Airflow Zero to Hero

郭益華

GitHub

目錄

Run Airflow in Docker

建立一個 airflow_docker 資料夾

```
Windows PowerShell
著作權（C） Microsoft Corporation。保留擁有權利。

安裝最新的 PowerShell 以取得新功能和改進功能！https://aka.ms/PSWindows

PS C:\Users\jerry> cd "C:\Users\jerry\Desktop\mastercourse\dataEngineer\airflow\airflow_docker"
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\airflow\airflow_docker>
```

搜尋 airflow docker compose

複製此指令

Fetching docker-compose.yaml

To deploy Airflow on Docker Compose, you should fetch docker-compose.yaml.

```
curl -Lf0 'https://airflow.apache.org/docs/apache-airflow/2.7.2/docker-compose.yaml'
```

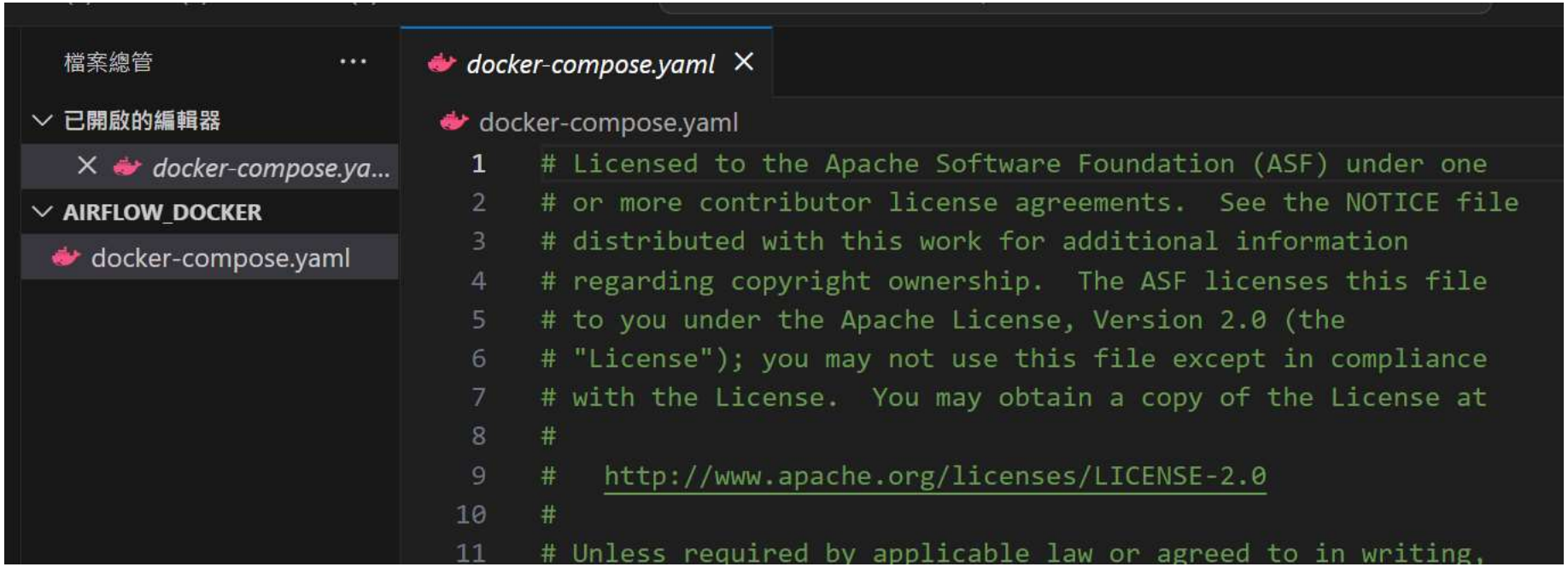
This file contains several service definitions:

Windows Powershell :
curl 'https://airflow.apache.org/docs/apache-airflow/2.7.2/docker-compose.yaml' –o 'docker-compose.yaml'
Linux:
curl –Lf0 'https://airflow.apache.org/docs/apache-airflow/2.7.2/docker-compose.yaml'

下載

```
Windows PowerShell

PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\airflow\airflow_docker> curl 'https://airflow.apache.org/docs/apache
-airflow/2.7.2/docker-compose.yaml' -o 'docker-compose.yaml'
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\airflow\airflow_docker> ls

    目錄: C:\Users\jerry\Desktop\mastercourse\dataEngineer\airflow\airflow_docker


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----       2023/10/26   下午 08:03          11783 docker-compose.yaml
```

查看docker-compose.yaml



可看到有dag, log, plugins

```yaml
volumes:
  - ${AIRFLOW_PROJ_DIR:-.}/dags:/opt/airflow/dags
  - ${AIRFLOW_PROJ_DIR:-.}/logs:/opt/airflow/logs
  - ${AIRFLOW_PROJ_DIR:-.}/config:/opt/airflow/config
  - ${AIRFLOW_PROJ_DIR:-.}/plugins:/opt/airflow/plugins
user: "${AIRFLOW_UID:-50000}:0"
```

在airflow_docker中新增三個資料夾

```
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\airflow\airflow_docker> mkdir dags

    目錄: C:\Users\jerry\Desktop\mastercourse\dataEngineer\airflow\airflow_docker

Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----      2023/10/26  下午 08:06                dags

PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\airflow\airflow_docker> mkdir logs

    目錄: C:\Users\jerry\Desktop\mastercourse\dataEngineer\airflow\airflow_docker

Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----      2023/10/26  下午 08:06                logs

PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\airflow\airflow_docker> mkdir plugins

    目錄: C:\Users\jerry\Desktop\mastercourse\dataEngineer\airflow\airflow_docker
```

新增成功

```
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\airflow\airflow_docker> ls


    目錄: C:\Users\jerry\Desktop\mastercourse\dataEngineer\airflow\airflow_docker


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----       2023/10/26  下午 08:06                dags
d-----       2023/10/26  下午 08:06                logs
d-----       2023/10/26  下午 08:06                plugins
-a----       2023/10/26  下午 08:03          11783 docker-compose.yaml
```

查看預設的airflow帳號密碼

```yaml
# yamllint enable rule:line-length
environment:
  <<: *airflow-common-env
  _AIRFLOW_DB_MIGRATE: 'true'
  _AIRFLOW_WWW_USER_CREATE: 'true'
  _AIRFLOW_WWW_USER_USERNAME: ${_AIRFLOW_WWW_USER_USERNAME:-airflow}
  _AIRFLOW_WWW_USER_PASSWORD: ${_AIRFLOW_WWW_USER_PASSWORD:-airflow}
  _PIP_ADDITIONAL_REQUIREMENTS: ''
user: "0:0"
```

Username: airflow
Password: airflow

建立docker airflow 初始化

Command:
docker compose up airflow-init

```
Windows PowerShell

PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\airflow\airflow_docker> docker compose up airflow-init
time="2023-10-26T20:09:36+08:00" level=warning msg="The \"AIRFLOW_UID\" variable is not set. Defaulting to a blank strin
g."
time="2023-10-26T20:09:36+08:00" level=warning msg="The \"AIRFLOW_UID\" variable is not set. Defaulting to a blank strin
g."
[+] Running 0/23
 - redis Pulling                                                                                                    4.4s
   - 4f4fb700ef54 Waiting                                                                                           0.4s
   - a378f10b3218 Waiting                                                                                           0.1s
   - b266cd8112a6 Waiting                                                                                           0.0s
```

建立完成畫面

```
airflow_docker-airflow-init-1  | [2023-10-26T12:14:33.937+0000] {manager.py:555} INFO - Added Permission %s to role %s
airflow_docker-airflow-init-1  | [2023-10-26T12:14:33.968+0000] {manager.py:499} INFO - Created Permission View: %s
airflow_docker-airflow-init-1  | [2023-10-26T12:14:33.977+0000] {manager.py:555} INFO - Added Permission %s to role %s
airflow_docker-airflow-init-1  | [2023-10-26T12:14:33.993+0000] {manager.py:499} INFO - Created Permission View: %s
airflow_docker-airflow-init-1  | [2023-10-26T12:14:34.001+0000] {manager.py:555} INFO - Added Permission %s to role %s
airflow_docker-airflow-init-1  | [2023-10-26T12:14:34.808+0000] {manager.py:211} INFO - Added user %s
airflow_docker-airflow-init-1  | User "airflow" created with role "Admin"
airflow_docker-airflow-init-1  | 2.7.2
airflow_docker-airflow-init-1 exited with code 0
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\airflow\airflow_docker>
```

啟動docker compose

Command:
docker compose up

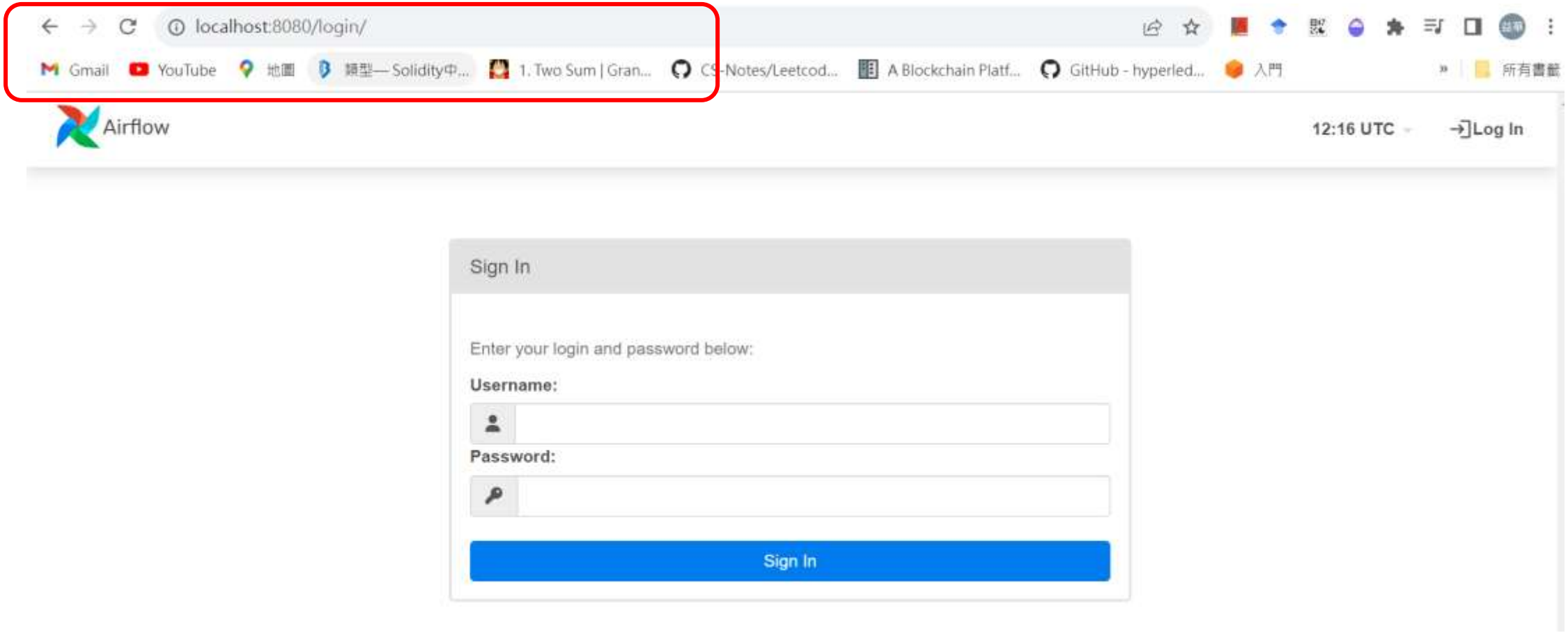
```
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\airflow\airflow_docker> docker compose up
time="2023-10-26T20:15:11+08:00" level=warning msg="The \"AIRFLOW_UID\" variable is not set. Defaulting to a blank strin
g."
time="2023-10-26T20:15:11+08:00" level=warning msg="The \"AIRFLOW_UID\" variable is not set. Defaulting to a blank strin
g."
[+] Running 7/7
 - Container airflow_docker-redis-1              Recreated                                                    0.5s
 - Container airflow_docker-postgres-1           Recreated                                                    0.5s
 - Container airflow_docker-airflow-init-1       Recreated                                                    0.1s
 - Container airflow_docker-airflow-webserver-1  Created                                                      0.1s
 - Container airflow_docker-airflow-worker-1     Created                                                      0.1s
 - Container airflow_docker-airflow-scheduler-1  Created                                                      0.1s
 - Container airflow_docker-airflow-triggerer-1  Created                                                      0.1s
Attaching to airflow_docker-airflow-init-1, airflow_docker-airflow-scheduler-1, airflow_docker-airflow-triggerer-1, airf
low_docker-airflow-webserver-1, airflow_docker-airflow-worker-1, airflow_docker-postgres-1, airflow_docker-redis-1
airflow_docker-redis-1              | 1:C 26 Oct 2023 12:15:12.601 # WARNING Memory overcommit must be enabled! Without
it, a background save or replication may fail under low memory condition. Being disabled, it can also cause failures wit
hout low memory condition, see https://github.com/jemalloc/jemalloc/issues/1328. To fix this issue add 'vm.overcommit_me
mory = 1' to /etc/sysctl.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this to take effect
```

查看當前container狀態，確定啟動成功

Command:
docker ps

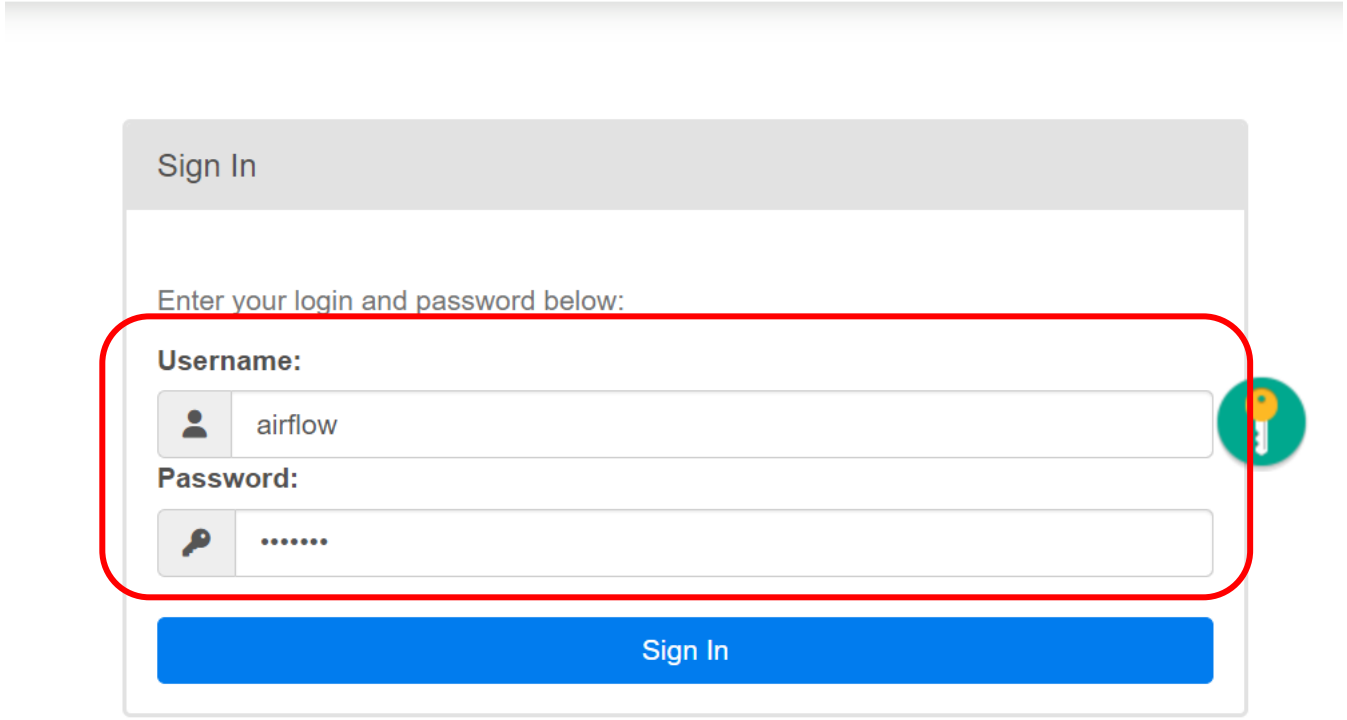
```
PS C:\Users\jerry> docker ps
CONTAINER ID   IMAGE                  COMMAND                  CREATED         STATUS                          PORTS
                   NAMES
362a77edfa0a   apache/airflow:2.7.2   "/usr/bin/dumb-init …"   35 seconds ago   Up 5 seconds (health: starting)   8080/t
cp                 airflow_docker-airflow-scheduler-1
e21ca6c0df98   apache/airflow:2.7.2   "/usr/bin/dumb-init …"   35 seconds ago   Up 5 seconds (health: starting)   8080/t
cp                 airflow_docker-airflow-worker-1
d1bd24aeeef4   apache/airflow:2.7.2   "/usr/bin/dumb-init …"   35 seconds ago   Up 5 seconds (health: starting)   0.0.0.
0:8080->8080/tcp   airflow_docker-airflow-webserver-1
884754c6a479   apache/airflow:2.7.2   "/usr/bin/dumb-init …"   35 seconds ago   Up 5 seconds (health: starting)   8080/t
cp                 airflow_docker-airflow-triggerer-1
14b2ed3ba694   postgres:13            "docker-entrypoint.s…"   35 seconds ago   Up 33 seconds (healthy)           5432/t
cp                 airflow_docker-postgres-1
a91b8584285d   redis:latest           "docker-entrypoint.s…"   35 seconds ago   Up 33 seconds (healthy)           6379/t
cp                 airflow_docker-redis-1
PS C:\Users\jerry>
```

瀏覽器輸入 localhost:8080

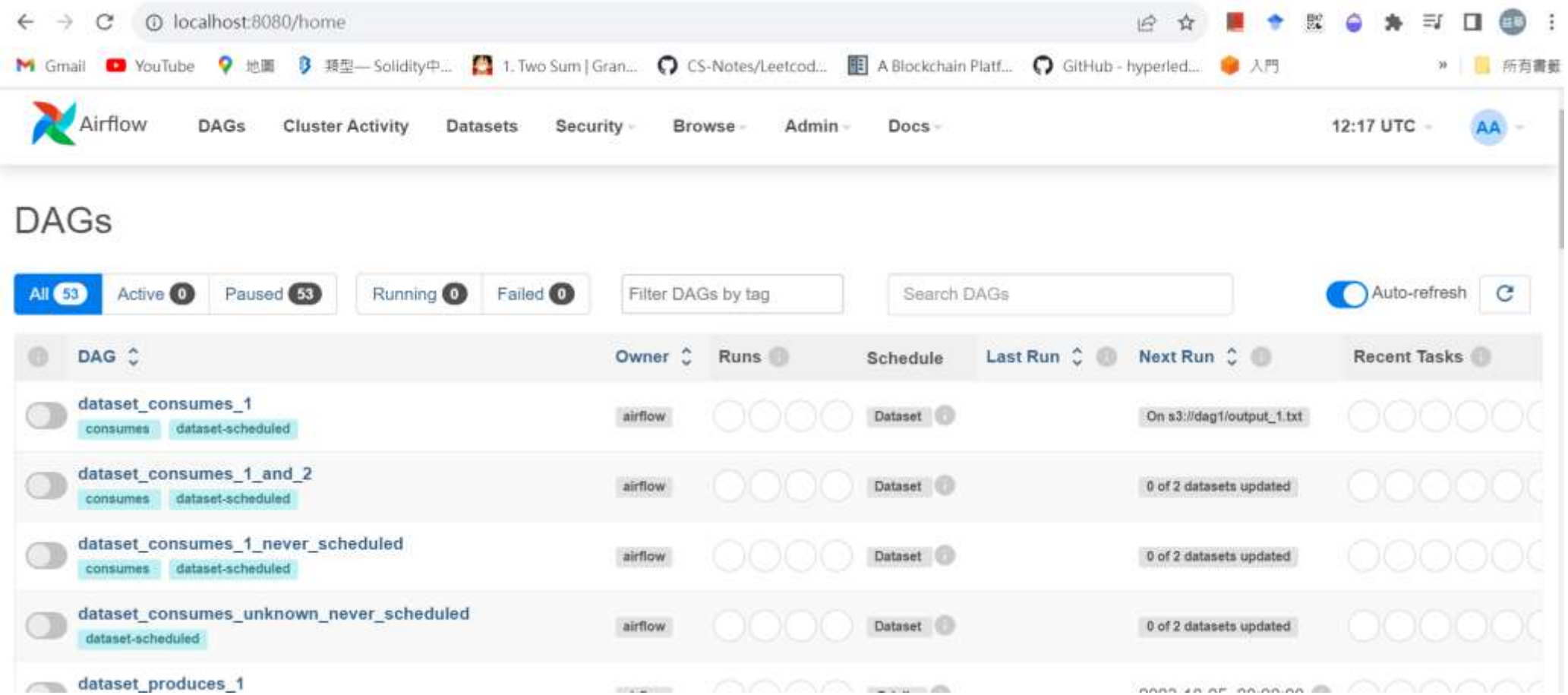


輸入帳號密碼

Username: airflow
Password: airflow

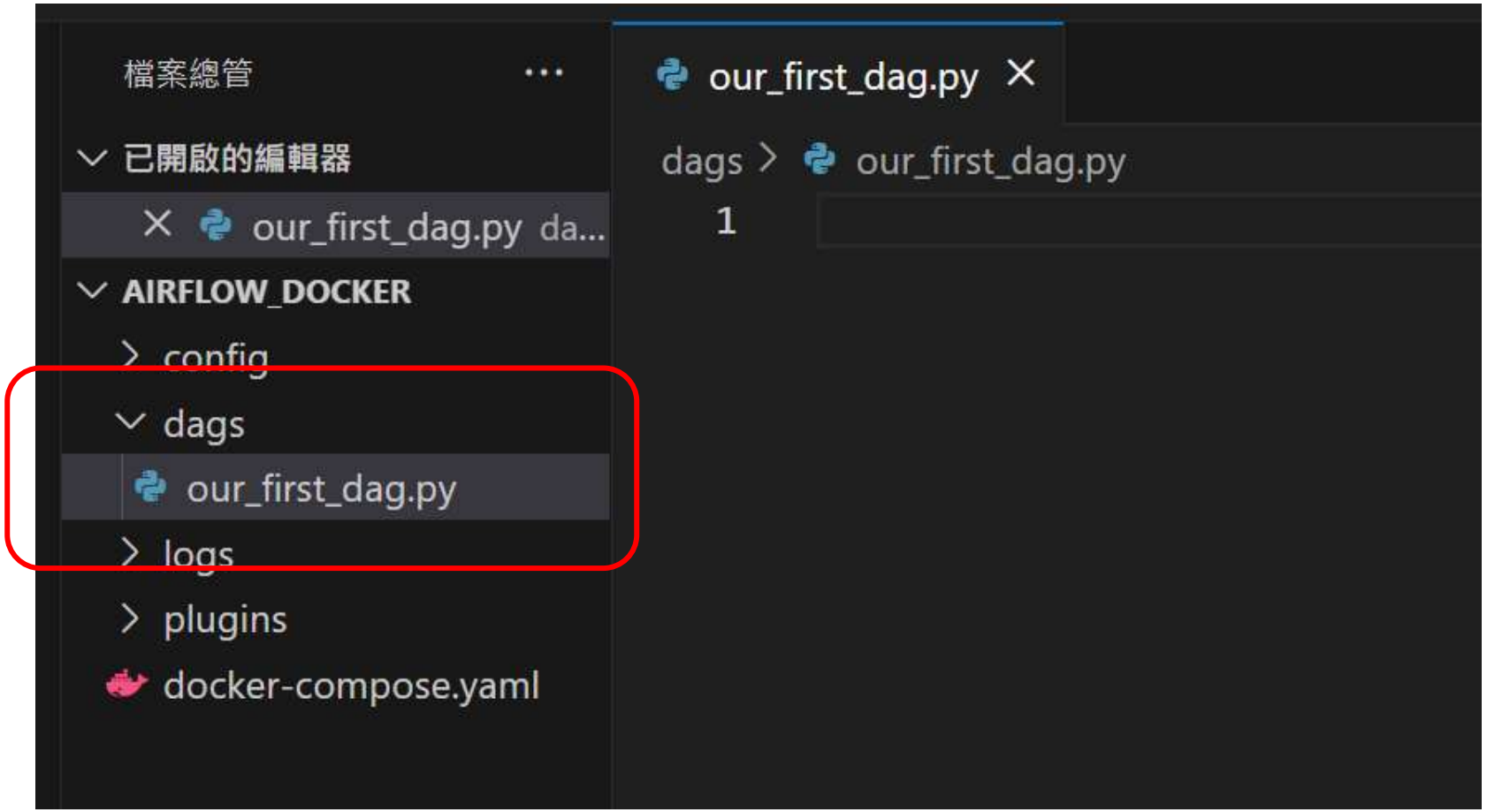


成功登入畫面


localhost:8080/home
Gmail
YouTube
地圖
類型— Solidity中...
1. Two Sum | Gran...
CS-Notes/Leetcod...
A Blockchain Platf...
GitHub - hyperled...
入門
所有書籍
Airflow
DAGs
Cluster Activity
Datasets
Security
Browse
Admin
Docs
12:17 UTC
AA
DAGs
All 53
Active 0
Paused 53
Running 0
Failed 0
Filter DAGs by tag
Search DAGs
Auto-refresh
DAG
Owner
Runs
Schedule
Last Run
Next Run
Recent Tasks
dataset_consumes_1
consumes
dataset-scheduled
airflow
Dataset
On s3://dag1/output_1.txt
dataset_consumes_1_and_2
consumes
dataset-scheduled
airflow
Dataset
0 of 2 datasets updated
dataset_consumes_1_never_scheduled
consumes
dataset-scheduled
airflow
Dataset
0 of 2 datasets updated
dataset_consumes_unknown_never_scheduled
dataset-scheduled
airflow
Dataset
0 of 2 datasets updated
dataset_produces_1

Airflow DAG with Bash Operator

於 dags 資料夾中建立一個.py檔



載入相關套件並定義參數

our_first_dag.py

dags > our_first_dag.py

```python
from datetime import datetime, timedelta
from airflow import DAG
from airflow.operator.bash import BashOperator

# 設定重置時間5分鐘&延遲2分鐘(每次等待重置的時間)
default_args = {
    'owner': 'coder2j',
    'retries': 5,
    'retry_delay': timedelta(minute=2)

}

```

撰寫程式碼

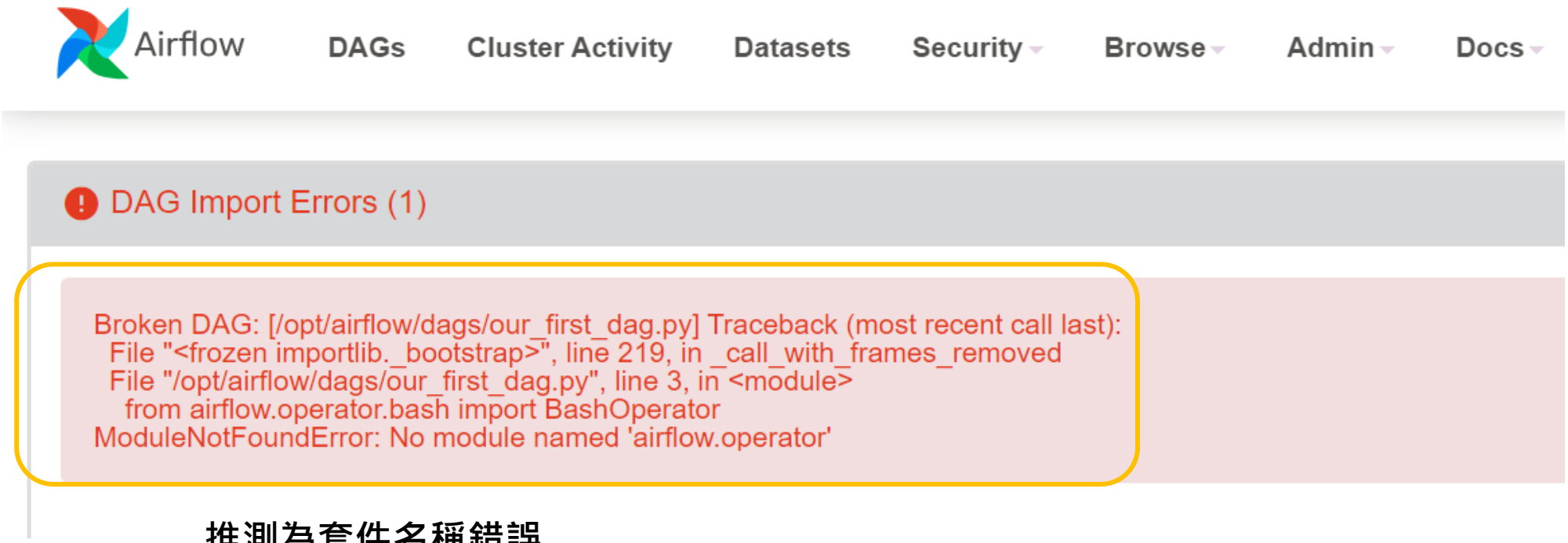
設定DAG中的相關資訊

```
13
14  with DAG(
15      dag_id='our_first_dag',
16      default_args=default_args,
17      description='This is our first dag that we write',
18      # 設定從甚麼時候啟動及按時啟動
19      # 2021年7月29日開始每天凌晨兩點啟動
20      start_date=datetime(2021, 7, 29, 2),
21      scheduler_interval='@daily'
22
23  ) as dag:
24  pass
```

使用bash執行task

```
22
23  ) as dag:
24      task1 = BashOperator(
25          task_id='first_task',
26          bash_command="echo Hello World. This is the first task!"
27      )
28
29      task1
```

執行發現有ERROR



推測為套件名稱錯誤

修正Module名稱

原本沒有+s

our_first_dag.py

dags > our_first_dag.py

```
1  from datetime import datetime, timedelta
2  from airflow import DAG
3  from airflow.operator.bash import BashOperator
```

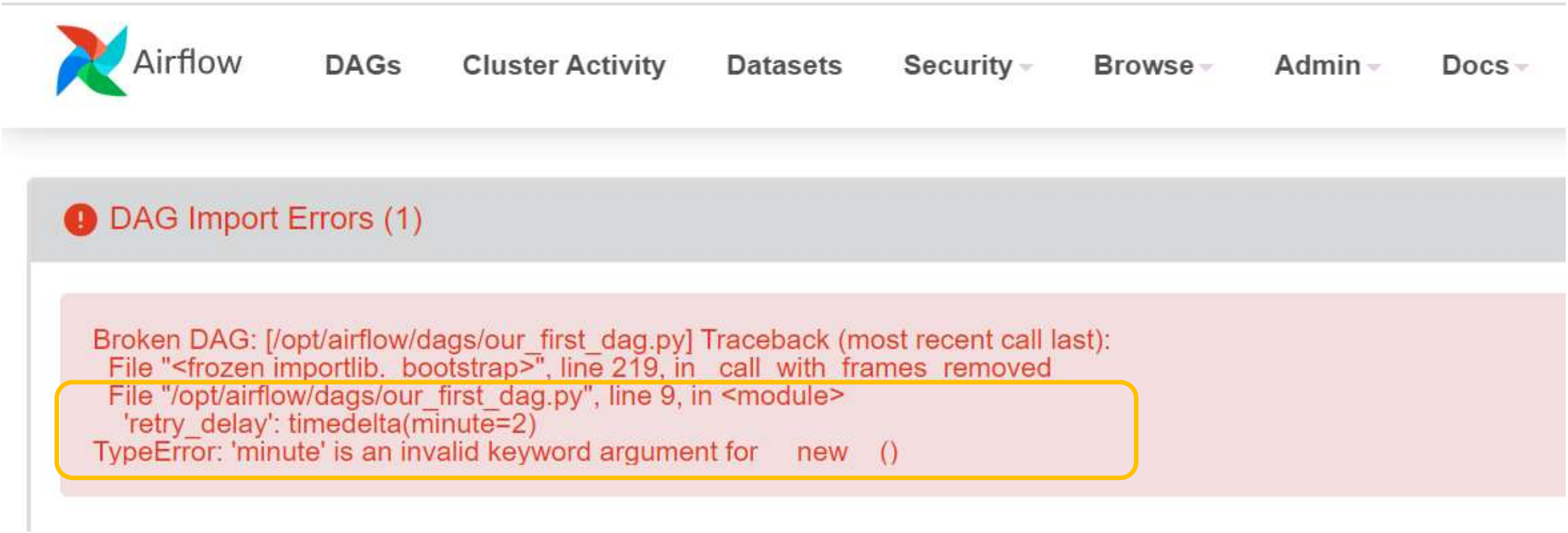
修改後

our_first_dag.py

dags > our_first_dag.py

```
1  from datetime import datetime, timedelta
2  from airflow import DAG
3  from airflow.operators.bash import BashOperator
```

timedeltea語法錯誤



修正timedelta語法

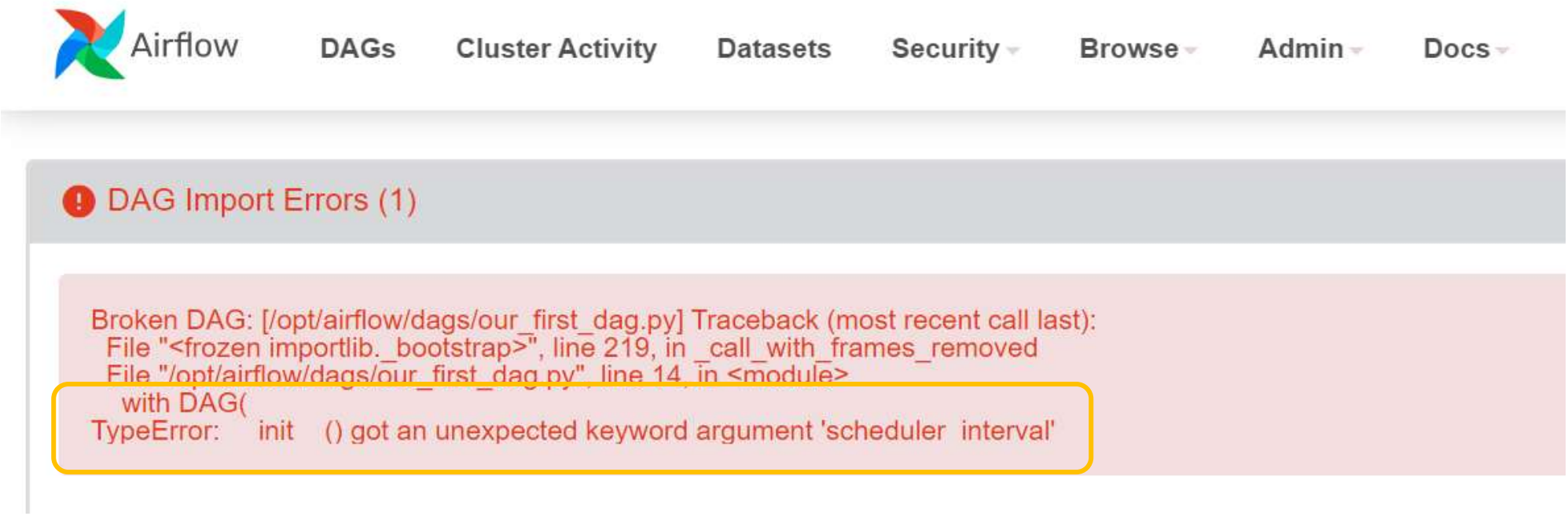
原本沒有+s

```
# 設定重置時間5分鐘&延遲2分鐘(每次等待重置的時間)
default_args = {
    'owner': 'coder2j',
    'retries': 5,
    'retry_delay': timedelta(minute=2)

}
```

修改後

```
5   # 設定重置時間5分鐘&延遲2分鐘(每次等待重置的時間)
6   default_args = {
7       'owner': 'coder2j',
8       'retries': 5,
9       'retry_delay': timedelta(minutes=2)
10
11  }
```

scheduler interval 有錯誤

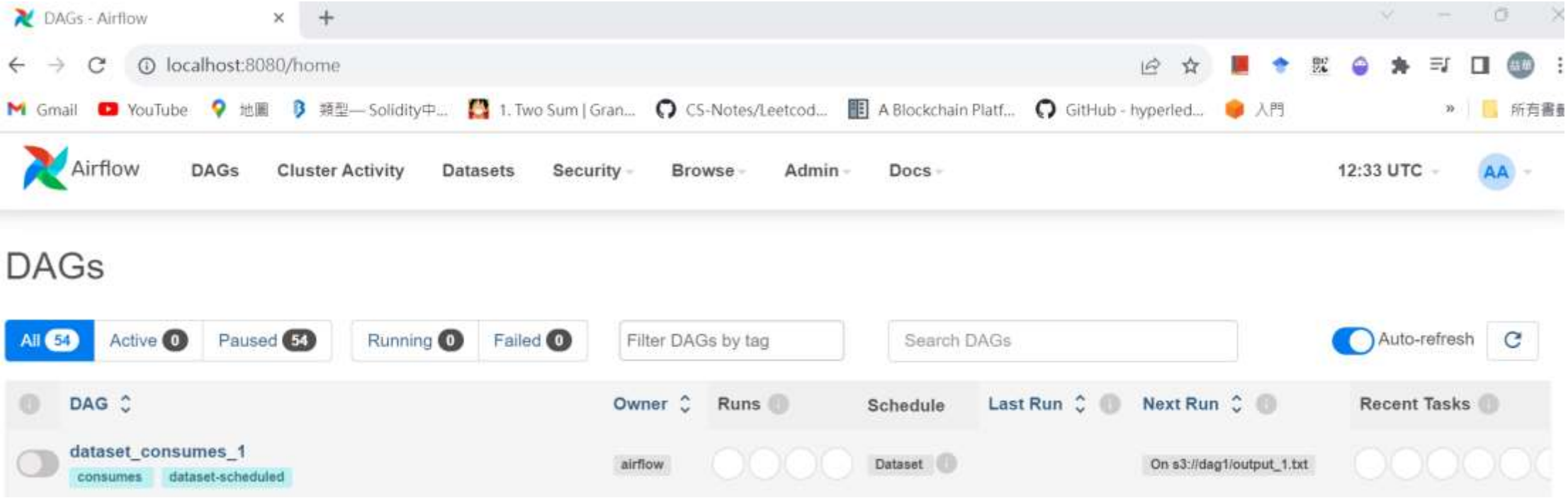


修改錯誤

```
14  with DAG(
15      dag_id='our_first_dag',
16      default_args=default_args,
17      description='This is our first dag that we w
18      # 設定從甚麼時候啟動及按時啟動
19      # 2021年7月29日開始每天凌晨兩點啟動
20      start_date=datetime(2021, 7, 29, 2),
21      scheduler_interval='@daily'
```

```
13
14  with DAG(
15      dag_id='our_first_dag',
16      default_args=default_args,
17      description='This is our first dag that we w
18      # 設定從甚麼時候啟動及按時啟動
19      # 2021年7月29日開始每天凌晨兩點啟動
20      start_date=datetime(2021, 7, 29, 2),
21      schedule_interval= @daily'
22
```

成功! 沒有錯誤訊息


DAGs - Airflow
localhost:8080/home
Airflow
DAGs
Cluster Activity
Datasets
Security
Browse
Admin
Docs
12:33 UTC
AA
DAGs
All 54
Active 0
Paused 54
Running 0
Failed 0
Filter DAGs by tag
Search DAGs
Auto-refresh
DAG
Owner
Runs
Schedule
Last Run
Next Run
Recent Tasks
dataset_consumes_1
consumes
dataset-scheduled
airflow
Dataset
On s3://dag1/output_1.txt

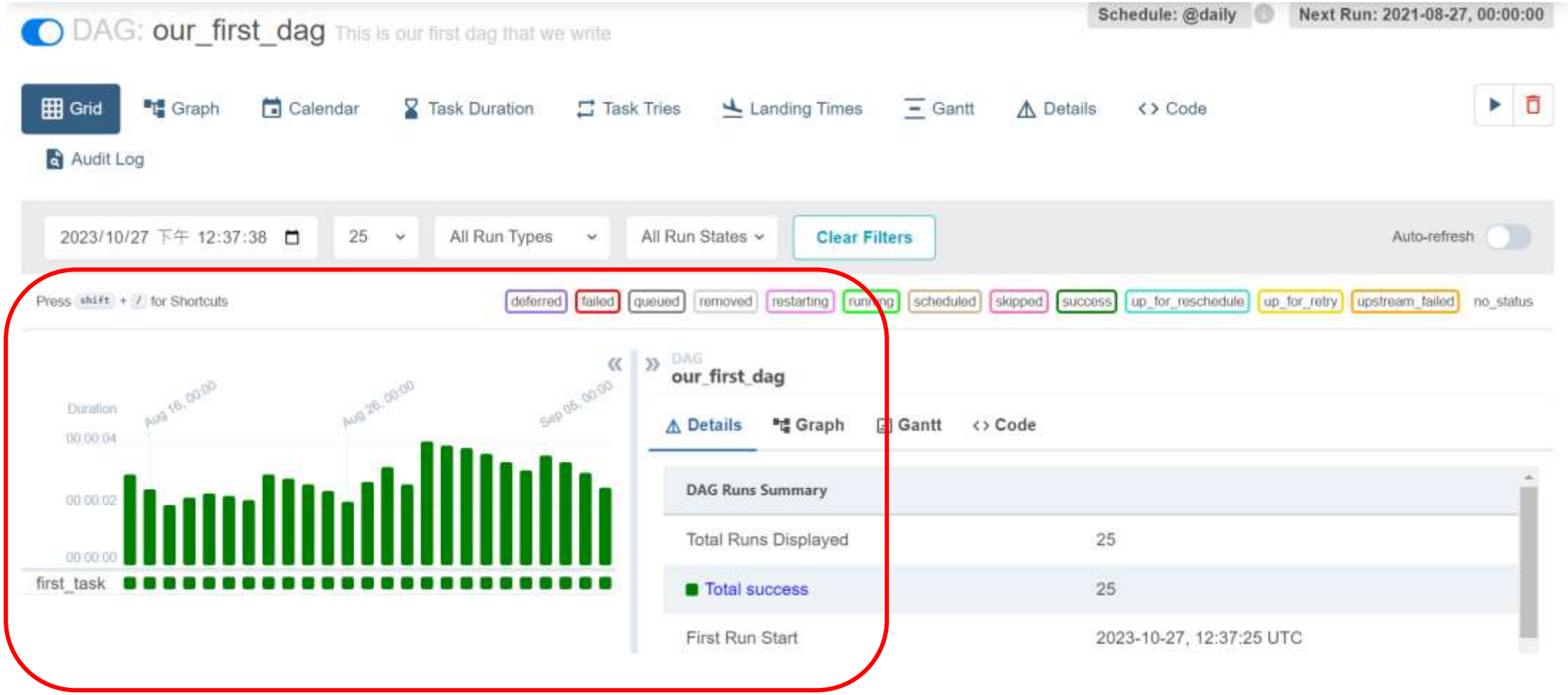
可看到我們所新增的dag

latest_only_with_trigger example3	airflow		4:00:00	2023-10-27, 08:33:23	
our_first_dag	coder2j		@daily	2021-07-30, 00:00:00	
tutorial example	airflow		1 day, 0:00:00	2023-10-26, 12:33:19	
tutorial_dag example	airflow		None		
tutorial_taskflow_api example	airflow		None		

啟動


our_first_dag
coder2j
@daily
tutorial
example
airflow
1 day, 0:00:00
tutorial_dag
example
airflow
None
tutorial_taskflow_api
example
airflow
None

綠色為成功啟動


DAG: our_first_dag
This is our first dag that we write
Schedule: @daily
Next Run: 2021-08-27, 00:00:00
Grid
Graph
Calendar
Task Duration
Task Tries
Landing Times
Gantt
Details
Code
Audit Log
2023/10/27 下午 12:37:38
25
All Run Types
All Run States
Clear Filters
Auto-refresh
Press shift + / for Shortcuts
deferred
failed
queued
removed
restarting
running
scheduled
skipped
success
up_for_reschedule
up_for_retry
upstream_failed
no_status
Duration
00:00:04
00:00:02
00:00:00
Aug 16, 00:00
Aug 26, 00:00
Sep 05, 00:00
first_task
DAG
our_first_dag
Details
Graph
Gantt
Code
DAG Runs Summary
Total Runs Displayed
25
Total success
25
First Run Start
2023-10-27, 12:37:25 UTC

新增task2

```
) as dag:
    task1 = BashOperator(
        task_id='first_task',
        bash_command="echo Hello World. This is the first task!"
    )

    task2 = BashOperator(
        task_id='second_task',
        bash_command='echo Hey, I am task2 and will be running after task1!'
    )

# 設定 task2 在 task1 後執行
task1.set_downstream(task2)
```

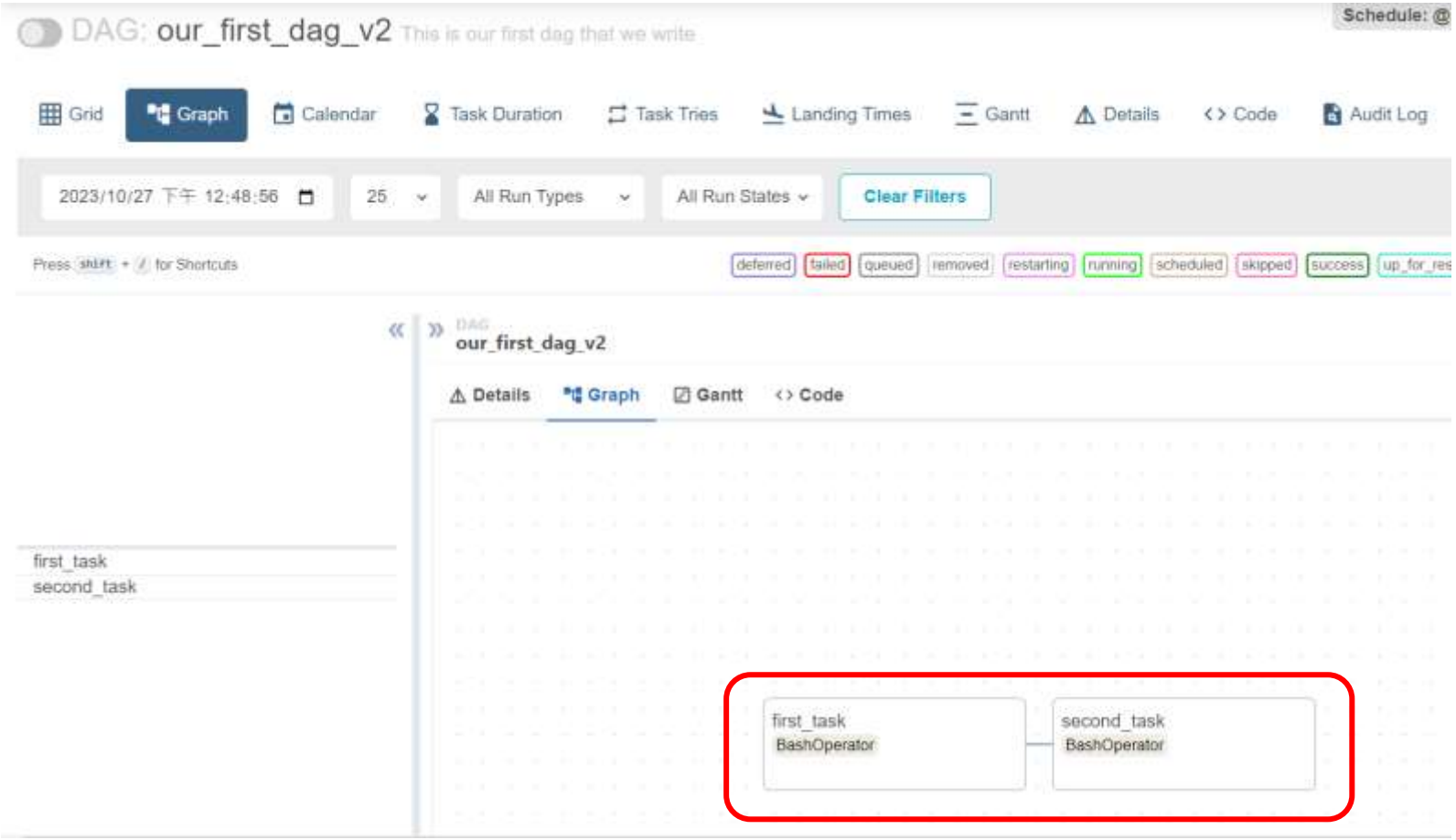
修改id避免混淆

```
14  with DAG(
15      dag_id='our_first_dag_v2',
16      default_args=default_args,
17      description='This is our first dag that we write',
18      # 設定從甚麼時候啟動及按時啟動
19      # 2021年7月29日開始每天凌晨兩點啟動
20      start_date=datetime(2021, 7, 29, 2),
21      schedule_interval='@daily'
```

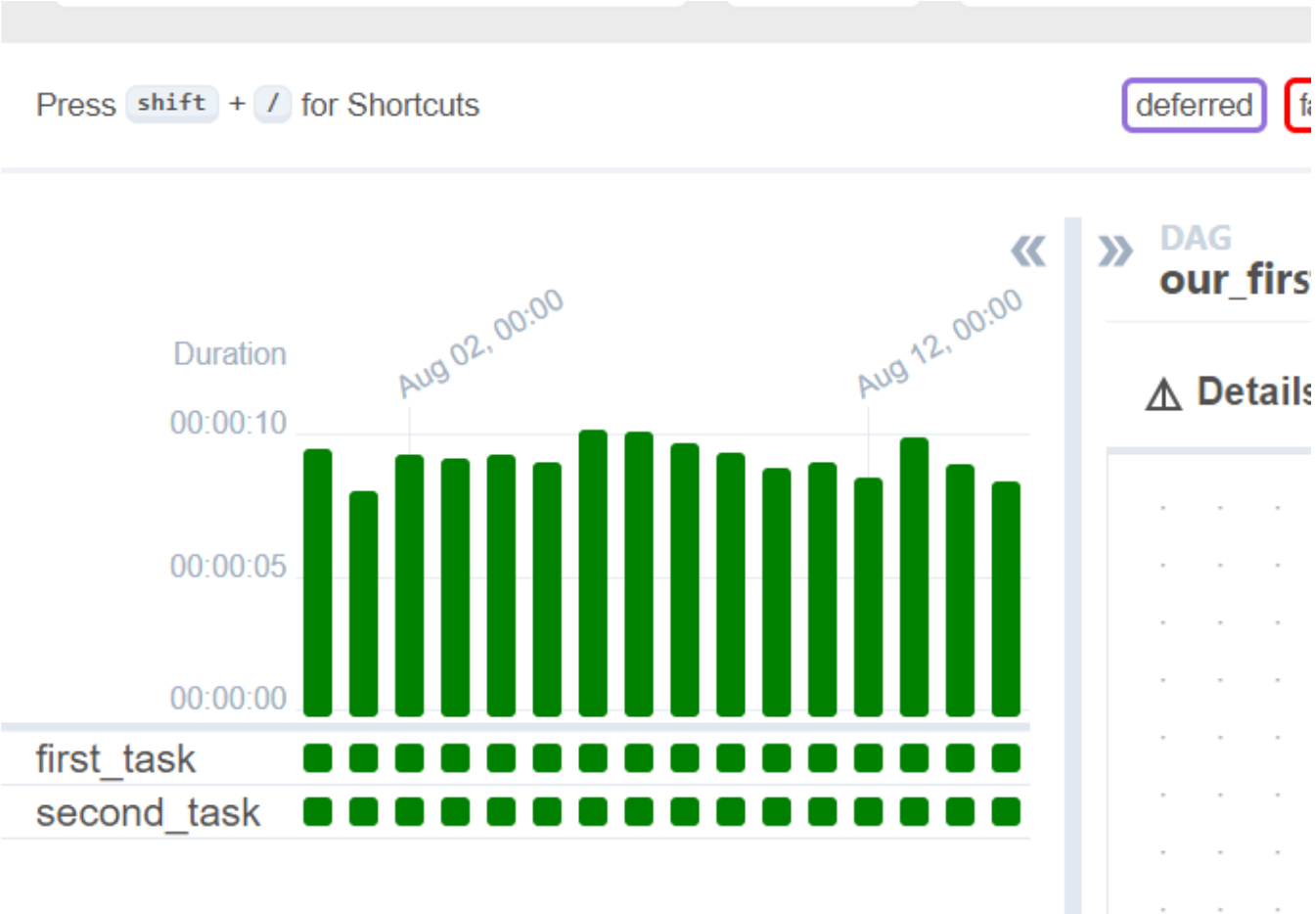
成功建立畫面


our_first_dag
coder2j
445
@daily
2022-10-17, 00:00:00
2022-10-18, 00:00:00
our_first_dag_v2
coder2j
@daily
2021-07-30, 00:00:00
tutorial
example
airflow
1 day, 0:00:00
2023-10-26, 12:47:36
tutorial_dag
example
airflow
None

task流程畫面


DAG: our_first_dag_v2
This is our first dag that we write
Schedule: @
Grid
Graph
Calendar
Task Duration
Task Tries
Landing Times
Gantt
Details
Code
Audit Log
2023/10/27 下午 12:48:56
25
All Run Types
All Run States
Clear Filters
Press shift + / for Shortcuts
deferred
failed
queued
removed
restarting
running
scheduled
skipped
success
DAG
our_first_dag_v2
Details
Graph
Gantt
Code
first_task
second_task
first_task
BashOperator
second_task
BashOperator

成功執行


Press shift + / for Shortcuts
deferred
DAG
our_firs
Details
Duration
Aug 02, 00:00
Aug 12, 00:00
00:00:10
00:00:05
00:00:00
first_task
second_task

再新增一個task3

```
task2 = BashOperator(
    task_id='second_task',
    bash_command='echo Hey, I am task2 and will be running after task1!'
)

task3 = BashOperator(
    task_id='third_task',
    bash_command='echo Hey, I am task3 and will be running after task1 at the same time as task2!'
)
```

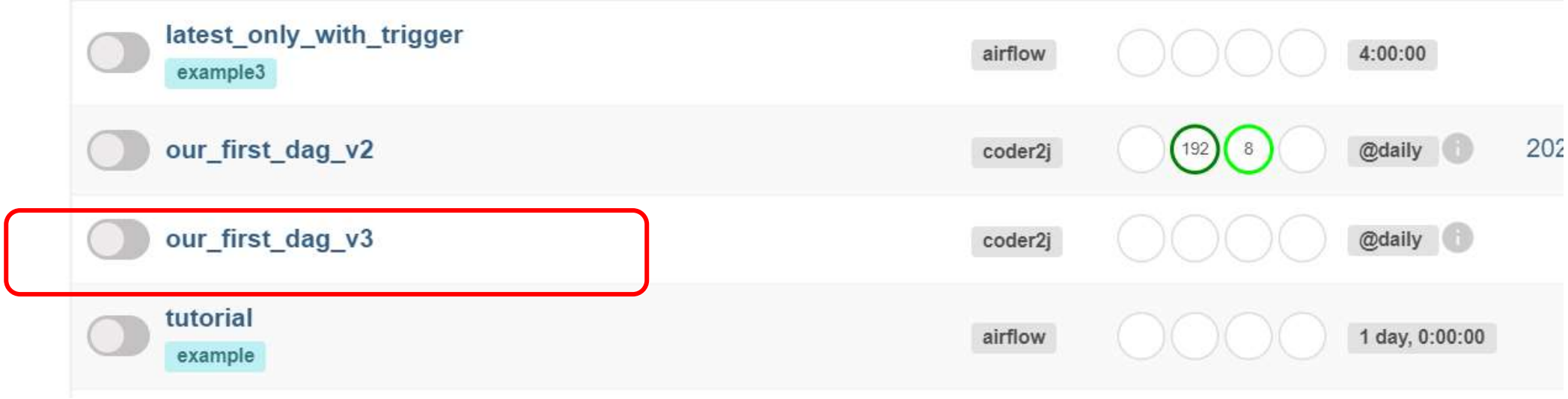
設定task執行順序，第一種方式

```
41
42    # task3 在task1後與task2同時執行
43    # 第一種方法
44    task1.set_downstream(task2)
45    task1.set_downstream(task3)
```

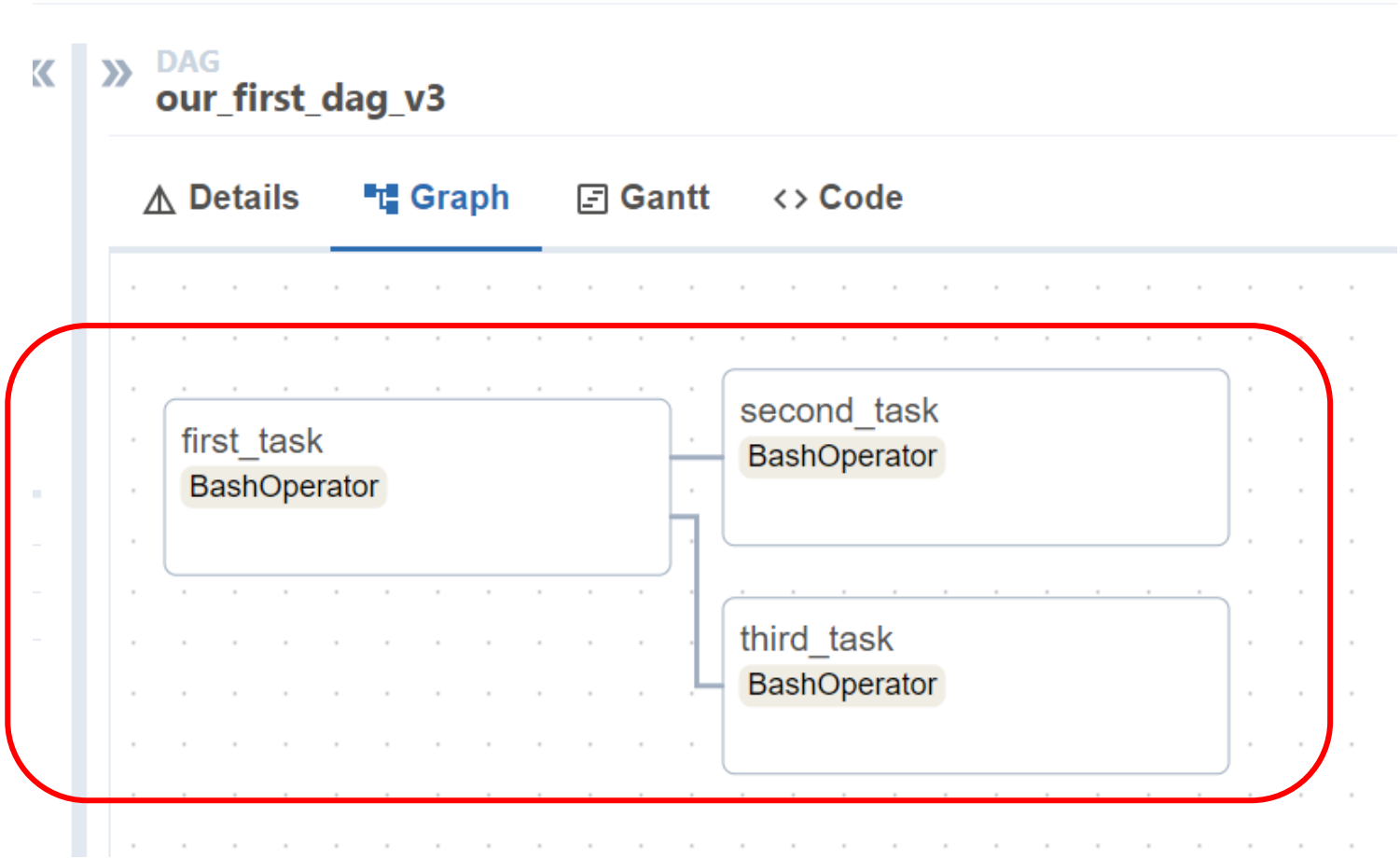
修改id避免混淆

```
with DAG(
    dag_id='our_first_dag_v3',
    default_args=default_args,
    description='This is our first dag that we write',
    # 設定從甚麼時候啟動及按時啟動
    # 2021年7月29日開始每天凌晨兩點啟動
    start_date=datetime(2021, 7, 29, 2),
    schedule_interval='@daily'
```

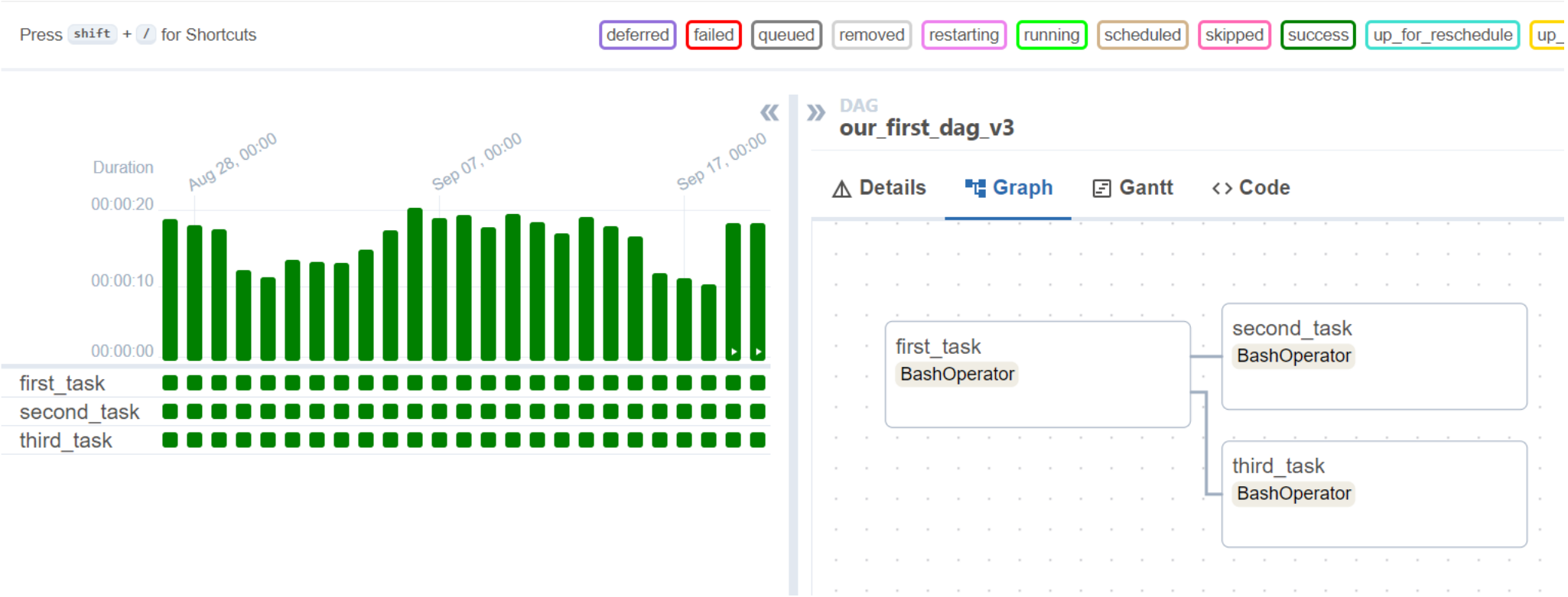
成功建立畫面


latest_only_with_trigger
example3
airflow
4:00:00
our_first_dag_v2
coder2j
192
8
@daily
202
our_first_dag_v3
coder2j
@daily
tutorial
example
airflow
1 day, 0:00:00

task流程畫面


DAG
our_first_dag_v3
Details
Graph
Gantt
Code
first_task
BashOperator
second_task
BashOperator
third_task
BashOperator

成功執行


Press shift + / for Shortcuts
deferred
failed
queued
removed
restarting
running
scheduled
skipped
success
up_for_reschedule
up_
Duration
Aug 28, 00:00
Sep 07, 00:00
Sep 17, 00:00
00:00:20
00:00:10
00:00:00
first_task
second_task
third_task
DAG
our_first_dag_v3
Details
Graph
Gantt
Code
first_task
BashOperator
second_task
BashOperator
third_task
BashOperator

設定task執行順序，第二種方式

```
# task3 在task1後與task2同時執行
# 第一種方法
# task1.set_downstream(task2)
# task1.set_downstream(task3)

# 第二種方法
task1 >> task2
task1 >> task3
```

設定task執行順序，第三種方式

```
# task3 在task1後與task2同時執行
# 第一種方法
# task1.set_downstream(task2)
# task1.set_downstream(task3)

# 第二種方法
# task1 >> task2
# task1 >> task3

# 第三種方法
task1 >> [task2, task3]
```

Airflow DAG with Python Operator

改用python操作airflow

載入airflow.python

our_first_dag.py python_operator.py

dags > python_operator.py

```python
from datetime import datetime, timedelta
from airflow import DAG
from airflow.operators.python import PythonOperator

default_args = {
    'owner': 'coder2j',
    'retries': 5,
    'retry_delay': timedelta(minutes=2)

}

```

使用python定義一個greet function

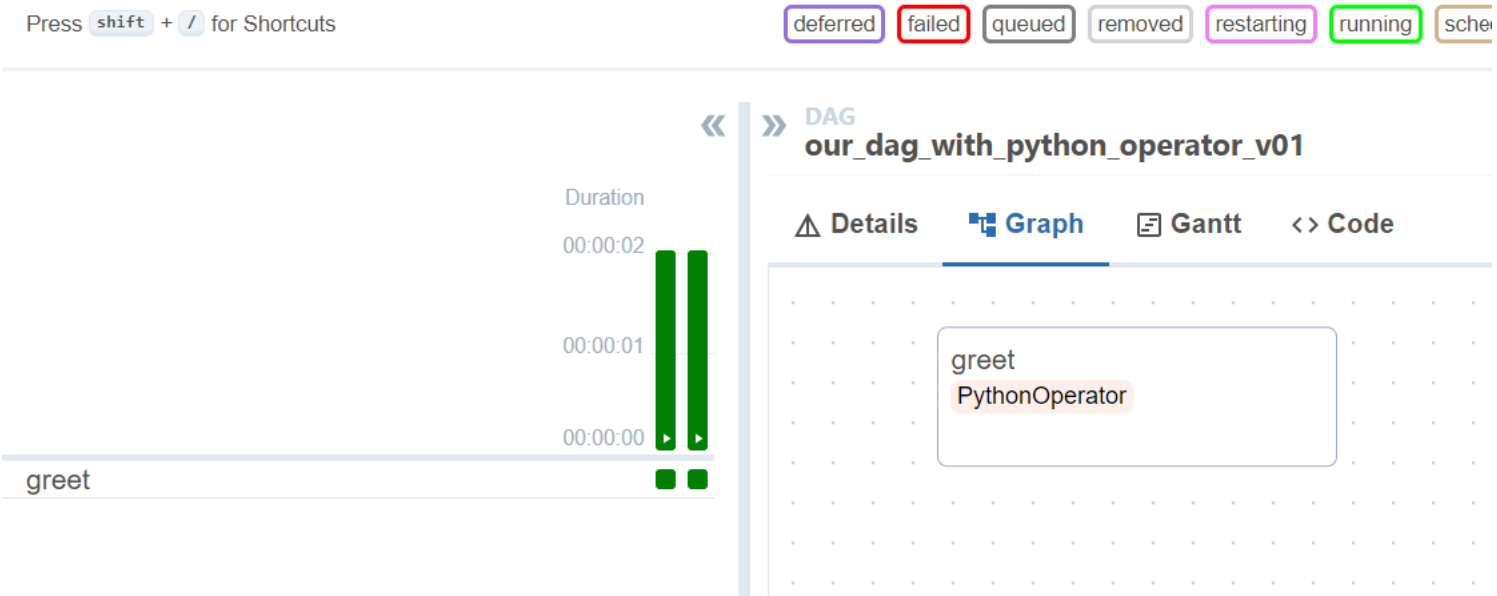
```python
def greet():
    print("Hello World!")

with DAG(
    default_args=default_args,
    dag_id='our_dag_with_python_operator_v01',
    description='Oure first dag using python operator',
    start_date=datetime(2023, 10, 28),
    schedule_interval='@daily'
) as dag:
    task1 = PythonOperator(
        task_id='greet',
        python_callable=greet
    )

task1
```

並將fun寫入task中

成功建立畫面


latest_only_with_trigger
example3
airflow
4:00:00
our_dag_with_python_operator_v01
coder2j
@daily
our_first_dag_v5
coder2j
@daily
tutorial
example
airflow
1 day, 0:00:00

成功執行


Press shift + / for Shortcuts
deferred
failed
queued
removed
restarting
running
Duration
00:00:02
00:00:01
00:00:00
greet
DAG
our_dag_with_python_operator_v01
Details
Graph
Gantt
Code
greet
PythonOperator

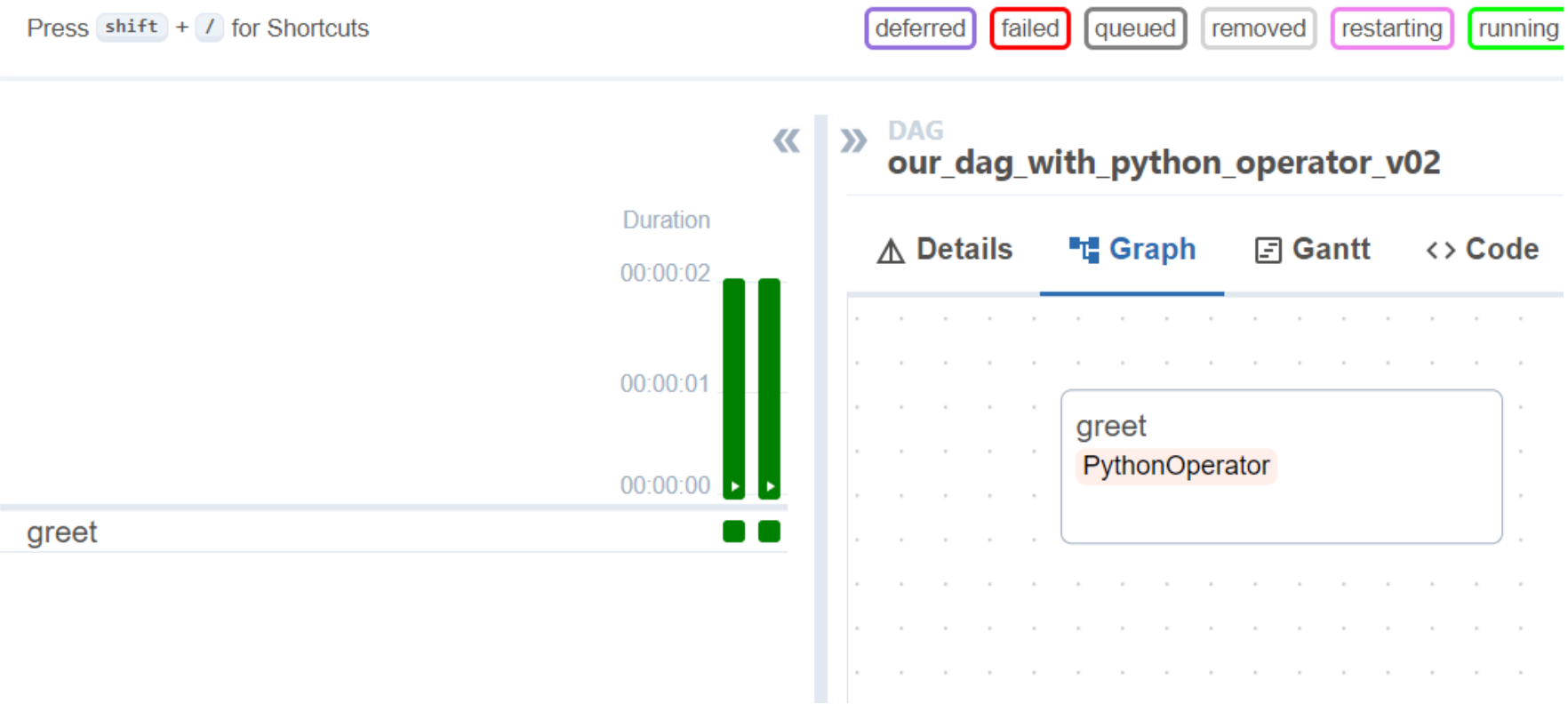
將greet()修改為回傳名字、年齡

```python
def greet(name, age):
    print(f"Hello World! My name is {name},"
          f"and I am {age} years old!")

with DAG(
    default_args=default_args,
    dag_id='our_dag_with_python_operator_v02',
    description='Our first dag using python operator',
    start_date=datetime(2023, 10, 28),
    schedule_interval='@daily'
) as dag:
    task1 = PythonOperator(
        task_id='greet',
        python_callable=greet,
        op_kwargs={'name': 'Tom', 'age': 20}
    )

task1
```

成功建立畫面


latest_only_with_trigger
example3
airflow
our_dag_with_python_operator_v02
coder2j
our_first_dag_v5
coder2j
tutorial
example
airflow

成功執行


Press shift + / for Shortcuts
deferred
failed
queued
removed
restarting
running
Duration
00:00:02
00:00:01
00:00:00
greet
DAG
our_dag_with_python_operator_v02
Details
Graph
Gantt
Code
greet
PythonOperator

Data Sharing via Airflow Xcoms

說明

- XComs 是 Airflow 中的一種機制，用於讓不同的 Task 之間進行通信。
- 預設情況下，Task 是完全隔離的，可能在完全不同的機器上運行。XComs 由一個 key（基本上是它的名稱），以及它來自的 task_id 和 dag_id 來識別。
- 它們可以有任何（可序列化）值，但它們只設計用於小量數據；
- 不要使用它們來傳遞大量數據

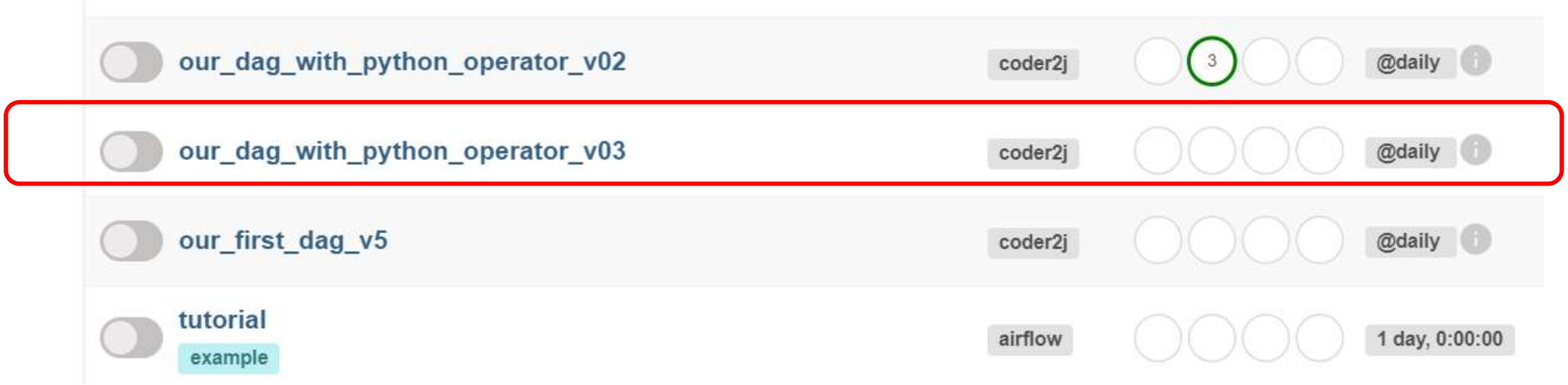
新增一個 get_name() function

```
12  def greet(name, age):
13      print(f"Hello World! My name is {name},"
14            f"and I am {age} years old!")
15
16  def get_name():
17      return 'Jerry'
```

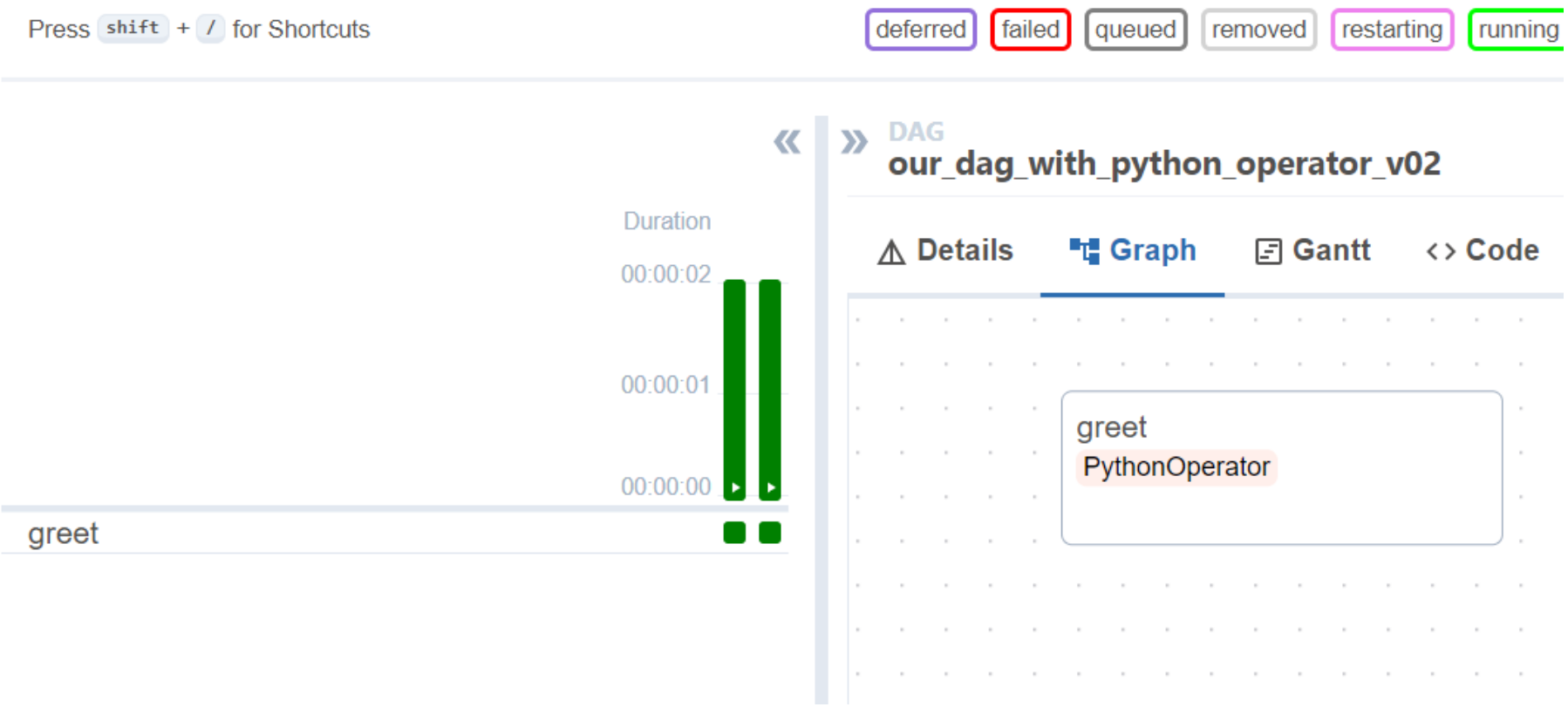
新增task2並寫入 get_name function

```python
with DAG(
    default_args=default_args,
    dag_id='our_dag_with_python_operator_v03',
    description='Our first dag using python operator',
    start_date=datetime(2023, 10, 28),
    schedule_interval='@daily'
) as dag:
    # task1 = PythonOperator(
    #     task_id='greet',
    #     python_callable=greet,
    #     op_kwargs={'name': 'Tom', 'age': 20}
    # )

    task2 = PythonOperator(
        task_id='get_name',
        python_callable=get_name
    )

# task1
task2
```

成功建立畫面


our_dag_with_python_operator_v02
coder2j
3
@daily
our_dag_with_python_operator_v03
coder2j
@daily
our_first_dag_v5
coder2j
@daily
tutorial
example
airflow
1 day, 0:00:00

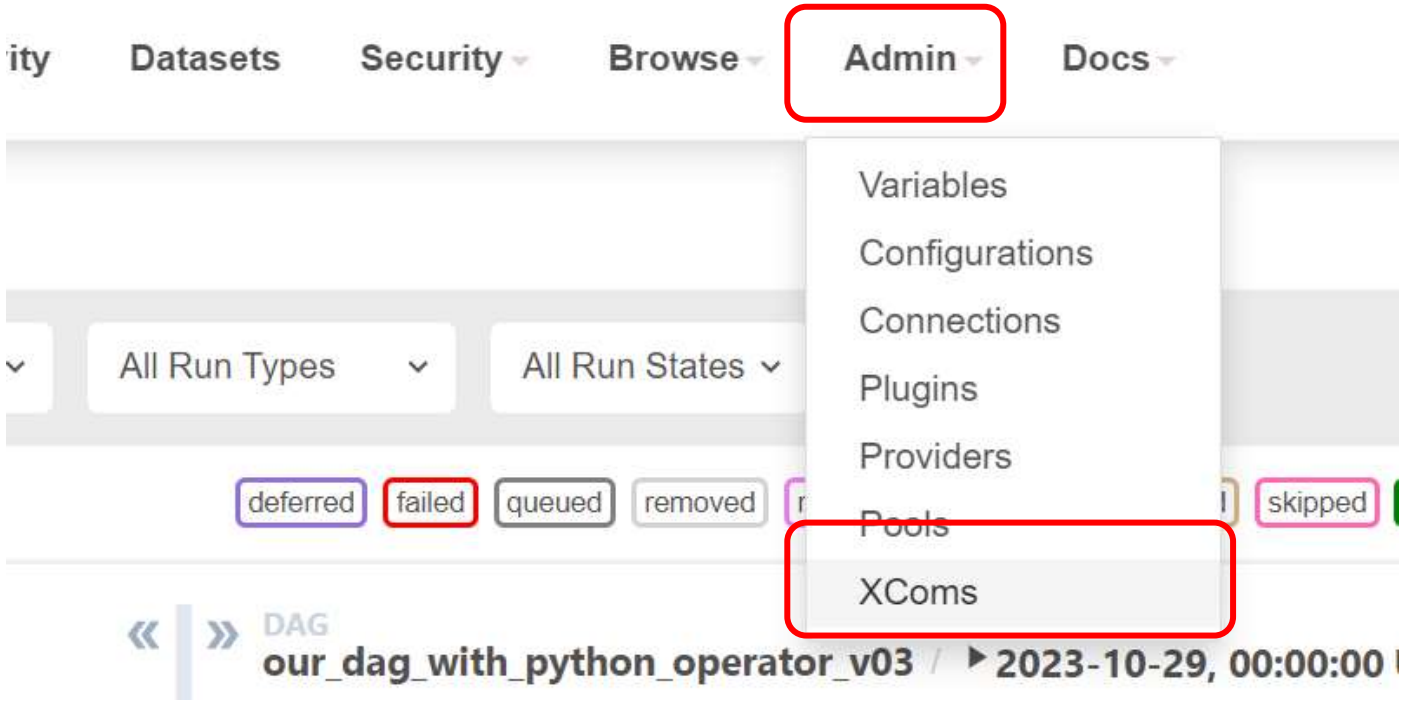
成功執行


Press shift + / for Shortcuts
deferred
failed
queued
removed
restarting
running
Duration
00:00:02
00:00:01
00:00:00
greet
DAG
our_dag_with_python_operator_v02
Details
Graph
Gantt
Code
greet
PythonOperator

查看log確實回傳Jerry

```
[2023-10-29, 10:43:14 UTC] {standard_task_runner.py:84} INFO - Running: ['***', 'tasks', 'run', 'ou
[2023-10-29, 10:43:14 UTC] {standard_task_runner.py:85} INFO - Job 1222: Subtask get_name
[2023-10-29, 10:43:14 UTC] {task_command.py:416} INFO - Running <TaskInstance: our_dag_with_python_
[2023-10-29, 10:43:14 UTC] {taskinstance.py:1662} INFO - Exporting env vars: AIRFLOW_CTX_DAG_OWNER=
[2023-10-29, 10:43:14 UTC] {python.py:194} INFO - Done. Returned value was: Jerry
[2023-10-29, 10:43:14 UTC] {taskinstance.py:1400} INFO - Marking task as SUCCESS. dag_id=our_dag_wi
[2023-10-29, 10:43:14 UTC] {local_task_job_runner.py:228} INFO - Task exited with return code 0
[2023-10-29, 10:43:14 UTC] {taskinstance.py:2778} INFO - 0 downstream tasks scheduled from follow-c
```

查看Xcoms


ity
Datasets
Security
Browse
Admin
Docs
Variables
Configurations
Connections
Plugins
Providers
Pools
XComs
All Run Types
All Run States
deferred
failed
queued
removed
skipped
DAG
our_dag_with_python_operator_v03
2023-10-29, 00:00:00

	Key	Value	Timestamp	Dag Id	Task Id
	return_value	Jerry	2023-10-29, 10:43:14	our_dag_with_python_operator_v03	get_name
	return_value	Jerry	2023-10-29, 10:43:14	our_dag_with_python_operator_v03	get_name
	return_value	Jerry	2023-10-29, 10:43:13	our_dag_with_python_operator_v03	get_name
		Hello			

將greet修改為 ti.xcoms_pull

也就說ti.xcom_pull會回傳get_name()中的值

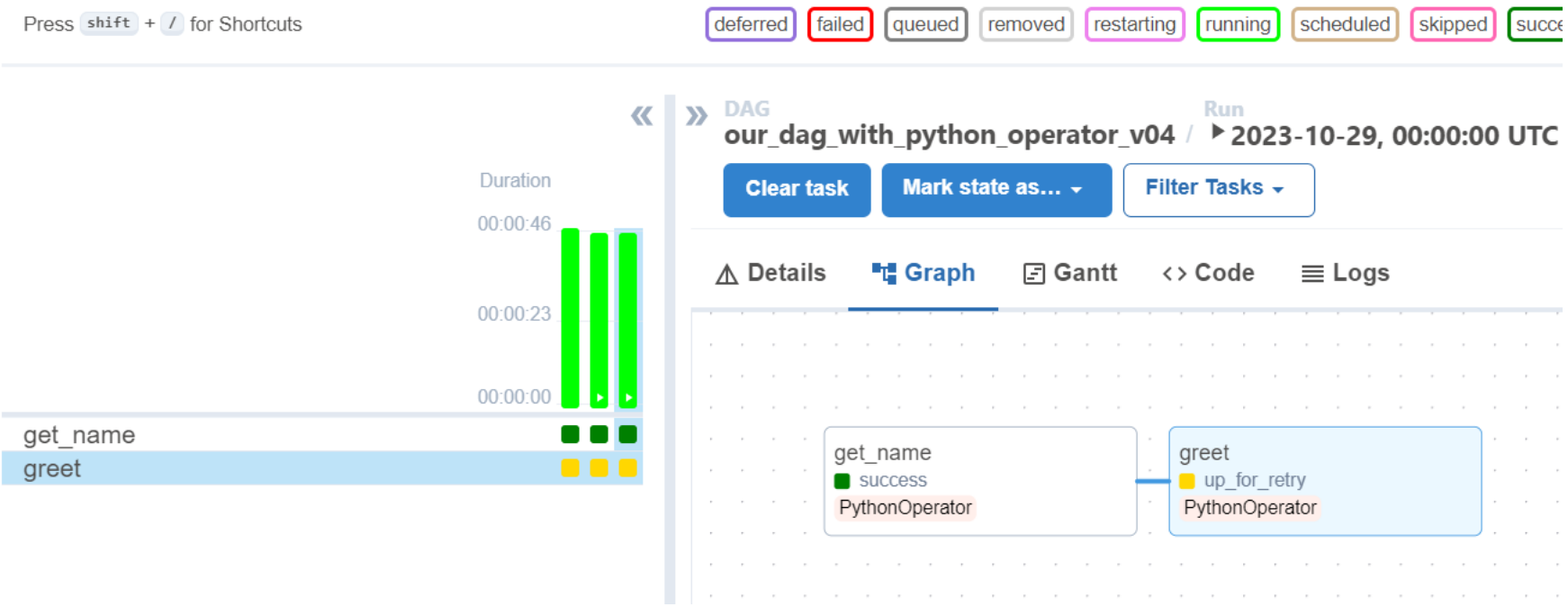
```
12  def greet(age, ti):
13      name = ti.xcoms_pull(task_ids='get_name')
14      print(f"Hello World! My name is {name},"
15            f"and I am {age} years old!")
16
17  def get_name():
18      return 'Jerry'
19
```

```python
with DAG(
    default_args=default_args,
    dag_id='our_dag_with_python_operator_v04',
    description='Our first dag using python operator',
    start_date=datetime(2023, 10, 28),
    schedule_interval='@daily'
) as dag:
    task1 = PythonOperator(
        task_id='greet',
        python_callable=greet,
        op_kwargs={'age': 20}
    )

    task2 = PythonOperator(
        task_id='get_name',
        python_callable=get_name
    )

# task1
task2 >> task1
```

成功建立畫面


latest_only_with_trigger
example3
airflow
our_dag_with_python_operator_v03
coder2j
3
our_dag_with_python_operator_v04
coder2j
our_first_dag_v5
coder2j

執行發現error


Press shift + / for Shortcuts
deferred
failed
queued
removed
restarting
running
scheduled
skipped
succe
DAG
our_dag_with_python_operator_v04
Run
2023-10-29, 00:00:00 UTC
Clear task
Mark state as...
Filter Tasks
Duration
00:00:46
00:00:23
00:00:00
get_name
greet
Details
Graph
Gantt
Code
Logs
get_name
success
PythonOperator
greet
up_for_retry
PythonOperator

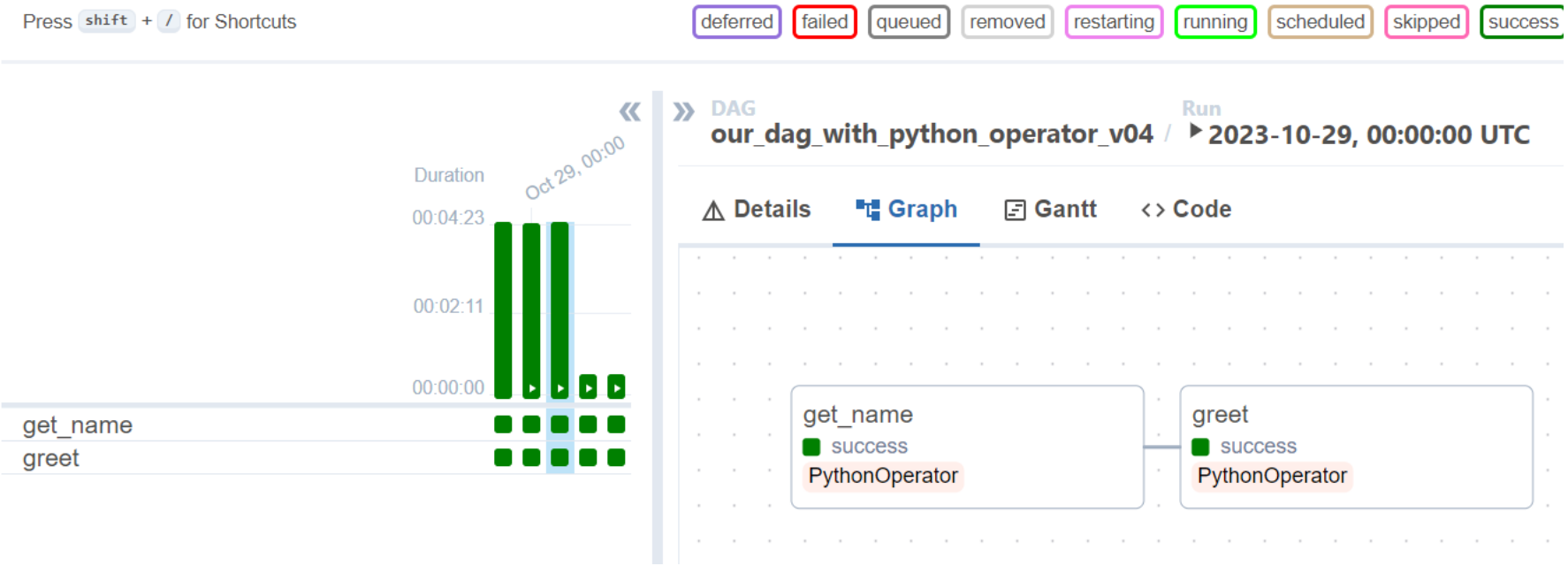
推測 xcoms_pull 語法錯誤

```
[2023-10-29, 10:53:22 UTC] {taskinstance.py:1662} INFO - Exporting env vars: AIRFLOW_CTX_DAG_OWNER='coder2j' AIRFLOW_C
[2023-10-29, 10:53:22 UTC] {taskinstance.py:1937} ERROR - Task failed with exception
Traceback (most recent call last):
  File "/home/airflow/.local/lib/python3.8/site-packages/airflow/operators/python.py", line 192, in execute
    return_value = self.execute_callable()
  File "/home/airflow/.local/lib/python3.8/site-packages/airflow/operators/python.py", line 209, in execute_callable
    return self.python_callable(*self.op_args, **self.op_kwargs)
  File "/opt/airflow/dags/python_operator.py", line 13, in greet
    name = ti.xcoms_pull(task_ids='get_name')
AttributeError: 'TaskInstance' object has no attribute 'xcoms_pull'
[2023-10-29, 10:53:22 UTC] {taskinstance.py:1400} INFO - Marking task as UP_FOR_RETRY. dag_id=our_dag_with_python_oper
[2023-10-29, 10:53:22 UTC] {standard_task_runner.py:104} ERROR - Failed to execute job 1227 for task greet ('TaskInsta
```

Xcom後面沒有s

```
10    }
11
12    def greet(age, ti):
13        name = ti.xcom_pull(task_ids='get_name')
14        print(f"Hello World! My name is {name},"
15              f"and I am {age} years old!")
16
17    def get_name():
18        return 'Jerry'
```

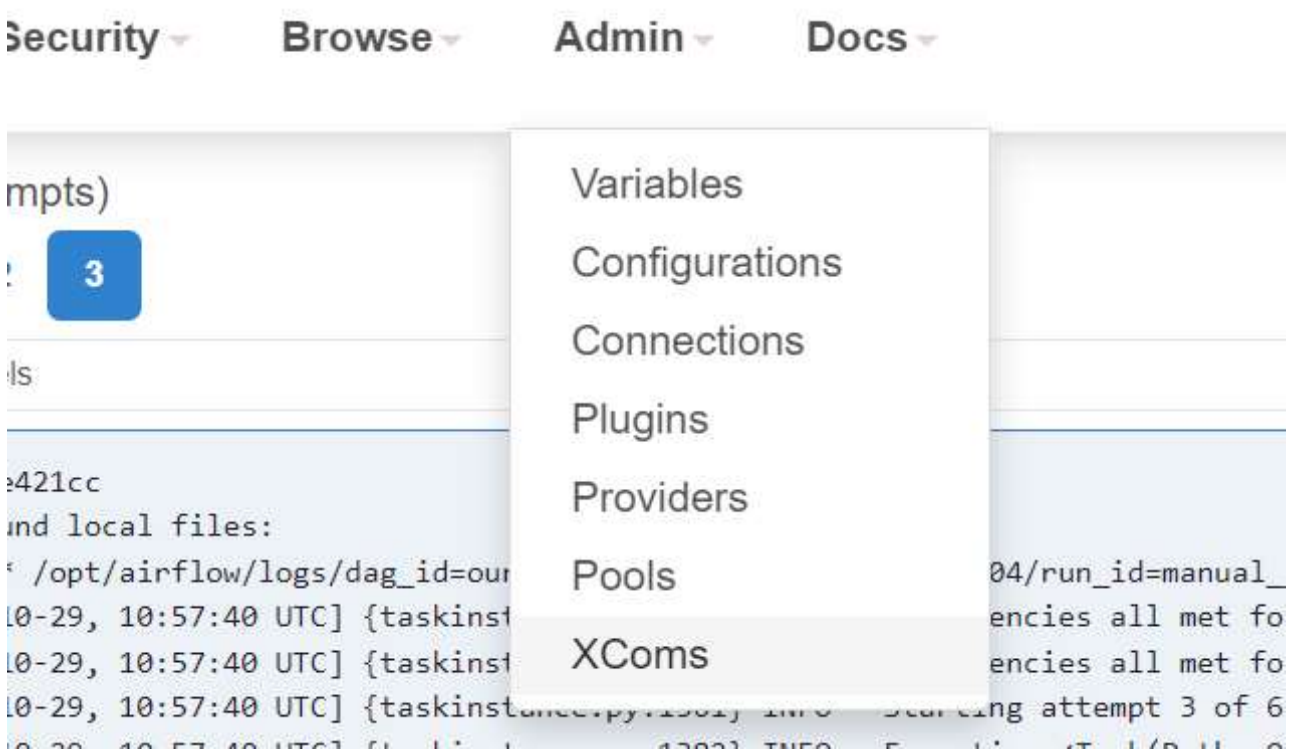
成功執行


Press shift + / for Shortcuts
deferred
failed
queued
removed
restarting
running
scheduled
skipped
success
Duration
Oct 29, 00:00
00:04:23
00:02:11
00:00:00
get_name
greet
DAG
our_dag_with_python_operator_v04
Run
2023-10-29, 00:00:00 UTC
Details
Graph
Gantt
Code
get_name
success
PythonOperator
greet
success
PythonOperator

查看logs

```
[2023-10-29, 10:57:40 UTC] {standard_task_runner.py:85} INFO - Job 1254: Subtask greet
[2023-10-29, 10:57:40 UTC] {task_command.py:416} INFO - Running <TaskInstance: our_dag_with_python_operator_v04.greet
[2023-10-29, 10:57:41 UTC] {taskinstance.py:1662} INFO - Exporting env vars: AIRFLOW_CTX_DAG_OWNER='coder2j' AIRFLOW_C
[2023-10-29, 10:57:41 UTC] {logging_mixin.py:151} INFO - Hello World! My name is Jerry,and I am 20 years old!
[2023-10-29, 10:57:41 UTC] {python.py:194} INFO - Done. Returned value was: None
[2023-10-29, 10:57:41 UTC] {taskinstance.py:1400} INFO - Marking task as SUCCESS. dag_id=our_dag_with_python_operator_
[2023-10-29, 10:57:41 UTC] {local_task_job_runner.py:228} INFO - Task exited with return code 0
[2023-10-29, 10:57:41 UTC] {taskinstance.py:2778} INFO - 0 downstream tasks scheduled from follow-on schedule check
```

查看Xcoms


Security
Browse
Admin
Docs
mpts)
3
Variables
Configurations
Connections
Plugins
Providers
Pools
XComs
421cc
nd local files:
/opt/airflow/logs/dag_id=our
04/run_id=manual_
10-29, 10:57:40 UTC] {taskinst
encies all met fo
10-29, 10:57:40 UTC] {taskinst
encies all met fo
10-29, 10:57:40 UTC] {taskinst
ng attempt 3 of 6

« < 1 2 3 4 5 6 7 > » Page size Actions

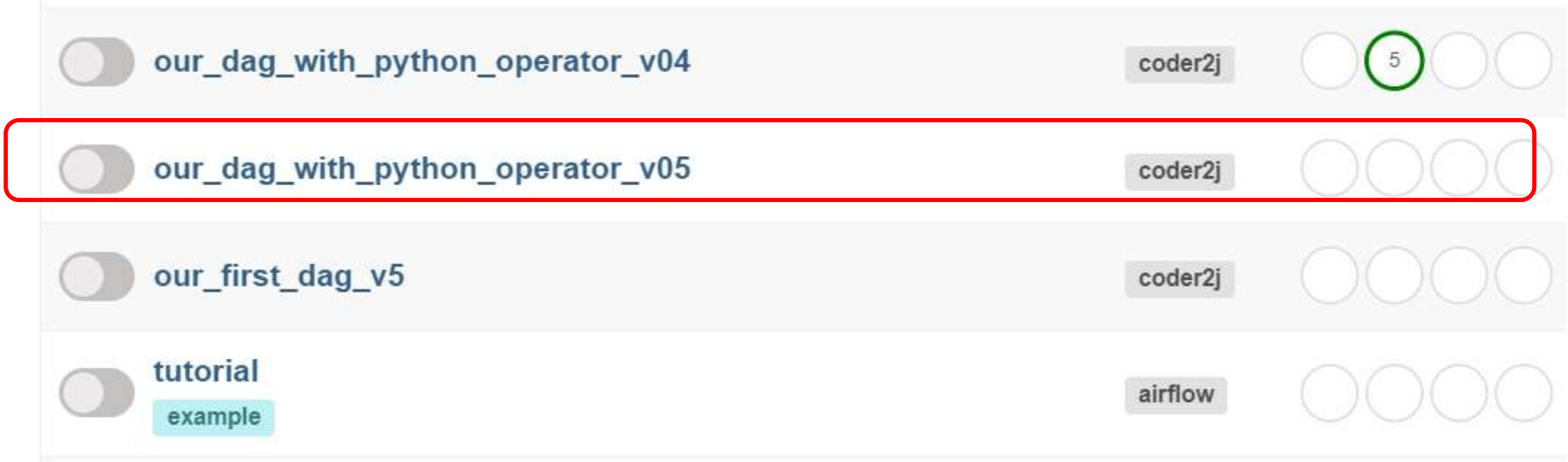
	Key	Value	Timestamp	Dag Id	Task Id
	return_value	Jerry	2023-10-29, 10:57:42	our_dag_with_python_operator_v04	get_name
	return_value	Jerry	2023-10-29, 10:57:42	our_dag_with_python_operator_v04	get_name
	return_value	Jerry	2023-10-29, 10:53:20	our_dag_with_python_operator_v04	get_name

Xcom push 多個值

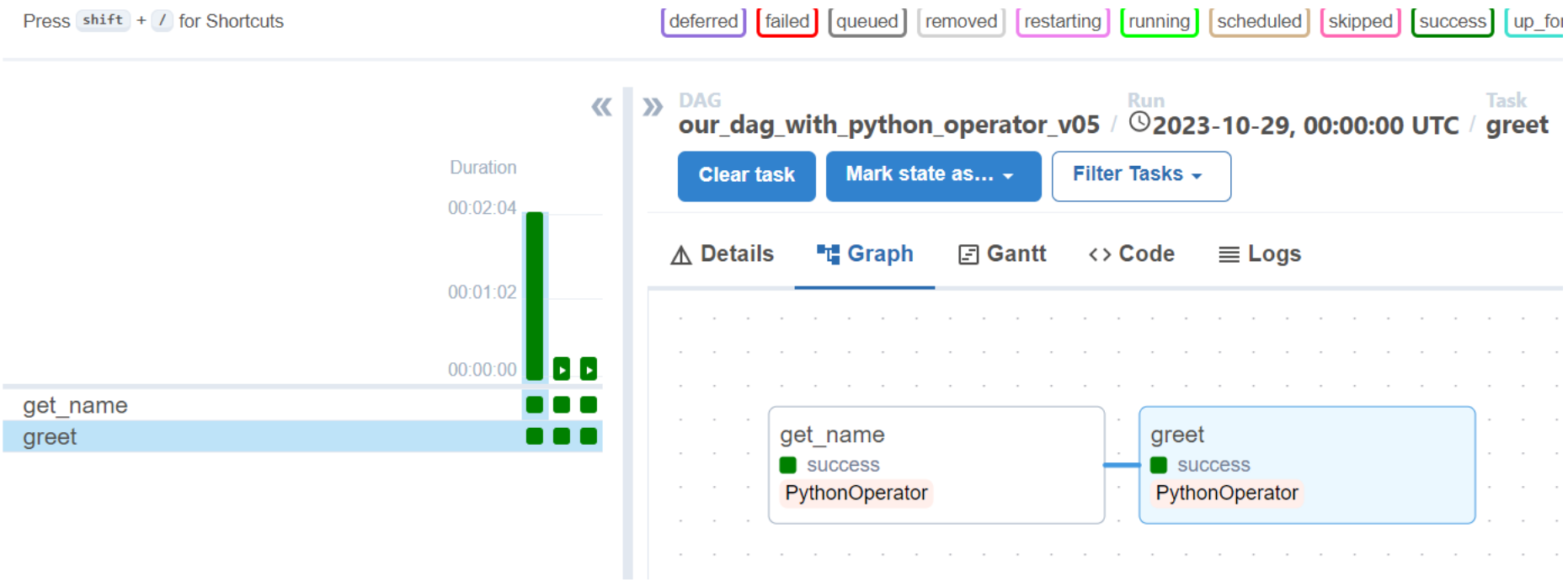
```python
def greet(age, ti):
    first_name = ti.xcom_pull(task_ids='get_name', key='first_name')
    last_name = ti.xcom_pull(task_ids='get_name', key='last_name')
    print(f"Hello World! My name is {first_name} {last_name},"
          f"and I am {age} years old!")

def get_name(ti):
    ti.xcom_push(key='first_name', value='Jerry')
    ti.xcom_push(key='last_name', value='Fridman')
```

```python
with DAG(
    default_args=default_args,
    dag_id='our_dag_with_python_operator_v05',
    description='Our first dag using python operator',
    start_date=datetime(2023, 10, 28),
    schedule_interval='@daily'
) as dag:
    task1 = PythonOperator(
        task_id='greet',
        python_callable=greet,
        op_kwargs={'age': 20}
    )

    task2 = PythonOperator(
        task_id='get_name',
        python_callable=get_name
    )

# task1
task2 >> task1
```

成功建立畫面


our_dag_with_python_operator_v04
coder2j
5
our_dag_with_python_operator_v05
coder2j
our_first_dag_v5
coder2j
tutorial
example
airflow

成功執行


Press shift + / for Shortcuts
deferred
failed
queued
removed
restarting
running
scheduled
skipped
success
DAG
our_dag_with_python_operator_v05
Run
2023-10-29, 00:00:00 UTC
Task
greet
Clear task
Mark state as...
Filter Tasks
Duration
00:02:04
00:01:02
00:00:00
get_name
greet
Details
Graph
Gantt
Code
Logs
get_name
success
PythonOperator
greet
success
PythonOperator

查看logs

```
[2023-10-29, 11:12:27 UTC] {standard_task_runner.py:85} INFO - Job 1245: Subtask greet
[2023-10-29, 11:12:27 UTC] {task_command.py:416} INFO - Running <TaskInstance: our_dag_with_python_operator_v05.greet scheduled__2
[2023-10-29, 11:12:27 UTC] {taskinstance.py:1662} INFO - Exporting env vars: AIRFLOW_CTX_DAG_OWNER='coder2j' AIRFLOW_CTX_DAG_ID='o
[2023-10-29, 11:12:27 UTC] {logging_mixin.py:151} INFO - Hello World! My name is Jerry Fridman,and I am 20 years old!
[2023-10-29, 11:12:27 UTC] {python.py:194} INFO - Done. Returned value was: None
[2023-10-29, 11:12:27 UTC] {taskinstance.py:1400} INFO - Marking task as SUCCESS. dag_id=our_dag_with_python_operator_v05, task_id
[2023-10-29, 11:12:27 UTC] {local_task_job_runner.py:228} INFO - Task exited with return code 0
[2023-10-29, 11:12:27 UTC] {taskinstance.py:2778} INFO - 0 downstream tasks scheduled from follow-on schedule check
```

查看xcom

« < 1 2 3 4 5 6 7 > » Page size Actions

	Key	Value	Timestamp	Dag Id	Task Id
	last_name	Fridman	2023-10-29, 11:11:55	our_dag_with_python_operator_v05	get_name
	first_name	Jerry	2023-10-29, 11:11:55	our_dag_with_python_operator_v05	get_name
	last_name	Fridman	2023-10-29, 11:11:55	our_dag_with_python_operator_v05	get_name
	first_name	Jerry	2023-10-29, 11:11:55	our_dag_with_python_operator_v05	get_name

新增 get_age() function

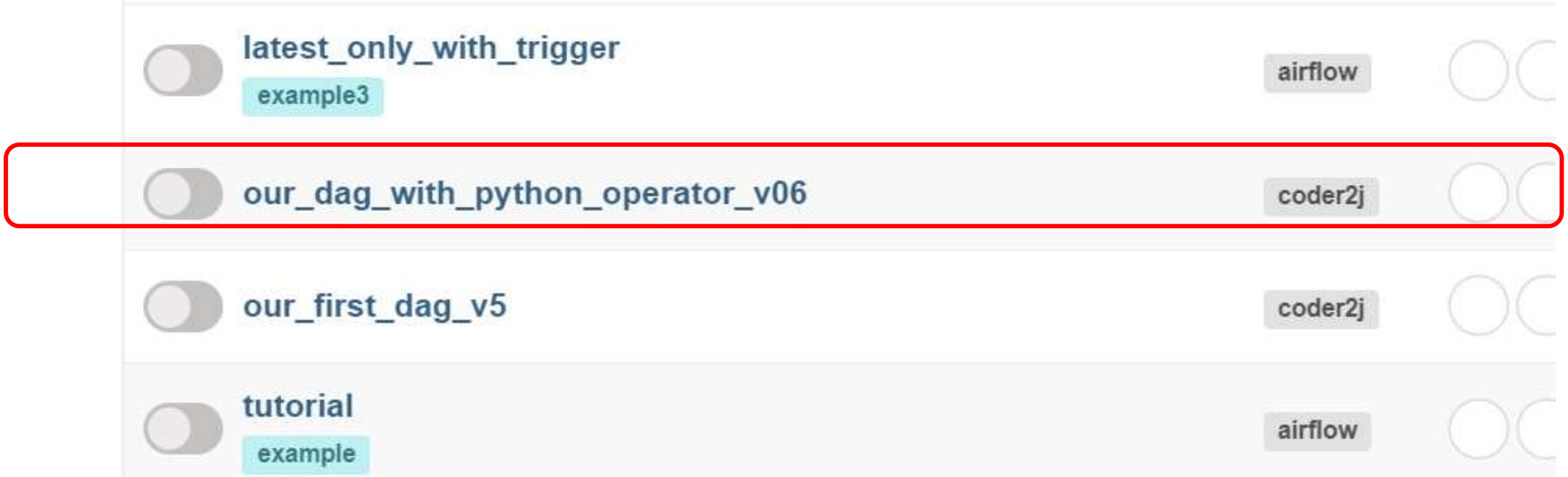
```python
def greet(ti):
    first_name = ti.xcom_pull(task_ids='get_name', key='first_name')
    last_name = ti.xcom_pull(task_ids='get_name', key='last_name')
    age = ti.xcom_pull(task_ids='get_age', key='age')
    print(f"Hello World! My name is {first_name} {last_name},"
            f"and I am {age} years old!")

def get_name(ti):
    ti.xcom_push(key='first_name', value='Jerry')
    ti.xcom_push(key='last_name', value='Fridman')

def get_age(ti):
    ti.xcom_push(key='age', value=19)
```

修改id避免混淆

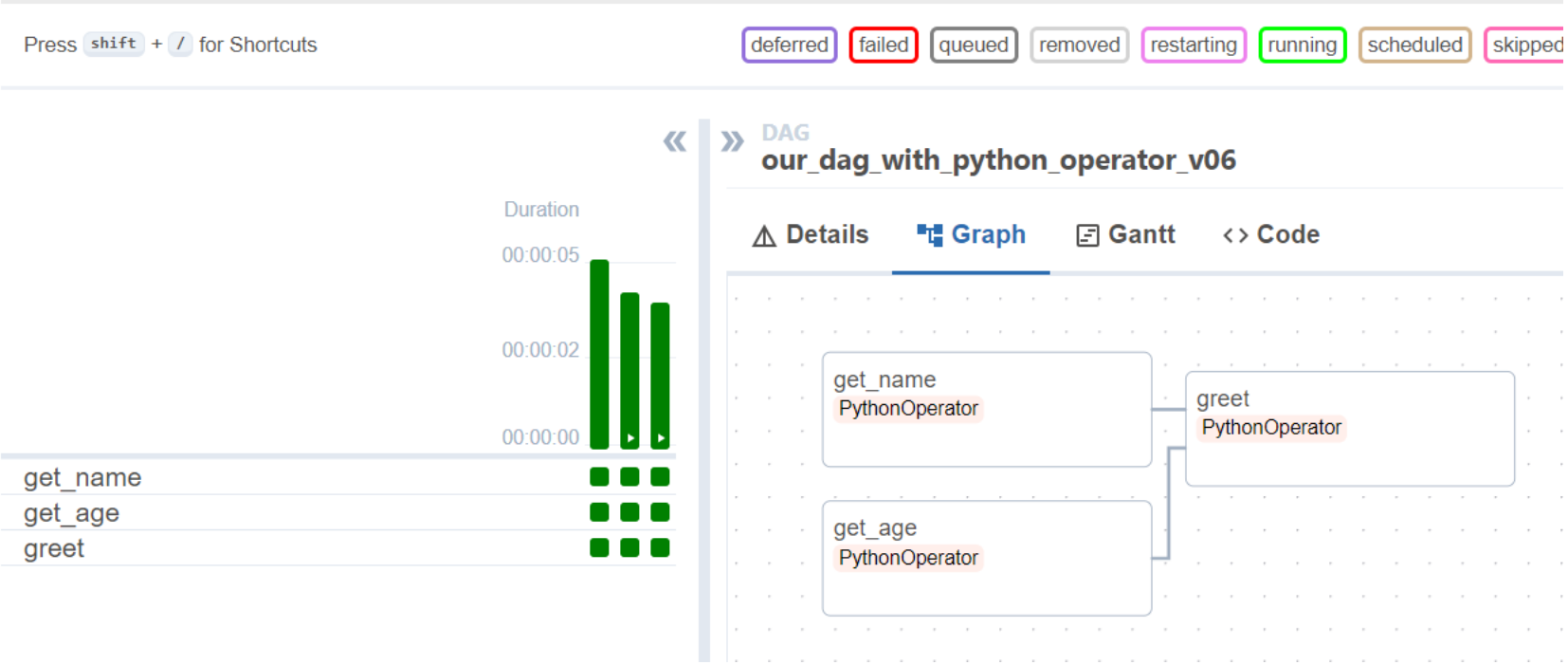
```
26  with DAG(
27      default_args=default_args,
28      dag_id='our_dag_with_python_operator_v06',
29      description='Our first dag using python operator',
30      start_date=datetime(2023, 10, 28),
31      schedule_interval='@daily'
32  ) as dag:
33      task1 = PythonOperator(
34          task_id='greet',
35          python_callable=greet
36          # op_kwargs={'age': 20}
37      )
```

```python
    task2 = PythonOperator(
        task_id='get_name',
        python_callable=get_name
    )

    task3 = PythonOperator(
        task_id='get_age',
        python_callable=get_age
    )

# task1
# task2 >> task1
[task2, task3] >> task1
```

成功建立畫面


latest_only_with_trigger
example3
airflow
our_dag_with_python_operator_v06
coder2j
our_first_dag_v5
coder2j
tutorial
example
airflow

成功執行


Press shift + / for Shortcuts
deferred
failed
queued
removed
restarting
running
scheduled
skipped
Duration
00:00:05
00:00:02
00:00:00
get_name
get_age
greet
DAG
our_dag_with_python_operator_v06
Details
Graph
Gantt
Code
get_name
PythonOperator
greet
PythonOperator
get_age
PythonOperator

查看logs

```
[2023-10-29, 11:22:34 UTC] {task_command.py:416} INFO - Running <TaskInstance: our_dag_with_python_operator_v06.greet manu
[2023-10-29, 11:22:35 UTC] {taskinstance.py:1662} INFO - Exporting env vars: AIRFLOW_CTX_DAG_OWNER='coder2j' AIRFLOW_CTX_D
[2023-10-29, 11:22:35 UTC] {logging_mixin.py:151} INFO - Hello World! My name is Jerry Fridman,and I am 19 years old!
[2023-10-29, 11:22:35 UTC] {python.py:194} INFO - Done. Returned value was: None
[2023-10-29, 11:22:35 UTC] {taskinstance.py:1400} INFO - Marking task as SUCCESS. dag_id=our_dag_with_python_operator_v06,
[2023-10-29, 11:22:35 UTC] {local_task_job_runner.py:228} INFO - Task exited with return code 0
[2023-10-29, 11:22:35 UTC] {taskinstance.py:2778} INFO - 0 downstream tasks scheduled from follow-on schedule check
```

查看xcom

	Key	Value	Timestamp	Dag Id	Task Id
	last_name	Fridman	2023-10-29, 11:22:33	our_dag_with_python_operator_v06	get_name
	first_name	Jerry	2023-10-29, 11:22:33	our_dag_with_python_operator_v06	get_name
	age	19	2023-10-29, 11:22:33	our_dag_with_python_operator_v06	get_age
	last_name	Fridman	2023-10-29, 11:22:33	our_dag_with_python_operator_v06	get_name
	first_name	Jerry	2023-10-29, 11:22:33	our_dag_with_python_operator_v06	get_name
	age	19	2023-10-29, 11:22:33	our_dag_with_python_operator_v06	get_age

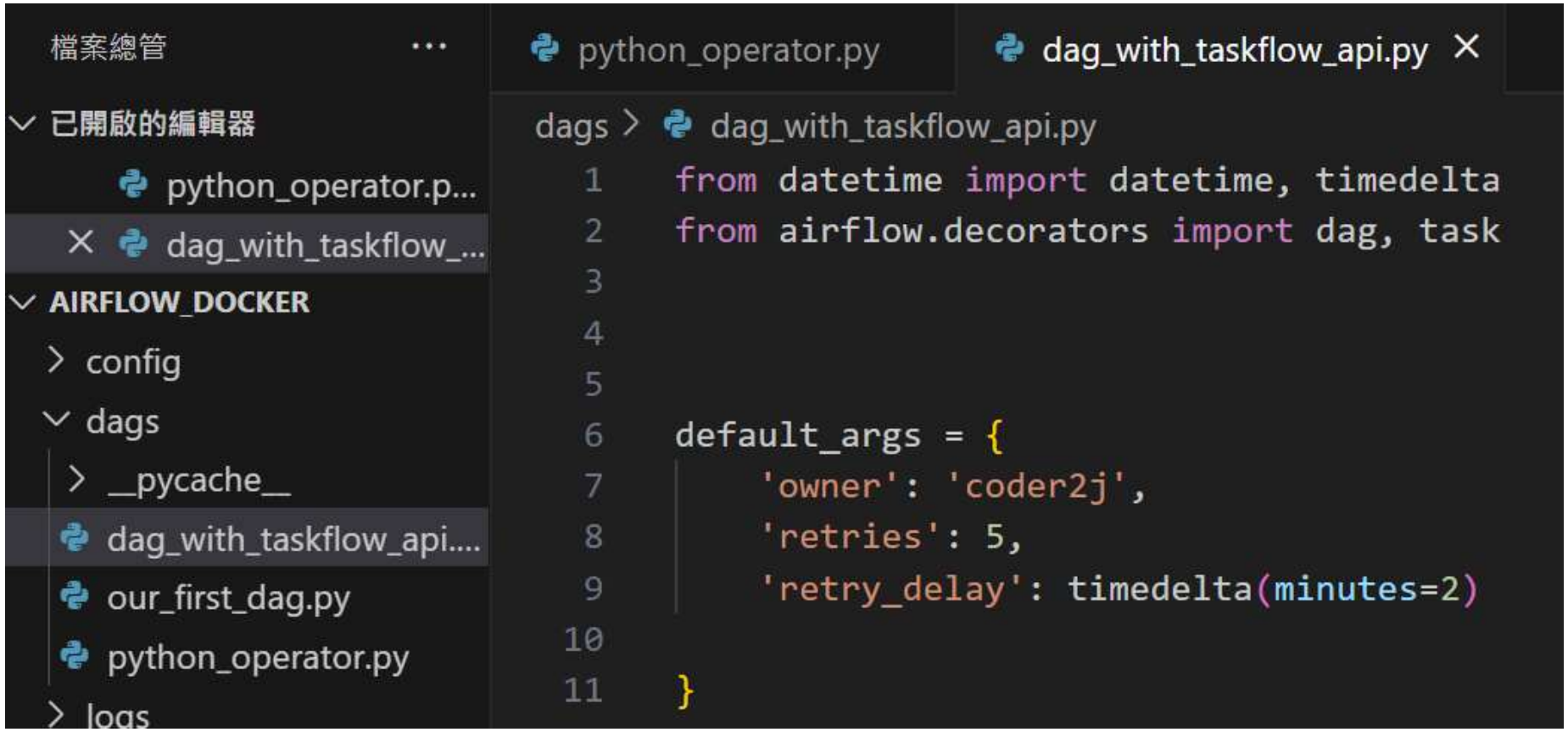
Airflow Task Flow API

說明

- Airflow TaskFlow API 是 Apache Airflow 2.0 中引入的一種撰寫數據流水線的新範例。它是一種函數式 API，允許您顯式聲明消息傳遞，同時隱式聲明任務依賴關係。

- TaskFlow API 與傳統的 DAGs 撰寫範例相比，更加靈活，易於編寫和維護。它採用了一種基於任務的編程模型，其中任務是數據流水線中的基本構建塊。TaskFlow API 通過提供一組簡單的裝飾器和函數，使得編寫數據流水線變得更加容易。
- 簡單來說使用 TaskFlow API撰寫會較於傳統DAGs更容易更清楚

新增一個 dag_with_taskflow_api.py



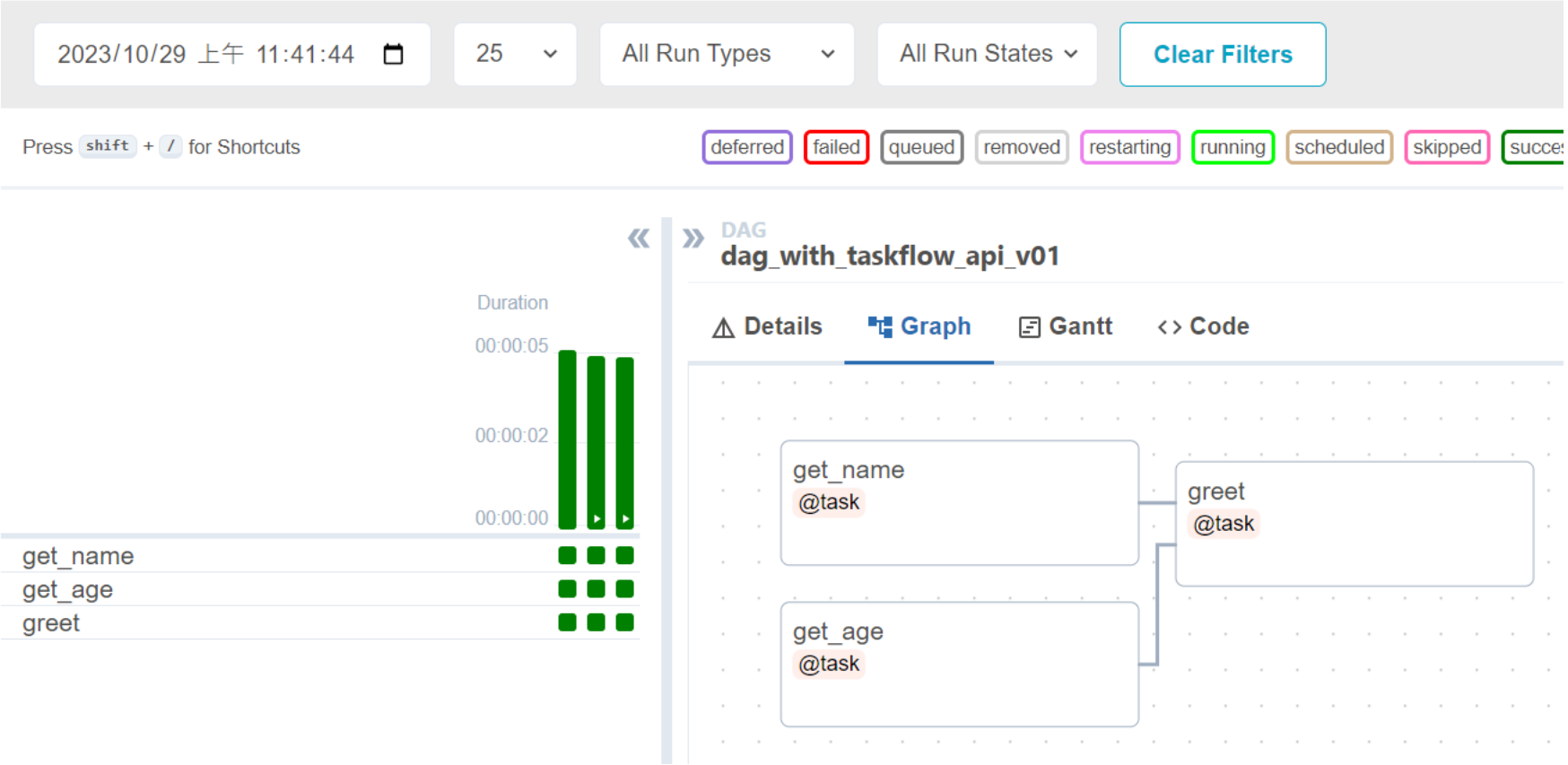
使用 taskflow api撰寫

```python

@dag(dag_id='dag_with_taskflow_api_v01',
    default_args=default_args,
    start_date=datetime(2023, 10, 28),
    schedule_interval='@daily')
def hello_world_etl():
    @task()
    def get_name():
        return "Jerry"
    @task()
    def get_age():
        return 19
    @task()
    def greet(name, age):
        print(f"Hello World! My name is {name}"
        f"and I am {age} years old!")
    name = get_name()
    age = get_age()
    greet(name=name, age=age)

greet_dag = hello_world_etl()
```

成功建立畫面


All 65
Active 1
Paused 64
Running 0
Failed 0
Filter DAGs by tag
Show DAGs with failed latest DAG run
DAG
Owner
Runs
dag_with_taskflow_api_v01
coder2j
dataset_consumes_1
consumes
dataset-scheduled
airflow
dataset_consumes_1_and_2
consumes
dataset-scheduled
airflow

成功執行


2023/10/29 上午 11:41:44
25
All Run Types
All Run States
Clear Filters
Press shift + / for Shortcuts
deferred
failed
queued
removed
restarting
running
scheduled
skipped
succes
Duration
00:00:05
00:00:02
00:00:00
get_name
get_age
greet
DAG
dag_with_taskflow_api_v01
Details
Graph
Gantt
Code
get_name
@task
get_age
@task
greet
@task

查看logs

```
{standard_task_runner.py:85} INFO - Job 1263: Subtask greet
{task_command.py:416} INFO - Running <TaskInstance: dag_with_taskflow_api_v01.greet manual__2023-10-2
{taskinstance.py:1662} INFO - Exporting env vars: AIRFLOW_CTX_DAG_OWNER='coder2j' AIRFLOW_CTX_DAG_ID=
{logging_mixin.py:151} INFO - Hello World! My name is Jerryand I am 19 years old!
{python.py:194} INFO - Done. Returned value was: None
{taskinstance.py:1400} INFO - Marking task as SUCCESS. dag_id=dag_with_taskflow_api_v01, task_id=gree
{local_task_job_runner.py:228} INFO - Task exited with return code 0
{taskinstance.py:2778} INFO - 0 downstream tasks scheduled from follow-on schedule check
```

查看xcom

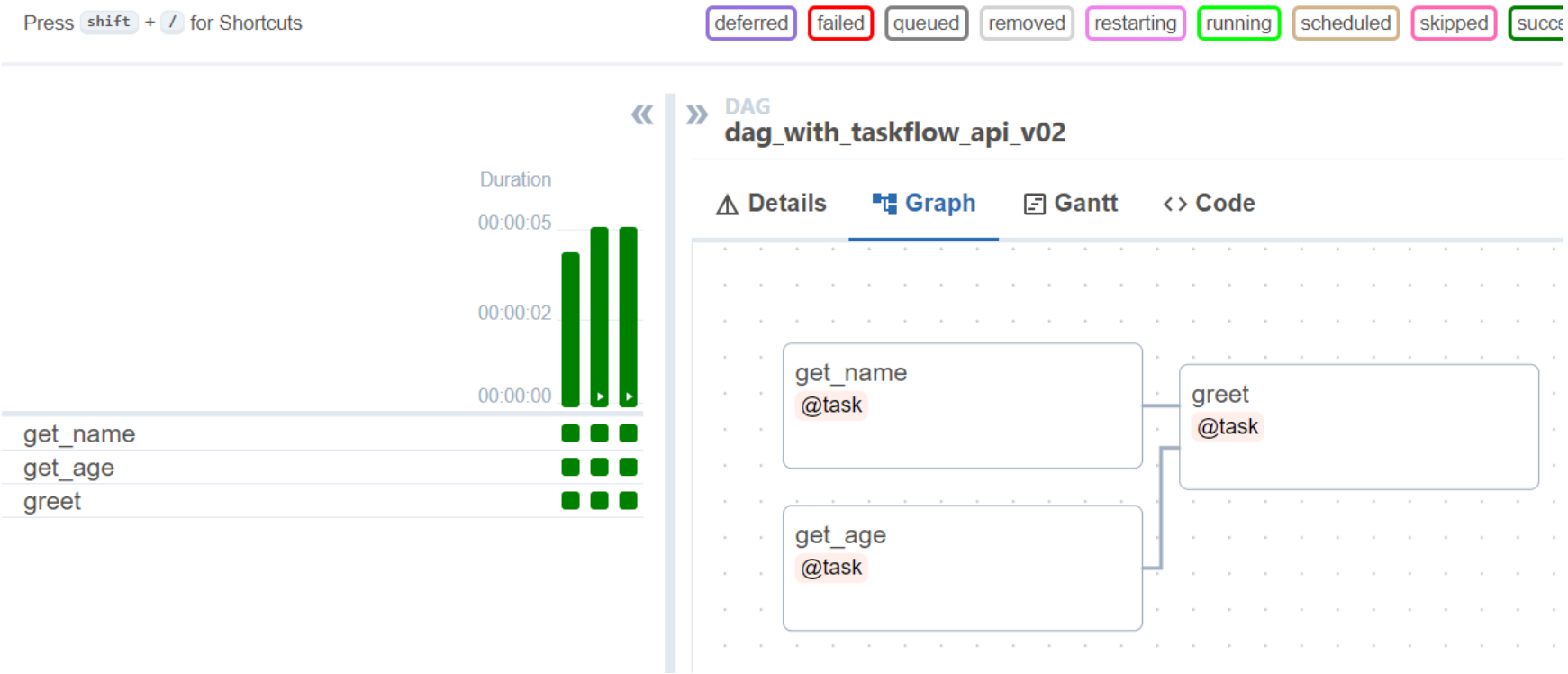
	Key	Value	Timestamp	Dag Id	Task Id
	return_value	Jerry	2023-10-29, 11:41:46	dag_with_taskflow_api_v01	get_name
	return_value	19	2023-10-29, 11:41:46	dag_with_taskflow_api_v01	get_age
	return_value	Jerry	2023-10-29, 11:41:46	dag_with_taskflow_api_v01	get_name
	return_value	19	2023-10-29, 11:41:46	dag_with_taskflow_api_v01	get_age

撰寫更複雜的taskflow api

```
13  @dag(dag_id='dag_with_taskflow_api_v02',
14      default_args=default_args,
15      start_date=datetime(2023, 10, 28),
16      schedule_interval='@daily')
17  def hello_world_etl():
18      @task(multiple_outputs=True)
19      def get_name():
20          return {
21              'first_name': 'Jerry',
22              'last_name': 'Fridman'
23          }
```

```python
    @task()
    def get_age():
        return 19
    @task()
    def greet(first_name, last_name, age):
        print(f"Hello World! My name is {first_name} {last_name}"
        f"and I am {age} years old!")
    name_dict = get_name()
    age = get_age()
    greet(first_name=name_dict['first_name'],
    last_name=name_dict['last_name'],
    age=age)

greet_dag = hello_world_etl()
```

成功啟動


Press shift + / for Shortcuts
deferred
failed
queued
removed
restarting
running
scheduled
skipped
succe
Duration
00:00:05
00:00:02
00:00:00
get_name
get_age
greet
DAG
dag_with_taskflow_api_v02
Details
Graph
Gantt
Code
get_name
@task
greet
@task
get_age
@task

查看logs

```
sk_runner.py:85} INFO - Job 1271: Subtask greet
d.py:416} INFO - Running <TaskInstance: dag_with_taskflow_api_v02.greet manual__2023-10-
e.py:1662} INFO - Exporting env vars: AIRFLOW_CTX_DAG_OWNER='coder2j' AIRFLOW_CTX_DAG_I
in.py:151} INFO - Hello World! My name is Jerry Fridmanand I am 19 years old!
94} INFO - Done. Returned value was: None
e.py:1400} INFO - Marking task as SUCCESS. dag_id=dag_with_taskflow_api_v02, task_id=gre
job_runner.py:228} INFO - Task exited with return code 0
e.py:2778} INFO - 0 downstream tasks scheduled from follow-on schedule check
```

查看xcom

	Key	Value	Timestamp	Dag Id	Task Id
	return_value	{'first_name': 'Jerry', 'last_name': 'Fridman'}	2023-10-29, 11:51:15	dag_with_taskflow_api_v02	get_name
	last_name	Fridman	2023-10-29, 11:51:15	dag_with_taskflow_api_v02	get_name
	first_name	Jerry	2023-10-29, 11:51:15	dag_with_taskflow_api_v02	get_name

可觀察到兩者程式碼行數有明顯差異

使用原本DAGs撰寫

```python
    task2 = PythonOperator(
        task_id='get_name',
        python_callable=get_name
    )

    task3 = PythonOperator(
        task_id='get_age',
        python_callable=get_age
    )

# task1
# task2 >> task1
[task2, task3] >> task1
```

使用taskflow API撰寫

dags > dag_with_taskflow_api.py

```python
    @task()
    def greet(first_name, last_name, age):
        print(f"Hello World! My name is {first_name} {last_name}"
        f"and I am {age} years old!")
    name_dict = get_name()
    age = get_age()
    greet(first_name=name_dict['first_name'],
    last_name=name_dict['last_name'],
    age=age)

greet_dag = hello_world_etl()
```

Airflow Catch-Up and Backfill

說明

- Airflow 中的 Catch-Up 和 Backfill 是兩個不同的概念。

- Catch-Up:

是指在啟用 DAG 時，將所有過去的 DAG Runs 都執行一次，以便追上當前時間點。

- Backfill:

則是指手動重新執行過去某個時間段內的 DAG Runs。

再新增一個.py

python_operator.py dag_with_catchup_and_backfill.py

dags > dag_with_catchup_and_backfill.py

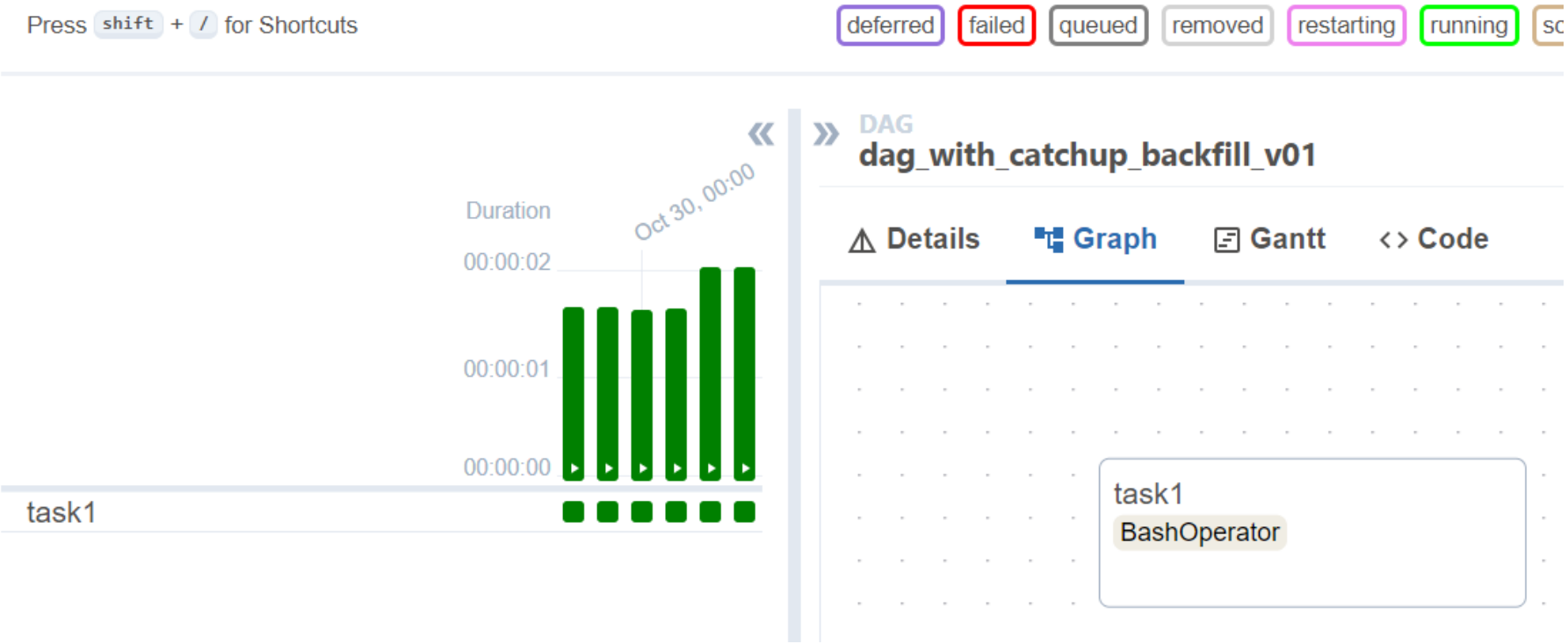
```python
from datetime import datetime, timedelta
from airflow import DAG
from airflow.operators.bash import BashOperator

default_args = {
    'owner': 'coder2j',
    'retries': 5,
    'retry_delay': timedelta(minutes=2)

}
```

```python

with DAG(
    dag_id='dag_with_catchup_backfill_v01',
    default_args=default_args,
    start_date=datetime(2023, 10, 30),
    schedule_interval='@daily',
    catchup=True
) as dag:
    task1 = BashOperator(
        task_id='task1',
        bash_command='echo This is a simple bash command!'
    )
```

成功建立畫面


DAG
Owner
Runs
dag_with_catchup_backfill_v01
coder2j
dag_with_taskflow_api_v02
coder2j
4
dataset_consumes_1
consumes
dataset-scheduled
airflow

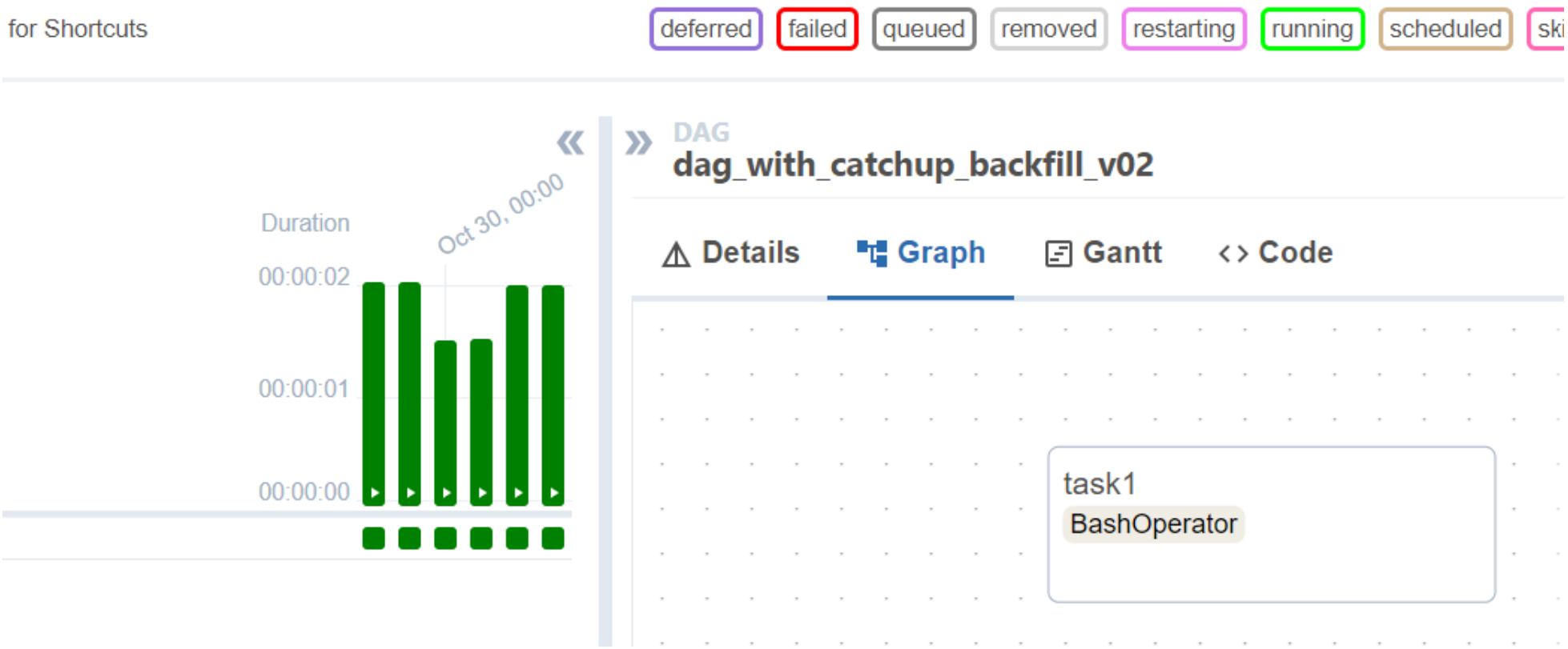
成功執行


Press shift + / for Shortcuts
deferred
failed
queued
removed
restarting
running
Duration
Oct 30, 00:00
00:00:02
00:00:01
00:00:00
task1
DAG
dag_with_catchup_backfill_v01
Details
Graph
Gantt
Code
task1
BashOperator

```python
with DAG(
    dag_id='dag_with_catchup_backfill_v02',
    default_args=default_args,
    start_date=datetime(2023, 10, 30),
    schedule_interval='@daily',
    catchup=False
) as dag:
    task1 = BashOperator(
        task_id='task1',
        bash_command='echo This is a simple bash command!'
    )
```

DAG	Owner	Runs
dag_with_catchup_backfill_v01	coder2j	6
dag_with_catchup_backfill_v02	coder2j	
dag_with_taskflow_api_v02	coder2j	4
dataset_consumes_1 consumes dataset-scheduled	airflow	

成功執行


for Shortcuts
deferred
failed
queued
removed
restarting
running
scheduled
Duration
Oct 30, 00:00
00:00:02
00:00:01
00:00:00
DAG
dag_with_catchup_backfill_v02
Details
Graph
Gantt
Code
task1
BashOperator

查看執行中的container

```
PS C:\Users\jerry> cd "C:\Users\jerry\Desktop\mastercourse\dataEngineer\airflow\airflow_docker"
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\airflow\airflow_docker> docker ps
CONTAINER ID   IMAGE                  COMMAND                  CREATED          STATUS                    PORTS
           NAMES
16896a554b5f   apache/airflow:2.7.2   "/usr/bin/dumb-init …"   19 minutes ago   Up 18 minutes (healthy)   8080/tcp
           airflow_docker-airflow-scheduler-1
8a76374c9b9f   apache/airflow:2.7.2   "/usr/bin/dumb-init …"   19 minutes ago   Up 18 minutes (healthy)   8080/tcp
           airflow_docker-airflow-triggerer-1
75878a44a270   apache/airflow:2.7.2   "/usr/bin/dumb-init …"   19 minutes ago   Up 18 minutes (healthy)   0.0.0.0:8080->
8080/tcp   airflow_docker-airflow-webserver-1
94b67b9093cd   apache/airflow:2.7.2   "/usr/bin/dumb-init …"   19 minutes ago   Up 18 minutes (healthy)   8080/tcp
           airflow_docker-airflow-worker-1
d65609074636   redis:latest           "docker-entrypoint.s…"   19 minutes ago   Up 19 minutes (healthy)   6379/tcp
           airflow_docker-redis-1
f82968dffe8e   postgres:13            "docker-entrypoint.s…"   19 minutes ago   Up 19 minutes (healthy)   5432/tcp
           airflow_docker-postgres-1
```

Command:
Docker exec –it <container name> bash

```
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\airflow\airflow_docker> docker exec -it 16896a554b5f bash
airflow@16896a554b5f:/opt/airflow$ airflow dags backfill -s 2023-10-30 -e 2023-11-08 dag_with_catchup_backfill_v02
/home/airflow/.local/lib/python3.8/site-packages/airflow/cli/commands/dag_command.py:129 RemovedInAirflow3Warning: --ign
ore-first-depends-on-past is deprecated as the value is always set to True
[2023-10-30T00:42:25.366+0000] {dagbag.py:536} INFO - Filling up the DagBag from /opt/airflow/dags
/home/airflow/.local/lib/python3.8/site-packages/airflow/models/dagbag.py:342 RemovedInAirflow3Warning: Param `schedule_
interval` is deprecated and will be removed in a future release. Please use `schedule` instead.
[2023-10-30T00:42:26.232+0000] {executor_loader.py:117} INFO - Loaded executor: CeleryExecutor
[2023-10-30T00:42:26.610+0000] {base_executor.py:144} INFO - Adding to queue: ['airflow', 'tasks', 'run', 'dag_with_catc
hup_backfill_v02', 'task1', 'backfill__2023-10-30T00:00:00+00:00', '--depends-on-past', 'ignore', '--local', '--pool', '
default_pool', '--subdir', 'DAGS_FOLDER/dag_with_catchup_and_backfill.py']
[2023-10-30T00:42:31.423+0000] {backfill_job_runner.py:412} INFO - [backfill progress] | finished run 0 of 10 | tasks wa
iting: 9 | succeeded: 0 | running: 1 | failed: 0 | skipped: 0 | deadlocked: 0 | not ready: 9
[2023-10-30T00:42:36.361+0000] {dagrun.py:653} INFO - Marking run <DagRun dag_with_catchup_backfill_v02 @ 2023-10-30T00:
00:00+00:00: backfill__2023-10-30T00:00:00+00:00, state:running, queued_at: None. externally triggered: False> successfu
```

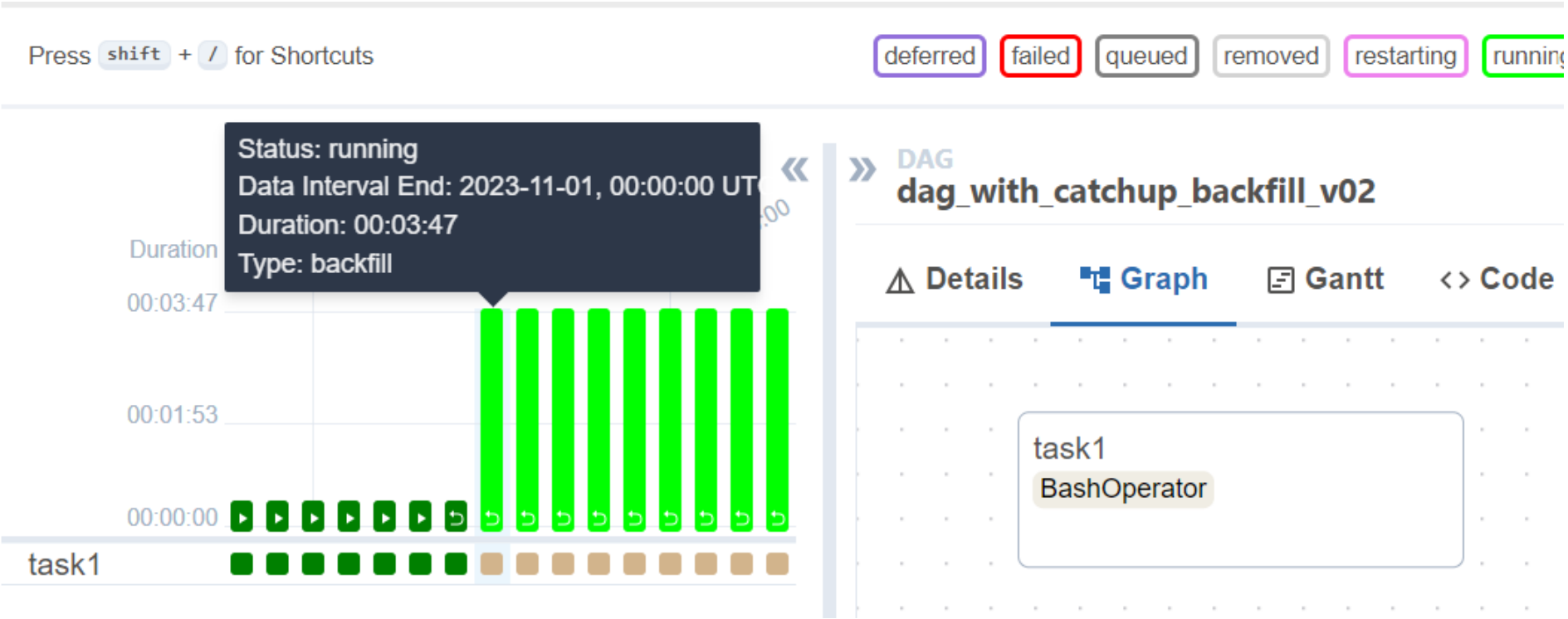
成功執行Catch-Up畫面(1/2)

```
These tasks have succeeded:
DAG ID                          Task ID    Run ID                                  Try number
------------------------------  ---------  ----------------------------------  ------------
dag_with_catchup_backfill_v02   task1      backfill__2023-10-30T00:00:00+00:00             1

These tasks are running:
DAG ID    Task ID    Run ID    Try number
--------  ---------  --------  ------------

These tasks have failed:
DAG ID    Task ID    Run ID    Try number
--------  ---------  --------  ------------

These tasks are skipped:
DAG ID    Task ID    Run ID    Try number
--------  ---------  --------  ------------
```

成功執行Catch-Up畫面(2/2)

```
These tasks are deadlocked:
DAG ID                          Task ID    Run ID                                  Try number
------------------------------  ---------  ------------------------------------  ------------
dag_with_catchup_backfill_v02   task1      backfill__2023-10-31T00:00:00+00:00             1
dag_with_catchup_backfill_v02   task1      backfill__2023-11-01T00:00:00+00:00             1
dag_with_catchup_backfill_v02   task1      backfill__2023-11-02T00:00:00+00:00             1
dag_with_catchup_backfill_v02   task1      backfill__2023-11-03T00:00:00+00:00             1
dag_with_catchup_backfill_v02   task1      backfill__2023-11-04T00:00:00+00:00             1
dag_with_catchup_backfill_v02   task1      backfill__2023-11-05T00:00:00+00:00             1
dag_with_catchup_backfill_v02   task1      backfill__2023-11-06T00:00:00+00:00             1
dag_with_catchup_backfill_v02   task1      backfill__2023-11-07T00:00:00+00:00             1
dag_with_catchup_backfill_v02   task1      backfill__2023-11-08T00:00:00+00:00             1
airflow@16896a554b5f:/opt/airflow$
```

成功執行


Press shift + / for Shortcuts
deferred
failed
queued
removed
restarting
Status: running
Data Interval End: 2023-11-01, 00:00:00 UT
Duration: 00:03:47
Type: backfill
Duration
00:03:47
00:01:53
00:00:00
task1
DAG
dag_with_catchup_backfill_v02
Details
Graph
Gantt
Code
task1
BashOperator

Airflow Scheduler with Cron Expression

說明

- Airflow 中的 Cron Expression 是一種用於調度 DAG 執行的表達式。它是一種基於時間的表達式，可以指定 DAG 的運行時間。Cron Expression 由五個或六個字段組成，分別表示分鐘、小時、日期、月份、星期和年份（可選）。

- Airflow 中的 Cron Expression 與傳統的 Cron 表達式略有不同。Airflow 的 Cron Expression 支援秒級精度，而傳統的 Cron 表達式只支持分鐘級別的精度。
- 此外，Airflow 的 Cron Expression 還支援使用 @yearly、@monthly、@weekly、@daily、@hourly 等預定義的時間間隔來指定 DAG 的運行時間。

新增一個.py

python_operator.py dag_with_cron_expression.py ×

dags > dag_with_cron_expression.py

```python
from datetime import datetime, timedelta
from airflow import DAG
from airflow.operators.bash import BashOperator

default_args = {
    'owner': 'coder2j',
    'retries': 5,
    'retry_delay': timedelta(minutes=2)

}
```

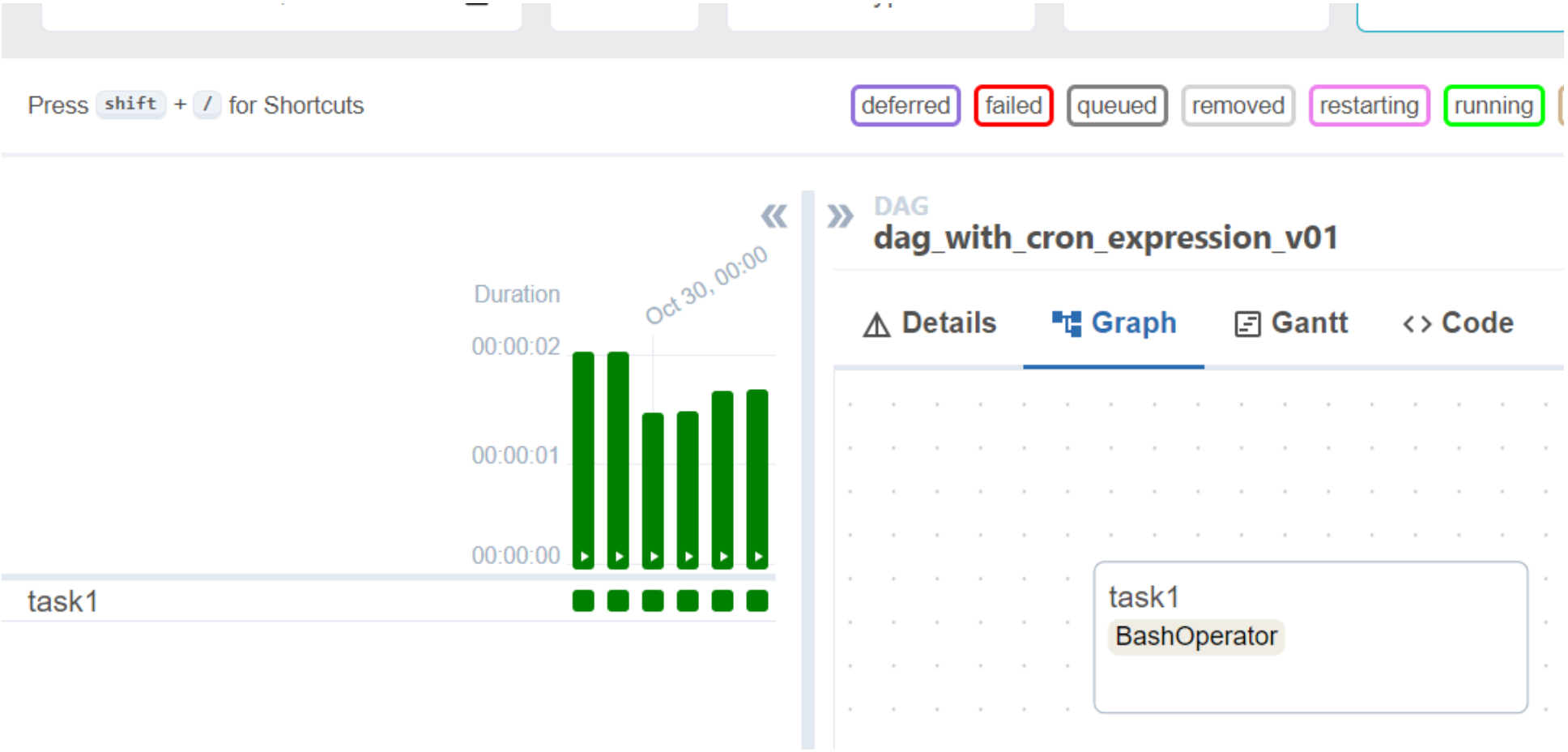
使用 Cron Expression

```
11
12  with DAG(
13      dag_id='dag_with_cron_expression_v01',
14      default_args=default_args,
15      start_date=datetime(2023, 10, 30),
16      schedule_interval='@daily'
17  ) as dag:
18      task1 = BashOperator(
19          task_id='task1',
20          bash_command='echo dag with cron expression!'
21      )
22      task1
```

成功建立畫面


All 69
Active 2
Paused 67
Running 9
Failed 0
Filter DAGs by tag
DAG
Owner
Runs
dag_with_catchup_backfill_v02
coder2j
7
9
dag_with_cron_expression_v01
coder2j
dag_with_taskflow_api_v02
coder2j
4

成功執行


Press shift + / for Shortcuts
deferred
failed
queued
removed
restarting
running
Duration
Oct 30, 00:00
00:00:02
00:00:01
00:00:00
task1
DAG
dag_with_cron_expression_v01
Details
Graph
Gantt
Code
task1
BashOperator

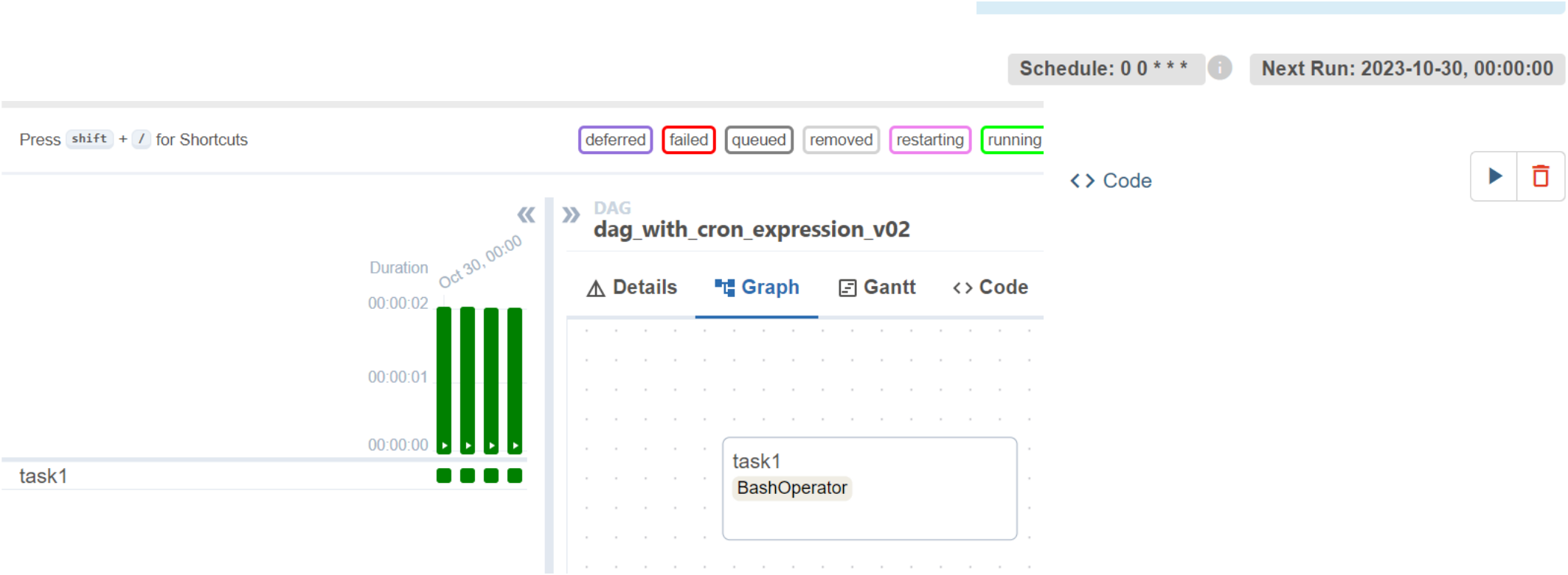
修改 schedule_interval

```
12  with DAG(
13      dag_id='dag_with_cron_expression_v02',
14      default_args=default_args,
15      start_date=datetime(2023, 10, 30),
16      schedule_interval='0 0 * * *'
17  ) as dag:
18      task1 = BashOperator(
19          task_id='task1',
20          bash_command='echo dag with cron expression!'
21      )
22      task1
```

成功建立畫面


All 70
Active 3
Paused 67
Running 9
Failed 0
Filter DAGs by tag
DAG
Owner
Runs
dag_with_catchup_backfill_v02
coder2j
7
9
dag_with_cron_expression_v01
coder2j
6
dag_with_cron_expression_v02
coder2j
dag_with_taskflow_api_v02
coder2j
4

成功執行


Schedule: 0 0 * * *
Next Run: 2023-10-30, 00:00:00
Press shift + / for Shortcuts
deferred
failed
queued
removed
restarting
running
Code
Duration
Oct 30, 00:00
00:00:02
00:00:01
00:00:00
task1
DAG
dag_with_cron_expression_v02
Details
Graph
Gantt
Code
task1
BashOperator

End