Step-by-Step Guide to Build Serverless File Sharing Platform

Project Description:

The Serverless File Sharing Platform allows users to securely upload and download files via a simple HTTP API. It leverages AWS Lambda for serverless compute, API Gateway for RESTful API management, and Amazon S3 for durable and scalable object storage. Users can interact with the platform using any HTTP client (like Postman), making it versatile for various use cases that involve file sharing and storage.

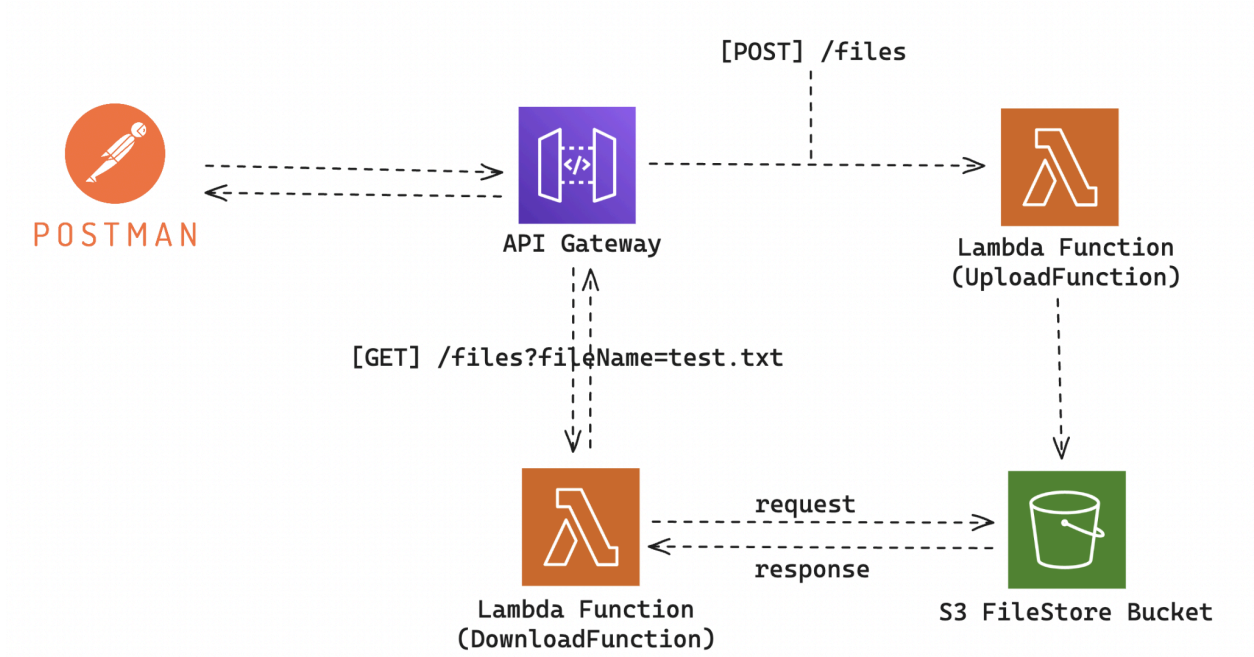
Use Cases:

File Sharing: Users can upload files to the platform using a POST request, specifying the file name and content. This can be useful for sharing documents, images, or any other digital assets securely.

File Distribution: Content creators can distribute files (e.g., software updates, media files) to consumers by generating download links through the platform.

Collaborative Work: Teams can share project resources, documents, and data securely, facilitating collaboration across different locations.

Project Architecture:



Steps to Build the Project:

Prerequisites

AWS Account with appropriate permissions to create Lambda functions, API Gateway, and S3 buckets.

Steps to Deploy

Step 1: Create an S3 bucket to store uploaded files:

myfilesharingbucket

Step 2: Create the Lambda function to handle file uploads (UploadFunction):
Name: UploadFunction
Runtime: Python 3.x
Execution role: IAM role with S3 write permissions
Code: Use the UploadFunction Python code.

Step 3: Create the Lambda function to handle file downloads (DownloadFunction):
Name: DownloadFunction
Runtime: Python 3.x
Execution role: IAM role with S3 write permissions
Code: Use the DownloadFunction Python code.

Step 4: Create an API Gateway
Name: my-file-sharing-api-amc
Create two resources: /files with POST and GET methods.
For each method, configure Lambda integration with UploadFunction and DownloadFunction respectively.

Step 5: Configure GET Method:
Method Request --> Edit --> Request validator --> Validate Query String Parameters and Headers
Method Request --> Edit --> Request Body --> text/plain
Integration Request --> Edit --> Mapping Templates --> Content Type: application/json --> Content Body:

```
{
  "queryStringParameters": {
      "fileName": "$input.params('fileName')"
  }
}
```

Step 6: Configure POST Method:
Integration Request --> Edit --> Mapping Templates --> Content Type: text/plain --> Content Body: \

```
{
  "body" : "$input.body",
  "queryStringParameters" : {
      "fileName" : "$input.params('fileName')"
  }
}
```

Step 7: Deploy API Gateway:
Deploy the API to a stage (e.g., dev):
Click on "Actions" > "Deploy API".
Choose your stage (e.g., dev) and deploy.

**Step 8: Testing **

1. Upload a File
Use Postman or another HTTP client:
POST to https://.execute-api..amazonaws.com/dev/files?fileName=test.txt
Set request body to raw and content type to text/plain.
Enter file content (e.g., Hello World!) and send the request.

or

Use CURL Utility:

curl --location
'https://<api-id>.execute-api.<region>.amazonaws.com/dev/files?fileName=test.txt' \
--header 'Content-Type: text/plain' \
--data 'Hello World from A Monk in Cloud!'

2. Download a File
Use Postman or another HTTP client:
GET to https://.execute-api..amazonaws.com/dev/files?fileName=test.txt
Verify file content is returned correctly.

or

Use CURL Utility

curl --location
'https://<api-id>.execute-api.<region>.amazonaws.com/dev/files?fileName=test.txt'