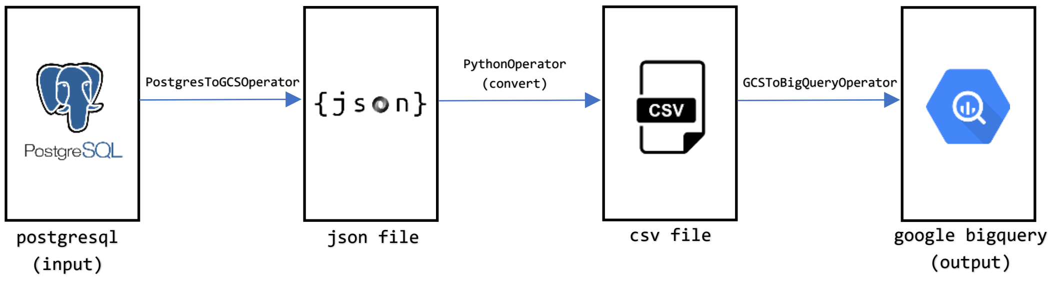
Interview Challenge (Data Engineer) – Document

Apache Airflow Architecture



Dags

- postgresql_to_gcs_user_log.py – Dag file that query 'user_log' data from postgresql and upload to GCS
- postgresql_to_gcs_users.py – Dag file that query 'users' data from postgresql and upload to GCS
- user_log_to_bigquery.py – Dag file that transform 'user_log' postgresql data to csv file and load to Google BQ
- users_to_bigquery.py – Dag file that transform 'users' postgresql data to csv file and load to Google BQ

Google Cloud Platform Service Usage

- Google Compute Engine
- Google Storage
- Google BigQuery
- Google Dataproc

----

airflow-vm - GCE instance

IP: 35.213.169.173

Machine Type: e2-highcpu-4 (4 vCPUs, 4 GB memory)

Description: Apache Airflow Single Node

Airflow URL: http://35.213.169.173:8080/home

Airflow Account:

- Admin
    - Username: jon
    - Password: password
- Viewer
    - Username: bluepi
    - Password: password

Airflow PATH: /srv/airflow

Airflow DAG(s) Directory: /srv/airflow/dags

Services:

1. airflow-webserver.service – autostart
2. airflow-scheduler.service - autostart

Usage Guide:

1. sudo su airflow
2. source /srv/airflow/bin/activate
3. sudo systemctl status airflow-webserver.service
4. sudo systemctl status airflow-scheduler.service

jupyter-bigquery-m - GCE instance

IP: 35.213.131.149

Machine Type: n1-standard-4 (4 vCPUs, 15 GB memory)

Description: Jupyter Notebook master node for Google Dataproc Cluster

jupyter-bigquery-w-0 - GCE instance

IP: 35.213.135.153

Machine Type: n1-standard-4 (4 vCPUs, 15 GB memory)

Description: Jupyter Notebook worker node 1 for Google Dataproc Cluster jupyter-bigquery-w-1 - GCE instance

IP: 35.213.143.219

Machine Type: n1-standard-4 (4 vCPUs, 15 GB memory)

Description: Jupyter Notebook worker node 2 for Google Dataproc Cluster

----

airflow-postgres - GCS instance

Description: Contain JSON and CSV file for Airflow Pipeline

bigquery_bluepi_output - GCS instance

Description: Contain files that use in Google Dataproc Cluster

dataproc-temp* - GCS instance

Description: Contain files when Google Dataproc Cluster get processing

----

sirapob-bluepi-de-exam:airflow_gcs_to_bigquery – Google BigQuery

Description: Google BigQuery Dataset

sirapob-bluepi-de-exam:airflow_gcs_to_bigquery.user_log_to_bigquery – Google BigQuery

Description: Google BigQuery ‘user_log’ Table

sirapob-bluepi-de-exam:airflow_gcs_to_bigquery.users_to_bigquery – Google BigQuery

Description: Google BigQuery ‘users’ Table

----

jupyter-bigquery – Google Dataproc

Jupyter Notebook URL: Jupyter Notebook

Type: Dataproc Cluster

Cluster Detail:

1. jupyter-bigquery-m – Master
2. jupyter-bigquery-w-0 – Worker
3. jupyter-bigquery-w-1 – Worker

Jupyter Notebook

Output Monitor Files

- pySpark_user_log_monitor.ipynb
- pySpark_users_monitor.ipynb

pySpark_user_log_monitor.ipynb - Jupyter Notebook IPYNB

In [2]:
```
from pyspark.sql import SparkSession
spark = SparkSession.builder \
  .appName('pySpark_user_log_monitoring')\
  .config('spark.jars', 'gs://spark-lib/bigquery/spark-bigquery-latest.jar') \
  .getOrCreate()
```

In [3]:
```
spark.conf.set("spark.sql.repl.eagerEval.enabled",True)
```

In [4]:
```
table = "sirapob-bluepi-de-exam:airflow_gcs_to_bigquery.user_log_to_bigquery"

user_log_data = spark.read \
  .format("bigquery") \
  .option("table", table) \
  .option("dateFormat", "yyyy-MM-dd HH:mm:ss") \
  .load()

user_log_data.printSchema()
```

```
root
 |-- action: string (nullable = false)
 |-- created_at: timestamp (nullable = false)
 |-- id: string (nullable = false)
 |-- success: boolean (nullable = false)
 |-- updated_at: timestamp (nullable = false)
 |-- user_id: string (nullable = false)
```

In [5]:
```
user_log_data_table = user_log_data \
  .select("id", "user_id", "action", "success", "created_at", "updated_at")

user_log_data_table.toPandas()
```

Out[5]:

	id	user_id	action	success	created_at	updated_at
0	4b796e06-3178-4133-ad30-dc505bfc13f5	55514cf0-3026-404f-8ea3-f41b00bdf6b5	login	False	2020-02-17 01:46:05.934519	2020-02-17 01:46:05.934519
1	8745cacb-f8aa-4294-b824-2d3a5c50f171	55514cf0-3026-404f-8ea3-f41b00bdf6b5	login	False	2020-02-17 01:46:06.934519	2020-02-17 01:46:06.934519
2	c12bcaa2-563c-416e-91db-36d846b0feae	55514cf0-3026-404f-8ea3-f41b00bdf6b5	change password	False	2021-03-22 02:10:15.010466	2021-03-22 02:10:15.010466
3	6cfd027d-b50b-4855-b279-91f99ce4476d	55514cf0-3026-404f-8ea3-f41b00bdf6b5	change password	True	2020-02-17 01:50:07.934519	2020-02-17 01:50:07.934519
4	c0c97762-694b-45fd-a41e-7304313eab82	55514cf0-3026-404f-8ea3-f41b00bdf6b5	login	True	2020-02-17 01:56:07.934519	2020-02-17 01:56:07.934519
5	59131a90-62c9-40d8-899e-31c3dee8ad7e	d0e73a35-ff6a-4f64-89b4-ed2b813782a3	login	True	2020-02-17 01:48:08.934519	2020-02-17 01:48:08.934519
6	5bccf37b-5d03-4c54-a76e-5eb54a3290a5	d0e73a35-ff6a-4f64-89b4-ed2b813782a3	logout	True	2020-02-17 01:52:07.934519	2020-02-17 01:52:07.934519

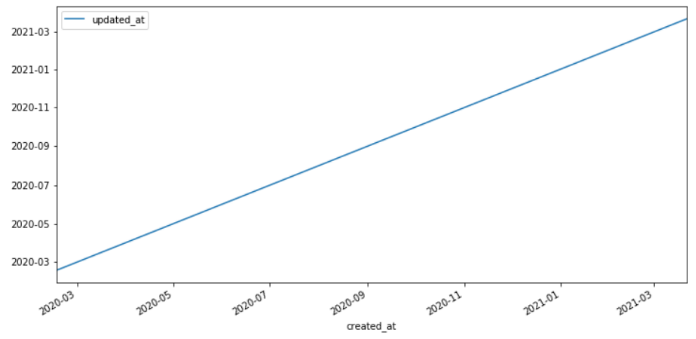
In [35]:
```
spark.conf.set("spark.sql.execution.arrow.enabled", "true")

user_log = user_log_data_table.toPandas()
user_log.set_index('created_at', inplace=True)
user_log.head()
```

Out[35]:

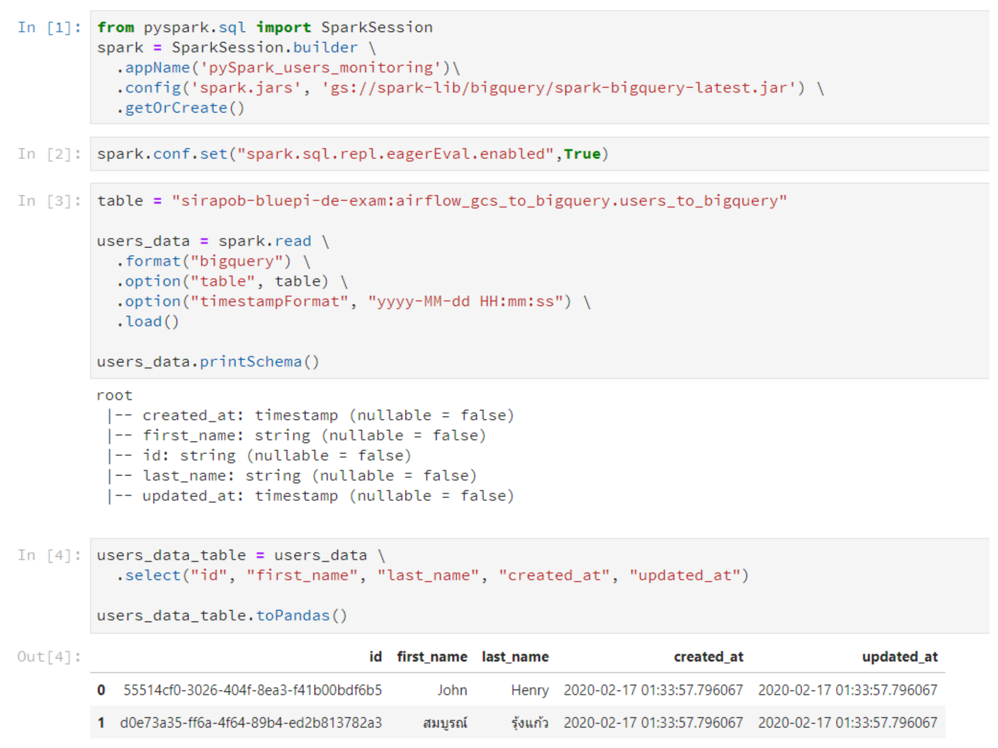
	id	user_id	action	success	updated_at
created_at					
2020-02-17 01:46:05.934519	4b796e06-3178-4133-ad30-dc505bfc13f5	55514cf0-3026-404f-8ea3-f41b00bdf6b5	login	False	2020-02-17 01:46:05.934519
2020-02-17 01:46:06.934519	8745cacb-f8aa-4294-b824-2d3a5c50f171	55514cf0-3026-404f-8ea3-f41b00bdf6b5	login	False	2020-02-17 01:46:06.934519
2021-03-22 02:10:15.010466	c12bcaa2-563c-416e-91db-36d846b0feae	55514cf0-3026-404f-8ea3-f41b00bdf6b5	change password	False	2021-03-22 02:10:15.010466
2020-02-17 01:50:07.934519	6cfd027d-b50b-4855-b279-91f99ce4476d	55514cf0-3026-404f-8ea3-f41b00bdf6b5	change password	True	2020-02-17 01:50:07.934519
2020-02-17 01:56:07.934519	c0c97762-694b-45fd-a41e-7304313eab82	55514cf0-3026-404f-8ea3-f41b00bdf6b5	login	True	2020-02-17 01:56:07.934519

In [36]:
```
user_log.plot(kind='line',figsize=(12,6))
```

Out[36]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f666e468d30>
```



pySpark_users_monitor.ipynb – Jupyter Notebook IPYNB



In [1]:
```
from pyspark.sql import SparkSession
spark = SparkSession.builder \
  .appName('pySpark_users_monitoring')\
  .config('spark.jars', 'gs://spark-lib/bigquery/spark-bigquery-latest.jar') \
  .getOrCreate()
```

In [2]:
```
spark.conf.set("spark.sql.repl.eagerEval.enabled",True)
```

In [3]:
```
table = "sirapob-bluepi-de-exam:airflow_gcs_to_bigquery.users_to_bigquery"

users_data = spark.read \
  .format("bigquery") \
  .option("table", table) \
  .option("timestampFormat", "yyyy-MM-dd HH:mm:ss") \
  .load()

users_data.printSchema()
```

```
root
 |-- created_at: timestamp (nullable = false)
 |-- first_name: string (nullable = false)
 |-- id: string (nullable = false)
 |-- last_name: string (nullable = false)
 |-- updated_at: timestamp (nullable = false)
```

In [4]:
```
users_data_table = users_data \
  .select("id", "first_name", "last_name", "created_at", "updated_at")

users_data_table.toPandas()
```

Out[4]:

	id	first_name	last_name	created_at	updated_at
0	55514cf0-3026-404f-8ea3-f41b00bdf6b5	John	Henry	2020-02-17 01:33:57.796067	2020-02-17 01:33:57.796067
1	d0e73a35-ff6a-4f64-89b4-ed2b813782a3	สมบูรณ์	รุ้งแก้ว	2020-02-17 01:33:57.796067	2020-02-17 01:33:57.796067