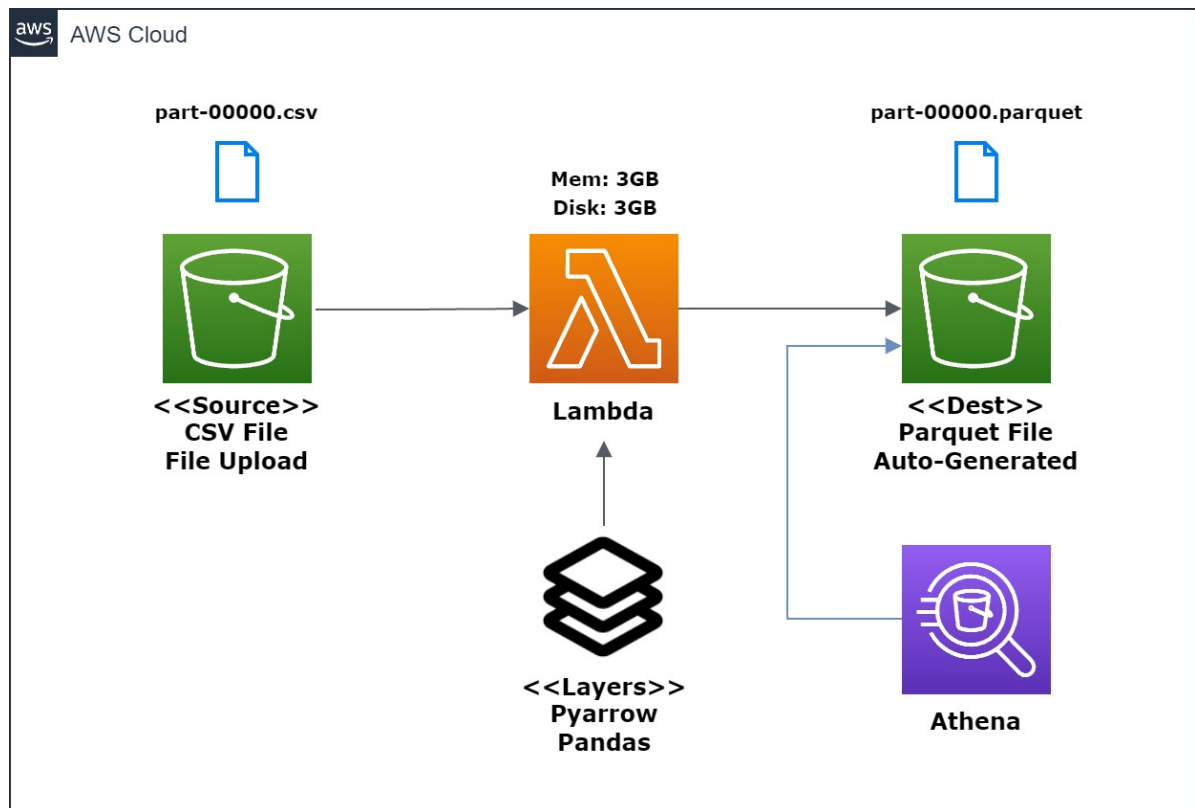


[IGAWorks] 과제 전형 (제출용)

0. 제출 (ZIP) 파일 구조

```
assignment_knk/
├─ pipeline_architecture.jpg
├─ README.md
├─ README.pdf
├─ src
│   ├── etl
│   │   └─ lambda_handler.py
│   └─ sql
│       ├── 00.\ 00. create_part_00000.sql
│       ├── 00.\ 00. create_part_00001.sql
│       ├── 01.\ active_user_by_date.sql
│       ├── 02.\ total_price_active_user_by_date.sql
│       ├── 03.\ active_user_by_event.sql
│       ├── 04.\ active_user_by_campaign.sql
│       ├── 05.\ total_price_active_user_by_campaign.sql
│       ├── 06.\ active_user_total_price_by_country_campaign.sql
│       ├── 07.\ hard_funnel(oracle).sql
│       ├── 07.\ hard_funnel.sql
│       ├── 08.\ hard_new_install_funnel_by_partner.sql
│       ├── 09.\ hard_retention_by_dates(oracle).sql
│       └─ query_result
│           ├── 01.\ active_user_by_date.csv
│           ├── 01.\ active_user_by_date.PNG
│           ├── 02.\ total_price_active_user_by_date.csv
│           ├── 02.\ total_price_active_user_by_date.PNG
│           ├── 03.\ active_user_by_event.csv
│           ├── 03.\ active_user_by_event.PNG
│           ├── 04.\ active_user_by_campaign.csv
│           ├── 04.\ active_user_by_campaign.PNG
│           ├── 05.\ total_price_active_user_by_campaign.csv
│           ├── 05.\ total_price_active_user_by_campaign.PNG
│           ├── 06.\ active_user_total_price_by_country_campaign.csv
│           ├── 06.\ active_user_total_price_by_country_campaign.PNG
│           ├── 07.\ hard_funnel.csv
│           ├── 07.\ hard_funnel.PNG
│           ├── 08.\ hard_new_install_funnel_by_partner.csv
│           └─ 08.\ hard_new_install_funnel_by_partner.PNG
```

1. 파이프라인 아키텍처



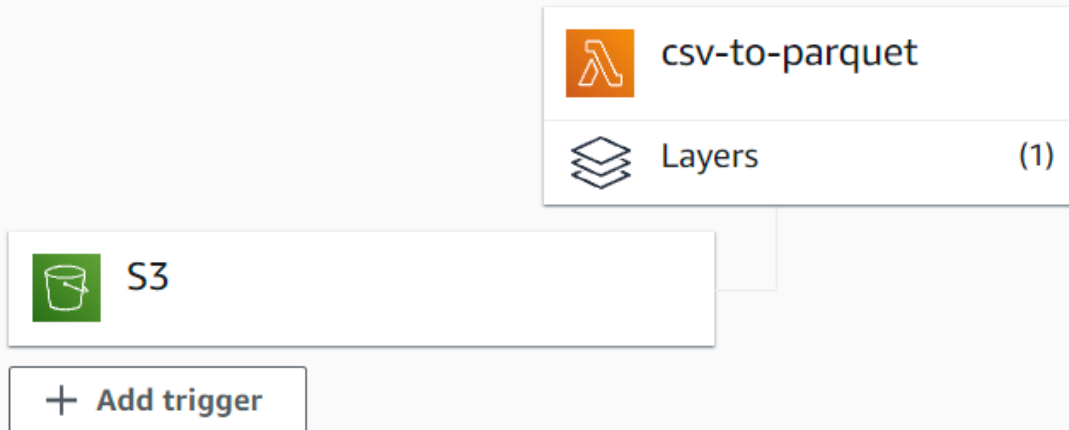
해당 아키텍처로 선택한 이유

- 과제 시작 직전에는 Ubuntu ECS 인스턴스 (컨테이너)에서 변환 작업을 하는줄 알았지만, 과제 목표중 하나인 ETL에 드는 노력을 0에 가깝게 합니다. 를 충족하지 못할거같아 과제 2일째 Pyspark 로 방향을 바꾸었습니다.
- S3에 파일이 업로드 될때마다 Spark Job을 Container로 기동시켜 변환을 하는 방법밖에 떠오르지 않아 시도해보았지만, SQS, ECS, EMR, Athena를 한번에 습득하기엔 Learning Curve가 너무 길고, Lambda, Athena로 파이프라인을 구현할 수 있을것 같아 해당 아키텍처를 선택했습니다.

Data Flow

Trigger

▼ Function overview [Info](#)



source-part 라는 Bucket에 데이터가 들어오면 Lambda Function이 Trigger 되는 형식입니다.

CloudWatch Logs on lambda function

▶	Timestamp	Message
		No older events at this moment. Retry
▶	2022-10-24T21:53:29.479+09:00	OpenBLAS WARNING - could not determine the L2 cache size on this system, assuming 256k
▶	2022-10-24T21:53:30.392+09:00	START RequestId: 8ef51f2e-c4fb-4721-828d-5e1b1a453012 Version: \$LATEST
▶	2022-10-24T21:53:31.600+09:00	part-00000.csv
▶	2022-10-24T21:53:31.658+09:00	END RequestId: 8ef51f2e-c4fb-4721-828d-5e1b1a453012
▶	2022-10-24T21:53:31.658+09:00	REPORT RequestId: 8ef51f2e-c4fb-4721-828d-5e1b1a453012 Duration: 1265.51 ms Billed Duration: 1265.51 ms Memory Usage: 128 MB Max Memory Usage: 128 MB
		No newer events at this moment. Auto retry paused. Resume

Layers

- CSV to Parquet을 달성하기 위해 Lambda function에서 Pandas, Pyarrow Library를 Import 했습니다.
- 해당 library를 Import 하려면 제가 사용하고자 하는 library를 ZIP 형태로 묶어 layers에 업로드합니다.

Runtime settings

Info

Actions

Edit

Runtime

Python 3.8

Handler

Info

lambda_function.lambda_handler

Architecture

Info

x86_64

Layers

Info

Edit

Add a layer

Merge order	Name	Layer version	Compatible runtimes	Compatible architectures	Version ARN
1	pyarrow	1	python3.8	x86_64	arn:aws:lambda:ap-northeast-2:810094538532:layer:pyarrow:1

```
[root@networkserver PyVenv]# ls
lambda-pyarrow-3.8
[root@networkserver PyVenv]# ls -al lambda-pyarrow-3.8/
total 73972
drwxr-xr-x.  5 root root    4096 Oct 20 08:06 .
drwxr-xr-x.  3 root root    4096 Oct 20 08:03 ..
drwxr-xr-x.  2 root root    4096 Oct 20 07:50 bin
-rw-r--r--.  1 root root     40 Oct 20 07:50 .gitignore
-rw-r--r--.  1 root root 75715591 Oct 20 08:06 layer.zip
drwxr-xr-x.  3 root root    4096 Oct 20 07:50 lib
drwxr-xr-x. 16 root root    4096 Oct 20 08:03 python
-rw-r--r--.  1 root root    228 Oct 20 07:50 pyvenv.cfg
[root@networkserver PyVenv]# ls -al lambda-pyarrow-3.8/python/
total 100
drwxr-xr-x. 16 root root    4096 Oct 20 08:03 .
drwxr-xr-x.  5 root root    4096 Oct 20 08:06 ..
drwxr-xr-x.  2 root root    4096 Oct 20 08:03 bin
drwxr-xr-x.  6 root root    4096 Oct 20 08:03 dateutil
drwxr-xr-x. 21 root root    4096 Oct 20 08:03 numpy
drwxr-xr-x.  2 root root    4096 Oct 20 08:03 numpy-1.23.4.dist-info
drwxr-xr-x.  2 root root    4096 Oct 20 08:03 numpy.libs
drwxr-xr-x. 16 root root    4096 Oct 20 08:03 pandas
drwxr-xr-x.  2 root root    4096 Oct 20 08:03 pandas-1.5.1.dist-info
drwxr-xr-x.  9 root root    4096 Oct 20 08:03 pyarrow
drwxr-xr-x.  2 root root    4096 Oct 20 08:03 pyarrow-9.0.0.dist-info
drwxr-xr-x.  2 root root    4096 Oct 20 08:03 __pycache__
drwxr-xr-x.  2 root root    4096 Oct 20 08:03 python_dateutil-2.8.2.dist-info
drwxr-xr-x.  4 root root    4096 Oct 20 08:03 pytz
drwxr-xr-x.  2 root root    4096 Oct 20 08:03 pytz-2022.5.dist-info
drwxr-xr-x.  2 root root    4096 Oct 20 08:03 six-1.16.0.dist-info
-rw-r--r--.  1 root root 34549 Oct 20 08:03 six.py
```

Destination

Amazon S3 > Buckets > destination-part > part-00000/

part-00000/ Copy S3 URI

Objects Properties

Objects (1)

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](#)

Refresh
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Create folder
Upload

Find objects by prefix

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	part-00000.parquet	parquet	October 23, 2022, 21:47:27 (UTC+09:00)	968.3 MB	Standard

- Lambda Function을 통해 변환 된 후, destination-part라는 Bucket으로 key가 생성 됩니다.

Lambda Function Source Code

src/etl/lambda_handler.py 에서 확인 하실 수 있습니다.

```
from io import StringIO
from io import BytesIO
from pyarrow import csv,parquet

import json
import boto3
import urllib.parse
import pandas as pd
import pyarrow as pa

endpoint_url = '*'
access_key = '*'
secret_key = '*'

s3_client = boto3.client('s3', region_name='ap-northeast2',
endpoint_url=endpoint_url, aws_access_key_id=access_key,
aws_secret_access_key=secret_key)
s3_resource = boto3.resource('s3', region_name='ap-northeast2',
endpoint_url=endpoint_url, aws_access_key_id=access_key,
aws_secret_access_key=secret_key)

def lambda_handler(event, context):
    csv_buffer = StringIO()
    par_buffer = BytesIO()

    bucket = event['Records'][0]['s3']['bucket']['name']
    csv = urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'],
encoding='utf-8')
    dest = csv.replace(".csv", "")
    name = csv.replace('csv', 'parquet')

    if name == 'part-00000':
        col = ['identity_adid', 'os', 'model', 'country', 'event_name', 'log_id',
'server_datetime', 'quantity', 'price']
        resp = s3_client.get_object(Bucket=bucket, Key=csv)
        data = pd.read_csv(resp['Body'], sep=',')
        data.columns = col
        data = data.fillna(0)
        data =
data.astype({'identity_adid':'str','os':'str','model':'str','quantity':'int',
'price':'float32', 'server_datetime':'str'})
    elif name == 'part-00001':
        col = ['partner', 'campaign', 'server_datetime', 'tracker_id', 'log_id',
'attribution_type', 'identity_adid']
```

```

        data = pd.read_csv(resp['Body'], sep=',')
        data.columns = col

        data = data.astype({'identity_adid':'str', 'server_datetime':'str'})
    else:
        print("Not a valid input.")

    try:
        data.to_parquet(par_buffer, engine='pyarrow', index=False)
        s3_resource.Object(f'destination-part/{dest}',
f'{name}').put(Body=par_buffer.getvalue())
    except Exception as e:
        print("Data Invalid.", e)

```

2. Athena

Event

```

CREATE EXTERNAL TABLE event (
    IDENTITY_ADID STRING,
    OS STRING,
    MODEL STRING,
    COUNTRY STRING,
    EVENT_NAME STRING,
    LOG_ID STRING,
    SERVER_DATETIME STRING,
    QUANTITY INT,
    PRICE FLOAT
) STORED AS PARQUET
LOCATION 's3://destination-part/part-00000/'
TBLPROPERTIES ('classification' = 'parquet')

```

Attribution

```
CREATE EXTERNAL TABLE ATTRIBUTION (
    PARTNER STRING,
    CAMPAIGN STRING,
    SERVER_DATETIME STRING,
    TRACKER_ID STRING,
    LOG_ID STRING,
    ATTRIBUTION_TYPE INT,
    IDENTITY_ADID STRING
) STORED AS PARQUET
LOCATION 's3://destination-part/part-00001/'
TBLPROPERTIES ('classification' = 'parquet')
```

server_datetime을 String으로 선언한 이유

- CSV에서 Parquet으로 변환시 datetime으로 선언했을 때는 문제가 되지 않았지만, Athena에서 Data를 조회할때마다 빈 화면이 보여 Parquet의 컬럼 타입이 안맞는다고 생각했습니다.
- 정확히 어떤 컬럼에서 에러가 발생하는지 찾기위해, 제 Hive Server에 직접 Parquet을 Load해보니 다음과 같은 에러가 발생했습니다.
- `org.apache.hadoop.io.LongWritable cannot be cast to org.apache.hadoop.hive.serde.io.TimestampWritable`
- 어떤 이유인지 모르겠지만, Hive에서 server_datetime을 지속적으로 LongWritable로 인식하여 timestamp로 load하지 못했던것입니다.













- ```
초기에 'server_datetime'을 datetime 형식으로 변환 시도 했지만, LongWritable로 인식합니다
data['server_datetime'] = pd.to_datetime(data['server_datetime'])
```

- 해당 이유로 인해, `partition by server_datetime` 은 테이블 create시 선언하지 않고, query로 조회할때 cast하여 사용했습니다.

## 3. Query Results

- sql 파일명 뒤에 (oracle)이 있는 파일은 athena query 결과가 확실하지 않아 oracle로 짜본 쿼리입니다.
- 9번 `hard_retention_by_dates(oracle).sql` 은 athena로 해결하지 못해 oracle 쿼리로 짜보았습니다.

## query\_result

-  00. create\_part\_00000
-  00. create\_part\_00001
-  01. active\_user\_by\_date
-  02. total\_price\_active\_user\_by\_date
-  03. active\_user\_by\_event
-  04. active\_user\_by\_campaign
-  05. total\_price\_active\_user\_by\_campaign
-  06. active\_user\_total\_price\_by\_country\_ca...
-  07. hard\_funnel(oracle)
-  07. hard\_funnel
-  08. hard\_new\_install\_funnel\_by\_partner
-  09. hard\_retention\_by\_dates(oracle) ✖