

Serverless Labs

1. Building a Serverless Web Application

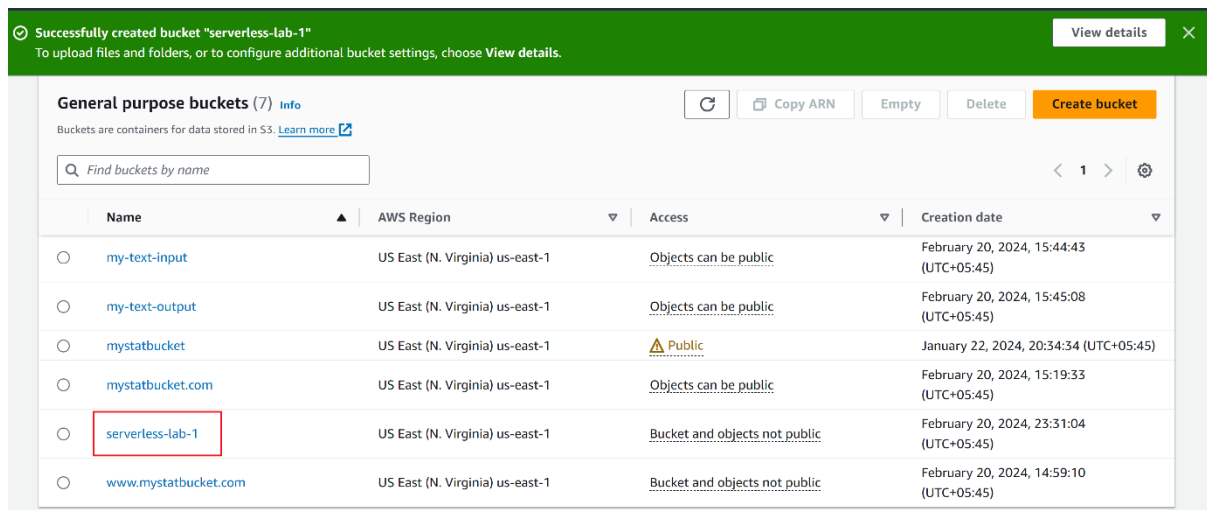
Objective: Create a serverless web application using AWS Lambda, API Gateway, S3, and DynamoDB.

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.

First, let's initiate the process by 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.'



From here, we will access our bucket > go to edit bucket policy, and open policy generator.

Edit bucket policy [Info](#)


Bucket policy

Policy examples [↗](#)

Policy generator [↗](#)

The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned by other accounts. [Learn more](#) [↗](#)

Bucket ARN

 arn:aws:s3:::serverless-lab-1

Policy



AWS Policy Generator

The AWS Policy Generator is a tool that enables you to create policies that control access to Amazon Web Services (AWS) products and resources. For more information about creating policies, see [key concepts in Using AWS Identity and Access Management](#). Here are sample policies.

Step 1: Select Policy Type

A Policy is a container for permissions. The different types of policies you can create are an IAM Policy, an S3 Bucket Policy, an SNS Topic Policy, a VPC Endpoint Policy, and an SQS Queue Policy.

Select Type of Policy S3 Bucket Policy ▾

Step 2: Add Statement(s)

A statement is the formal description of a single permission. See a [description of elements](#) that you can use in statements.

Effect ☒ Allow ☐ Deny

Principal

Use a comma to separate multiple values.

AWS Service

Amazon S3 ▾

☐ All Services (**)

Use multiple statements to add permissions for more than one service.

Actions

1 Action(s) Selected ☐ All Actions (**)

Amazon Resource Name (ARN)

ARN should follow the following format: arn:aws:s3:::(BucketName)/\$(KeyName).
Use a comma to separate multiple values.

[Add Conditions \(Optional\)](#)

[Add Statement](#)

Step 3: Generate Policy

A policy is a document (written in the [Access Policy Language](#)) that acts as a container for one or more statements.

Add one or more statements above to generate a policy.

This AWS Policy Generator is provided for informational purposes only, you are still responsible for your use of Amazon Web Services technologies and ensuring that your use is in compliance with all applicable terms and conditions. This AWS Policy Generator is provided as is without warranty of any kind, whether express, implied, or statutory. This AWS Policy Generator does not modify the applicable terms and conditions governing your use of Amazon Web Services technologies.

©2010, Amazon Web Services LLC or its affiliates. All rights reserved.

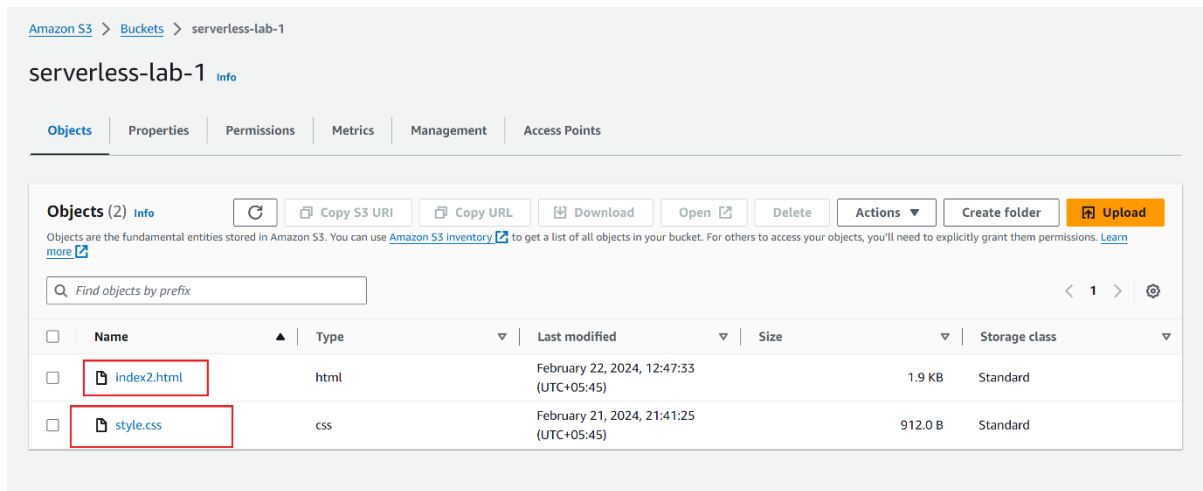
Policy JSON Document

Click below to edit. To save the policy, copy the text below to a text editor.
Changes made below will not be reflected in the policy generator tool.

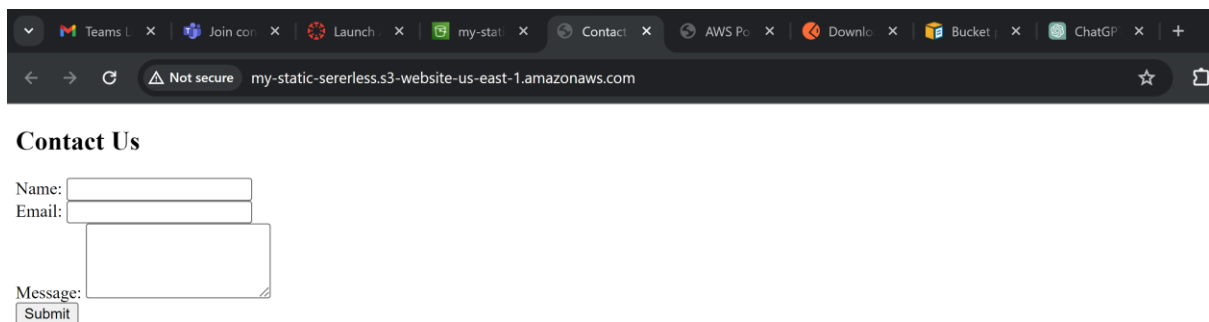
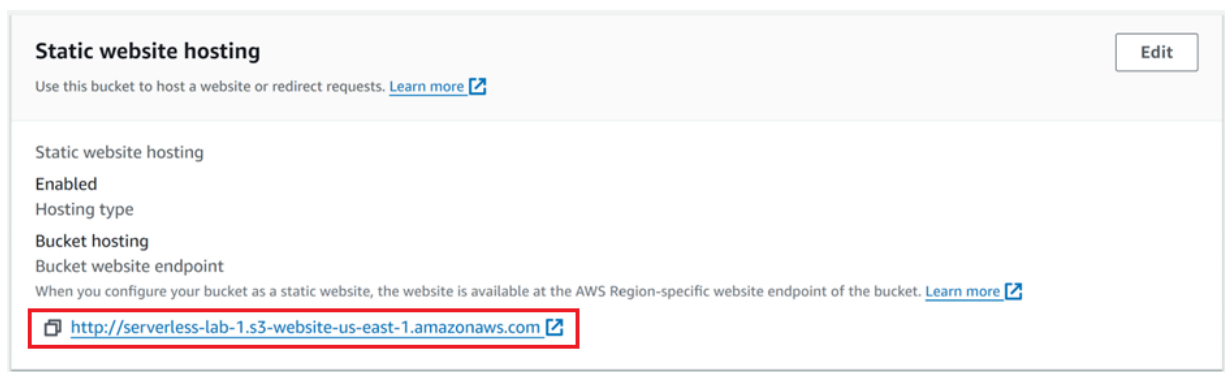
```
{
  "Id": "Policy1708581371646",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1708581368439",
      "Action": [
        "s3:GetObject"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::serverless-lab-1",
      "Principal": "*"
    }
  ]
}
```

This AWS Policy Generator is provided for informational purposes only, you are still responsible for your use of Amazon Web Services technologies and ensuring that your use is in compliance with all applicable terms and conditions. This AWS Policy Generator is provided as is without warranty of any kind.

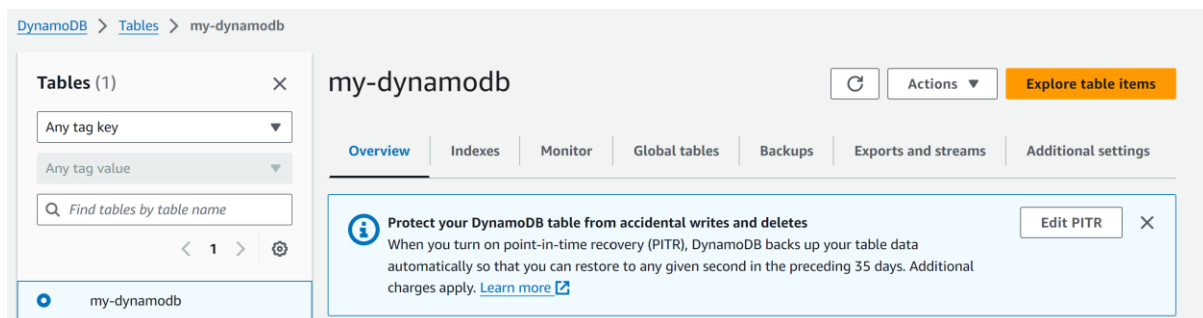
Close



We can access our static website from the following link:



Next, we will create a table in DynamoDB with necessary configurations.



Next, we will create lambda function with API Gateway and necessary configuration

Lambda > Functions > test_lambda

test_lambda

Throttle Copy ARN Actions

Function overview Info

Export to Application Composer Download

Diagram Template

test_lambda

Layers (0)

API Gateway (4)

+ Add trigger

+ Add destination

Description -

Last modified last month

Function ARN
arn:aws:lambda:us-east-1:794872146236:function:test_lambda

Function URL [Info](#)

Next, let's create a REST API with a resource. When configuring the resource, ensure to enable the proxy resource. By default, the resource is set to non-proxy integration, which means it only retrieves the request payload data in the event and not other details like headers.

✓ Successfully created REST API 'restapi-serverless (ysdesmfmxl)'

API Gateway > APIs > Resources - restapi-serverless (ysdesmfmxl) > Create resource

Create resource

Resource details

☒ Proxy resource [Info](#)
Proxy resources handle requests to all sub-resources. To create a proxy resource use a path parameter that ends with a plus sign, for example {proxy+}.

Resource path

Resource name

☒ CORS (Cross Origin Resource Sharing) [Info](#)
Create an OPTIONS method that allows all origins, all methods, and several common headers.

Cancel Create resource

We will now create a POST function and deploy the API

API Gateway > APIs > Resources - restapi-serverless (ysdesmfmxi)

Resources

API actions ▼ **Deploy API**

Create resource

- /
- /{proxy-resource+}
- ANY
- OPTIONS
- POST**

/{proxy-resource+} - POST - Method execution

Update documentation Delete

ARN: `arn:aws:execute-api:us-east-1:794872146236:ysdesmfmxi/*/POST/{proxy-resource+}` Resource ID: `xg5vvg`

```
graph LR; Client[Client] --> MR[Method request]; MR --> IR[Integration request]; IR --> LI[Lambda integration]; LI --> IRes[Integration response]; IRes --> MRs[Method response]; MRs --> Client
```

DynamoDB > Explore items > my-dynamodb

my-dynamodb

Autopreview View table details

Tables (1)

Any tag key

Any tag value

Find tables by table name

< 1 > ⚙

my-dynamodb

▼ Scan or query items

☒ Scan ☐ Query

Select a table or index: Table - my-dynamodb

Select attribute projection: All attributes

Filters

Run Reset

✓ Completed. Read capacity units consumed: 0.5

After this, we will go back to our static website and fill the forms.

Contact Us

Name:

Email:

Message:

The form POST when submitted, can be seen below.

Find tables by table name

1

my-dynamodb

my-new-table

my-table-aws

Scan

Query

Select a table or index

Table - my-new-table

Select attribute projection

All attributes

Filters

Run

Reset

Completed. Read capacity units consumed: 0.5

Items returned (2)

Actions

Create item

	name (String)	email	message
<input type="checkbox"/>	Maruti Suzuki	pujan.stha@gmail.com	sdedtwegfhijwekf

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

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

Lambda > Functions > todo-postman-lambda

todo-postman-lambda

Throttle Copy ARN Actions

▼ Function overview Info

Export to Application Composer Download

Diagram Template

todo-postman-lambda

Layers (0)

+ Add trigger

+ Add destination

Description

-

Last modified

5 seconds ago

Function ARN

arn:aws:lambda:us-east-1:794872146236:function:todo-postman-lambda

Create API Gateway

Lambda > Functions > todo-postman-lambda

todo-postman-lambda

Throttle Copy ARN Actions

☑ The trigger todo-postman-lambda-API was successfully added to function todo-postman-lambda. The function is now receiving events from the trigger.

▼ Function overview Info

Export to Application Composer Download

Diagram Template

todo-postman-lambda

Layers (0)

API Gateway

+ Add trigger

+ Add destination

Description

-

Last modified

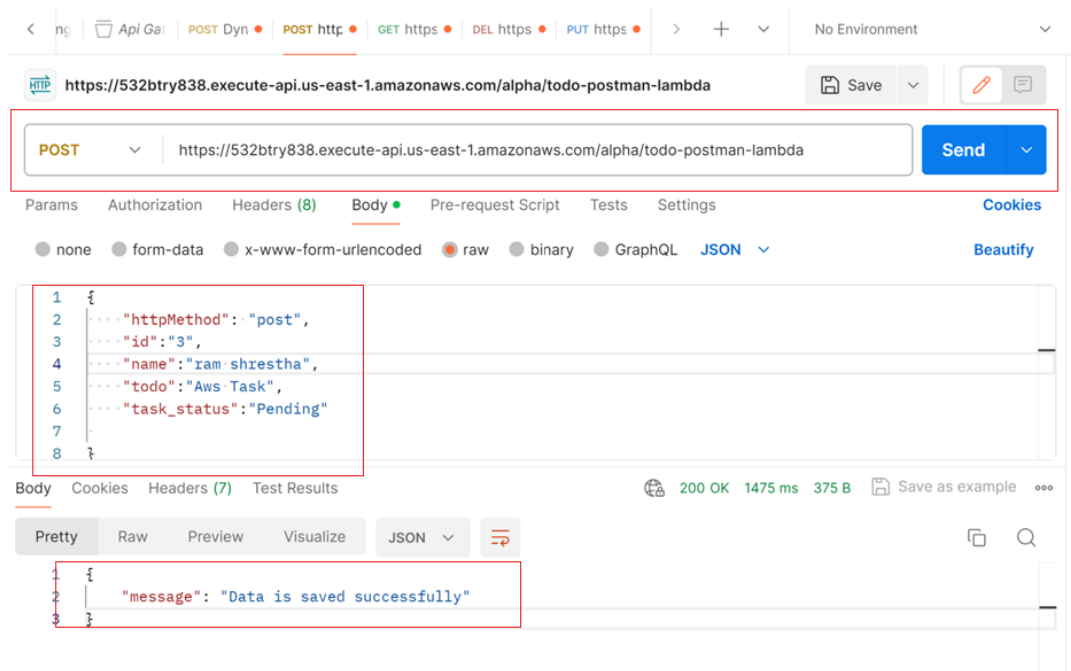
4 minutes ago

Function ARN

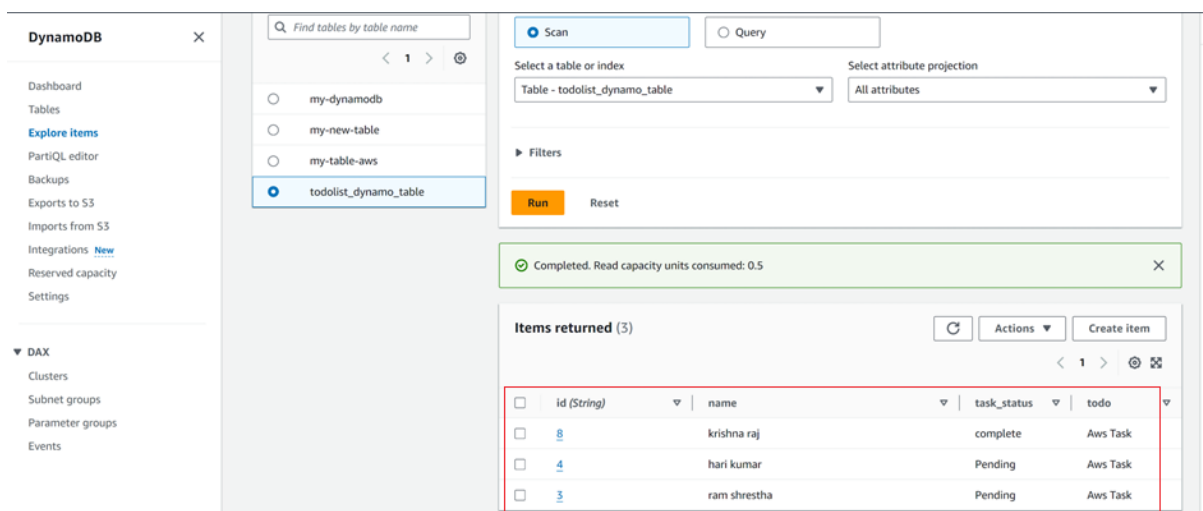
arn:aws:lambda:us-east-1:794872146236:function:todo-postman-lambda

Function URL [Info](#)

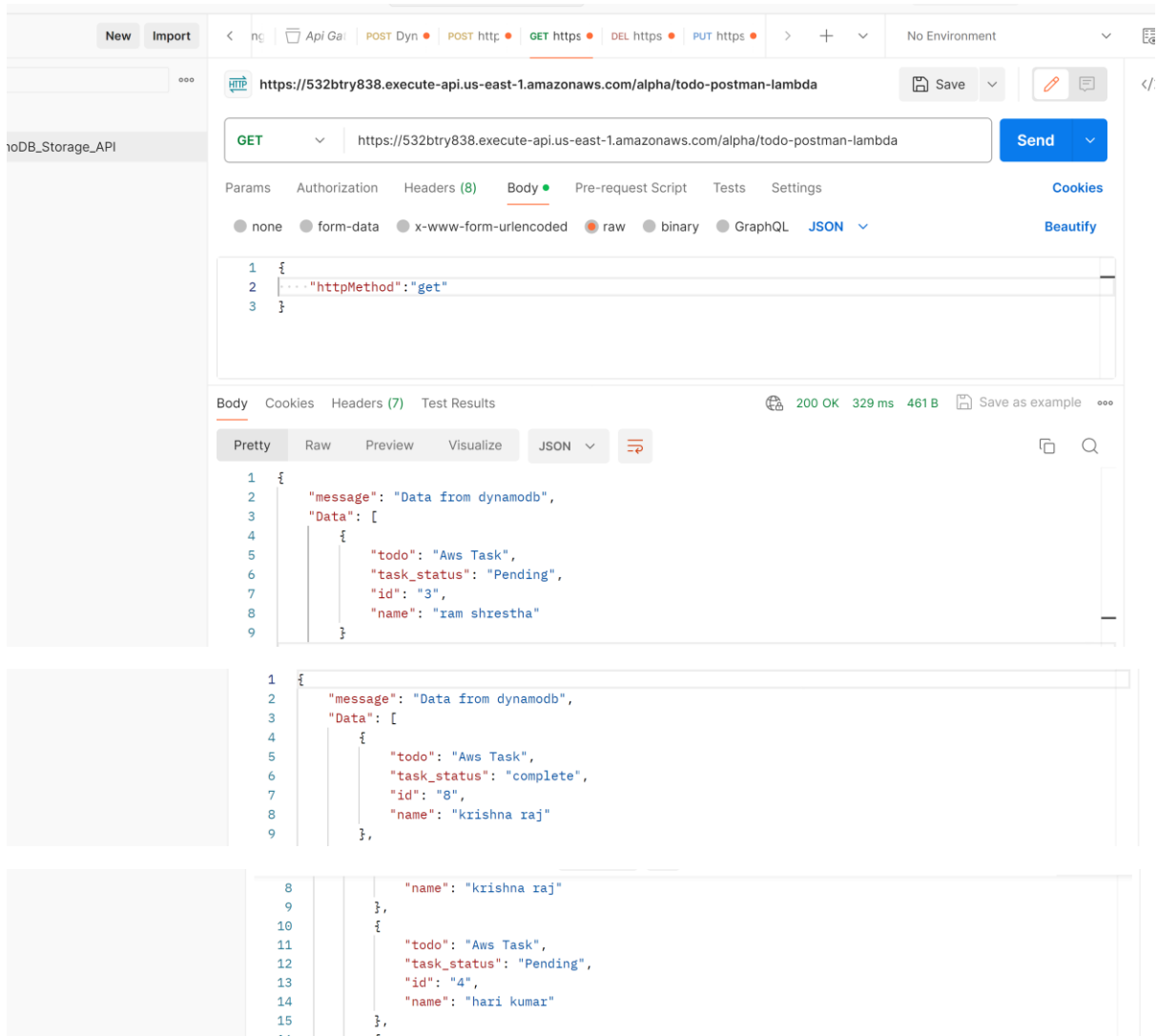
POST METHOD:



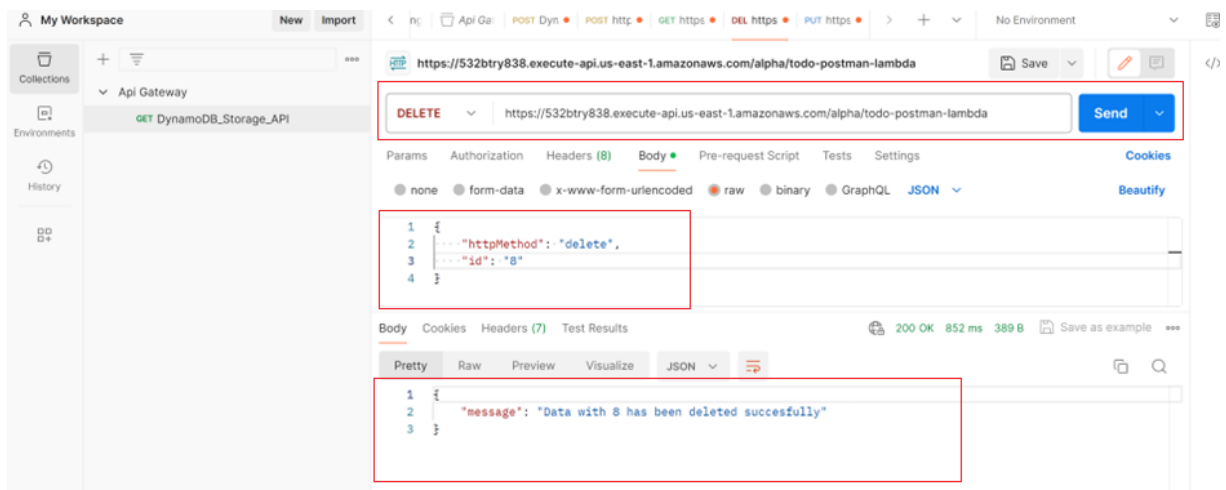
Now when we check in the DynamoDB:



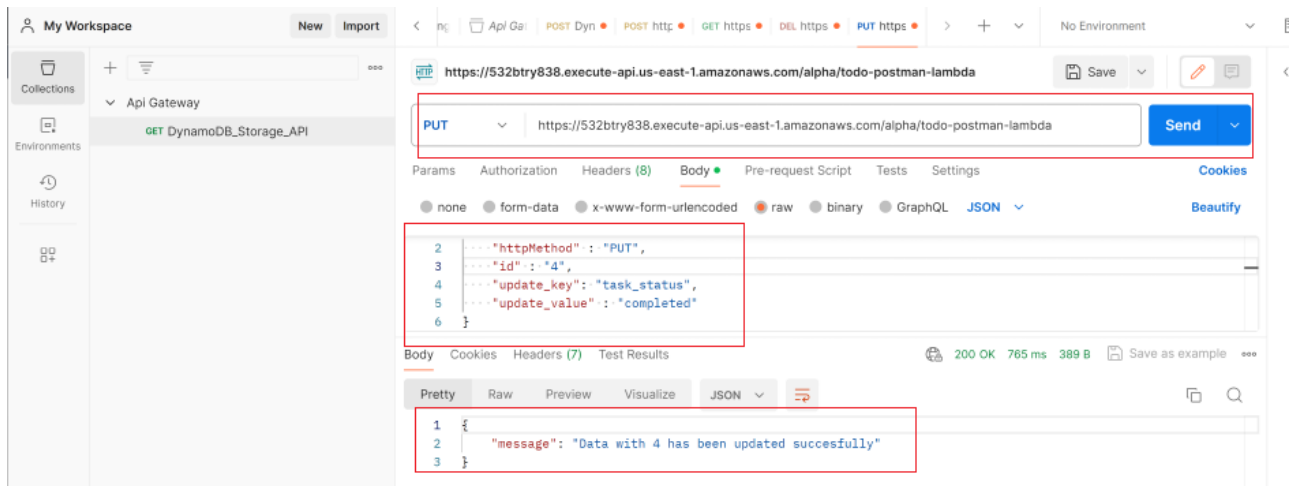
Now we will check GET METHOD:



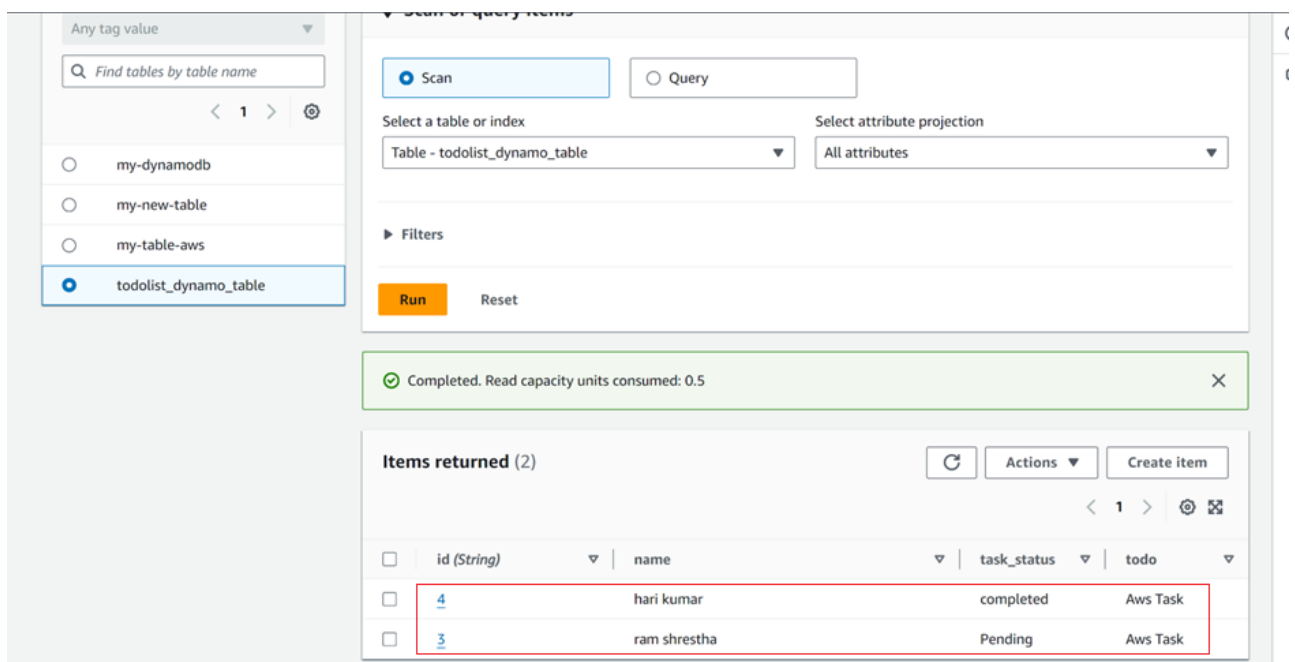
Now we will check DELETE METHOD:



Now we will use UPDATE(PUT) METHOD:



Now when we check our database:



3. Serverless Data Processing Pipeline

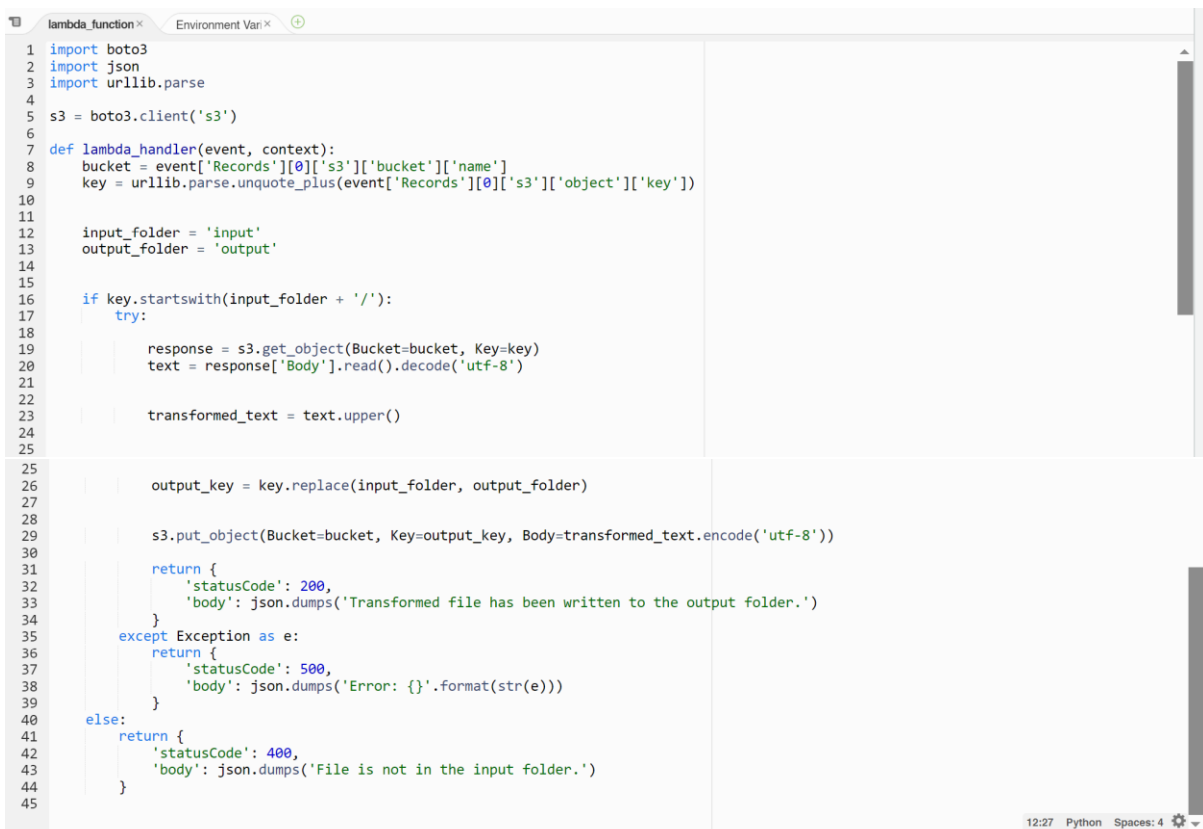
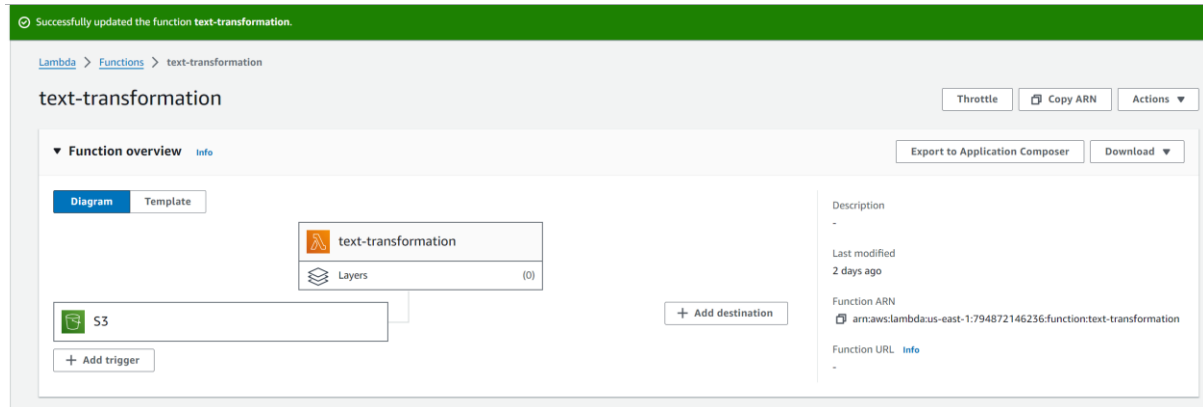
Objective: Build a serverless pipeline for processing data (e.g., log processing or ETL jobs).

Approach:

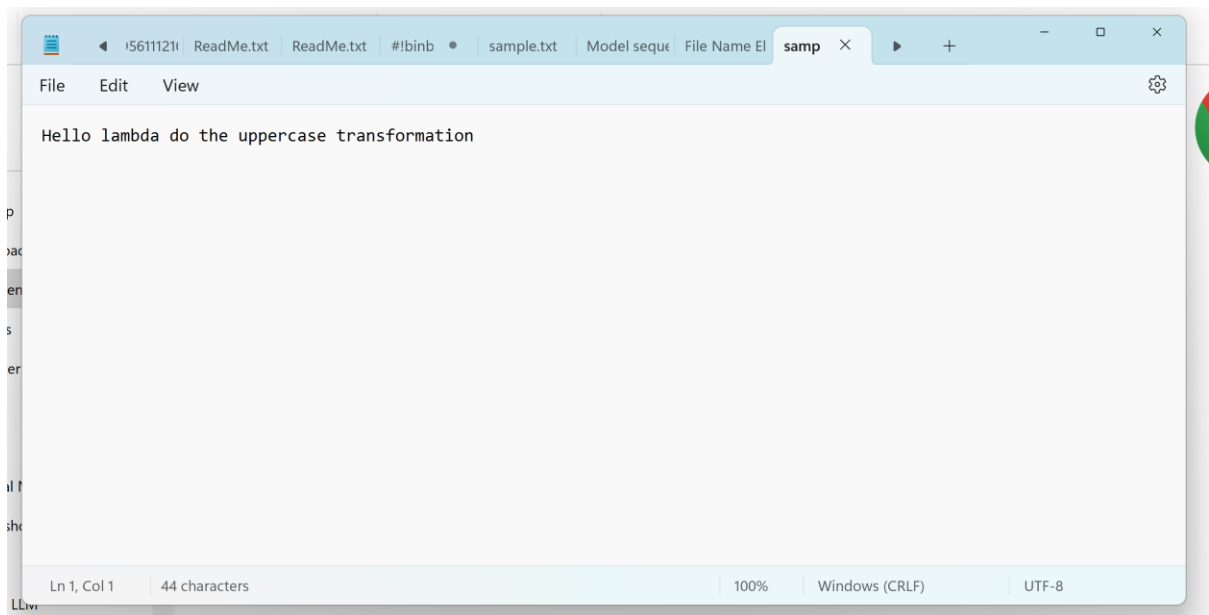
- **Data Ingestion:** Use AWS services like S3 or Kinesis to ingest data.
- **Processing:** Create Lambda functions to process the ingested data.
- **Storage:** Store the processed data in an appropriate AWS service, like S3 or DynamoDB.
- **Monitoring:** Set up CloudWatch to monitor the pipeline's performance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

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

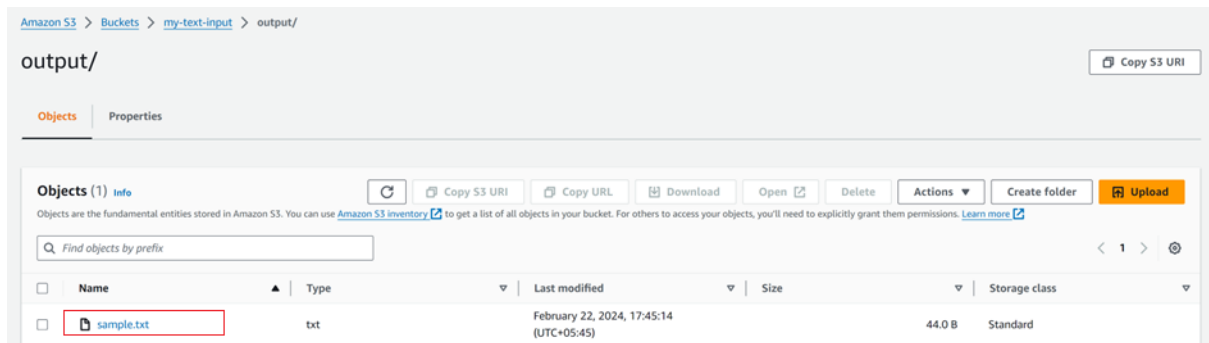
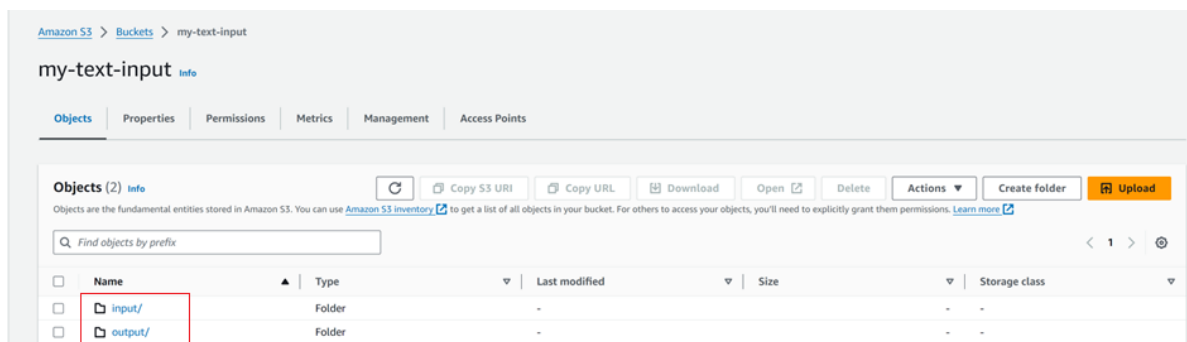
I have created a Lambda function that will trigger S3 bucket for text transformation which will transform the text into Uppercase and store in the S3 bucket inside output folder.



I have a text file in my input folder called sample.txt



Now, my lambda function will read the file from the input folder and right the transformed file in the output folder.



Amazon S3 > Buckets > my-text-input > output/ > sample.txt

sample.txt [Info](#)

[Copy S3 URI](#) [Download](#) [Open](#) [Object actions](#)

[Properties](#) [Permissions](#) [Versions](#)

Object overview

Owner awslabsOw6975139t1703163349	S3 URI s3://my-text-input/output/sample.txt
AWS Region US East (N. Virginia) us-east-1	Amazon Resource Name (ARN) arn:aws:s3::my-text-input/output/sample.txt
Last modified February 22, 2024, 17:45:14 (UTC+05:45)	Entity tag (Etag) a377a47561821ca1225d62c5639d8b1c
Size 44.0 B	Object URL

- Download as
- Share with a presigned URL
- Calculate total size
- Copy
- Move
- Initiate restore
- Query with S3 Select
- Edit actions
 - Rename object
 - Edit storage class
 - Edit server-side encryption
 - Edit metadata
 - Edit tags
 - Make public using ACL

SQL query

Amazon S3 Select supports only the SELECT SQL command. Using the S3 console, you can extract up to 40 MB of records from an object that is up to 128 MB in size. To work with larger files or more records, use the AWS CLI, AWS SDK, or Amazon S3 REST API. For more complex SQL queries, use [Amazon Athena](#).

[Add SQL from templates](#) [Run SQL query](#)

```
1 /* To create reference point for writing SQL queries, you can display the first 5 records of input data by running the following SQL query: SELECT * FROM s3object s LIMIT 5 */
2 SELECT * FROM s3object s LIMIT 5
```

SQL Ln 1, Col 1 Errors: 0 Warnings: 0

Query results

Query results are not available after you choose **Close** or navigate away. Choose **Download results** to download a copy of the following query results.

[Download results](#)

Status
✔ Successfully returned 1 record in 1236 ms
Bytes returned: 45 B

1	HELLO LAMBDA DO THE UPPERCASE TRANSFORMATION
2	

As you can see, our text has been transformed into uppercase.

We can check CloudWatch for the logs.

CloudWatch

Log groups (7)

By default, we only load up to 10000 log groups.

[Filter log groups or try prefix search](#) ☐ Exact match

<input type="checkbox"/>	Log group	Log class	Anomaly d...	Data prote...	Sensitive ...	Retention	Metric filters
<input type="checkbox"/>	/aws/lambda/RedshiftEventSubscription	Standard	Configure	-	-	Never expire	-
<input type="checkbox"/>	/aws/lambda/RedshiftOverwatch	Standard	Configure	-	-	Never expire	-
<input type="checkbox"/>	/aws/lambda/RoleCreationFunction	Standard	Configure	-	-	Never expire	-
<input type="checkbox"/>	/aws/lambda/my-lambda-2	Standard	Configure	-	-	Never expire	-
<input type="checkbox"/>	/aws/lambda/test_lambda	Standard	Configure	-	-	Never expire	-
<input type="checkbox"/>	/aws/lambda/text-transformation	Standard	Configure	-	-	Never expire	-
<input type="checkbox"/>	/aws/lambda/todo-postman-lambda	Standard	Configure	-	-	Never expire	-

CloudWatch

Favorites and recents

Dashboards

Alarms

24

In alarm

All alarms

Billing

Logs

Log groups

Log Anomalies

Live Tail

Logs Insights

Metrics

X-Ray traces

Events

Application Signals

Network monitoring

Insights

Container Insights

4 days ago

Retention

Never expire

Stored bytes

13.18 KB

KMS key ID

-

Log streams

Tags

Anomaly detection

Metric filters

Subscription filters

Contributor Insights

Data protection

Log streams (7)

Filter log streams or try prefix search

Exact match

Show expired

Info

1

Log stream	Last event time
2024/02/22/[\$LATEST]1396aa33ff22401687c29fd450ed1289	2024-02-22 17:45:12 (UTC+05:45)
2024/02/20/[\$LATEST]98493518634c4d998c888545124eba64	2024-02-20 16:39:45 (UTC+05:45)
2024/02/20/[\$LATEST]30b7d816cc36480f814c52933ab1725b	2024-02-20 16:35:33 (UTC+05:45)
2024/02/20/[\$LATEST]f2cfbc76e354d2f8595570e10d0d976	2024-02-20 16:32:11 (UTC+05:45)
2024/02/20/[\$LATEST]c911feb0c2a945559414d7c0e873787b	2024-02-20 16:28:02 (UTC+05:45)
2024/02/20/[\$LATEST]f8b7b2602a17414cba99a28e9442bddb	2024-02-20 16:08:21 (UTC+05:45)
2024/02/20/[\$LATEST]bccd8178a054742b19f24bf3478f58571	2024-02-20 16:00:45 (UTC+05:45)

CloudWatch

Favorites and recents

Dashboards

Alarms

24

In alarm

All alarms

Billing

Logs

Log groups

Log Anomalies

Live Tail

Logs Insights

Metrics

CloudWatch > Log groups > /aws/lambda/text-transformation > 2024/02/22/[\$LATEST]1396aa33ff22401687c29fd450ed1289

Log events

Filter events

Clear

1m

30m

1h

12h

Custom

Local timezone

Display

Timestamp

Message

No older events at this moment. [Retry](#)

2024-02-22T17:45:12.173+05:45

INIT_START Runtime Version: python3.12.v19 Runtime Version ARN: arn:aws:lambda:us-east-1:runtime:a79ae1de439e89e7a1dc89465a221e8fe9bb3c495c3ff8e61b...

2024-02-22T17:45:12.785+05:45

START RequestId: 5f11f5fc-18fc-4369-8b8a-47857df6d9d2 Version: \$LATEST

2024-02-22T17:45:13.467+05:45

END RequestId: 5f11f5fc-18fc-4369-8b8a-47857df6d9d2

2024-02-22T17:45:13.467+05:45

REPORT RequestId: 5f11f5fc-18fc-4369-8b8a-47857df6d9d2 Duration: 681.48 ms Billed Duration: 682 ms Memory Size: 128 MB Max Memory Used: 81 MB Init Du...

No newer events at this moment. [Auto retry paused. Resume](#)