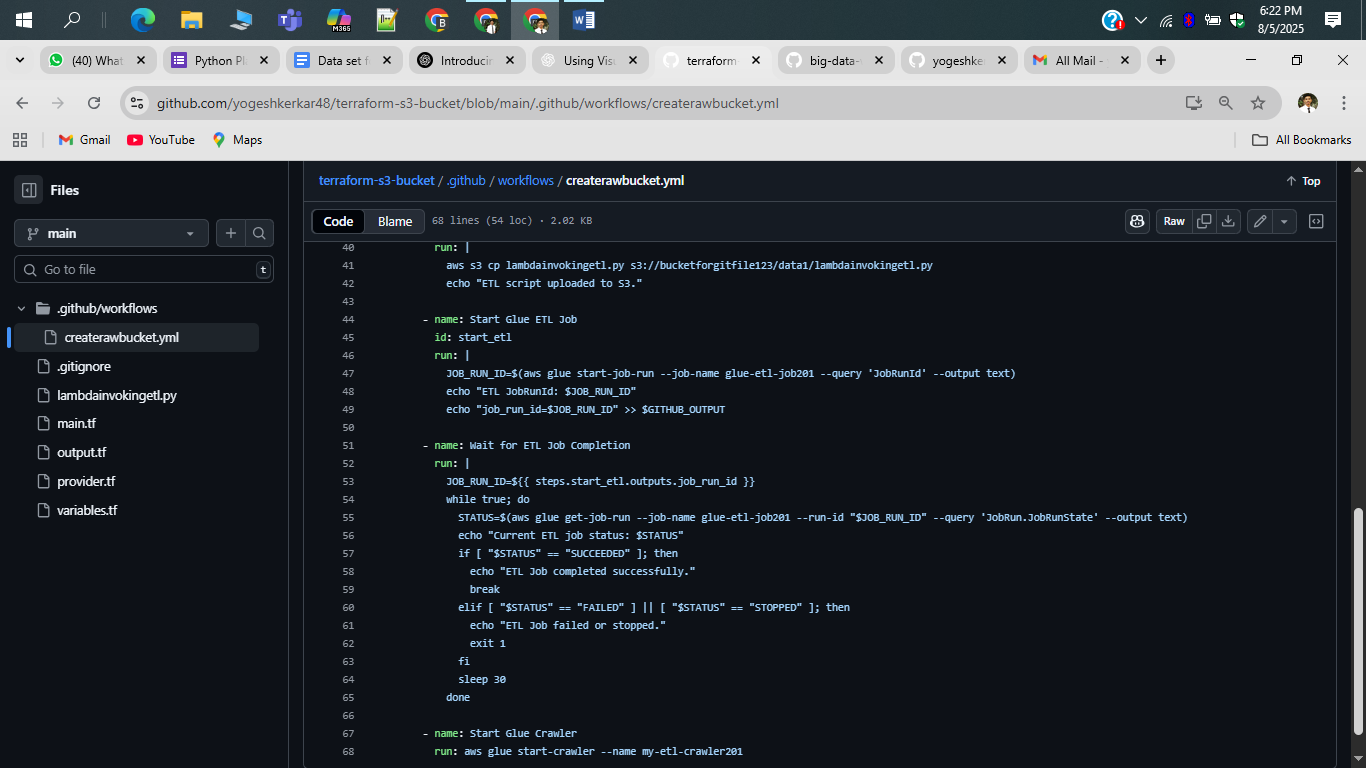
**Task Update – GitHub Repository Setup for Terraform Automation**

* Created a dedicated **GitHub repository** to manage infrastructure as code using **Terraform**.
* Added all key Terraform configuration files including:
  + main.tf: Core infrastructure definitions (S3 buckets, Glue jobs, Lambda triggers, etc.)
  + variables.tf: Parameterized variables for flexible deployment
  + output.tf: Outputs for referencing important resource values
  + provider.tf: AWS provider configuration
  + lambdainvokingetl.py: Python script to automate Lambda-based ETL pipeline trigger
* Included .gitignore and structured .github directory for workflow automation and CI/CD if needed.
* AWS Glue Workflow is a **visual orchestration tool** in AWS Glue that helps you **coordinate ETL activities like crawlers, jobs, and triggers**. It enables you to define a sequence or a **pipeline of tasks** that should run in a specific order, possibly based on success/failure conditions.

**Key Learnings:**

* Learned how to organize and modularize Terraform code for maintainability and reuse.
* Understood the integration of AWS services (S3, Lambda, Glue) via Terraform.
* Improved confidence in using **Git and GitHub** for version control and collaboration.



***Script for .yml file***

name: Terraform + ETL Automation

#123456

on:

  push:

    branches:

      - main

jobs:

  infra-etl:

    runs-on: ubuntu-latest

    steps:

      - name: Checkout Code

        uses: actions/checkout@v3

      - name: Configure AWS Credentials

        uses: aws-actions/configure-aws-credentials@v2

        with:

          aws-access-key-id: ${{ secrets.AWS\_ACCESS\_KEY\_ID }}

          aws-secret-access-key: ${{ secrets.AWS\_SECRET\_ACCESS\_KEY }}

          aws-session-token: ${{ secrets.AWS\_SESSION\_TOKEN }}

          aws-region: ${{ secrets.REGION }}

      - name: Setup Terraform

        uses: hashicorp/setup-terraform@v3

      - name: Terraform Init

        run: terraform init

      - name: Terraform Validate

        run: terraform validate

      - name: Terraform Apply

        run: terraform apply -auto-approve

      - name: Upload ETL Script to S3

        run: |

          aws s3 cp lambdainvokingetl.py s3://bucketforgitfile123/data1/lambdainvokingetl.py

          echo "ETL script uploaded to S3."

      - name: Start Glue ETL Job

        id: start\_etl

        run: |

          JOB\_RUN\_ID=$(aws glue start-job-run --job-name glue-etl-job201 --query 'JobRunId' --output text)

          echo "ETL JobRunId: $JOB\_RUN\_ID"

          echo "job\_run\_id=$JOB\_RUN\_ID" >> $GITHUB\_OUTPUT

      - name: Wait for ETL Job Completion

        run: |

          JOB\_RUN\_ID=${{ steps.start\_etl.outputs.job\_run\_id }}

          while true; do

            STATUS=$(aws glue get-job-run --job-name glue-etl-job201 --run-id "$JOB\_RUN\_ID" --query 'JobRun.JobRunState' --output text)

            echo "Current ETL job status: $STATUS"

            if [ "$STATUS" == "SUCCEEDED" ]; then

              echo "ETL Job completed successfully."

              break

            elif [ "$STATUS" == "FAILED" ] || [ "$STATUS" == "STOPPED" ]; then

              echo "ETL Job failed or stopped."

              exit 1

            fi

            sleep 30

          done

      - name: Start Glue Crawler

        run: aws glue start-crawler --name my-etl-crawler201

***Script for main.tf***

resource "aws\_s3\_bucket" "etl\_bucket" {

  bucket = var.bucket\_name\_prefix

}

resource "aws\_glue\_catalog\_database" "etl\_db" {

  name = "crime\_db123"

}

locals {

  glue\_role\_arn = "arn:aws:iam::951764799690:role/LabRole"

}

resource "aws\_glue\_job" "etl\_job" {

  name     = var.glue\_job\_name

  role\_arn = local.glue\_role\_arn

  command {

    name            = "glueetl"

    script\_location = var.script\_s3\_path

    python\_version  = "3"

  }

  glue\_version      = "4.0"

  number\_of\_workers = 2

  worker\_type       = "G.1X"

}

resource "aws\_glue\_crawler" "etl\_crawler" {

  name          = var.glue\_crawler\_name

  role          = local.glue\_role\_arn

  database\_name = aws\_glue\_catalog\_database.etl\_db.name

  s3\_target {

    path = "s3://${aws\_s3\_bucket.etl\_bucket.bucket}/cleaned\_data/"

 }

  depends\_on = [aws\_glue\_job.etl\_job]

}

***ETL SCRIPT.py***

from awsglue.context import GlueContext

from pyspark.context import SparkContext

import sys

from awsglue.utils import getResolvedOptions

from pyspark.sql import SparkSession

# Get arguments passed from Lambda or Glue

#args = getResolvedOptions(sys.argv, ['S3\_Source\_file', 'S3\_Destination\_file'])

source\_path = "s3://trans123321/abcd/"

destination\_path = "s3://automationteraform2310201/cleaned\_data/"

# Create Spark and Glue Context

sc = SparkContext()

glueContext = GlueContext(sc)

spark = glueContext.spark\_session

# Read CSV from S3

df = spark.read.csv(source\_path, inferSchema=True, header=True)

# Drop unwanted columns

columns\_to\_drop = ["community\_area", "ward"]

df = df.drop(\*columns\_to\_drop)

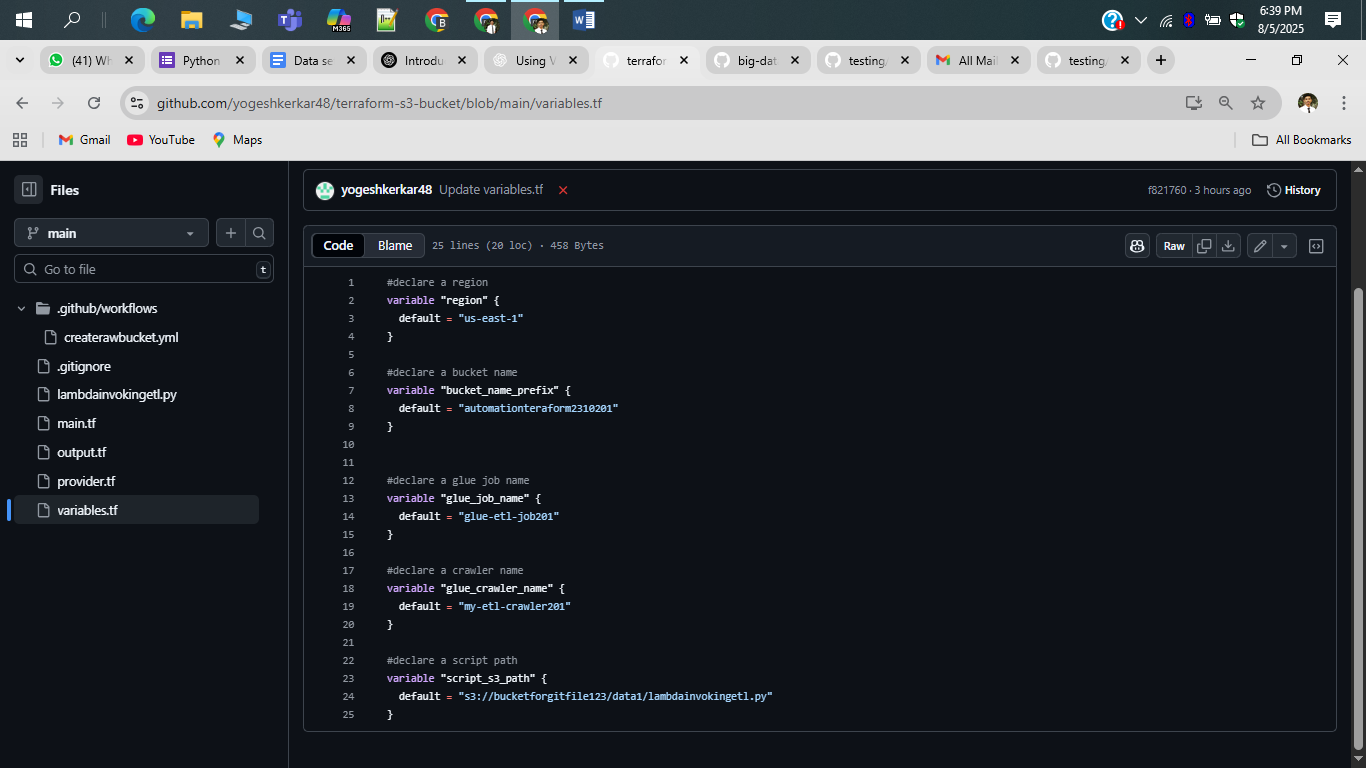
# Coalesce to a single output file

df\_single = df.coalesce(1)

# Write back to S3 in CSV format

df\_single.write.mode("overwrite").option("header", "true").csv(destination\_path)

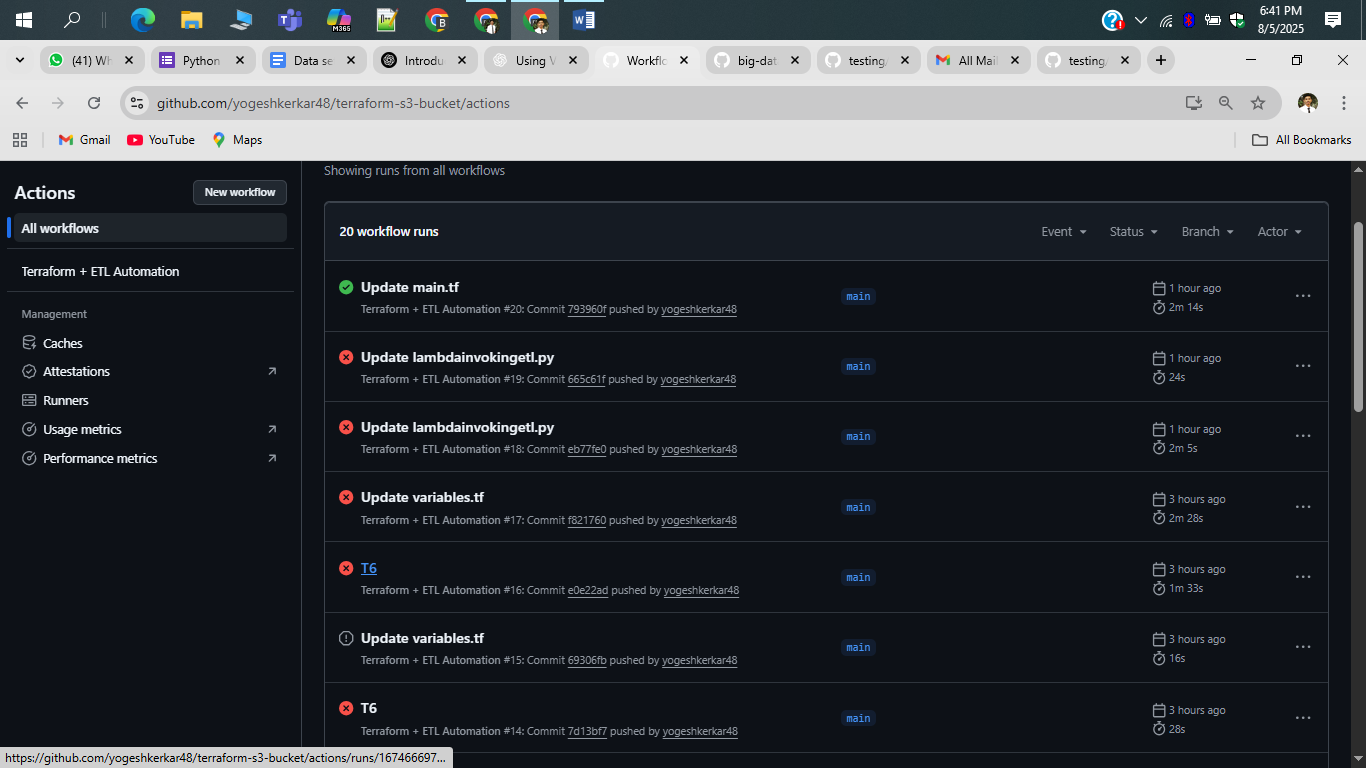
***VARIABLES.TF***



After **20+ iterations and debugging attempts**, the entire automation pipeline is now working as expected.

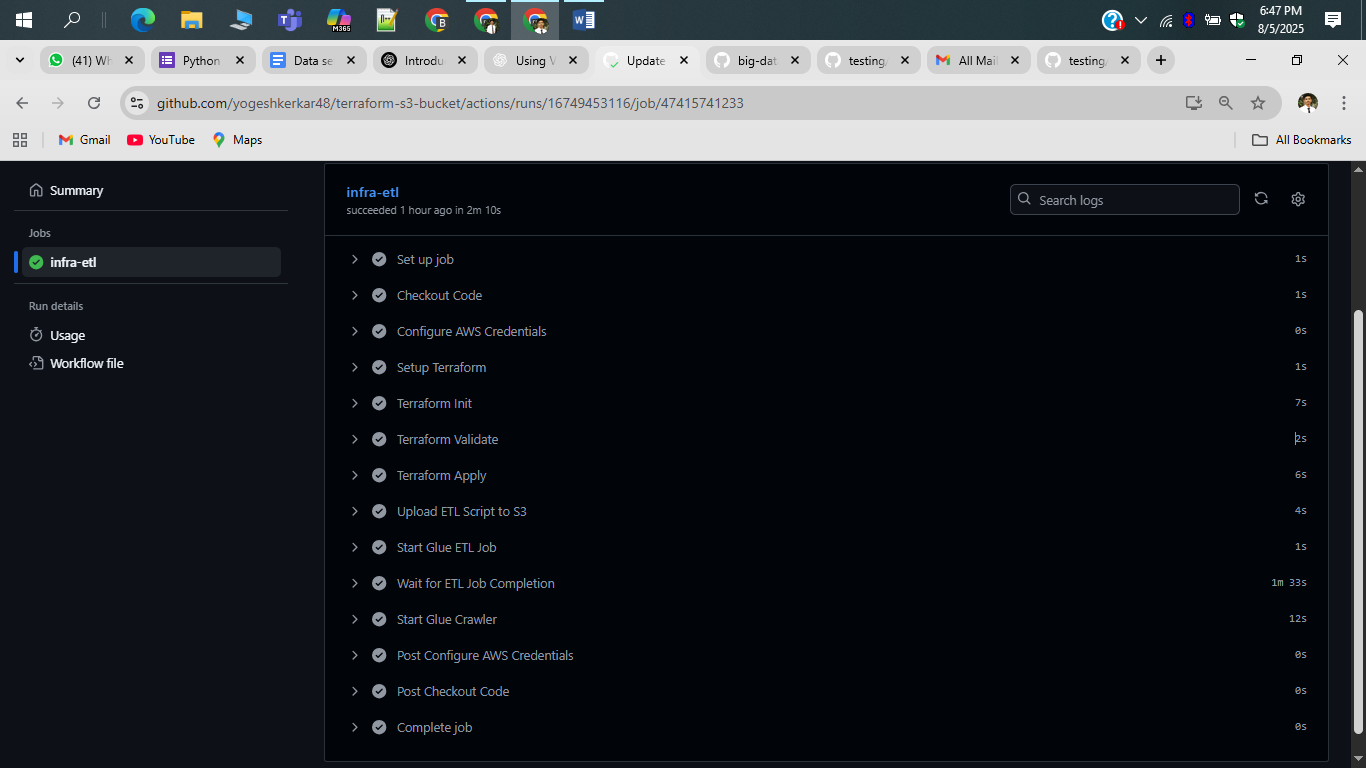
All components — including **Terraform provisioning**, **ETL workflow**, and **S3 integration** — have been successfully validated end-to-end.

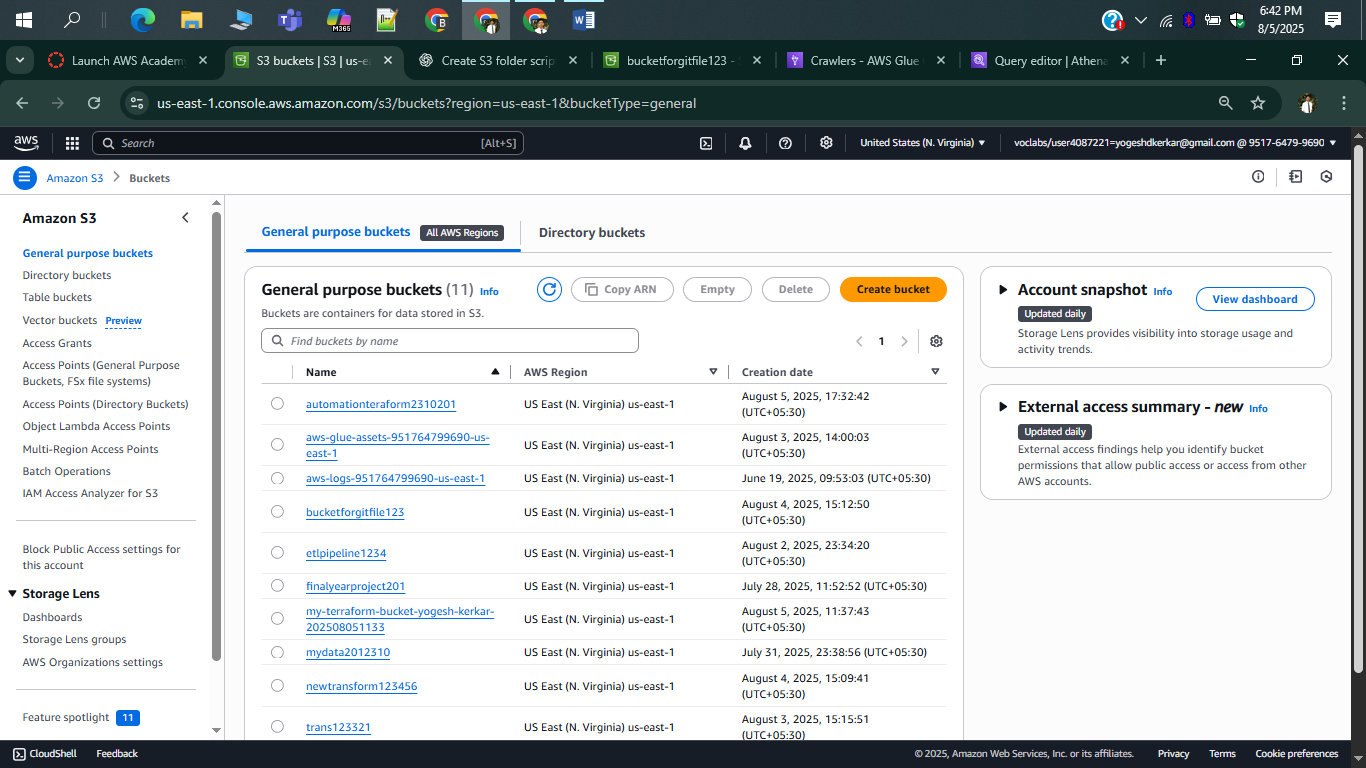
Final deployment tested thoroughly and confirmed working with expected outputs and data flow.



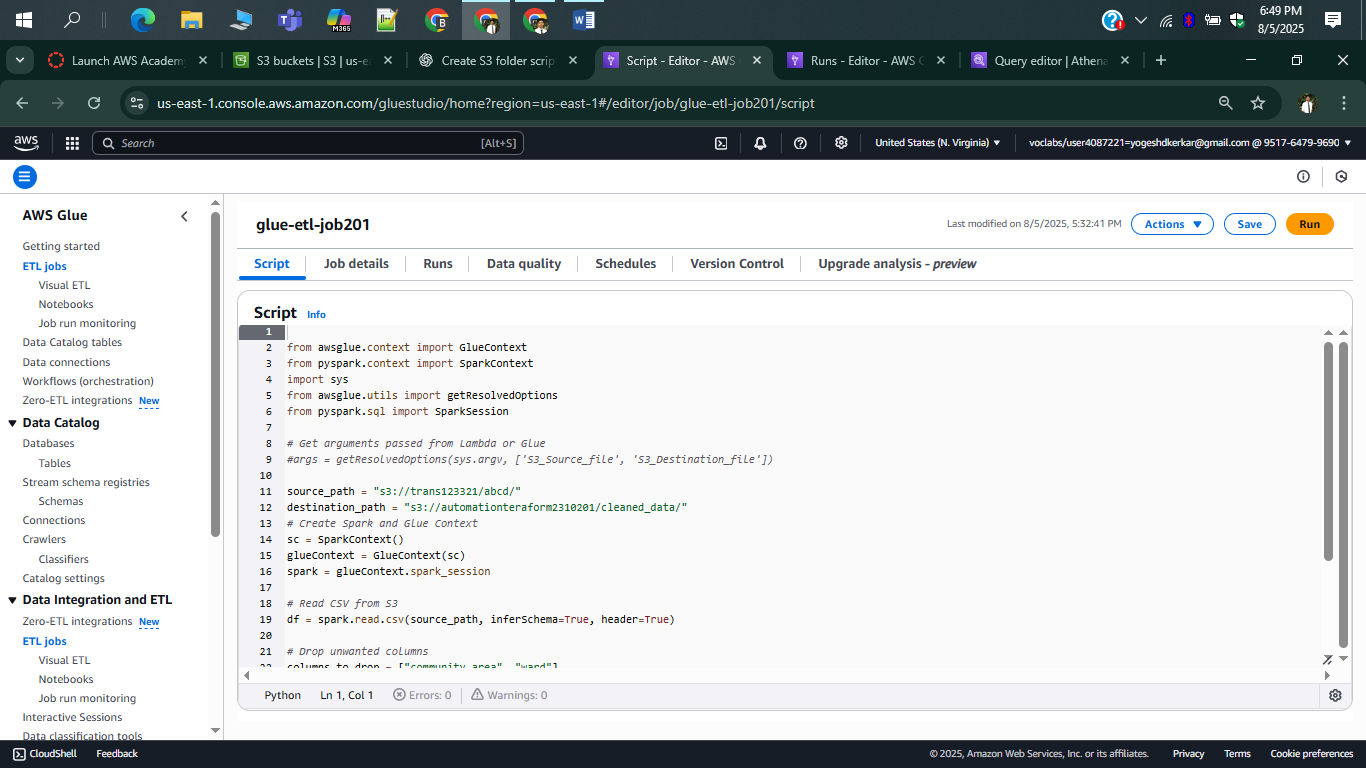
**Task Completed – ETL Automation via GitHub Actions**

* Successfully implemented and executed the **entire ETL automation pipeline** using **GitHub Actions**.
* The workflow includes the following stages, all of which completed successfully:
  + Checkout Code
  + Configure AWS Credentials
  + Terraform Setup, Init, Validate, and Apply
  + Upload ETL Script to S3
  + Trigger AWS Glue ETL Job and wait for its completion
  + Start Glue Crawler to catalog transformed data
* Verified that all steps executed correctly within the CI/CD pipeline on GitHub Actions.
* This automation ensures **infrastructure provisioning**, **ETL job triggering**, and **data cataloging** are now reproducible, scalable, and hands-free.

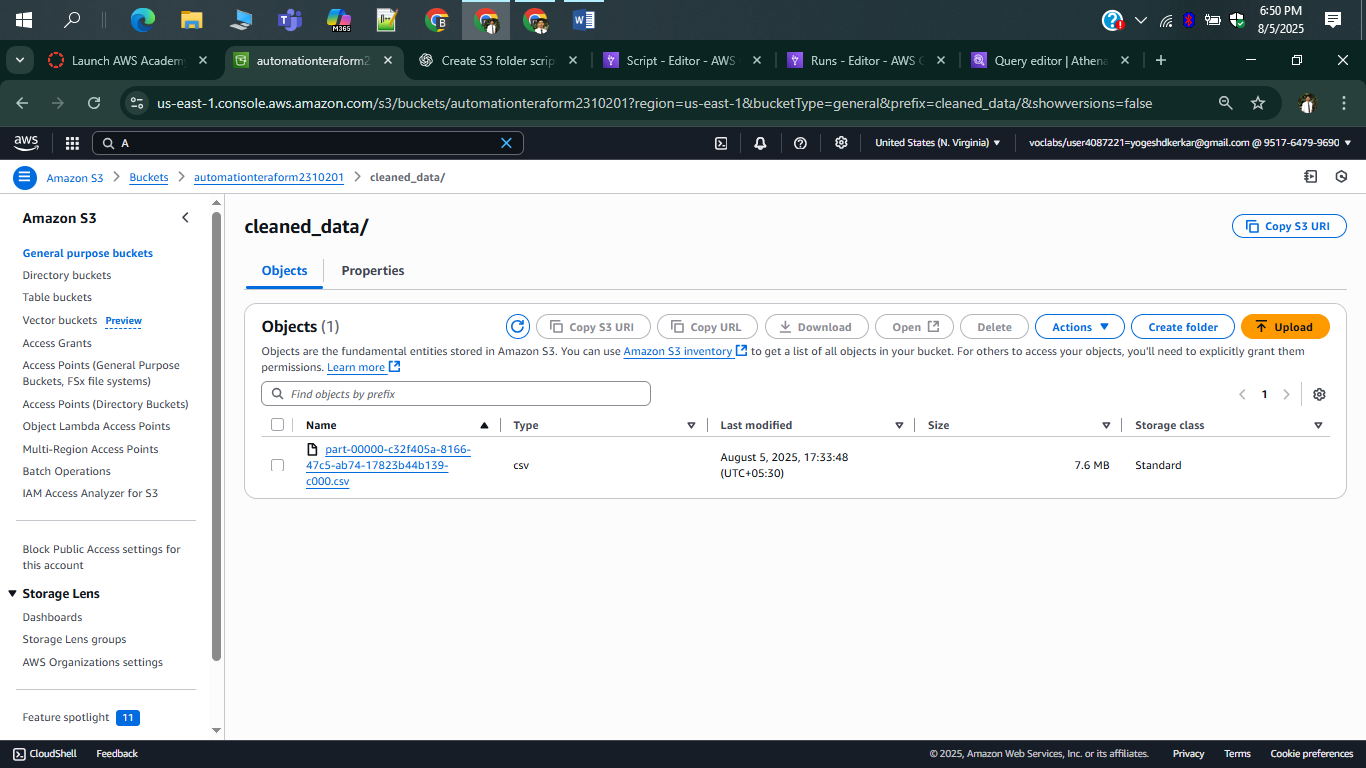


***Successfully created S3 buckets (automationerraform2310) using Terraform infrastructure as code***

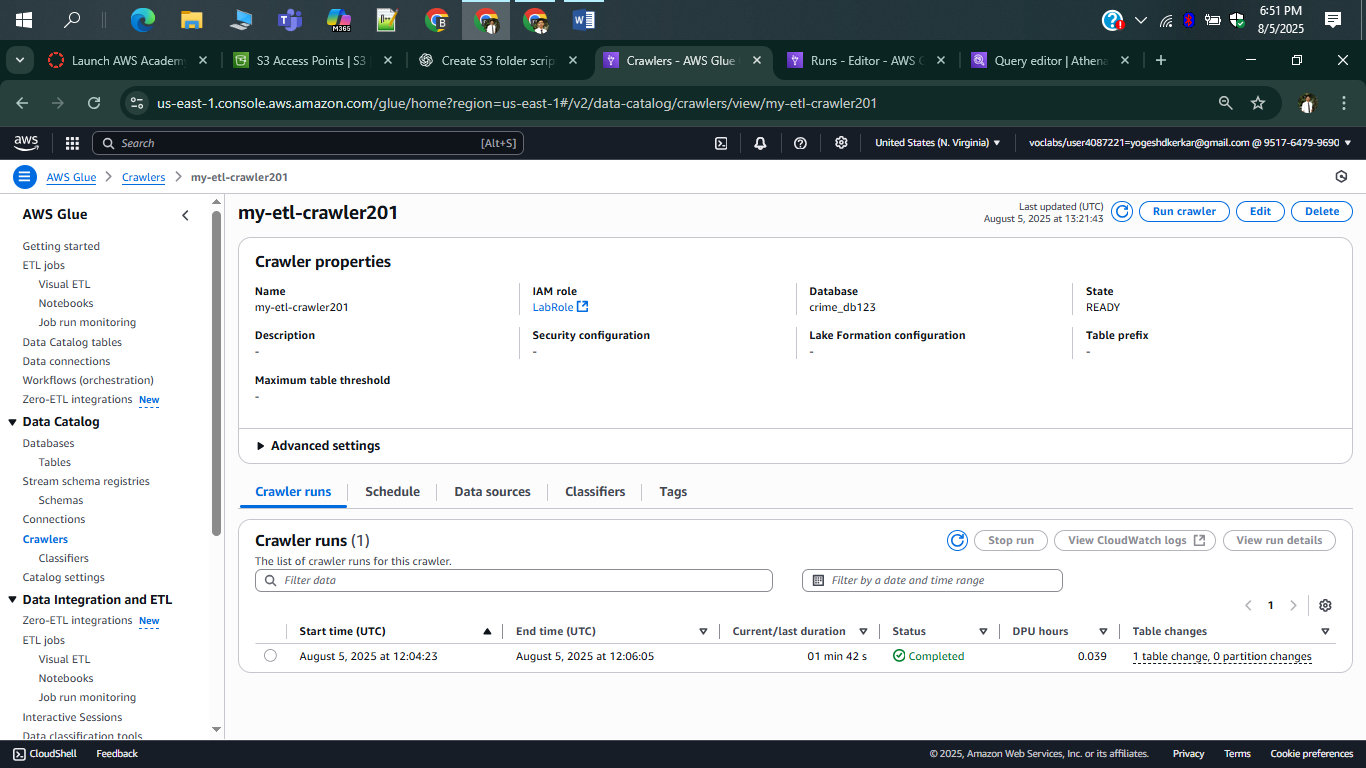
***GLUE ETL JOB CREATED BY TERRAFORM AND RUN***



DATA DUMP INTO BUCKET WHICH IS CREATED BY TERAFORM



AFTER THAT CRAWLER CREATED BY TERAFORM WAS RUN



CRAWLER RUN SUCESSFULLY AND READ THE SCHEMA

