Taxi Data Pipeline Step by Step

A diagram of a cloud

Description automatically generated

1. Part 1
   1. Save the CSV file.
   2. Create an EC2
      1. Instance type: t2.medium (2vCPU 4Gib Mem) – for Apache Airflow
      2. Attach the Key Pair (Can create new and then attach use for login to ec2 instance)
      3. Network setting, allow SSH, HTTPS & HTTP traffic.
      4. Launch instance.

Airflow Installation Steps on Ec2 : Ubuntu with 1cpu 2GiB

==================================

**Instance Update:**

sudo apt update

sudo apt install -y python3-pip

sudo apt install -y sqlite3

**Create Py environment:**

sudo apt install -y python3.10-venv

virtual environment: python3 -m venv ia\_taxi\_venv

sudo apt-get install -y libpq-dev

**activate virtual environment:**

source /home/ubuntu/ia\_taxi\_venv/bin/activate

**Install Postgres:**

pip install "apache-airflow[postgres]==2.5.0" --constraint "https://raw.githubusercontent.com/apache/airflow/constraints-2.5.0/constraints-3.7.txt"

airflow db init

sudo apt-get install -y postgresql postgresql-contrib

sudo -i -u postgres

psql

CREATE DATABASE airflow;

CREATE USER airflow WITH PASSWORD 'airflow';

GRANT ALL PRIVILEGES ON DATABASE airflow TO airflow;

**Update DB connections:**

cd airflow/

sed -i 's#sqlite:////home/ubuntu/airflow/airflow.db#postgresql+psycopg2://airflow:airflow@localhost/airflow#g' airflow.cfg

grep sql\_alchemy airflow.cfg

grep executor airflow.cfg

sed -i 's#SequentialExecutor#LocalExecutor#g' airflow.cfg

**Initilize Airflow**

airflow db init

**Create User**

- airflow users create -u airflow -f airflow -l airflow -r Admin -e airflow@gmail.com

**Start WebServer**

- airflow webserver &

**Start Scheduler**

- airflow scheduler

Crate bash scripts to Start and Stop Airflow Webserver and Scheduler

Start Airflow :

* vi startair.sh

#!/bin/bash

# Start Airflow scheduler

airflow scheduler > /dev/null 2>&1 &

scheduler\_pid=$!

# Start Airflow webserver

airflow webserver -p 8080 > /dev/null 2>&1 &

webserver\_pid=$!

echo "Airflow scheduler and webserver started."

echo "Scheduler PID: $scheduler\_pid"

echo "Webserver PID: $webserver\_pid"

Stop Airflow :

* vi stopair.sh

#!/bin/bash

# Stop Airflow scheduler

scheduler\_pid=$(pgrep -f "airflow scheduler")

if [[ -n $scheduler\_pid ]]; then

kill $scheduler\_pid

echo "Airflow scheduler stopped."

else

echo "Airflow scheduler is not running."

fi

# Stop Airflow webserver

webserver\_pid=$(pgrep -f "airflow webserver")

if [[ -n $webserver\_pid ]]; then

kill $webserver\_pid

echo "Airflow webserver stopped."

else

echo "Airflow webserver is not running."

fi

chmod a+x startair.sh

chmod a+x stopair.sh

Part 2:

Create a s3 bucket

Name : datalz1

Part 3: Glue

1. Create glue Crawlers

Name: taxi\_crawlers

Description: to crawler the dataset

A screenshot of a screenshot of a computer

Description automatically generated

Select the datasource already mapped as Not yet

A screenshot of a computer

Description automatically generated

Add a data source

A screenshot of a computer

Description automatically generated

Create an IAM role for Glue

Name: glue\_admin\_role

Grant: admin access

A screenshot of a computer

Description automatically generated

Assign the above IAM role to Glue Crawler:

A screenshot of a computer

Description automatically generated

Set output and scheduling:

1. Creating glue temp database

A screenshot of a computer

Description automatically generated

1. Assign the temp glue database to the crawler

Target database: taxi\_glue\_db

Crawler schedule; On Demand (as we will be calling it from airflow)

A screenshot of a computer

Description automatically generated

Glue Crawler created.

A screenshot of a computer

Description automatically generated

Now we have created the Glue Crawler, which know from where to get the data and also has the glue database to temporarily store the metadata of the crawled data.

Test data set has been uploaded to [taxi-landing](https://s3.console.aws.amazon.com/s3/buckets/taxi-landing?region=us-east-1) -🡪 [taxi\_zones.csv](https://s3.console.aws.amazon.com/s3/object/taxi-landing?region=us-east-1&prefix=taxi_zones.csv) to run and test the glue crawler.

Run crawler and meta database created.

A screenshot of a computer

Description automatically generated

Navigate to Amazon Athena to check if we an access the meta database created by crawler.

A screenshot of a computer

Description automatically generated

Query ran and data populated.

A screenshot of a computer

Description automatically generated

Created bucket to save the Athena query

A screenshot of a web page

Description automatically generated

AWS Redshift:

Go to redshift cluster and create new