Project Title:

Serverless Notes Web Application

Overview:

A simple web application that allows users to create, view, update, and delete notes. The entire application is built using AWS services in a serverless architecture.

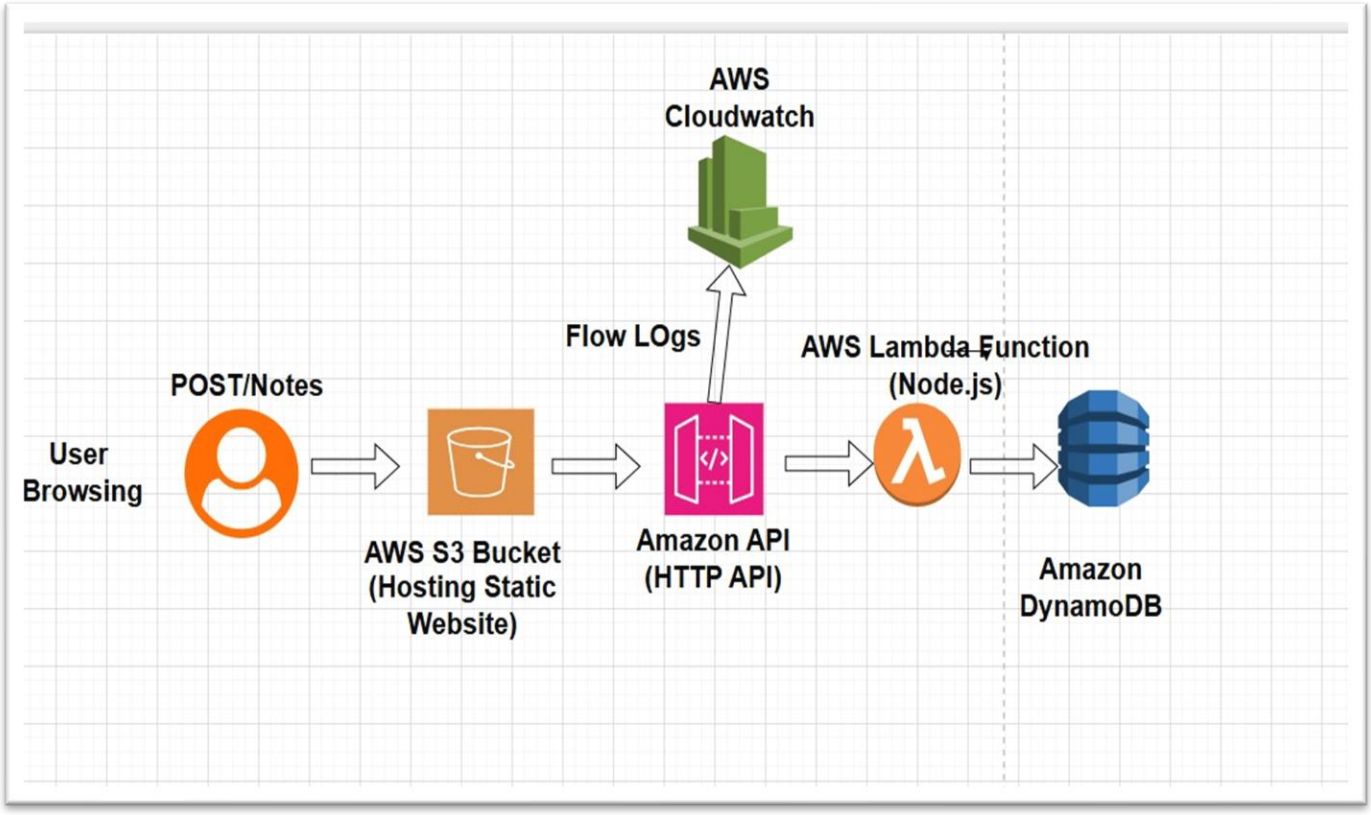
Tech Stack:

- **Frontend:** HTML, CSS, JavaScript (Hosted on Amazon S3)
- **Backend:** AWS Lambda + API Gateway
- **Database:** Amazon DynamoDB
- **Other Services:** IAM for access control

Main Features:

- Add a note
- View all notes
- Edit a note
- Delete a note
- Notes have timestamps

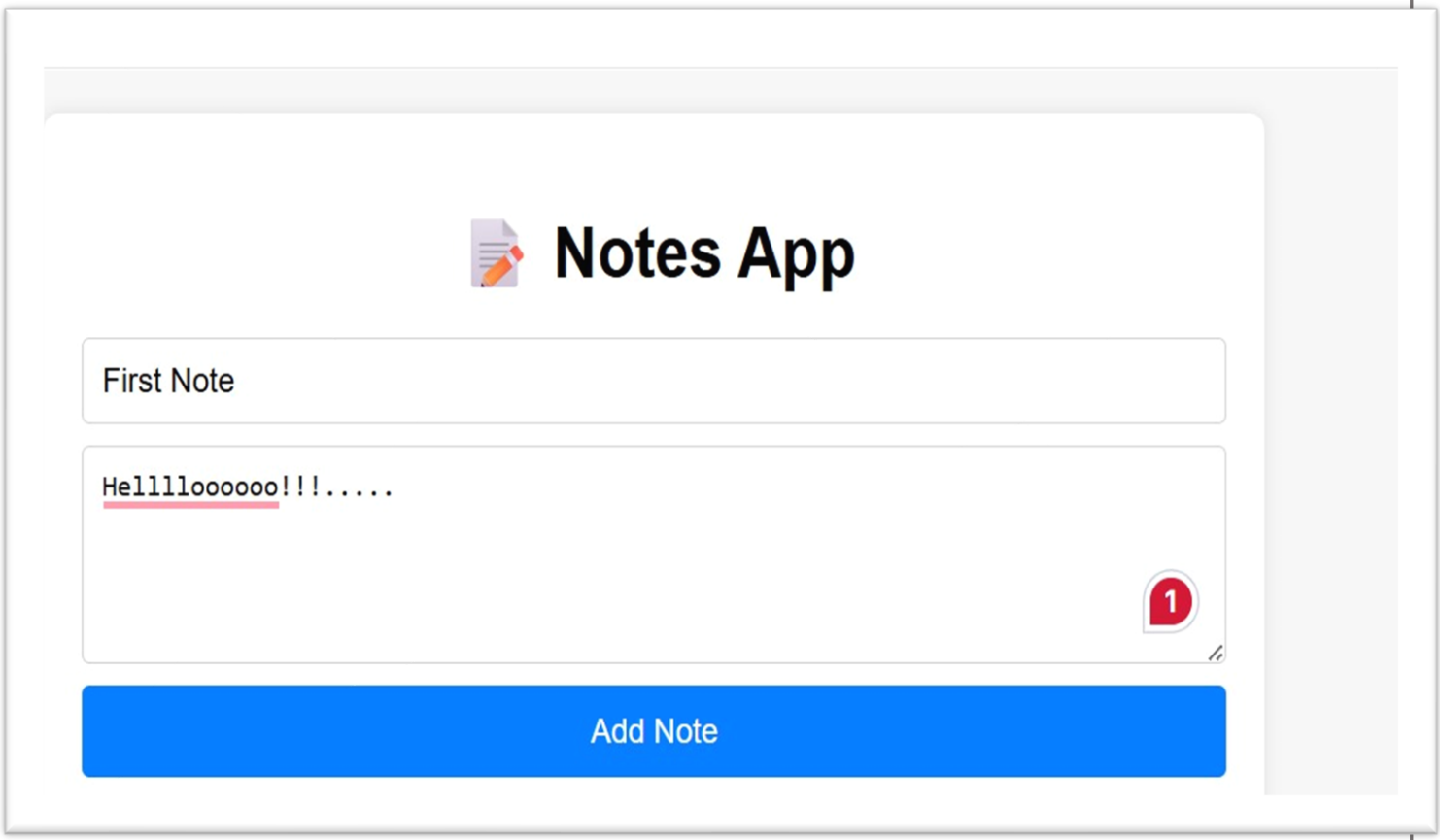
Architecture Summary:



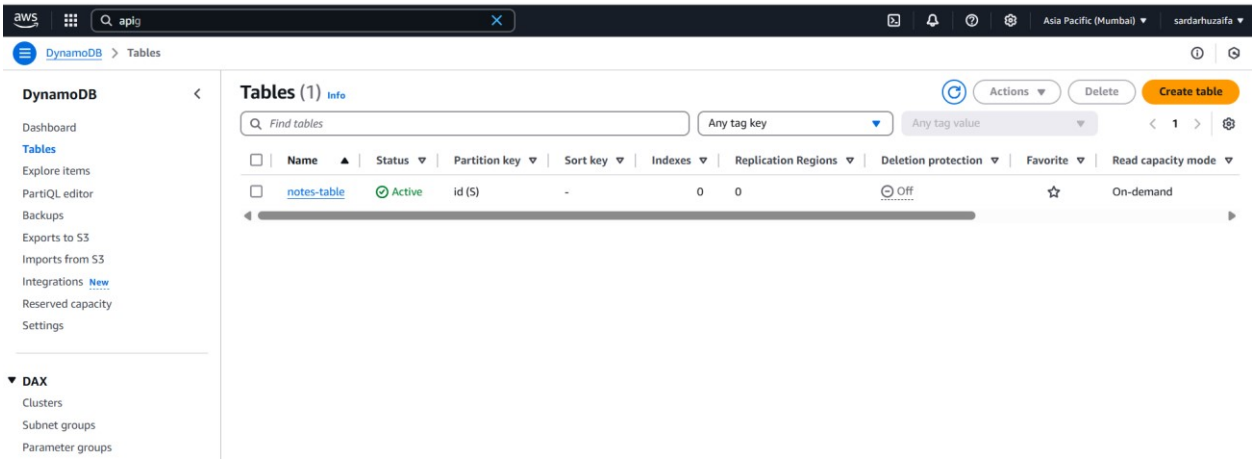
1. Static frontend hosted on **S3**
2. Frontend connects to **API Gateway**
3. **API Gateway** triggers **Lambda functions**
4. **Lambda** interacts with **DynamoDB**
5. Response is returned to frontend

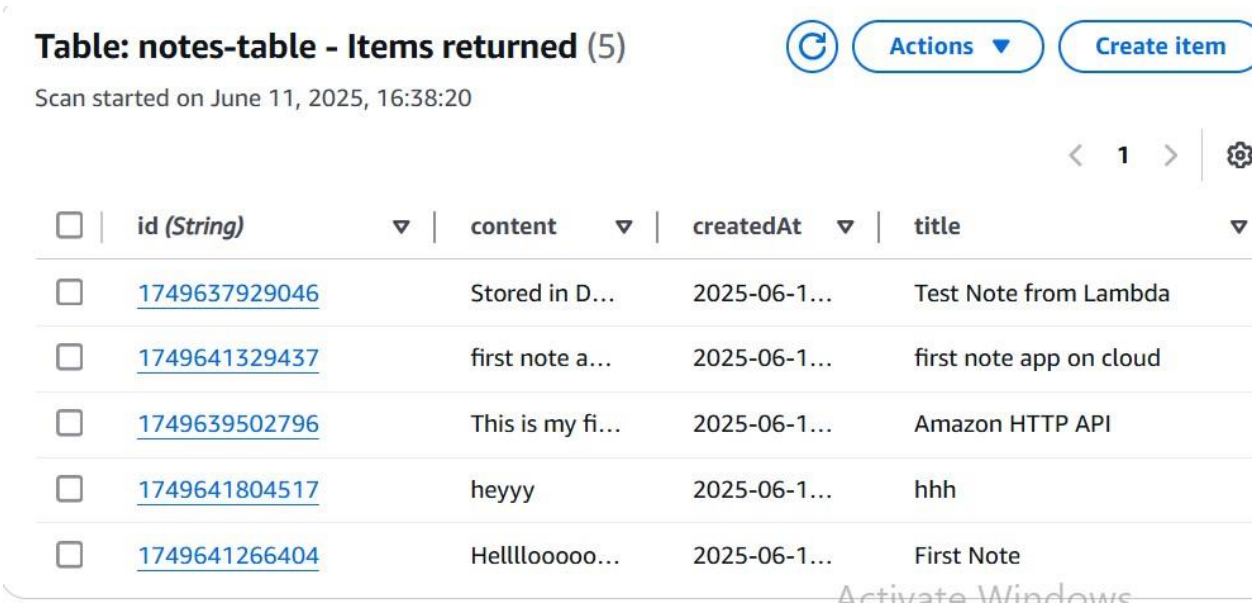
Screenshots:

- **Frontend of Notes Application.**

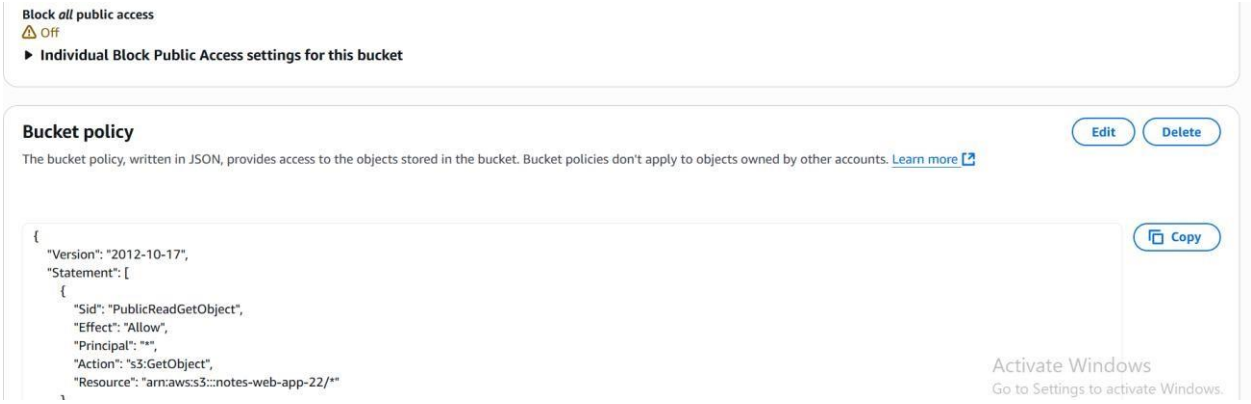


- **Here is the DynamoDB table where our notes have been saved**

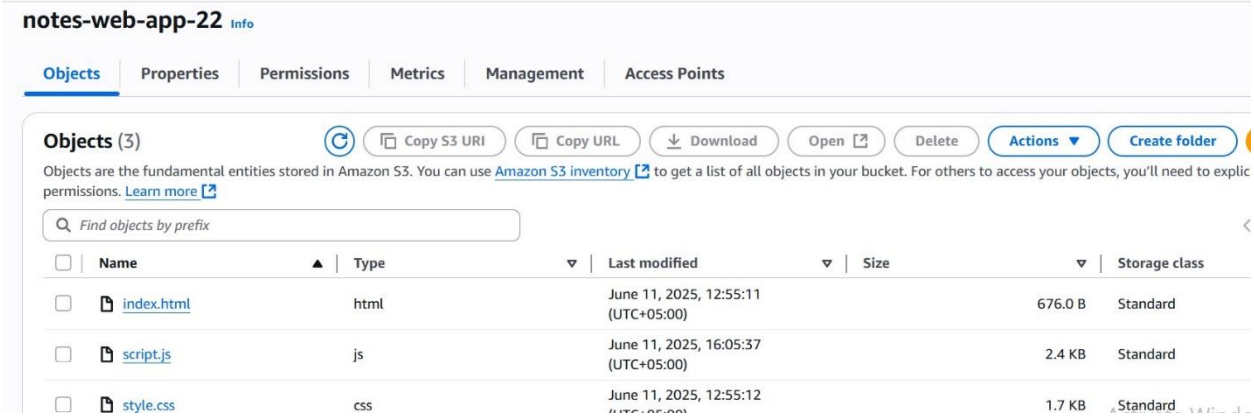




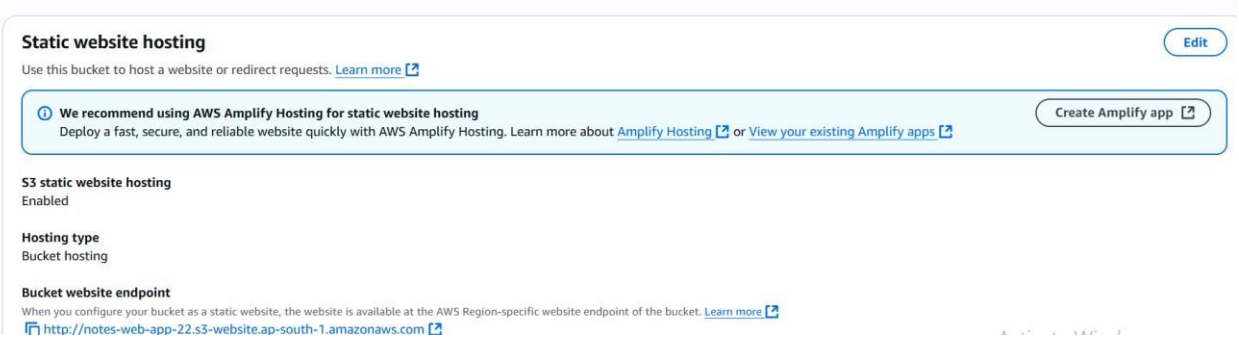
- **S3 Bucket Policy allowing public access.**



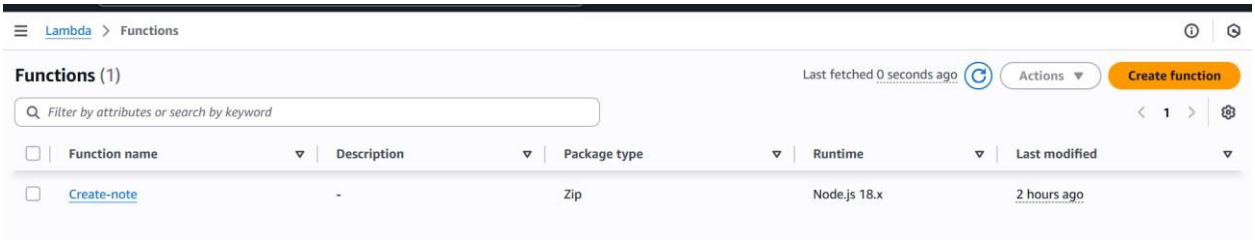
- **Uploaded objects i.e. for front end on S3 for hosting.**



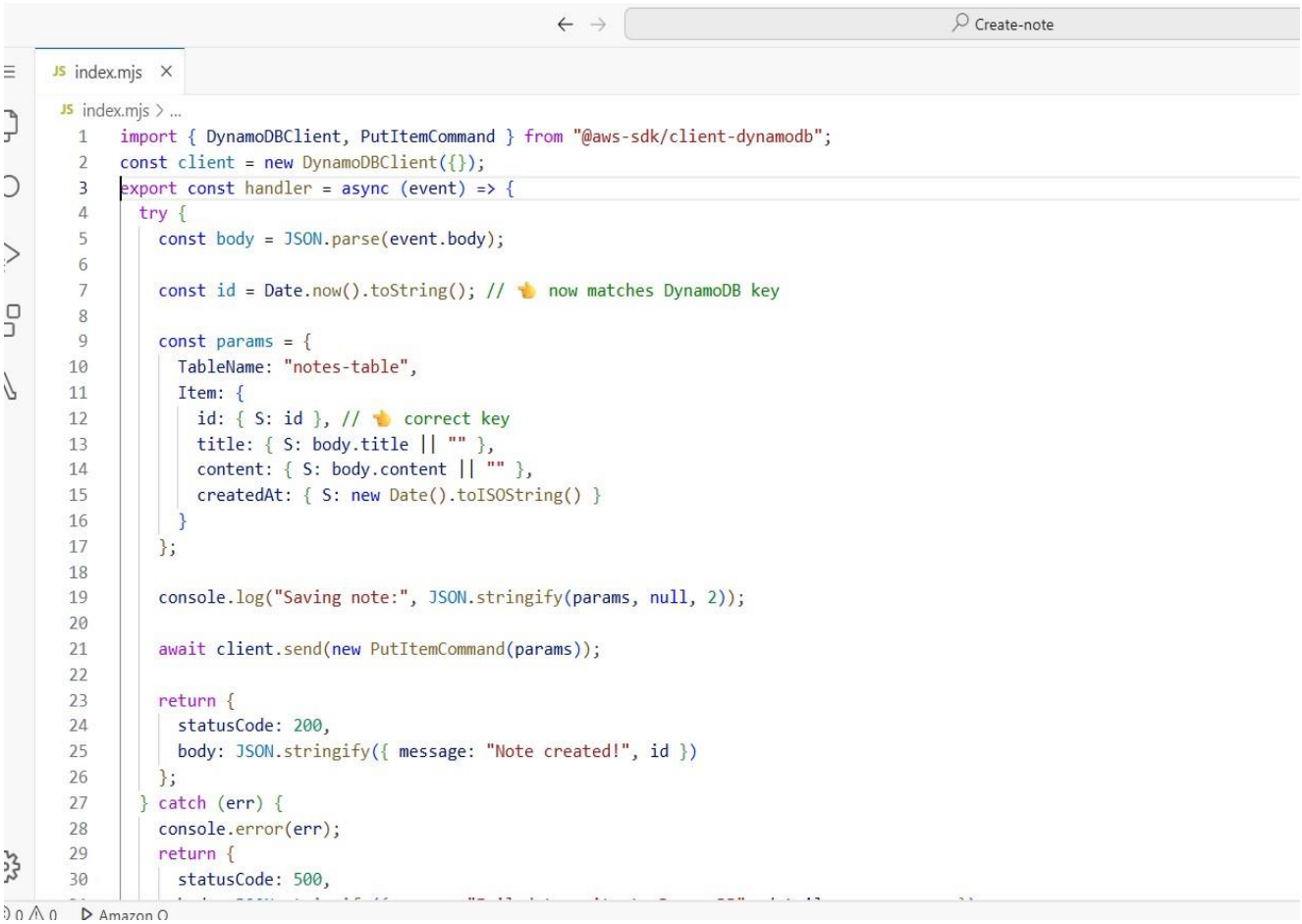
- **Enabling static website hosting on S3.**



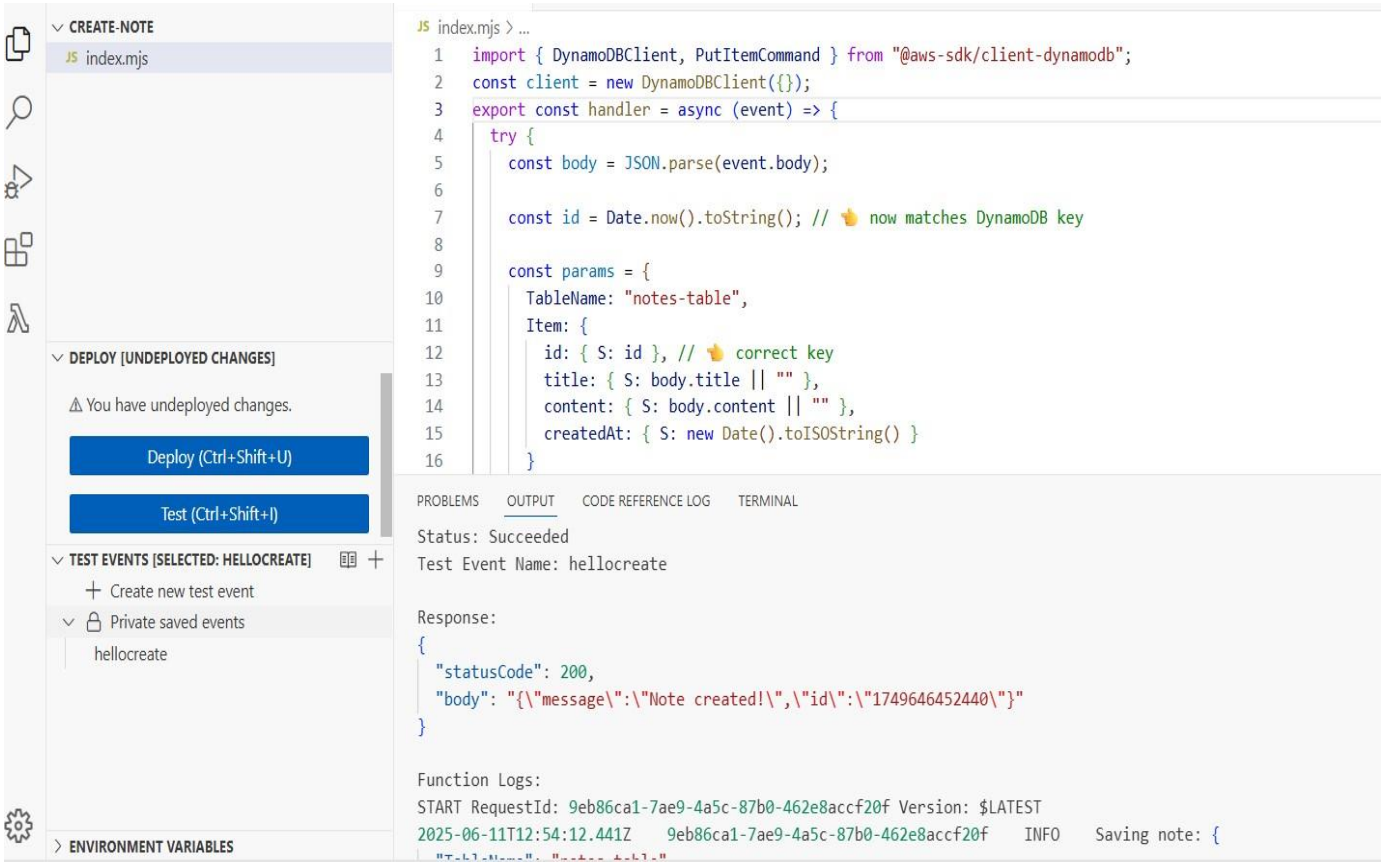
- **Create Note Function for AWS Lambda.**



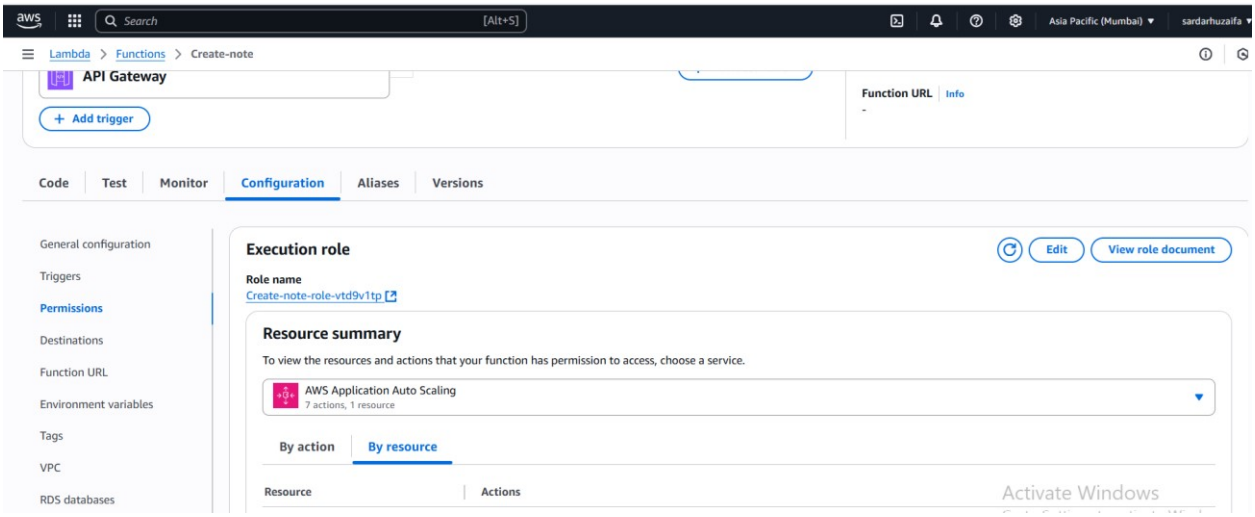
- **The code inside the** Lambda function's `index.mjs` **file is responsible for handling the backend** CRUD operations interacting with DynamoDB



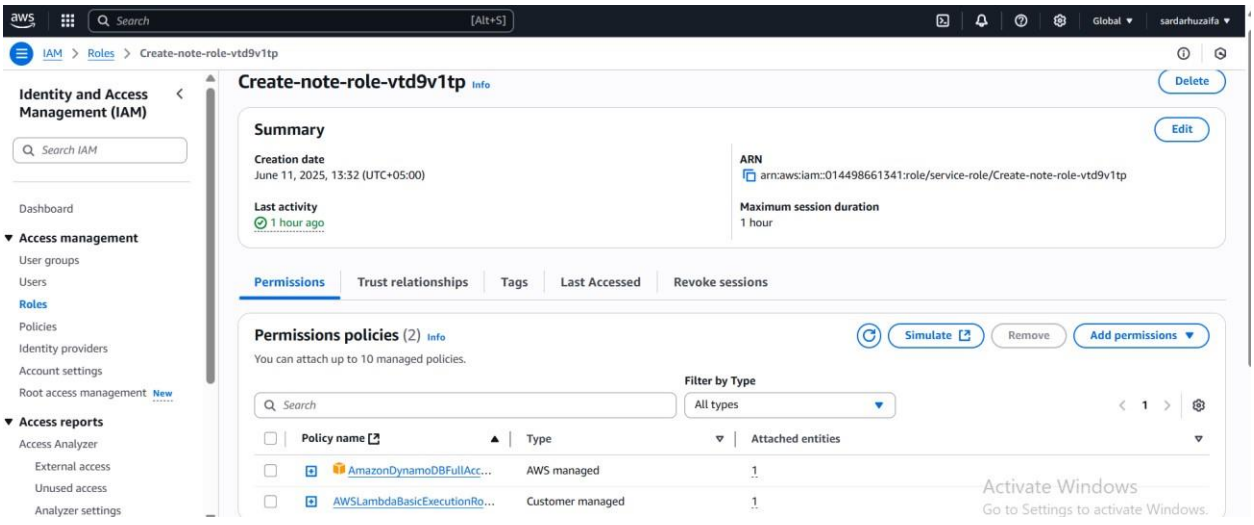
- **It is the test case A Status code of 200 means the request was successfully processed without any errors.**



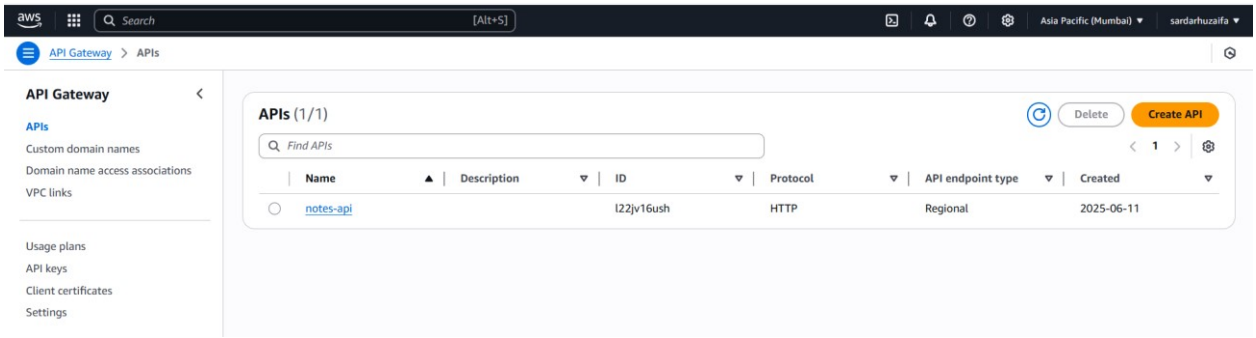
- **We created an AWS IAM Role to allow the Lambda function to securely read from and write to the DynamoDB table.**

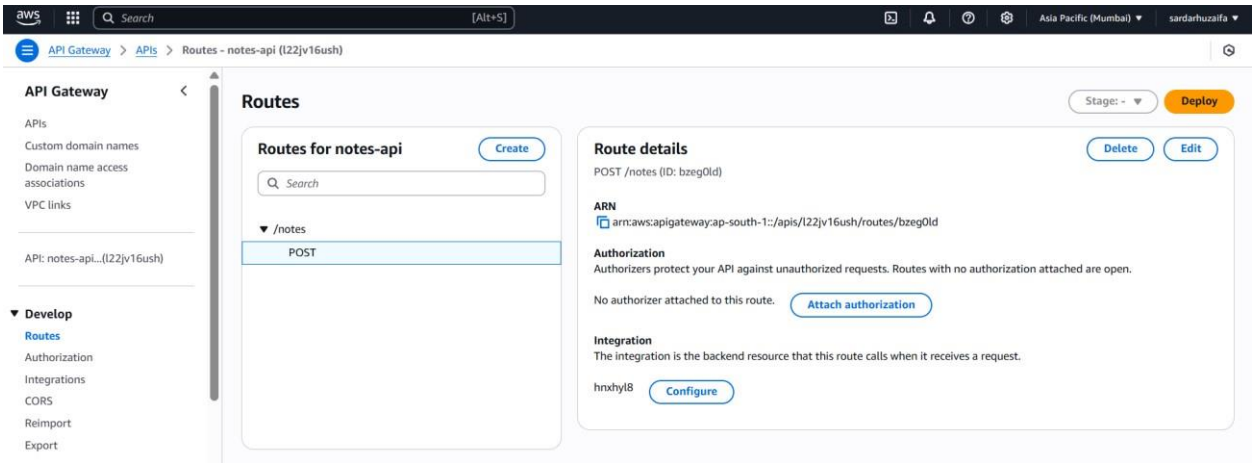


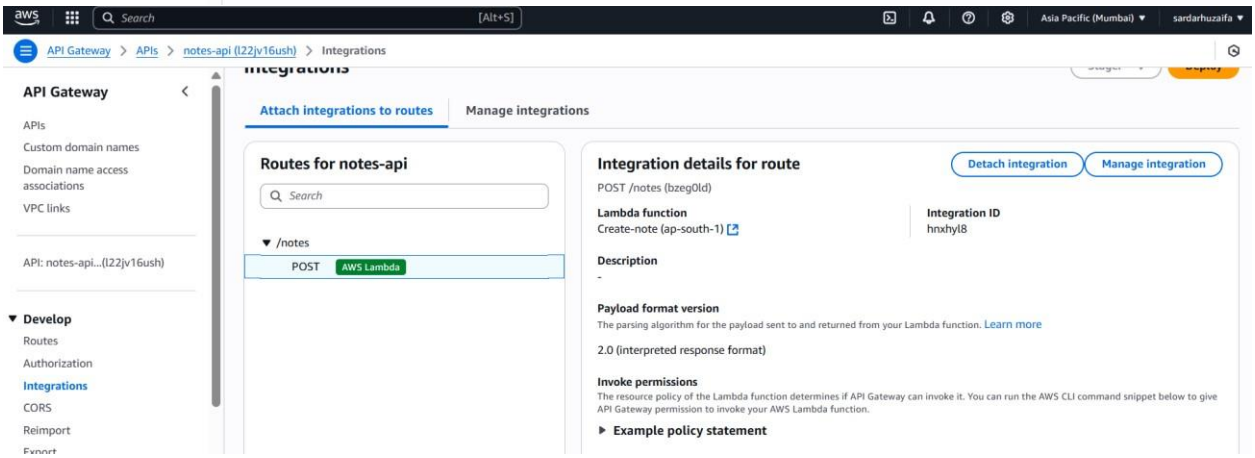
- **Here is the policy for AWS IAM Role allowing AWSDynamoDB Full Access.**



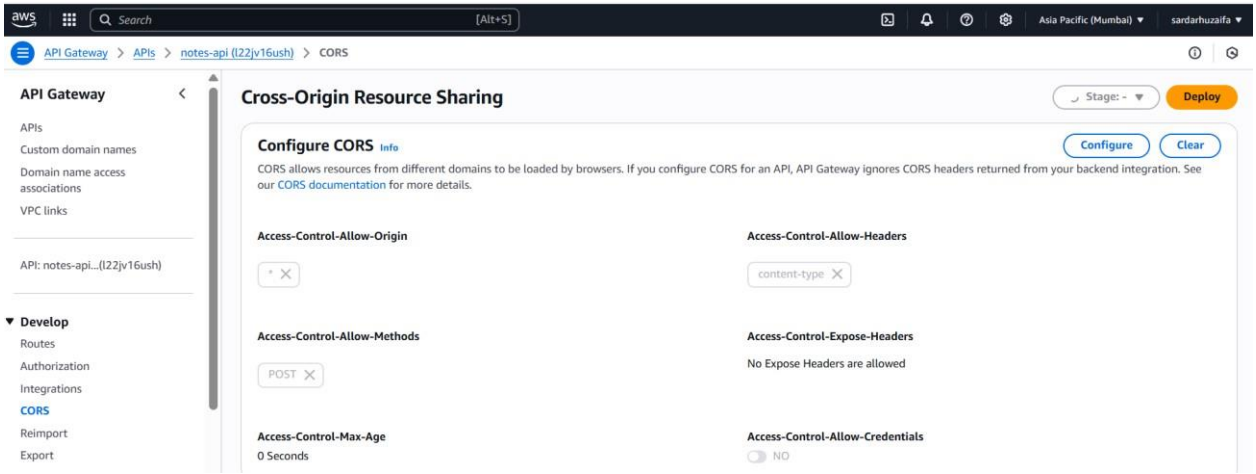
- **Now we have created API for Notes, setting up POST routes and integrating it with AWS Lambda**







- **CORS (Cross-Origin Resource Sharing) was enabled in API Gateway to allow the frontend (hosted on S3) to communicate with the backend (API Gateway) from a different domain.**



- **CloudWatch Logs** were used to monitor and debug Lambda function executions by capturing request and error logs.