
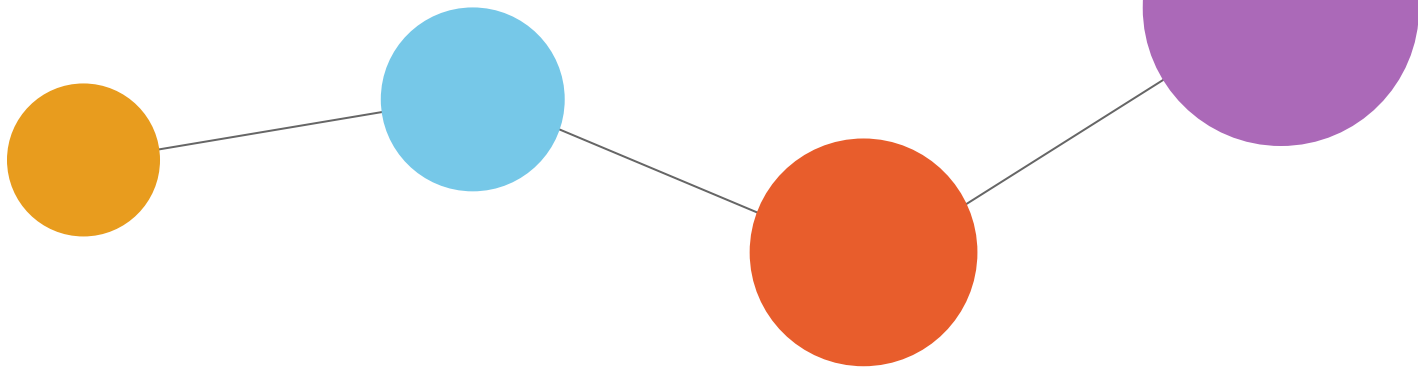


# Orquestación de datos con Apache Airflow

Fiorella Piriz Sapio   
Solution Architect en 



# Itinerario

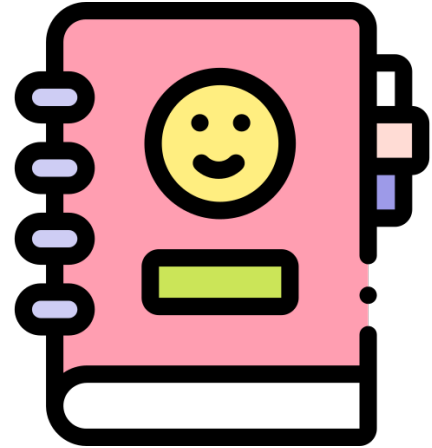
Introducción a la orquestación de datos

Introducción a Apache Airflow

Configuración de DAGs

Caso Práctico

Conclusión



01



# ¿Qué es la orquestación de datos?

# Orquestación de datos

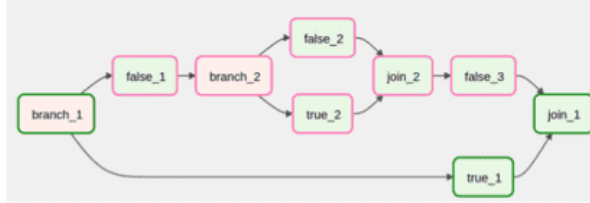


02



# ¿Qué es Apache Airflow?

# Apache Airflow

A screenshot of the Apache Airflow web interface. The top navigation bar includes links for DAGs, Datasets, Security, Browse, Admin, and Docs. The main section is titled "DAGs" and shows a list of DAGs with columns for DAG name, Owner, Runs, Schedule, Last Run, and Next Run. The list includes "example\_nested\_branch\_dag", "space\_life\_sciences\_pipeline", and "tutorial\_taskflow\_api".

DAG	Owner	Runs	Schedule	Last Run	Next Run
example_nested_branch_dag	airflow	1	interval	2023-03-31, 00:00:00	2023-04-01, 00:00:00
space_life_sciences_pipeline	airflow	1	interval	2023-01-30, 00:00:00	
tutorial_taskflow_api	airflow	1	interval	2023-02-09, 08:42:46	

A screenshot of the Apache Airflow web interface showing a list of XComs. The table has columns for Key, Value, Timestamp, Dag ID, and Task ID. The data shows multiple entries for "alphanumeric\_key" and "return\_value" across different DAGs and tasks.

Key	Value	Timestamp	Dag ID	Task ID
alphanumeric_key	["false_2"]	2023-04-01, 11:58:32	example_nested_branch_dag	branch_2
return_value	true_1	2023-04-01, 11:58:32	example_nested_branch_dag	branch_2
alphanumeric_key	["true_2"]	2023-04-01, 11:58:32	example_nested_branch_dag	branch_2
return_value	true_1	2023-04-01, 11:58:32	example_nested_branch_dag	branch_2
alphanumeric_key	["false_2"]	2023-04-01, 11:58:32	example_nested_branch_dag	branch_2
return_value	true_1	2023-04-01, 11:58:32	example_nested_branch_dag	branch_2
alphanumeric_key	["false_2"]	2023-04-01, 11:58:32	example_nested_branch_dag	branch_2
return_value	true_1	2023-04-01, 11:58:32	example_nested_branch_dag	branch_2

Opensource

Fácil de usar

Basada en Python

Múltiples conectores

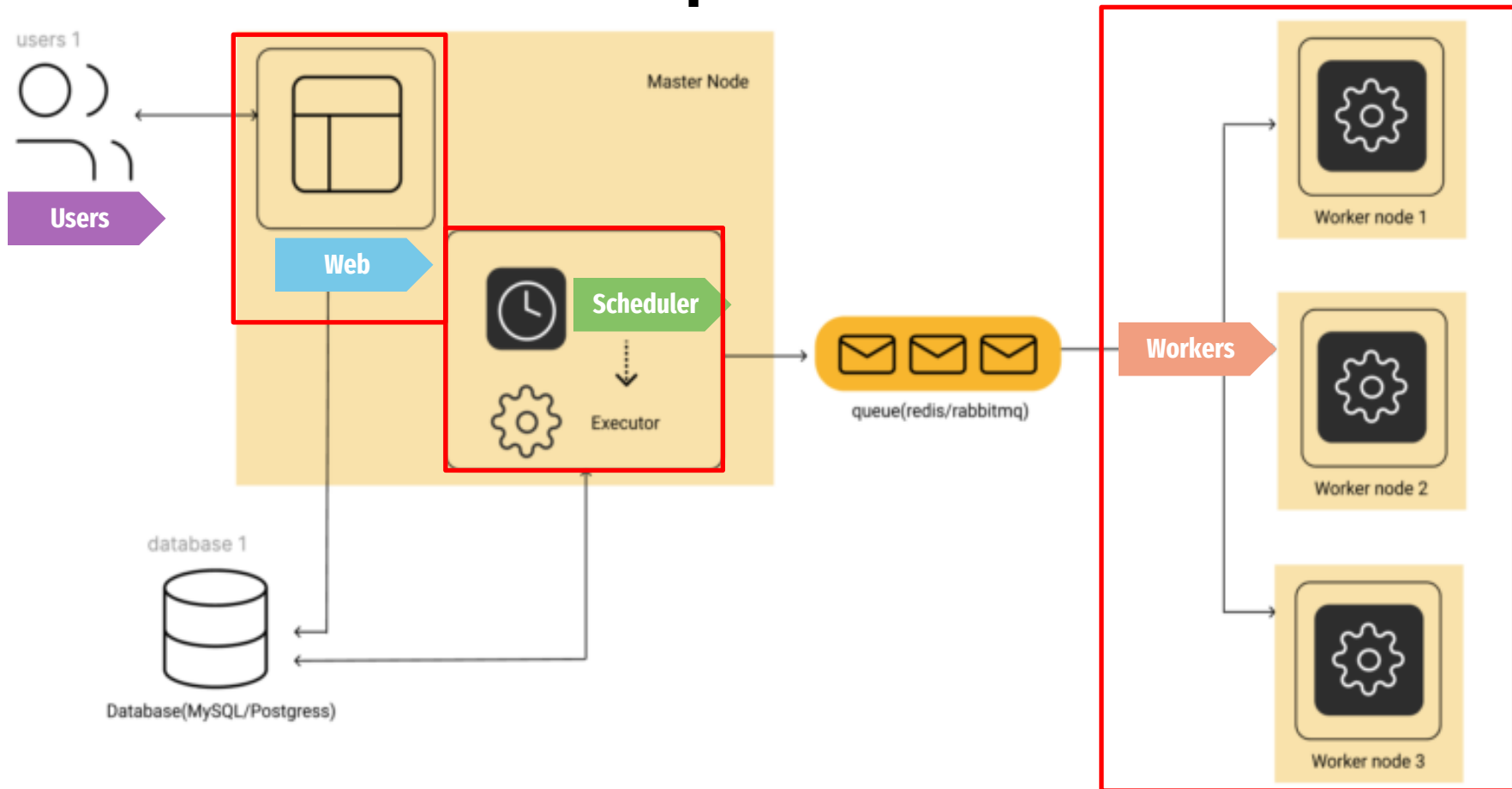
VS

Oozie

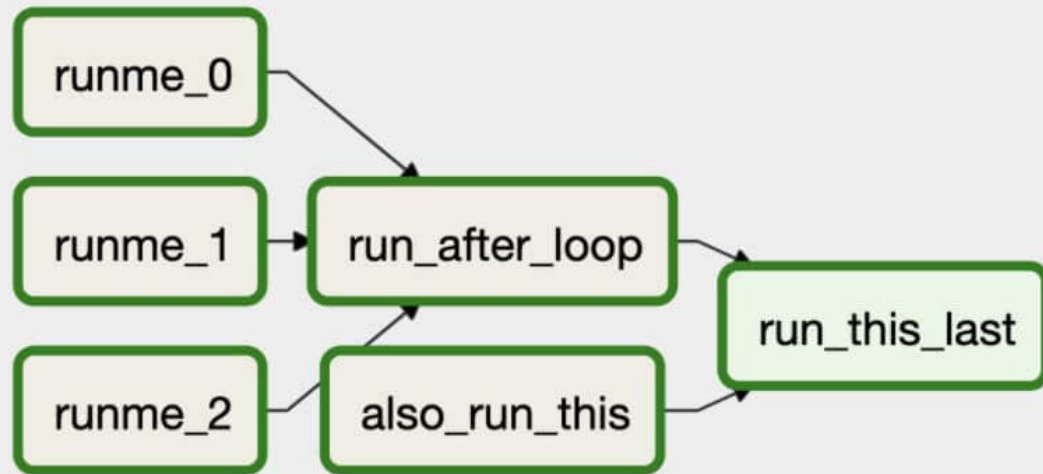
Azkaban

Luigi

# Componentes



# DAGS











03



# Instalación y preparación del entorno

# Instalación de Airflow en Linux

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

Pasos		
	01 Instalar Apache Airflow	<code>pip install "apache-airflow[gcp]"</code>
	02 Configurar variable de entorno	<code>export AIRFLOW_HOME=/home/airflow</code>
	03 Configuración	<code>nano /root/airflow/airflow.cfg</code>
	04 Comprobar la versión	<code>airflow version</code>
	05 Inicializar la base de datos	<code>airflow db init</code>
	06 Crear usuario	<code>airflow users create</code>
	07 Arrancar webserver	<code>airflow webserver</code>
	08 Arrancar scheduler	<code>airflow scheduler</code>

Downloads — root@ip-172-31-6-15:/home/ec2-user — ssh -i demo-airflow.pem ec2-...  
(airflow\_env) [root@ip-172-31-6-15 ec2-user]#

15.237.255.120  
15.237.255.120:9090/home  
Relaunch to update



## This site can't be reached

15.237.255.120 refused to connect.

Try:

- Checking the connection
- [Checking the proxy and the firewall](#)

ERR\_CONNECTION\_REFUSED

Details

Reload



04



# Configuración de un DAG

```

1  from airflow import DAG
2  from airflow.decorators import task
3  from airflow.operators.dummy import DummyOperator
4  from airflow.operators.bash import BashOperator
5  from datetime import datetime
6
7  ∨ default_args = {
8      'owner': 'ADMIN', # Usuario al que pertenece el DAG
9      'start_date': datetime(2022, 12, 1) # date.today(); days_ago(6) # Fecha de comienzo de la programación
10     #'email': ['airflow@example.com'],
11     #'email_on_failure': False,
12     #'email_on_retry': False,
13     #'retries': 1,
14     #'sla': timedelta(hours=3) # Tiempo máximo de duración del dag
15     #'end_date': datetime(2016, 1, 1),
16 }
17
18 ∨ dag_args = {
19     'dag_id': '01-check-file-dag', # Identificador único
20     'schedule_interval': '@daily', # 0 * * * *; None
21     'catchup': False, # Si se pone al día con la ejecuciones o no
22     'default_args': default_args,
23     "doc_md":(
24         """
25         # 01-check_file_dag
26         """,
27 )

```

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

Args: <https://airflow.apache.org/docs/apache-airflow/1.10.10/tutorial.html>

```

29  ✓ with DAG(**dag_args) as dag:
30
31      # Definir una función de tarea usando la TaskFlow API
32      @task
33      ✓ def taskflow_function():
34          | print("Este es un ejemplo de TaskFlow API")
35          | return "TaskFlow completado"
36
37      # DummyOperator para iniciar el flujo
38      ✓ start = DummyOperator(
39          | task_id='start'
40          )
41
42      # BashOperator para ejecutar un comando bash
43      ✓ bash_task = BashOperator(
44          | task_id='run_bash_command',
45          | bash_command='echo "Hello, Airflow from Bash!"'
46          )
47
48      # Definir dependencias
49      start >> taskflow_function() >> bash_task
50

```

Task dependencies: <https://www.astronomer.io/docs/learn/managing-dependencies>



## DAG: 01-first-dag

Schedule: @daily

Next Run ID: 2024-09-14, 00:00:00 UTC



DAG Docs

## 01-check\_file\_dag

15/09/2024 17:56:20

All Run Types

All Run States

Clear Filters

Auto-refresh

25

Press **shift** + **/** for Shortcuts

deferred

failed

queued

removed

restarting

running

scheduled

shutdown

skipped

success

up\_for\_reschedule

up\_for\_retry

upstream\_failed

no\_status



DAG

## 01-first-dag

Details

Graph

Gantt

&lt;&gt; Code

Event Log

Run Duration

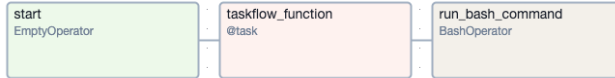
Task Duration

Calendar

start

taskflow\_function

run\_bash\_command



Layout:

Left -&gt; Right

05



## Caso práctico



# Caso Práctico Juegos Olímpicos

ETL

01

**get\_data**

API: <https://apis.codante.io/olympic-games/events>


**Generar diccionario  
con los datos de la  
API**

- PythonOperator
- Variables
- XcomPull



```

1  # Function to get data from the URL
2  def request_url():
3      url = Variable.get("url") # Use Airflow Variable for URL
4      response = requests.get(url)
5      if response.status_code == 200:
6          return response.json()
7      else:
8          raise ValueError(f"Failed to retrieve data. Status code: {response.status_code}")
9
10
11 # Task 1: Get Data from the URL
12 get_data = PythonOperator(
13     task_id='get_data',
14     python_callable=request_url,
15     provide_context=False
16 )
17

```


Airflow

[DAGs](#)
[Cluster Activity](#)
[Datasets](#)
[Security](#)
[Browse](#)
[Admin](#)
[Docs](#)











17:16 UTC


No file chosen
 ☒ Overwrite if exists
 ☐ Fail if exists
 ☐ Skip if exists
 

Variables
 Configurations
 Connections
 Plugins
 Providers
 Pools
 XComs

List Variable

Record Count: 3

<input type="checkbox"/>	Key ↑	Val ↑	Description ↑	Is Encrypted ↑
<input type="checkbox"/>	   local_file_path	/home/ec2-user/fast_api/...		False
<input type="checkbox"/>	   s3_bucket_name	airflow-bucket-s3-fps		False
<input type="checkbox"/>	   url	https://apis.codante.io/ol...		False

# Caso Práctico Juegos Olímpicos

ETL

01

**get\_data**

**Generar  
diccionario con los  
dato de la API**

- PythonOperator
- Variables
- XcomPush

02

**create\_json**

**Transformar datos**

- XcomPull

```

1  # Function to create JSON from the data
2  def create_json_olympic_games(**kwargs):
3      data = kwargs['ti'].xcom_pull(task_ids='get_data')
4      total_pages = data['meta']['last_page']
5      results_dict = {}
6
7      for page in range(1, total_pages + 1):
8          new_url = Variable.get("url").replace("{page_id}", str(page))
9          page_data = requests.get(new_url).json()
10         events_list = page_data['data']
11
12         for event in events_list:
13             ....
14             ....
15             ....
16
17 # Task 2: Create JSON from the Data
18 create_json = PythonOperator(
19     task_id='create_json',
20     python_callable=create_json_olympic_games,
21     provide_context=True
22 )

```



[DAGs](#)
[Cluster Activity](#)
[Datasets](#)
[Security](#)
[Browse](#)
[Admin](#)
[Docs](#)

## DAG: 03-create-json-olimpic-games

16/09/2024 17:27:30

All Run Types

All Run States

Clear Filters

Press **shift** + **/** for Shortcuts

[deferred](#)
[failed](#)
[queued](#)
[removed](#)
[restarting](#)
[running](#)
[scheduled](#)
[shutdown](#)
[skipped](#)
[success](#)

Duration

00:06:37

00:03:18

00:00:00

[get\\_data](#)  
[create\\_json](#)  
[write\\_json\\_to\\_s3](#)  
[write\\_json\\_to\\_local](#)

DAG

03-create-json-olimpic-games /

Run

2024-09-16, 17:19:38 UTC /

Task

create\_json

Details

Graph

Gantt

Code

Event Log

Logs

XCom

Task Duration

Key

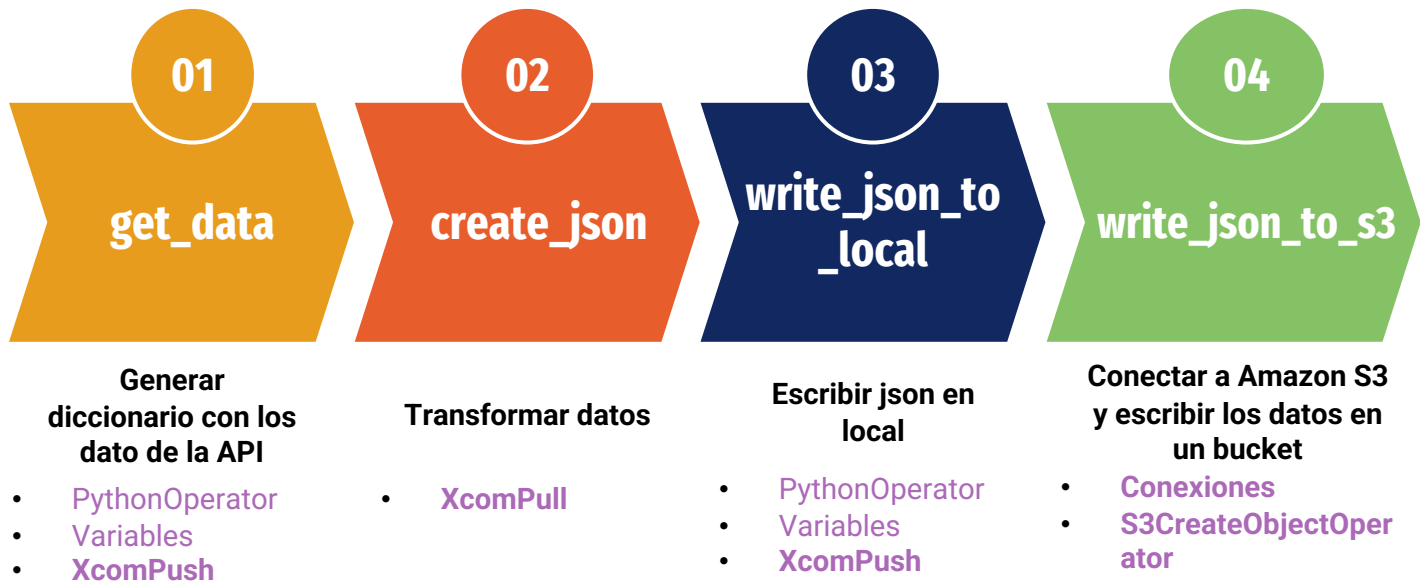
Value

6636 items

[▶ \[ 0 - 100 \]](#)  
[▶ \[ 100 - 200 \]](#)  
[▶ \[ 200 - 300 \]](#)  
[▶ \[ 300 - 400 \]](#)  
[▶ \[ 400 - 500 \]](#)

# Caso Práctico Juegos Olímpicos

ETL





## List Connection

Search



Actions



- Variables
- Configurations
- Connections
- Plugins
- Providers
- Pools
- XComs

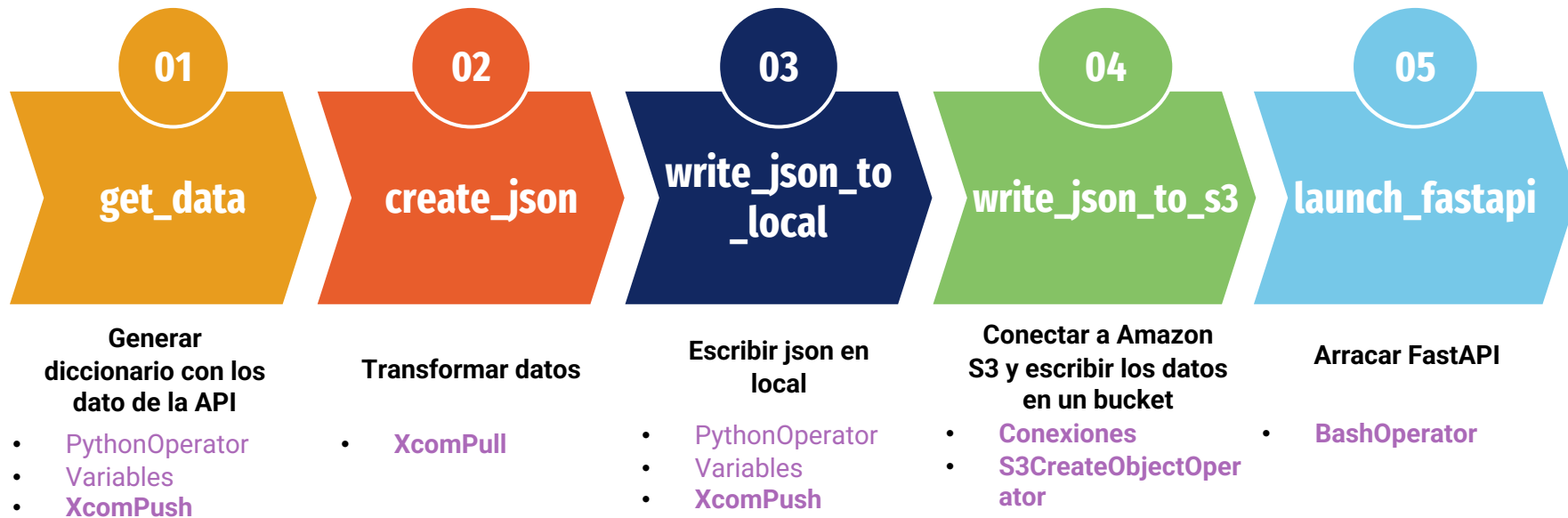
Record Count: 61

<input type="checkbox"/>	Conn Id	Conn Type	Description	Host	Port	Is Encrypted	Is Extra Encrypted
<input type="checkbox"/>	airflow_db	mysql		mysql		False	False
<input type="checkbox"/>	athena_default	athena				False	False
<input type="checkbox"/>	aws_default	aws				False	False
<input type="checkbox"/>	azure batch default	azure batch				False	False

```
1 # Task 3: Write the JSON to S3
2 write_json_to_s3 = S3CreateObjectOperator(
3     task_id="write_json_to_s3",
4     aws_conn_id='aws_default',
5     s3_bucket=Variable.get("s3_bucket_name"), # Use Airflow Variable for S3 bucket name
6     s3_key='data/olympic_results.json',
7     data="{{ ti.xcom_pull(task_ids='create_json') }}",
8     replace=True
9 )
```

# Caso Práctico Juegos Olímpicos

ETL



```

1 # Task: Launch FastAPI using uvicorn
2 launch_fastapi = BashOperator(
3     task_id='launch_fastapi',
4     bash_command='cd /home/ec2-user/fast_api && nohup uvicorn main:app --host 0.0.0.0 --port 8000 --reload &',
5     execution_timeout=None # Disable timeout to keep FastAPI running
6 )

```

## Juegos Olímpicos 2024

PARIS 2024



País:  Deporte:  Categoría:

### Filtered Events

Event Name	Venue	Competitors	Result
Men's Group B	Geoffroy-Guichard Stadium	Argentina-Marrocos	1-2
Men's Group B	Lyon Stadium	Argentina-Iraque	3-1
Men's Group B	Lyon Stadium	Ucrânia-Argentina	0-2
Men's Quarter-final	Bordeaux Stadium	-	-

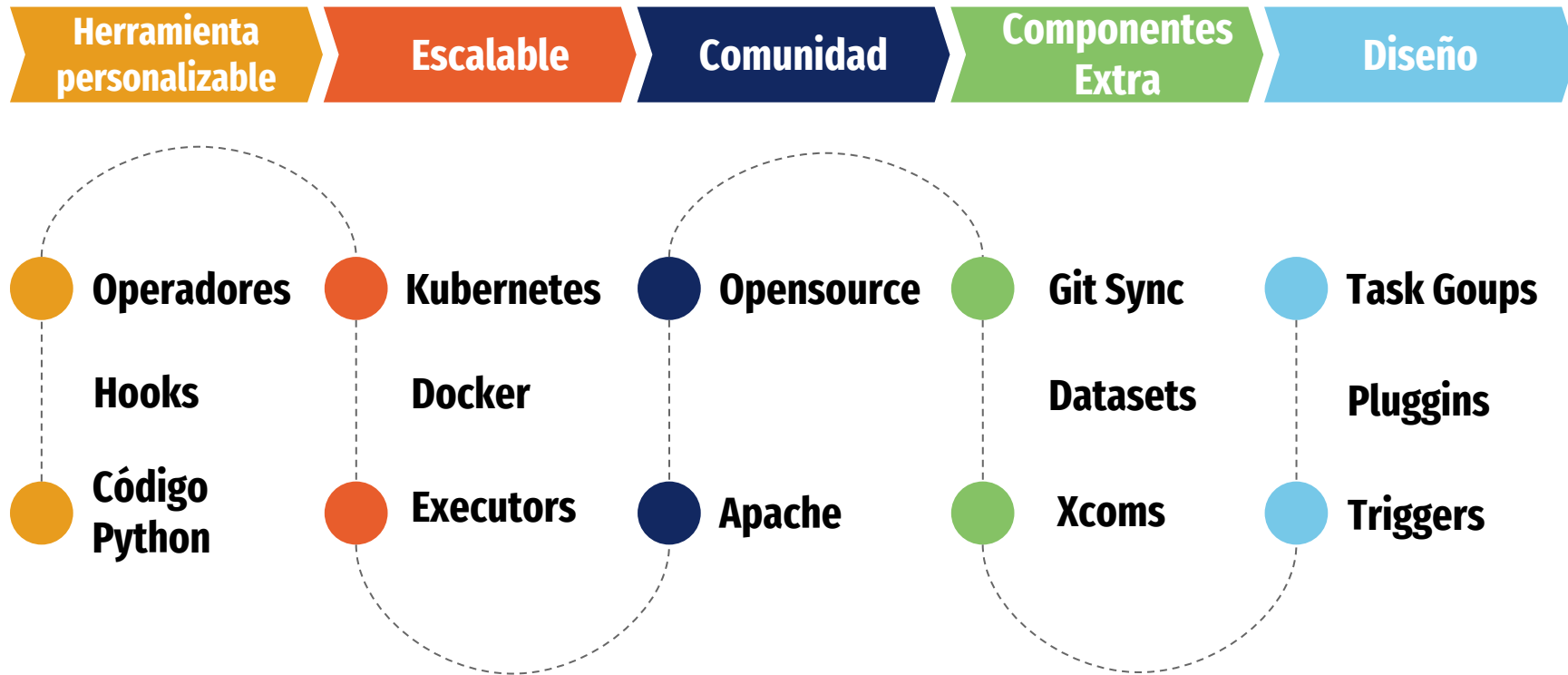


05



# Conclusión

# ¿Qué hemos aprendido?



# Q&A



**Fin** →

**Muchas gracias**

