*Pipeline description*

This pipeline makes an API call, uploads the returned file to S3, triggering a Lambda function, which calls a Glue job. It Includes:

1. A scheduled Airflow DAG with two tasks

- task1: PythonOperator calls a function that makes an API call, then saves the returned data to a local file.

- task2: PythonOperator calls a function that instantiates an s3 resource via boto3, then uploads the local file to an AWS S3 bucket.

2. An AWS Lambda function that is triggered when an object is placed in an S3 bucket, calls:

3a. AWS Glue job that prints a success message.

or

3b. An AWS Glue workflow that runs the (above) Glue job, then calls a Glue crawler..

1. Airflow DAG

| # 'Dynamic tasks': call multiple tasks from 'for' loop (this test makes single call) # built from 'find\_the\_file\_from\_binubuo.py' # API: # https://rapidapi.com/codemonth/api/binubuo # bucket = 'elt2-jja' # 'key' : f'api\_files/{file}.csv'  import requests import boto3 from datetime import datetime from airflow import DAG from airflow.operators.python import PythonOperator data\_file = 'local\_data\_file.csv' # local file to pass data task1 > task2 bucket = 'elt2-jja'  def file\_save(file\_name):  url = f"https://binubuo.p.rapidapi.com/data/custom/{file\_name}" # from func call in task  querystring = {"locale":"US","csv":"True","rows":"5","cols":"first\_name,last\_name,address"}  headers = {  "X-RapidAPI-Key": "7de9129cfamsh827e4e4491e9d8dp1a708ajsn464837a3ec51",  "X-RapidAPI-Host": "binubuo.p.rapidapi.com"  }  # response = requests.get(url, headers=headers, params=querystring) # trigger Glue job  response = glue.start\_workflow\_run(Name = 'elt2-workflow') # trigger workflow  # api call for .csv file  if response.status\_code == 200:  with open(data\_file, 'wb') as file:  file.write(response.content) # successful call, create local .csv file  print("The CSV file has been saved successfully.")  else: # else show error  print("Failed to retrieve data. Status code:", response.status\_code)  def retrieve\_upload\_file(bucket, key):  # IAM user 'external\_access' keys  s3 = boto3.resource( # configure s3 resource  service\_name='s3',  region\_name='us-east-1',  aws\_access\_key\_id='AKIAZ2NPLMMGFEGGYNOS',  aws\_secret\_access\_key='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'  )  # upload local file to s3 bucket  s3.Bucket(bucket).upload\_file(Filename=data\_file, Key=key)  with DAG(  dag\_id='dynamic\_tasks',  description='API call/save s3 file in for loop',  start\_date=datetime(2024, 1, 27),  # schedule\_interval=timedelta(seconds=60), # switch interval after test  schedule\_interval = '0 \*/1 \* \* \*', # every hour  catchup=False ) as dag:  files = ('quick\_fetch',) # tuple of files (all at same API endpoint)  for file in files:  task1 = PythonOperator(  task\_id = f'file\_save\_{file}',  python\_callable = file\_save,  op\_kwargs = {'file\_name' : file} # pass API file name to func  )  task2 = PythonOperator(  task\_id = f'retrieve\_file\_{file}',  python\_callable = retrieve\_upload\_file,  op\_kwargs = {'bucket' : bucket,'key' : f'api\_files/{file}.csv'}  )    task1 >> task2 |
| --- |

2. Lambda function

| import json import boto3  def lambda\_handler(event, context):  # Replace these values with your own  region = 'us-east-1'  glue\_job\_name = 'elt2-glue'   glue\_client = boto3.client('glue', region\_name=region) # Create a Glue client  try:  response = glue\_client.start\_job\_run(JobName=glue\_job\_name) # Trigger Glue job  print(f"Glue job run started successfully: {response['JobRunId']}")  return {  'statusCode': 200,  'body': f"elt2-glue job started: {response['JobRunId']}"  }  except Exception as e:  print(f"Error triggering Glue job: {str(e)}")  return {  'statusCode': 500,  'body': f"Error triggering Glue job: {str(e)}"  } |
| --- |

3. Glue job

| import sys from awsglue.transforms import \* from awsglue.utils import getResolvedOptions from pyspark.context import SparkContext from awsglue.context import GlueContext from awsglue.job import Job  ## @params: [JOB\_NAME] args = getResolvedOptions(sys.argv, ['JOB\_NAME'])  sc = SparkContext() glueContext = GlueContext(sc) spark = glueContext.spark\_session job = Job(glueContext) job.init(args['JOB\_NAME'], args)  print("elt2-glue job ran")  job.commit() |
| --- |