PROTONX

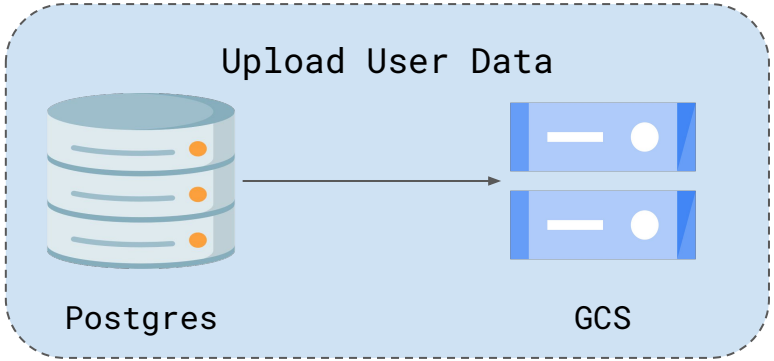
Apache Airflow để quản lý Quản lý Workflow

Workflow Orchestration
using
Apache Airflow

PROTON X

Task

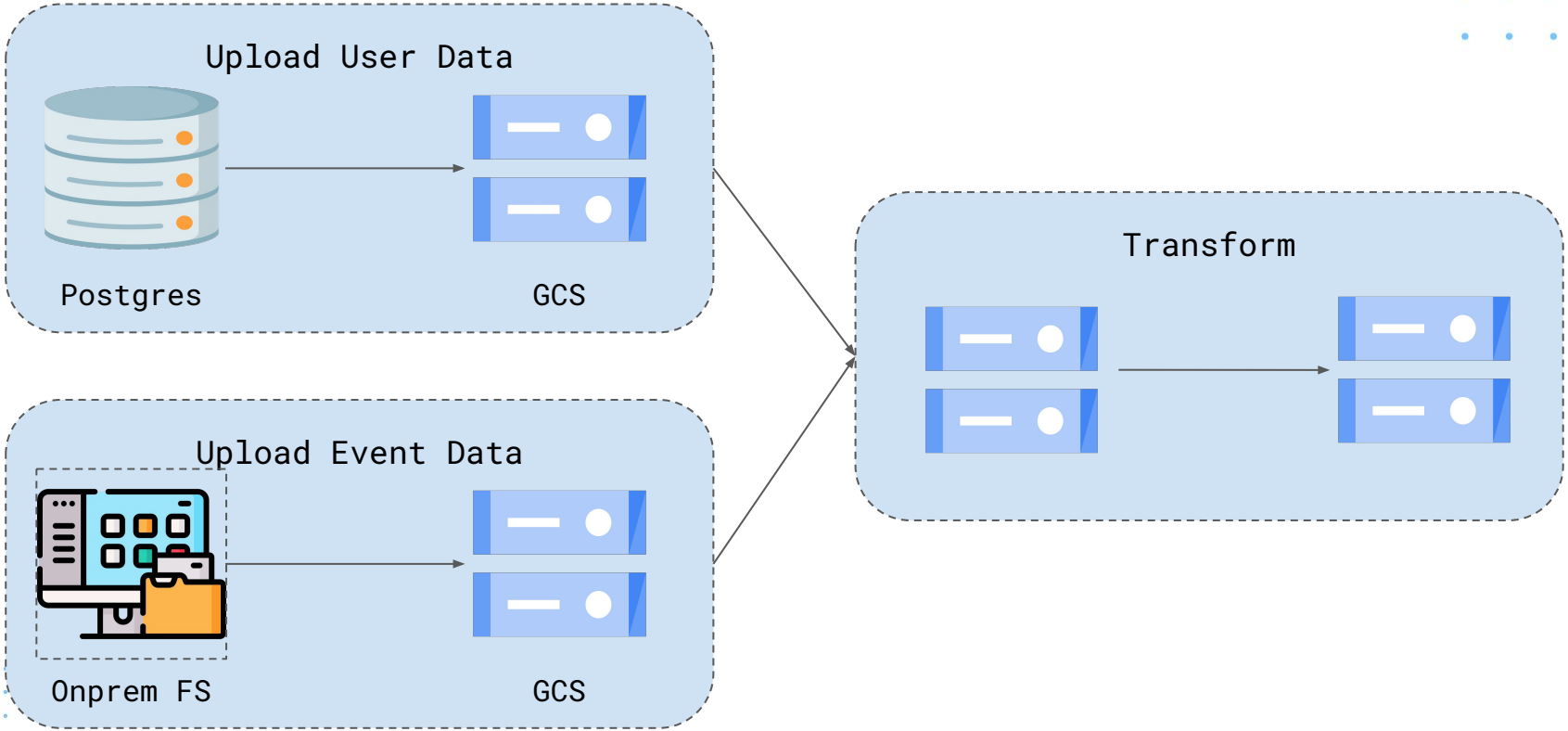
Task



PROTON X

Workflow

Workflow



PROTON X

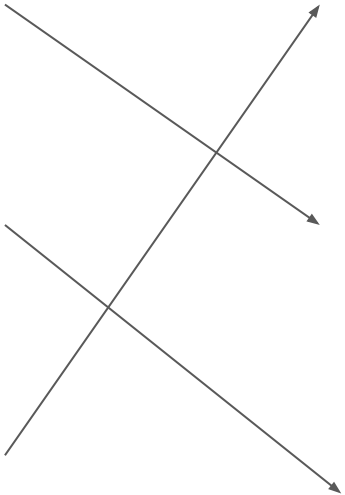
DAG

Directed Acyclic Graph

Directed

Acyclic

Graph



Đồ thị

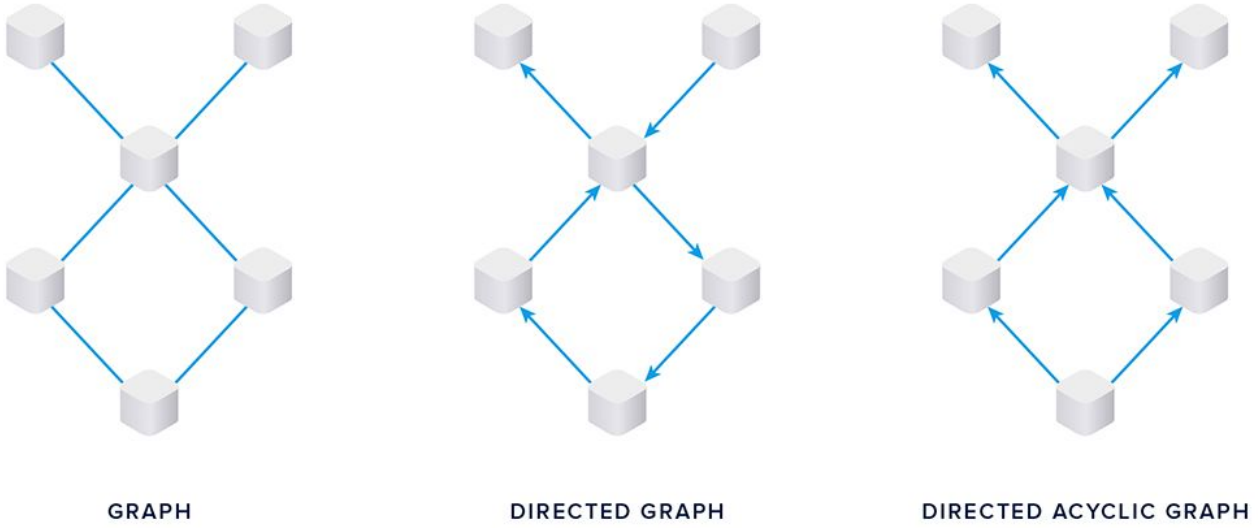
có hướng

không tuần hoàn

PROTON X

DAG

Directed Acyclic Graph

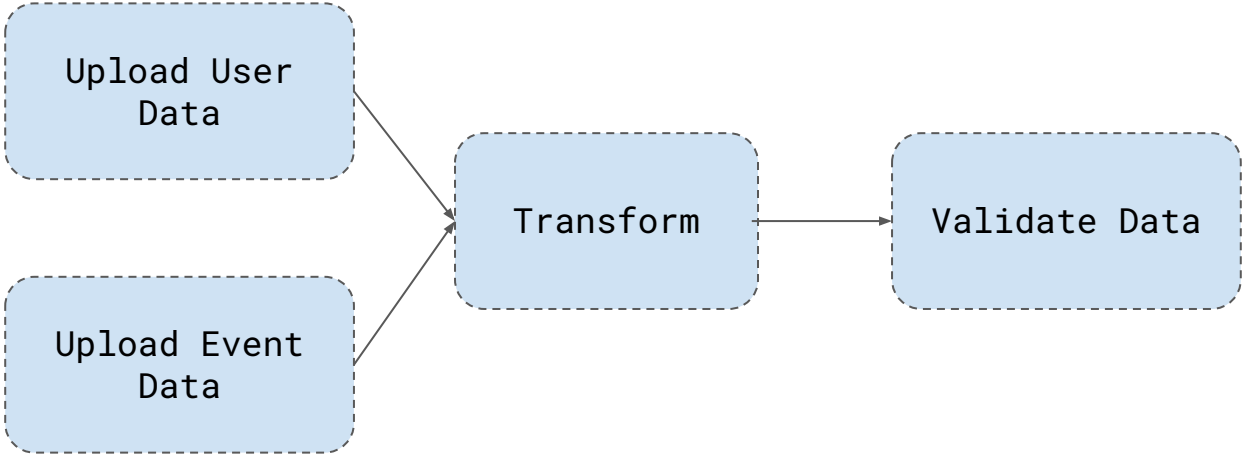


https://cdn.coin68.com/uploads/2022/10/DAG-2.jpg

PROTON X

DAG

Directed Acyclic Graph

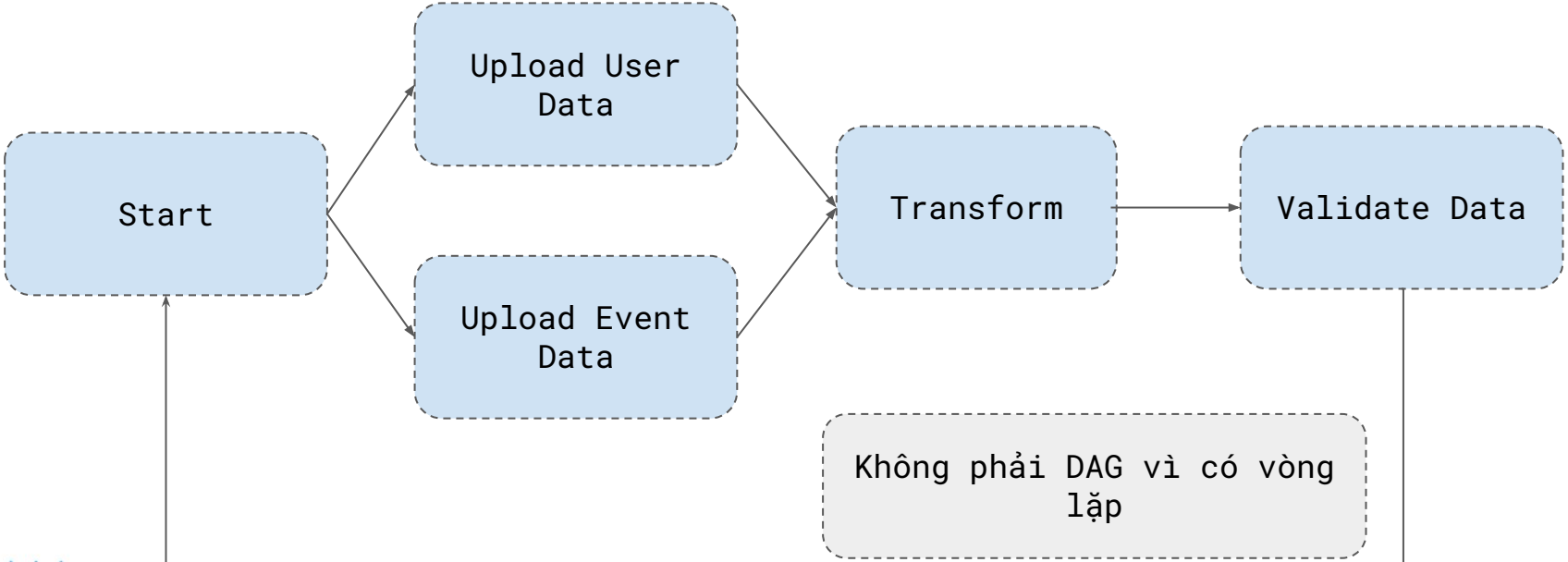


PROTON X

DAG

Directed Acyclic Graph

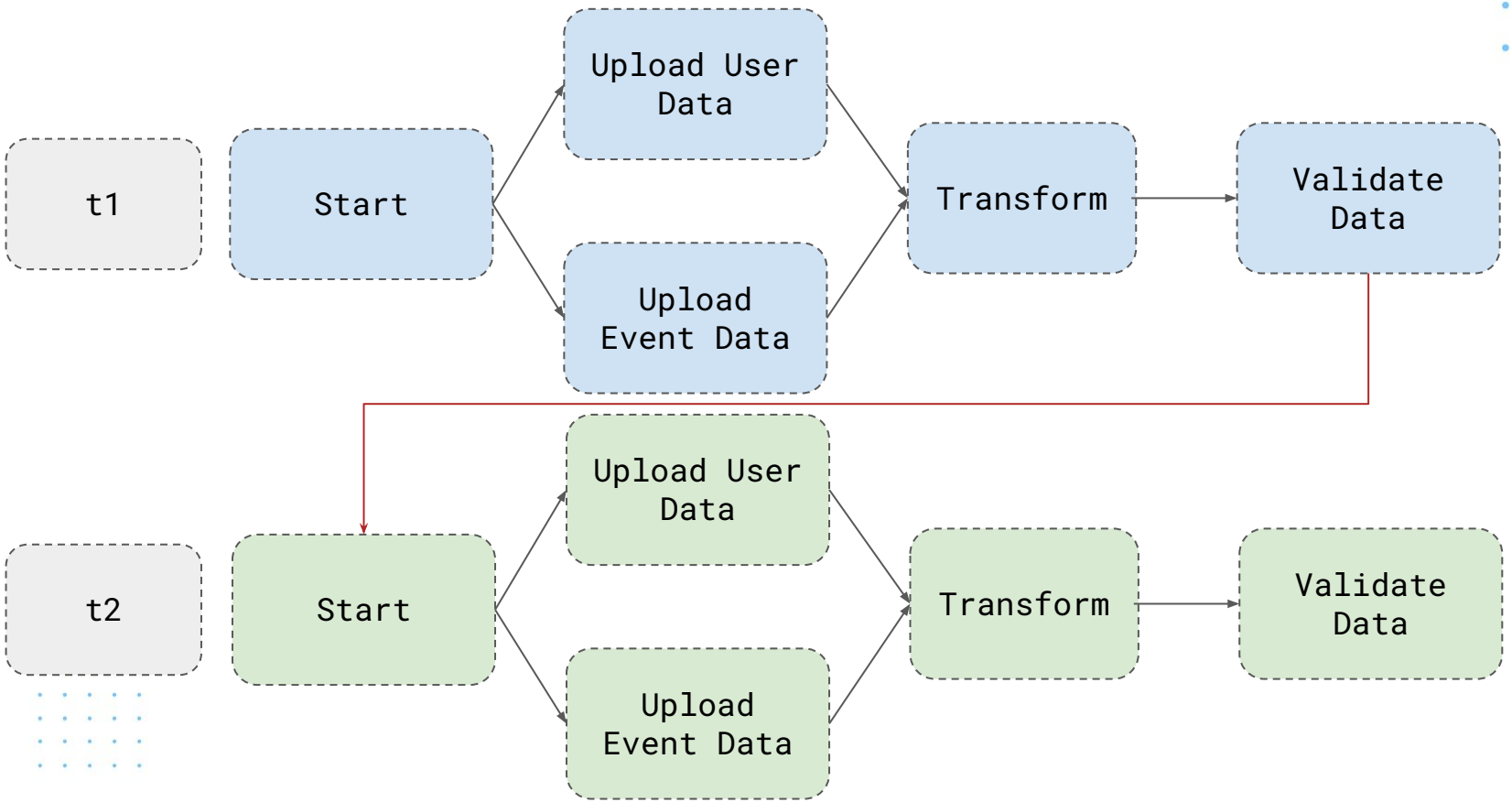
Cần chạy Workflow phụ thuộc vào ngày hôm trước ?



PROTON X

DAG

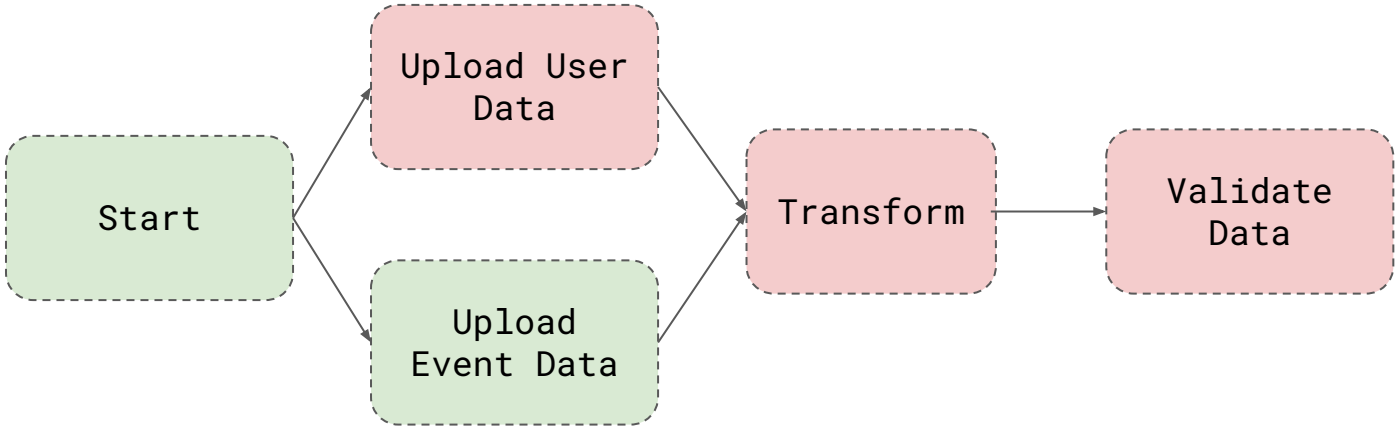
Directed Acyclic Graph



PROTON X

Quản lý

Orchestration



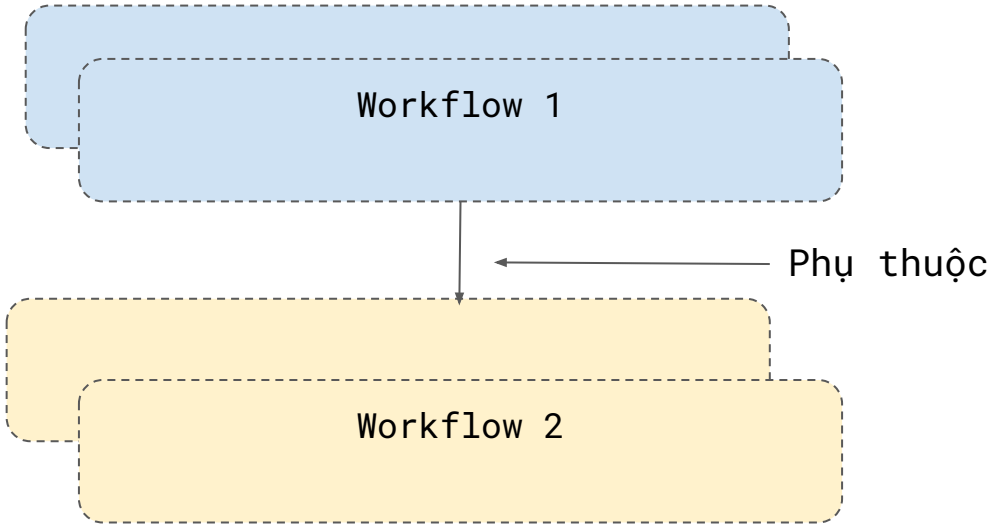
PROTON X

Quản lý

Orchestration

Chạy hằng ngày lúc 9:00

Chạy hằng tháng



PROTON X

Quản lý

Orchestration

9:00:00 2023-10-14

Workflow ngày
2023-10-14

Chạy lại
10:00:00 2023-10-15

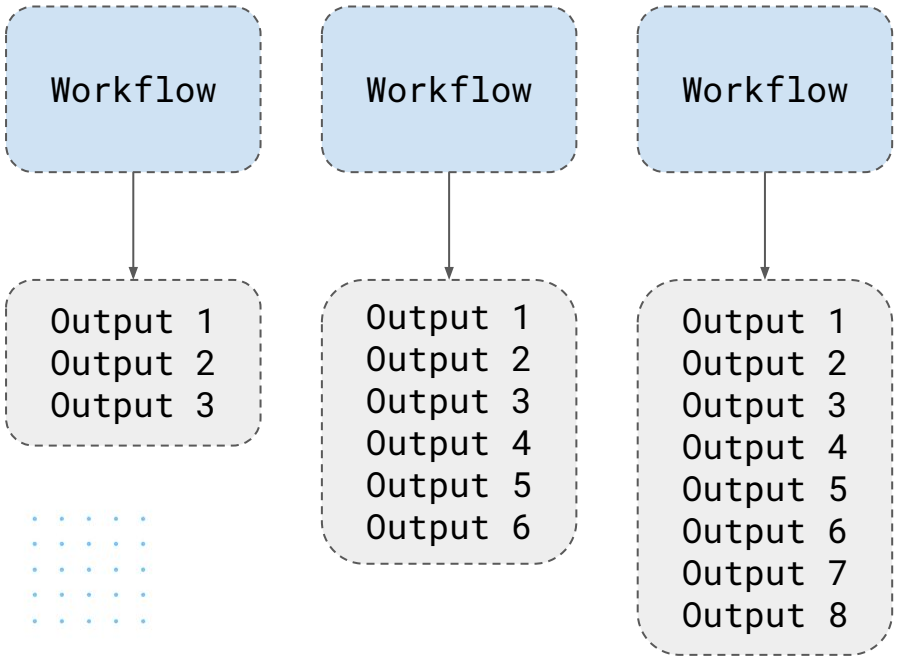
Workflow ngày
2023-10-14

PROTON X

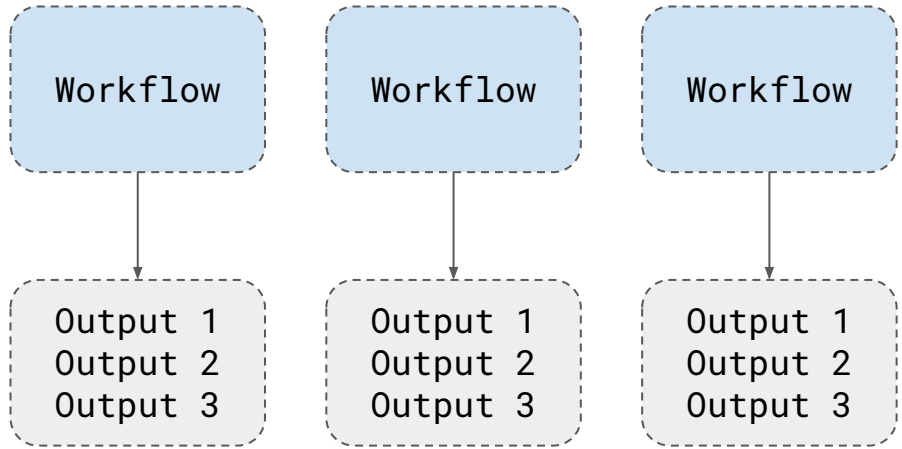
Idempotent

Idempotent

Non Idempotent



Idempotent



PROTON X

Idempotent

Idempotent

Workflow ngày 2023-10-14

Non Idempotent

2023-10-14	2023-10-15	2023-10-16
Lấy data ngày hôm nay	Lấy data ngày hôm nay	Lấy data ngày hôm nay
Output 1 Output 2 Output 3	Output 1 Output 2 Output 3 Output 4 Output 5 Output 6	Output 1 Output 2 Output 3 Output 4 Output 5 Output 6 Output 7 Output 8

Idempotent

2023-10-14	2023-10-15	2023-10-16
Lấy data ngày 2023-10-14	Lấy data ngày 2023-10-14	Lấy data ngày 2023-10-14
Output 1 Output 2 Output 3	Output 1 Output 2 Output 3	Output 1 Output 2 Output 3

Apache Airflow

Apache Airflow

PROTONX

PROTON X

Một số phần mềm

Applications

Apache Airflow

PREFECT

dagster

PROTON X

Cài đặt Airflow

Install Airflow

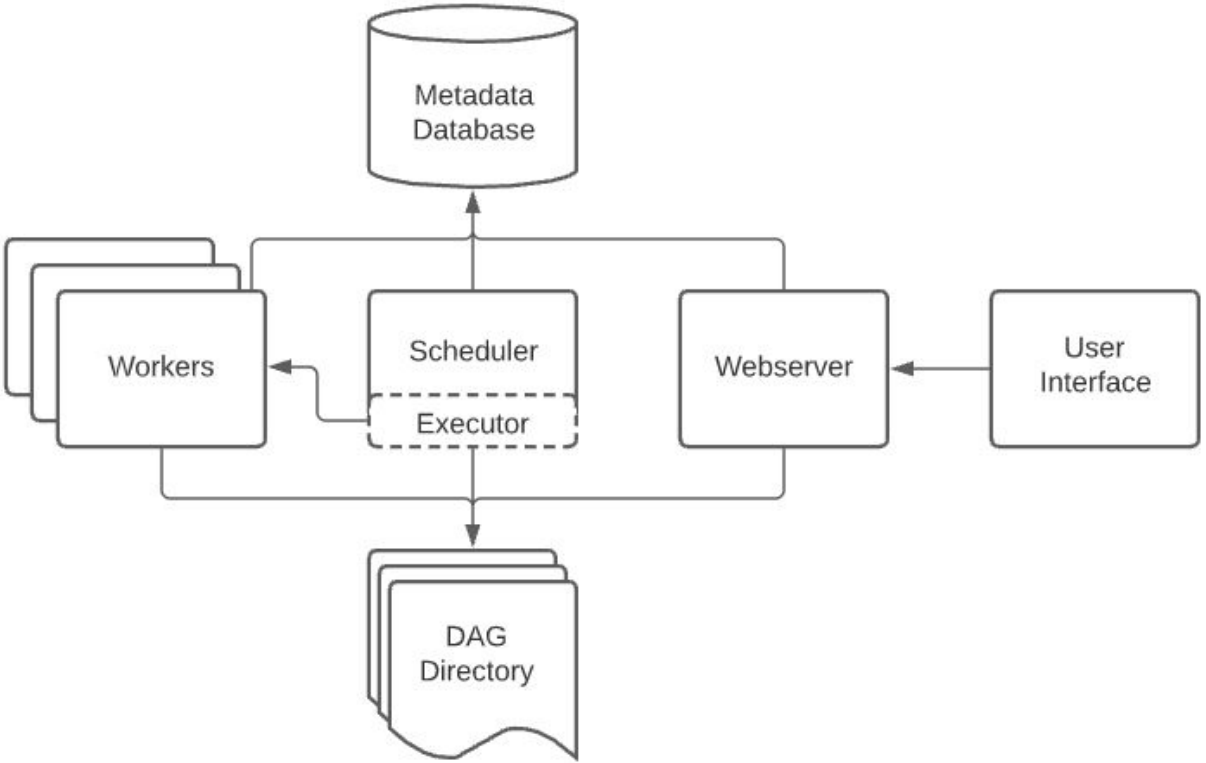
https://airflow.apache.org/docs/apache-airflow/stable/start.html

https://airflow.apache.org/docs/#providers-packages-docs-apache-airflow-providers-index-html

PROTON X

Kiến trúc Airflow

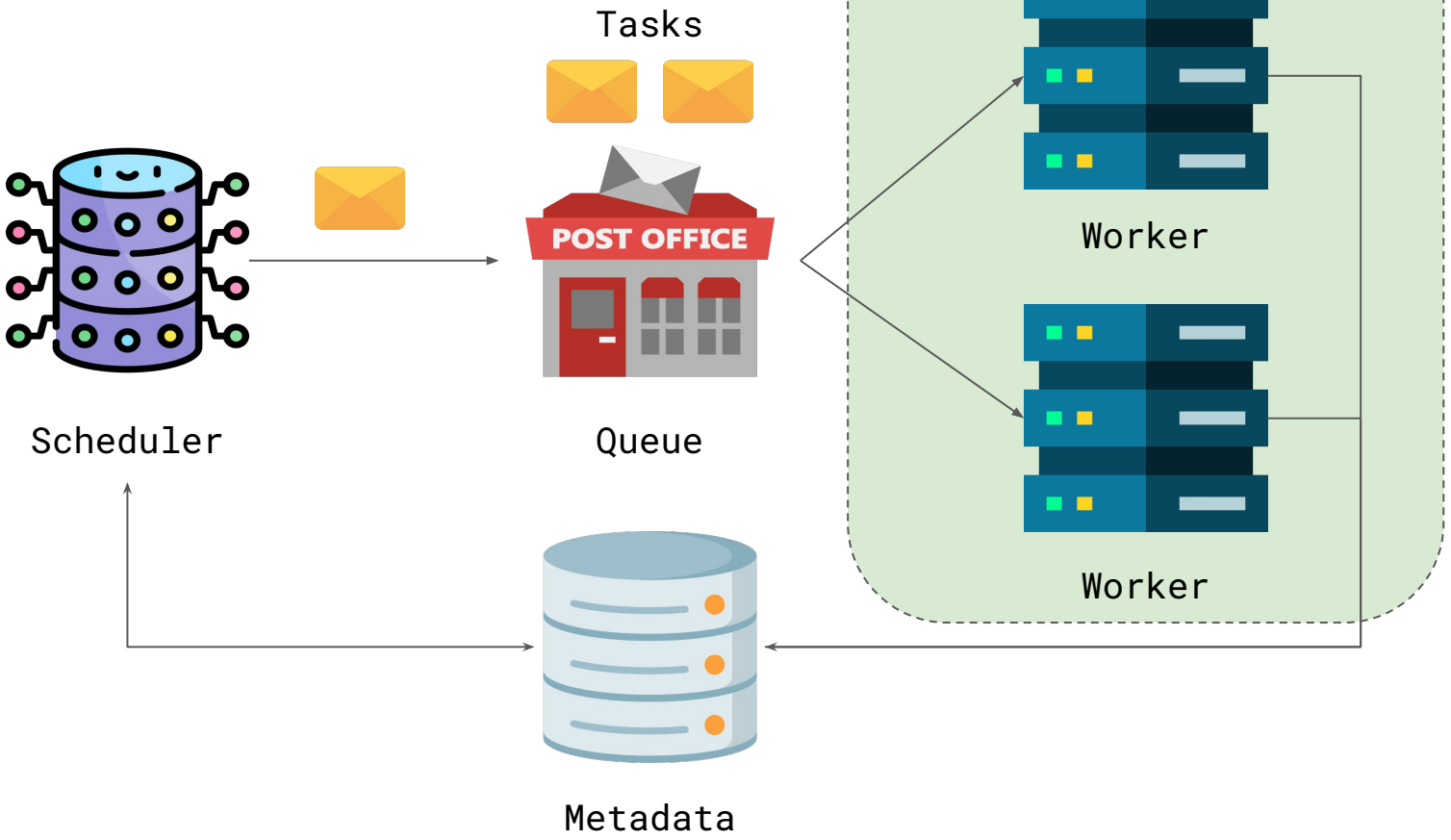
Airflow Architecture



PROTON X

TaskQueue

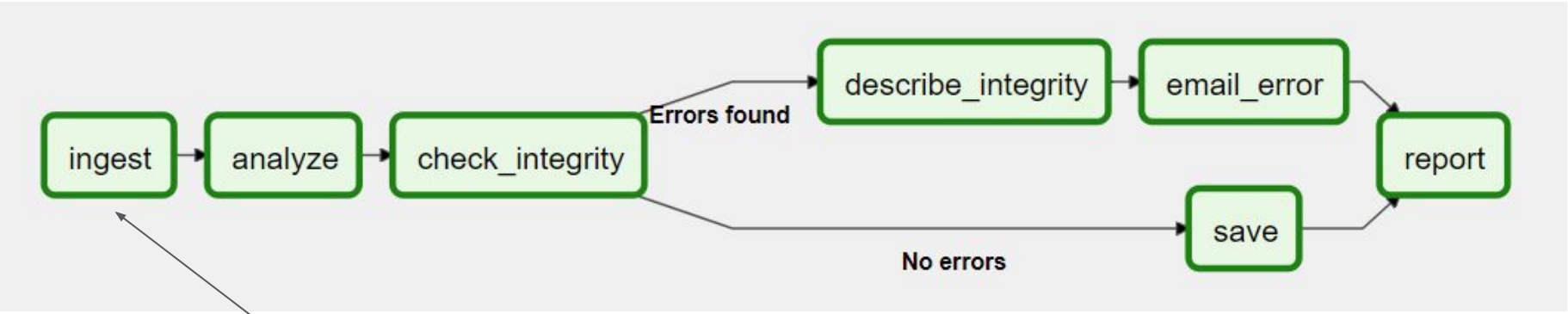
TaskQueue



PROTON X

Airflow DAG

Airflow DAG



Tasks

PROTON X

Airflow DAG

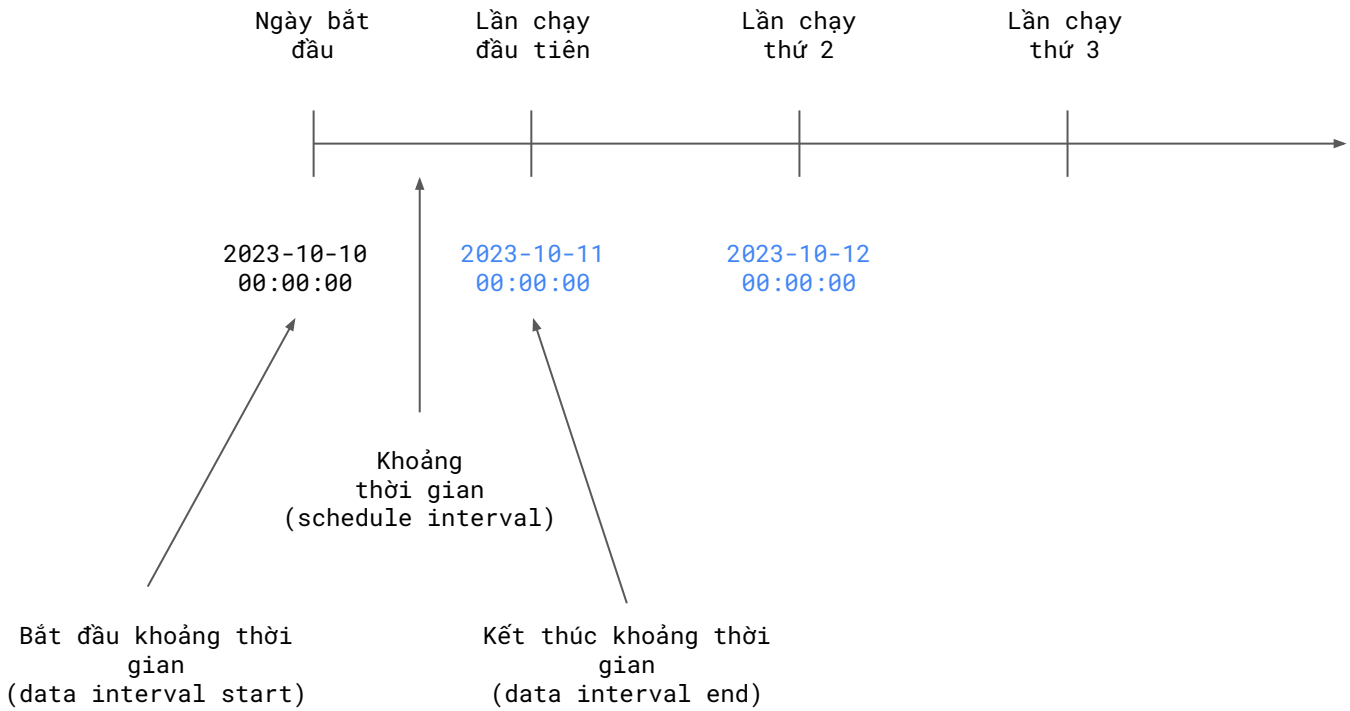
Airflow DAG

```python
import datetime

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

default_args_dict = {}

with DAG(
     dag_id="my_dag_name",
     start_date=datetime.datetime(2023, 10, 10),
     schedule="@daily",
     default_args={**default_args_dict}
):
     EmptyOperator(task_id="task")
```

PROTON X

Datetime trong Airflow

Datetime in Airflow



PROTON X

Datetime trong Airflow

Datetime in Airflow

Ngày chạy theo lịch	2023-10-11 00:00:00	2023-10-12 00:00:00
Ngày chạy logic (Logical Date)	2023-10-10 00:00:00	2023-10-11 00:00:00
Data Interval Start	2023-10-10 00:00:00	2023-10-11 00:00:00
Data Interval End	2023-10-11 00:00:00	2023-10-12 00:00:00

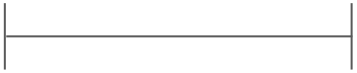
PROTON X

Datetime trong Airflow

Datetime in Airflow

Jobs chạy vào sáng ngày 2023-10-11

2023-10-10 00:00:00 — 2023-10-11 00:00:00



Ngày chạy theo lịch	2023-10-11 00:00:00
Ngày chạy logic (Logical Date)	2023-10-10 00:00:00
Data Interval Start	2023-10-10 00:00:00
Data Interval End	2023-10-11 00:00:00

DAG ID

scheduled__2023-10-10T00:00:00+00:00

Ví dụ:
Một ngân hàng cần upload những giao dịch trong ngày từ Onprem Database lên Data Warehouse hằng ngày

PROTON X

Datetime trong Airflow

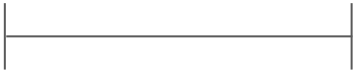
Datetime in Airflow

????

Jobs chạy vào sáng ngày 2023-10-11

2023-10-10 00:00:00

2023-10-11 00:00:00



DAG ID

manual__2023-10-11T00:00:01+00:00

Ví dụ:
Hệ thống bị lỗi và cần phải tự bật DAG

Ngày chạy theo lịch

Ngày chạy logic (Logical Date)

2023-10-11 00:00:01

Ví dụ:
Truyền vào logical date

Data Interval Start

2023-10-10 00:00:00

Data Interval End

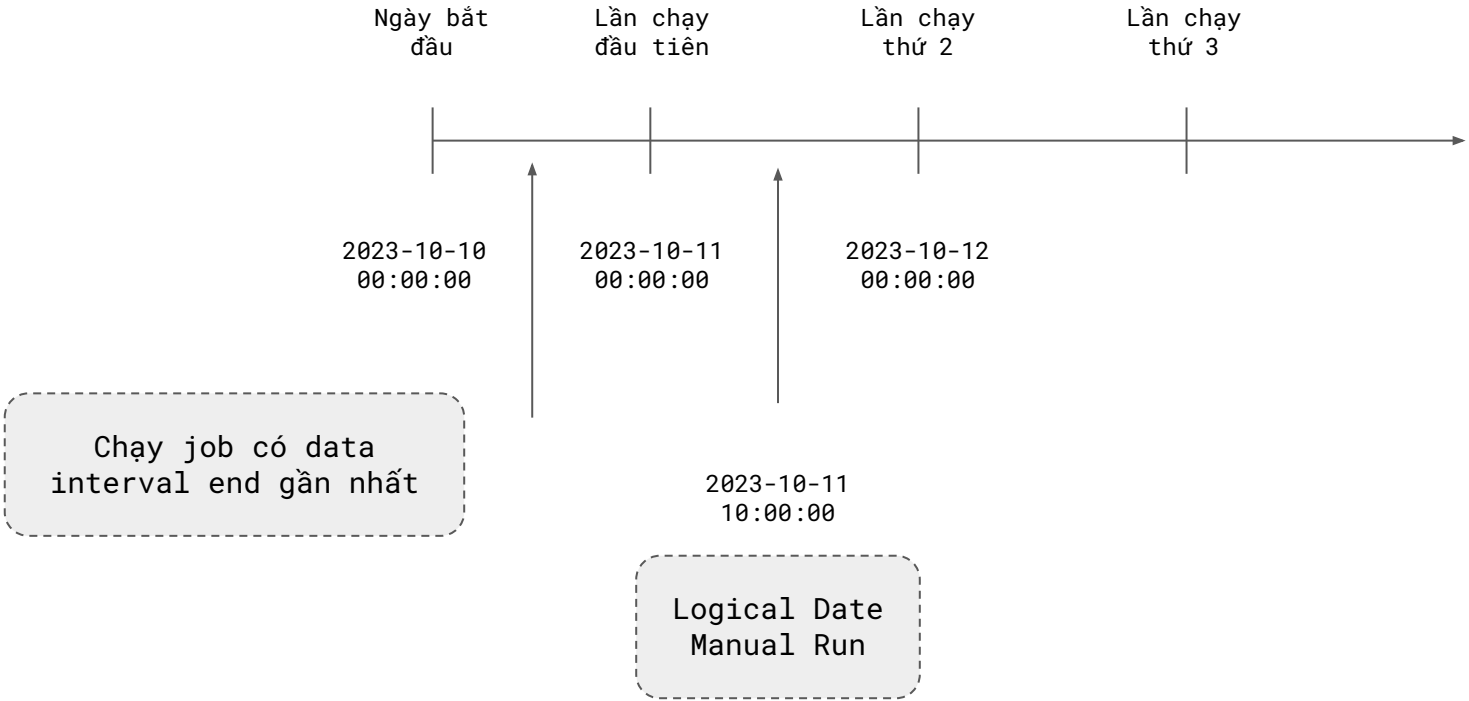
2023-10-11 00:00:00

https://airflow.apache.org/docs/apache-airflow/stable/core-concepts/dag-run.html

PROTON X

Datetime trong Airflow

Datetime in Airflow



PROTON X

Datetime trong Airflow

Datetime in Airflow

```
import pendulum

utc = pendulum.timezone("UTC")
with DAG(
     "start_date" : dt.datetime(2023, 10, 10, tzinfo=utc),
     schedule="0 0 * * *",
)
```

GMT+7	UTC
2023-10-11 07:00:00+07:00	2023-10-11 00:00:00+00:00
2023-10-12 07:00:00+07:00	2023-10-12 00:00:00+00:00

Chạy cron job theo giờ UTC (+ 0:00)

PROTON X

Datetime trong Airflow

Datetime in Airflow

```python
import pendulum

hcm_tz = pendulum.timezone("Asia/Ho_Chi_Minh")
with DAG(
     "start_date" : dt.datetime(2023, 10, 10, tzinfo=hcm_tz),
     schedule="0 0 * * *",
)
```

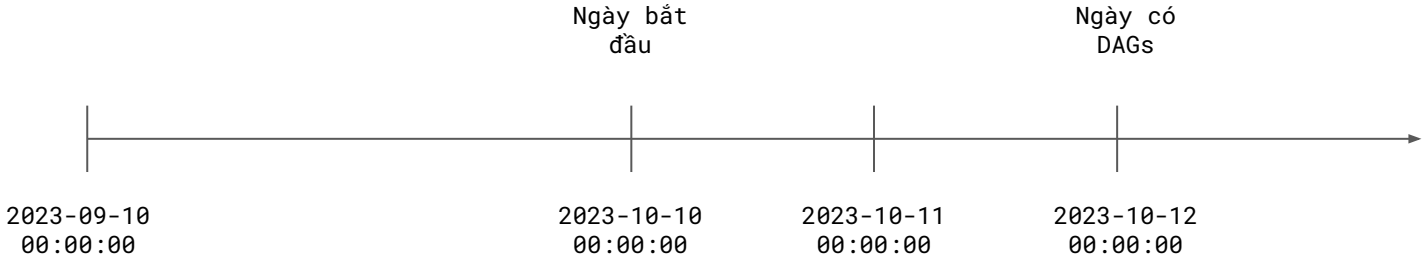
GMT+7	UTC
2023-10-11 00:00:00+07:00	2023-10-10 17:00:00+00:00
2023-10-12 00:00:00+07:00	2023-10-11 17:00:00+00:00

Chạy cron job theo múi giờ GMT+7 (+ 7:00)

PROTON X

Backfill và Catchup

Backfill and Catchup



Catchup từ start date

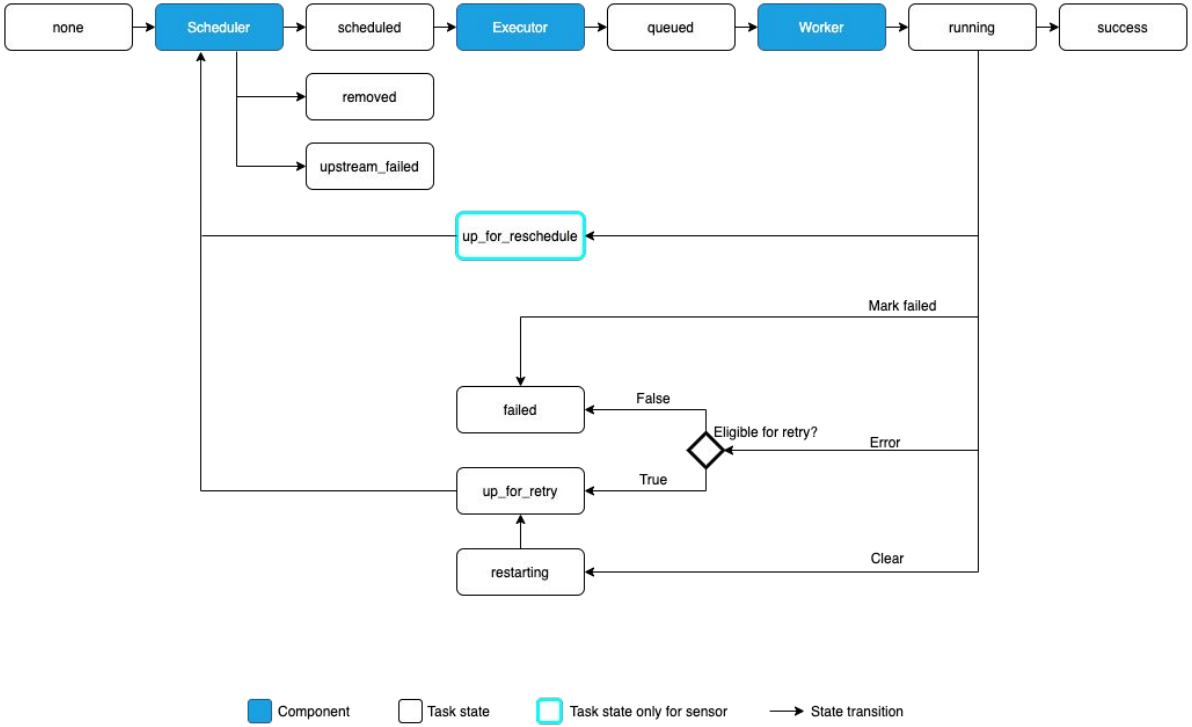
Backfill những ngày chưa chạy và không bị ràng buộc start date

```
airflow dags backfill \
    --start-date START_DATE \
    --end-date END_DATE \
      dag_id
```

PROTON X

Vòng đời của Task

Task lifecycle



https://airflow.apache.org/docs/apache-airflow/stable/_images/task_lifecycle_diagram.png

PROTON X

Một số Operators

Some Operators

Loại	Operator	Chức năng
Airflow Operator	EmptyOperator	Rỗng
	BashOperator	Chạy câu lệnh bash
	PythonOperator	Dùng để chạy hàm python
Providers Operator	KubernetesPodOperators	Tạo và chạy pod trên kubernetes
	BigQueryGetDataOperators	Lấy data trên bigquery

PROTON X

Python Operator

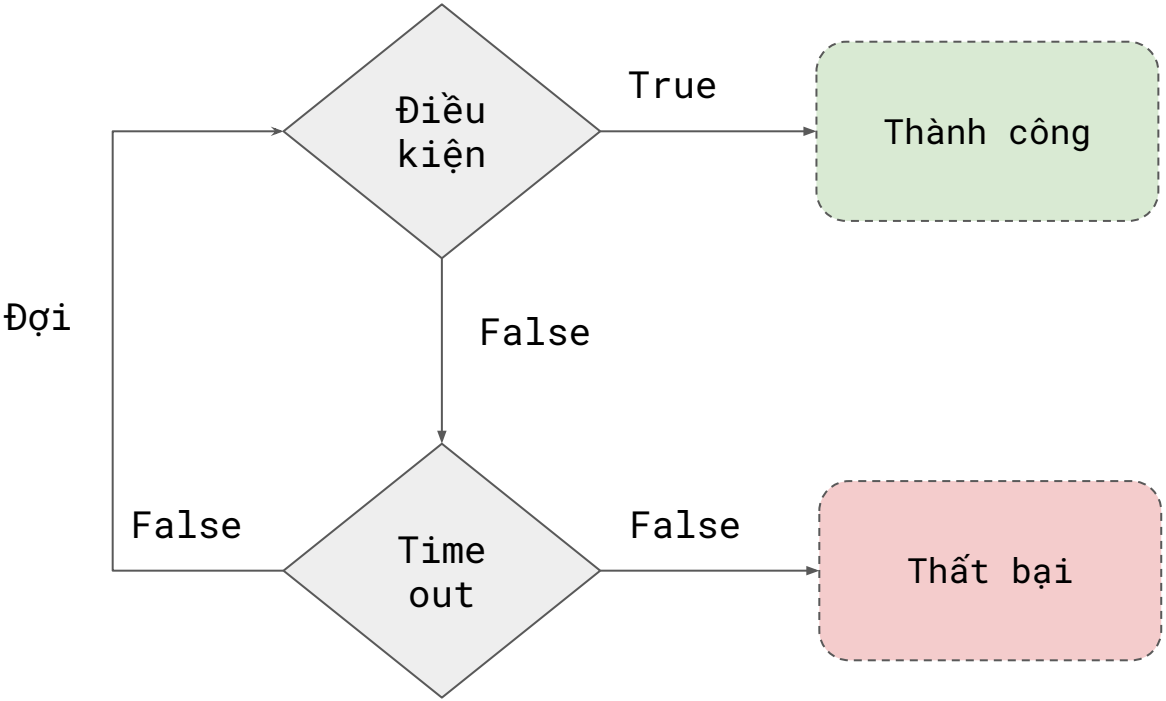
Python Operator

```
from airflow.operators.python import PythonOperator

def print_context(a,**context):
     print(a)
     print(context)

with DAG(...):
     print_context_task = PythonOperator(
                                task_id="print_context",
                                python_callable=print_context,
                                op_args= [ ],
                                op_kwargs= { "a": 1 },
                                provide_context=True,)
```

PROTON X

Sensor

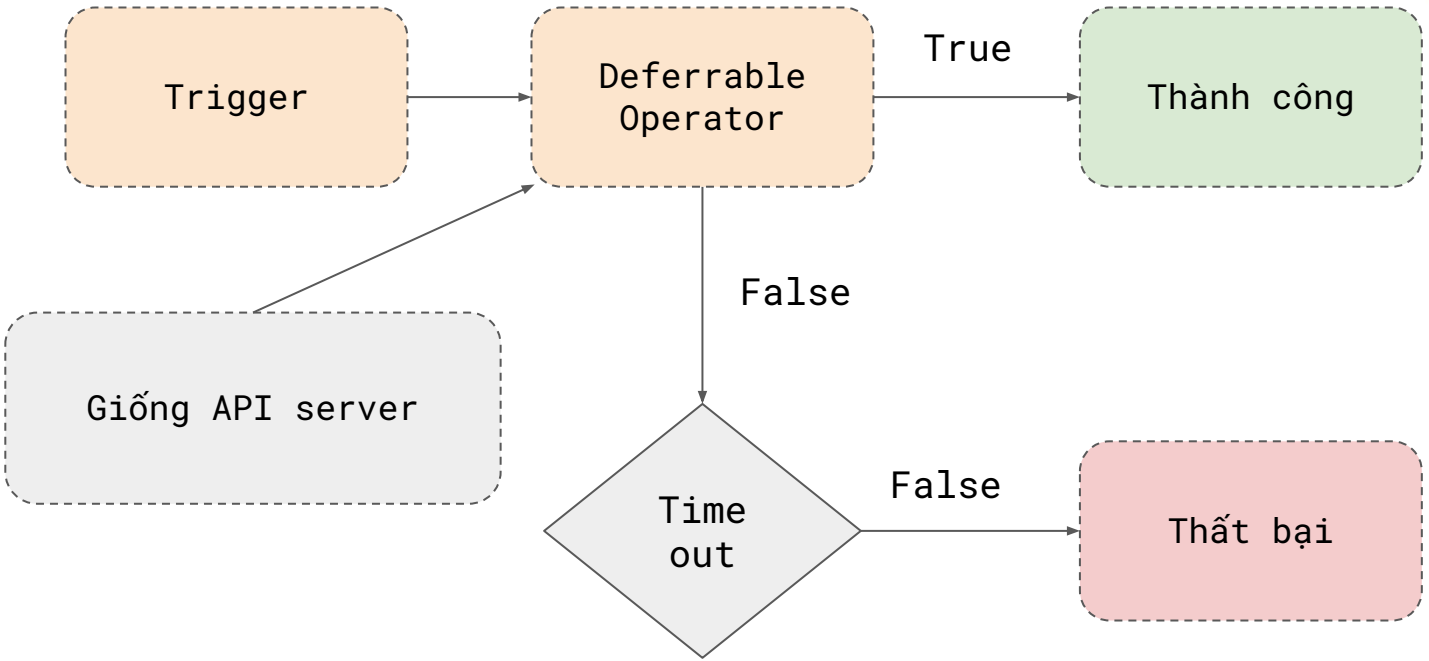
Sensor



PROTON X

Deferrable và Trigger

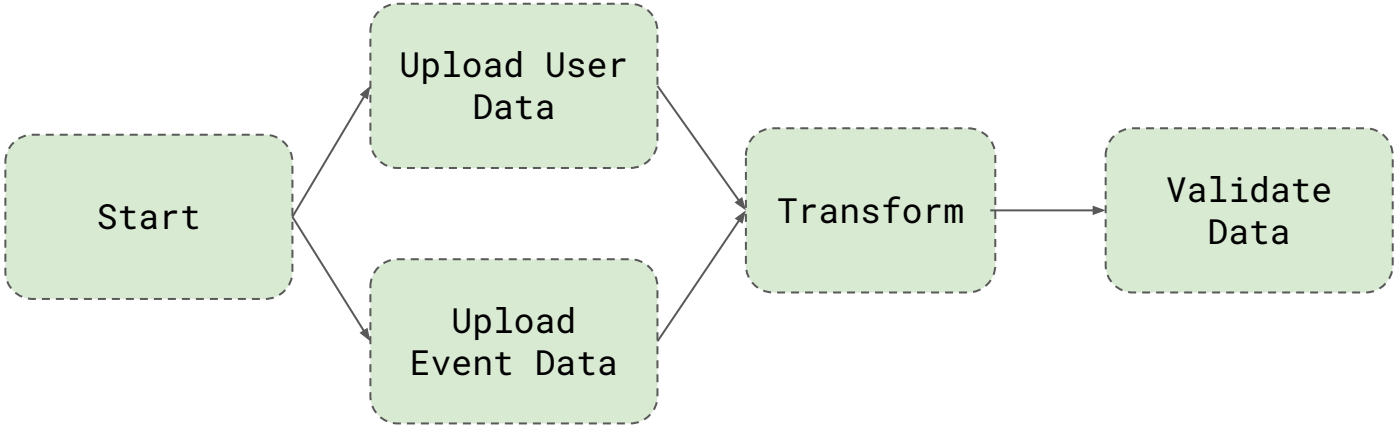
Deferrable and Trigger



PROTON X

Control Flow

Control Flow



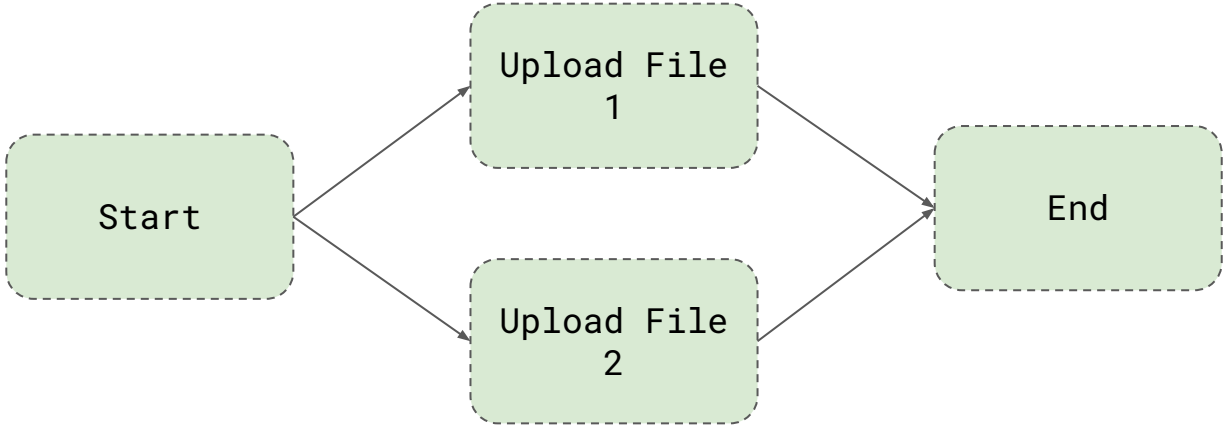

```
upload_tasks = [upload_user_data, consume_event_data]
start >> upload_tasks >> transform >> validate_data
```

```
upload_tasks = [upload_user_data, consume_event_data]
validate_data << transform << upload_tasks << start
```

PROTON X

DAGs động

Dynamic DAGs

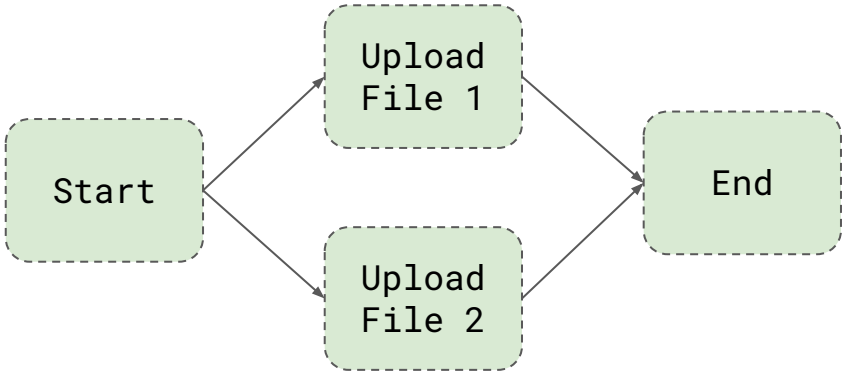


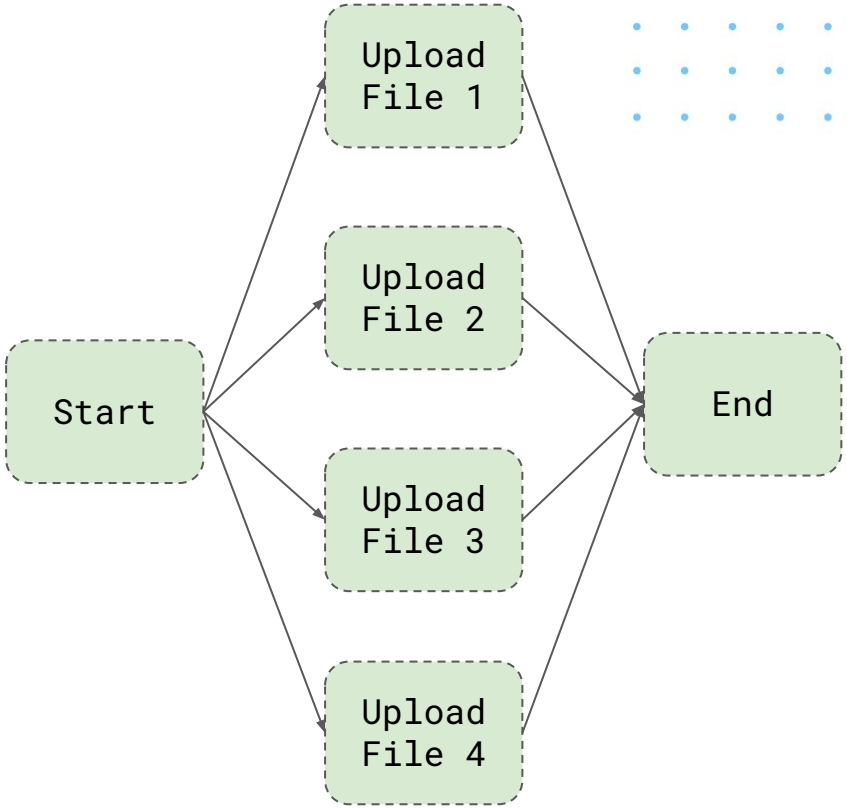

```
start = EmptyOperator(task_id="start")
list_task = []
for i in range(num_file):
     new_task = EmptyOperator(task_id=f"upload_file_{i+1}")
     list_task.append(new_task)
start >> list_task >> EmptyOperator(task_id="end")
```

PROTON X

DAGs động

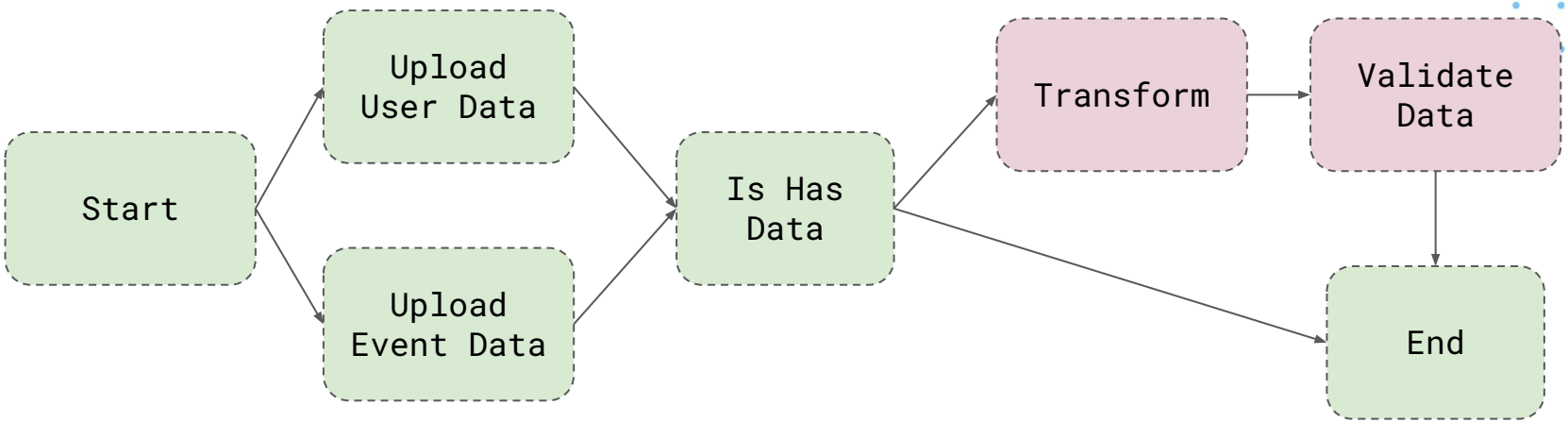
Dynamic DAGs





Rẽ nhánh

Branching



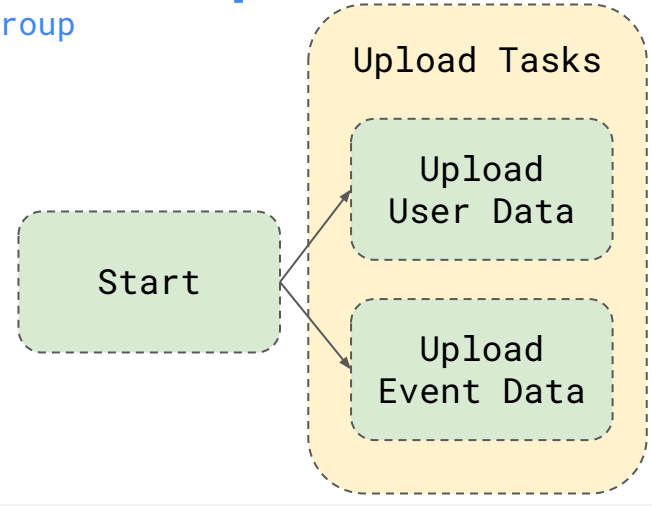

```python
@task.branch(task_id="is_has_data")
def if_have_data():
    check_is_has_data = False
    if check_is_has_data:
        return "transform"
    else:
        return "end"
```

```python
from airflow.utils.trigger_rule import TriggerRule

end = EmptyOperator(
task_id="end",
trigger_rule=TriggerRule.NONE_FAILED_MIN_ONE_SUCCESS
)
```

PROTON X

Task Group

Task Group

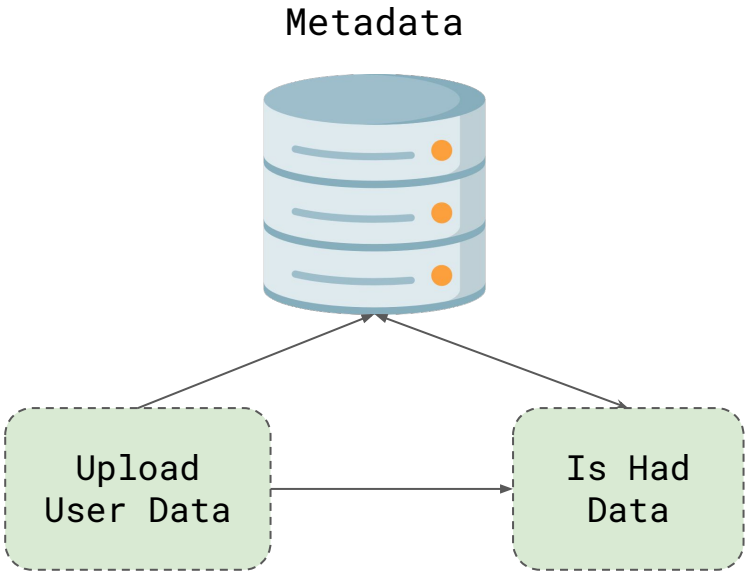



```python
from airflow.decorators import task_group

@task_group(default_args={})
def upload_tasks():
     upload_user_data = EmptyOperator(task_id="upload_user_data")
     upload_event_data = EmptyOperator(task_id="upload_event_data")

start >> upload_tasks()
```

PROTON X

Xcoms

Cross Communication

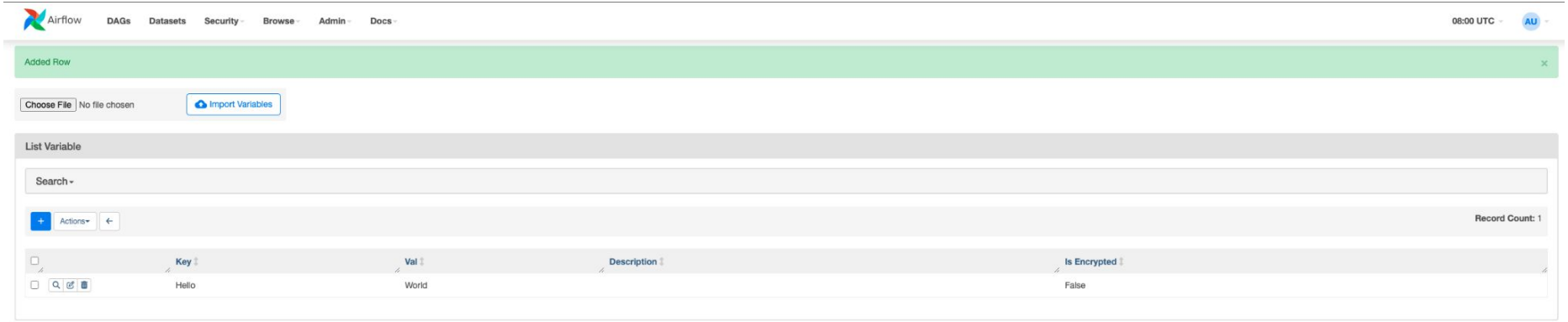



```python
def upload_tasks(**context):
     return True

def is_has_data(is_has_data):
     return True

is_has_data_node = PythonOperator(
          ...,
          python_callable=is_has_data,
          op_kwargs={
          "is_has_data":is_has_data.output
          }
)
```

PROTON X

Variable

Variable




```python
from airflow.models import Variable

# Auto-deserializes a JSON value
bar = Variable.get("Hello", deserialize_json=True,default_var=None)
```

PROTON X

Jinja Template

Jinja Template

Dùng để xử lý những thông tin động chỉ biết được tại thời điểm chạy DAGs

```
"{{ ds }}"
```

```
"2023-10-10"
```

```
"{{ ds_nodash }}"
```

```
"20231010"
```

```
"bigquery_table_shade_{{ ds_nodash }}"
```

```
"bigquery_table_shade_20231010"
```

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

PROTON X

Jinja Template

Jinja Template

```
with DAG(
     "dag_with_macro",
     schedule_interval="0 0 * * *",
     default_args=default_args,
     user_defined_macros={
          "hello_world" : hello_world()
     },
     catchup= False,
) as dag:
```

Có thể tự viết macro để parse trong jinja template của airflow bằng args user_defined_macros

PROTON X

Params

Params

```
from airflow import DAG
from airflow.models.param import Param

with DAG("the_dag",
     params={"x": Param(5, type="integer", minimum=3),
          "my_int_param": 6}):
     PythonOperator(
          task_id="from_template",
          op_args=["{{ params.my_int_param }}",],
          python_callable=(lambda my_int_param: print(my_int_param))
     )
```

https://airflow.apache.org/docs/apache-airflow/stable/core-concepts/params.html#params

PROTON X

Connection và Hook

Connection and Hook

hook_1
GSC client

connection_1
HOST
USERNAME
PASSWORD

Apache
Airflow

Postgres

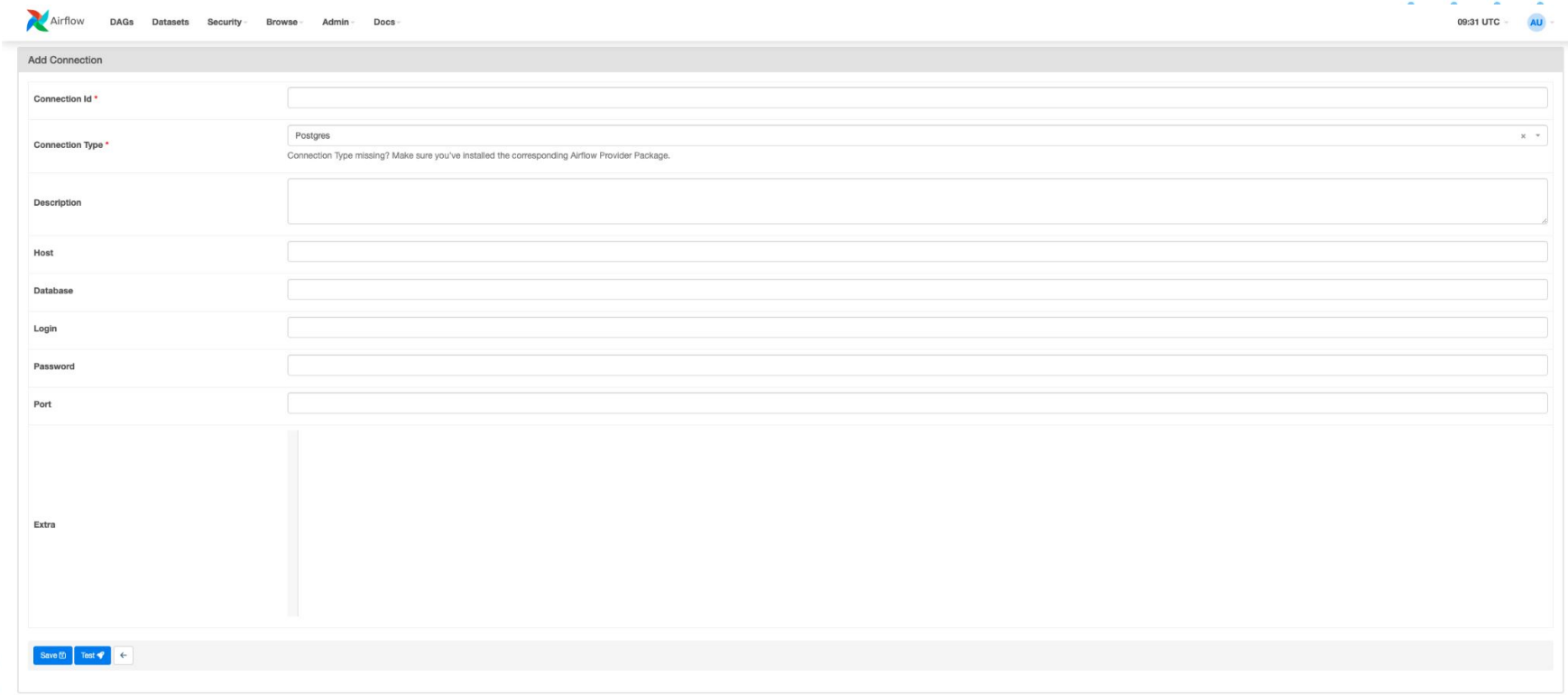
hook_2
Postgres Client

connection_2
HOST
USERNAME
PASSWORD

PROTON X

Connection

Connection



PROTON X

Connection

Connection

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


query = PostgresOperator(
     task_id = "postgres_query",
     postgres_conn_id="postgres_local",
     sql = 'select * from "public"."user_info" '
)
```