



Universidad de Guadalajara  
Centro Universitario de Ciencias Exactas e Ingenierías  
División de Tecnologías para la Integración Ciber-Humana  
Departamento de Ciencias Computacionales  
Ingeniería en Computación



# Apache Airflow

*Ejemplo de uso de Apache Airflow con Docker Container.*

## Alumno

Hernández Cortez Kevin Uriel.

217734547.

## Materia

Computación Tolerante a Fallas.

D06

## Profesor

Lopez Franco Michel Emanuel.

## Fecha de entrega

Lunes 15 de abril de 2024.

## Introducción

En el ámbito del manejo y automatización de flujos de trabajo, Apache Airflow se destaca como una herramienta esencial para la orquestación de tareas complejas y dependientes. Al facilitar la programación y ejecución de tareas en una secuencia definida, Airflow asegura una gestión eficiente y transparente de los procesos. En esta tarea se hará un ejemplo práctico; la implementación de un DAG sencillo denominado "hello\_airflow", que ejecuta una tarea básica de imprimir un mensaje. Este ejemplo elemental ilustra cómo Airflow permite a los usuarios definir, visualizar y monitorear flujos de trabajo de manera intuitiva y eficaz.

## Desarrollo

### Código

```
from datetime import datetime, timedelta
from airflow import DAG
from airflow.operators.dummy_operator import DummyOperator
from airflow.operators.python_operator import PythonOperator
def print_hello():
    return 'Hello Airflow!'

default_args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'start_date': datetime(2023, 1, 1),
    'email_on_failure': False,
    'email_on_retry': False,
    'retries': 1,
    'retry_delay': timedelta(minutes=5),
}
dag = DAG(
    'hello_airflow',
    default_args=default_args,
    description='A simple hello world DAG',
    schedule_interval=timedelta(days=1),
    catchup=False,
)
start = DummyOperator(
    task_id='start',
    dag=dag,
)
hello = PythonOperator(
    task_id='print_hello',
    python_callable=print_hello,
    dag=dag,
)
start >> hello
```

### Explicación de programa

El código consiste en un ejemplo de un DAG en Apache Airflow. Este mismo consiste en definir un flujo de trabajo (es decir, un DAG) y tareas dentro del flujo. En este ejemplo de tipo “Hola mundo” la tarea es simplemente ejecutar una función de Python que imprime “Hello Airflow!” usando la clase *PythonOperator*. La definición de la tarea y del DAG establece cómo y cuando se ejecutan las tareas.

Más adelante, se revisarán los resultados dentro de la UI de Airflow en el puerto 8080.

### Resultados

```

35 #
36 # _AIRFLOW_WWW_USER_PASSWORD - Password for the administrator account (if requested).
37 #
38 # _PIP_ADDITIONAL_REQUIREMENTS - Additional PIP requirements to add when starting all containers.
39 # Use this option ONLY for quick checks. Installing requirements at container
40 # startup is done EVERY TIME the service is started.
41 # A better way is to build a custom image or extend the official image
42 # as described in https://airflow.apache.org/docs/docker-stack/build.html.
43 #
44 #
45 # Feel free to modify this file to suit your needs.
46 ---
47 x-airflow-common:
48   &airflow-common
49   # In order to add custom dependencies or upgrade provider packages you can use your extended image.
50   # Comment the image line, place your Dockerfile in the directory where you placed the docker-compose.yaml
51   # and uncomment the "build" line below, Then run 'docker-compose build' to build the images.
52   image: ${AIRFLOW_IMAGE_NAME:-apache/airflow:2.9.0}
53   # build:
54   environment:
55     &airflow-common-env
56     AIRFLOW__CORE__EXECUTOR: CeleryExecutor
57     AIRFLOW__DATABASE__SQL_ALCHEMY_CONN: postgresql+psycopg2://airflow:airflow@postgres/airflow
58     AIRFLOW__CELERY__RESULT_BACKEND: db+postgresql://airflow:airflow@postgres/airflow
59     AIRFLOW__CELERY__BROKER_URL: redis://:@redis:6379/0
60     AIRFLOW__CORE__FERNET_KEY: ''
61     AIRFLOW__CORE__DAGS_ARE_PAUSED_AT_CREATION: true
62     AIRFLOW__CORE__LOAD_EXAMPLES: false
63   volumes:
64     - ./dags:/opt/airflow/dags
65     - ./logs:/opt/airflow/logs
66     - ./plugins:/opt/airflow/plugins
67     - ./config:/opt/airflow/config
68   networks:
69     - airflow
70   restart: unless-stopped
71
72 networks:
73   airflow:
74     driver: bridge
75
76 volumes:
77   postgres:
78     driver: local
79   redis:
80     image: redis:6-alpine
81     driver: local
82
83 services:
84   postgres:
85     image: postgres:13-alpine
86     environment:
87       POSTGRES_DB: airflow
88       POSTGRES_USER: airflow
89       POSTGRES_PASSWORD: airflow
90     volumes:
91       - postgres:/var/lib/postgresql/data
92     networks:
93       - airflow
94     restart: unless-stopped
95   redis:
96     image: redis:6-alpine
97     networks:
98       - airflow
99     restart: unless-stopped
100   airflow-init:
101     image: apache/airflow:2.9.0
102     command: sh -c "airflow db init"
103     networks:
104       - airflow
105     restart: unless-stopped
106   airflow:
107     image: apache/airflow:2.9.0
108     command: sh -c "airflow webserver"
109     networks:
110       - airflow
111     restart: unless-stopped
112
113 
```

```

airflow-init-1 [2024-04-16T02:34:06.143+0000] [db.py:1650] INFO - Creating tables
airflow-init-1 INFO [alembic.runtime.migration] Context impl PostgresqlImpl.
airflow-init-1 INFO [alembic.runtime.migration] Will assume transactional DDL.
airflow-init-1 INFO [alembic.runtime.migration] Context impl PostgresqlImpl.

```

Ilustración 1. Implementación del proyecto en Visual Studio Code.

Primeramente, se implementó el código *Hello Airflow!* en Visual Studio Code y se descargó el archivo *docker-composer.yaml* correspondiente a la versión más reciente. Luego, en una nueva terminal, se usó el comando *docker-compose up airflow-init* para inicializar Airflow dentro del contexto de Docker. Ya en la UI de la página, podemos observar lo siguiente:

# Apache Airflow

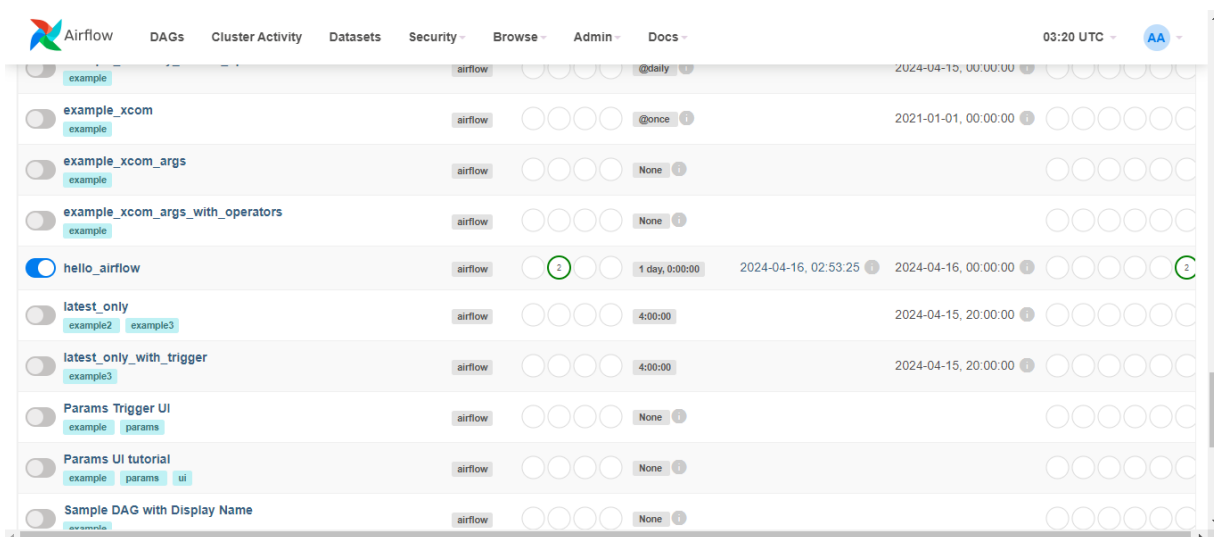
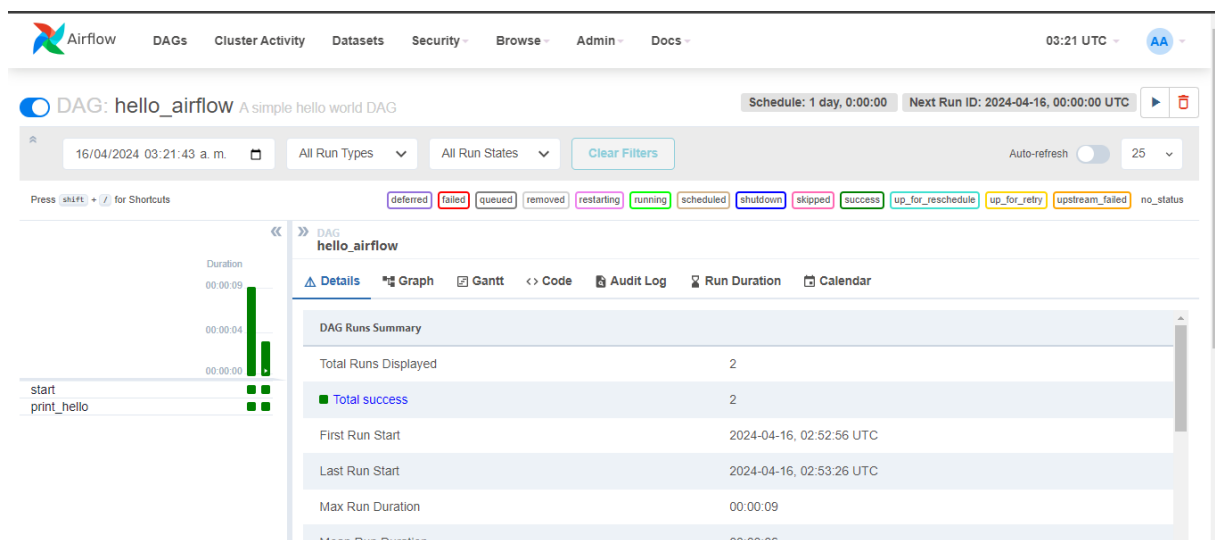


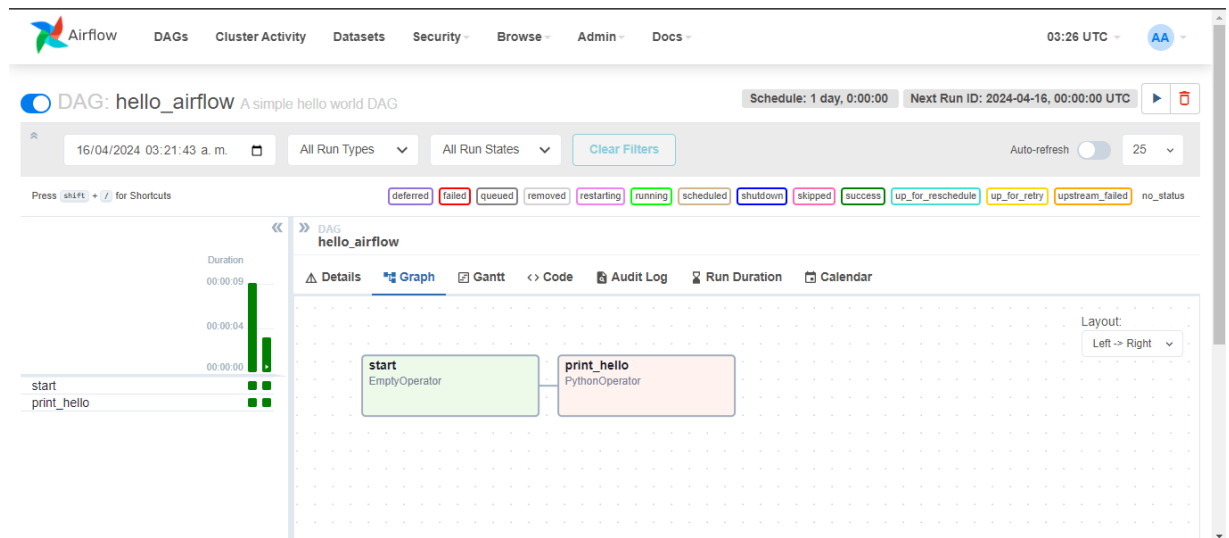
Ilustración 2. UI inicial.

## 1. Details



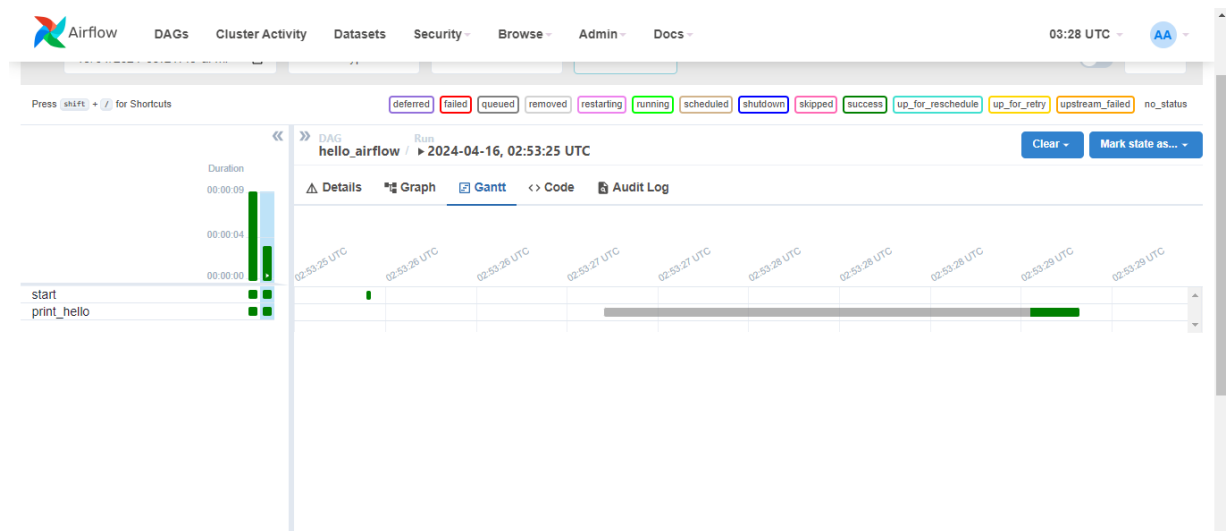
Esta pestaña podemos ver información del DAG seleccionado *hello\_airflow*. Incluye metadatos como el propietario del DAG, cuándo fue el último run, la frecuencia de ejecución, y si está pausado o no. También podemos ver la descripción del DAG.

## 2. Graph



Muestra una representación gráfica del DAG, con su tarea y su dependencia. Cada nodo en el gráfico representa una tarea, y las flechas muestran cómo fluyen los datos de una tarea a otra.

### 3. Gantt



El diagrama de Gantt muestra la línea de tiempo de ejecución de la tarea dentro de un run del DAG. La barra en el diagrama representa una instancia de tarea, mostrando cuándo comenzó y terminó.

### 4. Code

# Apache Airflow

The screenshot shows the Apache Airflow web interface. At the top, there's a navigation bar with links to DAGs, Cluster Activity, Datasets, Security, Browse, Admin, and Docs. The main header displays the DAG name 'hello\_airflow' and its description 'A simple hello world DAG'. Below this, there are filters for 'All Run Types' and 'All Run States', along with a 'Clear Filters' button. A status bar at the bottom of the filter section shows various task states like 'deferred', 'failed', 'queued', 'removed', 'restarting', 'running', 'scheduled', 'shutdown', 'skipped', 'success', 'up\_for\_reschedule', 'up\_for\_retry', 'upstream\_failed', and 'no\_status'. The main content area is divided into two tabs: 'Details' and 'Code'. The 'Code' tab is active, showing the Python source code for the DAG. The code defines a 'print\_hello' task using the 'PythonOperator' and includes a 'default\_args' dictionary. The 'Task' tab is also visible, showing the task name 'print\_hello' and its state 'running'.

Aquí se puede ver directamente el código fuente del DAG.

## 5. Audit Log

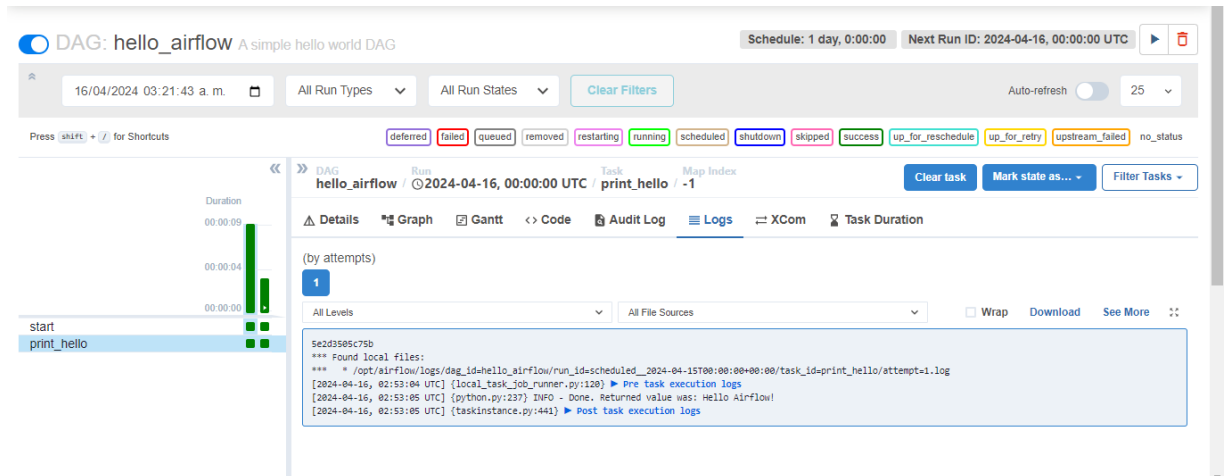
The screenshot shows the Apache Airflow web interface with the 'Audit Log' tab selected. The 'Audit Log' tab displays a table of events related to the DAG. The table has columns for 'WHEN', 'EVENT', 'OWNER', and 'EXTRA'. The events listed are:

WHEN	EVENT	OWNER	EXTRA
2024-04-16, 02:53:05 UTC	success	airflow	
2024-04-16, 02:53:04 UTC	running	airflow	

There are also filters for 'Show Logs After', 'Show Logs Before', 'Filter by Run ID', and 'Filter by Task ID'. The 'Show Logs After' filter is set to '2024-04-16T02:52:56Z' and the 'Show Logs Before' filter is set to '2024-04-16T02:53:05Z'. The 'Filter by Run ID' filter is set to 'scheduled\_\_2024-04-15T00:00:00+00:00' and the 'Filter by Task ID' filter is set to 'print\_hello'.

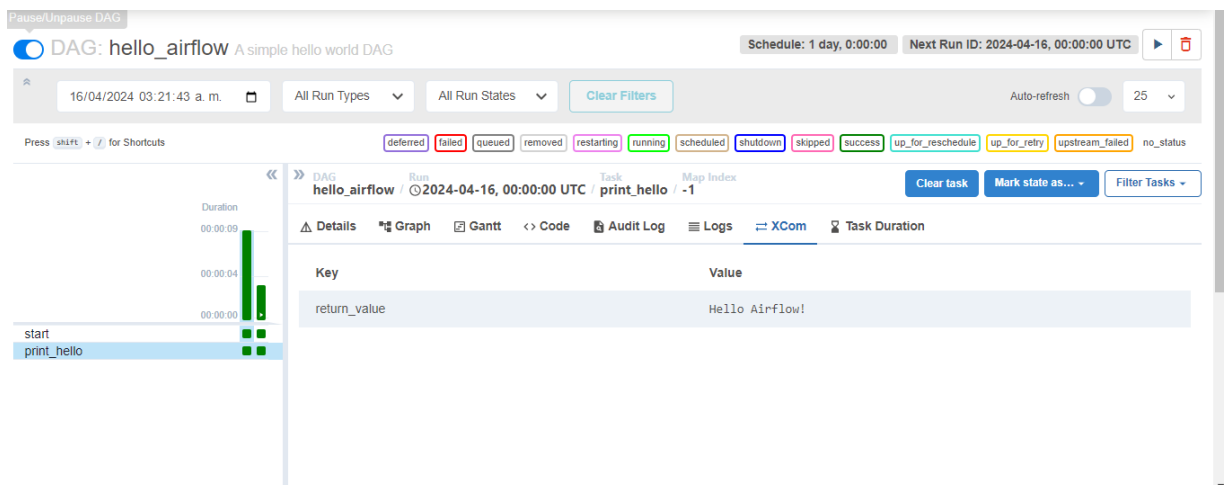
Aquí se puede ver un registro de auditoría de todas las acciones realizadas en el DAG. Esto incluye cambios en la configuración, activaciones/desactivaciones del DAG, y otras modificaciones.

## 6. Logs



Aquí se encontrará los registros de ejecución de la tarea. Estos logs son cruciales para temas de depuración.

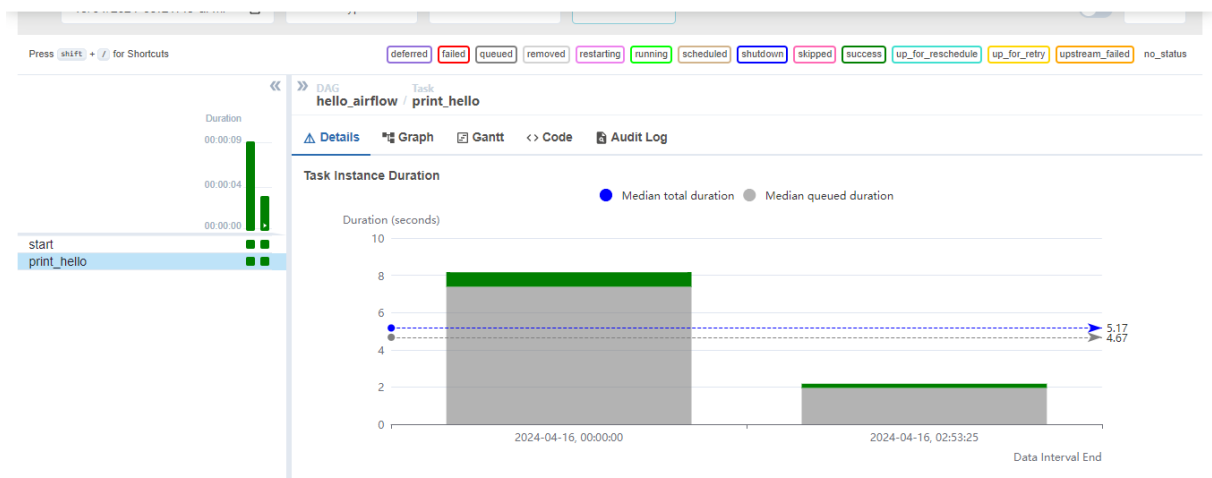
## 7. Xcom



XComs (short for "cross-communication") son mensajes que las tareas pueden empujar y extraer para compartir datos entre ellas (en este caso, la impresión de *Hello Airflow!*)

## 8. Task Duration

## Apache Airflow



Muestra gráficos de la duración de las tareas a lo largo del tiempo.

## Conclusión

La implementación del DAG "hello\_airflow" en Apache Airflow demuestra la flexibilidad y funcionalidad de la plataforma en la gestión de tareas automatizadas. A través de este ejemplo, se pudo observar cómo Airflow facilita la definición clara de tareas, la configuración de sus dependencias, y la visualización de su ejecución mediante interfaces gráficas como las pestañas Graph, Gantt, y Logs. Estas herramientas no solo mejoran la comprensibilidad y transparencia de los procesos, sino que también ofrecen capacidades esenciales para la depuración y optimización de flujos de trabajo.

## Bibliografía.

- *Public Interface of Airflow — Airflow Documentation.* (s. f.).  
<https://airflow.apache.org/docs/apache-airflow/stable/public-airflow-interface.html>
- Fernandez, O. (2023, 23 octubre). *¿Qué es Apache Airflow? Introducción.* Aprender BIG DATA. <https://aprenderbigdata.com/apache-airflow/>
- *Release Notes — Airflow documentation.* (2024, 19 enero).  
[https://airflow.apache.org/docs/apache-airflow/stable/release\\_notes.html#airflow-2-8-1-2024-01-19](https://airflow.apache.org/docs/apache-airflow/stable/release_notes.html#airflow-2-8-1-2024-01-19)



- *Running Airflow in Docker — Airflow Documentation.* (s. f.).  
<https://airflow.apache.org/docs/apache-airflow/stable/howto/docker-compose/index.html>
- Software Guru. (2021, 4 febrero). *#SGVirtual 20.12 - Automatizando ideas con Apache Airflow* [Vídeo]. YouTube. <https://www.youtube.com/watch?v=ewK4KszmeTI>
- Ashutosh Tripathi. (2024, 29 enero). *How to Install Apache Airflow on Windows using Docker Container* / *#airflow #mlops #ashutosh\_ai* [Vídeo]. YouTube.  
<https://www.youtube.com/watch?v=4gz9SogFh1Q>