DBT

- diferencias entre dbt core, dbt cloud free y dbt cloud de pago
- Paso a paso de un proyecto dbt + postgre
- Qué es dbt y para qué nos sirve (demostrado con el ejemplo anterior)?

INTRODUCCIÓN

Data Build Tool(dbt) es una herramienta de análisis de datos open source que permite a los equipos de análisis y datos transformar, documentar y orquestar datos de una manera eficiente.

Se enfoca principalmente en la transformación de datos, toma datos en crudo de diferentes fuentes, los limpia y los transforma. Posteriormente los entrega en un formato estructurado para que los analistas o científicos de datos puedan trabajar con ellos.

DBT permite trabajar con una amplia variedad de siestemas de almacenamiento de datos, como bases de datos relacionales, cloud, data warehouse, etc.

DBT se escribe en SQL y permite a los equipos de análisis colaborar en el código, documentar y automatizar la transformación de datos.

TARIFAS

A continuación, se muestran las principales diferencias entre DBT core, DBT Cloud Free y DBT Cloud de pago:

- DBT Core: Versión gratuita, puede ser instalador localmente en un ordenador o servidor. Incluye las funciones esenciales de DBT como transformación de datos, testeo y orquestación de tareas. También dispone de una comunidad activa para proporcionar soporte.

- DBT Cloud Free: Versión gratuita de DBT Cloud, plataforma que ofrece herramientas de orquestación, programación, ejecución y monitoreo de flujos de trabajo de DBT. Al ser gratuita tiene una limitación en cuanto a la cantidad de proyectos y ejecuciones que pueden realizar. Es la ideal para probar y familiarizar.
- DBT Cloud de pago: Versión de pago, ofrece más funcionalidades en comparación a la gratuita y más capacidad de procesamiento de datos. Incluye funciones avanzadas como la integración con otras herramientas de análisis de datos, soporte a múltiples data warehouse, programación avanzada de flujos y el monitoreo y análisis en tiempo real. Es ideal para usuarios que necesitan trabajar con grandes volúmenes de datos y equipos de análisis de datos más grandes y complejos.

Como podemos ver, tenemos dos versiones gratuitas, DBT Core y DBT Cloud Free. La primera es una herramienta de línea de comandos, que permite transformar datos, mientras que la segunda es una plataforma que ademas de permitir transformar datos, también tiene un seguimiento de versiones (gracias a su compenetración con Git), programación de tareas, alertas, etc. Además DBT Cloud Free no hay que tenerlo instalado en nuestra máquina, a diferencia de DBT Core que si que es necesario tenerlo instalado. El hecho de ser una plataforma en línea, nos permite ejecutar DBT Cloud Free desde cualquier dispositivo, mientras que DBT Core solo podremos hacerlo desde la máquina local.

CREAR PROYECTO DBT

Para crear un proyecto dbt tenemos que tener instalado Python en una versión superior a la 3.5, y también nuestra BBDD Postgre SQL, la cual instalaremos desde su propia página web: https://www.postgresql.org/download/

Una vez hemos descargado el instalador de Postgre SQL para nuestro SSOO, durante el proceso de instalación nos va a pedir que asignemos diferentes nombres de usuario y contraseñas para poder acceder a la base de datos.

A partir de aquí, deberemos instalar dbt. Para nuestro sistema operativo, Windows, lo vamos a hacer desde el cmd o línea de comandos, ejecutando pip install dbt. Al hacer esto nos indicará que debemos especificar un determinado paquete para instalar dbt, por lo que haremos pip install dbt-postgres.

Para la creación de un proyecto deberemos ejecutar dbt init "ProjectName", y nos creará una carpeta del proyecto en el directorio donde nos encontrábamos.

Alli deberemos configurar el archivo profiles.yml para adaptarlo a nuestra base de datos PostgreSQL con el siguiente codigo:

```
company-name:
  target: dev
  outputs:
    dev:
      type: postgres
      host: [hostname]
      user: [username]
      password: [password]
      port: [port]
      dbname: [database name] # or database instead of dbname
      schema: [dbt schema]
      threads: [optional, 1 or more]
      [keepalives_idle](#keepalives_idle): 0 # default 0, indicating the system default. See below
      connect_timeout: 10 # default 10 seconds
      [retries](#retries): 1  # default 1 retry on error/timeout when opening connections
      [search_path](#search_path): [optional, override the default postgres search_path]
      [role](#role): [optional, set the role dbt assumes when executing queries]
      [sslmode](#sslmode): [optional, set the sslmode used to connect to the database]
```

https://docs.getdbt.com/reference/warehouse-setups/postgres-setup

A la hora de crear un proyecto, o utilizar cualquier comando "dbt" puede dar alguna clase de problemas relacionados con no poder ejecutar dbt. Para comprobar que este paquete se ha instalado correctamente ejecutaremos:
pip list -> Para ver todos los paquetes
pip show dbt-postgres -> Para ver informacion sobre ese paquete
pip show dbt-core ->Para ver informacion sobre ese paquete

Si estos se encuentran instalados, deberemos añadir a nuestras variables del sistema la ruta de la carpeta Scripts o bin de nuestra version de Python instalada. Para concoer esta ruta ejecutaremos: "python -m site --user-base."

Ahora para añadir esta ruta buscaremos en el buscador de Windows:
- Editar las variables de entorno del sistema
    - Variables de entorno
        - "Editar", en Variables del sistema, variable Path
            - Nueva (Añadir Ruta) y aceptar ventanas

Al crear un proyecto, deberemos tener una carpeta dentro de nuestro usuario (C:\Users\<Nombre_De_Usuario>\.dbt), donde estará el archivo de perfil, "profiles.yml". También se nos crea una carpeta con el nombre del proyecto que le hemos dado en el repositorio desde que lanzasemos el comando de "dbt init nombre_de_proyecto".

Esta carpeta la abriremos con un editor de código como puede ser Visual Studio Code, y abriremos el fichero "<Nombre_Del_Proyecto>.yml". Aqui se encontrará toda la configuración del proyecto, en nuestro caso introduciremos el codigo anterior ya que es el adaptado al perfil de postgres.

La linea primera del "profiles.yml" donde dice company-name, debe coincidir con la linea del .yml del proyecto que dice profile, en este caso en el archivo de configuración del proyecto deberemos ver:
profile: 'company-name'

Una vez introducidos todos los datos correctamente (host, usuario, nombre de base de datos, contraseña, etc.), abriremos un terminal desde VS Code, dandole en la opción del menú superior Terminal > Nuevo Terminal, y escribiremos "dbt debug". De esta forma podremos comprobar que dbt y postgre se han conectado correctamente o por el contrario no se ha podido establecer la conexión.

Si nos aparecen las siguientes líneas será que todo ha ido de forma exitosa:

"Connection test: [OK connection ok]

All checks passed! "

AIRBYTE

Plataforma de integración de datos open source, que permite a las empresas mover datos de diversas fuentes a una amplia variedad de destinos de datos en tiempo real. Se enfoca en que la integración de datos sea fácil, rápida y confiable.

La instalación de Airbyte se puede realizar ya sea en máquinas locales, virtuales como en Docker o Kubernetes, etc. Airbyte ofrece instrucciones detalladas para mucha variedad de instalaciones.

A diferencia de otras herramientas de integración de datos, Airbyte ofrece:

- Código abierto
- Facilidad de uso
- Conectores preconstruidos
- Escalabilidad
- Comunidad

la versión de pago de Airbyte ofrece características adicionales y un soporte técnico para las empresas que necesitan integrar datos de manera más compleja y segura. La versión gratuita de Airbyte es útil para proyectos más pequeños o para aquellas personas que buscan una solución de integración de datos básica y gratuita.

INSTALACIÓN AIRBYTE

Primero de todo va a ser instalar docker desktop, a través de su página web oficial: https://www.docker.com/get-started/

Una vez tengamos instalado esto, vamos a proceder a instalar Airbyte, para ello iremos a su repositorio del código, ya que es un sistema de código abierto: https://github.com/airbytehq/airbyte. Haremos un clone de dicho repositorio en una carpeta de nuestra máquina local a través del comando "git clone https://github.com/airbytehq/airbyte", y a continuación ejecutaremos "docker-compose up". Esto comenzará a montar airbyte, y tardará unos minutos.

Ahora para acceder a la interfaz de la WebApp de Airbyte, iremos a nuestro Docker Desktop y desplegaremos la pestaña que dice airbyte, allí podremos ver diferentes servicios que se encuentran corriendo. El que nos interesa es el de Webapp, donde podremos ver el puerto que esta utilizando y poder abrirlo en nuestro navegador, aunque por defecto suele utilizar el 8000 (http://localhost:8000).

Aqui entraremos con nuestro correo y podremos ver una interfaz sencilla donde nos permite conectar una fuente y un destino, por lo que vamos a seguir los siguientes pasos:

- Lanzar contenedores Postgres origen y destino

Ejecutaremos los siguientes comandos:

"docker run --rm --name airbyte-source -e POSTGRES_PASSWORD=password -p 2000:5432 -d debezium/postgres:14"

Crea una fuente en el puerto 2000

"docker run --rm --name airbyte-destination -e POSTGRES_PASSWORD=password -p 3000:5432 -d debezium/postgres:14"

Crea un destino en el puerto 3000

- Crear tabla

"docker exec -it airbyte-source psql --username=postgres"
Abriremos psql para poder lanzar querys

"CREATE TABLE cdc_tut(
id integer PRIMARY KEY,
name VARCHAR(200));"
Crea tabla cdc_tut

- Insertar datos

"INSERT INTO cdc_tut(id, name) VALUES(1, 'A1 CDCSyn');"

"INSERT INTO cdc_tut(id, name) VALUES(2, 'A2 CDCSyn');"

- Configurar CDC replicación

Primero se crea un slot de replicación ejecutando:
"SELECT pg_create_logical_replication_slot('airbyte_tut_slot', 'pgoutput');"

Desoues creamos una PUBLICATION:
"CREATE PUBLICATION cdc_tut_pub FOR TABLE cdc_tut;"

- Configurar origen y destino en Airbyte

Para el origen, deberemos hacer click en "Source" en el menú izquierdo de la interfaz, y allí seleccionaremos como "Source type" nuestra base de datos Postgres. Rellenaremos los campos solicitados en función de los valores que hayamos elegido previamente.

También deberemos configurar el método de replicación y seleccionar "Logical Replication (CDC)", indicando la PUBLICATION creada anteriormente al configurar CDC.

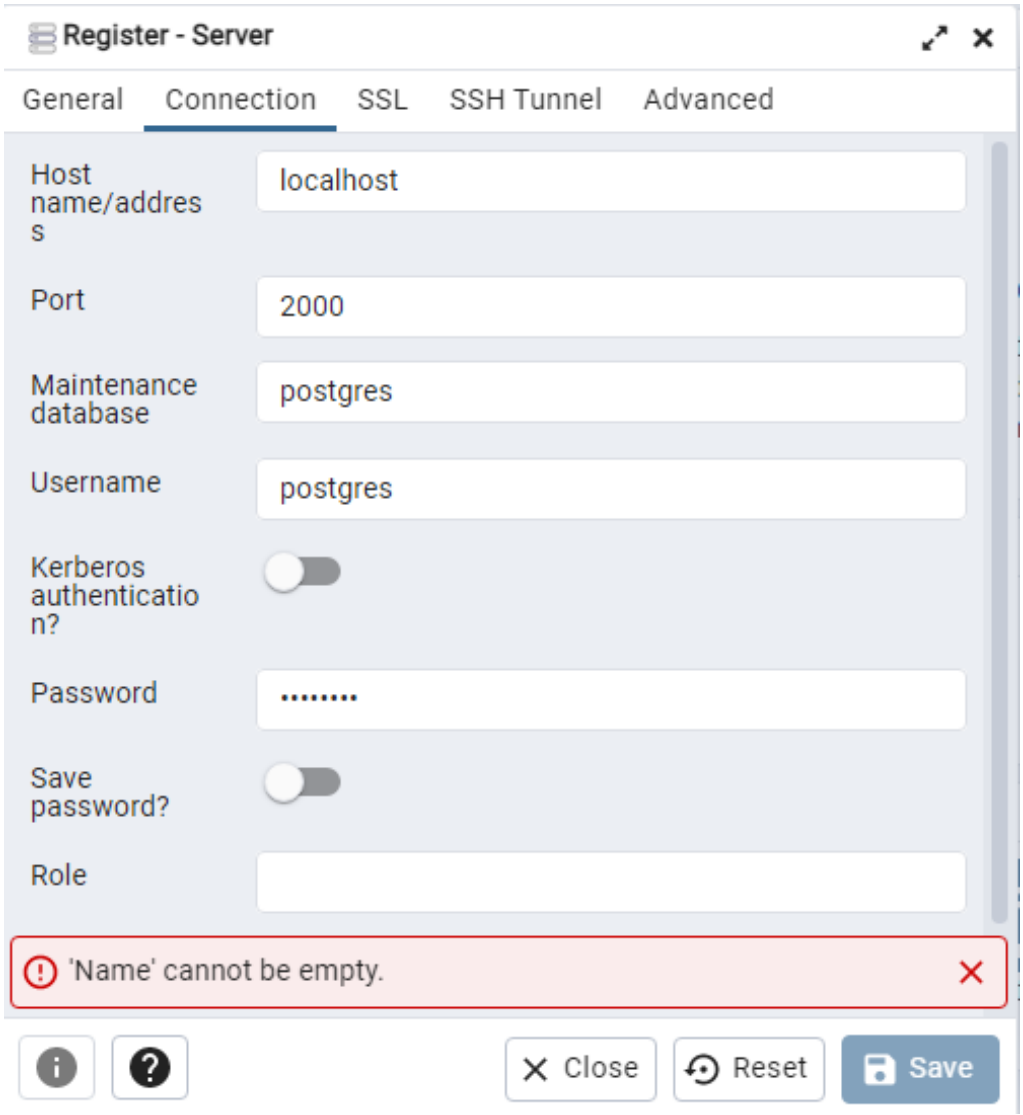
Haremos click en Set Up Source, y comprobaremos que todo transcurre sin problemas, en caso de que nos diese algún fallo deberemos revisar la configuración y volver a lanzarlo.

Para el destino, deberemos hacer prácticamente lo mismo, con la diferencia de modificar el puerto al 3000 (en caso de estar siguiendo el mismo esquema que el manual).

Al lanzar el destino, nos aparecerán diferentes opciones de configuración, como la tasa de refresco y algún valor más que podremos probar en función de nuestras necesidades.

- Seguimiento de BBDD

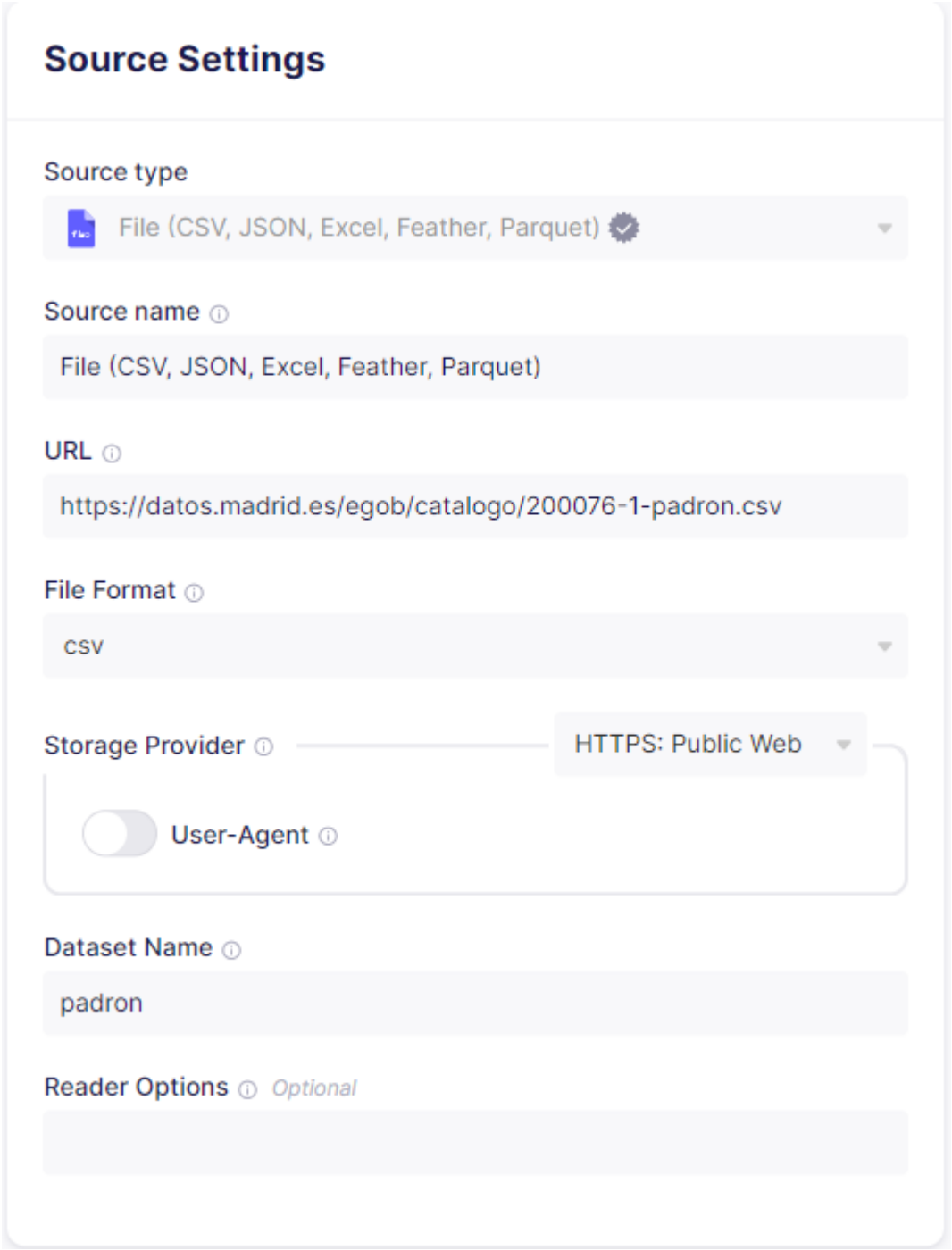
Para poder seguir todos estos cambios desde una interfaz instalada previamente como es el caso de pgAdmin 4, vamos a crear dentro de esta inferfaz un nuevo servidor a traves de la opción "Create New Server", al cuál daremos el nombre que consideremos, y en las opciones de conexión indicaremos el host, puerto, nombre, contraseña y esquema que hemos utilizado anteriormente. De esta forma podremos ver las tablas que hemos ido creando dentro del esquema que hemos asignado.



CARGAR CSV EN POSTGRES

Para cargar un CSV, el cual he sacado de internet a partir de uno usado durante el proceso de formación como es el del padrón de Madrid (https://datos.madrid.es/egob/catalogo/200076-1-padron.csv), hemos seguido una serie de pasos.

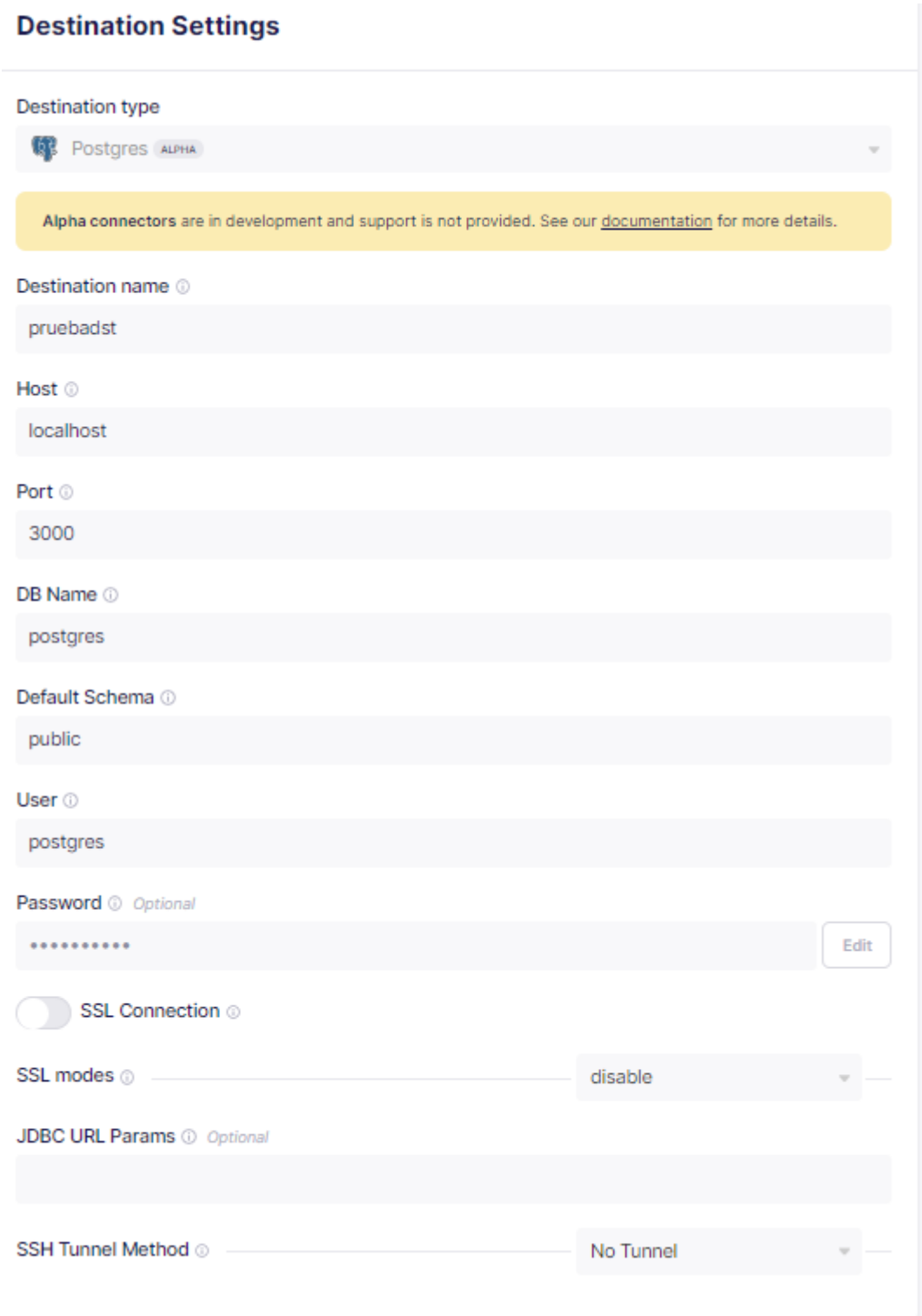
Lo primero será configurar la fuente o Source, seleccionando CSV File. Alli estableceremos la siguiente configuración:



El Dataset Name será el nombre del dataset donde se van a almacenar los datos contenidos en el CSV.

A continuación vamos a preparar el destino, para ello lanzamos el siguiente comando:
"docker run --rm --name airbyte-destination -e POSTGRES_PASSWORD=password -p 3000:5432 -d debezium/postgres:14"
Y dentro de pgAdmin crearemos un servidor con el puerdo correspondiente, nombre, etc

Una vez tengamos el Server creado, estableceremos el destino dentro de Airbyte con la siguiente configuración:



Una vez tengamos esto ya podremos ejecutar querys desde pgAdmin.