



Project Author

Name: Waleed Ahmed

GitHub: <https://github.com/Waleedcoderarch/Google-Drive-Clone-using-AWS-.git>

Course: CloudDevops



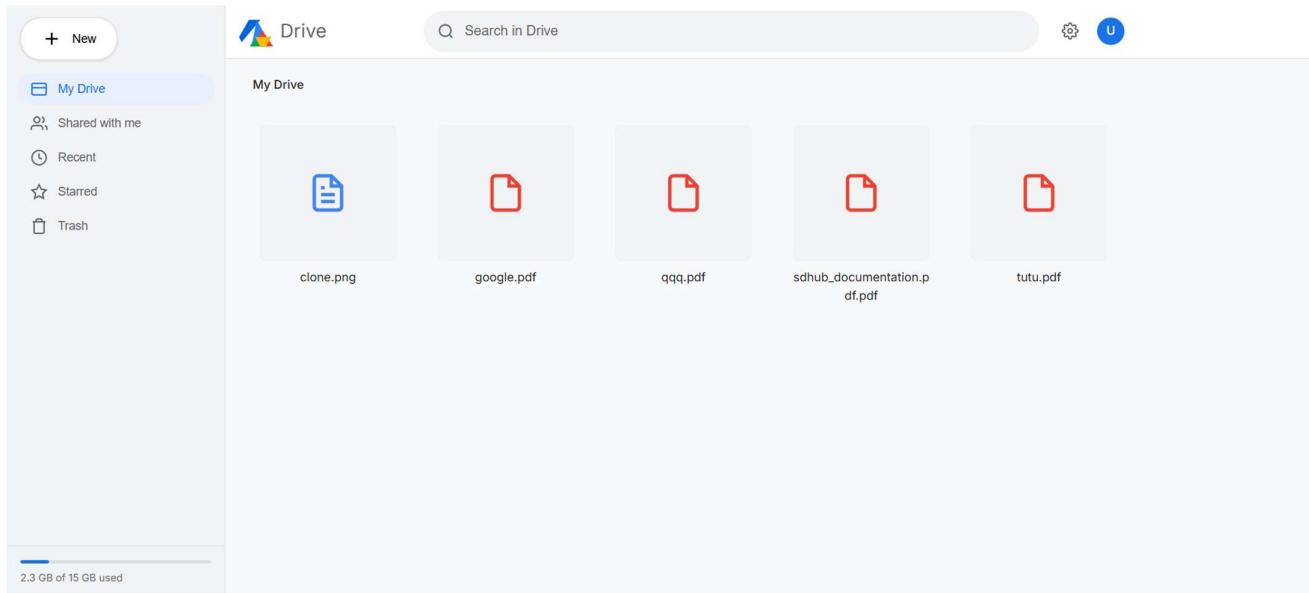
Project Title

Google Drive Clone Using AWS S3, Lambda, and API Gateway



App Preview

This application provides a Google Drive–style interface where users can view files stored in AWS cloud storage.



Project Explanation

This project is a **Google Drive Clone UI** where users can view files that are stored inside an **Amazon S3 bucket**.

Architecture Flow:

User → Google Drive Clone UI → API Gateway → Lambda Function → Amazon S3 Bucket

What Happens in This Project:

- The user opens the Google Drive Clone website.
- The UI requests file data from AWS.
- API Gateway receives the request and forwards it to Lambda.
- Lambda function fetches file information from the S3 bucket.
- The file list is sent back to the UI.

- The UI displays all cloud files to the user.

This makes the application fully cloud-connected and serverless.



⌚ Project Implementation

Below are the steps I followed to build this project.

Step 1 — Create S3 Bucket (Cloud Storage)

- I created an **Amazon S3 bucket**.
- Uploaded sample files such as PDFs and images.
- These files act as cloud storage data.

The screenshot shows the 'Objects' tab in the Amazon S3 console. The top navigation bar includes tabs for Objects, Metadata, Properties, Permissions, Metrics, Management, and Access Points. Below the tabs is a toolbar with actions: Copy S3 URI, Copy URL, Download, Open, Delete, Actions (with a dropdown), Create folder, and Upload. A search bar at the top says 'Find objects by prefix'. The main area lists five objects:

Name	Type	Last modified	Size	Storage class
clone.png	png	February 4, 2026, 23:44:37 (UTC+05:30)	671.4 KB	Standard
google.pdf	pdf	February 4, 2026, 22:50:49 (UTC+05:30)	199.6 KB	Standard
qqq.pdf	pdf	February 4, 2026, 22:37:16 (UTC+05:30)	4.2 MB	Standard
sdhub_documentation.pdf.pdf	pdf	February 4, 2026, 22:37:11 (UTC+05:30)	762.9 KB	Standard
tutu.pdf	pdf	February 4, 2026, 22:20:27 (UTC+05:30)	45.8 KB	Standard

S3 is used because it provides:

- Secure storage
 - High scalability
 - Fast access to files
-

Step 2 — Create IAM Role

- I created an IAM role for Lambda.
- Attached permission:

AmazonS3FullAccess

This allows Lambda to:

- Read files from S3
 - List bucket objects
 - Generate file access links
-

Step 3 — Create Lambda Function

- I created a Lambda function using **Python**.
- Inside the function, I wrote logic to:

✓ Fetch file names from S3

✓ Generate secure URLs

✓ Send data to API Gateway

Lambda acts as the backend without managing any servers.

Step 4 — Create API Gateway

- I created a REST API in API Gateway.
- Created resources and HTTP methods.
- Connected API Gateway with the Lambda function.
- Enabled **CORS** for frontend browser access.

- Deployed the API to production stage.

After deployment, API Gateway generated an **Invoke URL**.

Step 5 — API Connection Testing

To verify the connection:

- I pasted the API URL into the browser.
- If file names appeared → connection was successful ✓
- If not → configuration issue ✗

This confirmed:

- Lambda is connected to S3
 - API Gateway is working correctly
-

Step 6 — Frontend Integration

- I added the API Gateway URL inside my JavaScript code.
- The frontend fetches cloud data using this API.
- The UI dynamically loads files from S3.

Now when users open the website:

- ✓ Files are loaded from cloud
 - ✓ Real-time data is displayed
 - ✓ Serverless backend is active
-

✓ Final Outcome

This project successfully demonstrates:

- Serverless cloud architecture
- Frontend to AWS integration
- Real-time cloud data fetching
- Secure file access using AWS services

It works like a **basic Google Drive system** using AWS cloud services.