Project

CONSTRUCTING A CLOUD INFRASTRUCTURE USING AWS

2025

Prepared by

Pottipothu Divya

Table of Contents

1. Introduction:

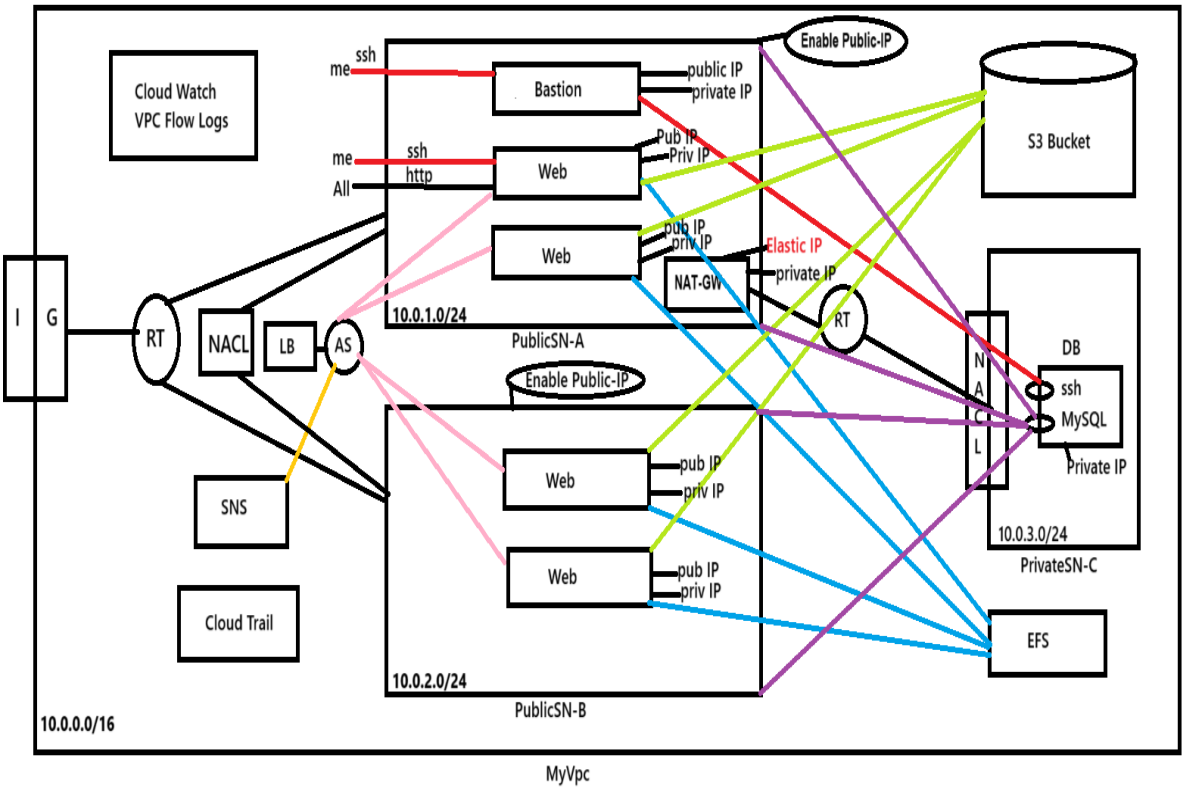
In modern businesses, cloud infrastructure is essential for scalability, security, and flexibility. Most companies leverage AWS services to build secure, scalable, and cost-effective solutions. This project focuses on building a Virtual Private Cloud (VPC) in AWS with various networking, security, and storage components, ensuring efficient network segmentation and internet accessibility.

2. Project Overview

The goal of this project is to set up a cloud-based infrastructure using AWS, where businesses can securely host applications. The setup includes:

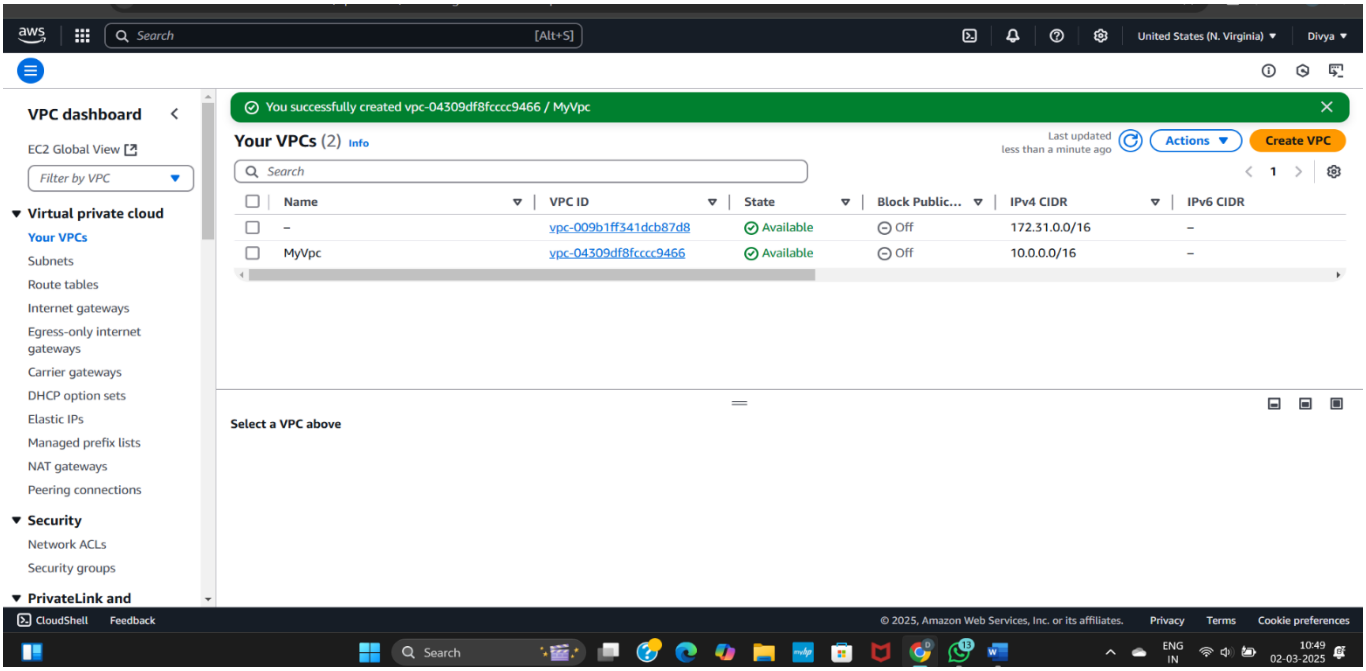
- **Networking & Security**
  - **VPC (Virtual Private Cloud)** with public and private subnets
  - **Internet Gateway (IG)** for external connectivity
  - **NAT Gateway** to enable internet access for private subnets
  - **Route Tables & Network ACLs (NACLs)** for traffic control
  - **Security Groups & IAM (Identity & Access Management)** for access control
- **Compute & Storage**
  - **EC2 (Elastic Compute Cloud) Instances** for scalable computing power
  - **Elastic Load Balancer (ELB)** for distributing incoming traffic
  - **Auto Scaling Group (ASG)** for high availability and fault tolerance
  - **Amazon S3** for object storage and backups
  - **Elastic File System (EFS)** for shared file storage
- **Monitoring & Logging**
  - **Amazon CloudWatch** for real-time monitoring and alerts
  - **AWS CloudTrail** for auditing API calls and activity tracking
  - **VPC Flow Logs** for analyzing network traffic patterns
- **Event-Driven & Messaging Services**
  - **Simple Notification Service (SNS)** for automated notifications

3. Cloud Architecture Diagram



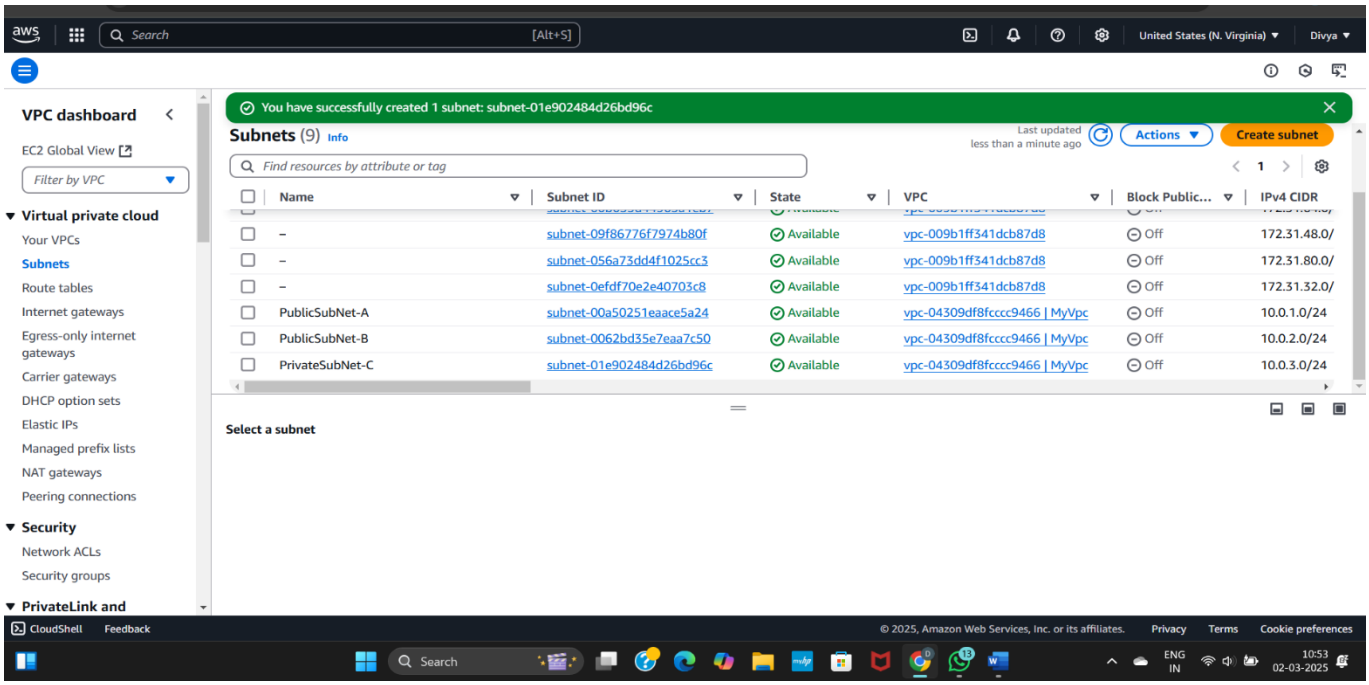
4. Creating a Virtual Private Cloud (VPC)

A VPC (MyVpc) is created in North Virginia with a CIDR block of **10.0.0.0/16**, which provides a secure and scalable cloud network. In this setup, subnets, a route table, and an internet gateway are automatically generated, simplifying network configuration. This ensures efficient resource management, controlled communication, and enhanced security for cloud deployments.



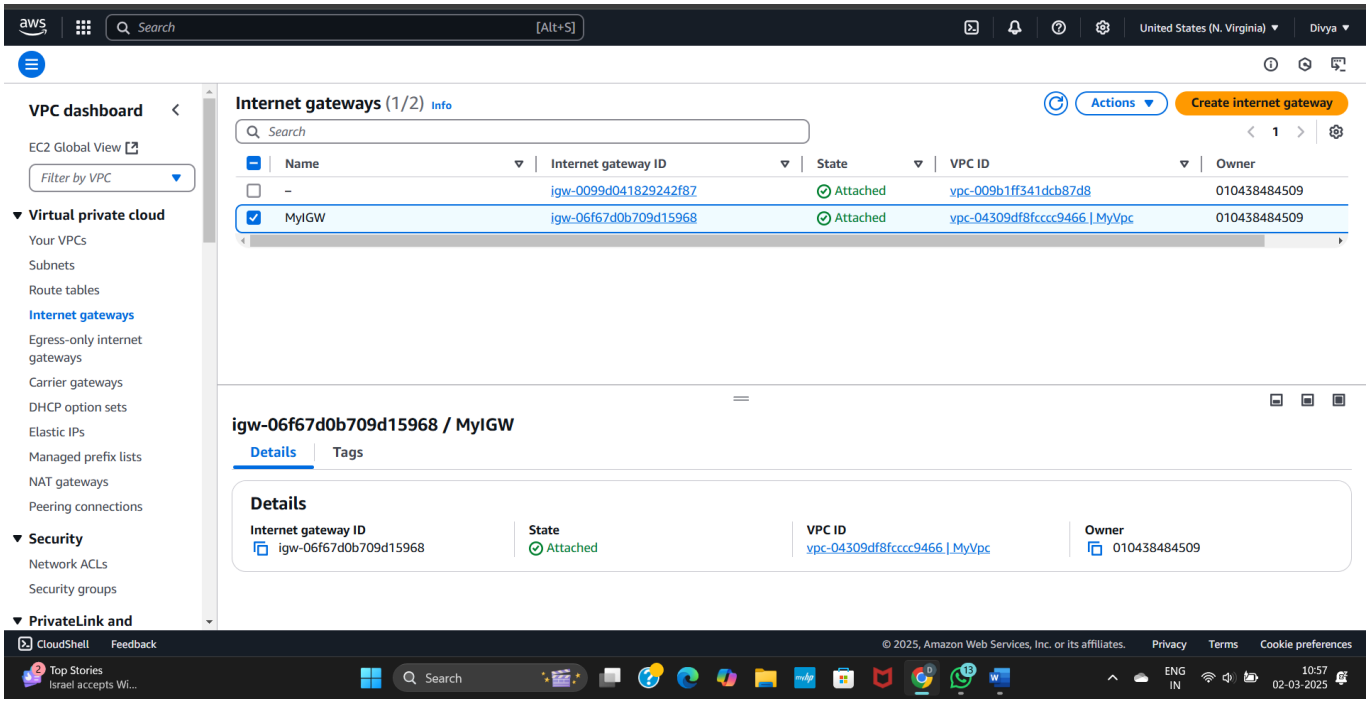
5.Setting Up Public and Private Subnets in VPC(MyVpc)

Three subnets are created: **PublicSubnet-A (10.0.1.0/24), PublicSubnet-B (10.0.2.0/24), and PrivateSubnet-C (10.0.3.0/24)**. By default, all AWS subnets are private. To enable public access, **PublicSubnet-A** and **PublicSubnet-B** are configured to assign public IPs to instances. The instances launched in these subnets retain the same public IP, ensuring consistent internet connectivity, while **PrivateSubnet-C** remains isolated for internal workloads.



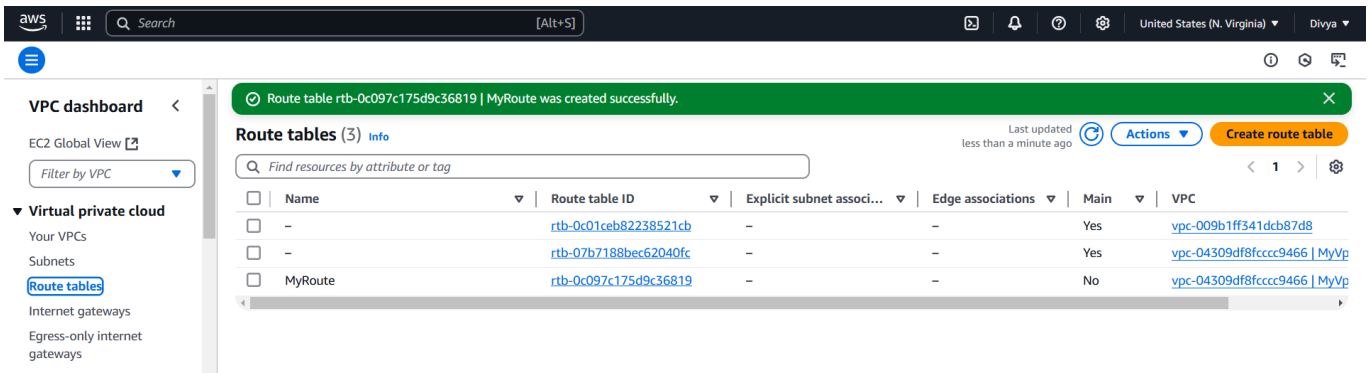
6.Configuring Internet Gateway for Public Access

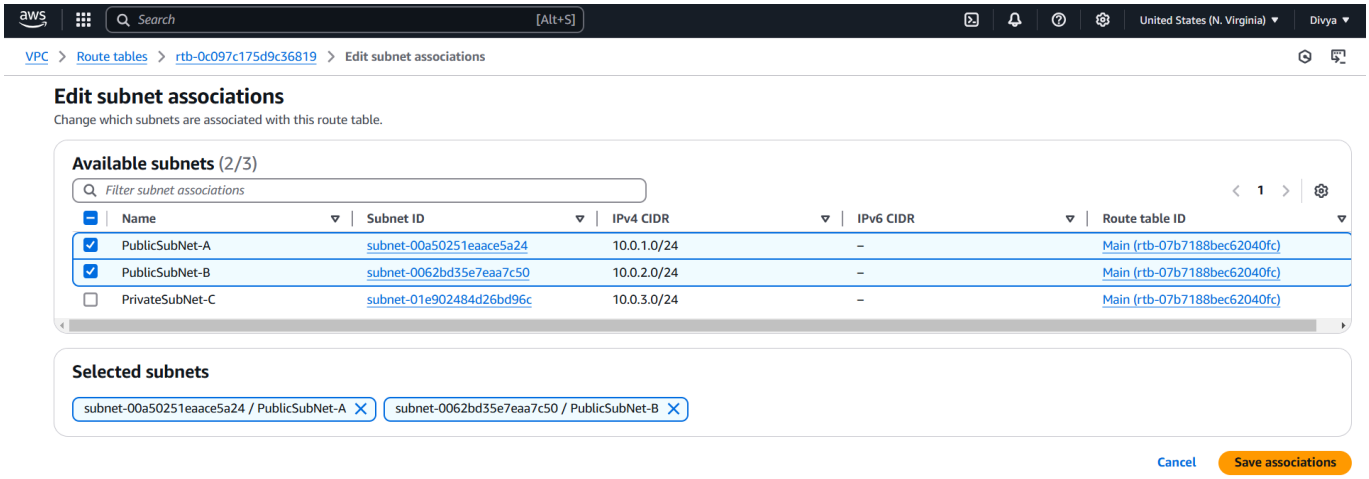
An **Internet Gateway (MyIGW)** is created and attached to the **VPC (MyVpc)** to enable public internet access. It allows resources in **PublicSubnet-A** and **PublicSubnet-B** to communicate with the internet while ensuring a secure network architecture. The **MyIGW** facilitates seamless inbound and outbound traffic for publicly accessible instances, enabling them to send and receive data over the internet.

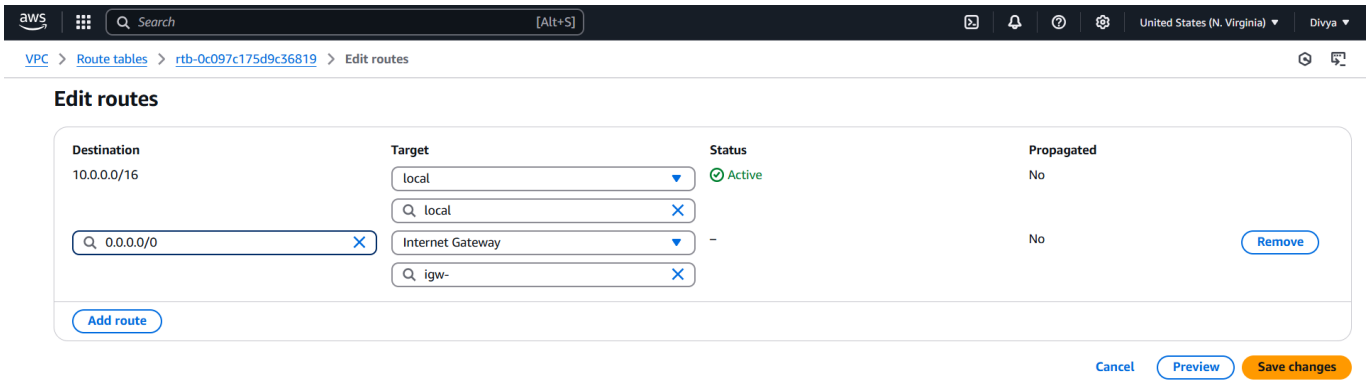


7.Establishing Route Table Connection Between IGW and Public Subnets

The **Route Table (MyRoute)** is created and configured to enable internet access for public subnets. It is linked to the **Internet Gateway (MyIGW)** and associated with **PublicSubnet-A** and **PublicSubnet-B**. This configuration ensures that instances within these subnets can establish outbound connections to the internet while securely managing inbound traffic as required.

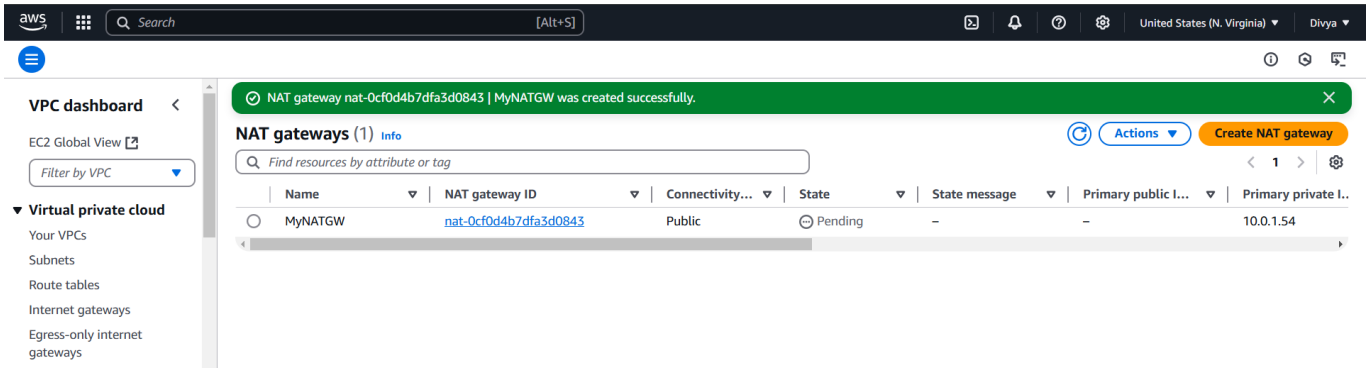






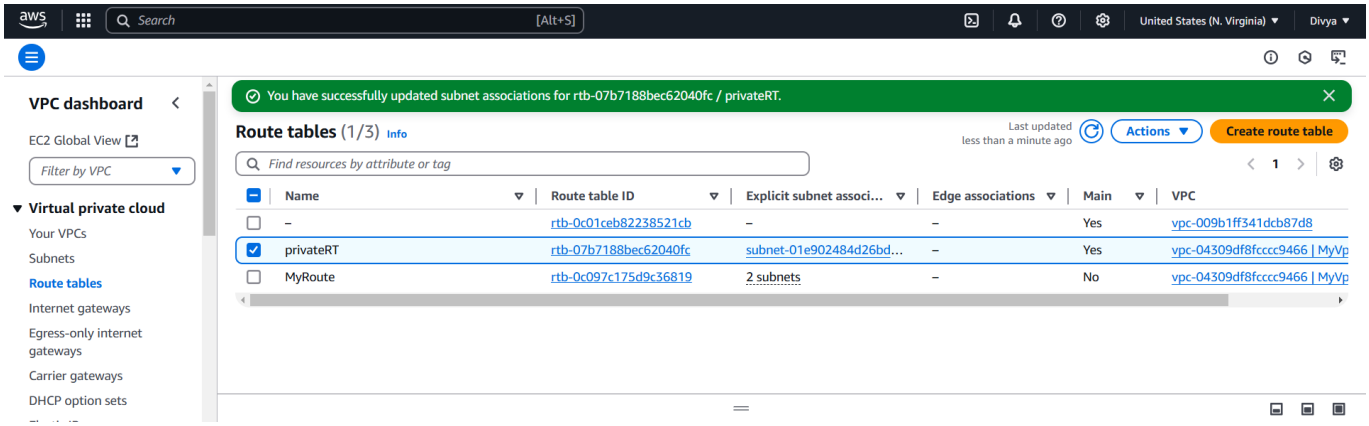
8. Configuring NAT Gateway for Private Subnet Internet Access

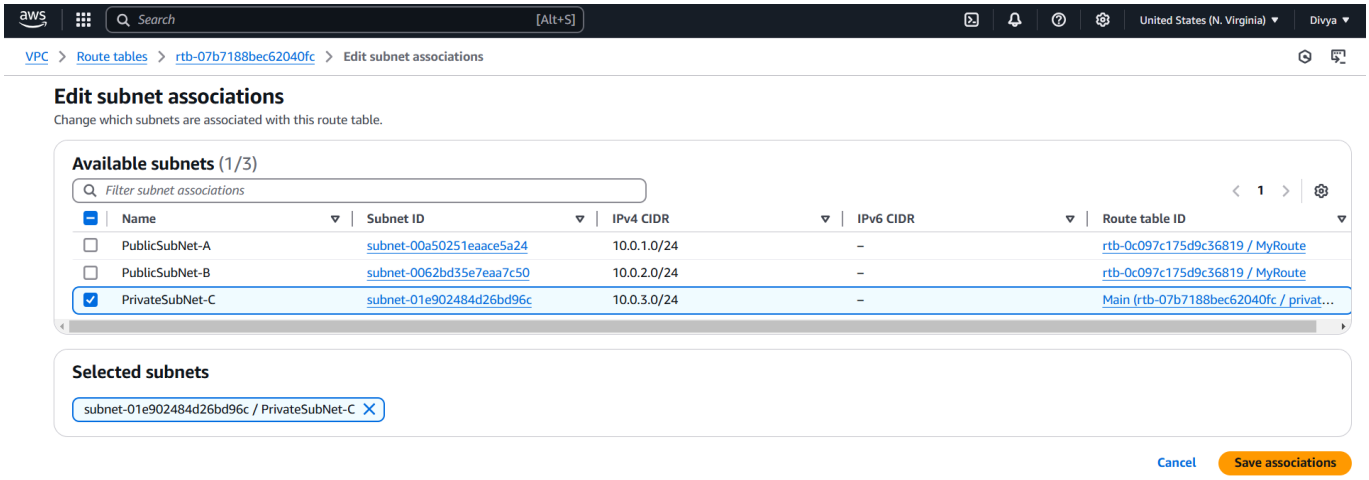
A **NAT Gateway(MyNATGW)** is created in **PublicSubnet-A** and assigned an **Elastic IP** to enable secure internet access for instances in the **private subnet**. Since private subnets do not have direct internet access, the NAT Gateway allows instances in **PrivateSubnet-C** to initiate outbound connections for tasks such as software updates, package installations, and API communications while preventing unsolicited inbound traffic.

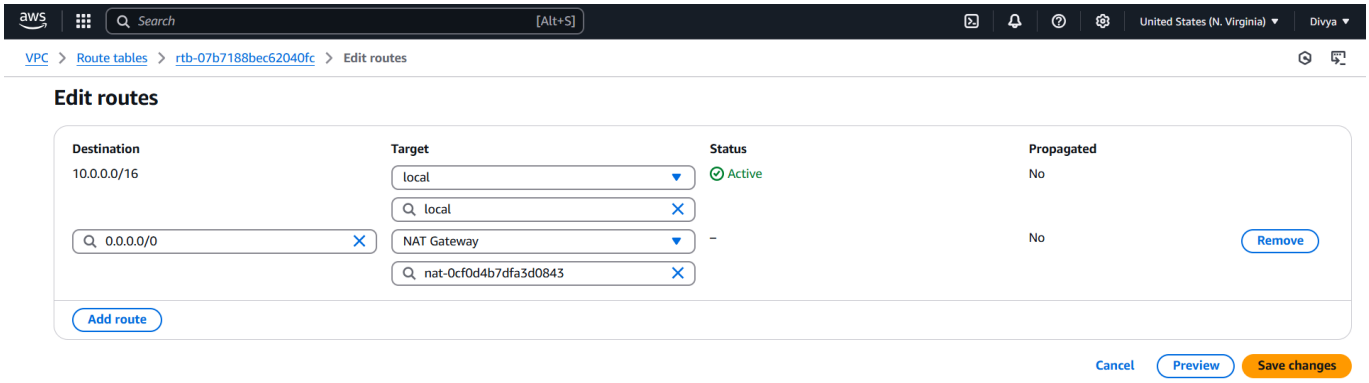


9. Integrating MyNATGW with PrivateSubnet-C via Route Table

MyNATGW is connected to **PrivateSubnet-C** through the default **route table** (**privateRT**), which is automatically created during the creation of MyVPC. This configuration ensures that outbound internet traffic from PrivateSubnet-C is routed through MyNATGW while maintaining security by restricting inbound access.

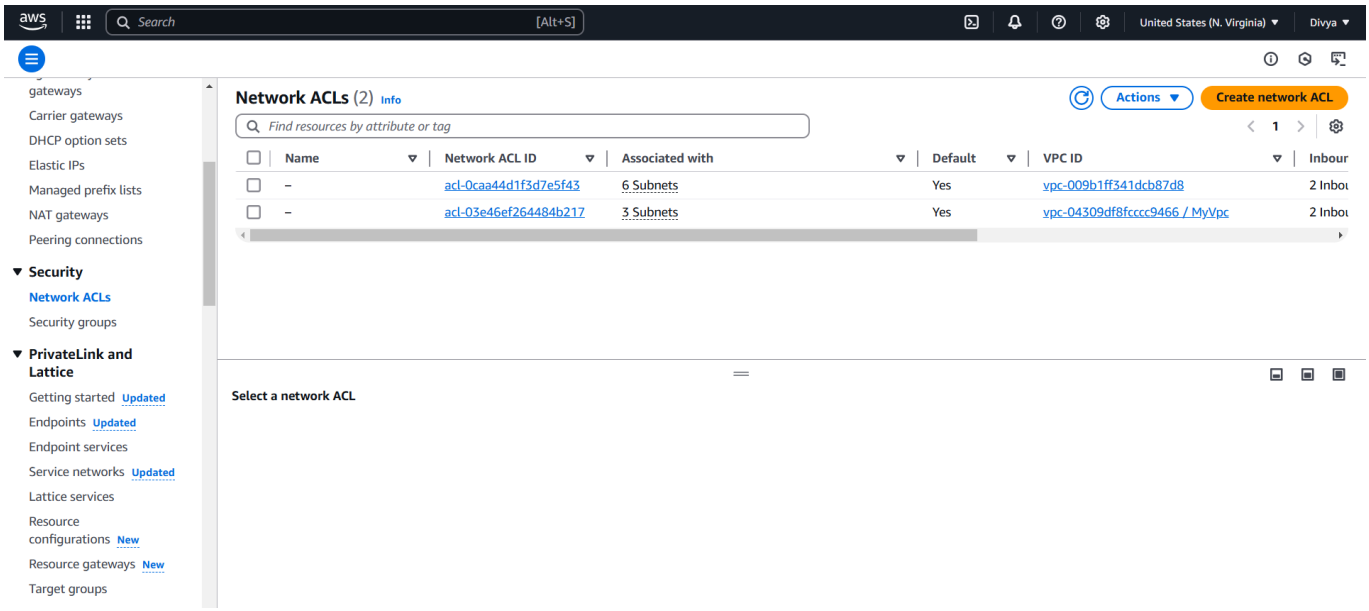






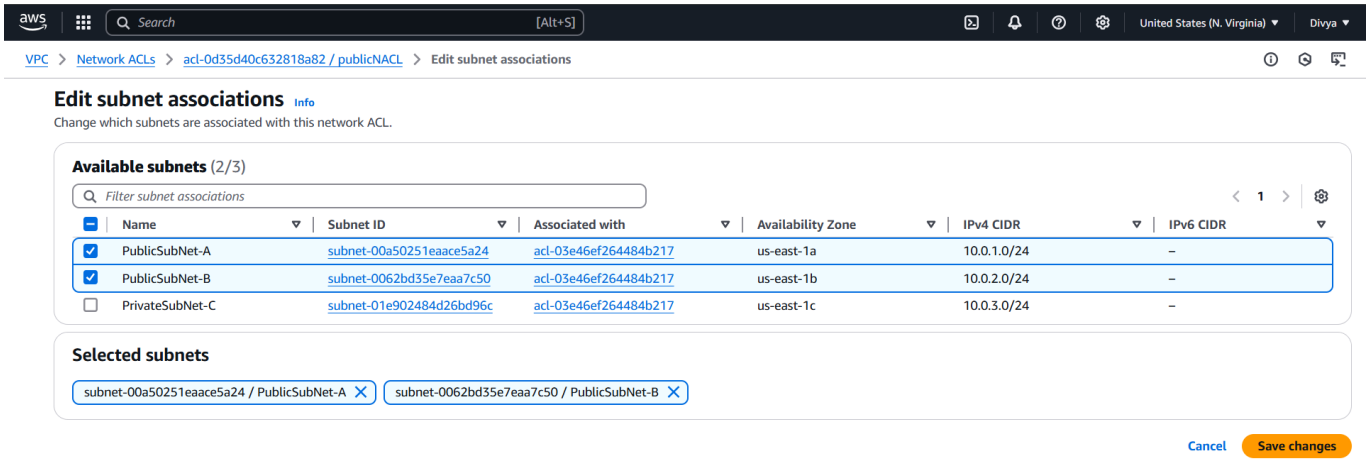
10.Creating NACL

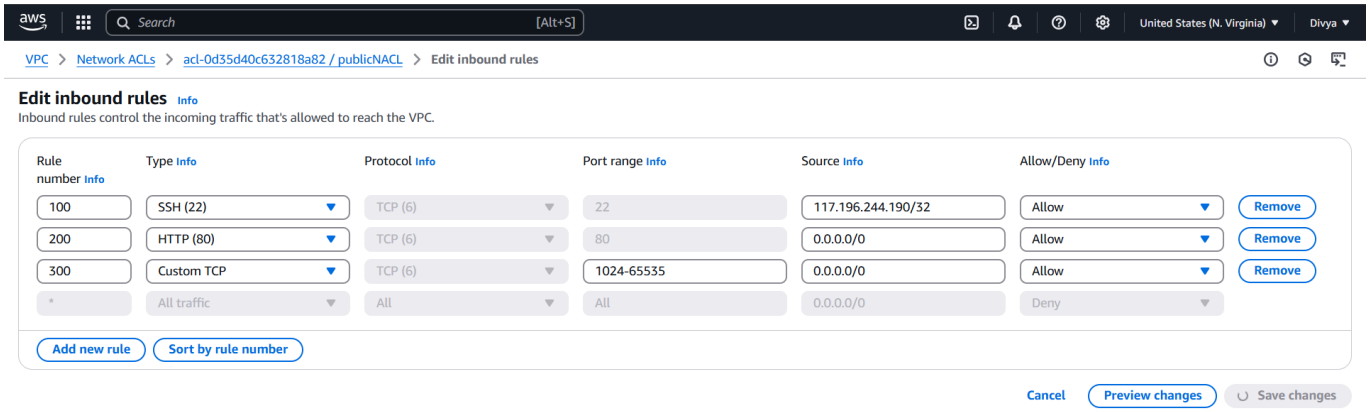
Initially, the default NACL of MyVPC is automatically associated with all three subnets (PublicSubnet-A, PublicSubnet-B, and PrivateSubnet-C). This default NACL allows all inbound and outbound traffic by default.

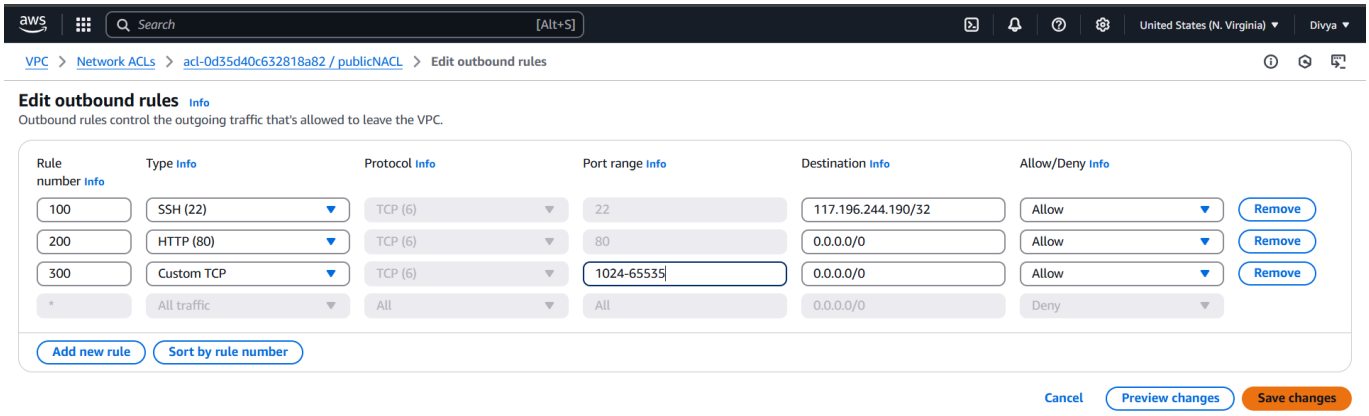


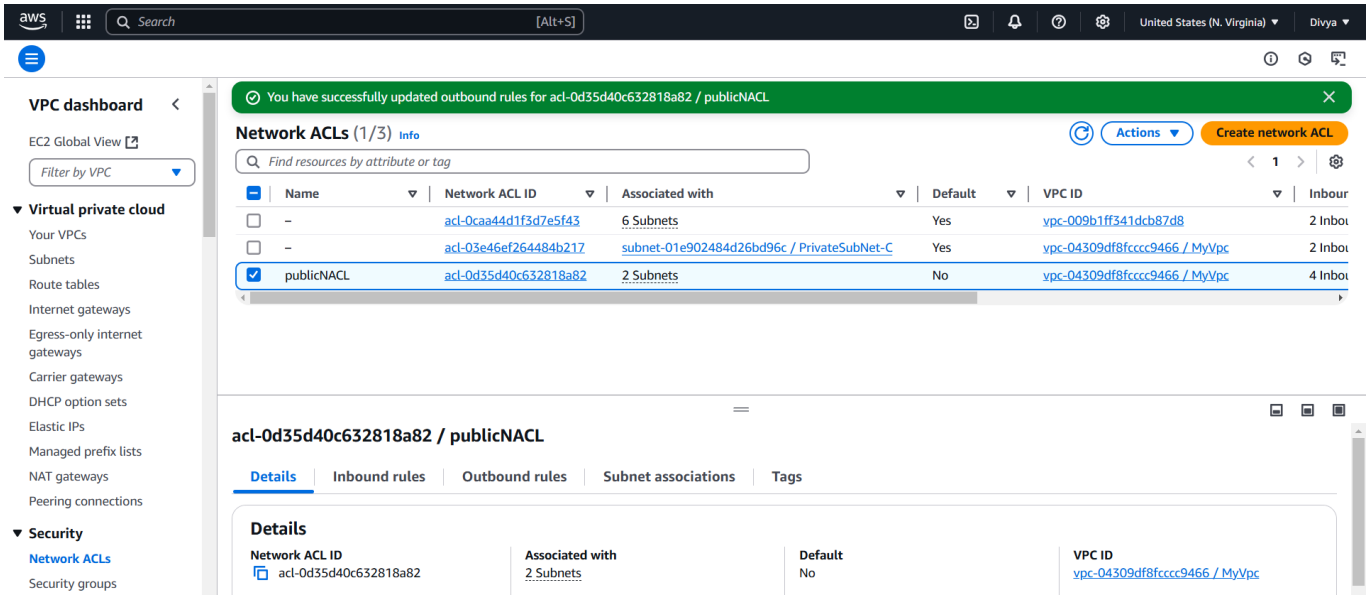
A Network Access Control List (NACL) named **publicNACL** is created within MyVPC to manage traffic flow at the subnet level. NACLs provide an additional layer of security by enforcing stateless inbound and outbound traffic rules.

This PublicNACL is associated with **PublicSubnet-A** and **PublicSubnet-B**, ensuring controlled access for internet-facing resources while maintaining security best practices.





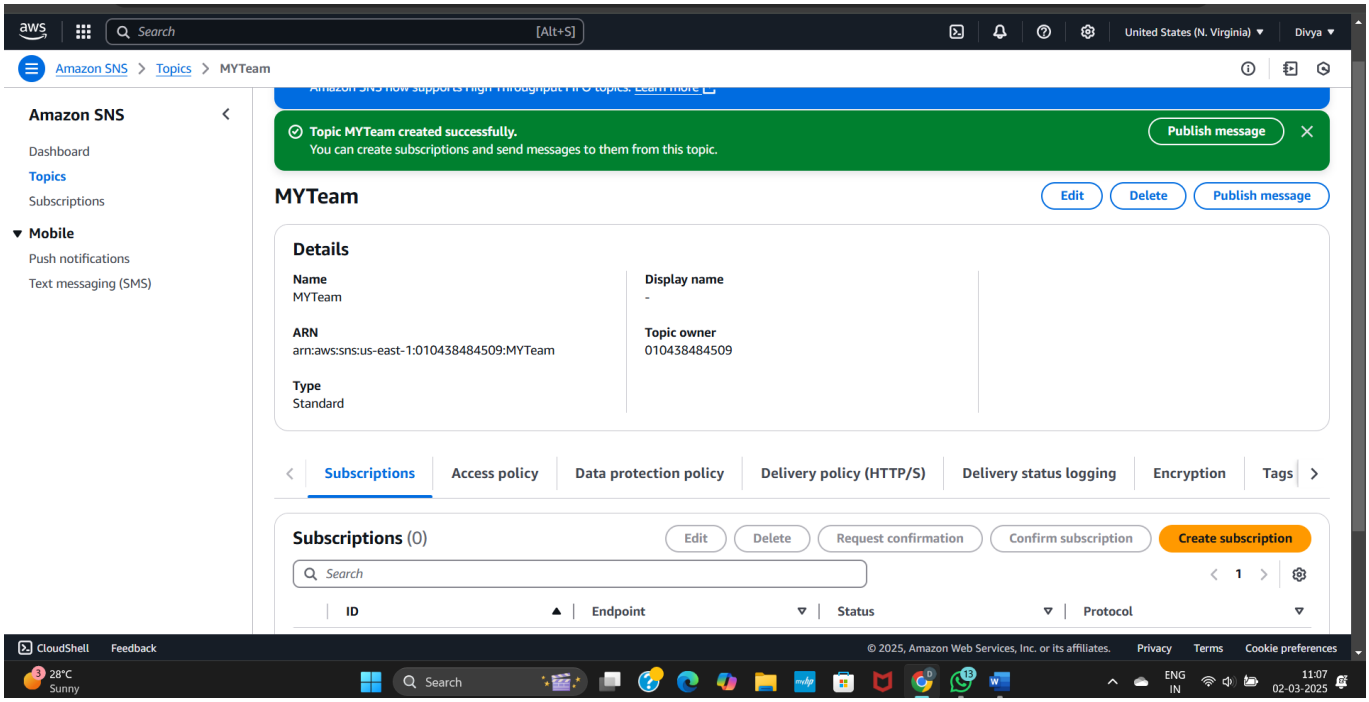


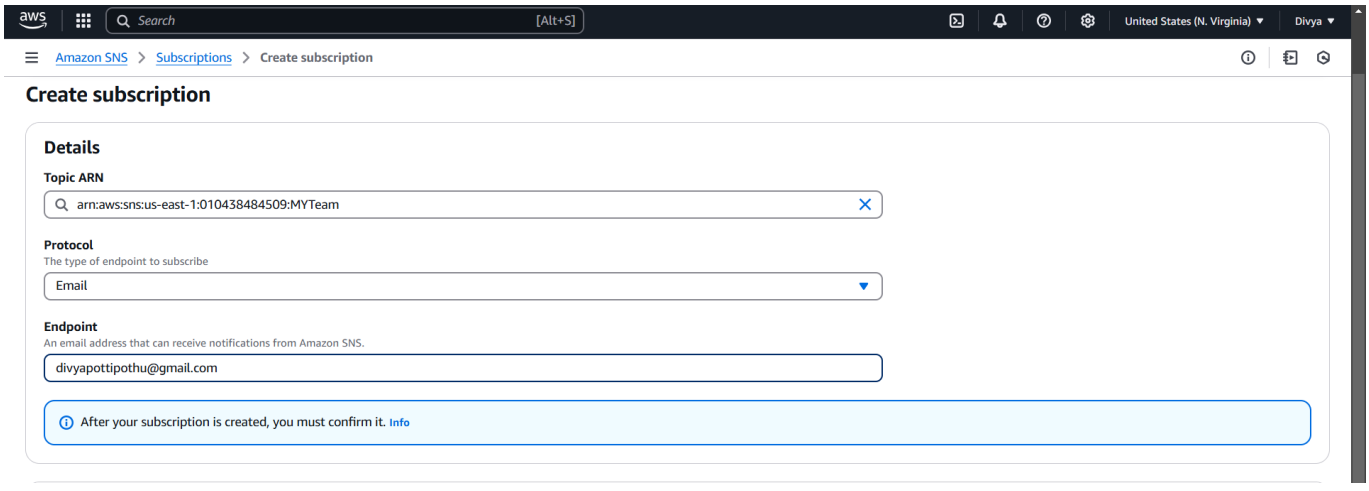


The publicNACL secures PublicSubnet-A and B, enabling SSH (MyIP), HTTP (0.0.0.0/0), and ephemeral ports for responses. The default NACL protects PrivateSubnet-C, restricting traffic while allowing internal communication via the NAT Gateway.

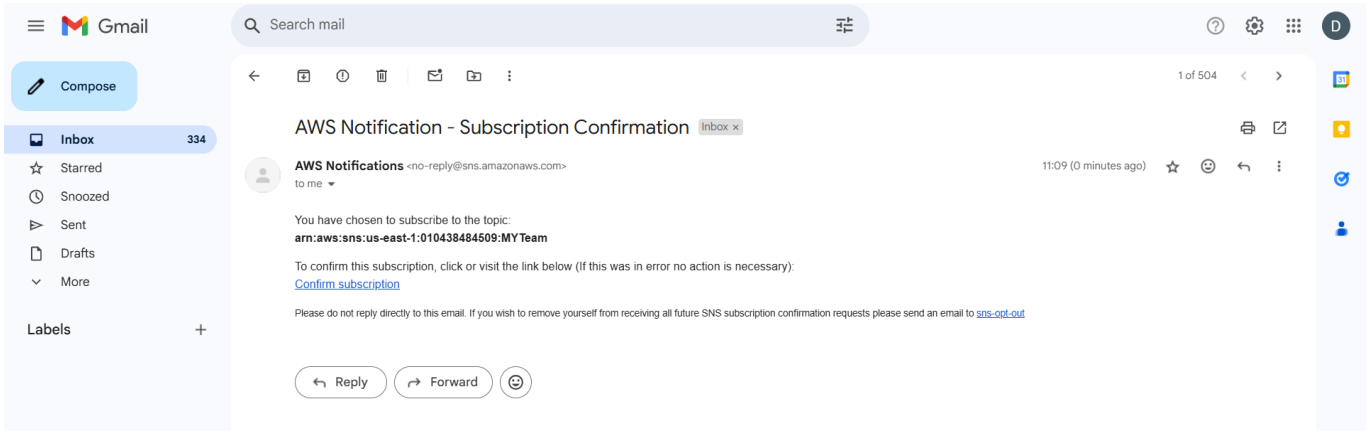
11. Creating Amazon SNS Topic and Subscriptions

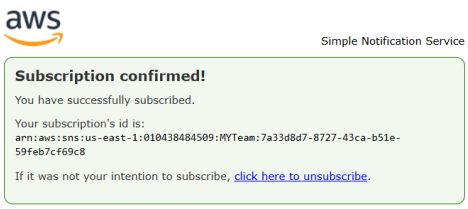
An **SNS Topic** (MyTeam) is created to enable event-driven notifications. **Subscriptions** are added to MyTeam, allowing multiple recipients (such as email addresses, SMS, or AWS services) to receive notifications whenever an event is published to the topic. This setup ensures efficient communication and automated alerts for system events or updates.

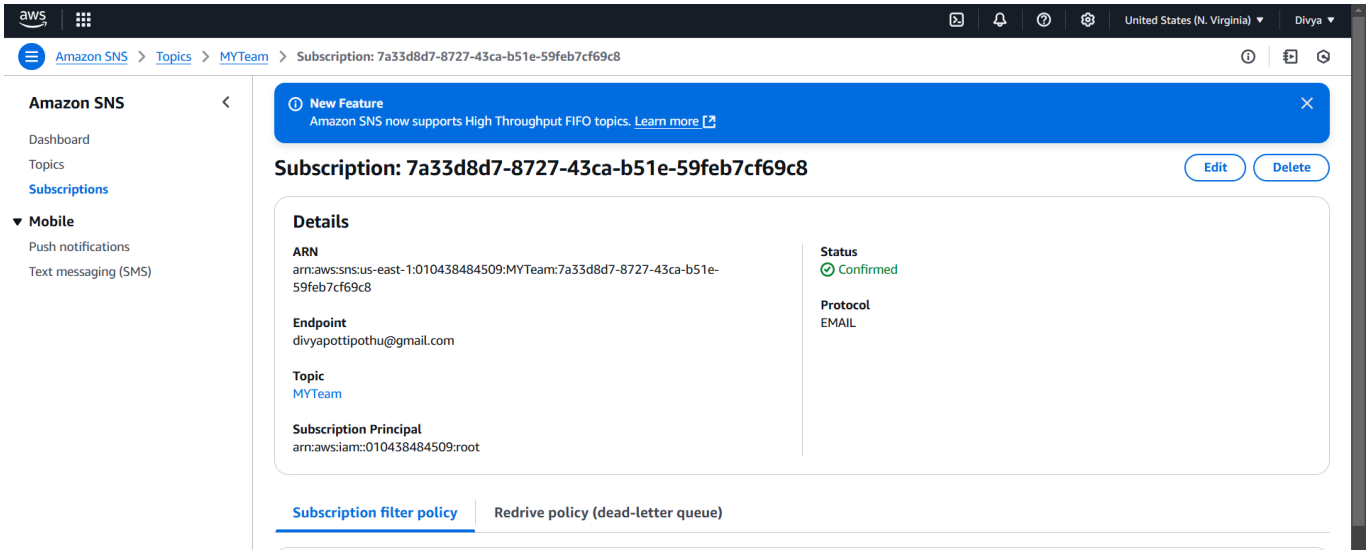




Subscriptions are added to **MyTeam**, and each subscriber receives a confirmation request. Once the subscription is confirmed, the recipient starts receiving notifications whenever an event is published to the topic.

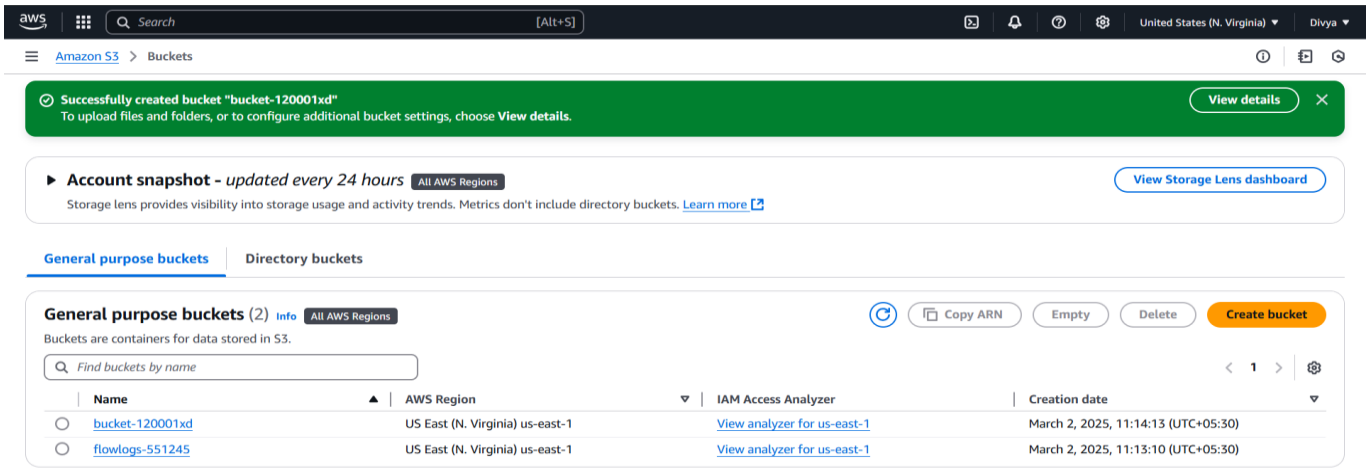






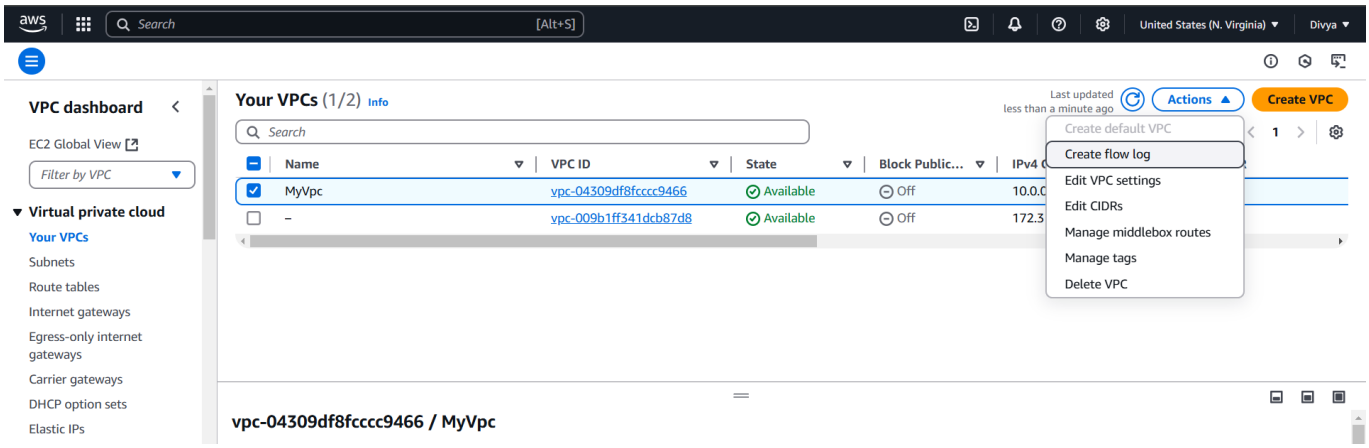
12.Configuring Amazon S3 for Scalable Storage

Two **Amazon S3 buckets** are created to serve distinct purposes. The FlowLogs Storage Bucket is designated for storing VPC(MyVpc) Flow Logs, enabling network traffic monitoring and security analysis. The General Purpose Bucket is used for storing application data, backups, and other essential files. These buckets ensure efficient data organization, security, and seamless integration with AWS services for scalable and reliable storage solutions.



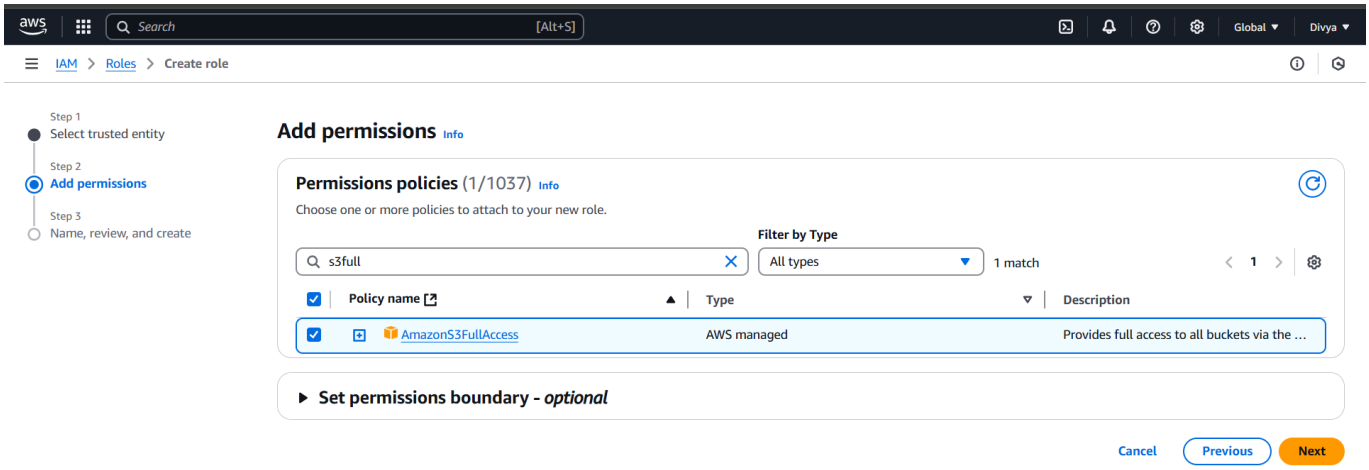
13.Enabling VPC Flow Logs and Storing in S3

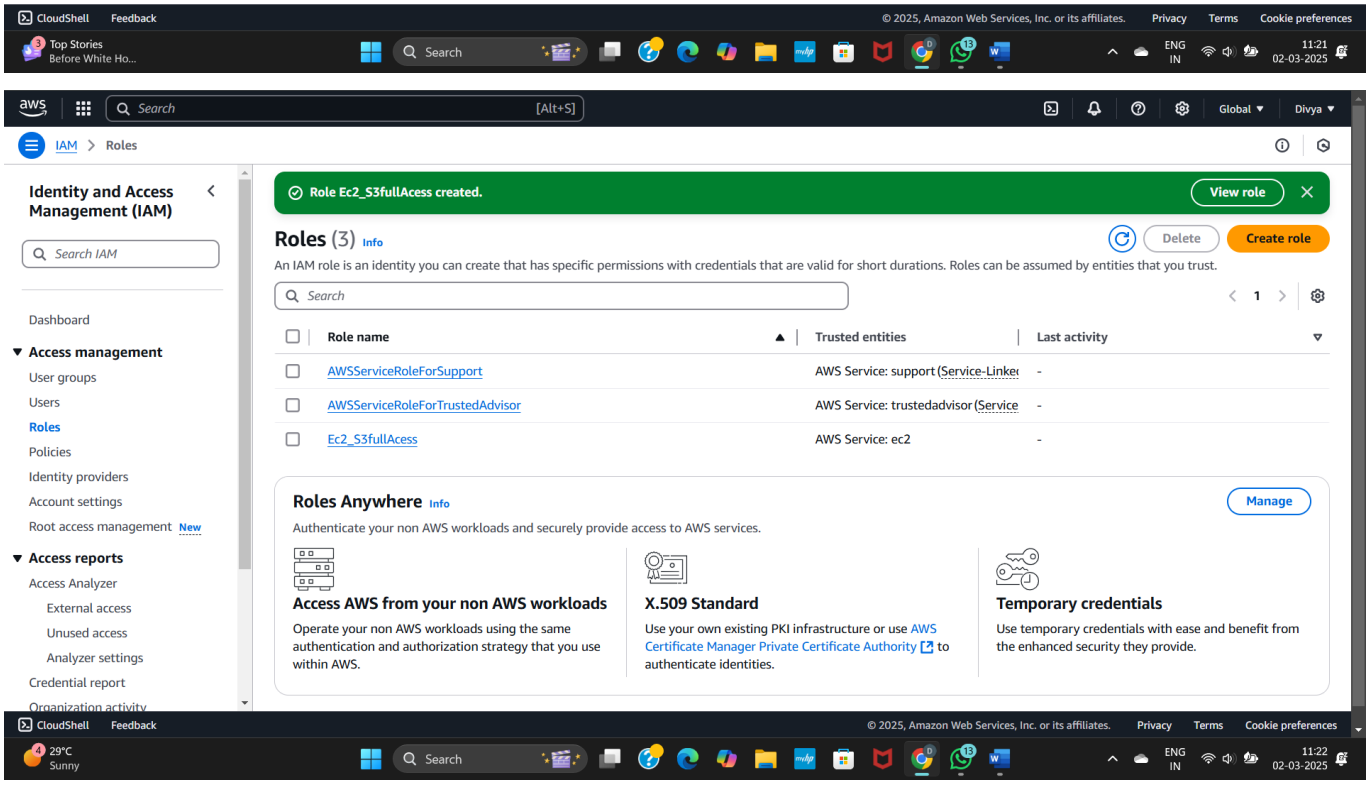
VPC **Flow Logs** are created for MyVPC to capture and monitor network traffic. The logs are configured to be stored in an **S3 bucket** using its ARN (Amazon Resource Name). This setup ensures centralized log storage, enabling security analysis, troubleshooting, and compliance auditing while maintaining efficient data management**.**



14. Creating IAM Role for EC2 with S3 Full Access

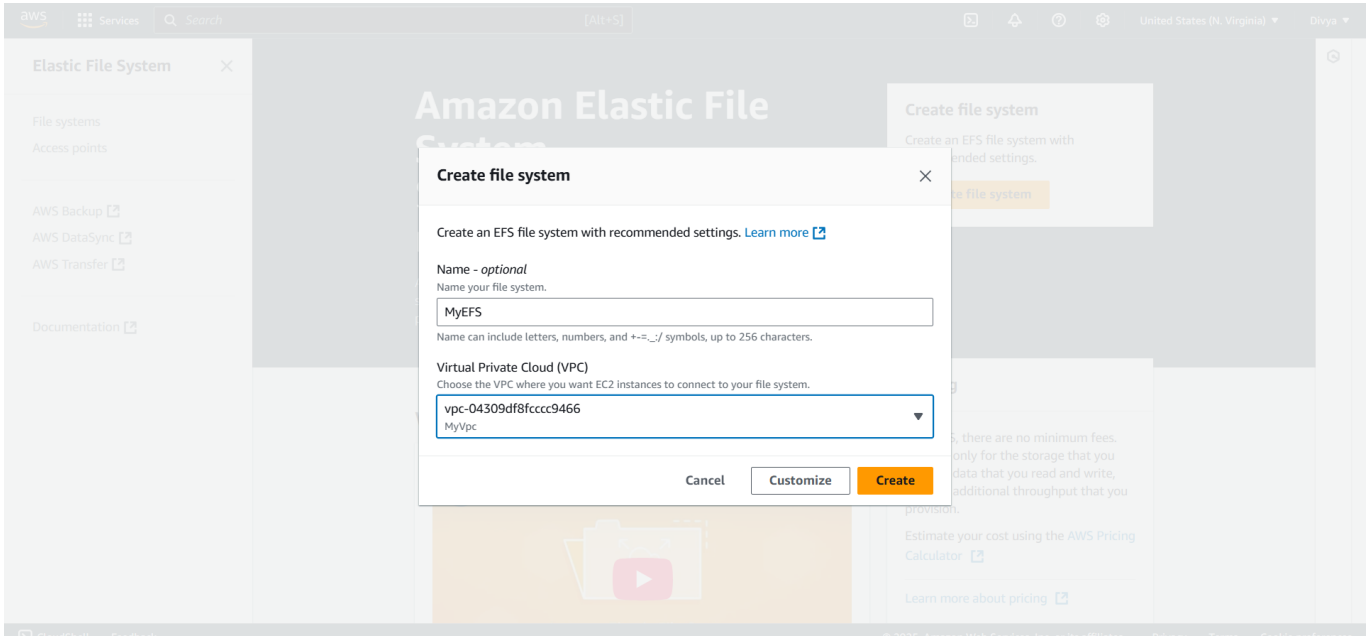
An **IAM role** (Ec2_S3fullAcess) is created to grant **EC2 instances** full access to **Amazon S3**. This role is assigned the **AmazonS3FullAccess** policy, allowing EC2 instances to read, write, and manage objects in S3 buckets without requiring manual credential configuration. By attaching this IAM role to EC2 instances, secure and seamless interaction with S3 is enabled.

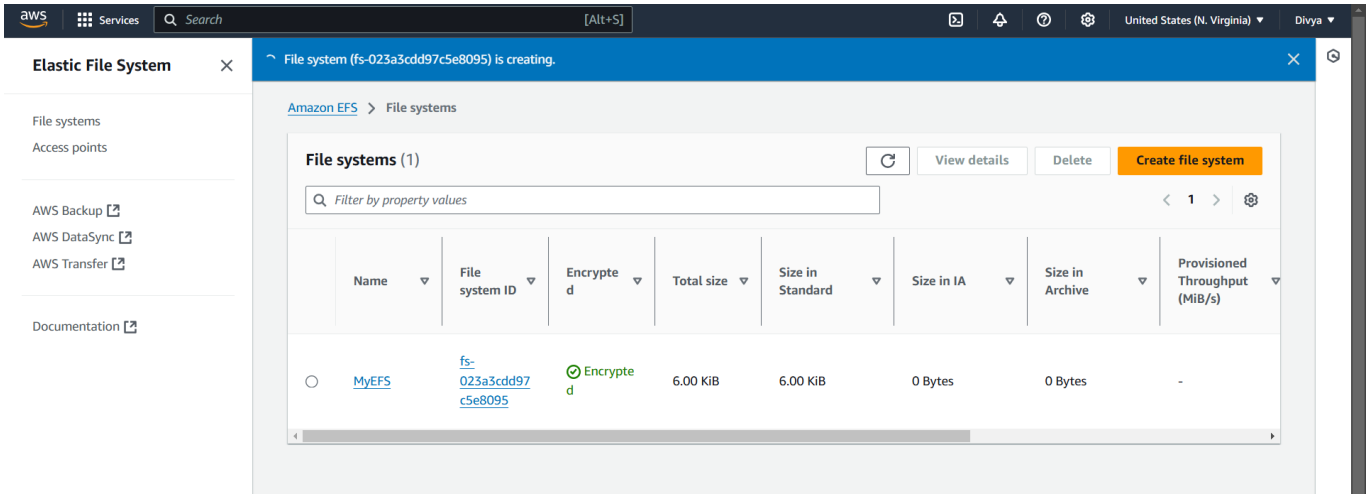




15. Setting Up Amazon Elastic File System (MyEFS)

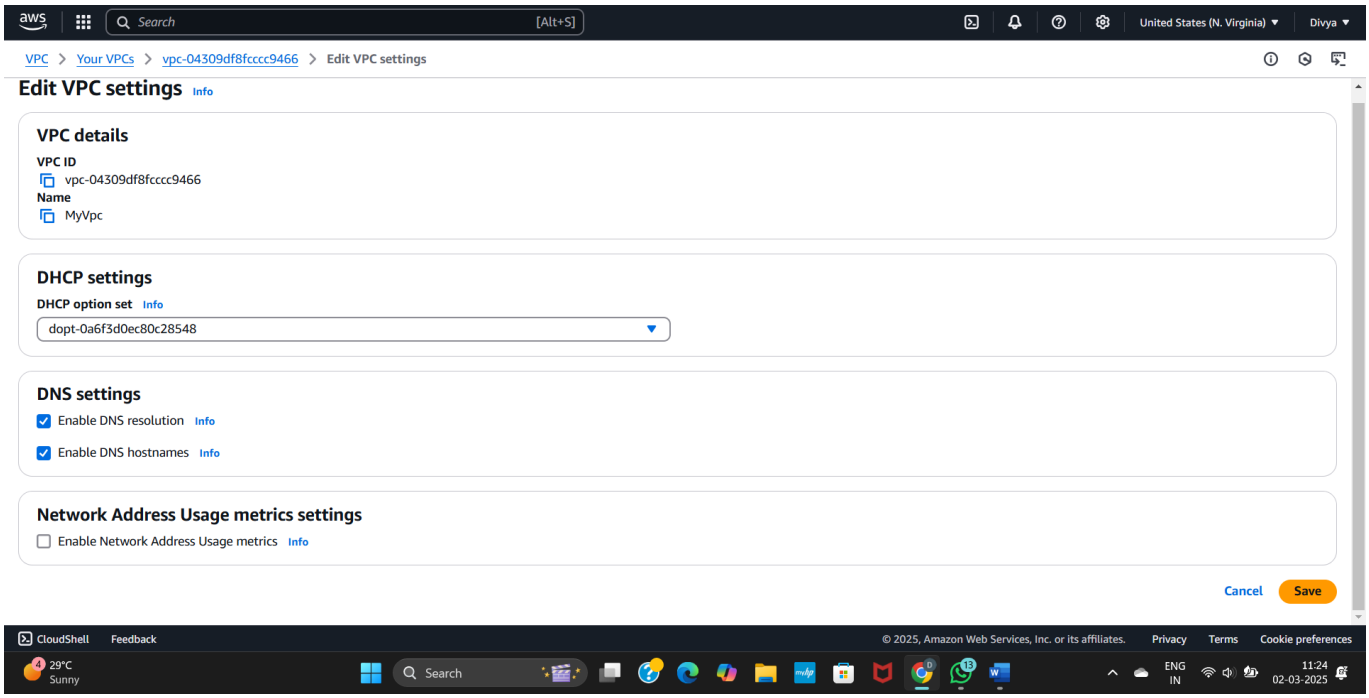
Amazon **Elastic File System** (MyEFS) is created to provide scalable and fully managed file storage for **Linux-based** EC2 instances. EFS allows multiple instances to access the same file system concurrently, making it ideal for shared storage, application hosting, and data persistence. It is designed to scale automatically based on demand, ensuring high availability and durability. By mounting EFS on Linux instances, applications can seamlessly read and write data without the need for manual storage management.





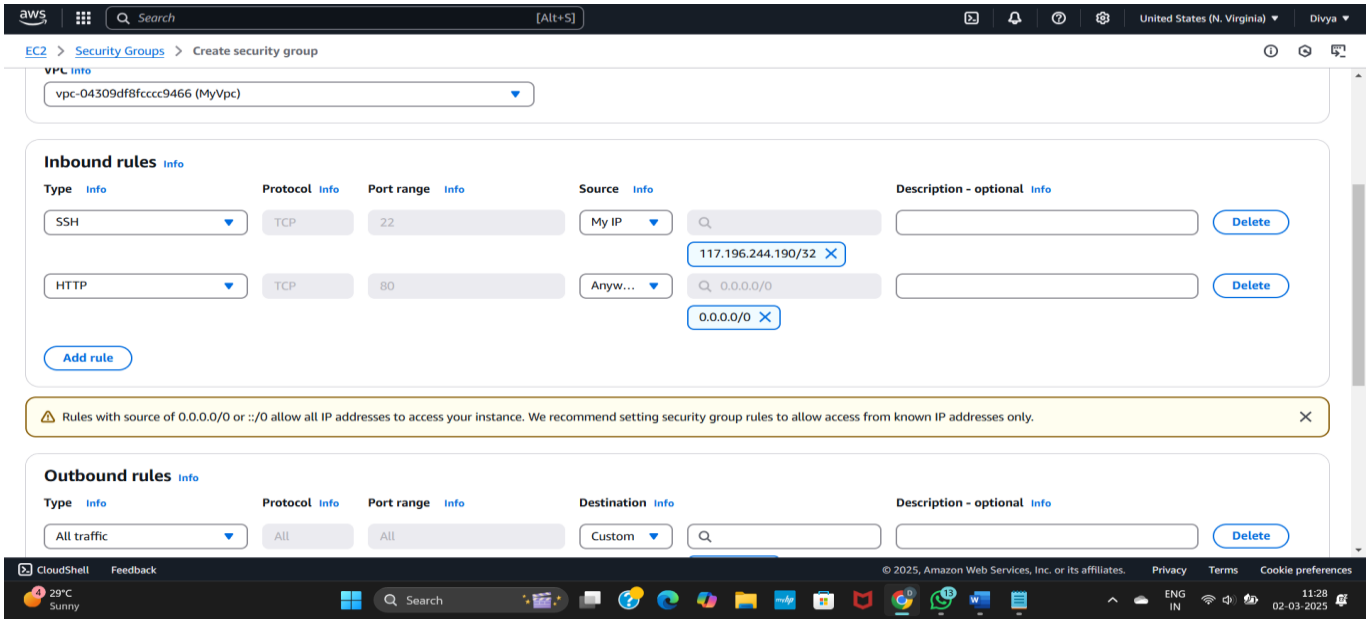
When creating **Amazon Elastic File System (EFS)**, enabling **DNS hostnames** in **MyVPC** is essential because EFS uses **DNS names** to allow EC2 instances to connect to the file system. Since EFS does not use IP addresses directly, enabling **DNS resolution** ensures that EC2 instances within the VPC can **mount**

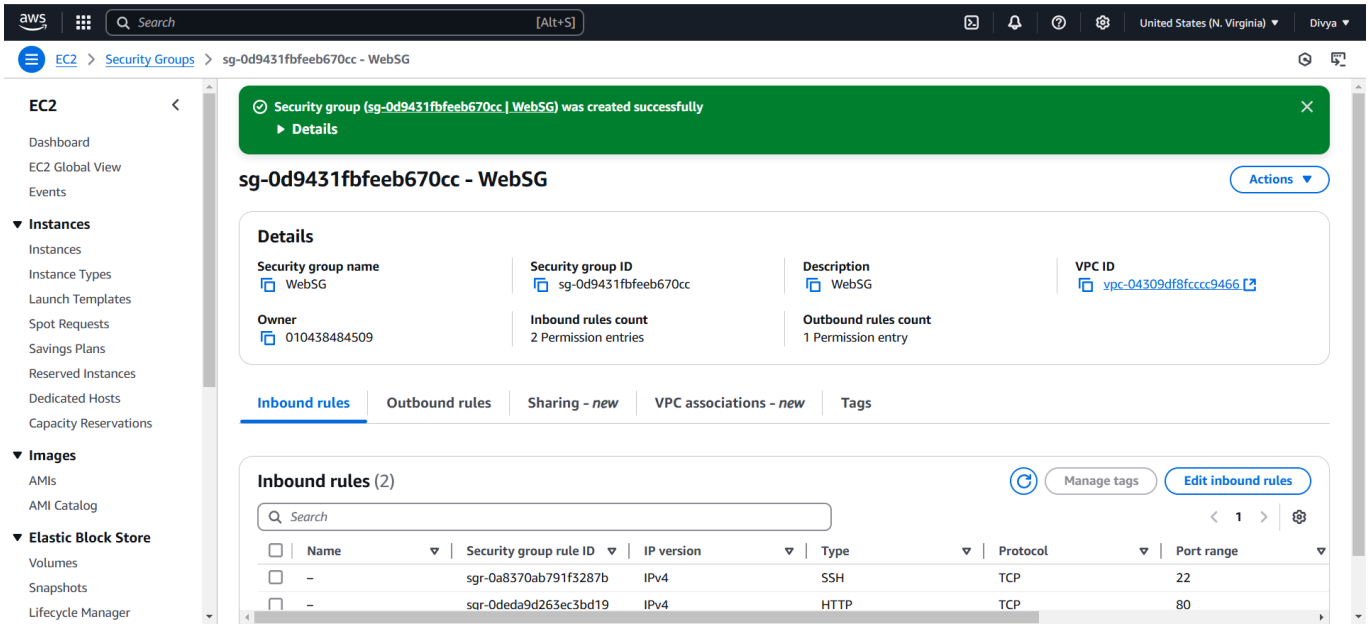
and access the EFS file system using the automatically assigned DNS names. Without DNS enabled, instances would not be able to resolve the EFS mount targets, leading to connection failures.



16. Defining Security Group for Web Server

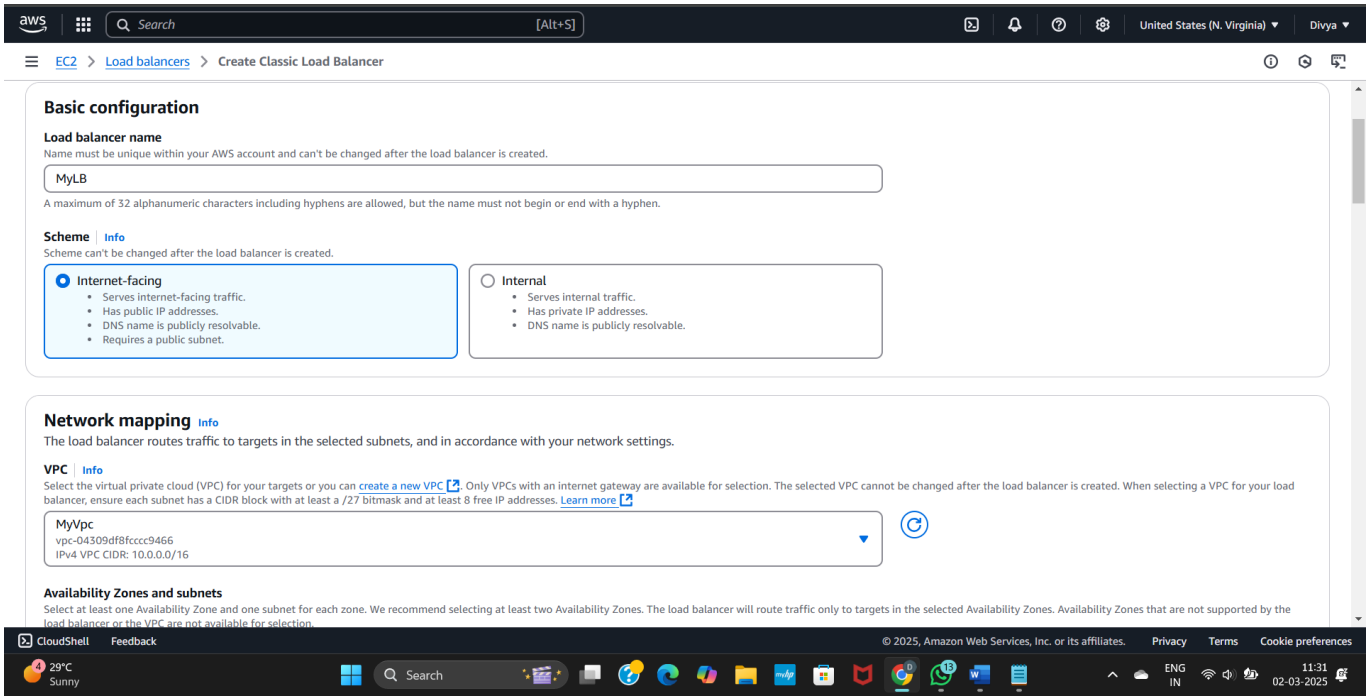
A Security Group named **WebSG** is created for **EC2** instances deployed in **MyVPC's** PublicSubnets. To ensure secure access, the inbound rules allow **SSH** (port 22) access only from **MyIP** for administrative control and **HTTP** (port 80) access from **all IP addresses** to enable public web traffic. The outbound rules remain open by default, allowing unrestricted outbound communication for the instances. This configuration ensures secure remote access while making the web server accessible to users.

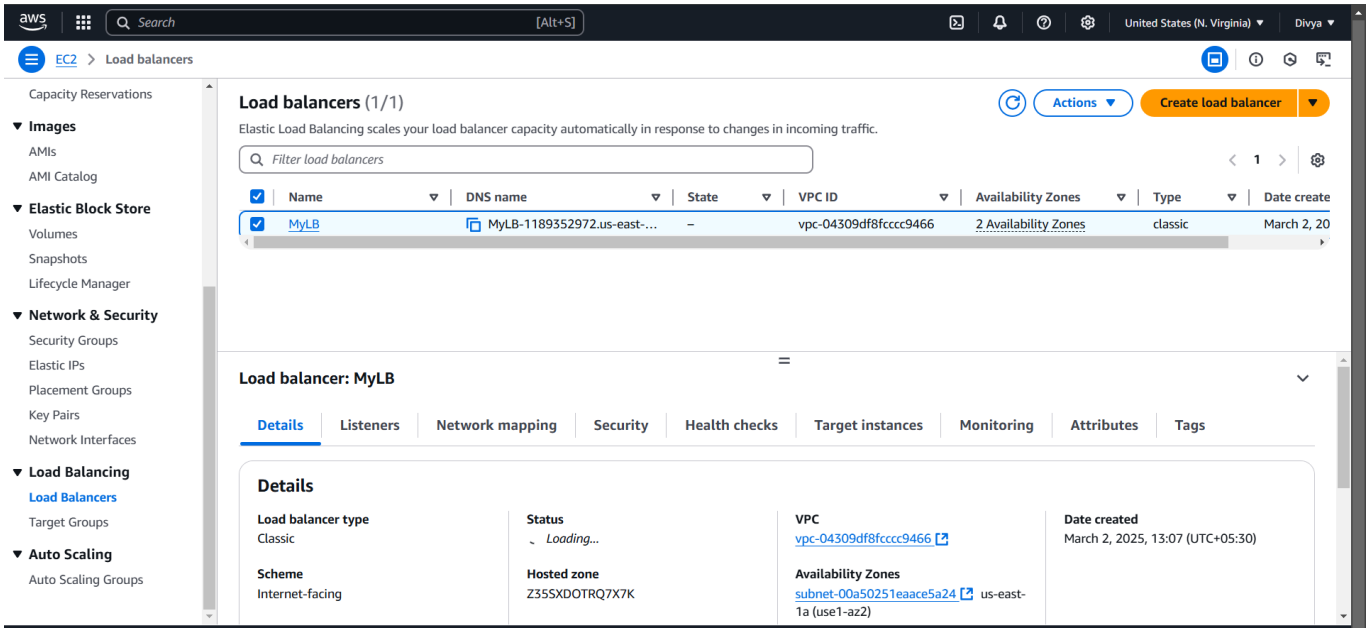




17.Creating LoadBalancer

A Classic Load Balancer (MyLB) is created in MyVPC, spanning Availability Zones A and B to ensure high availability and fault tolerance. It is associated with WebSG for web traffic security and the default Security Group, which is required for EFS access. The idle timeout is set to 300 seconds, ensuring stable session management for applications relying on persistent connections. This setup efficiently distributes incoming traffic across multiple instances, improving scalability and reliability.

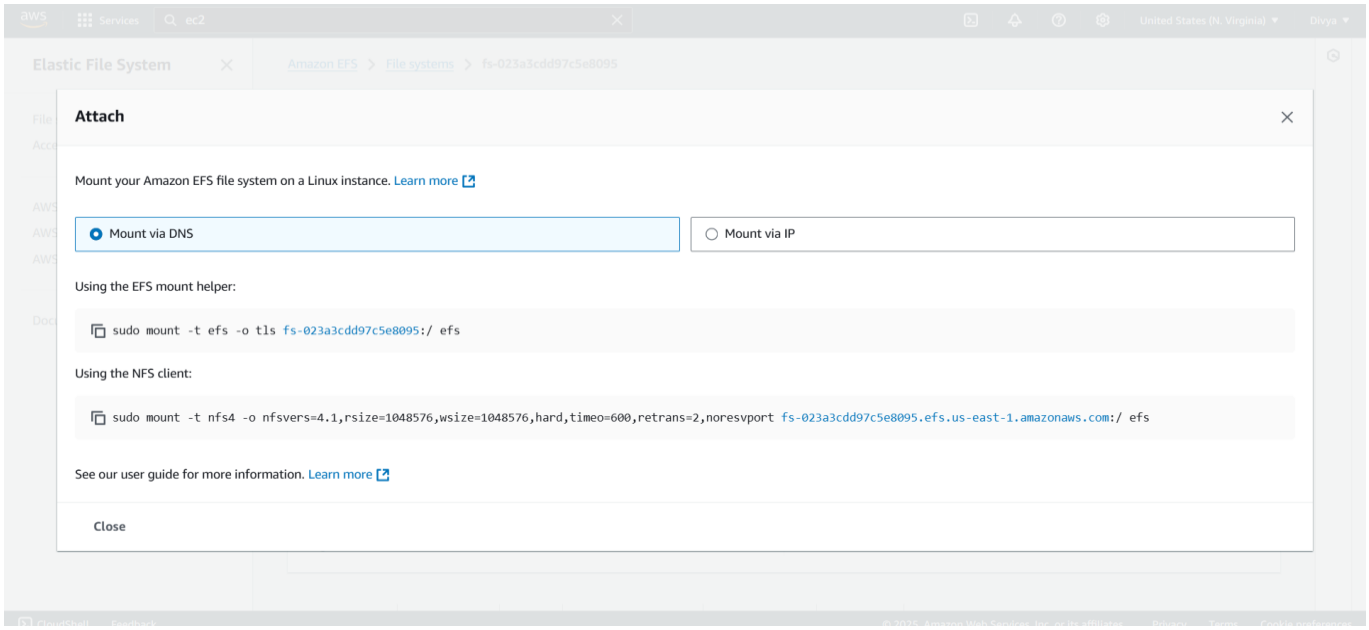


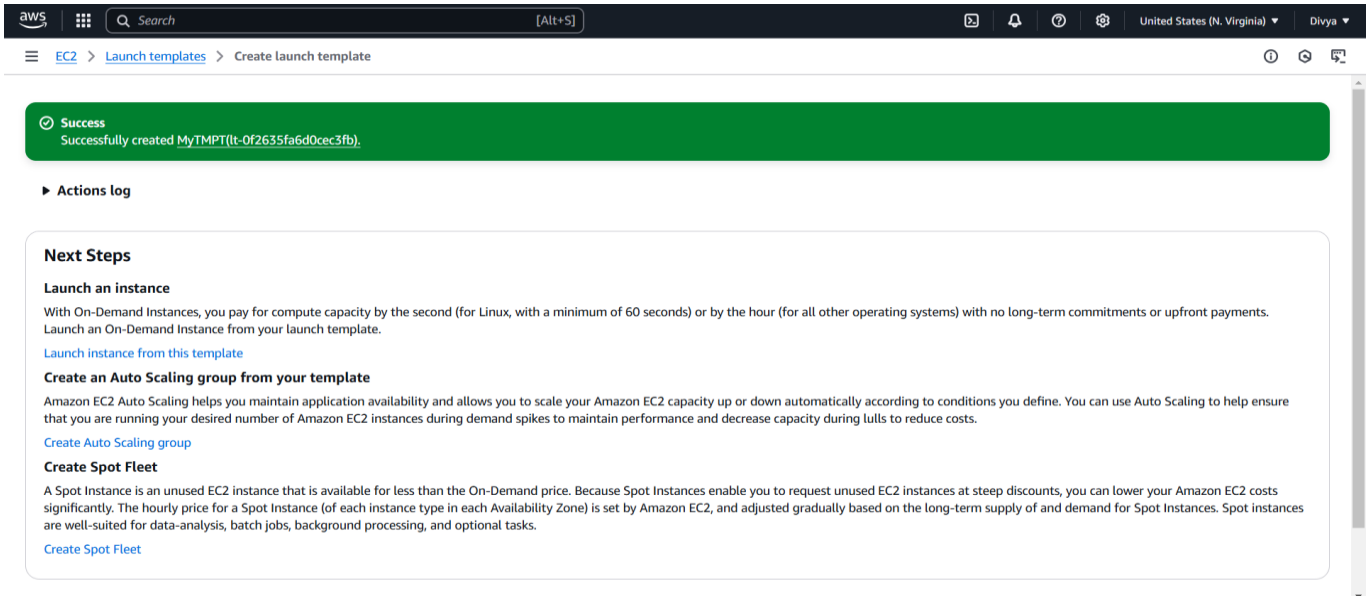


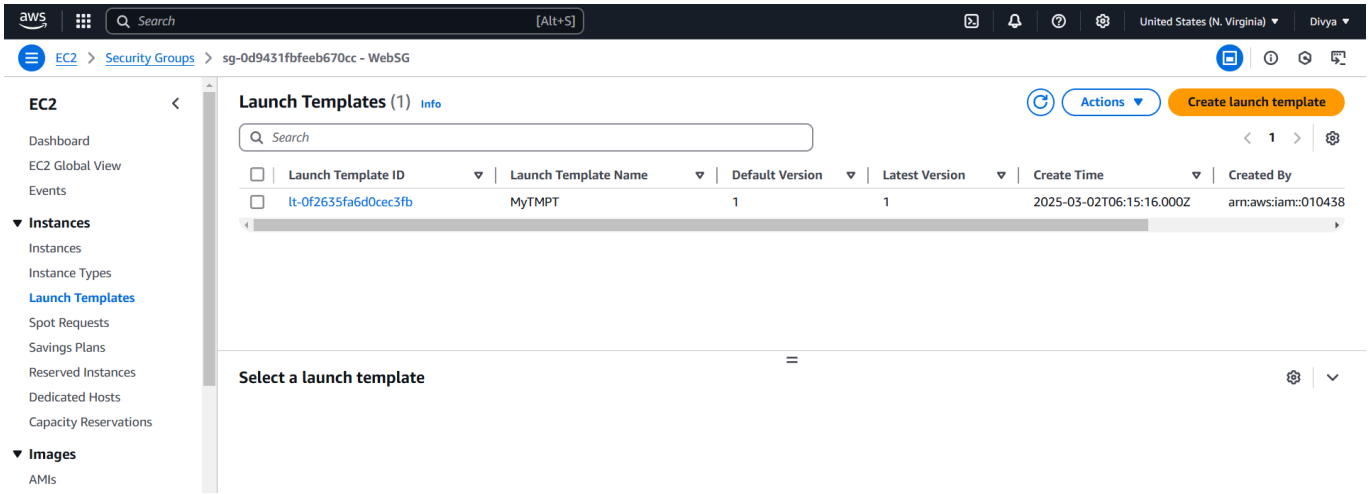
18. Configuring a Launch Template for EC2 with EFS and S3 Access

A **Launch Template (MyTMPT)** is created to streamline EC2 instance deployment with pre-configured settings. It is based on Amazon Linux and uses the **projectKey** key pair for secure SSH access. The instance is associated with WebSG for web security and the default Security Group for additional network controls. To enable seamless storage integration, the instance mounts an Amazon EFS file system, ensuring persistent and shared storage across multiple instances. Additionally, it is configured with an IAM role (EC2_S3_FullAccess) to securely interact with Amazon S3, allowing retrieval of web content. The Apache web server (HTTPD) is installed and configured to serve an index.html file from EFS, providing a scalable and highly available web hosting environment.

```
#!/bin/bash
sudo su -
yum update -y
mkdir /divya
#Attach EFS using NFS Client command
yum install httpd -y
cd /var/www/html
echo "Hello All .....!!!!" > index.html
service httpd start
chkconfig httpd on
```

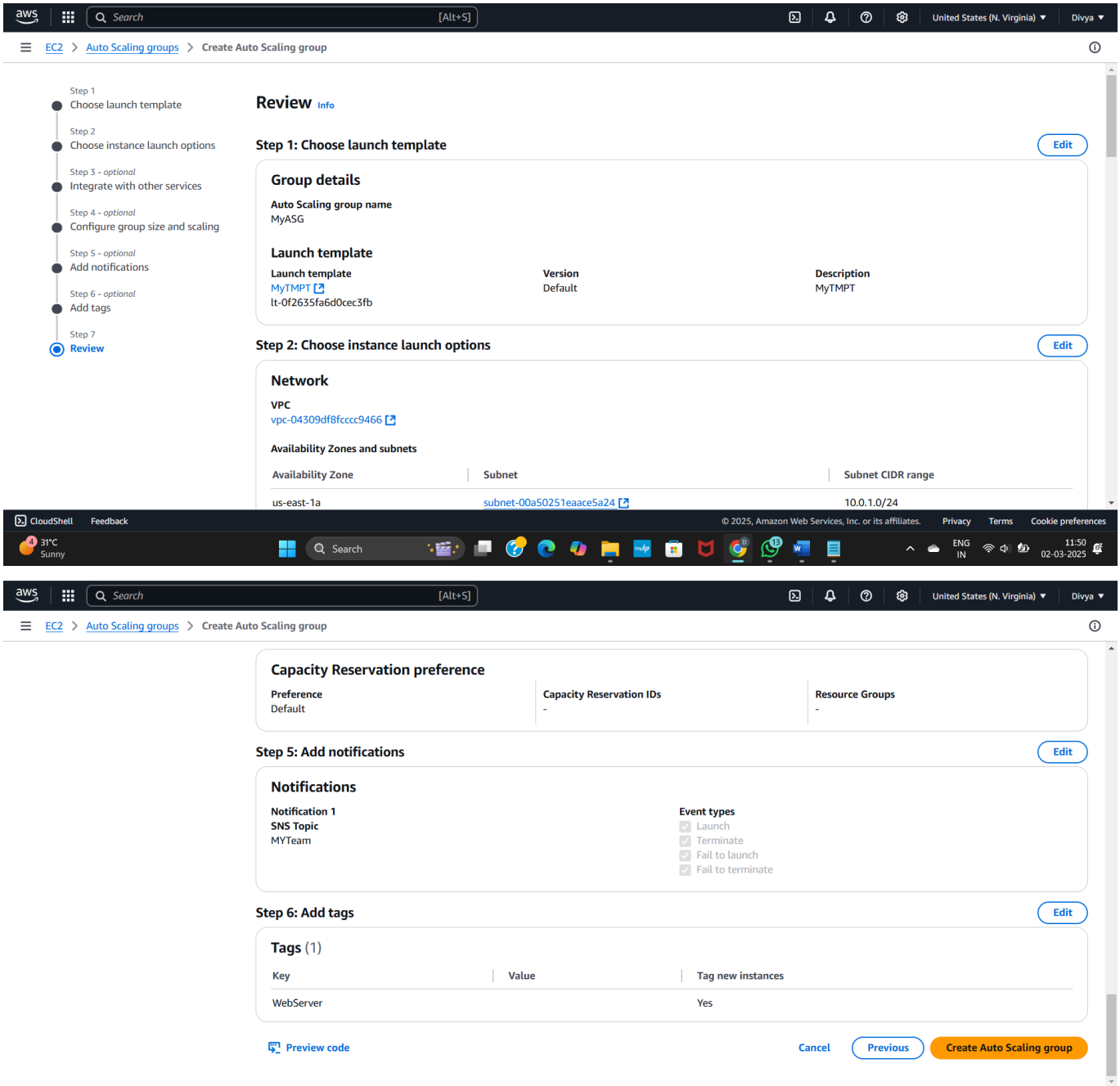

Elastic File System
Amazon EFS > File systems > fs-023a3cdd97c5e8095
Attach
Mount your Amazon EFS file system on a Linux instance. Learn more
Mount via DNS
Mount via IP
Using the EFS mount helper:
sudo mount -t efs -o tls fs-023a3cdd97c5e8095:/ efs
Using the NFS client:
sudo mount -t nfs4 -o nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport fs-023a3cdd97c5e8095.efs.us-east-1.amazonaws.com:/ efs
See our user guide for more information. Learn more
Close

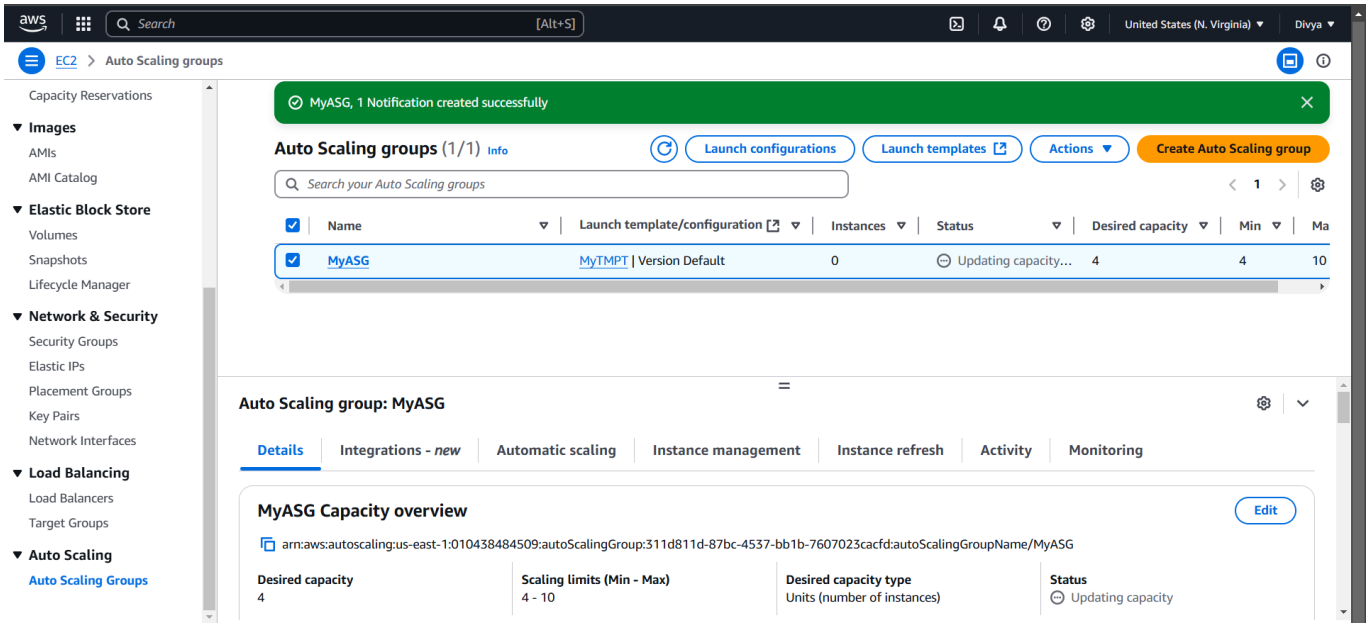

EC2 > Launch templates > Create launch template
Success
Successfully created MyTMPT(lt-0f2635fa6d0cec3fb).
Actions log
Next Steps
Launch an instance
With On-Demand Instances, you pay for compute capacity by the second (for Linux, with a minimum of 60 seconds) or by the hour (for all other operating systems) with no long-term commitments or upfront payments. Launch an On-Demand Instance from your launch template.
Launch instance from this template
Create an Auto Scaling group from your template
Amazon EC2 Auto Scaling helps you maintain application availability and allows you to scale your Amazon EC2 capacity up or down automatically according to conditions you define. You can use Auto Scaling to help ensure that you are running your desired number of Amazon EC2 instances during demand spikes to maintain performance and decrease capacity during lulls to reduce costs.
Create Auto Scaling group
Create Spot Fleet
A Spot Instance is an unused EC2 instance that is available for less than the On-Demand price. Because Spot Instances enable you to request unused EC2 instances at steep discounts, you can lower your Amazon EC2 costs significantly. The hourly price for a Spot Instance (of each instance type in each Availability Zone) is set by Amazon EC2, and adjusted gradually based on the long-term supply of and demand for Spot Instances. Spot instances are well-suited for data-analysis, batch jobs, background processing, and optional tasks.
Create Spot Fleet


EC2 > Security Groups > sg-0d9431fbfeeb670cc - WebSG
EC2
Dashboard
EC2 Global View
Events
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations
Images
AMIs
Launch Templates (1) Info
Actions
Create launch template
Search
Launch Template ID
Launch Template Name
Default Version
Latest Version
Create Time
Created By
lt-0f2635fa6d0cec3fb
MyTMPT
1
1
2025-03-02T06:15:16.000Z
arn:aws:iam::010438
Select a launch template

19. Setting Up Auto Scaling Group (MyASG) for High Availability

The Auto Scaling Group (MyASG) is created to ensure high availability and scalability of EC2 instances in MyVPC across Availability Zones A and B. It uses the MyTMPT launch template, ensuring consistent instance configurations. The MyLB load balancer is associated with the group to evenly distribute traffic and improve fault tolerance. Health checks are configured with a threshold of 150 seconds to ensure only healthy instances handle traffic. Additionally, an SNS notification is enabled to send alerts about scaling activities, and a tag (Name: WebServer) is assigned for easy identification and management of instances within the group.

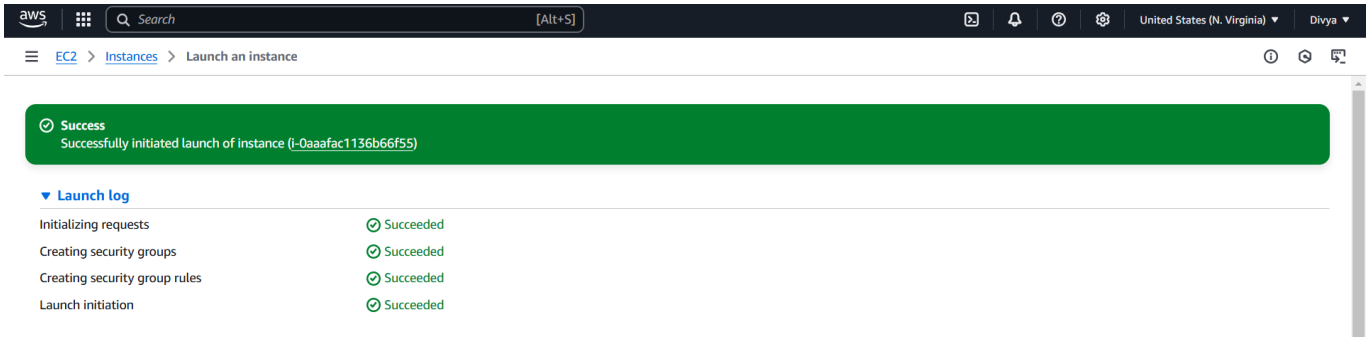


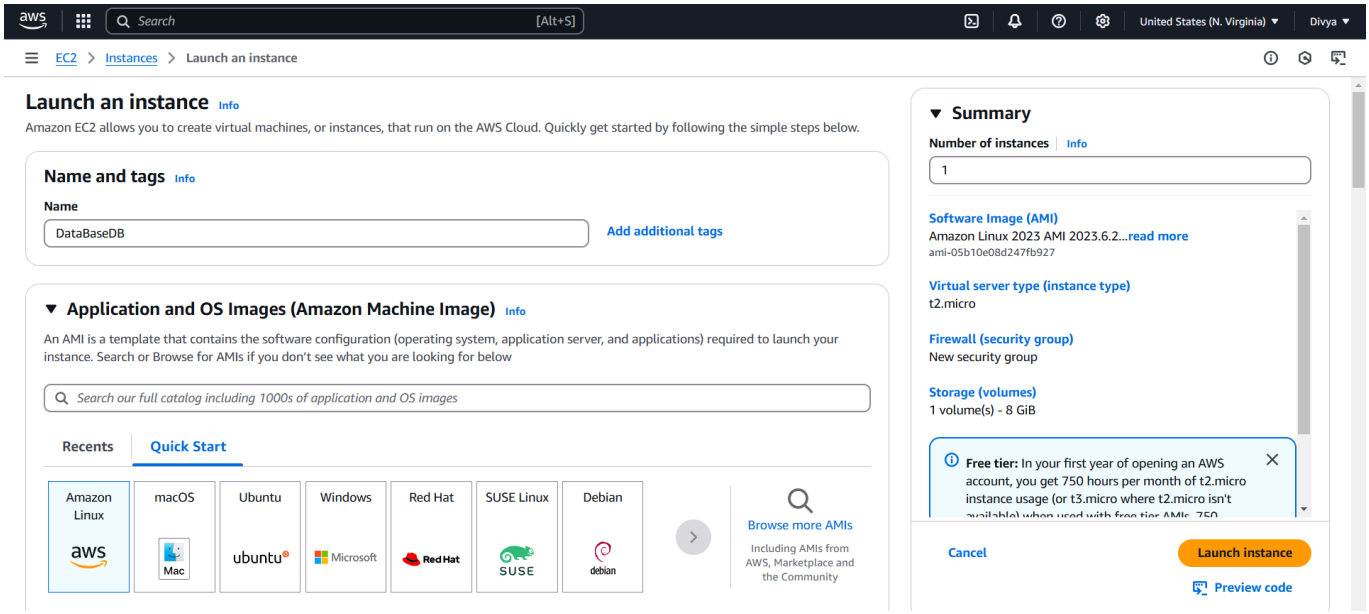


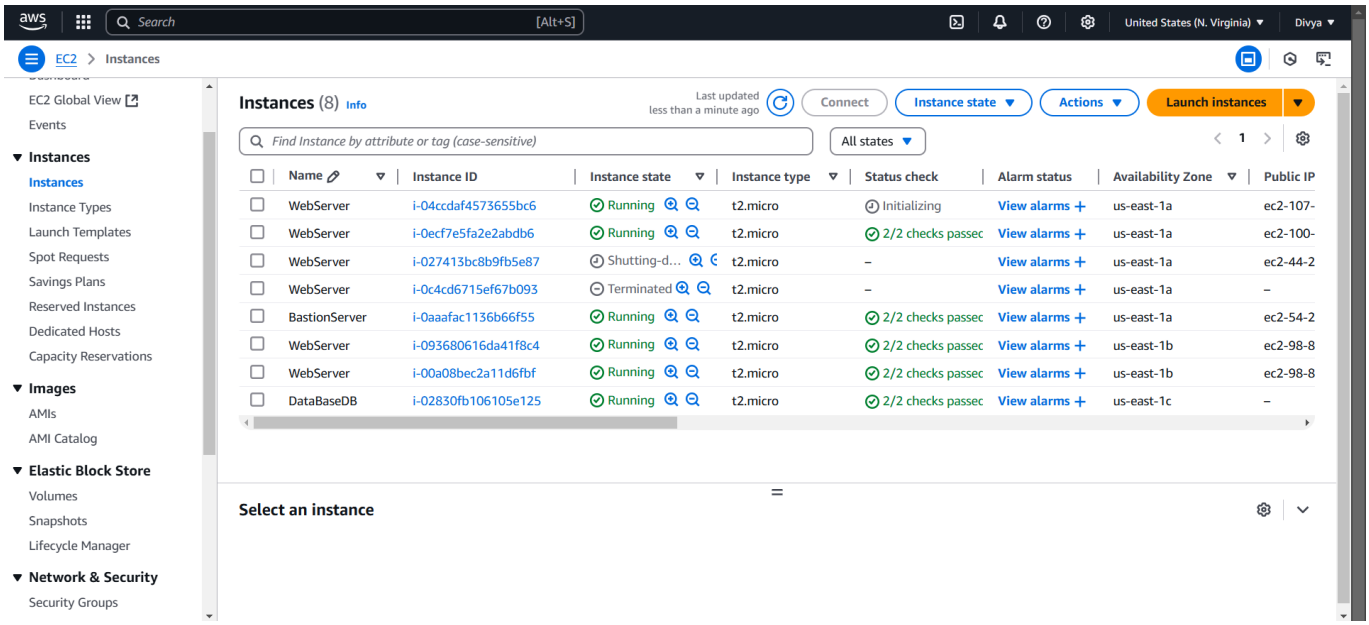
After creating the Auto Scaling Group (MyASG), four EC2 instances are automatically launched based on the defined scaling policies. These instances are evenly distributed across Availability Zones A and B, ensuring high availability and fault tolerance.

20. Setting Up Bastion and Database Servers

A Bastion Server is deployed in MyVpc within Subnet-A to securely manage access to private instances. It is configured with an inbound rule allowing SSH access only from MyIP, ensuring restricted and controlled entry. The Database Server is provisioned in Subnet-C, which is a private subnet with public IP disabled by default for enhanced security. Remote access to the database is only allowed via custom SSH from the Bastion Server's private IP, while MySQL connections are restricted to 10.0.1.0/24 and 10.0.2.0/24, ensuring secure and controlled communication within the network.

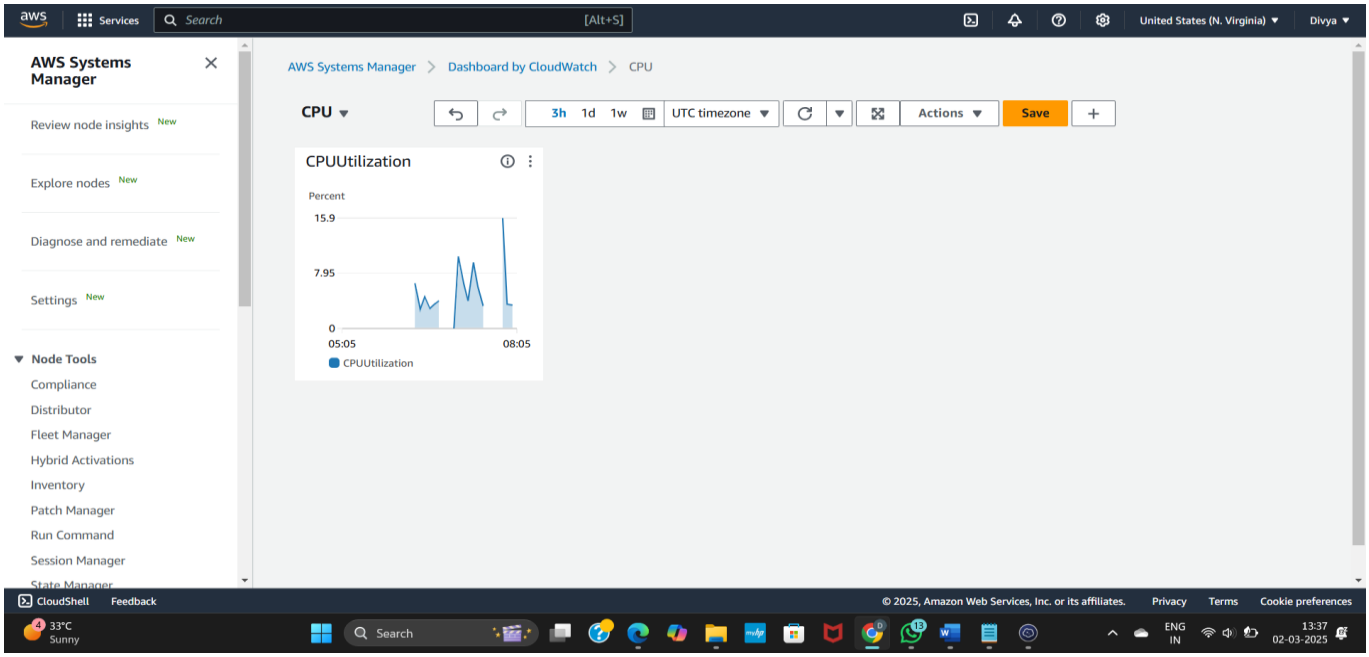






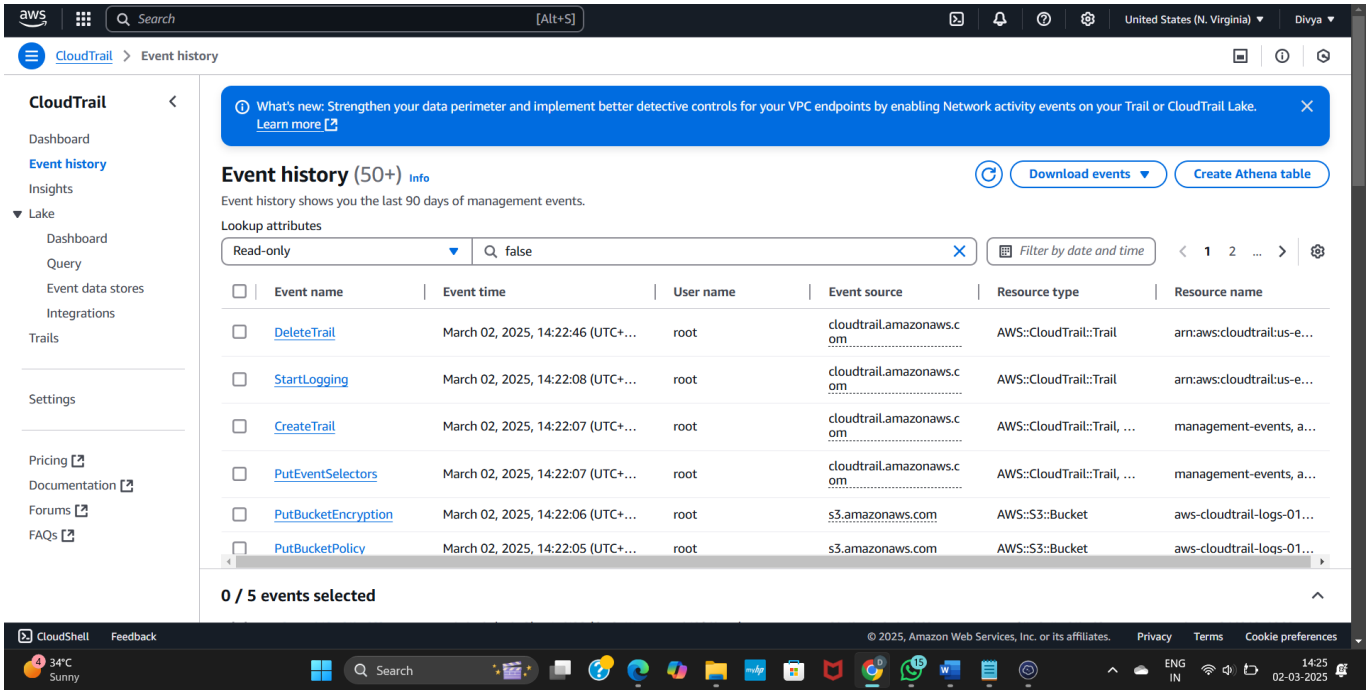
21.Cloud watch

A **CloudWatch Dashboard** is created to monitor the **CPU utilization** of **EC2** instances launched by **MyASG** (Auto Scaling Group). The process begins by creating a new dashboard in the CloudWatch Console and adding a Stacked Area widget for clear visualization. Under EC2 metrics, the monitoring is set to track instances by Auto Scaling Group, selecting MyASG and its CPU Utilization metric. Once configured, the dashboard is created and saved, ensuring real-time insights into resource performance, optimizing scaling decisions, and maintaining efficient workload management.



22. Cloud Trail

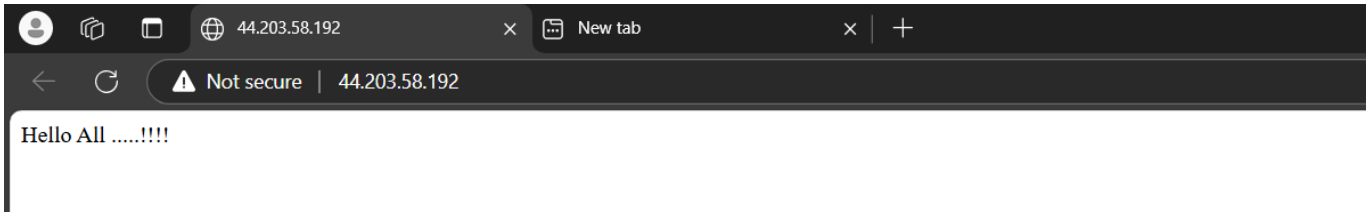
Navigating to the AWS **CloudTrail** Dashboard and accessing the **Event History** without creating a trail provides an overview of recent AWS account activity, including service usage and IAM actions. The Event History displays the last 90 days of recorded management events, such as API calls related to EC2, IAM, and S3. Since no CloudTrail trail has been created, events are only retained temporarily and are not stored for long-term auditing or analysis. Additionally, detailed insights such as **S3 object-level tracking** or **anomaly detection** are unavailable, limiting visibility into specific data access and security events.



23.Checking

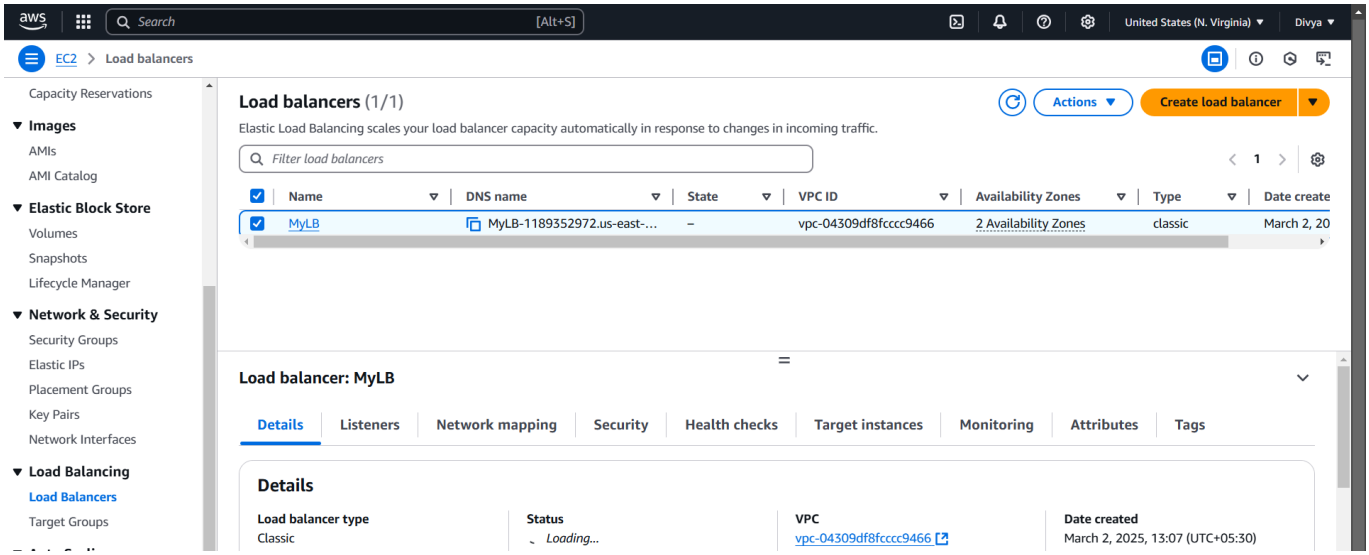
23.1 Checking WebServer

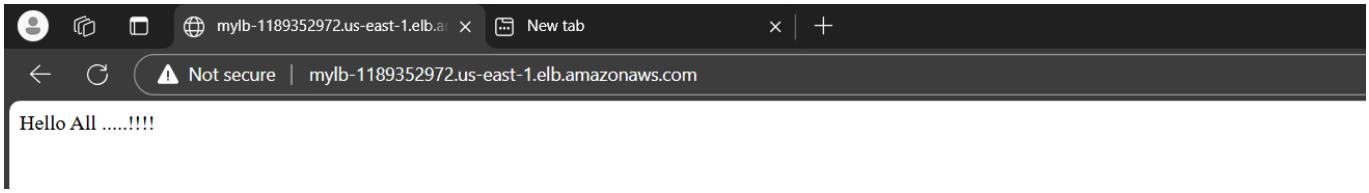
One of the web servers, with the public IP 44.203.58.192, was accessed by pasting it into the browser. The webpage loaded successfully, confirming that the web server is operational, the Apache (HTTPD) service is running, and the security group and routing configurations allow external HTTP traffic. This verifies that the Auto Scaling Group has properly launched the instance and is serving web requests as expected.



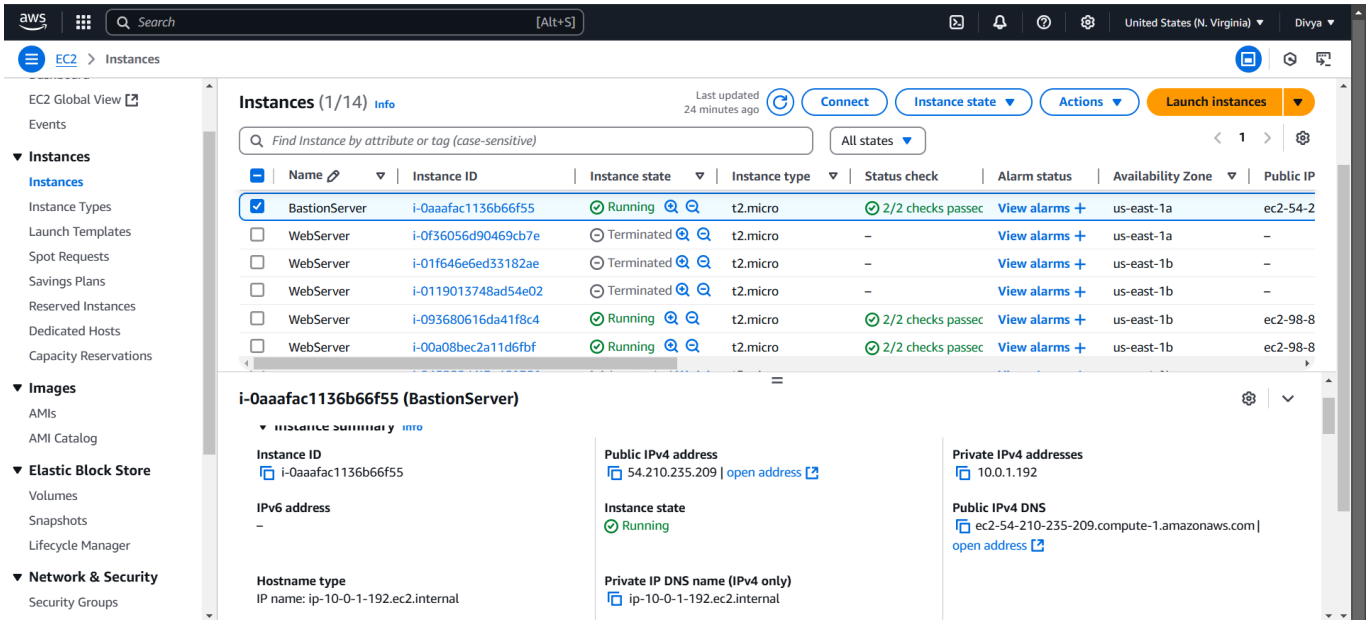
23.2 Checking Load Balancer

The Load Balancer (LB) DNS was pasted into the web browser, and the webpage loaded successfully. This confirms that the Load Balancer is correctly distributing traffic across the web servers, the target instances are healthy, and the security groups and routing configurations are properly set up. This also ensures that the Auto Scaling Group is functioning as expected, dynamically managing instances to handle incoming traffic efficiently.





23.3 Checking Bastion



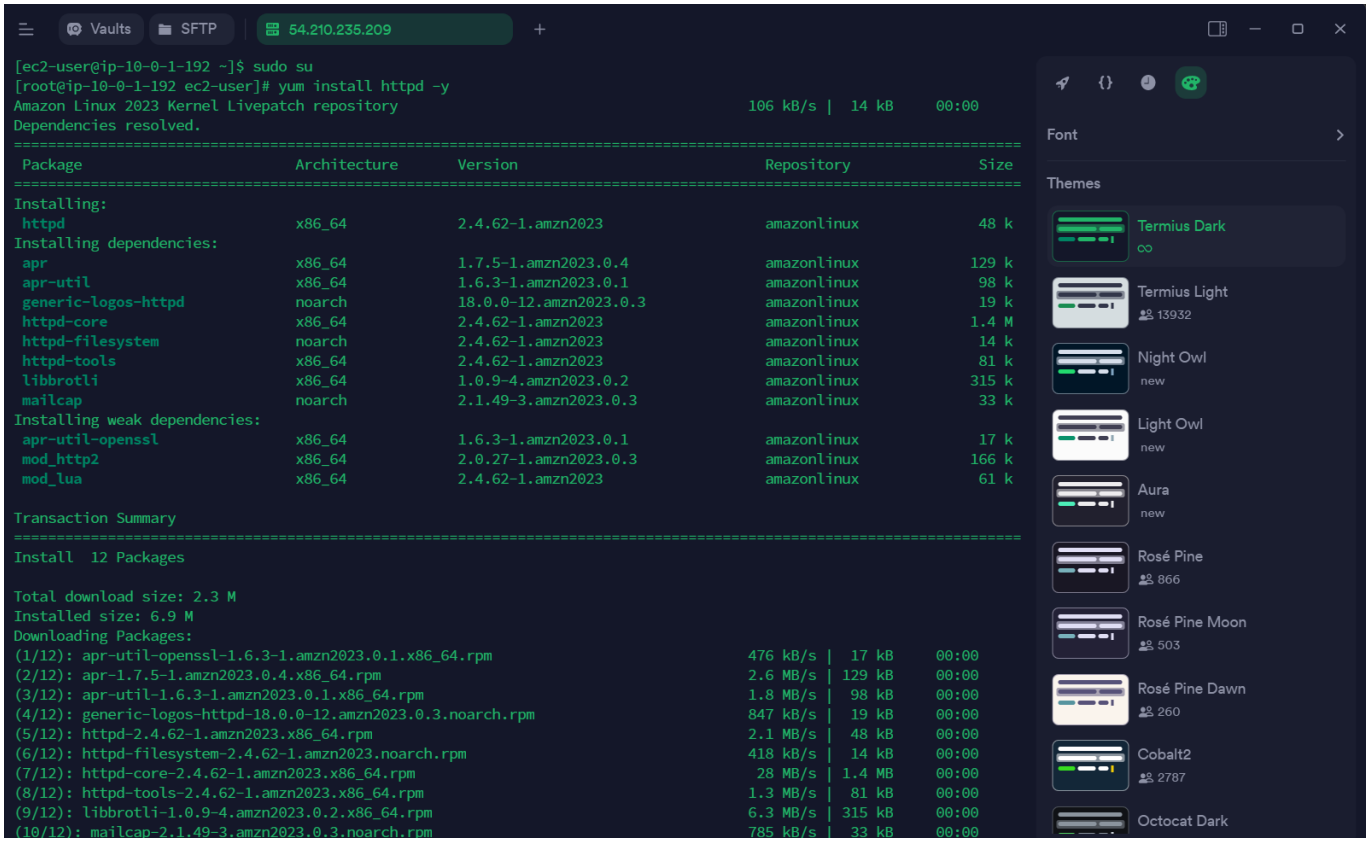
Connected to the Bastion Server using its public IP (54.210.235.209) in Termius, utilizing the projectKey for secure SSH access.

After connecting to the Bastion Server, internet access was verified by executing the following commands:

• sudo su

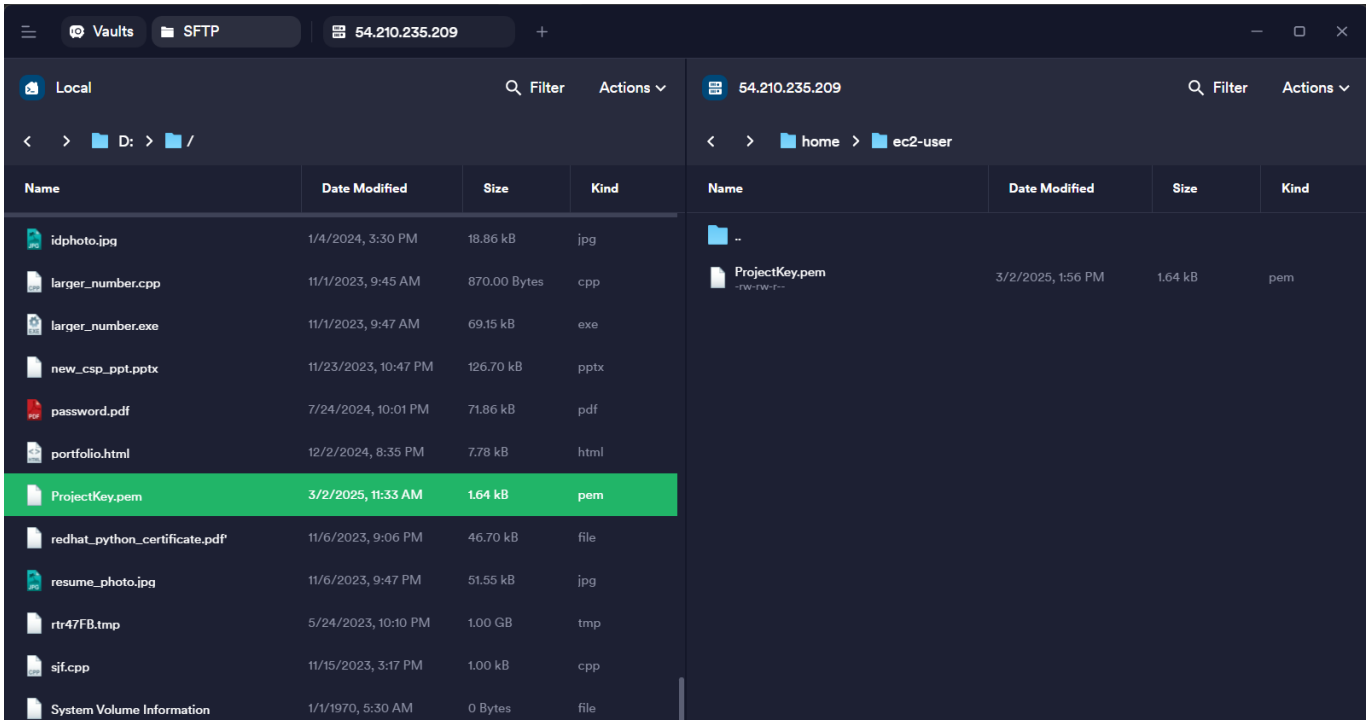
• yum install httpd -y

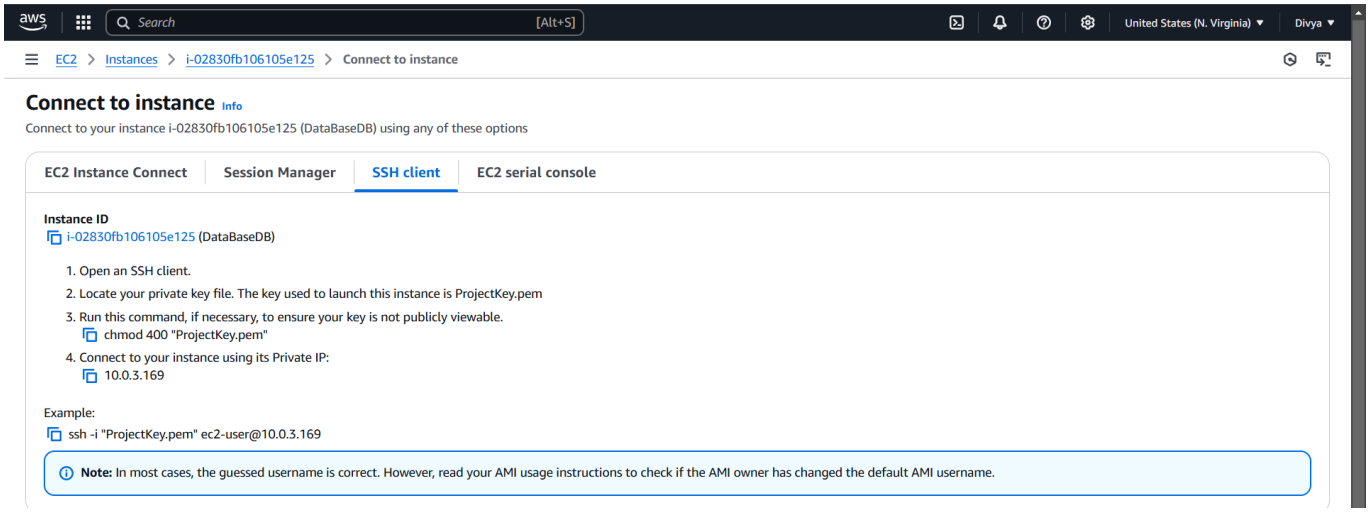
The installation was completed successfully, confirming that the Bastion Server has outbound internet access and can download and install packages from external repositories.



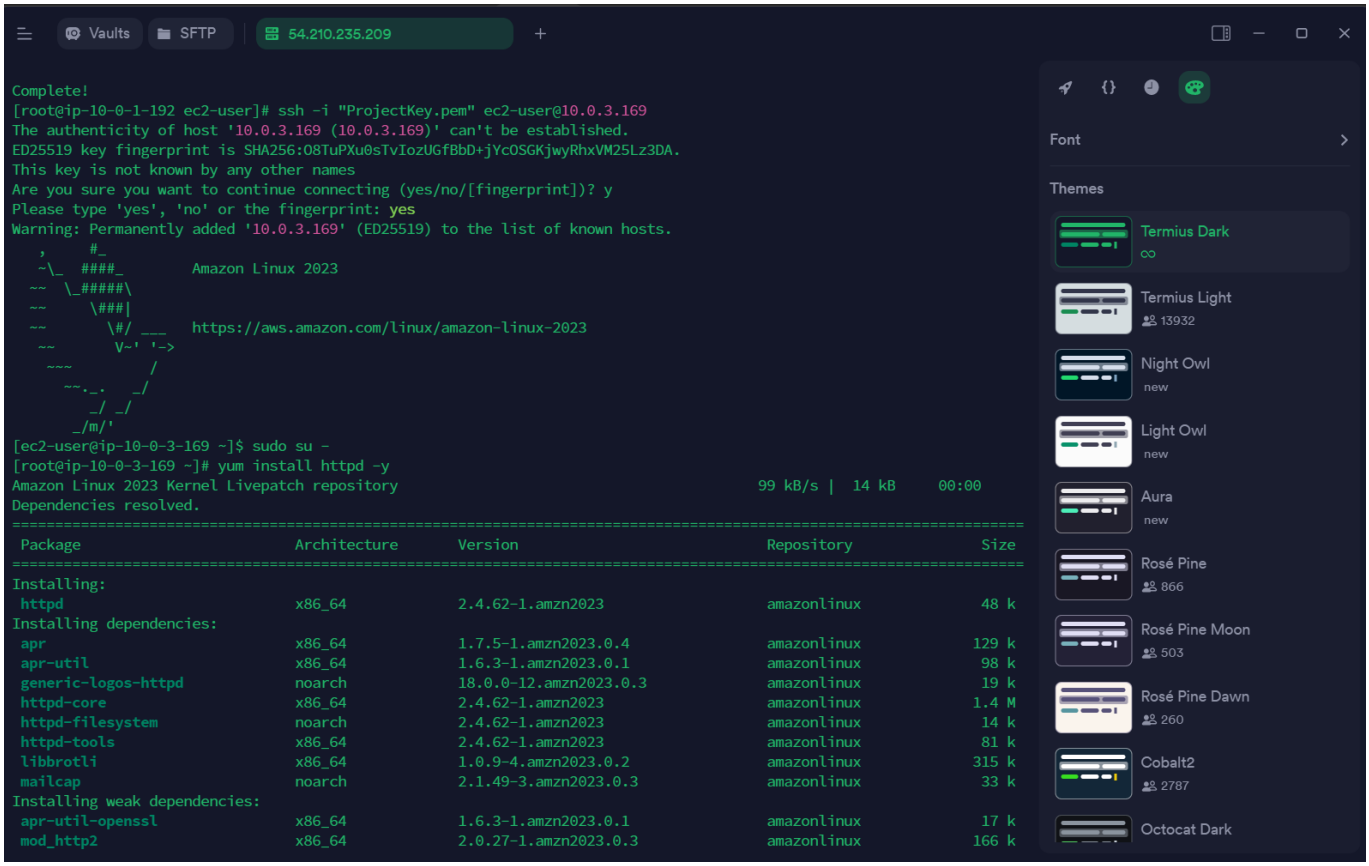
23.4 Checking DatabaseDB

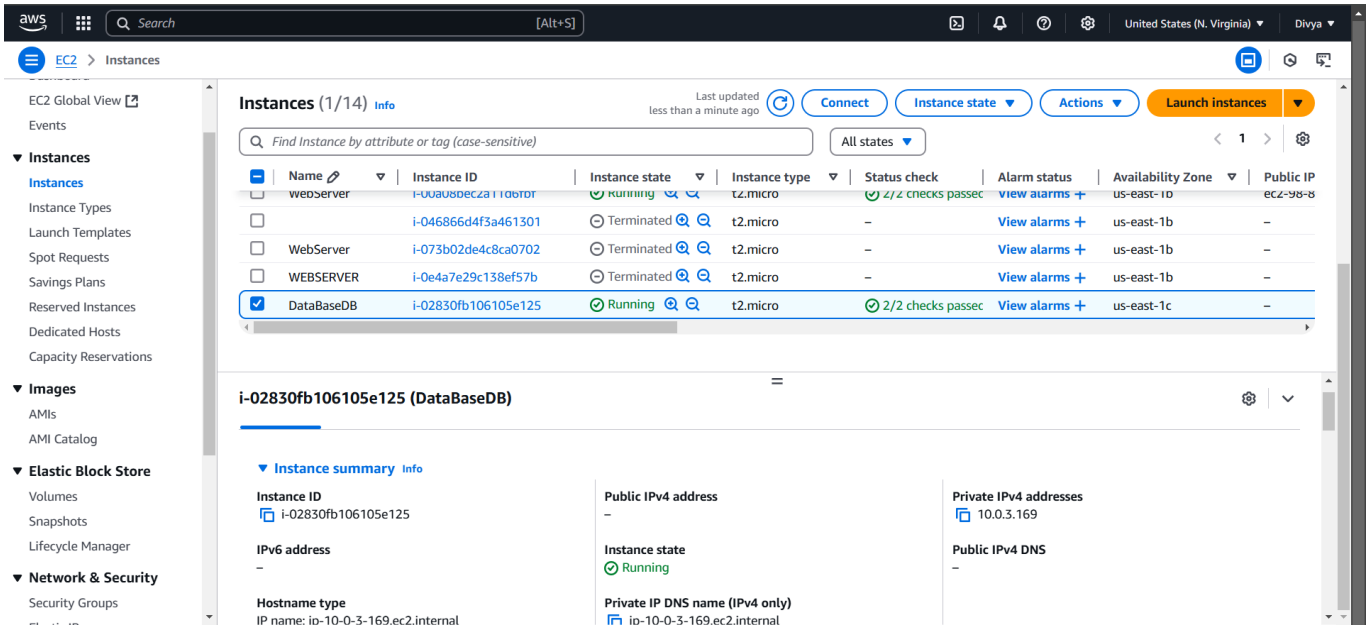
The projectKey was securely **uploaded** to the Bastion Server from the local system using SFTP (Secure File Transfer Protocol). This was done to facilitate secure SSH access to the Database Server from the Bastion.



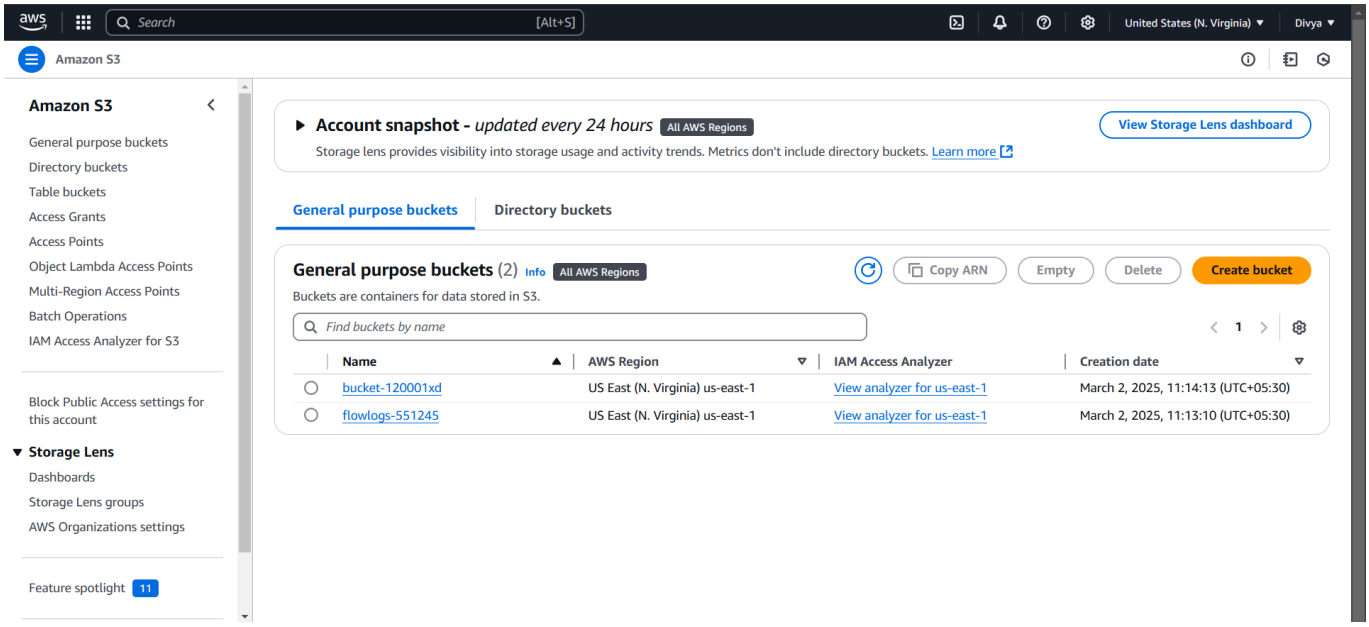


Connected to **DataBaseDB(10.0.3.169)** using the SSH command with the **ProjectKey.pem**. To verify internet access, attempted to install **HTTPD** using sudo yum install httpd -y. The installation completed successfully, confirming that **DataBaseDB** has outbound internet connectivity.

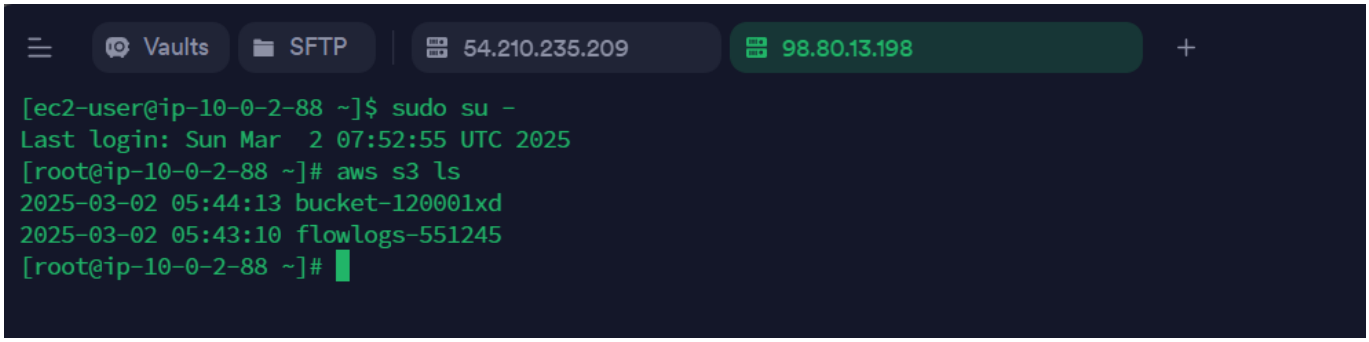




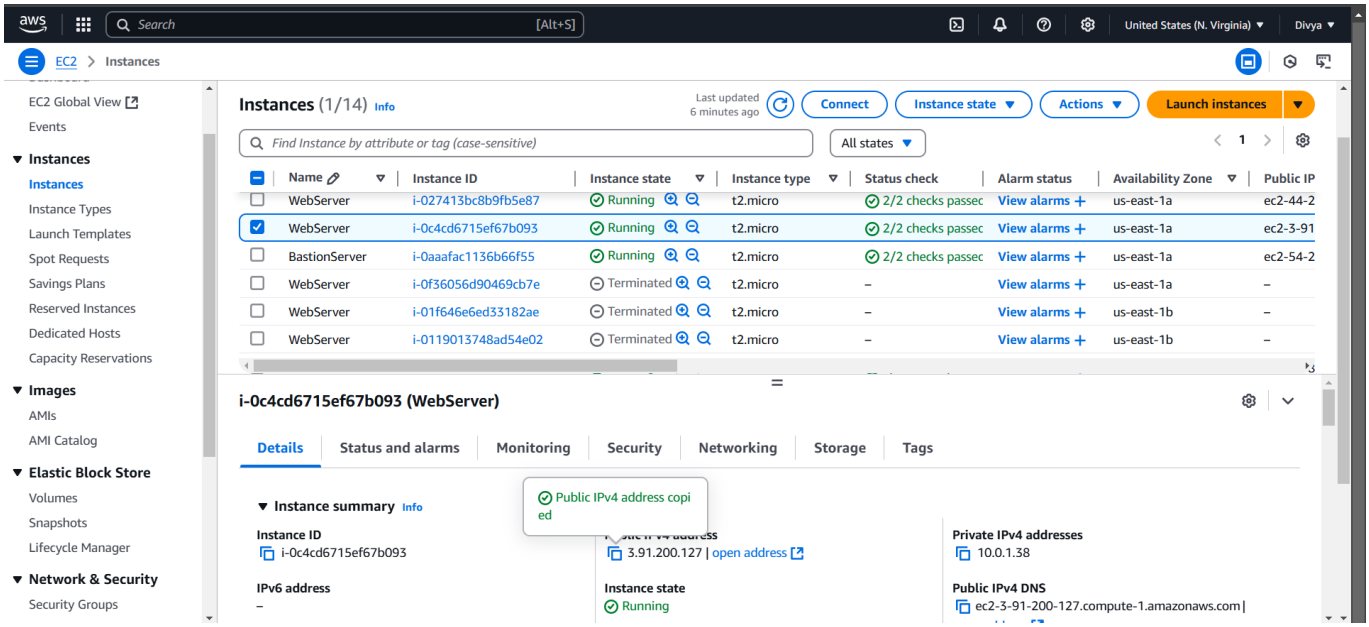
23.5 Checking S3 files & website are accessing



After connecting to a web server using its public IP in Termius, the command aws s3 ls was executed to list the available S3 buckets associated with the AWS account. The output displayed the names and timestamps of the buckets, confirming that the instance has the necessary AWS credentials and permissions to access Amazon S3.

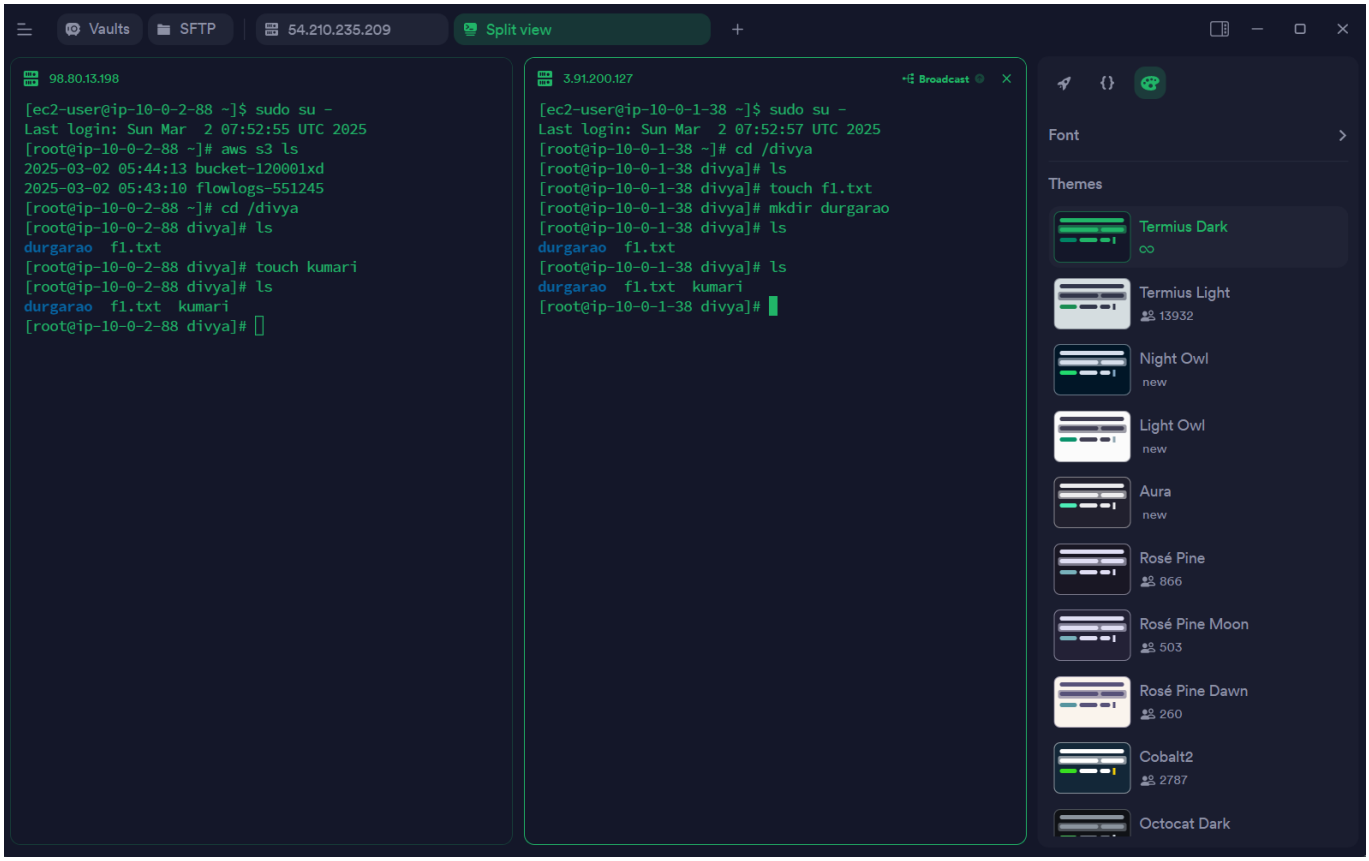


23.6 Checking EFS Storage on EC2

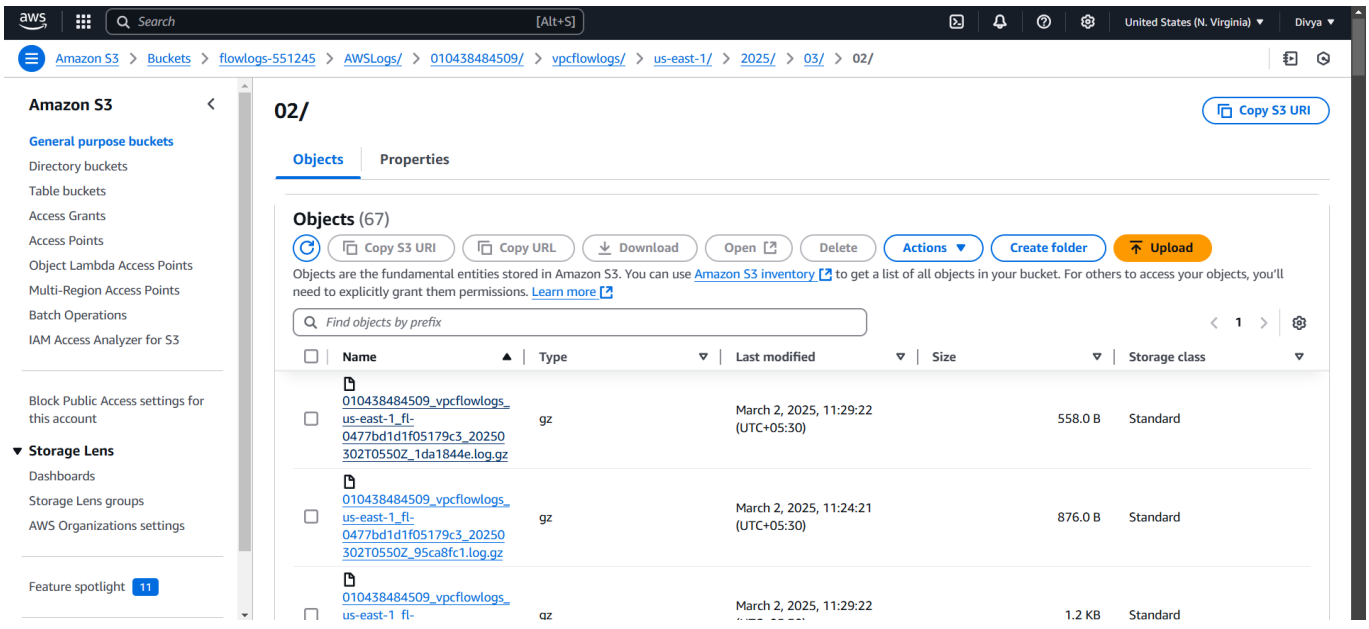


- EFS allows multiple EC2 instances to access the same file system simultaneously.
- Since both servers (connected via public IPs 98.80.13.198 and 3.91.200.127) are showing the same files (f1.txt, kumari, durgarao) in /divya, it confirms they are mounted to the same EFS volume.
- Any change made on one server (e.g., creating a file or directory) is instantly reflected on the other, proving shared storage.

This setup is commonly used for scalable and high-availability architectures, ensuring seamless file access across multiple instances.



23.7 Checking Flow Logs of MyVpc



24.Conclusion

In this project, we successfully built a scalable, secure, and highly available AWS cloud infrastructure using multiple AWS services. The deployment included networking, security, compute, storage, monitoring, and automation components, ensuring efficient resource management and seamless application hosting.

Key highlights of the project:

- Secure and Scalable Network Design: Implemented VPC, subnets, Internet Gateway, NAT Gateway, and NACLs to ensure controlled and secure traffic flow.
- Optimized Compute and Storage: Used EC2, Auto Scaling, Load Balancer, S3, and EFS for flexible, scalable, and fault-tolerant application hosting.
- Robust Security & Access Control: Configured IAM roles, Security Groups, and Bastion Host to enhance security while maintaining operational efficiency.
- Effective Monitoring & Logging: Enabled CloudWatch, CloudTrail, and VPC Flow Logs for real-time tracking, security auditing, and performance optimization.
- Automation & Notifications: Integrated Launch Templates, Auto Scaling Group, and SNS for dynamic resource management and event-driven alerts.

This AWS infrastructure ensures **high availability, fault tolerance, and secure access control**, making it a **reliable and scalable cloud solution** for hosting business applications.

Overall, this project demonstrates how **AWS cloud services can be leveraged to build a robust and production-ready infrastructure**, empowering businesses with **cost-effective, scalable, and secure cloud solutions**.