Serverless Labs

1. 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.

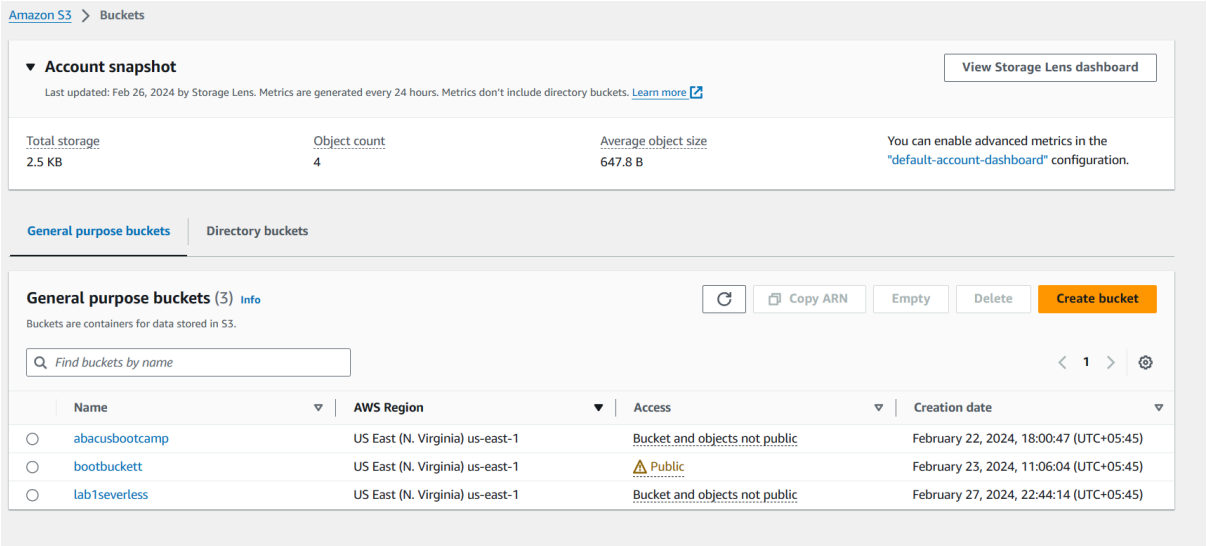
• **API Gateway**: Set up API Gateway to create RESTful endpoints that trigger the Lambda functions.

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

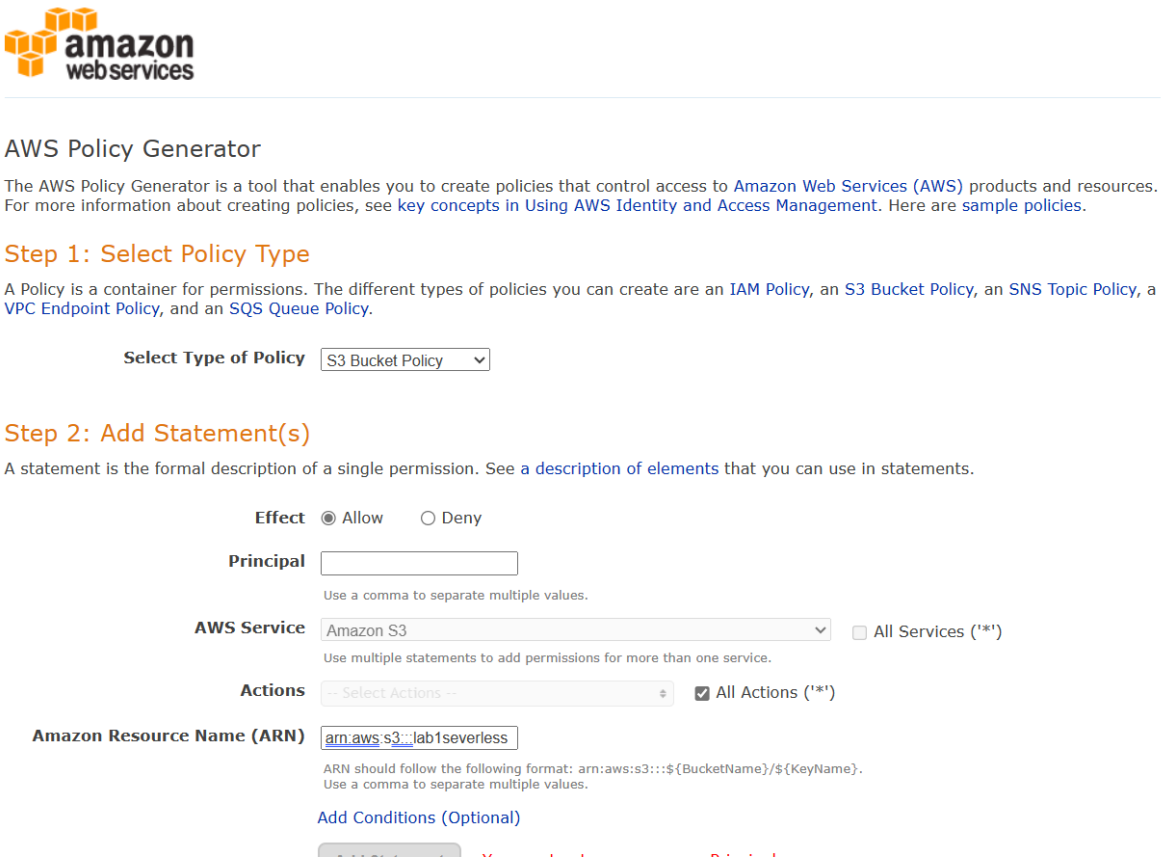
• **Integration:** Ensure that the frontend can successfully send requests to the backend and display responses.

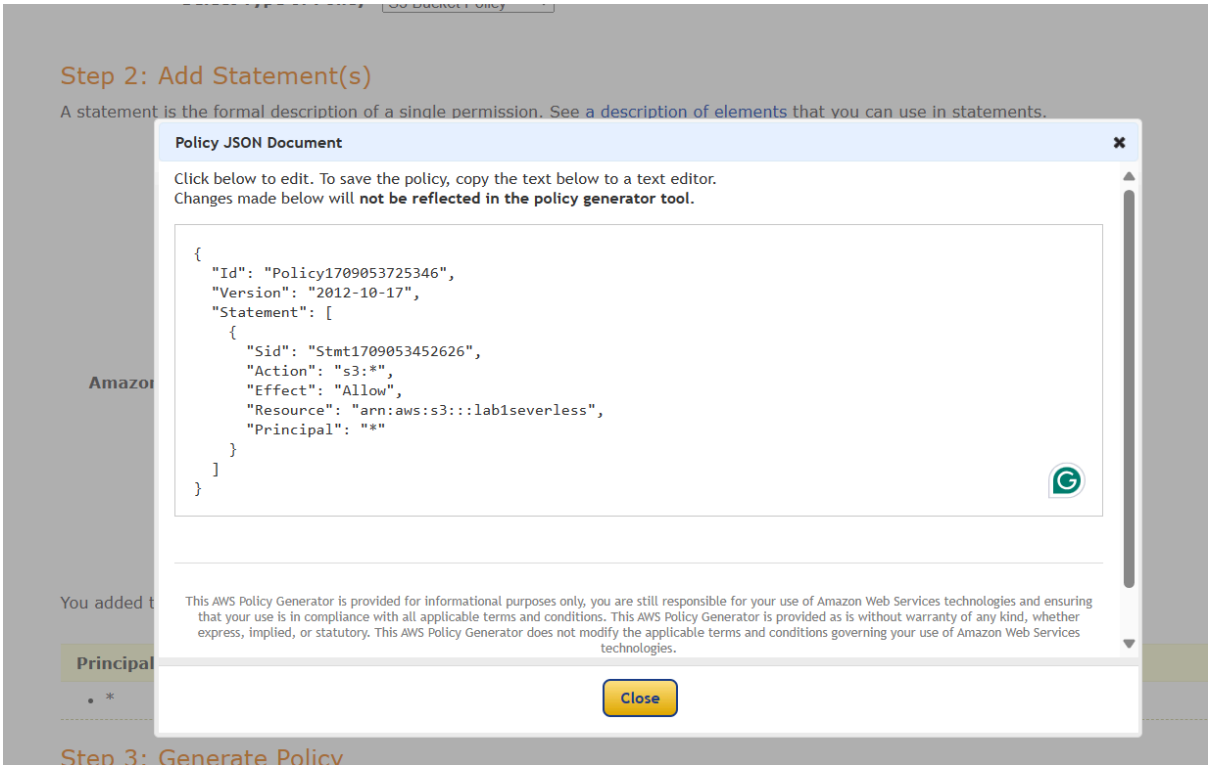
Goal: Understand the basics of building and connecting serverless backend services with a static frontend, enabling a fully serverless web application.

creating an S3 bucket with the required details and policies. After the bucket creation, we'll proceed to upload the static webpage file and activate static hosting by setting its property to 'true.'

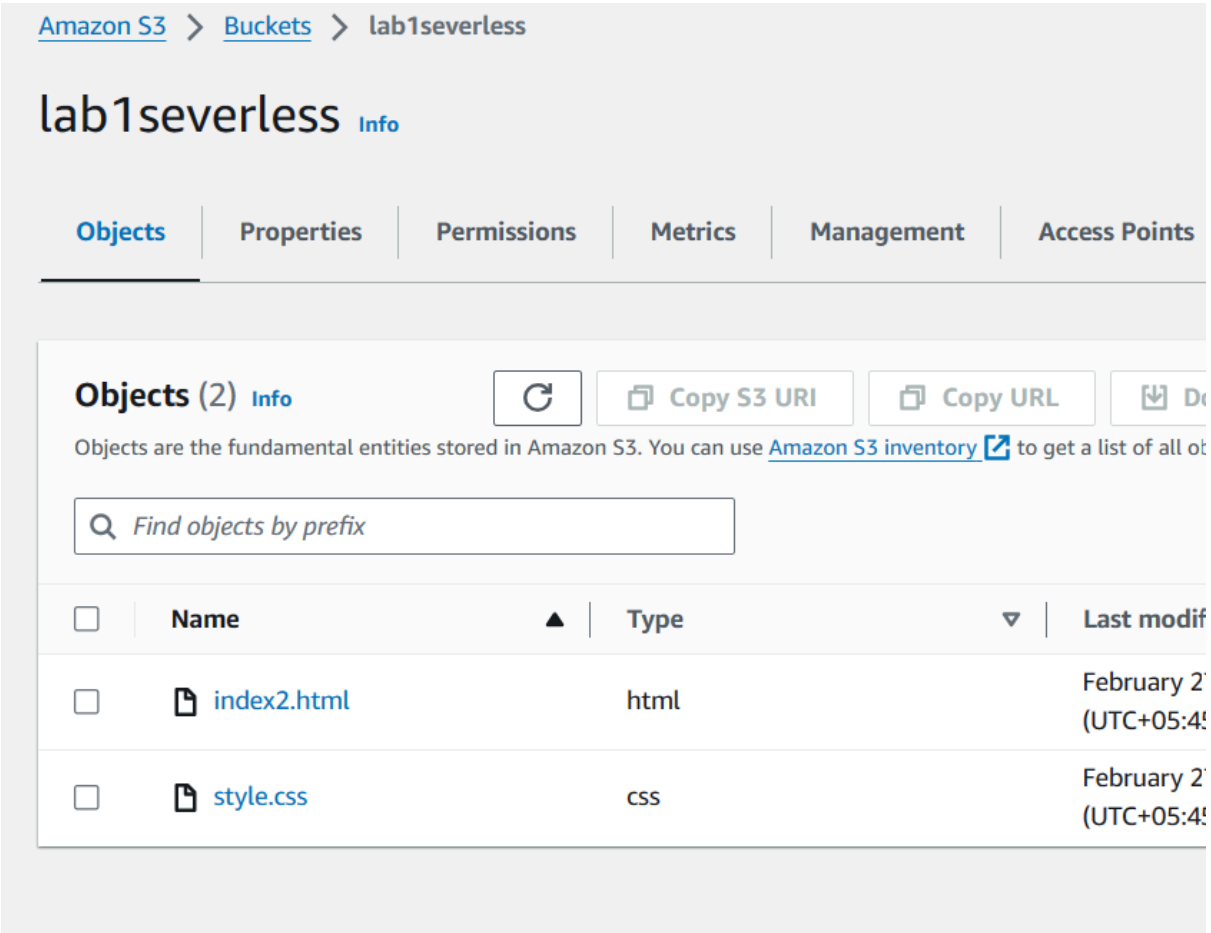


Generating web services policy

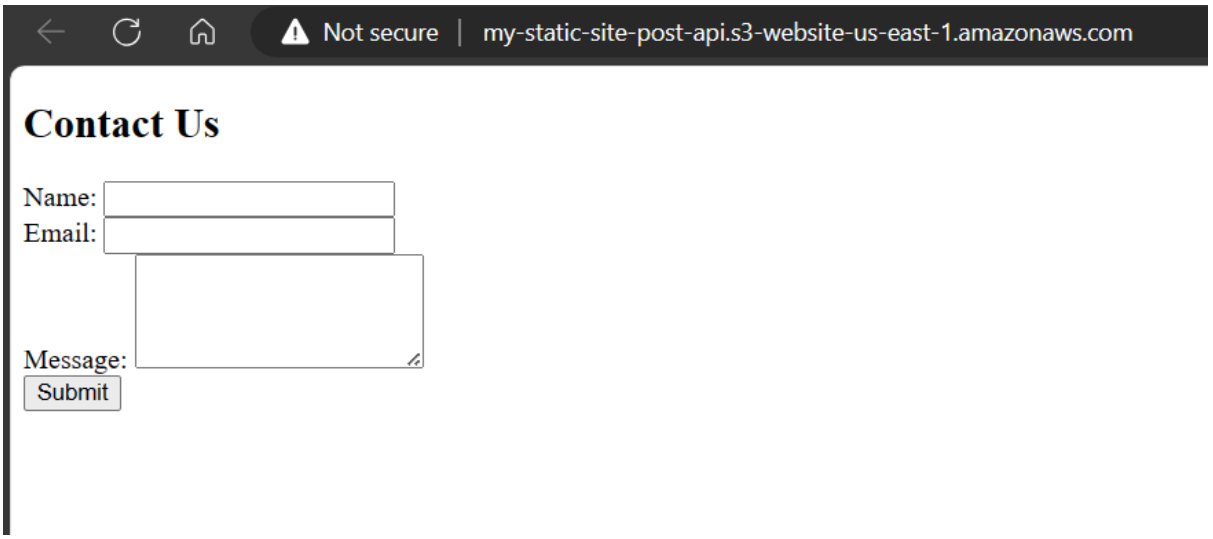




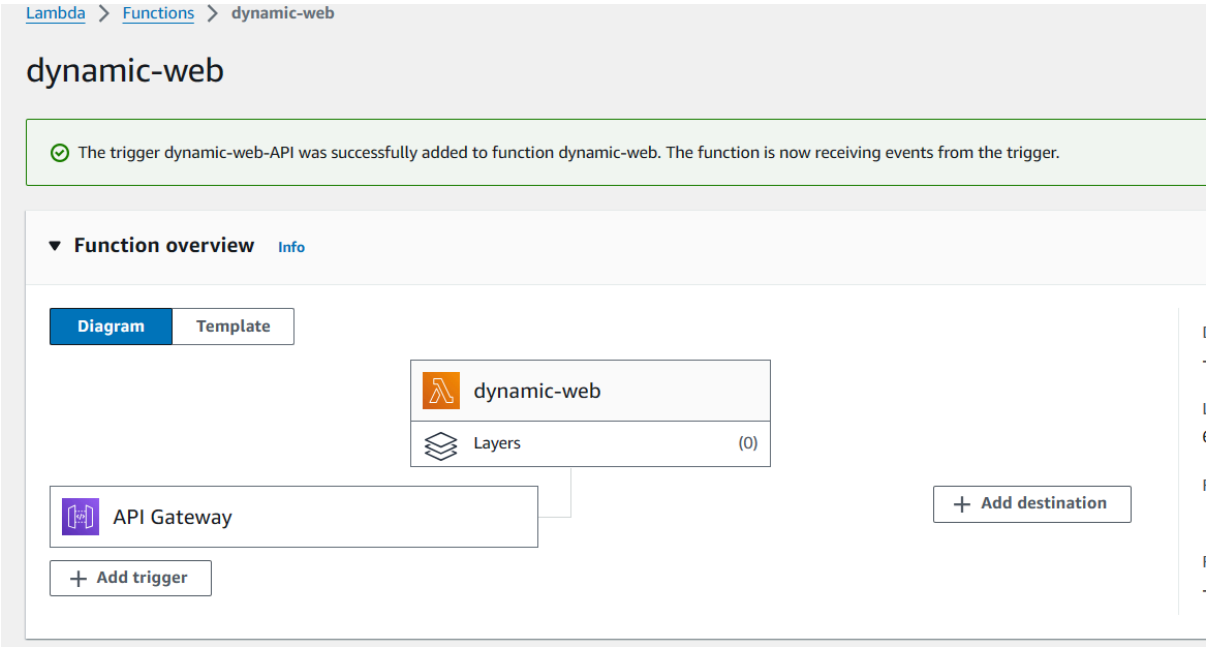
Uploading file in bucket



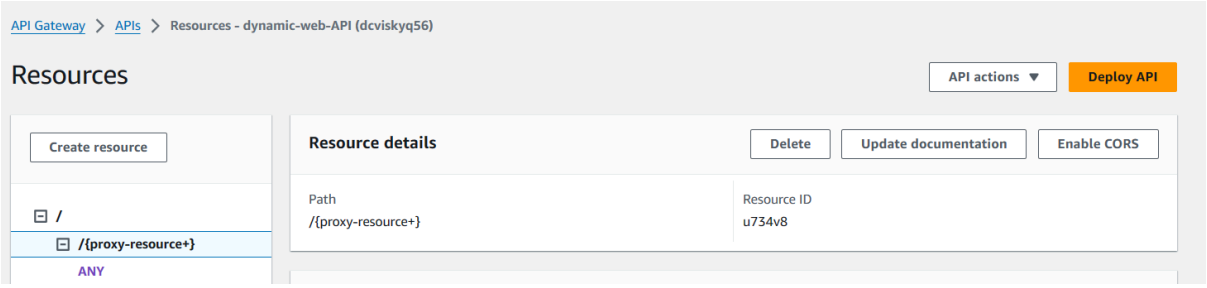
The static website is as



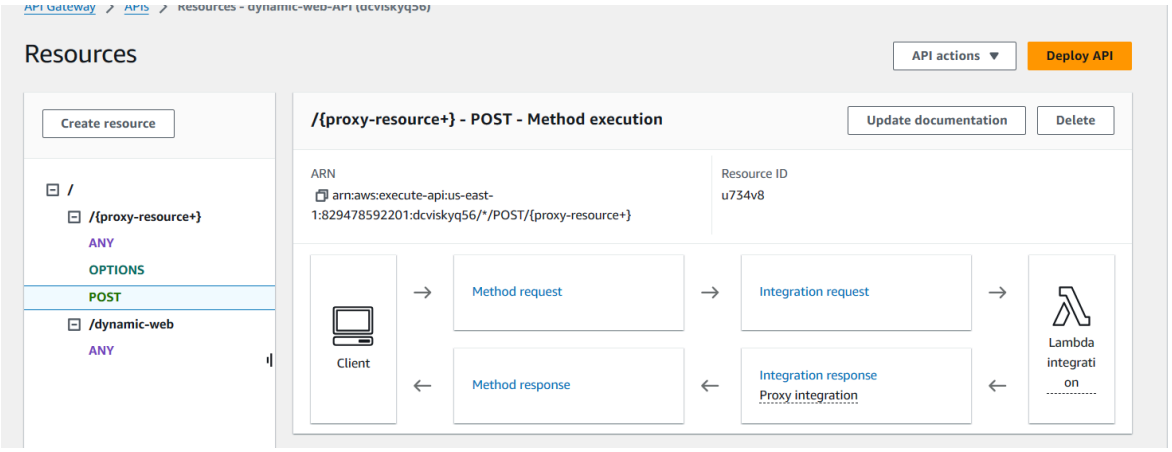
Now, we create a lambda function and add api gateway as

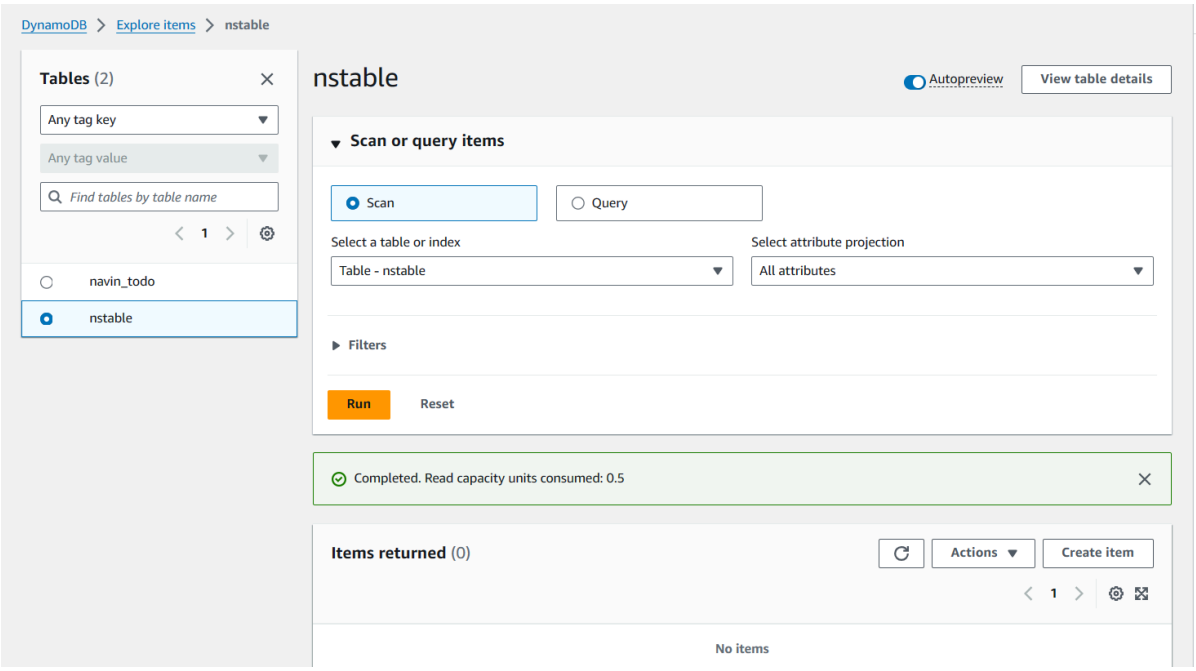


We created a proxy resource in api as

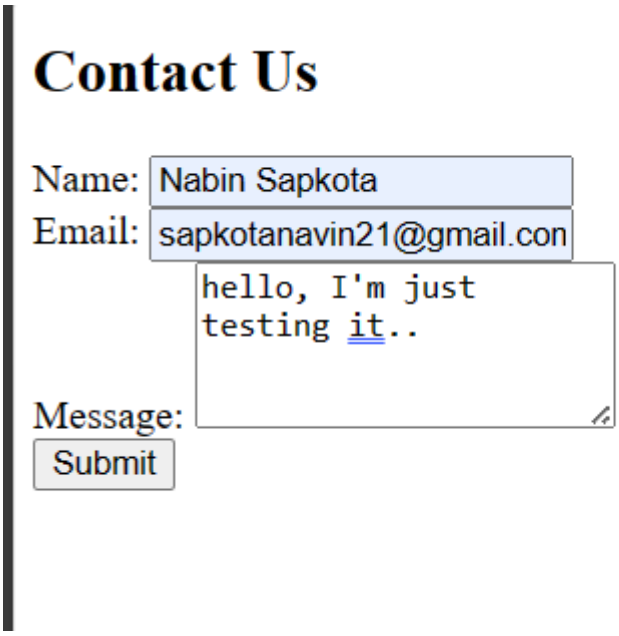


Now, we create a post method as

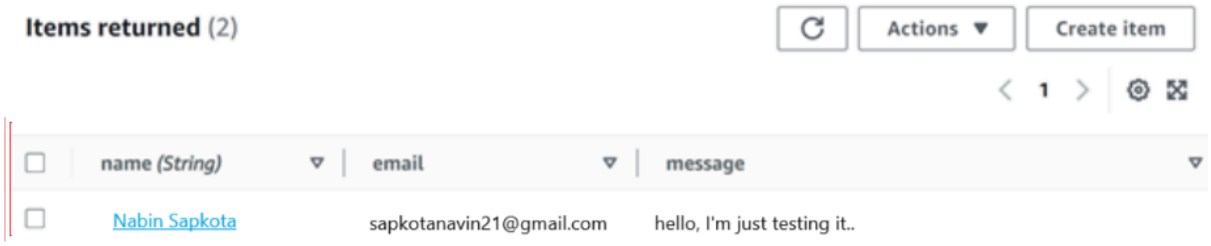




Now, we are going to fill the form from the static website as



We can see that our post has been registered in our Bucket.



2. Creating a Serverless API

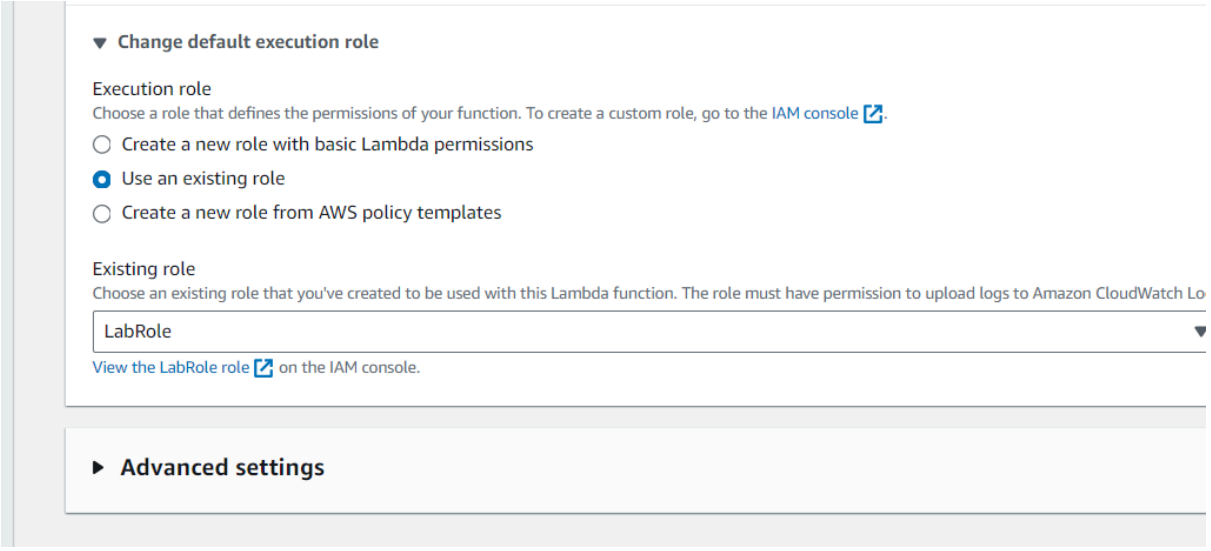
Objective: Develop a serverless API using AWS Lambda and API Gateway.

Approach:

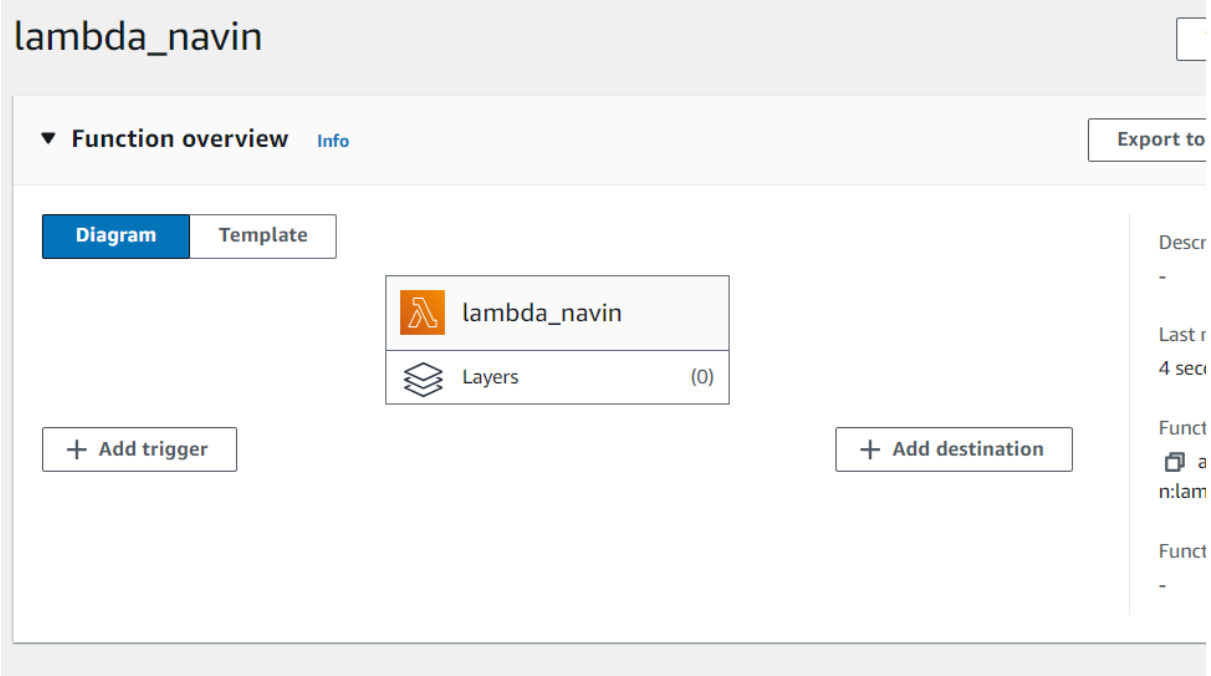
- **Define API:** Design a simple RESTful API (e.g., for a todo list application).
- **Lambda Functions:** Create Lambda functions for each API method (GET, POST, PUT, DELETE).
- **API Gateway Setup:** Use API Gateway to set up the API endpoints, connecting each endpoint to the corresponding Lambda function.
- **Testing:** Test the API using tools like Postman or AWS API Gateway test functionality.

Goal: Gain hands-on experience in building and deploying a serverless API, understanding the integration between Lambda and API Gateway

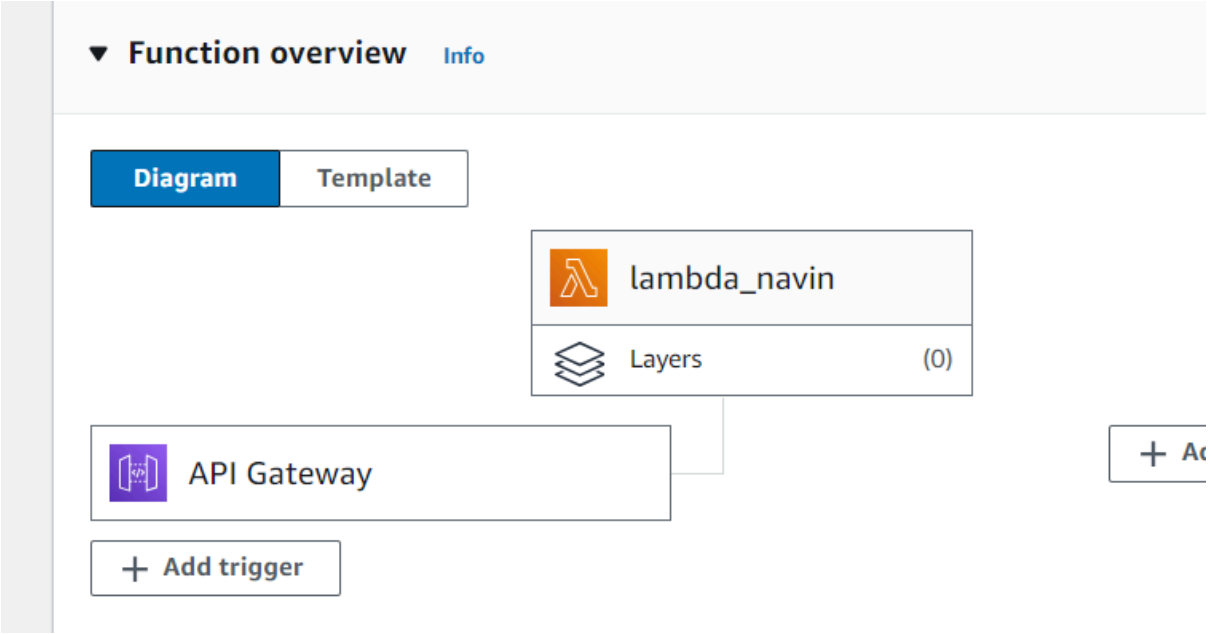
Creating a Lambda function



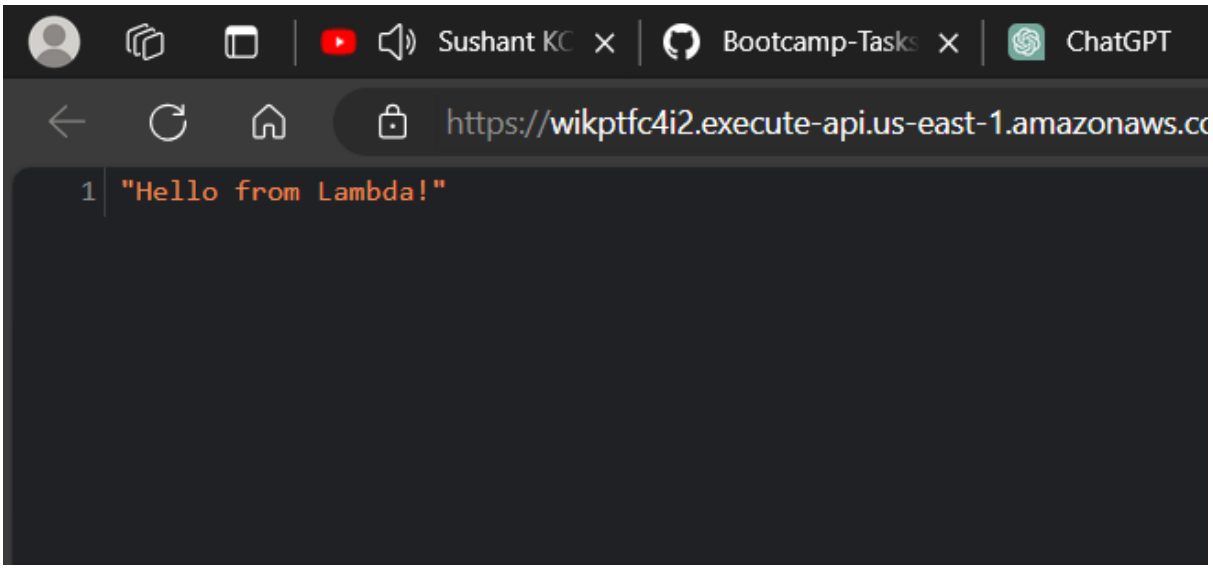
The lambda is created as



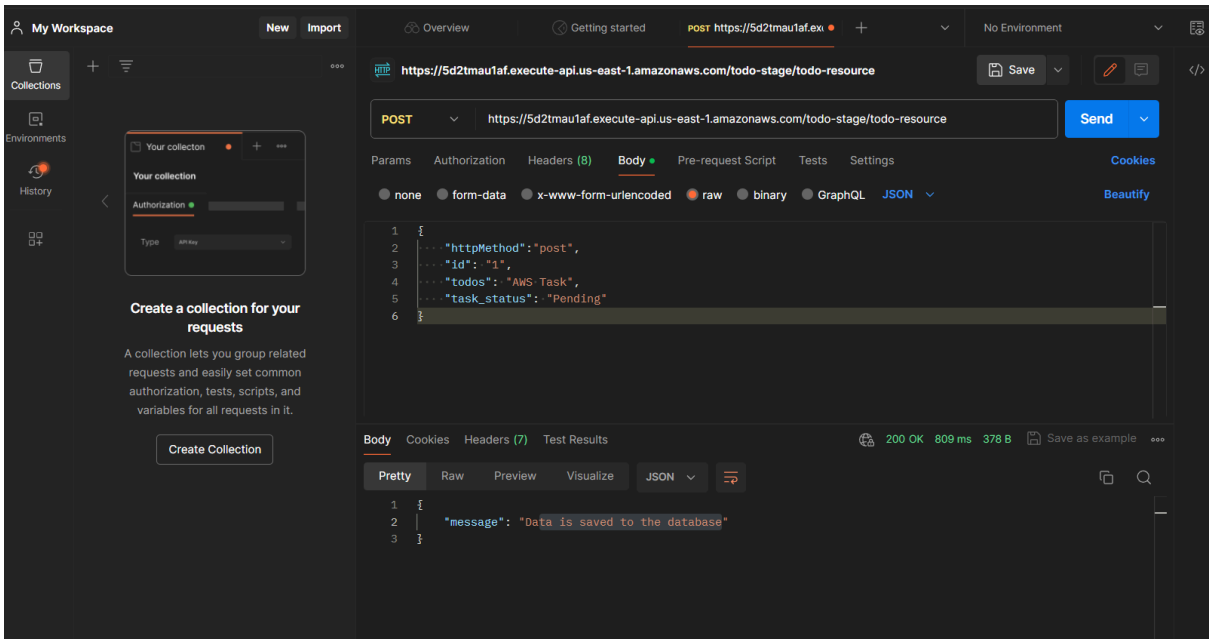
Adding a API gateway as

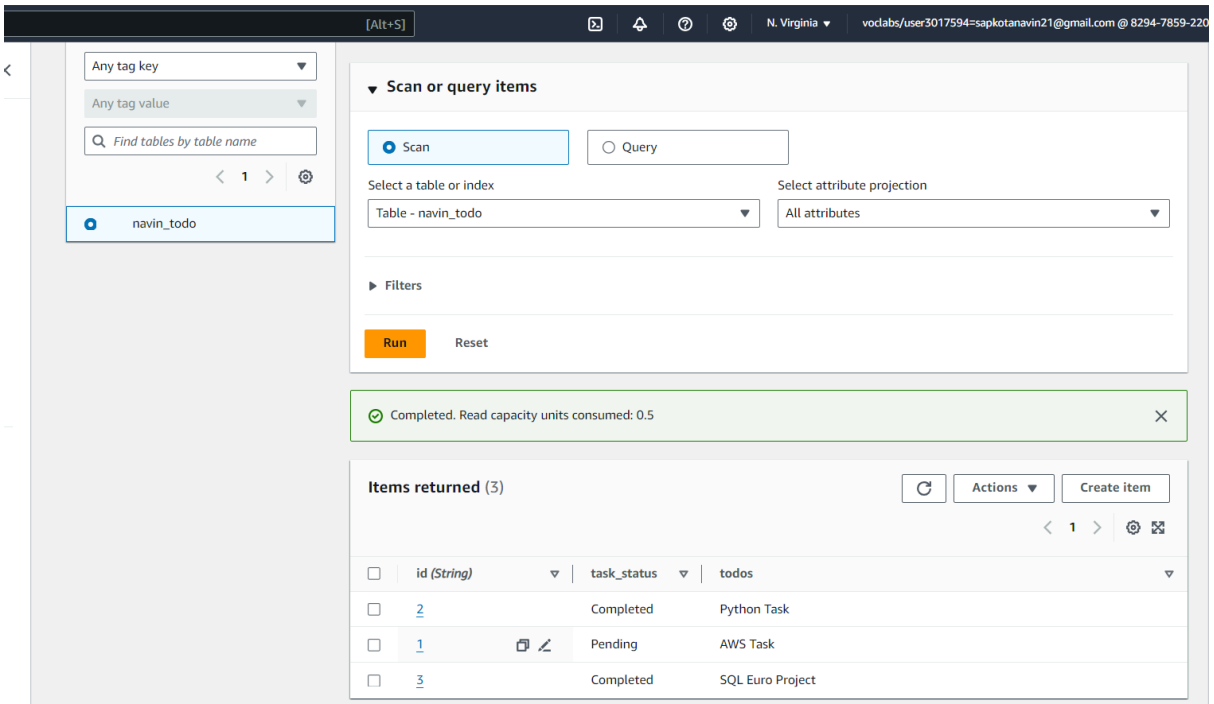


Response from API Gateway



Creating a Post, Get, POst and Put Functions as





Get method and message for get request

GET https://5d2tmau1af.execute-api.us-east-1.amazonaws.com/todo-stage/todo-resource Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

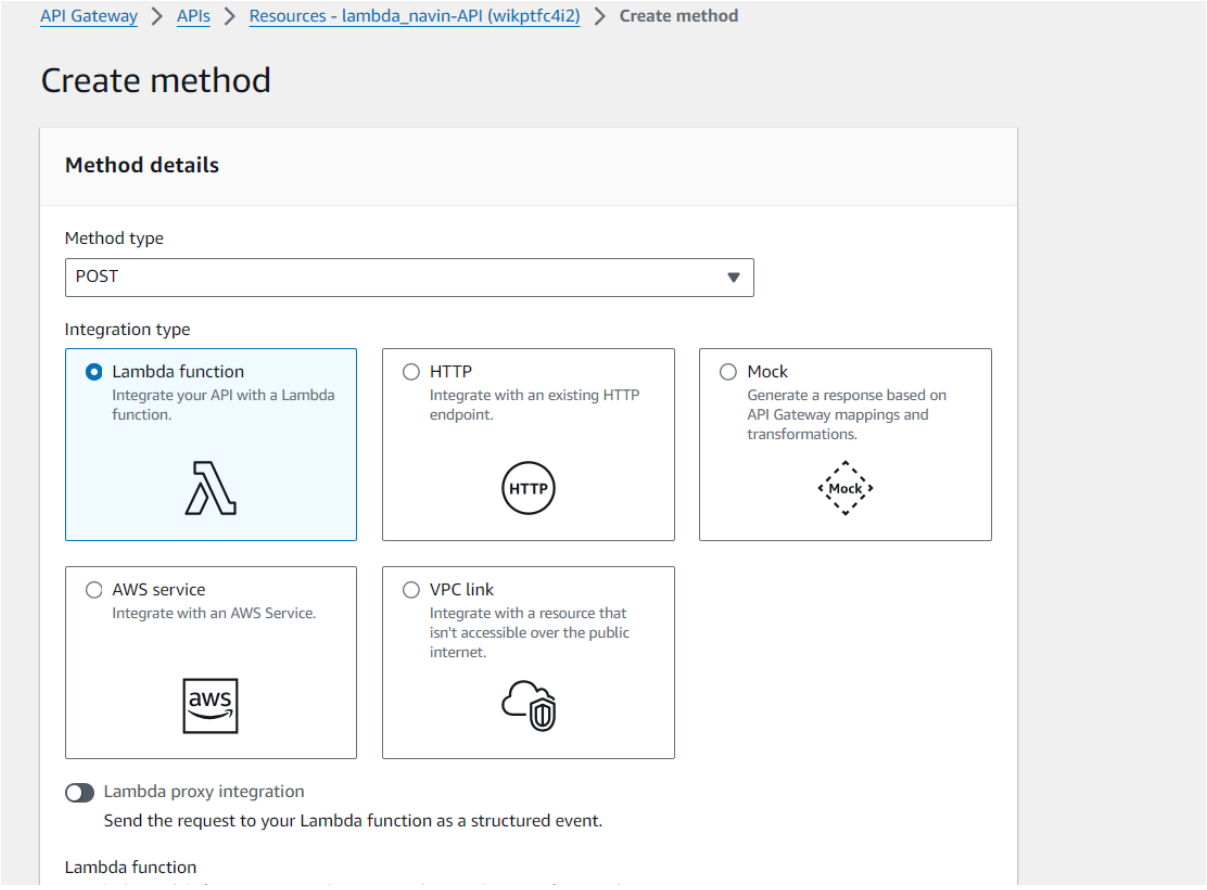
```
{
    "httpMethod":"get"
}
```

Body Cookies Headers (7) Test Results 200 OK 1411 ms 573 B Save as example

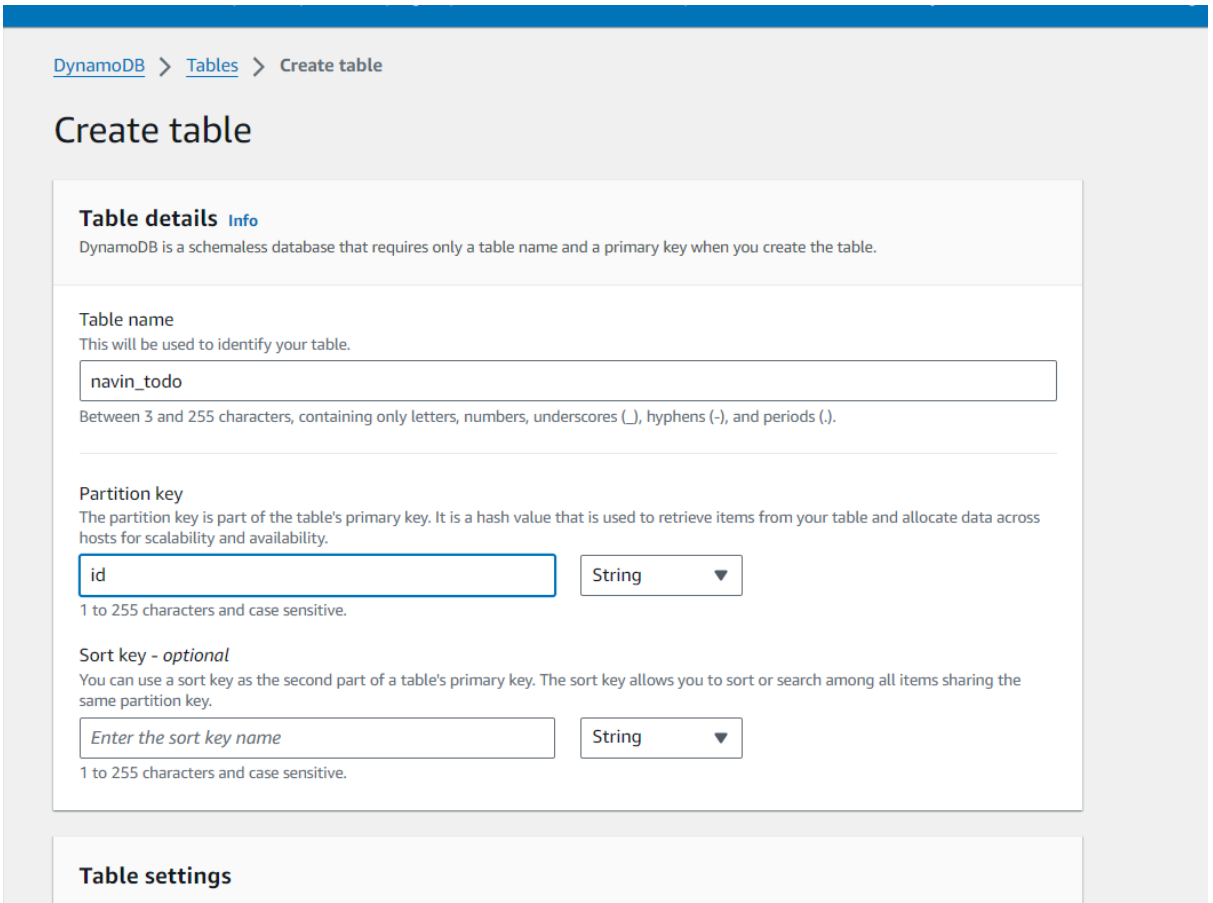
Pretty Raw Preview Visualize JSON


```
{
    "message": "Data from DynamoDB",
    "Data": [
        {
            "task_status": "Completed",
            "id": "2",
            "todos": "Python Task"
        },
        {
            "task_status": "Pending",
            "id": "1",
            "todos": "AWS Task"
        },
        {
            "task_status": "Completed",
            "id": "3",
            "todos": "SQL Euro Project"
```

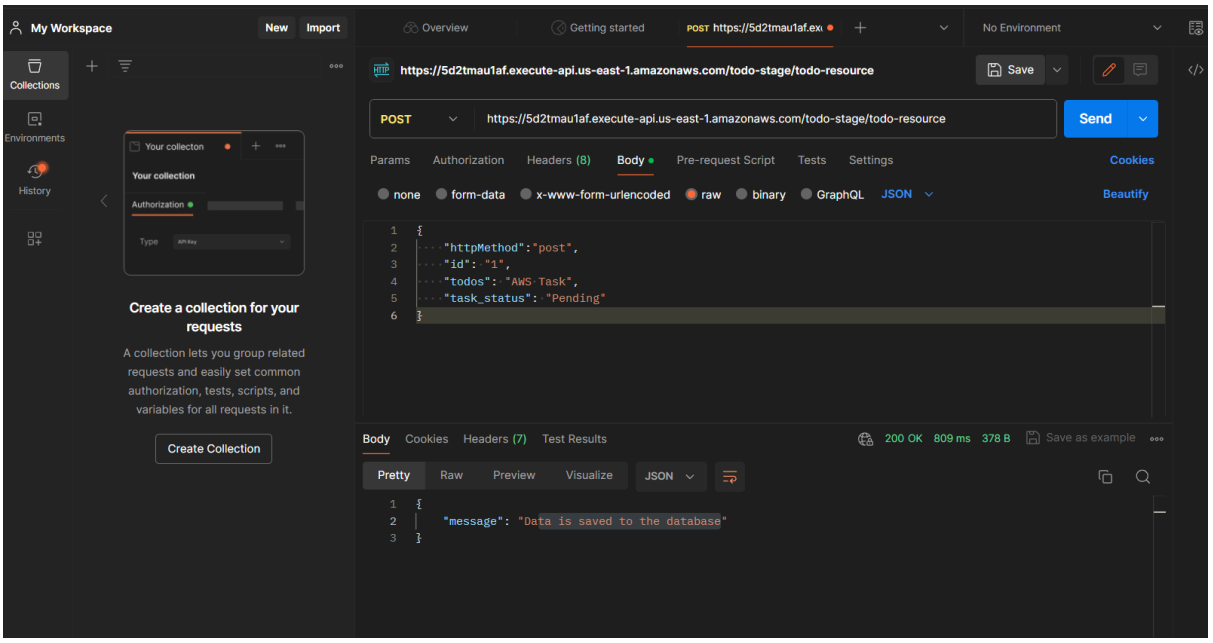
Creation of Post method as

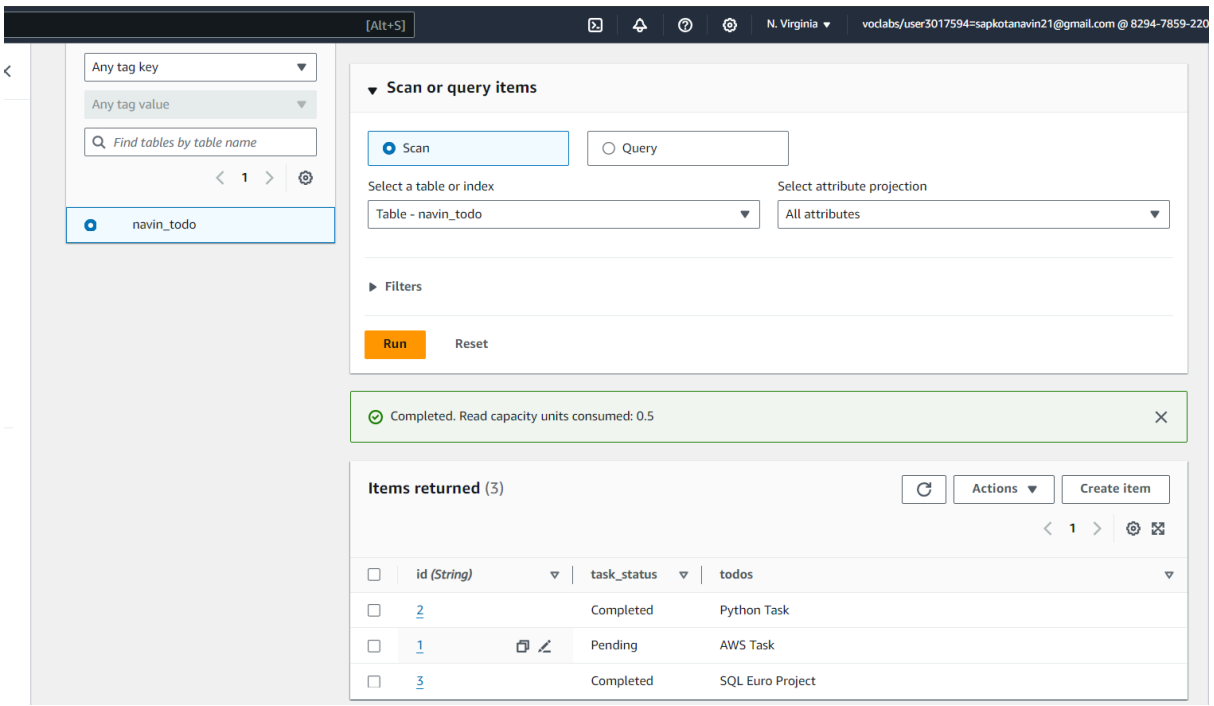


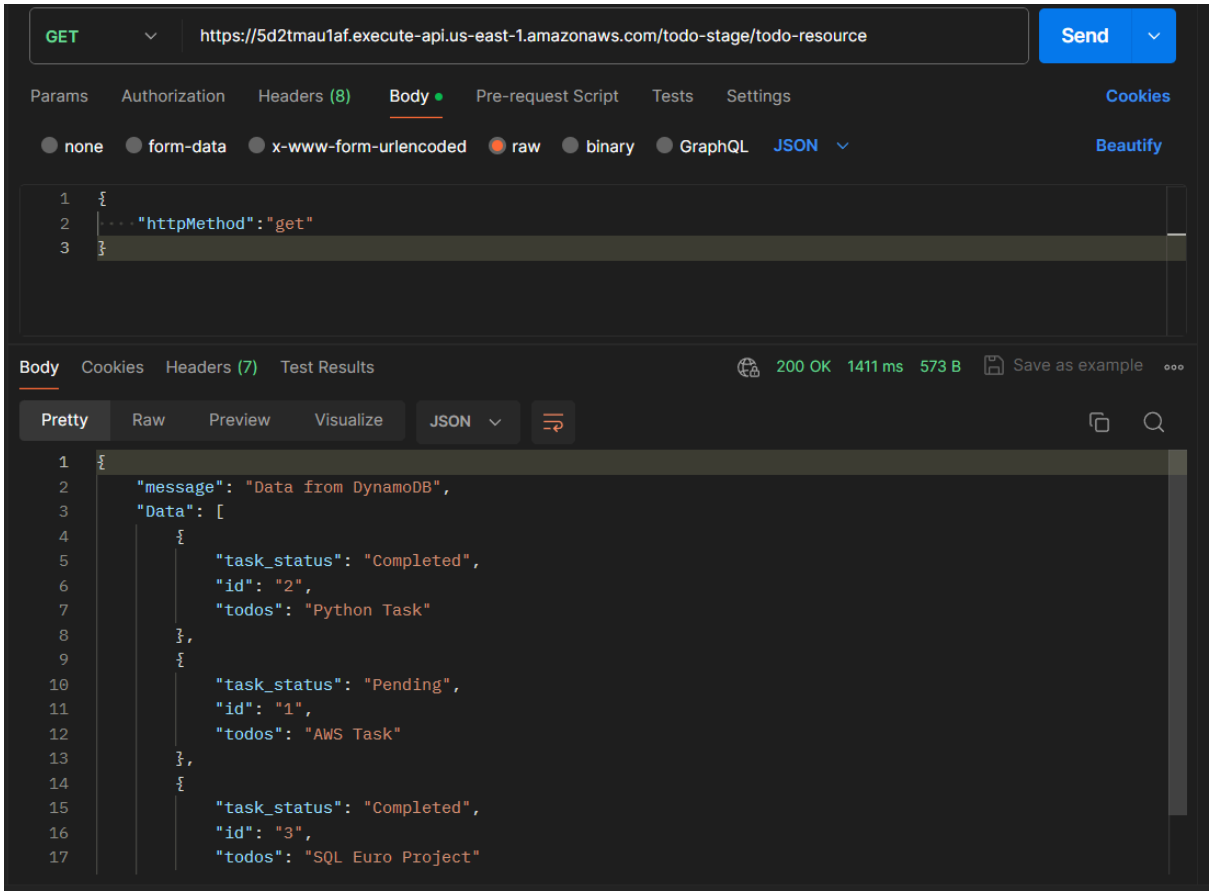
Creating a DynamoDB as



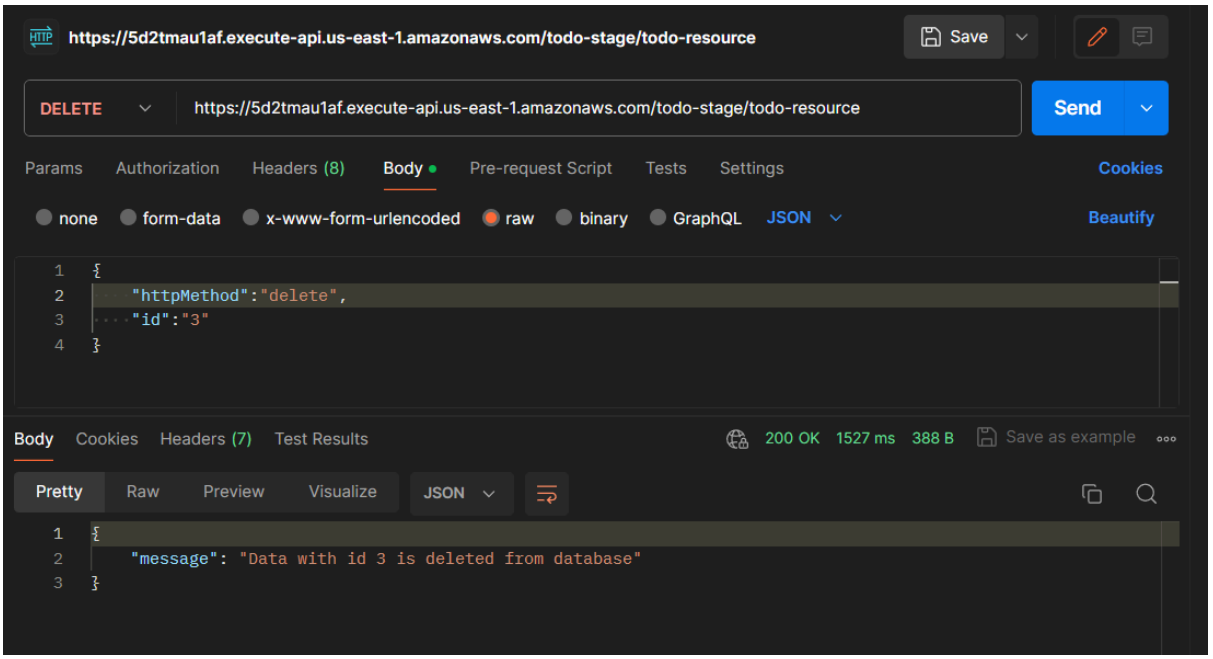
Message for posting the file in database




[Alt+S]
N. Virginia
voclabs/user3017594=sapkotanavin21@gmail.com @ 8294-7859-220
Any tag key
Any tag value
Find tables by table name
navin_todo
Scan or query items
Scan
Query
Select a table or index
Table - navin_todo
Select attribute projection
All attributes
Filters
Run
Reset
Completed. Read capacity units consumed: 0.5
Items returned (3)
Actions
Create item
id (String)
task_status
todos
2
Completed
Python Task
1
Pending
AWS Task
3
Completed
SQL Euro Project


GET
https://5d2tmau1af.execute-api.us-east-1.amazonaws.com/todo-stage/todo-resource
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"httpMethod":"get"
}
Body
Cookies
Headers (7)
Test Results
200 OK
1411 ms
573 B
Save as example
Pretty
Raw
Preview
Visualize
JSON
{
"message": "Data from DynamoDB",
"Data": [
{
"task_status": "Completed",
"id": "2",
"todos": "Python Task"
},
{
"task_status": "Pending",
"id": "1",
"todos": "AWS Task"
},
{
"task_status": "Completed",
"id": "3",
"todos": "SQL Euro Project"

Deleting the text and message for deleting the content from database



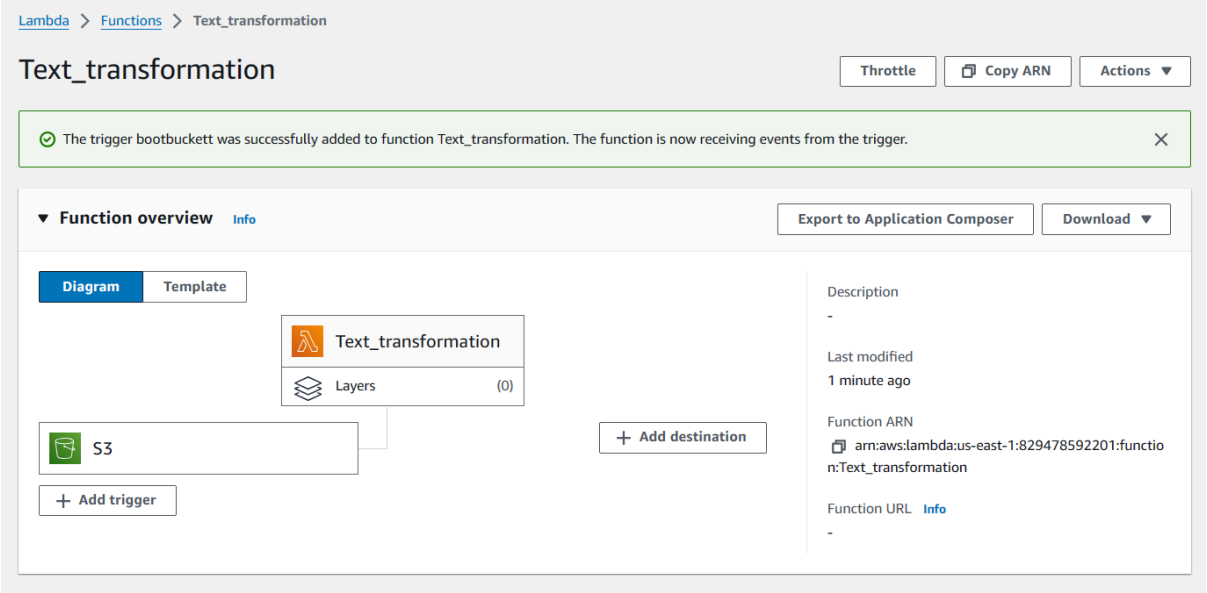
3. Serverless Data Processing Pipeline

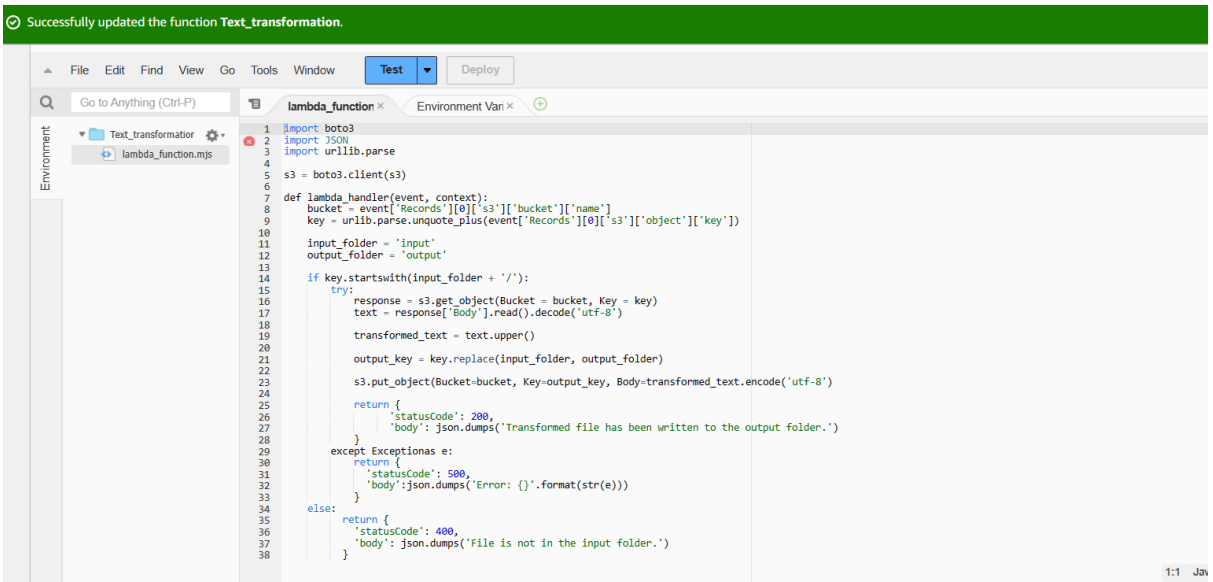
Objective: Build a serverless pipeline for processing data (e.g., log processing or ETL jobs).

Approach:
- **Data Ingestion:** Use AWS services like S3 or Kinesis to ingest data.
- **Processing:** Create Lambda functions to process the ingested data.
- **Storage:** Store the processed data in an appropriate AWS service, like S3 or DynamoDB.
- **Monitoring:** Set up CloudWatch to monitor the pipeline's performance and to log any issues.

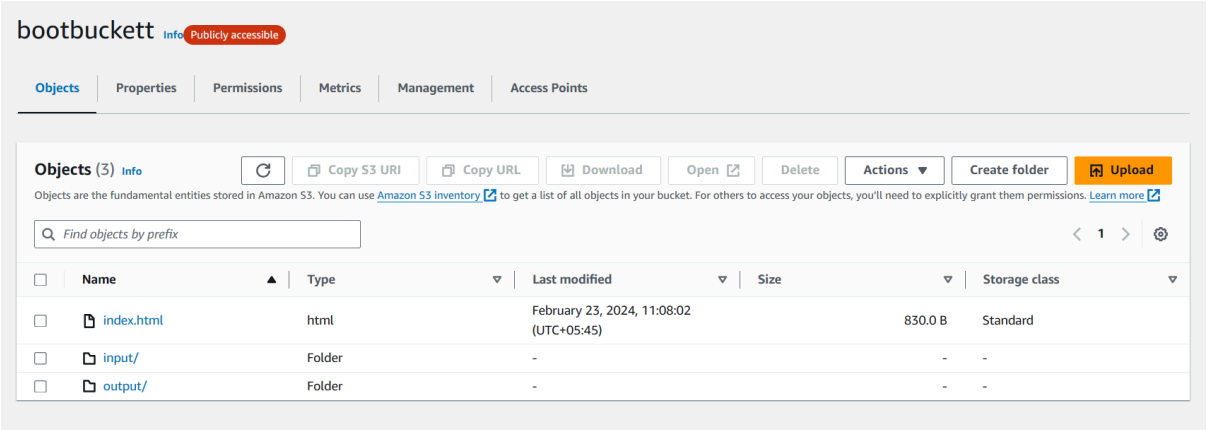
Goal: Learn to build a serverless data processing pipeline, understanding the flow of data through various AWS services.

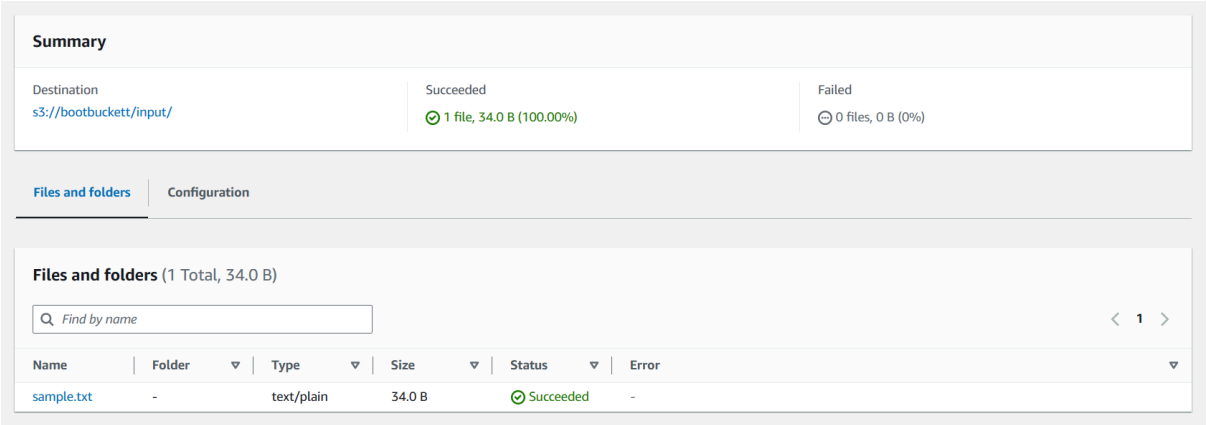
created a Lambda function that will trigger S3 bucket for text transformation which will transform the text into Uppercase and store in the S3 bucket inside output folder



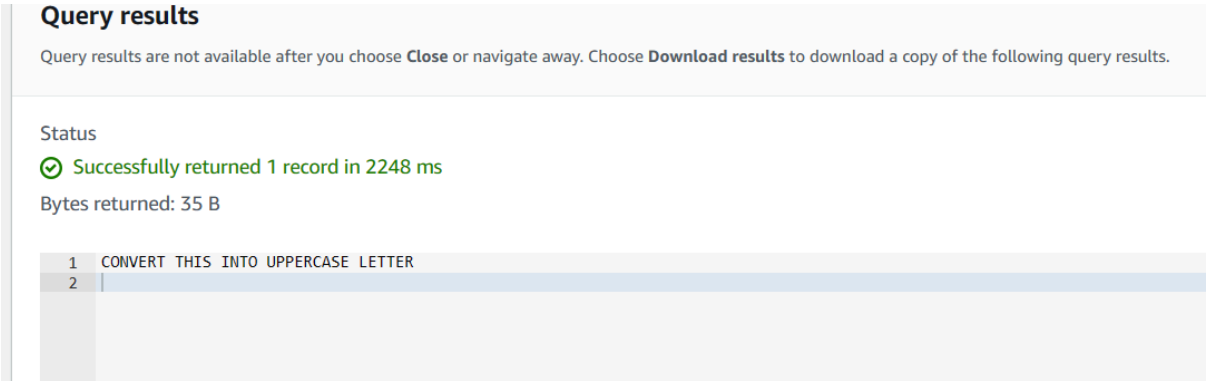


Here I have input and output folder where input folder hold sample.txt file





Here is the result for the text transformation



Here, how we can watch the cloud log