Expert Cloud Consulting
Enhance Optimise & Scale
ASCP GPUonCLOUD Pvt Ltd

“Expert Cloud Consulting” -

Introduction to Infrastructure as Code (IaC)

03. January .2025

version 1.0

—

Contributed by Prasad Bandagale
Approved by (In review)
Expert Cloud Consulting
Office #811, Gera Imperium Rise,
Hinjewadi Phase-II Rd, Pune, India – 411057

“Expert Cloud Consulting”
Introduction to Infrastructure as Code (IaC)

1.0 Contents

1.0 Contents] 1
2.0 General Information: 2
2.1 Document Jira/ Github Ticket(s) 2
2.2 Document Purpose 2
2.3 Document Revisions 2
2.4 Document References 2
3.0 Document Overview: 3
4.0 Steps / Procedure 4
4.1 CloudFormation Implementation 4
4.1.1 Template Structure 4
4.2: Terraform Implementation. 9
4.3: Deployment Steps for Terraform template 10
5 Annexure - I 17

2.0 General Information:

2.1 Document / Github URL(s)

Ticket(s) Name	Url
Introduction to Infrastructure as Code (IaC)	https://github.com/Prasad-b-git/Weekly-task

2.2 Document Purpose

This manual provides comprehensive guidelines for implementing infrastructure as code using both AWS CloudFormation and Terraform. It covers the creation of cloud resources including S3 buckets, Lambda functions, VPC architecture, and EC2 instances.

2.3 Document Revisions

Date	Version	Contributor(s)	Approver(s)	Section(s)	Change(s)
03/01/25	1.0	Prasad Bandagale	In Review	All Sections	New Document Created

2.4 Document References

The following artifacts are referenced within this document. Please refer to the original documents for additional information.

Date	Document	Filename / Url
2024	AWS CloudFormation User Guide	https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/
2020	Terraform AWS Provider Documentation	https://registry.terraform.io/providers/hashicorp/aws/latest/docs
2022	AWS Lambda Developer Guide	https://docs.aws.amazon.com/lambda/latest/dg/
2022	Amazon VPC Documentation	https://docs.aws.amazon.com/vpc/latest/userguide/

3.0 Document Overview:

This document outlines two Infrastructure as Code (IaC) implementations:

 AWS CloudFormation template for:

- S3 bucket with versioning
- Lambda function triggered by S3 events
- SNS notifications for S3 events

 Terraform configuration for:

- Multi-tier VPC architecture
- EC2 instances in private subnets
- Application Load Balancer setup
- Auto Scaling Group configuration

4.0 Steps / Procedure

4.1 CloudFormation Implementation

4.1.1 Template Structure

The CloudFormation template is organized into the following sections:

- Parameters
  - Email Address: For SNS notifications
- Resources
  - S3 Bucket with versioning
  - Lambda function
  - SNS Topic
  - IAM Roles and Policies
  -

4.1.3 Resource Details

S3 Bucket Configuration:

```
root@terraform:/home/azureuser/cloudformation# cat s3-lambda-notification.yaml
Resources:
  S3Bucket:
    Type: 'AWS::S3::Bucket'
    DependsOn: LambdaPermission
    Properties:
      BucketName: year24082025
      VersioningConfiguration:
        Status: Enabled
      NotificationConfiguration:
        LambdaConfigurations:
          - Event: 's3:ObjectCreated:*'
            Function: !GetAtt LambdaFunction.Arn
```

Lambda Function Configuration:

```
LambdaFunction:
  Type: 'AWS::Lambda::Function'
  Properties:
    Handler: index.handler
    Role: !GetAtt LambdaExecutionRole.Arn
    Code:
      ZipFile: |
        import json
        import boto3
        import os
        def handler(event, context):
            bucket = event['Records'][0]['s3']['bucket']['name']
            key = event['Records'][0]['s3']['object']['key']
            sns = boto3.client('sns')
            message = f"New file uploaded to bucket {bucket}:\nFile name: {key}"
            sns.publish(
                TopicArn=os.environ['SNS_TOPIC_ARN'],
                Message=message,
                Subject='New S3 Upload Notification'
            )
            return {'statusCode': 200, 'body': json.dumps('Notification sent successfully!')}
    Runtime: python3.9
    Timeout: 10
    MemorySize: 128
    Environment:
      Variables:
        SNS_TOPIC_ARN: !Ref SNSTopic
```

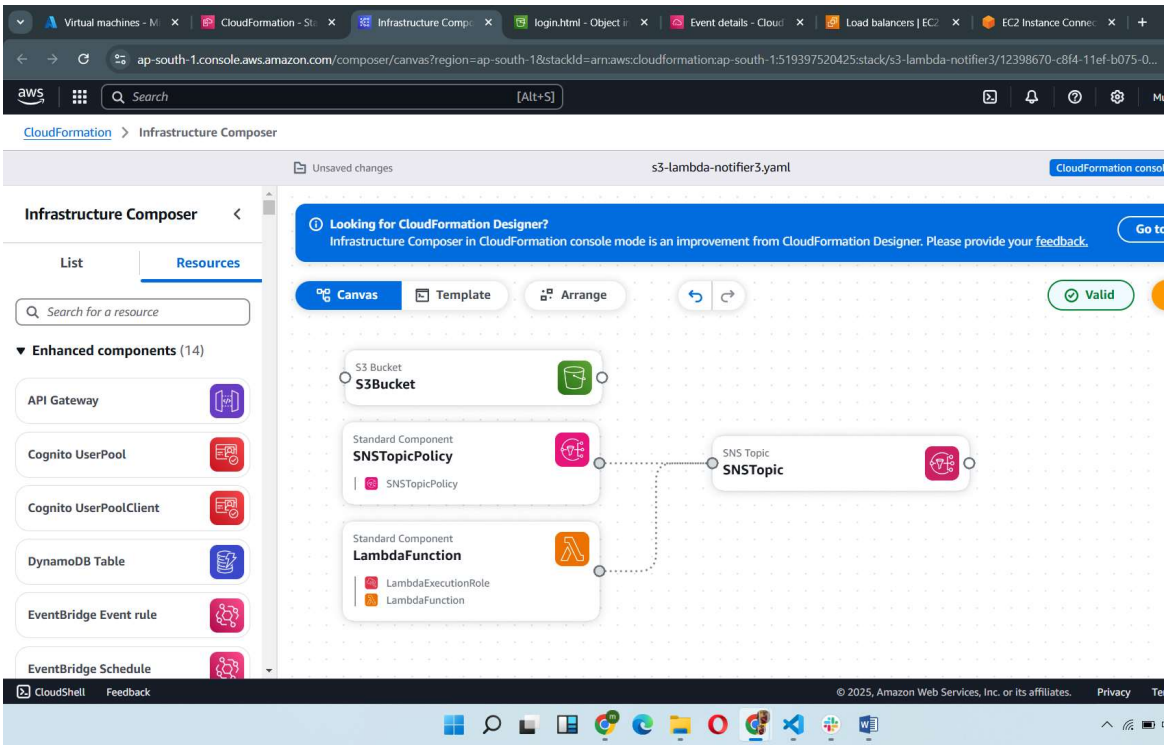
SNS Topic and its policy:

```
SNSTopic:
  Type: 'AWS::SNS::Topic'
  Properties:
    TopicName: 'year24082025-notifications'
    Subscription:
      - Protocol: email
        Endpoint: 'Prasadsb240801@gmail.com'

SNSTopicPolicy:
  Type: 'AWS::SNS::TopicPolicy'
  Properties:
    Topics:
      - !Ref SNSTopic
    PolicyDocument:
      Version: '2012-10-17'
      Statement:
        - Effect: Allow
          Principal:
            Service: lambda.amazonaws.com
          Action: 'sns:Publish'
          Resource: !Ref SNSTopic

LambdaExecutionRole:
  Type: 'AWS::IAM::Role'
  Properties:
```

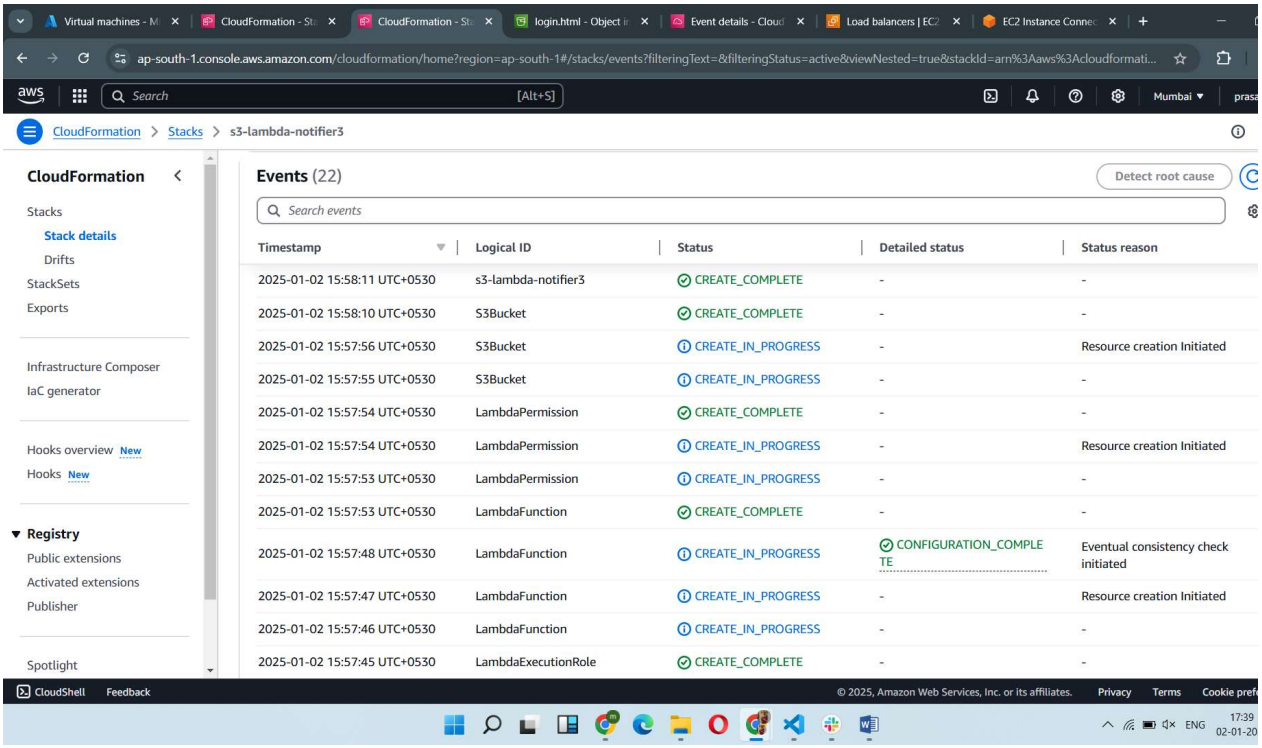
IAM Roles and Policy:

```
LambdaExecutionRole:
  Type: 'AWS::IAM::Role'
  Properties:
    AssumeRolePolicyDocument:
      Version: '2012-10-17'
      Statement:
        - Effect: Allow
          Principal:
            Service: lambda.amazonaws.com
          Action: 'sts:AssumeRole'
    ManagedPolicyArns:
      - 'arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole'
    Policies:
      - PolicyName: SNSPublishPolicy
        PolicyDocument:
          Version: '2012-10-17'
          Statement:
            - Effect: Allow
              Action: 'sns:Publish'
              Resource: !Ref SNSTopic
```

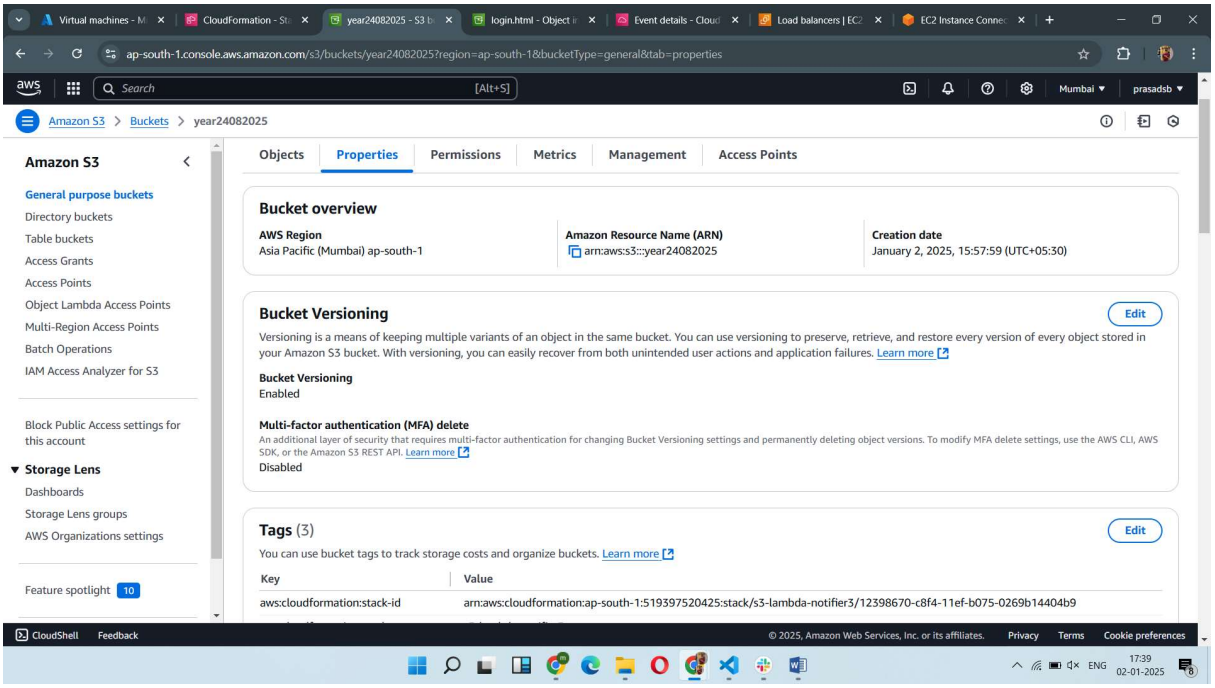
- Reviewing the Resources:
  Infrastructure composer



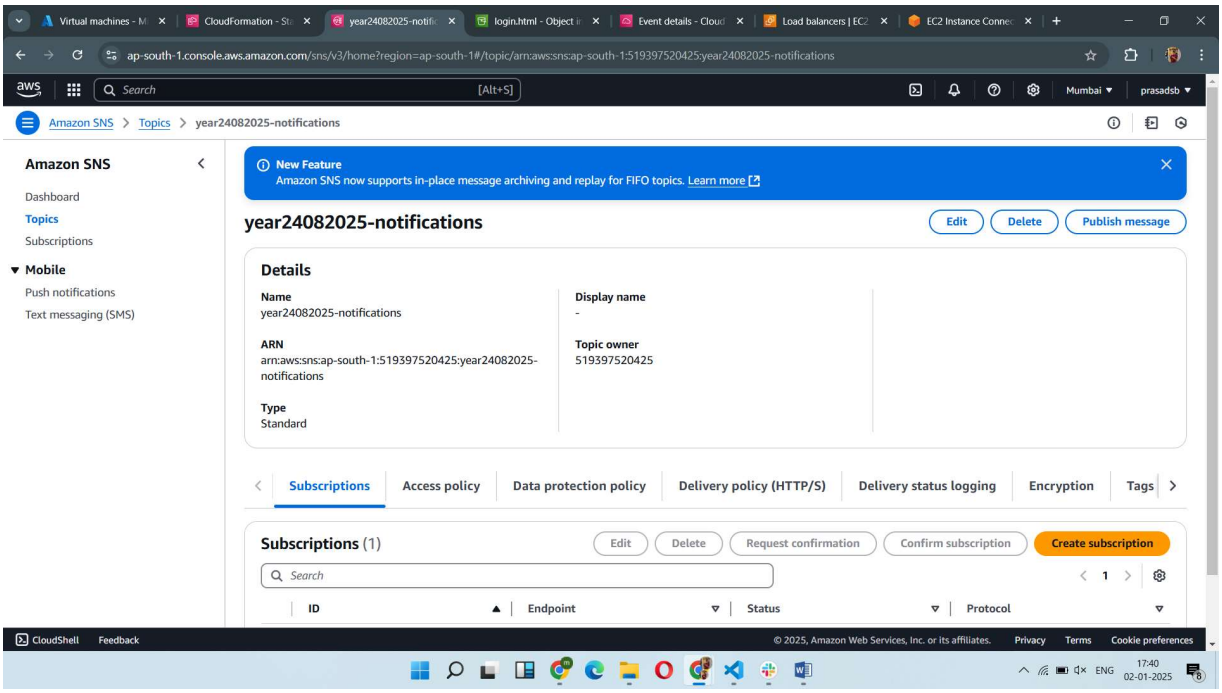
- CloudFormation Events.



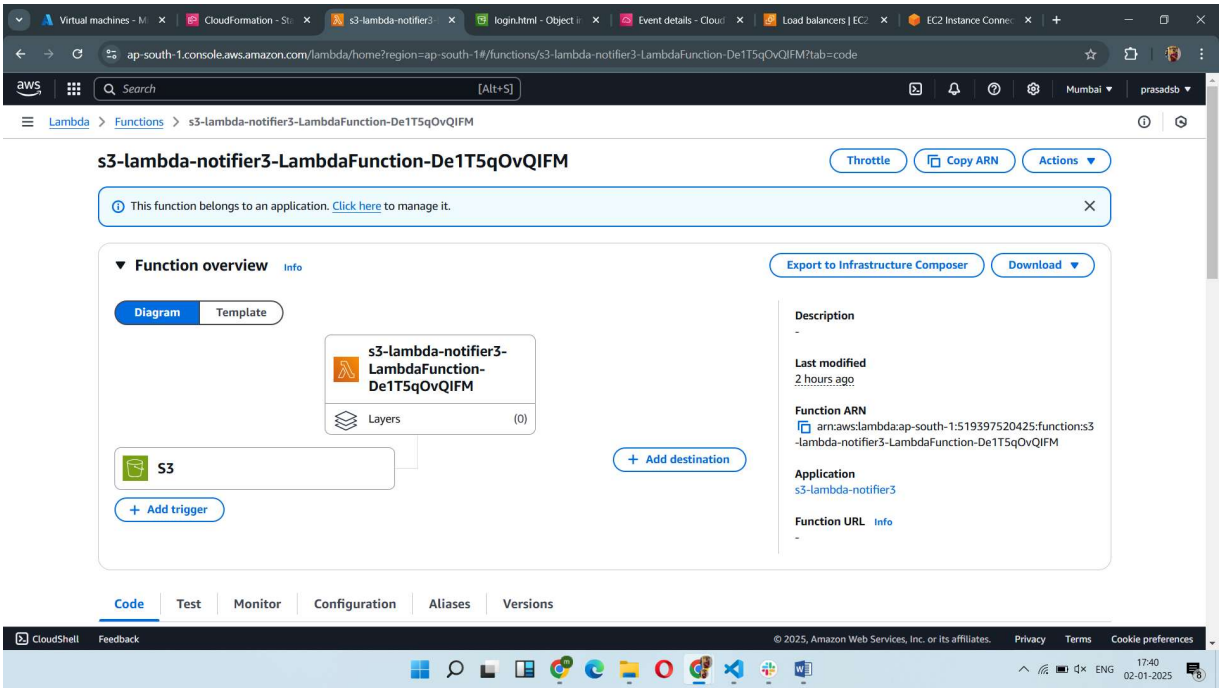
- S3 bucket with versioning



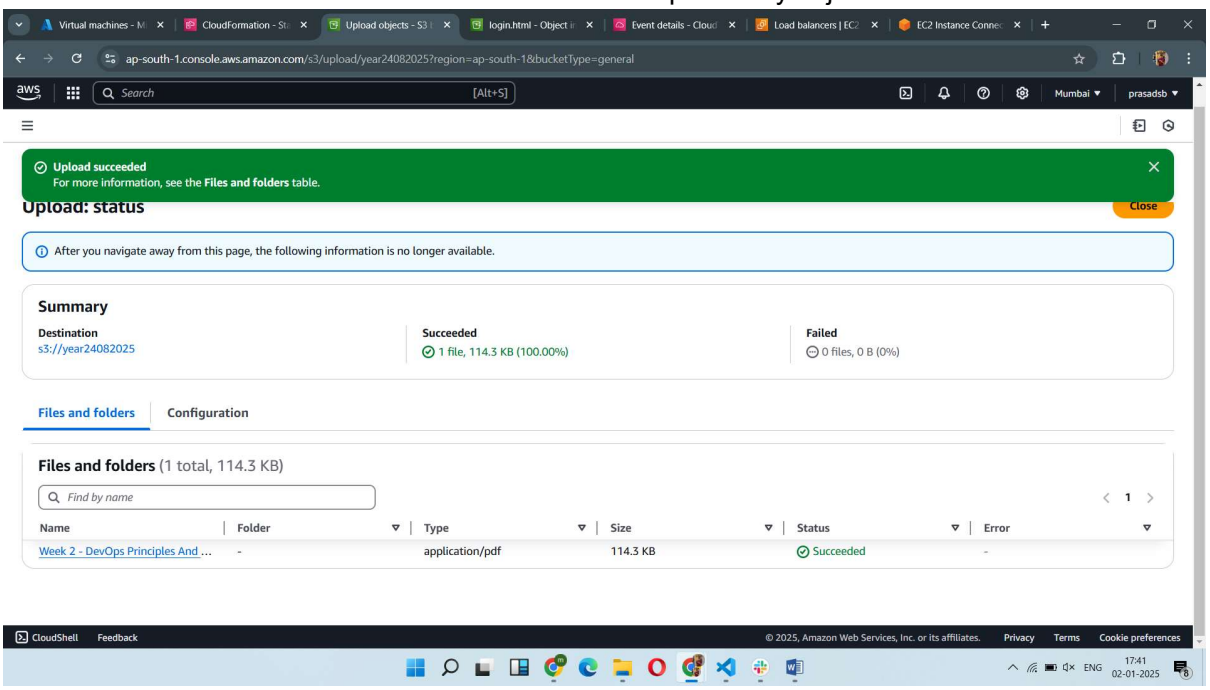
- SNS Service:



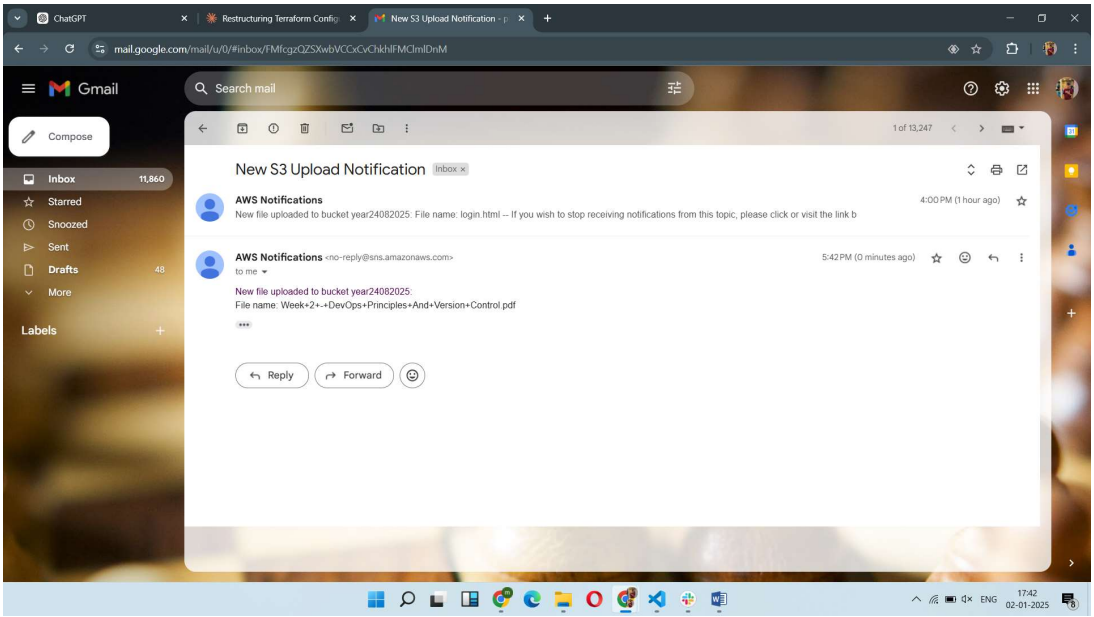
- Lambda Function:



- Let's check if the mail is sent to the user after upload any object to the S3:



- Received notification with name of the object.



4.2: Terraform Implementation.

4.2.1 Project Structure.

The Terraform configuration is organized into modules:

1. VPC
2. Security Groups
3. EC2 Instances
4. Load Balancer

Check it out.

Modules and their configurations files link.

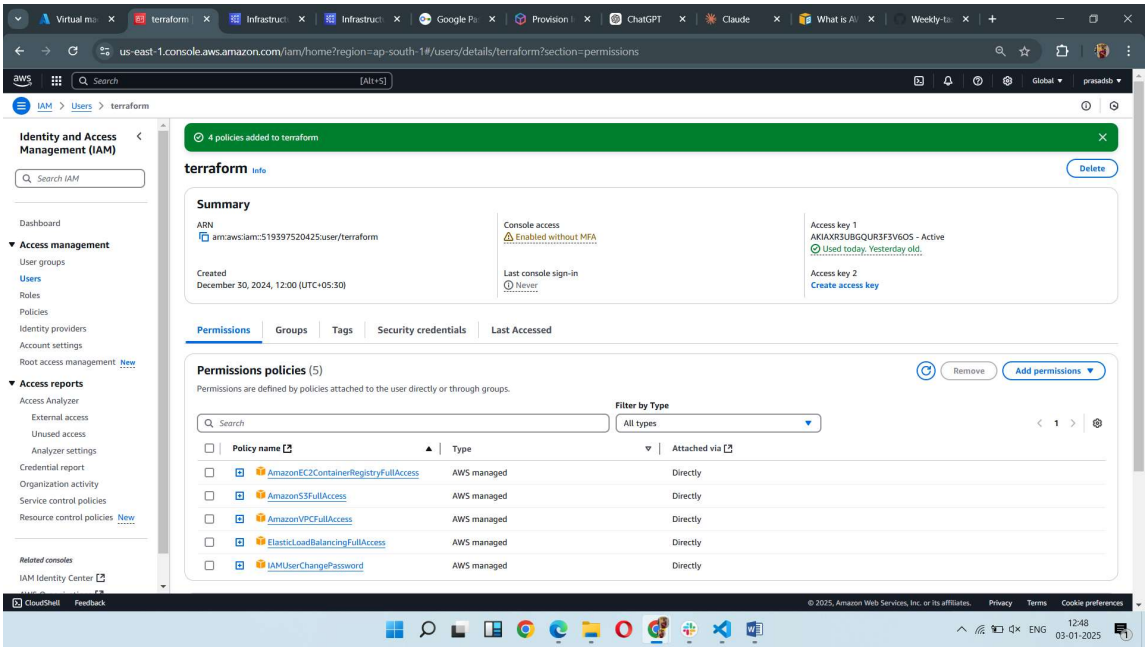
https://github.com/Prasad-b-git/Weekly-task/tree/eaa4be407d62a9357dc88eb888c3b4da4495cf62/03-01-25/terraform

4.3: Deployment Steps for Terraform template

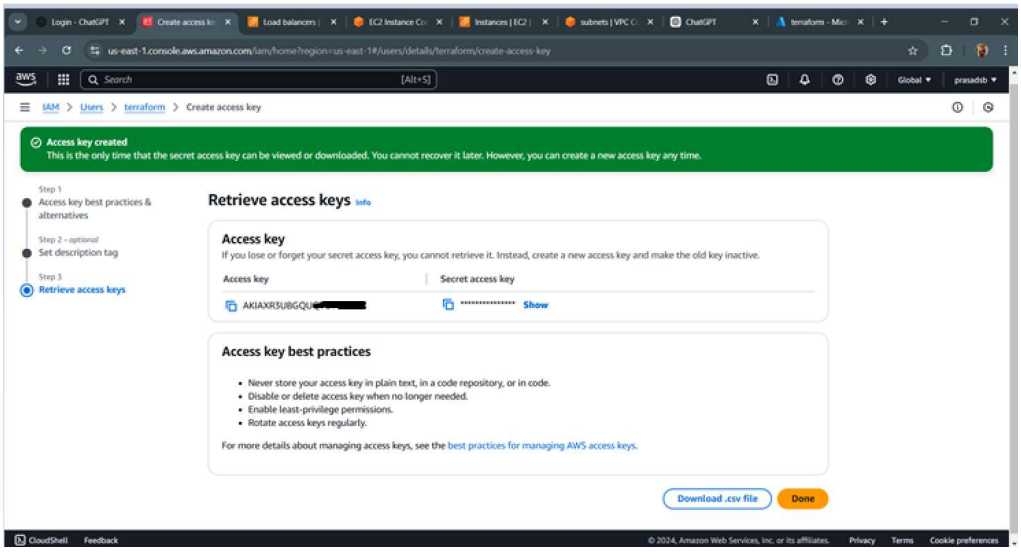
- AWS CLI and Terraform Installation:

AWS CLI is require for configuration of AWS account and to Access the Resources.

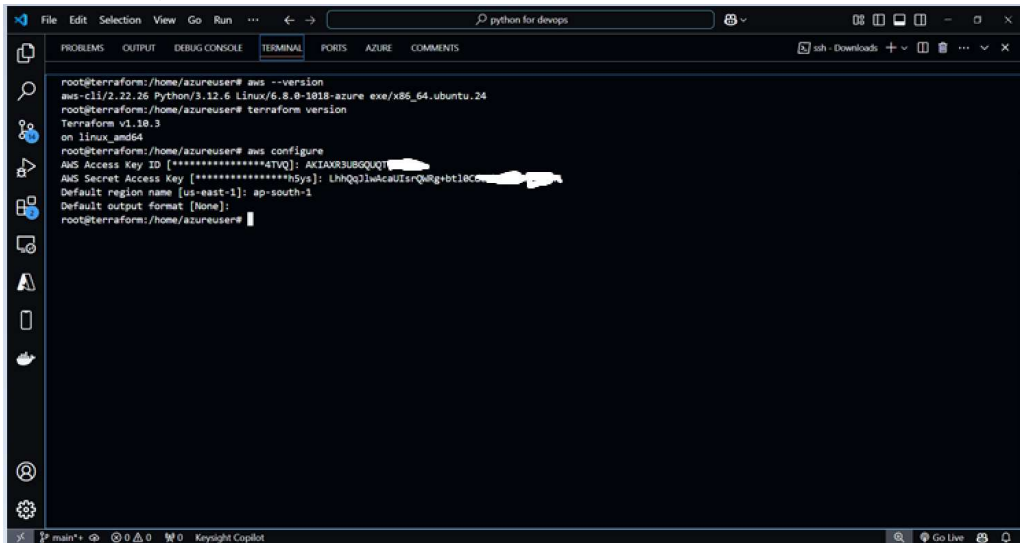
- Create IAM user that has access of required resource like EC2, S3, loadbalancer and VPC.



- Create the access key and Secret Access key for configuration of AWS CLI.



- Configure the AWS CLI with the AWS Account



- Initialize Terraform:

terraform init

```
root@terraform:/home/azureuser/terraform# terraform init
Initializing the backend...
Initializing modules...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.82.2

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
root@terraform:/home/azureuser/terraform#
```

- Review the execution plan:
  terraform plan

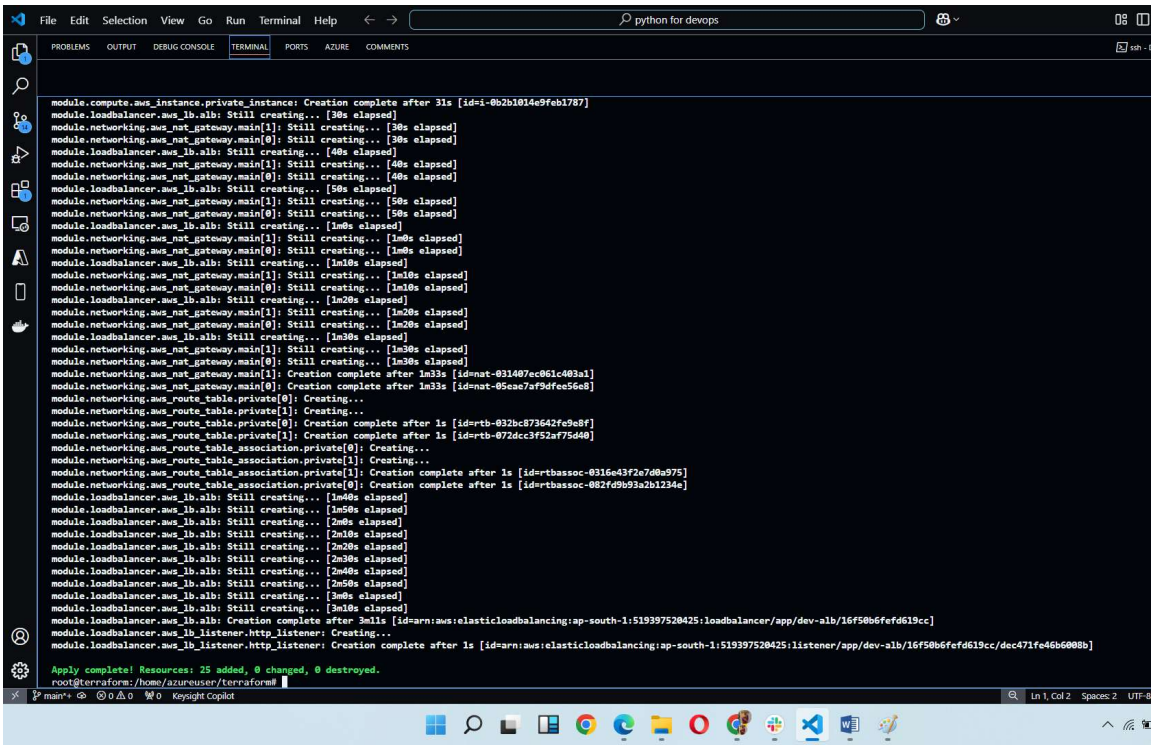
Shows that it will add 25 resources

```
            + self            = false
            + to_port         = 0
              # (1 unchanged attribute hidden)
          },
      ]
      + id                    = (known after apply)
      + ingress               = [
          + {
              + cidr_blocks      = [
                  + "0.0.0.0/0",
                ]
              + from_port        = 22
              + ipv6_cidr_blocks = []
              + prefix_list_ids  = []
              + protocol         = "tcp"
              + security_groups  = []
              + self             = false
              + to_port          = 22
                # (1 unchanged attribute hidden)
            },
          + {
              + cidr_blocks      = [
                  + "0.0.0.0/0",
                ]
              + from_port        = 80
              + ipv6_cidr_blocks = []
              + prefix_list_ids  = []
              + protocol         = "tcp"
              + security_groups  = []
              + self             = false
              + to_port          = 80
                # (1 unchanged attribute hidden)
            },
      ]
      + name                  = "public-sg"
      + name_prefix           = (known after apply)
      + owner_id              = (known after apply)
      + revoke_rules_on_delete = false
      + tags_all              = (known after apply)
      + vpc_id                = (known after apply)
    }

Plan: 25 to add, 0 to change, 0 to destroy.

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.
root@terraform:/home/azureuser/terraform#
```

- Apply the configuration:

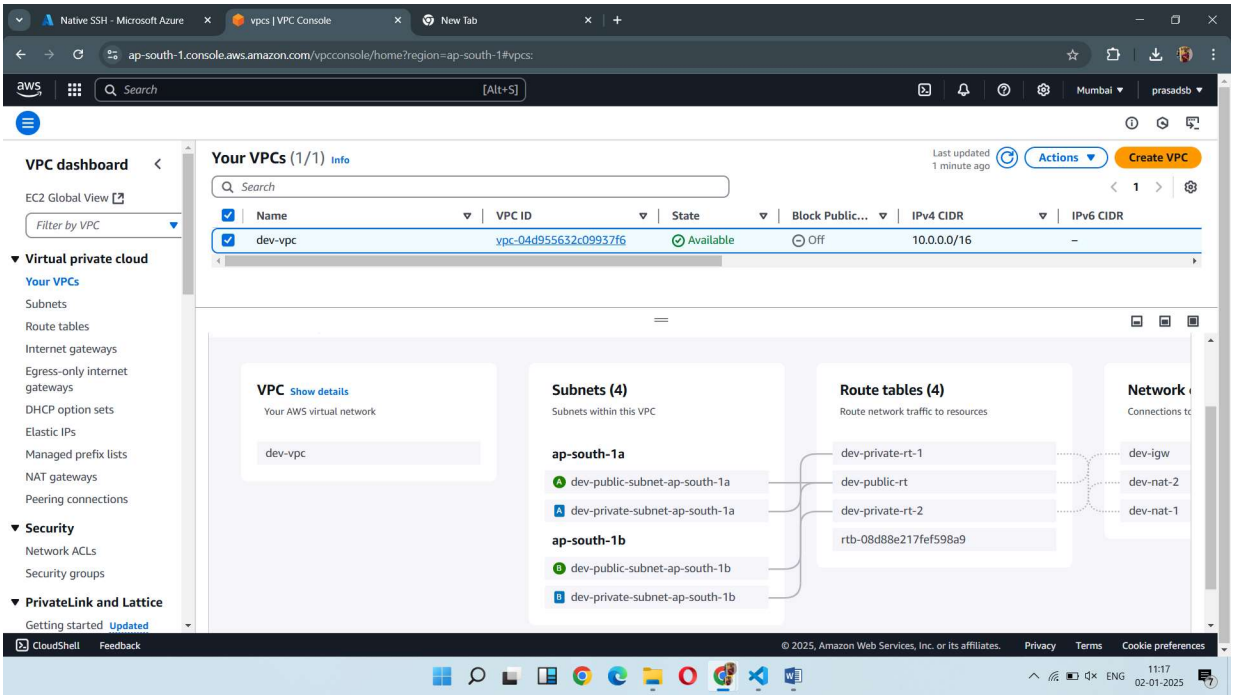
  terraform apply

It has add 25 resources

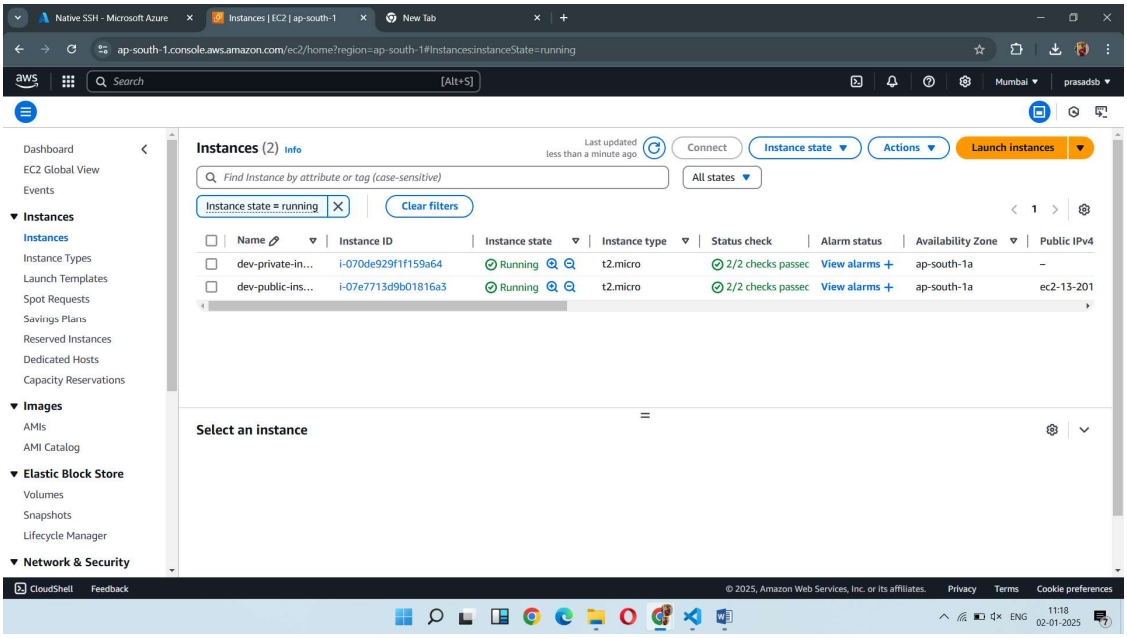


- Review the Resources:

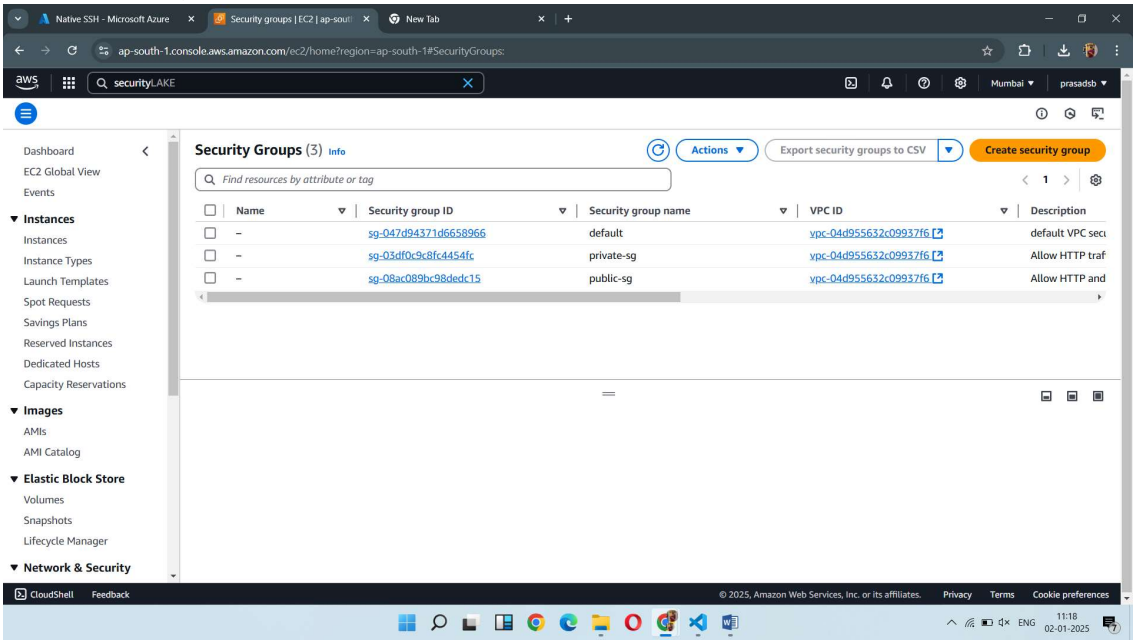
VPC, subnets, Internet gateway, Nat gateway and route tables.



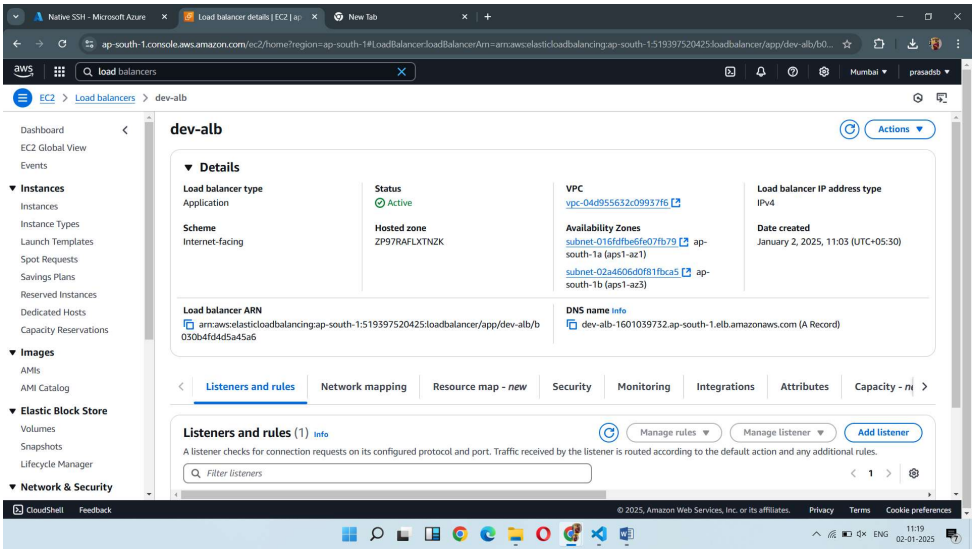
- Two EC2 Instances created, one with public ip and in public subnet and another without public ip and in private subnet.



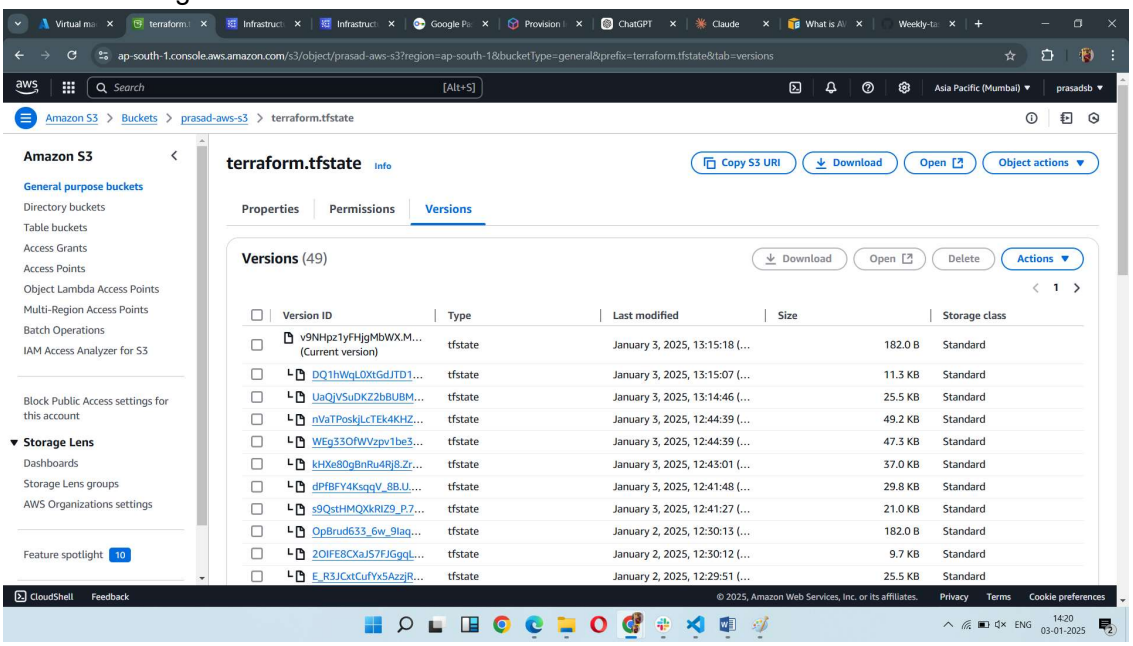
- Security Groups:



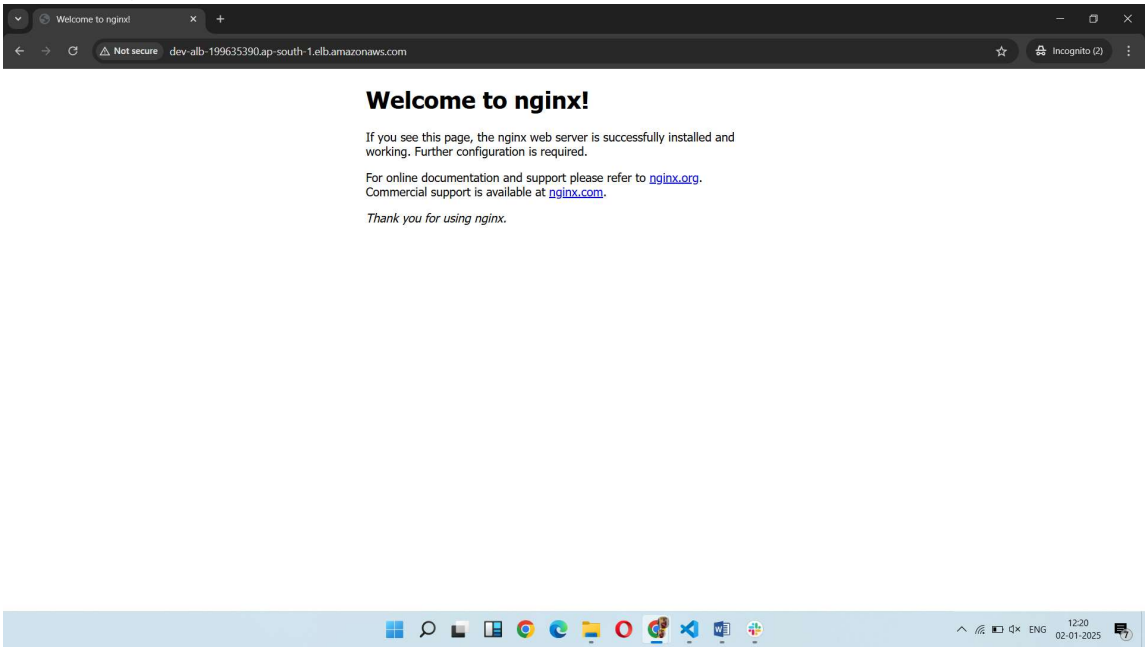
- Application Load Balancer.



- Existing bucket stores the .tfstate file.



- The target group is correctly set. By installing Nginx, the DNS returns the default Nginx page



5.0 Annexure - I

5.1: Testing and Validation

CloudFormation Stack Testing:

Verify S3 bucket creation
Test Lambda function triggers
Confirm SNS notifications

Terraform Infrastructure Testing

Validate VPC networking
Test EC2 instance accessibility
Verify load balancer functionality

5.2: Troubleshooting Guide

Common issues :

1. CloudFormation:
    - Stack creation failures
    - IAM permission issues
    - Lambda function errors
    - S3 creation failures
2. Terraform:
    - State file conflicts
    - Resource dependency issues
    - Network configuration problems