Vivekanand Education Society's

Institute of Technology

An Autonomous Institute Affiliated to University of Mumbai,, Approved by AICTE & Recognized by Govt. of Maharashtra

Hashu Advani Memorial Complex, Collector Colony, Chembur East, Mumbai - 400074.

Department of Information Technology

A.Y. 2024-25

Advance DevOps Lab
Assignment 02

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

Roll No.	42
Name	NAIKWADI YASH SHIVDAS
Class	D15B
Subject	Advance DevOps Lab
LO Mapped	LO6: To engineer a composition of nano services using AWS Lambda and Step Functions with the Serverless Framework
Grade:	

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

Theory :

Infrastructure as Code (IaC)
Infrastructure as Code (IaC) automates the management of IT infrastructure through code rather than manual processes, enabling consistent and repeatable deployments.

Overview of Terraform
Terraform is an open-source IaC tool that allows users to define cloud infrastructure using HashiCorp Configuration Language (HCL). Key features include:

- **Declarative Configuration:** Users specify the desired state of the infrastructure.
- **Execution Plan:** Terraform generates a plan detailing the actions needed to achieve that state.
- **Resource Management:** It manages the lifecycle of cloud resources.

Amazon S3 (Simple Storage Service)
Amazon S3 is a scalable storage solution that allows users to store and retrieve data from anywhere. Key features include:

- **Buckets:** Containers for organizing data.
- **Object Storage:** Stores data as objects with unique identifiers.
- **Use Cases:** Backup, data archiving, and serving static content.

Amazon SQS (Simple Queue Service)
Amazon SQS is a managed message queuing service that decouples application components, allowing for asynchronous communication. Key features include:

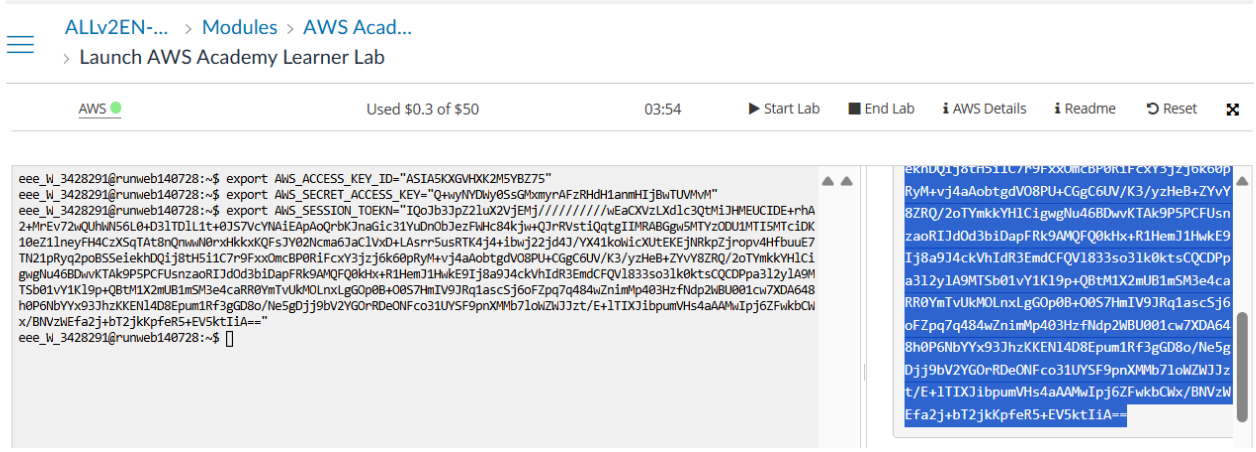
- **Queues:** Store messages for processing by consumers.
- **Message Retention:** Retains messages for a configurable time.
- **Use Cases:** Event-driven architectures and inter-service communication.

AWS Lambda
AWS Lambda is a serverless computing service that runs code without managing servers. Key features include:

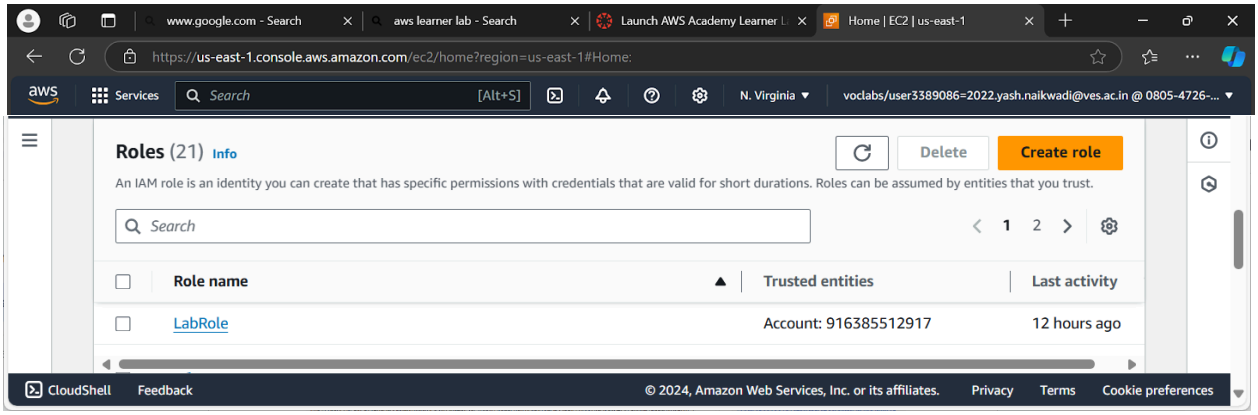
- **Event-Driven Execution:** Triggered by AWS services like S3 and SQS.
- **Pay-as-You-Go Pricing:** Users pay only for the compute time used.
- **Use Cases:** Data processing and responding to events.

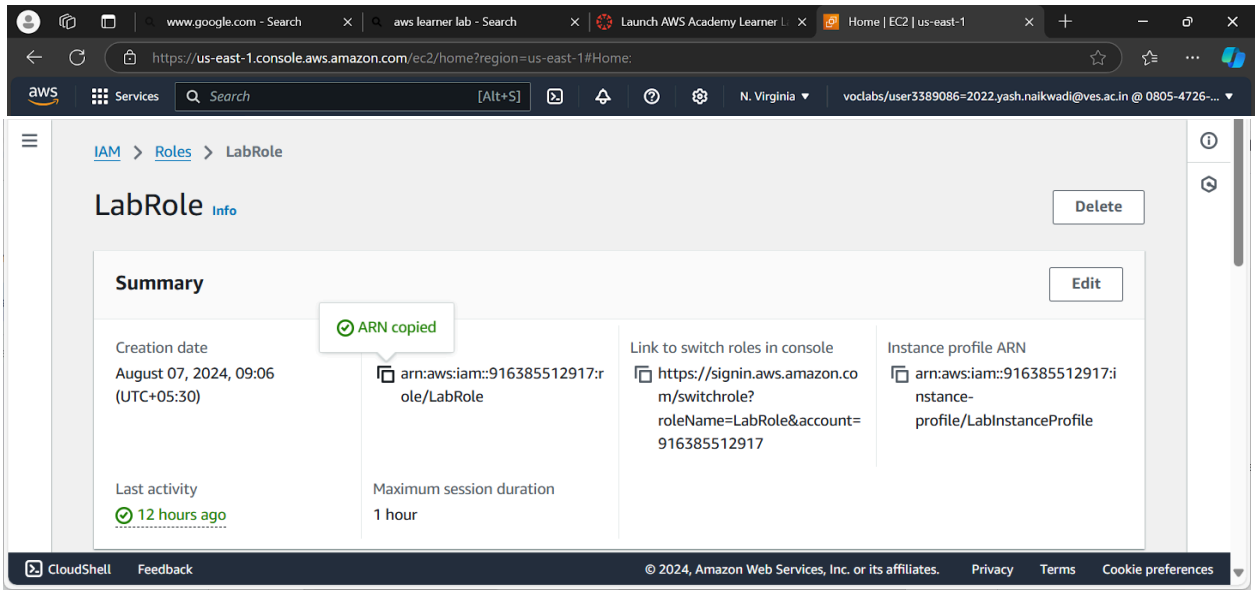
Integration of S3, SQS, and Lambda
Integrating these services enables powerful workflows. For example, an object uploaded to S3 can trigger a Lambda function, which processes the data and sends a message to SQS, allowing other services to react asynchronously.

Start the Learner Lab and export the credentials from the CLI.



In the Learner Lab, there is usually a predefined IAM role that you can use. This role should already have the necessary permissions to interact with AWS services (like Lambda and S3).





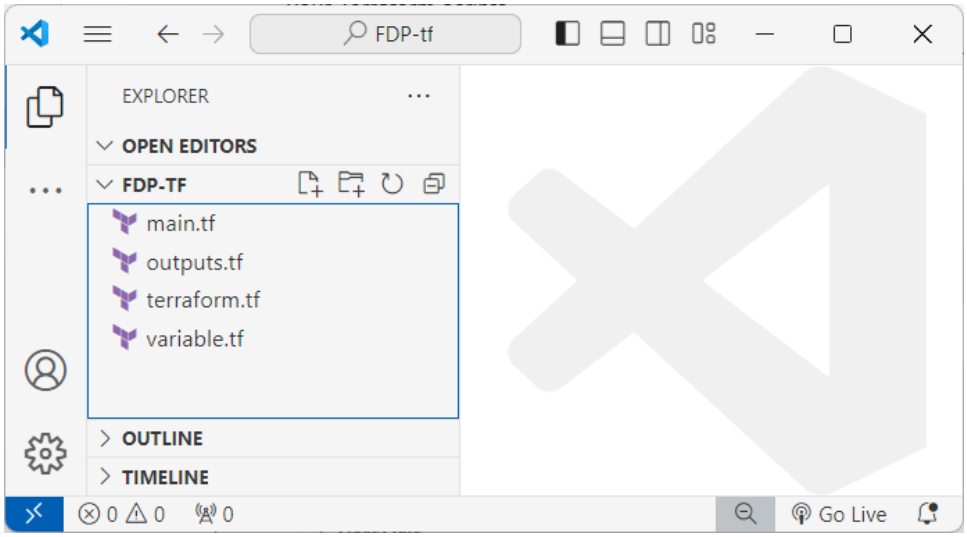
Create a Folder for the Project:

- Create a new folder on your local machine (for example: FDP-tf) where you will store your Terraform scripts.

Set Up Terraform Configuration:

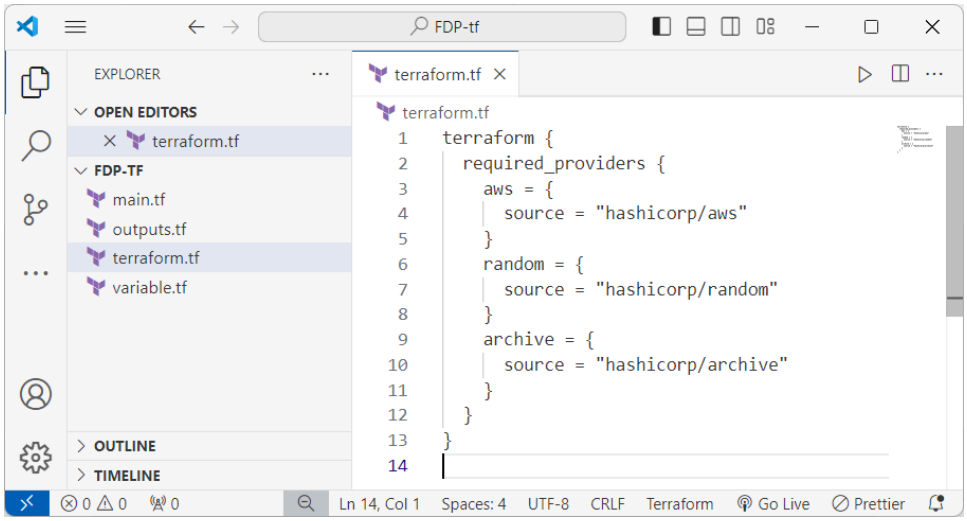
- Inside your folder, create four files:
  - terraform.tf
  - main.tf
  - variable.tf

- outputs.tf



terraform.tf

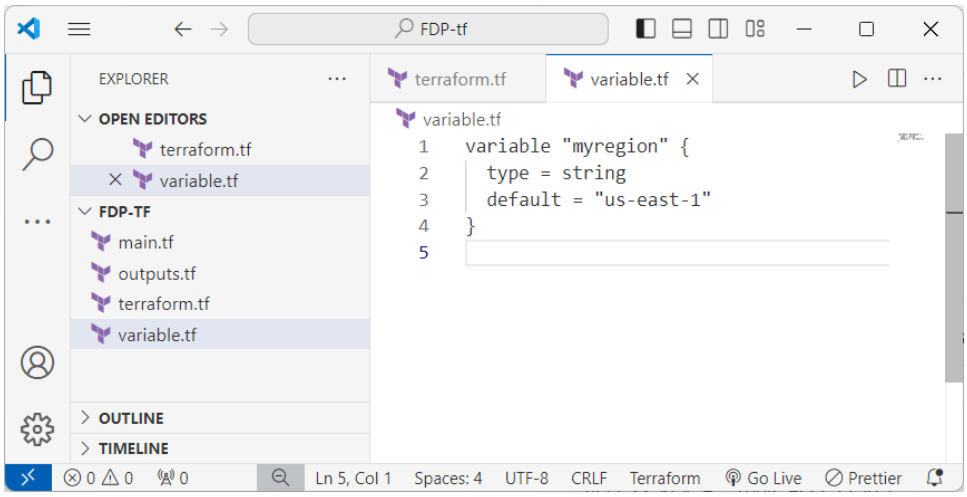
```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
    }
    random = {
      source = "hashicorp/random"
    }
    archive = {
      source = "hashicorp/archive"
    }
  }
}
```



variable.tf

```
variable "myregion" {
  type = string
  default = "us-east-1"
```

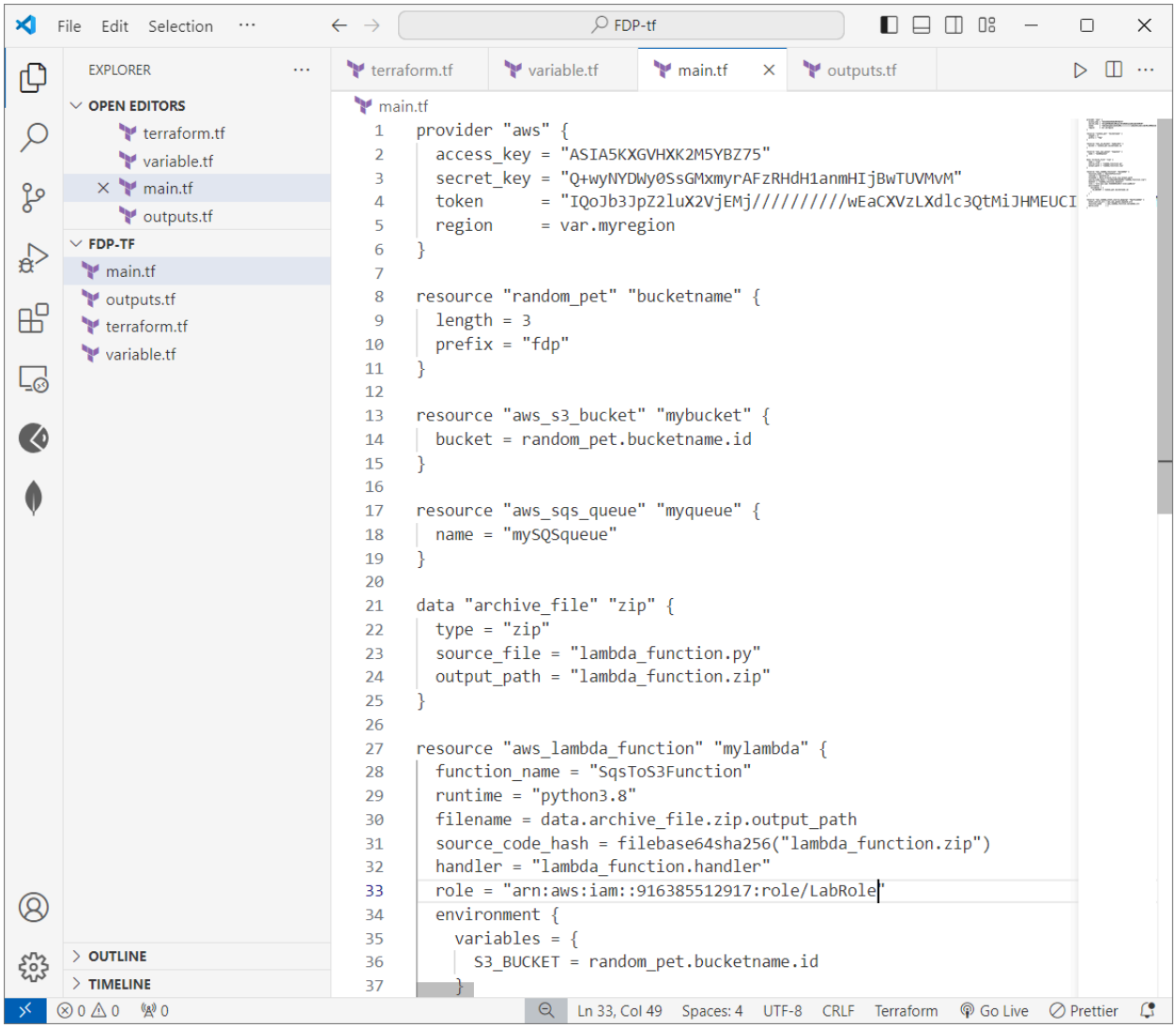
```
}
```



main.tf

```
provider "aws" {
  access_key = "YOUR_ACCESS_KEY"
  secret_key = "YOUR_SECRET_KEY"
  token      = "YOUR_TOKEN"
  region     = var.myregion
}

resource "random_pet" "bucketname" {
  length = 3
  prefix = "fdp"
}

resource "aws_s3_bucket" "mybucket" {
  bucket = random_pet.bucketname.id
}

resource "aws_sqs_queue" "myqueue" {
  name = "mySQSqueue"
}

data "archive_file" "zip" {
  type = "zip"
  source_file = "lambda_function.py"
  output_path = "lambda_function.zip"
}

resource "aws_lambda_function" "mylambda" {
  function_name = "SqsToS3Function"
  runtime = "python3.8"
  filename = data.archive_file.zip.output_path
  source_code_hash = filebase64sha256("lambda_function.zip")
```

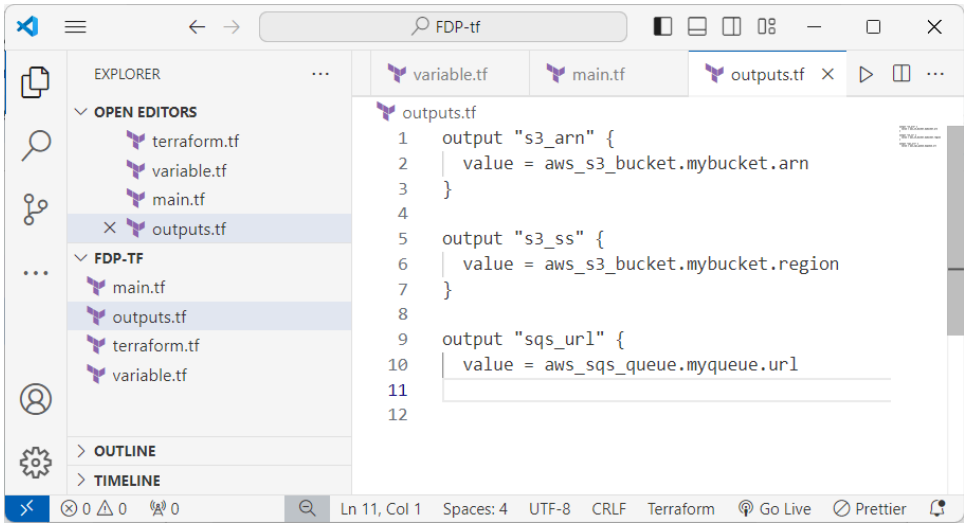
```
  handler = "lambda_function.handler"
  role = "arn:aws:iam::YOUR_IAM_ROLE"
  environment {
    variables = {
      S3_BUCKET = random_pet.bucketname.id
    }
  }
}

resource "aws_lambda_event_source_mapping" "SqsToLambda" {
  event_source_arn = aws_sqs_queue.myqueue.arn
  function_name    = aws_lambda_function.mylambda.arn
  batch_size       = 1
}
```



outputs.tf

```
output "s3_arn" {
  value = aws_s3_bucket.mybucket.arn
```

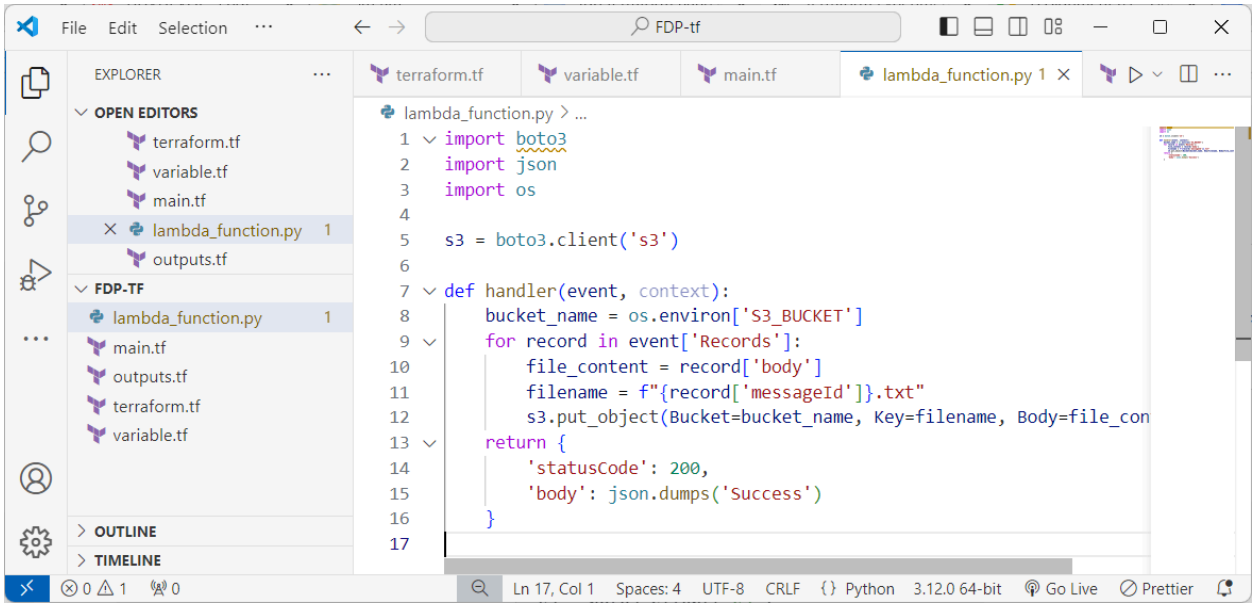
```
}

output "s3_ss" {
  value = aws_s3_bucket.mybucket.region
}

output "sqs_url" {
  value = aws_sqs_queue.myqueue.url
}
```



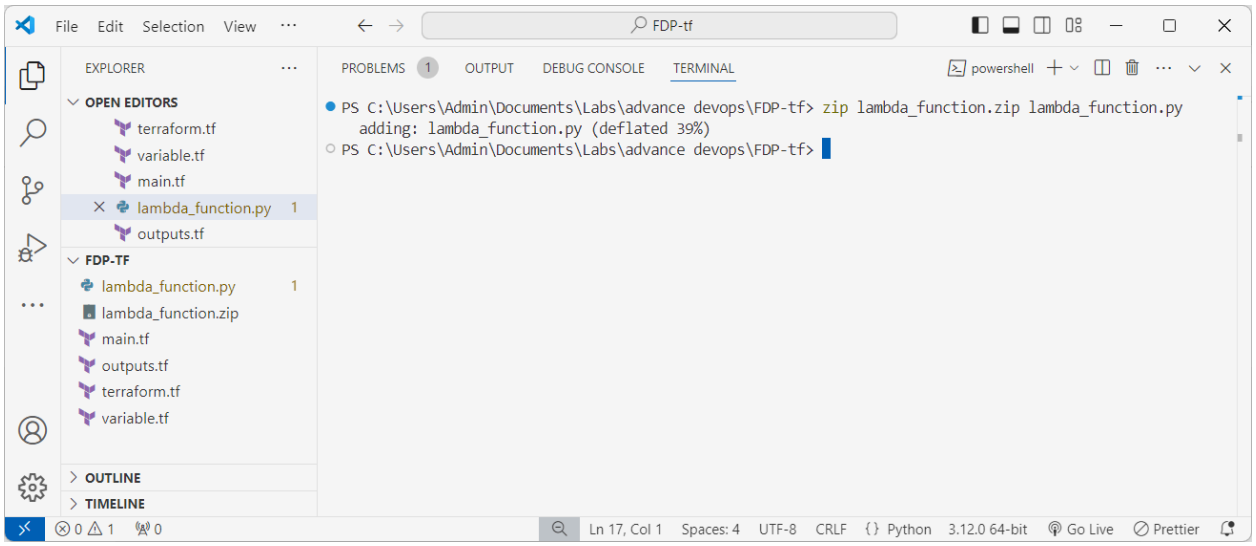
Create Lambda Python File:

- In the same directory, create a file named lambda_function.py and paste the following code:

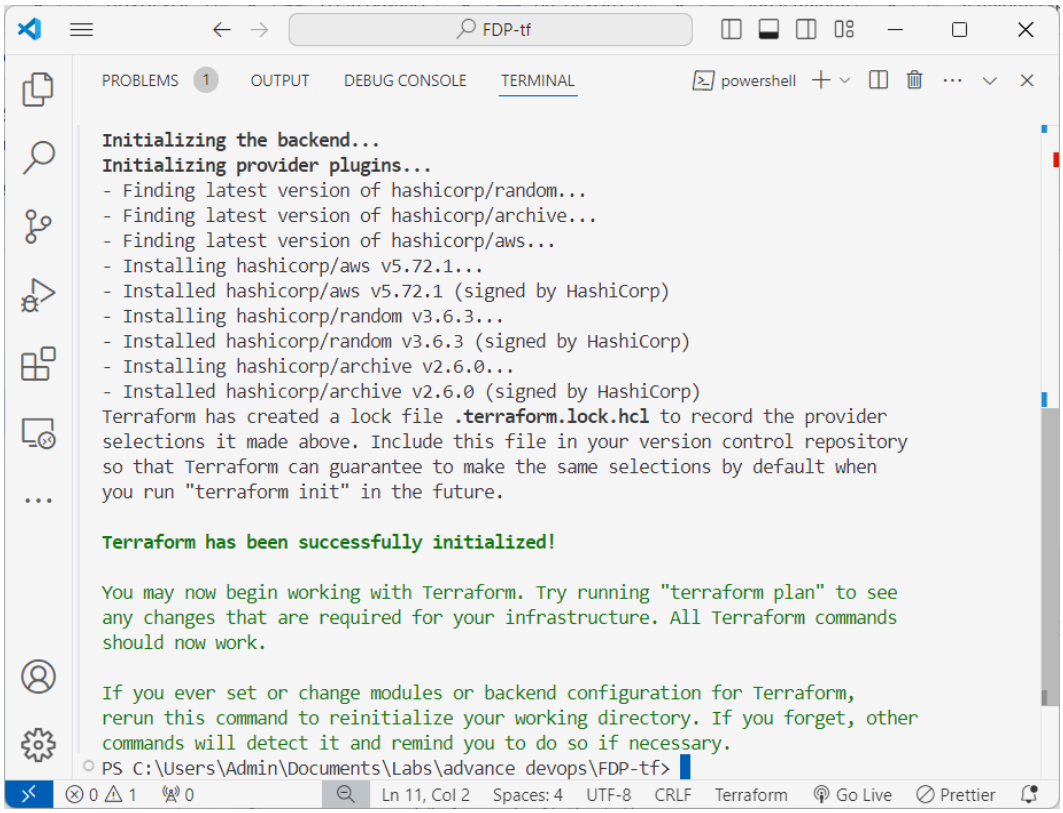
```
import boto3
import json
import os

s3 = boto3.client('s3')

def handler(event, context):
    bucket_name = os.environ['S3_BUCKET']
    for record in event['Records']:
        file_content = record['body']
        filename = f"{record['messageId']}.txt"
        s3.put_object(Bucket=bucket_name, Key=filename, Body=file_content)
    return {
        'statusCode': 200,
        'body': json.dumps('Success')
    }
```



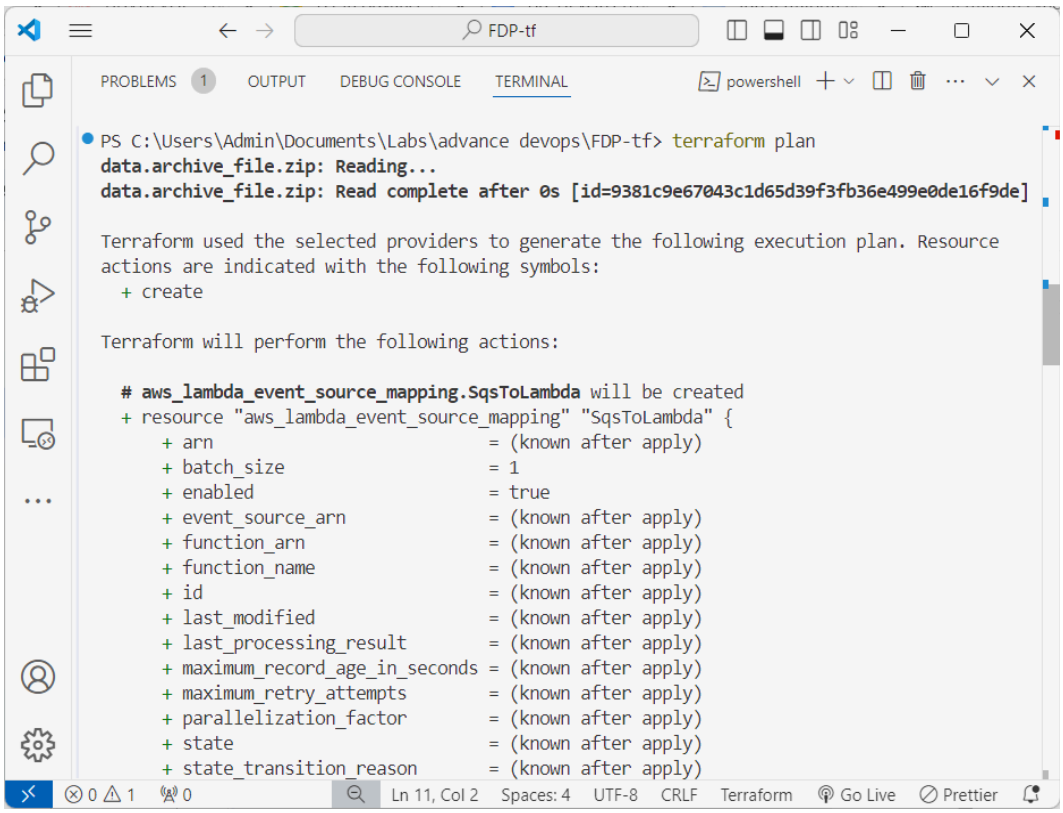
Open the terminal or command prompt in the same directory and run:
zip lambda_function.zip lambda_function.py

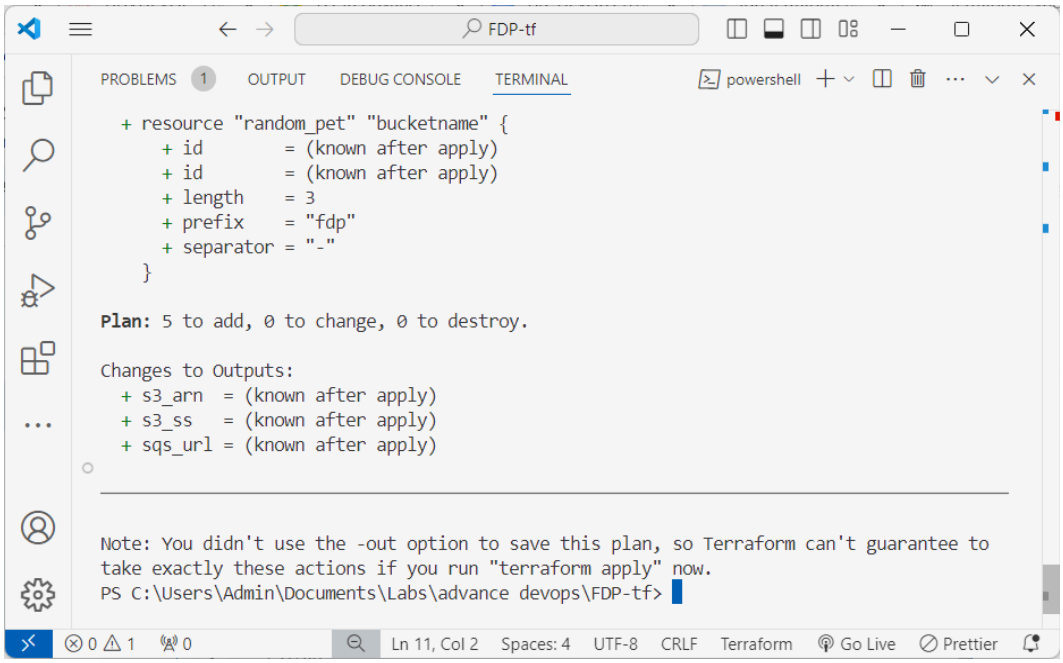


To Initialize Terraform, run the command:
terraform init



To **Plan the Infrastructure**, Run the command
terraform plan

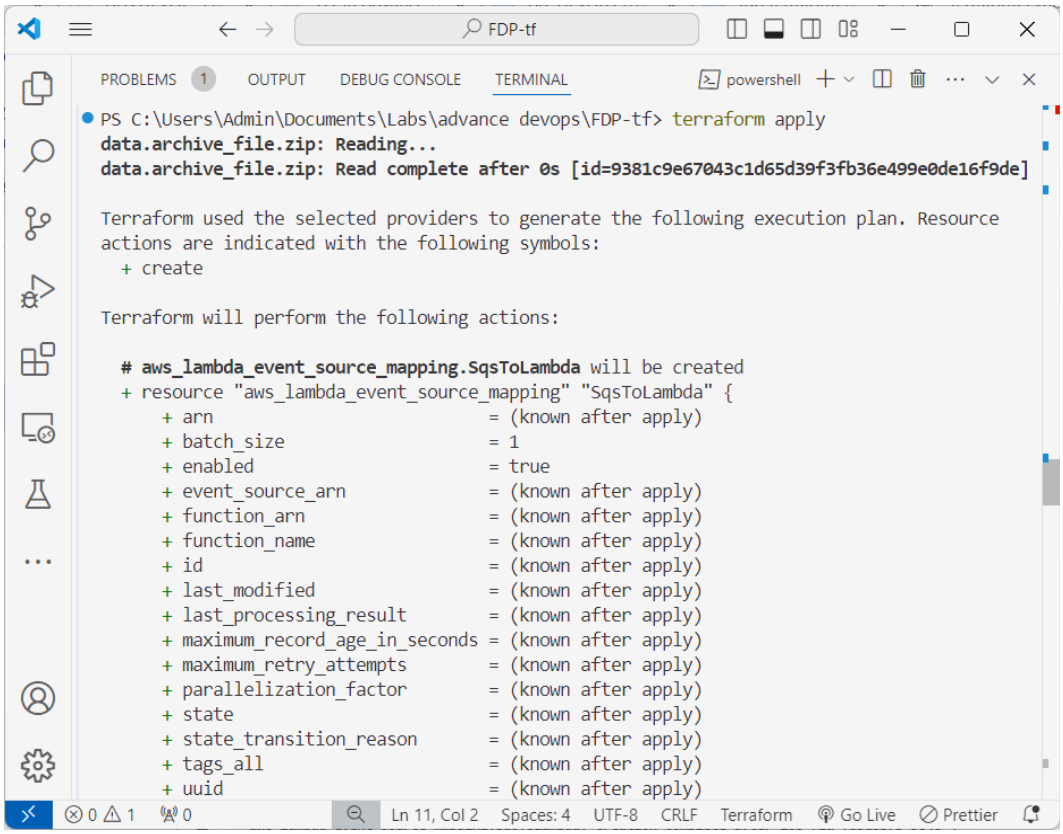


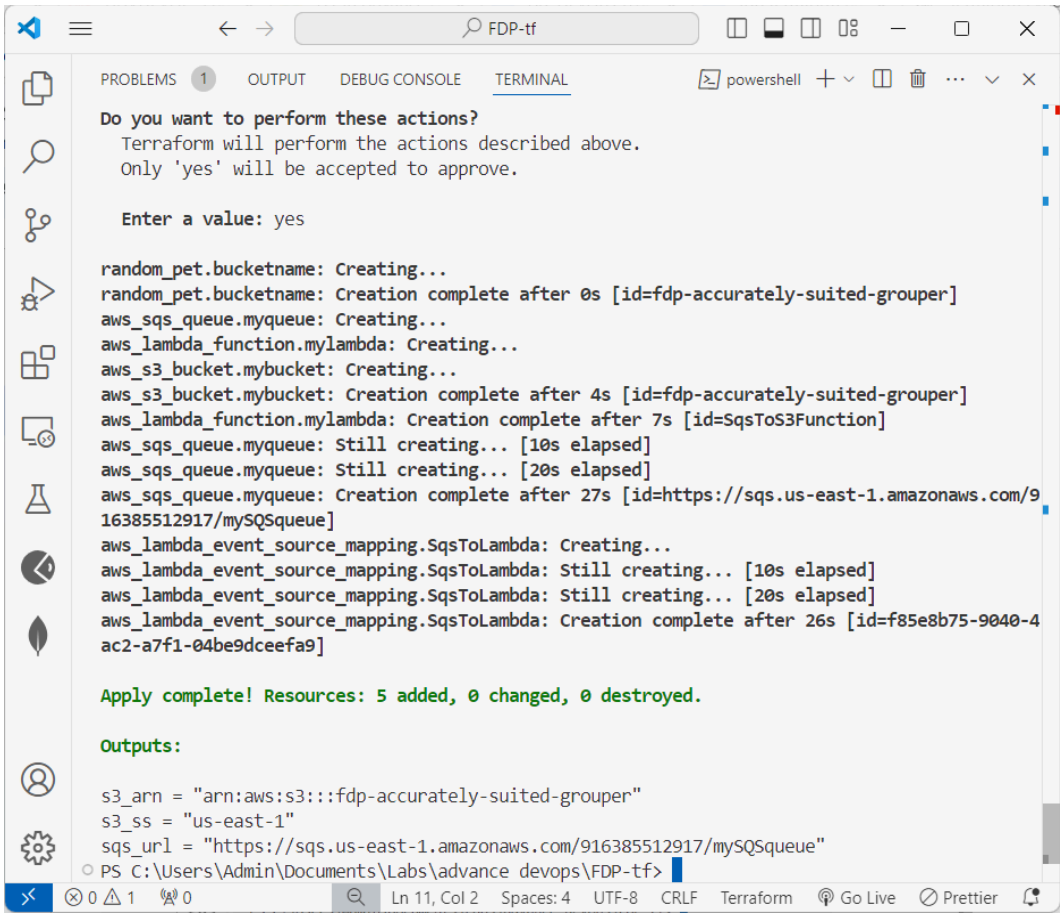


If everything looks good, apply the plan by running

terraform apply

(Enter yes when prompted. This will create the resources.)

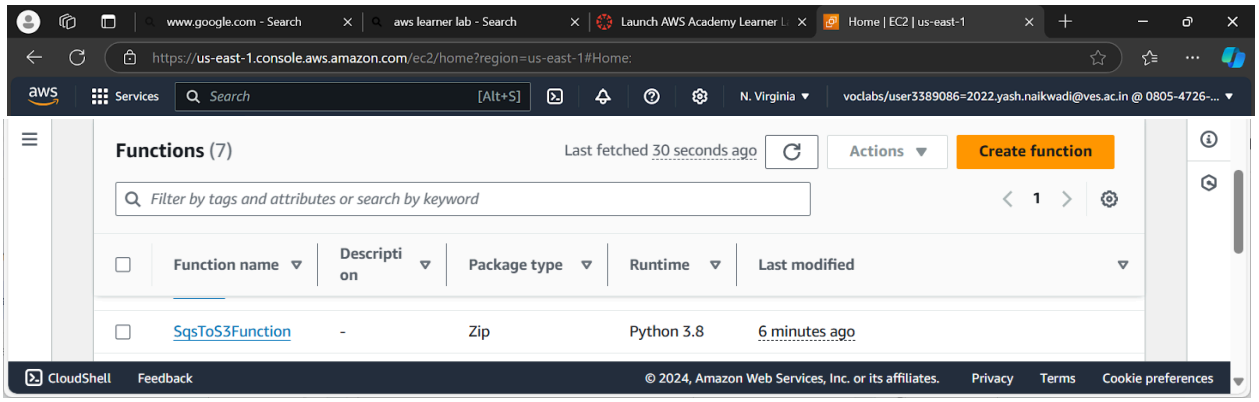


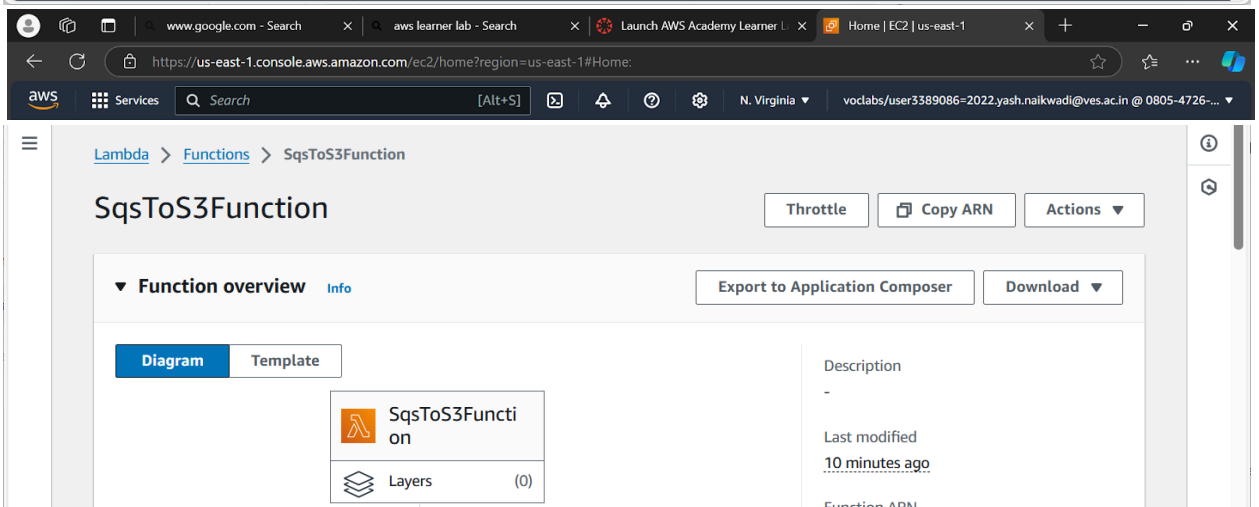


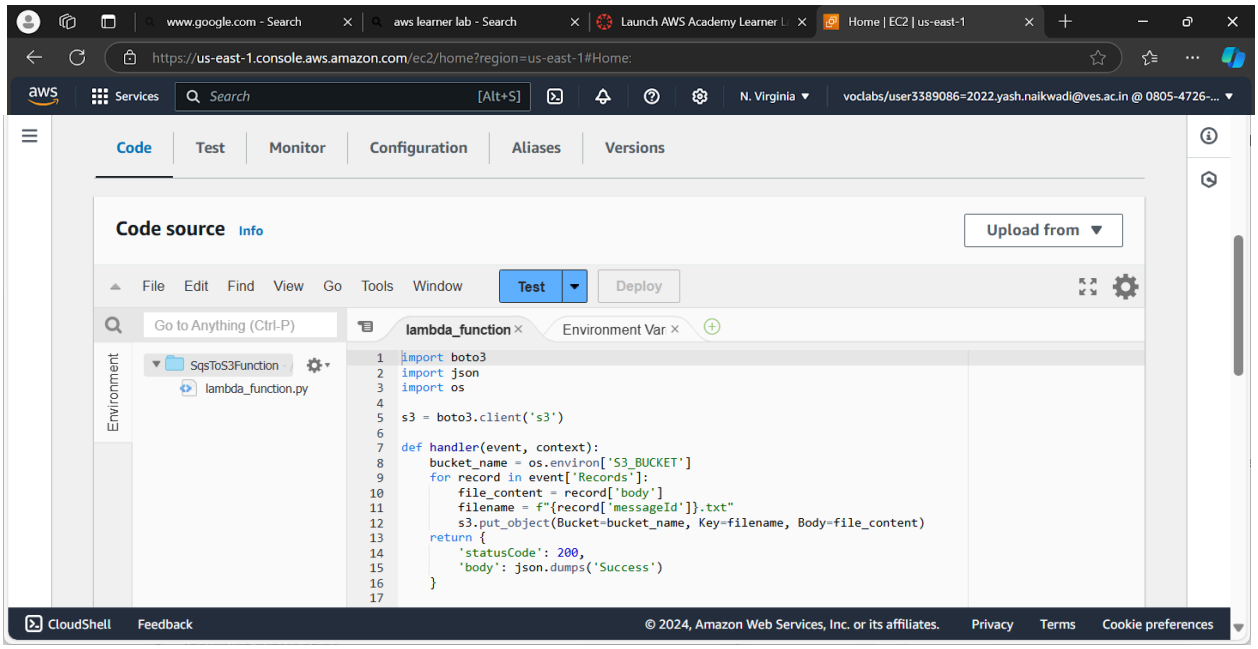
Once the resources are created, you can log into your AWS console and verify that:

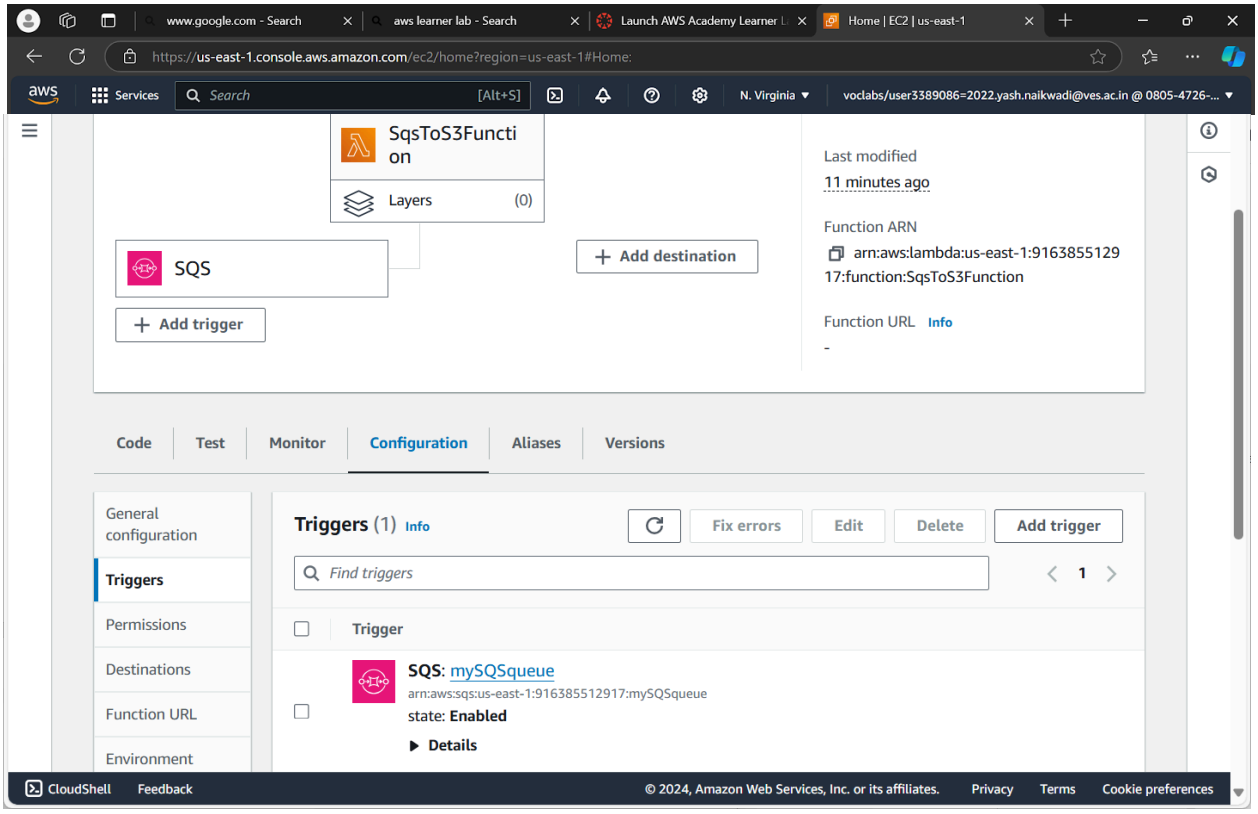
- An S3 bucket is created.
- An SQS queue is created.
- A Lambda function is created.

Lambda Function

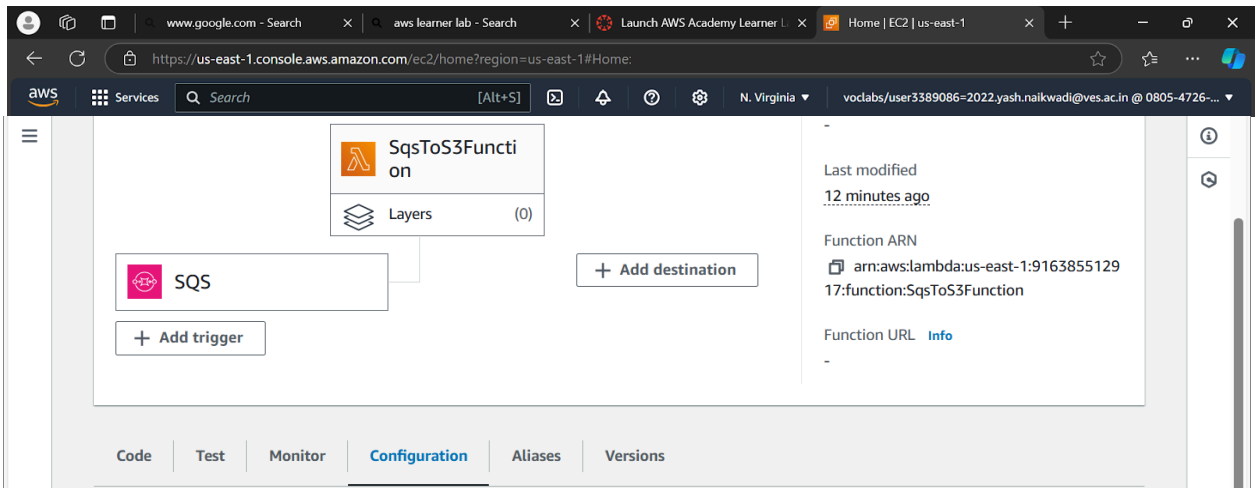


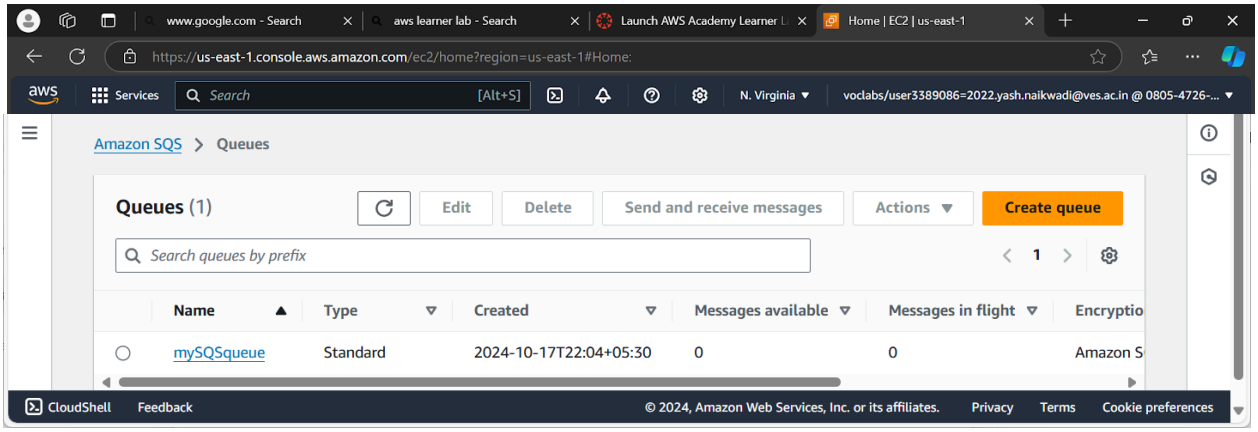


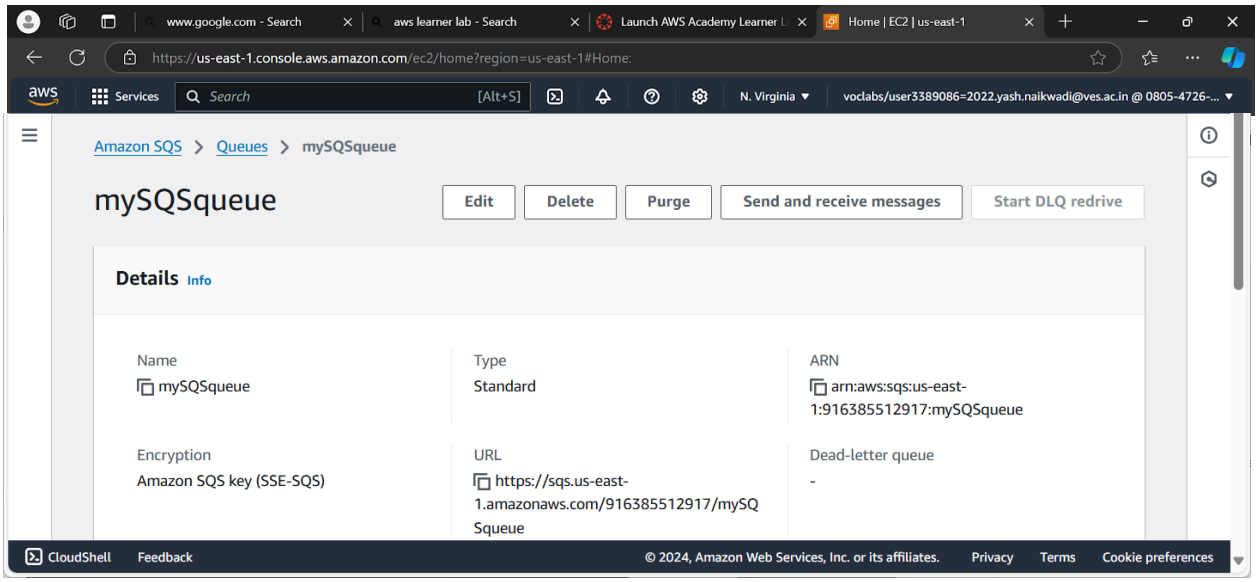

Code
Test
Monitor
Configuration
Aliases
Versions
Code source Info
Upload from
File Edit Find View Go Tools Window
Test
Deploy
Go to Anything (Ctrl-P)
lambda_function
Environment Var
Environment
SqsToS3Function
lambda_function.py
import boto3
import json
import os
s3 = boto3.client('s3')
def handler(event, context):
bucket_name = os.environ['S3_BUCKET']
for record in event['Records']:
file_content = record['body']
filename = f"{record['messageId']}.txt"
s3.put_object(Bucket=bucket_name, Key=filename, Body=file_content)
return {
'statusCode': 200,
'body': json.dumps('Success')
}
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


SqsToS3Function
Layers (0)
SQS
Add trigger
Add destination
Last modified
11 minutes ago
Function ARN
arn:aws:lambda:us-east-1:916385512917:function:SqsToS3Function
Function URL Info
-
Code
Test
Monitor
Configuration
Aliases
Versions
General configuration
Triggers
Permissions
Destinations
Function URL
Environment
Triggers (1) Info
Fix errors
Edit
Delete
Add trigger
Find triggers
Trigger
SQS: mySQSqueue
arn:aws:sqs:us-east-1:916385512917:mySQSqueue
state: Enabled
Details
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

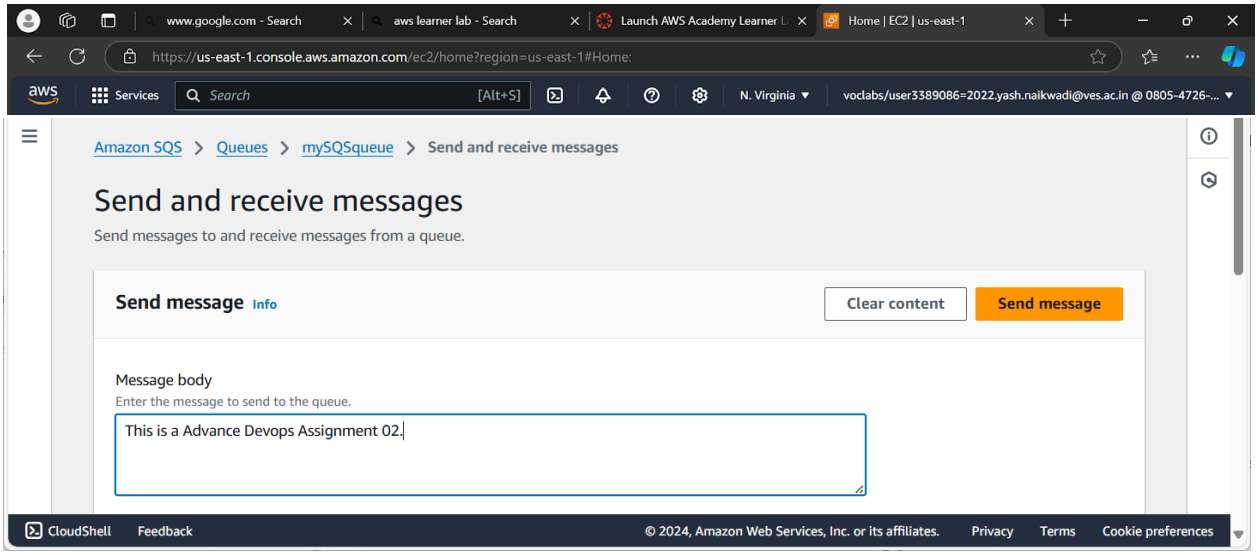
SQS queue

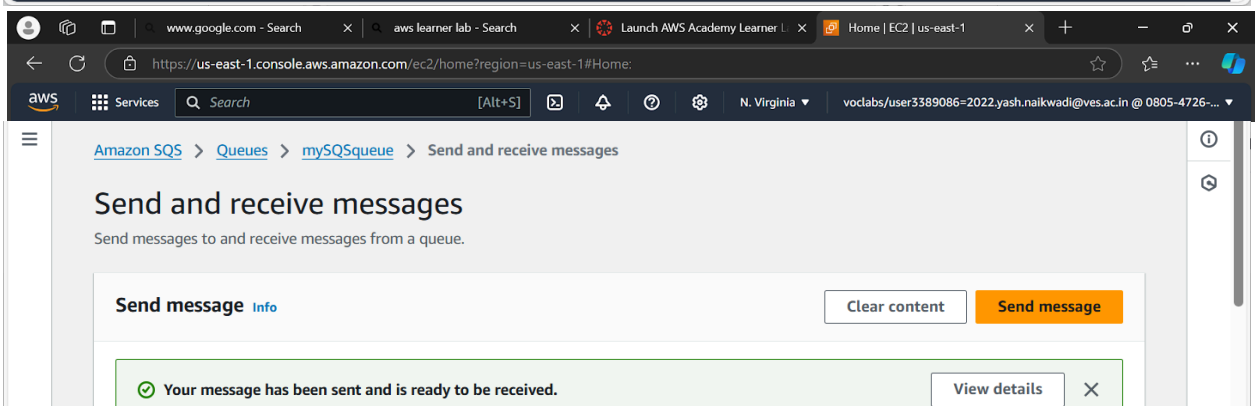

SqsToS3Function
Layers (0)
SQS
Add trigger
Add destination
Last modified
12 minutes ago
Function ARN
arn:aws:lambda:us-east-1:916385512917:function:SqsToS3Function
Function URL Info
-
Code
Test
Monitor
Configuration
Aliases
Versions



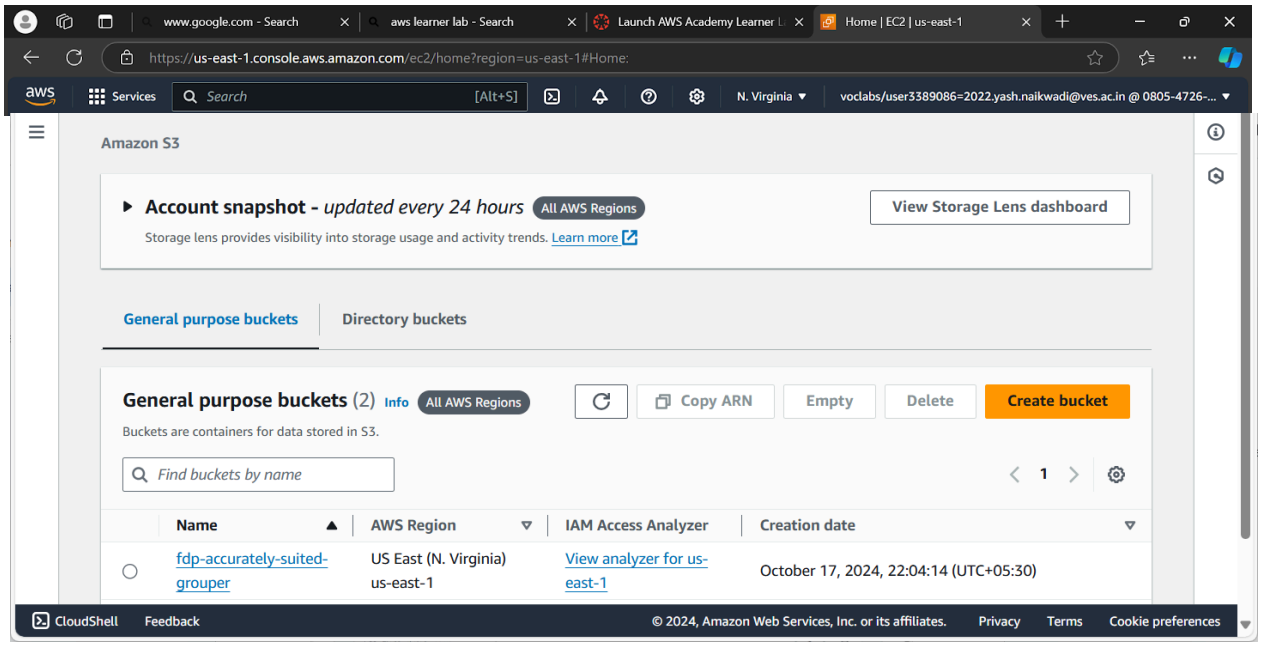


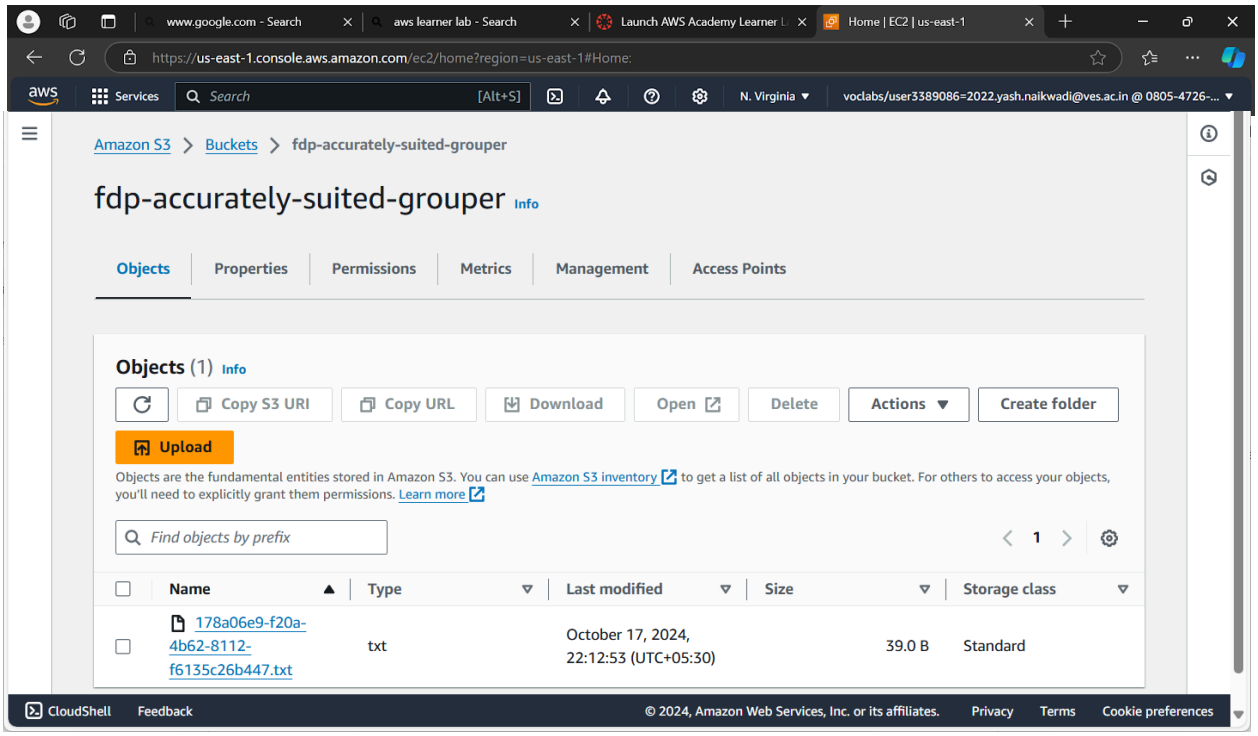
Send the message from the SQS.

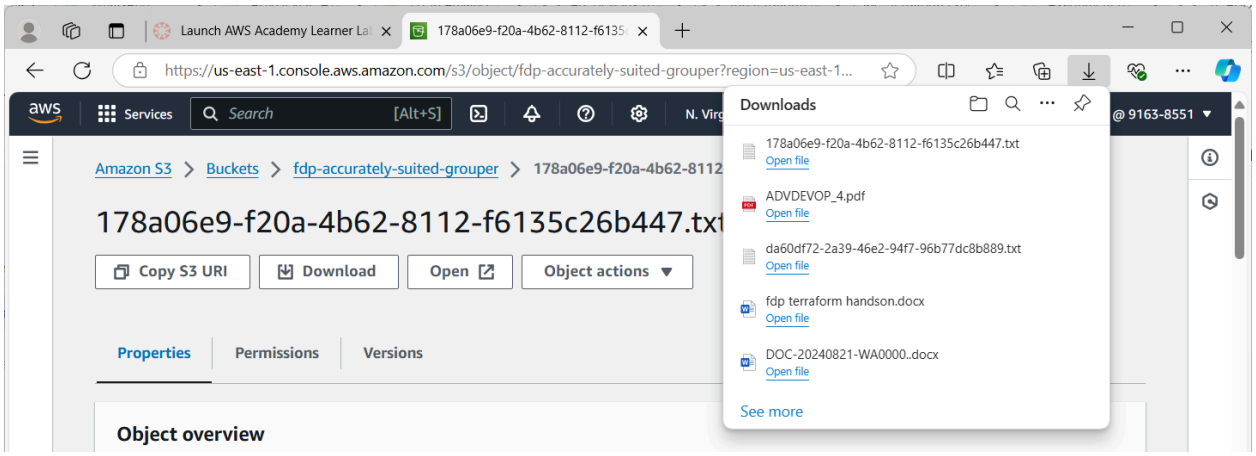


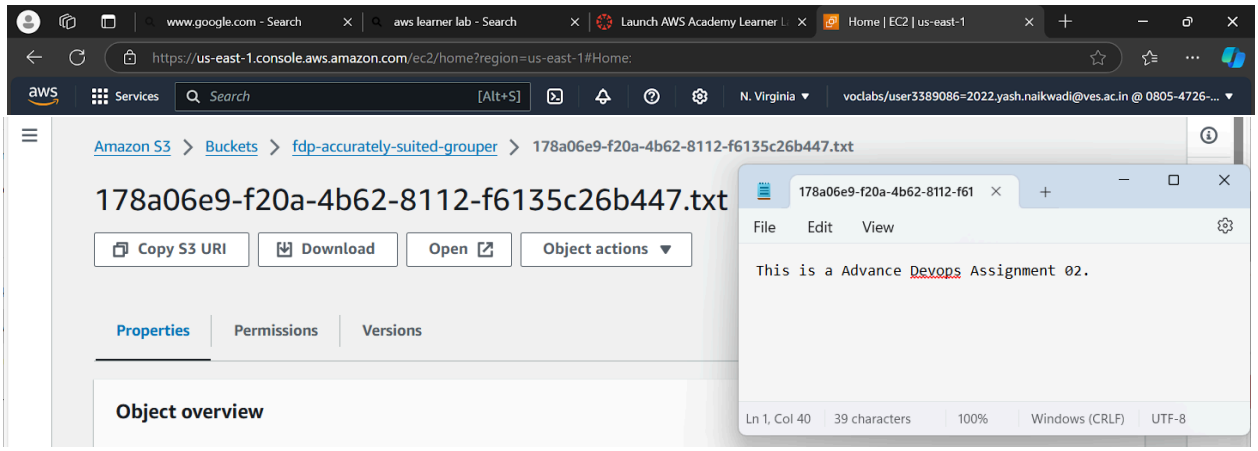


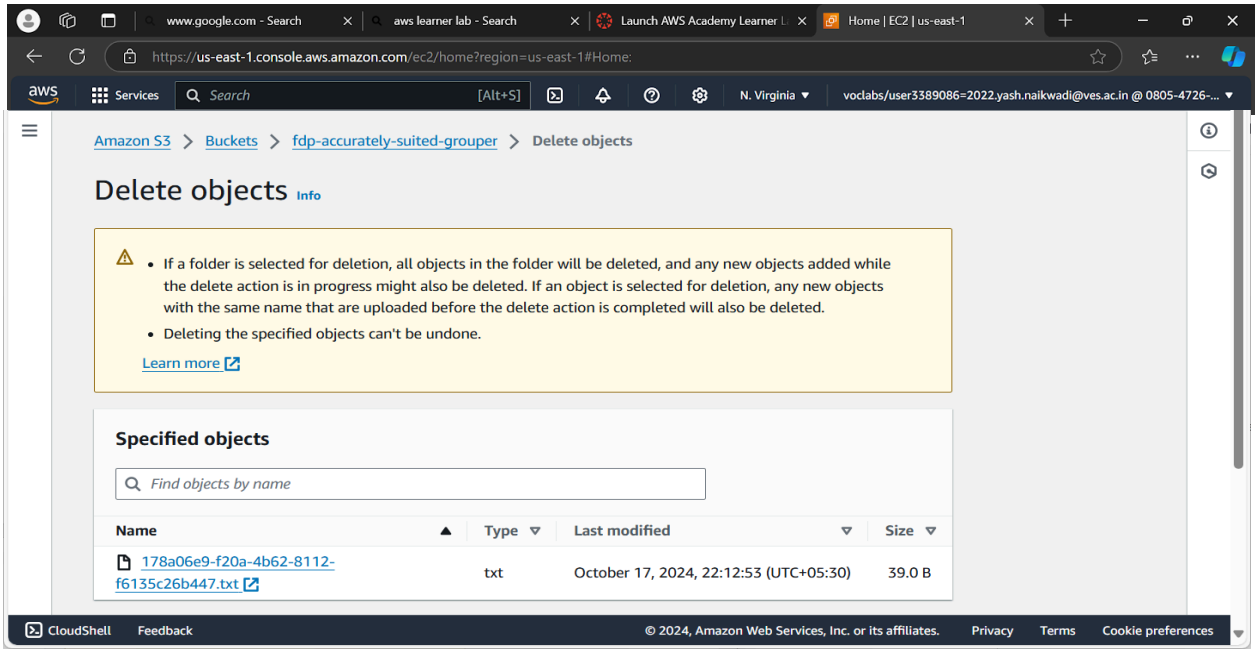
S3 bucket

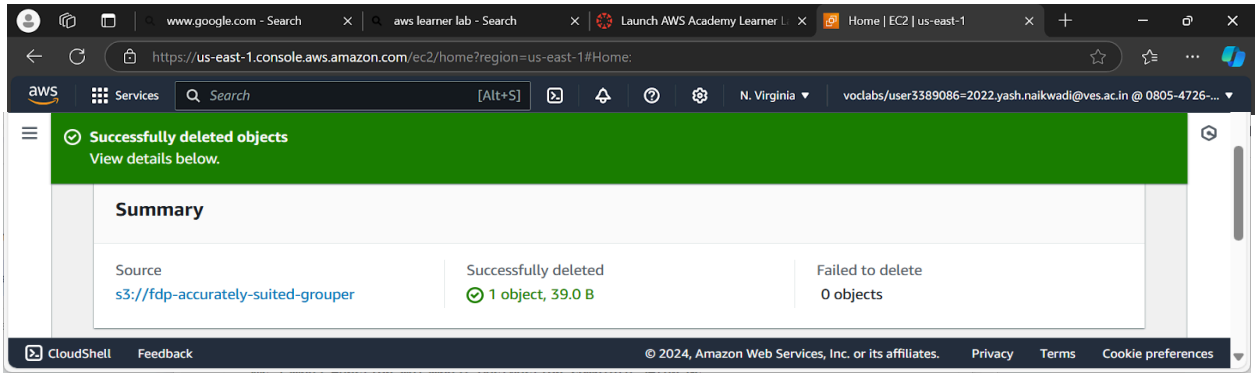

Amazon S3
Account snapshot - updated every 24 hours
All AWS Regions
View Storage Lens dashboard
Storage lens provides visibility into storage usage and activity trends. Learn more
General purpose buckets
Directory buckets
General purpose buckets (2) Info
All AWS Regions
Copy ARN
Empty
Delete
Create bucket
Buckets are containers for data stored in S3.
Find buckets by name
Name
AWS Region
IAM Access Analyzer
Creation date
fdp-accurately-suited-grouper
US East (N. Virginia) us-east-1
View analyzer for us-east-1
October 17, 2024, 22:04:14 (UTC+05:30)
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Amazon S3 > Buckets > fdp-accurately-suited-grouper
fdp-accurately-suited-grouper Info
Objects
Properties
Permissions
Metrics
Management
Access Points
Objects (1) Info
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Create folder
Upload
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more
Find objects by prefix
Name
Type
Last modified
Size
Storage class
178a06e9-f20a-4b62-8112-f6135c26b447.txt
txt
October 17, 2024, 22:12:53 (UTC+05:30)
39.0 B
Standard
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

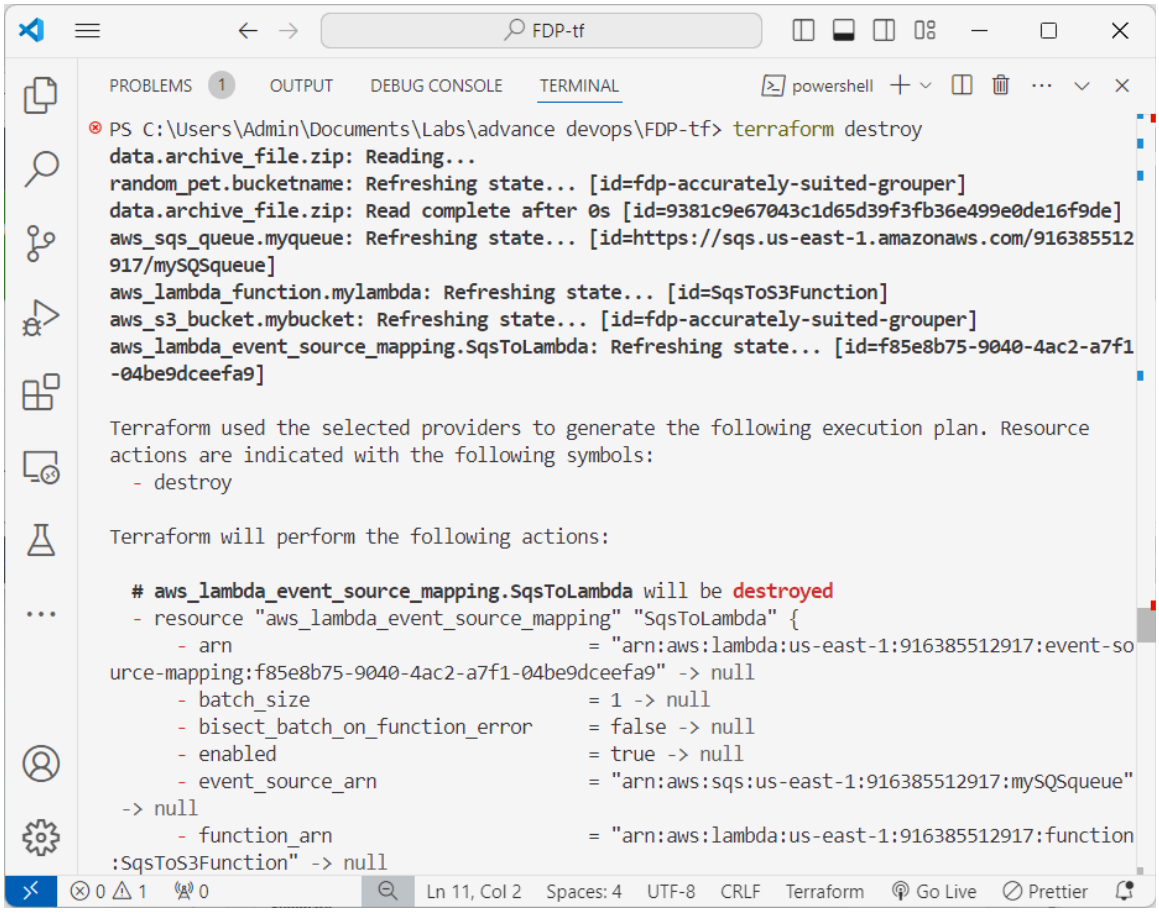

Downloads
178a06e9-f20a-4b62-8112-f6135c26b447.txt
Open file
ADVDEVOP_4.pdf
Open file
da60df72-2a39-46e2-94f7-96b77dc8b889.txt
Open file
fdp terraform handson.docx
Open file
DOC-20240821-WA0000..docx
Open file
See more
Amazon S3 > Buckets > fdp-accurately-suited-grouper > 178a06e9-f20a-4b62-8112
178a06e9-f20a-4b62-8112-f6135c26b447.txt
Copy S3 URI
Download
Open
Object actions
Properties
Permissions
Versions
Object overview

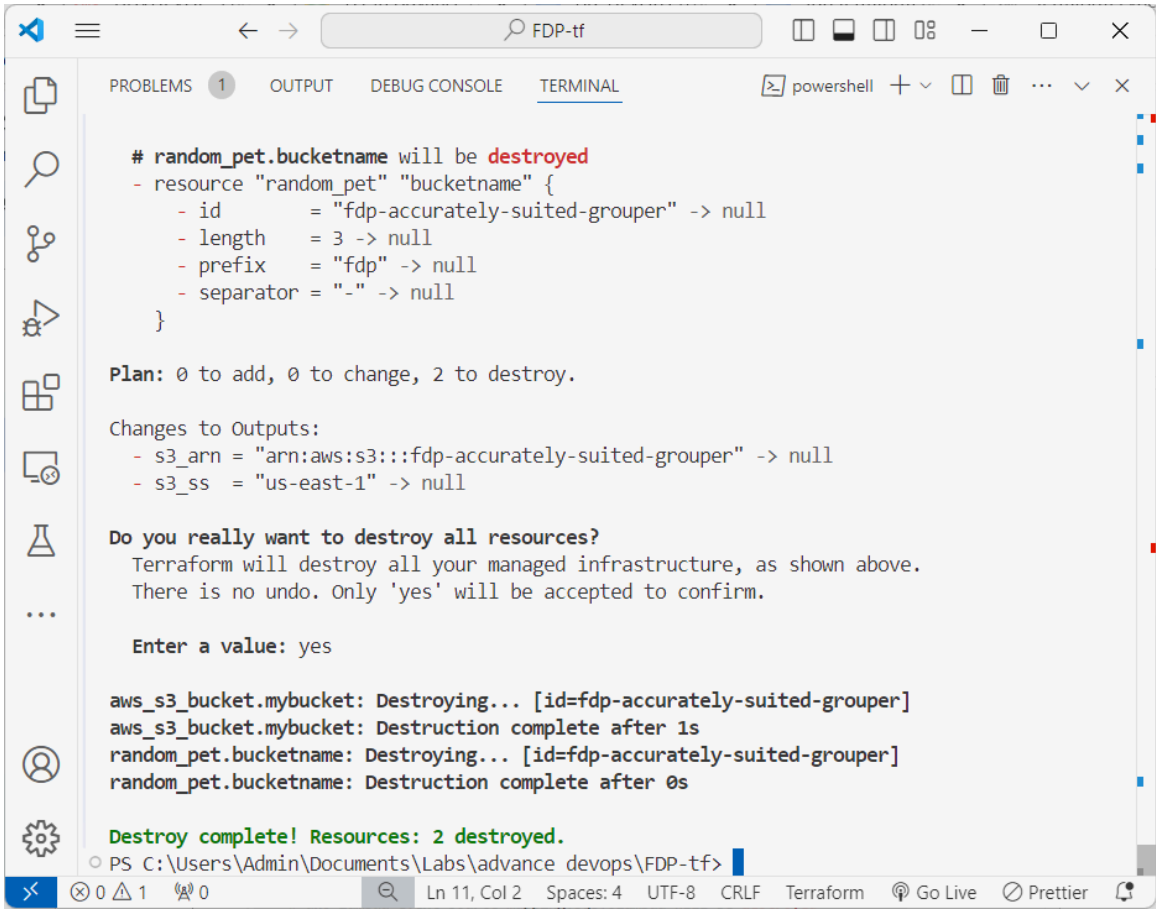






If you want to clean up the resources after testing, you can destroy them by running:
terraform destroy
(Confirm the destruction by typing yes.)


FDP-tf
PROBLEMS 1
OUTPUT
DEBUG CONSOLE
TERMINAL
powershell
PS C:\Users\Admin\Documents\Labs\advance devops\FDP-tf> terraform destroy
data.archive_file.zip: Reading...
random_pet.bucketname: Refreshing state... [id=fdp-accurately-suited-grouper]
data.archive_file.zip: Read complete after 0s [id=9381c9e67043c1d65d39f3fb36e499e0de16f9de]
aws_sqs_queue.myqueue: Refreshing state... [id=https://sqs.us-east-1.amazonaws.com/916385512917/mySQSqueue]
aws_lambda_function.mylambda: Refreshing state... [id=SqsToS3Function]
aws_s3_bucket.mybucket: Refreshing state... [id=fdp-accurately-suited-grouper]
aws_lambda_event_source_mapping.SqsToLambda: Refreshing state... [id=f85e8b75-9040-4ac2-a7f1-04be9dceefa9]
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:
# aws_lambda_event_source_mapping.SqsToLambda will be destroyed
- resource "aws_lambda_event_source_mapping" "SqsToLambda" {
- arn = "arn:aws:lambda:us-east-1:916385512917:event-source-mapping:f85e8b75-9040-4ac2-a7f1-04be9dceefa9" -> null
- batch_size = 1 -> null
- bisect_batch_on_function_error = false -> null
- enabled = true -> null
- event_source_arn = "arn:aws:sqs:us-east-1:916385512917:mySQSqueue" -> null
- function_arn = "arn:aws:lambda:us-east-1:916385512917:function:SqsToS3Function" -> null
Ln 11, Col 2 Spaces: 4 UTF-8 CRLF Terraform Go Live Prettier


FDP-tf
PROBLEMS 1
OUTPUT
DEBUG CONSOLE
TERMINAL
powershell
# random_pet.bucketname will be destroyed
- resource "random_pet" "bucketname" {
- id = "fdp-accurately-suited-grouper" -> null
- length = 3 -> null
- prefix = "fdp" -> null
- separator = "-" -> null
}
Plan: 0 to add, 0 to change, 2 to destroy.
Changes to Outputs:
- s3_arn = "arn:aws:s3:::fdp-accurately-suited-grouper" -> null
- s3_ss = "us-east-1" -> null
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_s3_bucket.mybucket: Destroying... [id=fdp-accurately-suited-grouper]
aws_s3_bucket.mybucket: Destruction complete after 1s
random_pet.bucketname: Destroying... [id=fdp-accurately-suited-grouper]
random_pet.bucketname: Destruction complete after 0s
Destroy complete! Resources: 2 destroyed.
PS C:\Users\Admin\Documents\Labs\advance devops\FDP-tf>
Ln 11, Col 2 Spaces: 4 UTF-8 CRLF Terraform Go Live Prettier

Conclusion

Deploying AWS infrastructure with Terraform and integrating S3, SQS, and Lambda creates robust and scalable cloud applications, essential for modern development.