

Airflow

Ejemplo de uso de Airflow



Gutiérrez Villalobos Dayal

Código: 216586382

Computación tolerante a fallos

Lunes y miércoles: 11:00-12:55

Sección D06

Maestro: López Franco Michel Emanuel

Debido a que Airflow está diseñado para trabajar solo en ambientes de Linux, se utilizará la herramienta **wsl**.

Lo primero que se tuvo que hacer es la terminal de Windows correr el comando “**wsl --install**” con el cual nos asegurábamos de poder tener una ambiente Linux en ele que se pueda ejecutar nuestro airflow incluso si es que estamos trabajando desde Windows. Para evitar que algunas descargas se paquetes se vieran afectados se tuvo que habilitar el modo de desarrollador

— □ ×

For developers

These settings are intended for development use only.

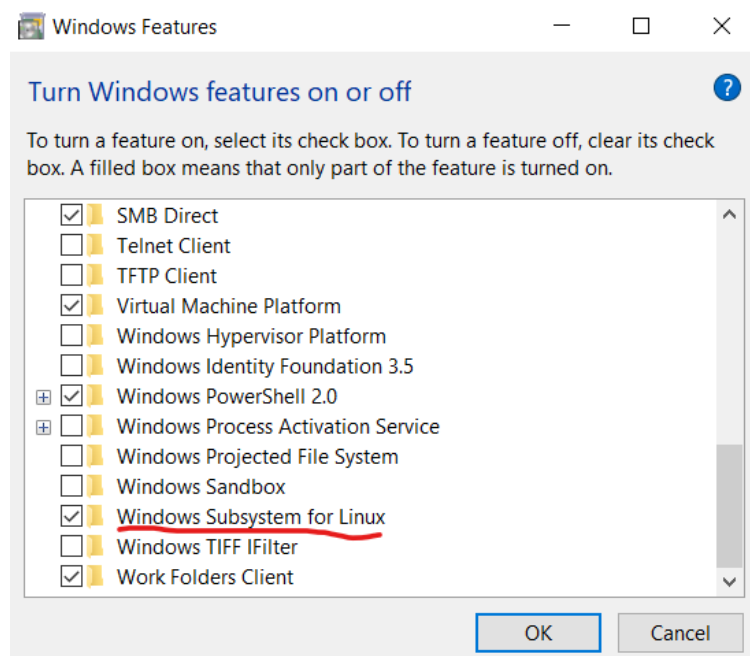
[Learn more](#)

Developer Mode

Install apps from any source, including loose files.

☒ On

Y una vez hecho esto verificar después que el subsistema para Linux esté habilitado



presionado la combinación de teclas Windos+R y escribiendo ahí **optionalfeatures.exe** y dentro de esta pestaña verificar que la casilla correspondiente esté marcada

Para finalizar se tuvo que reiniciar la computadora para que el ambiente se instalara de forma correcta. Una vez hecho esto se abrió una nueva terminal se escribio el comando **bash**, se comenzó a descargar las dependencias correspondientes y se ejecutó nuestro Linux, una vez creado un usuario y una contraseña.

Dentro de la terminal se escribió el comando **sudo nano /etc/wsl.conf** y editando el documento pertinente de la siguiente manera. Esto con el propósito de poder hacer que nuestro disco principal esté montado en el root en lugar de la carpeta /mnt/, para terminar con el proceso se uvo que guardar nuestro achivo con ctrl+s y salir con ctrl+x, después se tuvo que reiniciar Ubuntu verificando que esto se haya cumplido

```
day@DESKTOP-QR90KPG: /c/Users/Dayal/Airdemo
GNU nano 4.8 /etc/wsl.conf
[automount]
root = /
options = "metadata"
```

Después se tuvo que instalr todo lo necesario con los comandos **sudo apt update**, **sudo apt upgrade**, **sudo apt install python3-pip**. Esto para poder tener tanto Python en la última versión como el instalador pip para poder instalar Airflow. Para confirmar que se haya hecho de la forma correcta se corrió el siguiente comando

```
day@DESKTOP-QR90KPG:/c/Users/Dayal/Airdemo$ pip3 --version
pip 20.0.2 from /usr/lib/python3/dist-packages/pip (python 3.8)
```

Para instalar Airflow se hizo con el comando **pip3 install apache-airflow**, sin embargo para poder trabajar con los archivos en Windows se tuvo que correr el comando **nano ~/.bashrc** y agregar lo siguiente para hacer todo se guarde dentro de una carpeta que nosotros digamos capaz de visualizarla desde Windows

```
day@DESKTOP-QR90KPG: /c/Users/Dayal/Airdemo
GNU nano 4.8 /home/day/.bashrc
# ~/.bashrc: executed by bash(1) for non-login shells.
# see /usr/share/doc/bash/examples/startup-files (in the package bash-doc)
# for examples

export AIRFLOW_HOME=/c/Users/Dayal/Airdemo/airflow_
```

Para asegurarnos que todo se instaló de una manera correcta verificamos la versión de airflow de la siguiente manera

```
day@DESKTOP-QR90KPG:/c/Users/Dayal/Airdemo$ airflow version
2.2.4
```

Debido a que para este ejemplo se usará un entorno virtual de python se tuvo que instalar los paquetes necesarios para este propósito con lo siguiente

```
day@DESKTOP-QR90KPG:/c/Users/Dayal/Airdemo$ sudo apt install python3.8-venv
```

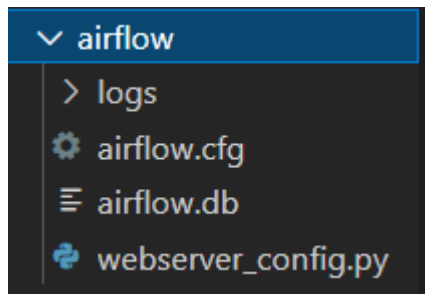
Se creó el entorno virtual y después se activó

```
day@DESKTOP-QR90KPG:/c/Users/Dayal/Airdemo$ python3 -m venv venv
day@DESKTOP-QR90KPG:/c/Users/Dayal/Airdemo$ source venv/bin/activate
(venv) day@DESKTOP-QR90KPG:/c/Users/Dayal/Airdemo$
```

Se inicializó la base de datos

```
day@DESKTOP-QR90KPG:/c/Users/Dayal/Airdemo$ airflow db init
DB: sqlite:///c/Users/Dayal/Airdemo/airflow/airflow.db
[2022-03-13 21:46:25,516] {db.py:919} INFO - Creating tables
INFO [alembic.runtime.migration] Context impl SQLiteImpl.
INFO [alembic.runtime.migration] Will assume non-transactional DDL.
INFO [alembic.runtime.migration] Running upgrade -> e3a246e0dc1, current schema
INFO [alembic.runtime.migration] Running upgrade e3a246e0dc1 -> 1507a7289a2f, create is_encrypted
/usr/local/lib/python3.8/dist-packages/alembic/ddl/sqlite.py:74 UserWarning: Skipping unsupported ALTER
for creation of implicit constraint. Please refer to the batch mode feature which allows for SQLite migr
ations using a copy-and-move strategy.
```

Creando ciertos archivos

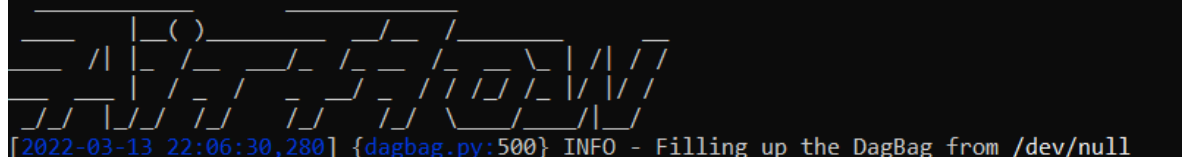


Se crea un usuario

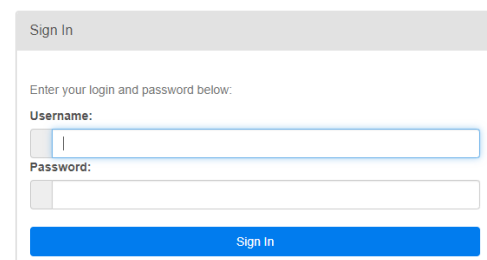
```
day@DESKTOP-QR90KPG:/c/Users/Dayal/Airdemo$ airflow users create --username dayal --firstname Dayal --last
name Gutierrez --role Admin --password contra --email > --email dayal.gutierrez5863@alumnos.udg.mx
```

Se inicia el servidor web

```
day@DESKTOP-QR90KPG:/c/Users/Dayal$ airflow webserver --port 8080
```



Y se entra con las credenciales que ya se han creado previamente



Sign In

Enter your login and password below:

Username:

Password:

Sign In

Una vez dentro podemos observar cómo es que hay algunos ejemplos

localhost:8080/home

🔍

🛡️

🔗

Airflow

DAGs

Security

Browse

Admin

Docs

04:11 UTC

DG

DAGs

All 30

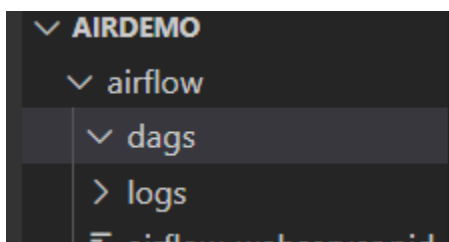
Active 0

Paused 30

Filter DAGs by tag

Search DAGs

DAG	Owner	Runs	Schedule	Last Run	Next Run
<div><div></div><div>example_bash_operator</div><div>example example2</div></div>	airflow	<div><div></div><div></div><div></div><div></div></div>	0 0 ***	2022-03-13, 00:00:00	
<div><div></div><div>example_branch_datetime_operator_2</div><div>example</div></div>	airflow	<div><div></div><div></div><div></div><div></div></div>	@daily	2022-03-13, 00:00:00	
<div><div></div><div>example_branch_dop_operator_v3</div><div>example</div></div>	airflow	<div><div></div><div></div><div></div><div></div></div>	*1 * * * *	2022-03-14, 04:06:00	
<div><div></div><div>example_branch_labels</div></div>	airflow	<div><div></div><div></div><div></div><div></div></div>	@daily	2022-03-13, 00:00:00	
<div><div></div><div>example_branch_operator</div><div>example example2</div></div>	airflow	<div><div></div><div></div><div></div><div></div></div>	@daily	2022-03-13, 00:00:00	



Dentro de la carpeta en donde estamos trabajando se creó una carpeta llamada dags en donde se guardarán todos los pipelines que queremos que se ejecuten.

Antes de poder comenzar a hacer nuestro script con nuestras funciones lo que se hará es empezar desde antes el scheduler

```
C:\Users\Dayal>bash
day@DESKTOP-QR90K6G:/c/Users/Dayal$ airflow scheduler

      ____          _              _
     / __ \        | |            | |
    / /_/_         | |            | |
   / ___ \         | |            | |
  / /___ \         | |            | |
 /_____\ \        | |            | |
/_____\"_/       |_|            |_|

[2022-03-13 22:07:12 -0600] [47] [INFO] Starting gunicorn 20.1.0
[2022-03-13 22:07:12 -0600] [47] [INFO] Listening at: http://0.0.0.0:8793 (47)
[2022-03-13 22:07:12 -0600] [47] [INFO] Using worker: sync
[2022-03-13 22:07:12 -0600] [48] [INFO] Booting worker with pid: 48
[2022-03-13 22:07:12,086] {scheduler_job.py:619} INFO - Starting the scheduler
[2022-03-13 22:07:12,086] {scheduler_job.py:624} INFO - Processing each file at most -1 times
[2022-03-13 22:07:12,088] {manager.py:163} INFO - Launched DagFileProcessorManager with pid: 49
[2022-03-13 22:07:12,089] {scheduler_job.py:1137} INFO - Resetting orphaned tasks for active dag runs
[2022-03-13 22:07:12,092] {settings.py:55} INFO - Configured default timezone Timezone('UTC')
[2022-03-13 22:07:12,108] {manager.py:441} WARNING - Because we cannot use more than 1 thread (parsing_processes = 2)
en using sqlite. So we set parallelism to 1.
[2022-03-13 22:07:12 -0600] [50] [INFO] Booting worker with pid: 50
[2022-03-13 22:12:12,260] {scheduler_job.py:1137} INFO - Resetting orphaned tasks for active dag runs
[2022-03-13 22:17:12,399] {scheduler_job.py:1137} INFO - Resetting orphaned tasks for active dag runs
[2022-03-13 22:22:12,564] {scheduler_job.py:1137} INFO - Resetting orphaned tasks for active dag runs
[2022-03-13 22:27:12,725] {scheduler_job.py:1137} INFO - Resetting orphaned tasks for active dag runs
[2022-03-13 22:32:12,889] {scheduler_job.py:1137} INFO - Resetting orphaned tasks for active dag runs
[2022-03-13 22:37:13,050] {scheduler_job.py:1137} INFO - Resetting orphaned tasks for active dag runs
[2022-03-13 22:42:13,081] {scheduler_job.py:1137} INFO - Resetting orphaned tasks for active dag runs
[2022-03-13 22:47:13,181] {scheduler_job.py:1137} INFO - Resetting orphaned tasks for active dag runs
[2022-03-13 22:51:53 -0600] [47] [INFO] Handling signal: winch
```

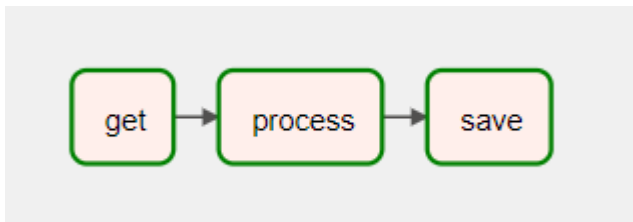
Se creó un pequeño script en el que se ponen los parámetros requeridos para que se tengan todos los datos acerca de los dags

Una vez que el script está completo y funcionando de la manera correcta con el scheduler corriendo podemos observar en la interfaz web cómo es que efectivamente nuestro DAG se ha reconocido

Si nos metemos a observar cómo es que se forma la estructura de árbol de nuestro DAG nos arroja lo siguiente

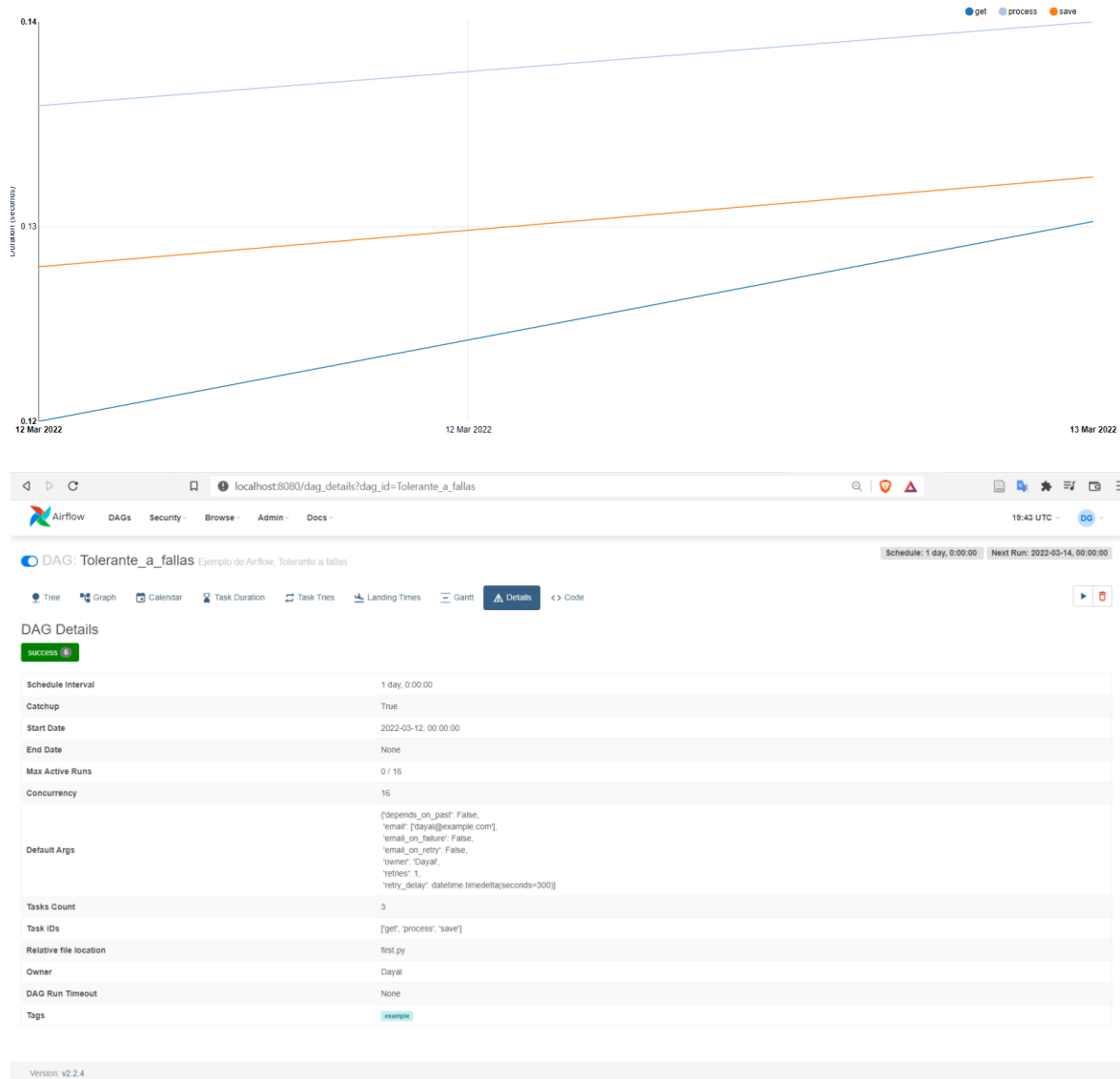


En donde se puede apreciar cómo es que se tienen que realizar las acciones para poder completarse con éxito



Pudiéndolo incluso llegar a observar como un grafo.

Llegando a tener información incluso acerca de el tiempo que se toma realizar cada tarea o parte de nuestro diagrama



Conclusión

La creación de este trabajo fue un poco complejo debido a que Airflow solo funciona en un ambiente de Linux por lo que para tener la configuración correcta en un máquina que tiene Windows es un poco más complicado y más si es que no se tiene un conocimiento aunque sea básico sobre Linux y cómo es que nos podemos mover entre los archivos en una terminal, en si el orquestador de Airflow, me gustó, siento que es una muy herramienta muy útil para poder tener bien controlados todos los procesos que se necesitan llevar a cabo además que gracias a la interfaz gráfica poder tener un control y entendimiento amplio sobre los mismos resulta de una manera mucho más sencillo a comparación de la librería Prefect. Otra cosa buena es que al ser de apache y de código libre se pueden llegar a encontrar con facilidad la respuesta a problemáticas que puedan surgir.

Bibliografía

- Microsoft. (2022, 24 febrero). Install WSL. Microsoft Docs. Recuperado 13 de marzo de 2022, de <https://docs.microsoft.com/en-us/windows/wsl/install>
- Microsoft. (2016, 11 abril). How to: Enable the Windows Subsystem for Linux. Microsoft answer. Recuperado 13 de marzo de 2022, de https://login.live.com/login.srf?wa=wsignin1.0&rpsnv=13&checkda=1&ct=1647288060&rver=6.7.6643.0&wp=MBI_SSL&wreply=https:%2F%2Fanswers.microsoft.com%2Fen-us%2Fsite%2Fcompletesignin%3Fsilent%3DTrue%26returnUrl%3Dhttps%2F53A%252F%252Fanswers.microsoft.com%252Fen-us%252Finsider%252Fforum%252Fall%252Fhow-to-enable-the-windows-subsystem-for-linux%252F16e8f2e8-4a6a-4325-a89a-fd28c7841775&id=273572&aadredir=1
- Astronomer. (s. f.). Running Airflow on Windows 10 & WSL. Recuperado 13 de marzo de 2022, de <https://www.astronomer.io/guides/airflow-wsl/>