

**Airflow is a platform to programmatically author,
schedule and monitor workflows.**



Nombre del alumno: José Ulises Vallejo Sierra

Código: 219747905

Sección: D06

Curso: Computación Tolerante a Fallas

Nombre del profesor: Michel Emanuel López Franco

Realizar un programa que sea capaz de revisar el estatus de tu aplicación.

Técnica a utilizar: [Airflow](#)

En esta ocasión se nos plantea un tema actual y novedoso que promete ser solución a diversos problemas o fallas en la computación y es una rama de la ingeniería que apoya a la tolerancia de estas fallas para mitigar los problemas presentados en las aplicaciones, se trata de la gestión de el flujo de trabajo, y en esta ocasión específicamente con la utilización del módulo airflow que nos ayudará precisamente con este rubro en el lenguaje Python, cabe destacar que apache airflow no está definido para Windows, por lo que se requiere de un interprete de Linux que nos facilitará su implementación.

Programa: Creación de una DAG

Usaremos Airflow para completar una tarea relativamente simple hoy: imprimir un mensaje de hola mundo para comprobar que la tarea se haya creado y sea capaz de visualizarse en el localhost:8080 de nuestras DAGs.

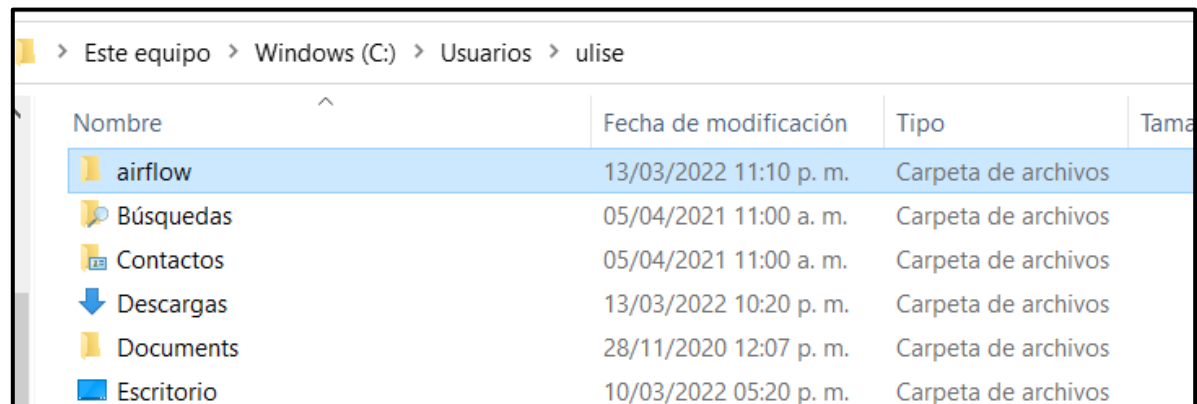
Para la instalación del subsistema de linux para windows, abrimos el powershell y ejecutamos el siguiente comando como administrador.

```
PS C:\WINDOWS\system32> dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart
Deployment Image Servicing and Management tool
Version: 10.0.19041.844

Image Version: 10.0.19044.1586

Enabling feature(s)
[=====100.0%=====]
The operation completed successfully.
```

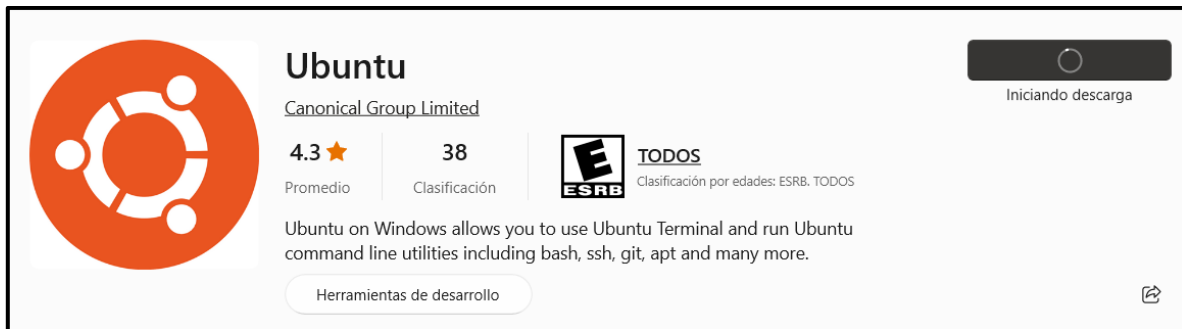
Posteriormente creamos la carpeta airflow en la unidad C: / Usuarios / ulise para alojar los dags a crear con el airflow



The screenshot shows a Windows File Explorer window with the address bar displaying the path: > Este equipo > Windows (C:) > Usuarios > ulise. The main area shows a list of folders and files. The 'airflow' folder is highlighted in blue. Below it are folders named 'Búsquedas', 'Contactos', 'Descargas', 'Documents', and 'Escritorio'.

Nombre	Fecha de modificación	Tipo	Tamaño
airflow	13/03/2022 11:10 p. m.	Carpeta de archivos	
Búsquedas	05/04/2021 11:00 a. m.	Carpeta de archivos	
Contactos	05/04/2021 11:00 a. m.	Carpeta de archivos	
Descargas	13/03/2022 10:20 p. m.	Carpeta de archivos	
Documents	28/11/2020 12:07 p. m.	Carpeta de archivos	
Escritorio	10/03/2022 05:20 p. m.	Carpeta de archivos	

Descargamos en Microsoft Store Ubuntu que servirá como distribución de Linux para Windows.



```
Seleccinar airflow@LAPTOP-HAOQCHEM: ~
Installing, this may take a few minutes...
Please create a default UNIX user account. The username does not need to match your Windows username.
For more information visit: https://aka.ms/wslusers
Enter new UNIX username: airflow
New password:
Retype new password:
passwd: password updated successfully
Installation successful!
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 4.4.0-19041-Microsoft x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/advantage

System information as of Sun Mar 13 23:59:22 CST 2022

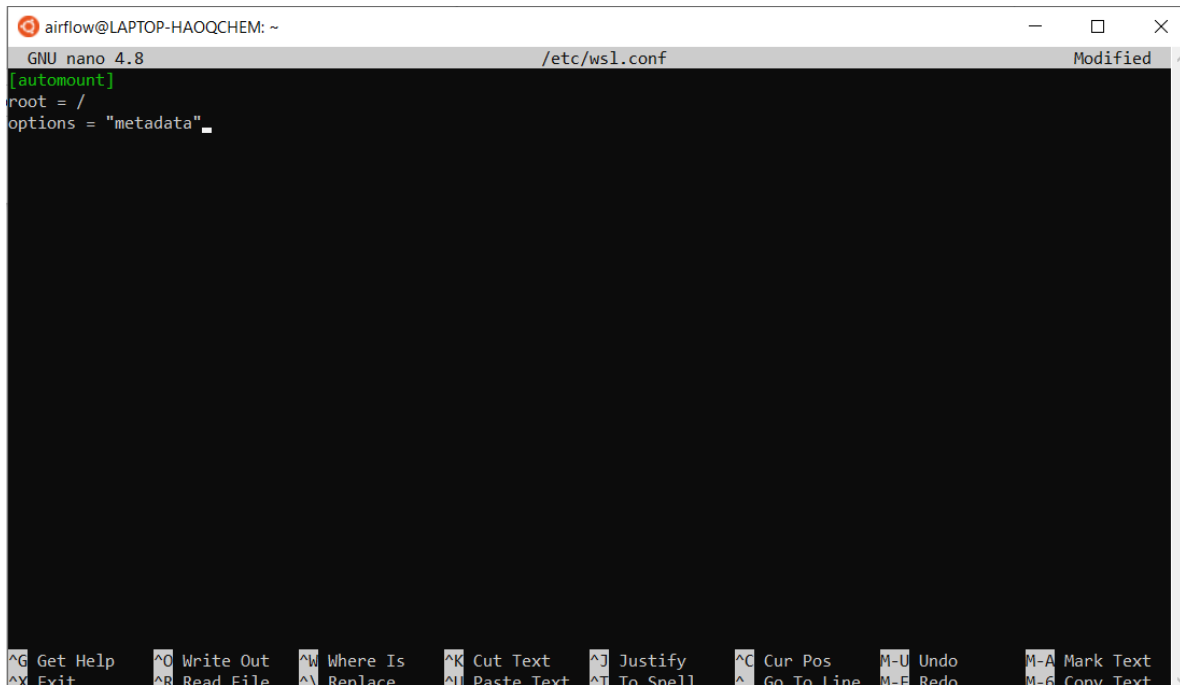
System load:  0.52      IPv4 address for eth1: 192.168.134.1
Usage of /home: unknown IPv4 address for eth2: 192.168.19.1
Memory usage: 71%      IPv4 address for eth3: 192.168.56.1
Swap usage:   0%        IPv4 address for wifi0: 192.168.100.18
Processes:    7          IPv6 address for wifi0: 2806:261:417:5a62::1
Users logged in: 0

1 update can be applied immediately.
To see these additional updates run: apt list --upgradable
```

Para asegurarnos de que todos los componentes de Ubuntu estén actualizados ejecutamos el siguiente comando

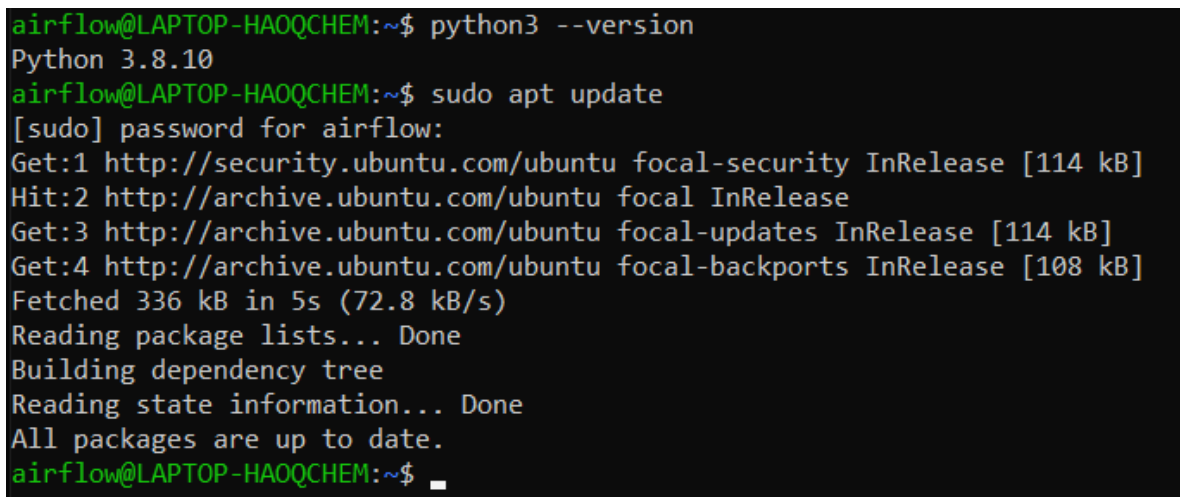
```
airflow@LAPTOP-OMJ7SHPI:~$ sudo apt update && sudo apt upgrade
[sudo] password for airflow:
Get:1 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Hit:2 http://archive.ubuntu.com/ubuntu focal InRelease
Get:3 http://archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:4 http://archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]
Get:5 http://security.ubuntu.com/ubuntu focal-security/main amd64 Packages [1317 kB]
Get:6 http://archive.ubuntu.com/ubuntu focal/universe amd64 Packages [8628 kB]
Get:7 http://security.ubuntu.com/ubuntu focal-security/main Translation-en [231 kB]
Get:8 http://security.ubuntu.com/ubuntu focal-security/main amd64 c-n-f Metadata [9808 B]
Get:9 http://security.ubuntu.com/ubuntu focal-security/restricted amd64 Packages [799 kB]
Get:10 http://security.ubuntu.com/ubuntu focal-security/restricted Translation-en [114 kB]
Get:11 http://security.ubuntu.com/ubuntu focal-security/universe amd64 Packages [692 kB]
Get:12 http://security.ubuntu.com/ubuntu focal-security/universe Translation-en [121 kB]
Get:13 http://security.ubuntu.com/ubuntu focal-security/universe amd64 c-n-f Metadata [14.0 kB]
```

Abrimos el `sudo nano /etc/wsl/.conf` para ajustar con el editor nano para anular la configuración de montaje y poder trabajar por lo que escribimos las siguientes tres líneas de código.



```
airflow@LAPTOP-HAOQCHEM: ~
GNU nano 4.8 /etc/wsl.conf Modified
[automount]
root = /
options = "metadata"
```

Corroboramos nuestra versión de Python y ejecutamos el comando `sudo apt update` para asegurarnos de que todo se mantenga actualizado.



```
airflow@LAPTOP-HAOQCHEM:~$ python3 --version
Python 3.8.10
airflow@LAPTOP-HAOQCHEM:~$ sudo apt update
[sudo] password for airflow:
Get:1 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Hit:2 http://archive.ubuntu.com/ubuntu focal InRelease
Get:3 http://archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:4 http://archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]
Fetched 336 kB in 5s (72.8 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
All packages are up to date.
airflow@LAPTOP-HAOQCHEM:~$
```

Instalación el gestor de paquetes de Python3 “pip”.

```

airflow@LAPTOP-HAOQCHEM:~$ sudo apt install python3-pip
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  libfwupdplugin1
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu build-essential cpp cpp-9 dpkg-dev fakeroot g++ g++-9 gcc gcc-9
  gcc-9-base libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl libasan5 libatomic1 libbinutils
  libc-dev-bin libc6-dev libcc1-0 libcrypt-dev libctf-nobfd0 libctf0 libdpkg-perl libexpat1-dev libfakeroot
  libfile-fcntllock-perl libgcc-9-dev libgomp1 libisl22 libitm1 liblsan0 libmpc3 libpython3-dev libpython3.8-dev
  libquadmath0 libstdc++-9-dev libtsan0 libubsan0 linux-libc-dev make manpages-dev python-pip-whl python3-dev
  python3-wheel python3.8-dev zlib1g-dev
Suggested packages:

```

Instalamos apache-airflow con los parámetros determinados para nuestro contexto

```

airflow@LAPTOP-HAOQCHEM:~$ pip3 install apache-airflow[gcp,statsd,sentry]==1.10.10
Collecting apache-airflow[gcp,sentry,statsd]==1.10.10
  Downloading apache_airflow-1.10.10-py2.py3-none-any.whl (4.7 MB)
    | 4.7 MB 2.1 MB/s
Collecting iso8601>=0.1.12
  Downloading iso8601-1.0.2-py3-none-any.whl (9.7 kB)
Collecting text-unidecode==1.2
  Downloading text_unidecode-1.2-py2.py3-none-any.whl (77 kB)
    | 77 kB 4.9 MB/s
Collecting lazy-object-proxy~=1.3
  Downloading lazy_object_proxy-1.7.1-cp38-cp38-manylinux_2_5_x86_64.manylinux1_x86_64.whl (60 kB)
    | 60 kB 3.6 MB/s
Collecting pandas<1.0.0,>=0.17.1
  Downloading pandas-0.25.3-cp38-cp38-manylinux1_x86_64.whl (10.4 MB)
    | 10.4 MB 75 kB/s
Collecting python-dateutil<3,>=2.3
  Downloading python_dateutil-2.8.2-py2.py3-none-any.whl (247 kB)
    | 247 kB 6.0 MB/s
Collecting future<0.19,>=0.16.0

```

Además, instalamos dos módulos que necesitamos para el pip

```

airflow@LAPTOP-HAOQCHEM:~$ pip install cryptography==2.9.2
Collecting cryptography==2.9.2
  Downloading cryptography-2.9.2-cp35-abi3-manylinux2010_x86_64.whl (2.7 MB)
    | 2.7 MB 2.2 MB/s
Collecting cffi!>=1.11.3,>=1.8
  Downloading cffi-1.15.0-cp38-cp38-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (446 kB)
    | 446 kB 14.1 MB/s
Requirement already satisfied: six>=1.4.1 in /usr/lib/python3/dist-packages (from cryptography==2.9.2) (1.14.0)
Collecting pycparser
  Downloading pycparser-2.21-py2.py3-none-any.whl (118 kB)
    | 118 kB 16.1 MB/s
Installing collected packages: pycparser, cffi, cryptography

```

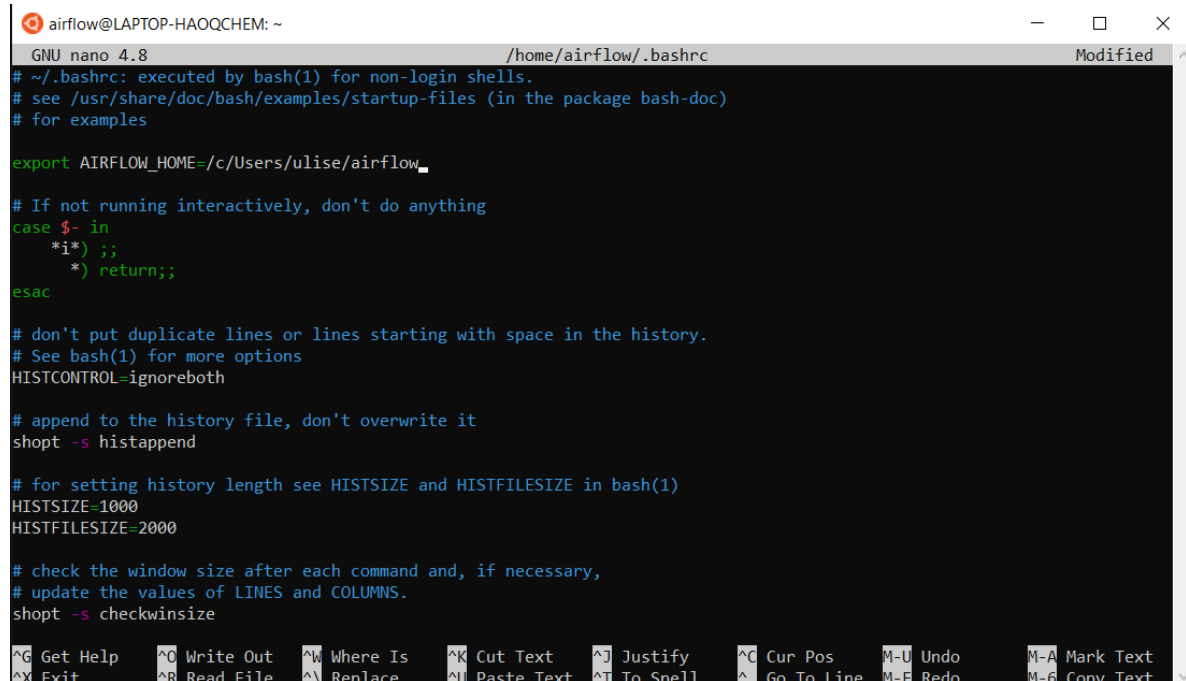
```

airflow@LAPTOP-HAOQCHEM:~$ pip3 install pyspark==2.4.5
Collecting pyspark==2.4.5
  Downloading pyspark-2.4.5.tar.gz (217.8 MB)
    | 217.8 MB 9.6 kB/s
Collecting py4j==0.10.7
  Downloading py4j-0.10.7-py2.py3-none-any.whl (197 kB)
    | 197 kB 650 kB/s
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py) ... done
  Created wheel for pyspark: filename=pyspark-2.4.5-py2.py3-none-any.whl size=218257927 sha256=fa5eb79adc5be2041eb0c66
  48965f84261965b39148b3521e59c70ddd131bb
  Stored in directory: /home/airflow/.cache/pip/wheels/40/1b/56/aa5b76fa0c55166784b69226bcf72e3480d08d248a16f0b15c
Successfully built pyspark
Installing collected packages: py4j, pyspark
Successfully installed py4j-0.10.7 pyspark-2.4.5
airflow@LAPTOP-HAOQCHEM:~$

```

Exportamos la ruta de la carpeta que creamos en el segundo paso tanto en el Ubuntu con el comando `export` como en el bash con el comando `nano ~/.bashrc` para agregar las variables de entorno y reiniciamos.

```
airflow@LAPTOP-HAOQCHEM:~$ export AIRFLOW_HOME=/c/Users/ulise/airflow
```



```
airflow@LAPTOP-HAOQCHEM: ~
GNU nano 4.8 /home/airflow/.bashrc Modified
# ~/.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/ulise/airflow_

# If not running interactively, don't do anything
case $- in
  *i*) ;;
  *) return;;
esac

# don't put duplicate lines or lines starting with space in the history.
# See bash(1) for more options
HISTCONTROL=ignoreboth

# append to the history file, don't overwrite it
shopt -s histappend

# for setting history length see HISTSIZE and HISTFILESIZE in bash(1)
HISTSIZE=1000
HISTFILESIZE=2000

# check the window size after each command and, if necessary,
# update the values of LINES and COLUMNS.
shopt -s checkwinsize

^G Get Help      ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify      ^C Cur Pos      M-U Undo        M-A Mark Text
^X Exit          ^R Read File    ^N Replace      ^U Paste Text   ^T To Spell     ^G Go To Line   M-E Redo        M-G Copy Text
```

Ahora comprobamos nuestra versión de airflow y vemos que no arroje errores

```
airflow@LAPTOP-HAOQCHEM:~$ airflow version
1.10.10
```

Inicializamos la base de datos de nuestra versión de airflow con este comando

```
airflow@LAPTOP-HAOQCHEM:~$ airflow initdb
DB: sqlite:///c:/Users/ulise/airflow/airflow.db
[2022-03-14 01:45:01.569] {db.py:378} 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
/home/airflow/.local/lib/python3.8/site-packages/alembic/ddl/sqlite.py:74: UserWarning: Skipping unsupported ALTER for
reation of implicit constraint. Please refer to the batch mode feature which allows for SQLite migrations using a copy-
nd-move strategy.
  util.warn(
INFO [alembic.runtime.migration] Running upgrade 1507a7289a2f -> 13eb55f81627, maintain history for compatibility with
earlier migrations
INFO [alembic.runtime.migration] Running upgrade 13eb55f81627 -> 338e90f54d61, More logging into task_instance
INFO [alembic.runtime.migration] Running upgrade 338e90f54d61 -> 52d714495f0, job_id indices
INFO [alembic.runtime.migration] Running upgrade 52d714495f0 -> 502898887f84, Adding extra to Log
INFO [alembic.runtime.migration] Running upgrade 502898887f84 -> 1b38cef5b76e, add dagrun
INFO [alembic.runtime.migration] Running upgrade 1b38cef5b76e -> 2e541a1dcfed, task_duration
INFO [alembic.runtime.migration] Running upgrade 2e541a1dcfed -> 40e67319e3a9, dagrun_config
INFO [alembic.runtime.migration] Running upgrade 40e67319e3a9 -> 561833c1c74b, add password column to user
INFO [alembic.runtime.migration] Running upgrade 561833c1c74b -> 4446e08588, dagrun start end
INFO [alembic.runtime.migration] Running upgrade 4446e08588 -> bbc73705a13e, Add notification_sent column to sla_miss
INFO [alembic.runtime.migration] Running upgrade bbc73705a13e -> bba5a7cfc896, Add a column to track the encryption st
te of the 'Extra' field in connection
INFO [alembic.runtime.migration] Running upgrade bba5a7cfc896 -> 1968acfc09e3, add is_encrypted column to variable tab
```

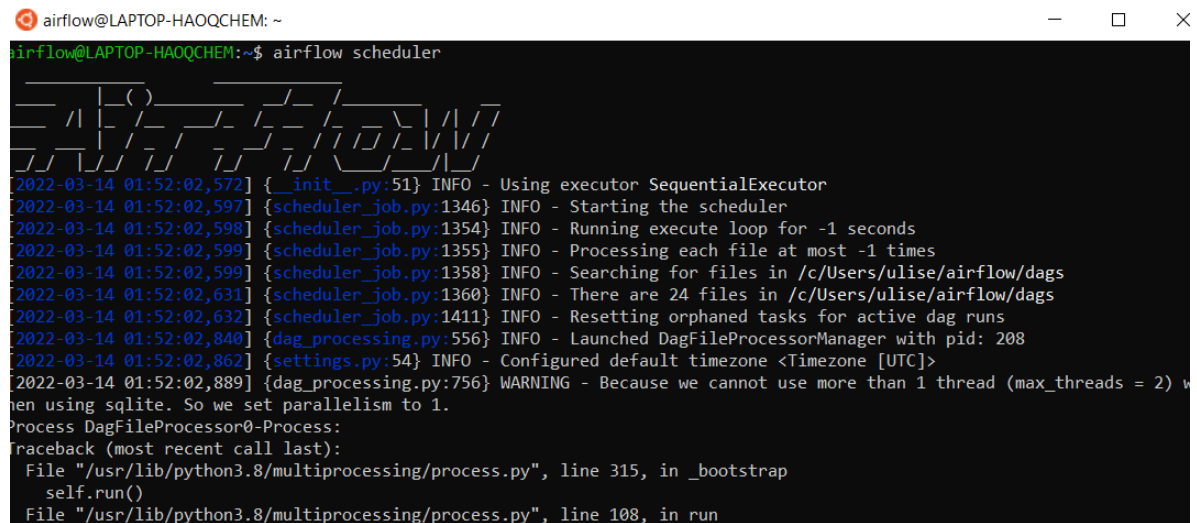

Corremos el apache airflow con el comando `airflow webserver` para ejecutar el servidor en `localhost:8080` y en otra ventana ejecutamos el comando `airflow scheduler`

```
airflow@LAPTOP-QM175MPI:~$ airflow webserver
[2022-03-14 13:53:45,627] {dagbag.py:500} INFO - Filling up the DagBag from /dev/null
[2022-03-14 13:53:45,839] {manager.py:779} WARNING - No user yet created, use flask fab command to do it.
[2022-03-14 13:53:46,254] {manager.py:496} INFO - Created Permission View: menu access on List Users
[2022-03-14 13:53:46,320] {manager.py:558} INFO - Added Permission menu access on List Users to role Admin
[2022-03-14 13:53:46,468] {manager.py:496} INFO - Created Permission View: menu access on Security
[2022-03-14 13:53:46,538] {manager.py:558} INFO - Added Permission menu access on Security to role Admin
[2022-03-14 13:53:46,719] {manager.py:496} INFO - Created Permission View: menu access on List Roles
[2022-03-14 13:53:46,769] {manager.py:558} INFO - Added Permission menu access on List Roles to role Admin
[2022-03-14 13:53:46,951] {manager.py:496} INFO - Created Permission View: can read on User Stats Chart
[2022-03-14 13:53:47,051] {manager.py:558} INFO - Added Permission can read on User Stats Chart to role Admin
[2022-03-14 13:53:47,193] {manager.py:496} INFO - Created Permission View: menu access on User's Statistics
[2022-03-14 13:53:47,262] {manager.py:558} INFO - Added Permission menu access on User's Statistics to role Admin
[2022-03-14 13:53:47,461] {manager.py:496} INFO - Created Permission View: menu access on Base Permissions
[2022-03-14 13:53:47,498] {manager.py:558} INFO - Added Permission menu access on Base Permissions to role Admin
[2022-03-14 13:53:47,658] {manager.py:496} INFO - Created Permission View: can read on View Menus
[2022-03-14 13:53:47,707] {manager.py:558} INFO - Added Permission can read on View Menus to role Admin
[2022-03-14 13:53:47,808] {manager.py:496} INFO - Created Permission View: menu access on Views/Menus
[2022-03-14 13:53:47,853] {manager.py:558} INFO - Added Permission menu access on Views/Menus to role Admin
[2022-03-14 13:53:47,987] {manager.py:496} INFO - Created Permission View: can read on Permission Views
```

Como observamos, en el browser ya se ejecuta normalmente el servidor con las dags de ejemplo, pero nos salta la advertencia o error de Schedule ya que aun no lo hemos implementado

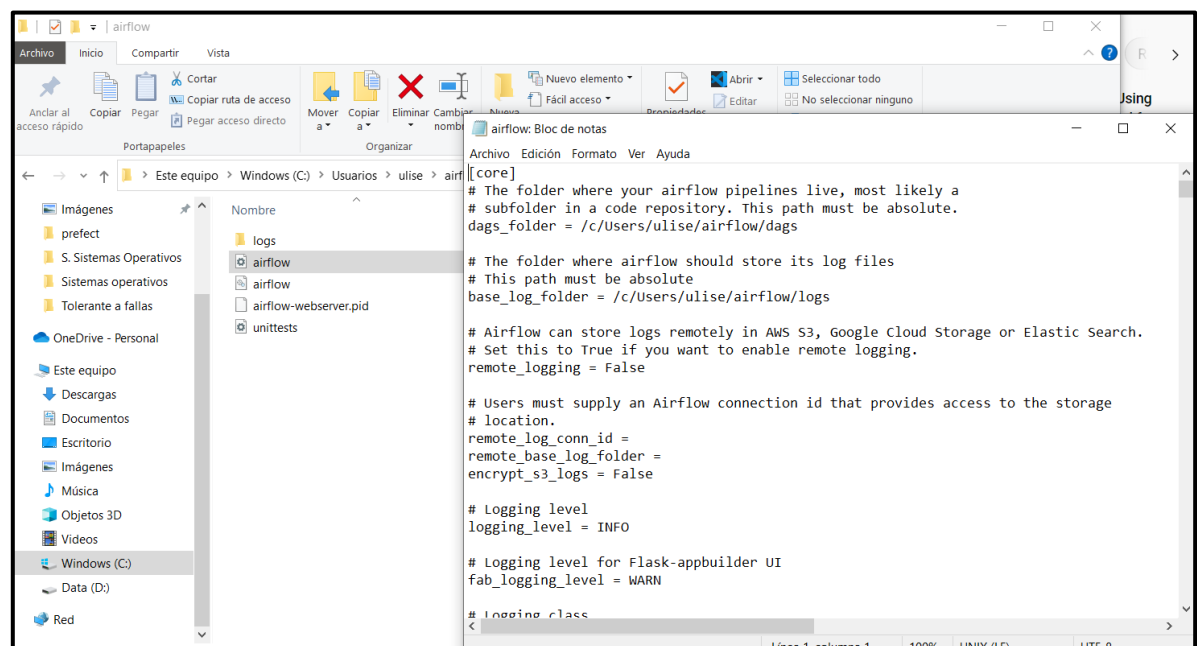
DAG	Schedule	Owner	Recent Tasks	Last Run	DAG Runs	Links
example_bash_operator	None	Airflow				
example_branch_dop_operator_v3	None	Airflow				
example_branch_operator	None	Airflow				
example_complex	None	airflow				
example_external_task_marker_child	None	airflow				
example_external_task_marker_parent	None	airflow				
example_http_operator	None	Airflow				
example_nested_branch_dag	None	airflow				
example_passing_params_via_test_command	None	airflow				
example_pig_operator	None	Airflow				
example_python_operator	None	Airflow				

Como observamos en la siguiente imagen, al ejecutar el Scheduler nos arroja algunos errores los cuales se solucionan de la siguiente manera



```
airflow@LAPTOP-HAOQCHEM: ~  
airflow@LAPTOP-HAOQCHEM:~$ airflow scheduler  
  
[2022-03-14 01:52:02,572] {__init__.py:51} INFO - Using executor SequentialExecutor  
[2022-03-14 01:52:02,597] {scheduler_job.py:1346} INFO - Starting the scheduler  
[2022-03-14 01:52:02,598] {scheduler_job.py:1354} INFO - Running execute loop for -1 seconds  
[2022-03-14 01:52:02,599] {scheduler_job.py:1355} INFO - Processing each file at most -1 times  
[2022-03-14 01:52:02,599] {scheduler_job.py:1358} INFO - Searching for files in /c/Users/ulise/airflow/dags  
[2022-03-14 01:52:02,631] {scheduler_job.py:1360} INFO - There are 24 files in /c/Users/ulise/airflow/dags  
[2022-03-14 01:52:02,632] {scheduler_job.py:1411} INFO - Resetting orphaned tasks for active dag runs  
[2022-03-14 01:52:02,840] {dag_processing.py:556} INFO - Launched DagFileProcessorManager with pid: 208  
[2022-03-14 01:52:02,862] {settings.py:54} INFO - Configured default timezone <Timezone [UTC]>  
[2022-03-14 01:52:02,889] {dag_processing.py:756} WARNING - Because we cannot use more than 1 thread (max_threads = 2) w  
hen using sqlite. So we set parallelism to 1.  
Process DagFileProcessor0-Process:  
Traceback (most recent call last):  
  File "/usr/lib/python3.8/multiprocessing/process.py", line 315, in _bootstrap  
    self.run()  
  File "/usr/lib/python3.8/multiprocessing/process.py", line 108, in run
```

Abrimos con ayuda de bloc de notas dentro de nuestra carpeta de Airflow el archivo airflow con extensión cfg



Buscamos la línea `load_examples` y cambiamos el valor de verdadero a falso, guardamos cambios y volvemos a ejecutar el servidor con `airflow webserver` y `airflow scheduler`

```
# Are DAGs paused by default at creation
days_are_paused_at_creation = True

# The maximum number of active DAG runs per DAG
max_active_runs_per_dag = 16

# Whether to load the DAG examples that ship with Airflow. It's good to
# get started, but you probably want to set this to False in a production
# environment
load_examples = False
```

```
airflow@LAPTOP-OMJ75MPI:~$ airflow webserver

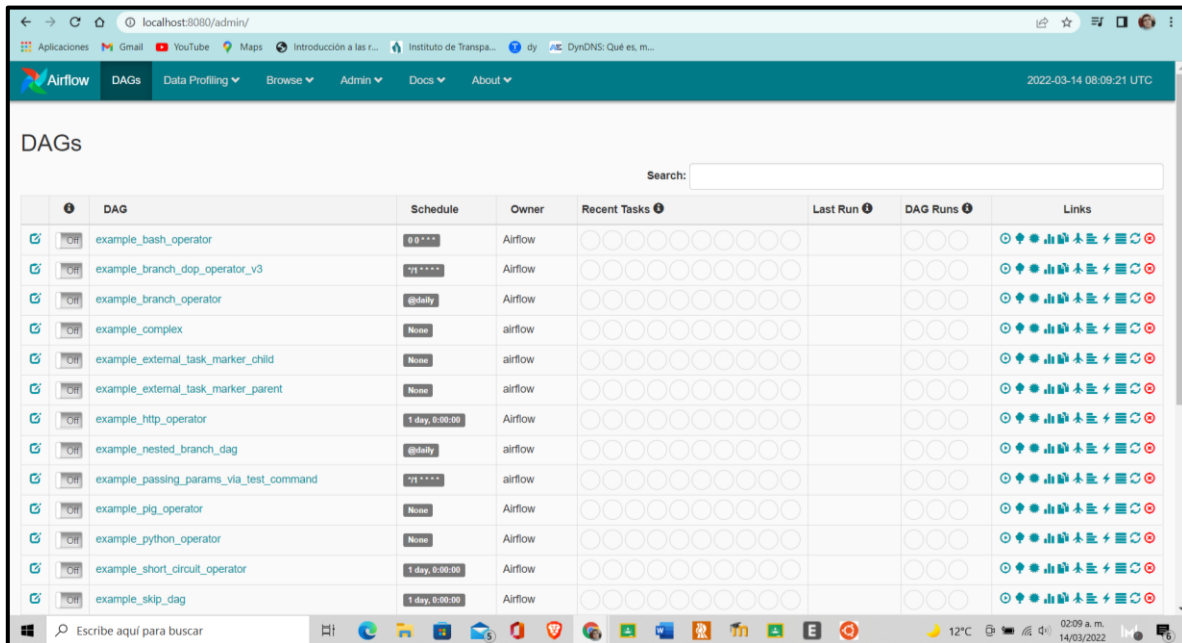
[2022-03-14 13:53:45,627] {dagbag.py:500} INFO - Filling up the DagBag from /dev/null
[2022-03-14 13:53:45,839] {manager.py:779} WARNING - No user yet created, use flask fab command to do it.
[2022-03-14 13:53:46,254] {manager.py:496} INFO - Created Permission View: menu access on List Users
[2022-03-14 13:53:46,320] {manager.py:558} INFO - Added Permission menu access on List Users to role Admin
[2022-03-14 13:53:46,468] {manager.py:496} INFO - Created Permission View: menu access on Security
[2022-03-14 13:53:46,538] {manager.py:558} INFO - Added Permission menu access on Security to role Admin
[2022-03-14 13:53:46,719] {manager.py:496} INFO - Created Permission View: menu access on List Roles
[2022-03-14 13:53:46,769] {manager.py:558} INFO - Added Permission menu access on List Roles to role Admin
[2022-03-14 13:53:46,951] {manager.py:496} INFO - Created Permission View: can read on User Stats Chart
[2022-03-14 13:53:47,051] {manager.py:558} INFO - Added Permission can read on User Stats Chart to role Admin
[2022-03-14 13:53:47,193] {manager.py:496} INFO - Created Permission View: menu access on User's Statistics
[2022-03-14 13:53:47,262] {manager.py:558} INFO - Added Permission menu access on User's Statistics to role Admin
[2022-03-14 13:53:47,481] {manager.py:496} INFO - Created Permission View: menu access on Base Permissions
[2022-03-14 13:53:47,498] {manager.py:558} INFO - Added Permission menu access on Base Permissions to role Admin
[2022-03-14 13:53:47,658] {manager.py:496} INFO - Created Permission View: can read on View Menus
[2022-03-14 13:53:47,707] {manager.py:558} INFO - Added Permission can read on View Menus to role Admin
[2022-03-14 13:53:47,988] {manager.py:496} INFO - Created Permission View: menu access on Views/Menus
[2022-03-14 13:53:47,853] {manager.py:558} INFO - Added Permission menu access on Views/Menus to role Admin
[2022-03-14 13:53:47,987] {manager.py:496} INFO - Created Permission View: can read on Permission Views
```

```
airflow@LAPTOP-HAOQCHEM: ~
airflow@LAPTOP-HAOQCHEM:~$ airflow scheduler

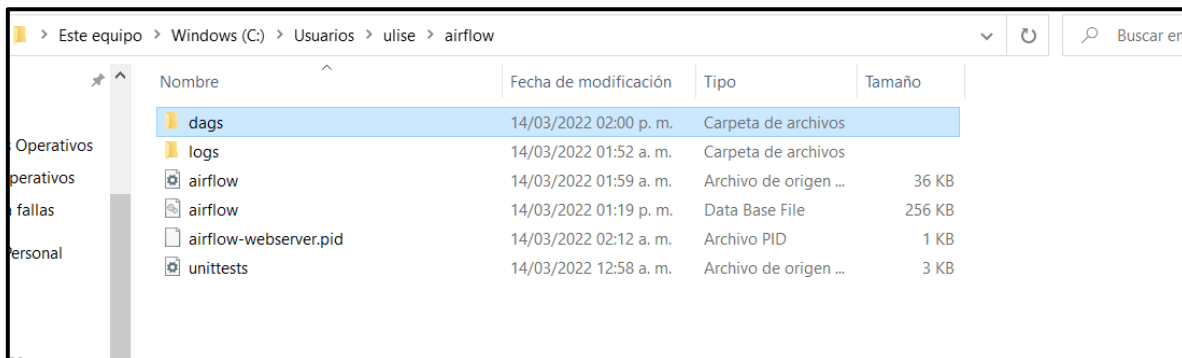
[2022-03-14 02:07:30,869] {__init__.py:51} INFO - Using executor SequentialExecutor
[2022-03-14 02:07:30,886] {scheduler_job.py:1346} INFO - Starting the scheduler
[2022-03-14 02:07:30,886] {scheduler_job.py:1354} INFO - Running execute loop for -1 seconds
[2022-03-14 02:07:30,887] {scheduler_job.py:1355} INFO - Processing each file at most -1 times
[2022-03-14 02:07:30,887] {scheduler_job.py:1358} INFO - Searching for files in /c/Users/ulise/airflow/dags
[2022-03-14 02:07:30,888] {scheduler_job.py:1360} INFO - There are 0 files in /c/Users/ulise/airflow/dags
[2022-03-14 02:07:30,888] {scheduler_job.py:1411} INFO - Resetting orphaned tasks for active dag runs
[2022-03-14 02:07:31,008] {dag_processing.py:556} INFO - Launched DagFileProcessorManager with pid: 81
[2022-03-14 02:07:31,020] {settings.py:54} INFO - Configured default timezone <Timezone [UTC]>
[2022-03-14 02:07:31,039] {dag_processing.py:756} WARNING - Because we cannot use more than 1 thread (max_threads = 2) w
hen using sqlite. So we set parallelism to 1.
```

Como se observa ya se ejecuta con éxito el comando `airflow scheduler`

Miramos en nuestro browser localhost:8080 que ya no salte el error de scheduler



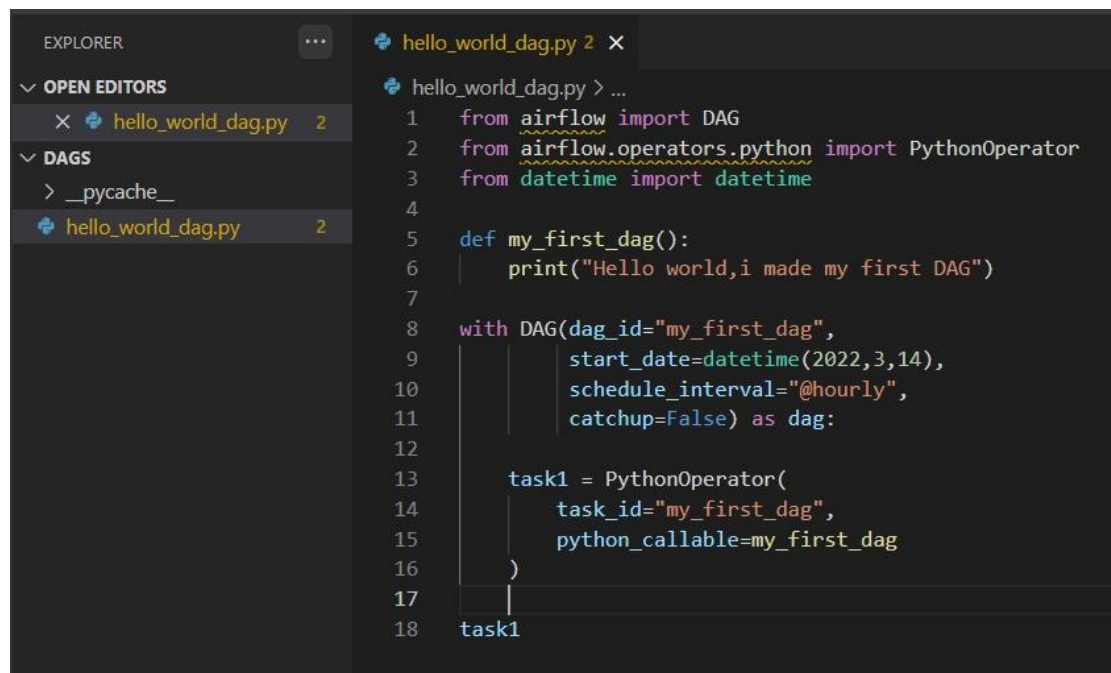
Ahora creamos dentro de la carpeta airflow una carpeta para los dags que vamos a crear y la abrimos con el intérprete de visual studio



Creamos nuestra dag, importando las librerías necesarias para que funcione el airflow además de una librería de datetime para para agendar las tareas en un tiempo definido y ejecutamos nuestra función para la dag, la cual imprimirá el mensaje de hola mundo y se visualizará en nuestro browser.

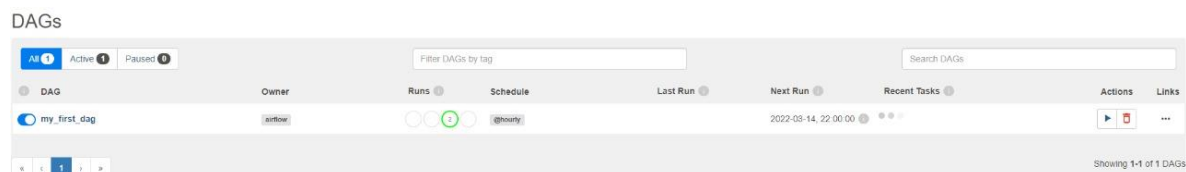
En nuestro constructor de la DAG pasamos los argumentos id, la fecha de inicio y el intervalo de ejecución, además del parámetro de nuestra task por medio de PythonOperator.

Finalmente establecemos el flujo de ejecución para nuestra task1



```
1 from airflow import DAG
2 from airflow.operators.python import PythonOperator
3 from datetime import datetime
4
5 def my_first_dag():
6     print("Hello world, i made my first DAG")
7
8 with DAG(dag_id="my_first_dag",
9         start_date=datetime(2022,3,14),
10        schedule_interval="@hourly",
11        catchup=False) as dag:
12
13     task1 = PythonOperator(
14         task_id="my_first_dag",
15         python_callable=my_first_dag
16     )
17
18 task1
```

Una vez definida la dag refrescamos nuestro browser en localhost:8080 y visualizamos que nuestra dag “my_first_dag” fue creada con éxito, además ejecutamos dicha dag en el botón de play y como se observa la corrida numero uno ha iniciado, coloreando el borde de verde.



Ahora navegamos por la dag para ver sus grafos y detalles

DAG Details

scheduled 2

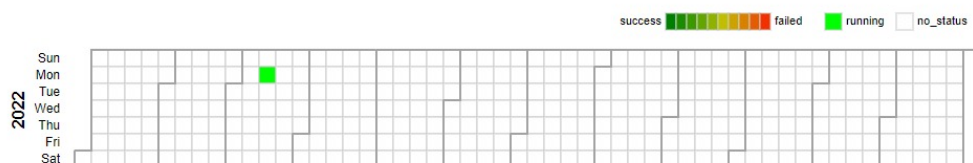
Schedule Interval	@hourly
Catchup	False
Started	True
End Date	None
Max Active Runs	1 / 16
Concurrency	16
Default Args	{}
Tasks Count	1
Task IDs	['my_first_dag']
Relative file location	main.py
Owner	airflow
DAG Run Timeout	None
Tags	None

Visualizamos también el calendario de ejecución que nos marca el momento en el que la dag se puso en marcha, además vemos el grafo de un estado que arroja

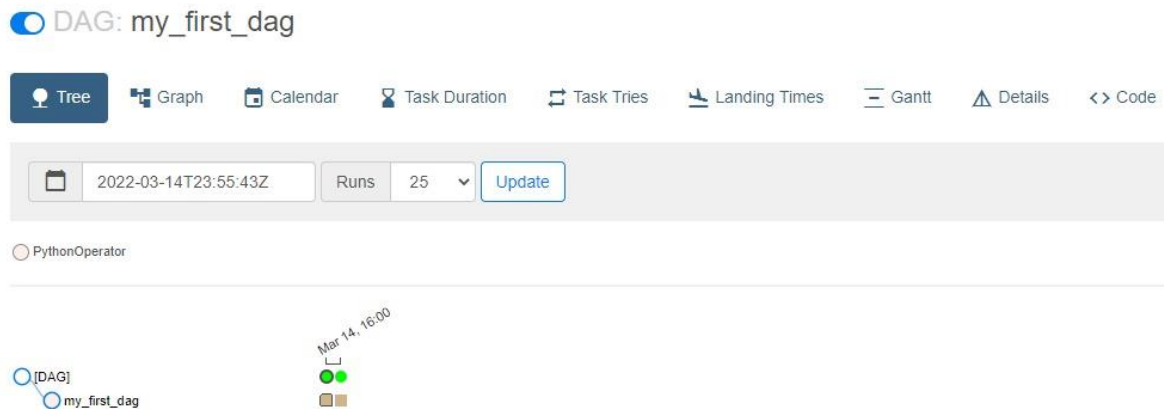


☒ DAG: my_first_dag

[Tree](#)
[Graph](#)
[Calendar](#)
[Task Duration](#)
[Task Tries](#)
[Landing Times](#)
[Gantt](#)
[Details](#)
[Code](#)



Por último, visualizamos las dependencias de nuestras task dentro del dag en el gráfico de árbol, el cual también indica la fecha de ejecución



Conclusión

Al igual que el módulo Prefect visto anteriormente, Airflow nos ayuda a gestionar el flujo de trabajo de las tareas de un programa a ejecutar, esto con objetivo de mitigar los errores que puedan llegar a presentarse, ya que al crear la canalización de dependencia entre tareas, seccionamos bien cada uno de los procesos y evitamos que un error pueda afectar subprocessos dependientes, la idea de apache airflow es crear una interfaz con la que puedas crear, visualizar y gestionar dichos flujos de trabajo de “dags” definidas mediante scripts de Python, en mi caso, la visualización del flujo de trabajo de un script que manda un mensaje de hola mundo; esto con la finalidad de crear trabajos más complejos y que nos den la libertad de personalizar nuestros procesos sin limitaciones como las canalizaciones etl de prefect.

En pocas palabras, este nuevo módulo Airflow permite la administración de las tareas visualizando su dependencia, flujo y estado, y también permitiendo fallas exitosas de cada tarea en dicho flujo, lo cual es muy útil en la tolerancia de fallas, ya que una canalización de datos implementada con Airflow, al igual que Prefect ignorará el estado de la tarea y solamente se dedicará a transmitir los datos, esto con la finalidad de permitir usar el estado para tomar decisiones y permitir que tareas fallidas continúen con un flujo funcional.

Bibliografías

- datastacktv [datastacktv]. (2020, agosto 24). Install Apache Airflow on Windows using Windows Subsystem for Linux (WSL). Youtube. <https://www.youtube.com/watch?v=M521KLHGazc>
- INSAID. (2021, agosto 18). Hello World using Apache-Airflow - INSAID. Medium. <https://insaid.medium.com/hello-world-using-apache-airflow-91859e3bbfd5>
- Sanchez, C., Cabrero, R. J., Guerra, R. S. G., Cortes, D. B., López, R. G., & Roces, V. (2021, abril 7). Apache Airflow: Python como motor de orquestación de flujos de trabajo. Blog de Hiberus Tecnología. <https://www.hiberus.com/crecemos-contigo/apache-airflow-python-como-motor-de-orquestacion-de-flujos-de-trabajo/>

LINK AL REPOSITORIO:

<https://github.com/UlisesVallejo/Airflow.git>