Data Migration & Warehousing Using AWS & ATLAS

Problem Statement:

Extract the data from a zip file that is available at a URL and load it into Amazon S3 bucket, then transfer to DynamoDB & MongoDB ATLAS.

Approach:

Following steps are describing the approach:

- Downloaded the 'companyfacts' zip from the URL using 'request' library.
- Extracted json files using 'zipfile' module.
- Used boto3 client library to transfer file to AWS S3.
- Loaded the data from S3 to DynamoDB & MongoDB using AWS Lambda trigger.

Technologies:

Python 3.9, Google Colab, Requests, Zipfile, boto3, AWS S3, AWS Lambda, AWS DynamoDB, MongoDB Atlas.

Note:

- Json files sent to S3 are less than 400kb, since AWS DynamoDB data limit is 400kb.
- Google Colab is used since most libraries are pre-installed & downloaded zip file will be easily accessible.

Code Explanation:

- Open ‘Data Migration.ipynb’ in Google Colab & run the first cell.

Downloading Zip File to COLAB

```
import requests

def download_zip_file(url, filename):
  r = requests.get(url, stream=True, headers={'User-Agent': 'Mozilla/5.0'})
  if r.status_code == 200:
    with open(filename, 'wb') as f:
      r.raw.decode_content = True
      f.write(r.content)
      print('Zip File Downloading Completed')


url = 'https://www.sec.gov/Archives/edgar/daily-index/xbrl/companyfacts.zip'
# filename = url.split('/')[-1]
SourceData = "companyfacts.zip"
download_zip_file(url, SourceData)
```

```
Zip File Downloading Completed
```

Here the code downloads the zip file into colab space. Here ‘request’ module is used and we use ‘headers’ parameter in get () else it return status_code would be 403/404.

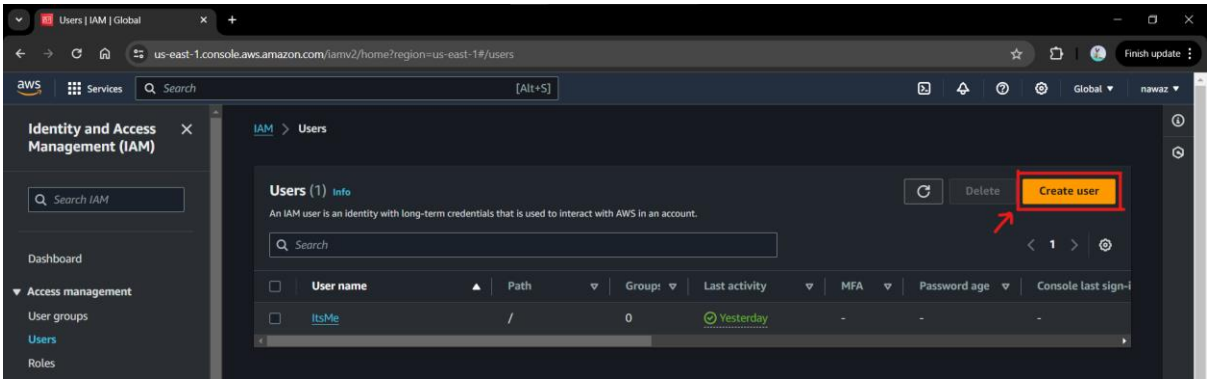
- Next the zipfile in extracted using ‘zipfile’ module into colab space.

Extracting Zip file

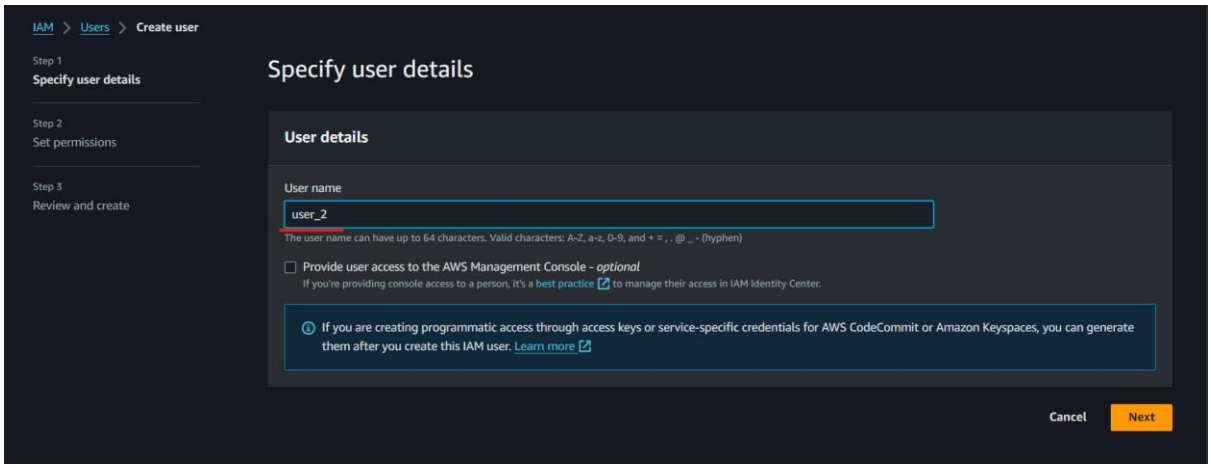
```
import zipfile

with zipfile.ZipFile('/content/companyfacts.zip', 'r') as zip_ref:
  zip_ref.extractall("Company Facts")
```

- Now files are extracted, next step would be to transfer files to AWS S3 bucket. But before that we need to have AWS account and access keys to access from colab.

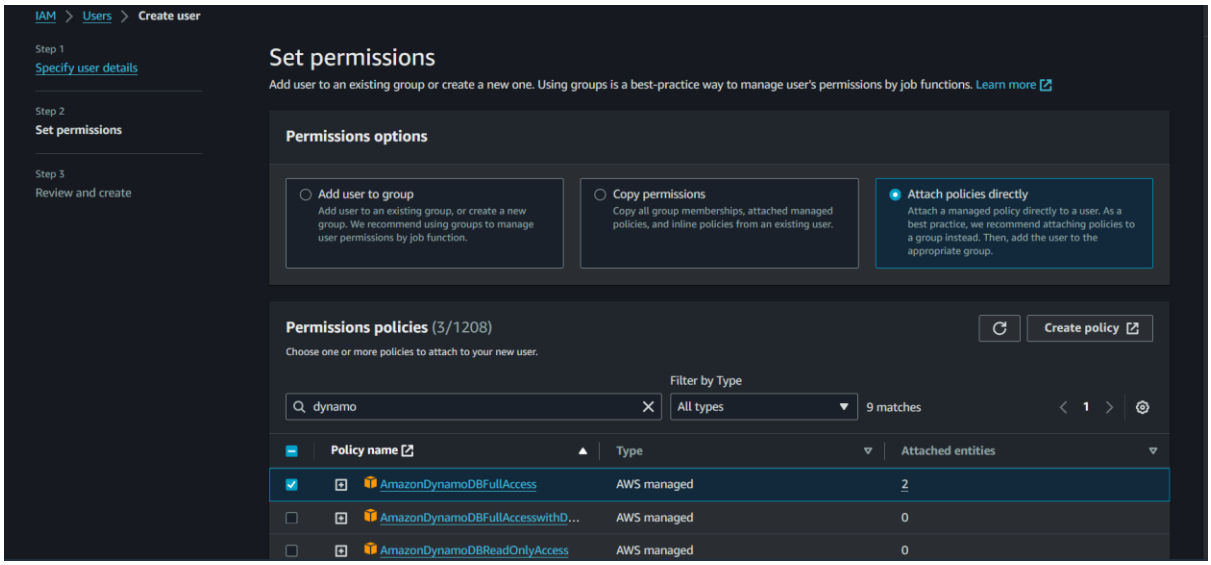
- Login into AWS Console & search User’s. Now click on ‘Create user’



- Give the username here.



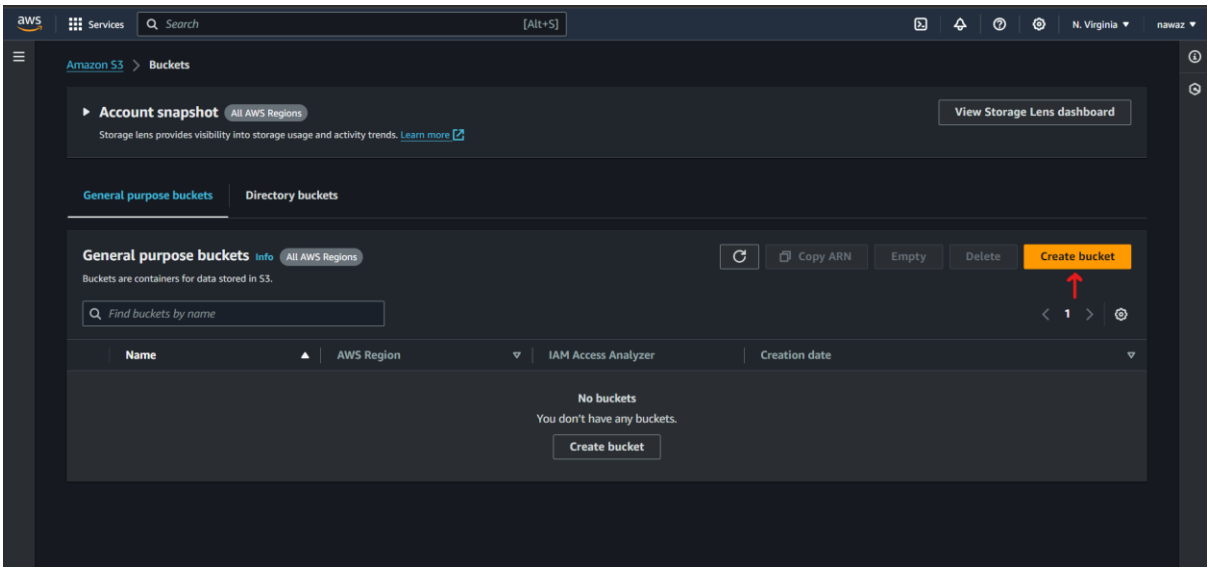
- Click on ‘Attach policies directly’ and select the resource permission’s you want to provide.
    1. J
    2. J
    3. J



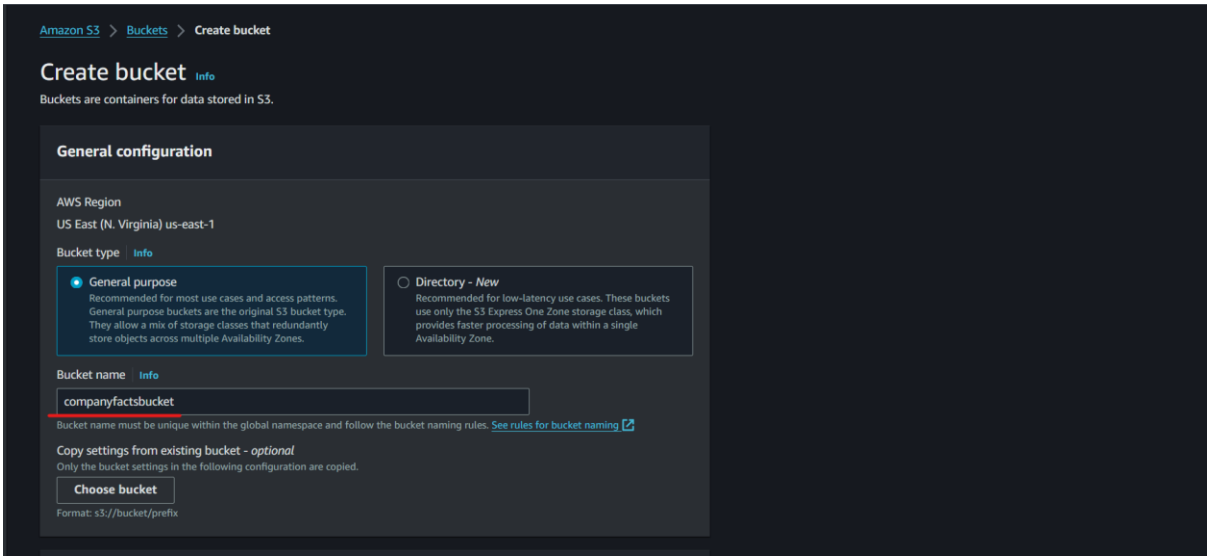
Click ‘Next’ & then click ‘Create User’, then User is created now.

- Now open the ‘user_2’ User and click on ‘Create access key’ to create access key.
Once its created, ‘Access key’ & ‘Secret access key’ is shown & even a file is shared which can be downloaded.

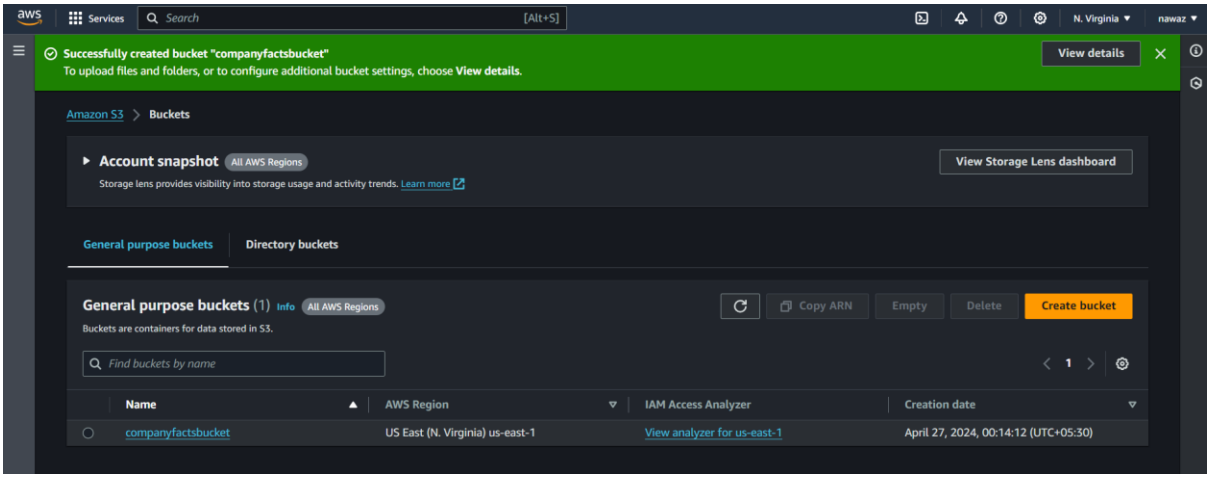
- Now we create a S3 bucket to store the json files. In AWS Console, Search S3 and click on ‘Create Bucket’.



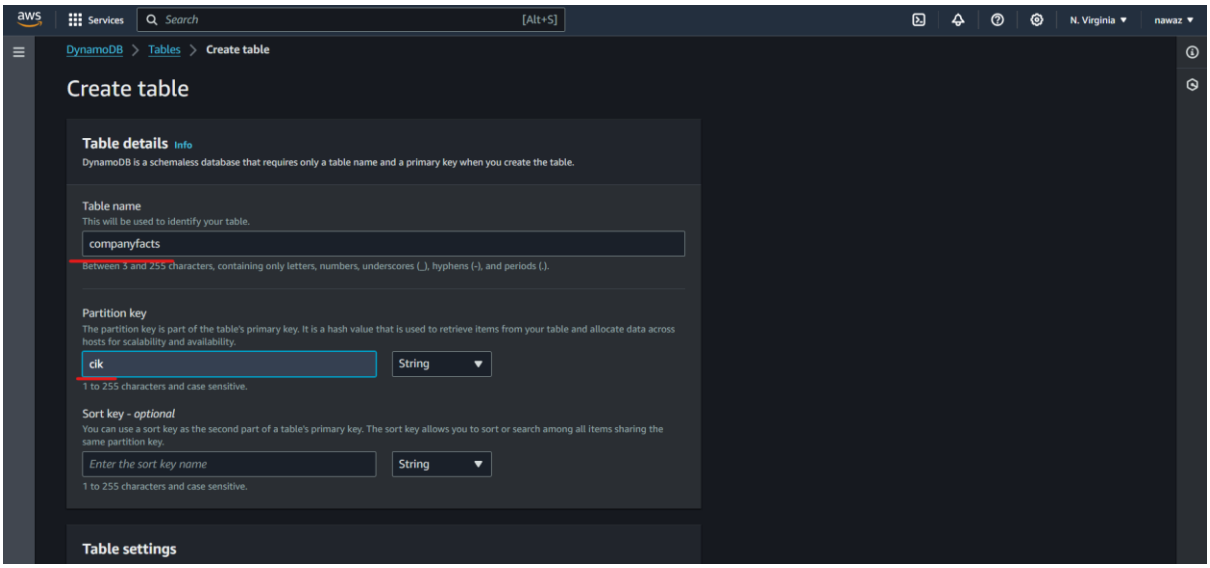
- Give the bucket name & scroll to bottom & click on ‘Create Bucket’.



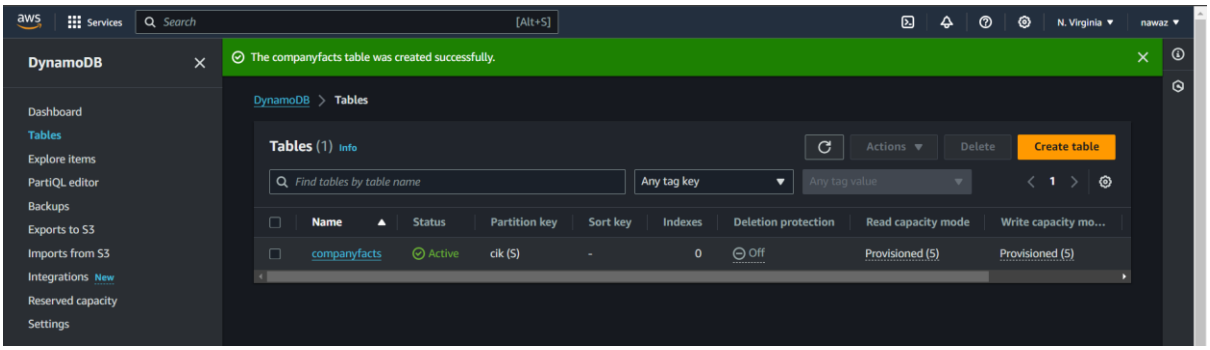
- Now bucket is created.



- Now before transferring file to bucket, we need to create a DynamoDB table & Mongo DB collection to store data, then we need to a function that will be automatically triggered whenever a file is inserted into S3.
- Now in AWS Console, search ‘DynamoDB’ click on it. Now click ‘Create Table’.



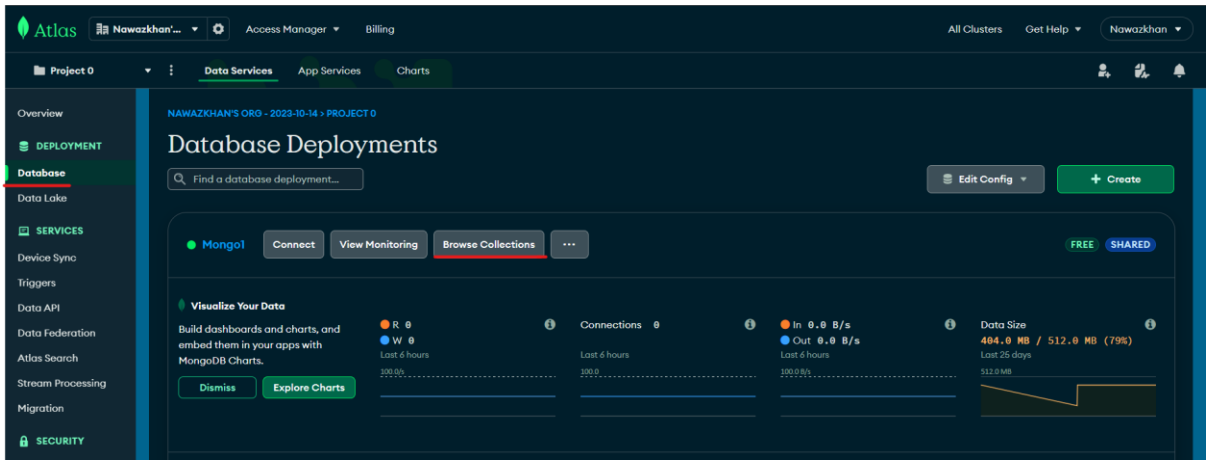
Here we give table name as ‘companyfacts’ and partition key as ‘cik’. ‘cik’ is a unique field in each json file, so this can be used as primary key for table. After this scroll to bottom & click on ‘Create Table’. Now table is created.



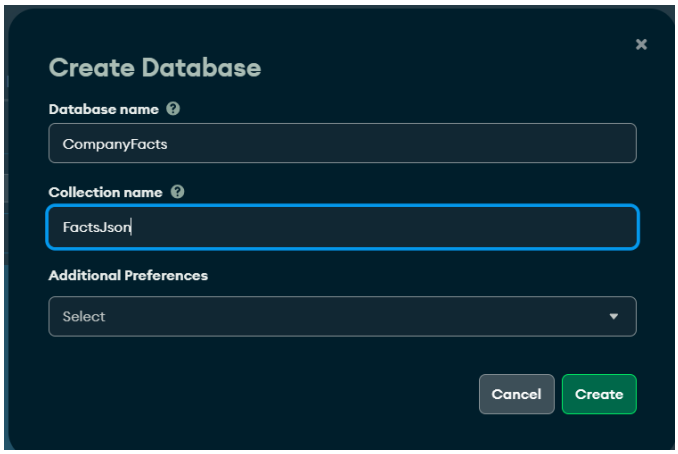
- Next step is to create a collection to store the Json file data in MongoDB Atlas.

- Login into MongoDB atlas, Please go through this link to create your first database. https://vinyldavyl.medium.com/how-to-create-a-database-in-mongodb-atlas-and-connect-your-database-to-your-application-step-by-9b63a2886b83

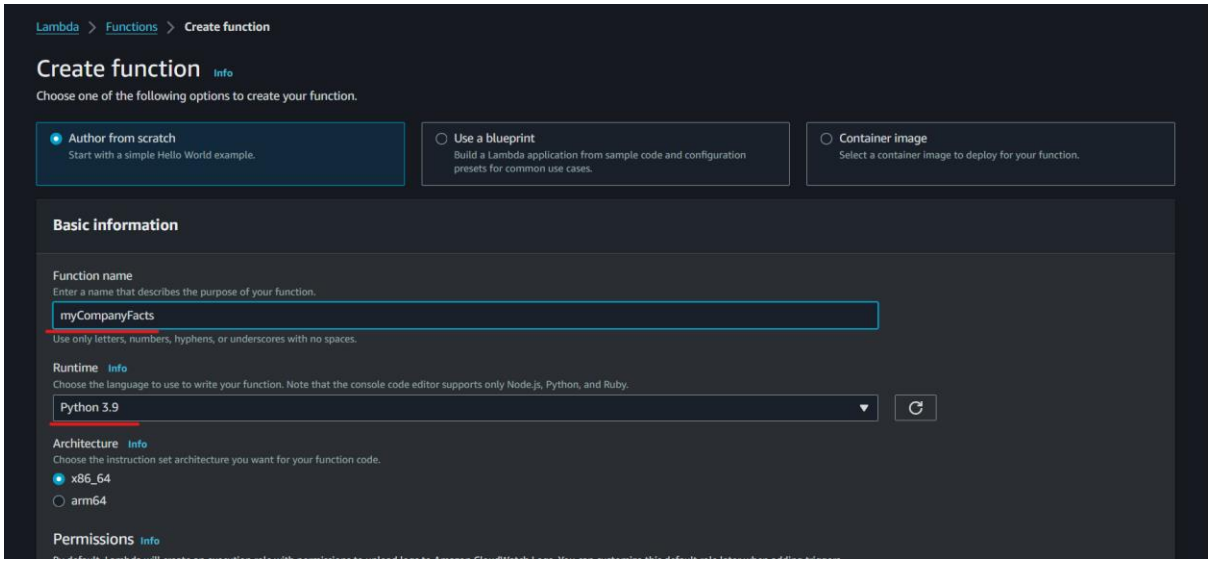
- Now click on ‘Databases’ in left lane & click on ‘Browse Collection’.



- Now click on ‘Create Database’ & Give Database name and Collection name and then click ‘Create’.

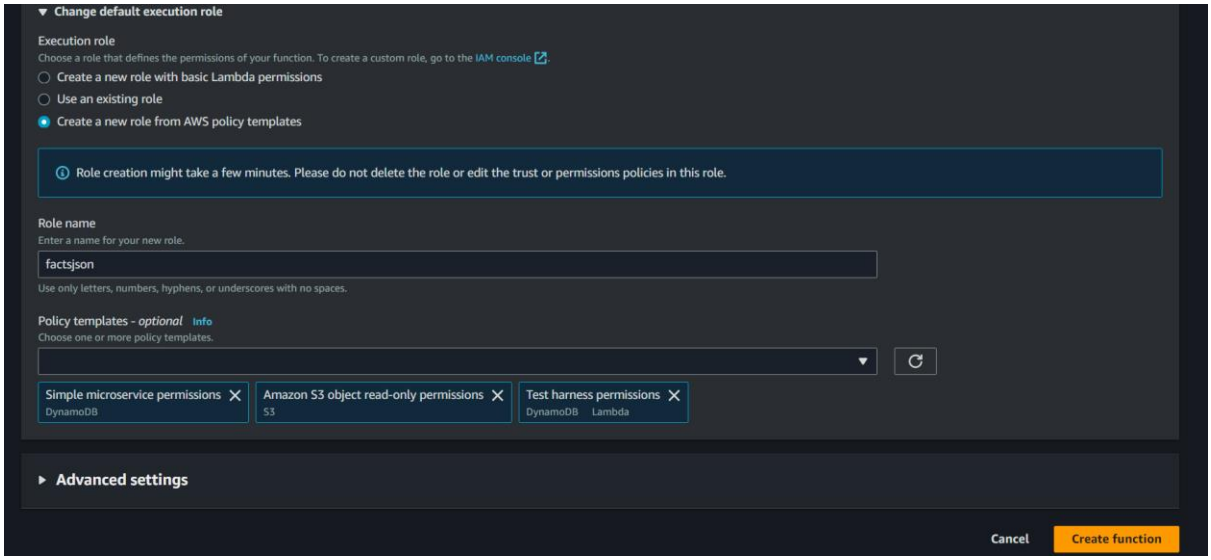


- Now create AWS Lambda function, which will get triggered automatically whenever a .json file is inserted into S3 bucket.
- In AWS Console, search Lambda and click on ‘Create Function’. Provide the function name and select the Runtime as python 3.9 [why 3.9 I will explain in upcoming step].

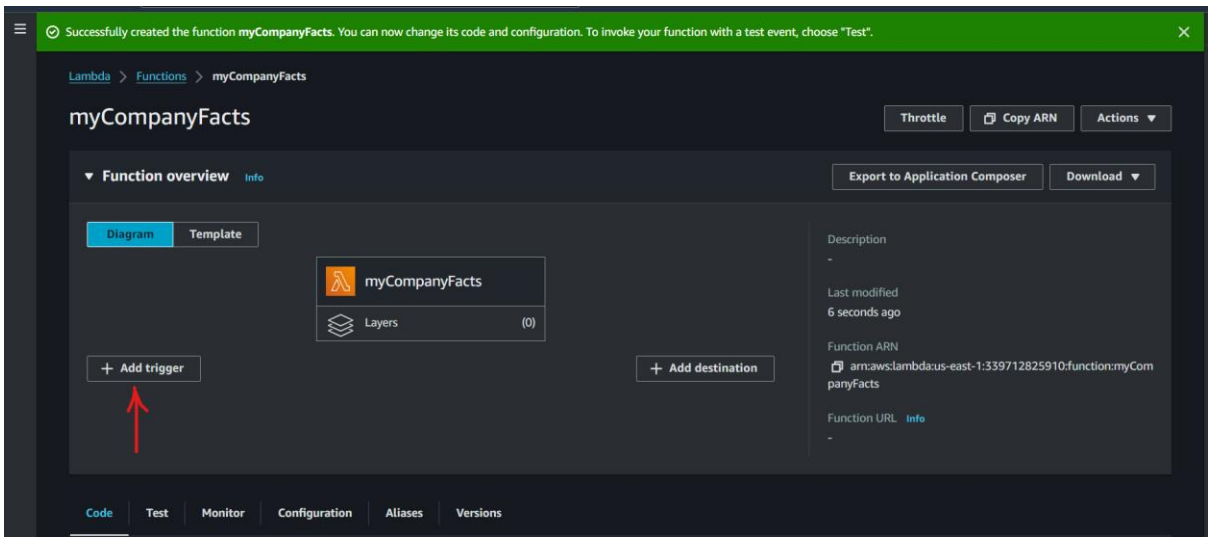


- Scroll below and expand ‘Change default execution role’, then select 'create a new role from AWS policy templates’. Here Provide ‘Role name’ and provide these 3 policies
    1. Amazon S3 object read-only permissions [S3]
    2. Test harness permissions [DynamoDB Lambda]
    3. Simple microservice permissions [DynamoDB]

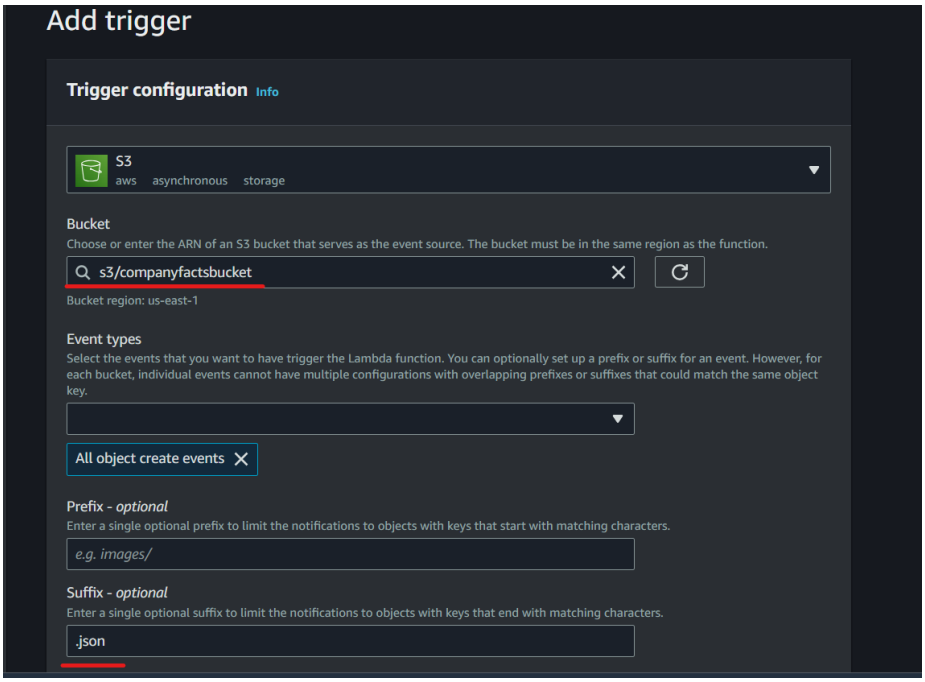
    Now click on ‘Create function’



- Click on ‘Add Trigger’ which will get triggered whenever file is added to S3.



- Now in drop down select S3, then below select the bucket name & provide suffix as ‘.json’, then check box the acknowledgement and click on ‘Add’.



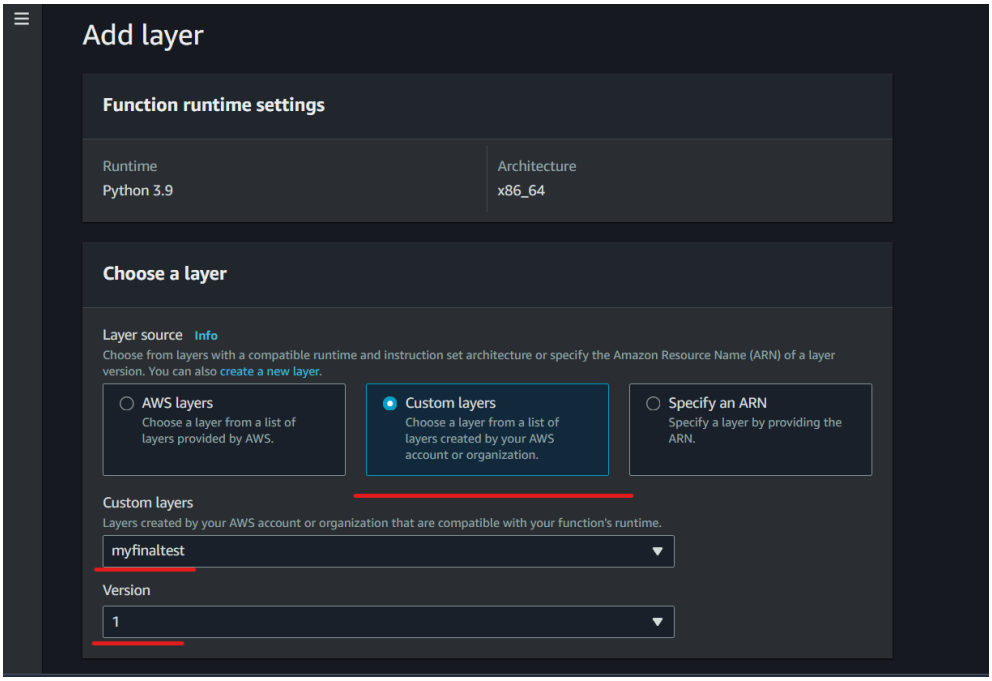
- Now open the code tab in Lambda & insert this code.

```
import os
from pymongo import MongoClient
import json
import urllib.parse
import boto3
import io
from decimal import Decimal

s3Client = boto3.client('s3')

connect = MongoClient(host = os.environ.get("ATLAS_URI"))

dydb = boto3.resource(service_name = 'dynamodb')


def lambda_handler(event, context):
    #Get bucket and file name
    bucket = event['Records'][0]['s3']['bucket']['name']
    key = event['Records'][0]['s3']['object']['key']

    print(bucket)
    print(key)
```

```
#Creates DB & Collection if not exist
db = connect[CompanyFacts]
col = db[FactsJson]


#Get our object
response = s3Client.get_object(Bucket=bucket,Key=key)

#Fetching Data
data = response['Body'].read().decode('utf-8')
print(data)

print(type(data))

data = eval(data)

print(type(data))

#Inserting it into DynamoDB
table = dydb.Table('companyfacts')
jsondata = json.loads(json.dumps(data), parse_float=Decimal)

print(type(jsondata))

table.put_item(Item = jsondata)


#Passing Document to MongoDB
result = col.insert_one(data)

if result.inserted_id:
   return "Inserted"
else:
   return "Not Inserted"
```

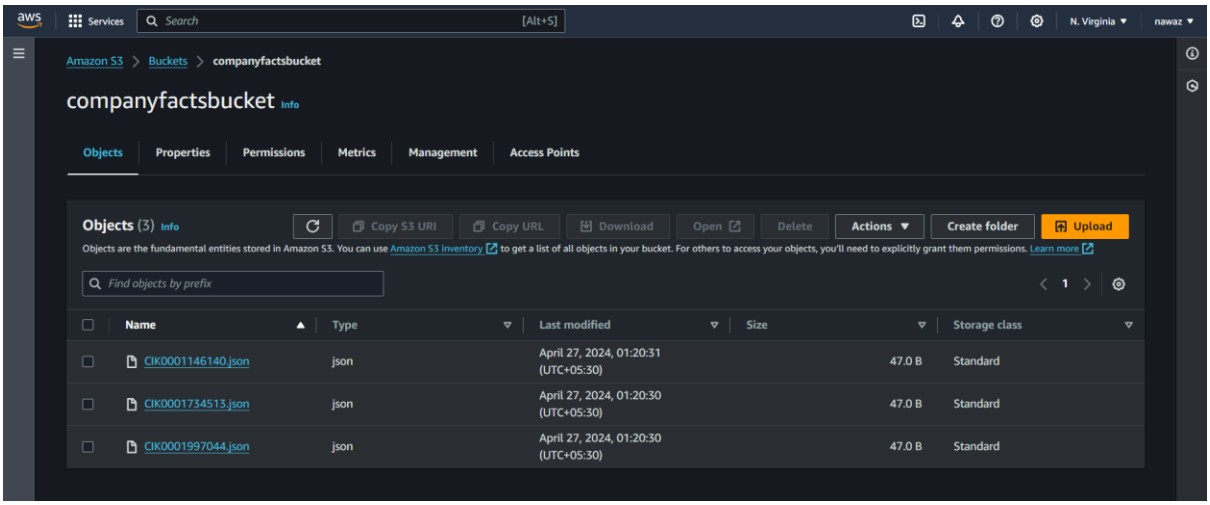
- Here we can see we have used ‘pymongo’ library, which is not present in lambda environment. We need to install this in a layer. To do this follow this URL & install pymongo. https://www.linkedin.com/pulse/add-external-python-libraries-aws-lambda-using-layers-gabe-olokun

- Now go below in your lambda function & click on ‘Add layer’. Choose ‘Custom Layer’ & select the layer and version and then click on ‘Add’.

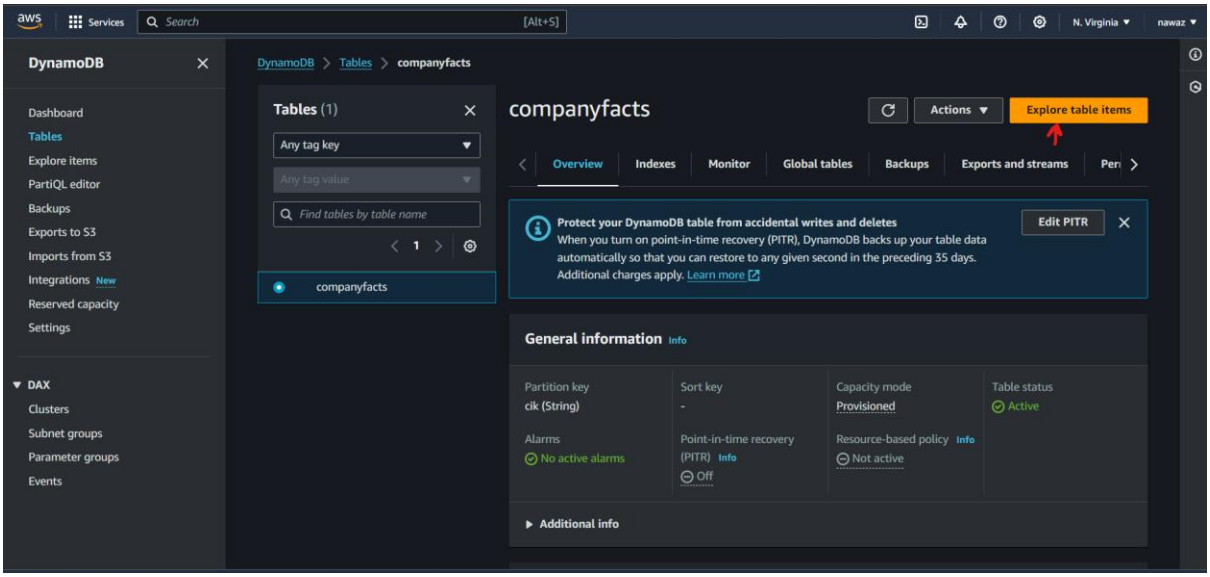


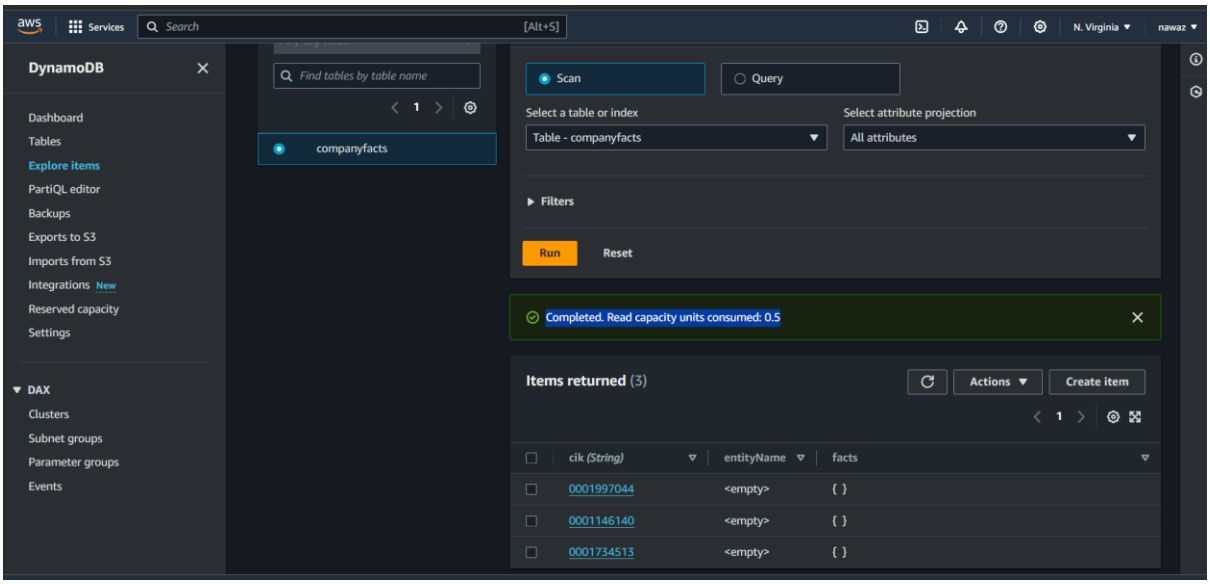
- Now in lambda, to go configuration->Environment Variable. Then click on ‘Edit’. Now give key as ‘ATLAS_URI’ and value as the URL which we used to connect to mongo DB.
- Need to add this step

- Now deploy the code

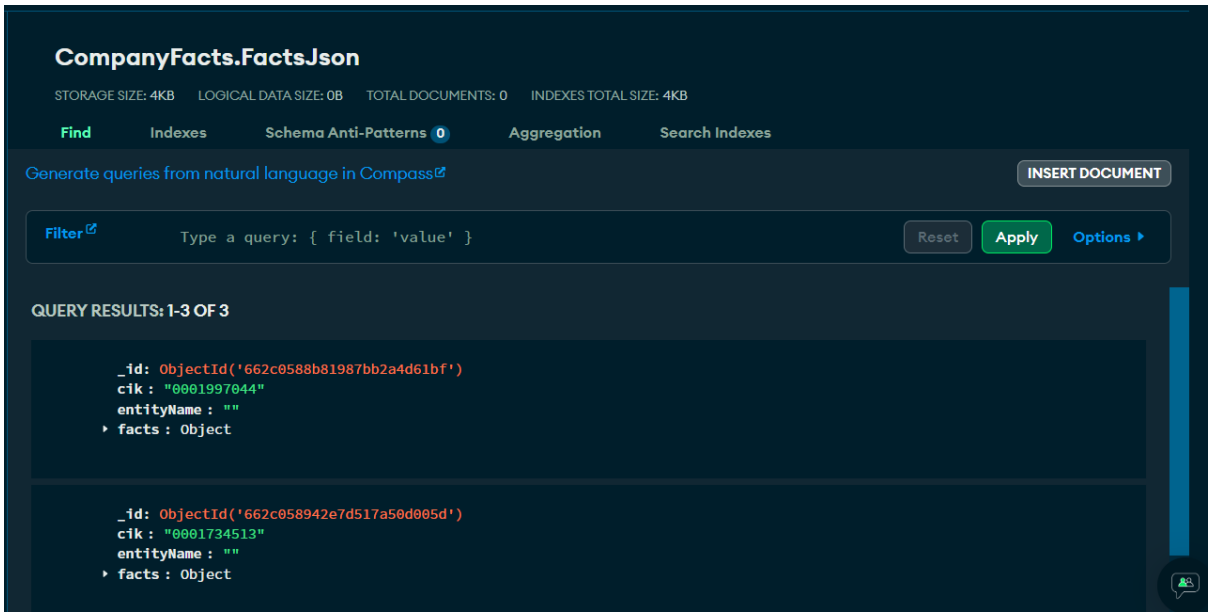


- Asdsa





- Mo


CompanyFacts.FactsJson
STORAGE SIZE: 4KB
LOGICAL DATA SIZE: 0B
TOTAL DOCUMENTS: 0
INDEXES TOTAL SIZE: 4KB
Find
Indexes
Schema Anti-Patterns 0
Aggregation
Search Indexes
Generate queries from natural language in Compass
INSERT DOCUMENT
Filter
Type a query: { field: 'value' }
Reset
Apply
Options
QUERY RESULTS: 1-3 OF 3
_id: ObjectId('662c0588b81987bb2a4d61bf')
cik : "0001997044"
entityName : ""
facts : Object
_id: ObjectId('662c058942e7d517a50d005d')
cik : "0001734513"
entityName : ""
facts : Object

-