Fall
2024

IST.615.M003.FALL24.Cloud Management

Automated File Processing and Web Display with Docker and Storage Integration (AWS S3)

Team Members
SEJAL SARDAL (NETID - SSARDAL)
NIRANJANA SRINATH (NETID - NSRINATH)
TEJAL PALWANKAR (NETID-TPALWANK)

Link to Video Report

Demo Video: Demo Link

Report Video: Video Report

1. Project Overview

Objective: Create a cloud-based solution that processes files automatically when uploaded to a storage account and displays the processed output via a web interface hosted on Docker within a virtual machine.

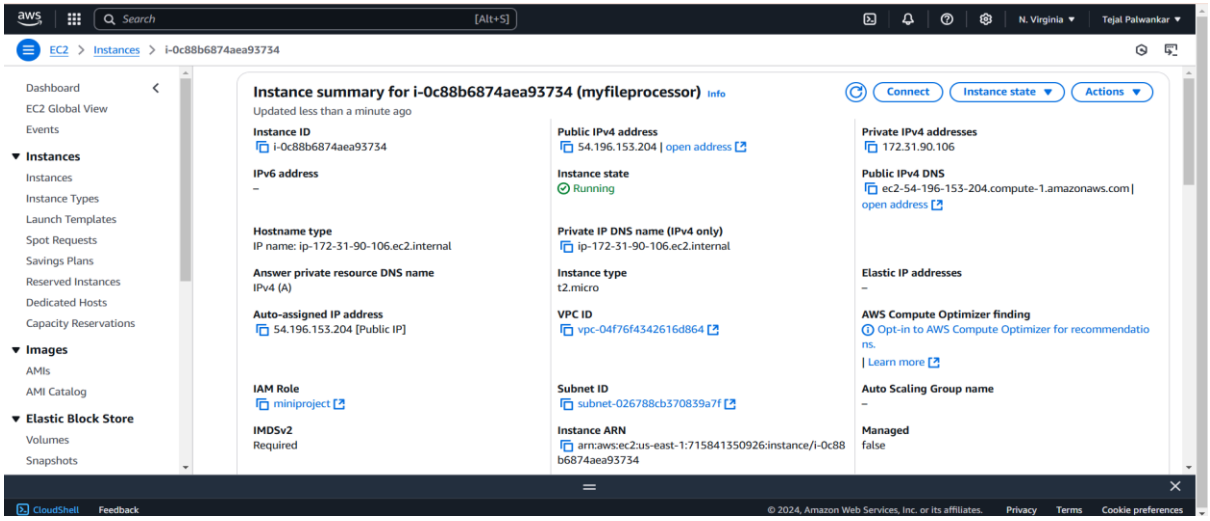
Project Goal: To build a dynamic cloud architecture where files uploaded to cloud storage (AWS S3) are
processed in near real time and displayed to users via a web server running in a Docker container. The goal is to seamlessly integrate storage and computer services, enabling scalable, secure file processing and easy user access to processed data.

Services Used- AWS EC2, S3, Docker, Flask

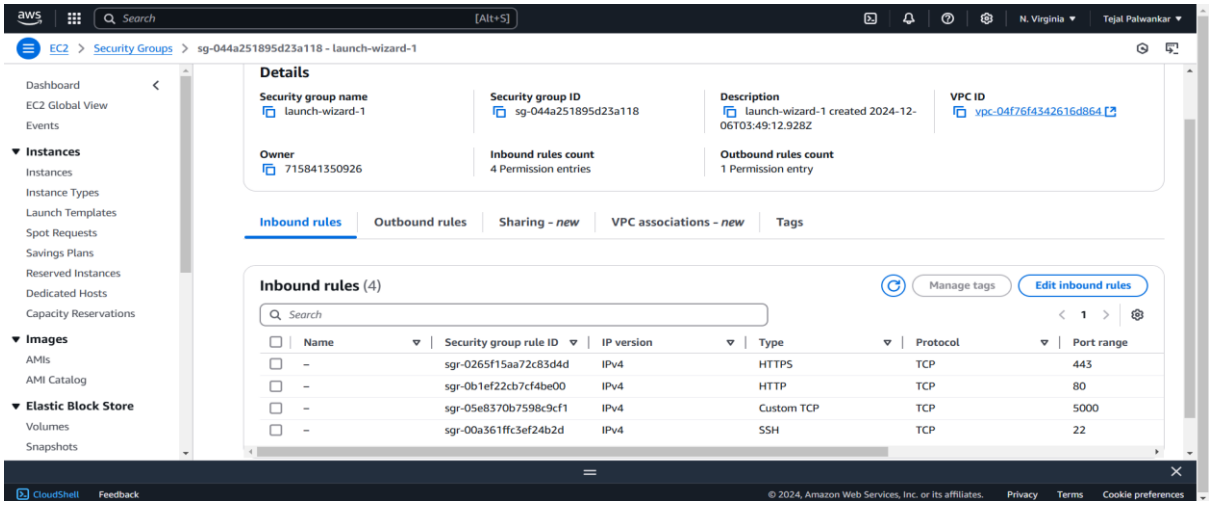
2. AWS EC2 Setup

Screenshots illustrates the launch of an EC2 instance with specific security group settings to enable SSH access and other required ports for the project.

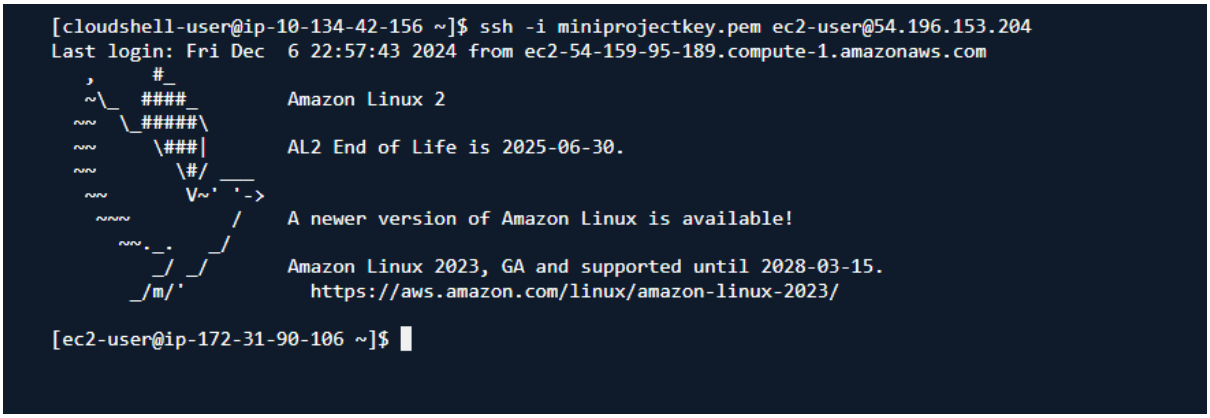
1. **Launch EC2 Instance**:



2. **Security Group Settings**:



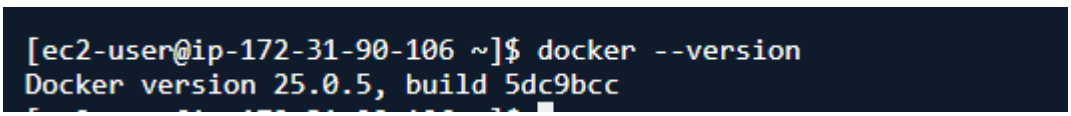
3. **SSH Connection**:



3. Docker Setup

Shows the installation of Docker on the EC2 instance and the creation of a custom Docker image to host the Flask application for web display.

1. **Docker Installation**:



2. **Docker Image Build**:

```
[ec2-user@ip-172-31-90-106 ~]$ ls
file_processing
[ec2-user@ip-172-31-90-106 ~]$ cd file_processing
[ec2-user@ip-172-31-90-106 file_processing]$ ls
Dockerfile  flask.log  function.zip  process_file.py  requirements.txt  templates  web_app.py
[ec2-user@ip-172-31-90-106 file_processing]$ docker build -t file-processor .
[+] Building 0.2s (9/9) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 388B
 => [internal] load metadata for docker.io/library/python:3.9-slim
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/4] FROM docker.io/library/python:3.9-slim@sha256:4ee0613170ac55ebc693a03b6655a5c6f387126f6bc3390e739c2e6c337880ef
 => [internal] load build context
 => => transferring context: 72B
 => CACHED [2/4] WORKDIR /app
 => CACHED [3/4] COPY process_file.py requirements.txt ./
 => CACHED [4/4] RUN pip install --no-cache-dir -r requirements.txt
 => exporting to image
 => => exporting layers
 => => writing image sha256:600a3a4061106a7e7559c415f4fc61ff6c88afe308e921d92393464ae9e99188
 => => naming to docker.io/library/file-processor
[ec2-user@ip-172-31-90-106 file_processing]$
```

```
[ec2-user@ip-172-31-90-106 file_processing]$ docker images
REPOSITORY       TAG       IMAGE ID       CREATED          SIZE
file-processor   latest    600a3a406110   33 minutes ago   157MB
<none>           <none>    0e675c3a9714   18 hours ago     153MB
<none>           <none>    b6ae975cc34a   19 hours ago     153MB
hello-world      latest    d2c94e258dcb   19 months ago    13.3kB
[ec2-user@ip-172-31-90-106 file_processing]$
```

3. **Docker Containers**:

```
[ec2-user@ip-172-31-90-106 ~]$ docker --version
Docker version 25.0.5, build 5dc9bcc
[ec2-user@ip-172-31-90-106 ~]$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
[ec2-user@ip-172-31-90-106 ~]$ docker ps -a
CONTAINER ID   IMAGE            COMMAND                  CREATED        STATUS                     PORTS     NAMES
11c10387f718   file-processor   "python process_file…"   4 hours ago    Exited (0) 4 hours ago               file-processor
b8d8411c6a18   b6ae975cc34a     "python process_file…"   23 hours ago   Exited (0) 23 hours ago              nifty_sammet
263a5300b6c3   hello-world      "/hello"                 23 hours ago   Exited (0) 23 hours ago              elated_shockley
[ec2-user@ip-172-31-90-106 ~]$
```

4. Flask App

Here we have deployed the Flask application, which processes files from the S3 bucket and displays the results on a web interface hosted in a Docker container.

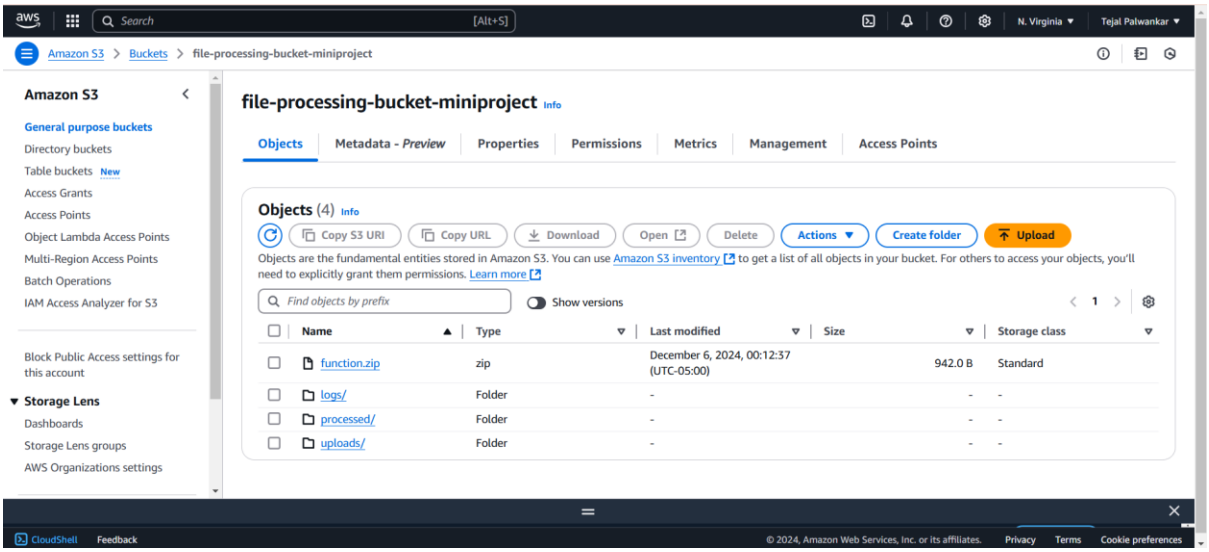
Flask App Code:

```
[ec2-user@ip-172-31-90-106 file_processing]$ python3 web_app.py
/home/ec2-user/.local/lib/python3.7/site-packages/boto3/compat.py:82: PythonDeprecationWarning: Boto3 will no longer support Python 3.7 starting December 13, 2023. To continue receiving service updates, bug fixes, and s
ecurity updates please upgrade to Python 3.8 or later. More information can be found here: https://aws.amazon.com/blogs/developer/python-support-policy-updates-for-aws-sdks-and-tools/
  warnings.warn(warning, PythonDeprecationWarning)
 * Serving Flask app 'web_app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.31.90.106:5000
Press CTRL+C to quit
```

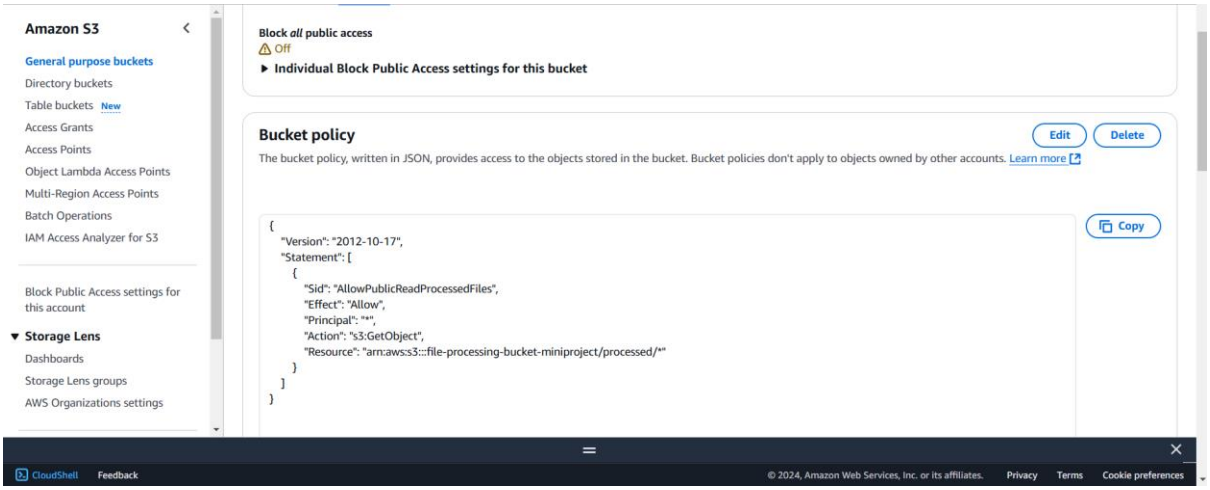
5. S3 Bucket Setup

Highlights the configuration of AWS S3, showing the creation of a bucket with folders for "uploaded" and "processed" files, including permission settings for seamless interaction with other AWS services.

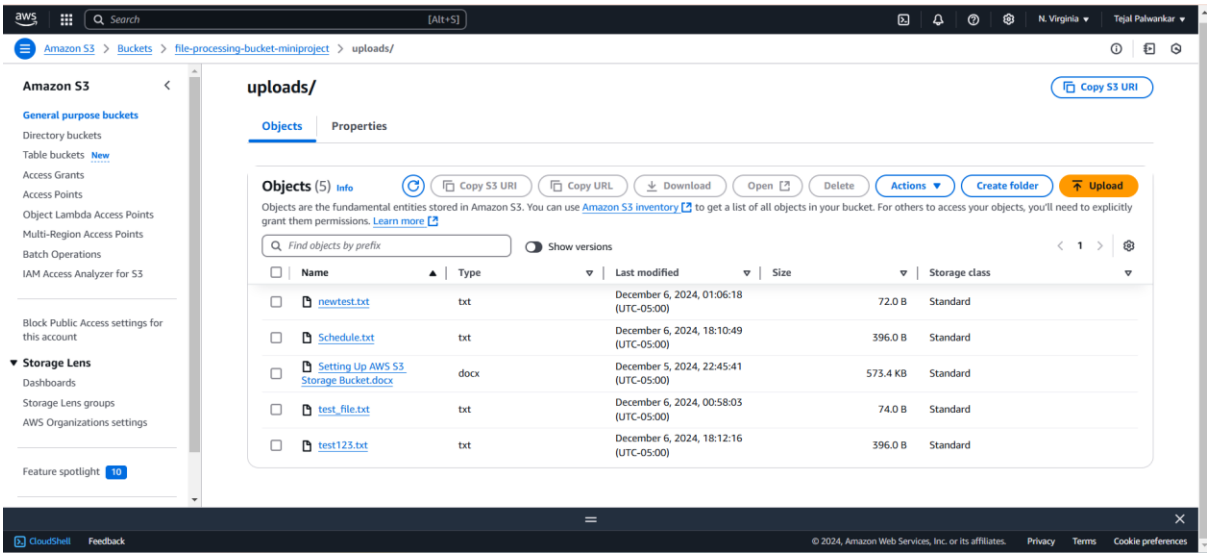
1. **Bucket Configuration**:



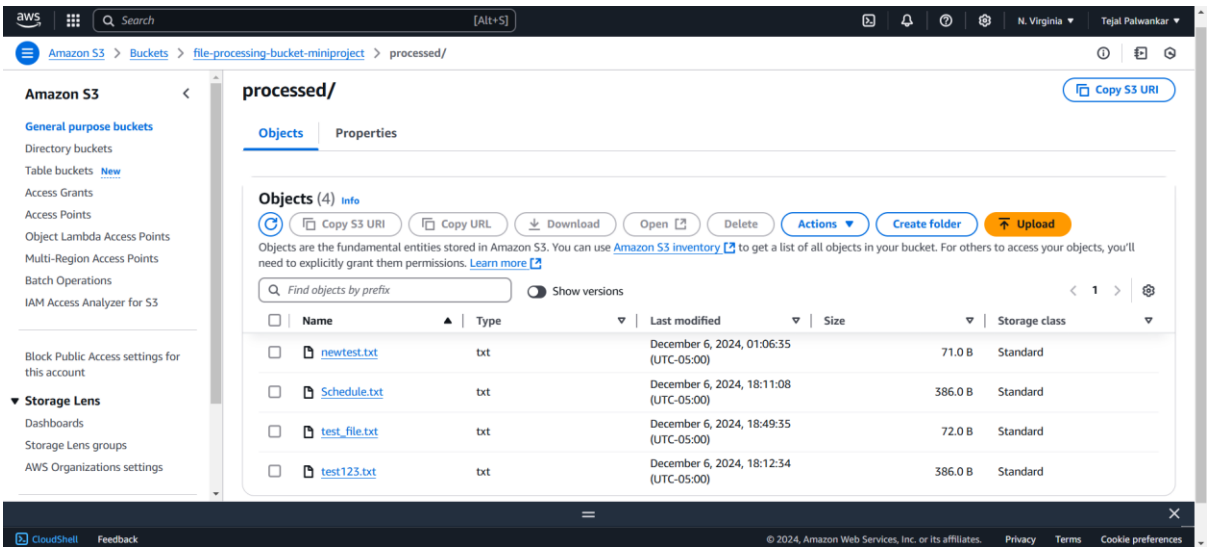
2. **Permissions**:



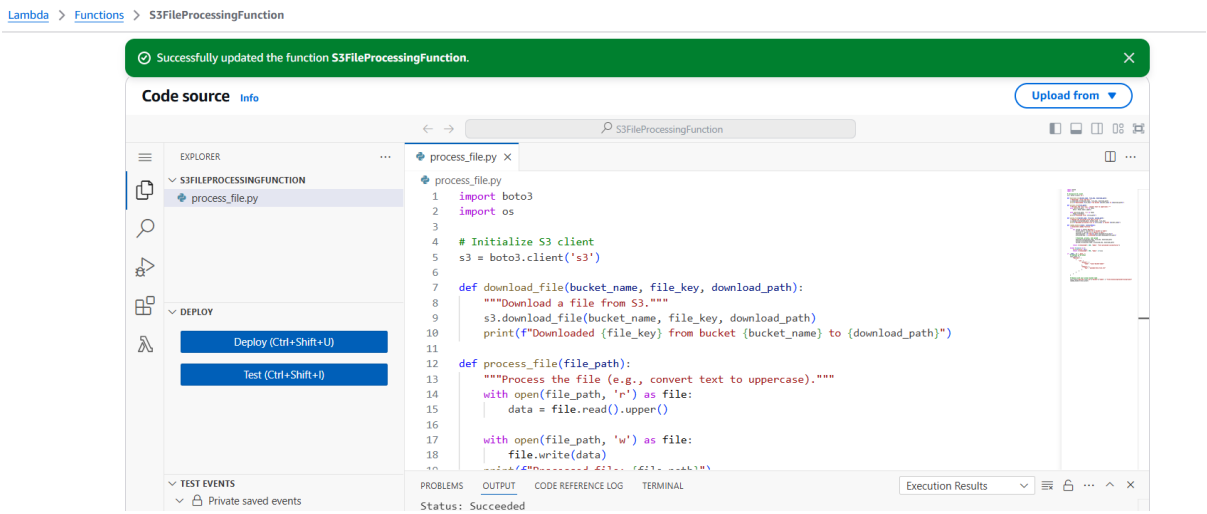
3. **Uploaded Folder**:



4. **Processed Folder**:



6. S3 File Processing Function Demonstrates the lambda function or server-side script interacting with the S3 bucket, showcasing automatic processing of uploaded files.



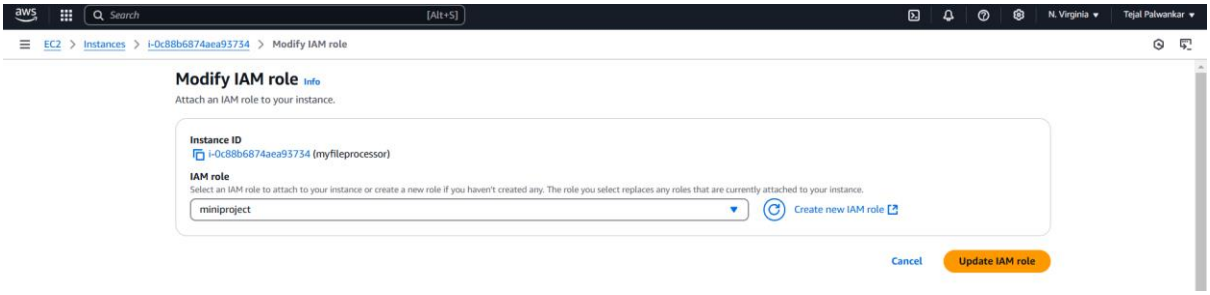
Demonstrates the lambda function or server-side script interacting with the S3 bucket, showcasing automatic processing of uploaded files.

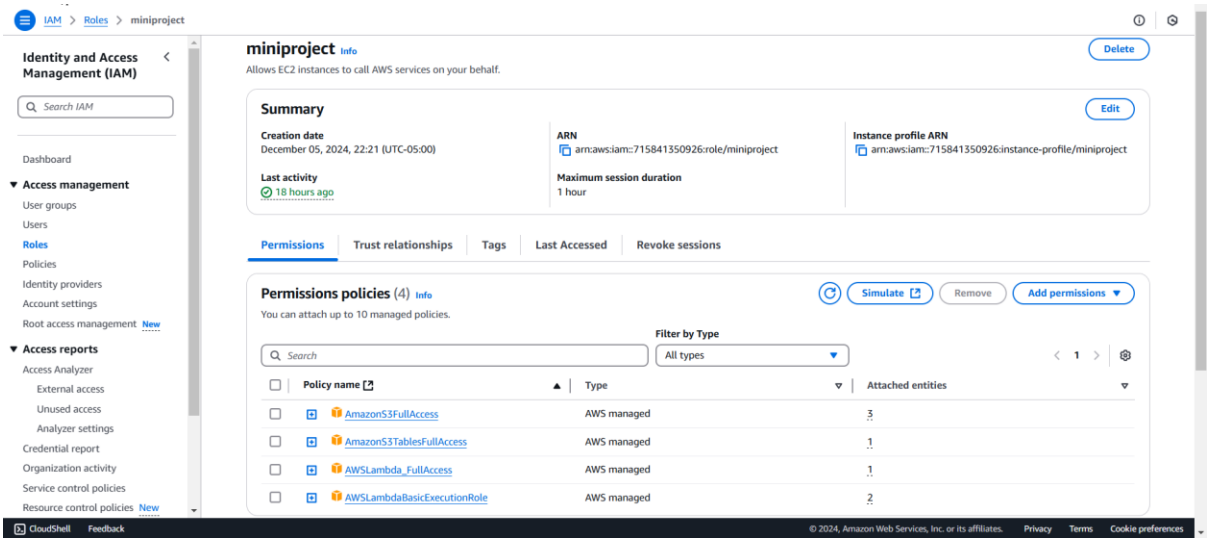
This Python code implements a serverless file-processing pipeline using AWS S3 and simulates an AWS Lambda function. It downloads a file from an S3 bucket, processes it (converts text to uppercase), and uploads the processed file back to a "processed" folder in the same bucket. The lambda_handler function mimics a Lambda trigger responding to S3 events.

processfiles.py

7. IAM Role and Policies

IAM Role:





7. Troubleshooting

Issues Encountered:

Errors we faced, such as `Access Denied`.

Resolutions:

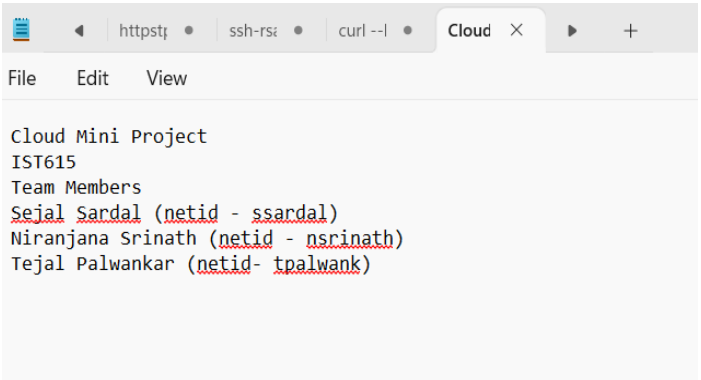
Ensuring the IAM role has the necessary permissions for the resource.

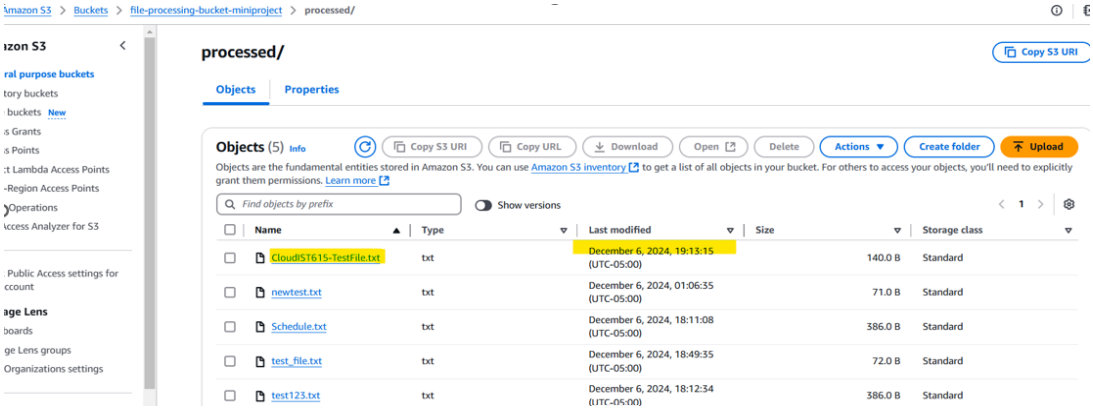
Ensuring the objects in the bucket inherit the correct permissions. This was done through manually setting permissions on individual files.

8. Input/Output Files:

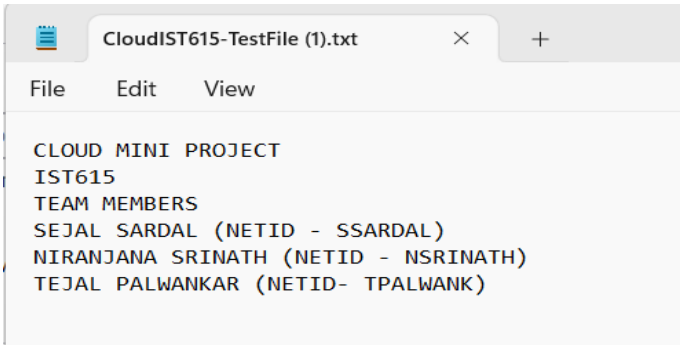
Test Case: Uploaded a file with Lower case alphabets and desired output after processing should be all Upper-case alphabets

Input File





Output File(Screenshot showing the downloaded processed file's content.)



9. Issues Face (Unresolved)

Web Application (During Test):

During out Testing phase or web link was accessible and we were able to see the processed file but later we were not able to resolve the issue hence attaching the screenshot taken during the testing of this project.