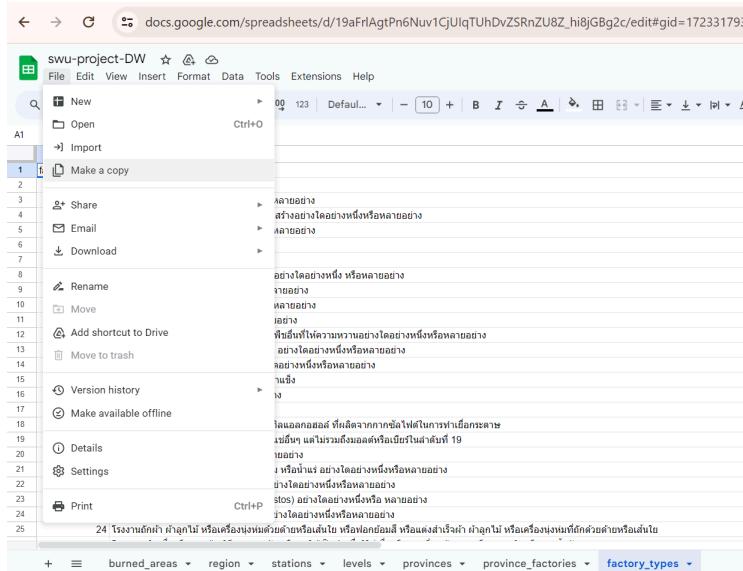


# Instruction on Running the Project

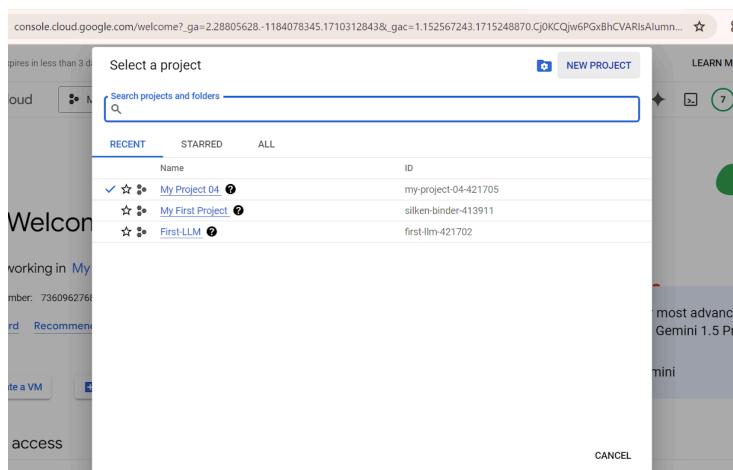
- เริ่มแรกให้ทำการ copy ข้อมูลจาก google sheet ของโครงการเข้าไปที่ google drive ของผู้ดูดตั้ง



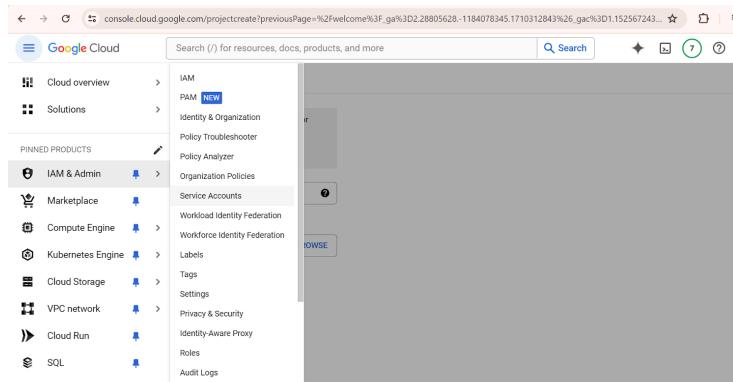
- ทำการ copy spreadsheet id จากใน google sheet ตามที่ໄช่ไฟล์ดังภาพด้านล่าง และเก็บไว้เพื่อใช้ระบุในไฟล์ data-api-upsert.py

B1	factory_type_name
1	กรรมการผู้จัดการใหญ่
2	กรรมการผู้จัดการใหญ่ที่ไม่ใช่หุ้นส่วน
3	หุ้นส่วนรายบุคคลที่ไม่เป็นกรรมการผู้จัดการใหญ่ของบริษัทและบุคคลอื่น
4	หุ้นส่วนที่ไม่ได้เป็นกรรมการผู้จัดการใหญ่ของบริษัทและบุคคลอื่น
5	บุคคลภายนอกที่ไม่ใช่หุ้นส่วน
6	เด็กนักเรียนของโรงเรียนของตน
7	เจ้าของบ้านที่อาศัยอยู่ในบ้านของตนโดยชอบด้วยกฎหมาย
8	บุคคลที่บ้านของตนไม่สามารถจ่ายค่าเช่าได้
9	แม่บ้านที่บ้านของตนไม่สามารถจ่ายค่าเช่าได้
10	อาจารย์และนักเรียนของมหาวิทยาลัยและบุคคลอื่น
11	ลูกน้อง ลูกจ้างและพนักงาน หรือช่างเชิงค้าขายของบ้านและบ้านที่เช่าโดยชอบด้วยกฎหมาย
12	ชาติไทย ชาติอื่น หรือชาติที่ไม่ใช่ชาติไทย ที่อาศัยอยู่ในบ้านของตนโดยชอบด้วยกฎหมาย
13	ผู้เช่าบ้านที่บ้านของตนไม่สามารถจ่ายค่าเช่าได้โดยชอบด้วยกฎหมาย
14	กรรมการผู้จัดการใหญ่ที่ไม่ใช่หุ้นส่วน
15	กรรมการผู้จัดการใหญ่ที่ไม่ใช่หุ้นส่วน
16	กรรมการผู้จัดการใหญ่ที่ไม่ใช่หุ้นส่วน
17	กรรมการผู้จัดการใหญ่ที่ไม่ใช่หุ้นส่วน
18	กรรมการผู้จัดการใหญ่ที่ไม่ใช่หุ้นส่วน ที่ไม่ได้เป็นกรรมการผู้จัดการใหญ่ที่ไม่ใช่หุ้นส่วน
19	กรรมการผู้จัดการใหญ่ที่ไม่ใช่หุ้นส่วน
20	บุคคลที่บ้านของตนไม่สามารถจ่ายค่าเช่าได้โดยชอบด้วยกฎหมาย

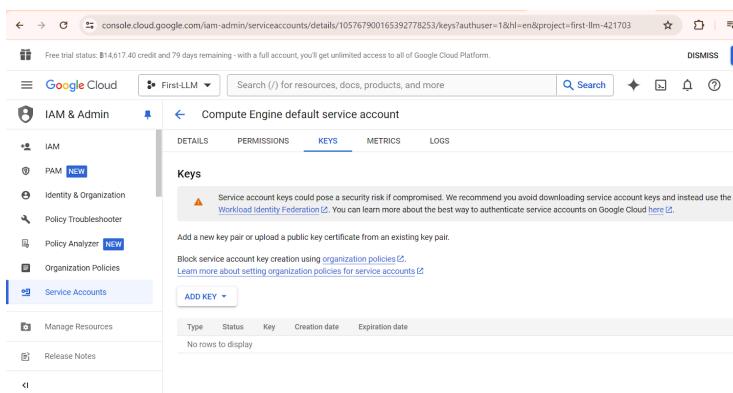
- เข้าไปที่ google cloud platform และทำการสร้าง project ใหม่



#### 4. หลังจากทำการสร้าง project เรียบร้อยแล้ว ให้เข้าไปที่ IAM & Admin >> Service accounts และทำการ create service account



#### 5. ทำการ add key ภายใต้ service account และเมื่อสร้างเสร็จแล้วจะได้รับไฟล์ token.json



#### 6. เปิดไฟล์ data-api-upsert.py เพื่อทำการแก้ไขข้อมูล spreadsheet id ก่อน โดยข้อมูลที่ต้องแก้ไขจะอยู่ในบรรทัดที่ 136 ภายใต้ชื่อตัวแปร SAMPLE\_SPREADSHEET\_ID

```
120 # Collect all results outside the loop
121 station_results = []
122 for station_id in data:
123     result = find_region_info(data[station_id])
124     if result:
125         station_results.append(result)
126
127 # Create DataFrame once outside the loop
128 df_stations = spark.createDataFrame(station_results, schema=["station_code", "station_name",
129
130 # Fetch Google Sheets data and convert to PySpark DataFrames
131 data_processed = []
132 creds = None
133
134 # Check if token file exists and credentials are valid
135 SCOPES = ["https://www.googleapis.com/auth/spreadsheets.readonly"]
136 SAMPLE_SPREADSHEET_ID = "19aFr1AgtPn6Nuv1CjUiqTUhDvZSRnZU8Z_hi8jGBg2c"
137
138 # columns to change type
139 long_cols = ['year', 'province_id', 'burned_area', 'factory_type_id', 'region_id', 'Level', 'strategic']
140 str_cols = ['factory_type_name', 'province_factory_area', 'province_name', 'region_name', 'strategic']
141 date_cols = ['date', 'string', 'date']
142 double_cols = ['value', 'lat', 'long']
143
144 for station_id in data:
145     processed_data = process_station_data(data[station_id])
146     data_processed.append(processed_data)
147
```

#### 7. เมื่อเตรียมข้อมูลครบถ้วนแล้ว ขั้นตอนถัดไปให้เข้าไปที่ databricks และทำการสร้าง workspace ใหม่สำหรับโปรเจค

The screenshot shows the Microsoft Azure Databricks workspace interface. On the left, there's a sidebar with navigation links like Home, Workspace, Catalog, Workflows, Compute, Data Engineering, Machine Learning, and more. The main area displays a list of items under 'Workspace > Home'. The list includes:

Name	Type	Owner	Created at
create_view_to_M4	Notebook	Jirasak Saimek	2024-05-07 16:25:30
create_view_to_M5.py	File	Jirasak Saimek	2024-05-07 16:26:17
credentials.json	File	Jirasak Saimek	2024-04-24 22:43:12
data-api-upsert.py	File	Jirasak Saimek	2024-04-24 17:05:50
git-api-source	Notebook	Jirasak Saimek	2024-04-20 17:50:50
prm_3 2024-05-07 13:43:37	Dashboard	Jirasak Saimek	2024-05-07 13:43:39
token.json	File	Jirasak Saimek	2024-05-03 10:27:04

## 8. ทำการ import ไฟล์ data-api-upsert.py ซึ่งสามารถดาวน์โหลดได้จาก github

This screenshot shows the same workspace interface as above, but with a right-click context menu open over the 'token.json' file. The menu options include 'Import', 'Export', 'Copy URL/path', and 'Add to favorites'. The 'Import' option is highlighted.

This screenshot shows the 'Import' dialog box open. It has two tabs: 'File' (selected) and 'URL'. The 'Target folder' field is set to '/Users/jirasak.saimek@gmail.com'. Below the dialog is a list of imported items:

Name	Type	Owner	Created at
create_view_to_M4	Notebook	Jirasak Saimek	2024-05-07 16:25:30
create_view_to_M5.py	File	Jirasak Saimek	2024-05-07 16:26:17
credentials.json	File	Jirasak Saimek	2024-05-07 16:26:51
data-api-upsert.py	File	Jirasak Saimek	2024-04-24 22:43:12
git-api-source	Notebook	Jirasak Saimek	2024-04-20 17:05:50
prm_3 2024-05-07 13:43:37	Dashboard	Jirasak Saimek	2024-05-07 13:43:39
prm_3.py	File	Jirasak Saimek	2024-05-09 00:10:48
token.json	File	Jirasak Saimek	2024-05-03 10:27:04
work-flow.yaml	File	Jirasak Saimek	2024-05-08 22:08:00

## 9. Upload ไฟล์ token.json ที่ได้มาจากการสร้าง key ใน google cloud platform ไว้ใน workspace ซึ่งวิธีการ import ไฟล์ เมื่อันกันกับขั้นตอนก่อนหน้า

## 10. Run code ในไฟล์ data-api-upsert.py เพื่อทดสอบ

```

File Edit View Run Help Last edit was 6 minutes ago New cell US ON
12:08 AM (6)
provinces.spark_df.write.format("delta").mode("overwrite").saveTable("default.provincefactories")
+ (4) Spark jobs
+ (1) 12:08 AM (6)
Import os
from google.auth import default
from google.auth.transport import requests
from googleapiclient import discovery
from googleapiclient.discovery import build
import json, ast
from pyspark.sql.functions import *
from pyspark.sql.functions import col, avg, max as spark_max, min as spark_min, round
from pyspark.read.table("default.levels")
v_levels = spark.read.table("default.levels")
v_boundaries = v_levels.withColumn("date", when(col("year") > 2500, col("year") - 543).otherwise(col("year")))
v_regions = spark.read.table("default.regions")
v_provinces = spark.read.table("default.provinces")
v_provincefactories = spark.read.table("default.provincefactories")
v_factortypes = spark.read.table("default.factortypes")

#process create_vine_name
v_boundaries = v_boundaries.withColumn("name", concat(v_boundaries["name"], " ", v_boundaries["name"]))
v_boundaries = v_boundaries.withColumn("date", when(col("year") > 2500, col("year") - 543).otherwise(col("year")))
v_boundaries = v_boundaries.withColumn("expr(concat(date, '-1-1'))")

#process create_vine_name_2
v_levels = spark.read.table("default.levels")
v_pm2_5 = spark.read.table("default.pm2_5")
v_pm2_5 = v_pm2_5.join(v_levels, v_pm2_5["value"] >= v_levels["Min"]) & (v_pm2_5["value"] <= v_levels["Max"]), "left"
v_pm2_5 = v_pm2_5.select("station_code", "date", "value", "level")
v_pm2_5 = v_pm2_5.select("station_code", "date", "value", "level")

```

11. เข้าไปที่ workflow และสร้าง task เพื่อบุลำดับงานหรือ notebook ที่จะใช้การ run รวมถึงกำหนด schedule เพื่อให้ทำงานที่เป็น data ingestion ในการดึงข้อมูลในทุกวัน

**Job details:**  
 Job ID: 22002594615839  
 Creator: irasakimk  
 Run as: irasakimk  
 Tag: None  
 Description: None  
**Schedule:**  
 At 07:00 AM (ET+0700 — Bangkok, Hanoi, Jakarta)  
 Edit schedule | Pause | Delete  
**Compute:**  
 Driver: Standard\_D4\_V2 Workers: Standard\_D4\_V2 - 4 workers - 13.3 LTS  
 Python packages: Apache-Spark 3.4.1, Scala 2.13  
**Job parameters:**  
 No job parameters are defined for this job  
 Edit parameters  
**Job notifications:**  
 No notifications  
 Edit notifications

### สิ่งที่ต้องทำในการสร้าง Task:

Task name: กรอกชื่อโปรเจคที่ต้องการตั้ง

Type: เลือก Python script

Source: เลือก Workspace

Path: เลือกไปที่ไฟล์ data-api-upsert.py ที่อยู่ใน workspace

Dependent libraries: เลือก +Add

**Add dependent library** Send feedback for library

An optional list of libraries to be installed on the cluster will execute the job.

Library Source: Workspace File Path/URL PyPI Maven CRAN DBFS

We recommend specifying an exact version of the library without -c or -e to provide a reproducible environment. Using a floating version may result in unexpected failures in future sessions. Learn more

Package: pyflame package (simpleton or simpleton==1.8.0)

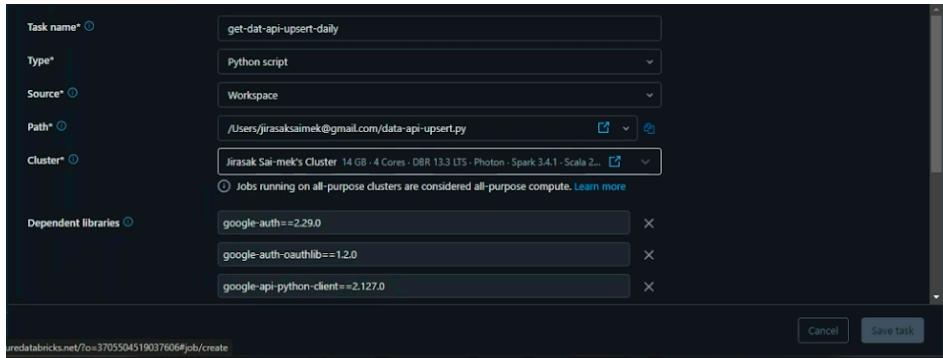
Index URL: Optional

Cancel Create task

**Job details:**  
 Job ID: 22002594615839  
 Creator: irasakimk  
 Run as: irasakimk  
 Tag: None  
 Description: None  
**Schedule:**  
 None  
**Job parameters:**  
 No job parameters are defined for this job  
 Edit parameters  
**Job notifications:**  
 No notifications  
 Edit notifications  
 Duration thresholds: No thresholds defined  
 Set duration thresholds

เลือกไปที่ PyPI และทำการเพิ่ม 3 libraries ตามด้านล่างนี้เข้าไปที่ task

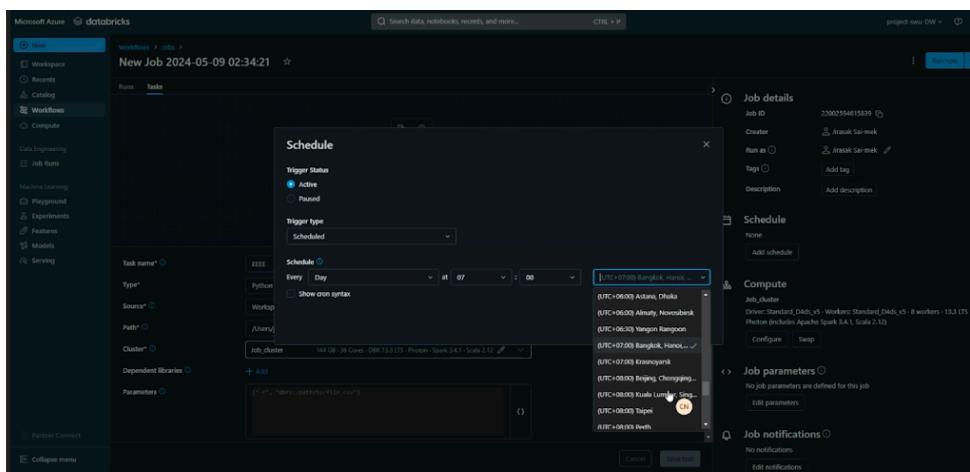
- google-auth==2.29.0
- google-auth-oauthlib==1.2.0
- google-api-python-client==2.127.0



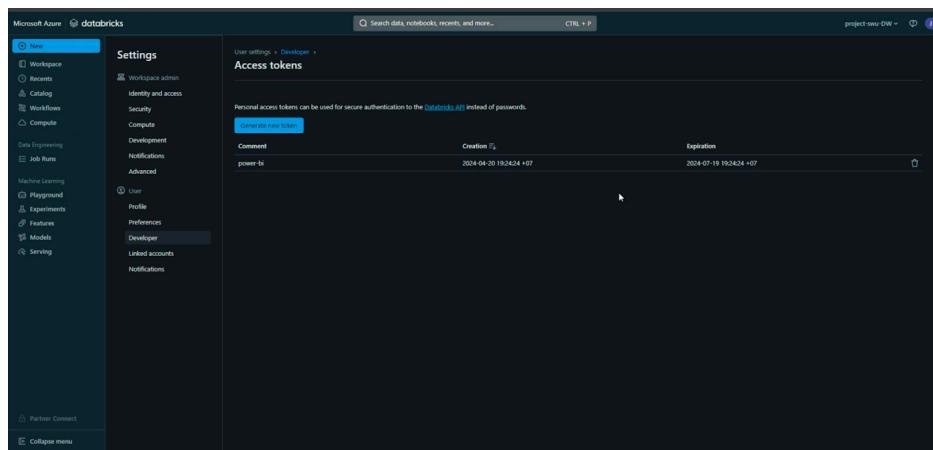
12. ทำการตั้งค่า schedule ของ task ที่สร้าง เพื่อกำหนดความถี่และช่วงเวลาที่ต้องการให้ระบบทำการ run notebook เพื่อดึงข้อมูลจาก API อัตโนมัติในทุกๆ วัน โดยสามารถตั้งค่าตามรูปแบบด้านล่างได้

Trigger Type: Scheduled

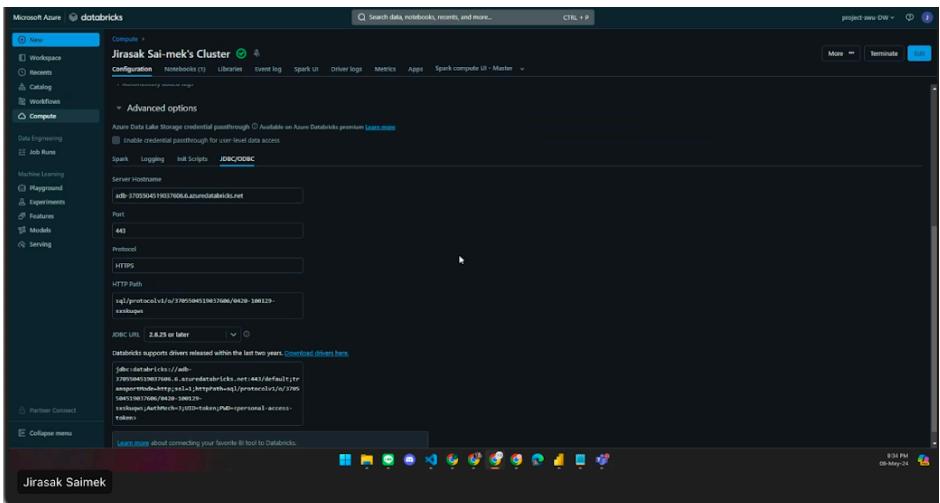
Schedule: Every Day at 07:00 (UTC+07:00 Bangkok)



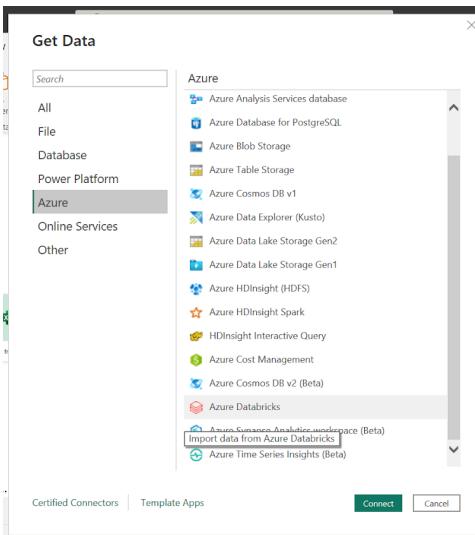
13. หลังจากสร้าง task เสร็จเรียบร้อยแล้ว ขั้นตอนถัดไปเราจะเริ่มทำเชื่อมต่อระหว่าง databricks และ power bi โดยให้เข้าไปที่ setting >> developer และทำการ generate new token เพื่อสร้าง token ในการเชื่อมต่อเข้ากับ Power BI



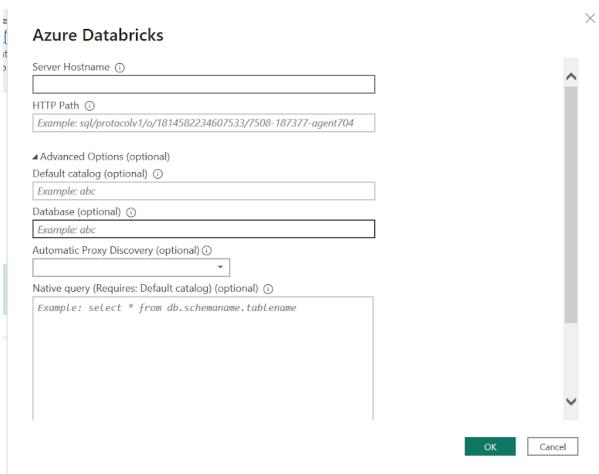
14. จากนั้นให้ไปที่ compute >> configuration >> advanced options >> JDBC/ODBC และทำการ copy server hostname และ HTTP Path



15. เมื่อสร้าง key และ copy ข้อมูลจาก databricks เรียบร้อยแล้ว ให้เข้าสู่ power bi เพื่อเริ่มทำการเชื่อมต่อ โดยเลือก get data จาก Azure >> Azure Databricks



16. เมื่อกด connect แล้ว จะกรอกข้อมูล server hostname และ HTTP Path ให้ กรอกข้อมูลที่เราได้ทำการ copy มาจาก databricks รวมถึง token ที่จะต้องกรอกในลำดับ กดไปด้วย



17. หลังจากการเชื่อมต่อเสร็จเรียบร้อยแล้ว ก็สามารถที่จะทำการสร้าง dashboard ใน power bi โดยใช้ข้อมูลที่เชื่อมต่อมาราบค์ได้