Name:Sujal.S.Tekwani
Class:D15B
Roll No:63

Adv DevOps Assignment-2

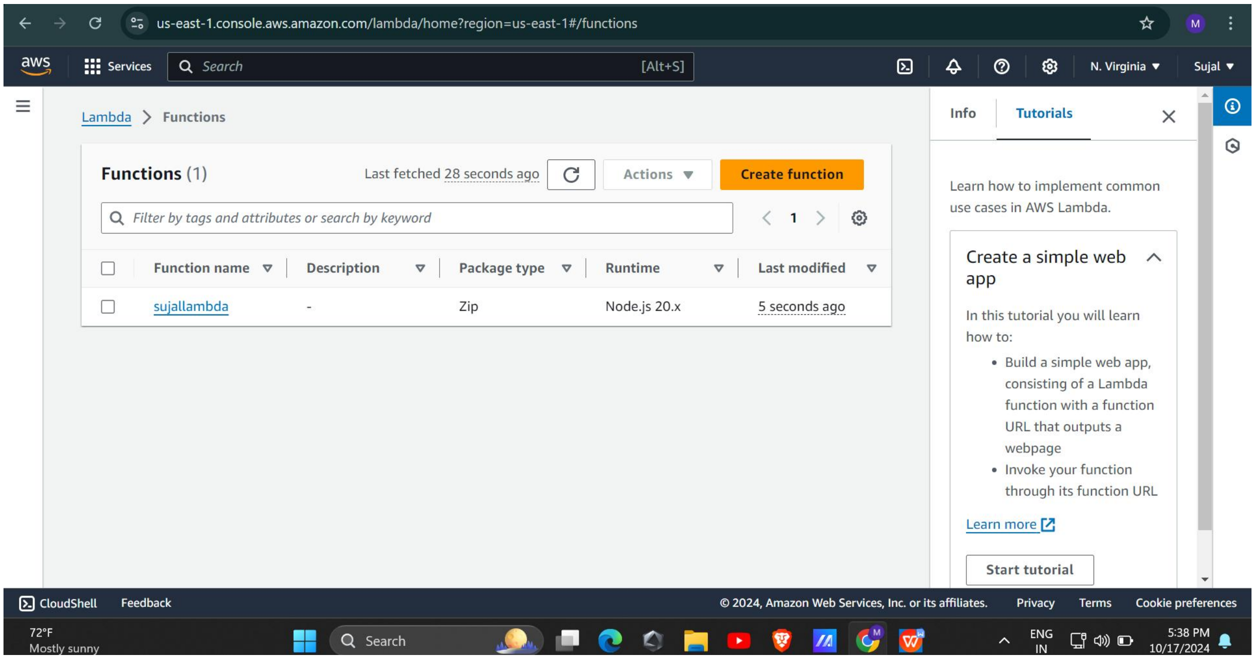
Code:

```
provider "aws" {
  region = "us-east-1"
}

# S3 Bucket
resource "aws_s3_bucket" "s3sujal" {
  bucket = "my-terraform-s3-bucket"
  acl    = "private"

  versioning {
    enabled = true
  }
}

# SQS Queue
resource "aws_sqs_queue" "sqssujal" {
  name = "my-terraform-sqs-queue"
}

# Lambda Function
resource "aws_lambda_function" "lambda_sujal" {
  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.sqssujal.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.sqssujal.arn
      },
      {
        Action = [
          "s3:GetObject"
        ],
        Effect   = "Allow",
        Resource = "${aws_s3_bucket.s3sujal.arn}/*"
      }
    ]
  })
}

# S3 Bucket Notification to trigger Lambda on object creation
resource "aws_s3_bucket_notification" "s3_notification" {
  bucket = aws_s3_bucket.s3sujal.id
  lambda_function {
    lambda_function_arn = aws_lambda_function.lambda_sujal.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_sujal.function_name
  principal     = "s3.amazonaws.com"

  source_arn = aws_s3_bucket.s3sujal.arn
}
```

Implementation:

1. Creating Lambda Function



us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/functions

Lambda > Functions

Functions (1) — Last fetched 28 seconds ago — Actions ▼ — Create function

Filter by tags and attributes or search by keyword

	Function name	Description	Package type	Runtime	Last modified
□	sujallambda	-	Zip	Node.js 20.x	5 seconds ago

Info | Tutorials

Learn how to implement common use cases in AWS Lambda.

Create a simple web app

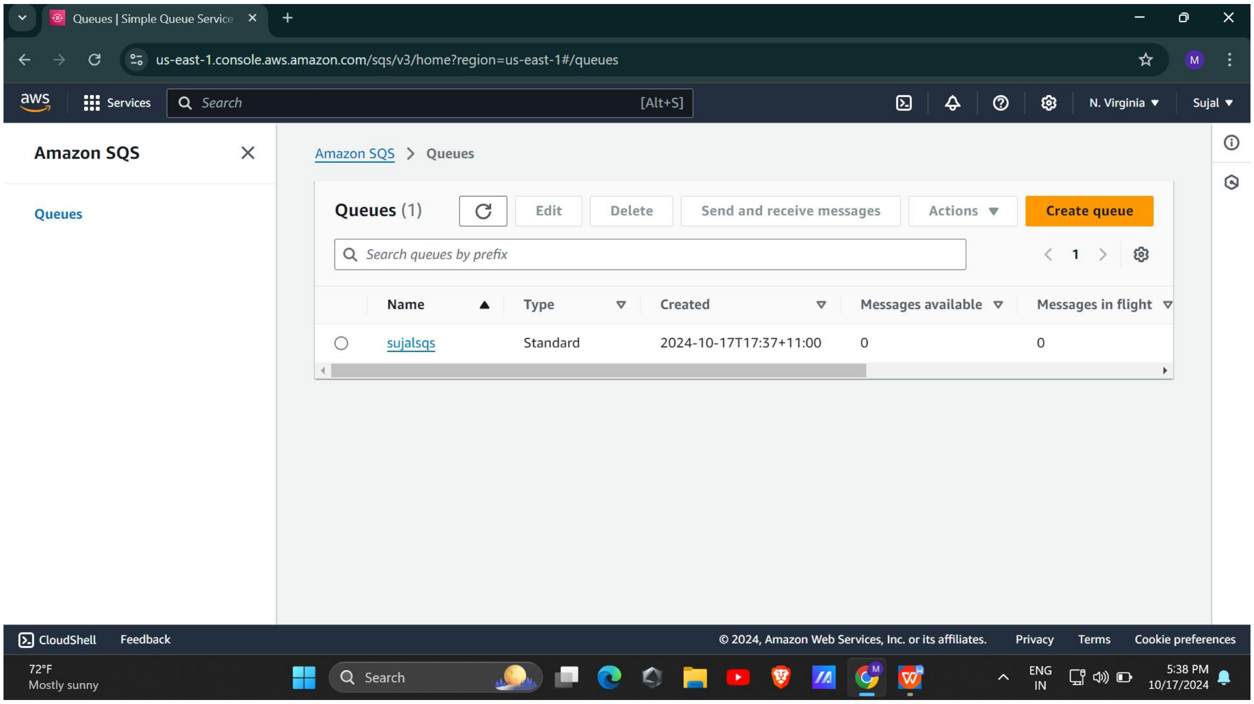
In this tutorial you will learn how to:

- Build a simple web app, consisting of a Lambda function with a function URL that outputs a webpage
- Invoke your function through its function URL

Learn more

Start tutorial

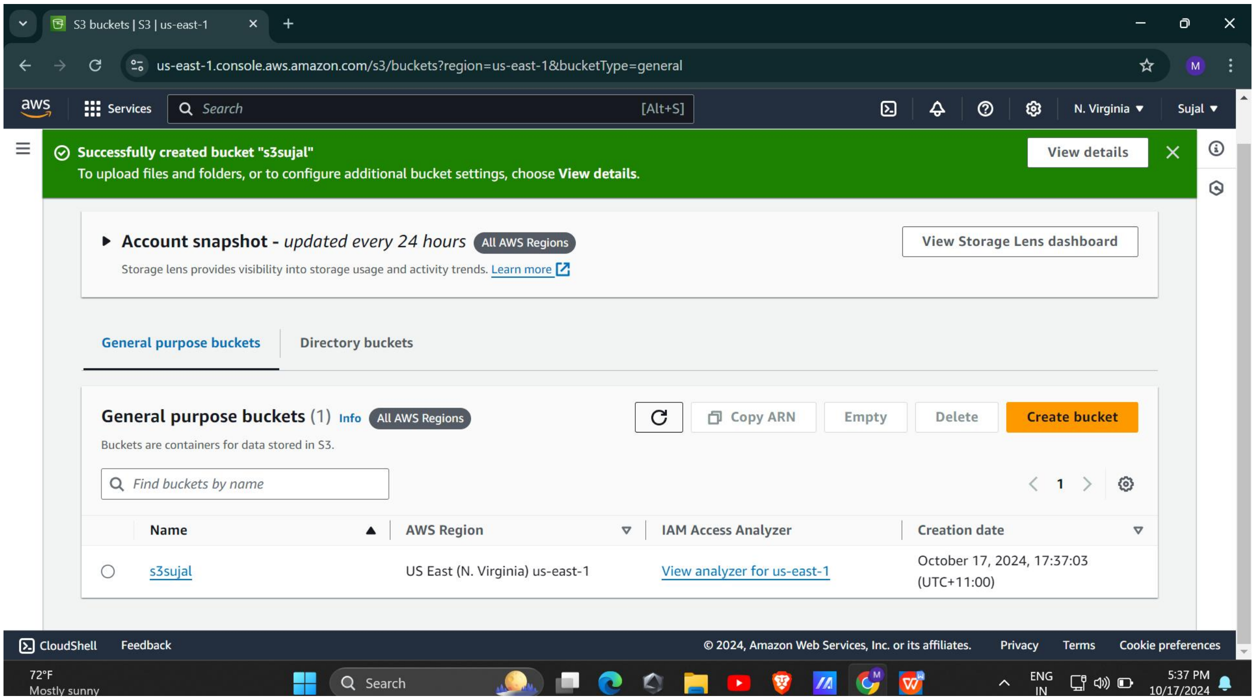
2. Creating Sqs Queue



Queues (1)

	Name	Type	Created	Messages available	Messages in flight
○	sujalsqs	Standard	2024-10-17T17:37+11:00	0	0

3. Creating S3 Bucket



Successfully created bucket "s3sujal"
To upload files and folders, or to configure additional bucket settings, choose **View details**.

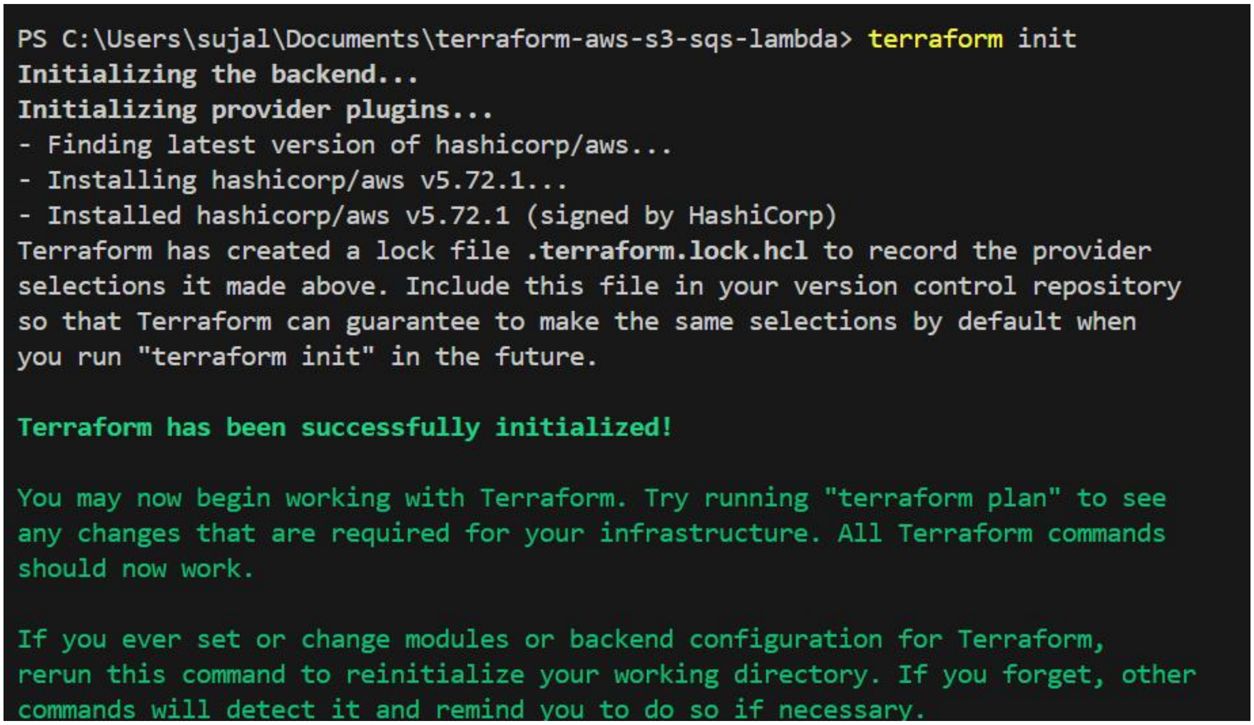
Account snapshot - *updated every 24 hours* All AWS Regions
Storage lens provides visibility into storage usage and activity trends. Learn more

General purpose buckets (1) Info All AWS Regions
Buckets are containers for data stored in S3.

	Name	AWS Region	IAM Access Analyzer	Creation date
○	s3sujal	US East (N. Virginia) us-east-1	View analyzer for us-east-1	October 17, 2024, 17:37:03 (UTC+11:00)

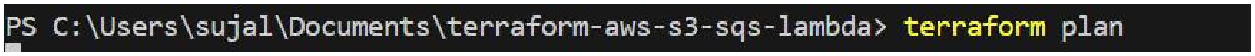
Performing Terraform commands

1. Terraform init

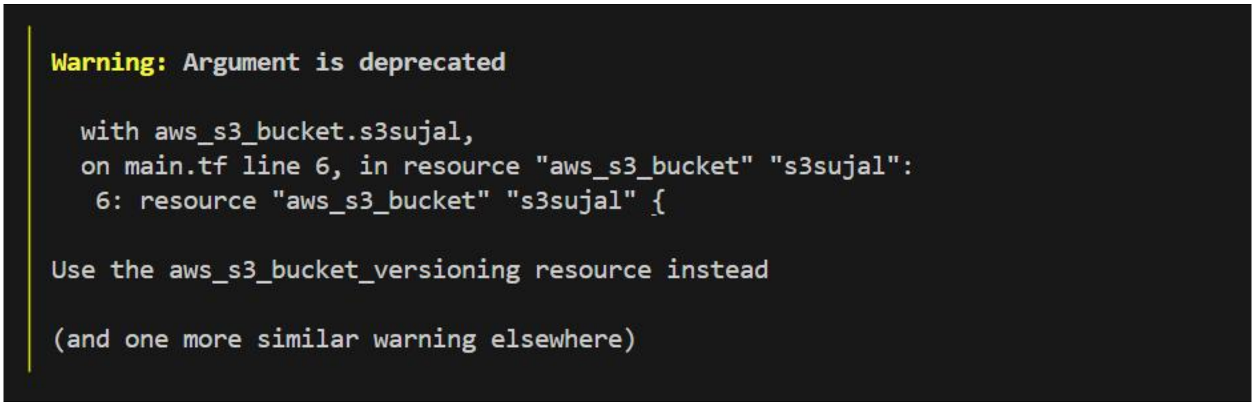


```
PS C:\Users\sujal\Documents\terraform-aws-s3-sqs-lambda> terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.72.1...
- Installed hashicorp/aws v5.72.1 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

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.
```

2. Terraform plan

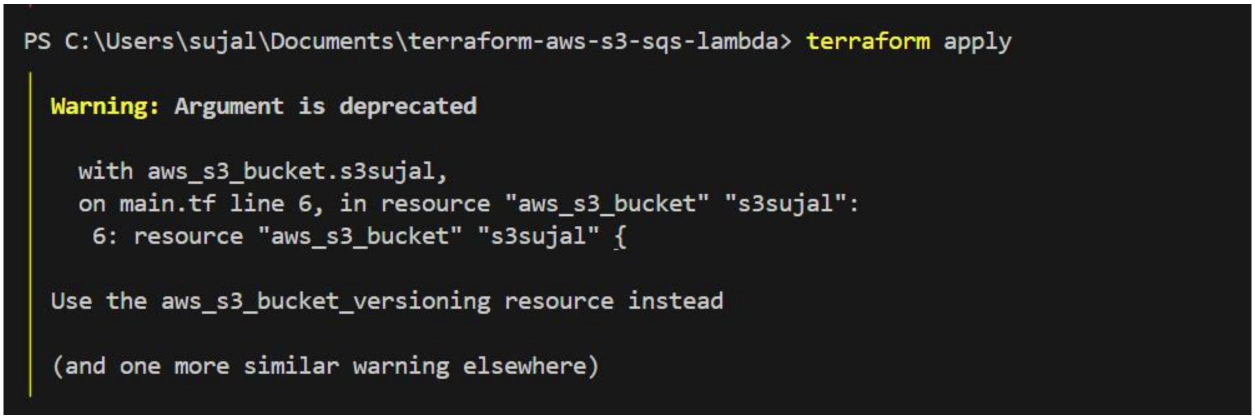


```
PS C:\Users\sujal\Documents\terraform-aws-s3-sqs-lambda> terraform plan
```



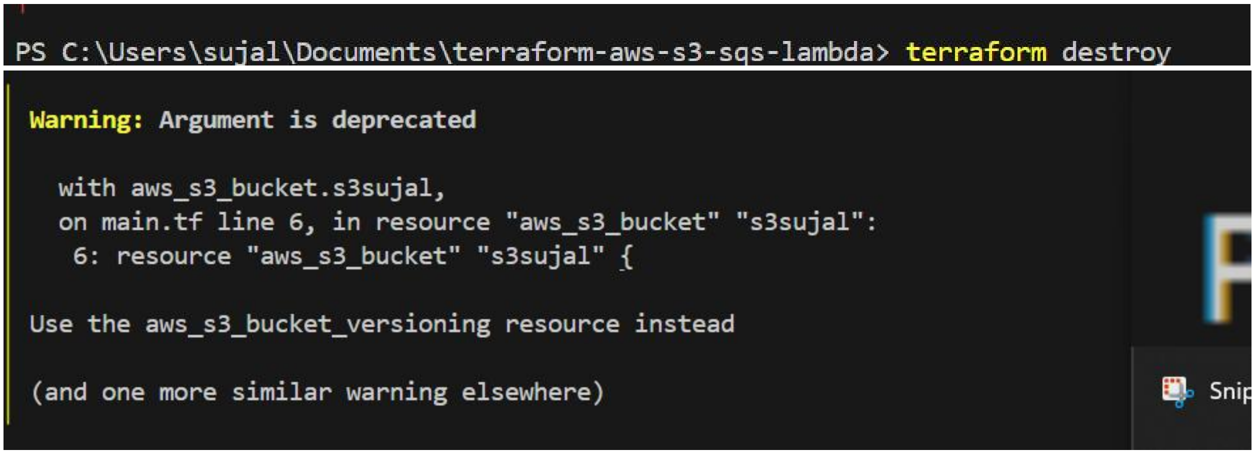
```
Warning: Argument is deprecated

  with aws_s3_bucket.s3sujal,
  on main.tf line 6, in resource "aws_s3_bucket" "s3sujal":
   6: resource "aws_s3_bucket" "s3sujal" {

Use the aws_s3_bucket_versioning resource instead

(and one more similar warning elsewhere)
```

3.Terraform apply



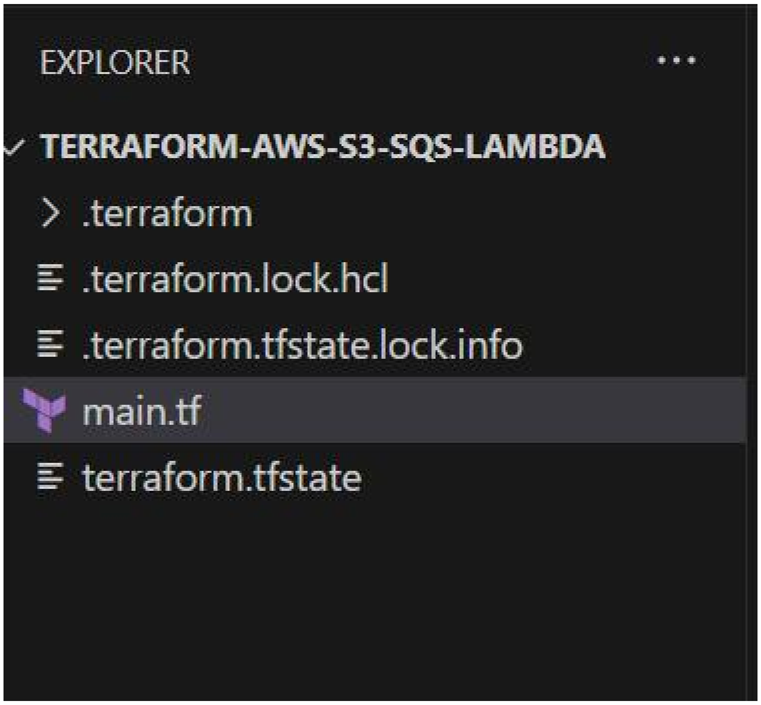
```
PS C:\Users\sujal\Documents\terraform-aws-s3-sqs-lambda> terraform apply

 Warning: Argument is deprecated

   with aws_s3_bucket.s3sujal,
   on main.tf line 6, in resource "aws_s3_bucket" "s3sujal":
    6: resource "aws_s3_bucket" "s3sujal" {

 Use the aws_s3_bucket_versioning resource instead

 (and one more similar warning elsewhere)
```

4.Terraform destroy



```
PS C:\Users\sujal\Documents\terraform-aws-s3-sqs-lambda> terraform destroy

 Warning: Argument is deprecated

   with aws_s3_bucket.s3sujal,
   on main.tf line 6, in resource "aws_s3_bucket" "s3sujal":
    6: resource "aws_s3_bucket" "s3sujal" {

 Use the aws_s3_bucket_versioning resource instead

 (and one more similar warning elsewhere)
```

Folder structure of main.tf file



```
TERRAFORM-AWS-S3-SQS-LAMBDA
  .terraform
  .terraform.lock.hcl
  .terraform.tfstate.lock.info
  main.tf
  terraform.tfstate
```

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.