

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.'

Amazon S3 > Buckets

▼ Account snapshot

Last updated: Feb 26, 2024 by Storage Lens. Metrics are generated every 24 hours. Metrics don't include directory buckets. [Learn more](#)

Total storage

2.5 KB

Object count

4

Average object size

647.8 B

You can enable advanced metrics in the ["default-account-dashboard"](#) configuration.

View Storage Lens dashboard

General purpose buckets

Directory buckets

General purpose buckets (3) Info

↻

Copy ARN

Empty

Delete

Create bucket

Buckets are containers for data stored in S3.


Find buckets by name

< 1 >

⚙

	Name	AWS Region	Access	Creation date
<input type="radio"/>	abacusbootcamp	US East (N. Virginia) us-east-1	Bucket and objects not public	February 22, 2024, 18:00:47 (UTC+05:45)
<input type="radio"/>	bootbuckett	US East (N. Virginia) us-east-1	Public	February 23, 2024, 11:06:04 (UTC+05:45)
<input type="radio"/>	lab1severless	US East (N. Virginia) us-east-1	Bucket and objects not public	February 27, 2024, 22:44:14 (UTC+05:45)

Generating web services 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

-- Select Actions --

☒ All Actions ('*')

Amazon Resource Name (ARN)

arn:aws:s3:::lab1severless

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

Add Conditions (Optional)

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.

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": "Policy1709053725346",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1709053452626",
      "Action": "s3:*",
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::lab1severless",
      "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, 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.

Close

Step 3: Generate Policy

Uploading file in bucket

Amazon S3 > Buckets > lab1severless

lab1severless [Info](#)

Objects

Properties

Permissions

Metrics

Management

Access Points

Objects (2) [Info](#)

🔄



📄 Copy S3 URI

📄 Copy URL

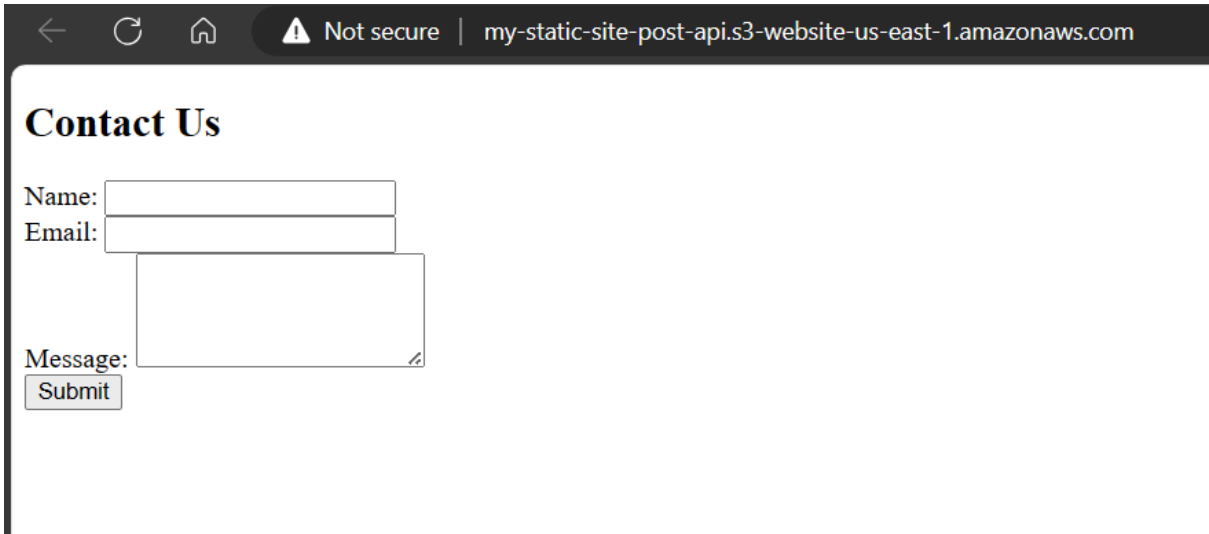
📄 Download

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in this bucket.

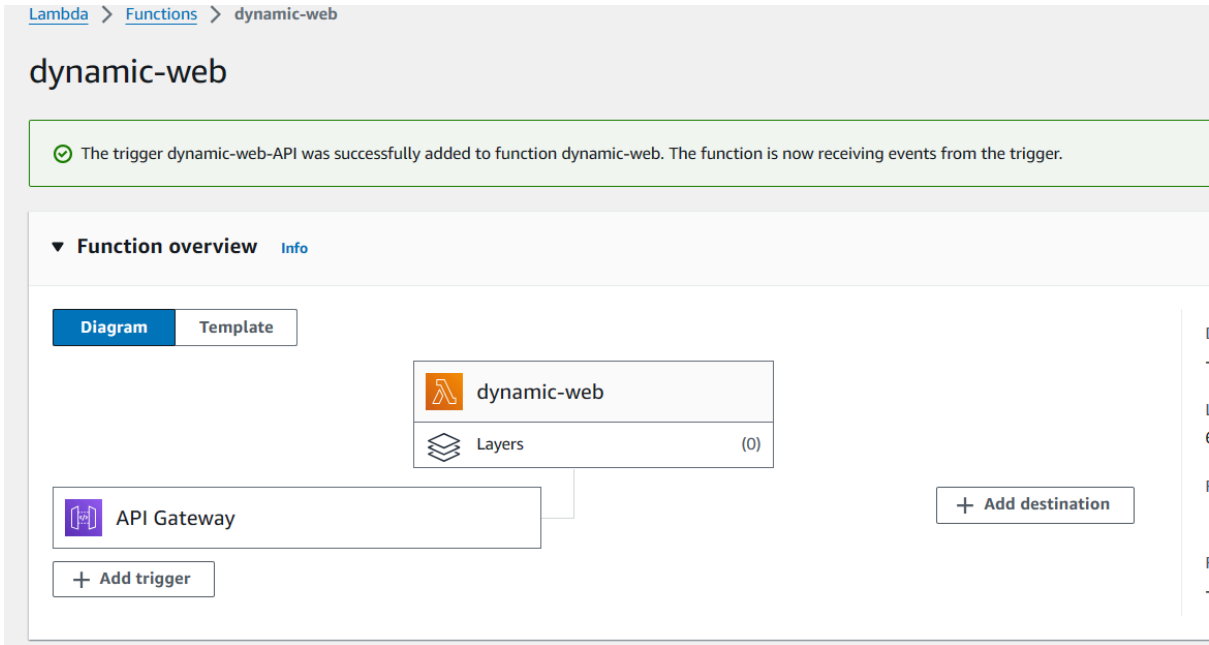
🔍 Find objects by prefix

<input type="checkbox"/>	Name	Type	Last modified
<input type="checkbox"/>	 index2.html	html	February 2, 2020 (UTC+05:45)
<input type="checkbox"/>	 style.css	css	February 2, 2020 (UTC+05:45)

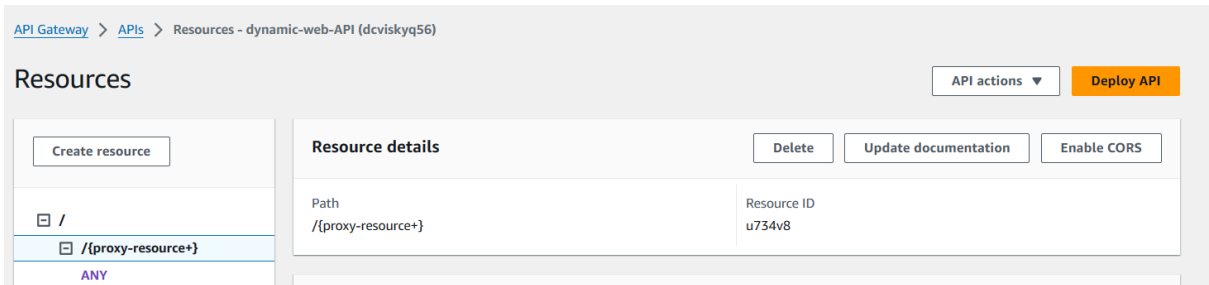
The static website is as



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

API Gateway

APIs

Resources - dynamic-web-API (dcviskyq56)

Resources

Create resource

/

/[proxy-resource+]

ANY

OPTIONS

POST

/dynamic-web

ANY

API actions

Deploy API

Update documentation

Delete

ARN

arn:aws:execute-api:us-east-1:829478592201:dcviskyq56/*/[POST]/[proxy-resource+]

Resource ID

u734v8

Client

Method request

Integration request

Method response

Integration response

Proxy integration

Lambda integration

DynamoDB

Explore items

nstable

Tables (2)

Any tag key

Any tag value

Find tables by table name

navin_todo

nstable

nstable

Autopreview

View table details

Scan or query items

Scan

Query

Select a table or index

Table - nstable

Select attribute projection

All attributes

Filters

Run

Reset

Completed. Read capacity units consumed: 0.5

Items returned (0)

Actions

Create item

1

No items

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

Contact Us

Name:

Email:

Message:

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

Items returned (2)

< 1 >

<input type="checkbox"/>	name (String)	email	message
<input type="checkbox"/>	Nabin Sapkota	sapkotanavin21@gmail.com	hello, I'm just testing it..

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

▼ 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 Lo

LabRole

▼

[View the LabRole role](#) on the IAM console.

► Advanced settings

The lambda is created as


lambda_navin


▼ Function overview [Info](#)

Export to

Diagram

Template

 lambda_navin

 Layers (0)

+ Add trigger

+ Add destination


Descr

-

Last r

4 sec

Funct

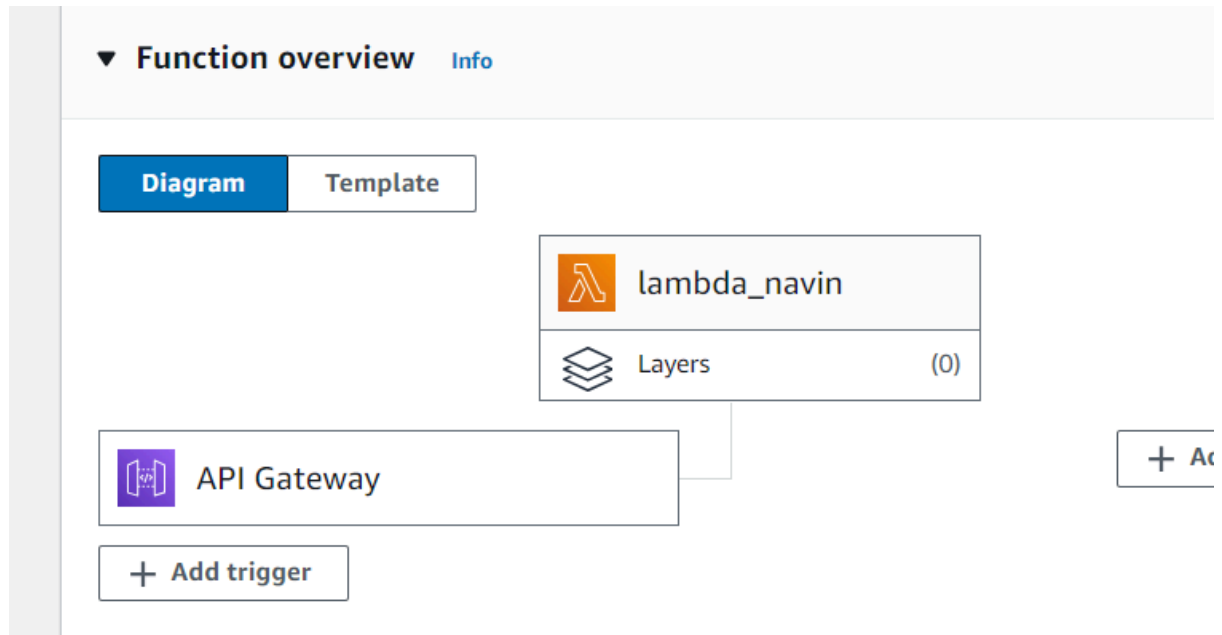
 a

n:lam

Funct

-

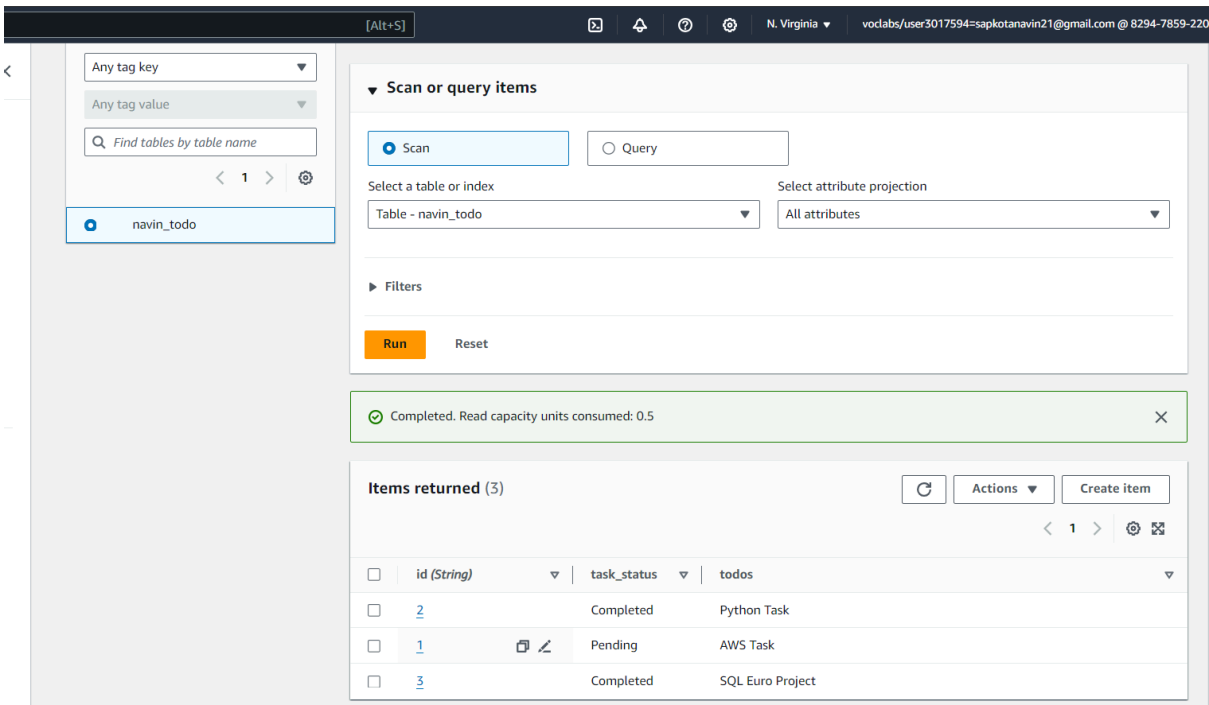
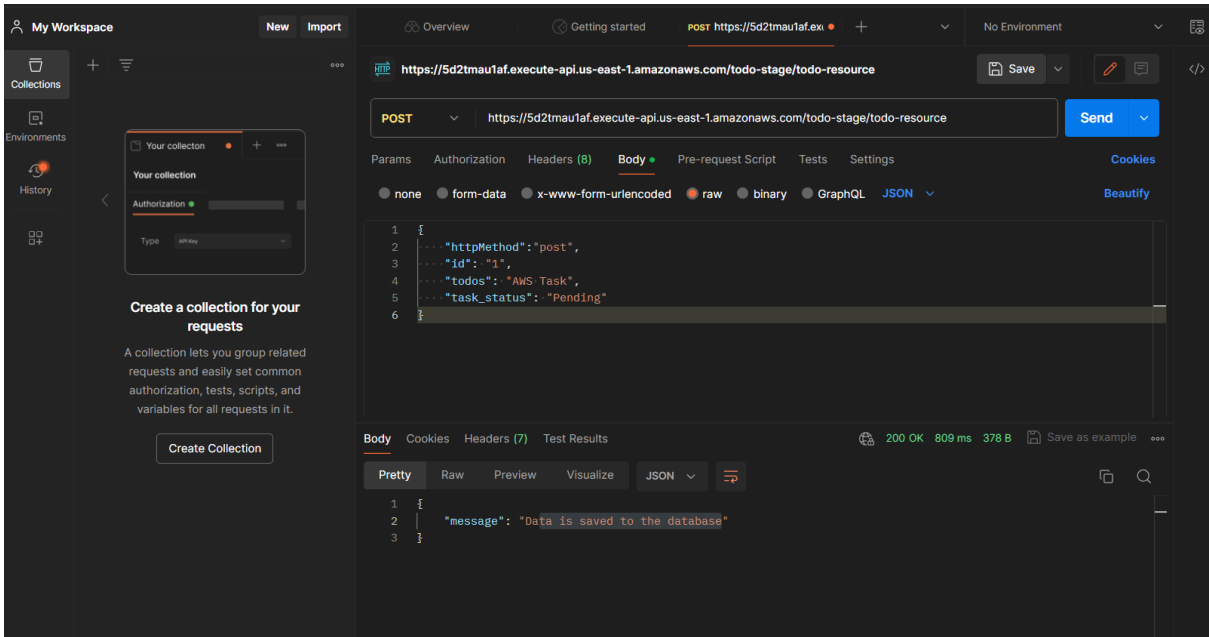
Adding a API gateway as



Response from API Gateway

1: {"message": "Hello from Lambda!"}

Creating a Post, Get, PPost and Put Functions as



Get method and message for get request

The screenshot displays a REST client interface with a GET request to the URL `https://5d2tmau1af.execute-api.us-east-1.amazonaws.com/todo-stage/todo-resource`. The request body is a JSON object: `{ "httpMethod": "get" }`. The response status is `200 OK` with a response time of `1411 ms` and a body size of `573 B`. The response body is a JSON object: `{ "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" }] }`.

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

```
1 {
2   "httpMethod": "get"
3 }
```

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

Pretty Raw Preview Visualize JSON

```
1 {
2   "message": "Data from DynamoDB",
3   "Data": [
4     {
5       "task_status": "Completed",
6       "id": "2",
7       "todos": "Python Task"
8     },
9     {
10      "task_status": "Pending",
11      "id": "1",
12      "todos": "AWS Task"
13    },
14    {
15      "task_status": "Completed",
16      "id": "3",
17      "todos": "SQL Euro Project"
18    }
19  ]
20 }
```

Creation of Post method as

[API Gateway](#) > [APIs](#) > [Resources - lambda_navin-API \(wikptfc4i2\)](#) > Create method

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

Creating a DynamoDB as

[DynamoDB](#) > [Tables](#) > Create table

Create table

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.

String

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.

String

1 to 255 characters and case sensitive.

Table settings

Message for posting the file in database

My Workspace

+

+

+

+

+

+

+

+

+

Overview

Getting started

POST https://5d2tmau1af.ex

No Environment

Save

Send

POST

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

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

1

2

3

4

5

6

1

2

3

200 OK

809 ms

378 B

Save as example

Create a collection for your requests

A collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it.

Create Collection

1

2

3

1

2

3

Any tag key

Any tag value

Find tables by table name

< 1 > ⚙

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)

↺ 1 ↻ ⚙

<input type="checkbox"/>	id (String)	task_status	todos
<input type="checkbox"/>	2	Completed	Python Task
<input type="checkbox"/>	1	Pending	AWS Task
<input type="checkbox"/>	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

1 {

2 "httpMethod": "get"

3 }

Body

Cookies

Headers (7)

Test Results

200 OK

1411 ms

573 B

Save as example

⋮

Pretty

Raw

Preview

Visualize

JSON

⌵

🔍

1 {

2 "message": "Data from DynamoDB",

3 "Data": [

4 {

5 "task_status": "Completed",

6 "id": "2",

7 "todos": "Python Task"

8 },

9 {

10 "task_status": "Pending",

11 "id": "1",

12 "todos": "AWS Task"

13 },

14 {

15 "task_status": "Completed",

16 "id": "3",

17 "todos": "SQL Euro Project"

Deleting the text and message for deleting the content from database

The screenshot displays a REST client interface with a dark theme. At the top, the URL bar shows `https://5d2tmau1af.execute-api.us-east-1.amazonaws.com/todo-stage/todo-resource`. Below the URL bar, the request method is set to **DELETE**, and the same URL is entered in the request path field. A **Send** button is located to the right of the path field. Below the request configuration, tabs for **Params**, **Authorization**, **Headers (8)**, **Body** (selected), **Pre-request Script**, **Tests**, and **Settings** are visible. The **Body** tab shows a JSON request body:

```
{  "httpMethod": "delete",  "id": "3"}
```

. Below the request body, tabs for **Body** (selected), **Cookies**, **Headers (7)**, and **Test Results** are shown. The **Body** tab displays the response in **Pretty** JSON format:

```
{  "message": "Data with id 3 is deleted from database"}
```

. Above the response body, the status is **200 OK**, the response time is **1527 ms**, and the response size is **388 B**. A **Save as example** button is also present.

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

[Lambda](#) > [Functions](#) > Text_transformation

Text_transformation

ThrottleCopy ARN


Actions ▾


✔ The trigger bootbuckett was successfully added to function Text_transformation. The function is now receiving events from the trigger. ✕


▼ Function overview [Info](#)

Diagram

Template

 Text_transformation

 Layers (0)

 S3

+ Add trigger

+ Add destination


Description

-

Last modified

1 minute ago

Function ARN

 arn:aws:lambda:us-east-1:829478592201:function:Text_transformation

Function URL [Info](#)

-

```
1 import boto3
2 import json
3 import urllib.parse
4
5 s3 = boto3.client(s3)
6
7 def lambda_handler(event, context):
8     bucket = event['Records'][0]['s3']['bucket']['name']
9     key = urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'])
10
11     input_folder = 'input'
12     output_folder = 'output'
13
14     if key.startswith(input_folder + '/'):
15         try:
16             response = s3.get_object(Bucket = bucket, Key = key)
17             text = response['Body'].read().decode('utf-8')
18
19             transformed_text = text.upper()
20
21             output_key = key.replace(input_folder, output_folder)
22             s3.put_object(Bucket=bucket, Key=output_key, Body=transformed_text.encode('utf-8'))
23
24             return {
25                 'statusCode': 200,
26                 'body': json.dumps('Transformed file has been written to the output folder.')
27             }
28         except Exception as e:
29             return {
30                 'statusCode': 500,
31                 'body': json.dumps('Error: {}'.format(str(e)))
32             }
33     else:
34         return {
35             'statusCode': 400,
36             'body': json.dumps('File is not in the input folder.')
37         }
38
```

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

bootbuckett

Info Publicly accessible

Objects

Properties

Permissions

Metrics

Management

Access Points

Objects (3) Info

Refresh

Copy S3 URI

Copy URL

Download

Open

Delete

Actions

Create folder

Upload

Find objects by prefix

1

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	index.html	html	February 23, 2024, 11:08:02 (UTC+05:45)	830.0 B	Standard
<input type="checkbox"/>	input/	Folder	-	-	-
<input type="checkbox"/>	output/	Folder	-	-	-

Summary

Destination

s3://bootbucket/input/

Succeeded

✔ 1 file, 34.0 B (100.00%)

Failed

⊖ 0 files, 0 B (0%)

Files and folders

Configuration

Files and folders (1 Total, 34.0 B)

Find by name

< 1 >

Name	Folder	Type	Size	Status	Error
sample.txt	-	text/plain	34.0 B	✔ Succeeded	-

Here is the result for the text transformation

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.

Status

✔ Successfully returned 1 record in 2248 ms

Bytes returned: 35 B

1	CONVERT THIS INTO UPPERCASE LETTER
2	

Here, how we can watch the cloud log

aws

Services

Q lambda

CloudWatch

Log groups

/aws/lambda/text-transform-lambda

2024/02/28/[LATEST]d7850db11ca84a31b03e84024ce0df57

Log events

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

Timestamp	Message
2024-02-28T18:59:03.875+05:45	INIT_START Runtime Version: python:3.12.v19 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:a79ae1de439e89e7a1dc80465a221a8fe9bb3c405c3ff8...
2024-02-28T18:59:04.324+05:45	START RequestId: 60390375-b377-4fed-9503-68b3e9922df2 Version: \$LATEST
2024-02-28T18:59:04.965+05:45	END RequestId: 60390375-b377-4fed-9503-68b3e9922df2
2024-02-28T18:59:04.965+05:45	REPORT RequestId: 60390375-b377-4fed-9503-68b3e9922df2 Duration: 641.61 ms Billed Duration: 642 ms Memory Size: 128 MB Max Memory Used: 81 MB In...

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