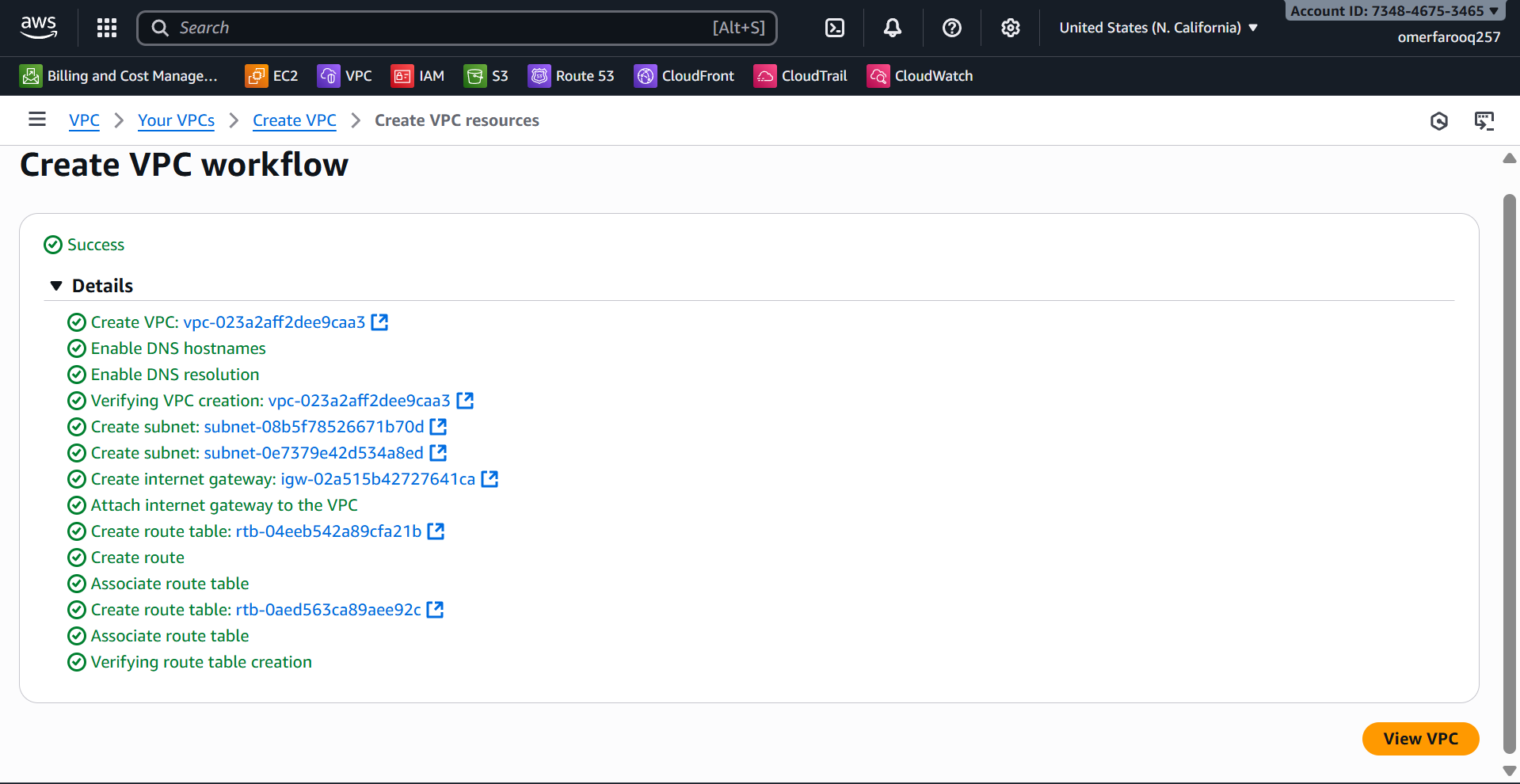
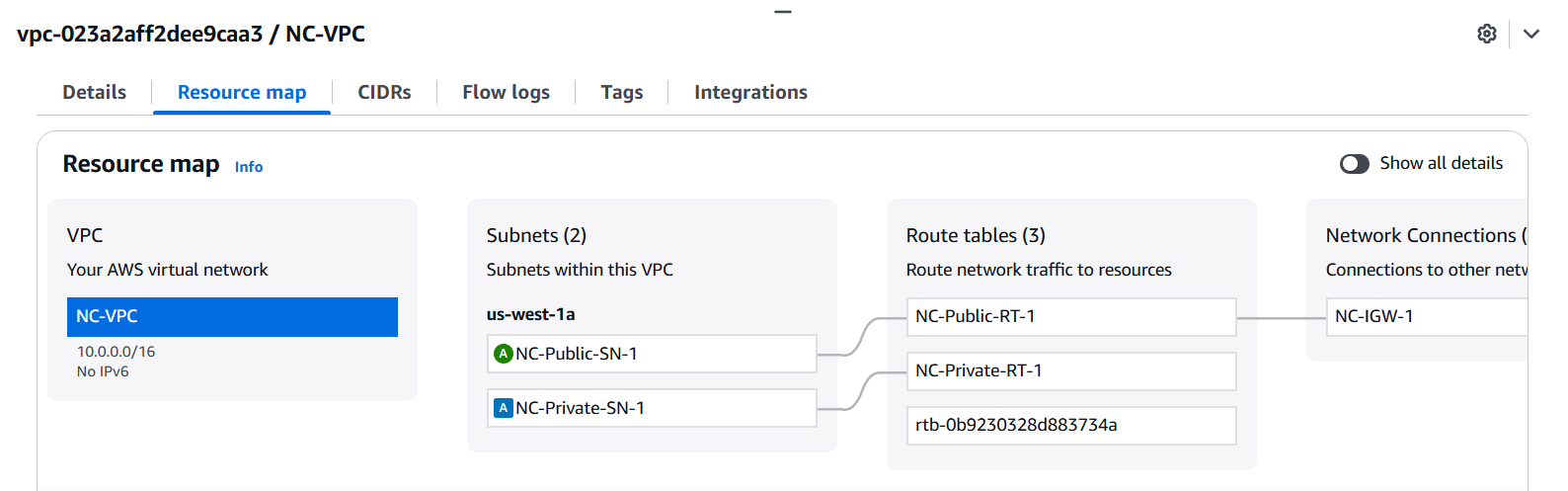
1::Configure VPC peering in cross regions.  
**T**o enable private network connectivity between VPCs without internet exposure.

*STEP TO ENABLE VPC PEERING FOR CROSS-REGION:*

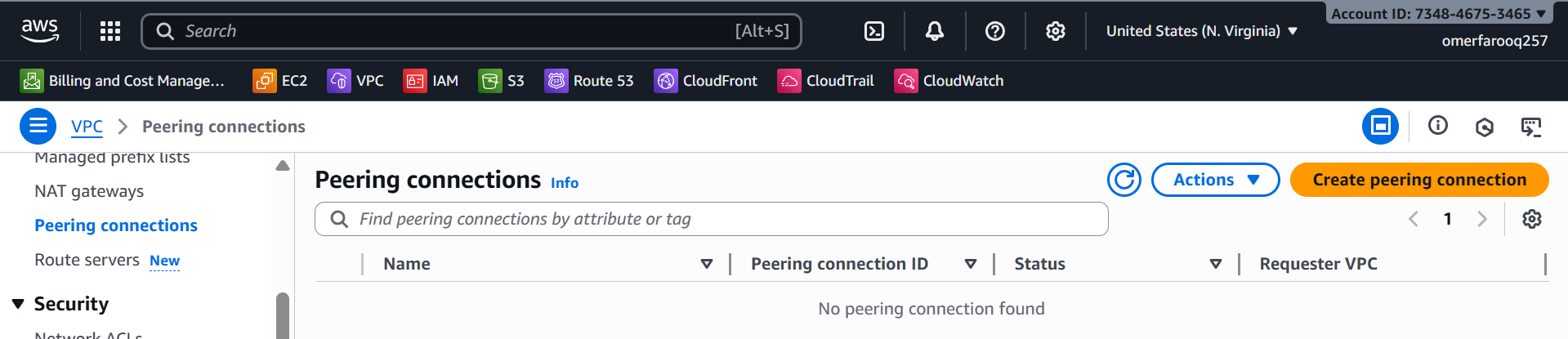
**Create a Second VPC in a Different Region – N. California**



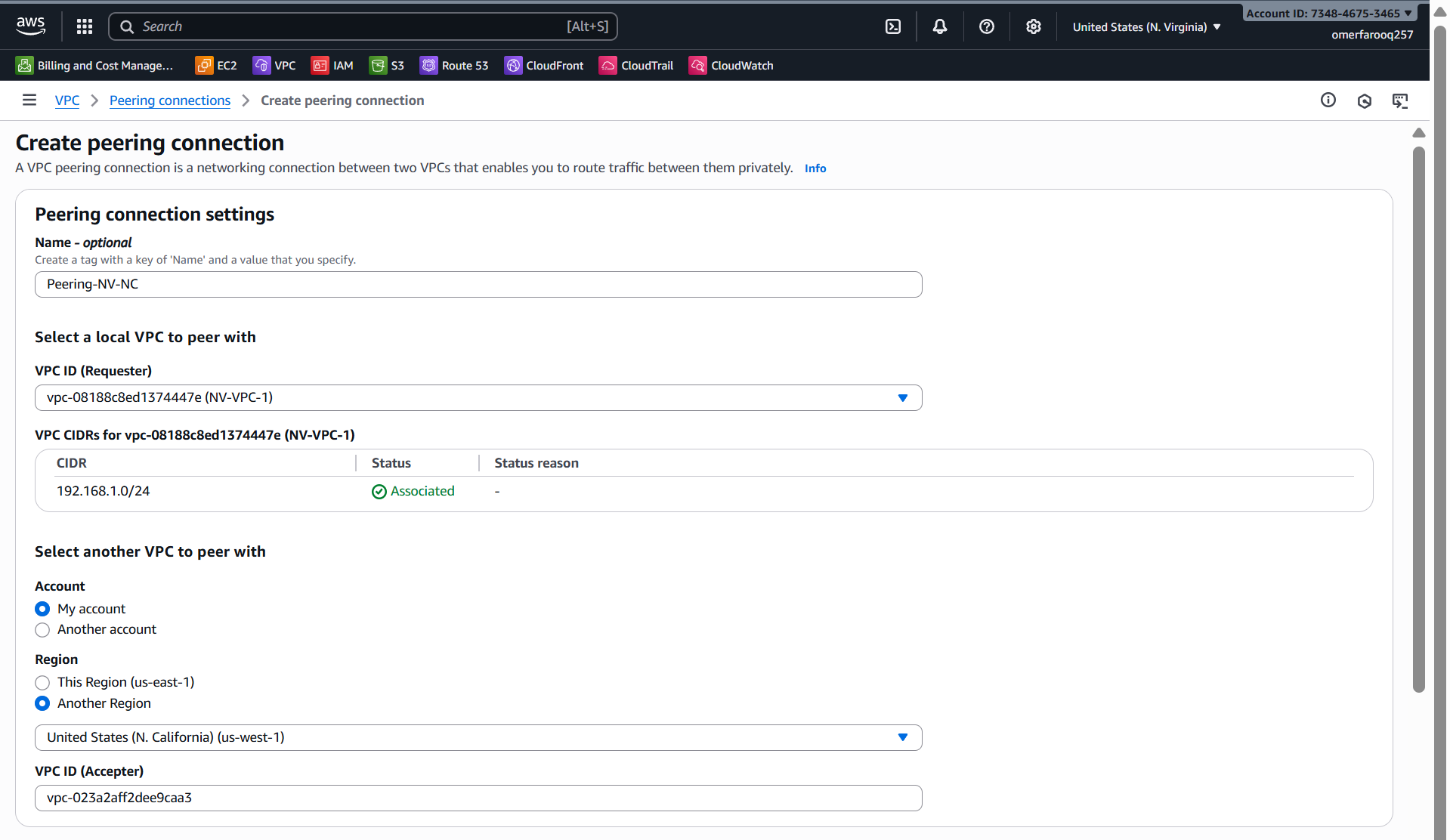
* The architecture of the VPC is shown in the image below.



**Create VPC Peering Connection (Requester Side)**

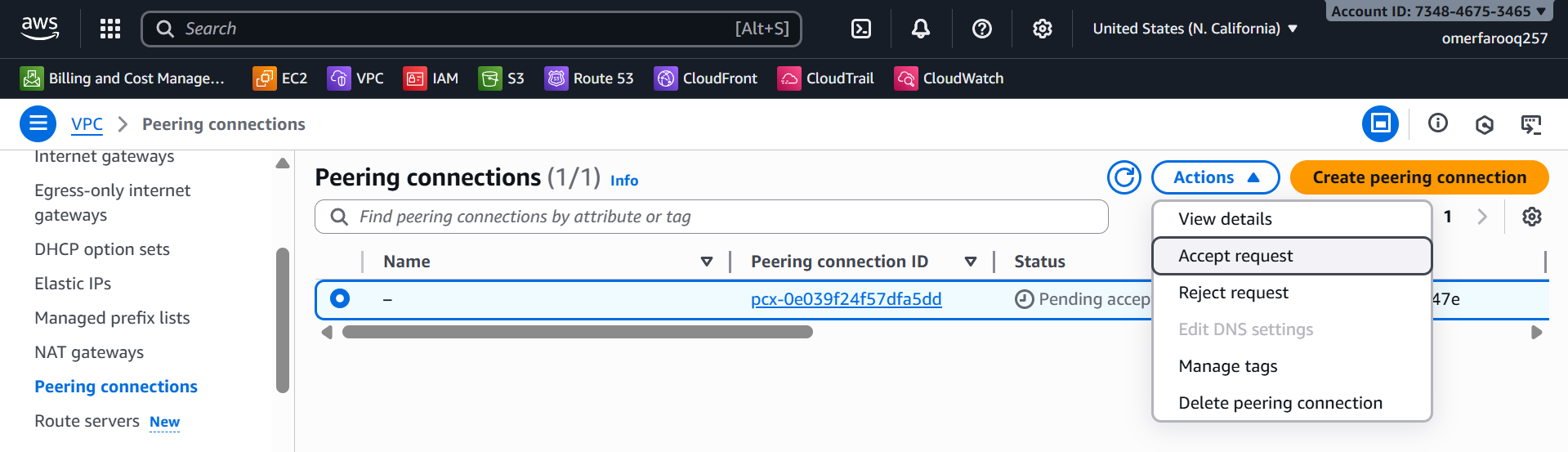


* Navigate to **VPC** → **Peering Connections**
* Click **Create Peering Connection**
* Configuration:
  + Peering connection name tag: Peering-NV-NC
  + **VPC ID (Requester):** vpc-0a1b2c3d4e5f6g7h8 (NV-VPC-1) - Region: us-east-1
  + VPC to accept peering: - Account: [Your AWS Account ID]
  + **VPC ID (Accepter)**: vpc-0i1j2k3l4m5n6o7p8 (NC-VPC) - Region: us-west-2
* Click **Create Peering Connection**



**Accept VPC Peering Connection (Accepter Side)**

* Switch to the us-west-2 region in the AWS Console
* Navigate to **VPC → Peering Connections**
* Find the pending peering connection
* Select it. **Actions → Accept Request**
* Click **Accept Peering Connection**





**Enable DNS Resolution Over Peering**

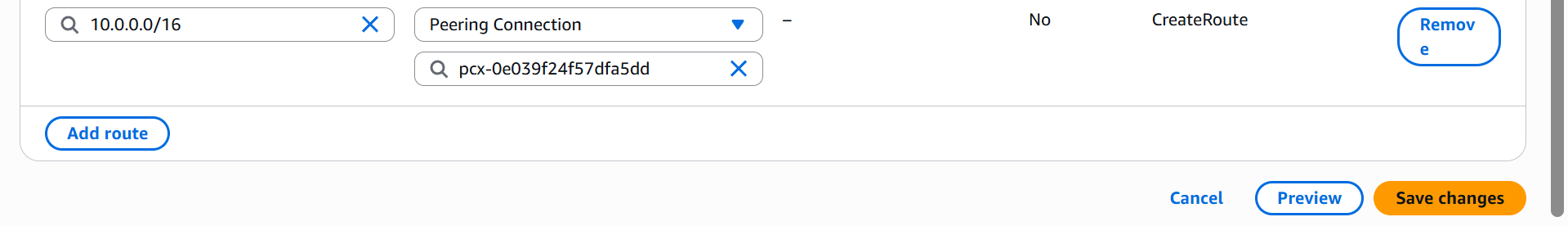
* Navigate to **VPC → Peering Connections**
* Select the peering connection
* Click **Actions → Edit Peering Connection Options**
* Check: ✅ **Allow DNS resolution from remote VPC endpoint**
* Click **Save**

**Update Route Tables (Requester Side - us-east-1)**

* Navigate to **VPC → Route Tables**
* Select **NV-Public-RT**
* **Routes tab → Edit Routes**

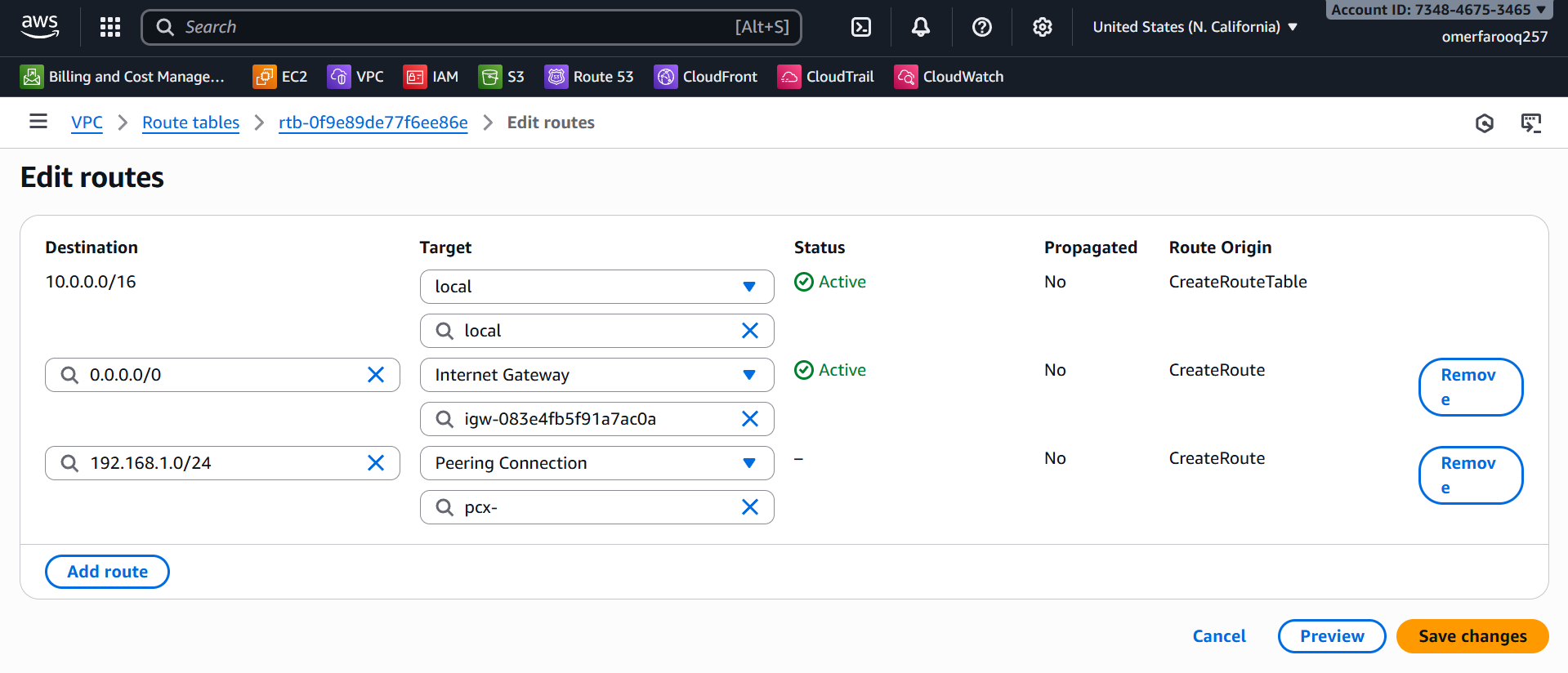


* Click **Add Route**
  + Destination: **10.0.0.0/16**
  + Target: **Peering Connection → Peering-NV-NC**
* Save
* Repeat for NV-Private-RT



**Update Route Tables (Accepter Side - us-west-2)**

* Switch to us-west-2 region
* Navigate to **VPC → Route Tables**
* Select the west subnet's route table
* Edit Routes
* Add route:
  + Destination: **192.168.1.0/24**
  + Target: **Peering Connection → East-West-Peering**

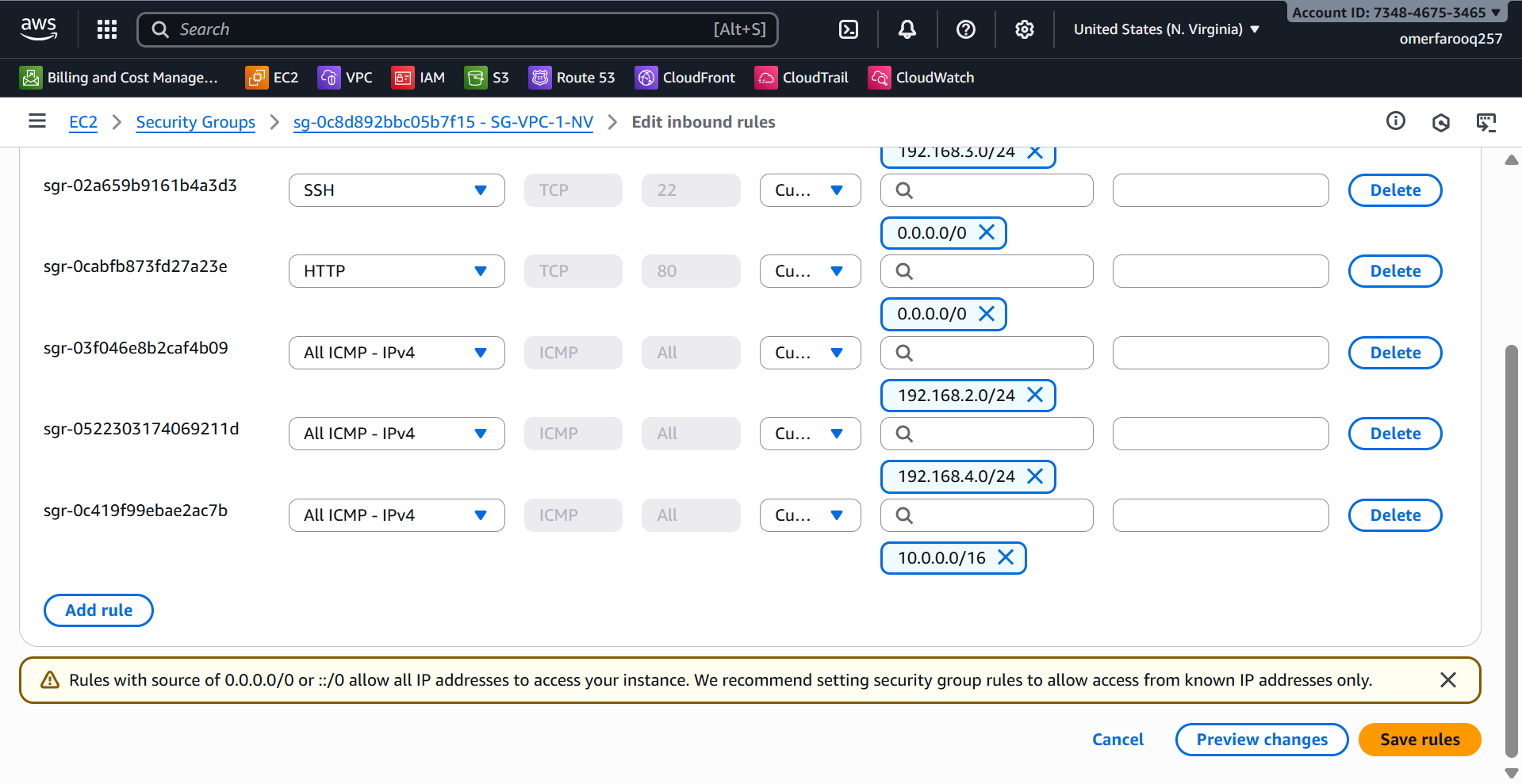
****

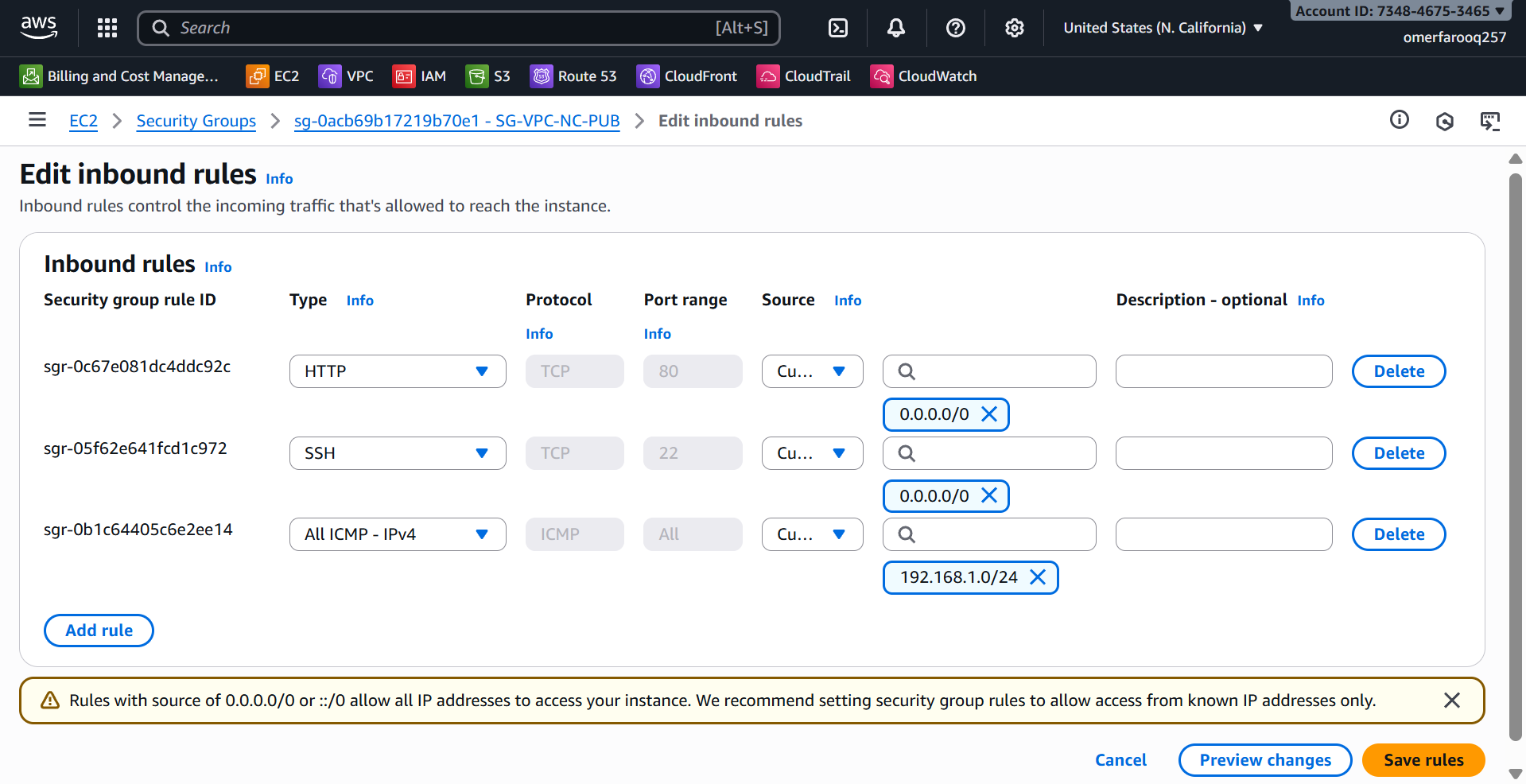
**Update Security Groups for Peering for instances in NV-VPC (us-east-1):**

* Click on **Add rule**
  + Protocol: **All ICMP – IPv4**
  + Custom CIDR: **10.0.0.0/16**

**Update Security Groups for Peering for instances in NC-VPC (us-west-1):**

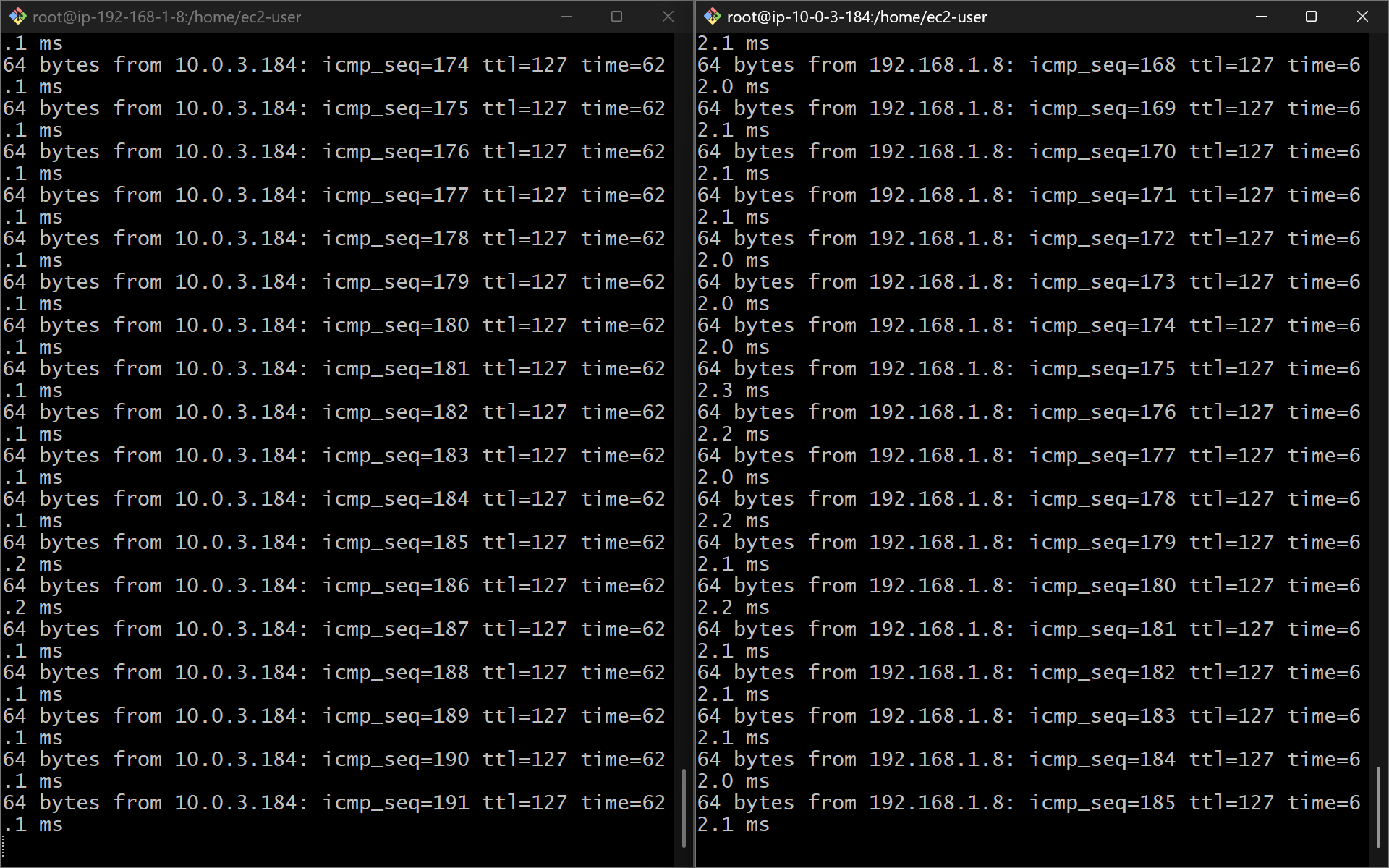
* Click on **Add rule**
  + Protocol: **All ICMP – IPv4**
  + Custom CIDR: **192.168.1.0/16**

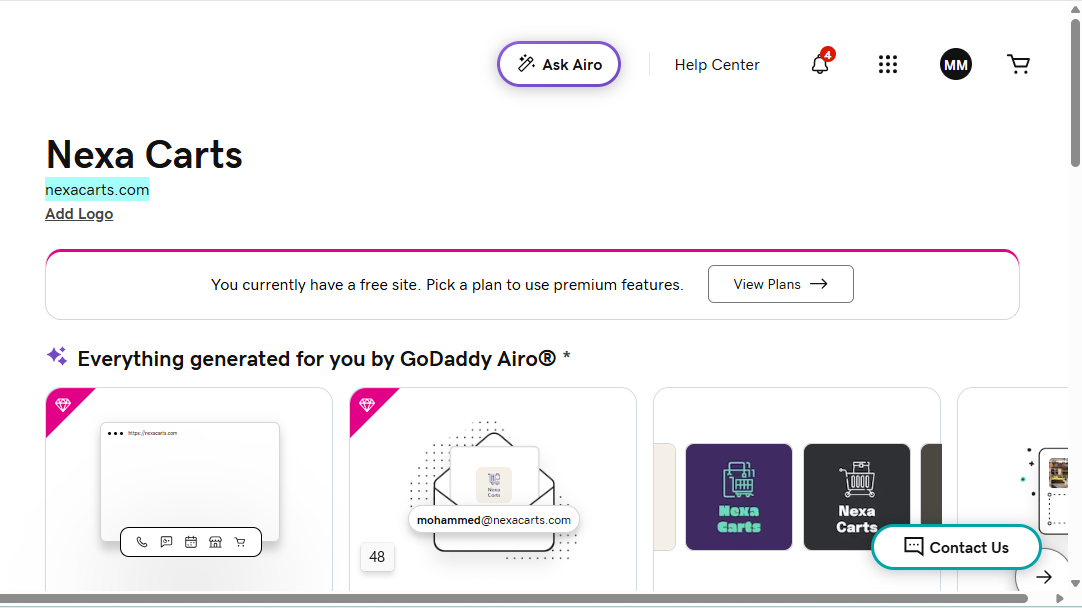




**Test the connection between two VPCs in different regions:**

* Launch an EC2 instance in N. Virginia and another in N. California.
* Configure the instances and assign the appropriate:
  + VPC
  + Subnet
  + Key
  + Security Group
* Launch the instances in Git Bash and check the connectivity between the two servers.
  + # ping 10.0.3.184
  + # ping 192.168.1.8



2: Purchase one domain from GoDaddy.  
  
purpose for puschasing domain: We need a domain name because it provides a human-friendly, memorable address for a website, which is essential for branding  
  
  
steps to purchase domain: go to browser and search godaddy and login and search for your desire domain name.   
#then purchase the domain and copy your domain name.

# 3:Deploy static website in S3. purpose: we can host static website by using EC2 as well but we prefer to go with S3 because S3 is more cheaper than EC2 and it is globally available and easy to host. Steps: first go to S3 services in aws and click on create bucket. >then select the unique name for your bucket. >disable block to public and enable ACL firewall for hosting and allow for public. >enable the virsioning option. Because in real time scenario you make some changes in your object so it will save both. >click on create stesp: > if you want to host your static website then. >click on bucket and go to bucket properties and scroll down to last and enable the static website option and save change. steps: upload the index.html named objects. >then select the files and go to actions and scroll down and make it public available. >then confirm and save. steps: then go to buckets and go to properties and scroll down. > then you see your buclket URL copy this and chack in your browser the web site is hosted successfully. steps: here you can see the website is sucssfully hosted and you can see the result.

# 4:Create a CDN and attach one SSL certificate.

## ****📌 Purpose****

To improve website performance, reduce latency, and securely deliver content over HTTPS by using AWS CloudFront as a Content Delivery Network (CDN).  
An SSL certificate (ACM) is attached to enable **HTTPS encryption** for the custom domain.

## steps: ****📌 Steps****

### ****1. Open CloudFront Console****

* Go to **AWS Console → CloudFront → Create Distribution**

### ****2. Configure Origin****

* **Origin Domain:** Select the S3 website endpoint  
  (Example: hosting-example.s3-website-us-east-1.amazonaws.com)
* **Protocol:** HTTP only  
  (CloudFront handles HTTPS on the viewer side)

### ****3. Viewer Settings****

* Viewer Protocol Policy → **Redirect HTTP to HTTPS**
* Allowed Methods → GET, HEAD

### ****4. Add Custom Domain (CNAME)****

Enter:

* yourdomain.com
* www.yourdomain.com

### ****5. Attach SSL Certificate****

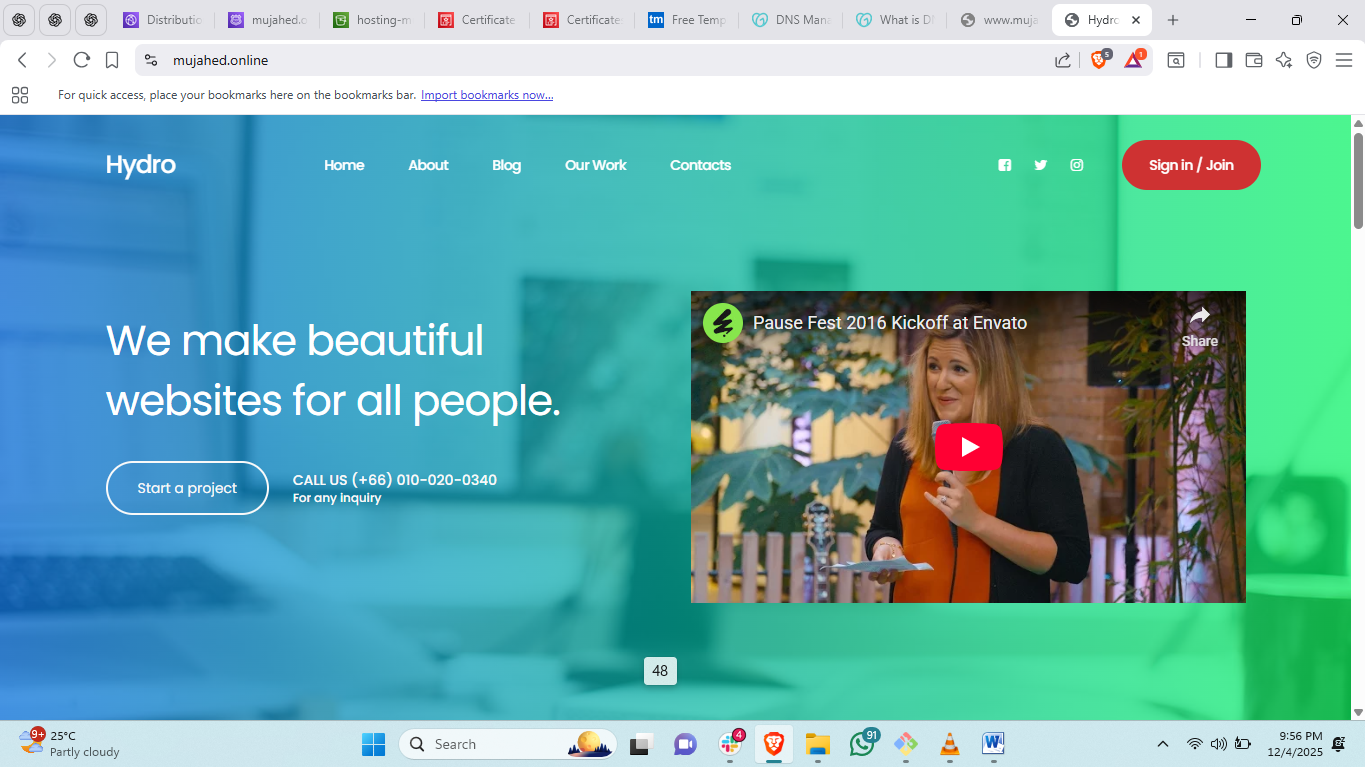
* Click **Request Certificate in ACM**
* Choose **DNS Validation**
* Add both domains:
  + yourdomain.com
  + www.yourdomain.com
* ACM → Creates **CNAME records** (auto-create in Route 53)
* Wait for status → **Issued**

### ****6. Assign Certificate to CloudFront****

* Return to CloudFront Distribution setup
* Choose the SSL certificate you created
* Create the distribution

### ****7. Wait for Deployment****

* Status changes from **InProgress → Deployed**



### 5:Create a Route 53 hosted zone and map the domain with the CDN. purposeTo route internet traffic from your GoDaddy domain name to the CloudFront distribution using DNS. Route 53 becomes the authoritative DNS server for your domain. steps: ****1. Create Hosted Zone****

* Go to **Route 53 → Hosted Zones → Create hosted zone**
* Domain: yourdomain.com
* Type: **Public hosted zone**

### ****2. Copy NS (Name Server) Records****

Route 53 generates 4 nameservers like:

ns-1690.awsdns-19.co.uk

ns-460.awsdns-57.com

ns-540.awsdns-03.net

ns-1245.awsdns-27.org

### ****3. Update Nameservers in GoDaddy****

* Open **GoDaddy → DNS Management**
* Change Nameservers → **Custom**
* Paste all 4 NS values from Route 53

### ****4. Create A Records for CloudFront****

In Route 53 Hosted Zone:

#### ****A Record 1 (root domain)****

* Record name: (leave blank)
* Type: A
* Alias: YES
* Target: CloudFront distribution domain  
  (Example: d3d0f3mdw3uy4n.cloudfront.net)

#### ****A Record 2 (www subdomain)****

* Record name: www
* Type: A
* Alias: YES
* Target: CloudFront distribution domain

### ****5. Save and propagate****

DNS propagation may take 5–30 minutes.

## 6:Update the index.html in the S3 bucket and ensure the updated file is accessible using the domain name. purpose: To confirm that the website is served through CloudFront and updates are reflected when accessing the domain name. This verifies end-to-end connectivity: S3 → CloudFront → Route 53 → Domain. steps: ****📌 Steps****

### ****1. Modify index.html****

* Open your local index.html
* Add some text for versioning, e.g.:
* <h1>Website Version 2</h1>

### ****2. Upload Updated File to S3****

* Go to **S3 → Bucket → Upload**
* Replace existing index.html

### ****3. Invalidate CloudFront Cache****

* Go to **CloudFront → Distribution → Invalidations → Create Invalidation**
* Path:
* /\*

### ****4. Test in Browser****

Visit:

* <https://yourdomain.com>
* <https://www.yourdomain.com>

Ensure the **updated content appears**.

7:Share the domain name in Slack to test the connectivity.

## my link: https://mujahed.online/