

Shrusti Chintawar

D15B

10

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:*"]
}
}

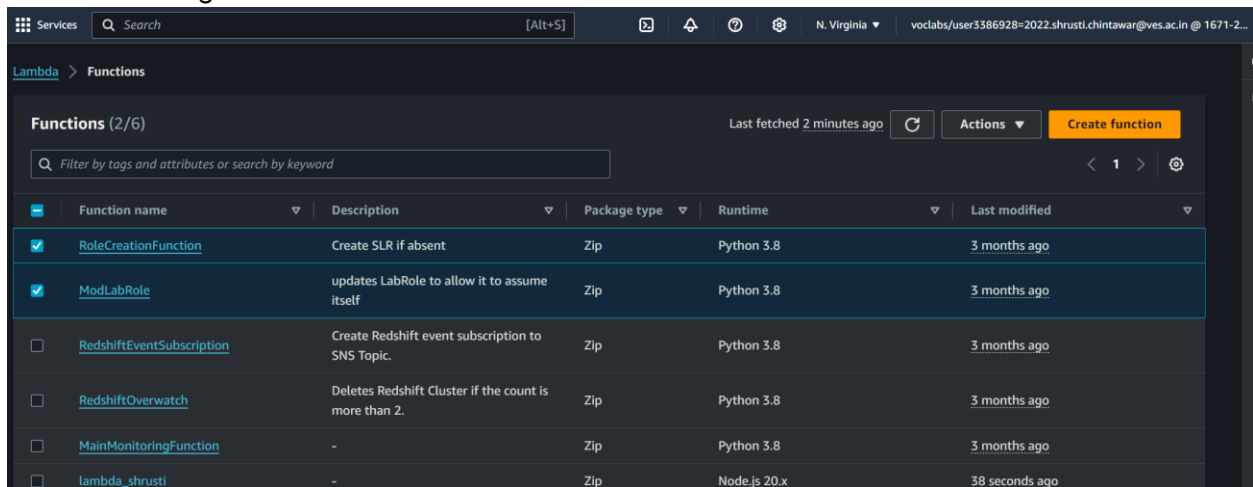
# Lambda Permission for S3 to invoke the Lambda function
resource "aws_lambda_permission" "allow_s3" {
  statement_id = "AllowS3InvokeLambda"
  action       = "lambda:InvokeFunction"
  function_name = aws_lambda_function.lambda_mayur.function_name
  principal     = "s3.amazonaws.com"

  source_arn = aws_s3_bucket.s3mayur.arn
}

```

Implementation:

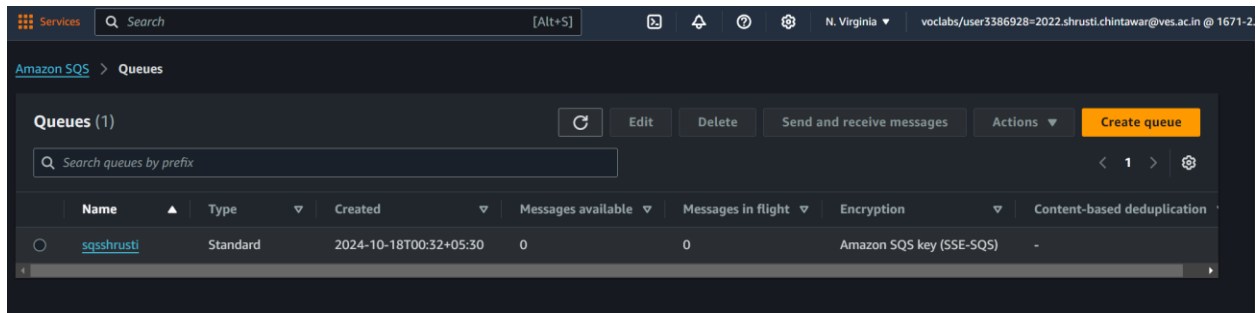
1. Creating Lambda Function



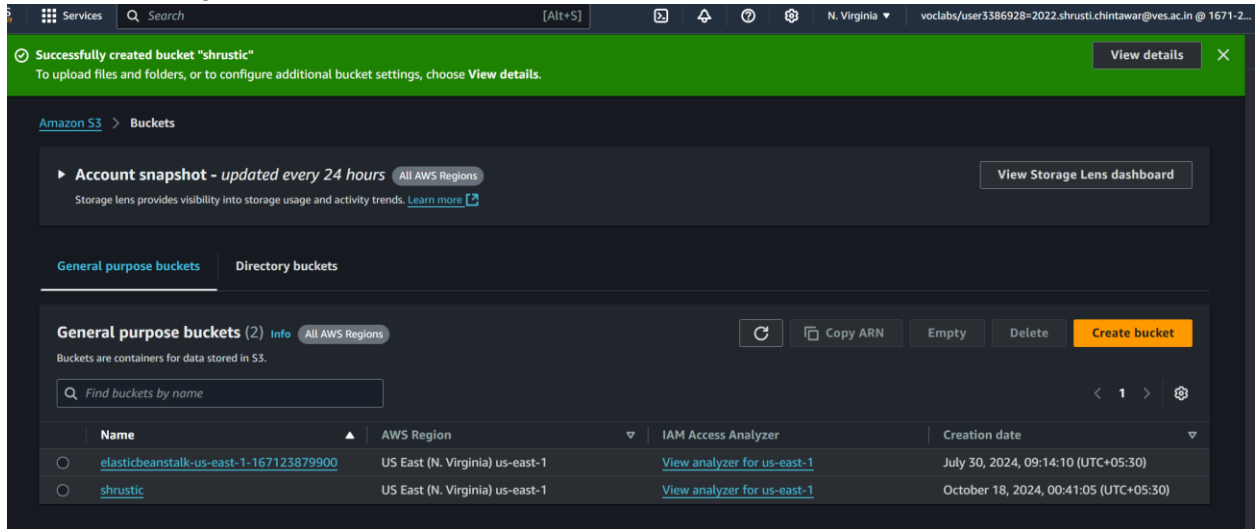
The screenshot shows the AWS Lambda console interface. At the top, there's a navigation bar with 'Services', a search bar, and the user's profile. Below the navigation bar, the 'Lambda' section is selected, and the 'Functions' page is displayed. The page shows a list of functions with columns for Function name, Description, Package type, Runtime, and Last modified. The first two functions, 'RoleCreationFunction' and 'ModLabRole', are checked. The 'Create function' button is visible in the top right corner.

Function name	Description	Package type	Runtime	Last modified
<input checked="" type="checkbox"/> RoleCreationFunction	Create SLR if absent	Zip	Python 3.8	3 months ago
<input checked="" type="checkbox"/> ModLabRole	updates LabRole to allow it to assume itself	Zip	Python 3.8	3 months ago
<input type="checkbox"/> RedshiftEventSubscription	Create Redshift event subscription to SNS Topic.	Zip	Python 3.8	3 months ago
<input type="checkbox"/> RedshiftOverwatch	Deletes Redshift Cluster if the count is more than 2.	Zip	Python 3.8	3 months ago
<input type="checkbox"/> MainMonitoringFunction	-	Zip	Python 3.8	3 months ago
<input type="checkbox"/> lambda_shrusti	-	Zip	Node.js 20.x	38 seconds ago

2. Creating Sqs Queue



3. Creating S3 Bucket



Performing Terraform commands

1. Terraform init

```
C:\Terraform_Scripts\S3>terraform init
```

```
Initializing the backend...
```

```
Initializing provider plugins...
```

```
- Finding latest version of hashicorp/aws...  
- Installing hashicorp/aws v4.25.0...  
- Installed hashicorp/aws v4.25.0 (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.
```

```
C:\Terraform_Scripts\S3>_
```

2. Terraform plan

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

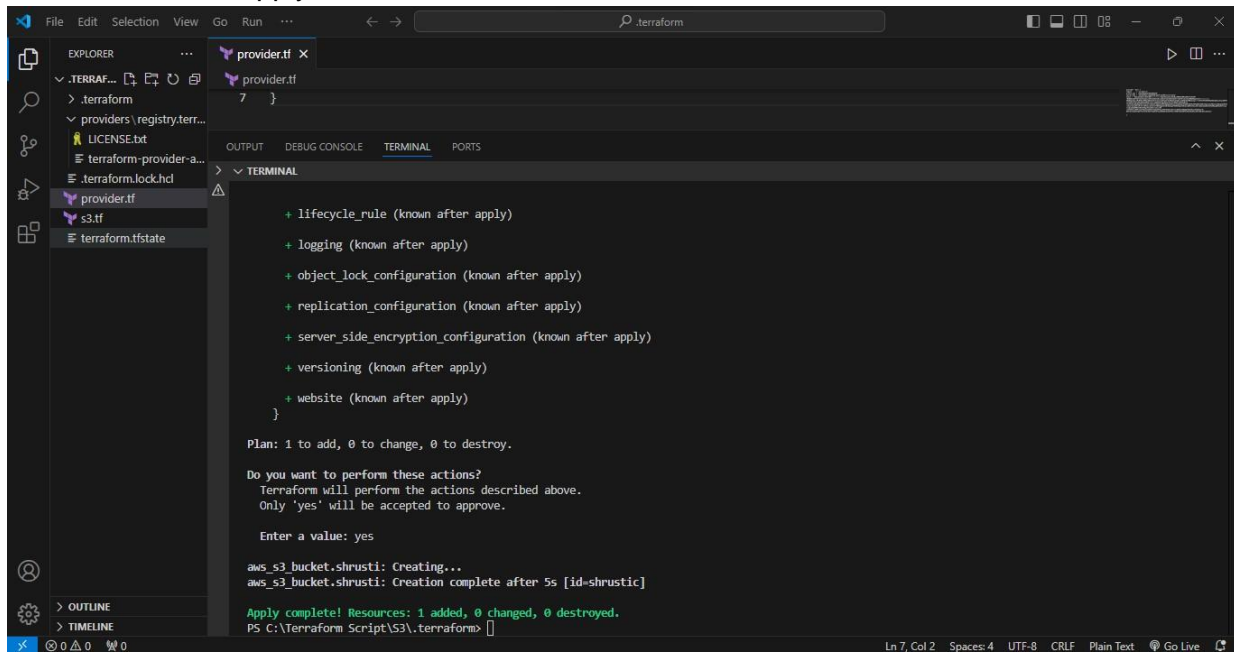
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

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

Terraform will perform the following actions:

```
# aws_s3_bucket.shrusti will be created  
+ resource "aws_s3_bucket" "shrusti" {  
  + acceleration_status = (known after apply)  
  + acl                 = (known after apply)  
  + arn                 = (known after apply)  
  + bucket              = "shrustic"  
  + bucket_domain_name = (known after apply)  
  + bucket_prefix       = (known after apply)  
  + bucket_regional_domain_name = (known after apply)  
  + force_destroy       = false  
  + hosted_zone_id      = (known after apply)  
  + id                  = (known after apply)  
  + object_lock_enabled = (known after apply)  
  + policy              = (known after apply)  
  + region              = (known after apply)  
  + request_payer       = (known after apply)  
  + tags                = {  
    + "Environment" = "Dev"  
    + "Name"        = "My bucket"  
  }  
+ tags_all              = {  
  + "Environment" = "Dev"  
  + "Name"        = "My bucket"  
}
```

3. Terraform apply



```
provider.tf
7 }
```

OUTPUT DEBUG CONSOLE TERMINAL PORTS

TERMINAL

```
+ lifecycle_rule (known after apply)
+ logging (known after apply)
+ object_lock_configuration (known after apply)
+ replication_configuration (known after apply)
+ server_side_encryption_configuration (known after apply)
+ versioning (known after apply)
+ website (known after apply)
}

Plan: 1 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

aws_s3_bucket.shrusti: Creating...
aws_s3_bucket.shrusti: Creation complete after 5s [id=shrustic]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Terraform Script\S3\terraform>
```

Folder structure of main.tf file

```
s3.tf
1 resource "aws_s3_bucket" "shrustic" {
2     bucket = "shrustic"
3
4     tags = {
5         Name      = "My Bucket"
6         Environment = "Dev"
7     }
8 }
```

Welcome s3.tf provider.tf

```
provider.tf
1 provider "aws" {
2     access_key= "ASIASN2K4A7OOR75NLWJ"
3     secret_key="ycT+D4TLr8GwjCVnpwrPUmzQogPoqXeNU4EIQBDK"
4     region "ap-south-1"
5 }
```

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.