VIRTUAL PRIVATE NETWORK

We created a **VPN using a Hub-and-Spoke architecture** in Azure.

- In the **Hub VNet**, we configured the **VPN Gateway**.
- In the **Spoke VNet** (where the application server is located), we **peered it with the Hub VNet**.

Our **database is in AWS** (in a VPC).
The Azure App Server (private IP) and AWS DB (private IP) must **communicate privately** even though they are in different cloud data centres.
To achieve this, we use a **VPN connection** between Azure and AWS.

In AWS, we also created a **Virtual Private Gateway (VPG)**.
The traffic flows over the **internet**, but through a **secure encrypted tunnel**.

Three Main Ways to Connect (Azure)

1. Site-to-Site VPN

- This connects **two networks** (Azure VNet ↔ AWS VPC).
- Traffic travels through a **private VPN tunnel**.
- All data is **encrypted and decrypted** over the internet.

2. Point-to-Site VPN

- Used by **individual users**, like Work-From-Home employees.
- They connect to the VPN **on demand**, and disconnect after work.
- This is not used between clouds, but to connect laptops to the cloud.

3. ExpressRoute

- This is a **dedicated private physical connection** (e.g., Hyderabad to Chennai).
- Cloud providers use **partner networks** like Jio, Airtel, Nokia, BSNL to provide it.
- This avoids the public internet and gives higher reliability, but is expensive.

Azure VM ↔ AWS VM Private Communication

Even though the Azure VM and AWS VM are in **different clouds**, we make them communicate privately by:

VIRTUAL PRIVATE NETWORK

In AWS

- Create **VPC**
- Create **subnets**
- Create **Internet Gateway (if needed for outbound access)**
- Create **Route Table**
- Create **Security Groups**
- Launch **EC2 instance**
- Create **AWS VPN Gateway (VPG)**

In Azure

- Create **Hub and Spoke VNets**
- Create **VPN Gateway in Hub**
- Peer **Hub ↔ Spoke**
- App VM lives in Spoke
- Use **Bastion** to log into the Azure VM securely

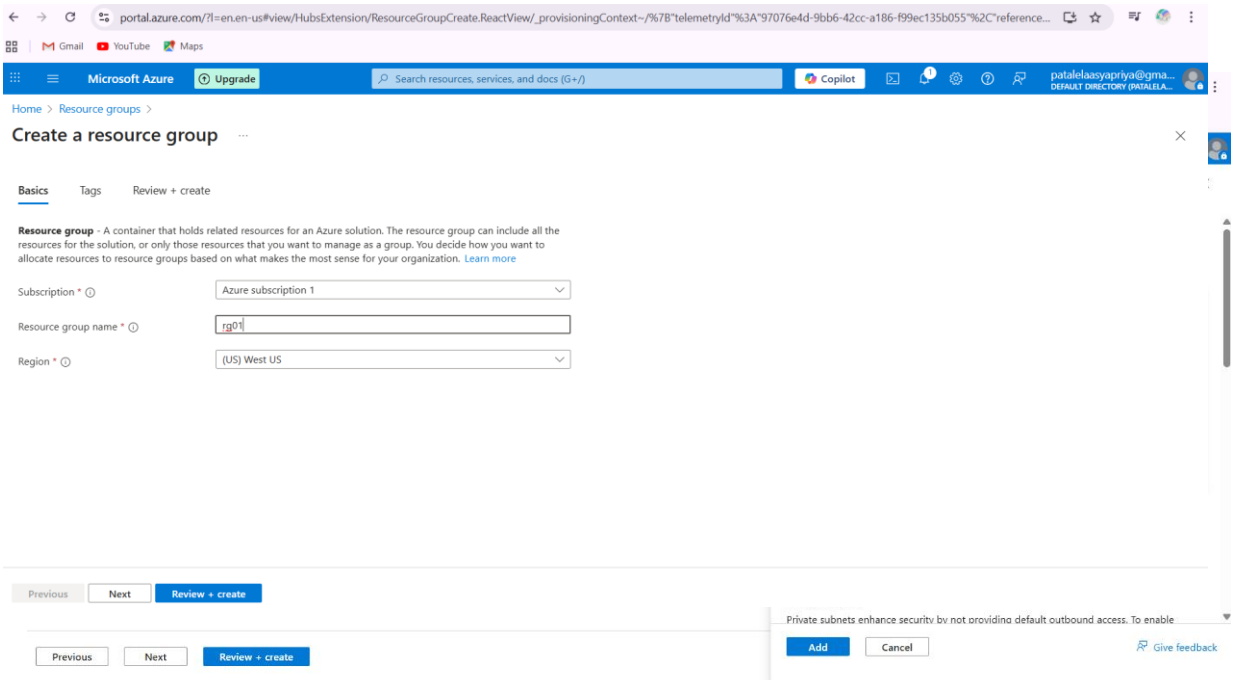
The VPN gateways in Azure and AWS establish a **site-to-site connection**, allowing **private IP-to-private IP** communication.

Enable the option in peering :

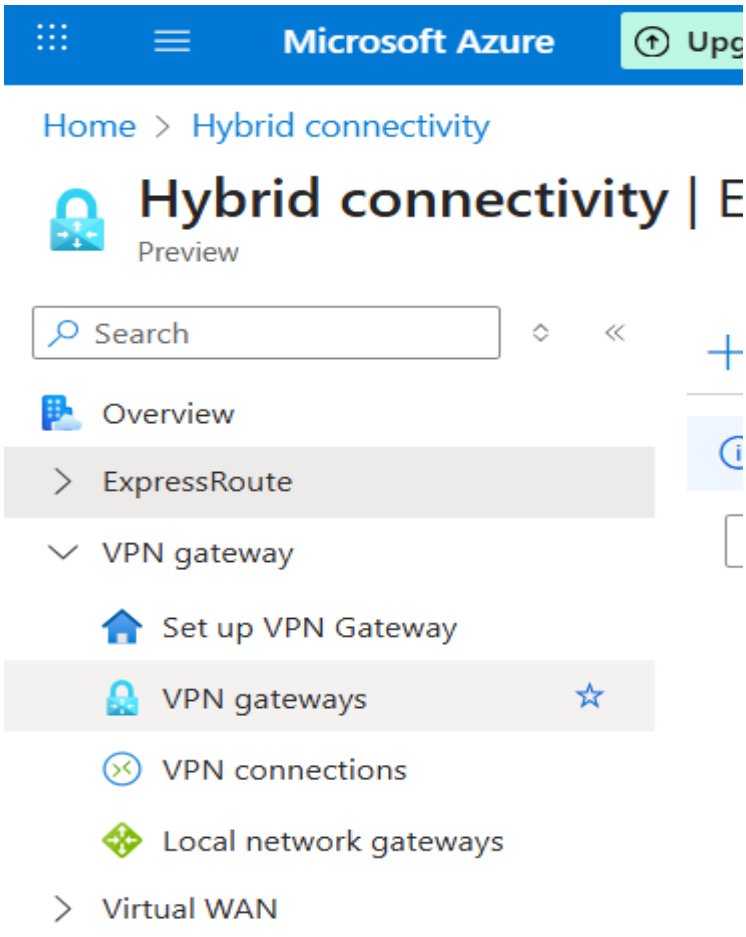
Enable the "**Allow gateway or route server to forward traffic**" option on the VNET peering between VNET01 (Hub/VPN VNet) and VNET02 (App Server VNet) to ensure proper traffic flow between the Azure VPN gateway and the application servers.

VIRTUAL PRIVATE NETWORK

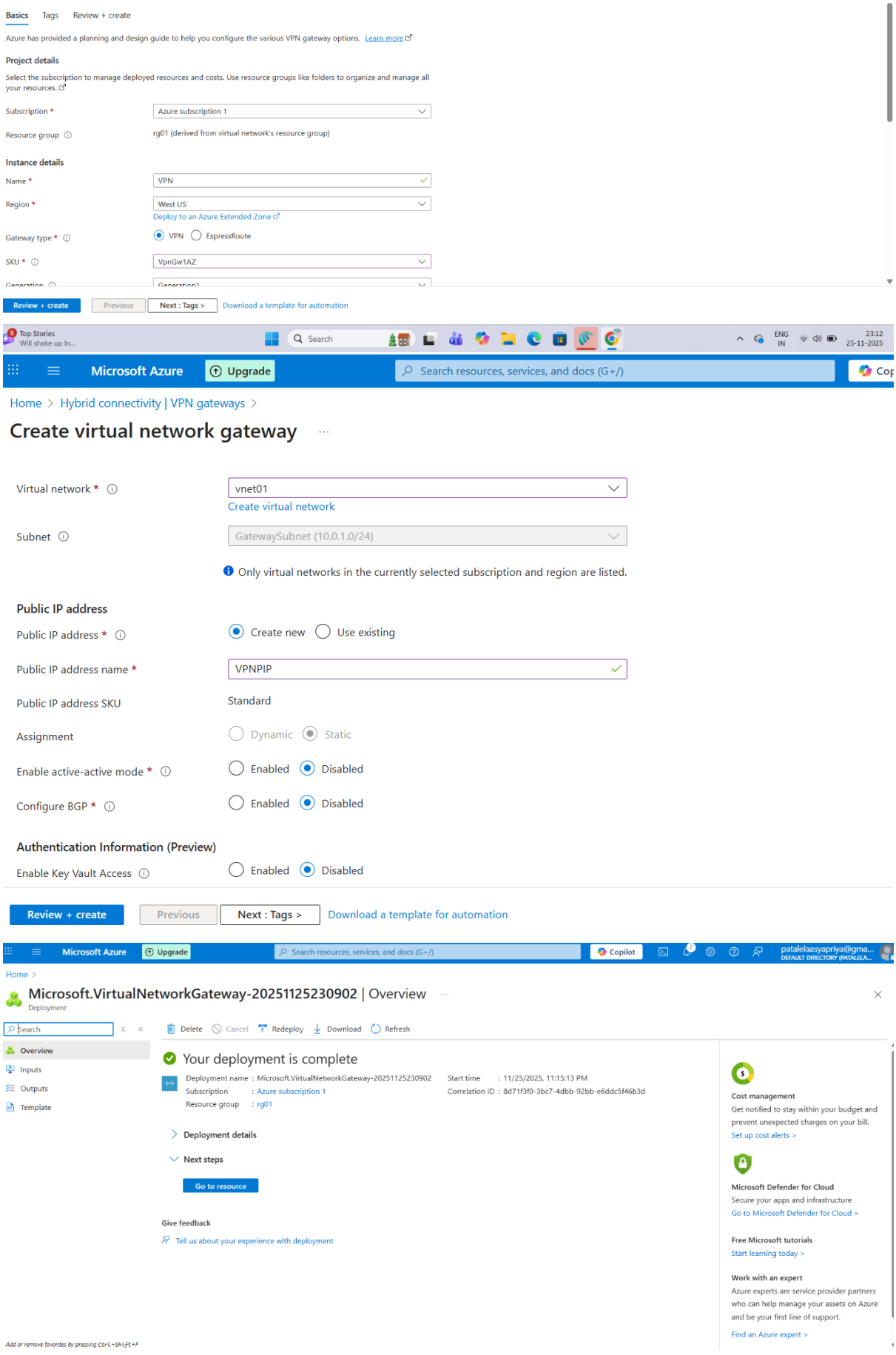
Create RG and VPN Gateway, you create a special subnet called vpnSubnet



Create a VPN



VIRTUAL PRIVATE NETWORK



VIRTUAL PRIVATE NETWORK

AWS Setup — Short Steps (for example)

1. Create VPC

- Go to **VPC → Your VPCs → Create VPC**
- Give name → Add CIDR (example: 10.0.0.0/16) → Create.

2. Create Subnets

- Go to **Subnets → Create subnet**
- Choose the VPC
- Create:

3. Create Internet Gateway (IGW)

- Go to **Internet Gateways → Create**
- Attach it to the VPC.

4. Create Route Tables

  - Add route: 0.0.0.0/0 → Target: IGW

5. Create Security Groups

- Go to **Security Groups → Create**
- Add inbound rules (example: SSH 22 from your IP, HTTP 80 if needed).

6. Launch EC2 Instance

- Go to **EC2 → Launch Instance**
- Select VPC + public subnet
- Enable public IP
- Choose key pair
- Select security group
- Launch.

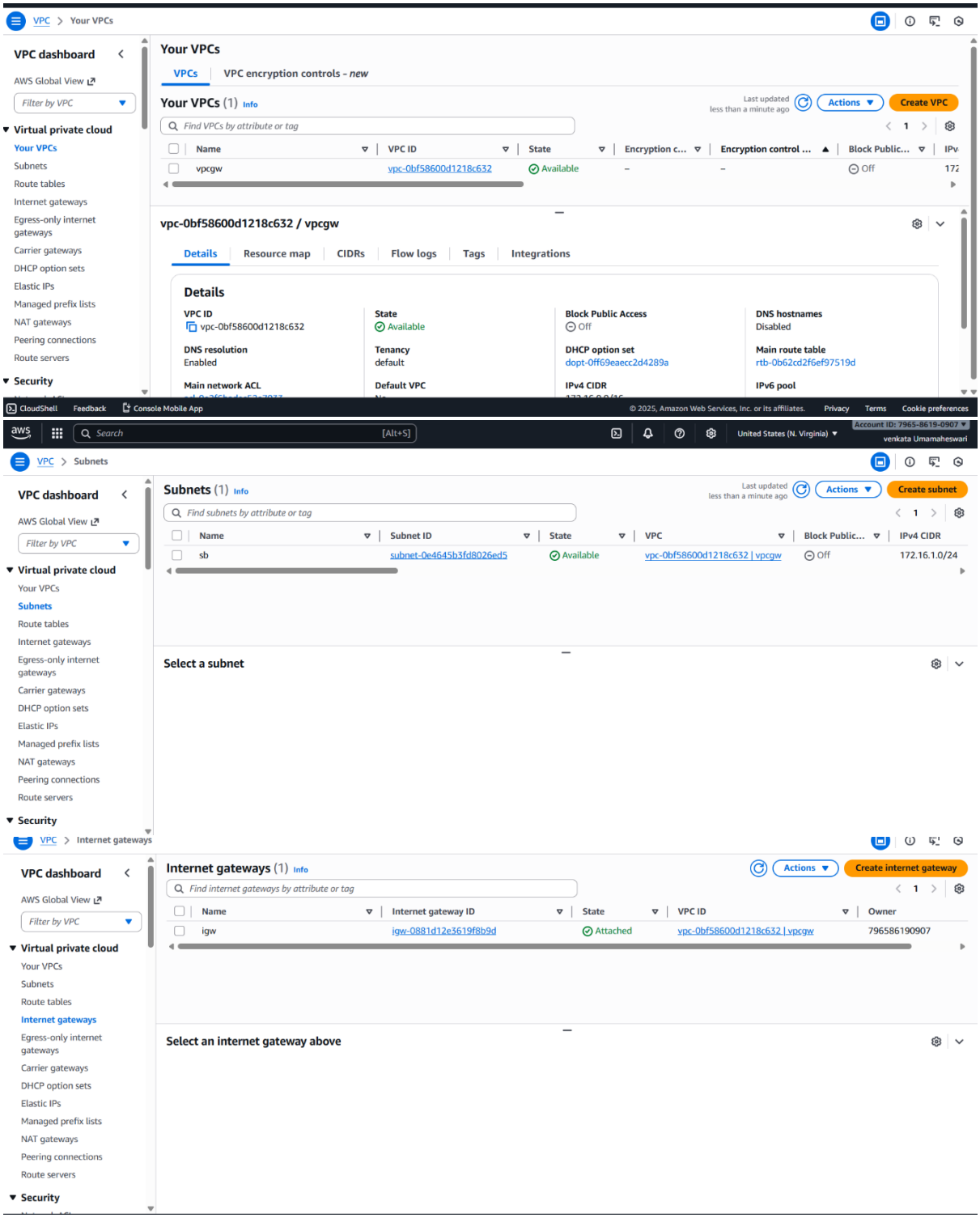
7. Create AWS VPN Gateway (VGW)

- Go to **Site-to-Site VPN → Virtual Private Gateways → Create**
- Attach to VPC
- Create **Customer Gateway** (use Azure VPN public IP)
- Create **VPN Connection** (VGW ↔ Customer Gateway)

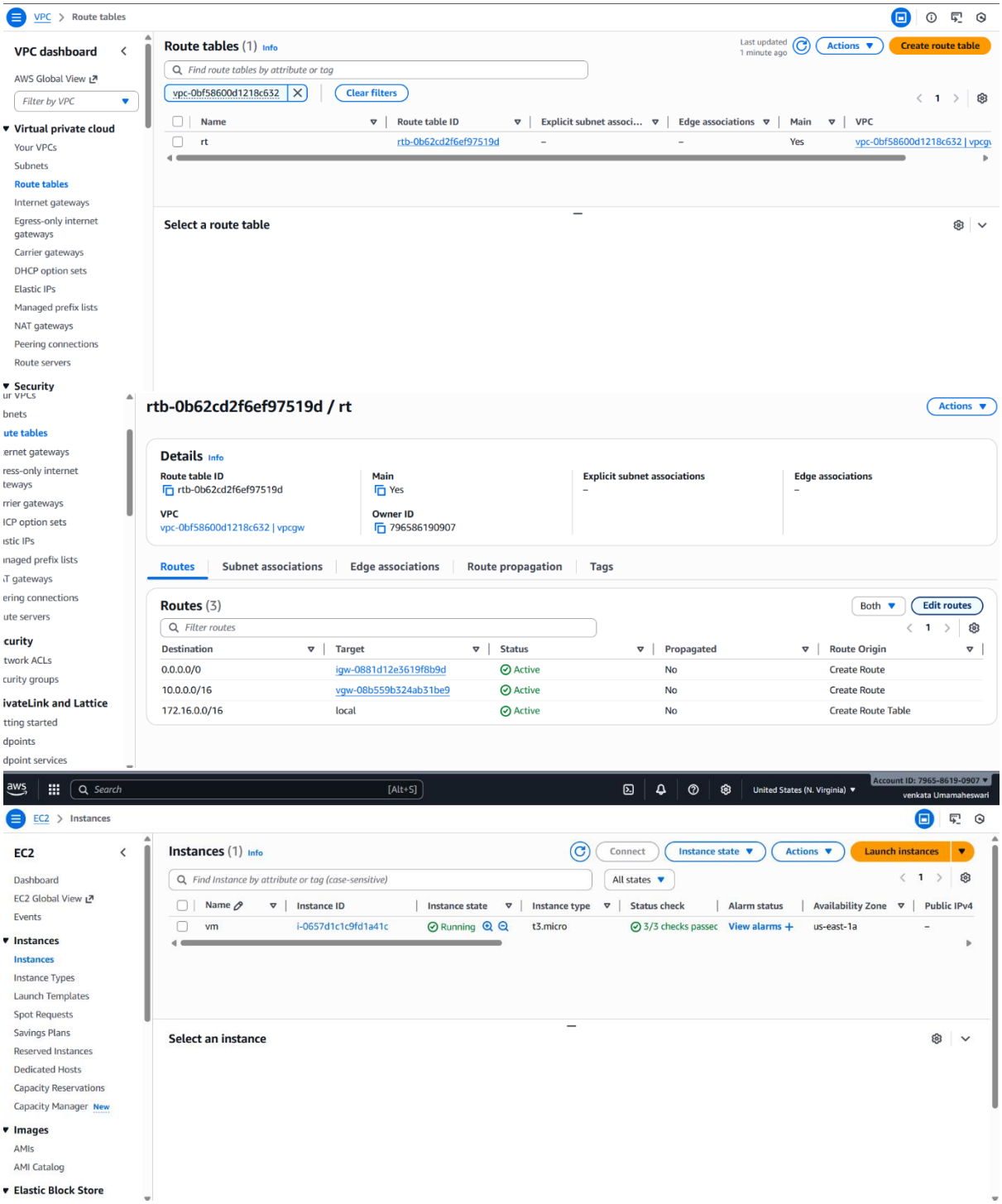
8. Add VPN Route

- In **private route table**, add:
  - Destination = Azure network CIDR

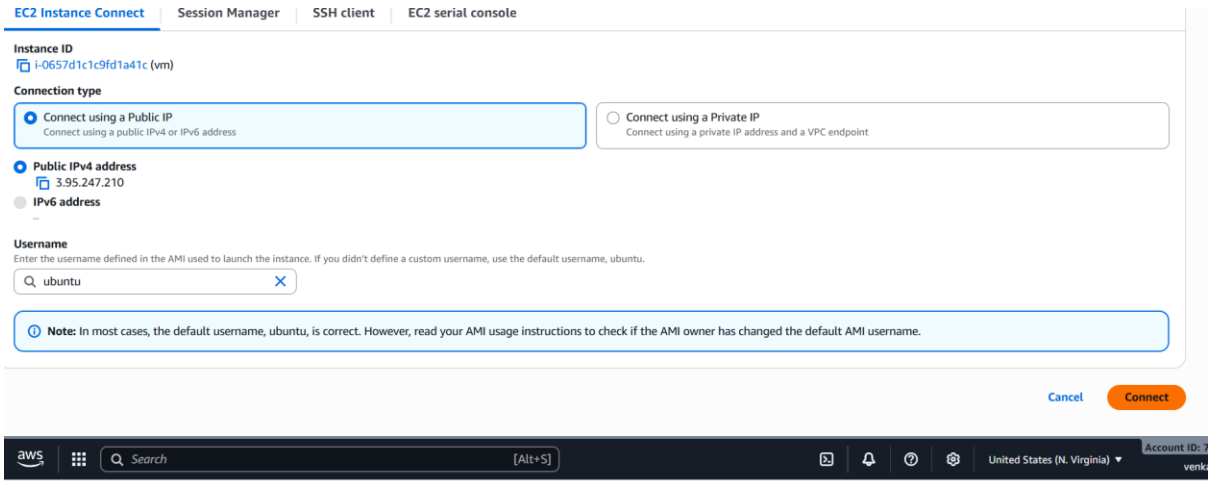
VIRTUAL PRIVATE NETWORK



VIRTUAL PRIVATE NETWORK



VIRTUAL PRIVATE NETWORK

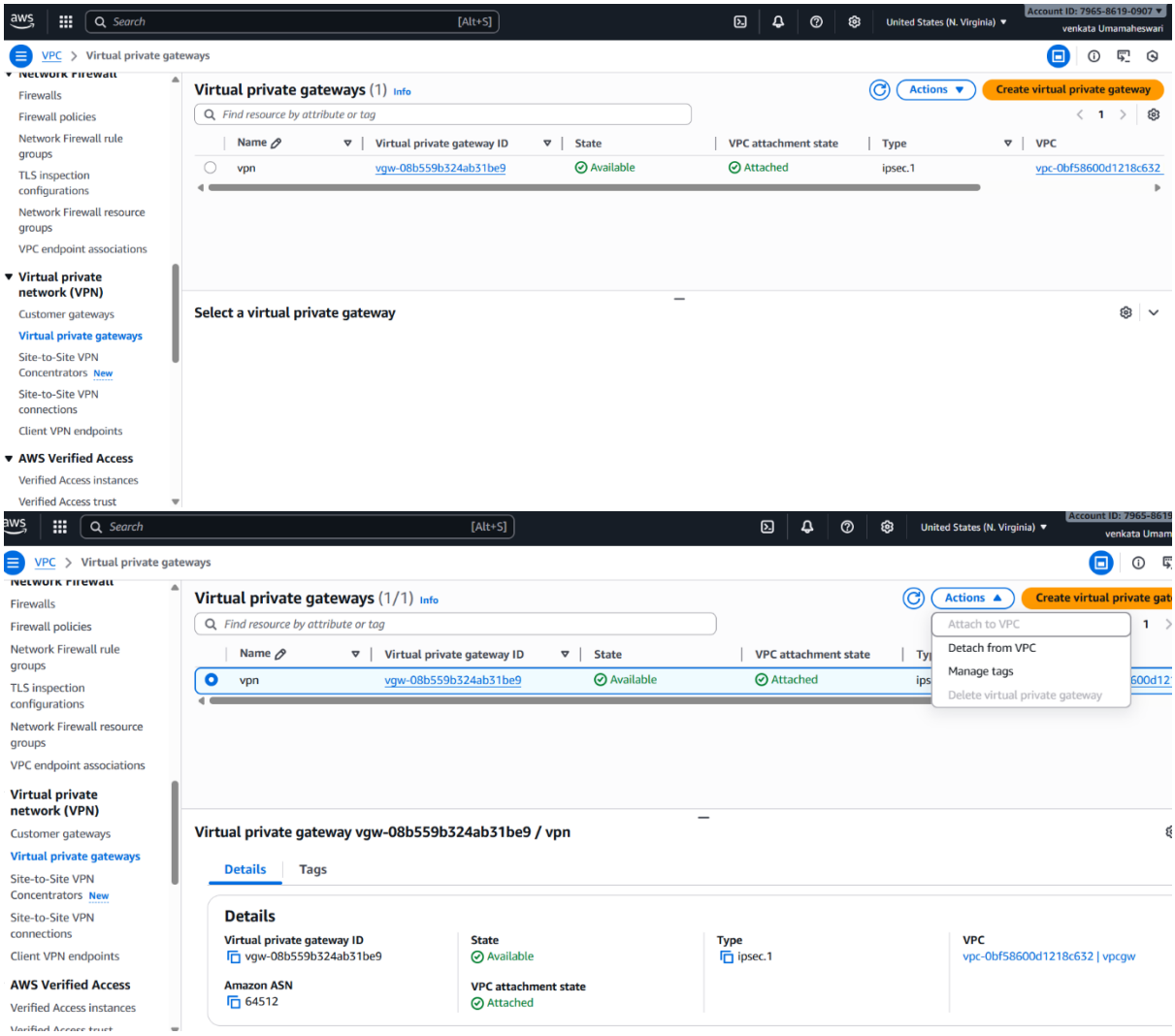



```
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-1015-aws x86_64)

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

 System information as of Tue Nov 25 18:33:13 UTC 2025

  System load:  0.1               Temperature:           -273.1 C
  Usage of /:   34.6% of 6.71GB   Processes:             112
  Memory usage: 28%               Users logged in:       0
  Swap usage:   0%                IPv4 address for ens5: 172.16.1.116


Expanded Security Maintenance for Applications is not enabled.

17 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

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


*** System restart required ***
Last login: Mon Nov 24 06:01:00 2025 from 18.206.107.29
ubuntu@ip-172-16-1-116:~$
```

i-0657d1c1c9fd1a41c (vm)

PublicIPs: 3.95.247.210 PrivateIPs: 172.16.1.116

VIRTUAL PRIVATE NETWORK



In Resource Group 2

- You created another **VNet**
- Inside that VNet you created a **Virtual Machine**

This VNet is your **SPOKE** (your application server lives here).

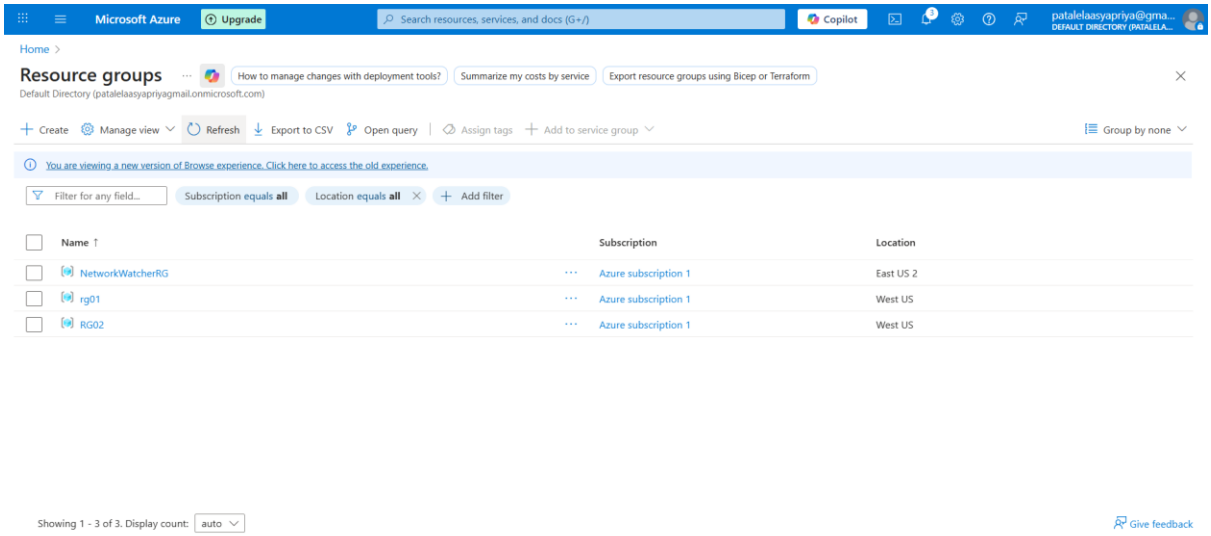
Next Step: VNet Peering

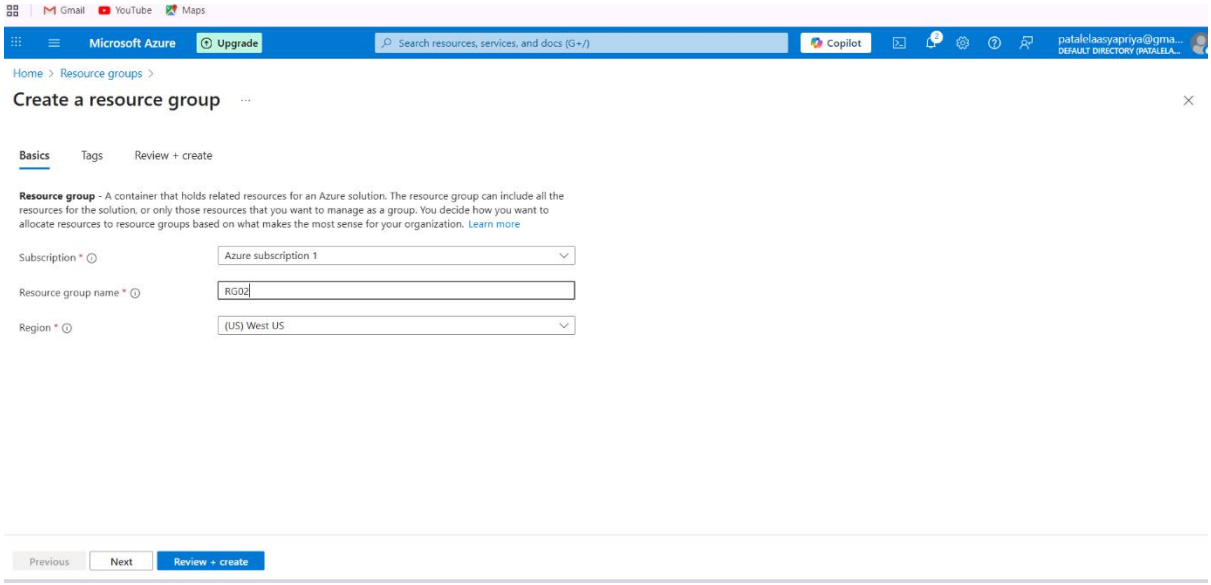
You connected the two VNets (HUB ↔ SPOKE) using **VNet peering**.

While creating the peering, you enabled this important option:

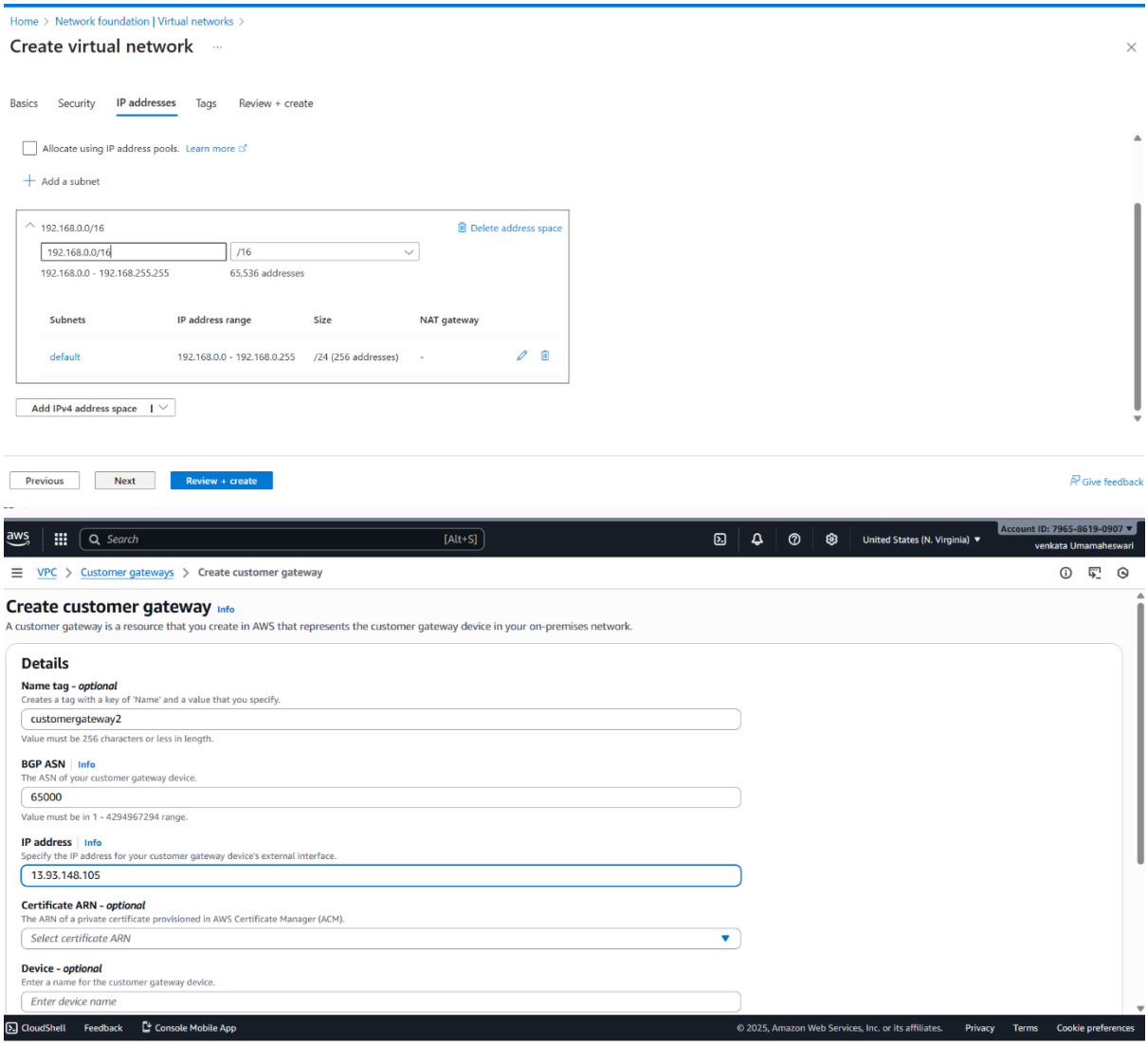
“Allow gateway or route server in ‘VNET01’ to forward traffic to ‘VNET02’ ”
This allows the VM in the SPOKE VNet to use the **VPN Gateway** created in the HUB VNet.

VIRTUAL PRIVATE NETWORK





VIRTUAL PRIVATE NETWORK



VIRTUAL PRIVATE NETWORK

Create Customer Gateway

- Go to **VPC → Customer Gateway → Create**
- Enter:
    - Name: Azure-CGW
    - IP Address: **Azure VPN Gateway private IP**
- Create.

Download the customer gateway configuration →This file contains the Pre-Shared Key (PSK).

Create Local Network Gateway

- Go to Local Network Gateway → Create
- Enter:
    - Name: AWS-LNG
    - Address space: (from the downloaded file)
- Create.

Create VPN Connection

- Go to VPN Gateway → Connections → Add
- Choose:
    - Type: Site-to-site
    - Local network gateway: AWS-LNG
    - Shared key: Paste PSK from AWS file
- Create.

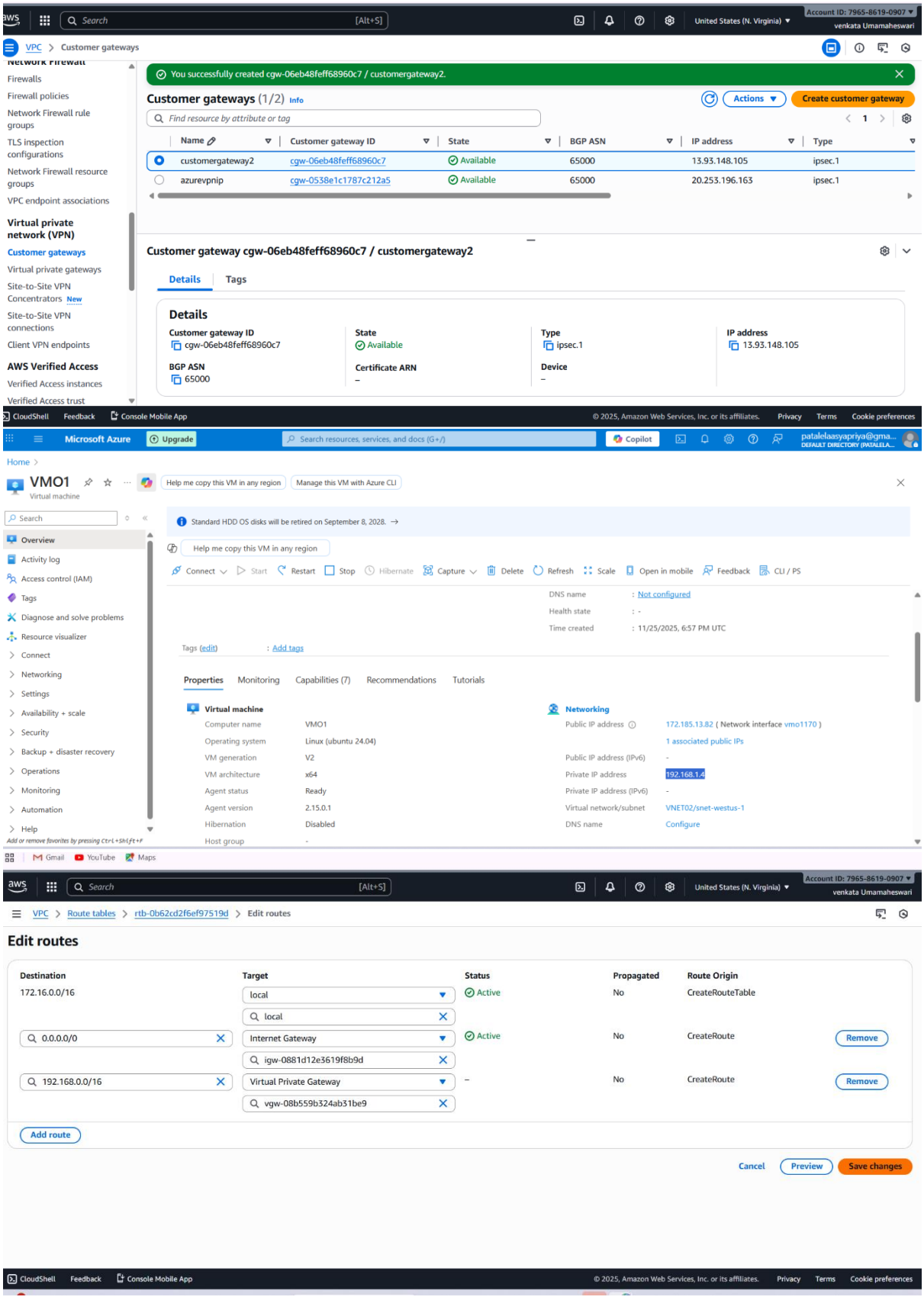
Test

- Go to your AWS EC2 terminal.
- Run:

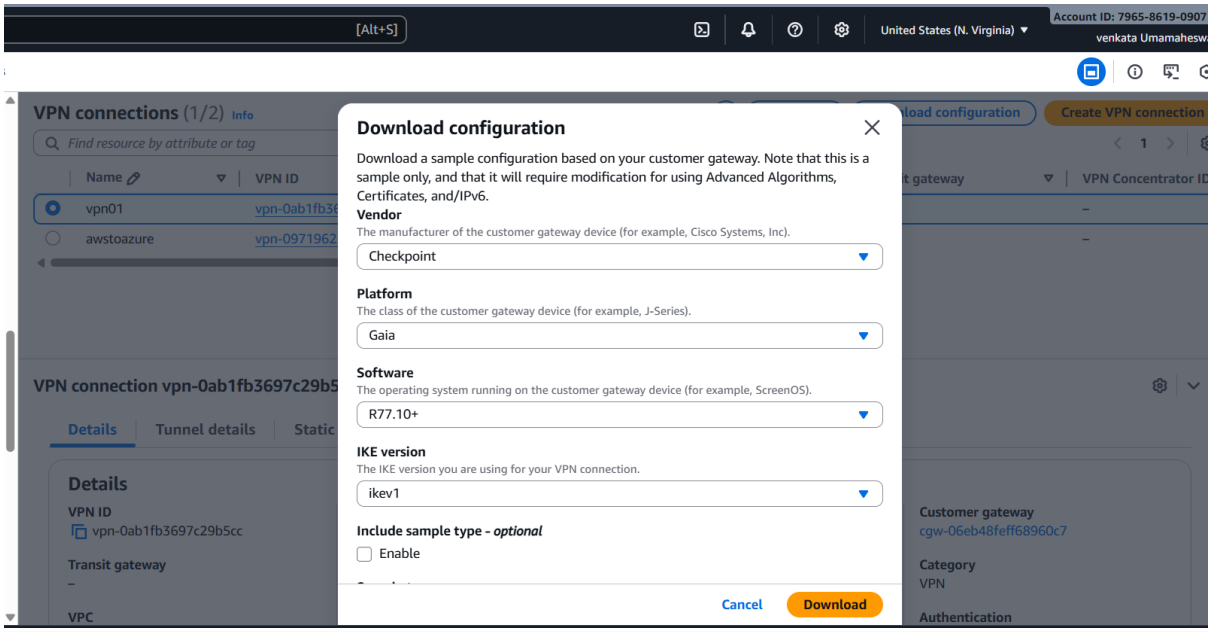
  ping <**Azure_VM_Private_IP**>

  If you get replies → VPN working**.**

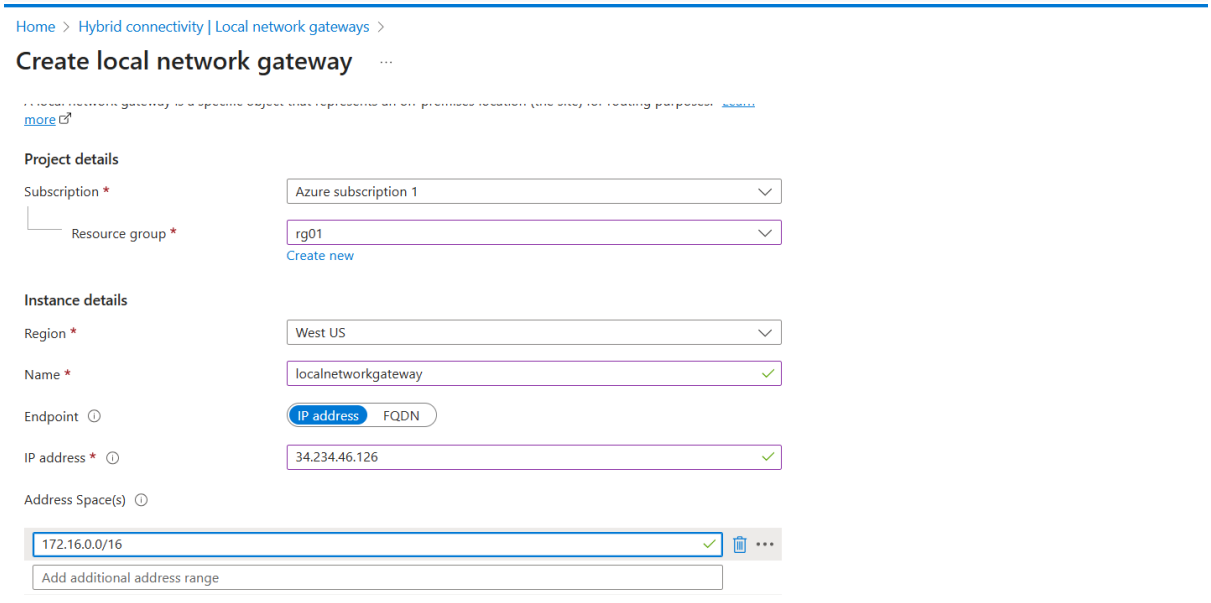
VIRTUAL PRIVATE NETWORK



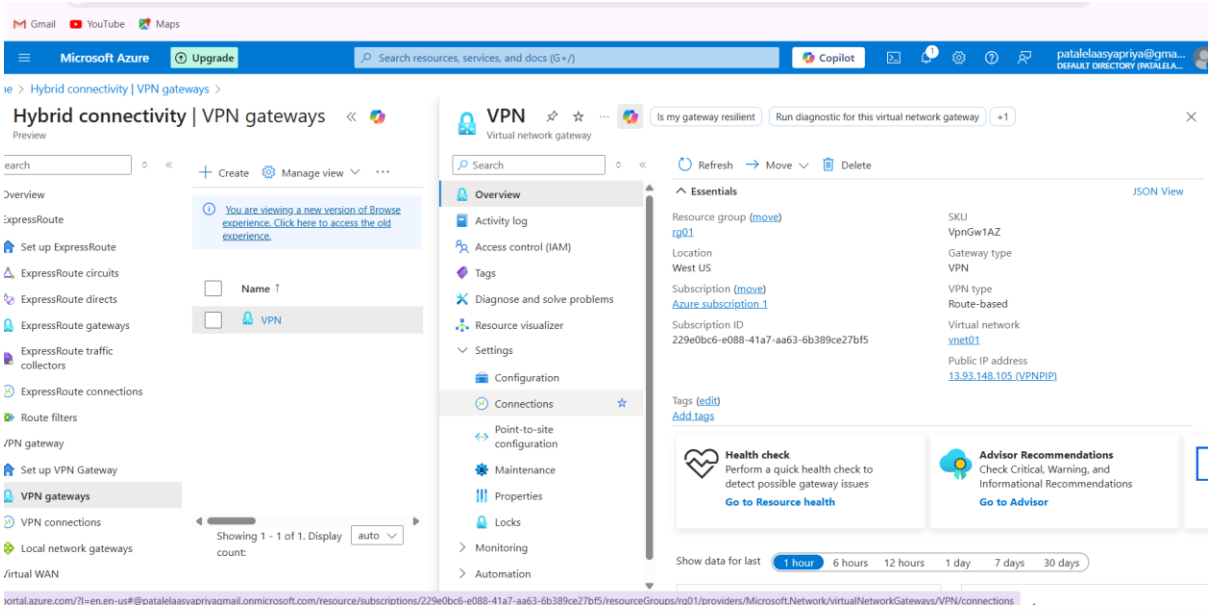
VIRTUAL PRIVATE NETWORK

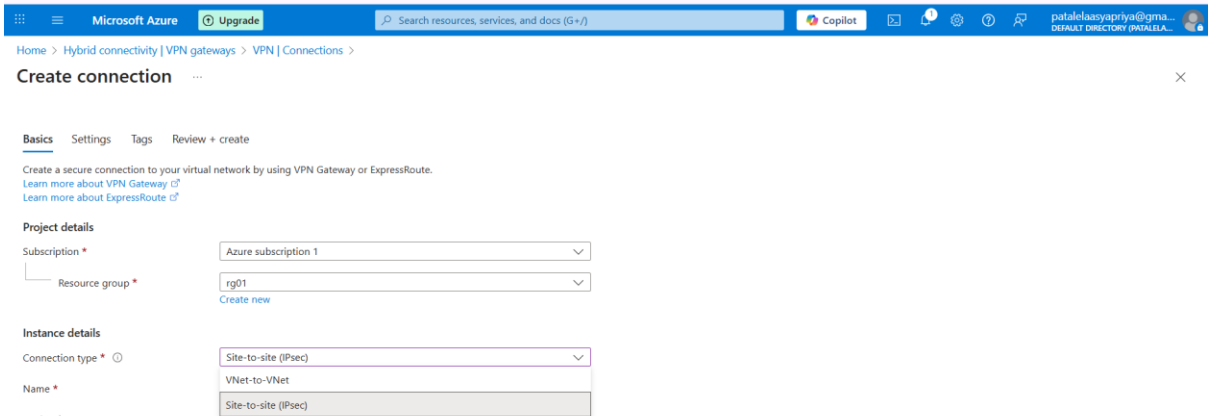


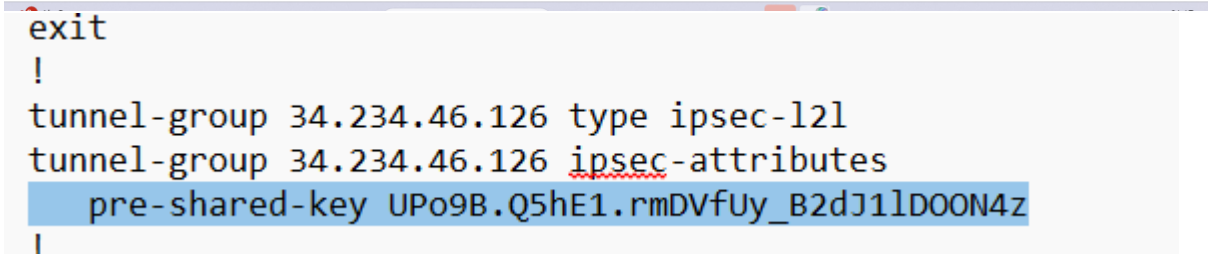

```
!
tunnel-group 34.234.46.126 type ipsec-l2l
tunnel-group 34.234.46.126 ipsec-attributes
   pre-shared-key UPo9B.Q5hE1.rmDVfUy_B2dJ1lDOON4z
!
! This option enables IPSec Dead Peer Detection, which causes semi-periodic
! messages to be sent to ensure a Security Association remains operational.
!
   isakmp keepalive threshold 10 retry 10
exit
```



VIRTUAL PRIVATE NETWORK

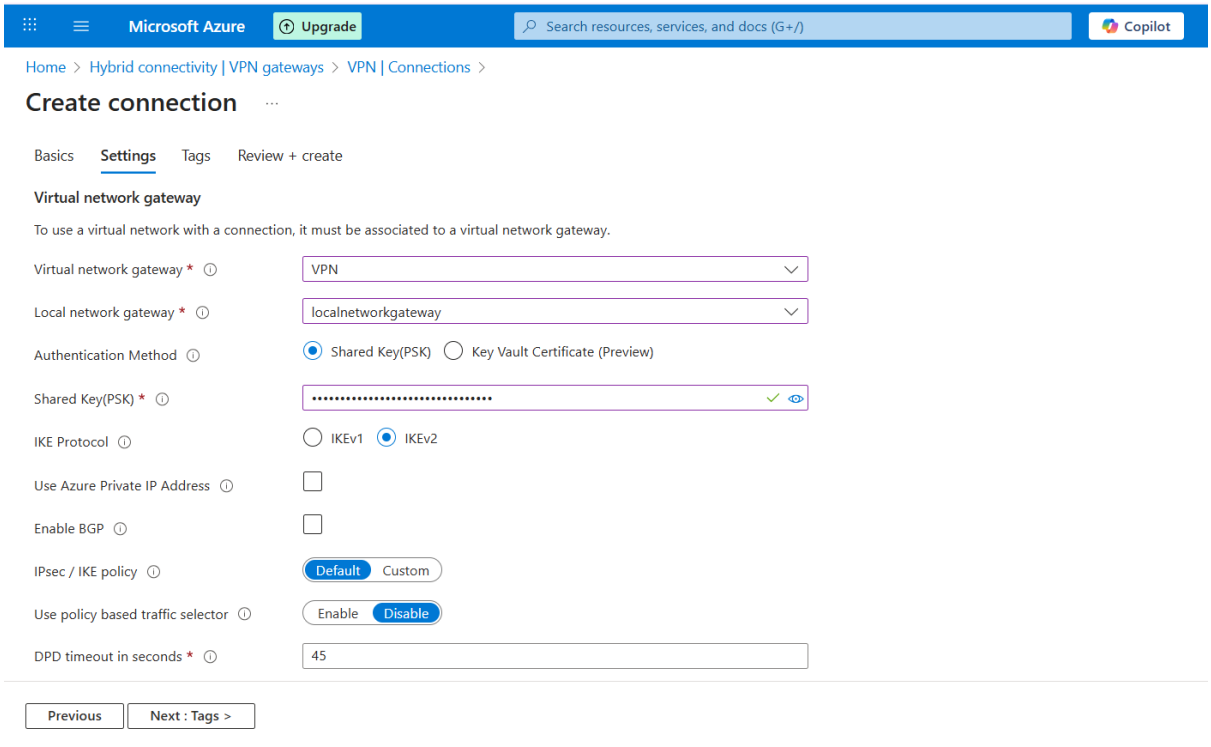


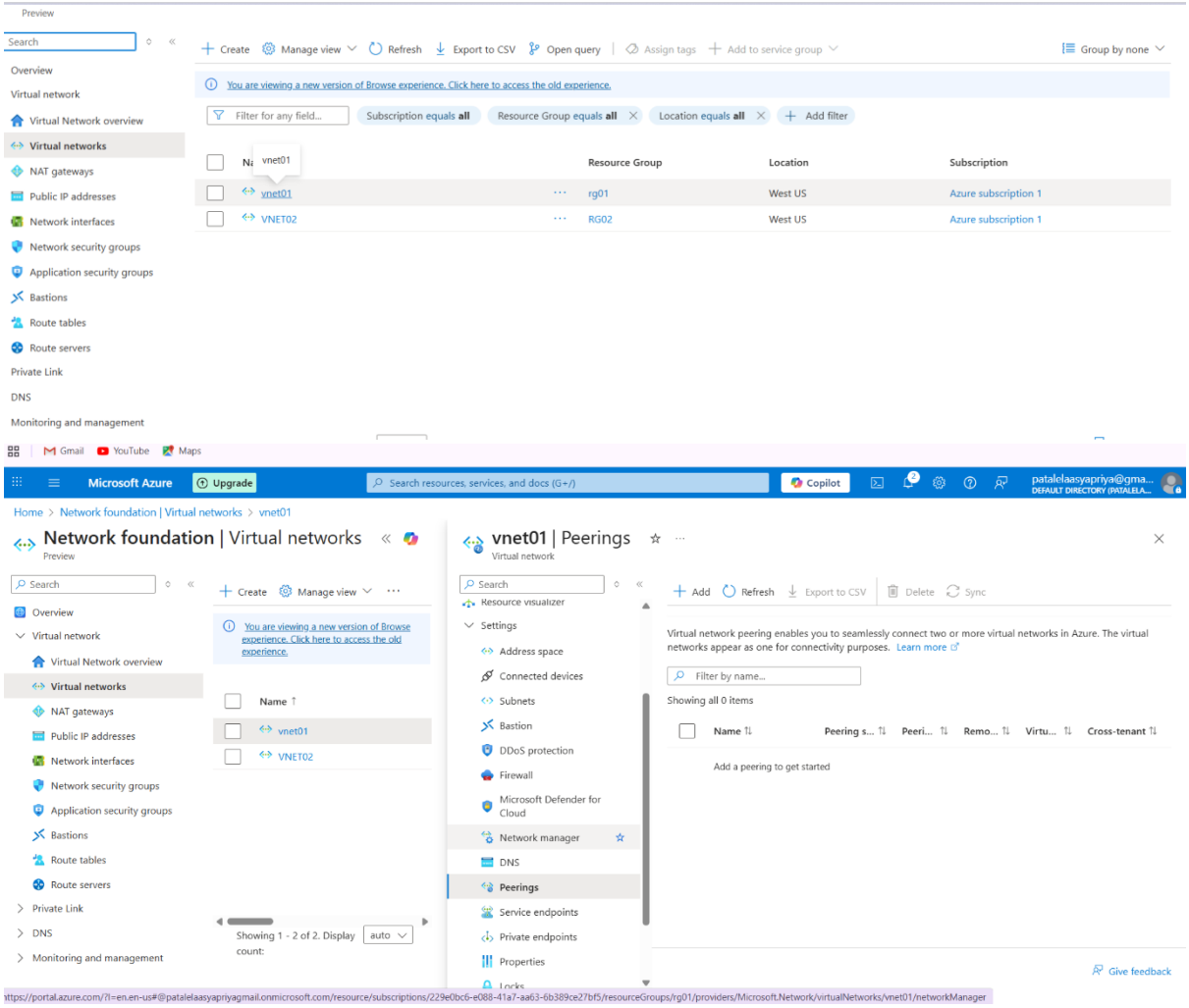




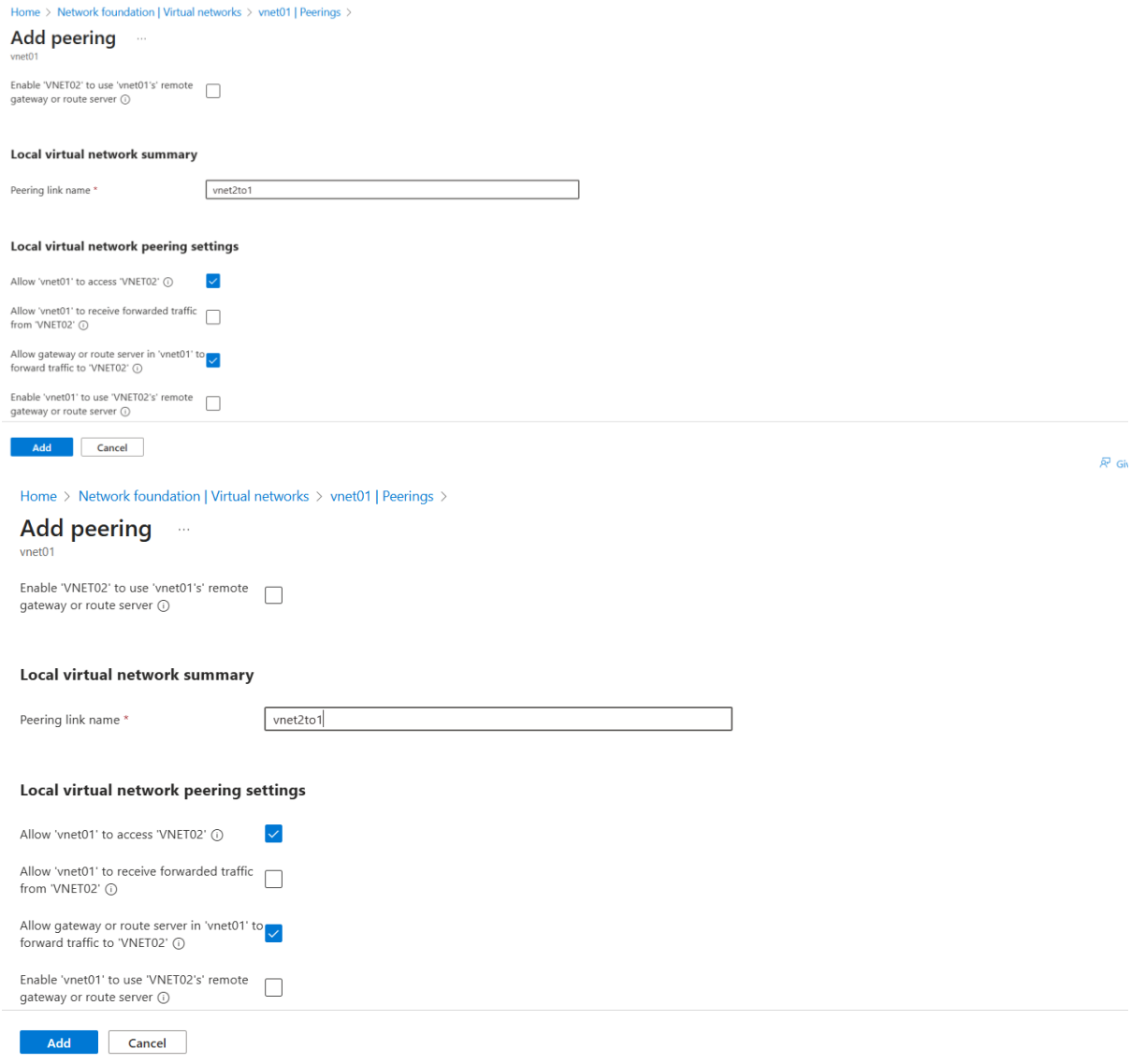
```
exit
!
tunnel-group 34.234.46.126 type ipsec-l2l
tunnel-group 34.234.46.126 ipsec-attributes
   pre-shared-key UPo9B.Q5hE1.rmDVfUy_B2dJ1lDOON4z
!
```

VIRTUAL PRIVATE NETWORK





VIRTUAL PRIVATE NETWORK



:nable 'VNET02' to use 'vnet01's' remote
jateway or route server ⓘ ☐

.ocal virtual network summary

'eering link name * 2to1

.ocal virtual network peering settings

\llow 'vnet01' to access 'VNET02' ⓘ ☑

\llow 'vnet01' to receive forwarded traffic
rom 'VNET02' ⓘ ☑

\llow gateway or route server in 'vnet01' to
orward traffic to 'VNET02' ⓘ ☑

:nable 'vnet01' to use 'VNET02's' remote
jateway or route server ⓘ ☐

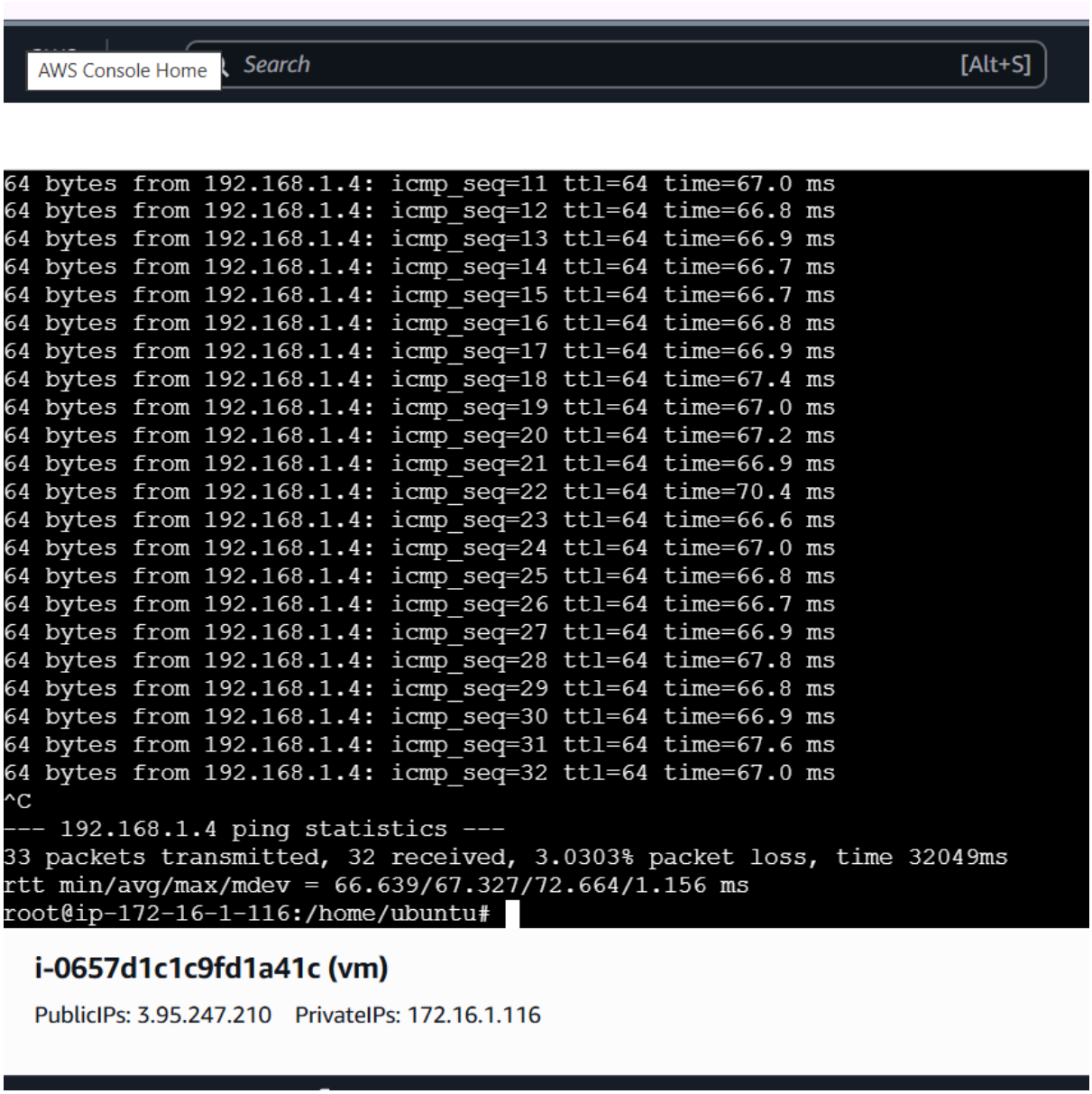
aws Search [Alt+S]

AWS Console Home

```
root@ip-172-16-1-116:/home/ubuntu# ping 192.168.1.4
PING 192.168.1.4 (192.168.1.4) 56(84) bytes of data.
64 bytes from 192.168.1.4: icmp_seq=1 ttl=64 time=67.8 ms
64 bytes from 192.168.1.4: icmp_seq=2 ttl=64 time=67.1 ms
64 bytes from 192.168.1.4: icmp_seq=3 ttl=64 time=67.1 ms
64 bytes from 192.168.1.4: icmp_seq=4 ttl=64 time=67.4 ms
64 bytes from 192.168.1.4: icmp_seq=5 ttl=64 time=67.0 ms
64 bytes from 192.168.1.4: icmp_seq=6 ttl=64 time=66.8 ms
64 bytes from 192.168.1.4: icmp_seq=7 ttl=64 time=67.0 ms
64 bytes from 192.168.1.4: icmp_seq=8 ttl=64 time=67.4 ms
64 bytes from 192.168.1.4: icmp_seq=9 ttl=64 time=72.7 ms
64 bytes from 192.168.1.4: icmp_seq=10 ttl=64 time=67.3 ms
64 bytes from 192.168.1.4: icmp_seq=11 ttl=64 time=67.0 ms
64 bytes from 192.168.1.4: icmp_seq=12 ttl=64 time=66.8 ms
64 bytes from 192.168.1.4: icmp_seq=13 ttl=64 time=66.9 ms
64 bytes from 192.168.1.4: icmp_seq=14 ttl=64 time=66.7 ms
64 bytes from 192.168.1.4: icmp_seq=15 ttl=64 time=66.7 ms
64 bytes from 192.168.1.4: icmp_seq=16 ttl=64 time=66.8 ms
64 bytes from 192.168.1.4: icmp_seq=17 ttl=64 time=66.9 ms
64 bytes from 192.168.1.4: icmp_seq=18 ttl=64 time=67.4 ms
64 bytes from 192.168.1.4: icmp_seq=19 ttl=64 time=67.0 ms
64 bytes from 192.168.1.4: icmp_seq=20 ttl=64 time=67.2 ms
64 bytes from 192.168.1.4: icmp_seq=21 ttl=64 time=66.9 ms
64 bytes from 192.168.1.4: icmp_seq=22 ttl=64 time=70.4 ms
64 bytes from 192.168.1.4: icmp_seq=23 ttl=64 time=66.6 ms
64 bytes from 192.168.1.4: icmp_seq=24 ttl=64 time=67.0 ms
```

i-0657d1c1c9fd1a41c (vm)

PublicIPs: 3.95.247.210 PrivateIPs: 172.16.1.116


AWS Console Home
Search
[Alt+S]
64 bytes from 192.168.1.4: icmp_seq=11 ttl=64 time=67.0 ms
64 bytes from 192.168.1.4: icmp_seq=12 ttl=64 time=66.8 ms
64 bytes from 192.168.1.4: icmp_seq=13 ttl=64 time=66.9 ms
64 bytes from 192.168.1.4: icmp_seq=14 ttl=64 time=66.7 ms
64 bytes from 192.168.1.4: icmp_seq=15 ttl=64 time=66.7 ms
64 bytes from 192.168.1.4: icmp_seq=16 ttl=64 time=66.8 ms
64 bytes from 192.168.1.4: icmp_seq=17 ttl=64 time=66.9 ms
64 bytes from 192.168.1.4: icmp_seq=18 ttl=64 time=67.4 ms
64 bytes from 192.168.1.4: icmp_seq=19 ttl=64 time=67.0 ms
64 bytes from 192.168.1.4: icmp_seq=20 ttl=64 time=67.2 ms
64 bytes from 192.168.1.4: icmp_seq=21 ttl=64 time=66.9 ms
64 bytes from 192.168.1.4: icmp_seq=22 ttl=64 time=70.4 ms
64 bytes from 192.168.1.4: icmp_seq=23 ttl=64 time=66.6 ms
64 bytes from 192.168.1.4: icmp_seq=24 ttl=64 time=67.0 ms
64 bytes from 192.168.1.4: icmp_seq=25 ttl=64 time=66.8 ms
64 bytes from 192.168.1.4: icmp_seq=26 ttl=64 time=66.7 ms
64 bytes from 192.168.1.4: icmp_seq=27 ttl=64 time=66.9 ms
64 bytes from 192.168.1.4: icmp_seq=28 ttl=64 time=67.8 ms
64 bytes from 192.168.1.4: icmp_seq=29 ttl=64 time=66.8 ms
64 bytes from 192.168.1.4: icmp_seq=30 ttl=64 time=66.9 ms
64 bytes from 192.168.1.4: icmp_seq=31 ttl=64 time=67.6 ms
64 bytes from 192.168.1.4: icmp_seq=32 ttl=64 time=67.0 ms
^C
--- 192.168.1.4 ping statistics ---
33 packets transmitted, 32 received, 3.0303% packet loss, time 32049ms
rtt min/avg/max/mdev = 66.639/67.327/72.664/1.156 ms
root@ip-172-16-1-116:/home/ubuntu#
i-0657d1c1c9fd1a41c (vm)
PublicIPs: 3.95.247.210 PrivateIPs: 172.16.1.116