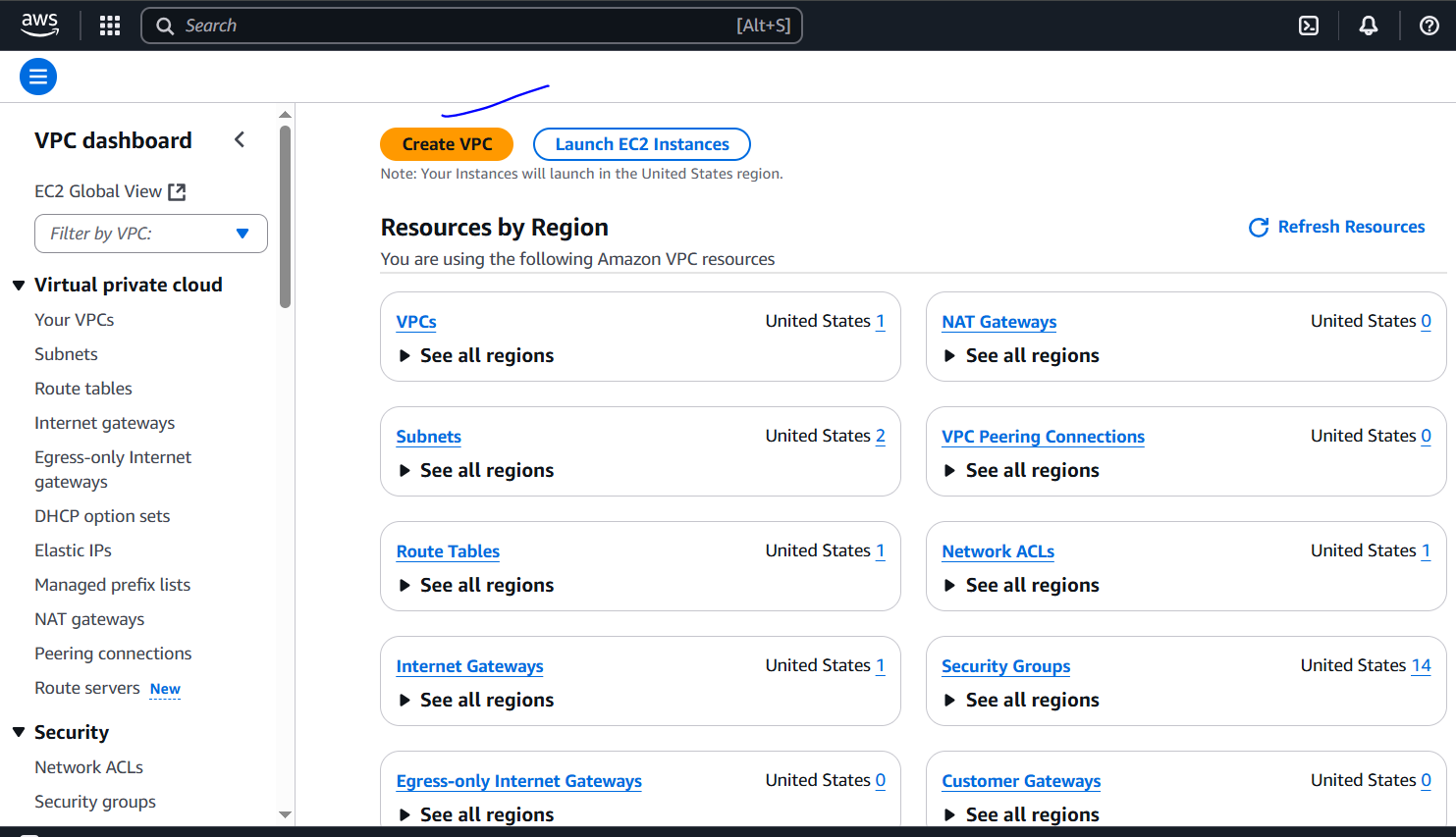
**VPC Task-01**

**1) Create VPC with 2 private and 2 public subnets.**

**Step 1: Create a VPC**

1. Go to the AWS Management Console and navigate to the VPC dashboard.

2. Click "Create VPC".



3. Enter the following details:

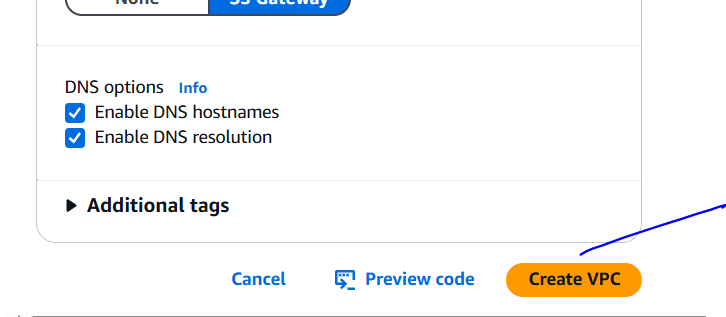
- IPv4 CIDR block: 10.0.0.0/16

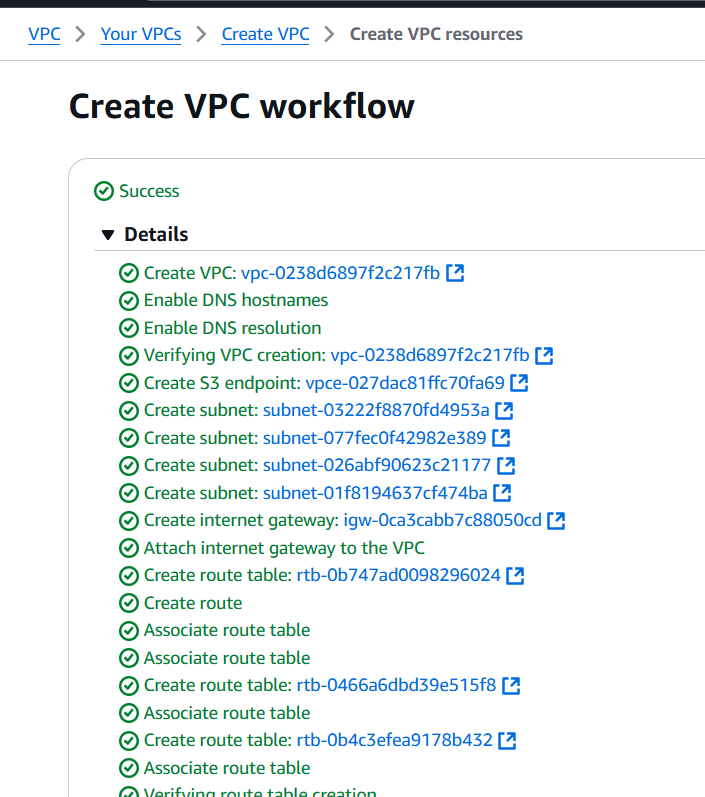
- IPv6 CIDR block: Optional (you can leave it as default)

- VPC name: MyVPC

- Tenancy: Default

4. Click "Create VPC".

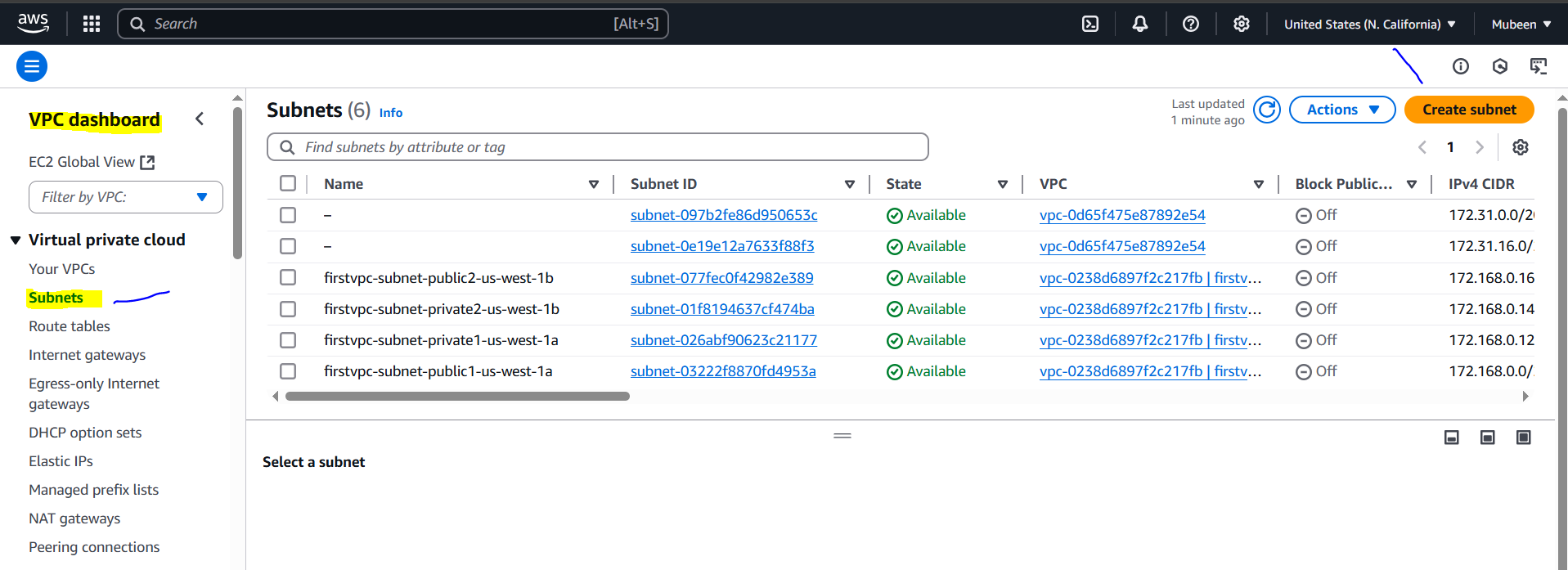




**Step 2: Create Subnets**

1. Click "Subnets" in the left sidebar.

2. Click "Create subnet".



**Public Subnets:**

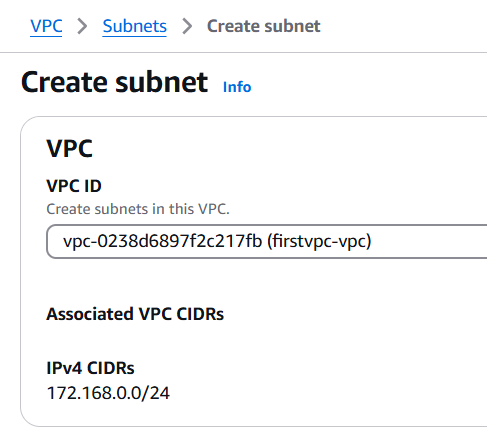
**1. Create Public Subnet 1:**

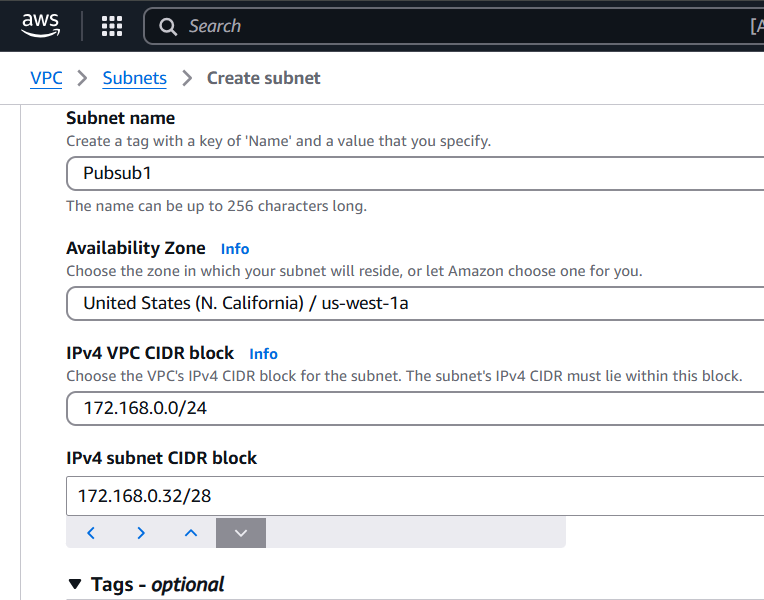
- VPC: Select the VPC you created (MyVPC)

- Subnet name: PubSub1

- Availability Zone: Choose an AZ (e.g., us-west-1a)

- IPv4 CIDR block: 172.168.0.32/28





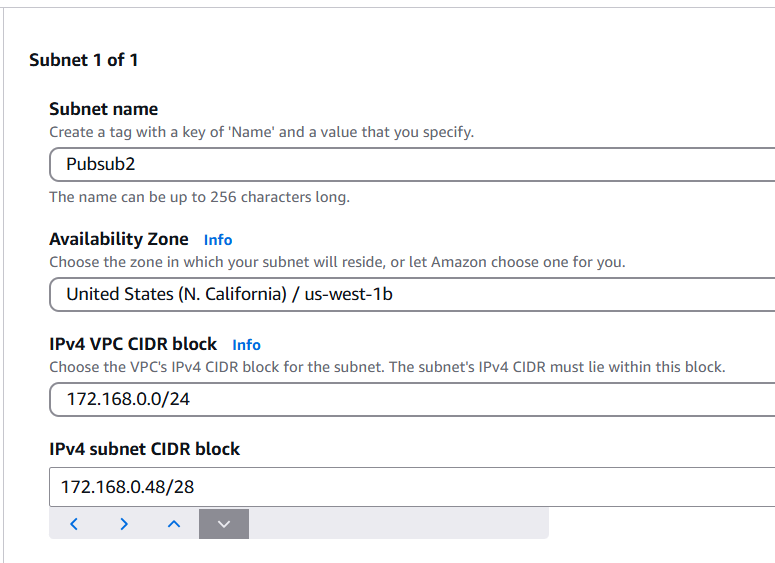
**2. Create Public Subnet 2:**

- VPC: Select the VPC you created (MyVPC)

- Subnet name: PubSub2

- Availability Zone: Choose a different AZ (e.g., us-west-1b)

- IPv4 CIDR block: 172.168.0.48/28



**Private Subnets:**

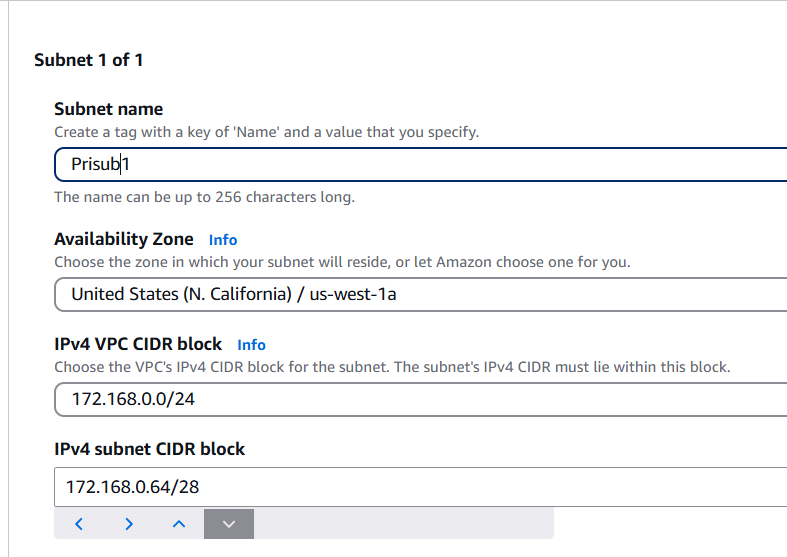
**1. Create Private Subnet 1:**

- VPC: Select the VPC you created (MyVPC)

- Subnet name: PriSub1

- Availability Zone: Choose an AZ (e.g., us-west-1a)

- IPv4 CIDR block: 172.168.0.64/28



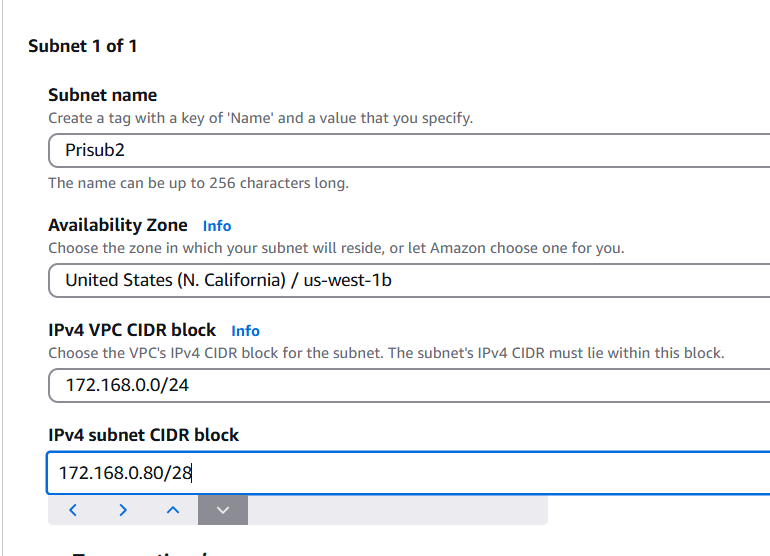
**2. Create Private Subnet 2:**

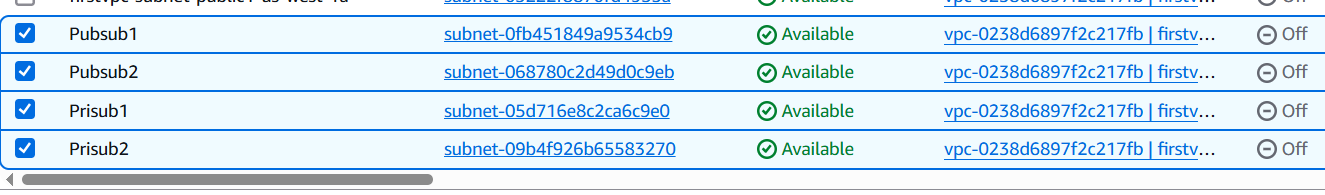
- VPC: Select the VPC you created (MyVPC)

- Subnet name: PrivateSubnet2

- Availability Zone: Choose a different AZ (e.g., us-west-1b)

- IPv4 CIDR block: 172.168.0.80/28



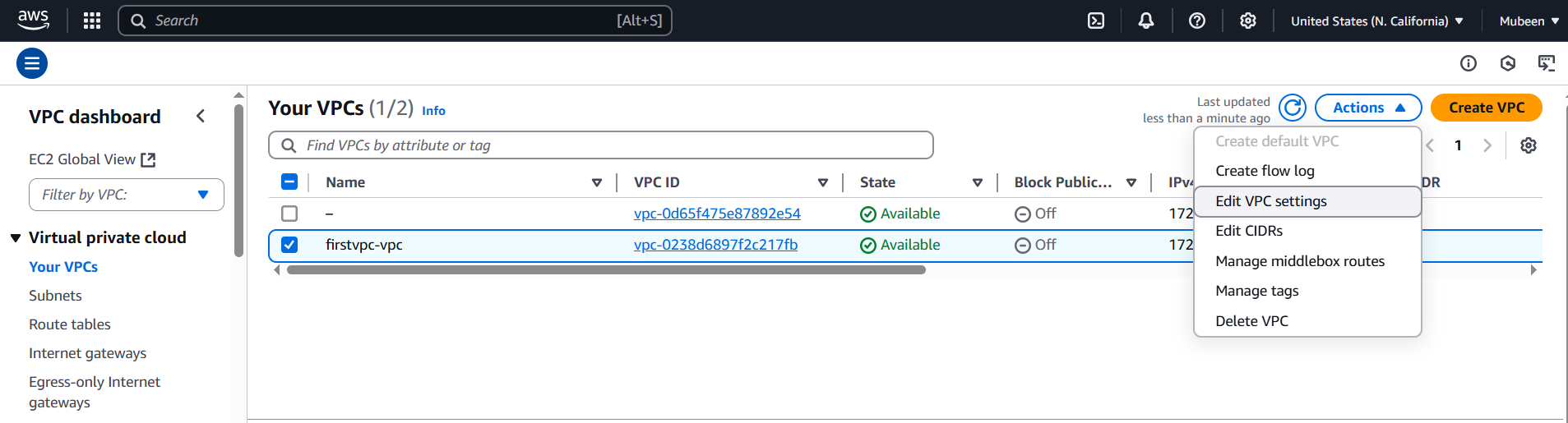


**2) Enable DNS Hostname in VPC**

1. Go to the AWS Management Console and navigate to the VPC dashboard.

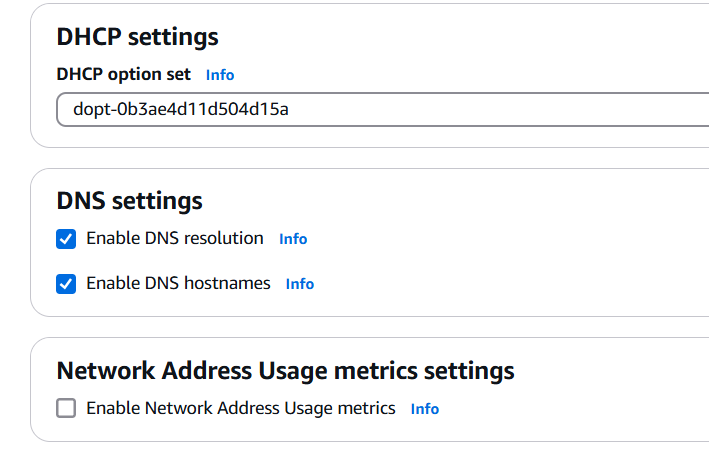
2. Select the VPC for which you want to enable DNS hostname.

3. Click "Actions" and select "Edit VPC settings".



4. In the "DNS settings" section, select the checkbox next to "DNS hostnames".

5. Click "Save changes".



By enabling DNS hostname, instances launched in this VPC will receive a DNS hostname that can be used to access them.

**3) Enable Auto Assign Public ip in 2 public subnets**

1. Go to the AWS Management Console and navigate to the VPC dashboard.

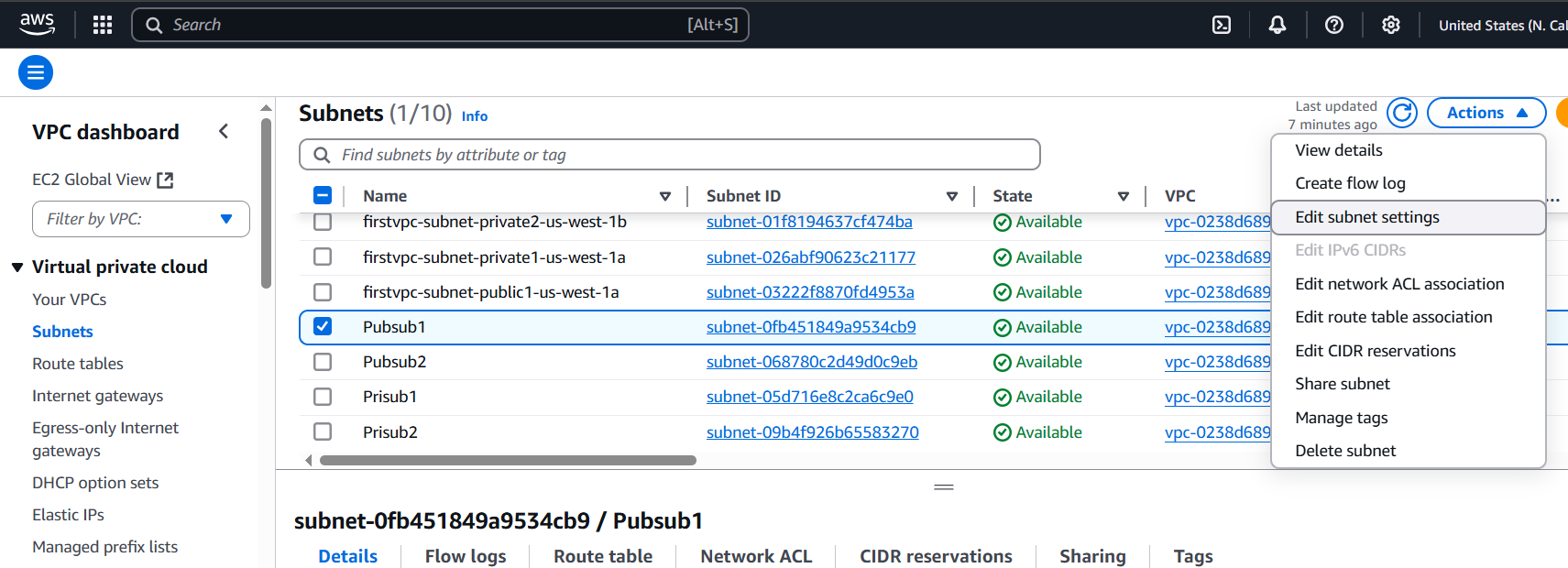
2. Click "Subnets" in the left sidebar.

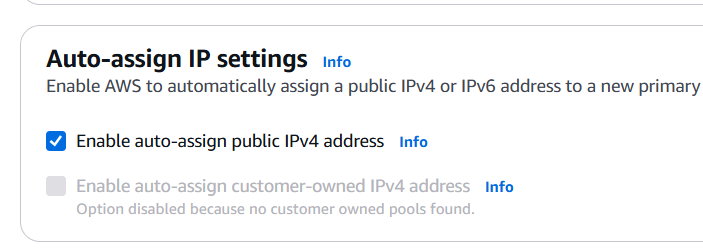
3. Select the public subnet by checking the box next to it.

4. Click "Actions" and select "Edit subnet".

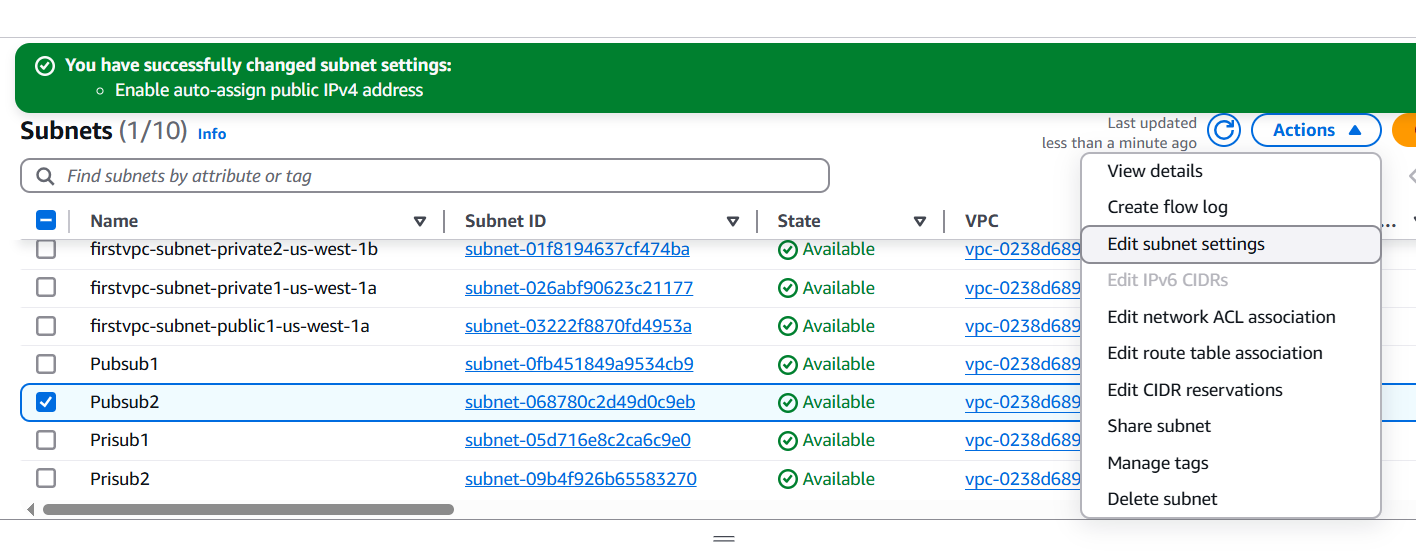
5. In the "Auto-assign public IPv4 address" section, select "Enable".

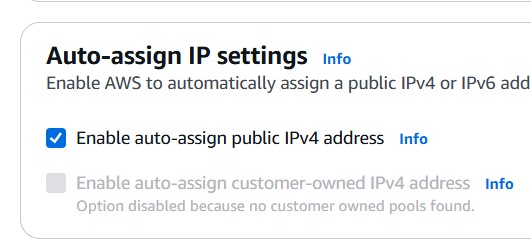
6. Click "Save".





Repeat the same steps for the second public subnet





By enabling Auto Assign Public IP, instances launched in these public subnets will automatically receive a public IP address.

**4) Add 2 private subnets in private route table**

**Associating Private Subnets with Private Route Table:**

1. Go to the AWS Management Console and navigate to the VPC dashboard.

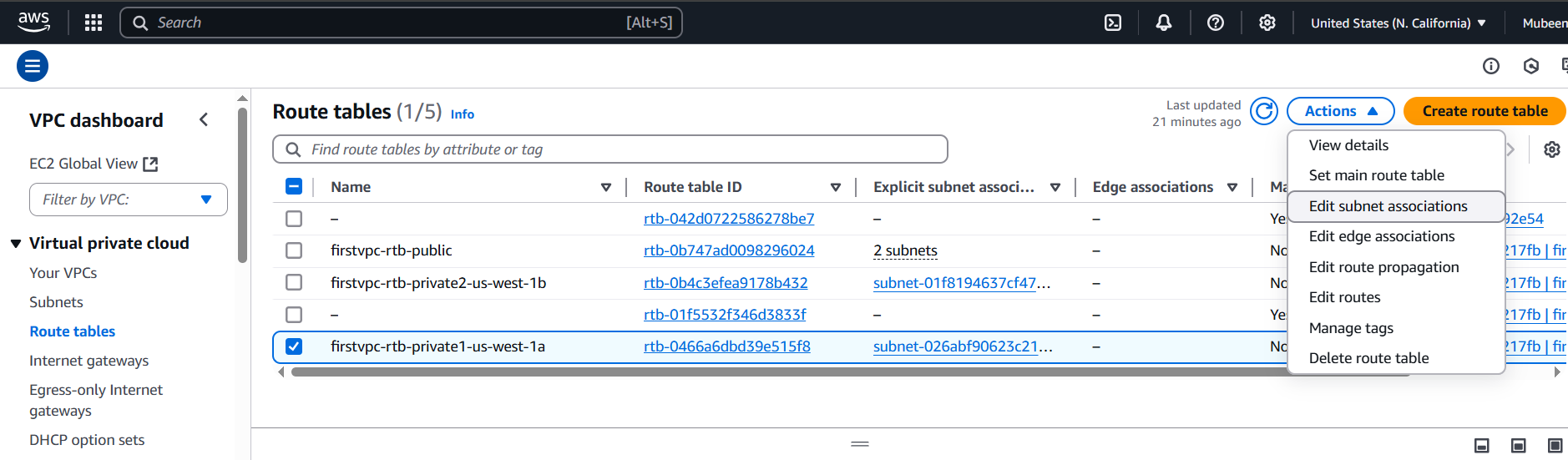
2. Click "Route Tables" in the left sidebar.

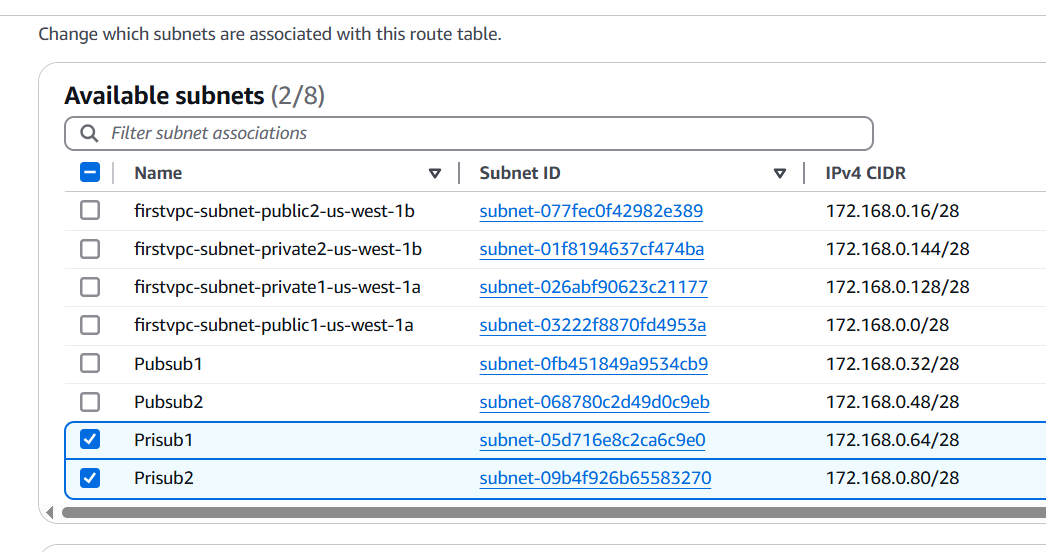
3. Select the private route table.

4. Click "Actions" and select "Edit subnet associations".

5. Select both private subnets (e.g., PriSub1 and PriSub2) by checking the boxes next to them.

6. Click "Save".





**5) Add 2 public subnets in public route table**

**Associating Public Subnets with Public Route Table:**

1. Go to the AWS Management Console and navigate to the VPC dashboard.

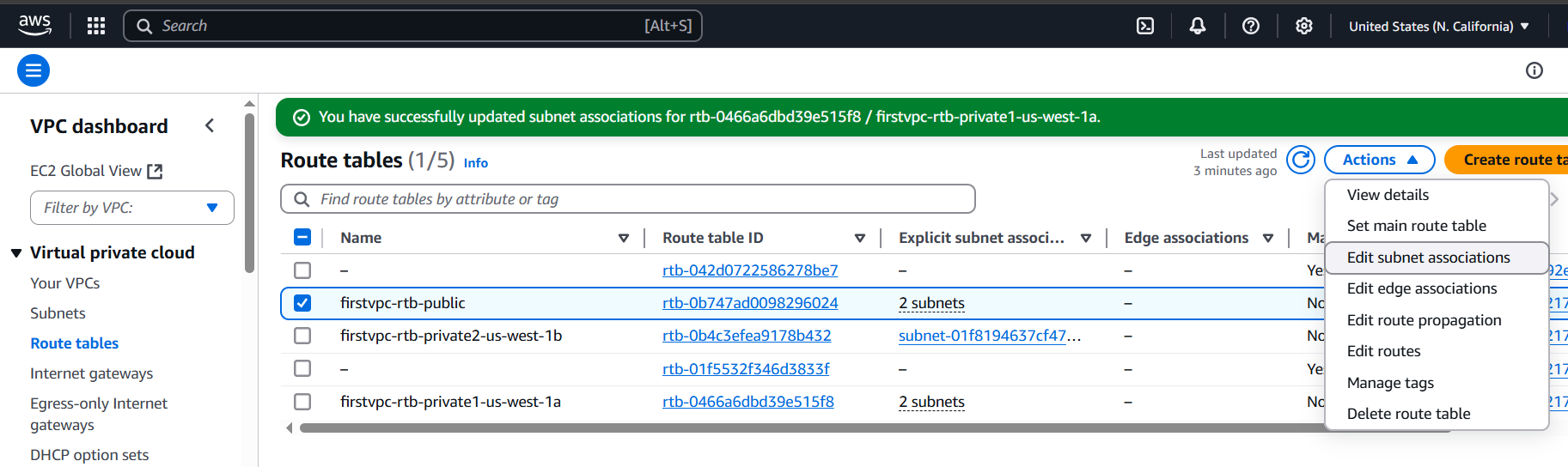
2. Click "Route Tables" in the left sidebar.

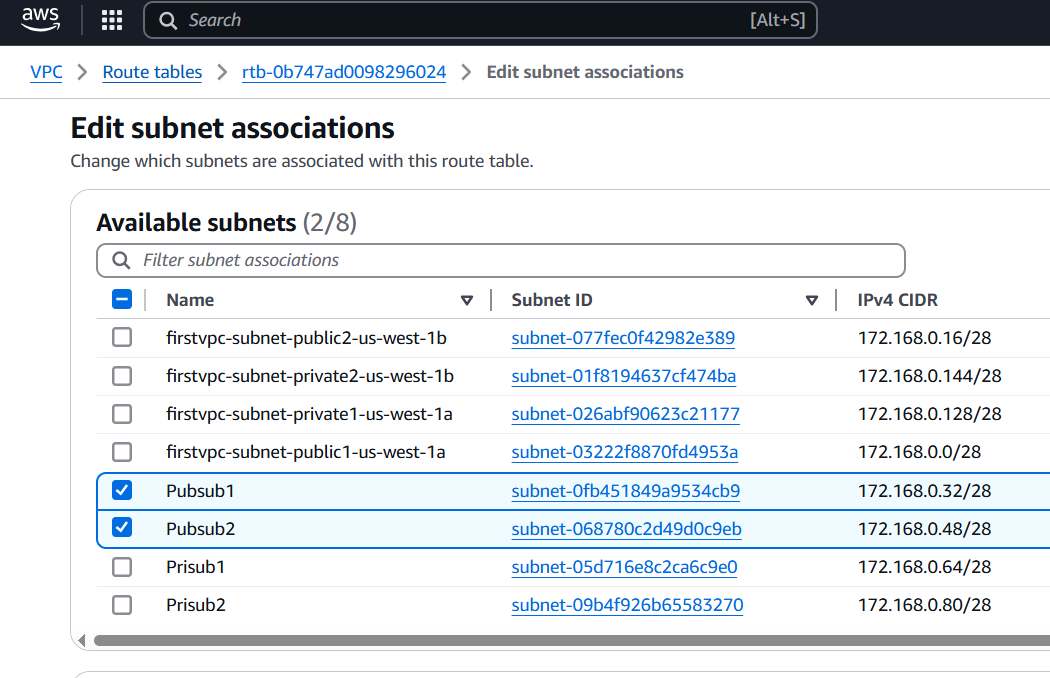
3. Select the public route table (e.g., PublicRouteTable).

4. Click "Actions" and select "Edit subnet associations".

5. Select both public subnets (e.g., PubSub1 and PubSub2) by checking the boxes next to them.

6. Click "Save".





**6) Public route table will have the routes to internet and local**

**Public Subnet Route Table:**

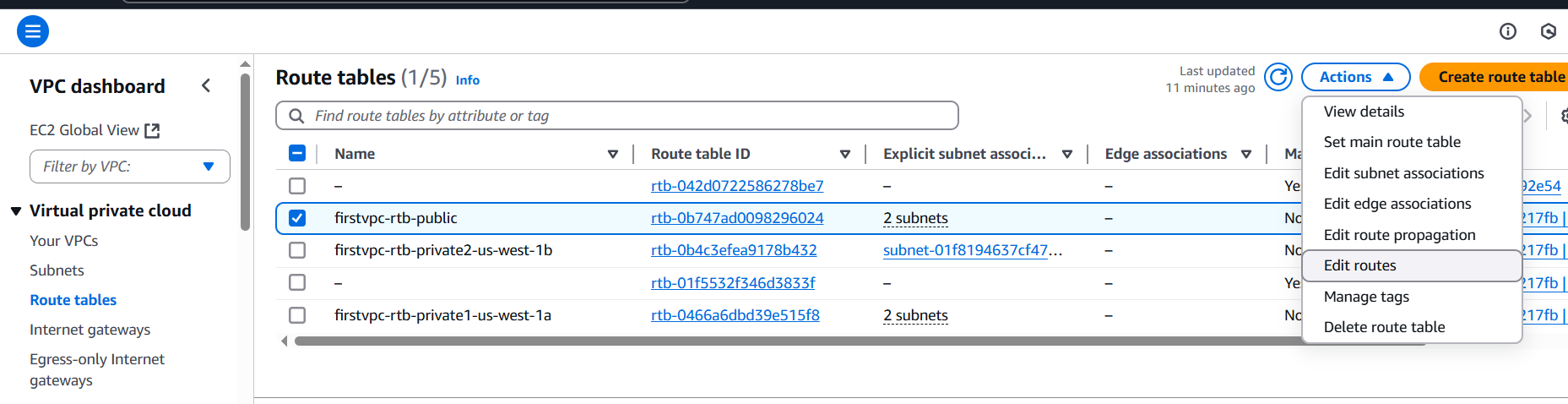
- **Local Route**: Allows communication within the VPC (e.g., 10.0.0.0/16 -> local)

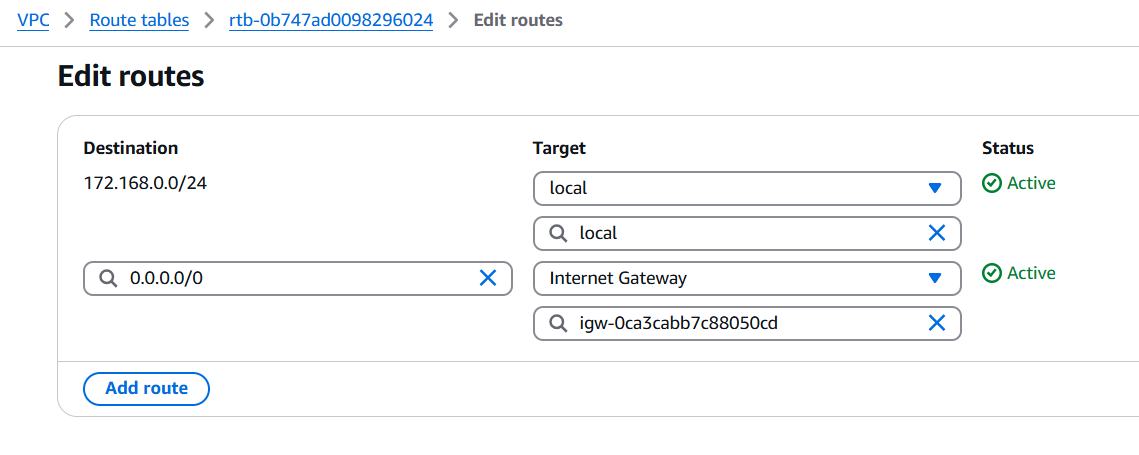
- **Internet Route**: Allows outbound traffic to the Internet (e.g., 0.0.0.0/0 -> Internet Gateway)

**Private Subnet Route Table:**

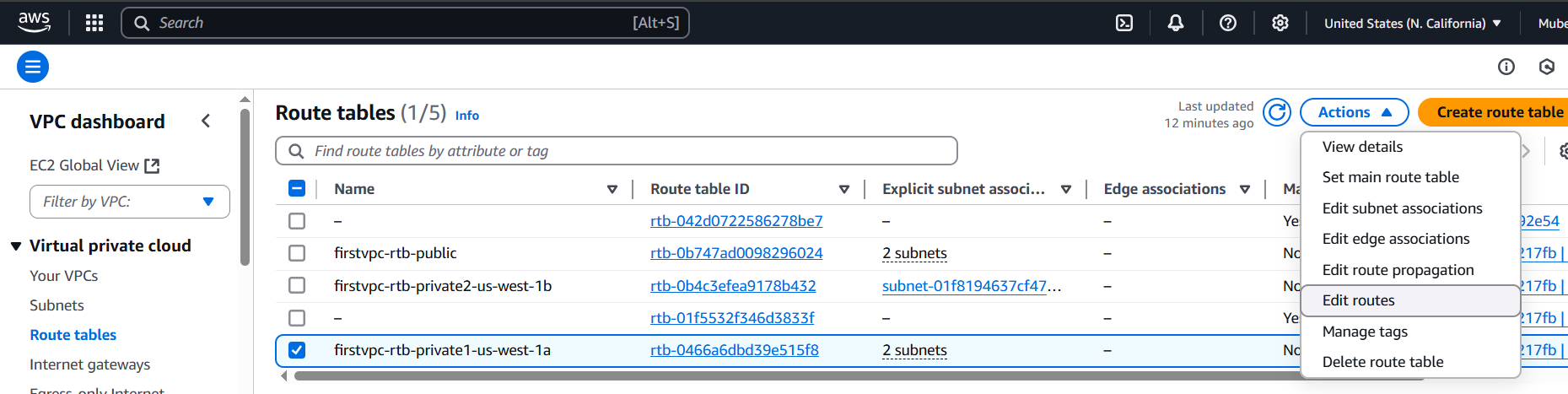
- **Local Route:** Allows communication within the VPC (e.g., 10.0.0.0/16 -> local)

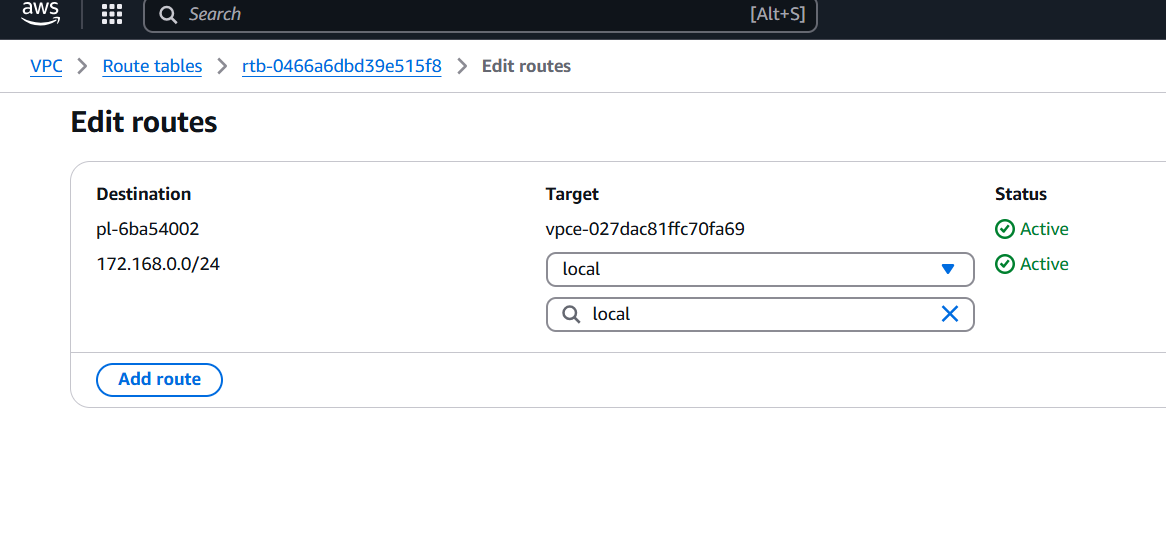
- **No Internet Route**: No direct route to the Internet (e.g., no 0.0.0.0/0 -> Internet Gateway)

Route tables for Public Subnet,  




Route tables for Private Subnet,

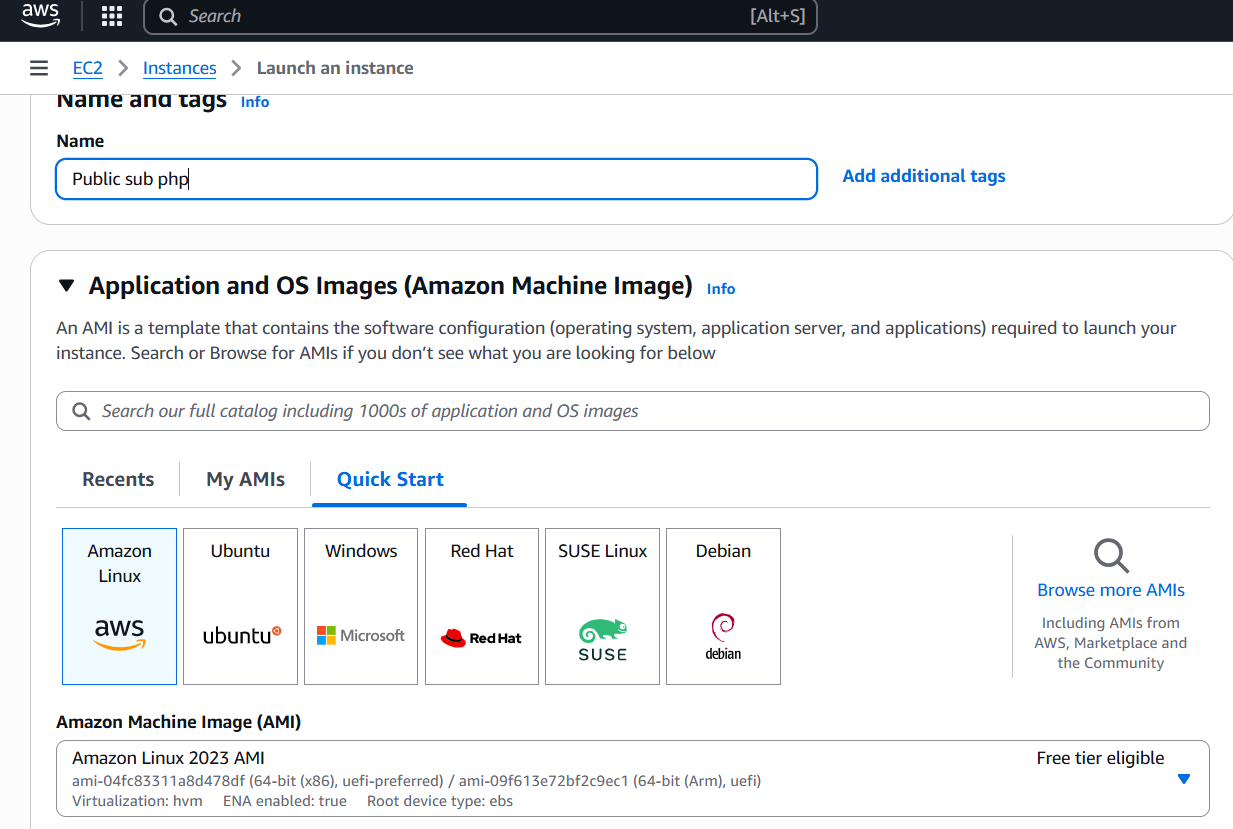




**7) Create Ec2 in public subnet with t2micro and install php**

1. Go to the AWS Management Console and navigate to the EC2 dashboard.

2. Click "Launch Instance" and select "Amazon Linux 2 AMI" (or your preferred OS).



3. Choose the instance type: t2.micro.

4. Configure instance details:

- VPC: Select your VPC.

- Subnet: Select a public subnet (e.g., PublicSubnet1).

- Auto-assign Public IP: Enable.

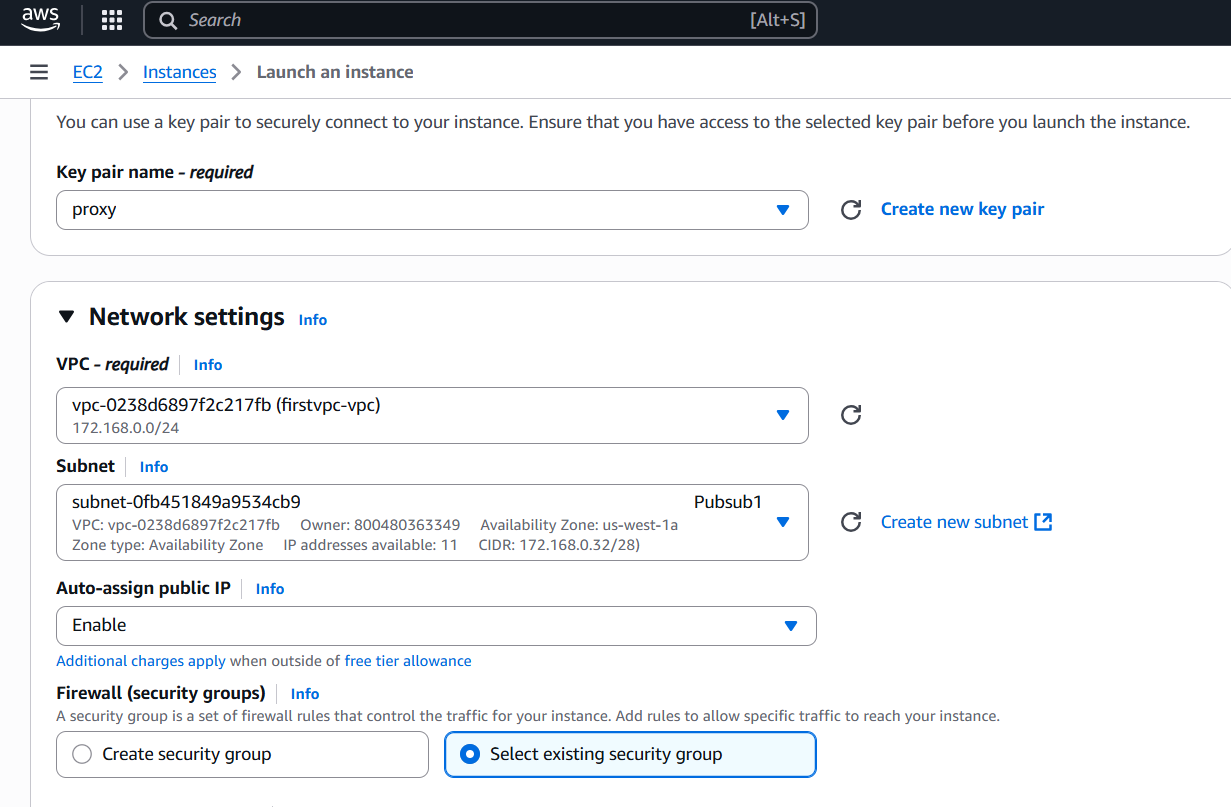
5. Add storage and tags as needed.

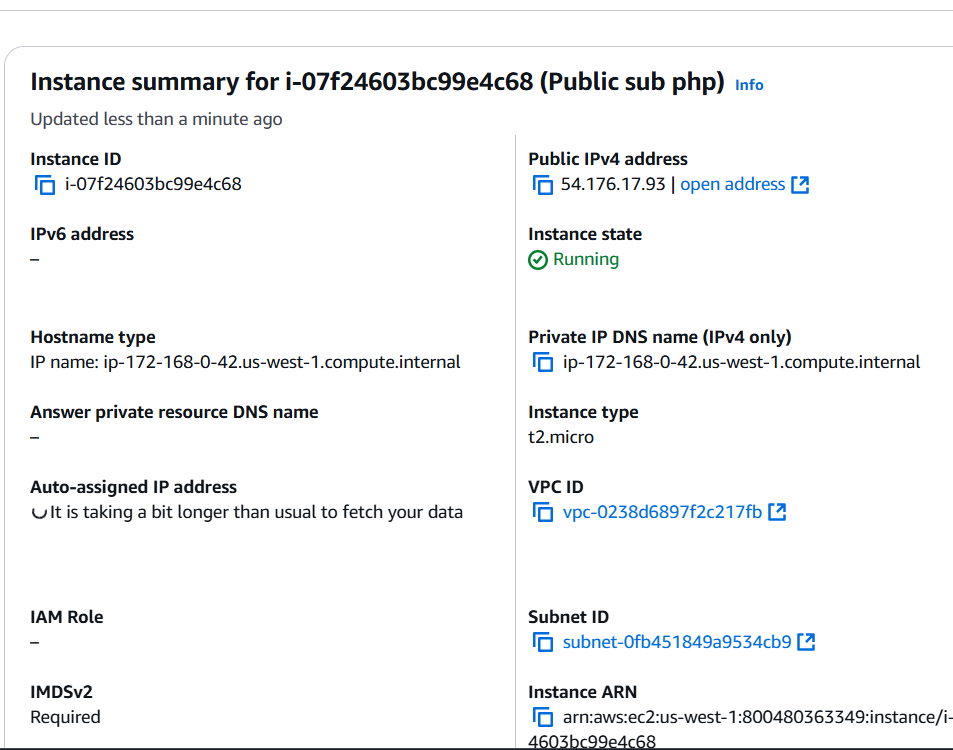
6. Configure security group:

- Create a new security group or select an existing one.

- Allow SSH (port 22) traffic.

7. Review and launch the instance.



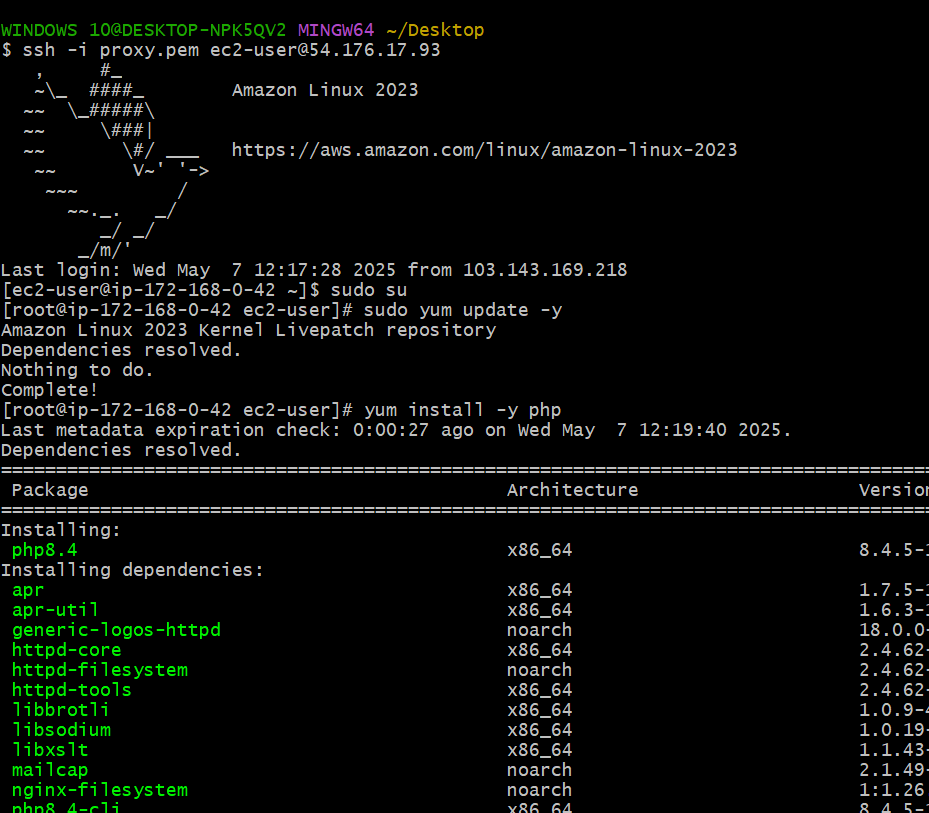


**Install PHP:**

1. Connect to your EC2 instance using SSH.

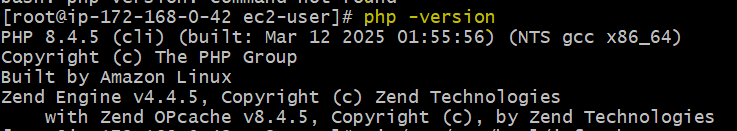
2. Update the package list: **yum update -y**

3. Install PHP: **yum install -y php** or **sudo amazon-linux-extras install -y php8** (for PHP 8.x)



**Verify PHP Installation:**

Check PHP version: php -v

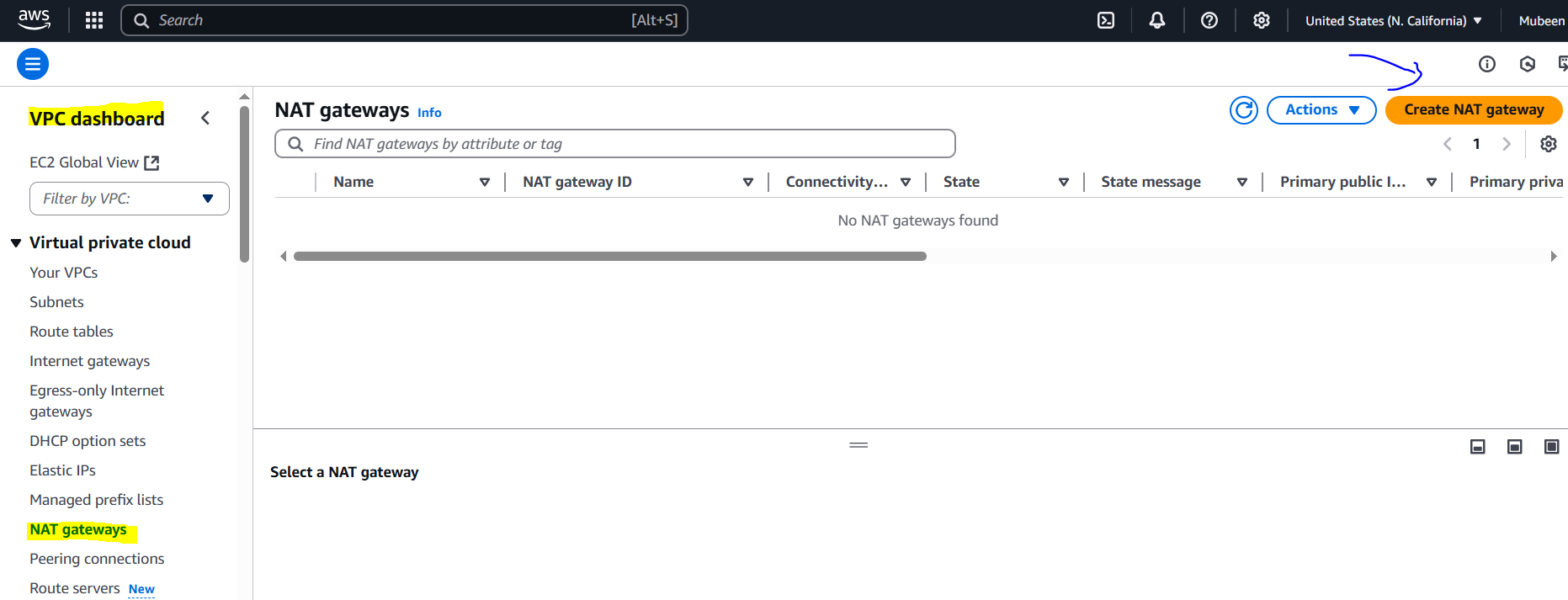


**8) Configure Nat gateway in public subnet and connect to private Instance**

**Step 1: Create a NAT Gateway**

1. Go to the VPC dashboard and navigate to the "NAT Gateways" section.

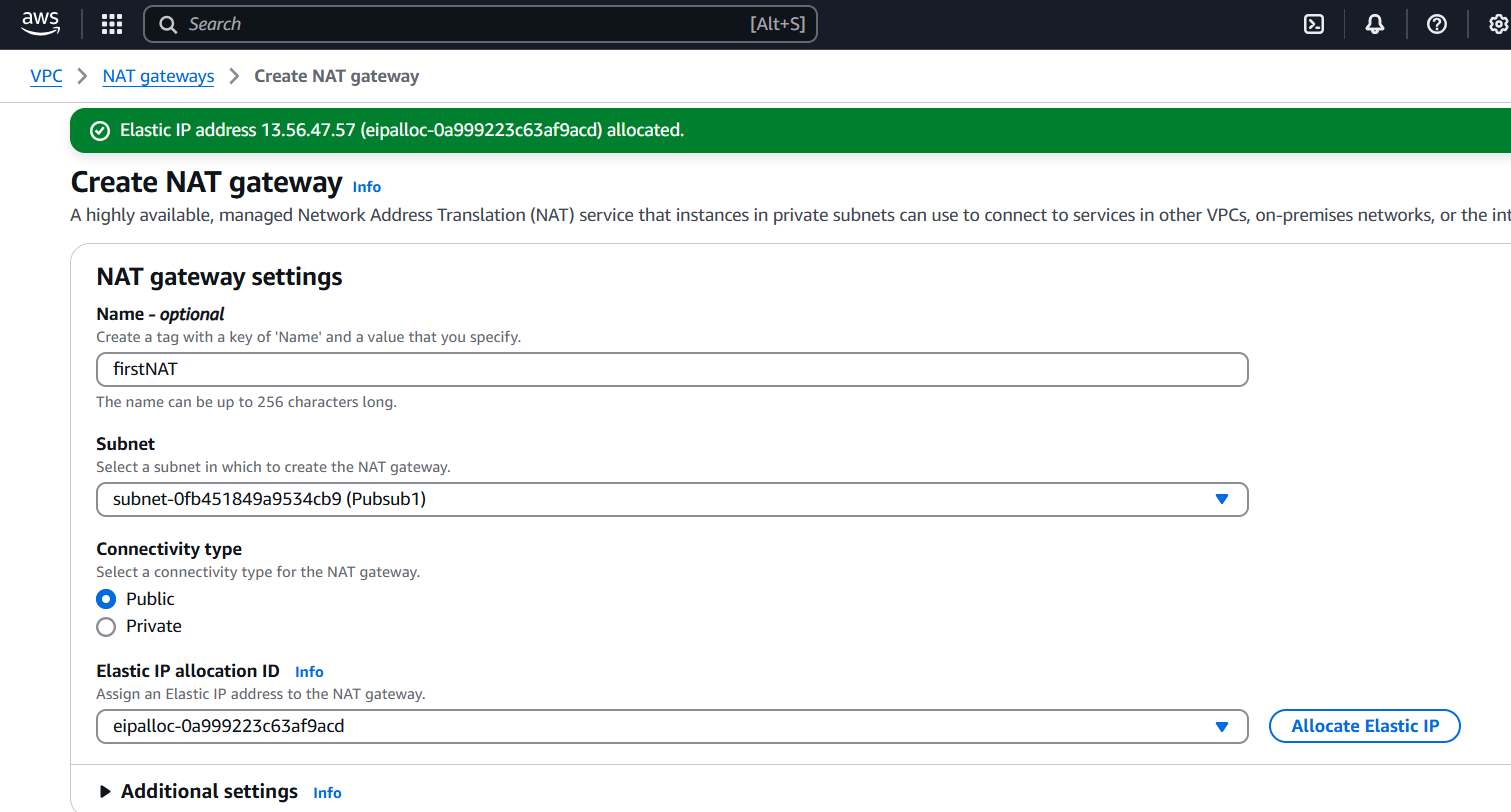
2. Click "Create NAT gateway".



3. Select the public subnet where you want to create the NAT gateway.

4. Choose an Elastic IP address (EIP) or create a new one.

5. Click "Create NAT gateway".

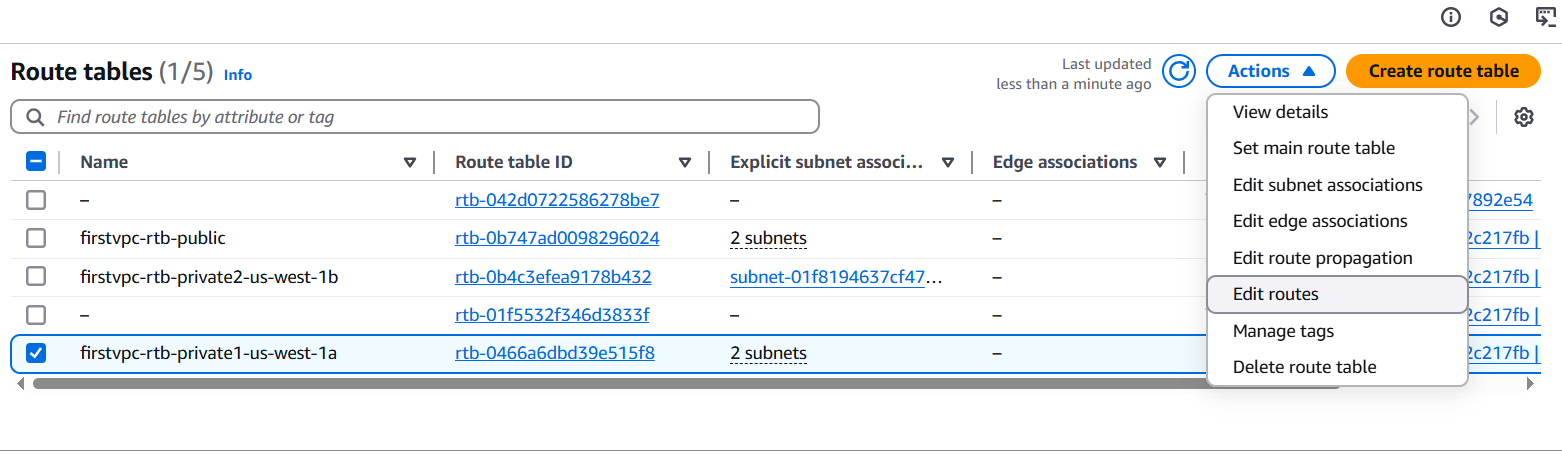


**Step 2: Update Route Table for Private Subnet**

1. Go to the VPC dashboard and navigate to the "Route Tables" section.

2. Select the route table associated with your private subnet.

3. Click "Actions" and then "Edit route table".

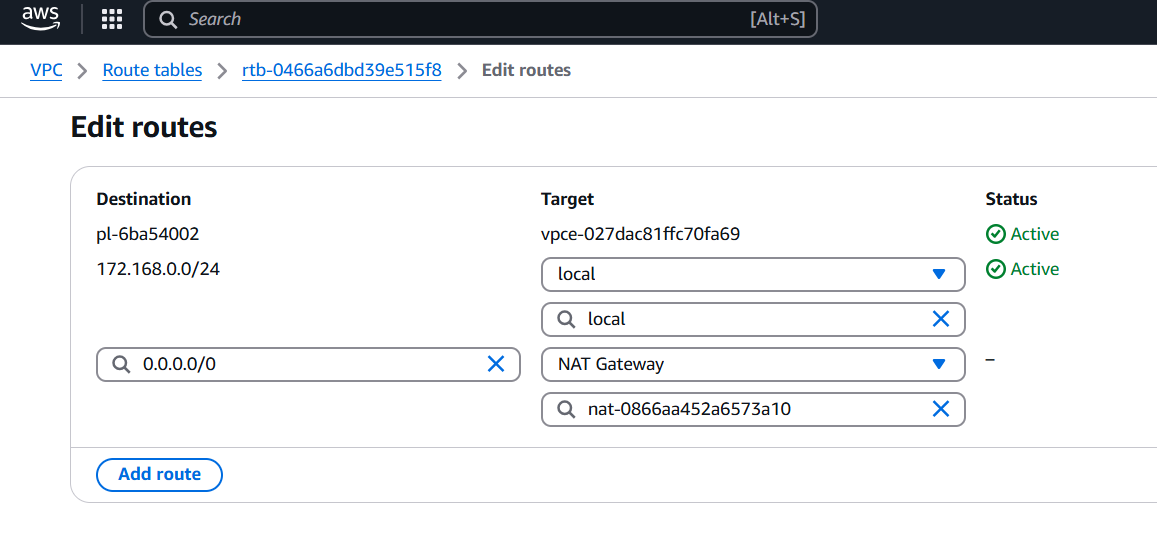


4. Add a new route:

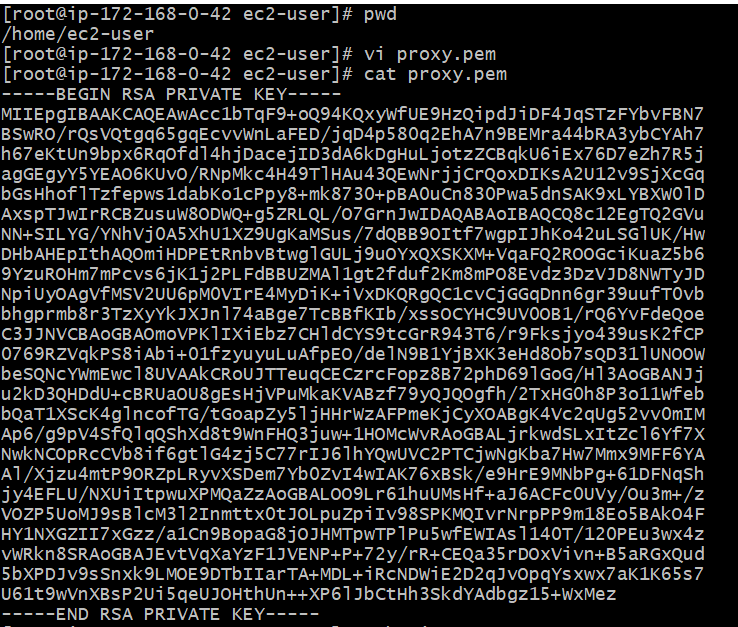
- Destination: 0.0.0.0/0

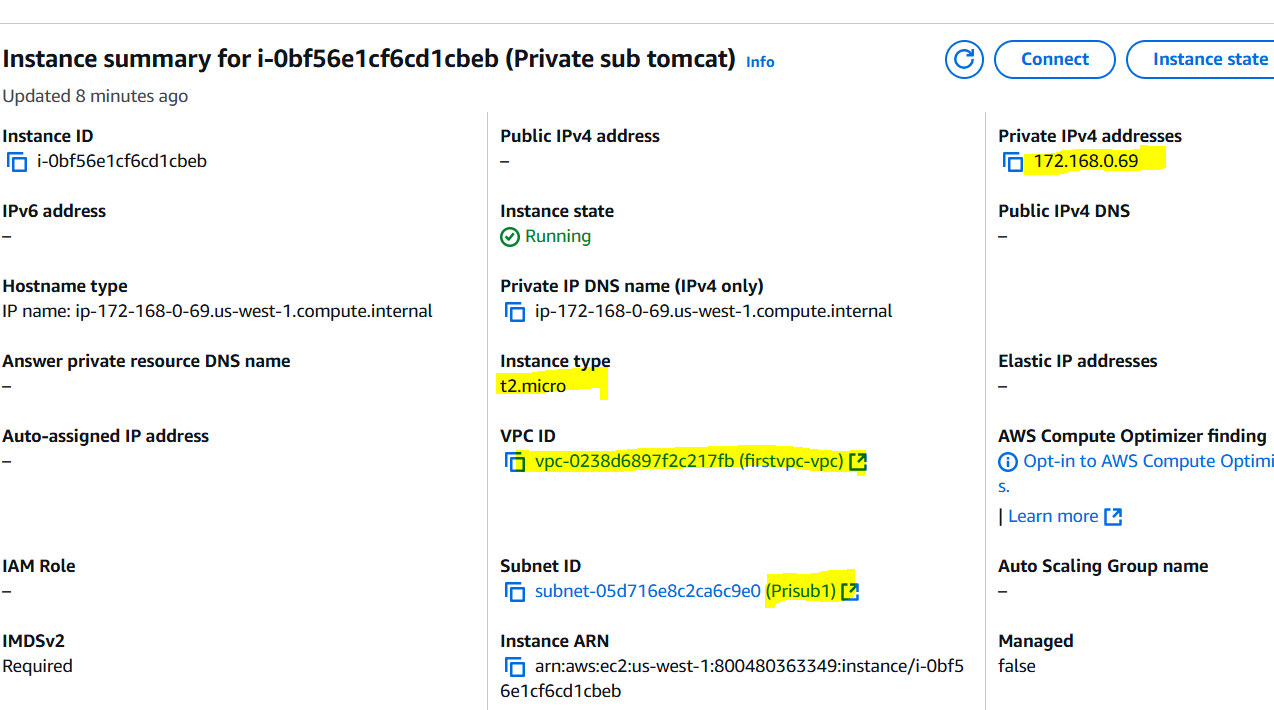
- Target: Select the NAT gateway you created earlier.

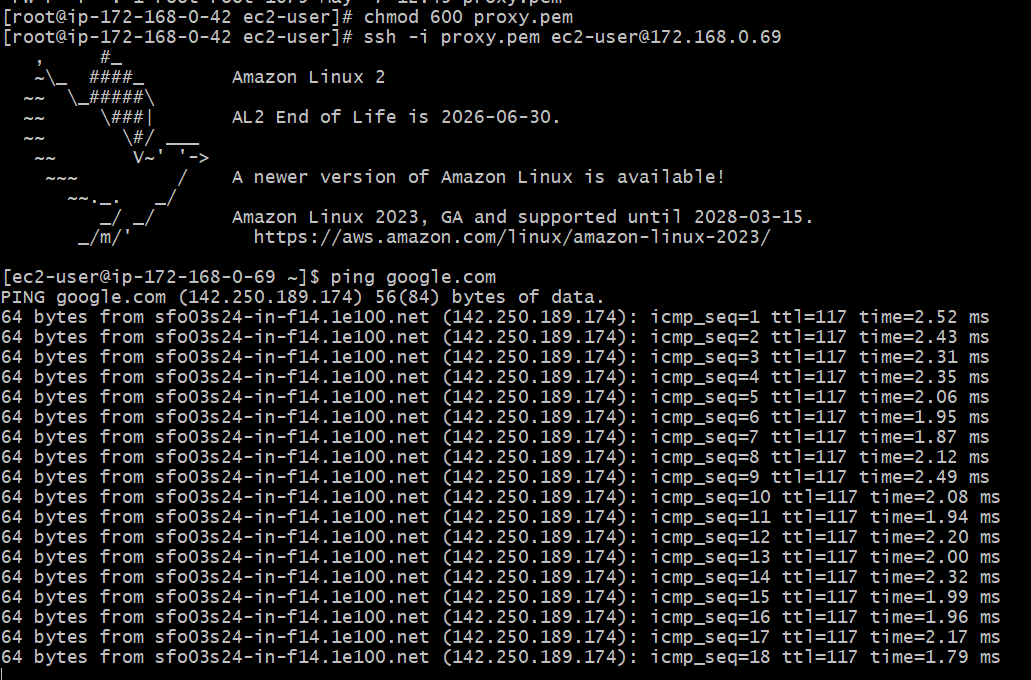
5. Click "Save changes".



The private server is connected to the internet through the public server's NAT gateway, allowing it to access external resources while maintaining its private IP address. This is confirmed by successfully pinging (google.com) from the private server. The NAT gateway translates the private IP to a public IP, enabling internet access. This setup ensures secure and controlled internet access for the private server.







**How it works:**

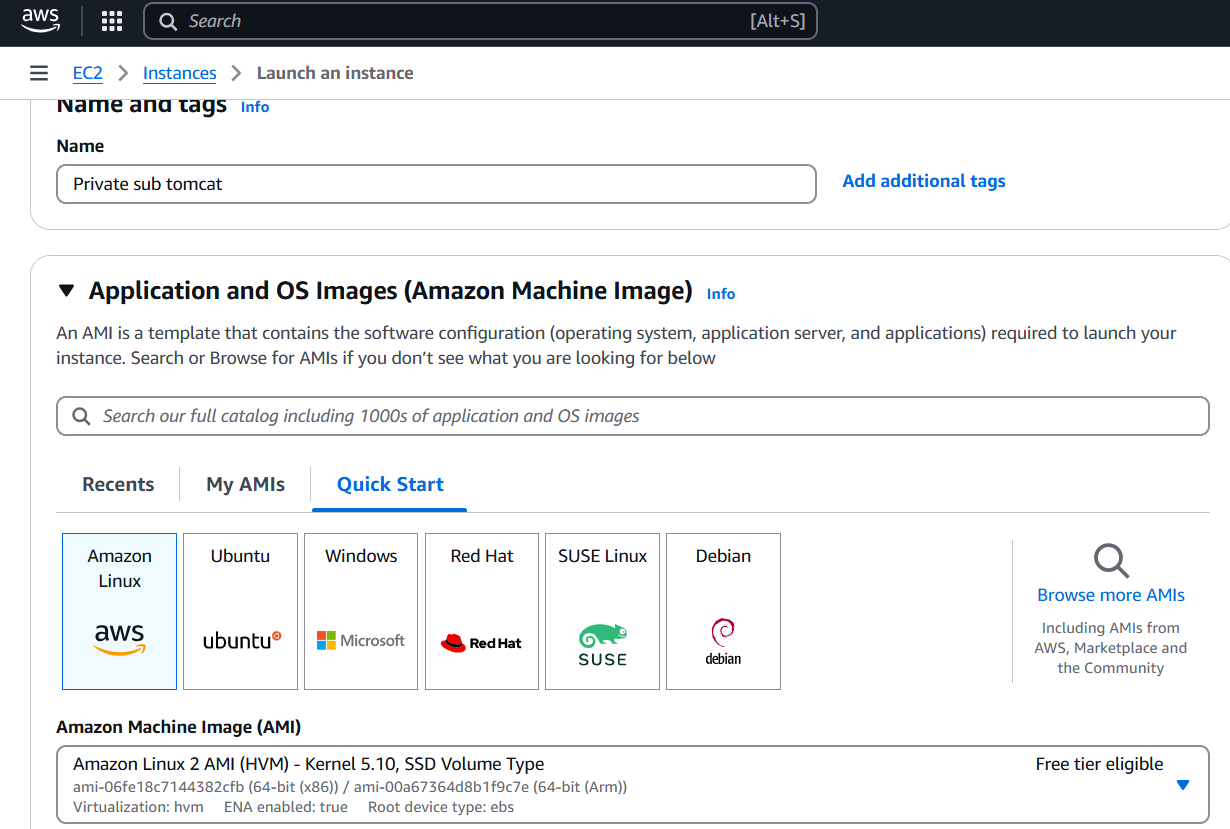
1. The NAT gateway translates the private IP address of the instance in the private subnet to the public IP address of the NAT gateway.

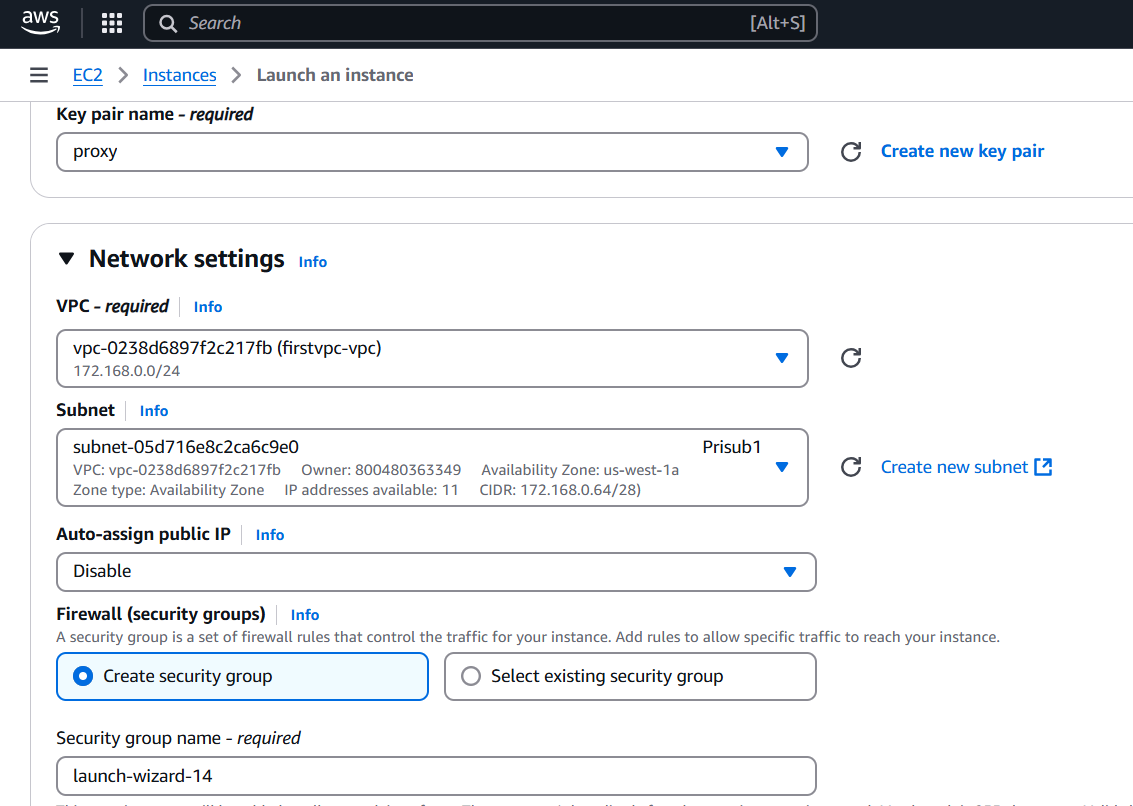
2. The NAT gateway then forwards the traffic to the internet.

3. When the response comes back, the NAT gateway translates the public IP address back to the private IP address of the instance.

**9) Install Apache Tomcat in private ec2 and deploy a sample app.**

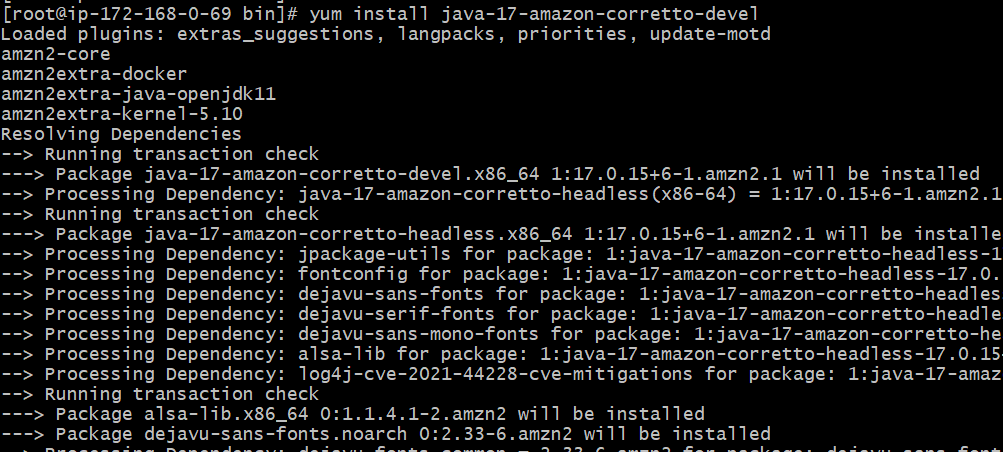
Private EC2 Launched.





**Step 1: Install Java 18**

**sudo amazon-linux-extras install java-openjdk11**

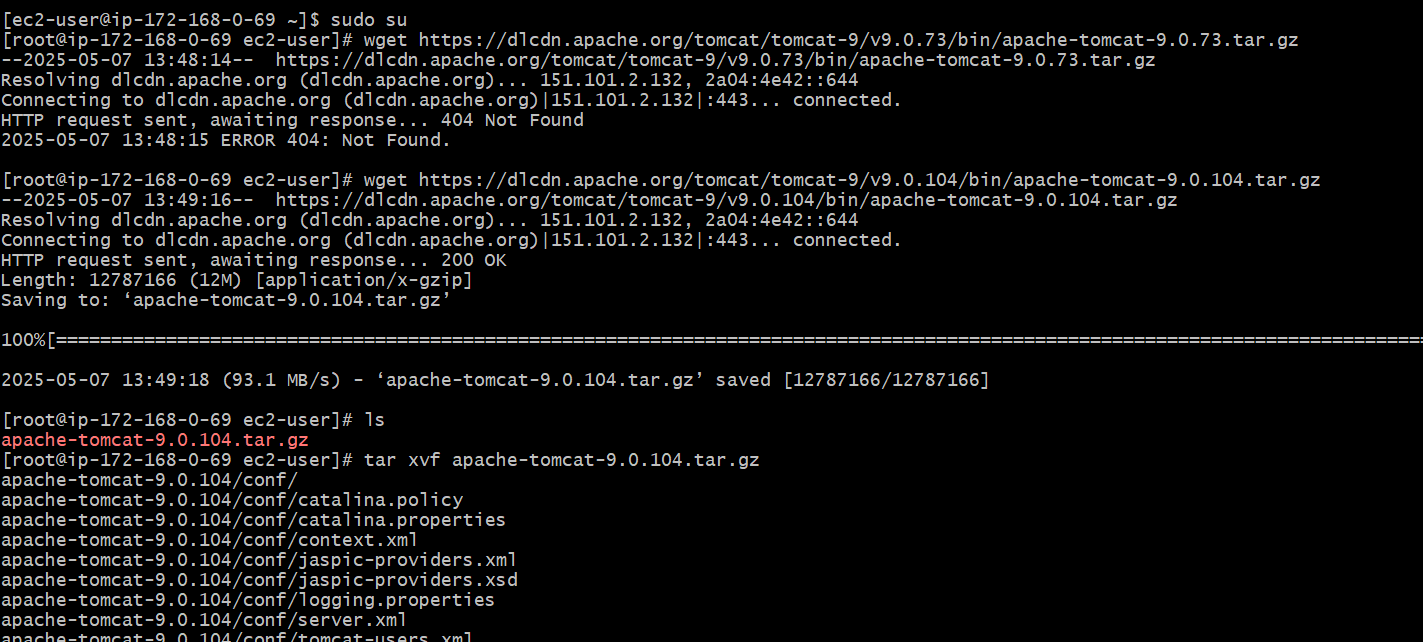


**Step 2: Install Tomcat 9**

Tomcat 9 might not be available directly via yum, download and install it manually:

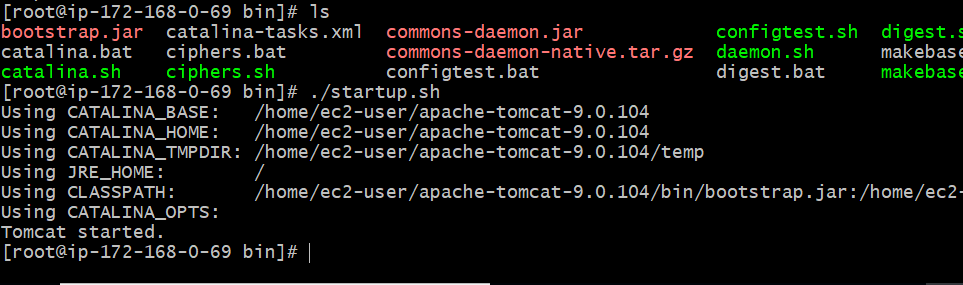
**wget https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.104/bin/apache-tomcat-9.0.104.tar.gz**

**tar -xvf apache-tomcat-9.0.104.tar.gz**



**Step 4: Start Tomcat**

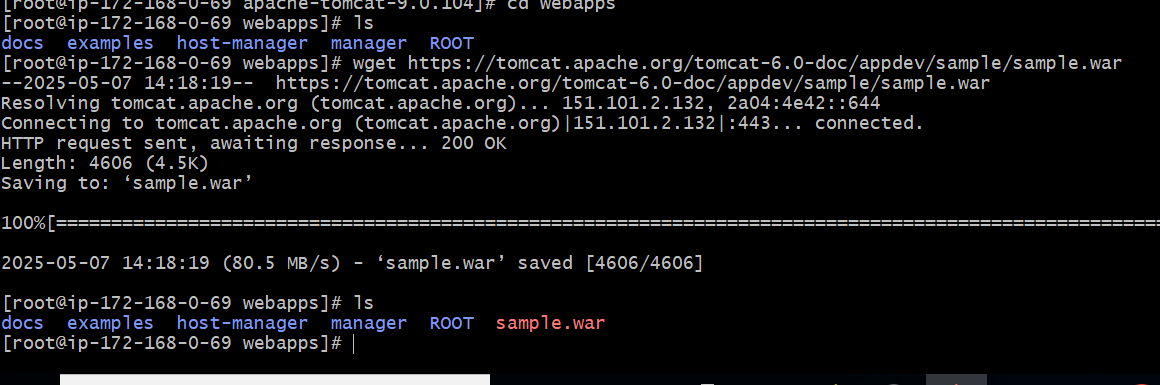
**./ startup.sh**



**Step 5: Deploy Sample App**

Let's assume you have a sample WAR file named sample.war. You can deploy it by copying it to the webapps directory:

**wget https://example.com/sample.war**



A NAT gateway allows **outbound traffic** but doesn't allow **inbound traffic**, so we are unable to open Tomcat sample in a browser from the internet even with a NAT gateway, however, we can achieve this using alternative methods such as a Load Balancer or Port Forwarding.

**10) Configure VPC flow logs and store the logs in s3 and cloudwatch.**

**Step 1: Create an S3 Bucket**

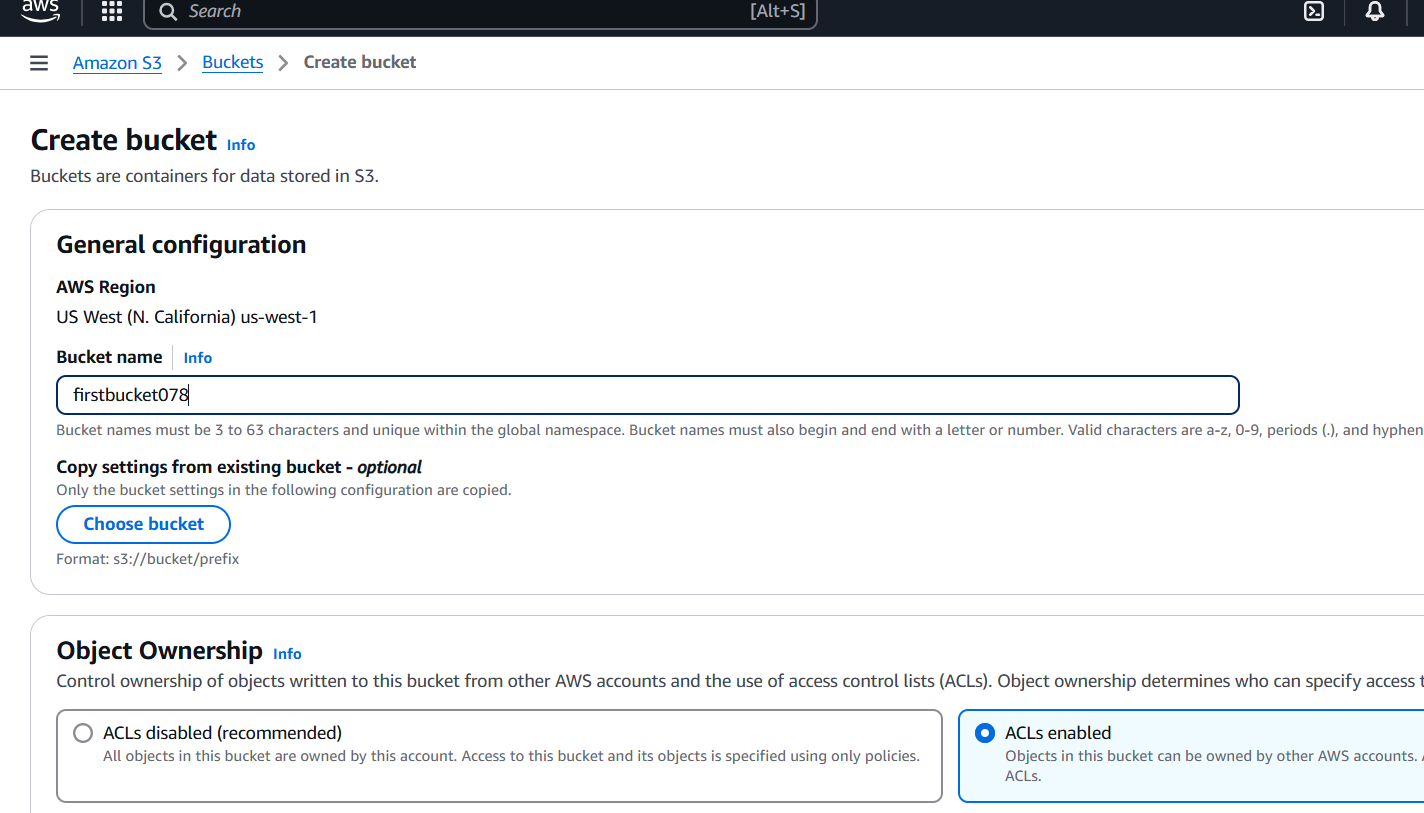
1. Go to S3 console

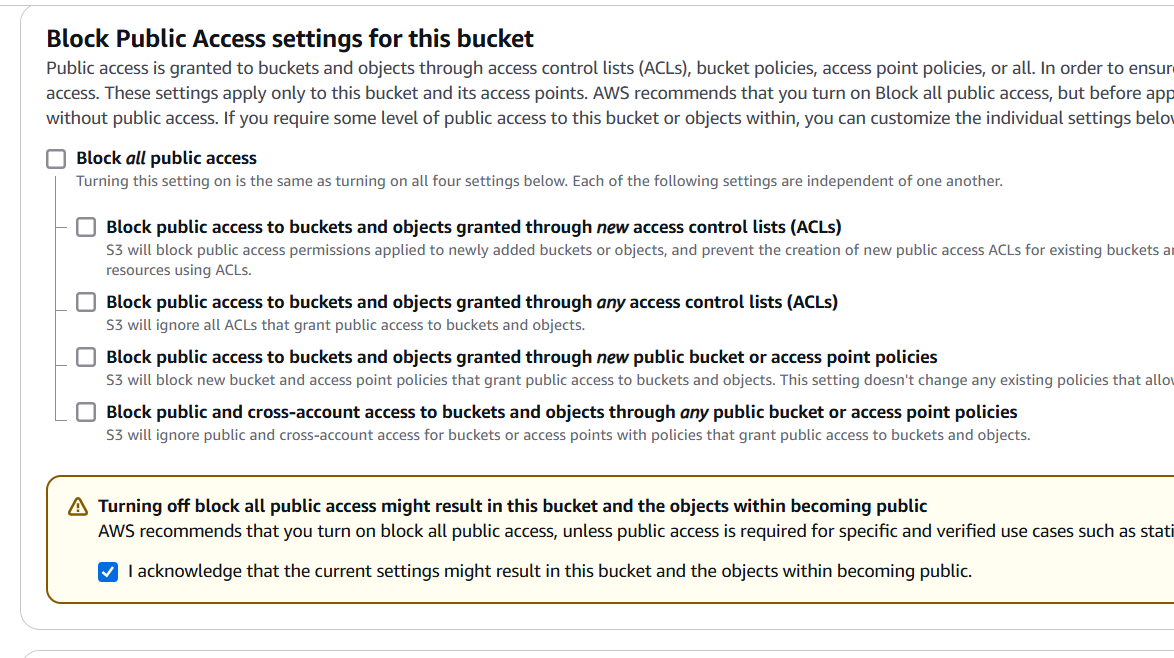
2. Click "Create bucket"

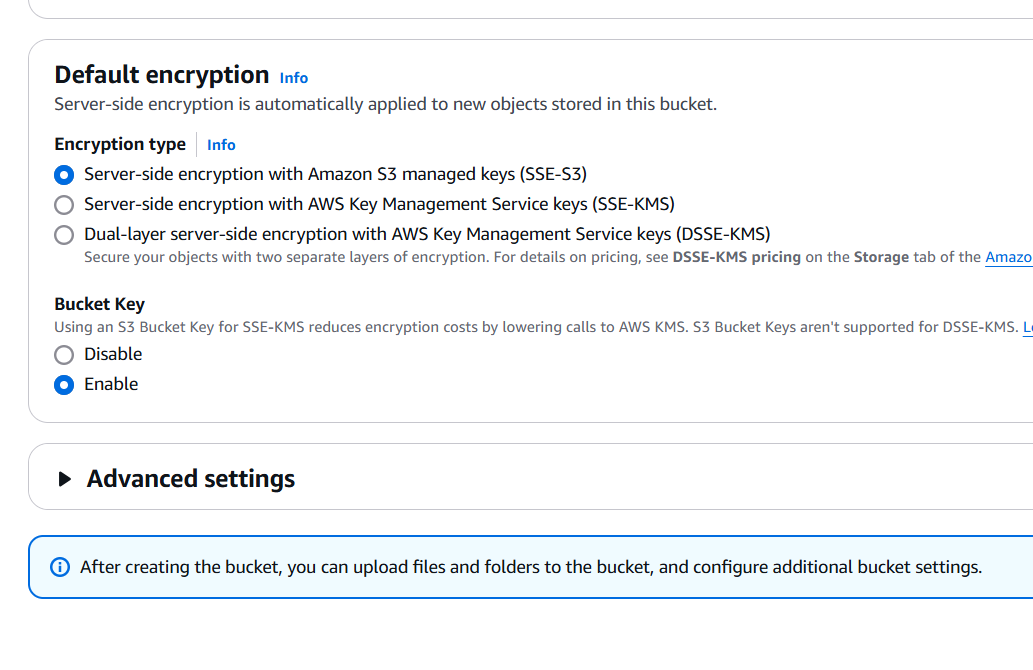
3. Enter bucket name

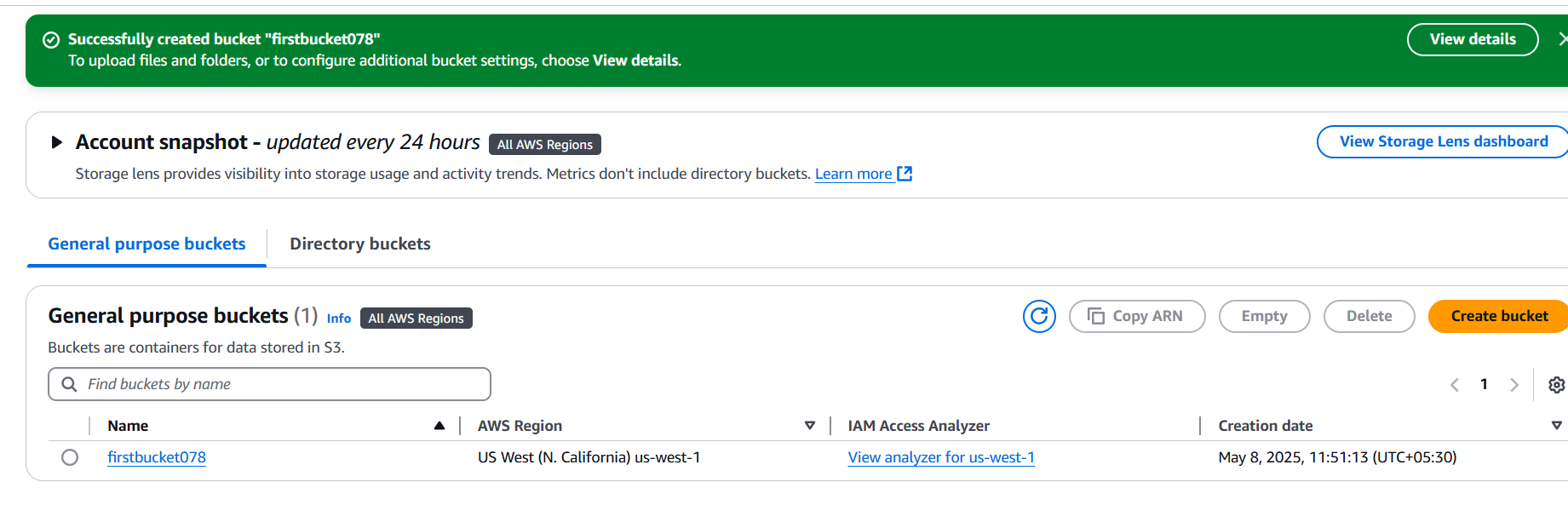
4. Select region

5. Click "Create bucket"









**Step 2: Create an IAM Role for VPC Flow Logs**

1. Go to IAM console

2. Click "Roles" > "Create role"

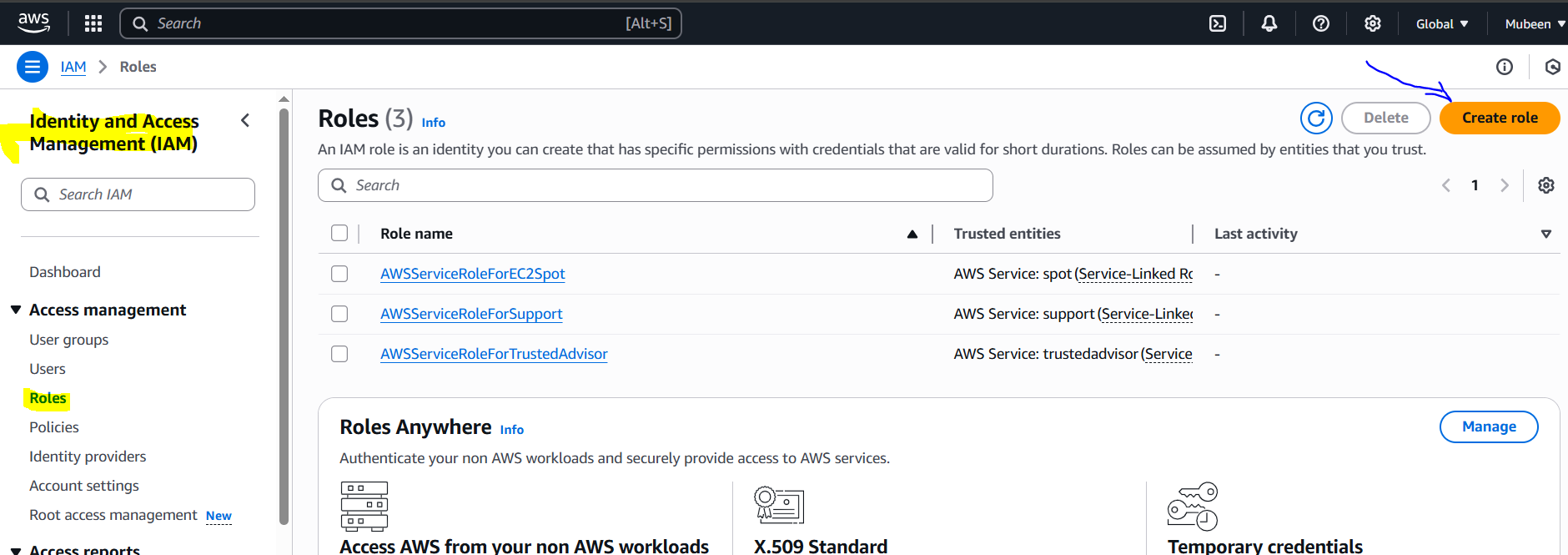
3. Choose "Custom role" and select "EC2" as the service

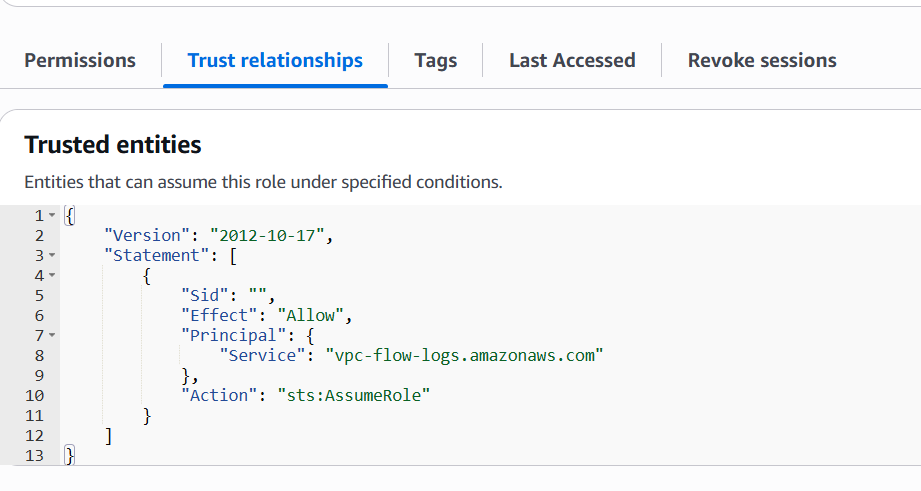
4. Attach policies:

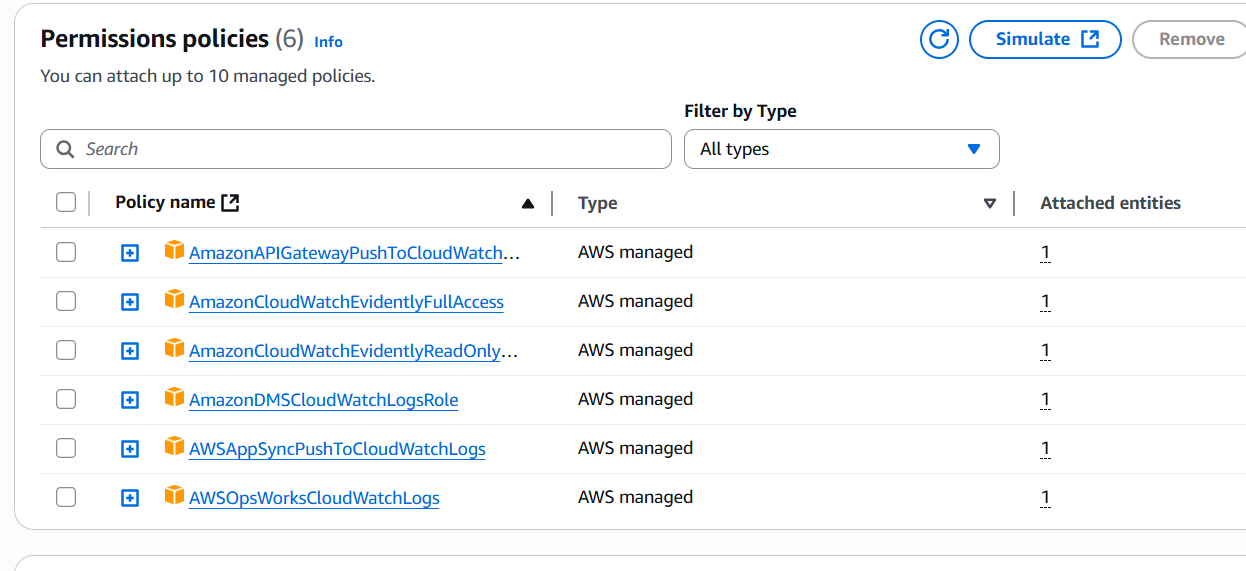
- "CloudWatchLogsReadOnlyAccess"

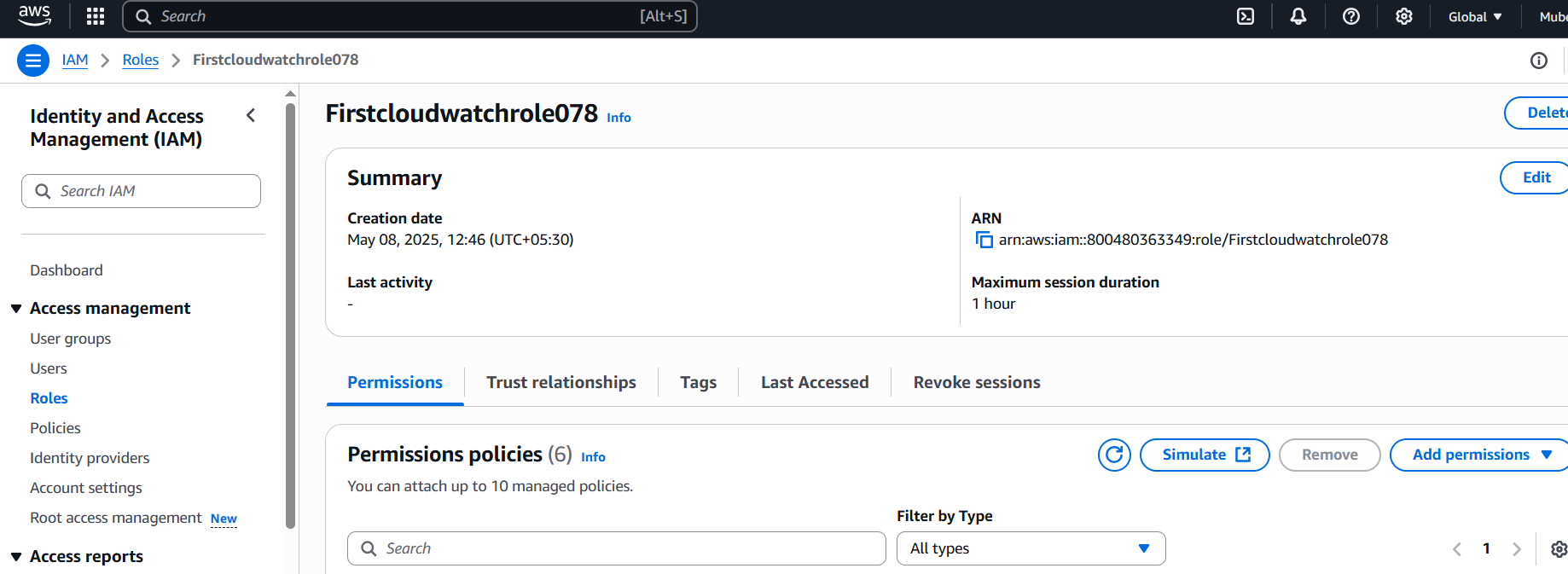
- "AmazonS3ReadOnlyAccess"

5. Name the role









**Step 3: Configure VPC Flow Logs**

1. Go to VPC console

2. Select VPC

3. Click "Actions" > "Create flow log"

4. Choose IAM role

5. Select destination:

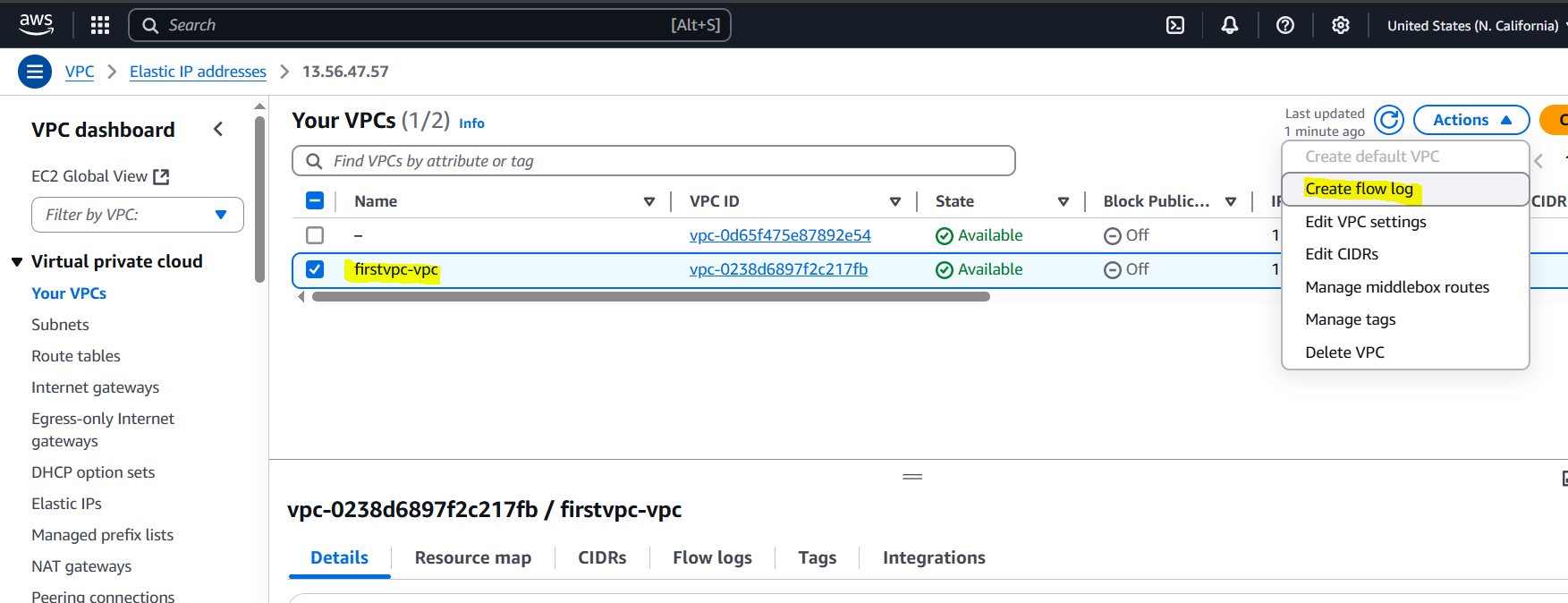
- S3: Enter bucket name

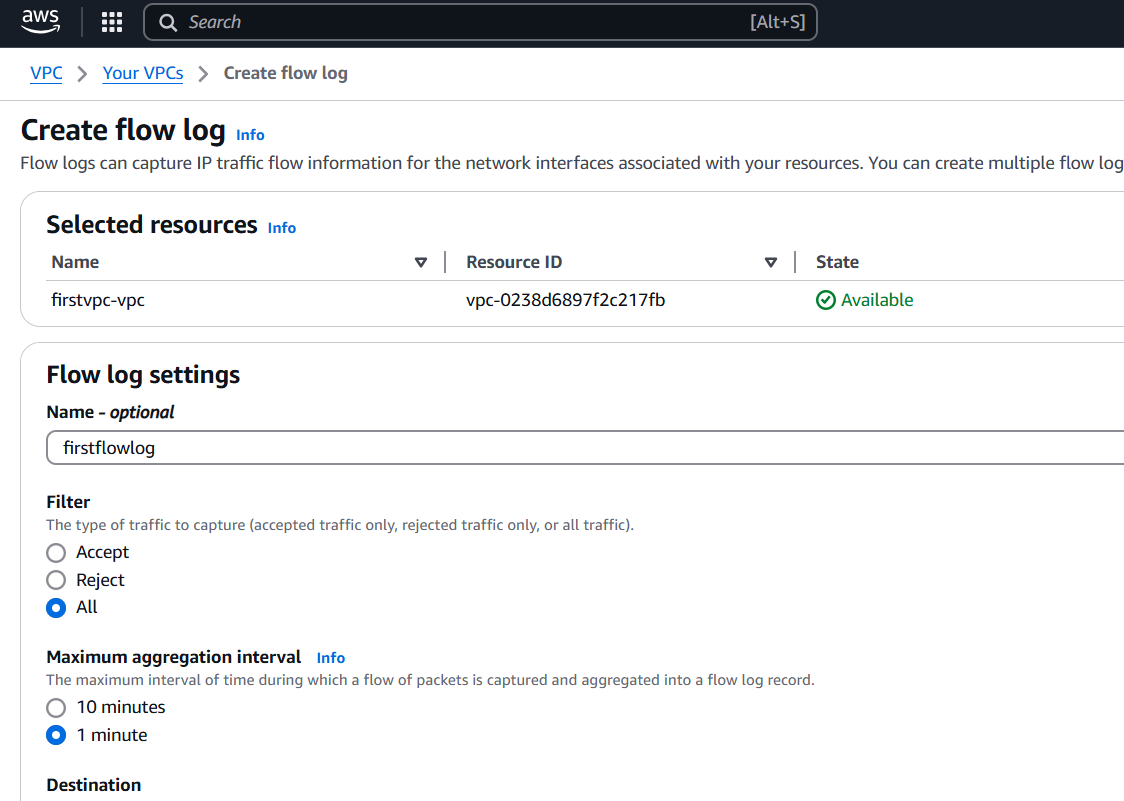
- CloudWatch Logs: Enter log group name

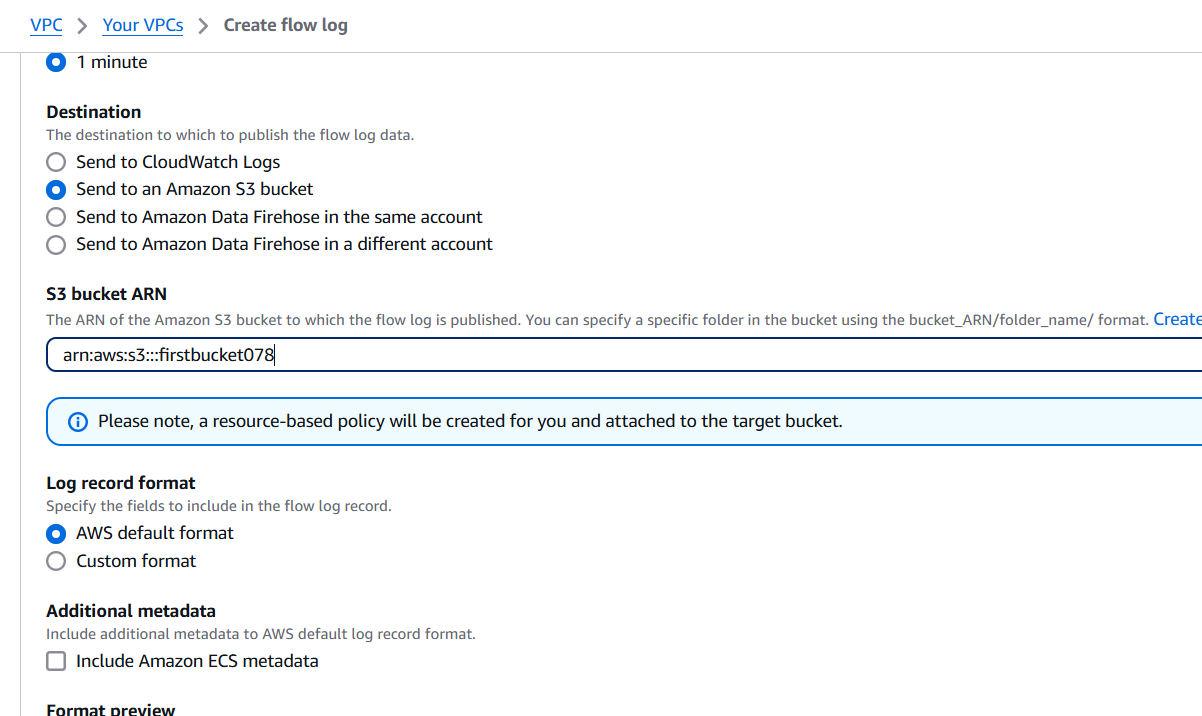
6. Choose log format

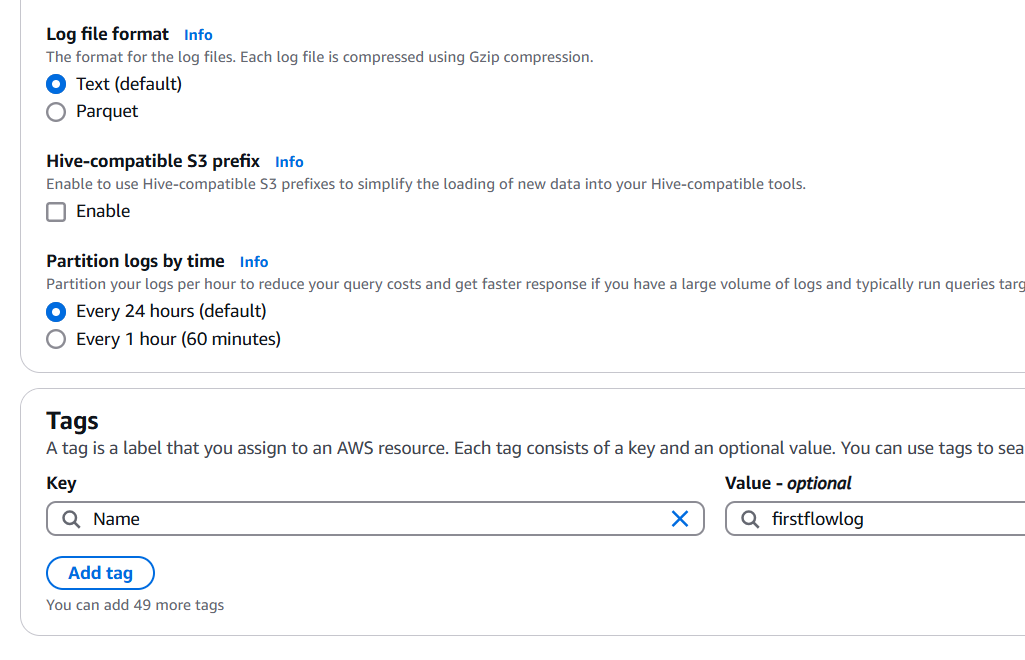
7. Click "Create flow log"

**Flow log for S3:**





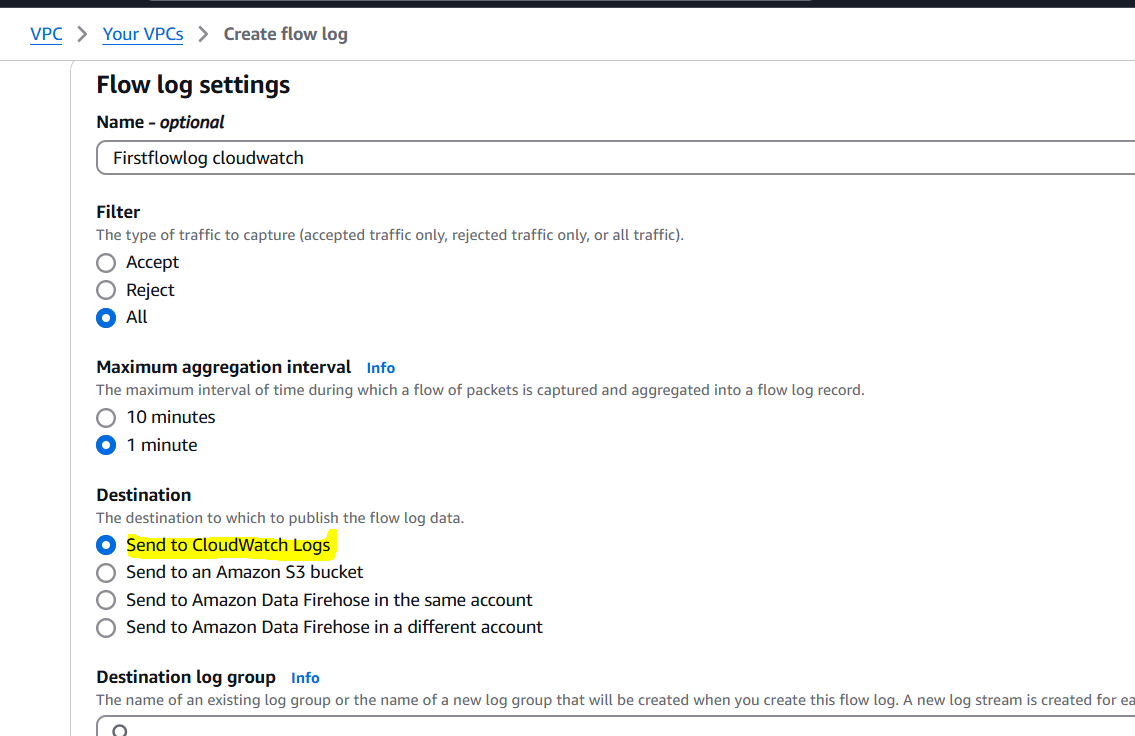


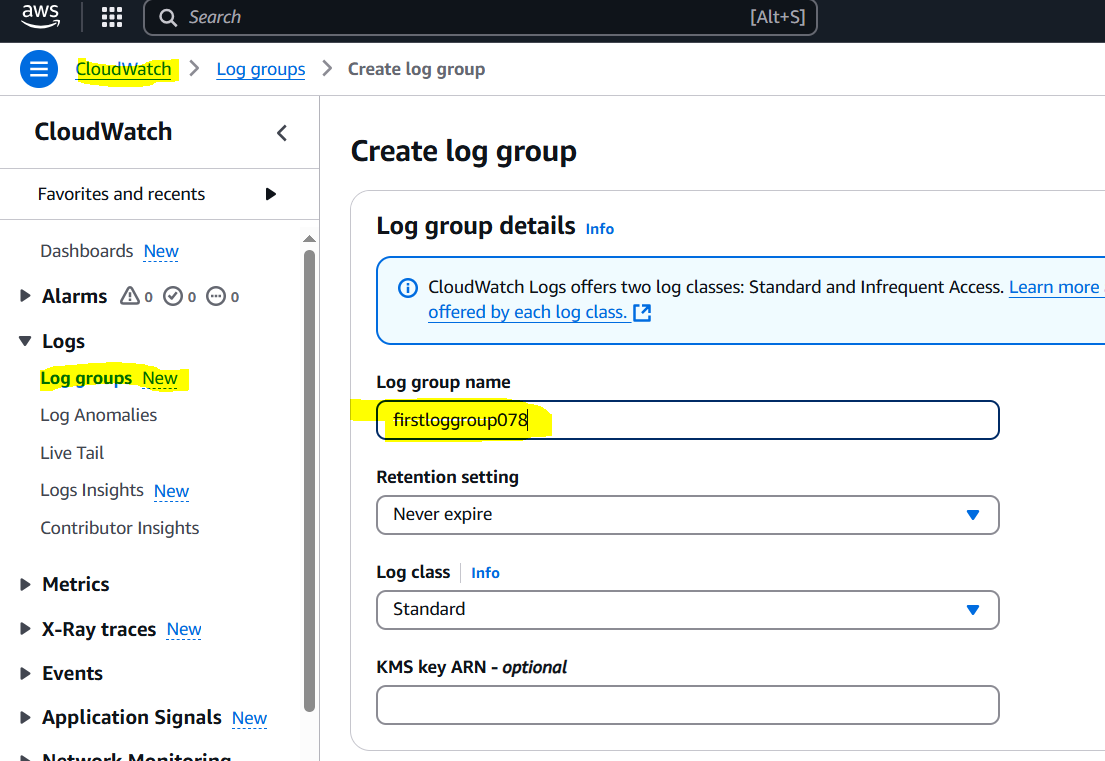


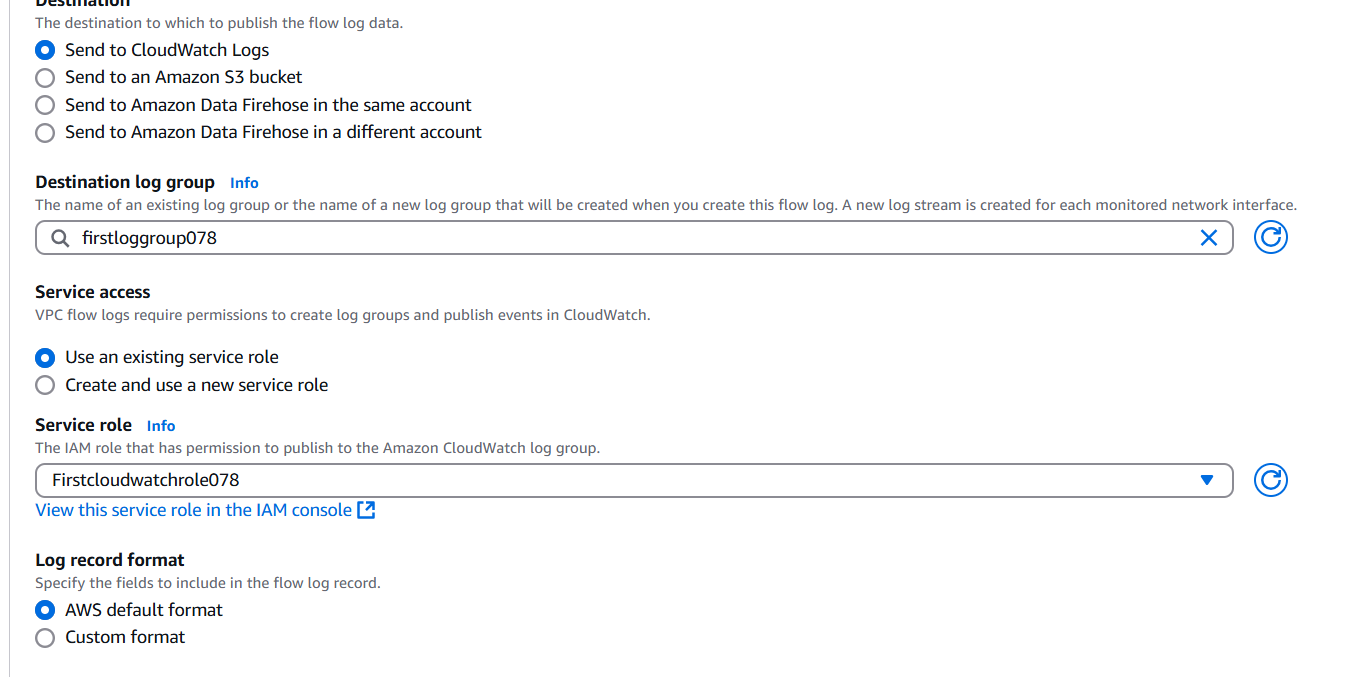
**Flow log for Cloudwatch:**

Same procedure as S3 bucket just select Cloudwatch at the place of S3 & Create log group.









**Checking Logs**

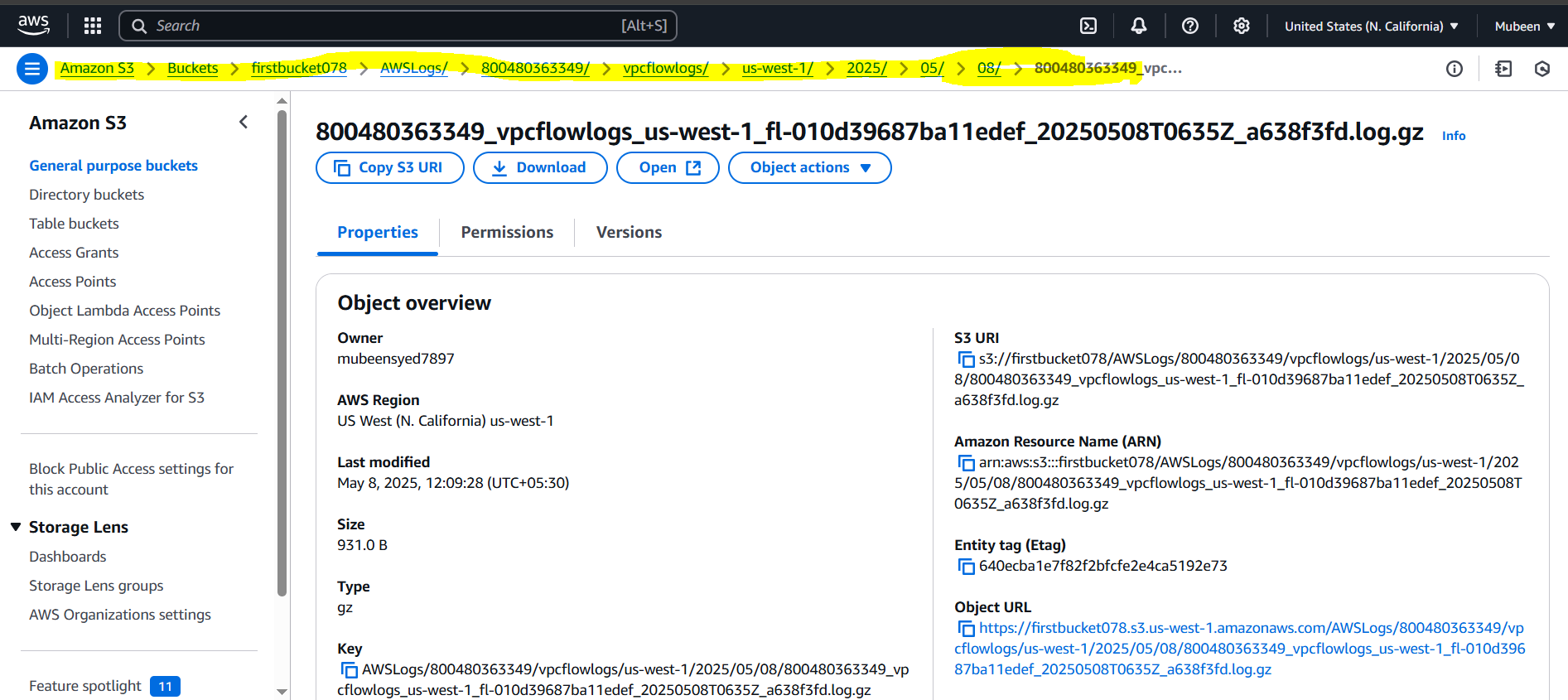
**S3:**

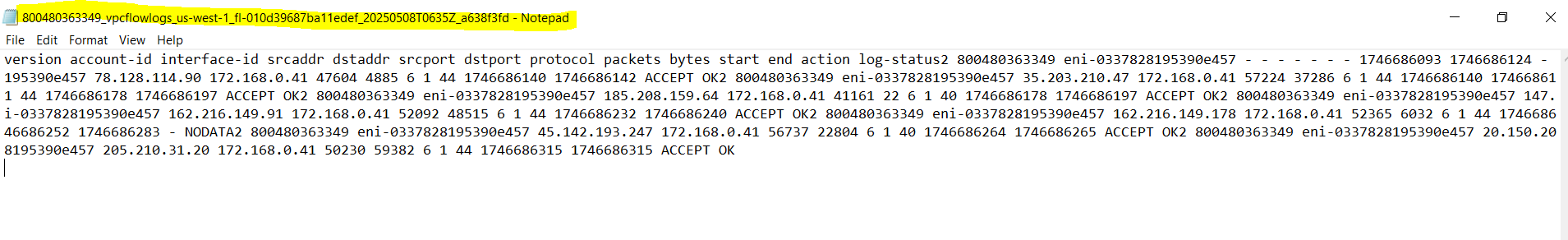
1. Go to S3 console

2. Select bucket

3. Find log files in the bucket

4. Download log files to analyse





**CloudWatch Logs:**

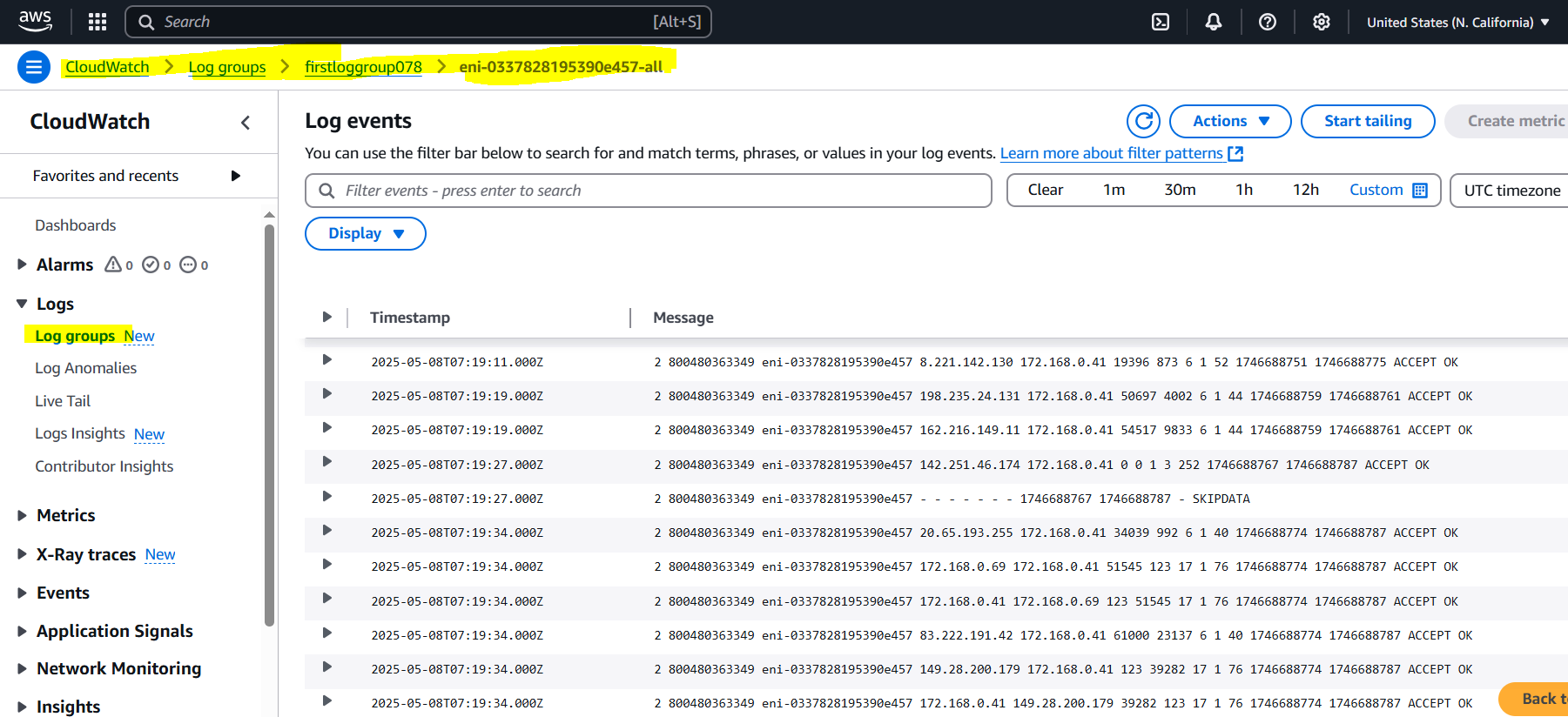
1. Go to CloudWatch console

2. Click "Logs" > "Log groups"

3. Find log group

4. Click on log group to view logs

5. Use log filters to filter logs



**Purpose:**

The purpose of configuring VPC Flow Logs is to capture and monitor network traffic in your VPC, allowing you to analyze and troubleshoot network issues, security threats, and performance optimization opportunities. By storing logs in S3 and CloudWatch Logs, you can retain and analyze log data for extended periods and set up alerts and notifications for real-time monitoring.

**The-End**