Universidad de Guadalajara

Centro Universitario de Ciencias Exactas e Ingenierías

División de Tecnologías para la Integración Ciber-Humana

Departamento de Ciencias Computacionales

Ingenieria 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

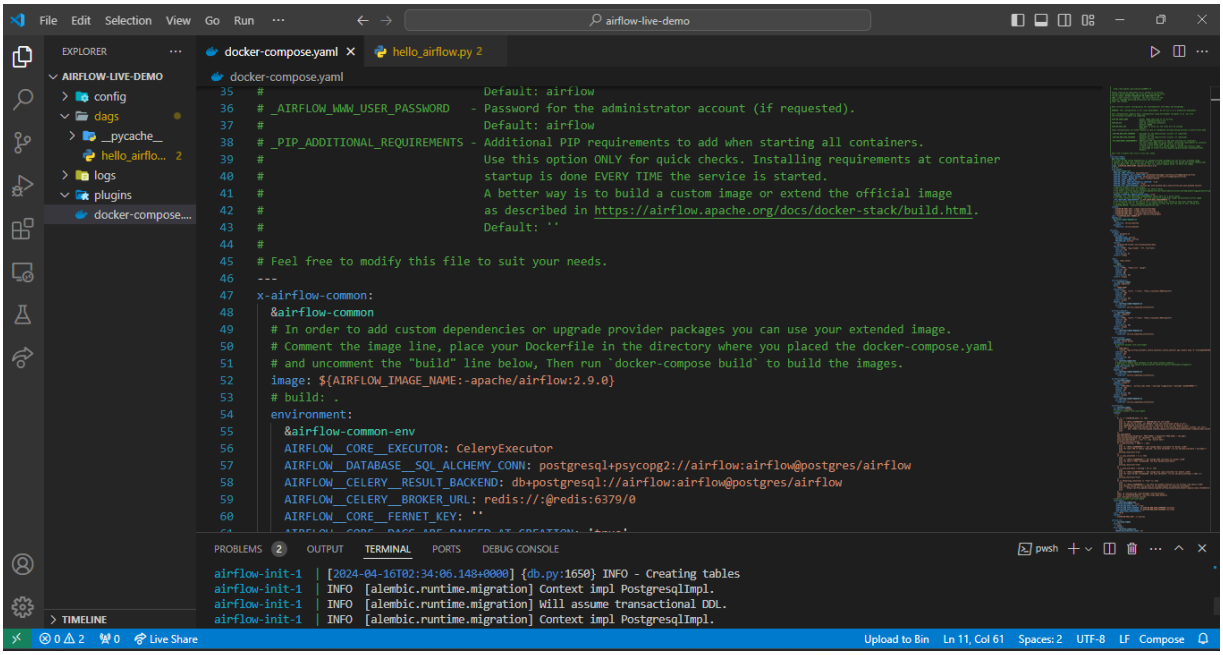


Ilustración 1. Implementación del proyecto en Visual Studio Code.

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:

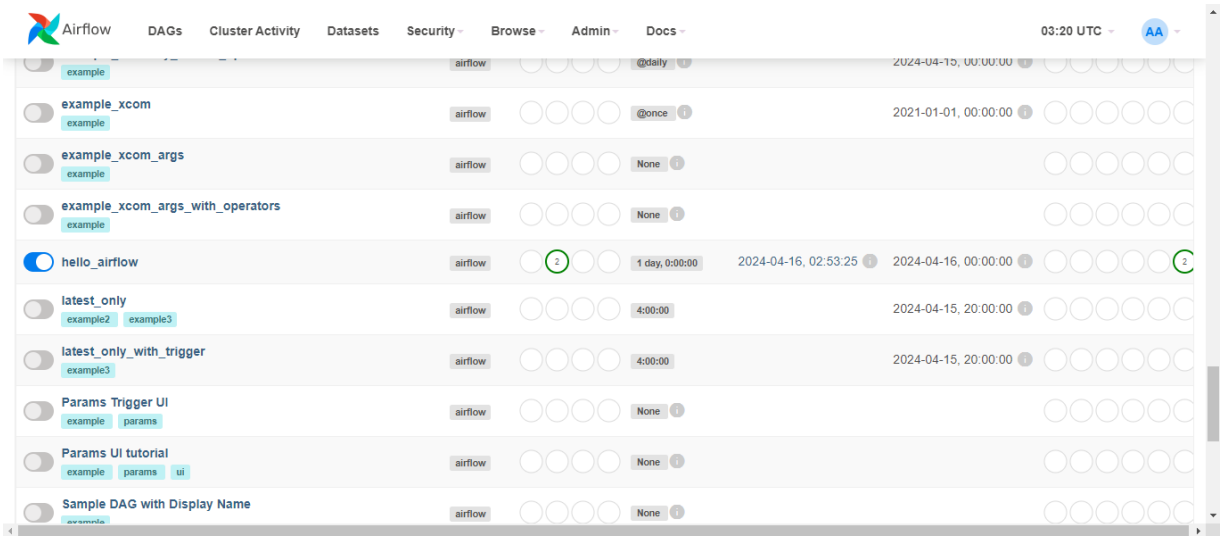
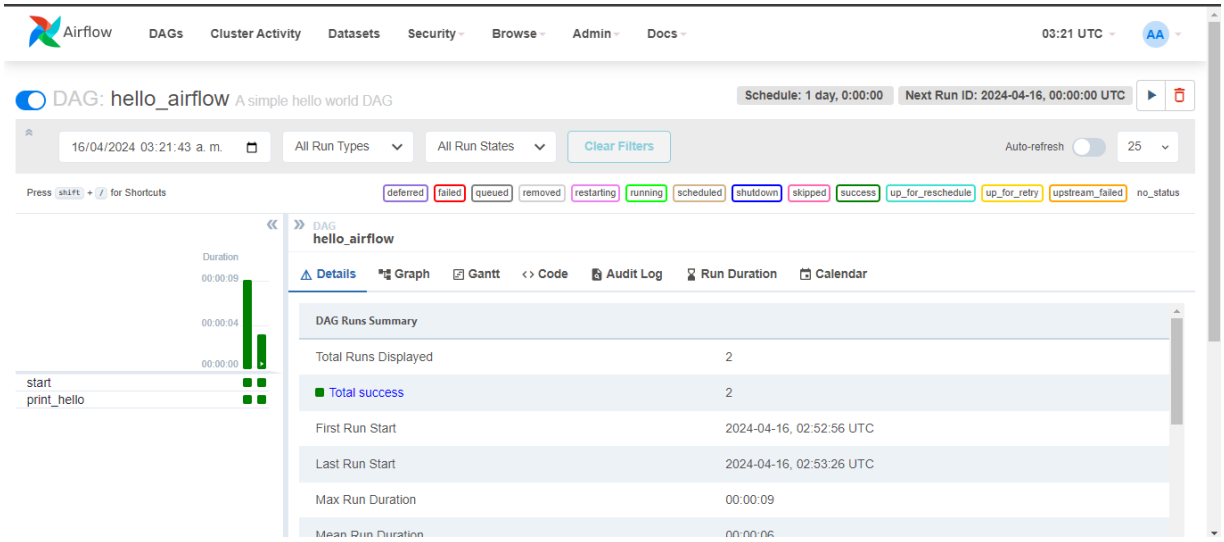


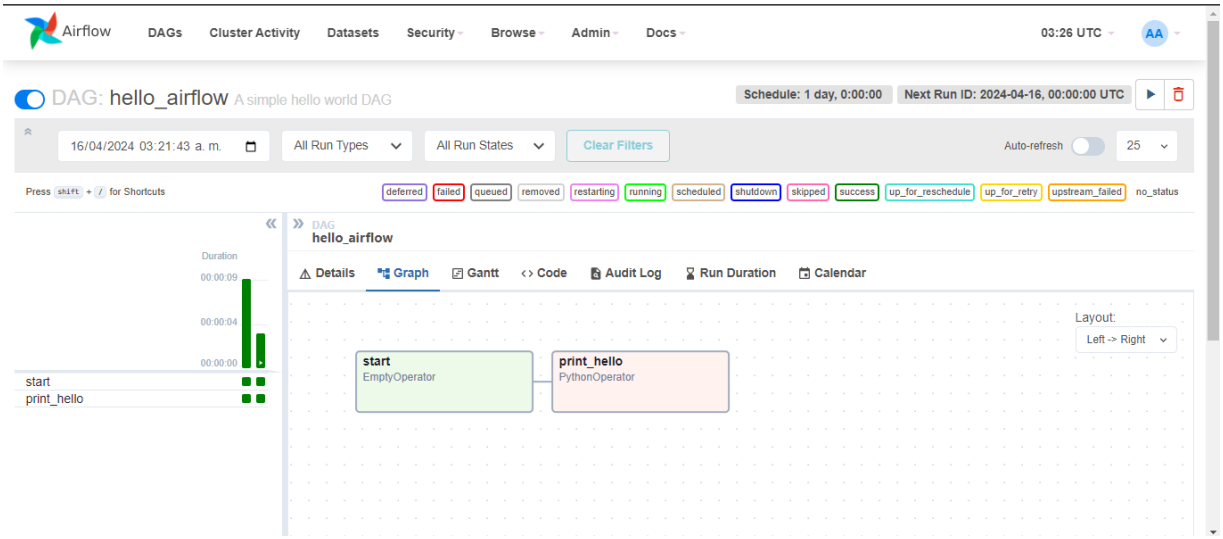
Ilustración 2. UI inicial.

1. Details



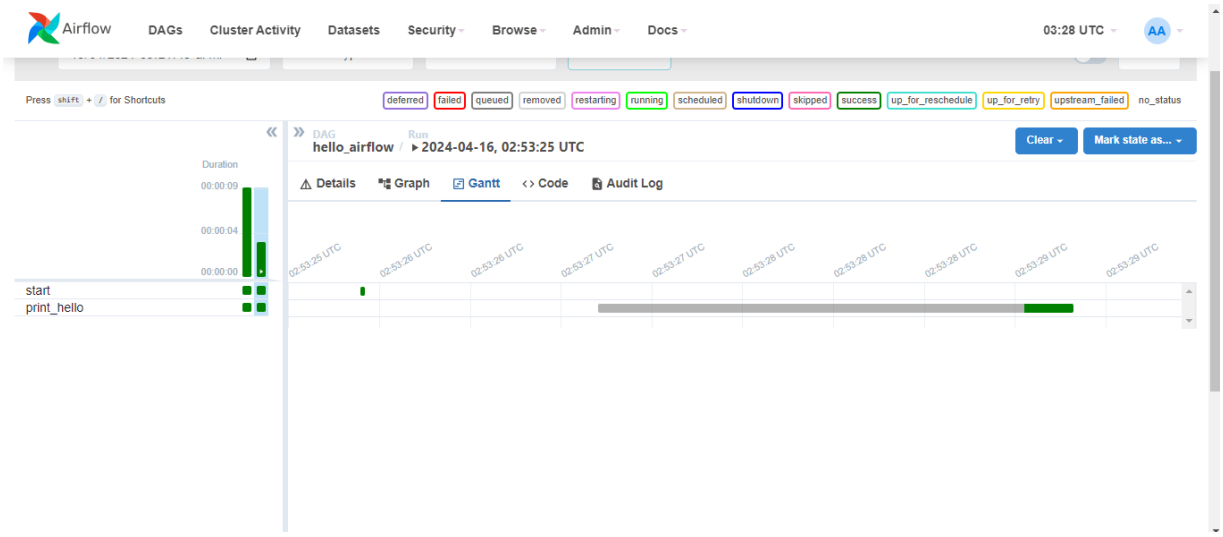
Esta pestaña podemos ver información información del DAG seleccionado *hello_airflow*. Incluye metadatos como el propietario del DAG, cuándo fue el último run, la frecuencia de ejecución, y si está pausado o no. También podemos ver la descripción del DAG.

2. Graph



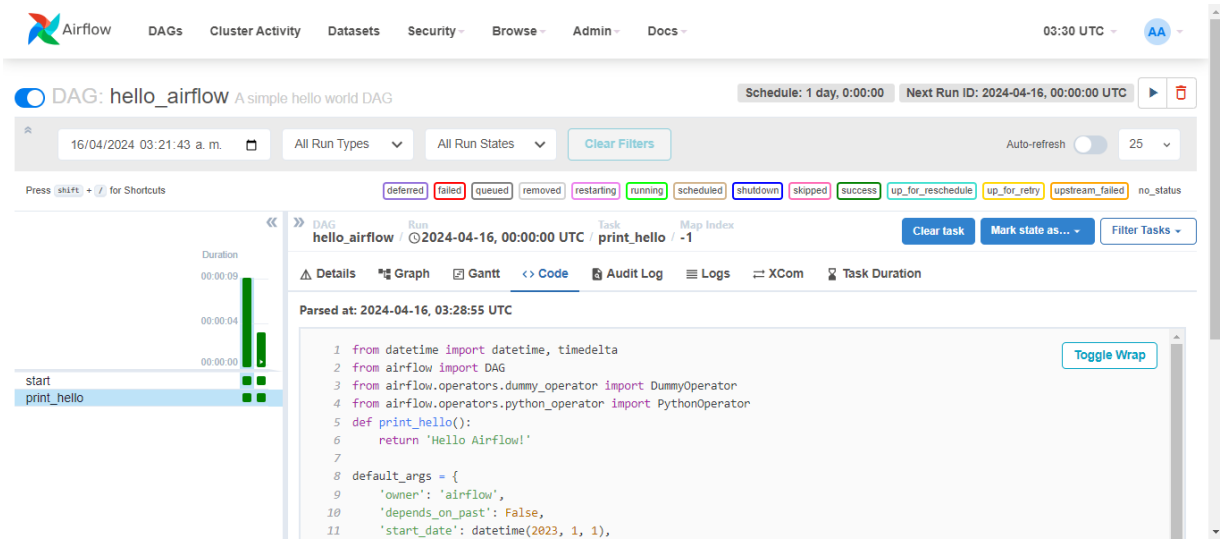
Muestra una representación gráfica del DAG, con su tarea y su dependencia. Cada nodo en el gráfico representa una tarea, y las flechas muestran cómo fluyen los datos de una tarea a otra.

3. Gantt



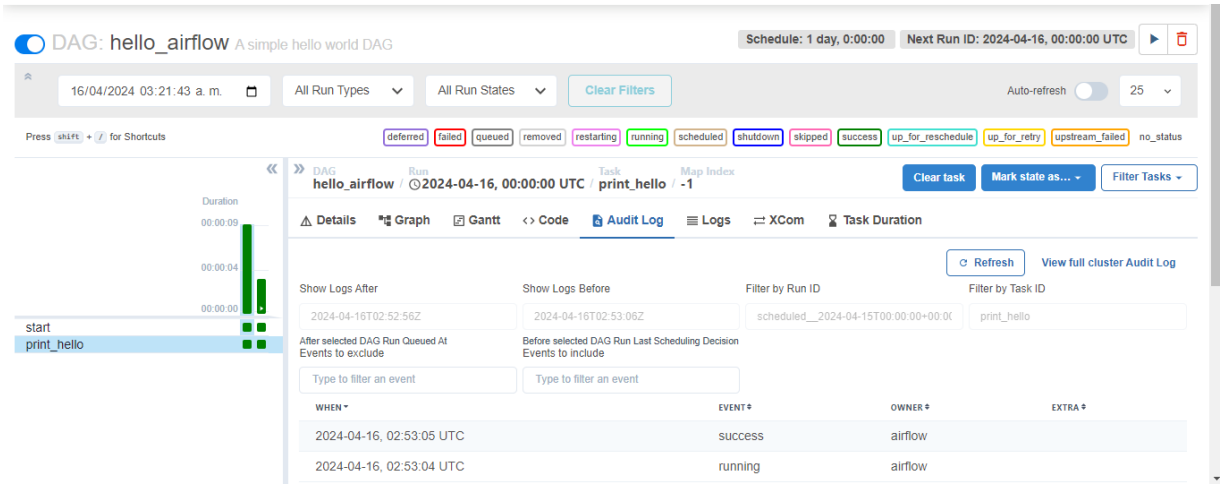
El diagrama de Gantt muestra la línea de tiempo de ejecución de la tarea dentro de un run del DAG. La barra en el diagrama representa una instancia de tarea, mostrando cuándo comenzó y terminó.

4. Code



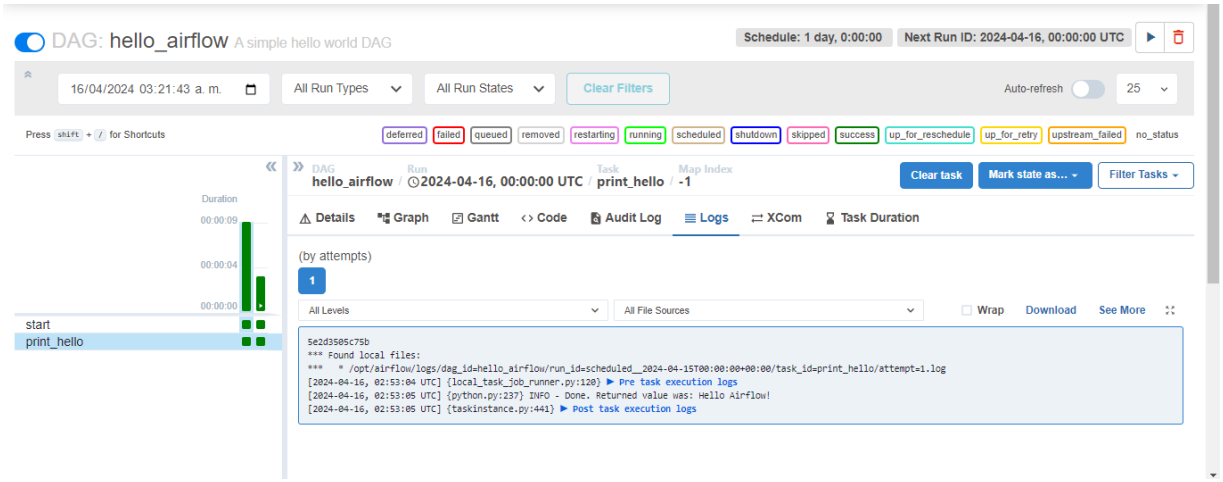
Aquí se puede ver directamente el código fuente del DAG.

5. Audit Log



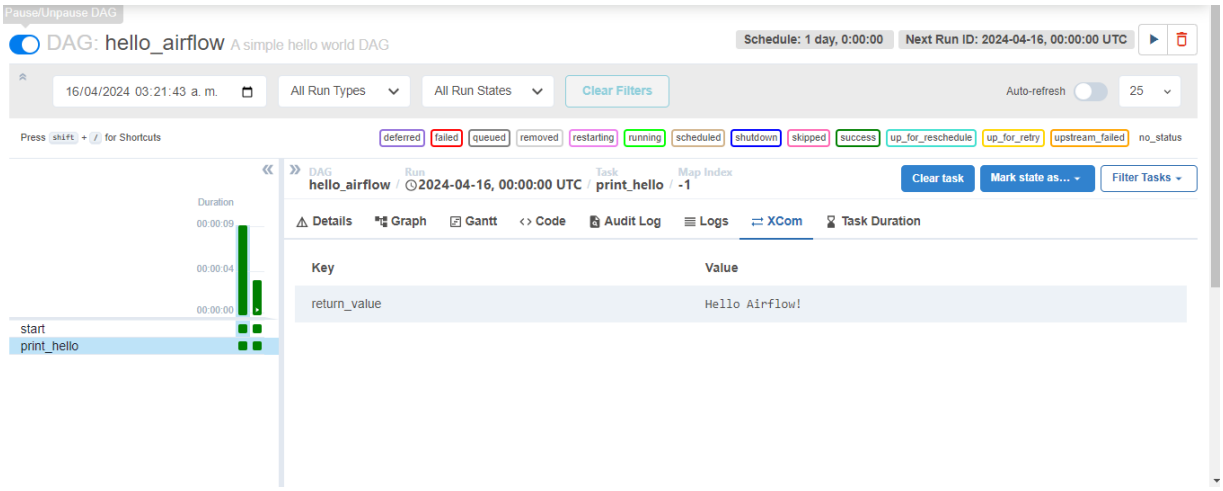
Aquí se puede ver un registro de auditoría de todas las acciones realizadas en el DAG. Esto incluye cambios en la configuración, activaciones/desactivaciones del DAG, y otras modificaciones.

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

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

- *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