Assignment No -02

Code:

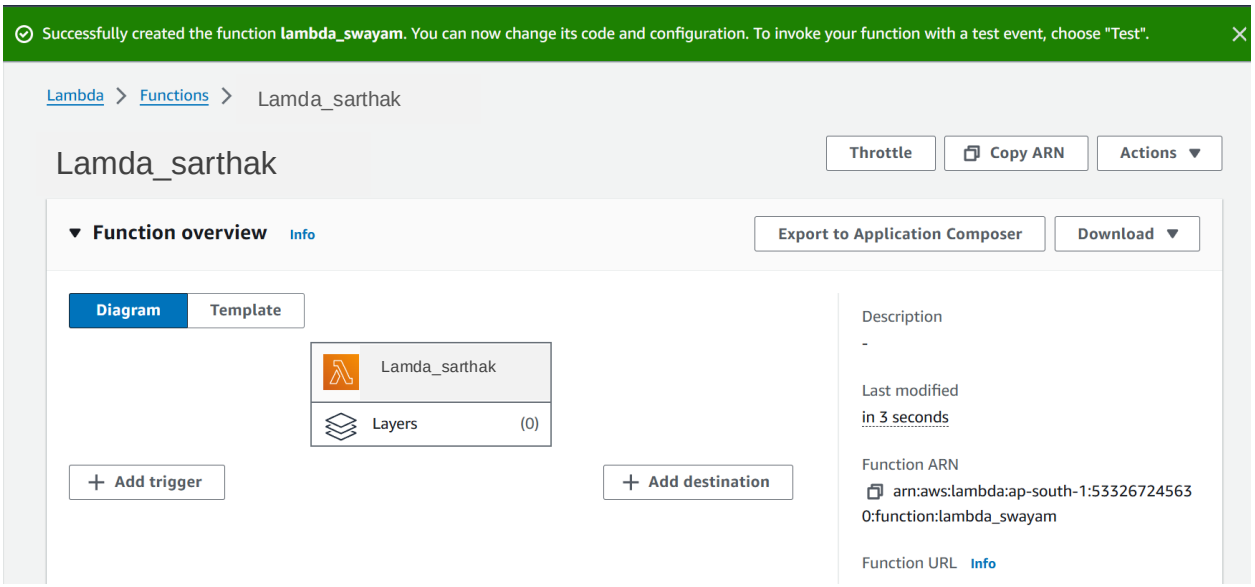
```
provider "aws" {
  region = "ap-south-1"
}

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

  versioning {
    enabled = true
  }
}

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

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

# S3 Bucket Notification to trigger Lambda on object creation
resource "aws_s3_bucket_notification" "s3_notification" {
  bucket = aws_s3_bucket. Sarthak.id

  lambda_function {
    lambda_function_arn = aws_lambda_function.lambda_Sarthak.arn
    events              = ["s3:ObjectCreated:*"]
  }
}

# Lambda Permission for S3 to invoke the Lambda function
```

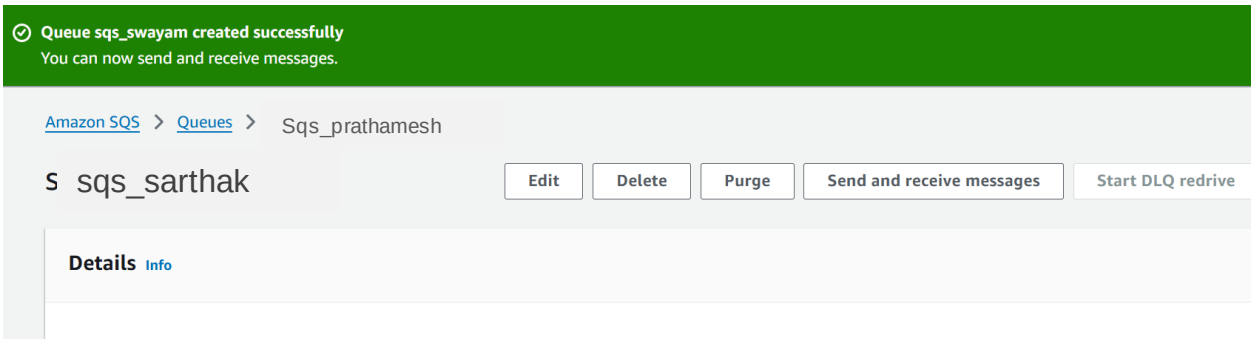
```
resource "aws_lambda_permission" "allow_s3" {
  statement_id  = "AllowS3InvokeLambda"
  action        = "lambda:InvokeFunction"
  function_name = aws_lambda_function.lambda_sSarthak.function_name
  principal     = "s3.amazonaws.com"

  source_arn = aws_s3_bucket. Sarthak.arn
}
```

Implementation:

1. Creating Lambda Function



2. Creating Sqs Queue



Performing Terraform commands

1. Terraform init

```
sarthak@DESKTOP-USER:~/terraform$ terraform init
Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Using hashicorp/aws v5.31.0...

Terraform has been successfully initialized!
```

2. Terraform plan

```
sarthak@DESKTOP-USER:~/terraform$ terraform plan

Terraform used the selected providers to generate the following execution
plan. Resource actions are indicated with the following symbols:
  + create

Plan: 7 to add, 0 to change, 0 to destroy.
```

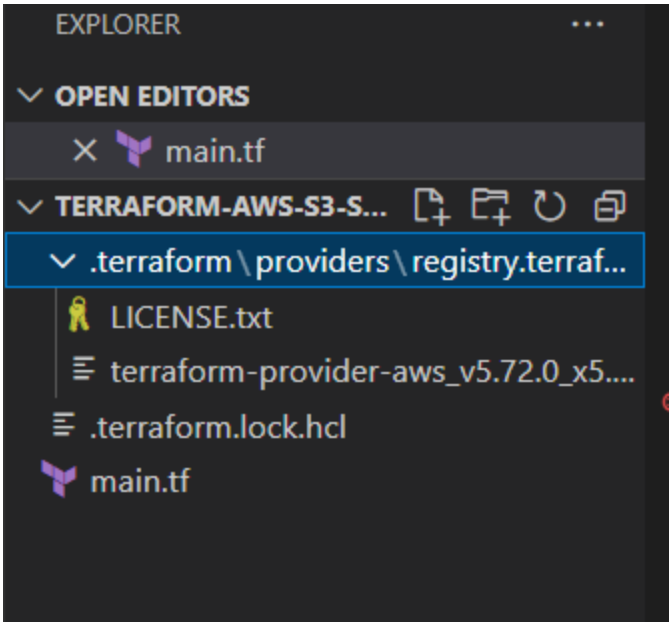
3. Terraform apply

```
sarthak@DESKTOP-USER:~/terraform$ terraform apply

Terraform used the selected providers to generate the following execution
plan. Resource actions are indicated with the following symbols:
  + create

Plan: 7 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

Apply complete! Resources: 7 added, 0 changed, 0 destroyed.
```

4. Terraform destroy

```
sarthak@DESKTOP-USER:~/terraform$ terraform destroy

Terraform used the selected providers to generate the following execution
plan. Resource actions are indicated with the following symbols:
  - destroy

Plan: 0 to add, 0 to change, 7 to destroy.

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

Destroy complete! Resources: 7 destroyed.
```

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.