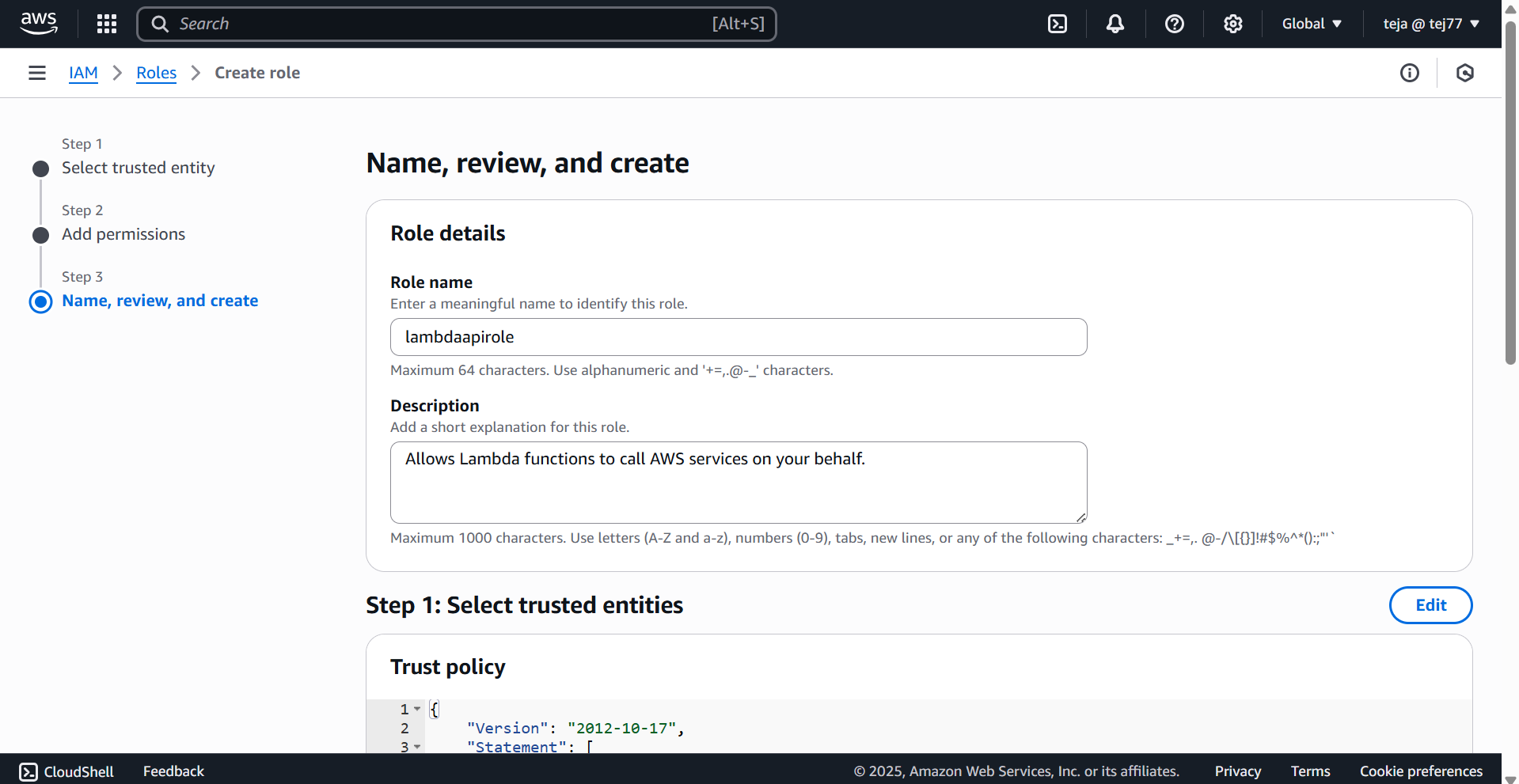
**Project Overview: Serverless Architecture using API Gateway, Lambda, and DynamoDB**

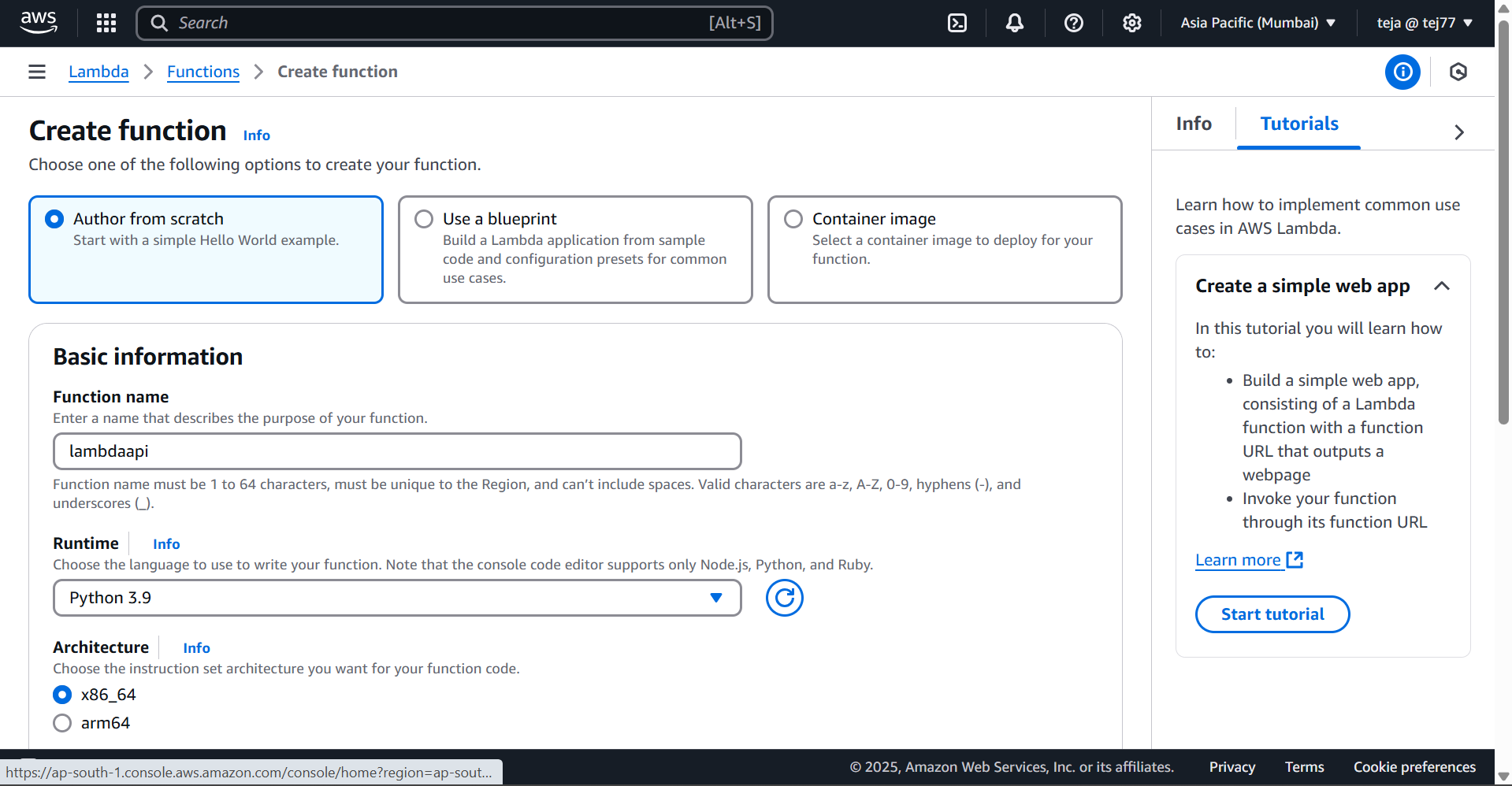
**Introduction**

This document provides a brief overview of a serverless project that uses three AWS services: **API Gateway**, **Lambda**, and **DynamoDB**. Each section includes one key setup step along with an explanation of the service, why it was chosen, and its advantages.

1. **Create IAM role  
   Aws Lambda Execution role**

**DynamoDB full access  
  
  
A screenshot of a computer

AI-generated content may be incorrect.**

**2.Open lambda create function select python 3.9 select existing role and attach the role we have created on step one  
  
  
A screenshot of a computer

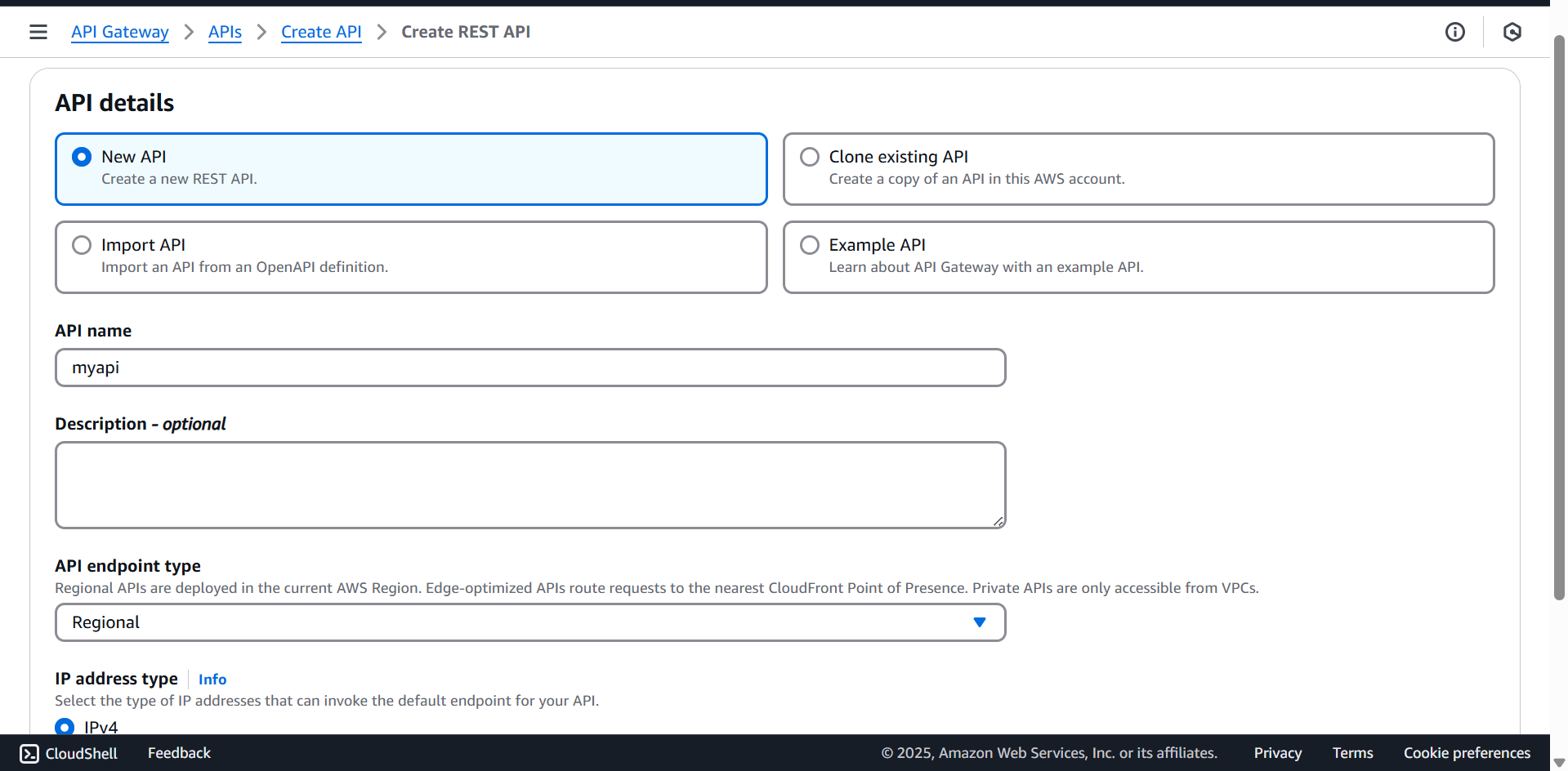
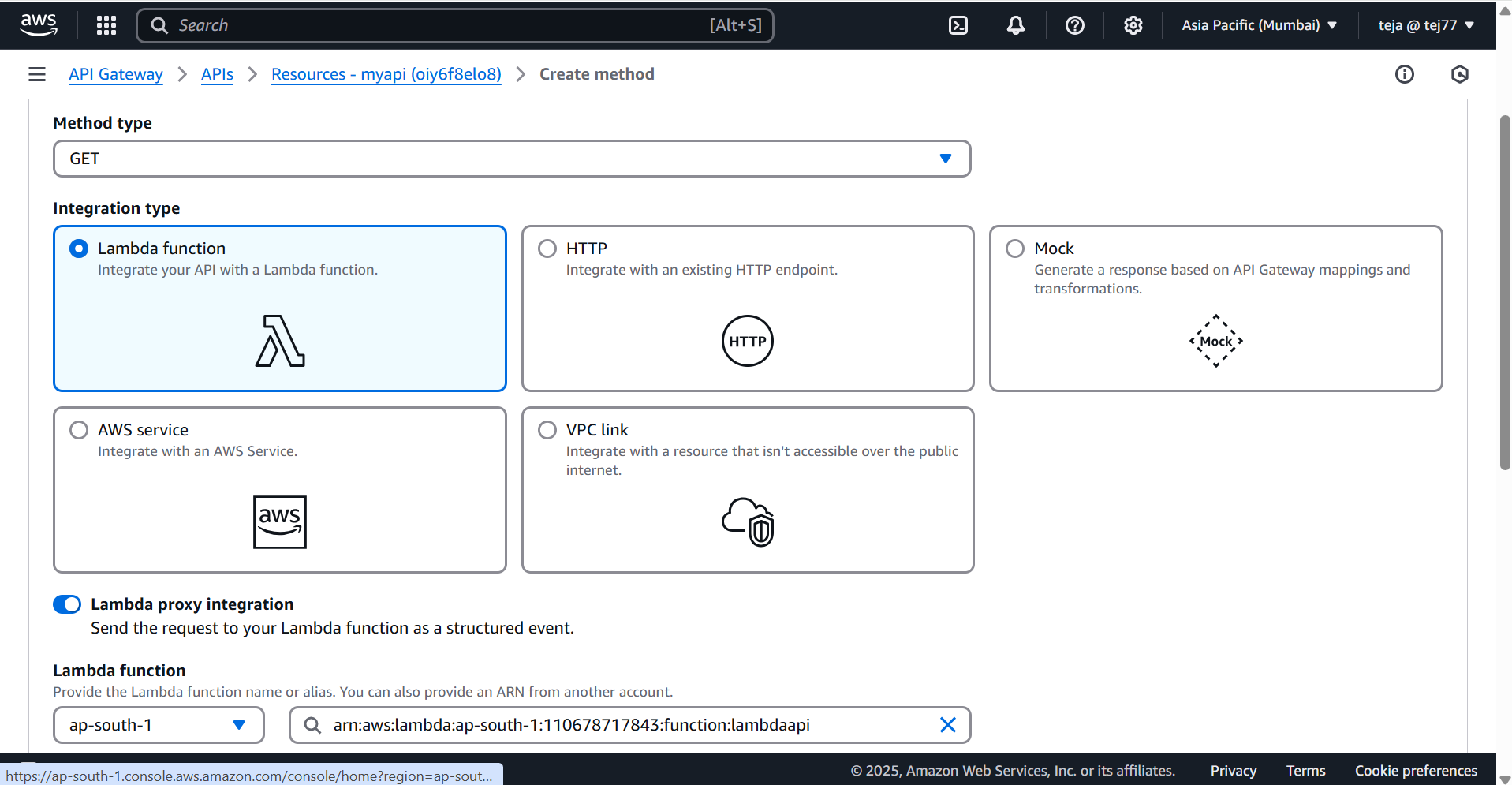
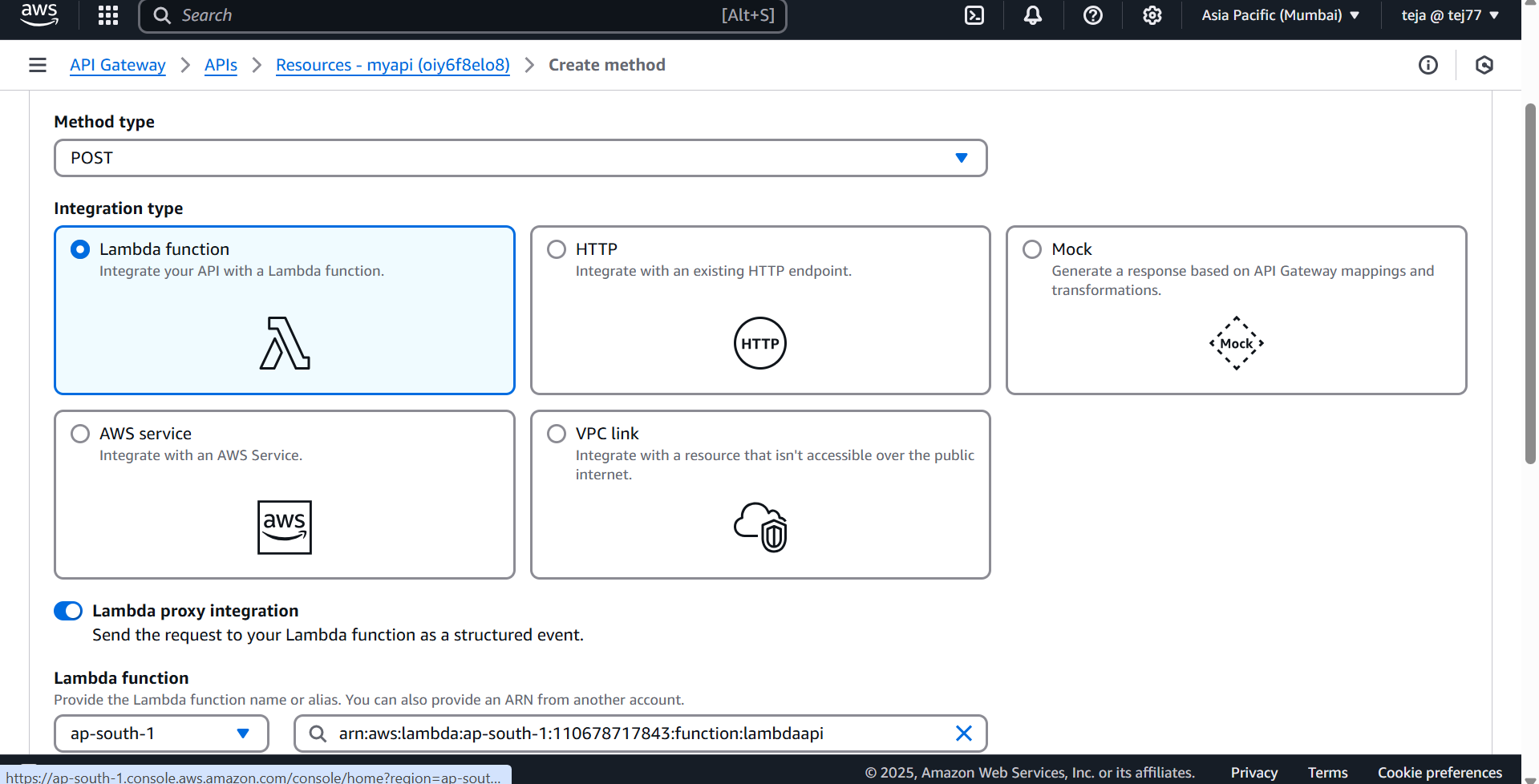
AI-generated content may be incorrect.**

**3. take the lambda function and make some changes like Dynamo table name and zip all the file and upload in lambda.A screenshot of a computer

AI-generated content may be incorrect.**

**4.create one table in DynamoDB and select partitions key has Email and value as string and replace the name in lambda code with which you have created the name with Dynamo table name   
A screenshot of a computer

AI-generated content may be incorrect.**

**5. Go to Rest API and select to create API like new API give API name and API endpoint has region after creating API the in API   
  
  
6.after creating API need to create methods get and post after creating methods need to add deploy and create a new stage like dev and PROD after creating take URL of the created stage you have created   
  
  
  
  
A screenshot of a computer screen

AI-generated content may be incorrect.  
  
A screenshot of a computer

AI-generated content may be incorrect.**A screenshot of a computer

AI-generated content may be incorrect. **A screenshot of a computer

AI-generated content may be incorrect.**

**✅ Module 1: API Gateway (Front-End)**

**🔹 What is the service?**

Amazon API Gateway is a fully managed service that allows developers to create, publish, maintain, and monitor REST, HTTP, and WebSocket APIs.

**🔹 Why was this service used?**

API Gateway is used to expose the Lambda function to users through a RESTful API, acting as the front-end interface for both GET and POST requests.

**🔹 Service Description:**

In this project, API Gateway handles incoming HTTP requests. It serves the "Contact Us" page for GET requests and forwards form data to the Lambda function for POST requests.

**🔹 Service Advantages:**

* Serverless and fully managed
* Scalable with built-in traffic management
* Supports throttling, authorization, and monitoring
* Easily integrates with Lambda and other AWS services

**📸 Key Step: Create REST API Method (GET)**

**✅ Module 2: AWS Lambda (Server/Compute)**

**🔹 What is the service?**

AWS Lambda is a serverless compute service that runs backend code in response to events, without provisioning or managing servers.

**🔹 Why was this service used?**

Lambda was used to process form data submitted from the front end and to interact with the DynamoDB table in the backend.

**🔹 Service Description:**

The Lambda function receives requests from API Gateway. On a POST request, it reads the form data, inserts it into the DynamoDB table, and returns a success page.

**🔹 Service Advantages:**

* No server management required
* Automatic scaling and high availability
* Integrated with many AWS services
* You only pay for what you use

**📸 Key Step: Lambda Code to Process POST Request**

*This Lambda function handles POST data and writes it to DynamoDB.*

**✅ Module 3: Amazon DynamoDB (Back-End/Database)**

**🔹 What is the service?**

Amazon DynamoDB is a fully managed NoSQL database service that offers fast and predictable performance with seamless scalability.

**🔹 Why was this service used?**

DynamoDB was chosen to store user-submitted data (name, email, message) from the form due to its speed, scalability, and serverless nature.

**🔹 Service Description:**

The table stores the data submitted through the form, using **email** as the primary key to ensure uniqueness of records.

**🔹 Service Advantages:**

* Fully managed NoSQL database
* Single-digit millisecond response times
* Serverless with automatic scaling
* Fine-grained access control via IAM

**📸 Key Step: Create DynamoDB Table with Email as Partition Key**

*This table will store the form submission data.*

**✅ Conclusion**

This project demonstrates how to build a fully serverless web application using **API Gateway**, **Lambda**, and **DynamoDB**. The user interacts with a form, data is processed in the cloud without managing any servers, and everything is stored securely in a scalable NoSQL database.

* API Gateway handles and routes web requests.
* Lambda runs backend logic on demand.
* DynamoDB stores submitted data efficiently.

This architecture is lightweight, scalable, and cost-effective, ideal for modern cloud-native applications.  
  
Reference video https://www.youtube.com/watch?v=dsH2QC6O3Gg&list=PLneBjIzDLECkz7SwM-HFqZsvAZDz4hcpq&index=4

Created by – Siva Teja