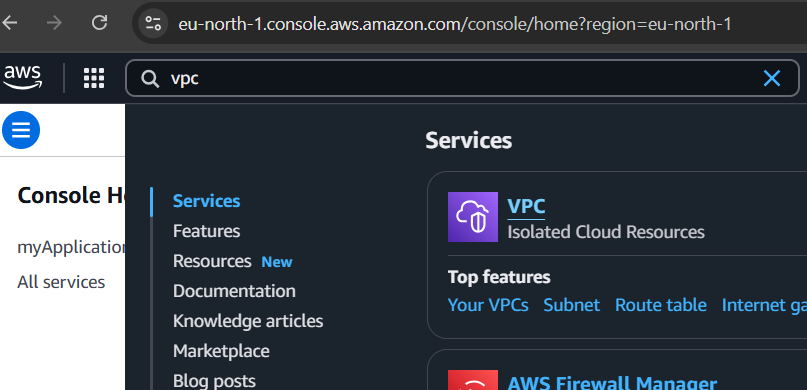
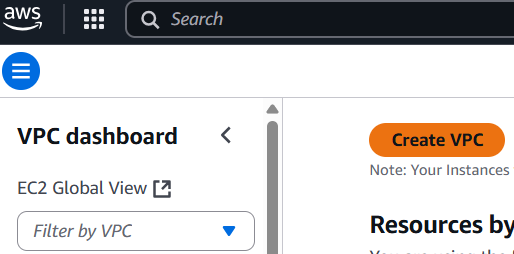
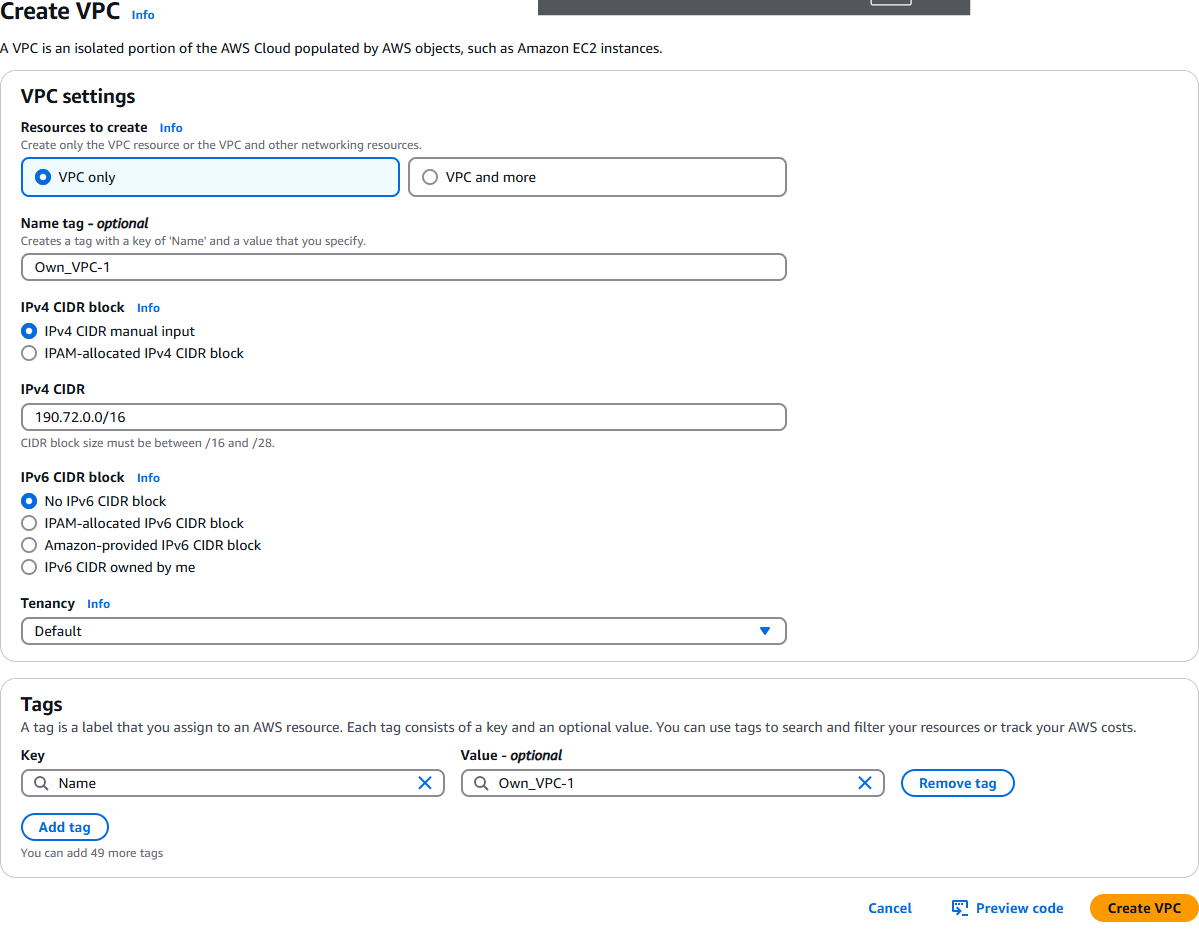
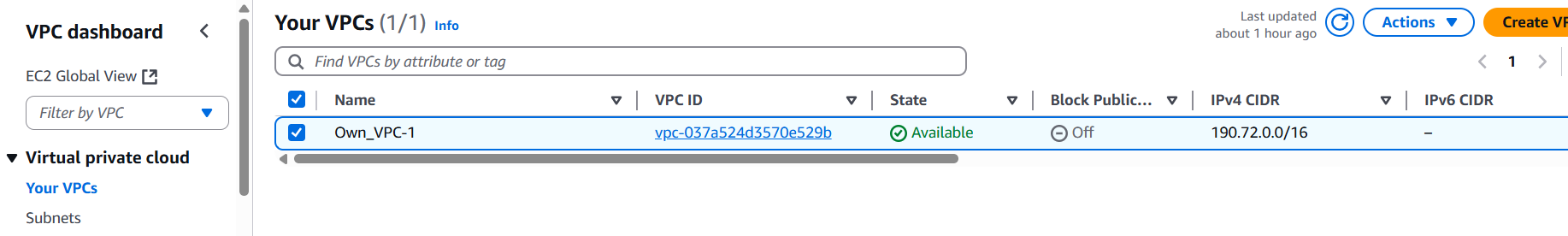
**Tasks on VPC-01**

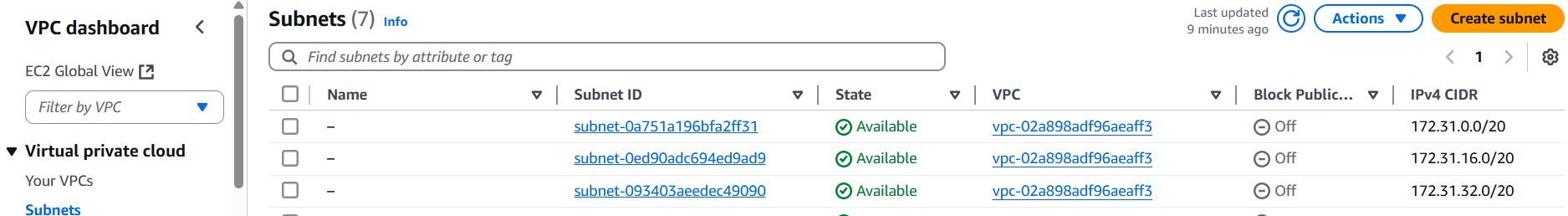
*1) Create VPC with 2 private and 2 public subnets.*

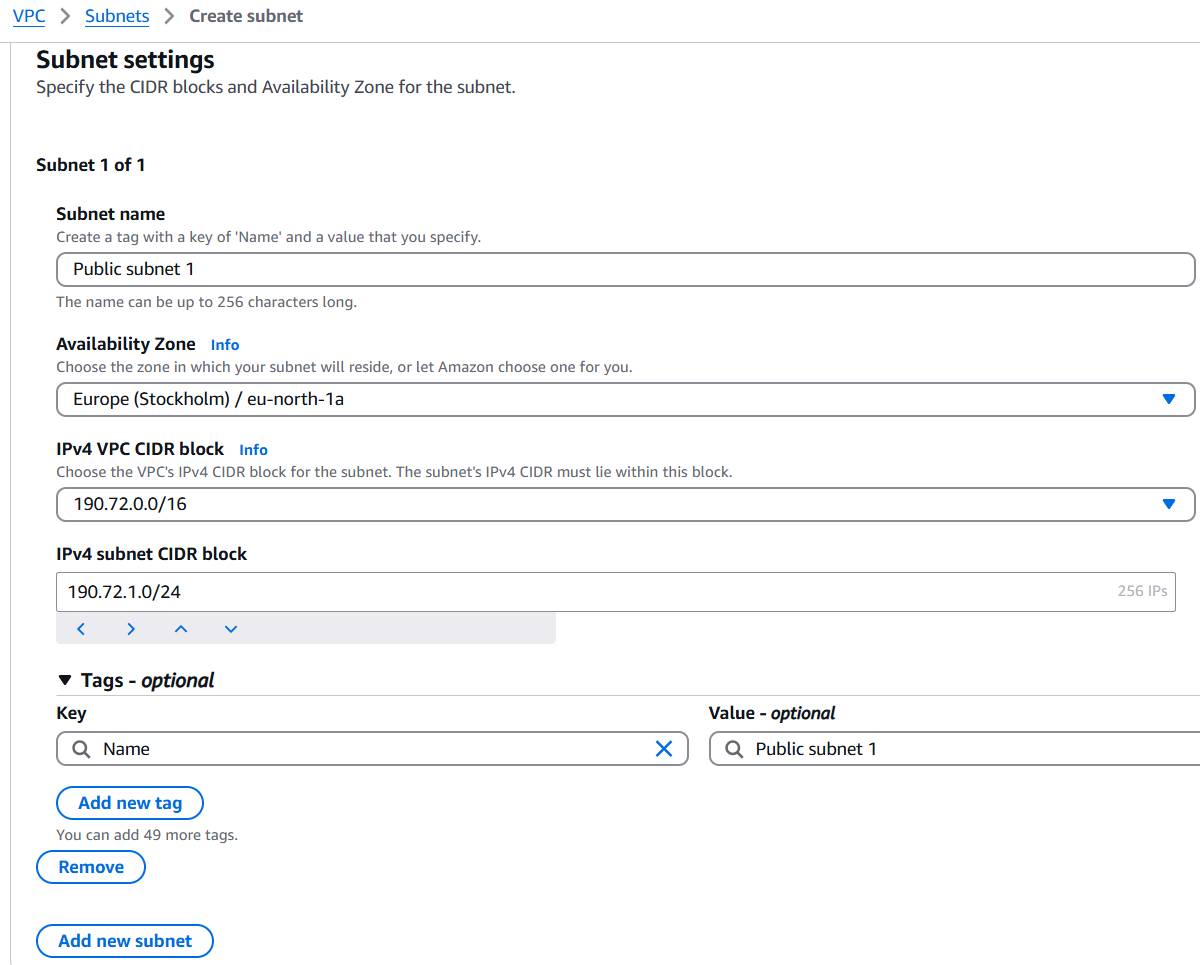
* From the AWS Console home search for VPC or you can click on All Services on the left hand side tab and search for it there.
* Once we are on the VPC dashboard, on there click on ‘Create VPC’
* Select VPC only, give it a name, select IPv4 CIDR manual manual input
* choose a valid IPv4 CIDR range, then click create VPC

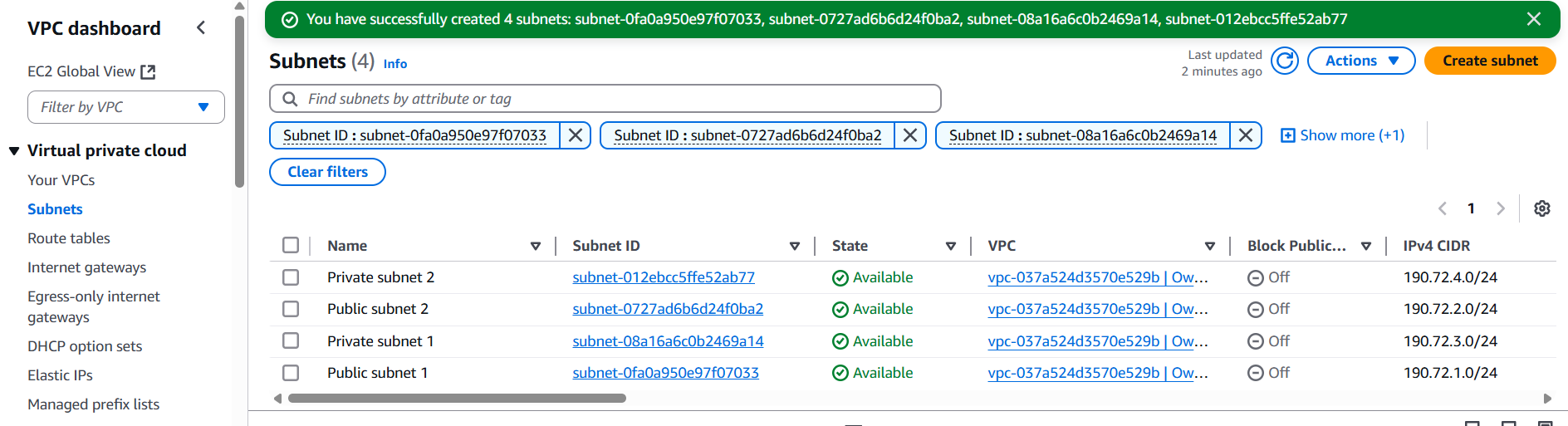




* Go to the Subnets section in the VPC console and click Create Subnet. Select your new VPC, and add 4 valid IPv4 subnet CIDR range for each of the 4 subnets you are creating and name them.
* Then click on create subnet at the bottom right when done.

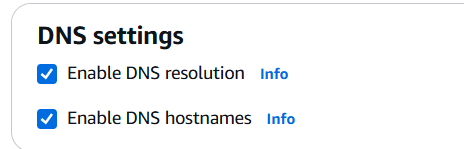
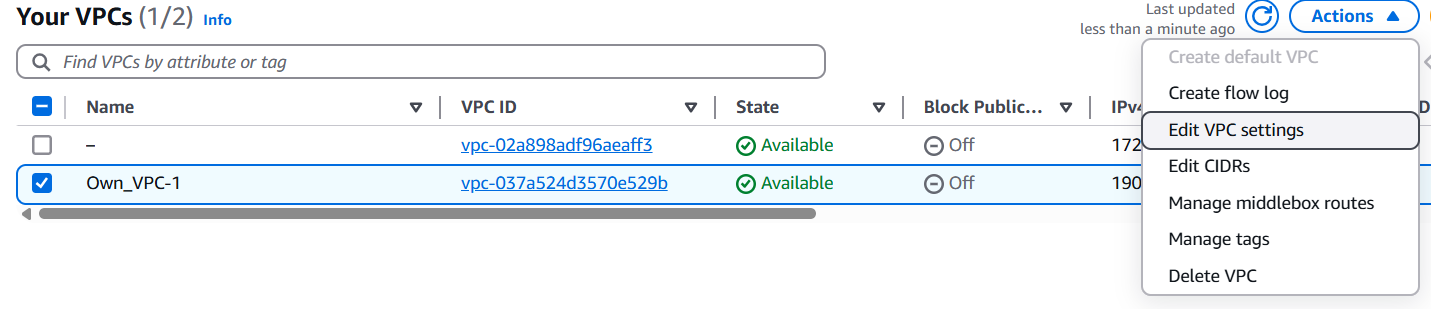


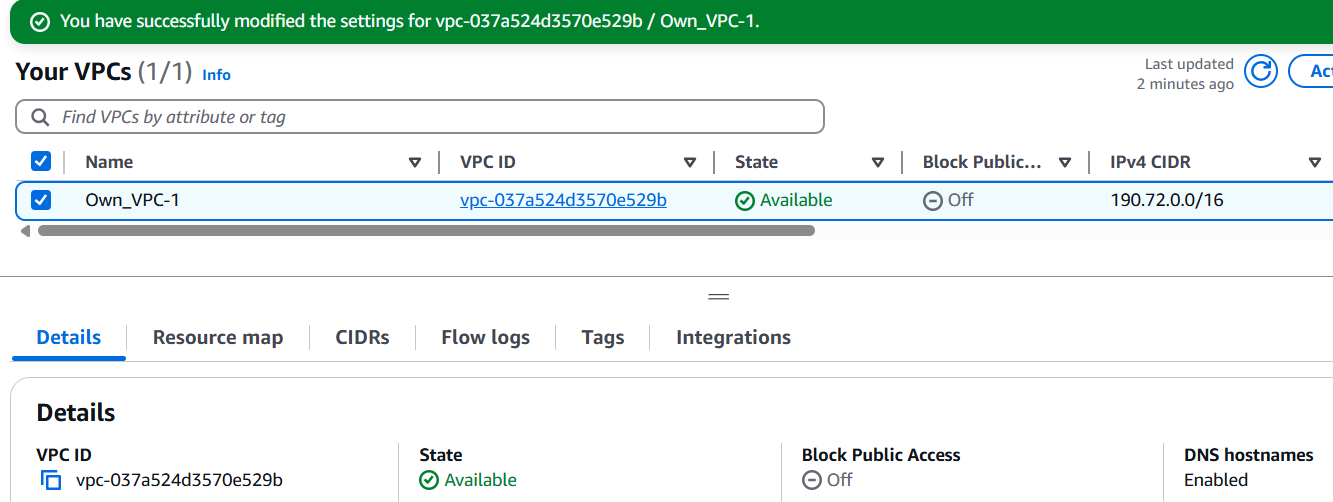




*2) Enable DNS Hostname in VPC*

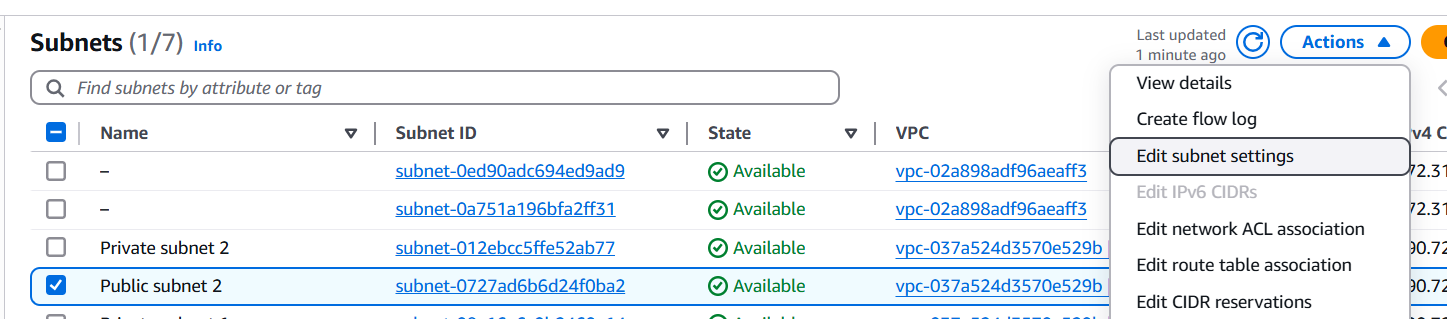
* Open the VPC Console.
* In the VPC Dashboard, click on Your VPCs.
* Select your VPC and click Actions > Edit VPC settings.
* Enable the option for DNS hostnames and click Save to allow EC2 instances in this VPC to have DNS names.

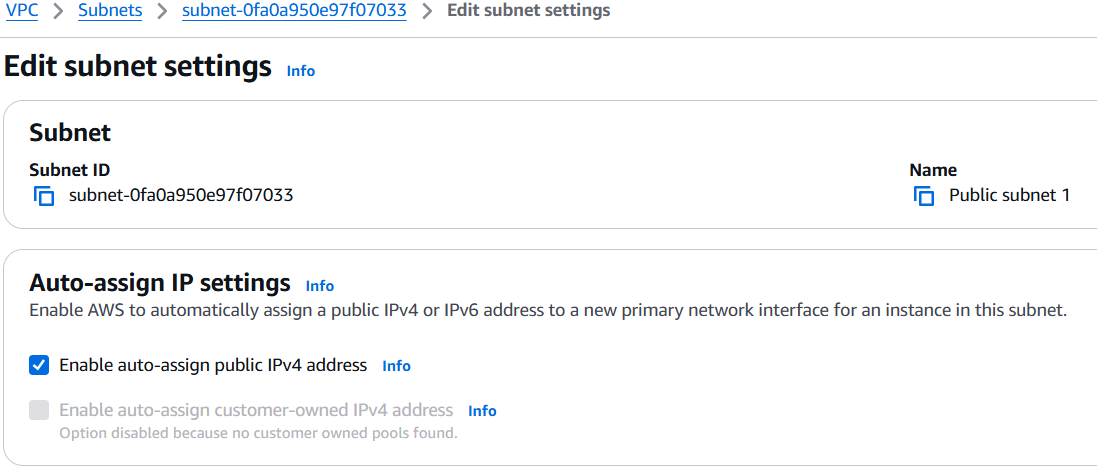




*3) Enable Auto Assign Public ip in 2 public subnets*

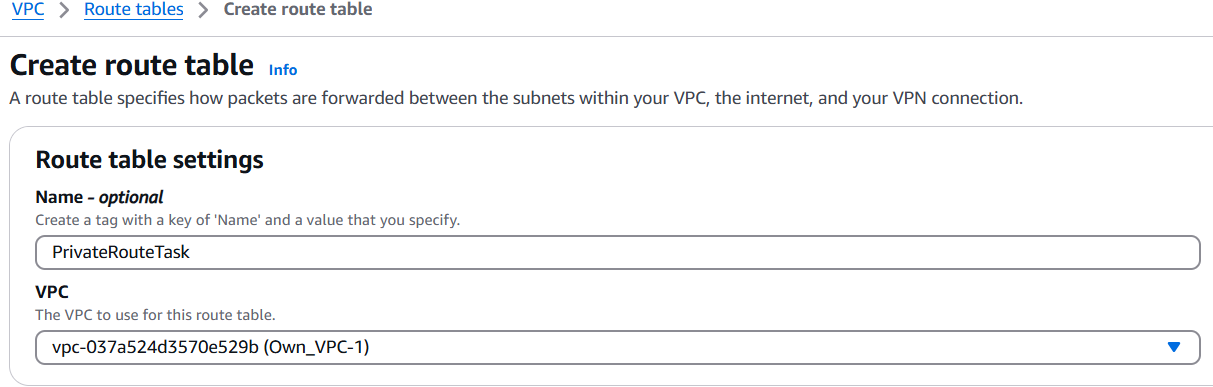
* Open the Subnet Console and select the public subnets
* Select Actions > Edit subnet settings.
* Enable Auto-assign Public IPv4 Address and save. This allows instances launched in these subnets to automatically get a public IP.

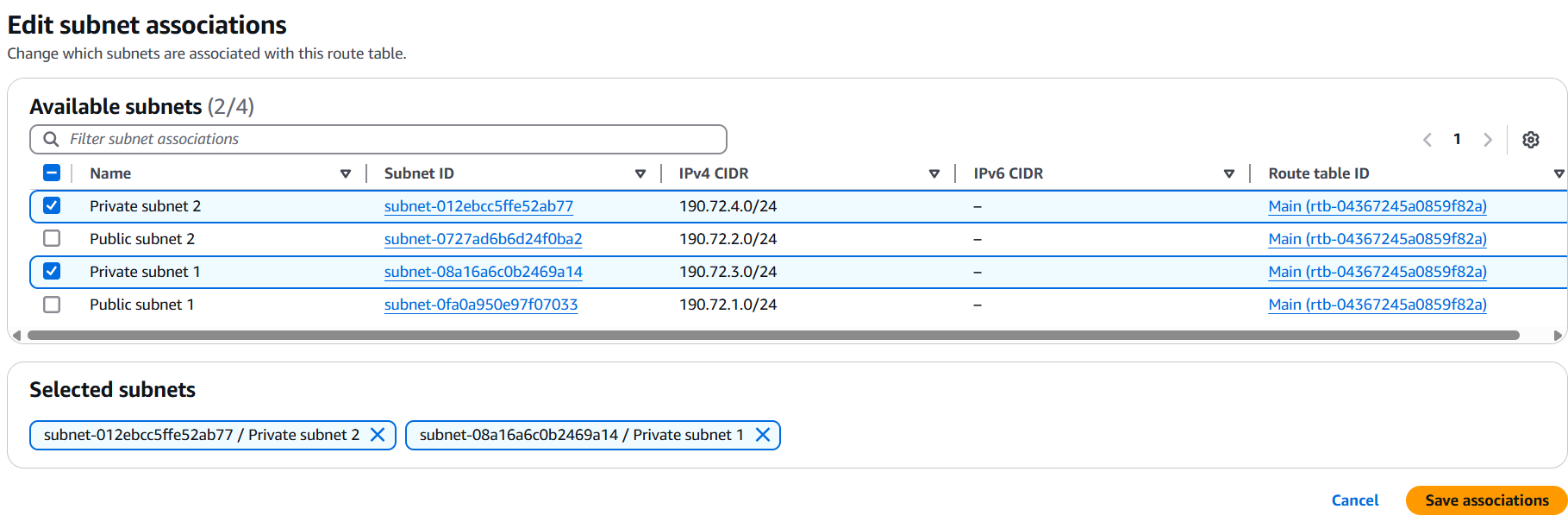


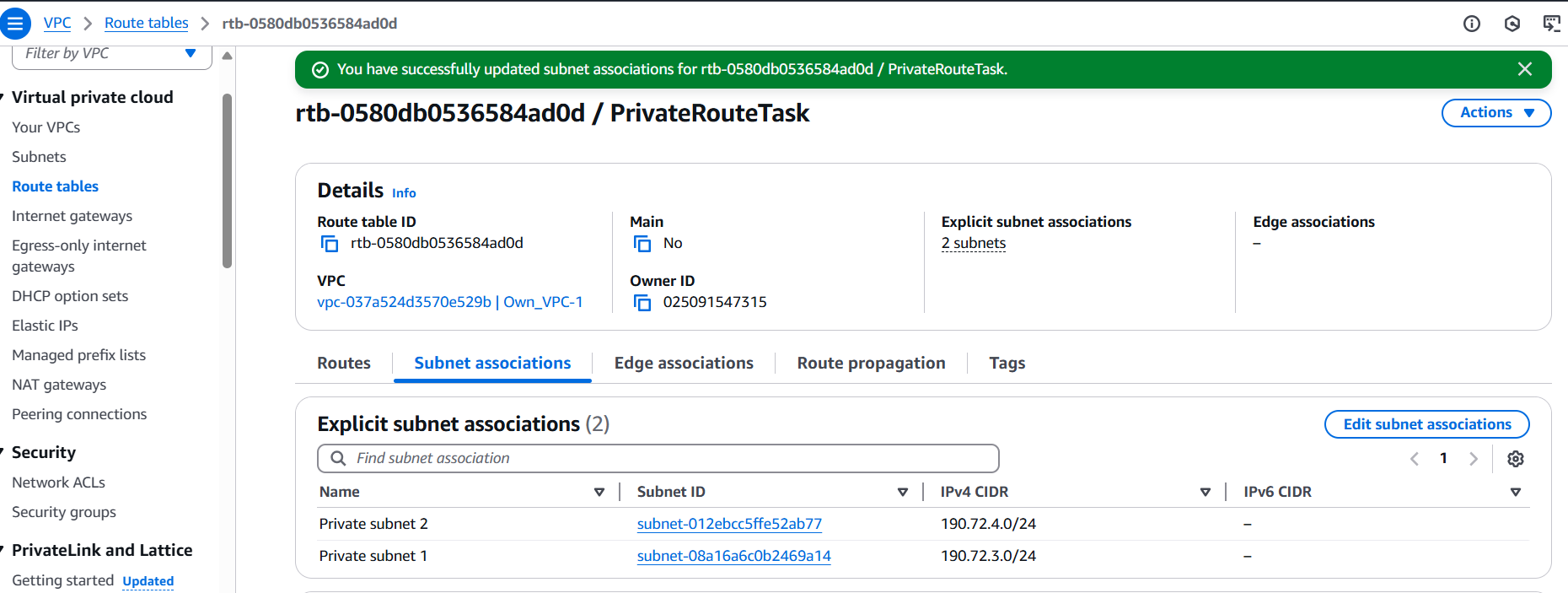


*4) Add 2 private subnets in private route table*

* Open the Route Tables section in the VPC Console.
* Click on Create a route table and give it a name and select the current VPC
* right click the created route table and select edit subnet associations
* add the two private subnets 1 and 2 and save.







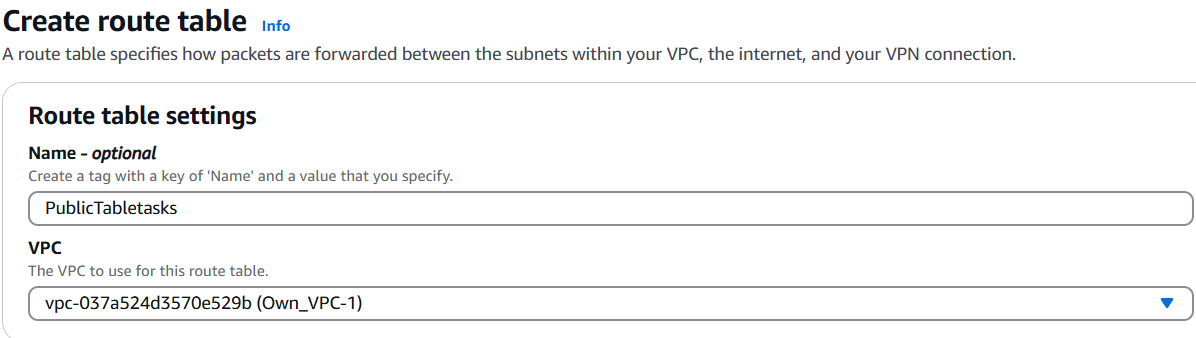
*5) Add 2 public subnets in public route table* ***&*** *6) Public route table will have the routes to internet and local*

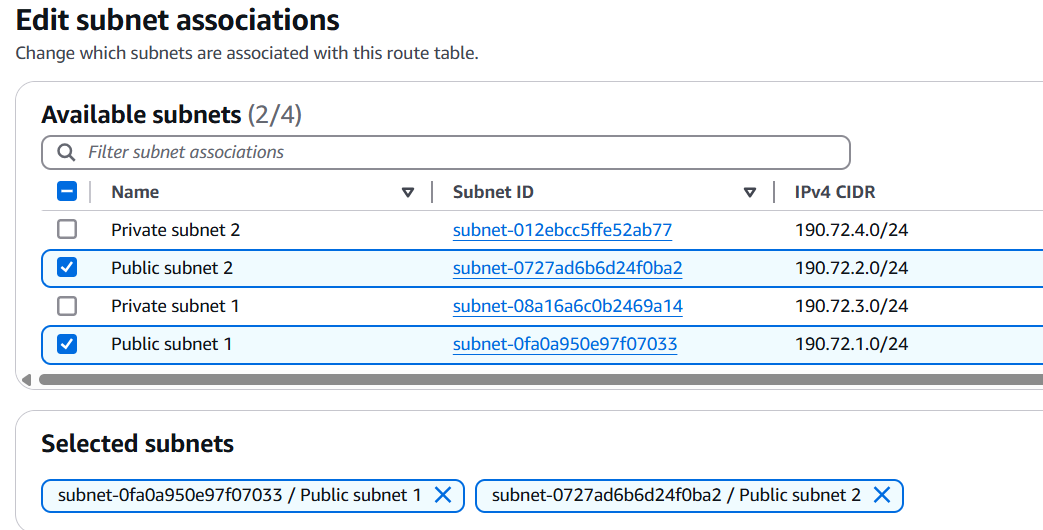
IGW allows public subnets and EC2 instances with public IPs to connect directly to the internet.

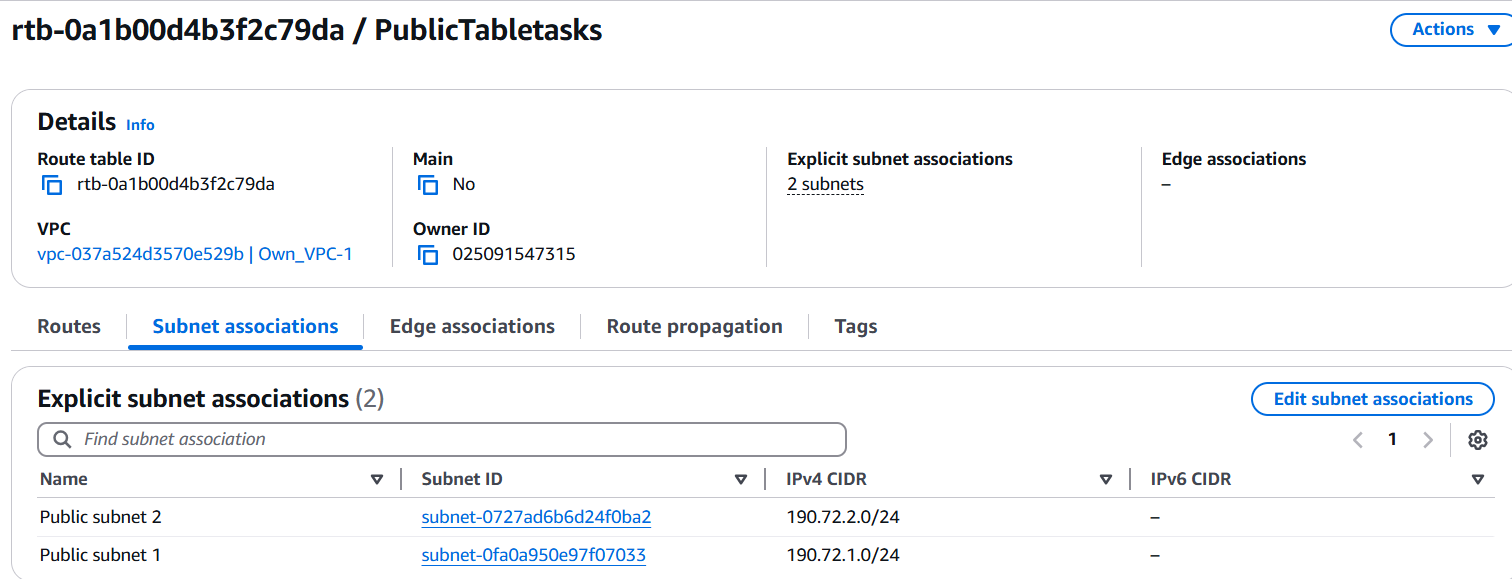
Destination: This is the IP range you want to route traffic to. **0.0.0.0/0**(which means anywhere on the internet and within the vpc and between the subnets)

Target: This is the resource used to reach that destination (Internet Gateway, NAT Gateway, Peering connection, etc.)

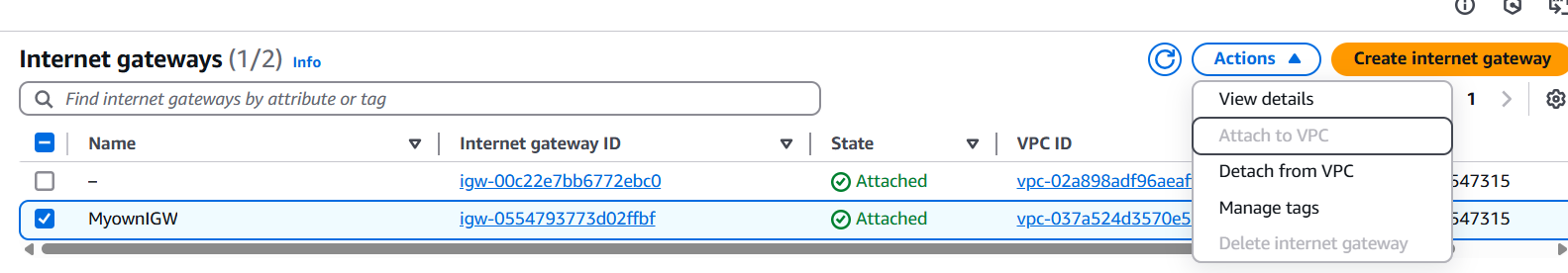
* A subnet is considered public if its associated route table includes a route pointing to an Internet Gateway (IGW). Without such a route, the subnet is considered private.
* Open the Route Tables section in the VPC Console.
* Click on Create a route table and give it a name and select the current VPC
* right click the created route table and select edit subnet associations
* add the two public subnets 1 and 2 and save.



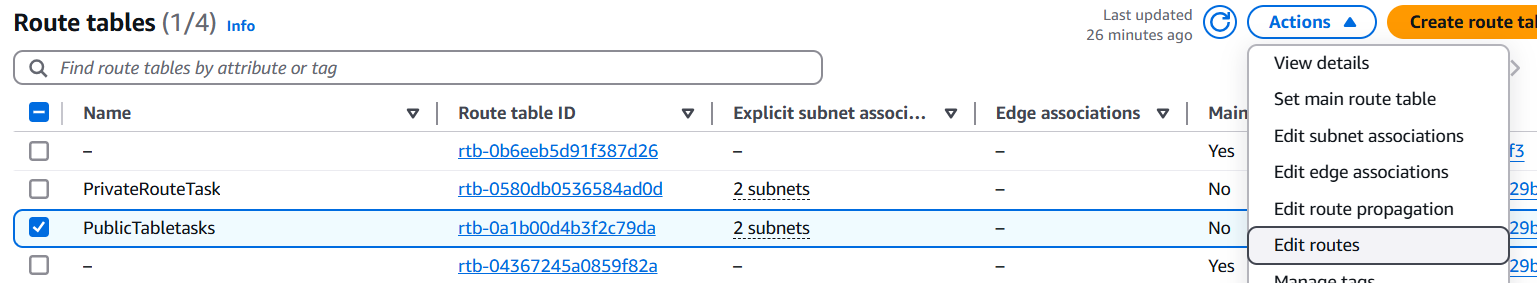


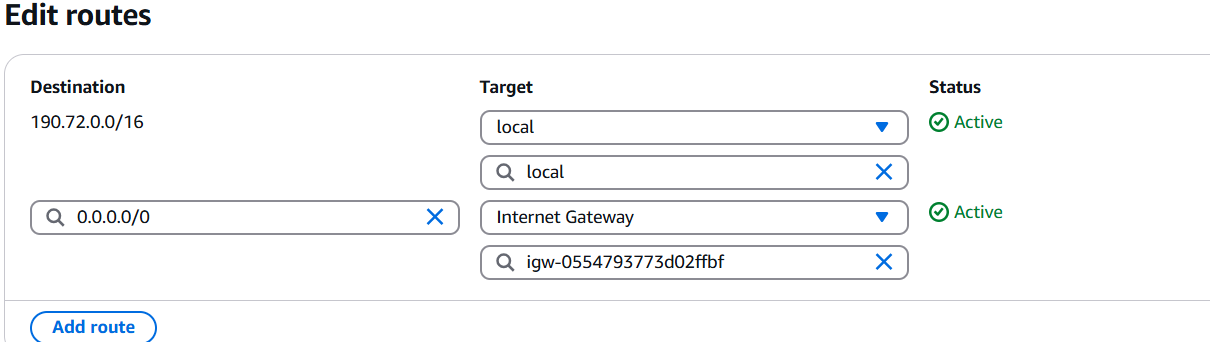


* From the VPC Dashboard → Internet Gateways → Create Internet Gateway.
* Then Select your new IGW → Click Actions → Attach to VPC.
* Choose your VPC from the dropdown and click Attach



* Click Routes tab → Edit routes, add routes
* Select the destination and for target select IGW and past the IGW ID associated to our VPC
* This allows instances in this subnet to access the internet.





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

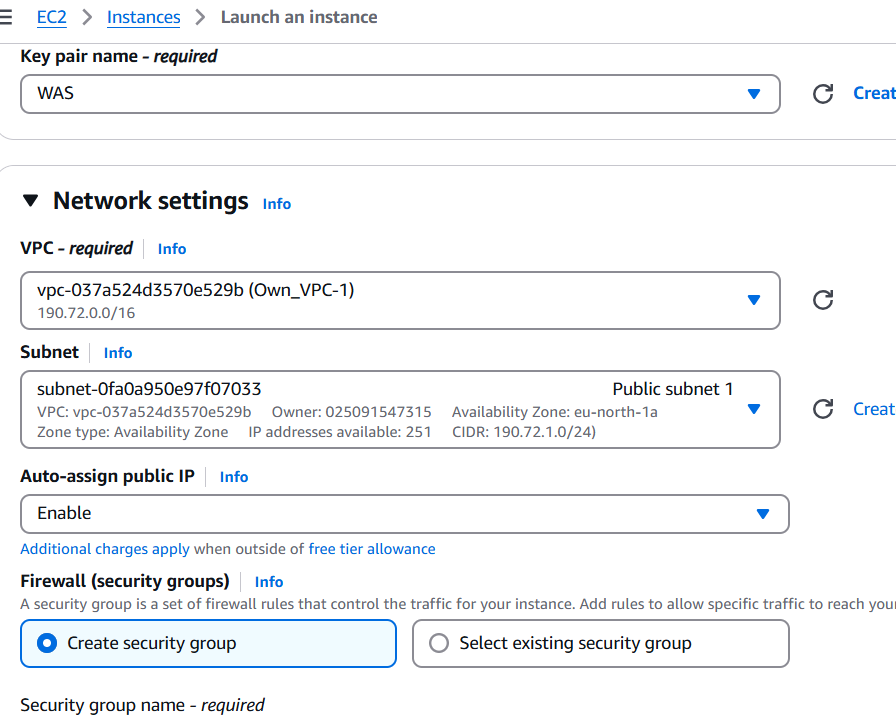
* Create and launce an instance like usual , just select your VPC and public subnet in the settings
* ssh into it and install php, we also need to install http cuz PHP scripts like index.php need a web server to be executed and shown in the browser
* yum update -y

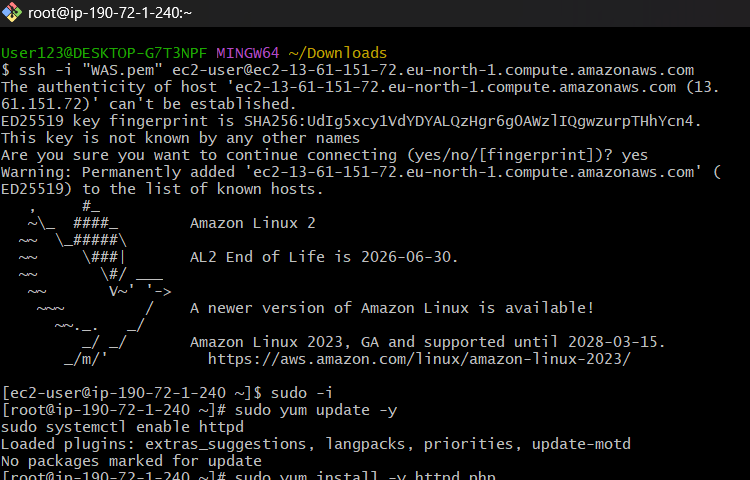
yum install -y httpd php

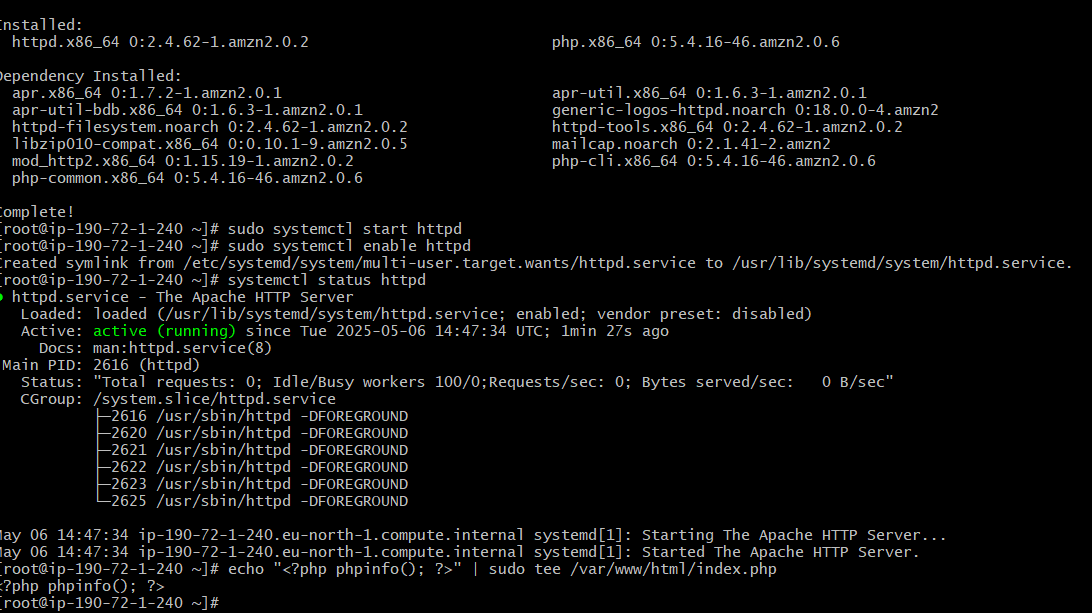
systemctl start httpd

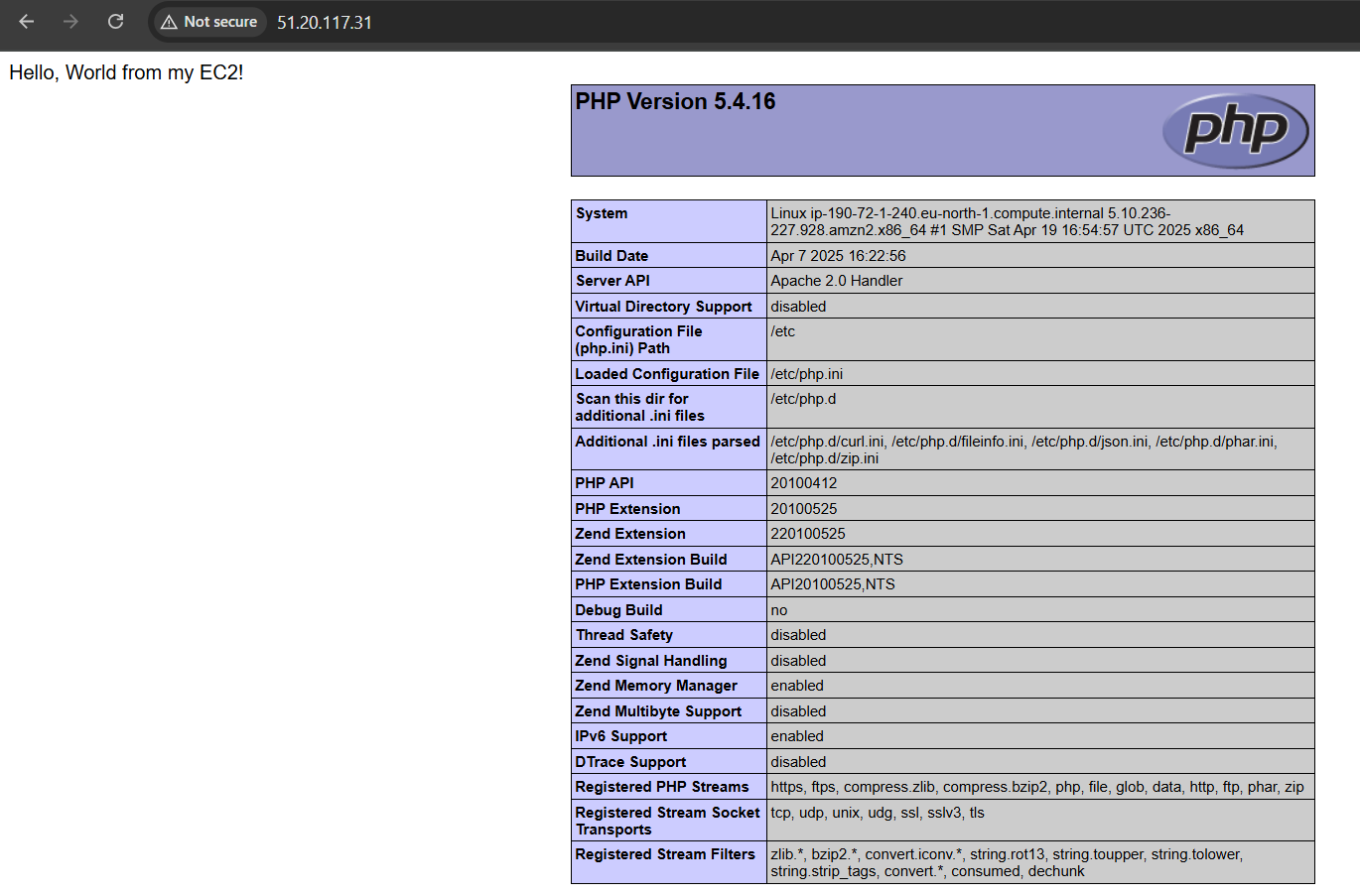
systemctl enable httpd

* add a php file with some php function or just add your content and check in through the browser









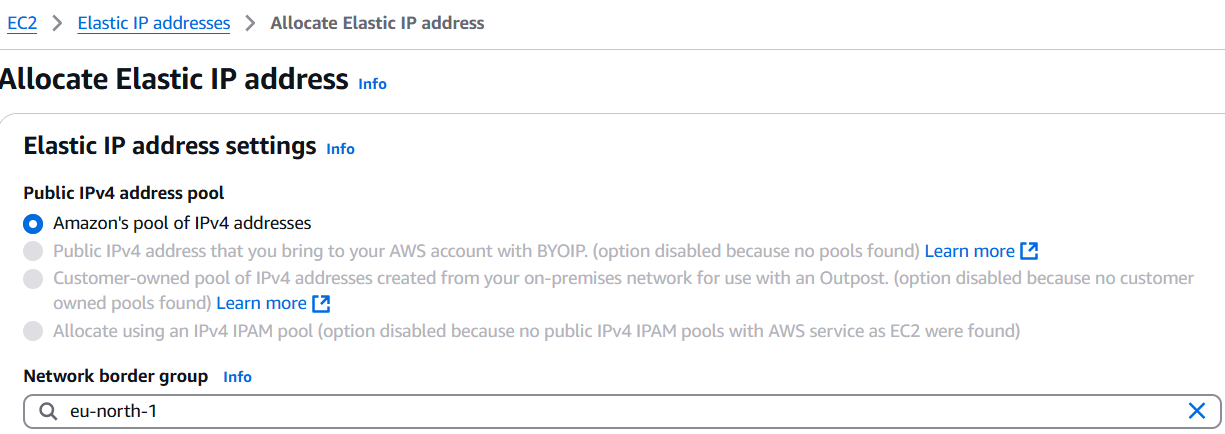
*8) Configure Nat gateway in public subnet and connect to private Instance*

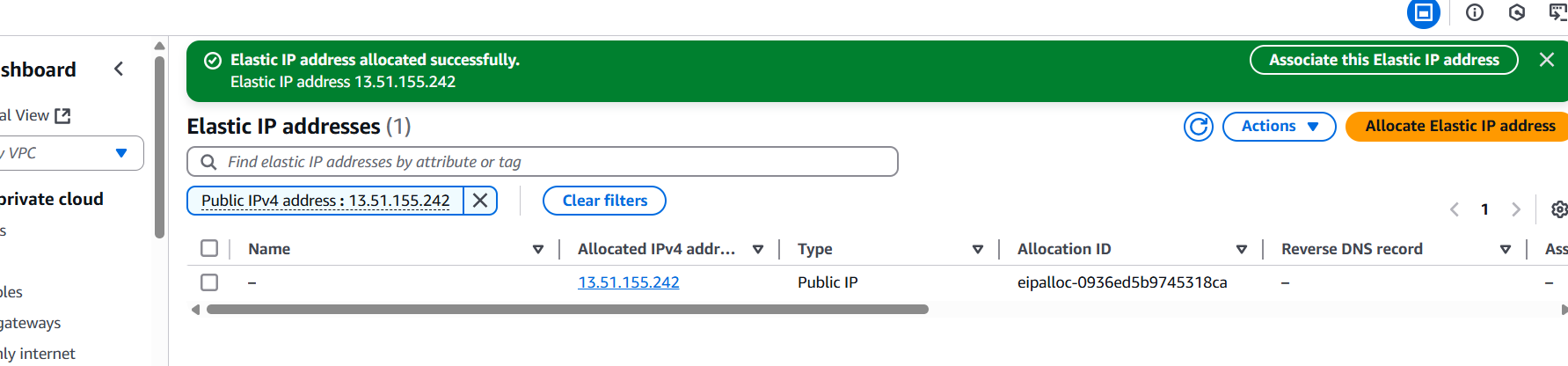
Elastic IP is a static, public IPv4. It is associated with your AWS account. Elastic IP remains the same even if the EC2 instance is stopped and restarted.

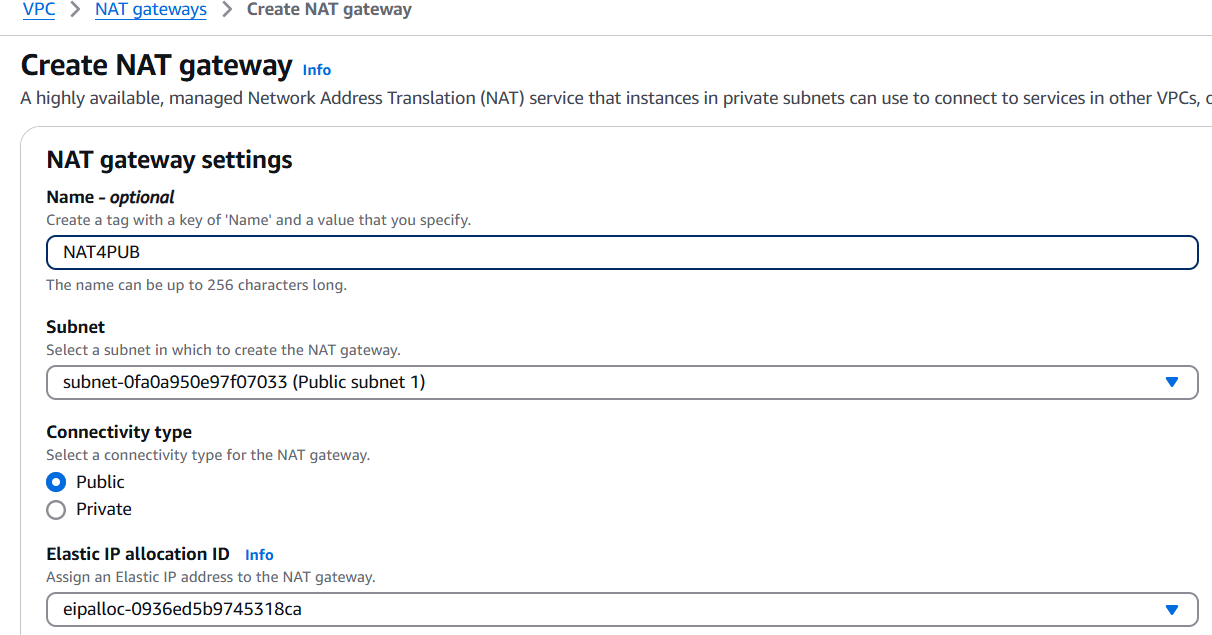
Elastic IPs allow internet access to your EC2 instances if they're in a private subnet

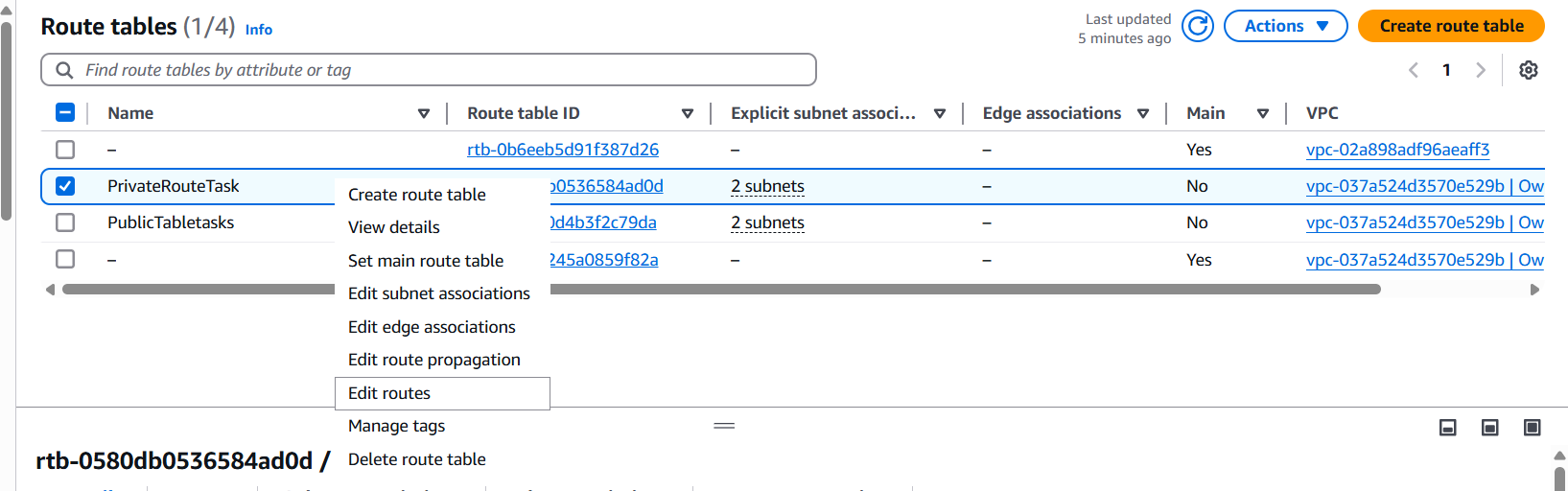
NAT Gateway allows instances in a private subnet to access the internet, but not be accessed from the internet.

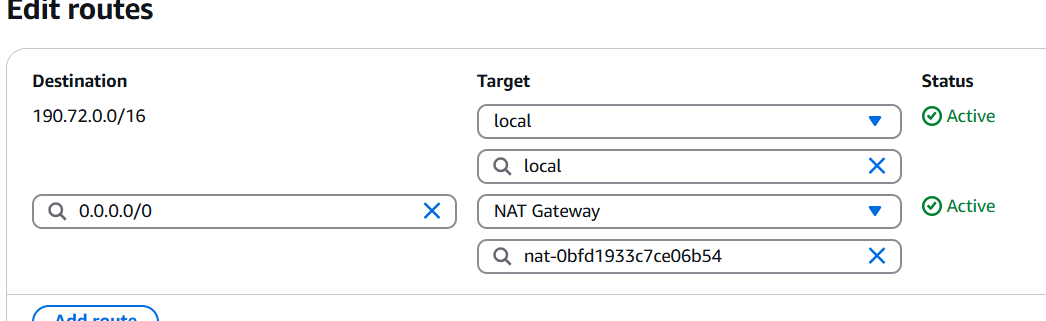
* Create Elastic IP- VPC dashboard→ Elastic IPs → Allocate
* from VPC console, select NAT Gateways and click Create NAT Gateway choose the public subnet, public as connectivity type and assign the Elastic IP.
* This allows instances in private subnets to access the internet through the NAT Gateway in the public subnet





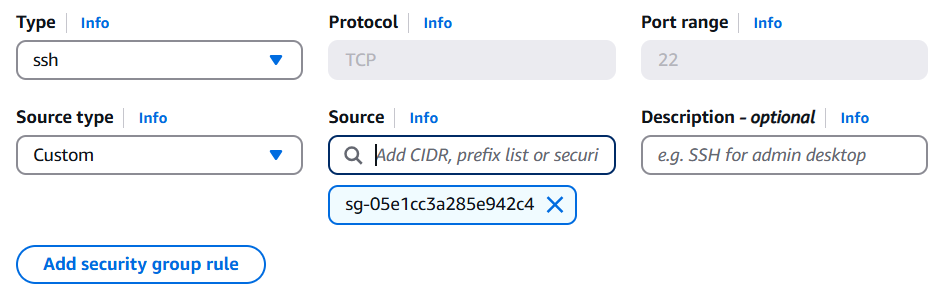


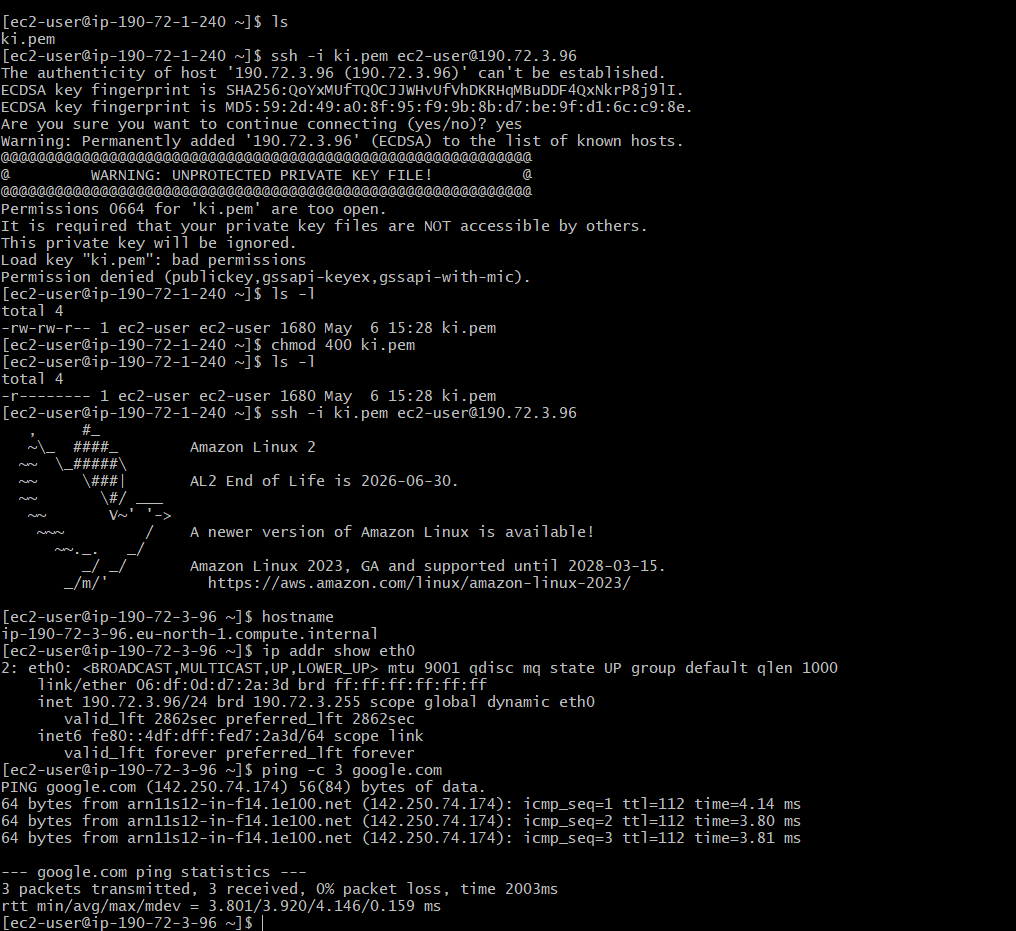




* To Ssh into a private instance and check it’s internet access-

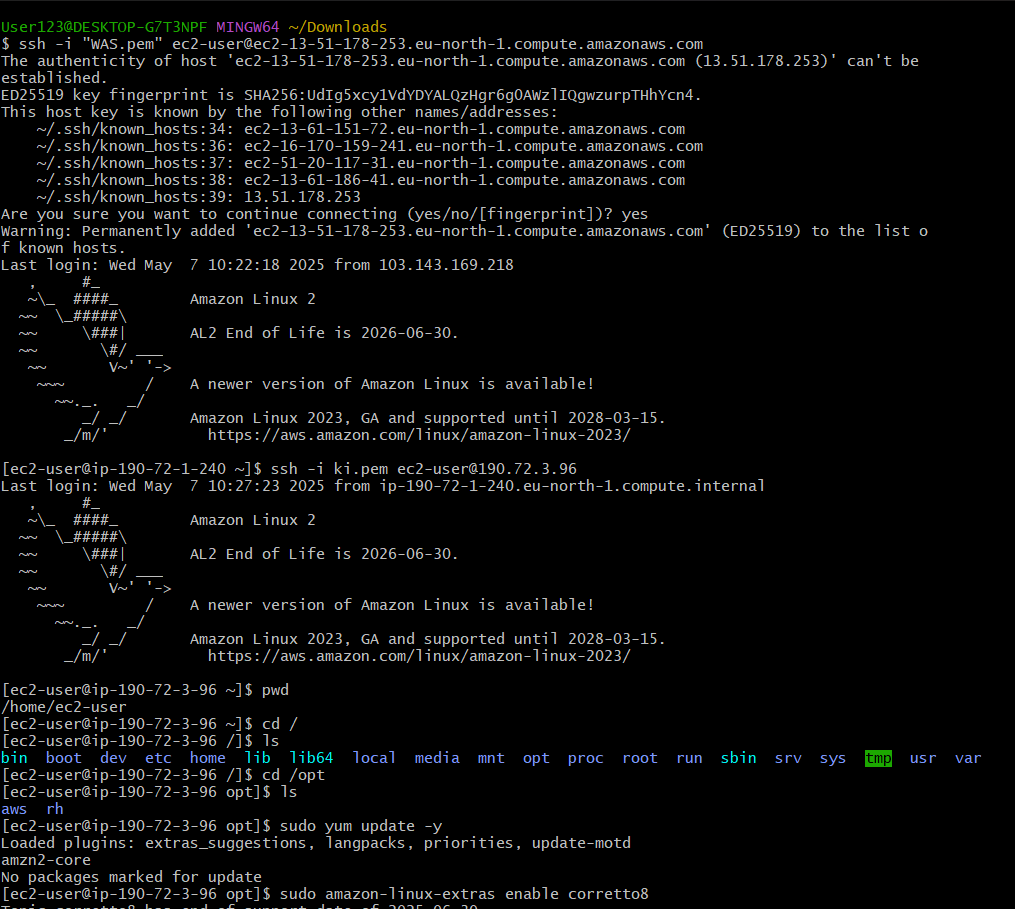
1. Launch Private EC2 using private subnet under your custom VPC, select an existing key pair
2. edit its security group,In Inbound Rules, add- Type: SSH ,Port: 22, Source: Security Group ID or Private IP of your Bastion Host( publicsubnet instance) not "Anywhere" or "My IP"
3. ssh into bastion
4. Open .pem file locally and copy
5. On Bastion Host, create a .pem and paste key
6. Set permission: chmod 400 key.pem
7. SSH From Bastion → Private EC2
8. Get the private IP of the private instance
9. **ssh -i key.pem ec2-user@<Private-EC2-IP>**
10. Confirm internet access using: **ping -c 3 google.com** (shows NAT Gateway is working properly)

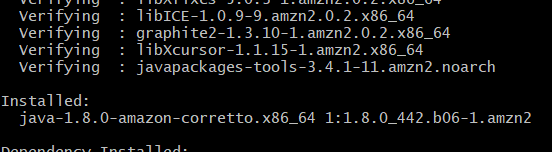


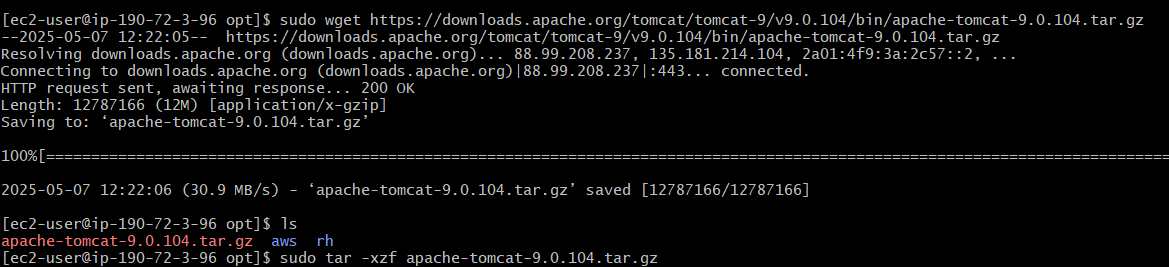


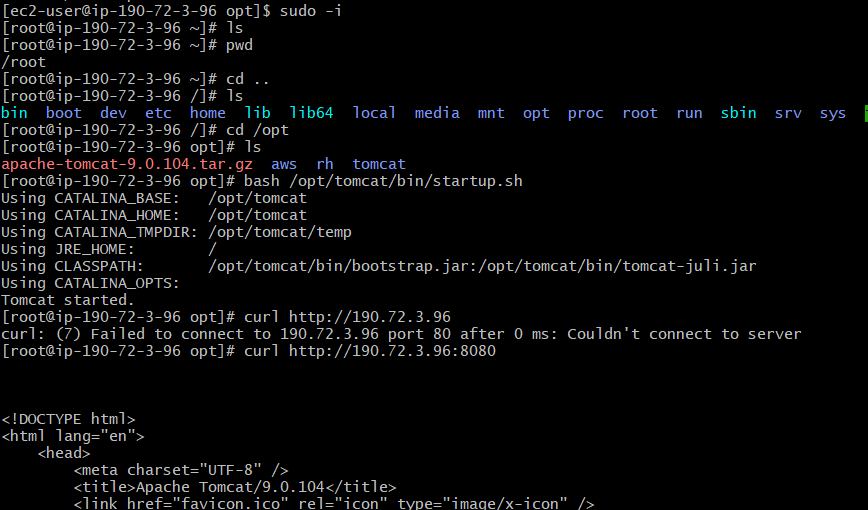
*9) Install Apache Tomcat in private ec2 and deploy a sample app.*

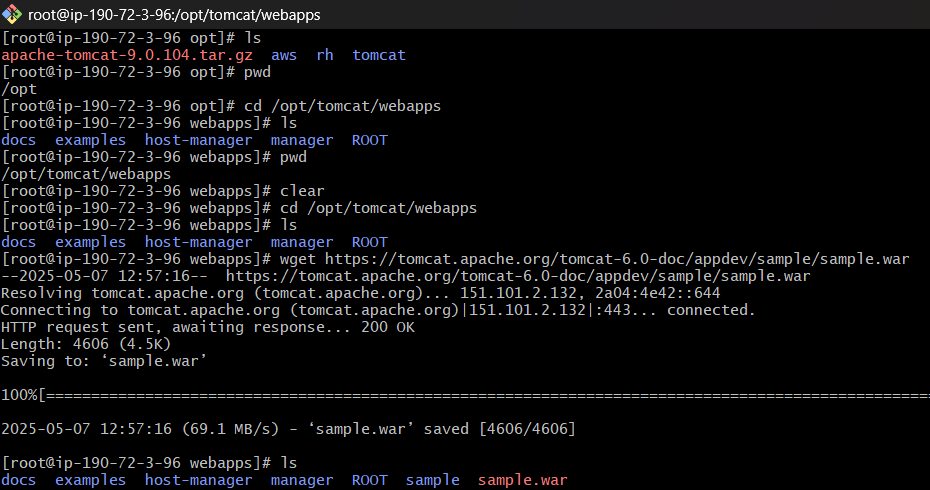
* SSH into the private instance from the bastion host
* make your self root user , switch /opt and install Tomcat the usual way
* Use absolute path to run the script **bash /opt/tomcat/bin/startup.sh**
* to check it here do **curl http://<private-ec2-private-ip>:8080**
* since it's in a private subnet, you cannot access it directly from a browser, we can expose tomcat via Load Balancer or reverse proxy but if its really necessary
* copy the link address of the sample.war and cd in to **/opt/tomcat/webapps/** and **wget <link>** there.





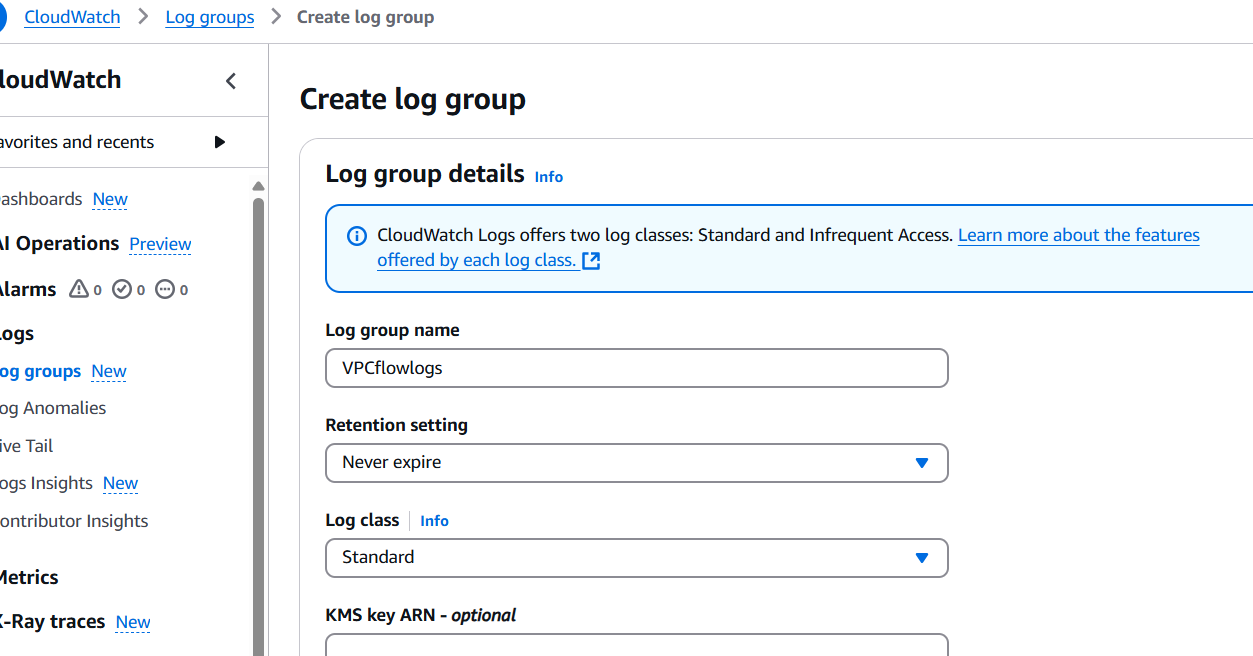


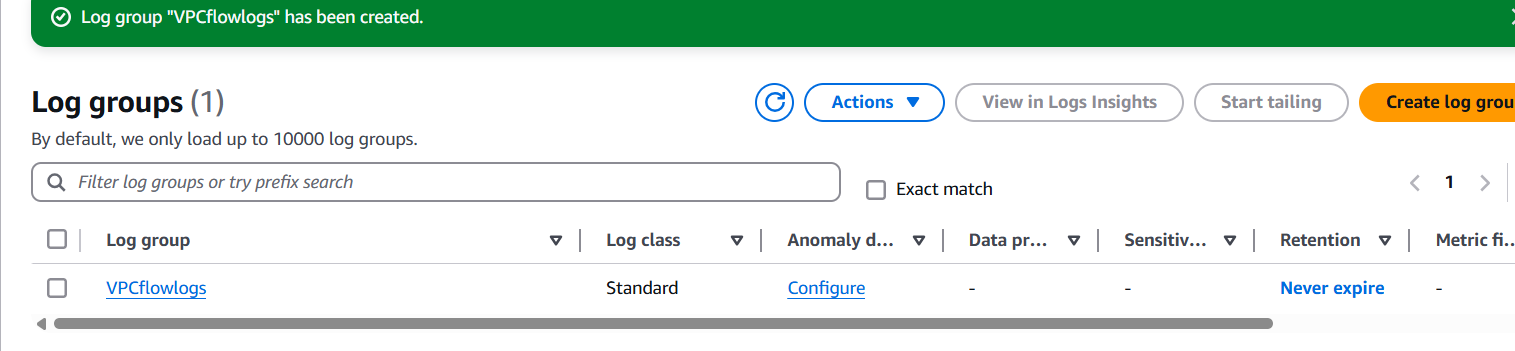




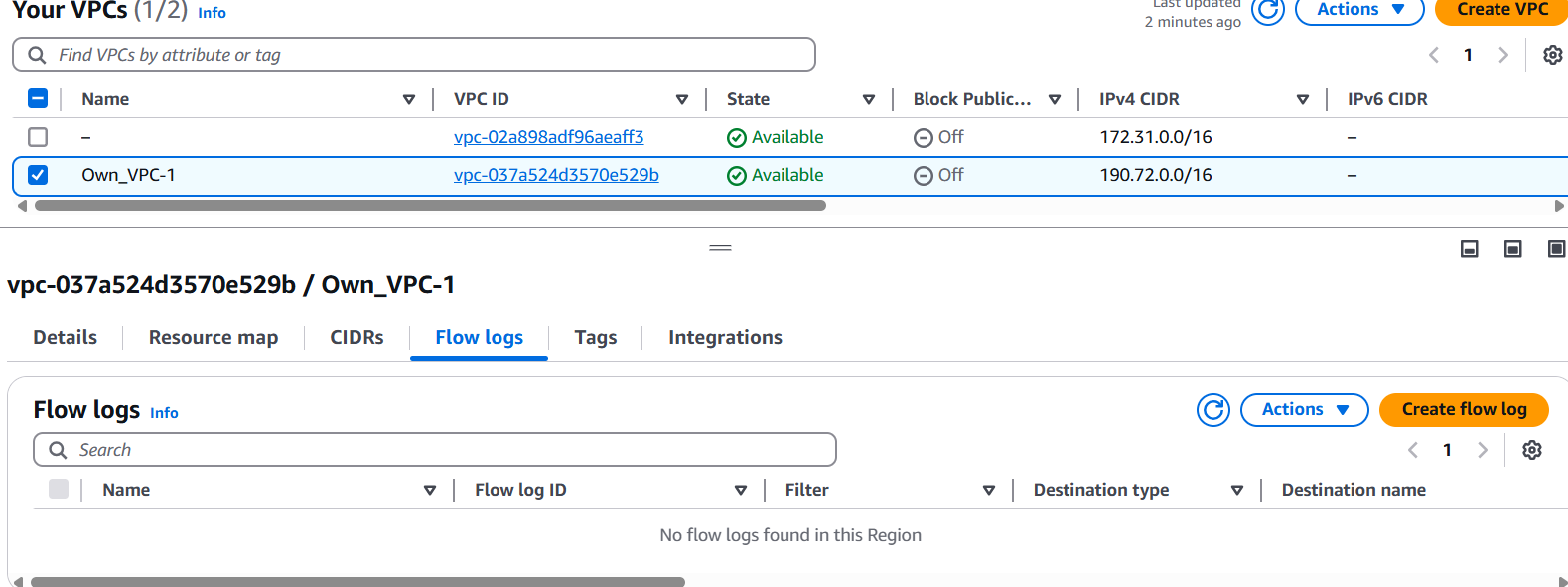
*10) COnfigure VPC flow logs and store the logs in s3 and cloudwatch.*

* Go to CloudWatch > Log Groups > Create Log Group . enter a name

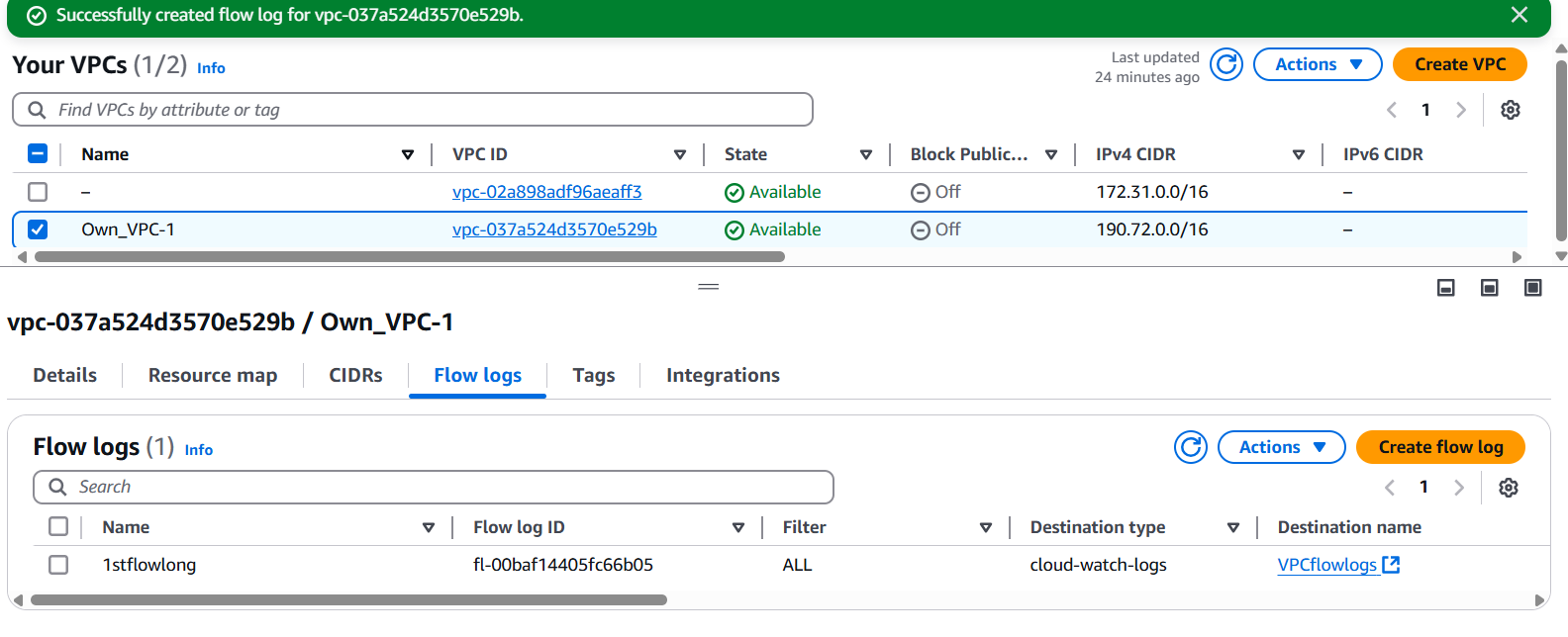


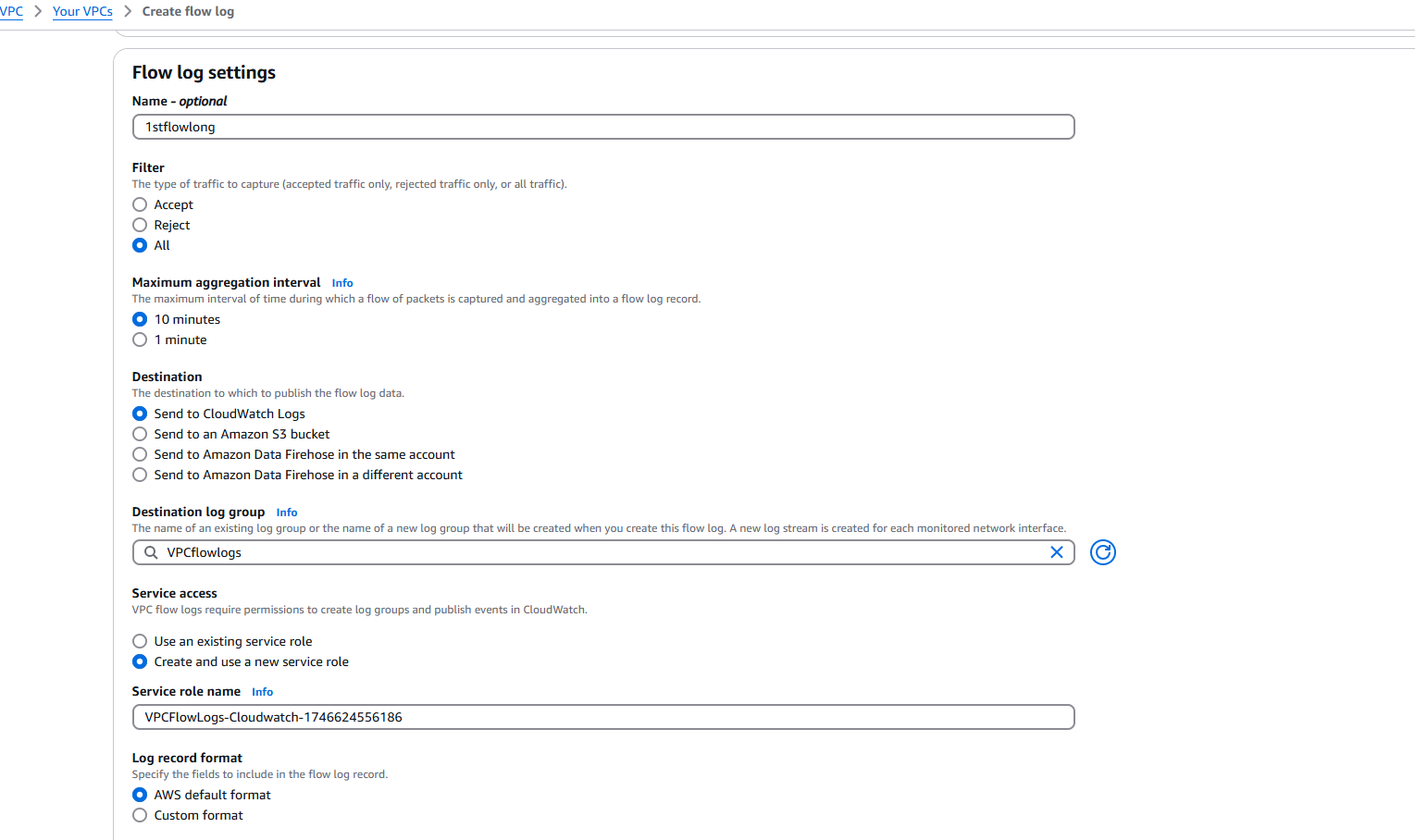


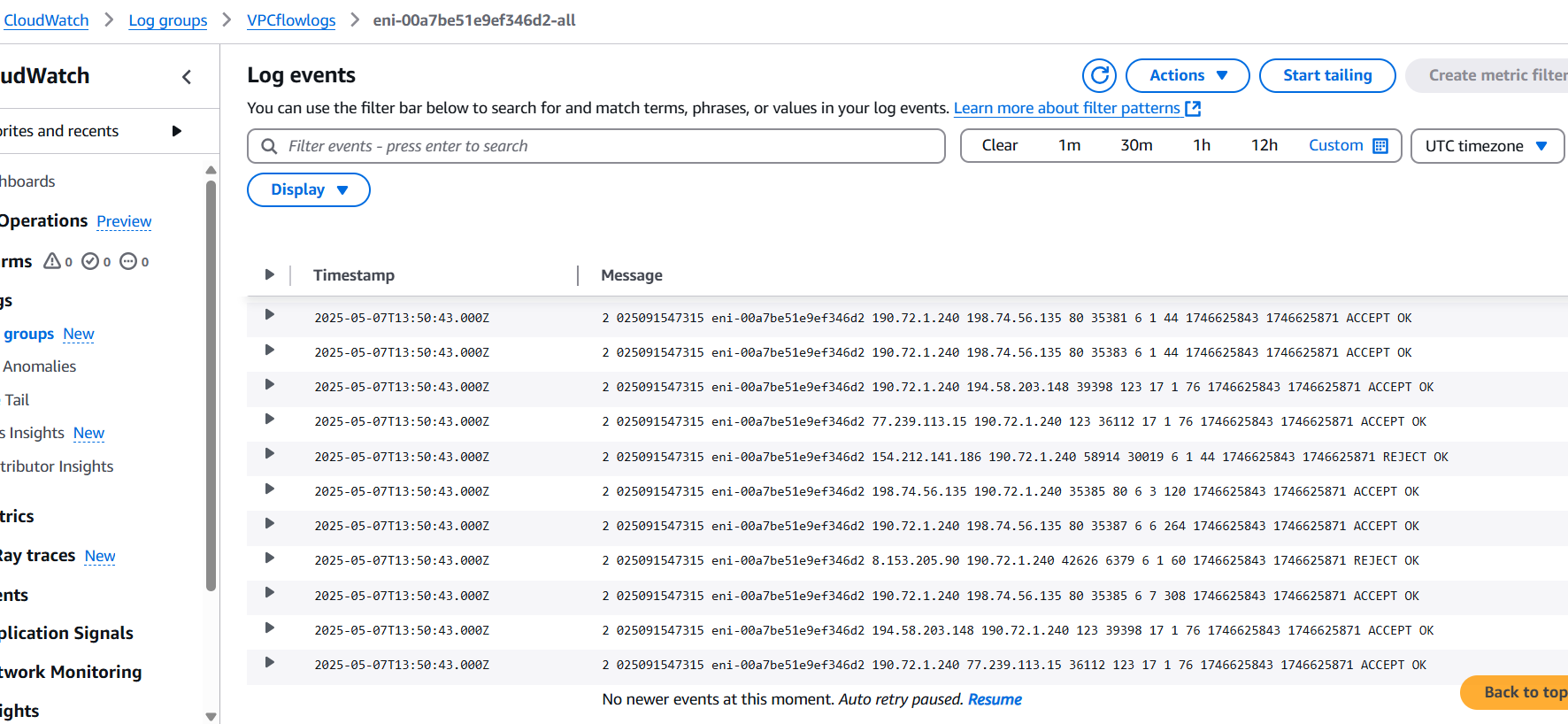
* Go to VPC > Your VPCs
* Select your VPC → Click Actions > Create Flow Log



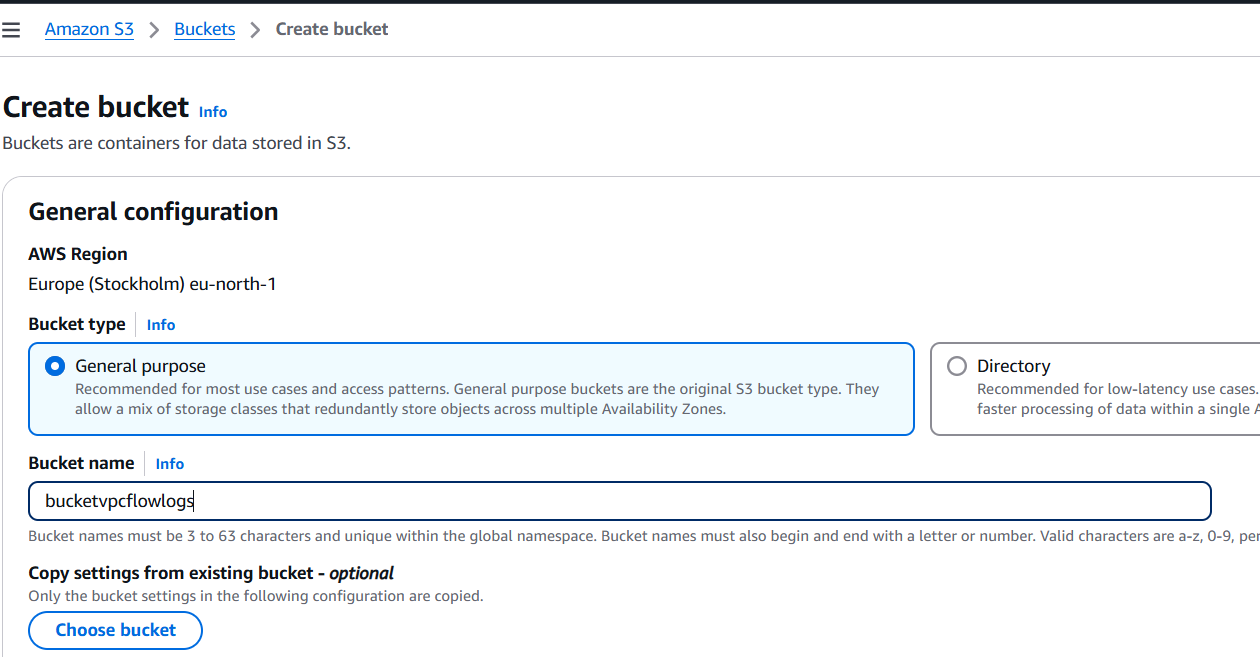
* Filter: All, Destination: Send to CloudWatch Logs, Log group: the name you gave earlier, create a use a new service role and click Create Flow Log
* To check the logs go to CloudWatch > Logs > Log Groups ><yougroup name>

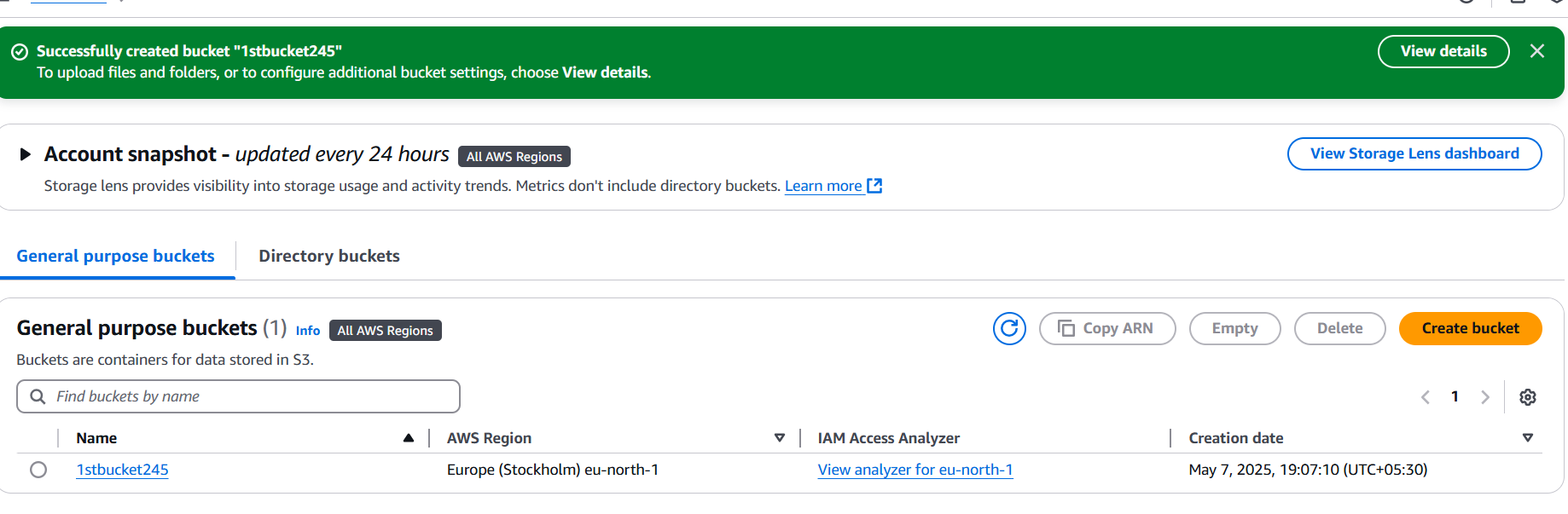






* Go to S3 > Create bucket and name, rest are settings on default
* Go to VPC > Your VPCs
* Select your VPC → Click Actions > Create Flow Log
* choose destination: Send to S3





* Enter Bucket ARN: arn:aws:s3:::<thenameyougave> and click Create Flow Log
* To check the logs go to S3 Console. Open your bucket and Navigate into the folders

