

Curso de **Fundamentos de Apache Airflow**

Eric Bellet



@eric_bellet



linkedin.com/in/belleteric



Prerrequisitos para el curso



Python intermedio



Docker básico



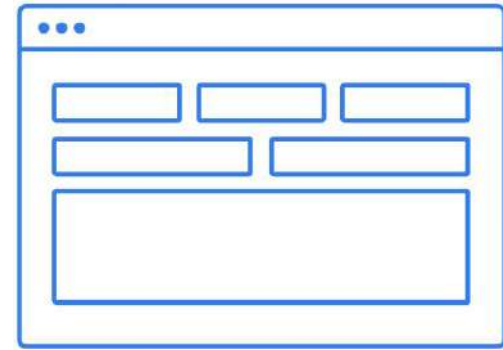
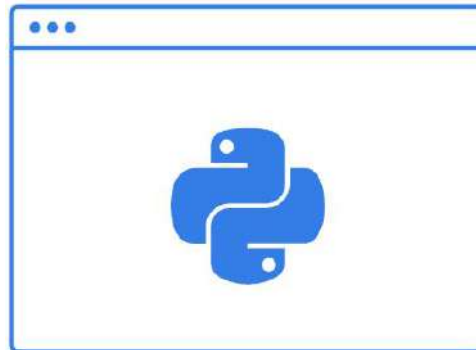
Línea de comandos básico

¿Qué es Airflow?



¿Qué es Airflow?

Airflow es una plataforma creada por la comunidad para crear, programar y supervisar flujos de trabajo de forma programada.





Breve historia



Maxime Beauchemin



Airflow was released on October, 2014.



Airflow was made open source on March, 2016.



Airflow was made a top-level Apache Software Foundation on January, 2019.



Airflow 2.0 was released on December 17th, 2020.

**¿Para qué
sirve Airflow?**



Terminologías

Flujos de datos

==

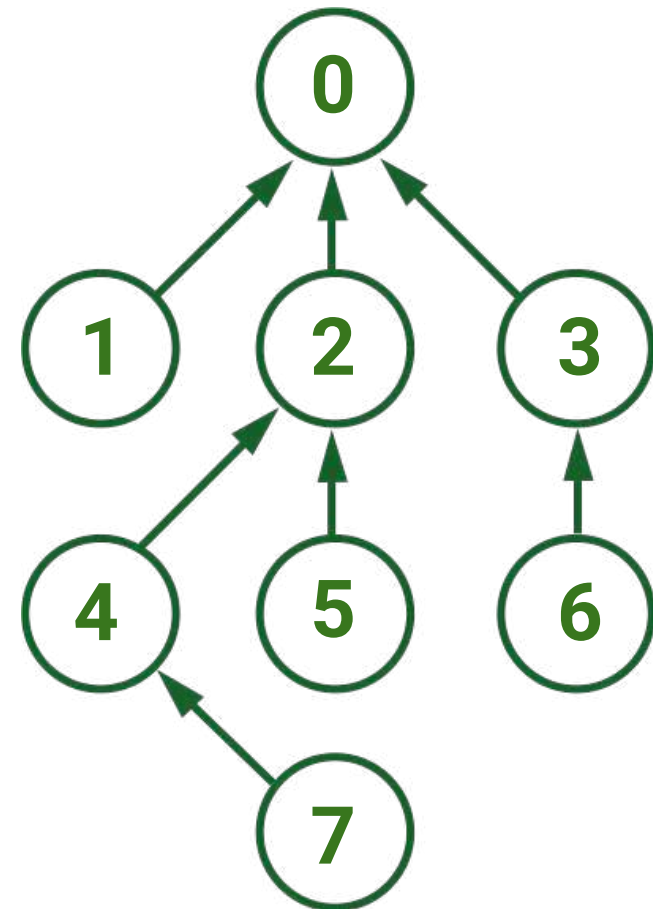
Workflow

==

Data pipelines

==

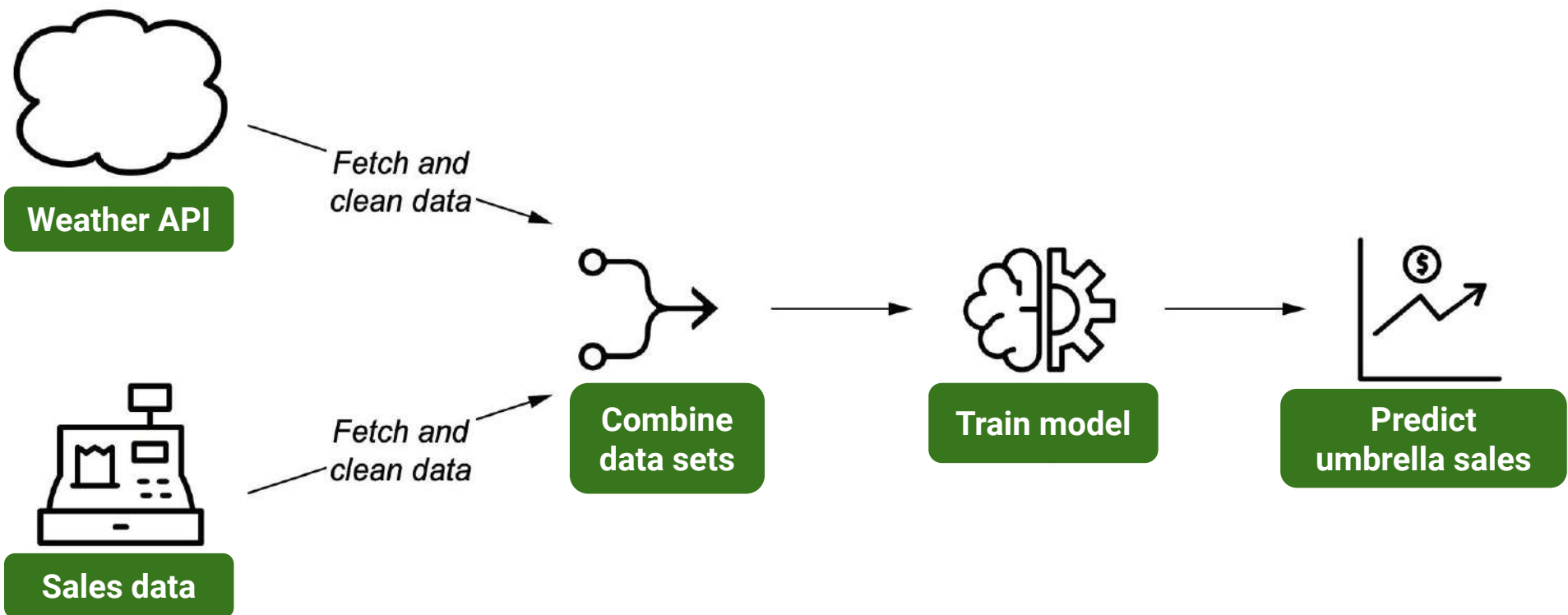
Pipelines de datos



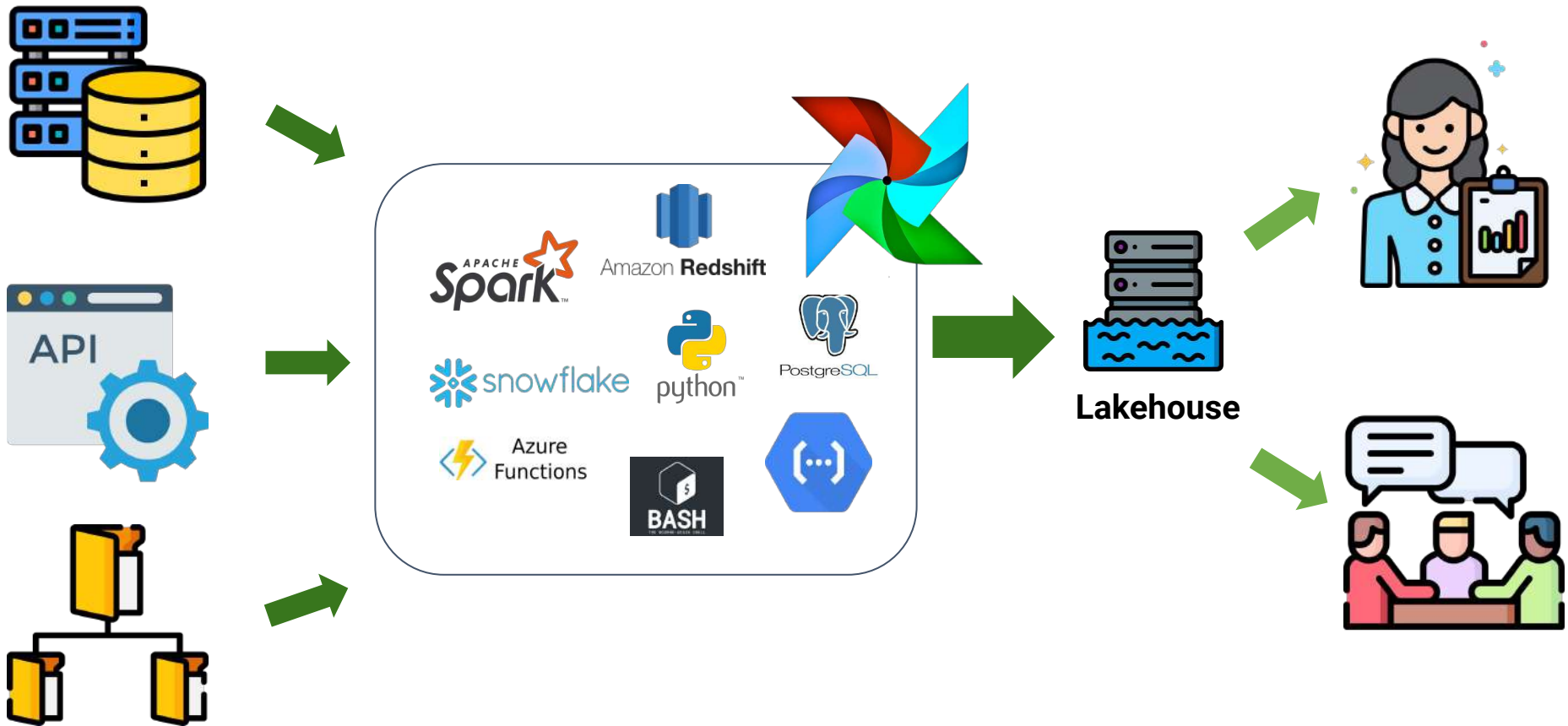
Grafo



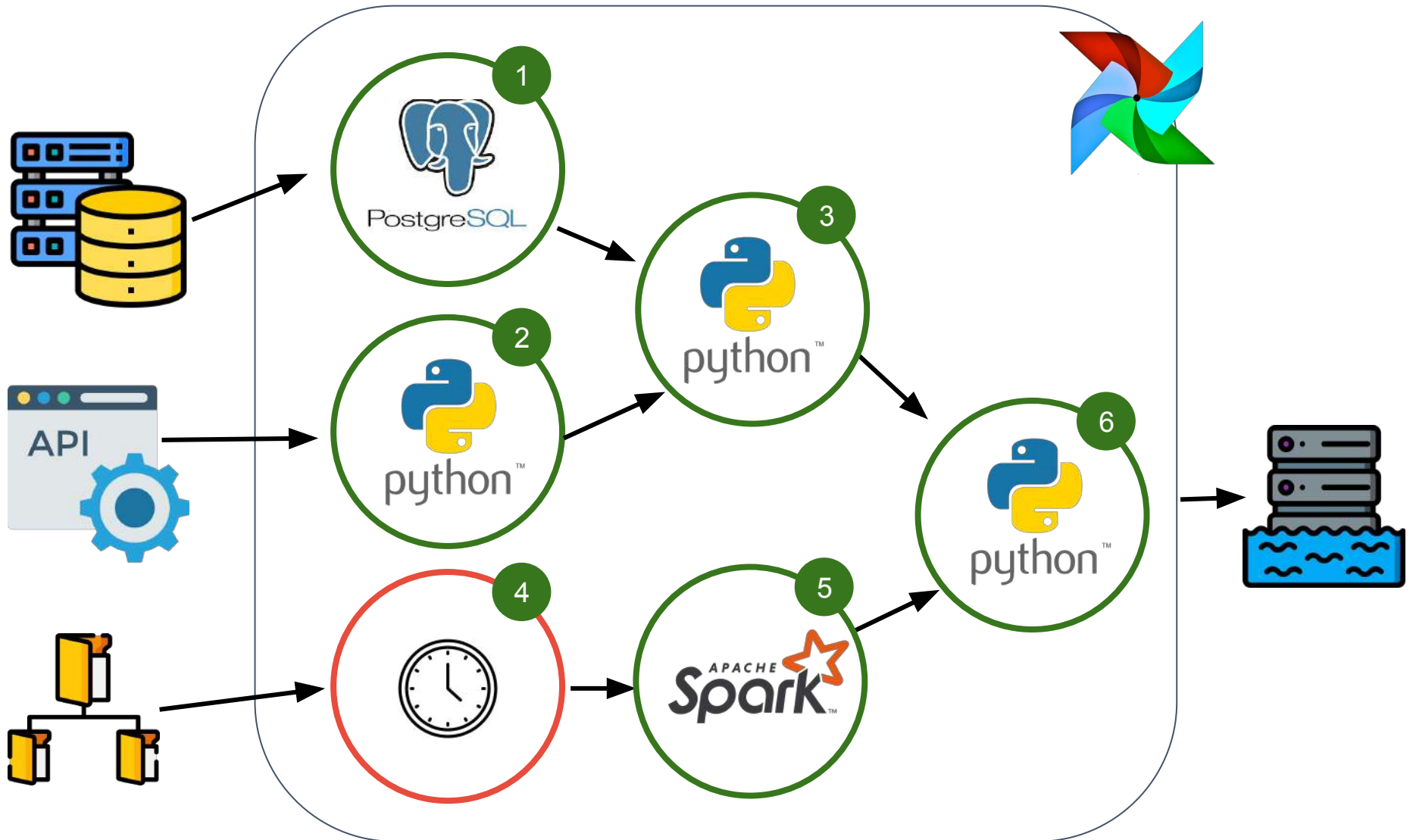
¿Para qué sirve Airflow?



¿Para qué sirve Airflow?



Representación: grafo



**¿Por qué
usar Airflow?**



Airflow

DAGs

Security

Browse

Admin

Docs



python™

21:11 UTC

RH

DAGs

All 26

Active 10

Paused 16

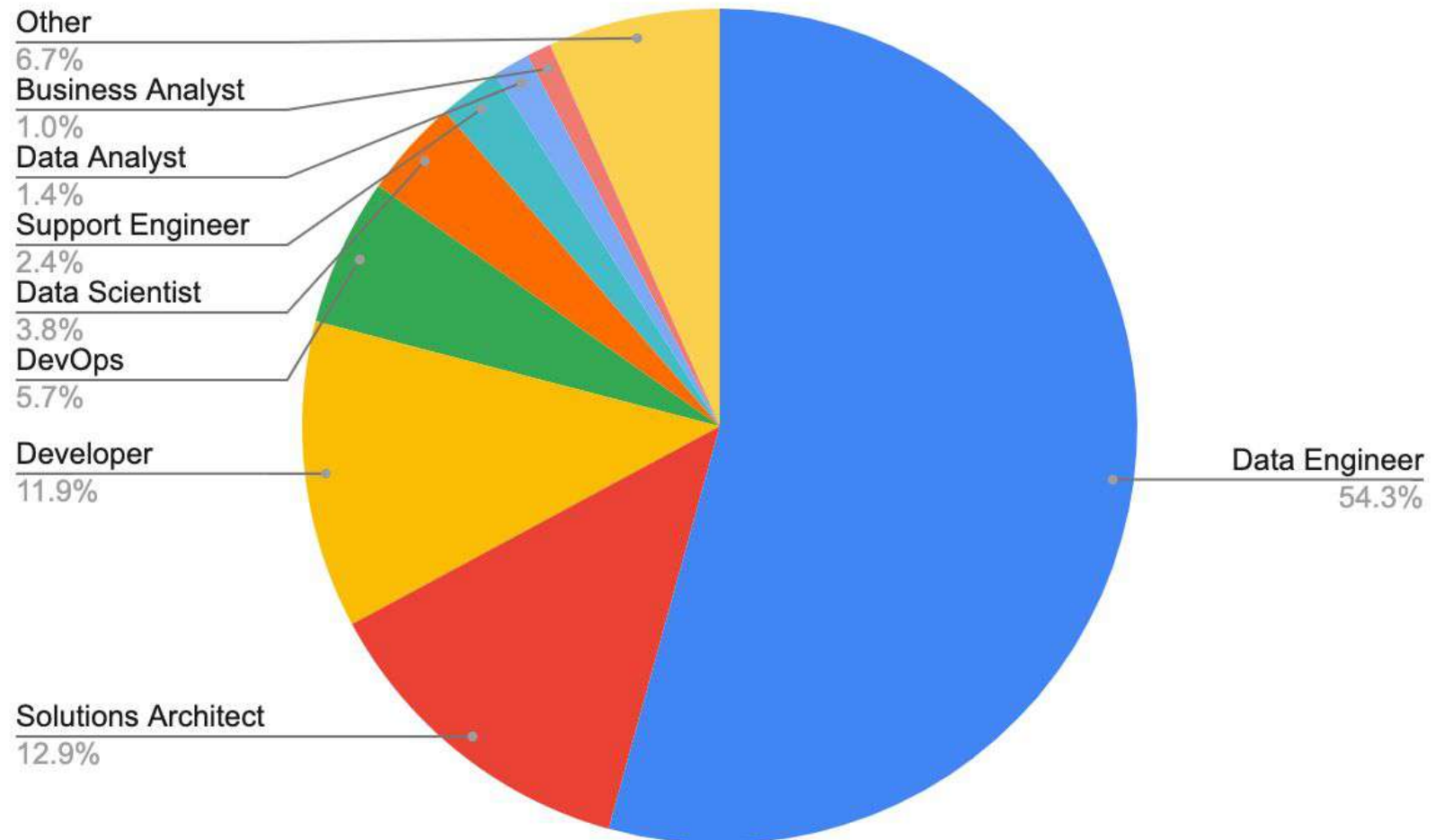
Filter DAGs by tag

Search DAGs

DAG	Owner	Runs	Schedule	Last Run	Recent Tasks	Actions	Links
<input checked="" type="checkbox"/> example_bash_operator example example2	airflow	<div><div>2</div><div></div><div></div></div>	00***	2020-10-26, 21:08:11	<div><div>6</div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>		...
<input checked="" type="checkbox"/> example_branch_dop_operator_v3 example	airflow	<div><div></div><div></div><div></div></div>	*/1****		<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>		...
<input type="checkbox"/> example_branch_operator example example2	airflow	<div><div></div><div>1</div><div></div></div>	@daily	2020-10-23, 14:09:17	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>		...
<input checked="" type="checkbox"/> example_complex example example2 example3	airflow	<div><div>1</div><div>1</div><div></div></div>	None	2020-10-26, 21:08:04	<div><div>37</div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>		...
<input checked="" type="checkbox"/> example_external_task_marker_child	airflow	<div><div></div><div>1</div><div></div></div>	None	2020-10-26, 21:07:33	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>		...
<input checked="" type="checkbox"/> example_external_task_marker_parent	airflow	<div><div></div><div>1</div><div></div></div>	None	2020-10-26, 21:08:34	<div><div>1</div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>		...
<input checked="" type="checkbox"/> example_kubernetes_executor example example2	airflow	<div><div></div><div></div><div></div></div>	None		<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>		...
<input checked="" type="checkbox"/> example_kubernetes_executor_config example3	airflow	<div><div></div><div>1</div><div></div></div>	None	2020-10-26, 21:07:40	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>		...
<input checked="" type="checkbox"/> example_nested_branch_dag example	airflow	<div><div></div><div>1</div><div></div></div>	@daily	2020-10-26, 21:07:37	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>		...
<input type="checkbox"/> example_passing_params_via_test_command example	airflow	<div><div></div><div></div><div></div></div>	*/1****		<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>		...

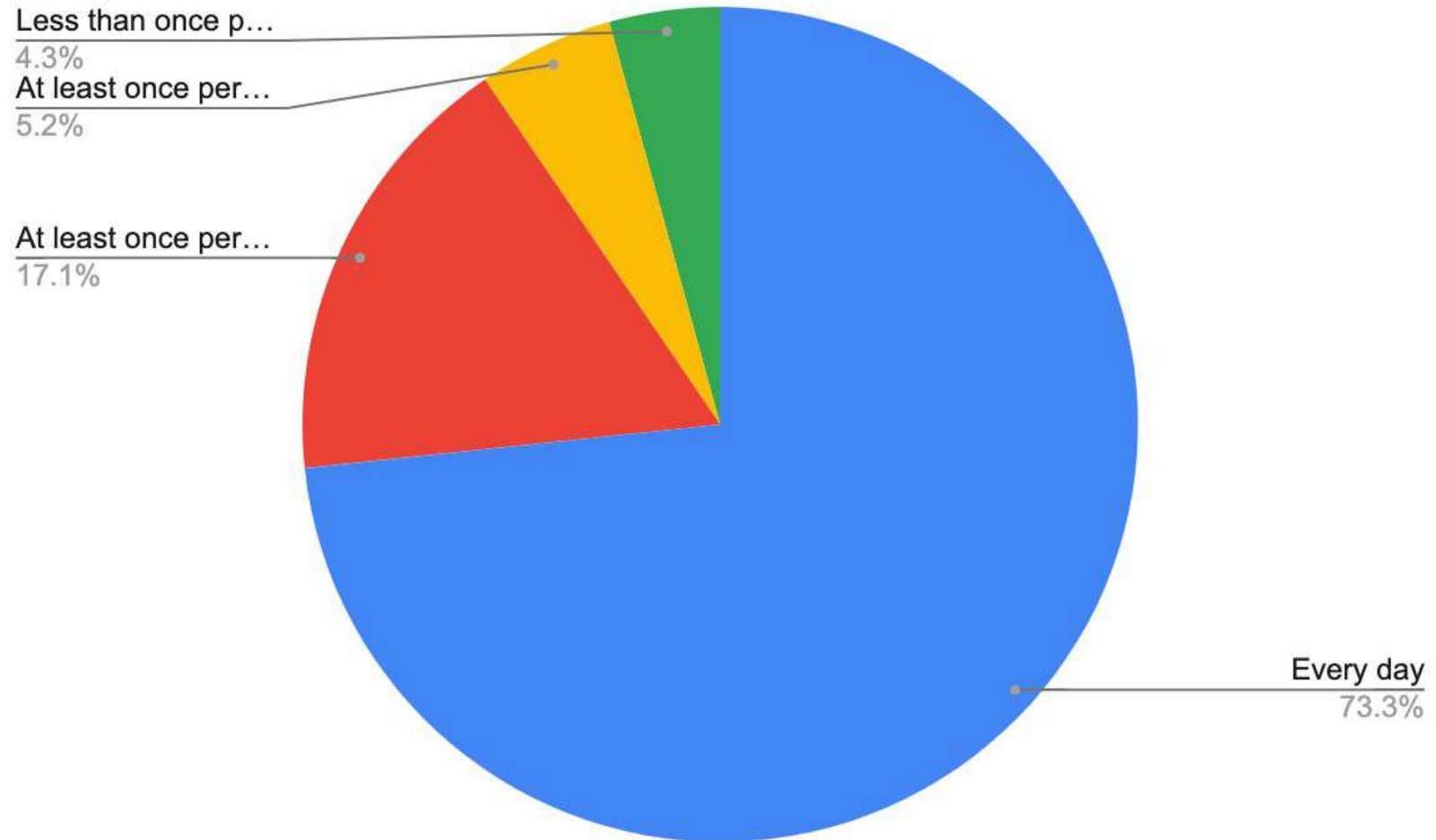


¿Qué perfiles suelen usar Airflow?



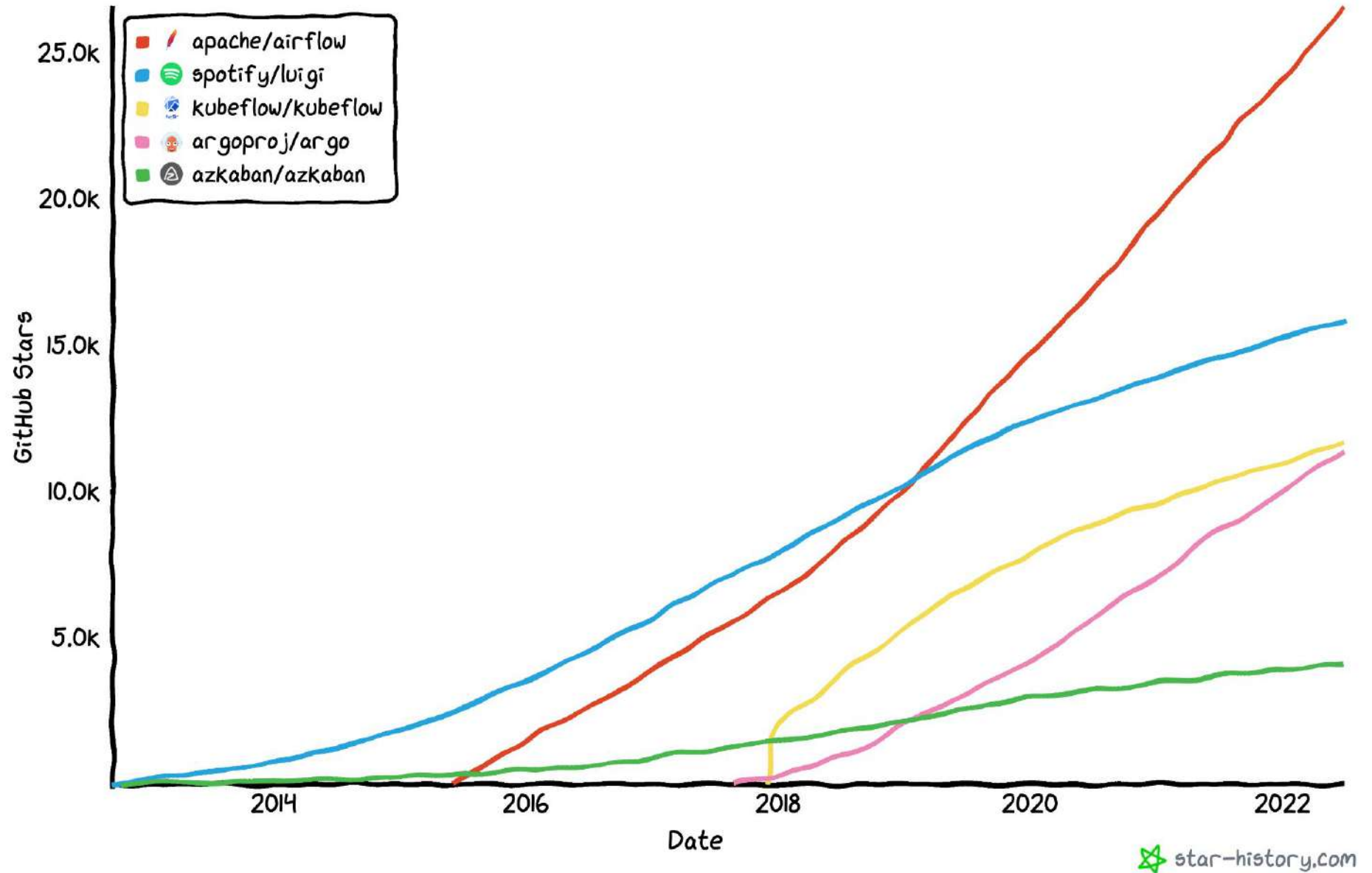
<https://airflow.apache.org/blog/airflow-survey-2022/>

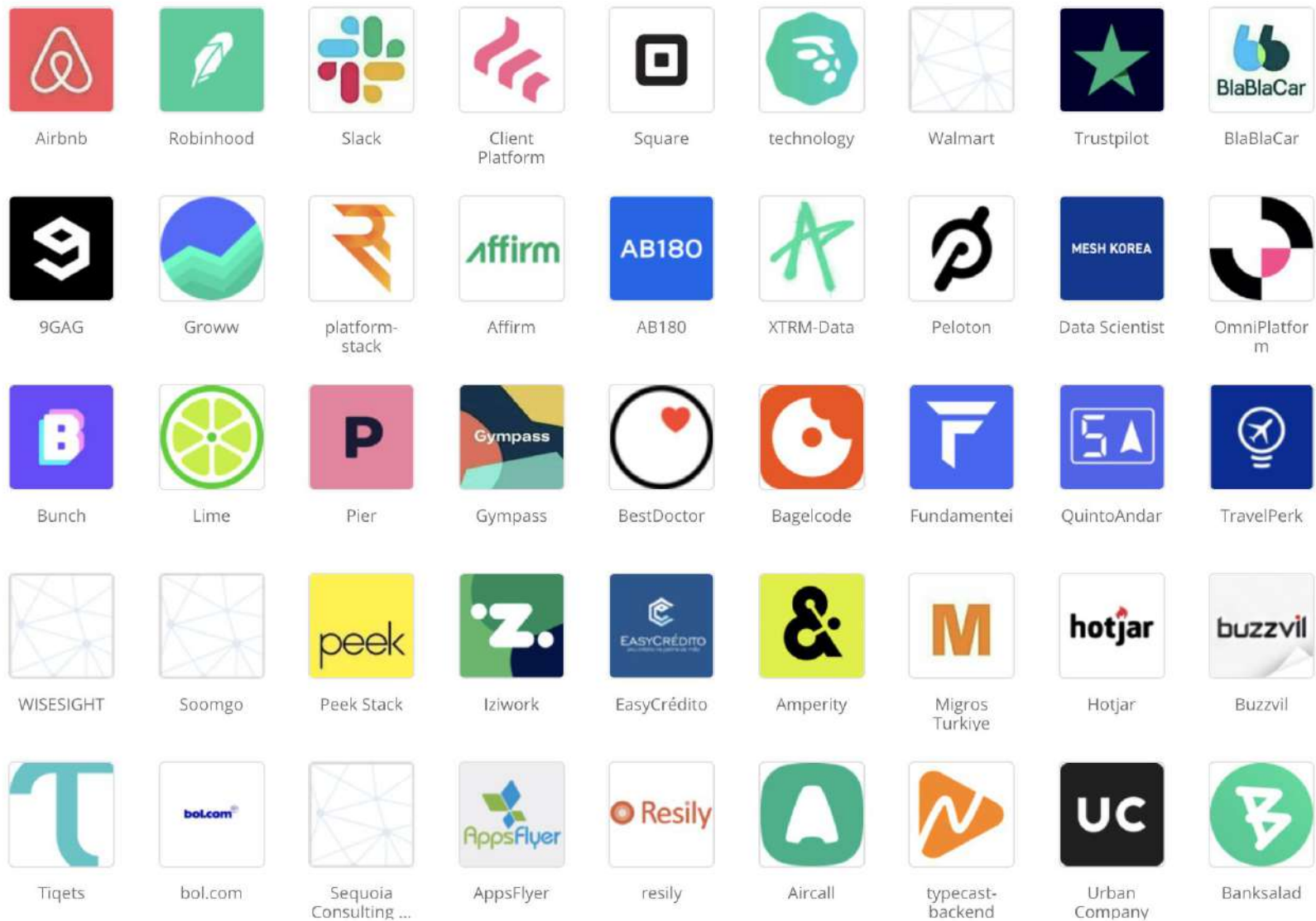
¿Qué tan seguido interactúas con Airflow?



<https://airflow.apache.org/blog/airflow-survey-2022/>

Star History





COMPANIES WE TRACK USING APACHE AIRFLOW

 **13,323**  **+30.34%**
12 MONTHS CHANGE

<https://stackshare.io/airflow>

<https://discovery.hgdata.com/product/apache-airflow>

Resumen



Resumen

1. Desarrollar “workflows” mediante Python.
2. Orquestar procesos.
3. Monitorear ejecuciones.
4. Una herramienta muy usada.
5. Un “job scheduler” con esteroides (CRON ++).

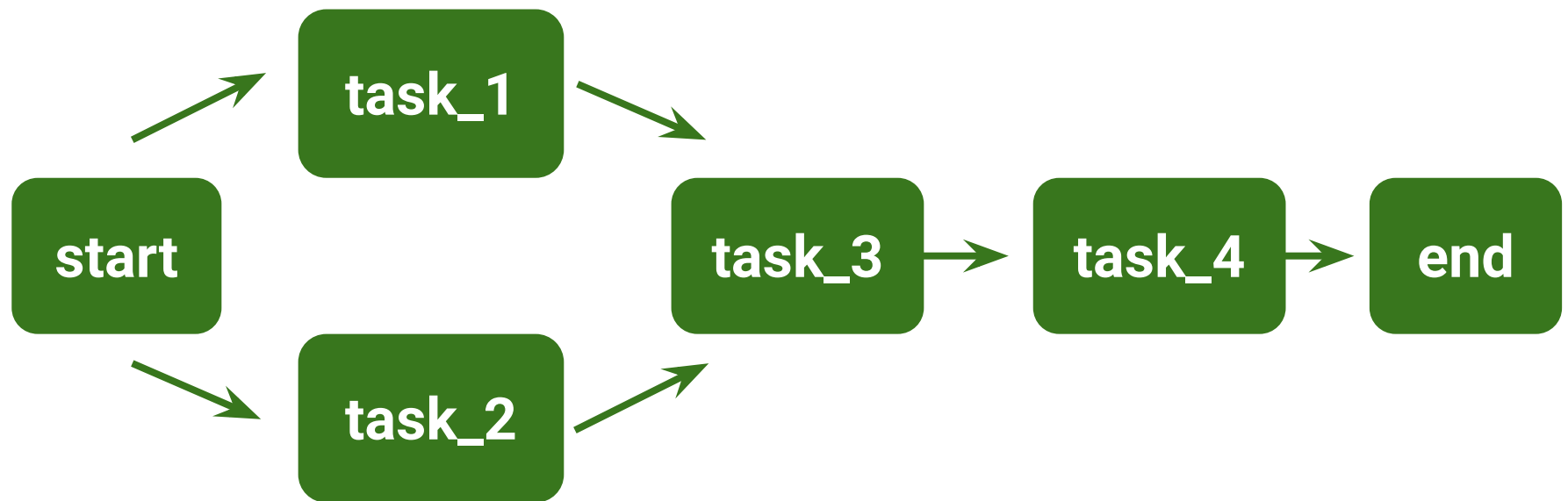
Conceptos básicos



DAG

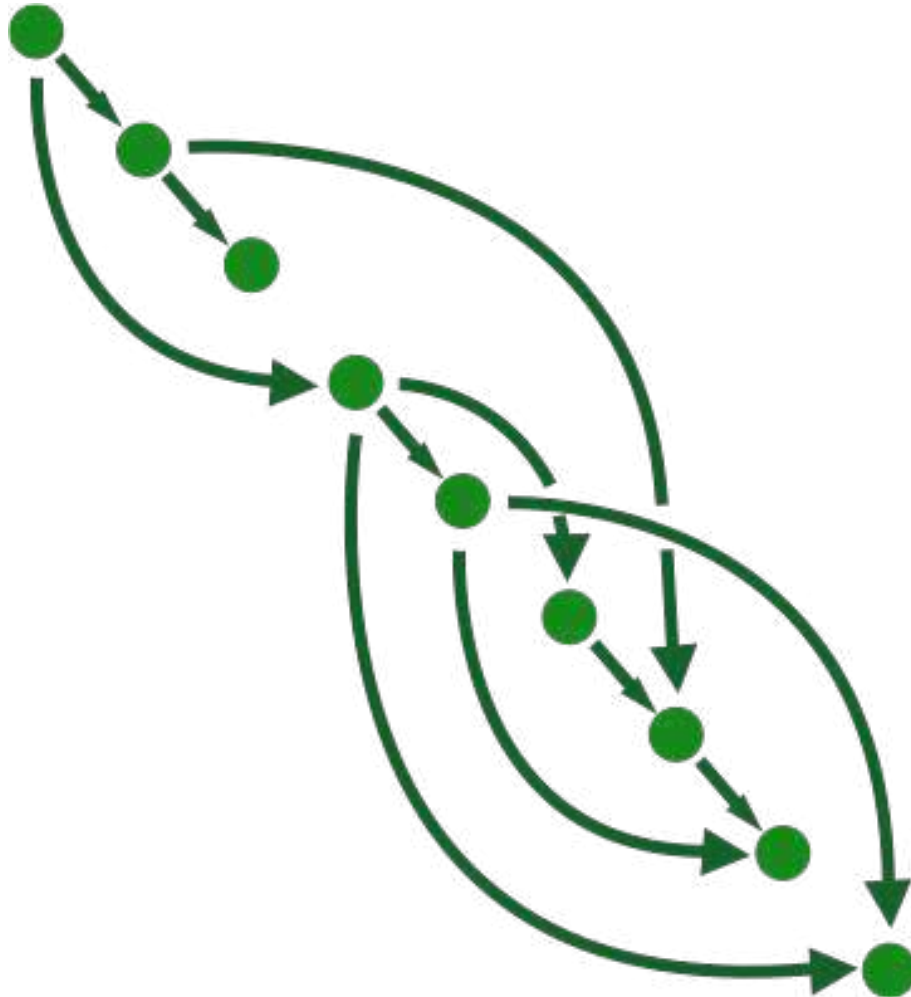


Flujo de trabajo (workflow)





DAG (Directed Acyclic Graph)



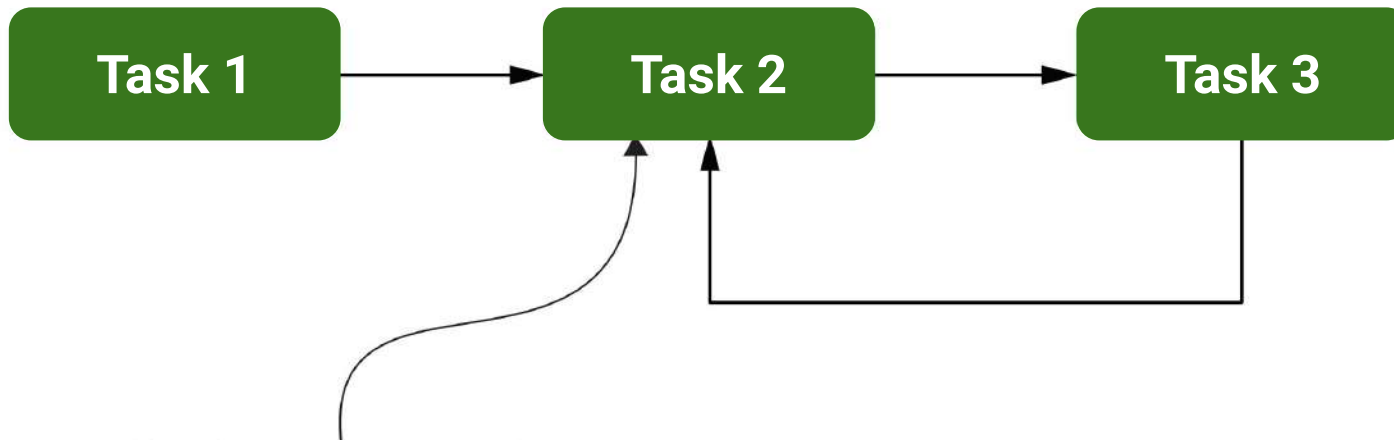


DAG (Directed Acyclic Graph)

A directed *acyclic* graph (DAG) of tasks



A directed *cyclic* graph of tasks



Task 2 will never be able to execute,
due to its dependency on task 3,
which in turn depends on task 2.

```
from airflow import DAG
```

```
with DAG(  
    'tutorial',  
    default_args={  
        'depends_on_past': False,  
        'email': ['airflow@example.com'],  
        'email_on_failure': False,  
        'email_on_retry': False,  
        'retries': 1,  
        'retry_delay': timedelta(minutes=5)  
    },  
    description='A simple tutorial DAG',  
    schedule_interval=timedelta(days=1),  
    start_date=datetime(2021, 1, 1),  
    catchup=False,  
    tags=['example'],  
) as dag:
```


DAG (Directed Acyclic Graph)

[DAGs](#)[Security](#)[Browse](#)[Admin](#)[Docs](#)

10:46 PDT (-07:00)



DAG: example_bash_operator

success schedule: 0 0 ***[Tree](#)[Graph](#)[Calendar](#)[Task Duration](#)[Task Tries](#)[Landing Times](#)[Gantt](#)[Details](#)[Code](#)

2021-06-02T09:27:27-07:00

Runs

25



Run

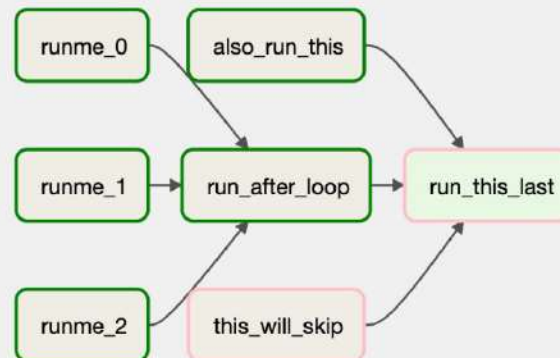
manual__2021-06-02T16:27:26.797940+00:00

Layout

Left > Right



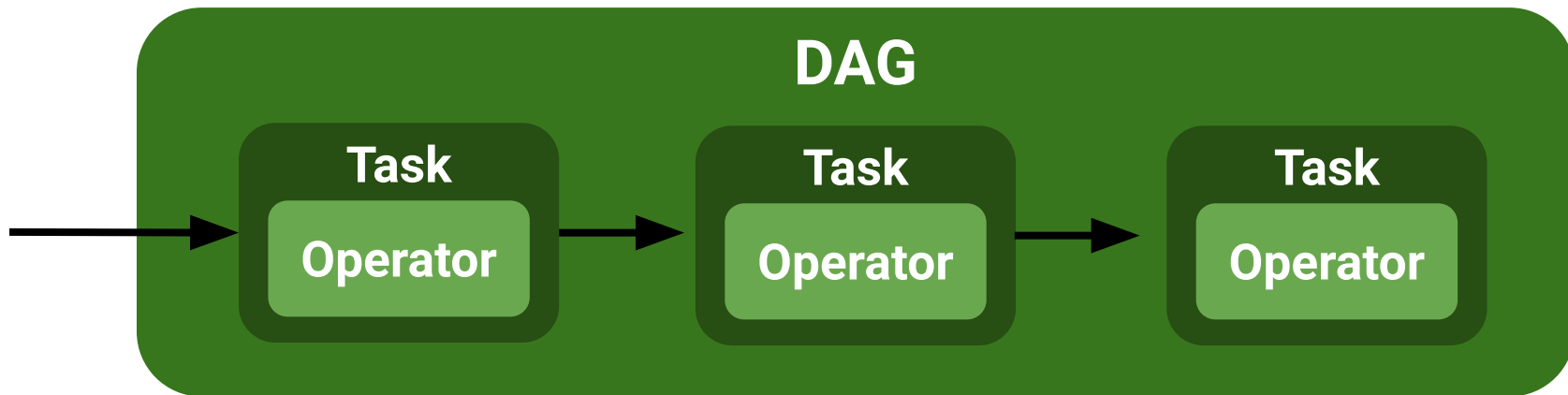
Find Task...

[Update](#)[BashOperator](#) [DummyOperator](#)[queued](#)[running](#)[success](#)[failed](#)[up_for_retry](#)[up_for_reschedule](#)[upstream_failed](#)[skipped](#)[scheduled](#)[no_status](#)☐ Auto-refresh

Tasks y Operators



Tasks



Existen 3 tipos básicos de tasks:

1. Operators
2. Sensors
3. TaskFlow-decorated @task



Operators



BashOperator
executes a bash
command.



PythonOperator
calls an arbitrary
Python function.

```
from airflow.operators.bash import BashOperator
```

```
t1 = BashOperator(  
    task_id='print_hello_bash',  
    bash_command='echo "Hello world!"',  
)
```

```
from airflow.operators.python import PythonOperator
```

```
def print_hello():  
    print('Hello world!')
```

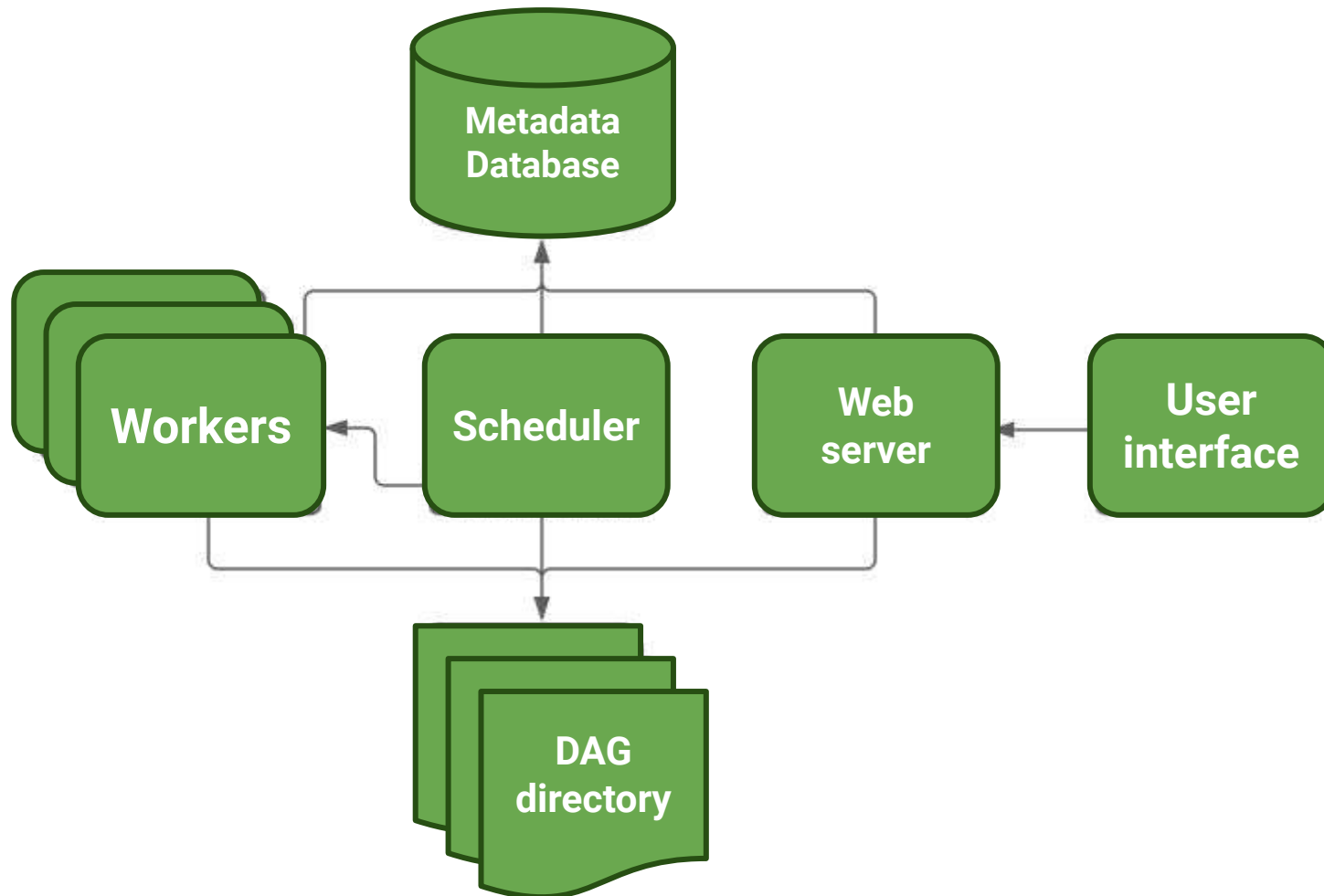
```
t2 = PythonOperator(  
    task_id='print_hello_python',  
    python_callable=print_hello  
)
```



Scheduler



Scheduler





Resumen



Resumen

1. Un flujo de trabajo se representa como un **DAG**.
2. Un **task** es un componente interno para ejecutar operaciones.
3. Un **operator** representa una única unidad de trabajo.
4. El **scheduler** se encarga de ejecutar los DAGs en intervalos de tiempo.

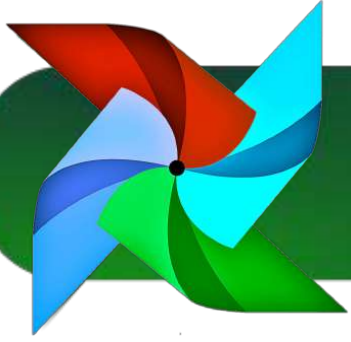
Instalación y configuración



Instalación

- Using released sources.
- Using PyPI.
- Using Production Docker Images.
- Using Official Airflow Helm Chart.
- Using Managed Airflow Services.
- Using 3rd-party images, charts, deployments.

<https://airflow.apache.org/docs/apache-airflow/stable/installation/index.html>



Instalación usando Docker

`docker-compose.yaml`

To deploy Airflow on Docker Compose, you should fetch [docker-compose.yaml](#).

```
curl -Lf0 'https://airflow.apache.org/docs/apache-airflow/2.3.3/docker-compose.yaml'
```

<https://airflow.apache.org/docs/apache-airflow/stable/start/docker.html>



Configuraciones



Referencia de configuración

Las configuraciones disponibles de Airflow se pueden establecer en el archivo **airflow.cfg** o mediante variables de entorno.

<https://airflow.apache.org/docs/apache-airflow/stable/configurations-ref.html>

Quiz

volumes:

- ./dags:/opt/airflow/dags
- ./logs:/opt/airflow/logs
- ./plugins:/opt/airflow/plugins

airflow-webserver:

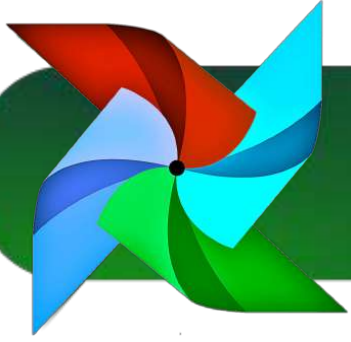
<<: *airflow-common

command: webserver

ports:

- 8080:8080

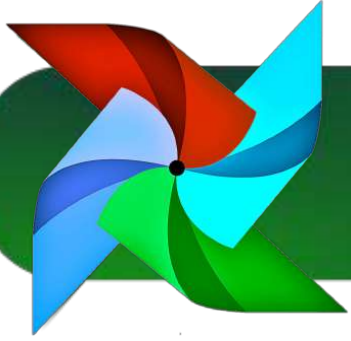
Variables y conexiones



Configurando variables

```
from airflow.models import Variable

org = Variable.get("organization")
foo_json = Variable.get("foo_baz",
                        deserialize_json=True)
```

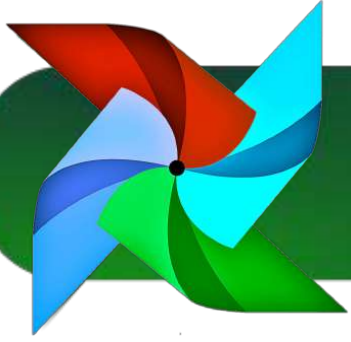


Configurando conexiones

```
from airflow.providers.postgres.operators.postgres import PostgresOperator

populate_pet_table = PostgresOperator(
    task_id="populate_pet_table",
    postgres_conn_id="my_postgres_conn",
    sql="sql/pet_schema.sql",
)
```

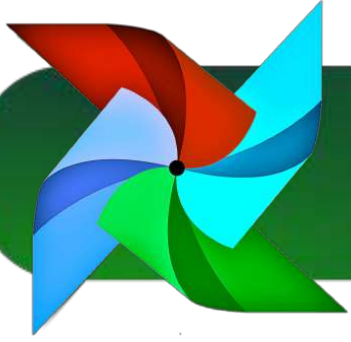
Implementando un DAG



Declarando un DAG

Standard constructor

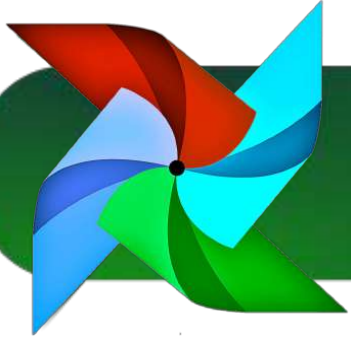
```
my_dag = DAG("my_dag_name", start_date=pendulum.datetime(2021, 1, 1, tz="UTC"),  
             schedule_interval="@daily", catchup=False)  
op = EmptyOperator(task_id="task", dag=my_dag)
```



Declarando un DAG

Context manager

```
with DAG(  
    "my_dag_name", start_date=pendulum.datetime(2021, 1, 1, tz="UTC"),  
    schedule_interval="@daily", catchup=False  
) as dag:  
    op = EmptyOperator(task_id="task")
```



Declarando un DAG

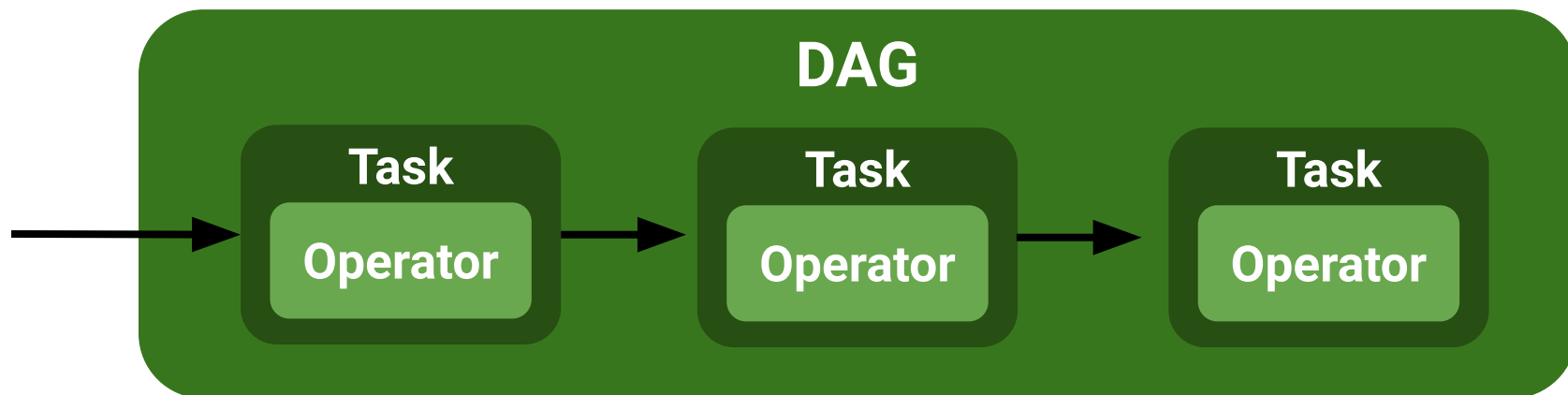
@dag decorator

```
@dag(start_date=pendulum.datetime(2021, 1, 1, tz="UTC"),  
      schedule_interval="@daily", catchup=False)  
def generate_dag():  
    op = EmptyOperator(task_id="task")  
  
dag = generate_dag()
```

Bash operator

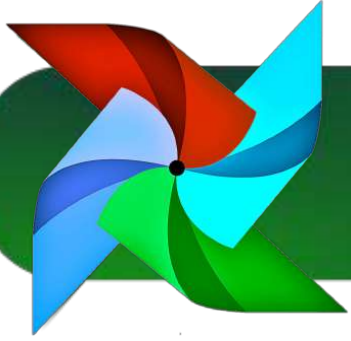


Declarando un operador



<https://airflow.apache.org/docs/apache-airflow/stable/concepts/operators.html>

Python operator



Python operator

```
from airflow.operators.python import PythonOperator

def print_test():
    return 'Test'

t2 = PythonOperator(
    task_id='print_test',
    python_callable=print_test
)
```

Dependencias entre tareas



Definiendo dependencias entre tareas

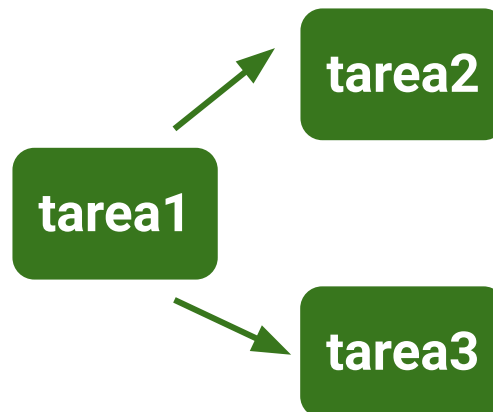
```
# Opción 1:  
tarea1.set_downstream(tarea2)  
tarea2.set_downstream(tarea3)  
  
# Opción 2: bitshift operators  
tarea1 >> tarea >> tarea3
```





Definiendo dependencias entre tareas

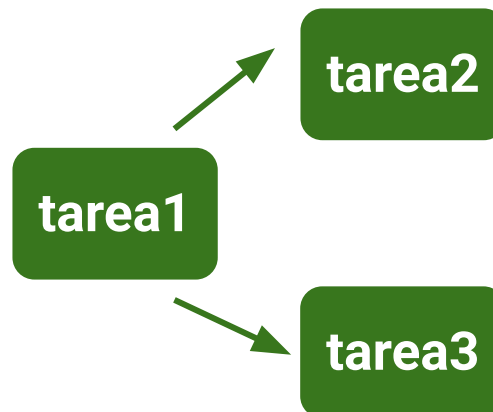
```
# Opción 1:  
tarea1.set_downstream([tarea2, tarea3])  
  
# Opción 2: bitshift operators  
tarea1 >> [tarea2, tarea3]
```





Definiendo dependencias entre tareas

```
# Opción 1:  
tarea1.set_upstream([tarea2, tarea3])  
  
# Opción 2: bitshift operators  
tarea1 << [tarea2, tarea3]
```





Custom operator



Creando un custom operator

```
from airflow.models.baseoperator import BaseOperator

class HelloOperator(BaseOperator):
    def __init__(self, name: str, **kwargs) -> None:
        super().__init__(**kwargs)
        self.name = name

    def execute(self, context):
        message = f"Hello {self.name}"
        print(message)
        return message
```

<https://airflow.apache.org/docs/apache-airflow/stable/howto/custom-operator.html>



Resumen



Resumen

1. Implementar un **DAG**.
2. Implementar un **operator**.
3. Definir **dependencias** entre las tareas.
4. Implementar un **custom operator**.

Orquestrar y monitorizar procesos



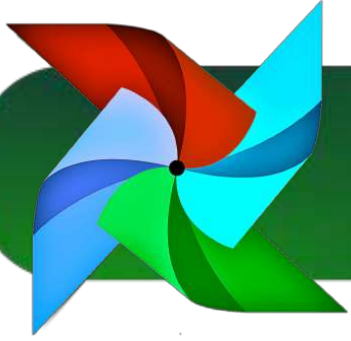
Orquestando un DAG

preset	meaning	cron
None	Don't schedule, use for exclusively "externally triggered" DAGs	
@once	Schedule once and only once	
@hourly	Run once an hour at the beginning of the hour	0 * * * *
@daily	Run once a day at midnight	0 0 * * *
@weekly	Run once a week at midnight on Sunday morning	0 0 * * 0
@monthly	Run once a month at midnight of the first day of the month	0 0 1 * *
@quarterly	Run once a quarter at midnight on the first day	0 0 1 */3 *
@yearly	Run once a year at midnight of January 1	0 0 1 1 *

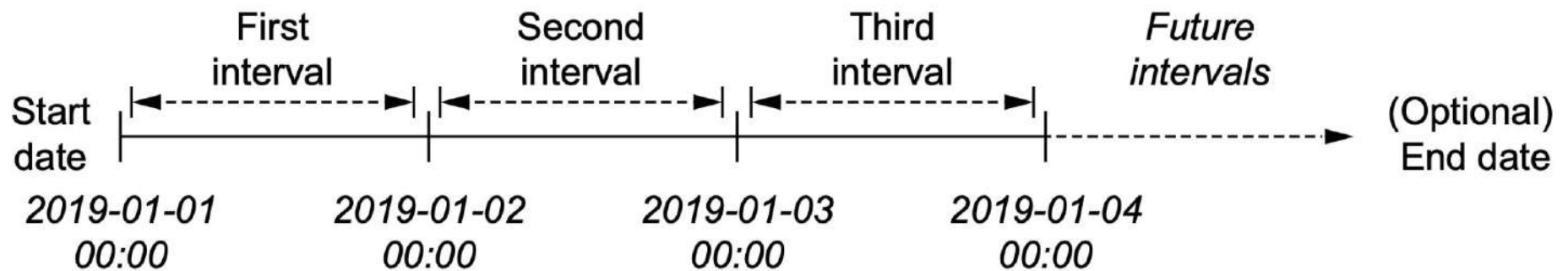
<https://airflow.apache.org/docs/apache-airflow/stable/dag-run.html?highlight=cron>



<https://crontab.guru/>



Orquestando un DAG





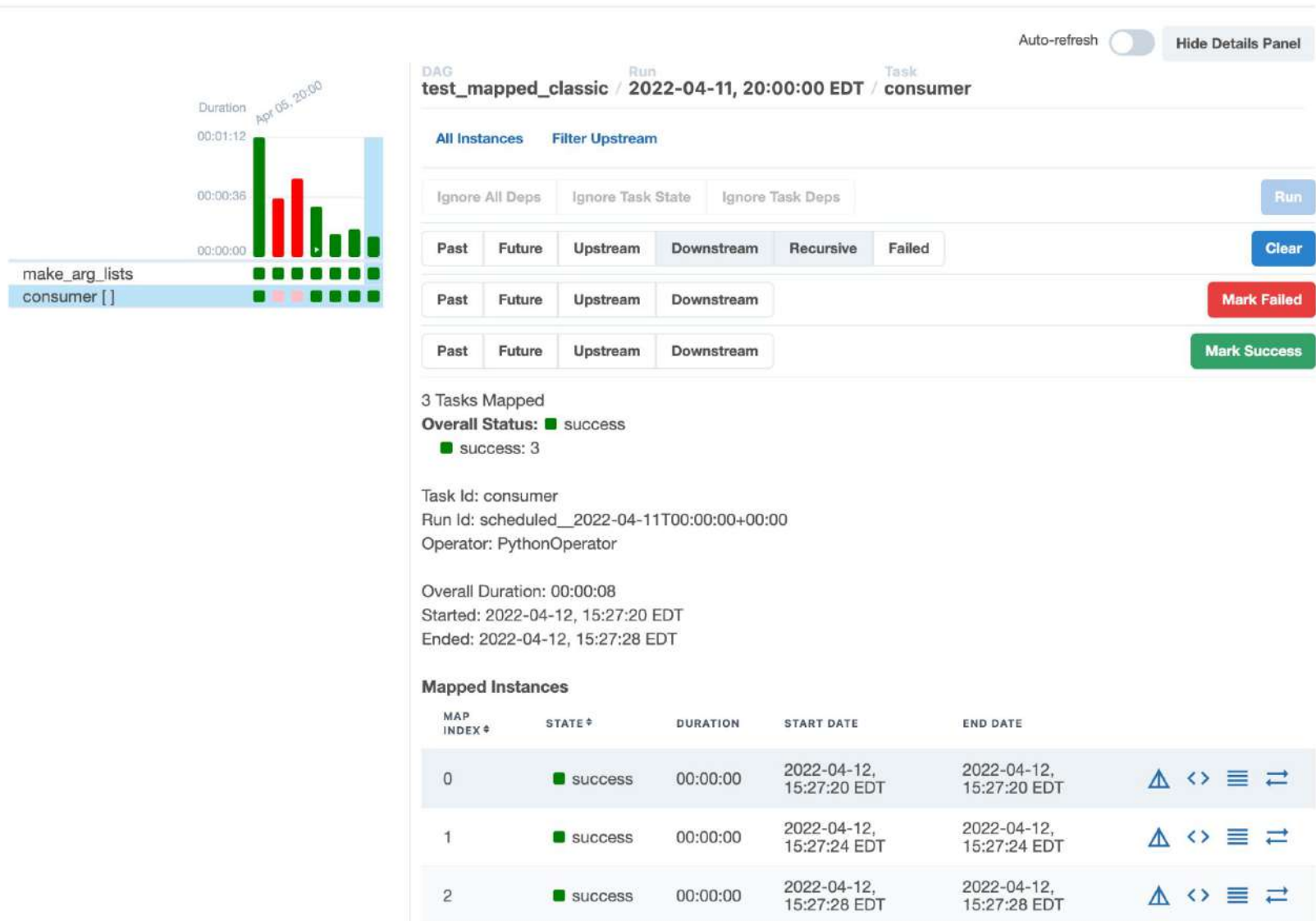
Monitoring



Ejemplos de fallos

1. La API a la que queremos acceder está caída.
2. Tenemos un error de lógica en nuestro código.
3. Algún proceso del que dependemos se ejecutó incorrectamente.
4. Se ha borrado una tabla de una base de datos.
5. Un sensor llegó al timeout.

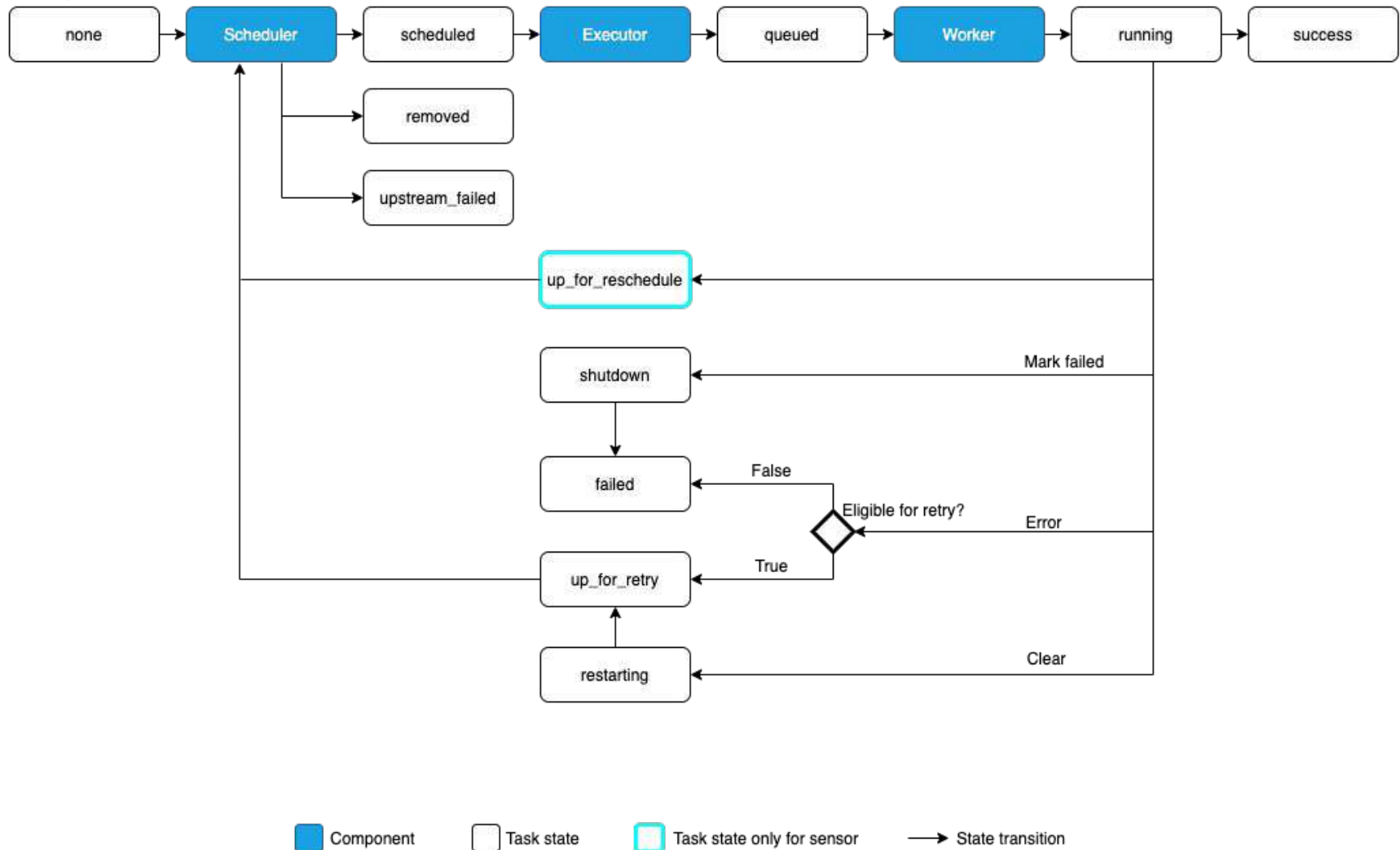
Monitorando un DAG



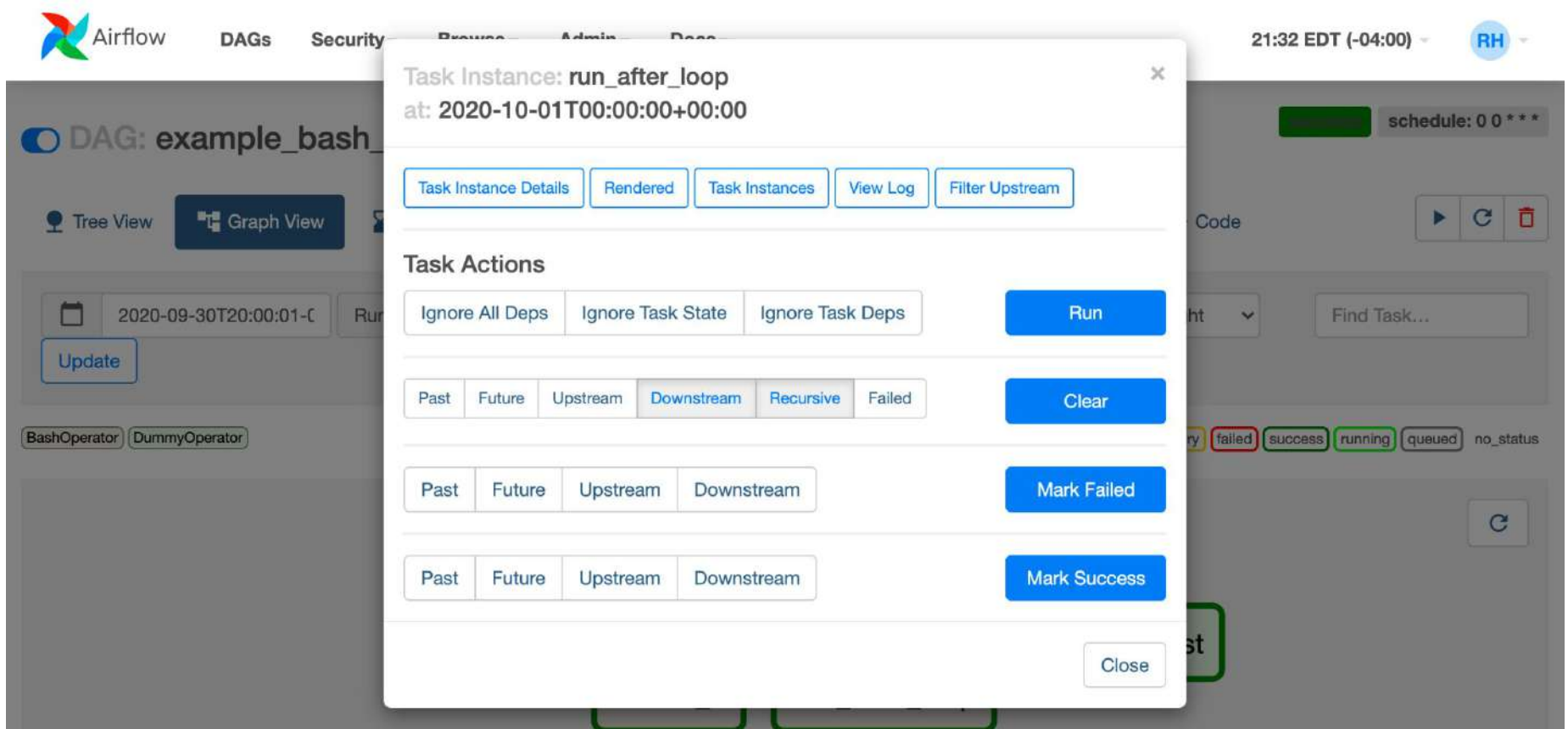


Task actions

Estados de los tasks



Task actions



The image shows the Apache Airflow web interface. In the background, the DAG 'example_bash' is displayed in 'Graph View'. The interface includes a top navigation bar with 'Airflow', 'DAGs', 'Security', 'Browse', 'Admin', and 'Docs'. On the right, the time is '21:32 EDT (-04:00)' and the user is 'RH'. The DAG details section shows a 'Tree View' and 'Graph View' toggle, a date range '2020-09-30T20:00:01-C', and an 'Update' button. Below this, there are buttons for 'BashOperator' and 'DummyOperator'. A modal window titled 'Task Instance: run_after_loop at: 2020-10-01T00:00:00+00:00' is open in the foreground. This modal contains several tabs: 'Task Instance Details', 'Rendered', 'Task Instances', 'View Log', and 'Filter Upstream'. The 'Task Actions' section within the modal provides various controls for the task instance, including buttons for 'Ignore All Deps', 'Ignore Task State', 'Ignore Task Dps', 'Run', 'Clear', 'Mark Failed', and 'Mark Success'. Each of these action buttons is preceded by a set of filter buttons: 'Past', 'Future', 'Upstream', 'Downstream', and 'Recursive' (for 'Clear'). A 'Close' button is located at the bottom right of the modal.

Task Instance: run_after_loop
at: 2020-10-01T00:00:00+00:00

Task Instance Details Rendered Task Instances View Log Filter Upstream

Task Actions

Ignore All Deps Ignore Task State Ignore Task Dps Run

Past Future Upstream Downstream Recursive Failed Clear

Past Future Upstream Downstream Mark Failed

Past Future Upstream Downstream Mark Success

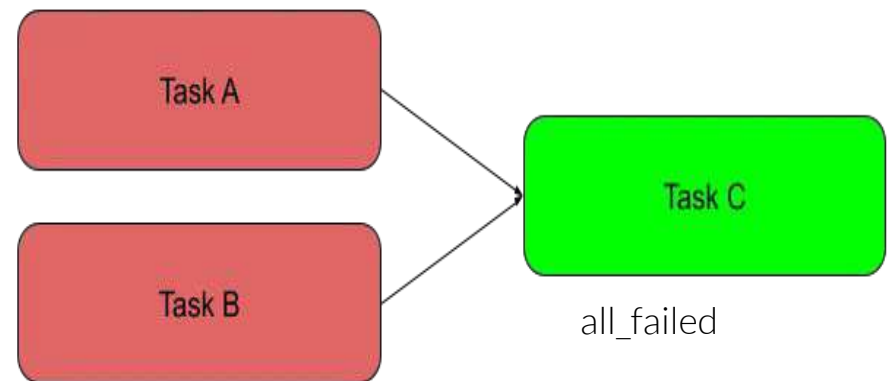
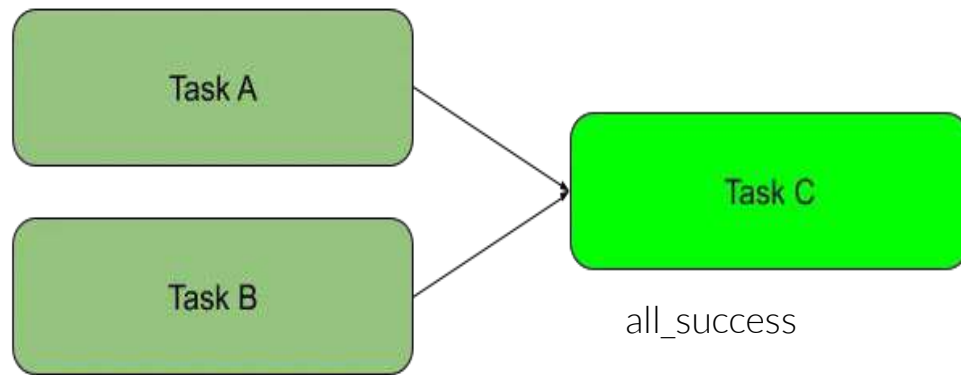
Close



Trigger rules

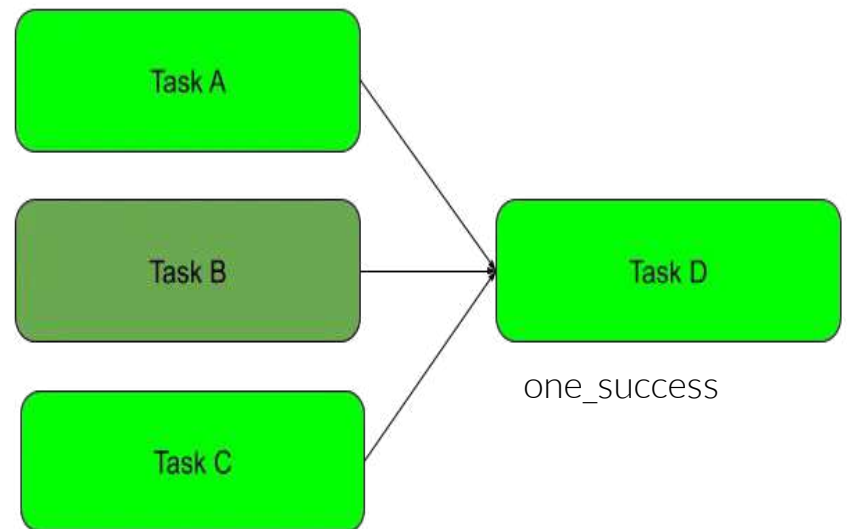
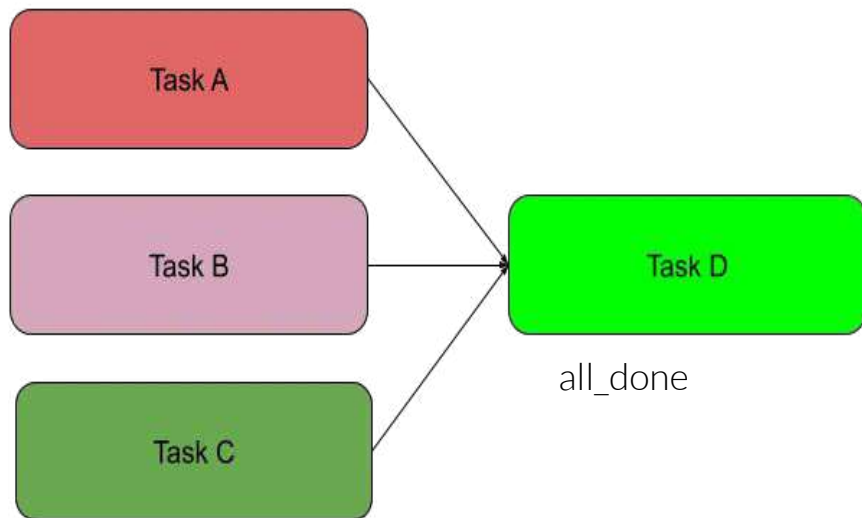


Trigger rules



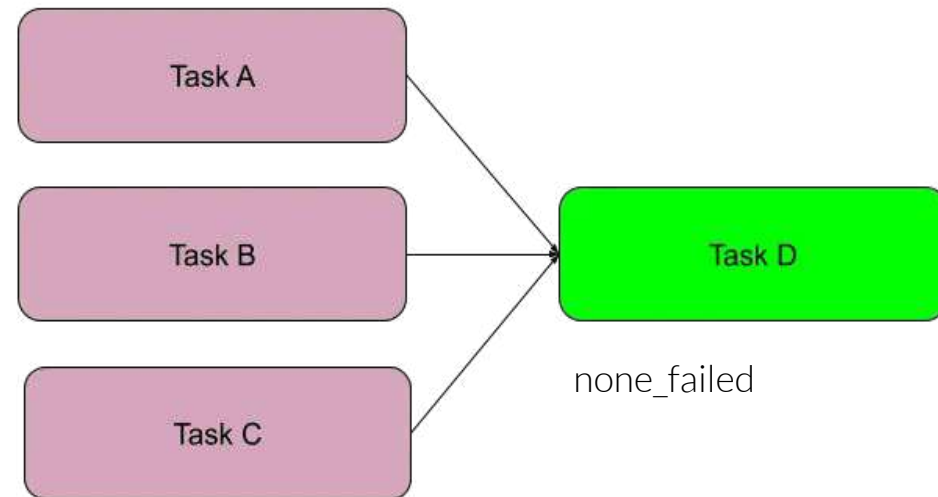
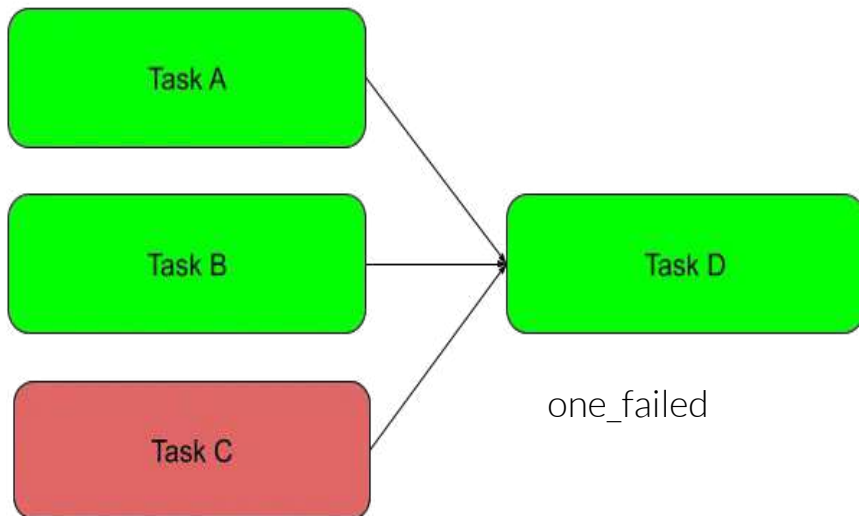


Trigger rules





Trigger rules





Resumen



Resumen

1. Podemos orquestar usando la sintaxis de Airflow o de Cron.
2. Podemos monitorear nuestros procesos.
3. Podemos realizar backfills.
4. Los tasks pueden pasar por varios estados.



Sensores

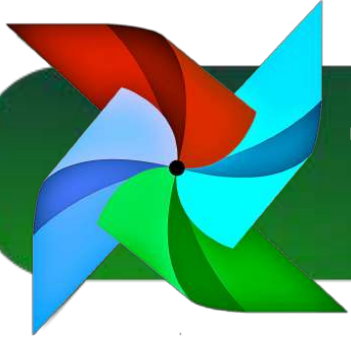


¿Qué son los sensores?

Los sensores son un tipo especial de operadores que están diseñados para hacer exactamente una cosa: **esperar a que ocurra algo.**



<https://airflow.apache.org/docs/apache-airflow/stable/concepts/sensors.html>



Tipos de sensores

- ExternalTaskSensor
- FileSensor
- HttpSensor
- S3KeySensor
- SqlSensor

https://airflow.apache.org/docs/apache-airflow/stable/_api/airflow/sensors/index.html?highlight=sensors#module-airflow.sensors

Templates con Jinja



¿Qué son los templates con Jinja?

- Nos permiten sustituir información durante la ejecución.
- Nos dan flexibilidad cuando definimos tareas.
- Son creados usando el “templating language” **Jinja**.

<https://airflow.apache.org/docs/apache-airflow/stable/templates-ref.html>

<https://airflow.apache.org/docs/apache-airflow/stable/tutorial.html#templating-with-jinja>



XComs



¿Qué son los XComs?

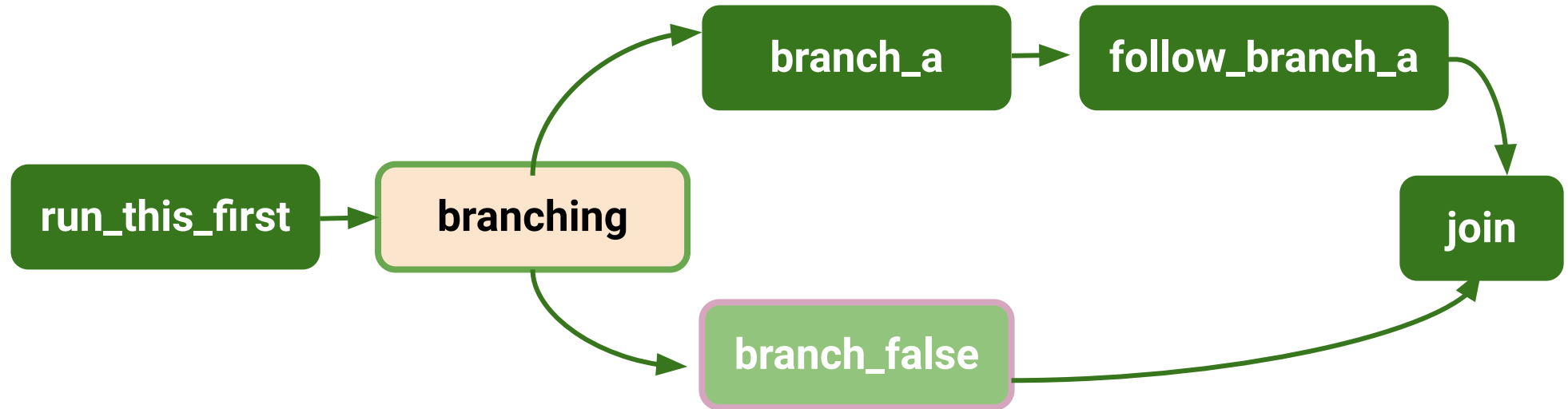
Los XComs (abreviatura de "cross-communications") son un mecanismo que permite que las tasks se comuniquen entre sí, ya que por defecto estas están totalmente aisladas y pueden estar ejecutándose en máquinas totalmente diferentes.

<https://airflow.apache.org/docs/apache-airflow/stable/concepts/xcoms.html>

BranchPythonOperator



BranchPythonOperator





Proyecto

Platzi explora el espacio

Primera **EdTech** de la historia en lanzar un satélite al espacio incursionando al mismo tiempo en la exploración espacial.

 Platzi
lanzará un satélite
al **Espacio**



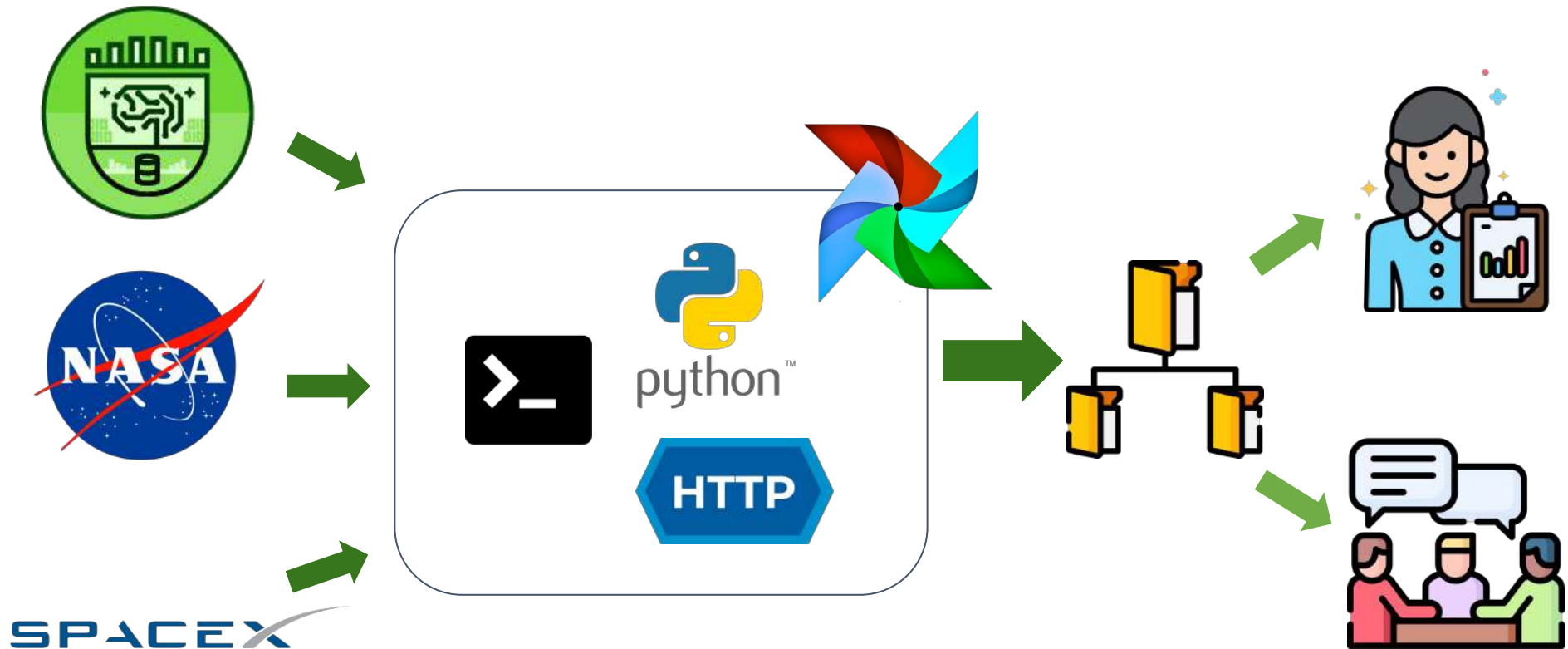


Objetivos

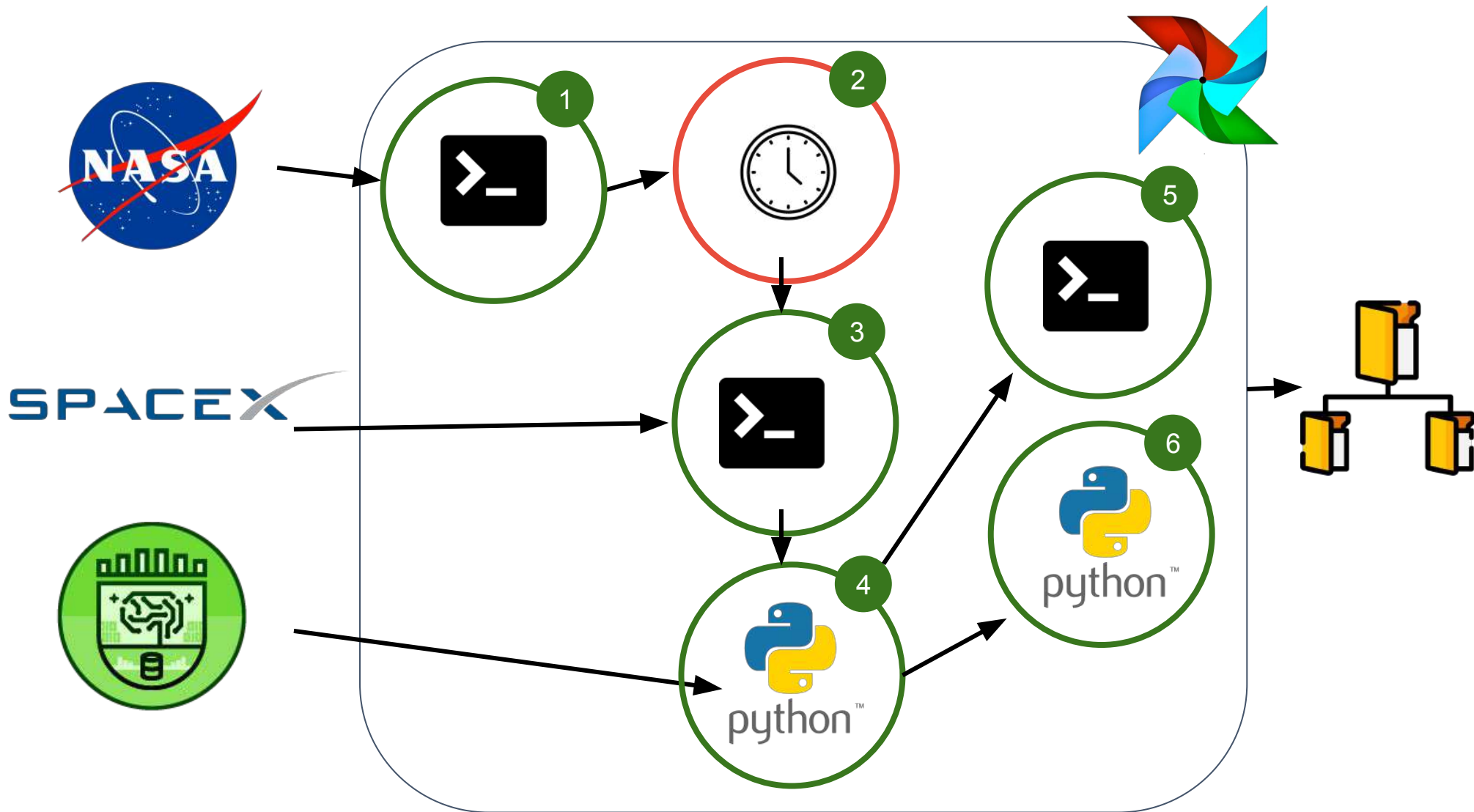
Los equipos de analistas y marketing de Platzi necesitan datos de los estudiantes que han accedido al satélite e información del historial de eventos de SpaceX, por lo tanto necesitamos que nos ayudes a ejecutar las siguientes tareas.

1. Esperar a que la NASA nos dé autorización para acceder a los datos del satélite.
2. Recolectar datos del satélite y dejarlos en un fichero.
3. Recolectar datos de la API de SpaceX y dejarlos en un fichero.
4. Enviar un mensaje a los equipos de que los datos finales están disponibles.

Ejemplo: workflow



Ejemplo: grafo



Cierre del curso

The background is a solid dark green. On the right side, there is a large, abstract, light green geometric shape that resembles a stylized pinwheel or a flower with multiple petals. A small, dark green circular dot is located near the center of this shape, slightly to the right of the main text.



Cierre del curso

1. ¿Qué es Airflow? ¿Para qué y por qué utilizarlo?
2. Instalación y configuraciones.
3. Creación de DAGs.
4. Utilización de operadores.
5. Orquestación y monitorización.
6. Sensores, templates, Xcoms y BranchPythonOperator.