

UNIVERSIDAD DE GUADALAJARA
Centro Universitario de Ciencias Exactas e Ingenierías

Computación Tolerante a Fallas



Nombre: Castillo Mares Gilberto

Código: 213330514

Profesor: Dr. Michel Emanuel López Franco

2024-B

Índice

Contenido

Desarrollo	3
------------------	---

Desarrollo

Utilice Airflow para crear flujos de trabajo como Directed Acyclic Graphs (DAG) de tareas. El planificador de Airflow ejecuta sus tareas en una matriz de trabajadores mientras sigue las dependencias especificadas. Las ricas utilidades de línea de comandos hacen que realizar consultas complejas en DAG sea pan comido. La rica interfaz de usuario facilita la visualización de las canalizaciones que se ejecutan en producción, el seguimiento del progreso y la resolución de problemas cuando sea necesario.

Primero necesitamos instalar la herramienta de Airflow, para esto debemos ejecutar el siguiente comando en la terminal:

```
pip install "apache-airflow[celery]==2.10.2" --constraint "https://raw.githubusercontent.com/apache/airflow/constraints-2.10.2/constraints-3.8.txt"
```

En mi caso necesité de dos librerías más para ejecutar correctamente algunas funciones, en cualquier caso, instalamos estas dos librerías más:

```
pip install kubernetes
```


```
pip install graphviz
```

En esta actividad utilicé un programa llamado Docker para facilitarme las dependencias de Apache Airflow, con solo instalar el programa y unos cuantos pasos más se podrá utilizar Airflow de manera fácil y rápida. Contiene todo lo que Airflow necesita así que me pareció una buena opción usarlo.

Explicare brevemente como instalar y configurar Docker, primero debemos ir a la documentación de Docker: <https://docs.docker.com/desktop/install/windows-install/> en mi caso estoy usando Windows, así que descargué el instalador para Windows.

Install Docker Desktop on Windows

Docker Desktop terms

Commercial use of Docker Desktop in larger enterprises (more than 250 employees OR more than \$10 million USD in annual revenue) requires a [paid subscription](#) .

This page contains the download URL, information about system requirements, and instructions on how to install Docker Desktop for Windows.

Docker Desktop for Windows - x86_64

Docker Desktop for Windows - Arm (Beta)

For checksums, see [Release notes](#)

Contiene una interfaz de usuario el cual te indicará la ubicación de instalación y la opción de utilizar WSL 2. (La funcionalidad de Docker Desktop permanece consistente tanto en WSL como en Hyper-V, sin preferencia por ninguna arquitectura. Hyper-V y WSL tienen sus propias ventajas y desventajas, dependiendo de tu configuración específica y el caso de uso que tengas planeado.)

Activamos la opción de utilizar WSL 2 (recomendado).

Una vez acabada la instalación debemos reiniciar la computadora.

Nota: Docker Desktop utiliza funciones de máquina virtual, así que debemos activar la función de VSM dependiendo de la BIOS de tu tarjeta madre, por defecto esta función está desactivada entonces debemos habilitarla.

Después de reiniciar la computadora, iniciamos el programa y veremos la interfaz de Docker:

Primero debemos configurar las opciones de Airflow y ajustar unas líneas de código:

<https://airflow.apache.org/docs/apache-airflow/2.5.1/docker-compose.yaml>

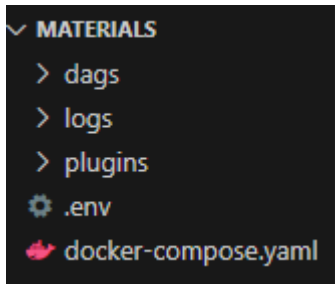
Debemos guardar este archivo con extensión .yaml

Una vez guardado debemos crear una carpeta en la siguiente ruta:

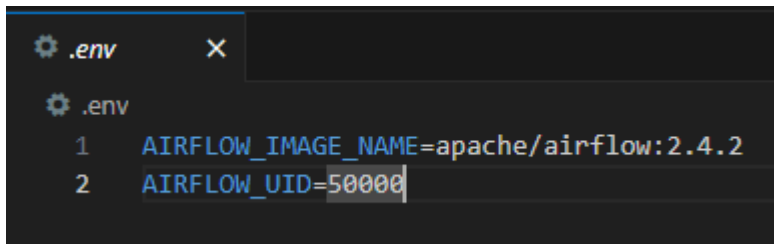
C:\Users\Gilberto

Dentro de la carpeta copiamos o guardamos el archivo .yaml

Utilizaremos Visual Studio Code para crear un nuevo archivo en esta carpeta llamado **.env**



En este archivo **.env** escribimos estas líneas de código:



Guardamos.

Estando en esta ubicación:

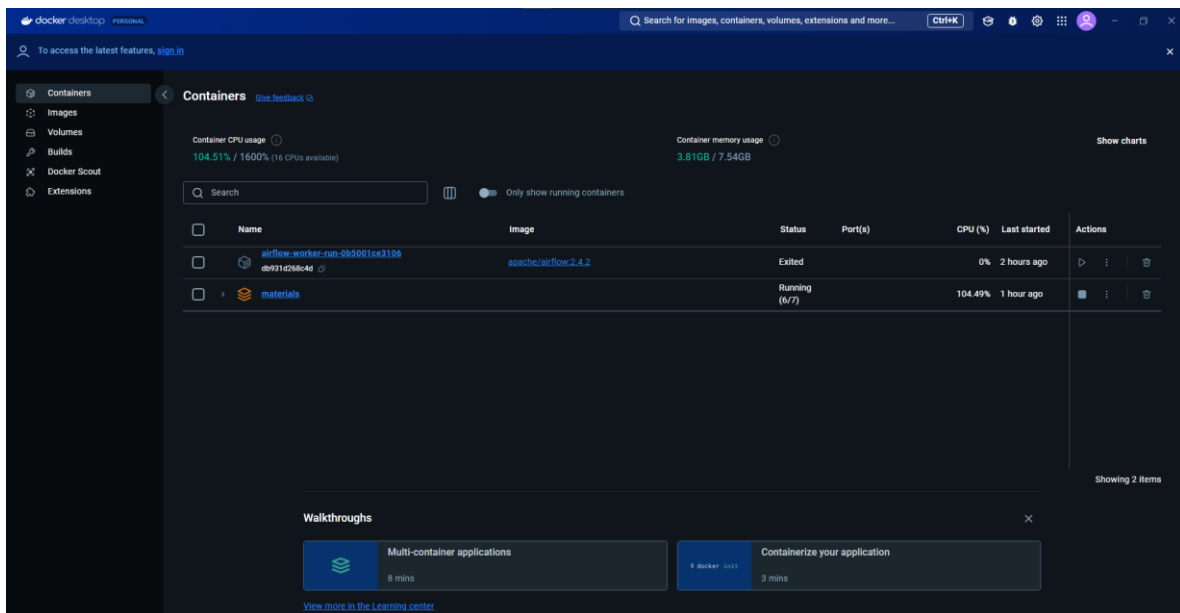
```
PS C:\Users\Gilberto\materials>
```

Ingresamos el siguiente comando:

```
docker-compose up -d
```

Tendremos que esperar algunos minutos en lo que se compilan los archivos.

Una vez terminado y con Docker Desktop abierto



ya podremos utilizar Airflow en nuestro navegador ingresando esta URL:
localhost:8080

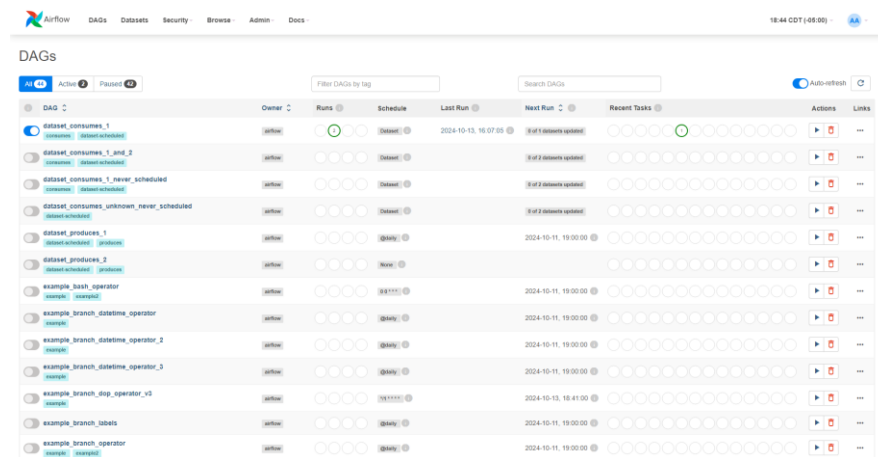
Nos pedirá un usuario y contraseña entonces crearemos una cuenta de administrador con todos los permisos, para esto debemos ingresar el siguiente comando en la terminal:

```
docker-compose run airflow-worker airflow users create --role Admin --username admin --email admin --firstname admin --lastname admin --password admin
```

```
docker-compose run airflow-worker airflow users create --role Admin  
--username admin --email admin --firstname admin --lastname admin -  
-password admin
```

Este comando crea una cuenta de usuario ***admin*** con contraseña ***admin***.

Veremos la interfaz de Airflow:



The screenshot shows the Apache Airflow web interface. At the top, there's a navigation bar with links for DAGs, Datasets, Security, Browse, Admin, and Docs. The main header displays 'DAGs' and a status bar with 'Active' and 'Paused' buttons, a search bar, and an 'Auto-refresh' toggle. Below this is a table listing various DAGs. The table columns include DAG name, Owner, Runs, Schedule, Last Run, Next Run, Recent Tasks, and Actions. The DAGs listed include 'dataset_consumes_1', 'dataset_consumes_1_and_2', 'dataset_consumes_1_never_scheduled', 'dataset_consumes_unknown_never_scheduled', 'dataset_produces_1', 'dataset_produces_2', 'example_task_operator', 'example_branch_datetime_operator', 'example_branch_datetime_operator_2', 'example_branch_datetime_operator_3', 'example_branch_dag_operator_v3', 'example_branch_labels', and 'example_branch_operator'.

DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks	Actions	Links
dataset_consumes_1	airflow	1	DAG	2024-10-11, 16:07:55	2 of 3 datasets updated	1 of 3 datasets updated	[Stop] [Refresh] [More]	
dataset_consumes_1_and_2	airflow	1	DAG		2 of 3 datasets updated	1 of 3 datasets updated	[Stop] [Refresh] [More]	
dataset_consumes_1_never_scheduled	airflow	1	DAG		2 of 3 datasets updated	1 of 3 datasets updated	[Stop] [Refresh] [More]	
dataset_consumes_unknown_never_scheduled	airflow	1	DAG		2 of 3 datasets updated	1 of 3 datasets updated	[Stop] [Refresh] [More]	
dataset_produces_1	airflow	1	DAG	2024-10-11, 19:00:00		1 of 3 datasets updated	[Stop] [Refresh] [More]	
dataset_produces_2	airflow	1	DAG	None		1 of 3 datasets updated	[Stop] [Refresh] [More]	
example_task_operator	airflow	1	DAG	2024-10-11, 19:00:00		1 of 3 datasets updated	[Stop] [Refresh] [More]	
example_branch_datetime_operator	airflow	1	DAG	2024-10-11, 19:00:00		1 of 3 datasets updated	[Stop] [Refresh] [More]	
example_branch_datetime_operator_2	airflow	1	DAG	2024-10-11, 19:00:00		1 of 3 datasets updated	[Stop] [Refresh] [More]	
example_branch_datetime_operator_3	airflow	1	DAG	2024-10-11, 19:00:00		1 of 3 datasets updated	[Stop] [Refresh] [More]	
example_branch_dag_operator_v3	airflow	1	DAG	2024-10-11, 19:00:00		1 of 3 datasets updated	[Stop] [Refresh] [More]	
example_branch_labels	airflow	1	DAG	2024-10-11, 19:00:00		1 of 3 datasets updated	[Stop] [Refresh] [More]	
example_branch_operator	airflow	1	DAG	2024-10-11, 19:00:00		1 of 3 datasets updated	[Stop] [Refresh] [More]	

Ahora si podremos utilizar Airflow para monitorear tareas, flujos de trabajo y controlarlos.

En esta actividad hice un ejemplo en Python de un flujo de trabajo, este ejecutará tres tareas simples, donde una tarea imprime un mensaje, otra realiza una operación matemática, y la tercera tarea depende de la segunda para imprimirse.

Aquí esta el código del archivo de Python que será un DAG:

```
from airflow import DAG
from airflow.operators.python_operator import PythonOperator
from datetime import datetime, timedelta

# Función que imprime un mensaje
def print_hello():
    print("Hello from Airflow!")

# Función que realiza una suma simple
def add_numbers():
    result = 5 + 7
    print(f"El resultado de la suma es: {result}")
    return result

# Función que imprime un mensaje dependiente
def print_result():
    print("La operación anterior fue exitosa.")

# Definir los argumentos por defecto del DAG
default_args = {
```

```

    'owner': 'airflow',
    'retries': 2,
    'retry_delay': timedelta(seconds=30),
    'start_date': datetime(2024, 10, 10), # Fecha de inicio del DAG
}

# Crear el DAG
with DAG(
    'mi_dag_ejemplo1', # Nombre del DAG
    default_args=default_args,
    description='Un DAG simple en Airflow',
    schedule_interval=timedelta(minutes=1), # Se ejecuta cada minuto
    catchup=False, # No ejecuta DAGs pasados si la fecha ya pasó
) as dag:

    # Definir las tareas
    tarea_1 = PythonOperator(
        task_id='imprimir_hello', # Identificador de la tarea
        python_callable=print_hello # Función que ejecutará
    )

    tarea_2 = PythonOperator(
        task_id='sumar_numeros',
        python_callable=add_numbers
    )

    tarea_3 = PythonOperator(
        task_id='imprimir_resultado',
        python_callable=print_result
    )

    # Definir las dependencias entre las tareas
    tarea_1 >> tarea_2 >> tarea_3 # Se ejecutan en secuencia

```

Una vez creado el archivo, debemos guardarlo dentro de la carpeta **dags** la cual creamos antes en esta ruta: C:\Users\Gilberto\nombre_de_tu_carpeta\dags

Después de haber hecho esto, vamos a la interfaz en el navegador, en el código tengo el flujo con el nombre de **mi_dag_ejemplo1**

Version: v2.4.2
Git Version: .release:2.4.2+febf35600d5de172e25280e9f5492257f8988d9f5

Después de actualizar la página lo podemos ver entre los demás DAGs de ejemplo.

Por defecto se crea en pausa, así que debemos activar el botón para que comience a trabajar.

Una vez dentro del DAG podemos ver esto:

DAG: mi_dag_ejemplo1 Un DAG simple en Airflow

Schedule: 0:01:00 | Next Run: 2024-10-12, 18:58:00

Grid | Graph | Calendar | Task Duration | Task Tiles | Landing Times | Gantt | Details | Code | Audit Log

13/10/2024 18:59 | 25 | All Run Types | All Run States | Clear Filters

Auto-refresh

Duration: 00:00:05 | 00:00:05 | 00:00:05

mi_dag_ejemplo1

DAG Details

DAG Runs Summary

Total Runs Displayed	25
Total success	25
First Run Start	2024-10-13, 18:34:00 CDT
Last Run Start	2024-10-13, 18:58:00 CDT
Max Run Duration	00:00:03
Mean Run Duration	00:00:03
Min Run Duration	00:00:02

DAG Summary

Total Tasks	3
PythonOperators	3

Version: v2.4.2
Git Version: .release:2.4.2+febf35600d5de172e25280e9f5492257f8988d9f5

Llevo tiempo ejecutando este DAG así que aquí se puede observar las veces que se ha ejecutado correctamente como esta en el script de Python, el cual indica que, primero imprime el mensaje “Hello from Airflow”, después hace una operación

matemática y si esta tarea se ejecuta correctamente entonces pasa a la siguiente que es imprimir el resultado. Las tres funciones o tareas entonces se observan que se ejecutaron correctamente cada minuto como esta ajustado en el script.