

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

สิ่งที่คาดหวังในโปรเจกต์นี้

1. นำโค้ดจาก [Project: Building a Data Modeling with Postgres \(SQL\)](#) มาสร้าง data pipeline โดยใช้ Airflow
2. มีการเขียน documentation อธิบายสิ่งที่ตัวเองทำลงไป รวมไปถึงการออกแบบ data model
3. มี instruction ในการรันโค้ดของตัวเอง

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

วิธีทำ Over All

- ① import DAG, time-zone
- ② สร้าง Data pipeline ชื่อ "MyDag"

③ save file python

④ กด ก้านั้นที่ Airflow รอสัปดาห์จะรันที่ Airflow

$t_1 \gg t_2$

⑤ bash operator, python operator

⑥ grid view

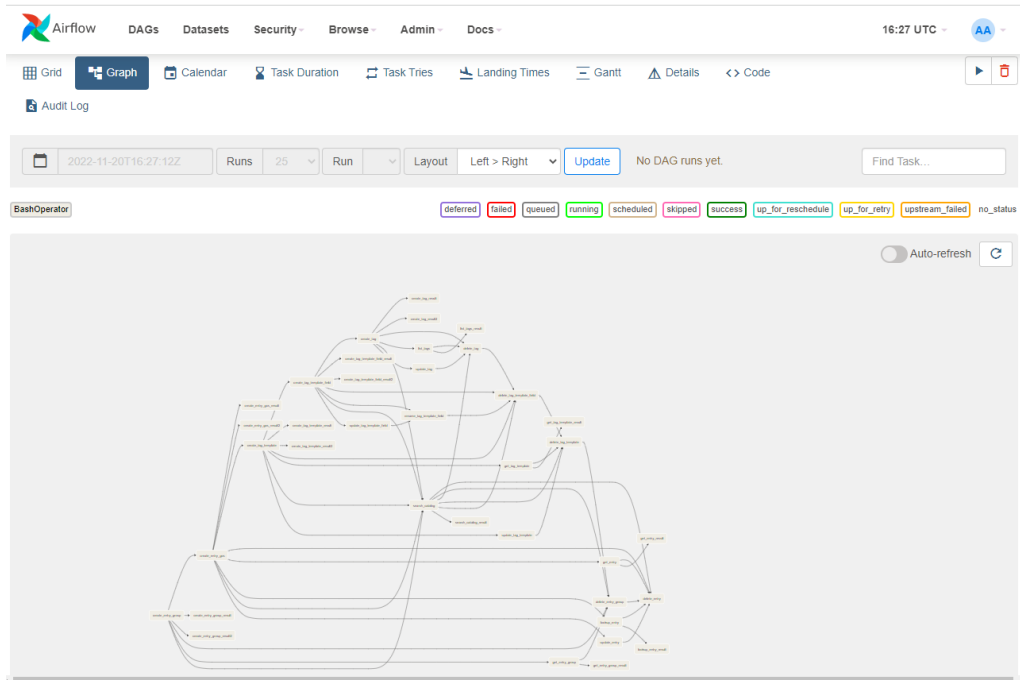
ค้นหา เวลาแล้ว
trigger 1 ที่ได้ 1 day run

clear task

$t_1 \rightarrow details \rightarrow clear \rightarrow confirm$
จะ run ใหม่อีกรอบ

cronitor \rightarrow expression เพื่อ set ใน run ตามวัน - เวลา

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 ยังไม่schedule
13 # step10
14 # schedule
15 # schedule เที่ยงคืน คือ 0
16 #v1 schedule = None ยังไม่set 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")

```


[DAGs](#)
[Datasets](#)
[Security](#)
[Browse](#)
[Admin](#)
[Docs](#)
16:55 UTC

All 4
Active 3
Paused 44

Auto-refresh

DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks	Actions
<input checked="" type="checkbox"/> my_dag workshop	airflow	<div> <div>22</div> <div>3</div> </div>	0 0 * * * *	2022-11-20, 16:24:00	2022-11-20, 00:00:00	<div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> </div>	<div>▶</div> <div>🗑</div>
<input checked="" type="checkbox"/> my_dag2 workshop	airflow	<div> <div>1</div> <div></div> </div>	*:30 * * * *	2022-11-20, 16:00:00	2022-11-20, 16:30:00	<div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> </div>	<div>▶</div> <div>🗑</div>
<input checked="" type="checkbox"/> my_dag_revise workshop	airflow	<div> <div></div> <div></div> <div></div> <div></div> </div>	None			<div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> </div>	<div>▶</div> <div>🗑</div>

Trigger DAG

Trigger DAG w/ config


[Airflow](#)
[DAGs](#)
[Datasets](#)
[Security](#)
[Browse](#)
[Admin](#)
[Docs](#)
16:56 UTC 

DAG: my_dag_revise

success
Schedule: None
Next Run: None

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

Audit Log

2022-11-20T16:56:00Z
Runs: 25
Run: manual__2022-11-20T16:55:59.972252+00:00
Layout: Left > Right
Find Task...

Update

EmptyOperator

deferred
failed
queued
running
scheduled
skipped
success
up_for_reschedule
up_for_retry
upstream_failed
no_status

t1

t2

☐ Auto-refresh

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 ยังไม่schedule
13 # step10
14 # schedule
15 # schedule เพียงคืน คือ 0
16 # tags = ['workshop'] จะทำให้เป็นชัดเจน
17 #v1 schedule = None ยังไม่set schedule
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
```

Airflow DAGs Datasets Security Browse Admin Docs 16:59 UTC AA

Triggered my_dag_revise, it should start any moment now.

DAG: my_dag_revise **queued** Schedule: None Next Run: None

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

EmptyOperator deferred failed queued running scheduled skipped success up_for_reschedule up_for_retry upstream_failed no_status

Auto-refresh

t1 → t2

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 รันก่อน 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 รันก่อน t2
49     t1 >> t2
```

DAG: my_dag_revise success Schedule: None Next Run: None

Grid Graph Calendar Task Duration Task Times 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

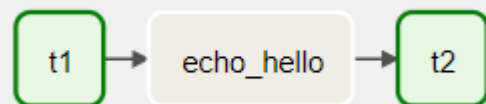
deferred failed queued running scheduled skipped success up_for_reschedule up_for_retry upstream_failed no_status

Auto-refresh

echo_hello

t1 → t2

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



DAG: my_dag_revise

Schedule: NoneNext Run: None

GridGraphCalendarTask DurationTask TriesLanding TimesGanttDetailsCodeAudit Log

11/20/2022 05:12:27 PM25All Run TypesAll Run StatesClear Filters

Auto-refresh

Duration

00:00:03

00:00:01

00:00:00

t1

echo_hello

t2

DAG

my_dag_revise

DAG Details

DAG Runs Summary

Total Runs Displayed	3
Total success	3
First Run Start	2022-11-20, 16:56:00 UTC
Last Run Start	2022-11-20, 17:12:19 UTC
Max Run Duration	00:00:03
Mean Run Duration	00:00:01
Min Run Duration	00:00:00

DAG Summary

Total Tasks	3
Empty Operators	2
BashOperator	1

Triggered my_dag_revise, it should start any moment now.

DAG: my_dag_revise

Schedule: NoneNext Run: None

GridGraphCalendarTask DurationTask TriesLanding TimesGanttDetailsCodeAudit Log

11/20/2022 05:14:10 PM25All Run TypesAll Run StatesClear Filters

Auto-refresh

Duration

00:00:03

00:00:01

00:00:00

t1

echo_hello

t2

DAG

my_dag_revise

Run

2022-11-20, 17:12:18 UTC

Task

echo_hello

Task Instance DetailsRendered TemplateLogXComList Instances, all runsFilter Upstream

DetailsLogs

Task Actions

Ignore All DepsIgnore Task StateIgnore Task DepsRun

PastFutureUpstreamDownstreamRecursiveFailedClear

PastFutureUpstreamDownstreamMark Failed

PastFutureUpstreamDownstreamMark Success

Status

success

Task ID

echo_hello

Run ID

manual__2022-11-20T17:12:18.932212+00:00

Operator

BashOperator

Duration

00:00:00

Started

2022-11-20, 17:12:21 UTC

Ended

2022-11-20, 17:12:22 UTC

Airflow DAGs Datasets Security Browse Admin Docs 17:14 UTC

Triggered my_dag_revise, it should start any moment now.

DAG: my_dag_revise Scheduler: 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

Auto-refresh

Duration Nov 20, 16:58

my_dag_revise Run 2022-11-20, 17:12:18 UTC Task echo_hello

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

Details Logs

(by attempts)

All Levels All File Sources Wrap Full Logs Download See More

```

*** Reading Local File: /opt/airflow/logs/dag_id=my_dag_revise/run_id=manual_2022-11-20T17:12:18.93222+00:00/task_id=
[2022-11-20, 17:12:21 UTC] (taskinstance.py:1165) INFO - Dependencies all met for <taskinstance: my_dag_revise,echo_hello>
[2022-11-20, 17:12:21 UTC] (taskinstance.py:1165) INFO - Dependencies all met for <taskinstance: my_dag_revise,echo_hello>
[2022-11-20, 17:12:21 UTC] (taskinstance.py:1162) INFO -
[2022-11-20, 17:12:21 UTC] (taskinstance.py:1163) INFO - Starting attempt 1 of 1
[2022-11-20, 17:12:21 UTC] (taskinstance.py:1164) INFO -
[2022-11-20, 17:12:21 UTC] (taskinstance.py:1163) INFO - Executing <task(BashOperator): echo_hello on 2022-11-20 17:12:
[2022-11-20, 17:12:21 UTC] (standard_task_runner.py:34) INFO - Started process 24279 to run task
[2022-11-20, 17:12:21 UTC] (standard_task_runner.py:82) INFO - Running: [***, 'task', 'run', 'my_dag_revise', 'echo_h
[2022-11-20, 17:12:21 UTC] (standard_task_runner.py:83) INFO - Job 687: Subtask echo_hello
[2022-11-20, 17:12:21 UTC] (dagbag.py:525) INFO - Killing up the dagbag from /opt/***/dagrev_dag_revise.py
[2022-11-20, 17:12:21 UTC] (taskinstance.py:285) WARNING - Dependency <task(BashOperator): create_entry_group, delete_entry
[2022-11-20, 17:12:21 UTC] (taskinstance.py:285) WARNING - Dependency <task(BashOperator): delete_entry_group, create_entry
[2022-11-20, 17:12:21 UTC] (taskinstance.py:285) WARNING - Dependency <task(BashOperator): create_entry_group, delete_entry
[2022-11-20, 17:12:21 UTC] (taskinstance.py:285) WARNING - Dependency <task(BashOperator): delete_entry, create_entry_group
[2022-11-20, 17:12:21 UTC] (taskinstance.py:285) WARNING - Dependency <task(BashOperator): create_tag, delete_tag already
[2022-11-20, 17:12:21 UTC] (taskinstance.py:285) WARNING - Dependency <task(BashOperator): delete_tag, create_tag already
[2022-11-20, 17:12:21 UTC] (taskinstance.py:285) WARNING - Dependency <task(BashOperator): get_ip, prepare_email air

```

```

#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 >= t2
    t1 >> echo_hello >> print_hey >> t2

```

Airflow DAGs Datasets Security Browse Admin Docs 17:24 UTC

DAG: my_dag_revise success Scheduler: None Next Run: None

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

2022-11-20T17:17:00Z Runs 25 Run manual_2022-11-20T17:16:59.347340+00:00 Layout Left ~ Right Update Find Task...

BashOperator EmptyOperator PythonOperator

Auto-refresh

11 echo_hello print_hey 12

← → ↺

crontab.guru

poll_form.php

Canny edge detecti...

Your Repositories

Simple guide on ho...

A Neural Network P...

Learner Lab

Image Feature Extra...

Canny edge detecti...

How to tune hyper...

Cron Job Monitoring

crontab guru

The quick and simple editor for cron schedule expressions by Cronitor

"At 04:05."

next at 2022-11-21 04:05:00

random

5 4 * * *

minute	hour	day (month)	month	day (week)
*			any value	
,			value list separator	
-			range of values	
/			step values	
@yearly			(non-standard)	
@annually			(non-standard)	
@monthly			(non-standard)	
@weekly			(non-standard)	
@daily			(non-standard)	
@hourly			(non-standard)	
@reboot			(non-standard)	

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.


```

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

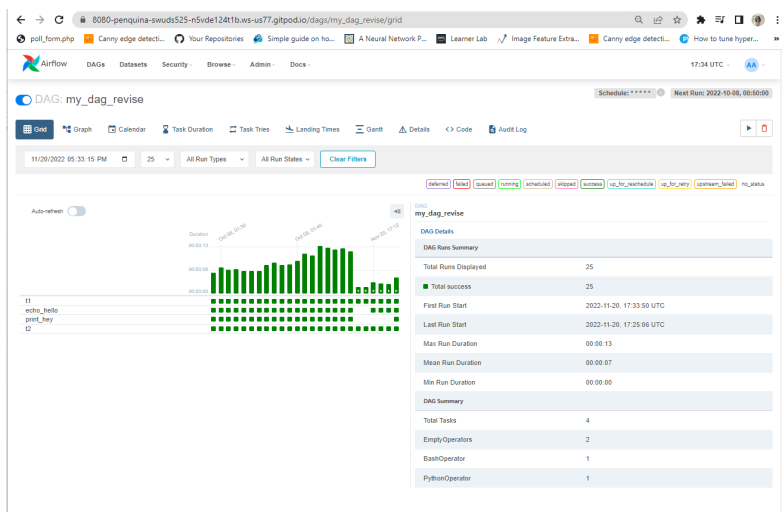
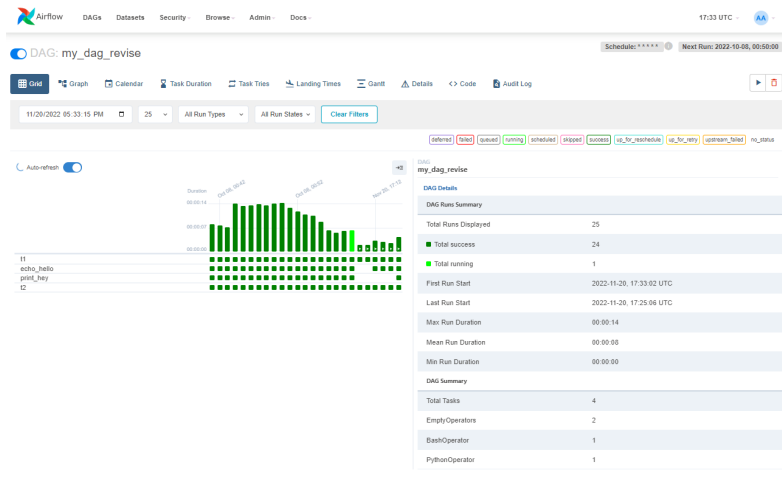
    echo_hello = BashOperator(
        task_id = "echo_hello",
        bash_command= "echo 'hello'",
    )
    def _print_hey():
        print("Hey!")

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

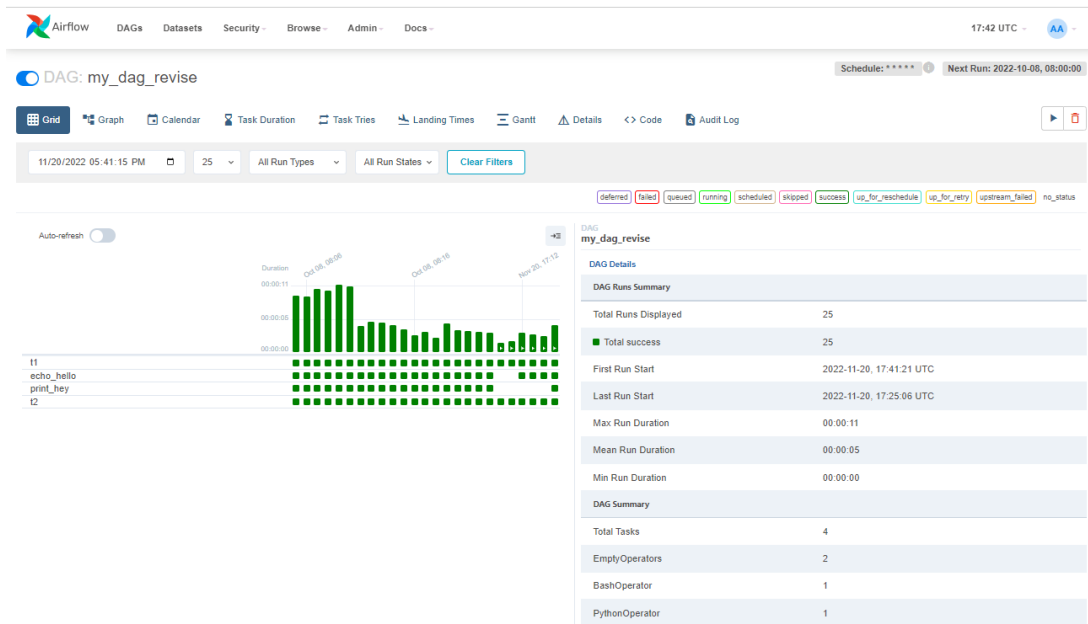
    t2 = EmptyOperator( task_id = "t2")

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

```



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



Week9 ช่วงทบทวนweek 8

Airflow Concepts

```
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
```

DAG

Operator

Task

Define the dependencies here


























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 4	Paused 1	x workshop		Search DAGs	Auto-refresh	
DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks	Actions	Links
 my_dag workshop	airflow		0 8 ***	2022-11-20, 16:24:00	2022-11-21, 00:00:00		 	...
 my_dag2 workshop	airflow		10 8 ***	2022-11-21, 14:30:00	2022-11-21, 15:00:00		 	...
 my_dag_revise workshop	airflow		10 8 ***	2022-11-20, 17:25:06	2022-11-21, 08:30:00		 	...
 my_dag_w9 workshop	airflow		10 8 ***	2022-11-21, 10:30:00	2022-11-21, 08:00:00		 	...
 test_xcom workshop	airflow		@daily		2022-11-20, 00:00:00		 	...

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

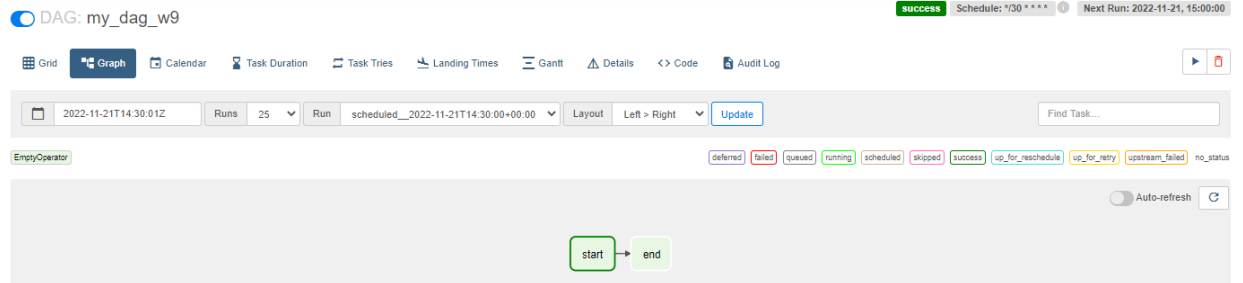
```

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

```

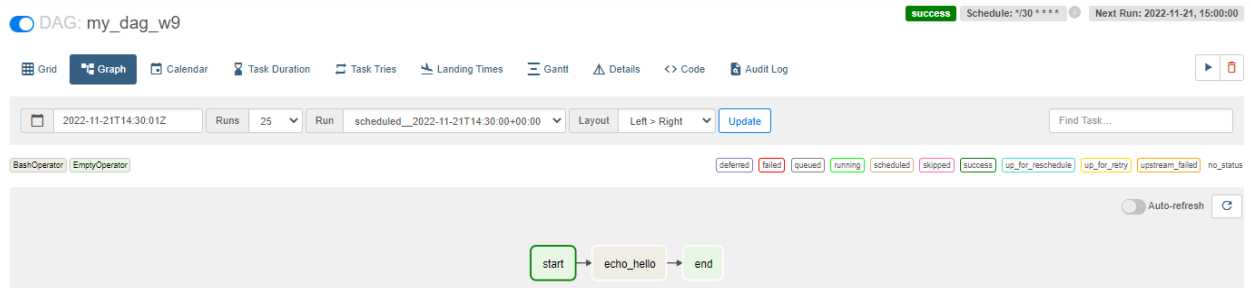


05-creating-and-scheduling-data-pipelines > dags > my_dag_w9.py

```



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  #ชื่อฟังก์ชันต้องไม่เหมือนชื่อtask เลยใส่ _ นำหน้า
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 |
32 |      end = EmptyOperator(task_id = "end")
33
34 |      start >> echo_hello >> say_hello >> end
35
```



 Airflow DAGs Datasets Security Browse Admin Docs 15:13 UTC 


 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 



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  #ชื่อฟังก์ชันต้องไม่เหมือนชื่อtask เลยใส่ _ หน้าหน้า
10
11  def _say_hello():
12      print("Hello")
13
14  #logging มีระดับการlog => info, debug แล้วแต่การใช้งาน
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
38      print_log_messages = PythonOperator(
39          task_id = "print_log_messages",
40          python_callable = _print_log_messages,
41      )
42
43
44      end = EmptyOperator(task_id = "end")
45
46      start >> echo_hello >> say_hello >> print_log_messages >> end
47
```


Airflow DAGs Datasets Security Browse Admin Docs 15:19 UTC AA

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

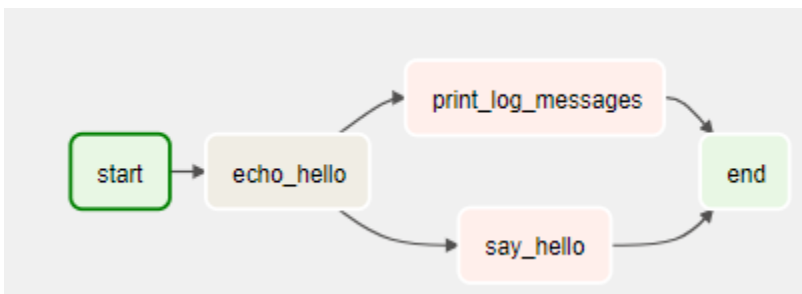
deferred failed queued running scheduled skipped success up_for_reschedule up_for_retry upstream_failed no_status

Auto-refresh

```

graph LR
    start([start]) --> echo_hello[echo_hello]
    echo_hello --> say_hello[say_hello]
    say_hello --> print_log_messages[print_log_messages]
    print_log_messages --> end([end])
  
```

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



Auto-refresh

DAG my_dag_w9 Run Task 2022-11-21, 15:00:00 UTC say_hello

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

```

graph LR
    start([start]) --> echo_hello[echo_hello]
    echo_hello --> say_hello[say_hello]
    say_hello --> print_log_messages[print_log_messages]
    print_log_messages --> end([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]
```

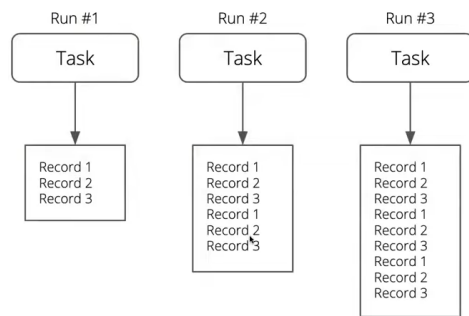
```
new_fruits = add_new("Pineapple")
print (" Idempotent ")
print(new_fruits)
```

```
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']
```

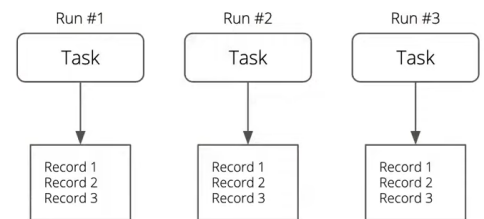
Non-idempotent vs. Idempotent

Non-idempotent

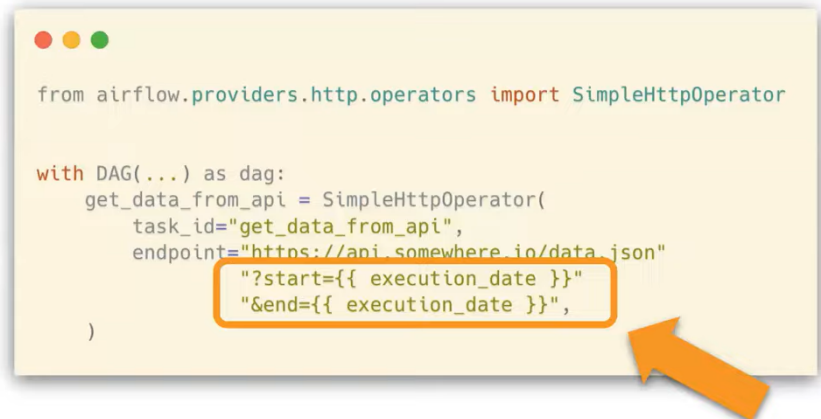


vs.

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