

UNIVERSIDAD NACIONAL DE ASUNCIÓN
Facultad Politécnica
Proyecto Centro de Innovación TIC – Paraguay-KOICA

Curso Básico de Introducción a Big Data

Prof. Ing. Richard D. Jiménez-R., <rjimenez@pol.una.py>

Despliegue Apache Airflow en Windows con WSL

“Manual de Instalación y Configuración en una PC local”

Apache Airflow®

Apache Airflow es una plataforma creada por la comunidad para *crear, programar y monitorear flujos de trabajo (workflows)* de manera programada (*schedule*).

Principios:

- **Escalable:** tiene una arquitectura modular y utiliza una cola de mensajes para organizar una cantidad arbitraria de trabajadores. Airflow está listo para escalar sin límites.
- **Dinámico:** los *pipelines* de Apache Airflow se definen en Python, lo que permite la generación dinámica de pipeline. Esto permite escribir código que crea instancias de pipelines de forma dinámica.
- **Extensible:** defina fácilmente sus propios operadores y amplíe las bibliotecas para que se ajusten al nivel de abstracción adecuado para su entorno.
- **Elegante:** los pipelines de Apache Airflow son simples y explícitos. La parametrización está integrada en su núcleo mediante el potente motor de plantillas Jinja.

Características principales:

- **Python puro:** se utilizan las funciones estándar de Python para crear los flujos de trabajo, incluidos los formatos de fecha y hora para la programación y los bucles para generar tareas de forma dinámica. Esto permite mantener una flexibilidad total al crear los flujos de trabajo.
- **Interfaz de usuario útil:** para el monitoreo, programación y administración de los flujos de trabajo a través de una aplicación web robusta y moderna. No se necesita aprender a usar las interfaces antiguas, como las de cron. Siempre se tendrá una visión completa del estado y los registros de las tareas completadas y en curso.
- **Integraciones sólidas:** Apache Airflow ofrece numerosos operadores listos para ejecutar las tareas en Google Cloud Platform, Amazon Web Services, Microsoft Azure y muchos otros servicios de terceros. Esto hace que Airflow sea fácil de aplicar a la infraestructura actual y de extender a tecnologías de última generación.
- **Fácil de usar:** Cualquier persona con conocimientos de Python puede implementar un flujo de trabajo. Apache Airflow no limita el alcance de sus flujos de trabajo; puede usarlo para crear modelos de aprendizaje automático, transferir datos, administrar su infraestructura y más.
- **Código abierto:** Dondequiera que quieras compartir tu mejora, puedes hacerlo abriendo una PR. Es así de simple, sin barreras ni procedimientos prolongados. Airflow tiene muchos usuarios activos que comparten voluntariamente sus experiencias. Está disponible Apache Airflow® community.

Casos de uso

Apache Airflow permite definir prácticamente cualquier flujo de trabajo en código Python, sin importar lo complejo que sea. Debido a su versatilidad, Airflow es utilizado por empresas de todo el mundo para una variedad de casos de uso. Visita su sitio oficial para consultar los cuatro casos de uso más comunes para Airflow, pero las posibilidades son muchas.

Docker Image for Apache Airflow

Airflow tiene un Dockerfile oficial y una imagen de Docker publicada en *DockerHub* como un paquete de conveniencia para la instalación. Puede ampliar y personalizar la imagen según sus requisitos y usarla en sus propias implementaciones. Lea la documentación.

Arquitectura básica de Apache Airflow

Airflow es una plataforma que permite crear y ejecutar flujos de trabajo. Un flujo de trabajo se representa como un **DAG (un gráfico acíclico dirigido)** y contiene piezas de trabajo individuales denominadas tareas, organizadas teniendo en cuenta las dependencias y los flujos de datos.

Un DAG especifica las dependencias entre tareas y el orden en que se deben ejecutar y ejecutar reintentos; las tareas en sí mismas describen qué hacer, ya sea obtener datos, ejecutar análisis, activar otros sistemas o más.

Componentes principales:

Una instalación de Airflow generalmente consta de los siguientes componentes:

1. **DagBag:** Módulo encargado de leer todos los DAGs definidos en los archivos Python y los carga en la base de datos de Airflow.
2. **Scheduler:** Planifica y lanza las tareas que deben ejecutarse según la programación definida en los DAGs.
3. **Executor:** Ejecuta las tareas planificadas por el *Scheduler*. Airflow admite varios tipos de ejecutores (por ejemplo, LocalExecutor, CeleryExecutor, KubernetesExecutor).
4. **Workers:** Máquinas o procesos que realizan las tareas asignadas por el *Executor*.
5. **Metadata Database:** Base de datos que almacena el estado y la información de los DAGs y las tareas. Comúnmente es una base de datos SQL.
6. **Web Server:** Interfaz de usuario que permite a los usuarios interactuar con Airflow, visualizar el estado de los DAGs y tareas, y administrar el sistema.
7. **Logs:** Sistema de almacenamiento de logs para registrar la ejecución de las tareas.

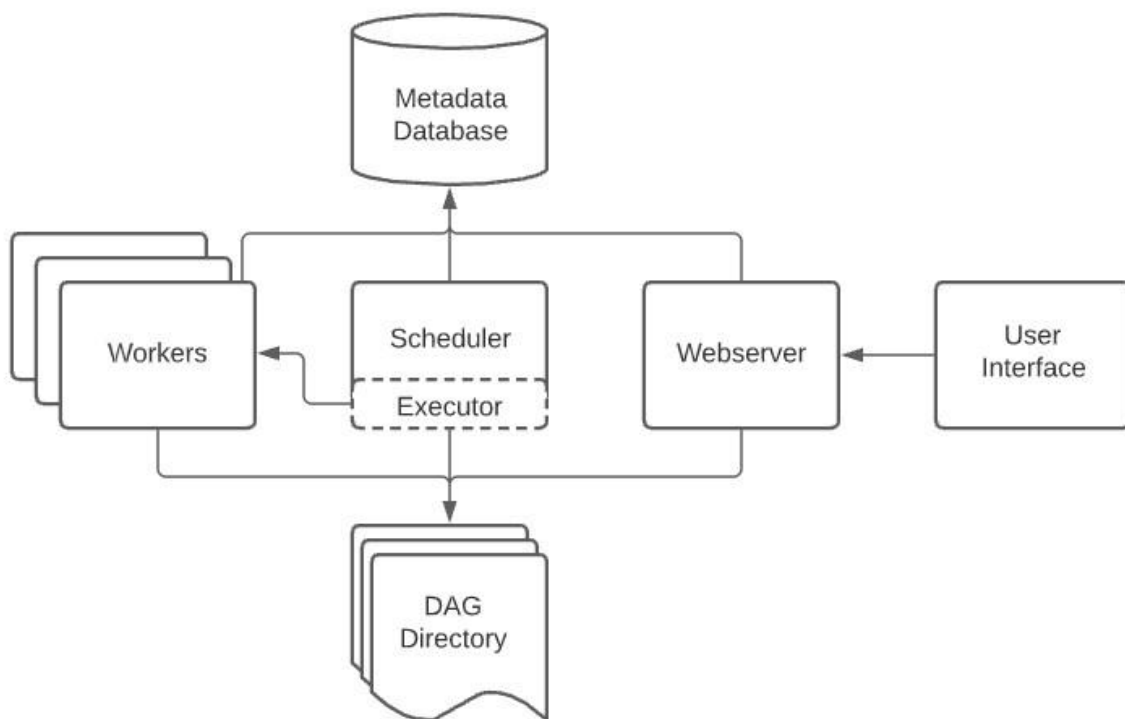


Figura 1. Arquitectura básica de apache airflow.

Descripción de la Interacción:

1. **Usuario:** Interactúa con el sistema a través del *Web Server* para crear, visualizar y gestionar DAGs y tareas.
2. **Web Server:** Proporciona una interfaz gráfica para la administración de Airflow. Se comunica con la *Metadata Database* para mostrar el estado y la información de los DAGs y tareas.
3. **Scheduler:** Lee los DAGs desde el *DagBag* y los programa según la definición. Guarda el estado de las tareas en la *Metadata Database*.
4. **Executor:** Recibe las tareas programadas por el *Scheduler* y las asigna a los *Workers*.
5. **Workers:** Ejecutan las tareas asignadas y almacenan los logs de ejecución en el sistema de Logs.
6. **Metadata Database:** Guarda toda la información del estado de los DAGs, tareas, y logs de ejecución.

- 7. Logs:** Almacena los logs generados por los *Workers* durante la ejecución de las tareas.

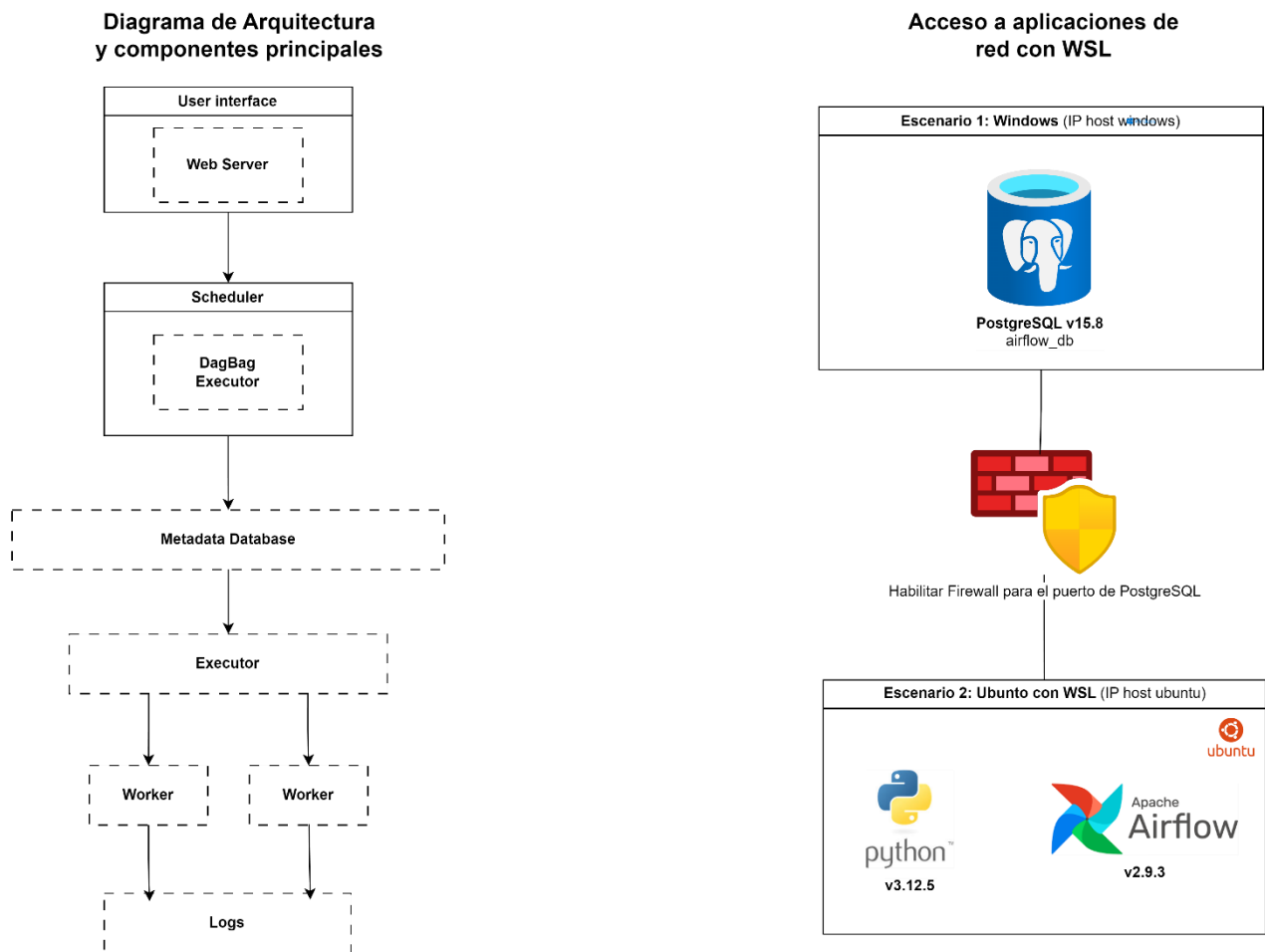


Figura 2. Diagrama de la infraestructura de instalación.

Identificar la dirección IP:

Hay dos escenarios que se deben tener en cuenta al identificar la dirección IP que se usa para una distribución de Linux que se ejecuta a través de WSL:

- 1. Escenario uno:** desde el host de Windows, para consultar la dirección IP de una distribución de Linux (Ubuntu) que se ejecuta a través de WSL, de modo que un programa en el host de Windows pueda conectarse a un programa de servidor que se ejecuta dentro de la distribución (instancia).

El host de Windows puede usar el comando:

```
wsl hostname -I
```

- 2. Escenario dos:** un programa que se ejecuta dentro de una distribución de Linux (Ubuntu) a través de WSL (instancia) quiere conocer la dirección IP del host de Windows, para que un programa de Linux (Ubuntu) pueda conectarse a un programa de servidor host de Windows.

El usuario de WSL Linux puede usar el comando:

```
ip route show | grep -i default | awk '{print $3}'
```

Prerrequisitos

Antes de comenzar, asegúrate de tener lo siguiente instalado en tu PC:

- 1. PostgreSQL v15:** Para la base de datos de la *metadata* de Airflow.
 - www.postgresql.org/download/
- 2. Windows Subsystem for Linux (WSL):** Habilite WSL en su sistema Windows siguiendo la documentación oficial de Microsoft.
 - <https://learn.microsoft.com/es-es/windows/wsl/install>

3. **Python 3.12.5:** Una vez habilitado WSL, puedes verificar la versión de Python instalada ejecutando `python --version` o `python3 --version` en su sistema Linux.
4. **pip:** El gestor de paquetes de Python. Generalmente, viene preinstalado con Python, pero puedes actualizarlo ejecutando `python -m pip install --upgrade pip`
5. **virtualenv:** Herramienta para crear entornos virtuales en Python. Puedes instalar ejecutando `pip install virtualenv`

Guía de Instalación paso a paso

Aquí, se detalla el manual paso a paso para la instalación y configuración Airflow en una PC local.

Paso 1: Instalación de WSL en Windows.

Primeramente, asegúrese de haber habilitado el Subsistema de Windows para Linux (WSL) e instalado la distribución de Linux como Ubuntu.

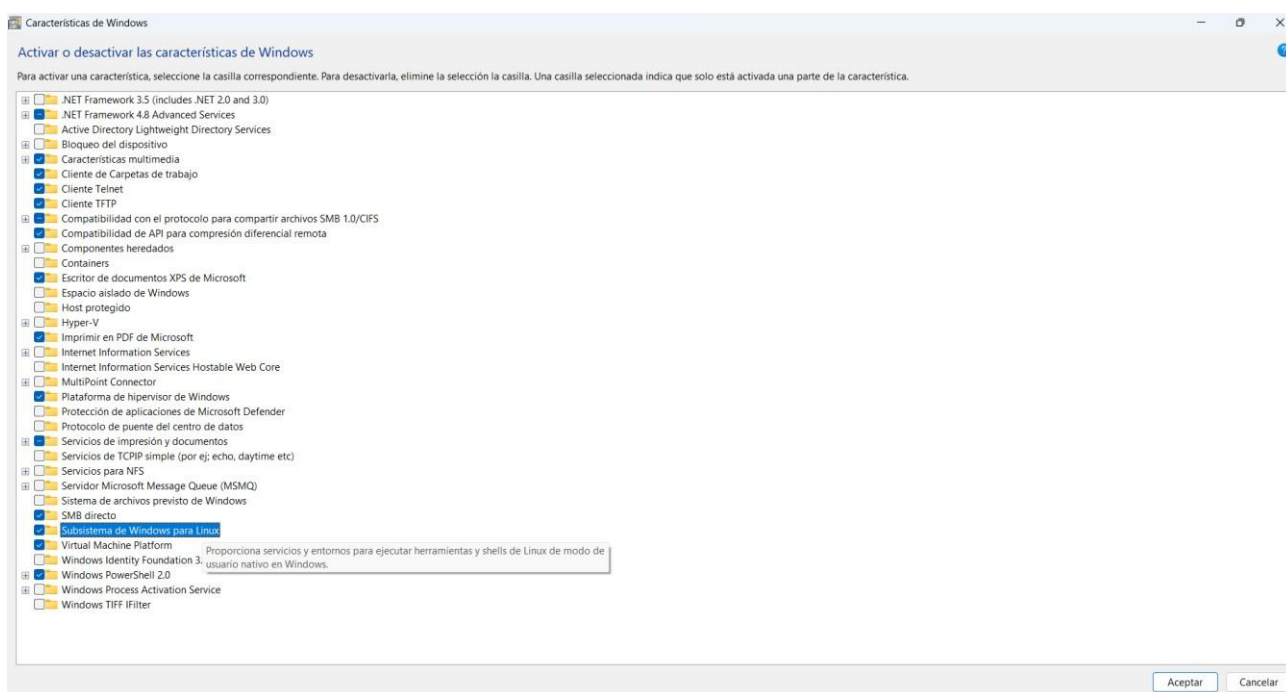


Figura 3. Panel de características de windows.

1. Instalar WSL

Abrir una Terminal de Windows como administrador y ejecuta:

```
wsl --install
```

Luego, reinicie la máquina. Este comando habilitará las características necesarias para ejecutar WSL e instalará la distribución Ubuntu de Linux.

Notas:

- La primera vez que inicie una distribución de Linux recién instalada, se abrirá una ventana de la consola y se le pedirá que espere a que los archivos se descompriman y se almacenen en el equipo.
- Actualiza el paquete de Subsistema de Windows para Linux: `wsl --update`
- Para ver la información de uso: `wsl -help`
- Muestra información de la versión: `wsl --version`
- Para verificar el estado del Subsistema de Windows para Linux:

```
wsl --status
```

- Si ya tienes WSL instalado, puedes asegurarte de que está actualizado y que estás usando WSL 2:

```
wsl --set-default-version 2
```

- Para verificar las distribuciones de subsistema de Windows para Linux disponible:

```
wsl --list
```

- Para enumerar las distribuciones de Linux instaladas y comprobar en qué versión de WSL está establecida cada una, puede escribir el comando:

```
wsl -l -v
```

- Para listar las distribuciones válidas que se pueden instalar:

```
wsl --list --online
```

Paso 2: Instalar Python.

Aunque Ubuntu viene con una versión de Python preinstalada, instalar desde fuente te permite tener mayor control sobre la versión y la ubicación de la instalación. Esto es especialmente útil si necesitas una versión específica o si quieres gestionar múltiples versiones de Python en tu sistema.

1. Actualizar los paquetes instalados del sistema.

Si deseas actualizar los paquetes instalados en tu sistema, puedes ejecutar:

```
sudo apt update && sudo apt upgrade -y
```

2. Instalar las dependencias necesarias.

Para compilar e instalar Python desde la fuente, necesitas algunas herramientas y bibliotecas. A continuación, puedes ejecutar:

```
sudo apt install -y build-essential libssl-dev zlib1g-dev  
libncurses5-dev libncursesw5-dev libreadline-dev  
libsqlite3-dev libgdbm-dev libdb5.3-dev libbz2-dev  
libexpat1-dev liblzmadev tk-dev libffi-dev uuid-dev
```

3. Descargar el código fuente de Python 3.12.5

Descarga el código fuente de Python 3.12.5 desde el sitio oficial:

```
cd /usr/src  
sudo wget https://www.python.org/ftp/python/3.12.5/Python-3.12.5.tgz
```

4. Descomprimir el archivo Python-3.12.5.tgz

Extrae el contenido del archivo descargado:

```
sudo tar xzf Python-3.12.5.tgz
```

5. Configurar la instalación.

Ingresa al directorio recién creado:

```
cd Python-3.12.5
```

Configura la instalación. Aquí puedes personalizar la ubicación de instalación, si deseas. Para una instalación estándar, ejecuta:

```
sudo ./configure --enable-optimizations
```

La opción `--enable-optimizations` activa ciertas optimizaciones del compilador para mejorar el rendimiento de Python.

6. Compilar e instalar Python 3.12.5

Compila y luego instala Python:

```
sudo make -j $(nproc)
sudo make install
```

La opción `-j $(nproc)` utiliza todos los núcleos de tu CPU para acelerar la compilación. Usamos `install` para evitar sobrescribir la versión predeterminada de Python del sistema. Sin embargo, El comando `sudo make altinstall` instala Python en un directorio diferente al de la versión de sistema, evitando conflictos.

7. Verificar la instalación de Python 3.12.5

Para verificar si la instalación fue exitosa, ejecuta:

```
python3 --version
```

8. Instalar pip

`pip` debería estar instalado automáticamente con Python 3.12.5. Se puede verificar ejecutando:

```
python3 -m pip --version
```

Si `pip` no está instalado, puedes instalarlo manualmente descargando `get-pip.py` e instalándolo:

```
sudo apt install python3-pip
```

Para verificar la versión de `pip` dentro del entorno virtual:

```
pip --version
```

Ejecuta el siguiente comando para actualizar `pip` a la última versión globalmente (fuera del entorno virtual):

```
python3 -m pip install --upgrade pip
```

9. Instalar virtualenv

Si `virtualenv` no está instalado, puedes instalarlo usando `pip`. Abre la línea de comandos (CMD) y ejecuta:

```
pip install virtualenv
```

Has instalado con éxito Python 3.12.5 en tu sistema Ubuntu. Ahora puedes utilizar esta versión de Python para desarrollar y ejecutar tus aplicaciones.

Paso 3: Configuración de PostgreSQL.

Para permitir conexiones desde cualquier máquina remota. Además, te mostraremos cómo configurar el firewall para permitir el tráfico entrante al puerto de PostgreSQL (por defecto, el 5432).

1. Identificar la ubicación del archivo `pg_hba.conf`

El archivo `pg_hba.conf` controla qué usuarios pueden conectarse a una base de datos PostgreSQL. Su ubicación puede variar ligeramente según instalación, pero generalmente se encuentra en el directorio de datos de PostgreSQL:

```
C:\Program Files\PostgreSQL\15\data\
```

2. Editar el archivo `pg_hba.conf`

Una regla que permite conexiones desde cualquier dirección:


```
# IPv4 local connections:
host all all 0.0.0.0/0 md5
```

Esta línea permite a cualquier usuario conectarse a cualquier base de datos desde cualquier dirección IP utilizando el método de autenticación MD5.

3. Editar el archivo postgresql.conf

Configuración de conexiones:

```
listen_addresses = '*'
```

4. Reiniciar el servicio de PostgreSQL.

Para que los cambios surtan efecto, reinicia el servicio de PostgreSQL.

5. Configurar el Firewall.

Crear una nueva regla para permitir conexiones entrantes al puerto 5432:

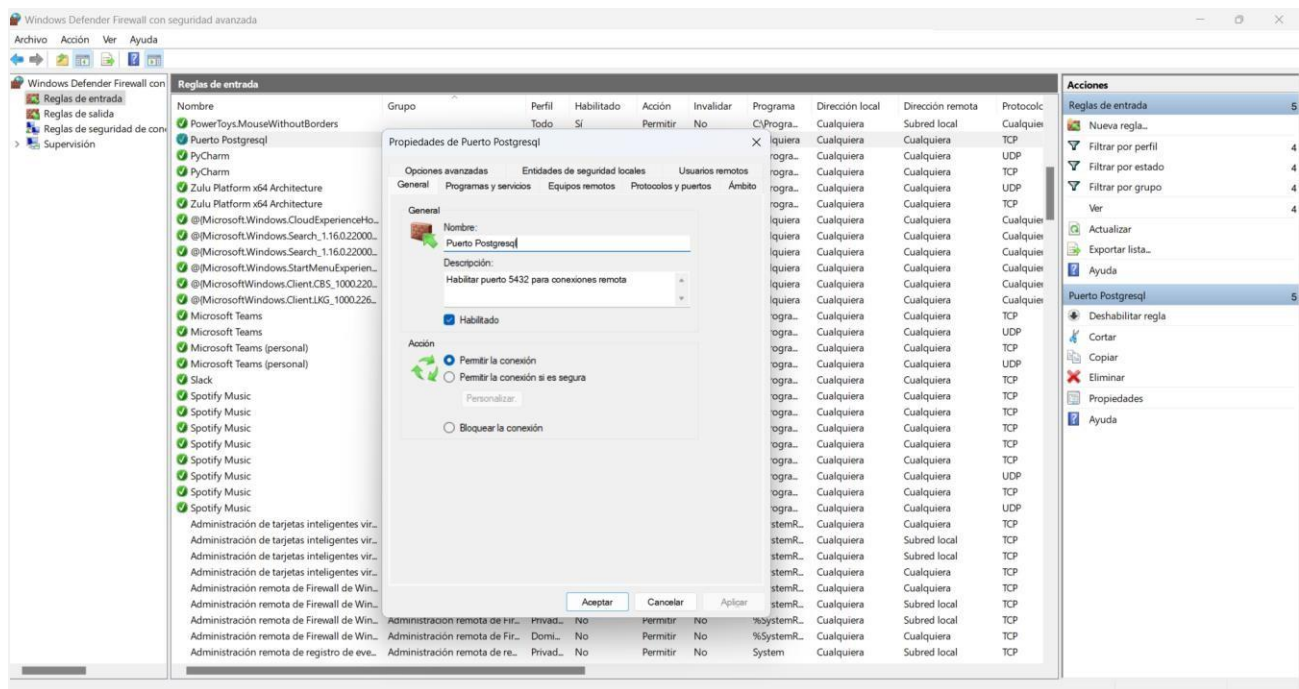


Figura 4. Panel de configuración de firewall de windows.

6. Verificar la conexión a la base de datos.

Utilizar telnet para intentar conectarte desde Ubuntu WSL al host de Windows, por ejemplo:

```
ip route show | grep -i default | awk '{print $3}'
```

Si la conexión se establece exitosamente, se habrá configurado correctamente PostgreSQL para el acceso remoto.

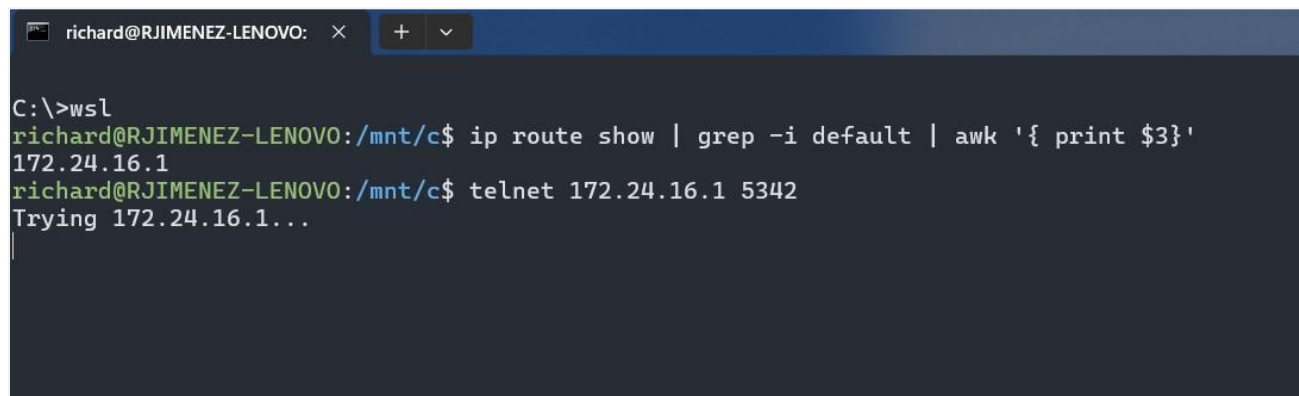


Figura 5. Verificar conexión con el servicio de postgresql.

Paso 4: Instalación y Configuración de Apache Airflow 2.9.3

1. Crear un entorno virtual.

1. Abre un terminal e iniciar: `wsl`
2. Crea un directorio para tu proyecto Airflow:

```
cd /home/<nombre_usuario>
mkdir analytics
cd analytics
mkdir airflow_project
cd airflow_project
```

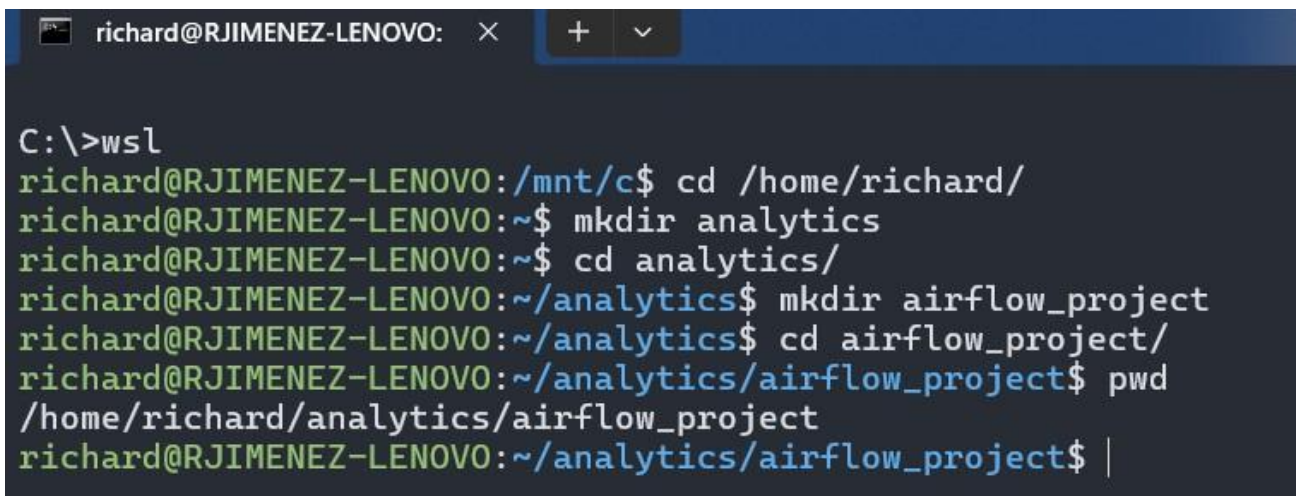


Figura 6. Directorio para la instalación.

3. Crea un entorno virtual:

```
python3 -m venv venv
```

También, se puede usar el comando: `virtualenv venv`

4. Activa el entorno virtual:

```
source venv/bin/activate
```

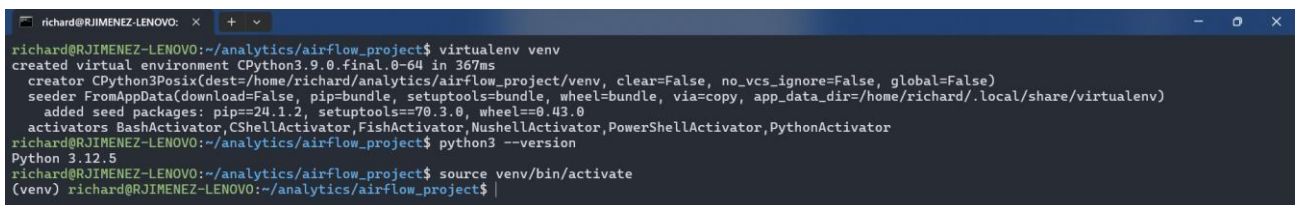


Figura 7. Crear entorno virtual para la instalación aislada.

Para salir del entorno virtual utilizar el comando: `source venv/bin/deactivate`

2. Instalar Apache Airflow.

Ahora que tienes el entorno virtual activado, puedes instalar Apache Airflow. Asegúrate de especificar la versión de Airflow y las restricciones de versiones de dependencias correspondientes.

1. Establece las variables de entorno necesarias para la instalación:

```
sudo wget
https://raw.githubusercontent.com/apache/airflow/constrain
ts2.9.3/constraints-3.12.txt
```

2. Instala Apache Airflow utilizando `pip`:


```
pip install "apache-airflow==2.9.3" -constraint
"constraints3.12.txt"
```

3. Configuración de una base de datos PostgreSQL.

1. Crear una base de datos que Airflow utilizará para acceder a esta base de datos. Para crear una base de datos `airflow_db` y para el usuario `postgres` y contraseña `postgres`:

```
CREATE DATABASE airflow_db;

GRANT ALL PRIVILEGES ON DATABASE airflow_db TO postgres;

ALTER USER postgres SET search_path = public;
ALTER DATABASE airflow_db OWNER TO postgres;

-- PostgreSQL 15 requiere de privilegios adicionales:
USE airflow_db;
GRANT ALL ON SCHEMA public TO postgres;
```

2. Instalar el controlador `psycopg2`:

```
pip install psycopg2-binary
```

3. Editar el archivo `/home/<nombre_usuario>/airflow/airflow.cfg`

```
[core]
sql_alchemy_conn=postgresql+psycopg2://postgres:postgres@<ip_wi
ndows>:5432/airflow_db

dags_folder = /home/<usuario>/analytics/airflow_project/dags

default_timezone = "America/Asuncion"

executor = LocalExecutor

load_examples = False

[webserver]
default_ui_timezone = "America/Asuncion"

expose_config = True

test_connection = Enabled
```

Opcional:

Para ocultar los ejemplos de DAG que se envían con Airflow:

```
load_examples = Falase
```

Formato de la URI para la cadena de conexión SQLAlchemy:

```
postgresql+psycopg2://<user>:<password>@<host>/<db>
```

4. Inicializar la base de datos de Airflow.

Airflow utiliza una base de datos para almacenar información sobre flujos de trabajo, tareas, y usuarios.

1. Inicializa la base de datos de Airflow:

```
airflow db migrate
```

5. Crear un usuario administrador.

1. Crear usuario administrador:

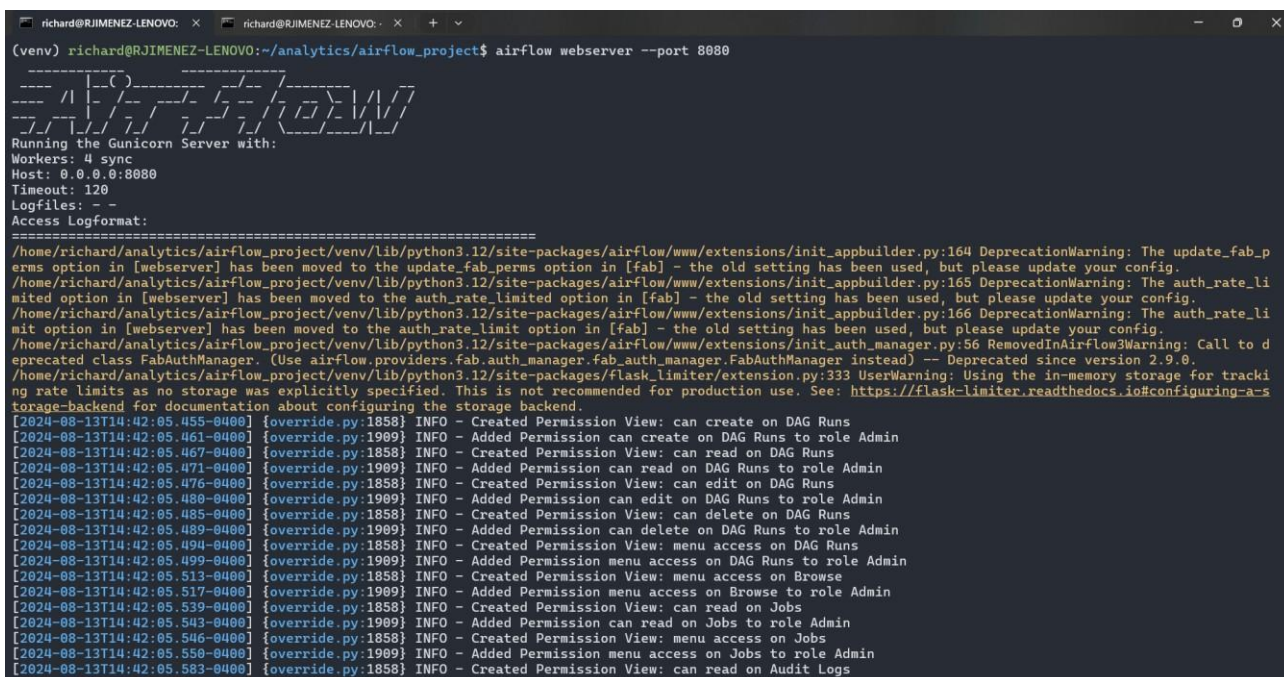
```
airflow users create
--username admin --password admin
--firstname Richard --lastname Jiménez
--role Admin --email rjimenez@pol.una.py
```

6. Iniciar el servidor web de Airflow.

La **consola web** de Apache Airflow es una interfaz gráfica que permite a los usuarios administrar, monitorear y ejecutar flujos de trabajo (DAGs) de manera eficiente. Desde esta interfaz, puedes visualizar el estado de las tareas, acceder a registros de ejecución, configurar parámetros y realizar acciones administrativas sobre los DAGs.

1. Iniciar el servidor web de Airflow:

```
airflow webserver --port 8080
```



```
(venv) richard@RJIMENEZ-LENOVO:~/analytics/airflow_project$ airflow webserver --port 8080
Running the Gunicorn Server with:
Workers: 4 sync
Host: 0.0.0.0:8080
Timeout: 120
Logfiles: -
Access Logformat:
/home/richard/analytics/airflow_project/venv/lib/python3.12/site-packages/airflow/www/extensions/init_appbuilder.py:164 DeprecationWarning: The update_fab_perms option in [webserver] has been moved to the update_fab_perms option in [fab] - the old setting has been used, but please update your config.
/home/richard/analytics/airflow_project/venv/lib/python3.12/site-packages/airflow/www/extensions/init_appbuilder.py:165 DeprecationWarning: The auth_rate_limit option in [webserver] has been moved to the auth_rate_limit option in [fab] - the old setting has been used, but please update your config.
/home/richard/analytics/airflow_project/venv/lib/python3.12/site-packages/airflow/www/extensions/init_appbuilder.py:166 DeprecationWarning: The auth_rate_limit option in [webserver] has been moved to the auth_rate_limit option in [fab] - the old setting has been used, but please update your config.
/home/richard/analytics/airflow_project/venv/lib/python3.12/site-packages/airflow/www/extensions/init_auth_manager.py:56 RemovedInAirflow3Warning: Call to deprecated class FabAuthManager. (Use airflow.providers.fab.auth_manager.fab_auth_manager.FabAuthManager instead) -- Deprecated since version 2.9.0.
/home/richard/analytics/airflow_project/venv/lib/python3.12/site-packages/flask_limiter/extension.py:333 UserWarning: Using the in-memory storage for tracking rate limits as no storage was explicitly specified. This is not recommended for production use. See: https://flask-limiter.readthedocs.io#configuring-a-storage-backend-for-documentation about configuring the storage backend.
[2024-08-13T14:42:05.455-0400] [override.py:1858] INFO - Created Permission View: can create on DAG Runs
[2024-08-13T14:42:05.461-0400] [override.py:1989] INFO - Added Permission can create on DAG Runs to role Admin
[2024-08-13T14:42:05.467-0400] [override.py:1858] INFO - Created Permission View: can read on DAG Runs
[2024-08-13T14:42:05.471-0400] [override.py:1989] INFO - Added Permission can read on DAG Runs to role Admin
[2024-08-13T14:42:05.476-0400] [override.py:1858] INFO - Created Permission View: can edit on DAG Runs
[2024-08-13T14:42:05.480-0400] [override.py:1989] INFO - Added Permission can edit on DAG Runs to role Admin
[2024-08-13T14:42:05.485-0400] [override.py:1858] INFO - Created Permission View: can delete on DAG Runs
[2024-08-13T14:42:05.489-0400] [override.py:1989] INFO - Added Permission can delete on DAG Runs to role Admin
[2024-08-13T14:42:05.494-0400] [override.py:1858] INFO - Created Permission View: menu access on DAG Runs
[2024-08-13T14:42:05.499-0400] [override.py:1989] INFO - Added Permission menu access on DAG Runs to role Admin
[2024-08-13T14:42:05.513-0400] [override.py:1858] INFO - Created Permission View: menu access on Browse
[2024-08-13T14:42:05.517-0400] [override.py:1989] INFO - Added Permission menu access on Browse to role Admin
[2024-08-13T14:42:05.539-0400] [override.py:1858] INFO - Created Permission View: can read on Jobs
[2024-08-13T14:42:05.543-0400] [override.py:1989] INFO - Added Permission can read on Jobs to role Admin
[2024-08-13T14:42:05.546-0400] [override.py:1858] INFO - Created Permission View: menu access on Jobs
[2024-08-13T14:42:05.550-0400] [override.py:1989] INFO - Added Permission menu access on Jobs to role Admin
[2024-08-13T14:42:05.583-0400] [override.py:1858] INFO - Created Permission View: can read on Audit Logs
```

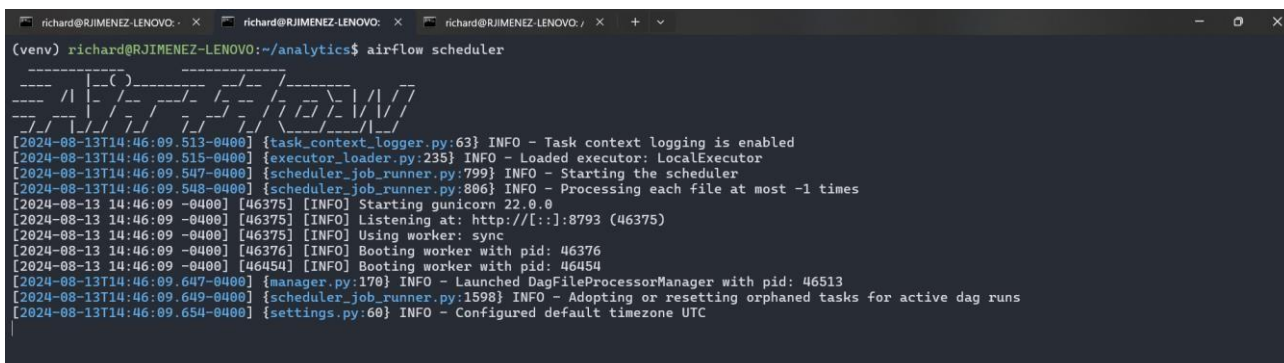
Figura 8. Iniciar el servicio webserver.

7. Iniciar el planificador de Airflow.

El **planificador** de Airflow se encarga de programar las tareas y ejecutarlas en el orden correcto.

1. En una nueva terminal (con el entorno virtual activado), inicia el planificador:

```
airflow scheduler
```



```
(venv) richard@RJIMENEZ-LENOVO:~/analytics$ airflow scheduler
[2024-08-13T14:46:09.513-0400] [task_context_logger.py:63] INFO - Task context logging is enabled
[2024-08-13T14:46:09.515-0400] [executor_loader.py:235] INFO - Loaded executor: LocalExecutor
[2024-08-13T14:46:09.547-0400] [scheduler_job_runner.py:799] INFO - Starting the scheduler
[2024-08-13T14:46:09.548-0400] [scheduler_job_runner.py:806] INFO - Processing each file at most -1 times
[2024-08-13 14:46:09 -0400] [46375] [INFO] Starting gunicorn 22.0.0
[2024-08-13 14:46:09 -0400] [46375] [INFO] Listening at: http://[::]:8793 (46375)
[2024-08-13 14:46:09 -0400] [46375] [INFO] Using worker: sync
[2024-08-13 14:46:09 -0400] [46376] [INFO] Booting worker with pid: 46376
[2024-08-13 14:46:09 -0400] [46454] [INFO] Booting worker with pid: 46454
[2024-08-13T14:46:09.647-0400] [manager.py:170] INFO - Launched DagFileProcessorManager with pid: 46513
[2024-08-13T14:46:09.649-0400] [scheduler_job_runner.py:1598] INFO - Adopting or resetting orphaned tasks for active dag runs
[2024-08-13T14:46:09.654-0400] [settings.py:60] INFO - Configured default timezone UTC
```

Figura 9. Iniciar el servicio scheduler.

8. Verificar la configuración.

1. Abre tu navegador web y navega a `http://localhost:8080`. Inicia sesión con las credenciales del usuario administrador creado en el paso anterior.

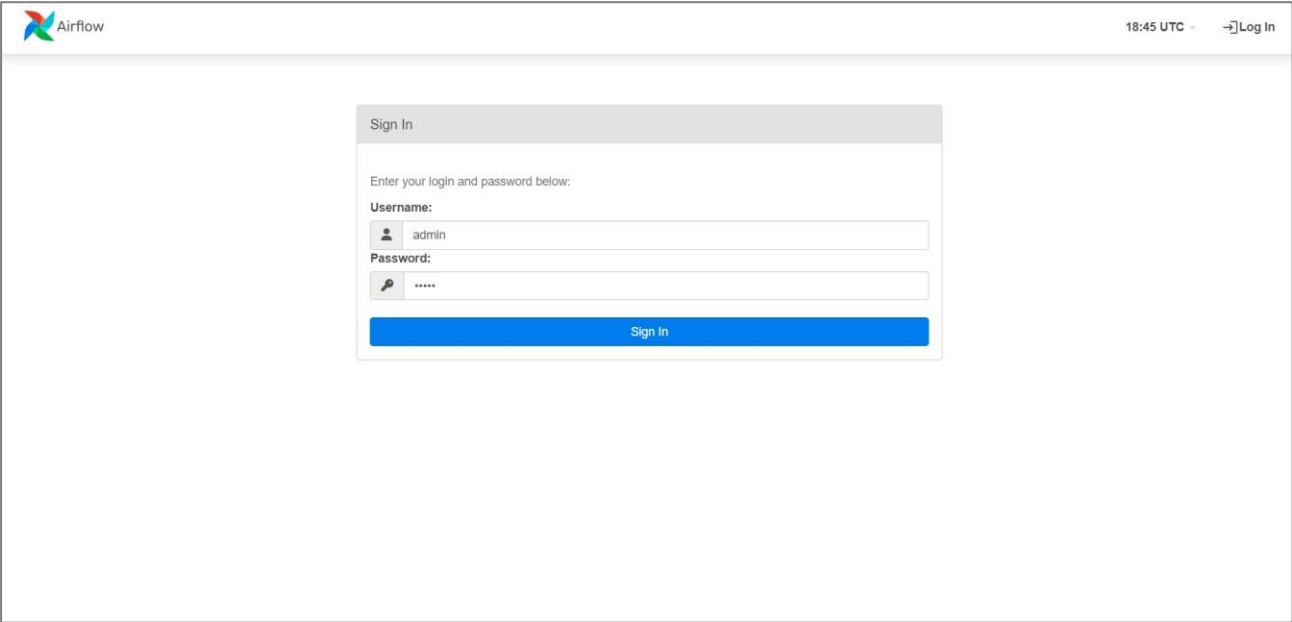


Figura 10. Panel web para iniciar sesión con las credenciales de usuario administrador.

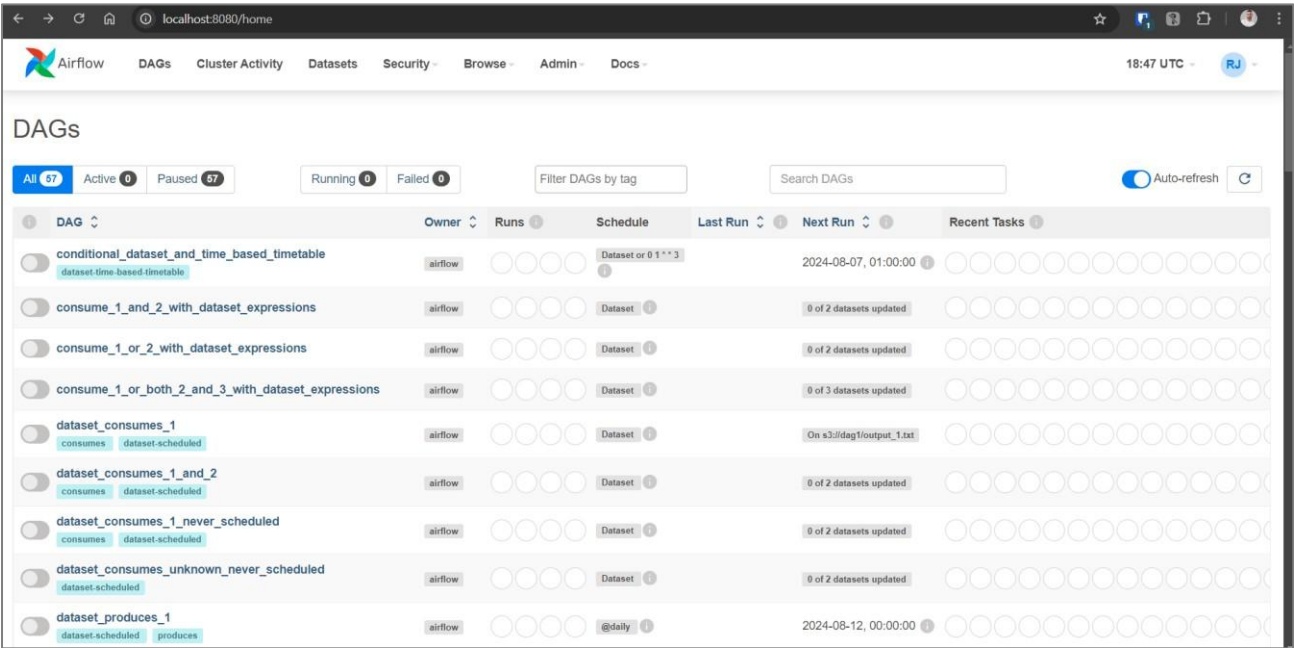


Figura 11. Panel de administración de los DAGs.

Paso 5: Crear y configurar una conexión PostgreSQL.

1. Instalar proveedor de conexión PostgreSQL para Airflow:

```
pip install apache-airflow-providers-postgres==6.0.0
```

2. Luego, reiniciar los servicios para que tome los nuevos valores.

```
# Detener el servicio:
pkill -f "airflow-webserver"

# Activar el servicio (Opción 1):
airflow webserver --port 8080

# Iniciar el WebServer en segundo plano (Opción 2):
airflow webserver -D
```

Has instalado y configurado correctamente Apache Airflow en tu PC local. Ahora puedes comenzar a crear y ejecutar flujos de trabajo más complejos según tus necesidades. Recuerda siempre trabajar dentro del entorno virtual activado para garantizar que las dependencias de tu proyecto estén aisladas y correctamente gestionadas.

Package Name	Version	Description
apache-airflow-providers-common-io	1.3.2	Common IO Provider
apache-airflow-providers-common-sql	1.14.2	Common SQL Provider
apache-airflow-providers-fab	1.2.2	Flask App Builder
apache-airflow-providers-ftp	3.10.0	File Transfer Protocol (FTP)
apache-airflow-providers-http	4.12.0	Hypertext Transfer Protocol (HTTP)
apache-airflow-providers-imap	3.6.1	Internet Message Access Protocol (IMAP)
apache-airflow-providers-postgres	5.11.3	PostgreSQL
apache-airflow-providers-smtp	1.7.1	Simple Mail Transfer Protocol (SMTP)
apache-airflow-providers-sqlite	3.8.1	SQLite

Version: v2.9.3
Git Version: .release:81845de9d95a733b4eb7826aaabe23ba9813eba3

Figura 12. Panel de controladores instalados.

Asegúrate de que tu entorno tenga la versión adecuada de Apache Airflow y las dependencias necesarias. Para verificar la instalación, puedes ejecutar:

```
pip show apache-airflow-providers-postgres
```

Recursos

A continuación, se deja el acceso a los siguientes enlaces para más información:

- **Sitio Oficial Apache Airflow:**

<https://airflow.apache.org/>

- **Use Cases:**

<https://airflow.apache.org/use-cases/>

- **Apache Airflow community:**

<https://airflow.apache.org/community/>

- **Documentación Técnica – Airflow versión 2.9.3:**

<https://airflow.apache.org/docs/apache-airflow/stable/index.html>

- **Install Airflow 2.9.3:**

<https://airflow.apache.org/docs/apache-airflow/2.9.3/start.html>

- **Descripción general de la arquitectura:**

<https://airflow.apache.org/docs/apache-airflow/2.9.3/core-concepts/overview.html>

- **Configuración de una base de datos PostgreSQL:**

<https://airflow.apache.org/docs/apache-airflow/2.7.3/howto/set-up-database.html#setting-up-a-postgresql-database>

- **Introducción a las bases de datos en el Subsistema Windows para Linux:**

<https://learn.microsoft.com/es-es/windows/wsl/tutorials/wsl-database>

- **Comandos básicos para WSL:**

<https://learn.microsoft.com/es-es/windows/wsl/basic-commands>

- **Acceso a aplicaciones de red con WSL:**
<https://learn.microsoft.com/es-es/windows/wsl/networking>
- **¿Cómo instalar fácilmente Apache Airflow en Windows?:**
<https://vivekjadhavr.medium.com/how-to-easily-install-apache-airflow-on-windows-6f041c9c80d2>
- **Documentación Psycopg 2.9.10.dev1:**
<https://www.psycopg.org/docs/install.html>
- **Documentación - apache-airflow-providers-postgres package:**
<https://pypi.org/project/apache-airflow-providers-postgres/>
- **Docker Image for Apache Airflow:**
<https://airflow.apache.org/docs/docker-stack/index.html>
- **Jinja Templating:**
<https://airflow.apache.org/docs/apache-airflow/2.9.3/core-concepts/operators.html#concepts-jinjatemplating>