PERSONAL FILE SHARING SYSTEM USING AWS LAMBDA, API GATEWAY & S3

A Cloud-Native Serverless Project

Prepared by:
Gopika Rani G

September 2025

Personal File Sharing System using AWS API Gateway, Lambda, and S3

1. Introduction

This project demonstrates a cloud-native file sharing system using AWS services. Users can securely upload and download files without directly accessing the S3 bucket. The system is built using Amazon API Gateway, AWS Lambda, and Amazon S3, providing a scalable and serverless architecture.

2. Problem Statement & Objective

Problem: Traditional file sharing often exposes storage directly or requires third-party tools, which may introduce security risks.

Objective: Build a secure and scalable personal file sharing system where files are uploaded and downloaded using API Gateway endpoints that trigger Lambda functions, which interact with S3.

3. AWS Services Used

- **Amazon S3**: Storage service to hold uploaded files.
- **AWS IAM**: Manages roles and permissions for secure access.
- **AWS Lambda:** Serverless compute to run upload/download logic.
- **Amazon API Gateway:** Exposes REST API endpoints for client interaction.
- **CloudWatch**: For monitoring and debugging logs.

4. Step-by-Step Implementation

4.1 Create S3 Bucket

A new S3 bucket was created with block public access enabled for security. This bucket stores all uploaded files.

4.2 Create IAM Role

An IAM role with AmazonS3FullAccess and CloudWatchLogsFullAccess was created and attached to Lambda functions to allow file operations and logging.

4.3 Create Lambda Functions

Two Lambda functions were developed: Upload and Download.

Upload Lambda Function (Python)

```
import json
import boto3
import base64

s3 = boto3.client('s3')
BUCKET = 'your-bucket-name'

def lambda_handler(event, context):
  try:
    body = json.loads(event['body'])
    file_content = base64.b64decode(body['fileContent'])
    file_name = body['fileName']

    s3.put_object(Bucket=BUCKET, Key=file_name, Body=file_content)

    return {'statusCode': 200, 'body': json.dumps({'message': 'File uploaded
successfully!'})}
  except Exception as e:
    return {'statusCode': 500, 'body': json.dumps({'error': str(e)})}
```

Download Lambda Function (Python)

```
import json
import boto3
import base64

s3 = boto3.client('s3')
```

```
BUCKET = 'your-bucket-name'

def lambda_handler(event, context):
  try:
    file_name = event['queryStringParameters']['fileName']
    obj = s3.get_object(Bucket=BUCKET, Key=file_name)
    file_content = obj['Body'].read()

    return {
      'statusCode': 200,
      'headers': {'Content-Type': 'application/json'},
      'body': json.dumps({'fileName': file_name, 'fileContent':
base64.b64encode(file_content).decode('utf-8')})
    }
  except Exception as e:
    return {'statusCode': 500, 'body': json.dumps({'error': str(e)})}
```

4.4 Configure API Gateway

- Created REST API with two endpoints: POST /upload and GET /download.
- Integrated with respective Lambda functions.
- Enabled CORS for browser-based frontend integration.

4.5 Frontend Integration

A simple HTML, CSS, and JavaScript frontend was created for uploading and downloading files.
Users select a file, which is converted to Base64 and uploaded via API Gateway.
Files can also be downloaded by entering the filename.

5. Errors Faced & Solutions

- NoSuchBucket Error: Fixed by correcting the S3 bucket name.
- Internal Server Error: Solved by adjusting Lambda code to handle input format.
- CORS Error: Resolved by enabling CORS in API Gateway.
- Fetch Failed Error: Solved by running the frontend via local server instead of file:// protocol.

6. Final Outcome

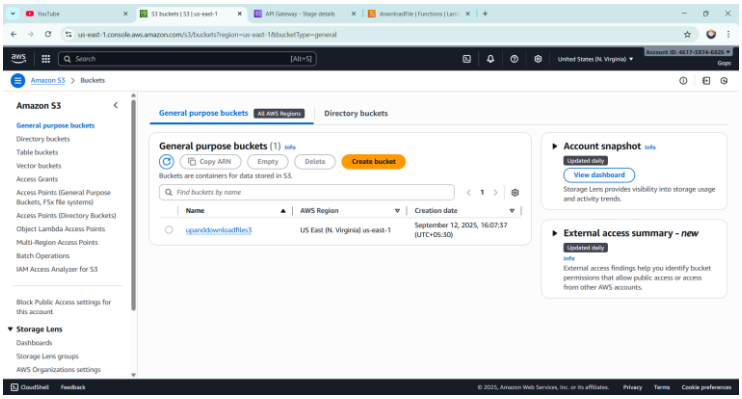
The system successfully uploads and downloads files securely using API Gateway and Lambda. Frontend interaction allows users to share files without exposing the S3 bucket.

7. Conclusion & Future Scope

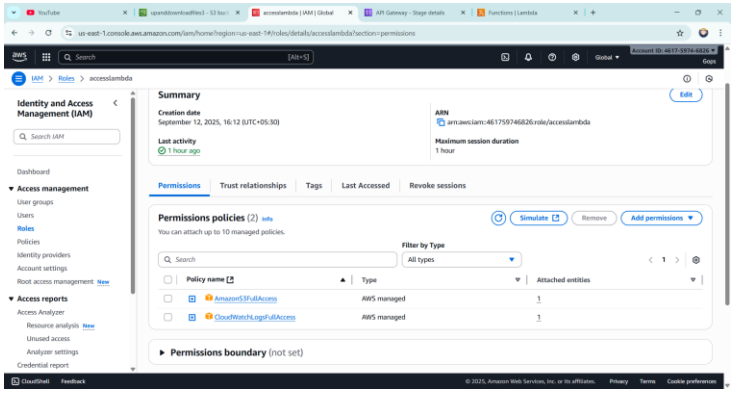
This project demonstrates how to build a secure and serverless file sharing system using AWS. Future improvements may include user authentication via Amazon Cognito, restricting permissions using fine-grained IAM policies, and adding file versioning.

Screenshots

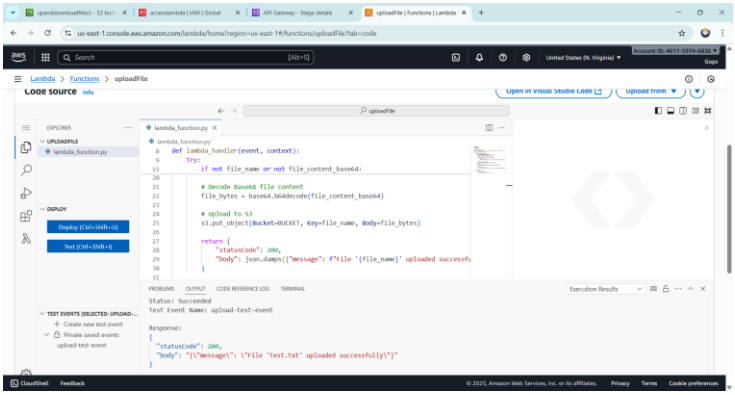
1. AWS S3 Bucket creation



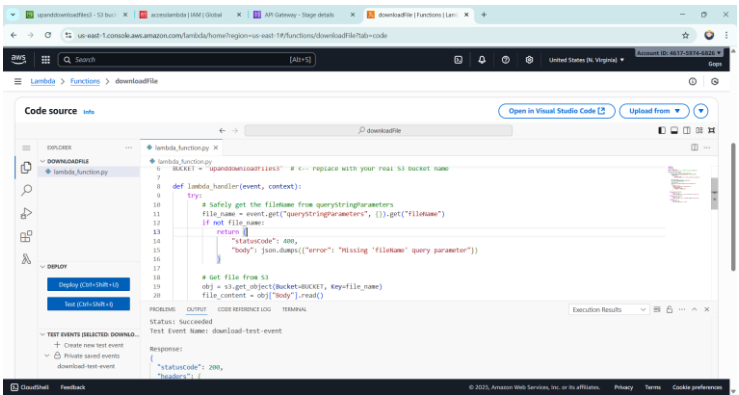
2. IAM Role configuration



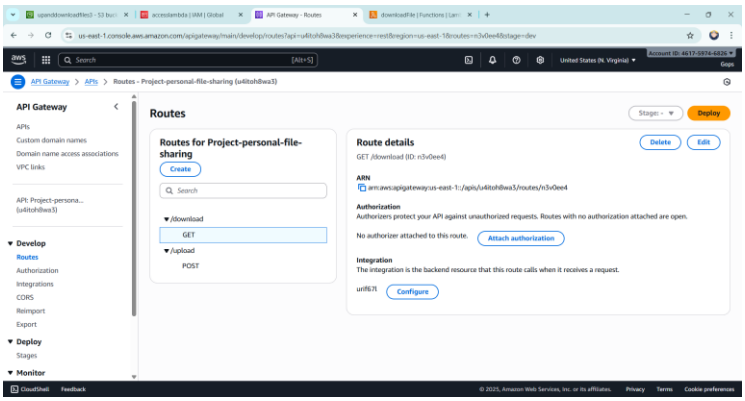
3. Lambda Upload Function code & test

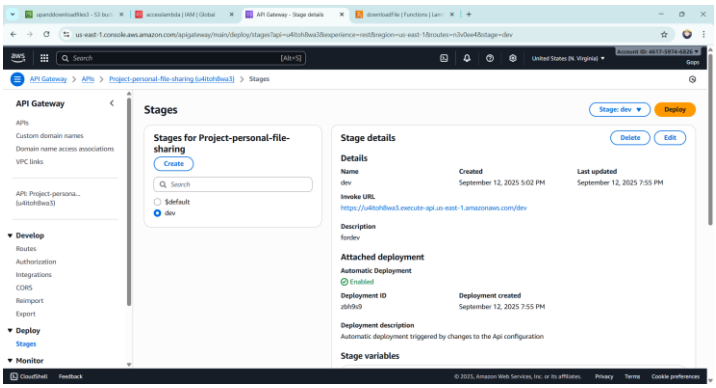


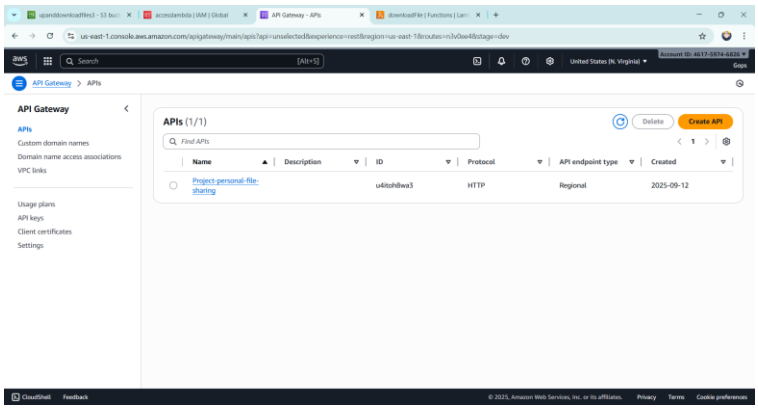
4. Lambda Download Function code & test



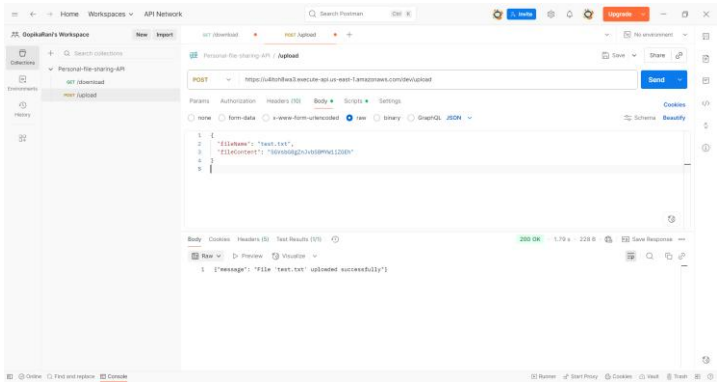
5. API Gateway Routes & Integration

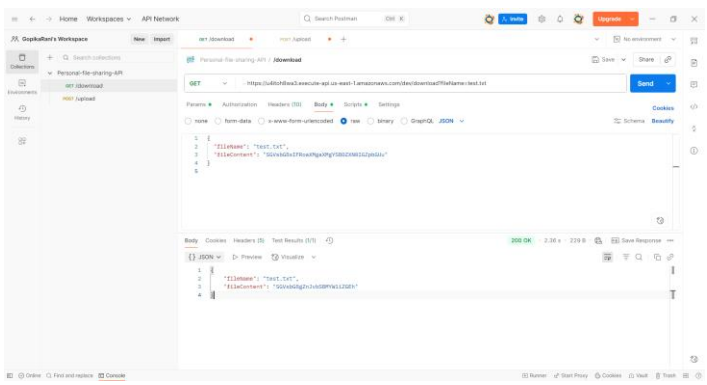




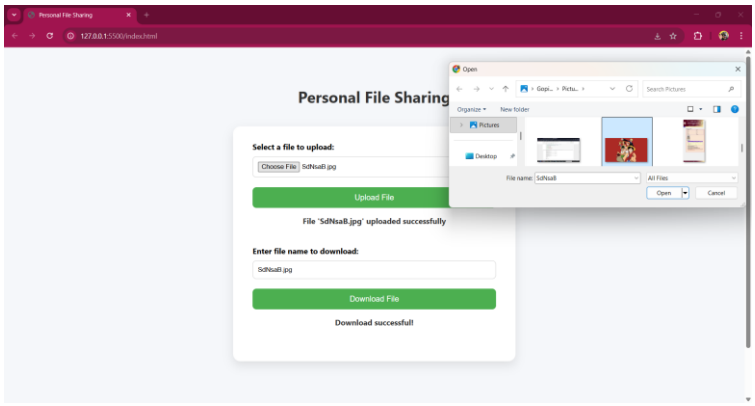


6. Postman test for Upload API & Download API

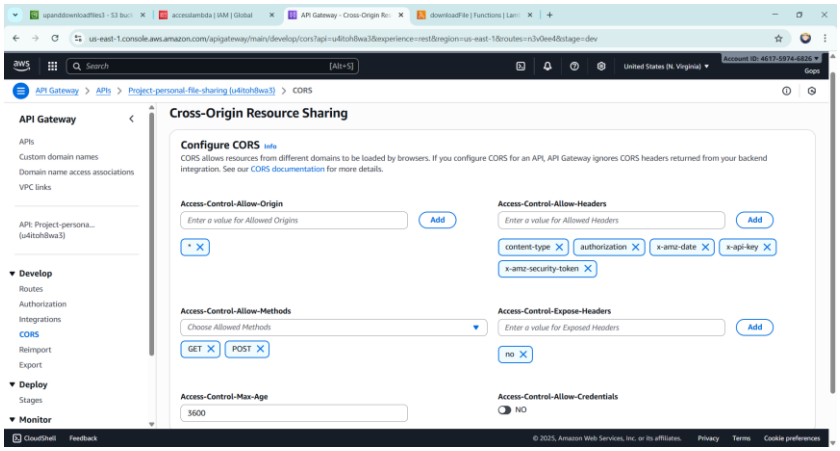




7. Frontend Webpage (Upload + Download)



8. CORS configuration in API Gateway



9. Final Successful Upload & Download in S3-Bucket