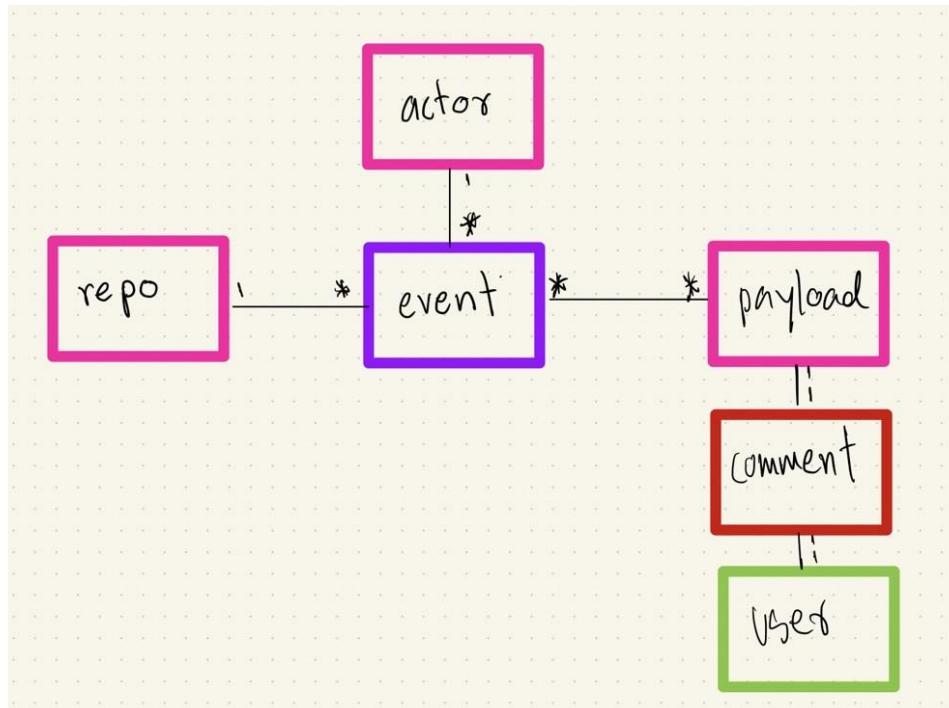


Building an Creating and Automating a Set of Data Pipelines with Airflow

วิธีการดำเนินการ

Data Model

ใช้ Data Model ที่ออกแบบไว้ จาก Project 01-data-modeling-i



1. เข้าไปที่ Gitpod สร้าง Folder

05-creating-and-scheduling-data-pipelines และเข้าไปที่ Folder ดังกล่าว
cd 05-creating-and-scheduling-data-pipelines

2. ตั้งค่า Environment

```
mkdir -p ./dags ./logs ./plugins  
echo -e "AIRFLOW_UID=$(id -u)" > .env
```

Project : 05-creating-and-scheduling-data-pipelines

Nattakan Towpunwong (64199130043)

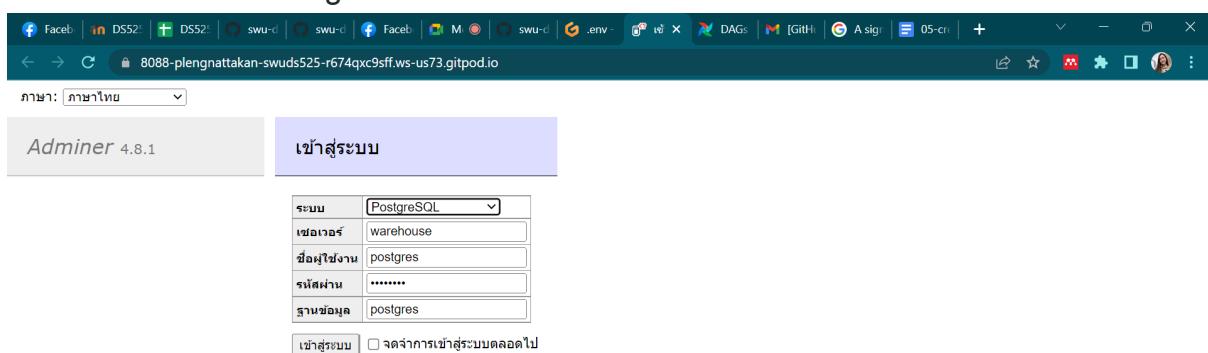
The screenshot shows a Gitpod workspace interface. The Explorer sidebar on the left lists files and folders: SWU-DS525, 05-creating-and-scheduling-data-pipelines (containing dags, data, etl.py, my_dag.py, and docker-compose.yaml), .env, README.md, data, and ENV. The main area contains three tabs: my_dags.py, etl.py, and my_dag.py. The etl.py tab shows code for importing json, glob, os, and typing.List. The my_dag.py tab shows a DAG definition. The terminal tab shows a bash session with commands like 'gitpod /workspace/swu-ds525' and environment variable 'AIRFLOW_UID=33333'. The Jupyter Variables panel shows no variables defined.

3. รัน Docker

docker-compose up

4. ย้าย data ไปไว้ใน dag และ เพิ่มไฟล์ .py ไว้สำหรับรัน

5. เข้าไปตั้งค่า Postgres



กำหนดให้

ระบบ : PostgreSQL

server : warehouse

ชื่อผู้ใช้งาน : postgres

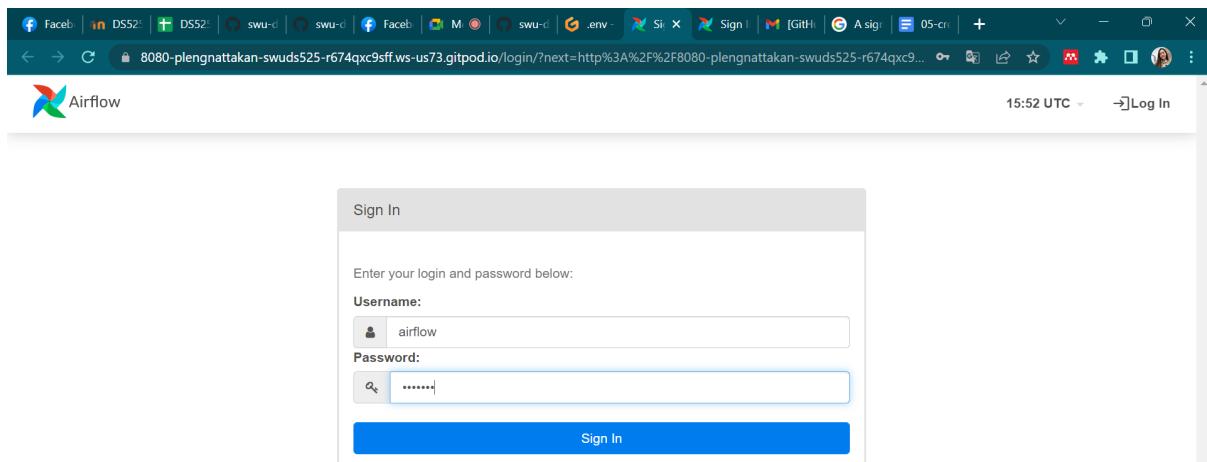
รหัสผ่าน : postgres

ฐานข้อมูล : postgres

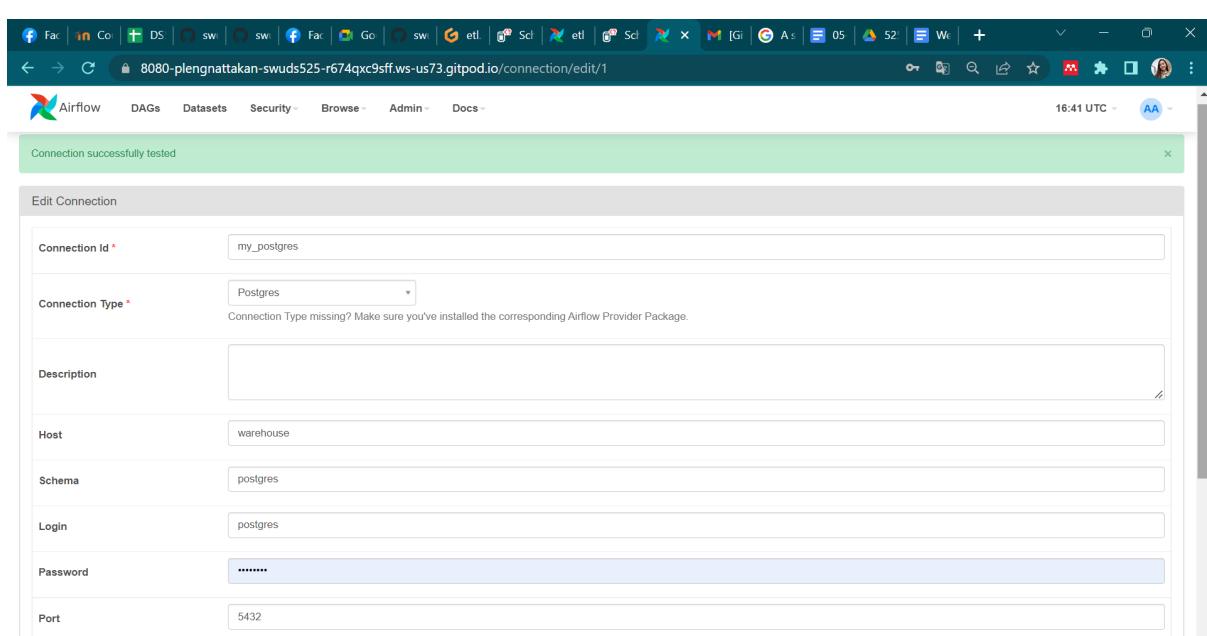
Project : 05-creating-and-scheduling-data-pipelines

Nattakan Towpunwong (64199130043)

6. เข้าไปที่ Airflow และตั้งค่าการเชื่อมต่อ กับ postgres ที่ Admin> Connections

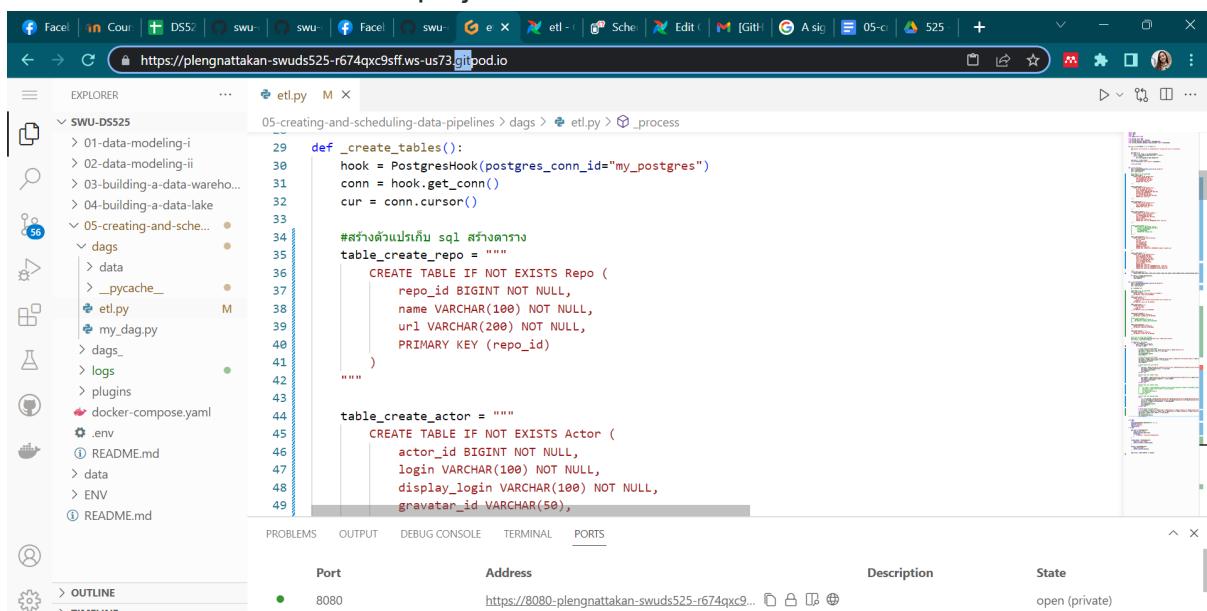


The screenshot shows the Airflow Sign In page. A modal window titled "Sign In" is displayed, prompting for "Username" and "Password". The "Username" field contains "airflow" and the "Password" field contains ".....". Below the fields is a blue "Sign In" button.



The screenshot shows the Airflow Admin > Connections page. A green success message at the top states "Connection successfully tested". Below it, there is a form titled "Edit Connection" for a connection named "my_postgres". The form includes fields for Connection Id (my_postgres), Connection Type (Postgres), Host (warehouse), Schema (postgres), Login (postgres), Password (hidden), and Port (5432). A note below the Connection Type dropdown says: "Connection Type missing? Make sure you've installed the corresponding Airflow Provider Package."

7. ปรับแก้ไข code ในไฟล์ .py ตามที่ออกแบบไว้ กำหนด scheduled ไว้เพื่อรัน ในส่วนนี้ได้ทำการนำ code จาก project 1 คือการ create table และ insert data มาใส่



The screenshot shows a code editor with a Python file named "etl.py" open. The code defines two functions: `_create_tables()` and `_process()`. The `_create_tables()` function uses a Postgres hook to create two tables: `Repo` and `Actor`. The `Repo` table has columns `repo_id`, `name`, and `url`. The `Actor` table has columns `actor_id`, `login`, `display_login`, and `gravatar_id`. The `_process()` function calls `_create_tables()` and then performs an insert operation into the `Repo` table.

```
def _create_tables():
    hook = PostgresHook(postgres_conn_id="my_postgres")
    conn = hook.get_conn()
    cur = conn.cursor()

    #สร้างตัวบล็อก sql สำหรับรัน
    table_create_repo = """
        CREATE TABLE IF NOT EXISTS Repo (
            repo_id BIGINT NOT NULL,
            name VARCHAR(100) NOT NULL,
            url VARCHAR(200) NOT NULL,
            PRIMARY KEY (repo_id)
        )
    """

    table_create_actor = """
        CREATE TABLE IF NOT EXISTS Actor (
            actor_id BIGINT NOT NULL,
            login VARCHAR(100) NOT NULL,
            display_login VARCHAR(100) NOT NULL,
            gravatar_id VARCHAR(50),
            PRIMARY KEY (actor_id)
        )
    """

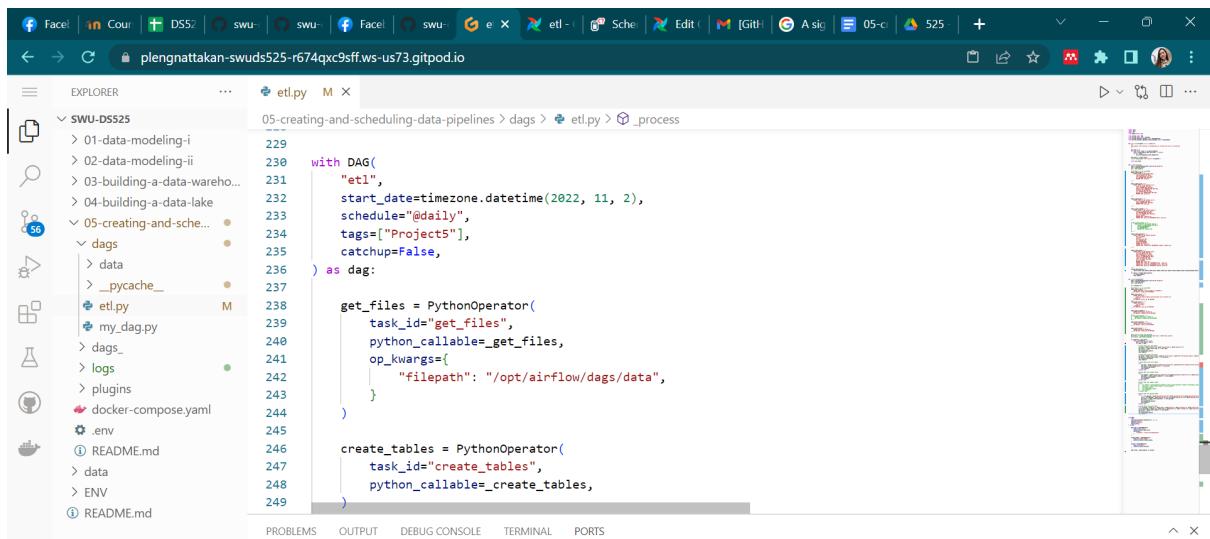
    conn.commit()

_process()
def _process():
    hook = PostgresHook(postgres_conn_id="my_postgres")
    conn = hook.get_conn()
    cur = conn.cursor()

    #insert data
    cur.execute(table_create_repo)
    cur.execute(table_create_actor)

    conn.commit()
```

Project : 05-creating-and-scheduling-data-pipelines
Nattakan Towpunwong (64199130043)

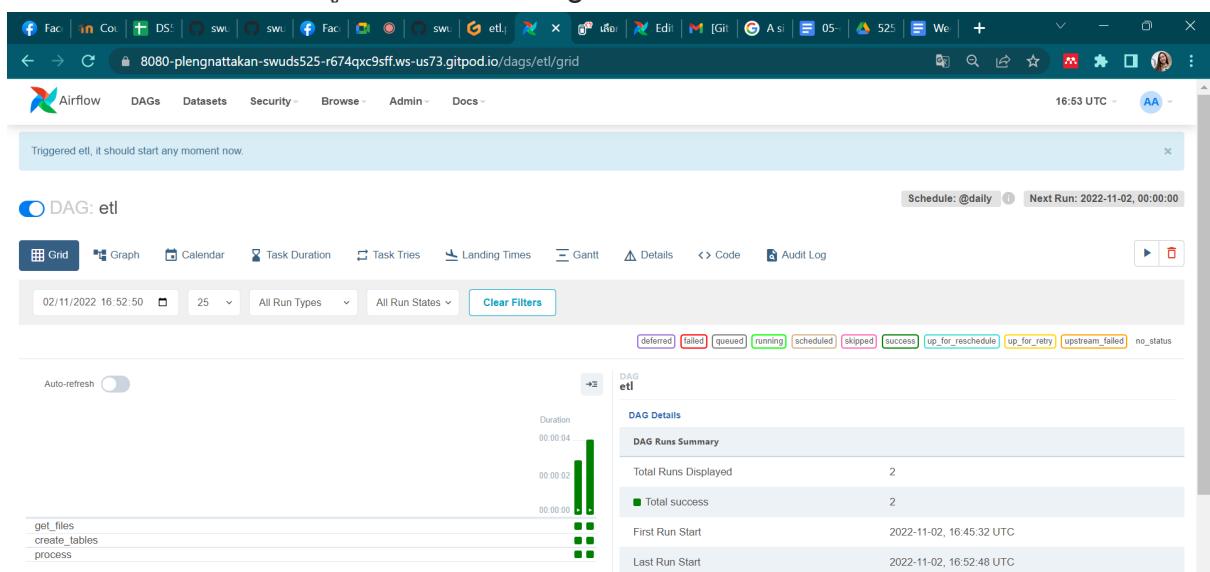


The screenshot shows a code editor with the file `etl.py` open. The code defines a DAG named `etl` with a daily schedule. It contains two tasks: `get_files` and `create_tables`, both using the `PythonOperator`. The code is well-formatted with line numbers and syntax highlighting.

```
229 with DAG(
230     "etl",
231     start_date=timezone.datetime(2022, 11, 2),
232     schedule="@daily",
233     tags=["Project5"],
234     catchup=False,
235 ) as dag:
236
237     get_files = PythonOperator(
238         task_id="get_files",
239         python_callable=_get_files,
240         op_kwargs={
241             "filepath": "/opt/airflow/dags/data",
242         }
243     )
244
245     create_tables = PythonOperator(
246         task_id="create_tables",
247         python_callable=_create_tables,
248     )
249 
```

ส่วนนี้จะเป็นการกำหนดว่าจะเริ่มรันตั้งแต่วันไหน และความถี่เป็นเท่าไหร่

8. ทดลองรัน เข้าไปดูผลการรัน ที่ Postgres



The screenshot shows the Airflow web interface with the URL `8080-plengnattakan-swuds525-r674qxc9sff.ws-us73.gitpod.io/dags/etl/grid`. The page displays the DAG `etl` with its run history. It shows two successful runs, with the most recent one starting at 2022-11-02, 16:45:32 UTC. The interface includes various filters and a timeline chart showing the duration of each run.

Schedule: @daily | Next Run: 2022-11-02, 00:00:00

DAG: etl

Triggered etl, it should start any moment now.

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

02/11/2022 16:52:50 25 All Run Types All Run States Clear Filters

Duration: 00:00:04, 00:00:03, 00:00:00

get_files, create_tables, process

DAG etl

DAG Details

Total Runs Displayed	2
Total success	2
First Run Start	2022-11-02, 16:45:32 UTC
Last Run Start	2022-11-02, 16:52:48 UTC

Project : 05-creating-and-scheduling-data-pipelines

Nattakan Towpunwong (64199130043)

ภาษา: ภาษาไทย

PostgreSQL » warehouse » postgres » Schema: public

ออกจากระบบ

Schema: public

เปลี่ยนแปลง schema Schema ของฐานข้อมูล

ตารางและวิว

ค้นหาในตาราง (8)

	ตาราง	ชื่อต้องฐานข้อมูล	การตรวจสอบ	ความพยายามของชั่วโมง?	ความพยายามของลับมี?	พื้นที่ร่าง	เพื่อคลิกโดยอัตโนมัติ	แก้?	หมายเหตุ?
<input type="checkbox"/>	actor	table		24 576	49 152	?	?	127	
<input type="checkbox"/>	actors	table		8 192	24 576	?	?	127	
<input type="checkbox"/>	comment	table		8 192	16 384	?	?	0	
<input type="checkbox"/>	event	table		16 384	40 960	?	?	150	
<input type="checkbox"/>	events	table		16 384	49 152	?	?	150	
<input type="checkbox"/>	payload	table		16 384	40 960	?	?	90	
<input type="checkbox"/>	repo	table		16 384	40 960	?	?	135	
<input type="checkbox"/>	user	table		8 192	16 384	?	?	0	

Selected (0)

Vacuum | เพิ่มประวัติการ | ล็อกทั้ง | ลบ | นำกลับ

ป้ายไปยังฐานข้อมูลอื่น: public

The screenshot shows the Adminer 4.8.1 web-based database management tool. The top navigation bar includes links for Facebook, Cloud, DS, SWI, etl, MySQL, Edit, Git, and others. The main title is "PostgreSQL » warehouse » postgres » public » เลือก: events". The left sidebar displays the Adminer logo, version 4.8.1, and dropdown menus for "DB: postgres" and "Schema: public". Below these are sections for "คำสั่ง SQL" (SQL Commands) containing "SELECT * FROM "events" LIMIT 50 (0.000 วินาที)" and "แก้ไข" (Edit), and a list of table names: actor, actors, comment, event, events, payload, repo, and user. The main content area has a purple header "เลือก: events". It features a search bar with placeholder "เลือกชื่อคุณ" and a "เลือก" button. Below the search bar are buttons for "ดูหน้า", "เรียงลำดับ", "จำนวน", and "ดำเนินการ". A table titled "SELECT * FROM "events" LIMIT 50 (0.000 วินาที)" lists 150 rows of event data. The table columns are "Modify", "id", "type", and "actor_id". Each row contains a checkbox, the event ID, the event type, and the actor ID. The "type" column values include IssueCommentEvent, PushEvent, CreateEvent, and IssuesEvent.

Modify	id	type	actor_id
<input type="checkbox"/>	23487929637	IssueCommentEvent	1696078
<input type="checkbox"/>	23487929676	PushEvent	66924041
<input type="checkbox"/>	23487929674	PushEvent	74971347
<input type="checkbox"/>	23487929661	PushEvent	91143053
<input type="checkbox"/>	23487929682	PushEvent	60825784
<input type="checkbox"/>	23487929673	PushEvent	102357035
<input type="checkbox"/>	23487929588	PushEvent	46447321
<input type="checkbox"/>	23487929636	CreateEvent	41898282
<input type="checkbox"/>	23487929580	IssuesEvent	21972349
<input type="checkbox"/>	23487929591	PushEvent	17869732

โดยกราฟขั้นตอนในการรับจะได้ดังนี้

