TASK 1

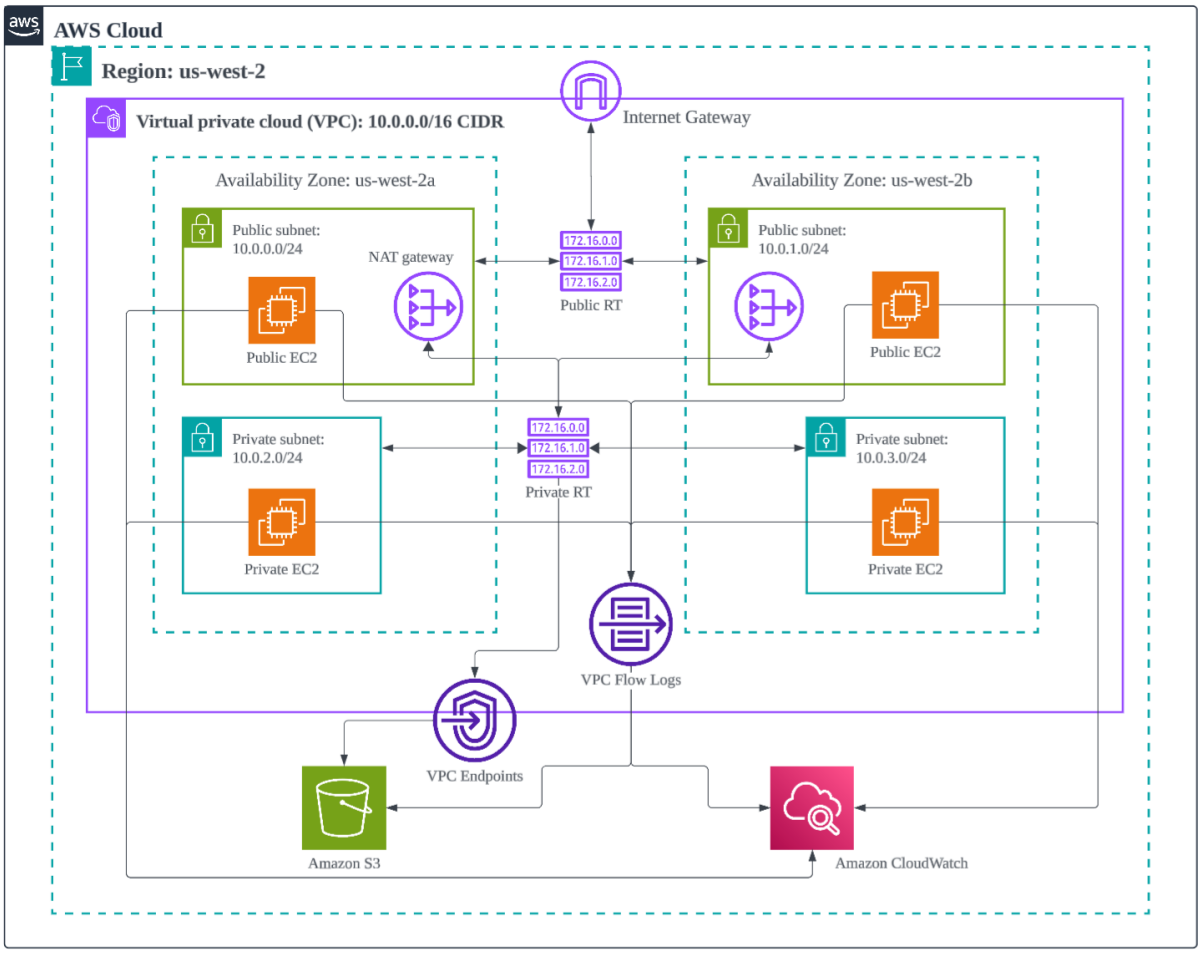
Create VPC Infrastructure

Umar Satti

Task Description

Custom Virtual Private Cloud (VPC) setup to host secure and scalable AWS resources across public and private subnets with proper routing, NAT, and security configurations.

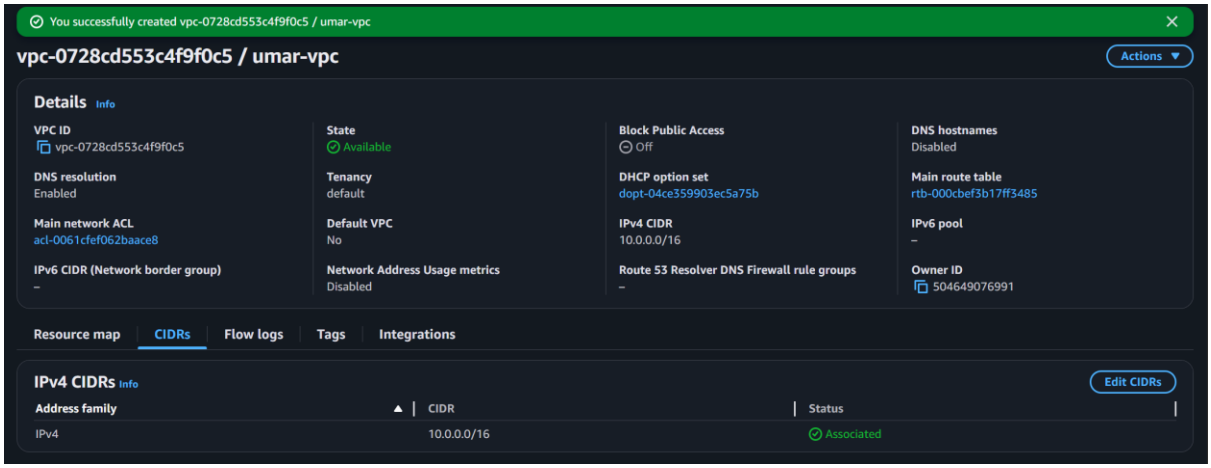
VPC Architecture Diagram



Task 1.1: Create a custom VPC with a specified CIDR block

Steps:

1. Navigate to the “VPC” service from the AWS Management Console search bar
2. In the left navigation panel, select “Your VPCs”
3. Click on Create VPC button located in the top right corner
4. Under Resources to create, select “VPC only”
5. Enter a Name and specify an IPv4 CIDR block range
6. Click “Create VPC” button

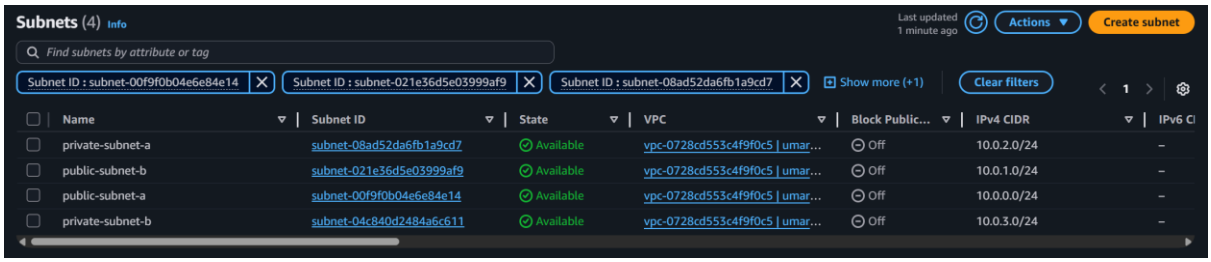


The VPC name is umar-vpc and the IPv4 CIDR block range is 10.0.0.0/16.

Task 1.2: Define public and private subnets across multiple Availability Zones

Steps:

1. Click on “Subnets” section in the left navigation panel
2. Click on “Create Subnet” button and select the VPC created in previous task
3. Define each subnet with a name, Availability Zone, and IPv4 CIDR block
4. Repeat the process to create both public and private subnets



The following subnets were created as illustrated above:

Public Subnets

- public-subnet-a, AZ is us-west-2a, and CIDR range is 10.0.0.0/24
- public-subnet-b, AZ is us-west-2b, and CIDR range 10.0.1.0/24

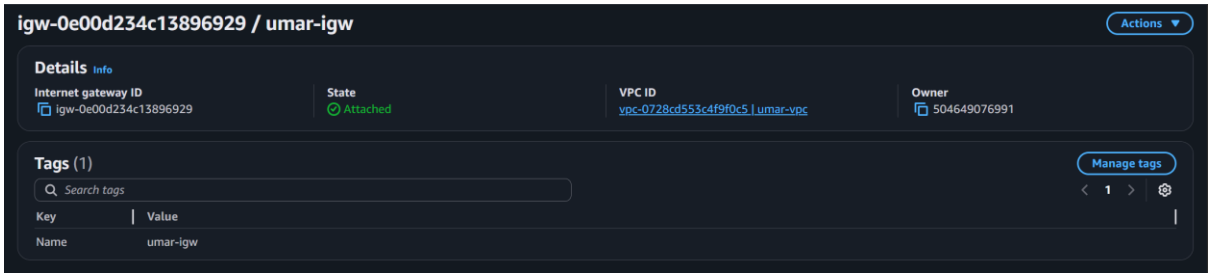
Private Subnets

- private-subnet-a, AZ is us-west-2a, and CIDR range is 10.0.2.0/24
- private-subnet-b, AZ is us-west-2a, and CIDR range is 10.0.3.0/24

Task 1.3: Set up an Internet Gateway and attach it to the VPC

Steps:

1. Click on “Internet gateways” section in the left navigation panel
2. Click “Create internet gateway” button on the top right
3. Enter a name tag and click Create Internet Gateway
4. After creating the internet gateway, attach it to the VPC created earlier
5. Click “Actions” button and select “Attach to VPC”. Choose the VPC ID and click “Attach internet gateway”



The Internet gateway name is umar-igw and it is attached to the VPC as shown by the status “Attached”

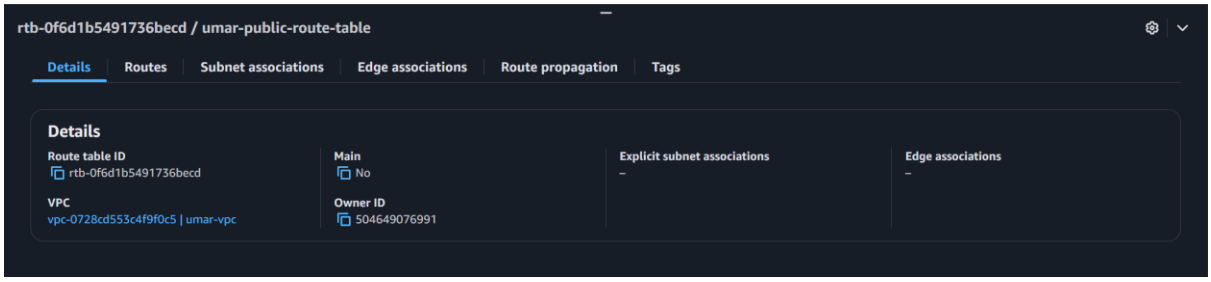
Task 1.4: Create route tables and associate them with appropriate subnets

Steps:

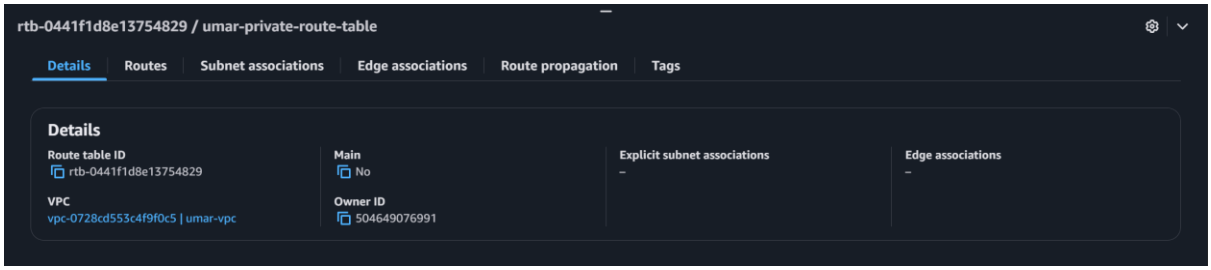
1. Click on “Route Tables” section in the left navigation panel
2. Click the “Create route table” button on the top right
3. Choose a name and select the VPC created in Task 1.1
4. Create one route table for public subnets and one for private subnets
5. Add the appropriate “Routes” and “Subnet associations” to these subnets

I created one route table for public subnet with the name “umar-public-route-table” and one route table for private subnet with the name “umar-private-route-table.”

Public Subnet Route Table:



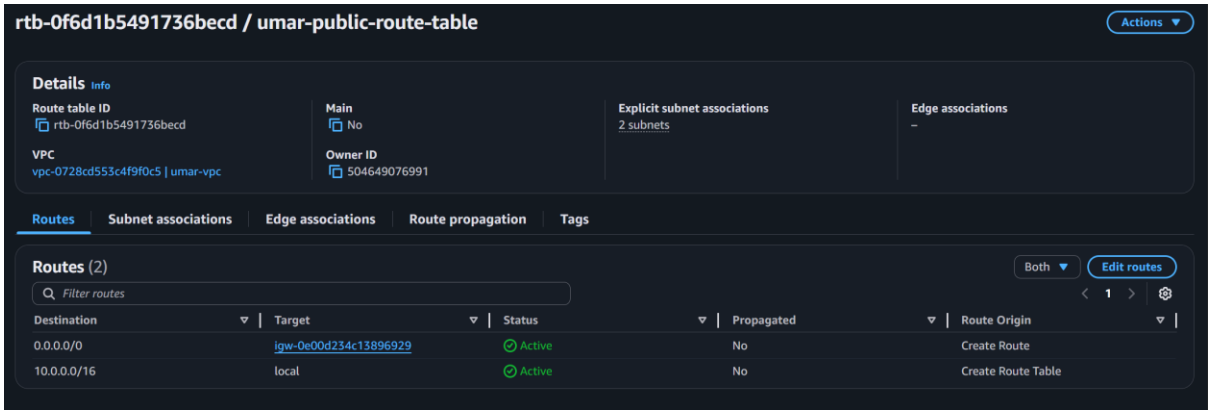
Private Subnet Route Table:



Configure the public route table to direct traffic to the internet gateway.

1. Select the public route table, click on “Routes” and then click “Edit routes”
2. Click “Add route”, select Destination and Target, and then click “Save changes”

I chose Destination as 0.0.0.0/0 and the Target as Internet Gateway (umar-igw)

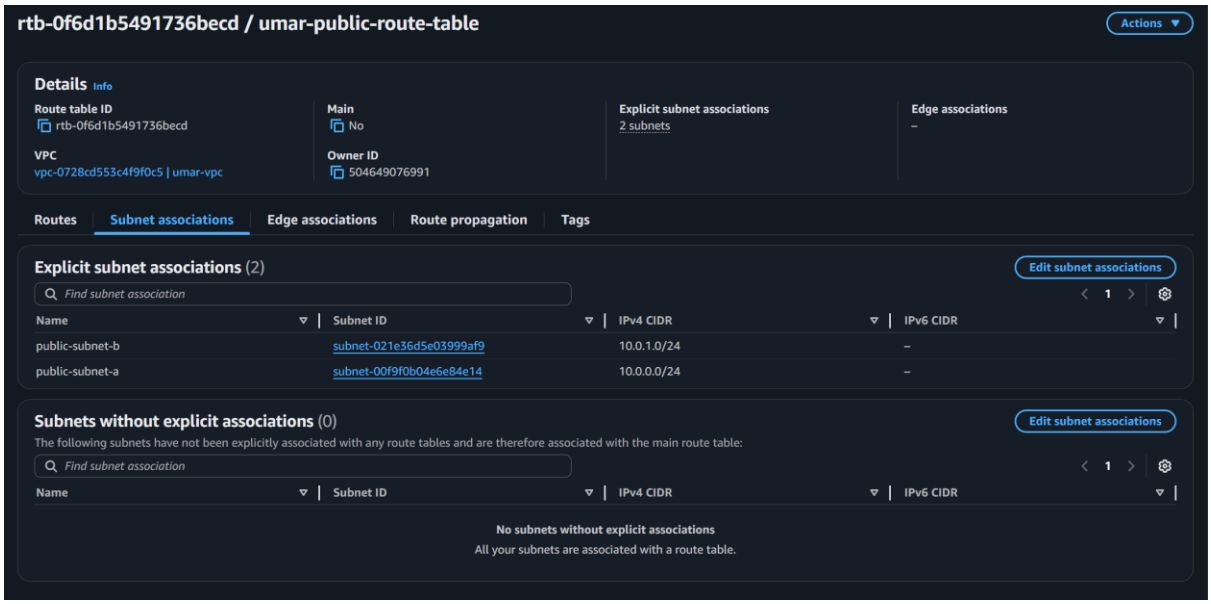


Configure the route tables to add subnet associations

1. Select the route table.
2. Click on “Subnet associations”
3. Click the “Edit subnet associations” button and choose the appropriate subnets

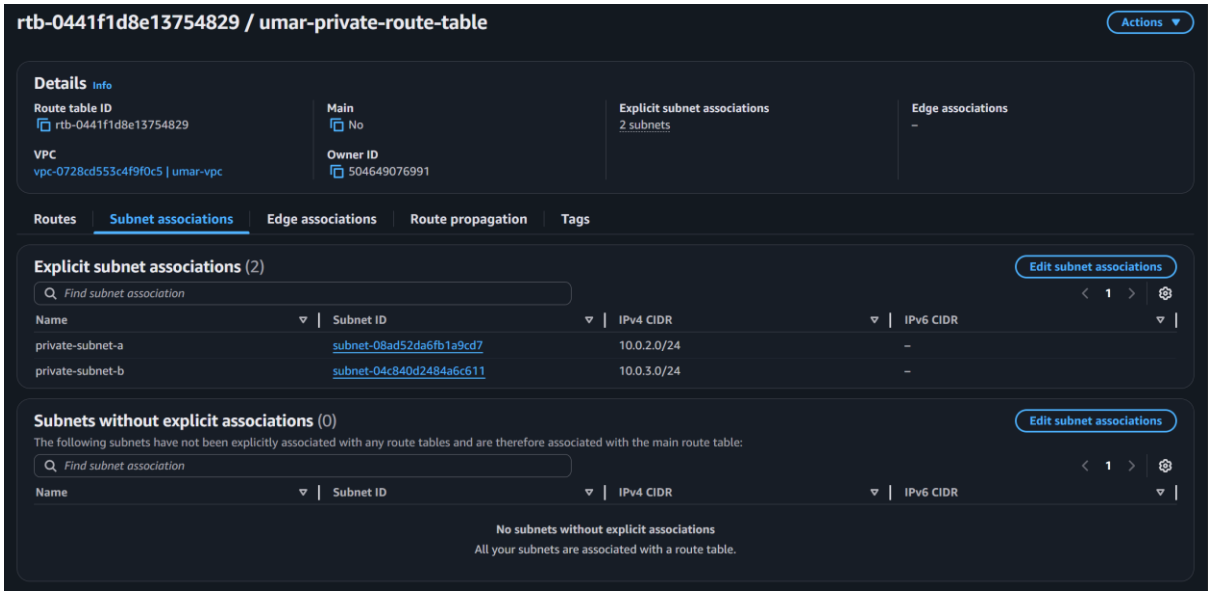
Public Route Table Subnet Association:

- public-subnet-a with the IPv4 CIDR range 10.0.0.0/24
- public-subnet-b with the IPv4 CIDR range 10.0.1.0/24



Private Route Table Subnet Association:

- private-subnet-a with the IPv4 CIDR range 10.0.2.0/24
- private-subnet-b with the IPv4 CIDR range 10.0.3.0/24



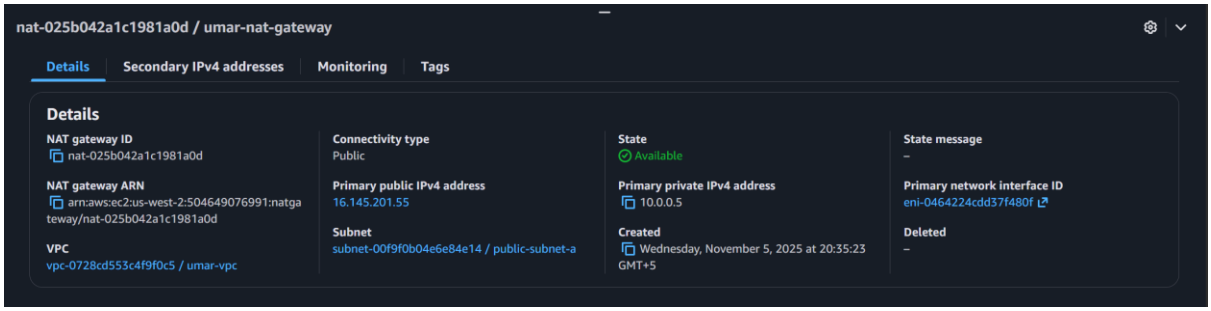
Task 1.5: Configure NAT Gateway in a public subnet for private subnet internet access

Steps:

1. Click on “NAT Gateways” section in the left navigation panel
2. Click “Create NAT gateway” button on the top right
3. Provide a name, select a public subnet, and allocate an Elastic IP
4. Choose Connectivity Type as “Public”
5. Click “Create NAT Gateway”

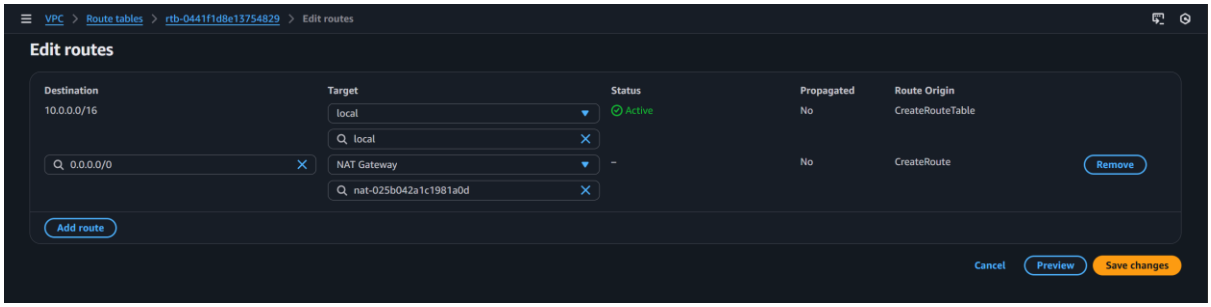
I configured a NAT Gateway in a public subnet to allow instances in private subnets to access the internet

- NAT Gateway Name is “umar-nat-gateway”, Subnet is **“**public-subnet-a”, Elastic IP is “eipalloc-0d07b3272e0c6b5a2”
- Destination as 0.0.0.0/0 to Target as “umar-nat-gateway”

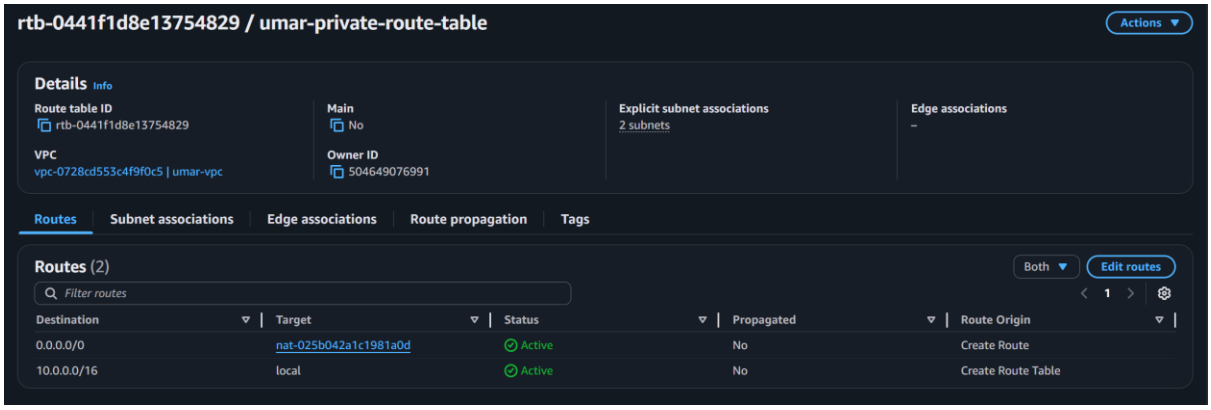


Configure the private route table to direct traffic to NAT gateway.

1. Select the private route table, click on “Routes” and then click “Edit routes”
2. Click “Add route”, select Destination and Target, and then click “Save changes”



I chose Destination as 0.0.0.0/0 and Target as NAT Gateway (umar-nat-gateway) as shown below:



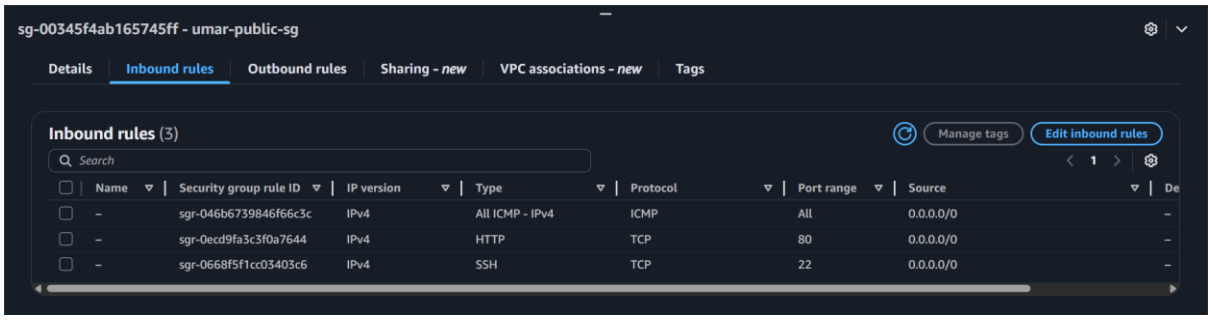
Task 1.6: Implement security groups and network ACLs for traffic control

Steps:

1. Navigate to the “EC2” service from the AWS Management Console search bar
2. In the left navigation panel, select “Security Groups”
3. Click the “Create security group” button on the top right
4. Choose a security group name, description, and VPC
5. Add Inbound and Outbound rules

For public resources, I chose the following security group configuration:

- Security group name: umar-public-sg
- VPC: umar-vpc
- Inbound Rules:
    - All ICMP - IPv4 from Anywhere-IPv4 (0.0.0.0/0)
    - HTTP from Anywhere-IPv4 (0.0.0.0/0)
    - SSH from Anywhere-IPv4 (0.0.0.0/0)
- Outbound Rules:
    - All Traffic to Destination 0.0.0.0/0



For private resources, I chose the following security group configuration:

- Security group name: umar-private-sg
- VPC: umar-vpc
- Inbound Rules:
    - All ICMP - IPv4 traffic from umar-public-sg
    - HTTP traffic from umar-public-sg
    - SSH traffic from umar-public-sg
- Outbound Rules:
    - All Traffic to Destination 0.0.0.0/0

Task 1.7: Launch EC2 instances in private and public subnets as needed

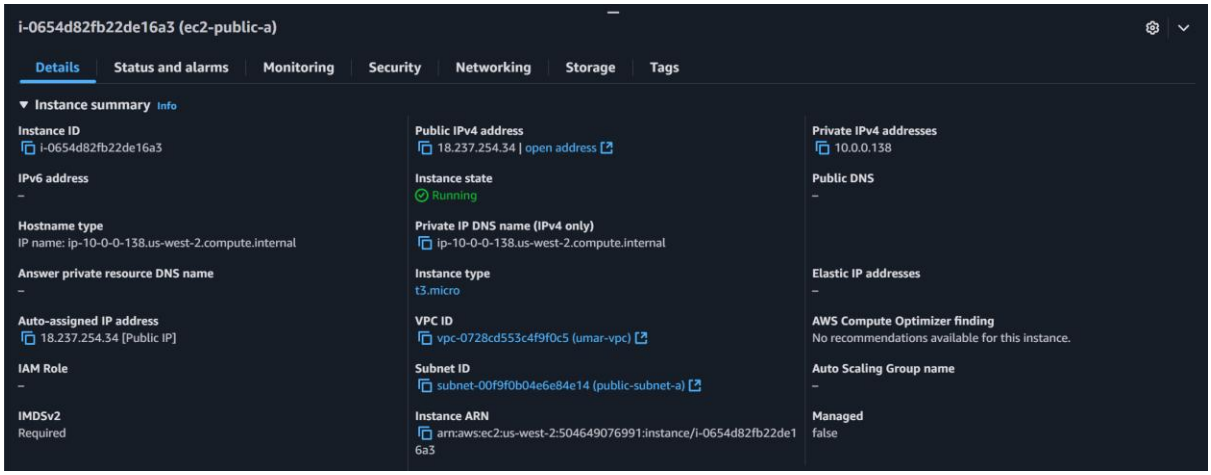
Steps:

1. Navigate to the “EC2” service from the AWS Management Console search bar
2. In the left navigation panel, select “Instances”
3. Click the “Launch instances” button on the top right
4. Choose a name, OS image, AMI, Instance type, Key pair, Network Settings, and Storage
5. Click “Launch Instance” button

I launched two EC2 instances, one in a public subnet and another in a private subnet.

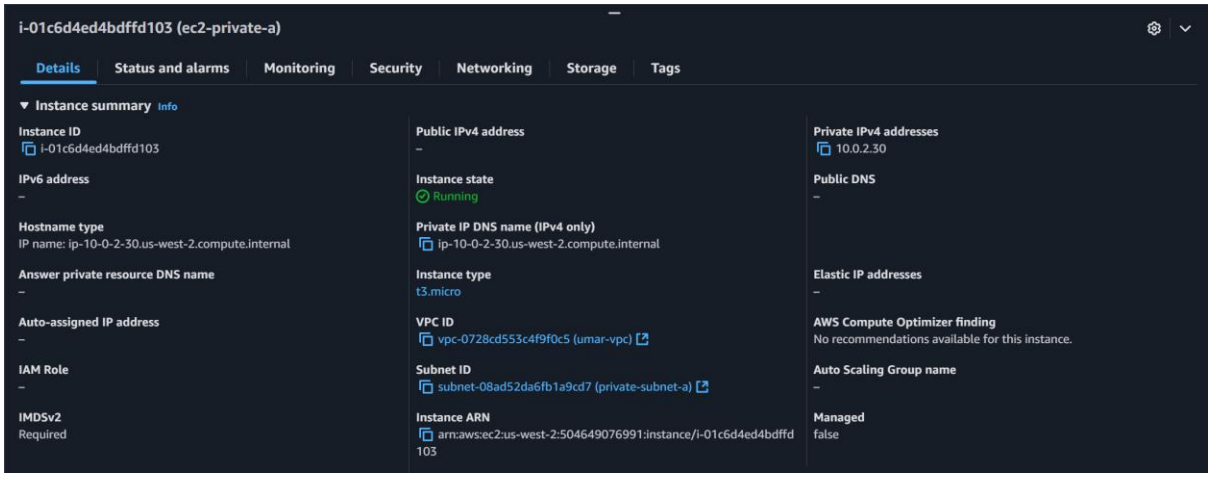
Public EC2 Instance:

- Name: ec2-public-a
- AMI: Ubuntu 24.04 LTS
- Instance Type: t3.micro
- Key Pair: umarsatti.pem
- Security Group: umar-public-sg



Private EC2 Instance:

- Name: ec2-private-a
- AMI: Ubuntu 24.04 LTS
- Instance Type: t3.micro
- Key Pair: umarsatti.pem
- Security Group: umar-private-sg



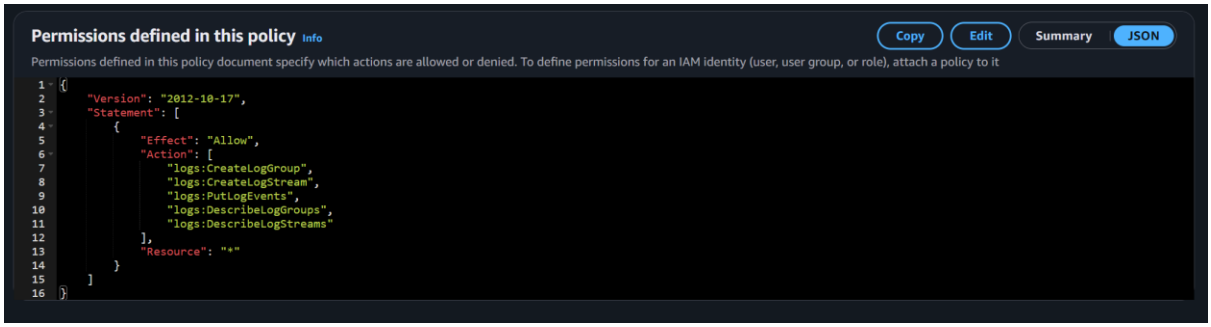
Task 1.8: Create IAM Role and Policy for VPC Flow Logs

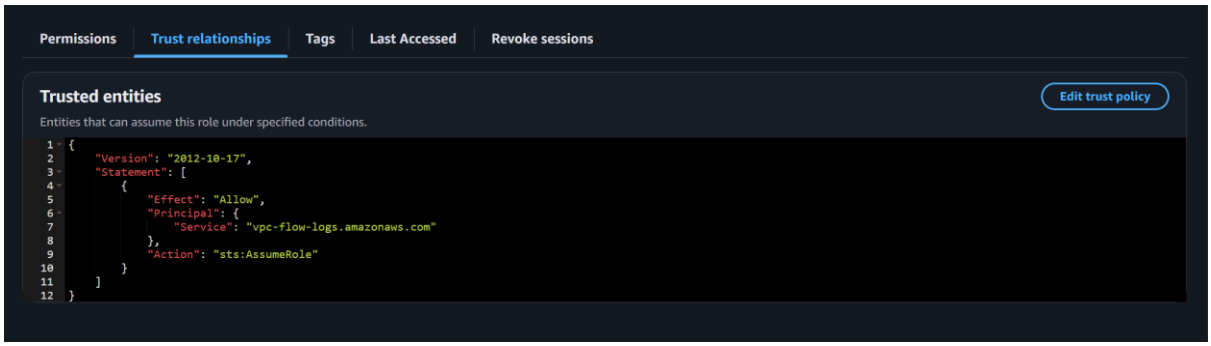
Steps:

1. Navigate to the IAM service from the AWS Management Console search bar
2. Create a new policy that allows the VPC Flow Logs service to write to CloudWatch Logs
3. Create an IAM role and attach the custom policy to it
4. Add a trust relationship that allows the VPC Flow logs to assume the IAM role.

I created an IAM policy and IAM role to grant permission for VPC Flow Logs to send logs to CloudWatch

- Policy Permissions:
    - logs:CreateLogGroup
    - logs:CreateLogStream
    - logs:PutLogEvents
    - logs:DescribeLogGroups
    - logs:DescribeLogStreams
- IAM Role: umar-vpc-flowlogs-role
- Trusted Entity: vpc-flow-logs.amazonaws.com



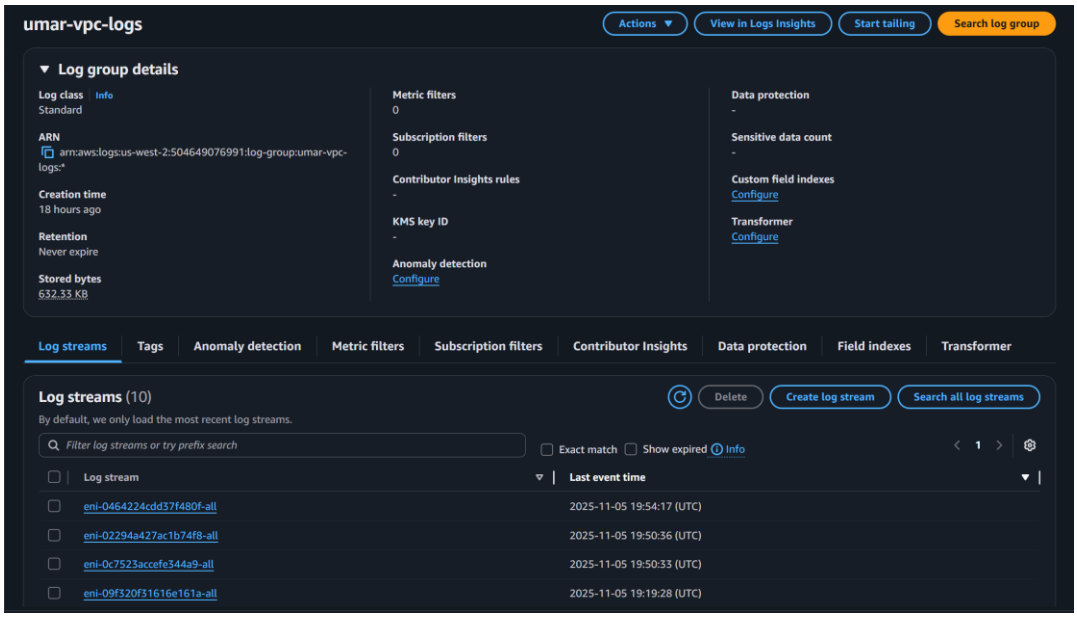


Task 1.9: Create CloudWatch Logs to publish VPC Flow Logs

Steps:

1. Navigate to “CloudWatch” service from the AWS Management Console search bar
2. Click on “Log groups” under “Logs” in the left navigation panel
3. Click on “Create log group” button
4. Choose a Log group name, Retention setting, and Log class.
5. Click on “Create” button

I created a CloudWatch Log group with the name “umar-vpc-logs” and retention period of “Never Expire”. This Log group will publish the logs generated from the VPC flow logs.



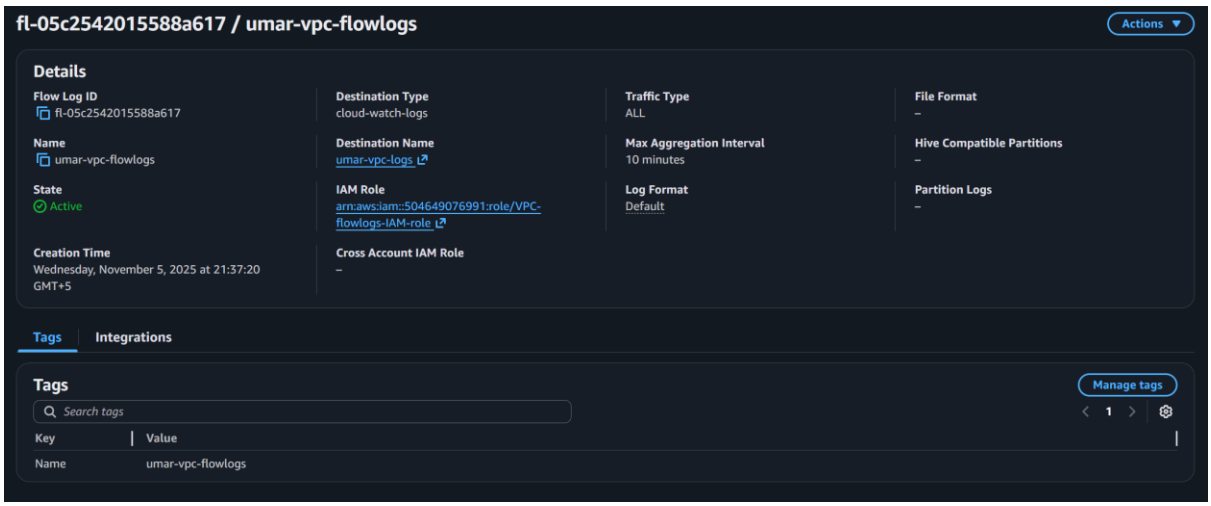
Task 1.10: Enable VPC Flow Logs for network traffic monitoring

Steps:

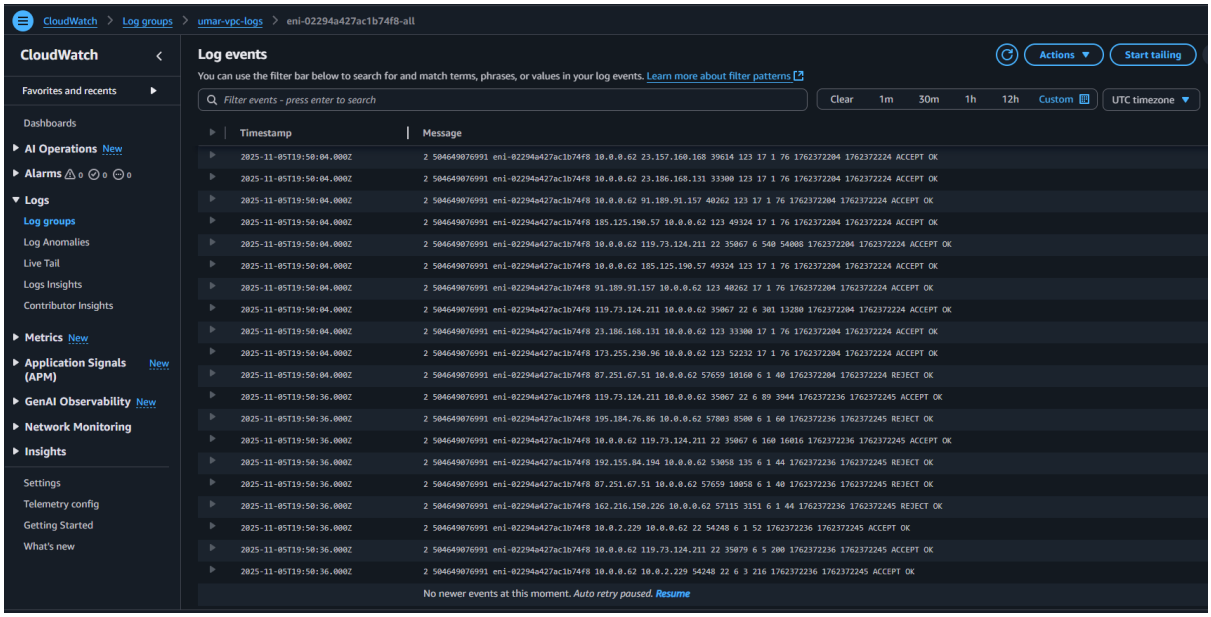
1. Navigate to “VPC” from the AWS Management Console search bar
2. Choose the target VPC and open the “Flow Logs” tab
3. Click “Create Flow Log”
4. Select “Destination” as CloudWatch Logs or S3
5. Choose the IAM role created in Task 1.8
6. Click “Create Flow Log”

I enabled VPC Flow Logs to capture detailed network traffic information for analysis and monitoring.

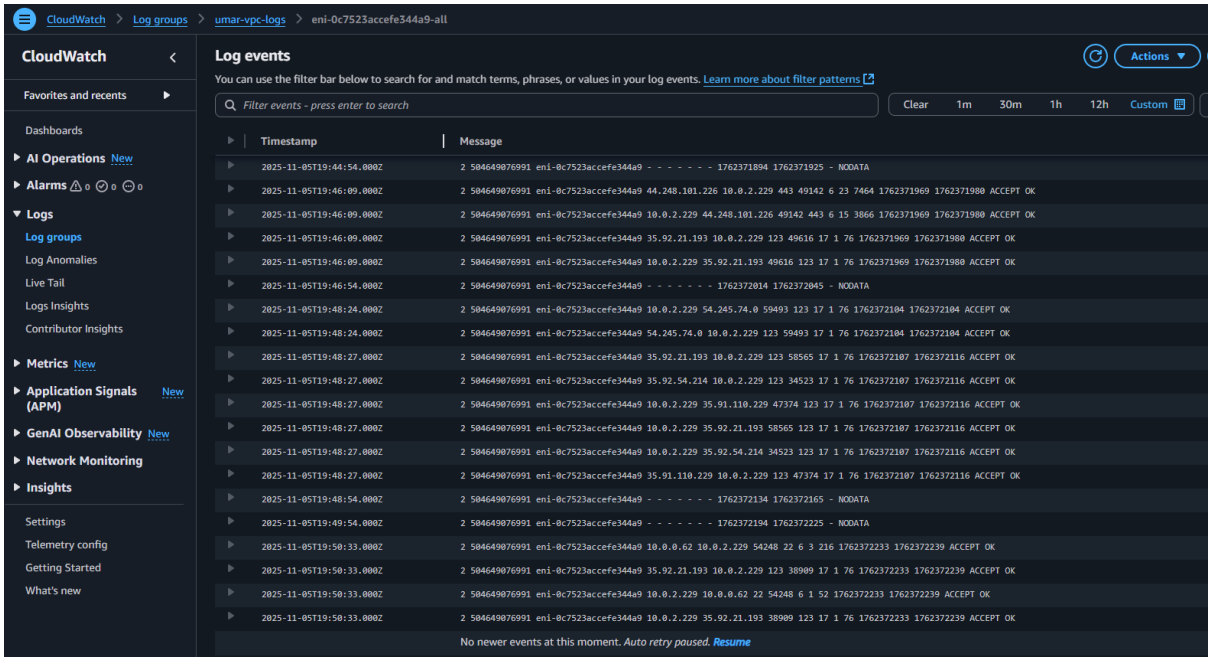
- VPC Flow Logs Name: umar-vpc-flowlogs
- Destination: CloudWatch Logs
- Log Group: umar-vpc-logs
- IAM Role: VPC-flowlogs-IAM-role



The following image shows the CloudWatch Log Stream of public EC2 instance with Elastic Network Interface ID eni-02294a427ac1b74f8



The following image shows the CloudWatch Log Stream of private EC2 instance with Elastic Network Interface ID eni-02294a427ac1b74f8



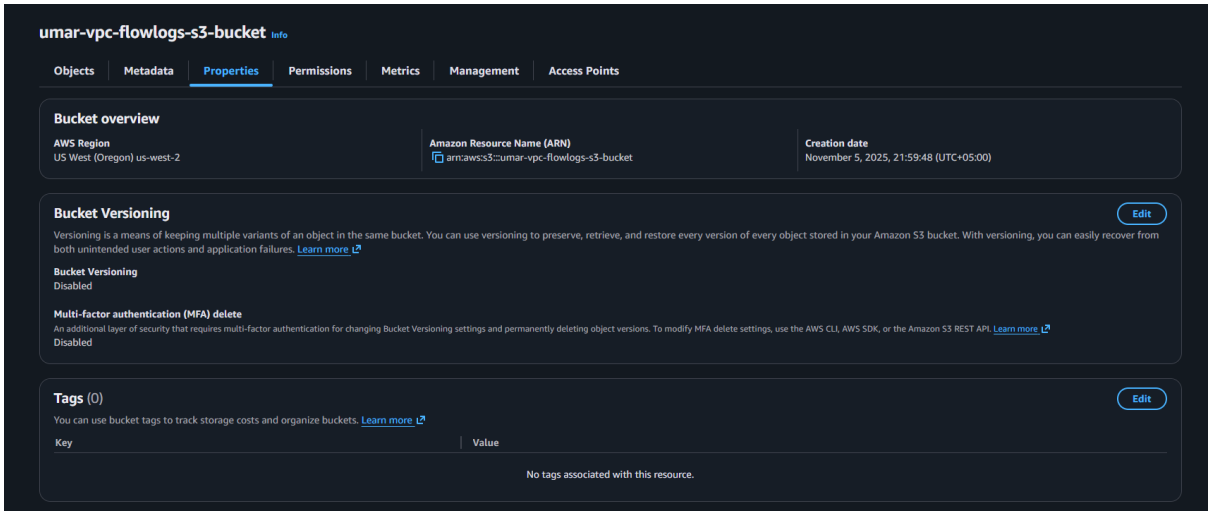
Task 1.11: Create S3 bucket to store VPC Flow logs

Steps:

1. Navigate to “S3” from the AWS Management Console search bar
2. Click on “Create bucket” button
3. Choose bucket type, name, and encryption settings
4. Click on “Create bucket” button

I created an S3 bucket to store VPC Flow logs

- **Bucket Name:** umar-vpc-flowlogs-s3-bucket
- **Object Ownership:** ACLs disabled
- **Block All Public Access:** Enabled
- **Versioning:** Disabled
- **Encryption:** SSE-S3 enabled



Task 1.12: Create VPC Endpoints for secure private connectivity to AWS services

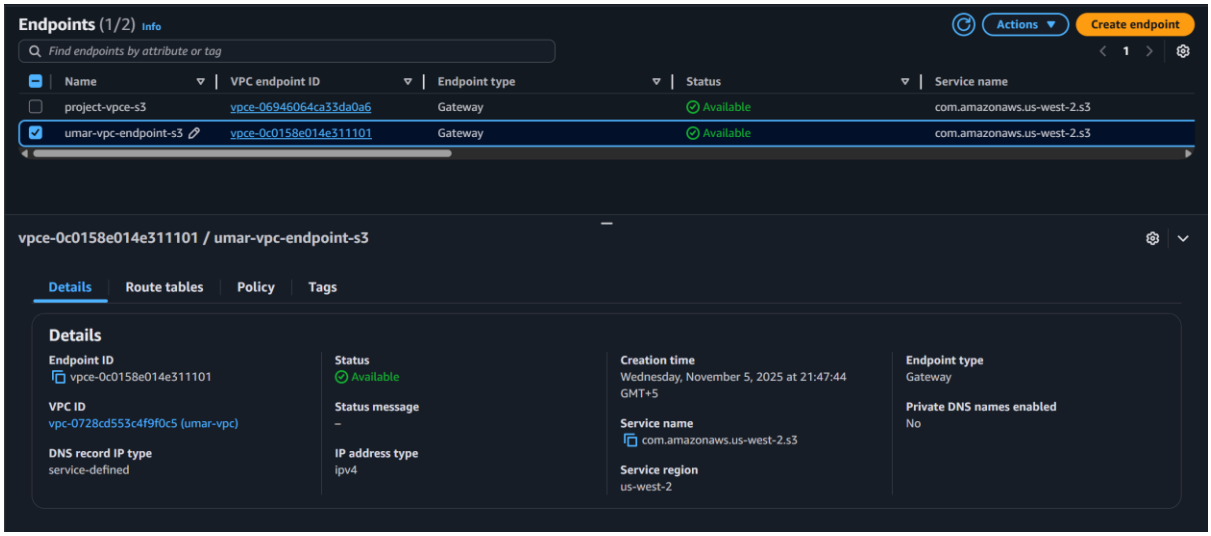
Steps:

1. Navigate to “VPC” from the AWS Management Console search bar
2. Click on “Endpoints” in the left navigation panel
3. Click “Create endpoint”
4. Choose Name, Type, and Services, and VPC
5. Click on “Create endpoint” button

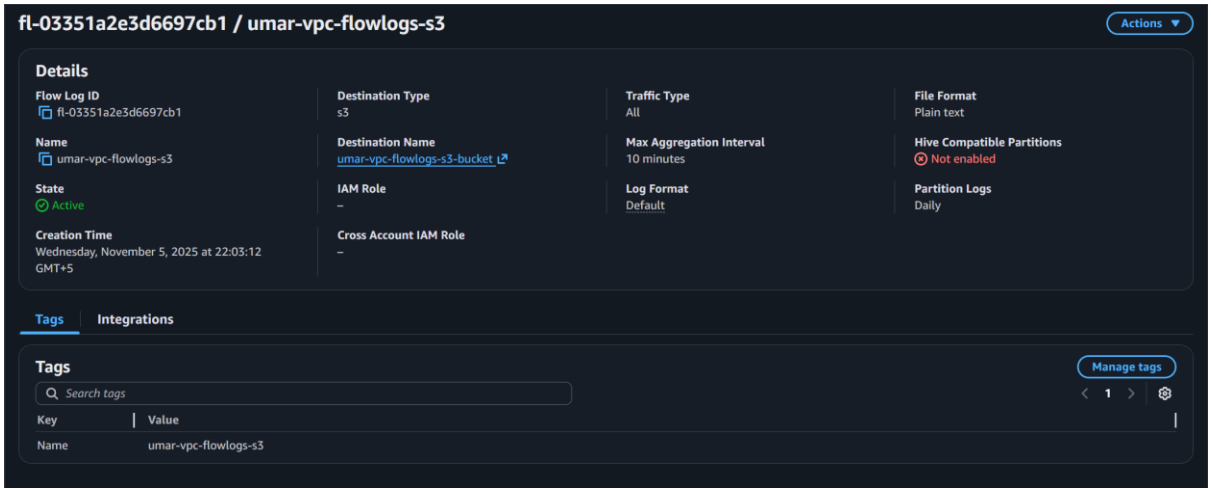
6. Additionally, create a VPC flow log with the same steps defined in Task 1.10 but change the Destination to “Send to an Amazon S3 bucket” by adding the bucket ARN

I created a VPC Endpoint for Amazon S3 to allow private communication between the VPC and S3 without using the public internet

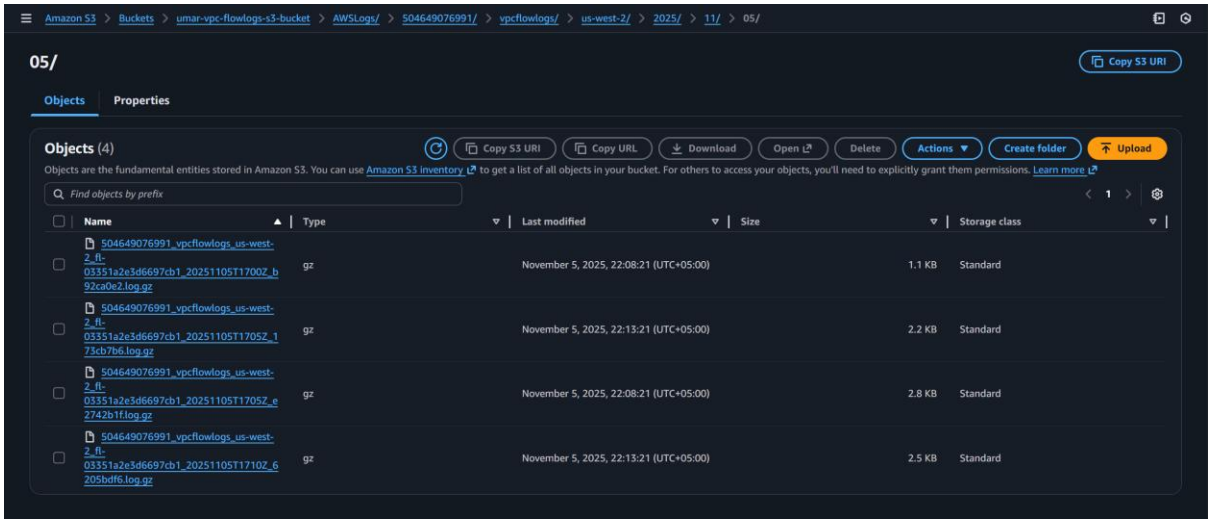
- Endpoint Name: umar-vpc-endpoint-s3
- Service: com.amazonaws.us-west-2.s3
- Endpoint type: Gateway



VPC Flow logs using S3 as the Destination type:



VPC Flow logs are stored inside the newly created S3 bucket as shown below:



Task 1.13: Test connectivity between subnets and to the internet

Steps:

1. Navigate to the “EC2” service from the AWS Management Console search bar
2. In the left navigation panel, select “Instances”
3. Select the instance and click “Connect” button
4. Choose “EC2 Instance Connect” to SSH into the instance
5. Make sure the key pair has the necessary permissions by using “chmod 400 <keypair.pem> command”
6. Connect to the EC2 instance using the command SSH command.
7. For Ubuntu instances, use “ssh -I <keypair.pem> ubuntu@<ip address>” command
8. For Amazon Linux instances, use “ssh -I <keypair.pem> ec2-user@<ip address>” command

I downloaded the keypair and changed its permission by using “chmod 400 umarsatti.pem” command. Instead of using EC2 Instance Connect or a local terminal on my computer, I used an SSH Client (MobaXterm) to connect to the EC2 instance.

The image below shows the ping command from the EC2 in public subnet A with the internal IP 10.0.0.62 to EC2 in private subnet A with the internal IP 10.0.2.229:

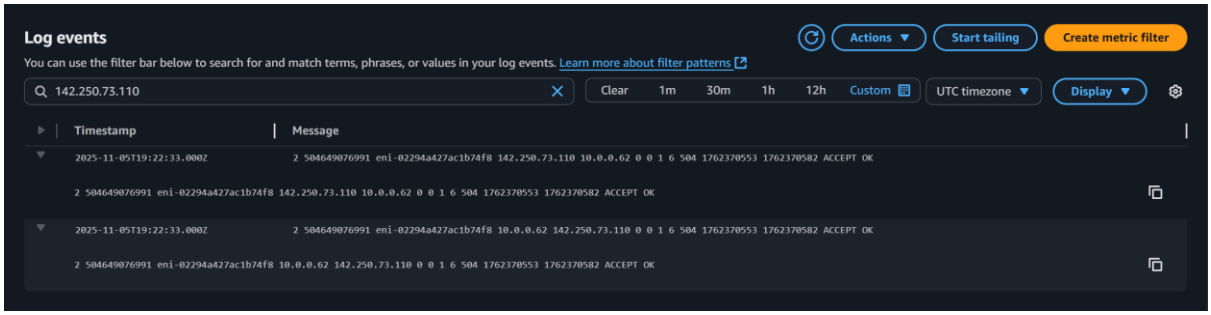
```
ubuntu@ip-10-0-0-62:~$ whoami
ubuntu
ubuntu@ip-10-0-0-62:~$ ping google.com
PING google.com (142.250.73.110) 56(84) bytes of data.
64 bytes from pnseaa-an-in-f14.1e100.net (142.250.73.110): icmp_seq=1 ttl=117 time=5.52 ms
64 bytes from pnseaa-an-in-f14.1e100.net (142.250.73.110): icmp_seq=2 ttl=117 time=5.57 ms
64 bytes from pnseaa-an-in-f14.1e100.net (142.250.73.110): icmp_seq=3 ttl=117 time=5.59 ms
64 bytes from pnseaa-an-in-f14.1e100.net (142.250.73.110): icmp_seq=4 ttl=117 time=5.54 ms
64 bytes from pnseaa-an-in-f14.1e100.net (142.250.73.110): icmp_seq=5 ttl=117 time=5.60 ms
64 bytes from pnseaa-an-in-f14.1e100.net (142.250.73.110): icmp_seq=6 ttl=117 time=5.56 ms
^C
--- google.com ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5009ms
rtt min/avg/max/mdev = 5.517/5.563/5.601/0.027 ms
ubuntu@ip-10-0-0-62:~$ ping 10.0.2.229
PING 10.0.2.229 (10.0.2.229) 56(84) bytes of data.
64 bytes from 10.0.2.229: icmp_seq=1 ttl=127 time=0.560 ms
64 bytes from 10.0.2.229: icmp_seq=2 ttl=127 time=0.173 ms
64 bytes from 10.0.2.229: icmp_seq=3 ttl=127 time=0.185 ms
64 bytes from 10.0.2.229: icmp_seq=4 ttl=127 time=0.182 ms
64 bytes from 10.0.2.229: icmp_seq=5 ttl=127 time=0.164 ms
64 bytes from 10.0.2.229: icmp_seq=6 ttl=127 time=0.176 ms
64 bytes from 10.0.2.229: icmp_seq=7 ttl=127 time=0.183 ms
^C
--- 10.0.2.229 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6156ms
rtt min/avg/max/mdev = 0.164/0.231/0.560/0.134 ms
ubuntu@ip-10-0-0-62:~$
```

The image below shows the ping command from the EC2 in private subnet A with the internal IP 10.0.2.229 to EC2 in public subnet A with the internal IP 10.0.0.62:

```
[ec2-user@ip-10-0-2-229 ~]$ whoami
ec2-user
[ec2-user@ip-10-0-2-229 ~]$ ls
[ec2-user@ip-10-0-2-229 ~]$ ping google.com
PING google.com (142.250.73.78) 56(84) bytes of data.
64 bytes from pnseaa-am-in-f14.1e100.net (142.250.73.78): icmp_seq=1 ttl=116 time=6.14 ms
64 bytes from pnseaa-am-in-f14.1e100.net (142.250.73.78): icmp_seq=2 ttl=116 time=5.65 ms
64 bytes from pnseaa-am-in-f14.1e100.net (142.250.73.78): icmp_seq=3 ttl=116 time=5.65 ms
64 bytes from pnseaa-am-in-f14.1e100.net (142.250.73.78): icmp_seq=4 ttl=116 time=5.71 ms
64 bytes from pnseaa-am-in-f14.1e100.net (142.250.73.78): icmp_seq=5 ttl=116 time=5.70 ms
64 bytes from pnseaa-am-in-f14.1e100.net (142.250.73.78): icmp_seq=6 ttl=116 time=5.65 ms
64 bytes from pnseaa-am-in-f14.1e100.net (142.250.73.78): icmp_seq=7 ttl=116 time=5.65 ms
^C
--- google.com ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6007ms
rtt min/avg/max/mdev = 5.645/5.735/6.137/0.165 ms
[ec2-user@ip-10-0-2-229 ~]$ ping 10.0.0.62
PING 10.0.0.62 (10.0.0.62) 56(84) bytes of data.
64 bytes from 10.0.0.62: icmp_seq=1 ttl=64 time=0.219 ms
64 bytes from 10.0.0.62: icmp_seq=2 ttl=64 time=0.193 ms
64 bytes from 10.0.0.62: icmp_seq=3 ttl=64 time=0.280 ms
64 bytes from 10.0.0.62: icmp_seq=4 ttl=64 time=0.192 ms
64 bytes from 10.0.0.62: icmp_seq=5 ttl=64 time=0.191 ms
64 bytes from 10.0.0.62: icmp_seq=6 ttl=64 time=0.216 ms
64 bytes from 10.0.0.62: icmp_seq=7 ttl=64 time=0.202 ms
^C
--- 10.0.0.62 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6264ms
rtt min/avg/max/mdev = 0.191/0.213/0.280/0.029 ms
[ec2-user@ip-10-0-2-229 ~]$
```

Both the public and private instances can ping google.com which determines that both instances have access to the internet.

CloudWatch Logs to indicate successful ping to google.com (142.250.73.110)



CloudWatch Logs indicating successful ping from public EC2 instance (10.0.0.62) to private EC2 instance (10.0.2.229)