

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.

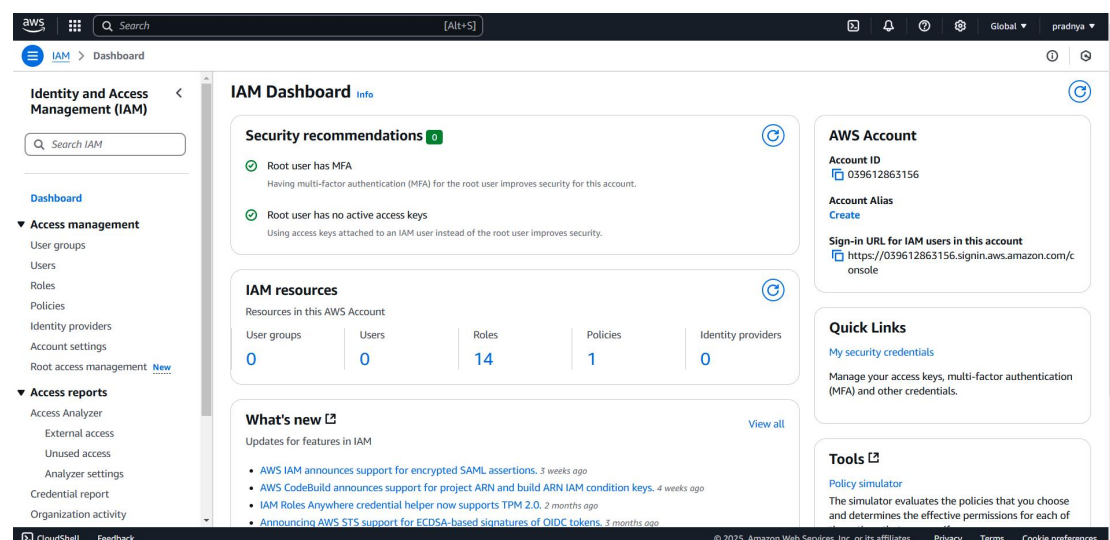
Key Components:

- **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:

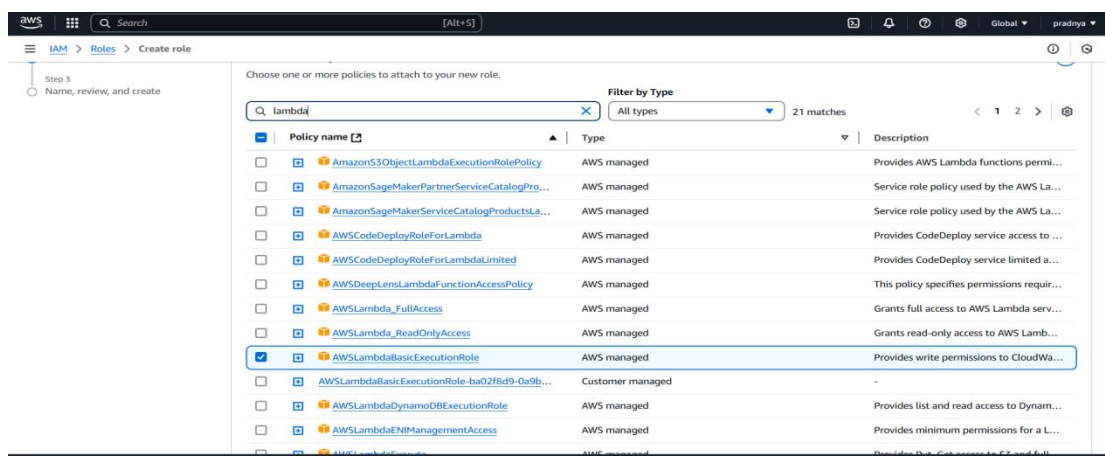
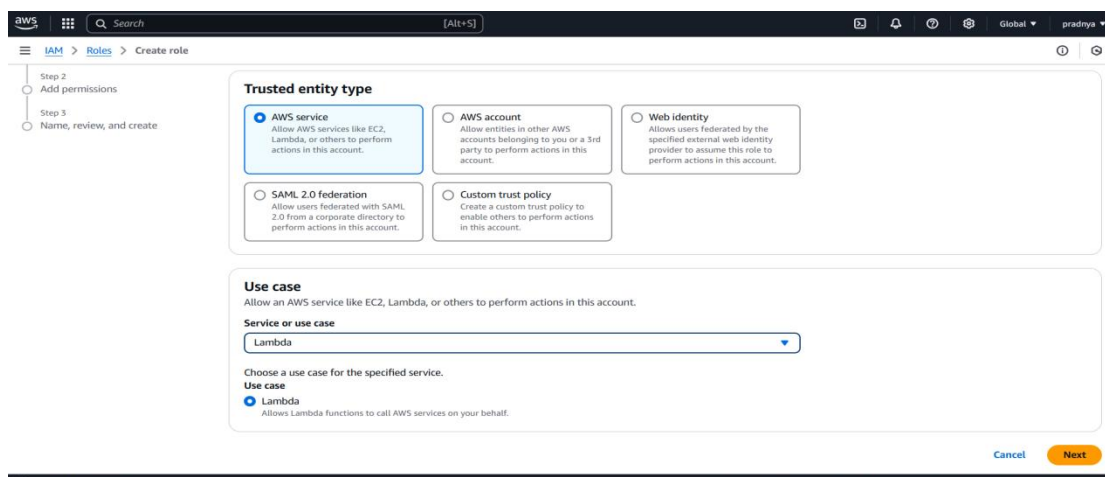
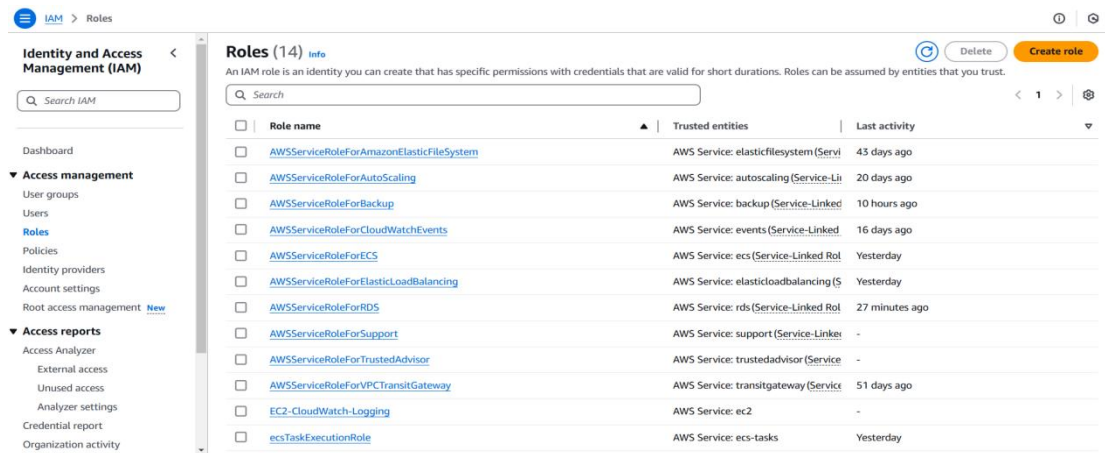
1. Navigate to the IAM service in the AWS console.

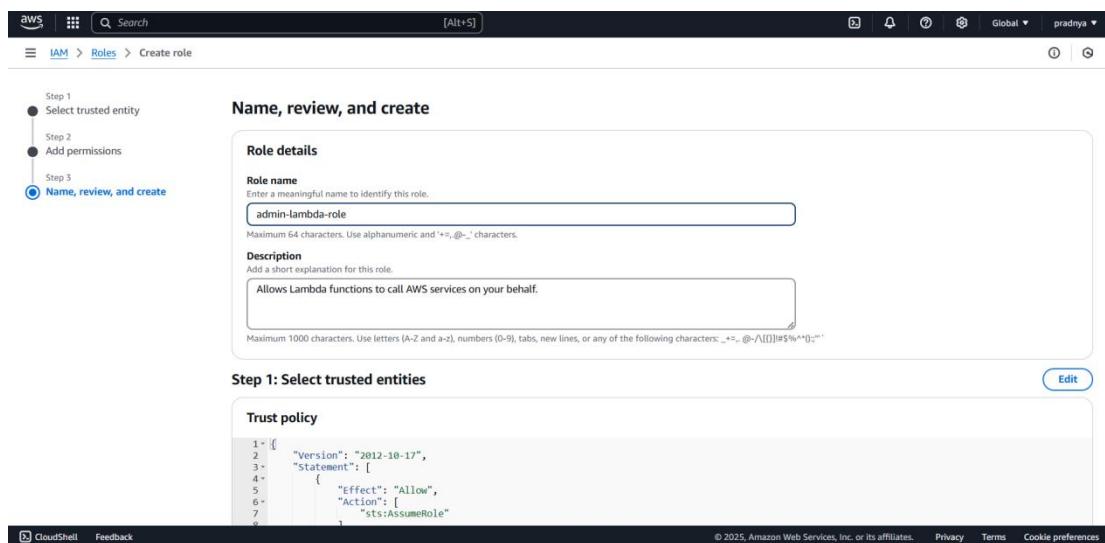
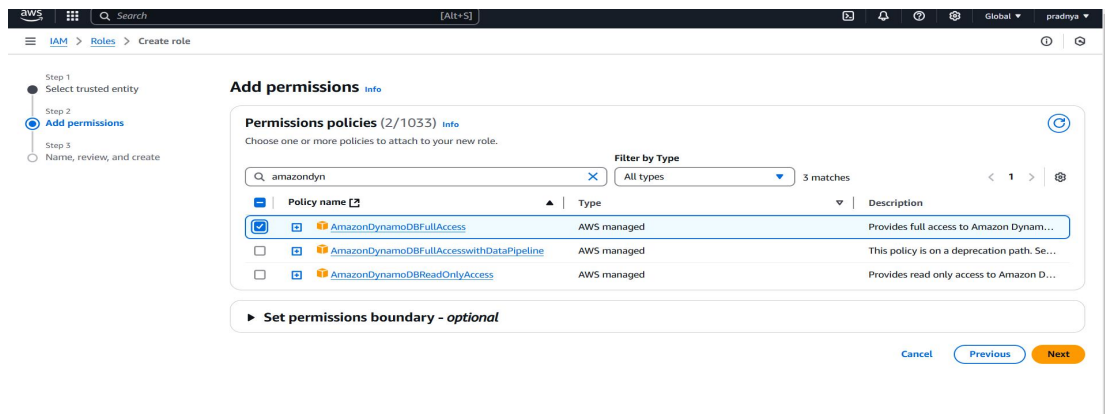


2. Create a New Role:

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.

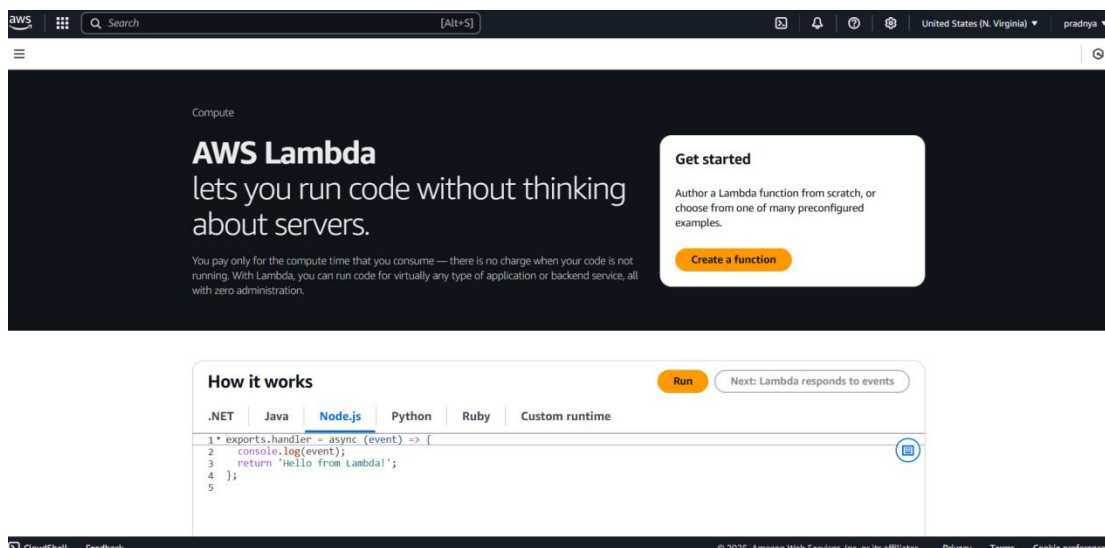




Step 2: Create Lambda Function

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

Create function [Info](#)

Choose one of the following options to create your function.

- ☒ **Author from scratch**
Start with a simple Hello World example.
- ☐ **Use a blueprint**
Build a Lambda application from sample code and configuration presets for common use cases.
- ☐ **Container image**
Select a container image to deploy for your function.

Basic information

Function name
Enter a name that describes the purpose of your function.

Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).

Runtime [Info](#)
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

Architecture [Info](#)
Choose the instruction set architecture you want for your function code.
☒ x86_64
☐ arm64

Permissions [Info](#)
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

[Change default execution role](#)

Permissions [Info](#)

By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

Change default execution role

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

- ☐ Create a new role with basic Lambda permissions
- ☒ Use an existing role
- ☐ Create a new role from AWS policy templates

Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

[View the admin-lambda-role](#) on the IAM console.

Additional Configurations
Use additional configurations to set up code signing, function URL, tags, and Amazon VPC access for your function.

[Cancel](#) [Create function](#)

Function overview [Info](#)

Successfully created the function **webapplication**. You can now change its code and configuration. To invoke your function with a test event, choose "Test".

[Export to Infrastructure Composer](#) [Download](#)

Description

- Last modified**
43 seconds ago
- Function ARN**
[arn:aws:lambda:us-east-1:039612863156:function:webapplication](#)
- Function URL** [Info](#)

Code source [Info](#)

[Upload from](#)

[Code](#) [Test](#) [Monitor](#) [Configuration](#) [Aliases](#) [Versions](#)

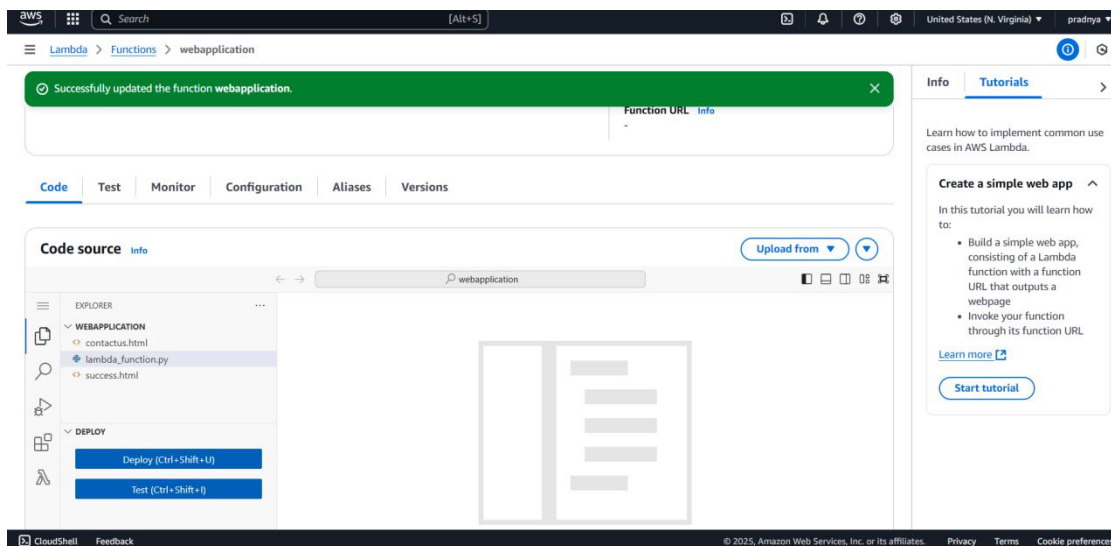
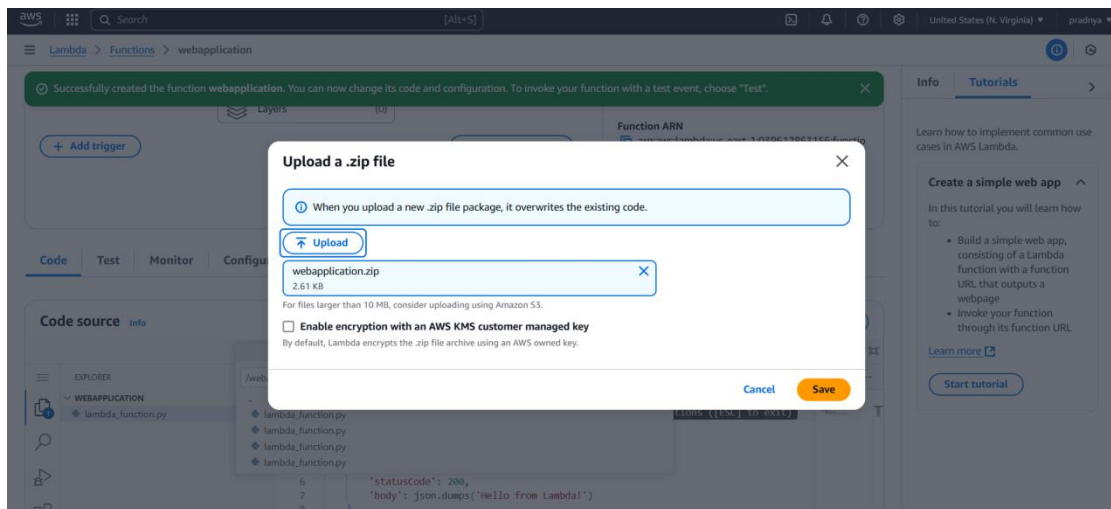
Step 3: Prepare Lambda Code and HTML Files

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.



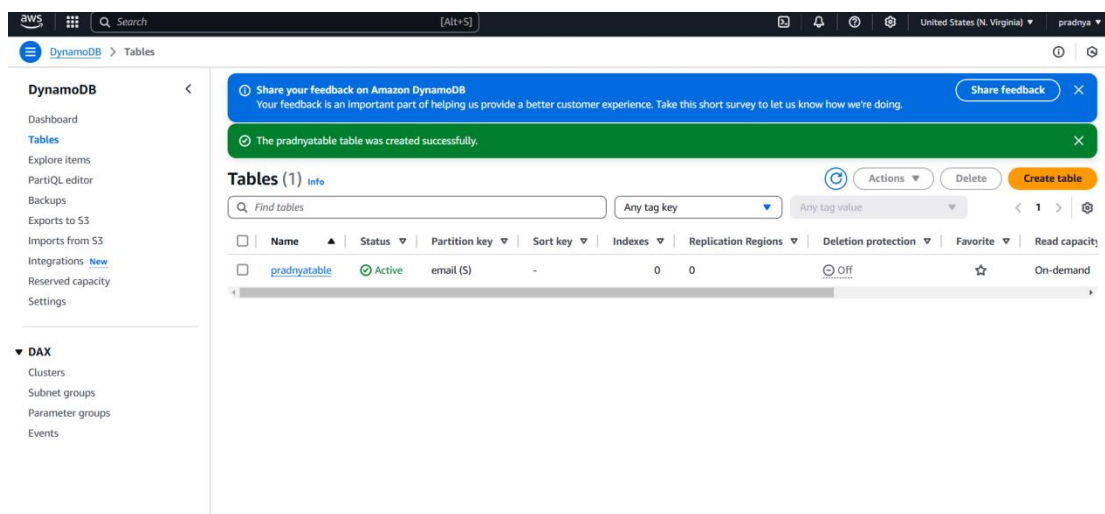
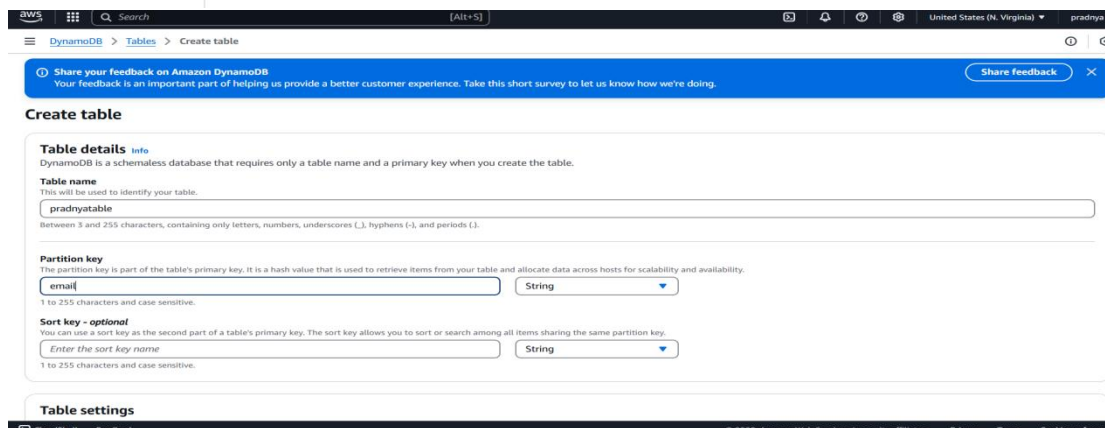
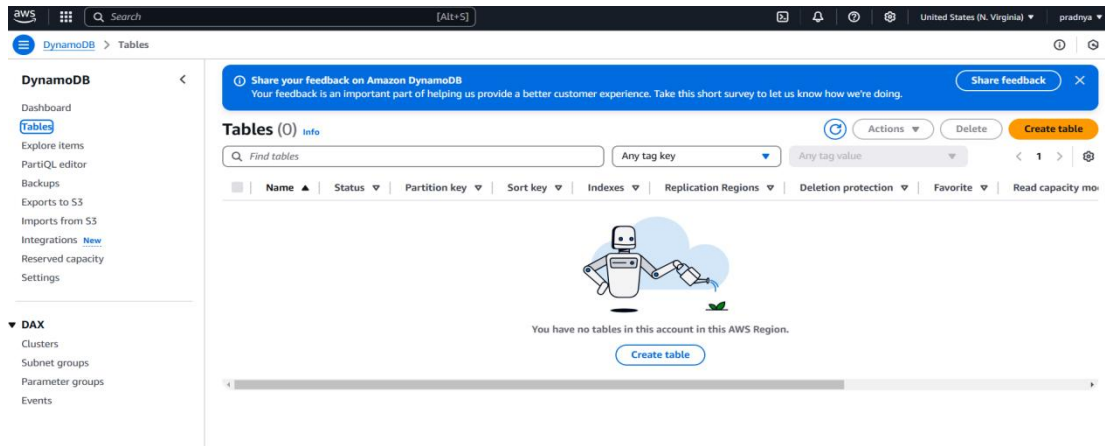
Step 4: Upload Lambda Code

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.

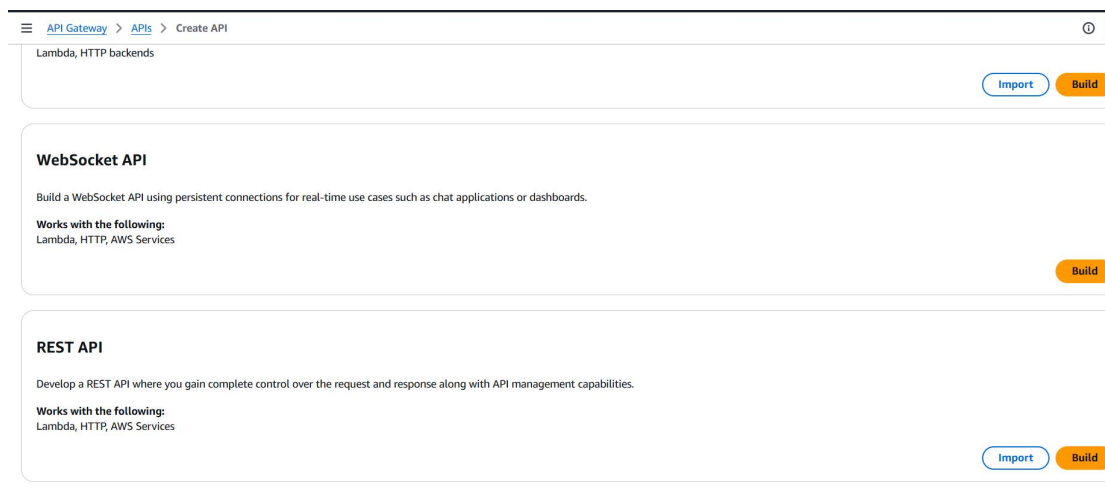
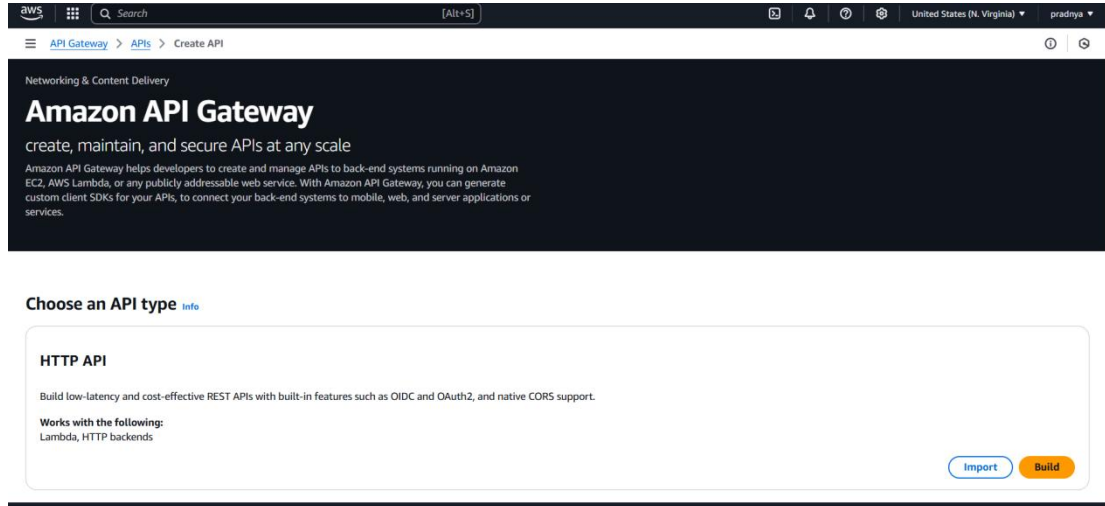
Step 5: Create DynamoDB Table

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.



Create REST API

API details

☒ **New API**
Create a new REST API.

☐ **Clone existing API**
Create a copy of an API in this AWS account.

☐ **Import API**
Import an API from an OpenAPI definition.

☐ **Example API**
Learn about API Gateway with an example API.

API name

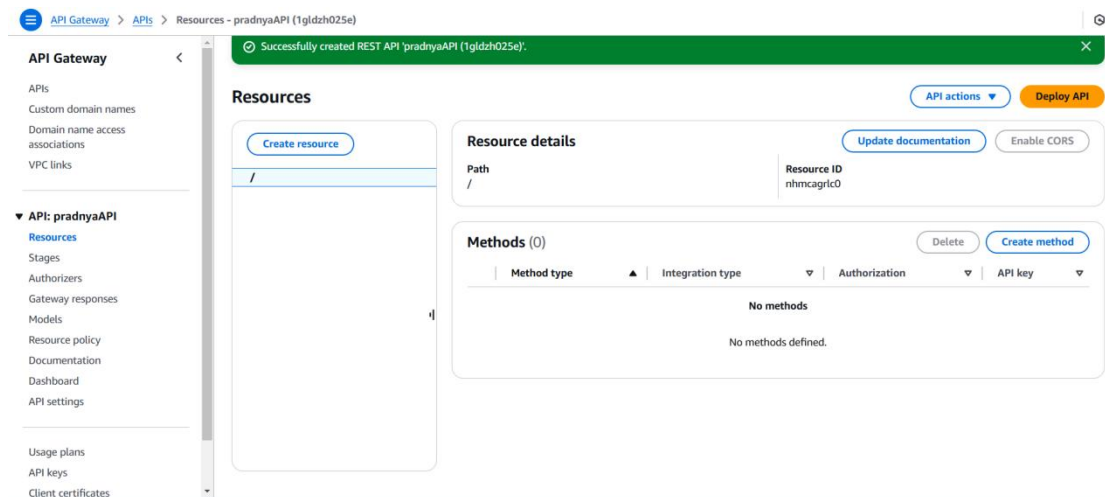
pradnyaAPI

Description - optional

API endpoint type
Regional APIs are deployed in the current AWS Region. Edge-optimized APIs route requests to the nearest CloudFront Point of Presence. Private APIs are only accessible from VPCs.

Regional

Cancel Create API



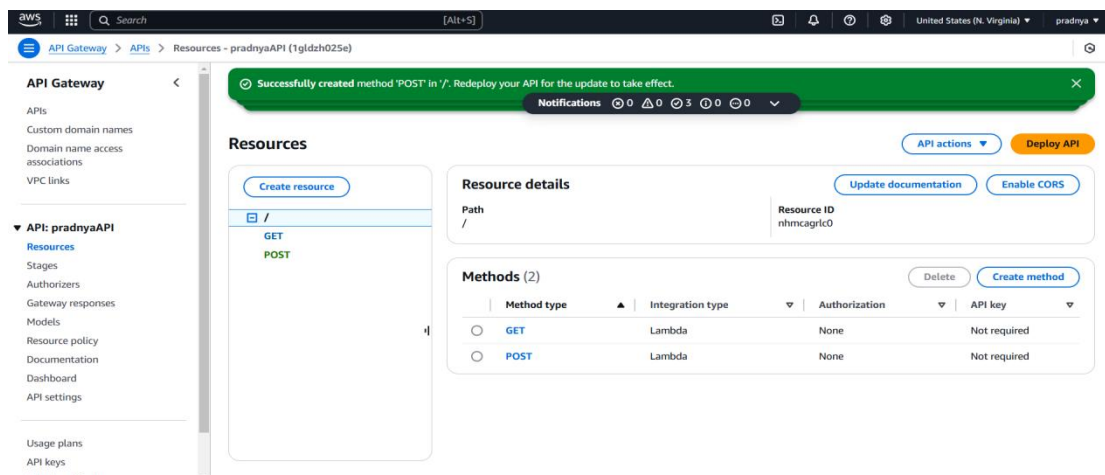
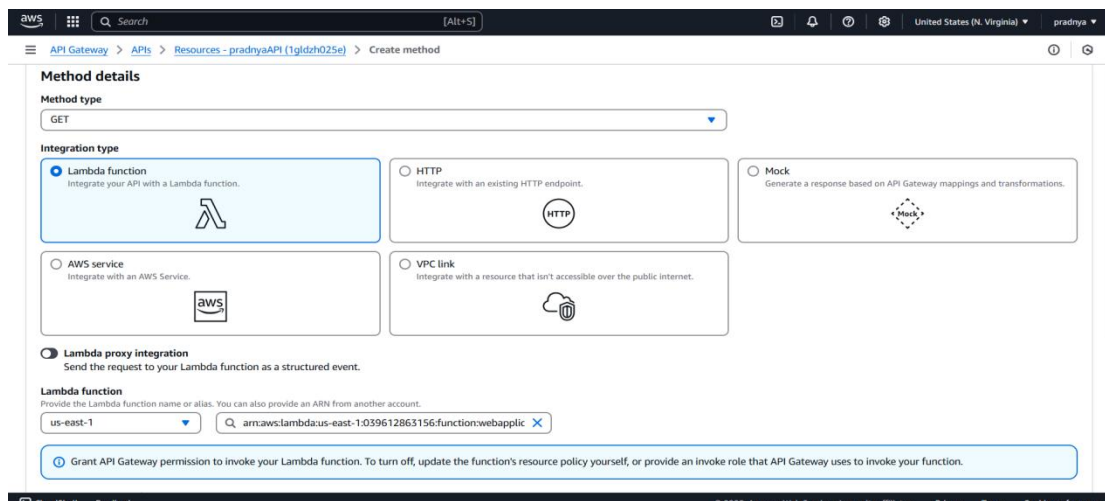
4. Create the following methods:

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.



API Gateway > APIs > Resources - pradnyaAPI (1gldzh025e) > Create method

Method details

Method type
POST

Integration type

- ☒ **Lambda function**
Integrate your API with a Lambda function.
- ☐ **HTTP**
Integrate with an existing HTTP endpoint.
- ☐ **Mock**
Generate a response based on API Gateway mappings and transformations.
- ☐ **AWS service**
Integrate with an AWS Service.
- ☐ **VPC link**
Integrate with a resource that isn't accessible over the public internet.

☒ **Lambda proxy integration**
Send the request to your Lambda function as a structured event.

Lambda function
Provide the Lambda function name or alias. You can also provide an ARN from another account.

us-east-1

Grant API Gateway permission to invoke your Lambda function. To turn off, update the function's resource policy yourself, or provide an invoke role that API Gateway uses to invoke your function.

5. Deploy the API:

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

API Gateway > APIs > Resources - pradnyaAPI (1gldzh025e)

Successfully created resource

Deploy API

Create or select a stage where your API will be deployed. You can use the deployment history to revert or change the active deployment for a stage. [Learn more](#)

Stage
New stage

Stage name
dev

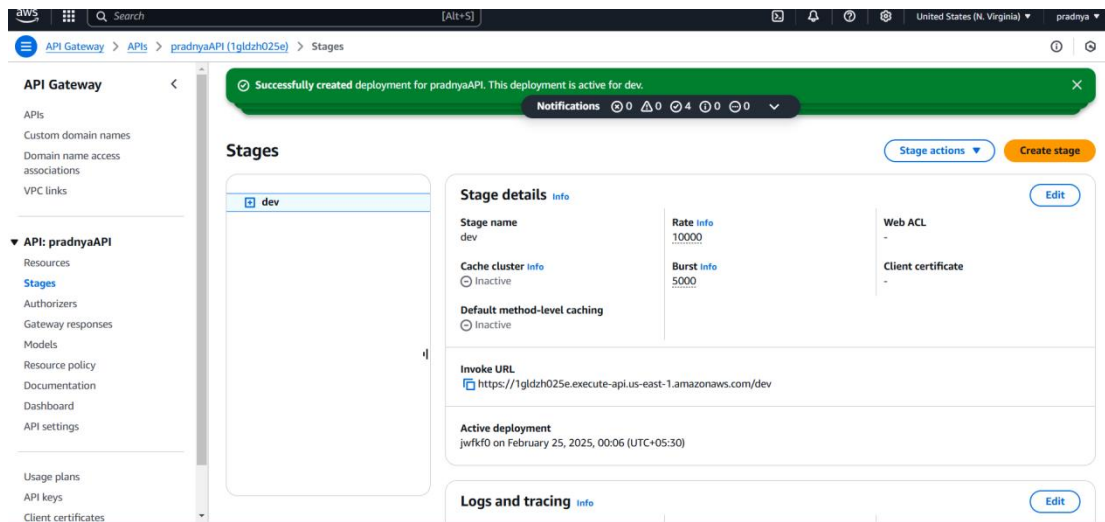
A new stage will be created with the default settings. Edit your stage settings on the Stage page.

Deployment description

Cancel Deploy

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.



Welcome to my AWS Project of webhosting

Contact Us

First Name:

Last Name:

Email ID:

Phone No:

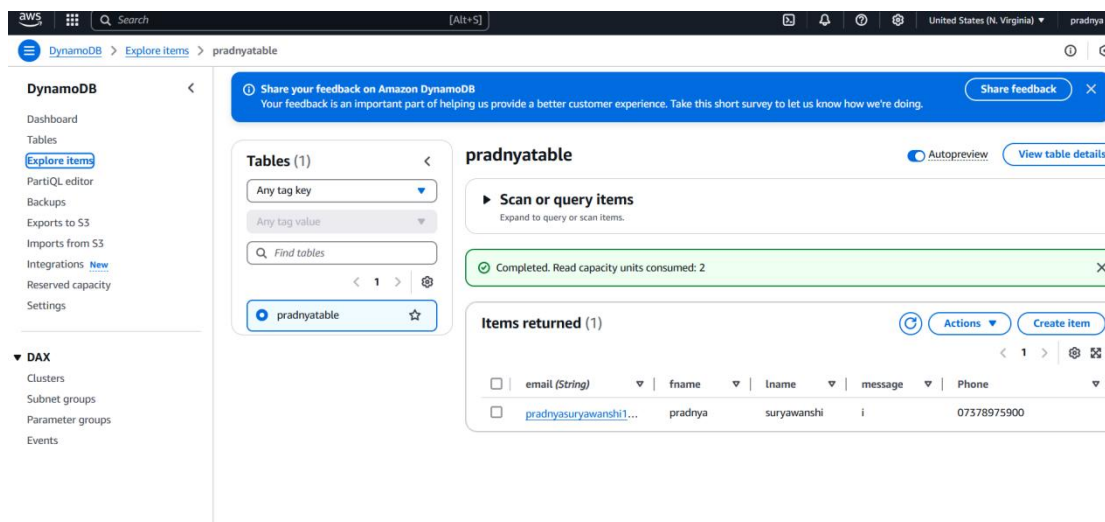
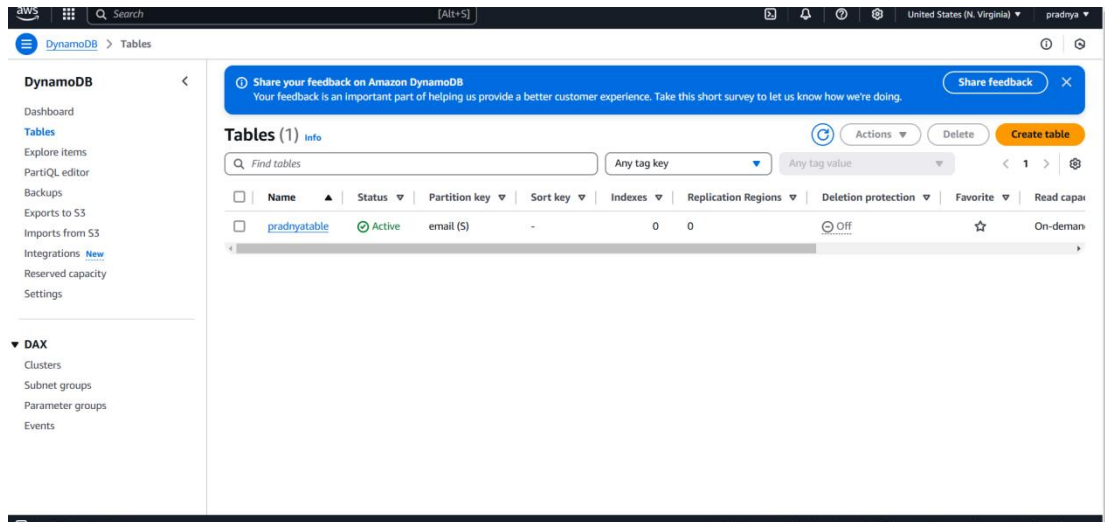
Message:

An AWS Project by Pradnya Suryawanshi

Thanks for trying this Project. you can verify data in DynamoDB Table.

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.