



Arellano Granados Angel Mariano

218123444

Computación Tolerante a Fallas

D06 2023B

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

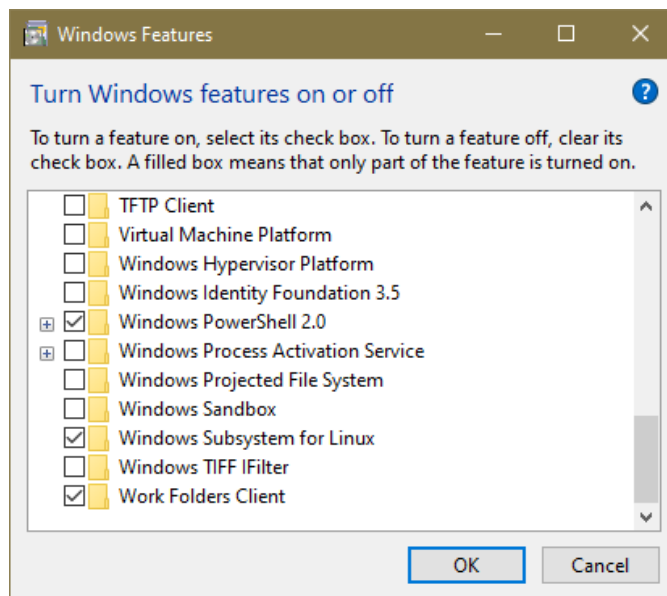
Introducción

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.

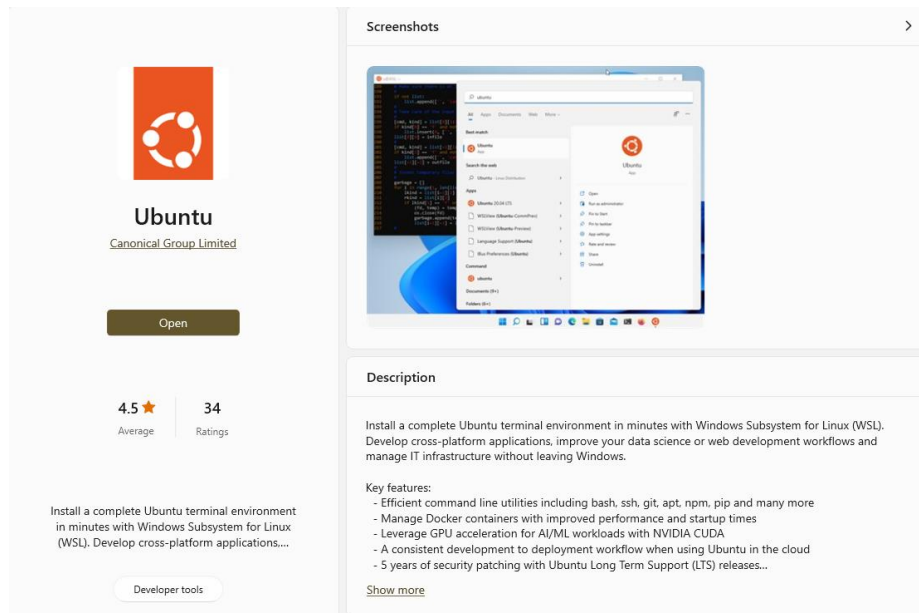
Desarrollo

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*

```

root@DESKTOP-K39Q5I7: /home/servidor
root@DESKTOP-K39Q5I7:/home/servidor# pip install apache-airflow
Collecting apache-airflow
  Downloading apache_airflow-2.7.2-py3-none-any.whl (12.9 MB)
    12.9/12.9 MB 1.2 MB/s eta 0:00:00
Collecting argcomplete>=1.10
  Downloading argcomplete-3.1.2-py3-none-any.whl (41 kB)
    41.5/41.5 KB 2.0 MB/s eta 0:00:00
Collecting gunicorn>=20.1.0
  Downloading gunicorn-21.2.0-py3-none-any.whl (80 kB)
    80.2/80.2 KB 1.4 MB/s eta 0:00:00
Collecting flask<2.3,>=2.2
  Downloading Flask-2.2.5-py3-none-any.whl (101 kB)
    101.8/101.8 KB 1.2 MB/s eta 0:00:00
Collecting apache-airflow-providers-common-sql
  Downloading apache_airflow_providers_common_sql-1.7.2-py3-none-any.whl (31 kB)
Collecting configupdater>=3.1.1
  Downloading ConfigUpdater-3.1.1-py2.py3-none-any.whl (34 kB)
Requirement already satisfied: blinker in /usr/lib/python3/dist-packages (from apache-airflow) (1.4)
Collecting markdown>=3.0
  Downloading Markdown-3.5-py3-none-any.whl (101 kB)
    101.7/101.7 KB 1.1 MB/s eta 0:00:00
Collecting dill>=0.2.2
  Downloading dill-0.3.7-py3-none-any.whl (115 kB)
    115.3/115.3 KB 1.1 MB/s eta 0:00:00
Collecting linkify-it-py>=2.0.0
  Downloading linkify_it_py-2.0.2-py3-none-any.whl (19 kB)
Collecting asgiref
  Downloading asgiref-3.7.2-py3-none-any.whl (24 kB)
Collecting opentelemetry-exporter-otlp
  Downloading opentelemetry_exporter_otlp-1.20.0-py3-none-any.whl (7.0 kB)

```

Iniciamos la base de datos de airflow

```

root@DESKTOP-K39Q5I7: /home/servidor
root@DESKTOP-K39Q5I7:/home/servidor# airflow db init
/usr/local/lib/python3.10/dist-packages/airflow/cli/commands/db_command.py:43 DeprecationWarning: Use `db migrate` instead to migrate the db and/or airflow connections create-default-connections
DB: sqlite:///root/airflow/airflow.db
[2023-10-15T21:03:55.678-0500] {migration.py:213} INFO - Context impl SQLiteImpl.
[2023-10-15T21:03:55.696-0500] {migration.py:216} INFO - Will assume non-transactional DDL.
INFO [alembic.runtime.migration] Context impl SQLiteImpl.
INFO [alembic.runtime.migration] Will assume non-transactional DDL.
INFO [alembic.runtime.migration] Running stamp_revision -> 405de8318b3a
WARNI [airflow.models.crypto] empty cryptography key - values will not be stored encrypted.
Initialization done
root@DESKTOP-K39Q5I7:/home/servidor#

```

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

```

root@DESKTOP-K39Q5I7: /home/servidor
Options:
-h, --help          show this help message and exit

airflow command error: unrecognized arguments: stop, see help above.
root@DESKTOP-K39Q5I7:/home/servidor# airflow standalone
standalone | Starting Airflow Standalone
standalone | Checking database is initialized
INFO [alembic.runtime.migration] Context impl SQLiteImpl.
INFO [alembic.runtime.migration] Will assume non-transactional DDL.
WARNI [unusual_prefix_13903d1433e73a905598409829be4c61c9cdea5f_example_kubernetes_executor] The example_kubernetes_executor example DAG requires the kubernetes provider. Please install it with: pip install apache-airflow[cncf.kubernetes]
WARNI [unusual_prefix_4c1c0063a36d033a56d4343efc723c3a45c1bd59_example_local_kubernetes_executor] Could not import DAGs in example_local_kubernetes_executor.py
Traceback (most recent call last):
  File "/usr/local/lib/python3.10/dist-packages/airflow/example_dags/example_local_kubernetes_executor.py", line 37, in <module>
    from kubernetes.client import models as k8s
ModuleNotFoundError: No module named 'kubernetes'
WARNI [unusual_prefix_4c1c0063a36d033a56d4343efc723c3a45c1bd59_example_local_kubernetes_executor] Install Kubernetes dependencies with: pip install apache-airflow[cncf.kubernetes]
WARNI [unusual_prefix_de310399792f7c66d0607289cf7f70223c00f7ca_example_python_operator] The virtualenv_python example task requires virtualenv, please install it.
WARNI [unusual_prefix_e1e1e746b8b87627f53519aca03f086299788fc2_tutorial_taskflow_api_virtualenv] The tutorial_taskflow_api_virtualenv example DAG requires virtualenv, please install it.
WARNI [unusual_prefix_45fb6c494355141389b80b1d50f93a2d0d272ee8_workday] Could not import pandas. Holidays will not be considered.
WARNI [airflow.models.crypto] empty cryptography key - values will not be stored encrypted.
standalone | Database ready
/usr/local/lib/python3.10/dist-packages/flask_limiter/extension.py:336 UserWarning: Using the in-memory storage for tracking rate limits as no storage was explicitly specified. This is not recommended for production use. See: https://flask-

```

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*

DAGs
Cluster Activity
Datasets
Security
Browse
Admin
Docs

Do not use **SQLite** as metadata DB in production – it should only be used for dev/testing. We recommend using Postgres or MySQL. [Click here](#) for more information.

Do not use the **SequentialExecutor** in production. [Click here](#) for more information.

DAGs

All

Active

Paused

Running

Failed

☐ Auto-refresh

refresh

info	DAG unfold_more	Owner unfold_more	Runs info	Schedule	Last Run unfold_more info	Next Run unfold_more info	Recent Tasks info	Actions	Links
<input type="radio"/>	dataset_consumes_1 consumes (dataset scheduled)	airflow	<div><div></div><div></div><div></div></div>	Dataset info	<div><div></div><div></div><div></div></div>	On schedule/timeout, fail	<div><div></div><div></div><div></div></div>	play_arrow delete_outlet refresh_horiz	
<input type="radio"/>	dataset_consumes_1_and_2 consumes (dataset scheduled)	airflow	<div><div></div><div></div><div></div></div>	Dataset info	<div><div></div><div></div><div></div></div>	0 of 2 datasets updated	<div><div></div><div></div><div></div></div>	play_arrow delete_outlet refresh_horiz	
<input type="radio"/>	dataset_consumes_1_never_scheduled consumes (dataset scheduled)	airflow	<div><div></div><div></div><div></div></div>	Dataset info	<div><div></div><div></div><div></div></div>	0 of 2 datasets updated	<div><div></div><div></div><div></div></div>	play_arrow delete_outlet refresh_horiz	
<input type="radio"/>	dataset_consumes_unknown_never_scheduled dataset scheduled	airflow	<div><div></div><div></div><div></div></div>	Dataset info	<div><div></div><div></div><div></div></div>	0 of 2 datasets updated	<div><div></div><div></div><div></div></div>	play_arrow delete_outlet refresh_horiz	
<input type="radio"/>	dataset_produces_1 dataset scheduled produces	airflow	<div><div></div><div></div><div></div></div>	Daily info	<div><div></div><div></div><div></div></div>	2023-10-15 00:00:00+00:00 info	<div><div></div><div></div><div></div></div>	play_arrow delete_outlet refresh_horiz	
<input type="radio"/>	dataset_produces_2 dataset scheduled produces	airflow	<div><div></div><div></div><div></div></div>	None info	<div><div></div><div></div><div></div></div>		<div><div></div><div></div><div></div></div>	play_arrow delete_outlet refresh_horiz	
<input type="radio"/>	example_bash_operator example example2	airflow	<div><div></div><div></div><div></div></div>	0 0 *** info	<div><div></div><div></div><div></div></div>	2023-10-15 00:00:00+00:00 info	<div><div></div><div></div><div></div></div>	play_arrow delete_outlet refresh_horiz	
<input type="radio"/>	example_branch_datetime_operator example	airflow	<div><div></div><div></div><div></div></div>	@daily info	<div><div></div><div></div><div></div></div>	2023-10-15 00:00:00+00:00 info	<div><div></div><div></div><div></div></div>	play_arrow delete_outlet refresh_horiz	
<input type="radio"/>	example_branch_datetime_operator_2 example	airflow	<div><div></div><div></div><div></div></div>	@daily info	<div><div></div><div></div><div></div></div>	2023-10-15 00:00:00+00:00 info	<div><div></div><div></div><div></div></div>	play_arrow delete_outlet refresh_horiz	
localhost:8080	example_branch_datetime_operator_3	airflow	<div><div></div><div></div><div></div></div>	@daily info	<div><div></div><div></div><div></div></div>	2023-10-15 00:00:00+00:00 info	<div><div></div><div></div><div></div></div>	play_arrow delete_outlet refresh_horiz	

Prueba de Trigger DAG: example_bash_operator

DAGs
Cluster Activity
Datasets
Security
Browse
Admin
Docs

02:53 UTC

AU

Trigger DAG: example_bash_operator

Logical date

Run id (Optional)

DAG conf Parameters

example_key:

Generated Configuration JSON

To access configuration in your DAG use `{{ dag_run.conf }}`. As `core_dag_run_conf_overrides_params` is set to `True`, so passing any configuration here will override task params which can be accessed via `{{ params }}`.

☒ Unpause DAG when triggered

Trigger

Cancel

DAGs
Cluster Activity
Datasets
Security
Browse
Admin
Docs

02:54 UTC

AU

DAG: example_bash_operator

Schedule: 0 0 ***

Next Run: 2023-10-16, 00:00:00

Grid

Graph

Calendar

Task Duration

Task Times

Landing Times

Gantt

Details

<> Code

Audit Log

16/10/2023 02:54:15

25

All Run Types

All Run States

Clear Filters

Press `SHIFT` + `J` for Shortcuts

refresh

cancel

queued

removed

retrying

running

scheduled

skipped

success

up_for_reschedule

up_for_retry

upstream_failed

no_status

example_bash_operator

2023-10-16, 00:00:00 UTC

Clear

Mark state as...

Details

Graph

Gantt

<> Code

runme_2

success

BashOperator

this_will_skip

queued

BashOperator

runme_1

success

BashOperator

run_after_loop

scheduled

BashOperator

runme_0

success

BashOperator

also_run_this

queued

BashOperator

run_this_last

EmptyOperator

Layout: Left → Right

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:

```
#
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
6# to you under the Apache License, Version 2.0 (the
7# "License"); you may not use this file except in compliance
8# with the License. You may obtain a copy of the License at
9#
10# http://www.apache.org/licenses/LICENSE-2.0
11#
12# Unless required by applicable law or agreed to in writing,
13# software distributed under the License is distributed on an
14# "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
15# KIND, either express or implied. See the License for the
16# specific language governing permissions and limitations
17# under the License.
18"""Example DAG demonstrating the usage of the BashOperator."""
19from __future__ import annotations
20
21import datetime
22
23import pendulum
24
25from airflow import DAG
26from airflow.operators.bash import BashOperator
27from airflow.operators.empty import EmptyOperator
28
29with DAG(
30    dag_id="example_bash_operator",
31    schedule="0 0 * * *",
32    start_date=pendulum.datetime(2021, 1, 1, tz="UTC"),
33    catchup=False,
34    dagrun_timeout=datetime.timedelta(minutes=60),
35    tags=["example", "example2"],
36    params={"example_key": "example_value"},
37) as dag:
38    run_this_last = EmptyOperator(
39        task_id="run_this_last",
```

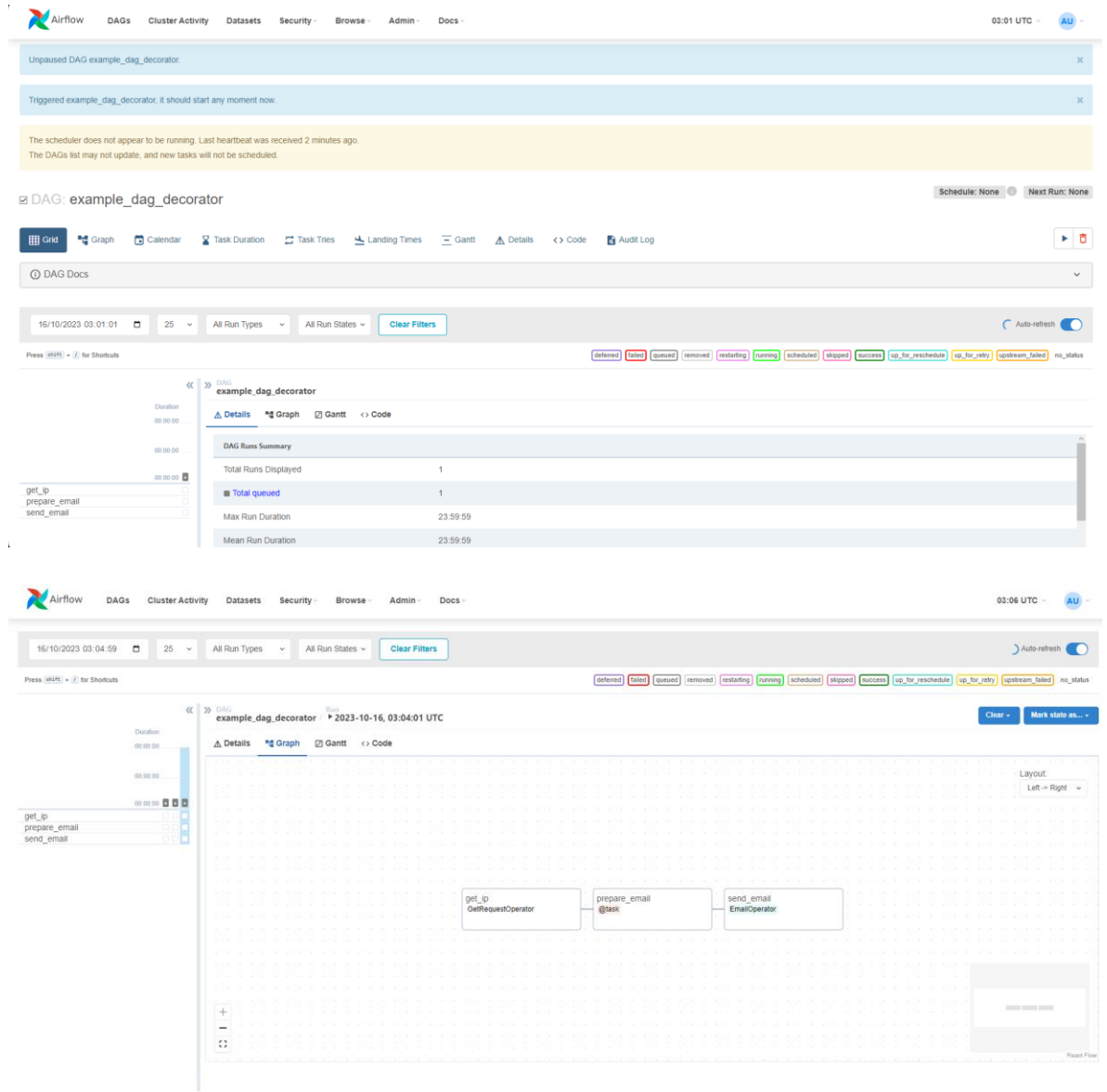
```

40     )|
41
42     # [START howto_operator_bash]
43     run_this = BashOperator(
44         task_id="run_after_loop",
45         bash_command="echo 1",
46     )
47     # [END howto_operator_bash]
48
49     run_this >> run_this_last
50
51     for i in range(3):
52         task = BashOperator(
53             task_id=f"runme_{i}",
54             bash_command='echo "{{ task_instance_key_str }}" && sleep
55 1',
56         )
57         task >> run_this
58     # [START howto_operator_bash_template]
59     also_run_this = BashOperator(
60         task_id="also_run_this",
61         bash_command='echo "ti_key={{ task_instance_key_str }}"',
62     )
63     # [END howto_operator_bash_template]
64     also_run_this >> run_this_last
65
66# [START howto_operator_bash_skip]
67this_will_skip = BashOperator(
68     task_id="this_will_skip",
69     bash_command='echo "hello world"; exit 99;',
70     dag=dag,
71)
72# [END howto_operator_bash_skip]
73this_will_skip >> run_this_last
74
75if __name__ == "__main__":
76    dag.test()
77

```

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
```



```

6# to you under the Apache License, Version 2.0 (the
7# "License"); you may not use this file except in compliance
8# with the License. You may obtain a copy of the License at
9#
10# http://www.apache.org/licenses/LICENSE-2.0
11#
12# Unless required by applicable law or agreed to in writing,
13# software distributed under the License is distributed on an
14# "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
15# KIND, either express or implied. See the License for the
16# specific language governing permissions and limitations
17# under the License.
18from __future__ import annotations
19
20from typing import Any
21
22import httpx
23import pendulum
24
25from airflow.decorators import dag, task
26from airflow.models.baseoperator import BaseOperator
27from airflow.operators.email import EmailOperator
28from airflow.utils.context import Context
29
30
31class GetRequestOperator(BaseOperator):
32    """Custom operator to send GET request to provided url"""
33
34    def __init__(self, *, url: str, **kwargs):
35        super().__init__(**kwargs)
36        self.url = url
37
38    def execute(self, context: Context):
39        return httpx.get(self.url).json()
40
41
42# [START dag_decorator_usage]
43@dag(
44    schedule=None,
45    start_date=pendulum.datetime(2021, 1, 1, tz="UTC"),
46    catchup=False,
47    tags=["example"],
48)
49def example_dag_decorator(email: str = "example@example.com"):
50    """
51    DAG to send server IP to email.
52

```

```

53     :param email: Email to send IP to. Defaults to example@example.com.
54     """
55     get_ip = GetRequestOperator(task_id="get_ip",
url="http://httpbin.org/get")
56
57     @task(multiple_outputs=True)
58     def prepare_email(raw_json: dict[str, Any]) -> dict[str, str]:
59         external_ip = raw_json["origin"]
60         return {
61             "subject": f"Server connected from {external_ip}",
62             "body": f"Seems like today your server executing Airflow is
connected from IP {external_ip}<br>",
63         }
64
65     email_info = prepare_email(get_ip.output)
66
67     EmailOperator(
68         task_id="send_email", to=email, subject=email_info["subject"],
html_content=email_info["body"]
69     )
70
71
72 example_dag = example_dag_decorator()
73 # [END dag_decorator_usage]
74

```

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.