**Documentación del Challenge de Taligent**

## **1. Introducción**

Este documento describe la instalación, configuración e integración de Apache Airflow, OpenMetadata y PostgreSQL en un entorno local utilizando Docker. Se mostrará la configuración de cada herramienta, la creación de DAGs en Airflow, la gestión de metadata en OpenMetadata y la integración con PostgreSQL.

## **2. Instalación y Configuració**

### **2.1 Instalación de Docker y Docker Compose**

Para gestionar los contenedores de cada servicio se utiliza **Docker Compose**.

explicacion de uso:

El uso de Docker Compose permite definir y gestionar todos los contenedores relacionados de una aplicación en un solo archivo de configuración (docker-compose.yml). En este archivo, puedes especificar cómo deben ejecutarse cada uno de los contenedores, sus variables de entorno, puertos expuestos, volúmenes, redes, etc. Esto facilita la gestión de configuraciones complejas.

instalar Docker y Docker Compose

verificar si la instalación fue con éxito

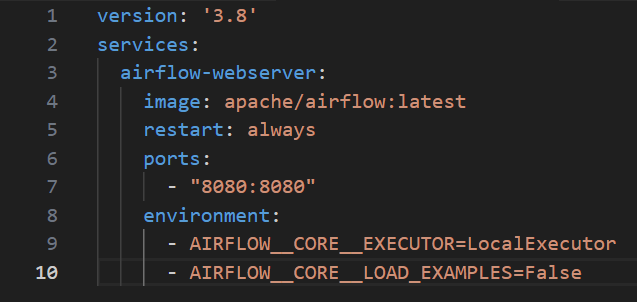
**docker --version**

**docker-compose --version**

### **3 Instalación de Apache Airflow**

se utiliza Docker para levantar un entorno de Airflow con los componentes básicos:

Clonar el repositorio oficial de Airflow o utilizar el siguiente docker-compose.yml:



Levantar los contenedores:

**docker-compose up -d**

### **Instalación de OpenMetadata**

Descargar el docker-compose.yml desde el repositorio de OpenMetadata.

iniciar OpenMetadata con:

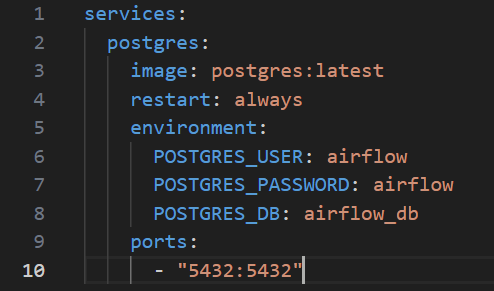
docker-compose up -d

Acceder a OpenMetadata en http://localhost:8585

### **Instalación y Configuración de PostgreSQL**

Se usa un contenedor Docker para PostgreSQL:

Agregar la configuración en docker-compose.yml:



Levantar PostgreSQL:

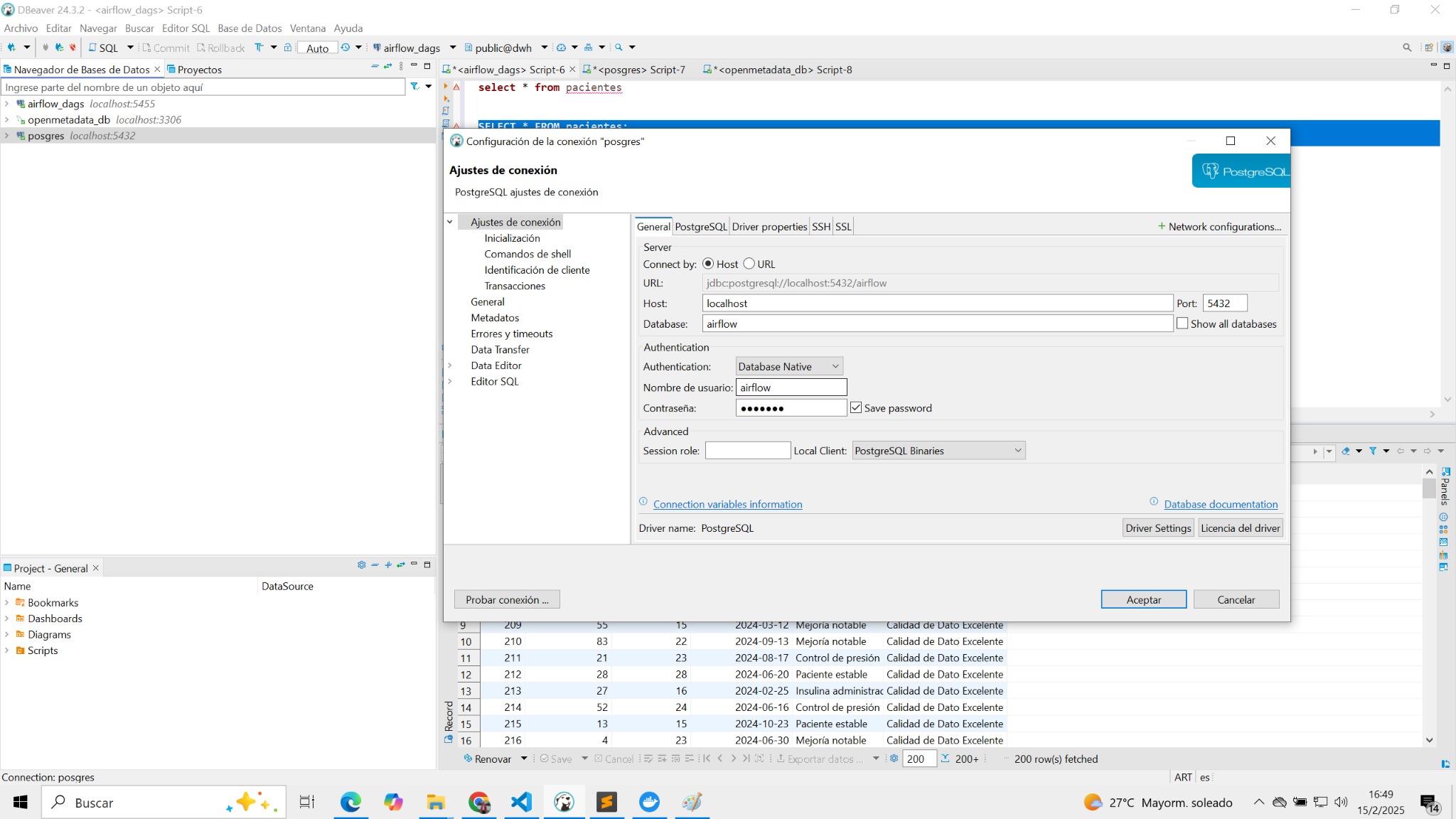
docker-compose up -d

Conectarse a la base de datos:

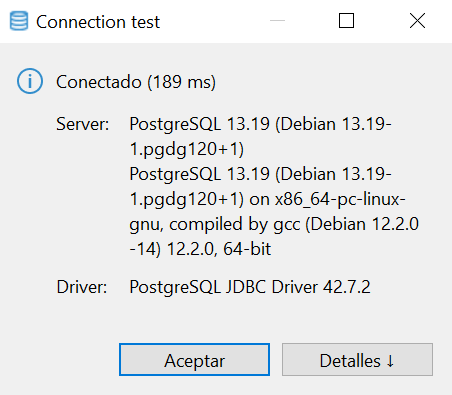
lo pueden hacer de esta manera :

docker exec -it postgres psql -U airflow -d airflow\_db

o tambien podemos conectarnos desde DBeaver:



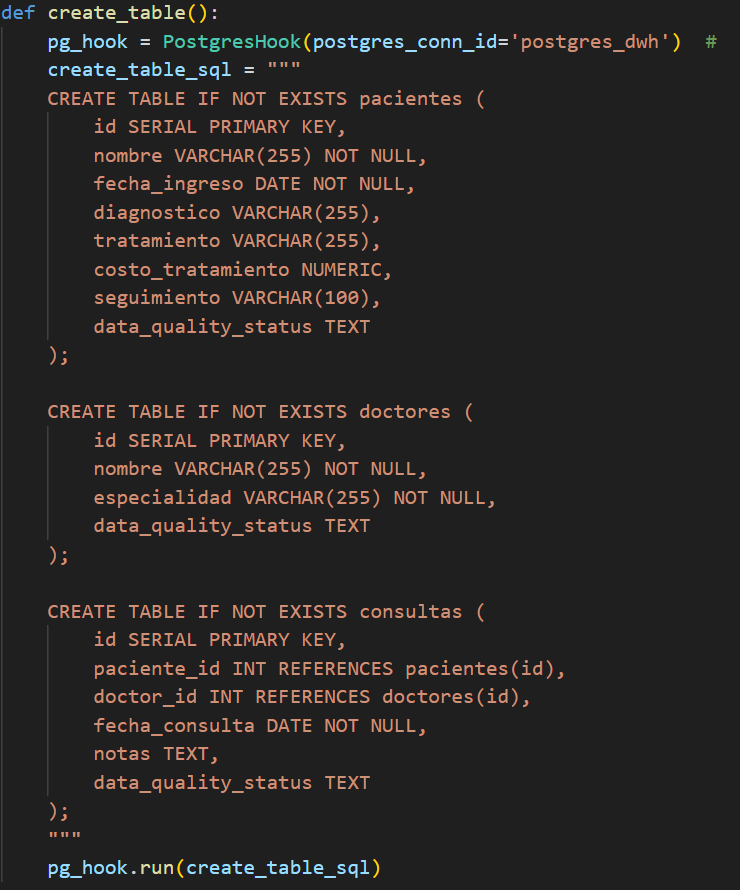
luego probamos la conexion:

****

## **3. Creación de DAGs en Airflow**

Funciones del proceso ETL (Extract, Transform, Load) :

primero creamos 3 tablas :

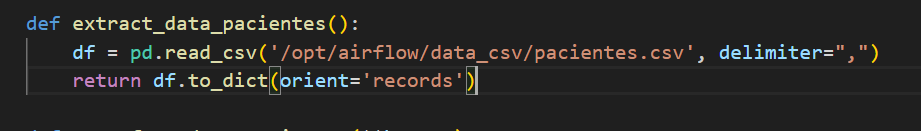


las cuales vamos a utilizar para los archivos csv: pacientes,doctores y consultas.

### **DAG de Extracción**

Funcion extract\_data\_pacientes:

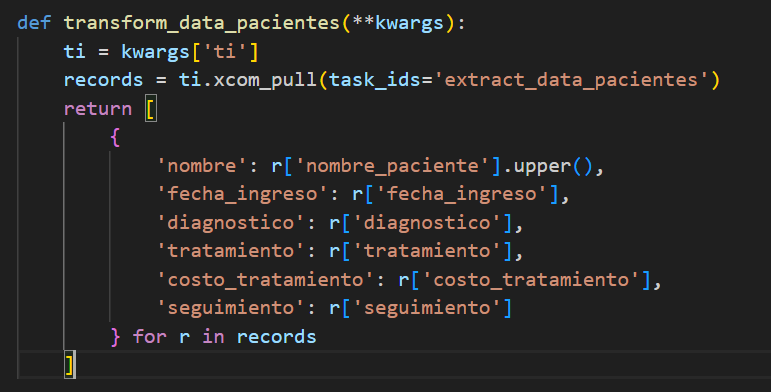
Se crean una funcion para extraer las informacion de un csv: en este caso seria el ejemplo para el csv pacientes, exactamente lo mismo seria , para los demas.



luego convierte el contenido en un formato de diccionario. Este formato facilita la transferencia de datos a las siguientes funciones.

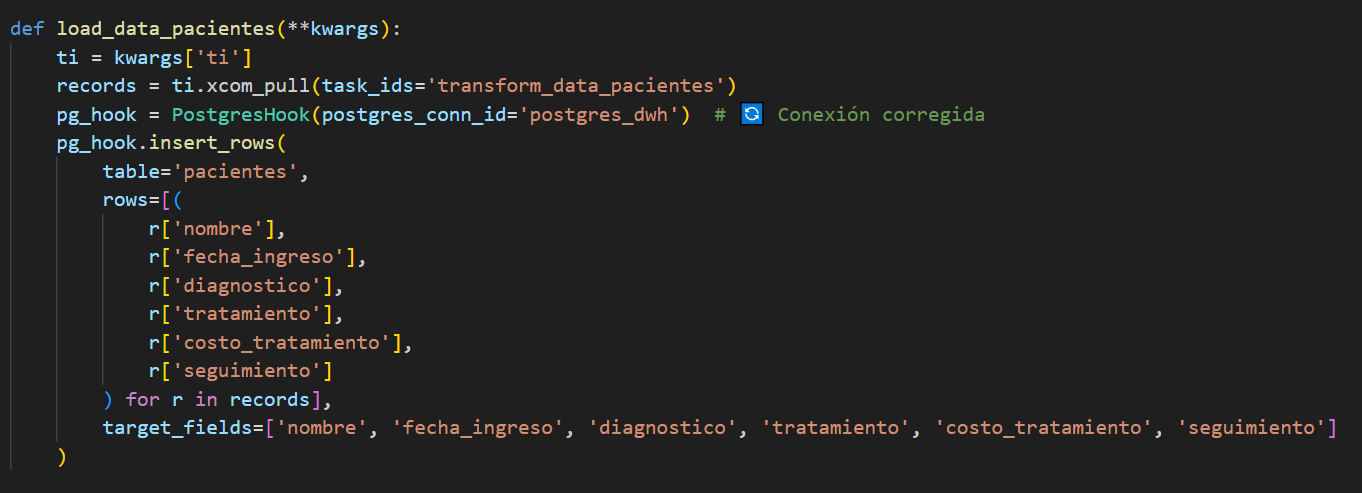
Funcion transform\_data\_pacientes:

Toma los registros extraídos y realiza una transformación en los datos. En este caso, convierte el nombre del paciente a mayúsculas y mapea el resto de los campos como están.



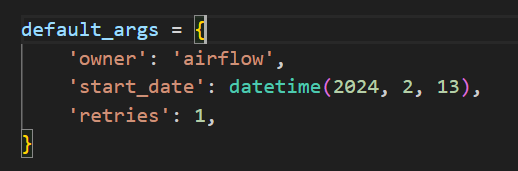
Funcion load\_data\_pacientes:

Toma los registros transformados y los inserta en la tabla pacientes de PostgreSQL usando el PostgresHook. Se realiza una inserción masiva en la tabla de la base de datos.



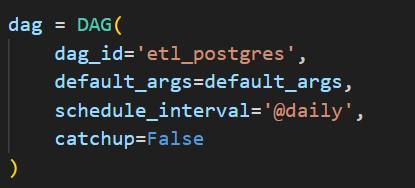
**Definición de DAG (Directed Acyclic Graph):**

Establece los argumentos predeterminados para el DAG. Aquí se define el propietario del DAG (owner), la fecha de inicio (start\_date) y el número de intentos (retries).

****

Definición de tareas:

**PythonOperator**: Es el operador que se usa para ejecutar funciones Python dentro de un DAG. Para cada uno de los pasos del ETL (crear tablas, extraer, transformar y cargar datos), se define una tarea utilizando este operador.

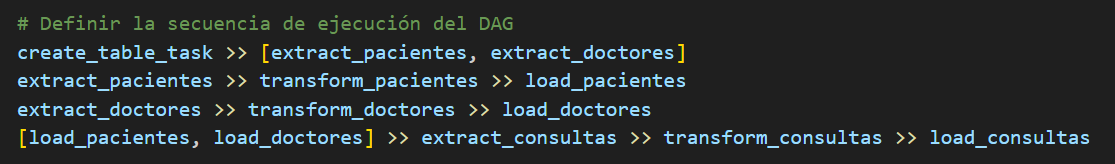


Aquí, la tarea create\_table\_task ejecutará la función create\_table.

y asi para las demas.

5. **Secuencia de ejecución del DAG**

Las tareas relacionadas con las consultas (extract\_consultas, transform\_consultas, load\_consultas) se ejecutan después de que se completen las tareas de carga de pacientes y doctores.

****

### **DAG de Carga**

**Definición de la función check\_and\_update\_data\_quality**

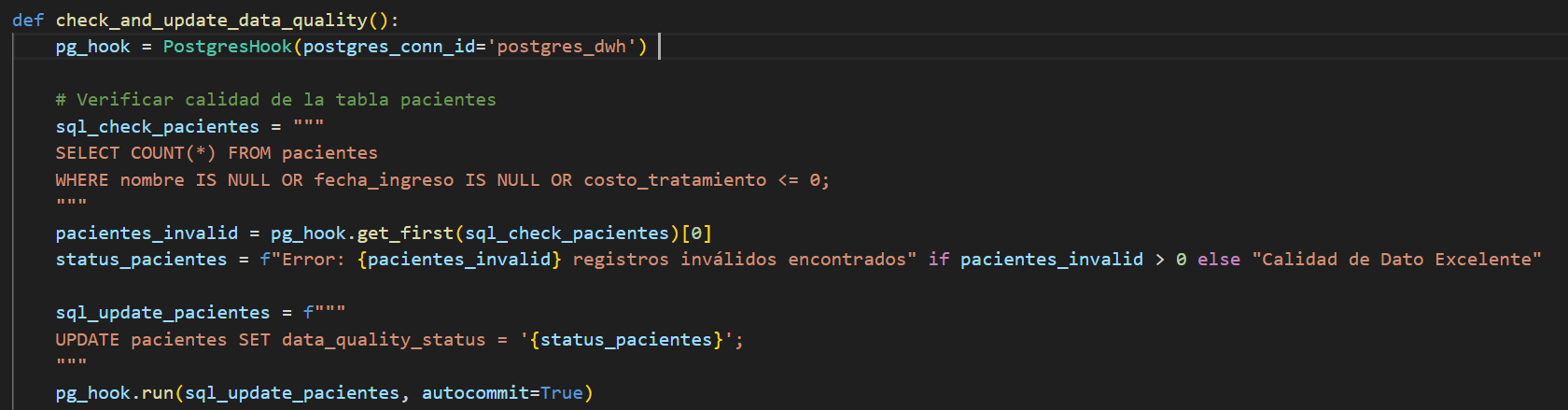
Esta es la función que realiza la verificación de la calidad de los datos en las tres tablas y actualiza la columna data\_quality\_status según los resultados.

Se realizan tres verificaciones, una para cada tabla (pacientes, doctores, y consultas):

* **pacientes**: Se verifica si los campos nombre, fecha\_ingreso y costo\_tratamiento son NULL o si costo\_tratamiento es menor o igual a 0. Si se encuentran registros inválidos, se cuenta cuántos hay y se crea un mensaje indicando el error. Si no hay registros inválidos, se indica que la calidad de los datos es excelente.

#### **Actualización de data\_quality\_status**

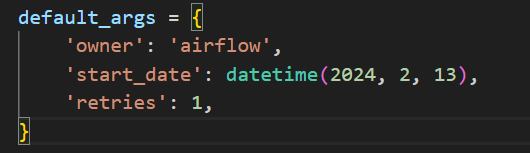
Después de verificar los registros inválidos en cada tabla, se actualiza la columna data\_quality\_status con el mensaje generado, ya sea de error o de calidad excelente. Esto se hace a través de una sentencia SQL de UPDATE.

****

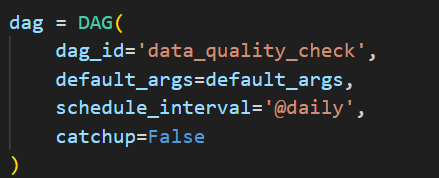
**Esto se repite para las otras dos tablas (doctores y consultas).**

**Definición del DAG en Airflow**

default\_args: Define los argumentos por defecto para el DAG. En este caso, se establece el propietario (owner), la fecha de inicio (start\_date), y el número de reintentos (retries).

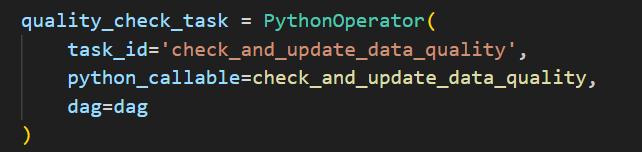
****

**dag: Se crea el DAG que define el flujo de trabajo. El dag\_id es data\_quality\_check, se ejecuta a diario (schedule\_interval='@daily') y no realiza "catchup" (es decir, no ejecuta tareas pasadas si el DAG se activa después de una fecha de inicio).**

****

**Definición de la tarea quality\_check\_task**

**quality\_check\_task: Define la tarea que ejecutará la función check\_and\_update\_data\_quality. Se utiliza un PythonOperator que llamará a la función cuando se ejecute el DAG.**

****

## **4. Integración de OpenMetadata con Airflow y PostgreSQL**

### **4.1 Configuración en OpenMetadata**

1. **Ir a http://localhost:8585/.**
2. **Crear una conexión con Airflow:**
   * **Connection Type: HTTP**
   * **Host: http://airflow-webserver:8080**
   * **Port: 8080**

**Agregar una conexión con PostgreSQL:**

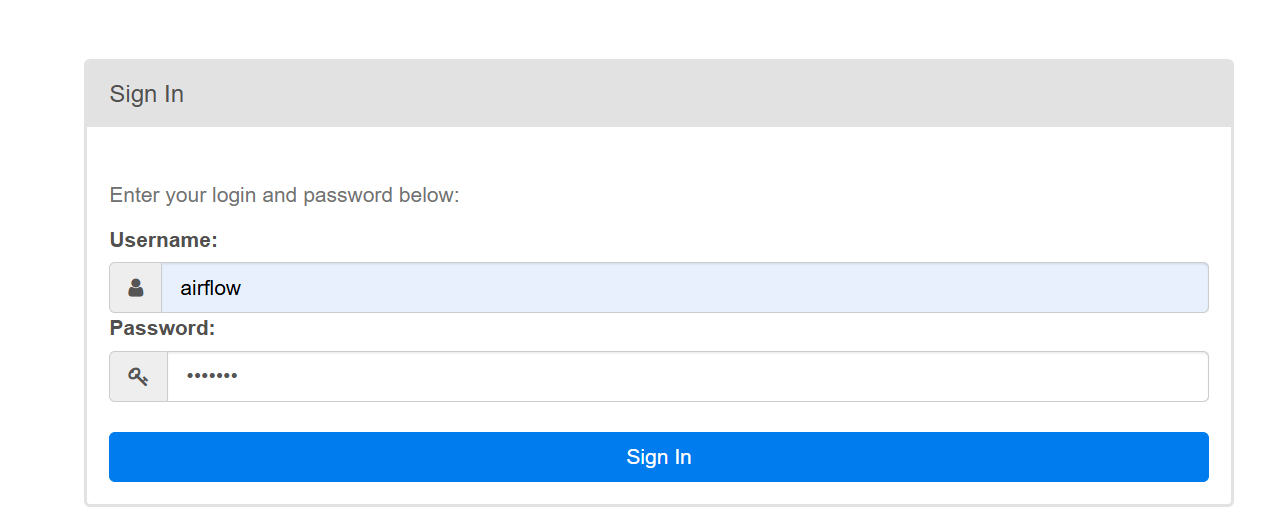
* **Host: postgres**
* **Port: 5432**
* **Database: airflow\_db**
* **Username: airflow**
* **Password: airflow**

### **4.2 Verificación de Integración**

**Ejecutar:**

**docker exec -it airflow-webserver curl** [**http://openmetadata\_server:8585/api/v1/services**](http://openmetadata_server:8585/api/v1/services)

**"Configuración de Airflow para la Ejecución de DAGs: Acceso y Primeros Pasos"**

****

**no logueamos en este caso usamos la contraseña que definimos**

**airflow para usuario y contraseña.**

### **Conexiones en Airflow**

**Airflow permite definir conexiones a servicios externos como bases de datos, APIs y otros recursos para que los DAGs puedan interactuar con ellos. Las conexiones son configuradas desde la interfaz de usuario de Airflow.**

****

### 1. Connection Id

* Descripción: Este es el identificador único para la conexión. Este Connection Id es usado para referirse a la conexión dentro de los DAGs de Airflow.
* Valor: En este caso, es postgres\_dwh. Este nombre debe ser único para cada conexión.

### 2. Connection Type

* Descripción: Define el tipo de conexión. Airflow soporta múltiples tipos de conexión como Postgres, MySQL, Http, etc.
* Valor: En este caso, se selecciona Postgres de la lista desplegable, ya que la conexión es a una base de datos PostgreSQL.

### 3. Description

* Descripción: Este campo es opcional y se utiliza para proporcionar una descripción adicional sobre la conexión.
* Valor: Puedes dejarlo en blanco o escribir una breve descripción que explique el propósito de esta conexión. Ejemplo: "Conexión a la base de datos de Data Warehouse".

### 4. Host

* Descripción: Aquí se coloca el nombre del host o la dirección IP del servidor donde se encuentra la base de datos PostgreSQL.
* Valor: En este caso, se ha establecido como postgres\_dwh, que parece ser el nombre de un servicio o un alias de un servidor que Airflow puede resolver. Si tu servidor tiene una dirección IP específica o un nombre de dominio, colócalo aquí.

### 5. Schema

* Descripción: Este campo es opcional y se utiliza para especificar el esquema de la base de datos donde se almacenarán los datos. Un esquema es una estructura que contiene tablas y objetos relacionados dentro de la base de datos.
* Valor: En este caso, se ha especificado dwh, que puede ser el nombre del esquema que contiene las tablas de tu Data Warehouse.

### 6. Login

* Descripción: Aquí debes colocar el nombre de usuario con el que Airflow se conectará a la base de datos PostgreSQL.
* Valor: En este caso, se usa dwh, lo cual es el nombre de usuario configurado para acceder a la base de datos.

### 7. Password

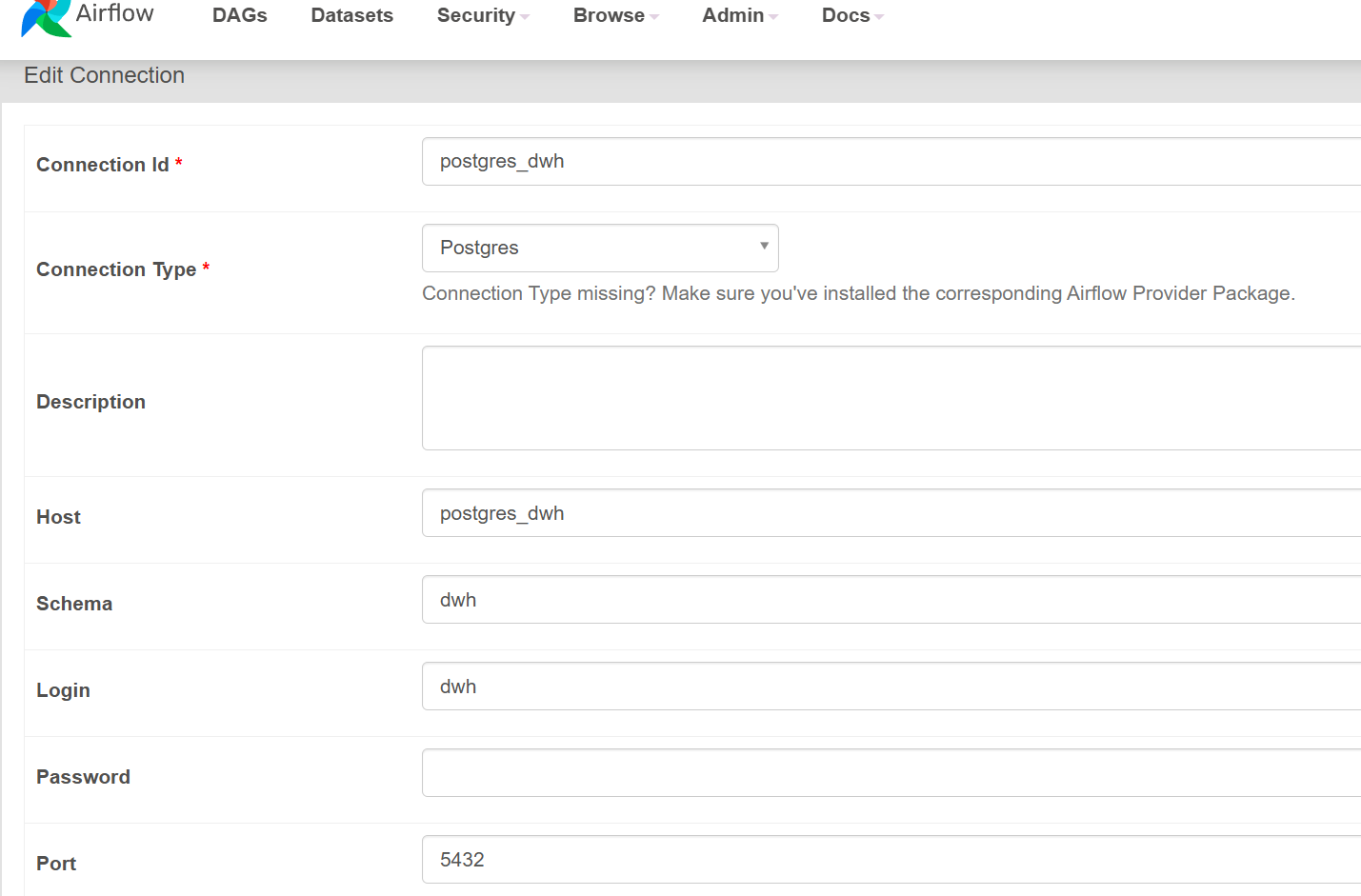
* Descripción: Este campo es donde se debe ingresar la contraseña asociada al nombre de usuario (Login) para autenticar la conexión a PostgreSQL.
* Valor: Debes ingresar la contraseña correspondiente al usuario dwh. Asegúrate de que esté segura y de que sea correcta.

### 8. Port

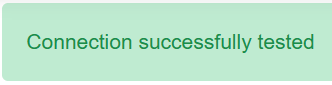
* Descripción: Este es el puerto en el que la base de datos PostgreSQL escucha las conexiones. El puerto predeterminado para PostgreSQL es 5432, que es el valor que aparece en este formulario.
* Valor: Si tu base de datos usa el puerto predeterminado, puedes dejar este valor como está (5432). Si utiliza un puerto personalizado, asegúrate de cambiarlo a la configuración correcta.

### 9. Test Connection

* Descripción: Al finalizar la configuración, es recomendable hacer clic en el botón Test Connection (que no aparece en la imagen, pero está disponible en la interfaz) para verificar si los parámetros de conexión son correctos y si Airflow puede conectarse con éxito a la base de datos.

****

**una vez que probamos todas las configuraciones probamos el test de conexion**

****

**y luego procedemos a guardar.**

### **1. Lista de DAGs**

**En la interfaz se muestran dos DAGs activos: data\_quality\_check y etl\_postgres. Los DAGs son representaciones de los flujos de trabajo que ejecutan tareas en Airflow.**

### **2. Estado de los DAGs**

* **data\_quality\_check: Tiene un total de 5 ejecuciones pasadas (Runs), y está configurado para ejecutarse de manera diaria (@daily). La última ejecución fue el 15 de febrero de 2025 a las 18:34:02.**
* **etl\_postgres: Tiene 3 ejecuciones pasadas (Runs), y también está configurado para ejecutarse de manera diaria (@daily). La última ejecución fue el 15 de febrero de 2025 a las 18:33:40.**

### **3. Acciones disponibles**

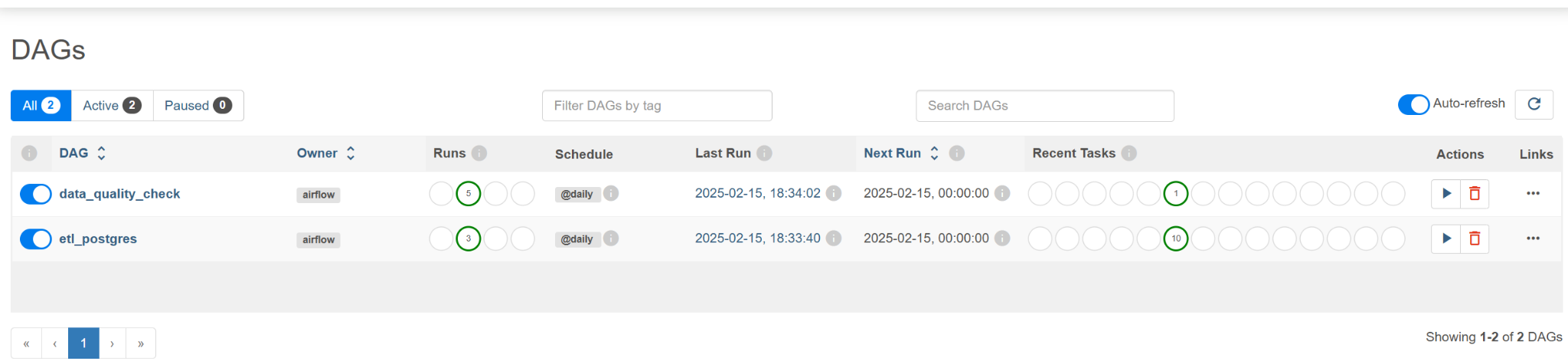
* **Cada DAG tiene un botón de acción:**
  + **Play Button : Permite ejecutar el DAG manualmente. Al hacer clic en este botón, Airflow iniciará la ejecución del DAG de acuerdo con su programación (o inmediatamente si se ejecuta manualmente).**
  + **Delete Button : Elimina el DAG, aunque esto es poco frecuente, ya que generalmente los DAGs se mantienen en el sistema.**

### **4. Detalles adicionales**

* **Schedule: Los DAGs están configurados para ejecutarse todos los días (@daily), lo que significa que Airflow ejecutará cada uno de estos DAGs automáticamente una vez al día.**
* **Last Run: Muestra la fecha y hora de la última ejecución exitosa del DAG.**
* **Next Run: Muestra cuándo está programado para ejecutarse el DAG nuevamente, en este caso, ambos están programados para ejecutarse a las 00:00:00 del 15 de febrero de 2025.**

### **5. Botones y filtros**

* **Auto-refresh: Esta opción permite que la vista se actualice automáticamente para mostrar el estado más reciente de los DAGs sin necesidad de recargar la página manualmente.**
* **Search DAGs: Un campo de búsqueda que permite filtrar los DAGs por su nombre o etiqueta.**

****

### 1. Barra de navegación superior

* Auto-refresh: El toggle que permite la actualización automática de la interfaz para reflejar los cambios de estado sin necesidad de recargar manualmente la página.
* Schedule: El DAG está configurado para ejecutarse diariamente (@daily).
* Next Run: El próximo run del DAG está programado para 2025-02-15, 00:00:00 UTC.

### 2. Resumen de los DAG Runs

* Total Runs Displayed: Se han mostrado 3 ejecuciones del DAG.
* Total success: Las 3 ejecuciones fueron exitosas.
* First Run Start: La primera ejecución del DAG comenzó el 2025-02-15 a las 16:50:59 UTC.
* Last Run Start: La última ejecución comenzó el 2025-02-15 a las 18:33:41 UTC.
* Max Run Duration: La duración máxima de ejecución fue de 5 minutos y 14 segundos.
* Mean Run Duration: La duración promedio de ejecución fue de 1 minuto y 53 segundos.
* Min Run Duration: La duración mínima fue de 12 segundos.

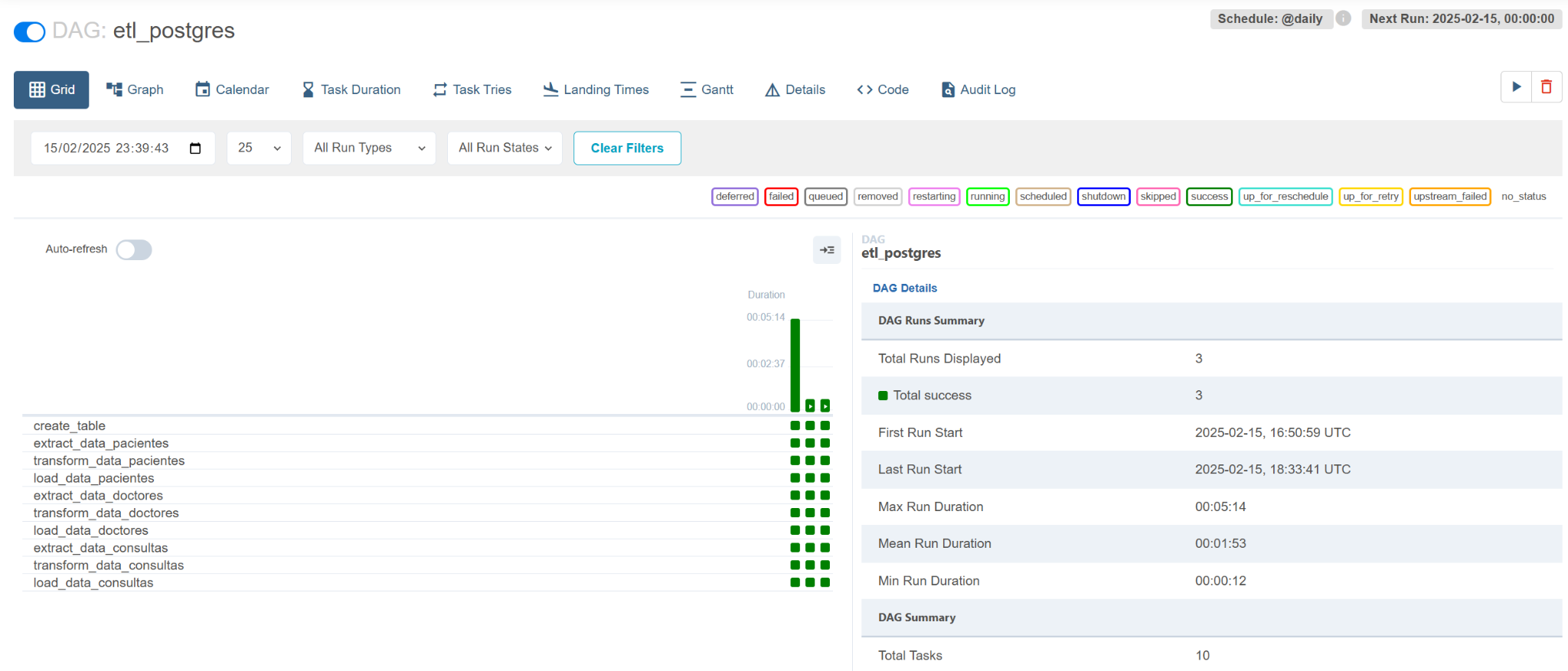
### 3. Vista de los tasks del DAG (Grid)

La tabla muestra una lista de todas las tareas que componen el DAG etl\_postgres. Cada fila corresponde a una tarea y muestra las siguientes columnas:

* Task Names: Los nombres de las tareas dentro del DAG, como create\_table, extract\_data\_pacientes, transform\_data\_pacientes, etc.
* Duración de las tareas: Al lado de cada nombre de tarea, se visualiza la duración en un gráfico de barras. Las barras verdes indican que las tareas se ejecutaron exitosamente. En este caso, la tarea create\_table duró 5 minutos y 14 segundos, mientras que otras tareas, como extract\_data\_pacientes, transform\_data\_pacientes, etc., tienen una duración considerablemente menor.
* Color de la barra: Las barras verdes indican que las tareas fueron exitosas en todas las ejecuciones. No hay tareas fallidas o en estado de error (lo cual sería indicado con colores diferentes como failed o up\_for\_retry).

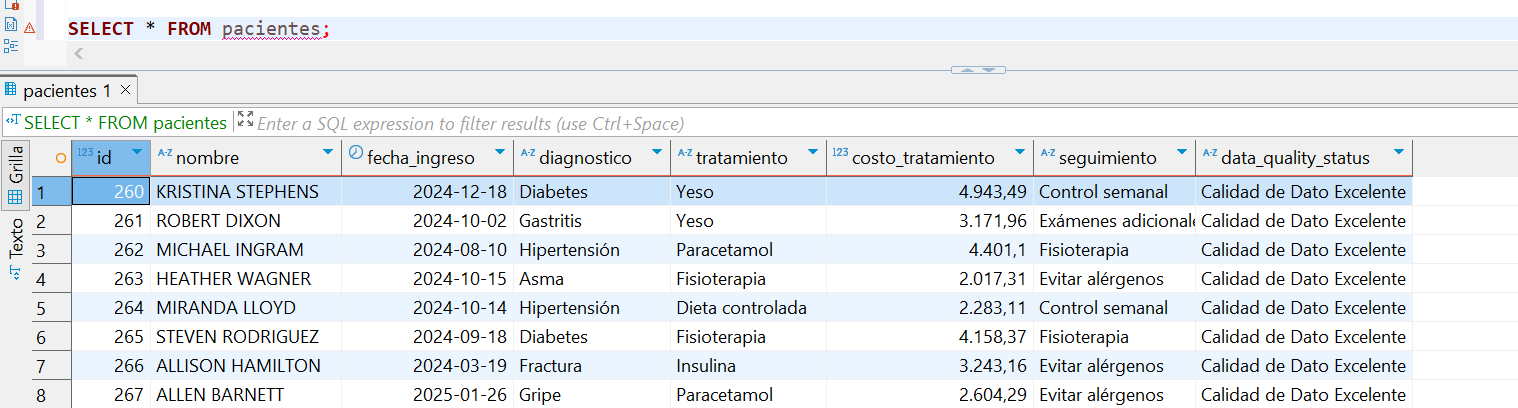
### 4. Detalles adicionales

* DAG Summary: Se resumen estadísticas sobre las tareas de este DAG:
  + Total Tasks: El DAG tiene un total de 10 tareas.
  + Las demás métricas de rendimiento son calculadas automáticamente, proporcionando una visión rápida sobre el rendimiento y la eficiencia del DAG.

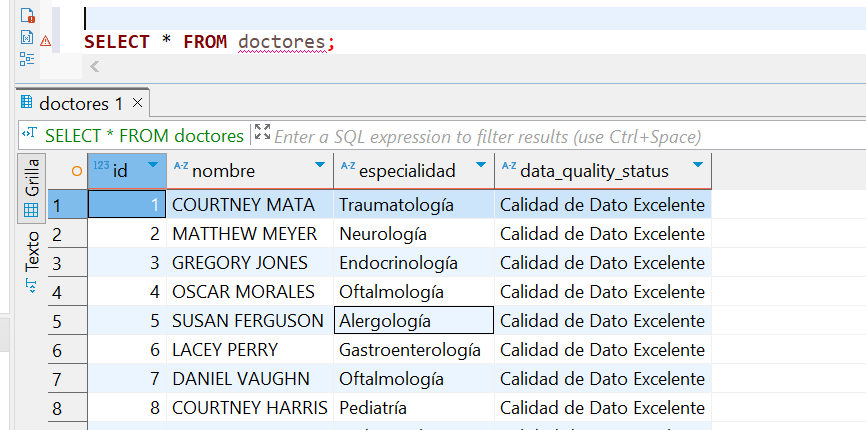
****

**luego que vemos que corrió bien el dag, revisamos una consulta si creo la tabla en sql.**

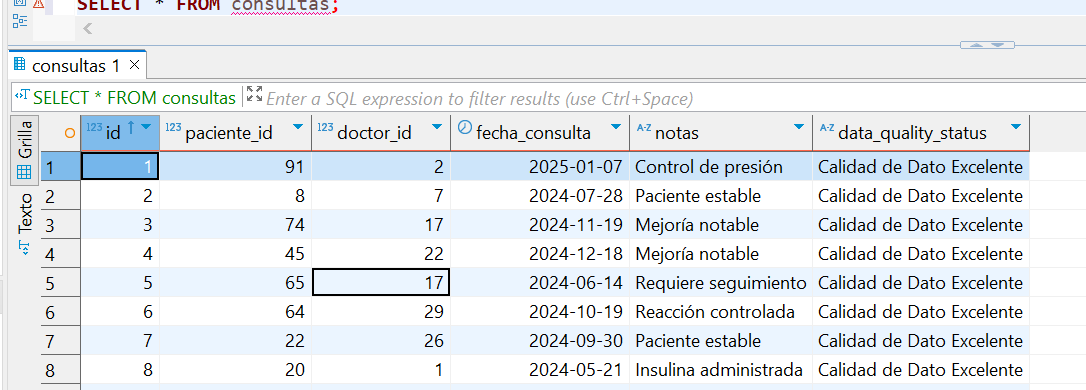
**Como vemos creo bien la tabla pacientes.**

****

**luego revisamos si tambien creo la tabla de doctores y contiene todos los datos que insertamos.**

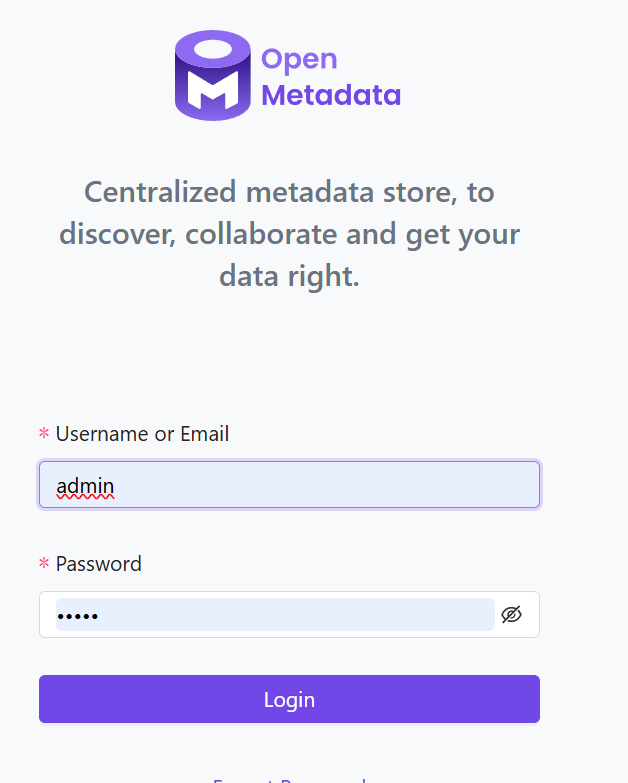
****

**por ultimo controlamos la ultima tabla creada**

****

### **5.3 Verificación en OpenMetadata**

1. **Ir a http://localhost:8585/.**
2. **Buscar los DAGs en la sección de Pipelines.**
3. **Verificar la metadata de PostgreSQL en la sección de Bases de Datos.**

****

### 1. Servicios de Bases de Datos (Databases)

En la sección de Databases (Bases de datos), se pueden ver los servicios que están configurados en OpenMetadata para la ingesta de metadatos relacionados con las bases de datos. Aquí se muestra el servicio postgres\_dwh, que está asociado a una base de datos PostgreSQL.

### 2. Información del Servicio

* Nombre del Servicio: postgres\_dwh es el nombre de la base de datos configurada en OpenMetadata.
* Descripción: Actualmente, no hay ninguna descripción asignada a este servicio (indicado como "No description").
* Tipo: El tipo del servicio es Postgres, lo que indica que se trata de una base de datos PostgreSQL.

### 3. Acciones Disponibles

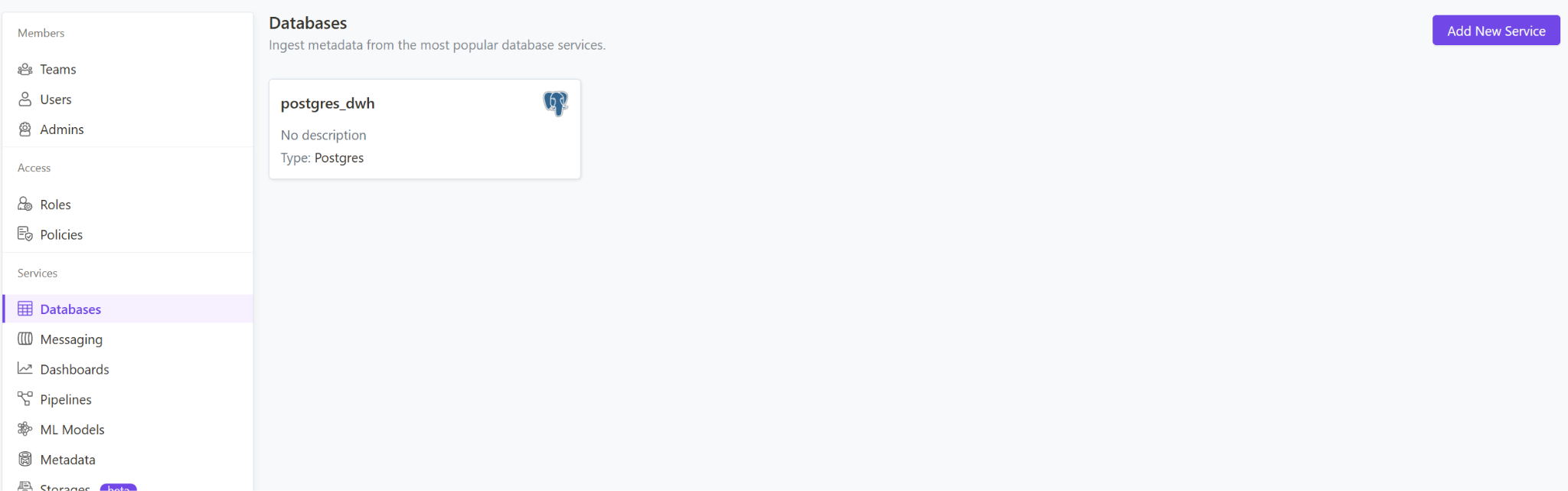
* Add New Service: El botón "Add New Service" permite agregar nuevos servicios de base de datos o integraciones dentro de OpenMetadata.

### 4. Menú Lateral (Izquierda)

* Miembros: Muestra las secciones de miembros, roles y políticas dentro de OpenMetadata. Aquí se gestionan los accesos y permisos.
* Servicios: Aquí se encuentran diferentes tipos de servicios relacionados con bases de datos, mensajería, tableros de control, pipelines, y más.
* Notificaciones: Sección donde puedes configurar alertas y feeds de actividad para mantenerte informado sobre los cambios y eventos.
* Tablas: Acceso a las tablas almacenadas y configuradas dentro de OpenMetadata.

### 5. Interfaz y Búsqueda

* Buscar: En la parte inferior, puedes usar el cuadro de búsqueda para localizar rápidamente cualquier servicio, tabla o recurso dentro de OpenMetadata.

****

### 1. Ingestions Configuradas

En la tabla de Ingestions se muestran tres procesos de ingesta configurados para el servicio postgres\_dwh:

* postgres\_dwh\_lineage:
  + Type: Lineage (Línea de tiempo o linaje de datos).
  + Recent Runs: La última ejecución de esta ingesta fue exitosa (Success).
  + Actions: Las acciones disponibles para esta ingesta incluyen Run (Ejecutar), Re Deploy (Volver a desplegar), Pause (Pausar), Edit (Editar), Delete (Eliminar), Kill (Cancelar), y Logs (Ver registros de ejecución).
* postgres\_dwh\_metadata\_01:
  + Type: Metadata (Ingesta de metadatos).
  + Recent Runs: Esta ingesta también tuvo una ejecución exitosa.
  + Actions: Las mismas acciones están disponibles para este proceso, que incluyen ejecutar, pausar, editar, eliminar, etc.
* postgres\_dwh\_profiler\_01:
  + Type: Profiler (Ingesta de perfiles de datos, utilizada para análisis de calidad o estadísticas de los datos).
  + Recent Runs: Esta ingesta también ha tenido ejecuciones exitosas.
  + Actions: Al igual que las otras, se pueden realizar las mismas acciones como pausar, eliminar, editar, entre otras.

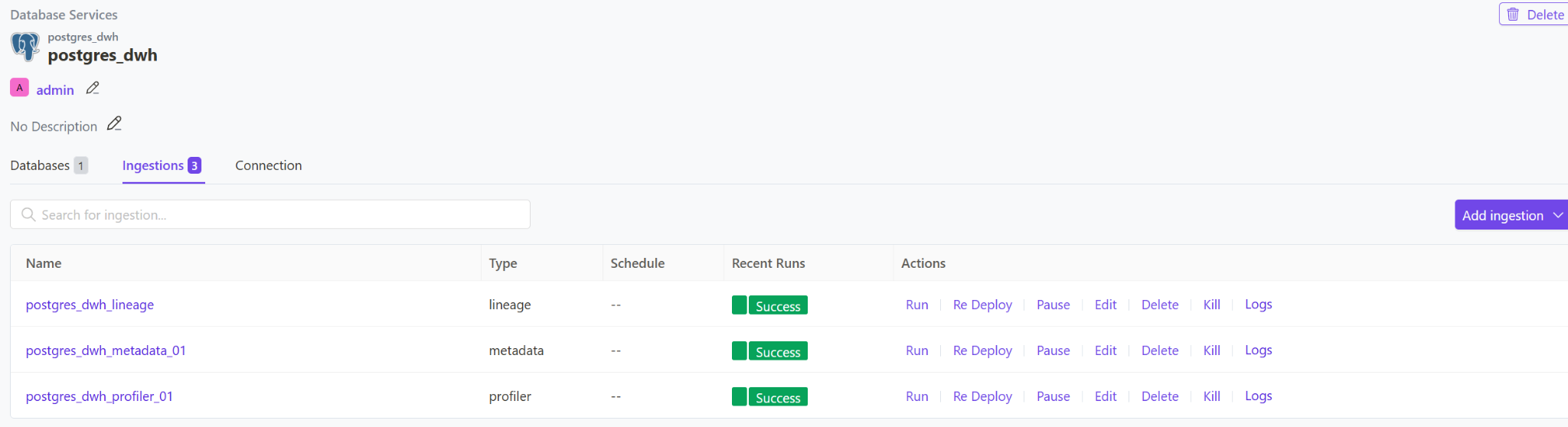
### 2. Acciones Disponibles

Para cada ingesta, hay varias acciones que puedes tomar:

* Run: Ejecutar la ingesta.
* Re Deploy: Volver a desplegar la ingesta, útil si ha habido cambios en la configuración.
* Pause: Pausar la ingesta si no deseas que se ejecute temporalmente.
* Edit: Modificar la configuración de la ingesta, por ejemplo, para ajustar la frecuencia o el origen de los datos.
* Delete: Eliminar la ingesta si ya no es necesaria.
* Kill: Detener una ingesta que se está ejecutando.
* Logs: Ver los registros de ejecución de la ingesta para revisar el historial o los detalles de la ejecución.

### 3. Agregar una Nueva Ingesta

* Add ingestion: Hay un botón "Add ingestion" que permite agregar nuevas ingestas para este servicio. Esto sería útil si deseas configurar nuevas ingestas desde la base de datos postgres\_dwh hacia OpenMetadata.

****

### Connection Scheme

* Descripción: Aquí se selecciona el esquema de conexión que se utilizará para conectarse a PostgreSQL.
* Valor: El valor seleccionado es postgresql+psycopg2, que es un esquema estándar de conexión utilizado en PostgreSQL a través de la biblioteca psycopg2 para Python.

### 2. Username

* Descripción: Este es el nombre de usuario utilizado para autenticar la conexión a la base de datos.
* Valor: El valor configurado es dwh, que es el nombre de usuario utilizado para acceder a la base de datos PostgreSQL.

### 3. Password

* Descripción: Aquí se ingresa la contraseña del usuario configurado en la base de datos PostgreSQL.
* Valor: El valor se oculta por razones de seguridad, pero aquí se ingresaría la contraseña correspondiente al usuario dwh.

### 4. Host and Port

* Descripción: Se especifica el host (servidor) y el puerto al que OpenMetadata debe conectarse para acceder a la base de datos PostgreSQL.
* Valor: El host es postgres\_dwh, que probablemente sea un alias o nombre de servicio que apunta al servidor de base de datos. El puerto no está especificado directamente en la imagen, pero típicamente el valor para PostgreSQL es el puerto 5432.

### 5. Database

* Descripción: Se indica la base de datos específica en PostgreSQL a la que se debe conectar OpenMetadata.
* Valor: El valor aquí es dwh, que es el nombre de la base de datos configurada en PostgreSQL.

### 6. SSL Mode

* Descripción: Este campo permite configurar si la conexión a PostgreSQL utilizará SSL (Secure Sockets Layer).
* Valor: El valor está configurado como disable, lo que significa que la conexión no usará SSL.

### 7. SslConfig

* Descripción: Aquí se pueden agregar configuraciones adicionales relacionadas con SSL, como certificados de cliente o configuraciones avanzadas para la conexión segura.
* Valor: Este campo parece no tener configuraciones adicionales en la imagen.

### 8. ValidateSSLClientConfig

* Descripción: Un parámetro adicional que valida la configuración SSL del cliente. Este campo no está configurado en la imagen.
* Valor: No está especificado en la imagen.

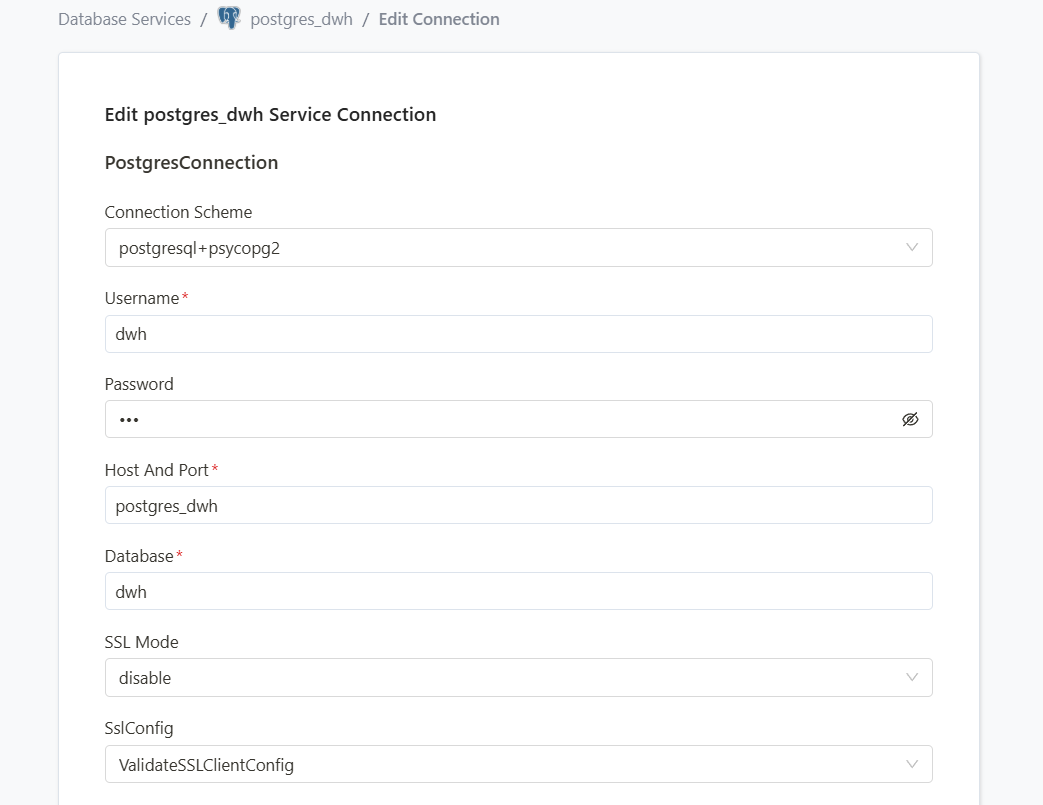
### 9. Información sobre el Profiler y la Calidad de los Datos

En la parte derecha de la pantalla se muestra información relacionada con el uso de profiler y las pruebas de calidad de los datos:

* Profiler & Data Quality: Si se van a ejecutar flujos de trabajo de profiler o pruebas de calidad de datos, el usuario debe tener permisos SELECT en las tablas y esquemas involucrados.
* Usage and Lineage: Se explica cómo OpenMetadata extrae la información de linaje y uso desde PostgreSQL, basándose en la tabla pg\_stat\_statements.
* Recomendación: OpenMetadata extrae datos de la tabla pg\_stat\_statements de PostgreSQL, que contiene estadísticas de las consultas realizadas, como su ejecución y rendimiento. Este sistema no incluye un tiempo específico de ejecución, pero muestra todas las consultas realizadas, por lo que se puede restablecer con el comando pg\_stat\_statements\_reset().

### 10. Acciones disponibles

* Aunque no se mencionan en la imagen, es común que haya botones como Save (Guardar) para aplicar la configuración y Cancel para descartar los cambios.

****

### Tablas Mostradas

Se muestran cuatro tablas bajo el esquema public de la base de datos dwh:

* pacientes:
  + Tipo: Regular
  + Columnas: 8 columnas.
  + No tiene consultas asociadas y no se ha proporcionado una descripción.
  + En la columna de Profiler & Data Quality, se indica que no se ha habilitado el profiler para esta tabla.
* doctores:
  + Tipo: Regular
  + Similar a la tabla pacientes, sin descripción y sin consultas asociadas.
* pg\_stat\_statements:
  + Tipo: Vista (View).
  + Este es un tipo especial de tabla que almacena estadísticas sobre las consultas ejecutadas en PostgreSQL. Se utiliza para monitorear el rendimiento de la base de datos.
* consultas:
  + Tipo: Regular
  + Similar a las otras tablas en cuanto a la falta de descripción.

### 3. Detalles de la Tabla pacientes

Al hacer clic en la tabla pacientes, se despliegan los detalles de la estructura de la tabla:

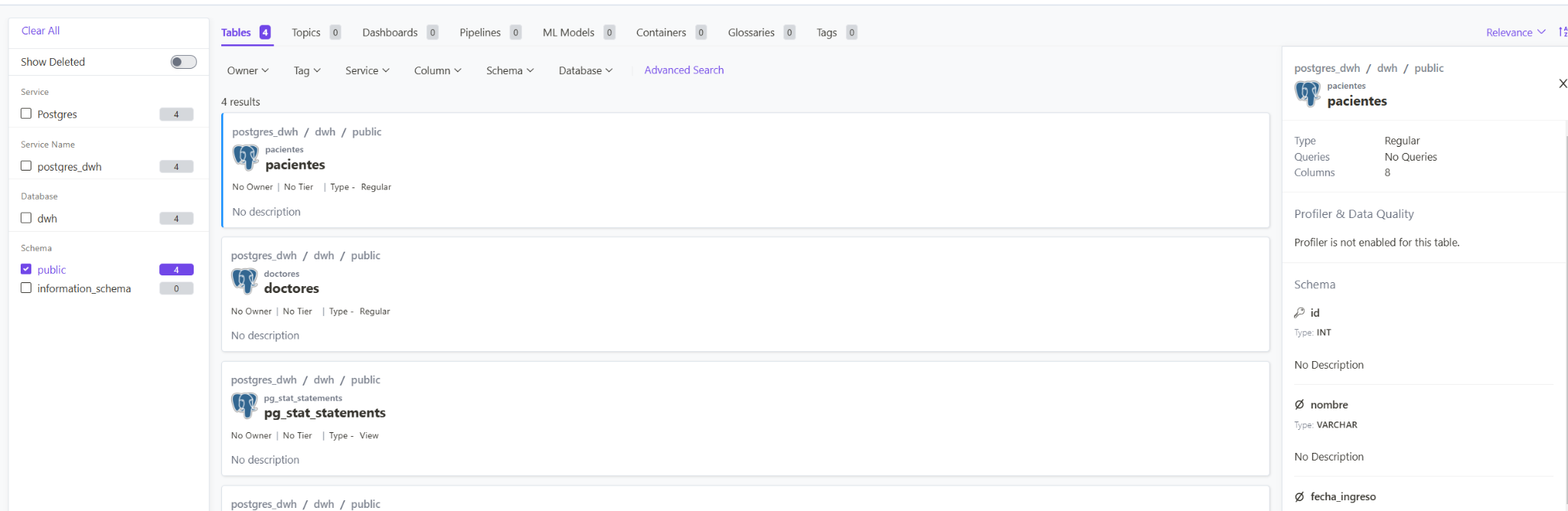
* Nombre de las columnas:
  + id: Tipo INT (entero).
  + nombre: Tipo VARCHAR (cadena de caracteres variable).
  + fecha\_ingreso: Tipo DATE (fecha).
  + diagnostico: Tipo VARCHAR (cadena de caracteres).
  + tratamiento: Tipo VARCHAR (cadena de caracteres).
* En cada columna se indica el tipo de datos y se deja claro que no se ha proporcionado una descripción adicional para las columnas.

### 4. Acciones Disponibles

* Aunque no se muestran explícitamente en la imagen, es probable que en la interfaz también se pueda realizar acciones como editar, eliminar o ver más detalles de cada tabla. Estas opciones están disponibles para gestionar las tablas de manera eficiente.

### 5. Otros Detalles

* La interfaz de OpenMetadata permite explorar las tablas de la base de datos y ver sus metadatos, como las columnas y tipos de datos, así como la configuración de calidad de datos (Profiler).
* Profiler & Data Quality: Se menciona que no se ha habilitado el profiler para la tabla pacientes. El profiler es una herramienta para analizar la calidad de los datos, asegurando que los datos en las tablas sean precisos y completos.

****

## **Conclusiones**

**Este documento describe la implementación de Airflow, OpenMetadata y PostgreSQL usando Docker. Se crearon DAGs para ETL, se configuró OpenMetadata y se verificó la integración. Se logró una solución funcional para la gestión de flujos de trabajo y gobierno de datos.**