AIM: To import CSV data into DynamoDB using Lambda and S3 Event Triggers:

As part of my learning curve on DynamoDB and its interaction with various AWS services, Here S3 event triggers an action on a Lambda function to import CSV data from S3 Bucket and do some transformation and saved into a DynamoDB table using AWS Management Console.

S3 Bucket

Amazon Simple Storage (Amazon S3) is an object storage service. You can store and protect any amount of data for a range of use cases, like backup and restore, data lakes, archive, mobile application, and websites. Infact this object storage service offering industry-leading scalability, data availability, security, and performance.

When someone creates an object or modifies (removes or updates) an object stored in your S3 bucket, S3 will trigger an event. You can use Lambda function to process such event notifications from S3.

So in this project we are adding a trigger to our Amazon S3 bucket to call our Lambda function whenever new data arrives. Amazon Lambda needs permission from the function's resource-based policy to invoke Lambda function so that we must give the S3 access permission to lambda.

An IAM Role

An **IAM Role** manages Amazon Web Services (AWS) users and their access to AWS accounts and services. It controls the level of access a user can have over an AWS account & set user, grant permission, and allows a user to use different features of an AWS account. Identity and access management is mainly used to manage users, groups, roles, and Access policies The account we created to sign in to Amazon web services is known as the root account and it holds all the administrative rights and has access to all parts of the account. The new user created an AWS account, by default they have no access to any services in the account & it is done with the help of IAM that the root account holder can implement access policies and grant permission to the user to access certain services.

AWS Lambda

AWS Lambda AWS Lambda is an Amazon serverless computing system that runs code and automatically manages the underlying computing resources like EC2. It is an event-driven computing service. It lets a person automatically run code in response to many types of events, such as HTTP requests from the Amazon API gateway, table updates in Amazon DynamoDB, and state transitions. It also enables the person to extend to other AWS services with custom logic and even creates its own back-end services. For example, just write the code and then upload it as a .zip file or any container image. The service works by running code on high-availability computer infrastructure.

DynamoDB

DynamoDB is a fully managed NoSQL database service that allows to create database tables that can store and retrieve any amount of data. It automatically manages the data traffic of tables over multiple servers and maintains performance. It also relieves the customers from the burden of operating and scaling a distributed database. Hence, hardware provisioning, setup, configuration, replication, software patching, cluster scaling, etc. is managed by Amazon.

DynamoDB is fully integrated with AWS Backups. You can use the DynamoDB console, API, and AWS Command Line Interface (AWS CLI) to enable automatic backups and restore for your DynamoDB tables.

Objectives:

- Create an S3 bucket.

- Upload a CSV file.
- Creating Lambda Function with a timeout of more than 1 minute, which contains the code to import the CSV data into DynamoDB.

- Create a Amazon DynamoDB table.

- All associated IAM roles needed for the solution, configured according to the principle of least privilege.
- Test the CSV Data Import in Lambda
- Adding Event Triggers to the S3 Bucket to call our Lambda function whenever new data arrives.
- Test the setup - Testing S3 Event Trigger to Import New Data into DynamoDB
- Cleanup

Pre-requisites:

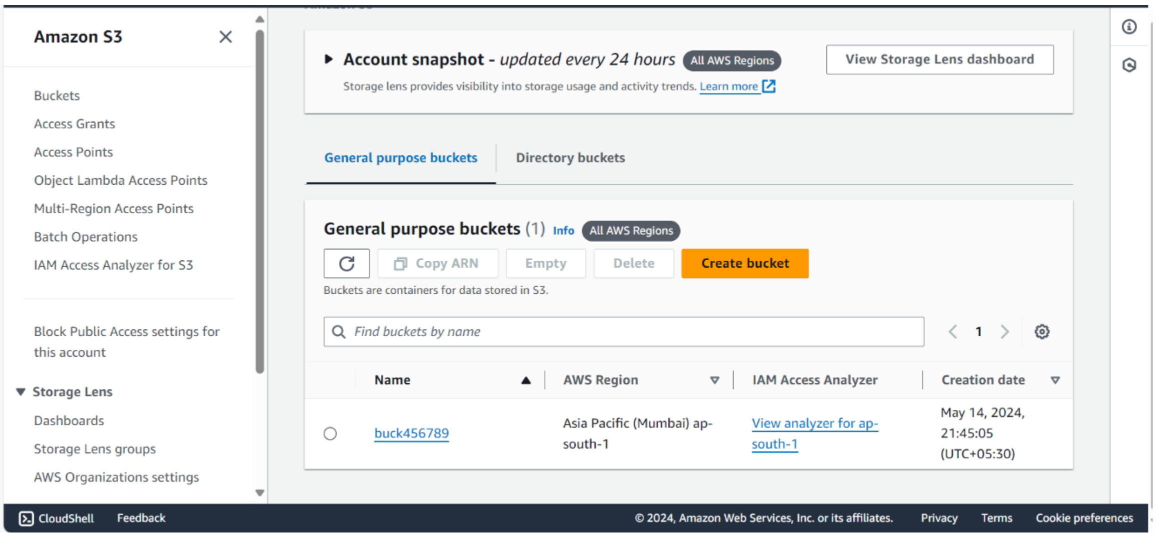
- AWS user account with admin access, not a root account.
- Create an IAM role.

1. Create an S3 bucket

- On Amazon S3 Console / Create bucket / Under Create bucket and do general configuration.

Bucket name: bucket456789

Create bucket.



Storage lens provides visibility into storage usage and activity trends. Learn more

General purpose buckets (1) Info All AWS Regions

Buckets are containers for data stored in S3.

Name	AWS Region	IAM Access Analyzer	Creation date
buck456789	Asia Pacific (Mumbai) ap-south-1	View analyzer for ap-south-1	May 14, 2024, 21:45:05 (UTC+05:30)

© 2024, Amazon Web Services, Inc. or its affiliates.

2. Upload a CSV file

- Click on the S3 bucket – bucket456789
- Upload a csv file here

File name: friends.csv

- Under bucket456789, Objects, Upload / Under Upload, For Files and folders / Add files

select - friends.csv

Upload

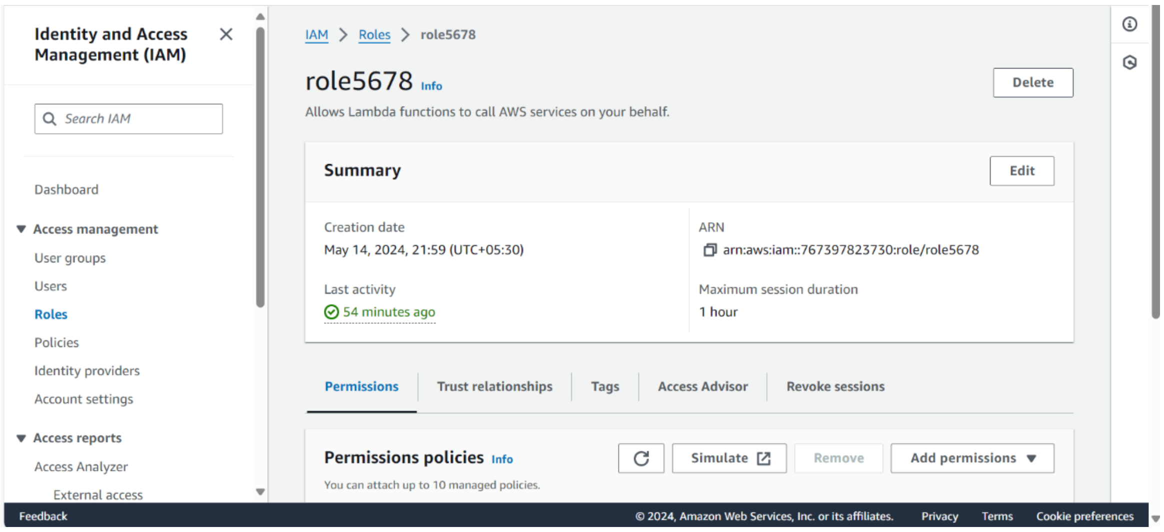
```
Id,Name,Subject
1,Sumit-Jindal,Maths
2,Varun,Science
3,Unnati,English
4,Shanu,Social
5,Sahil,Sports
6,Varnika,Arts
7,Vaibhav,nothing
```

3. Creating IAM Roles

All associated IAM roles needed for the solution, configured according to the principle of least privilege

- On the IAM dashboard / Roles / IAM dashboard / Create Role / Under Select trusted entity, Trusted entity type / Select AWS service / Under Use case / Select Lambda /
- Next
- Under Name, review, and create, Role details /

Role name: role5678
Create role



IAM > Roles > role5678

role5678

Allows Lambda functions to call AWS services on your behalf.

Summary

Creation date: May 14, 2024, 21:59 (UTC+05:30)

ARN: arn:aws:iam::767397823730:role/role5678

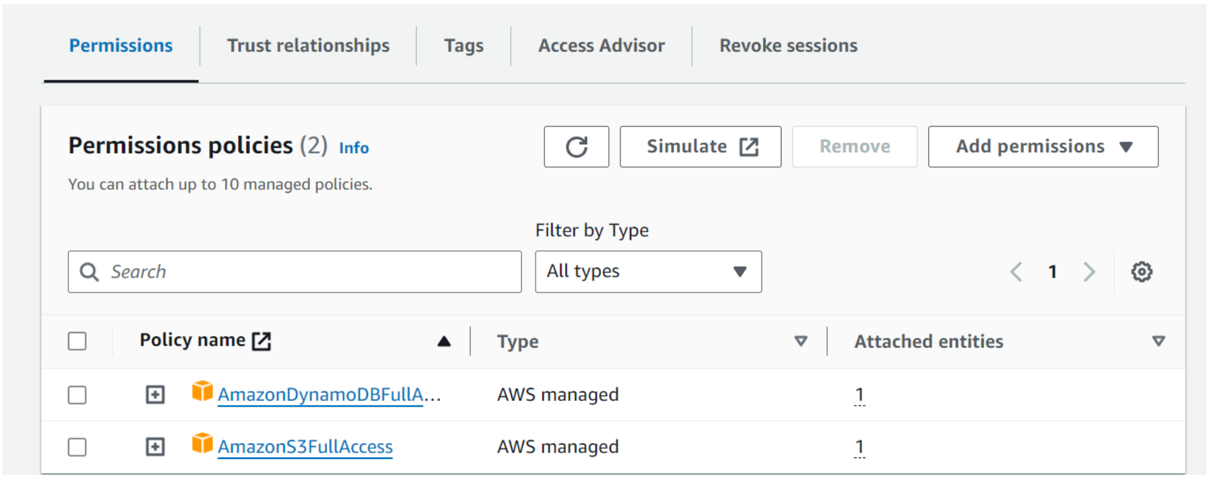
Last activity: 54 minutes ago

Maximum session duration: 1 hour

Permissions policies

You can attach up to 10 managed policies.

- Click the role you just created **role5678**
  drop-down Add permissions / select Attach policies
- Under Attach policy to csv-lambda-role / Search for **AmazonDynamoDBFullAccess** / Check the box / Attach poilicies
-
- Attach **AmazonS3FullAccess** in the same way.



Permissions policies (2) Info

You can attach up to 10 managed policies.

Policy name	Type	Attached entities
AmazonDynamoDBFullA...	AWS managed	1
AmazonS3FullAccess	AWS managed	1

4. Creating Lambda Function

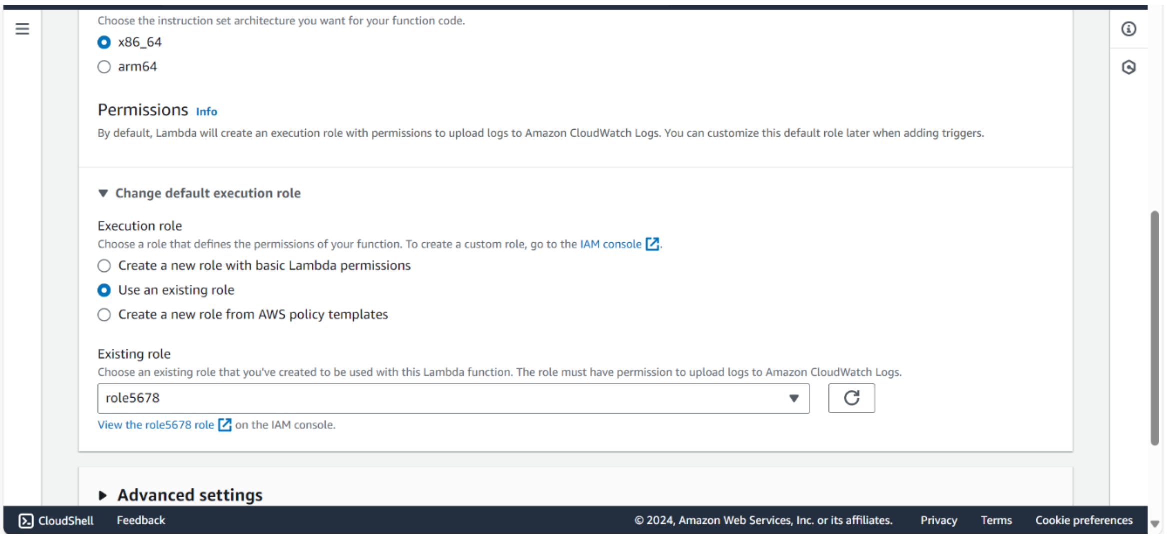
We have to create a Lambda function with a timeout of more than 1 minute, which already contains the code to import the CSV data from S3 bucket and transform that file and then saved into DynamoDB.

- On the Lambda Console / Functions / Create function / Select Author from scratch / Under Basic information

Function name: function5678
Runtime: From the drop-down choose **Python 3.12**

- Click on Change default execution role / select Use an existing Role
- Select the role you created just now - **csv-lambda-role**

Create function


Choose the instruction set architecture you want for your function code.

[✓ x86_64]
arm64

Permissions Info

By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▼ 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.

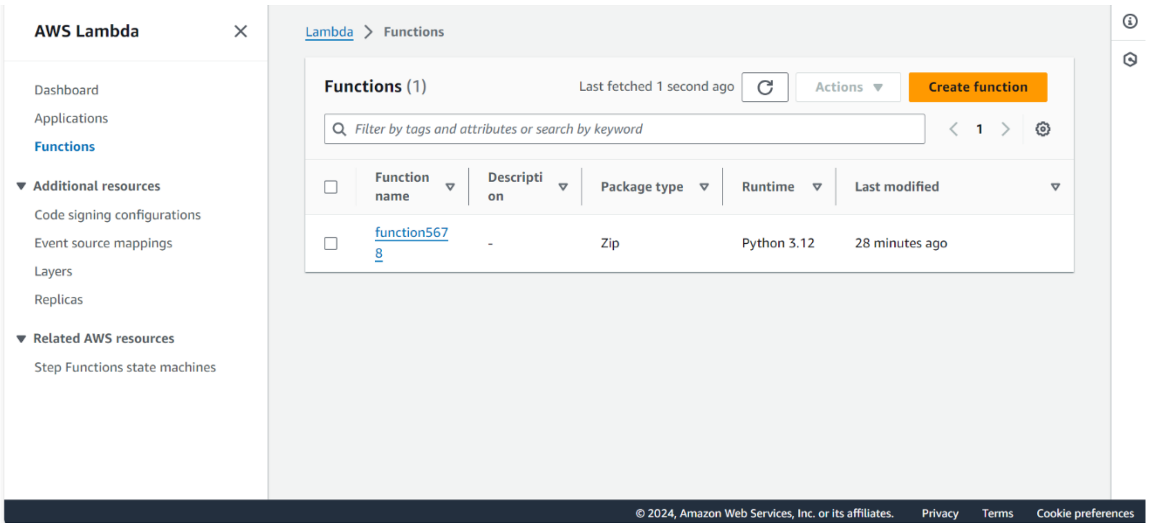
Create a new role with basic Lambda permissions
[✓ Use an existing role]
Create a new role from AWS policy templates

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 Logs.

role5678

View the role5678 role on the IAM console.



AWS Lambda

Dashboard
Applications
Functions

Additional resources
Code signing configurations
Event source mappings
Layers
Replicas

Related AWS resources
Step Functions state machines

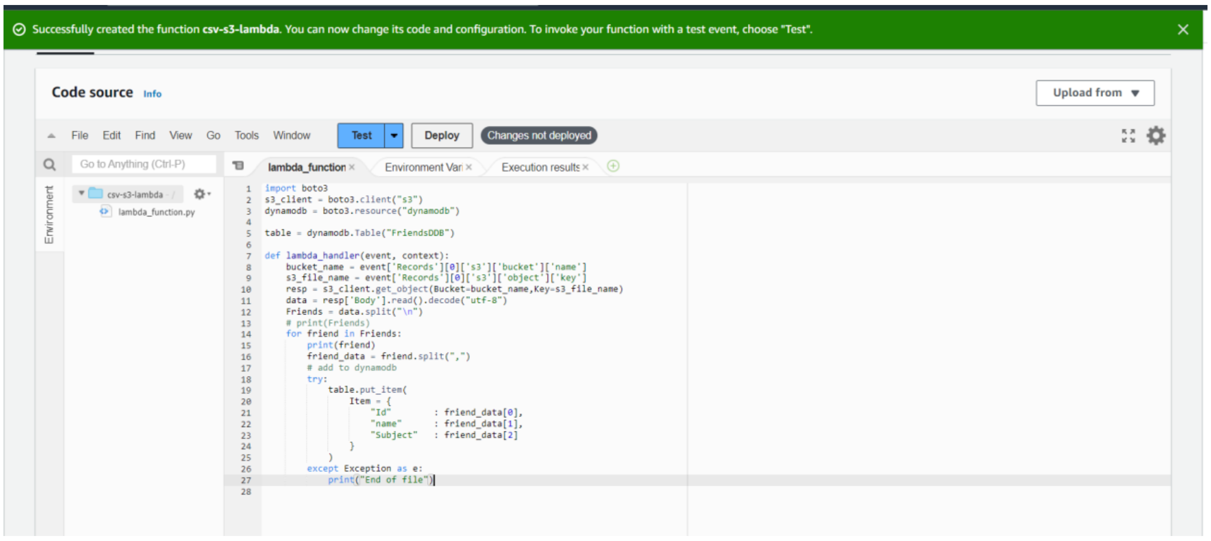
Lambda > Functions

Functions (1) Last fetched 1 second ago

Function name	Description	Package type	Runtime	Last modified
function5678	-	Zip	Python 3.12	28 minutes ago

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

- Once the function is created, it will open the main page of the Lambda function.
- The python code do the following:
    - Imports the CSV file from S3 bucket.
    - Splits the CSV data into multiple strings.
    - Uploads data into the DynamoDB table.
- Remove the existing code in the function code environment window.
- Write the code into the lambda_function.py



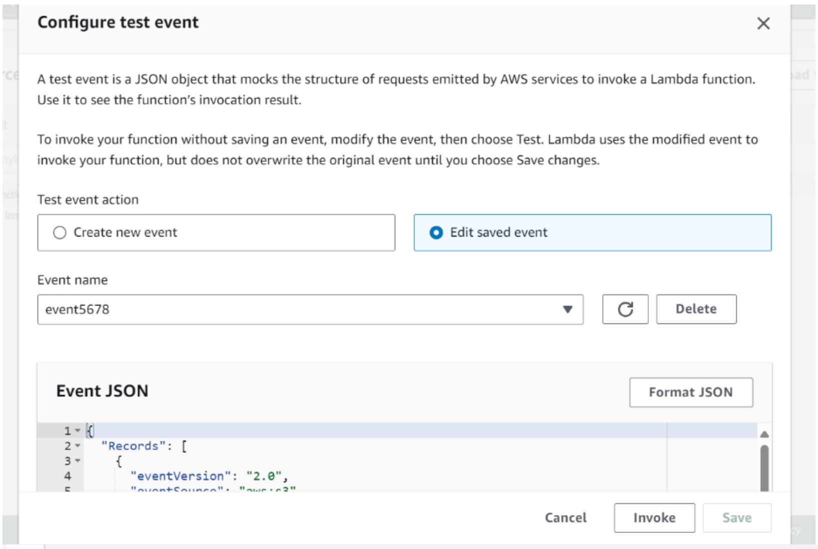
Successfully created the function **csv-s3-lambda**. You can now change its code and configuration. To invoke your function with a test event, choose "Test".

Code source

```python
import boto3
s3_client = boto3.client("s3")
dynamodb = boto3.resource("dynamodb")

table = dynamodb.Table("FriendsDDB")

def lambda_handler(event, context):
    bucket_name = event['Records'][0]['s3']['bucket']['name']
    s3_file_name = event['Records'][0]['s3']['object']['key']
    resp = s3_client.get_object(Bucket=bucket_name,Key=s3_file_name)
    data = resp['Body'].read().decode("utf-8")
    Friends = data.split("\n")
    # print(Friends)
    for friend in Friends:
        print(friend)
        friend_data = friend.split(",")
        # add to dynamodb
        try:
            table.put_item(
                Item = {
                    "Id"        : friend_data[0],
                    "name"      : friend_data[1],
                    "Subject"   : friend_data[2]
                }
            )
        except Exception as e:
            print("End of file")

```

- After updating the code, Click on **Deploy** button to save the code.
- Change the function timeout as follows:
    - Navigate to the Configuration
    - click on General configuration / click on Edit

- In the Edit Basic setting / change the Timeout value to 1 min
- Click on Save button



Configure test event

A test event is a JSON object that mocks the structure of requests emitted by AWS services to invoke a Lambda function. Use it to see the function's invocation result.

To invoke your function without saving an event, modify the event, then choose Test. Lambda uses the modified event to invoke your function, but does not overwrite the original event until you choose Save changes.

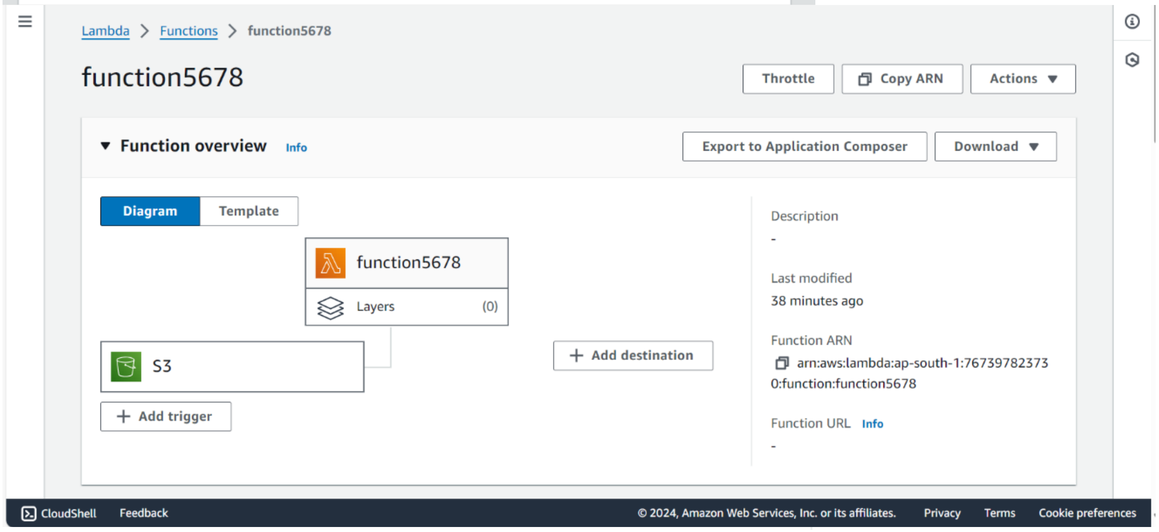
Test event action

( ) Create new event [✓ Edit saved event]

Event name: event5678

Event JSON

```
{
  "Records": [
    {
      "eventVersion": "2.0",
```



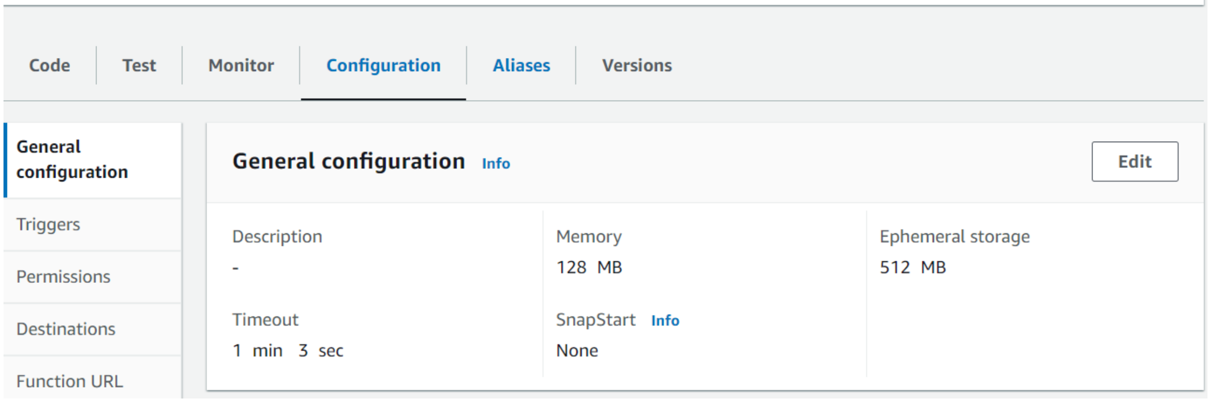
Lambda > Functions > function5678

function5678

Function overview

Description	-
Last modified	38 minutes ago
Function ARN	arn:aws:lambda:ap-south-1:767397823730:function:function5678
Function URL	-

© 2024, Amazon Web Services, Inc. or its affiliates.



General configuration

Description	Memory	Ephemeral storage
-	128 MB	512 MB
Timeout	SnapStart	
1 min 3 sec	None	

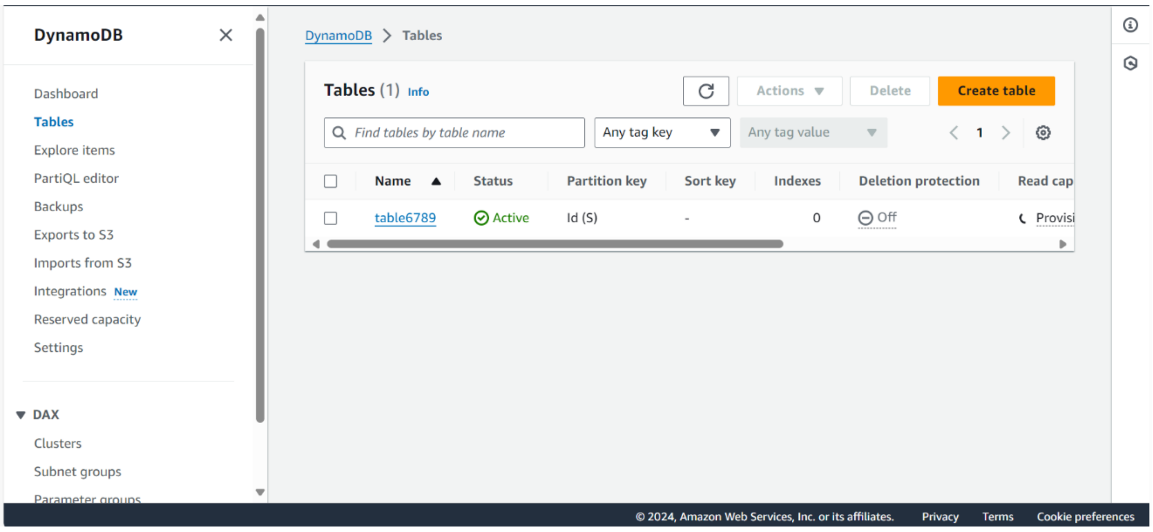
5. Creating table in DynamoDB

- On DynamoDB Dashboard / Tables / Under Create table , Table Details

Table Name: table6789
Partition key: Id
Type: String

Create Table

Status should be Active



Name	Status	Partition key	Sort key	Indexes	Deletion protection	Read cap
table6789	Active	Id (S)	-	0	Off	Provisi

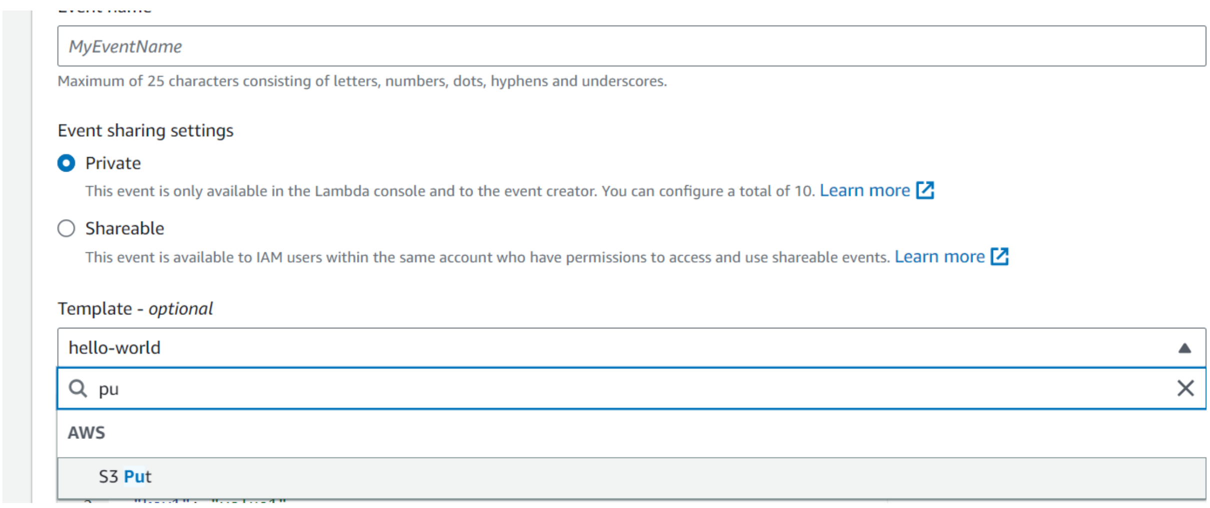
6. Test the CSV Data Import in Lambda:

- In the csv-s3-lambda lambda function page, click on the Test tab.
- Configure to test event data as follows:

Test event action: Create a new event

Name: Event name - csv

Template: Select Amazon S3 Put, Upon selection, it will be displayed as s3-put



MyEventName

Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.

Event sharing settings

● Private

This event is only available in the Lambda console and to the event creator. You can configure a total of 10. Learn more

○ Shareable

This event is available to IAM users within the same account who have permissions to access and use shareable events. Learn more

Template - *optional*

hello-world

pu

AWS

S3 Put

- Below in the JSON code:

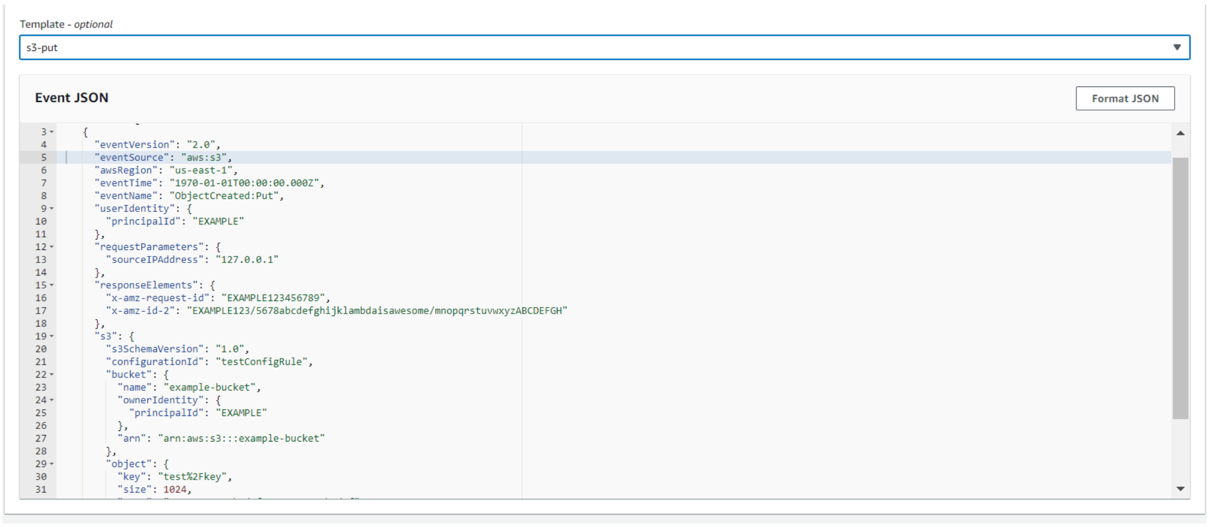
Under S3 → bucket → name → Enter Students_Details

arn": "arn:aws:s3:::table6789"

Under S3 → object → Key → Enter friends.csv

- Click on Create and then Save to save the changes.

Note: Make sure the S3 bucket name and file name are correct in the JSON.

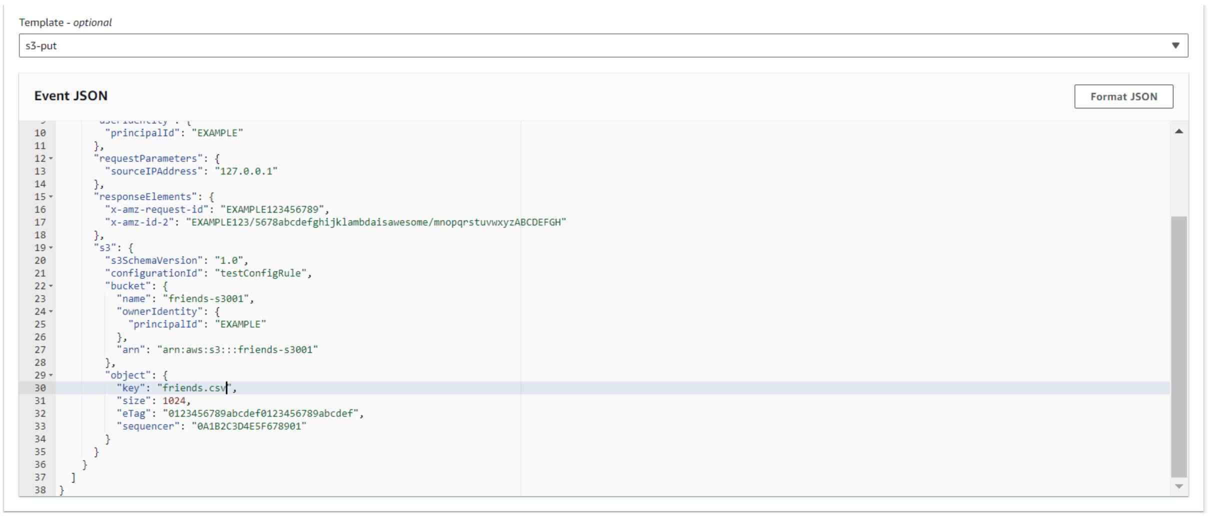


Template - *optional*

s3-put

Event JSON

```
    {
      "eventVersion": "2.0",
      "eventSource": "aws:s3",
      "awsRegion": "us-east-1",
      "eventTime": "1970-01-01T00:00:00.000Z",
      "eventName": "ObjectCreated:Put",
      "userIdentity": {
        "principalId": "EXAMPLE"
      },
      "requestParameters": {
        "sourceIPAddress": "127.0.0.1"
      },
      "responseElements": {
        "x-amz-request-id": "EXAMPLE123456789",
        "x-amz-id-2": "EXAMPLE123/5678abcdefghijklambdaisawesome/mnopqrstuvwxyzABCDEFGH"
      },
      "s3": {
        "s3SchemaVersion": "1.0",
        "configurationId": "testConfigRule",
        "bucket": {
          "name": "example-bucket",
          "ownerIdentity": {
            "principalId": "EXAMPLE"
          },
          "arn": "arn:aws:s3:::example-bucket"
        },
        "object": {
          "key": "test%2Fkey",
          "size": 1024,
```



Template - *optional*

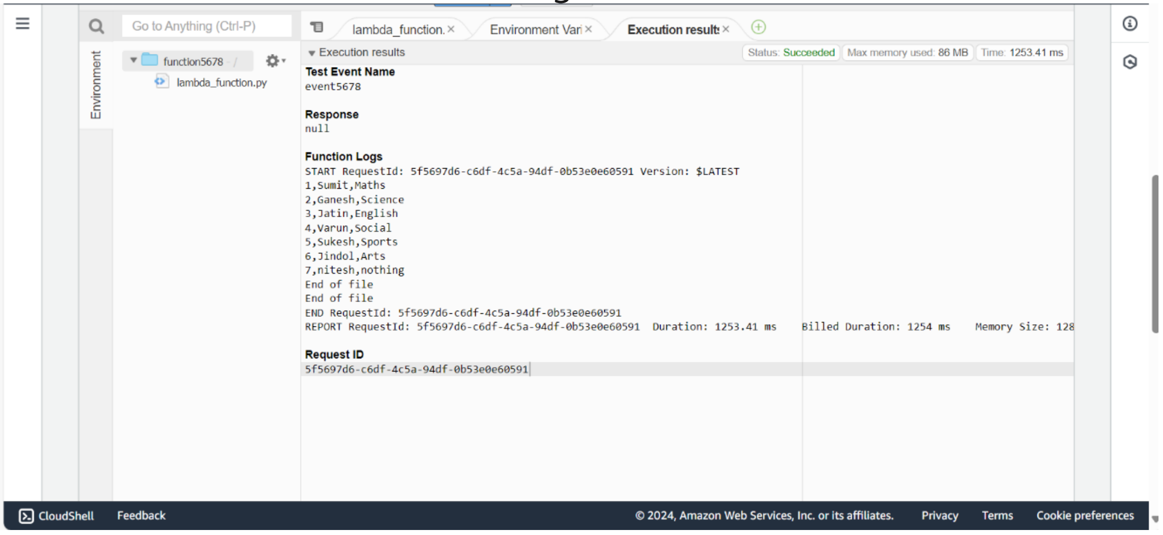
s3-put

Event JSON

Format JSON

```
          "principalId": "EXAMPLE"
        },
        "requestParameters": {
          "sourceIPAddress": "127.0.0.1"
        },
        "responseElements": {
          "x-amz-request-id": "EXAMPLE123456789",
          "x-amz-id-2": "EXAMPLE123/5678abcdefghijklambdaisawesome/mnopqrstuvwxyzABCDEFGH"
        },
        "s3": {
          "s3SchemaVersion": "1.0",
          "configurationId": "testConfigRule",
          "bucket": {
            "name": "friends-s3001",
            "ownerIdentity": {
              "principalId": "EXAMPLE"
            },
            "arn": "arn:aws:s3:::friends-s3001"
          },
          "object": {
            "key": "friends.csv",
            "size": 1024,
            "eTag": "0123456789abcdef0123456789abcdef",
            "sequencer": "0A1B2C3D4E5F678901"
          }
        }
      }
    ]
}
```

- Click on Test in top-right Corner to trigger the lambda function.
- Once the lambda function is successfully executed, you will be able to see a detailed success message with table data.



Status: Succeeded | Max memory used: 86 MB | Time: 1253.41 ms

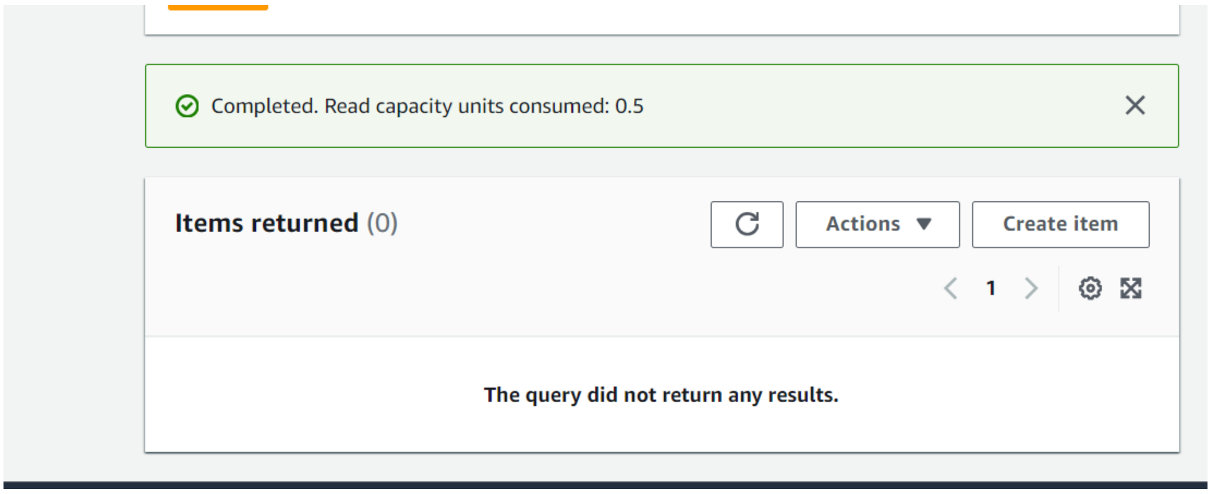
Test Event Name
```
event5678
```

Response
```
null
```

Function Logs
```
START RequestId: 5f5697d6-c6df-4c5a-94df-0b53e0e60591 Version: $LATEST
1,Sumit,Maths
2,Ganesh,Science
3,Jatin,English
4,Varun,Social
5,Sukesh,Sports
6,Jindol,Arts
7,nitesh,nothing
End of file
End of file
END RequestId: 5f5697d6-c6df-4c5a-94df-0b53e0e60591
REPORT RequestId: 5f5697d6-c6df-4c5a-94df-0b53e0e60591  Duration: 1253.41 ms    Billed Duration: 1254 ms    Memory Size: 128
```

Request ID
```
5f5697d6-c6df-4c5a-94df-0b53e0e60591
```

- Go to the DynamoDB table and then select the table6789 and click on Explore Table Items


Completed. Read capacity units consumed: 0.5

Items returned (0)

The query did not return any results.

7. Adding Event Triggers to the S3 Bucket to call our Lambda function whenever new data arrives.

- On the S3 Console / Click on the s3 bucket named **jindal1234**
- Click on the Properties tab / go down to Event notifications.
- Click on Create event notification button
- Under General configuration /

Name:

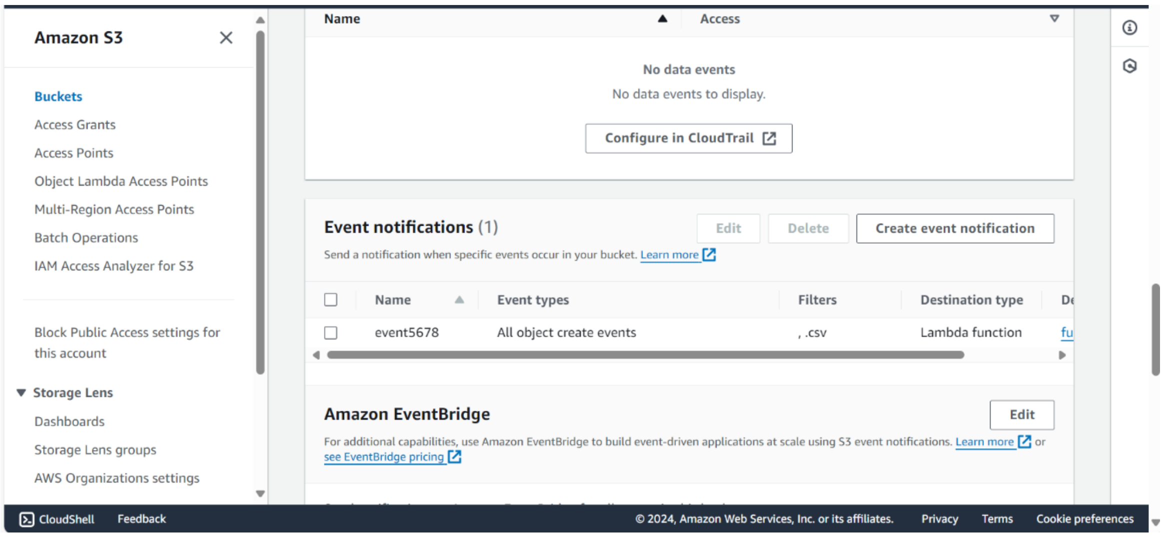
Suffix: .csv

All Object create events: check

Destination: Select Lambda Function

Lambda: Select function5678

Click on Save changes



Amazon S3

Buckets
Access Grants
Access Points
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
IAM Access Analyzer for S3

Block Public Access settings for this account

Storage Lens
Dashboards
Storage Lens groups
AWS Organizations settings

Name	Access

No data events
No data events to display.

Configure in CloudTrail

Event notifications (1)

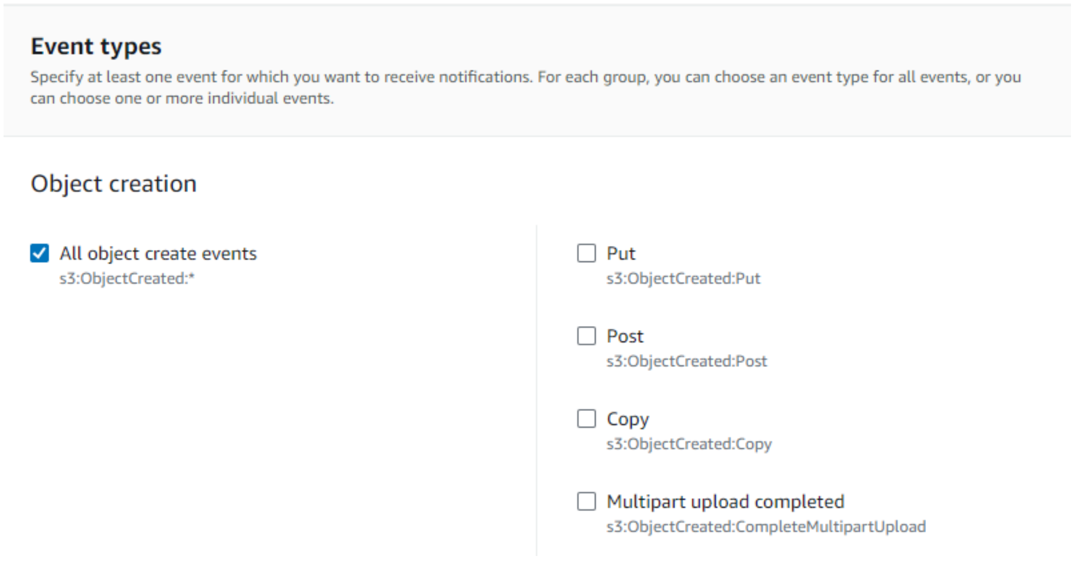
Send a notification when specific events occur in your bucket. Learn more

Name	Event types	Filters	Destination type	De
event5678	All object create events	, .csv	Lambda function	fu

Amazon EventBridge

For additional capabilities, use Amazon EventBridge to build event-driven applications at scale using S3 event notifications. Learn more or see EventBridge pricing

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences



Event types

Specify at least one event for which you want to receive notifications. For each group, you can choose an event type for all events, or you can choose one or more individual events.

Object creation

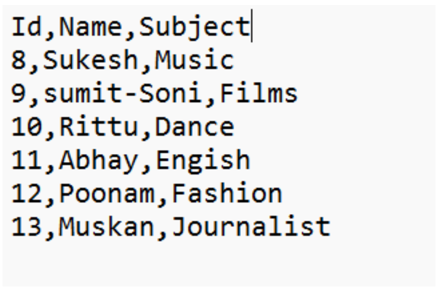
- [✓] All object create events s3:ObjectCreated:*
- [ ] Put s3:ObjectCreated:Put
- [ ] Post s3:ObjectCreated:Post
- [ ] Copy s3:ObjectCreated:Copy
- [ ] Multipart upload completed s3:ObjectCreated:CompleteMultipartUpload

- Now every time a CSV file is uploaded to our S3 bucket, it will trigger the lambda to import the CSV data into the DynamoDB table

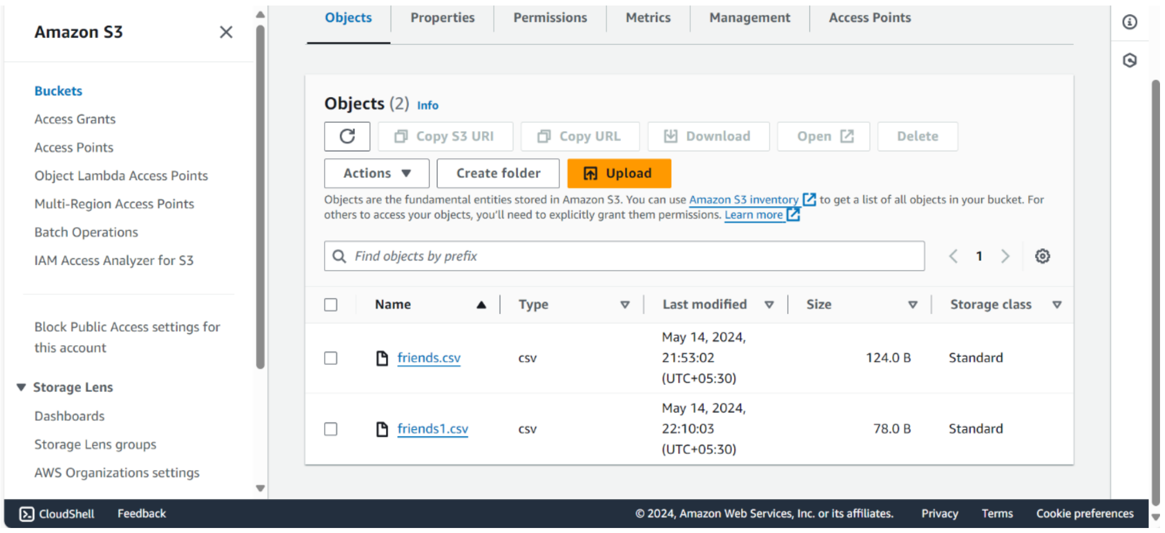
8. Test the setup - Testing S3 Event Trigger to Import New Data into DynamoDB

file: friends1.csv

Upload the friends1.csv file to the friends-s3 bucket



```
Id,Name,Subject
8,Sukesh,Music
9,sumit-Soni,Films
10,Rittu,Dance
11,Abhay,Engish
12,Poonam,Fashion
13,Muskan,Journalist
```

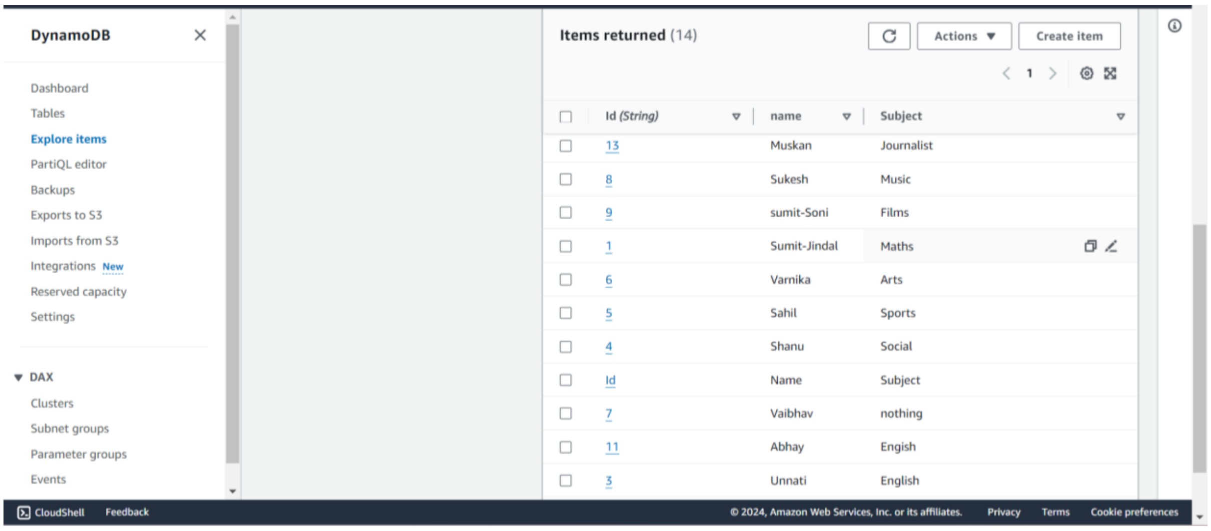


Objects (2) Info

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

Name	Type	Last modified	Size	Storage class
friends.csv	csv	May 14, 2024, 21:53:02 (UTC+05:30)	124.0 B	Standard
friends1.csv	csv	May 14, 2024, 22:10:03 (UTC+05:30)	78.0 B	Standard

- This upload event should triggered our Lambda function to import the CSV data into the DynamoDB table **table6789**.
- Go to the DynamoDB table **table6789** to see the changes.
- Click on the refresh button if items have not yet changed.
- You can see that new CSV data has been successfully imported into the DynamoDB table.



Items returned (14)

Id (String)	name	Subject
13	Muskan	Journalist
8	Sukesh	Music
9	sumit-Soni	Films
1	Sumit-Jindal	Maths
6	Varnika	Arts
5	Sahil	Sports
4	Shanu	Social
Id	Name	Subject
7	Vaibhav	nothing
11	Abhay	Engish
3	Unnati	English

9. Cleanup

- delete the Lambda function
- delete the S3 bucket
- delete the DynamoDB table
- delete IAM Role

What we have done so far

- We have successfully created an Amazon DynamoDB Table.
- We have successfully created a Lambda function and configured it to import CSV data from S3 into DynamoDB.
- We have created an S3 event to trigger our Lambda function.
- We have tested the import of a new CSV file to the DynamoDB table.