Giảng viên biên soạn: Phạm Thị Xuân Hiền

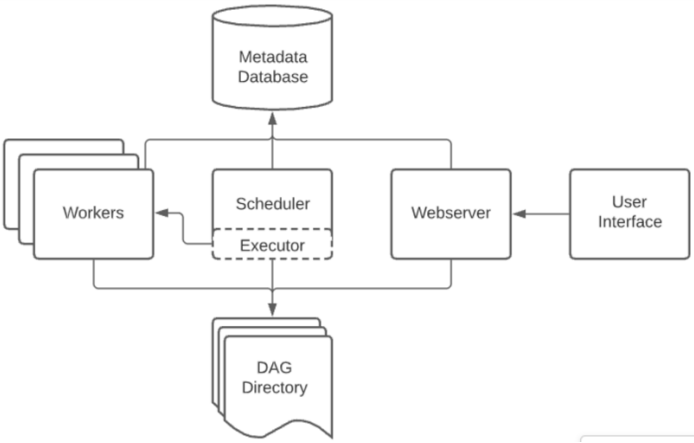
HƯỚNG DẪN APACHE AIRFLOW

THEORETICAL BASIS

1. Definition

Apache airflow là một nền tảng giúp hỗ trợ việc lập lịch và quản lý các workflow thông qua việc lập trình

2. Overview Architecture



I. DOWNLOAD AND INSTALL

Method 1: Direct Installation on Local Machine

a) Installation

- Install Airflow via Pip:

```
pip install apache-airflow
```

- Initialize database

```
airflow db init
```

- Create user

```
airflow users create --username [tên đăng nhập] --password [mật khẩu] --firstname [Tên] --lastname [Họ] --role Admin --email [email]
```

- Configuration: customize the settings in the '**airflow.cfg**' file

b) Run

- Initialize web server

```
airflow webserver -p 8080
```

- Initialize Scheduler (in new terminal)

```
airflow scheduler
```

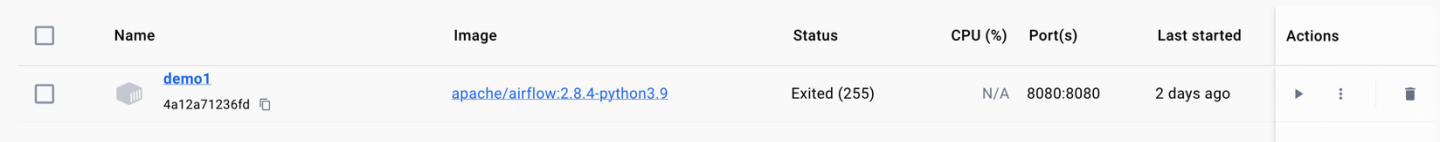
Method 2: Use Docker

Giảng viên biên soạn: Phạm Thị Xuân Hiền

- Open new terminal

```
docker run -it --name demo1 -v
/Users/macintoshhd/Desktop/apache_airflow/dags:/opt/airflow/dags -p 8080:8080 -d
apache/airflow:2.8.4-python3.9 bash
```

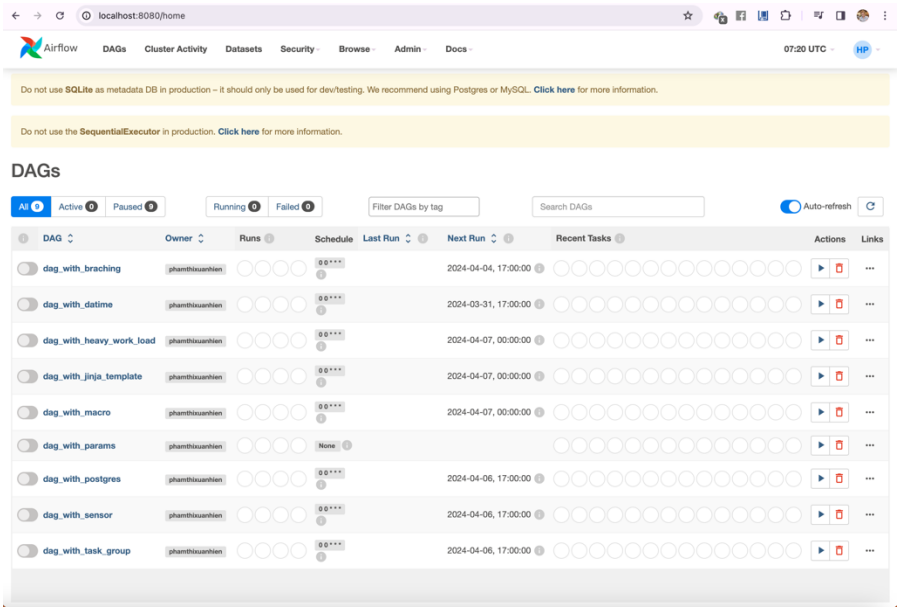
- Open Docker Desktop



- Copy all files from the dags local directory to the dags folder on Apache Airflow

```
bash -c '(airflow db init && airflow users create --username admin --password admin
--firstname Hien --lastname Pham --role Admin --email phamthixuanhien@iuh.edu.vn);
airflow webserver & airflow scheduler'
```

- Open "**localhost:8080**" in Google Chrome



link ref: https://airflow.apache.org/docs/apache-airflow/1.10.4/start.html

Giảng viên biên soạn: Phạm Thị Xuân Hiền

II. INTRODUCE

1. DAG with Branch

dags > dag_with_braching.py > ...

```python
import datetime as dt
import pendulum
from airflow import DAG
from airflow.decorators import task, task_group
from airflow.operators.empty import EmptyOperator
from airflow.utils.trigger_rule import TriggerRule

# utc = pendulum.timezone("UTC")
hcm_tz = pendulum.timezone("Asia/Ho_Chi_Minh")

default_args = {
    "owner": "phamthixuanhien",
    # "start_date": dt.datetime(2024, 4, 5, tzinfo=hcm_tz),
    "start_date": dt.datetime(year=2024, month=4, day= 5, tzinfo=hcm_tz),
    # "start_date": dt.datetime(year=2024, month=4, day= 1, tzinfo=utc),
    "retries": 1,
    "retry_delay": dt.timedelta(minutes=5),
    "depend_on_past": True,
    "email": ["phamthixuanhien@iuh.edu.vn"]
}

@task.branch(task_id="branching")
def task_braching():
    return "task_1"

with DAG(
    "dag_with_braching",
    schedule_interval="0 0 * * *",
    default_args=default_args,
    catchup= True,
) as dag:
    start = EmptyOperator(task_id="start")


    branch = task_braching()

    task_1 = EmptyOperator(task_id="task_1")
    task_2 = EmptyOperator(task_id="task_2")

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

    # Flow
    start >> branch >> [ task_1 , task_2 ] >> end

```

with DAG(…): Khởi tạo 1 DAG với tên là dag_with_braching

Schedule_interval= “0 0 * *”: DAG này chạy hàng ngày vào lúc nửa đêm

Retries:1 - số lần thử lại là 1

Retry_delay: khoảng thời gian chờ giữa các lần thử lại: 5 phút

start = EmptyOperator(task_id="start"): Tạo một task trống với ID là ‘start’

branch = task_braching(): Một task cho phép thực hiện nhánh điều kiện (chưa rõ hành động cụ thể vì nó phụ thuộc vào định nghĩa của task_branching()

trigger_rule=TriggerRule.NONE_FAILED_MIN_ONE_SUCCESS: nghĩa là task này chỉ thực thi khi ít nhất một trong các tasks trước nó thành công và không có task nào thất bại.

Định nghĩa luồng công việc (workflow): start chạy đầu tiên >> tiếp theo là branch, sau đó dựa trên kết quả của branch, một trong hai (hoặc cả hai) task_1, task_2 sẽ được chạy >> cuối cùng là end

Giảng viên biên soạn: Phạm Thị Xuân Hiền

2. DAG with param

```python
default_args = {
    "owner": "phamthixuanhien",
    "start_date": dt.datetime(2024, 4, 6, tzinfo=hcm_tz),
    "retries": 1,
    "retry_delay": dt.timedelta(minutes=5),
    "depend_on_past": True,
    "email": ["phamthixuanhien@iuh.edu.vn"]
}


def print_params(stt, email,sex):
    print(stt)
    print(email)
    print(sex)

with DAG(
    "dag_with_params",
    schedule=None,
    default_args=default_args,
    params={
        "stt": Param(1, type="integer", minimum=1),
        "email": 'phamthixuanhien@iuh.edu.vn',
        "sex" : Param("Female", type="string", enum = ["Female","Male"]),
    },
    catchup= False,
) as dag:

    start = EmptyOperator(task_id="start")

    print_string_task = PythonOperator(
        task_id="print_string",
        python_callable=print_params,
        op_kwargs={
            "stt": "{{ params.stt }}",
            "email": "{{ params.email }}",
            "sex" : "{{ params.sex }}"
        },
    )

    end = EmptyOperator(task_id="end")

    # Flow
    start >> print_string_task >> end
```

params={…}: đây là một phần đặc biệt của đoạn mã. Đối tượng params được dùng để truyền các tham số tuỳ chỉnh vào DAG. 3 tham số params được định nghĩa sau:

- "stt": tham số x với giá trị mặc định là 1, kiểu dữ liệu là số nguyên, và giá trị tối thiểu là 1
- "email": giá trị mặc định là phamthixuanhien@iuh.edu.vn
- "sex": tham số choice có giá trị mặc định là Female, kiểu dữ liệu là chuỗi, và chỉ chấp nhận các giá trị "Female" hoặc "Male"

Tạo một tác vụ sử dụng PythonOperator, loại tác vụ này cho phép thực thi một hàm Pytho trong quá trình chạy DAG

- task_id: đặt tên cho tác vụ
- python_callable=print_params: chỉ định hàm python (print_params) để chạy. Hàm này cần được định nghĩa ở nơi khác trong mã
- op_kwargs={…}: truyền các tham số cho hàm print_params: Các giá trị lấy giá trị tương ứng từ đối tượng 'params' đã được định nghĩa trong DAG

3. DAG with Task_Group

```python
with DAG(
    "dag_with_task_group",
    schedule_interval="0 0 * * *",
    default_args=default_args,
    catchup= False,
) as dag:
    start = EmptyOperator(task_id="start")

    @task_group()
    def task_group_1():
        EmptyOperator(task_id="do_something_1") >> [EmptyOperator(task_id="do_something_2"),
                                                    EmptyOperator(task_id="do_something_3")]

    end = EmptyOperator(task_id="end")

    # Flow
    start >> task_group_1() >> end
```

task_group_1(): được thực hiện bằng cách sử dụng @task_group()

Nhóm này bao gồm 3 emptyOperator với là công việc đầu tiên task_id="do_something_1" , sau đó 2 công việc song song là task_id="do_something_2" Và task_id="do_something_3"

Giảng viên biên soạn: Phạm Thị Xuân Hiền

Trong đoạn mã này mô tả một luồng công việc đơn giản trong airflow với nhóm các công việc bao gồm công việc đơn lẻ và hai công việc song song. Điều này giúp tổ chức và quản lý các công việc phức tạp hơn một cách dễ dàng hơn.

4. DAG with PostgreSQL

Apache Airflow to run an SQL query on PostgreSQL use PostgresOperator

```
with DAG(
    "dag_with_postgres",
    schedule_interval="0 0 * * *",
    default_args=default_args,
    catchup= False,
) as dag:
    start = EmptyOperator(task_id="start")

    query = PostgresOperator(
        task_id = "postgres_query",
        postgres_conn_id="postgres_local",
        sql = 'select * from "QLSV"."SinhVien" '
    )


    end = EmptyOperator(task_id="end")
    # Flow

    start >> query >> end
```

query = PostgresOperator(…):Tạo một task sử dụng PostgresOperator, dùng để thực hiện các truy vấn SQL trên cơ sở dữ liệu PostgreSQL. task_id = "postgres_query"đặt tên cho task, postgres_conn_id="postgres_local"định nghĩa kết nối đến cơ sở dữ liệu PostgreSQL (được cấu hình trước trong Airflow), và sql = 'select * from "QLSV"."SinhVien" ' là câu lệnh SQL được thực hiện, trong trường hợp này là truy vấn tất cả dữ liệu từ bảng SinhVien trong schema QLSV.

5. DAG with Bigquery

Apache Airflow to run an SQL query on Bigquery use BigQueryExecuteQueryOperator

```
# Tạo DAG
with DAG(
    'bigquery_airflow_example',
    default_args=default_args,
    schedule_interval='@daily',  # Chạy DAG hàng ngày
    catchup=False
) as dag:
    start = EmptyOperator(task_id="start")
    # Tạo một task để thực hiện truy vấn trên BigQuery
    bq_query_task = BigQueryExecuteQueryOperator(
        task_id='bigquery_query',
        sql='''
            SELECT * FROM `your-project.QLSV.SinhVien`
            WHERE MaBM = 'HTTT'
            LIMIT 100
        ''',
        use_legacy_sql=False,  # Sử dụng Standard SQL (không phải SQL Legacy của BigQuery)
        location='your-location',  # Thay thế bằng vị trí của BQ dataset của bạn
        gcp_conn_id='your_gcp_connection',  # Thay thế bằng ID của kết nối GCP trong Airflow
    )


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

    # Flow
    start >> bq_query_task >> end
```

Giảng viên biên soạn: Phạm Thị Xuân Hiền

III. EXERCISE

Ex1: Làm lại 5 bài tập trên

lưu ý bài 4,5 cần cấu hình

Ex2:

Xây dựng một DAG trong Airflow để thực hiện các công việc sau:

- task 1: in ra dòng lệnh "Công việc 1"
- Task 2: viết một hàm xử lý dữ liệu process_data(), nội dung bên trong tuỳ ý.
- Task 3: viết một hàm lưu trữ dữ liệu save_data(), nội dung bên trong tuỳ ý

Yêu cầu:

- Đặt DAG ID là 'MSSV_DAG2' (MSSV của sinh viên)
- Định nghĩa các tham số trong DAG
    - Owner là họ tên của sinh viên
    - Không phụ thuộc vào các phiên bản trước đó (depends_on_past: False)
    - Thời điểm bắt đầu: ngày hiện tại theo múi giờ ASIA/HoChiMinh (giờ Việt Nam)
    - Số lần thử lại là 1 (retries:1)
    - Khoảng thời gian chờ giữa các lần thử lại là 2 phút
    - Thiết lập cho DAG chạy mỗi phút

Submission Regulations

- Thực hiện các bài tập sau, nhớ chụp màn hình kết quả vào word.
- Nộp file word và folder chứ code
- Quy tắc đặt tên: MSSV.zip