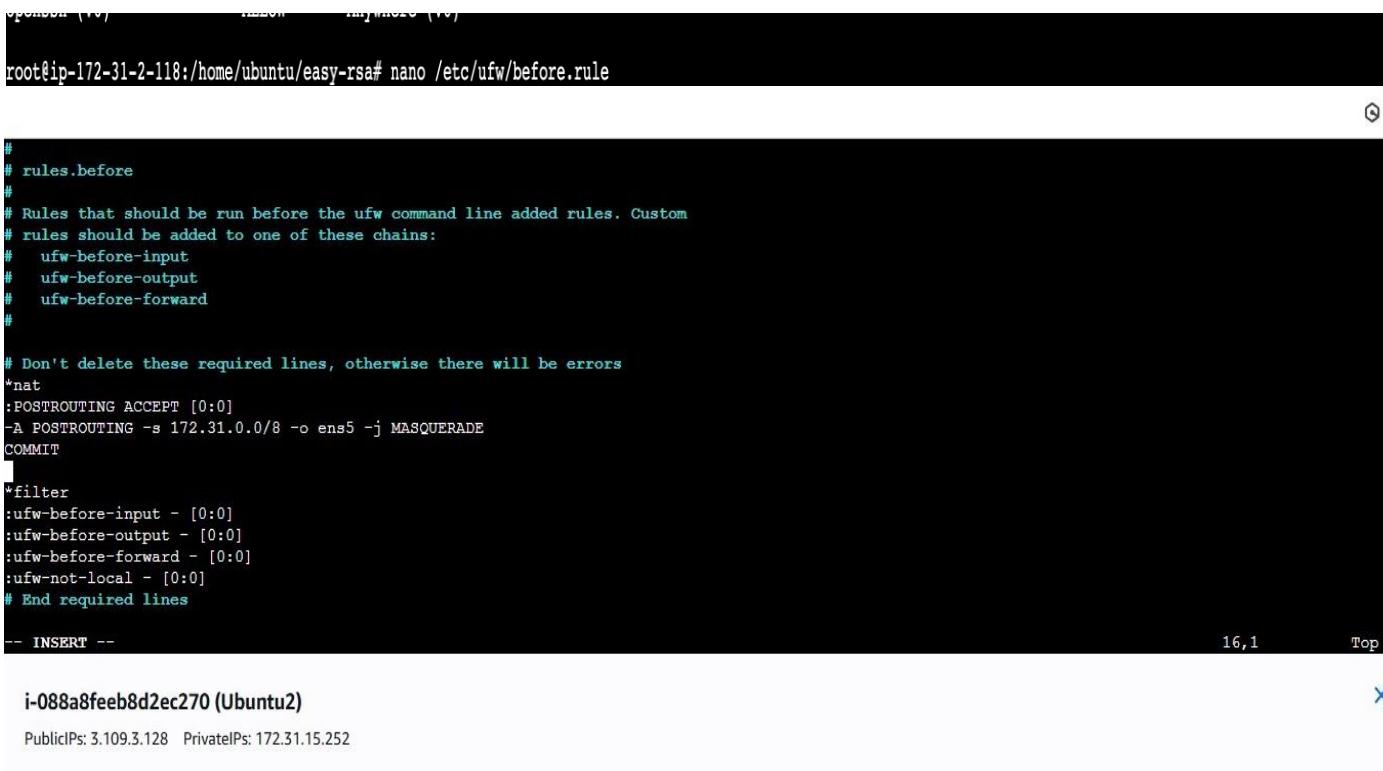
i-0062e8cc6e6ba6d75 (ubuntu)

PublicIPs: 3.111.60.174 PrivateIPs: 172.31.2.118

Step 4: Configure NAT and Forwarding rules

- Edit the UFW rules file to add NAT configuration before the *filter* section:
`sudo nano /etc/ufw/before.rules`
- Add the following lines:

```
*nat  
:POSTROUTING ACCEPT [0:0]  
-A POSTROUTING -s 172.31.0.0/8 -o ens5 -j MASQUERADE  
COMMIT
```



The screenshot shows a terminal window with a black background and white text. At the top, it says "root@ip-172-31-2-118:/home/ubuntu/easy-rsa# nano /etc/ufw/before.rule". The main content of the terminal is the UFW rules file with the new NAT rule added:

```
#  
# rules.before  
#  
# Rules that should be run before the ufw command line added rules. Custom  
# rules should be added to one of these chains:  
#   ufw-before-input  
#   ufw-before-output  
#   ufw-before-forward  
#  
# Don't delete these required lines, otherwise there will be errors  
*nat  
:POSTROUTING ACCEPT [0:0]  
-A POSTROUTING -s 172.31.0.0/8 -o ens5 -j MASQUERADE  
COMMIT  
|  
*filter  
:ufw-before-input - [0:0]  
:ufw-before-output - [0:0]  
:ufw-before-forward - [0:0]  
:ufw-not-local - [0:0]  
# End required lines  
-- INSERT --
```

At the bottom right of the terminal window, there are status indicators: "16,1" and "Top". Below the terminal window, the system status bar shows "i-088a8feeb8d2ec270 (Ubuntu2)" and "PublicIPs: 3.109.3.128 PrivateIPs: 172.31.15.252". A red circular icon is located in the bottom right corner of the slide.

- Update the UFW default configuration to enable packet forwarding:
`sudo nano /etc/default/ufw`
 - Set:
`DEFAULT_FORWARD_POLICY="ACCEPT"`
 - These settings allow VPN clients to route traffic through the VPN server and access internal or internet resources securely.
-

```
#  
  
# Set to yes to apply rules to support IPv6 (no means only IPv6 on loopback  
# accepted). You will need to 'disable' and then 'enable' the firewall for  
# the changes to take affect.  
IPV6=yes  
  
# Set the default input policy to ACCEPT, DROP, or REJECT. Please note that if  
# you change this you will most likely want to adjust your rules.  
DEFAULT_INPUT_POLICY="DROP"  
  
# Set the default output policy to ACCEPT, DROP, or REJECT. Please note that if  
# you change this you will most likely want to adjust your rules.  
DEFAULT_OUTPUT_POLICY="ACCEPT"  
  
# Set the default forward policy to ACCEPT, DROP or REJECT. Please note that  
# if you change this you will most likely want to adjust your rules  
DEFAULT_FORWARD_POLICY="ACCEPT"  
  
# Set the default application policy to ACCEPT, DROP, REJECT or SKIP. Please  
# note that setting this to ACCEPT may be a security risk. See 'man ufw' for  
# details  
DEFAULT_APPLICATION_POLICY="SKIP"  
"/etc/default/ufw" 47L, 1899B
```

19,30 4%

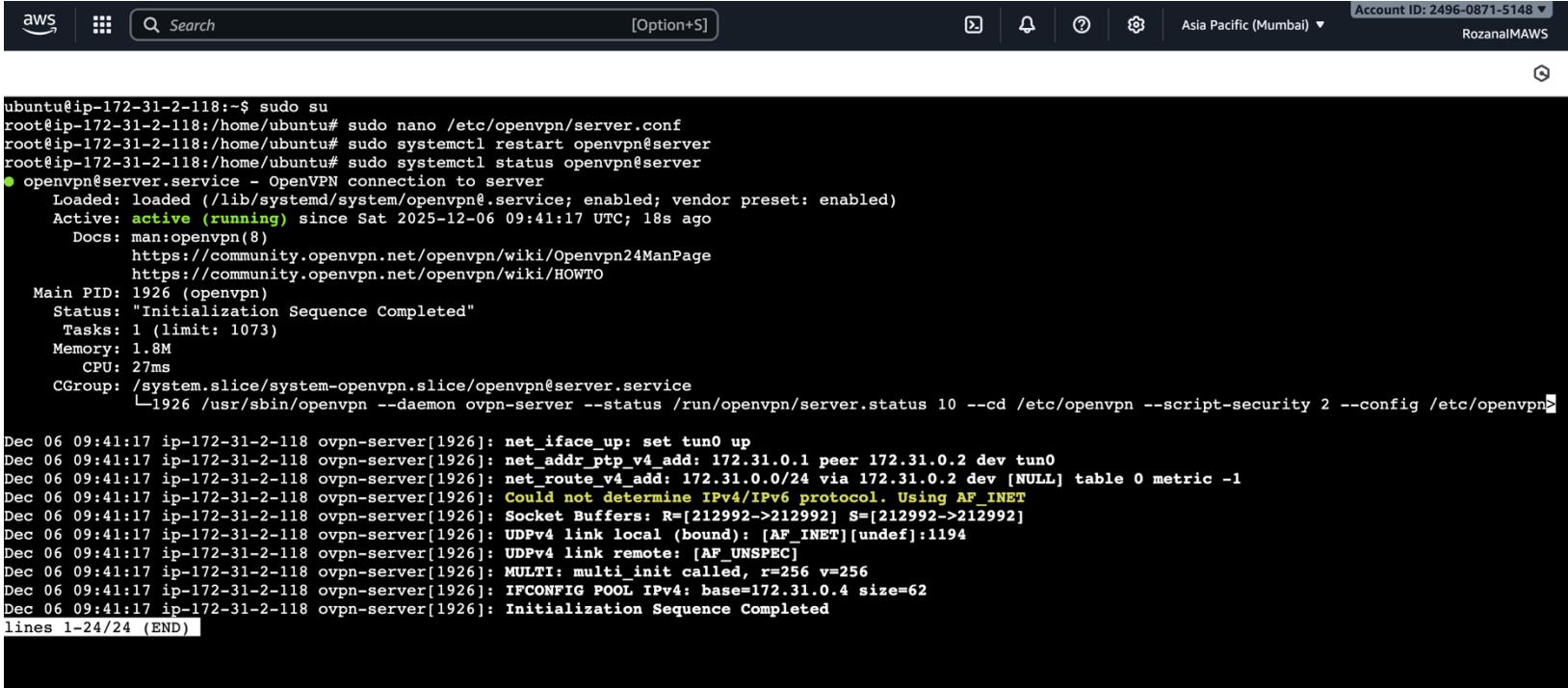
i-088a8feeb8d2ec270 (Ubuntu2)

Public IPs: 3.109.3.128 Private IPs: 172.31.15.252



Step 5: Start and enable the OpenVPN service

- Start the OpenVPN service using: **sudo systemctl start openvpn@server**
- This command launches the OpenVPN server and begins handling VPN connections.
- Enable the service to start automatically on system boot:
sudo systemctl enable openvpn@server
- These commands ensure the VPN remains active and available even after rebooting the instance.



The screenshot shows a terminal window within the AWS CloudShell interface. The terminal output is as follows:

```
ubuntu@ip-172-31-2-118:~$ sudo su
root@ip-172-31-2-118:/home/ubuntu# sudo nano /etc/openvpn/server.conf
root@ip-172-31-2-118:/home/ubuntu# sudo systemctl restart openvpn@server
root@ip-172-31-2-118:/home/ubuntu# sudo systemctl status openvpn@server
● openvpn@server.service - OpenVPN connection to server
   Loaded: loaded (/lib/systemd/system/openvpn@.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2025-12-06 09:41:17 UTC; 18s ago
     Docs: man:openvpn(8)
           https://community.openvpn.net/openvpn/wiki/Openvpn24ManPage
           https://community.openvpn.net/openvpn/wiki/HOWTO
   Main PID: 1926 (openvpn)
      Status: "Initialization Sequence Completed"
        Tasks: 1 (limit: 1073)
       Memory: 1.8M
          CPU: 27ms
        CGroup: /system.slice/system-openvpn.slice/openvpn@server.service
                  └─1926 /usr/sbin/openvpn --daemon ovpn-server --status /run/openvpn/server.status 10 --cd /etc/openvpn --script-security 2 --config /etc/openvpn

Dec 06 09:41:17 ip-172-31-2-118 ovpn-server[1926]: net_iface_up: set tun0 up
Dec 06 09:41:17 ip-172-31-2-118 ovpn-server[1926]: net_addr_ptp_v4_add: 172.31.0.1 peer 172.31.0.2 dev tun0
Dec 06 09:41:17 ip-172-31-2-118 ovpn-server[1926]: net_route_v4_add: 172.31.0.0/24 via 172.31.0.2 dev [NULL] table 0 metric -1
Dec 06 09:41:17 ip-172-31-2-118 ovpn-server[1926]: Could not determine IPv4/IPv6 protocol. Using AF_INET
Dec 06 09:41:17 ip-172-31-2-118 ovpn-server[1926]: Socket Buffers: R=[212992->212992] S=[212992->212992]
Dec 06 09:41:17 ip-172-31-2-118 ovpn-server[1926]: UDPv4 link local (bound): [AF_INET][undef]:1194
Dec 06 09:41:17 ip-172-31-2-118 ovpn-server[1926]: UDPv4 link remote: [AF_UNSPEC]
Dec 06 09:41:17 ip-172-31-2-118 ovpn-server[1926]: MULTI: multi_init called, r=256 v=256
Dec 06 09:41:17 ip-172-31-2-118 ovpn-server[1926]: IFCONFIG POOL IPv4: base=172.31.0.4 size=62
Dec 06 09:41:17 ip-172-31-2-118 ovpn-server[1926]: Initialization Sequence Completed
lines 1-24/24 (END)
```

i-0062e8cc6e6ba6d75 (ubuntu)

PublicIPs: 3.111.60.174 PrivateIPs: 172.31.2.118

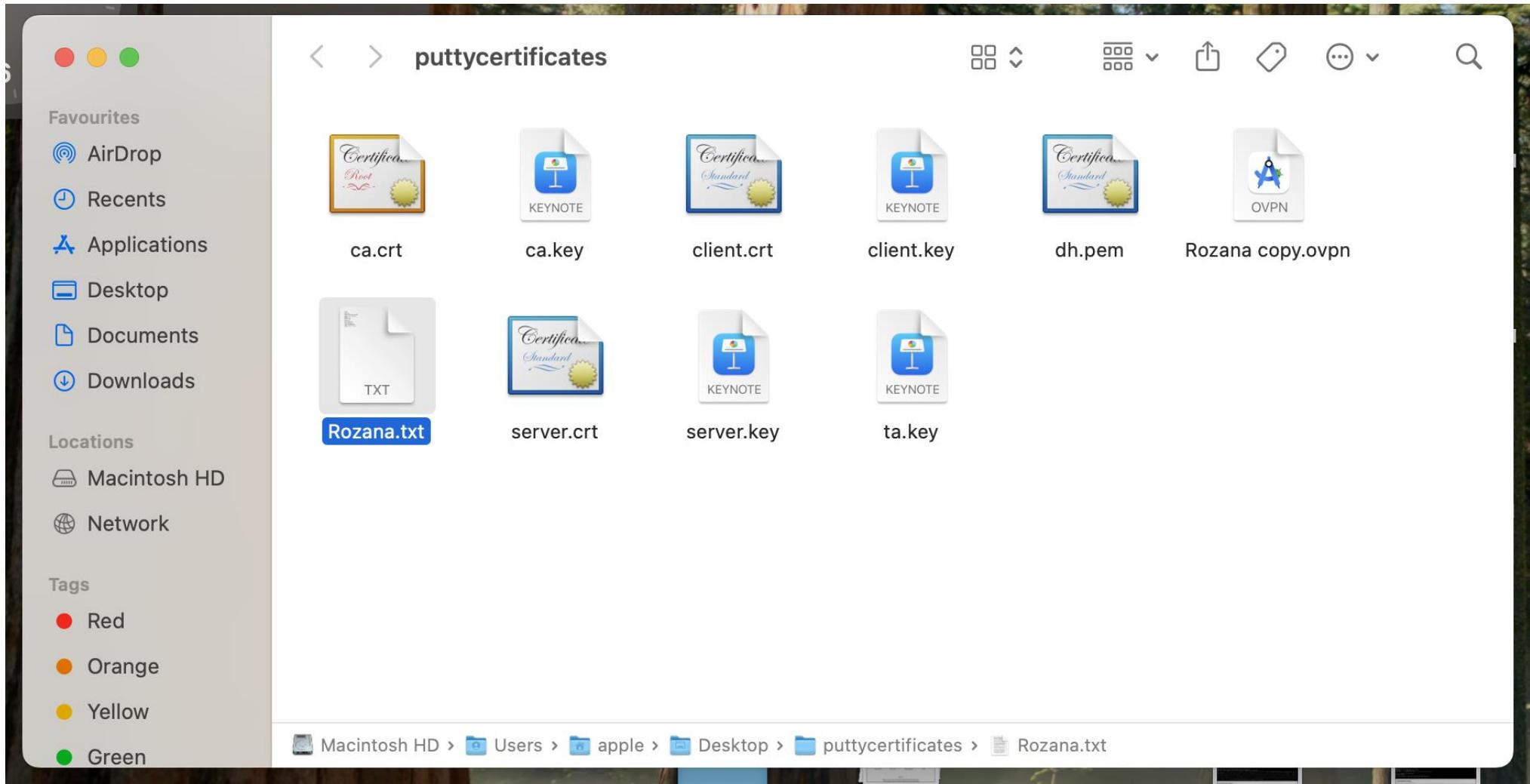
Step 6: Set up Client folder and Configuration file

- On your local system (Mac), create a new folder to store all VPN client files — for example:
mkdir openvpn-client
- Add the downloaded files into this folder using Cyberduck or manually:
 - 1.ca.crt
 - 2.client.crt
 - 3.client.key
 - 4.ta.key etc
- Inside the same folder, create a new configuration file named **client.ovpn** using any text editor.
- Add the following Configuration content :

```
client
dev tun
proto udp
remote <instance-public-ip> 1194
resolv-retry infinite
nobind
persist-key
persist-tun
ca ca.crt
cert client.crt
key client.key
tls-auth ta.key 1
cipher AES-256-CBC
remote-cert-tls server
verb 3
```



- Save the file — this client.ovpn file will be imported into the **OpenVPN client** (like Tunnelblick on macOS) for connection.
- I have saved the file name as Rozana copy.ovpn



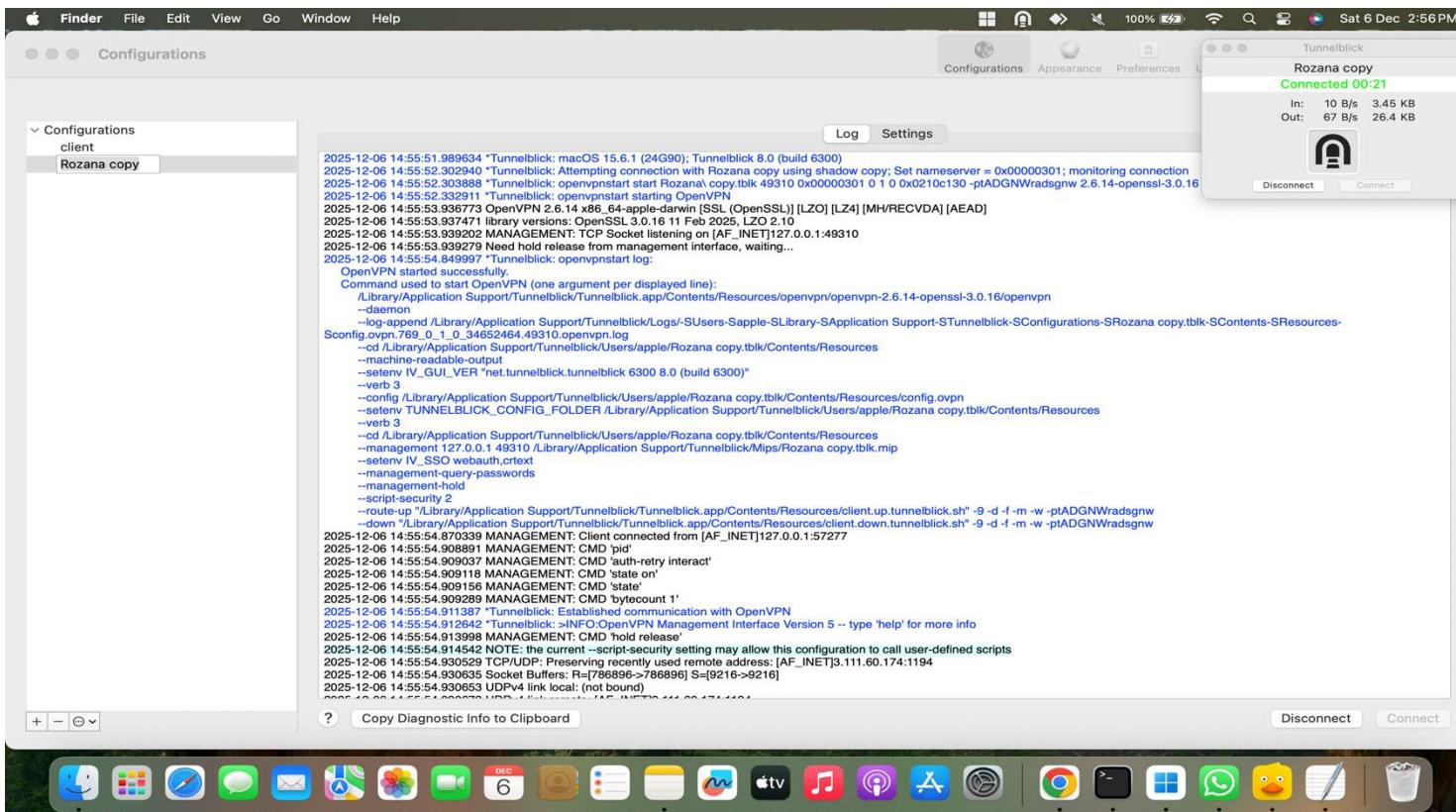
Step 7: Install Tunnelblick (OpenVPN Client for macOS)

- Option 1 – From Website
Visit  tunnelblick.net
- Download the latest .dmg file and install it manually.
- Option 2 – From Terminal (Homebrew)
Run: **brew install --cask tunnelblick**
- Open Tunnelblick from Applications after installation.



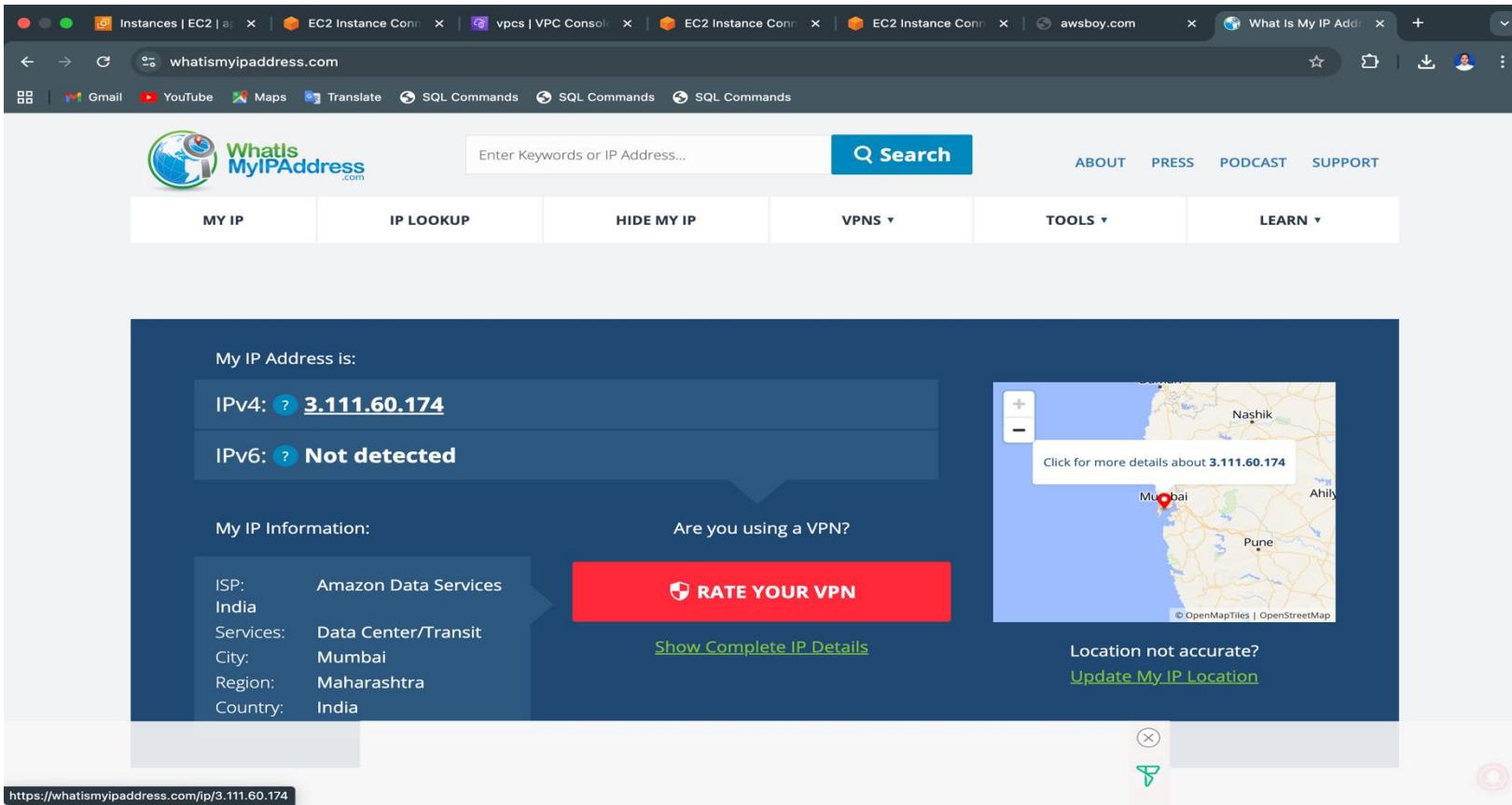
Step 8: Connect using Tunnelblick

- Open Tunnelblick — your **client.ovpn** file appears automatically in the left sidebar.
- If it doesn't appear, click "I have configuration files" → import the **client.ovpn** file manually.
- Click on the VPN profile and then select "Connect."
- Enter your Mac password if prompted for permissions.
- Once connected, a tunnel icon appears on the menu bar - indicating a successful VPN connection.



Step 9: Verify VPN Connection

- After connecting in Tunnelblick, open any browser and visit:
 <https://whatismyipaddress.com>
- You'll see your public IP changed to match the VPN server — confirming the VPN is active.



The screenshot shows a web browser window with multiple tabs open, including various AWS-related pages and the 'What Is My IP Address' website. The main content of the page displays the user's IP information:

My IP Address is:

- IPv4: [3.111.60.174](#)
- IPv6: [Not detected](#)

My IP Information:

ISP:	Amazon Data Services
India	
Services:	Data Center/Transit
City:	Mumbai
Region:	Maharashtra
Country:	India

Are you using a VPN?

[RATE YOUR VPN](#)

[Show Complete IP Details](#)

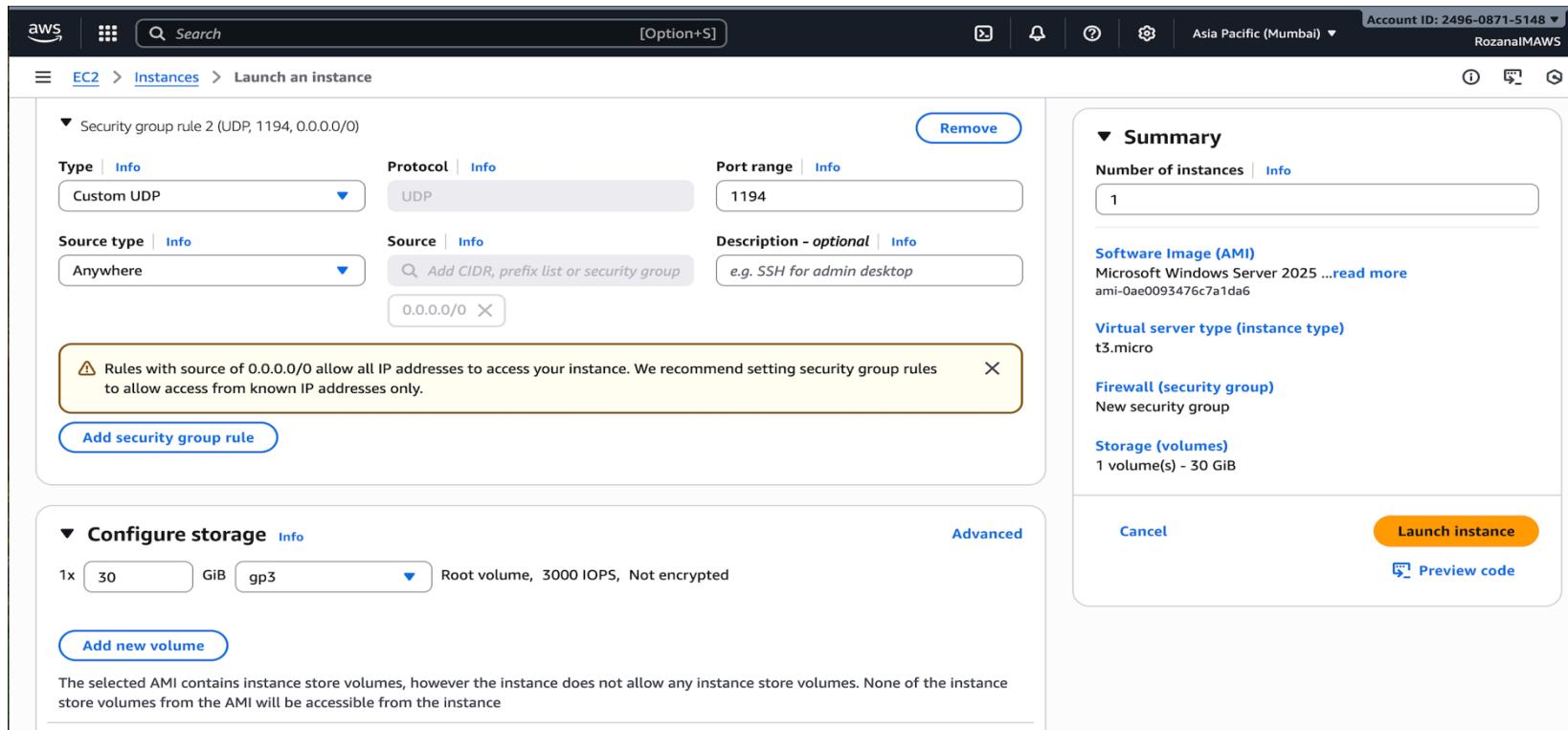
A map of the Mumbai area is shown, with a red dot indicating the location. A tooltip over the map says: "Click for more details about 3.111.60.174".

Location not accurate?
[Update My IP Location](#)

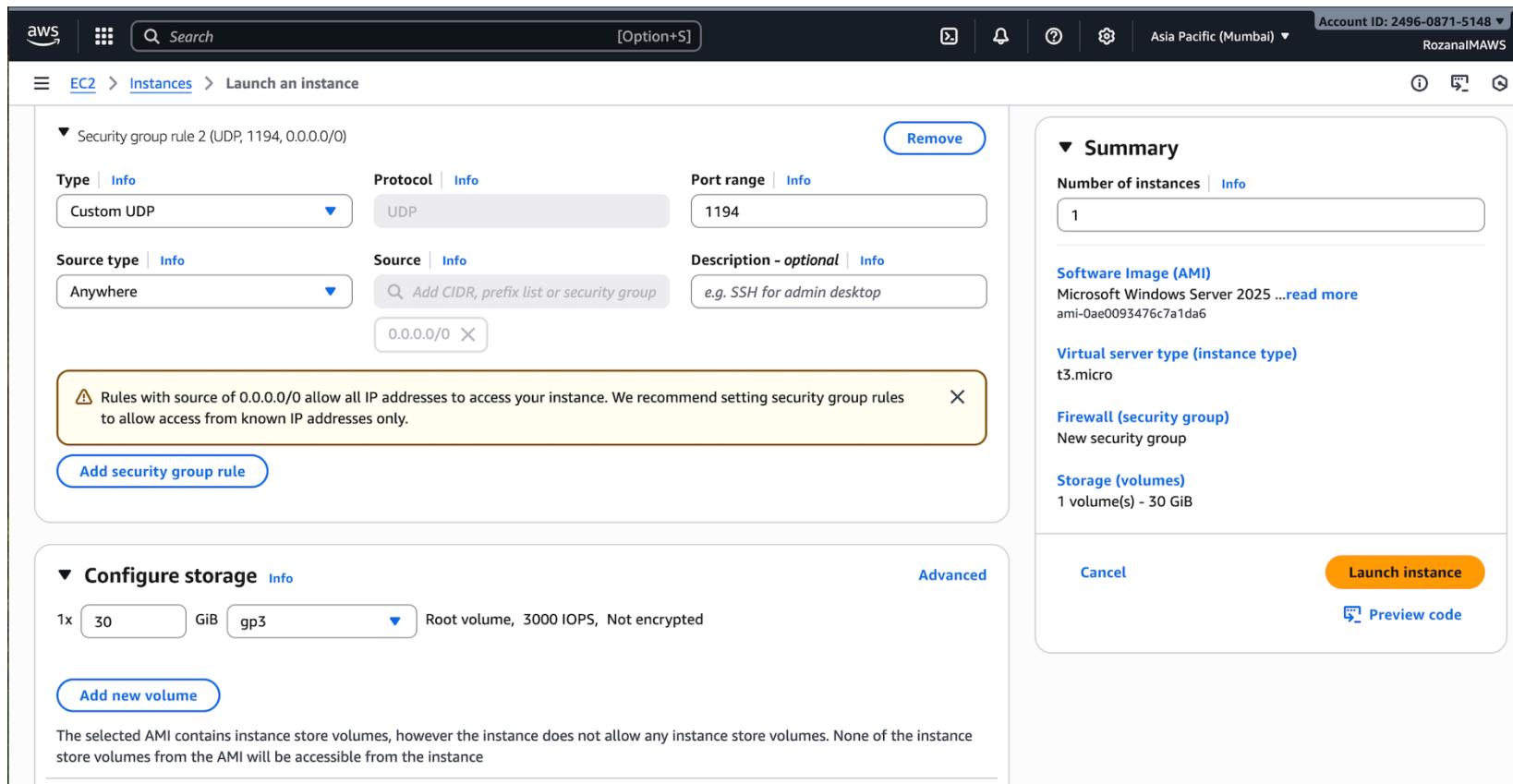
At the bottom of the page, the URL <https://whatismyipaddress.com/ip/3.111.60.174> is visible.

Step 1 : Launch a Private Windows Instance and Configure Security Groups

- Login to the AWS Management Console and navigate to the EC2 Dashboard.
- Click “Launch Instance” and select a Windows Server AMI as the operating system.
- Choose an appropriate instance type (e.g., t3.micro) and continue with the setup.
- Under Network Settings, select the same VPC used for the VPN instance, but choose the private subnet (no public IP) to ensure internal access only.



- Configure a Security Group that allows:
 - RDP (TCP, Port 3389) → for remote desktop access
 - Custom UDP (Port 1194) → to allow VPN traffic
- Launch the instance — this will create a private Windows server accessible only through the VPN connection.



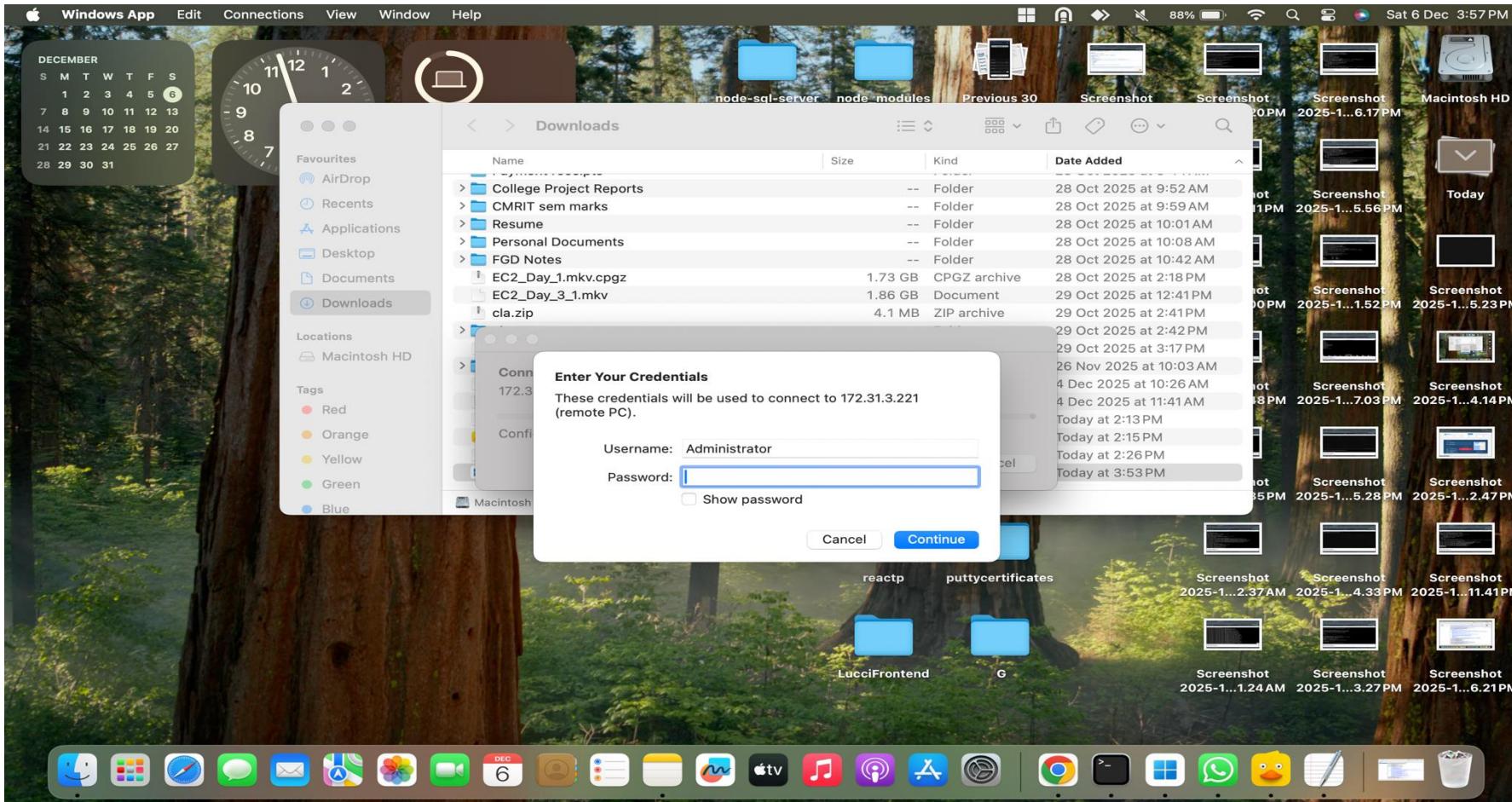
Step 3: Connect to the Windows Instance via RDP

- In the AWS EC2 Console, select your running Windows instance and click on “Connect.”
- Choose the RDP Client tab, then click “Get Password.”
- Upload your .pem key file (used during instance creation) to decrypt and reveal the Administrator password.
- Click “Download Remote Desktop File” to save the .rdp file on your system.

The screenshot shows the AWS EC2 Connect interface. At the top, there's a navigation bar with the AWS logo, search bar, and account information (Account ID: 2496-0871-5148, Asia Pacific (Mumbai), RozanalMAWS). Below it, the URL is EC2 > Instances > i-0ecd0180ea50e52ed > Connect to instance. The main area is titled "Connect" with an "Info" link. It says "Connect to an instance using the browser-based client." There are three tabs: "Session Manager" (disabled), "RDP client" (selected, highlighted in blue), and "EC2 serial console". A message box says "Record RDP connections" and "You can now record RDP connections using AWS Systems Manager just-in-time node access." Below this, under "Connection Type", "Connect using RDP client" is selected (radio button checked). A note says "Download a file to use with your RDP client and retrieve your password." Another note for "Connect using Fleet Manager" is present. At the bottom, there's a link to "Download remote desktop file" and a note about connecting using a username and password. On the right, there's a "Username" dropdown set to "Administrator".

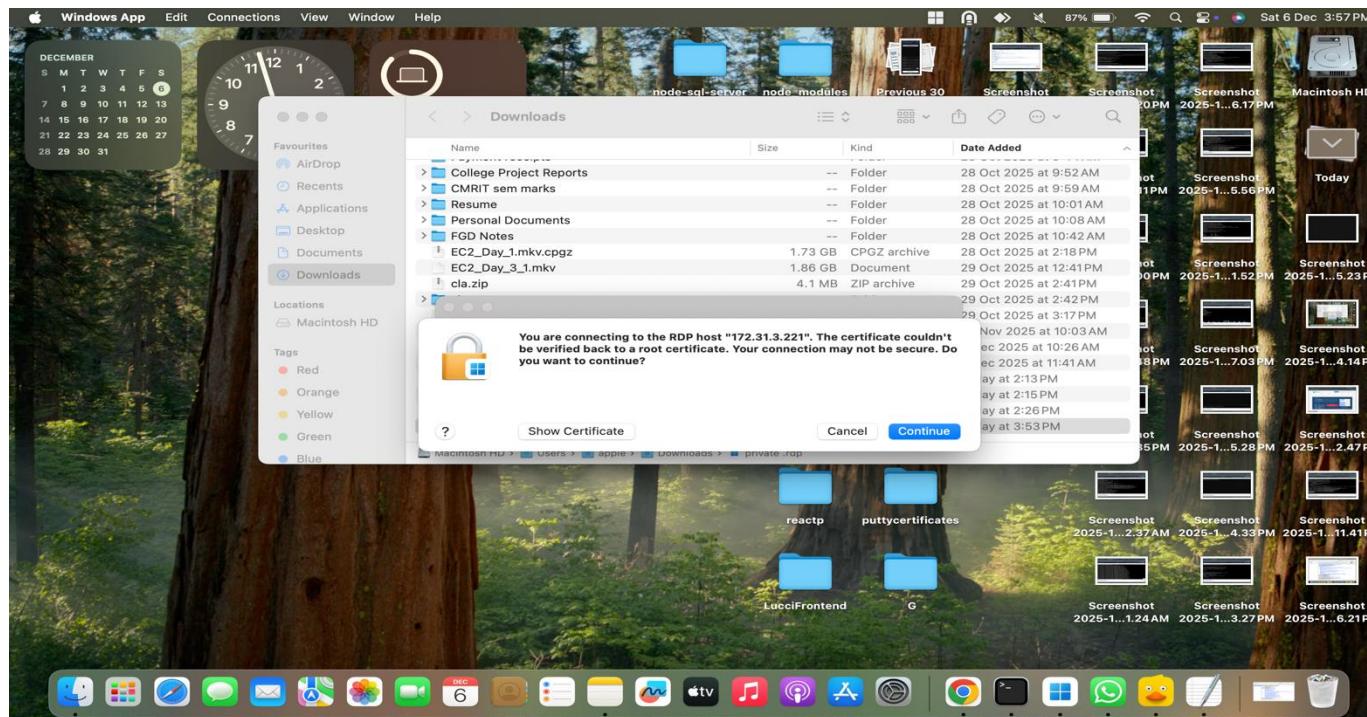
The screenshot shows the "Get Windows password" interface. The URL is EC2 > Instances > i-0ecd0180ea50e52ed > Get Windows password. The title is "Get Windows password" with an "Info" link. It says "Use your private key to retrieve and decrypt the initial Windows administrator password for this instance." Under "Instance ID", "i-0ecd0180ea50e52ed (private)" is listed. Under "Key pair associated with this instance", "my2ndkey" is listed. Under "Private key", it says "Either upload your private key file or copy and paste its contents into the field below." There's a "Upload private key file" button and a "Private key contents" text area. At the bottom right, there are "Cancel" and "Decrypt password" buttons.

- Open the downloaded file — it will prompt for the Hostname (Public IPv4 Address) and Password.
- Enter the credentials and click “Connect.”
- A new Remote Desktop window opens, giving full access to your Windows server instance.



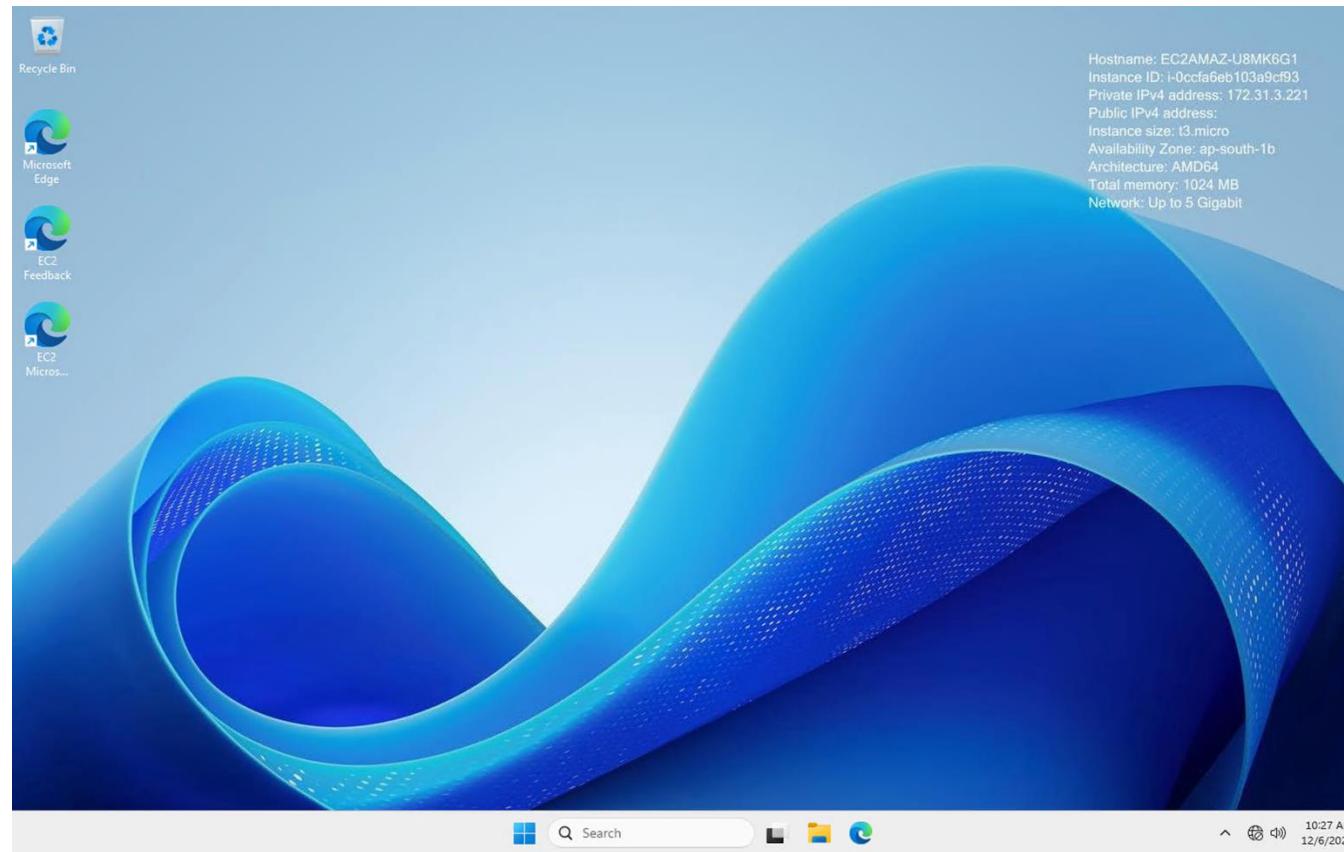
Step 4: Connection Failed Without VPN

- Attempting to open the downloaded .rdp file and connect to the private Windows instance without turning on the VPN will fail.
- Since the instance is launched in a private subnet, it has no public IP, meaning it cannot be reached directly over the internet.
- This error confirms that the VPN tunnel must be active to securely access resources inside the private subnet.
- The screenshot below shows the failed connection attempt before enabling the VPN.



Step 5: Successful Connection After Enabling VPN

- Once the VPN connection is established, the private Windows instance becomes accessible through the secure tunnel.
- Open the .rdp file, enter the private IP and password, then click Connect.
- The Remote Desktop window opens successfully. This confirms that the VPN is functioning correctly, providing secure internal access to private AWS resources.



THANK YOU

– Rozana I M

