

Assignment No -02

Aim: Deploying AWS Infrastructure Using Terraform: A Hands-On Approach with S3, SQS, and Lambda Integration

Code:

```
provider "aws" {  
  region = "ap-south-1"  
}  
  
# S3 Bucket  
resource "aws_s3_bucket" "prathameshnewbucket" {  
  bucket = "my-terraform-s3-bucket"  
  acl    = "private"  
  
  versioning {  
    enabled = true  
  }  
}  
  
# SQS Queue  
resource "aws_sqs_queue" "sqs-prathamesh" {  
  name = "my-terraform-sqs-queue"  
}  
  
# Lambda Function  
resource "aws_lambda_function" "lambda_prathamesh" {  
  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.sqsprathamesh.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.sqsprathamesh.arn
      },
      {
        Action = [
          "s3:GetObject"
        ],
        Effect   = "Allow",
        Resource = "${aws_s3_bucket.prathameshnewbucket.arn}/*"
      }
    ]
  })
}

```

S3 Bucket Notification to trigger Lambda on object creation

```

resource "aws_s3_bucket_notification" "s3_notification" {
  bucket = aws_s3_bucket.prathameshnewbucket.id

  lambda_function {
    lambda_function_arn = aws_lambda_function.lambda_prathamesh.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_prathamesh.function_name
  principal     = "s3.amazonaws.com"

  source_arn = aws_s3_bucket.prathameshnewbucket.arn
}

```

Implementation:

1. Creating Lambda Function

The screenshot shows the AWS Lambda console interface for a function named 'Lamda_prathamesh'. The breadcrumb navigation at the top reads 'Lambda > Functions > Lamda_prathamesh'. The function name is displayed prominently, with buttons for 'Throttle', 'Copy ARN', and 'Actions' to its right. Below the name, there are tabs for 'Function overview' (selected) and 'Info'. To the right of these tabs are buttons for 'Export to Application Composer' and 'Download'. The main content area is divided into two sections: 'Diagram' and 'Template'. The 'Diagram' section shows a visual representation of the function with a Lambda icon and a box labeled 'Lamda_prathamesh'. Below this, there is a 'Layers' section with a stack icon and '(0)' layers. To the left of the diagram is a '+ Add trigger' button, and to the right is a '+ Add destination' button. The 'Template' section is currently empty. On the right side of the console, there is a sidebar with the following information: 'Description' (empty), 'Last modified' (in 3 seconds), 'Function ARN' (arn:aws:lambda:ap-south-1:53326724563:0:function:lambda_swayam), and 'Function URL' (with an 'Info' link).

2. Creating Sqs Queue

The screenshot shows the AWS Amazon SQS console interface for a queue named 'sqs_prathamesh'. The breadcrumb navigation at the top reads 'Amazon SQS > Queues > Sqs_prathamesh'. The queue name is displayed prominently, with buttons for 'Edit', 'Delete', 'Purge', 'Send and receive messages', and 'Start DLQ redrive' to its right. Below the name, there is a 'Details' section with an 'Info' link. The rest of the console area is currently empty.

Performing Terraform commands

1. Terraform init

```
prathamesh@DESKTOP-PRATHAMESH:~/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!
You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure.
```

2. Terraform plan

```
prathamesh@DESKTOP-PRATHAMESH:~/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.
Changes to Outputs:
+ instance_id = (known after apply)
```

3. Terraform apply

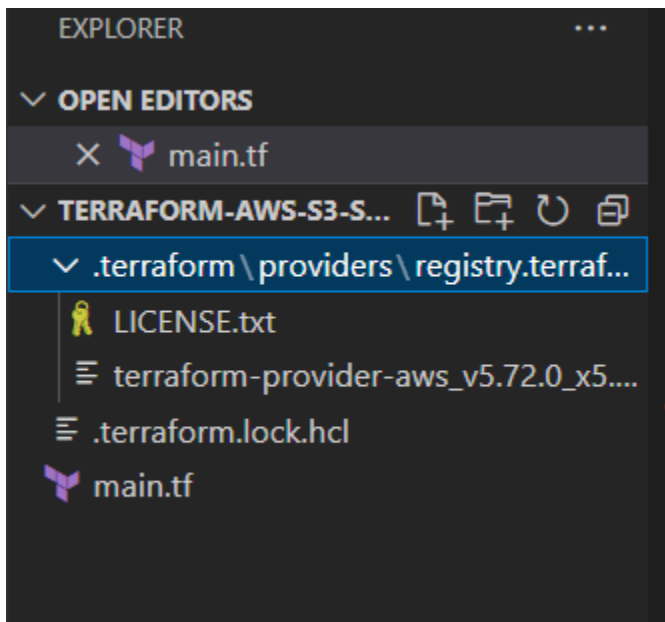
```
prathamesh@DESKTOP-PRATHAMESH:~/terraform$ terraform apply
Terraform used the selected providers to generate the following execution
plan. Resource actions are indicated with the following symbols:
+ create
Plan: 7 to add, 0 to change, 0 to destroy.
Do you want to perform these actions?
  Terraform 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.
```

```
Terraform will perform the following actions:
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.
```

4. Terraform destroy

```
prathamesh@DESKTOP-PRATHAMESH:~/terraform$ terraform destroy
Terraform used the selected providers to generate the following execution
plan. Resource actions are indicated with the following symbols:
- destroy
Terraform will perform the following actions:
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.