

# Project: Creating and Automating a Set of Data Pipelines with Airflow

สิ่งที่คาดหวังในโปรเจคนี้

- นำโคดจาก [Project: Building a Data Modeling with Postgres \(SQL\)](#) มาสร้าง data pipeline โดยใช้ Airflow
- มีการเขียน documentation อธิบายสิ่งที่ตัวเองทำลงไว้ รวมไปถึงการออกแบบ data model
- มี instruction ในการรันโค้ดของตัวเอง

gitpod /workspace/swu-ds525/05-creating-and-scheduling-data-pipelines (main) \$  
docker-compose up

ขั้นตอน Overall

① Import DAG, timezone  
② สร้าง Data pipeline ชื่อ "MyDag"

③ save file python

④ กด รันเพลย์ท์ Airflow จนไฟเขียวชี้เขียวที่ Airflow

$t_1 > t_2$

⑤ bash operator, python operator

⑥ gridview

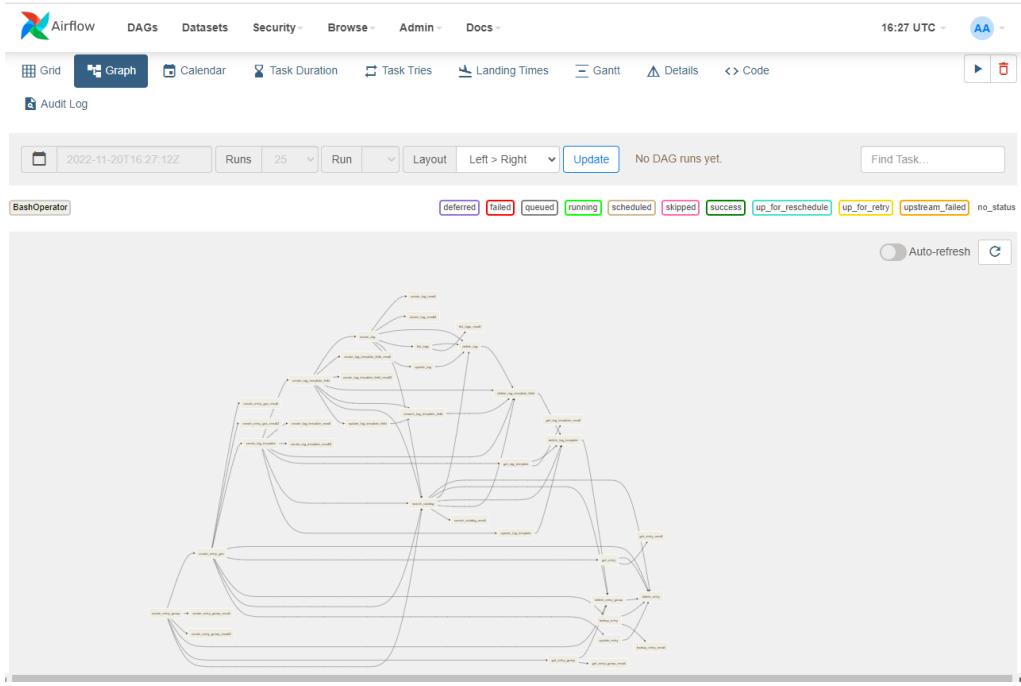
ดูตาม กำหนดเวลา

trigger 1 ที่ต้อง 1 dag run

clear task

$t_1 \rightarrow$  details  $\rightarrow$  clear  $\rightarrow$  confirm  
as run failed or error

cronitor  $\rightarrow$  expression ที่จะ set ให้ run 自動化 - ผ่านไป  $\frac{24}{26}$



05-creating-and-scheduling-data-pipelines > dags > my\_dag\_revise.py

```

1  #import DAG เป็น pipeline
2  from airflow import DAG
3  #import Timezone
4  from airflow.utils import timezone
5  #step 2 ใช้ EmptyOperator เลย import EmptyOperator เข้ามา
6  from airflow.operators.empty import EmptyOperator
7
8
9  #context manager เป็นการประกาศหัว
10 #""my_dag"" ชื่อเดียวกับชื่อ file
11 #start_date 2022, 10, 8
12 # schedule = None ปั้งไม่schedule
13 # step10
14 # schedule
15 # schedule เที่ยงคืน คือ 0
16 #v1 schedule = None ปั้งไม่set schedule
17 with DAG(
18     "my_dag",
19     start_date = timezone.datetime(2022, 10, 8),
20     schedule = None,
21 ):
22     t1 = EmptyOperator( task_id = "t1")
23     t2 = EmptyOperator( task_id = "t2")

```

16:49 UTC AA

DAG ID	Owner	Last Run	Runs
latest_only_with_trigger	airflow	2022-11-20, 12:48:07	0
my_dag	airflow	2022-11-20, 16:24:00	3
my_dag2	airflow	2022-11-20, 16:00:00	7
my_dag_revise	airflow	None	0
example2	airflow	2022-11-20, 12:40:07	0
example3	airflow	2022-11-20, 12:40:07	0

16:55 UTC AA

## DAGs

All 47 Active 3 Paused 44

Filter DAGs by tag:  Search DAGs:

Auto-refresh  Refresh

DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks	Actions
my_dag	airflow	325	0 0 * * *	2022-11-20, 16:24:00	2022-11-20, 00:00:00	4	
my_dag2	airflow	7	*30 * * * *	2022-11-20, 16:00:00	2022-11-20, 16:30:00	5	
my_dag_revise	airflow	0	None	2022-11-20, 16:55:59	None	0	

Trigger DAG Trigger DAG w/ config

16:56 UTC AA

Triggered my\_dag\_revise, it should start any moment now.

## DAGs

All 47 Active 3 Paused 44

Filter DAGs by tag:  Search DAGs:

Auto-refresh  Refresh

DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks	Actions
my_dag	airflow	325	0 0 * * *	2022-11-20, 16:24:00	2022-11-20, 00:00:00	4	
my_dag2	airflow	7	*30 * * * *	2022-11-20, 16:00:00	2022-11-20, 16:30:00	5	
my_dag_revise	airflow	1	None	2022-11-20, 16:55:59	None	2	

Showing 1-3 of 3 DAGs

16:56 UTC AA

DAG: my\_dag\_revise

success Schedule: None Next Run: None

Grid Graph Calendar Task Duration Task Tries Landing Times Gantt Details Code Audit Log

Update

2022-11-20T16:56:00Z Runs 25 Run manual\_\_2022-11-20T16:55:59.972252+00:00 Layout Left > Right Find Task...

EmptyOperator

Auto-refresh

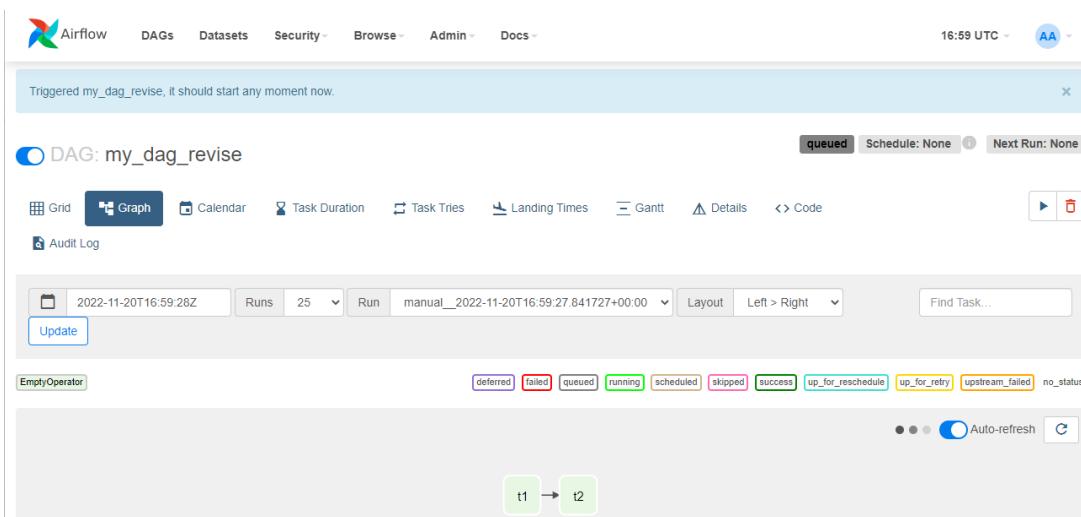
t1

t2

```

05-creating-and-scheduling-data-pipelines > dags > my_dag_revise.py
1 #import DAG เป็น pipeline
2 from airflow import DAG
3 #import Timezone
4 from airflow.utils import timezone
5 #step 2 ใช้ EmptyOperator เลย import EmptyOperator เข้ามา
6 from airflow.operators.empty import EmptyOperator
7
8
9 #context manager เป็นการประกาศหัว
10 #""my_dag"" ชื่อเดียวกับชื่อ file
11 #start date 2022, 10, 8
12 # schedule = None ยังไม่กำหนด
13 # step10
14 # schedule
15 # schedule เพียงคืน คือ 0
16 # tags =['workshop'] จะทำให้เป็นชุดเดน
17 #v1 schedule = None ยังไม่กำหนด
18 with DAG(
19     "my_dag_revise",
20     start_date = timezone.datetime(2022, 10, 8),
21     schedule = None,
22     tags =["workshop"],
23 ):
24     t1 = EmptyOperator( task_id = "t1")
25     t2 = EmptyOperator( task_id = "t2")
26
27 #t1 รันก่อน t2
28     t1 >> t2
29

```



```

05-creating-and-scheduling-data-pipelines > dags > my_dag_revise.py
25  # ):
26  #     t1 = EmptyOperator( task_id = "t1")
27  #     t2 = EmptyOperator( task_id = "t2")
28
29  # #t1 依赖于 t2
30  #     t1 >> t2
31
32  #v2
33  with DAG(
34      "my_dag_revise",
35      start_date = timezone.datetime(2022, 10, 8),
36      schedule = None,
37      tags =["workshop"],
38  ):
39      t1 = EmptyOperator( task_id = "t1")
40
41      echo_hello = BashOperator(
42          task_id = "echo_hello",
43          bash_command= "echo 'hello'",
44      )
45
46      t2 = EmptyOperator( task_id = "t2")
47
48  #t1 依赖于 t2
49  |     t1 >> t2

```

DAG: my\_dag\_revise

Grid  Graph  Calendar  Task Duration  Task Tries  Landing Times  Gantt  Details  Code ▶ ⚙

Audit Log

2022-11-20T16:59:28Z Runs 25 Run manual\_2022-11-20T16:59:27.841727+00:00 Layout Left > Right Find Task...

Update

BashOperator EmptyOperator deferred failed queued running scheduled skipped success up\_for\_reschedule up\_for\_retry upstream\_failed no\_status

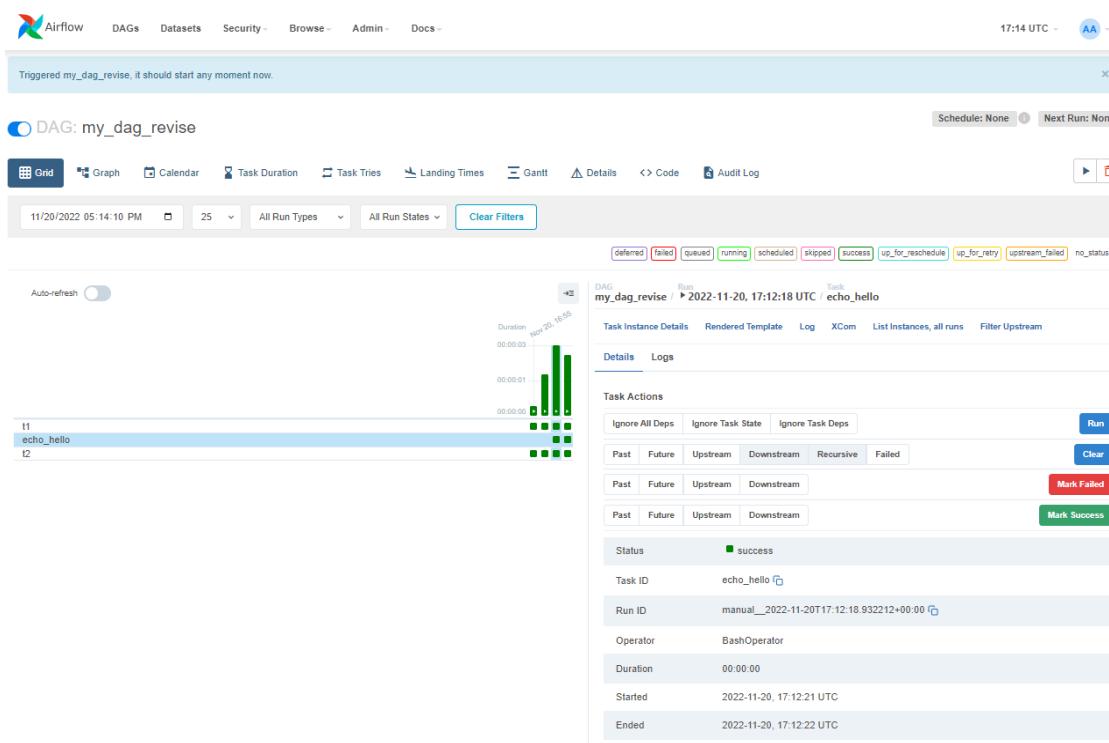
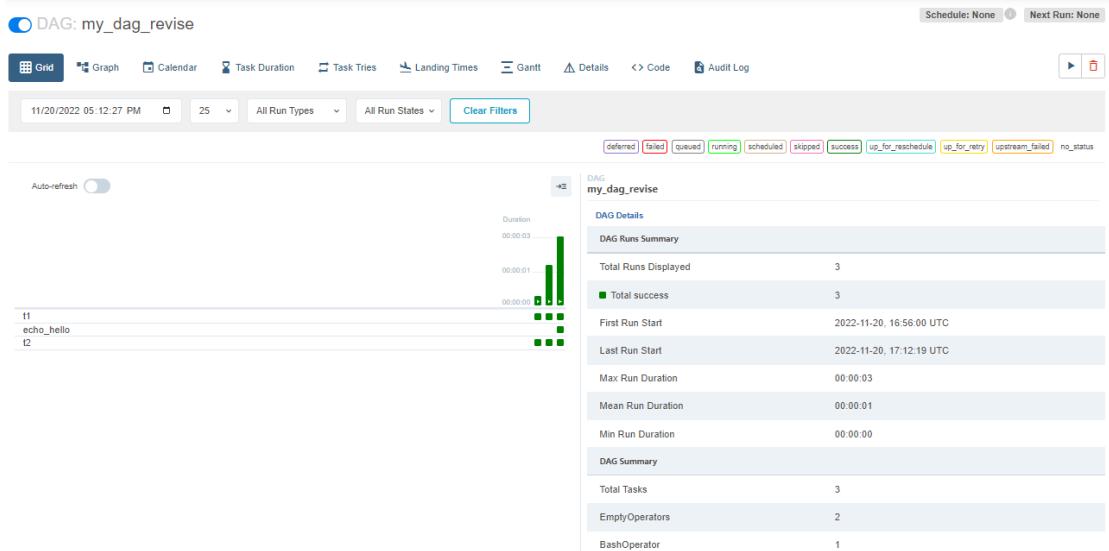
Auto-refresh ↻ C

```

graph LR
    t1[t1] --> echo_hello[echo_hello]
    echo_hello --> t2[t2]

```

#t1 依赖于 t2  
| t1 >> echo\_hello >> t2



Triggered my\_dag\_revise, it should start any moment now.

DAG: my\_dag\_revise

Schedule: None | Next Run: None

Grid | Graph | Calendar | Task Duration | Task Tries | Landing Times | Gantt | Details | <> Code | Audit Log

11/20/2022 05:14:10 PM | 25 | All Run Types | All Run States | Clear Filters

Duration: Net 20:16:55

Task Instance Details | Rendered Template | Log | XCom | List Instances, all runs | Filter Upstream

Details | Logs

(by attempts)

1

All Levels | All File Sources | Wrap | Full Logs | Download | See More

```
130794f1390d
*** Reading local file /opt/airflow/logs/dag_id=echo_dag_revise/run_id=manual_2022-11-20T17:12:18.93222+00:00/task_id=0
[2022-11-20, 17:12:21 UTC] (taskinstance.py:116) INFO - Dependencies all met for <TaskInstance: my_dag_revise.echo_hello>
[2022-11-20, 17:12:21 UTC] (taskinstance.py:116) INFO - Dependencies all met for <TaskInstance: my_dag_revise.echo_hello>
[2022-11-20, 17:12:21 UTC] (taskinstance.py:116) INFO -
[2022-11-20, 17:12:21 UTC] (taskinstance.py:116) INFO - Starting attempt 1 of 1
[2022-11-20, 17:12:21 UTC] (taskinstance.py:116) INFO -
[2022-11-20, 17:12:21 UTC] (taskinstance.py:116) INFO - Executing <Task(BashOperator): echo_hello> on 2022-11-20 17:12:21 UTC
[2022-11-20, 17:12:21 UTC] (standard_task_runner.py:14) INFO - Started process 24279 to run Task
[2022-11-20, 17:12:21 UTC] (standard_task_runner.py:14) INFO - Running: ['***', 'task', 'run', 'my_dag_revise', 'echo_h
[2022-11-20, 17:12:21 UTC] (standard_task_runner.py:14) INFO - Job 487: Subtask echo_hello
[2022-11-20, 17:12:21 UTC] (taskinstance.py:116) INFO - Cleaning up the temporary directory /tmp/airflow/_tmp/dags/my_dag_revise.py
[2022-11-20, 17:12:21 UTC] (taskinstance.py:116) WARNING - Dependency <Task(BashOperator): create_entry_group>, delete_ent
[2022-11-20, 17:12:21 UTC] (taskinstance.py:116) WARNING - Dependency <Task(BashOperator): delete_entry_group>, create_ent
[2022-11-20, 17:12:21 UTC] (taskinstance.py:116) WARNING - Dependency <Task(BashOperator): create_tag>, delete_t
[2022-11-20, 17:12:21 UTC] (taskinstance.py:116) WARNING - Dependency <Task(BashOperator): delete_entry>, create_ent
[2022-11-20, 17:12:21 UTC] (taskinstance.py:116) WARNING - Dependency <Task(BashOperator): create_tag>, delete_t
[2022-11-20, 17:12:21 UTC] (taskinstance.py:116) WARNING - Dependency <Task(BashOperator): delete_tag>, create_t
[2022-11-20, 17:12:21 UTC] (taskinstance.py:116) WARNING - Dependency <Task(BashOperator): get_id>, prepare_email
[2022-11-20, 17:12:21 UTC] (taskinstance.py:116) INFO -
```

```
#v3
with DAG(
    "my_dag_revise",
    start_date = timezone.datetime(2022, 10, 8),
    schedule = None,
    tags = ["workshop"],
):
    t1 = EmptyOperator( task_id = "t1")

    echo_hello = BashOperator(
        task_id = "echo_hello",
        bash_command= "echo 'hello'",
    )
    def _print_hey():
        print("Hey!")

    print_hey = PythonOperator(
        task_id = "print_hey",
        python_callable = _print_hey,
    )

    t2 = EmptyOperator( task_id = "t2")

    #t1 รันก่อน t2
    t1 >> echo_hello >> print_hey >> t2
```

17:24 UTC | AA

DAG: my\_dag\_revise

Schedule: None | Next Run: None

Grid | Graph | Calendar | Task Duration | Task Tries | Landing Times | Gantt | Details | <> Code | Audit Log

2022-11-20T17:17:00Z | Run | manual\_2022-11-20T17:16:59.347340+00:00 | Layout | Left > Right | Update | Find Task...

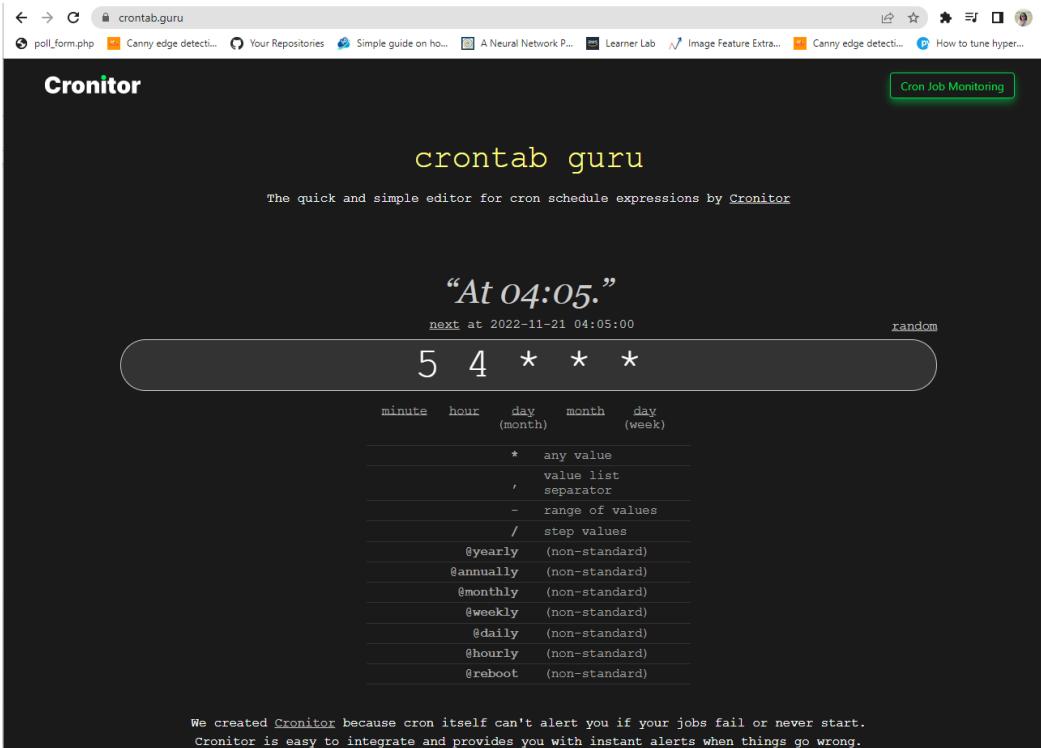
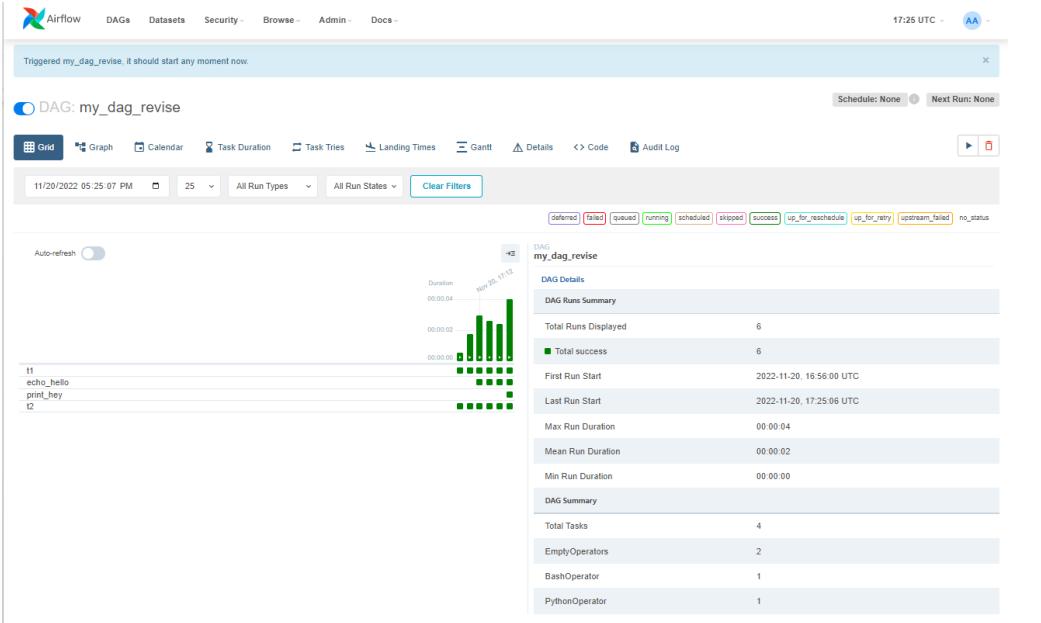
EmptyOperator | EmptyOperator | PythonOperator

defered failed queued running scheduled skipped success up\_for\_reschedule up\_for\_retry upstream\_failed no\_status

Auto-refresh

```

graph LR
    t1[t1] --> echo_hello[echo_hello]
    echo_hello --> print_hey[print_hey]
    print_hey --> t2[t2]
  
```



← → ⌛ crontab.guru/\*\_\*\_\*\_\*\_\*

poll\_form.php Canny edge detect... Your Repositories Simple guide on ho... A Neural Network P... Learner Lab Image Feature Extra... Canny edge detect... How to tune hyper...

**Cronitor**

**Cron Job Monitoring**

## crontab guru

The quick and simple editor for cron schedule expressions by Cronitor

*"At every minute."*

next at 2022-11-21 00:29:00 random

★ \* \* \* \*

minute	hour	day	month	day
(month)				(week)
*				
,				
-				
/				
@yearly				
@annually				
@monthly				
@weekly				
@daily				
@hourly				
@reboot				

We created Cronitor because cron itself can't alert you if your jobs fail or never start.  
Cronitor is easy to integrate and provides you with instant alerts when things go wrong.

← → ⌛ crontab.guru#30\_8\_\*\_\*\_\*

poll\_form.php Canny edge detect... Your Repositories Simple guide on ho... A Neural Network P... Learner Lab Image Feature Extra... Canny edge detect... How to tune hyper...

**Cronitor**

**Cron Job Monitoring**

## crontab guru

The quick and simple editor for cron schedule expressions by Cronitor

*"At 08:30."*

next at 2022-11-21 08:30:00 random

30 8 \* \* \*

minute	hour	day	month	day
(month)				(week)
*				
,				
-				
/				
@yearly				
@annually				
@monthly				
@weekly				
@daily				
@hourly				
@reboot				

We created Cronitor because cron itself can't alert you if your jobs fail or never start.  
Cronitor is easy to integrate and provides you with instant alerts when things go wrong.

← → ⌛ crontab.guru#0\_17\_16\_\*\_\*

poll\_form.php Canny edge detect... Your Repositories Simple guide on ho... A Neural Network P... Learner Lab Image Feature Extra... Canny edge detect... How to tune hyper...

## “At 17:00 on day-of-month 16.”

next at 2022-12-16 17:00:00 random

0 17 16 \* \*

minute	hour	day	month	day
(month)				(week)
0	17	16	*	*

```

with DAG(
    "my_dag_revise",
    start_date = timezone.datetime(2022, 10, 8),
    schedule = "* * * * *",
    tags =["workshop"],
):
    t1 = EmptyOperator( task_id = "t1")

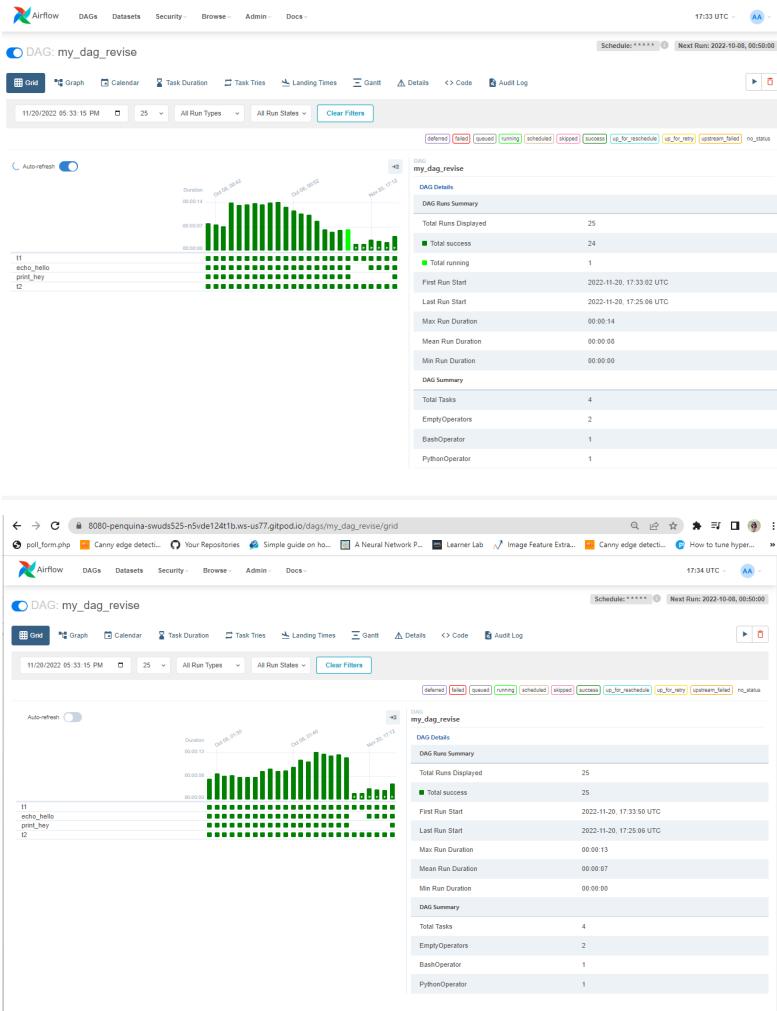
    echo_hello = BashOperator(
        task_id = "echo_hello",
        bash_command= "echo 'hello'",
    )
    def _print_hey():
        print("Hey!")

    print_hey = PythonOperator(
        task_id = "print_hey",
        python_callable = _print_hey,
    )

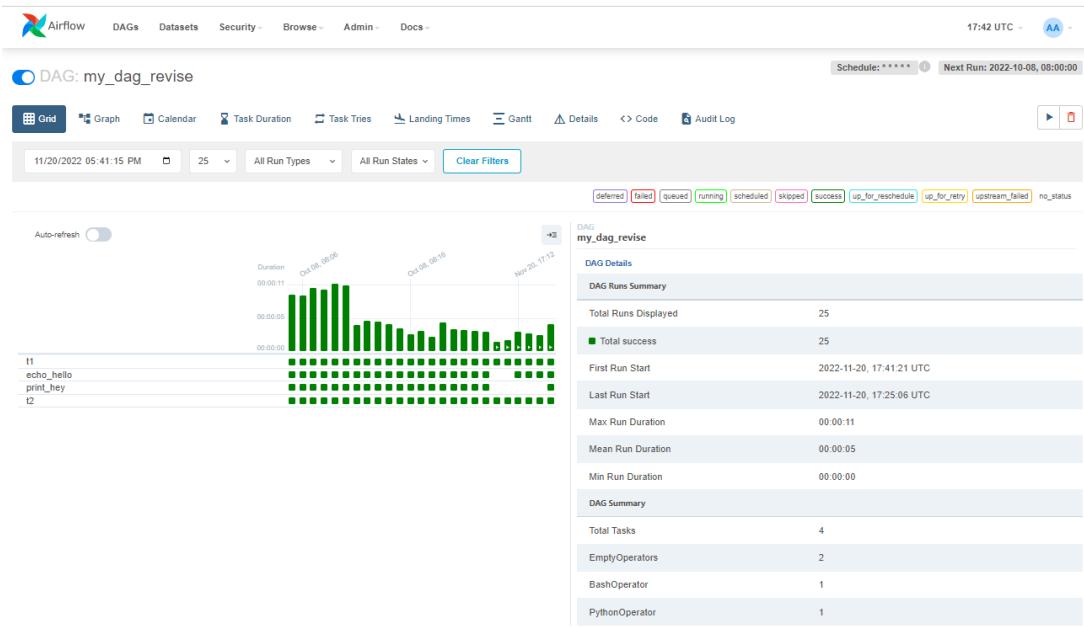
    t2 = EmptyOperator( task_id = "t2")

#t1 នឹងរាយ t2
    t1 >> echo_hello >> print_hey >> t2

```



```
#t1 รันก่อน t2
| #t1 >> echo_hello >> print_hey >> t2
| t1 >> [echo_hello, print_hey] >> t2
```




---

Week9 ช่วงทบทวน week 8

## Airflow Concepts

The screenshot shows a Python script for an Airflow DAG. The code defines a DAG named 'my\_dag' with a schedule interval of '@daily'. It contains three tasks: 'get\_data\_from\_db', 'clean\_timestamp\_data', and 'upload\_file\_to\_s3'. Arrows indicate dependencies between the tasks. Annotations on the right side explain the components:

- DAG**: Points to the outermost curly braces defining the DAG.
- Operator**: Points to the first task definition, 'get\_data\_from\_db'.
- Task**: Points to the second task definition, 'clean\_timestamp\_data'.
- Define the dependencies here**: Points to the final line of code showing the task dependencies: `get_data_from_db >> clean_timestamp_data >> upload_file_to_s3`.

```
default_args = {
    "owner": "Kan Ouivirach",
    "start_date": days_ago(2),
}

with DAG(
    "my_dag",
    schedule_interval="@daily",
    default_args=default_args,
    catchup=False,
) as dag:

    get_data_from_db = PythonOperator(
        task_id="get_data_from_db",
        python_callable=_get_data_from_db,
    )

    clean_timestamp_data = PythonOperator(
        task_id="clean_timestamp_data",
        python_callable=_clean_timestamp_data,
    )

    upload_file_to_s3 = PythonOperator(
        task_id="upload_file_to_s3",
        python_callable=_upload_file_to_s3,
    )

    get_data_from_db >> clean_timestamp_data >> upload_file_to_s3
```

## Let's Build this Airflow DAG



DAG runs every 30 minutes

1. EmptyOperator: start
2. BashOperator: echo\_hello
3. PythonOperator: say\_hello
4. PythonOperator: print\_log\_messages
5. EmptyOperator: end

## DAGs

All (5)	Active (0)	Paused (1)	Runs	Schedule	Last Run	Next Run	Recent Tasks	Actions	Links
my_dag [workshop]	airflow	0	0/30 * * * *	2022-11-20, 16:24:00	2022-11-21, 00:00:00	0/30 * * * *	0/30 * * * *	[▶]	[...]
my_dag2 [workshop]	airflow	1	*/30 * * * *	2022-11-21, 14:30:00	2022-11-21, 15:00:00	0/30 * * * *	0/30 * * * *	[▶]	[...]
my_dag_revise [workshop]	airflow	0	0/8 * * * *	2022-11-20, 17:25:06	2022-11-21, 08:30:00	0/30 * * * *	0/30 * * * *	[▶]	[...]
my_dag_w9 [workshop]	airflow	0	*/30 * * * *	2022-11-21, 10:30:00	2022-11-21, 08:00:00	0/30 * * * *	0/30 * * * *	[▶]	[...]
test_xcom [workshop]	airflow	0	daily		2022-11-20, 00:00:00		0/30 * * * *	[▶]	[...]

Showing 1-5 of 5 DAGs

05-creating-and-scheduling-data-pipelines > dags >  my\_dag\_w9.py

```
1  from airflow import DAG
2  from airflow.utils import timezone
3  from airflow.operators.empty import EmptyOperator
4
5  with DAG(
6      "my_dag_w9",
7      start_date = timezone.datetime(2022,11,21),
8      schedule = "*/30 * * * *",
9      tags = ["workshop"],
10 ) as dag:
11
12     start = EmptyOperator(task_id = "start")
13
```

05-creating-and-scheduling-data-pipelines > dags >  my\_dag\_w9.py

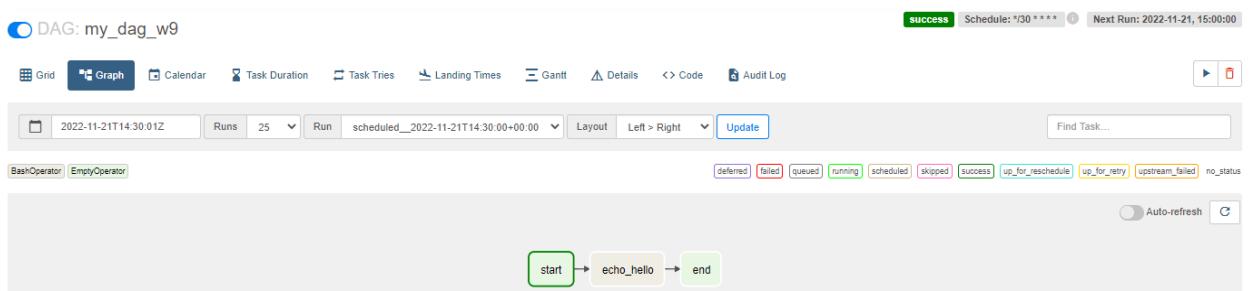
```
1  from airflow import DAG
2  from airflow.utils import timezone
3  from airflow.operators.empty import EmptyOperator
4
5  with DAG(
6      "my_dag_w9",
7      start_date = timezone.datetime(2022,11,21),
8      schedule = "*/30 * * * *",
9      tags = ["workshop"],
10 ) as dag:
11
12     start = EmptyOperator(task_id = "start")
13
14     end =  EmptyOperator(task_id = "end")
15
16     start >> end
```

DAG: my\_dag\_w9

```

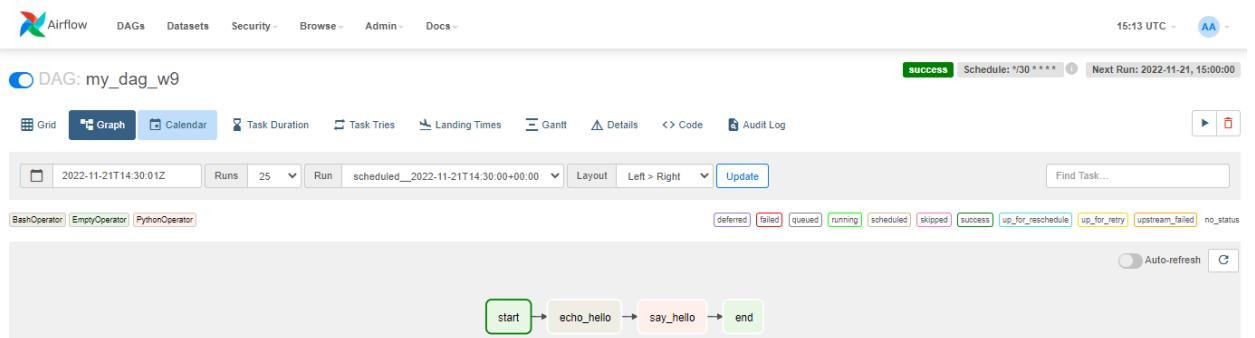
05-creating-and-scheduling-data-pipelines > dags > my_dag_w9.py
1  from airflow import DAG
2  from airflow.utils import timezone
3  from airflow.operators.empty import EmptyOperator
4  from airflow.operators.bash import BashOperator
5
6  with DAG(
7      "my_dag_w9",
8      start_date = timezone.datetime(2022,11,21),
9      schedule = "*/30 * * * *",
10     tags = ["workshop"],
11 ) as dag:
12
13     start = EmptyOperator(task_id = "start")
14
15     echo_hello = BashOperator(
16         task_id = "echo_hello",
17         bash_command = "echo 'hello'",
18     )
19
20     end = EmptyOperator(task_id = "end")
21
22     start >> echo_hello >> end
23

```



05-creating-and-scheduling-data-pipelines > dags > my\_dag\_w9.py

```
1  from airflow import DAG
2  from airflow.utils import timezone
3  from airflow.operators.empty import EmptyOperator
4  from airflow.operators.bash import BashOperator
5  from airflow.operators.python import PythonOperator
6
7  # ข้อพึงกั้นต้องไม่เหมือนชื่อ task เลยใส่ _ หน้าหัว
8
9  def _say_hello():
10     print("Hello")
11
12 with DAG(
13     "my_dag_w9",
14     start_date = timezone.datetime(2022,11,21),
15     schedule = "*/30 * * * *",
16     tags = ["workshop"],
17 ) as dag:
18
19     start = EmptyOperator(task_id = "start")
20
21     echo_hello = BashOperator(
22         task_id = "echo_hello",
23         bash_command = "echo 'hello'",
24     )
25
26     say_hello = PythonOperator(
27         task_id = "say_hello",
28         python_callable = _say_hello,
29     )
30
31     end = EmptyOperator(task_id = "end")
32
33     start >> echo_hello >> say_hello >> end
34
```



```
05-creating-and-scheduling-data-pipelines > dags > my_dag_w9.py
 1 import logging
 2
 3 from airflow import DAG
 4 from airflow.utils import timezone
 5 from airflow.operators.empty import EmptyOperator
 6 from airflow.operators.bash import BashOperator
 7 from airflow.operators.python import PythonOperator
 8
 9 #ชื่อฟังก์ชันต้องไม่เหมือนชื่อ task เมนูสี _ นำหน้า
10
11 def _say_hello():
12     print("Hello")
13
14 #logging มีระดับการ log => info, debug และแต่ละการที่ทำงาน
15 def _print_log_messages():
16     logging.info("Hello from Log")
17
18 with DAG(
19     "my_dag_w9",
20     start_date = timezone.datetime(2022,11,21),
21     schedule = "*/*30 * * * *",
22     tags = ["workshop"],
23 ) as dag:
24
25     start = EmptyOperator(task_id = "start")
26
27     echo_hello = BashOperator(
28         task_id = "echo_hello",
29         bash_command = "echo 'hello'",
30     )
31
32     say_hello = PythonOperator(
33         task_id = "say_hello",
34         python_callable = _say_hello,
35     )
36
37     print_log_messages = PythonOperator(
38         task_id = "print_log_messages",
39         python_callable = _print_log_messages,
40     )
41
42
43     end = EmptyOperator(task_id = "end")
44
45     start >> echo_hello >> say_hello >> print_log_messages >> end
```

**DAG: my\_dag\_w9**

success Schedule: \*30 \* \* \* \* Next Run: 2022-11-21, 15:00:00

Grid Graph Calendar Task Duration Task Tries Landing Times Gantt Details < Code Audit Log

2022-11-21T14:30:01Z Runs 25 Run scheduled\_2022-11-21T14:30:00+00:00 Layout Left > Right Update Find Task...  
BashOperator EmptyOperator PythonOperator deferred failed queued running scheduled skipped success up\_for\_reschedule up\_for\_retry upstream\_failed no\_status

Auto-refresh C

```
# start >> echo_hello >> say_hello >> print_log_messages >> end
start >> echo_hello >> [ say_hello , print_log_messages ] >> end
```

Task Instance Details Rendered Template Log XCom List Instances, all runs Filter Upstream

Details Logs

Task Actions

- Ignore All Deps Ignore Task State Ignore Task Deps Run
- Past Future Upstream Downstream Recursive Failed Clear
- Past Future Upstream Downstream Mark Failed
- Past Future Upstream Downstream Mark Success

Status success

Task ID say\_hello

Run ID manual\_2022-11-21T15:27:38.535289+00:00

Operator PythonOperator

Duration 00:00:00

Started 2022-11-21, 15:27:42 UTC

Ended 2022-11-21, 15:27:42 UTC

**DAG: my\_dag\_w9**

success Schedule: \*30 \* \* \* \* Next Run: 2022-11-21, 15:30:00

Grid Graph Calendar Task Duration Task Tries Landing Times Gantt Details < Code Audit Log

2022-11-21T15:31:51Z Runs 25 Run manual\_2022-11-21T15:31:50.577031+00:00 Layout Left > Right Update Find Task...  
BashOperator EmptyOperator PythonOperator deferred failed queued running scheduled skipped success up\_for\_reschedule up\_for\_retry upstream\_failed no\_status

Auto-refresh C

```
start -> echo_hello -> say_hello -> print_log_messages -> end
```

# Good Data Pipeline

It should have these 4 characteristics

1. **Reproducible:** deterministic and idempotent
2. **Future proof:** backfilling and versioning
3. **Fault tolerance:** automatic retry of failed tasks
4. **Transparent:** clarity of where data are

## Idempotent

```
● ● ●  
fruits = ["Apple", "Orange", "Grape"]  
print(fruits)  
  
def add(fruit):  
    fruits.append(fruit)  
  
add("Pineapple")  
print(fruits)
```

This code changes the value  
stored in fruits



# Idempotent

```
fruits = ["Apple", "Orange", "Grape"]
print(fruits)

def add(fruit):
    fruits.append(fruit)

add("Pineapple")
print(fruits)
```

This code changes the value stored in fruits



```
fruits = ["Apple", "Orange", "Grape"]

def add_new(fruit):
    return fruits + [fruit]

new_fruits = add_new("Pineapple")
print(fruits)
print(new_fruits)
```

This code does **NOT** change the value stored in fruits



```
fruits = ["Apple", "Orange", "Grape"]
def add(fruit):
    fruits.append(fruit)

add("Pineapple")
print (" Not Idempotent ")
print(fruits)

add("Pineapple")
add("Pineapple")
add("Pineapple")
add("Pineapple")
print(fruits)

fruits2 = ["Apple", "Orange", "Grape"]
def add_new(fruit2):
    return fruits2 + [fruit2]

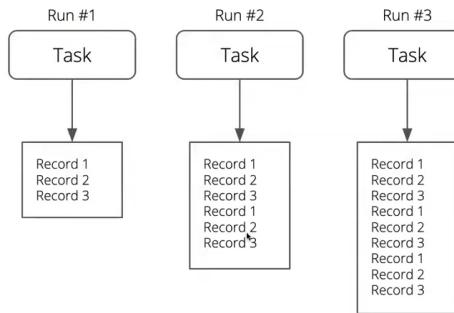
new_fruits = add_new("Pineapple")
print (" Idempotent ")
print(new_fruits)

new_fruits = add_new("Pineapple")
new_fruits = add_new("Pineapple")
new_fruits = add_new("Pineapple")
new_fruits = add_new("Pineapple")
print(new_fruits)
```

● gitpod /workspace/swu-ds525/05-creating-and-scheduling-data-pipelines/dags (main) \$ python fruits.py  
Not Idempotent  
['Apple', 'Orange', 'Grape', 'Pineapple']  
['Apple', 'Orange', 'Grape', 'Pineapple', 'Pineapple', 'Pineapple', 'Pineapple', 'Pineapple']  
Idempotent  
['Apple', 'Orange', 'Grape', 'Pineapple']  
['Apple', 'Orange', 'Grape', 'Pineapple']

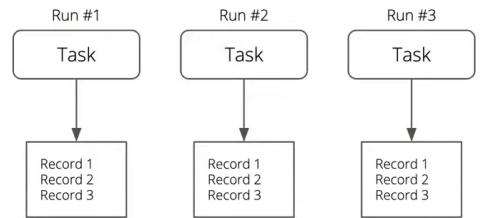
# Non-idempotent vs. Idempotent

## Non-idempotent



VS.

## Idempotent



# Making Pipeline Idempotent

```
from airflow.providers.http.operators import SimpleHttpOperator

with DAG(...) as dag:
    get_data_from_api = SimpleHttpOperator(
        task_id="get_data_from_api",
        endpoint="https://api.somewhere.io/data.json"
        "?start={{ execution_date }}"
        "&end={{ execution_date }}",
    )
```

## Templating and Macros

Way to pass dynamic data to your DAGs at runtime

### Use cases:

- Turning tasks idempotent
- Timestamp for incremental ETL
- Custom user defined parameters for complex operators

See the default variables and macros in the Macros reference in Airflow docs

# airflow template macro

The screenshot shows the Apache Airflow documentation website. At the top, there is a navigation bar with links to Community, Meetups, Documentation, Use-cases, Announcements, Blog, and Ecosystem. Below the navigation bar, there is a search bar with the placeholder "Search docs" and a magnifying glass icon. To the left of the search bar, the text "Version: 2.4.1" is displayed. On the far left, there is a sidebar titled "CONTENT" containing a list of links to various Airflow documentation pages, such as Overview, Project, License, Quick Start, Installation, Upgrading from 1.10 to 2, Tutorials, How-to Guides, UI / Screenshots, Concepts, Executor, DAG Runs, Plugins, Security, Logging & Monitoring, Time Zones, Using the CLI, Integration, Kubernetes, Lineage, Listeners, DAG Serialization, Modules Management, Release Policies, Release Notes, Best Practices, Production Deployment, FAQ, and Privacy Notice. Below this, there is a section titled "REFERENCES" with links to Operators and hooks and CLI. The main content area has a breadcrumb navigation path "Home / Templates reference". The title "Templates reference" is bolded. Below the title, there is a note: "Variables, macros and filters can be used in templates (see the [Jinja Templating](#) section)". Another note states: "The following come for free out of the box with Airflow. Additional custom macros can be added globally through [Plugins](#), or at a DAG level through the `DAG.user_defined_macros` argument.". A section titled "Variables" is present, stating: "The Airflow engine passes a few variables by default that are accessible in all templates". A table titled "Variables" provides a detailed list of these variables:

Variable	Description
<code>{{ data_interval_start }}</code>	Start of the data interval ( <code>pendulum.DateTime</code> ).
<code>{{ data_interval_end }}</code>	End of the data interval ( <code>pendulum.DateTime</code> ).
<code>{{ ds }}</code>	The DAG run's logical date as <code>YYYY-MM-DD</code> . Same as <code>{{ dag_run.logical_date   ds }}</code> .
<code>{{ ds_nodash }}</code>	Same as <code>{{ dag_run.logical_date   ds_nodash }}</code> .
<code>{{ ts }}</code>	Same as <code>{{ dag_run.logical_date   ts }}</code> . Example: <code>2018-01-01T00:00:00+00:00</code> .
<code>{{ ts_nodash_with_tz }}</code>	Same as <code>{{ dag_run.logical_date   ts_nodash_with_tz }}</code> . Example: <code>20180101T000000+0000</code> .
<code>{{ ts_nodash }}</code>	Same as <code>{{ dag_run.logical_date   ts_nodash }}</code> . Example: <code>20180101T000000</code> .
<code>{{ prev_data_interval_start_success }}</code>	Start of the data interval from prior successful DAG run ( <code>pendulum.DateTime</code> or <code>None</code> ).
<code>{{ prev_data_interval_end_success }}</code>	End of the data interval from prior successful DAG run ( <code>pendulum.DateTime</code> or <code>None</code> ).
<code>{{ prev_start_date_success }}</code>	Start date from prior successful dag run (if available) ( <code>pendulum.DateTime</code> or <code>None</code> ).
<code>{{ dag }}</code>	The DAG object.
<code>{{ task }}</code>	The Task object.
<code>{{ macros }}</code>	A reference to the <code>macros</code> package, described below.

## Variables

The Airflow engine passes a few variables by default that are accessible in all templates

Variable	Description
<code>{{ data_interval_start }}</code>	Start of the data interval ( <code>pendulum.DateTime</code> ).
<code>{{ data_interval_end }}</code>	End of the data interval ( <code>pendulum.DateTime</code> ).
<code>{{ ds }}</code>	The DAG run's logical date as <code>YYYY-MM-DD</code> . Same as <code>{{ dag_run.logical_date   ds }}</code> .
<code>{{ ds_nodash }}</code>	Same as <code>{{ dag_run.logical_date   ds_nodash }}</code> .
<code>{{ ts }}</code>	Same as <code>{{ dag_run.logical_date   ts }}</code> . Example: <code>2018-01-01T00:00:00+00:00</code> .
<code>{{ ts_nodash_with_tz }}</code>	Same as <code>{{ dag_run.logical_date   ts_nodash_with_tz }}</code> . Example: <code>20180101T000000+0000</code>
<code>{{ ts_nodash }}</code>	Same as <code>{{ dag_run.logical_date   ts_nodash }}</code> . Example: <code>20180101T000000</code> .
<code>{{ prev_data_interval_start_success }}</code>	Start of the data interval from prior successful DAG run ( <code>pendulum.DateTime</code> or <code>None</code> ).
<code>{{ prev_data_interval_end_success }}</code>	End of the data interval from prior successful DAG run ( <code>pendulum.DateTime</code> or <code>None</code> ).
<code>{{ prev_start_date_success }}</code>	Start date from prior successful dag run (if available) ( <code>pendulum.DateTime</code> or <code>None</code> ).
<code>{{ dag }}</code>	The DAG object.
<code>{{ task }}</code>	The Task object.

```
05-creating-and-scheduling-data-pipelines > dags > my_dag_w9.py
 1 import logging
 2
 3 from airflow import DAG
 4 from airflow.utils import timezone
 5 from airflow.operators.empty import EmptyOperator
 6 from airflow.operators.bash import BashOperator
 7 from airflow.operators.python import PythonOperator
 8
 9 #ชื่อฟังก์ชันต้องไม่เหมือนชื่อ task เลยใช่ _ นำหน้า
10
11 def _say_hello():
12     print("Hello")
13
14 #logging มีระดับของ log => info, debug และการทิ้งงาน
15 def _print_log_messages():
16     logging.info("Hello from Log")
17
18 with DAG(
19     "my_dag_w9",
20     start_date = timezone.datetime(2022,11,21),
21     schedule = "*/*/*/*/*",
22     tags = ["workshop"],
23     catchup = False,
24 ) as dag:
25
26     start = EmptyOperator(task_id = "start")
27
28     echo_hello = BashOperator(
29         task_id = "echo_hello",
30         bash_command = "echo 'hello' on {{ ds }}",
31     )
32
33     say_hello = PythonOperator(
34         task_id = "say_hello",
35         python_callable = _say_hello,
36     )
37
38     print_log_messages = PythonOperator(
39         task_id = "print_log_messages",
40         python_callable = _print_log_messages,
41     )
42
43     end = EmptyOperator(task_id = "end")
44
45     start >> echo_hello >> say_hello >> print_log_messages >> end
46
47 #    start >> echo_hello >> [ say_hello , print_log_messages ] >> end
48
49
50
```

Airflow DAGs Datasets Security Browse Admin Docs 16:13 UTC AA

Grid Graph Calendar Task Duration Task Tries Landing Times Gantt Details Code Audit Log

11/21/2022 04:11:59 PM 25 All Run Types All Run States Clear Filters

deferred failed queued running scheduled skipped success up\_for\_reschedule up\_for\_retry upstream\_failed no\_status

Auto-refresh

Duration Nov 21, 06:00 Nov 21, 11:00 Nov 21, 15:00

00:00:07  
00:00:03  
00:00:00

start echo\_hello say\_hello print\_log\_messages end

DAG my\_dag\_w9 / Run 2022-11-21, 16:00:00 UTC / Task echo\_hello

Task Instance Details Rendered Template Log XCom List Instances, all runs Filter Upstream

Details Logs

(by attempts) 1

All Levels All File Sources Wrap Full Logs Download See More

```
[2022-11-21, 16:12:01 UTC] [taskmixin.py:205] WARNING - Dependency <Task_PythonDecoratedOperator>; transform, load alre...[2022-11-21, 16:12:01 UTC] [taskmixin.py:205] WARNING - Dependency <Task_PythonDecoratedOperator>; transform, load alre...[2022-11-21, 16:12:01 UTC] [taskmixin.py:205] WARNING - Dependency <Task_PythonDecoratedOperator>; transform, load alre...[2022-11-21, 16:12:01 UTC] [taskmixin.py:205] WARNING - Dependency <Task_PythonDecoratedOperator>; transform, load alre...[2022-11-21, 16:12:01 UTC] [taskmixin.py:205] WARNING - Dependency <Task_PythonDecoratedOperator>; transform, load alre...[2022-11-21, 16:12:01 UTC] [task_command.py:384] INFO - Running <TaskInstance: my_dag_w9.echo_hello.manual_2022-11-21T16:12:01>[2022-11-21, 16:12:01 UTC] [taskinstance.py:1592] INFO - Exporting the following env vars:[AIRFLOW_CTX_DAG_ID=my_dag_w9][AIRFLOW_CTX_TASK_ID=echo_hello][AIRFLOW_CTX_EXECUTION_DATE=2022-11-21T16:11:58.955654+00:00][AIRFLOW_CTX_TRY_NUMBER=1][AIRFLOW_CTX_DAG_RUN_ID=manual_2022-11-21T16:11:58.955654+00:00][2022-11-21, 16:12:01 UTC] [subprocess.py:63] INFO - Tmp dir root location: /tmp[2022-11-21, 16:12:01 UTC] [subprocess.py:75] INFO - Running command: ['>>> echo "Hello" on 2022-11-21'...][2022-11-21, 16:12:01 UTC] [subprocess.py:86] INFO - Output: Hello[2022-11-21, 16:12:01 UTC] [subprocess.py:87] INFO - Command exited with return code 0[2022-11-21, 16:12:01 UTC] [taskinstance.py:1406] INFO - Marking task as SUCCESS, dag_id=my_dag_w9, task_id=echo_hello, ex...[2022-11-21, 16:12:01 UTC] [local_task_job.py:164] INFO - Task exited with return code 0[2022-11-21, 16:12:01 UTC] [local_task_job.py:273] INFO - 1 downstream tasks scheduled from follow-on schedule check
```

```
def _say_hello(**context):
    print(context)
    print("Hello")
```

Airflow DAGs Datasets Security Browse Admin Docs 16:18 UTC AA

Grid Graph Calendar Task Duration Task Tries Landing Times Gantt Details Code Audit Log

11/21/2022 04:18:32 PM 25 All Run Types All Run States Clear Filters

deferred failed queued running scheduled skipped success up\_for\_reschedule up\_for\_retry upstream\_failed no\_status

Auto-refresh

Duration Nov 21, 06:30 Nov 21, 11:30 Nov 21, 15:30

00:00:07  
00:00:03  
00:00:00

start echo\_hello say\_hello print\_log\_messages end

DAG my\_dag\_w9 / Run 2022-11-21, 16:00:00 UTC / Task say\_hello

Task Instance Details Rendered Template Log XCom List Instances, all runs Filter Upstream

Details Logs

(by attempts) 1

All Levels All File Sources Wrap Full Logs Download See More

```
[2022-11-21, 16:18:35 UTC] [taskmixin.py:205] WARNING - Dependency <Task_PythonDecoratedOperator>; load, transform, load alre...[2022-11-21, 16:18:35 UTC] [taskmixin.py:205] WARNING - Dependency <Task_PythonDecoratedOperator>; transform, load alre...[2022-11-21, 16:18:35 UTC] [taskmixin.py:205] WARNING - Dependency <Task_PythonDecoratedOperator>; load, transform, load alre...[2022-11-21, 16:18:35 UTC] [taskmixin.py:205] WARNING - Dependency <Task_PythonDecoratedOperator>; transform, load alre...[2022-11-21, 16:18:35 UTC] [taskmixin.py:205] WARNING - Dependency <Task_PythonDecoratedOperator>; transform, load alre...[2022-11-21, 16:18:35 UTC] [taskmixin.py:205] WARNING - Dependency <Task_PythonDecoratedOperator>; transform, load alre...[2022-11-21, 16:18:35 UTC] [taskmixin.py:205] WARNING - Dependency <Task_PythonDecoratedOperator>; transform, load alre...[2022-11-21, 16:18:35 UTC] [task_command.py:384] INFO - Running <TaskInstance: my_dag_w9.say_hello.manual_2022-11-21T16:18:35>[2022-11-21, 16:18:35 UTC] [taskinstance.py:1592] INFO - Exporting the following env vars:[AIRFLOW_CTX_DAG_ID=my_dag_w9][AIRFLOW_CTX_TASK_ID=say_hello][AIRFLOW_CTX_EXECUTION_DATE=2022-11-21T16:18:31.924719+00:00][AIRFLOW_CTX_TRY_NUMBER=1][AIRFLOW_CTX_DAG_RUN_ID=manual_2022-11-21T16:18:31.924719+00:00][2022-11-21, 16:18:35 UTC] [log_subscriber.py:117] INFO - Configuring AirflowConfigParser object at 0x7f33...[2022-11-21, 16:18:36 UTC] [logging_mixin.py:117] INFO - Hello[2022-11-21, 16:18:36 UTC] [python.py:177] INFO - Done. Returned value was: None[2022-11-21, 16:18:36 UTC] [taskinstance.py:1406] INFO - Marking task as SUCCESS, dag_id=my_dag_w9, task_id=say_hello, ex...[2022-11-21, 16:18:36 UTC] [local_task_job.py:164] INFO - Task exited with return code 0[2022-11-21, 16:18:36 UTC] [local_task_job.py:273] INFO - 1 downstream tasks scheduled from follow-on schedule check
```

The screenshot shows the Airflow web interface. At the top, there are navigation links: DAGs, Datasets, Security, Browse, Admin, Docs, and a timestamp of 16:23 UTC. On the far right, there's a user icon.

The main area displays a DAG run timeline from Nov 21, 11:30 to Nov 21, 15:31. The timeline shows tasks starting at 00:00:00 and ending at 00:00:07. Tasks include 'start', 'echo\_hello', 'say\_hello' (which is highlighted in blue), 'print\_log\_messages', and 'end'. Each task has a series of green squares representing attempts.

To the right of the timeline, there's a sidebar with 'Task Instance Details', 'Rendered Template', 'Log', 'XCom', 'List Instances, all runs', and 'Filter Upstream' buttons. Below this, tabs for 'Details' and 'Logs' are shown, with 'Logs' being active. A dropdown menu '(by attempts)' is open, showing options 1 and 2.

The 'Logs' section shows log entries for the 'say\_hello' task. The log entries are as follows:

```
[2022-11-21, 16:23:03 UTC] [taskmixin.py:205] WARNING - Dependency <Task(_PythonDecoratedOperator): load>, transform alres
[2022-11-21, 16:23:03 UTC] [taskmixin.py:205] WARNING - Dependency <Task(_PythonDecoratedOperator): transform>, load alre
[2022-11-21, 16:23:03 UTC] [taskmixin.py:205] WARNING - Dependency <Task(_PythonDecoratedOperator): load>, transform alre
[2022-11-21, 16:23:03 UTC] [taskmixin.py:205] WARNING - Dependency <Task(_PythonDecoratedOperator): transform>, load alre
[2022-11-21, 16:23:03 UTC] [taskmixin.py:205] WARNING - Dependency <Task(_PythonDecoratedOperator): load>, transform alre
[2022-11-21, 16:23:03 UTC] [taskmixin.py:205] WARNING - Dependency <Task(_PythonDecoratedOperator): transform>, load alre
[2022-11-21, 16:23:03 UTC] [taskmixin.py:205] WARNING - Dependency <Task(_PythonDecoratedOperator): load>, transform alre
[2022-11-21, 16:23:03 UTC] [taskmixin.py:205] WARNING - Dependency <Task(_PythonDecoratedOperator): transform>, load alre
[2022-11-21, 16:23:03 UTC] [taskmixin.py:205] WARNING - Dependency <Task(_PythonDecoratedOperator): load>, transform alre
[2022-11-21, 16:23:03 UTC] [taskmixin.py:205] WARNING - Dependency <Task(_PythonDecoratedOperator): transform>, load alre
[2022-11-21, 16:23:03 UTC] [taskmixin.py:205] WARNING - Dependency <Task(_PythonDecoratedOperator): load>, transform alre
[2022-11-21, 16:23:03 UTC] [taskmixin.py:205] WARNING - Dependency <Task(_PythonDecoratedOperator): transform>, load alre
[2022-11-21, 16:23:03 UTC] [taskmixin.py:205] WARNING - Dependency <Task(_PythonDecoratedOperator): load>, transform alre
[2022-11-21, 16:23:03 UTC] [taskmixin.py:205] WARNING - Dependency <Task(_PythonDecoratedOperator): transform>, load alre
[2022-11-21, 16:23:03 UTC] [task_command.py:384] INFO - Running <TaskInstance: my_dag_w9.say_hello manual_2022-11-21T16:18:31.924719+00:00>
[2022-11-21, 16:23:03 UTC] [taskinstance.py:1592] INFO - Exporting the following env vars:
AIRFLOW_CTX_DAG_OWNER=**
AIRFLOW_CTX_DAG_ID=my_dag_w9
AIRFLOW_CTX_TASK_ID=say_hello
AIRFLOW_CTX_EXECUTION_DATE=2022-11-21T16:18:31.924719+00:00
AIRFLOW_CTX_TRY_NUMBER=2
AIRFLOW_CTX_DAG_RUN_ID=manual_2022-11-21T16:18:31.924719+00:00
[2022-11-21, 16:23:03 UTC] [logging_mixin.py:117] INFO - {'conf': '**.configuration.AirflowConfigParser object at 0x7f43'}
[2022-11-21, 16:23:03 UTC] [logging_mixin.py:117] INFO - Hello! on 2022-11-21
[2022-11-21, 16:23:03 UTC] [python.py:177] INFO - Done. Returned value was: None
[2022-11-21, 16:23:03 UTC] [taskinstance.py:1406] INFO - Marking task as SUCCESS. dag_id=my_dag_w9, task_id=say_hello, ex
[2022-11-21, 16:23:03 UTC] [local_task_job.py:164] INFO - Task exited with return code 0
[2022-11-21, 16:23:03 UTC] [local_task_job.py:273] INFO - 1 downstream tasks scheduled from follow-on schedule check
```

```
def _say_hello(**context):
    print(context)
    datestamp = context["ds"]
    print(f"Hello! on {datestamp}")
```

## BashOperator airflow

All

Shopping

Videos

News

Images

More

About 28,800 results (0.36 seconds)

<https://airflow.apache.org> › howto › operator › bash

⋮

## [BashOperator — Airflow Documentation](#)

Use the **BashOperator** to execute commands in a Bash shell. `airflow/example_dags/example_bash_operator.py`[source]. `run_this = BashOperator ...`

You've visited this page 4 times. Last visit: 10/13/22

The screenshot shows the Apache Airflow documentation for the `airflow.operators.bash` module. The top navigation bar includes links for Community, Meetups, Documentation, Use-cases, Announcements, Blog, and Ecosystem. A sidebar on the left lists various documentation sections such as Overview, Project, License, Quick Start, Installation, Upgrading from 1.10 to 2, Tutorials, How-to Guides, UI / Screenshots, Concepts, Executor, DAG Runs, Plugins, Security, Logging & Monitoring, Time Zones, Using the CLI, Integration, Kubernetes, Lineage, Listeners, DAG Serialization, Modules Management, Release Policies, Release Notes, Best Practices, Production Deployment, FAQ, Privacy Notice, and References. The main content area displays the `airflow.operators.bash` module contents, showing the `BashOperator` class which executes a Bash script or command set. The code snippet for the class definition is:

```
class airflow.operators.bash.BashOperator( ..., bash_command, env=None, append_env=False, output_encoding='utf-8', skip_exit_code=99, cwd=None, **kwargs) [source]
```

The `BashOperator` class inherits from `airflow.models.baseoperator.BaseOperator`. The `bash_command` parameter is described as executing a Bash script or command set. The `env` parameter is a dict or str, defining environment variables for the new process. The `append_env` parameter is a boolean indicating whether to inherit current process environment variables. The `output_encoding` parameter specifies the encoding for the output. The `skip_exit_code` parameter is an int, and the `cwd` parameter is a str, both used for specifying the working directory.

The right sidebar shows the module contents for `airflow.operators.bash`, listing classes like `BashOperator`, `template_fields`, `template_field_renderers`, `template_ext`, `ui_color`, `subprocess_hook`, `get_env`, `execute`, and `on_kill`.

`template_fields:Sequence[str] = ['templates_dict', 'op_args', 'op_kwargs']` [source] ↗

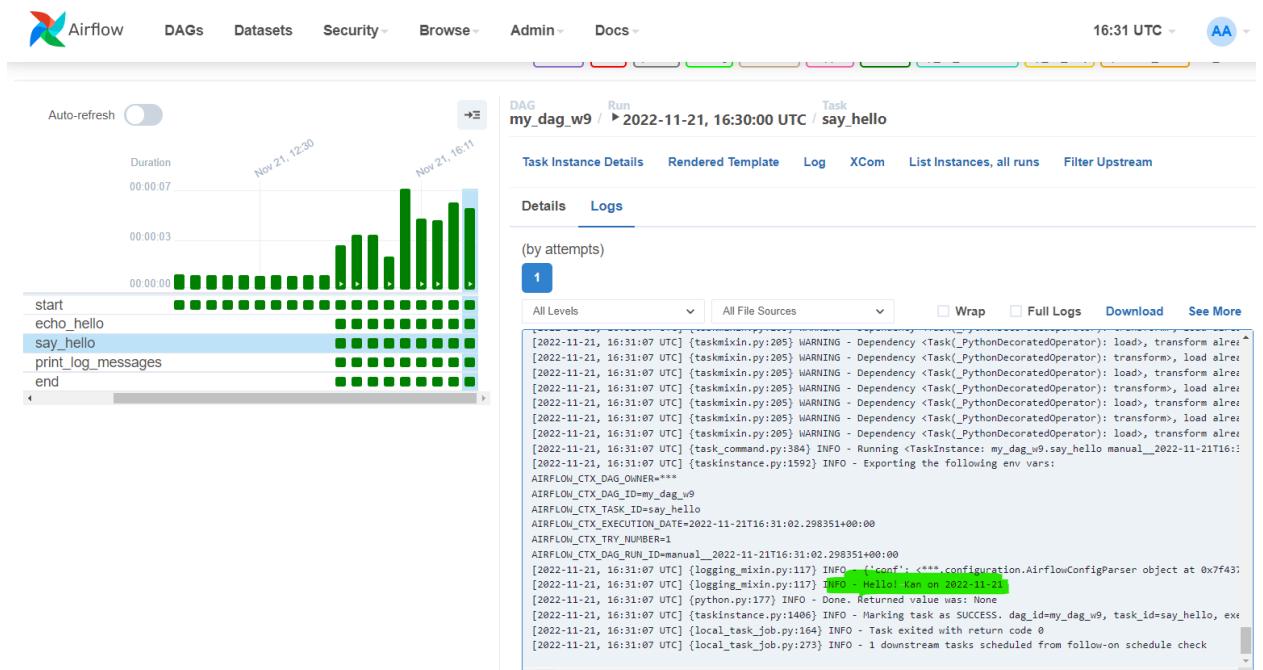
```
def _say_hello( name="", **context ):
    print(context)
    datestamp = context["ds"]
    print(f"Hello! {name} on {datestamp}")

#logging มีระดับการ log => info, debug และแม่ก้ารที่งาน
def _print_log_messages():
    logging.info("Hello from Log")

with DAG(
    "my_dag_w9",
    start_date = timezone.datetime(2022,11,21),
    schedule = "*/*30 * * * *",
    tags = ["workshop"],
    catchup = False,
) as dag:
    start = EmptyOperator(task_id = "start")

    echo_hello = BashOperator(
        task_id = "echo_hello",
        bash_command = "echo 'hello' on {{ ds }}"
    )

# op_kwargs รับข้อมูลเป็น dict
    say_hello = PythonOperator(
        task_id = "say_hello",
        python_callable = _say_hello,
        op_kwargs = {
            "name": "Kan",
        }
    )
```



```
def _say_hello( name="", **context ):
    print(context)
    datestamp = context["ds"]
    print(f"Hello! {name} on {datestamp}")

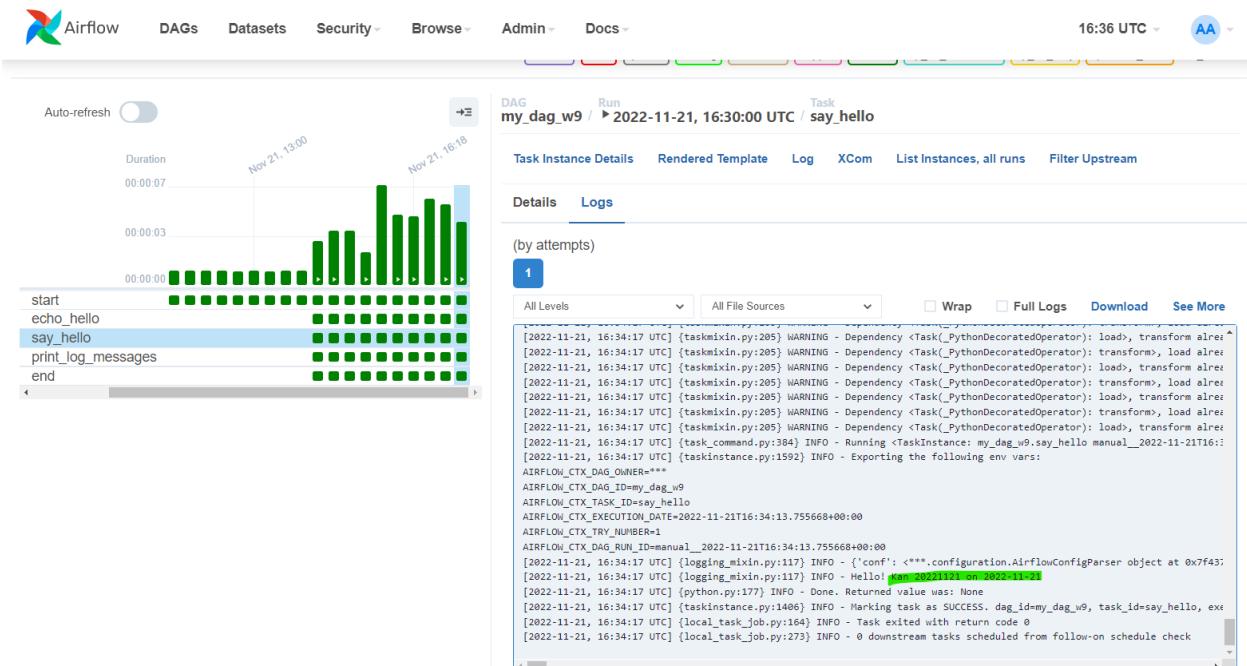
#logging မြန်တံ့ခါန log => info, debug မေးမားမြန်တံ့ခါနရှိခဲ့ခြင်း
def _print_log_messages():
    logging.info("Hello from Log")

with DAG(
    "my_dag_w9",
    start_date = timezone.datetime(2022,11,21),
    schedule = "*/30 * * * *",
    tags = ["workshop"],
    catchup = False,
) as dag:

    start = EmptyOperator(task_id = "start")

    echo_hello = BashOperator(
        task_id = "echo_hello",
        bash_command = "echo 'hello' on {{ ds }}",
    )

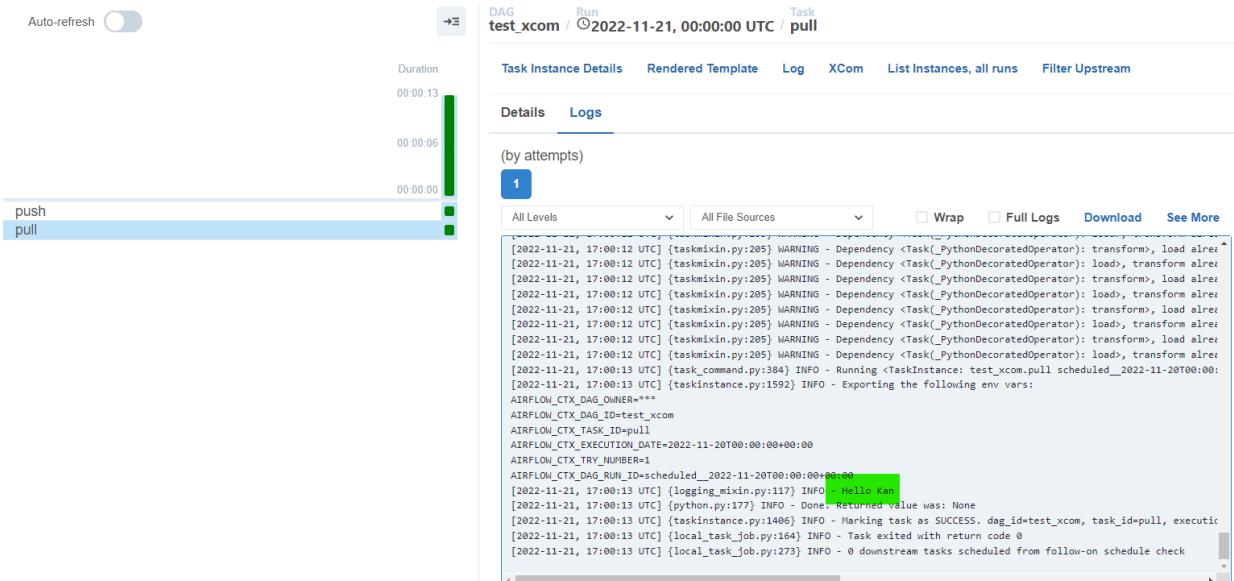
    # op_kwargs သည်ပုံစံများဖြစ်သော dict
    say_hello = PythonOperator(
        task_id = "say_hello",
        python_callable = _say_hello,
        op_kwargs = {
            "name": f"Kan {{ ds_nodash }}",
        }
    )
```



```

05-creating-and-scheduling-data-pipelines > dags > test_xcom.py
 1  from airflow import DAG
 2  from airflow.utils import timezone
 3  from airflow.operators.python import PythonOperator
 4
 5  #task instance မရ၏ context
 6  def _push(**context):
 7      ti = context["ti"]
 8      ti.xcom_push(key="name", value="Kan")
 9
10     #return ["Kan", "Chin", "Peeyapak", "Lyn", "Peerawit", "Kruewan", "Pongthanin", "Mindii", "Nuttapol", "Nattakan",]
11
12  def _pull(**context):
13      ti = context["ti"]
14      name = ti.xcom_pull(task_ids="push", key="name")
15      print(f"Hello {name}")
16
17  with DAG(
18      "test_xcom",
19      start_date=timezone.datetime(2022, 10, 15),
20      schedule="@daily",
21      tags=["workshop"],
22      catchup=False,
23  ) as dag:
24
25      push = PythonOperator(
26          task_id="push",
27          python_callable=_push,
28      )
29
30      pull = PythonOperator(
31          task_id="pull",
32          python_callable=_pull,
33      )
34
35      push >> pull

```



Airflow DAGs Datasets Security Browse Admin Docs

Auto-refresh

Duration  
00:00:13   
00:00:06

Variables Configurations Connections Plugins Providers Pools XComs

Airflow DAGs Datasets Security Browse Admin Docs 17:04 UTC AA

List XComs

Search ▾

« < 1 2 3 4 5 6 7 > » Page size Actions Record Count: 1457

Key	Value	Timestamp	Dag Id	Task Id	Run Id	Map Index	Execution Date
return_value	hello on 2022-11-21	2022-11-21, 17:00:06	my_dag_w9	echo_hello	scheduled_2022-11-21T16:30:00+00:00		2022-11-21, 16:30:00
return_value	hello on 2022-11-21	2022-11-21, 17:00:06	my_dag2	echo_hello	scheduled_2022-11-21T16:30:00+00:00		2022-11-21, 16:30:00
name	Kan	2022-11-21, 17:00:06	test_xcom	push	scheduled_2022-11-20T00:00:00+00:00		2022-11-20, 00:00:00

```
05-creating-and-scheduling-data-pipelines > dags > test_xcom.py
 1  from airflow import DAG
 2  from airflow.utils import timezone
 3  from airflow.operators.python import PythonOperator
 4
 5  #task instance မှတ်၏ context
 6  def _push(**context):
 7      # ti = context["ti"]
 8      # ti.xcom_push(key="name", value="Kan")
 9
10     #return ["Kan", "Chin", "Peeyapak", "Lyn", "Peerawit", "Kri"]
11     return "Karn"
12
13  def _pull(**context):
14      ti = context["ti"]
15      name = ti.xcom_pull(task_ids="push", key="return_value")
16      print(f"Hello {name}")
17
18  with DAG(
19      "test_xcom",
20      start_date=timezone.datetime(2022, 10, 15),
21      schedule="@daily",
22      tags=["workshop"],
23      catchup=False,
24  ) as dag:
25
26      push = PythonOperator(
27          task_id="push",
28          python_callable=_push,
29      )
30
31      pull = PythonOperator(
32          task_id="pull",
33          python_callable=_pull,
34      )
35
36      push >> pull
```

 DAG: test\_xcom

Grid

Graph

 Calendar

## Task Duration

## Task Tries

 Landi

11/21/2022 05:08:44 PM

1

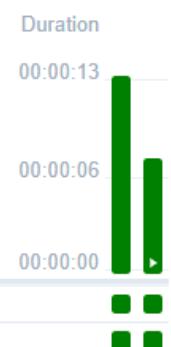
25

## All Run Types

All Run States ▾

Auto-refresh

三



push  
pull

Auto-refresh

push  
pull

DAG Run Task  
test\_xcom / ► 2022-11-21, 00:00:00 UTC / pull

Task Instance Details    Rendered Template    Log    XCom    List Instances, all runs    Filter Upstream

Details Log

(by attempts

List XComs

Search:

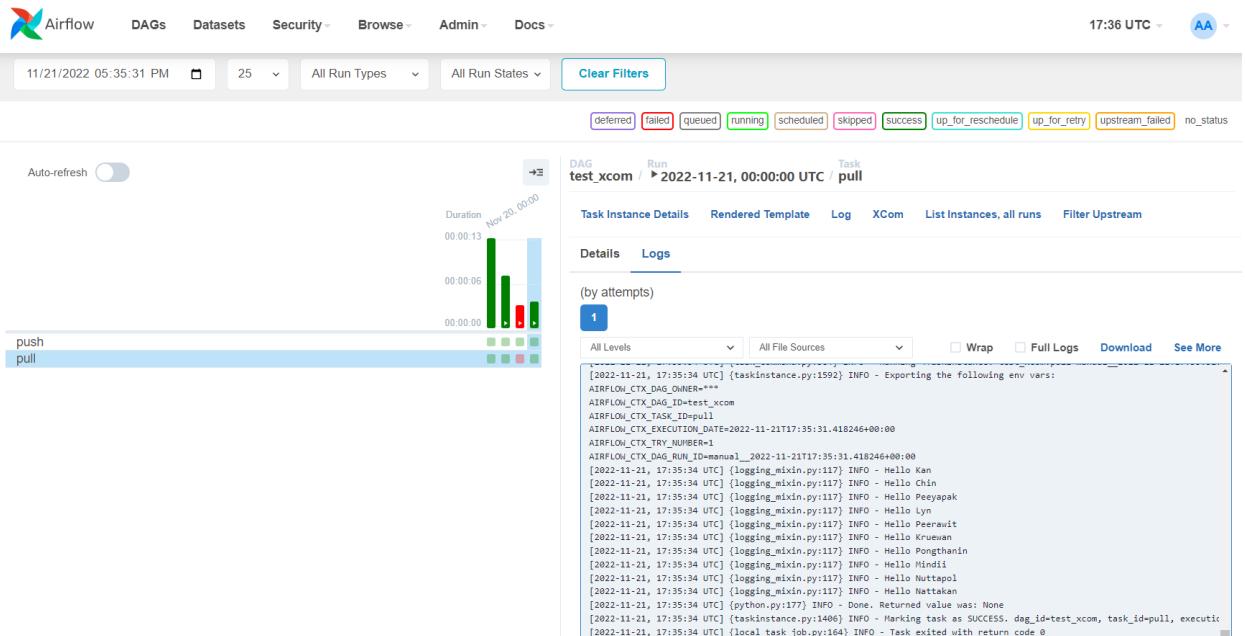
	Key	Value	Timestamp	Dag Id	Task Id	Run Id	Map Index	Execution Date
<input type="checkbox"/>	<input type="checkbox"/> return_value	Karn	2022-11-21, 17:07:46	test_xcom	push	manual__2022-11-21T17:07:40.630376+00:00		2022-11-21, 17:07:40
<input type="checkbox"/>	<input type="checkbox"/> return_value	hello on 2022-11-21	2022-11-21, 17:00:06	my_dag_w9	echo_hello	scheduled__2022-11-21T16:30:00+00:00		2022-11-21, 16:30:00
<input type="checkbox"/>	<input type="checkbox"/> return_value	hello on 2022-11-21	2022-11-21, 17:00:06	my_dag2	echo_hello	scheduled__2022-11-21T16:30:00+00:00		2022-11-21, 16:30:00

Record Count: 1458

```

05-creating-and-scheduling-data-pipelines > dags > test_xcom.py
1  from airflow import DAG
2  from airflow.utils import timezone
3  from airflow.operators.python import PythonOperator
4
5  #task instance မေတ္တာ context
6  def _push(**context):
7      # ti = context["ti"]
8      # ti.xcom_push(key="name", value="Karn")
9
10     return ["Karn", "Chin", "Peeyapak", "Lyn", "Peerawit", "Kruewan", "Pongthanin", "Mindii", "Nuttapol", "Nattakan",]
11     #return "Karn"
12
13 def _pull(**context):
14     ti = context["ti"]
15     name = ti.xcom_pull(task_ids="push", key="return_value")
16
17     for each in name:
18         print(f"Hello {each}")
19
20
21 with DAG(
22     "test_xcom",
23     start_date=timezone.datetime(2022, 10, 15),
24     schedule="@daily",
25     tags=["workshop"],
26     catchup=False,
27 ) as dag:
28
29     push = PythonOperator(
30         task_id="push",
31         python_callable=_push,
32     )
33
34     pull = PythonOperator(
35         task_id="pull",
36         python_callable=_pull,
37     )
38
39     push >> pull

```



List XComs

Search ▾

Record Count: 1462

	Key	Value	Timestamp	Dag ID	Task ID	Run ID	Map Index	Execution Date
<input type="checkbox"/>	<input type="checkbox"/>	['Kan', 'Chin', 'Peeyapak', 'Lyn', 'Peerawit', 'Kruewan', 'Pongthanin', 'Mindi', 'Nuttapol', 'Nattakan']	2022-11-21, 17:35:33	test_xcom	push	manual__2022-11-21T17:35:31.418246+00:00		2022-11-21, 17:35:31
<input type="checkbox"/>	<input type="checkbox"/>	['Kan', 'Chin', 'Peeyapak', 'Lyn', 'Peerawit', 'Kruewan', 'Pongthanin', 'Mindi', 'Nuttapol', 'Nattakan']	2022-11-21, 17:34:28	test_xcom	push	manual__2022-11-21T17:34:26.298463+00:00		2022-11-21, 17:34:26

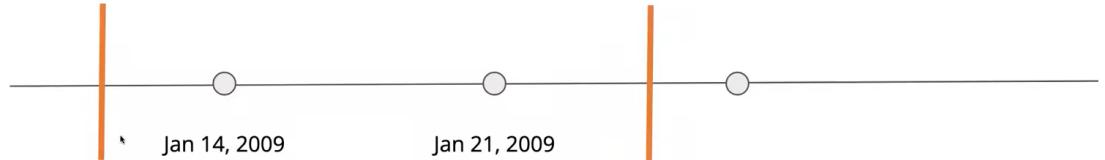
## Using Backfill in Airflow

Running the backfill of the test\_dag with the execution date set to 2019-01-01 and end date set to 2019-01-04

```
airflow dags backfill -s 2019-01-01 -e 2019-01-04 test_dag
```

# Backfill Explained

```
airflow dags backfill -s 2009-01-10 -e 2009-01-25 some_data_pipeline
```



```
05-creating-and-scheduling-data-pipelines > dags > test_xcom.py
 1  from airflow import DAG
 2  from airflow.utils import timezone
 3  from airflow.operators.python import PythonOperator
 4
 5  #task instance များ၏ context
 6  def _push(**context):
 7      # ti = context["ti"]
 8      # ti.xcom_push(key="name", value="Kan")
 9
10     return ["Kan", "Chin", "Peeyapak", "Lyn", "Peerawit", "Kruewan", "Pongthanin", "Mindii", "Nuttapol", "Nattakan",]
11
12
13  def _pull(**context):
14      ti = context["ti"]
15      ds = context["ds"]
16
17      name = ti.xcom_pull(task_ids="push", key="return_value")
18
19      for each in name:
20          print(f"Hello {each} on {ds}")
21
22  with DAG(
23      "test_xcom",
24      start_date=timezone.datetime(2022, 10, 15),
25      schedule="@daily",
26      tags=["workshop"],
27      catchup=False,
28  ) as dag:
29
30      push = PythonOperator(
31          task_id="push",
32          python_callable=_push,
33      )
34
35      pull = PythonOperator(
36          task_id="pull",
37          python_callable=_pull,
38      )
39
40      push >> pull
```

Airflow DAGs Datasets Security Browse Admin Docs 17:44 UTC AA

Grid Graph Calendar Task Duration Task Tries Landing Times Gantt Details Code Audit Log

11/21/2022 05:43:53 PM 25 All Run Types All Run States Clear Filters deferred failed queued running scheduled skipped success up\_for\_reschedule up\_for\_retry upstream\_failed no\_status

Auto-refresh

Duration Nov 21, 17:07

push pull

DAG test\_xcom Run 2022-11-21, 00:00:00 UTC Task pull

Task Instance Details Rendered Template Log XCom List Instances, all runs Filter Upstream

Details Logs (by attempts) 1

All Levels All File Sources Wrap Full Logs Download See More

```
[2022-11-21, 17:43:59 UTC] {taskinstance.py:1592} INFO - Exporting the following env vars:
AIRFLOW_CTX_DAG_OWNER=***
AIRFLOW_CTX_DAG_ID=test_xcom
AIRFLOW_CTX_TASK_ID=pull
AIRFLOW_CTX_EXECUTION_DATE=2022-11-21T17:43:52.853725+00:00
AIRFLOW_CTX_TRY_NUMBER=1
AIRFLOW_CTX_RUN_ID=manual_2022-11-21T17:43:52.853725+00:00
[2022-11-21, 17:43:59 UTC] {logging_mixin.py:117} INFO - Hello Kan on 2022-11-21
[2022-11-21, 17:43:59 UTC] {logging_mixin.py:117} INFO - Hello Chin on 2022-11-21
[2022-11-21, 17:43:59 UTC] {logging_mixin.py:117} INFO - Hello Peeyapak on 2022-11-21
[2022-11-21, 17:43:59 UTC] {logging_mixin.py:117} INFO - Hello Lynn on 2022-11-21
[2022-11-21, 17:43:59 UTC] {logging_mixin.py:117} INFO - Hello Peerawit on 2022-11-21
[2022-11-21, 17:43:59 UTC] {logging_mixin.py:117} INFO - Hello Kruewan on 2022-11-21
[2022-11-21, 17:43:59 UTC] {logging_mixin.py:117} INFO - Hello Pongthanin on 2022-11-21
[2022-11-21, 17:43:59 UTC] {logging_mixin.py:117} INFO - Hello Mindii on 2022-11-21
[2022-11-21, 17:43:59 UTC] {logging_mixin.py:117} INFO - Hello Nuttapol on 2022-11-21
[2022-11-21, 17:43:59 UTC] {logging_mixin.py:117} INFO - Hello Nattakan on 2022-11-21
[2022-11-21, 17:43:59 UTC] {python.py:177} INFO - Done. Returned value was: None
[2022-11-21, 17:43:59 UTC] {taskinstance.py:1466} INFO - Marking task as SUCCESS. dag_id=test_xcom, task_id=pull, executi
[2022-11-21, 17:44:00 UTC] {local_task_job.py:164} INFO - Task exited with return code 0
[2022-11-21, 17:44:00 UTC] {local_task_job.py:273} INFO - 0 downstream tasks scheduled from follow-on schedule check
```

gitpod /workspace/swu-ds525/05-creating-and-scheduling-data-pipelines (main) \$ docker ps

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
a85f837c40cc	apache/airflow:2.4.1	"/usr/bin/durb-init .."	3 hours ago	Up 3 hours (healthy)	0.0.0.0:8080->8080/tcp, :::8080->8080/tcp	05-creating-and-scheduling-data-pipelines-airflow-webserver-1
109ae6d1e5bc	apache/airflow:2.4.1	"/usr/bin/durb-init .."	3 hours ago	Up 3 hours (healthy)	8080/tcp	05-creating-and-scheduling-data-pipelines-airflow-triggerer-1
26bbdaeb7c3a	apache/airflow:2.4.1	"/usr/bin/durb-init .."	3 hours ago	Up 3 hours (healthy)	8080/tcp	05-creating-and-scheduling-data-pipelines-airflow-scheduler-1
65acd4300294	adminer:4.8.1	"entrypoint.sh docke..."	3 hours ago	Up 3 hours	0.0.0.0:8088->8080/tcp, :::8088->8080/tcp	05-creating-and-scheduling-data-pipelines-adminer-1
2e74f59056c5	postgres:13	"docker-entrypoint.s..."	3 hours ago	Up 3 hours	5432/tcp	05-creating-and-scheduling-data-pipelines-warehouse-1
b7b0d88547cd	postgres:13	"docker-entrypoint.s..."	3 hours ago	Up 3 hours (healthy)	5432/tcp	05-creating-and-scheduling-data-pipelines-postgres-1

gitpod /workspace/swu-ds525/05-creating-and-scheduling-data-pipelines (main) \$ ]

gitpod /workspace/swu-ds525/05-creating-and-scheduling-data-pipelines (main) \$ docker exec -it 05-creating-and-scheduling-data-pipelines-airflow-scheduler-1 bash

```
airflow.exceptions.BackfillUnfinished: BackfillJob is deadlocked.
```

```
These tasks have succeeded:
```

DAG ID	Task ID	Run ID	Try number
test_xcom	pull	backfill_2022-11-15T00:00:00+00:00	1
test_xcom	push	backfill_2022-11-15T00:00:00+00:00	1
test_xcom	pull	backfill_2022-11-16T00:00:00+00:00	1
test_xcom	push	backfill_2022-11-16T00:00:00+00:00	1
test_xcom	pull	backfill_2022-11-17T00:00:00+00:00	1
test_xcom	push	backfill_2022-11-17T00:00:00+00:00	1
test_xcom	pull	backfill_2022-11-18T00:00:00+00:00	1
test_xcom	push	backfill_2022-11-18T00:00:00+00:00	1
test_xcom	pull	backfill_2022-11-19T00:00:00+00:00	1
test_xcom	push	backfill_2022-11-19T00:00:00+00:00	1
test_xcom	pull	backfill_2022-11-21T00:00:00+00:00	1
test_xcom	push	backfill_2022-11-21T00:00:00+00:00	1
test_xcom	pull	scheduled_2022-11-20T00:00:00+00:00	2
test_xcom	push	scheduled_2022-11-20T00:00:00+00:00	2

```
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

```
These tasks are deadlocked:
```

DAG ID	Task ID	Run ID	Try number
test_xcom	pull	backfill_2022-11-22T00:00:00+00:00	1
test_xcom	push	backfill_2022-11-22T00:00:00+00:00	1

```
default@26bbdaeb7c3a:/opt/airflow$ ]
```



DAGs

Datasets

Security

Browse

Admin

Docs

17:58 UTC

AA

## DAG: test\_xcom

Schedule: @daily

Next Run: 2022-11-23, 00:00:00

Grid

Graph

Calendar

Task Duration

Task Tries

Landing Times

Gantt

Details

&lt;&gt; Code

Audit Log

11/21/2022 05:53:43 PM

25

All Run Types

All Run States

Clear Filters

deferred failed queued running scheduled skipped success up\_for\_reschedule up\_for\_retry upstream\_failed no\_status

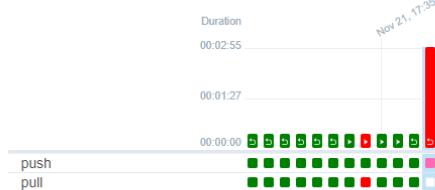
Auto-refresh



DAG

test\_xcom / Run

2022-11-23, 00:00:00 UTC



DAG Run Details

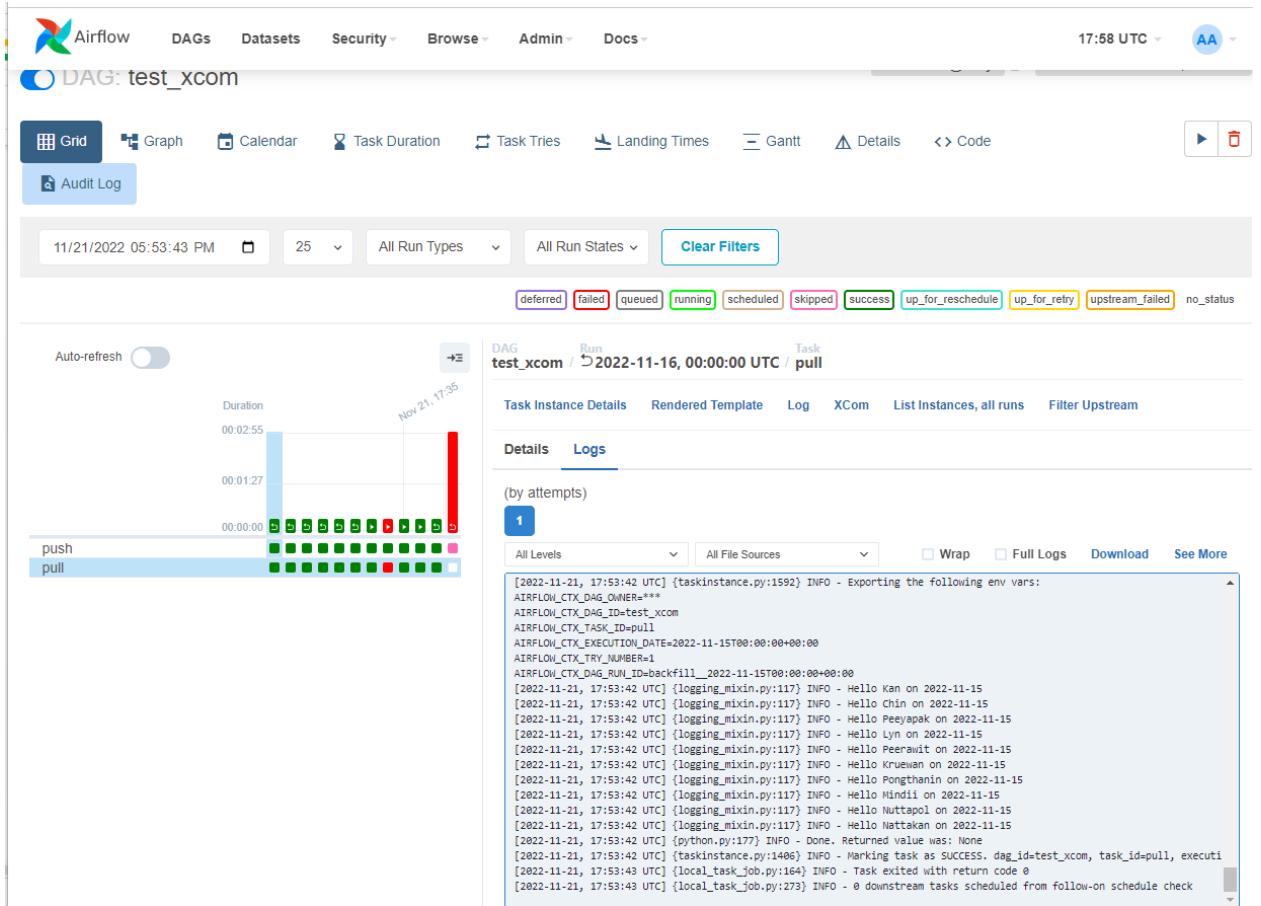
Graph

Mark Failed

Mark Success

Re-run: Clear existing tasks Queue up new tasks

Status	<span style="color:red;">■</span> failed
Run ID	backfill__2022-11-22T00:00:00+00:00
Run type	↳ backfill
Run duration	00:02:55
Last scheduling decision	2022-11-21, 17:53:51 UTC
Started	2022-11-21, 17:53:26 UTC
Ended	2022-11-21, 17:56:22 UTC
Data interval start	2022-11-22, 00:00:00 UTC
Data interval end	2022-11-23, 00:00:00 UTC



gitpod /workspace/swu-ds525 (main) \$ cd  
05-creating-and-scheduling-data-pipelines/

gitpod /workspace/swu-ds525/05-creating-and-scheduling-data-pipelines (main) \$  
docker ps

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
PORTS		NAMES		
a85f837c40cc	apache/airflow:2.4.1	"/usr/bin/dumb-init ..."	3 hours ago	Up 3 hours (healthy)
	0.0.0.0:8080->8080/tcp, :::8080->8080/tcp			
	5-creating-and-scheduling-data-pipelines-airflow-webserver-1			
109ae6d1e50c	apache/airflow:2.4.1	"/usr/bin/dumb-init ..."	3 hours ago	Up 3 hours (healthy)
	8080/tcp			
	5-creating-and-scheduling-data-pipelines-airflow-triggerer-1			
26bbda0b7c3a	apache/airflow:2.4.1	"/usr/bin/dumb-init ..."	3 hours ago	Up 3 hours (healthy)
	8080/tcp			
	5-creating-and-scheduling-data-pipelines-airflow-scheduler-1			

```
65acd4300294 adminer:4.8.1      "entrypoint.sh docke..." 3 hours ago Up 3
hours      0.0.0.0:8088->8080/tcp, :::8088->8080/tcp
05-creating-and-scheduling-data-pipelines-adminer-1
```

```
2e74f59056c5 postgres:13      "docker-entrypoint.s..." 3 hours ago Up 3
hours      5432/tcp
05-creating-and-scheduling-data-pipelines-warehouse-1
```

```
b7b0d86547cd postgres:13      "docker-entrypoint.s..." 3 hours ago Up 3
hours (healthy) 5432/tcp
05-creating-and-scheduling-data-pipelines-postgres-1
```

```
gitpod /workspace/swu-ds525/05-creating-and-scheduling-data-pipelines (main) $ docker exec 05-creating-and-scheduling-data-pipelines-a
```

```
05-creating-and-scheduling-data-pipelines-adminer-1
05-creating-and-scheduling-data-pipelines-airflow-triggerer-1
```

```
05-creating-and-scheduling-data-pipelines-airflow-scheduler-1
05-creating-and-scheduling-data-pipelines-airflow-webserver-1
```

```
gitpod /workspace/swu-ds525/05-creating-and-scheduling-data-pipelines (main) $ docker exec 05-creating-and-scheduling-data-pipelines-airflow-scheduler-1 bash
```

```
default@26bbda0b7c3a:/opt/airflow$ airflow dags
```

```
default@26bbda0b7c3a:/opt/airflow$ airflow dags list
```

```
default@26bbda0b7c3a:/opt/airflow$ airflow dags backfill -s 2022-11-15 -e 2022-11-22 test_xcom
```

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
-----	-----	-----	-----

These tasks are deadlocked:

DAG ID	Task ID	Run ID	Try number
-----	-----	-----	-----

test_xcom	pull	backfill__2022-11-22T00:00:00+00:00	1
-----------	------	-------------------------------------	---

test_xcom	push	backfill__2022-11-22T00:00:00+00:00	1
-----------	------	-------------------------------------	---

---

ช่วงทำโปรเจค