A PROJECT REPORT

on

“Data Migration and Transformation Tool for Amazon NO SQL Data Bases”

Submitted to

GUVI GEEK NETWORK

In partial fulfilment of the requirements for the award CCM

IN

MASTER DATA ENGINEERING COURSE

By

PALUKURI SRINIVAS

E5 – Data Engineering batch

Under the esteemed guidance of
DE mentors,

“GUVI”

GIVEN PROJECT STATEMENT

Project 1:

Project Title	**Data Migration and Transformation Tool for Amazon NO SQL Data Bases**
Technologies	**Python/PySpark,Requests,Zipfile,boto3,pandas,sqlalchemy, Amazon S3,Amazon NO SQL Data Bases**

Problem Statement:

You have a URL that points to a zip file. The zip file contains multiple JSON files. The JSON files contain multiple documents with various data structures. Your goal is to download the zip file from the URL, extract the data from the JSON files, store it in Amazon S3, and load it into Amazon RDS. You want to use Python or PySpark to perform these tasks. You may use any libraries or tools that are necessary to complete the task.

Approach:

To extract the data from a zip file that is available at a URL and load it into Amazon S3 and Amazon RDS (NoSQL), you can follow these steps:

1. Use the requests library to download the zip file from the URL.
2. Use the zipfile module to extract the data from the zip file.
3. Use the boto3 library or PySpark to store the data in Amazon S3.
4. Use the pandas library and sqlalchemy or PySpark to load the data from S3 into Amazon RDS (NoSQL).

Results:

The result of following these steps should be that the data from the zip file is extracted and stored in a list of dictionaries (if you are using Python) or a DataFrame (if you are using PySpark). Each dictionary or DataFrame row will represent a document from one of the JSON files in the zip file.

The data in the list or DataFrame will then be stored in Amazon S3 as JSON files. You will be able to access these JSON files using the boto3 library or the Amazon S3 web interface.

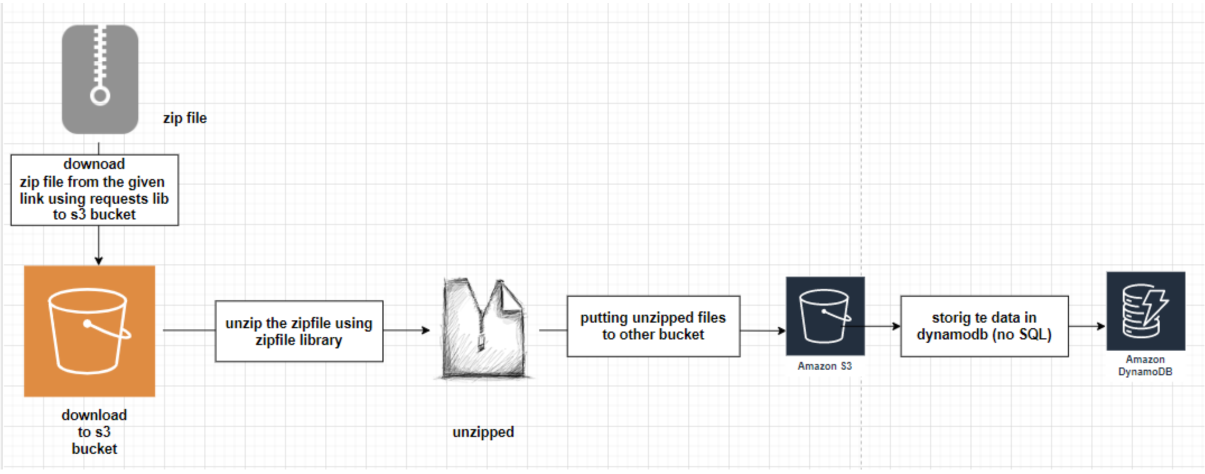
The data from the JSON files will also be loaded into Amazon RDS (NoSQL). You will be able to access the data in RDS using SQL queries. The data will be stored in a table in RDS, and the schema of the table will be determined by the structure of the JSON documents.

I hope this helps. Let me know if you have any further questions or need more assistance.

Project description:

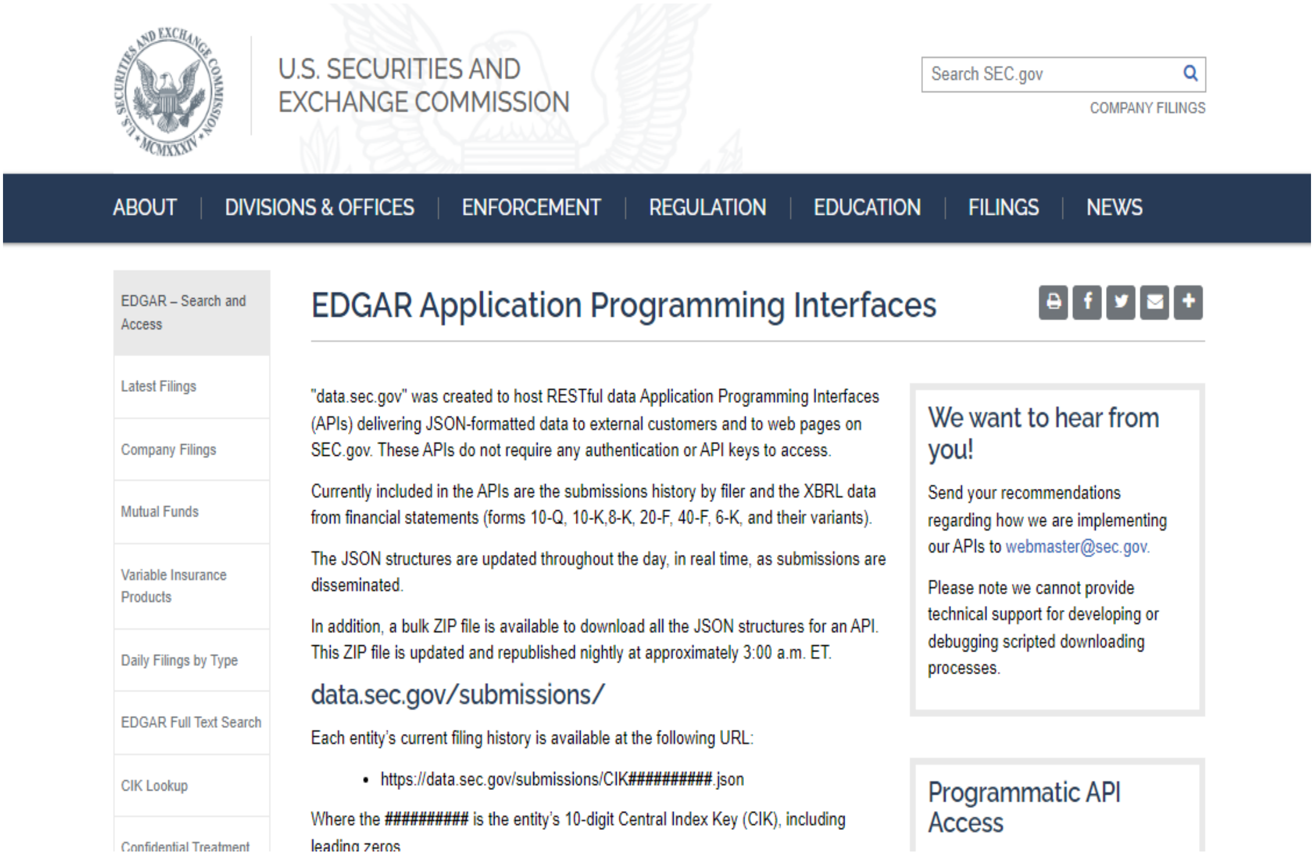
Data Migration and Transformation Tool for Amazon NOSQL Data Storages.

Project explained in below diagram:



Given link for downloading the zip file : https://www.sec.gov/edgar/sec-api-documentation

By following this link we can get this page.



EDGAR Application Programming Interfaces

"data.sec.gov" was created to host RESTful data Application Programming Interfaces (APIs) delivering JSON-formatted data to external customers and to web pages on SEC.gov. These APIs do not require any authentication or API keys to access.

Currently included in the APIs are the submissions history by filer and the XBRL data from financial statements (forms 10-Q, 10-K,8-K, 20-F, 40-F, 6-K, and their variants).

The JSON structures are updated throughout the day, in real time, as submissions are disseminated.

In addition, a bulk ZIP file is available to download all the JSON structures for an API. This ZIP file is updated and republished nightly at approximately 3:00 a.m. ET.

data.sec.gov/submissions/

Each entity's current filing history is available at the following URL:

- https://data.sec.gov/submissions/CIK##########.json

Where the ########## is the entity's 10-digit Central Index Key (CIK), including leading zeros

We want to hear from you!

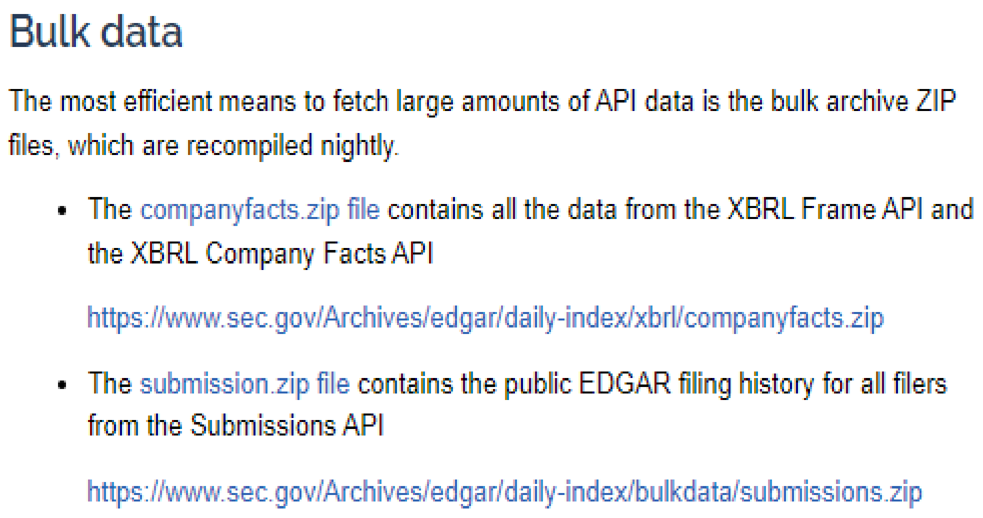
Send your recommendations regarding how we are implementing our APIs to webmaster@sec.gov.

Please note we cannot provide technical support for developing or debugging scripted downloading processes.

Programmatic API Access

After reaching this page we can see the option two options for downloading the zip files as shown in the below picture we can download any one for completing this project.

I have selected the submissions.zip because it contains the each organization fillings submissions.



Bulk data

The most efficient means to fetch large amounts of API data is the bulk archive ZIP files, which are recompiled nightly.

- The companyfacts.zip file contains all the data from the XBRL Frame API and the XBRL Company Facts API

  https://www.sec.gov/Archives/edgar/daily-index/xbrl/companyfacts.zip

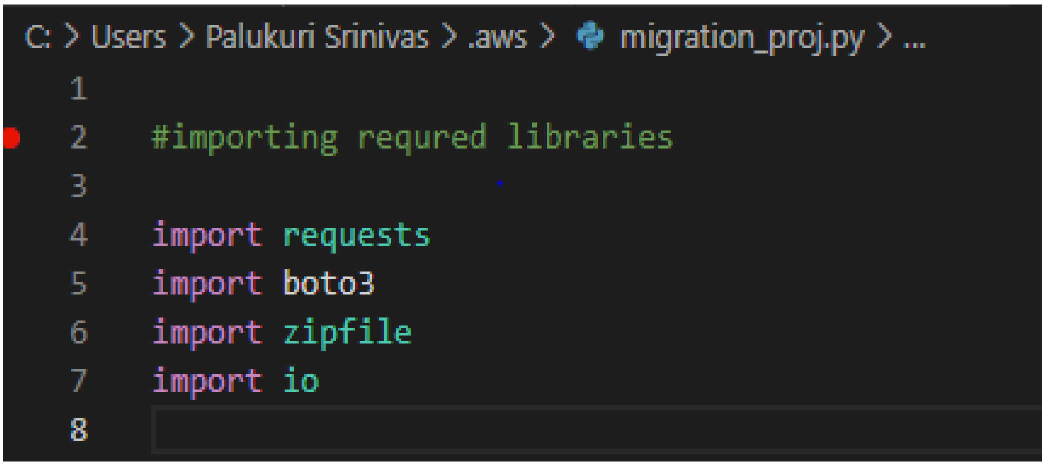
- The submission.zip file contains the public EDGAR filing history for all filers from the Submissions API

  https://www.sec.gov/Archives/edgar/daily-index/bulkdata/submissions.zip

Selected link for downloading: https://www.sec.gov/Archives/edgar/daily-index/bulkdata/submissions.zip

So now first part of the project: downloading the ZIPFILE through given link using requests library.

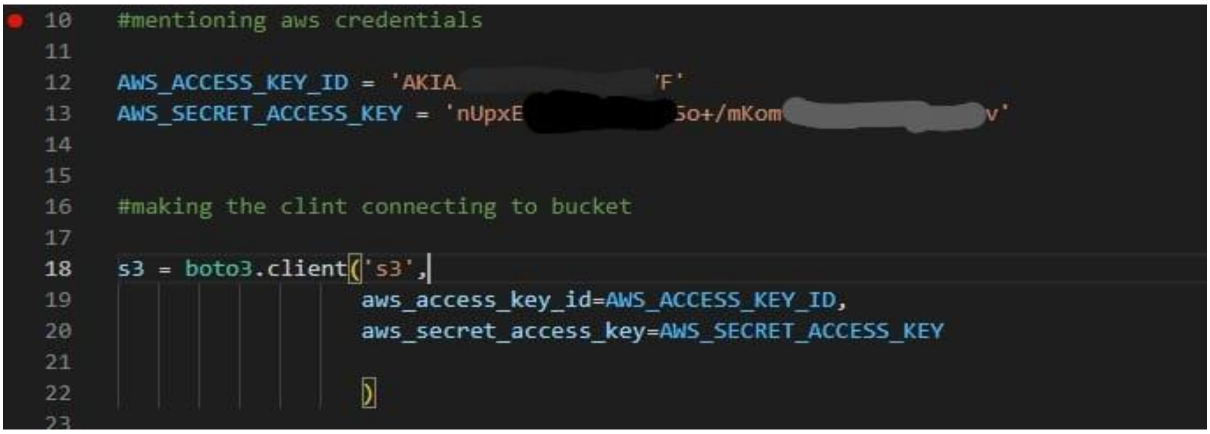
- Importing the required libraries
  1. requests: for downloading the zip file
  2. boto3: a python SDK for AWS
  3. zipfile: for unzipping the zipped file
  4. io : to do input and output operations



C: > Users > Palukuri Srinivas > .aws > migration_proj.py > ...

```

#importing requred libraries

import requests
import boto3
import zipfile
import io

```

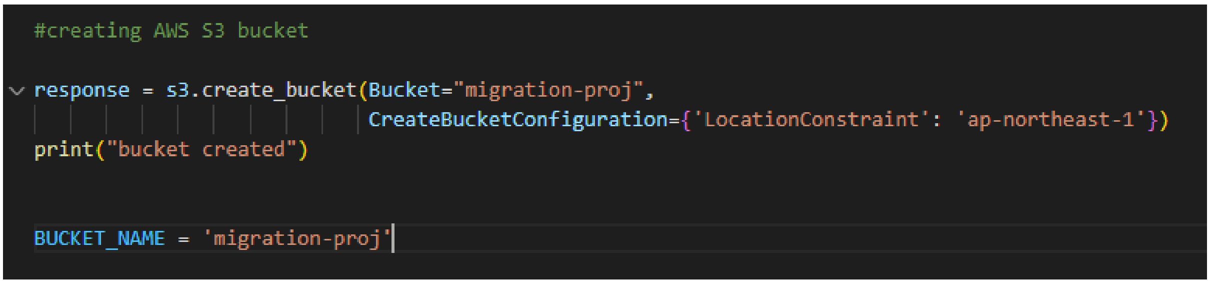
Motioning AWS credentials to connect the AWS resources to python ide (VS code)



```python
#mentioning aws credentials

AWS_ACCESS_KEY_ID = 'AKIA[redacted]F'
AWS_SECRET_ACCESS_KEY = 'nUpxE[redacted]5o+/mKom[redacted]v'


#making the clint connecting to bucket

s3 = boto3.client('s3',
                  aws_access_key_id=AWS_ACCESS_KEY_ID,
                  aws_secret_access_key=AWS_SECRET_ACCESS_KEY

                  )
```

Observe the above picture we have given the we have mentioned the **AWS_ACCESS_KEY_ID** and **AWS_SECRET_ACCESS_KEY** to connect the AWS resources.

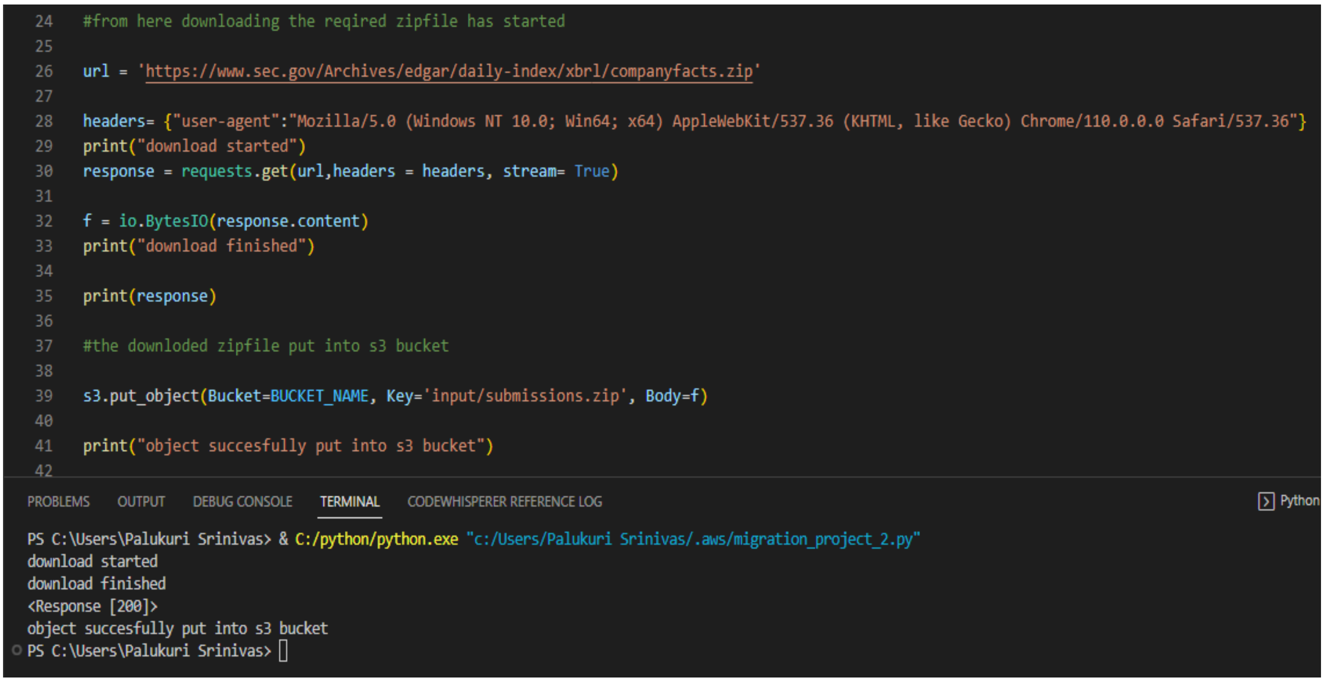
- Created the AWS S3 bucket for downloading the zip file.



```python
#creating AWS S3 bucket

response = s3.create_bucket(Bucket="migration-proj",
                            CreateBucketConfiguration={'LocationConstraint': 'ap-northeast-1'})
print("bucket created")


BUCKET_NAME = 'migration-proj'
```

Note: bucket name must be unique in your region so select the unique name to overcome the invalid location constraint error while executing.

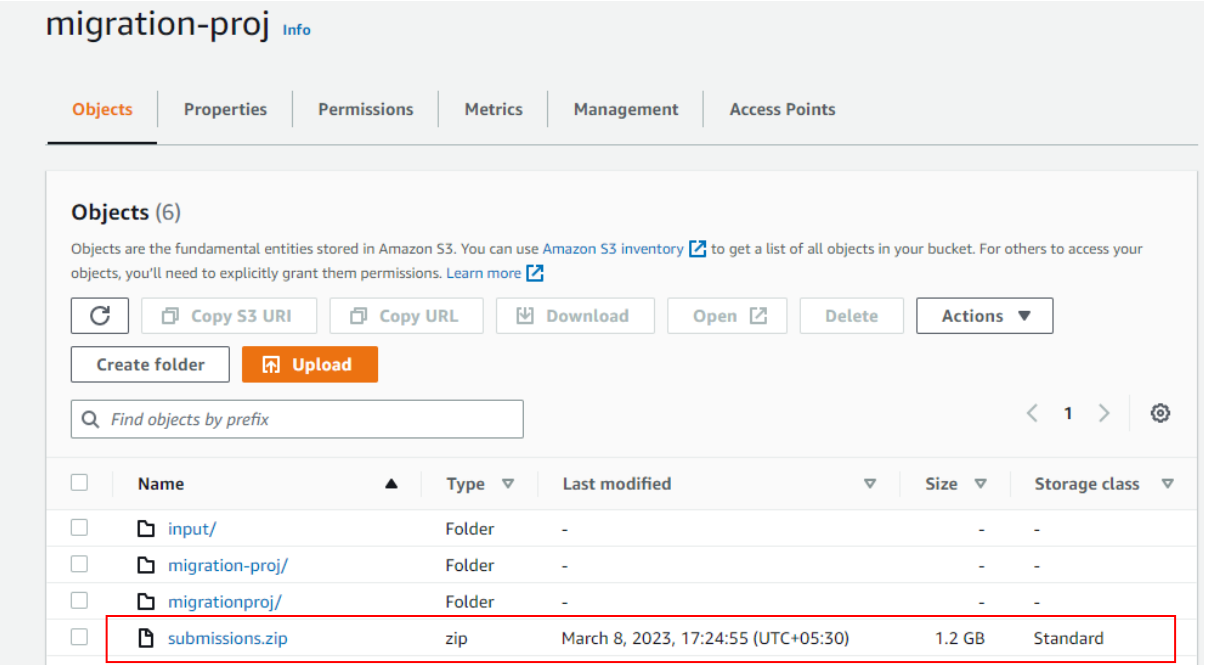
- Downloading the zip file into AWS S3 bucket using requests library and using boto3 putting the downloaded file in to S3 bucket.



```python
#from here downloading the reqired zipfile has started

url = 'https://www.sec.gov/Archives/edgar/daily-index/xbrl/companyfacts.zip'

headers= {"user-agent":"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.0.0 Safari/537.36"}
print("download started")
response = requests.get(url,headers = headers, stream= True)

f = io.BytesIO(response.content)
print("download finished")

print(response)

#the downloded zipfile put into s3 bucket

s3.put_object(Bucket=BUCKET_NAME, Key='input/submissions.zip', Body=f)

print("object succesfully put into s3 bucket")
```

```
PS C:\Users\Palukuri Srinivas> & C:/python/python.exe "c:/Users/Palukuri Srinivas/.aws/migration_project_2.py"
download started
download finished
<Response [200]>
object succesfully put into s3 bucket
PS C:\Users\Palukuri Srinivas> 
```

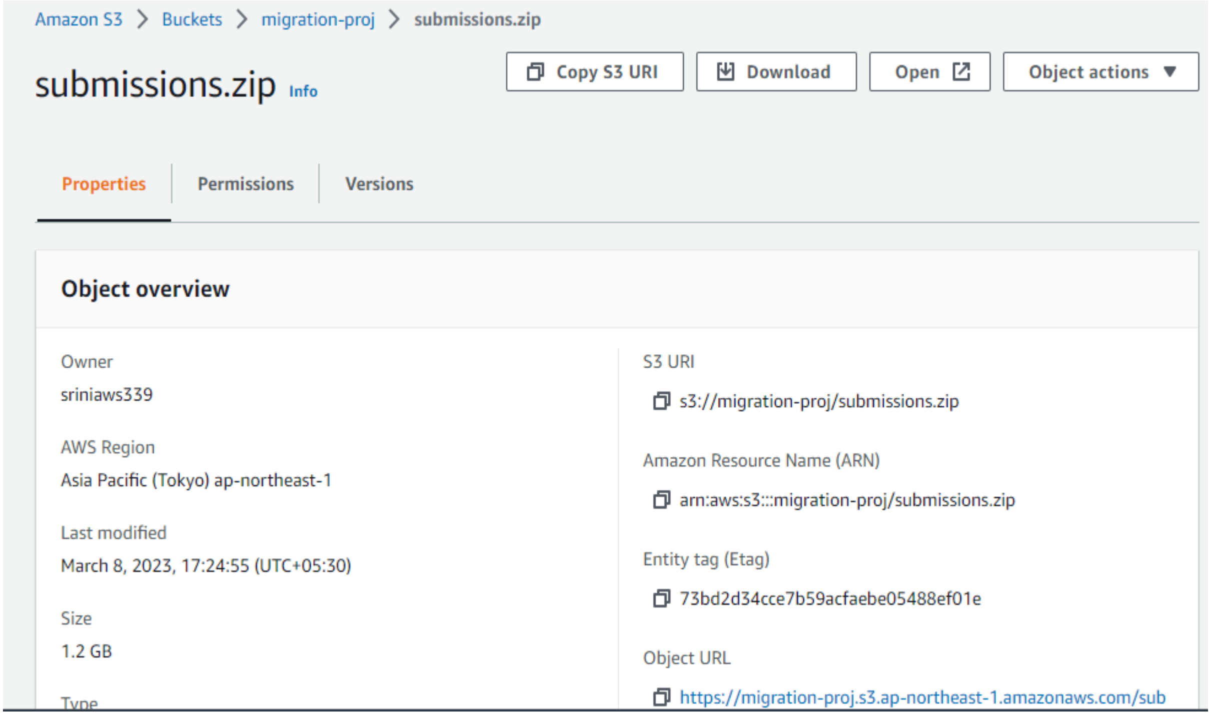
The mentioned zip file successfully downloaded and put into mentioned S3 bucket: result picture attached below



Objects (6)

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
□	input/	Folder	-	-	-
□	migration-proj/	Folder	-	-	-
□	migrationproj/	Folder	-	-	-
□	submissions.zip	zip	March 8, 2023, 17:24:55 (UTC+05:30)	1.2 GB	Standard



Object overview

Owner
sriniaws339

AWS Region
Asia Pacific (Tokyo) ap-northeast-1

Last modified
March 8, 2023, 17:24:55 (UTC+05:30)

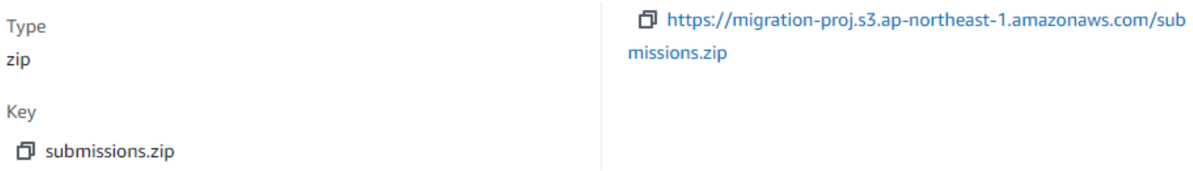
Size
1.2 GB

S3 URI
s3://migration-proj/submissions.zip

Amazon Resource Name (ARN)
arn:aws:s3:::migration-proj/submissions.zip

Entity tag (Etag)
73bd2d34cce7b59acfaebe05488ef01e

Object URL
https://migration-proj.s3.ap-northeast-1.amazonaws.com/sub



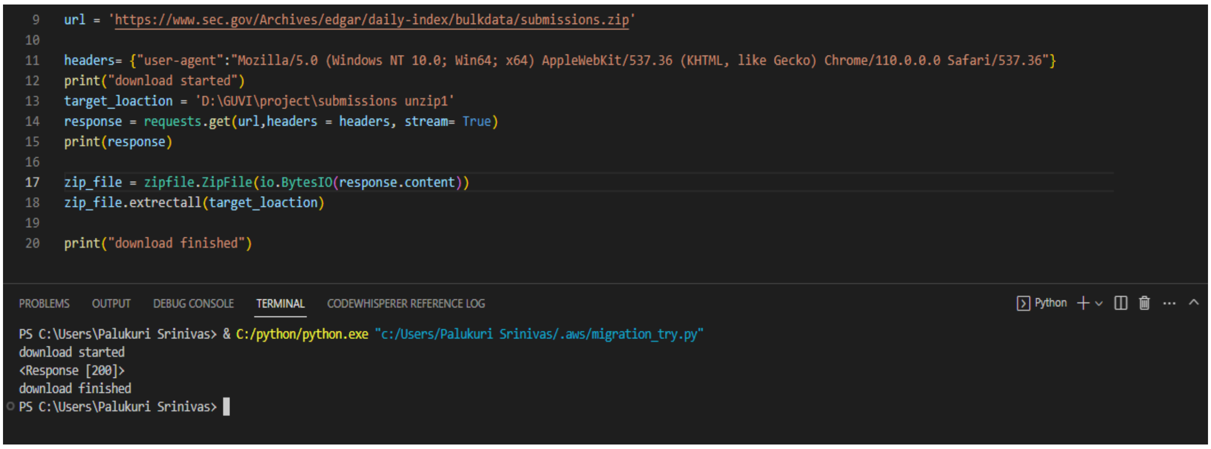
Type
zip

Key
submissions.zip

https://migration-proj.s3.ap-northeast-1.amazonaws.com/submissions.zip

Unzipped the files using the zipfile module:

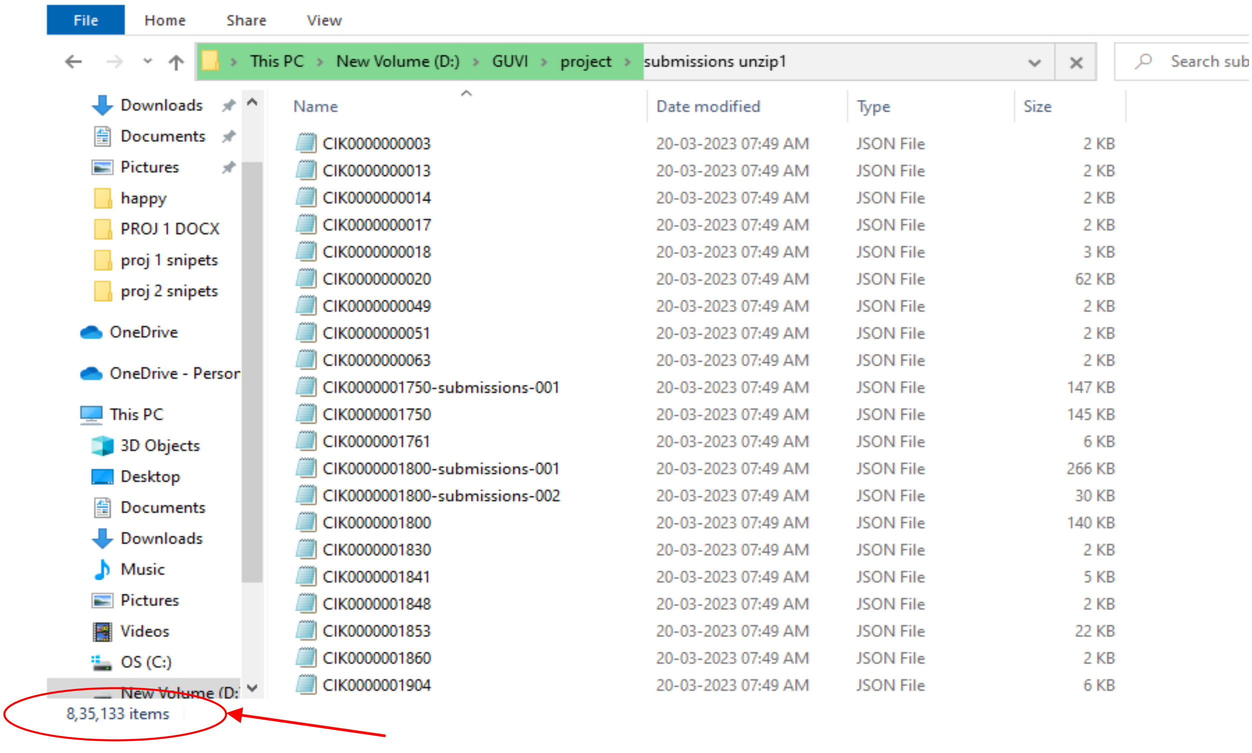
Tried to unzip the file using the zip file it took more than 8hrs of time but task has not completed so then I wanted to check what present in that zip file so downloaded to local storage and extracted using extractall



```
url = 'https://www.sec.gov/Archives/edgar/daily-index/bulkdata/submissions.zip'

headers= {"user-agent":"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.0.0 Safari/537.36"}
print("download started")
target_loaction = 'D:\GUVI\project\submissions unzip1'
response = requests.get(url,headers = headers, stream= True)
print(response)

zip_file = zipfile.ZipFile(io.BytesIO(response.content))
zip_file.extrectall(target_loaction)

print("download finished")
```

```
PS C:\Users\Palukuri Srinivas> & C:/python/python.exe "c:/Users/Palukuri Srinivas/.aws/migration_try.py"
download started
<Response [200]>
download finished
PS C:\Users\Palukuri Srinivas> 
```

Files downloaded and extracted to local storage.



Name	Date modified	Type	Size
CIK0000000003	20-03-2023 07:49 AM	JSON File	2 KB
CIK0000000013	20-03-2023 07:49 AM	JSON File	2 KB
CIK0000000014	20-03-2023 07:49 AM	JSON File	2 KB
CIK0000000017	20-03-2023 07:49 AM	JSON File	2 KB
CIK0000000018	20-03-2023 07:49 AM	JSON File	3 KB
CIK0000000020	20-03-2023 07:49 AM	JSON File	62 KB
CIK0000000049	20-03-2023 07:49 AM	JSON File	2 KB
CIK0000000051	20-03-2023 07:49 AM	JSON File	2 KB
CIK0000000063	20-03-2023 07:49 AM	JSON File	2 KB
CIK0000001750-submissions-001	20-03-2023 07:49 AM	JSON File	147 KB
CIK0000001750	20-03-2023 07:49 AM	JSON File	145 KB
CIK0000001761	20-03-2023 07:49 AM	JSON File	6 KB
CIK0000001800-submissions-001	20-03-2023 07:49 AM	JSON File	266 KB
CIK0000001800-submissions-002	20-03-2023 07:49 AM	JSON File	30 KB
CIK0000001800	20-03-2023 07:49 AM	JSON File	140 KB
CIK0000001830	20-03-2023 07:49 AM	JSON File	2 KB
CIK0000001841	20-03-2023 07:49 AM	JSON File	5 KB
CIK0000001848	20-03-2023 07:49 AM	JSON File	2 KB
CIK0000001853	20-03-2023 07:49 AM	JSON File	22 KB
CIK0000001860	20-03-2023 07:49 AM	JSON File	2 KB
CIK0000001904	20-03-2023 07:49 AM	JSON File	6 KB

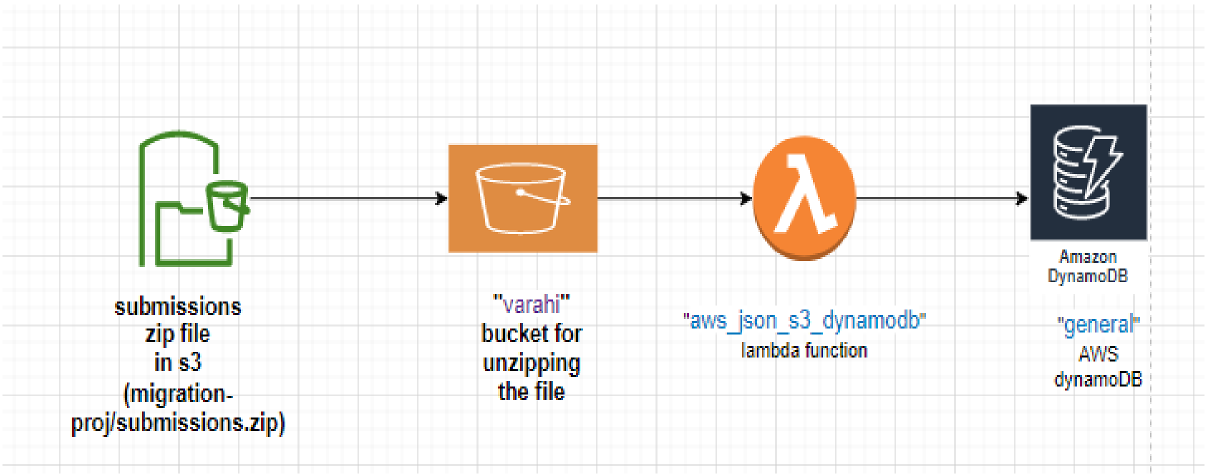
8,35,133 items

After downloading the file to my local storage then I can know that it containing 835133 (8lakhs of json files)

NOTE: Here we have an issue for unzipping, the zip file contains more than 8 lakh of files, its taking more time to unzip and im using AWS free tier account then I have asked the DE MENTOR during Project doubt sessions as per his suggestion to unzip 100 no of json files are enough to complete the project. But we should have the basic idea to unzip all the files in the zipfile. As per mentor suggestion completed the project by unzipping 150 no of json files.

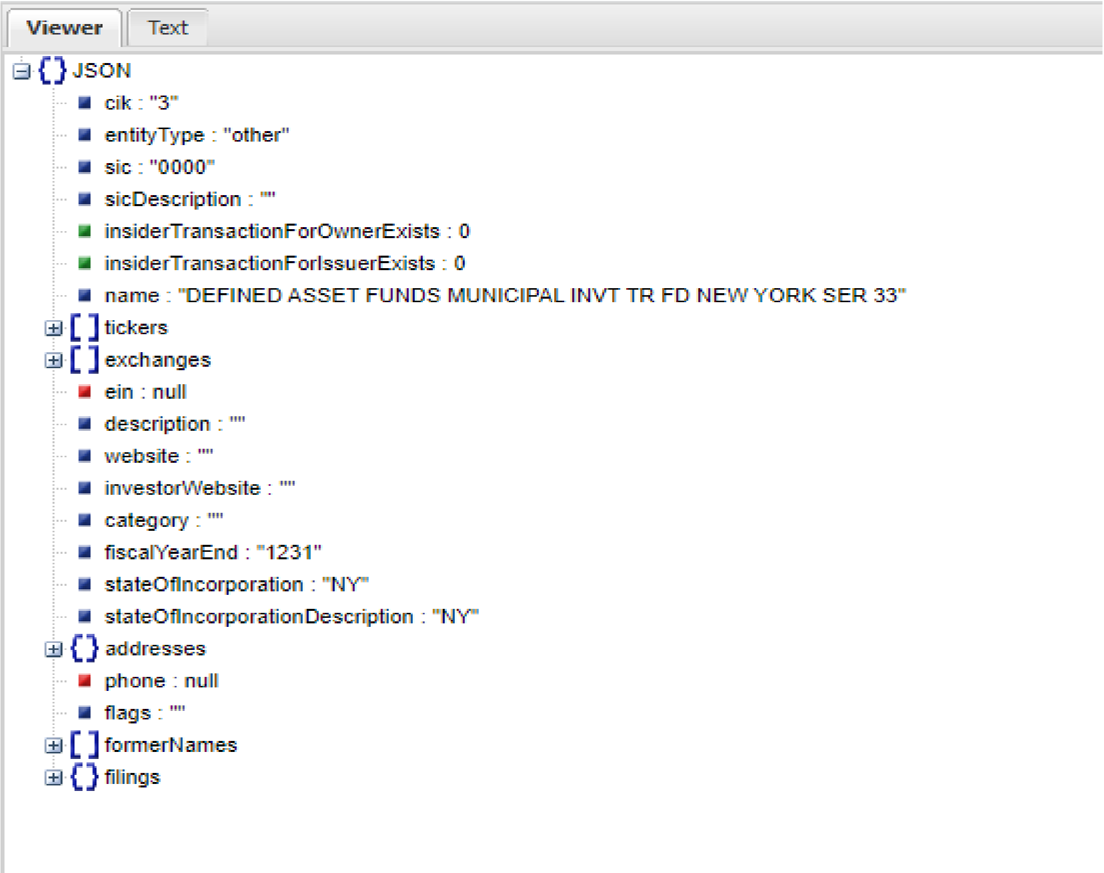
Our strategy to store the json files to dynamoDB from the zip file:

Creating the AWS lambda function which is triggered by S3 “varahi” bucket, then the lambda function will post the data to Dynamo DB “general” as a record.


submissions zip file in s3 (migration-proj/submissions.zip)
"varahi" bucket for unzipping the file
"aws_json_s3_dynamodb" lambda function
Amazon DynamoDB
"general" AWS dynamoDB

I have checked the content present in the json in json viewer.

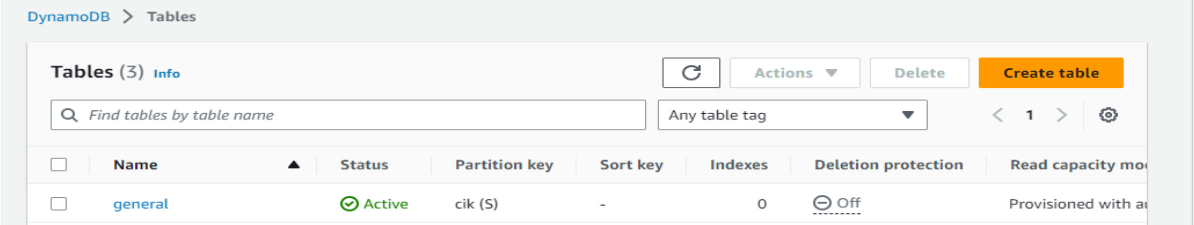
I got the result as shown in below picture and verified many json files then I can find the partition key to create the dynamoDB is “cik” data type is “string” I have observed the many files similar data type and partition key found .



```
⊟ { } JSON
      ■ cik : "3"
      ■ entityType : "other"
      ■ sic : "0000"
      ■ sicDescription : ""
      ■ insiderTransactionForOwnerExists : 0
      ■ insiderTransactionForIssuerExists : 0
      ■ name : "DEFINED ASSET FUNDS MUNICIPAL INVT TR FD NEW YORK SER 33"
   ⊞ [ ] tickers
   ⊞ [ ] exchanges
      ■ ein : null
      ■ description : ""
      ■ website : ""
      ■ investorWebsite : ""
      ■ category : ""
      ■ fiscalYearEnd : "1231"
      ■ stateOfIncorporation : "NY"
      ■ stateOfIncorporationDescription : "NY"
   ⊞ { } addresses
      ■ phone : null
      ■ flags : ""
   ⊞ [ ] formerNames
   ⊞ { } filings
```

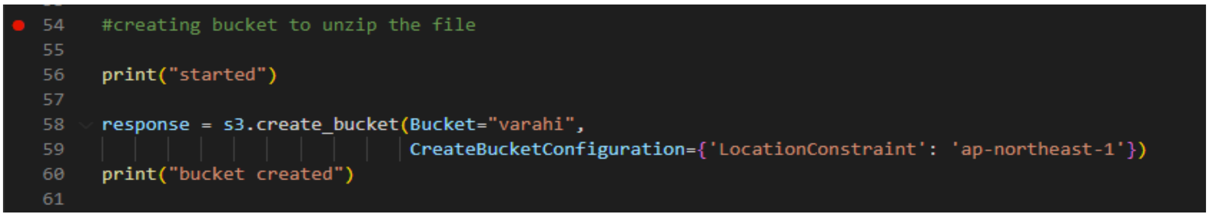
This is that json files containing.

- Creation of dynamo DB for saving the Json data which is present in unzipped files.
- DynamoDB name : general created with partition key as “cik” in “string” data type.



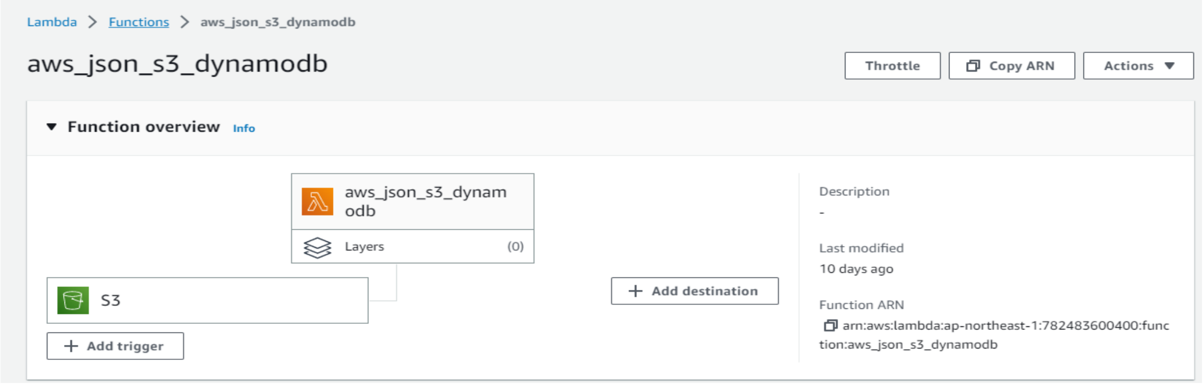
□	Name	Status	Partition key	Sort key	Indexes	Deletion protection	Read capacity mo
□	general	✓ Active	cik (S)	-	0	⊖ Off	Provisioned with a

- Creation of S3 bucket for unzipping, s3 bucket name: “varahi”



```
#creating bucket to unzip the file

print("started")

response = s3.create_bucket(Bucket="varahi",
                            CreateBucketConfiguration={'LocationConstraint': 'ap-northeast-1'})
print("bucket created")

```

- Creation of AWS lambda function with name “aws_json_s3_dynamodb”

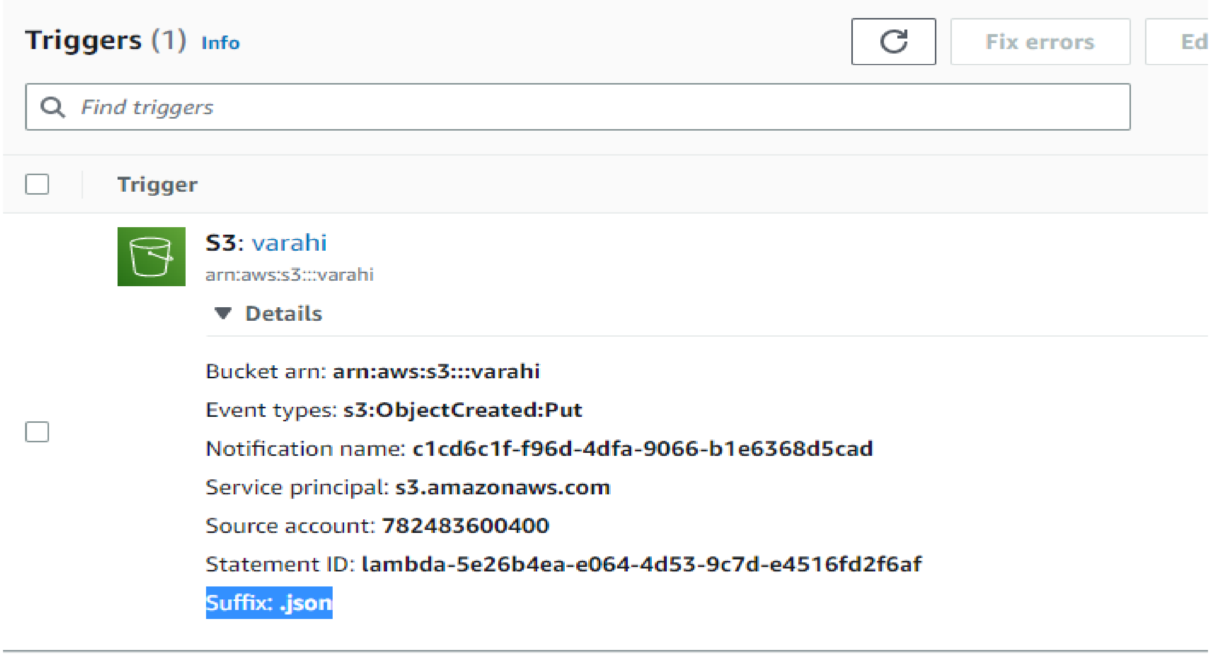


Description
-

Last modified
10 days ago

Function ARN
arn:aws:lambda:ap-northeast-1:782483600400:function:aws_json_s3_dynamodb

Added the trigger to lambda function as s3 bucket “varahi” , and Suffix: .json : so that any json file added to “varahi” bucket which have partition key as “cik” in “string” data type can be added to “general” dynamoDB table as an row.



S3: varahi
arn:aws:s3:::varahi

Details

Bucket arn: arn:aws:s3:::varahi

Event types: s3:ObjectCreated:Put

Notification name: c1cd6c1f-f96d-4dfa-9066-b1e6368d5cad

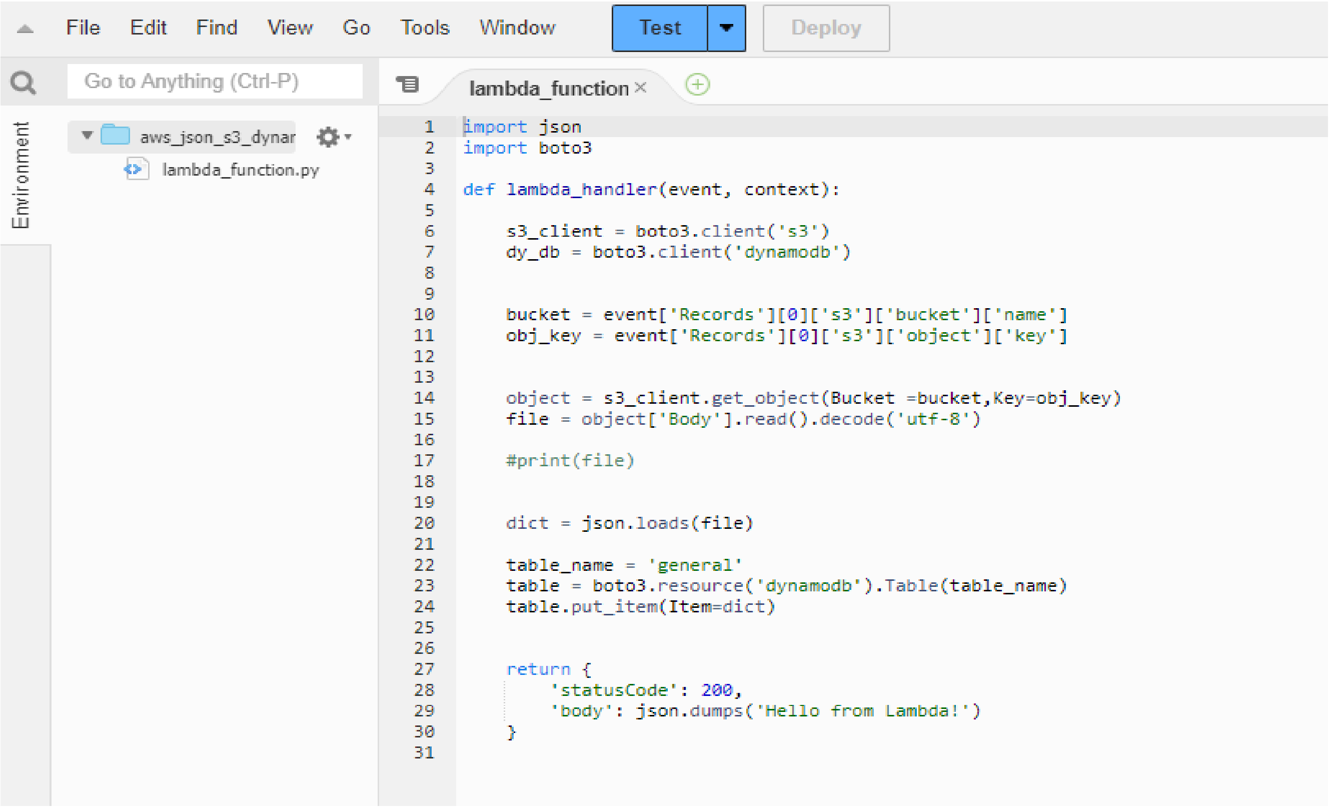
Service principal: s3.amazonaws.com

Source account: 782483600400

Statement ID: lambda-5e26b4ea-e064-4d53-9c7d-e4516fd2f6af

Suffix: .json

Used lambda code given below:



```python
import json
import boto3

def lambda_handler(event, context):

    s3_client = boto3.client('s3')
    dy_db = boto3.client('dynamodb')


    bucket = event['Records'][0]['s3']['bucket']['name']
    obj_key = event['Records'][0]['s3']['object']['key']


    object = s3_client.get_object(Bucket =bucket,Key=obj_key)
    file = object['Body'].read().decode('utf-8')

    #print(file)


    dict = json.loads(file)

    table_name = 'general'
    table = boto3.resource('dynamodb').Table(table_name)
    table.put_item(Item=dict)


    return {
        'statusCode': 200,
        'body': json.dumps('Hello from Lambda!')
    }

```

In the above code we can see :

1. Imported the required libraries.
2. Created S3 client.
3. Created the DynamoDB client.
4. Getting the bucket name and object key by pain event .
5. Retrieving object with bucket name and object key.
6. Reading the content to file variable.
7. Converting the file variable from json to dictionary (as per project statement).
8. Inserting the data to dynamo Db using Put item method.

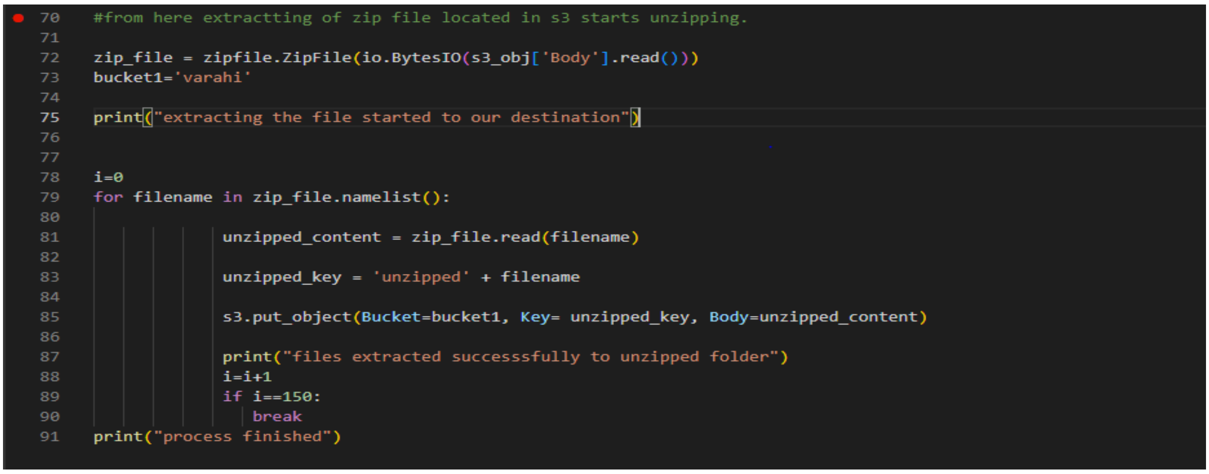
So to complete our process

1. DynamoDB table created
2. S3 bucket created for unzipping and triggering the lambda function
3. Lambda function also created with required code

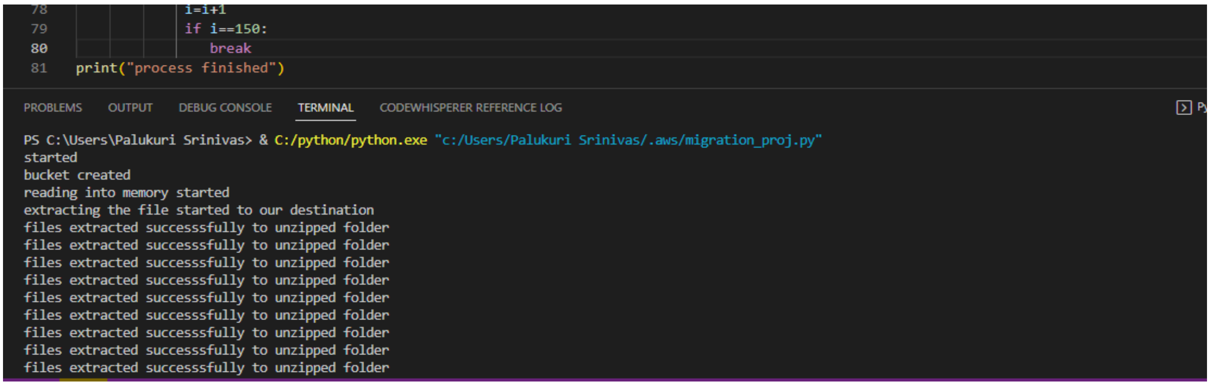
Now when any file added to “varahi” we know any Json file have partition key a “cik” with “string”data type it will be added to dynamo db as new record.

- Unzipping the “Submissions.zip” file to “varahi” setting limit as 150 nos, code snippet attached below

s



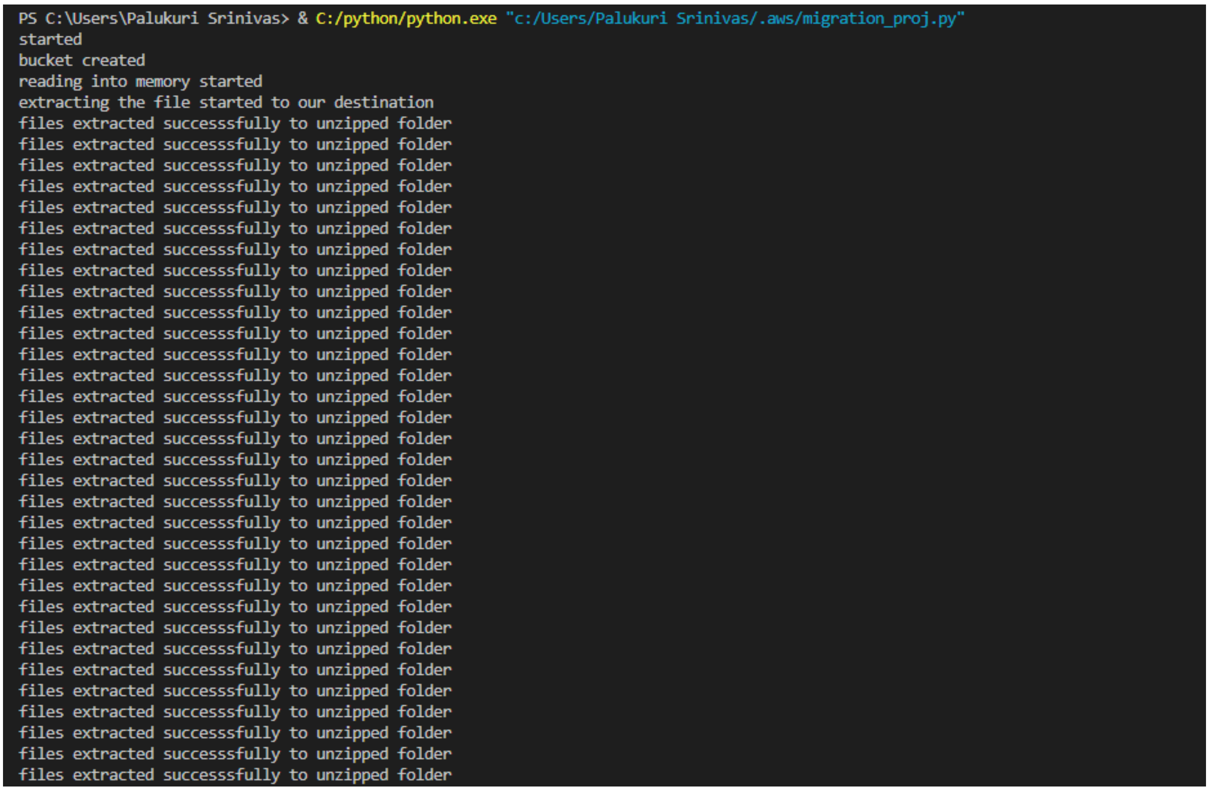
```
#from here extractting of zip file located in s3 starts unzipping.

zip_file = zipfile.ZipFile(io.BytesIO(s3_obj['Body'].read()))
bucket1='varahi'

print("extracting the file started to our destination")


i=0
for filename in zip_file.namelist():

            unzipped_content = zip_file.read(filename)

            unzipped_key = 'unzipped' + filename

            s3.put_object(Bucket=bucket1, Key= unzipped_key, Body=unzipped_content)

            print("files extracted successsfully to unzipped folder")
            i=i+1
            if i==150:
               break
print("process finished")
```



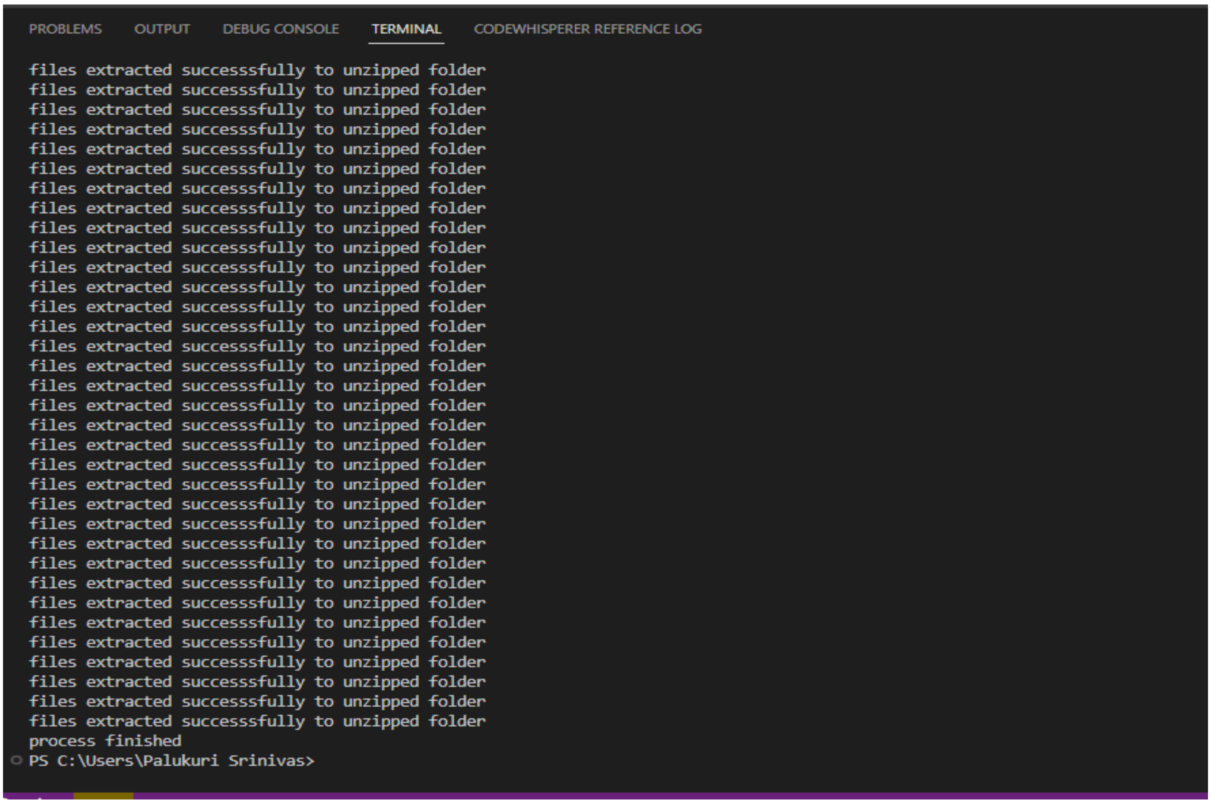
```
            i=i+1
            if i==150:
               break
print("process finished")
```

```
PS C:\Users\Palukuri Srinivas> & C:/python/python.exe "c:/Users/Palukuri Srinivas/.aws/migration_proj.py"
started
bucket created
reading into memory started
extracting the file started to our destination
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
```

Results attached below for executing the above code:

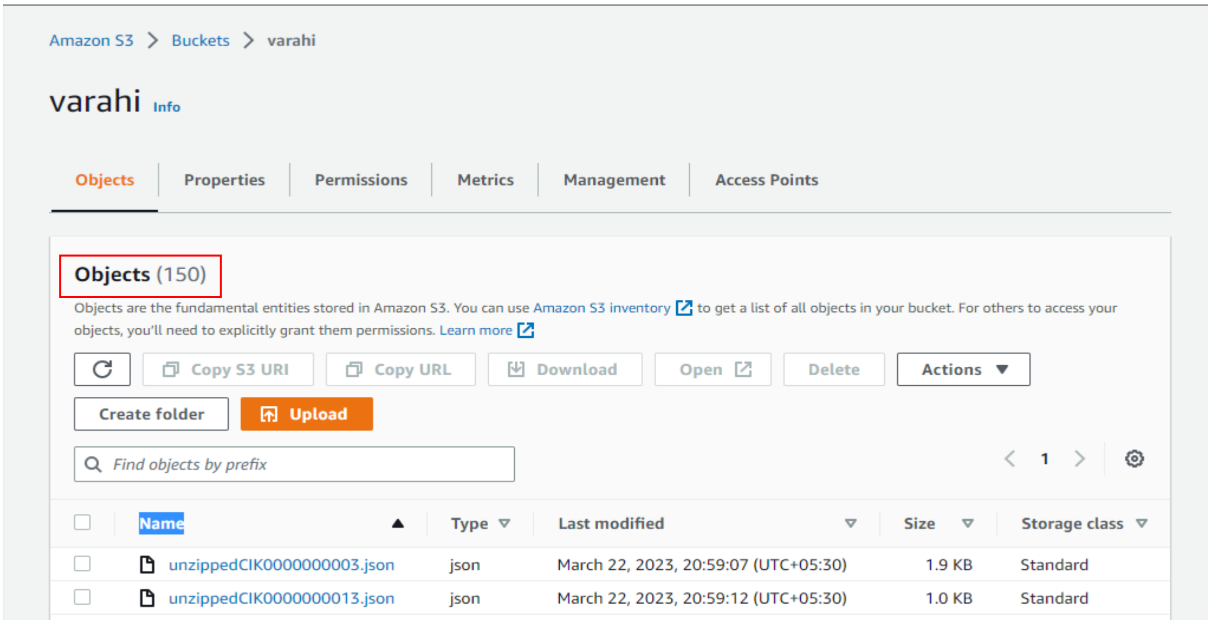


```
PS C:\Users\Palukuri Srinivas> & C:/python/python.exe "c:/Users/Palukuri Srinivas/.aws/migration_proj.py"
started
bucket created
reading into memory started
extracting the file started to our destination
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
```



```
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
files extracted successsfully to unzipped folder
process finished
PS C:\Users\Palukuri Srinivas>
```

Unzipped files to varahi bucket :

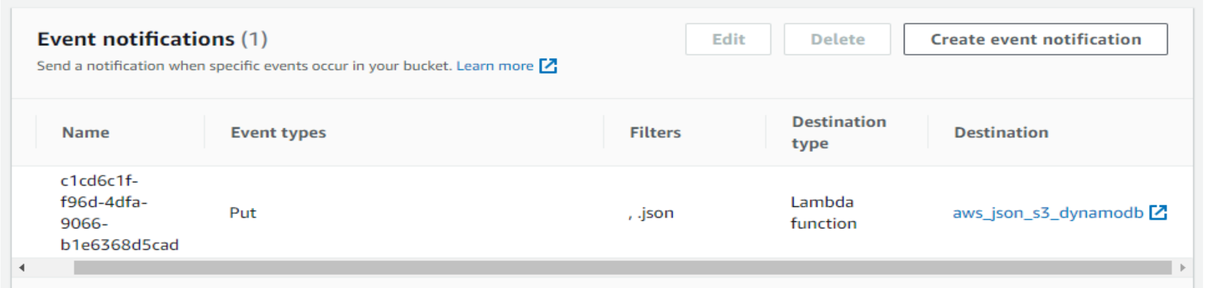


Objects (150)

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 ▽
□	unzippedCIK0000000003.json	json	March 22, 2023, 20:59:07 (UTC+05:30)	1.9 KB	Standard
□	unzippedCIK0000000013.json	json	March 22, 2023, 20:59:12 (UTC+05:30)	1.0 KB	Standard

We can see the destination as lambda function “aws_json_s3_dynamodb”

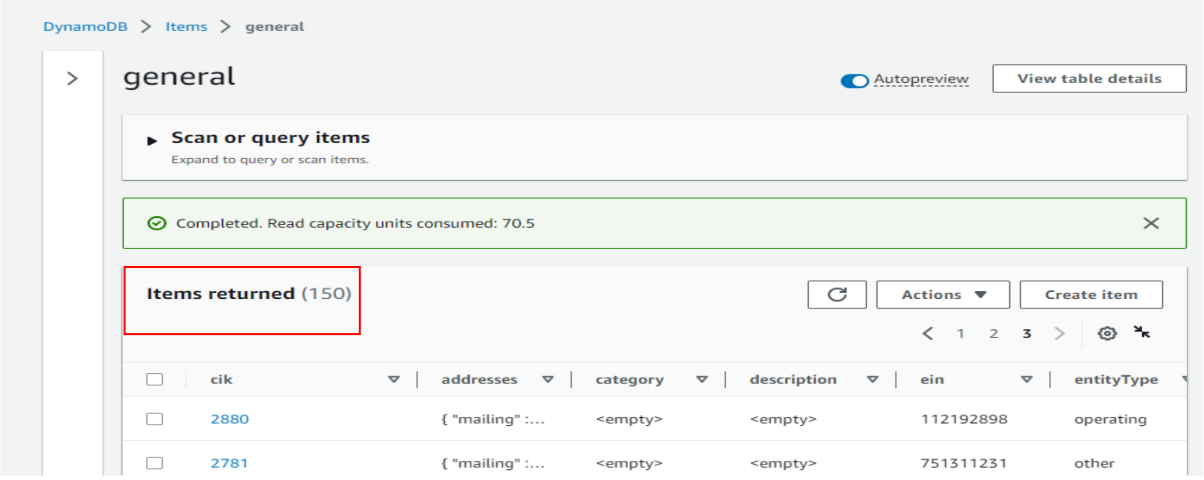


Event notifications (1)

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

Name	Event types	Filters	Destination type	Destination
c1cd6c1f-f96d-4dfa-9066-b1e6368d5cad	Put	, .json	Lambda function	aws_json_s3_dynamodb

Finally as a results the json files updated to dynamoDB as a records (Rows) snippets attached below. We can find the 150 rows updated to “general” dynamoDB.

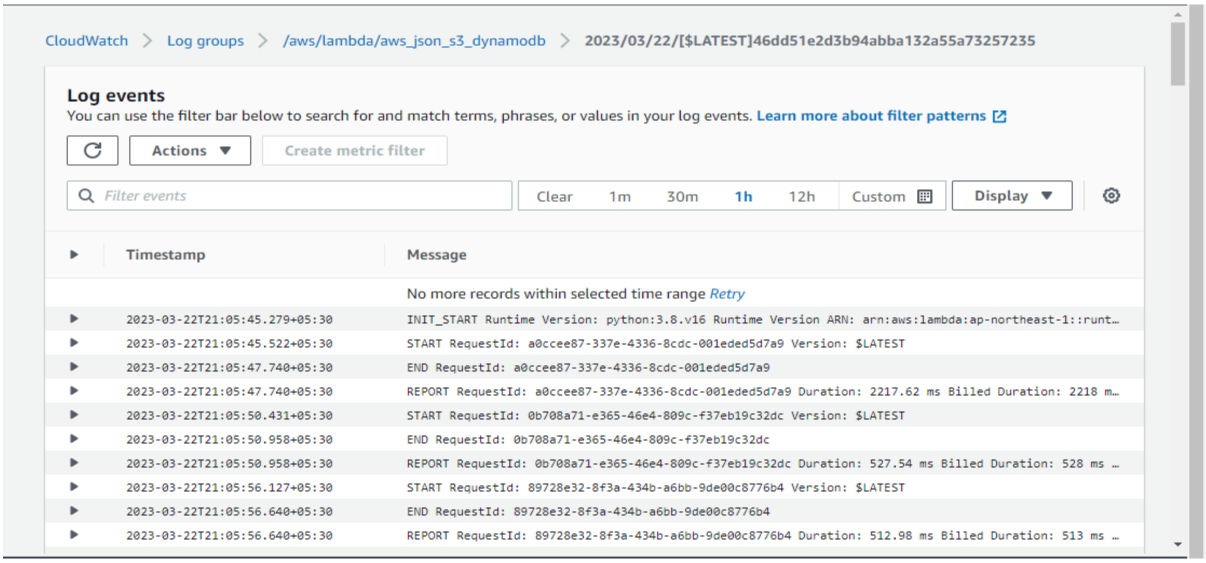


Completed. Read capacity units consumed: 70.5

Items returned (150)

	cik	addresses	category	description	ein	entityType
□	2880	{ "mailing" :...	<empty>	<empty>	112192898	operating
□	2781	{ "mailing" :...	<empty>	<empty>	751311231	other

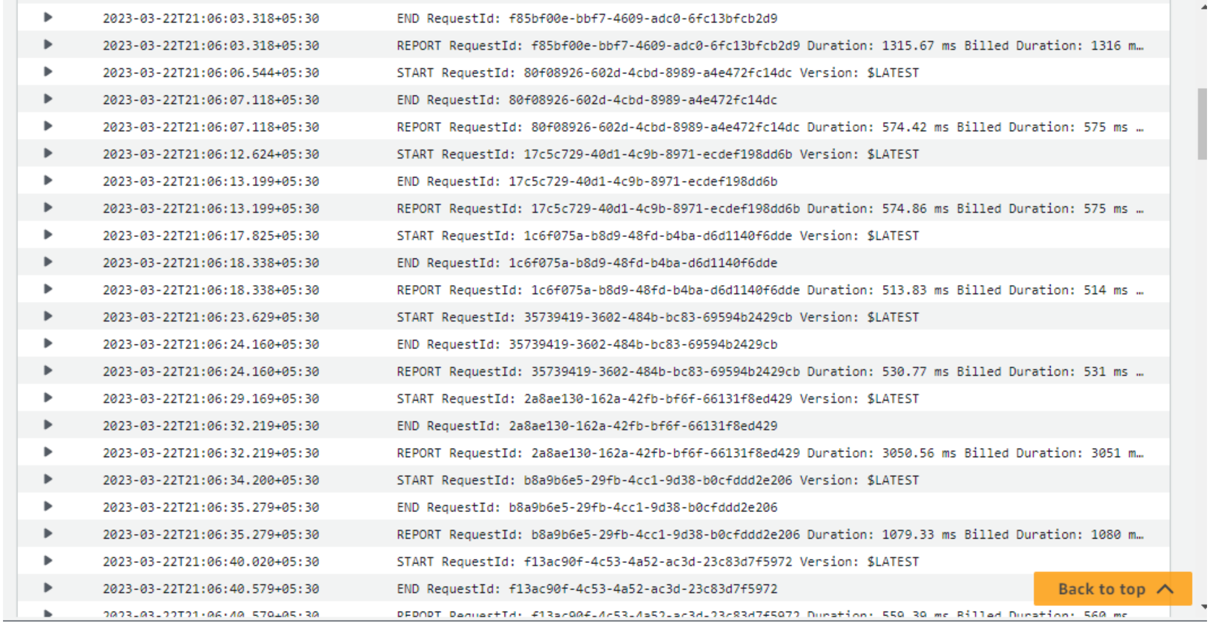
Cloud watch logs samples attached below as a snippet:



CloudWatch > Log groups > /aws/lambda/aws_json_s3_dynamodb > 2023/03/22/[$LATEST]46dd51e2d3b94abba132a55a73257235
Log events
Actions
Create metric filter
Filter events
Clear 1m 30m 1h 12h Custom
Display

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 more records within selected time range Retry
▶  2023-03-22T21:05:45.279+05:30  INIT_START Runtime Version: python:3.8.v16 Runtime Version ARN: arn:aws:lambda:ap-northeast-1::runt…
▶  2023-03-22T21:05:45.522+05:30  START RequestId: a0ccee87-337e-4336-8cdc-001eded5d7a9 Version: $LATEST
▶  2023-03-22T21:05:47.740+05:30  END RequestId: a0ccee87-337e-4336-8cdc-001eded5d7a9
▶  2023-03-22T21:05:47.740+05:30  REPORT RequestId: a0ccee87-337e-4336-8cdc-001eded5d7a9 Duration: 2217.62 ms Billed Duration: 2218 m…
▶  2023-03-22T21:05:50.431+05:30  START RequestId: 0b708a71-e365-46e4-809c-f37eb19c32dc Version: $LATEST
▶  2023-03-22T21:05:50.958+05:30  END RequestId: 0b708a71-e365-46e4-809c-f37eb19c32dc
▶  2023-03-22T21:05:50.958+05:30  REPORT RequestId: 0b708a71-e365-46e4-809c-f37eb19c32dc Duration: 527.54 ms Billed Duration: 528 ms …
▶  2023-03-22T21:05:56.127+05:30  START RequestId: 89728e32-8f3a-434b-a6bb-9de00c8776b4 Version: $LATEST
▶  2023-03-22T21:05:56.640+05:30  END RequestId: 89728e32-8f3a-434b-a6bb-9de00c8776b4
▶  2023-03-22T21:05:56.640+05:30  REPORT RequestId: 89728e32-8f3a-434b-a6bb-9de00c8776b4 Duration: 512.98 ms Billed Duration: 513 ms …
```



```
▶  2023-03-22T21:06:03.318+05:30  END RequestId: f85bf00e-bbf7-4609-adc0-6fc13bfcb2d9
▶  2023-03-22T21:06:03.318+05:30  REPORT RequestId: f85bf00e-bbf7-4609-adc0-6fc13bfcb2d9 Duration: 1315.67 ms Billed Duration: 1316 m…
▶  2023-03-22T21:06:06.544+05:30  START RequestId: 80f08926-602d-4cbd-8989-a4e472fc14dc Version: $LATEST
▶  2023-03-22T21:06:07.118+05:30  END RequestId: 80f08926-602d-4cbd-8989-a4e472fc14dc
▶  2023-03-22T21:06:07.118+05:30  REPORT RequestId: 80f08926-602d-4cbd-8989-a4e472fc14dc Duration: 574.42 ms Billed Duration: 575 ms …
▶  2023-03-22T21:06:12.624+05:30  START RequestId: 17c5c729-40d1-4c9b-8971-ecdef198dd6b Version: $LATEST
▶  2023-03-22T21:06:13.199+05:30  END RequestId: 17c5c729-40d1-4c9b-8971-ecdef198dd6b
▶  2023-03-22T21:06:13.199+05:30  REPORT RequestId: 17c5c729-40d1-4c9b-8971-ecdef198dd6b Duration: 574.86 ms Billed Duration: 575 ms …
▶  2023-03-22T21:06:17.825+05:30  START RequestId: 1c6f075a-b8d9-48fd-b4ba-d6d1140f6dde Version: $LATEST
▶  2023-03-22T21:06:18.338+05:30  END RequestId: 1c6f075a-b8d9-48fd-b4ba-d6d1140f6dde
▶  2023-03-22T21:06:18.338+05:30  REPORT RequestId: 1c6f075a-b8d9-48fd-b4ba-d6d1140f6dde Duration: 513.83 ms Billed Duration: 514 ms …
▶  2023-03-22T21:06:23.629+05:30  START RequestId: 35739419-3602-484b-bc83-69594b2429cb Version: $LATEST
▶  2023-03-22T21:06:24.160+05:30  END RequestId: 35739419-3602-484b-bc83-69594b2429cb
▶  2023-03-22T21:06:24.160+05:30  REPORT RequestId: 35739419-3602-484b-bc83-69594b2429cb Duration: 530.77 ms Billed Duration: 531 ms …
▶  2023-03-22T21:06:29.169+05:30  START RequestId: 2a8ae130-162a-42fb-bf6f-66131f8ed429 Version: $LATEST
▶  2023-03-22T21:06:32.219+05:30  END RequestId: 2a8ae130-162a-42fb-bf6f-66131f8ed429
▶  2023-03-22T21:06:32.219+05:30  REPORT RequestId: 2a8ae130-162a-42fb-bf6f-66131f8ed429 Duration: 3050.56 ms Billed Duration: 3051 m…
▶  2023-03-22T21:06:34.200+05:30  START RequestId: b8a9b6e5-29fb-4cc1-9d38-b0cfddd2e206 Version: $LATEST
▶  2023-03-22T21:06:35.279+05:30  END RequestId: b8a9b6e5-29fb-4cc1-9d38-b0cfddd2e206
▶  2023-03-22T21:06:35.279+05:30  REPORT RequestId: b8a9b6e5-29fb-4cc1-9d38-b0cfddd2e206 Duration: 1079.33 ms Billed Duration: 1080 m…
▶  2023-03-22T21:06:40.020+05:30  START RequestId: f13ac90f-4c53-4a52-ac3d-23c83d7f5972 Version: $LATEST
▶  2023-03-22T21:06:40.579+05:30  END RequestId: f13ac90f-4c53-4a52-ac3d-23c83d7f5972
▶  2023-03-22T21:06:40.579+05:30  REPORT RequestId: f13ac90f-4c53-4a52-ac3d-23c83d7f5972 Duration: 559.39 ms Billed Duration: 560 ms
```

Note: in the above unzipping code we have make limit for 150nos if we remove that each file will be unzipped and uploaded to dynamoDB records.

CONCLUSION:

- As per project statement we have downloaded the zip file using.
- Zip file is unzipped using zip file python module.
- Using boto3 module uploaded to S3 bucket.
- The json files converted to dictionary and updated to dynamoDB using boto3 SDK

Task completed.