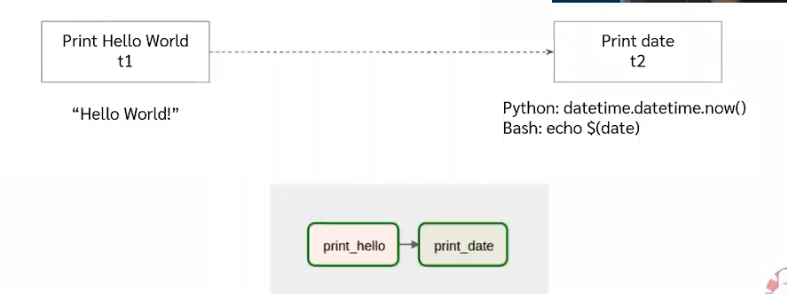
Exercise 1



import datetime

from airflow.models import DAG

from airflow.operators.python import PythonOperator

from airflow.operators.bash import BashOperator

from airflow.utils.dates import days\_ago

default\_args = {

    'owner': 'datath',

}

def my\_function(something: str):

    print(something)

with DAG(

    "exercise1\_simple\_dag",

    start\_date=days\_ago(1),

    schedule\_interval=None,

    tags=["exercise"]

) as dag:

    # Exercise1: Simple Pipeline - Hello World Airflow!

    # ใน exercise นี้จะได้รู้จักกับ PythonOperator (และ BashOperator)

    # และลองเขียน task dependencies

    t1 = PythonOperator(

        task\_id="print\_hello",

        python\_callable=my\_function,

        op\_kwargs={"something": "Hello World!"},

    )

    t2 = BashOperator(

        task\_id="print\_date",

        bash\_command="echo $(date)",

    )

    #TODO: ใส่ task dependencies

     t1>>t2

การใช้ Cloud Shell รันคำสั่ง เพื่อจัดการ file ใน GCG

t1 Print Hello World

r2 Print date

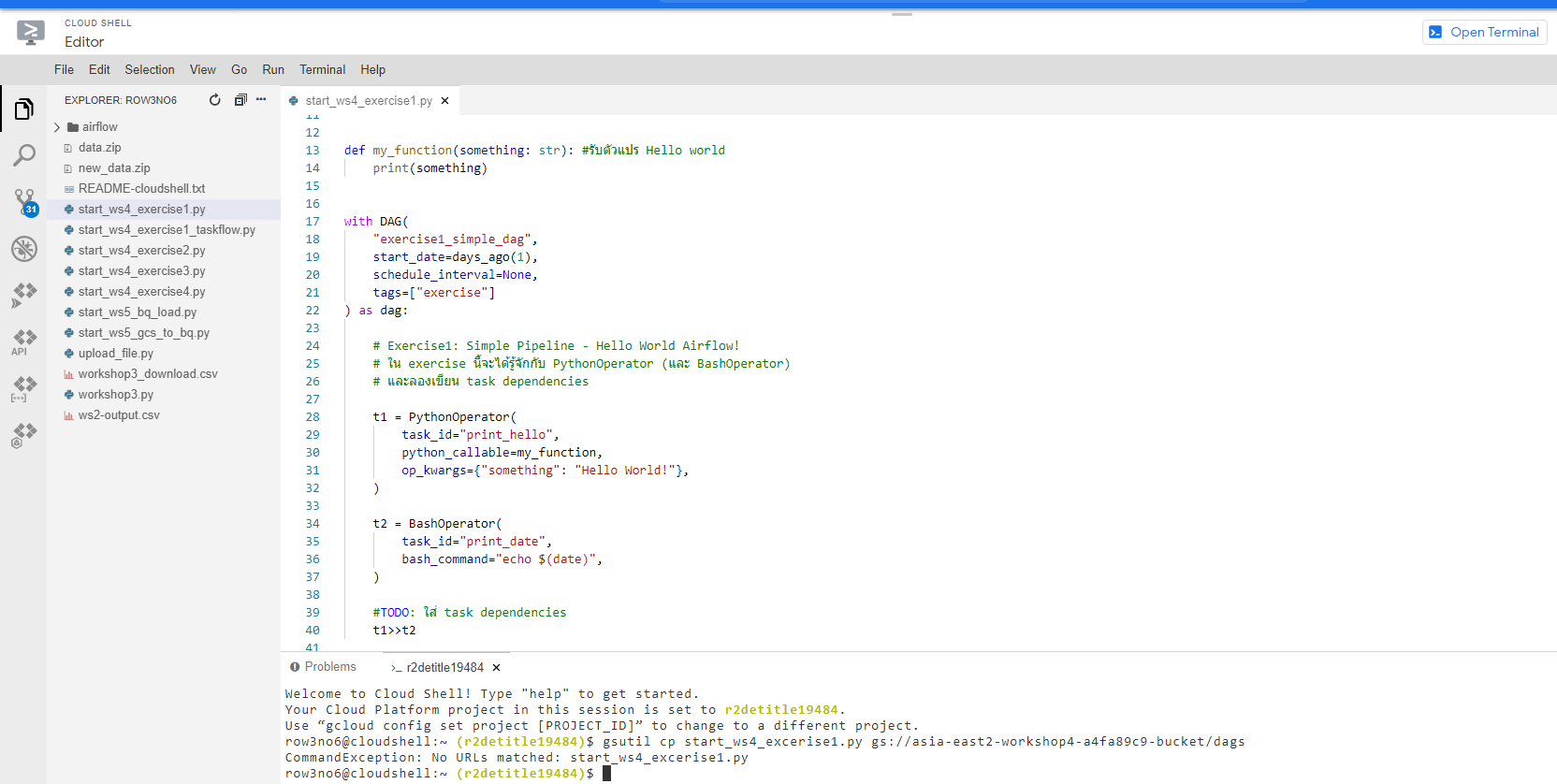
python : datatime.datime.now()

Copy from clud shell to bucket เพื่อมันใช้งาน air flow ได้

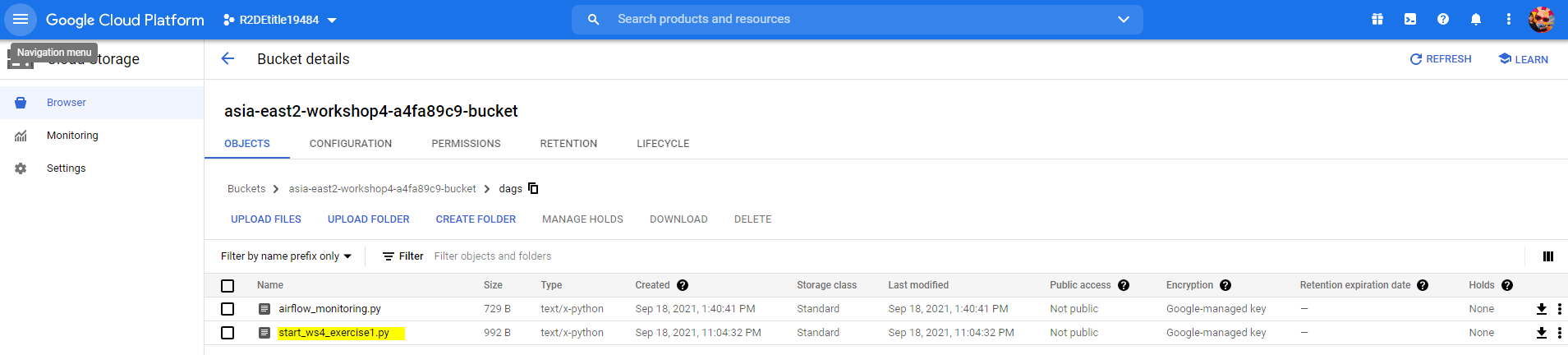
gsutil cp start\_ws4\_exercise1.py gs://asia-east2-workshop4-a4fa89c9-bucket/dags

Copying file://start\_ws4\_exercise1.py [Content-Type=text/x-python]...

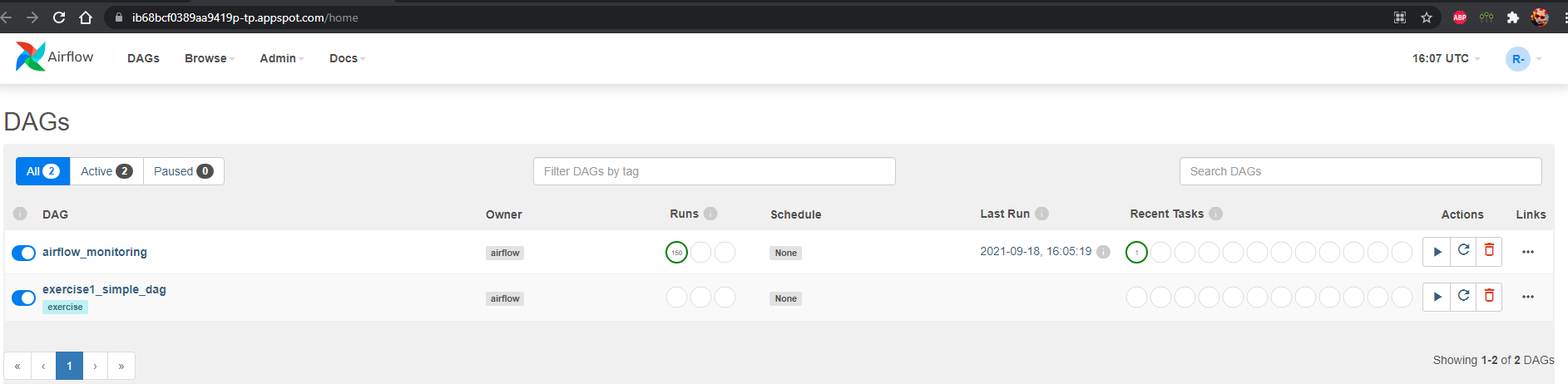
/ [1 files][ 992.0 B/ 992.0 B]

Operation completed over 1 objects/992.0 B.

ตรวจสอบการเชื่อม

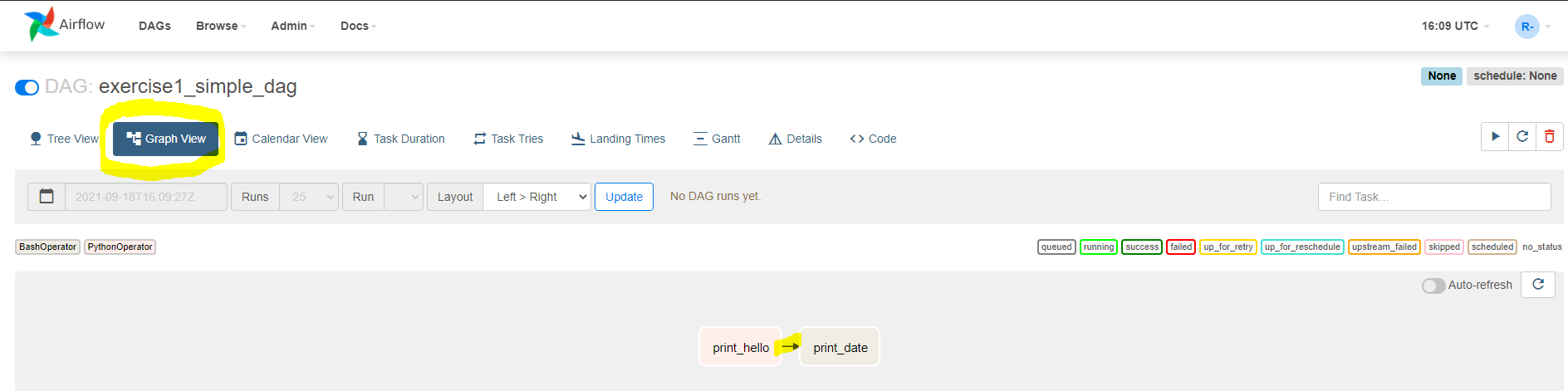


<https://ib68bcf0389aa9419p-tp.appspot.com/home>

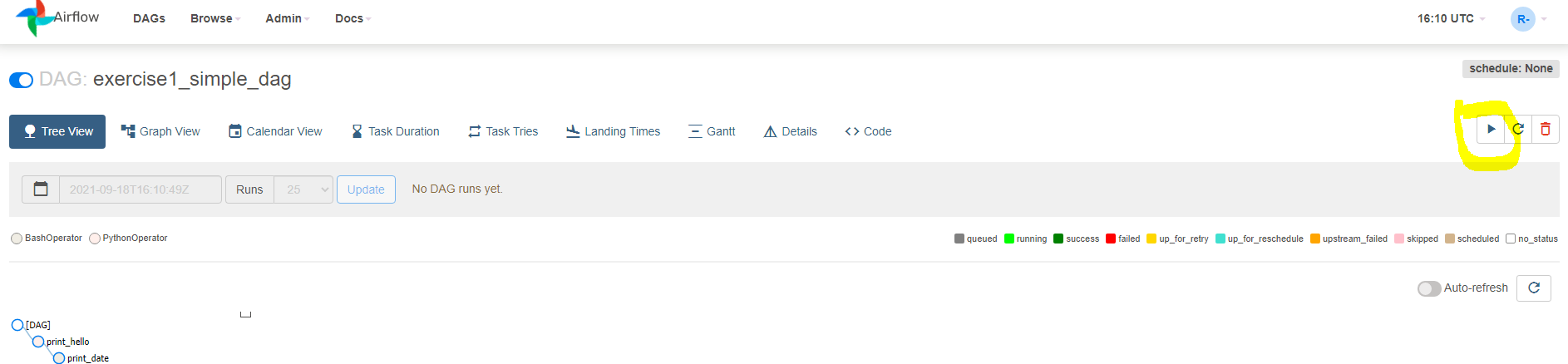


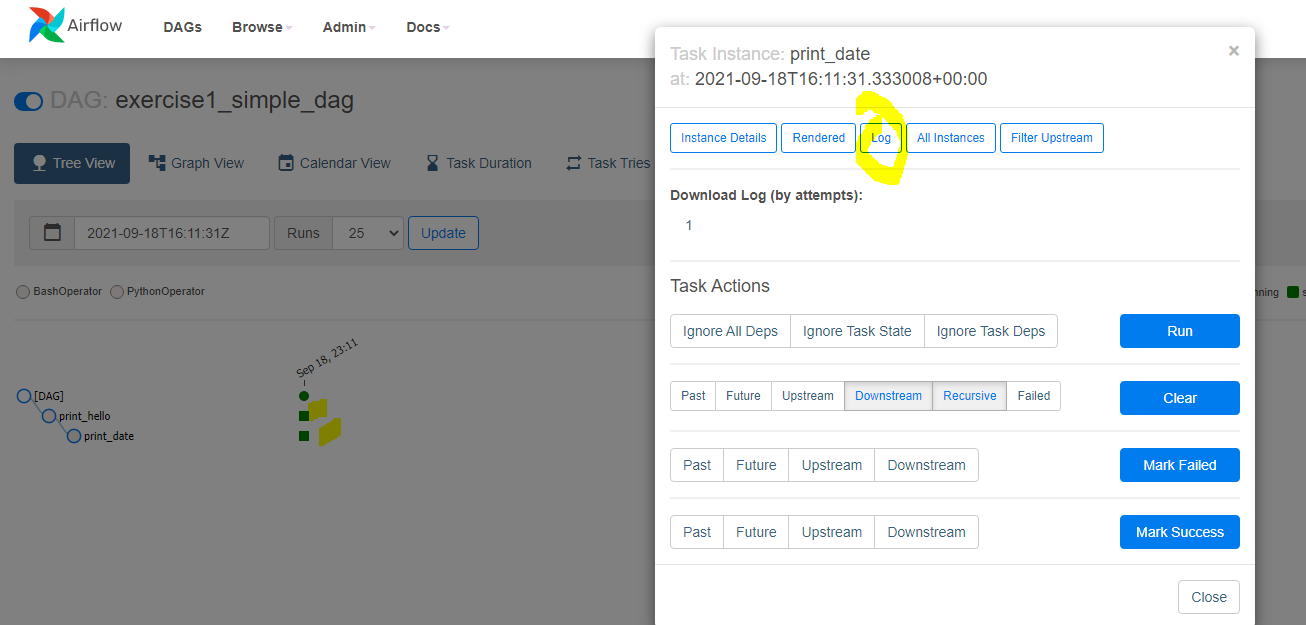
<https://ib68bcf0389aa9419p-tp.appspot.com/tree?dag_id=exercise1_simple_dag>

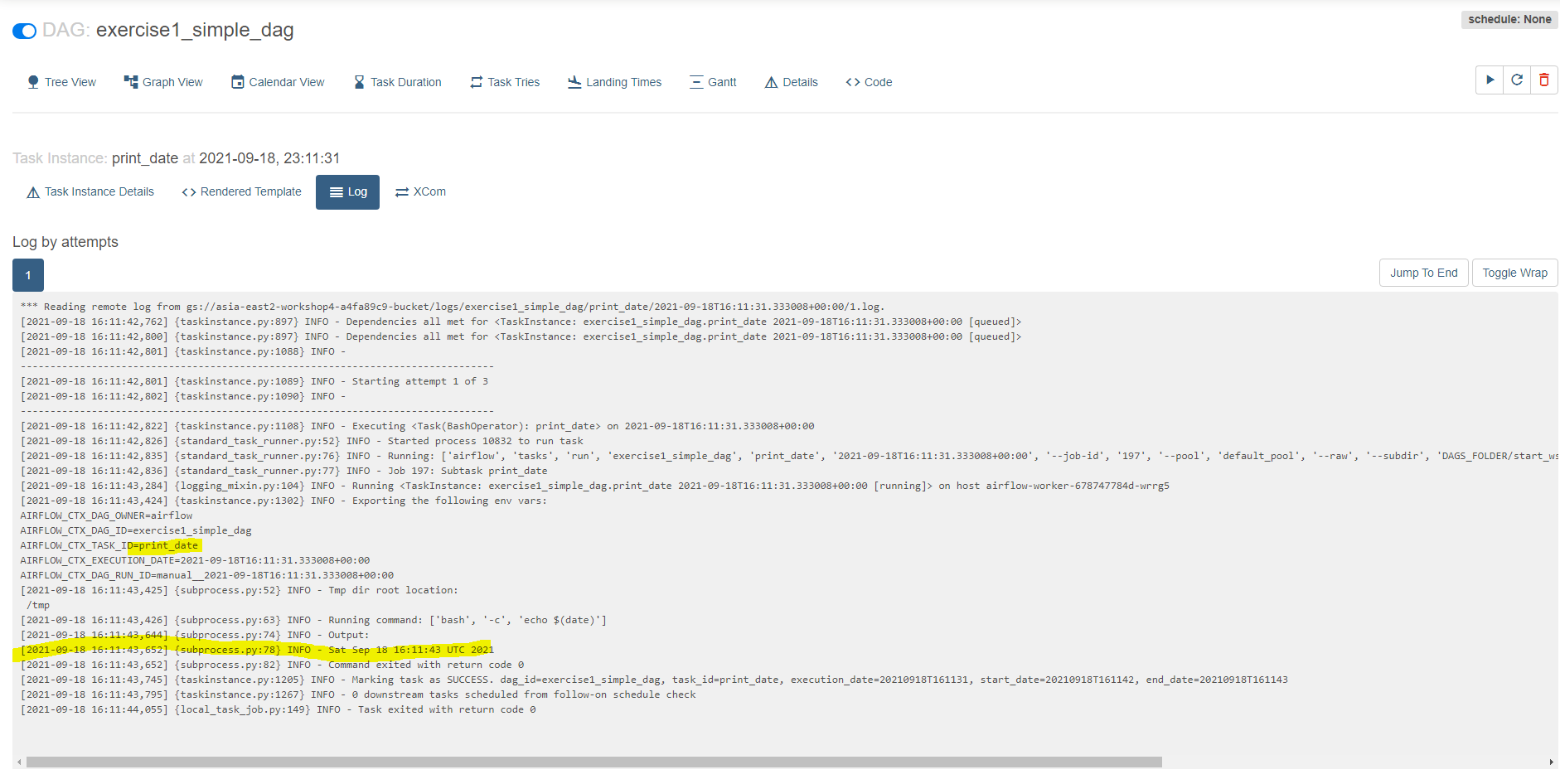
กดเข้าไปดูใน ex.1 simple dag

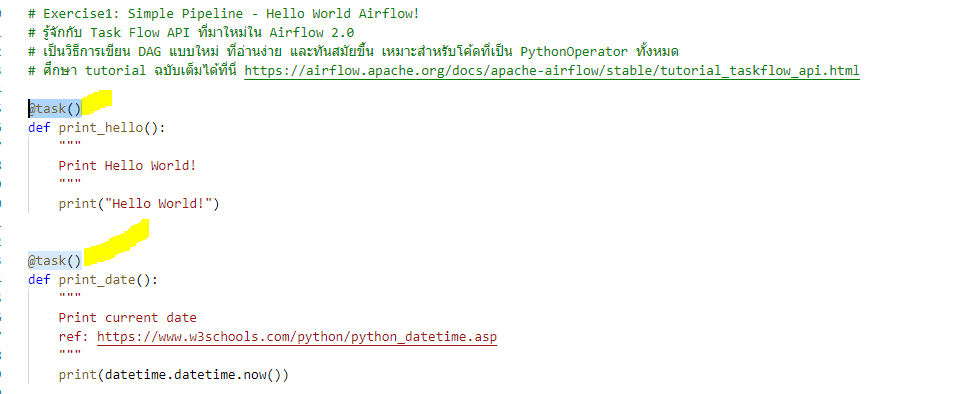


กด manual run Trigger DAG







Task flow exercise1โดยการใช้ @task()

import datetime

from airflow.decorators import dag, task

from airflow.utils.dates import days\_ago

default\_args = {

    'owner': 'datath',

}

# Exercise1: Simple Pipeline - Hello World Airflow!

# รู้จักกับ Task Flow API ที่มาใหม่ใน Airflow 2.0

# เป็นวิธีการเขียน DAG แบบใหม่ ที่อ่านง่าย และทันสมัยขึ้น เหมาะสำหรับโค้ดที่เป็น PythonOperator ทั้งหมด

# ศึกษา tutorial ฉบับเต็มได้ที่นี่ https://airflow.apache.org/docs/apache-airflow/stable/tutorial\_taskflow\_api.html

@task()

def print\_hello():

    """

    Print Hello World!

    """

    print("Hello World!")

@task()

def print\_date():

    """

    Print current date

    ref: https://www.w3schools.com/python/python\_datetime.asp

    """

    print(datetime.datetime.now())

@dag(default\_args=default\_args, schedule\_interval=None, start\_date=days\_ago(1), tags=['exercise'])

def exercise1\_taskflow\_dag():#เรียกใช้ฟังก์ชั่น DAG

    t1 = print\_hello()

    t2 = print\_date()

    # TODO: สร้าง task dependencies ที่นี่

    t1>>t2

exercise1\_dag = exercise1\_taskflow\_dag()

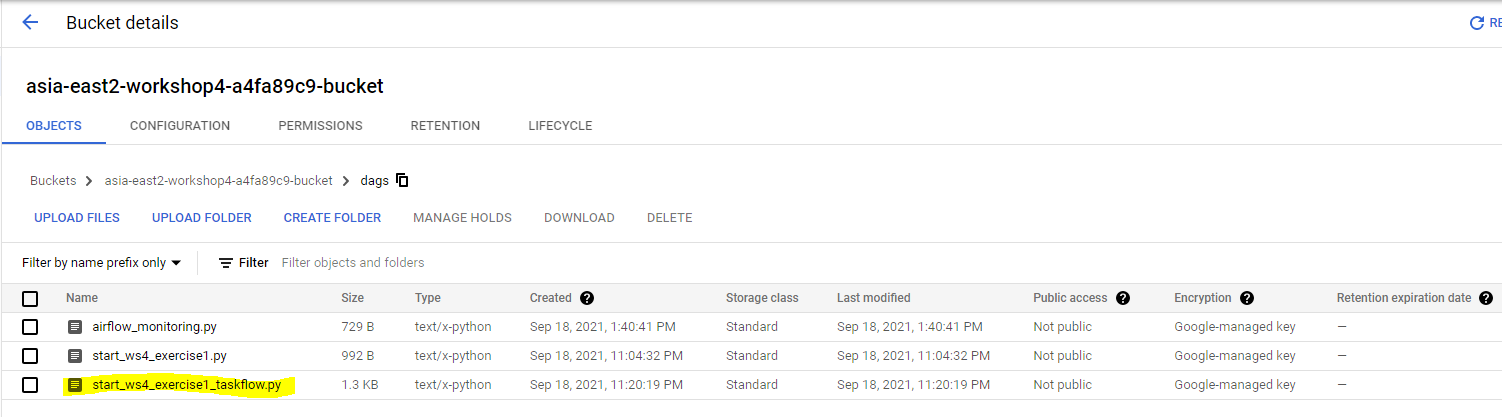
ใน cloud shell

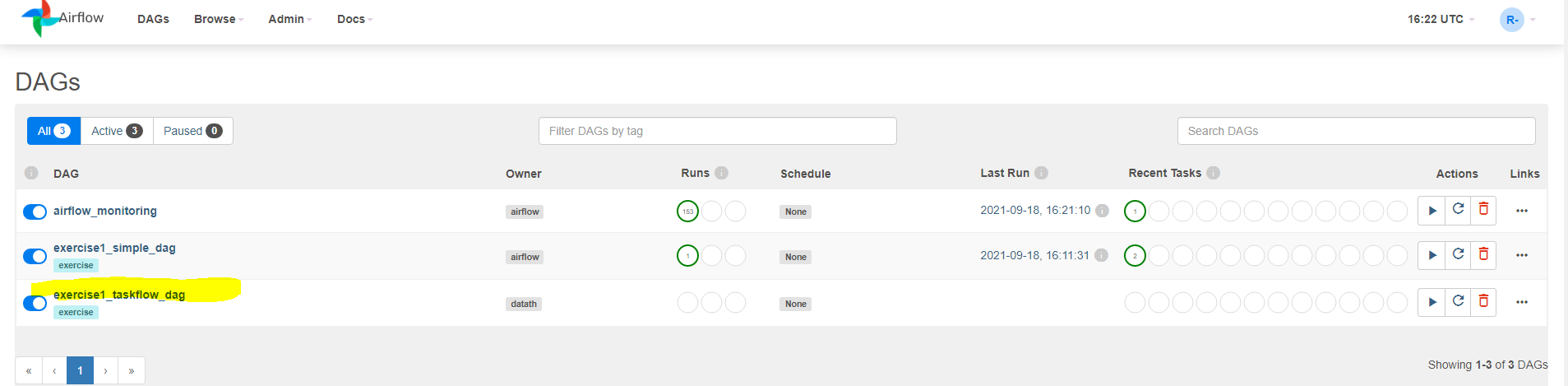
gsutil cp start\_ws4\_exercise1\_taskflow.py gs://asia-east2-workshop4-a4fa89c9-bucket/dags

Copying file://start\_ws4\_exercise1\_taskflow.py [Content-Type=text/x-python]...

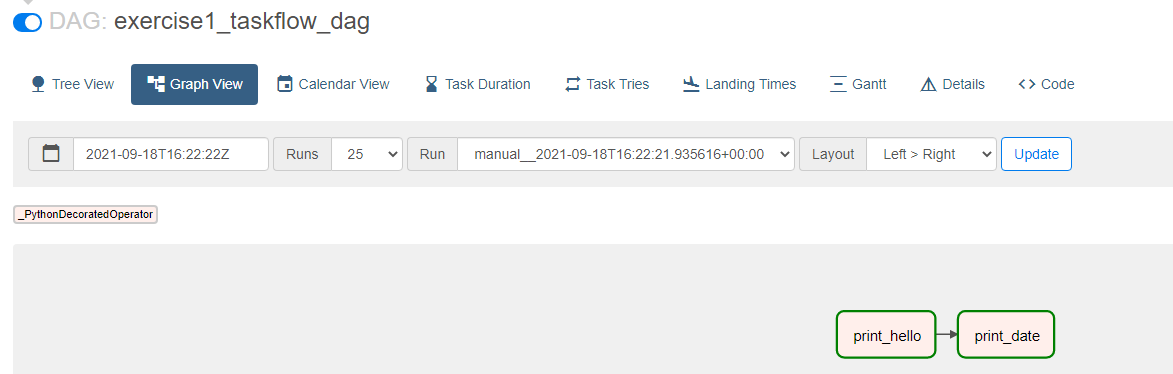
/ [1 files][ 1.3 KiB/ 1.3 KiB]

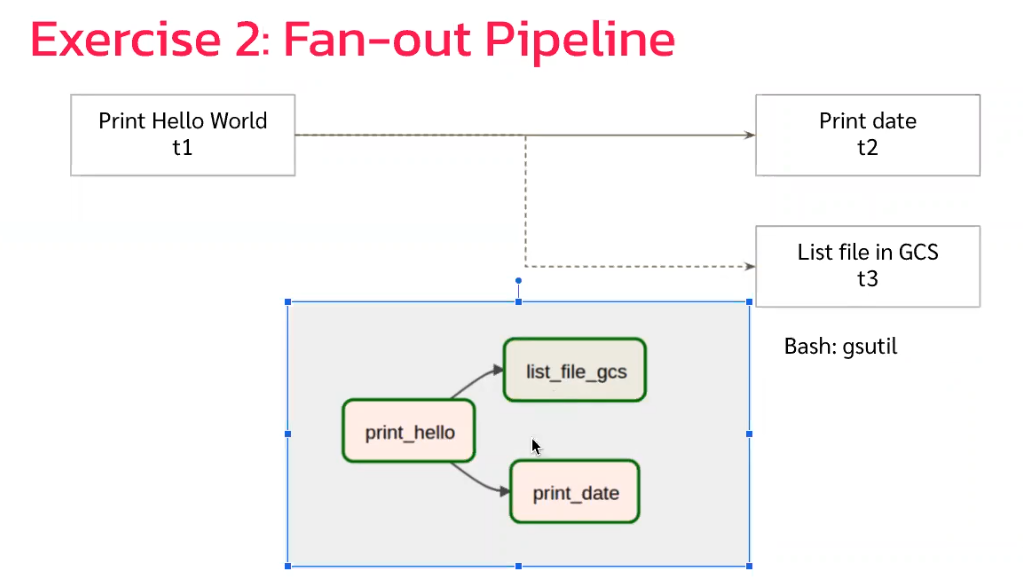
Operation completed over 1 objects/1.3 KiB.





ลองกด run





Exercise2 Fan-out Pipe line

import datetime

from airflow.models import DAG

from airflow.operators.python import PythonOperator

from airflow.operators.bash import BashOperator

from airflow.utils.dates import days\_ago

default\_args = {

    'owner': 'datath',

}

def my\_function(something: str):

    print(something)

with DAG(

    "exercise2\_fan\_out\_dag",

    start\_date=days\_ago(1),

    schedule\_interval=None,

    tags=["exercise"]

) as dag:

    # Exercise2: Fan-out Pipeline

    # ใน exercise นี้จะได้รู้จักกับการแยก pipeline ออกเพื่อให้ทำงานแบบ parallel พร้อมกันได้

    # และทดลองใช้คำสั่ง gsutil จาก BashOperator

    t1 = PythonOperator(

        task\_id="print\_hello",

        python\_callable=my\_function,

        op\_kwargs={"something": "Hello World!"},

    )

    t2 = BashOperator(

        task\_id="print\_date",

        bash\_command="echo $(date)",

    )

    #TODO: ใส่ task t3 สำหรับ list ไฟล์ใน GCS bucket ที่เป็น DAGs folder

    #hint: ใช้ BashOperator ร่วมกับ command gsutil

    t3 = BashOperator(

        task\_id="list\_file\_gsc",

        bash\_command="gsutil ls gs://asia-east2-workshop4-a4fa89c9-bucket/dags",

    )

    # TODO: ใส่ task dependencies ที่ทำให้รัน t3 พร้อมกับ t2 ได้

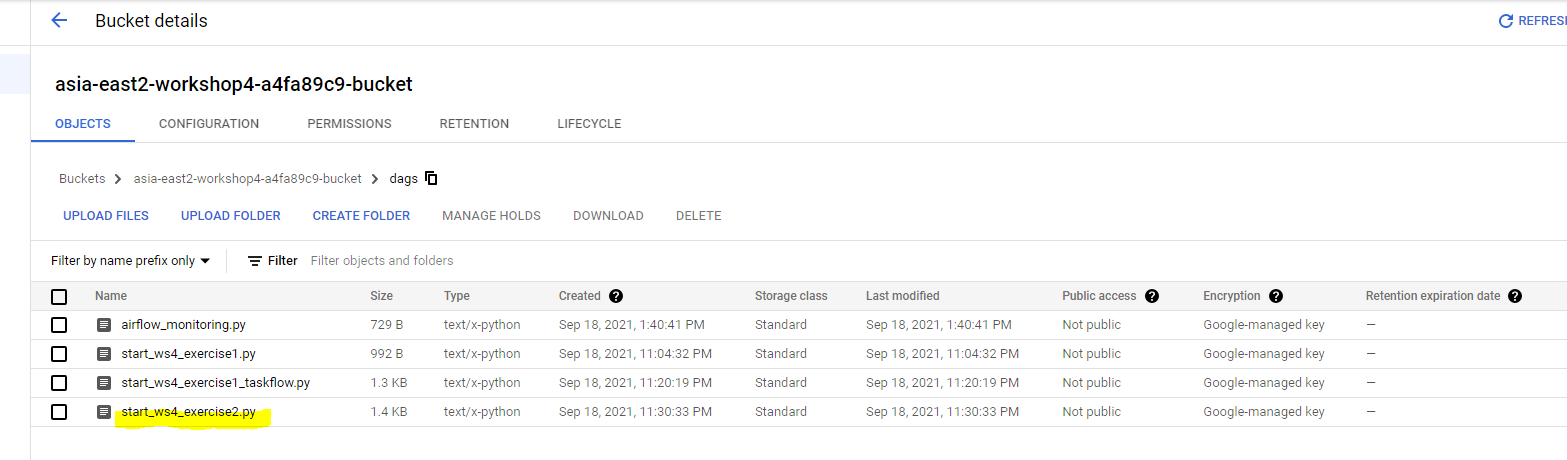
    t1>>[t2,t3]

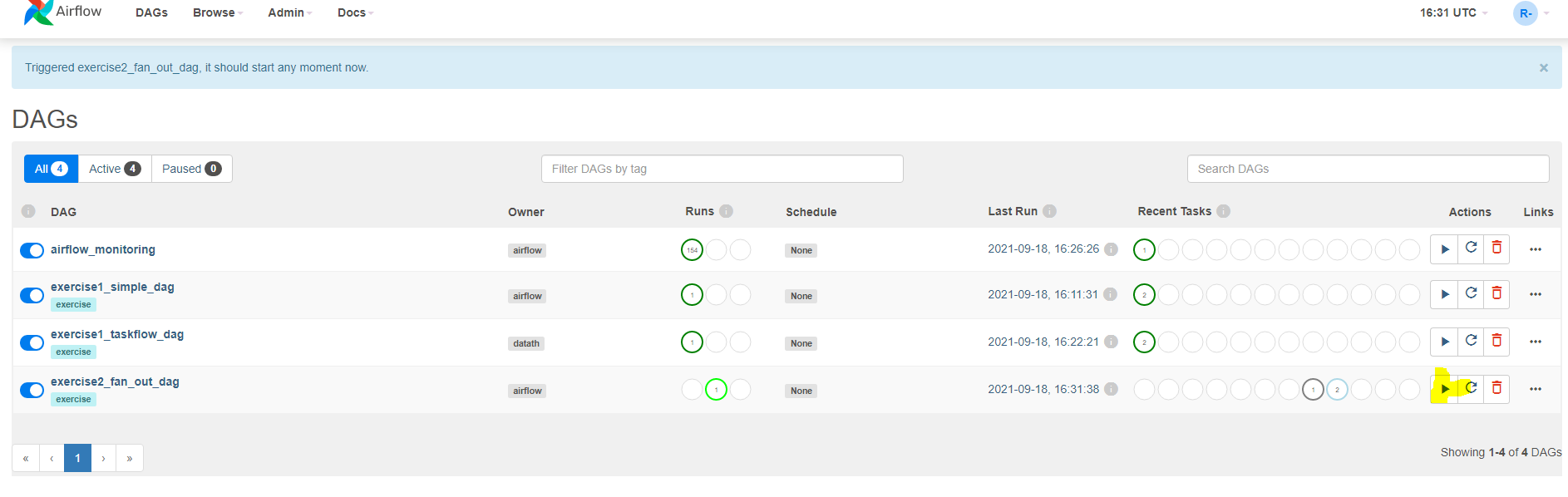
row3no6@cloudshell:~ (r2detitle19484)$ gsutil cp start\_ws4\_exercise2.py gs://asia-east2-workshop4-a4fa89c9-bucket/dags

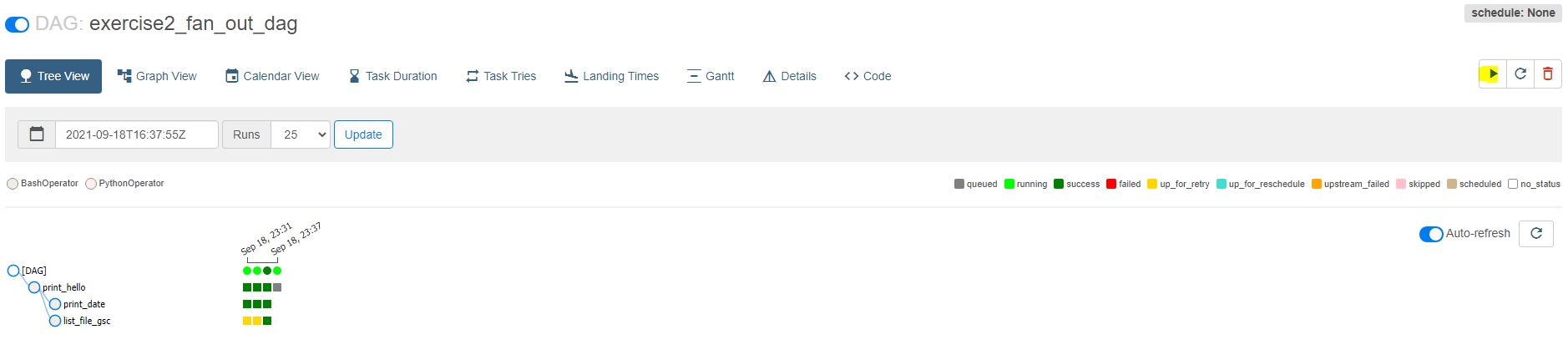
Copying file://start\_ws4\_exercise2.py [Content-Type=text/x-python]...

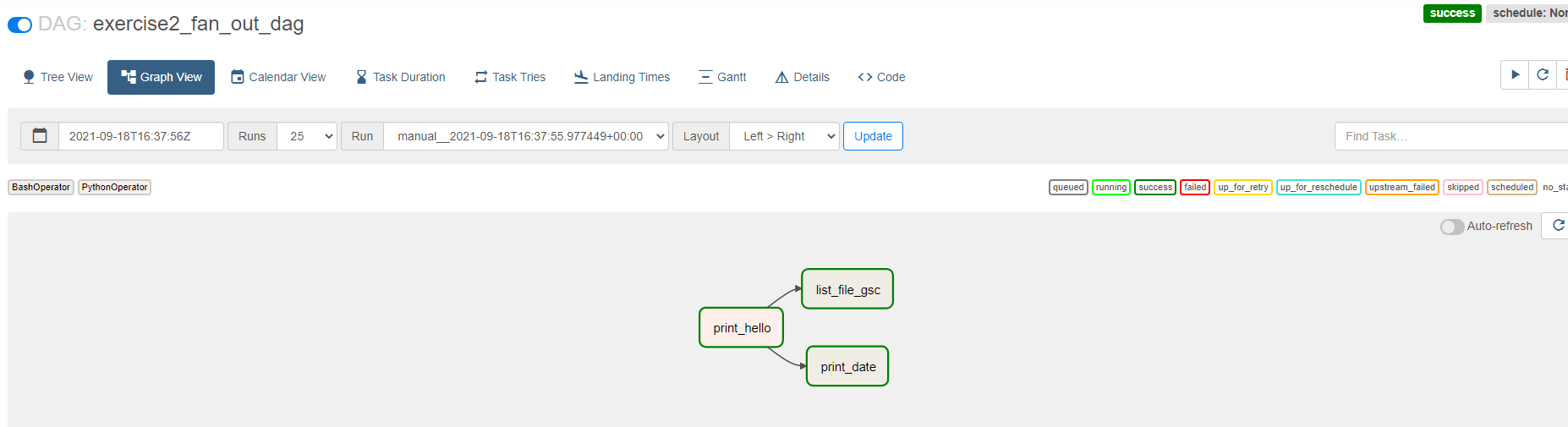
/ [1 files][ 1.4 KiB/ 1.4 KiB]

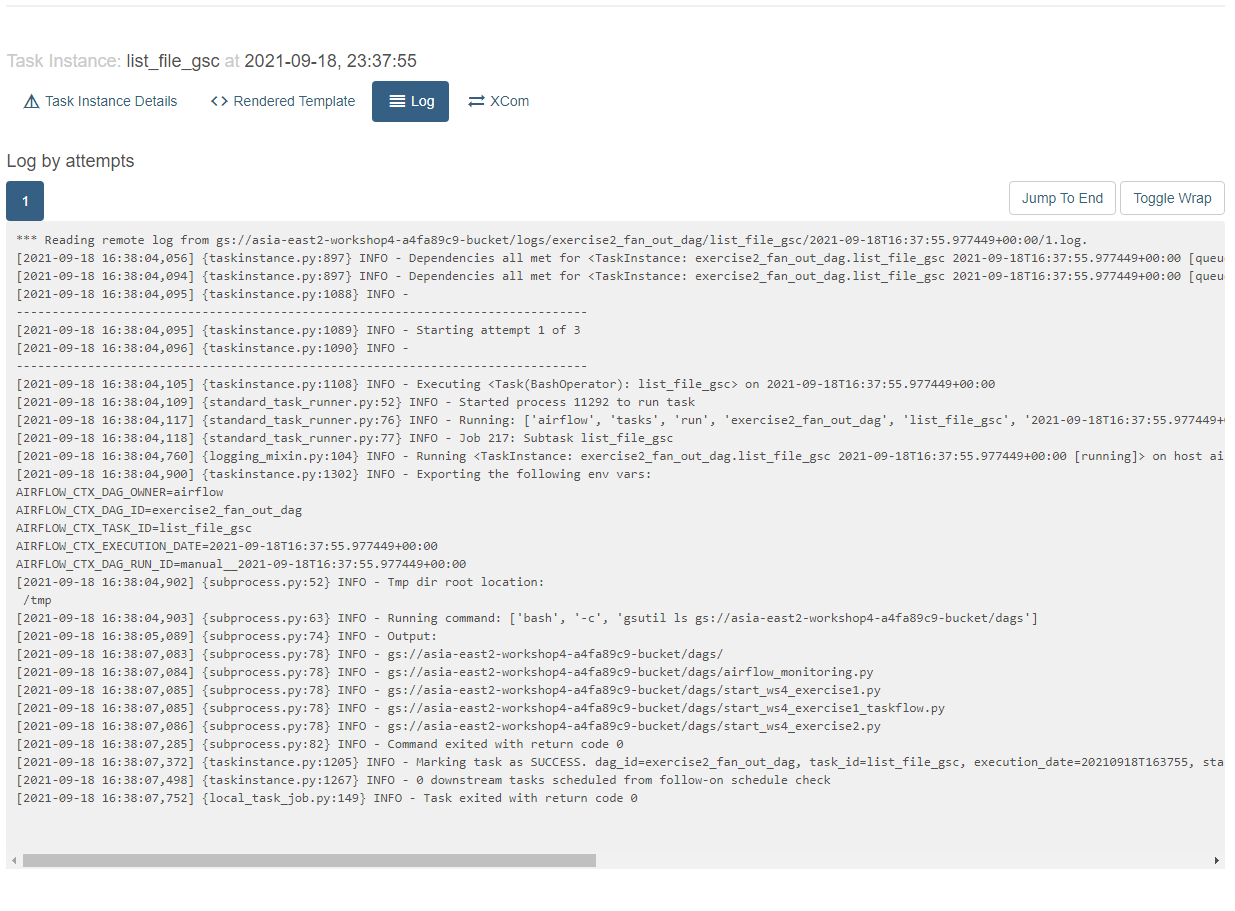
Operation completed over 1 objects/1.4 KiB.



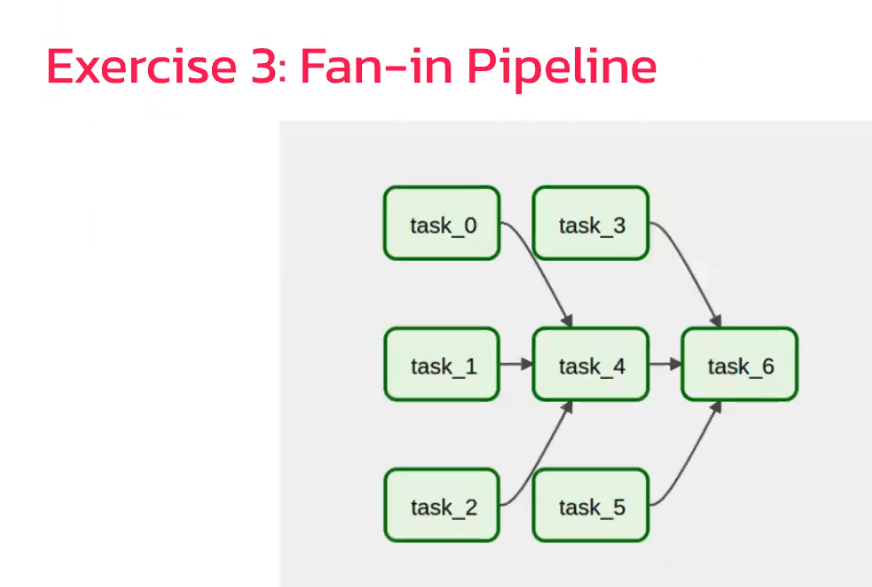








Exercise3 Fan-out Pipe line



from airflow.models import DAG

from airflow.operators.dummy import DummyOperator

from airflow.utils.dates import days\_ago

with DAG(

    "exercise3\_fan\_in\_dag",

    start\_date=days\_ago(1),

    schedule\_interval=None,

    tags=["exercise"]

) as dag:

    # Exercise3: Fan-in Pipeline

    # ใน exercise นี้จะได้รู้จักการเขียน task ใน pipeline ขั้นตอนเยอะขึ้น

    # ใช้ DummyOperator เป็น task จำลอง

    t0 = DummyOperator(task\_id="task\_0")

    t4 = DummyOperator(task\_id="task\_4")

    t0 >> t4

    # TODO: สร้าง DummyOperator เพื่อสร้าง dependency ที่ซับซ้อน

from airflow.models import DAG

from airflow.operators.dummy import DummyOperator

from airflow.utils.dates import days\_ago

with DAG(

    "exercise3\_fan\_in\_dag",

    start\_date=days\_ago(1),

    schedule\_interval=None,

    tags=["exercise"]

) as dag:

    # Exercise3: Fan-in Pipeline

    # ใน exercise นี้จะได้รู้จักการเขียน task ใน pipeline ขั้นตอนเยอะขึ้น

    # ใช้ DummyOperator เป็น task จำลอง

    t0 = DummyOperator(task\_id="task\_0")

    t1 = DummyOperator(task\_id="task\_1")

    t2 = DummyOperator(task\_id="task\_2")

    t3 = DummyOperator(task\_id="task\_3")

    t4 = DummyOperator(task\_id="task\_4")

    t5 = DummyOperator(task\_id="task\_5")

    t6 = DummyOperator(task\_id="task\_6")

    [t0,t1,t2] >> t4

    [t3,t4,t5] >> t6

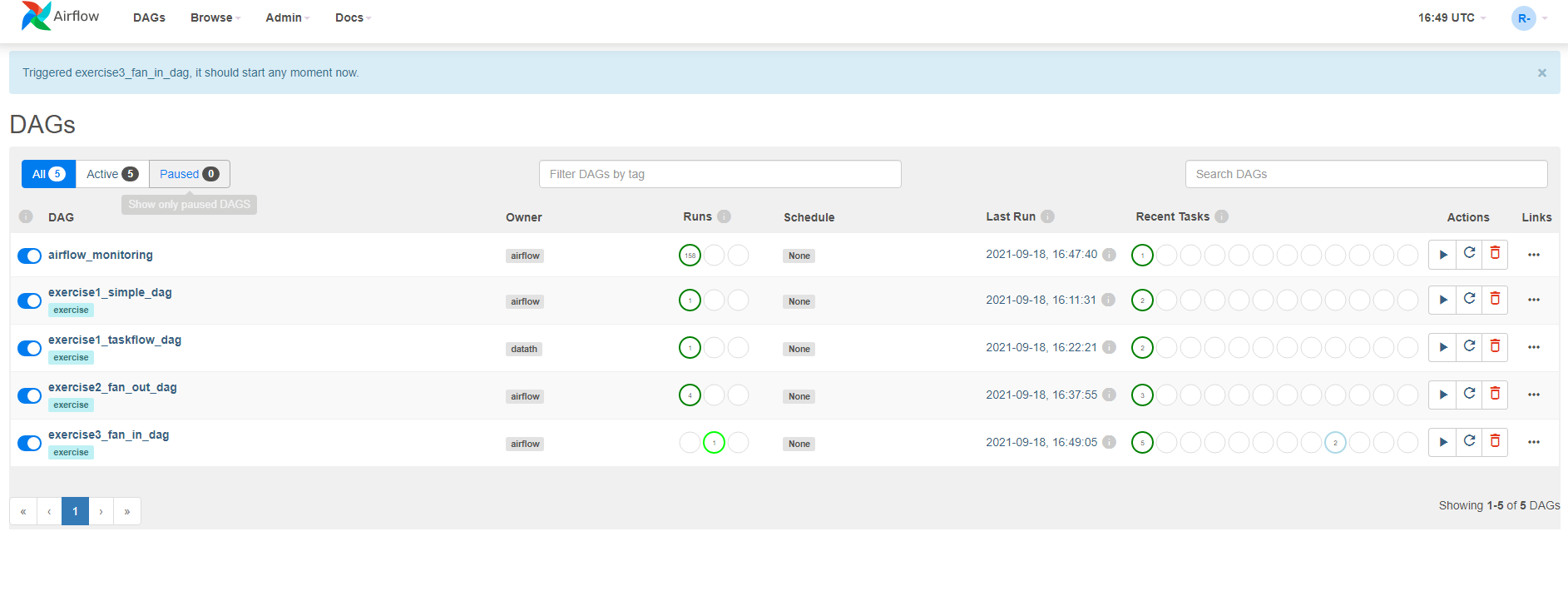
    # TODO: สร้าง DummyOperator เพื่อสร้าง dependency ที่ซับซ้อน

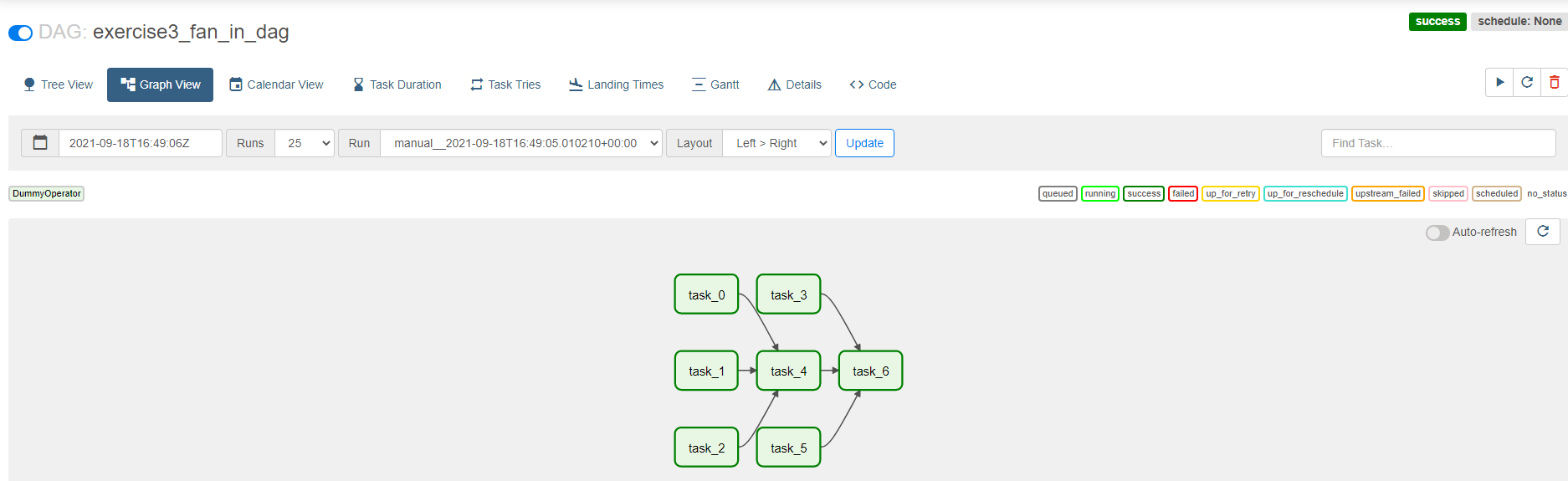
gsutil cp start\_ws4\_exercise3.py gs://asia-east2-workshop4-a4fa89c9-bucket/dags

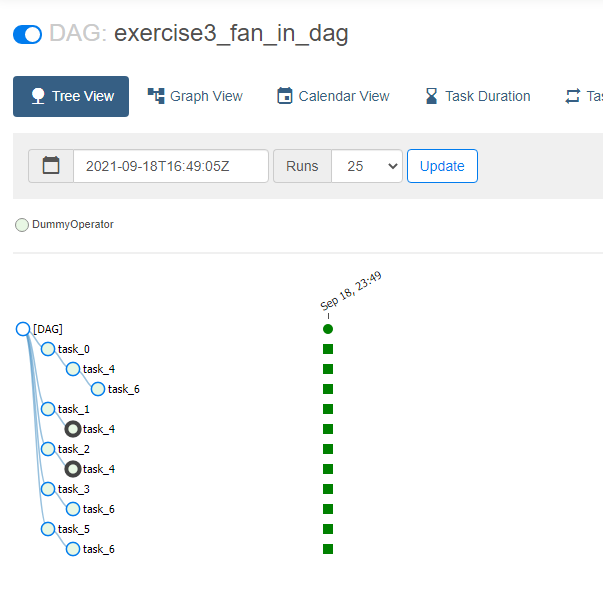
Copying file://start\_ws4\_exercise3.py [Content-Type=text/x-python]...

/ [1 files][ 968.0 B/ 968.0 B]

Operation completed over 1 objects/968.0 B.







จะเจอ dummy เยอะเลยไม่ต้องรอนาน

ลองอีกรอบ สร้างเป็น loop ในการสร้าง dummy operator

from airflow.models import DAG

from airflow.operators.dummy import DummyOperator

from airflow.utils.dates import days\_ago

with DAG(

    "exercise3\_fan\_in\_dag\_with\_loop",

    start\_date=days\_ago(1),

    schedule\_interval=None,

    tags=["exercise"]

) as dag:

    # Exercise3: Fan-in Pipeline

    # ใน exercise นี้จะได้รู้จักการเขียน task ใน pipeline ขั้นตอนเยอะขึ้น

    # ใช้ DummyOperator เป็น task จำลอง

    t = [DummyOperator(task\_id =f"task\_id\_{i}") for i in range(7)]

    [t[0],t[1],t[2] ]>> t[4]

    [t[3],t[4],t[5]] >> t[6]

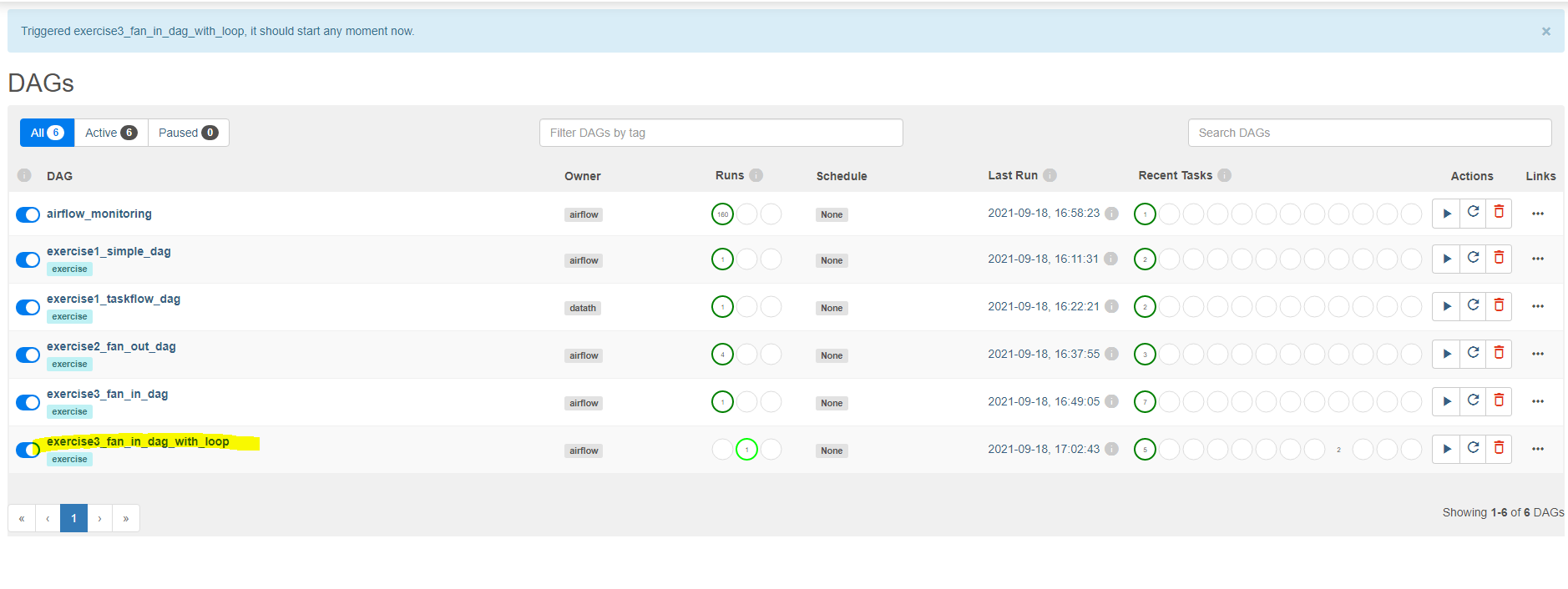
    # TODO: สร้าง DummyOperator เพื่อสร้าง dependency ที่ซับซ้อน

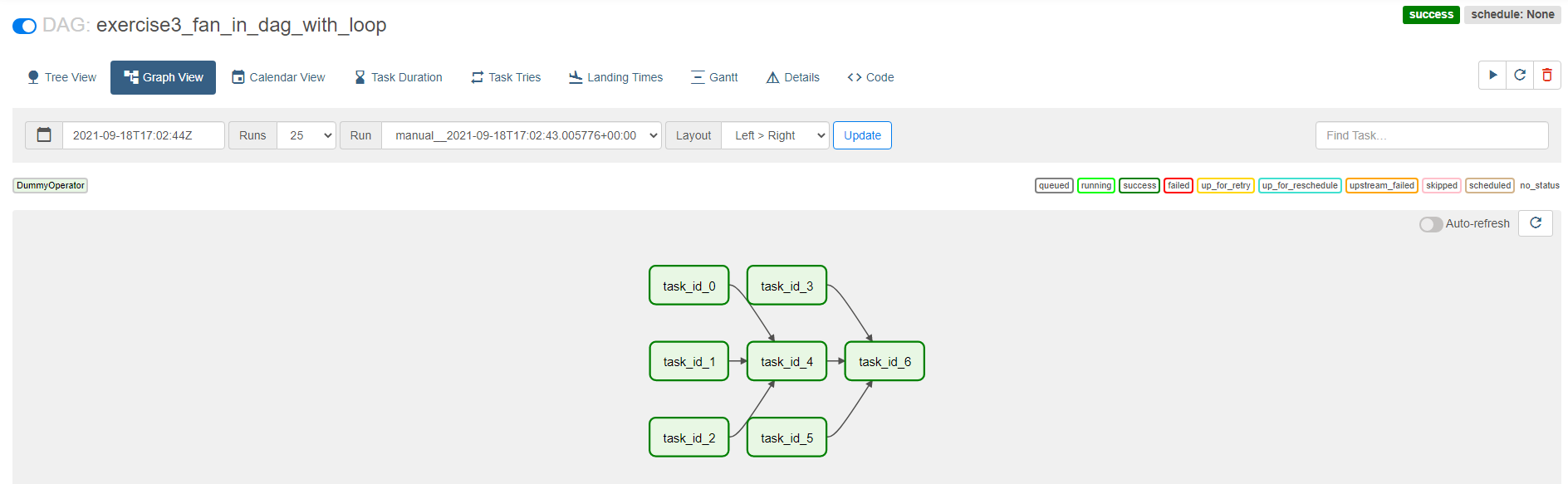
row3no6@cloudshell:~ (r2detitle19484)$ gsutil cp start\_ws4\_exercise3\_loop.py gs://asia-east2-workshop4-a4fa89c9-bucket/dags

Copying file://start\_ws4\_exercise3\_loop.py [Content-Type=text/x-python]...

/ [1 files][ 758.0 B/ 758.0 B]

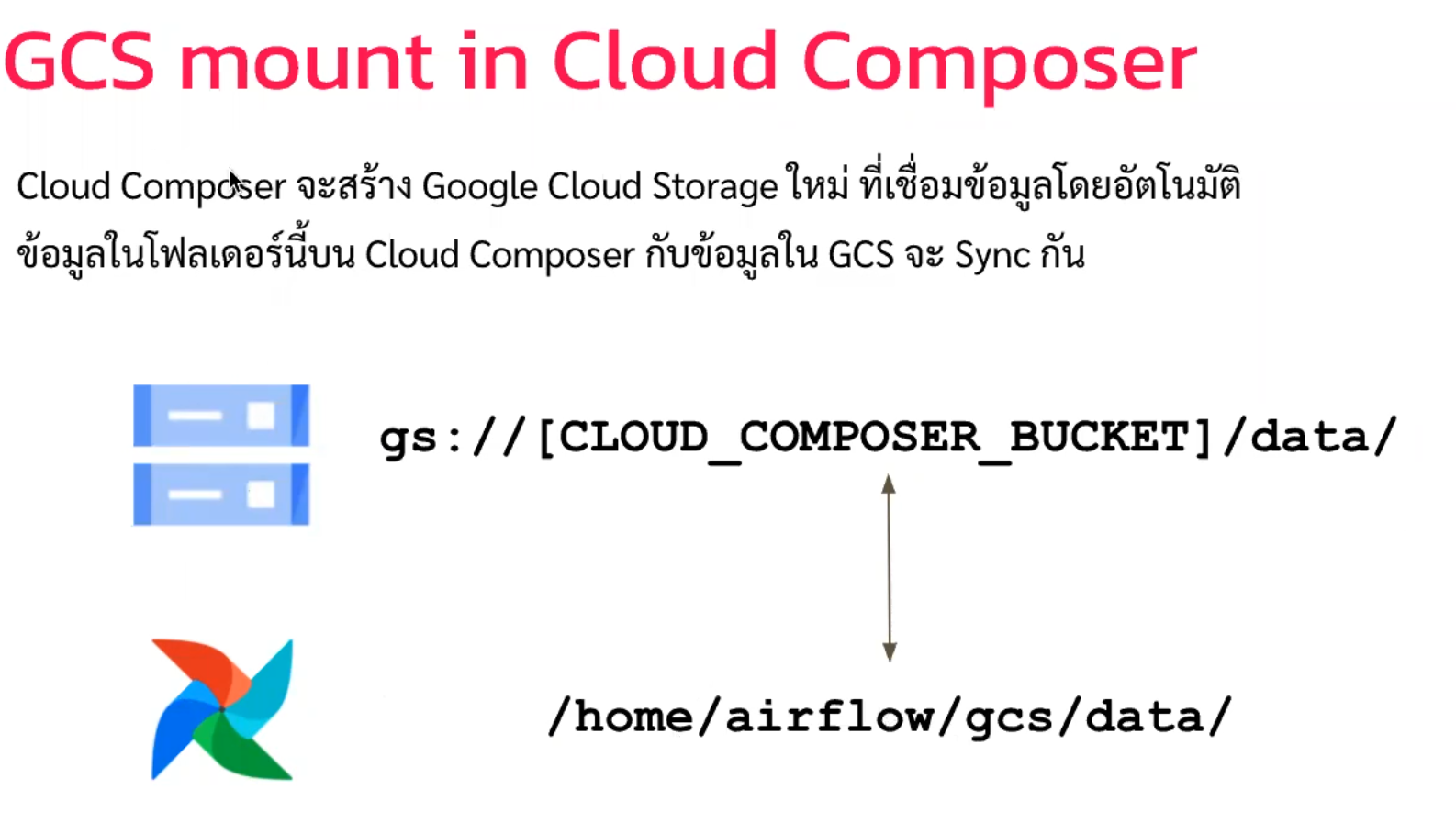
Operation completed over 1 objects/758.0 B.

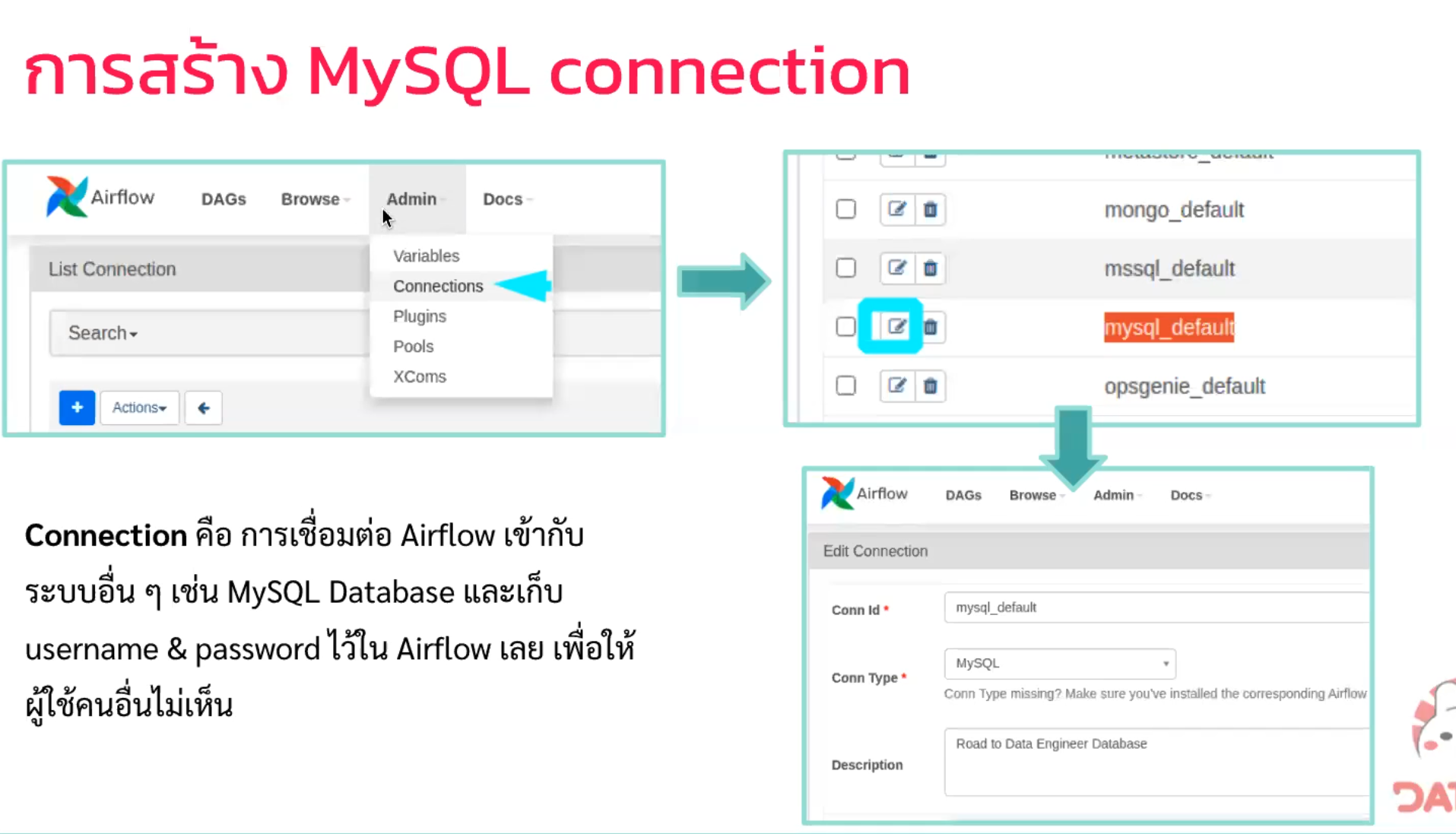




Exercise 4 Final Pipeline

วาง = sync ให้เลย





MYSQL\_HOST: 34.136.184.58

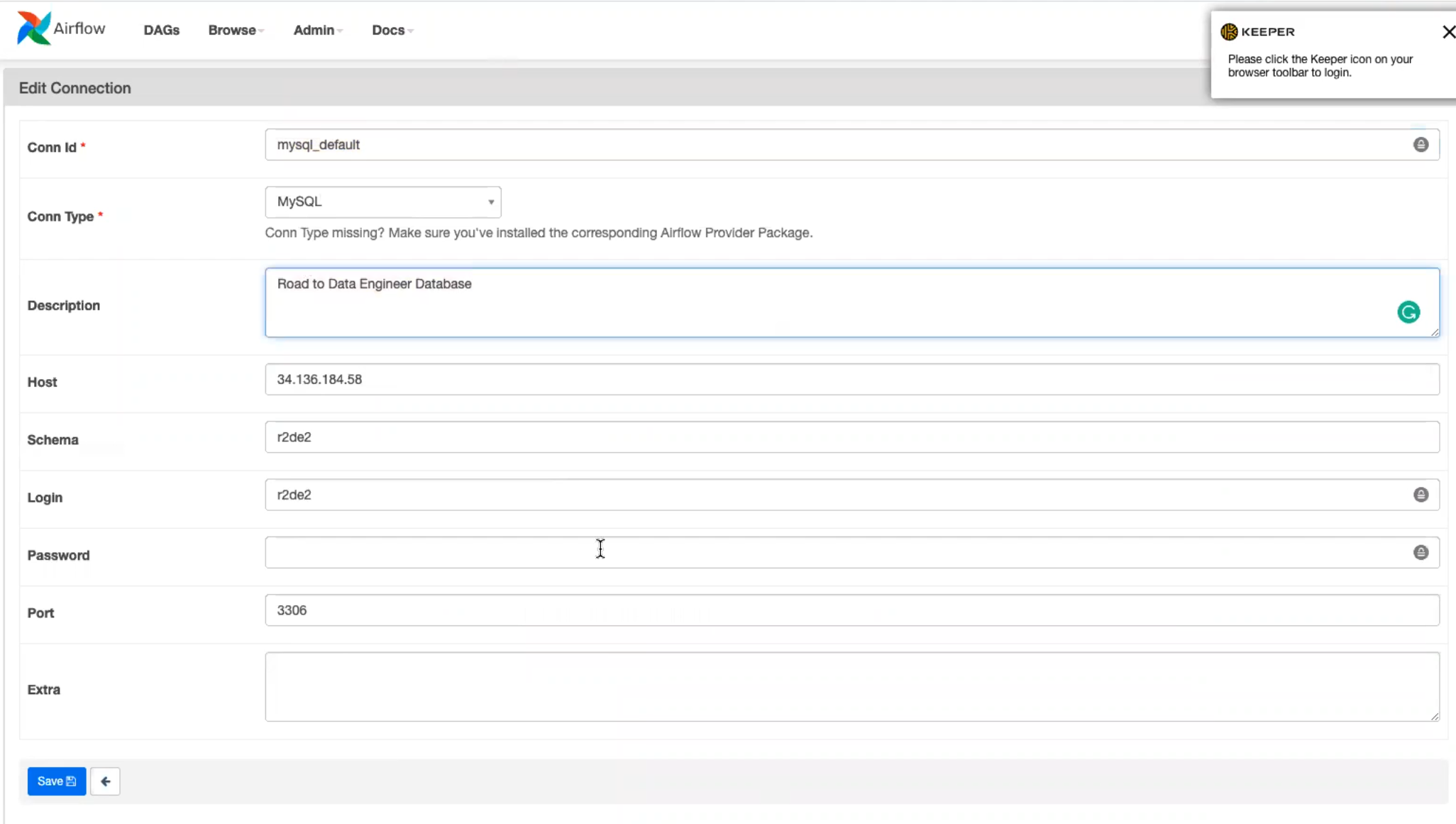
MYSQL\_DB (Schema): r2de2

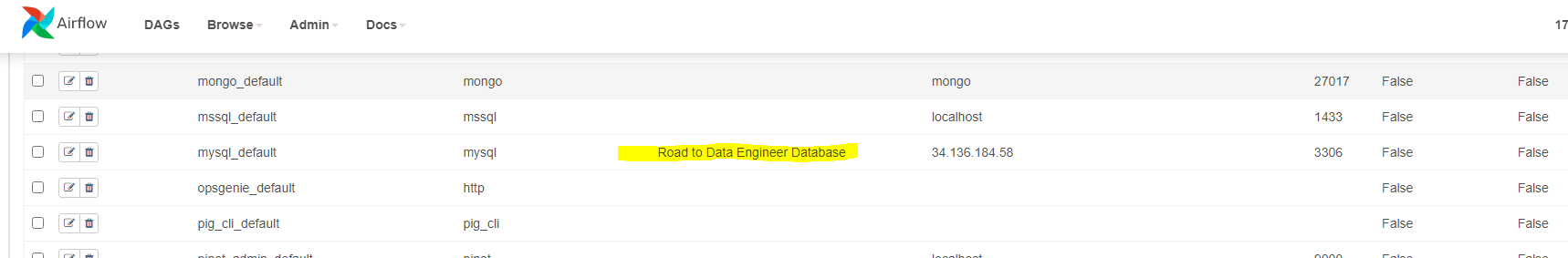
MYSQL\_USER (Login): r2de2

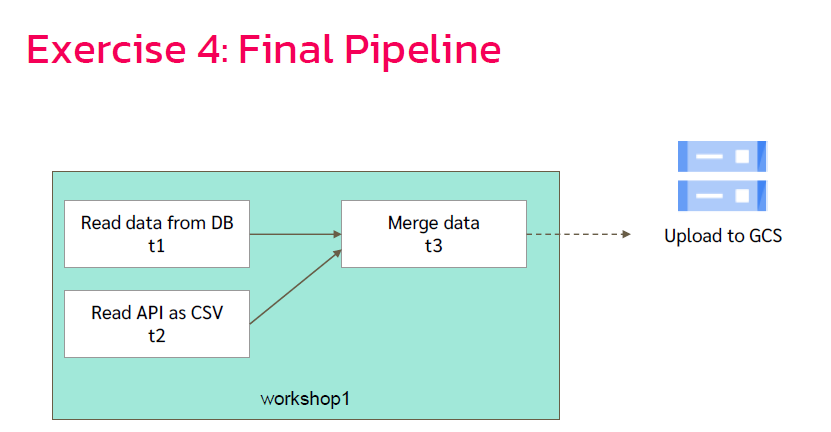
MYSQL\_PASSWORD: I\_Love\_Data\_Engineer

MYSQL\_PORT: 3306

มา Automate งาน Workshop1







from airflow.models import DAG

from airflow.operators.python import PythonOperator

from airflow.providers.mysql.hooks.mysql import MySqlHook #เพื่อ get data

from airflow.utils.dates import days\_ago

import pandas as pd

import requests

MYSQL\_CONNECTION = "mysql\_default"   # ชื่อของ connection ใน Airflow ที่เซ็ตเอาไว้

CONVERSION\_RATE\_URL = "https://r2de2-workshop-vmftiryt6q-ts.a.run.app/usd\_thb\_conversion\_rate"

# path ที่จะใช้

mysql\_output\_path = "/home/airflow/gcs/data/audible\_data\_merged.csv"

conversion\_rate\_output\_path = "/home/airflow/gcs/data/conversion\_rate.csv"

final\_output\_path = "/home/airflow/gcs/data/output.csv"

def get\_data\_from\_mysql(transaction\_path):

    # รับ transaction\_path มาจาก task ที่เรียกใช้

    # เรียกใช้ MySqlHook เพื่อต่อไปยัง MySQL จาก connection ที่สร้างไว้ใน Airflow

    mysqlserver = MySqlHook(MYSQL\_CONNECTION)

    # Query จาก database โดยใช้ Hook ที่สร้าง ผลลัพธ์ได้ pandas DataFrame

    audible\_data = mysqlserver.get\_pandas\_df(sql="SELECT \* FROM audible\_data")

    audible\_transaction = mysqlserver.get\_pandas\_df(sql="SELECT \* FROM audible\_transaction")

    # Merge data จาก 2 DataFrame เหมือนใน workshop1

    df = audible\_transaction.merge(audible\_data, how="left", left\_on="book\_id", right\_on="Book\_ID")

    # Save ไฟล์ CSV ไปที่ transaction\_path ("/home/airflow/gcs/data/audible\_data\_merged.csv")

    # จะไปอยู่ที่ GCS โดยอัตโนมัติ

    df.to\_csv(transaction\_path, index=False)

    print(f"Output to {transaction\_path}")

def get\_conversion\_rate(conversion\_rate\_path):

    r = requests.get(CONVERSION\_RATE\_URL)

    result\_conversion\_rate = r.json()

    df = pd.DataFrame(result\_conversion\_rate)

    # เปลี่ยนจาก index ที่เป็น date ให้เป็น column ชื่อ date แทน แล้วเซฟไฟล์ CSV

    df = df.reset\_index().rename(columns={"index": "date"})

    df.to\_csv(conversion\_rate\_path, index=False)

    print(f"Output to {conversion\_rate\_path}")

def merge\_data(transaction\_path, conversion\_rate\_path, output\_path):

    # อ่านจากไฟล์ สังเกตว่าใช้ path จากที่รับ parameter มา

    transaction = pd.read\_csv(transaction\_path)

    conversion\_rate = pd.read\_csv(conversion\_rate\_path)

    transaction['date'] = transaction['timestamp']

    transaction['date'] = pd.to\_datetime(transaction['date']).dt.date

    conversion\_rate['date'] = pd.to\_datetime(conversion\_rate['date']).dt.date

    # merge 2 DataFrame

    final\_df = transaction.merge(conversion\_rate, how="left", left\_on="date", right\_on="date")

    # แปลงราคา โดยเอาเครื่องหมาย $ ออก และแปลงให้เป็น float

    final\_df["Price"] = final\_df.apply(lambda x: x["Price"].replace("$",""), axis=1)

    final\_df["Price"] = final\_df["Price"].astype(float)

    final\_df["THBPrice"] = final\_df["Price"] \* final\_df["conversion\_rate"]

    final\_df = final\_df.drop("date", axis=1)

    # save ไฟล์ CSV

    final\_df.to\_csv(output\_path, index=False)

    print(f"Output to {output\_path}")

    print("== End of Workshop 4 ʕ•́ᴥ•̀ʔっ♡ ==")

with DAG(

    "exercise4\_final\_dag",

    start\_date=days\_ago(1),

    schedule\_interval="@once",

    tags=["workshop"]

) as dag:

    # TODO: สร้าง t1, t2, t3 ที่ใช้ PythonOperator

    # และสร้าง task dependencies

from airflow.models import DAG

from airflow.operators.python import PythonOperator

from airflow.providers.mysql.hooks.mysql import MySqlHook #เพื่อ get data  ใช้ provider ของ airflow

from airflow.utils.dates import days\_ago

import pandas as pd

import requests

MYSQL\_CONNECTION = "mysql\_default"   # ชื่อของ connection ใน Airflow ที่เซ็ตเอาไว้

CONVERSION\_RATE\_URL = "https://r2de2-workshop-vmftiryt6q-ts.a.run.app/usd\_thb\_conversion\_rate"

# path ที่จะใช้

mysql\_output\_path = "/home/airflow/gcs/data/audible\_data\_merged.csv"

conversion\_rate\_output\_path = "/home/airflow/gcs/data/conversion\_rate.csv"

final\_output\_path = "/home/airflow/gcs/data/output.csv"

def get\_data\_from\_mysql(transaction\_path):

    # รับ transaction\_path มาจาก task ที่เรียกใช้

    # เรียกใช้ MySqlHook เพื่อต่อไปยัง MySQL จาก connection ที่สร้างไว้ใน Airflow

    mysqlserver = MySqlHook(MYSQL\_CONNECTION)

    # Query จาก database โดยใช้ Hook ที่สร้าง ผลลัพธ์ได้ pandas DataFrame

    audible\_data = mysqlserver.get\_pandas\_df(sql="SELECT \* FROM audible\_data")

    audible\_transaction = mysqlserver.get\_pandas\_df(sql="SELECT \* FROM audible\_transaction")

    # Merge data จาก 2 DataFrame เหมือนใน workshop1

    df = audible\_transaction.merge(audible\_data, how="left", left\_on="book\_id", right\_on="Book\_ID")

    # Save ไฟล์ CSV ไปที่ transaction\_path ("/home/airflow/gcs/data/audible\_data\_merged.csv")

    # จะไปอยู่ที่ GCS โดยอัตโนมัติ

    df.to\_csv(transaction\_path, index=False)

    print(f"Output to {transaction\_path}") #save ลง output path

def get\_conversion\_rate(conversion\_rate\_path):

    r = requests.get(CONVERSION\_RATE\_URL)

    result\_conversion\_rate = r.json()

    df = pd.DataFrame(result\_conversion\_rate)

    # เปลี่ยนจาก index ที่เป็น date ให้เป็น column ชื่อ date แทน แล้วเซฟไฟล์ CSV

    df = df.reset\_index().rename(columns={"index": "date"})

    df.to\_csv(conversion\_rate\_path, index=False)

    print(f"Output to {conversion\_rate\_path}")

def merge\_data(transaction\_path, conversion\_rate\_path, output\_path):

    # อ่านจากไฟล์ สังเกตว่าใช้ path จากที่รับ parameter มา

    transaction = pd.read\_csv(transaction\_path)

    conversion\_rate = pd.read\_csv(conversion\_rate\_path)

    transaction['date'] = transaction['timestamp']

    transaction['date'] = pd.to\_datetime(transaction['date']).dt.date

    conversion\_rate['date'] = pd.to\_datetime(conversion\_rate['date']).dt.date

    # merge 2 DataFrame

    final\_df = transaction.merge(conversion\_rate, how="left", left\_on="date", right\_on="date")

    # แปลงราคา โดยเอาเครื่องหมาย $ ออก และแปลงให้เป็น float

    final\_df["Price"] = final\_df.apply(lambda x: x["Price"].replace("$",""), axis=1)

    final\_df["Price"] = final\_df["Price"].astype(float)

    final\_df["THBPrice"] = final\_df["Price"] \* final\_df["conversion\_rate"]

    final\_df = final\_df.drop("date", axis=1)

    # save ไฟล์ CSV

    final\_df.to\_csv(output\_path, index=False)

    print(f"Output to {output\_path}")

    print("== End of Workshop 4 ʕ•́ᴥ•̀ʔっ♡ ==")

with DAG(

    "exercise4\_final\_dag",

    start\_date=days\_ago(1),

    schedule\_interval="@once",

    tags=["workshop"]

) as dag:

    # TODO: สร้าง t1, t2, t3 ที่ใช้ PythonOperator

    # และสร้าง task dependencies

    '''

    # path ที่จะใช้

mysql\_output\_path = "/home/airflow/gcs/data/audible\_data\_merged.csv"

conversion\_rate\_output\_path = "/home/airflow/gcs/data/conversion\_rate.csv"

final\_output\_path = "/home/airflow/gcs/data/output.csv"

'''

    t1 = PythonOperator(

        task\_id="get\_data\_from\_mysql",

        python\_callable=get\_data\_from\_mysql,  #  def get\_conversion\_rate(conversion\_rate\_path):

        op\_kwargs={"transaction\_path": mysql\_output\_path,

        }, # path รับ

    )

    t2 = PythonOperator(

        task\_id="get\_conversion\_rate",

        python\_callable=get\_conversion\_rate,  #def get\_conversion\_rate(conversion\_rate\_path):

        op\_kwargs={"conversion\_rate\_path": conversion\_rate\_output\_path,

        }, # path รับ

    )

    t3 = PythonOperator(

        task\_id="merge\_data",

        python\_callable=merge\_data,  #def merge\_data(transaction\_path, conversion\_rate\_path, output\_path):

        op\_kwargs={

            "transaction\_path": mysql\_output\_path,

            "conversion\_rate\_path": conversion\_rate\_output\_path,

            "output\_path":final\_output\_path,

            }, # path รับ

    )

    [t1,t2] >> t3

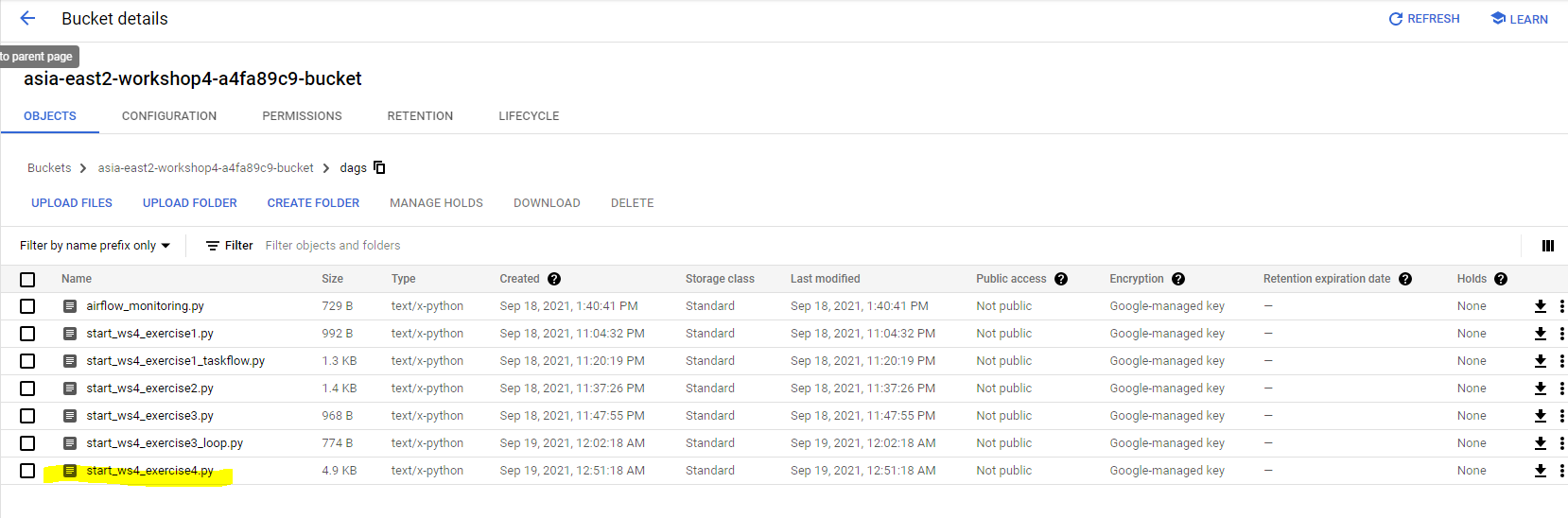
Cloud Shell รันคำสั่ง copy

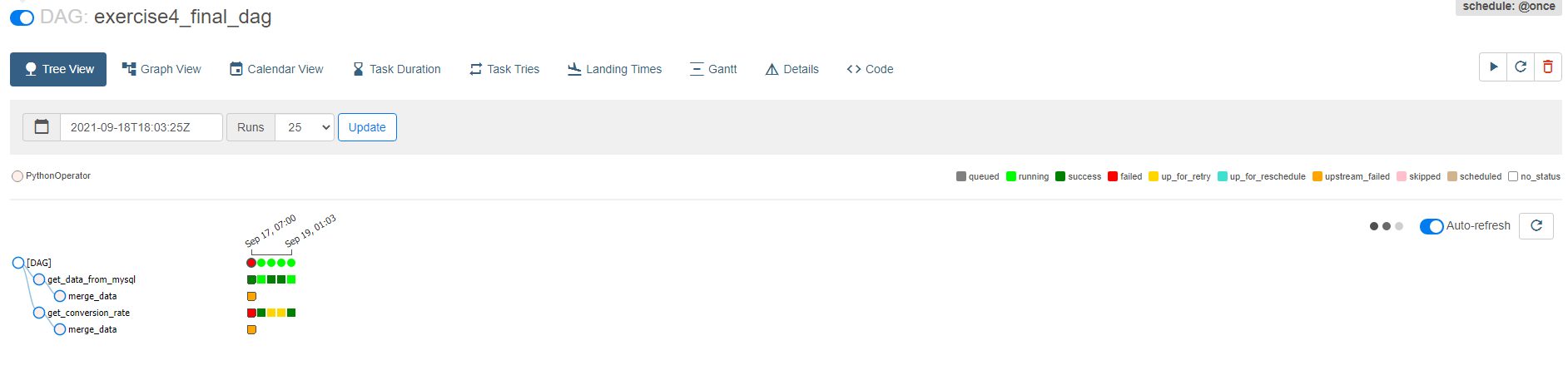
row3no6@cloudshell:~ (r2detitle19484)$ gsutil cp start\_ws4\_exercise4.py gs://asia-east2-workshop4-a4fa89c9-bucket/dags

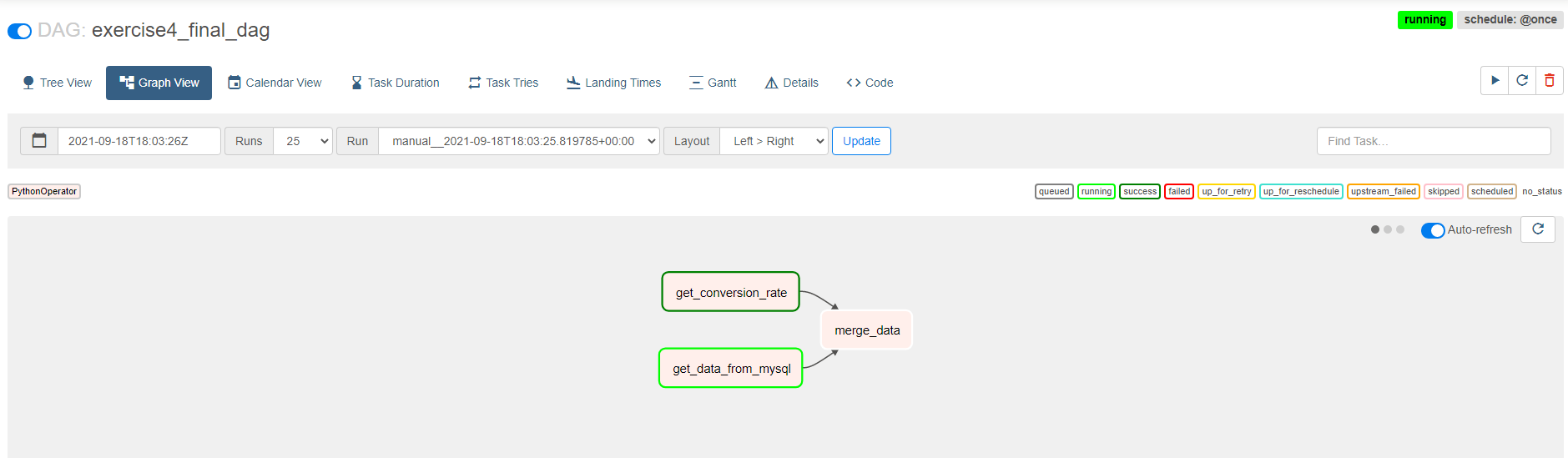
Copying file://start\_ws4\_exercise4.py [Content-Type=text/x-python]...

/ [1 files][ 4.8 KiB/ 4.8 KiB]

Operation completed over 1 objects/4.8 KiB.

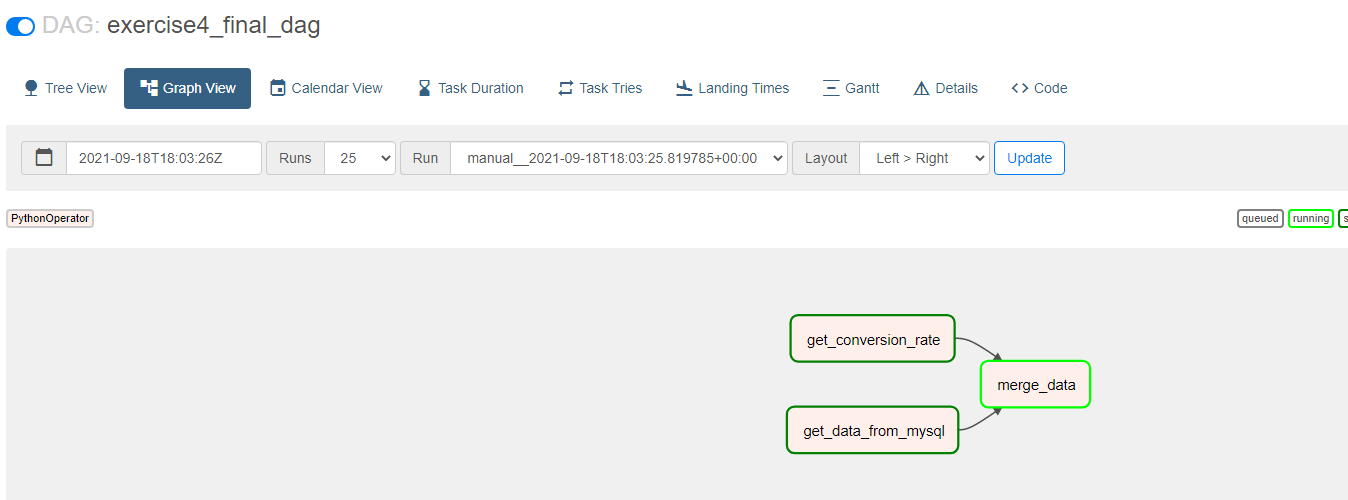
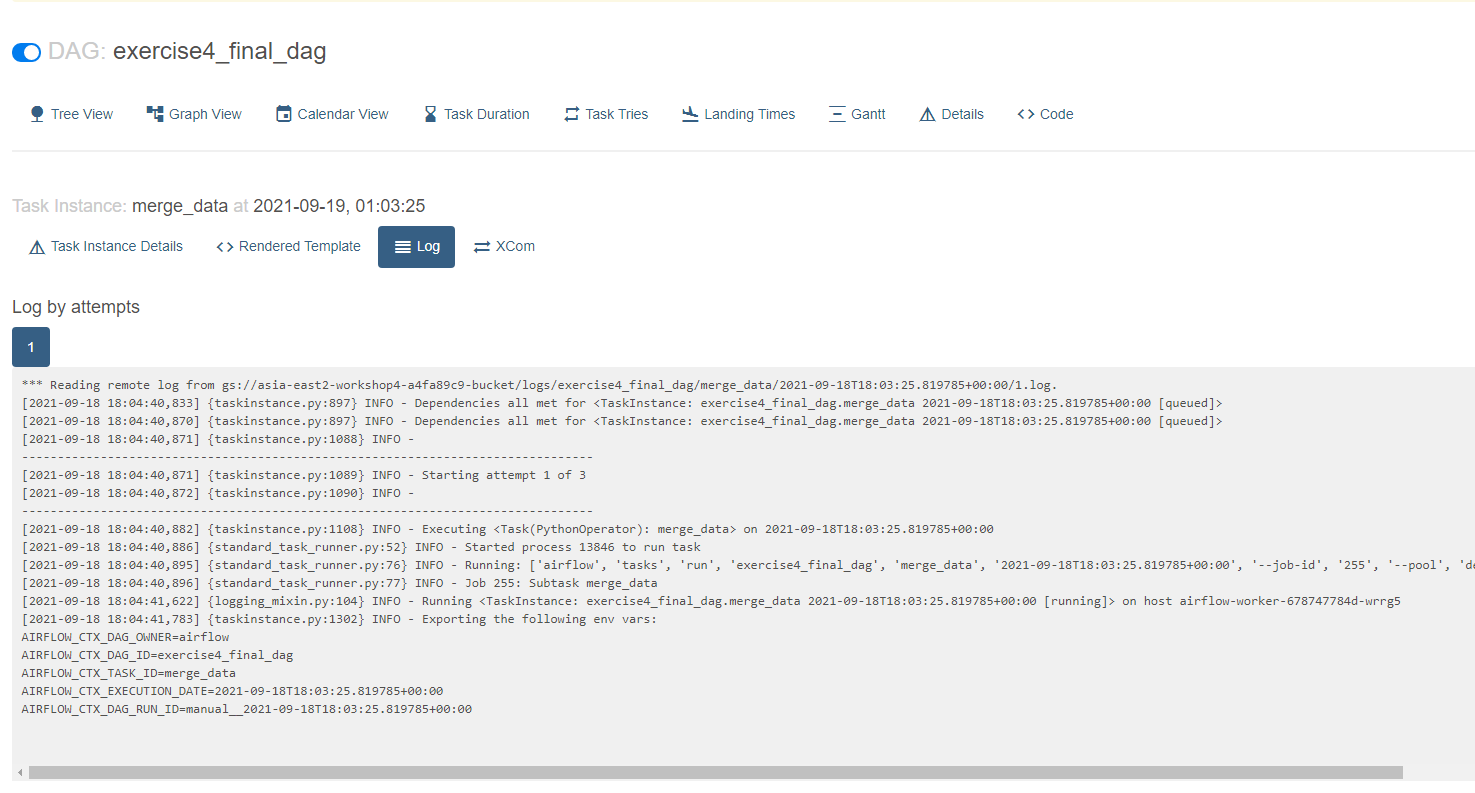




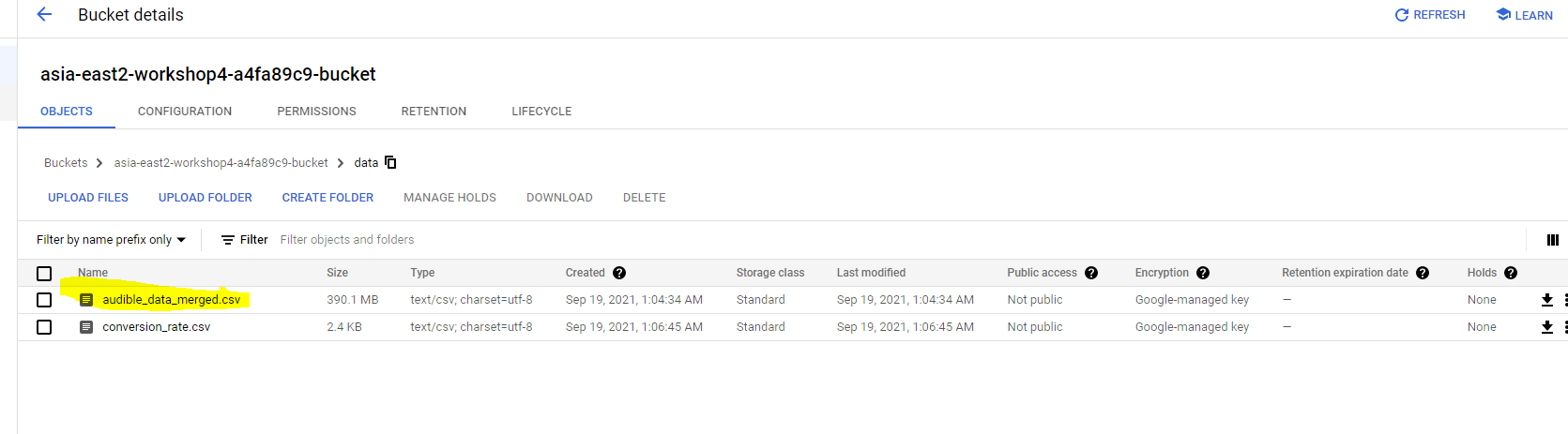


กำลังดึงข้อมูล กำลัง merge\_data

mysql\_output\_path = "/home/airflow/gcs/data/audible\_data\_merged.csv"

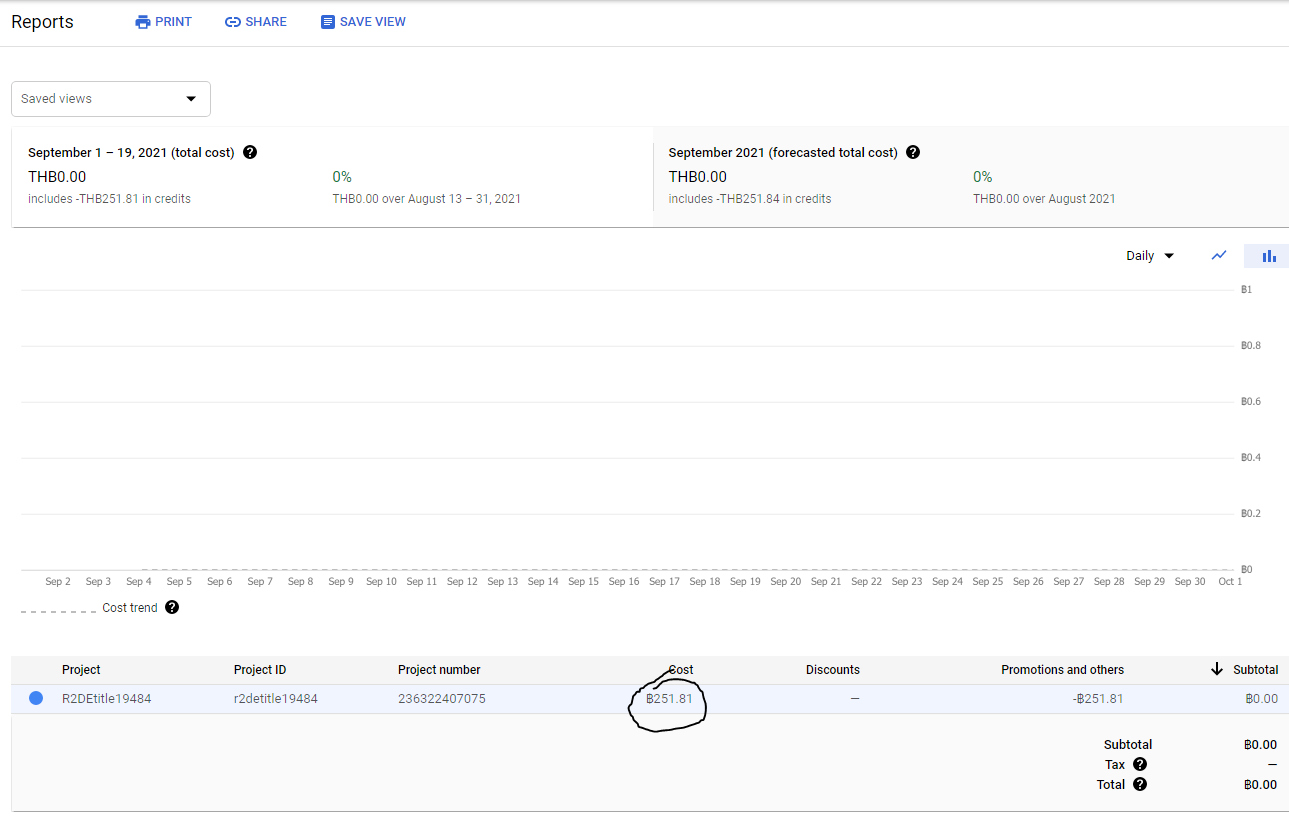
asia-east2-workshop4-a4fa89c9-bucket/data

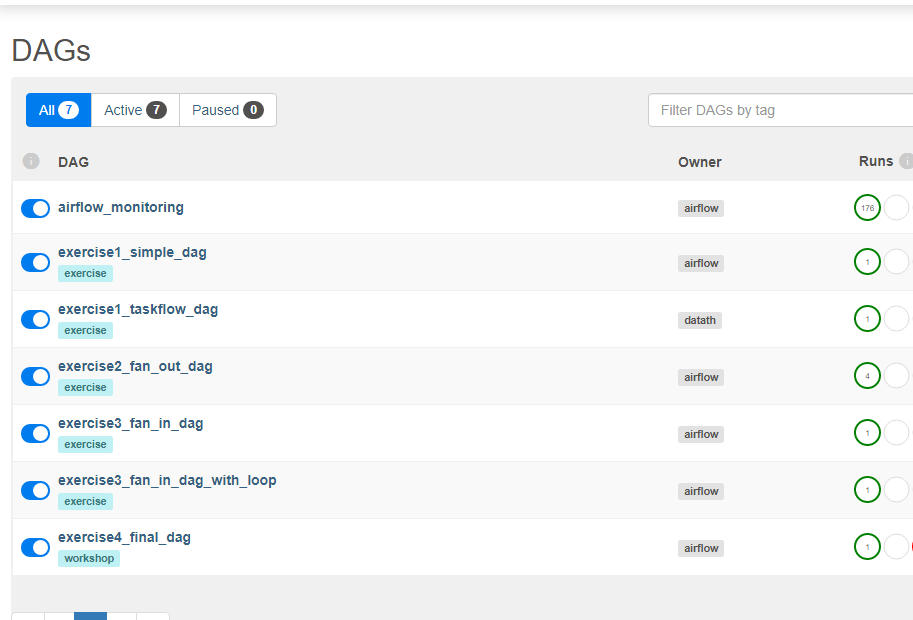


Airflow ไม่เหมาะที่จะประมวลผลขนาดใหญ่ที่มัน



กดดูค่าใช้จ่าย





การปิด Composer

