Name-Tarunkumar Sharma
Div-D15B
Roll no:56

Adv Devops Assignment 2

Code:

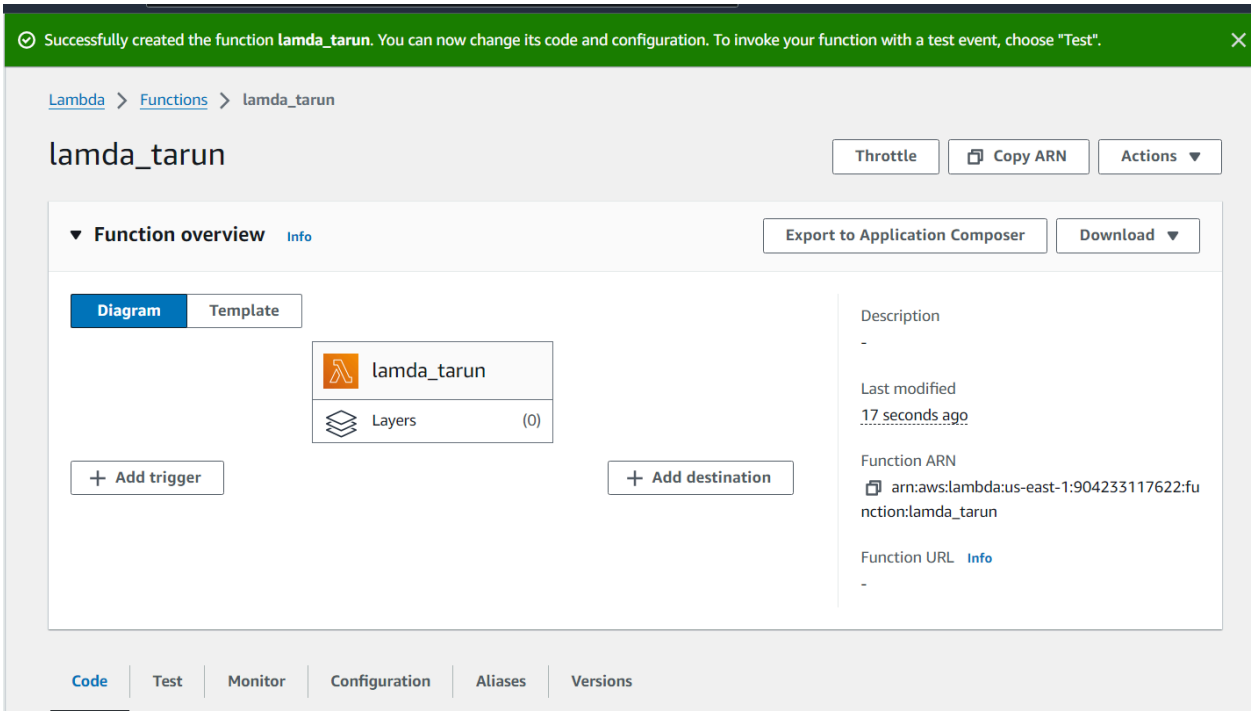
```
provider "aws" {
  region = "ap-south-1"
}

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

  versioning {
    enabled = true
  }
}

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

# Lambda Function
resource "aws_lambda_function" "lambda_swayam" {
  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.sqsswayam.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",
```

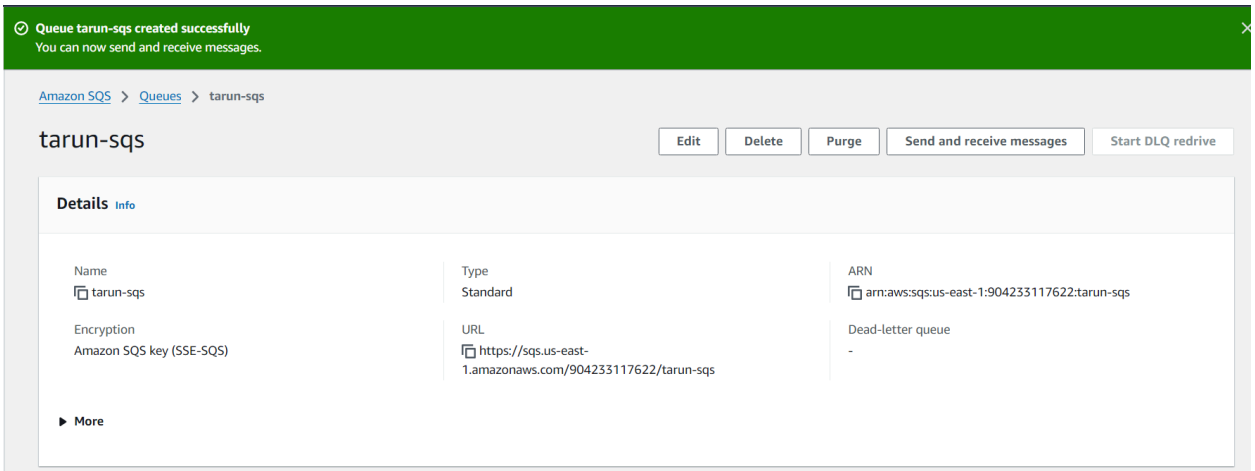
```
      Resource = aws_sqs_queue.sqsswayam.arn
    },
    {
      Action = [
        "s3:GetObject"
      ],
      Effect   = "Allow",
      Resource = "${aws_s3_bucket.swayamnewbucket.arn}/*"
    }
  ]
 })
}

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

 lambda_function {
  lambda_function_arn = aws_lambda_function.lambda_swayam.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_swayam.function_name
 principal     = "s3.amazonaws.com"

 source_arn = aws_s3_bucket.swayamnewbucket.arn
}
```

Implementation:

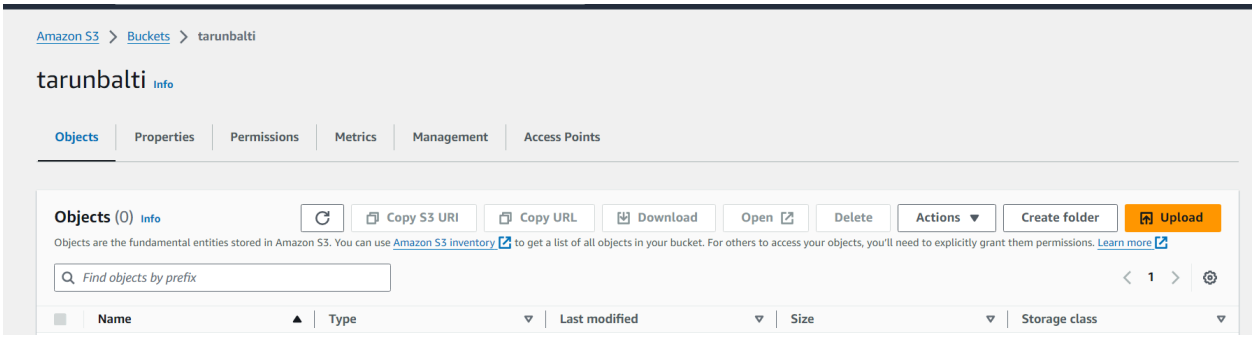
1. Creating Lambda Function



2. Creating Sqs Queue



3. Creating S3 Bucket



Performing Terraform commands

1. Terraform init

```
tarun@DESKTOP-TARUN:~/terraform$ terraform init
2024-03-21T10:15:32.123+0530 [INFO] Terraform version: 1.5.7
2024-03-21T10:15:32.123+0530 [INFO] Go runtime version: go1.20.7
2024-03-21T10:15:32.123+0530 [INFO] CLI args: []string{"terraform", "init"}

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.31.0...
- Installed hashicorp/aws v5.31.0 (signed by HashiCorp)

Terraform has been successfully initialized!
```

2. Terraform plan

```
tarun@DESKTOP-TARUN:~/terraform$ terraform plan

Terraform used the selected providers to generate the following execution plan. Resource a
  + create

Terraform will perform the following actions:

  # aws_s3_bucket.tarunbalti will be created
  + resource "aws_s3_bucket" "tarunbalti" {
      + bucket = "tarunbalti"
      + arn    = (known after apply)
      + id     = (known after apply)
      # ... other attributes ...
    }

  # aws_sqs_queue.sqs_tarun will be created
  + resource "aws_sqs_queue" "sqs_tarun" {
      + name = "sqs_tarun"
      + url  = (known after apply)
      # ... other attributes ...
    }
```

3. **Terraform apply**

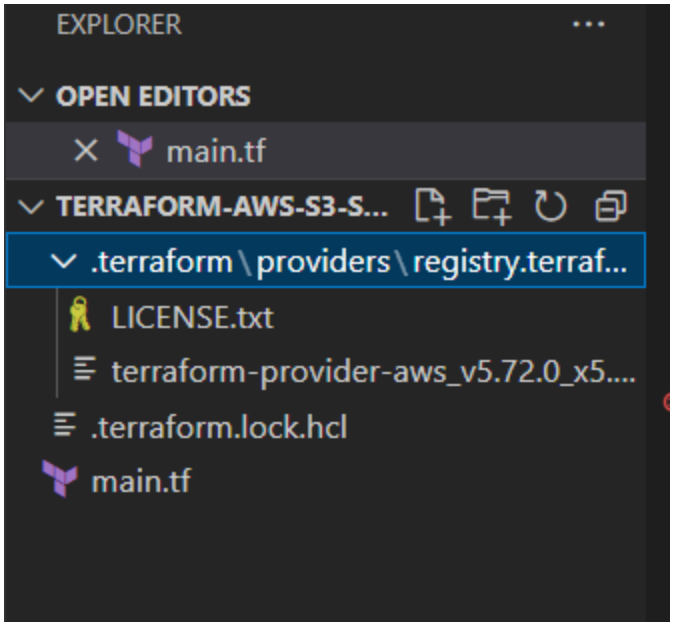
```
tarun@DESKTOP-TARUN:~/terraform$ terraform apply

Terraform used the selected providers to generate the following execution plan. Resource a
  + create

Terraform will perform the following actions:

  # aws_s3_bucket.tarunbalti will be created
  # aws_sqs_queue.sqs_tarun will be created
  # aws_lambda_function.lambda_tarun will be created
  # ... (other resources) ...

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

4. **Terraform destroy**

```
tarun@DESKTOP-TARUN:~/terraform$ terraform destroy
aws_s3_bucket.tarunbalti: Refreshing state... [id=tarunbalti]
aws_sqs_queue.sqs_tarun: Refreshing state... [id=https://sqs.us-west-2.amazonaws.com/12345
aws_lambda_function.lambda_tarun: Refreshing state... [id=lambda_tarun]
# ... (other resource refreshes) ...

Terraform used the selected providers to generate the following execution plan. Resource a
  - destroy

Terraform will perform the following actions:

  # aws_s3_bucket.tarunbalti will be destroyed
  # aws_sqs_queue.sqs_tarun will be destroyed
  # aws_lambda_function.lambda_tarun will be destroyed
  # ... (other resources) ...

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

aws_lambda_function.lambda_tarun: Destroying... [id=lambda_tarun]
aws_sqs_queue.sqs_tarun: Destroying... [id=https://sqs.us-west-2.amazonaws.com/123456789012
aws_s3_bucket.tarunbalti: Destroying... [id=tarunbalti]
aws_lambda_function.lambda_tarun: Destruction complete after 2s
aws_sqs_queue.sqs_tarun: Destruction complete after 1s
aws_s3_bucket.tarunbalti: Destruction complete after 3s
# ... (other resource destructions) ...

Destroy complete! Resources: 7 destroyed.

tarun@DESKTOP-TARUN:~/terraform$
```

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.