PRABHAKAR CHOUDHARY

Abstract

Develop an ETL automated event based triggering serverless process using AWS Services like Lambda, Glue, Event Bridge and S3 to load and transform CSV file

AWS Glue etl event based trigger serverless pipeline

Table of Contents

1. Serverless Architectural Design
2. Copy Data from On-premise Data Center
3. Create S3 Bucket
4. Create Serverless Glue Crawler to read CSV/TSV files (TAB delimited)
5. Launch Notebook (From Development Perspective, not required in production)
6. Create Serverless Glue job with PySpark
7. Create Serverless Lambda Functions

* Create Lambda function for S3 event-based trigger to Crawler
* Create another Lambda function for Crawler state change to trigger to Glue job
* Create third Lambda function for Email trigger by SNS after Glue job status change

1. Create IAM role
2. Create SNS topic to get email about Glue job completion or failure
3. Create an Event Bridge rule to get Crawler log and trigger Lambda function
4. script to re-name the file from S3 and Load to EC2 (optional as per project requirement and infrastructure available)

AWS S3

AWS GLUE JOB ( PySpark)

AWS LAMBDA

SNS/ Email Job Status Change

AWS LAMBDA

EVENT BRIDGE

Event based Trigger

AWS GLUE CRAWLER

AWS LAMBDA

AWS EC2

SearchPerforamanceKey.tab (Target)/ EC2 Linux server

Data.CSV file ( Source)

AWS S3

**Step: - 2 Create an EC2 instance for Linux and run below: -**

Create a Python / Shell script like below to copy data from

Process: - 1

%sh

Echo “Start file transmission process in below Shell script “

Sftp -v -oPORT=<Port Number> <userid>@<domain> <<! >> $logfile >&<log directory>

Put “$src\_file” “$Target Directory”

Ls -l “$Target Directory”

Quit

!

Schedule above using CRONTAB -L

Process 2: - If there is AWS CLI present we can transfer file from EC2 to S3 using below command on AWS CLI and scheduled using Crontab

Aws s3 copy test.csv s3:// testadobeglue

**Step: - 3 Create S3 Bucket – Source #**

Go to AWS Console under S3

Create source S3 bucket, where source data will be uploaded and provide below role to get it accessed Publicly.

{

"Version": "2008-10-17",

"Statement": [

{

"Sid": "AllowPublicRead",

"Effect": "Allow",

"Principal": {

"AWS": "\*"

},

"Action": "s3:GetObject",

"Resource": "arn:aws:s3::: [testadobeglue](https://s3.console.aws.amazon.com/s3/buckets/testadobeglue?region=us-west-1)/\*"

}

]

}

Testadobeglue - arn:aws:s3:::testadobeglue

**Create S3 Bucket – Target** searchkeywordperformance

Provide below IAM role

{

"Version": "2008-10-17",

"Statement": [

{

"Sid": "AllowPublicRead",

"Effect": "Allow",

"Principal": {

"AWS": "\*"

},

"Action": "s3:GetObject",

"Resource": "arn:aws:s3:::[searchkeywordperformance](https://s3.console.aws.amazon.com/s3/buckets/searchkeywordperformance?region=us-west-1)/\*"

}

]

}



**Step 4: - Create a crawler using this S3 path.**

**#** pbkadobecrawler

Data source: - s3://testadobeglue

Give this role: - [AWSGlueServiceRole-readS3](https://console.aws.amazon.com/iam/home#/roles/AWSGlueServiceRole-readS3)

And it will catalog below table in this Database

Database: - [pbkadobeproject](https://us-west-1.console.aws.amazon.com/glue/home?region=us-west-1#/v2/data-catalog/databases/view/pbkadobeproject?catalogId=382078627397)

Table :- [testadobeglue](https://us-west-1.console.aws.amazon.com/glue/home?region=us-west-1#/v2/data-catalog/tables/view/testadobeglue?database=pbkadobeproject&catalogId=382078627397&versionId=latest)

**Step5: - Launch notebook and provide below roles so that developer endpoint could be connected using PY-Spark**

Amazon S3 FULL ACCESS

Amazon Glue Service Role

AWS GlueSessionUserRestrictedNotebookpolicy

AWS Glue passrole

{

"Version": "2012-10-17",

"Statement": [{

"Effect": "Allow",

"Action": [

"iam:GetRole",

"iam:PassRole"

],

"Resource": "arn:aws:iam::account-id:role/EC2-roles-for-XYZ-\*"

}]

}

**Step 6: -Glue Py-Spark job to read the Crawler and transform and stage the data to S3 bucket (SearchkeywordPerformance)**

Glue Job name :- [pbkadobegluejob](https://us-west-1.console.aws.amazon.com/gluestudio/home?region=us-west-1#/editor/job/pbkadobegluejob)

Role - CrawlerTriggerGlueRole

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

from pyspark.sql.functions import lit

from pyspark.sql.types import NullType

from awsglue.dynamicframe import DynamicFrame

from awsglue.transforms import DropNullFields

from pyspark.sql.functions import split,col

from pyspark.sql.functions import regexp\_extract, col

from urllib.parse import urlparse, parse\_qs

from pyspark.sql.functions import col, coalesce, lit

sc = SparkContext.getOrCreate()

glueContext = GlueContext(sc)

spark = glueContext.spark\_session

job = Job(glueContext)

dynamicFrameCustomers = glueContext.create\_dynamic\_frame.from\_catalog(

database = "pbkadobeproject",

table\_name = "testadobeglue"

)

dyfCustomerSelectFields = dynamicFrameCustomers.select\_fields(["ip","product\_list", "referrer","event\_list"])

sparkDF1= dyfCustomerSelectFields.toDF()

sparkDF2 = sparkDF1.withColumn("product\_list\_new",split(col("product\_list"),"[;]"))

sparkDF3 = sparkDF2.withColumn("product\_list\_new",split(col("product\_list"),"[;]"))\

.withColumn("Prod\_Type",col("product\_list\_new")[1])\

.withColumn("Revenue",col("product\_list\_new")[3])

sparkDF4 = sparkDF3.withColumn('Search Keyword', regexp\_extract(col('referrer'), 'search\?[\w]+[\=]((\w\*\+\*)\*)&',1))\

.withColumn('Search Engine Domain', regexp\_extract(col('referrer'), '[https|http]+://+[\w]\*[\.]+(\w\*\.+\w\*)[\/|\S]',1))\

.filter(col("Search Engine Domain").isNotNull())\

.drop('product\_list','referrer','Prod\_Type','product\_list\_new')

sparkDF5= sparkDF4.filter(col("event\_list") =="1").drop('event\_list','Search Engine Domain','Search Keyword')\

.withColumnRenamed("Revenue","DF5Revenue")

sparkDF6 = sparkDF5.join(sparkDF4,["ip"],"right")\

.filter(~ col("Search Engine Domain").like("esshopzilla.co%"))\

.drop('ip','event\_list','Revenue')\

.withColumnRenamed("DF5Revenue","Revenue")\

.filter(col("Search Engine Domain").isNotNull())

sparkDF7 = sparkDF6.sort(col("Revenue").desc())

dyfCustomersConvert = DynamicFrame.fromDF(sparkDF7, glueContext, "convert")

dyfCustomersConvert = dyfCustomersConvert.repartition(1)

glueContext.write\_dynamic\_frame.from\_options(

frame = dyfCustomersConvert,

connection\_type="s3",

connection\_options = {"path": "s3://searchkeywordperformance"},

format = "csv",

format\_options={

"separator": " ",

"writeHeader": True,

"quoteChar": -1

},

transformation\_ctx = "datasink2")

**Step 7: -Create LAMBDA functions to automate the entire workflow, as a part of that we are going to create 3 Lambda functions below**

* [LambdaTriggerGlueCrawler](https://us-west-1.console.aws.amazon.com/lambda/home?region=us-west-1#/functions/LambdaTriggerGlueCrawler)- This Lambda function is designed to be triggered on S3 event for any new file upload to this bucket.

Role provided :- [LambdaTriggerGlueCrawler](https://us-west-1.console.aws.amazon.com/iam/home#/roles/LambdaTriggerGlueCrawler?section=permissions) This role will give all the access to Glue console

# Set up logging

import json

import os

import logging

logger = logging.getLogger()

logger.setLevel(logging.INFO)

# Variables for the job:

REGION = 'us-west-1'

CRAWLER = 'pbkadobecrawler'

# Import Boto 3 for AWS Glue

import boto3

from botocore.exceptions import ClientError

glue = boto3.client(service\_name='glue',region\_name=REGION)

# Define Lambda function

def lambda\_handler(event, context):

try:

logger.info('## INITIATED BY S3 PUT EVENT: ')

logger.info(f'## starting glue CRAWLER :{CRAWLER}')

glue.start\_crawler(Name=CRAWLER)

except ClientError as e:

if e.response.get('Error', {}).get('Code') == 'CrawlerRunningException':

logger.info('## GLUE CRAWLER: {CRAWLER} Already Running')

else:

raise e

Above Lambda is triggered by below S3 bucket

##### ****S3****: [testadobeglue](https://us-east-1.console.aws.amazon.com/s3/buckets/testadobeglue?region=us-east-1" \t "_blank)

* [LambdaCrawlerTriggerGlueJob](https://us-west-1.console.aws.amazon.com/lambda/home?region=us-west-1#/functions/LambdaCrawlerTriggerGlueJob)- This lambda function is designed to be triggered by Event Bridge Bus after status change in Crawler job and in turn it will trigger Glue job where transformation will take place by python/ Spark platform in code written in Py-Spark

Role Provided to this function is :- [LambdaTriggerGlueCrawler](https://us-west-1.console.aws.amazon.com/iam/home#/roles/LambdaTriggerGlueCrawler?section=permissions)

# Set up logging

import json

import os

import logging

logger = logging.getLogger()

logger.setLevel(logging.INFO)

# Variables for the job:

REGION = 'us-west-1'

GLUEJOB = 'pbkadobegluejob'

# Import Boto 3 for AWS Glue

import boto3

from botocore.exceptions import ClientError

glue = boto3.client(service\_name='glue',region\_name=REGION)

# Define Lambda function

def lambda\_handler(event, context):

try:

logger.info('## INITIATED BY Glue Crawler EVENT from Event Bridge: ')

logger.info(f'## starting glue Job :{GLUEJOB}')

glue.start\_job\_run(JobName=GLUEJOB)

except ClientError as e:

if e.response.get('Error', {}).get('Code') == 'CrawlerRunningException':

logger.info('## GLUE JOB: {GLUEJOB} Already Running')

else:

raise e

* [Lambda\_Trigger\_Email](https://us-west-1.console.aws.amazon.com/lambda/home?region=us-west-1#/functions/Lambda_Trigger_Email)- This lambda function is designed to trigger the SNS topic and send email after Glue job status change.

import boto3

def lambda\_handler(event, context):

# TODO implement

client = boto3.client('sns')

snsArn = 'arn:aws:sns:us-west-1:<your account id>:Glue\_Job\_Failed'

message = "Production :- Glue job loading Adobe Data has failed."

name = event.get("name")

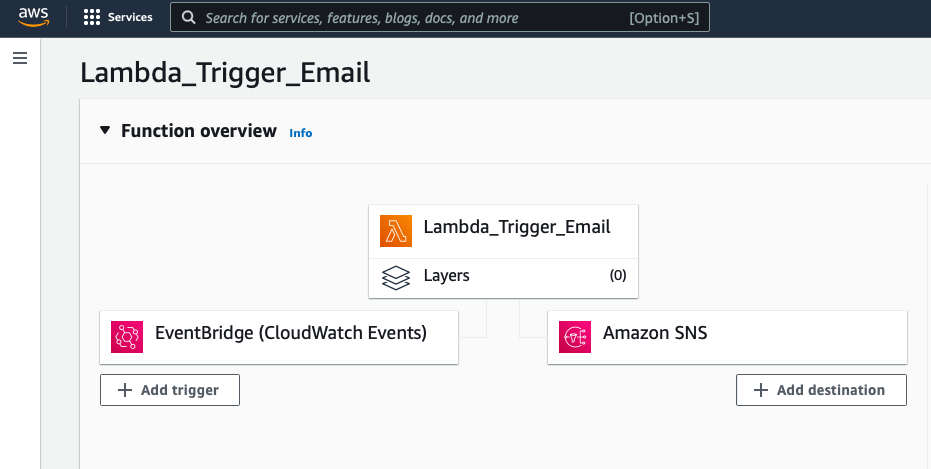
response = client.publish(

TopicArn = snsArn,

Message = message ,

Subject='Prod Glue job failed'

)



**Step 8: - Below IAM roles has been used in above process**

[AWSGlueServiceAdobeRole](https://us-east-1.console.aws.amazon.com/iamv2/home?region=us-west-1#/roles/details/AWSGlueServiceAdobeRole)

Pass role: -

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"iam:GetRole",

"iam:PassRole"

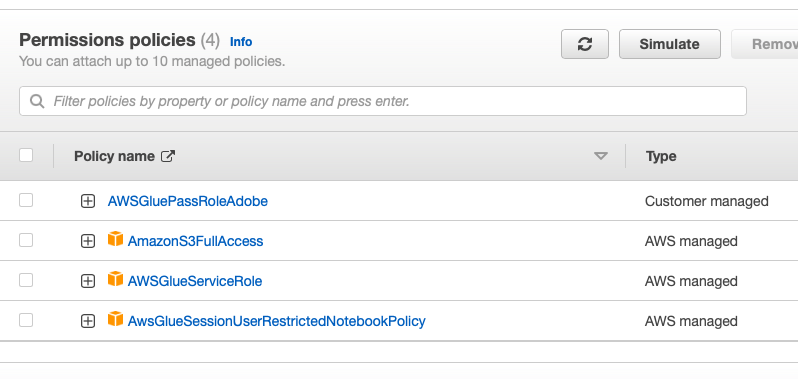
],

"Resource": "arn:aws:iam::382078627397:role/\*"

}

]

}



# First: - **AWSGlueServiceRole-readS3**

## AWSGlueServiceRole-readS3-EZCRC-s3Policy

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"s3:GetObject",

"s3:PutObject"

],

"Resource": [

"arn:aws:s3:::pbkadobeproject\*"

]

}

]

}

## Second: - **AWSGlueServiceRole**

# Third: - **CrawlerTriggerGlueRole**

## AmazonS3FullAccess

## CloudWatchFullAccess

## AWSGlueConsoleFullAccess

# Fourth :- **lambdaglues3access**

## AmazonS3FullAccess

## CloudWatchFullAccess

## AWSGlueConsoleFullAccess

# Fifth: - **LambdaTriggerEmail**

## AWSLambdaSNSTopicDestinationExecutionRole-abcfd725-3999-4ba5-af28-250a3f59d5fd

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": "sns:Publish",

"Resource": "arn:aws:sns:us-west-1:<put your account id >:Glue\_Job\_Failed"

}

]

}

## AmazonSESFullAccess

## AWSLambdaExecute

# Sixth: -**LambdaTriggerGlueCrawler**

## AWSGlueServiceRole

## AWSLambdaBasicExecutionRole

# Seventh: -**LambdaWriteCloudWatchLogRole**

## CloudWatchFullAccess

## CloudWatchLogsFullAccess

**Step 9: - Create SNS topic and rule for Email trigger**

**Topic #** [Glue\_Job\_Failed](https://us-west-1.console.aws.amazon.com/sns/v3/home?region=us-west-1#/topic/arn:aws:sns:us-west-1:382078627397:Glue_Job_Failed)

**Access policy**

{

"Version": "2008-10-17",

"Id": "\_\_default\_policy\_ID",

"Statement": [

{

"Sid": "\_\_default\_statement\_ID",

"Effect": "Allow",

"Principal": {

"AWS": "\*"

},

"Action": [

"SNS:GetTopicAttributes",

"SNS:SetTopicAttributes",

"SNS:AddPermission",

"SNS:RemovePermission",

"SNS:DeleteTopic",

"SNS:Subscribe",

"SNS:ListSubscriptionsByTopic",

"SNS:Publish"

],

"Resource": "arn:aws:sns:us-west-1:<replace with account id >:Glue\_Job\_Failed",

"Condition": {

"StringEquals": {

"AWS:SourceOwner": "<replace with account id >"

}

}

}

]

}

**Step 10: - Create an Event Bridge Rule for Email trigger and Crawler trigger (Here I used default event bus although it is preferred to create a custom one as per project demand)**

[**Glue\_Event\_Rule\_Email**](https://us-west-1.console.aws.amazon.com/events/home?region=us-west-1#/eventbus/default/rules/Glue_Event_Rule_Email)

**Event Pattern**

**{**

**"source": ["aws.glue"],**

**"detail-type": ["Glue Job State Change"]**

**}**

[**event\_Glue\_Crawler\_Rule**](https://us-west-1.console.aws.amazon.com/events/home?region=us-west-1#/eventbus/default/rules/event_Glue_Crawler_Rule)

**Event Pattern**

**{**

**"source": ["aws.glue"],**

**"detail-type": ["Glue Crawler State Change"],**

**"detail": {**

**"state": ["Succeeded"]**

**}**

**}**

**Step 11: - Create a Shel / Python script to rename and stage to desired EC2 instances (Linux))**

**Automate it using crontab**

**%sh**

Aws s3 copy s3://searchperformancetab searchperformance.tab