Expert Cloud Consulting
Enhance Optimise & Scale
ASCP GPUonCLOUD Pvt Ltd

“Expert Cloud Consulting” -

Introduction to infrastructure as code (IAC) [Title,18, Arial]

03-01-2025 [Subtitle,14, Arial]

version 1.0

—

Contributed by Shraddha Chaudhari [Normal text,14, Arial]
Approved by (In Review)
Expert Cloud Consulting
Office #811, Gera Imperium Rise,
Hinjewadi Phase-II Rd, Pune, India – 411057

1.0 Contents **[Heading3,14, Arial]**

Contents

2.0 General Information: **[Heading3,14, Arial]**

2.1 Document Jira/ Github Ticket(s) **[Heading4,12, Arial]**

Ticket(s) Name	**Url**
Introduction to infrastructure as code (IAC) **[Normal text,10, Arial]**	https://github.com/shaanicha/Weekly_Tasks/tree/main/30dec-03jan

2.2 Document Purpose

Secure, scalable infrastructure with public/private tiers, high availability, and secure internet access via a load balancer. **[Normal text,10, Arial, Justify Alignment]**

2.3 Document Revisions

Date	**Version**	**Contributor(s)**	**Approver(s)**	**Section(s)**	**Change(s)**
03/01/2025	1.0	Shraddha Chaudhari	Akshay Shinde	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	Official HashiCorp Documentation for terraform installation	https://developer.hashicorp.com/terraform/tutorials/aws-get-started/install-cli
2024	HashiCorp's Official Tutorial - Multi-Tier Applications	https://developer.hashicorp.com/terraform/tutorials/aws/aws-network
2024	Cloudformation	https://linkon.com/cloudformation/getting-started/

2025	Automate cloudformation template	https://aws.amazon.com/blogs/infrastructure-and-automation/best-practices-automating-deployments-with-aws-cloudformation/

3.0 Steps / Procedure

First we need to install Terraform and AWS cli on our local and verify installation by checking version of it on terminal

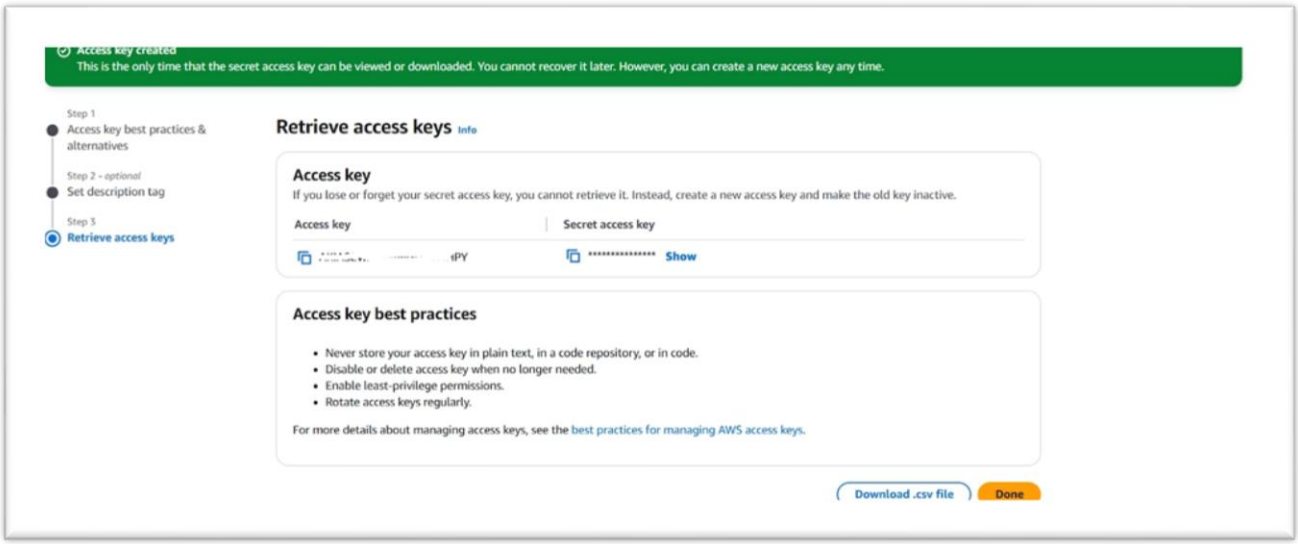
Open command prompt and check version

Command Prompt

```
Microsoft Windows [Version 10.0.18362.1]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\HP>aws --version
aws-cli/2.13.26 Python/3.11.6 Windows/10 exe/AMD64 prompt/off

C:\Users\HP>terraform --version
Terraform v1.10.3
on windows_386

C:\Users\HP>
```

3.1: Create IAM user and give permissions

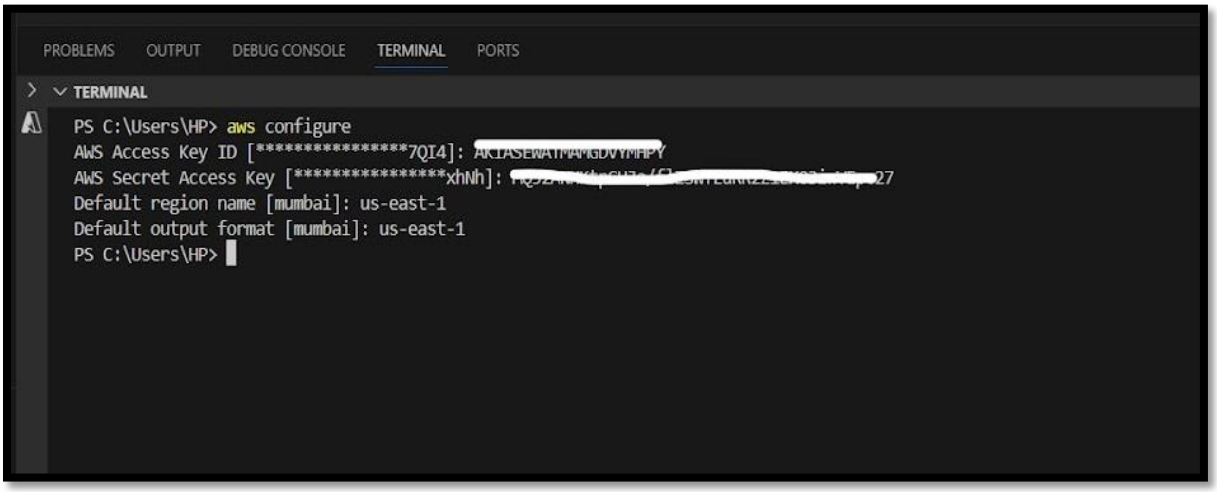
create an access key and a secret access key for login;
for that, we need to go to our AWS account and go inside the IAM user,
create a user, and give permission for EC2, VPC, ALB, Security Group, and S3 permissions,
then create an access key and a secret access key.



3.2: AWS Configuration

Next, we have to configure AWS on VScode for access of AWS on our VScode.

.



4.0: Implementation Steps:

4.1: Project Structure:

```
Terraform_modules/
├── main.tf
├── variables.tf
├── outputs.tf
├── terraform.tfvars
└── modules/
    ├── vpc/
    │   ├── main.tf
    │   ├── variables.tf
    │   └── outputs.tf
    ├── security_group/
    │   ├── main.tf
    │   ├── variables.tf
    │   └── outputs.tf
    ├── ec2/
    │   ├── main.tf
    │   ├── variables.tf
    │   └── outputs.tf
    └── alb/
        ├── main.tf
        ├── variables.tf
        └── outputs.tf
```

1. Set up the project structure:

```
mkdir -p Terraform_modules/modules/{vpc,security_group,ec2,alb}
```

```
PS C:\Users\HP\Desktop\Terraform_modules> cd .\modules\
PS C:\Users\HP\Desktop\Terraform_modules\modules> ls


    Directory: C:\Users\HP\Desktop\Terraform_modules\modules


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        01-01-2025     10:48                alb
d-----        01-01-2025     10:46                ec2
d-----        31-12-2024     17:38                security_group
d-----        01-01-2025     10:44                vpc
```

4.2: Create module files

```
# Create main.tf, variables.tf, and outputs.tf in each module directory
touch modules/vpc/{main.tf,variables.tf,outputs.tf}
touch modules/security_group/{main.tf,variables.tf,outputs.tf}
touch modules/ec2/{main.tf,variables.tf,outputs.tf}
touch modules/alb/{main.tf,variables.tf,outputs.tf}
```

1. VPC module directory:

```
PS C:\Users\HP\Desktop\Terraform_modules\modules> cd .\vpc\
PS C:\Users\HP\Desktop\Terraform_modules\modules\vpc> ls


    Directory: C:\Users\HP\Desktop\Terraform_modules\modules\vpc


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        02-01-2025     09:58           1452 main.tf
-a----        02-01-2025     10:00            585 outputs.tf
-a----        02-01-2025     09:59            536 variables.tf
```

2. Security group module directory:

```
TERMINAL
PS C:\Users\HP\Desktop\Terraform_modules\modules> cd .\security_group\
PS C:\Users\HP\Desktop\Terraform_modules\modules\security_group> ls


    Directory: C:\Users\HP\Desktop\Terraform_modules\modules\security_group


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        02-01-2025     10:54            794 main.tf
-a----        02-01-2025     15:00            185 outputs.tf
-a----        02-01-2025     10:37             74 variables.tf
```

3. EC2 module directory:

```
PS C:\Users\HP\Desktop\Terraform_modules> cd .\modules\
PS C:\Users\HP\Desktop\Terraform_modules\modules> cd .\ec2\
PS C:\Users\HP\Desktop\Terraform_modules\modules\ec2> ls


    Directory: C:\Users\HP\Desktop\Terraform_modules\modules\ec2


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        02-01-2025     11:02           1232 main.tf
-a----        02-01-2025     11:06            333 outputs.tf
-a----        02-01-2025     10:50            303 variables.tf
```

4. ALB module directory:

```
TERMINAL
PS C:\Users\HP\Desktop\Terraform_modules\modules> cd .\alb\
PS C:\Users\HP\Desktop\Terraform_modules\modules\alb> ls


    Directory: C:\Users\HP\Desktop\Terraform_modules\modules\alb


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        01-01-2025     11:56            883 main.tf
-a----        01-01-2025     10:48             58 outputs.tf
-a----        01-01-2025     11:54            358 variables.tf
```

4.3: Create Root Configuration File

Create main.tf: Resource definitions, variables.tf: Input variables, outputs.tf: Output values, terraform.tfvars: Default variable values.

```
touch main.tf variables.tf outputs.tf terraform.tfvars
```

```
PS C:\Users\HP\Desktop\Terraform_modules> ls


    Directory: C:\Users\HP\Desktop\Terraform_modules


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----        02-01-2025     11:03                .terraform
d-----        01-01-2025     11:04                modules
-a----        31-12-2024     17:22           1377 .terraform.lock.hcl
-a----        01-01-2025     11:41            502 main.tf
-a----        01-01-2025     10:42            254 outputs.tf
-a----        02-01-2025     11:09          40614 terraform.tfstate
-a----        02-01-2025     11:08          26690 terraform.tfstate.backup
-a----        01-01-2025     10:41            161 variables.tf
```

Root directory are created and they are ready to use.

4.4: Initialize Terraform

Now, let's do terraform init for initialize terraform

```
terraform init
```

```
TERMINAL

PS C:\Users\HP\Desktop\Terraform_modules\modules> terraform init
Terraform initialized in an empty directory!

The directory has no Terraform configuration files. You may begin working
with Terraform immediately by creating Terraform configuration files.
PS C:\Users\HP\Desktop\Terraform_modules\modules> terraform validate
```

4.5: Validate Configuration

Let's do terraform validate for check configurartion syntax

```
PS C:\Users\HP\Desktop\Terraform_modules> terraform validate
Success! The configuration is valid.
```

4.6: Plan Infrastructure

Let's do terraform plan for previews changes

```
terraform plan
```

```
PS C:\Users\HP\Desktop\Terraform_modules> terraform plan
module.vpc.data.aws_availability_zones.available: Reading...
module.vpc.aws_vpc.main: Refreshing state... [id=vpc-054c6271764d98a8c]
module.ec2.aws_security_group.ec2: Refreshing state... [id=sg-0070be2fdfd23209b]
module.vpc.data.aws_availability_zones.available: Read complete after 1s [id=us-east-1]
module.vpc.aws_internet_gateway.igw: Refreshing state... [id=igw-05e0e5837244fef10]
module.vpc.aws_subnet.public[0]: Refreshing state... [id=subnet-023855e14dc92b5a5]
module.vpc.aws_subnet.public[1]: Refreshing state... [id=subnet-0f47e89568dbd18e6]
module.vpc.aws_subnet.private[0]: Refreshing state... [id=subnet-083de97a751036b30]
module.vpc.aws_subnet.private[1]: Refreshing state... [id=subnet-00d159fdf0c156ee6]
module.alb.aws_security_group.lb: Refreshing state... [id=sg-027e8776a90dd7bfb]
module.ec2.aws_lb_target_group.example: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:147506552856:targetgroup/app-tg/59559d0b491f1b14]
module.vpc.aws_route_table.public: Refreshing state... [id=rtb-02b421bad40eb836e]
module.alb.aws_lb.main: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:147506552856:loadbalancer/app/app-lb/86a03d070e0c4d2f]
module.vpc.aws_route_table_association.public[0]: Refreshing state... [id=rtbassoc-0b1bf0cf0976fec7a]
module.vpc.aws_route_table_association.public[1]: Refreshing state... [id=rtbassoc-008649759a11bab60]
module.alb.aws_lb_listener.http: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:147506552856:listener/app/app-lb/86a03d070e0c4d2f/c6c1c074640fe248]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create
  ~ update in-place
  - destroy

Terraform will perform the following actions:

  # module.ec2.aws_instance.app[0] will be created
  + resource "aws_instance" "app" {
      + ami                                  = "ami-0e2c8caa4b6378d8c"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + enable_primary_ipv6                  = (known after apply)
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + iam_instance_profile                 = (known after apply)
      + id                                   = (known after apply)
```

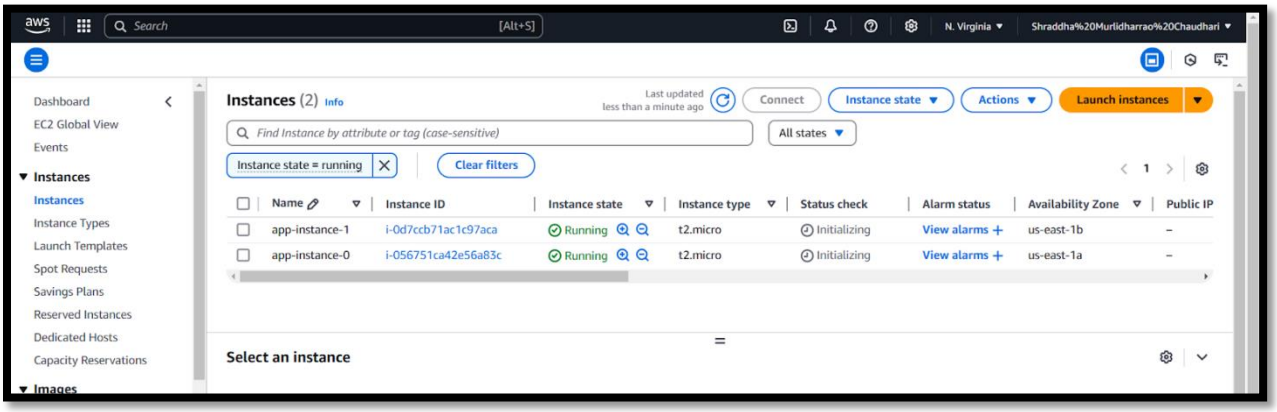
4.7: Apply Configurartion

```
terraform apply
```

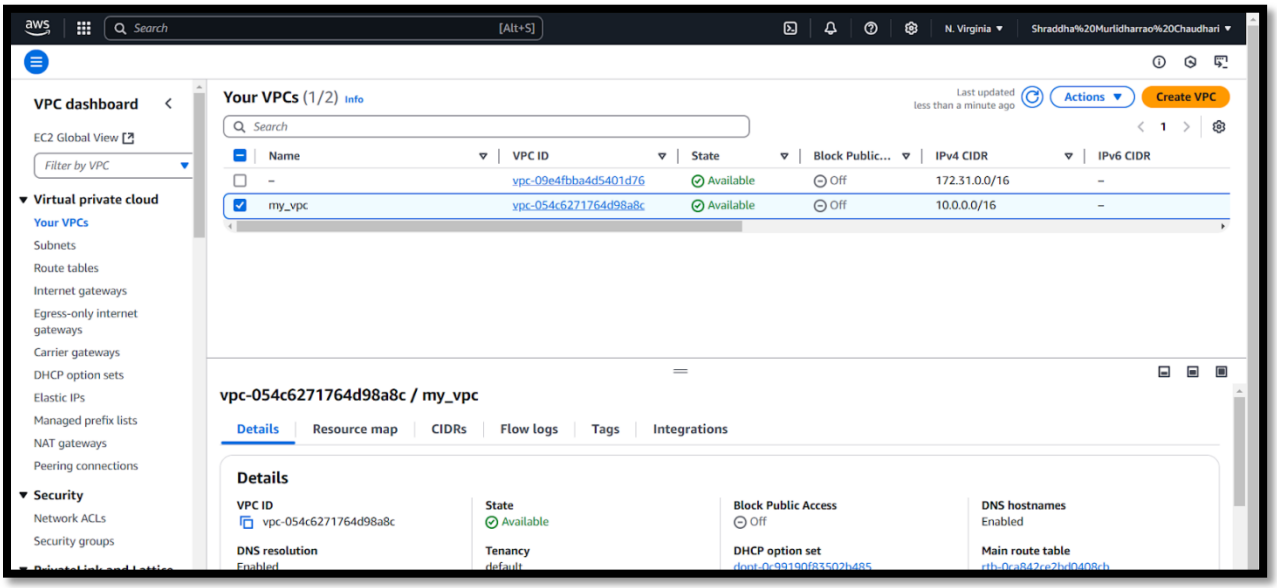
```
TERMINAL
Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.
PS C:\Users\HP\Desktop\Terraform_modules> terraform apply
module.vpc.data.aws_availability_zones.available: Reading...
module.vpc.aws_vpc.main: Refreshing state... [id=vpc-054c6271764d98a8c]
module.ec2.aws_security_group.ec2: Refreshing state... [id=sg-0070be2fdfd23209b]
module.vpc.data.aws_availability_zones.available: Read complete after 1s [id=us-east-1]
module.vpc.aws_internet_gateway.igw: Refreshing state... [id=igw-05e0e5837244fef10]
module.vpc.aws_subnet.public[0]: Refreshing state... [id=subnet-023855e14dc92b5a5]
module.vpc.aws_subnet.public[1]: Refreshing state... [id=subnet-0f47e89568dbd18e6]
module.vpc.aws_subnet.private[0]: Refreshing state... [id=subnet-083de97a751036b30]
module.alb.aws_security_group.lb: Refreshing state... [id=sg-027e8776a90dd7bfb]
module.vpc.aws_subnet.private[1]: Refreshing state... [id=subnet-00d159fdf0c156ee6]
module.ec2.aws_lb_target_group.example: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:147506552856:targetgroup/app-tg/59559d0b491f1b14]
module.vpc.aws_route_table.public: Refreshing state... [id=rtb-02b421bad40eb836e]
module.alb.aws_lb.main: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:147506552856:loadbalancer/app/app-lb/86a03d070e0c4d2f]
module.vpc.aws_route_table_association.public[1]: Refreshing state... [id=rtbassoc-008649759a11bab60]
module.vpc.aws_route_table_association.public[0]: Refreshing state... [id=rtbassoc-0b1bf0cf0976fec7a]
module.alb.aws_lb_listener.http: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:147506552856:listener/app/app-lb/86a03d070e0c4d2f/c6c1c074640fe248]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create
  ~ update in-place
  - destroy

Terraform will perform the following actions:

  # module.ec2.aws_instance.app[0] will be created
  + resource "aws_instance" "app" {
      + ami                                  = "ami-0e2c8caa4b6378d8c"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + enable_primary_ipv6                  = (known after apply)
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + iam_instance_profile                 = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_lifecycle                   = (known after apply)
      + instance_state                       = (known after apply)
```

Once done, then verify on the AWS account if all resources were created as per plan or not.

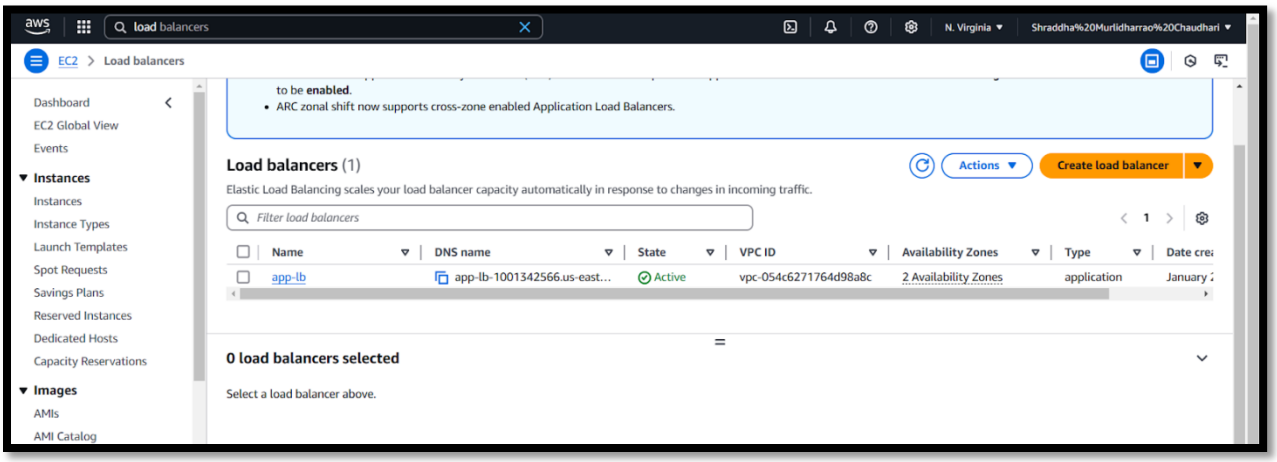
EC2 created:



VPC:



Load Balancer:



4.8: Cleanup

After done all process if we want to destroy the infrastructure we can do by using cmd:

```
terraform destroy
```

5. Document Overview:

This project implements a multi-tier architecture in AWS using Terraform modules. The architecture includes:

- A custom VPC with public and private subnets
- EC2 instances deployed in private subnets
- Security groups for EC2 instances and ALB
- An internet-facing Application Load Balancer (ALB

2. S3 lambda SNS notification system

Components:

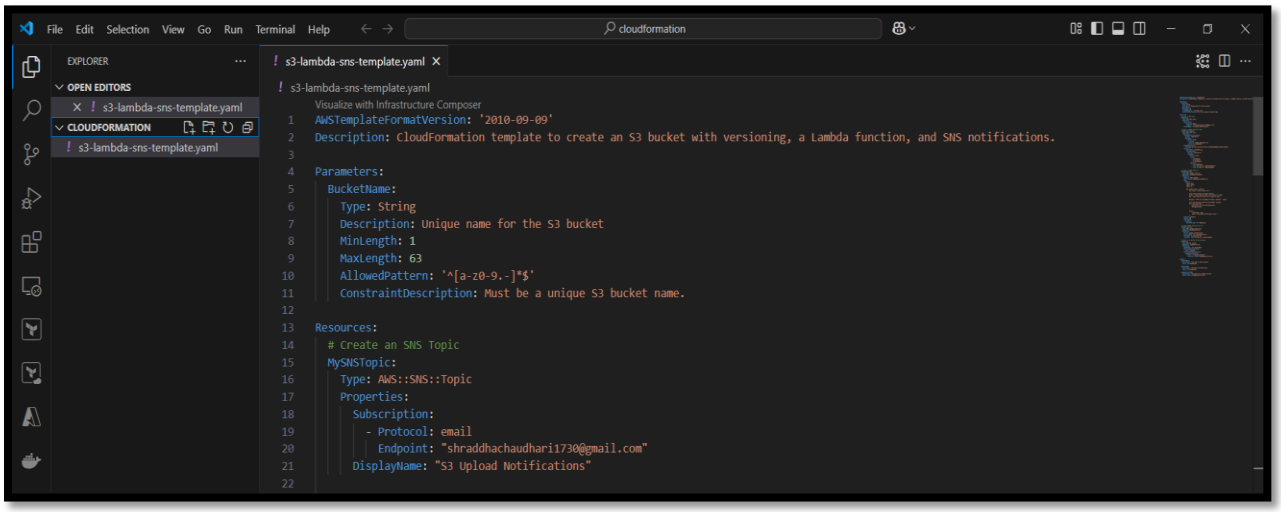
1. **S3 Bucket**: Storage for files with versioning enabled
2. **Lambda Function**: Processes S3 events and sends notifications
3. **SNS Topic**: Handles email notifications distribution
4. **IAM Roles**: Manages service permissions

Implementation Steps

1. Prerequisites

- AWS CLI installed and configured
- Access to AWS Console
- Necessary IAM permissions.

Already install AWS CLI for terraform so no need to install again.
We need to do aws configure on vscode for access aws accout that already done so no need to do it again.
After then, We need to create one folder on our local ex. “Cloudformation” and go to vscode and click on file and add folder and select created folder on local that is “Cloudformation”
Next, We need to create yaml file for create cloudformation template.
Ex- s3-lambda-sns-template.yaml



Next, have to create template to automate clouformation. In this template it will create cloudformation stack also create lambda function as well as create s3 bucket and bucket name should be globally unique. And lambda trigger s3 events get email subscription confirmation once it's done and if user uploads any file in s3 bucket get notification on mail through SNS.

This is Cloudformation template:

```yaml
AWSTemplateFormatVersion: '2010-09-09'
Description: CloudFormation template to create an S3 bucket with versioning, a Lambda function, and SNS notifications.

Parameters:
  BucketName:
    Type: String
    Description: Unique name for the S3 bucket
    MinLength: 1
    MaxLength: 63
    AllowedPattern: '^[a-z0-9.-]*$'
    ConstraintDescription: Must be a unique S3 bucket name.

Resources:
  # Create an SNS Topic
  MySNSTopic:
    Type: AWS::SNS::Topic
    Properties:
      Subscription:
        - Protocol: email
          Endpoint: "shraddhachaudhari1730@gmail.com"
      DisplayName: "S3 Upload Notifications"

  # Create a Lambda Execution Role
  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: S3ToSNSPolicy
          PolicyDocument:
            Version: '2012-10-17'
            Statement:
              - Effect: Allow
                Action:
                  - sns:Publish
                  - s3:GetObject
                  - s3:ListBucket
                Resource:
                  - !Ref MySNSTopic
                  - !Sub "arn:aws:s3:::${BucketName}/*"
                  - !Sub "arn:aws:s3:::${BucketName}"

  # Create a Lambda Function
  MyLambdaFunction:
    Type: AWS::Lambda::Function
    DependsOn: LambdaExecutionRole
    Properties:
      Handler: index.handler
      Role: !GetAtt LambdaExecutionRole.Arn
      Code:
        ZipFile: |
          import json
          import boto3
          import os

          def handler(event, context):
              sns_client = boto3.client('sns')

              # Get the S3 bucket and object details
              bucket = event['Records'][0]['s3']['bucket']['name']
              key = event['Records'][0]['s3']['object']['key']

              message = f"New file uploaded to bucket '{bucket}': {key}"
```

```
            # Use the SNS Topic ARN from environment variable
            sns_client.publish(
                TopicArn=os.environ['SNS_TOPIC_ARN'],
                Message=message
            )

            return {
                'statusCode': 200,
                'body': json.dumps('Notification sent!')
            }
      Runtime: python3.8
      Timeout: 30
      Environment:
        Variables:
          SNS_TOPIC_ARN: !Ref MySNSTopic

  # Create Lambda Permission for S3
  LambdaPermission:
    Type: AWS::Lambda::Permission
    DependsOn: MyLambdaFunction
    Properties:
      Action: lambda:InvokeFunction
      FunctionName: !Ref MyLambdaFunction
      Principal: s3.amazonaws.com
      SourceArn: !Sub "arn:aws:s3:::${BucketName}"

  # Create an S3 Bucket with Versioning
  MyS3Bucket:
    Type: AWS::S3::Bucket
    DependsOn: LambdaPermission
    Properties:
      BucketName: !Ref BucketName
      VersioningConfiguration:
        Status: Enabled
      NotificationConfiguration:
        LambdaConfigurations:
          - Event: 's3:ObjectCreated:*'
            Function: !GetAtt MyLambdaFunction.Arn

Outputs:
  S3BucketName:
    Description: "The name of the S3 bucket"
    Value: !Ref MyS3Bucket

  SNSTopicARN:
    Description: "The ARN of the SNS topic"
    Value: !Ref MySNSTopic
```

```
  LambdaFunctionARN:
    Description: "The ARN of the Lambda function"
    Value: !GetAtt MyLambdaFunction.Arn
```

5.1. Deploy stack

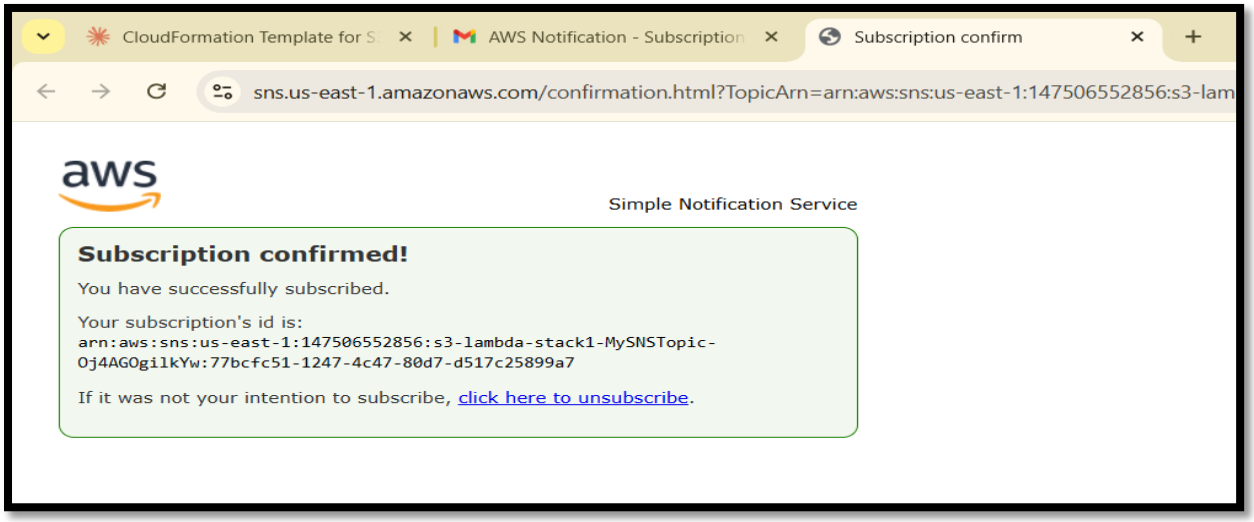
Run the following AWS CLI command:

```
aws cloudformation create-stack \
  --stack-name s3-lambda-stack1 \
  --template-body file://s3-lambda-sns.yaml \
  --parameters ParameterKey=BucketName,ParameterValue=s3-object-yyyl \
  --capabilities CAPABILITY_IAM
```

5.2. Post-Deployment Steps

SNS Subscription Confirmation

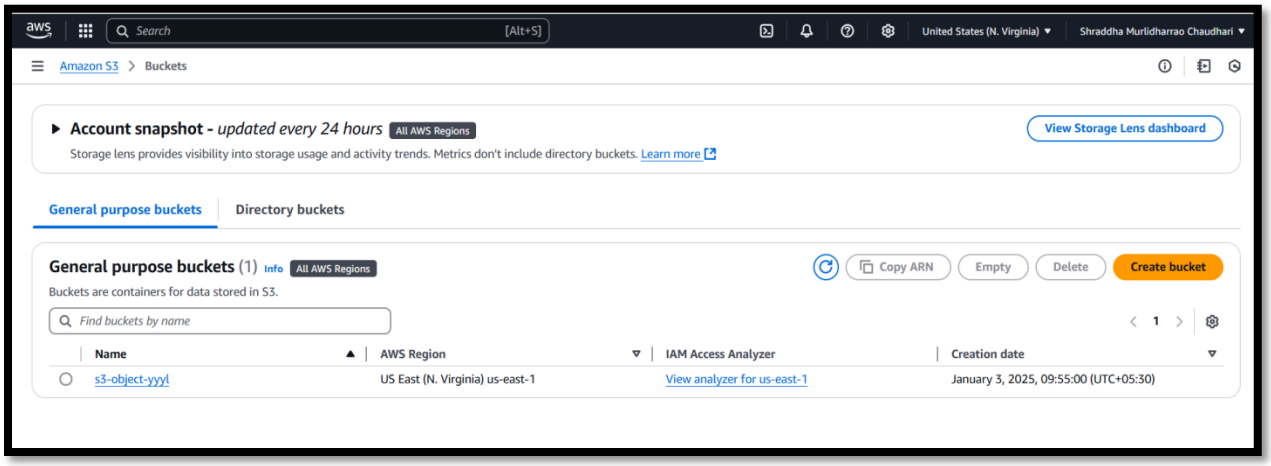
1. Check your email for AWS Notification subscription confirmation
2. Click the "Confirm subscription" link
3. Verify successful subscription on the confirmation page

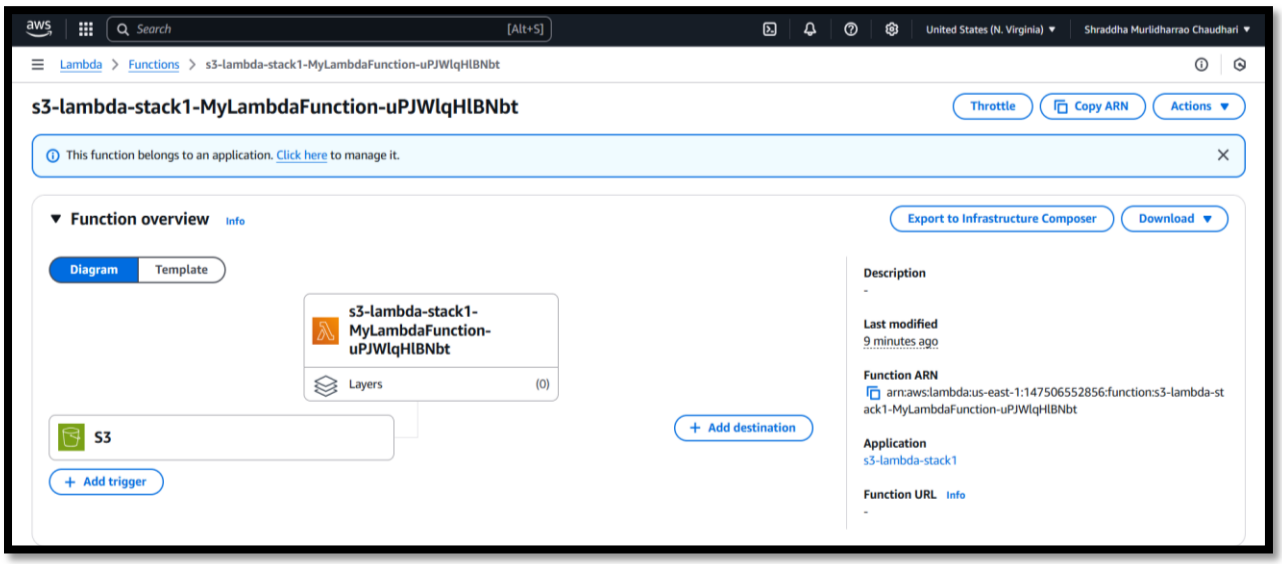


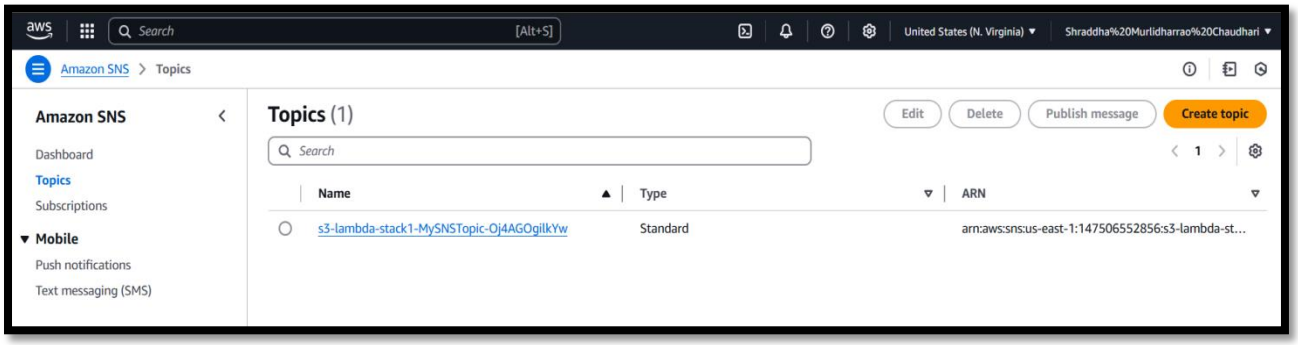
Resource Verification

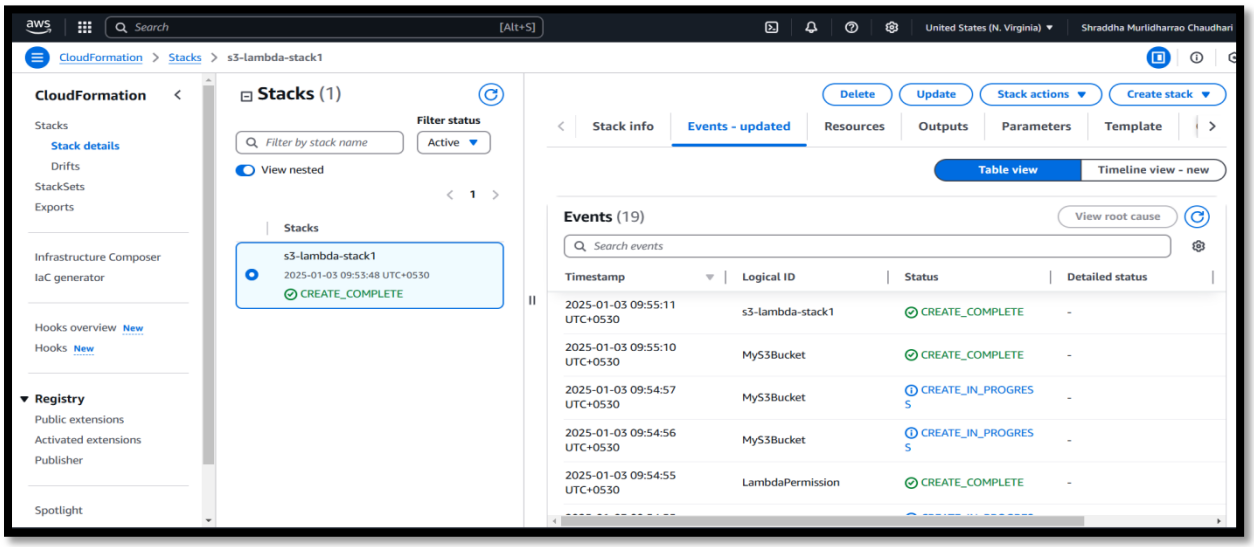
Check the following resources in AWS Console:

- S3 bucket creation
- Lambda function deployment
- SNS topic setup
- Cloudformation stack



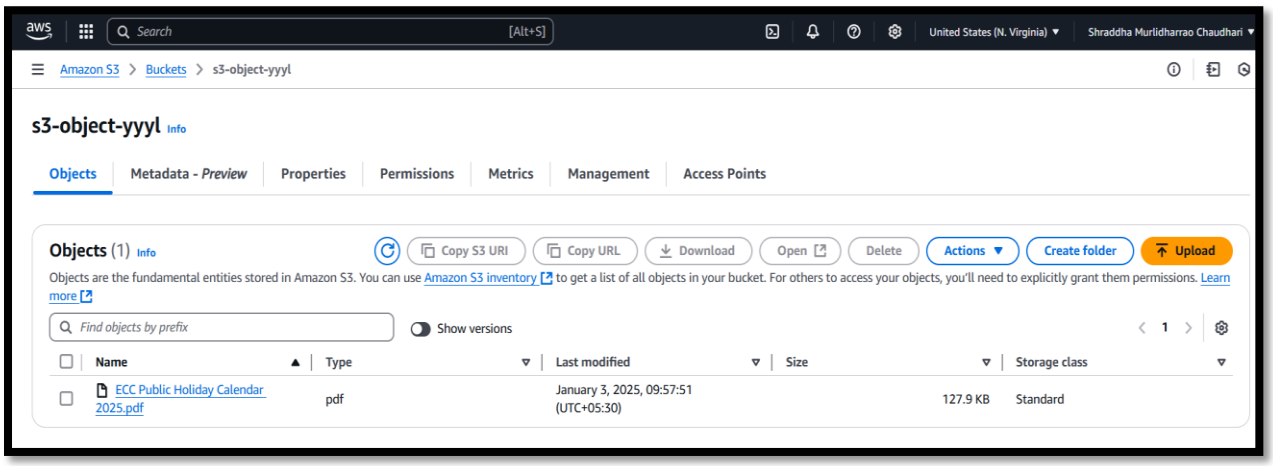






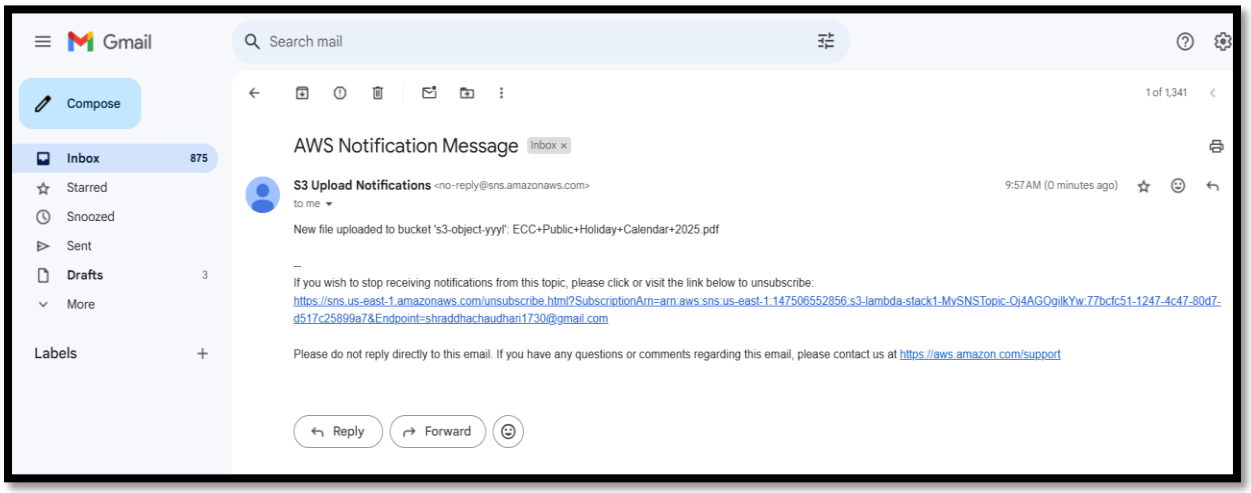
5.3: Testing the setup

Upload a file in s3 bucket



Verify Workflow

1. File upload triggers Lambda function
2. Lambda processes S3 event
3. SNS notification sent
4. Email received by subscriber



Once done if we want to Cleanup Resource then,

5.4: Delete Stack

```
aws cloudformation delete-stack --stack-name s3-lambda-stack1
```

This will remove:

- S3 bucket
- Lambda function
- SNS topic
- IAM roles

Overview

This documentation covers the implementation of an automated notification system using AWS S3, Lambda, and SNS services. When a file is uploaded to an S3 bucket, it triggers a Lambda function that sends a notification via SNS to subscribed email addresses.