

I	L'environnement et projet à réaliser	2
I.1	Contexte.....	2
I.2	Problématique	2
I.3	Glossaire de données.....	2
I.4	Schéma d'architecture du projet à réaliser : architecture en médaillons (bronze, silver, gold)	3
I.5	Type des sources de données.....	3
I.6	Architecture en médaillons	4
I.7	Concepts clés utilisés dans le projet	4
I.8	Présentation générale de l'interface de microsoft fabric	4
I.9	Composants de Microsoft Fabric.....	5
I.10	Création d'un espace de travail pour le projet.....	5
II	Ingestion et transformation des données avec data engineering.....	6
II.1	Création du Lakehouse Bronze et chargement de fichier CSV.....	6
II.2	Traitement de la couche Bronze (Données brutes).....	8
II.3	Méthodes à utiliser :.....	9
II.3.1.1	1 ^{re} méthode : Dataflow Gen2.....	9
II.3.1.1.1	Extraction de données.....	9
II.3.1.1.2	Transformation et enrichissement de données	10
II.3.1.2	2 ^{ème} méthode : Notebook Spark python	13
II.3.1.2.1	Extraction des données	14
II.3.1.2.2	Transformation et enrichissement des données	14
II.4	Création du Lakehouse Gold dans l'architecture en médaillon.....	15
II.5	Création d'un schéma en étoile dans le Lakehouse Gold	15
II.6	Alimentation du Lakehouse (niveau Gold) et récupération des données finales	16
III	Exploration et analyse de données avec Power BI	19
III.1	Mettre en place un Modèle Sémantique (Dataset Power BI)	19
III.2	Les relations entre les tables	20
III.3	Création des rapports Power BI	22
IV	Orchestration des tâches de données avec Data factory	23
IV.1	Création du notebook: Ingestion de données quotidiennes	26
IV.2	Test du notebook d'ingestion de données quotidiennes.....	28
IV.3	Exécution du Pipeline d'orchestration.....	29
IV.4	Programmer un Pipeline Data Factory pour qu'il s'exécute automatiquement	29
IV.5	Configuration des alertes Outlook et Teams en cas d'erreur sur le Pipeline	30
IV.6	Créer une alerte à partir d'un rapport Power BI	31
IV.7	Configuration une alerte activator pour envoyer des alertes Outlook et Team	31
V	Conclusion	32

I L'environnement et projet à réaliser

I.1 Contexte

Ce projet s'inscrit dans une démarche de valorisation des données issues de capteurs installés sur trois éoliennes. Ces capteurs transmettent automatiquement, toutes les 10 minutes, des informations sur la quantité d'électricité produite. L'objectif est de mettre en place une solution automatisée avec Microsoft Fabric pour collecter, traiter, analyser et visualiser ces données, le tout dans un environnement unifié et sans intervention manuelle.

Cela permettra de mieux suivre la production, de détecter d'éventuels problèmes et de prendre de meilleures décisions.

Ce projet permet aussi d'explorer des fonctionnalités essentielles de Microsoft Fabric et de développer des compétences solides sur cet outil.

I.2 Problématique

Comment concevoir une solution de bout en bout sur Microsoft Fabric permettant d'exploiter automatiquement les données de production d'électricité des éoliennes, reçues toutes les 10 minutes, afin de faciliter leur suivi, leur traitement, leur visualisation et la détection des anomalies ?



I.3 Glossaire de données

La Base de données suivant décrit les différentes colonnes présentes dans le fichier source «wind_power_production.csv », contenant les données brutes issues des capteurs :

Voici une description simple des colonnes du fichier « wind_power_production »

- **production_id** : Identifiant unique de la production.
- **date** : Date de la production.
- **time** : Heure de la production.
- **turbine_name** : Nom de l'éolienne.
- **capacity** : Capacité maximale de l'éolienne.
- **location_name** : Nom du lieu où se trouve l'éolienne.
- **latitude** : Latitude de l'éolienne.
- **longitude** : Longitude de l'éolienne.
- **region** : Région géographique.
- **status** : État de l'éolienne (active, arrêtée, en maintenance).
- **responsible_department** : Département responsable de l'éolienne.

- **wind_speed** : Vitesse du vent au moment de la mesure.
- **wind_direction** : Direction du vent.
- **energy_produced** : Énergie produite (en kWh).

Voici un petit extrait des données sources produites par les capteurs des éoliennes.

Comme on peut le constater dans la colonne « `turbine_name` », on retrouve les trois éoliennes mentionnées précédemment: Turbine A, Turbine B et Turbine C.

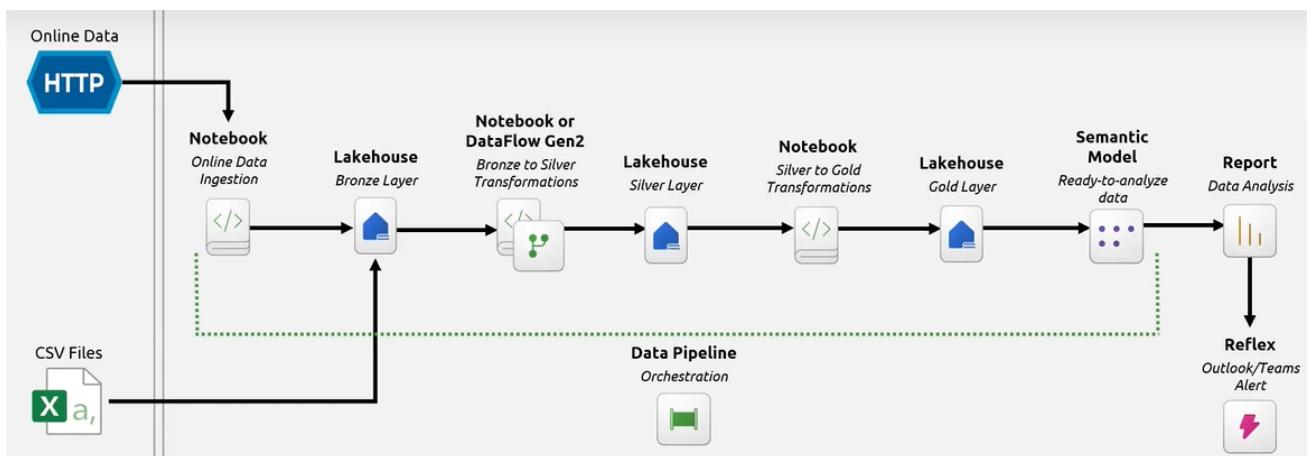
Pour chacune de ces turbines, des données sont envoyées toutes les 10 minutes.

Cela est visible dans les colonnes « `date` » (le jour d'envoi des données) et « `time` » (l'heure précise à laquelle les données ont été transmises: par exemple, à minuit, puis à 00h10, 00h20, etc.), et ce pour chaque turbine.

```
production_id,date,time,turbine_name,capacity,location_name,latitude,longitude,region,status,responsible_department,wind_speed,wind_direction,energy_produced
1,2024-06-01,00:00:00,Turbine A,2200,Location 1,34.0522,-118.2437,Region A,Online,Operations,18.44936,SE,1786.91843
2,2024-06-01,00:00:00,Turbine B,2000,Location 2,36.7783,-119.4179,Region B,Online,Operations,16.34197,E,1156.38082
3,2024-06-01,00:00:00,Turbine C,2500,Location 3,40.7128,-74.006,Region C,Online,Operations,24.16424,E,1216.49768
4,2024-06-01,00:10:00,Turbine A,2200,Location 1,34.0522,-118.2437,Region A,Online,Operations,13.03947,NE,1888.39496
5,2024-06-01,00:10:00,Turbine B,2000,Location 2,36.7783,-119.4179,Region B,Online,Operations,4.00397,SE,1852.36231
6,2024-06-01,00:10:00,Turbine C,2500,Location 3,40.7128,-74.006,Region C,Reactive Maintenance,Maintenance,5.22807,N,0.0
7,2024-06-01,00:10:00,Turbine A,2200,Location 1,34.0522,-118.2437,Region A,Reactive Maintenance,Maintenance,20.38065,SE,0.0
8,2024-06-01,00:20:00,Turbine B,2000,Location 2,36.7783,-119.4179,Region B,Online,Operations,17.61423,NW,1459.88153
9,2024-06-01,00:20:00,Turbine C,2500,Location 3,40.7128,-74.006,Region C,Online,Operations,23.72739,NW,658.86141
10,2024-06-01,00:30:00,Turbine A,2200,Location 1,34.0522,-118.2437,Region A,Online,Operations,6.28564,E,1184.2255
11,2024-06-01,00:30:00,Turbine B,2000,Location 2,36.7783,-119.4179,Region B,Online,Operations,2.43217,SW,124.5123
12,2024-06-01,00:30:00,Turbine C,2500,Location 3,40.7128,-74.006,Region C,Online,Operations,10.88925,S,1522.73045
13,2024-06-01,00:40:00,Turbine A,2200,Location 1,34.0522,-118.2437,Region A,Online,Operations,12.05173,NE,648.92053
14,2024-06-01,00:40:00,Turbine B,2000,Location 2,36.7783,-119.4179,Region B,Reactive Maintenance,Maintenance,17.02222,SE,0.0
15,2024-06-01,00:40:00,Turbine C,2500,Location 3,40.7128,-74.006,Region C,Online,Operations,4.9653,W,1411.746
16,2024-06-01,00:50:00,Turbine A,2200,Location 1,34.0522,-118.2437,Region A,Online,Operations,2.88378,SE,1982.56076
17,2024-06-01,00:50:00,Turbine B,2000,Location 2,36.7783,-119.4179,Region B,Online,Operations,6.88417,SE,1992.94935
18,2024-06-01,00:50:00,Turbine C,2500,Location 3,40.7128,-74.006,Region C,Online,Operations,11.53048,S,866.63839
19,2024-06-01,01:00:00,Turbine A,2200,Location 1,34.0522,-118.2437,Region A,Online,Operations,4.53863,SW,975.80261
20,2024-06-01,01:00:00,Turbine B,2000,Location 2,36.7783,-119.4179,Region B,Online,Operations,23.84813,NW,1731.48984
21,2024-06-01,01:00:00,Turbine C,2500,Location 3,40.7128,-74.006,Region C,Online,Operations,21.27273,E,1957.9175
22,2024-06-01,01:10:00,Turbine A,2200,Location 1,34.0522,-118.2437,Region A,Online,Operations,13.72136,N,1487.26828
23,2024-06-01,01:10:00,Turbine B,2000,Location 2,36.7783,-119.4179,Region B,Online,Operations,2.90132,E,1969.38009
24,2024-06-01,01:10:00,Turbine C,2500,Location 3,40.7128,-74.006,Region C,Online,Operations,13.06055,NE,976.97477
25,2024-06-01,01:20:00,Turbine A,2200,Location 1,34.0522,-118.2437,Region A,Online,Operations,3.47539,E,1347.7833
26,2024-06-01,01:20:00,Turbine B,2000,Location 2,36.7783,-119.4179,Region B,Online,Operations,13.70629,W,640.91077
27,2024-06-01,01:30:00,Turbine C,2500,Location 3,40.7128,-74.006,Region C,Online,Operations,23.37381,SE,514.03506
28,2024-06-01,01:30:00,Turbine A,2200,Location 1,34.0522,-118.2437,Region A,Online,Operations,16.88501,W,934.10914
29,2024-06-01,01:30:00,Turbine B,2000,Location 2,36.7783,-119.4179,Region B,Online,Operations,15.4898,N,747.41069
30,2024-06-01,01:30:00,Turbine C,2500,Location 3,40.7128,-74.006,Region C,Online,Operations,5.36816,E,985.01196
31,2024-06-01,01:40:00,Turbine A,2200,Location 1,34.0522,-118.2437,Region A,Online,Operations,24.13834,N,1634.16004
32,2024-06-01,01:40:00,Turbine B,2000,Location 2,36.7783,-119.4179,Region B,Online,Operations,22.60888,N,834.62245
33,2024-06-01,01:40:00,Turbine C,2500,Location 3,40.7128,-74.006,Region C,Reactive Maintenance,Maintenance,12.28388,S,0.0
34,2024-06-01,01:50:00,Turbine A,2200,Location 1,34.0522,-118.2437,Region A,Reactive Maintenance,Maintenance,4.67264,W,0.0
```

I.4 Schéma d'architecture du projet à réaliser: architecture en médaillons (bronze, silver, gold)

Le projet repose sur une architecture en trois couches (médaillons) pour organiser et traiter les données de manière progressive.



I.5 Type des sources de données

Nous avons deux sources principales :

- Un fichier CSV contenant l'historique des données des éoliennes (données anciennes des jours précédents). Ce fichier servira de base pour les premiers tests.

- Des données quotidiennes extraites d'une page web, représentant les nouvelles données générées chaque jour par les éoliennes. Un nouveau fichier y est disponible chaque jour.

I.6 Architecture en médaillons

- **Bronze:** Stockage des données brutes, issues directement de la page web et du fichier CSV, sans modification.
- **Silver:** Stockage des données nettoyées et enrichies à partir de la couche Bronze.
- **Gold:** Stockage des données prêtes à être analysées et utilisées dans les rapports.

I.7 Concepts clés utilisés dans le projet

- Lakehouse: C'est une couche de Microsoft Fabric permettant de stocker les données comme dans une base de données, tout en profitant de la flexibilité du stockage de type data lake.
- Notebook: Outil permettant d'écrire et d'exécuter du code (Python, SQL, R ou Scala) pour manipuler, transformer ou analyser les données.
- Dataflow Gen2: Une méthode sans code (no-code) pour créer des flux de données, transformer les données et les charger dans une destination cible.
- Modèle sémantique: Modèle en étoile construit à partir des données de la couche Gold. Il est conçu pour faciliter l'analyse et la création de rapports visuels dans Power BI.
- Rapport: Tableau de bord ou visualisation permettant de suivre la production quotidienne des éoliennes.
- Reflex: Mécanisme d'alerte automatique. Il permet de recevoir des notifications (par exemple via Microsoft Teams, Outlook ou e-mail) si certaines données dépassent des seuils définis, comme une baisse anormale de la production électrique.
- Data Pipeline: Outil d'orchestration qui automatise l'ensemble du processus. Il prend en charge:
 - La récupération quotidienne des données depuis la page web,
 - Le traitement automatique à travers les couches Bronze, Silver et Gold,
 - La mise à jour des rapports,
 - L'envoi d'alertes si nécessaire.

Tous les éléments du projet, du premier notebook jusqu'aux alertes, seront intégrés dans un pipeline de données qui s'exécute de manière automatisée. Par exemple, on peut la configurer pour qu'elle s'exécute tous les jours à 8h du matin, sans intervention manuelle.

I.8 Présentation générale de l'interface de microsoft fabric

Microsoft Fabric est une plateforme tout-en-un qui permet de stocker, traiter et analyser des données, le tout dans un environnement centralisé.

C'est un outil incontournable pour les data scientists, data engineers, data analysts et les équipes travaillant autour de la data factory, car il répond aux besoins croissants des entreprises qui gèrent de grands volumes de données et souhaitent les exploiter pleinement.

Son principal avantage: il permet aux différents métiers de la data de collaborer sur une seule et même plateforme.

C'est une solution très intéressante, qui permet de couvrir l'ensemble des besoins en matière de données, que ce soit pour le stockage, le traitement, l'analyse, etc.

The screenshot shows the Microsoft Fabric Home page. At the top, it says "Microsoft Fabric" and "All your data. In one location. Organize. Collaborate. Create." Below this, there's a section titled "Explore the experience" with eight cards:

- Power BI**: Find insights, track progress, and make decisions faster using rich visualizations.
- Data Factory**: Solve complex data ingestion, transformation, and orchestration scenarios using cloud-scale data movement and data transformation services.
- Data Activator**: Detect patterns and conditions in your Power BI reports and streaming data, and then take actions such as alert users or kick-off workflows.
- Industry Solutions**: Use out-of-the-box industry data solutions and resources.
- Real-Time Intelligence**: Discover insights from your streaming data. Quickly ingest, index, and partition any data source or format, then query the data and create visualizations. You can also create alerts to flag anomalies.
- Synapse Data Engineering**: Create a lakehouse and operationalize your workflow to build, transform, and share your data estate.
- Synapse Data Science**: Unlock powerful insights using AI and machine learning technology.
- Synapse Data Warehouse**: Scale up your insights by storing and analyzing data in a secure SQL warehouse. Benefit from top-tier performance at petabyte scale in an open-data format.

At the bottom, there are two buttons: "Read documentation" and "Explore community".

I.9 Composants de Microsoft Fabric

Microsoft Fabric propose plusieurs expériences, c'est-à-dire des espaces de travail adaptés à chaque métier de la data. Chacune a un rôle précis. Voici les principales:

Data Factory:

C'est l'expérience pour créer des pipelines de données (chargement, transformation, automatisation). Elle remplace un peu Azure Data Factory, mais dans Fabric.

Synapse Data Engineering:

C'est là qu'on peut écrire du code Spark, SQL, Scala, R pour faire des traitements avancés sur les données. Utilisé par les data engineers.

Synapse Data Science:

Permet aux data scientists de faire des analyses poussées avec Python ou R, créer des modèles de machine learning. Tout est intégré dans Fabric.

Synapse Data Warehousing:

Sert à créer un entrepôt de données (Data Warehouse) performant. Parfait pour stocker et interroger de grandes quantités de données.

Real-Time Analytics:

Pour analyser des données en temps réel (comme des logs, etc.).

Data Activator:

Permet de créer des alertes automatiques ou des actions quand certaines données changent. Exemple: envoyer un e-mail si un seuil est dépassé comme pour le mini-projet.

Power BI:

C'est l'espace pour créer des rapports interactifs et des tableaux de bord. Très utilisé par les data analysts et les métiers.

I.10 Crédation d'un espace de travail pour le projet

Un espace de travail est un répertoire dans lequel seront stockés tous les éléments que nous allons créer sur Microsoft Fabric, comme par exemple des rapports, des notebooks, des modèles sémantiques, etc.

Pour créer un espace de travail, on peut se rendre dans n'importe quelle expérience, puis accéder à la section Espace de travail (Workspace), cliquer sur nouveau workspace (New workspace) et donner un nom, par exemple WindProductionGeneration. Les étapes sont présentées ci-dessous

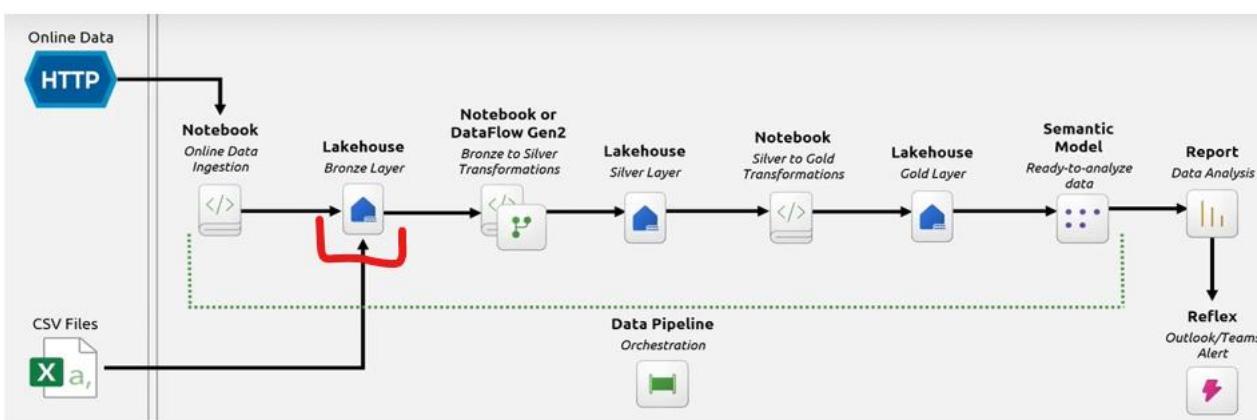
Étape 1: Saisir le nom de l'espace de travail.

Ex : WindProductionGeneration

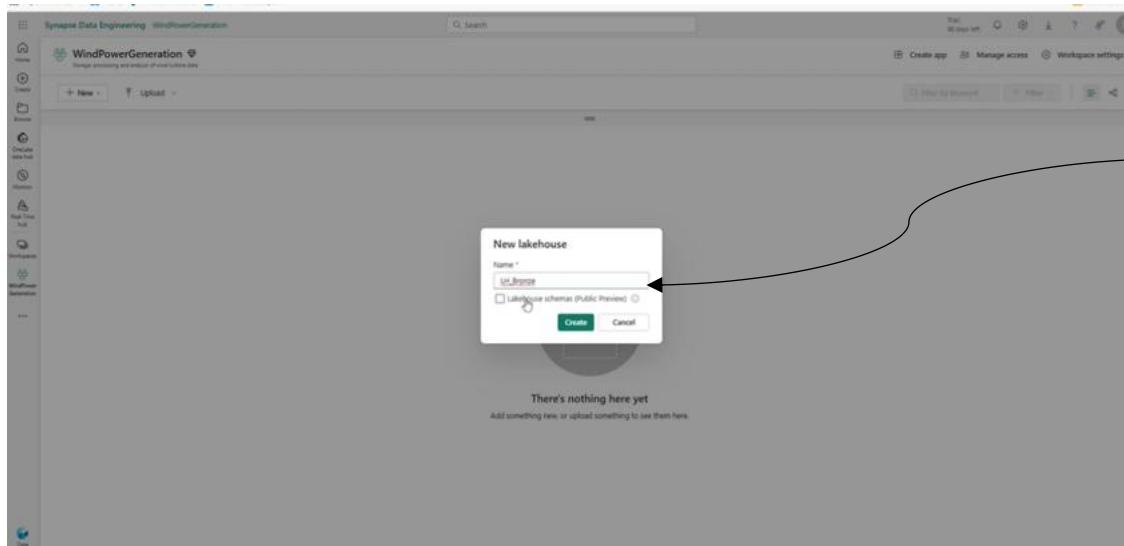
Étape 2 : Aperçu de l'interface de l'espace de travail

II Ingestion et transformation des données avec data engineering

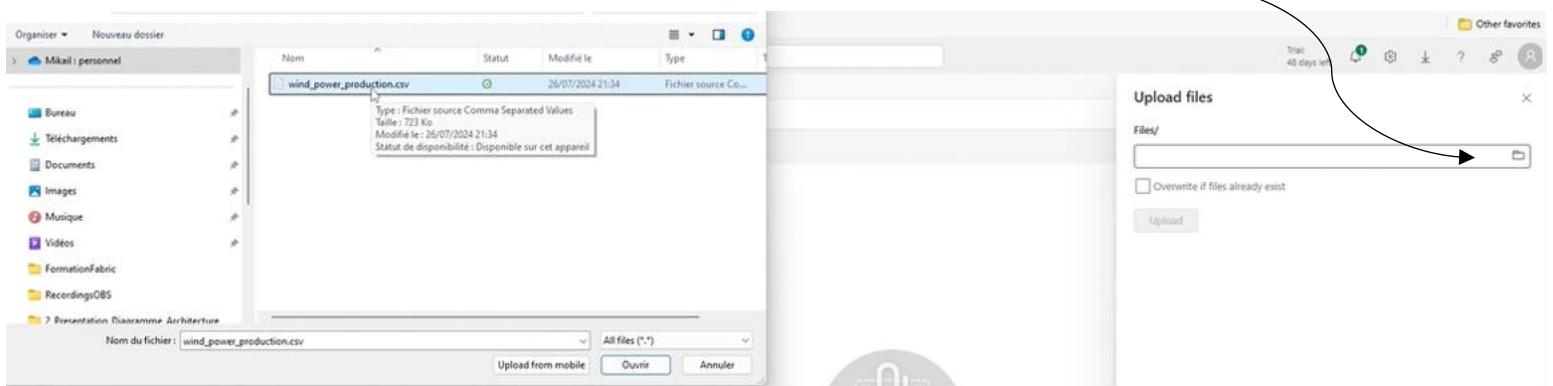
II.1 Crédation du Lakehouse Bronze et chargement de fichier CSV



Stockage des données brutes, issues directement de la page web et des fichiers CSV, sans aucune modification. Les étapes sont présentées ci-dessous.



Étape1 : Création du Lakehouse « LH_Bronze »



Étape2 : Importer le fichier CSV

File	Content Preview
wind_power_production.csv	1 production_id,data_center,turbine_name,capacity,location_name,latitude,longitude,region,status,responsible_department,wind_speed,wind_direction,energy_produced 2,1,2024-06-01,00:00:00,Turbine A,2200,Location 1,34.0522,-119.2437,Region A,Online,Operations,16.44936,M,1786.91843 3,2,2024-06-01,00:00:00,Turbine B,2000,Location 2,36.7783,-119.4179,Region B,Online,Operations,16.34197,E,1156.38082 4,3,2024-06-01,00:00:00,Turbine C,2500,Location 3,40.7128,-74.0066,Region C,Online,Operations,24.16424,E,1216.49768 5,4,2024-06-01,00:00:00,Turbine A,2200,Location 1,34.0522,-119.2437,Region A,Online,Operations,13.03947,N,1888.39496 6,5,2024-06-01,00:00:00,Turbine B,2000,Location 2,36.7783,-119.4179,Region B,Online,Operations,4.00397,S,1192.36231 7,6,2024-06-01,00:00:00,Turbine C,2500,Location 3,40.7128,-74.0066,Region C,Online,Operations,13.03947,N,1888.39496 8,7,2024-06-01,00:20:00,Turbine A,2200,Location 1,34.0522,-119.2437,Region A,Reactive Maintenance,Maintenance,20.38065,SE,B,0.0 9,8,2024-06-01,00:20:00,Turbine B,2000,Location 2,36.7783,-119.4179,Region B,Online,Operations,17.63423,M,1459.88155 10,9,2024-06-01,00:20:00,Turbine C,2500,Location 3,40.7128,-74.0066,Region C,Online,Operations,23.72739,M,658.86141 11,10,2024-06-01,01:00:00,Turbine A,2200,Location 1,34.0522,-119.2437,Region A,Online,Operations,13.03947,N,1888.39496 12,11,2024-06-01,01:00:00,Turbine B,2000,Location 2,36.7783,-119.4179,Region B,Online,Operations,2.43217,N,1512.73043 13,12,2024-06-01,01:00:00,Turbine C,2500,Location 3,40.7128,-74.0066,Region C,Online,Operations,10.89517,M,1522.73045 14,13,2024-06-01,01:00:00,Turbine A,2200,Location 1,34.0522,-119.2437,Region A,Online,Operations,12.05173,N,648.50893 15,14,2024-06-01,01:00:00,Turbine B,2000,Location 2,36.7783,-119.4179,Region B,Online,Operations,17.02222,SE,0.0 16,15,2024-06-01,01:00:00,Turbine C,2500,Location 3,40.7128,-74.0066,Region C,Online,Operations,4.9653,M,1411.746 17,16,2024-06-01,01:00:00,Turbine A,2200,Location 1,34.0522,-119.2437,Region A,Online,Operations,13.03947,N,1888.39496 18,17,2024-06-01,01:00:00,Turbine B,2000,Location 2,36.7783,-119.4179,Region B,Online,Operations,13.03947,N,1888.39496 19,18,2024-06-01,01:00:00,Turbine C,2500,Location 3,40.7128,-74.0066,Region C,Online,Operations,13.03947,N,1888.39496 20,19,2024-06-01,01:00:00,Turbine A,2200,Location 1,34.0522,-119.2437,Region A,Online,Operations,23.72739,M,658.86141 21,20,2024-06-01,01:00:00,Turbine B,2000,Location 2,36.7783,-119.4179,Region B,Online,Operations,23.84013,M,1731.48908 22,21,2024-06-01,01:00:00,Turbine C,2500,Location 3,40.7128,-74.0066,Region C,Online,Operations,23.72739,M,658.86141 23,22,2024-06-01,01:00:00,Turbine A,2200,Location 1,34.0522,-119.2437,Region A,Online,Operations,2.43217,N,1512.73043 24,23,2024-06-01,01:00:00,Turbine B,2000,Location 2,36.7783,-119.4179,Region B,Online,Operations,2.00132,E,1069.38869 25,24,2024-06-01,01:00:00,Turbine C,2500,Location 3,40.7128,-74.0066,Region C,Online,Operations,13.0695,M,976.97477 26,25,2024-06-01,01:20:00,Turbine A,2200,Location 1,34.0522,-119.2437,Region A,Online,Operations,3.47539,E,1347.7833 27,26,2024-06-01,01:20:00,Turbine B,2000,Location 2,36.7783,-119.4179,Region B,Online,Operations,13.03947,N,1888.39496 28,27,2024-06-01,01:20:00,Turbine C,2500,Location 3,40.7128,-74.0066,Region C,Online,Operations,13.03947,N,1888.39496 29,28,2024-06-01,01:20:00,Turbine A,2200,Location 1,34.0522,-119.2437,Region A,Online,Operations,16.08591,N,934.10914 30,29,2024-06-01,01:20:00,Turbine B,2000,Location 2,36.7783,-119.4179,Region B,Online,Operations,15.48917,H,747.42069 31,30,2024-06-01,01:20:00,Turbine C,2500,Location 3,40.7128,-74.0066,Region C,Online,Operations,5.36816,E,985.01196 32,31,2024-06-01,01:20:00,Turbine A,2200,Location 1,34.0522,-119.2437,Region A,Online,Operations,24.13634,M,1634.16604 33,32,2024-06-01,01:20:00,Turbine B,2000,Location 2,36.7783,-119.4179,Region B,Online,Operations,12.92136,M,1054.05902 34,33,2024-06-01,01:20:00,Turbine C,2500,Location 3,40.7128,-74.0066,Region C,Reactive Maintenance,Maintenance,12.23186,S,0.0 35,34,2024-06-01,01:50:00,Turbine A,2200,Location 1,34.0522,-119.2437,Region A,Reactive Maintenance,Maintenance,4.67264,M,0.0 36,35,2024-06-01,01:50:00,Turbine B,2000,Location 2,36.7783,-119.4179,Region B,Online,Operations,11.13963,M,874.69443 37,36,2024-06-01,01:50:00,Turbine C,2500,Location 3,40.7128,-74.0066,Region C,Online,Operations,24.20457,N,1587.88142 38,37,2024-06-01,02:00:00,Turbine A,2200,Location 1,34.0522,-119.2437,Region A,Online,Operations,23.98992,M,1091.30994 39,38,2024-06-01,02:00:00,Turbine B,2000,Location 2,36.7783,-119.4179,Region B,Online,Operations,12.92136,M,1054.05902 40,39,2024-06-01,02:00:00,Turbine C,2500,Location 3,40.7128,-74.0066,Region C,Online,Operations,8.6764,M,1634.16604 41,40,2024-06-01,02:00:00,Turbine A,2200,Location 1,34.0522,-119.2437,Region A,Online,Operations,4.36566,M,1354.94737

Étape 3 : Voici le fichier CSV chargé dans le Lakehouse « LH_Bronze »

A screenshot of the Microsoft Power BI Data Explorer interface. On the left, there's a sidebar with various icons for Home, Create, Browse, OneLake data hub, Monitor, Real-Time hub, Workspaces, WindPower Generation, and LH_Bronze. The main area shows a folder 'LH_Bronze' containing 'Tables' and 'Files'. Under 'Files', there's a file named 'wind_power_production.csv'. To the right, a modal dialog box titled 'Load file to new table' is displayed. It asks for a 'New table name' (set to 'wind_power_produced') and a 'Separator' (set to ';'). There are also checkboxes for 'Column header' (unchecked) and 'Use header for column names' (checked). At the bottom of the dialog are 'Load' and 'Cancel' buttons.

Étape 4:
Conversion du fichier CSV en table requetable

Étape 5: Le fichier CSV a été converti en table interrogeable

A screenshot of the Microsoft Fabric Data Explorer interface. The top navigation bar shows 'LH_Bronze - Synapse Data Engine' and a URL. The sidebar on the left is identical to the one in the previous screenshot. The main area shows the 'wind_power_production' table under 'LH_Bronze / Tables'. The table has 14 columns: wind_production_id, date, AM_time, AM_turbine_name, capacity, location_name, latitude, longitude, region, status, responsible, wind_speed, wind_direction, and energy_usage. The table contains 1000 rows of data, with the first few rows visible. A message at the bottom says 'Succeeded (21 sec 239 ms)'.

II.2 Traitement de la couche Bronze (Données brutes)

Après la création de la couche Bronze, on peut constater que les données ne sont pas encore dans un état idéal.

Par exemple :

- Dans la colonne « Time », les heures, minutes et secondes sont séparées par des tirets, alors qu'en général on utilise des deux points (ex.: 00-10-00 au lieu de 00:10:00).
- Dans la colonne « wind_speed », on remarque plusieurs décimales. Il serait plus lisible d'arrondir ces valeurs à deux chiffres après la virgule.
- De la même manière, d'autres colonnes numériques peuvent nécessiter un arrondi ou un formatage cohérent.

Pour cela, un nettoyage et un enrichissement des données sont nécessaires, avant de les stocker dans le Lakehouse Silver.

Pour cela, 2 méthodes courantes pour effectuer ces traitements.

NB: Ces 2 méthodes mènent au même objectif: transformer et enrichir les données de la couche Bronze et les préparer pour la couche Silver.

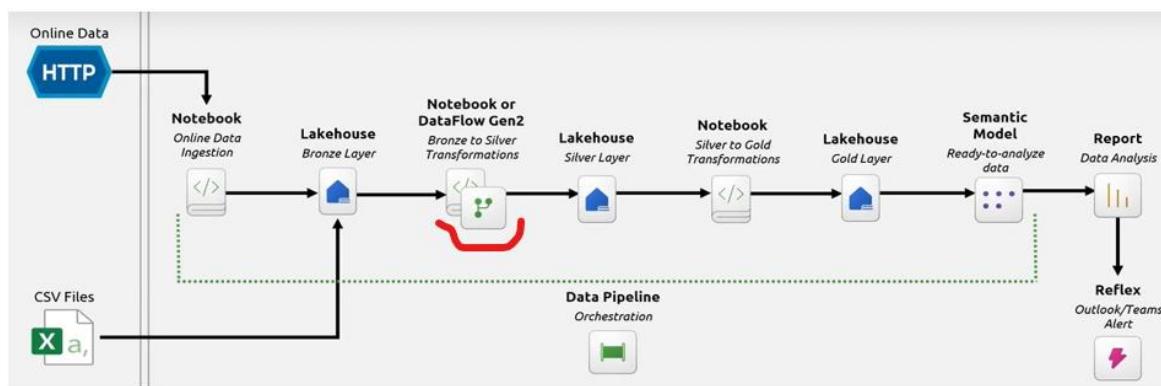
II.3 Méthodes à utiliser :

- Dataflow Gen2: méthode sans code (no-code) pour créer un flux de données visuellement.
- Notebook (Spark Python): permet d'écrire du code pour transformer les données.

II.3.1.1 1^{re} méthode: Dataflow Gen2

C'est une méthode no-code, simple et visuelle, qui permet:

- De connecter la source de données (Bronze),
- D'appliquer des transformations (nettoyage, enrichissement, formatage),
- Puis de charger les données transformées dans une nouvelle destination (le Lakehouse Silver).



II.3.1.1.1 Extraction de données

Les étapes sont présentées ci-dessous

Étape 1: Extraction des données de la couche Bronze (« LH-Bronze »)

L'interface utilisateur de Dataflow Gen2 affiche les options pour l'extraction de données. On peut voir des boutons pour Import from Excel, Import from SQL Server, Import from a Text/CSV file et Import from dataflows. Des liens hypertextes permettent d'accéder à d'autres sources de données et à des templates Power Query.

Aperçu de l'interface utilisateur de Dataflow

Étape 2: Choisir le « OneLake Data Hub »

Étape 4: Sélectionner la table nommée « wind_power_production »

II.3.1.1.2 Transformation et enrichissement de données

Les étapes sont présentées ci-dessous :

Étape 2: Sélectionner « Transform »

Étape 3: Sélectionner « Round »

Étape1: Arrondir les colonnes « wind_speed » et « energy_produced » à deux chiffres après la virgule.

Étape 3: Convertir les colonnes Day, Month, Quarter et Year du type décimal au type entier

The screenshot shows the Power Query Editor interface with the 'wind_power_production' data source. The 'Transform' tab is active. In the main area, there is a table with various columns like 'location_name', 'latitude', 'longitude', etc. A callout from the top right points to the 'Data type' dropdown for the 'Year' column, which is currently set to 'Decimal number'. Another callout from the bottom right points to the 'Replace values...' button.

Étape 4: Remplace le tiret (-) de la colonne time par le deux-points (:) (édition de la colonne)

The screenshot shows the Power Query Editor interface with the 'wind_power_production' data source. The 'Transform' tab is active. A callout from the top right points to the 'Replace values...' dialog box, which is open over the main table area. The 'Value to find' field contains '-' and the 'Replace with' field contains ':'.

Étape 5: Extraire la colonne time en colonnes séparées (Heure, Minute et Seconde)

The screenshot shows the Power Query Editor interface with the 'wind_power_production' data source. The 'Transform' tab is active. A callout from the top right points to the 'Split column' dialog box, which is open over the main table area. The 'Separator' dropdown is set to 'Comma'.

Étape 6: Création d'une colonne calculée indiquant si c'est le matin (Morning), l'après-midi (Afternoon), le soir (Evening) ou la nuit (Night)

The screenshot shows the Power Query Editor interface. A modal dialog titled "Custom column" is open, prompting the user to add a column computed from other columns or values. The new column name is set to "time_period" and its data type is "Text". The formula is defined as:

```
= if [hour_of_day] >= 5 and [hour_of_day] < 12 then "Morning"
else if [hour_of_day] >= 12 and [hour_of_day] < 17 then "Afternoon"
else if [hour_of_day] >= 17 and [hour_of_day] < 21 then "Evening"
else "Night"
```

The available column(s) dropdown lists "hour_of_day" and other date/time columns like "month", "quarter", "year", "minute_of_hour", and "second_of_minute". The "Insert column" button is visible at the bottom right of the dialog.

Une fois les données modifiées, transformées et enrichies, nous les charge ensuite dans le Lakehouse Silver.

The screenshot shows the Power Query Editor after transformation. The table now includes a new column "time_period" with values corresponding to the time of day. The completed query is named "wind_power_production". The "Applied steps" pane on the right shows the history of transformations applied to the data.

	status	responsible_department	wind_speed	wind_direction	energy_produced	day	month	quarter	year	hour_of_day	minute_of_hour	second_of_minute	time_period
1	Fault	Engineering	11.21	SE	0	1	6	2	2024	21	40	0	Night
2	Fault	Engineering	22.14	NE	0	2	6	2	2024	7	0	0	Night
3	Fault	Engineering	11.48	SE	0	2	6	2	2024	10	50	0	Morning
4	Fault	Engineering	2.03	SW	0	4	6	2	2024	8	50	0	Morning
5	Fault	Engineering	18.72	E	0	4	6	2	2024	11	50	0	Morning
6	Fault	Engineering	13.6	E	0	5	6	2	2024	20	0	0	Evening
7	Fault	Engineering	17.31	NE	0	6	6	2	2024	2	10	0	Night
8	Fault	Engineering	14.12	NW	0	7	6	2	2024	19	40	0	Evening
9	Fault	Engineering	13.01	E	0	9	6	2	2024	0	50	0	Night
10	Fault	Engineering	22.07	N	0	10	6	2	2024	3	50	0	Night
11	Fault	Engineering	13.74	SE	0	12	6	2	2024	14	50	0	Afternoon
12	Fault	Engineering	11.71	S	0	13	6	2	2024	2	20	0	Night
13	Fault	Engineering	2.73	W	0	13	6	2	2024	11	30	0	Morning
14	Fault	Engineering	15.82	E	0	13	6	2	2024	17	0	0	Evening
15	Fault	Engineering	17.2	NW	0	13	6	2	2024	20	0	0	Evening
16	Fault	Engineering	3.03	SW	0	14	6	2	2024	0	50	0	Night
17	Fault	Engineering	16.4	SE	0	14	6	2	2024	23	10	0	Night
18	Fault	Engineering	19.46	W	0	1	6	2	2024	11	50	0	Morning
19	Fault	Engineering	23.3	SE	0	2	6	2	2024	0	20	0	Night
20	Fault	Engineering	18.69	NE	0	3	6	2	2024	7	20	0	Morning
21	Fault	Engineering	16.85	SE	0	3	6	2	2024	19	40	0	Evening
22	Fault	Engineering	7.91	NE	0	4	6	2	2024	8	0	0	Morning
23	Fault	Engineering	6.61	N	0	5	6	2	2024	14	30	0	Afternoon
24	Fault	Engineering	11.89	SW	0	6	6	2	2024	13	0	0	Night
25	Fault	Engineering	3.61	SE	0	7	6	2	2024	0	30	0	Night
26	Fault	Engineering	4.18	SE	0	7	6	2	2024	10	50	0	Morning

Étape 7: Paramétrage de la destination

Power Query Dataflow saved

Choose destination target

New table Existing table

Display options

Lakehouse My workspace WindPowerGeneration [3] DataflowsStagingLakehouse... LH_Bronze LH_Silver

A new table will be created in LH_Silver

Table name * wind_power_production

Back Cancel Next

Étape 8: Les données sont chargées dans le Lakehouse Silver via le Dataflow Gen2.

Synapse Data Engineering WindPowerGeneration

WindPowerGeneration Storage, processing and analysis of wind turbine data

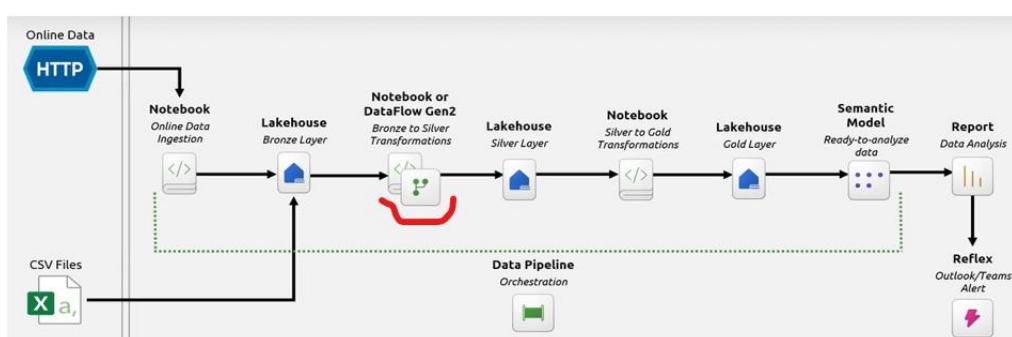
+ New + Upload

Name	Type	Task
DFL_Bronze_To_Silver_Transformations	Dataflow Gen2	—
LH_Bronze	Lakehouse	—
LH_Bronze	Semantic model (d...)	—
LH_Bronze	SQL analytics endp...	—
LH_Silver	Lakehouse	—
LH_Silver	Semantic model (d...)	—
LH_Silver	SQL analytics endp...	—

II.3.1.2 2^{ème} méthode: Notebook Spark python

Les étapes sont présentées ci-dessous:

Ce notebook effectue des transformations sur les données du lakehouse Bronze et enregistre les données de transformation dans le lakehouse Silver.



II.3.1.2.1 Extraction des données

Les étapes sont présentées ci-dessous:

Ce notebook effectue des transformations sur les données du lakehouse Bronze et enregistre les données de transformation dans le lakehouse Silver.

Transformations du bronze en Silver

```
# Importation des fonctions nécessaires pour le projet  
from pyspark.sql.functions import round, col, dayofmonth, month, year, to_date, quarter, substring, when, regexp_replace
```

→ **Étape 1:** Importation des fonctions nécessaires

```
# Définis le chemin d'accès à la table wind_power_production dans Lakehouse Bronze  
bronze_table_path = "abfss://WindPowerGeneration@onelake.dfs.fabric.microsoft.com/LH_Bronze.Lakehouse/Tables/wind_power_production"
```

→ **Étape 2:** Récupére le chemin d'accès à la table dans le Lakehouse Bronze

```
# Charge la table wind_power_production dans un DataFrame spark  
df = spark.read.format("delta").load(bronze_table_path)
```

→ **Étape 3:** Charge la table dans un DataFrame

II.3.1.2.2 Transformation et enrichissement des données

Les étapes sont présentées ci-dessous:

```
# Nettoyage et enrichissement des données  
df_transformed = (df  
    .withColumn("wind_speed", round(col("wind_speed"), 2))  
    .withColumn("energy_produced", round(col("energy_produced"), 2))  
    .withColumn("day", dayofmonth(col("date")))  
    .withColumn("month", month(col("date")))  
    .withColumn("quarter", quarter(col("date")))  
    .withColumn("year", year(col("date")))  
    .withColumn("time", regexp_replace(col("time"), "-", ":"))  
    .withColumn("hour_of_day", substring(col("time"), 1, 2).cast("int"))  
    .withColumn("minute_of_hour", substring(col("time"), 4, 2).cast("int"))  
    .withColumn("second_of_minute", substring(col("time"), 7, 2).cast("int"))  
    .withColumn("time_period", when((col("hour_of_day") >= 5) & (col("hour_of_day") < 12), "Morning")  
        .when((col("hour_of_day") >= 12) & (col("hour_of_day") < 17), "Afternoon")  
        .when((col("hour_of_day") >= 17) & (col("hour_of_day") < 21), "Evening")  
        .otherwise("Night"))  
)
```

→ **Étape 4:** Nettoyage et enrichissement des données :

Arrondis les colonnes *wind_speed* et *energy_produced* à deux chiffres après la virgule.

Ajoute les colonnes *Day*, *Month*, *Quarter* et *Year* à partir de la colonne *date*.

Remplace le tiret (-) par un deux-points (:) dans la colonne *time*.

Convertis le type de données des colonnes *Day*, *Month*, *Quarter* et *Year* de décimal à entier.

Crée une colonne calculée indiquant si l'enregistrement correspond au matin (Morning), à l'après-midi (Afternoon), au soir (Evening) ou à la nuit (Night).

```
#Spécifie le chemin vers la table wind_power_production située dans le Lakehouse (niveau Silver)  
silver_table_path = "abfss://WindPowerGeneration@onelake.dfs.fabric.microsoft.com/LH_Silver.Lakehouse/Tables/wind_power_production"
```

→ **Étape 5:** Définis le chemin d'accès à la table « *wind_power_production* » dans le Lakehouse niveau Silver

```
#Sauvegarde la version transformée de la table wind_power_production dans le Lakehouse (niveau Silver)  
df_transformed.write.format("delta").mode("overwrite").save(silver_table_path)
```

→ **Étape 6:** Charge les données dans le Lakehouse Silver en utilisant la méthode « **overwrite** », qui écrase les anciennes données et les remplace par de nouvelles.

A SQL analytics endpoint for SQL querying and a default Power BI semantic model for reporting were created with this item.

Showing 1000 rows

Columns 22 Rows 1,000

Explorer

wind_power_production

LH_Silver

Tables

wind_power_production

Files

WindPower Generation

LH_Silver

NB_Bronze_to_Silver_...

NB_Amonite_to_Silver_...

Data Engineering

Succeeded (5 sec 623 ms)

ABC responsible...	ABC wind_direc...	12 wind_speed	12 energy_produ...	12 day	12 month	12 quarter	12 year	ABC time	12 hour_of_day	12 minute_of_h...	12 second_of_h...	ABC time_period
Engineering	SE	11.21	0	1	1	2	2024	21:40:00	21	40	0	Night
Engineering	NE	23.14	0	2	6	2	2024	01:00:00	1	0	0	Morning
Engineering	SE	11.48	0	2	6	2	2024	10:50:00	10	50	0	Morning
Engineering	SW	2.03	0	4	6	2	2024	08:50:00	8	50	0	Morning
Engineering	E	18.72	0	4	6	2	2024	11:50:00	11	50	0	Morning
Engineering	E	13.6	0	5	6	2	2024	20:00:00	20	0	0	Soir
Engineering	NE	17.31	0	6	6	2	2024	02:10:00	2	10	0	Night
Engineering	NW	14.12	0	7	6	2	2024	19:40:00	19	40	0	Soir
Engineering	E	13.01	0	9	6	2	2024	00:50:00	0	50	0	Night
Engineering	N	22.07	0	10	6	2	2024	02:50:00	2	50	0	Night
Engineering	SE	12.74	0	12	6	2	2024	14:50:00	14	50	0	Afternoon
Engineering	S	11.71	0	13	6	2	2024	02:20:00	2	20	0	Night
Engineering	W	2.73	0	13	6	2	2024	11:30:00	11	30	0	Morning
Engineering	E	15.82	0	13	6	2	2024	17:00:00	17	0	0	Soir
Engineering	NW	17.2	0	13	6	2	2024	20:00:00	20	0	0	Soir
Engineering	SW	3.03	0	14	6	2	2024	00:50:00	0	50	0	Night
Engineering	SE	16.4	0	14	6	2	2024	23:10:00	23	10	0	Night
Engineering	W	19.46	0	1	6	2	2024	11:50:00	11	50	0	Morning
Engineering	SE	23.3	0	2	6	2	2024	00:20:00	0	20	0	Night
Engineering	NE	18.69	0	3	6	2	2024	07:20:00	7	20	0	Morning
Engineering	SE	16.85	0	3	6	2	2024	19:40:00	19	40	0	Soir
Engineering	NE	7.91	0	4	6	2	2024	08:00:00	8	0	0	Morning
Engineering	N	6.61	0	5	6	2	2024	14:30:00	14	30	0	Afternoon
Engineering	SW	11.89	0	6	6	2	2024	13:00:00	13	0	0	Afternoon
Engineering	SE	3.61	0	7	6	2	2024	00:30:00	0	30	0	Night
Engineering	E	4.18	0	7	6	2	2024	10:50:00	10	50	0	Morning

Voici l'aperçu des données nettoyées et enrichies chargées dans le Lakehouse niveau Silver

II.4 Crédit du Lakehouse Gold dans l'architecture en médaillon.

Ensuite, on va créer un notebook pour faire la transformation entre la couche Silver et la couche Gold. Il faudra aussi créer le Lakehouse Gold. Une fois cette couche prête, on pourra créer un rapport Power BI et mettre en place toute notre architecture avec un pipeline Data Factory.

Synapse Data Engineering WindPowerGeneration

WindPowerGeneration Storage, processing and analysis of wind turbine data

+ New Upload

Folder (preview)

Data pipeline

Dataflow Gen2

Environment

Eventhouse

Eventstream

Experiment

KQL Queryset

Lakehouse

ML model

Notebook

Reflex (Preview)

Report

Spark Job Definition

Warehouse

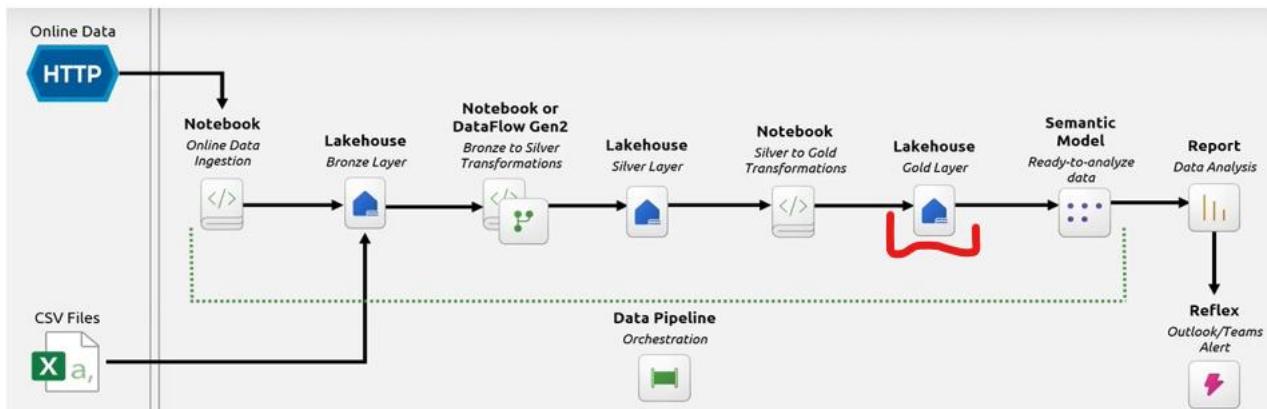
More options

Import item Import notebook

WindPower Generation

Création de Lakehouse Gold

II.5 Crédit d'un schéma en étoile dans le Lakehouse Gold



Si on revient sur le Lakehouse Silver, il contient toutes nos données. C'est ce qu'on appelle un modèle « big table », une grande table avec toutes les informations. Cette table va continuer à grandir, car on reçoit trois nouvelles lignes toutes les 10 minutes, soit 18 lignes par heure, donc 432 lignes par jour.

Quand on travaille avec une table très grande, avec beaucoup de lignes et de colonnes, ça peut poser des problèmes, surtout pour les mises à jour.

La solution, c'est de normaliser la table, c'est-à-dire la découper en plusieurs petites tables plus simples, avec moins de données dans chacune. Ça rend le tout plus facile à maintenir, à analyser et à interroger.

L'objectif est donc de passer de ce modèle à un schéma en étoile, plus adapté pour l'analyse.

1-Modèle dans le
Lakehouse Silver

2-Schéma en étoile

wind_power_production
production_id
date
time
turbine_name
capacity
location_name
latitude
longitude
region
status
responsible_department
wind_speed
wind_direction
energy_produced
day
month
quarter
year
hour_of_day
minute_of_hour
second_of_minute
time_period

dim_time
time_id
hour_of_day
minute_of_hour
second_of_minute
time_period

dim_turbine
turbine_id
turbine_name
capacity
location_name
latitude
longitude
region

fact_wind_power_production
production_id
date_id
time_id
turbine_id
status_id
wind_speed
wind_direction
energy_produced

dim_date
date_id
day
month
quarter
year

dim_operational_status
status_id
status
responsible_department

II.6 Alimentation du Lakehouse (niveau Gold) et récupération des données finales

Les étapes sont présentées ci-dessous :

Transformations de Silver en Gold

Ce Notebook effectue des transformations sur les données du Lakehouse Silver et enregistre les données transformées dans le Lakehouse Gold.

```
from pyspark.sql.window import Window
from pyspark.sql.functions import row_number

# Définissez le chemin d'accès à la table wind_power_production dans Silver Lakehouse
silver_table_path = "abfss://WindPowerGeneration@onelake.dfs.fabric.microsoft.com/LH_Silver.Lakehouse/Tables/wind_power_production"

# Charger la table wind_power_production
df = spark.read.format("delta").load(silver_table_path)
```

```
# Crée la table de dimension 'Date'
date_dim = df.select("date", "day", "month", "quarter", "year").distinct() \
    .withColumnRenamed("date", "date_id")
```

	date	day	month	quarter	year
1	2024-06-01	1	6	2	2024
2	2024-06-02	2	6	2	2024
3	2024-06-03	3	6	2	2024
4	2024-06-04	4	6	2	2024
5	2024-06-05	5	6	2	2024
6	2024-06-06	6	6	2	2024
7	2024-06-07	7	6	2	2024
8	2024-06-08	8	6	2	2024
9	2024-06-09	9	6	2	2024
10	2024-06-10	10	6	2	2024
11	2024-06-11	11	6	2	2024
12	2024-06-12	12	6	2	2024
13	2024-06-13	13	6	2	2024
14	2024-06-14	14	6	2	2024

```
# Crée la table de dimension 'Time'
time_dim = df.select("time", "hour_of_day", "minute_of_hour", "second_of_minute", "time_period").distinct() \
    .withColumnRenamed("time", "time_id")
```

	time_id	hour_of_day	minute_of_hour	second_of_minute	time_period
1	00:00:00	0	0	0	Night
2	00:10:00	0	10	0	Night
3	00:20:00	0	20	0	Night
4	00:30:00	0	30	0	Night
5	00:40:00	0	40	0	Night
6	00:50:00	0	50	0	Night
7	01:00:00	1	0	0	Night
8	01:10:00	1	10	0	Night
9	01:20:00	1	20	0	Night
10	01:30:00	1	30	0	Night
11	01:40:00	1	40	0	Night
12	01:50:00	1	50	0	Night
13	02:00:00	2	0	0	Night
14	02:10:00	2	10	0	Night
15	02:20:00	2	20	0	Night
16	02:30:00	2	30	0	Night

```
# Crée la table de dimension des turbines 'turbine_dim'
turbine_dim = df.select("turbine_name", "capacity", "location_name", "latitude", "longitude", "region").distinct() \
    .withColumn("turbine_id", row_number().over(Window.orderBy("turbine_name", "capacity", "location_name", "latitude", "longitude", "region")))
```

	turbine_name	capacity	location_name	latitude	longitude	region
1	Turbine A	2200	Location 1	34.0522	-118.2437	Region A
2	Turbine C	2500	Location 3	40.7128	-74.006	Region C
3	Turbine B	2000	Location 2	36.7783	-119.4179	Region B

```
# Crée la table de dimension Statut opérationnel'operational_status_dim'
operational_status_dim = df.select("status", "responsible_department").distinct() \
    .withColumn("status_id", row_number().over(Window.orderBy("status", "responsible_department")))
```

	status	responsible_department	status_id
1	Fault	Engineering	1
2	Inspection	Maintenance	2
3	Offline	Operations	3
4	Online	Operations	4
5	Preventive ...	Maintenance	5
6	Reactive M...	Maintenance	6

```
# Fais la jointure entre les tables de dimension turbine_dim et operational_status_dim avec le DataFrame d'origine
df = df.join(turbine_dim, ["turbine_name", "capacity", "location_name", "latitude", "longitude", "region"], "left") \
    .join(operational_status_dim, ["status", "responsible_department"], "left")
```

Table	Chart	Download	Showing rows 1 - 1000	Inspect	Search																
12	energy_produced	123	day	123	month	123	quarter	123	year	123	hour_of_day	123	minute_of_hour	123	second_of_minute	abc	time_period	123	turbine_id	123	status_id
0.0	1	6	2	2024	21	40	0	Night	3	1											
0.0	2	6	2	2024	1	0	0	Night	3	1											
0.0	2	6	2	2024	10	50	0	Morning	3	1											
0.0	4	6	2	2024	8	50	0	Morning	3	1											
0.0	4	6	2	2024	11	50	0	Morning	3	1											
0.0	5	6	2	2024	20	0	0	Evening	3	1											
0.0	6	6	2	2024	2	10	0	Night	3	1											
0.0	7	6	2	2024	19	40	0	Evening	3	1											
0.0	9	6	2	2024	0	50	0	Night	3	1											
0.0	10	6	2	2024	2	50	0	Night	3	1											
0.0	12	6	2	2024	14	50	0	Afternoon	3	1											
0.0	13	6	2	2024	2	20	0	Night	3	1											
0.0	13	6	2	2024	11	30	0	Morning	3	1											
0.0	13	6	2	2024	17	0	0	Evening	3	1											
0.0	13	6	2	2024	20	0	0	Evening	3	1											
0.0	14	6	2	2024	0	50	0	Night	3	1											

```
# Crée la table de faits
```

```
fact_table = df.select("production_id", "date", "time", "turbine_id", "status_id", "wind_speed", "wind_direction", "energy_produced") \
    .withColumnRenamed("date", "date_id").withColumnRenamed("time", "time_id")
```

Table	Chart	Download	Showing rows 1 - 1000						
	123 production_id	date	ABC time	123 turbine_id	123 status_id	12 wind_speed	ABC wind_direction	12 energy_produced	
1	393	2024-06-01	21:40:00	3	1	11.21	SE	0.0	
2	453	2024-06-02	01:00:00	3	1	22.14	NE	0.0	
3	630	2024-06-02	10:50:00	3	1	11.48	SE	0.0	
4	1458	2024-06-04	08:50:00	3	1	2.03	SW	0.0	
5	1512	2024-06-04	11:50:00	3	1	18.72	E	0.0	
6	2091	2024-06-05	20:00:00	3	1	13.6	E	0.0	
7	2202	2024-06-06	02:10:00	3	1	17.31	NE	0.0	
8	2949	2024-06-07	19:40:00	3	1	14.12	NW	0.0	
9	3474	2024-06-09	00:50:00	3	1	13.01	E	0.0	
10	3942	2024-06-10	02:50:00	3	1	22.07	N	0.0	
11	5022	2024-06-12	14:50:00	3	1	13.74	SE	0.0	
12	5229	2024-06-13	02:20:00	3	1	11.71	S	0.0	
13	5394	2024-06-13	11:30:00	3	1	2.73	W	0.0	
14	5493	2024-06-13	17:00:00	3	1	15.82	E	0.0	
15	5547	2024-06-13	20:00:00	3	1	17.2	NW	0.0	
16	5634	2024-06-14	00:50:00	3	1	3.03	SW	0.0	

Définis les chemins d'accès aux tables situées dans le Lakehouse, niveau Gold

```
# Définis les chemins d'accès aux tables dans le Lakehouse Gold
```

```
gold_date_dim_path = "abfss://WindPowerGeneration@onelake.dfs.fabric.microsoft.com/LH_Gold.Lakehouse/Tables/dim_date"
gold_time_dim_path = "abfss://WindPowerGeneration@onelake.dfs.fabric.microsoft.com/LH_Gold.Lakehouse/Tables/dim_time"
gold_turbine_dim_path = "abfss://WindPowerGeneration@onelake.dfs.fabric.microsoft.com/LH_Gold.Lakehouse/Tables/dim_turbine"
gold_operational_status_dim_path = "abfss://WindPowerGeneration@onelake.dfs.fabric.microsoft.com/LH_Gold.Lakehouse/Tables/dim_operational_status"
gold_fact_table_path = "abfss://WindPowerGeneration@onelake.dfs.fabric.microsoft.com/LH_Gold.Lakehouse/Tables/fact_wind_power_production"
```

- Enregistre les tables dans le lakehouse Gold

Sauvegarde les tables dans le Lakehouse Gold

```
# Sauvegarde les tables dans le Lakehouse Gold
date_dim.write.format("delta").mode("overwrite").save(gold_date_dim_path)
time_dim.write.format("delta").mode("overwrite").save(gold_time_dim_path)
turbine_dim.write.format("delta").mode("overwrite").save(gold_turbine_dim_path)
operational_status_dim.write.format("delta").mode("overwrite").save(gold_operational_status_dim_path)
fact_table.write.format("delta").mode("overwrite").save(gold_fact_table_path)
```

Toutes les tables sont chargées dans le lakehouse Gold, donc toutes les données qu'on va pouvoir utiliser pour faire nos rapports Power BI, mettre en place des alertes

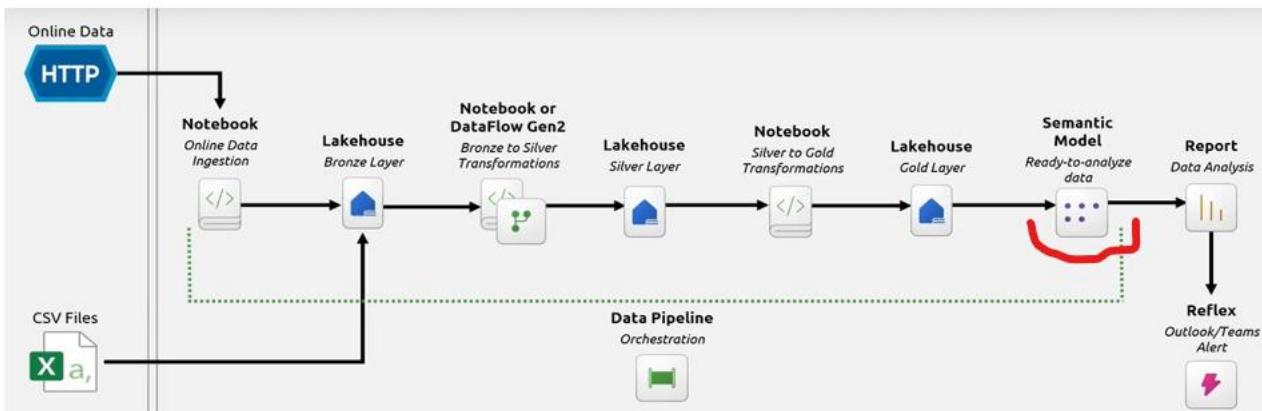
Toutes les tables de dimensions et de fait

Showing 1000 rows

	127 production_id	date_id	ABC_time_id	127 turbine_id	127 status_id	12 wind_speed	ABC wind_direction	12 energy_produced
1	1029	6/3/2024 12:00:00 AM	09:00:00	3	4	9.62	NW	1997.51
2	4485	6/11/2024 12:00:00 AM	09:00:00	3	4	14.93	NW	1995.42
3	4065	6/10/2024 12:00:00 AM	09:40:00	3	4	3.72	NW	1993.55
4	5328	6/13/2024 12:00:00 AM	07:50:00	3	4	12.66	NW	1992.13
5	2049	6/5/2024 12:00:00 AM	17:40:00	3	4	7.32	NW	1990.58
6	3069	6/8/2024 12:00:00 AM	02:20:00	3	4	18.93	NW	1988.4
7	3414	6/8/2024 12:00:00 AM	21:30:00	3	4	5.43	W	1988.29
8	1947	6/5/2024 12:00:00 AM	12:00:00	3	4	3.95	NW	1987.2
9	4356	6/11/2024 12:00:00 AM	01:50:00	3	4	10.07	W	1984.88
10	4470	6/11/2024 12:00:00 AM	08:10:00	3	4	15.75	W	1984.38
11	4263	6/10/2024 12:00:00 AM	20:40:00	3	4	22.58	NW	1983.17
12	2262	6/6/2024 12:00:00 AM	05:30:00	3	4	23.21	NW	1981.1
13	4956	6/12/2024 12:00:00 AM	11:10:00	3	4	16.36	NW	1980.02
14	1410	6/4/2024 12:00:00 AM	06:10:00	3	4	16.37	W	1976.82
15	5484	6/13/2024 12:00:00 AM	16:30:00	3	4	24	W	1975.61
16	3324	6/8/2024 12:00:00 AM	16:30:00	3	4	10.19	NW	1970.61
17	468	6/2/2024 12:00:00 AM	01:50:00	3	4	10.95	W	1966.52
18	2484	6/6/2024 12:00:00 AM	17:50:00	3	4	19.51	S	1950.2
19	5040	6/12/2024 12:00:00 AM	15:50:00	3	4	3.5	NW	1938.84
20	975	6/3/2024 12:00:00 AM	06:00:00	3	4	4.25	W	1938.53
21	861	6/2/2024 12:00:00 AM	23:40:00	3	4	16.2	W	1927.45
22	5715	6/14/2024 12:00:00 AM	05:20:00	3	4	6.52	W	1918.73

III Exploration et analyse de données avec Power BI

III.1 Mettre en place un Modèle Sémantique (Dataset Power BI)

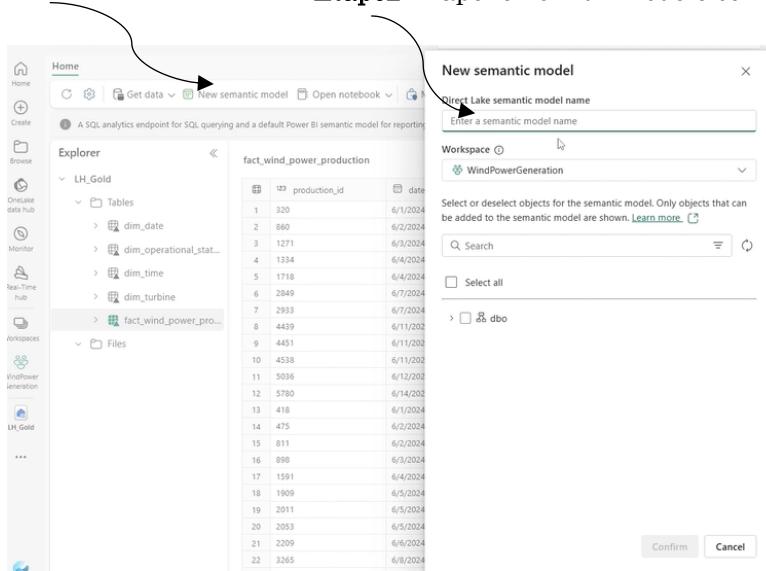


Maintenant, les données finales sont contenues dans le Lakehouse Gold, on va pouvoir créer nos rapports Power BI.

Le modèle sémantique c'est une couche qui fait l'intermédiaire entre le Lakehouse Gold et les rapports. C'est dans cette couche que l'on va définir les relations entre les différentes tables dans le Lakehouse Gold pour que les tables soient reliées entre dans les rapports Power BI.

Les étapes sont présentées ci-dessous :

Étape 1 : Nouveau modèle sémantique



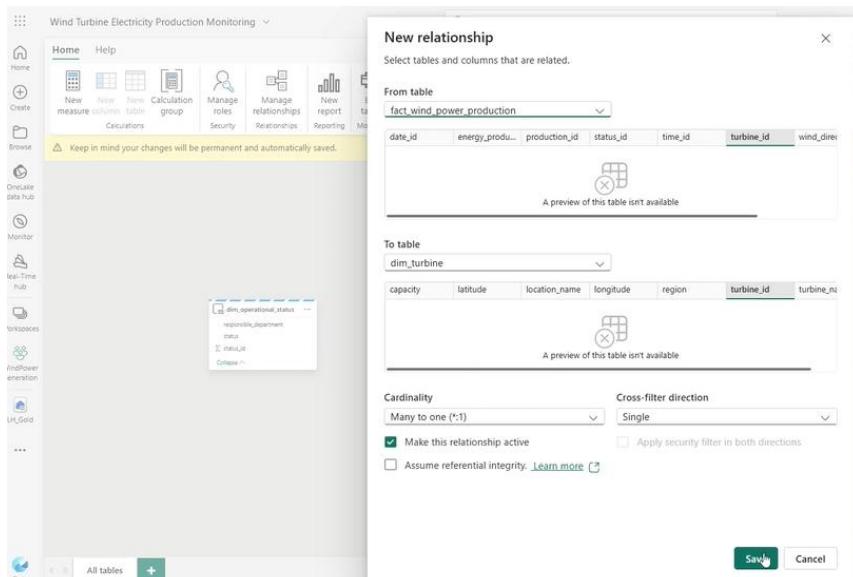
Étape2 : Tape le nom du modèle sémantique

Lakehouse Share

Showing 1000 rows

s_id	12_wind_speed	ABC_wind_direction	12_energy_produced
17.44	NE	0	
22.36	N	0	
22.34	S	0	
20.74	N	0	
19.38	S	0	
5.76	W	0	
17.98	SW	0	
5.13	S	0	
13.82	E	0	
3.67	NE	0	
12.43	SE	0	
24.24	N	0	
3.85	SW	0	
6.57	SW	0	
9.11	W	0	
3.32	SW	0	
12.65	SE	0	
4.25	E	0	
8.5	SE	0	
4.27	SW	0	
12.89	SE	0	
11.72	NE	0	

III.2 Les relations entre les tables



Définis les relations entre les tables de dimensions et de faits, ainsi que les cardinalités.

Par exemple :
la relation entre *dim_turbine* et
fait_wind_power_production.
Cardinalité : (1:N).

Tables des dimensions

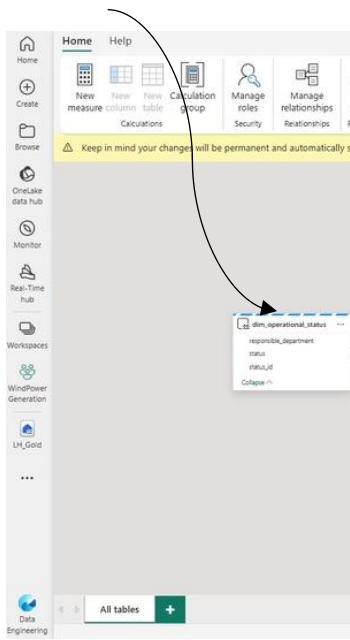
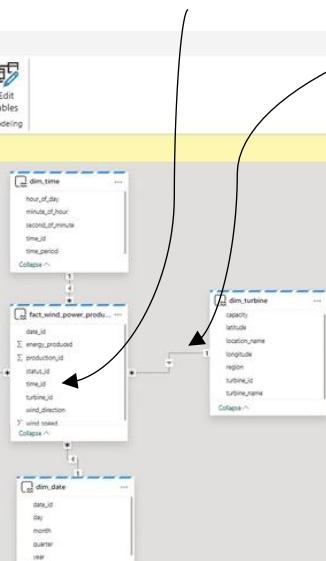
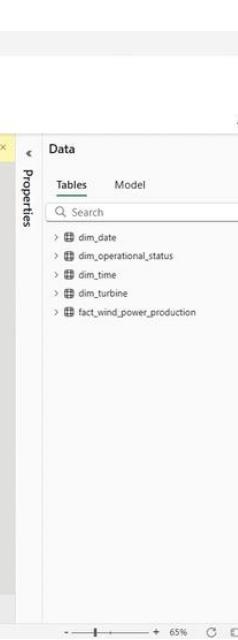


Table de Fait

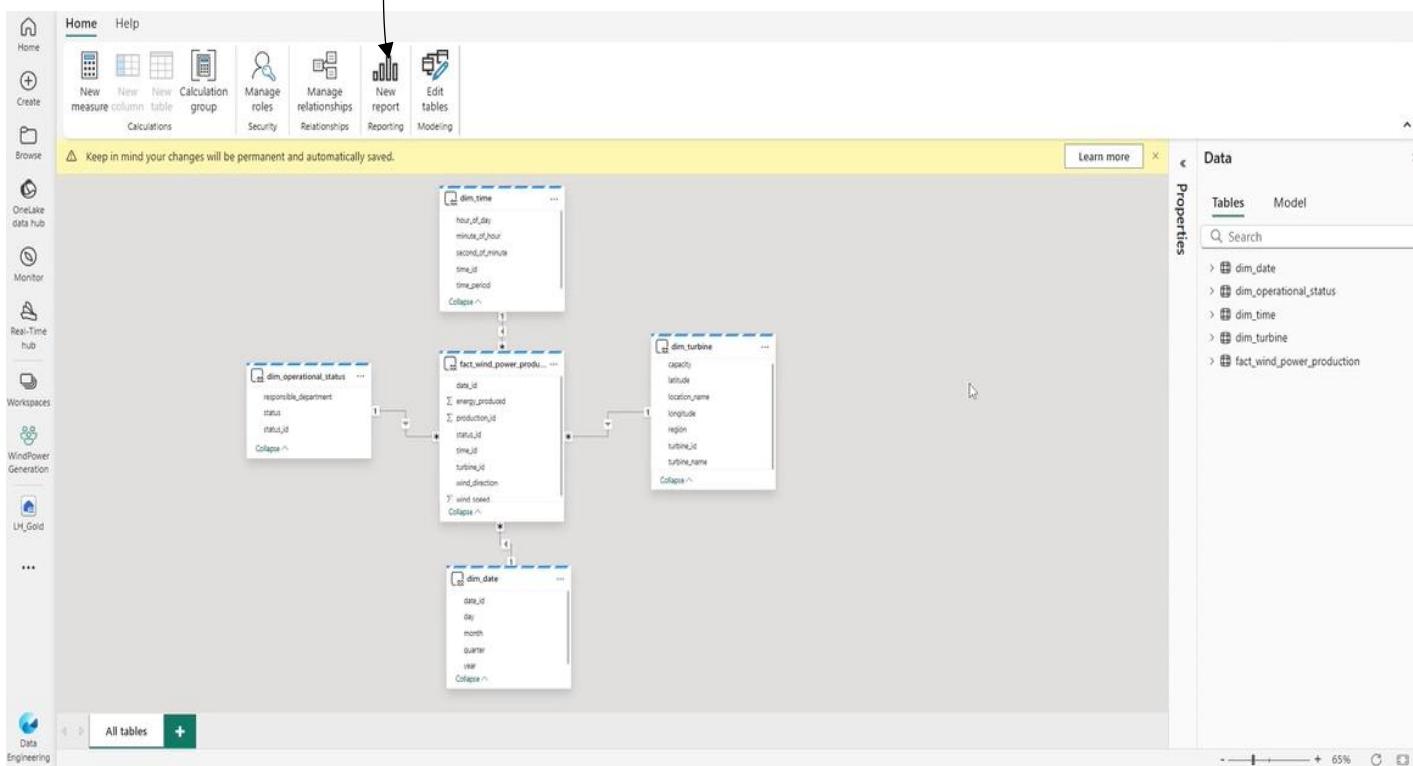


Les relations



Le modèle sémantique est prêt, on peut maintenant créer les rapports Power BI.

Choisir « New report » dans le volet Reporting



L'interface power BI dans Fabric

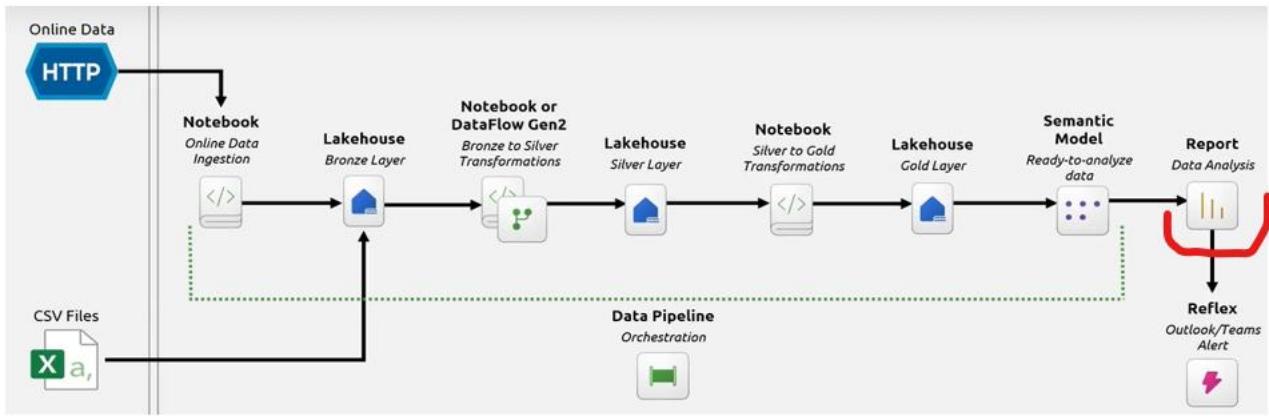
Bouton pour sauvegarder le rapport

Toutes les tables

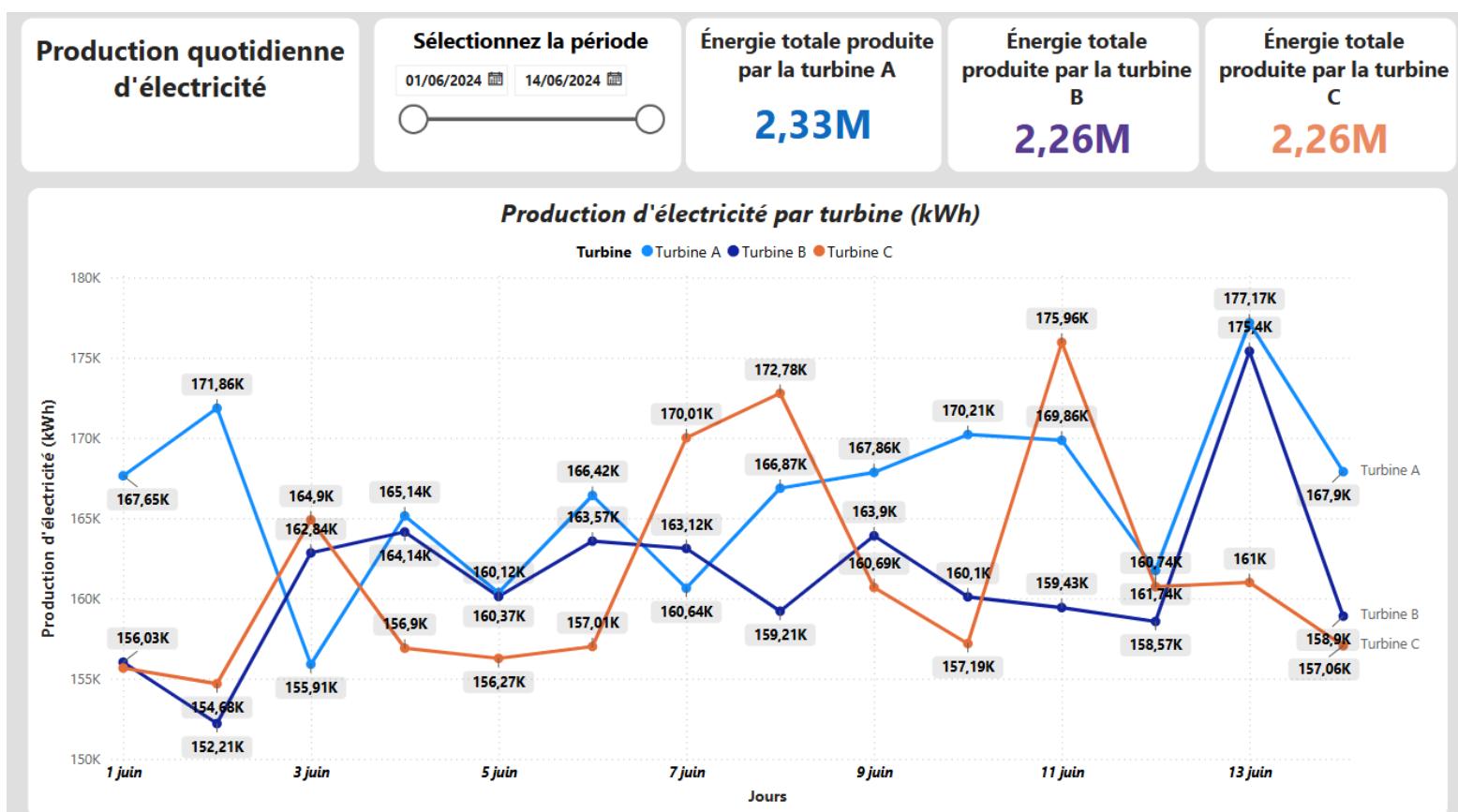
The screenshot shows the Power BI Synapse Data Engineering interface. On the left, there's a navigation bar with Home, Create, Browse, OneLake data hub, Monitor, Real-Time hub, Workspaces, WindPower Generation, and Untitled report. The main area has tabs for Synapse Data Engineering and WindPowerGeneration. It features a "Build visuals with your data" section where users can select or drag fields from the Data pane onto the report canvas. The Data pane on the right contains sections for Filters, Visualizations, and Data. The Data section lists various tables: dim_date, dim_operational_status, dim_time, dim_turbine, and fact_wind_power_production. Arrows point from the text labels to specific parts of the interface: one arrow points to the "Build visuals with your data" section, another to the "Data" button in the top right, and a third to the "Data" section in the Data pane.

Les visuels

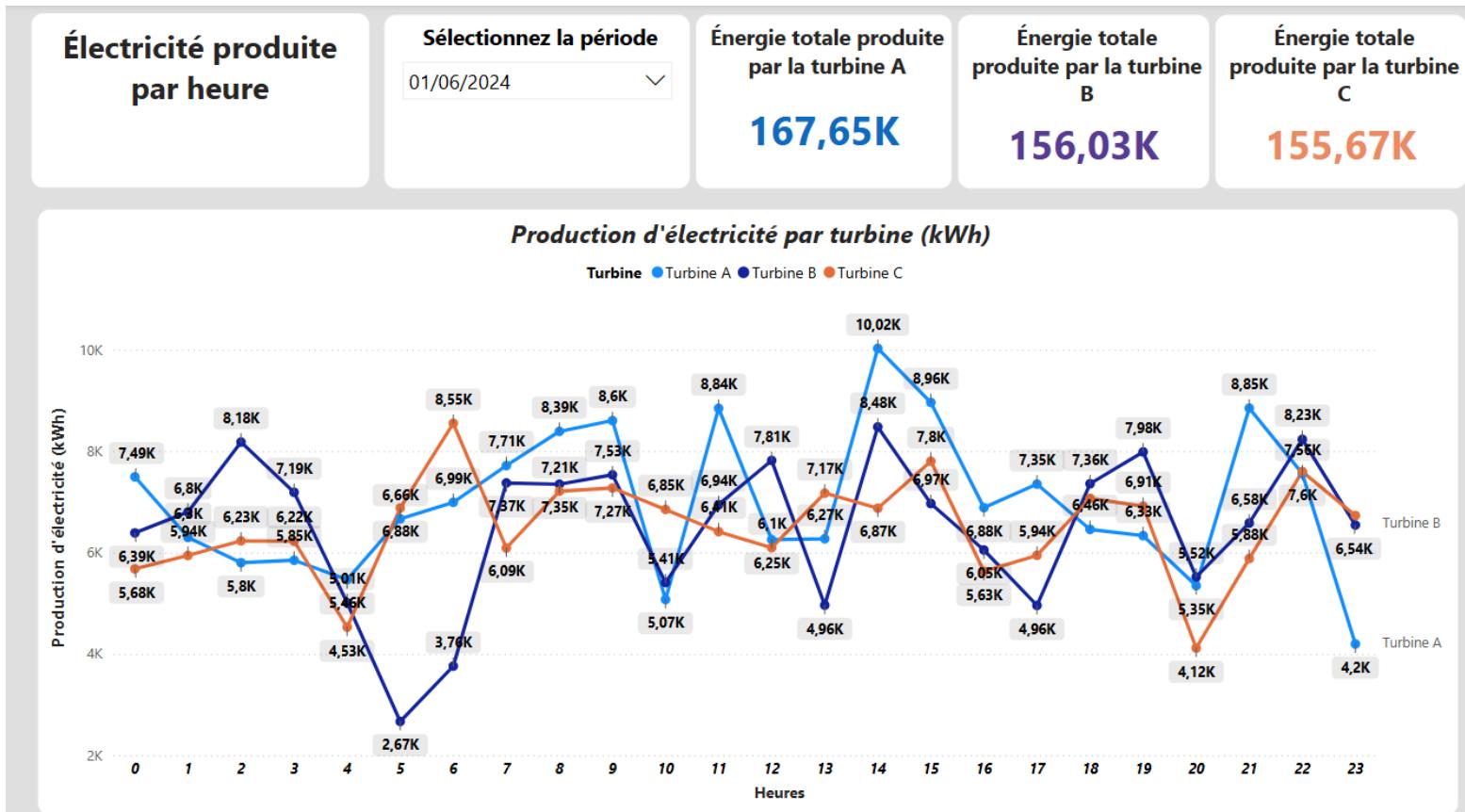
III.3 Créeation des rapports Power BI



Ce rapport donne un aperçu journalier de l'électricité produite par les turbines.



Ce rapport donne une vue détaillée de l'électricité produite chaque heure par les turbines

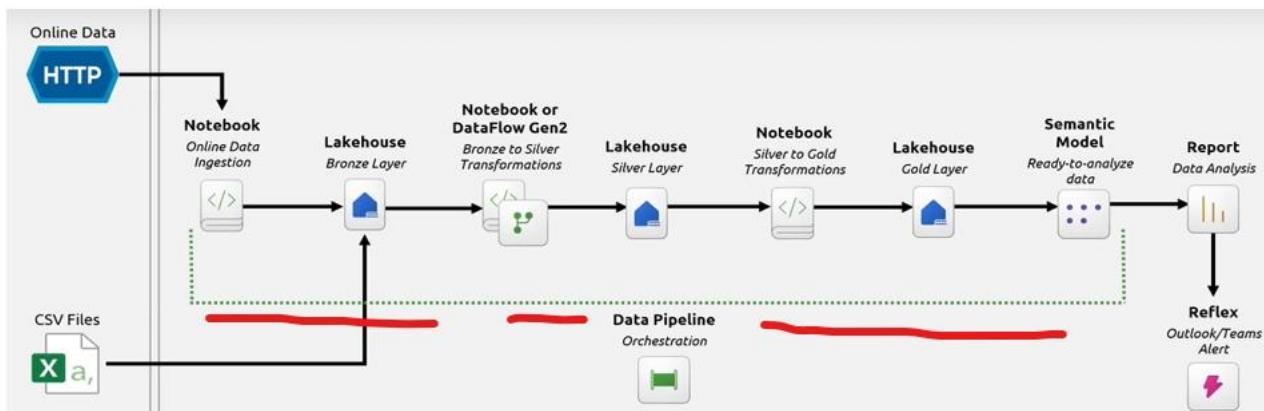


Maintenant, le rapport Power BI ainsi que tous les notebooks, dataflows Gen2 et le Lakehouse nécessaires pour effectuer les transformations de données afin qu'elles s'affichent correctement dans les rapports Power BI.

- Charger le fichier CSV dans la couche Bronze.
- Ensuite, on a créé un dataflow Gen2 ou un notebook pour effectuer les transformations entre la couche Bronze et la couche Silver. Les données ont été nettoyées et enrichies à cette étape.
- Une fois les données dans la couche Silver, nous avons créé un notebook Python pour les modéliser sous forme de schéma en étoile, un format optimisé pour l'analyse.
- Enfin, ces données modélisées sont utilisées dans le rapport Power BI pour l'analyse.

Cette phase couvre toute la partie Data Engineering ainsi que la partie visualisation avec Power BI. Maintenant, on peut intéresser à la partie Data Factory.

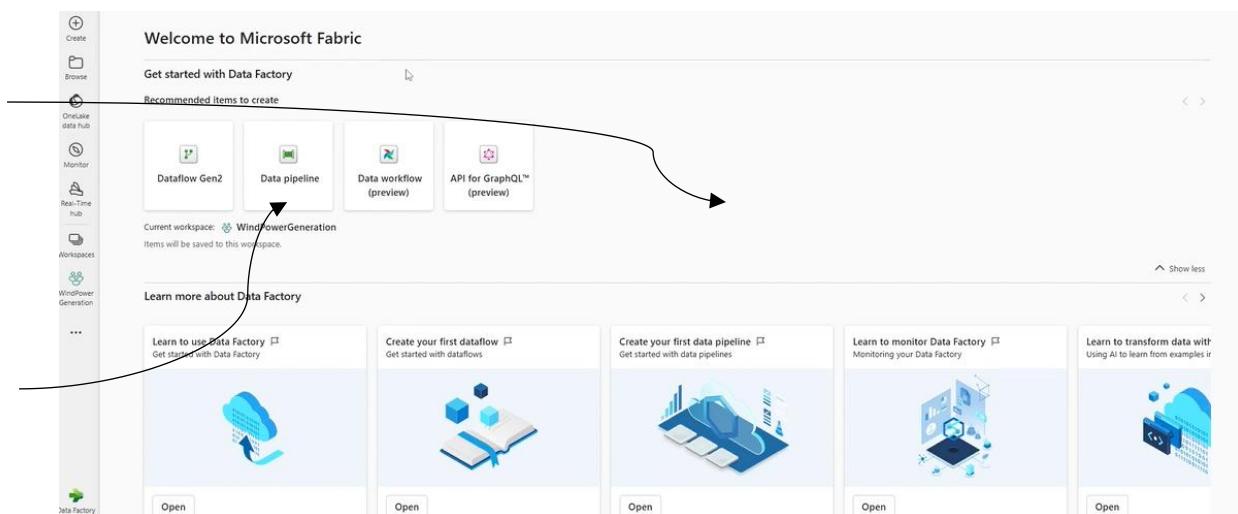
IV Orchestration des tâches de données avec Data factory



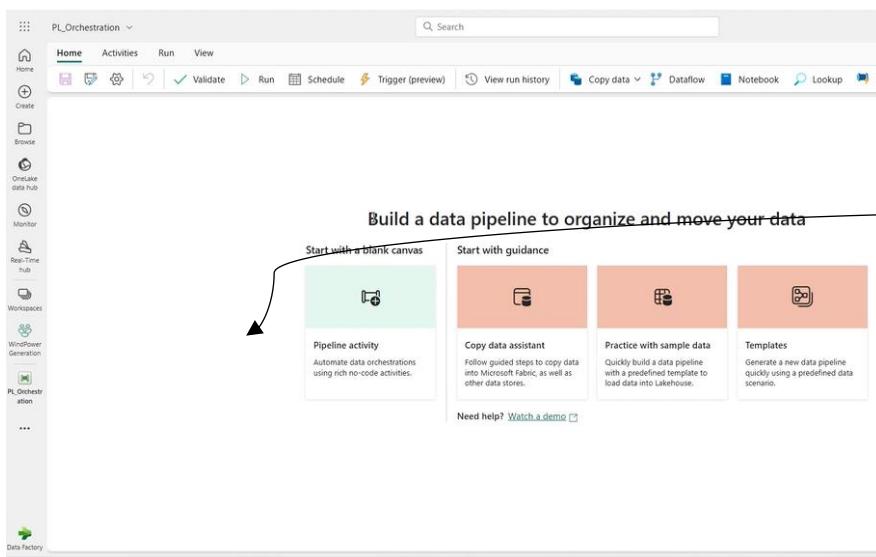
Ce pipeline permet d'orchestrer l'ensemble du processus pour qu'il soit automatiquement sans avoir besoin d'intervenir.

Les étapes sont présentées ci-dessous :

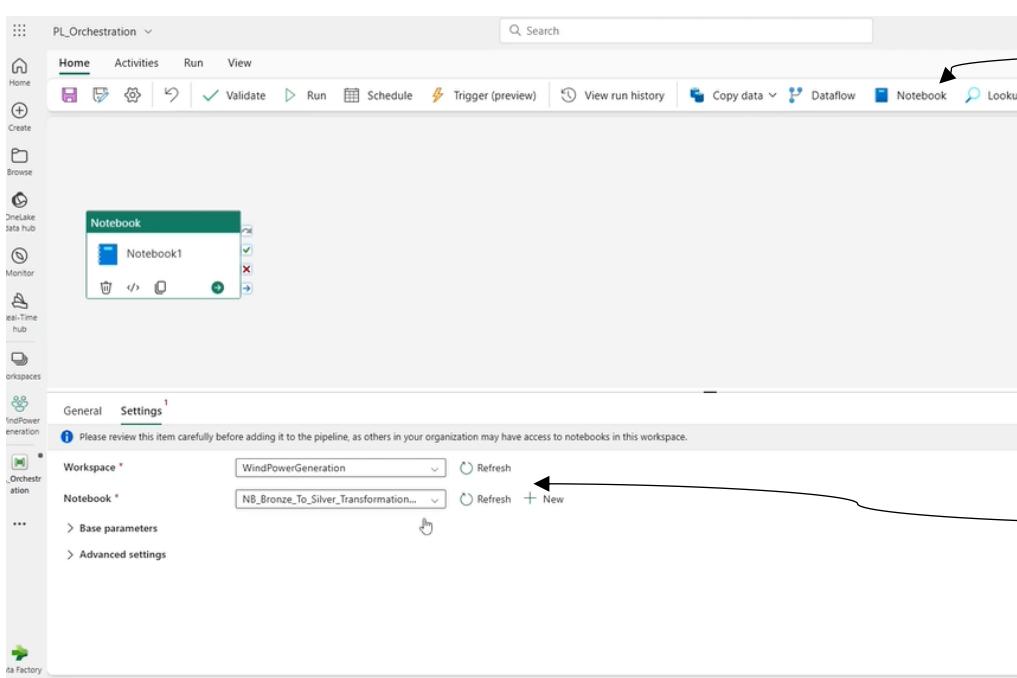
L'aperçu de l'interface de Data Factory



Étape 1 : Clique sur Data pipeline



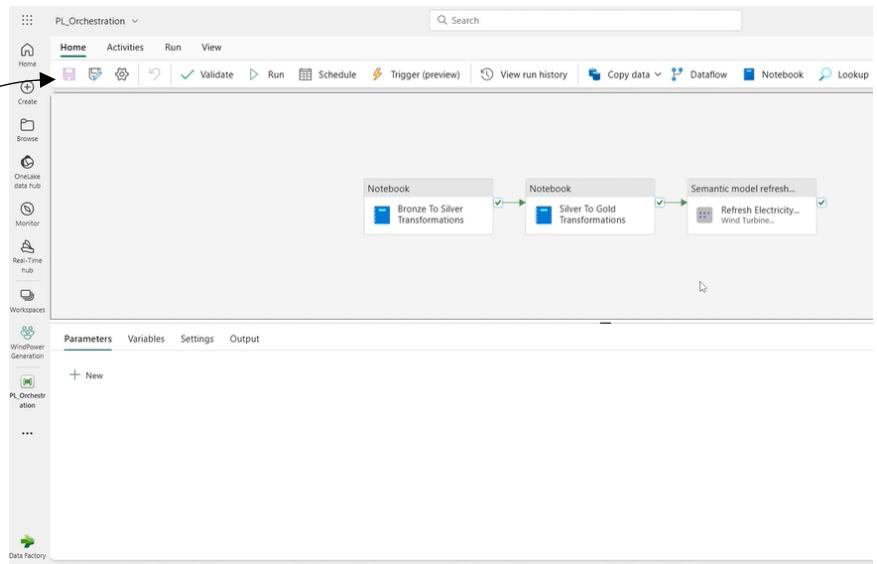
Étape 2 :
Aperçu de
l'interface de
Data pipeline



Étape 3: On créer
une activité notebook

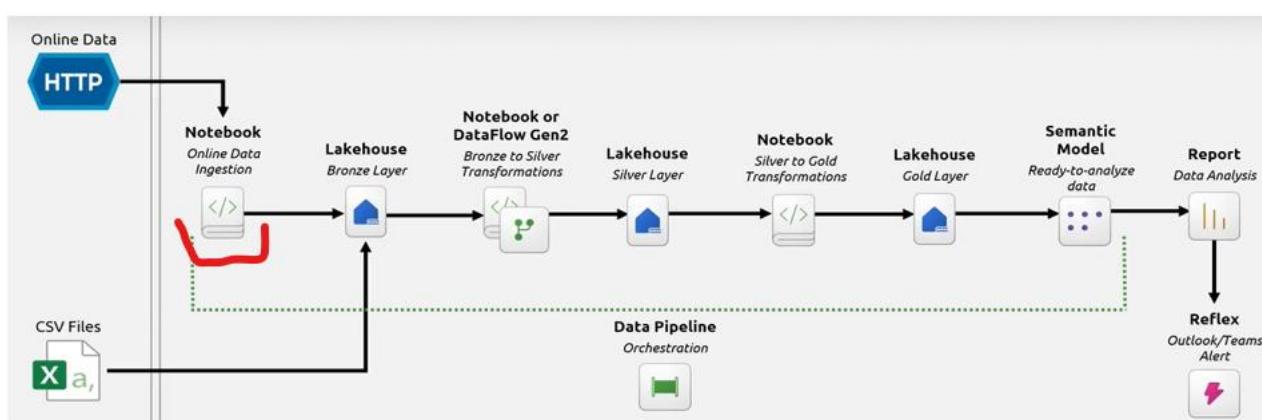
Étape 4: sélectionne
l'espace de travail ainsi
que le notebook associé
à chaque cas.

Étape 5 : On sauvegarde le pipeline « PL_Operation »



Le dernier notebook permet de récupérer les données qui arrivent chaque jour. Selon le schéma d'architecture, elles passent d'abord par un fichier CSV chargé dans le Lakehouse Bronze. Ensuite, elles subissent les différentes transformations à travers les couches Bronze, Silver et Gold, selon l'approche en médaillon, jusqu'à l'obtention du rapport final dans Power BI.

La dernière étape restante consiste donc à automatiser la récupération quotidienne des données depuis le site internet. Ces données seront récupérées par un notebook, puis chargées dans le Lakehouse Bronze déjà mis en place.



Les fichiers transmis quotidiennement.

Name
...
20240615_wind_power_production.csv
20240616_wind_power_production.csv
20240617_wind_power_production.csv
20240618_wind_power_production.csv
20240619_wind_power_production.csv
20240620_wind_power_production.csv
20240621_wind_power_production.csv

IV.1 Création du notebook: Ingestion de données quotidiennes

Création du notebook ingestion de données quotidiennes

Ce notebook a pour objectif d'automatiser l'ingestion quotidienne des données issues des turbines éoliennes provenant de la page web.

Il exécute les étapes suivantes :

- Connexion à la source de données: récupération des données depuis la page web
- Nettoyage et formatage: transformation des données brutes en un format exploitable.
- Chargement dans le Lakehouse Bronze: insertion des données dans la couche Bronze, étape initiale du traitement.

Les étapes sont présentées ci-dessous :

#Importation des bibliothèques

```
import requests
import io
import pandas as pd
from datetime import timedelta
```

→ Étape 1: Exportation des bibliothèques nécessaires

Lien principal des fichiers CSV d'origine

```
base_url = "https://raw.githubusercontent.com/datasets-for-training/main/wind-power-dataset/"
```

→ Étape 2: Récupère le chemin d'accès à la page web

Chemin vers la table wind_power_production dans le Lakehouse Bronze

```
chemin_table_bronze = "abfss://WindPowerGeneration@onelake.dfs.fabric.microsoft.com/LH_Bronze.Lakehouse/Tables/wind_power_production"
```

→ Étape 3: Récupère le chemin d'accès à la table « **wind_power_production** » dans le lakehouse niveau Bronze

Charge la table wind_power_production existante dans un DataFrame Pandas

```
df_spark = spark.read.format("delta").load(chemin_table_bronze)
df_pandas = df_spark.toPandas()
```

→ Étape 4: Convertis le DataFrame Spark en un DataFrame Pandas

	production_id	date	time	turbine_name	capacity	location_name	latitude	longitude	region	status	responsible_depart
8	2949	2024-06-07	19:40:00	Turbine C	2500	Location 3	40.7128	-74.006	Region C	Fault	Engineering
9	3474	2024-06-09	00:50:00	Turbine C	2500	Location 3	40.7128	-74.006	Region C	Fault	Engineering
10	3942	2024-06-10	02:50:00	Turbine C	2500	Location 3	40.7128	-74.006	Region C	Fault	Engineering
11	5022	2024-06-12	14:50:00	Turbine C	2500	Location 3	40.7128	-74.006	Region C	Fault	Engineering
12	5229	2024-06-13	02:20:00	Turbine C	2500	Location 3	40.7128	-74.006	Region C	Fault	Engineering
13	5394	2024-06-13	11:30:00	Turbine C	2500	Location 3	40.7128	-74.006	Region C	Fault	Engineering
14	5493	2024-06-13	17:00:00	Turbine C	2500	Location 3	40.7128	-74.006	Region C	Fault	Engineering
15	5547	2024-06-13	20:00:00	Turbine C	2500	Location 3	40.7128	-74.006	Region C	Fault	Engineering
16	5634	2024-06-14	00:50:00	Turbine C	2500	Location 3	40.7128	-74.006	Region C	Fault	Engineering
17	6036	2024-06-14	23:10:00	Turbine C	2500	Location 3	40.7128	-74.006	Region C	Fault	Engineering
18	215	2024-06-01	11:50:00	Turbine B	2000	Location 2	36.7783	-119.4179	Region B	Fault	Engineering
19	440	2024-06-02	00:20:00	Turbine B	2000	Location 2	36.7783	-119.4179	Region B	Fault	Engineering
20	998	2024-06-03	07:20:00	Turbine B	2000	Location 2	36.7783	-119.4179	Region B	Fault	Engineering
21	1220	2024-06-03	19:40:00	Turbine B	2000	Location 2	36.7783	-119.4179	Region B	Fault	Engineering
22	1442	2024-06-04	08:00:00	Turbine B	2000	Location 2	36.7783	-119.4179	Region B	Fault	Engineering

Aperçu de la table sous forme d'un DataFrame Spark.
Display(df_spark)

	production_id	date	time	turbine_name	capacity	location_name	latitude	longitude	region	status	responsible_depart
8	2949	2024-06-07	19:40:00	Turbine C	2500	Location 3	40.7128	-74.006	Region C	Fault	Engineering
9	3474	2024-06-09	00:50:00	Turbine C	2500	Location 3	40.7128	-74.006	Region C	Fault	Engineering
10	3942	2024-06-10	02:50:00	Turbine C	2500	Location 3	40.7128	-74.006	Region C	Fault	Engineering
11	5022	2024-06-12	14:50:00	Turbine C	2500	Location 3	40.7128	-74.006	Region C	Fault	Engineering
12	5229	2024-06-13	02:20:00	Turbine C	2500	Location 3	40.7128	-74.006	Region C	Fault	Engineering
13	5394	2024-06-13	11:30:00	Turbine C	2500	Location 3	40.7128	-74.006	Region C	Fault	Engineering
14	5493	2024-06-13	17:00:00	Turbine C	2500	Location 3	40.7128	-74.006	Region C	Fault	Engineering
15	5547	2024-06-14	20:00:00	Turbine C	2500	Location 3	40.7128	-74.006	Region C	Fault	Engineering
16	5634	2024-06-14	00:50:00	Turbine C	2500	Location 3	40.7128	-74.006	Region C	Fault	Engineering
17	6036	2024-06-14	23:10:00	Turbine C	2500	Location 3	40.7128	-74.006	Region C	Fault	Engineering
18	215	2024-06-01	11:50:00	Turbine B	2000	Location 2	36.7783	-119.4179	Region B	Fault	Engineering
19	440	2024-06-02	00:20:00	Turbine B	2000	Location 2	36.7783	-119.4179	Region B	Fault	Engineering
20	998	2024-06-03	07:20:00	Turbine B	2000	Location 2	36.7783	-119.4179	Region B	Fault	Engineering
21	1220	2024-06-03	19:40:00	Turbine B	2000	Location 2	36.7783	-119.4179	Region B	Fault	Engineering
22	1442	2024-06-04	08:00:00	Turbine B	2000	Location 2	36.7783	-119.4179	Region B	Fault	Engineering

Aperçu de la table sous forme d'un DataFrame Pandas.
Display(df_pandas)

```
# Récupère la date la plus récente et le jour suivant  
date_plus_recente = pd.to_datetime(df_pandas['date'], format = '%Y%m%d').max()  
date_suivante = (date_plus_recente + timedelta(days = 1)).strftime('%Y%m%d')
```

→ **Étape 5:** Récupère le jour en cours et le jour suivant.

... `Timestamp('2024-06-14 00:00:00')` → Aperçu de jour en cours :
`Display(date_plus_recente)`

· 20240615 · → Aperçu de jour suivant :
Display(date_suivante)

Puisque la date du jour et celle du lendemain ont déjà été récupérées, on peut maintenant se concentrer sur la récupération des données depuis le site web. Pour cela, il faut combiner les variables `base_url`, `date_suivante` et `wind_power_production.csv` en une seule chaîne de caractères

```
# Créer l'URL et récupérer le fichier CSV du jour suivant
URL_fichier_suivant = f"{base_url}{date_suivante}_wind_power_production.csv"
contenu_fichier = requests.get(URL_fichier_suivant).content
```

→ **Étape 6:** Crée URL et récupère le contenu de fichier sous format bit

... → Aperçu du URL :
Display(URL_fichier_suivant)

Aperçu du contenu du fichier sous format bit:
Display(contenu_fichier)

Ici, les données ne sont pas dans un format optimal : elles sont au format *bit*. Il faudra donc les décoder pour les obtenir en UTF-8

Charge le fichier CSV dans un DataFrame Pandas
df_pandas_nouveau = pd.read_csv(io.StringIO(contenu_fichier.decode('utf-8')))

→ Étape 7: décode le contenu au format UTF-8.

	Production ID	Date	Time	Turbine Name	Capacity	Location Name	Latitude	Longitude	Region	Status	Responsible Dept.
1	6049	2024-06-15	00:00:00	Turbine A	2200	Location 1	34.0522	-118.2437	Region A	Online	Operations
2	6050	2024-06-15	00:00:00	Turbine B	2000	Location 2	36.7783	-119.4179	Region B	Inspection	Maintenance
3	6051	2024-06-15	00:00:00	Turbine C	2500	Location 3	40.7128	-74.006	Region C	Online	Operations
4	6052	2024-06-15	00:10:00	Turbine A	2200	Location 1	34.0522	-118.2437	Region A	Online	Operations
5	6053	2024-06-15	00:10:00	Turbine B	2000	Location 2	36.7783	-119.4179	Region B	Online	Operations
6	6054	2024-06-15	00:10:00	Turbine C	2500	Location 3	40.7128	-74.006	Region C	Online	Operations
7	6055	2024-06-15	00:20:00	Turbine A	2200	Location 1	34.0522	-118.2437	Region A	Online	Operations
8	6056	2024-06-15	00:20:00	Turbine B	2000	Location 2	36.7783	-119.4179	Region B	Online	Operations
9	6057	2024-06-15	00:20:00	Turbine C	2500	Location 3	40.7128	-74.006	Region C	Online	Operations
10	6058	2024-06-15	00:30:00	Turbine A	2200	Location 1	34.0522	-118.2437	Region A	Online	Operations
11	6059	2024-06-15	00:30:00	Turbine B	2000	Location 2	36.7783	-119.4179	Region B	Online	Operations
12	6060	2024-06-15	00:30:00	Turbine C	2500	Location 3	40.7128	-74.006	Region C	Online	Operations
13	6061	2024-06-15	00:40:00	Turbine A	2200	Location 1	34.0522	-118.2437	Region A	Online	Operations
14	6062	2024-06-15	00:40:00	Turbine B	2000	Location 2	36.7783	-119.4179	Region B	Preventive ...	Maintenance
15	6063	2024-06-15	00:40:00	Turbine C	2500	Location 3	40.7128	-74.006	Region C	Online	Operations
16	6064	2024-06-15	00:50:00	Turbine A	2200	Location 1	34.0522	-118.2437	Region A	Online	Operations

► Aperçu du DataFrame :
Display(df_pandas_nouveau)

```
#Convertir la colonne de date au format datetime dans le DataFrame Pandas
df_pandas_nouveau['date'] = pd.to_datetime(df_pandas_nouveau['date'])
```

→ Étape 9: transformer la colonne « date » en format datetime pour faciliter le traitement des dates.

	123 production_id	124 date	ABC time	ABC turbine_name	125 capacity	ABC location_name	126 latitude	127 longitude	ABC region	ABC status	ABC responsible_dept
1	6049	2024-06-1...	00:00:00	Turbine A	2200	Location 1	34.0522	-118.2437	Region A	Online	Operations
2	6050	2024-06-1...	00:00:00	Turbine B	2000	Location 2	36.7783	-119.4179	Region B	Inspection	Maintenance
3	6051	2024-06-1...	00:00:00	Turbine C	2500	Location 3	40.7128	-74.006	Region C	Online	Operations
4	6052	2024-06-1...	00:10:00	Turbine A	2200	Location 1	34.0522	-118.2437	Region A	Online	Operations
5	6053	2024-06-1...	00:10:00	Turbine B	2000	Location 2	36.7783	-119.4179	Region B	Online	Operations
6	6054	2024-06-1...	00:10:00	Turbine C	2500	Location 3	40.7128	-74.006	Region C	Online	Operations
7	6055	2024-06-1...	00:20:00	Turbine A	2200	Location 1	34.0522	-118.2437	Region A	Online	Operations
8	6056	2024-06-1...	00:20:00	Turbine B	2000	Location 2	36.7783	-119.4179	Region B	Online	Operations
9	6057	2024-06-1...	00:20:00	Turbine C	2500	Location 3	40.7128	-74.006	Region C	Online	Operations
10	6058	2024-06-1...	00:30:00	Turbine A	2200	Location 1	34.0522	-118.2437	Region A	Online	Operations
11	6059	2024-06-1...	00:30:00	Turbine B	2000	Location 2	36.7783	-119.4179	Region B	Online	Operations

```
# Convertir le DataFrame Pandas en DataFrame Spark
df_spark_new = spark.createDataFrame(df_pandas_new, schema = df_spark.schema)
```

→ Étape 10 : transforme le DataFrame Pandas en DataFrame Spark reconnu par le Lakehouse, puis fais le lien entre les données de la page web et celles du niveau Bronze

Maintenant que les données du Lakehouse « Bronze » sont au même format que celles provenant du site web, on peut les fusionner.

```
# Insére les nouvelles données dans la table wind_power_production du Lakehouse Bronze
df_spark_new.write.format("delta").mode("append").save(bronze_table_path)
```

→ Étape 11: charge les nouvelles données dans la table *wind_power_production* (dans le Lakehouse, niveau Bronze).

IV.2 Test du notebook d'ingestion de données quotidiennes

Le notebook a été conçu pour récupérer automatiquement, chaque jour, les données disponibles sur le site. Une fois exécuté, il charge les données du jour suivant dans le Lakehouse Bronze. Par exemple, voici celles du 15 juin 2024. Si on le relance, il ira chercher les données du 16, puis du 17, et ainsi de suite.

Une requête pour tester l'affichage de données « 2024-06-15 » dans SQL

	123 production_id	124 date	ABC time	ABC turbine_name	125 capacity	ABC location_name	126 latitude	127 longitude	ABC region	ABC status	ABC responsible_dept
1	6088	2024-06-15	02:10:00	Turbine A	2200	Location 1	34.0522	-118.2437	Region A	Offline	Operations
2	6093	2024-06-15	02:20:00	Turbine C	2500	Location 3	40.7128	-74.006	Region C	Fault	Engineering
3	6106	2024-06-15	03:10:00	Turbine A	2200	Location 1	34.0522	-118.2437	Region A	Offline	Operations
4	6110	2024-06-15	03:20:00	Turbine B	2000	Location 2	36.7783	-119.4179	Region B	Fault	Engineering
5	6122	2024-06-15	04:00:00	Turbine B	2000	Location 2	36.7783	-119.4179	Region B	Offline	Operations
6	6206	2024-06-15	08:40:00	Turbine B	2000	Location 2	36.7783	-119.4179	Region B	Fault	Engineering
7	6229	2024-06-15	10:00:00	Turbine A	2200	Location 1	34.0522	-118.2437	Region A	Offline	Operations
8	6253	2024-06-15	11:20:00	Turbine A	2200	Location 1	34.0522	-118.2437	Region A	Offline	Operations
9	6269	2024-06-15	12:10:00	Turbine B	2000	Location 2	36.7783	-119.4179	Region B	Fault	Engineering
10	6280	2024-06-15	12:50:00	Turbine A	2200	Location 1	34.0522	-118.2437	Region A	Fault	Engineering
11	6293	2024-06-15	13:30:00	Turbine B	2000	Location 2	36.7783	-119.4179	Region B	Offline	Operations
12	6337	2024-06-15	16:00:00	Turbine A	2200	Location 1	34.0522	-118.2437	Region A	Offline	Operations
13	6368	2024-06-15	17:40:00	Turbine B	2000	Location 2	36.7783	-119.4179	Region B	Offline	Operations

IV.3 Exécution du Pipeline d'orchestration

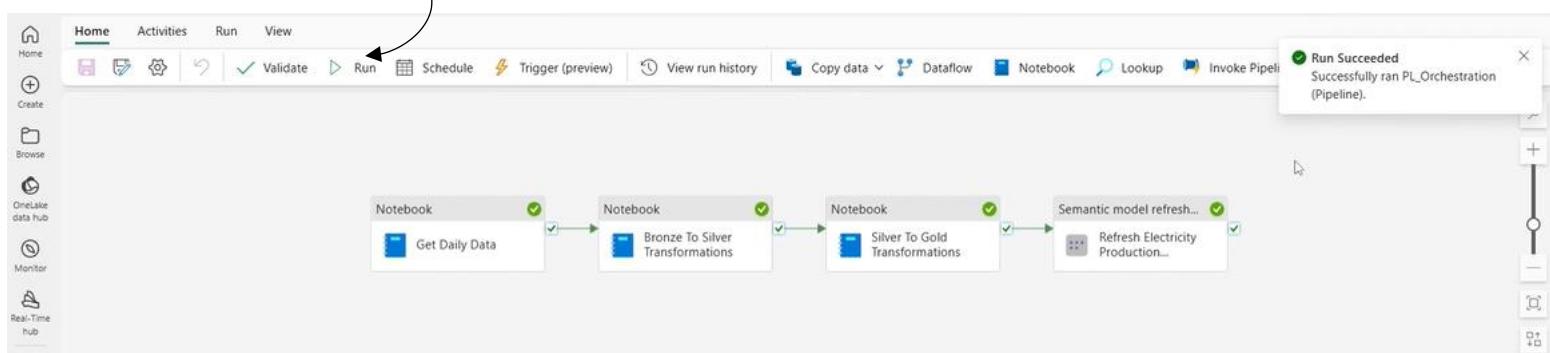
Le pipeline d'orchestration a été configuré pour s'exécuter automatiquement chaque jour. Il commence par récupérer les données depuis la page web, les charge dans le Lakehouse Bronze, puis lance tous les notebooks qu'on a mis en place pour les transformer. Les données passent ensuite par les couches Silver et Gold, avant que le rapport Power BI ne soit mis à jour.

Actuellement, les données sont stockées dans le Lakehouse Bronze. Il faut maintenant les déplacer manuellement vers le Lakehouse Silver, puis vers le Lakehouse Gold, et enfin rafraîchir manuellement le rapport Power BI.

Faire cela tous les jours serait assez fastidieux. C'est justement pour cela qu'on a conçu ce pipeline de données : pour automatiser ce processus et gagner du temps.

Pour exécution un pipeline :

On clique sur « Run »



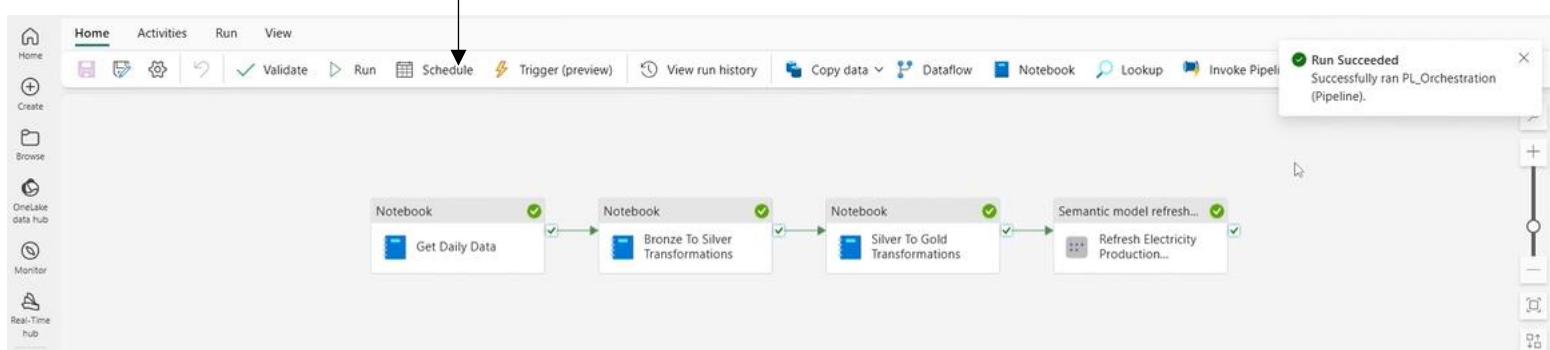
Si on réexécute ce pipeline on aura les données du 16 juin

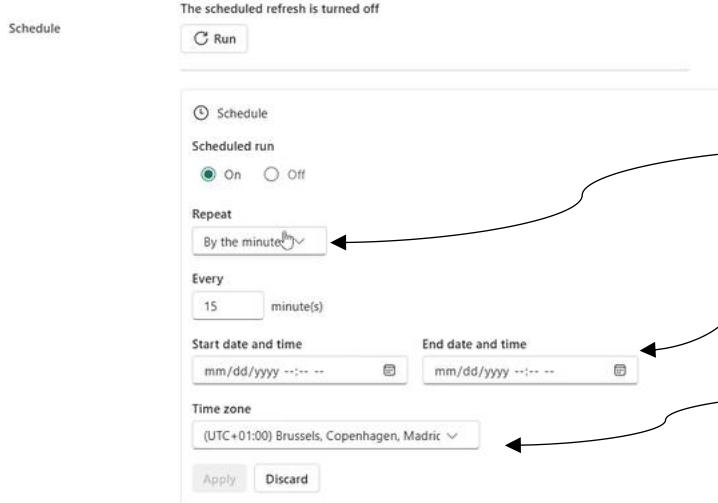
L'idée ensuite ce de faire en sorte que ce pipeline s'exécute tout seule tout le jour pour afficher les données actualisées du jour encours (ça a d récupération la dernière donnée est qu'elles sont ensuite affichées dans le rapport Power BI).

IV.4 Programmer un Pipeline Data Factory pour qu'il s'exécute automatiquement

Pour éviter d'exécuter manuellement tous les jours le pipeline on va le planifier, l'idée est d'avoir une solution complètement automatisée.

Étape 2 :Clique sur « schedule »



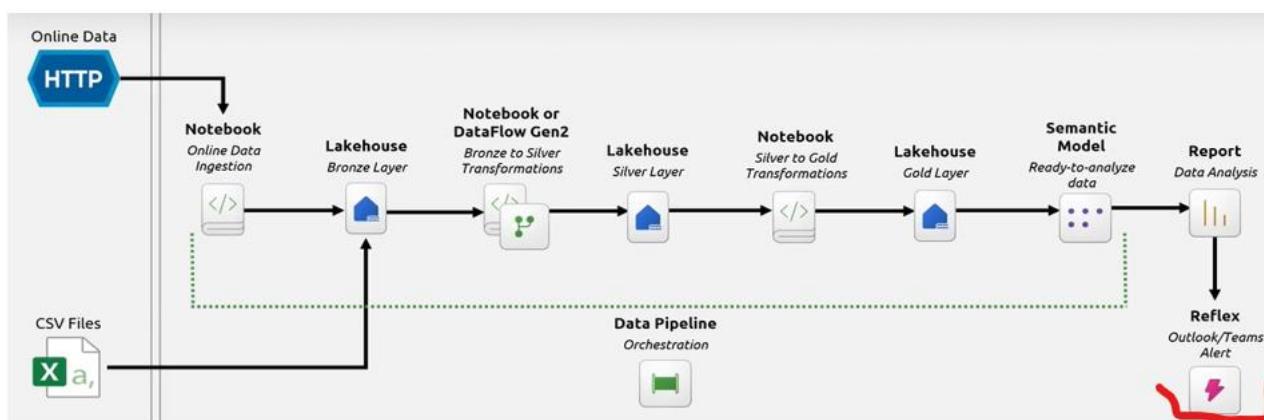


Étape 2: On définit la fréquence d'exécution de pipeline(/jour)

Étape 3: On Définit la date de démarrage et fin (si nécessaire)

Étape 4: On définit la zone horaire

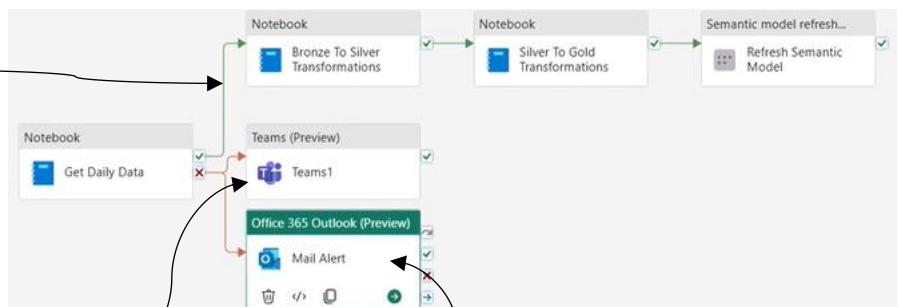
IV.5 Configuration des alertes Outlook et Teams en cas d'erreur sur le Pipeline



Pour signaler des événements, comme un bogue ou simplement pour indiquer que le pipeline s'est bien exécuté, on peut mettre en place des alertes de deux façons :

- Recevoir une alerte par e-mail: en cas d'erreur dans l'une des activités, un mail peut être envoyé pour nous prévenir.
- Recevoir une notification sur Teams: un message peut nous être envoyé pour signaler le bogue.

Étape1
:Linke
Sevice



Étape 2: Composant Teams

Étape 3: Composant Outlook

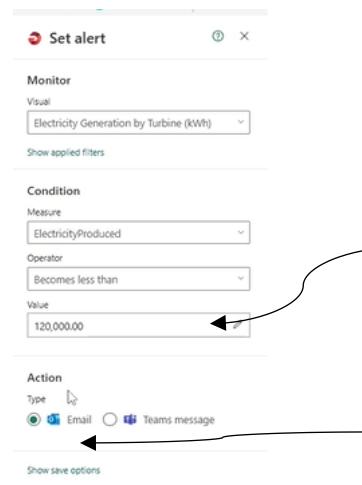
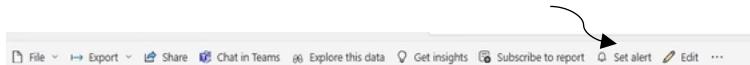
IV.6 Créer une alerte à partir d'un rapport Power BI

Si l'on consulte le rapport, on constate que la production d'électricité tourne autour de 150 000 kWh par jour.

Si un jour la production devient inférieure à 150 000 kWh, cela pourrait indiquer une anomalie sur l'une des éoliennes.

L'idéal serait alors de recevoir une alerte, par e-mail ou via Teams, si une production inférieure à 120 000 kWh est détectée.

Étape 1: Clique sur « Set alert »



Étape 2: on définit le seuil à 120 000 kWh

Étape 3: Sélection l'une des options « Email » ou « Teams »

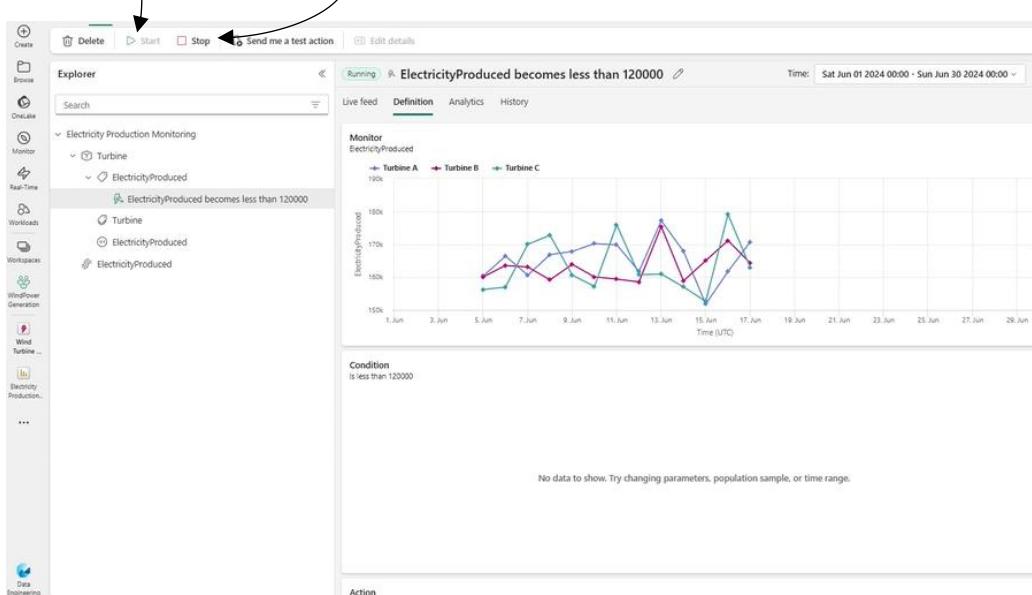
Étape 4: Clique sur « Créer »

IV.7 Configuration une alerte activator pour envoyer des alertes Outlook et Team

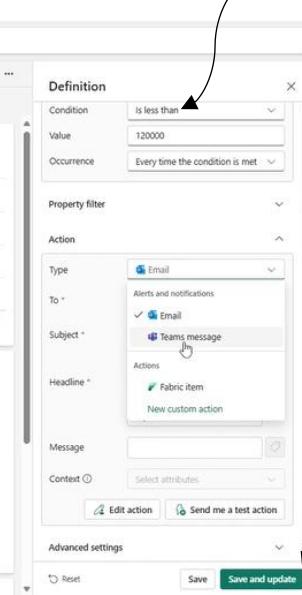
Pour démarrer les alertes

Pour arrêter les alertes

Étape 1: Sélection une condition « Is less than »

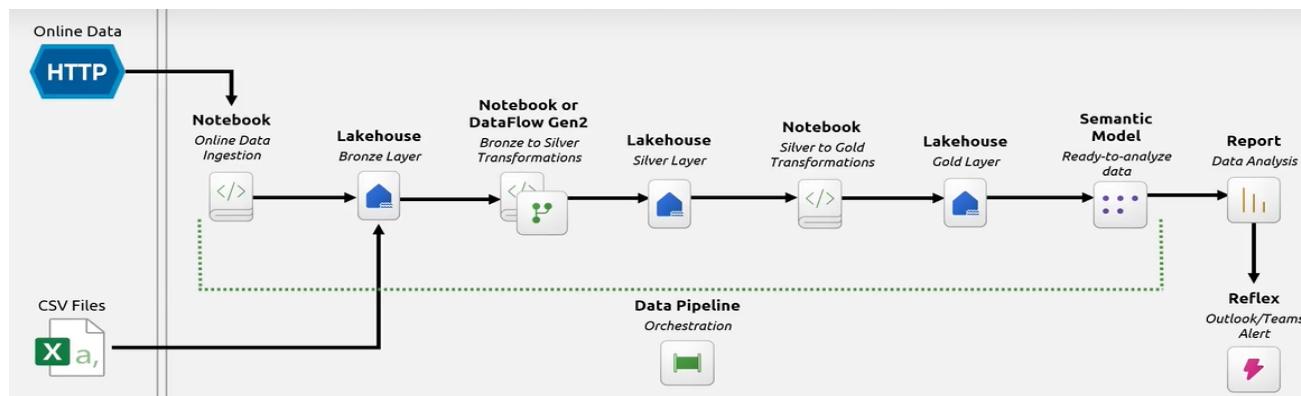


Étape 2 :
Clique sur « Enregistrer » ou bien sur « Enregistrer et mettre à jour », selon ce qu'on veut faire



Cela y est, la configuration des alertes sont terminées et les utilisateurs seront notifiés à chaque fois que la production électrique sera inférieure à 120 000 kWh, que ce soit par e-mail ou via Outlook.

V Conclusion



L'architecture en médaillon repose sur deux sources de données. La première vient d'un fichier CSV qu'on a chargé dans le Lakehouse Bronze. La deuxième provient d'une page web, dont les données sont aussi stockées. Une fois les données dans le Lakehouse Bronze, elles ont été nettoyées et enrichies avant d'être transférées dans le Lakehouse Silver, en utilisant deux approches : les Dataflows Gen2 et un notebook Python.

Ensuite, les données du Silver ont été retravaillées dans un notebook pour les structurer selon un schéma en étoile, avec une table de faits et plusieurs tables de dimensions, avant d'être stockées dans le Lakehouse Gold.

À partir du Gold, on a construit un modèle sémantique en définissant les relations entre les différentes tables. Ce modèle a permis de créer un rapport Power BI pour suivre la production d'électricité. On a conçu deux rapports: une par jour et une autre par heure, ce qui permet une analyse claire de l'évolution de la production d'électricité.

Un pipeline Data Factory a ensuite été mis en place pour automatiser toute cette chaîne. Il exécute les notebooks chaque jour, sans action manuelle. Enfin, des alertes ont été configurées pour nous prévenir automatiquement si certains seuils sont dépassés.