

Serverless Labs

Building a Serverless Web Application

Objective:

Create a serverless web application using AWS Lambda, API Gateway, S3, and DynamoDB.

Approach:

Set Up Backend: Create Lambda functions to handle backend logic. These functions will interact with a DynamoDB table for data storage.

API Gateway: Set up API Gateway to create RESTful endpoints that trigger the Lambda functions.

Frontend Hosting: Host a static website on S3 that interacts with the backend via API Gateway.

Integration: Ensure that the frontend can successfully send requests to the backend and display responses.

Goal:

Understand the basics of building and connecting serverless backend services with a static frontend, enabling a fully serverless web application.

1. Creating table in DynamoDB

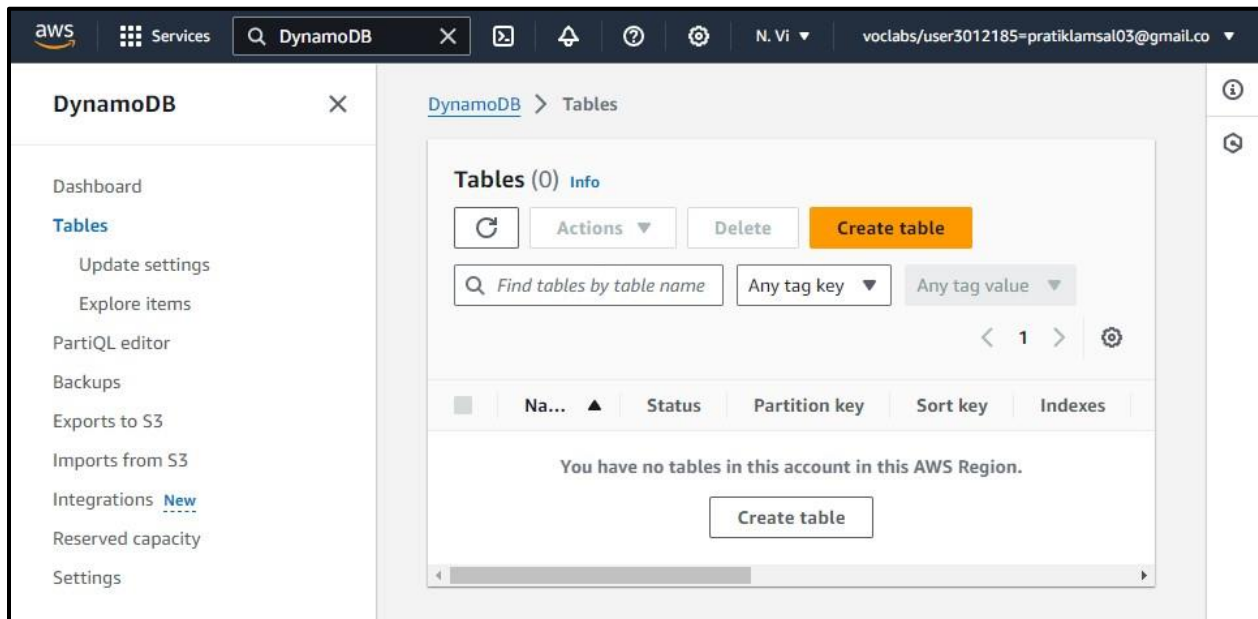


Figure 1 Table Creation

2. Table Details

Table details [Info](#)

DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

Table name
This will be used to identify your table.

Between 3 and 255 characters, containing only letters, numbers, underscores (_), hyphens (-), and periods (.).

Partition key
The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.

1 to 255 characters and case sensitive.

Sort key - optional
You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.

1 to 255 characters and case sensitive.

Figure 2 Table Details

3. Table Created Successfully

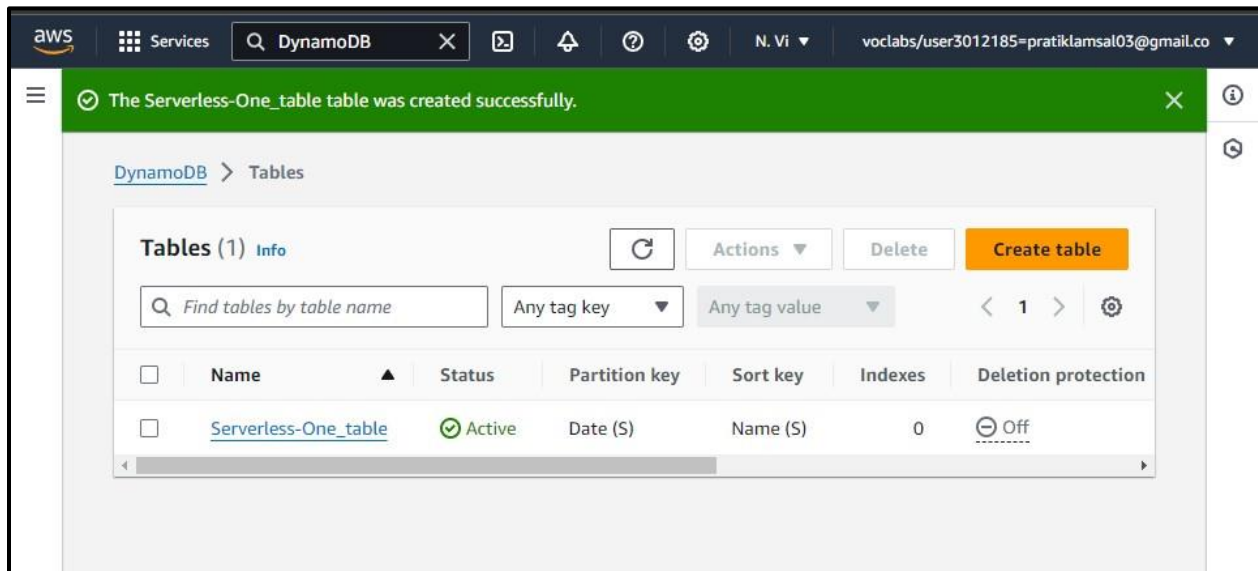


Figure 3 Successful Table Creation

4. Lambda Function Creation

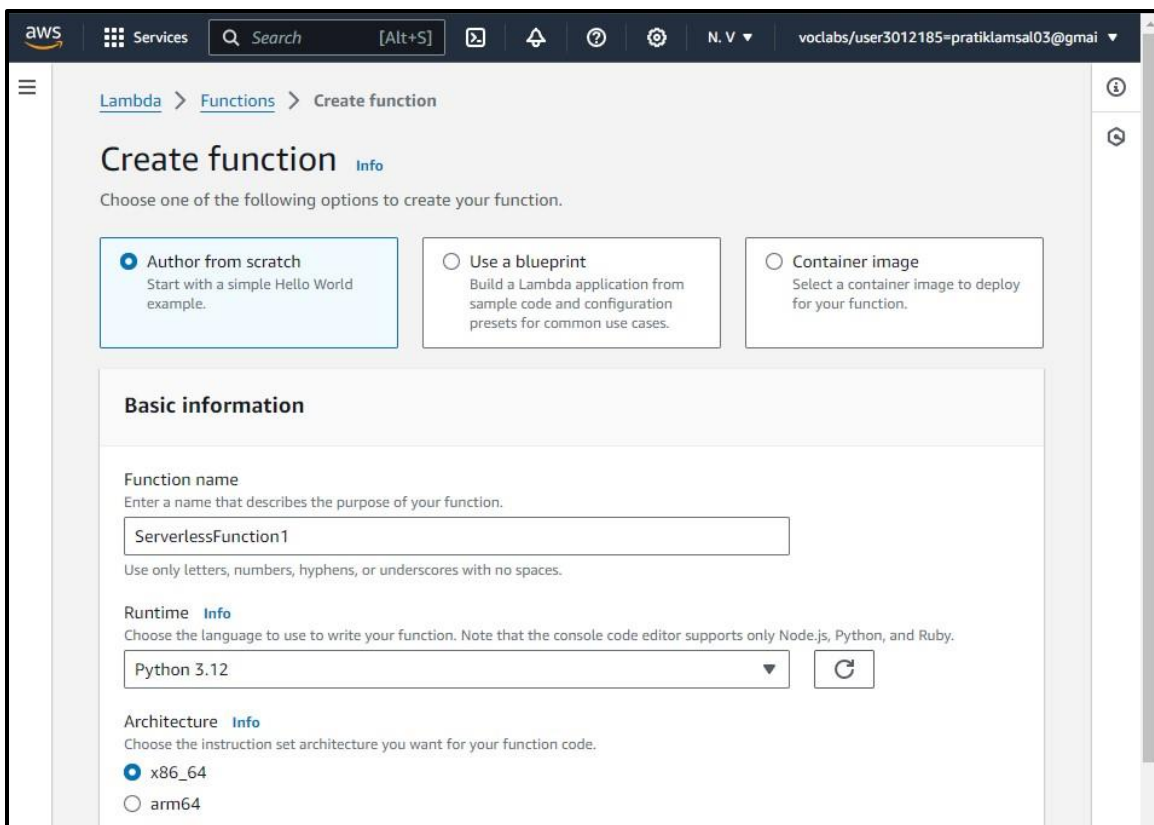


Figure 4 Lambda Function Creation

5. Permission Assignment

An existing role (LabRole) is assigned to the Function

The screenshot shows the 'Permissions' tab in the AWS Lambda console. It includes an 'Info' section about default execution roles, a 'Change default execution role' section with radio buttons for 'Create a new role with basic Lambda permissions', 'Use an existing role' (selected), and 'Create a new role from AWS policy templates'. Below, the 'Existing role' section shows 'LabRole' selected in a dropdown. A red error message at the bottom states: 'User: arn:aws:sts::553106245162:assumed-role/voclabs/user3012185=pratiklamsal03@gmail.com is not authorized to perform: iam:CreateRole on resource: arn:aws:iam::553106245162:role/service-role/ServerlessFunction1-role-837hguws because no identity-based policy allows the iam:CreateRole action'. At the bottom right are 'Cancel' and 'Create function' buttons.

Figure 5 Role Assignment

6. Successful Function Creation

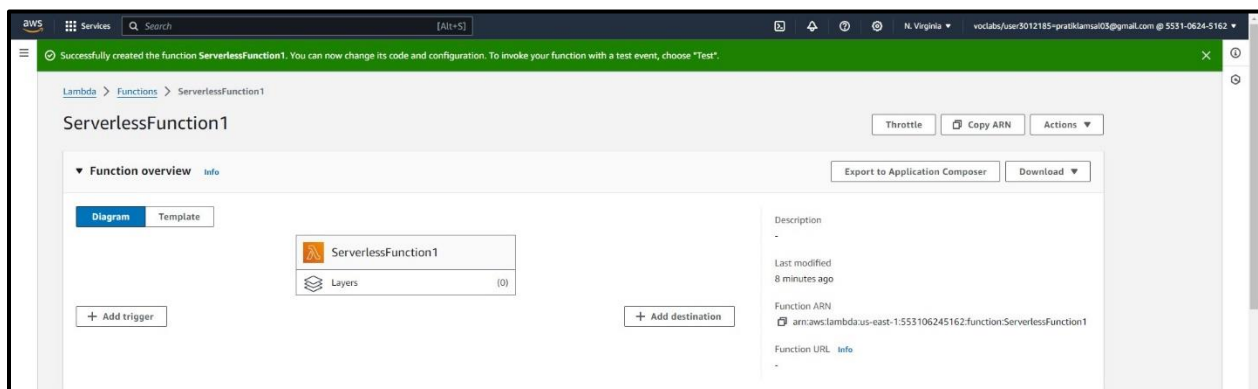


Figure 6 Function Created Successfully

7. Lambda Function Code Source

A Code Source is Added to the Lambda Function and deployed.

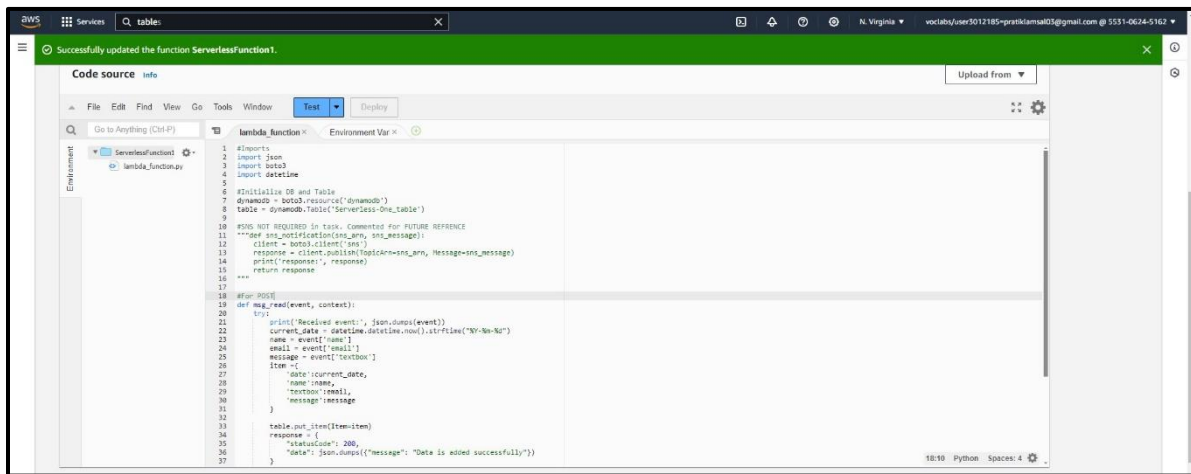


Figure 7 Lambda Function Code Source

8. REST API Creation

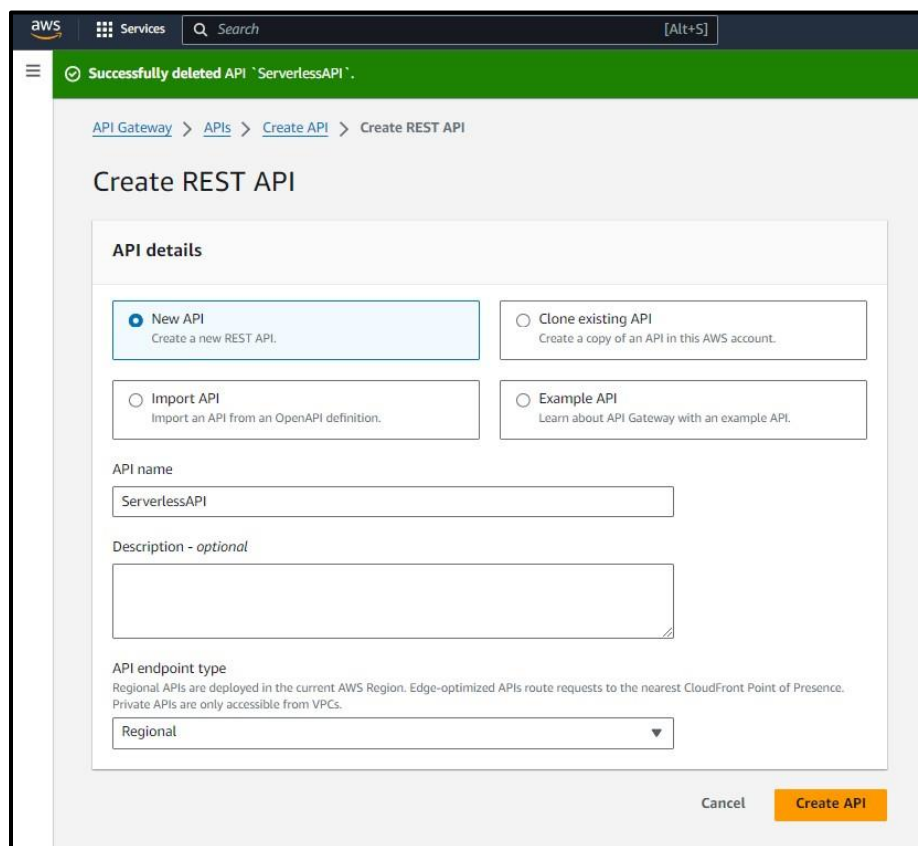


Figure 8 REST API Creation

9. Resource Creation

REST API is created successfully. Now, resources required are to be created.

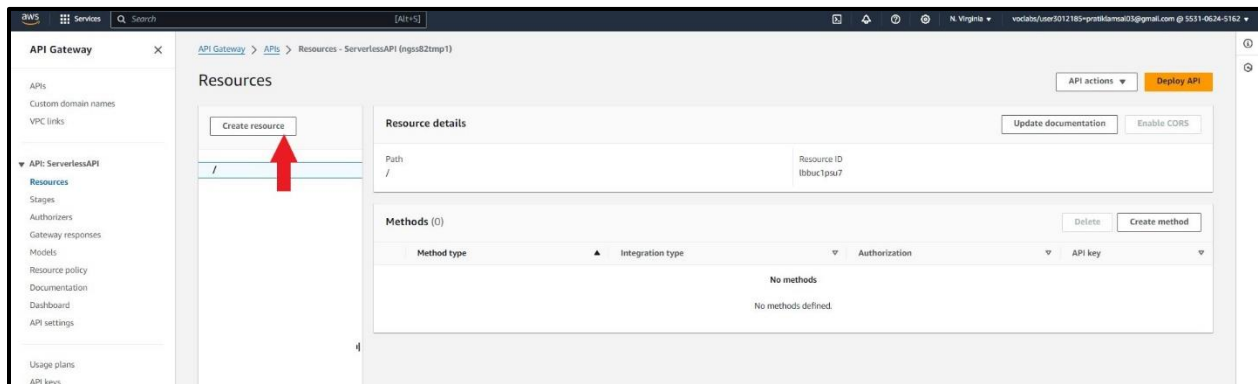


Figure 9 Resource Creation

10. Resource Details

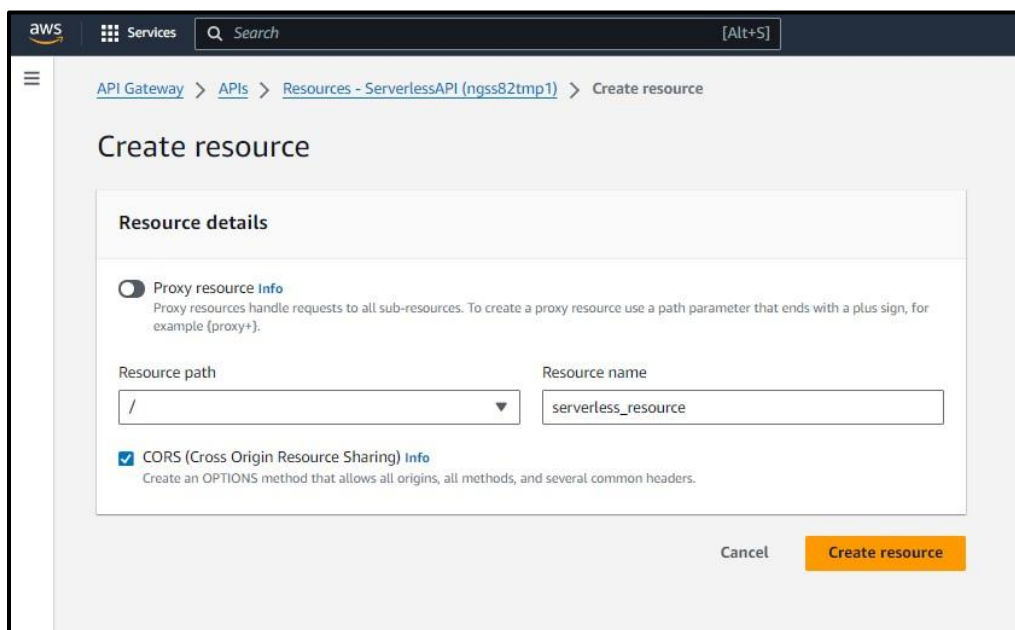


Figure 10 Resource Details

11. Resource Created Successfully

Resource is Created Successfully. Now, methods are to be created.

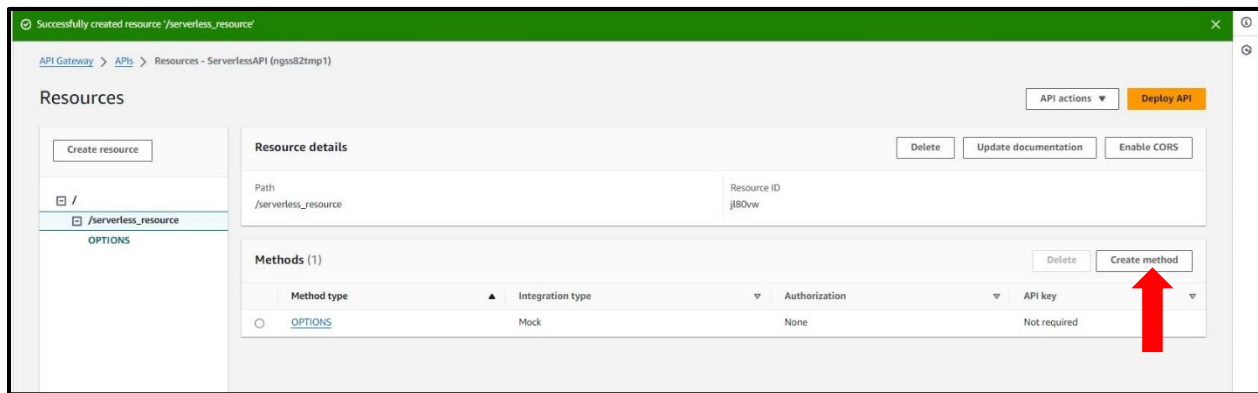


Figure 11 Successful Resource Creation

12. POST Method Creation

POST method is created and Lambda Function Created earlier is selected.

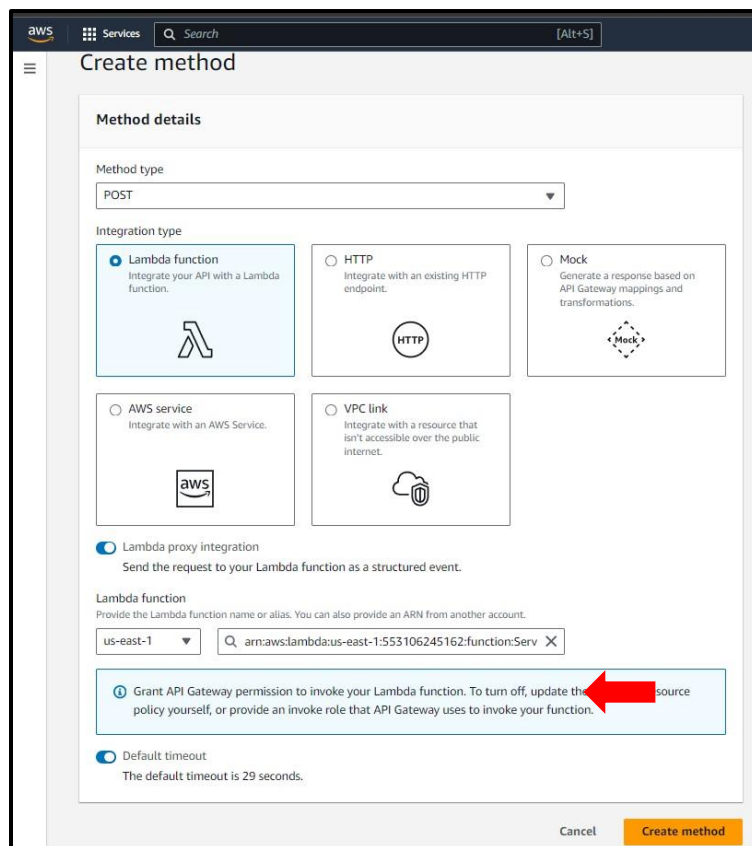


Figure 12 POST Method

13. Successful Method Creation

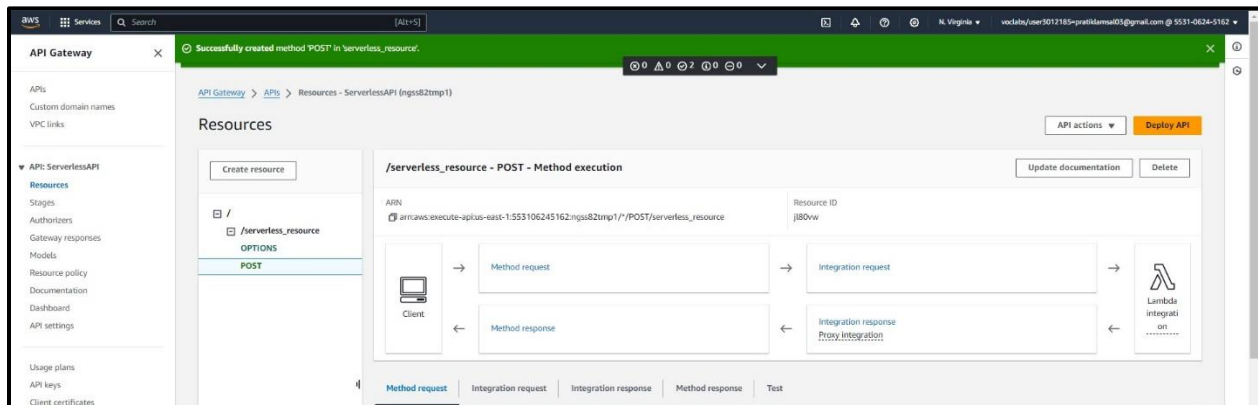


Figure 13 Successful Method Creation

14. Enabling CORS

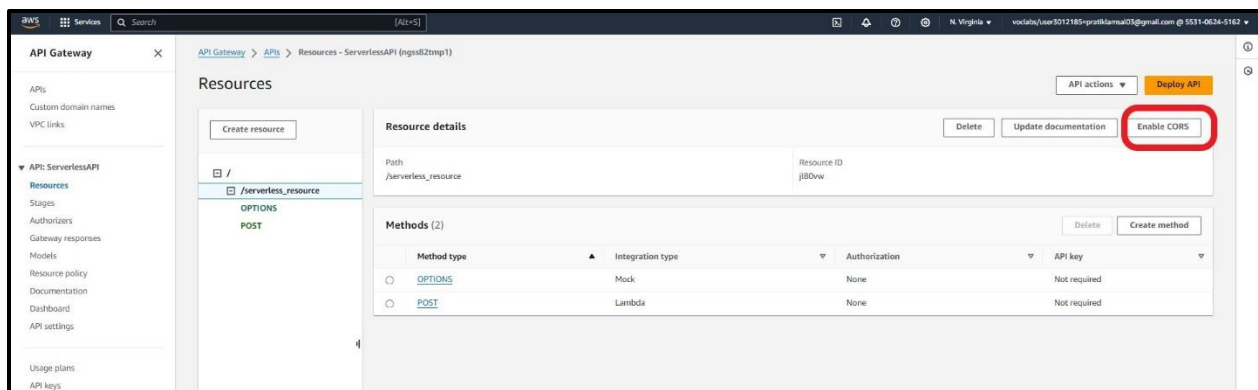


Figure 14 Enabling CORS

15. CORS Settings

The screenshot shows the 'Enable CORS' configuration page in the AWS API Gateway console. The breadcrumb trail is 'API Gateway > APIs > Resources - ServerlessAPI (ngss82tmp1) > Enable CORS'. The page title is 'Enable CORS'. Below the title, there is a section 'CORS settings Info' with a sub-header 'To allow requests from scripts running in the browser, configure cross-origin resource sharing (CORS) for your API.' The configuration options are as follows:

- Gateway responses:** API Gateway will configure CORS for the selected gateway responses. There are two checkboxes: 'Default 4XX' and 'Default 5XX', both of which are unchecked.
- Access-Control-Allow-Methods:** There are two checkboxes: 'OPTIONS' and 'POST', both of which are checked.
- Access-Control-Allow-Headers:** API Gateway will configure CORS for the selected gateway responses. The text input field contains 'Content-Type,X-Amz-Date,Authorization,X-API-Key,X-Amz-Security-Token'.
- Access-Control-Allow-Origin:** Enter an origin that can access the resource. Use a wildcard '*' to allow any origin to access the resource. The text input field contains '*'.

At the bottom of the configuration section, there is a link 'Additional settings'. At the bottom right of the page, there are two buttons: 'Cancel' and 'Save'.

Figure 15 CORS Settings

16. Successful CORS Enabling

The screenshot shows the 'Resources' page in the AWS API Gateway console. A green banner at the top indicates 'Successfully enabled CORS'. Below the banner, the breadcrumb trail is 'API Gateway > APIs > Resources - ServerlessAPI (ngss82tmp1)'. The page title is 'Resources'. On the left, there is a sidebar with a 'Create resource' button and a list of resources: '/' and '/serverless_resource'. The resource '/serverless_resource' is selected, and its details are shown on the right. The 'Resource details' section shows the path '/serverless_resource' and the resource ID 'j80vww'. The 'Methods (2)' section shows a table with two methods: 'OPTIONS' and 'POST'. The 'OPTIONS' method is selected, and its details are shown in the table below.

Method type	Integration type	Authorization	API key
<input type="radio"/> OPTIONS	Mock	None	Not required
<input type="radio"/> POST	Lambda	None	Not required

At the top right of the 'Resources' page, there is a button 'Deploy API' with a red arrow pointing to it. Below the 'Deploy API' button, there is a button 'Enable CORS'.

Figure 16 CORS Enabled Successfully

17. API Gateway for Function

API Gateway is added automatically.

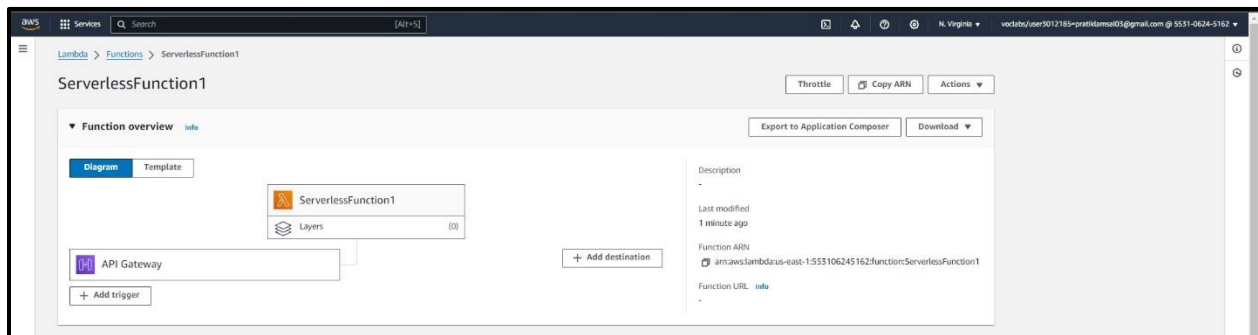


Figure 17 API Gateway Addition

18. Bucket Creation

Create bucket Info

Buckets are containers for data stored in S3. [Learn more](#)

General configuration

AWS Region

US East (N. Virginia) us-east-1

Bucket type Info

☒ General purpose

Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.

☐ Directory - New

Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.

Bucket name Info

serverlessbucketpratik

Bucket name must be unique within the global namespace and follow the bucket naming rules. [See rules for bucket naming](#)

Copy settings from existing bucket - optional

Only the bucket settings in the following configuration are copied.

Format: s3://bucket/prefix

Figure 18 Bucket Creation

19. ACL Settings

Object Ownership Info

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

☐ ACLs disabled (recommended)

All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.

☒ ACLs enabled

Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

⚠ We recommend disabling ACLs, unless you need to control access for each object individually or to have the object writer own the data they upload. Using a bucket policy instead of ACLs to share data with users outside of your account simplifies permissions management and auditing.

Object Ownership

☒ **Bucket owner preferred**

If new objects written to this bucket specify the bucket-owner-full-control canned ACL, they are owned by the bucket owner. Otherwise, they are owned by the object writer.

☐ **Object writer**

The object writer remains the object owner.


i If you want to enforce object ownership for new objects only, your bucket policy must specify that the bucket-owner-full-control canned ACL is required for object uploads. [Learn more](#) 

Figure 19 Bucket ACL Settings

20. Public Access Settings

Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

- ☐ **Block all public access**
Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.
- ☐ **Block public access to buckets and objects granted through new access control lists (ACLs)**
S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
- ☐ **Block public access to buckets and objects granted through any access control lists (ACLs)**
S3 will ignore all ACLs that grant public access to buckets and objects.
- ☐ **Block public access to buckets and objects granted through new public bucket or access point policies**
S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
- ☐ **Block public and cross-account access to buckets and objects through any public bucket or access point policies**
S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

⚠ Turning off block all public access might result in this bucket and the objects within becoming public
AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting.

☐ I acknowledge that the current settings might result in this bucket and the objects within becoming public.

Figure 20 Bucket Public Access Settings

21. Successful Bucket Creation

[illegible]

Figure 21 Successful Bucket Creation

22. API Deployment

Forgot to Deploy API in previous steps as errors occurred and new API was created again. Now Deploying by creating a new stage.

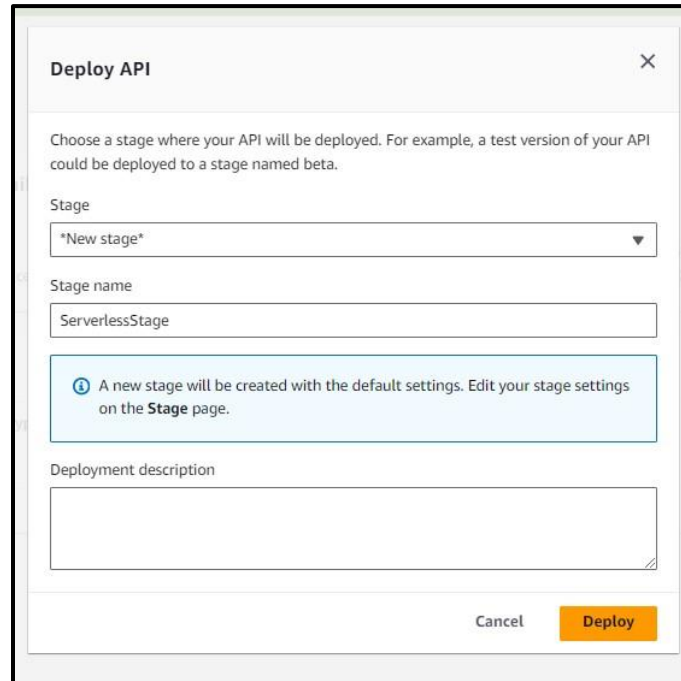


Figure 22 API Deployment

23. Stage Created

Copied for use in the code mentioned above.

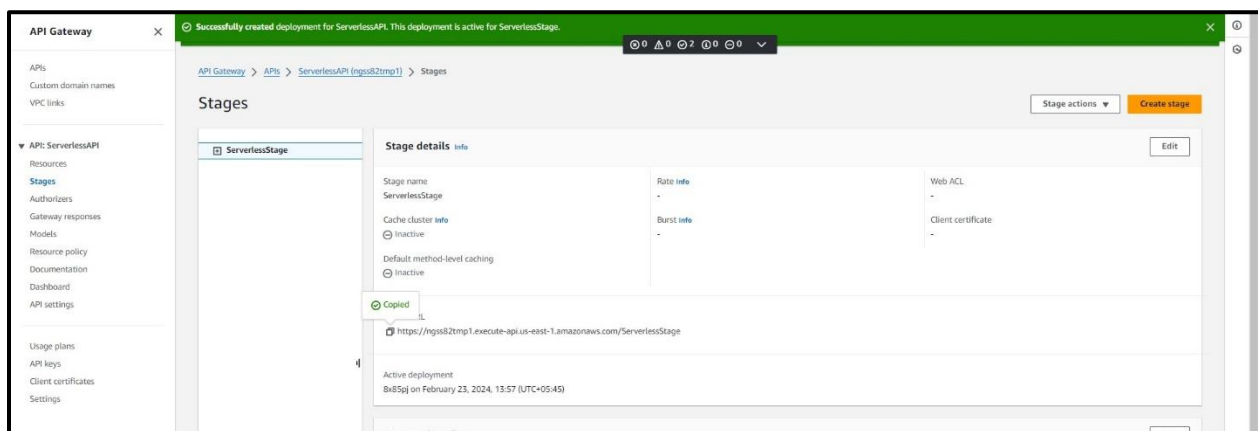


Figure 23 Stage Created Successfully

24. File Upload

HTML file uploaded to bucket.

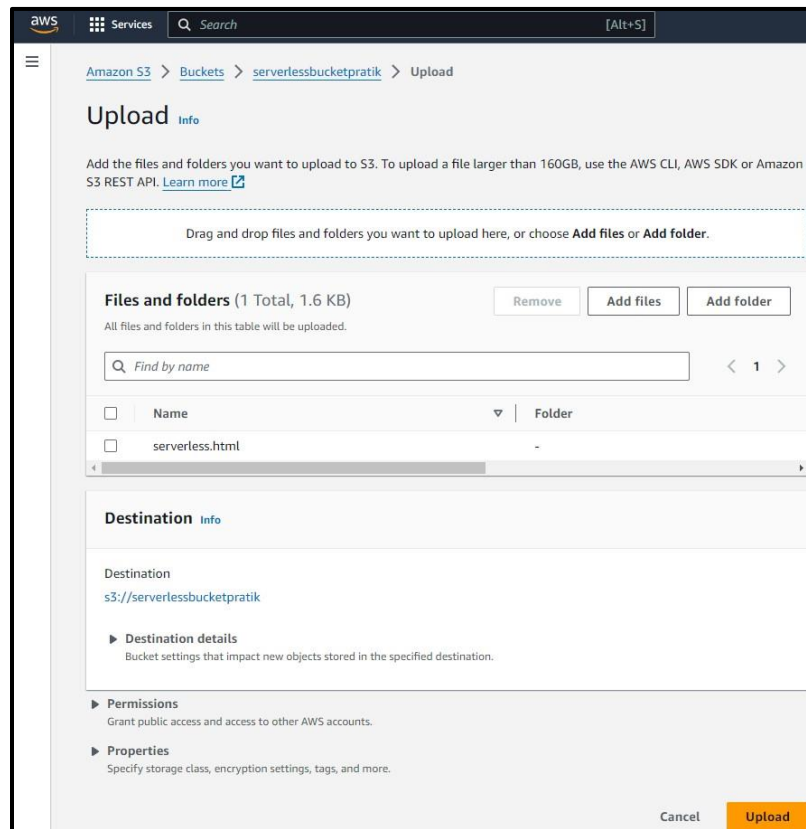


Figure 24 HTML File Upload

25. File Uploaded Successfully

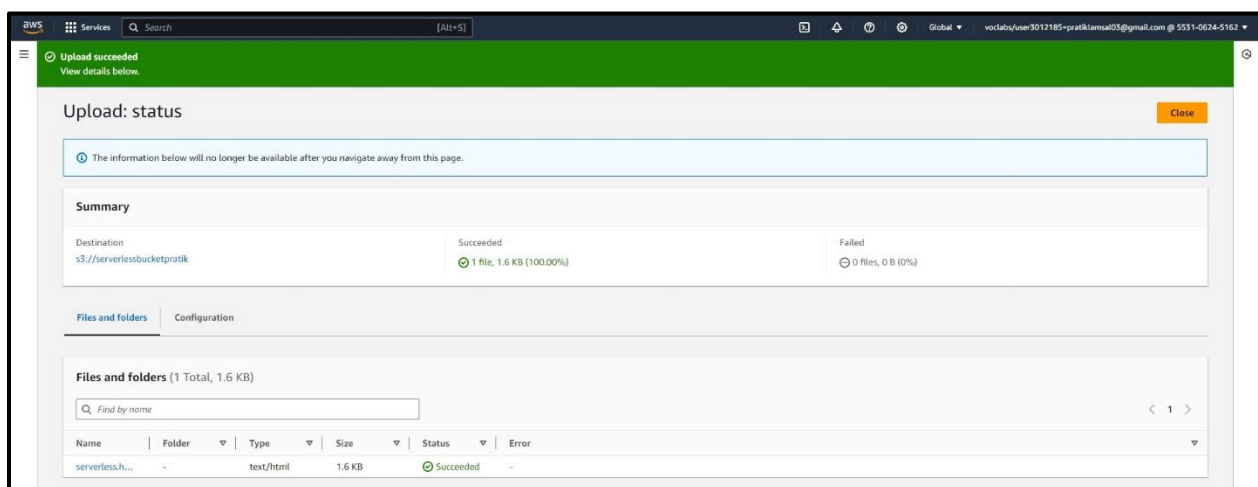


Figure 25 Successful File Upload

26. Object URL

Object URL is copied. This can be done by clicking on the object and clicking on the url.

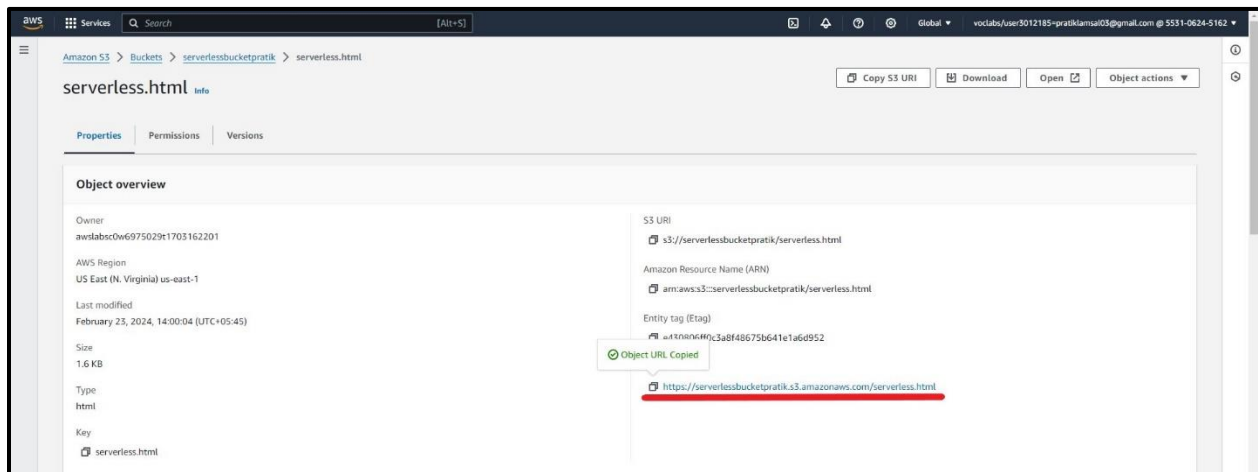


Figure 26 Object URL

27. Unsuccessful Attempt

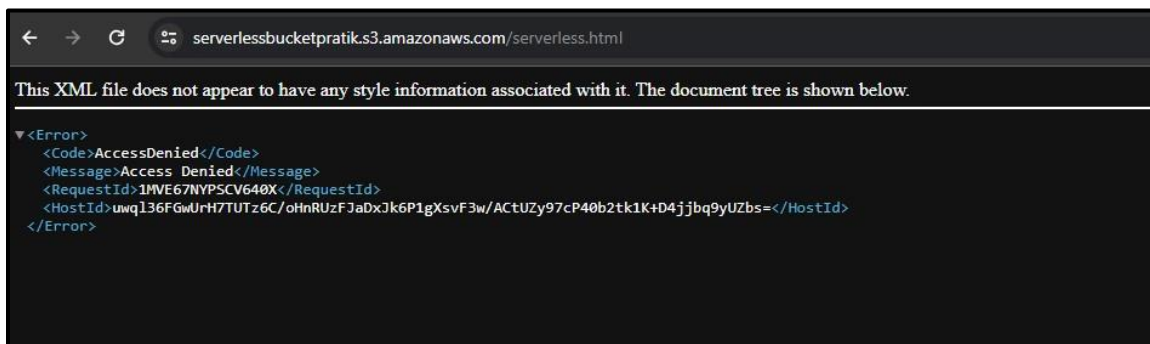


Figure 27 Unsuccessful Attempt

28. Bucket ACL

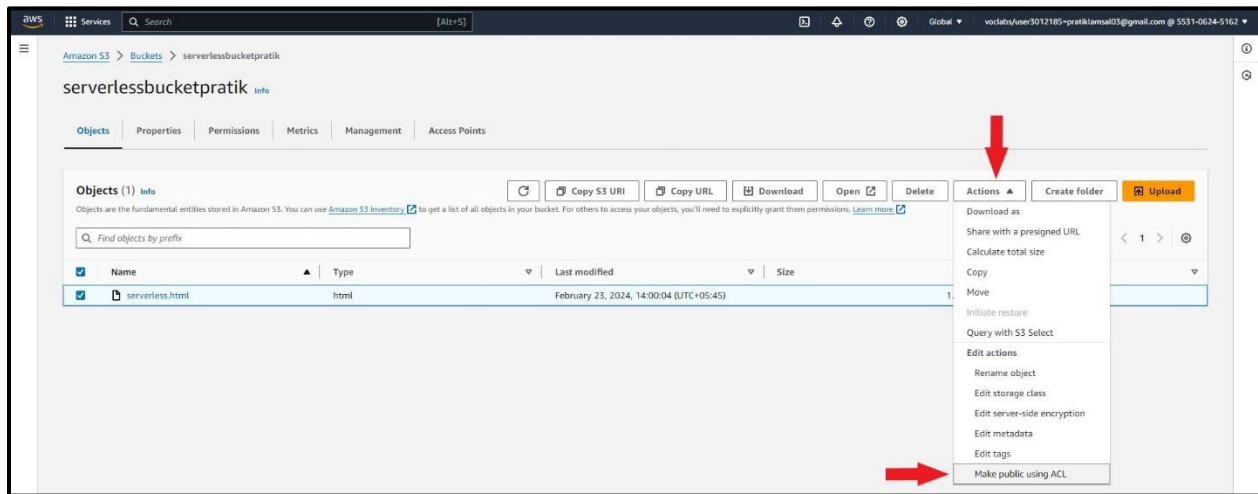


Figure 28 Changing Bucket ACL

29. Public Bucket

Bucket is made public.

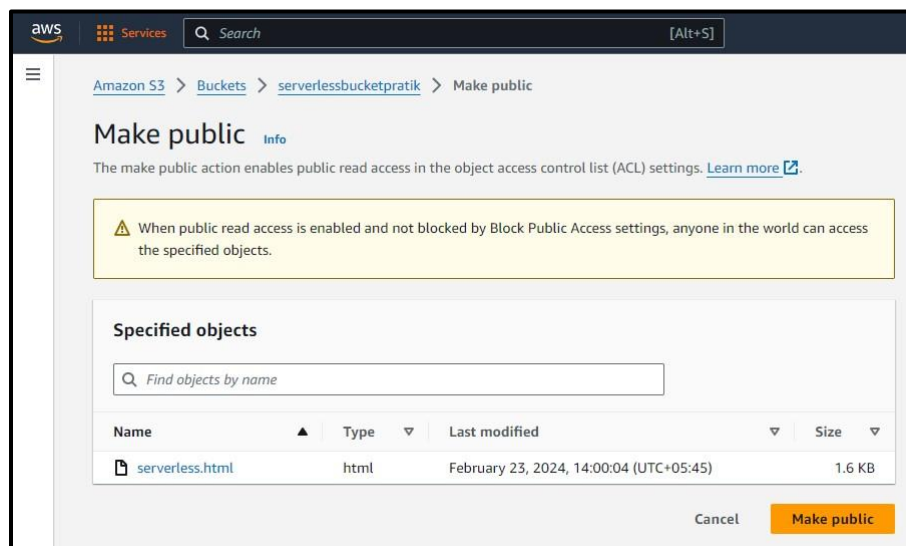


Figure 29 Public Bucket

30. Made Public Successfully

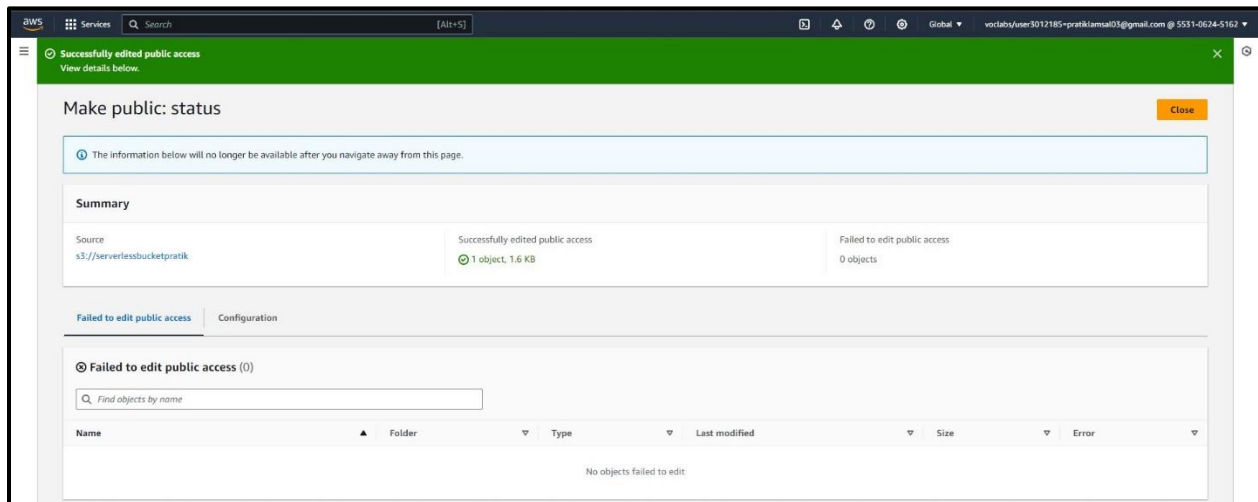


Figure 30 Successfully Made Public

31. Form Open Successful

The screenshot shows a web browser window with the URL `serverlessbucketpratik.s3.amazonaws.com/serverless.html`. The page displays an "Enquiry Form" with the following fields:

- Name:
- Email:
- Message:
- Submit:

Figure 31 Form Opens Successful

32. Static Website Hosting

Further error occurred. Changing setting by going into bucket settings and scrolling a little down. Edit static website hosting is there. Click the EDIT button and change settings.

Amazon S3 > Buckets > serverlessbucketpratik > Edit static website hosting

Edit static website hosting [Info](#)

Static website hosting
Use this bucket to host a website or redirect requests. [Learn more](#)

Static website hosting

☐ Disable

☒ Enable

Hosting type

☒ Host a static website
Use the bucket endpoint as the web address. [Learn more](#)

☐ Redirect requests for an object
Redirect requests to another bucket or domain. [Learn more](#)

Info For your customers to access content at the website endpoint, you must make all your content publicly readable. To do so, you can edit the S3 Block Public Access settings for the bucket. For more information, see [Using Amazon S3 Block Public Access](#)

Index document
Specify the home or default page of the website.

serverless.html

Error document - optional
This is returned when an error occurs.

error.html

Redirection rules - optional
Redirection rules, written in JSON, automatically redirect webpage requests for specific content. [Learn more](#)

Figure 32 Edit Static Website Hosting

33. Log Events

Data entered in the page. On checking logs in CloudWatch, no errors are seen and payload is received.

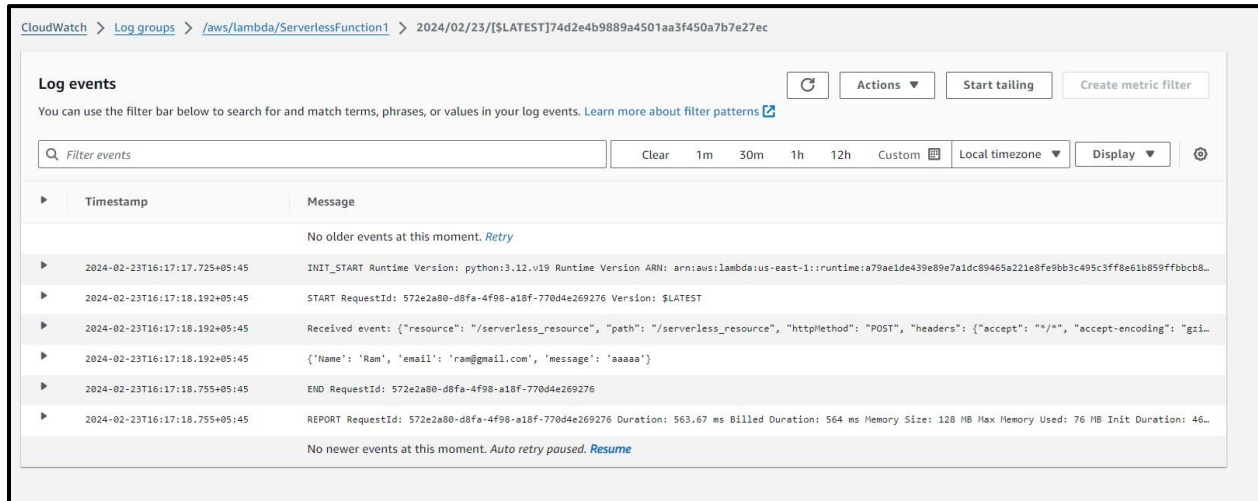


Figure 33 Log Events

34. Inspecting Network in Browser

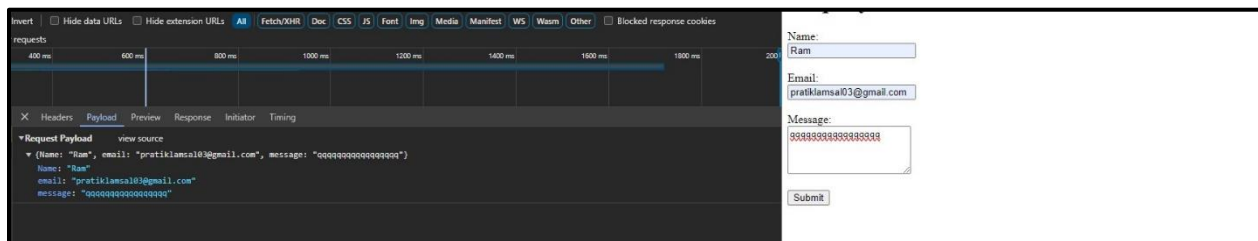


Figure 34 Payload while inspecting in browser

35. Data in Table

Data is stored in the table successfully.

Serverless-One_table

Autopreview

View table details

▼ Scan or query items

☒ Scan

☐ Query

Select a table or index

Table - Serverless-One_table ▼

Select attribute projection

All attributes ▼

► Filters

Run

Reset

✓ Completed. Read capacity units consumed: 0.5

×

Items returned (2)

↺

Actions ▼

Create item

< 1 >

⚙️

✖

<input type="checkbox"/>	Date (String) ▼	Name (String) ▼	email ▼	message ▼
<input type="checkbox"/>	2024-02-23	Ram	pratiklamsa...	qqqqqqqqqqqqqqqq
<input type="checkbox"/>	2024-02-23	fghjkl;	ram@gmail...	qqqqqqqqqqqq

Figure 35 Data in Table

36. TASK COMPLETED