



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.	43
Name	Harsh Pramod Padyal
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.

```
ALLv2EN-... > Modules > AWS Acad...
> Launch AWS Academy Learner Lab

AWS Used $0.3 of $50 03:54 ▶ Start Lab ■ End Lab ⓘ AWS Details ⓘ Readme ↺ Reset ✕
```

```
eee_w_3428291@runweb140728:~$ export AWS_ACCESS_KEY_ID="ASIA5GXQVGHK2M5VBZ75"
eee_w_3428291@runweb140728:~$ export AWS_SECRET_ACCESS_KEY="Q+ywIYDw0ysGbmxyAfzRhdH1anmHIjBwTUMwM"
eee_w_3428291@runweb140728:~$ export AWS_SESSION_TOKEN="IQ0b3pJ2zLxU2vJEMj////////wEaCkvZLXd3c3QtHlJHMEUCIDE+rHA
2+MrEv72yQUhWmS6L0d31TD1L1t+035VcWNAIEApoArbKJnaGic31YuDn0bJezFMHc84k+jw+QrRVstIQtgtIIMRABgwm5MTyZ0DU1MTISMTcIDK
10eZ11nezYQhKcXSqTAt8nQmwl0RpxHkKdQKf5YJ02Ncma6JaClVxD+Lasr+SumSRtK4j4+1bwj22jd4J/YX41kowiCjXtEKEjNRkpZjropv4HFbue7
TN21Pr9y2p0BS5e1ekhDQj18tH511C7r9Fxo0mCBPRiFcxY3zjz6k60pRyM+j4aAobtdV08PUu+CGc6UW/K3/yzHeB+ZYv8ZRQ/2oYmkkYHlC1
gwgNu46BdwKtAK9P5PCFUSzaoR1Jd0d3b1DapFrk9AMQFQ8kHx+R1HemJ1HwKE91j8a9J4cKvHIDR3EmdCFQV1833s0i3k0ktsCQCDPa312y1A9M
TSb01vY1K19p+QbTH1X2m1B5m3e4caRR0YmTvUkM0LnxLgG0p08+0057HmIV9JR1asc5j6oFZpq7q484wZniMmp483Hxflndp2WBU001cu7XDA64
h0P6NbYwY93JzhKKEN14D8EpumRf3gGD80/Ne5gDj39bV2YG0rRDeONFco31UYSF9pnXmMb71oWZwJ3zt/E+1TXD31bpmVH54aAwWIpj6ZFWkbCw
x/BNVzfWfa2j+btZjkKpfeR5+EV5kt1IA==
eee_w_3428291@runweb140728:~$ █
```

The screenshot shows the AWS IAM console 'Roles' page. At the top, there's a navigation bar with the AWS logo, 'Services', a search bar, and a user profile. The main header shows 'Roles (21) Info' with a refresh button, a 'Delete' button, and a 'Create role' button. Below this is a descriptive paragraph about IAM roles. A search bar is present. The table below has three columns: 'Role name', 'Trusted entities', and 'Last activity'. The first row shows a role named 'LabRole' with a trusted entity 'Account: 916385512917' and a last activity of '12 hours ago'. The bottom of the image shows a footer with 'CloudShell', 'Feedback', and copyright information for Amazon Web Services.

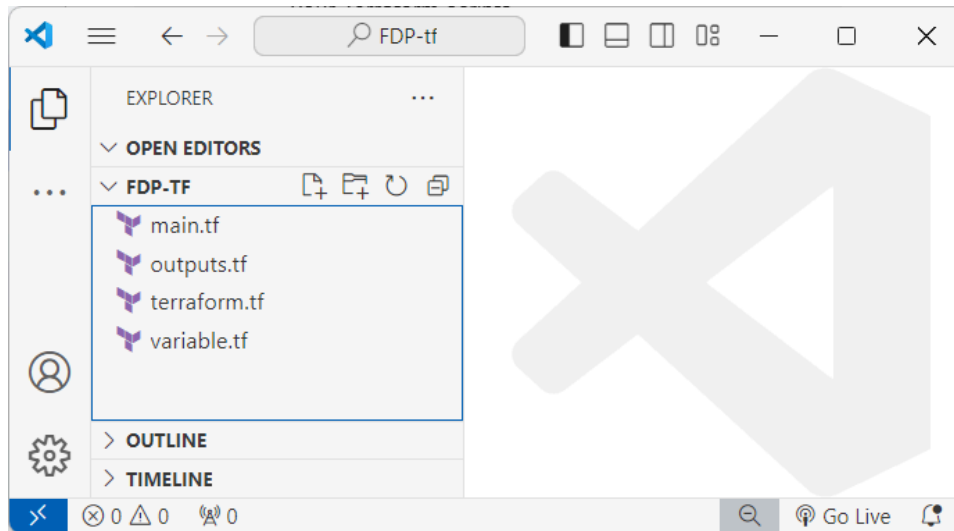
The screenshot shows the AWS IAM console interface for a role named 'LabRole'. The breadcrumb navigation at the top indicates the path: IAM > Roles > LabRole. The role's summary is displayed in a table-like format with the following details:

- Creation date:** August 07, 2024, 09:06 (UTC+05:30)
- Last activity:** 12 hours ago (indicated by a green checkmark icon)
- Maximum session duration:** 1 hour
- ARN:** arn:aws:iam::916385512917:role/LabRole (indicated by a green checkmark icon and a tooltip saying 'ARN copied')
- Link to switch roles in console:** https://signin.aws.amazon.com/switchrole?roleName=LabRole&account=916385512917
- Instance profile ARN:** arn:aws:iam::916385512917:instance-profile/LabInstanceProfile

At the top right of the role details section, there are two buttons: 'Delete' and 'Edit'.

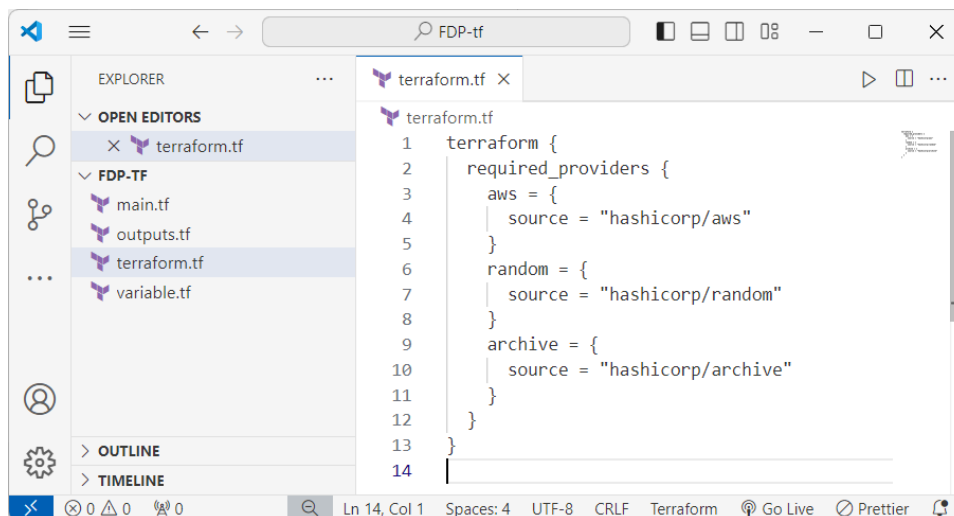
Page | 3

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



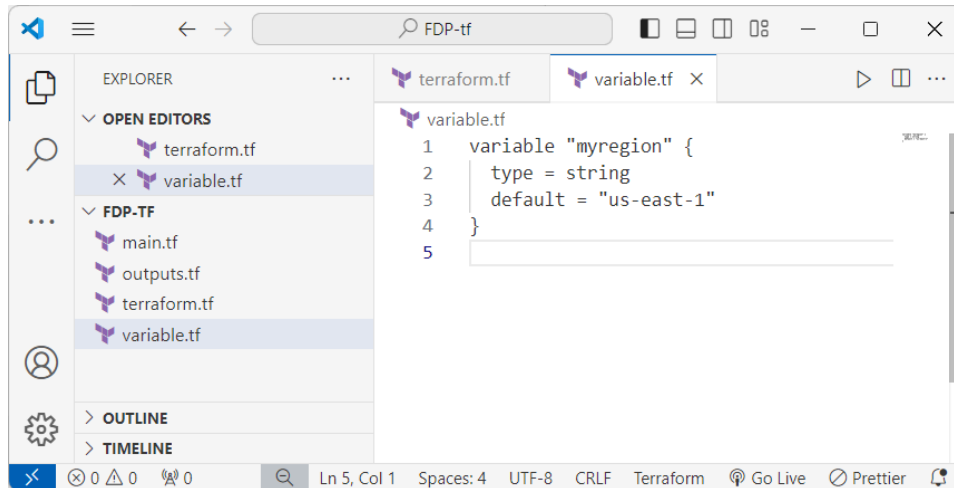
terraform.tf

```
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"  
}
```

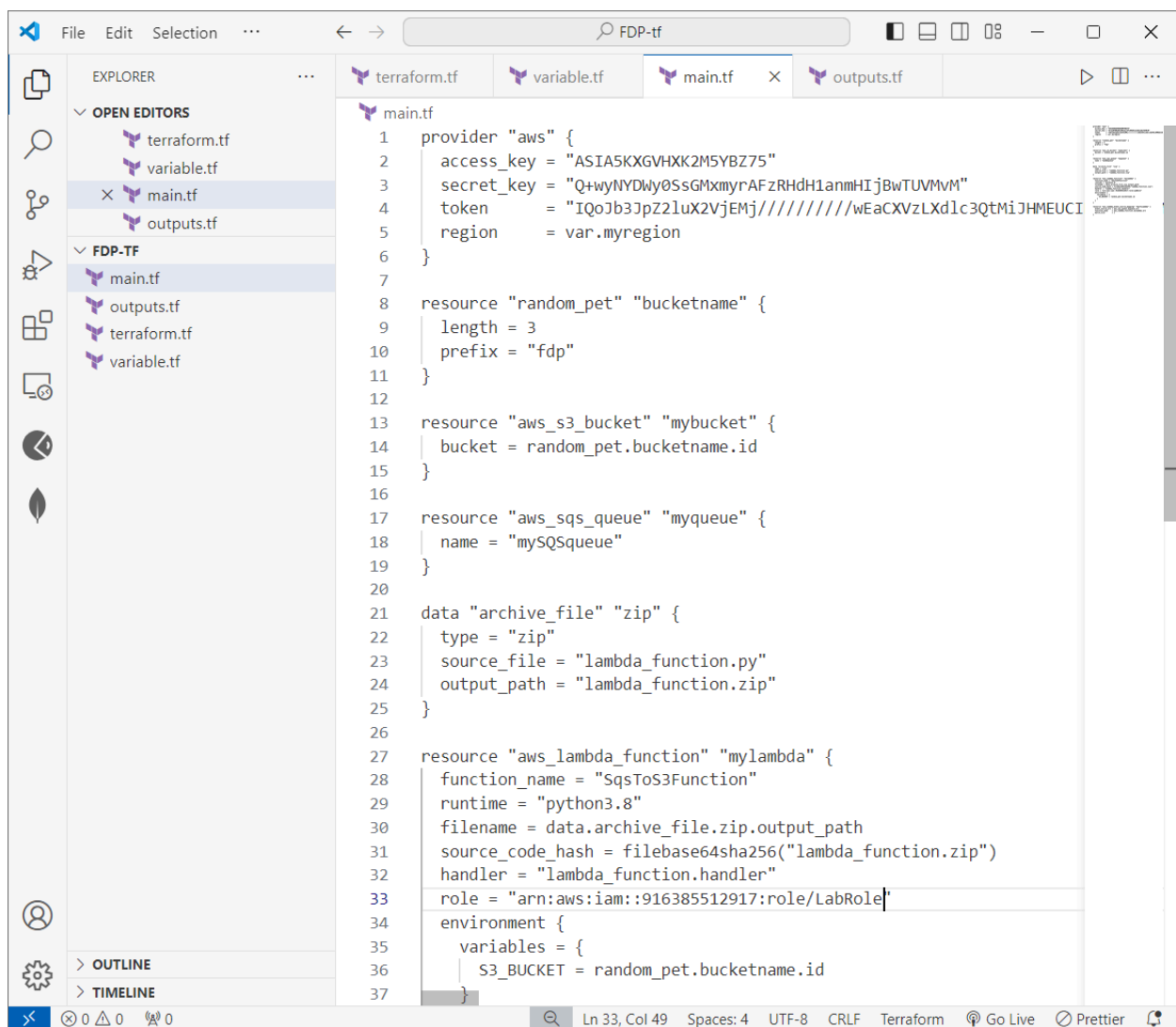


main.tf

```
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
}
```



The screenshot shows a code editor with the following components:

- EXPLORER:** Lists files under 'OPEN EDITORS' (terraform.tf, variable.tf, main.tf, outputs.tf) and 'FDP-TF' (main.tf, outputs.tf, terraform.tf, variable.tf). It also has sections for 'OUTLINE' and 'TIMELINE'.
- main.tf:** The active file, showing Terraform configuration for an AWS provider, random_pet resource, aws_s3_bucket, aws_sqs_queue, data archive_file, and aws_lambda_function.
- variable.tf:** A file with a single variable definition: `variable "myregion" { type = string }`.
- outputs.tf:** A file with a single output definition: `output "myregion" { value = var.myregion }`.
- terraform.tf:** A file with a single provider definition: `provider "aws" { access_key = "ASIA5KXGVHXX2M5YBZ75" secret_key = "Q+wyNYDWy0SsGMxmyrAFzRHdH1anmHIjBwTUVmVm" token = "IQoJb3JpZ2luX2VjEMj////////wEaCXVzLXdlc3QtMiJHMEUCI" region = var.myregion }`.

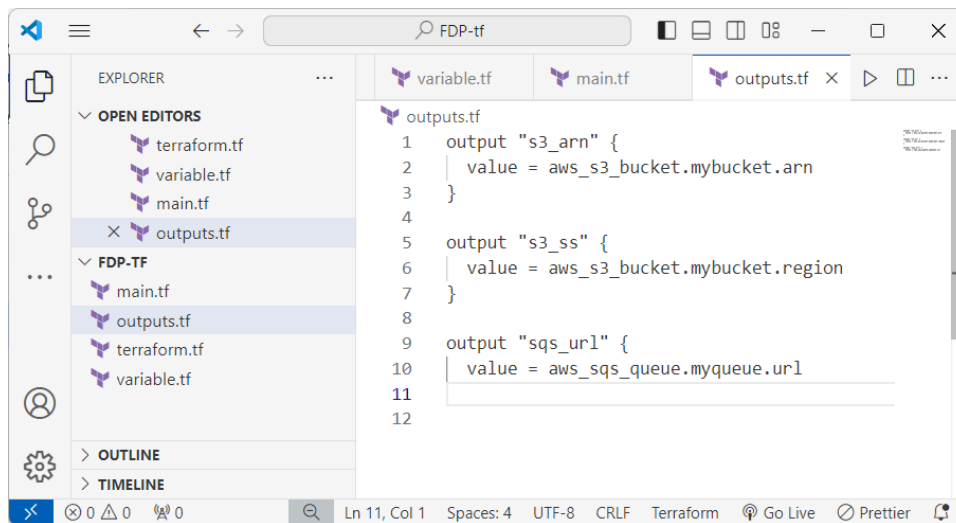
The status bar at the bottom indicates: Ln 33, Col 49, Spaces: 4, UTF-8, CRLF, Terraform, Go Live, Prettier.

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
}
```



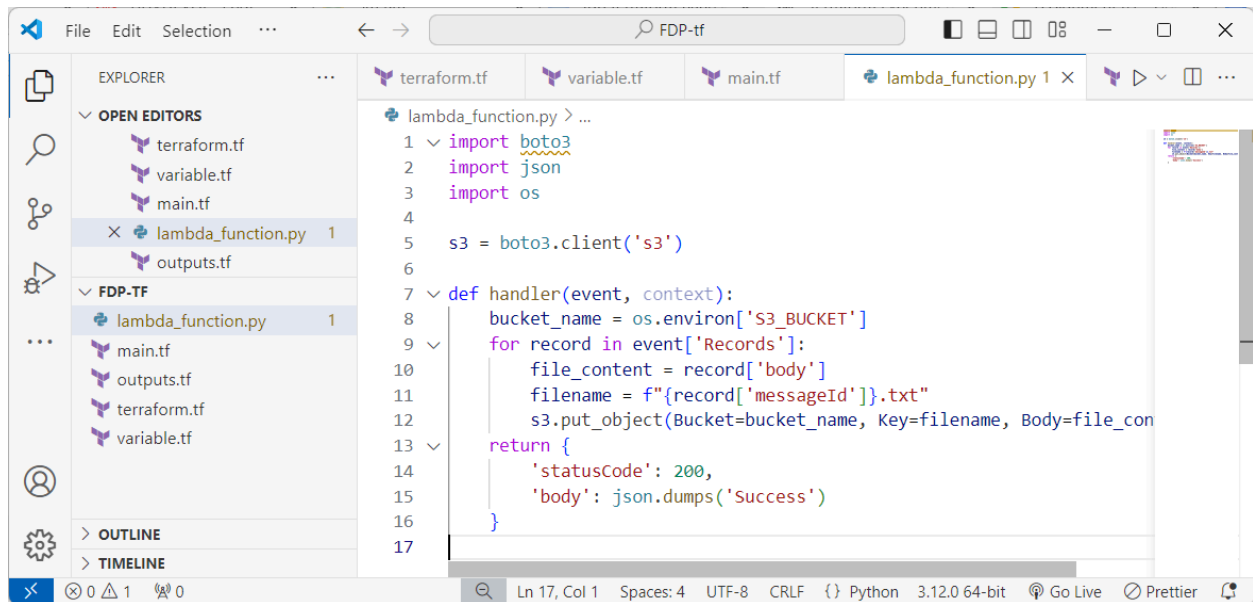
Create Lambda Python File:

- 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')
    }
```

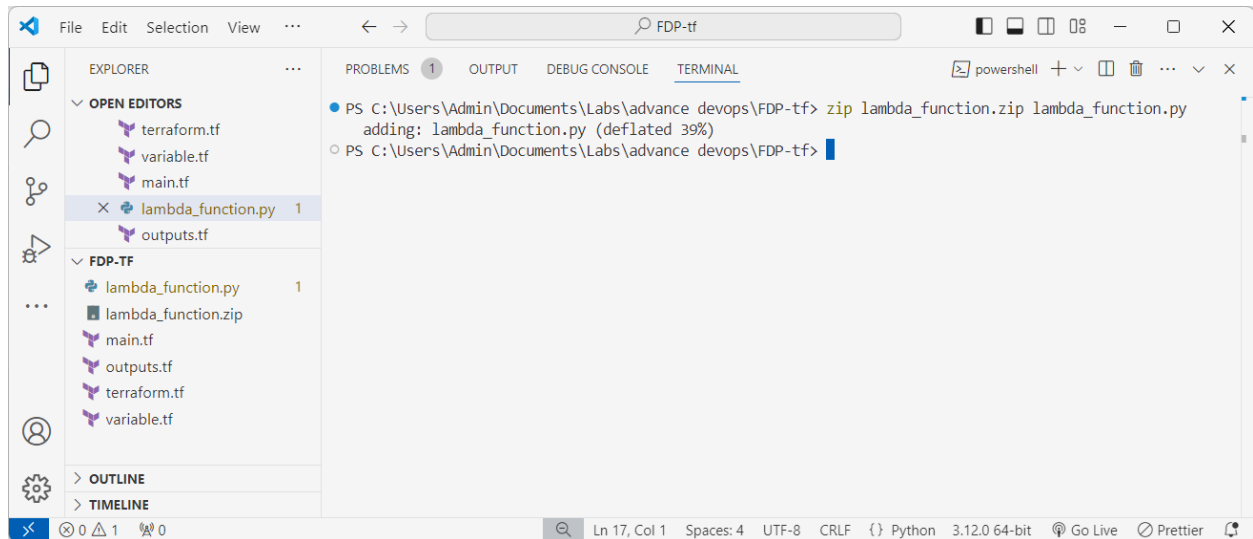


The screenshot shows the Visual Studio Code editor with the file `lambda_function.py` open. The Explorer sidebar on the left shows the project structure with files `terraform.tf`, `variable.tf`, `main.tf`, `lambda_function.py`, and `outputs.tf`. The main editor area displays the following Python code:

```
1 import boto3
2 import json
3 import os
4
5 s3 = boto3.client('s3')
6
7 def handler(event, context):
8     bucket_name = os.environ['S3_BUCKET']
9     for record in event['Records']:
10         file_content = record['body']
11         filename = f"{record['messageId']}.txt"
12         s3.put_object(Bucket=bucket_name, Key=filename, Body=file_content)
13     return {
14         'statusCode': 200,
15         'body': json.dumps('Success')
16     }
```

Open the terminal or command prompt in the same directory and run:

zip lambda_function.zip lambda_function.py

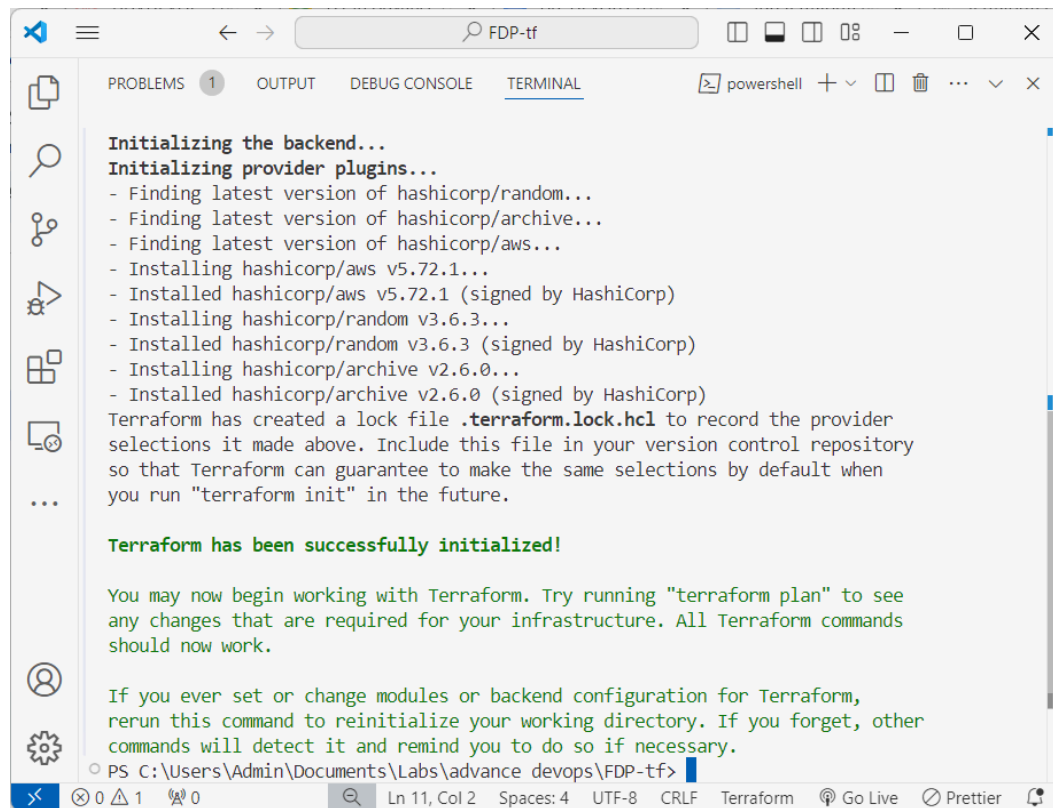


The screenshot shows the Visual Studio Code editor with the `TERMINAL` tab active. The terminal output shows the execution of the command `zip lambda_function.zip lambda_function.py` in a PowerShell session. The output is as follows:

```
PS C:\Users\Admin\Documents\Labs\advance devops\FDP-tf> zip lambda_function.zip lambda_function.py
adding: lambda_function.py (deflated 39%)
PS C:\Users\Admin\Documents\Labs\advance devops\FDP-tf>
```

To Initialize Terraform, run the command:

terraform init



```
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/random...
- Finding latest version of hashicorp/archive...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.72.1...
- Installed hashicorp/aws v5.72.1 (signed by HashiCorp)
- Installing hashicorp/random v3.6.3...
- Installed hashicorp/random v3.6.3 (signed by HashiCorp)
- Installing hashicorp/archive v2.6.0...
- Installed hashicorp/archive v2.6.0 (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.

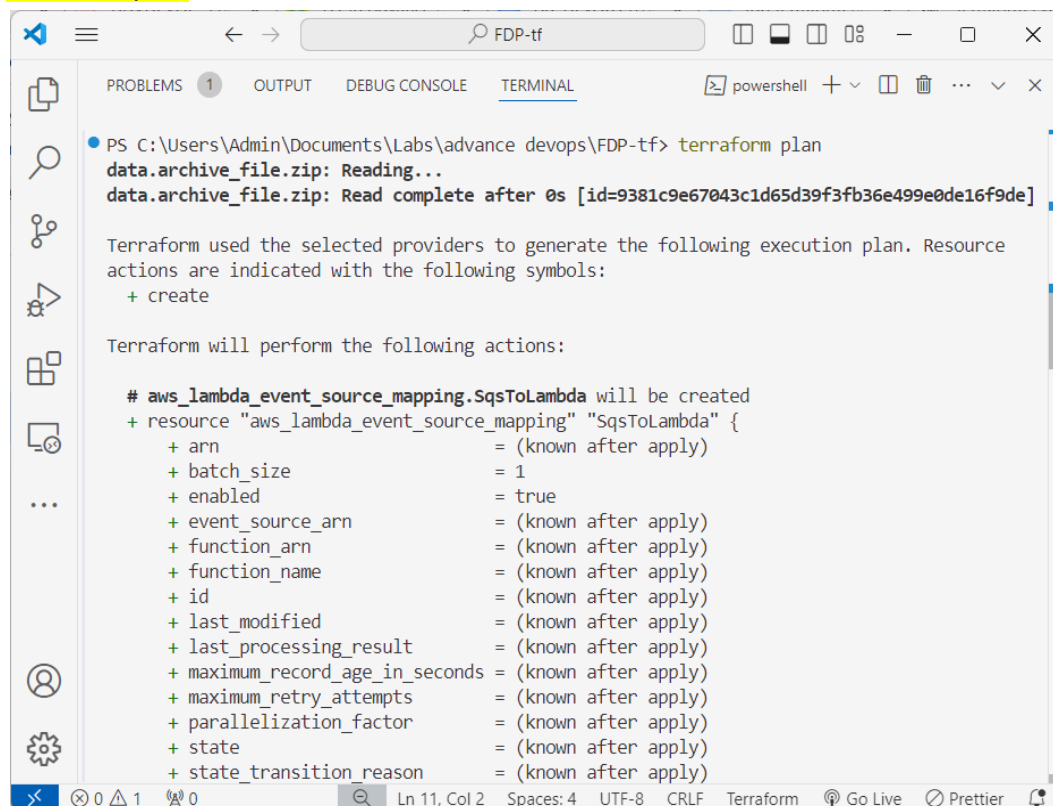
Terraform has been successfully initialized!

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.
PS C:\Users\Admin\Documents\Labs\advance devops\FDP-tf>
```

To Plan the Infrastructure, Run the command

terraform plan

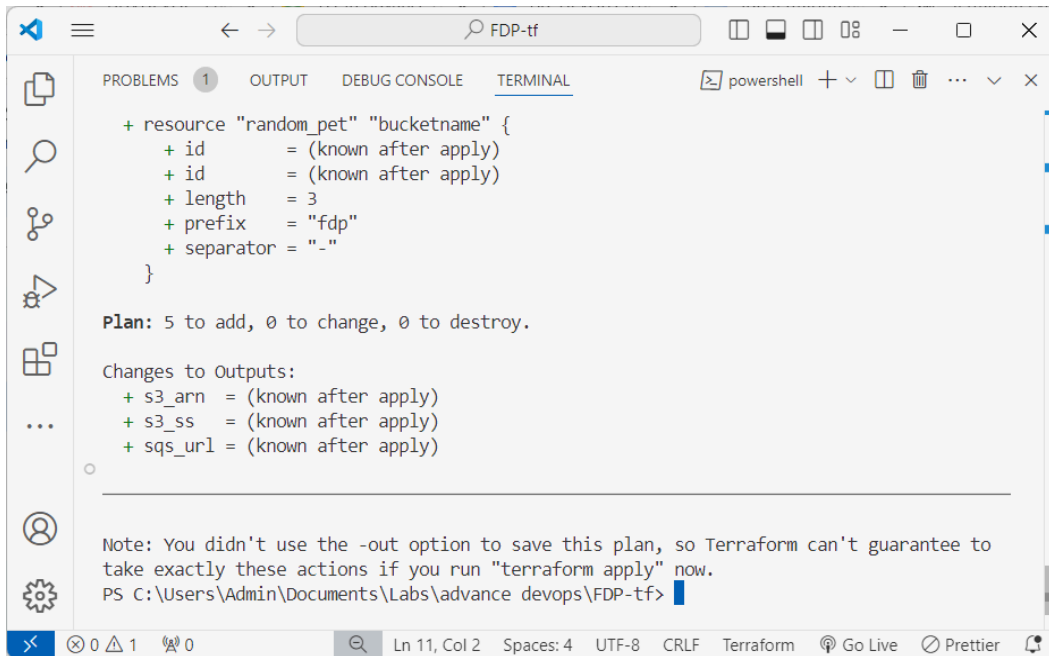


```
PS C:\Users\Admin\Documents\Labs\advance devops\FDP-tf> terraform plan
data.archive_file.zip: Reading...
data.archive_file.zip: Read complete after 0s [id=9381c9e67043c1d65d39f3fb36e499e0de16f9de]

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_lambda_event_source_mapping.SqsToLambda will be created
+ resource "aws_lambda_event_source_mapping" "SqsToLambda" {
  + arn                = (known after apply)
  + batch_size         = 1
  + enabled            = true
  + event_source_arn   = (known after apply)
  + function_arn       = (known after apply)
  + function_name      = (known after apply)
  + id                 = (known after apply)
  + last_modified      = (known after apply)
  + last_processing_result = (known after apply)
  + maximum_record_age_in_seconds = (known after apply)
  + maximum_retry_attempts = (known after apply)
  + parallelization_factor = (known after apply)
  + state              = (known after apply)
  + state_transition_reason = (known after apply)
}
```



```
PS C:\Users\Admin\Documents\Labs\advance devops\FDP-tf> terraform plan

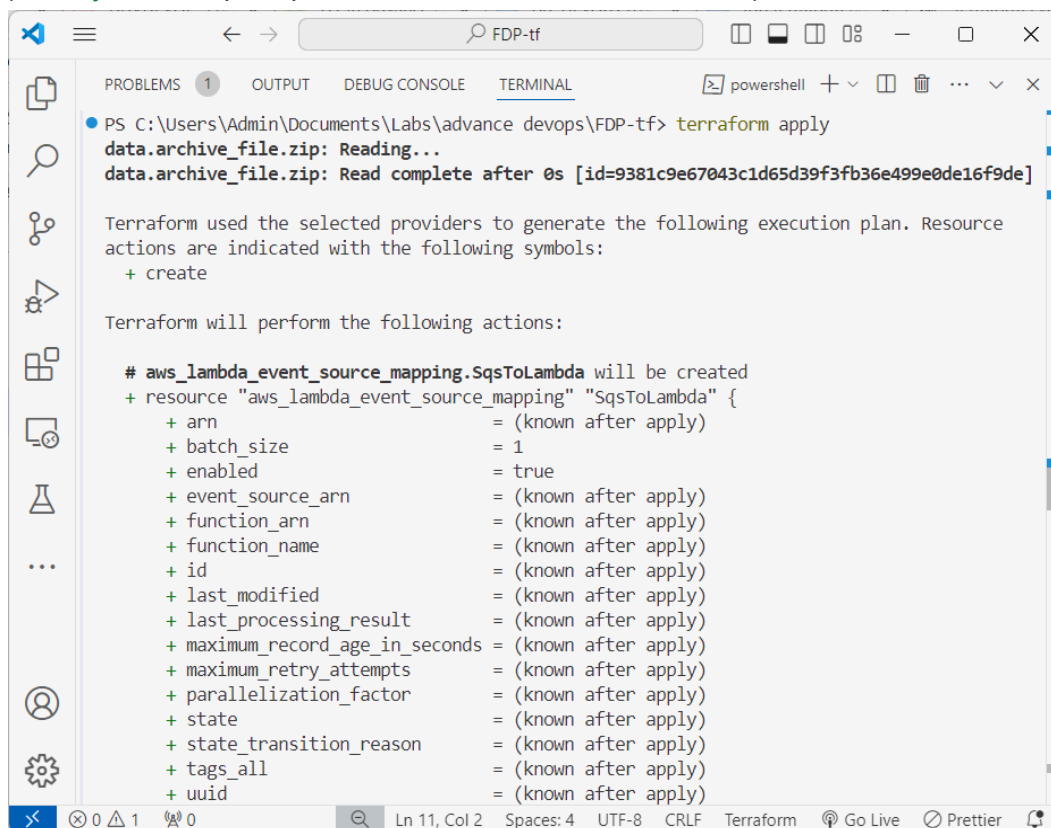
+ resource "random_pet" "bucketname" {
+   id       = (known after apply)
+   id       = (known after apply)
+   length   = 3
+   prefix   = "fdp"
+   separator = "-"
}

Plan: 5 to add, 0 to change, 0 to destroy.

Changes to Outputs:
+ s3_arn = (known after apply)
+ s3_ss  = (known after apply)
+ sqs_url = (known after apply)

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to
take exactly these actions if you run "terraform apply" now.
PS C:\Users\Admin\Documents\Labs\advance devops\FDP-tf>
```

If everything looks good, apply the plan by running **terraform apply**
(Enter **yes** when prompted. This will create the resources.)



```
PS C:\Users\Admin\Documents\Labs\advance devops\FDP-tf> terraform apply
data.archive_file.zip: Reading...
data.archive_file.zip: Read complete after 0s [id=9381c9e67043c1d65d39f3fb36e499e0de16f9de]

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_lambda_event_source_mapping.SqsToLambda will be created
+ resource "aws_lambda_event_source_mapping" "SqsToLambda" {
+   arn              = (known after apply)
+   batch_size       = 1
+   enabled          = true
+   event_source_arn = (known after apply)
+   function_arn     = (known after apply)
+   function_name    = (known after apply)
+   id               = (known after apply)
+   last_modified    = (known after apply)
+   last_processing_result = (known after apply)
+   maximum_record_age_in_seconds = (known after apply)
+   maximum_retry_attempts = (known after apply)
+   parallelization_factor = (known after apply)
+   state            = (known after apply)
+   state_transition_reason = (known after apply)
+   tags_all         = (known after apply)
+   uuid             = (known after apply)
}
```

```

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

random_pet.bucketname: Creating...
random_pet.bucketname: Creation complete after 0s [id=fdp-accurately-suited-grouper]
aws_sqs_queue.myqueue: Creating...
aws_lambda_function.mylambda: Creating...
aws_s3_bucket.mybucket: Creating...
aws_s3_bucket.mybucket: Creation complete after 4s [id=fdp-accurately-suited-grouper]
aws_lambda_function.mylambda: Creation complete after 7s [id=SqsToS3Function]
aws_sqs_queue.myqueue: Still creating... [10s elapsed]
aws_sqs_queue.myqueue: Still creating... [20s elapsed]
aws_sqs_queue.myqueue: Creation complete after 27s [id=https://sqs.us-east-1.amazonaws.com/916385512917/mySQSqueue]
aws_lambda_event_source_mapping.SqsToLambda: Creating...
aws_lambda_event_source_mapping.SqsToLambda: Still creating... [10s elapsed]
aws_lambda_event_source_mapping.SqsToLambda: Still creating... [20s elapsed]
aws_lambda_event_source_mapping.SqsToLambda: Creation complete after 26s [id=f85e8b75-9040-4ac2-a7f1-04be9dceefa9]

Apply complete! Resources: 5 added, 0 changed, 0 destroyed.

Outputs:

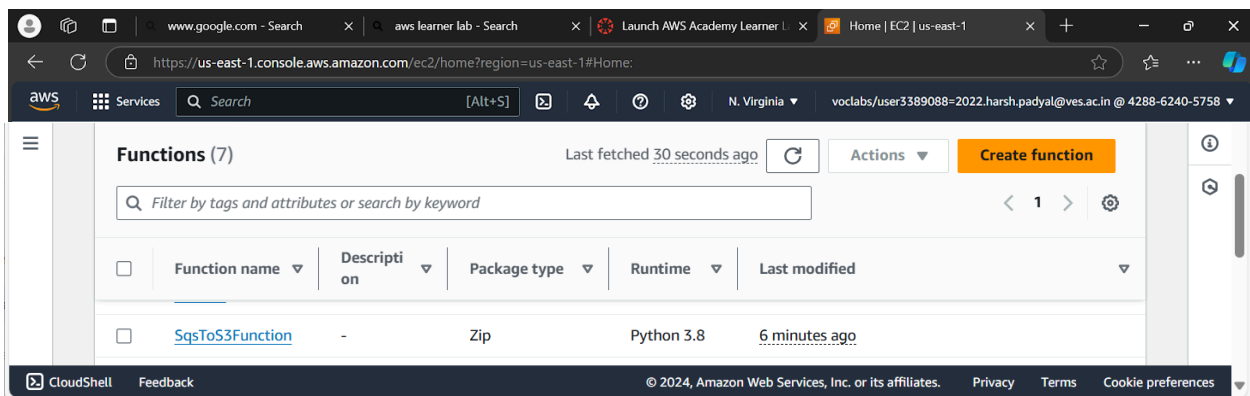
s3_arn = "arn:aws:s3:::fdp-accurately-suited-grouper"
s3_ss = "us-east-1"
sqs_url = "https://sqs.us-east-1.amazonaws.com/916385512917/mySQSqueue"
PS C:\Users\Admin\Documents\Labs\advance_devops\FDP-tf>

```

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



The screenshot shows the AWS Lambda console for the function 'SqsToS3Function'. The 'Function overview' tab is active, displaying a diagram of the function architecture. The diagram shows an SQS queue as the trigger for the SqsToS3Function. The function is currently in the 'Test' state. The 'Layers' section shows 0 layers. The 'Description' field is empty. The 'Last modified' timestamp is '10 minutes ago'. The 'Function ARN' is 'arn:aws:lambda:us-east-1:9163855129:17:function:SqsToS3Function'. The 'Function URL' is also displayed. The 'Add destination' button is visible. The 'Export to Application Composer' and 'Download' buttons are also present.

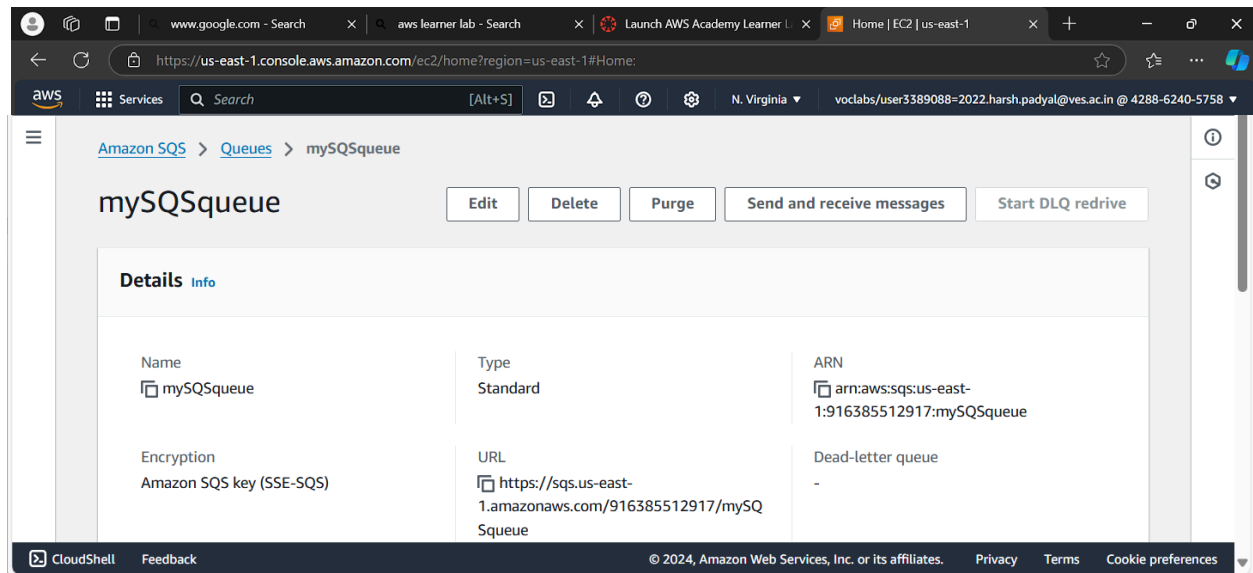
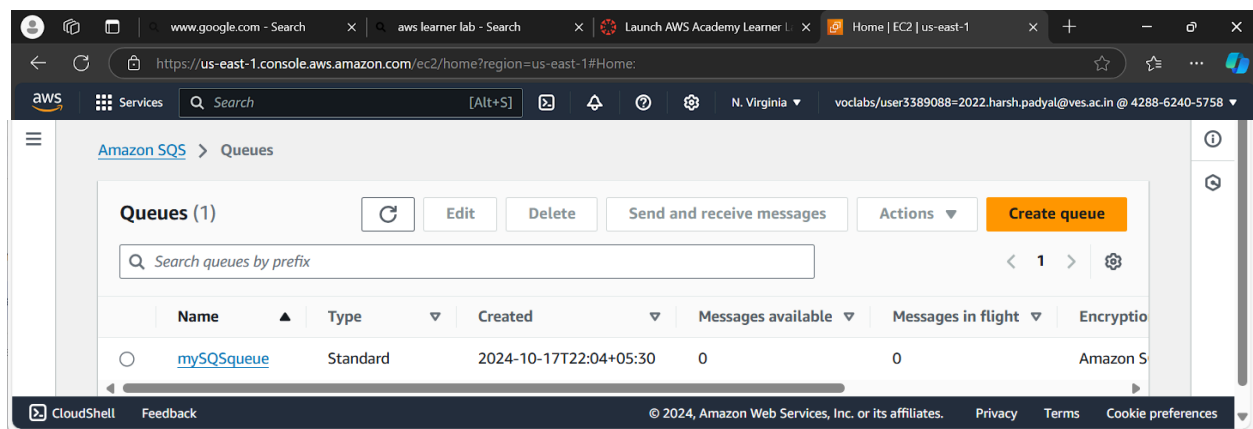
The screenshot shows the 'Code source' tab for the 'SqsToS3Function'. The code is written in Python and uses the boto3 library to interact with S3. The code is as follows:

```
1 import boto3
2 import json
3 import os
4
5 s3 = boto3.client('s3')
6
7 def handler(event, context):
8     bucket_name = os.environ['S3_BUCKET']
9     for record in event['Records']:
10         file_content = record['body']
11         filename = f"{record['messageId']}.txt"
12         s3.put_object(Bucket=bucket_name, Key=filename, Body=file_content)
13     return {
14         'statusCode': 200,
15         'body': json.dumps('Success')
16     }
17
```

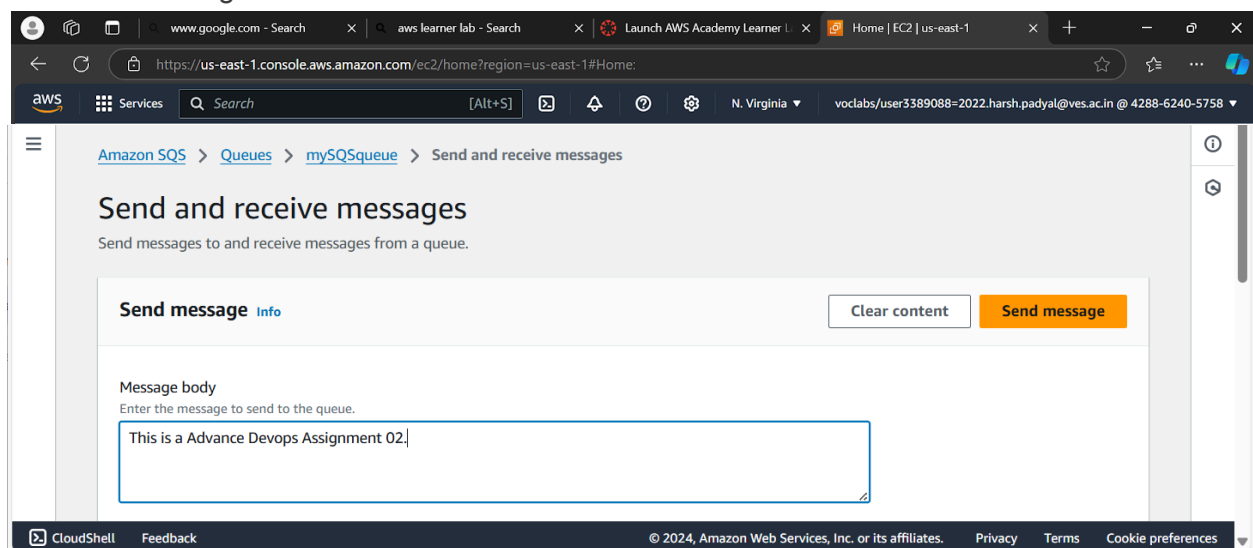
The screenshot shows the AWS Lambda console for the 'SqsToS3Function'. The 'Configuration' tab is selected, and the 'Triggers' section is active. A single trigger is listed, named 'mySQSqueue', which is an SQS queue. The trigger is in an 'Enabled' state. The function's ARN is 'arn:aws:lambda:us-east-1:9163855129:function:SqsToS3Function'. The function was last modified '11 minutes ago'. The console also shows a sidebar with 'General configuration', 'Triggers', 'Permissions', 'Destinations', 'Function URL', and 'Environment'. The bottom of the console displays 'CloudShell' and 'Feedback' buttons, along with the copyright notice '© 2024, Amazon Web Services, Inc. or its affiliates.' and links for 'Privacy', 'Terms', and 'Cookie preferences'.

SQS queue

This screenshot shows the same AWS Lambda console view as the previous one, but with the 'Environment variables' section expanded. It shows one environment variable named 'S3_BUCKET' with the value 'fdp-accurately-suited-grouper'. The function's ARN is 'arn:aws:lambda:us-east-1:9163855129:function:SqsToS3Function', and it was last modified '12 minutes ago'. The sidebar on the left now highlights 'Environment variables' under 'General configuration'. The bottom of the console shows the same footer information as the previous screenshot.



Send the message from the SQS.



The screenshot shows the AWS Management Console interface for an Amazon SQS queue. The breadcrumb navigation is 'Amazon SQS > Queues > mySQSqueue > Send and receive messages'. The main heading is 'Send and receive messages' with a subtext 'Send messages to and receive messages from a queue.' Below this, there is a 'Send message' section with a 'Clear content' button and a 'Send message' button. A green success message states: 'Your message has been sent and is ready to be received.' with a 'View details' link. The 'Message body' section contains a text input field with the value 'This is a Advance Devops Assignment 02.' The footer of the console shows '© 2024, Amazon Web Services, Inc. or its affiliates.' and links for 'Privacy', 'Terms', and 'Cookie preferences'.

S3 bucket

The screenshot shows the AWS Management Console for Amazon S3. The breadcrumb navigation is 'Amazon S3'. A banner for 'Account snapshot - updated every 24 hours' is visible. Below this, there are tabs for 'General purpose buckets' and 'Directory buckets'. The 'General purpose buckets (2)' section shows a list of buckets. A search bar 'Find buckets by name' is present. The bucket list has columns for 'Name', 'AWS Region', 'IAM Access Analyzer', and 'Creation date'. One bucket is listed: 'fdp-accurately-suited-grouper' in the 'US East (N. Virginia) us-east-1' region, with a creation date of 'October 17, 2024, 22:04:14 (UTC+05:30)'. The footer of the console shows '© 2024, Amazon Web Services, Inc. or its affiliates.' and links for 'Privacy', 'Terms', and 'Cookie preferences'.

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)

Amazon S3 > Buckets > fdp-accurately-suited-grouper

fdp-accurately-suited-grouper [Info](#)

Objects Properties Permissions Metrics Management Access Points

Objects (1) [Info](#)

[Refresh](#) [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](#)

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	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

Amazon S3 > Buckets > fdp-accurately-suited-grouper > 178a06e9-f20a-4b62-8112-f6135c26b447.txt

178a06e9-f20a-4b62-8112-f6135c26b447.txt

[Copy S3 URI](#) [Download](#) [Open](#) [Object actions](#)

Properties Permissions Versions

Object overview

Downloads

- [178a06e9-f20a-4b62-8112-f6135c26b447.txt](#)
- [ADVDEVOP_4.pdf](#)
- [da60df72-2a39-46e2-94f7-96b77dc8b889.txt](#)
- [fdp terraform handson.docx](#)
- [DOC-20240821-WA0000...docx](#)

[See more](#)

Amazon S3 > Buckets > fdp-accurately-suited-grouper > 178a06e9-f20a-4b62-8112-f6135c26b447.txt

178a06e9-f20a-4b62-8112-f6135c26b447.txt

[Copy S3 URI](#) [Download](#) [Open](#) [Object actions](#)

Properties Permissions Versions

Object overview

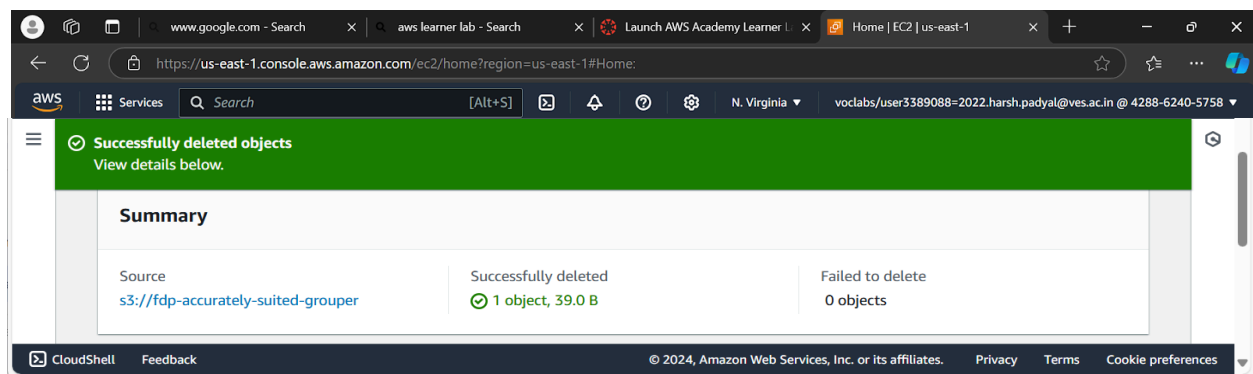
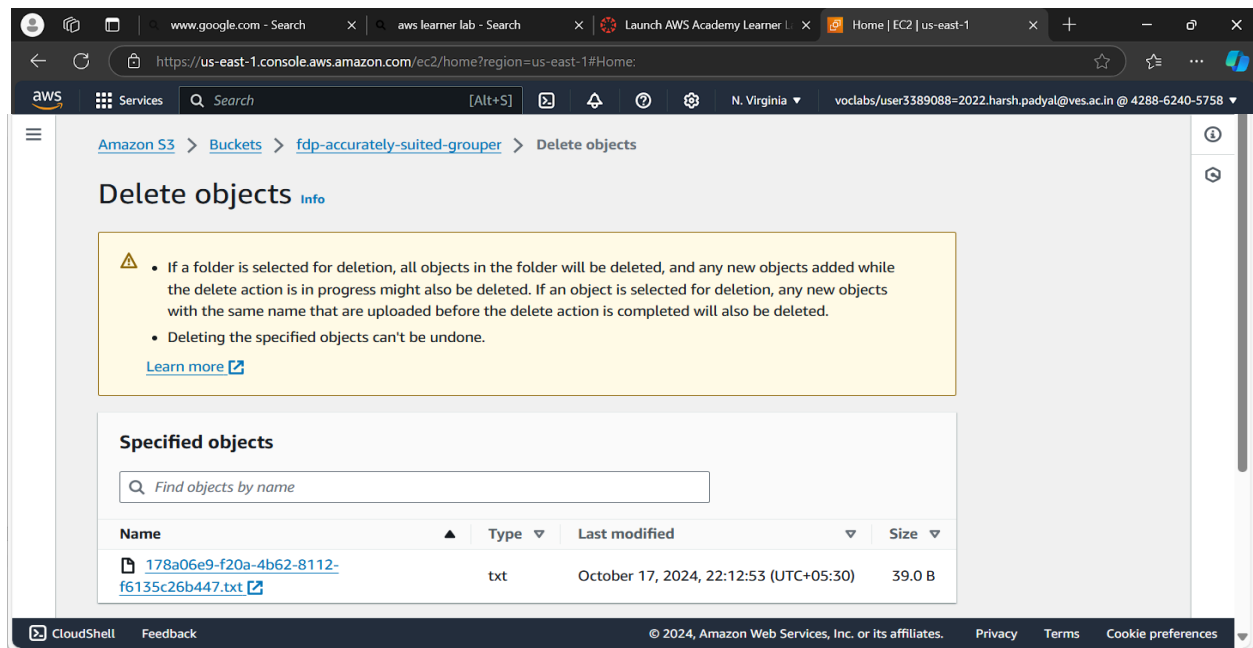
File Editor

178a06e9-f20a-4b62-8112-f61

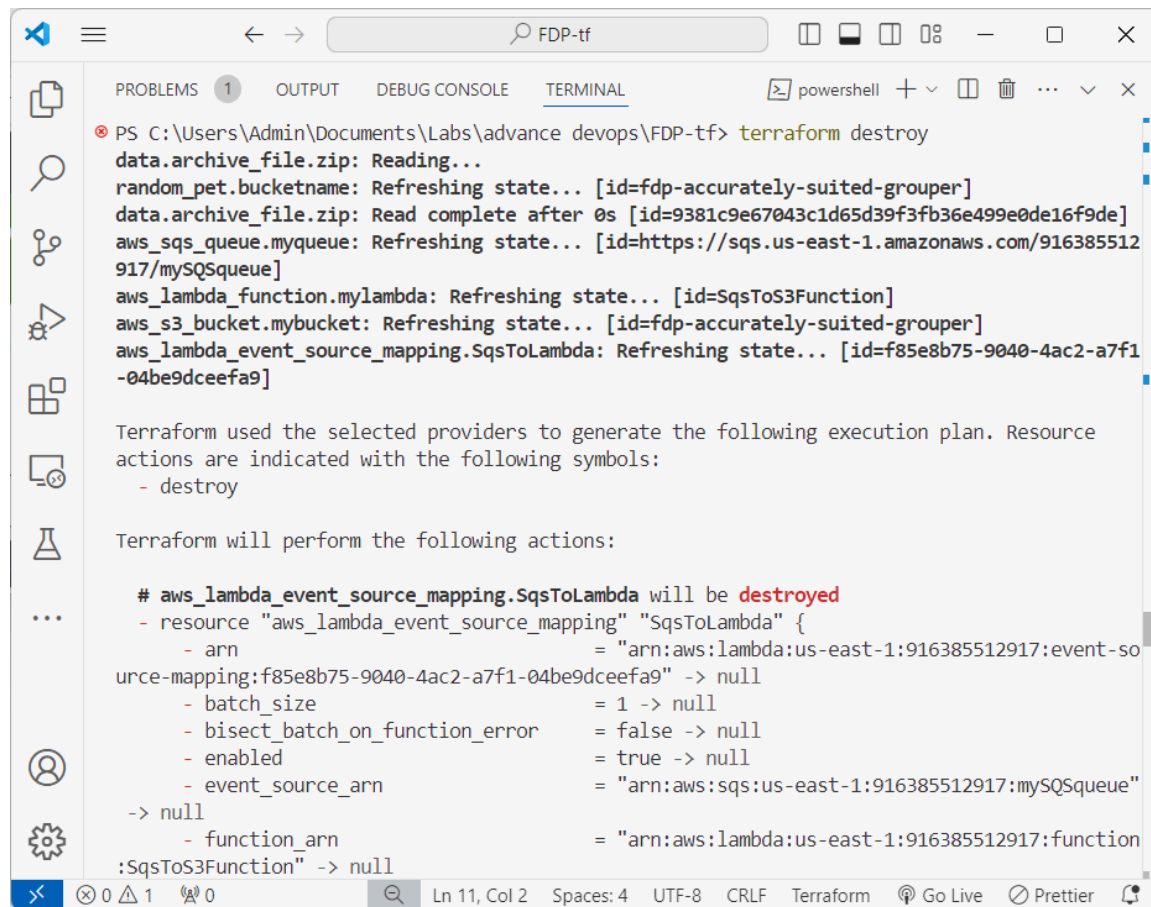
File Edit View

This is a Advance Devops Assignment 02.

Ln 1, Col 40 39 characters 100% Windows (CRLF) UTF-8



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



```

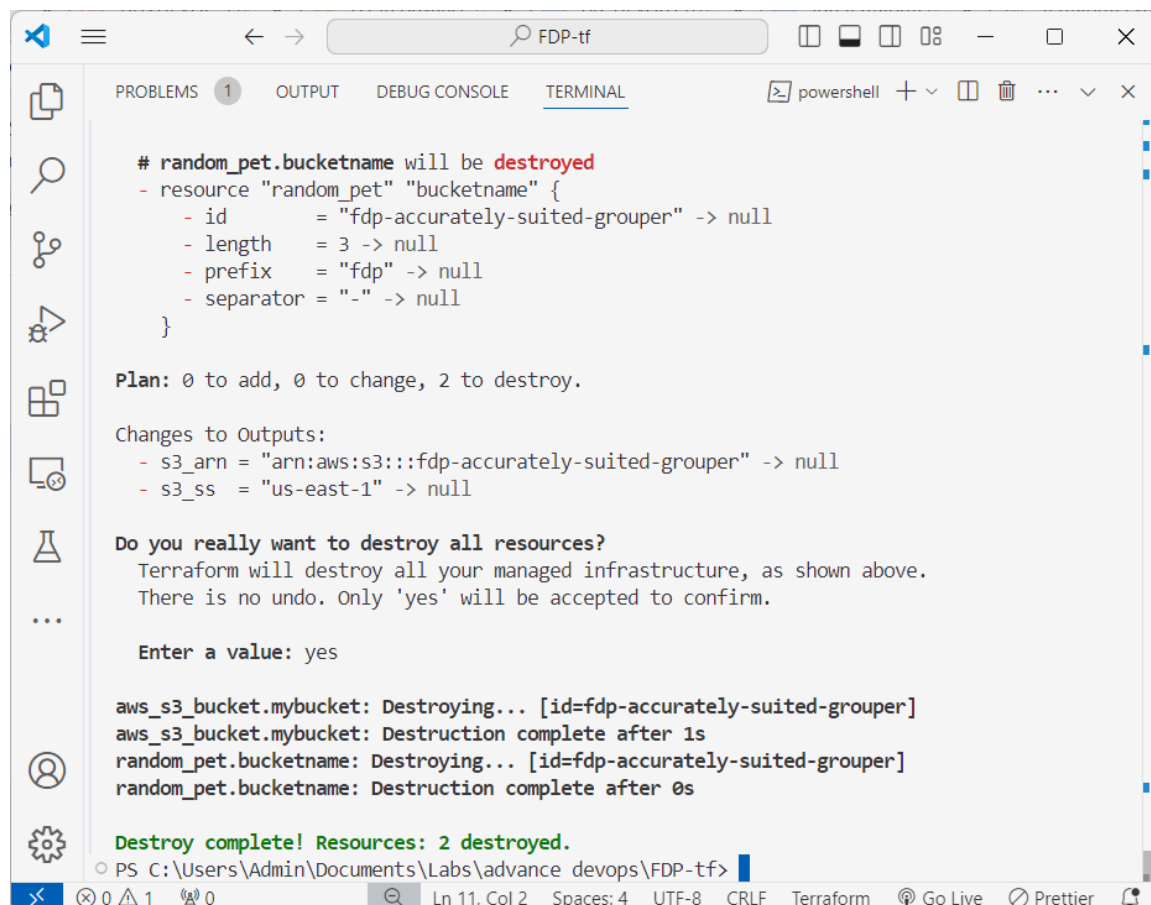
PS C:\Users\Admin\Documents\Labs\advance devops\FDP-tf> terraform destroy
data.archive_file.zip: Reading...
random_pet.bucketname: Refreshing state... [id=fdp-accurately-suited-grouper]
data.archive_file.zip: Read complete after 0s [id=9381c9e67043c1d65d39f3fb36e499e0de16f9de]
aws_sqs_queue.myqueue: 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-so
    ource-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

```



```

# 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>

```

Conclusion

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