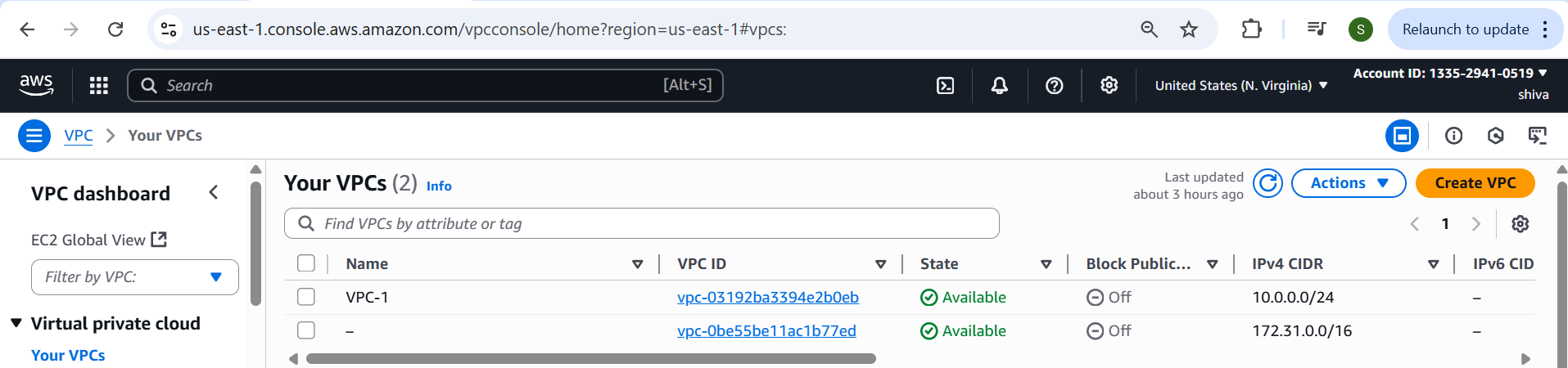
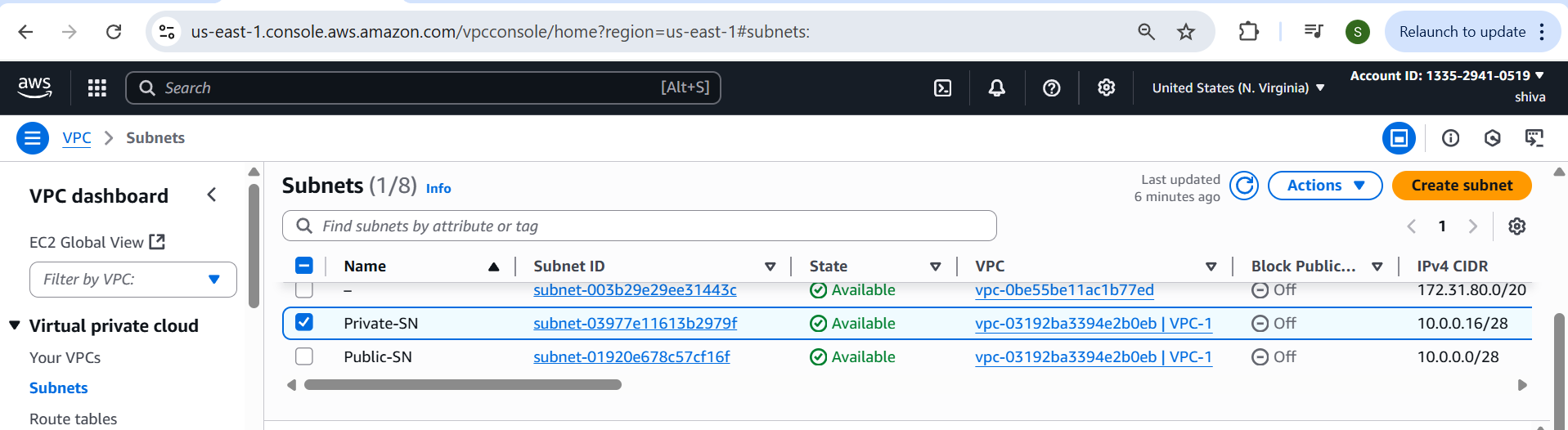
1. Create one VPC in N.virginia region.

Create one VPC in N.virginia region.  
 Step 1: Select N. Virginia Region  
1. Go to the AWS Management Console  
2. In the top-right corner, click on the region selector  
3. Select N. Virginia (us-east-1)

Click “Create VPC”  
Choose “VPC only” option

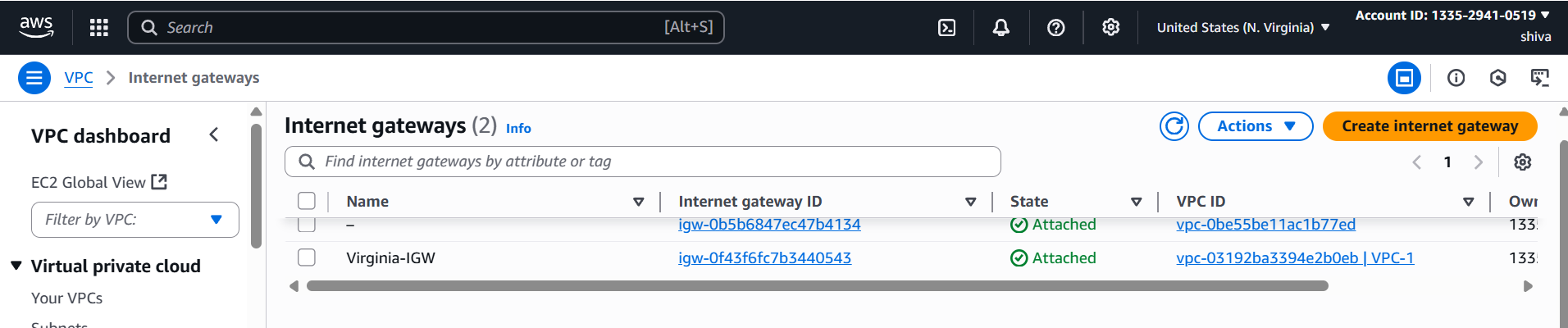


1. Create two subnets. One Public subnet and one private subnet.
2. Open AWS Console  
   2. Go to VPC → Subnets  
   3. Click “Create subnet”  
   Step 2: Create Public Subnet  
   1. VPC ID: Select your VPC (e.g., MyVPC)  
   2. Subnet name: public-subnet  
   3. Availability Zone: Choose one (e.g., us-east-1a)  
   4. IPv4 CIDR block: 10.0.0.0/28  
   5. Click Create subnet  
    Step 3: Create Private Subnet  
   1. Click “Create subnet” again  
   2. VPC ID: Same VPC  
   3. Subnet name: private-subnet  
   4. Availability Zone: Choose another or same (e.g., us-east-1b)  
   5. IPv4 CIDR block: 10.0.0.16/28  
   6. Click Create subnet  
    Step 4: Make the Public Subnet Public  
   1. Go to VPC → Route Tables  
   2. Find the main or custom route table associated with PublicSubnet  
   3. Edit routes:  
    Add route: 0.0.0.0/0 → target Internet Gateway  
   4. Go to Subnet Associations  
    Associate the route table with PublicSubnet  
   3. Provide the IGW to the vpc.



1. Provide the IGW to the vpc.

Navigate to VPC → Internet Gateways

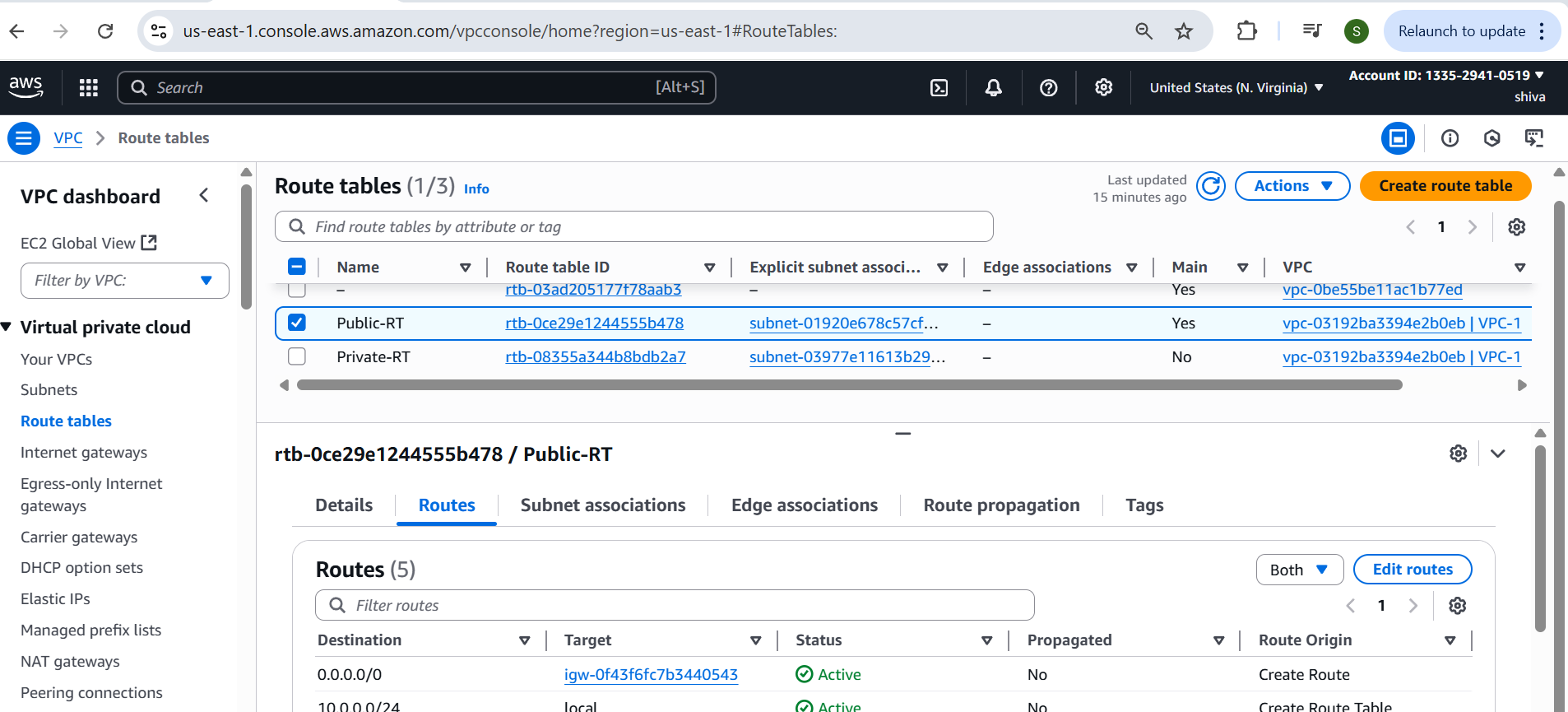
Click “Create internet gateway”  
2. Name tag: MyIGW (or any name)  
3. Click Create internet gateway  
 Step 3: Attach IGW to Your VPC  
1. After creating, click “Actions → Attach to VPC”  
2. Select your VPC (e.g., MyVPC)  
3. Click Attach internet gateway  


1. Create One public RT and one private RT.

Step 1: Go to VPC Dashboard  
1. Open AWS Console  
2. Navigate to VPC → Route Tables  
3. Click “Create route table”  
Step 2: Create Public Route Table  
1. Name tag: PublicRT  
2. VPC: Select your VPC  
3. Click Create route table

Step 3: Create Private Route Table  
1. Go back to Route Tables  
2. Click “Create route table”  
3. Name tag: PrivateRT  
4. VPC: Select the same VPC  
Click Create route  
Step 4: Edit Public Route Table for Internet Access  
1. Select PublicRT → Click “Routes” → Edit routes  
2. Add route:

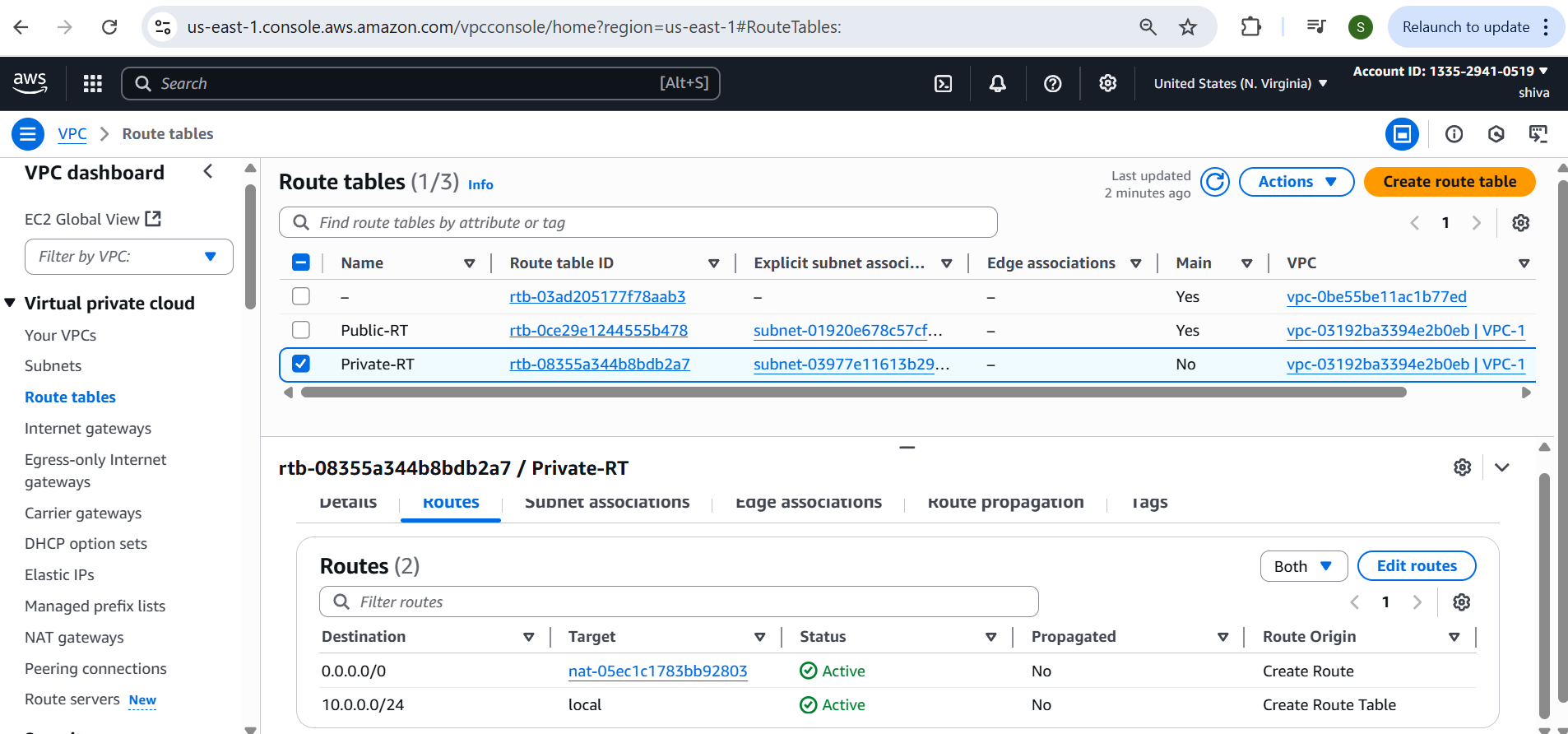
oDestination: 0.0.0.0/0  
oTarget: Select your Internet Gateway (IGW)



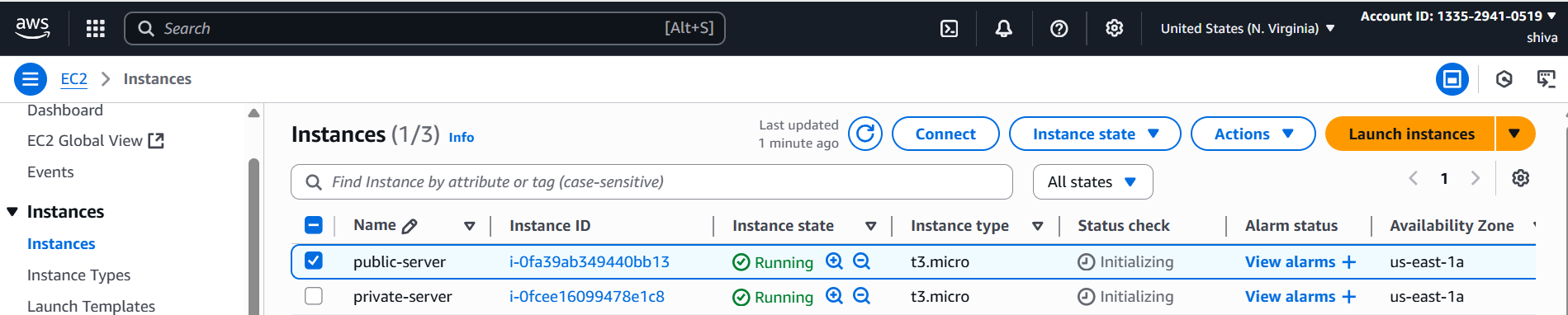
1. Deploy NAT gateway on public subnet and attach the NAT gatewat to private subnet.

Step 1: Allocate an Elastic IP  
1. Go to EC2 → Elastic IPs  
2. Click “Allocate Elastic IP address”  
3. Click Allocate  
 Step 2: Create NAT Gateway in Public Subnet  
1. Go to VPC → NAT Gateways  
2. Click “Create NAT Gateway”  
3. Name: MyNATGateway  
4. Subnet: Select your Public Subnet  
5. Elastic IP: Select the EIP you just allocated  
6. Click Create NAT Gateway

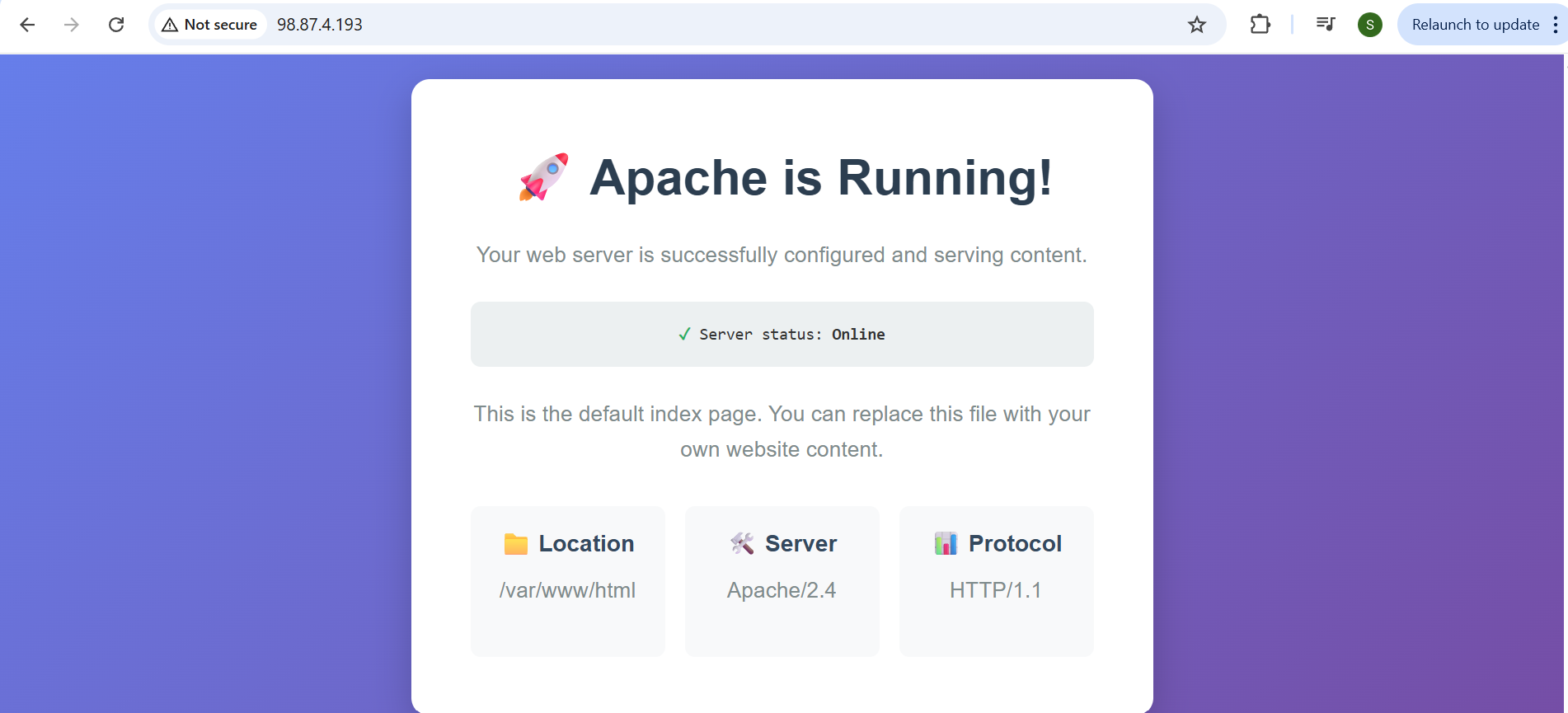
Step 3: Update Private Route Table  
1. Go to VPC → Route Tables  
2. Select your Private Route Table  
3. Click “Routes” → Edit routes  
4. Add a route:  
oDestination: 0.0.0.0/0  
oTarget: Select the NAT Gateway  
5. Click Save changes  
 Now, instances in your private subnet can access the internet via the NAT   
Gateway, while still remaining inaccessible from outside



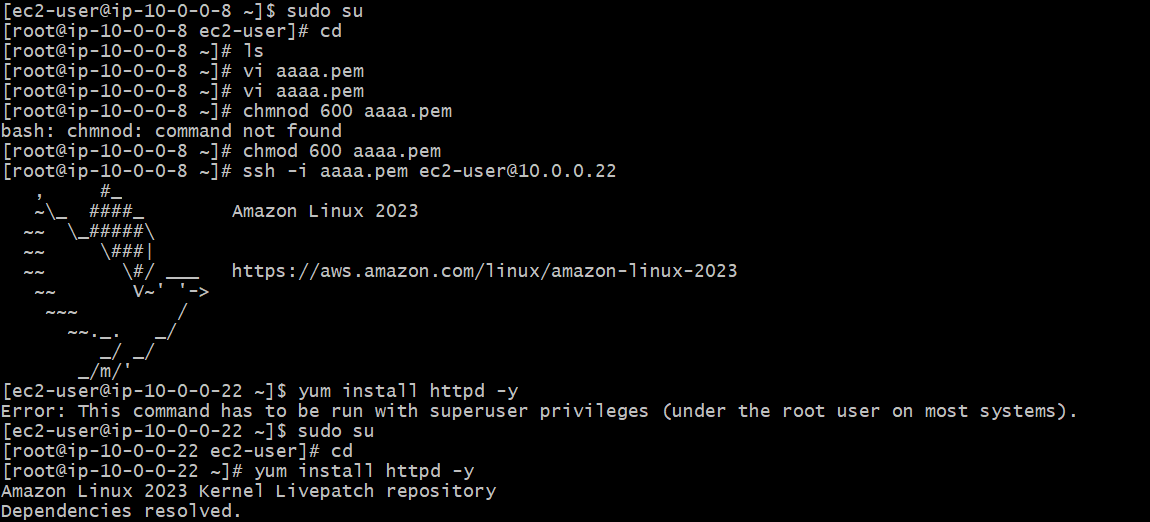
1. Create Two instances,one in public subnet and one in private subnet.



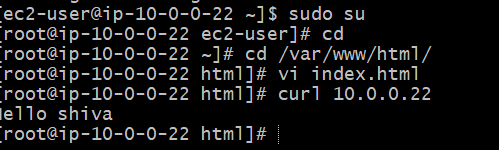
1. Deploy Apache server on both the ec2 instances with sample index.html file.



Private ec2



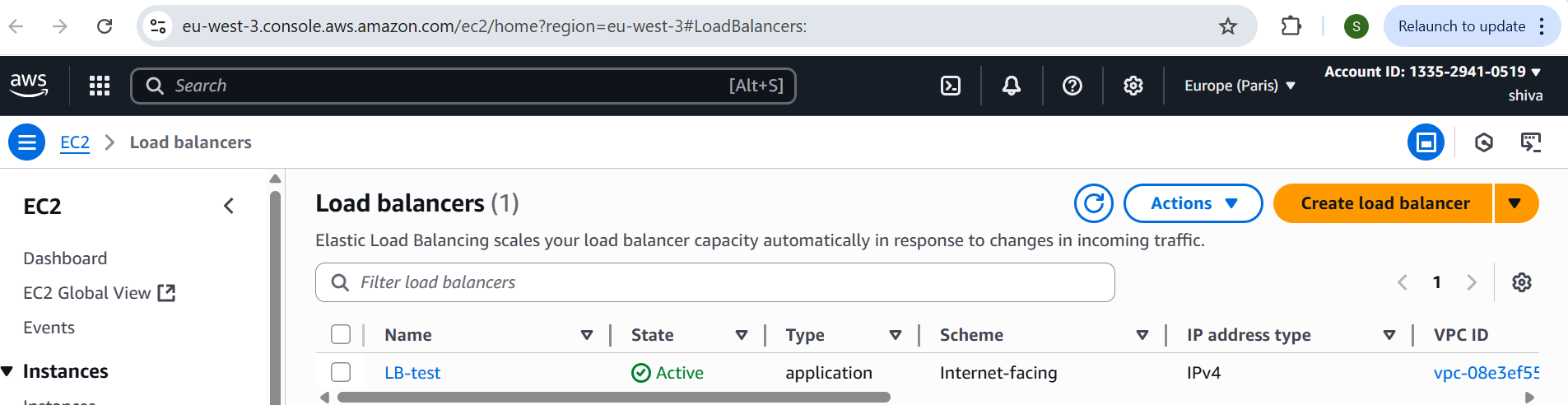


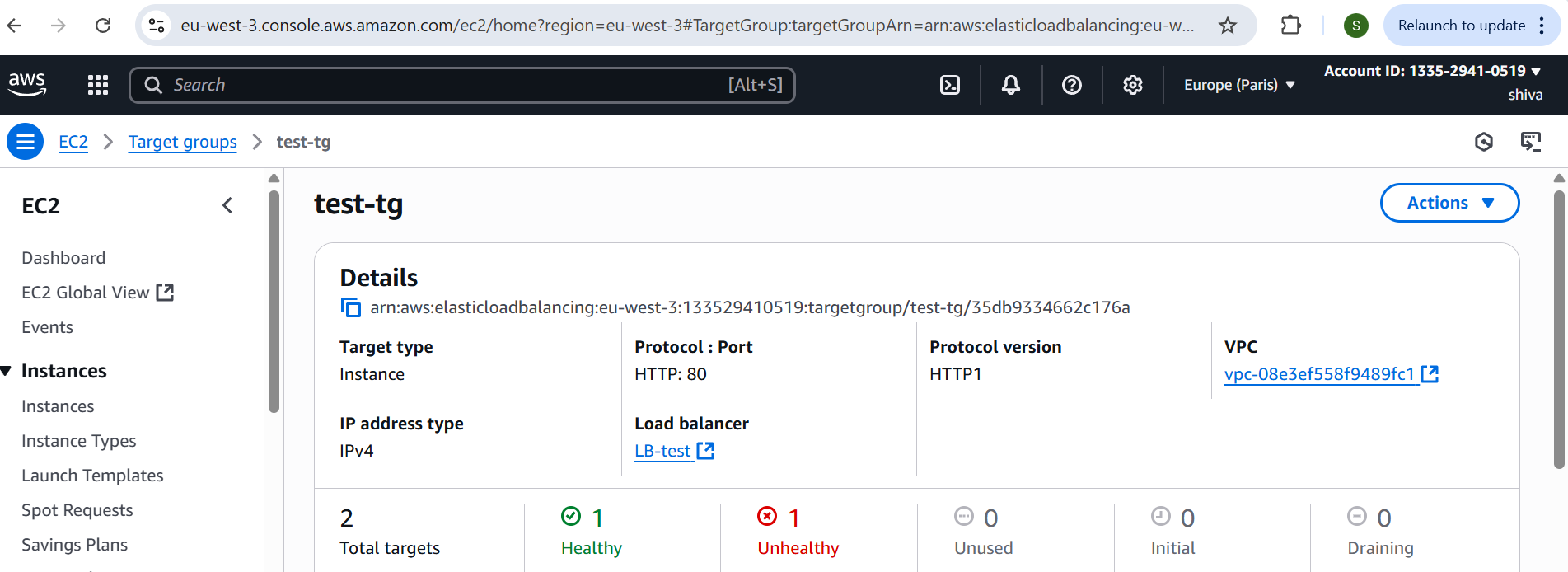


1. Create one application load balancer and attach the load balancer to both the ec2 instances.

Step 1: Go to EC2 Console → Load Balancers

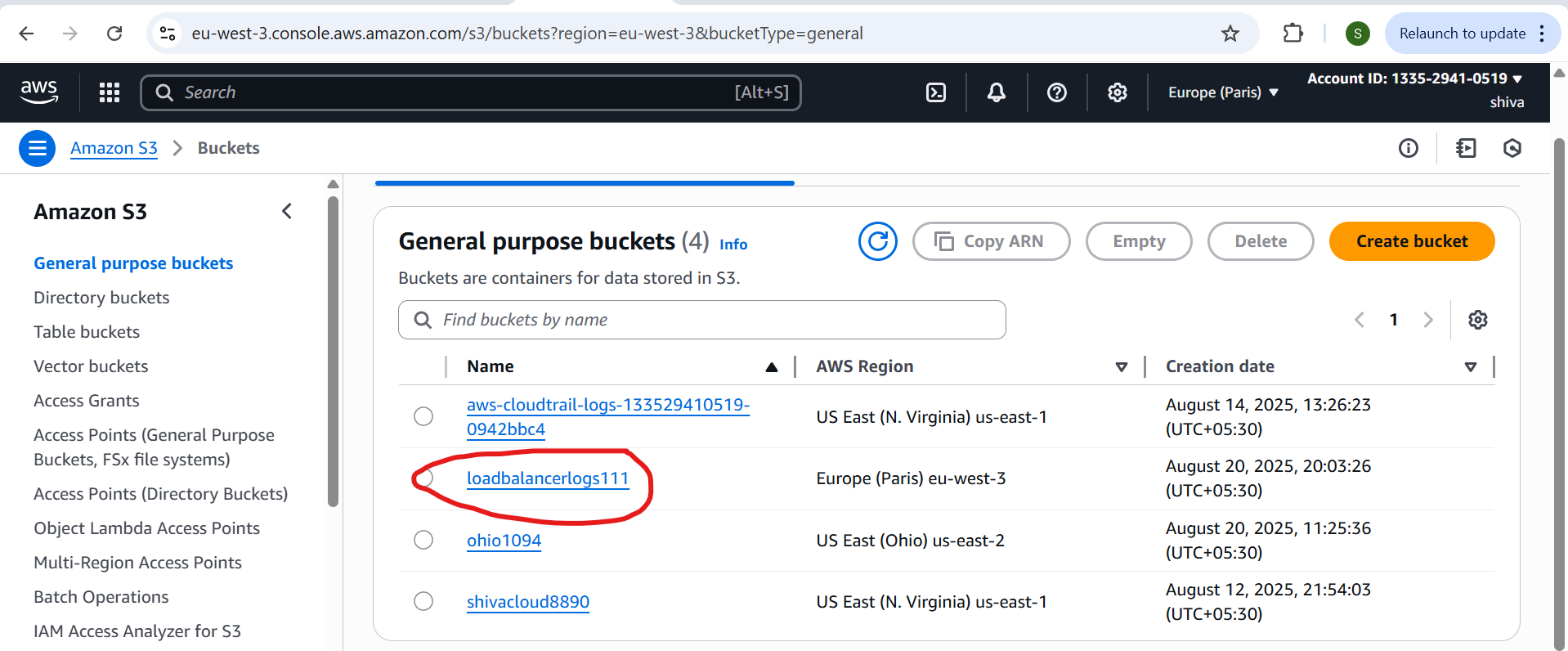
1. 1. Open AWS Console  
   2. Navigate to EC2 → Load Balancers  
   3. Click “Create Load Balancer”  
   4. Choose Application Load Balancer → Click Create  
   Step 2: Configure Load Balancer  
   1. Name: test-LB  
   2. Scheme: internet-facing  
   3. IP address type: IPv4  
   4. Listeners: HTTP (port 80) is default  
   5. Availability Zones:  
   oSelect your VPC  
   oCheck both subnets (public ones) for different AZs  
   Step 3: Configure Security Group  
   1. Select or create a Security Group that allows inbound HTTP (port   
   80)  
   Step 4: Configure Target Group  
   1. Target group name: test-tg  
   2. Target type: Instance  
   3. Protocol: HTTP | Port: 80  
   4. VPC: Choose your VPC  
   5. Click Next  
   Step 5: Register Targets (EC2 Instances)  
   1. Select both EC2 instances (public and private)  
   2. Click Add to registered  
   3. Click Create target group  
   Step 6: Review and Create ALB  
   1. Go back to ALB config  
   2. Select the target group you just created  
   3. Review and click Create load balancer  
   Your Application Load Balancer now distributes traffic to both EC2   
   instances. You can access it via the ALB's DNS name shown in the EC2 →   
   Load Balancer dashboard.





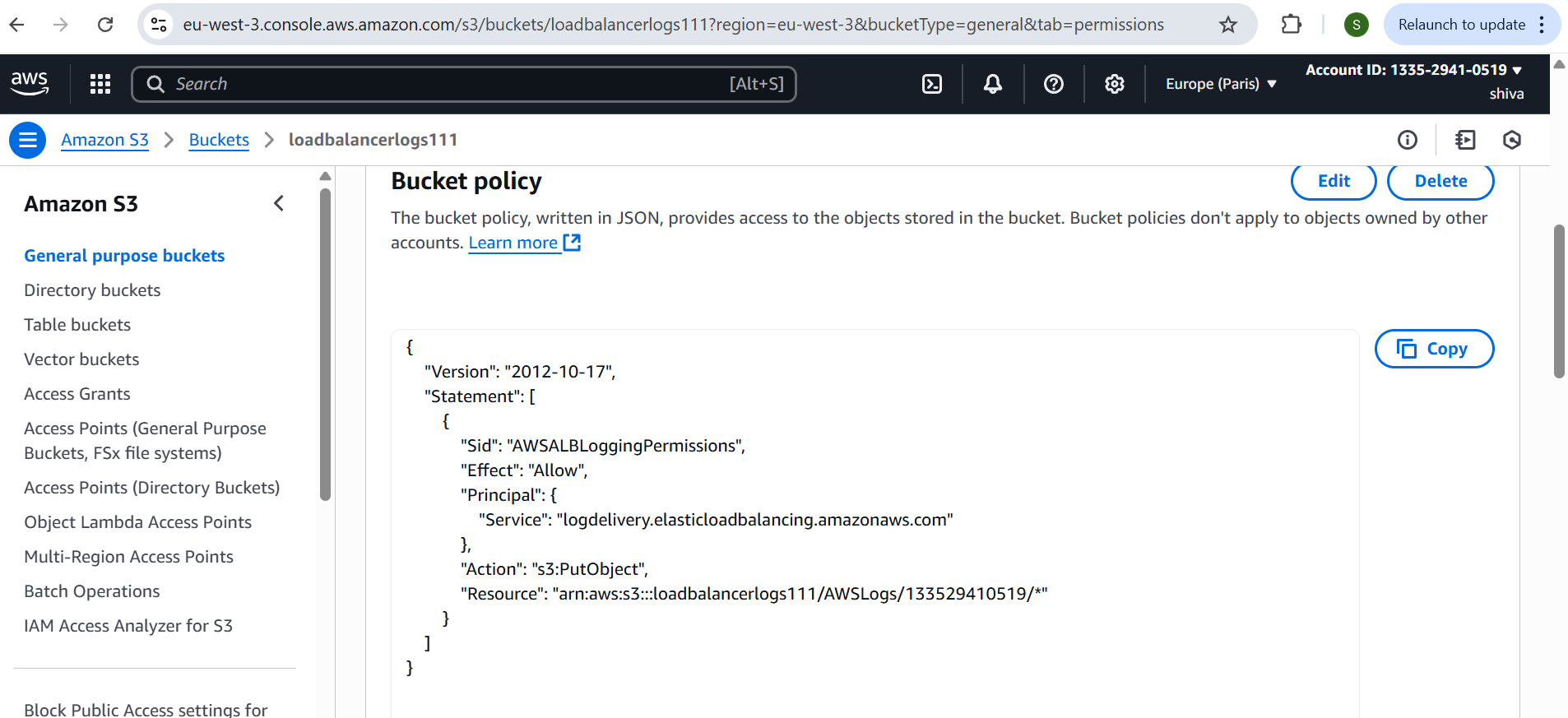
1. Store Application load balancer logs to s3.

STEP:1  
Create s3 bucket  
Amazon s3 ---> Buckets ---> click create buckets  
Select bucket type General purpose  
Enter bucket name  
Select ACLs enabled in object ownership  
Deselect block all public access  
Click create bucket



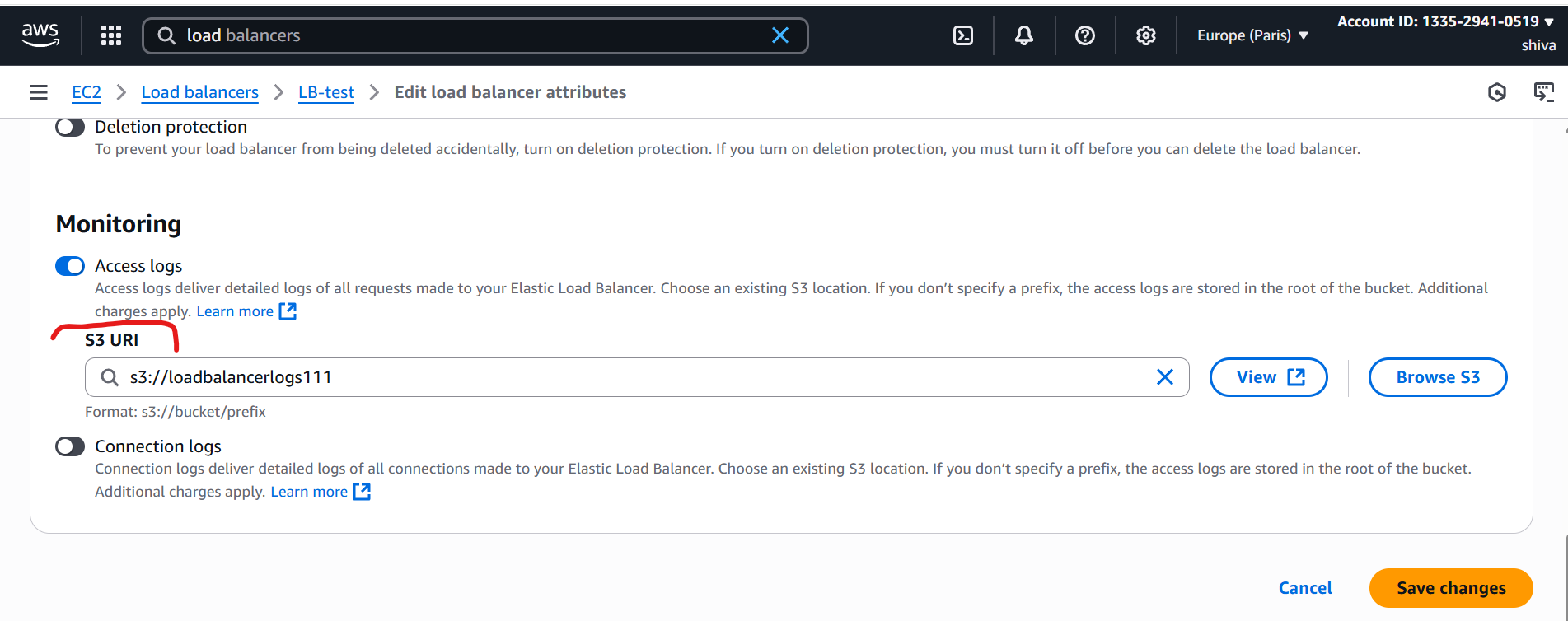
Add required bucket policy  
   
Here BUCKET-NAME: your bucket name  
 ACCOUNT-ID:your account Id  
{  
 "Version": "2012-10-17",  
 "Statement": [  
 {  
 "Sid": "AWSALBLoggingPermissions",

"Effect": "Allow",  
 "Principal": {  
 "Service": "logdelivery.elasticloadbalancing.amazonaws.com"  
 },  
 "Action": "s3:PutObject",  
 "Resource": "arn:aws:s3:::BUCKET-NAME/AWSLogs/ACCOUNT-ID/\*"  
 }  
 ]  
}  
This allows the ALB logging service to write logs into your bucket.  
Amazon s3 ---> Buckets ---> select your bucket ---> Permissions  
---> Edit bucket policy ---> click edit --->click save changes

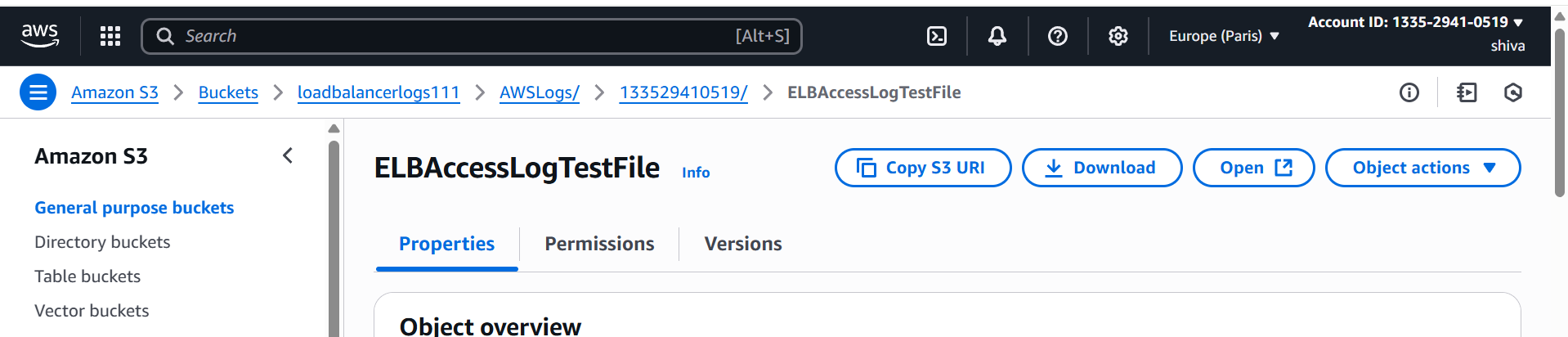


STEP:2  
Enable access logs on ALB  
EC2 ---> Load balancer ---> select your ALB ---> open Attributes tab

--->click edit ---> enable access log ---> add s3 url ---> click save changes

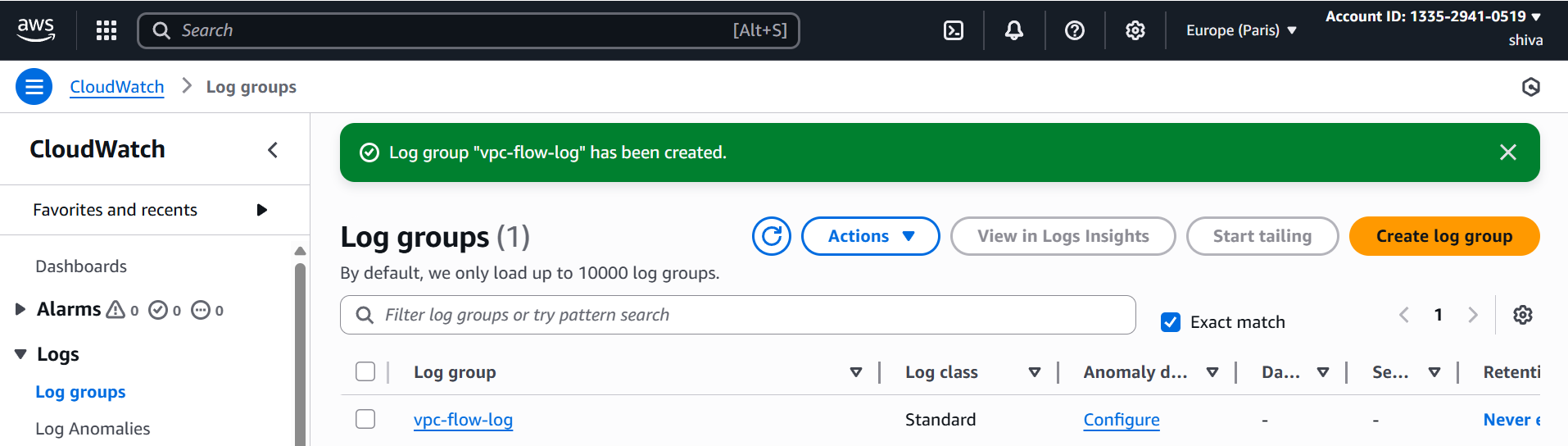


Go to check in s3 bucket to verify logs

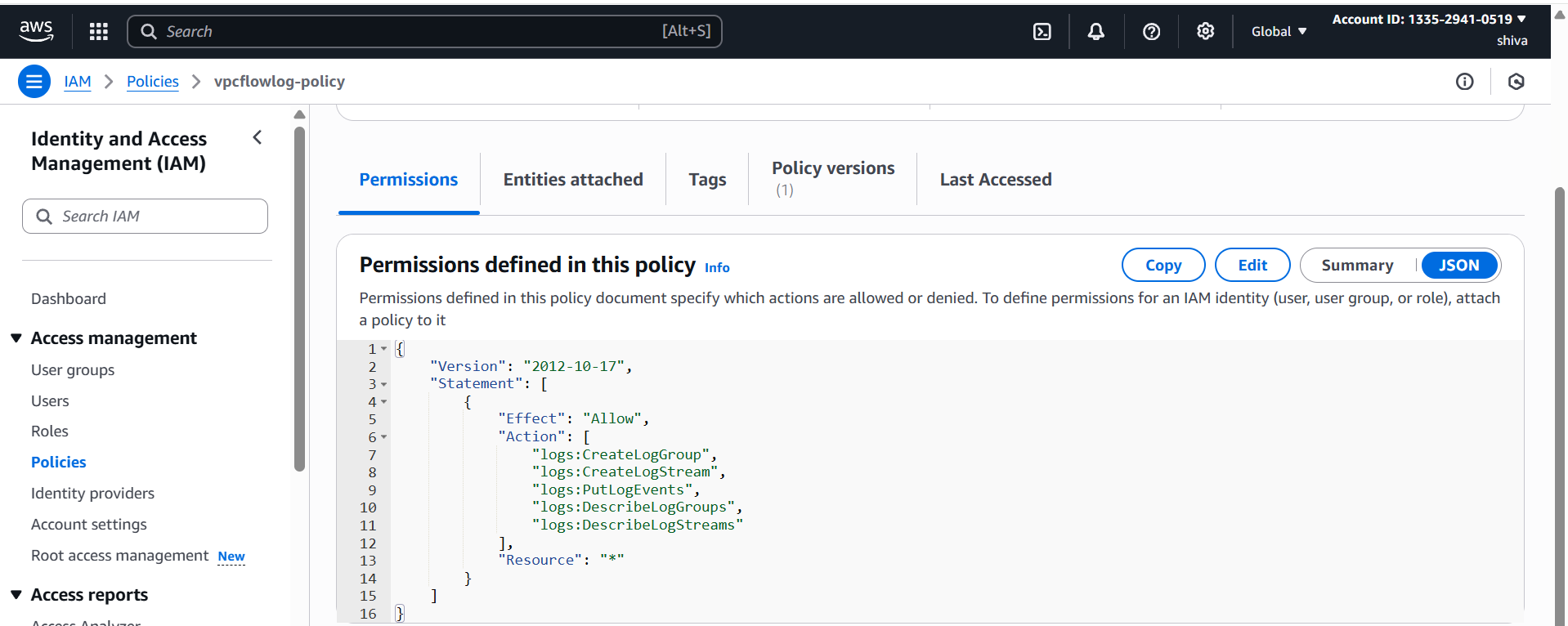


1. Store the VPC flow logs to CloudWatch group.

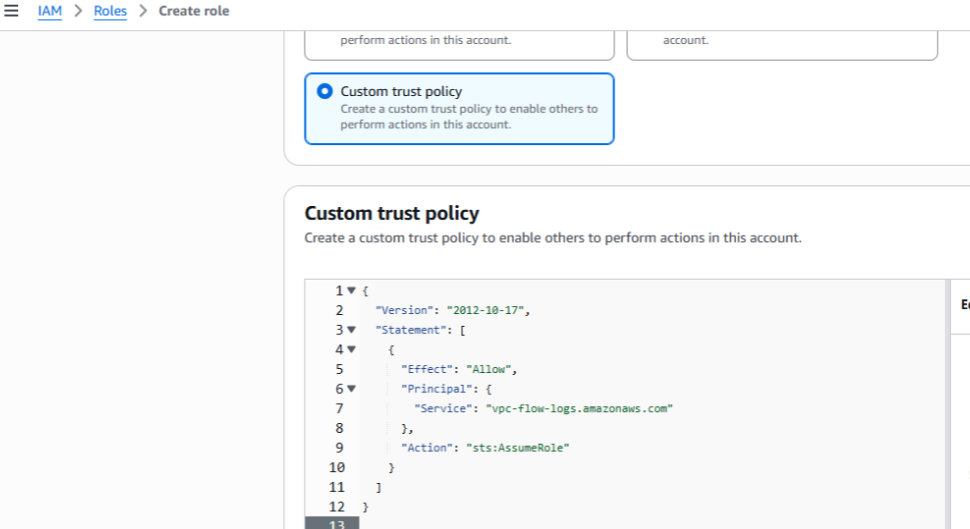
Cloudwatch ---> Logs ---> Log groups ---> Create Log Group



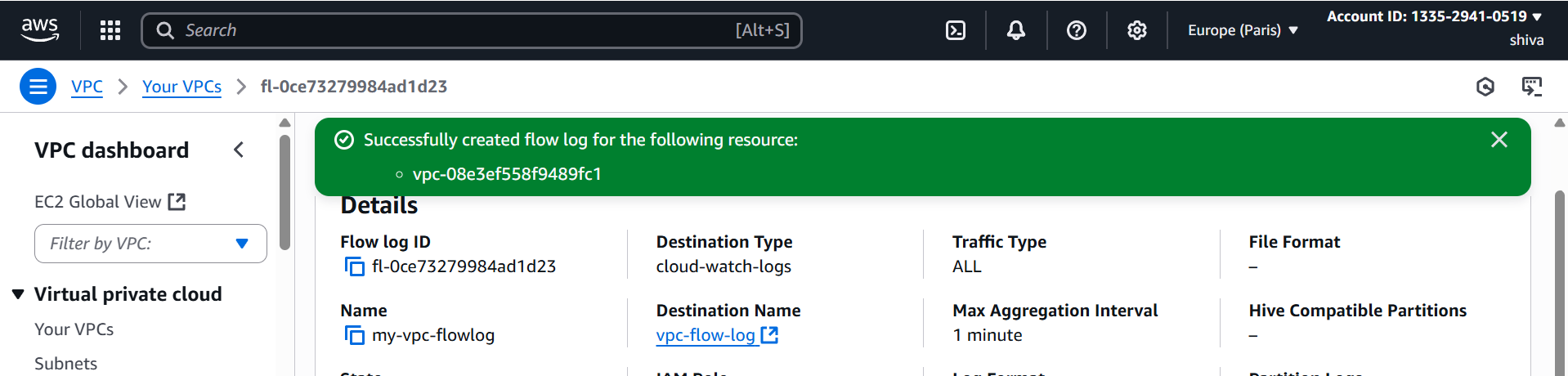
STEP:2  
IAM ---> Policies ---> Create policy --->Select policy editor in JSON format ---  
>



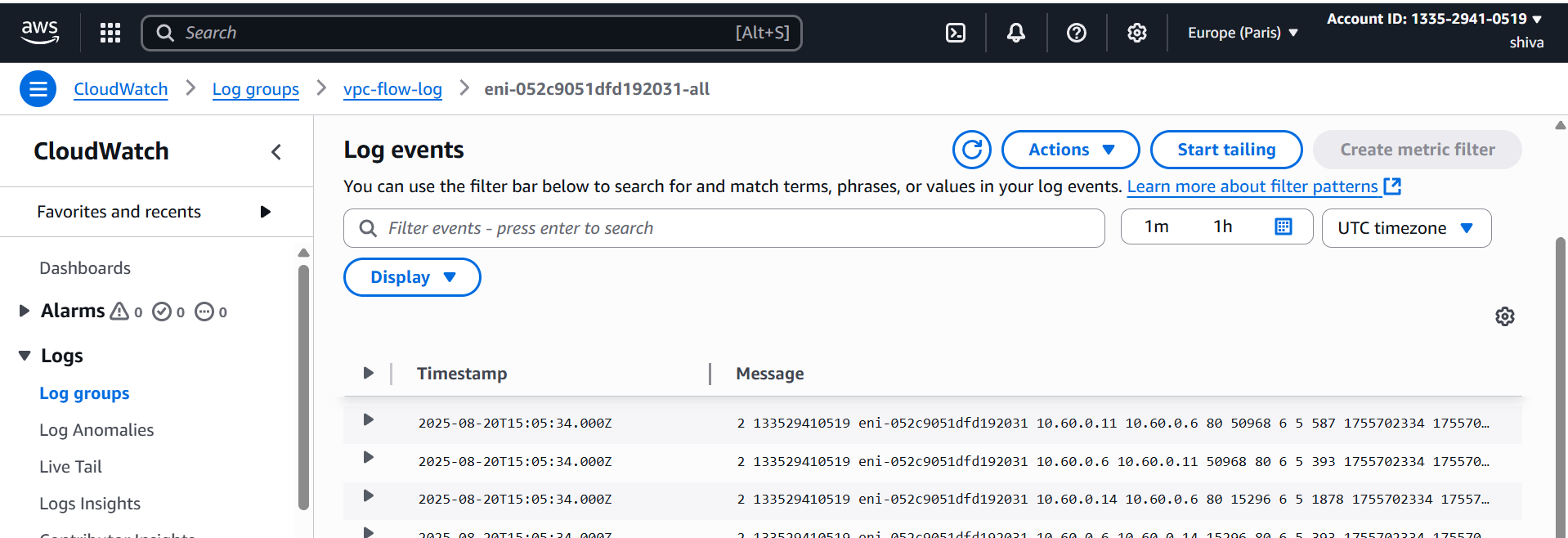
IAM ---> Roles ---> Create role ---> select custom trust policy ---> click Next



VPC ---> Your VPCs --->select your vpc ---> Flow Logs --->Create Flow log



CloudWatch ---> Log Groups ---> Select created log group ---> Select Log   
streams



1. Create Monitoring Dashboards to monitor CPU utilization and to monitor Apache service.

Step 1: Open CloudWatch Console  
1. Open AWS Console  
2. Navigate to CloudWatch → Dashboards  
3. Click Create dashboard  
4. Enter a Dashboard name   
5. Click Create dashboard

Step 2: Add CPU Utilization Widget  
1. Choose “Line” widget

1. Click Configure  
   3. Under Metrics, select:  
   EC2 → Per-Instance Metrics → CPUUtilization  
   Select your EC2 instance(s)

Now for apache write a script in EC2. vi apache-monitor.bash  
#!/bin/bash

REGION="eu-west-3"

METRIC\_NAME="ApacheRunning"

NAMESPACE="Custom/Apache"

/bin/systemctl is-active --quiet httpd

STATUS=$?

if [ "$STATUS" -eq 0 ]; then

VALUE=1

else

VALUE=0

fi

/usr/bin/aws cloudwatch put-metric-data \

--metric-name "$METRIC\_NAME" \

--namespace "$NAMESPACE" \

--value "$VALUE" \

--region "$REGION"

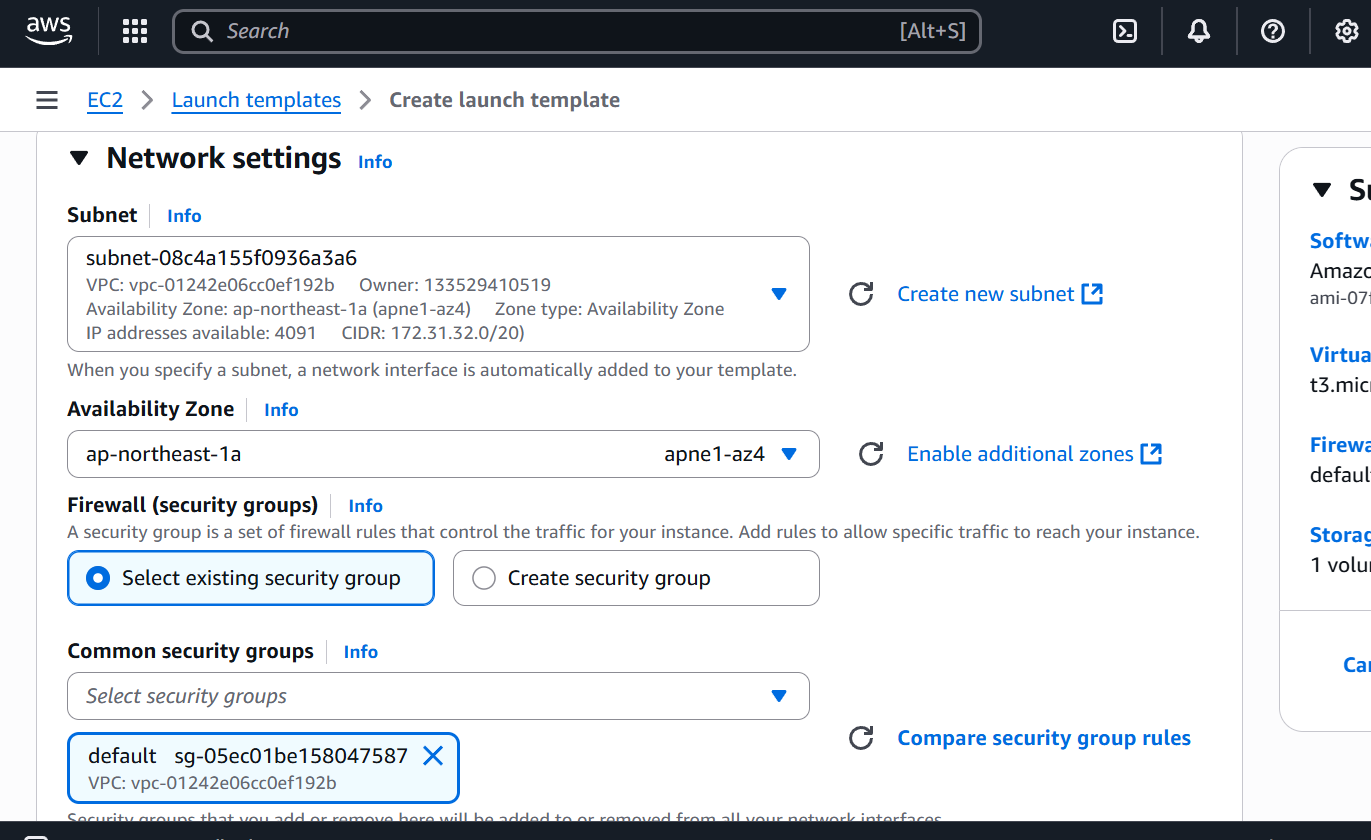
Give permissions:  
chmod +x apache-monitor.sh  
Install crontab then  
Crontab -e  
\*/1 \* \* \* \* /root/apache-monitor.sh  
./apache-monitor.bash

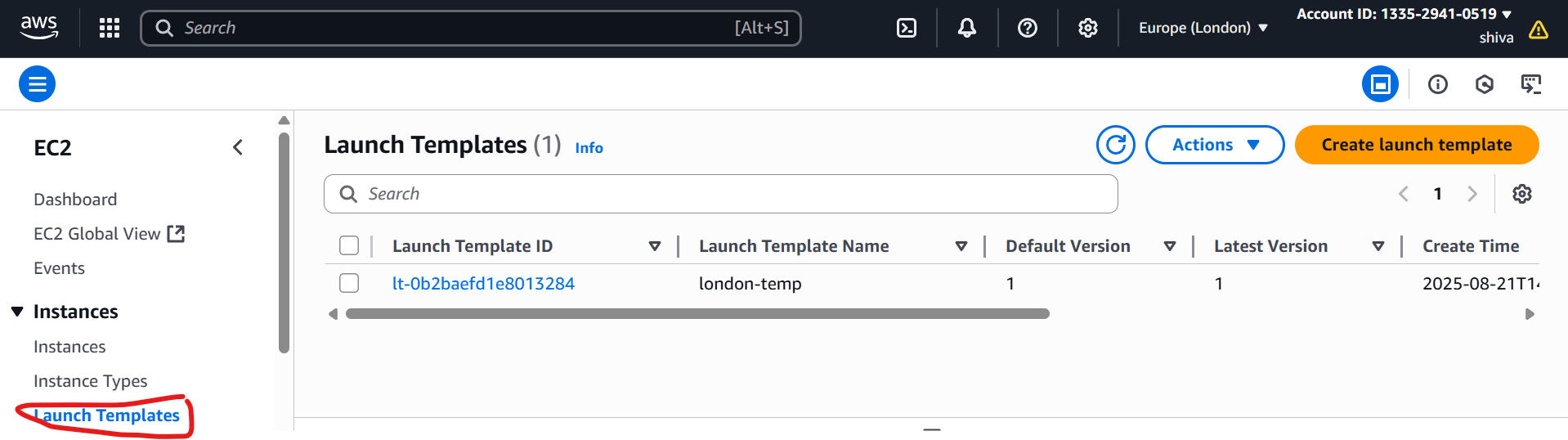
Add Apache Service Widget  
1. Back on the dashboard, click Add widget  
2. Choose “Number” or “Line” widget  
3. Select your custom Apache metric namespace  
4. Select metric representing Apache service status  
5. Click Create widget  
 Save and Review Dashboard  
•The dashboard now shows CPU usage and Apache service status



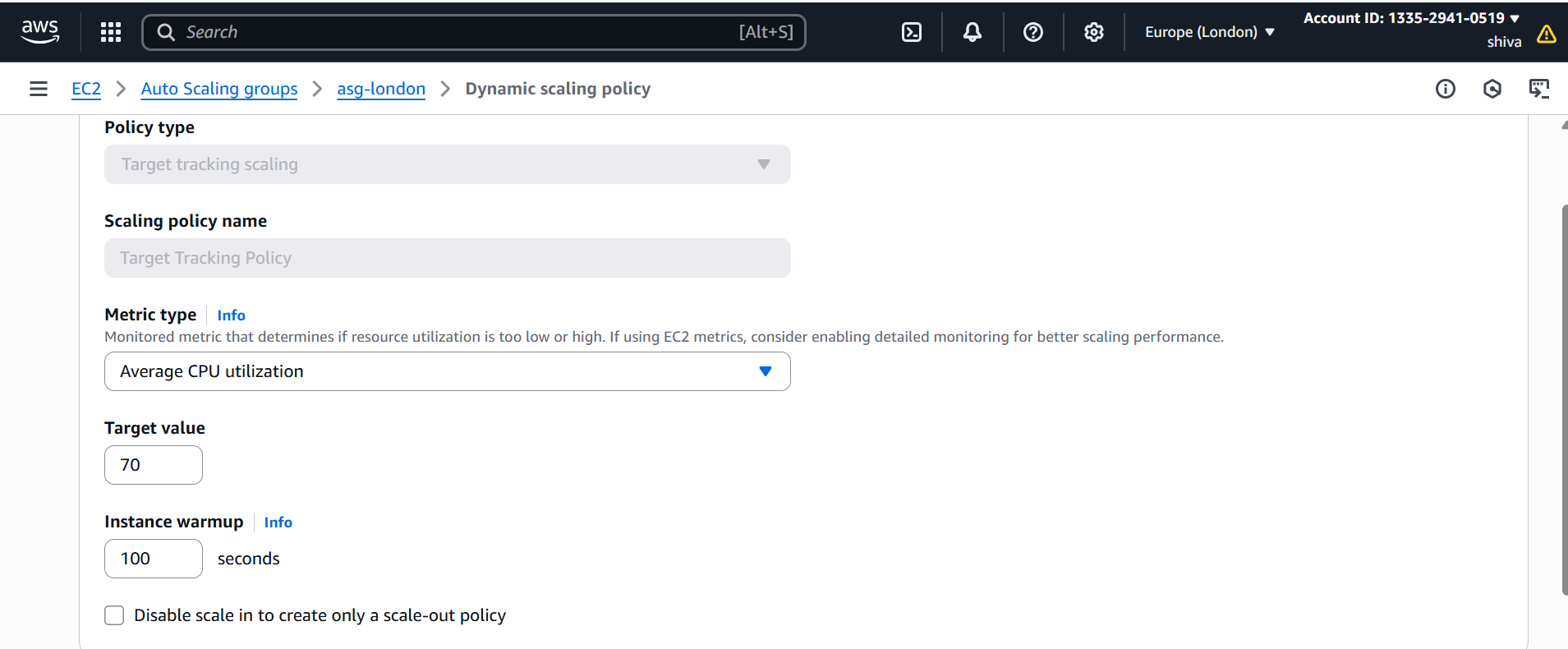
1. CPU utilizations more than 70% then it should triggered Autoscaling and launch new instance.

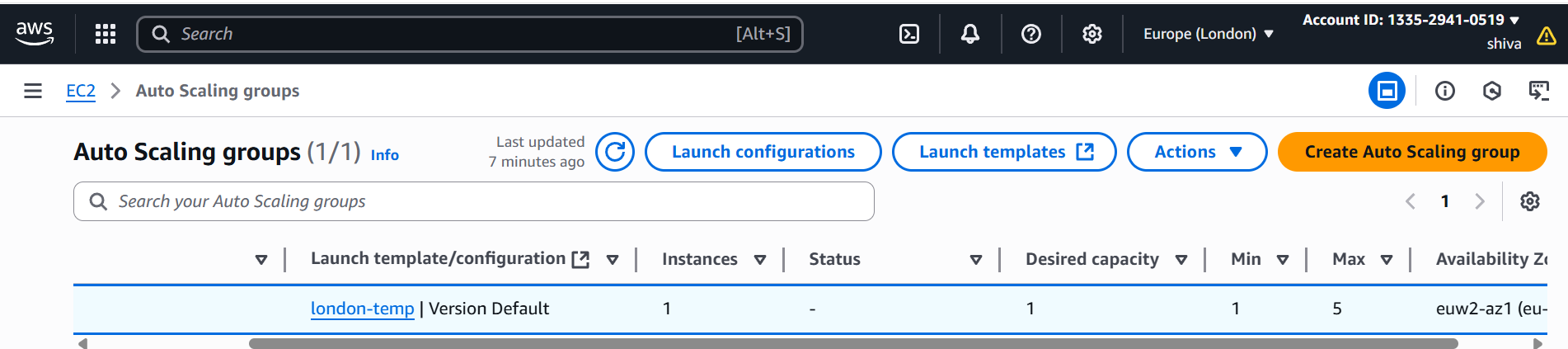
Create a template with EC2 configuration





Create autoscaling group with take or attach the template and in   
automatic scaling  
select Target Tracking Policy ---> metric type avg CPU   
utilization --->if Target value 70 ---> 100 seconds and then   
create





Once your EC2 reached CPU load above 70 automatically one EC2  
will be launched you can check EC2 Instances

