**Use Case: Setting up Transit Gateway and VPC Endpoints for a Multi-VPC Architecture**

**Scenario:**

A large organization is migrating its on-premises infrastructure to the AWS cloud.

The organization's architecture involves multiple VPCs for different departments and applications, each requiring secure communication with centralized services and external resources.

The IT team needs to design and implement a scalable and efficient network architecture to accommodate the organization's growth and ensure robust connectivity between VPCs and external services.

**Objectives:**

* Design and deploy a scalable network architecture using AWS Transit Gateway to simplify network connectivity between multiple VPCs.
* Configure VPC endpoints to securely access AWS services without internet gateways or NAT gateways, ensuring data privacy and minimizing exposure to external threats.

**Note:**

* Create 4 VPCs in 4 different regions and set up Transit Gateway.
* Do not use default VPCs.
* Prepare a document with all the screenshots and push it to a GitHub repo, then share the URL.

## **🔹 Step 1: Create 4 VPCs (No Default VPCs)**

We need 4 VPCs in different regions:

* **us-east-1 (Hub)**
* **us-west-2 (Spoke 1)**
* **eu-west-1 (Spoke 2)**
* **ap-south-1 (Spoke 3)**

👉 In each region, do the following:

1. Go to **VPC Console** → **Your VPCs** → **Create VPC**.  
   * Name: vpc-hub-us-east-1 (adjust per region).
   * IPv4 CIDR: 10.0.0.0/16 (use different CIDRs for each region). Example:  
     + Hub (us-east-1): 10.0.0.0/16
     + Spoke1 (us-west-2): 10.1.0.0/16
     + Spoke2 (eu-west-1): 10.2.0.0/16
     + Spoke3 (ap-south-1): 10.3.0.0/16
   * Tenancy: Default.
2. Subnets: Create **2 Private Subnets** in each VPC.  
   * Example (Hub):  
     + 10.0.1.0/24 (AZ1)
     + 10.0.2.0/24 (AZ2)
   * Repeat for other VPCs.
3. **No Internet Gateway** (since we want private).

## **🔹 Step 2: Create Transit Gateways**

1. In **each region**, go to **Transit Gateway Console** → **Create Transit Gateway**.  
   * Name: tgw-us-east-1 (Hub), tgw-us-west-2, etc.
   * Amazon side ASN: leave default (64512).
   * Multicast support: disable.
   * DNS support: enable.
2. After creation, note down **TGW IDs** for all 4.

## **🔹 Step 3: Attach VPCs to TGWs**

For each region:

1. Go to **Transit Gateway Attachments** → **Create Attachment**.  
   * Select your **TGW**.
   * Attachment type: **VPC**.
   * Select your VPC & private subnets.
2. Do this for all 4 VPCs.

## **🔹 Step 4: Create TGW Peerings (Hub-and-Spoke)**

1. Go to **us-east-1 (Hub)** → **Transit Gateway Peering Attachments** → **Create**.  
   * Request Peering to tgw-us-west-2.
   * Accept the request in us-west-2.
2. Repeat for eu-west-1 and ap-south-1.

✅ Now TGWs are peered with Hub.

## **🔹 Step 5: Update Route Tables**

### **VPC Route Tables:**

* In each VPC’s **Route Table**, add a route for other VPC CIDRs → point to TGW.

Example (Spoke1):

* Destination: 10.0.0.0/16 (Hub) → Target: TGW.
* Destination: 10.2.0.0/16 (Spoke2) → Target: TGW.
* Destination: 10.3.0.0/16 (Spoke3) → Target: TGW.

Do similarly for others.

### **TGW Route Tables:**

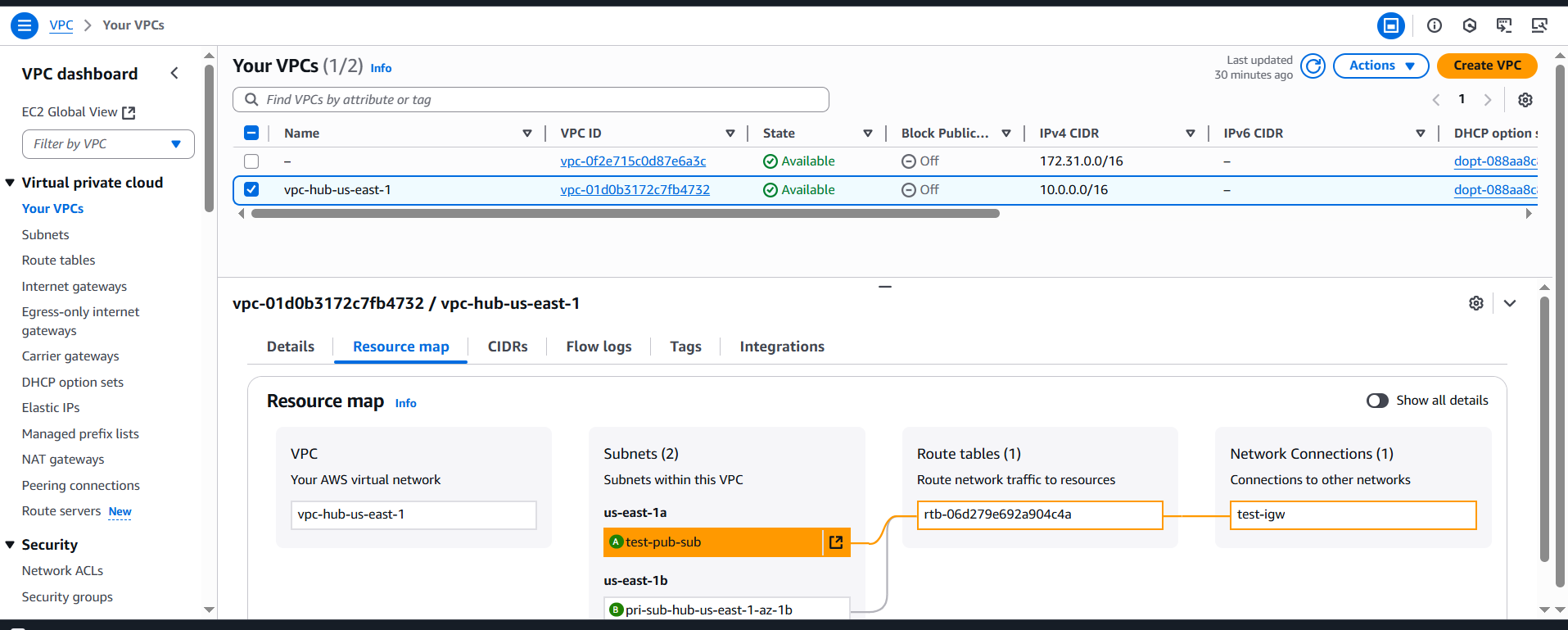
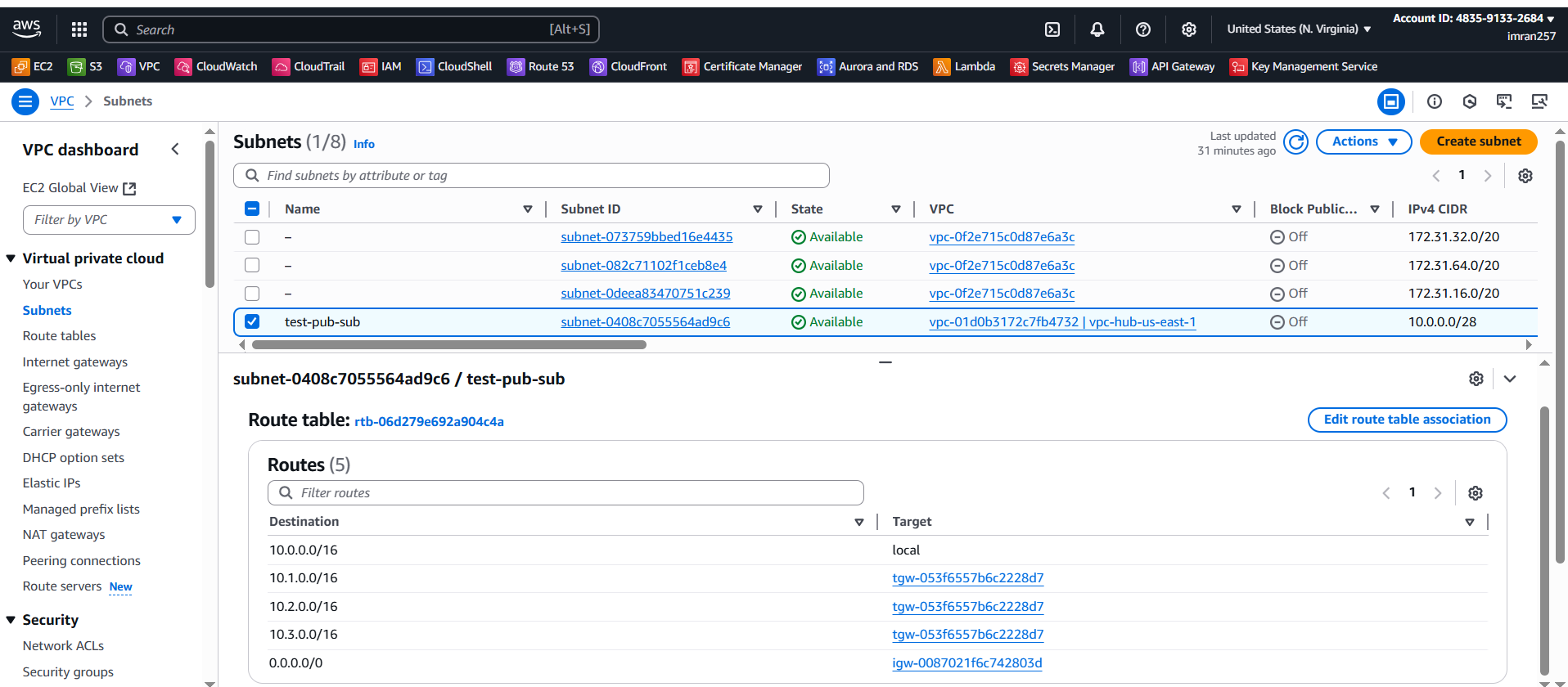
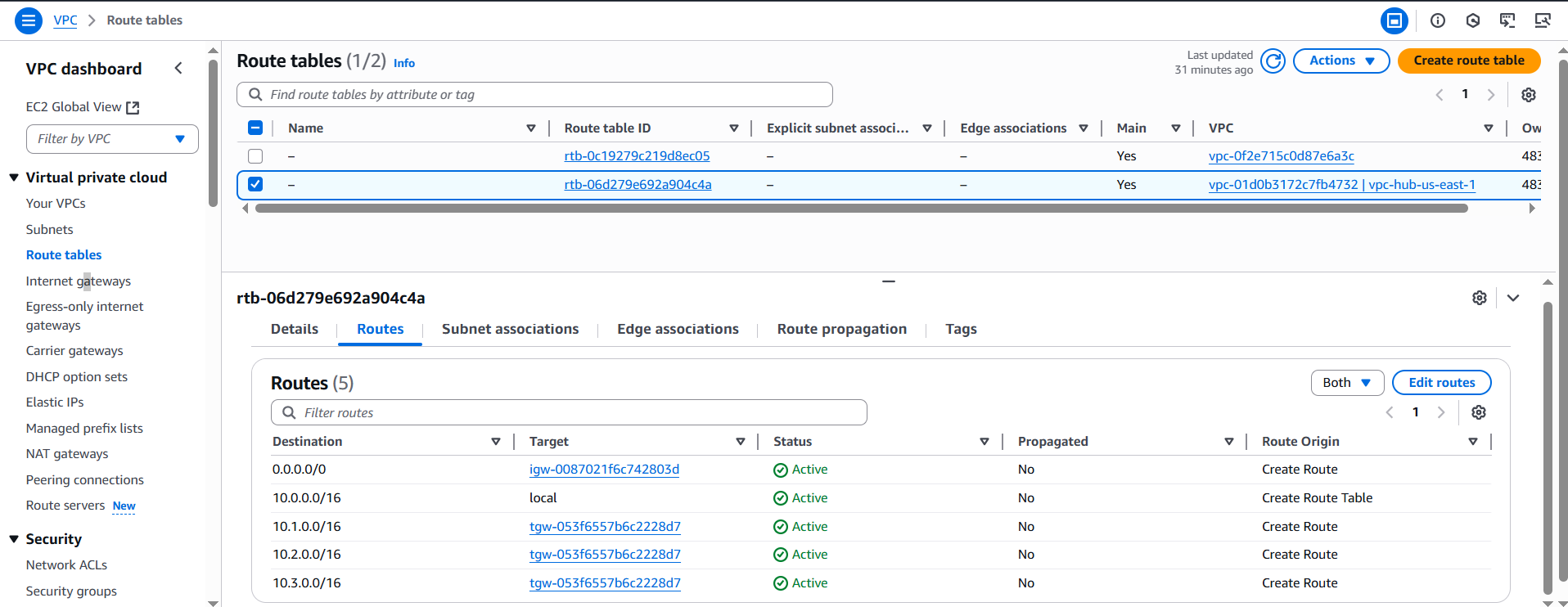
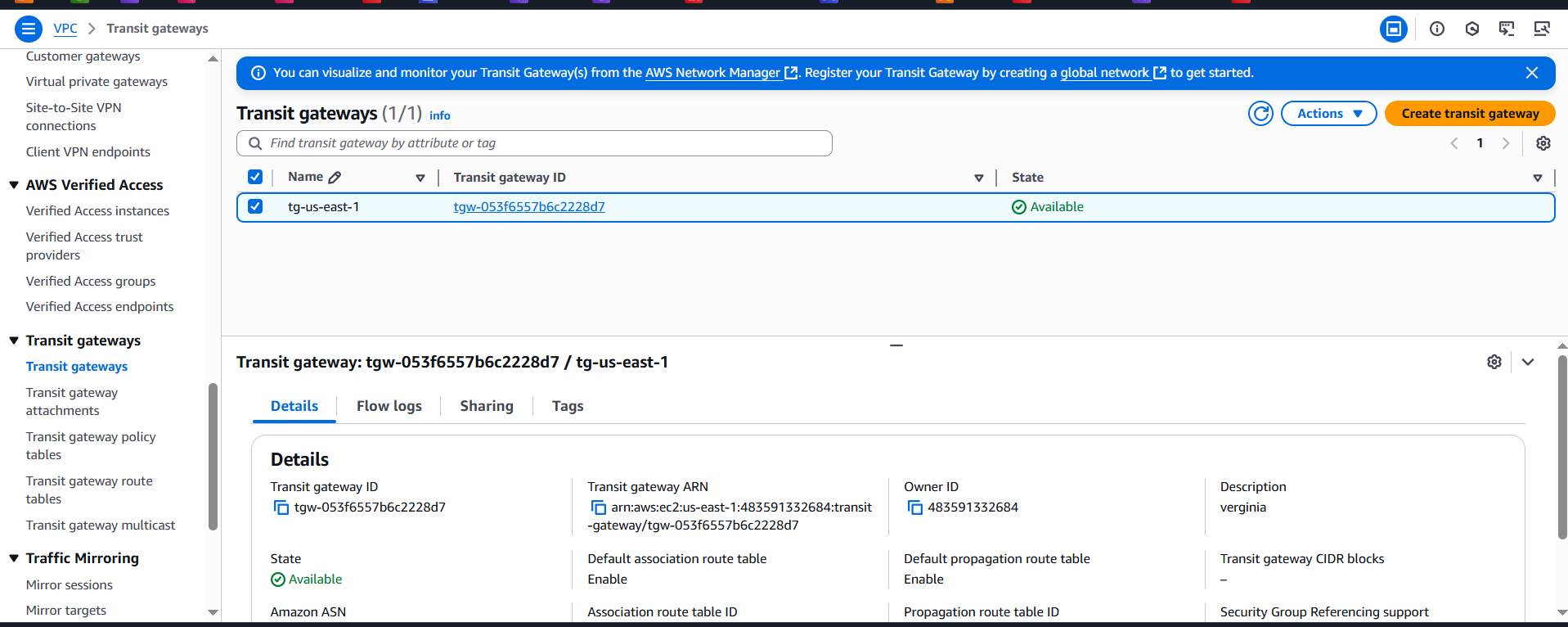
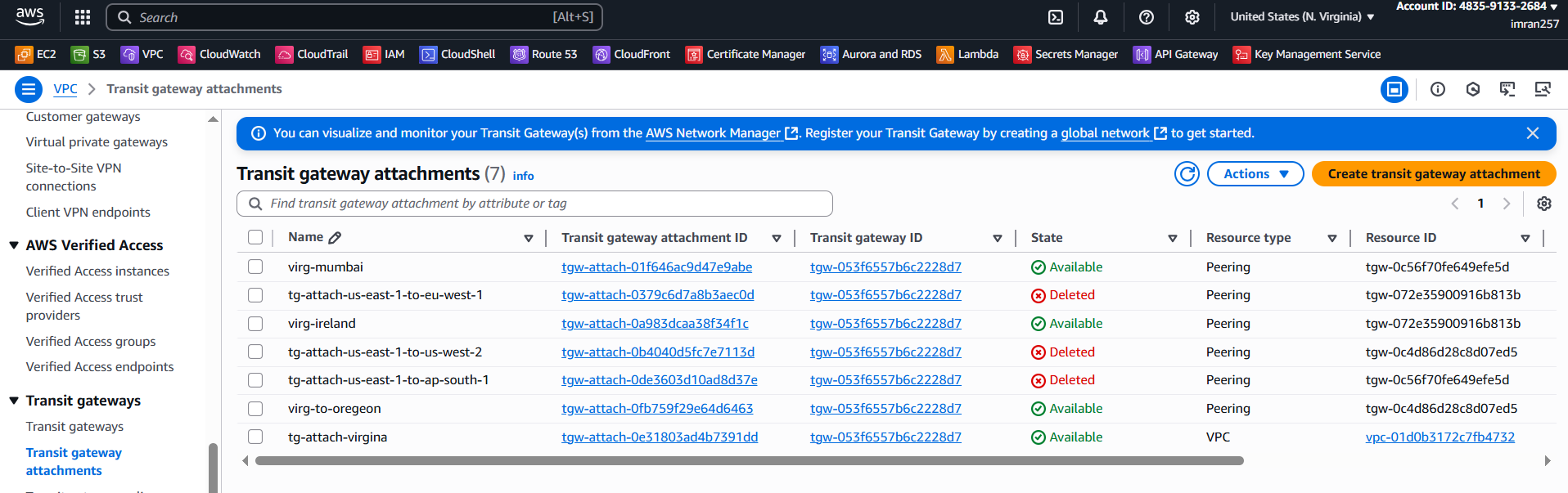
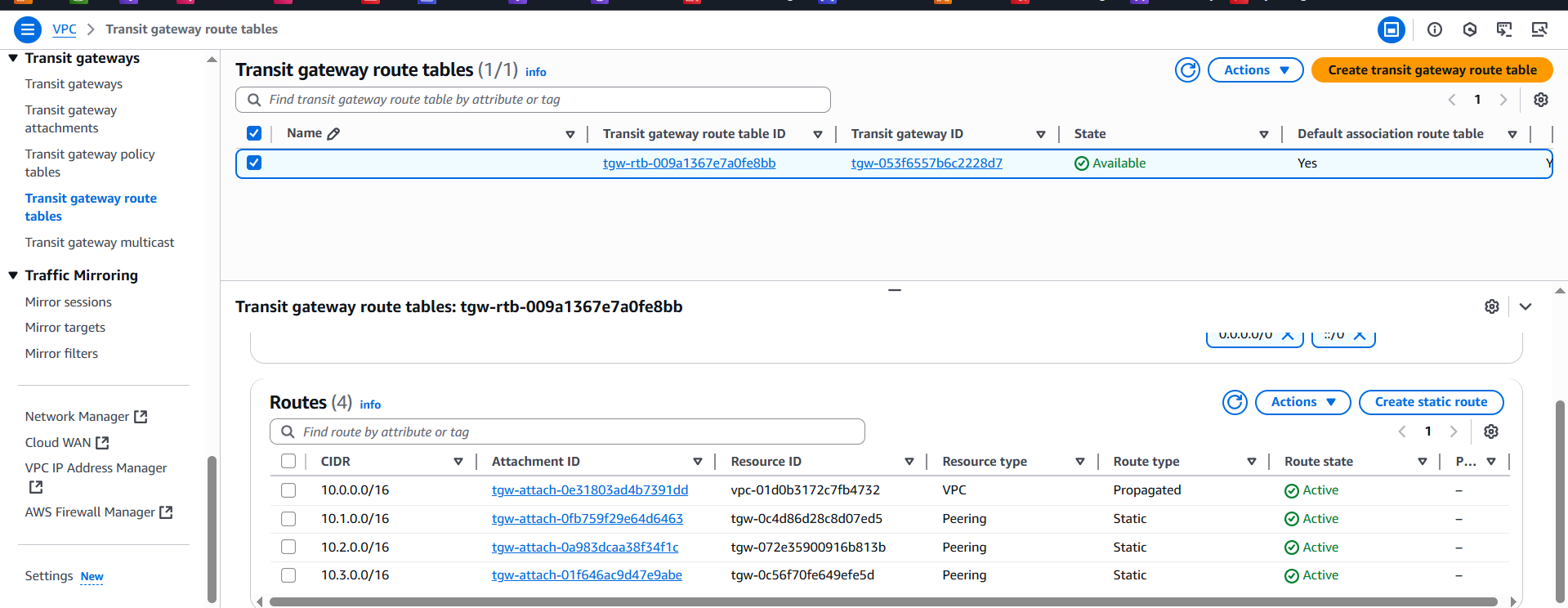
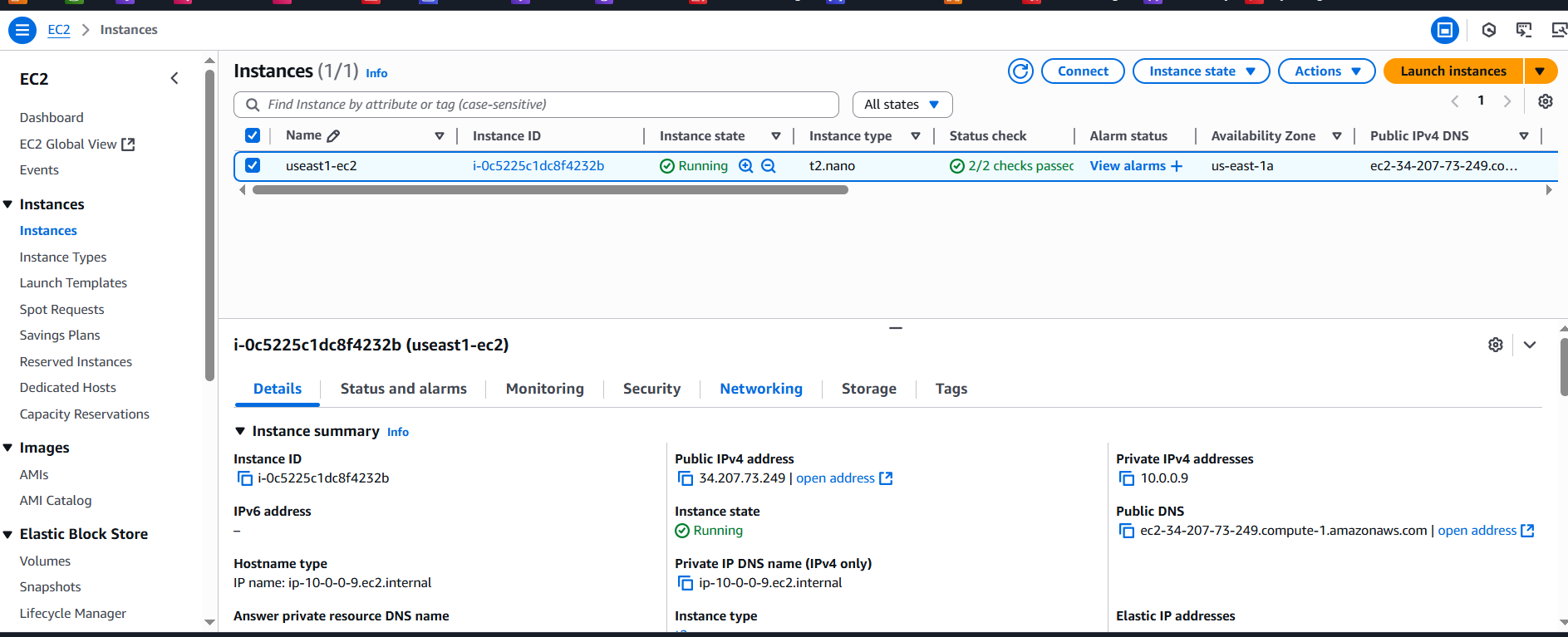
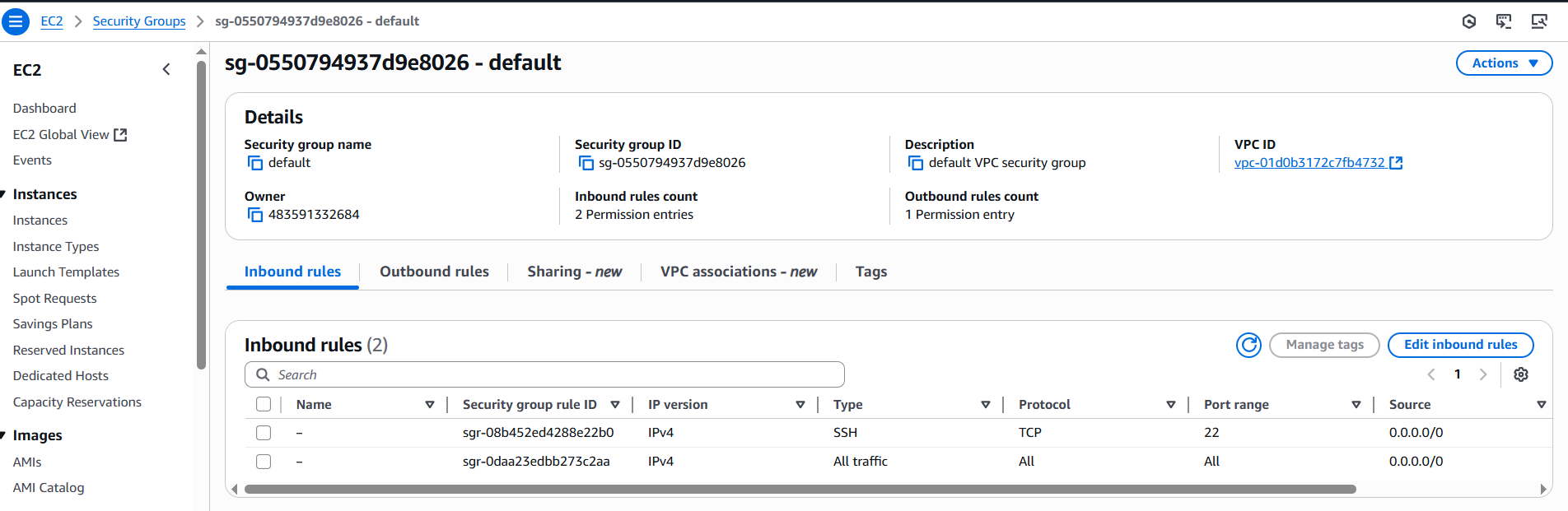
* Hub TGW: Add routes for **all spoke CIDRs** pointing to their attachments.
* Spokes TGW: Add route to **Hub CIDR** pointing to peering.

## **🔹 Step 6: Create VPC Endpoints**

In each VPC, go to **VPC Console → Endpoints → Create Endpoint**:

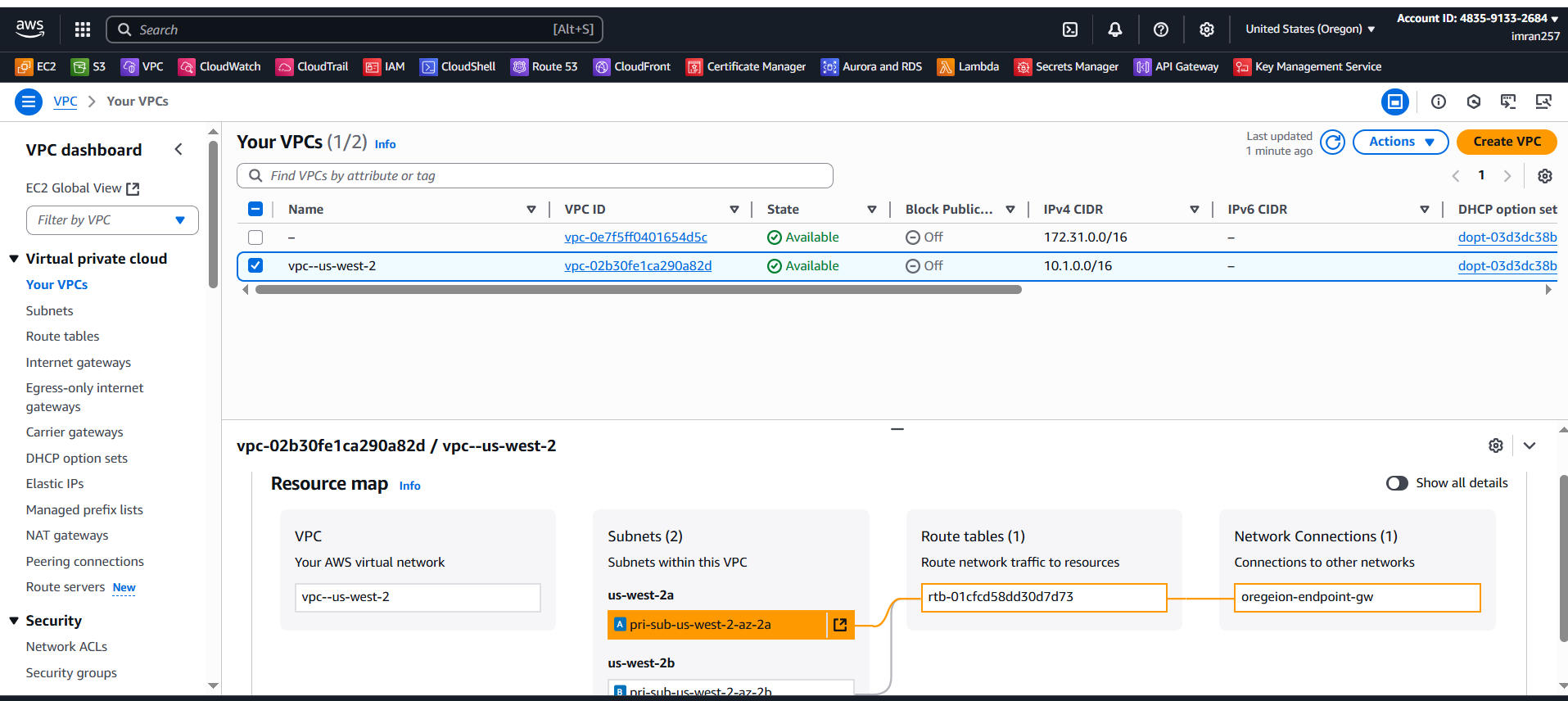
1. **Gateway Endpoint**:  
   * Service: **S3**.
   * Route table: Select your VPC route table.
2. **Interface Endpoints** (one by one):  
   * **SSM**
   * **SSMMessages**
   * **EC2**
   * **ECR.api**
   * **ECR.dkr**
   * **CloudWatch Logs**
   * **Ensure private DNS is enabled.**

## **🔹 Step 7: Validation**

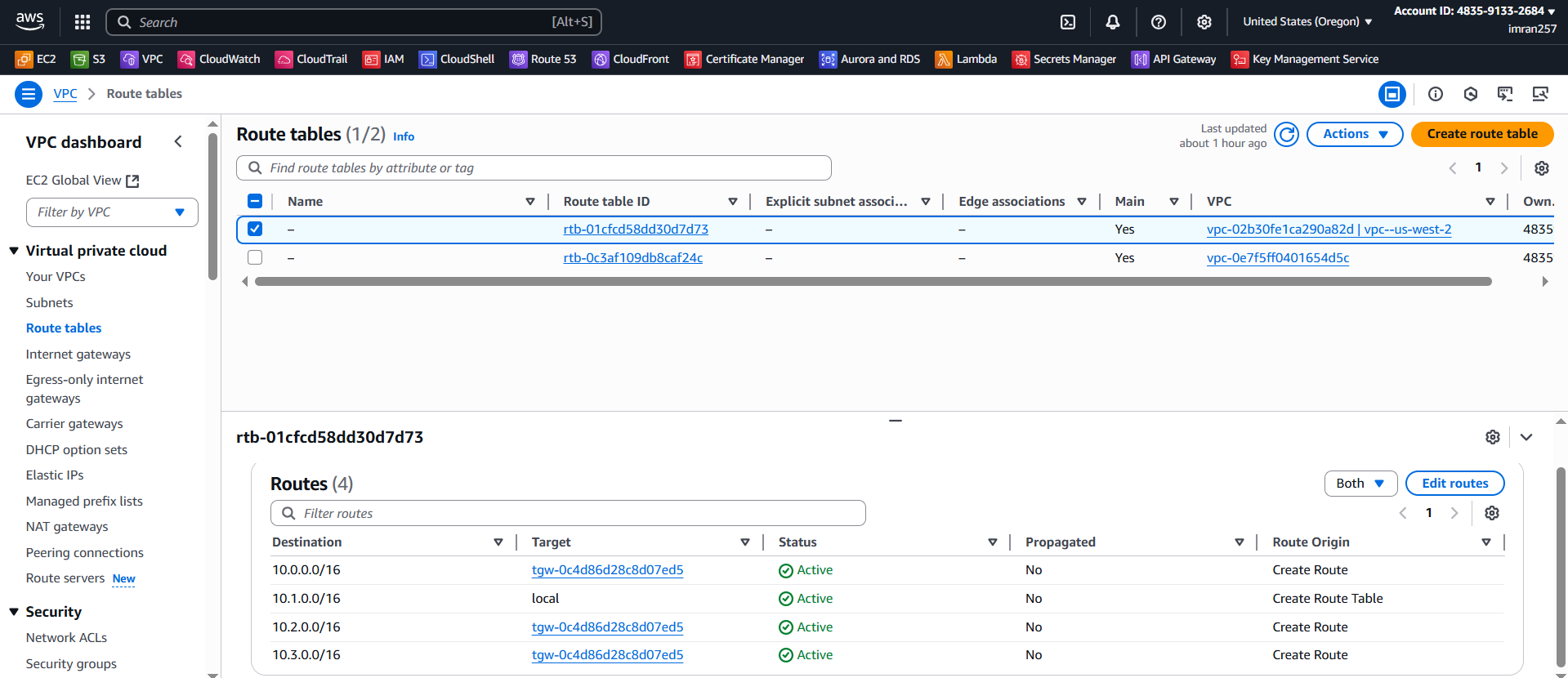
1. Launch **EC2 Instances** in private subnets (use SSM agent enabled AMI, like Amazon Linux 2/2023).
2. Connect via **SSM Session Manager** (no need for SSH key).
3. Validate connectivity:  
   * Ping from Hub → Spoke VPCs.
   * Run aws s3 ls (should work via VPC endpoint).
   * Check ECR & SSM access.
4. Created 4 VPC’s in Virginia, Oregon, Ireland , Mumbai region having vpc range
5. 10.0.0.0 , 10.1.0.0, 10.2.0.0 10.3.0.0
6. Showing vpc of Virginia. Connect igw to treat as primary source to connect other regions ec2.
7. 
8. Create subnets make routes to the all region destinations.
9. 
10. Create route tables and make destinations
11. 
12. Create transit gateway in all regions
13. 
14. Create attachments 1 is VPC connection for self and 3 Peering connections to other regions.need to send request and other regions need to accept.
15. 
16. Create transit gateway route tables with other regions use their CIDR.
17. 
18. Create EC2 and edit inbound rules to allow traffic rules in security group.
19. 
20. 

Oregon

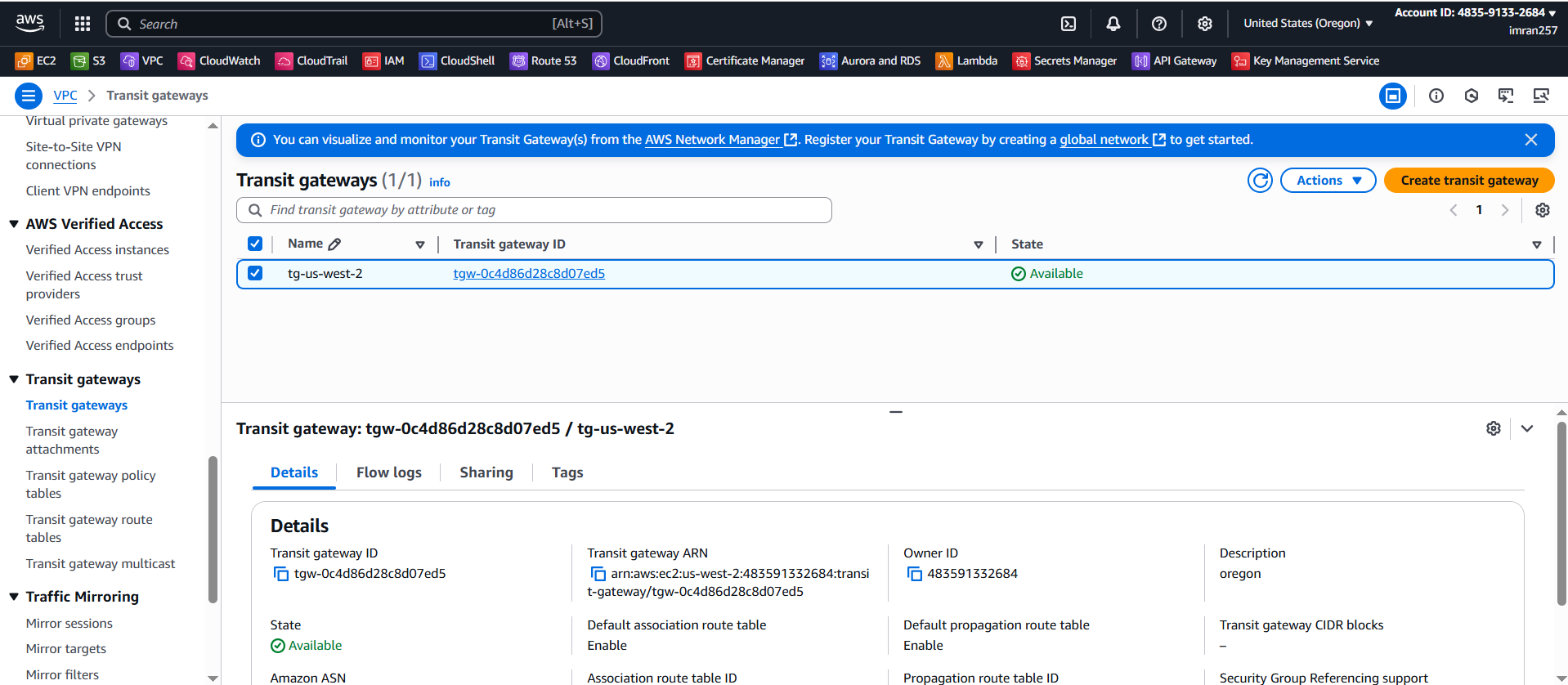
1. Created vpc in oregon region. And created end point only here to access the S3 Resource or S3 bucket of another region.

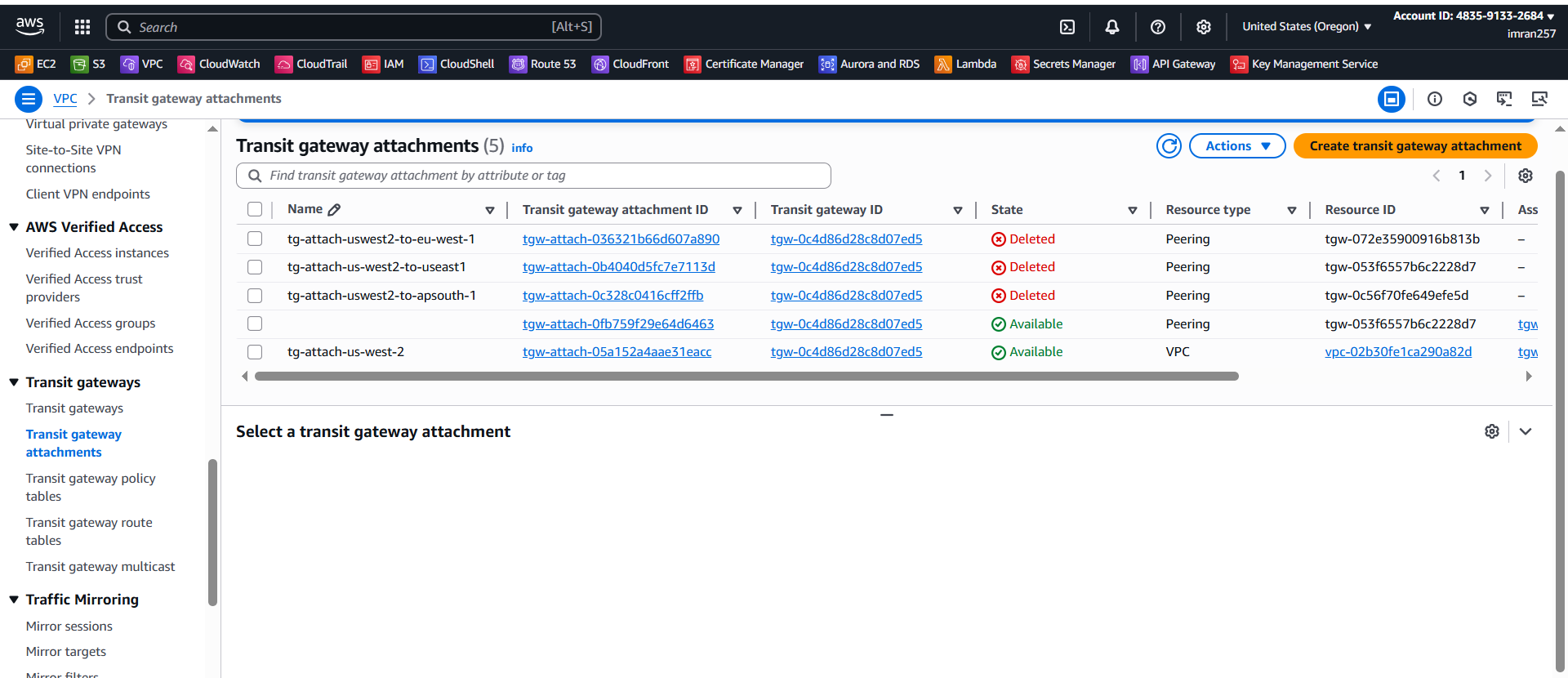


1. Created route table and attached routes of other regions.

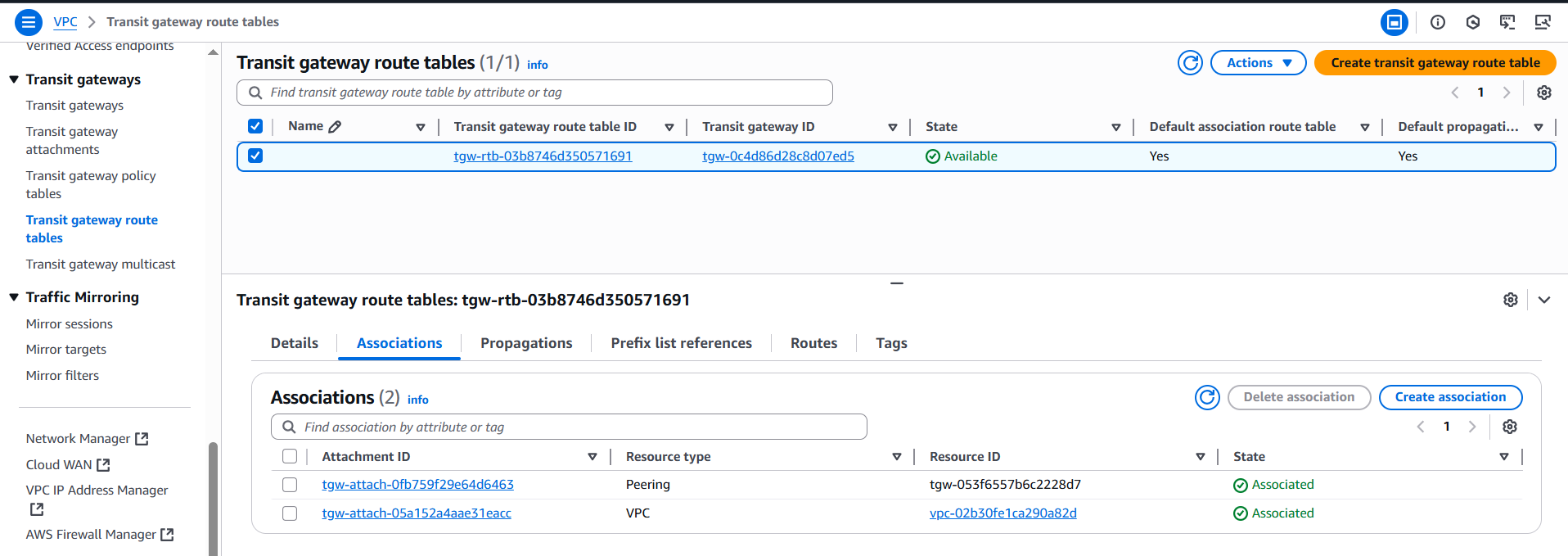


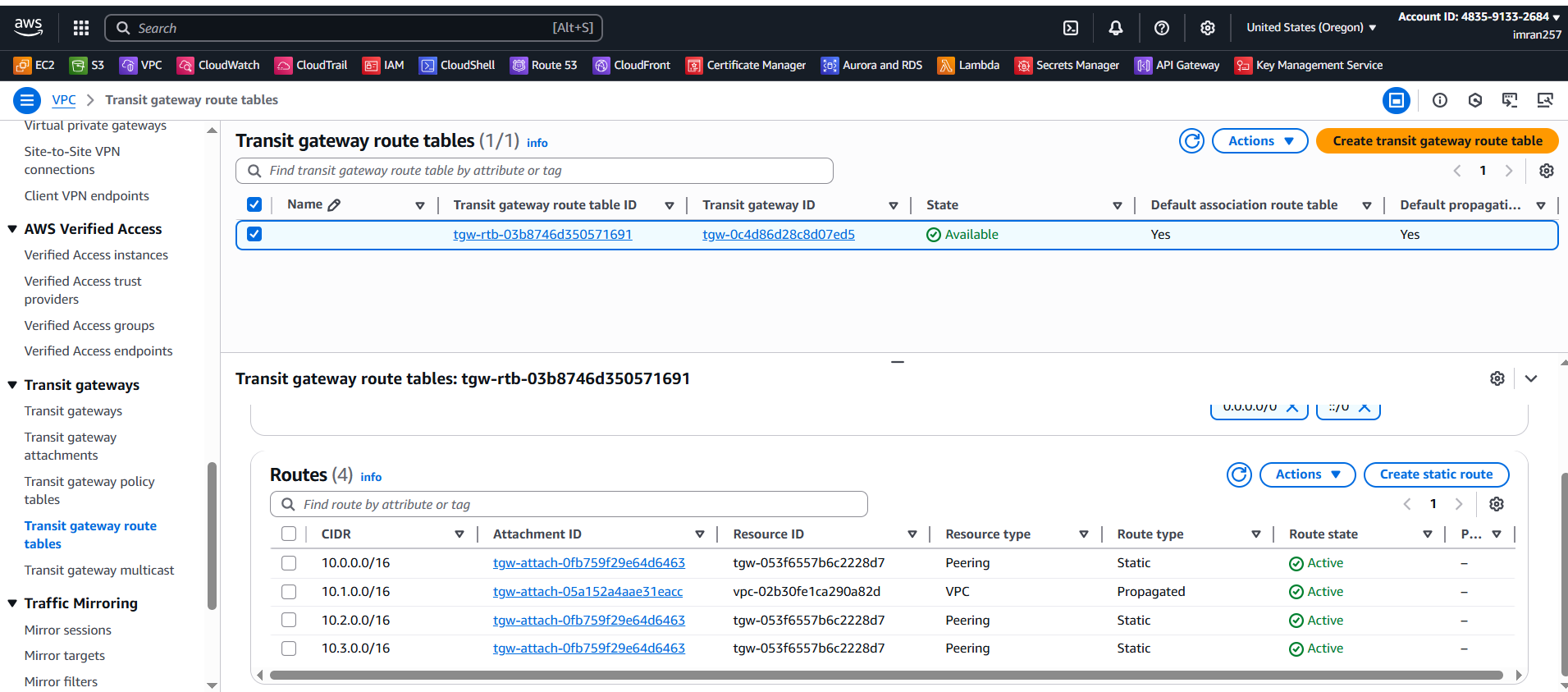
1. Created transit gateway and attachments with resource type 1 VPC for self and 1 peering with virginia.





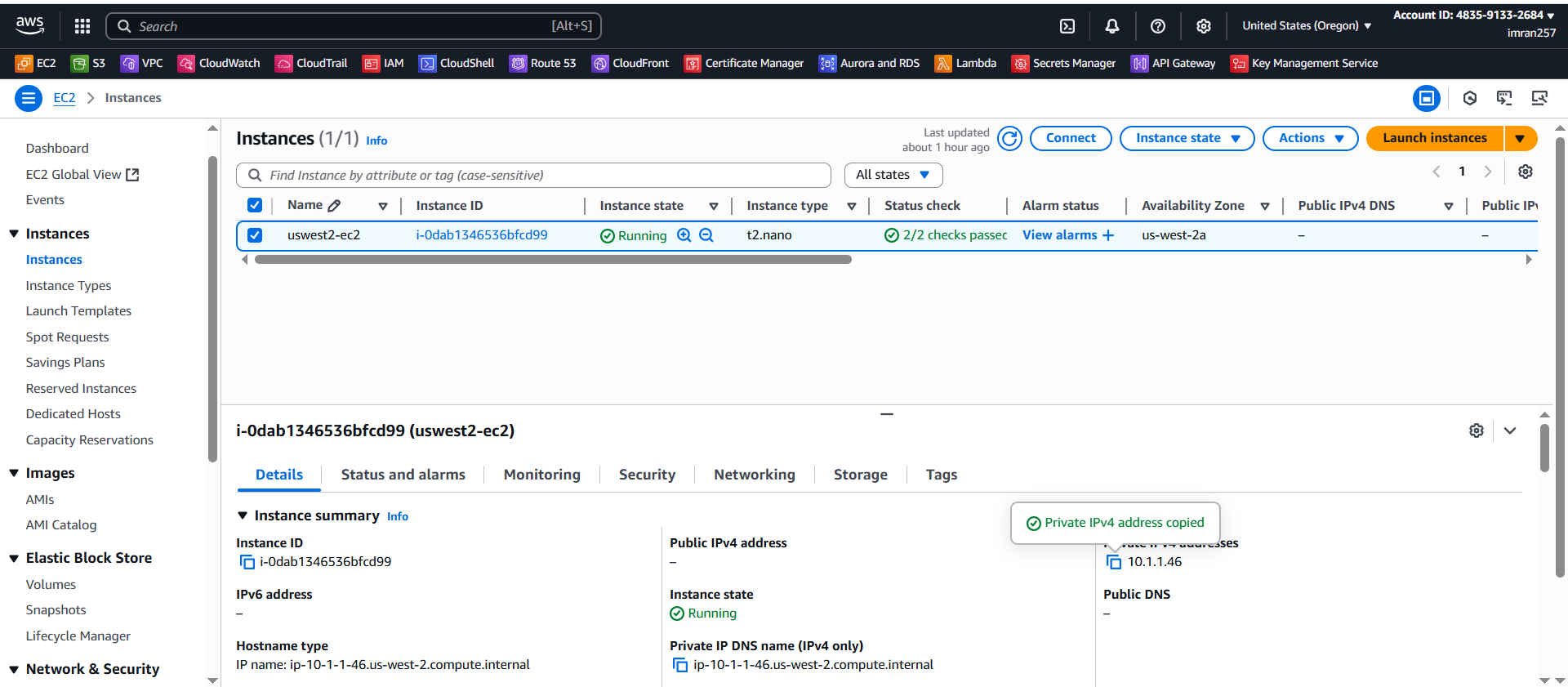
1. Adding associations with self VPC and peering to virginia. And i route tables add routes to all regions.

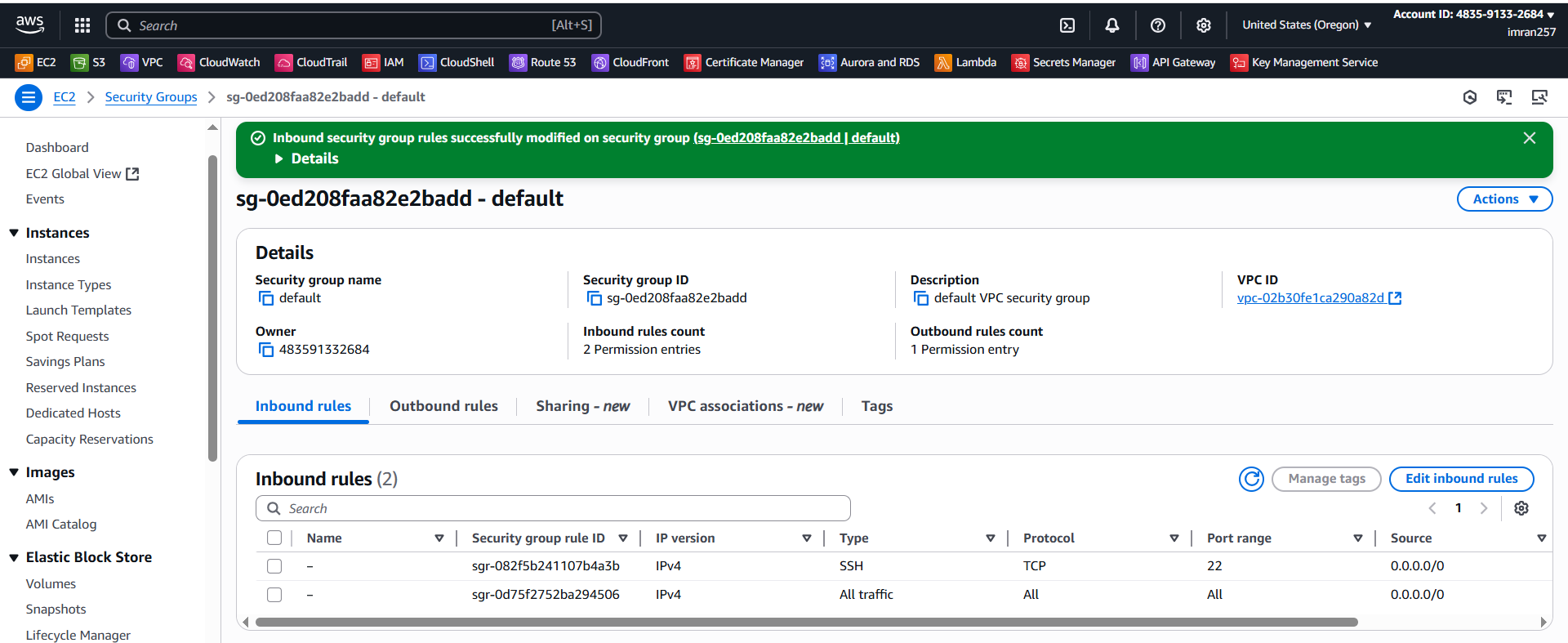




In same way the above things need to do in other regions also,

1. Now created EC2 with having igw as a primary source. And in inbound rules of security group allow ssh and all traffic.

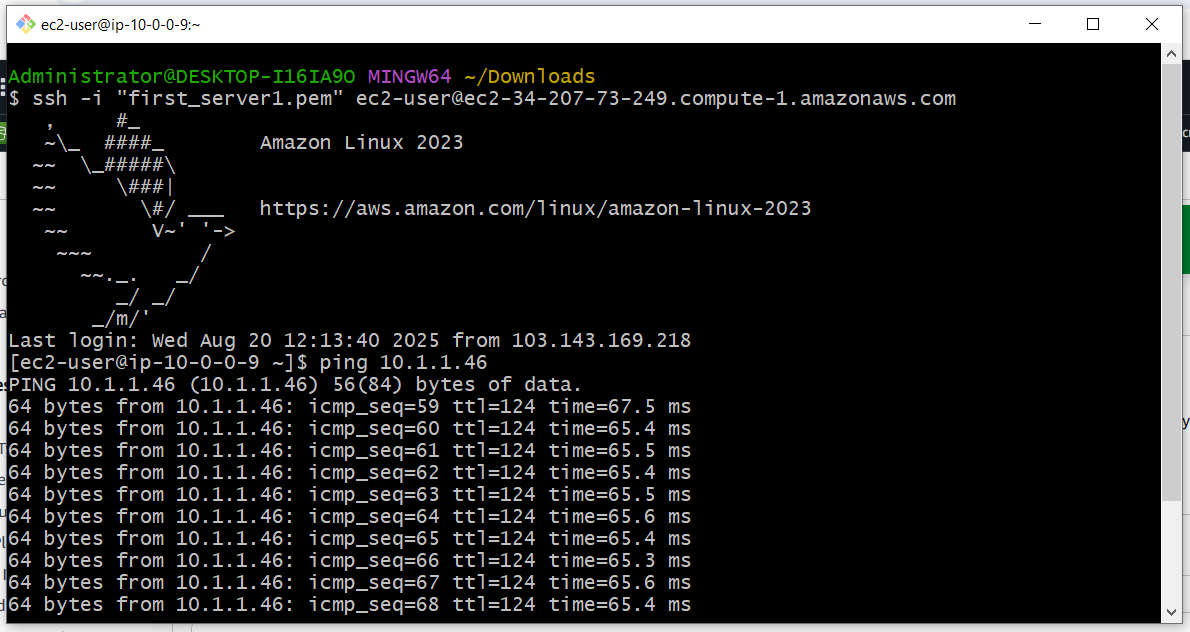




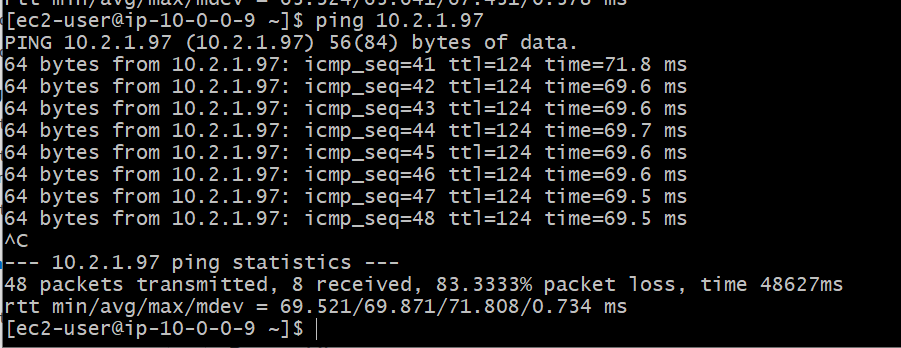
1. Created instacnces in other regions as private having no internet.

TEST:

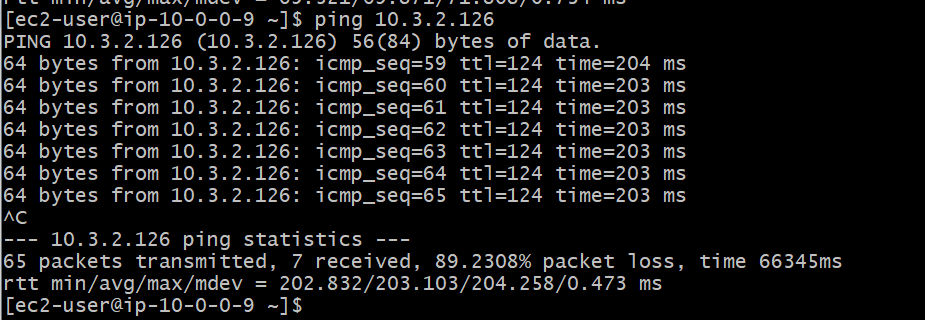
1. Now tried to ping from VIRGINIA to OREGEON



1. Tried to ping from Virginia to oregon

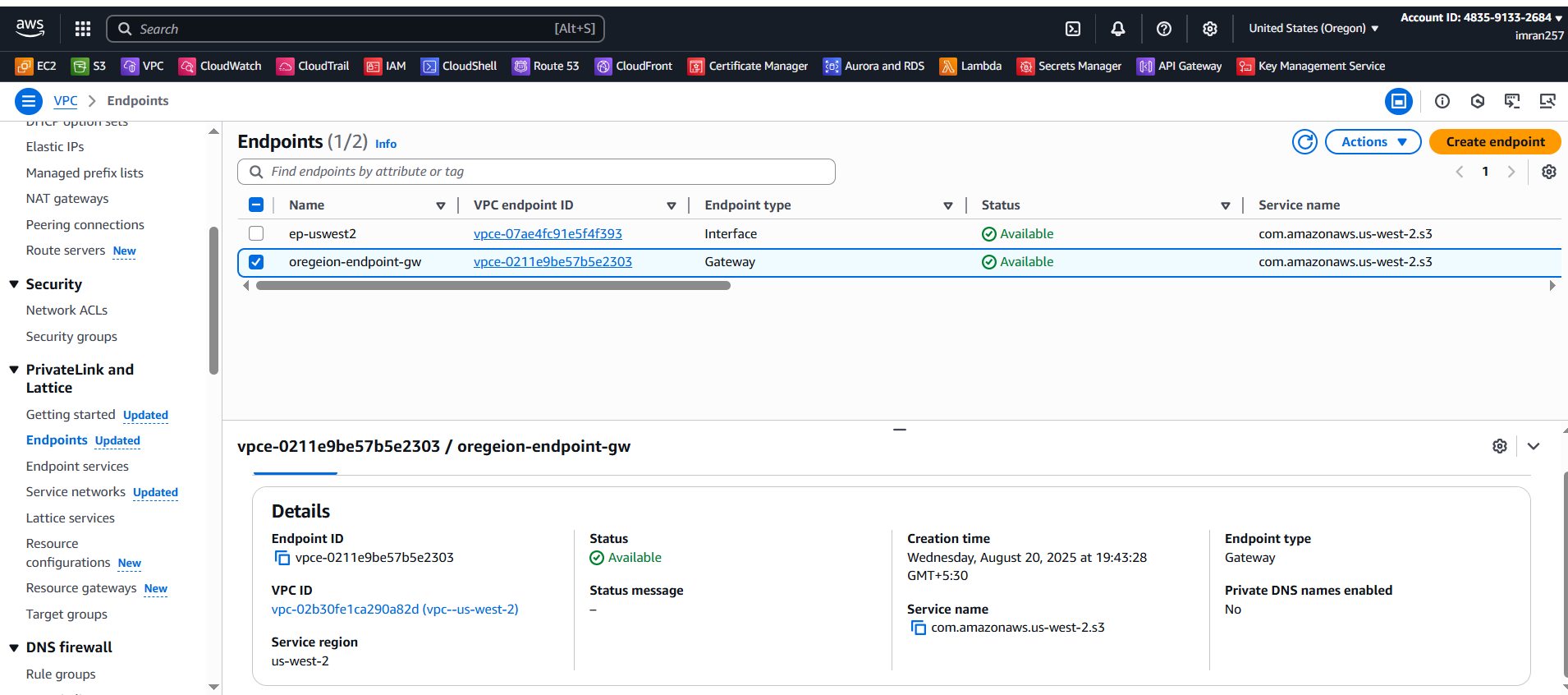


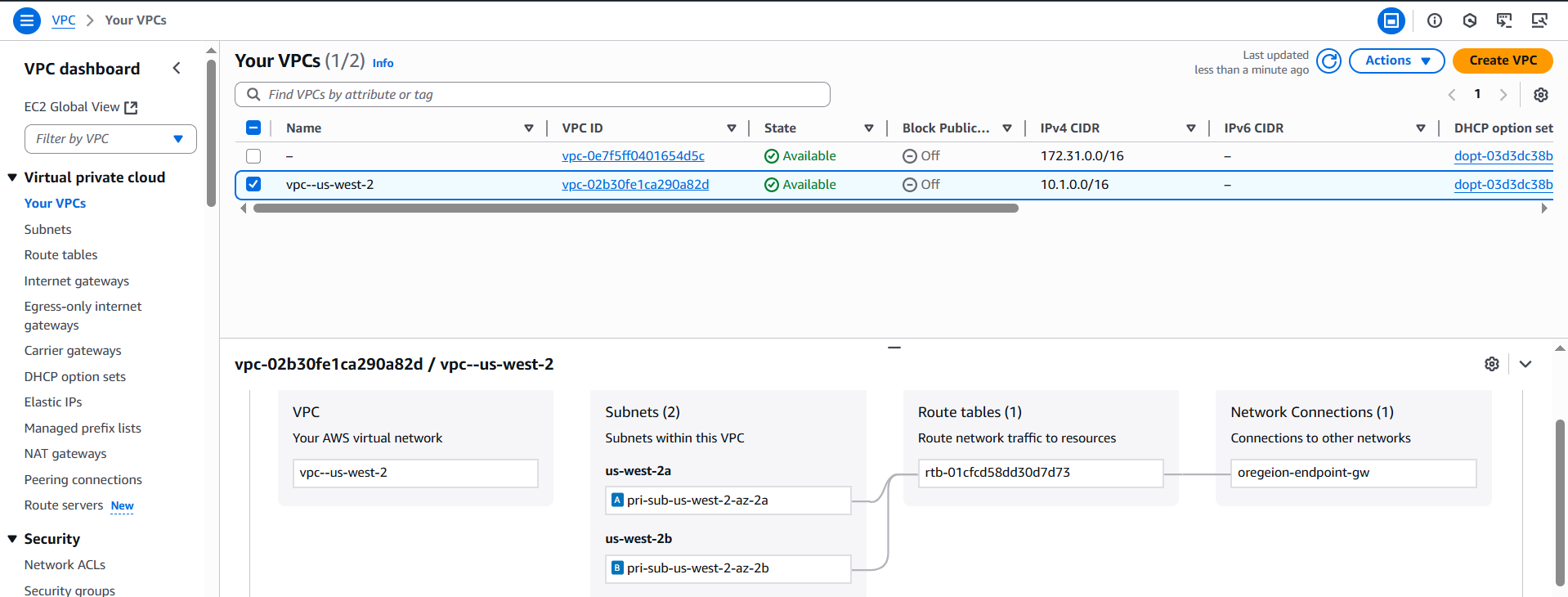
1. Tried to ping Virginia to ireland



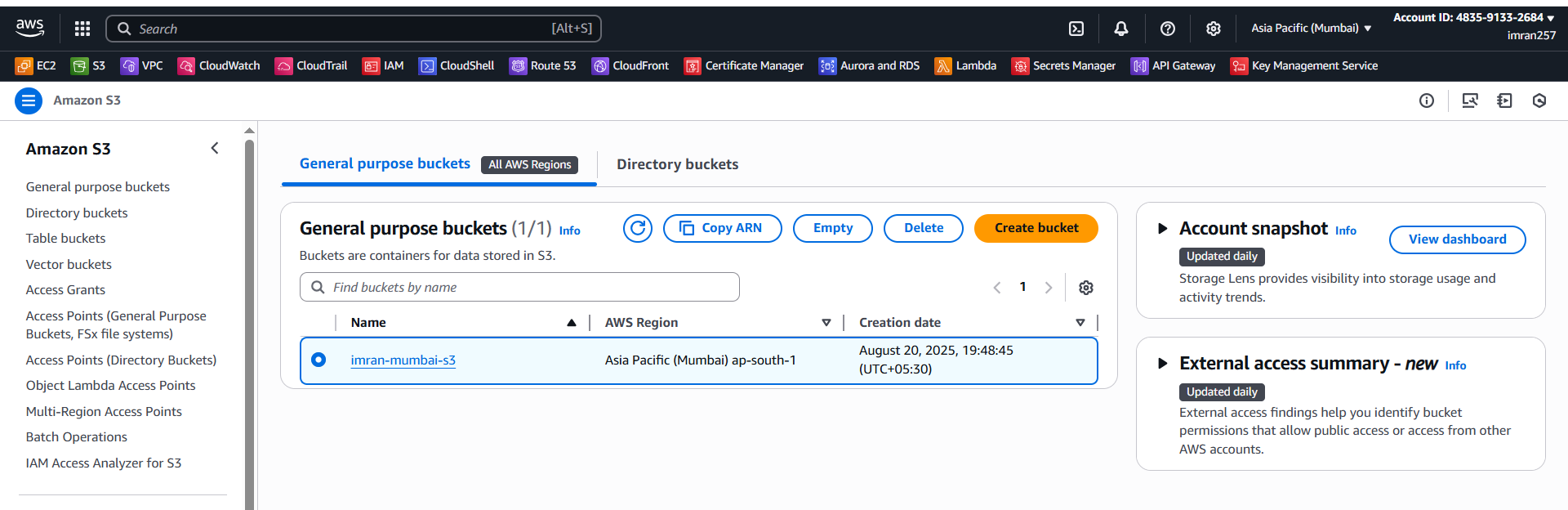
**ACCESSING S3 RESOURCE using ENDPOINT without Internet.**

1. End point created in oregon





1. Created s3 in mumbai region



1. Login ec2 of virginia and make this as jump server , taken pem key of ec2 of oregon and then login to ec2 of oregon which is have private ip and now configure aws and

Aws s3 ls

