Algolia Document

Table des matières

Project overview

This project has as goal to prepare a test for Algolia hiring process, this project will be developed by Python, this document will describe the step, the details and prerequisite of this project.

Prerequisite:

To work on this project, we should have the bellow environment.

1- Python 3 should be installed in your machine
2- Install git and jupyter if you want work on the repository
3- Prepare a Postgres target database
4- Make sure that we have all the required privileges and access

Environment

Python

Install python 3 in your machine, you can do it by installing Anaconda https://www.anaconda.com/ and pip install the necessary libraries

Repository

You can access to github repository of the project by clone it in your local machine , the repository is open to public, the link of this repository is https://github.com/boudalis/algolia_test

Database

A Postgres database should be installed and prepared, the database should be accessible from your Python environment , in our case we have installed a Postgres in AWS RDS algdb-1.c3kwj6orkqge.eu-west-3.rds.amazonaws.com

Access

Make sure that there are no issues with proxy, firewall, user privileges and network connection

Project

Target Database

We have created the target database in AWS RDS, the services is named “algdb-1”

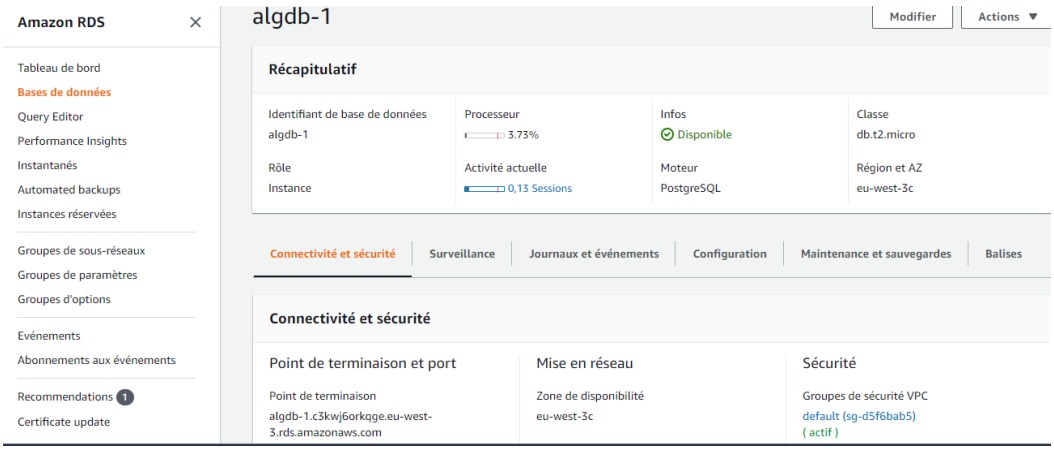

Figure 1 Algdb-1 AWS RDS overview

The algdb-1 AWS RDS service has the below properties

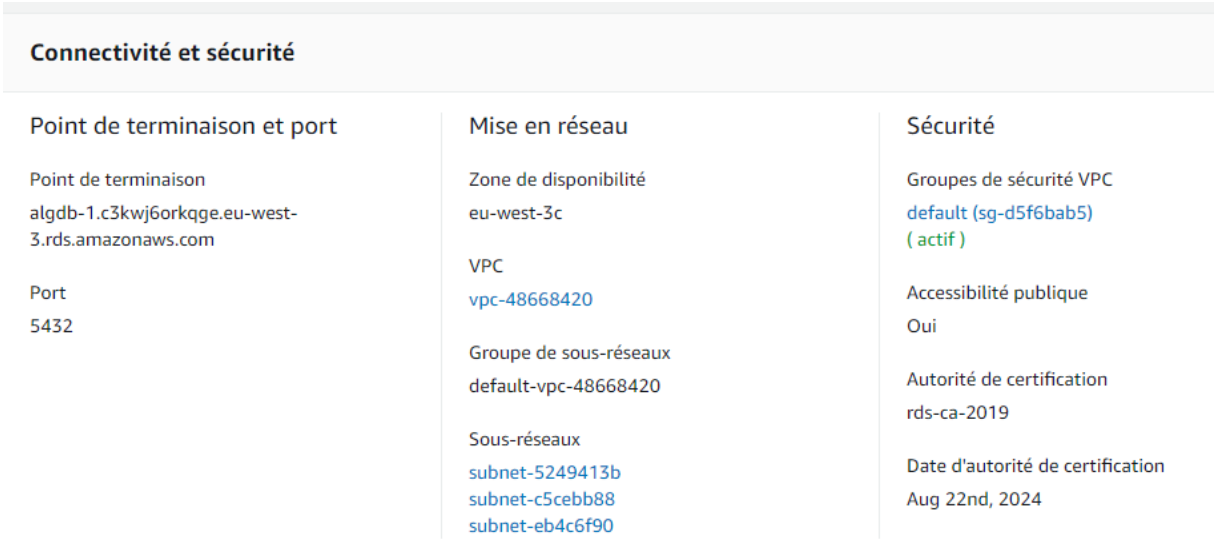

Figure 2 algdb-1 AWS RDS properties

you can access to data base with the below credential:

connection: algdb-1.c3kwj6orkqge.eu-west-3.rds.amazonaws.com

User : postgres

Pwd : postgres

Port : 5432

You can manage the database by installing the PgAdmin in your machine

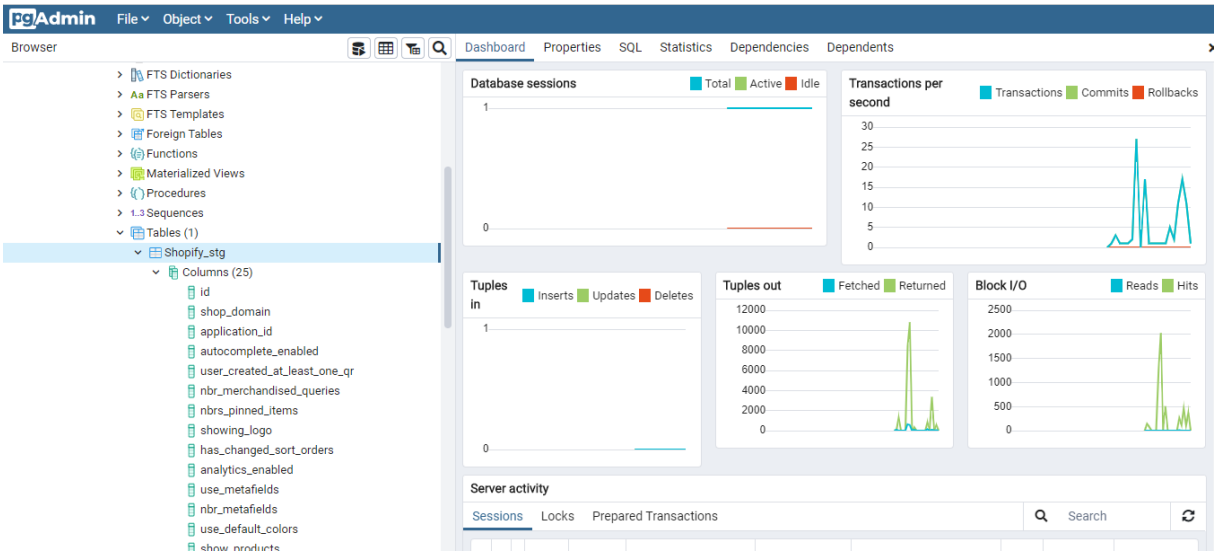

Figure 3 PgAdmin overview

Before starting using the program you should prepare you database:

1-create a database:

```
CREATE DATABASE algtst
    WITH
    OWNER = postgres
    ENCODING = 'UTF8'
    LC_COLLATE = 'en_US.UTF-8'
    LC_CTYPE = 'en_US.UTF-8'
    TABLESPACE = pg_default
    CONNECTION LIMIT = -1;
COMMENT ON DATABASE algtst     IS 'algtst';
```

2-create a schema:

```
CREATE SCHEMA alg_stg
    AUTHORIZATION postgres;
```

3-create a table:

```
CREATE TABLE alg_stg."Shopify_stg"
(
    id character varying(100) COLLATE pg_catalog."default",
    shop_domain character varying(100) COLLATE pg_catalog."default",
    application_id character varying(100) COLLATE pg_catalog."default",
    autocomplete_enabled character varying(100) COLLATE pg_catalog."default",
    user_created_at_least_one_qr character varying(100) COLLATE pg_catalog."default",
    nbr_merchandised_queries character varying(100) COLLATE pg_catalog."default",
    nbrs_pinned_items character varying(100) COLLATE pg_catalog."default",
    showing_logo character varying(100) COLLATE pg_catalog."default",
    has_changed_sort_orders character varying(100) COLLATE pg_catalog."default",
    analytics_enabled character varying(100) COLLATE pg_catalog."default",
    use_metafields character varying(100) COLLATE pg_catalog."default",
```

```
    nbr_metafields character varying(100) COLLATE pg_catalog."default",
    use_default_colors character varying(100) COLLATE pg_catalog."default",
    show_products character varying(100) COLLATE pg_catalog."default",
    instant_search_enabled character varying(100) COLLATE pg_catalog."default",
    instant_search_enabled_on_collection character varying(100) COLLATE pg_catalog."default",
    only_using_faceting_on_collection character varying(100) COLLATE pg_catalog."default",
    use_merchandising_for_collection character varying(100) COLLATE pg_catalog."default",
    index_prefix character varying(100) COLLATE pg_catalog."default",
    indexing_paused character varying(100) COLLATE pg_catalog."default",
    install_channel character varying(100) COLLATE pg_catalog."default",
    export_date character varying(100) COLLATE pg_catalog."default",
    has_specific_prefix character varying(100) COLLATE pg_catalog."default",
    date_insert character varying(100) COLLATE pg_catalog."default",
    file_name character varying(200) COLLATE pg_catalog."default"
)

TABLESPACE pg_default;

ALTER TABLE alg_stg."Shopify_stg"
    OWNER to postgres;
```

Python Development

The development was done by creating 5 files:

constants.py

this file contains the initial parameters, those parameters can update based on the need of the user like the target database, rage of date to load and the URL of the file to load.

Note that we can change the structure of the source file, but we should update the structure of the target table

```
url = 'https://alg-data-public.s3.amazonaws.com/'

db_url='postgresql+psycopg2://postgres:postgres@algdb-1.c3kwj6orkqge.eu-west-3.rds.amazonaws.com:5432/algtst'

tgt_table='alg_stg."Shopify_stg"'

start_date = '2019-04-01'
```

end_date = '2019-04-07'

alg_def.py

This file contains all the function and the logic of development, before starting the development we should import the necessary libraries (Pandas, io, requests, sqlalchemy, psycopg2, datetime, logging)

The file contains 6 functions:

1- read_file(url): this function has as goal to read the file from the url and load it in a dataframe.
2- prepare_data(df,url): this function clean and prepare the data to be loaded in the target.
3- insert_date(df,db_url): insert the dataframe in the target table.
4- load_all_files(url,db_url,start_date,end_date): this function load the files between two date. It will be used in the history load.
5- load_daily_file (url,db_url): this function load the file of the current date , it can be used for the scheduling .
6- display_query(query): this function returns the result of a query in a dataframe, it will be used in the unit test

alg_main_daily.py

this file should be used if we want to schedule our project at daily level, he will load the file of each day.

We can run this file by the below cmd :

Python alg_main_daily.py

alg_main_his.py

this file should be used to load the historical data, he will load the files between a range of dates, those dates can be initialized in the constants.py file (start_date, end_date)

We can run this file by the below cmd :

Python alg_main_his.py

unit_tests.py

this file contains a list of tests that can give us a visibility about the quality of the data and if all the loadings are done without problems, you can run the test one by one or you can run all of them on the same time the result will be displayed in the console.

Note that you can add your own tests by adding new line in the script, the function display_query(SqlQuery) is used to load the result of the SqlQuery in a dataframe that we will diplay it (print(display_query(SqlQuery)).

Figure 4 Console, result of the unit tests

Conclusion

This project can be updated and improved, if you have some remarks or ideas to improve the project feel free to share them.