**Mayur Jaiswal**

**D15B**

**26**

**Adv Devops Assignment 2**

**Code:**

provider "aws" {

region = "ap-south-1"

}

# S3 Bucket

resource "aws\_s3\_bucket" "s3mayur" {

bucket = "my-terraform-s3-bucket"

acl = "private"

versioning {

enabled = true

}

}

# SQS Queue

resource "aws\_sqs\_queue" "sqsmayur" {

name = "my-terraform-sqs-queue"

}

# Lambda Function

resource "aws\_lambda\_function" "lambda\_mayur" {

function\_name = "s3-to-sqs-lambda"

role = aws\_iam\_role.lambda\_exec.arn

handler = "index.handler"

runtime = "nodejs14.x"

timeout = 10

filename = "lambda.zip" # Path to the Lambda zip file

environment {

variables = {

QUEUE\_URL = aws\_sqs\_queue.sqsmayur.id

}

}

}

# IAM Role for Lambda execution

resource "aws\_iam\_role" "lambda\_exec" {

name = "lambda\_exec\_role"

assume\_role\_policy = jsonencode({

Version = "2012-10-17",

Statement = [{

Action = "sts:AssumeRole",

Effect = "Allow",

Principal = {

Service = "lambda.amazonaws.com"

}

}]

})

}

# IAM Role Policy for Lambda (grant permissions to interact with S3 and SQS)

resource "aws\_iam\_role\_policy" "lambda\_exec\_policy" {

role = aws\_iam\_role.lambda\_exec.id

policy = jsonencode({

Version = "2012-10-17",

Statement = [

{

Action = [

"sqs:SendMessage"

],

Effect = "Allow",

Resource = aws\_sqs\_queue.sqsmayur.arn

},

{

Action = [

"s3:GetObject"

],

Effect = "Allow",

Resource = "${aws\_s3\_bucket.s3mayur.arn}/\*"

}

]

})

}

# S3 Bucket Notification to trigger Lambda on object creation

resource "aws\_s3\_bucket\_notification" "s3\_notification" {

bucket = aws\_s3\_bucket.s3mayur.id

lambda\_function {

lambda\_function\_arn = aws\_lambda\_function.lambda\_mayur.arn

events = ["s3:ObjectCreated:\*"]

}

}

# Lambda Permission for S3 to invoke the Lambda function

resource "aws\_lambda\_permission" "allow\_s3" {

statement\_id = "AllowS3InvokeLambda"

action = "lambda:InvokeFunction"

function\_name = aws\_lambda\_function.lambda\_mayur.function\_name

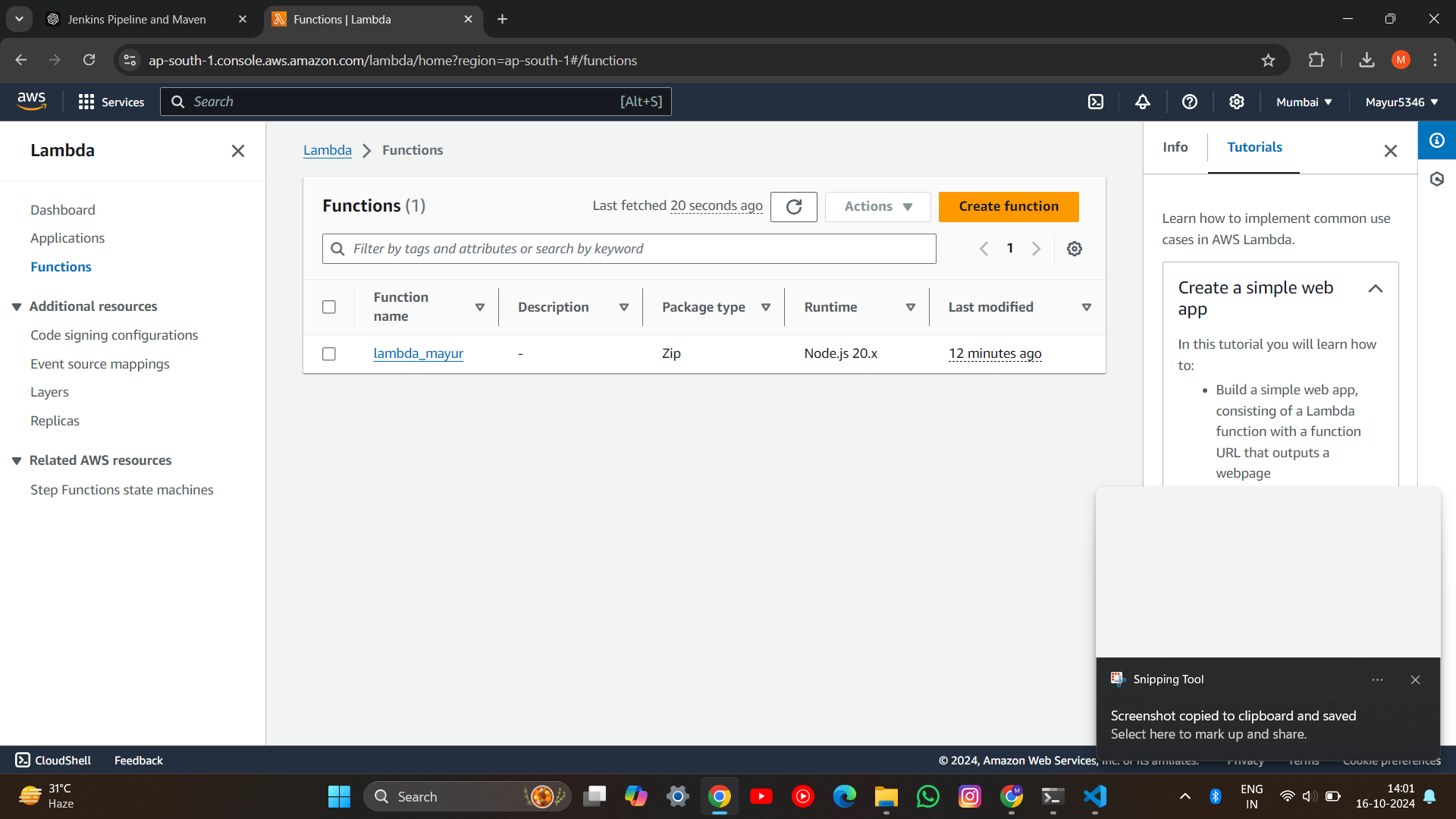
principal = "s3.amazonaws.com"

source\_arn = aws\_s3\_bucket.s3mayur.arn

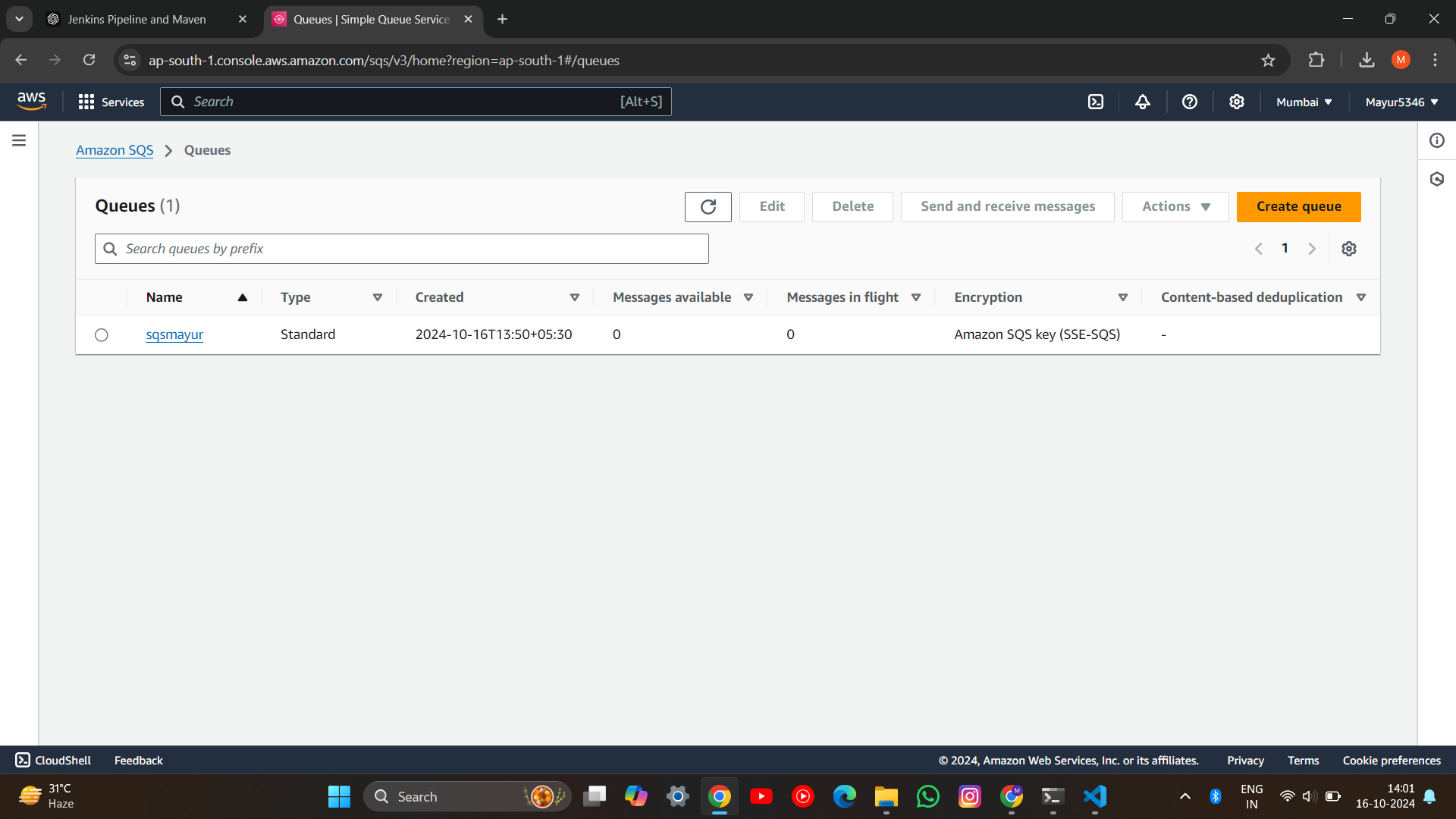
}

**Implementation:**

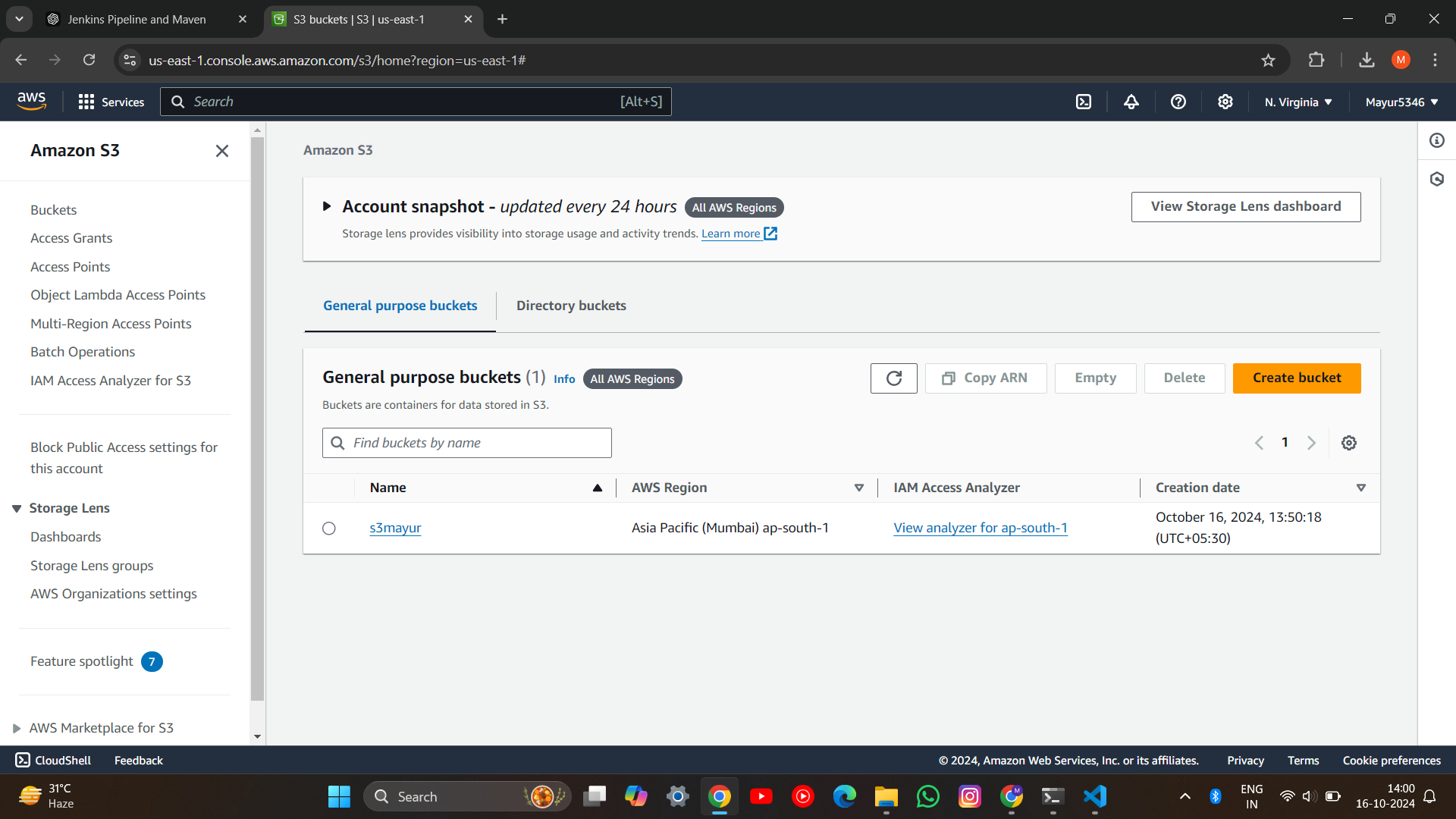
1. Creating Lambda Function



1. Creating Sqs Queue

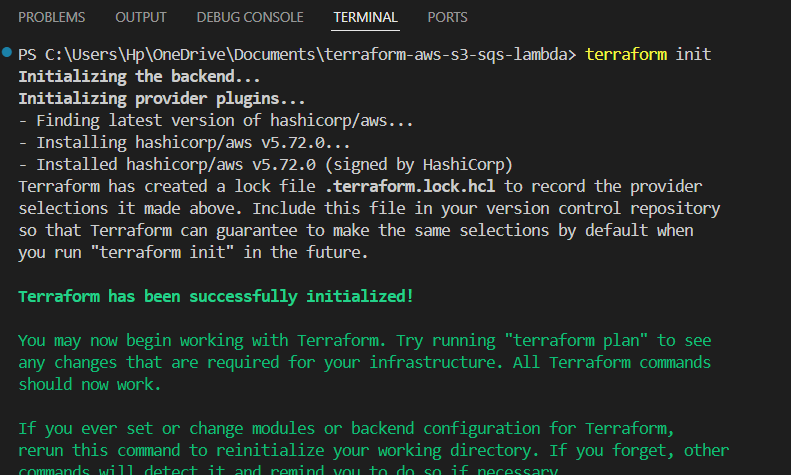


1. Creating S3 Bucket



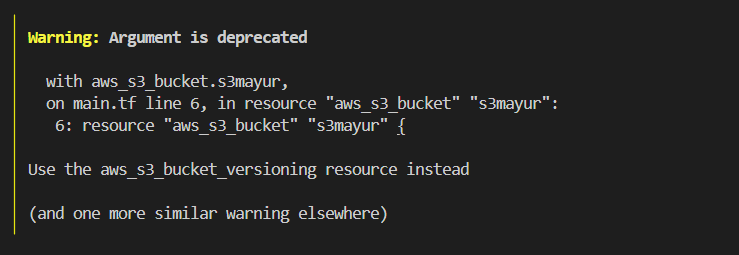
Performing Terraform commands

1. Terraform init

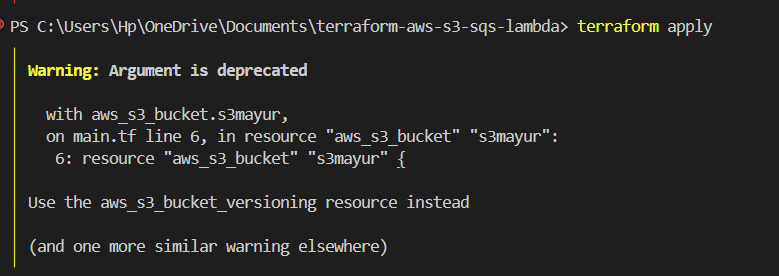


1. Terraform plan



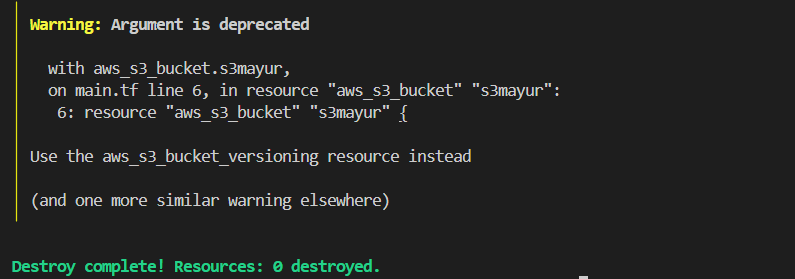


1. Terraform apply

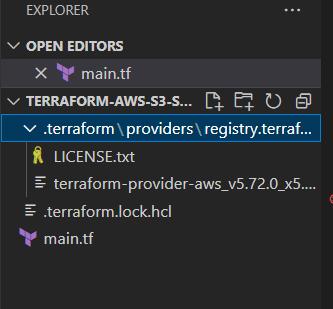


1. Terraform destroy





Folder structure of main.tf file



Conclusion:

In this experiment, we successfully deployed an AWS infrastructure using Terraform, integrating essential services such as Amazon S3, SQS, and Lambda. By leveraging Terraform's infrastructure as code capabilities, we were able to automate the provisioning and configuration of cloud resources, ensuring consistency and reproducibility in our deployments.