**Project name: Site to Site VPN by using. Strongswan**

Contents

[Objective: 2](#_Toc219131244)

[Project Demo Description – Site-to-Site VPN using AWS & strongSwan 3](#_Toc219131245)

[Architecture 3](#_Toc219131246)

[3](#_Toc219131247)

[Requirements: 3](#_Toc219131248)

[Architecture Overview (What I built) 3](#_Toc219131249)

[Resources: 4](#_Toc219131250)

[Critical Routing (Cloud VPC) 4](#_Toc219131251)

[Customer Gateways 6](#_Toc219131252)

[Virtual Private Gateway 7](#_Toc219131253)

[Site-to-Site VPN connections 7](#_Toc219131254)

[iptables Configuration 14](#_Toc219131255)

[Final Verification Commands 27](#_Toc219131256)

[**R&D Decision Matrix (FINAL FIX LOGIC)** 29](#_Toc219131257)

[AWS LOGGING – THE SINGLE SOURCE OF TRUTH 29](#_Toc219131258)

[Enable VPC Flow Logs (SUBNET level) 30](#_Toc219131259)

Objective:

My main aim to achieve to set up Site – to – Site VPN by using strong swan enabling private IP communication, ICMP testing, and SSH access across the VPN tunnel. I’m setting up two Virtual Private Clouds (VPCs) in different AWS regions and deploying EC2 instances in their public subnets. For our simulation of an on-premises environment, I’ll configure one of these EC2 instances with strongSwan.

This instance will act as the on-premises gateway, routing all traffic through it to mimic a real-world on-prem setup.

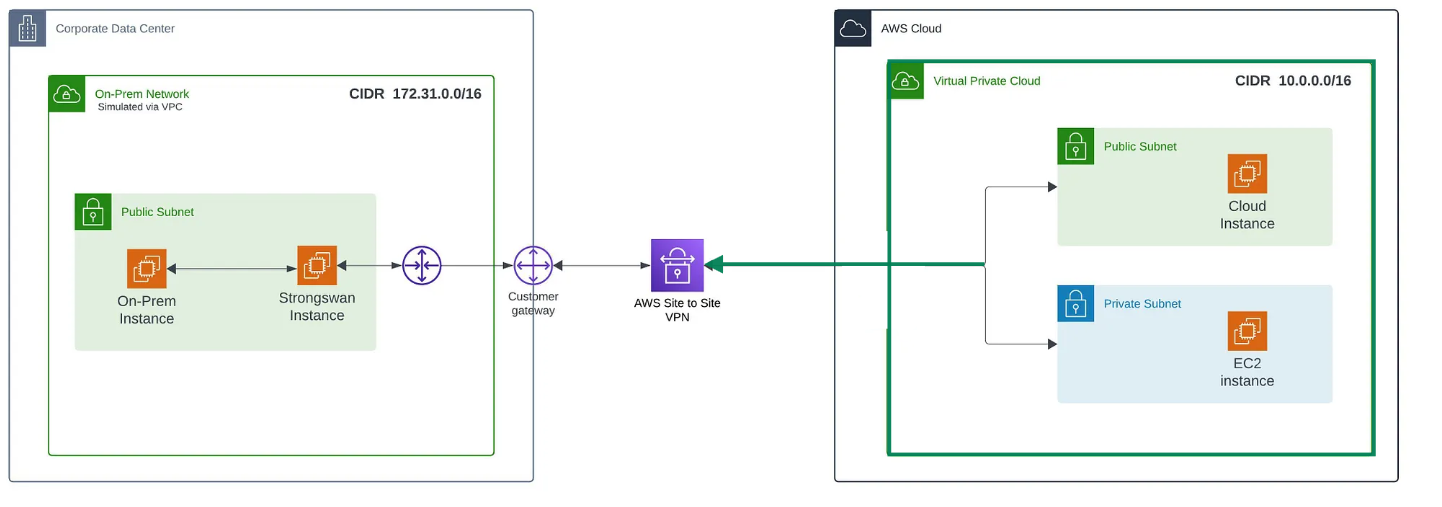
Project Demo Description – Site-to-Site VPN using AWS & strongSwan

A **Site-to-Site VPN** enables organizations to securely connect their **on-premises network** with an **AWS Virtual Private Cloud (VPC)**, allowing private and seamless communication between resources across environments. This setup ensures that applications hosted in AWS can securely communicate with systems located in an on-premises data center over an encrypted tunnel.

Since a physical on-premises infrastructure was not available for this demo, the on-prem environment was **simulated using an EC2 instance running strongSwan**. This instance acts as a **Customer Gateway (CGW)**, representing an on-premises firewall or VPN device. The AWS cloud side consists of a separate VPC with a **Virtual Private Gateway (VGW)**, connected through an **AWS Site-to-Site VPN**.

This architecture closely mirrors a real-world **hybrid cloud setup**, where organizations operate workloads across multiple locations or regions. By implementing this solution, secure private connectivity was established between the simulated on-prem network and the AWS VPC, enabling **private IP communication, ICMP testing, and SSH access** across the VPN tunnel. Such deployments are widely used by enterprises to ensure secure access to centralized applications, improve network resilience, and support geographically distributed teams.

Architecture



Requirements:

1. **2 VPC in different region in AWS console**
2. **On premises server with Elastic IP and Cloud server**
3. **Route table edit VGW**
4. **NACLs update**

Architecture Overview (What I built)

Resources:

**On-Prem (Sydney – simulated)**

* EC2 Ubuntu instance
* strongSwan installed
* Acts as **Customer Gateway (CGW)**
* CIDR: 172.31.0.0/16

**Cloud Server (Singapore)**

* VPC with private subnet
* EC2 Ubuntu instance (10.0.1.209)
* Virtual Private Gateway (VGW)
* CIDR: 10.0.0.0/16

**Post-Deployment Action:**

* Assign an Elastic IP (EIP) to the strongSwan instance.
* Stop Change Source/Destination check on the Strongswan instance.

**VPN Overview**

* AWS Site-to-Site VPN (Static routing)
* Policy-based IPsec
* No VTI required

## Critical Routing (Cloud VPC)

Cloud subnet route table must include:

**Destination Target**

**--------------------------------**

**10.0.0.0/16 local**

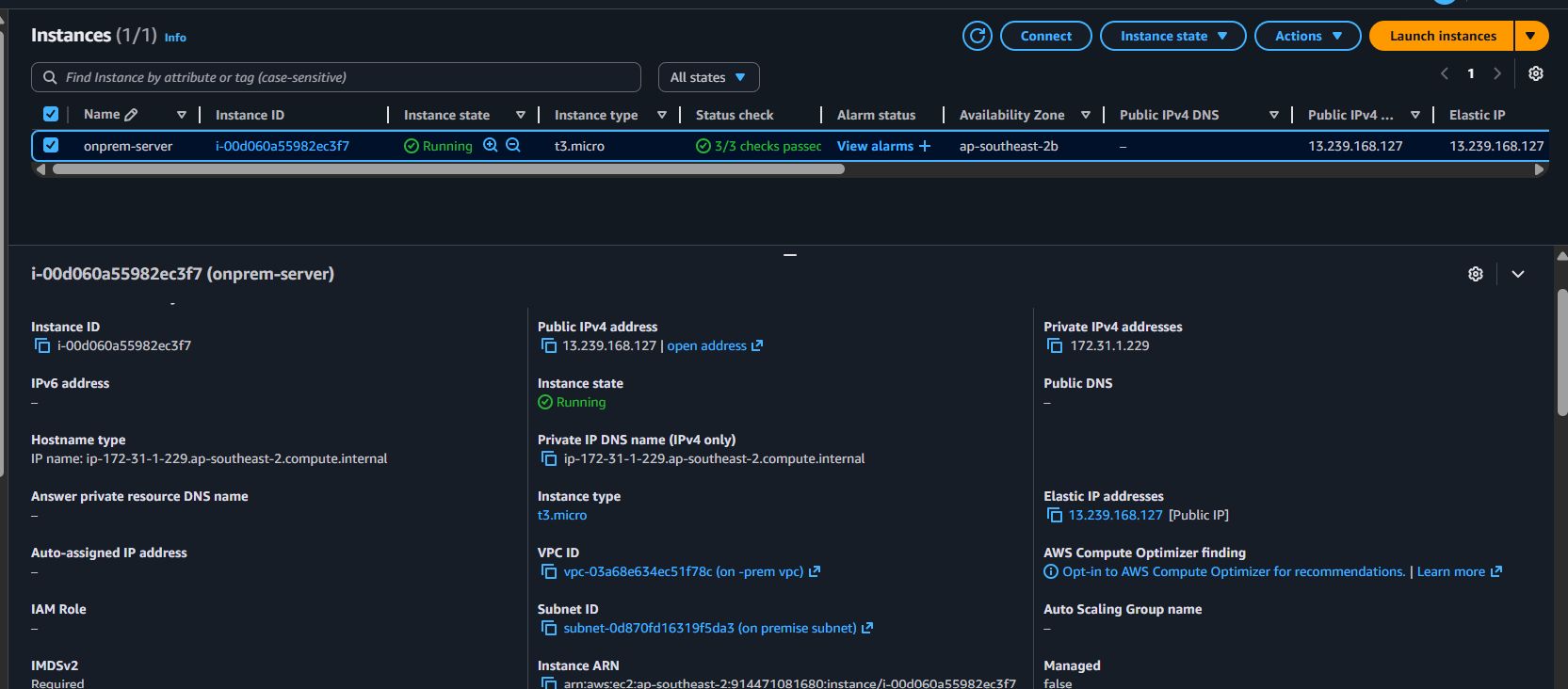
**172.31.0.0/16 Virtual Private Gateway (VGW)**

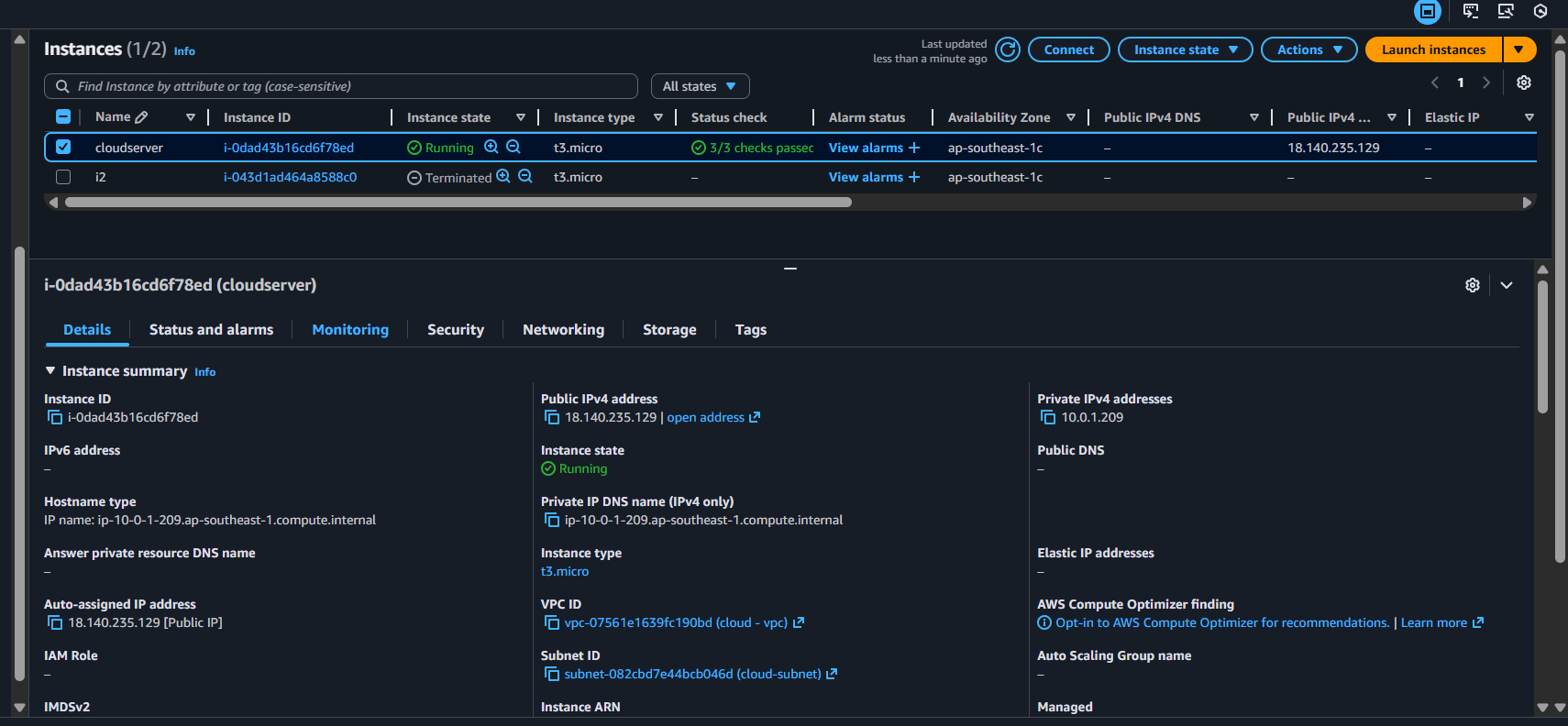
**0.0.0.0/0 Internet Gateway (IGW)**

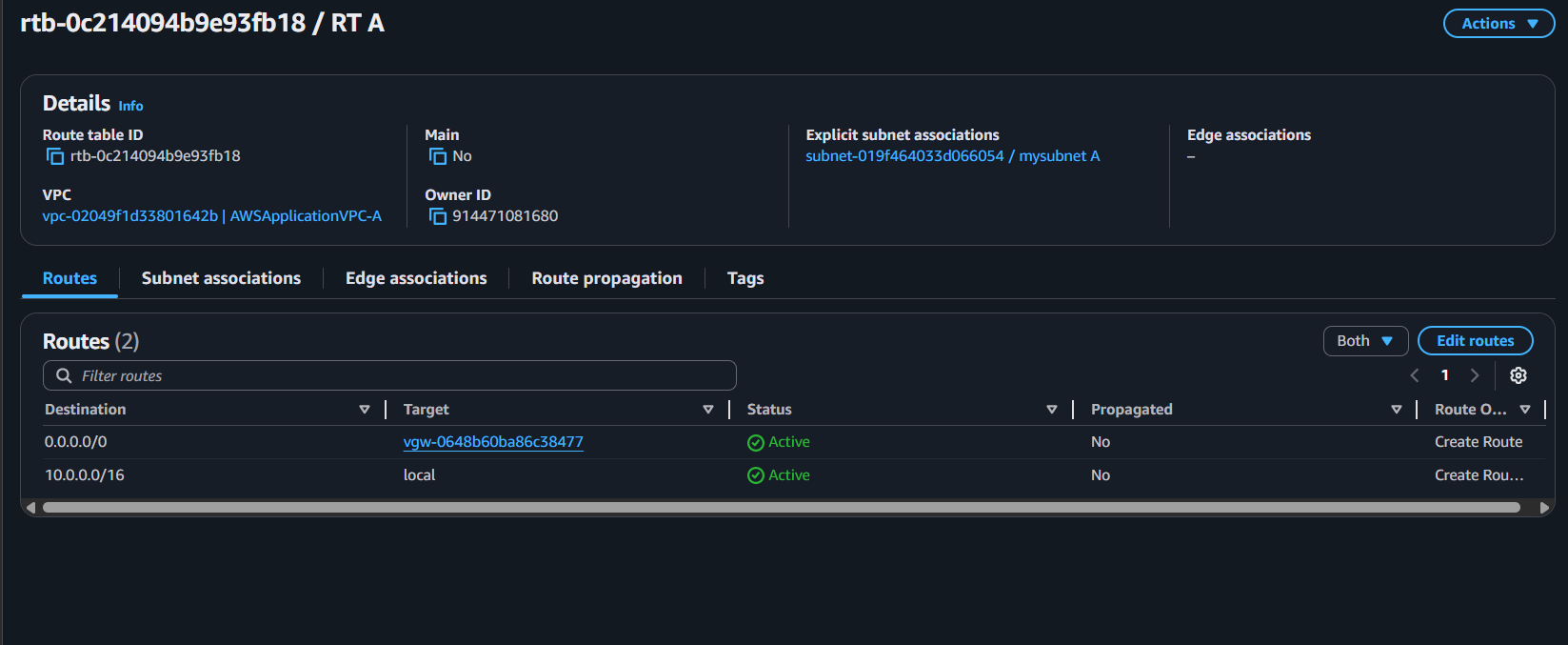
📌 This route is **mandatory** for ping / SSH to work.

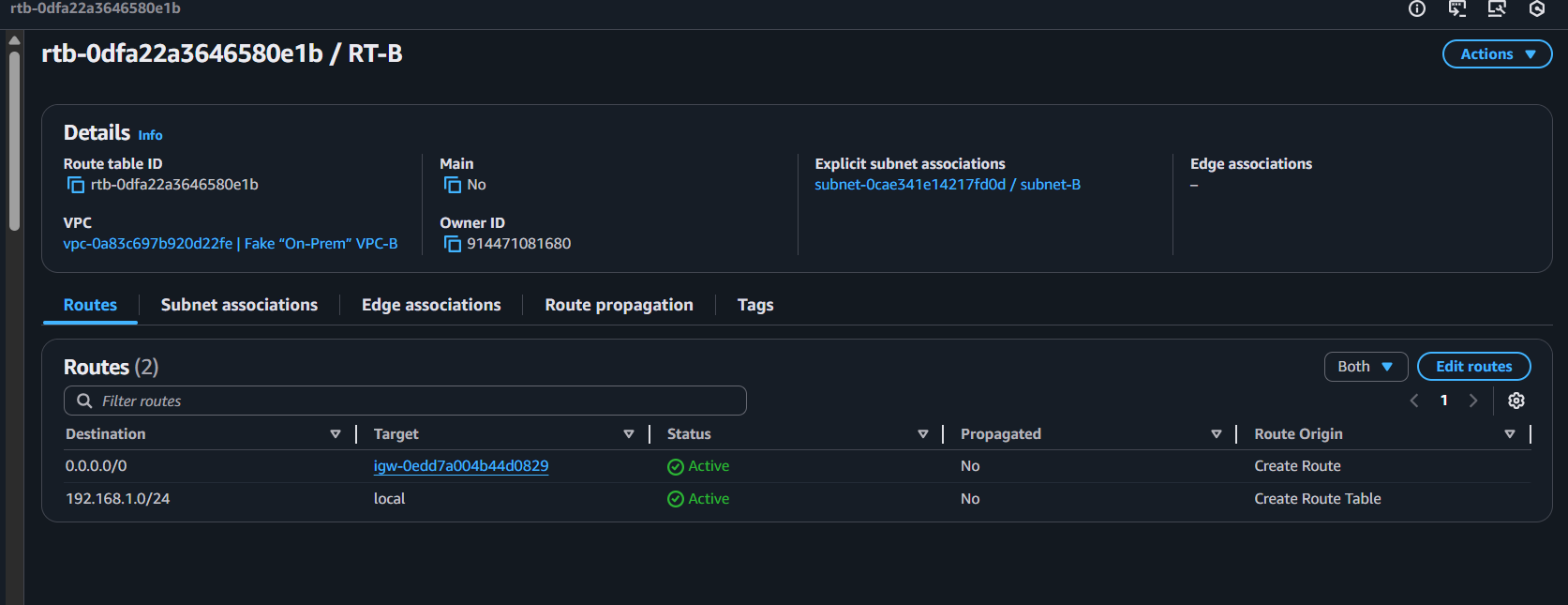
**Note:**

Here I have attached the screenshot of my above resources for the reference.



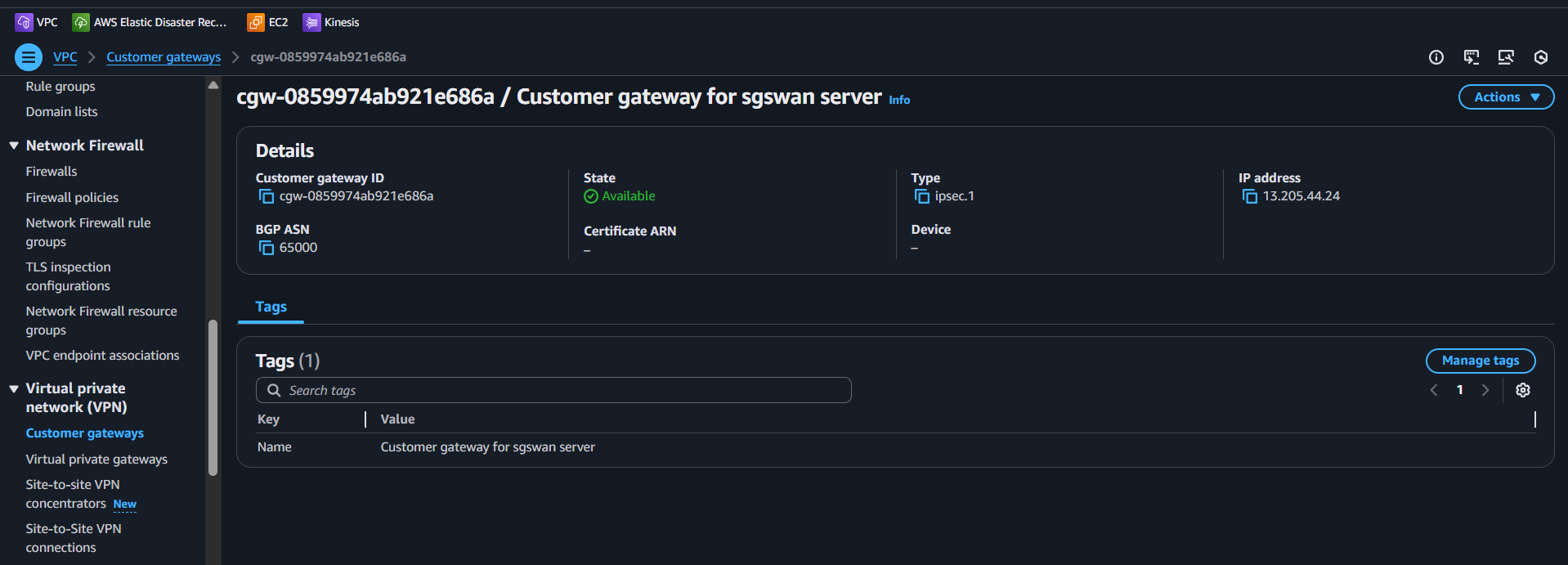






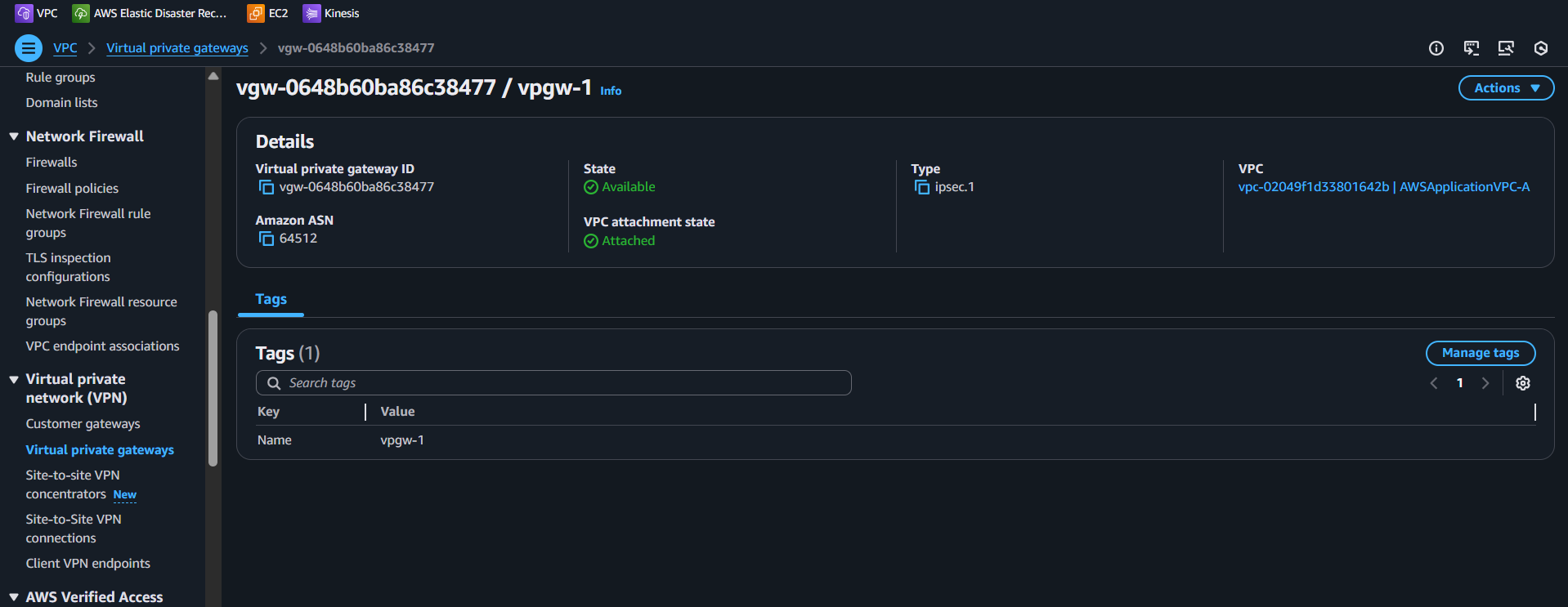
Customer Gateways

* In VPC console find Customer gateway
* Click Customer gateway and name it
* Put the public IP of the customer device or router. In my case I have put the **Elastic IP of my O- premise server.**



Virtual Private Gateway

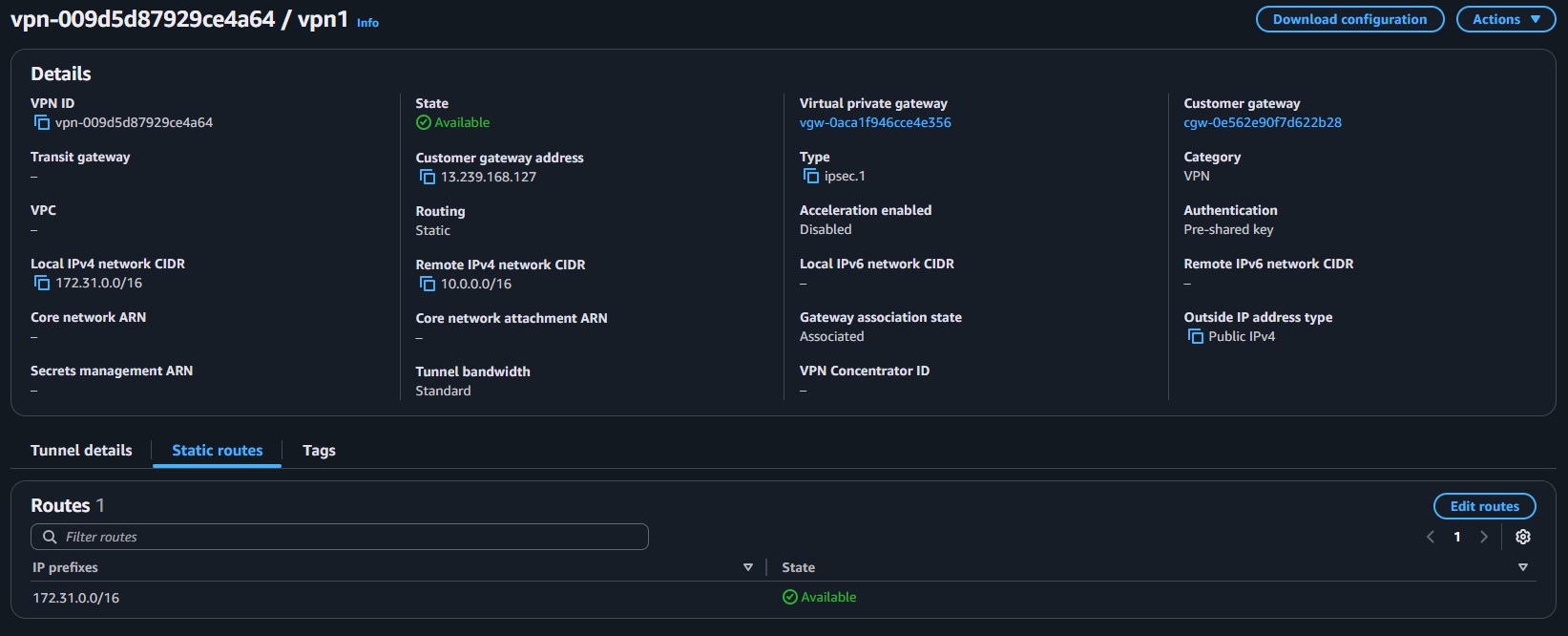
* In VPC console, find the Virtual Private Gateway
* Click on new one and name it VGW-1.
* Once it is created, click on the action and attach it to the cloud-vpc.
* To add the routes, Navigate to the route table, select the public route table cloud-vpc-rtb-public Click on Actions, edit route propagation, and check on Enable.



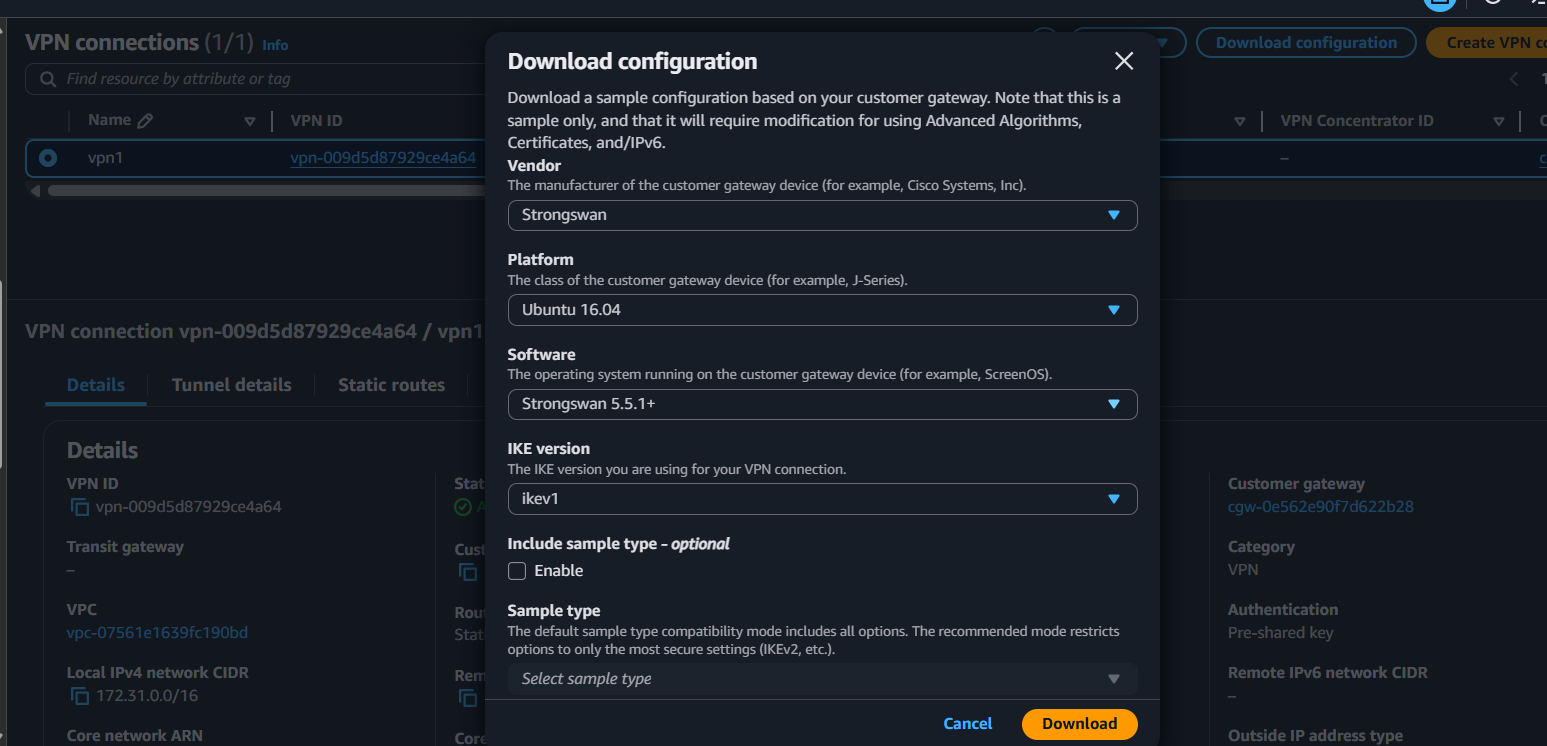
Now we have finally created our Customer Gateway and Virtual Private Gateway. We will now move on our main resource on Site to site VPN connection.  
while creating this, we will select the previous required components that we created.

Site-to-Site VPN connections

* Select the Virtual Private Gateway created previously, it is the same for the customer gateway. Select the previously created customer gateway
* Select Routing options to Static, and enter 172.31.0.0/16 Remote CIDR 10.0.0.0/16
* Once created, it may take some time to get into the Available state.

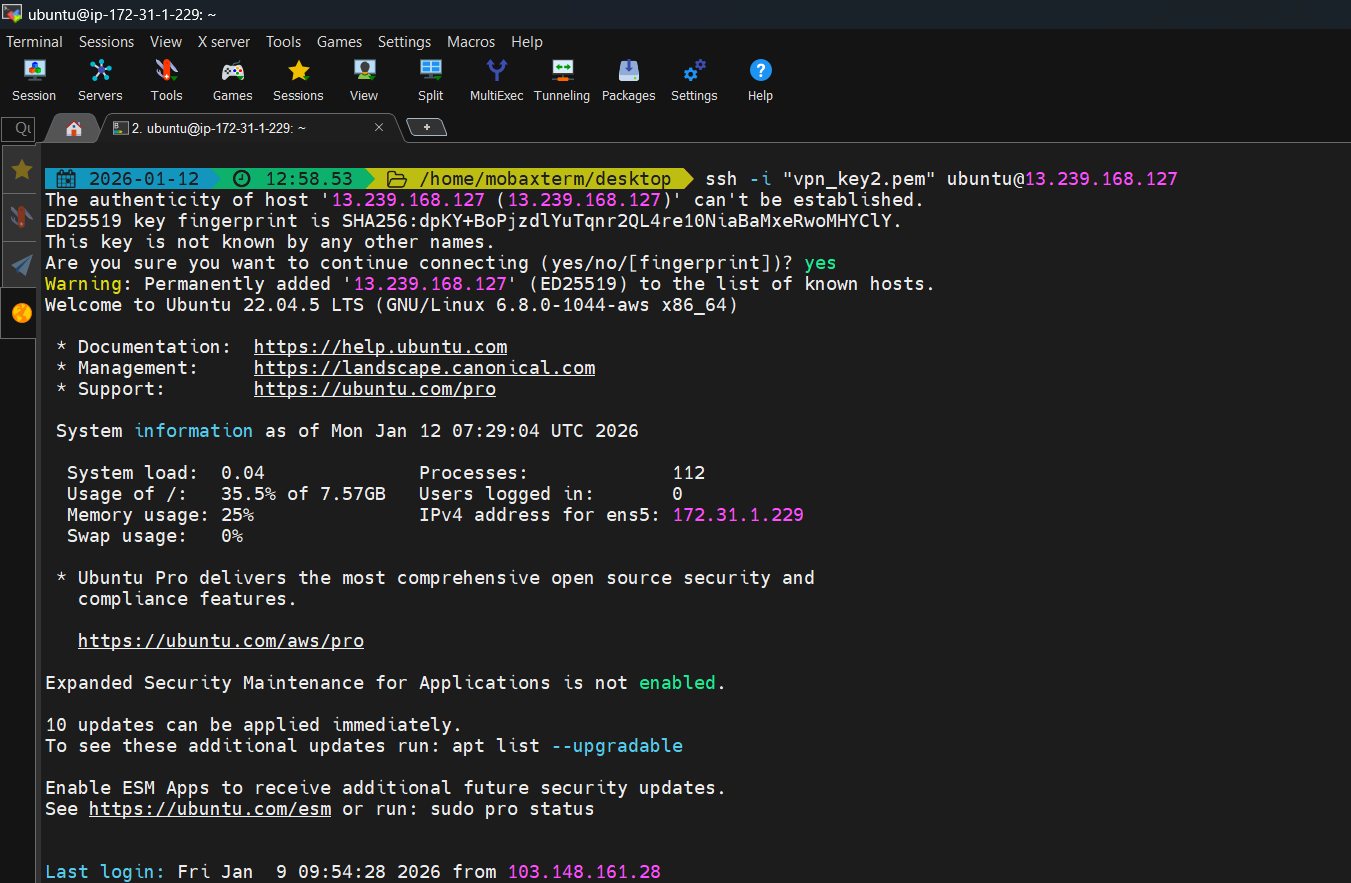


* While the VPN connection is in a pending state, select it and click on Download Configuration.
* Select the Vendor as Strongswan and click on Download



**Region: On-PremSydney**

Now, navigate to the On-prem region **(Sydney)** and take the SSH session for the strongSwan instance.

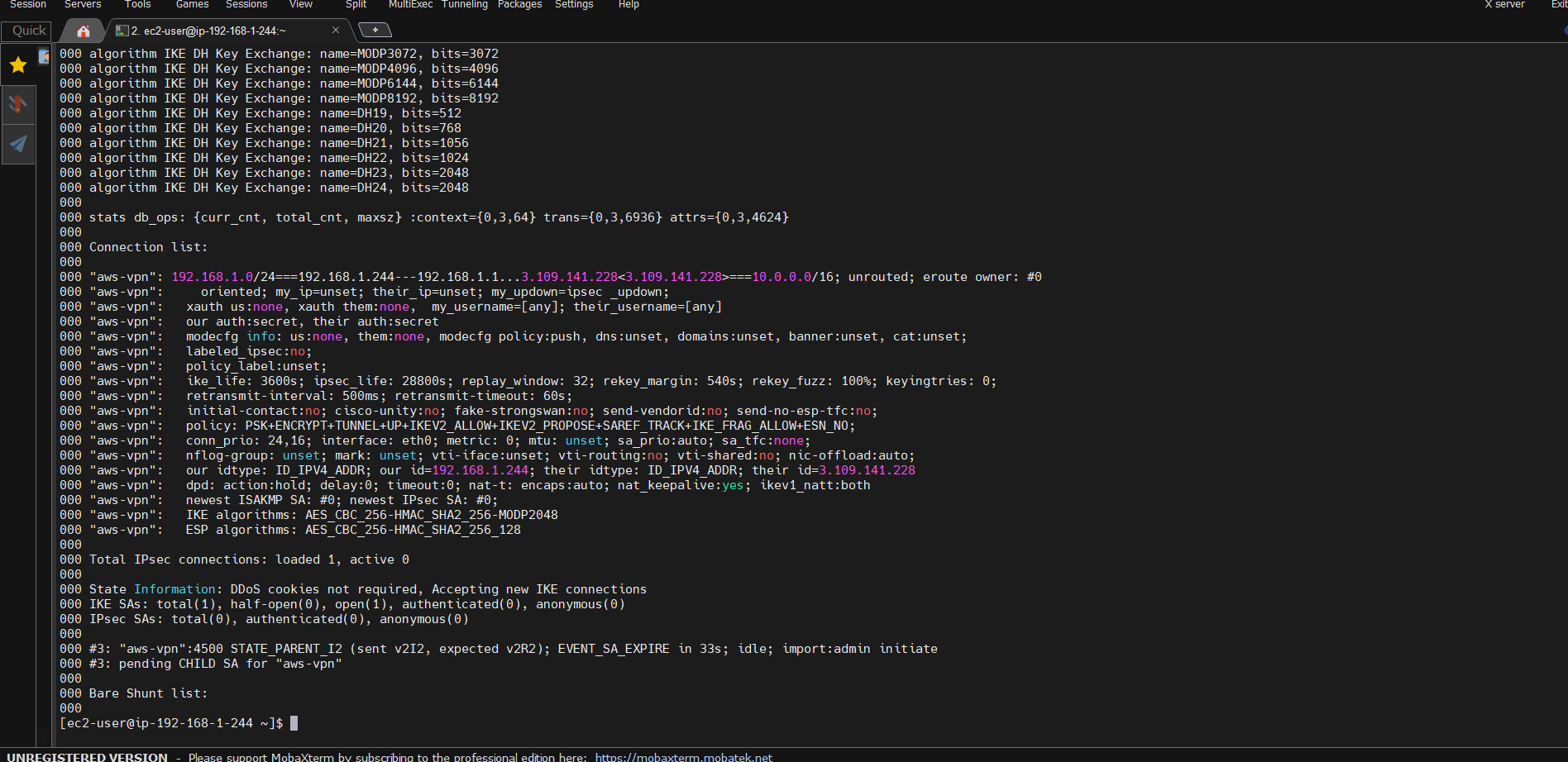


**Install strongswan**

sudo su

sudo apt update -y

sudo apt install strongswan -y

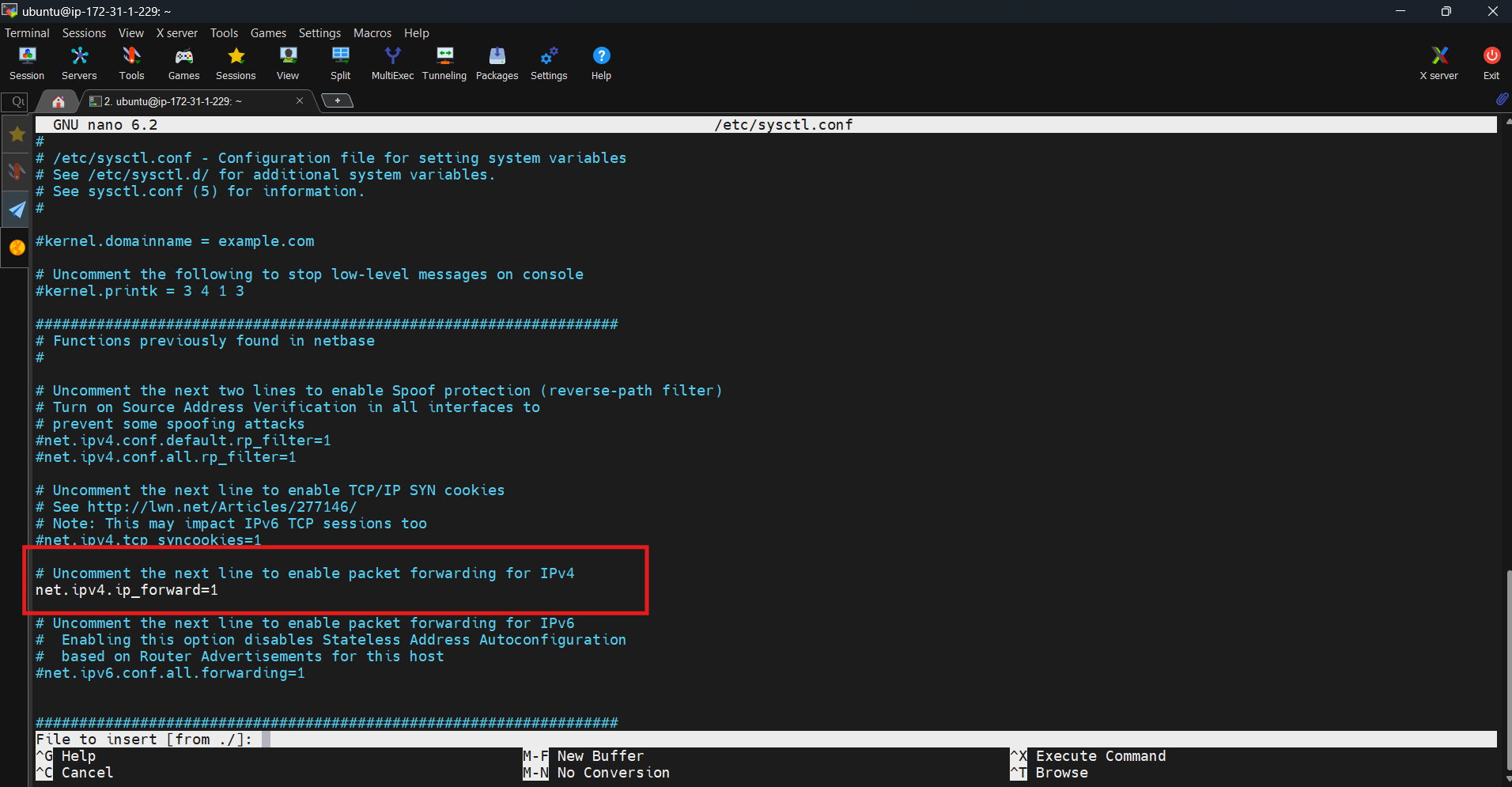


**Note:**

Open the previously downloaded configuration file and Follow the instructions mentioned.

Open **/etc/sysctl.conf** and uncomment the following line to enable IP packet forwarding:

net.ipv4.ip\_forward = 1

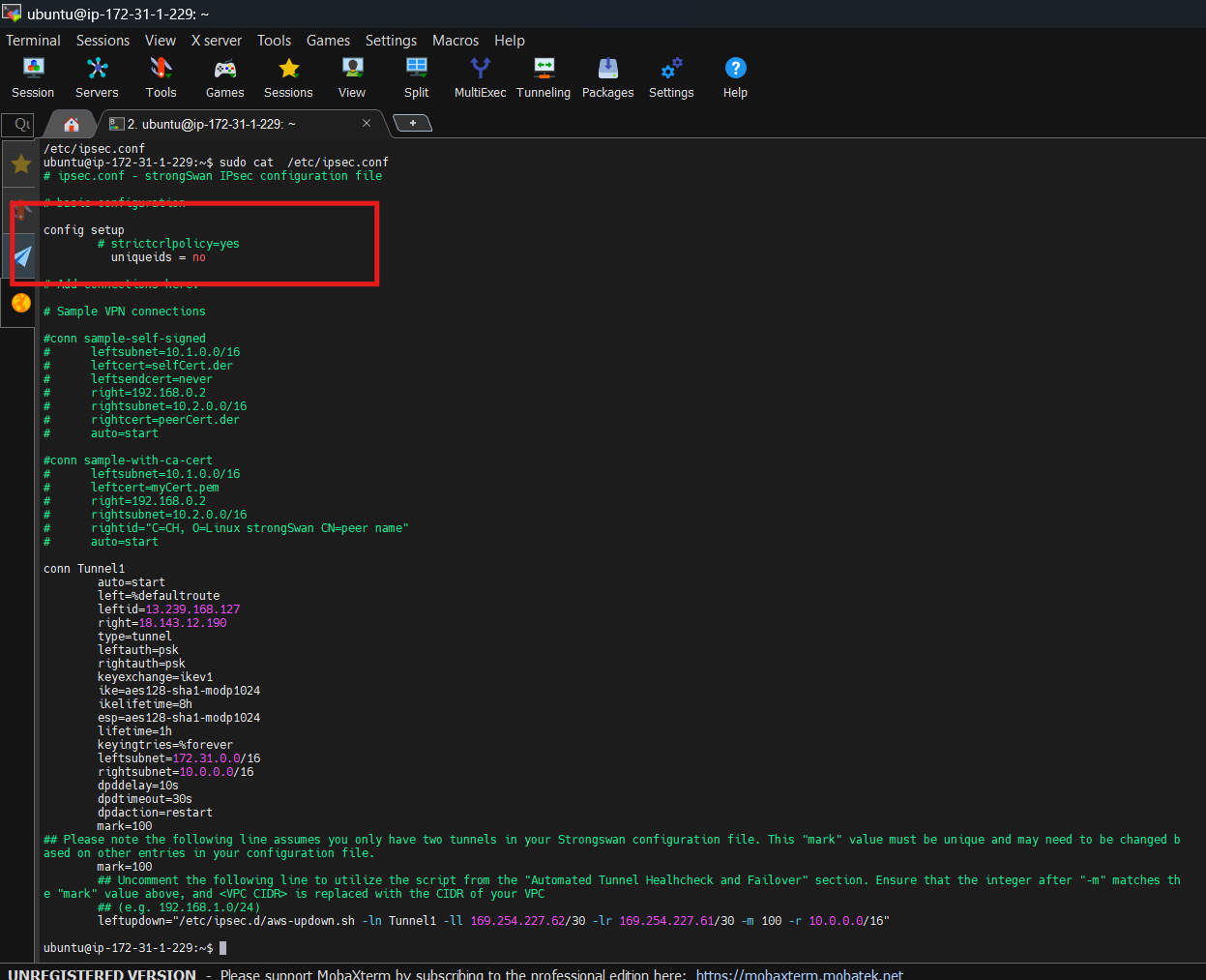


2. Apply the changes in step 1 by executing the command

sudo sysctl –p

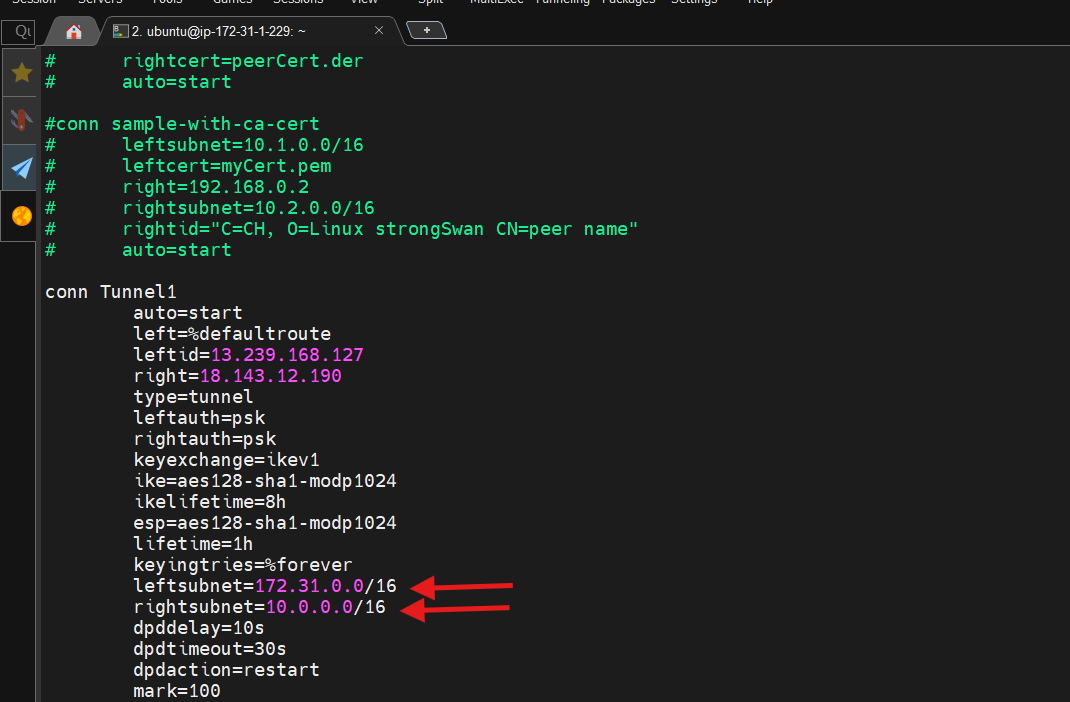


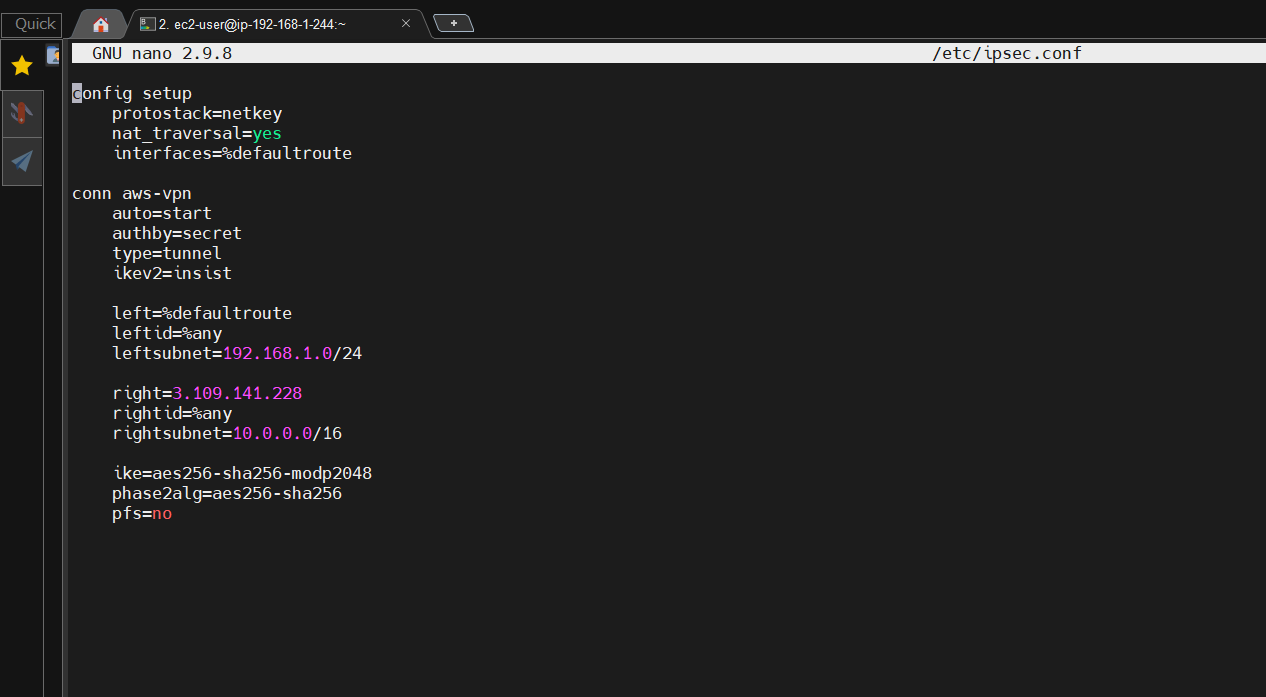
3. Create a new file at /etc/ipsec.conf if doesn’t already exist, and then open it. Uncomment the line “uniqueids=no” under the ‘config setup’ section. Append the following configuration to the end of the file:



Your file may contain different leftsubnet and rightsubnet. I recommend you modify the IPs below to the file.

leftsubnet=172.31.0.0/16  
rightsubnet=10.0.0.0/16



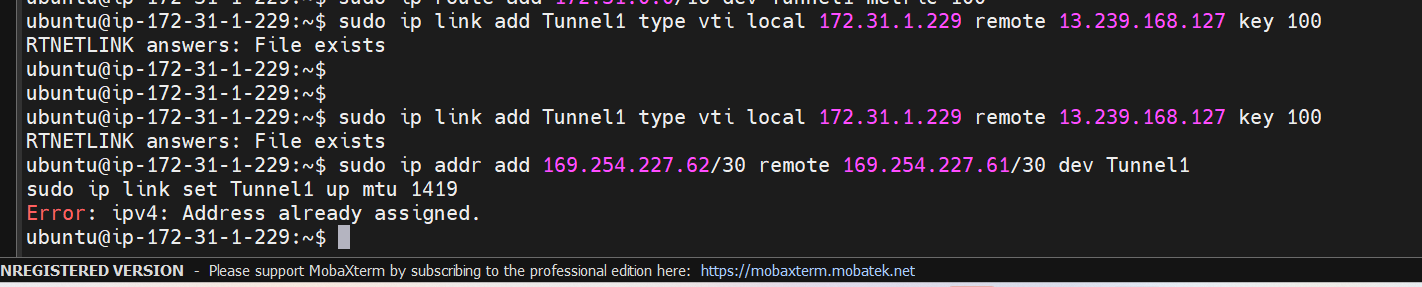


Copy and paste the remaining commands on the terminal. <Public IP> should be mentioned in your file.

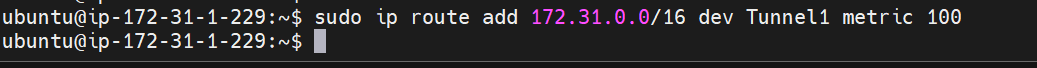
sudo ip link add Tunnel1 type vti local <LOCAL IP> remote 18.143.12.190 key 100

sudo ip addr add 169.254.227.62/30 remote 169.254.227.61/30 dev Tunnel1

sudo ip link set Tunnel1 up mtu 1419

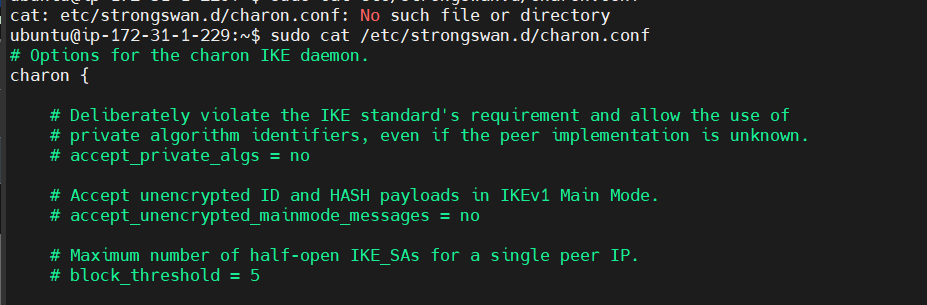
****

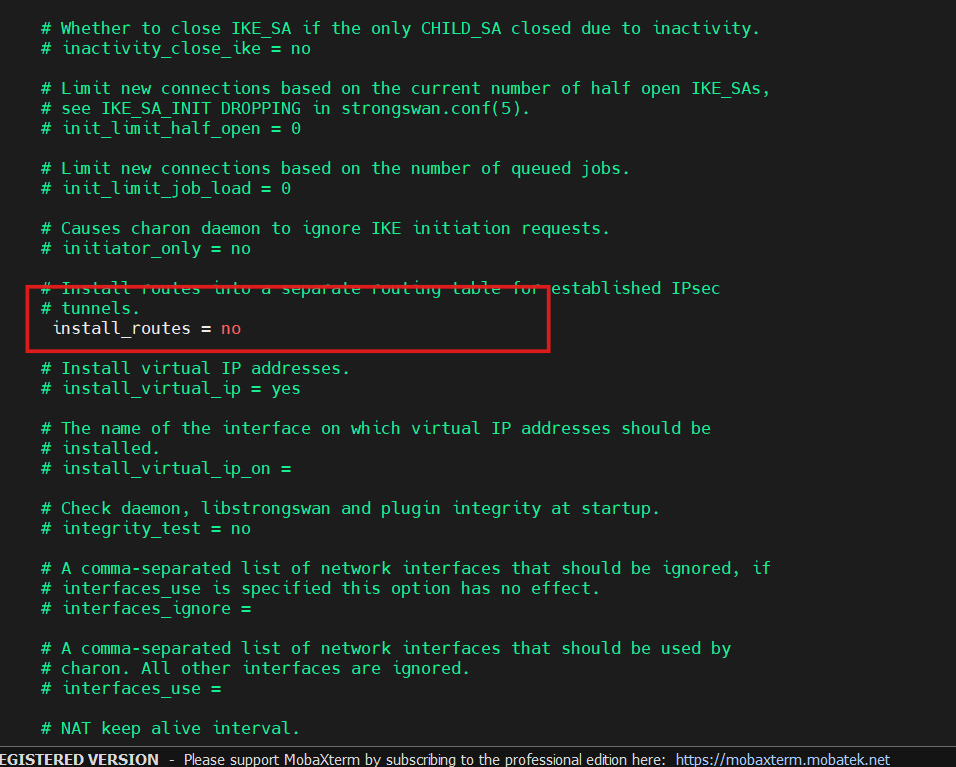
2. Depending on how you plan to handle routing, you can optionally set up a static route pointing to your VPC for your new tunnel interface. Replace <VPC CIDR> with the CIDR of your VPC (e.g. 10.0.0.0/16):



3. By default, Strongswan will create a routing entry in a different route table at launch

Open the file /etc/strongswan.d/charon.conf  
- Change the value of the line to "install\_routes=no"





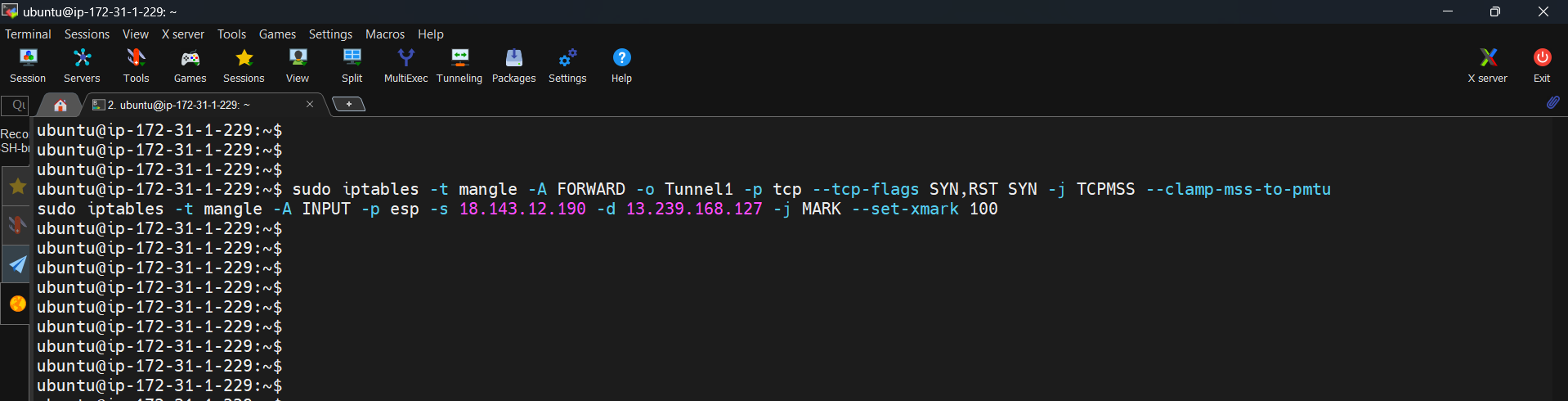
iptables Configuration

iptables is a program designed to act as a firewall for the Linux kernel. It can be used to set up, maintain, and inspect packet filter values entered into several different tables.

iptables rules must be set when using tunnel interfaces so the Linux kernel knows to forward and accept packets on the logical interface. The "--set-xmark" value MUST match the integer you placed as the "mark" value in your configuration file.

sudo iptables -t mangle -A FORWARD -o Tunnel1 -p tcp --tcp-flags SYN,RST SYN -j TCPMSS --clamp-mss-to-pmtu

sudo iptables -t mangle -A INPUT -p esp -s 18.143.12.190 -d 13.239.168.127 -j MARK --set-xmark 100



**sysctl Modifications**

In order to use your tunnel interface effectively, you will need to do some additional sysctl modifications.

1) Open **/etc/sysctl.conf** and append the following values to the end of the file. Replace <PHYSICAL INTERFACE> with the name of the physical interface your logical tunnel interface resides on (e.g. eth0).

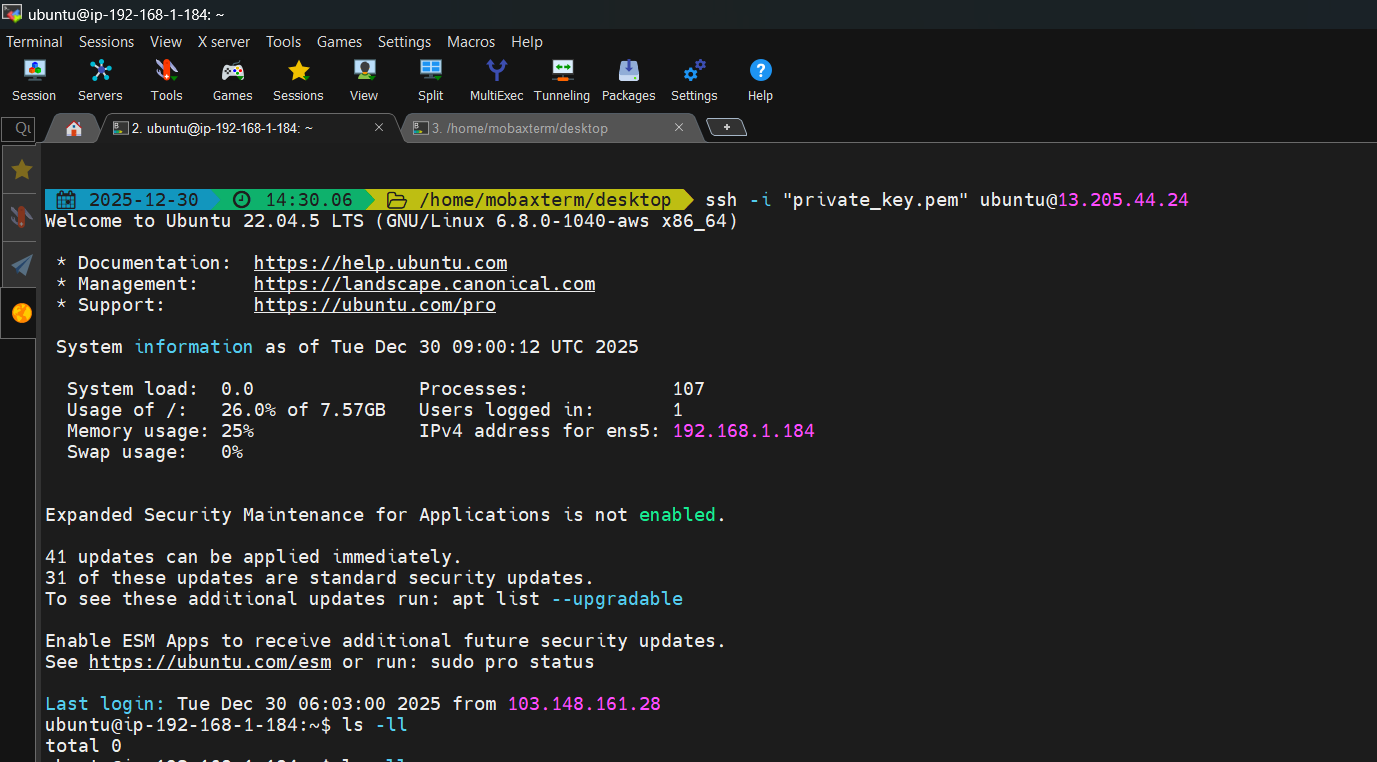
net.ipv4.conf.Tunnel1.rp\_filter=2 #This value allows the Linux kernel to handle asymmetric routing

net.ipv4.conf.Tunnel1.disable\_policy=1 #This value disables IPsec policy (SPD) for the interface

net.ipv4.conf.<PHYSICAL INTERFACE>.disable\_xfrm=1 #This value disables crypto transformations on the physical interface

net.ipv4.conf.<PHYSICAL INTERFACE>.disable\_policy=1 #This value disables IPsec policy (SPD) for the interface

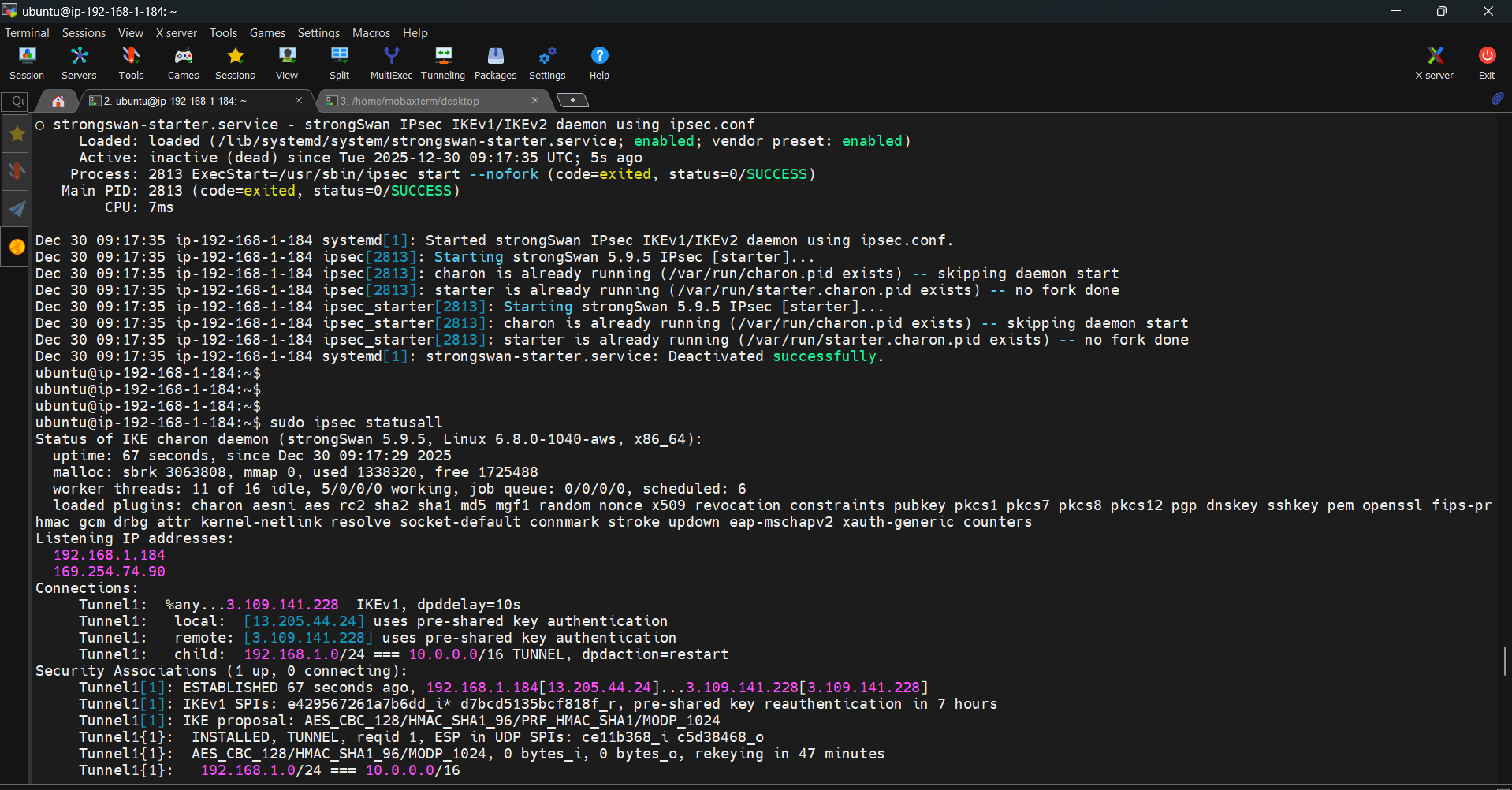
2) Apply the changes in step 1 by executing the command **'sudo sysctl -p'**



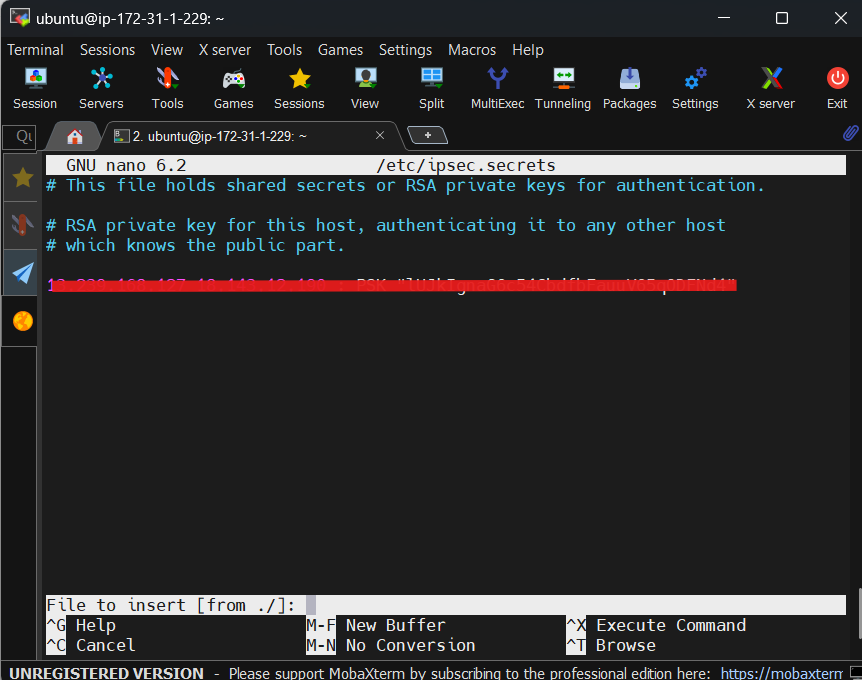


4) Create a new file at /etc/ipsec.secrets if it doesn't already exist, and append this line to the file (be mindful of the spacing!). This value authenticates the tunnel endpoints:

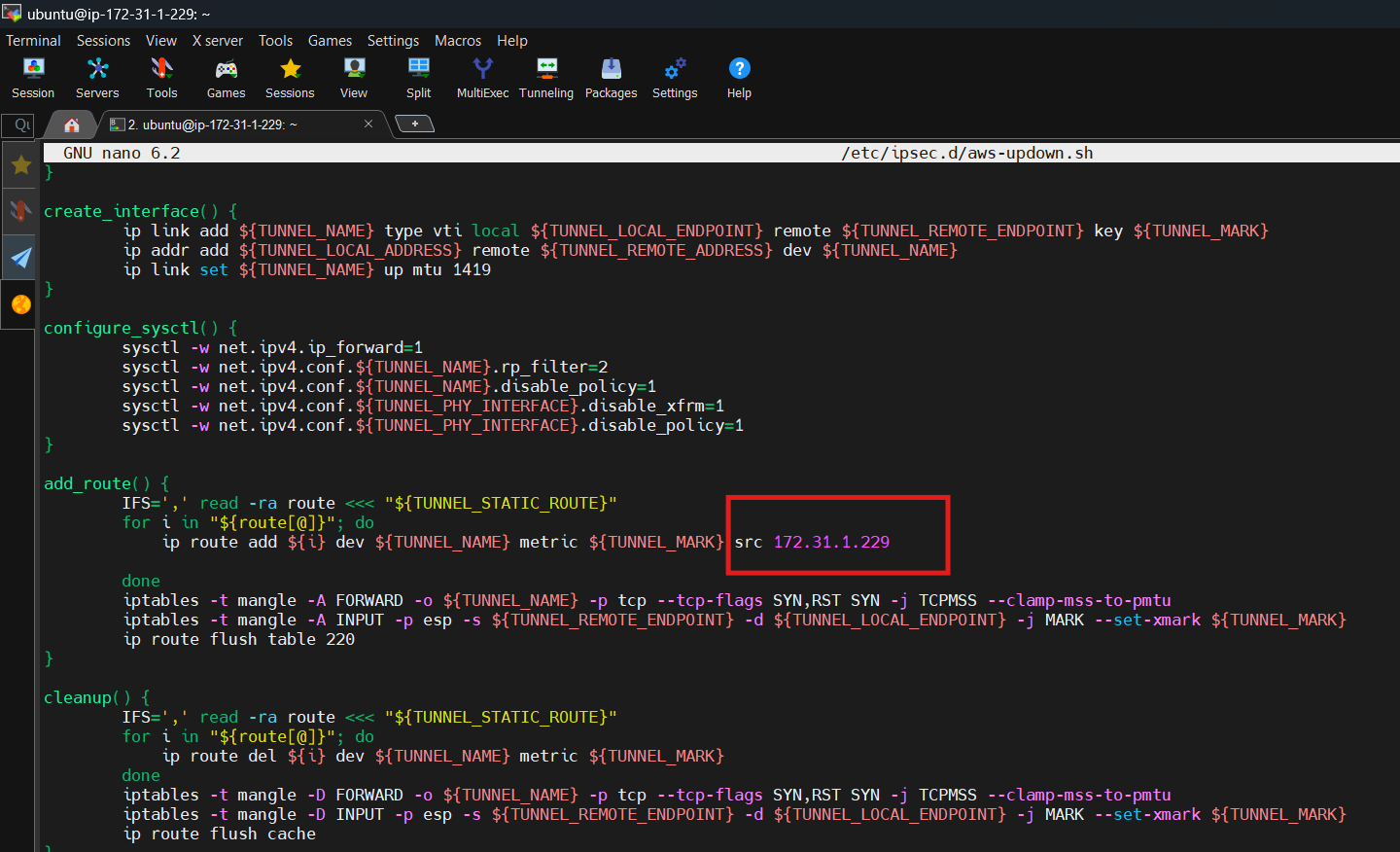
13.239.168.127 18.143.12.190 : PSK "lUJkIgnaG6c54CbdfbFauuV65qODFNd4"



4. Create a new file at /etc/ipsec.secrets if it doesn’t already exist, and append this line to the file (be mindful of the spacing!).

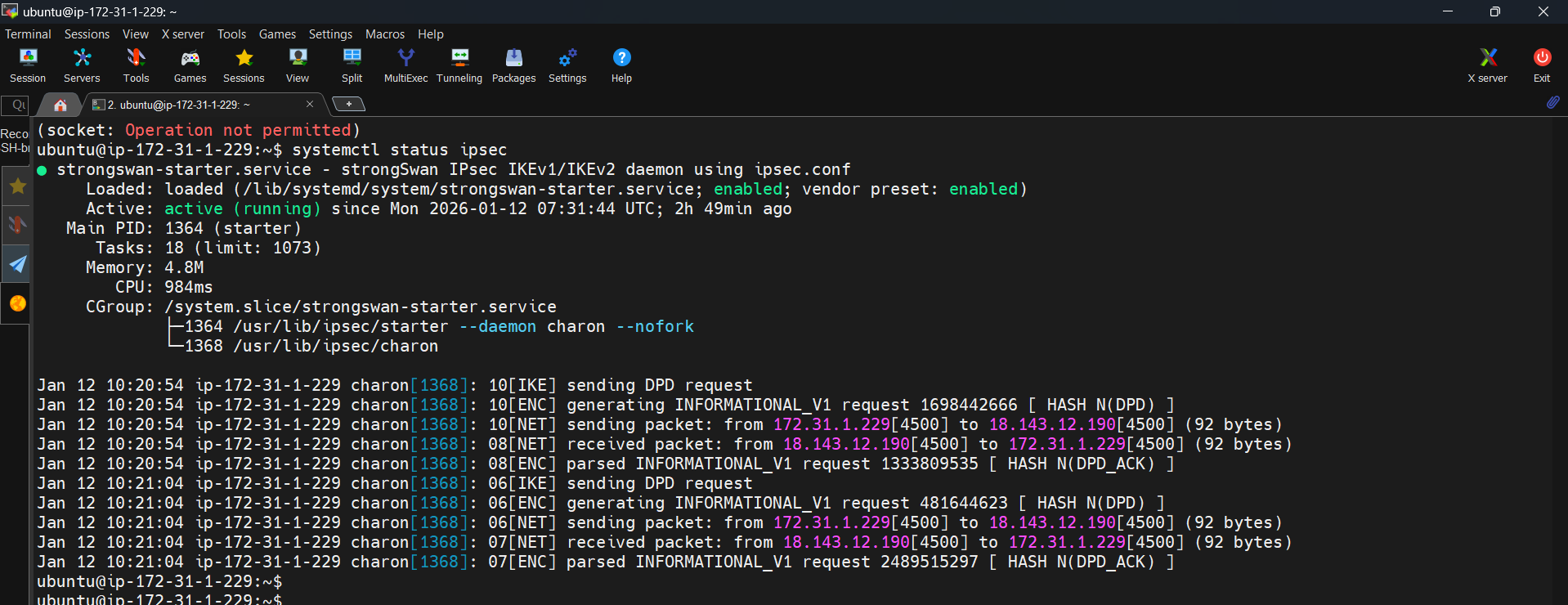


**Tunnel Interface Configuration**



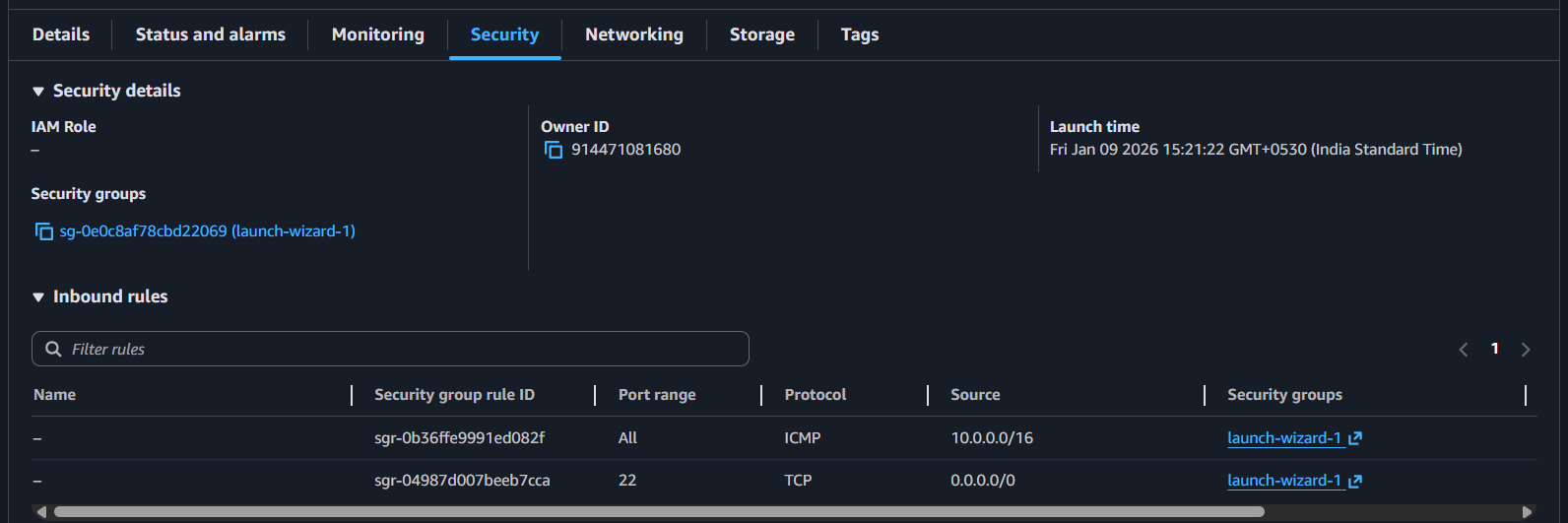
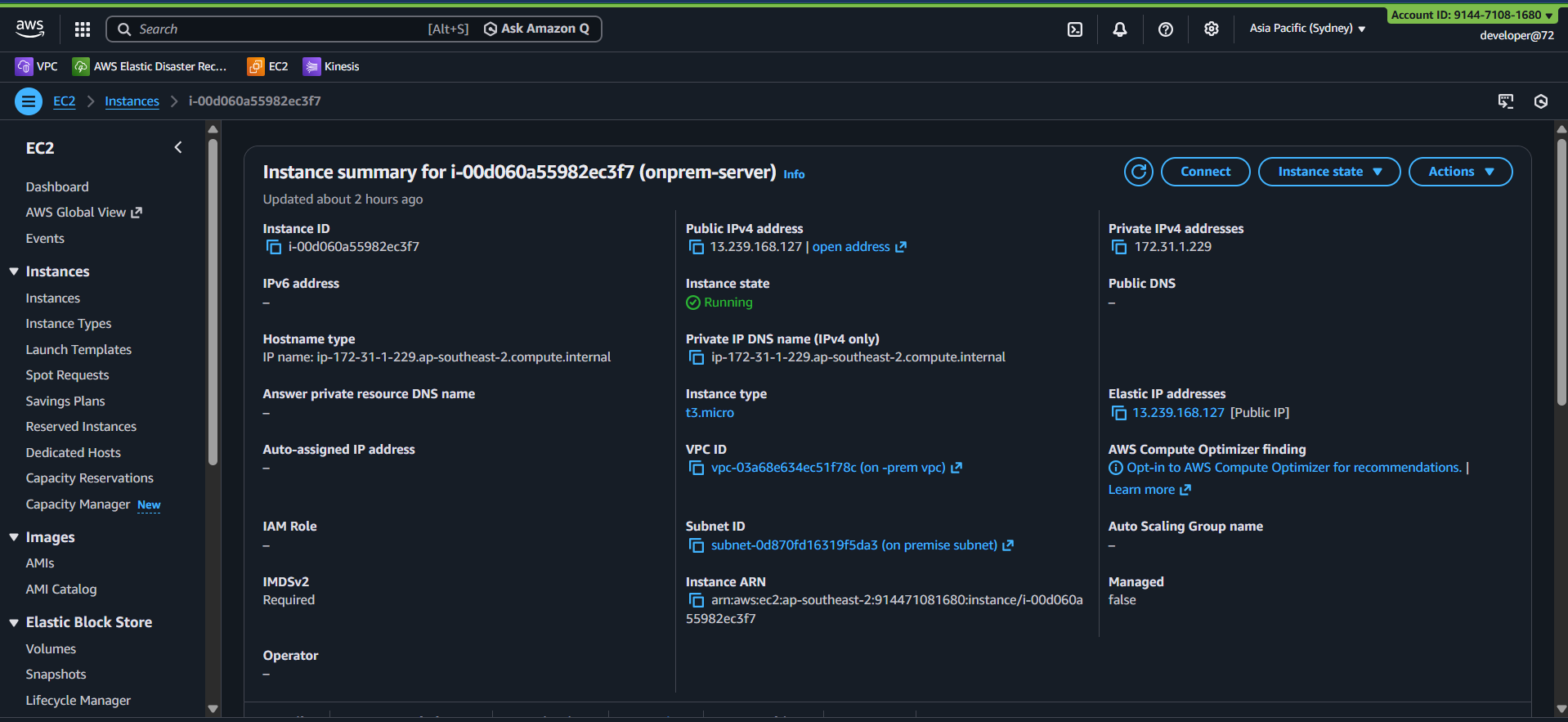
Check the status of the service, it should be active (running).

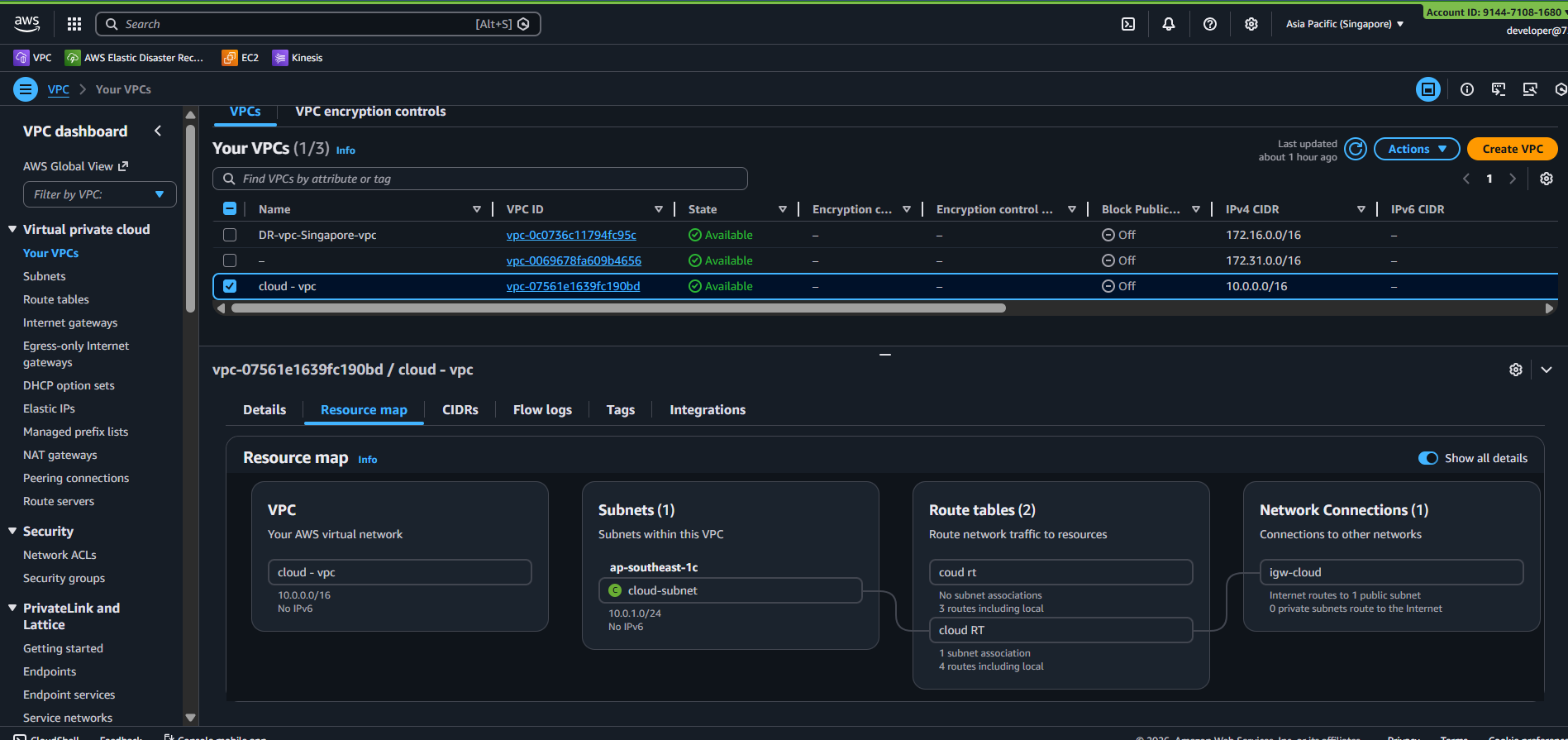
systemctl status ipsec

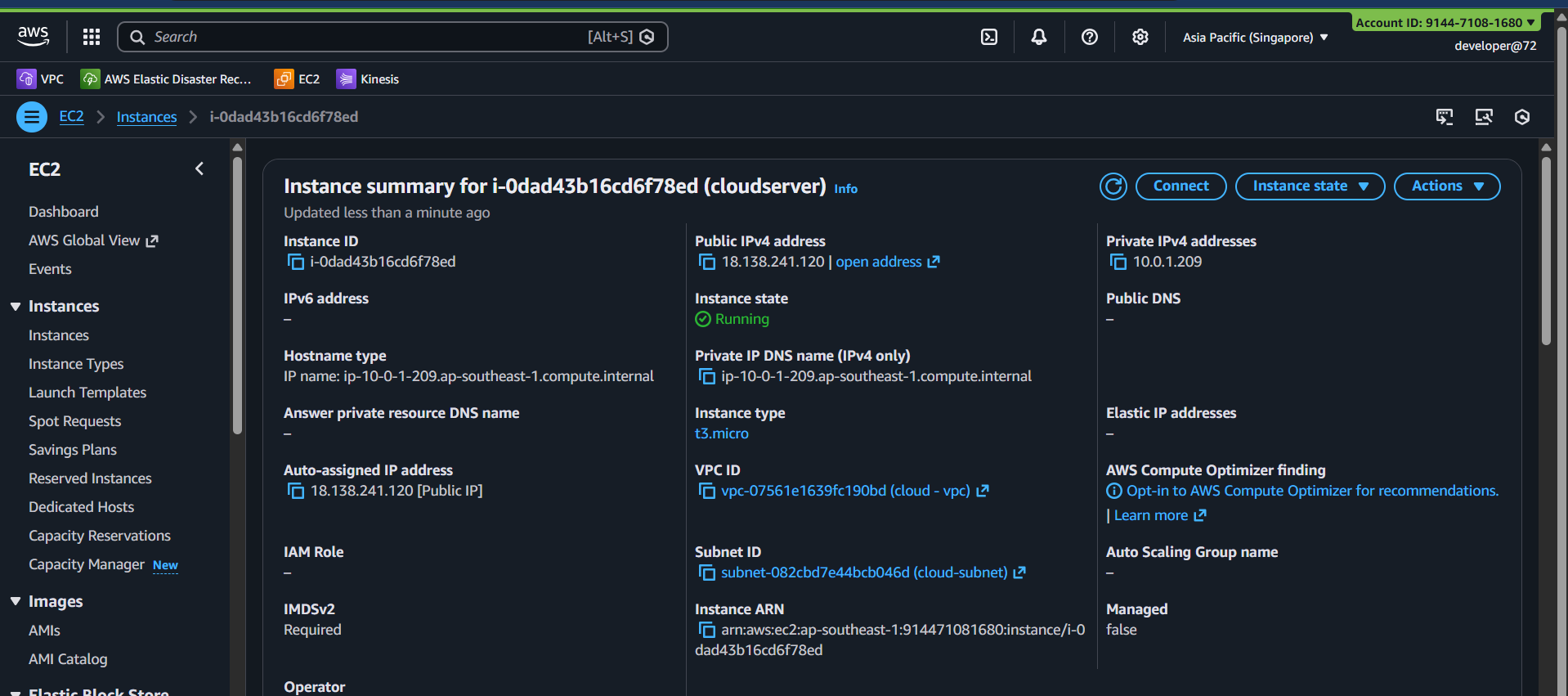


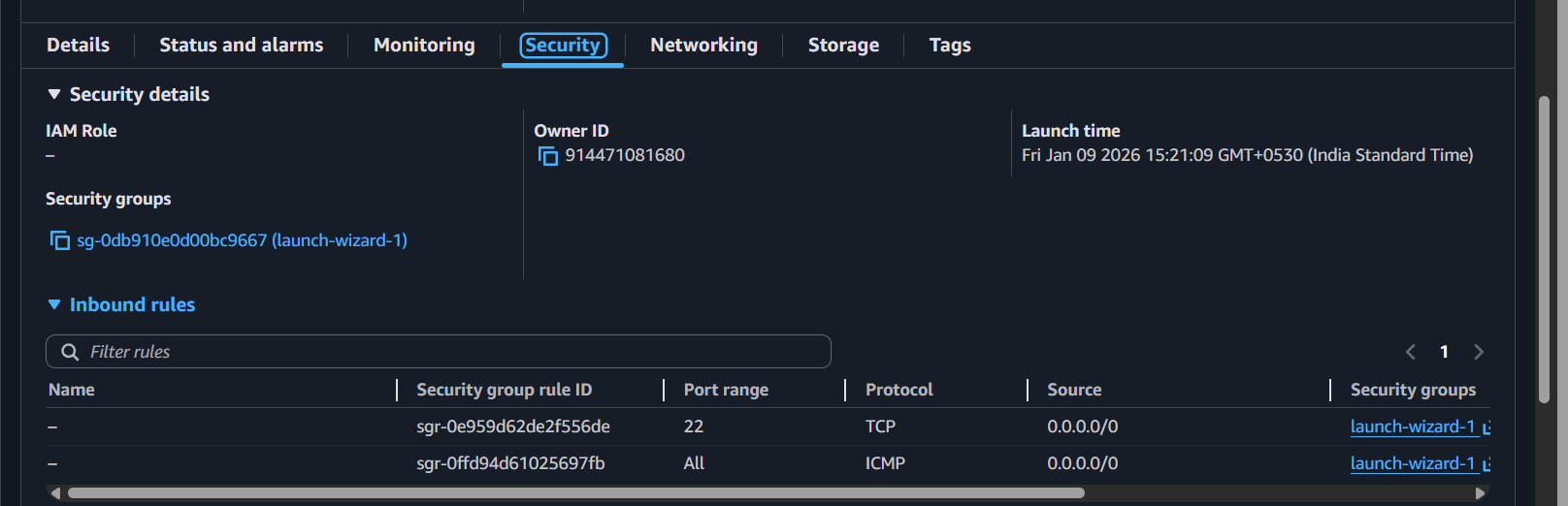
|  |  |
| --- | --- |
| **Component** | **Details** |
| On-prem VPN device | Ubuntu EC2 instance |
| VPN software | strongSwan (IKEv1) |
| AWS Gateway | Virtual Private Gateway (VGW) |
| Tunnel type | Policy-based IPSec |
| Encryption | ESP |
| Traffic type | Private IP traffic only |

| **Side** | **IP / CIDR** |
| --- | --- |
| Linux EC2 (inside) | 172.31.1.229 |
| Linux EC2 (public/EIP) | 13.239.168.127 |
| AWS VPN endpoint | 18.143.12.190 |
| AWS private subnet | 10.0.1.0/24 |
| Test instance | 10.0.1.209 |

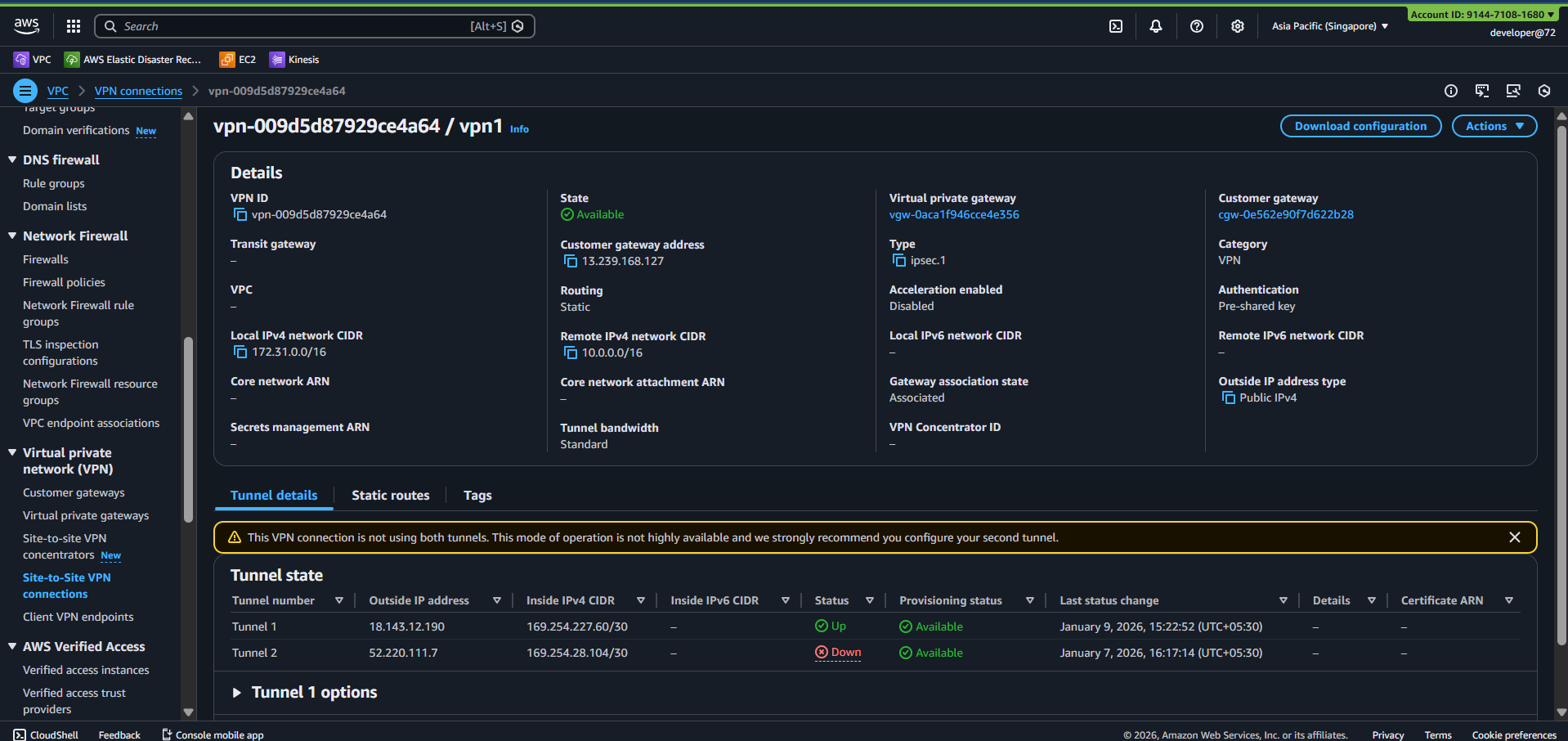


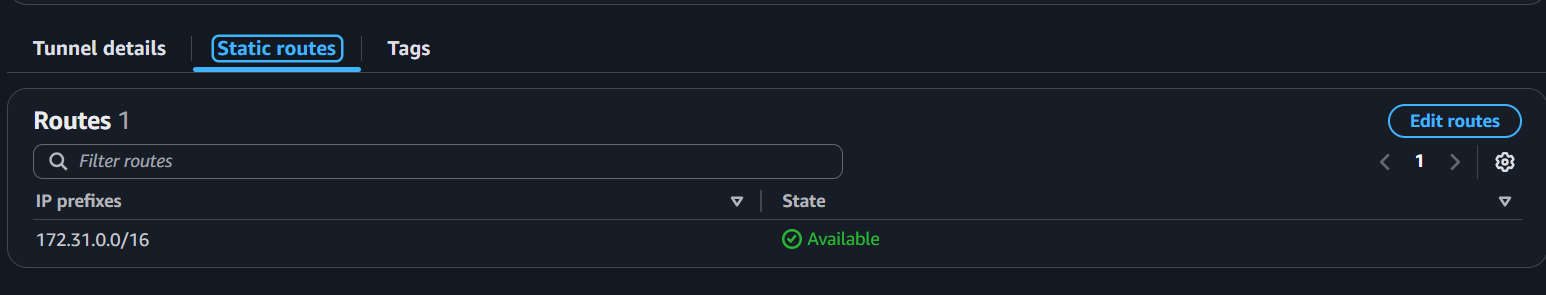


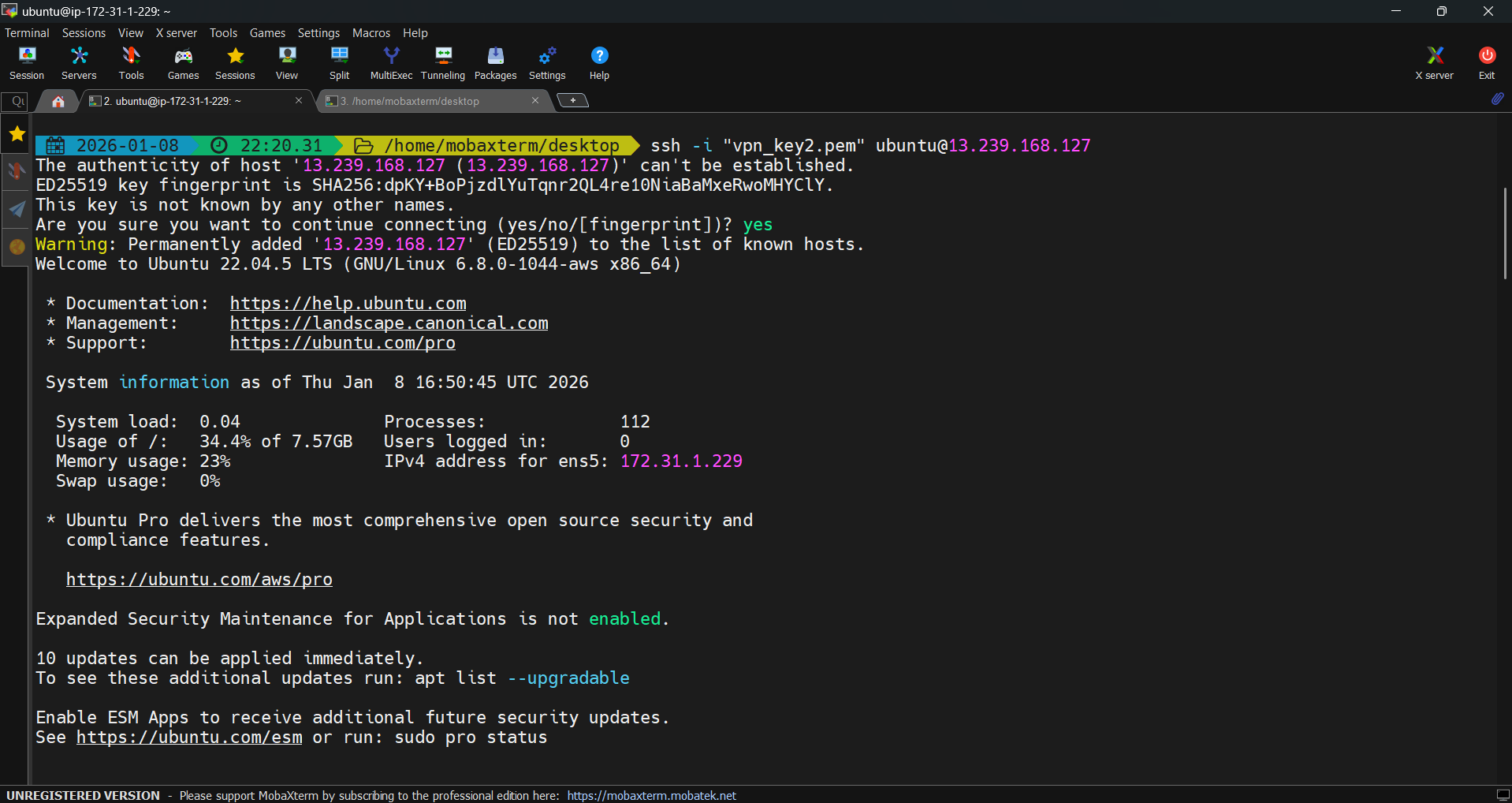


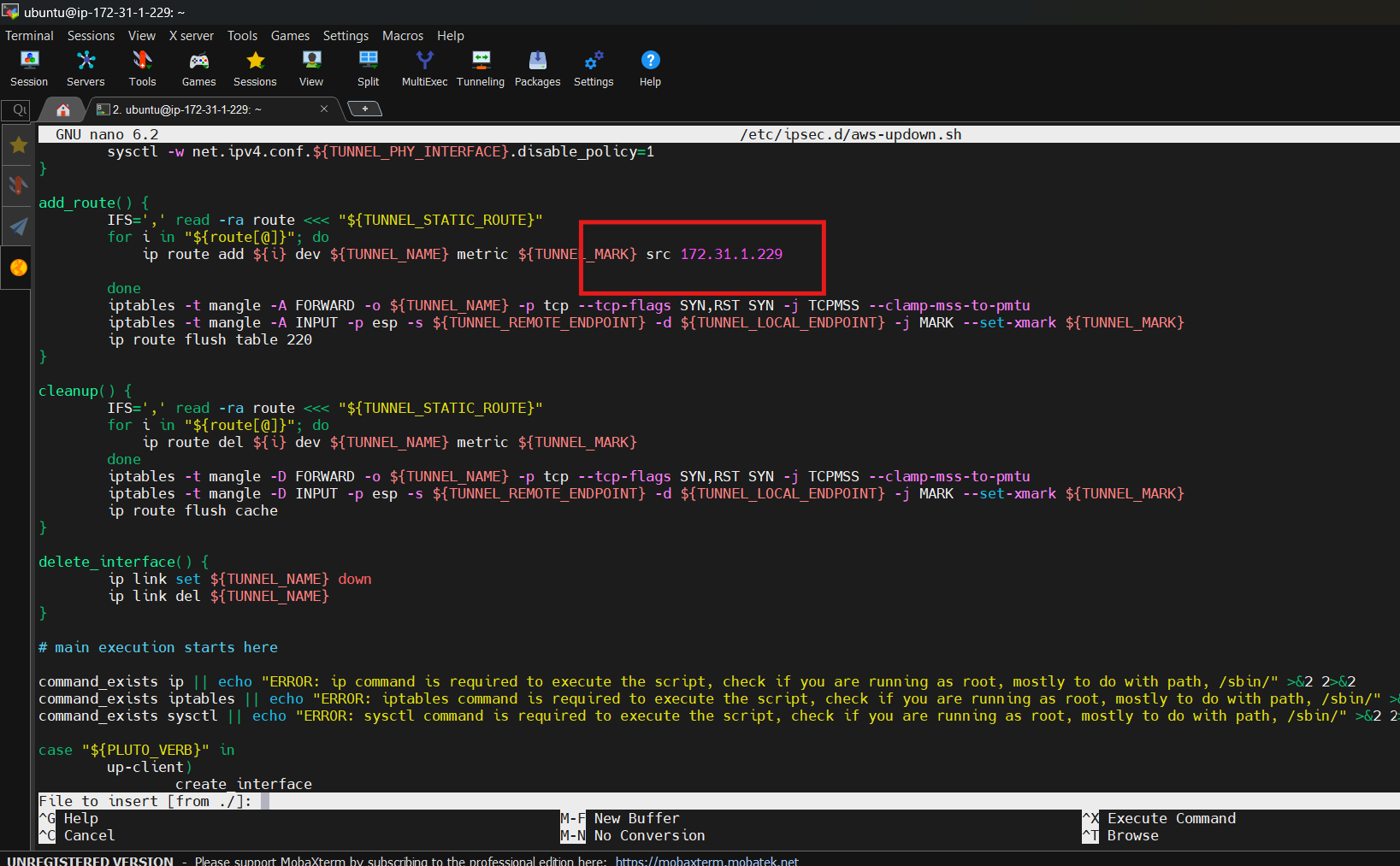


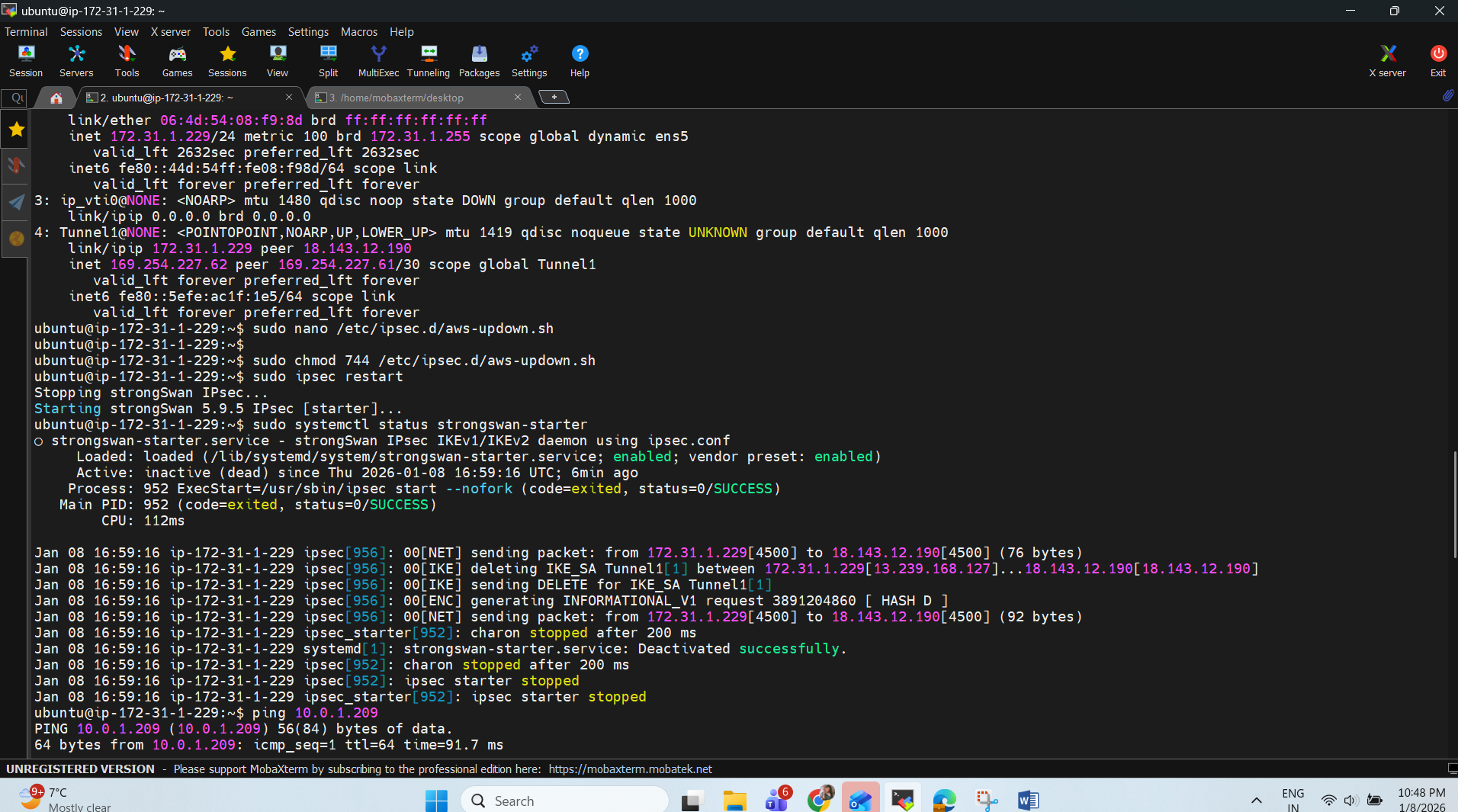


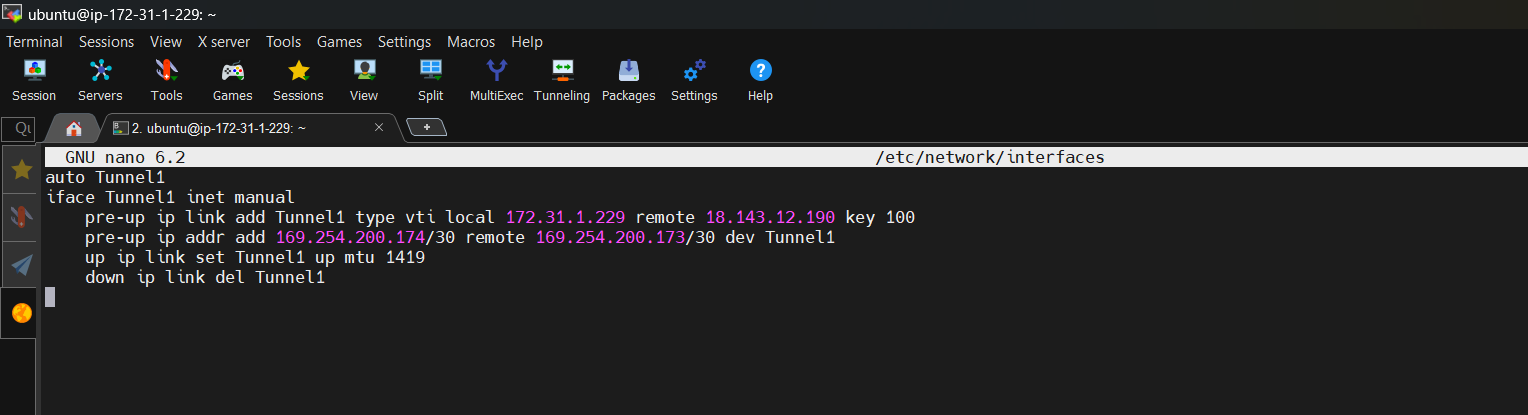


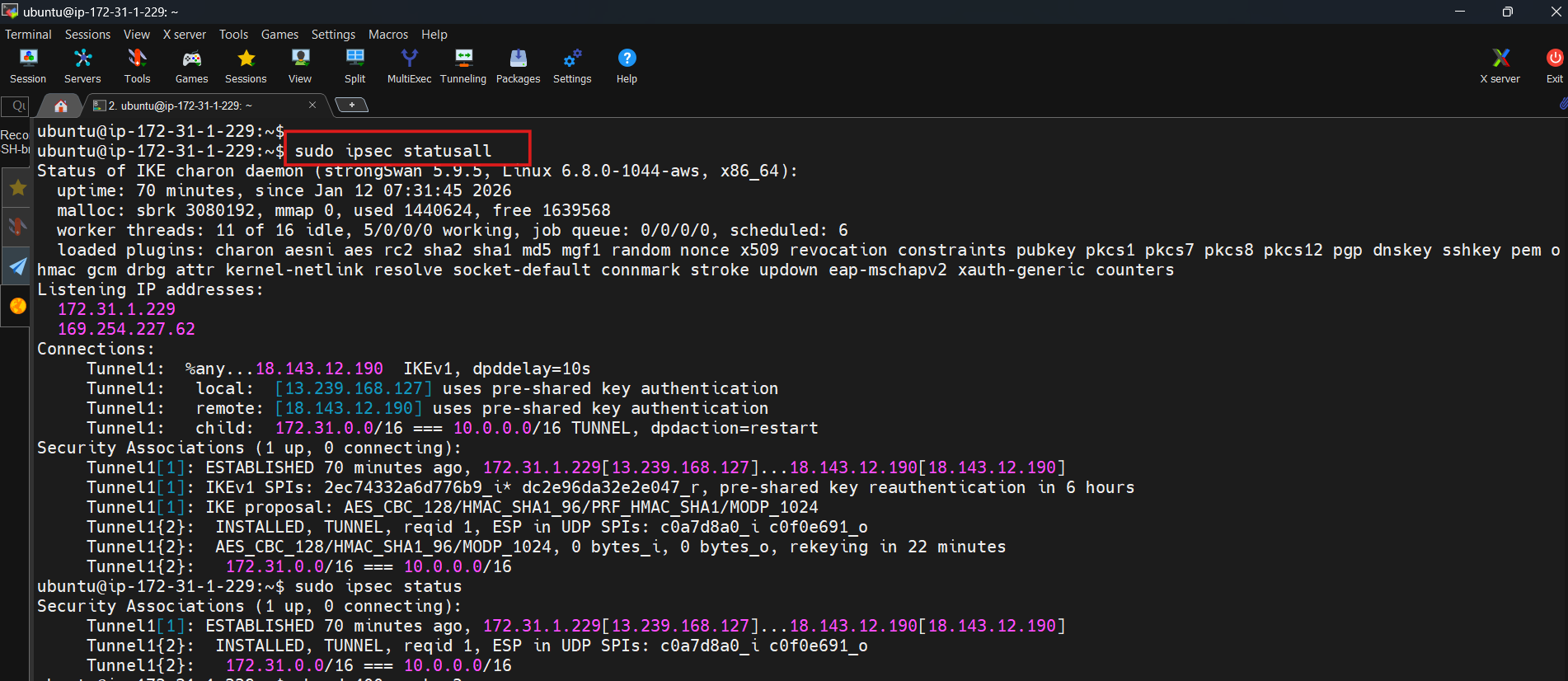


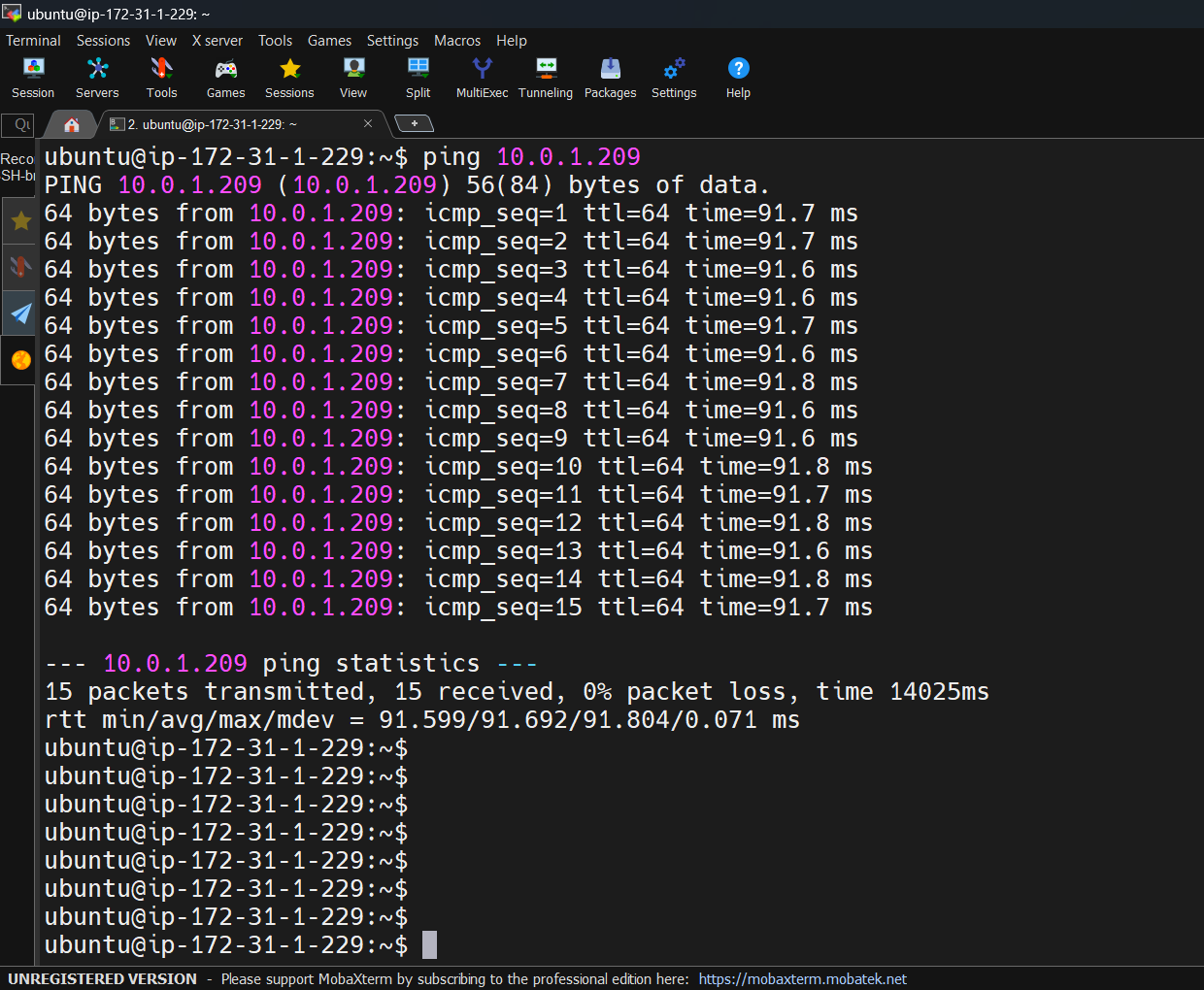








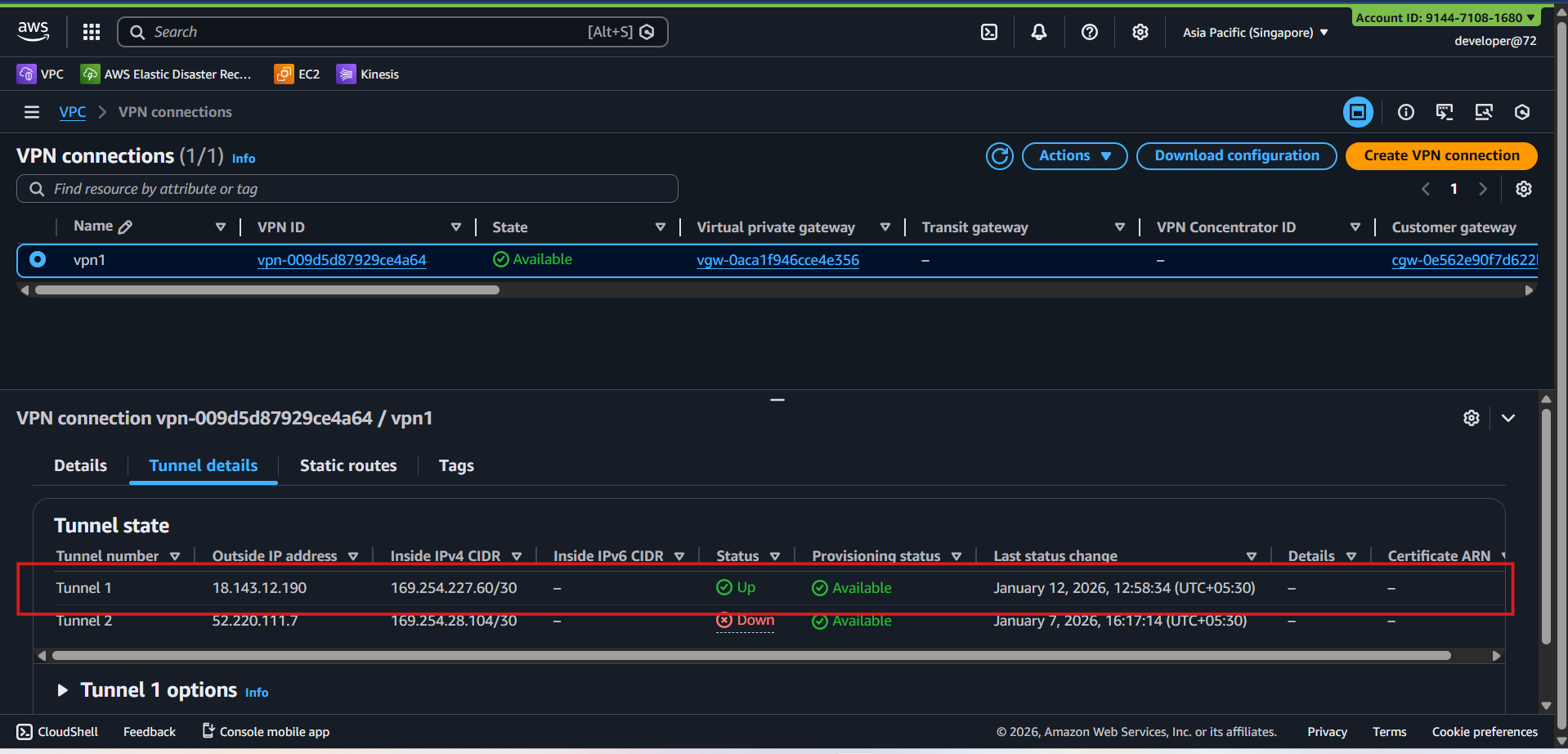




With the above, we have successfully configured Tunnel1. You can follow the steps mentioned on your download configuration file to configure the Tunnel2.

**Navigate back to Singapore (Cloud Region)**

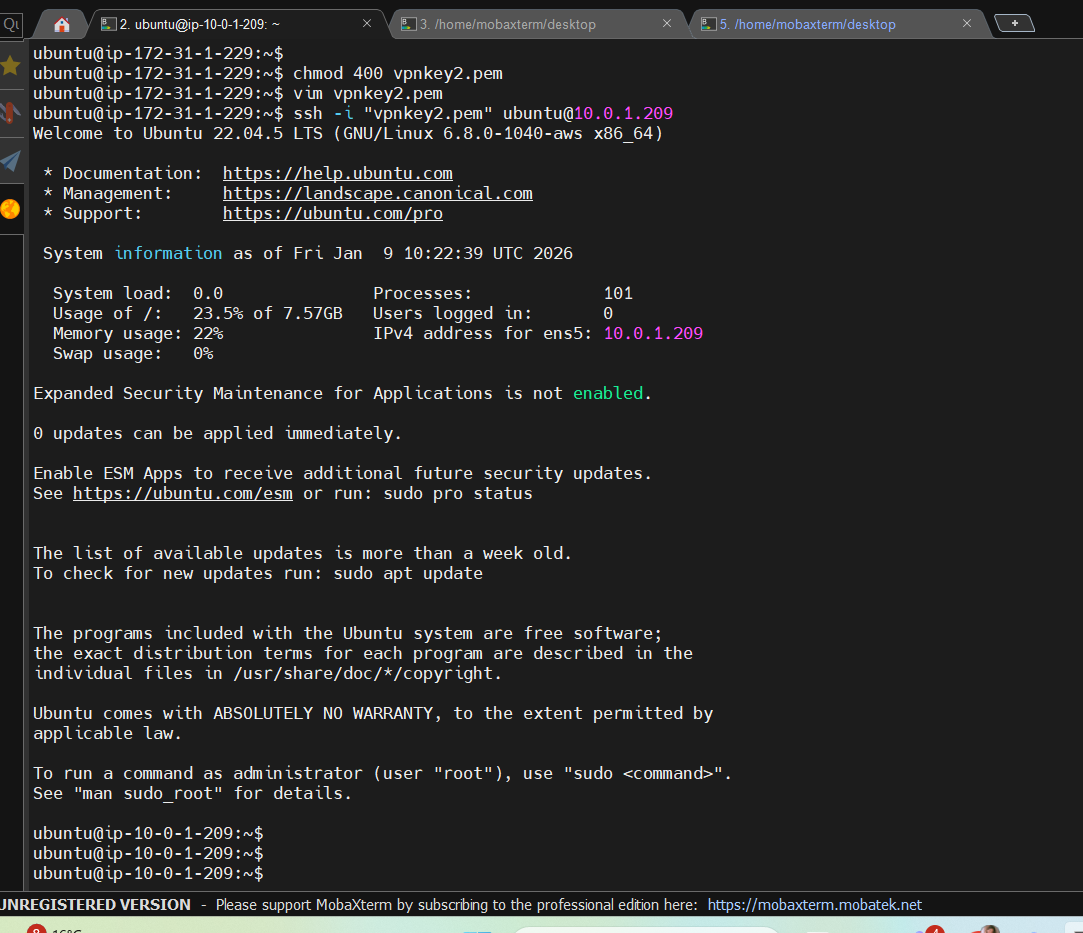
Check the VPN connection tunnel detail. Go to the VPC, Select Site-to-Site VPN connection. It is under VPN. Select the VPN connection and click on Tunnel Details. Tunnel 1 status should be UP.

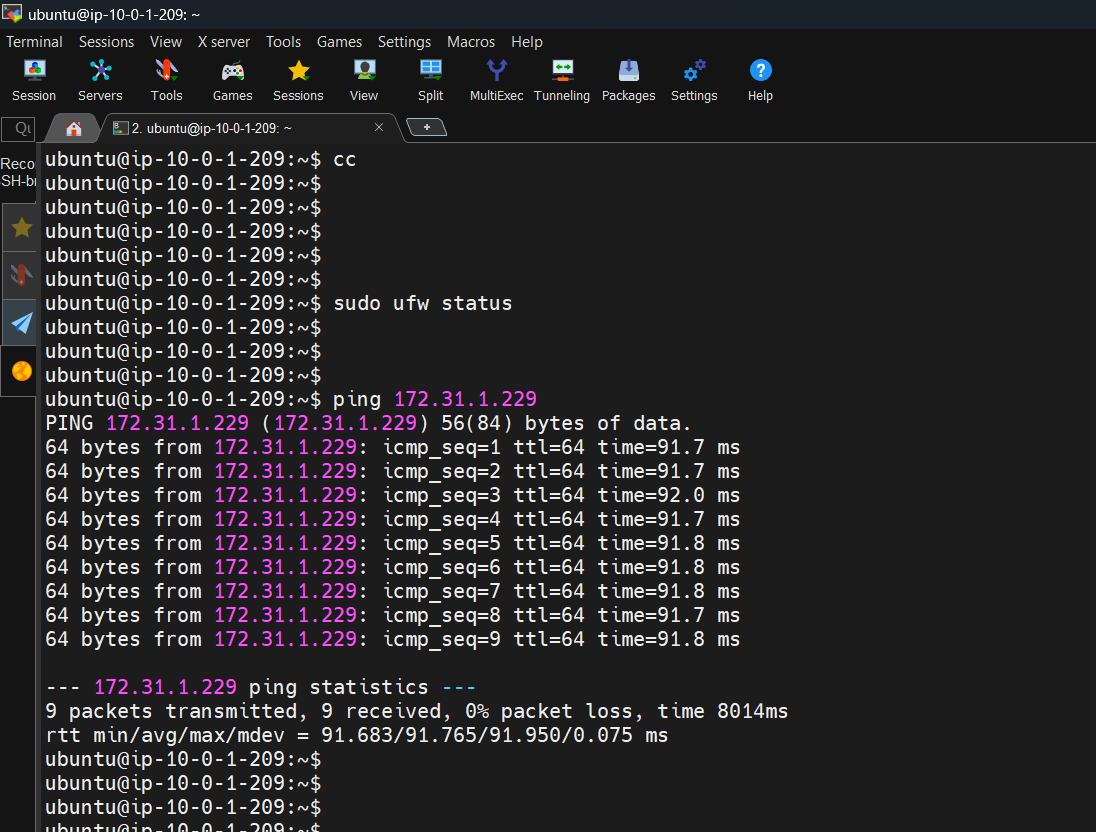


# ****Final Verification Commands****

**Site-to-Site Tunnel de**tails Now SSH into the Cloud Instance and try pinging to the On-prem Instance with the private IP. For this we have already allowed all the inbound traffic in the On-prem instance with the VPC CIDR of the cloud instance, i.e. 10.0.0.0/16

You will get the ping response as follow





This ss is showing ping of my public server private ip from my cloud server.

**R&D Decision Matrix (FINAL FIX LOGIC)**

When ping fails, use this:

|  |  |  |
| --- | --- | --- |
| **Evidence** | **Root Cause** | **Action** |
| VPN DOWN | strongSwan / PSK | Fix IPsec |
| VPN UP + 0 bytes | Route missing | Add VGW route |
| Flow log REJECT | NACL | Fix NACL |
| Flow log ACCEPT, no reply | OS firewall | Disable UFW |
| ACCEPT both ways, still fail | ICMP disabled | sysctl fix |

## AWS LOGGING – THE SINGLE SOURCE OF TRUTH

### Enable VPC Flow Logs (SUBNET level)

**Path**

VPC → Flow Logs → Create flow log

**Settings**

* Resource: **Subnet**
* Subnet: Cloud EC2 subnet
* Filter: ALL
* Destination: CloudWatch Logs
* Log group: /aws/vpc/flowlogs/cloud-subnet

Wait 1–2 minutes.