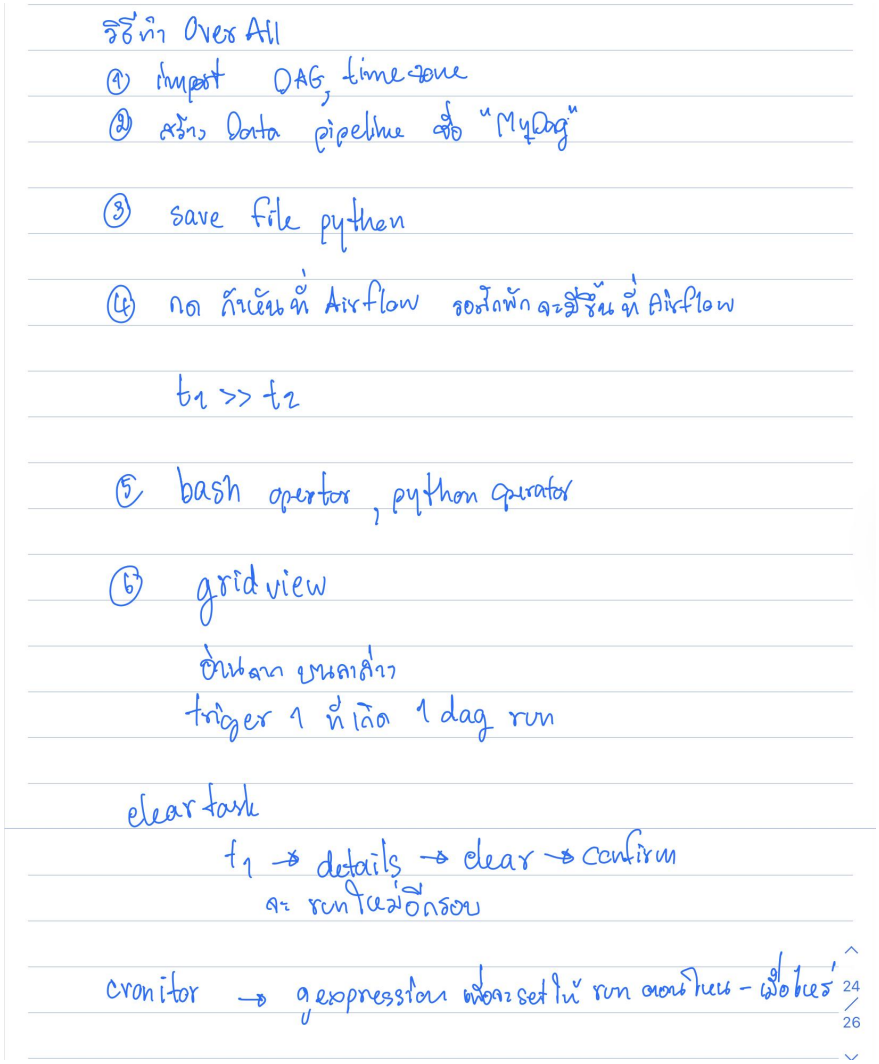
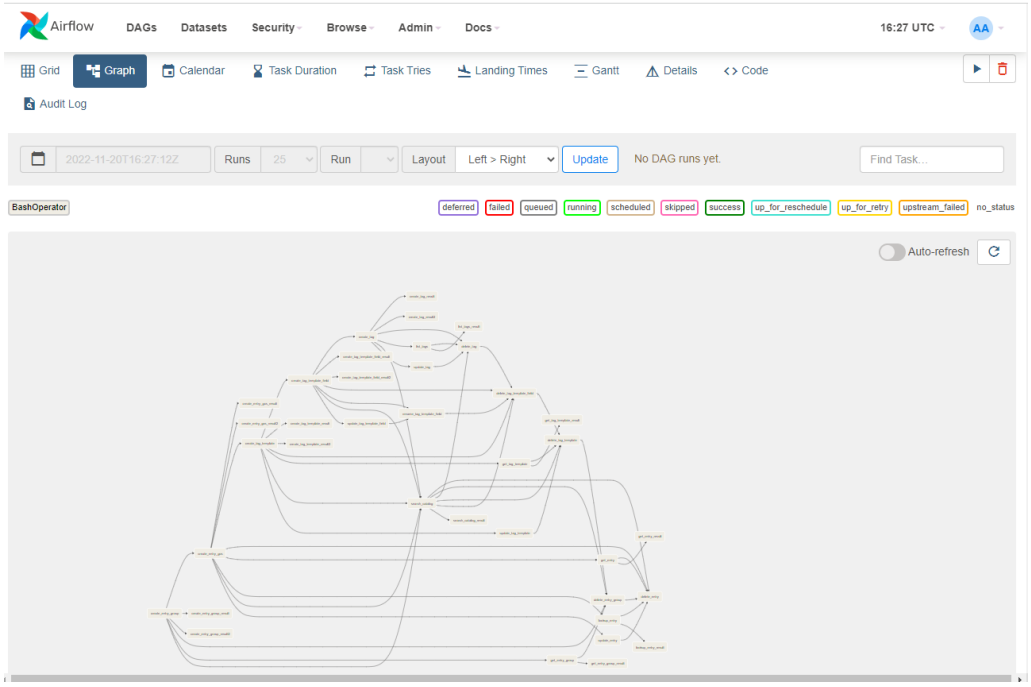
Project: Creating and Automating a Set of Data Pipelines with Airflow

สิ่งที่คาดหวังในโปรเจคนี้

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

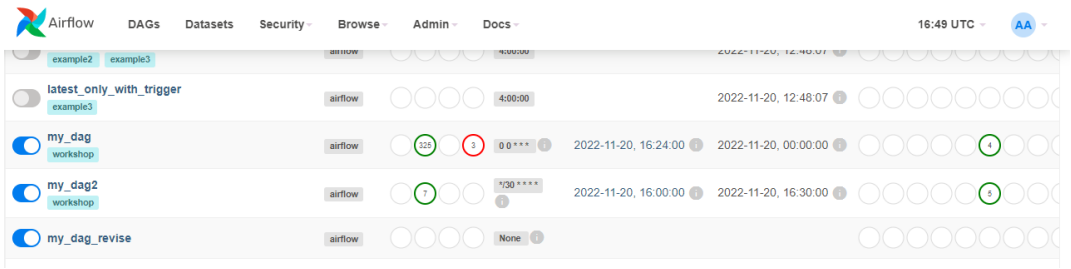
gitpod /workspace/swu-ds525/05-creating-and-scheduling-data-pipelines (main) $ docker-compose up

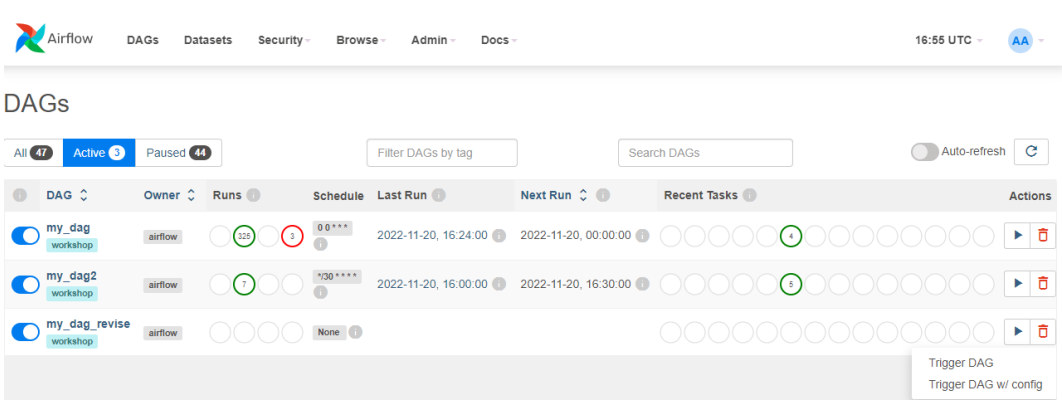


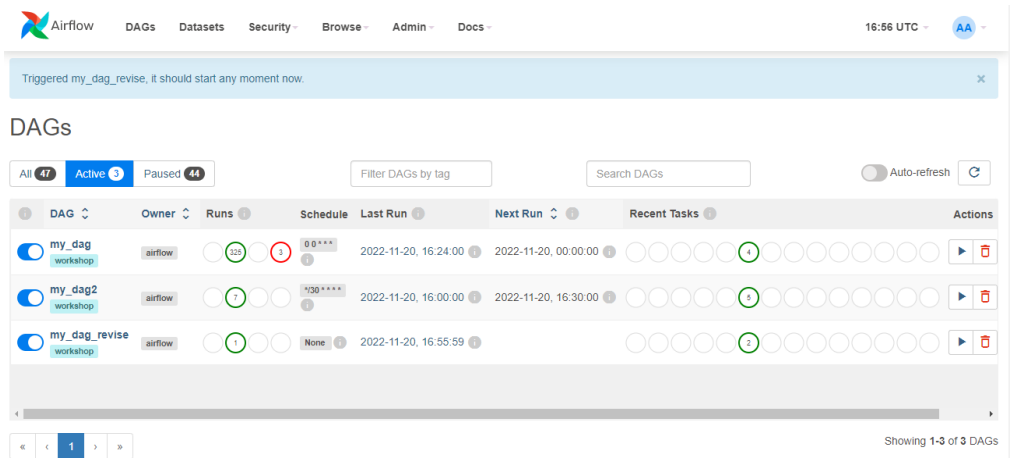


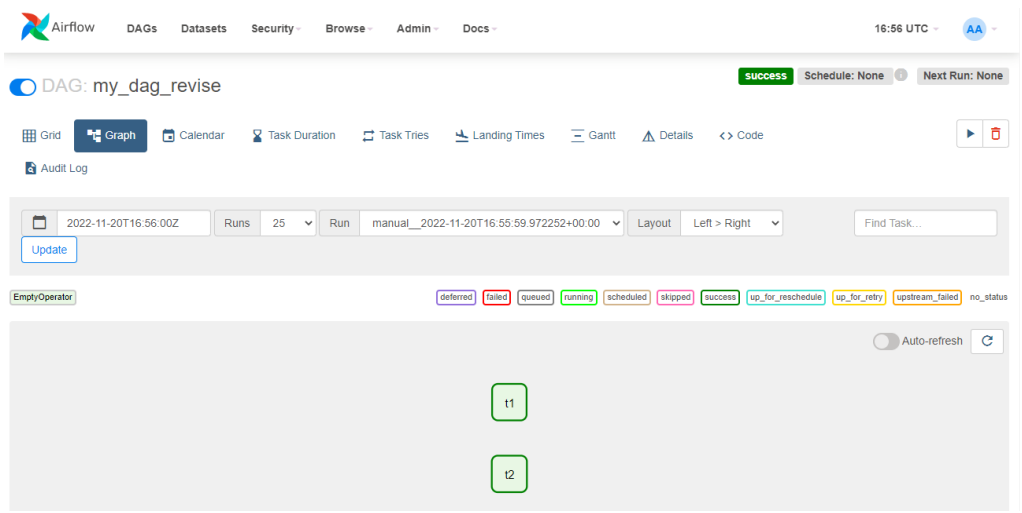
05-creating-and-scheduling-data-pipelines > dags > my_dag_revise.py

```python
#import DAG เป็น pipeline
from airflow import DAG
#import Timezone
from airflow.utils import timezone
#step 2 ใช้ EmptyOperator เลย import EmptyOperator เข้ามา
from airflow.operators.empty import EmptyOperator


#context manager เป็นการประกาศหัว
#"my_dag" ชื่อเดียวกับชื่อ file
#start date 2022, 10, 8
# schedule = None ยังไม่schedule
# step10
# schedule
# schedule เที่ยงคืน คือ 0
#v1  schedule = None ยังไม่set schedule
with DAG(
    "my_dag",
    start_date = timezone.datetime(2022, 10, 8),
    schedule = None,
):
    t1 = EmptyOperator( task_id = "t1")
    t2 = EmptyOperator( task_id = "t2")
```


Airflow
DAGs
Datasets
Security
Browse
Admin
Docs
16:49 UTC
AA
example2
example3
latest_only_with_trigger
example3
airflow
4:00:00
2022-11-20, 12:48:07
my_dag
workshop
airflow
0 0 * * *
2022-11-20, 16:24:00
2022-11-20, 00:00:00
my_dag2
workshop
airflow
*/30 * * * *
2022-11-20, 16:00:00
2022-11-20, 16:30:00
my_dag_revise
airflow
None

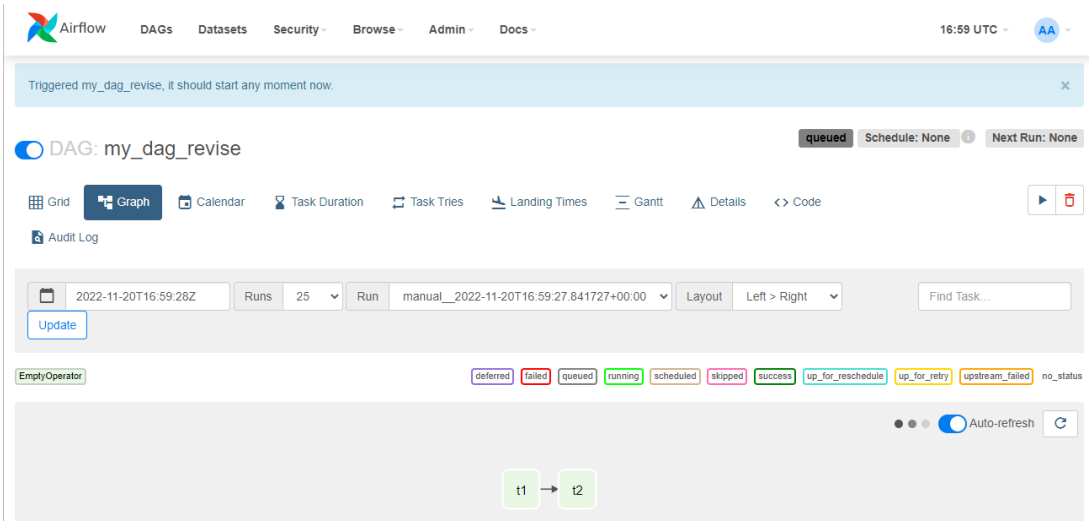

Airflow
DAGs
Datasets
Security
Browse
Admin
Docs
16:55 UTC
AA
DAGs
All 47
Active 3
Paused 44
Filter DAGs by tag
Search DAGs
Auto-refresh
DAG
Owner
Runs
Schedule
Last Run
Next Run
Recent Tasks
Actions
my_dag
workshop
airflow
0 0 * * *
2022-11-20, 16:24:00
2022-11-20, 00:00:00
my_dag2
workshop
airflow
*/30 * * * *
2022-11-20, 16:00:00
2022-11-20, 16:30:00
my_dag_revise
workshop
airflow
None
Trigger DAG
Trigger DAG w/ config


Airflow
DAGs
Datasets
Security
Browse
Admin
Docs
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
DAG
Owner
Runs
Schedule
Last Run
Next Run
Recent Tasks
Actions
my_dag
workshop
airflow
0 0 * * *
2022-11-20, 16:24:00
2022-11-20, 00:00:00
my_dag2
workshop
airflow
*/30 * * * *
2022-11-20, 16:00:00
2022-11-20, 16:30:00
my_dag_revise
workshop
airflow
None
2022-11-20, 16:55:59
Showing 1-3 of 3 DAGs


Airflow
DAGs
Datasets
Security
Browse
Admin
Docs
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
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
Auto-refresh
t1
t2

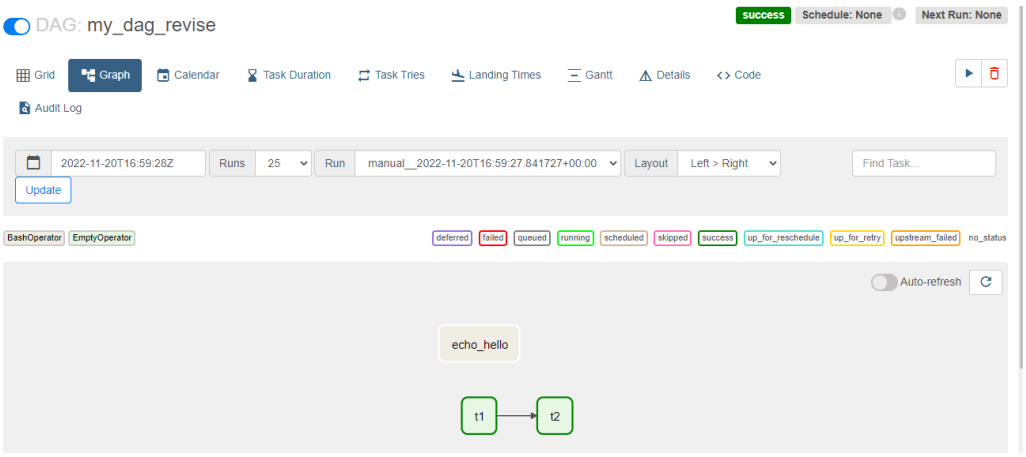
05-creating-and-scheduling-data-pipelines > dags > my_dag_revise.py

```python
#import DAG เป็น pipeline
from airflow import DAG
#import Timezone
from airflow.utils import timezone
#step 2 ใช้ EmptyOperator เลย import EmptyOperator เข้ามา
from airflow.operators.empty import EmptyOperator


#context manager เป็นการประกาศหัว
#"my_dag" ชื่อเดียวกับชื่อ file
#start date 2022, 10, 8
# schedule = None ยังไม่schedule
# step10
# schedule
# schedule เที่ยงคืน คือ 0
# tags =['workshop'] จะทำให้เป็นชัดเจน
#v1  schedule = None ยังไม่set schedule
with DAG(
    "my_dag_revise",
    start_date = timezone.datetime(2022, 10, 8),
    schedule = None,
    tags =["workshop"],
):
    t1 = EmptyOperator( task_id = "t1")
    t2 = EmptyOperator( task_id = "t2")

#t1 รันก่อน t2
    t1 >> t2

```

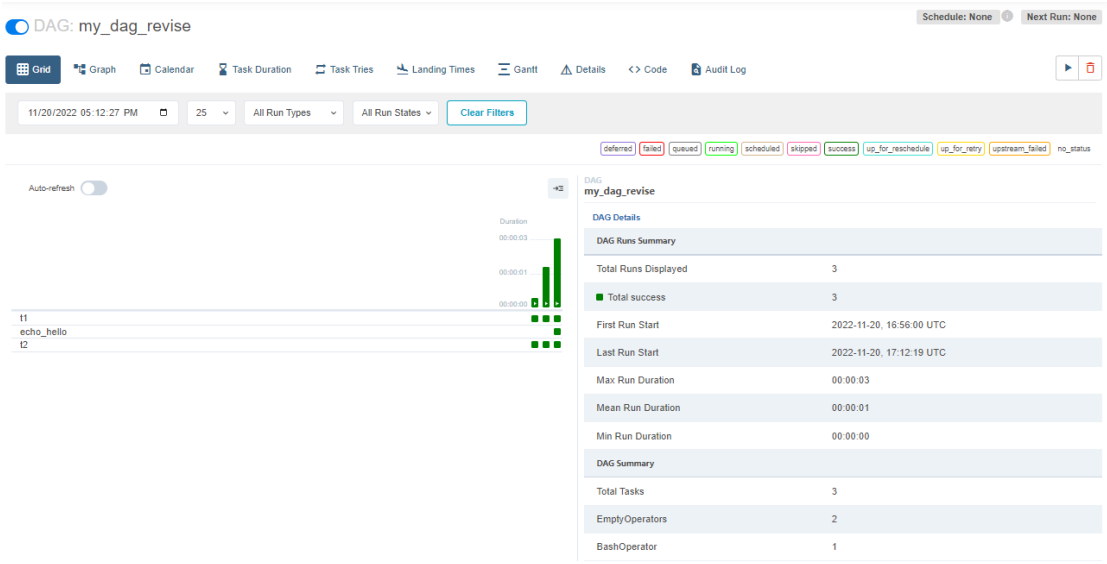


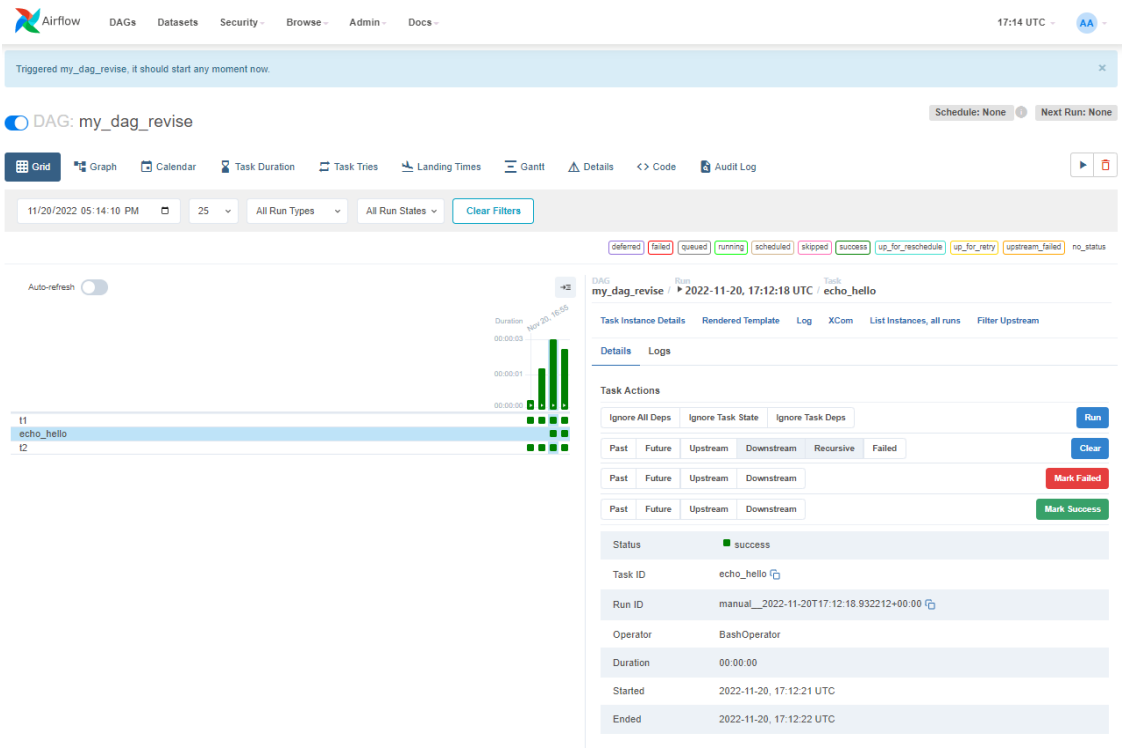
05-creating-and-scheduling-data-pipelines > dags > my_dag_revise.py

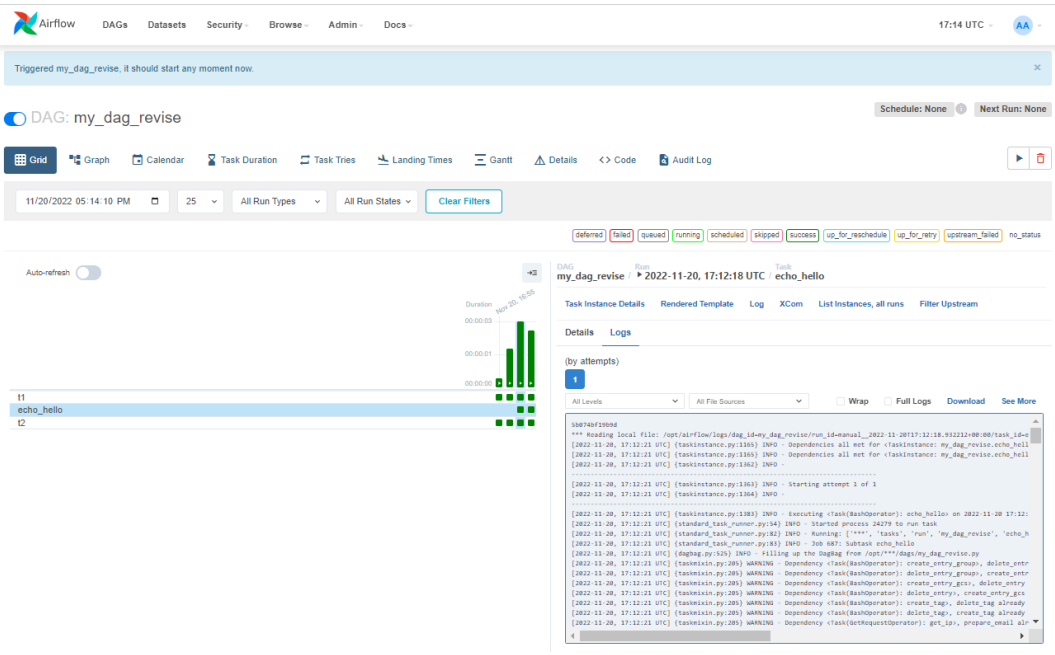
```python
# ):
#     t1 = EmptyOperator( task_id = "t1")
#     t2 = EmptyOperator( task_id = "t2")

# #t1 รันก่อน t2
#     t1 >> t2

#v2
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'",
    )

    t2 = EmptyOperator( task_id = "t2")

#t1 รันก่อน t2
    t1 >> t2
```



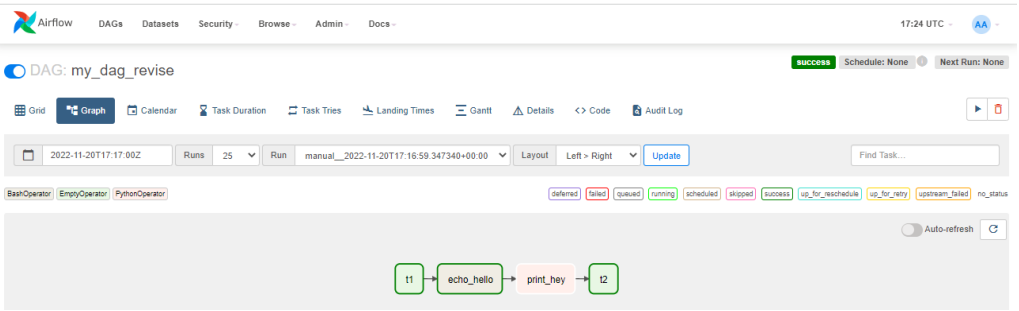

```python
#t1 รันก่อน t2
    t1 >> echo_hello >> t2
```

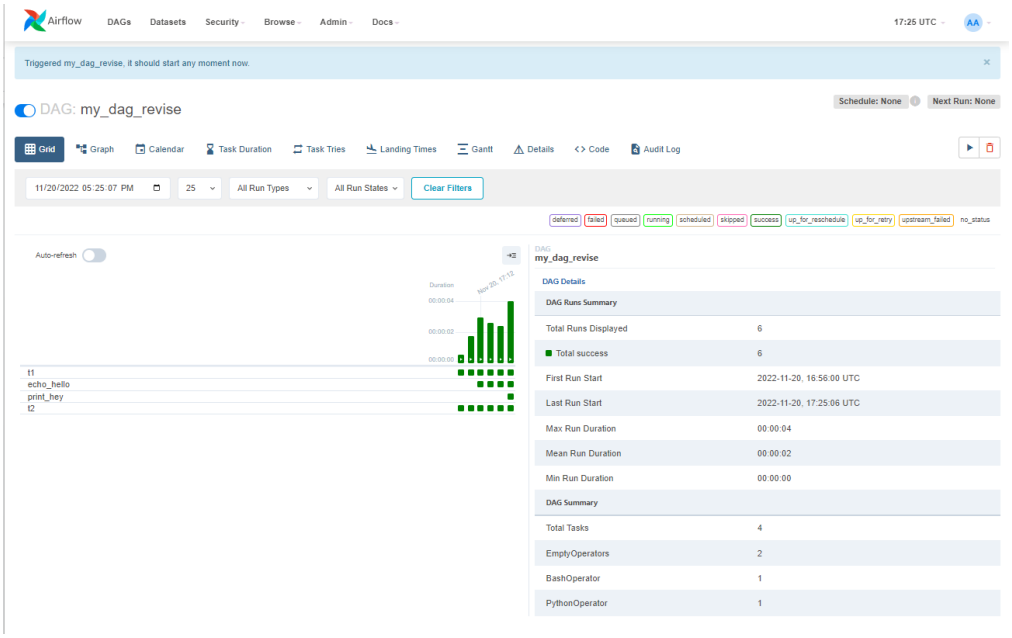

DAG: my_dag_revise
Schedule: None
Next Run: None
Grid
Graph
Calendar
Task Duration
Task Tries
Landing Times
Gantt
Details
Code
Audit Log
11/20/2022 05:12:27 PM
25
All Run Types
All Run States
Clear Filters
deferred
failed
queued
running
scheduled
skipped
success
up_for_reschedule
up_for_retry
upstream_failed
no_status
Auto-refresh
Duration
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
EmptyOperators 2
BashOperator 1

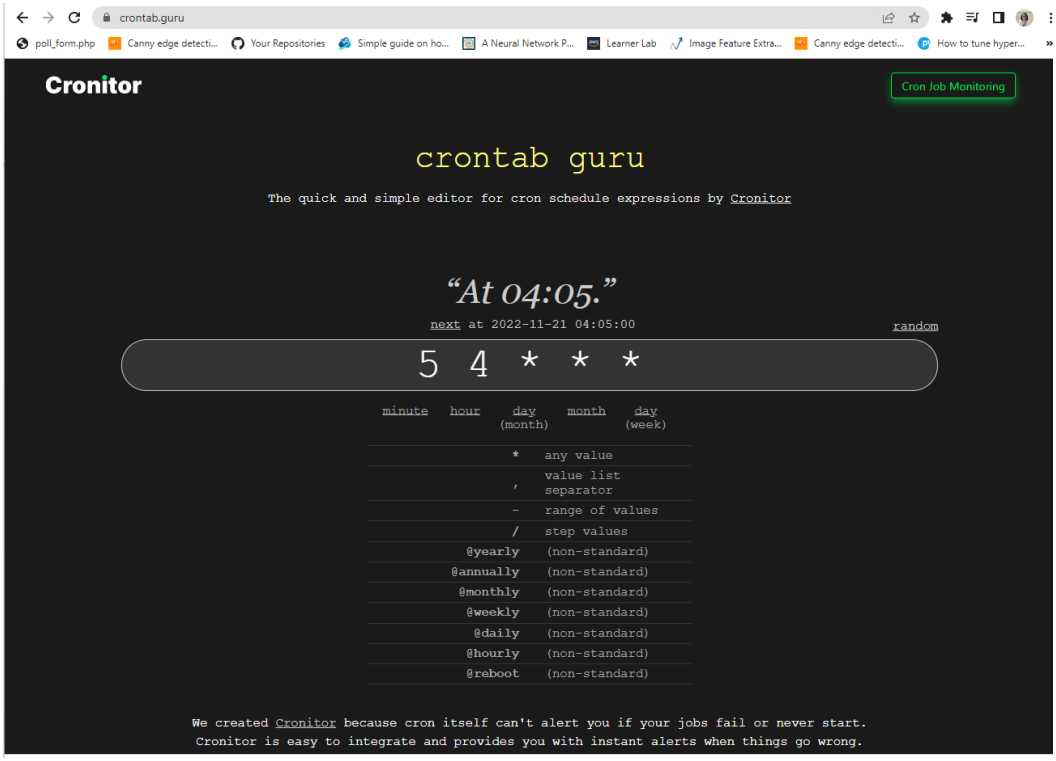

Airflow
DAGs
Datasets
Security
Browse
Admin
Docs
17:14 UTC
AA
Triggered my_dag_revise, it should start any moment now.
DAG: my_dag_revise
Schedule: None
Next Run: None
Grid
Graph
Calendar
Task Duration
Task Tries
Landing Times
Gantt
Details
Code
Audit Log
11/20/2022 05:14:10 PM
25
All Run Types
All Run States
Clear Filters
deferred
failed
queued
running
scheduled
skipped
success
up_for_reschedule
up_for_retry
upstream_failed
no_status
Auto-refresh
Duration
Nov 20, 16:55
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 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
Mark Failed
Mark 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

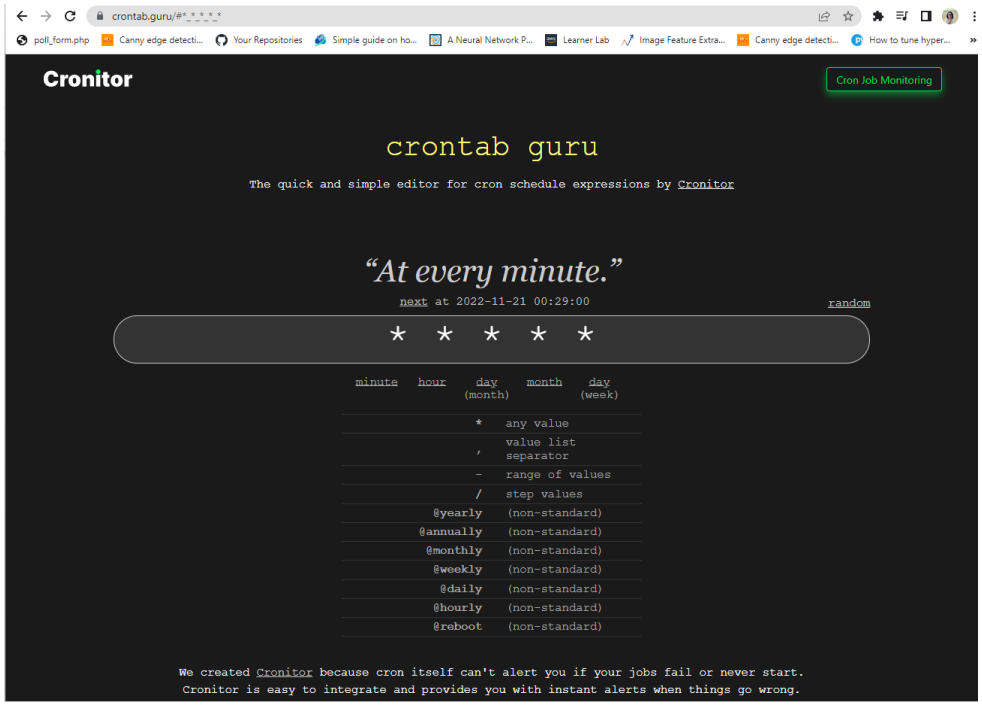


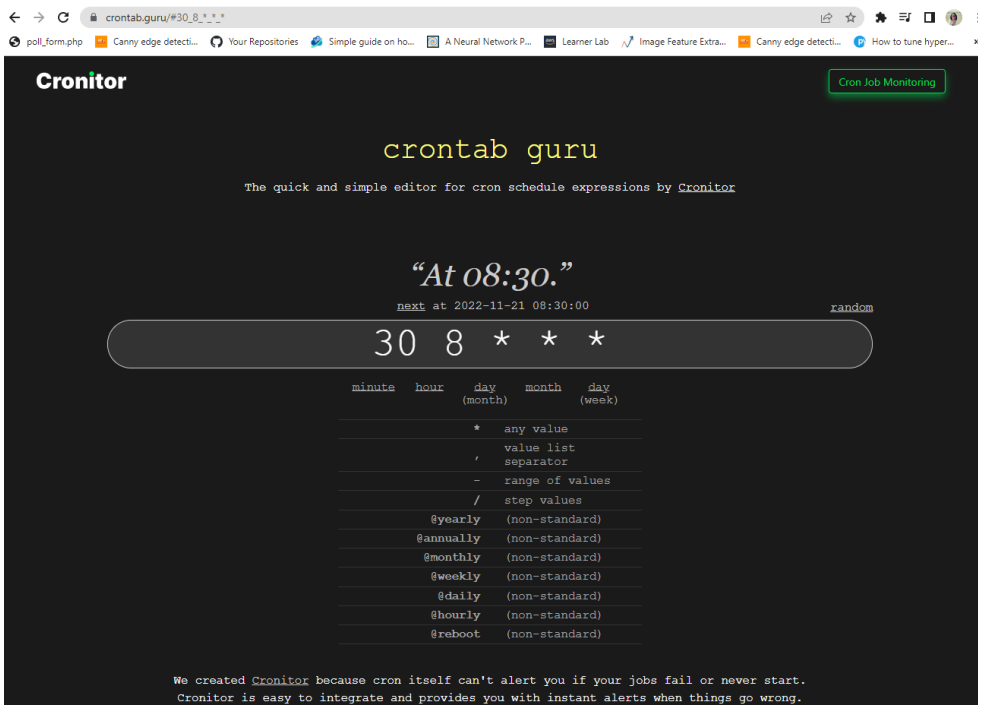

```python
#v3
with DAG(
    "my_dag_revise",
    start_date = timezone.datetime(2022, 10, 8),
    schedule = None,
    tags =["workshop"],
):
    t1 = EmptyOperator( task_id = "t1")

    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
```

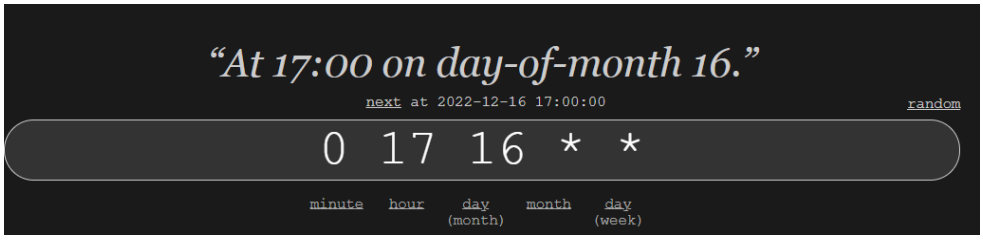



Airflow
DAGs
Datasets
Security
Browse
Admin
Docs
17:25 UTC
Triggered my_dag_revise, it should start any moment now.
DAG: my_dag_revise
Schedule: None
Next Run: None
Grid
Graph
Calendar
Task Duration
Task Tries
Landing Times
Gantt
Details
Code
Audit Log
11/20/2022 05:25:07 PM
25
All Run Types
All Run States
Clear Filters
deferred
failed
queued
running
scheduled
skipped
success
up_for_reschedule
up_for_retry
upstream_failed
no_status
Auto-refresh
Duration
00:00:04
00:00:02
00:00:00
t1
echo_hello
print_hey
t2
DAG
my_dag_revise
DAG Details
DAG Runs Summary
Total Runs Displayed 6
Total success 6
First Run Start 2022-11-20, 16:56:00 UTC
Last Run Start 2022-11-20, 17:25:06 UTC
Max Run Duration 00:00:04
Mean Run Duration 00:00:02
Min Run Duration 00:00:00
DAG Summary
Total Tasks 4
EmptyOperators 2
BashOperator 1
PythonOperator 1

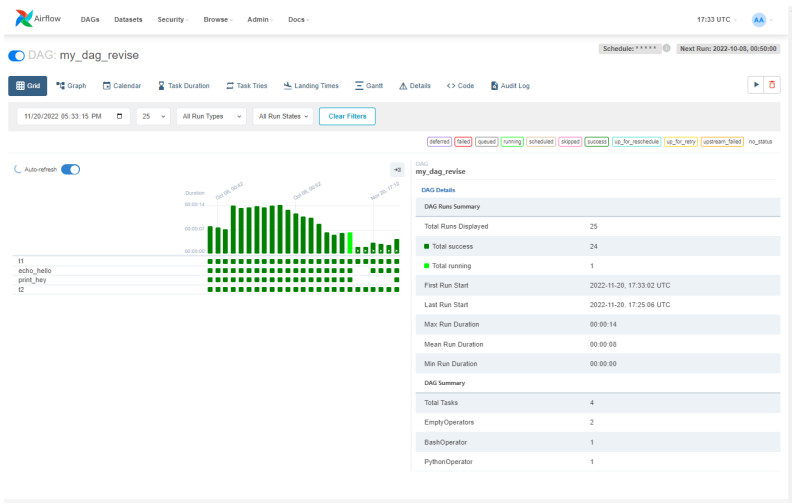

crontab.guru
poll_form.php
Canny edge detecti...
Your Repositories
Simple guide on ho...
A Neural Network P...
Learner Lab
Image Feature Extra...
Canny edge detecti...
How to tune hyper...
Cronitor
Cron Job Monitoring
crontab guru
The quick and simple editor for cron schedule expressions by Cronitor
“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.

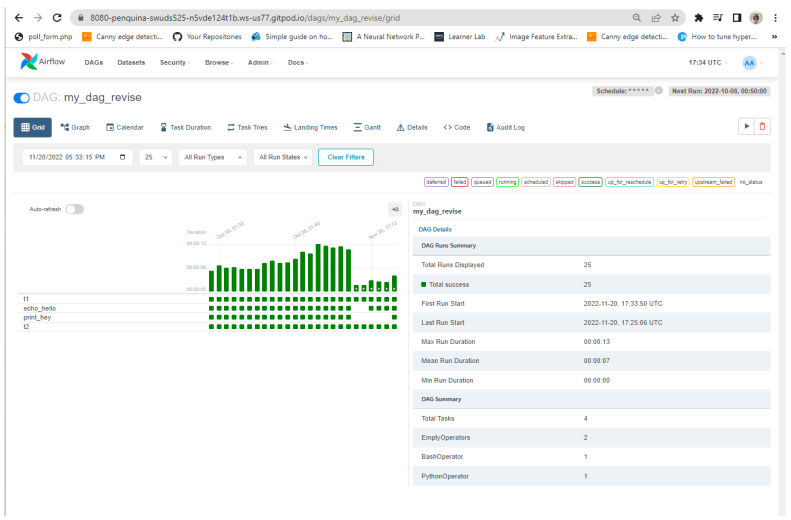

crontab.guru/#*_*_*_*_*
poll_form.php
Canny edge detecti...
Your Repositories
Simple guide on ho...
A Neural Network P...
Learner Lab
Image Feature Extra...
Canny edge detecti...
How to tune hyper...
Cronitor
Cron Job Monitoring
crontab guru
The quick and simple editor for cron schedule expressions by Cronitor
“At every minute.”
next at 2022-11-21 00:29:00
random
* * * * *
minute hour day (month) 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.


crontab.guru/#30_8_*_*_*
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...
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) 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.

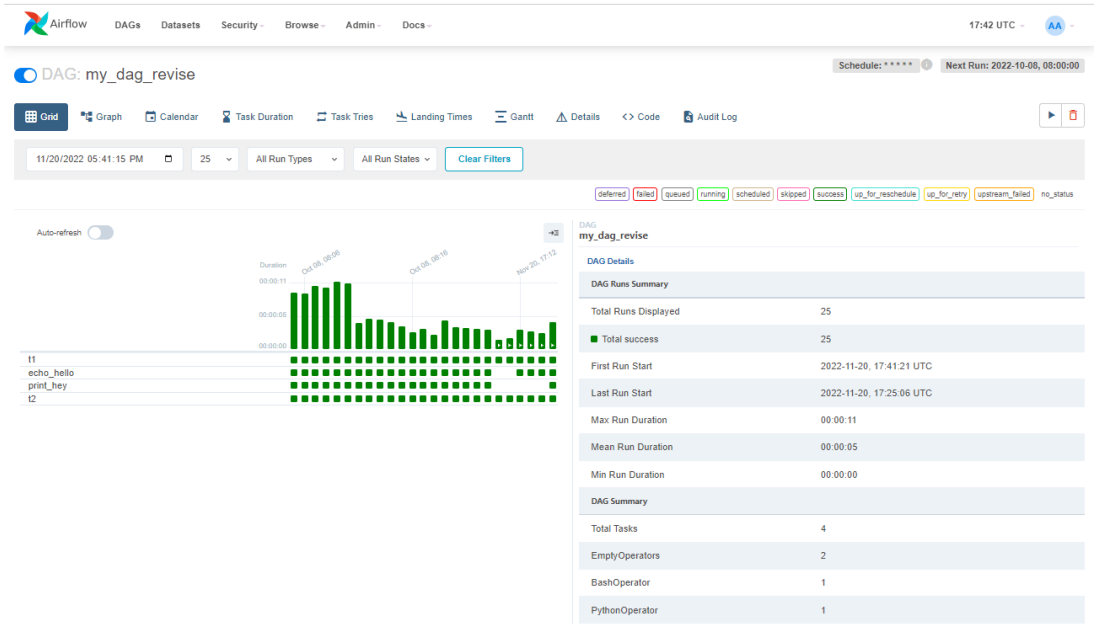

“At 17:00 on day-of-month 16.”
next at 2022-12-16 17:00:00
random
0 17 16 * *
minute hour day (month) month day (week)

```python
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
```

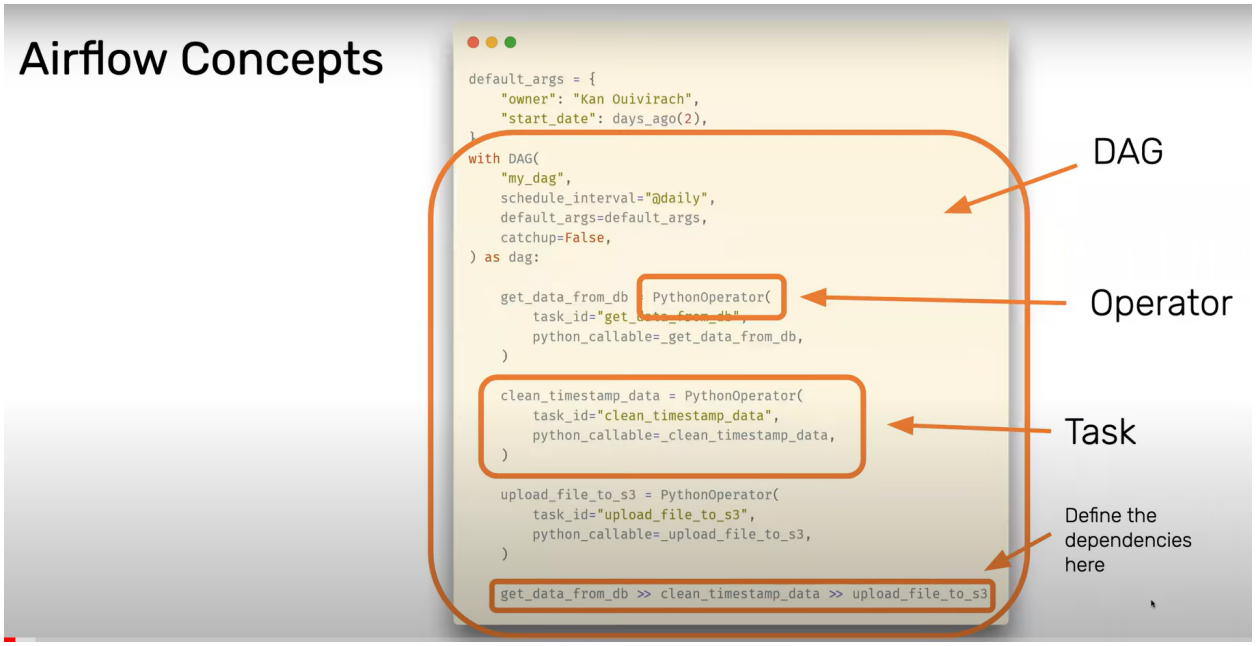




```
#t1 รันก่อน t2
    #t1 >> echo_hello >> print_hey >> t2
    t1 >> [ echo_hello, print_hey] >> t2
```



Week9 ช่วงทบทวนweek 8

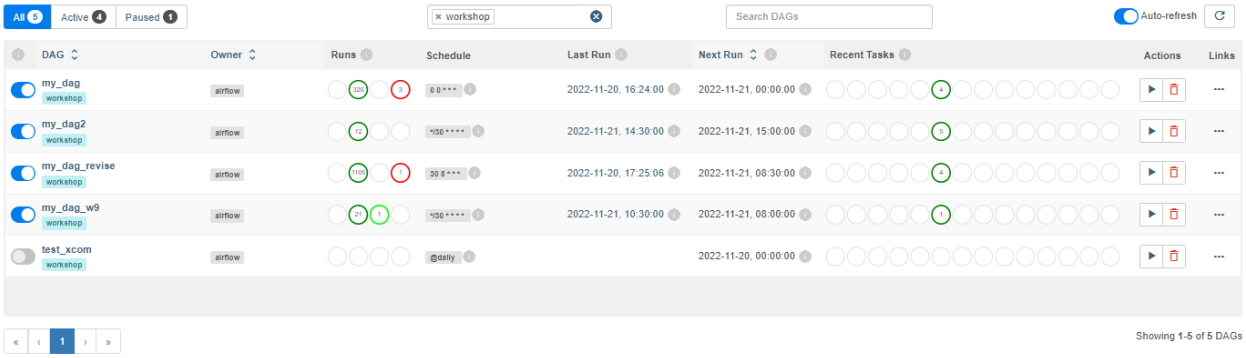


Let's Build this Airflow DAG

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

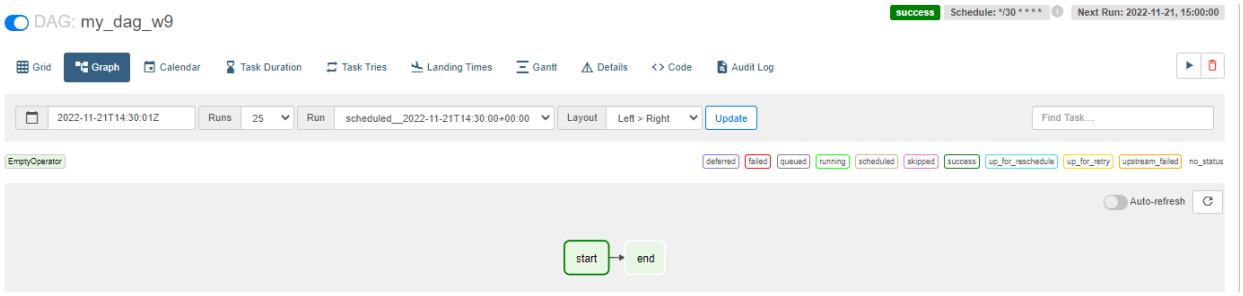


05-creating-and-scheduling-data-pipelines > dags > my_dag_w9.py

```python
from airflow import DAG
from airflow.utils import timezone
from airflow.operators.empty import EmptyOperator

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

    start = EmptyOperator(task_id = "start")

```

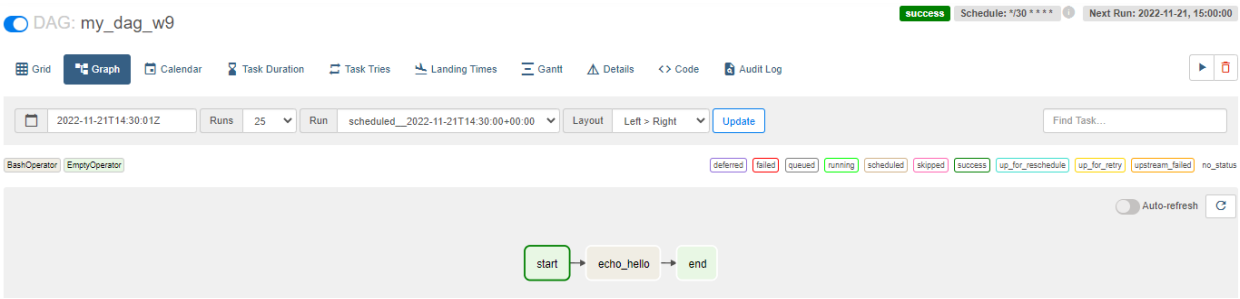
05-creating-and-scheduling-data-pipelines > dags > my_dag_w9.py

```python
from airflow import DAG
from airflow.utils import timezone
from airflow.operators.empty import EmptyOperator

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

    start = EmptyOperator(task_id = "start")

    end =  EmptyOperator(task_id = "end")

    start >> end
```



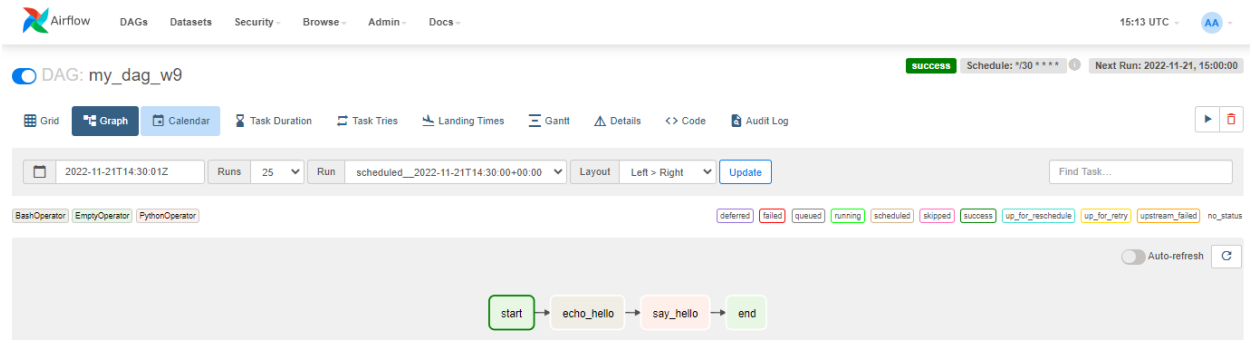
05-creating-and-scheduling-data-pipelines > dags > my_dag_w9.py

```python
from airflow import DAG
from airflow.utils import timezone
from airflow.operators.empty import EmptyOperator
from airflow.operators.bash import BashOperator

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

    start = EmptyOperator(task_id = "start")

    echo_hello = BashOperator(
        task_id = "echo_hello",
        bash_command = "echo 'hello'",
    )

    end =  EmptyOperator(task_id = "end")

    start >> echo_hello >>end

```



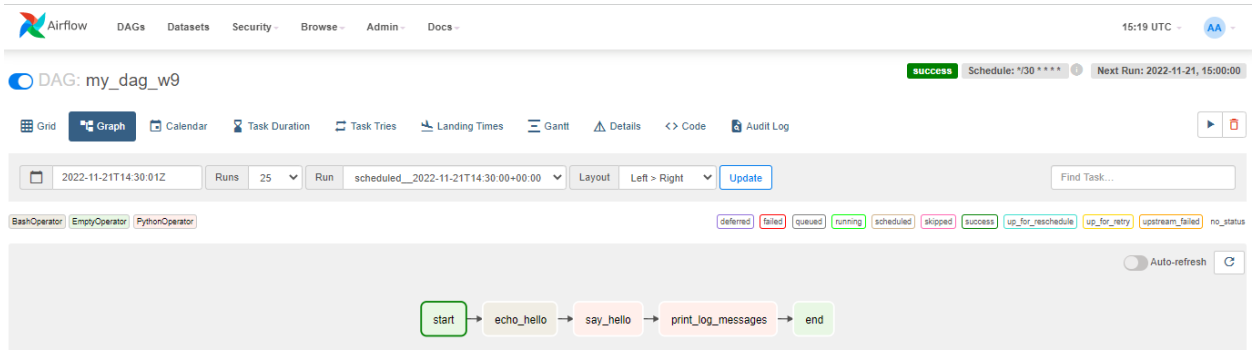
05-creating-and-scheduling-data-pipelines > dags > my_dag_w9.py

```python
from airflow import DAG
from airflow.utils import timezone
from airflow.operators.empty import EmptyOperator
from airflow.operators.bash import BashOperator
from airflow.operators.python import PythonOperator

#ชื่อฟังก์ชั่นต้องไม่เหมือนชื่อtask เลยใส่ _ นำหน้า

def _say_hello():
    print("Hello")

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

    start = EmptyOperator(task_id = "start")

    echo_hello = BashOperator(
        task_id = "echo_hello",
        bash_command = "echo 'hello'",
    )

    say_hello = PythonOperator(
        task_id = "say_hello",
        python_callable = _say_hello,

    )

    end =  EmptyOperator(task_id = "end")

    start >> echo_hello >> say_hello >> end
```

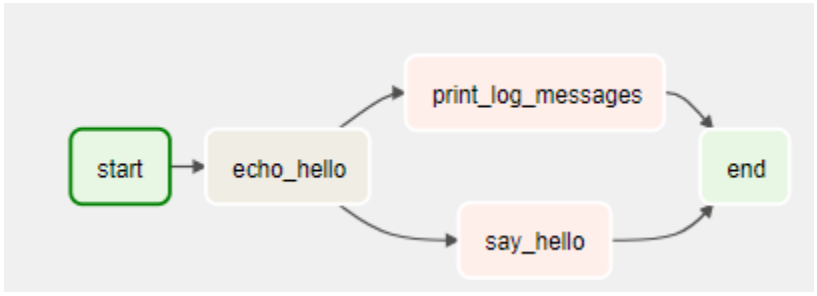


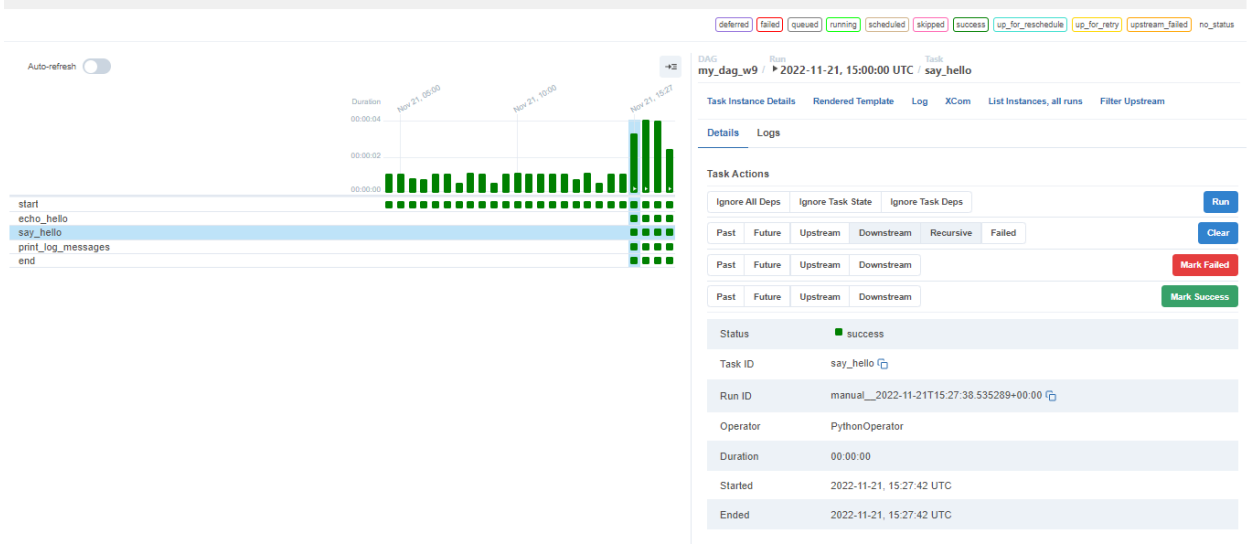
05-creating-and-scheduling-data-pipelines > dags > my_dag_w9.py

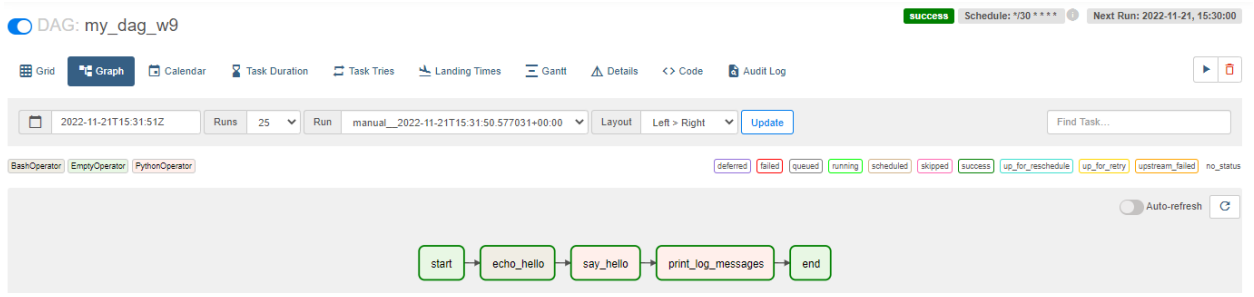
```python
import logging

from airflow import DAG
from airflow.utils import timezone
from airflow.operators.empty import EmptyOperator
from airflow.operators.bash import BashOperator
from airflow.operators.python import PythonOperator

#ชื่อฟังก์ชั่นต้องไม่เหมือนชื่อtask เลยใส่ _ นำหน้า

def _say_hello():
    print("Hello")

#logging มีระดับการlog => info, debug แล้วแต่การทำงาน
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"],
) as dag:

    start = EmptyOperator(task_id = "start")

    echo_hello = BashOperator(
        task_id = "echo_hello",
        bash_command = "echo 'hello'",
    )

    say_hello = PythonOperator(
        task_id = "say_hello",
        python_callable = _say_hello,

    )

    print_log_messages = PythonOperator(
        task_id = "print_log_messages",
        python_callable = _print_log_messages,

    )

    end =  EmptyOperator(task_id = "end")

    start >> echo_hello >> say_hello >> print_log_messages >> end

```




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







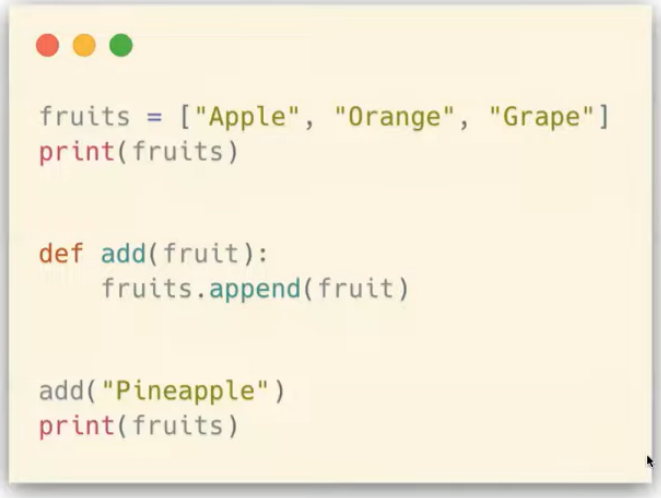
Week 9

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

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


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


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

This code changes the value stored in fruits 

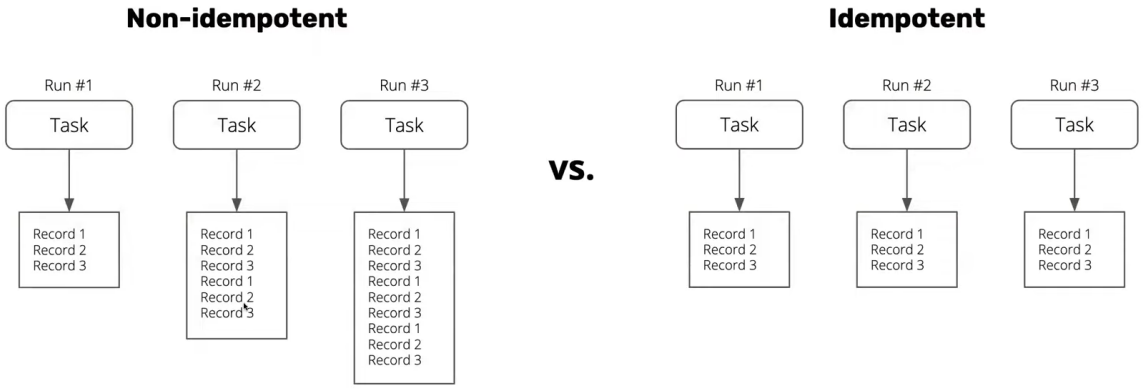
```python
fruits = ["Apple", "Orange", "Grape"]


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 

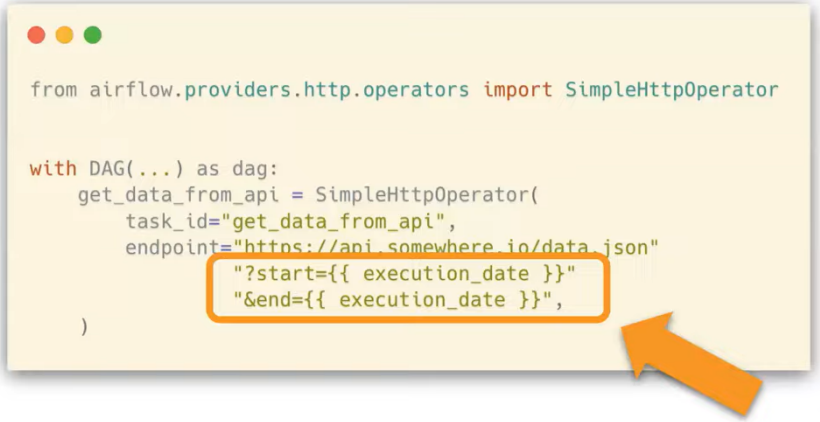
```python
fruits = ["Apple", "Orange", "Grape"]
def add(fruit):
    fruits.append(fruit)

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



Making Pipeline Idempotent



Templating and Macros

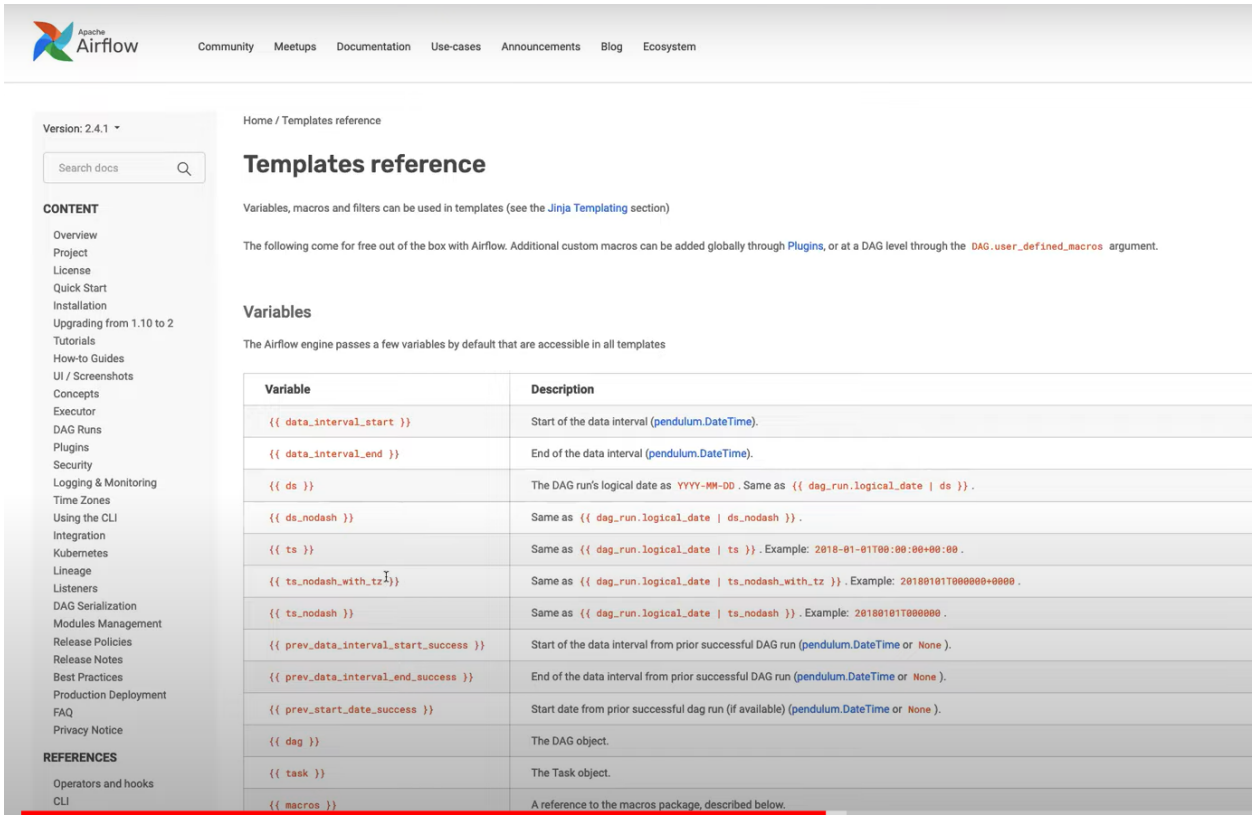
Way to pass dynamic data to your DAGs at runtime

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



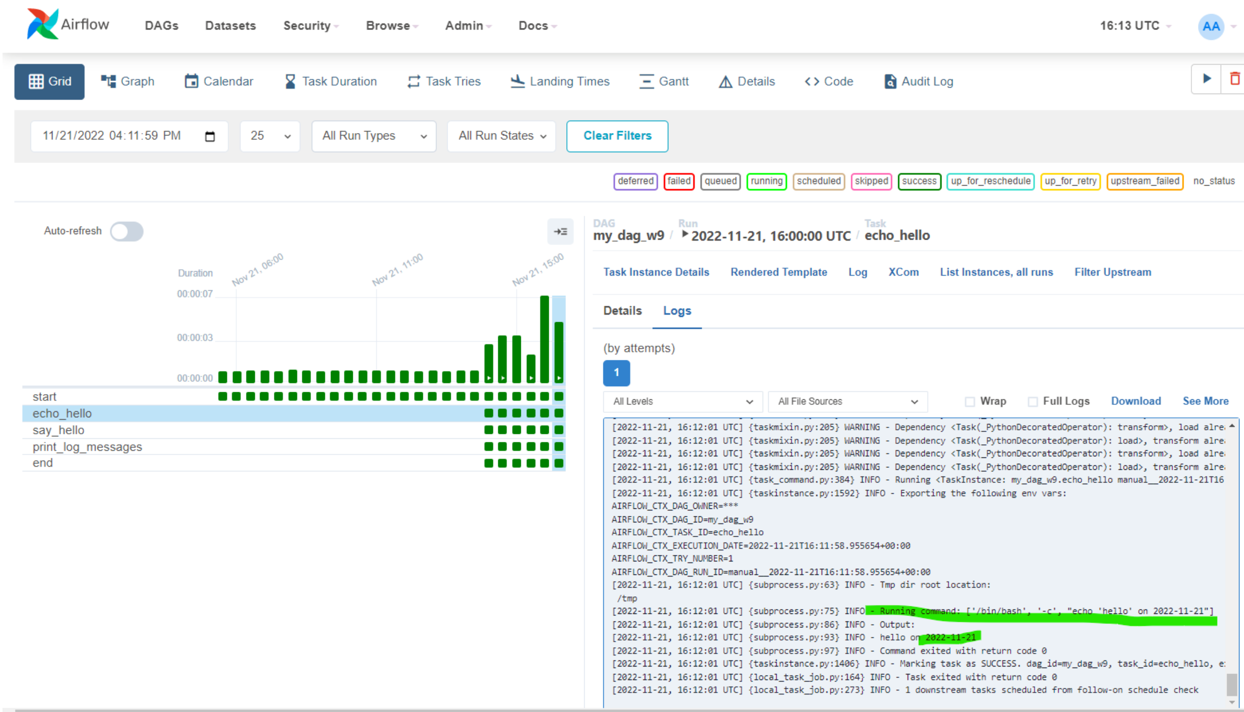
Variables

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

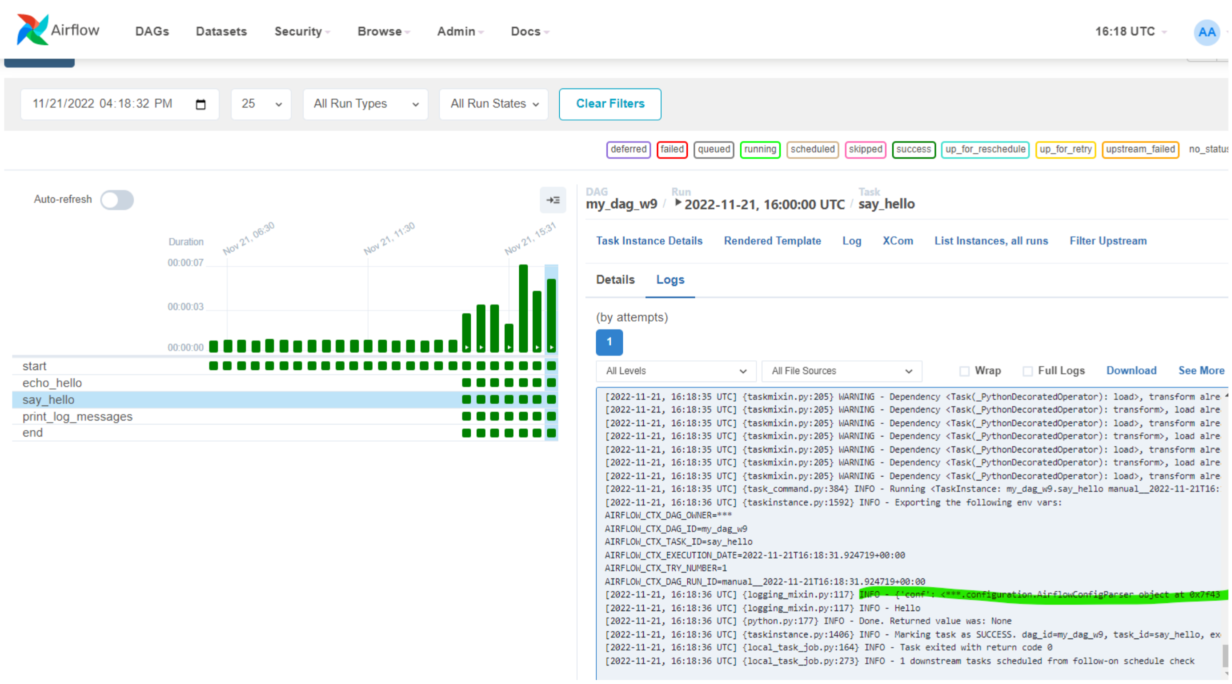
Variable	Description
`{{ data_interval_start }}`	Start of the data interval (pendulum.DateTime).
`{{ data_interval_end }}`	End of the data interval (pendulum.DateTime).
`{{ ds }}`	The DAG run's logical date as `YYYY-MM-DD` . Same as `{{ dag_run.logical_date | ds }}` .
`{{ ds_nodash }}`	Same as `{{ dag_run.logical_date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

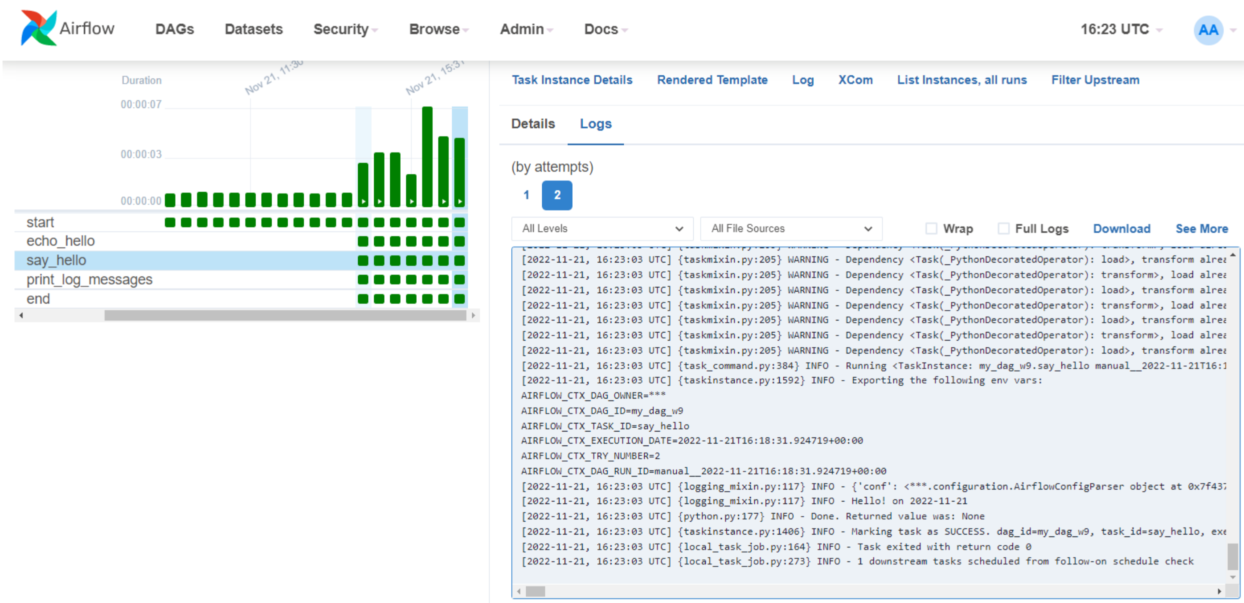
05-creating-and-scheduling-data-pipelines > dags > my_dag_w9.py

```python
import logging

from airflow import DAG
from airflow.utils import timezone
from airflow.operators.empty import EmptyOperator
from airflow.operators.bash import BashOperator
from airflow.operators.python import PythonOperator

#ชื่อฟังก์ชั่นต้องไม่เหมือนชื่อtask เลยใส่ _ นำหน้า

def _say_hello():
    print("Hello")

#logging มีระดับการlog => info, debug แล้วแต่การทำงาน
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 }}",
    )

    say_hello = PythonOperator(
        task_id = "say_hello",
        python_callable = _say_hello,

    )

    print_log_messages = PythonOperator(
        task_id = "print_log_messages",
        python_callable = _print_log_messages,

    )

    end =  EmptyOperator(task_id = "end")

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


```

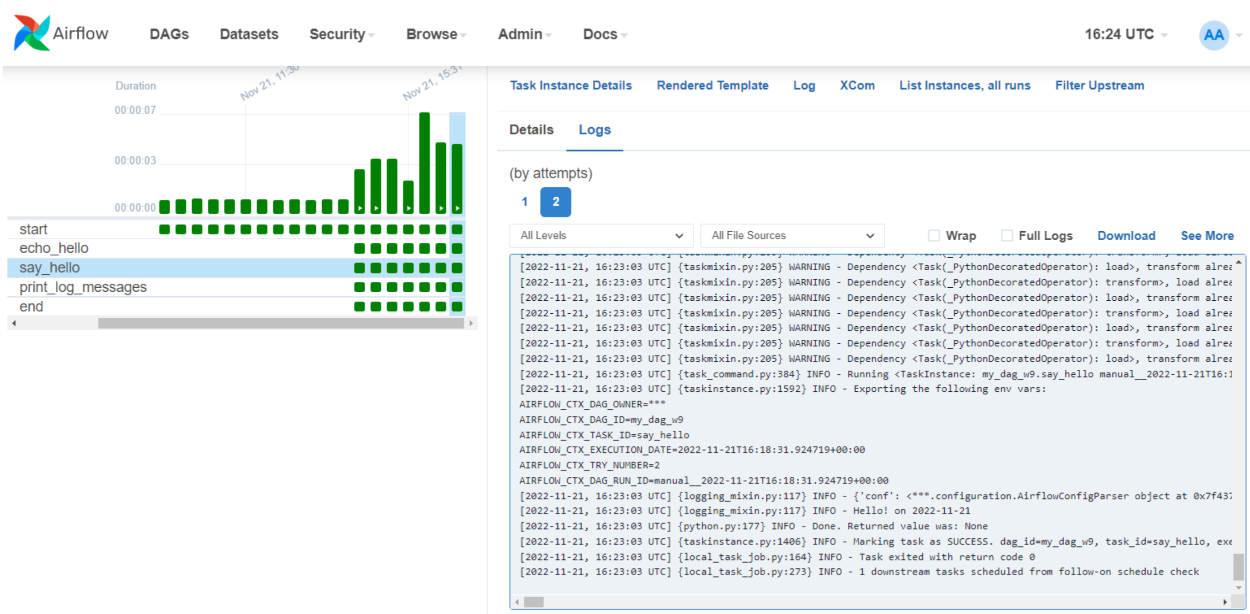


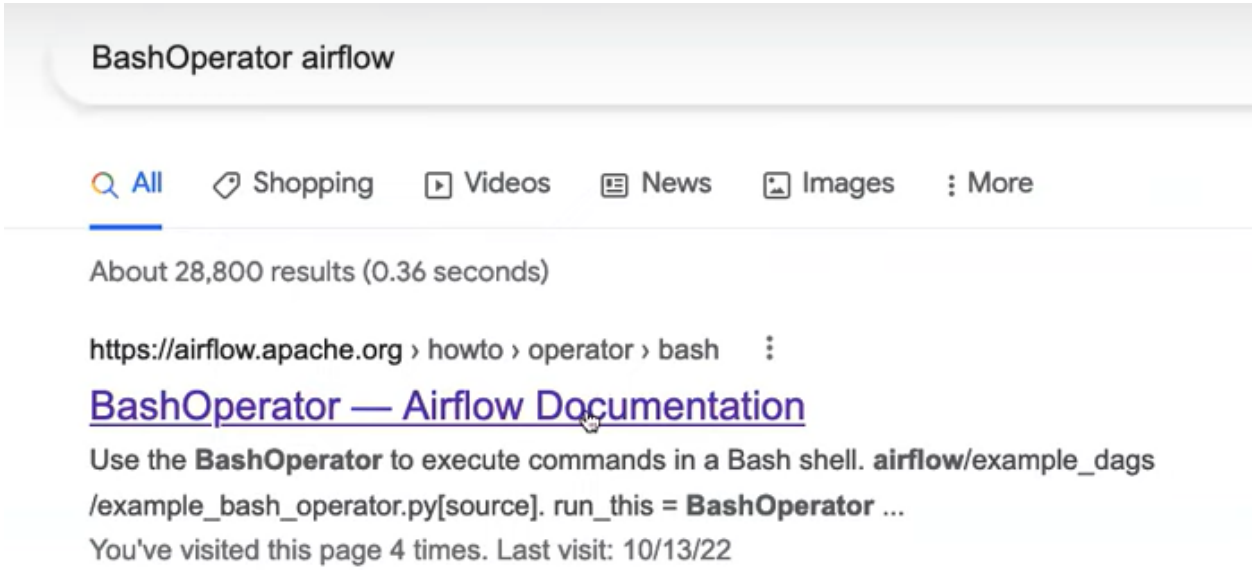

```python
def _say_hello(**context):
    print(context)
    print("Hello")
```

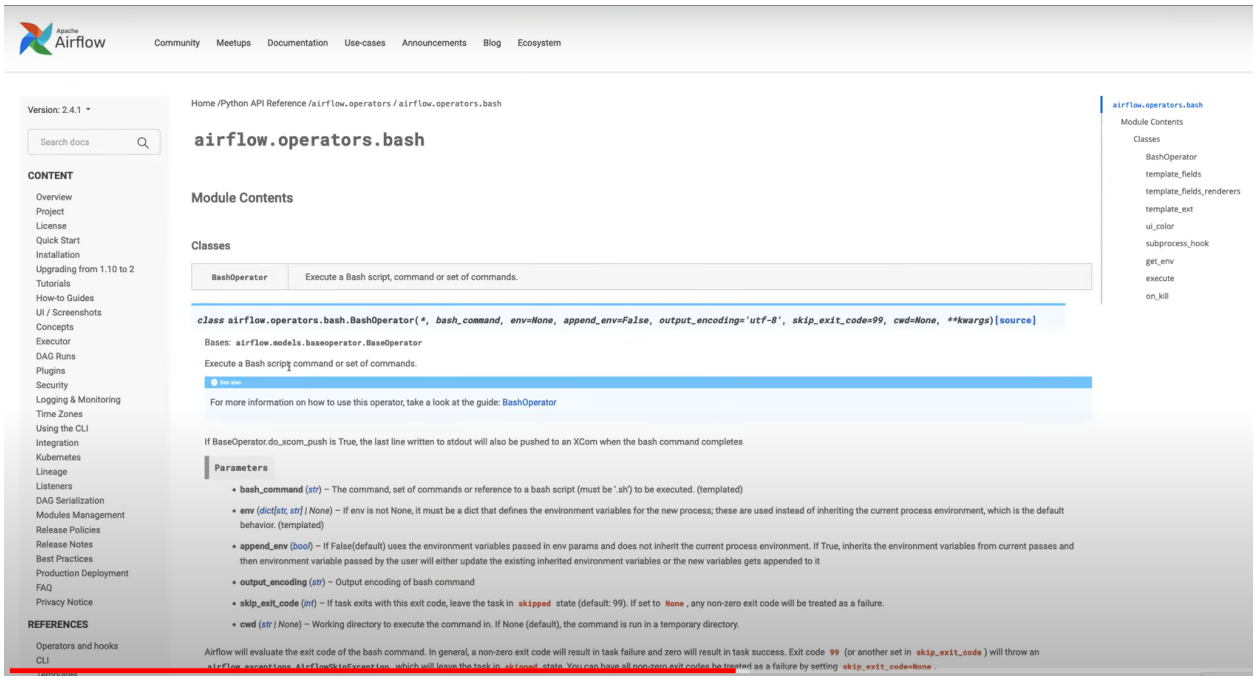





```python
def _say_hello(**context):
    print(context)
    datestamp = context["ds"]
    print(f"Hello! on {datestamp}")
```

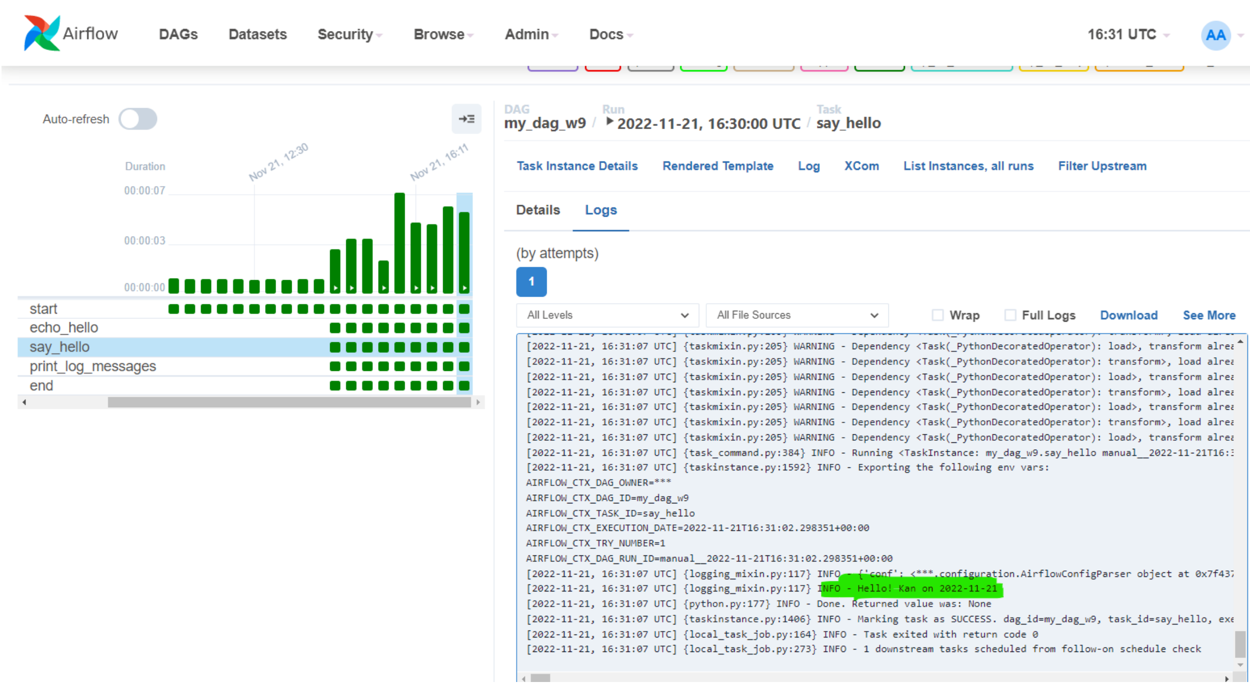




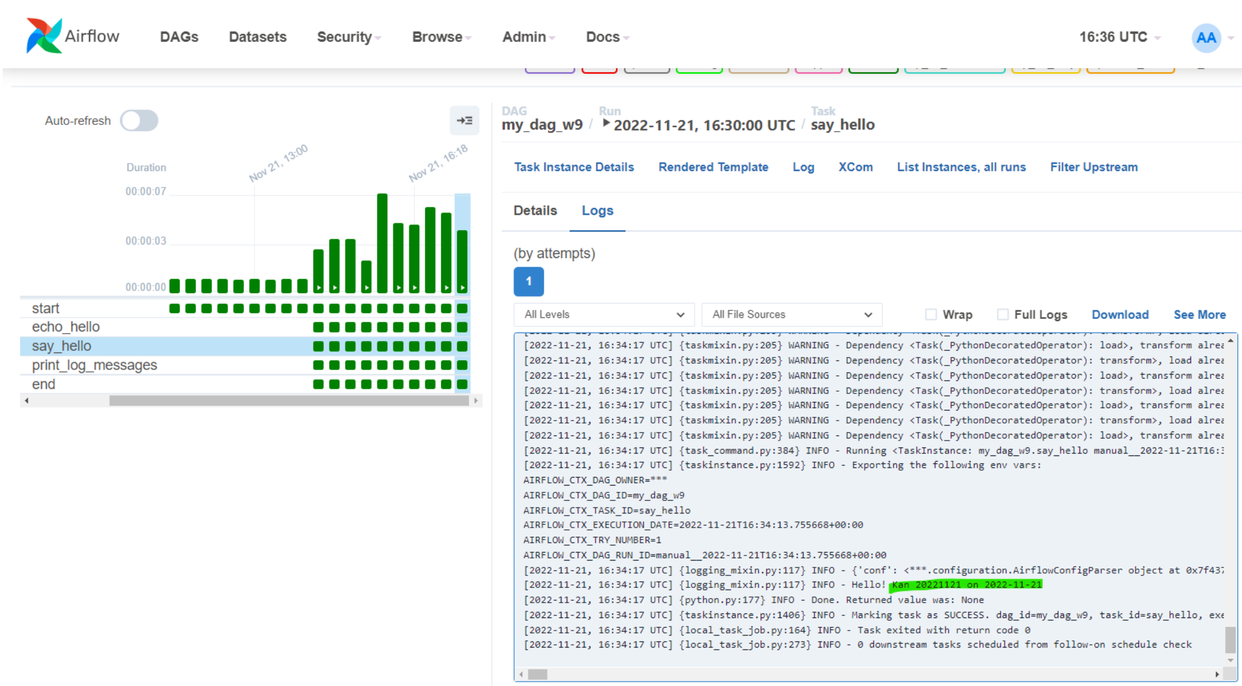



```
template_fields:Sequence[str] = ['templates_dict', 'op_args', 'op_kwargs'] [source]
```

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

#logging มีระดับการlog => info, debug แล้วแต่การทำงาน
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",
        }

    )
```

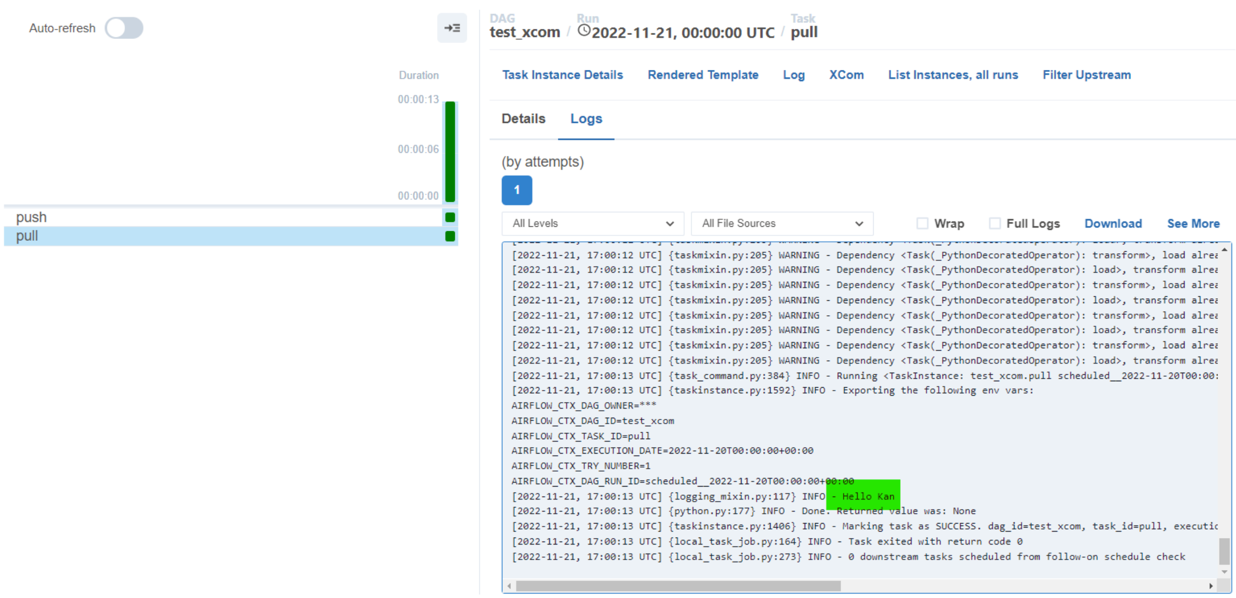


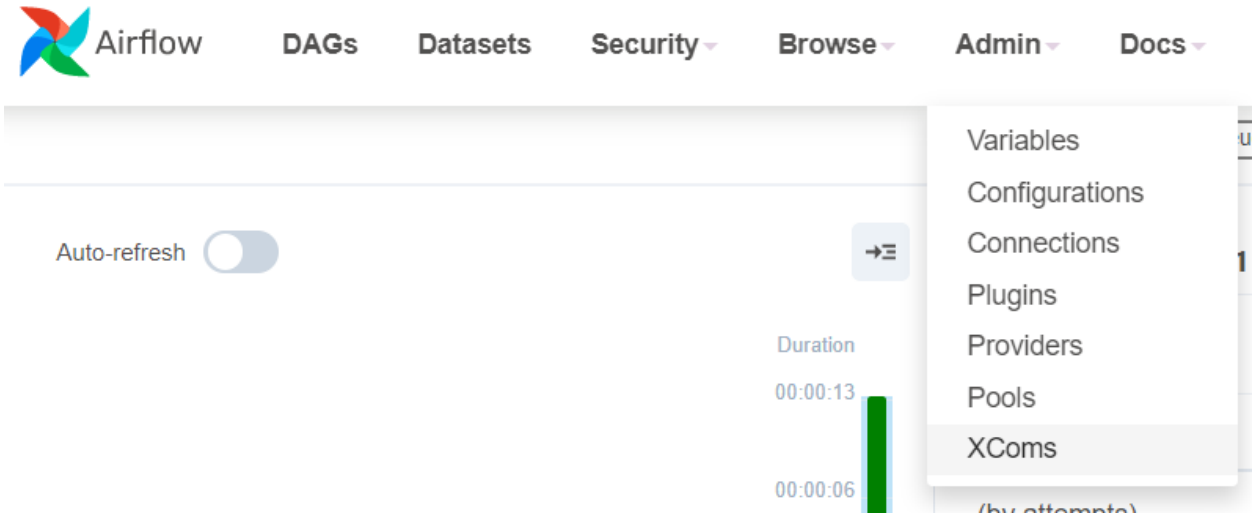
```python
def _say_hello( name="", **context ):
    print(context)
    datestamp = context["ds"]
    print(f"Hello! {name} on {datestamp}")

#logging มีระดับการlog => info, debug แล้วแต่การทำงาน
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 {{ ds_nodash }}",
        }
    )
```

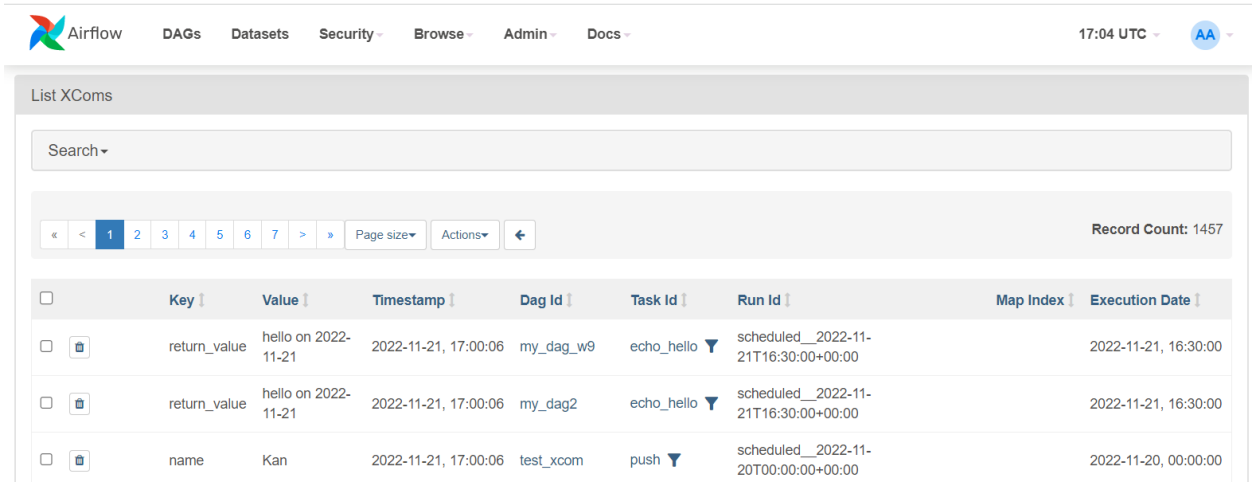


05-creating-and-scheduling-data-pipelines > dags > test_xcom.py

```python
from airflow import DAG
from airflow.utils import timezone
from airflow.operators.python import PythonOperator

#task instance มาจาก context
def _push(**context):
    ti = context["ti"]
    ti.xcom_push(key="name", value="Kan")

    #return ["Kan", "Chin", "Peeyapak", "Lyn", "Peerawit", "Kruewan", "Pongthanin", "Mindii", "Nuttapol", "Nattakan",]

def _pull(**context):
    ti = context["ti"]
    name = ti.xcom_pull(task_ids="push", key="name")
    print(f"Hello {name}")

with DAG(
    "test_xcom",
    start_date=timezone.datetime(2022, 10, 15),
    schedule="@daily",
    tags=["workshop"],
    catchup=False,
) as dag:

    push = PythonOperator(
        task_id="push",
        python_callable=_push,
    )

    pull = PythonOperator(
        task_id="pull",
        python_callable=_pull,
    )

    push >> pull
```

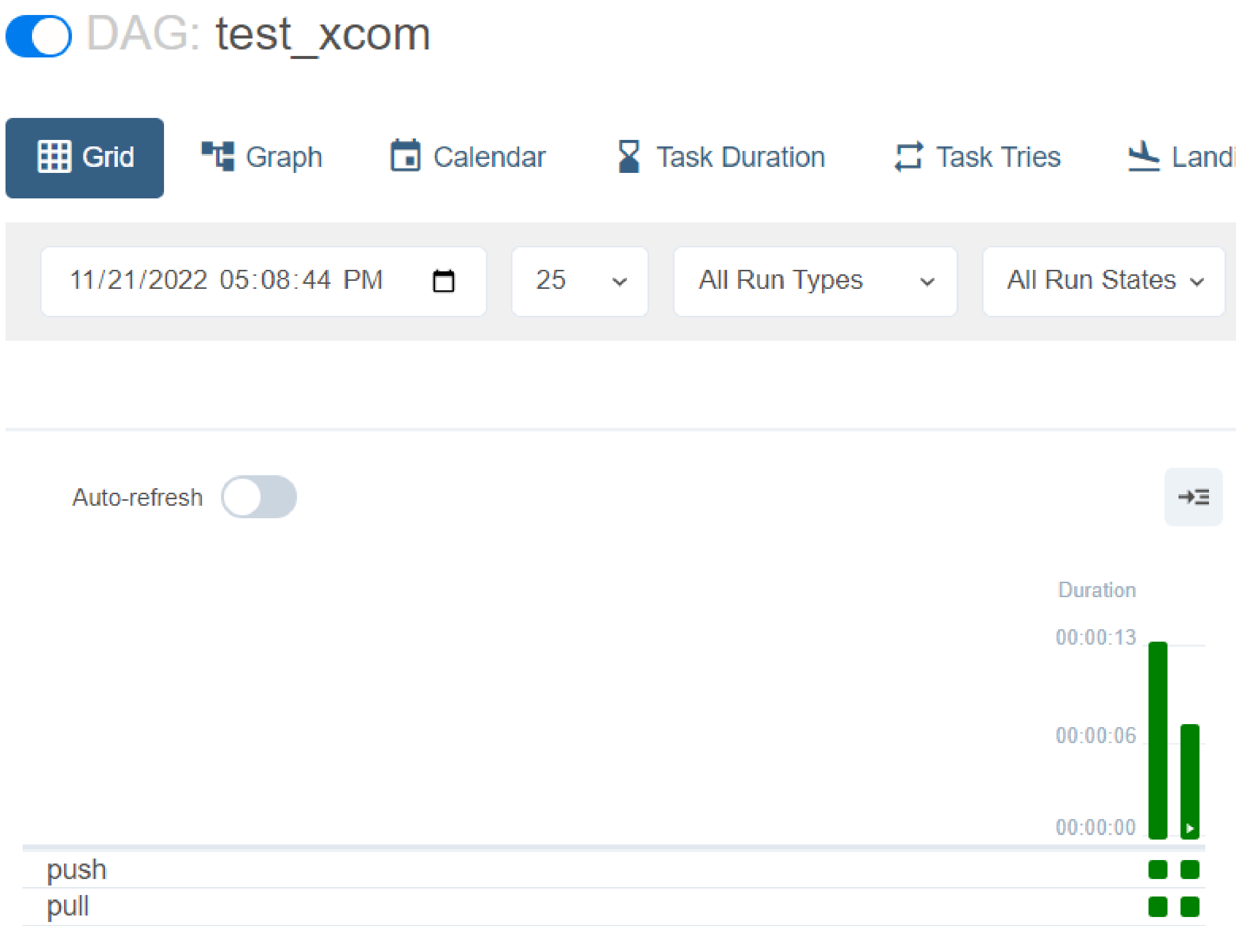


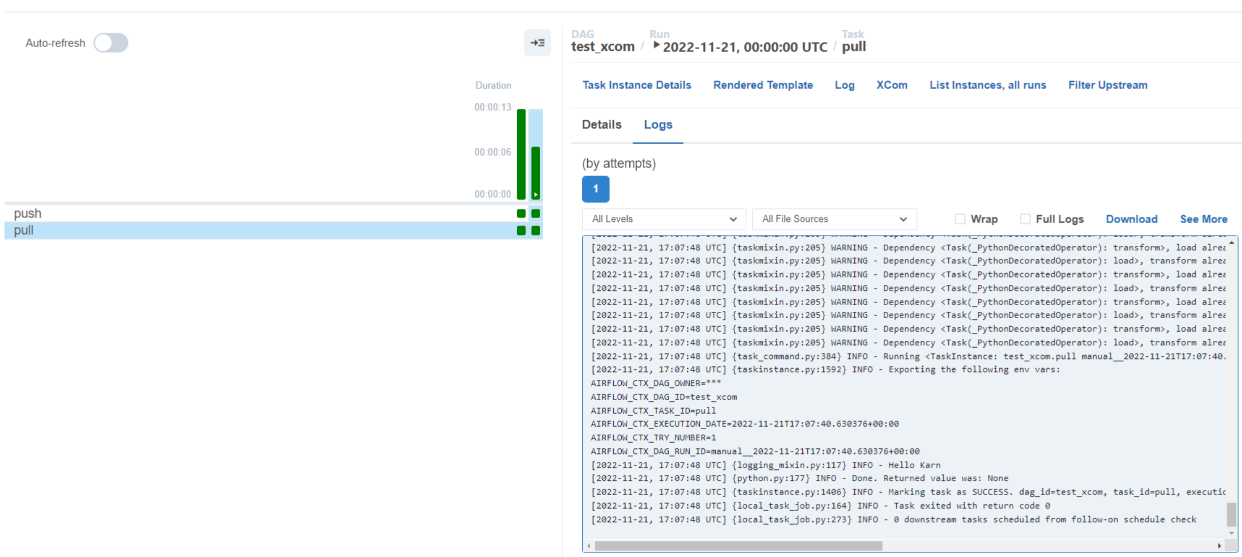

Airflow
DAGs
Datasets
Security
Browse
Admin
Docs
Variables
Configurations
Connections
Plugins
Providers
Pools
XComs
Auto-refresh
Duration
00:00:13
00:00:06

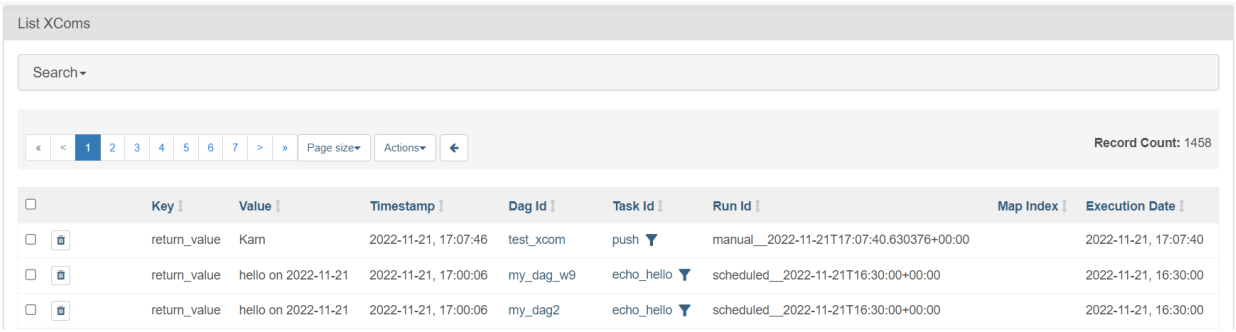

Airflow
DAGs
Datasets
Security
Browse
Admin
Docs
17:04 UTC
AA
List XComs
Search
Page size
Actions
Record Count: 1457
Key
Value
Timestamp
Dag Id
Task Id
Run Id
Map Index
Execution Date
return_value
hello on 2022-11-21
2022-11-21, 17:00:06
my_dag_w9
echo_hello
scheduled__2022-11-21T16:30:00+00:00
2022-11-21, 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

```python
from airflow import DAG
from airflow.utils import timezone
from airflow.operators.python import PythonOperator

#task instance และ context
def _push(**context):
    # ti = context["ti"]
    # ti.xcom_push(key="name", value="Kan")

    #return ["Kan", "Chin", "Peeyapak", "Lyn", "Peerawit", "Kru
    return "Karn"

def _pull(**context):
    ti = context["ti"]
    name = ti.xcom_pull(task_ids="push", key="return_value")
    print(f"Hello {name}")

with DAG(
    "test_xcom",
    start_date=timezone.datetime(2022, 10, 15),
    schedule="@daily",
    tags=["workshop"],
    catchup=False,
) as dag:

    push = PythonOperator(
        task_id="push",
        python_callable=_push,
    )

    pull = PythonOperator(
        task_id="pull",
        python_callable=_pull,
    )

    push >> pull
```

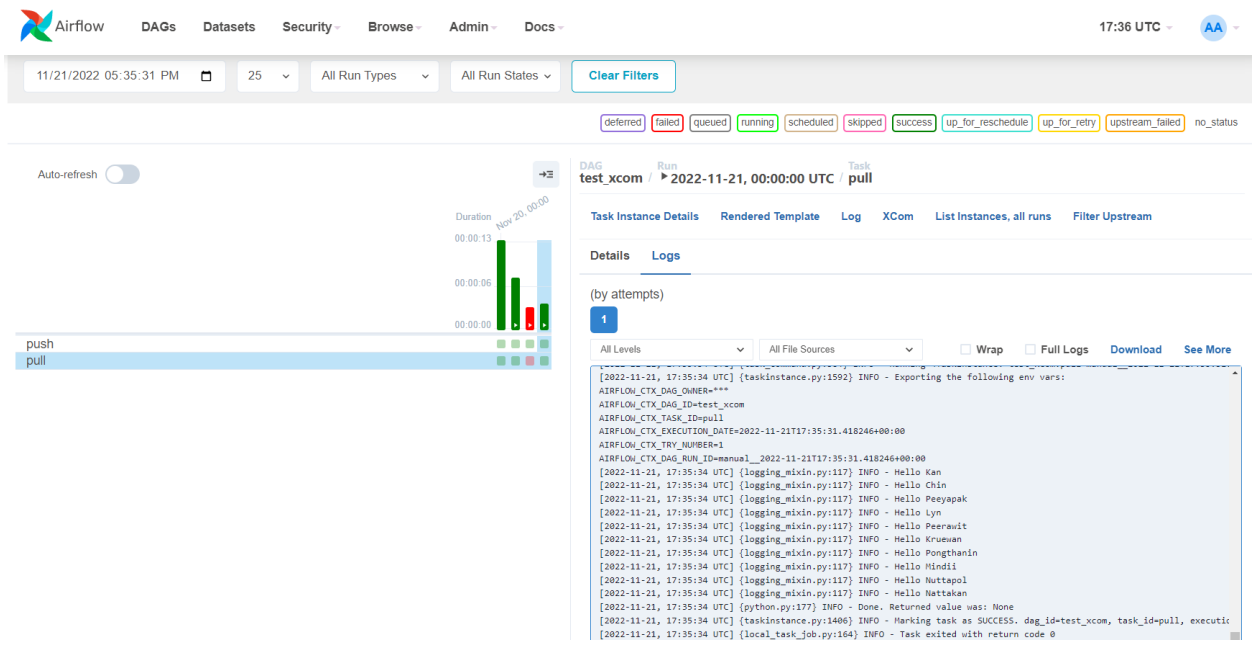

DAG: test_xcom
Grid
Graph
Calendar
Task Duration
Task Tries
Landi
11/21/2022 05:08:44 PM
25
All Run Types
All Run States
Auto-refresh
Duration
00:00:13
00:00:06
00:00:00
push
pull

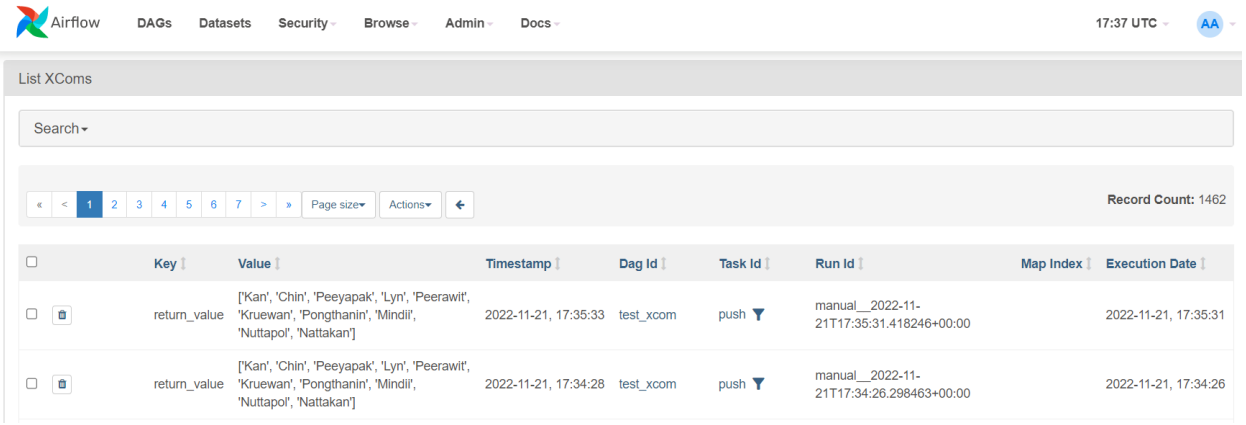

Auto-refresh
Duration
00:00:13
00:00:06
00:00:00
push
pull
DAG
test_xcom
Run
2022-11-21, 00:00:00 UTC
Task
pull
Task Instance Details
Rendered Template
Log
XCom
List Instances, all runs
Filter Upstream
Details
Logs
(by attempts)
1
All Levels
All File Sources
Wrap
Full Logs
Download
See More
[2022-11-21, 17:07:48 UTC] {taskmixin.py:205} WARNING - Dependency <Task(_PythonDecoratedOperator): transform>, load alrea
[2022-11-21, 17:07:48 UTC] {taskmixin.py:205} WARNING - Dependency <Task(_PythonDecoratedOperator): load>, transform alrea
[2022-11-21, 17:07:48 UTC] {taskmixin.py:205} WARNING - Dependency <Task(_PythonDecoratedOperator): transform>, load alrea
[2022-11-21, 17:07:48 UTC] {taskmixin.py:205} WARNING - Dependency <Task(_PythonDecoratedOperator): load>, transform alrea
[2022-11-21, 17:07:48 UTC] {taskmixin.py:205} WARNING - Dependency <Task(_PythonDecoratedOperator): transform>, load alrea
[2022-11-21, 17:07:48 UTC] {taskmixin.py:205} WARNING - Dependency <Task(_PythonDecoratedOperator): load>, transform alrea
[2022-11-21, 17:07:48 UTC] {taskmixin.py:205} WARNING - Dependency <Task(_PythonDecoratedOperator): transform>, load alrea
[2022-11-21, 17:07:48 UTC] {taskmixin.py:205} WARNING - Dependency <Task(_PythonDecoratedOperator): load>, transform alrea
[2022-11-21, 17:07:48 UTC] {task_command.py:384} INFO - Running <TaskInstance: test_xcom.pull manual__2022-11-21T17:07:40.
[2022-11-21, 17:07:48 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=2022-11-21T17:07:40.630376+00:00
AIRFLOW_CTX_TRY_NUMBER=1
AIRFLOW_CTX_DAG_RUN_ID=manual__2022-11-21T17:07:40.630376+00:00
[2022-11-21, 17:07:48 UTC] {logging_mixin.py:117} INFO - Hello Karn
[2022-11-21, 17:07:48 UTC] {python.py:177} INFO - Done. Returned value was: None
[2022-11-21, 17:07:48 UTC] {taskinstance.py:1406} INFO - Marking task as SUCCESS. dag_id=test_xcom, task_id=pull, executic
[2022-11-21, 17:07:48 UTC] {local_task_job.py:164} INFO - Task exited with return code 0
[2022-11-21, 17:07:48 UTC] {local_task_job.py:273} INFO - 0 downstream tasks scheduled from follow-on schedule check



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

```python
from airflow import DAG
from airflow.utils import timezone
from airflow.operators.python import PythonOperator

#task instance และ context
def _push(**context):
    # ti = context["ti"]
    # ti.xcom_push(key="name", value="Kan")

    return ["Kan", "Chin", "Peeyapak", "Lyn", "Peerawit", "Kruewan", "Pongthanin", "Mindii", "Nuttapol", "Nattakan",]
    #return "Karn"

def _pull(**context):
    ti = context["ti"]
    name = ti.xcom_pull(task_ids="push", key="return_value")

    for each in name:
        print(f"Hello {each}")


with DAG(
    "test_xcom",
    start_date=timezone.datetime(2022, 10, 15),
    schedule="@daily",
    tags=["workshop"],
    catchup=False,
) as dag:

    push = PythonOperator(
        task_id="push",
        python_callable=_push,
    )

    pull = PythonOperator(
        task_id="pull",
        python_callable=_pull,
    )

    push >> pull
```





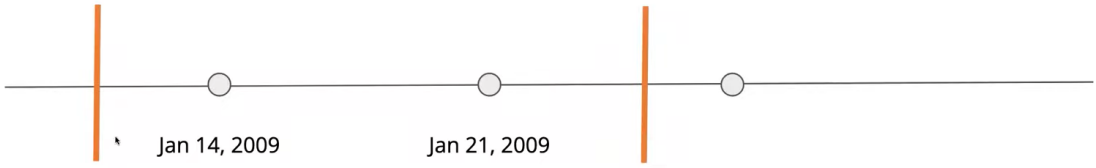
Using Backfill in Airflow

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
```

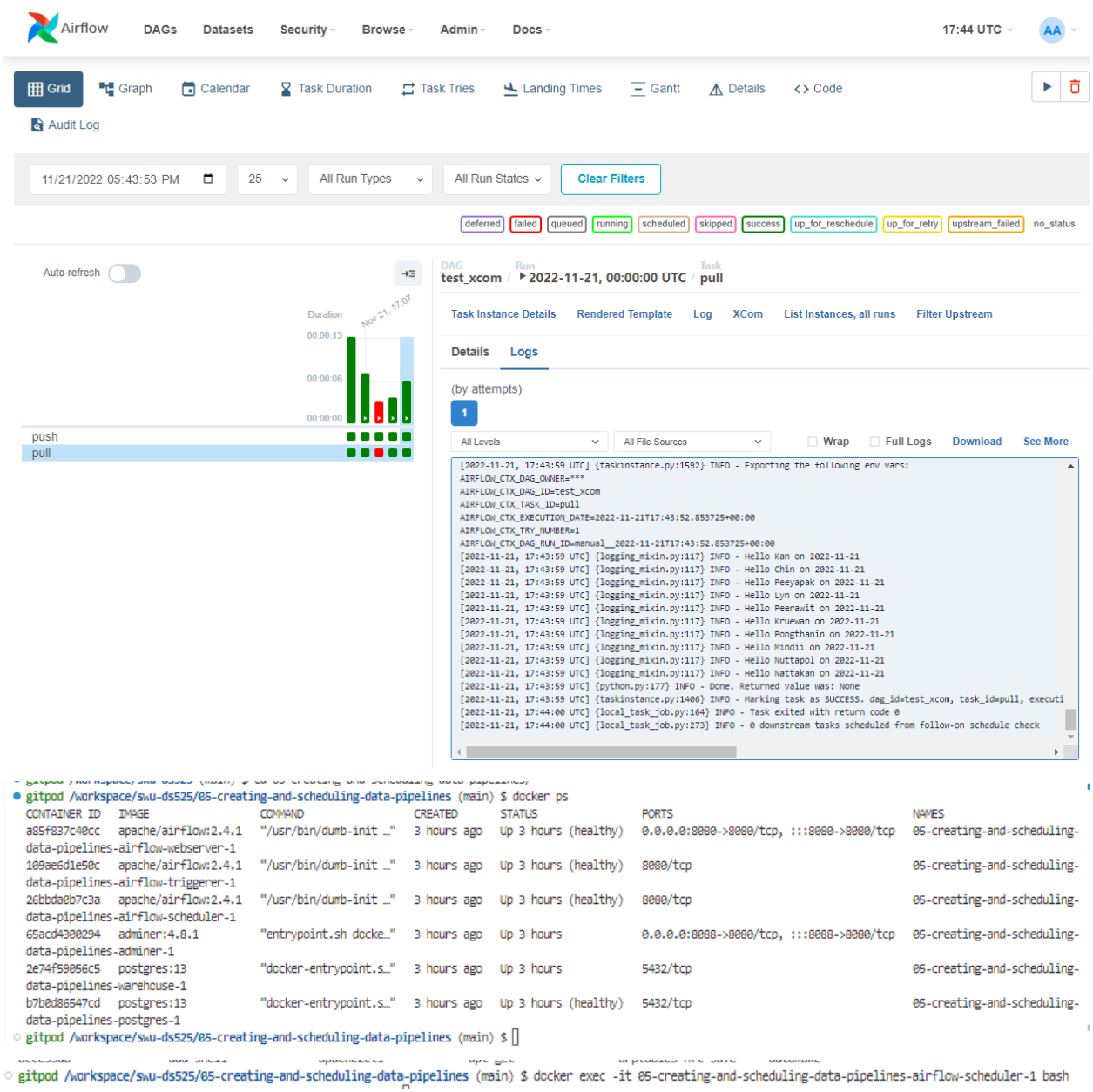
Backfill Explained

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



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

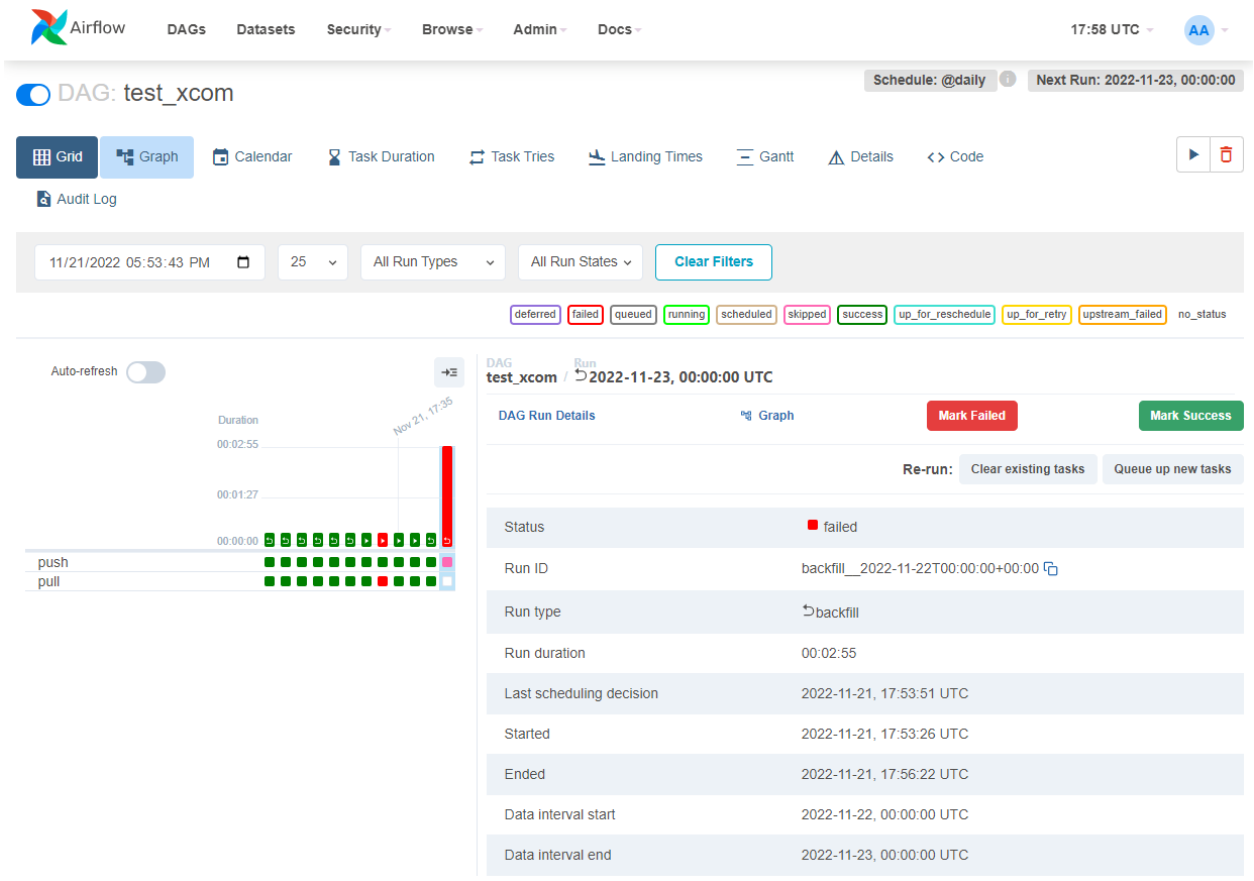
```python
from airflow import DAG
from airflow.utils import timezone
from airflow.operators.python import PythonOperator

#task instance ภายใน context
def _push(**context):
    # ti = context["ti"]
    # ti.xcom_push(key="name", value="Kan")

    return ["Kan", "Chin", "Peeyapak", "Lyn", "Peerawit", "Kruewan", "Pongthanin", "Mindii", "Nuttapol", "Nattakan",]
    #return "Karn"

def _pull(**context):
    ti = context["ti"]
    ds = context["ds"]

    name = ti.xcom_pull(task_ids="push", key="return_value")

    for each in name:
        print(f"Hello {each} on {ds}")

with DAG(
    "test_xcom",
    start_date=timezone.datetime(2022, 10, 15),
    schedule="@daily",
    tags=["workshop"],
    catchup=False,
) as dag:

    push = PythonOperator(
        task_id="push",
        python_callable=_push,
    )

    pull = PythonOperator(
        task_id="pull",
        python_callable=_pull,
    )

    push >> pull
```

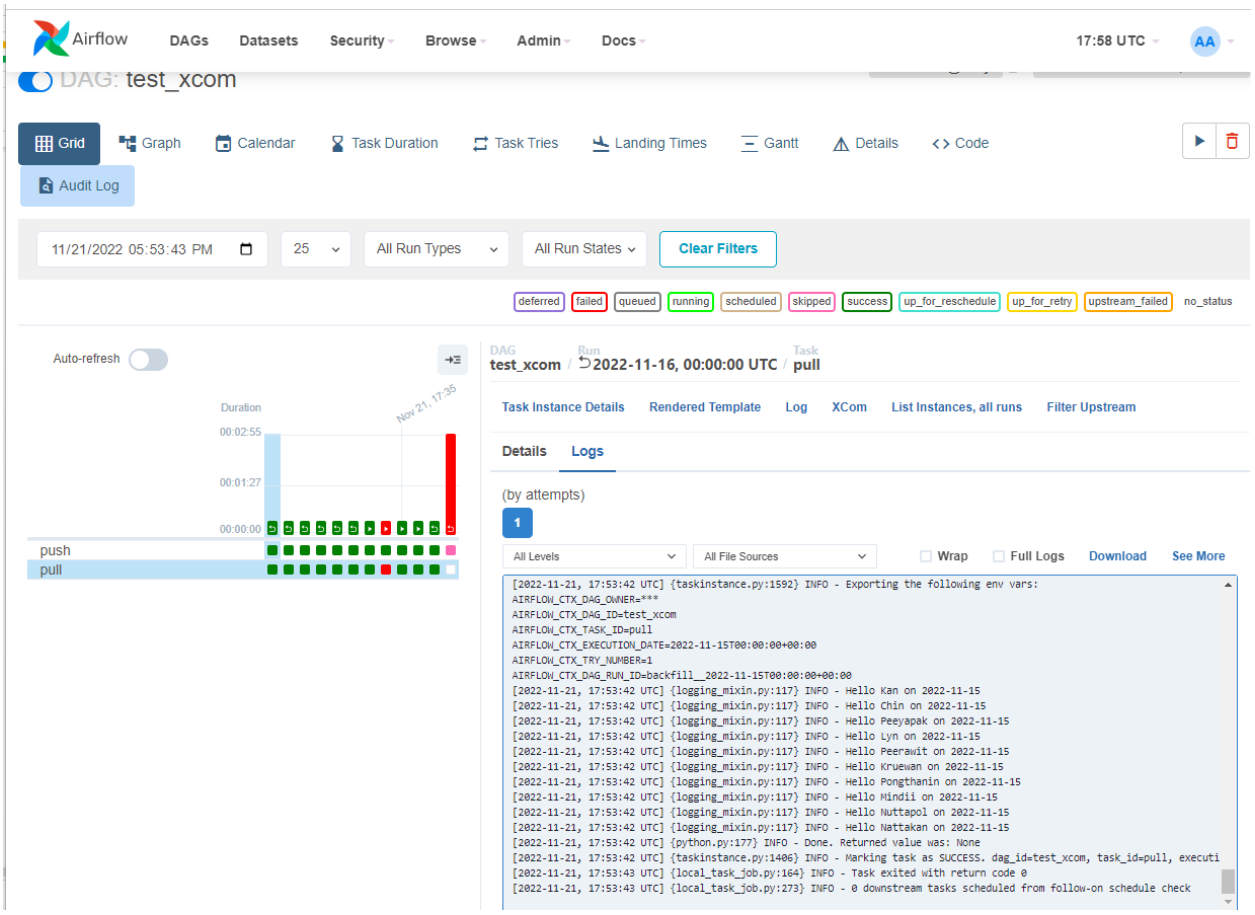



```
gitpod /workspace/swu-ds525/05-creating-and-scheduling-data-pipelines (main) $ docker ps
CONTAINER ID   IMAGE                  COMMAND                  CREATED       STATUS                   PORTS                                       NAMES
a85f837c40cc   apache/airflow:2.4.1   "/usr/bin/dumb-init …"   3 hours ago   Up 3 hours (healthy)     0.0.0.0:8080->8080/tcp, :::8080->8080/tcp   05-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                                    05-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                                    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
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) $
```

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


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
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
Duration
Nov 21, 17:35
00:02:55
00:01:27
00:00:00
push
pull
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
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 05-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 05-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 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

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

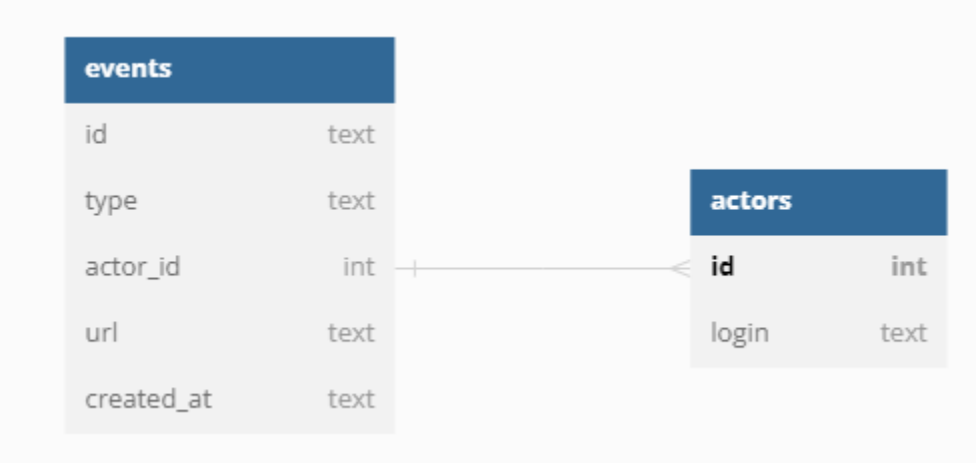
--------- --------- --------------------------------- ------------

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

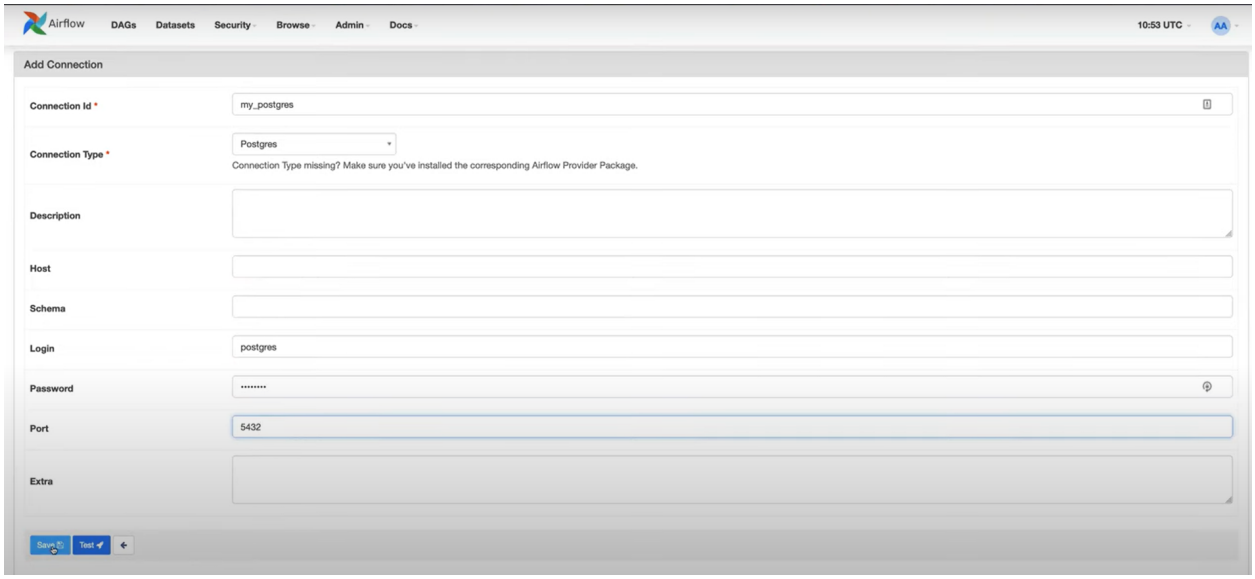
Data Model



Week 9

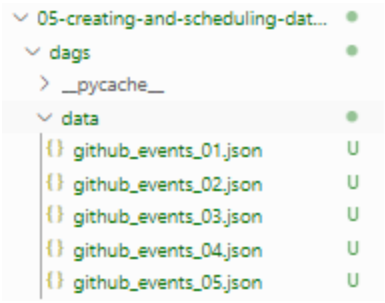
Airflow Practices

- Use date as a partition and add dates throughout the pipeline. Idempotent!
- Use UPSERT (insert or update) since INSERT may lead to duplicate rows when re-run a task
- To share large data between tasks, use a remote storage such as S3/HDFS
- Use Airflow's Connections to store sensitive data securely and retrieve them using a unique connection ID
- Always test a DAG



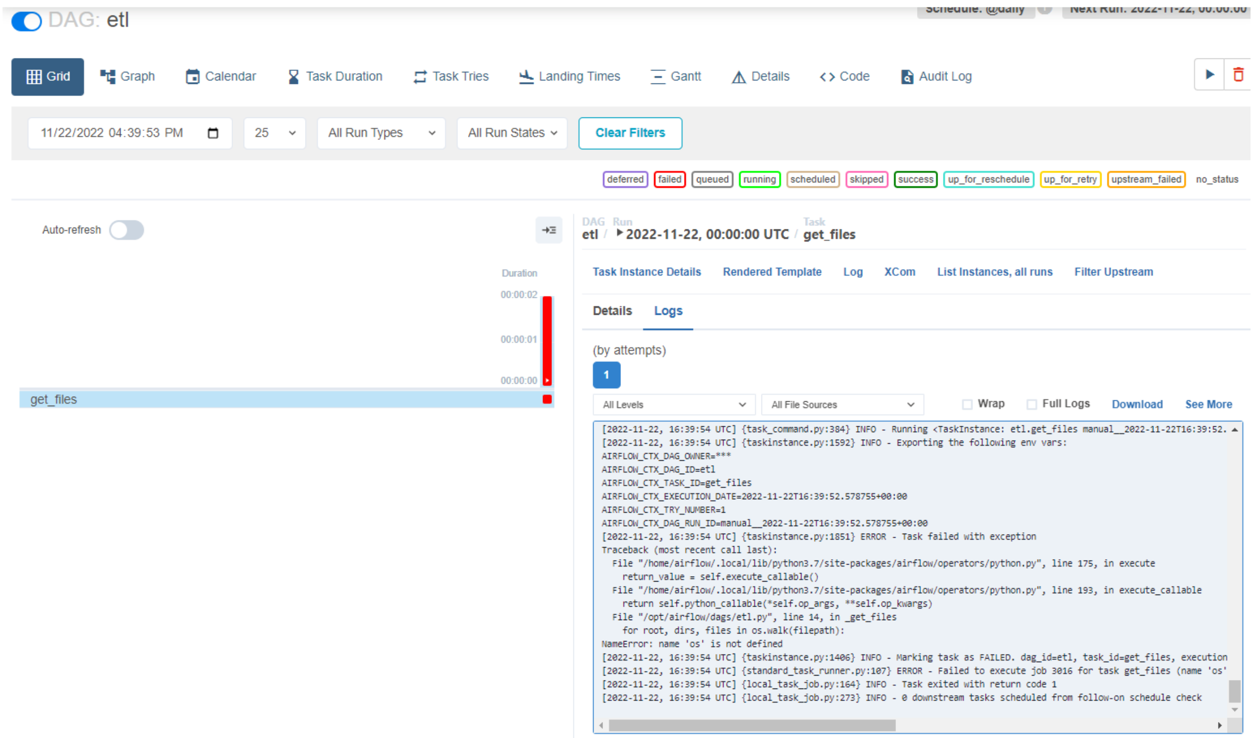
05-creating-and-scheduling-data-pipelines > dags > etl.py

```python
from airflow import DAG
from airflow.utils import timezone
from airflow.operators.python import PythonOperator

with DAG(
    "etl",
    start_date = timezone.datetime(2022, 11, 22),
    schedule = "@daily",
    tags = ["workshop"],
    catchup = False,
) as dag:
```



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

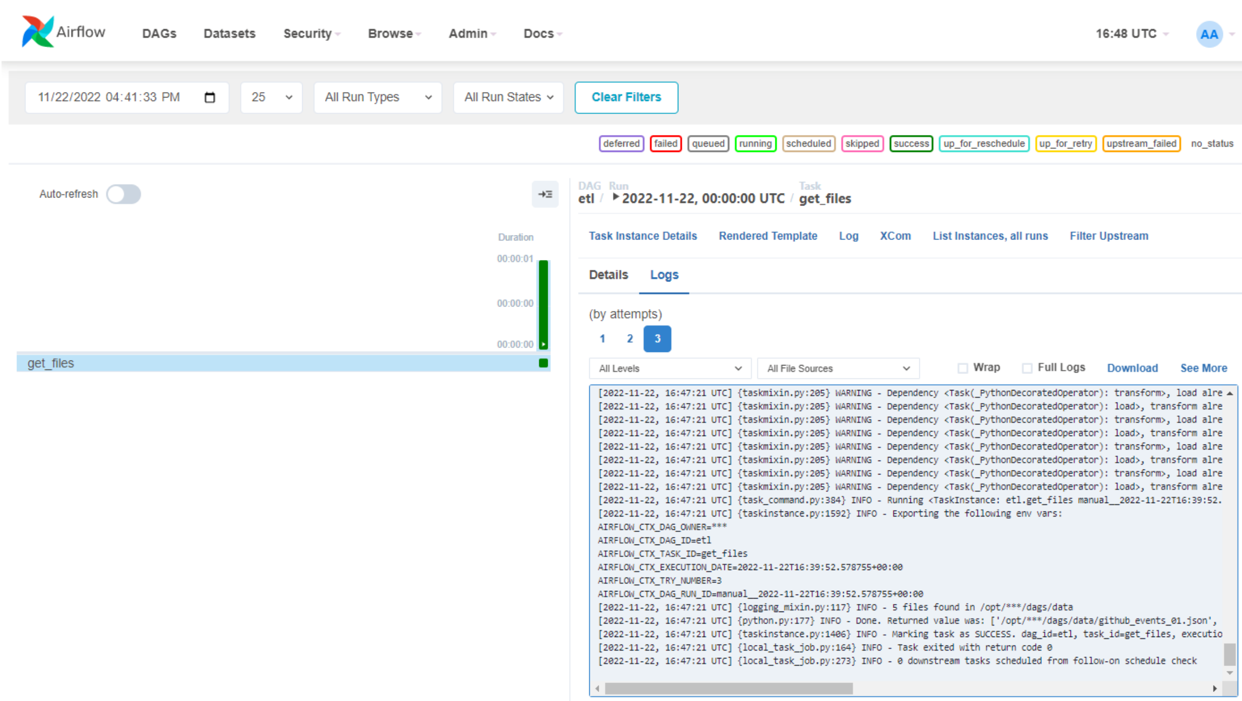
```python
from typing import List

from airflow import DAG
from airflow.utils import timezone
from airflow.operators.python import PythonOperator


def _get_files(filepath: str) -> List[str]:
    """
    Description: This function is responsible for listing the files in a directory
    """

    all_files = []
    for root, dirs, files in os.walk(filepath):
        files = glob.glob(os.path.join(root, "*.json"))
        for f in files:
            all_files.append(os.path.abspath(f))

    num_files = len(all_files)
    print(f"{num_files} files found in {filepath}")

    return all_files

#อ่านข้อมูลจากfolder data

with DAG(
    "etl",
    start_date = timezone.datetime(2022, 11, 22),
    schedule = "@daily",
    tags = ["workshop"],
    catchup = False,
) as dag:

    get_files = PythonOperator(
        task_id = "get_files",
        python_callable = _get_files,
        op_kwargs={
            "filepath":"./data",
        }
    )
```

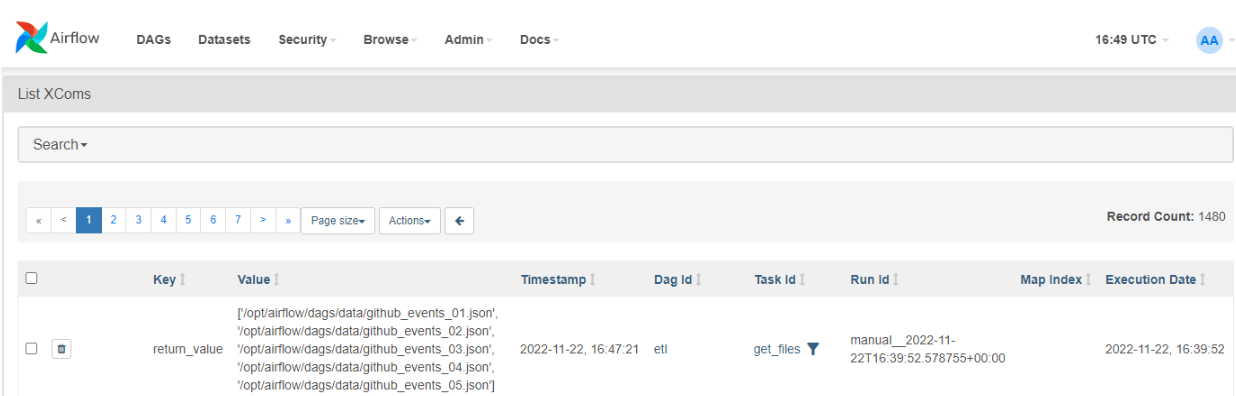



```
gitpod /workspace/swu-ds525 (main) $ cd 05-creating-and-scheduling-data-pipelines/
gitpod /workspace/swu-ds525/05-creating-and-scheduling-data-pipelines (main) $ cd ../
gitpod /workspace/swu-ds525 (main) $ docker exec -it 05-creating-and-scheduling-data-pipelines-
05-creating-and-scheduling-data-pipelines-adminer-1           05-creating-and-scheduling-data-pipelines-airflow-triggerer-1  05-creating-and-scheduling-data-pipelines-postgres-1
05-creating-and-scheduling-data-pipelines-airflow-scheduler-1  05-creating-and-scheduling-data-pipelines-airflow-webserver-1  05-creating-and-scheduling-data-pipelines-warehouse-1
gitpod /workspace/swu-ds525 (main) $ docker exec -it 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 (main) $ docker exec -it 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 (main) $ docker exec -it 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 (main) $ docker exec -it 05-creating-and-scheduling-data-pipelines-airflow-scheduler-1 bash
default@eb5fa062788b:/opt/airflow$ pwd
/opt/airflow
default@eb5fa062788b:/opt/airflow$ ls
airflow.cfg  dags  logs  plugins  webserver_config.py
default@eb5fa062788b:/opt/airflow$ cd dags/
default@eb5fa062788b:/opt/airflow/dags$ cd data/
default@eb5fa062788b:/opt/airflow/dags/data$ ls -ltr
total 776
-rw-r--r-- 1 default 33333 337840 Nov 22 16:31 github_events_03.json
-rw-r--r-- 1 default 33333 169666 Nov 22 16:31 github_events_02.json
-rw-r--r-- 1 default 33333  73641 Nov 22 16:31 github_events_01.json
-rw-r--r-- 1 default 33333 164062 Nov 22 16:31 github_events_05.json
-rw-r--r-- 1 default 33333  38882 Nov 22 16:31 github_events_04.json
default@eb5fa062788b:/opt/airflow/dags/data$ pwd
/opt/airflow/dags/data
default@eb5fa062788b:/opt/airflow/dags/data$ 
```

```python
with DAG(
    "etl",
    start_date = timezone.datetime(2022, 11, 22),
    schedule = "@daily",
    tags = ["workshop"],
    catchup = False,
) as dag:

    get_files = PythonOperator(
        task_id = "get_files",
        python_callable = _get_files,
        op_kwargs={
            "filepath":"/opt/airflow/dags/data",
        }
    )
```

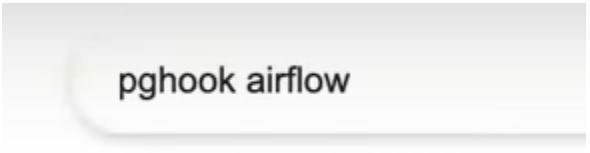
```
[2022-11-22, 16:47:21 UTC] {logging_mixin.py:117} INFO - 5 files found in /opt/***/dags/data
```





```python
#1
def _get_files(filepath: str) -> List[str]:
    """
    Description: This function is responsible for listing the files in a directory
    """

    all_files = []
    for root, dirs, files in os.walk(filepath):
        files = glob.glob(os.path.join(root, "*.json"))
        for f in files:
            all_files.append(os.path.abspath(f))

    num_files = len(all_files)
    print(f"{num_files} files found in {filepath}")

    return all_files


#2 ได้file จาก task #1แล้ว ไม่ต้อง all_files = get_files(filepath)
# แต่ดึงจาก context

def _process(**context):
    ti = context["ti"]
    # Get list of files from filepath
    all_files = ti.xcom_pull(task_ids = "get_files", key = "return_value")
    #all_files = get_files(filepath)
```

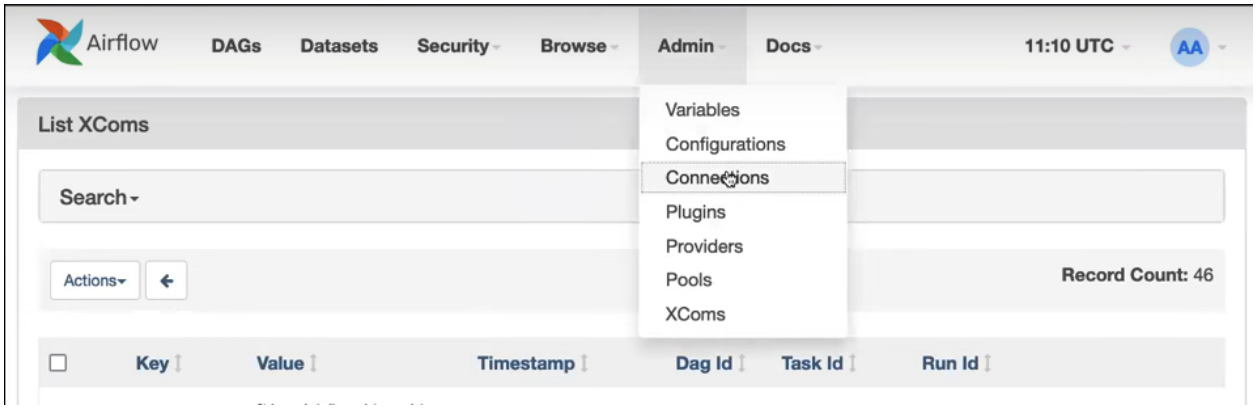
```python
  #Importance in "ON CONFLICT (id) DO NOTHING "
  #If insert actor same id code is do nothing

  # print(insert_statement)
  cur.execute(insert_statement)
```



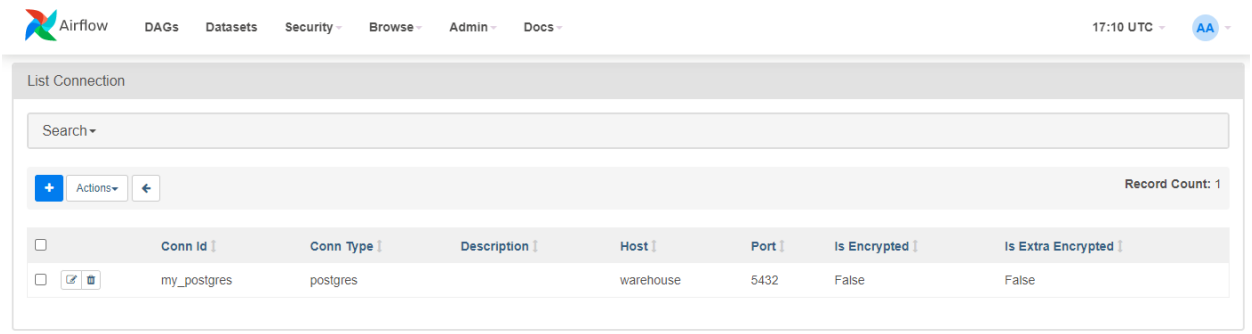
05-creating-and-scheduling-data-pipelines > dags > etl.py

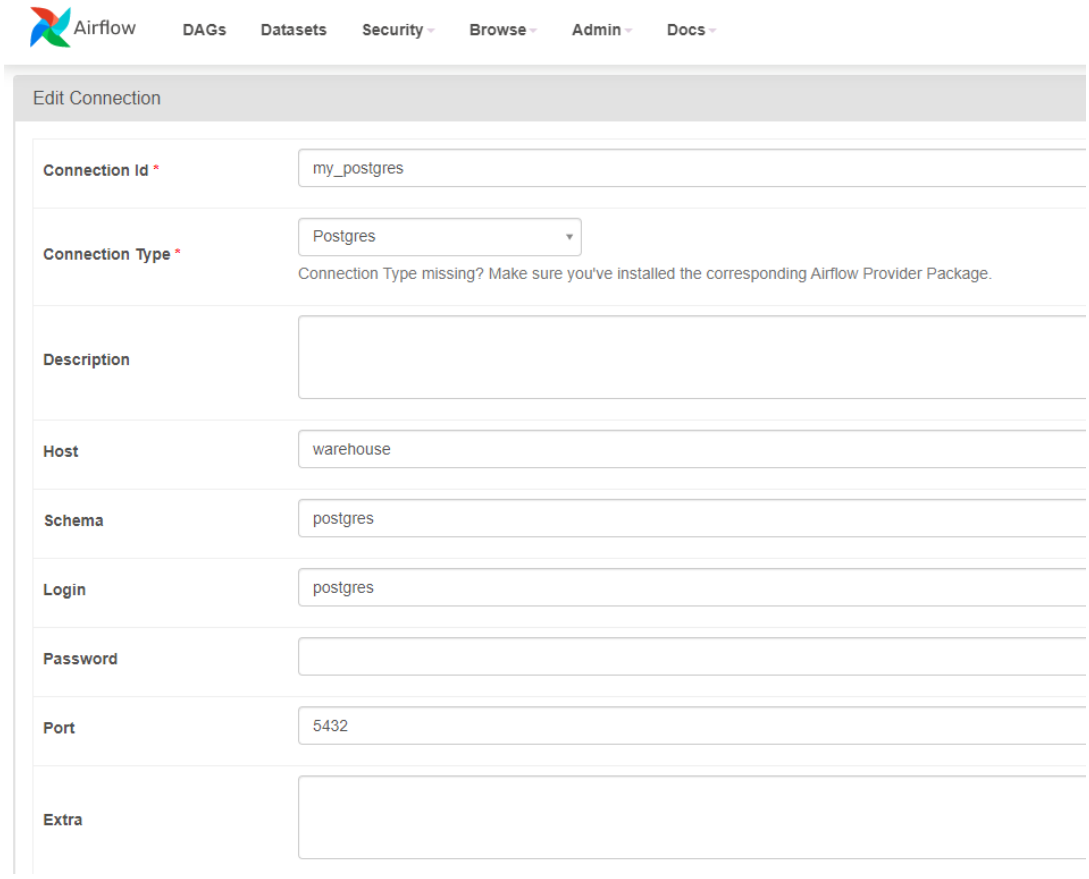
```python
import glob
import os
from typing import List

from airflow import DAG
from airflow.utils import timezone
from airflow.operators.python import PythonOperator
from airflow.providers.postgres.hooks.postgres import PostgresHook

```

```python
from airflow.providers.postgres.hooks.postgres import PostgresHook
```

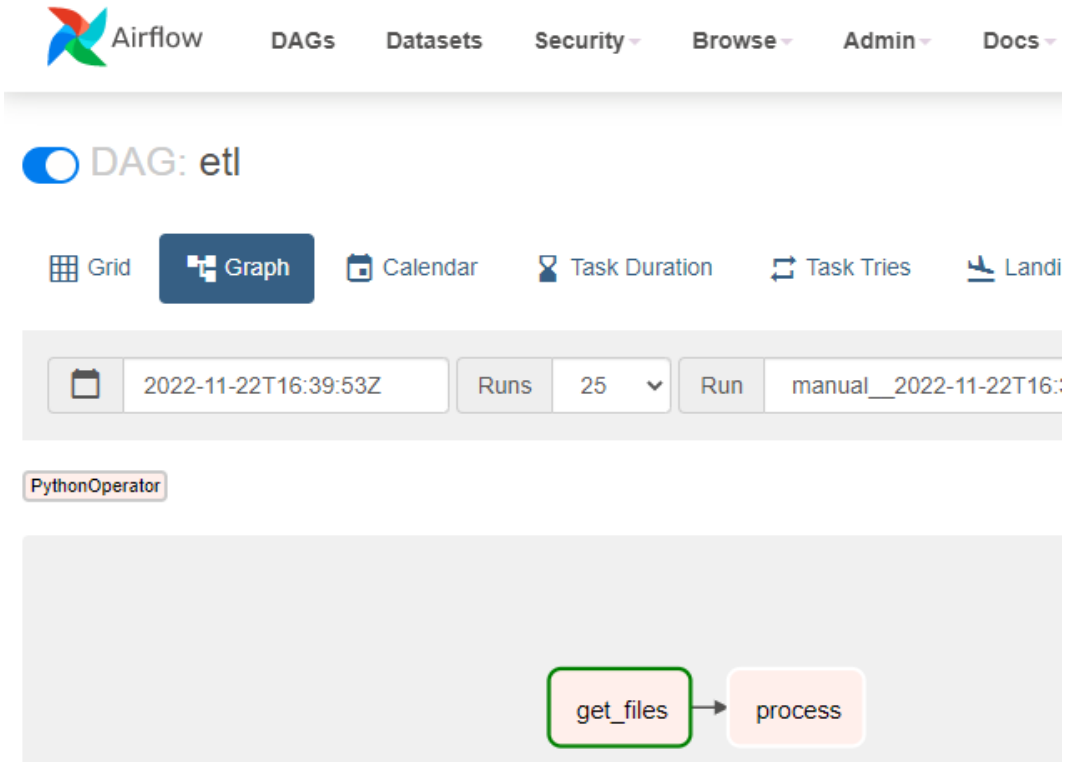


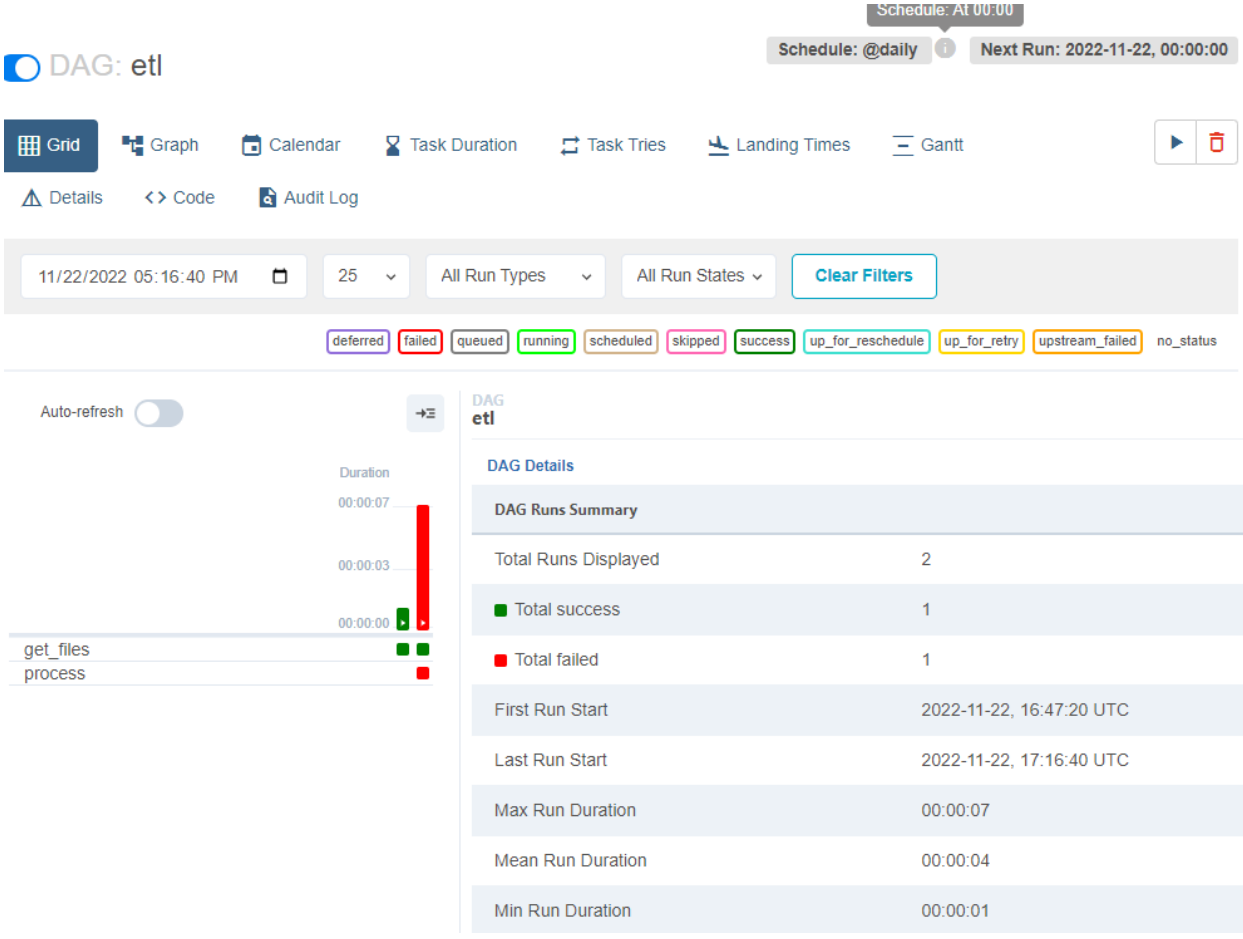

```python
def _process(**context):
    PostgresHook(postgres_conn_id ="my_postgres")

    ti = context["ti"]
    # Get list of files from filepath
    all_files = ti.xcom_pull(task_ids = "get_files", key = "return_value")
    #all_files = get_files(filepath)
```

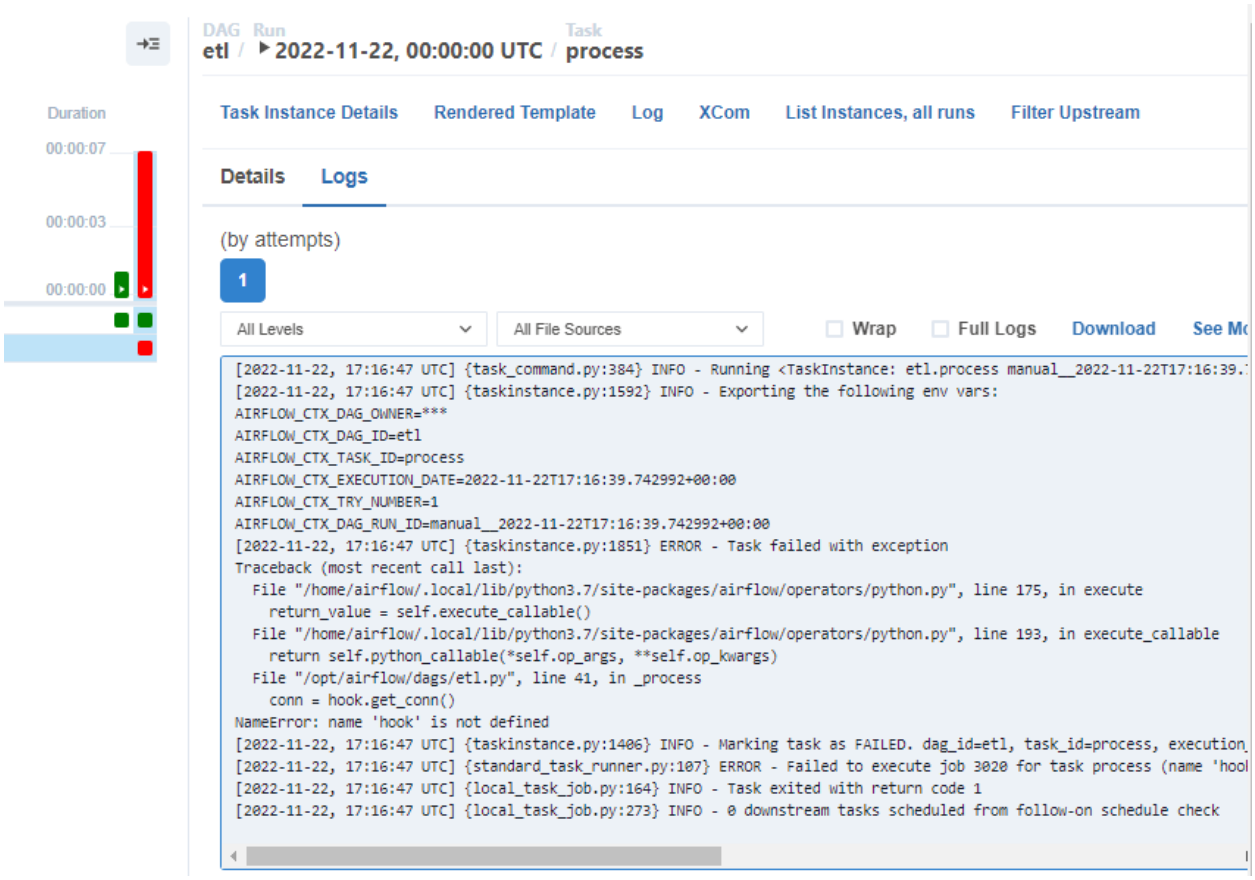





```
#4
    get_files >> process
```

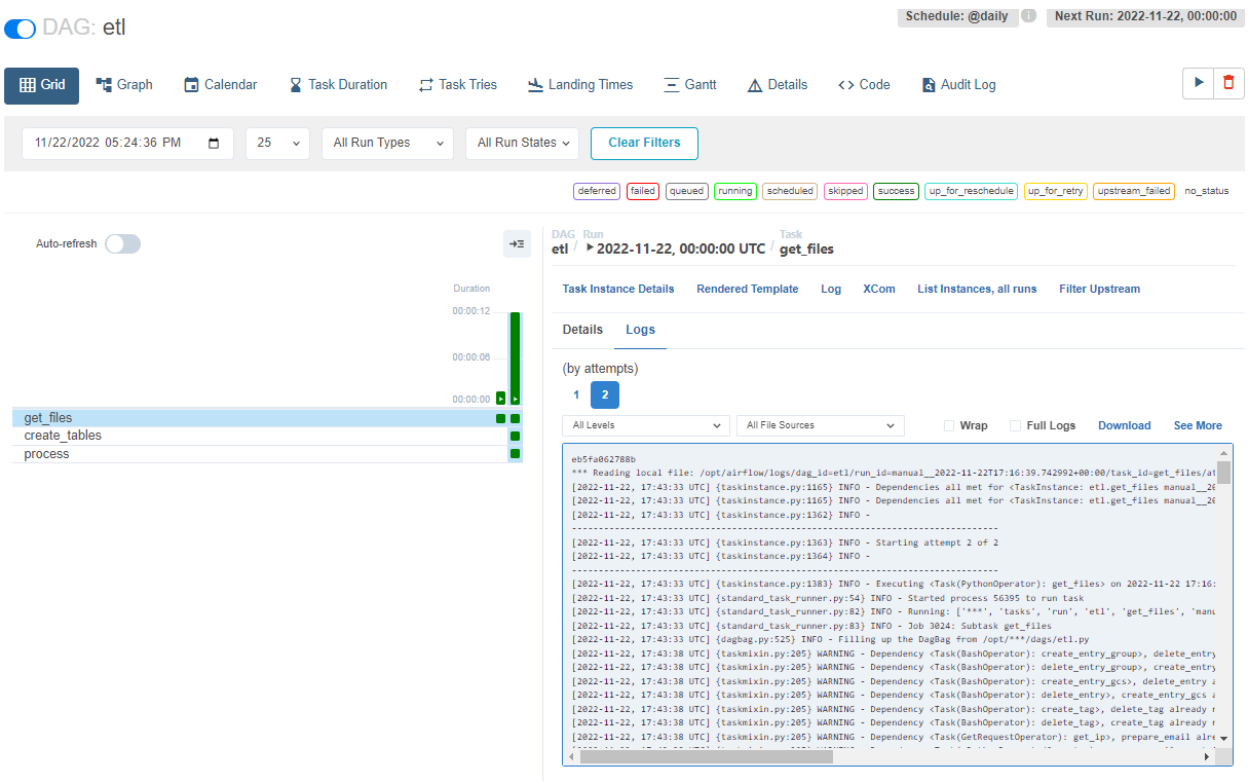

Airflow
DAGs
Datasets
Security
Browse
Admin
Docs
DAG: etl
Grid
Graph
Calendar
Task Duration
Task Tries
Landi
2022-11-22T16:39:53Z
Runs
25
Run
manual__2022-11-22T16:
PythonOperator
get_files
process


Schedule: At 00:00
Schedule: @daily
Next Run: 2022-11-22, 00:00:00
DAG: etl
Grid
Graph
Calendar
Task Duration
Task Tries
Landing Times
Gantt
Details
Code
Audit Log
11/22/2022 05:16:40 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
00:00:07
00:00:03
00:00:00
get_files
process
DAG
etl
DAG Details
DAG Runs Summary
Total Runs Displayed 2
Total success 1
Total failed 1
First Run Start 2022-11-22, 16:47:20 UTC
Last Run Start 2022-11-22, 17:16:40 UTC
Max Run Duration 00:00:07
Mean Run Duration 00:00:04
Min Run Duration 00:00:01




```
psycopg2.errors.UndefinedTable: relation "actors" does not exist
LINE 2:                     INSERT INTO actors (
                                        ^


[2022-11-22, 17:24:57 UTC] {taskinstance.py:1406} INFO - Marking task as FAILED. dag_id=etl, task_id=process, execution_da
[2022-11-22, 17:24:57 UTC] {standard_task_runner.py:107} ERROR - Failed to execute job 3023 for task process (relation "ac
LINE 2:                     INSERT INTO actors (
                                        ^
```

```python
#version2 แยกงานเป็นแต่ละ task
# สร้าง task create table
def _create_tables():
    hook = PostgresHook(postgres_conn_id ="my_postgres")
    conn = hook.get_conn()
    cur = conn.cursor()

    #5 Create table
    #Create table actors
    table_create_actors = """
        CREATE TABLE IF NOT EXISTS actors (
            id int,
            login text,
            PRIMARY KEY(id)
        )
    """
    #Create table events
    table_create_events = """
        CREATE TABLE IF NOT EXISTS events (
            id text,
            type text,
            actor_id int,
            actor_url text,
            created_at text,
            PRIMARY KEY(id),
            CONSTRAINT fk_actor FOREIGN KEY(actor_id) REFERENCES actors(id)
        )
    """
    #Lis for table to create , order is importance, create main first
    create_table_queries = [
        table_create_actors,
        table_create_events,
    ]

    for query in create_table_queries:
        cur.execute(query)
        conn.commit()


create_tables = PythonOperator(
    task_id = "create_tables",
    python_callable = _create_tables,
)                                       get_files >> create_tables >> process
```


DAG: etl
Schedule: @daily
Next Run: 2022-11-22, 00:00:00
Grid
Graph
Calendar
Task Duration
Task Tries
Landing Times
Gantt
Details
Code
Audit Log
11/22/2022 05:24:36 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
00:00:12
00:00:06
00:00:00
get_files
create_tables
process
DAG Run
etl / 2022-11-22, 00:00:00 UTC / Task
get_files
Task Instance Details
Rendered Template
Log
XCom
List Instances, all runs
Filter Upstream
Details
Logs
(by attempts)
1
2
All Levels
All File Sources
Wrap
Full Logs
Download
See More
eb5fa062788b
*** Reading local file: /opt/airflow/logs/dag_id=etl/run_id=manual__2022-11-22T17:16:39.742992+00:00/task_id=get_files/at
[2022-11-22, 17:43:33 UTC] {taskinstance.py:1165} INFO - Dependencies all met for <TaskInstance: etl.get_files manual__2
[2022-11-22, 17:43:33 UTC] {taskinstance.py:1165} INFO - Dependencies all met for <TaskInstance: etl.get_files manual__2
[2022-11-22, 17:43:33 UTC] {taskinstance.py:1362} INFO -
[2022-11-22, 17:43:33 UTC] {taskinstance.py:1363} INFO - Starting attempt 2 of 2
[2022-11-22, 17:43:33 UTC] {taskinstance.py:1364} INFO -
[2022-11-22, 17:43:33 UTC] {taskinstance.py:1383} INFO - Executing <Task(PythonOperator): get_files> on 2022-11-22 17:16:
[2022-11-22, 17:43:33 UTC] {standard_task_runner.py:54} INFO - Started process 56395 to run task
[2022-11-22, 17:43:33 UTC] {standard_task_runner.py:82} INFO - Running: ['***', 'tasks', 'run', 'etl', 'get_files', 'manu
[2022-11-22, 17:43:33 UTC] {standard_task_runner.py:83} INFO - Job 3024: Subtask get_files
[2022-11-22, 17:43:33 UTC] {dagbag.py:525} INFO - Filling up the DagBag from /opt/***/dags/etl.py
[2022-11-22, 17:43:38 UTC] {taskmixin.py:205} WARNING - Dependency <Task(BashOperator): create_entry_group>, delete_entry
[2022-11-22, 17:43:38 UTC] {taskmixin.py:205} WARNING - Dependency <Task(BashOperator): delete_entry_group>, create_entry
[2022-11-22, 17:43:38 UTC] {taskmixin.py:205} WARNING - Dependency <Task(BashOperator): create_entry_gcs>, delete_entry
[2022-11-22, 17:43:38 UTC] {taskmixin.py:205} WARNING - Dependency <Task(BashOperator): delete_entry>, create_entry_gcs
[2022-11-22, 17:43:38 UTC] {taskmixin.py:205} WARNING - Dependency <Task(BashOperator): create_tag>, delete_tag already
[2022-11-22, 17:43:38 UTC] {taskmixin.py:205} WARNING - Dependency <Task(BashOperator): delete_tag>, create_tag already
[2022-11-22, 17:43:38 UTC] {taskmixin.py:205} WARNING - Dependency <Task(GetRequestOperator): get_ip>, prepare_email alre

Language: English

Adminer 4.8.1

DB: postgres
Schema: public

SQL command Import
Export Create table

select actors
select events

PostgreSQL » warehouse » postgres » public » Select: actors

Select: actors

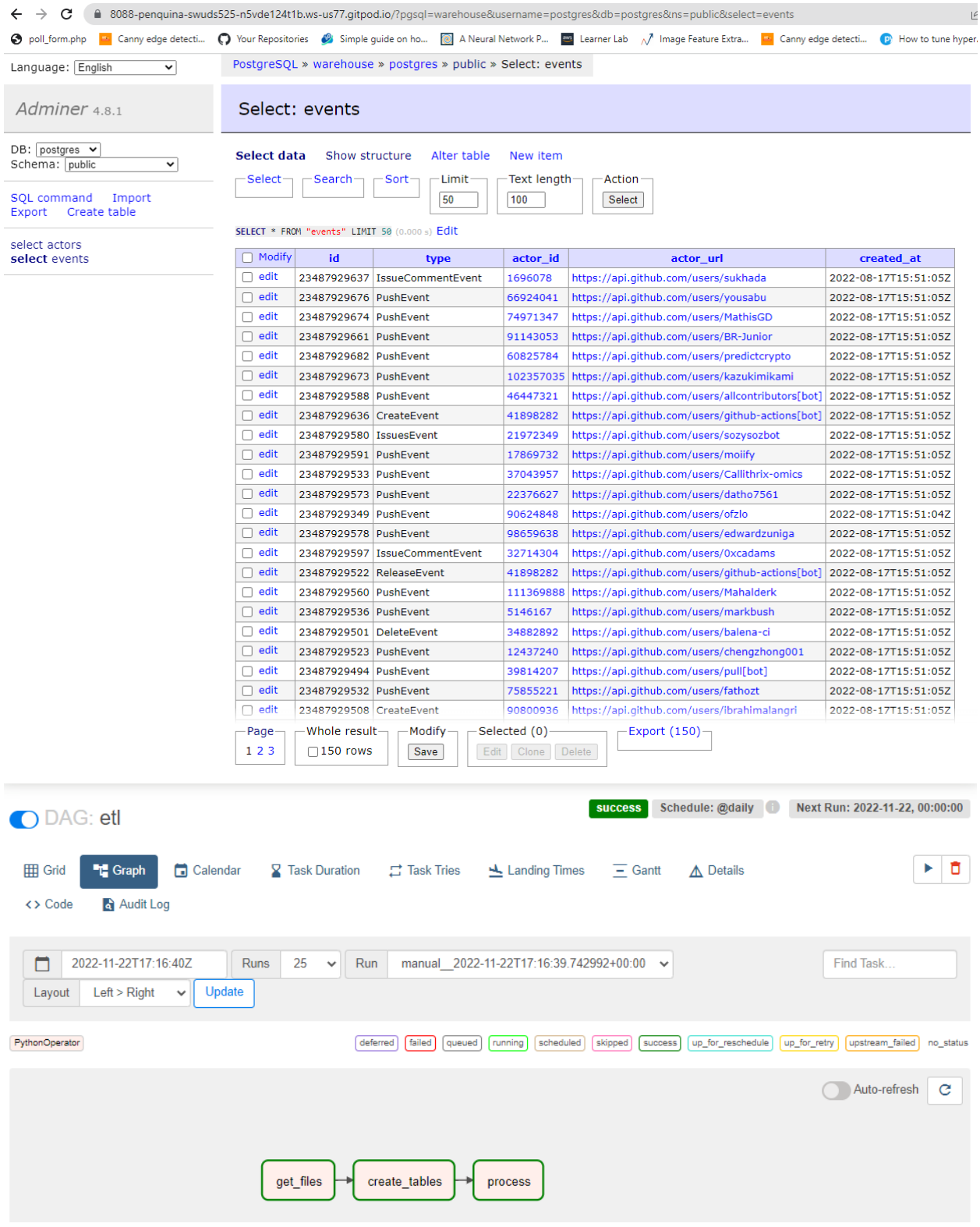
Select data Show structure Alter table New item

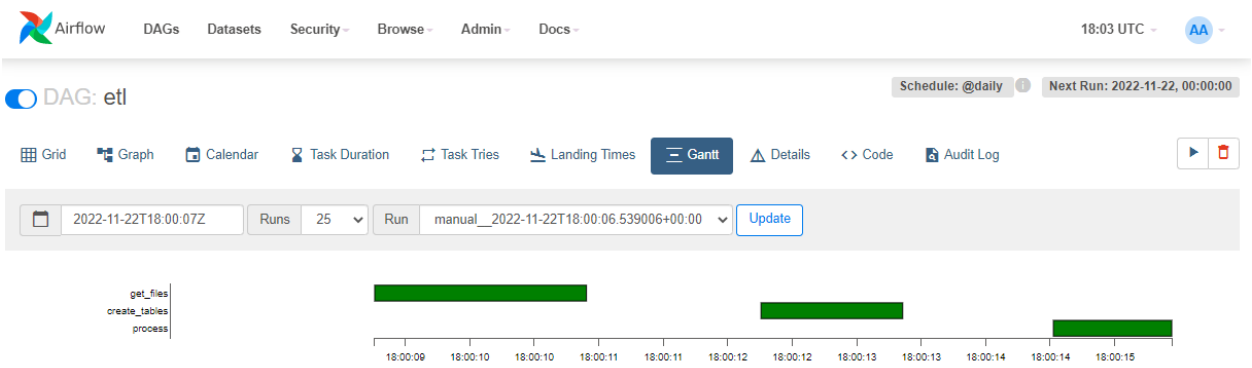
Select | Search | Sort | Limit: 50 | Text length: 100 | Action: Select

```
SELECT * FROM "actors" LIMIT 50 (0.000 s) Edit
```

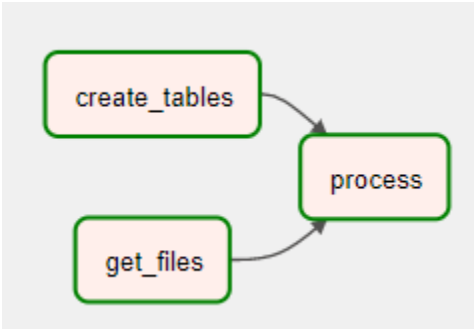
Modify	id	login
edit	1696078	sukhada
edit	66924041	yousabu
edit	74971347	MathisGD
edit	91143053	BR-Junior
edit	60825784	predictcrypto
edit	102357035	kazukimikami
edit	46447321	allcontributors[bot]
edit	41898282	github-actions[bot]
edit	21972349	sozysozbot
edit	17869732	moiify
edit	37043957	Callithrix-omics
edit	22376627	datho7561
edit	90624848	ofzlo
edit	98659638	edwardzuniga
edit	32714304	0xcadams
edit	111369888	Mahalderk
edit	5146167	markbush
edit	34882892	balena-ci
edit	12437240	chengzhong001
edit	39814207	pull[bot]
edit	75855221	fathozt
edit	90800936	ibrahimalangri
edit	102809970	InonAngel98

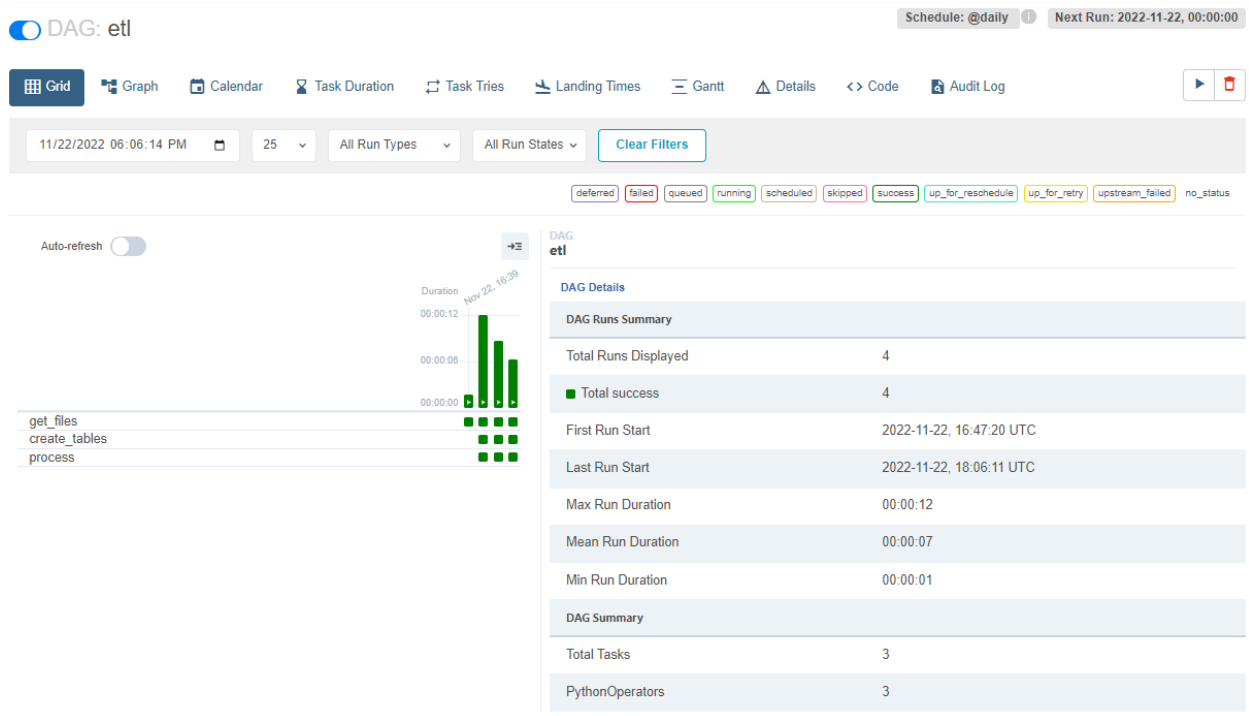
Page: 1 2 3 | Whole result: 127 rows | Modify: Save | Selected (0): Edit Clone Delete | Export (127)

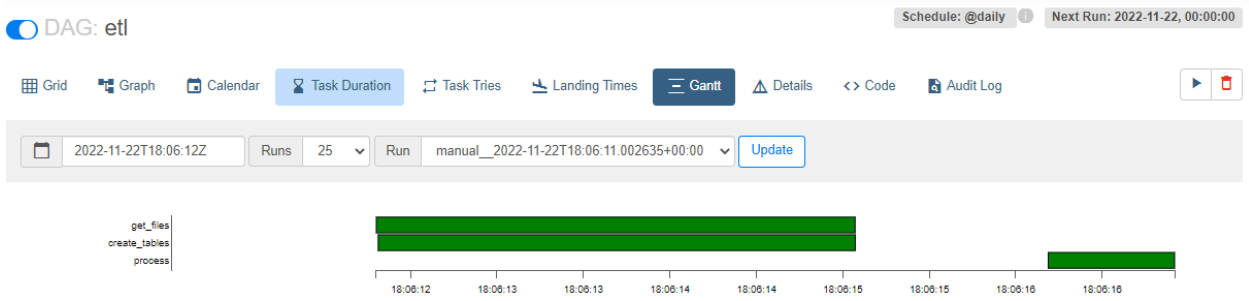

8088-penquina-swuds525-n5vde124t1b.ws-us77.gitpod.io/?pgsql=warehouse&username=postgres&db=postgres&ns=public&select=events
Language: English
PostgreSQL » warehouse » postgres » public » Select: events
Adminer 4.8.1
DB: postgres
Schema: public
SQL command Import
Export Create table
select actors
select events
Select: events
Select data Show structure Alter table New item
Select Search Sort Limit 50 Text length 100 Action Select
SELECT * FROM "events" LIMIT 50 (0.000 s) Edit
Modify id type actor_id actor_url created_at
edit 23487929637 IssueCommentEvent 1696078 https://api.github.com/users/sukhada 2022-08-17T15:51:05Z
edit 23487929676 PushEvent 66924041 https://api.github.com/users/yousabu 2022-08-17T15:51:05Z
edit 23487929674 PushEvent 74971347 https://api.github.com/users/MathisGD 2022-08-17T15:51:05Z
edit 23487929661 PushEvent 91143053 https://api.github.com/users/BR-Junior 2022-08-17T15:51:05Z
edit 23487929682 PushEvent 60825784 https://api.github.com/users/predictcrypto 2022-08-17T15:51:05Z
edit 23487929673 PushEvent 102357035 https://api.github.com/users/kazukimikami 2022-08-17T15:51:05Z
edit 23487929588 PushEvent 46447321 https://api.github.com/users/allcontributors[bot] 2022-08-17T15:51:05Z
edit 23487929636 CreateEvent 41898282 https://api.github.com/users/github-actions[bot] 2022-08-17T15:51:05Z
edit 23487929580 IssuesEvent 21972349 https://api.github.com/users/sozysozbot 2022-08-17T15:51:05Z
edit 23487929591 PushEvent 17869732 https://api.github.com/users/moiify 2022-08-17T15:51:05Z
edit 23487929533 PushEvent 37043957 https://api.github.com/users/Callithrix-omics 2022-08-17T15:51:05Z
edit 23487929573 PushEvent 22376627 https://api.github.com/users/datho7561 2022-08-17T15:51:05Z
edit 23487929349 PushEvent 90624848 https://api.github.com/users/ofzlo 2022-08-17T15:51:04Z
edit 23487929578 PushEvent 98659638 https://api.github.com/users/edwardzuniga 2022-08-17T15:51:05Z
edit 23487929597 IssueCommentEvent 32714304 https://api.github.com/users/0xcadams 2022-08-17T15:51:05Z
edit 23487929522 ReleaseEvent 41898282 https://api.github.com/users/github-actions[bot] 2022-08-17T15:51:05Z
edit 23487929560 PushEvent 111369888 https://api.github.com/users/Mahalderk 2022-08-17T15:51:05Z
edit 23487929536 PushEvent 5146167 https://api.github.com/users/markbush 2022-08-17T15:51:05Z
edit 23487929501 DeleteEvent 34882892 https://api.github.com/users/balena-ci 2022-08-17T15:51:05Z
edit 23487929523 PushEvent 12437240 https://api.github.com/users/chengzhong001 2022-08-17T15:51:05Z
edit 23487929494 PushEvent 39814207 https://api.github.com/users/pull[bot] 2022-08-17T15:51:05Z
edit 23487929532 PushEvent 75855221 https://api.github.com/users/fathozt 2022-08-17T15:51:05Z
edit 23487929508 CreateEvent 90800936 https://api.github.com/users/ibrahimalangri 2022-08-17T15:51:05Z
Page 1 2 3
Whole result 150 rows
Modify Save
Selected (0) Edit Clone Delete
Export (150)
DAG: etl
success
Schedule: @daily
Next Run: 2022-11-22, 00:00:00
Grid Graph Calendar Task Duration Task Tries Landing Times Gantt Details
Code Audit Log
2022-11-22T17:16:40Z
Runs 25
Run manual__2022-11-22T17:16:39.742992+00:00
Find Task...
Layout Left > Right Update
PythonOperator
deferred failed queued running scheduled skipped success up_for_reschedule up_for_retry upstream_failed no_status
Auto-refresh
get_files
create_tables
process




```
#test รันแบบขนาน
  [get_files , create_tables] >> process
```




DAG: etl
Schedule: @daily
Next Run: 2022-11-22, 00:00:00
Grid
Graph
Calendar
Task Duration
Task Tries
Landing Times
Gantt
Details
Code
Audit Log
11/22/2022 06:06:14 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 22, 16:39
00:00:12
00:00:06
00:00:00
get_files
create_tables
process
DAG
etl
DAG Details
DAG Runs Summary
Total Runs Displayed 4
Total success 4
First Run Start 2022-11-22, 16:47:20 UTC
Last Run Start 2022-11-22, 18:06:11 UTC
Max Run Duration 00:00:12
Mean Run Duration 00:00:07
Min Run Duration 00:00:01
DAG Summary
Total Tasks 3
PythonOperators 3


DAG: etl
Schedule: @daily
Next Run: 2022-11-22, 00:00:00
Grid
Graph
Calendar
Task Duration
Task Tries
Landing Times
Gantt
Details
Code
Audit Log
2022-11-22T18:06:12Z
Runs
25
Run
manual__2022-11-22T18:06:11.002635+00:00
Update
get_files
create_tables
process
18:06:12
18:06:13
18:06:13
18:06:14
18:06:14
18:06:15
18:06:15
18:06:16
18:06:16