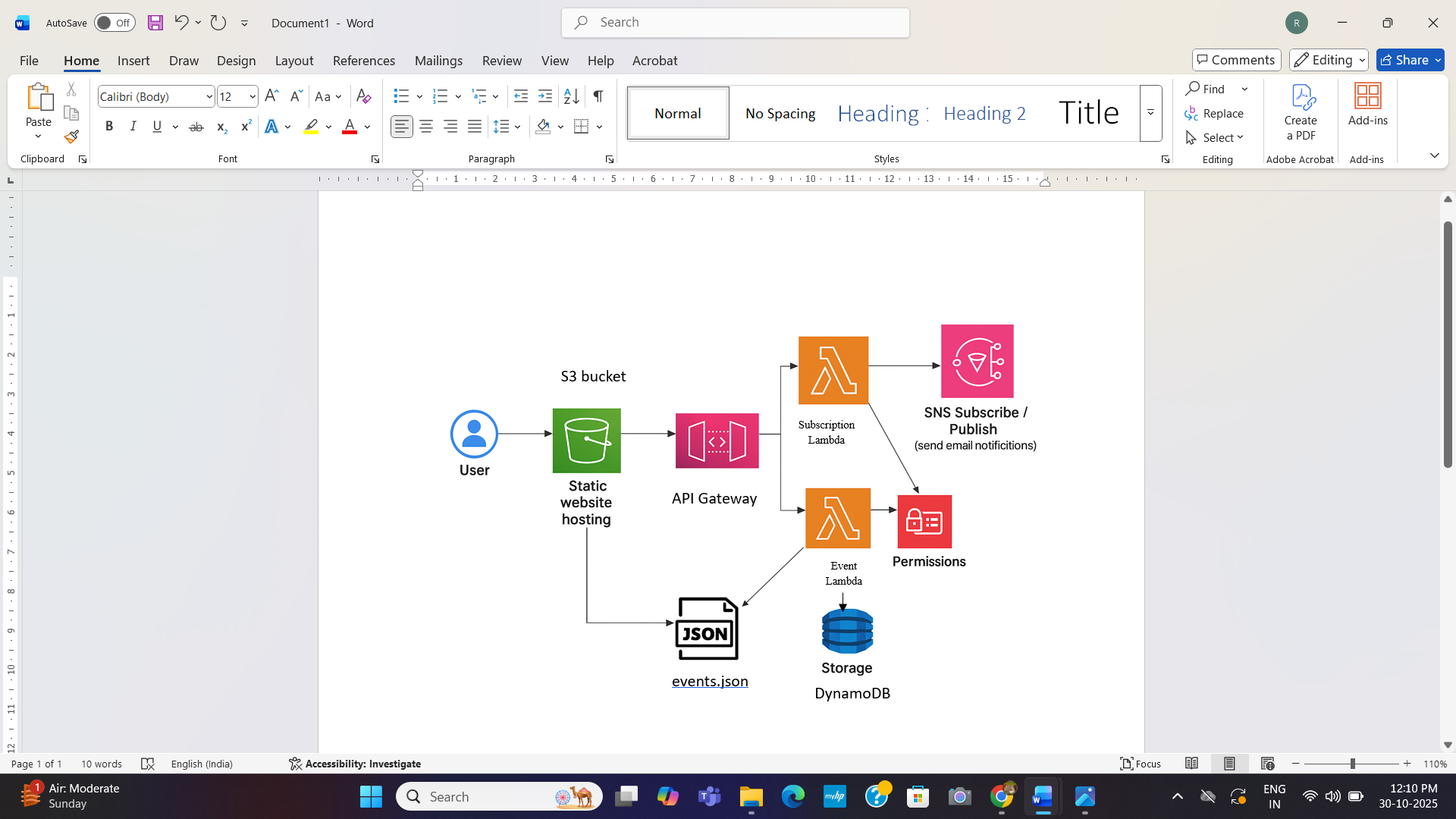
**EVENT ANNOUNCEMENT SYSTEM**

The *Event Announcement System* is a fully serverless web application built using AWS services such as S3, Lambda, API Gateway, SNS, IAM, and DynamoDB. It enables colleges, organizations, or communities to post and announce upcoming events online. Users can subscribe using their email to receive instant notifications about new events. The frontend is hosted on Amazon S3 for static website hosting. AWS API Gateway handles communication between the frontend and backend Lambda functions. Lambda functions process subscriptions, store event details in DynamoDB, and trigger SNS notifications. SNS automatically sends email alerts to all registered users about new event announcements. IAM ensures secure access control for each AWS service and Lambda function. The system eliminates manual event notifications and enhances communication efficiency. It provides a cost-effective, scalable, and maintenance-free solution for real-time event updates.

**Tools used in the project:**

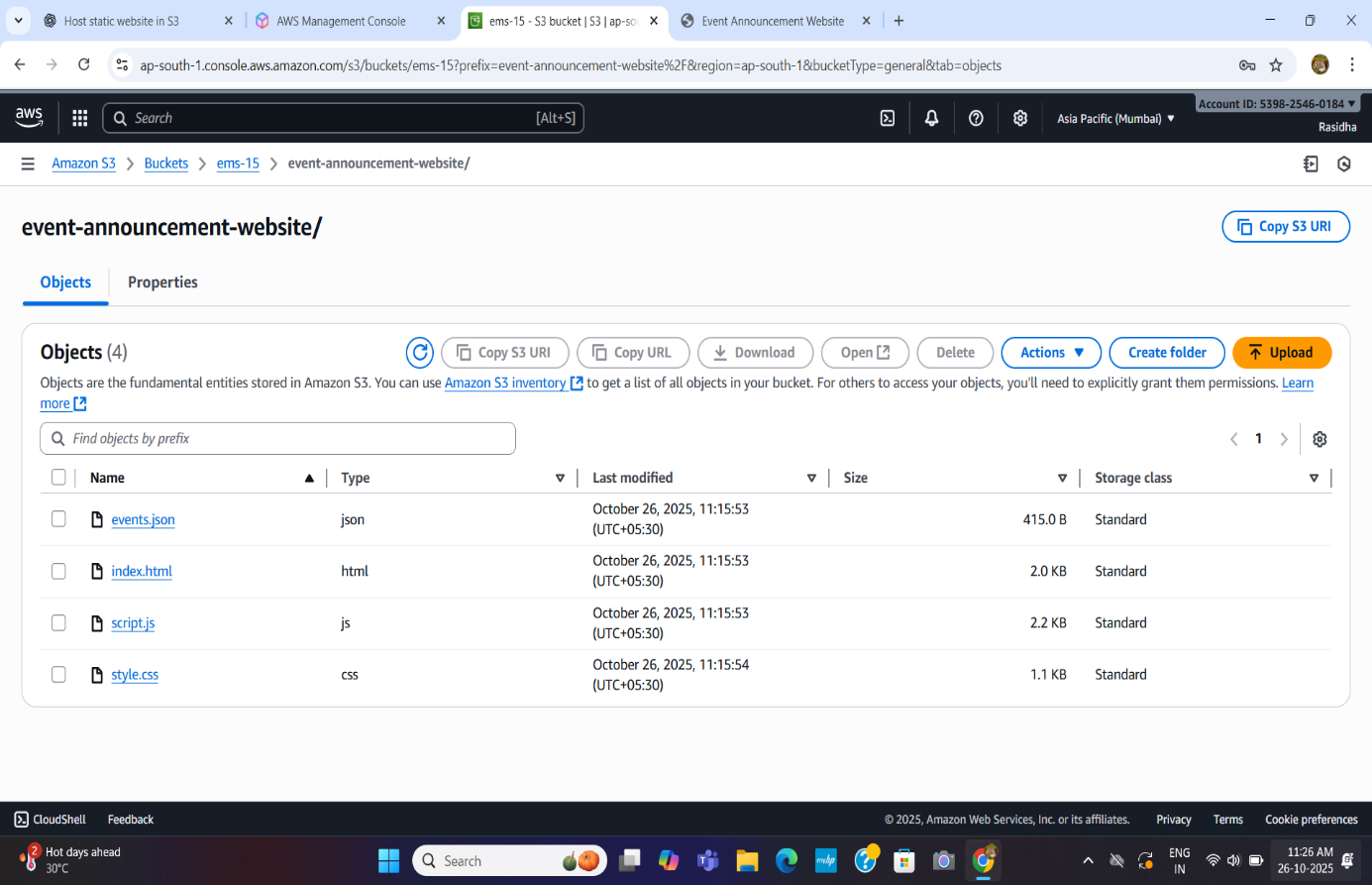
1. **S3:** Hosts the website and event files.
2. **SNS:** Sends event notifications to users.
3. **API Gateway:** Connects frontend with backend.
4. **Lambda:** Runs backend code automatically.
5. **DynamoDB:** Stores event and user data.
6. **IAM:** Controls secure access to AWS services.

**Architectural Diagram:**



**Step 1: Create S3 Bucket**

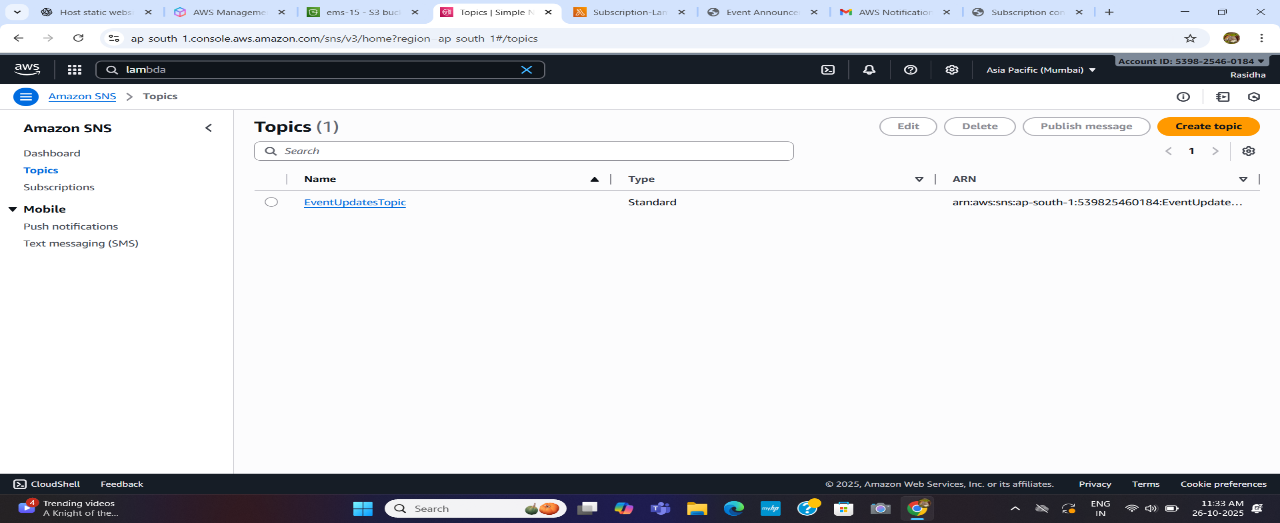
Created an Amazon S3 bucket named ems-15 to host the static website files (index.html, script.js, and events.json). Configured it for public access and enabled static website hosting.

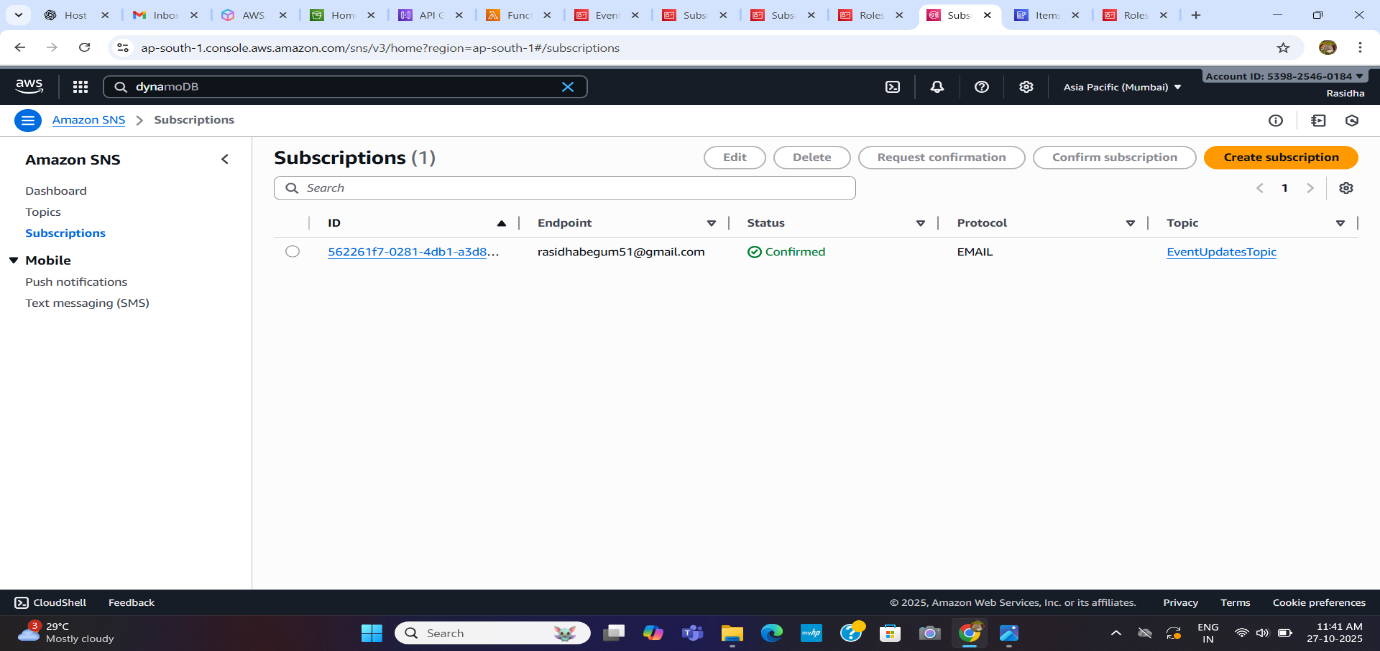


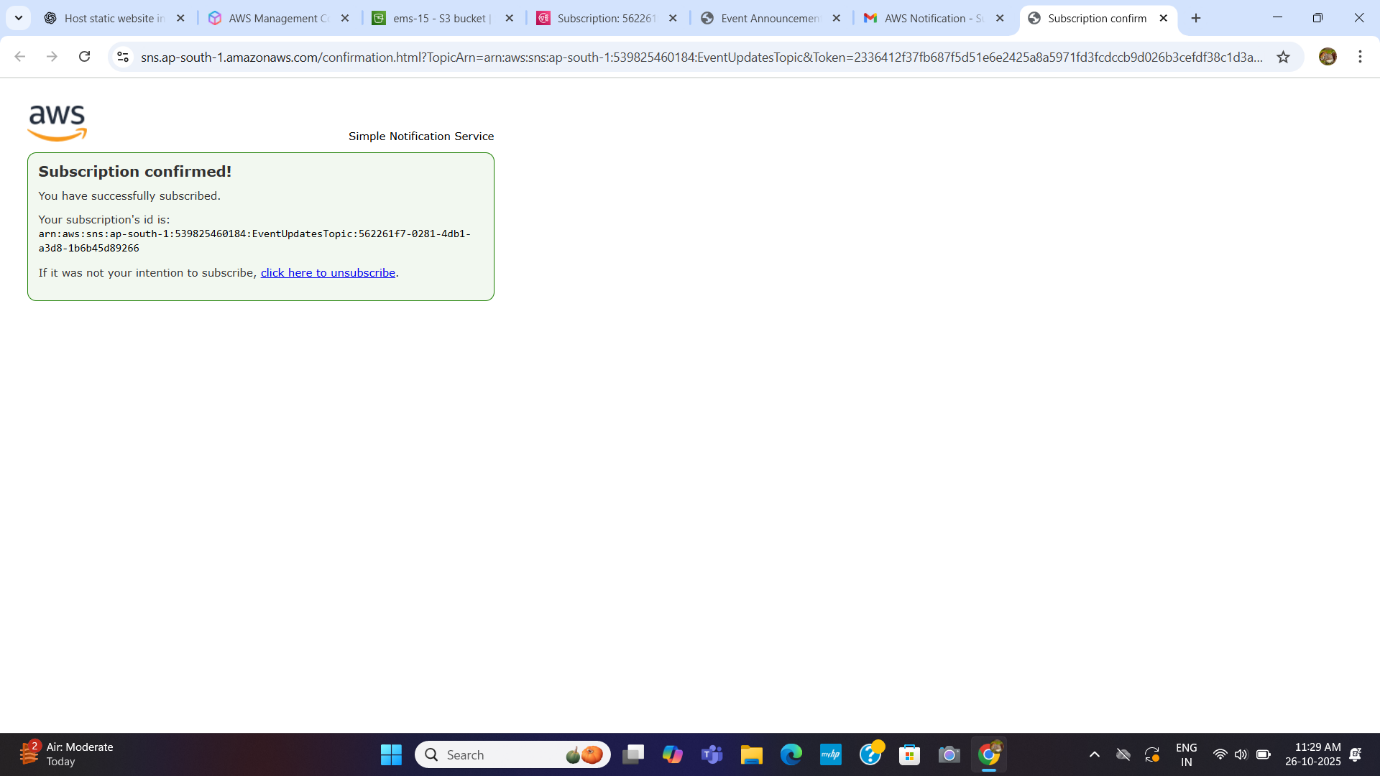


**Step 2: Create SNS Topic**

Created an SNS topic named EventUpdatesTopic to send notifications to subscribed users whenever a new event is added.

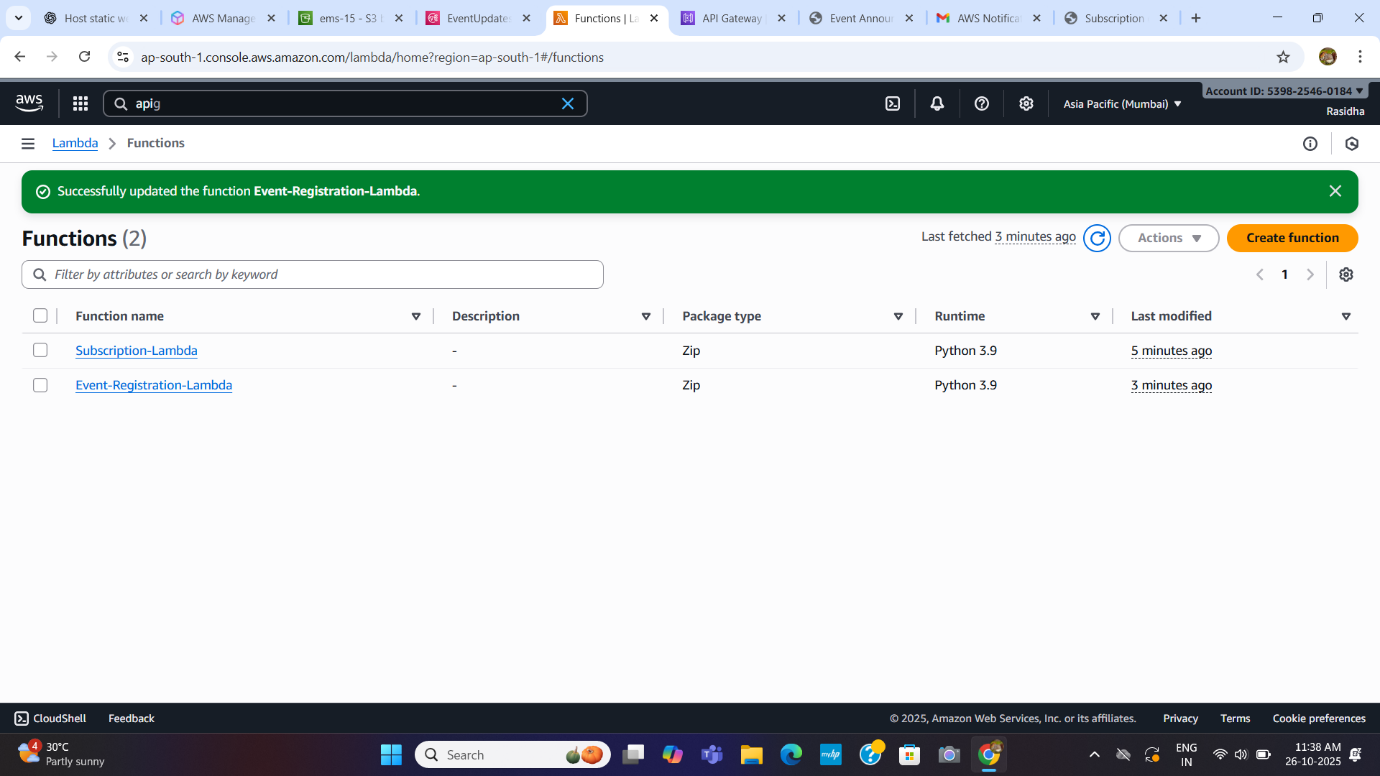




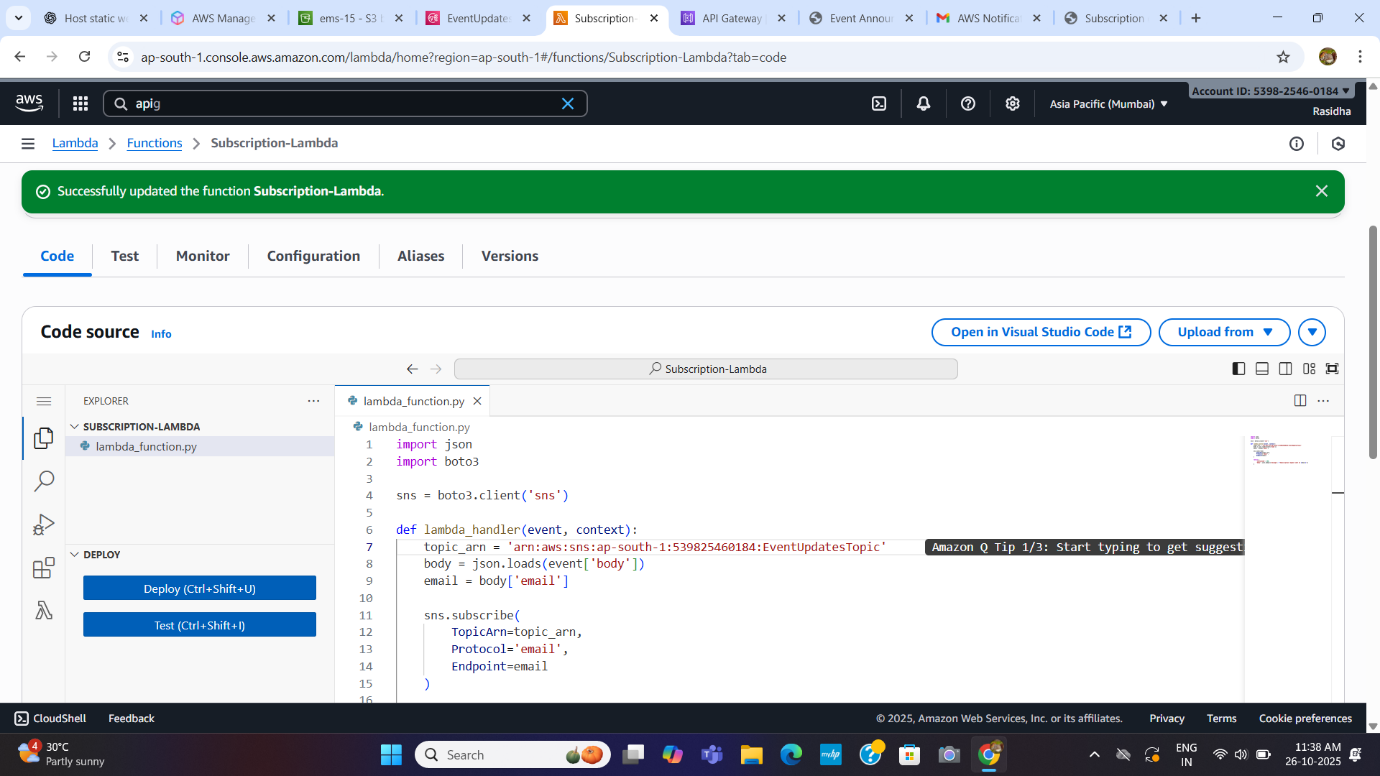


**Step 3: Create Lambda Functions**

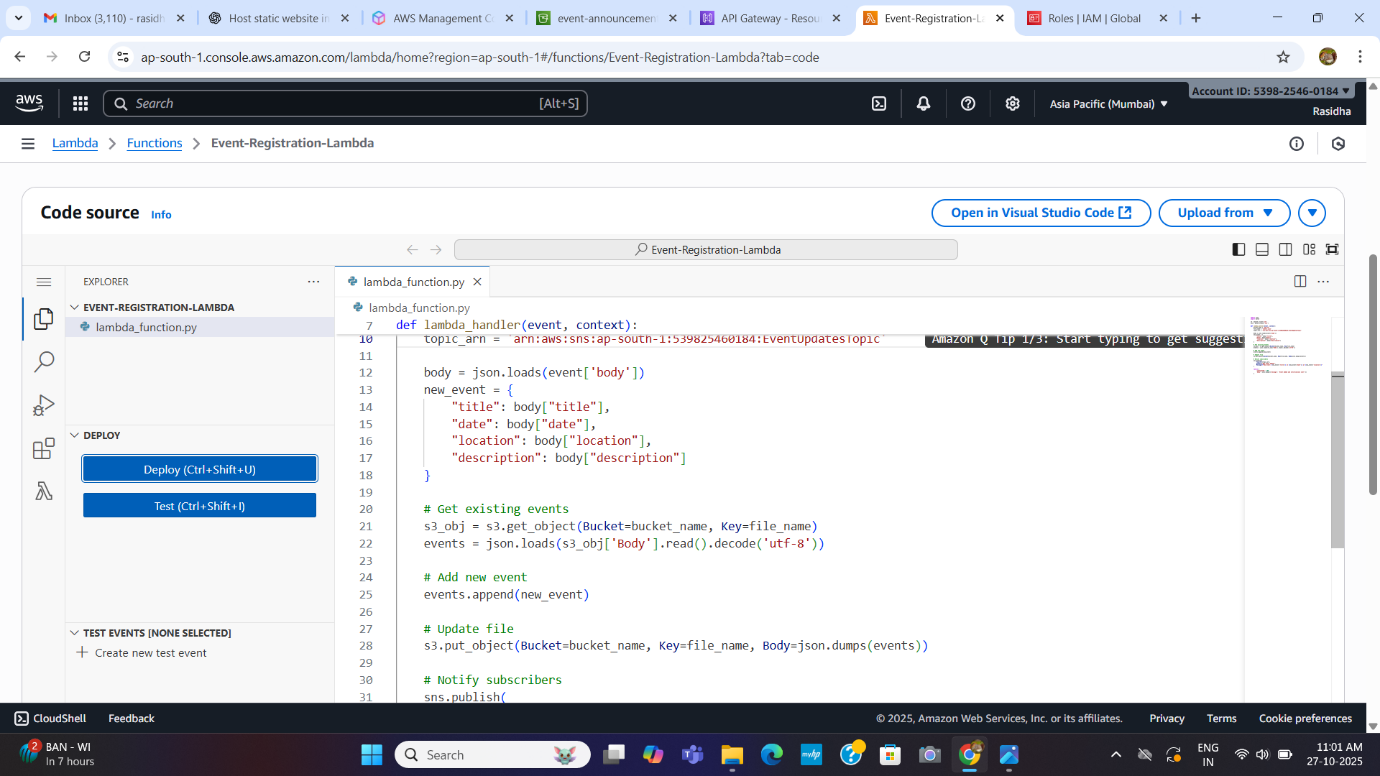
Created two Lambda functions — Subscription-Lambda (handles email subscriptions) and Event-Lambda (adds new events and publishes messages to SNS).



Code for Subscription Lambda

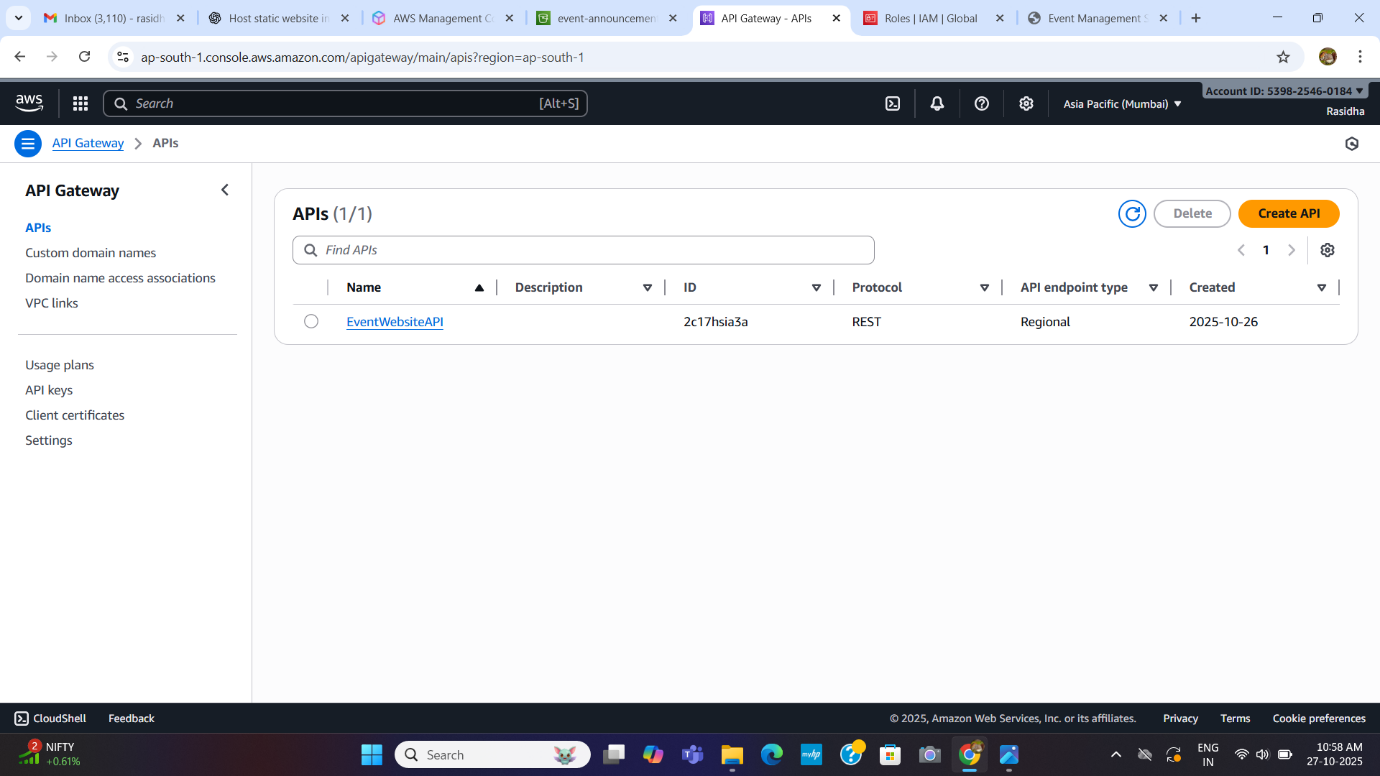


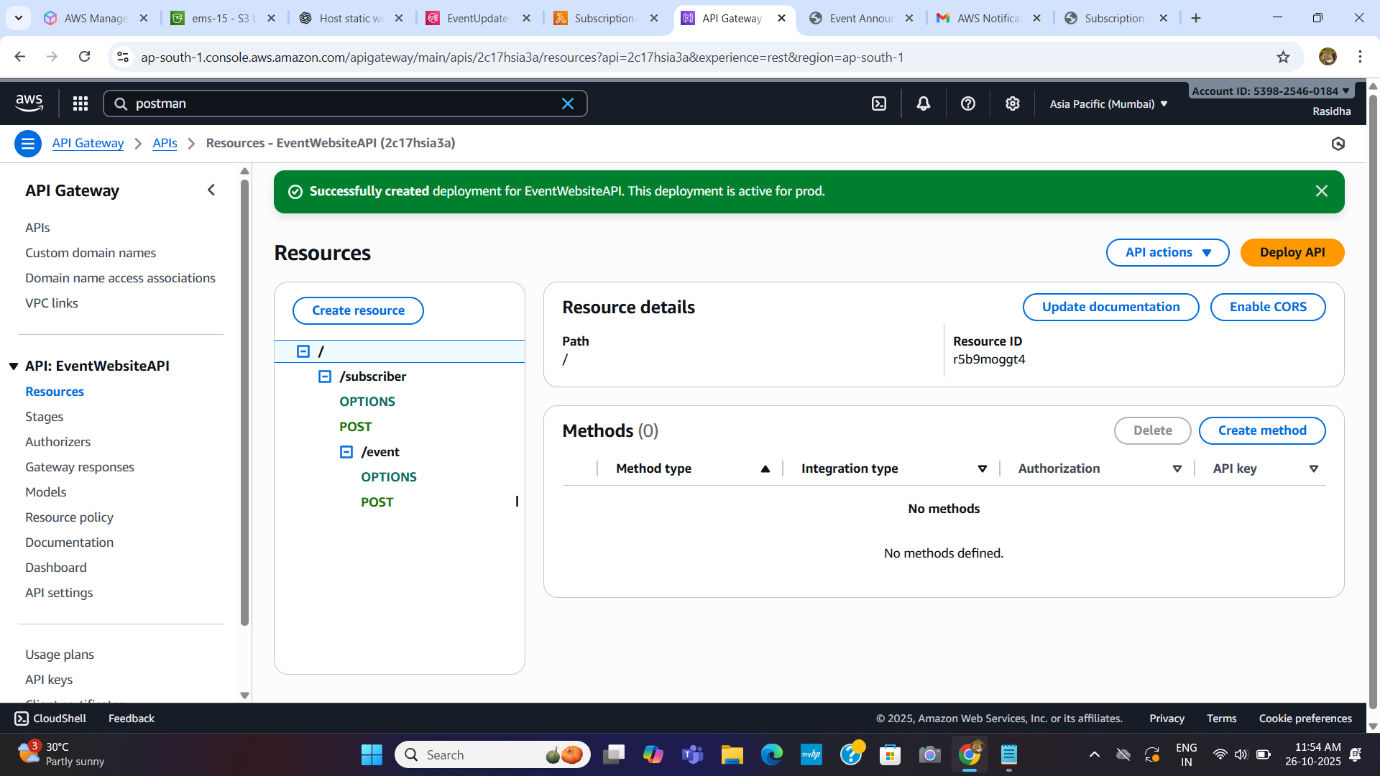
Code for Event Lambda



**Step 4: Configure API Gateway**

Created a **REST API** with /subscriber and /event resources.

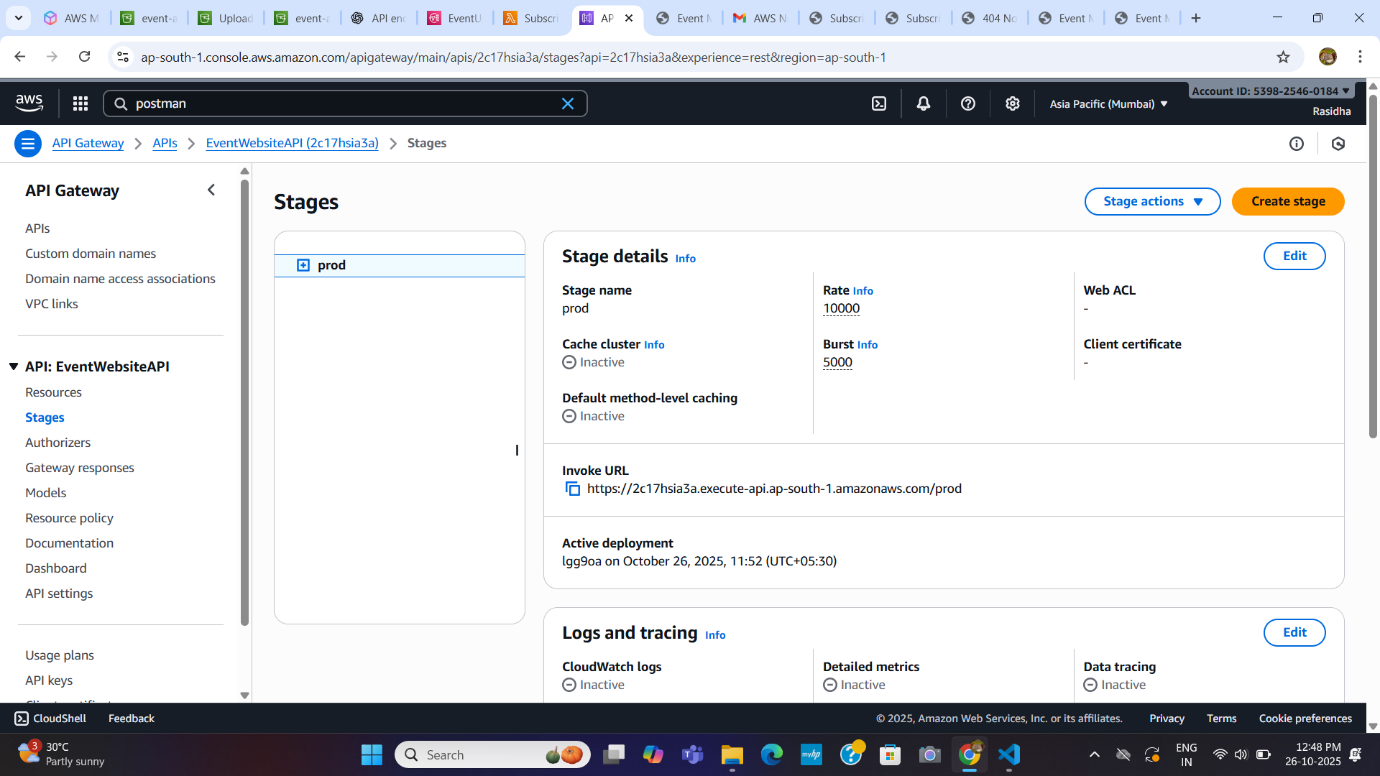




**API Gateway** configuration where a **POST request** to /subscriber/event triggers a **Lambda function** to handle event subscription processing.

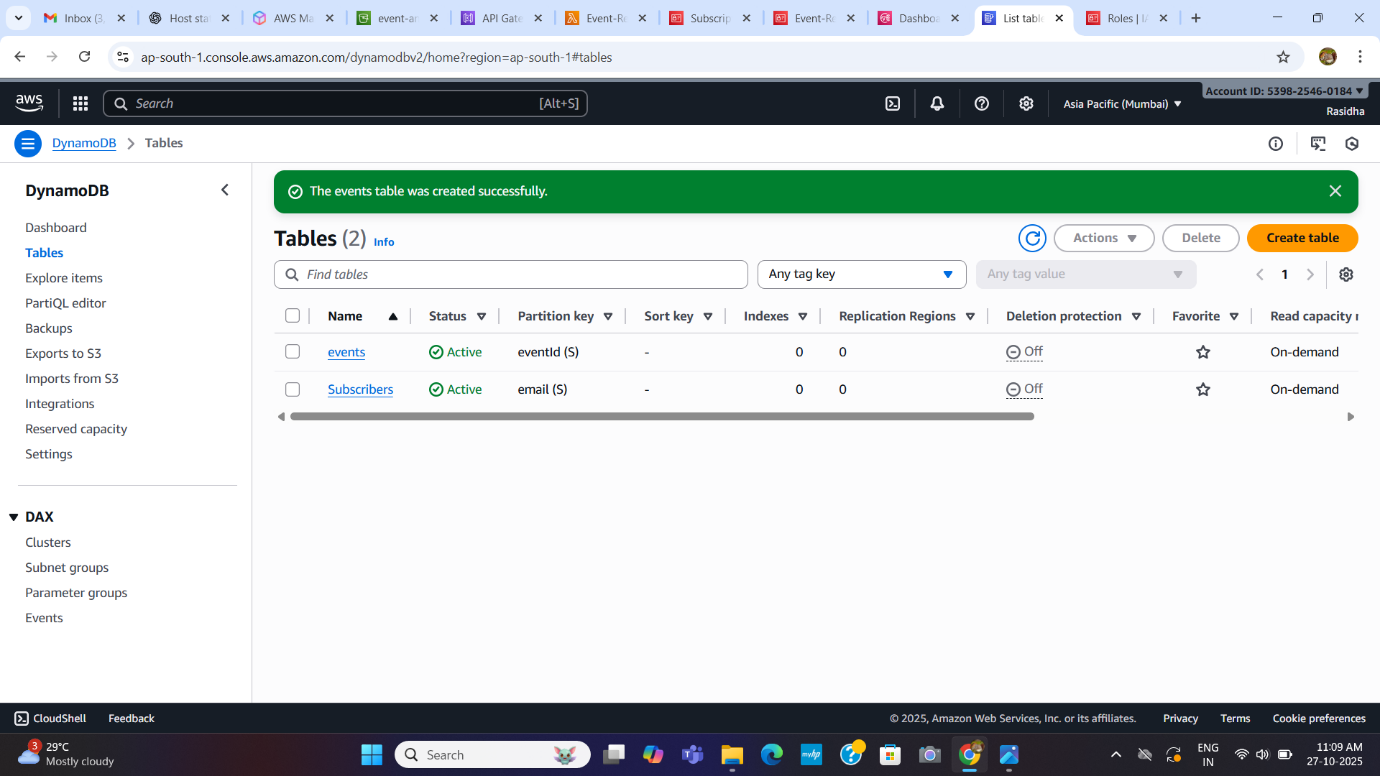


Connected each resource to its respective Lambda function and deployed the API to the prod stage.



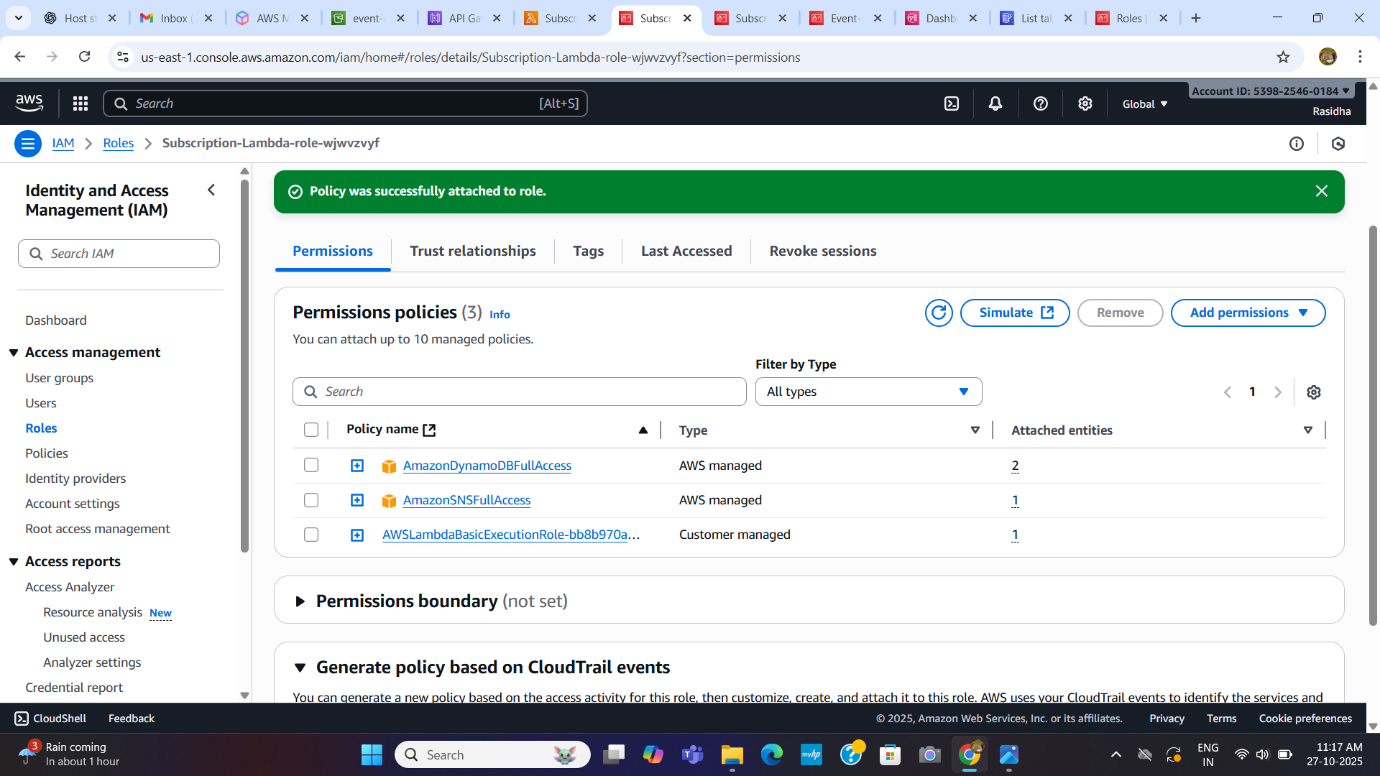
**Step 6: Add DynamoDB Integration**

Created a DynamoDB table named events to store all event data like event name, date, and description. Updated the Lambda code to insert and fetch records from this table.



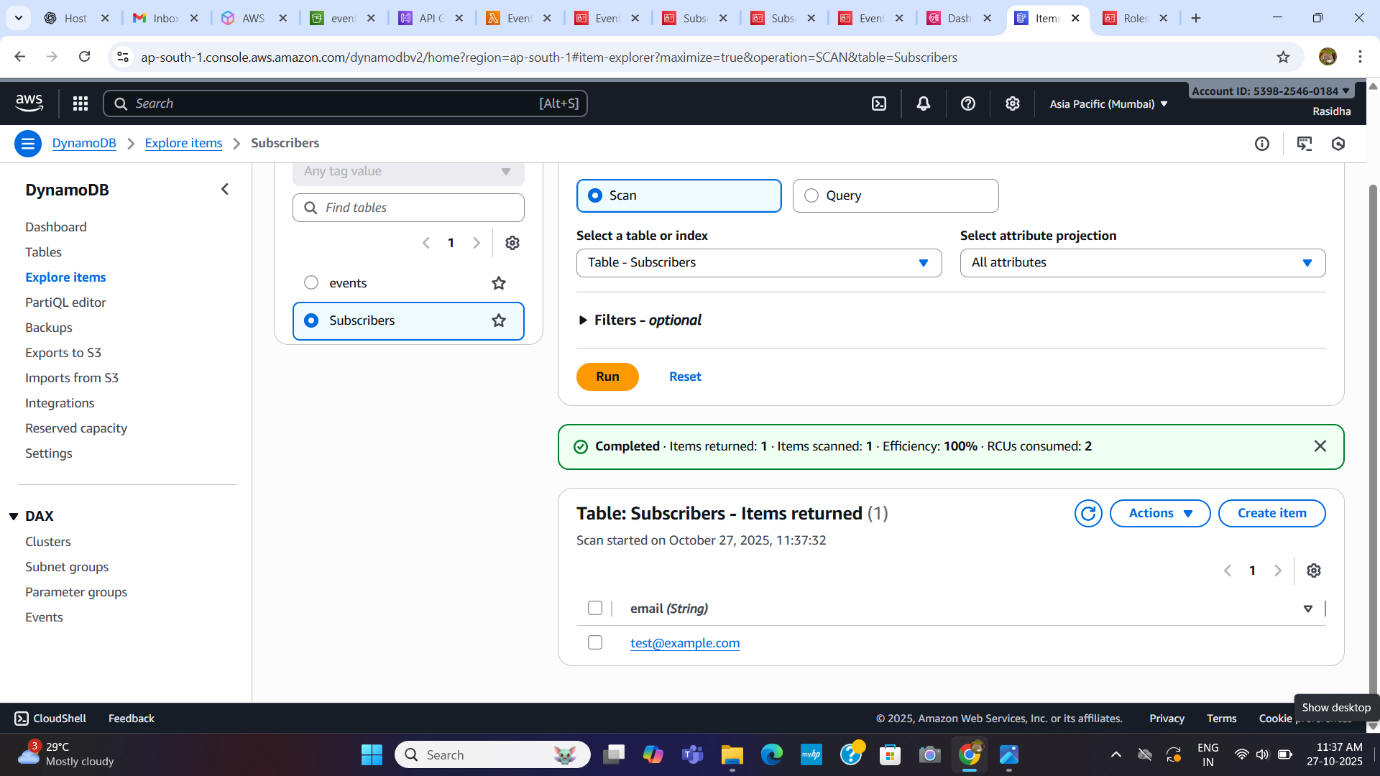
**Step 7: Add IAM permissions**

**IAM role** configuration where permissions for **DynamoDB, SNS, and Lambda execution** are attached to allow secure access between AWS services.



**Step 8: Test End-to-End Application**

Tested the full workflow — subscribing users, adding new events, verifying data in DynamoDB, and receiving email notifications via SNS.



**Step 9: Results**

Verified successful notifications, API responses (status 200), and data stored in DynamoDB. The project demonstrates integration of multiple AWS services in a serverless event management workflow.

