

Serverless Labs

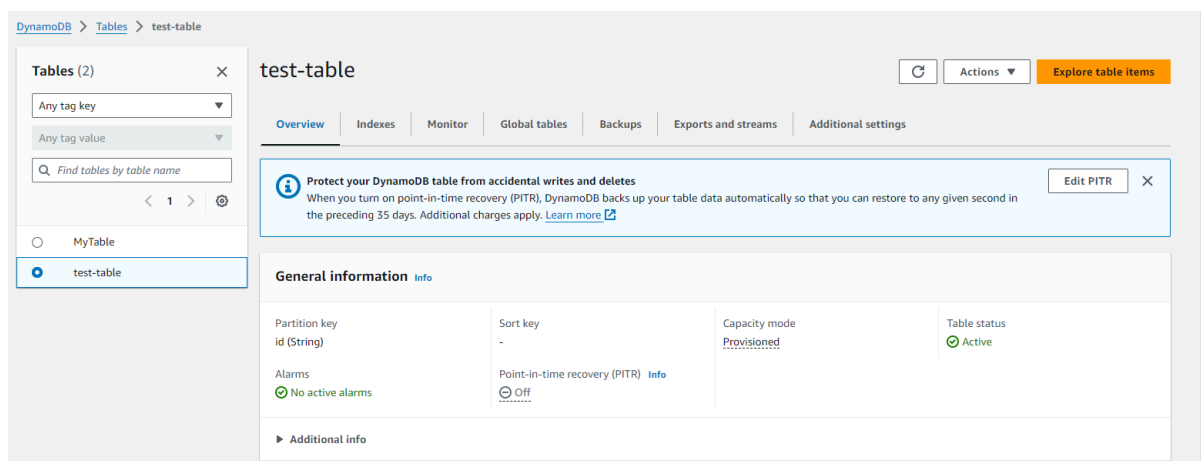
Creating a Serverless API

Define API:

I will be designing a simple RESTful API for a note application that allows us to

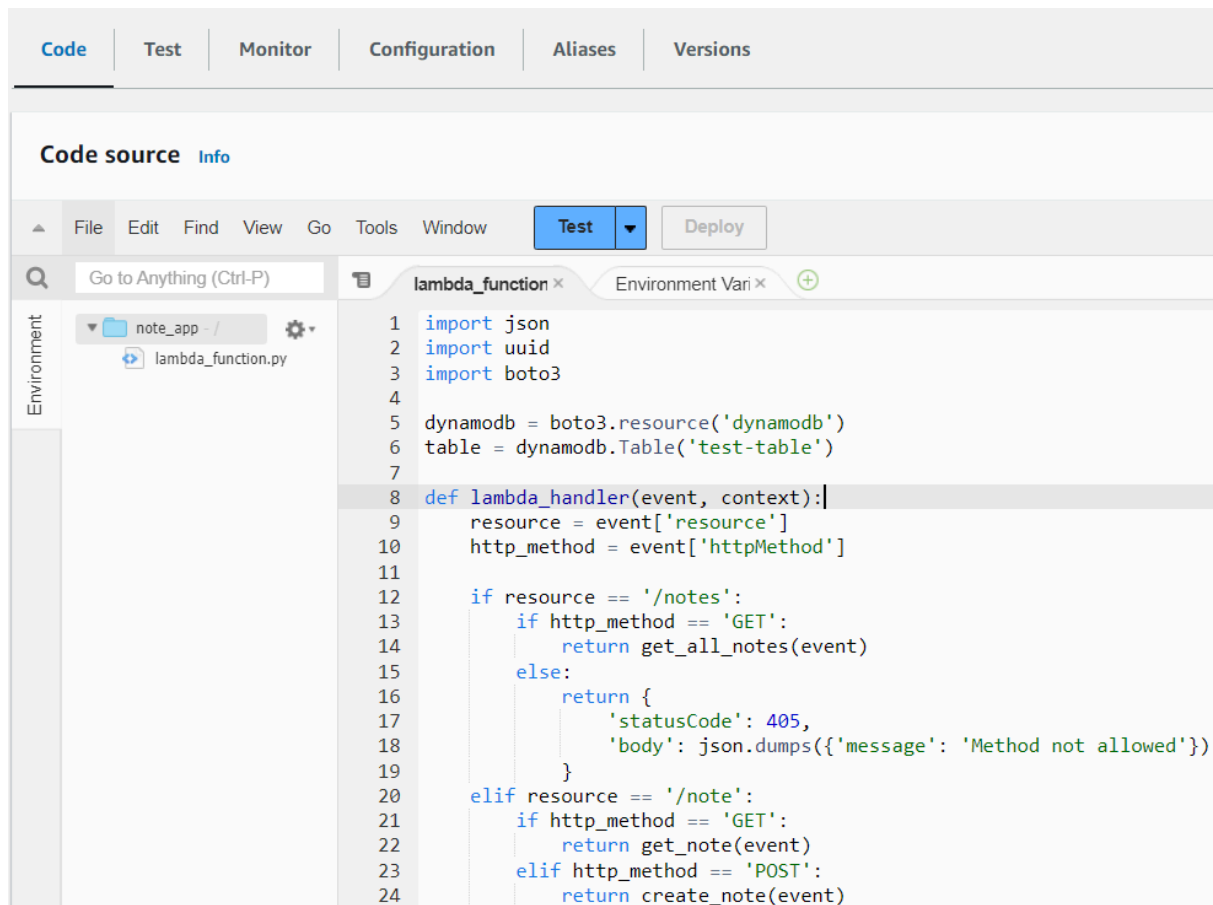
- List all the notes(GET)
- List a note(GET)
- Create a note(POST)
- Update a note(PUT)
- Delete a note(DELETE)

Step 1: Create a table in DynamoDB to store all the notes. I created a table named “test-table” with partition key “id”.



Step 2: Create a lambda function

I created a lambda function named “note_app”:



The above lambda function includes following code:

```
import json
```

```
import uuid
```

```
import boto3
```

```
dynamodb = boto3.resource('dynamodb')
```

```
table = dynamodb.Table('test-table')
```

```
def lambda_handler(event, context):
```

```
    resource = event['resource']
```

```
    http_method = event['httpMethod']
```

```
    if resource == '/notes':
```

```
if http_method == 'GET':
    return get_all_notes(event)
else:
    return {
        'statusCode': 405,
        'body': json.dumps({'message': 'Method not allowed'})
    }
elif resource == '/note':
    if http_method == 'GET':
        return get_note(event)
    elif http_method == 'POST':
        return create_note(event)
    elif http_method == 'PUT':
        return update_note(event)
    elif http_method == 'DELETE':
        return delete_note(event)
    else:
        return {
            'statusCode': 405,
            'body': json.dumps({'message': 'Method not allowed'})
        }
else:
    return {
        'statusCode': 404,
        'body': json.dumps({'message': 'Resource not found'})
    }
```



```

        "Access-Control-Allow-Headers": "Content-Type",
        "Access-Control-Allow-Methods": "GET"
    },
    'body': json.dumps(response['Item'])
}
else:
    return {
        'statusCode': 404,
        'body': json.dumps({'message': 'Note not found'})
    }
except Exception as e:
    return {
        'statusCode': 500,
        'body': json.dumps({'message': str(e)})
    }

def create_note(event):
    try:
        note_id = str(uuid.uuid4())
        note_data = json.loads(event['body'])
        note_data['id'] = note_id
        table.put_item(Item=note_data)
        return {
            'statusCode': 201,
            "headers": {
                "Access-Control-Allow-Origin": "*",
                "Access-Control-Allow-Headers": "Content-Type",

```

```

        "Access-Control-Allow-Methods": "POST"

    },

    'body': json.dumps({'message': 'Note created successfully'})

}

except Exception as e:

    return {

        'statusCode': 500,

        'body': json.dumps({'message': str(e)})

    }

def update_note(event):

    try:

        note_id = event['queryStringParameters']['id']

        update_data = json.loads(event['body'])

        update_expression = 'SET ' + ', '.join([f'#{k} = :{k}' for k in update_data.keys()])

        expression_attribute_names = {f'#{k}': k for k in update_data.keys()}

        expression_attribute_values = {f':{k}': v for k, v in update_data.items()}

        table.update_item(

            Key={'id': note_id},

            UpdateExpression=update_expression,

            ExpressionAttributeNames=expression_attribute_names,

            ExpressionAttributeValues=expression_attribute_values

        )

    return {

        'statusCode': 200,

        "headers": {

            "Access-Control-Allow-Origin": "*",


```

```

        "Access-Control-Allow-Headers": "Content-Type",
        "Access-Control-Allow-Methods": "PUT"
    },
    'body': json.dumps({'message': 'Note updated successfully'})
}

except Exception as e:
    return {
        'statusCode': 500,
        'body': json.dumps({'message': str(e)})
    }

def delete_note(event):
    try:
        note_id = event['queryStringParameters']['id']
        table.delete_item(Key={'id': note_id})
        return {
            'statusCode': 204,
            'body': json.dumps({'message': 'Note deleted successfully'})
        }
    except Exception as e:
        return {
            'statusCode': 500,
            'body': json.dumps({'message': str(e)})
        }
}

```

Step 3: Navigate to API Gateway to create API. Select lambda function (note_app) for all API requests. Here is a sample of the GET request. Similarly, other methods are created.

Method type

GET

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

arn:aws:lambda:us-east-1:695898526652:function:note

i 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.

☒ **Default timeout**

The default timeout is 29 seconds.

Cancel

Create method

After creating all the methods, it looks as below:

[API Gateway](#) > [APIs](#) > Resources - note-app (ovsr98va4c)

Resources

Create resource

/

/note

DELETE

GET

OPTIONS

POST

PUT

/notes

GET

OPTIONS

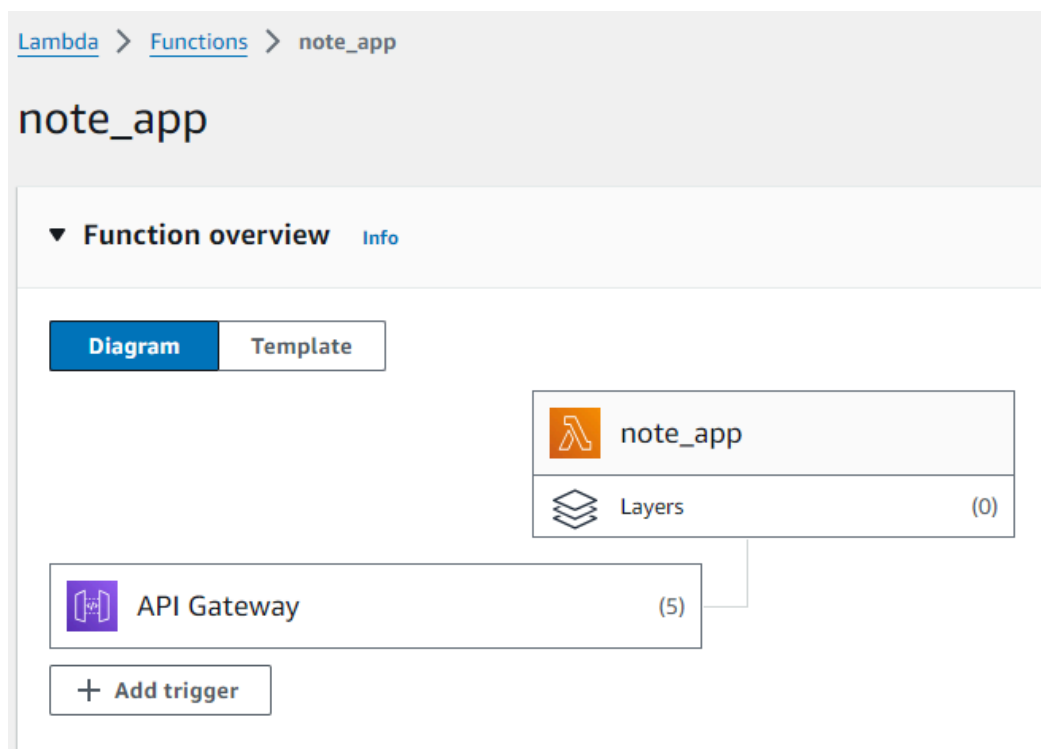
Resource details

Path
/

Methods (0)

Method type	Integration type
-------------	------------------

Overview of lambda function architecture:



Step 4: Lets now test all the APIs in postman.

Post note

The screenshot shows the Postman interface for a POST request. The URL is `https://ovsr98va4c.execute-api.us-east-1.amazonaws.com/test/note`. The request body is a JSON object: `{ "title": "Grocery List", "content": "Need to pick up eggs, milk, bread, apples, and chicken breasts for the week. Running low on essentials and need to restock to ensure meals are covered." }`. The response status is 201 Created, with a time of 325 ms and a size of 491 B. The response body is a JSON object: `{ "message": "Note created successfully" }`.

HTTP note app / Post note

POST `https://ovsr98va4c.execute-api.us-east-1.amazonaws.com/test/note` 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   "title": "Grocery List",
3   "content": "Need to pick up eggs, milk, bread, apples, and chicken breasts for the week. Running low on essentials and need to restock
  to ensure meals are covered."
4 }
```

Body Cookies Headers (10) Test Results (1/1) Status: 201 Created Time: 325 ms Size: 491 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "message": "Note created successfully"
3 }
```

Get note by id

The screenshot shows the Postman interface for a GET request. The URL is `https://ovsr98va4c.execute-api.us-east-1.amazonaws.com/test/note?id=3ebc94a8-f9a5-4353-adea-5a4915f269f5`. The request has a query parameter `id=3ebc94a8-f9a5-4353-adea-5a4915f269f5`. The response status is 200 OK, with a time of 335 ms and a size of 683 B. The response body is a JSON object: `{ "content": "Need to pick up eggs, milk, bread, apples, and chicken breasts for the week. Running low on essentials and need to restock to ensure meals are covered.", "id": "3ebc94a8-f9a5-4353-adea-5a4915f269f5", "title": "Grocery List" }`.

HTTP note app / Get note

GET `https://ovsr98va4c.execute-api.us-east-1.amazonaws.com/test/note?id=3ebc94a8-f9a5-4353-adea-5a4915f269f5` Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

<input checked="" type="checkbox"/> Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/> id	3ebc94a8-f9a5-4353-adea-5a4915f269f5			
Key	Value	Description		

Body Cookies Headers (10) Test Results Status: 200 OK Time: 335 ms Size: 683 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "content": "Need to pick up eggs, milk, bread, apples, and chicken breasts for the week. Running low on essentials and need to restock
  to ensure meals are covered.",
3   "id": "3ebc94a8-f9a5-4353-adea-5a4915f269f5",
4   "title": "Grocery List"
5 }
```

Update note by id

HTTP

note app / Update note

PUT

https://ovsr98va4c.execute-api.us-east-1.amazonaws.com/test/note?id=3ebc94a8-f9a5-4353-adea-5a4915f269f5

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

1

{

2

"title": "Grocery List",

3

"content": "Need to pick up eggs, milk, bread, and apples"

4

}

Body

Cookies

Headers (10)

Test Results (1/1)

Status: 200 OK

Time: 73

Pretty

Raw

Preview

Visualize

JSON

1

{

2

"message": "Note updated successfully"

3

}

Delete note

HTTP

note app / Delete note

📄

DELETE

https://ovsr98va4c.execute-api.us-east-1.amazonaws.com/test/note?id=3ebc94a8-f9a5-4353-adea-5a4915f269f5

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Query Params

<input checked="" type="checkbox"/>	Key	Value	Description
<input checked="" type="checkbox"/>	id	3ebc94a8-f9a5-4353-adea-5a4915f269f5	
	Key	Value	Description

Body

Cookies

Headers (7)

Test Results (1/1)

Status: 204 No Content

Time: 2.19 s

Size: 341 B

Pretty

Raw

Preview

Visualize

JSON

1

Get all notes

HTTP note app / Get notes

GET ▼ https://ovsr98va4c.execute-api.us-east-1.amazonaws.com/test/notes

Params Authorization Headers (6) Body Pre-request Script Tests ● Settings

Query Params

	Key	Value
	Key	Value

Body Cookies Headers (10) Test Results (1/1)

Pretty Raw Preview Visualize JSON ▼ ↻

```
1  [
2    {
3      "content": "Exercise, read, save money, learn new skill.",
4      "id": "8dce263c-4e54-48cf-b833-61410af2f080",
5      "title": "Personal Goals"
6    },
7    {
8      "content": "Review last week, assign tasks, address concerns, set goals."
9      "id": "c080ee8e-fbcc-462d-a6ab-6efc32099ffd",
10     "title": "Meeting Agenda"
11   },
12   {
13     "content": "Need eggs, milk, bread, apples, and chicken breasts.",
14     "id": "d6437487-9fe7-42a9-a81c-1b05933e859f",
15     "title": "Grocery List"
16   }
17 ]
```

All the methods are functioning well.

Building a Serverless Web Application

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

I have already created an API Gateway that triggers Lambda which interacts with the DynamoDB table as represented above.

Now, I will be hosting a static web application on S3 bucket.

Step 1: Created a bucket named “static-website-by-kristina” .

"default-account-dashboard" configuration.

General purpose buckets

Directory buckets

General purpose buckets (5) [Info](#)

Refresh

Copy ARN

Empty

Delete

Create bucket

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

Find buckets by name

< 1 > ⚙

	Name ▲	AWS Region ▼	Access ▼	Creation date ▼
<input type="radio"/>	bucket1-for-task	US East (N. Virginia) us-east-1	Bucket and objects not public	February 20, 2024, 11:50:53 (UTC+05:45)
<input type="radio"/>	bucket2-for-task	US East (N. Virginia) us-east-1	Bucket and objects not public	February 20, 2024, 11:51:04 (UTC+05:45)
<input type="radio"/>	bucket3-for-task	US East (N. Virginia) us-east-1	Bucket and objects not public	February 20, 2024, 11:51:17 (UTC+05:45)
<input type="radio"/>	kristina-bucket	US East (N. Virginia) us-east-1	Public	February 16, 2024, 13:31:28 (UTC+05:45)
<input type="radio"/>	static-website-by-kristina	US East (N. Virginia) us-east-1	Public	February 20, 2024, 19:59:11 (UTC+05:45)

Step 2: Enable static website hosting which is under the “Properties” tab inside the bucket.

Static website hosting

Edit

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

Static website hosting

Enabled

Hosting type

Bucket hosting

Bucket website endpoint


When you configure your bucket as a static website, the website is available at the AWS Region-specific website endpoint of the bucket. [Learn more](#)

<http://static-website-by-kristina.s3-website-us-east-1.amazonaws.com>


Step 3: Made the bucket publicly accessible by disabling block public access and also added a statement that allows public access to all the objects inside the “static-website-by-kristina” bucket.

Permissions overview

Access

 Public

Block public access (bucket settings)

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these set objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#) 

Block *all* public access

 Off

► Individual Block Public Access settings for this bucket

Bucket policy

The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects own

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::static-website-by-kristina/*"
    }
  ]
}
```

Step 4: Uploaded a folder named “static website” which has two files “index.html” that includes the home page and “error.html” that includes the error page.

Objects

Properties

Permissions

Metrics

Management

Access Point

Objects (1) Info

Copy S3 URI

Copy URL

Download

Open

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in a bucket and grant them permissions. [Learn more](#)

Find objects by prefix

<input type="checkbox"/>	Name	Type	Last modified
<input type="checkbox"/>	static_website/	Folder	-

Amazon S3

>

Buckets

>

static-website-by-kristina

>

static_website/

static_website/

Objects

Properties

Objects (2) Info

Copy S3 URI

Copy URL

Download

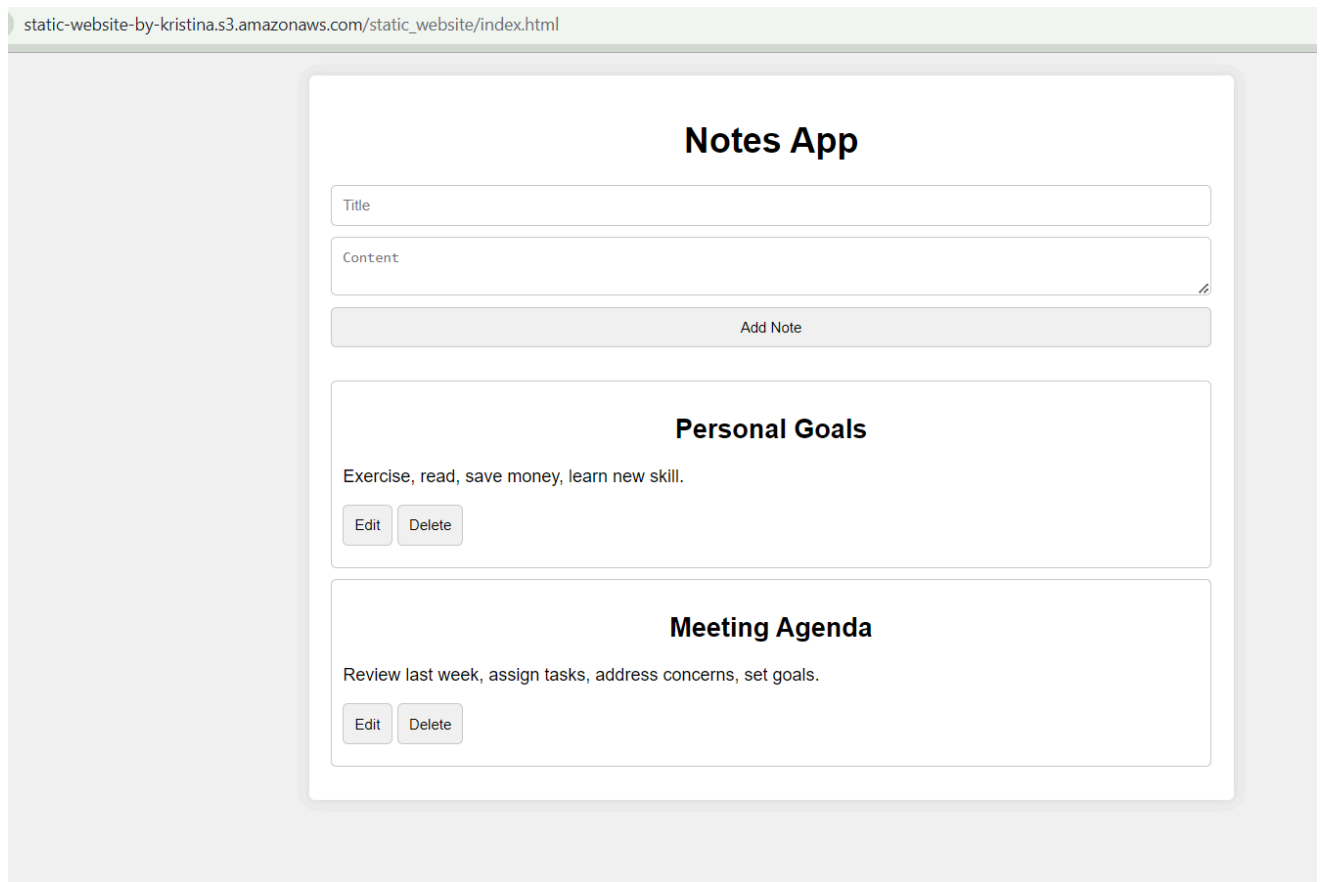
Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in a bucket and grant them permissions. [Learn more](#)

Find objects by prefix

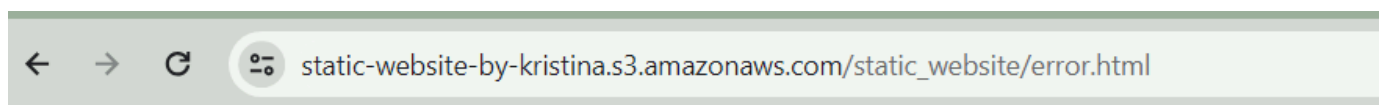
<input type="checkbox"/>	Name	Type	Last modified
<input type="checkbox"/>	error.html	html	Feb 10 2020
<input type="checkbox"/>	index.html	html	Feb 10 2020

Now, we can access the static website by using the object url of “index.html”.

Step 5: Copy object url of “index.html” and paste it in new tab. You can see a note application.



If there is any error, then you will be redirected to the error page.





Error

New note has been added as follow:

Notes App

Grocery List

Need eggs, milk, bread, apples, and chicken breasts.





Add Note

Notes App

Title

Content



Add Note

Grocery List

Need eggs, milk, bread, apples, and chicken breasts.

EditDelete

Personal Goals

Exercise, read, save money, learn new skill.

EditDelete

Serverless Data Processing Pipeline

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

I will create 3 buckets. When I upload a text file on bucket 1, it will automatically replicate on bucket 2 and the text inside the file will be converted to uppercase and automatically uploaded to bucket 3.

Step 1: Create three buckets.

General purpose buckets					Directory buckets	
General purpose buckets (5) Info <div> <input type="button" value="Refresh"/> <input type="button" value="Copy ARN"/> <input type="button" value="Empty"/> <input type="button" value="Delete"/> <input type="button" value="Create bucket"/> </div>						
Buckets are containers for data stored in S3. Learn more						
<input type="text" value="Find buckets by name"/>					< 1 > <input type="button" value="Settings"/>	
	Name ▲	AWS Region ▼	Access ▼	Creation date ▼		
<input type="radio"/>	bucket1-for-task	US East (N. Virginia) us-east-1	Bucket and objects not public	February 20, 2024, 11:50:53 (UTC+05:45)		
<input type="radio"/>	bucket2-for-task	US East (N. Virginia) us-east-1	Bucket and objects not public	February 20, 2024, 11:51:04 (UTC+05:45)		
<input type="radio"/>	bucket3-for-task	US East (N. Virginia) us-east-1	Bucket and objects not public	February 20, 2024, 11:51:17 (UTC+05:45)		

Step 2: Create a lambda function named “replicate” that performs specified functionality and deploy it.

Code source [Info](#)

File
Edit
Find
View
Go
Tools
Window

replicate - /

☒ lambda_function.py

lambda_function ×

Environment Vari ×

```

1 import boto3
2 import json
3
4 s3 = boto3.client('s3')
5
6 def lambda_handler(event, context):
7     source_bucket = event['Records'][0]['s3']['bucket']['name']
8     source_key = event['Records'][0]['s3']['object']['key']
9
10     destination_bucket = 'bucket2-for-task'
11     destination_key = source_key
12     s3.copy_object(CopySource={'Bucket': source_bucket, 'Key': source_key},
13                   Bucket=destination_bucket, Key=destination_key)
14
15     response = s3.get_object(Bucket=destination_bucket, Key=destination_key)
16     content = response['Body'].read().decode('utf-8').upper()
17     destination_bucket_c = 'bucket3-for-task'
18     transformed_key = destination_key
19
20     print(content)
21     s3.put_object(Bucket=destination_bucket_c, Key=transformed_key, Body=content)
22     s3_a = boto3.resource('s3')
23     s3_a.meta.client.download_file(destination_bucket_c, transformed_key, 'C:/Users/kristinan/Downloads/destination_key')
24
25     print(f"File {key} copied to 'boto-s3-2' and updated version saved as '{destination_key}' in 'boto-s3'")
26
27     return {
28         'statusCode': 200,
29         'body': f"File {key} processed and saved successfully."
30     }
31

```

Step 3: Add trigger. The lambda function will be triggered by “bucket1-for-task” when a file is uploaded.

Add trigger

Trigger configuration [Info](#)

 **S3**
aws asynchronous storage

Bucket

Choose or enter the ARN of an S3 bucket that serves as the event source. The bucket must be in the same region as the function.

×

↺

Bucket region: us-east-1

Event types

Select the events that you want to have trigger the Lambda function. You can optionally set up a prefix or suffix for an event. However, for each bucket, individual events cannot have multiple configurations with overlapping prefixes or suffixes that could match the same object key.

All object create events

×

Prefix - optional

Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters.

e.g. images/

Suffix - optional

Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters.

e.g. .jpg

Recursive invocation

If your function writes objects to an S3 bucket, ensure that you are using different S3 buckets for input and output. Writing to the same bucket increases the risk of creating a recursive invocation, which can result in increased Lambda usage and increased costs. [Learn more](#)

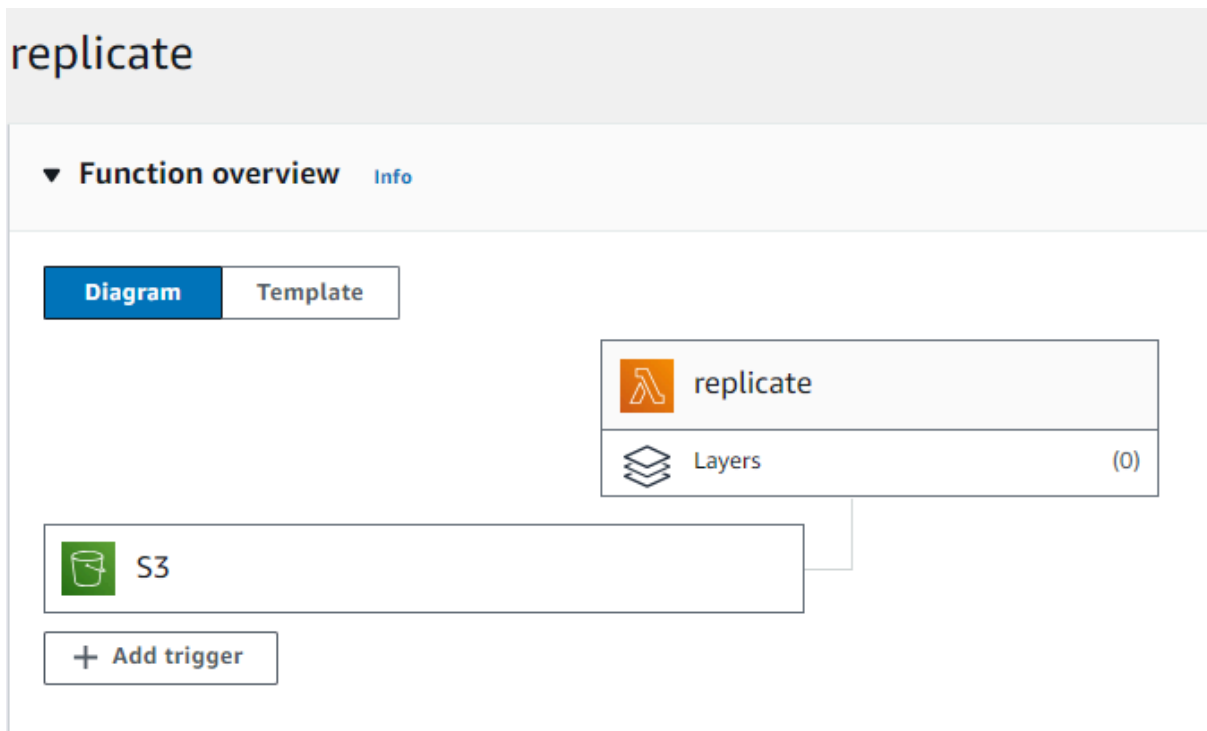
- ☒ I acknowledge that using the same S3 bucket for both input and output is not recommended and that this configuration can cause recursive invocations, increased Lambda usage, and increased costs.

Lambda will add the necessary permissions for AWS S3 to invoke your Lambda function from this trigger. [Learn more](#) about the Lambda permissions model.

Cancel

Add

This is the final lambda architecture:



Step 4: Now, we will try uploading a text file in “bucket1-for-task”.

Drag and drop files and folders you want to upload here, or choose **Add files** or **Add folder**.

Files and folders (1 Total, 32.0 B) [Remove](#) [Add files](#) [A](#)

All files and folders in this table will be uploaded.

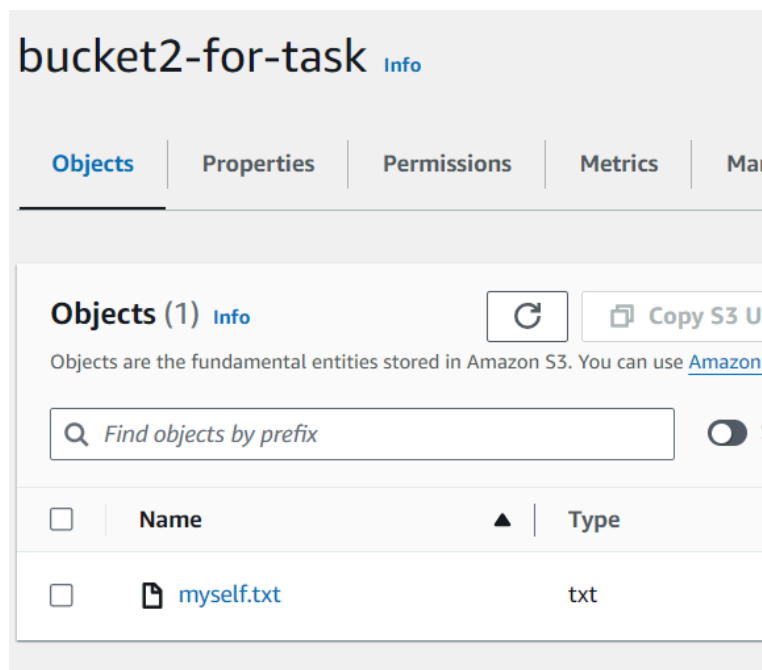
Find by name

<input type="checkbox"/>	Name	Folder
<input type="checkbox"/>	myself.txt	-

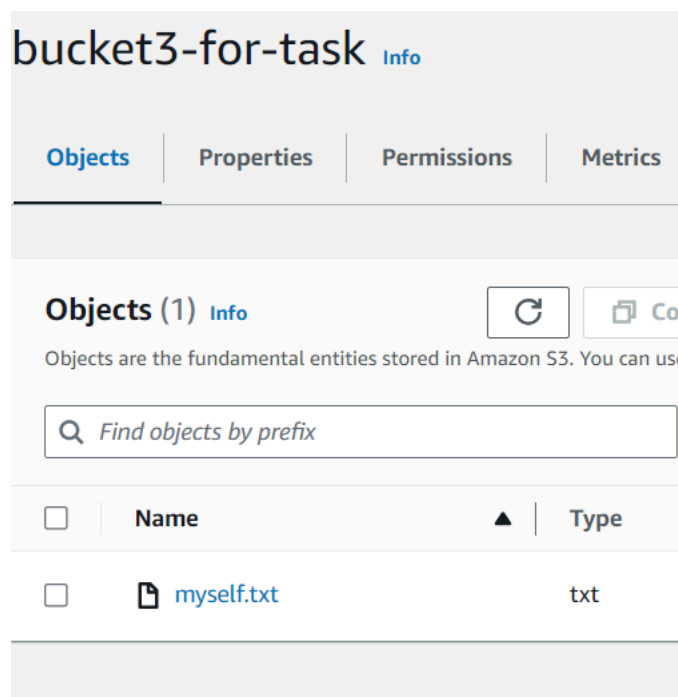
The text file contains following content:

```
myself
File Edit View
Hi! My name is Kristina Neupane.
```

In “bucket2-for-task”, the text file is automatically replicated:



Similarly, in “bucket3-for-task”, a text file is uploaded.



Step 5: Let’s check the logs in “CloudWatch” to check the pipeline performance and verify if the lambda function is executed.

Navigate to the CloudWatch and inside Logs, there is a “Log groups”. There you can see a list of lambda functions. Select the “replicate” lambda function and there will be the log streams. You can select the latest log stream.

Log events

You can use the filter bar below to search for and match terms, phrases, or values in your log events. [Learn more about filter patterns](#)

	Timestamp	Message
		No older events at this moment. Retry
▶	2024-02-23T21:00:26.742+05:45	INIT_START Runtime Version: python:3.12.v19 Runtime Version ARN: arn:aws:lambda:us-east-:
▶	2024-02-23T21:00:27.249+05:45	START RequestId: c5e01a95-2e0f-4469-82c0-2dd4573658ce Version: \$LATEST
▶	2024-02-23T21:00:27.899+05:45	HI! MY NAME IS KRISTINA NEUPANE.
▶	2024-02-23T21:00:29.419+05:45	[ERROR] FileNotFoundError: [Errno 2] No such file or directory: 'C:/Users/kristinan/Down:
▶	2024-02-23T21:00:29.440+05:45	END RequestId: c5e01a95-2e0f-4469-82c0-2dd4573658ce
▶	2024-02-23T21:00:29.440+05:45	REPORT RequestId: c5e01a95-2e0f-4469-82c0-2dd4573658ce Duration: 2190.58 ms Billed Durat:
▶	2024-02-23T21:01:34.592+05:45	START RequestId: c5e01a95-2e0f-4469-82c0-2dd4573658ce Version: \$LATEST
▶	2024-02-23T21:01:35.144+05:45	HI! MY NAME IS KRISTINA NEUPANE.
▶	2024-02-23T21:01:36.318+05:45	[ERROR] FileNotFoundError: [Errno 2] No such file or directory: 'C:/Users/kristinan/Down:
▶	2024-02-23T21:01:36.339+05:45	END RequestId: c5e01a95-2e0f-4469-82c0-2dd4573658ce
▶	2024-02-23T21:01:36.339+05:45	REPORT RequestId: c5e01a95-2e0f-4469-82c0-2dd4573658ce Duration: 1746.94 ms Billed Durat:
▶	2024-02-23T21:03:34.591+05:45	START RequestId: c5e01a95-2e0f-4469-82c0-2dd4573658ce Version: \$LATEST
▶	2024-02-23T21:03:35.091+05:45	HI! MY NAME IS KRISTINA NEUPANE.
▶	2024-02-23T21:03:36.318+05:45	[ERROR] FileNotFoundError: [Errno 2] No such file or directory: 'C:/Users/kristinan/Down:
▶	2024-02-23T21:03:36.319+05:45	END RequestId: c5e01a95-2e0f-4469-82c0-2dd4573658ce
▶	2024-02-23T21:03:36.319+05:45	REPORT RequestId: c5e01a95-2e0f-4469-82c0-2dd4573658ce Duration: 1728.23 ms Billed Durat:

Here, we can see the text is printed in uppercase as we have added print statement in our lambda function.