Danny Lo

Serverless-Project-Management

Purpose

- Problem Statement: A customer frequently receives large numbers of files that need to be processed for metadata extraction. They need an automated, scalable solution that can handle variable load, extract metadata efficiently, and provide easy access to this information. Manual processing is time-consuming and error-prone, and they lack visibility into the processing status of individual files.

- Goal Implementation: To implement a webapp to further abstract and streamline the process of managing documents to help customers and their customers

---

S3

- Purpose: Serves as the focal point of the project
    - Stores objects to be further processed
    - Keeps objects indefinitely or until use no longer has use of object

- Implementation: Created a S3 bucket with a AWS specific name
    - Granted CORS for further abstraction of the application (See: ddddd)
    - Public access of for security

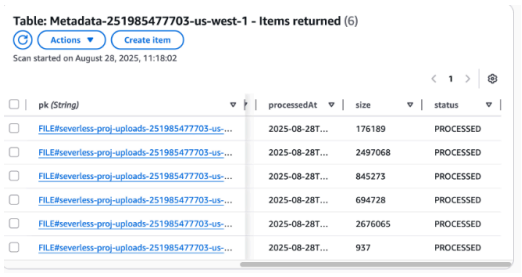
Cross-origin resource sharing (CORS)

The CORS configuration, written in JSON, defines a way for client web applica

```
[
  {
    "AllowedHeaders": [
      "*"
    ],
    "AllowedMethods": [
      "PUT"
    ],
    "AllowedOrigins": [
      "https://main.d28wvz3bs3pz1c.amplifyapp.com"
    ],
    "ExposeHeaders": [
      "ETag"
    ]
  },
  {
    "AllowedHeaders": [
      "*"
    ],
    "AllowedMethods": [
```

DyanmoDB

- Purpose: Serves as the endpoint of the metadata extraction.
    - Allows users visibility into the processing status of individual file
    - Fast lookup
    - Stores immutable metadata



- Implementation: Created a table
  - Set pk as partition key’
    - Determines which data partition each table belongs to
  - Set sk as sort key
    - sort order

EventBridge

- Purpose: Receives event notification from S3, indicating that a object has been created
  - Starts the event pattern (target). (See ...)

Event pattern Info

```
{
  "detail-type": ["Object Created"],
  "source": ["aws.s3"],
  "detail": {
    "bucket": {
      "name": ["severless-proj-uploads-251985477703-us-west-1"]
    }
  }
}
```

- Implementation: Created a rule to indicate which pattern to track
  - Specified target

Targets

Details	Target Name	Type
▼	S3ObjectCreatedWorkflow-251985477703-us-west-1	Step Functions state machine

Input to target: Matched event

Identity and Access Management

- Purpose: Grant minimum permissions for each components, ensuring security of assets.
  - Allows Lambba function to pregenerate presigned URLs to write into S3
  - Allows Lambda function access into S3 bucket

Cognito

- Purpose: Provide secure user authentication via Cognito Hosted UI using a authorization code flow.

- Admin creates a user: User manages is allowed are allowed access to resources within the webapp

- Implementation: Created user pool with region of applications
  - Created app client within the user pool

User Interface

- Purpose: To further abstract the process of managing documents to allow customers more time to focus on growing their buisness

- Implementation: Use Amplify to host webapp using github repository

---

API Gateway

- Purpose: Provide webapp access to AWS resources in a secure manner through admin provided routes

- Implementation:

▼ /downloads

▼ /create

POST

▼ /files

▼ /list

GET

▼ /uploads

▼ /create

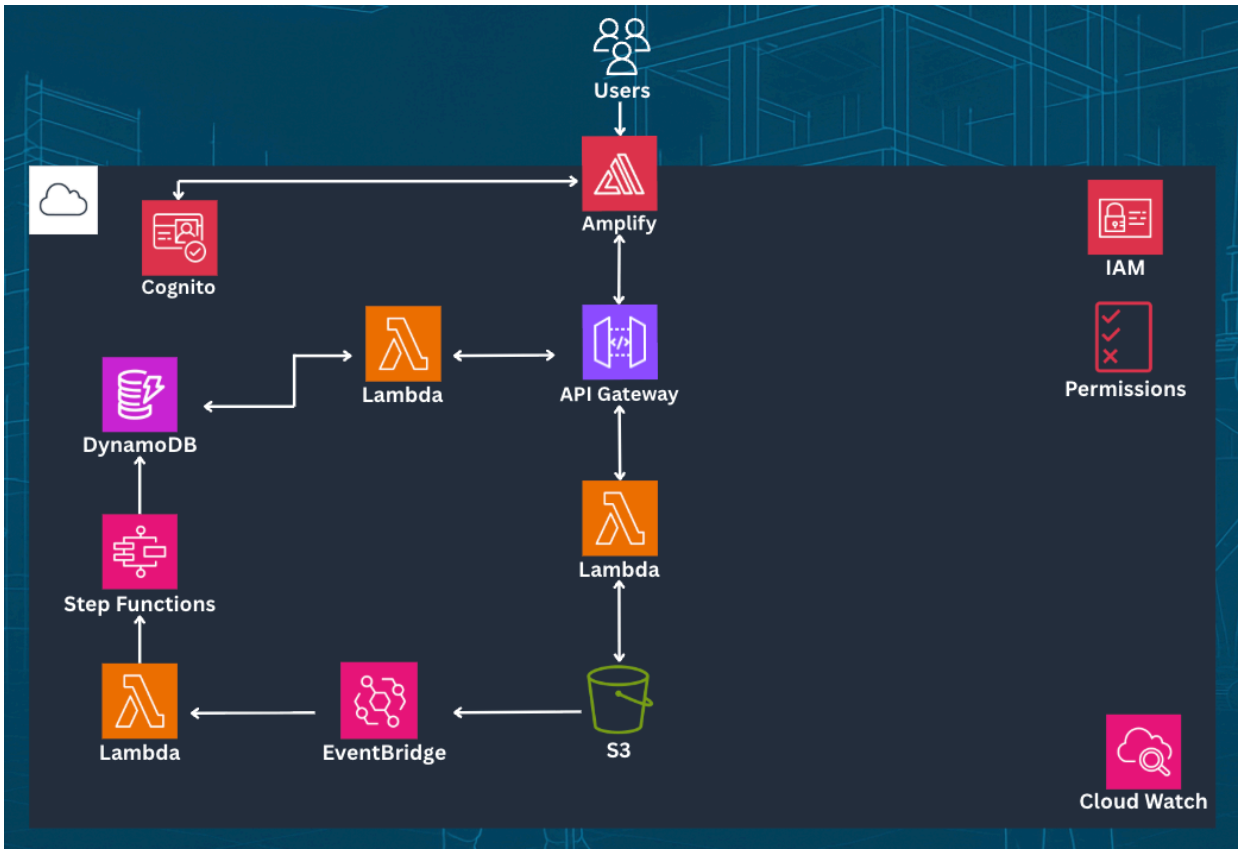
POST

Lambda / Stepfunctions

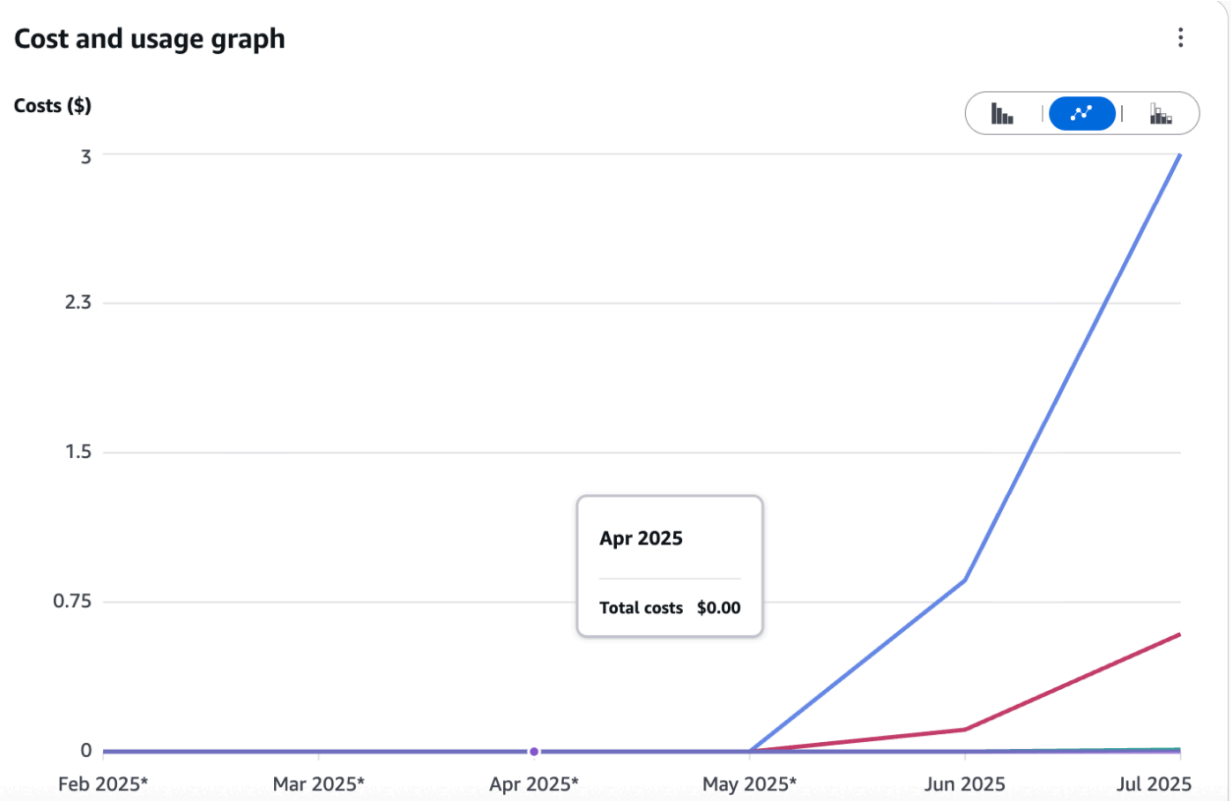
-

Cloudwatch

Architecture at a Glance



Cost Analysis



Total cost of implementation: $4.25

Cost to run the service monthly: $23.92

- Amplify: $21.35
- API Gateway: $0.11
- Lambda: $0.00
- DynamoDB: $0.10
- S3: $2.36

Future Implementation

- Improvements to user experience
  - QuickSight
  - S3 bucket tiering
  - Allow to view documents on webapp
- Improvements on infrastructure
  - Cloud Front
  - Database Backups
  - Clean up code

Questions Answered

Resources