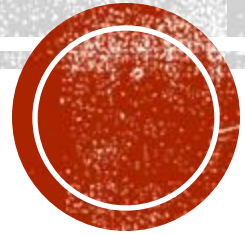
CROSS-PLATFORM VPN DEPLOYMENT ON AWS USING MACOS



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*.

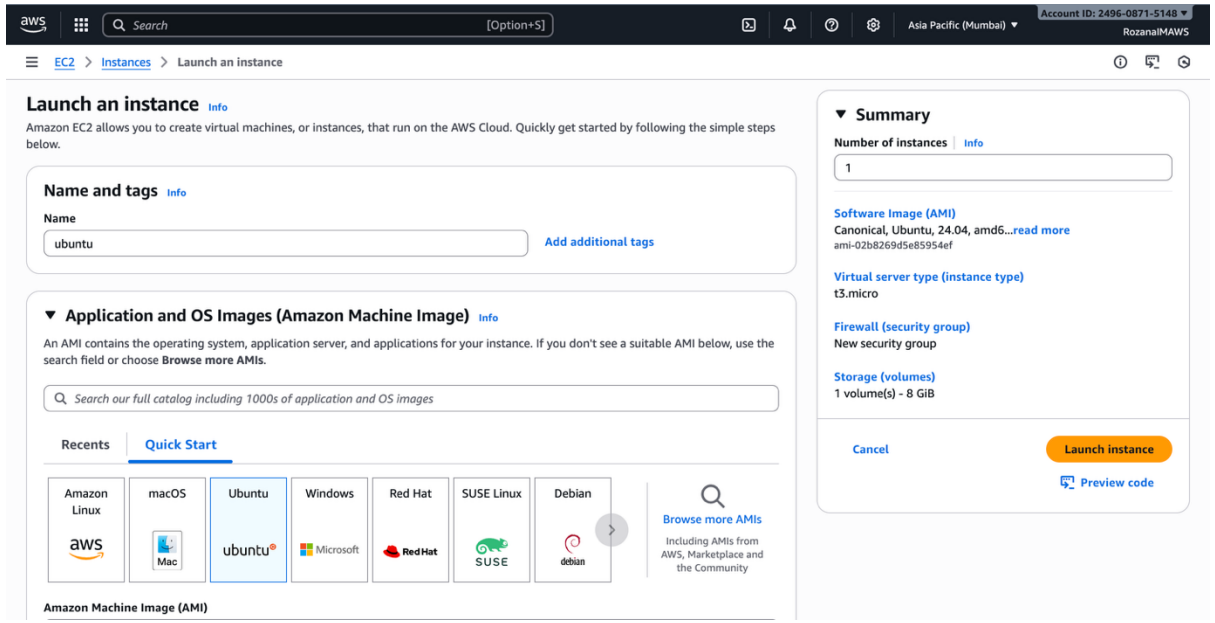
Tools and Technologies

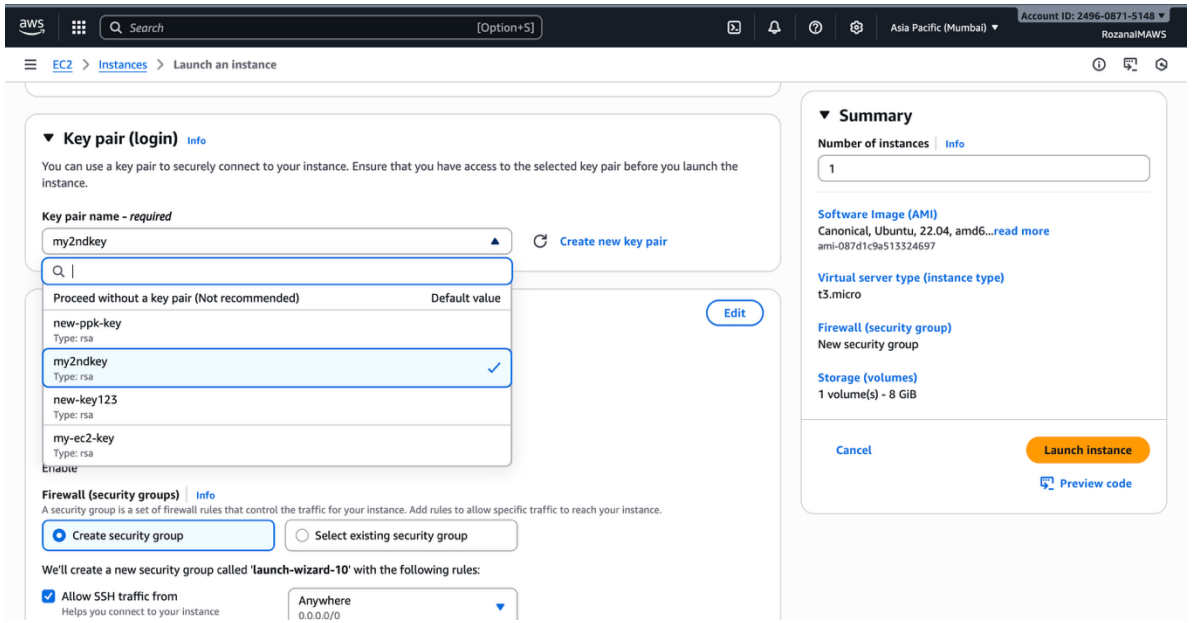
Category	Tools / Technologies
Cloud Platform	AWS EC2
Operating System	Ubuntu 22.04 LTS (Server), macOS (Client)
VPN Software	OpenVPN
PKI Tool	Easy-RSA
File Management	Cyberduck
Remote Access	AWS EC2 Instance Connect
VPN Client	Tunnelblick
Verification Tool	WhatIsMyIP.com

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.

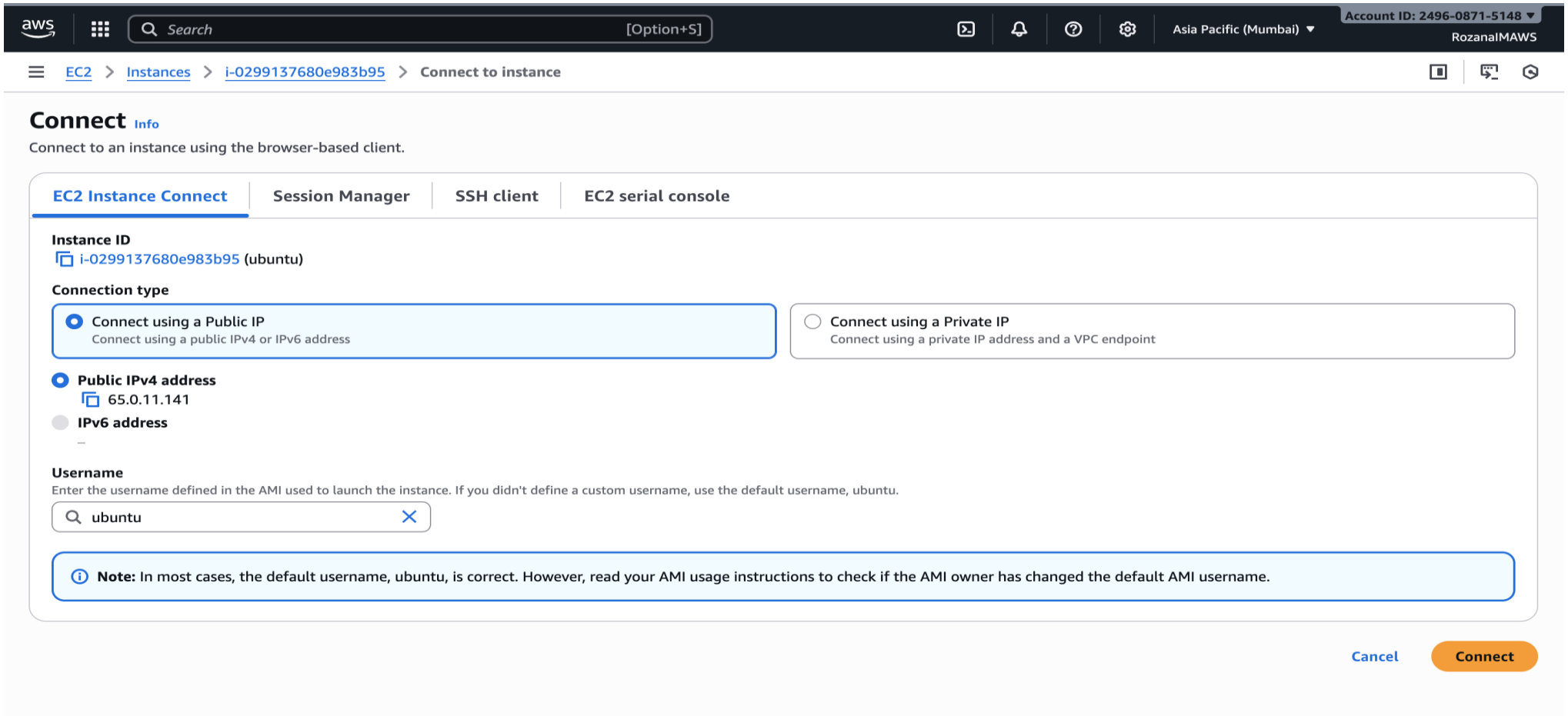
LAUNCHING AN UBUNTU INSTANCE





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.



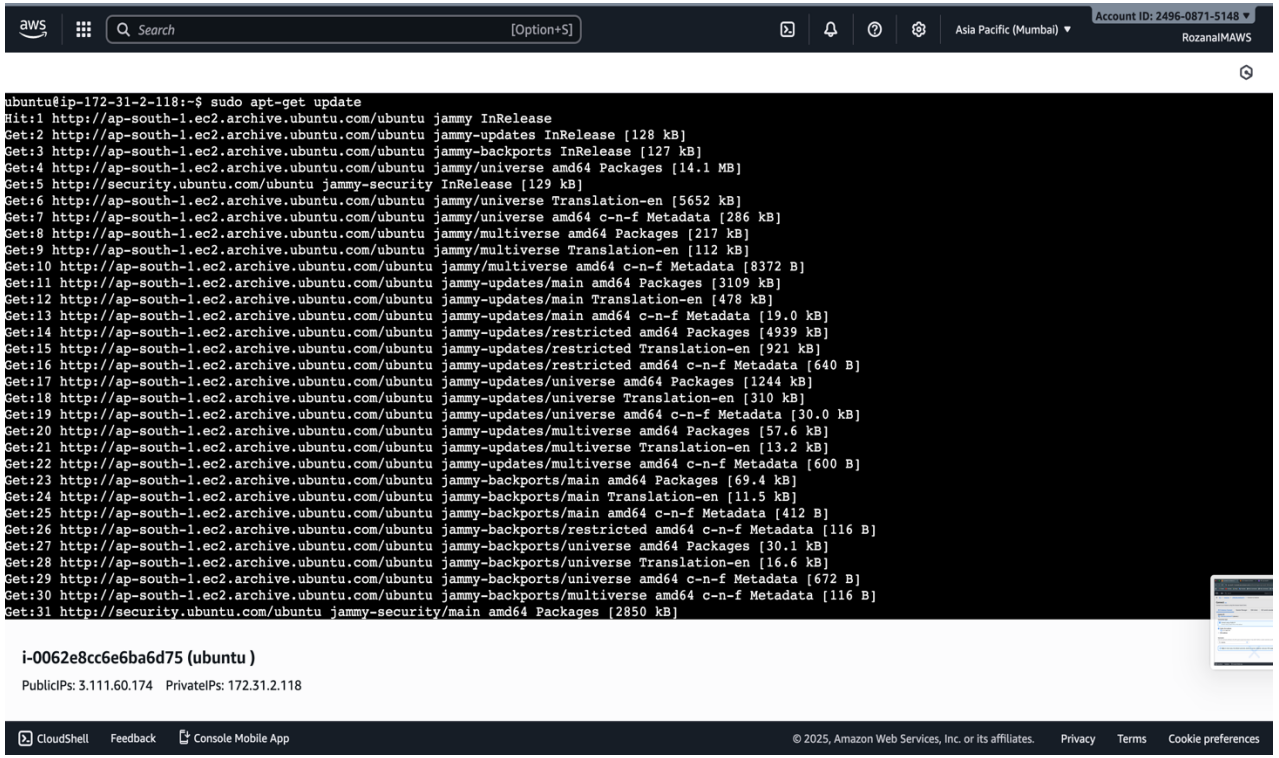
Step 3: Setting up and Configuring openvpn

3.1 Update Packages and Install OpenVPN & Easy-RSA

- Once the terminal session is open, update the package list using the command:

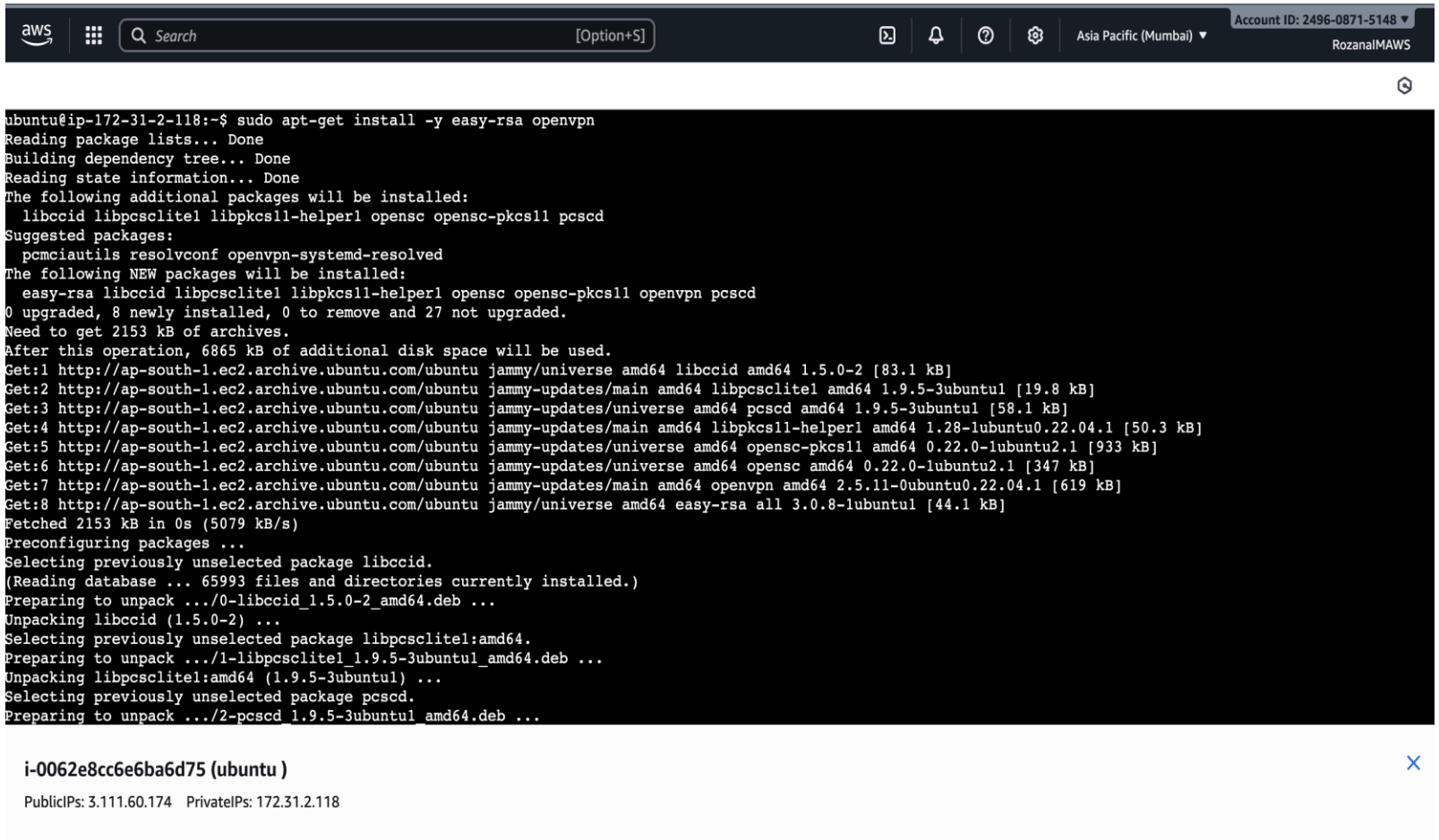
  sudo apt-get update

- This ensures your server retrieves the latest package information from the repositories.



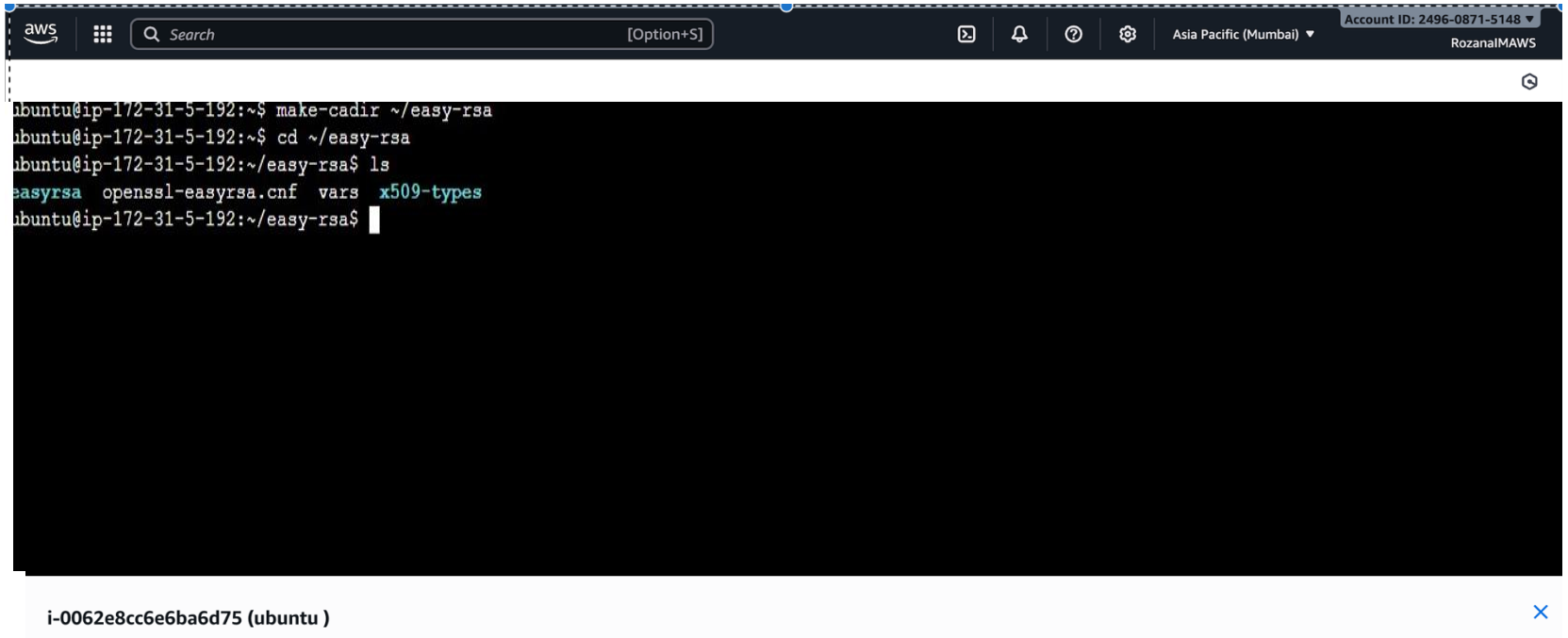
- Next, install the required tools with the command:

  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.



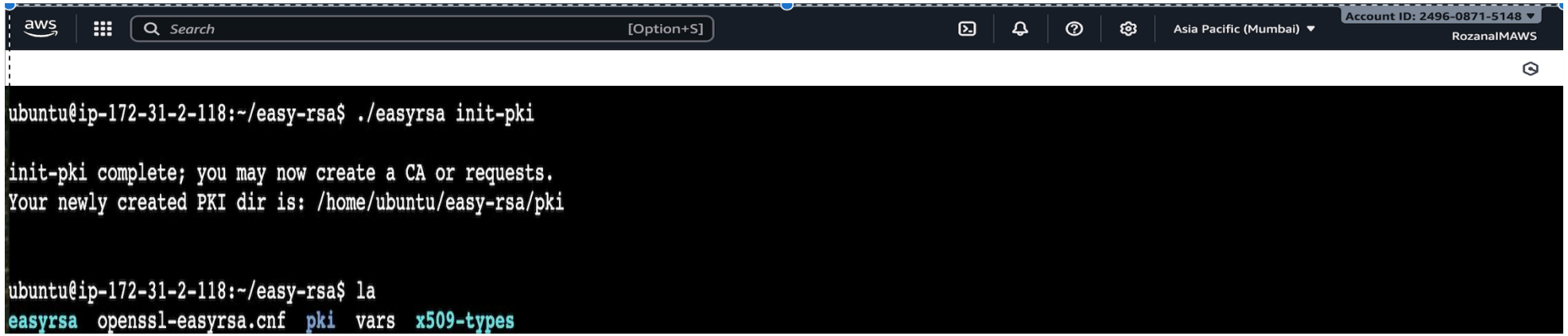
3.2 Set up the Easy- RSA Directory

- 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.



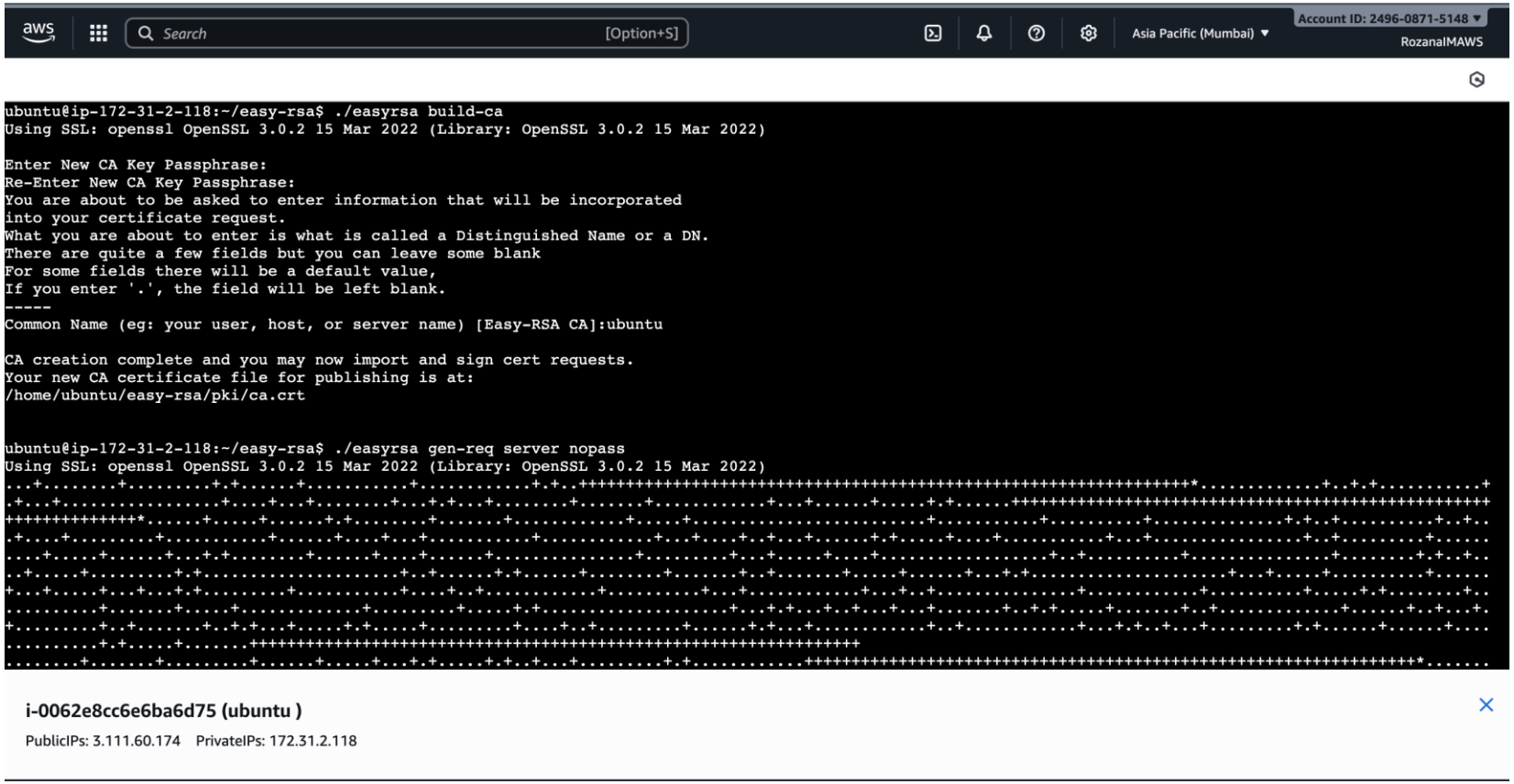
3.3 Initialize PKI and Build the Certificate Authority (CA)

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



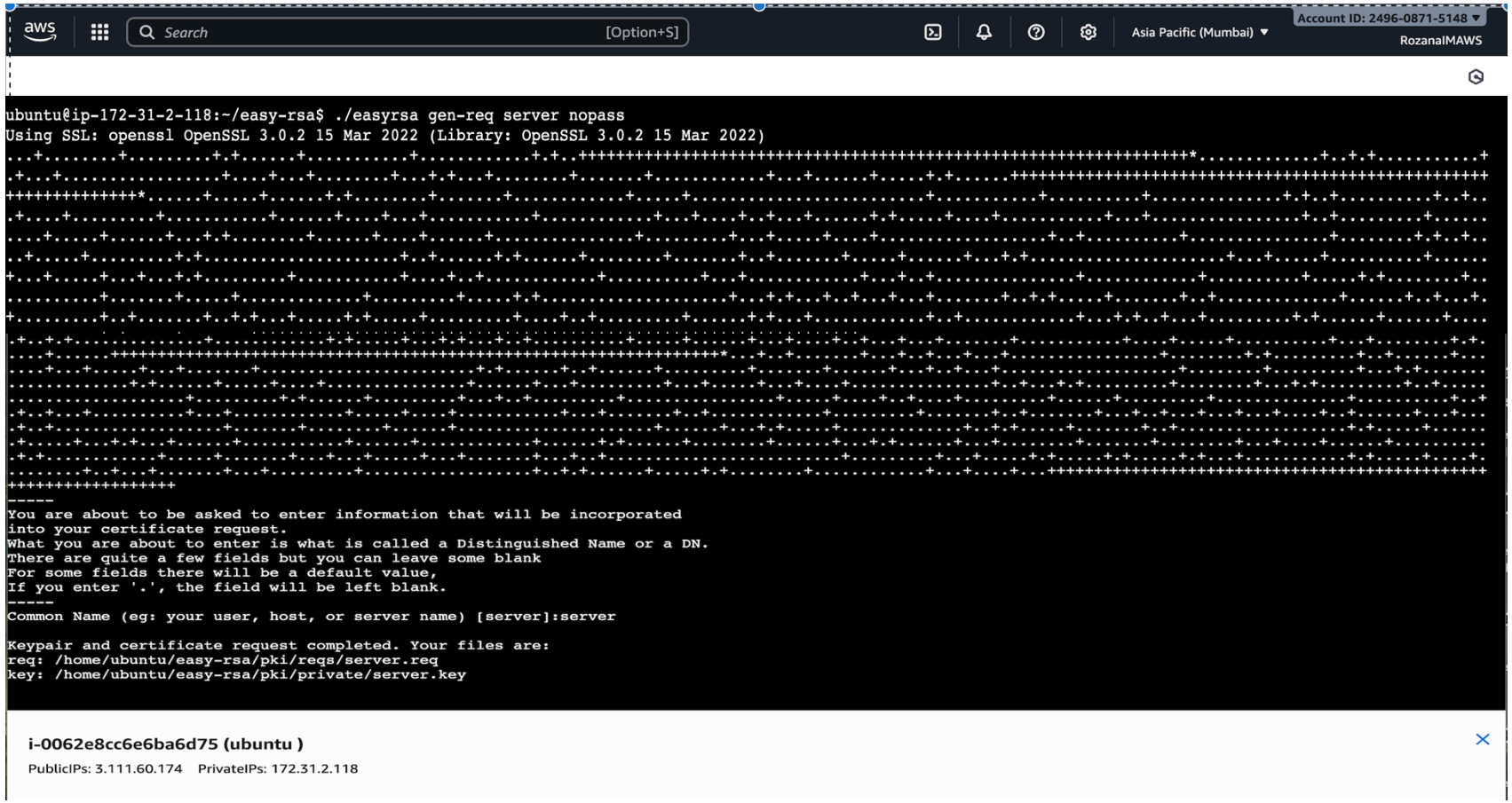
- Next, build the Certificate Authority using: **./easyrsa build-ca**
- When prompted, provide the CA passphrase and enter a Common Name (e.g., "ubuntu / OpenVPN-CA etc ") to generate the root certificate.

- 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.



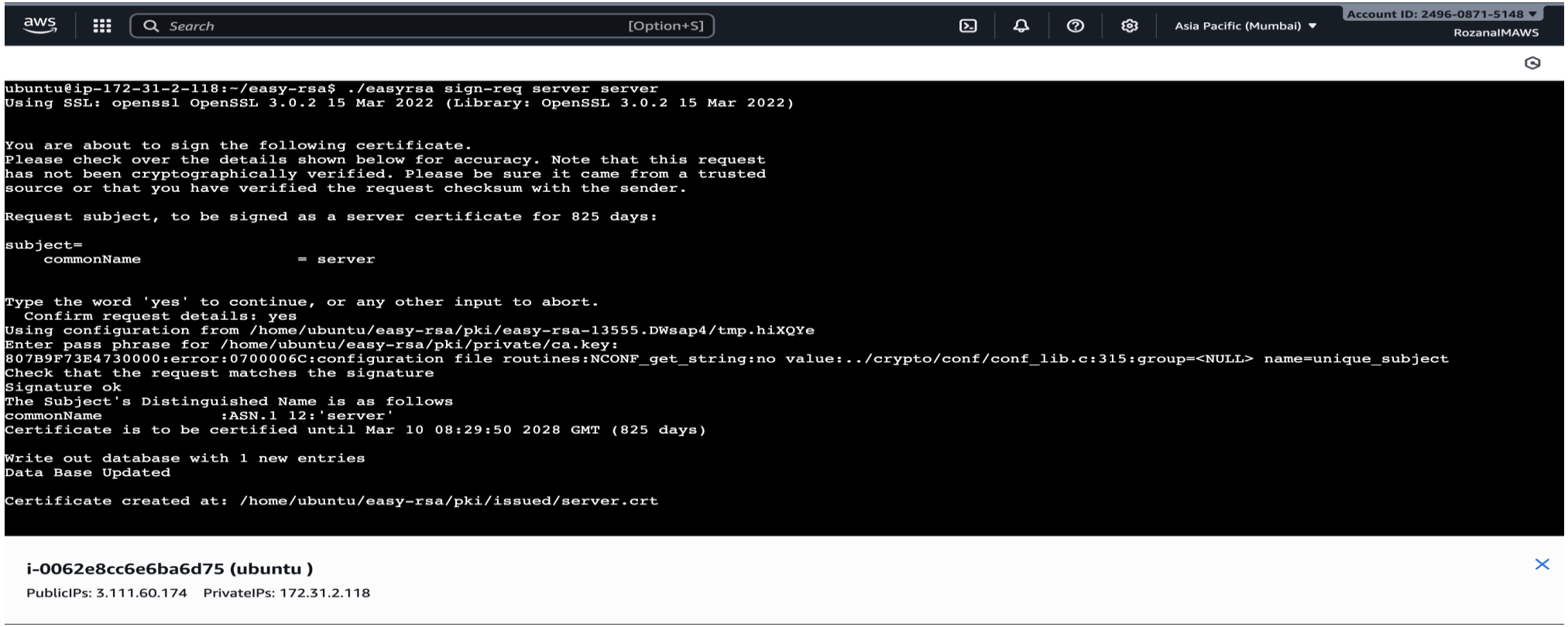
3.4 Generate the Server Certificate Request

- Generate a server certificate request by running: **./easyrsa 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.



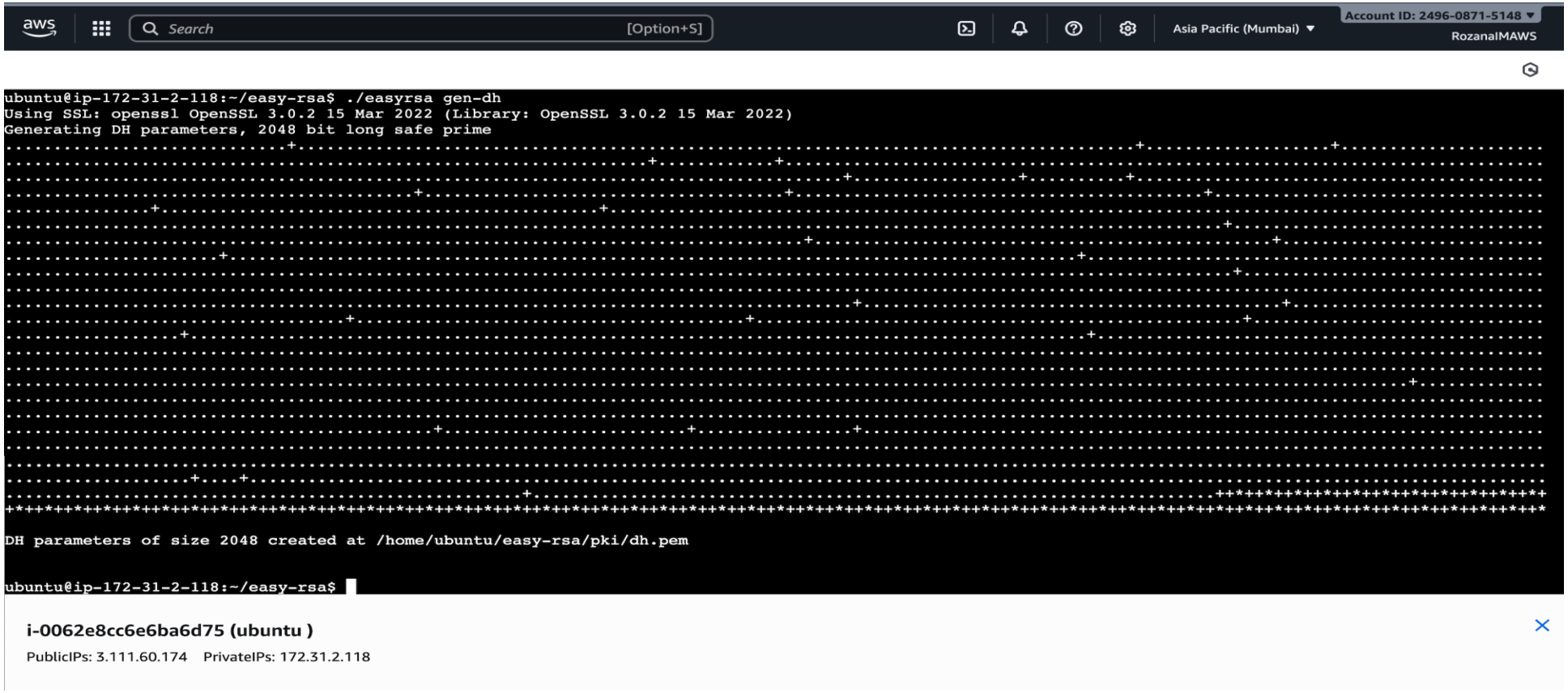
3.5 Sign the Server Certificate Request

- 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



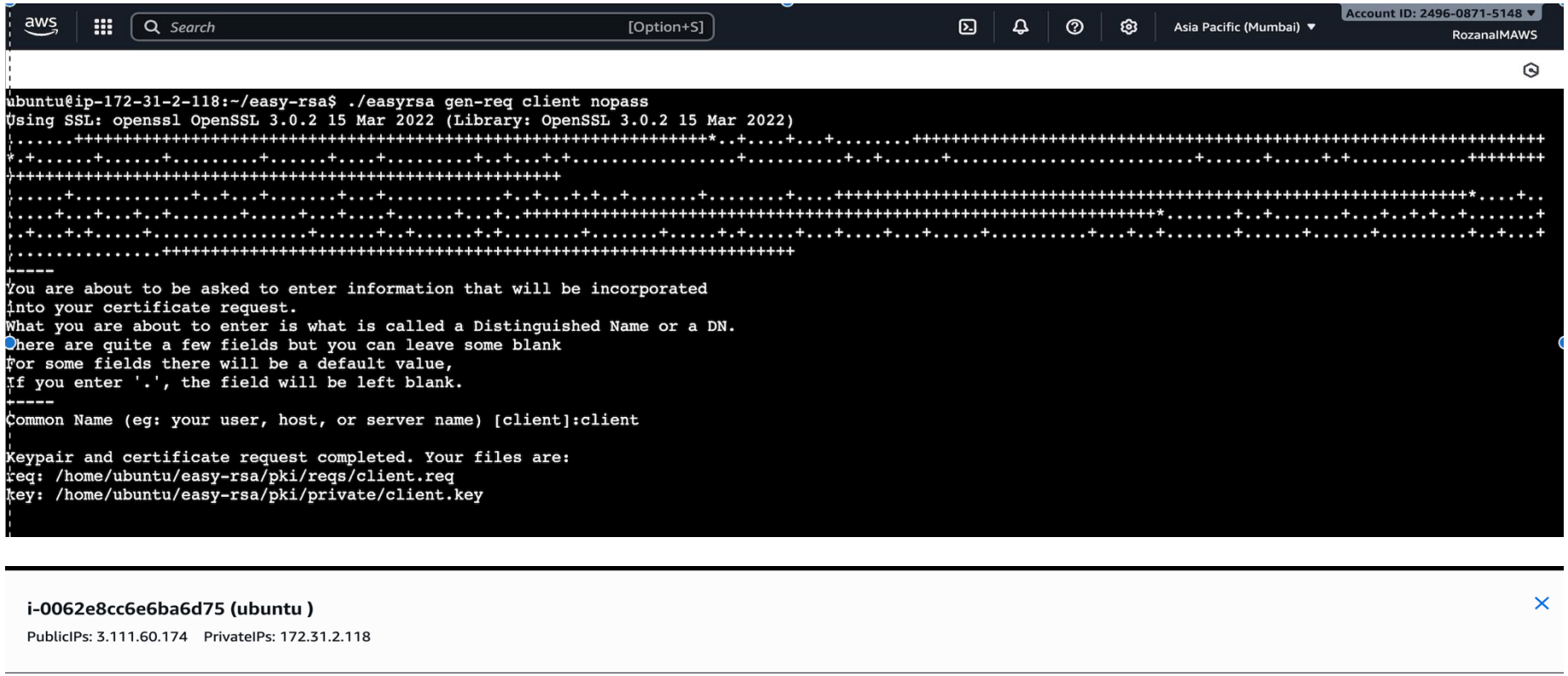
Step 3.6 : Generate Diffie-Hellman (DH) Parameters

- 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.



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.

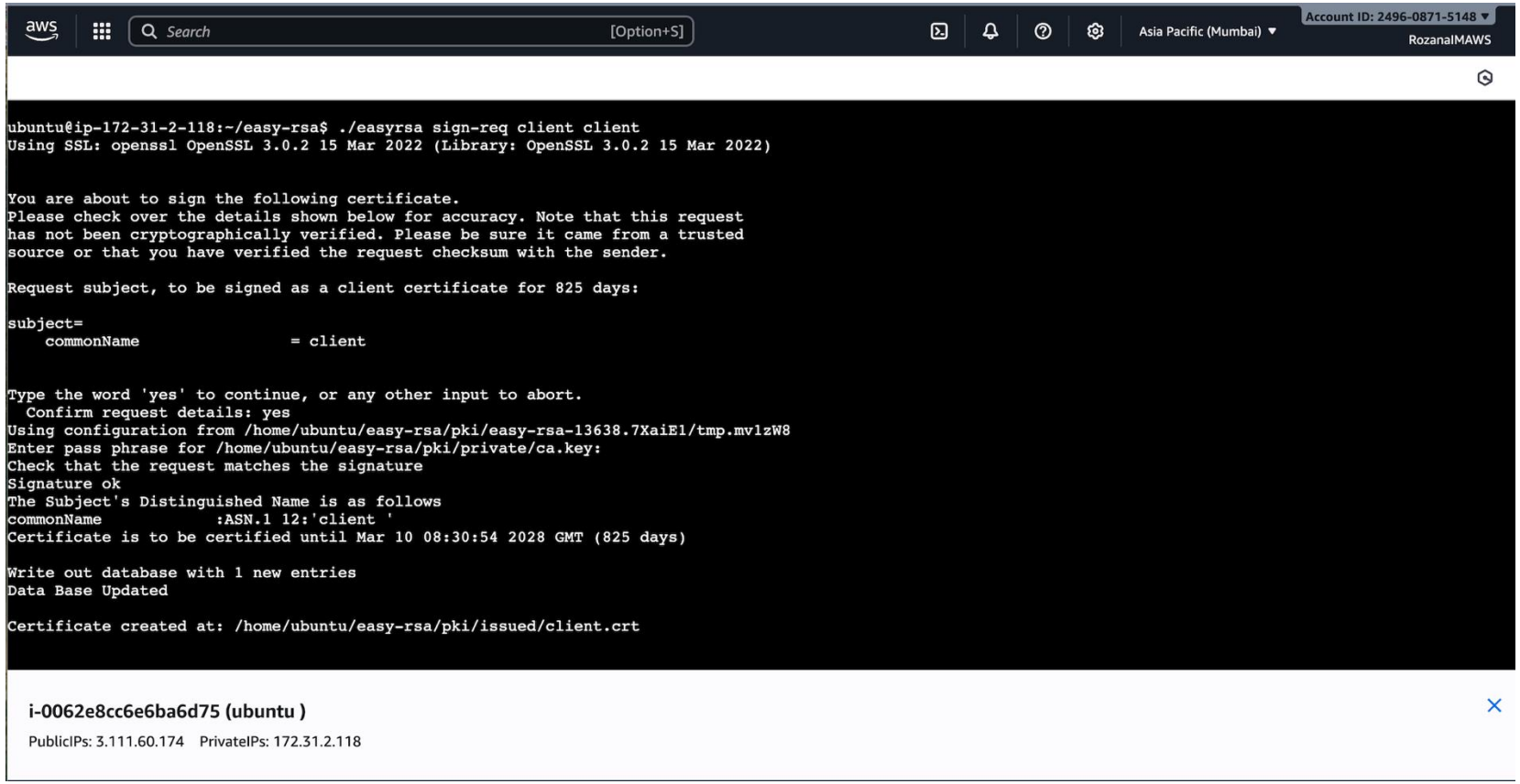


- Sign the client certificate request with the established Certificate Authority (CA):

  ./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

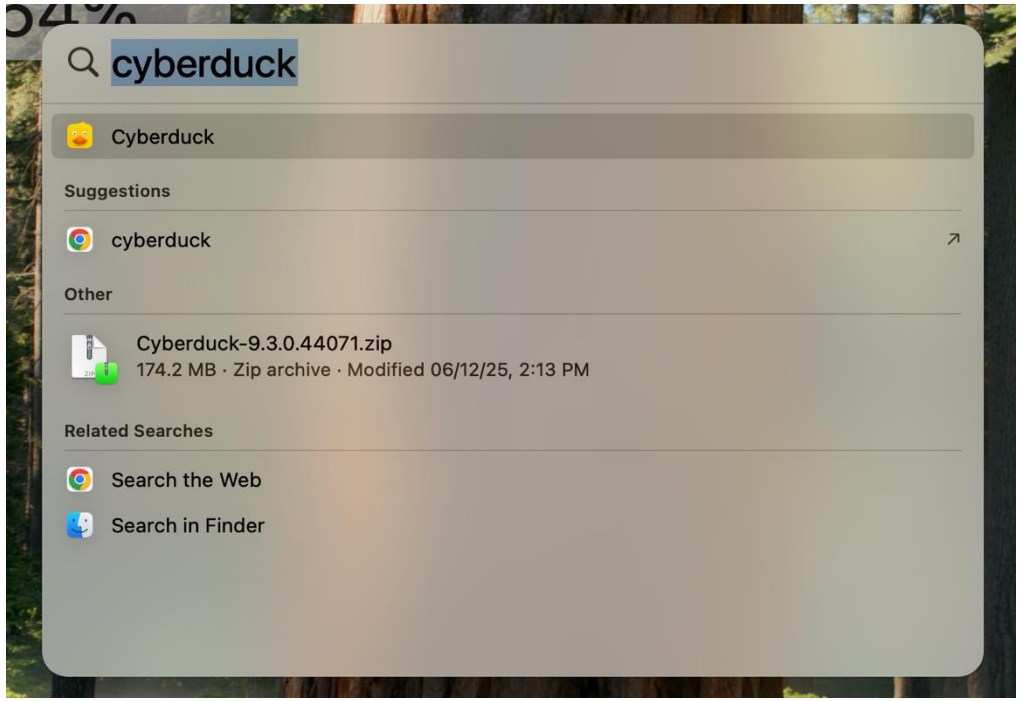


Step 3.8 : Verify PKI File Creation and Directory Structure

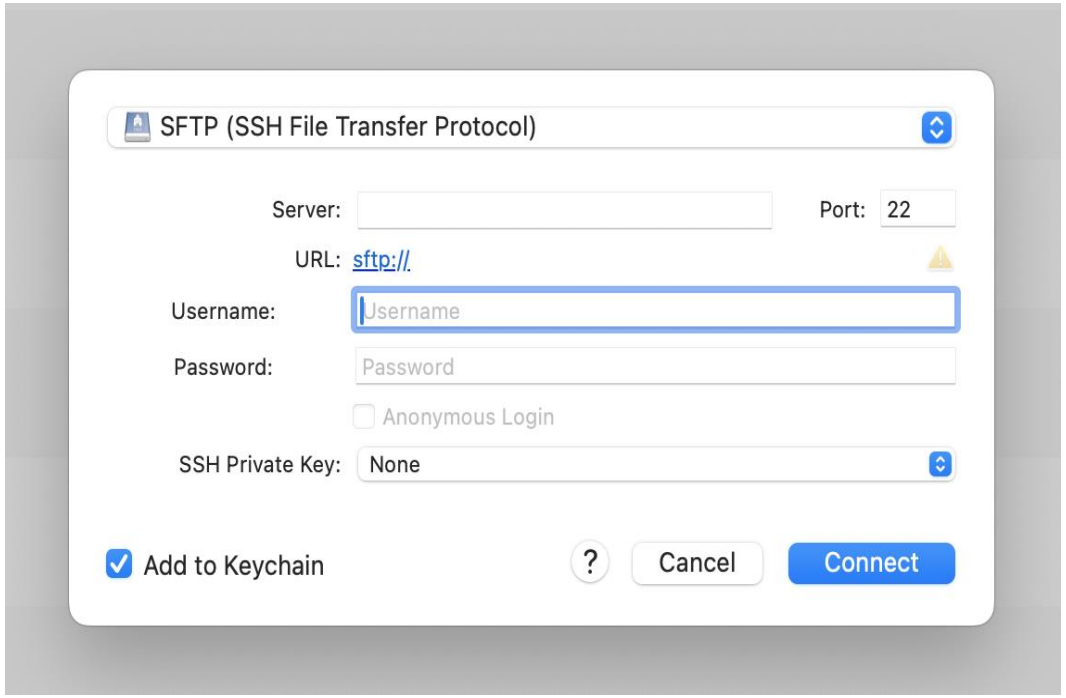
- List the contents of the easy-rsa directory using: ls
- This confirms that all required files — including the CA certificate, server certificate, client certificate, and Diffie-Hellman parameters — have been generated successfully.
- The presence of these files in the appropriate folders (e.g., /pki/issued/,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

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

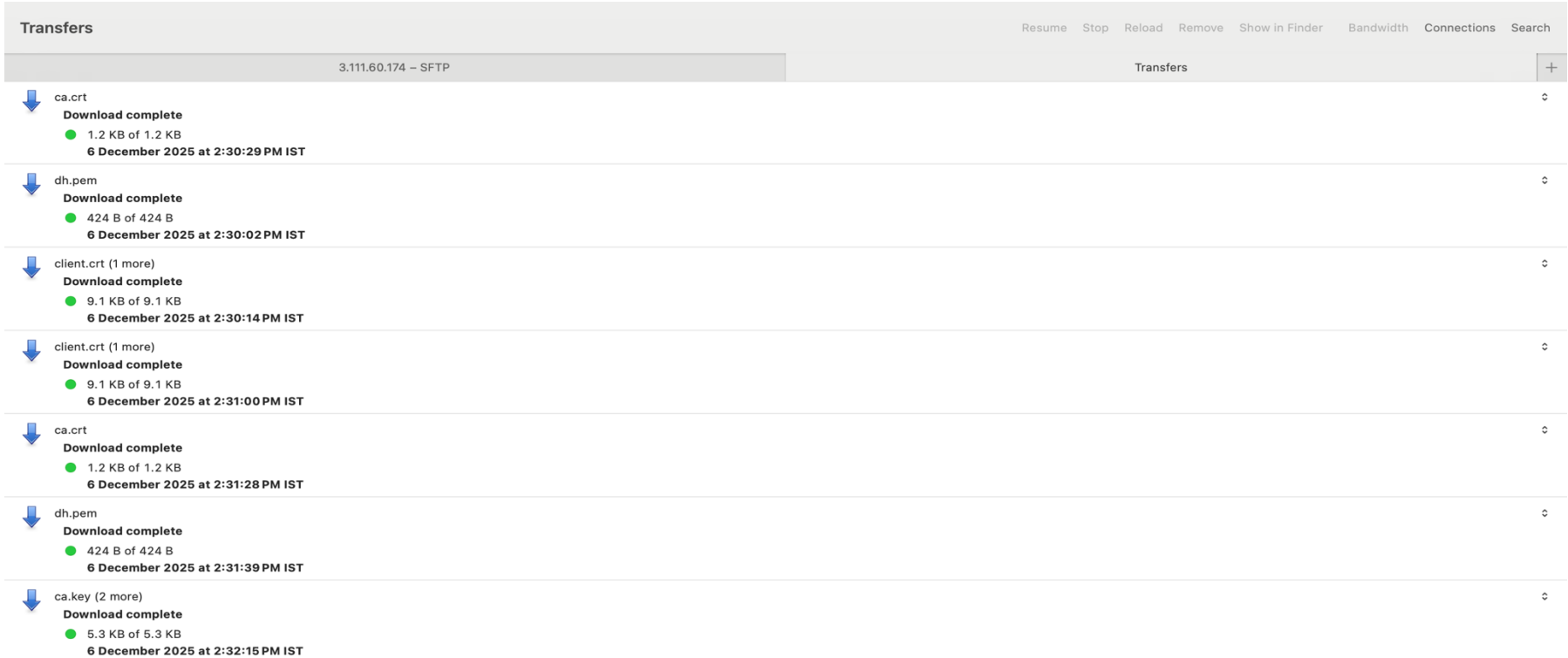
- Open Cyberduck on your macOS for Secure File Transfer.
- Some applications like Cyberduck or PuTTY require a .ppk key instead of a .pem file.
- Convert the PEM key to PPK format using PuTTYgen with the following command(in terminal):
  puttygen /Users/apple/Downloads/my-ec2-key.pem -o /Users/apple/Downloads/my-ec2-key.ppk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



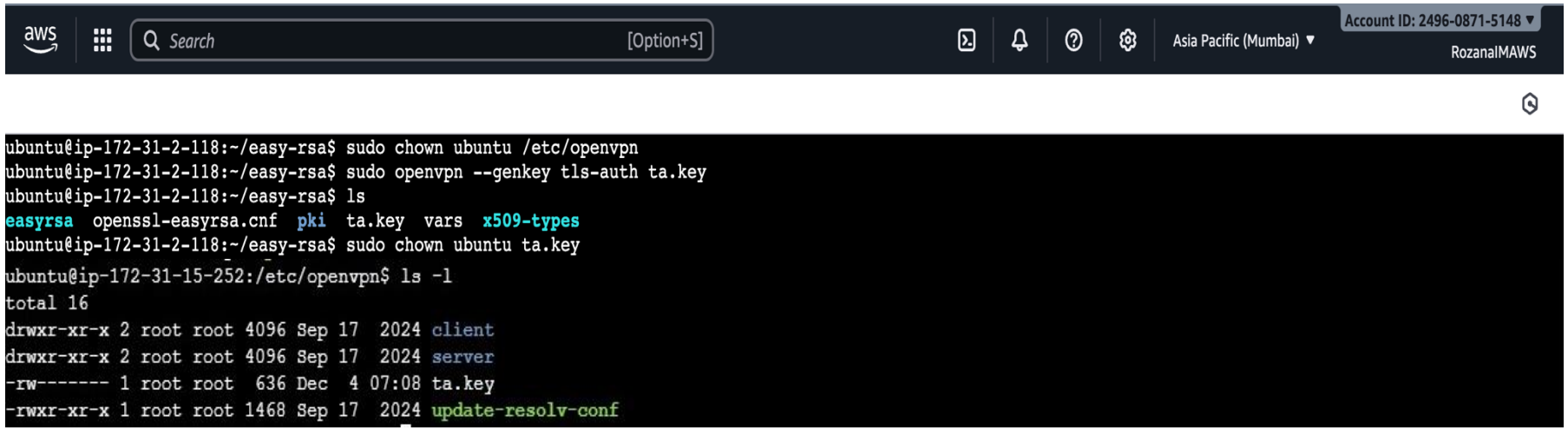
OpenVPN Server Configuration and Network Setup

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

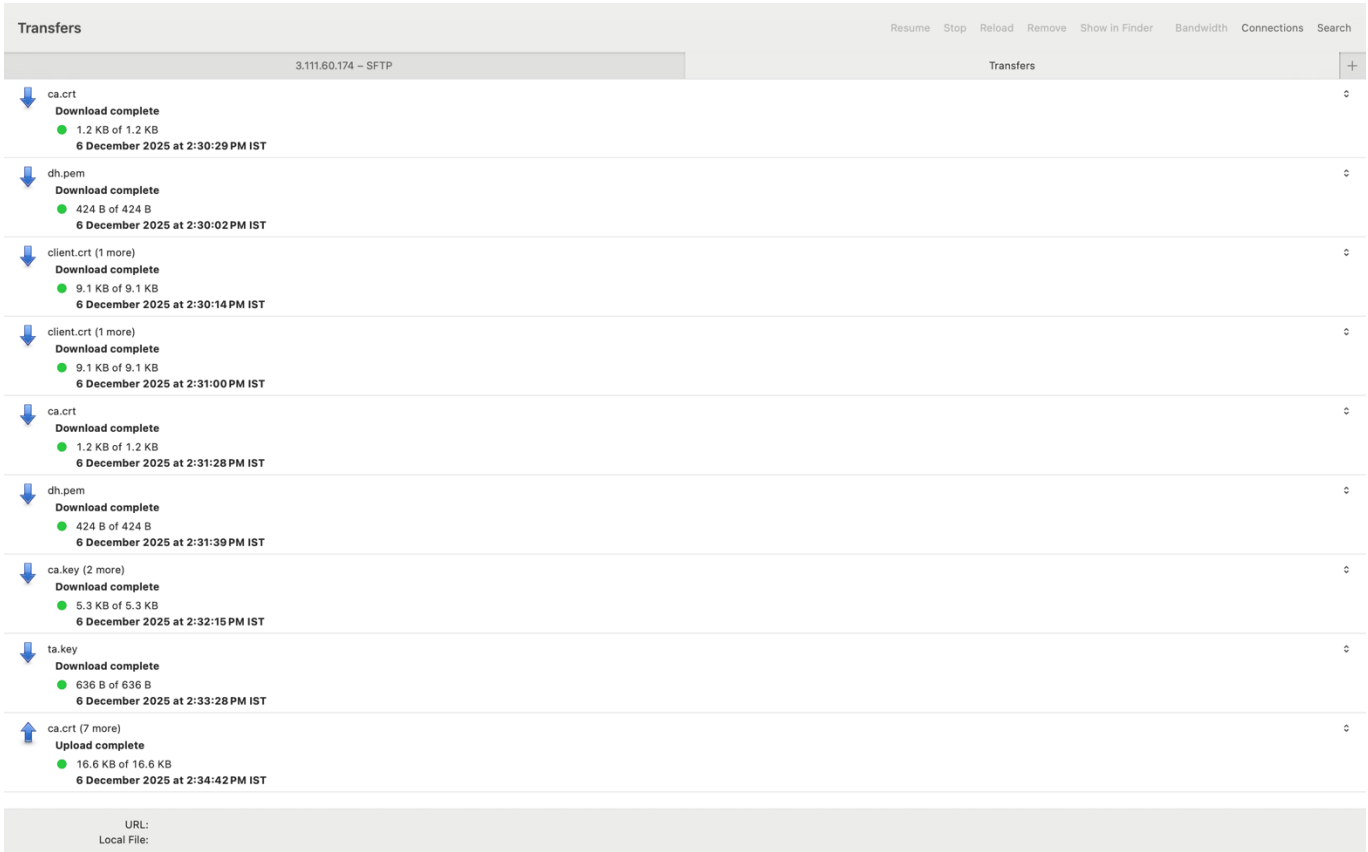
- As OpenVPN is already installed, proceed to create the TLS authentication key by running:

  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.



- 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

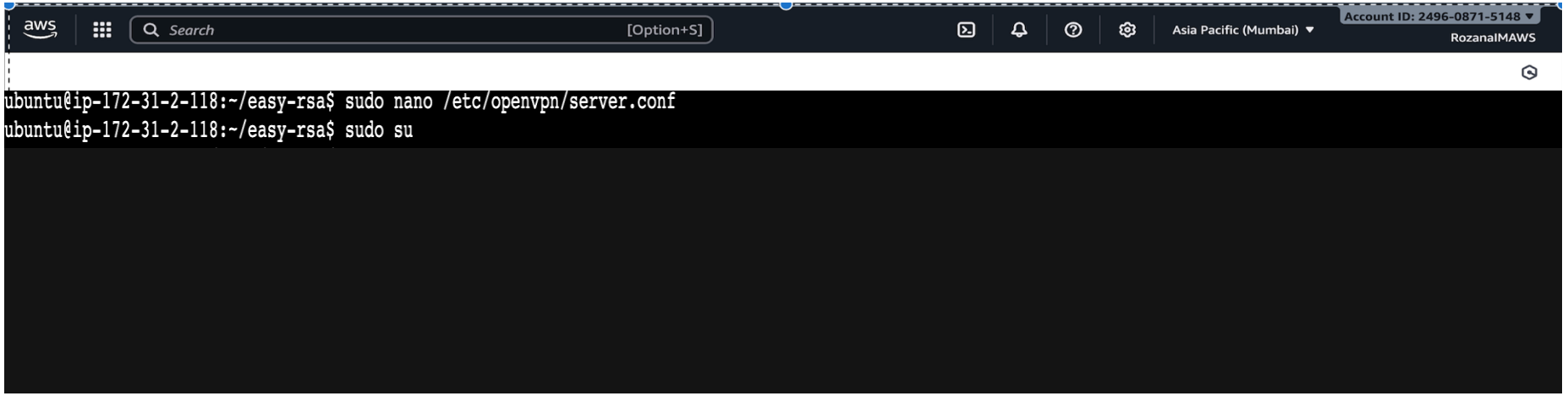
- Copy all the required VPN certificates and keys to the OpenVPN configuration directory using:

```
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.

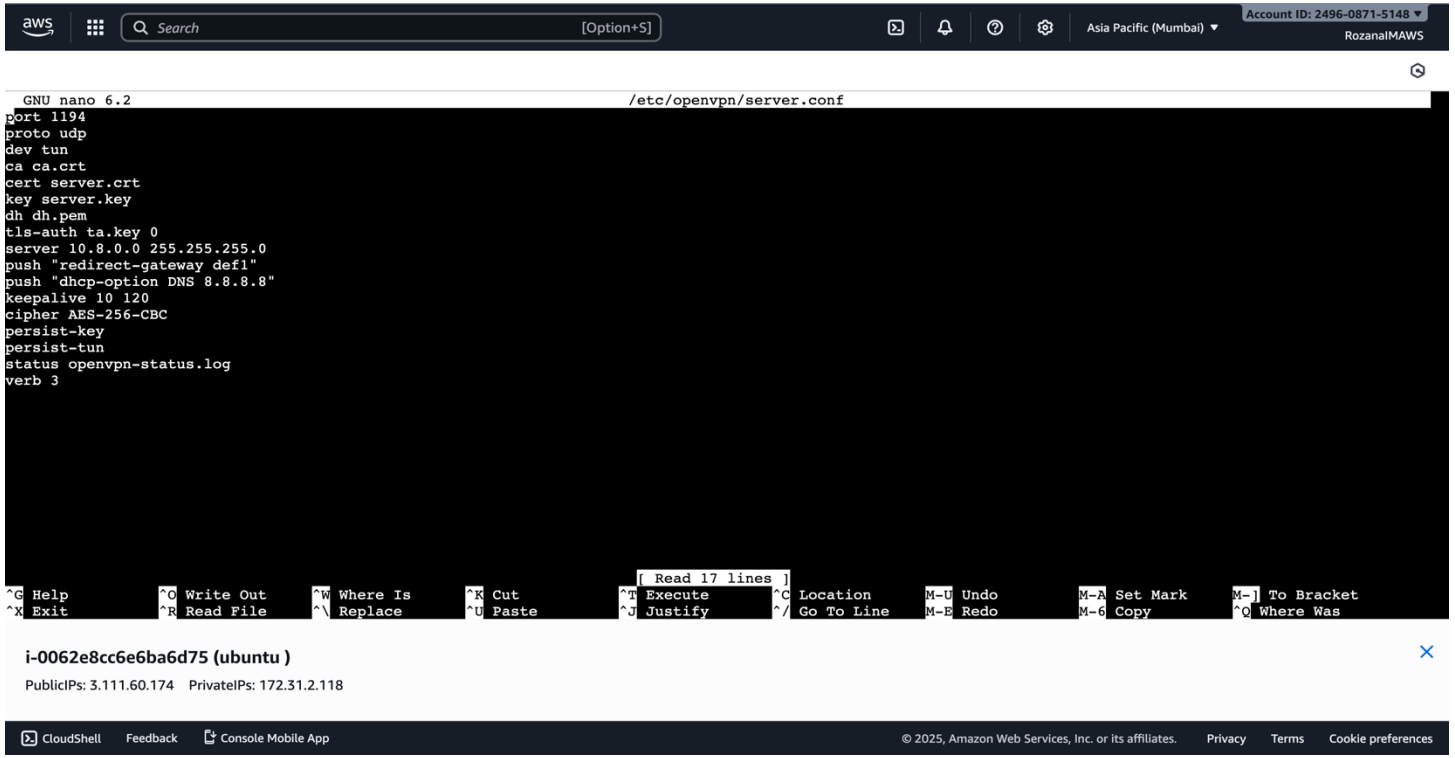
Step 3: Configure the OpenVPN server

- 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



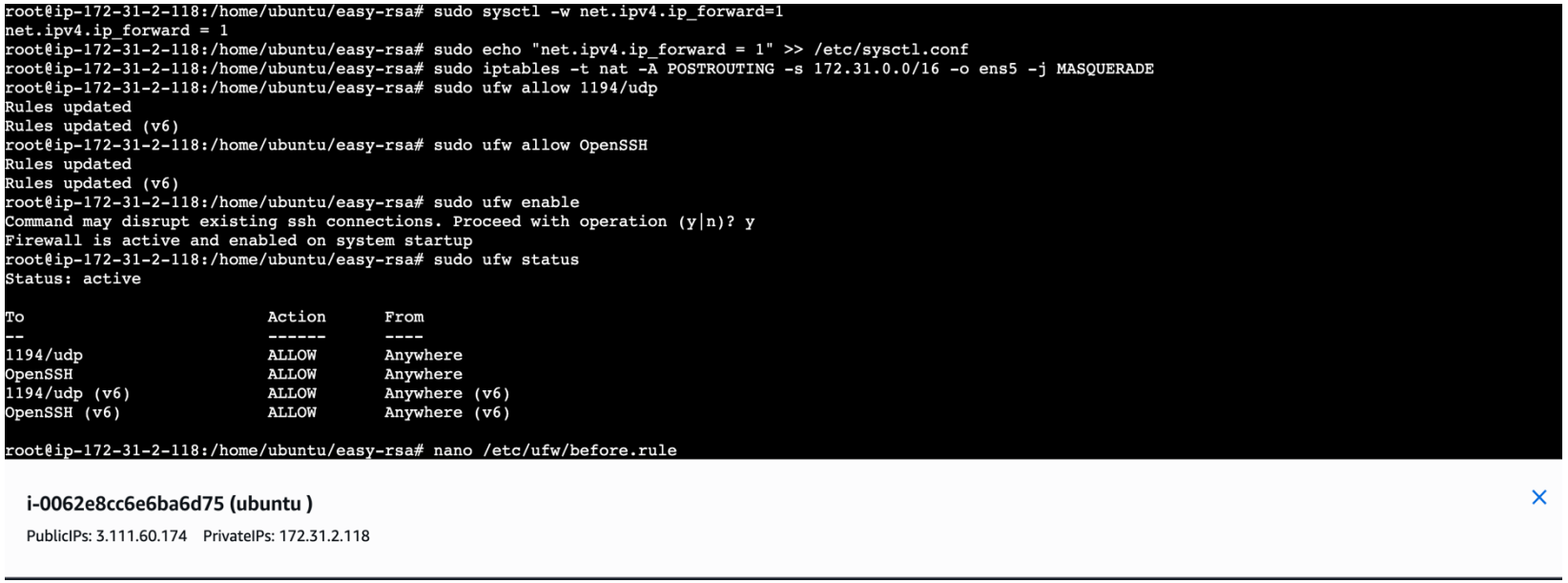
3.1 Update the VPN CIDR in Configuration

- 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.



- 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

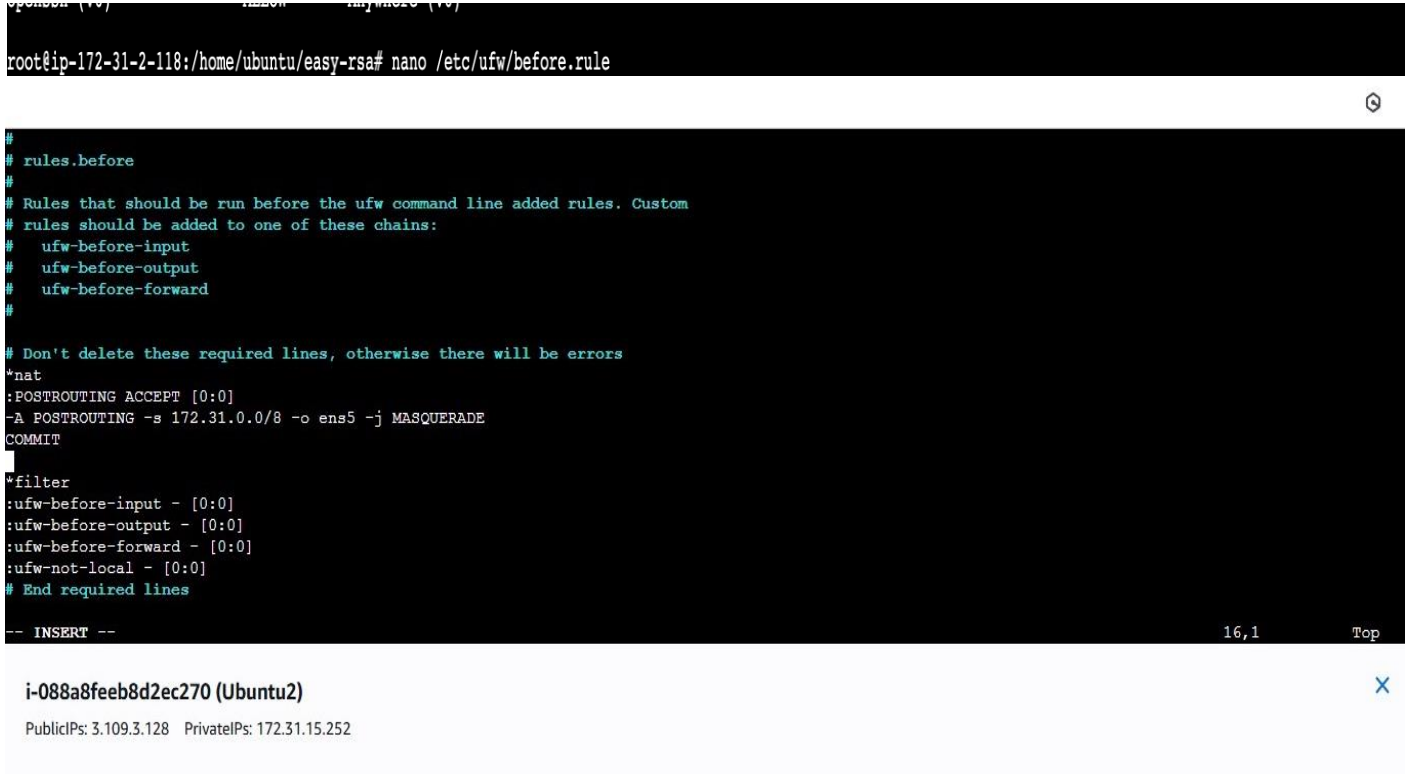
- Set up firewall (UFW) and NAT rules to allow VPN traffic and secure communication:
  sudo iptables -t nat -A POSTROUTING -s 172.31.0.0/24 -o ens5 -j MASQUERADE
  sudo ufw allow 1194/udp
  sudo ufw allow OpenSSH
  sudo ufw enable



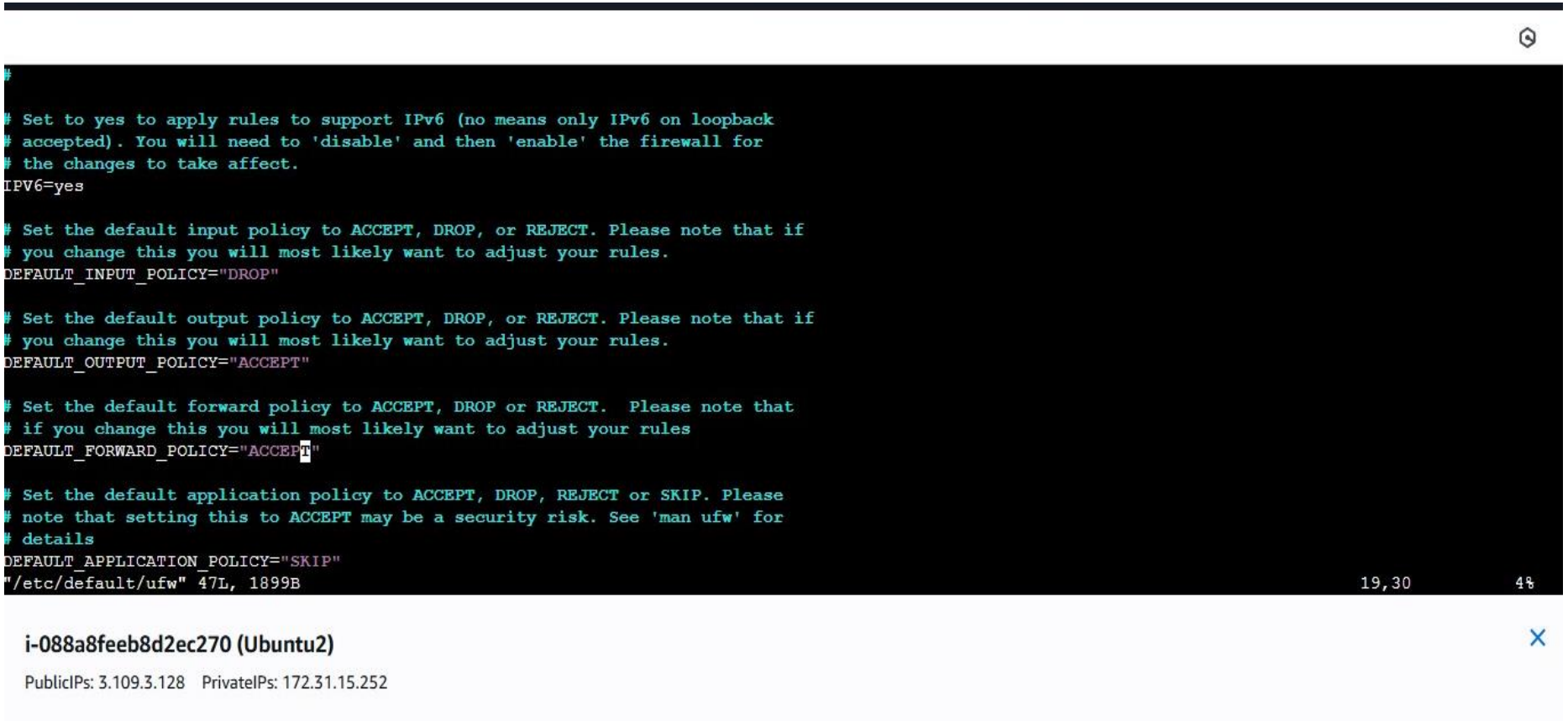
Step 4: Configure NAT and Forwarding rules

- 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

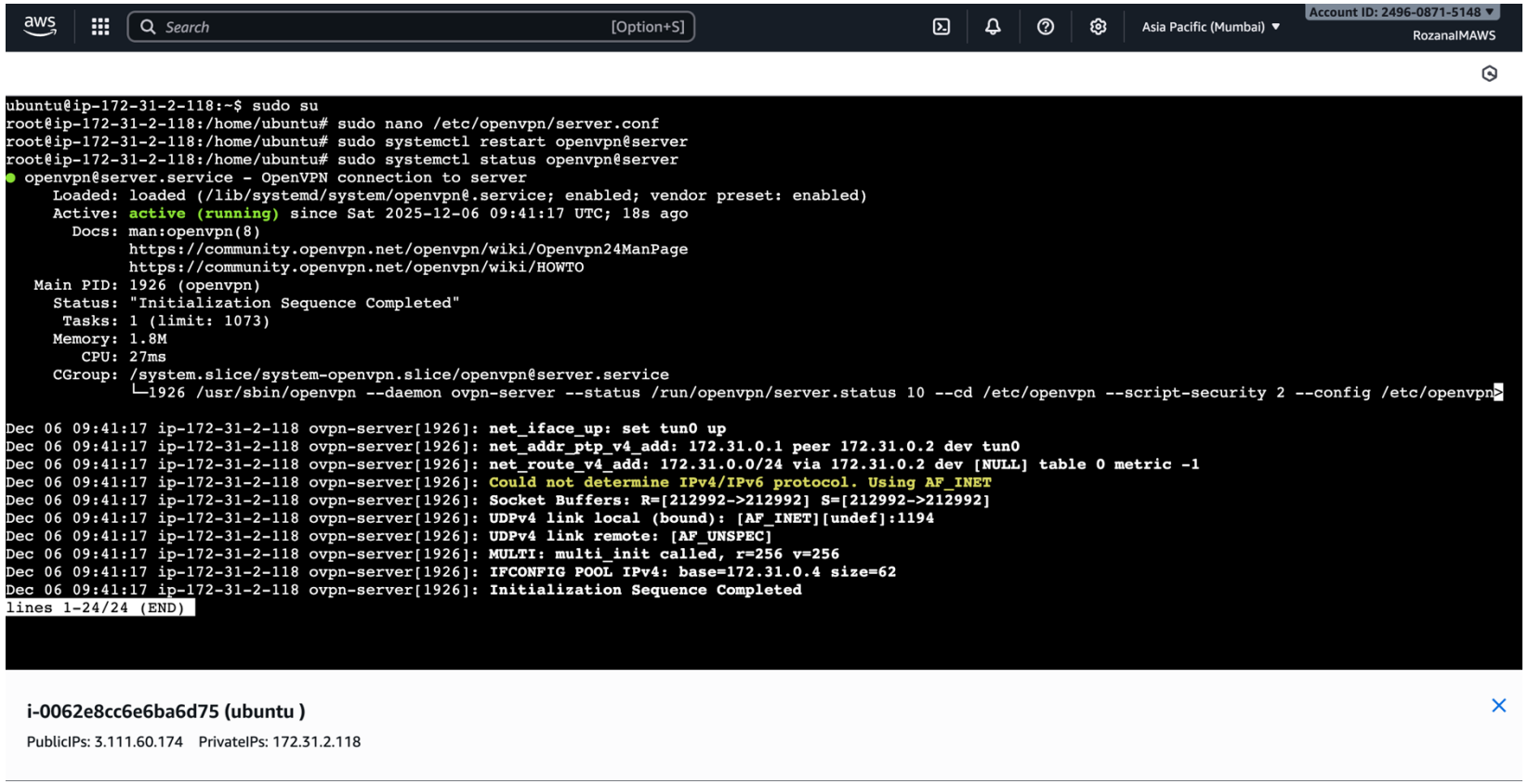


- Update the UFW default configuration to enable packet forwarding:
  sudo nano /etc/default/ufw
- Set:
  DEFAULT_FORWARD_POLICY="ACCEPT"
- These settings allow VPN clients to route traffic through the VPN server and access internal or internet resources securely.



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.

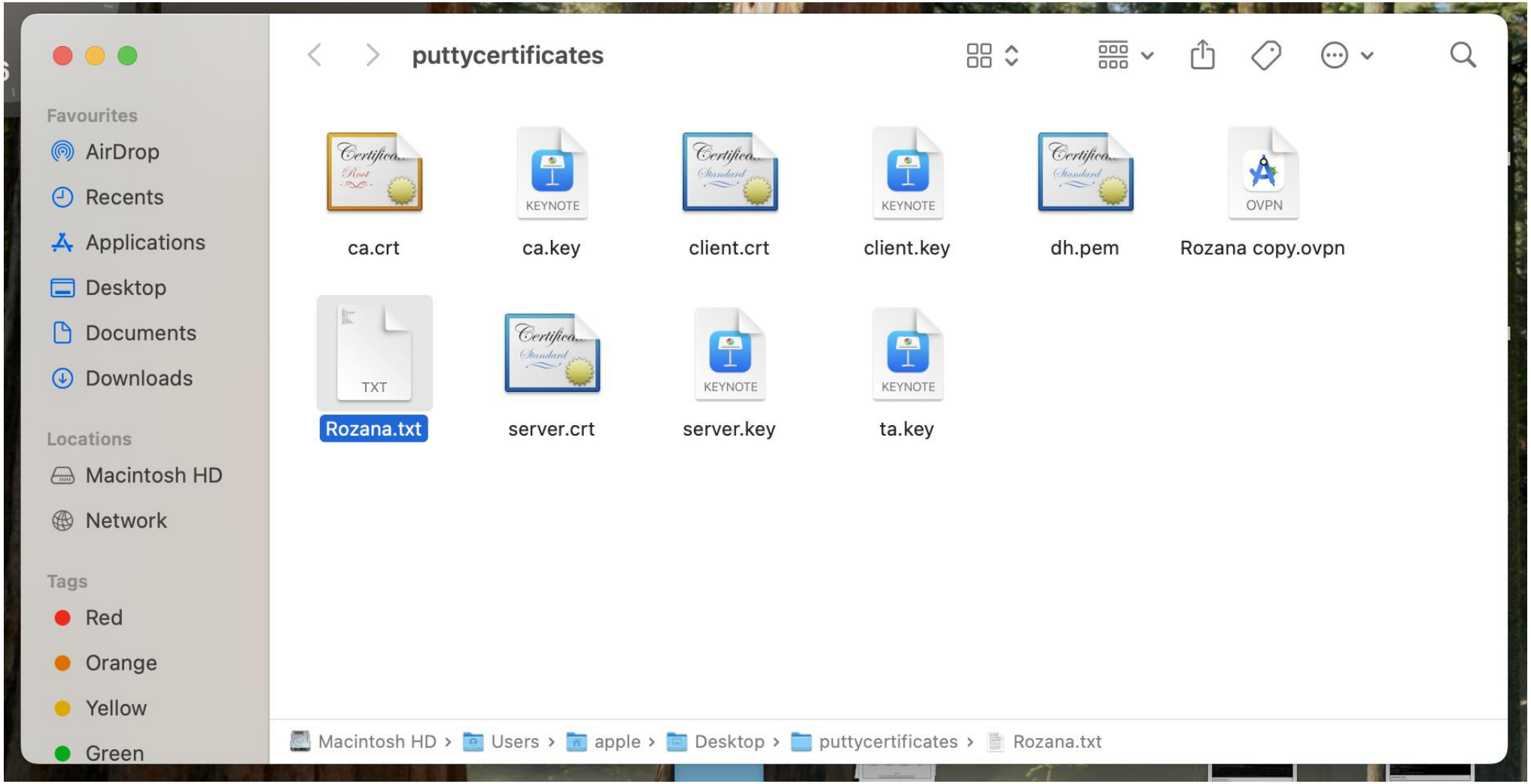


Step 6: Set up Client folder and Configuration file

- 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)

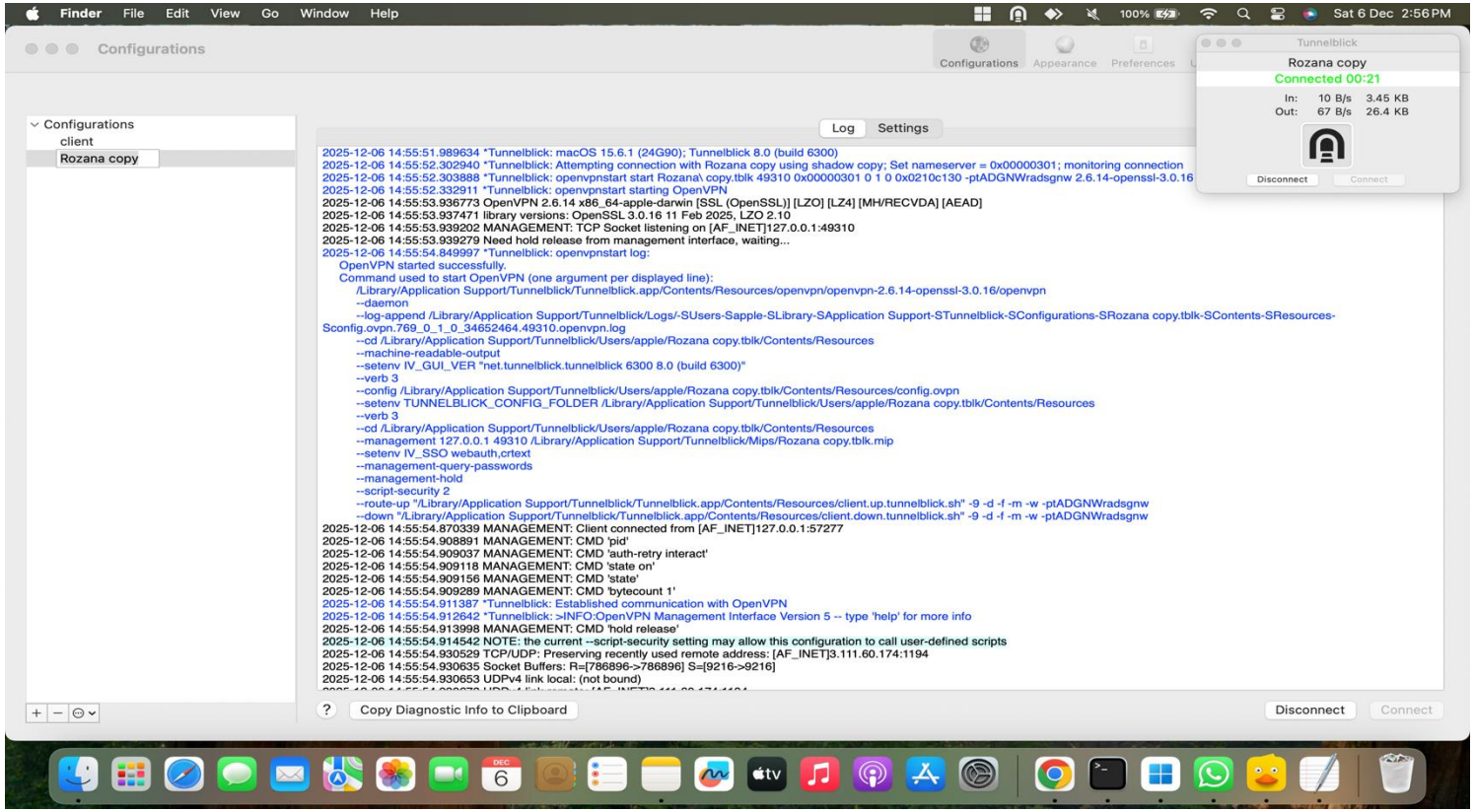
- Option 1 – From Website
  Visit 🔗 tunnelblick.net
- Download the latest .dmg file and install it manually.

- Option 2 – From Terminal (Homebrew)
  Run: **brew install --cask tunnelblick**

- Open Tunnelblick from Applications after installation.

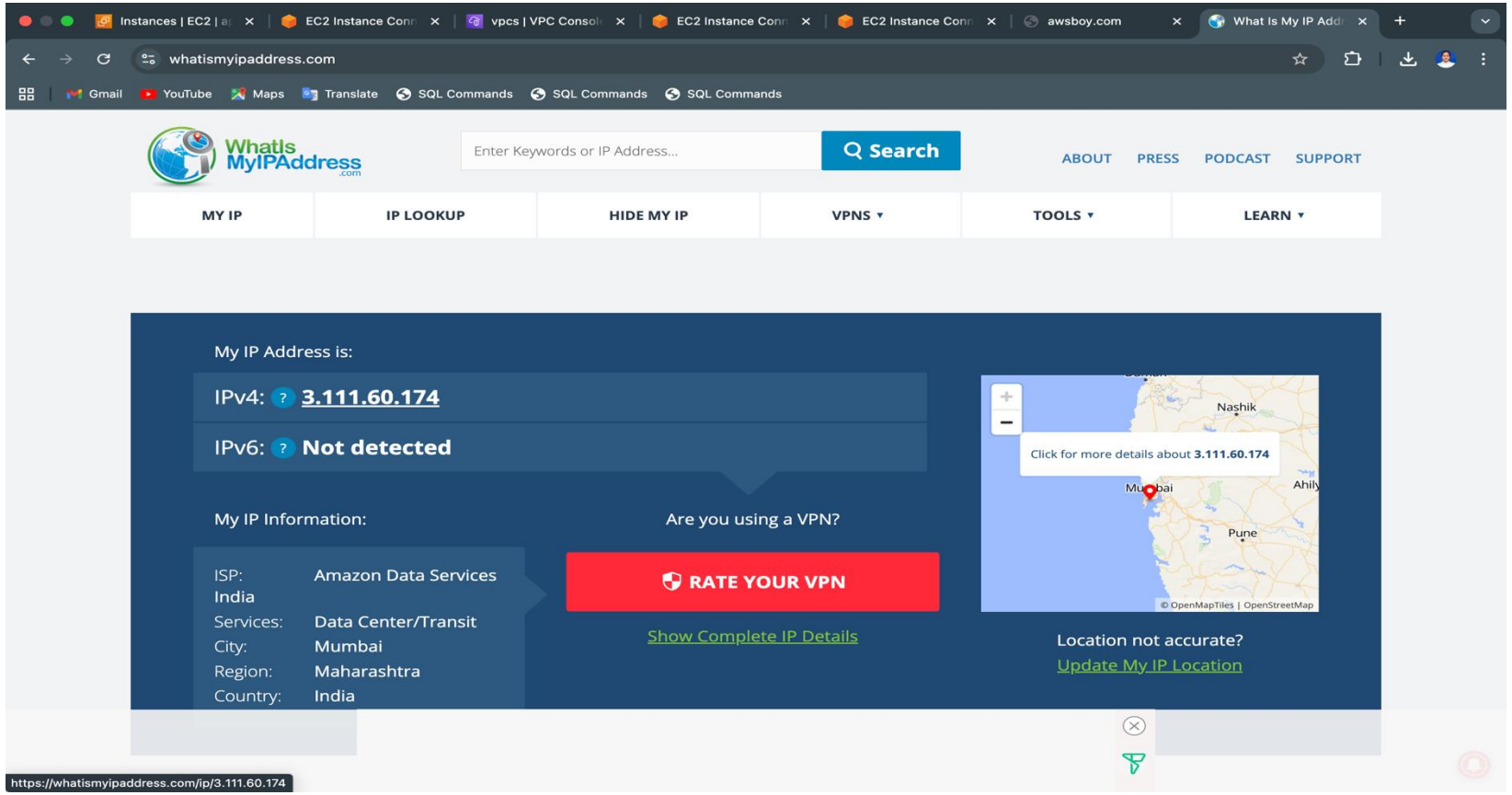
Step 8: Connect using Tunnelblick

- 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.



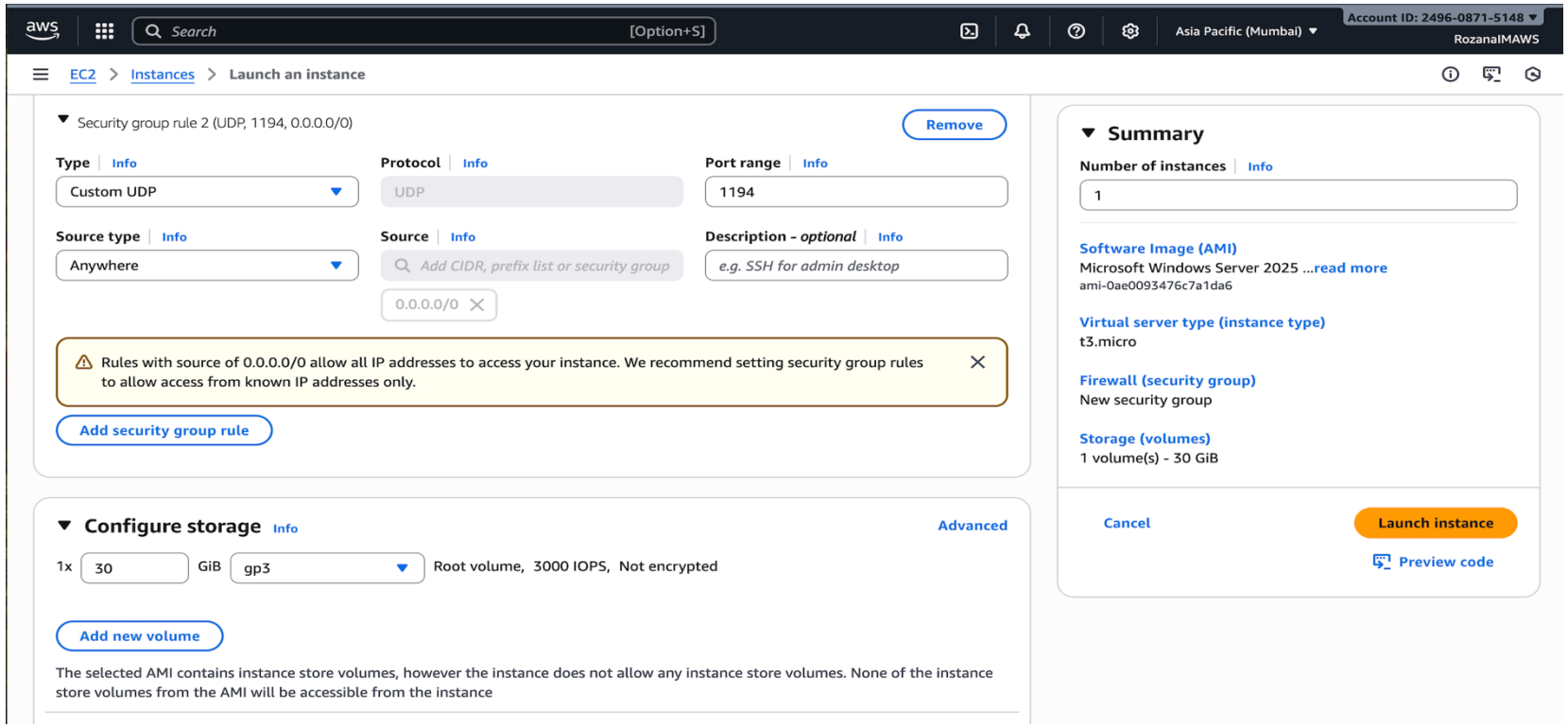
Step 9: Verify VPN Connection

- 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.

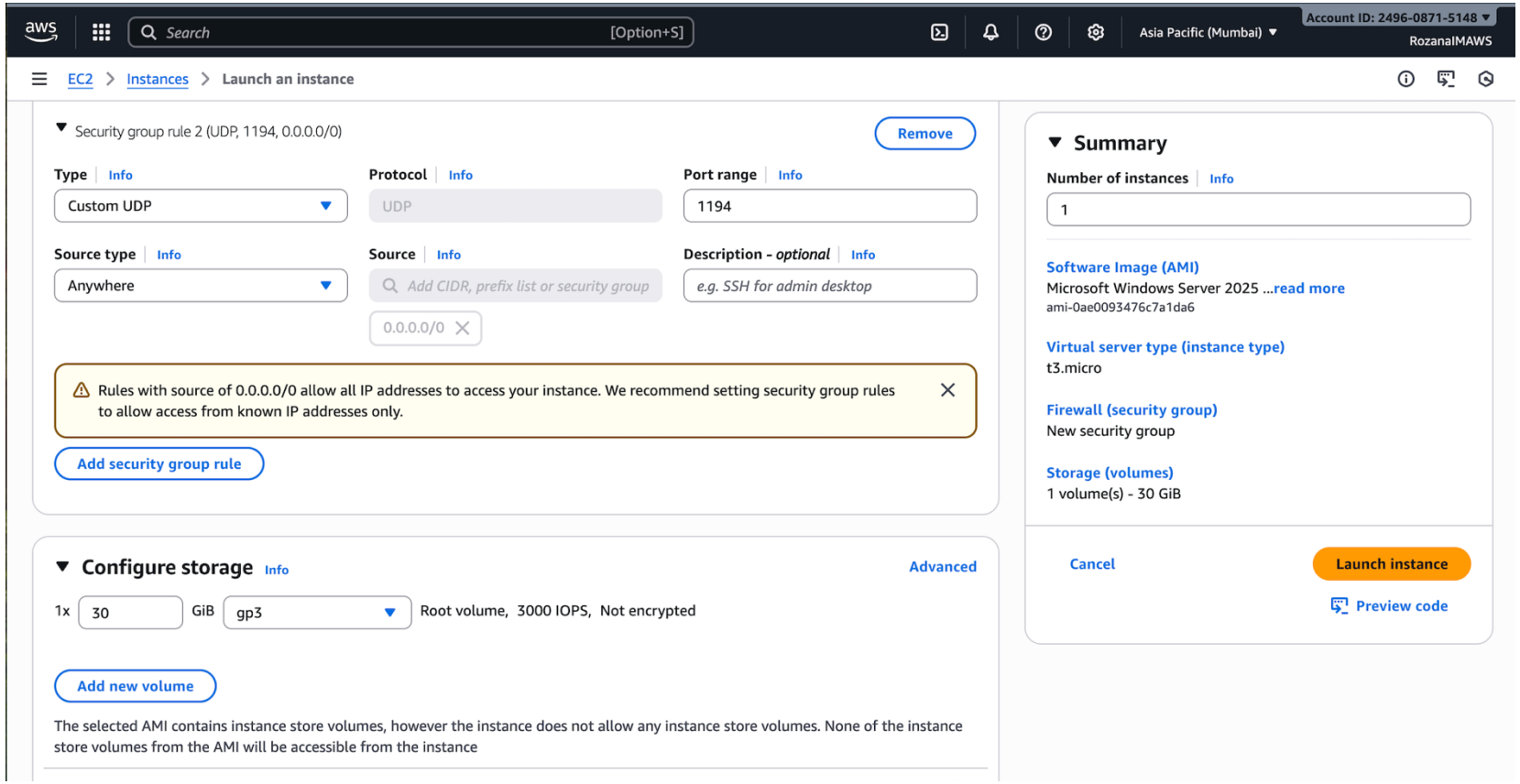


Step 1 : Launch a Private Windows Instance and Configure Security Groups

- 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.

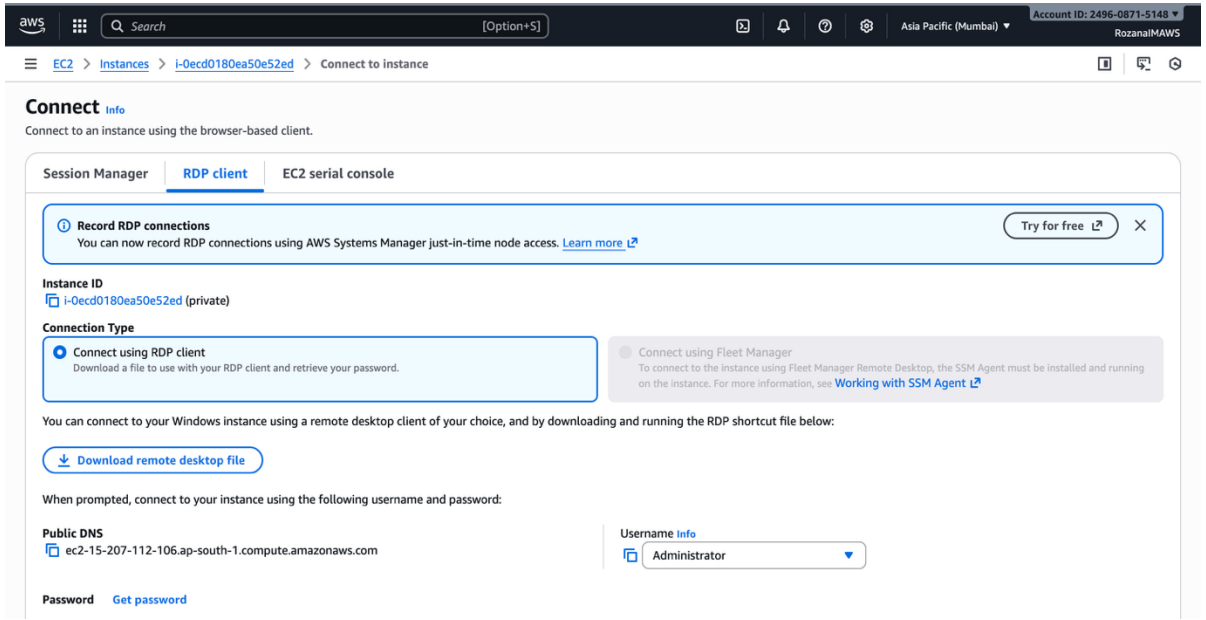


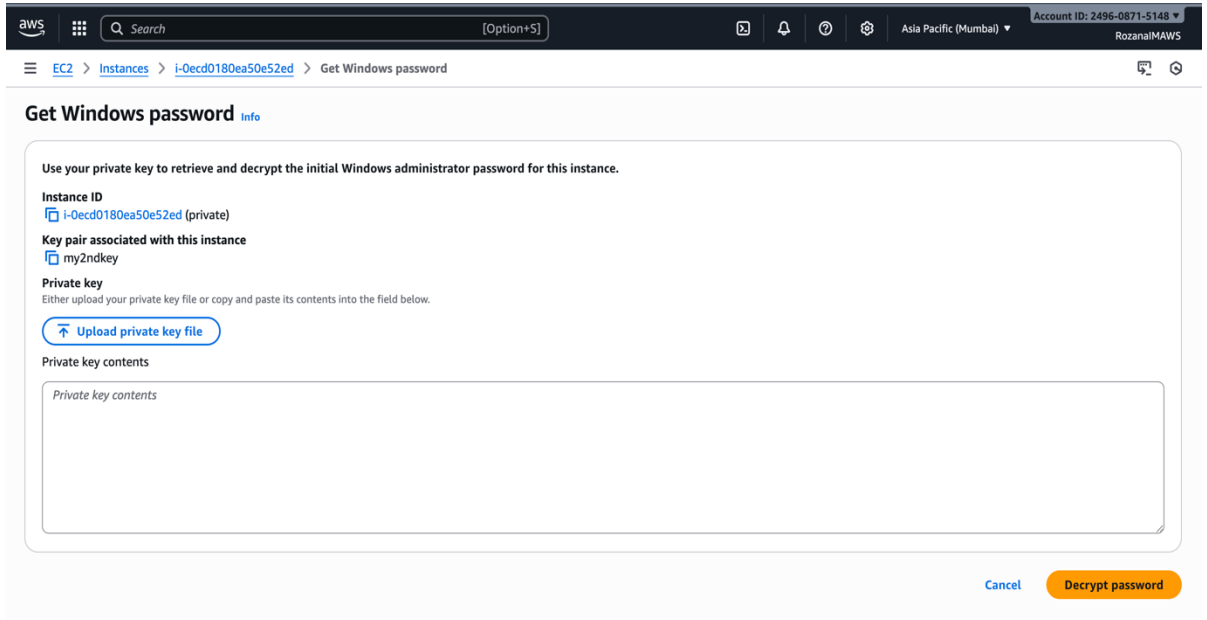
- Configure a Security Group that allows:
  - RDP (TCP, Port 3389) → for remote desktop access
  - Custom UDP (Port 1194) → to allow VPN traffic
- Launch the instance — this will create a private Windows server accessible only through the VPN connection.



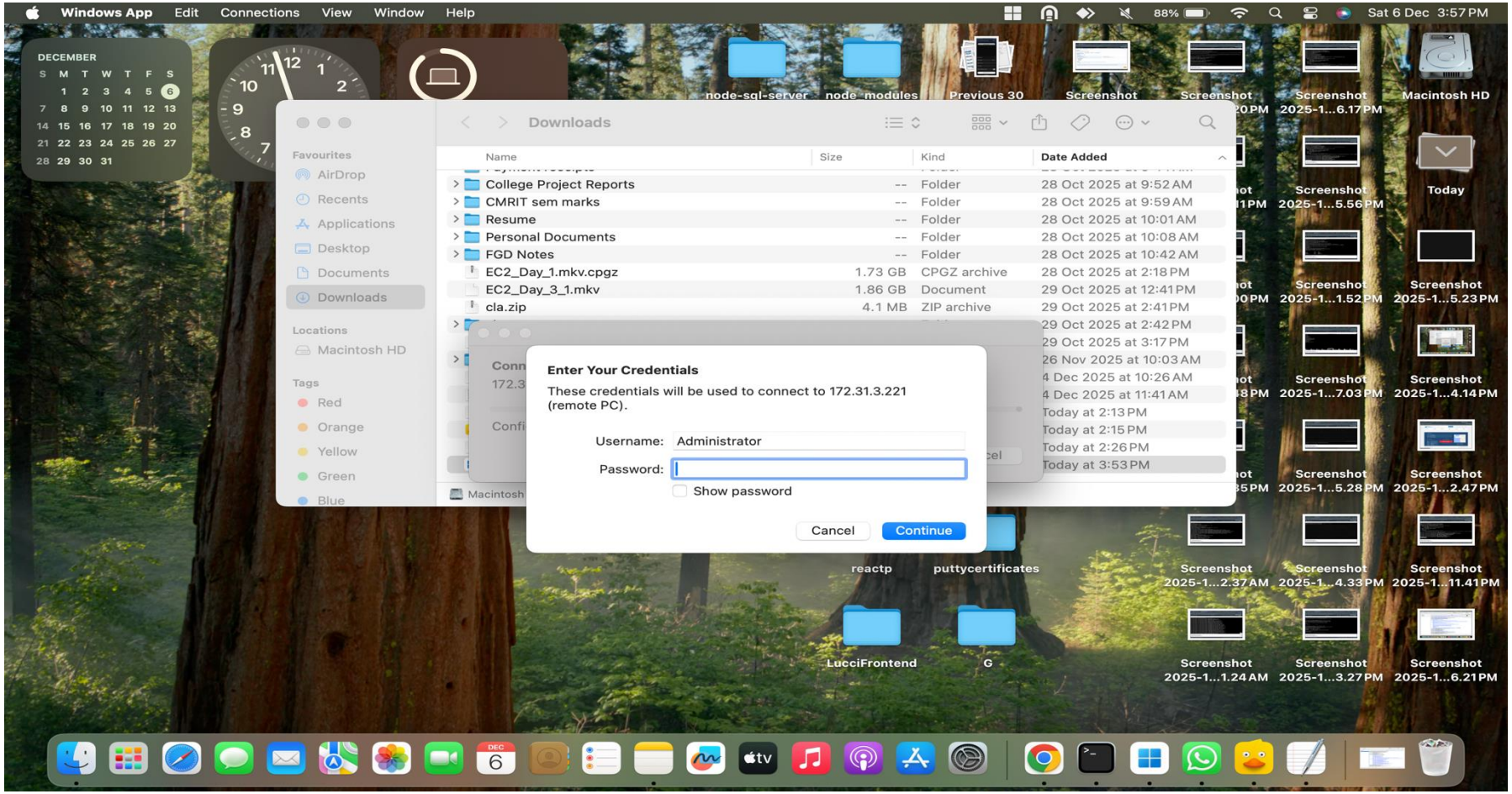
Step 3: Connect to the Windows Instance via RDP

- 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.



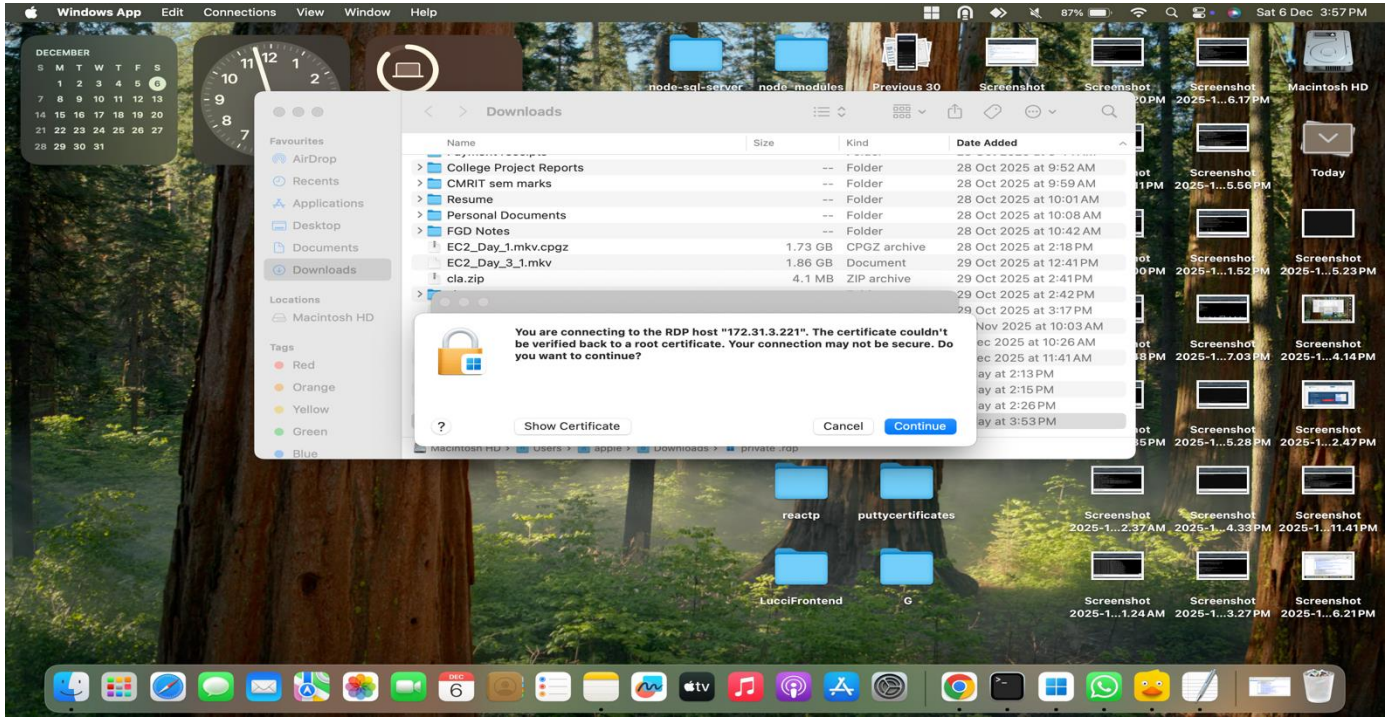


- Open the downloaded file — it will prompt for the Hostname (Public IPv4 Address) and Password.
- Enter the credentials and click “Connect.”
- A new Remote Desktop window opens, giving full access to your Windows server instance.



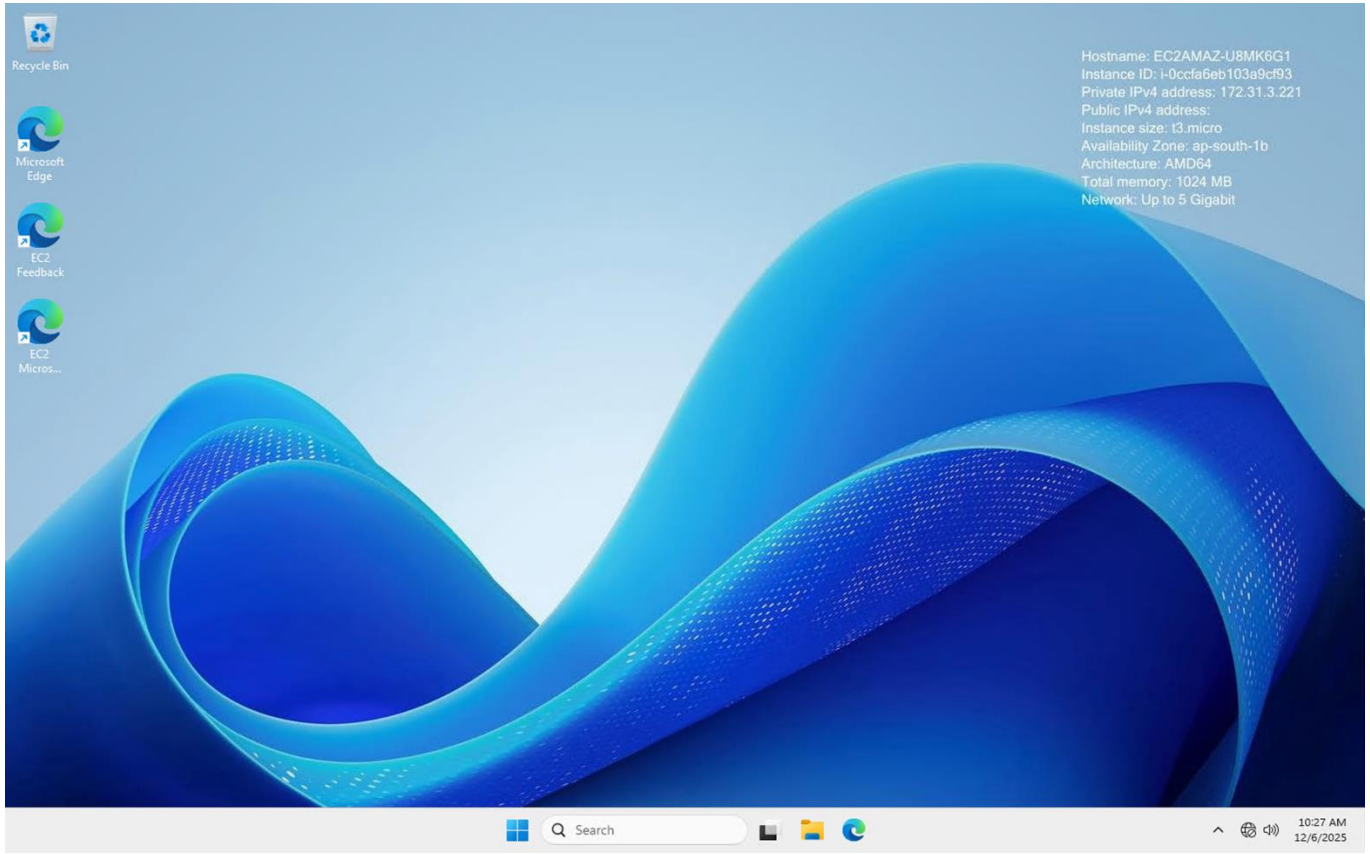
Step 4: Connection Failed Without VPN

- 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**