**Serverless Web Application Deployment with AWS Lambda, API Gateway, and DynamoDB**

### ****Project Overview:****

This project demonstrates the deployment of a web application using **AWS Lambda**, **API Gateway**, and **DynamoDB** in a serverless architecture. The application provides a simple contact form where users can submit their details (such as name, email, phone number, and message), and the data is stored in an **Amazon DynamoDB** table.

The project integrates various AWS services to handle the front-end (contact form), back-end processing (Lambda), and data storage (DynamoDB). The **API Gateway** enables communication between the front-end and the serverless Lambda function, which processes the data and returns a success message to the user.

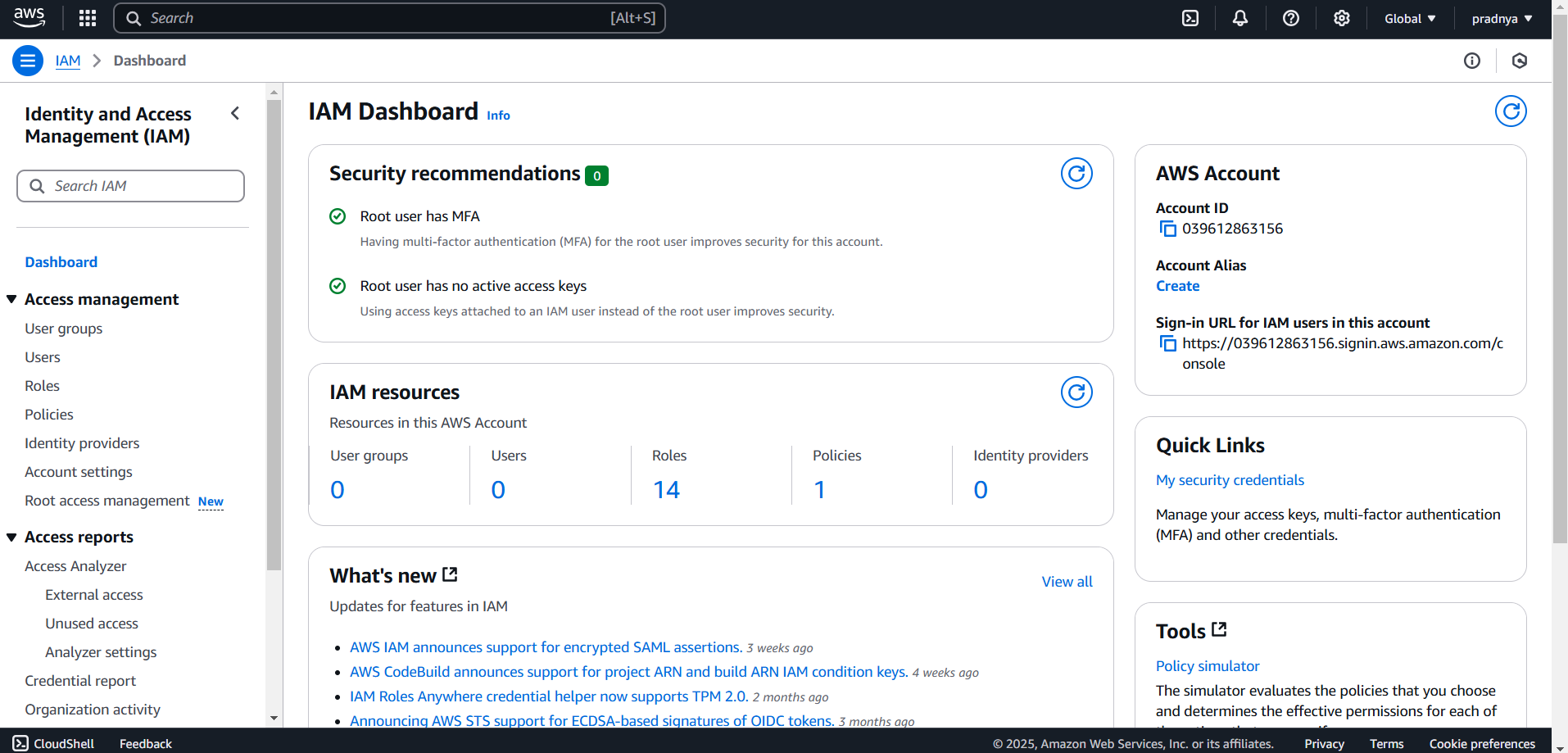
#### ****Key Components:****

* ****AWS Lambda**:** Handles the logic for the GET (display form) and POST (submit data) requests.
* ****Amazon API Gateway**:** Serves as the intermediary to expose Lambda functions via HTTP.
* ****Amazon DynamoDB**:** Stores the user-submitted data from the contact form.

### ****Step 1: Create an IAM Role with Required Permissions****

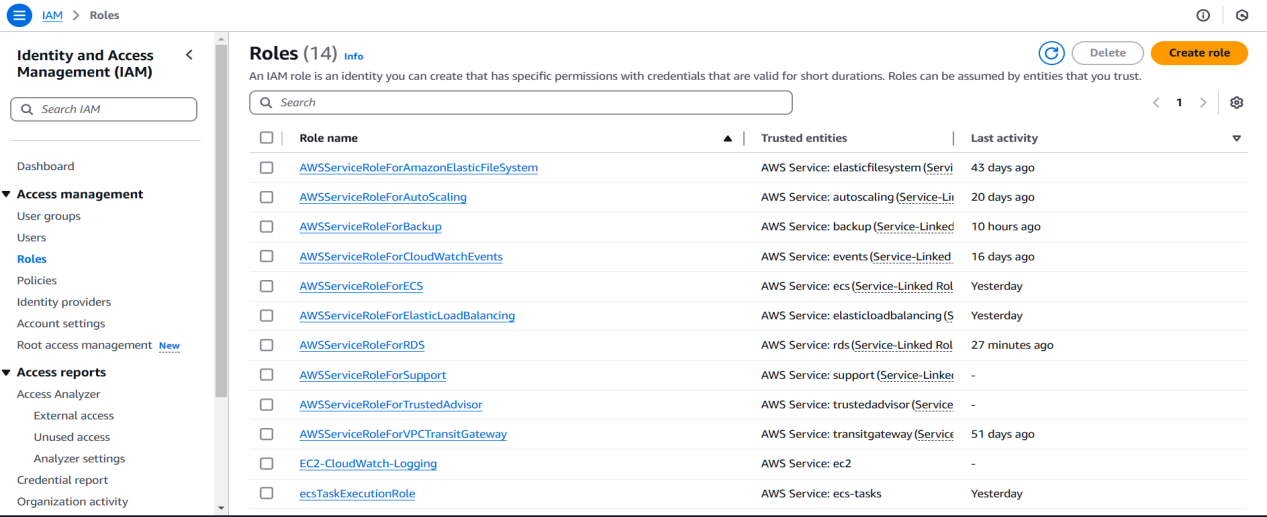
**1.Go to IAM in AWS Console**:

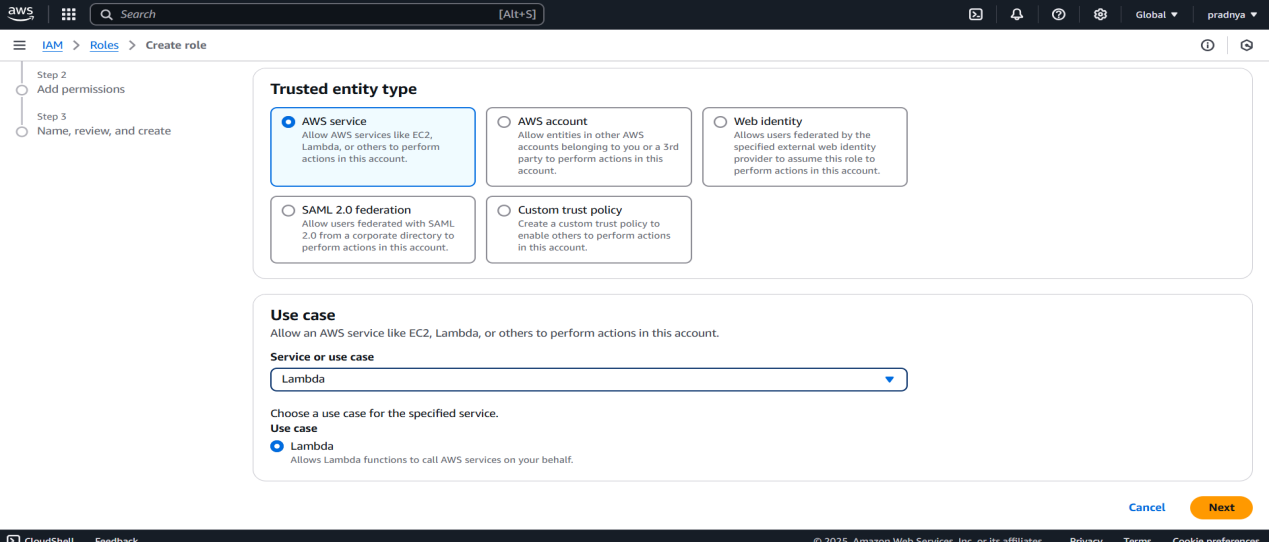
* 1. Navigate to the **IAM** service in the AWS console.

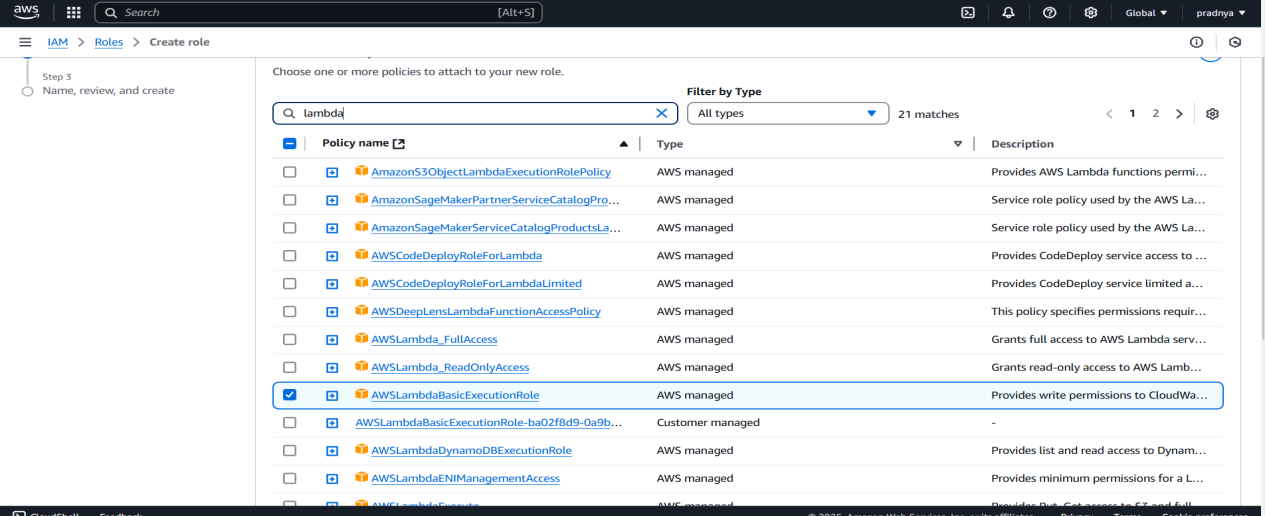


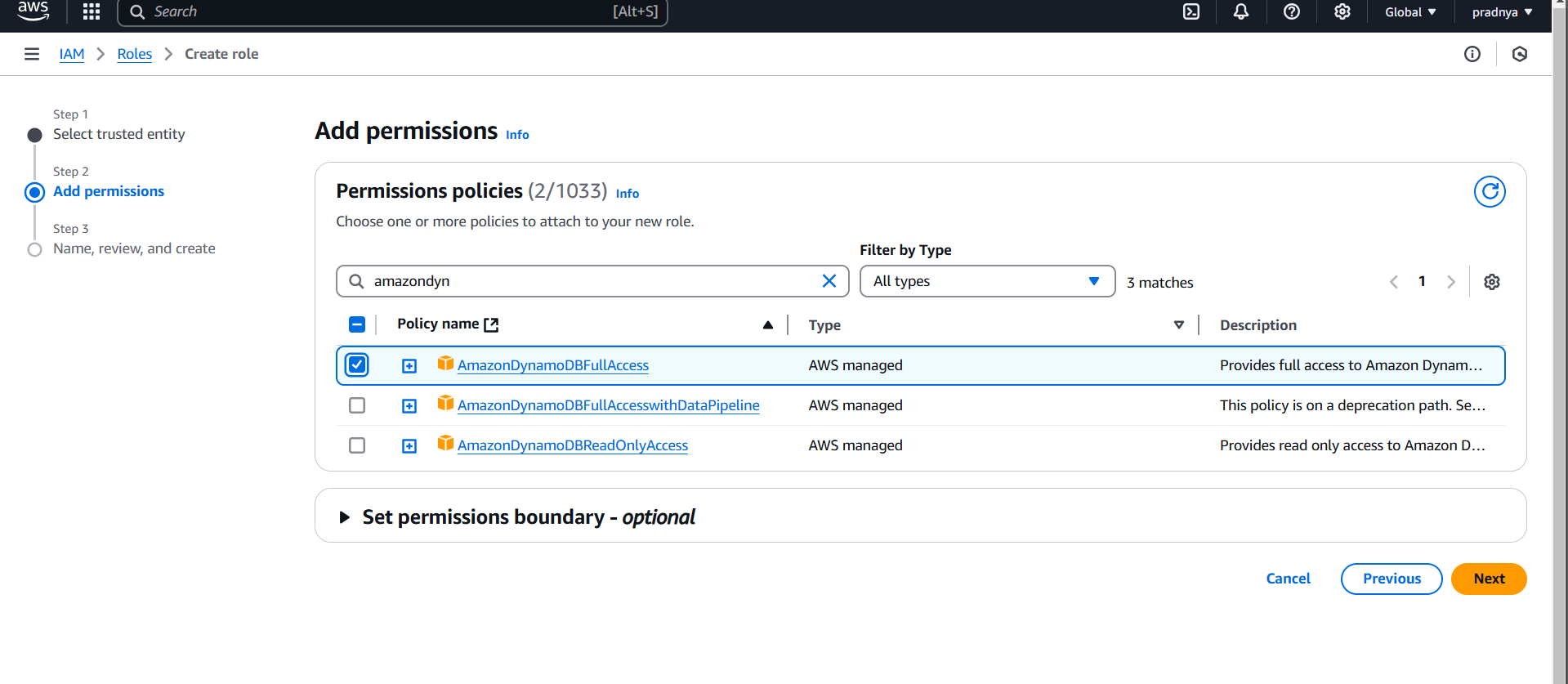
**2. Create a New Role**:

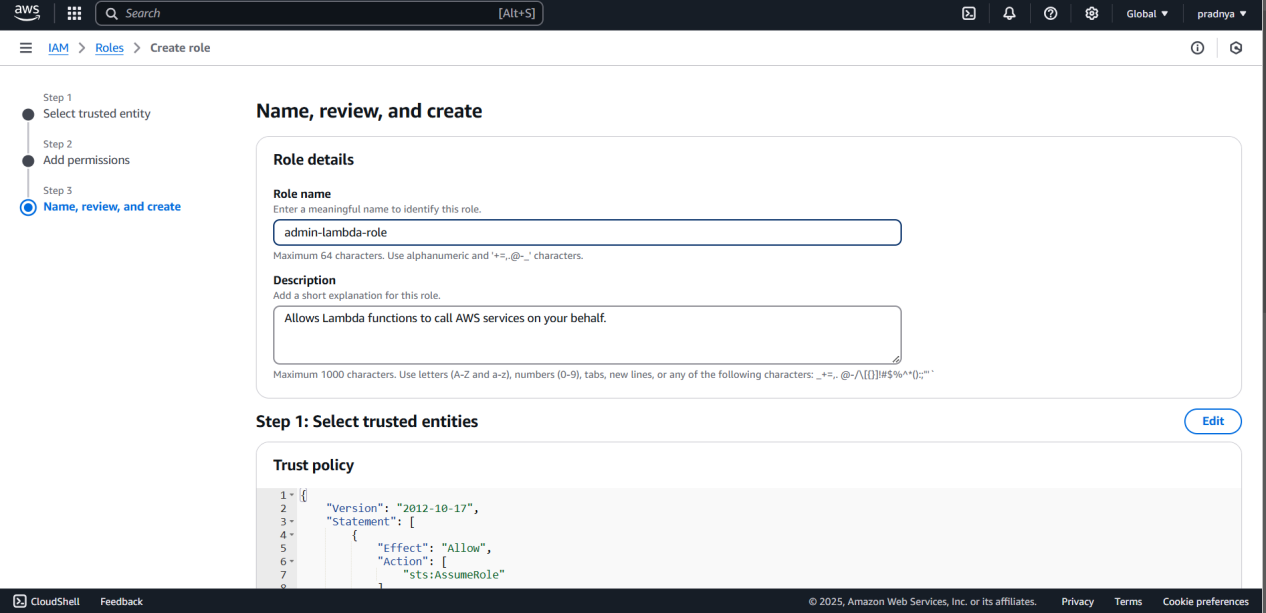
* 1. Click on **Roles** in the sidebar and then **Create Role**.
  2. Select **AWS service** as the trusted entity, and choose **Lambda** for the use case.
  3. Attach the following policies:
     1. **AWSLambdaBasicExecutionRole**
     2. **AmazonDynamoDBFullAccess**
  4. Name the role **admin-lambda-role** and create it.







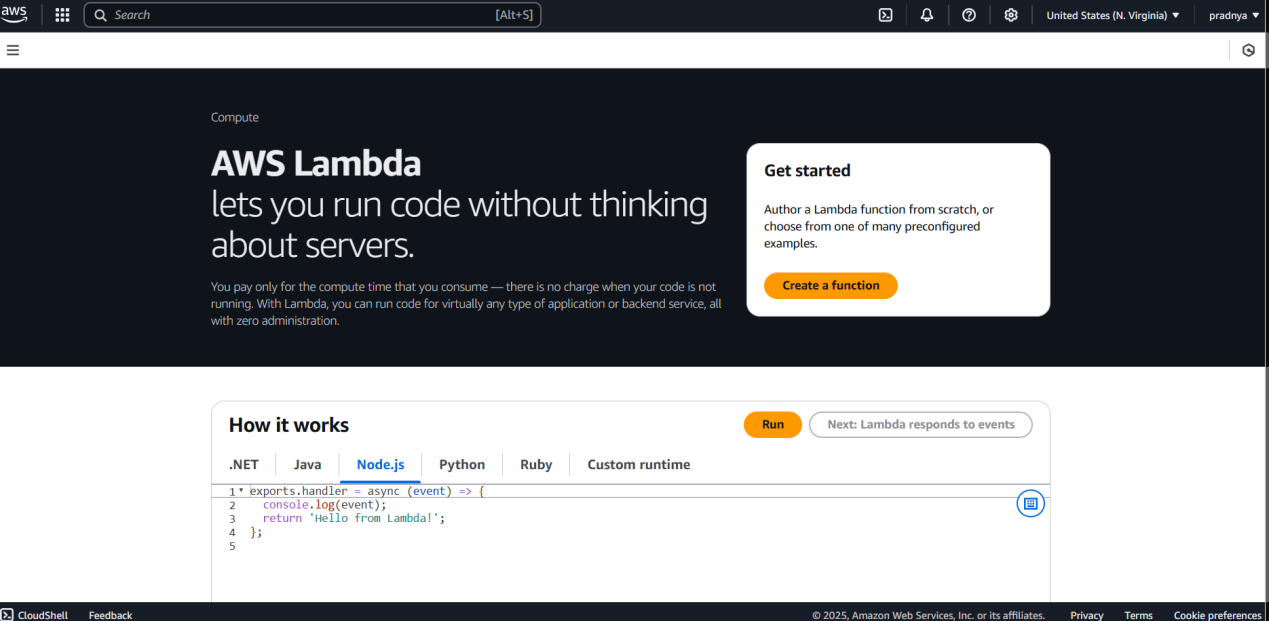




### ****Step 2: Create Lambda Function****

**1.Go to Lambda in AWS Console**:

* 1. Navigate to **AWS Lambda** in the AWS console.

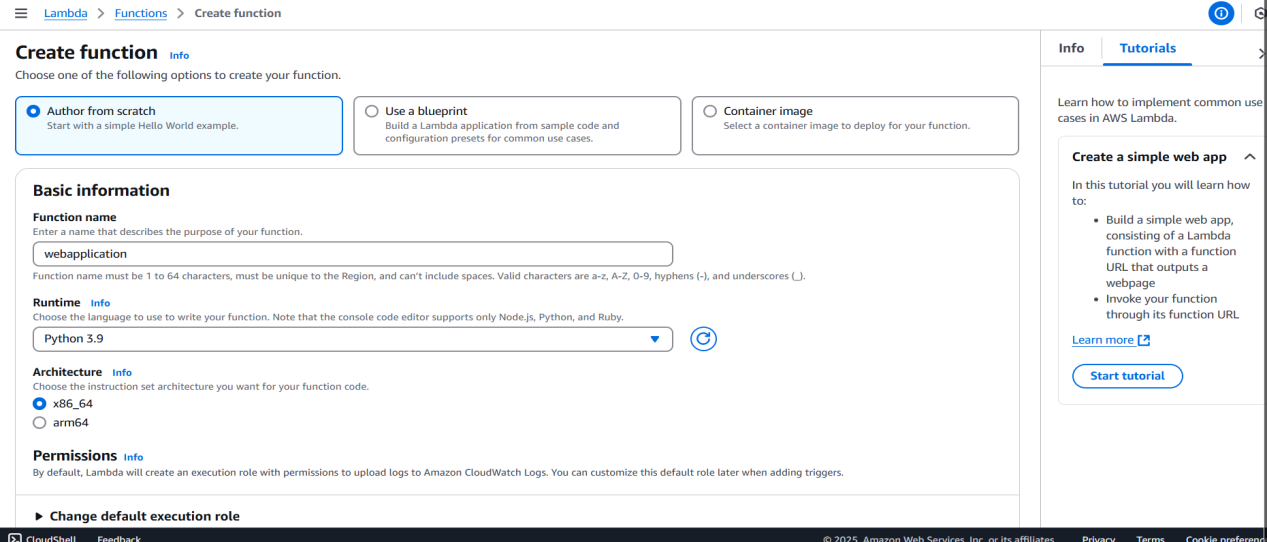


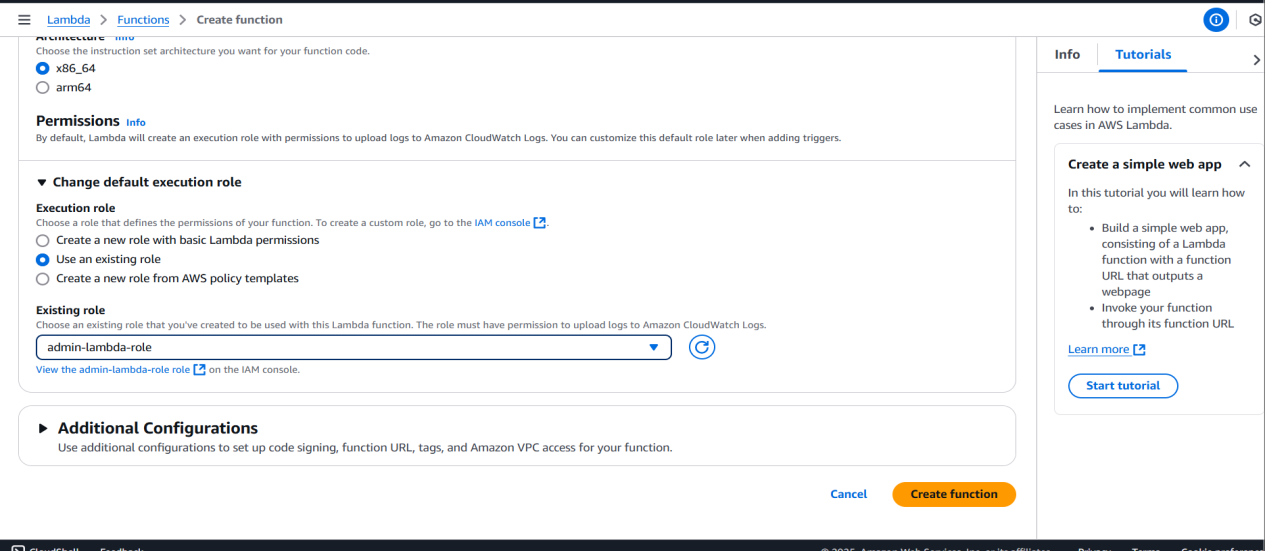
**2.Create a Lambda Function**:

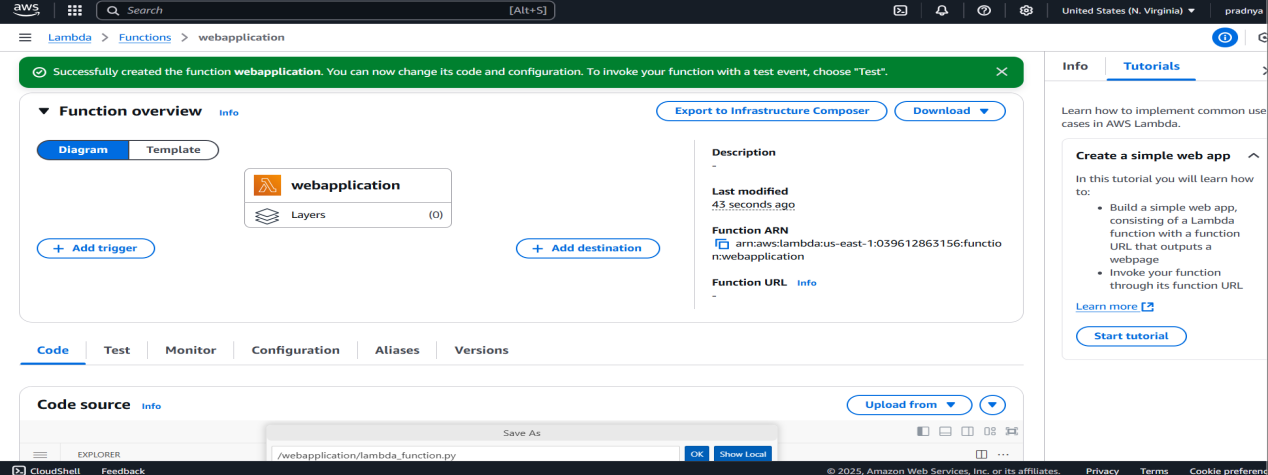
* 1. Click **Create function**.
  2. Select **Author from scratch**.
  3. Enter the function name as **webapplication**.
  4. Choose **Python 3.9** as the runtime.
  5. Under **Permissions**, select the previously created role **admin-lambda-role**.

**3.Write Lambda Function Code**:

The code is in my git now







### ****Step 3: Prepare Lambda Code and HTML Files****

**1.Prepare files**:

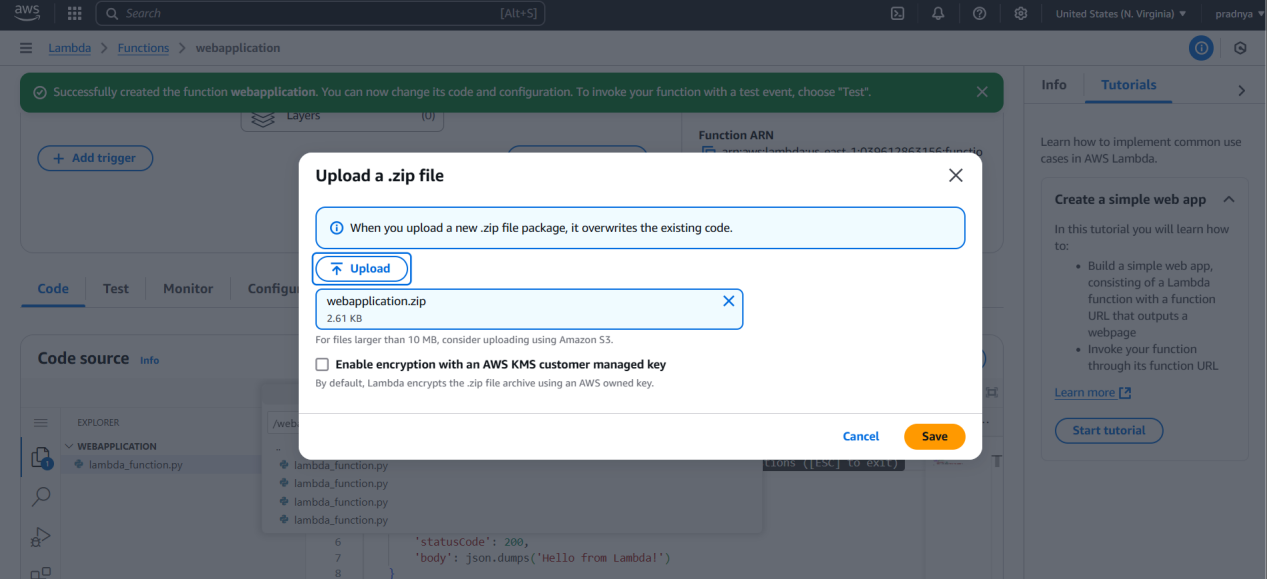
1.lambda\_function.py (Lambda function code)

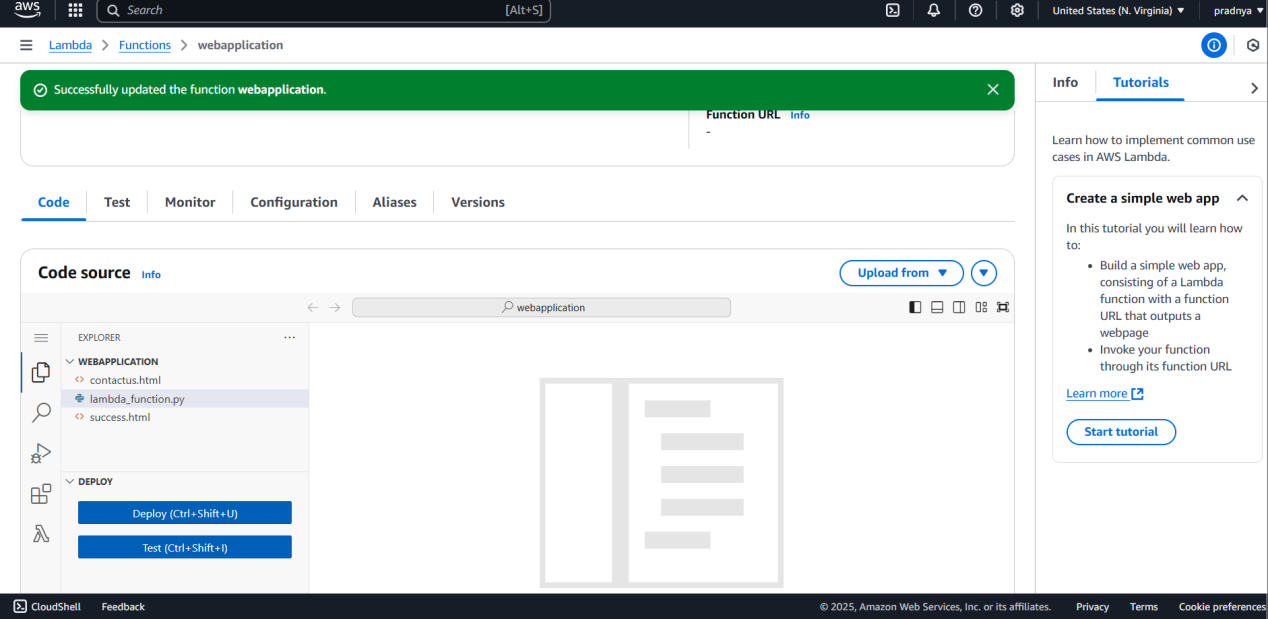
2.contactus.html (HTML file for the contact form)

3.success.html (HTML file for the success message)

**2.Create a ZIP File**:

* 1. Package these three files into a single ZIP file.



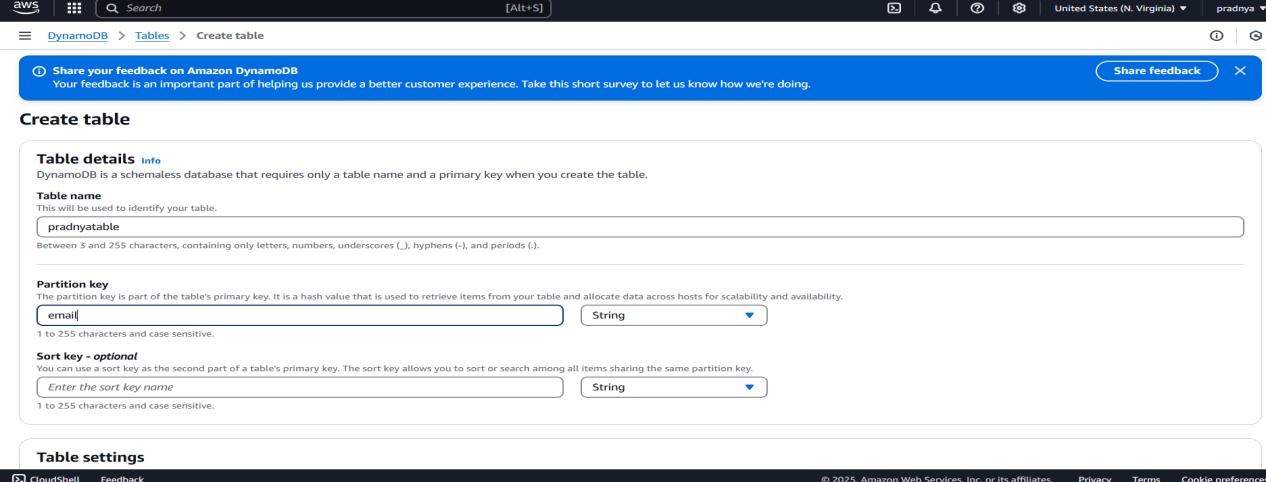
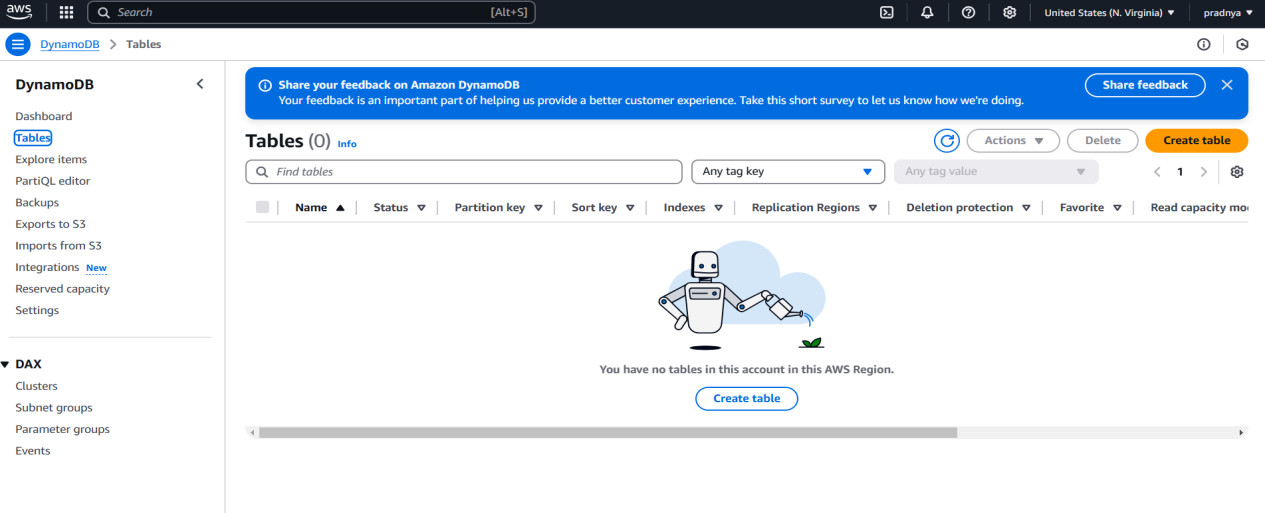


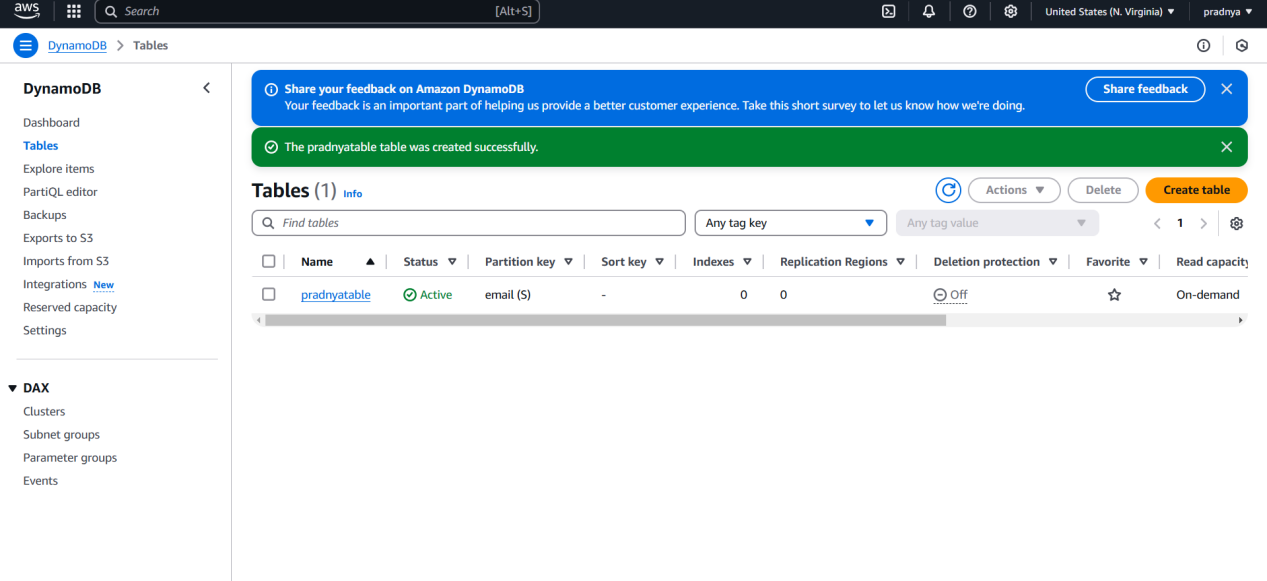
### ****Step 4: Upload Lambda Code****

1. In the **AWS Lambda Console**, select **Lambda function** (e.g., webapplication).
2. Scroll down to the **Function code** section.
3. In the **Code source** area, select **Upload from .zip**.
4. Click **Upload** and choose the lambda\_package.zip file you created earlier.
5. Click **Save**.

### ****Step 5: Create DynamoDB Table****

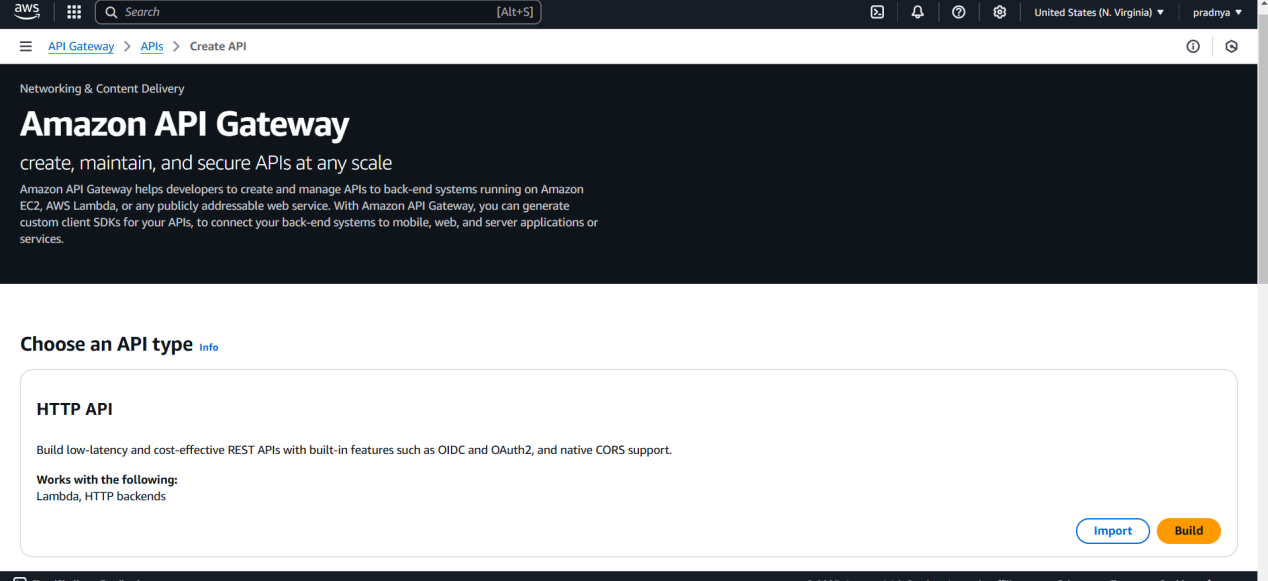
1. Go to the **DynamoDB** service in the AWS Console.
2. Click on **Create table** and configure it as follows:
   1. **Table name**: pradnyatable
   2. **Primary Key**: id (String)
3. Click **Create** to create the table.

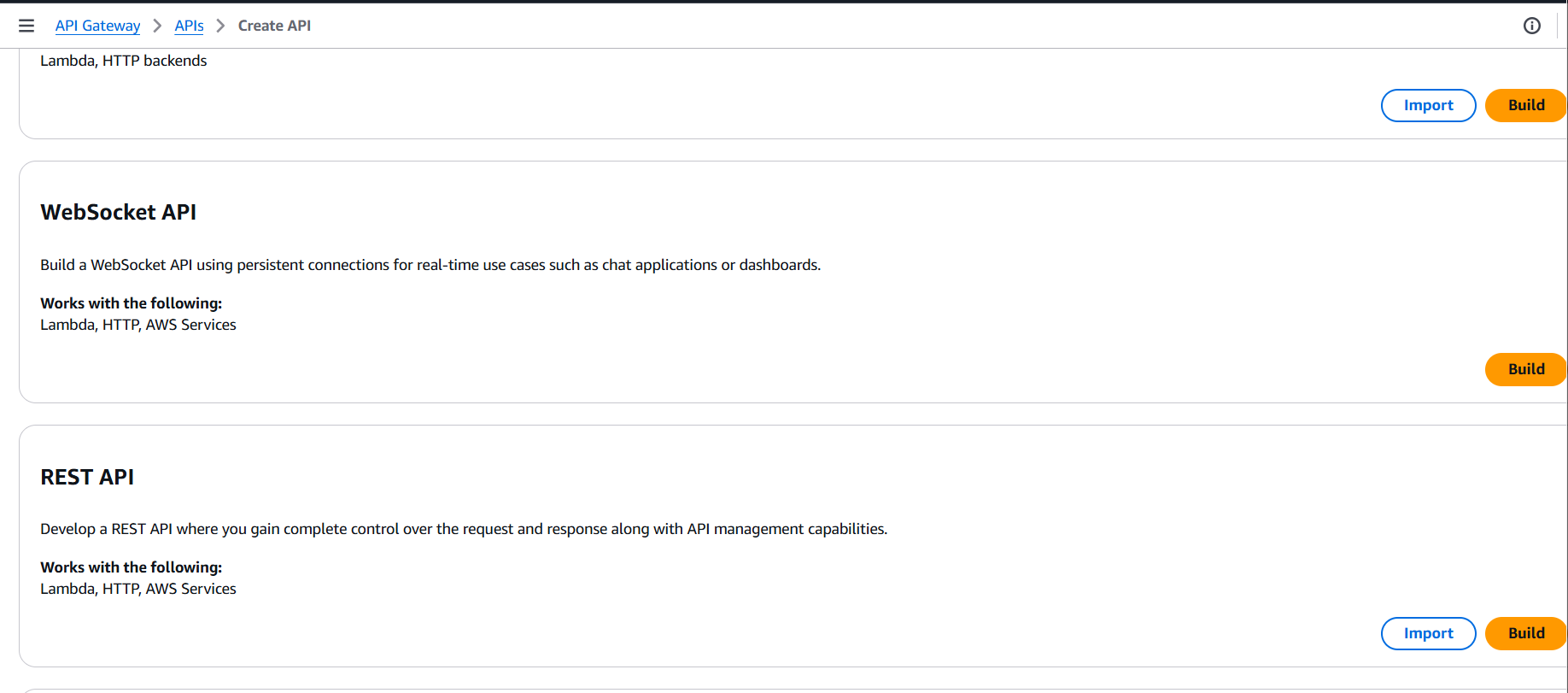


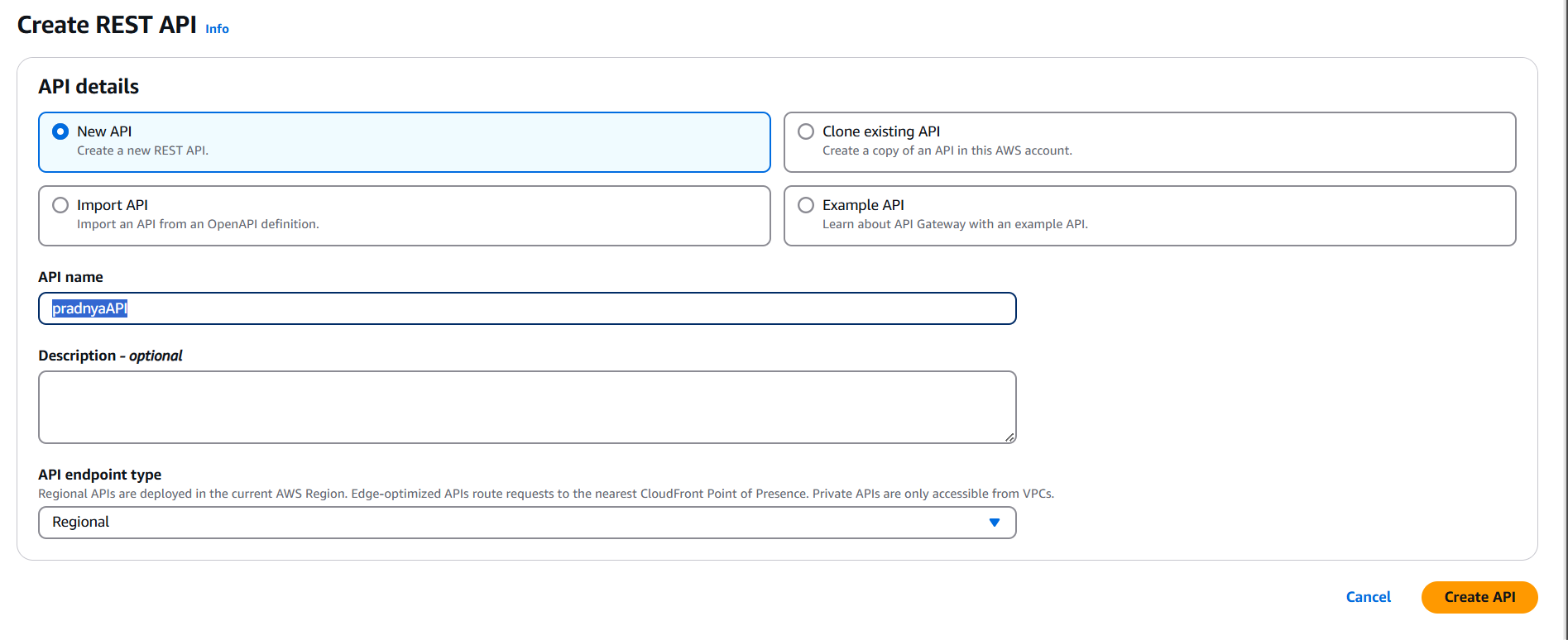


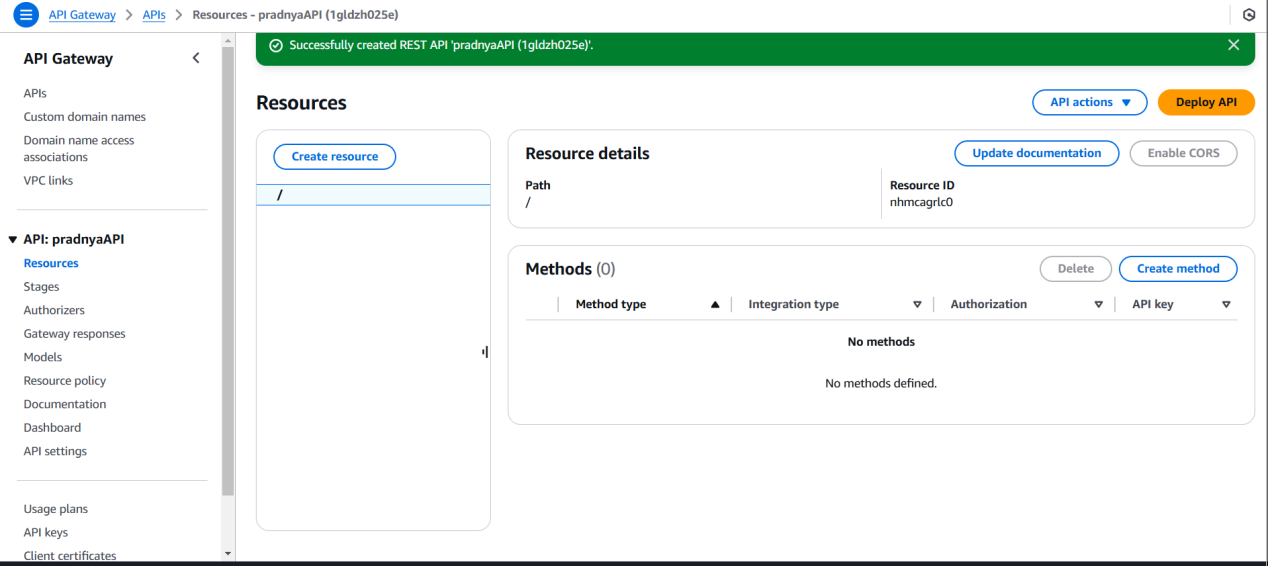
### ****Step 6: Configure API Gateway****

1. Go to the **API Gateway** service in the AWS Console.
2. Click on **Create API** and choose **REST API**.
3. Name the API as pradnyaAPI.

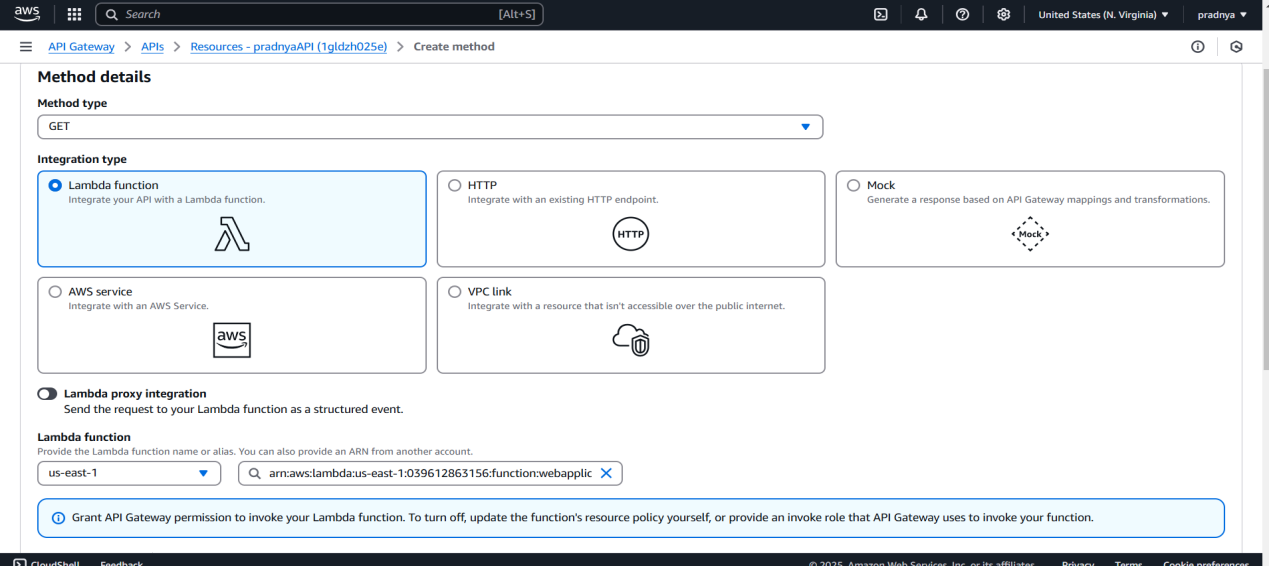


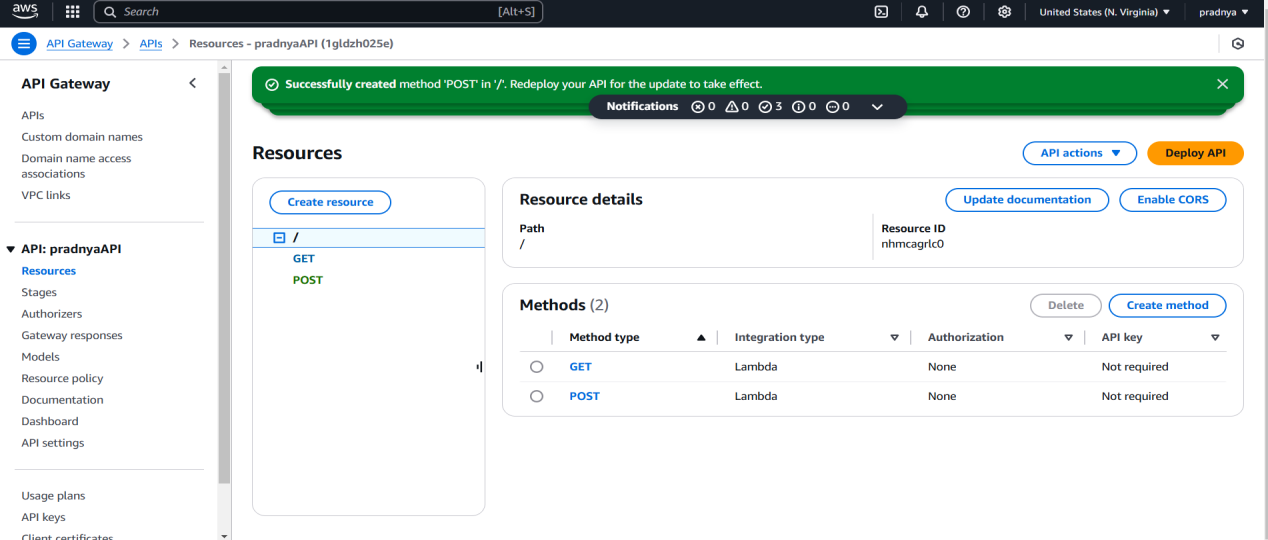


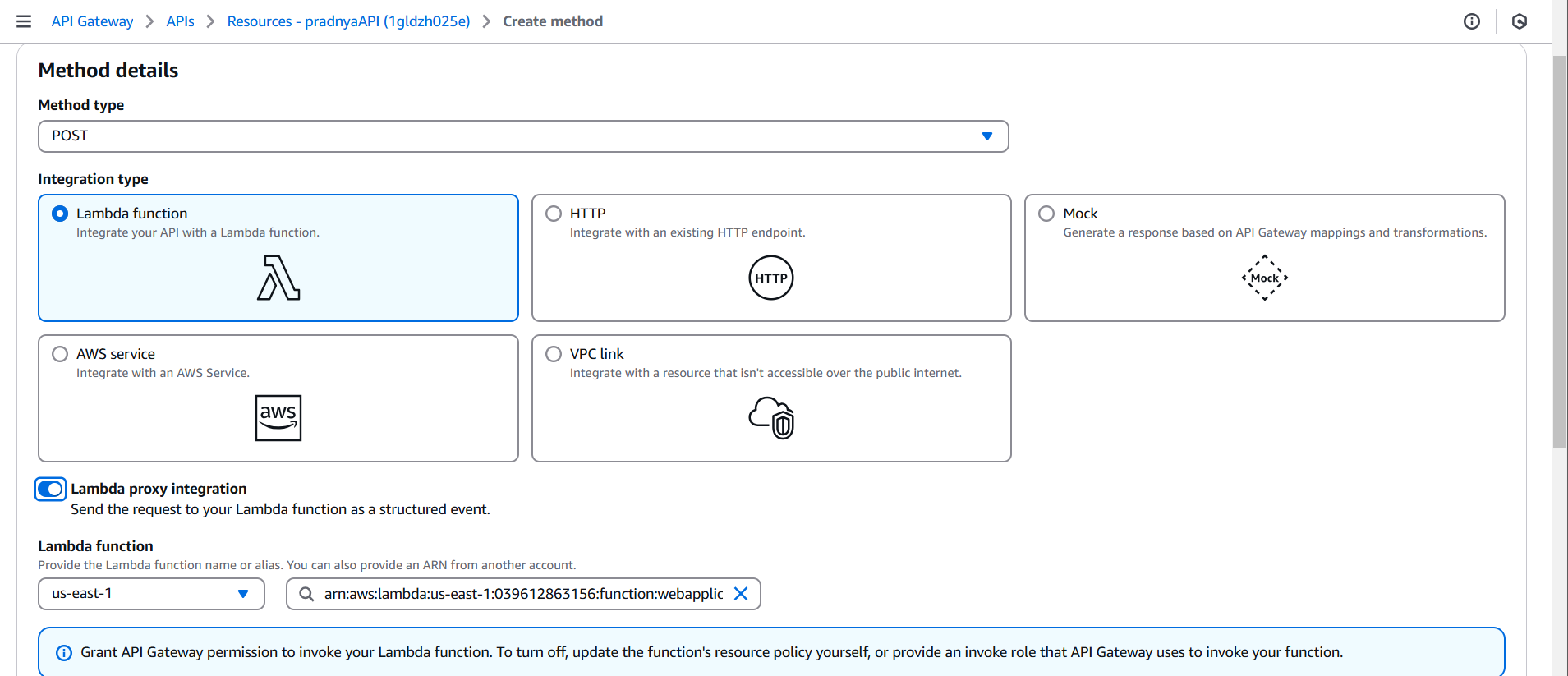




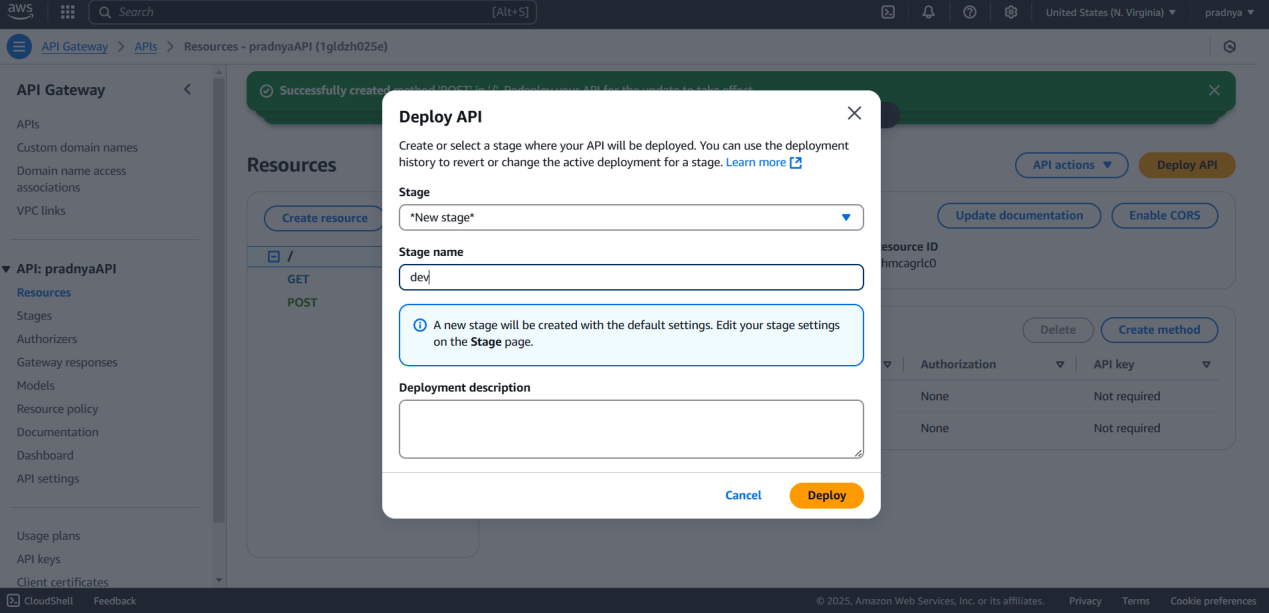
1. Create the following methods:
   1. ****GET Method**:**
      1. Select **Lambda Function** as the integration type.
      2. Enable **Lambda Proxy Integration**.
      3. Choose the webapplication Lambda function.
   2. ****POST Method**:**
      1. Repeat the above steps for the POST method.





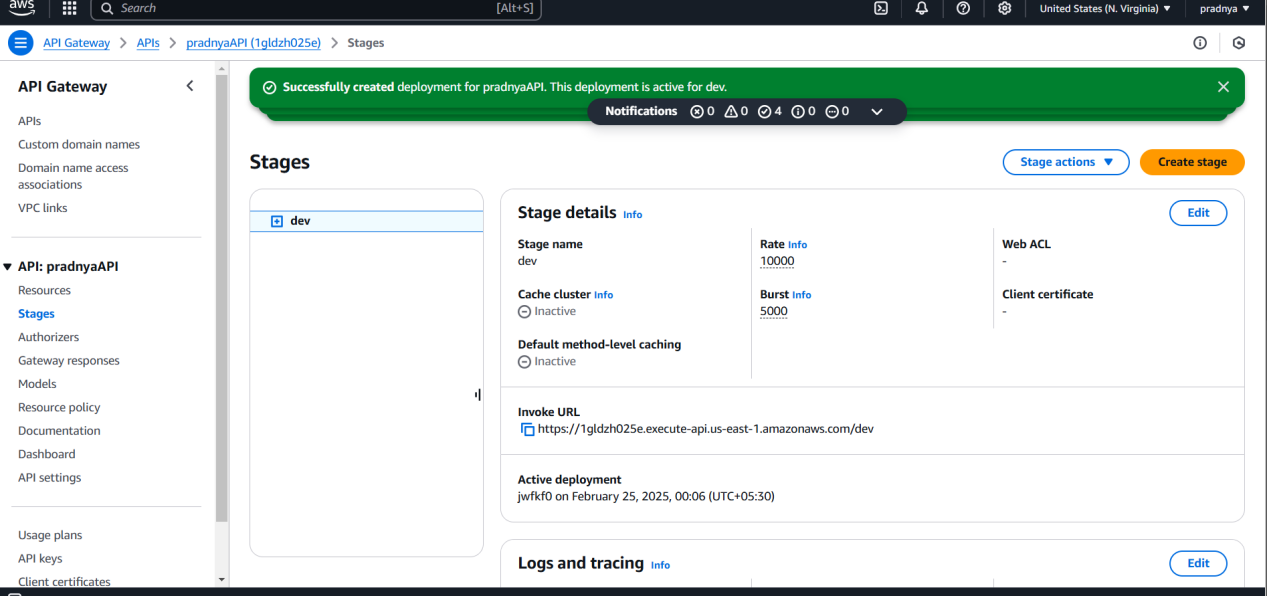


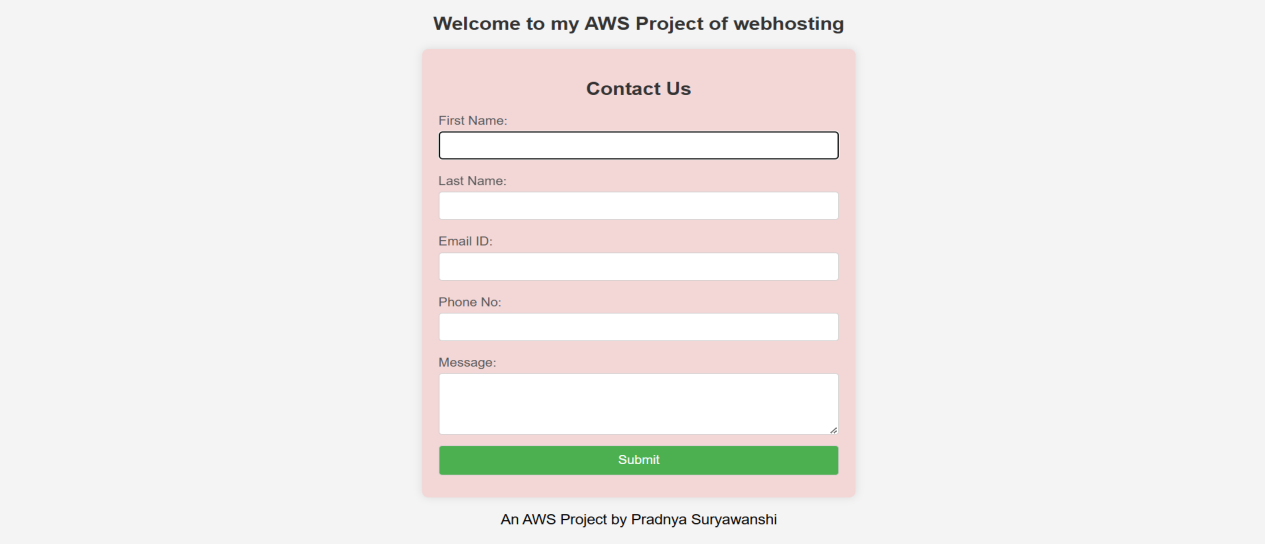
1. Deploy the API:
2. Click on **Deploy API.**
3. Create a new stage called dev and deploy the API.

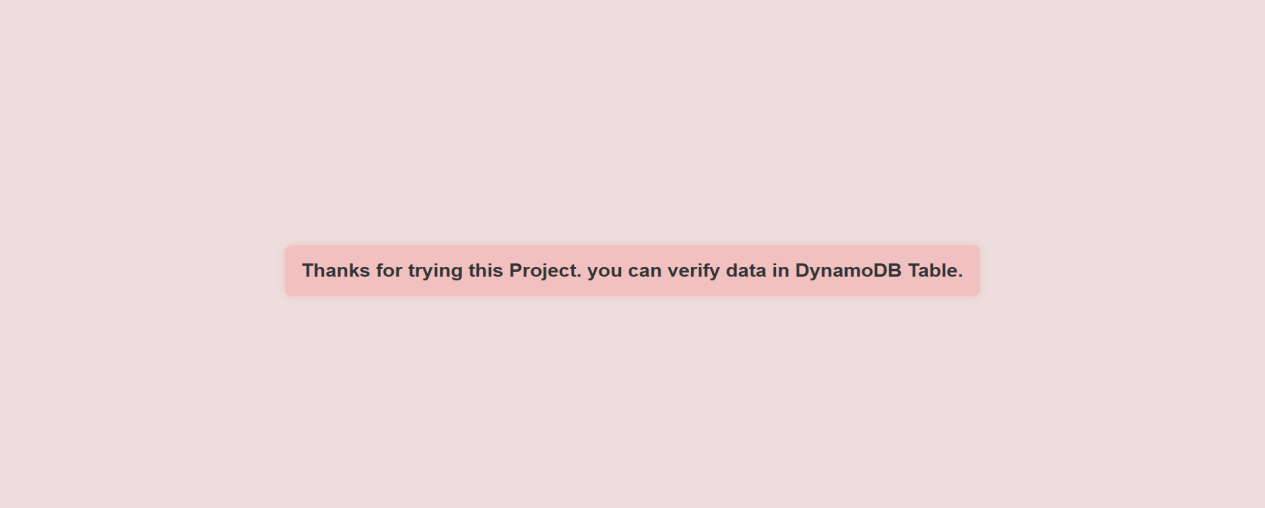


### ****Step 7: Test the Application****

1. After deploying the API, I will get an **Invoke URL**
2. Open this URL in a browser to test the contact form
3. Fill in the form and submit it.
4. I should see a success message and the data will be stored in DynamoDB.

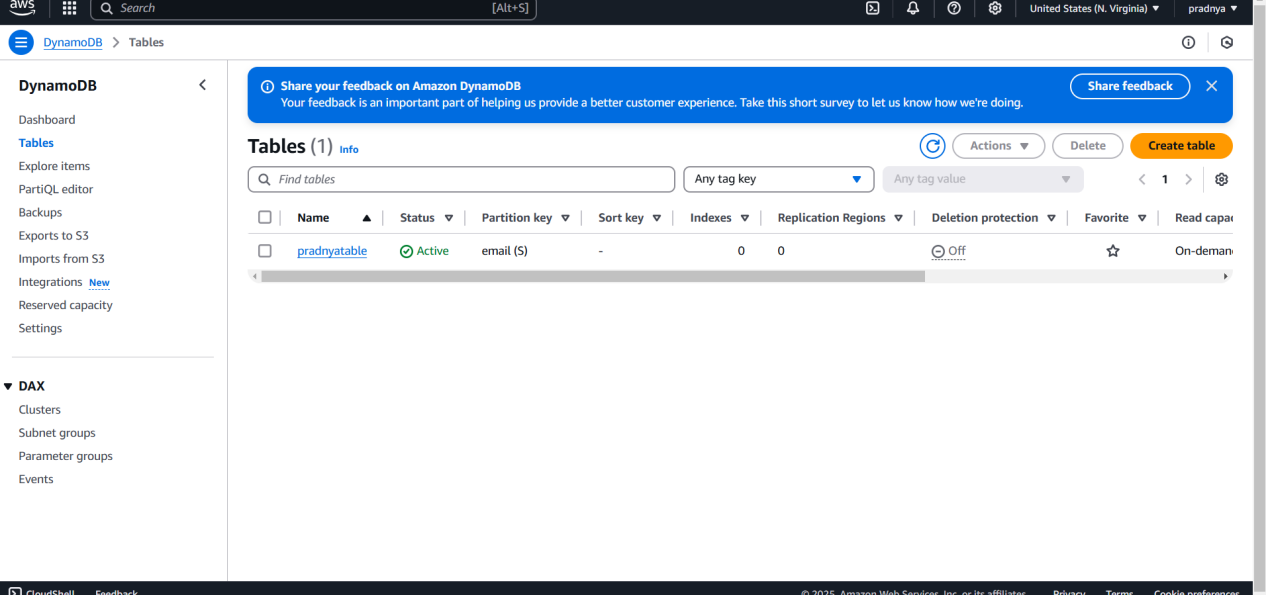


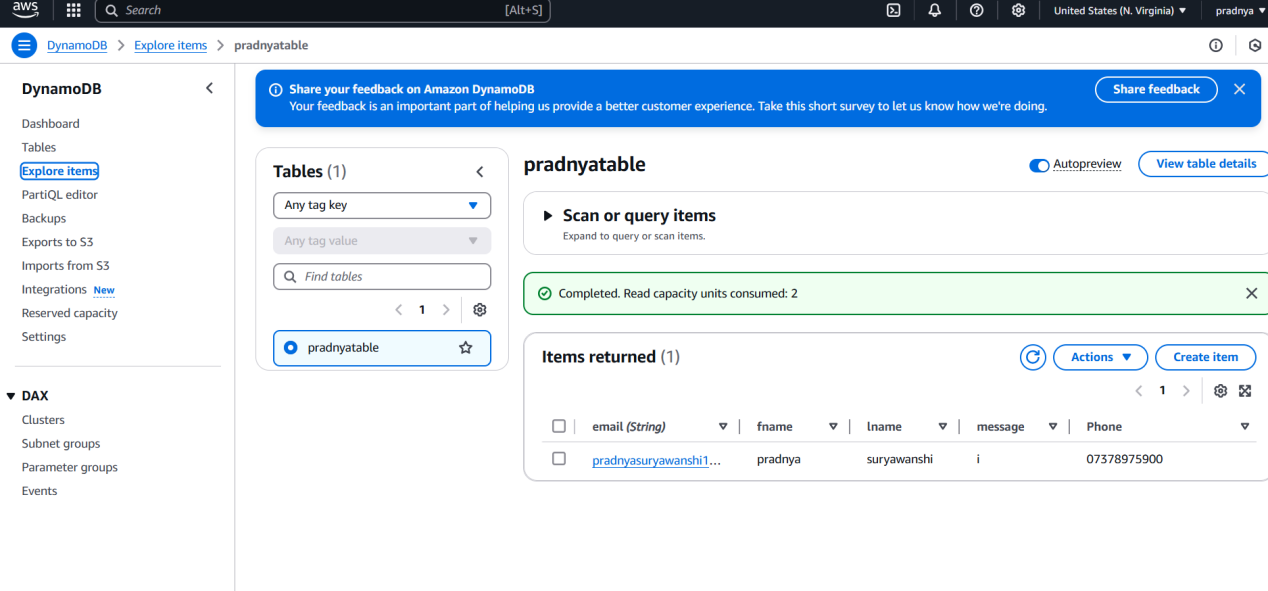




### ****Step 8: Verify Data in DynamoDB****

1. Go to the **DynamoDB Console**.
2. Select your table (pradnyatable).
3. Click on **Explore items** to view the stored data





### ****Conclusion****

This completes the setup and deployment of AWS Serverless application. I can now view the contact form, submit it, and verify the data stored in DynamoDB via the AWS Console.