Arellano Granados Angel Mariano

218123444

Computación Tolerante a Fallas

D06 2023B

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

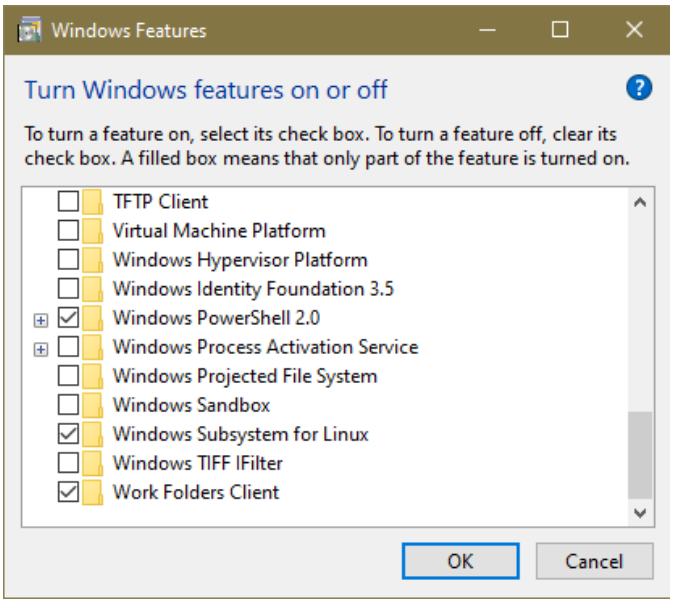
Introducción

Apache Airflow es una plataforma de gestión de flujo de trabajo de código abierto escrita en Python, donde los flujos de trabajo se crean a través de scripts de Python. Fue creada por Airbnb en octubre de 2014 como solución para la gestión de flujos de trabajo dentro de la empresa.

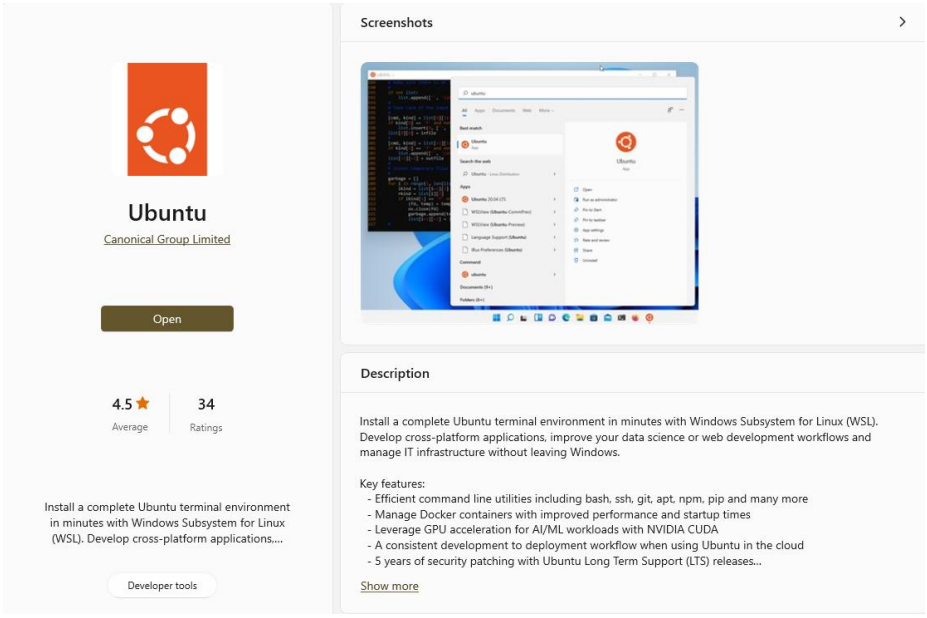
Desarrollo

Instalación:

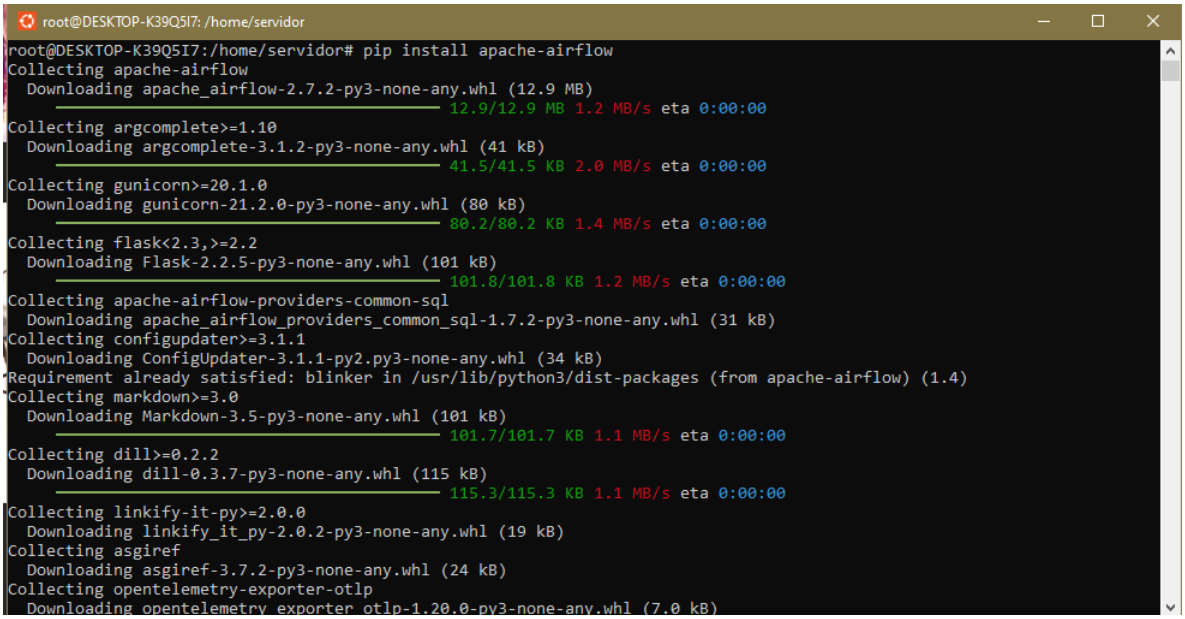
Para instalar la herramienta en mi sistema operativo Windows 10 primero se habilito los subsistemas de Linux



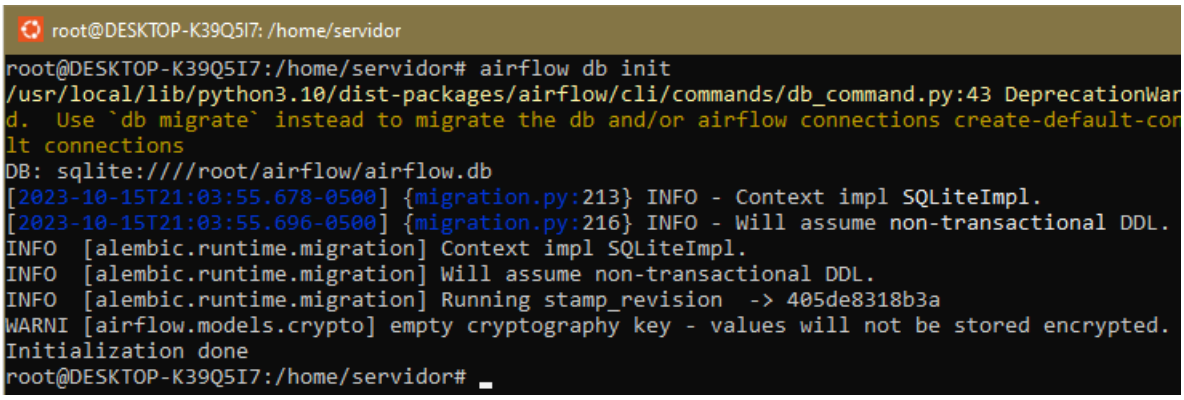
Después se descargo una terminal de Ubuntu dentro de la Microsoft store



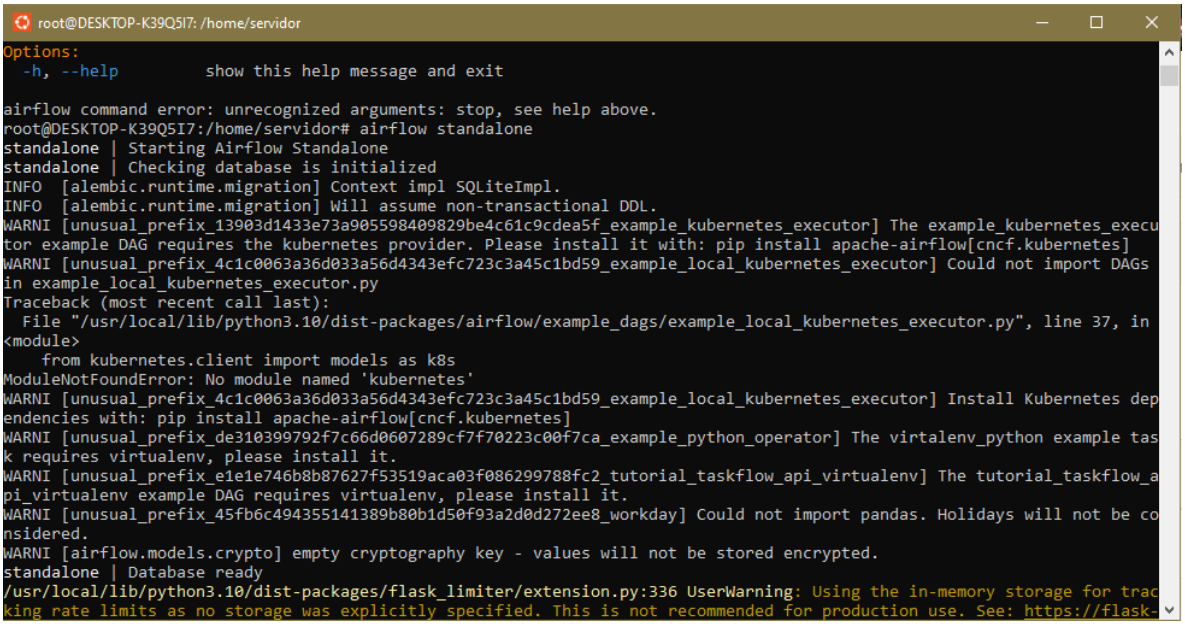
Tras actualizar *apt-get*, pip y Python podemos instalar airflow por el comando *pip install apache-airflow*



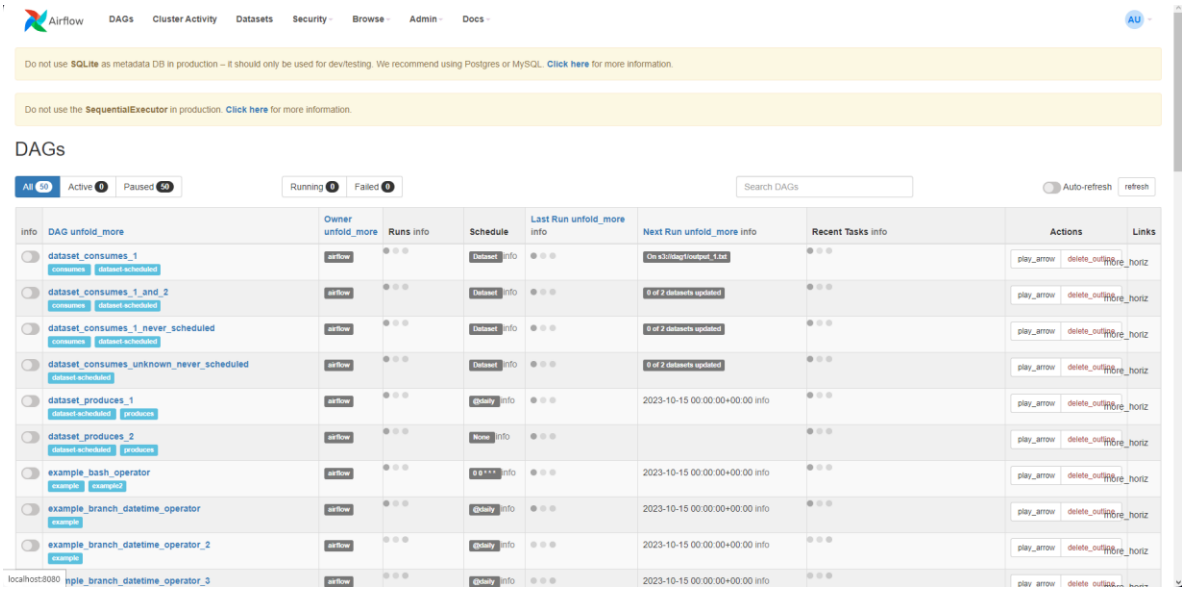
Iniciamos la base de datos de airflow



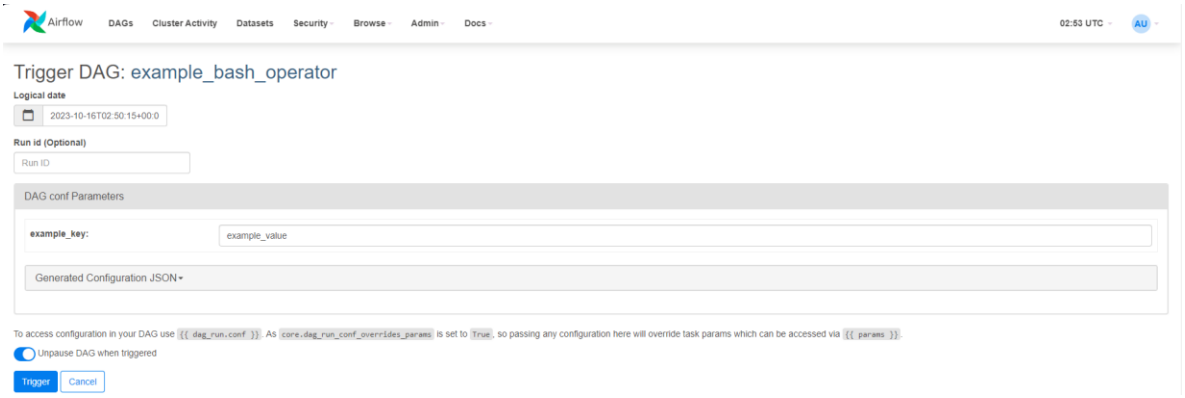
Después iniciamos un servidor web para poder visualizar la interfaz en un navegador con el comando *airflow standalone*

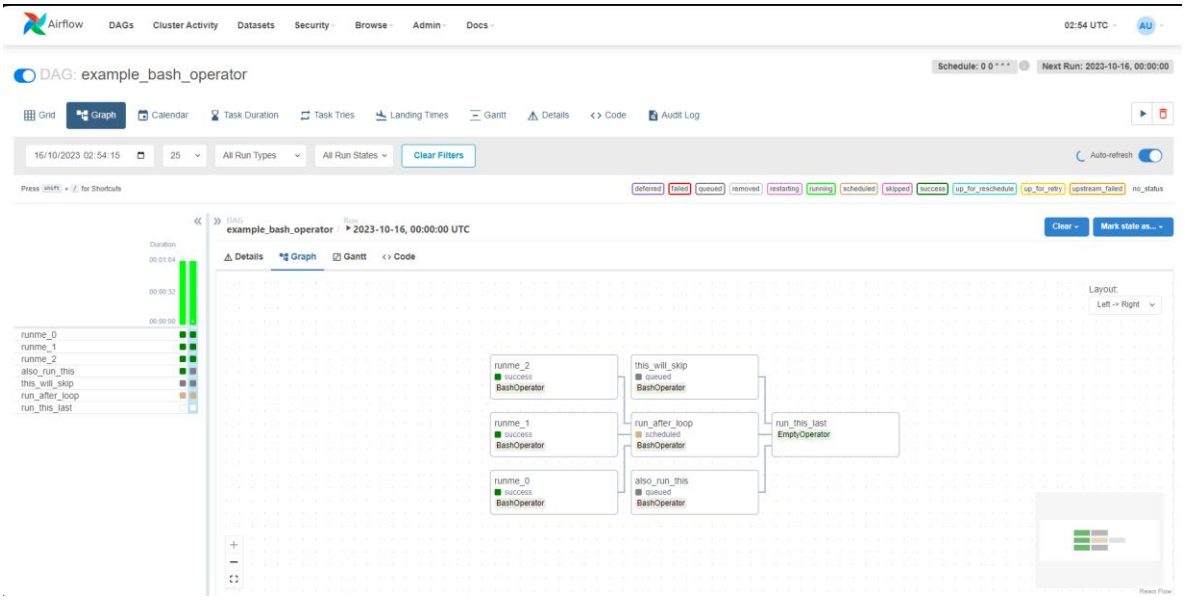


Así después de iniciar sesión con el nombre de usuario y contraseña que nos proporcionaron en consola podemos acceder al menú principal de airflow desde cualquier navegador en la url *localhost:8080/home*



Prueba de Trigger DAG: example_bash_operator





Podemos observar como funciona un DAG y como podemos iniciarlo para posteriormente revisar los resultados de sus ejecuciones, así como el grafo de acciones.

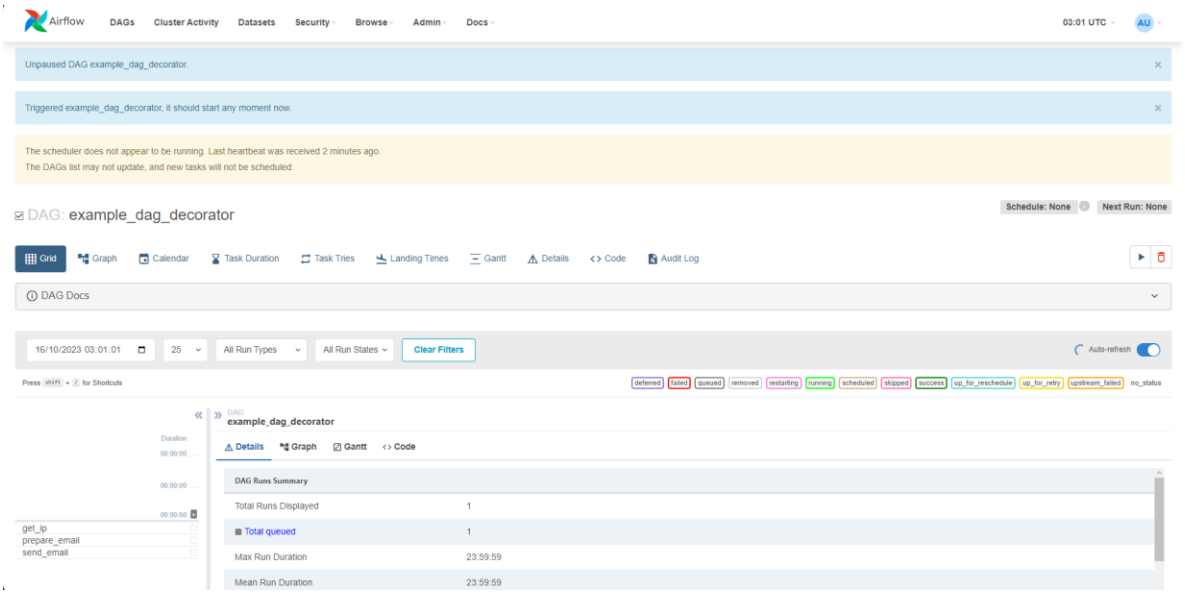
Código:

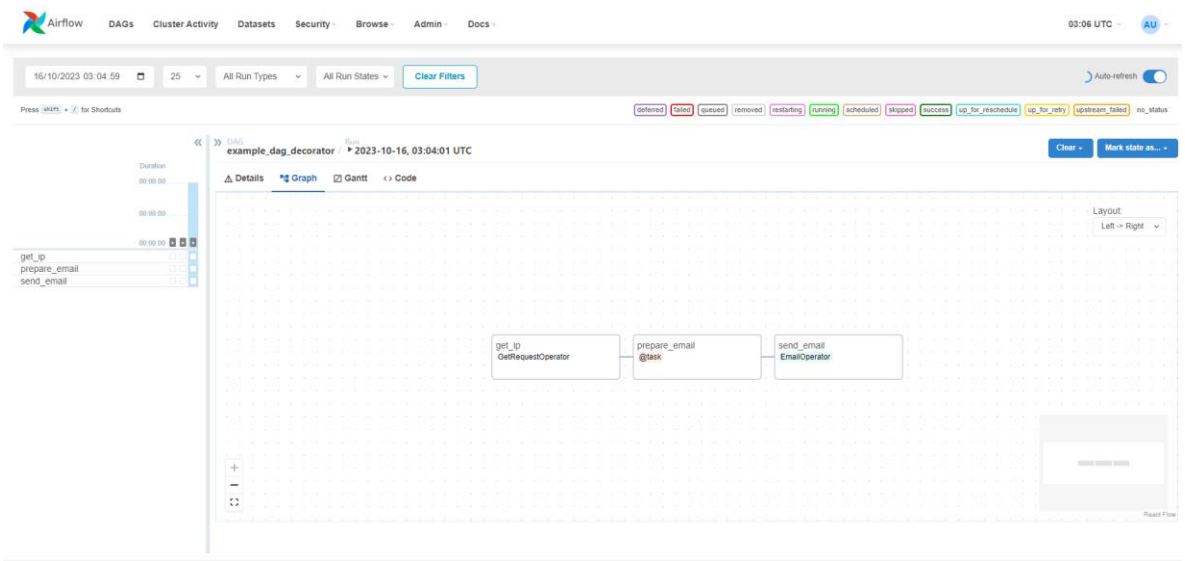
```
#
# Licensed to the Apache Software Foundation (ASF) under one
# or more contributor license agreements.  See the NOTICE file
# distributed with this work for additional information
# regarding copyright ownership.  The ASF licenses this file
# to you under the Apache License, Version 2.0 (the
# "License"); you may not use this file except in compliance
# with the License.  You may obtain a copy of the License at
#
#   http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing,
# software distributed under the License is distributed on an
# "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
# KIND, either express or implied.  See the License for the
# specific language governing permissions and limitations
# under the License.
"""Example DAG demonstrating the usage of the BashOperator."""
from __future__ import annotations

import datetime

import pendulum

from airflow import DAG
from airflow.operators.bash import BashOperator
from airflow.operators.empty import EmptyOperator

with DAG(
    dag_id="example_bash_operator",
    schedule="0 0 * * *",
    start_date=pendulum.datetime(2021, 1, 1, tz="UTC"),
    catchup=False,
    dagrun_timeout=datetime.timedelta(minutes=60),
    tags=["example", "example2"],
    params={"example_key": "example_value"},
) as dag:
    run_this_last = EmptyOperator(
        task_id="run_this_last",
```

```
    )

    # [START howto_operator_bash]
    run_this = BashOperator(
        task_id="run_after_loop",
        bash_command="echo 1",
    )
    # [END howto_operator_bash]

    run_this >> run_this_last

    for i in range(3):
        task = BashOperator(
            task_id=f"runme_{i}",
            bash_command='echo "{{ task_instance_key_str }}" && sleep
1',
        )
        task >> run_this

    # [START howto_operator_bash_template]
    also_run_this = BashOperator(
        task_id="also_run_this",
        bash_command='echo "ti_key={{ task_instance_key_str }}"',
    )
    # [END howto_operator_bash_template]
    also_run_this >> run_this_last

# [START howto_operator_bash_skip]
this_will_skip = BashOperator(
    task_id="this_will_skip",
    bash_command='echo "hello world"; exit 99;',
    dag=dag,
)
# [END howto_operator_bash_skip]
this_will_skip >> run_this_last

if __name__ == "__main__":
    dag.test()

```

Prueba de example_dag_decorator

Este ejemplo automatiza en proceso de mandar correos a un destinatario en una fecha predeterminada.





Código:

```
#
2# Licensed to the Apache Software Foundation (ASF) under one
3# or more contributor license agreements.  See the NOTICE file
4# distributed with this work for additional information
5# regarding copyright ownership.  The ASF licenses this file
```

```
# to you under the Apache License, Version 2.0 (the
# "License"); you may not use this file except in compliance
# with the License.  You may obtain a copy of the License at
#
#   http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing,
# software distributed under the License is distributed on an
# "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
# KIND, either express or implied.  See the License for the
# specific language governing permissions and limitations
# under the License.
from __future__ import annotations

from typing import Any

import httpx
import pendulum

from airflow.decorators import dag, task
from airflow.models.baseoperator import BaseOperator
from airflow.operators.email import EmailOperator
from airflow.utils.context import Context


class GetRequestOperator(BaseOperator):
    """Custom operator to send GET request to provided url"""

    def __init__(self, *, url: str, **kwargs):
        super().__init__(**kwargs)
        self.url = url

    def execute(self, context: Context):
        return httpx.get(self.url).json()


# [START dag_decorator_usage]
@dag(
    schedule=None,
    start_date=pendulum.datetime(2021, 1, 1, tz="UTC"),
    catchup=False,
    tags=["example"],
)
def example_dag_decorator(email: str = "example@example.com"):
    """
    DAG to send server IP to email.

```

```
    :param email: Email to send IP to. Defaults to example@example.com.
    """
    get_ip = GetRequestOperator(task_id="get_ip",
url="http://httpbin.org/get")

    @task(multiple_outputs=True)
    def prepare_email(raw_json: dict[str, Any]) -> dict[str, str]:
        external_ip = raw_json["origin"]
        return {
            "subject": f"Server connected from {external_ip}",
            "body": f"Seems like today your server executing Airflow is
connected from IP {external_ip}<br>",
        }

    email_info = prepare_email(get_ip.output)

    EmailOperator(
        task_id="send_email", to=email, subject=email_info["subject"],
html_content=email_info["body"]
    )


example_dag = example_dag_decorator()
# [END dag_decorator_usage]

```

Conclusión

Airflow es una alternativa de la anterior herramienta de control de flujo que vimos Prefect, la cual aún ser mucho más compleja de instalar ofrece mayor variedad y libertad a la hora de implementarla en una mayor variedad de proyectos con diferentes necesidades y objetivos.