**AWS Lambda with S3 Integration: System Documentation**

**1. System Overview**

This system automates the process of handling file uploads to an S3 bucket. When a file is uploaded to the S3 bucket, it triggers an AWS Lambda function that processes the file, performs actions, and stores results back to S3 or other destinations.

**2. AWS Configuration**

**Lambda Function Configuration**

* **Function Name**: my-lambda-function
* **Runtime**: Node.js 22.x
* **Handler**: index.handler
  + The Lambda function is triggered by the S3 event, and the index.mjs file contains the handler function.
* **IAM Role**:
  + The function assumes an IAM role that has necessary permissions to read/write from/to the S3 bucket.

**IAM Role Permissions**

* **Role Name**: lambda-s3-role
* **Permissions**:
  + s3:GetObject - Allows reading files from S3.
  + s3:PutObject - Allows writing files to S3.

**S3 Bucket Configuration**

* **Bucket Name**: my-file-bucket
* **Event Trigger**:
  + The Lambda function is triggered when a new file is uploaded to the S3 bucket.
  + Event: s3:ObjectCreated:\*

**IAM Role Trust Relationship**

* The Lambda function's IAM role trusts the Lambda service (lambda.amazonaws.com) to assume the role.

**3. API Documentation**

**Lambda Function Endpoint**

* **API URL**: There is no direct API URL for this system; it is triggered by an S3 event.

**S3 Event Trigger**

* **Trigger Event**: s3:ObjectCreated:\*
* **Lambda Function Trigger**: When an object is uploaded to the S3 bucket, the event triggers the Lambda function.

**4. Security Implementation Details**

* **IAM Role**: Only necessary permissions are granted to the Lambda function using IAM policies. This ensures the principle of least privilege.
* **S3 Bucket Policy**: The S3 bucket has policies to allow Lambda access while preventing public access.

**5. Steps to Set Up AWS Components**

**Lambda Function Setup**

1. Go to the AWS Lambda Console and create a new function.
2. Select Node.js as the runtime (e.g., Node.js 22.x).
3. Upload your function code (either ZIP or directly via the editor).
4. Set the **Handler** to index.handler.
5. Assign the IAM role (lambda-s3-role) to the Lambda function.

**S3 Bucket Setup**

1. Go to the AWS S3 Console and create a new bucket (e.g., my-file-bucket).
2. Set up the event notification to trigger the Lambda function upon object creation.
3. Attach the **Lambda function trigger** to the bucket.

**IAM Role Setup**

1. Go to the IAM Console and create a new role for Lambda.
2. Attach policies like s3:GetObject, s3:PutObject to the role.
3. Set the trust relationship to allow Lambda to assume the role.

**Screenshots**

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screen shot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

**6. Challenges Faced & Resolutions**

* **Lambda Deployment**: Initially, the order of files in the deployment package caused issues with module imports. The issue was resolved by ensuring the index.mjs file was placed at the root of the ZIP file, followed by node\_modules.
* **Module Import Error**: The error Cannot use import statement outside a module was caused by not configuring the package.json correctly for ES modules. This was fixed by adding "type": "module" in the package.json file.
* **Permissions**: Incorrect IAM policies led to errors while accessing the S3 bucket. These were resolved by ensuring the IAM role had the correct permissions (s3:GetObject and s3:PutObject).