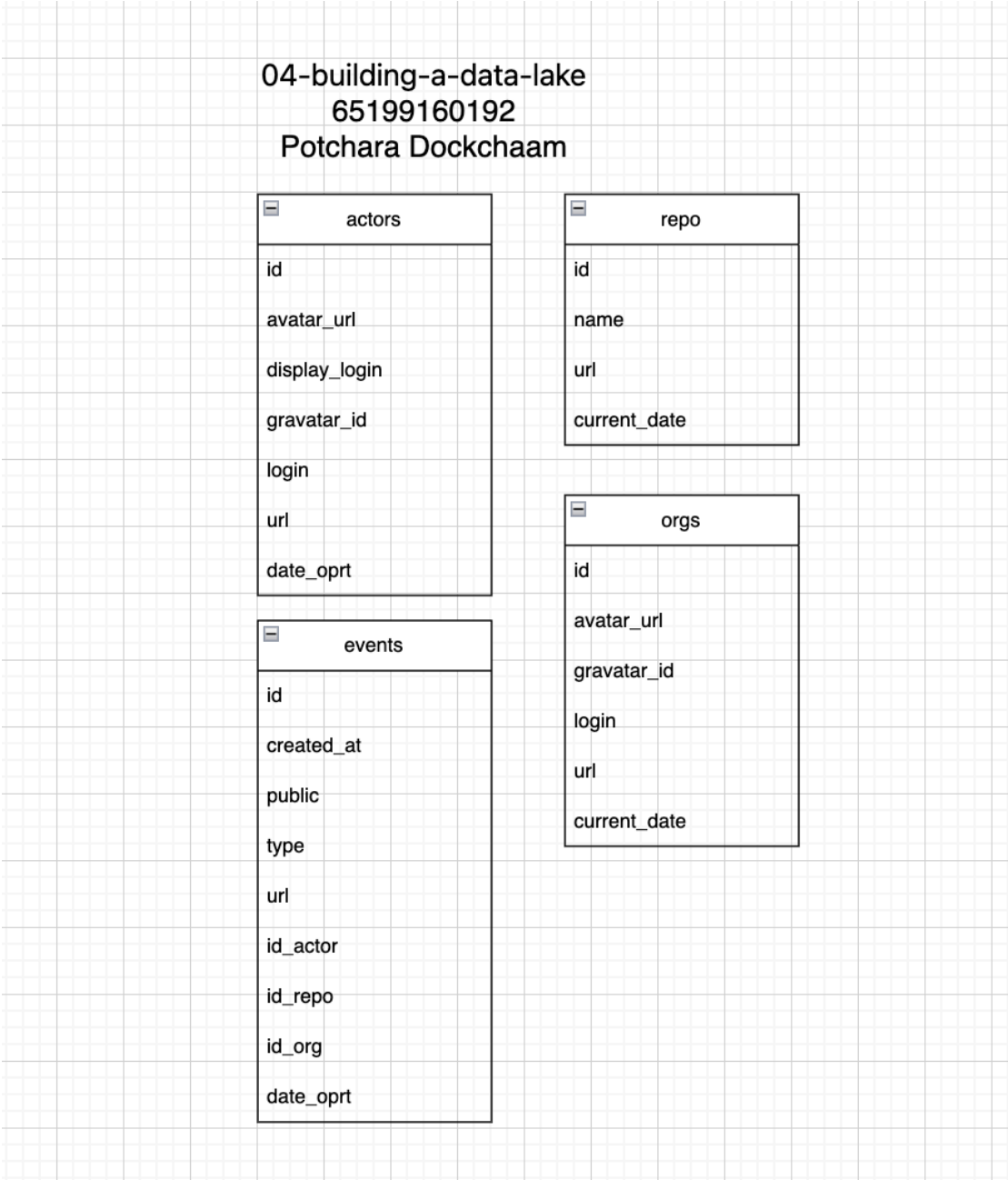


04-building-a-data-lake

65199160192

Potchara Dockchaam

Data model



## Instruction

1. เพื่อให้เราสามารถสร้างไฟล์ได้จาก Jupyter Lab ให้รันคำสั่งด้านล่างนี้  
`sudo chmod 777 .`
2. เพื่อให้สามารถเข้า Jupyter Notebook ผ่าน Port 8888:8888 ได้

`docker-compose up`

The screenshot shows a JupyterLab interface with a code cell containing the following code:

```
data.createOrReplaceTempView("staging_events")

|-- url: string (nullable = true)
|-- type: string (nullable = true)
```

Below the code cell, the output is truncated. A button labeled "data.select('id', 'type').show()" is visible. The output shows a table with columns 'id' and 'type'.

id	type
23487963576	WatchEvent
23487963624	CreateEvent
23487963529	PushEvent

At the bottom of the interface, there is a table showing the ports forwarded by Docker:

Port	Forwarded Address	Running Process	Visibility	Origin
4040	https://fantastic-s...	/usr/bin/docker-proxy -proto tcp -host-ip...	Private	Auto Forwarded
4041	https://fantastic-s...	/usr/bin/docker-proxy -proto tcp -host-ip...	Private	Auto Forwarded
8888	https://fantastic-s...	/usr/bin/docker-proxy -proto tcp -host-ip...	Private	Auto Forwarded

3. Lab ที่ใช้ชื่อ `etl_local.ipynb`

The screenshot shows a JupyterLab interface with a file explorer on the left and a code cell in the center. The file explorer shows a list of files and folders, with `etl_local.ipynb` highlighted. The code cell contains the following code:

```
ETL with Spark (Local)

[1]: from pyspark.sql import SparkSession
    # from pyspark.sql.types import StructType, StructField, DoubleType, StringType, IntegerType, DateType, TimestampType
    # import pyspark.sql.functions as F

[2]: import pandas as pd
    import glob

[3]: p = glob.glob("data/*.json")

[4]: p

[5]: data = "github_events_01.json"

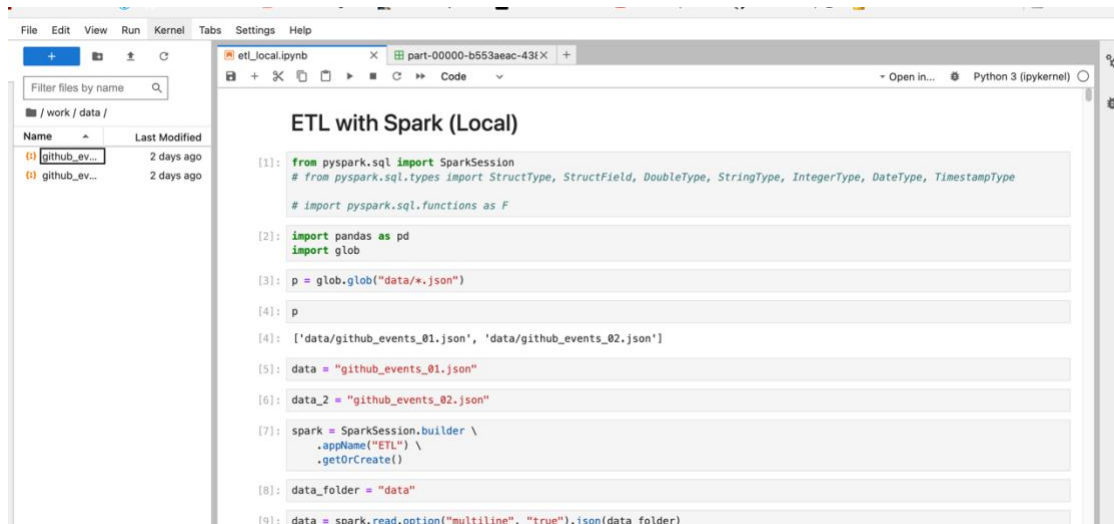
[6]: data_2 = "github_events_02.json"

[7]: spark = SparkSession.builder \
    .appName("ETL") \
    .getOrCreate()

[8]: data_folder = "data"

[9]: data = spark.read.option("multiline", "true").json(data_folder)
```

4. ข้อมูลที่จะอยู่ใน Folder data และมีไฟล์ JSON 2 ไฟล์ ซึ่งข้อมูลที่ใช้ในการ Test จะใช้เพียง data = "github\_events\_01.json"



```
[1]: from pyspark.sql import SparkSession
# from pyspark.sql.types import StructType, StructField, DoubleType, StringType, IntegerType, DateType, TimestampType
# import pyspark.sql.functions as F

[2]: import pandas as pd
import glob

[3]: p = glob.glob("data/*.json")

[4]: p

[5]: data = "github_events_01.json"

[6]: data_2 = "github_events_02.json"

[7]: spark = SparkSession.builder \
    .appName("ETL") \
    .getOrCreate()

[8]: data_folder = "data"

[9]: data = spark.read.option("multiline", "true").json(data_folder)
```

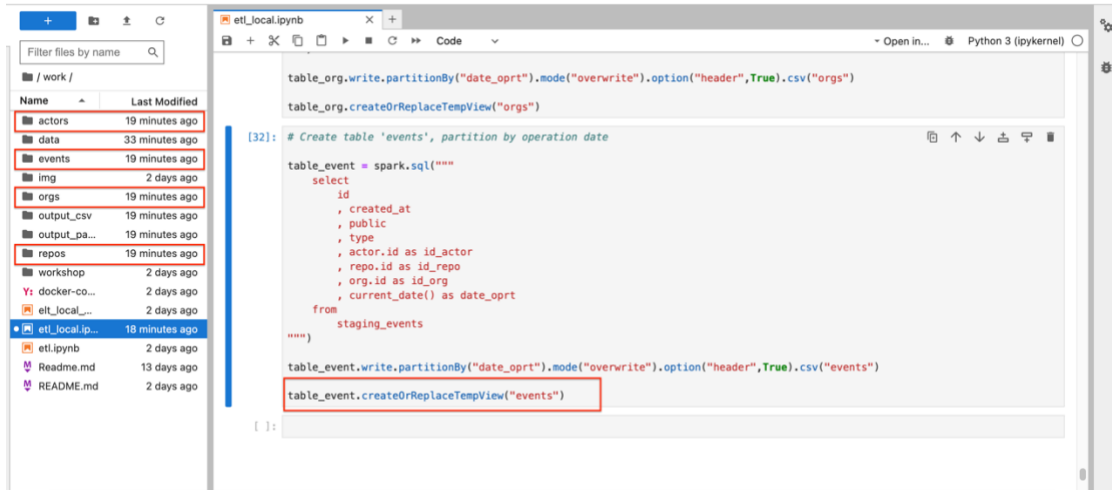
5. จากนั้นใช้ฟังก์ชัน `data.createOrReplaceTempView("staging_events")` เพื่อให้ Spark สามารถใช้ Coding ด้วย SQL ได้

```
[14]: data.createOrReplaceTempView("staging_events")
```

```
[15]: table = spark.sql("""
    select
        *

    from
        staging_events
    """).show()
```

6. จากนั้นเก็บข้อมูลให้เป็นตารางที่แบ่งข้อมูลเป็น Partition ด้วย SQL ให้สอดคล้องกับ Data model แล้วสร้าง Folder เพื่อใช้จัดเก็บไฟล์



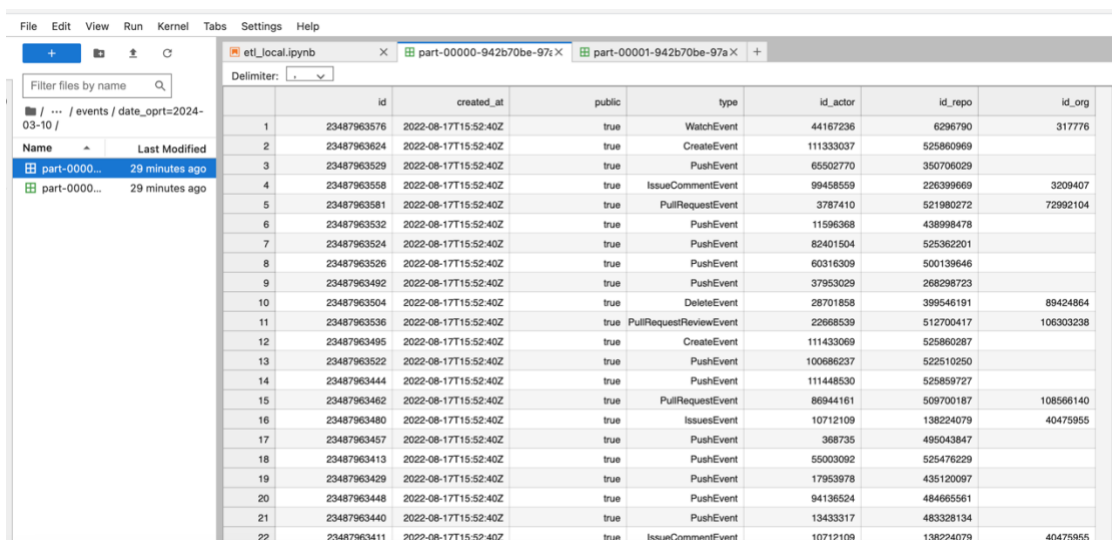
```
table_org.write.partitionBy("date_oprt").mode("overwrite").option("header",True).csv("orgs")
table_org.createOrReplaceTempView("orgs")

[32]: # Create table 'events', partition by operation date

table_event = spark.sql("""
select
    id
    , created_at
    , public
    , type
    , actor.id as id_actor
    , repo.id as id_repo
    , org.id as id_org
    , current_date() as date_oprt
from
    staging_events
""")

table_event.write.partitionBy("date_oprt").mode("overwrite").option("header",True).csv("events")
table_event.createOrReplaceTempView("events")
```

7. ข้อมูลจะถูกจัดเก็บใน Folder โดยแบ่งข้อมูลออกเป็นหลายๆส่วน ในตัวอย่างเป็นข้อมูล events ถูกแบ่งเป็นไฟล์ 2 ไฟล์



	id	created_at	public	type	id_actor	id_repo	id_org
1	23487963576	2022-08-17T15:52:40Z	true	WatchEvent	44167236	6296790	317776
2	23487963624	2022-08-17T15:52:40Z	true	CreateEvent	111333037	525860969	
3	23487963529	2022-08-17T15:52:40Z	true	PushEvent	65502770	350706029	
4	23487963558	2022-08-17T15:52:40Z	true	IssueCommentEvent	99458559	226399669	3209407
5	23487963581	2022-08-17T15:52:40Z	true	PullRequestEvent	3787410	521980272	72992104
6	23487963532	2022-08-17T15:52:40Z	true	PushEvent	11596368	438998478	
7	23487963524	2022-08-17T15:52:40Z	true	PushEvent	82401504	525362201	
8	23487963526	2022-08-17T15:52:40Z	true	PushEvent	60316309	500139646	
9	23487963492	2022-08-17T15:52:40Z	true	PushEvent	37953029	268298723	
10	23487963504	2022-08-17T15:52:40Z	true	DeleteEvent	28701858	399546191	89424864
11	23487963536	2022-08-17T15:52:40Z	true	PullRequestReviewEvent	22668539	512700417	106303238
12	23487963495	2022-08-17T15:52:40Z	true	CreateEvent	111433069	525860287	
13	23487963522	2022-08-17T15:52:40Z	true	PushEvent	100686237	522510250	
14	23487963444	2022-08-17T15:52:40Z	true	PushEvent	111448530	525859727	
15	23487963462	2022-08-17T15:52:40Z	true	PullRequestEvent	86944161	509700187	108566140
16	23487963480	2022-08-17T15:52:40Z	true	IssuesEvent	10712109	138224079	40475955
17	23487963457	2022-08-17T15:52:40Z	true	PushEvent	368735	495043847	
18	23487963413	2022-08-17T15:52:40Z	true	PushEvent	55003092	525476229	
19	23487963429	2022-08-17T15:52:40Z	true	PushEvent	17953978	435120097	
20	23487963448	2022-08-17T15:52:40Z	true	PushEvent	94136524	484665561	
21	23487963440	2022-08-17T15:52:40Z	true	PushEvent	13433317	483328134	
22	23487963411	2022-08-17T15:52:40Z	true	IssueCommentEvent	10712109	138224079	40475955