Solución Reto - Data Pipelines en Apache Airflow

Presentado por:

Marcelo Diez Morales
Andrés Soto Hincapie
Mayo 2023

Resumen

En el presente documento se muestran los resultados obtenidos para la implementación de un data pipeline usando la herramienta Apache Airflow con el objetivo de realizar los procesos de extracción, transformación y carga de datos en un datawarehouse que dará soporte al "Sistema Integrado de Egresados Universitarios". Una base de datos construida a partir de información pública obtenida de distintos portales de información en la Unión Europea.

Tecnologías usadas

Como se mencionó inicialmente, la principal herramienta usada para la construcción del data pipeline ha sido Apache Airflow, sin embargo esta a su vez requiere de un ecosistema de servicios adicionales que soportan su correcto funcionamiento, a continuación se presenta una lista de las herramientas usadas y una breve descripción por la cual fue su uso en el proyecto.

1. **Apache Airflow:** Plataforma para programar y orquestar flujos de trabajo de manera automatizada y controlada. Es la principal herramienta donde se programan los flujos de datos para el proyecto.

2. **MySQL:** Sistema de gestión de bases de datos relacionales ampliamente utilizado para almacenar y administrar datos estructurados. Es la base de datos usada para la construcción del datawarehouse.

3. **PostgreSQL:** Sistema de gestión de bases de datos relacional y de código abierto con características avanzadas como soporte para datos geoespaciales y tipos de datos personalizados. Este motor de base de datos se utiliza para almacenar las configuraciones y metadata de Apache Airflow.

4. **Redis:** Base de datos en memoria de alta velocidad utilizada para almacenar datos en estructuras de clave-valor y para cachear información. Redis se utiliza como backend de Airflow para mejorar el rendimiento y la escalabilidad de la plataforma, permitiendo ejecutar flujos de trabajo más grandes y manejar mayores volúmenes de datos de manera eficiente.

5. **GitHub:** Plataforma de alojamiento de código fuente y colaboración para desarrolladores, que permite gestionar y compartir proyectos utilizando el sistema de control de versiones Git. En esta plataforma se almacena el código fuente de la solución.

6. **WSL** (Windows Subsystem for Linux): Capa de compatibilidad de Microsoft que permite ejecutar aplicaciones y herramientas de Linux en entornos Windows. A través de esta herramienta se realiza la creación de una máquina virtual con Ubuntu para facilitar la ejecución de los contenedores de docker y tener mayor compatibilidad con ciertas herramientas open source.

7. **Docker:** Plataforma para crear y gestionar contenedores virtuales que facilitan la implementación y ejecución de aplicaciones en diferentes entornos sin problemas de compatibilidad. A través de docker se aprovisionan todas las herramientas necesarias para la ejecución del proyecto.

8. **Docker Hub:** Registro de imágenes de contenedores Docker, donde los desarrolladores pueden compartir y distribuir imágenes preconfiguradas para su uso en entornos Docker. Desde docker hub se realiza la descarga de las distintas imágenes usadas en el despliegue de la aplicación, como pueden ser postgres, MySQL, Airflow, Redis, etc

Aprovisionamiento de recursos

Para la correcta ejecución del proceso es necesario hacer la configuración de las distintas aplicaciones en un entorno local. Los pasos realizados para lograrlo fueron:

1. Instalar WSL y Ubuntu en Windows.

2. Clonar el repositorio del proyecto dentro de WSL.

3. Definir las variables de entorno en un archivo .env para la instalación de librerías adicionales o configuraciones iniciales de Airflow.

4. Realizar la instalación de docker en conjunto con WSL.

5. Descargar las imágenes y desplegar los contenedores usando la herramienta docker compose.

6. Ejecutar el script de SQL dentro de MySQL para crear la base de datos con su respectivo esquema.

Una vez realizados los pasos anteriores, se cuenta con el ambiente de trabajo listo para iniciar con la creación de los data pipelines en el servidor web de Airflow.

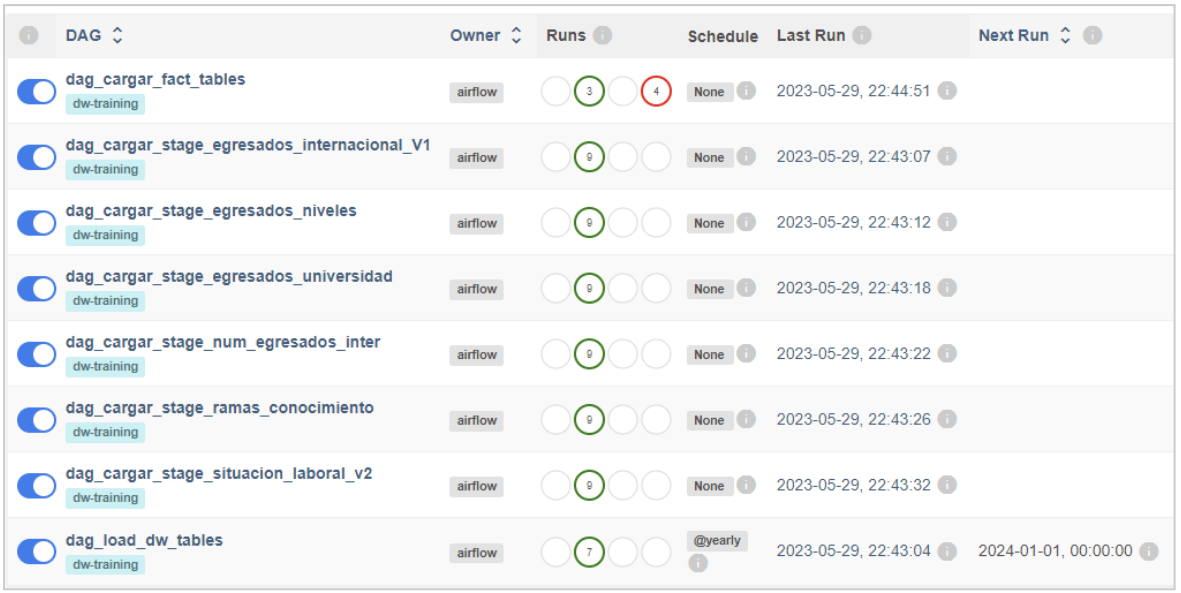
Proceso de creación y resultados del Pipeline

Para la solución del proyecto se crearon un total de 8 DAG de Airflow con sus respectivas tareas y 9 ejecutables de python con el código necesario para realizar los procesos de transformación y carga de los datos.

A continuación se explica el funcionamiento de cada uno de estos:

- Archivo ***/dags/dag_load_all_dw_tables.py*** → Es el DAG principal que se encarga de orquestar la ejecución de los demás DAG así como las tareas y sensores de cada proceso.
- Archivo ***/dags/dag_load_fact_tables.py*** → A través de este DAG se realiza la carga de las tablas de hechos.
- Archivo ***/dags/helpers/load_dimms.py*** → Contiene las funciones de python para la transformación y carga de las tablas de dimensión.
- Archivo ***/dags/dag_stage_egresados_inter.py*** → A través de este DAG se realiza la carga de las tablas de staging - stage_egresados_internacional.
- Archivo ***/dags/dag_stage_egresados_niveles.py*** → A través de este DAG se realiza la carga de las tablas de staging - stage_egresados_niveles.
- Archivo ***/dags/dag_stage_egresados_universidad.py*** →A través de este DAG se realiza la carga de las tablas de staging - stage_egresados_universidad.
- Archivo ***/dags/dag_stage_num_egresados_inter.py*** →A través de este DAG se realiza la carga de las tablas de staging - stage_num_egresados_internacional.
- Archivo ***/dags/dag_stage_ramas_conocimiento.py*** →A través de este DAG se realiza la carga de las tablas de staging - stage_ramas_conocimiento.
- Archivo ***/dags/dag_stage_situacion_laboral_2.py*** → A través de este DAG se realiza la carga de las tablas de staging - stage_situacion_laboral_egresados.
- Archivo ***/dags/helpers/load_fact.py*** → Contiene las funciones de python para la transformación y carga de las tablas de hechos.
- Archivo ***/dags/helpers/load_file1.py*** → Contiene las funciones de python para la transformación y carga de las tablas de staging - egresados universidad SEGR1.csv y SEGR2.csv
- Archivo ***/dags/helpers/load_file2.py*** → Contiene las funciones de python para la transformación y carga de las tablas de staging - archivos egresados niveles grad_5sc.csv.
- Archivo ***/dags/helpers/load_file3.py*** → Contiene las funciones de python para la transformación y carga de las tablas de staging - archivos porcentaje egresados internacional educ_uoe_grad05.xlsx
- Archivo ***/dags/helpers/load_file4.py*** → Contiene las funciones de python para la transformación y carga de las tablas de staging - archivos archivo situacion_laboral_egresados 03003.xlsx.
- Archivo ***/dags/helpers/load_file5.py*** → Contiene las funciones de python para la transformación y carga de las tablas de staging - archivos archivo ramas del conocimiento ISCED_2013.csv.
- Archivo ***/dags/helpers/load_file6.py*** → Contiene las funciones de python para la transformación y carga de las tablas de staging - num_egresados_inter educ_uoe_grad01.xlsx.

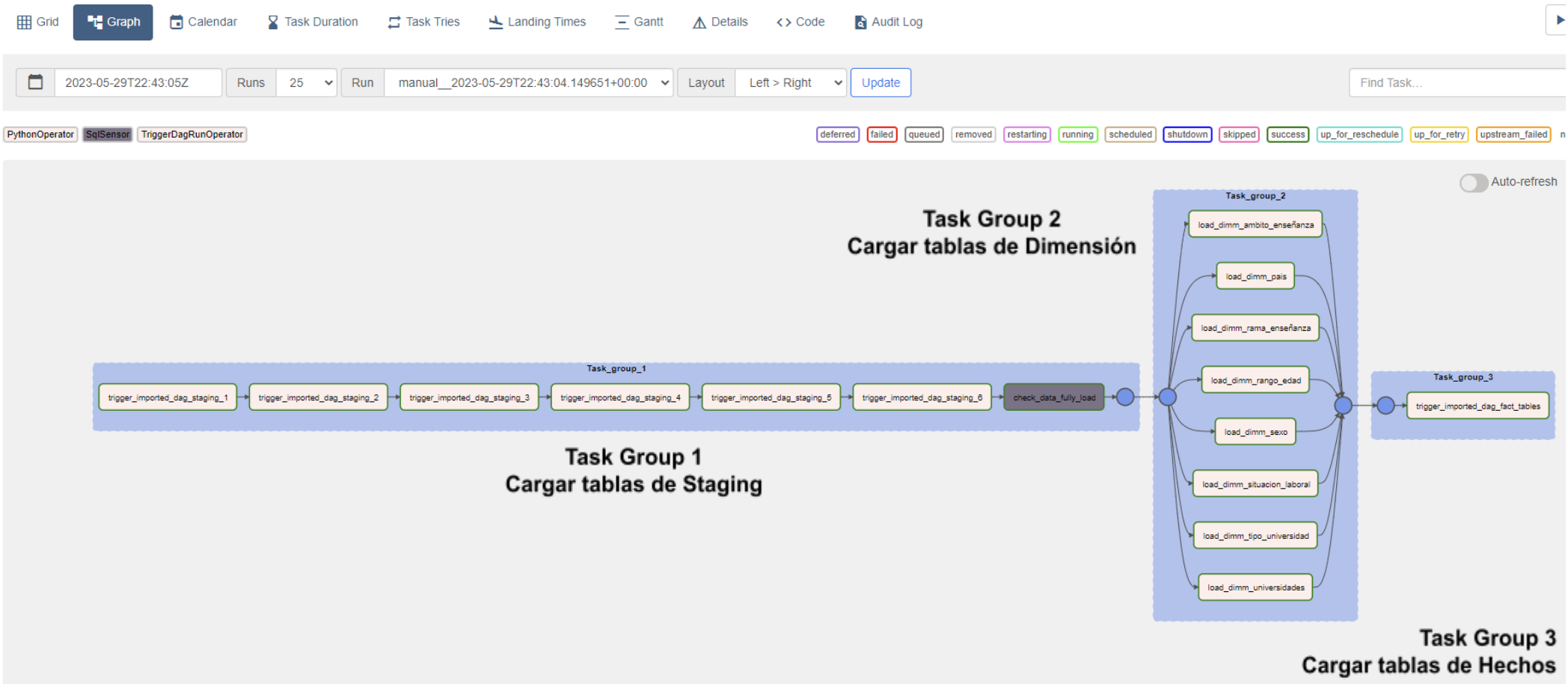
Una vez definidos los distintos DAG, estos se cargan de manera automática en Airflow gracias a las configuraciones definidas en el archivo docker-compose.yaml



DAG	Owner	Runs	Schedule	Last Run	Next Run
dag_cargar_fact_tables dw-training	airflow	3 4	None	2023-05-29, 22:44:51	
dag_cargar_stage_egresados_internacional_V1 dw-training	airflow	9	None	2023-05-29, 22:43:07	
dag_cargar_stage_egresados_niveles dw-training	airflow	9	None	2023-05-29, 22:43:12	
dag_cargar_stage_egresados_universidad dw-training	airflow	9	None	2023-05-29, 22:43:18	
dag_cargar_stage_num_egresados_inter dw-training	airflow	9	None	2023-05-29, 22:43:22	
dag_cargar_stage_ramas_conocimiento dw-training	airflow	9	None	2023-05-29, 22:43:26	
dag_cargar_stage_situacion_laboral_v2 dw-training	airflow	9	None	2023-05-29, 22:43:32	
dag_load_dw_tables dw-training	airflow	7	@yearly	2023-05-29, 22:43:04	2024-01-01, 00:00:00

En la imagen anterior podemos ver los distintos DAG cargados de manera exitosa, así como su histórico de ejecuciones. En la siguiente imagen se muestra el workflow del DAG principal **dag_load_dw_tables** el cual se encarga de orquestar la ejecución de todas las tareas del pipeline.

En este DAG se importan los demás DAG definidos en el proyecto usando el operador de Airflow TriggerDagRunOperator, así mismo se crean 3 grupos de tareas para distinguir las fases de carga entre las tablas de staging, dimensiones y hechos. Dado que la tarea de carga de las tablas de staging es la que toma más tiempo, se implementa un SqlSensor con el objetivo de monitorizar cuando las tablas se encuentran cargadas y proceder con las demás tareas del pipeline. El proceso de carga de las tablas de staging se realiza de forma secuencial debido a que son procesos que tardan más en completarse, así mismo se evita la creación de un SqlSensor para cada una de estas dado que es indispensable que se encuentren cargadas al momento de iniciar con la carga de las tablas de dimensión y hechos.


Grid
Graph
Calendar
Task Duration
Task Tries
Landing Times
Gantt
Details
Code
Audit Log
2023-05-29T22:43:05Z
Runs
25
Run
manual__2023-05-29T22:43:04.149651+00:00
Layout
Left > Right
Update
Find Task...
PythonOperator
SqlSensor
TriggerDagRunOperator
deferred
failed
queued
removed
restarting
running
scheduled
shutdown
skipped
success
up_for_reschedule
up_for_retry
upstream_failed
Auto-refresh
Task Group 2
Cargar tablas de Dimensión
Task_group_2
load_dimm_ambito_enseñanza
load_dimm_pais
load_dimm_rama_enseñanza
load_dimm_rango_edad
load_dimm_sexo
load_dimm_situacion_laboral
load_dimm_tipo_universidad
load_dimm_universidades
Task_group_1
trigger_imported_dag_staging_1
trigger_imported_dag_staging_2
trigger_imported_dag_staging_3
trigger_imported_dag_staging_4
trigger_imported_dag_staging_5
trigger_imported_dag_staging_6
check_data_fully_load
Task Group 1
Cargar tablas de Staging
Task_group_3
trigger_imported_dag_fact_tables
Task Group 3
Cargar tablas de Hechos

Conclusiones

- Apache Airflow es un orquestador de tareas muy potente que nos permite definir flujos de datos entre una gran cantidad de plataformas, ya sea on-premise o cloud, pero es importante tener en cuenta que la aplicación solo debe usarse para esto, orquestar tareas. Las cargas de trabajo pesadas y complejas, así como el almacenamiento de los datos, deben administrarse desde otros servicios, como EC2 + Spark o S3 para el caso de la nube AWS.

- A través de este reto pudimos poner en práctica y desarrollar distintas habilidades respecto al uso de la herramienta Apache Airflow, que son de gran relevancia dentro del rol de ingeniería de datos; siendo Airflow un software open source ampliamente usado y con múltiples capacidades para la creación de pipelines de datos.

- El uso de contenedores es una excelente manera de acelerar los ciclos de desarrollo y prueba de aplicaciones dado que permiten aprovisionar ambientes de manera rápida, sin tener complicaciones por versiones de software o librerías, así mismo evita tener que realizar distintas configuraciones a nivel del OS host para cada nueva implementación.

Anexos

- Repositorio de GIT: fork-dw-airflow-training
- Capturas de pantalla de la solución: Screenshots