

---

## **Conception d'un Pipeline de Données en Temps Réel sur Azure:**

### **Cas d'Usage OpenSky**

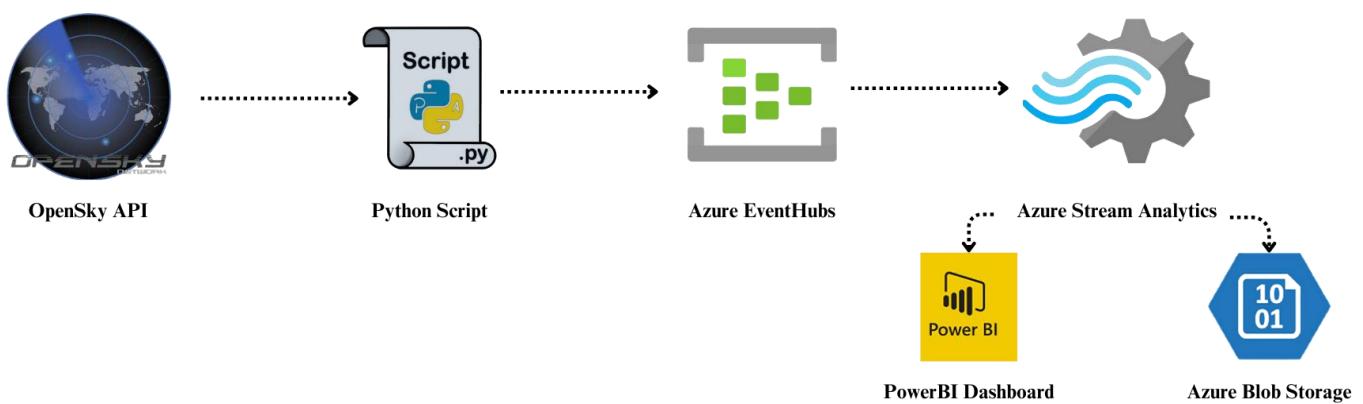
---

#### **Objectifs Du Projet:**

- Comprendre et mettre en œuvre un pipeline de traitement de données en temps réel.
- Configurer et optimiser les services pour un traitement efficace.
- Sécuriser et partager des données avec les bonnes pratiques du cloud.
- Intégrer les données issues de différentes sources pour des analyses avancées.

#### **Vue d'ensemble du travail :**

Dans ce travail, nous avons mis en place un flux de traitement des données en temps réel basé sur Azure. Les données ont été collectées à l'aide de l'API OpenSky et transmises à **Azure Event Hubs**, qui sert de passerelle pour gérer les flux entrants. Ensuite, nous avons utilisé **Azure Stream Analytics** pour effectuer un traitement en temps réel sur les données grâce à des requêtes SQL. Les données transformées ont été stockées dans un **Blob Storage Azure**, une solution fiable pour gérer les volumes importants. Enfin, nous avons créé des visualisations dynamiques avec **Power BI** pour permettre une exploration simple et intuitive des résultats. Ce flux est sécurisé grâce à des techniques de chiffrement et un contrôle d'accès strict, garantissant la confidentialité des données à chaque étape.



#### **Présentation des outils :**

- Kafka ou équivalent pour la gestion des flux de données.

- ⇒ Dans notre flux, nous avons utilisé **Azure Event Hubs**, qui est l'équivalent de **Kafka** sur la plateforme Azure. Cet outil permet de gérer les flux de données en temps réel en agissant comme un collecteur et un distributeur de messages provenant de différentes sources.



**Azure EventHubs**

- **Utilisation d'un service cloud pour le traitement en temps réel (ex. : AWS Kinesis, Azure Event Hub, Google Pub/Sub).**
- ⇒ Nous avons intégré **Azure Stream Analytics** pour analyser les données en temps réel. Cet outil est idéal pour exécuter des requêtes SQL sur les données en transit et extraire des informations exploitables avant de les stocker.



**Azure Stream Analytics**

- **Méthodes de sécurisation des données dans le cloud (chiffrement, contrôle d'accès).**
- ⇒ La sécurité des données a été assurée à chaque étape grâce au chiffrement de chaîne de connexion et des flux dans Azure et au contrôle d'accès basé sur les rôles (RBAC), garantissant que seules les personnes autorisées peuvent accéder aux données.

## Étape 1 : Mise en place d'un pipeline de flux de données en temps réel

**Objectif : Configurer un environnement pour capturer, traiter et stocker des données en temps réel.**

**1. Configuration d'un producteur de données (ex. : script Python pour envoyer des événements en temps réel).**

- **Configuration du l'API OpenSky :**

L'API OpenSky est un service qui permet d'accéder à des données en temps réel et historiques sur le trafic aérien. Elle offre des informations précieuses sur les positions des avions, les altitudes, les vitesses, et bien d'autres paramètres, idéales pour les applications d'analyse et de visualisation de données aériennes. Pour l'utiliser, il est nécessaire de créer un compte OpenSky pour obtenir les informations d'accès à l'API.

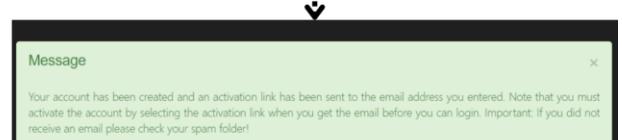
Voici ci-dessous les étapes pour configurer un compte OpenSky et accéder à l'API :

⇒ Accéder au site [OpenSky Network](https://opensky-network.org)

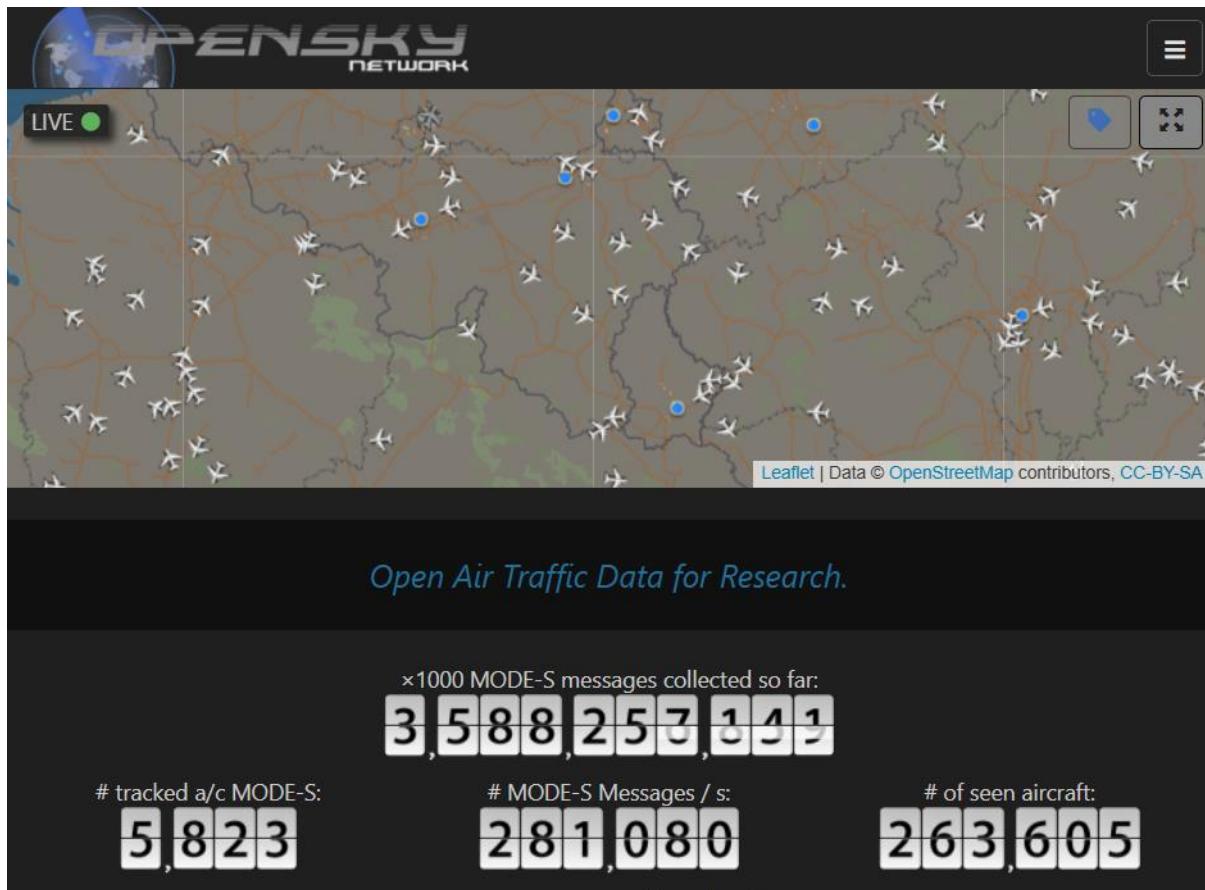
The screenshot shows a Google search result for "opensky api". The top result is "The OpenSky Network - Free ADS-B and Mode S data for ...". A dashed arrow points from this result to a screenshot of the "User Registration" form on the OpenSky website. The registration form includes fields for Name, Username, Password, Confirm Password, Email Address, Confirm Email Address, Country, and a checkbox for agreeing to the Terms of Use. Below the form is a "Message" box containing a confirmation message about account creation and activation.

⇒ Créer un compte :

The screenshot shows the "User Registration" form on the OpenSky website. It includes fields for Name, Username, Password, Confirm Password, Email Address, Confirm Email Address, Country, and a checkbox for agreeing to the Terms of Use. The "Name" field is filled with "Wafa", "Username" with "Wafou", and "Email Address" with "jebaliwafa00@gmail.com". The "Country" dropdown is set to "Tunisia". The "Do you agree to the Terms of Use?" section contains two radio buttons: "I agree" (selected) and "I do not agree". At the bottom are "Register" and "Cancel" buttons.



⇒ Connecter et récupérer less identifiants (nom d'utilisateur et mot de passe).



⇒ Tester l'API OpenSky en mode anonyme avec la requête suivante, qui cible uniquement la France :

<https://opensky-network.org/api/states/all?lamin=41.0&lomin=-5.0&lamax=51.0&lomax=10.0>

Les paramètres de cette requête permettent de définir une zone géographique précise :

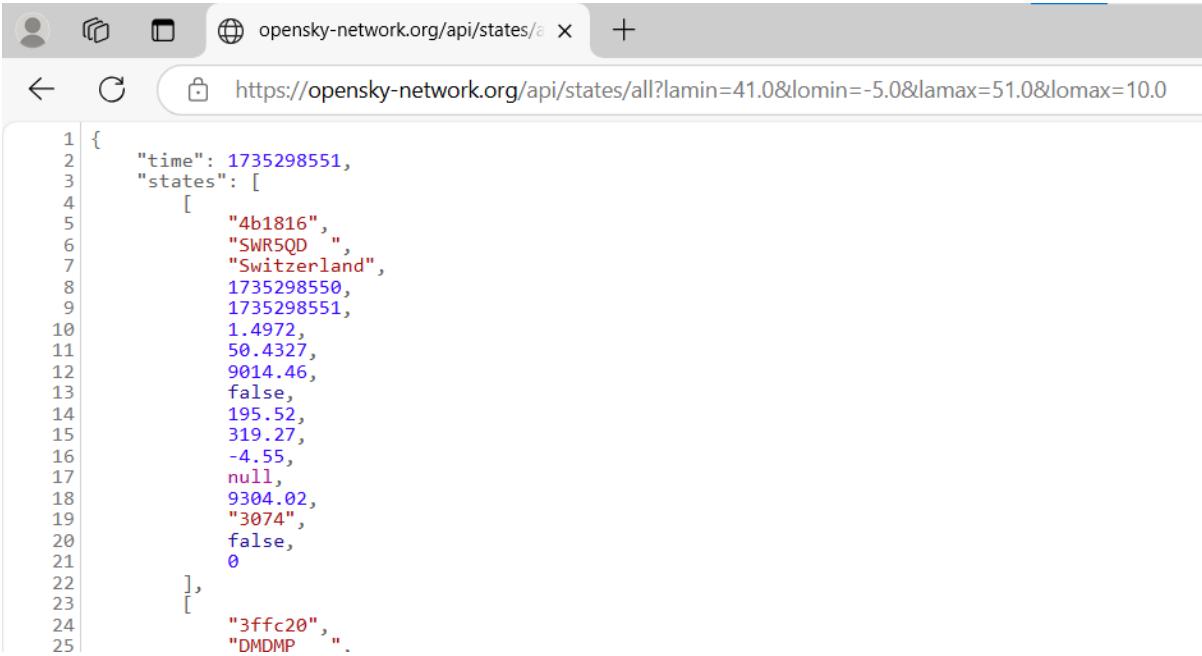
- **lamin** : Latitude minimale (sud).
- **lomin** : Longitude minimale (ouest).
- **lamax** : Latitude maximale (nord).
- **lomax** : Longitude maximale (est).

Ces coordonnées couvrent la France, ce qui nous permet de limiter les données reçues aux vols situés dans cette région. Cela réduit la quantité de données à traiter, optimise nos ressources et contrôle les coûts sur Azure.

Nous analysons les résultats localement avant de les intégrer au flux complet.

**(!) NB :** Cette limitation a été choisie car le trafic aérien est actuellement très élevé en période de Noël. En ciblant uniquement la France, nous réduisons le volume de données traitées, ce qui nous permet de mieux contrôler les dépenses liées à l'utilisation des services Azure plus tard.

⇒ Exemple de réponse JSON :



```

1 {
2   "time": 1735298551,
3   "states": [
4     [
5       "4b1816",
6       "SWR5QD ",
7       "Switzerland",
8       1735298550,
9       1735298551,
10      1.4972,
11      50.4327,
12      9014.46,
13      false,
14      195.52,
15      319.27,
16      -4.55,
17      null,
18      9304.02,
19      "3074",
20      false,
21      0
22    ],
23    [
24      "3fffc20",
25      "DMDMP"
    ]
  ]
}

```

⇒ Documentation des champs retournés par l'API :

Index	Property	Type	Description	12	sensors	int[]	IDs of the receivers which contributed to this state vector. Is null if no filtering for sensor was used in the request.
0	icao24	string	Unique ICAO 24-bit address of the transponder in hex string representation.	13	geo_altitude	float	Geometric altitude in meters. Can be null.
1	callsign	string	Callsign of the vehicle (8 chars). Can be null if no callsign has been received.	14	squawk	string	The transponder code aka Squawk. Can be null.
2	origin_country	string	Country name inferred from the ICAO 24-bit address.	15	spi	boolean	Whether flight status indicates special purpose indicator.
3	time_position	int	Unix timestamp (seconds) for the last position update. Can be null if no position report was received by OpenSky within the past 15s.	16	position_source	int	Origin of this state's position. <ul style="list-style-type: none"> <li>0 = ADS-B</li> <li>1 = ASTERIX</li> <li>2 = MLAT</li> <li>3 = FLARM</li> </ul>
4	last_contact	int	Unix timestamp (seconds) for the last update in general. This field is updated for any new, valid message received from the transponder.				Aircraft category. <ul style="list-style-type: none"> <li>0 = No information at all</li> <li>1 = No ADS-B Emitter Category Information</li> <li>2 = Light (&lt; 15500 lbs)</li> <li>3 = Small (15500 to 75000 lbs)</li> <li>4 = Large (75000 to 300000 lbs)</li> <li>5 = High Vortex Large (aircraft such as B-757)</li> <li>6 = Heavy (&gt; 300000 lbs)</li> <li>7 = High Performance (&gt; 5g acceleration and 400 kts)</li> <li>8 = Rotorcraft</li> <li>9 = Glider / sailplane</li> <li>10 = Lighter-than-air</li> <li>11 = Parachutist / Skydiver</li> <li>12 = Ultraflight / hanse-elider / nareglider</li> </ul>
5	longitude	float	WGS-84 longitude in decimal degrees. Can be null.				
6	latitude	float	WGS-84 latitude in decimal degrees. Can be null.				
7	baro_altitude	float	Barometric altitude in meters. Can be null.				
8	on_ground	boolean	Boolean value which indicates if the position was retrieved from a surface position report.				
9	velocity	float	Velocity over ground in m/s. Can be null.				
10	true_track	float	True track in decimal degrees clockwise from north (north=0°). Can be null.				
11	vertical_rate	float	Vertical rate in m/s. A positive value indicates that the airplane is climbing, a negative value indicates that it descends. Can be null.	17	category	int	

(!) NB : L'accès peut être en mode **anonyme** ou avec un compte. Le mode anonyme permet de tester l'API rapidement, mais il limite le nombre de requêtes par jour. Pour un usage intensif, il est recommandé d'utiliser un compte pour obtenir un accès étendu.

- Configuration du Script Producteur :

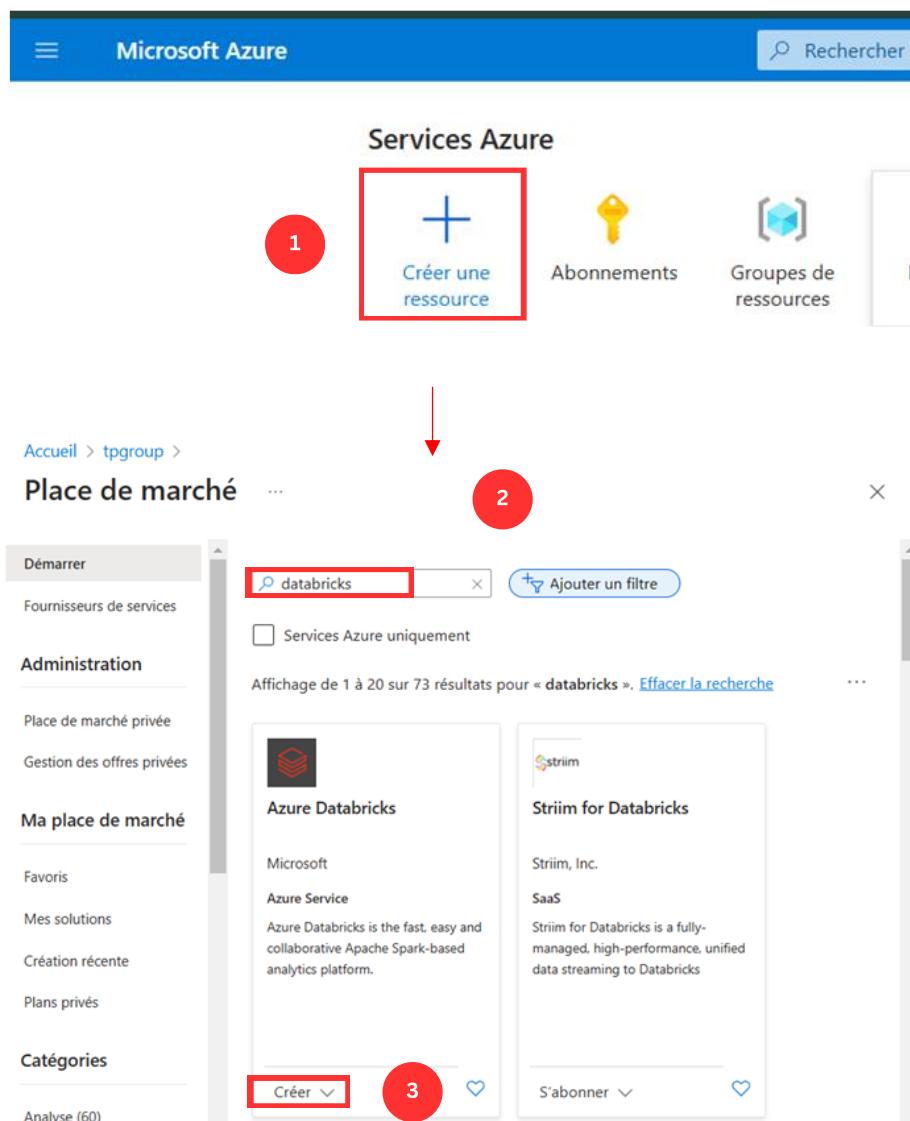
Avant de configurer le script Python responsable de l'extraction des données à partir de l'API OpenSky, il est nécessaire de créer un notebook dans Azure Databricks. Azure Databricks est une plateforme d'analyse de données basée sur Apache Spark, qui permet de collaborer de manière interactive pour explorer et analyser des données massives. Elle offre un

environnement optimisé pour le traitement des données, le machine learning, et l'intégration avec d'autres services Azure, comme Event Hub.

Une fois le notebook créé dans Azure Databricks, nous allons utiliser le script Python pour récupérer les données de l'API OpenSky. Ce script prendra les informations extraites et les enverra à Event Hub, où elles pourront être traitées et analysées en temps réel. La configuration d'Event Hub sera détaillée dans l'étape suivante.

Ainsi, le flux de travail complet consiste à :

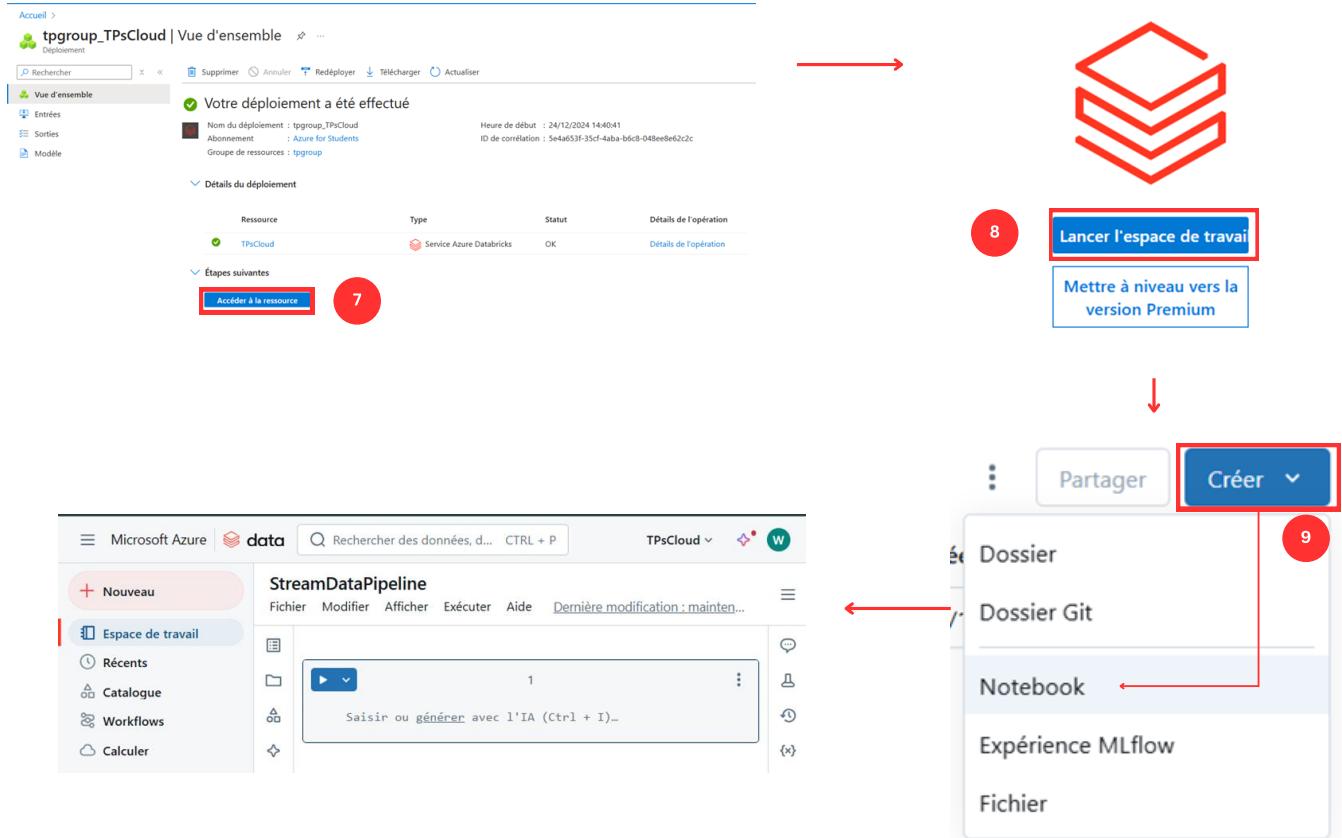
- **Créer une ressource Databricks :**



**(!)NB :**

- **Groupe de ressources :** Un **groupe de ressources** dans Azure est un conteneur logique qui regroupe et organise les ressources Azure (comme les machines virtuelles, bases de données, Event Hubs, etc.) liées à un même projet.
- **Région :** En Azure, une région est un emplacement géographique physique où Microsoft possède des centres de données pour héberger et gérer nos ressources. Choisir une région signifie décider où déployer nos services et données. Par exemple, Nous avons choisi **West Europe** comme région car elle est proche de la **Tunisie**, ce qui réduit la latence et optimise les performances. De plus, choisir une région plus éloignée pourrait augmenter les coûts, car les tarifs Azure varient selon la localisation.
- **Niveau Tarifaire :** Azure propose différents niveaux tarifaires selon les besoins en performance et fonctionnalités. Nous avons opté pour le niveau **Standard**, ce niveau est idéal pour des projets avec des besoins modérés en performance et une optimisation des coûts. Si des exigences particulières (comme des temps de réponse extrêmement bas ou une mise à l'échelle massive) sont nécessaires, il serait préférable de passer à un niveau supérieur comme le **Premium**.

- **Créer un notebook dans Azure Databricks pour l'exécution du script.**



Après avoir créé un notebook dans Azure Databricks, il est essentiel de lui **attribuer un cluster** pour exécuter le code.

- ⇒ Un **cluster dans Azure Databricks** est un groupe de machines virtuelles connectées entre elles, conçu pour exécuter des tâches de calcul intensif, comme le traitement de grandes quantités de données ou l'exécution d'algorithmes d'apprentissage automatique. Dans Azure Databricks, un cluster est utilisé pour :
- **Exécuter des notebooks** : Il fournit la puissance de calcul nécessaire pour exécuter le code Python, Scala, SQL ou R.
- **Traiter des données massives** : Grâce à Spark, il permet de distribuer le traitement sur plusieurs nœuds pour accélérer les calculs.
- **Collaborer en temps réel** : Plusieurs utilisateurs peuvent partager un cluster pour travailler ensemble.
- ⇒ **Si un cluster n'existe pas déjà** : Si aucun cluster n'est disponible, on doit en créer un. Pour cela :

Créer et joindre à une nouvelle ressource de calcul

**Policy**: Personal Compute

**Nom**: Cluster1

**Type d'instance**: Standard\_DS3\_v2

**Environnement d'exécution**: Runtime: 16.1 ML (Scala 2.12, Spark 3.5.0)

**Résumé**

1 driver | 14 Go de mémoire, 4 coeurs  
Standard\_DS3\_v2 | 0,75 DBU/heure

**Configuration avancée**

**Créer** (Step 11)

Annuler

Attendre entre 1 min et 3 min

et voilà notre cluster est bien connecté

- **Configurer le script pour récupérer les données de l'API OpenSky.**
- ⇒ Pour le script, nous devons tout d'abord installer le package EventHub. Ce package permettra de connecter notre script producer à l'Event Hub que nous allons créer juste après.

Microsoft Azure | databricks

Rechercher des données, des notebooks, des récents et bien plus ... CTRL + P

**Calculer**

**Cluster1**

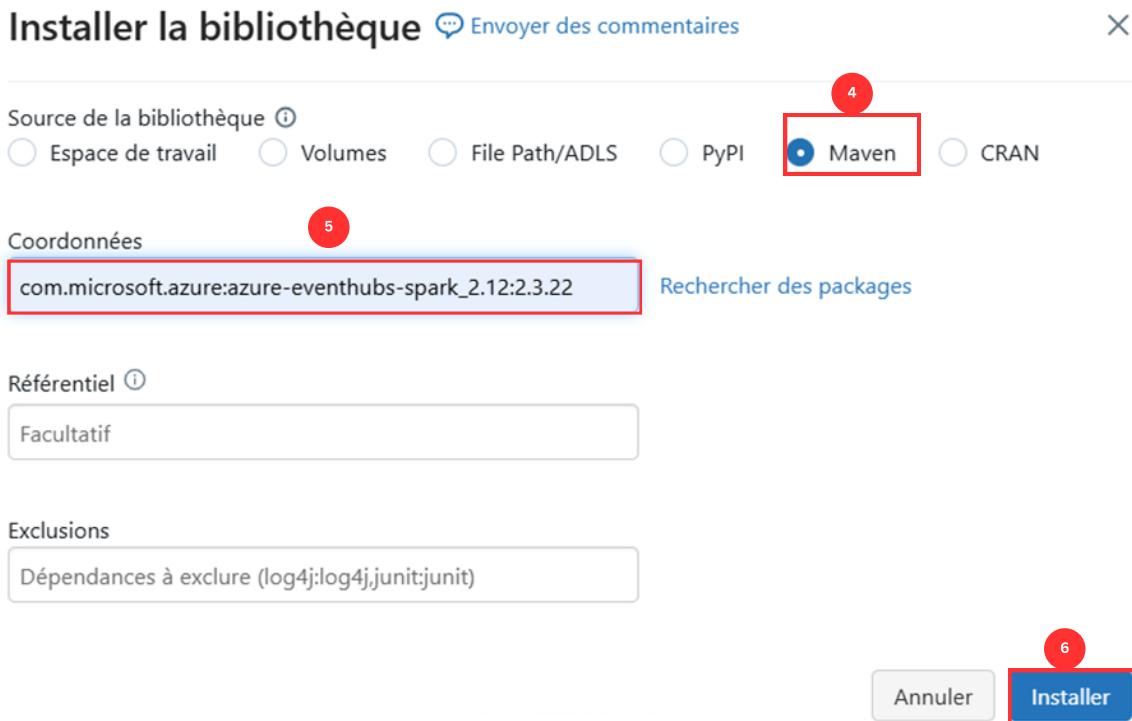
**Configuration**: Personal Compute

**Mode d'accès**: Utilisateur unique | Hamdi Belanez

**Bibliothèques**

**Installer une nouvelle bibliothèque**

**Terminer** | **Modifier**



- ⇒ Une fois l'installation terminée, le package sera disponible pour utilisation dans notre script Python.

Ce script extrait des données de vol en temps réel depuis l'API OpenSky et les envoie à Azure Event Hub.

- Extraction : Il récupère des informations comme la position, la vitesse, et l'altitude des vols sur une zone géographique spécifique (la France).
- Envoi : Les messages sont regroupés et transmis à Event Hub, qui agit comme un pipeline de données massives pour leur exploitation en aval.
- Automatisation : Le processus se répète toutes les 60 secondes, garantissant un flux continu en temps réel.

```

import json
import requests
from azure.eventhub import EventHubProducerClient, EventData
import time

# Configuration de l'Event Hub
connection_str =
"Endpoint=sb://projecteventhubs.servicebus.windows.net/;SharedAccessKeyName=RootManageSharedAccessKey;
SharedAccessKey=wNe+CrVSHJQfpnH0jWc2C01GYT/0sBsh5+6H4a"
eventhub_name = "demo2"

# URL de l'API OpenSky pour récupérer les données de la France (zone spécifiée)
url = "https://opensky-network.org/api/states/all?lamin=41.0&lomin=-5.0&lamax=51.0&lomax=10.0"

# Fonction pour envoyer les événements à Event Hub
def send_data_to_eventhub():
    try:
        # Obtenir les données en temps réel de l'API OpenSky avec authentification
        response = requests.get(url, auth=HTTPBasicAuth("WafaJB", "WafaApi123"))
        response.raise_for_status() # Vérifie si la requête a réussi
        data = response.json()
        states = data.get("states", [])
    
```

```

# Afficher le nombre de vols récupérés
print(f"Nombre de vols récupérés : {len(states)}")

# Création du Producteur pour connecter à Event Hub
producer = EventHubProducerClient.from_connection_string(connection_str,
eventhub_name=eventhub_name)
event_data_batch = producer.create_batch()

# Préparer les événements et les ajouter au lot
for state in states:
    # Construire les données du message
    event_data = {
        "icao24": state[0], # Identifiant unique du vol (adresse ICAO 24 bits)
        "callsign": state[1], # Numéro ou identifiant du vol
        "origin_country": state[2], # Pays d'origine du vol
        "time_position": state[3], # Heure de la dernière mise à jour de la position
        "last_contact": state[4], # Heure de la dernière mise à jour générale
        "longitude": state[5], # Longitude pour la cartographie
        "latitude": state[6], # Latitude pour la cartographie
        "baro_altitude": state[7], # Altitude barométrique pour la visualisation de la
hauteur
        "on_ground": state[8], # Indique si le vol est au sol
        "velocity": state[9], # Vitesse du vol
        "true_track": state[10], # Direction du mouvement
        "vertical_rate": state[11], # Taux de montée/descente
        "geo_altitude": state[13], # Altitude géométrique
        "squawk": state[14], # Code transpondeur
    }

    # Afficher chaque message avant de l'envoyer
    print("Message JSON à envoyer :")
    print(json.dumps(event_data, indent=2))

    # Ajouter l'événement au lot
    event_data_batch.add(EventData(json.dumps(event_data)))

# Envoyer le lot d'événements à Event Hub
producer.send_batch(event_data_batch)
producer.close()
print("Données envoyées à Event Hub avec succès")
except requests.RequestException as e:
    print(f"Erreur lors de la récupération des données : {e}")
except Exception as e:
    print(f"Erreur lors de l'envoi des données à Event Hub : {e}")

# Appel de la fonction pour envoyer des données toutes les 60 secondes
while True:
    send_data_to_eventhub()
    time.sleep(60)

```

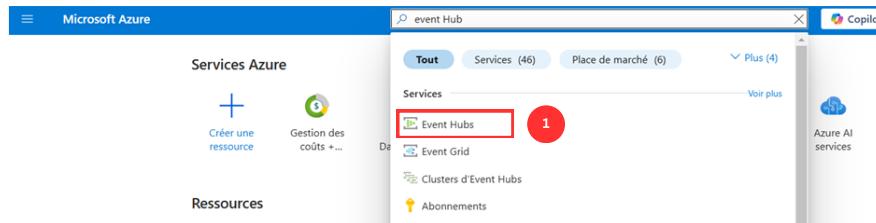
- **Création d'un *topic* dans un service de streaming (ex. : Kafka, AWS Kinesis Stream).**

- ⇒ Un **topic** est une structure utilisée dans les systèmes de streaming comme Kafka ou AWS Kinesis pour organiser les données en flux. Les producteurs publient des messages sur un topic, et les consommateurs s'abonnent pour lire ces messages. Cela permet une communication asynchrone et scalable entre les différentes parties d'une architecture.
- ⇒ Dans ce travail, on a choisi de travailler avec l'équivalent d'un topic dans Azure, à savoir **Event Hub**. Chaque Event Hub agit comme un canal dédié où les données en streaming sont envoyées par les producteurs et récupérées par les consommateurs. Cela correspond parfaitement à notre besoin de gérer des flux de données en temps

réel tout en bénéficiant des services natifs d'Azure pour la scalabilité, la sécurité et l'intégration.

⇒ Pour travailler avec Event Hubs, nous devons d'abord :

- **Créer un Event Hub Namespace** : Un namespace est l'espace logique regroupant plusieurs Event Hubs.



**Event Hubs**

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique (mesrs.tn)

+ Crée Gérer la vue Actualiser Exporter au format CSV Ouvrir une requête ...

Filtrer un champ... Abonnement égal à tout Ajouter un filtre Plus (2)

Affichage de 0 à 0 sur 0 enregistrements. Aucun regroupement Vue liste

Nom ↑↓	État ↑↓	Niveau ↑↓	Capacité ↑↓	E
--------	---------	-----------	-------------	---



### Aucun espaces de noms event hubs à afficher

Un espace de noms Event Hubs est un conteneur de gestion pour des hubs d'événements qui fournit des points de terminaison réseau intégrés au DNS et une plage de contrôle d'accès et de gestion de l'intégration réseau.

[Créer espace de noms event hubs](#)

2

[Découvrir plus d'informations sur Event Hubs](#)

Accueil > Event Hubs >

**Créer un espace de noms**

Event Hubs

Informations de base Avancé Réseau Étiquettes Vérifier + créer

Détails du projet

Abonnement \* Azure for Students

Groupe de ressources \* tpgroup

Détails de l'instance

Nom de l'espace de noms \* openskyDemo

Emplacement \* West Europe

Niveau tarifaire \* Standard (~\$22 USD par TU par mois)

Unités de débit \* 1

Activer l'augmentation automatique

Informations de base Validation réussie.

Informations de base Avancé Réseau Étiquettes Vérifier + créer

Espace de noms Event Hubs par Microsoft

Informations de base

Nom de l'espace de noms openskyDemo

Abonnement Azure for Students

Groupe de ressources tpgroup

Emplacement West Europe

Niveau tarifaire Standard

Unités de débit 1

Zones de disponibilité (redondance de zone) Activé

Augmentation automatique des unités de débit maximales Désactivé

Réseau

Méthode de connectivité Accès public

Sécurité

Version TLS minimale 1.2

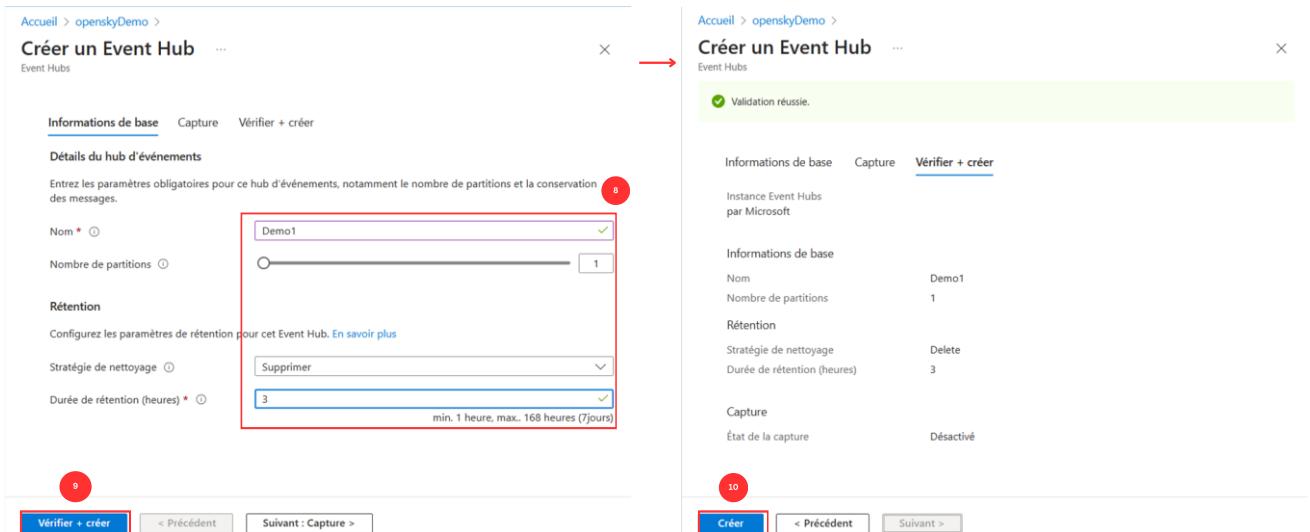
Authentification locale Activé

**Vérifier + créer** **Créer** < Précédent Suivant >

**(!)NB :**

- ⇒ **Unité de débit:** C'est la capacité de traitement de l'Event Hub. Une unité de débit détermine combien de données peuvent être envoyées et reçues par seconde. Plus nous avons d'unités de débit, plus l'Event Hub peut gérer de données simultanément.
  - ⇒ **Activer l'augmentation automatique :** Cette option permet à Event Hubs d'ajuster automatiquement le nombre d'unités de débit en fonction du volume de données. Si notre utilisation dépasse la capacité actuelle, Event Hubs augmentera les unités de débit pour s'adapter à la demande, nous évitant ainsi des interruptions.
- **Créer un Event Hub :** L'Event Hub agit comme une unité de gestion des messages, où les données sont envoyées (producteurs) et consommées (consommateurs), de manière similaire à un topic.

The screenshot shows two screenshots of the Azure portal interface. The top screenshot is titled 'Accueil > openskyDemo | Vue d'ensemble'. It displays deployment details for 'openSkyDemo' with a green checkmark indicating successful deployment. A red box highlights the 'Accéder à la ressource' button, which is circled with a red number '6'. The bottom screenshot is titled 'Accueil > openskyDemo Espace de noms Event Hubs'. It shows the 'Event Hub' blade with a red box highlighting the '+ Event Hub' button, which is circled with a red number '7'. Both screenshots show a sidebar with various navigation options like Entrées, Sorties, Modèle, Journal d'activité, Contrôle d'accès (IAM), Étiquettes, Diagnostiquer et résoudre les problèmes, Data Explorer, Événements, Paramètres, Entités, Supervision, Automatisation, and Aide.



### (!)NB :

- ⇒ **Nombre de partitions** : Une partition est une sous-division du flux de données permettant le traitement parallèle. Nous avons choisi un nombre minimal pour commencer, mais nous ajusterons cela lors de l'optimisation.
- ⇒ **Stratégie de nettoyage** : Cette stratégie définit comment les anciens messages sont nettoyés > supprimés.
- ⇒ **Durée de rétention** : Elle détermine combien de temps les données restent dans l'Event Hub avant d'être supprimées. Nous avons défini une valeur par défaut pour commencer.
- ⇒ **Capture** : Cette option permet de sauvegarder les messages dans un stockage à long terme (par exemple, Azure Blob Storage). Nous l'avons laissé désactivée pour l'instant, mais elle pourra être activée selon nos besoins futurs.

Nous avons opté pour des configurations simples, que nous mettrons à jour dans l'étape d'optimisation en fonction des performances et des besoins.

Et voilà donc on a créé notre **EventHub** au sein de **NameSpace Event Hubs** où on trouve :

CONTENU DE L'ESPACE D...	SURFACE KAFKA ACTIVÉ	REDONDANCE DE ZONE ACTIVÉ								
<b>1 EVENT HUB</b>										
<b>Event Hubs (1)</b> <hr/> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> <span style="color: blue;">🔍</span> Rechercher des éléments en les filtrant par nom...         </div> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding: 5px;">Nom</th> <th style="text-align: left; padding: 5px;">État</th> <th style="text-align: left; padding: 5px;">Rétention des messa...</th> <th style="text-align: left; padding: 5px;">Nombre de partitions</th> </tr> </thead> <tbody> <tr> <td style="padding: 5px;">demo1</td> <td style="padding: 5px;">Active</td> <td style="padding: 5px;">3 heures</td> <td style="padding: 5px;">1</td> </tr> </tbody> </table>			Nom	État	Rétention des messa...	Nombre de partitions	demo1	Active	3 heures	1
Nom	État	Rétention des messa...	Nombre de partitions							
demo1	Active	3 heures	1							

## (!)NB :

Dans l'interface du **Namespace Event Hubs**, voici ce que signifient les champs mentionnés :

- ⇒ **Contenu de l'espace** : Ce champ affiche le nombre d'Event Hubs créés dans le namespace. Cela nous permet de savoir combien de flux de données sont gérés dans cet espace.
- ⇒ **Surface Kafka > activée** : Cela indique si le namespace est compatible avec les applications utilisant le protocole Apache Kafka. En activant cette option, on peut utiliser des outils ou des clients Kafka avec Event Hubs.
- ⇒ **Redondance de zone > activée** : Cette option garantit une haute disponibilité en répliquant les données sur plusieurs zones géographiques d'Azure. Si activée, cela améliore la résilience en cas de panne, mais peut entraîner des coûts supplémentaires.

Pour la chaîne de connexion utilisée précédemment dans le script pour envoyer des données à Event Hub, voici où nous pouvons la récupérer :

The screenshot shows two windows from the Azure portal. The left window is titled 'openSkyDemo | Stratégies d'accès partagé' and displays a list of shared access keys under the 'Stratégies d'accès partagé' section. One key, 'RootManageSharedAccessKey', is selected and highlighted with a red box. A red circle with the number '1' is placed over the 'Stratégies d'accès partagé' link in the sidebar. The right window is titled 'Stratégie SAP : RootManageSharedAccessKey' and provides detailed settings for this key. It includes sections for 'Gérer', 'Envoyer', and 'Écouter'. Under 'Clé primaire de la chaîne de connexion', the endpoint URL is shown as 'Endpoint=sb://openskydemo.servicebus.windows.net/SharedAccessKey?name=RootMa...'. A red circle with the number '3' is placed over this URL. A red arrow points from the 'Envoyer' button in the left window to the right window.

Cette chaîne de connexion est essentielle pour authentifier le script avec Event Hubs et établir la communication.

- **Utilisation d'un outil de traitement (ex. : Apache Spark Streaming) pour :**
  - **Filtrer les données inutiles.**
  - **Aggréger les données en temps réel.**
  - **Sauvegarder les résultats dans une base de données ou stockage cloud.**

⇒ Dans cette étape, nous avons utilisé deux outils de traitement des données en temps réel pour répondre aux besoins de filtrage, d'agrégation et de stockage des résultats : **Apache Spark Streaming** et **Azure Stream Analytics**. Voici une explication simple de notre approche et une comparaison des deux outils :

⇒ **Spark Streaming :**

- Nous avons utilisé **Spark Streaming** pour démontrer le traitement des flux en temps réel dans un environnement Databricks.
- Les données inutiles ont été **filtrées**, des calculs ont permis **l'agrégation des données en temps réel**, et les résultats ont été **stockés dans Azure Blob Storage**, une solution de stockage cloud.

⇒ **Azure Stream Analytics :**

- Nous avons également utilisé **Stream Analytics** pour accomplir des tâches similaires. Avec cet outil, nous avons traité les flux de données en définissant des requêtes SQL-like simples pour filtrer et agréger les informations avant de les stocker directement dans **Azure Blob Storage**.

Critère	Spark Streaming	Stream Analytics
<b>Flexibilité</b>	Très flexible grâce à l'utilisation de code (Python, Scala).	Moins flexible, mais plus simple à configurer.
<b>Interface</b>	Basée sur des notebooks (Databricks).	Interface graphique conviviale.
<b>Langage utilisé</b>	Supporte plusieurs langages comme Python et Scala.	SQL-like pour les requêtes.
<b>Complexité</b>	Plus adapté aux cas d'usage complexes ou avancés.	Idéal pour les scénarios simples et rapides.
<b>Performance</b>	Optimisé pour les traitements complexes et volumineux.	Performant pour des flux de données modérés.
<b>Intégration</b>	Compatible avec de nombreux services et technologies.	Étroitement intégré dans l'écosystème Azure.

Avant de commencer ce travail, il est important d'avoir une **ressource Blob Storage** correctement configurée avec un **conteneur**.

- **Blob Storage** : C'est un service de stockage cloud proposé par Azure. Il permet de stocker de grandes quantités de données non structurées, comme des fichiers, des images ou des données de flux.
- **Conteneur** : Dans Blob Storage, un conteneur est une unité logique où les blobs (fichiers) sont organisés. On peut y penser comme un dossier dans lequel on stocke les données. Chaque conteneur appartient à une ressource Blob Storage.

## 1. Configuration de Blob Storage :

**Services Azure**

**Créer un compte de stockage**

**Créer un compte de stockage**

**openskystorageaccount\_1735384687504 | Vue d'en...**

**Nouveau conteneur**

Pour accéder à cette ressource et y stocker des données, il est nécessaire d'utiliser une **clé d'accès** ou une **chaîne de connexion**. Pour récupérer ces identifiants :

The screenshot shows the Azure Storage Account keys page for 'openskystorageaccount'. The left sidebar has a 'Clés d'accès' section highlighted with a red box. A red circle with the number 10 is on the sidebar. The main area shows the 'key1' section with a 'Clé' input field containing a redacted value and an 'Afficher' button. Below it is a 'Chaîne de connexion' input field with a redacted value and an 'Afficher' button. A red box highlights these two fields. A red circle with the number 11 is on the right side of the page.

## 2. Spark Streaming :

Pour Spark Streaming, nous avons besoin d'un notebook dans notre espace Databricks (créé en suivant les mêmes étapes que pour le notebook du script Producer). Dans ce notebook, nous allons monter le Blob Storage en utilisant les identifiants récupérés précédemment, puis écrire un script qui consommera les données générées par le script Producer.

**(!) NB :** Les deux scripts (Producer et Consumer) doivent être exécutés simultanément pour assurer le traitement en temps réel.

- **Monter le Blob Storage :**

```
# paramètres d'accès du notre Azure Blob Storage
STORAGE_ACCOUNT_NAME = "openskystorageaccount"
STORAGE_ACCOUNT_KEY =
"HTGNSbe516CqKCFMq+UMGqApc4pwEJuwjfR0gREuK8KkEb1jVvUGrg+Ac=="
CONTAINER_NAME = "mycontainer"
MOUNT_POINT = "/mnt/mycontainer"

# Monter le stockage Azure Blob Storage

try:
    dbutils.fs.mount(
        source=f"wasbs://{CONTAINER_NAME}@{STORAGE_ACCOUNT_NAME}.blob.core.windows.net",
```

```

        mount_point=MOUNT_POINT,
        extra_configs={f"fs.azure.account.key.{STORAGE_ACCOUNT_NAME}.blob.core.windows.net": STORAGE_ACCOUNT_KEY}
    )
    print(f"Monté avec succès {CONTAINER_NAME} à {MOUNT_POINT}.")
except Exception as e:
    print(f"Azure Blob Storage déjà monté ou erreur : {e}")

```

Monté avec succès mycontainer à /mnt/mycontainer.

- Consommer les données :

```

from pyspark.sql import SparkSession
from pyspark.sql.functions import col, from_json, decode, unix_timestamp, count, avg, when, window
from pyspark.sql.types import StructType, StructField, StringType, LongType, FloatType, BooleanType
import time

# Initialiser une session Spark
# Spark est utilisé pour traiter des flux de données en temps réel ou des grandes quantités de données distribuées.
spark = SparkSession.builder.appName("OpenSkyStreamingConsumer").getOrCreate()

# Définir la chaîne de connexion à Event Hubs
# Event Hubs est un service de streaming permettant de recevoir les données en temps réel.
connection_str =
"Endpoint=sb://openskydemo.servicebus.windows.net/;SharedAccessKeyName=RootManageSharedAccessKey;SharedAccessKey=J4w3v04068bNxmkFVwrpVzvC0gM8UtJBj+AEh02SJ5A=;EntityPath=demo2"

# Chiffrer la chaîne de connexion pour des versions Spark compatibles (>= 2.3.15)
# Cela permet de sécuriser les identifiants de connexion en les chiffrant.
encrypted_connection_str =
spark.sparkContext._jvm.org.apache.spark.eventhubs.EventHubsUtils.encrypt(connection_str)

# Configuration des paramètres pour Event Hubs
ehConf = {
    'eventhubs.connectionString': encrypted_connection_str,
    'eventhubs.eventHubName': 'demo2',
}

# Lire le flux brut depuis Event Hubs
# Cette partie permet de consommer les données en direct à partir d'Event Hubs.
raw_stream = spark.readStream \
    .format("eventhubs") \
    .options(**ehConf) \
    .load()

# Décoder la colonne 'body' encodée en Base64 en une chaîne de caractères lisible
decoded_stream = raw_stream.selectExpr("CAST(body AS STRING)").alias("raw_body")

# Définir le schéma des données JSON reçues
# Cette structure permet de décrire le format attendu des données pour une extraction correcte.
schema = StructType([
    StructField("icao24", StringType(), True), # Identifiant unique de l'avion
    StructField("callsign", StringType(), True), # Indicateur d'appel
    StructField("origin_country", StringType(), True), # Pays d'origine
])

```

```

StructField("time_position", LongType(), True), # Temps (en Unix epoch)
StructField("last_contact", LongType(), True), # Dernier contact
StructField("longitude", FloatType(), True), # Longitude
StructField("latitude", FloatType(), True), # Latitude
StructField("baro_altitude", FloatType(), True), # Altitude barométrique
StructField("on_ground", BooleanType(), True), # Indique si l'avion est au
sol
StructField("velocity", FloatType(), True), # Vitesse
StructField("true_track", FloatType(), True), # Cap vrai
StructField("vertical_rate", FloatType(), True), # Taux de montée/descente
StructField("geo_altitude", FloatType(), True), # Altitude géographique
StructField("squawk", StringType(), True), # Code transpondeur
])

# Parser les données JSON reçues à l'aide du schéma défini
parsed_stream = decoded_stream.select(from_json(col("body"),
schema).alias("data"))

# Extraire les colonnes nécessaires depuis les données JSON parsées
final_stream = parsed_stream.select(
    "data.icao24", "data.callsign", "data.origin_country", "data.time_position",
    "data.longitude", "data.latitude", "data.baro_altitude", "data.velocity",
    "data.vertical_rate", "data.geo_altitude", "data.on_ground"
).withColumn(
    # Ajouter une colonne pour catégoriser le statut du vol
    "flight_status", when(col("on_ground") == True, "On Ground") # Au sol
        .when(col("vertical_rate") > 2, "Climbing") # Montée
        .when(col("vertical_rate").between(-2, 2), "Cruising") #
Croisière
        .when(col("vertical_rate") < -2, "Descending") # Descente
)

# Supprimer les lignes ayant des valeurs manquantes
# Ceci améliore la qualité des données analysées.
final_stream = final_stream.na.drop()

# Convertir 'time_position' (en secondes depuis l'époque Unix) en format timestamp
final_stream = final_stream.withColumn("time_position", (col("time_position") /
1000).cast("timestamp"))

# Ajouter une watermark pour gérer les données tardives
# Cela permet d'ignorer les données arrivant avec un retard supérieur à 1 minute.
final_stream = final_stream.withWatermark("time_position", "1 minute")

final_stream.display()

# Effectuer des agrégations sur une fenêtre de temps glissante
windowed_stream = final_stream.groupBy(
    window(col("time_position"), "1 minute", "10 seconds"), # Fenêtre de 1 minute
    glissante toutes les 10 secondes
    "origin_country"
).agg(
    count("icao24").alias("number_of_flights"), # Compte le nombre de vols
    avg("velocity").alias("avg_velocity"), # Vitesse moyenne
    avg("geo_altitude").alias("avg_altitude"), # Altitude moyenne
    count(when(col("on_ground") == True, 1)).alias("on_ground_count"), # Nombre
    d'avions au sol
    count(when(col("flight_status") == "Climbing", 1)).alias("climbing_count"), # Nombre
    d'avions en montée
    count(when(col("flight_status") == "Cruising", 1)).alias("cruising_count"), # Nombre
    d'avions en croisière
    count(when(col("flight_status") == "Descending",
1)).alias("descending_count") # Nombre d'avions en descente
)

# Aplatir la colonne 'window' (STRUCT) pour pouvoir la sauvegarder dans un format
JSON
windowed_stream_flattened = windowed_stream.select(

```

```

    col("window.start").alias("window_start"), # Début de la fenêtre
    col("window.end").alias("window_end"), # Fin de la fenêtre
    "origin_country",
    "number_of_flights",
    "avg_velocity",
    "avg_altitude",
    "on_ground_count",
    "climbing_count",
    "cruising_count",
    "descending_count"
)
windowed_stream_flattened.display()

# Sauvegarder les résultats dans un stockage temporaire (mémoire)
windowed_stream_flattened.writeStream \
    .outputMode("append") \
    .format("json") \
    .option("path", "/mnt/flight_stream")

```

### Explication du traitement effectué :

#### ⇒ Nettoyage des données :

- **Suppression des données inutiles** : Les lignes contenant des valeurs manquantes ou corrompues sont éliminées avec la méthode `.na.drop()`. Cela garantit que seules les données complètes et exploitables sont conservées.
- **Décodage des données en Base64** : Les messages bruts reçus sont décodés pour obtenir un format lisible (texte clair).
- **Parsing JSON** : Les données reçues au format JSON sont extraites et converties en colonnes structurées selon un schéma prédéfini.

#### ⇒ Agrégations appliquées :

- **Ajout de colonnes dérivées** : Création de la colonne `flight_status`, qui catégorise l'état du vol en fonction des valeurs (ex. : "On Ground", "Climbing", "Cruising", "Descending").
- **Conversion des timestamps** : La colonne `time_position` (en secondes depuis l'époque Unix) est convertie en un format **timestamp** pour faciliter les analyses temporelles.
- **Fenêtre glissante** : Les données sont regroupées par une fenêtre de temps glissante (1 minute, glissant toutes les 10 secondes) pour effectuer des agrégations. Cela permet de suivre les événements en temps réel de manière structurée.

- Agrégations effectuées :

- Nombre total de vols (count).
- Moyenne de la vitesse (avg).
- Moyenne de l'altitude (avg).
- Comptage des vols au sol ou dans différents statuts (count conditionnel).

#### ⇒ Stockage des données :

- Les données agrégées sont sauvegardées dans un conteneur Azure Blob Storage monté sous le chemin `mnt/flight_stream`. Avant le stockage, la colonne `window` (STRUCT) est aplatie en deux colonnes distinctes : `window_start` et `window_end`,

correspondant respectivement au début et à la fin de chaque fenêtre temporelle. Les autres colonnes résultant des agrégations (comme le nombre total de vols, la vitesse moyenne, etc.) sont également sélectionnées pour être structurées correctement.

- Les données sont ensuite écrites en streaming au format JSON, en mode append, ce qui permet d'ajouter de nouveaux résultats sans écraser les données précédemment stockées. Cette approche garantit que les données sont disponibles pour une utilisation ultérieure

Alors l'exécution de nos scripts donne comme résultat :

#### ❖ Dans la console Databricks :

Voici une explication détaillée des résultats obtenus dans la console, des jobs exécutés, et des graphiques de performances observés :

	ICAO24	callsign	origin_country	time_position	longitude	latitude	baro_altitude	
1	4b1819	SWR87C	Switzerland	1970-01-21T02:04:34.505+00....	5.873499870300293	45.02629852294922	10668	23
2	4b1802	SWR48X	Switzerland	1970-01-21T02:04:34.506+00....	5.107999801635742	47.11909866333008	8808.7197265625	20
3	4952ca	TAP434	Portugal	1970-01-21T02:04:34.505+00....	-3.42449998555908	44.93949890136719	10363.2001953125	22
4	4952c9	TAP52UP	Portugal	1970-01-21T02:04:34.506+00....	-2.8487000465393066	48.6411018371582	10363.2001953125	24
5	4952c4	TAP1034	Portugal	1970-01-21T02:04:34.505+00....	0.3635999858379364	41.35710144042969	6979.919921875	17
6	488252	SAH48P	Poland	1970-01-21T02:04:34.505+00....	-4.19350004196167	48.22309875488281	11894.8203125	19

	window_start	window_end	origin_country	number_of_flights	avg_velocity	avg_altitude
1	1970-01-21T02:04:30.000+00...	1970-01-21T02:05:30.000+00...	Switzerland	40	163.41224966049194	6541.198516845703
2	1970-01-21T02:04:00.000+00...	1970-01-21T02:05:00.000+00...	Malta	47	208.0372334744068	9480.090617727727
3	1970-01-21T02:04:10.000+00...	1970-01-21T02:05:10.000+00...	Lithuania	1	224.10000610351562	10873.740234375
4	1970-01-21T02:04:20.000+00...	1970-01-21T02:05:20.000+00...	Bulgaria	2	159.9849967956543	5989.320129394531
5	1970-01-21T02:04:00.000+00...	1970-01-21T02:05:00.000+00...	Algeria	8	207.64499759674072	9789.794998168945
6	1970-01-21T02:03:40.000+00...	1970-01-21T02:04:40.000+00...	Malta	47	208.0372334744068	9480.090617727727
7	1970-01-21T02:04:10.000+00...	1970-01-21T02:05:10.000+00...	Bulgaria	2	159.9849967956543	5989.320129394531
8	1970-01-21T02:04:10.000+00...	1970-01-21T02:05:10.000+00...	Canada	4	233.39500045776367	10673.71484375

#### ⇒ Les Jobs et leurs étapes :

- ✓ Un **job** représente une tâche globale, générée par Spark pour traiter ou transformer les données de streaming.
- ✓ Un **stage** est une sous-division du job, correspondant à une étape logique nécessitant une action ou une transformation spécifique. Les stages sont divisés en fonction des partitions des données ou des transformations appliquées.

Dans nos résultats :

- ✓ Le **Job 4** comporte **deux stages**. Cela correspond aux opérations simples, comme l'affichage des données nettoyées (après le décodage et le parsing).
- ✓ Le **Job 5** comporte **trois stages**. Ce job inclut des transformations plus complexes, comme les agrégations temporelles sur des fenêtres glissantes (calcul des moyennes, totaux, ou comptages) qui nécessitent des regroupements et des réorganisations de partitions.

**(!) NB :** La différence dans le nombre de stages entre les jobs est directement liée à la complexité des transformations effectuées.

**(!) NB :** Si on lance pas le script Producer en parallèle avec ce script consumer (SparkStreaming) les jobs seront lancés et le streaming sera initialisé mais rien n'est affiché dans les dataframes car pas de données et donc on verra une résultat dans le console comme ceci

```

4 jobs Spark [progress bar]
  ↗ Initialisation du Stream...
  ↗ Initialisation du Stream...
  ↗ Initialisation du Stream...
  ↗ display_query_26 (id: f6554505-40c6-4d3e-8def-9b90e0e214d6)   Dernière mise à jour : Il y a 10 secondes
  ↗ decoded_stream: pyspark.sql.dataframe.DataFrame = [body: string]
  ↗ final_stream: pyspark.sql.dataframe.DataFrame = [icao24: string, callsign: string ... 10 autres champs]
  ↗ parsed_stream: pyspark.sql.dataframe.DataFrame = [data: struct]
  ↗ raw_stream: pyspark.sql.dataframe.DataFrame = [body: binary, partition: string ... 7 autres champs]
  ↗ windowed_stream: pyspark.sql.dataframe.DataFrame = [window: struct, origin_country: string ... 7 autres champs]
  ↗ windowed_stream_flattened: pyspark.sql.dataframe.DataFrame = [window_start: timestamp, window_end: timestamp ... 8 autres champs]
  <pyspark.sql.streaming.StreamingQuery at 0x7f7615f43110>

```

Table	+	Q	Y	□			
	window	origin_country	number_of_flights	avg_velocity	avg_altitude	on_ground_count	climbing_count
Aucune ligne renvoyée							

**(!) NB :** Exemple de résultat du script Producer :

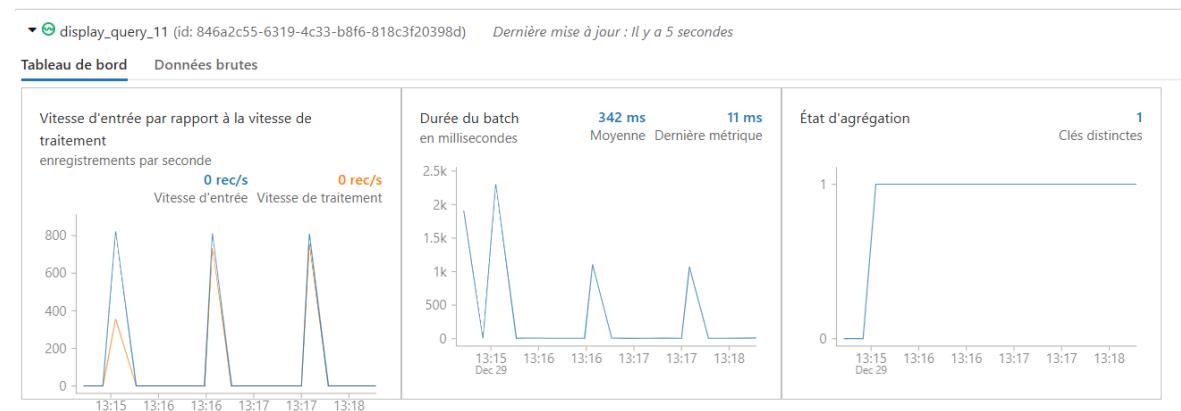
```

Message JSON à envoyer :
{
    "icao24": "398571",
    "callsign": "AFR1502",
    "origin_country": "France",
    "time_position": 1735162296,
    "last_contact": 1735162297,
    "longitude": 2.7717,
    "latitude": 48.5657,
    "baro_altitude": 5562.6,
    "on_ground": false,
    "velocity": 189.05,
    "true_track": 162.26,
    "vertical_rate": 11.38,
    "geo_altitude": 10652.76,
    "squawk": "1000"
}
Données envoyées à Event Hub avec succès

```

⇒ **Données des DataFrames affichés :**

- ✓ **DataFrame final\_stream** : Données nettoyées après réception via Event Hub
- ✓ **DataFrame windowed\_stream\_flattened** : DataFrame final après agrégations



## ⇒ Graphiques de performances Spark

Les performances des jobs Spark Streaming sont visualisées à travers trois graphiques principaux, qui permettent d'évaluer l'efficacité du traitement des données en streaming.

- **Vitesse d'entrée vs. vitesse de traitement :**
- ✓ Ce graphique montre le nombre d'enregistrements traités par Spark par rapport au nombre d'enregistrements entrants.

Dans nos résultats :

- ✓ Les **pics** représentent les périodes où les données ont été ingérées et traitées efficacement.
- ✓ Les **creux** indiquent des moments où le flux entrant était moins dense ou temporairement réduit.

- **Durée du batch**

- ✓ Ce graphique mesure le temps nécessaire pour traiter chaque lot (micro-batch).

Dans nos résultats :

- ✓ Les durées faibles (par exemple, 7 ms) reflètent des opérations simples, comme le décodage ou l'affichage des données.
- ✓ Les durées plus élevées (par exemple, > 2000 ms) apparaissent lors de transformations plus complexes, comme les agrégations sur des fenêtres glissantes.

**(!!!) NB :** Lorsqu'on travaille avec **Spark Streaming**, il peut sembler choquant de voir des graphes de "Batch" dans la console ou les résultats des jobs, alors qu'on parle de streaming. Cette confusion provient de la manière dont Spark Streaming gère les données en interne.

Par exemple : Si on configure un intervalle de batch de 5 secondes, Spark accumule les données reçues pendant 5 secondes, puis les traite comme un seul lot.

Spark Streaming est techniquement **un traitement par micro-batch** déguisé en traitement continu. Cela peut prêter à confusion, mais c'est une solution puissante pour traiter des flux de données de manière efficace et scalable.

- **État d'agrégation (Clés distinctes)**

- ✓ Ce graphique indique le nombre de clés uniques (par exemple, origin\_country) actuellement suivies dans les agrégations.

Dans nos résultats :

- ✓ Un nombre élevé de clés peut augmenter la complexité et le temps de traitement, surtout si les données nécessitent un regroupement et des calculs sur plusieurs fenêtres.

- ❖ **SparkUI :**

Pour accéder à l'interface utilisateur de spark dans Daatbricks on peut soit accéder via le job dans le console directement :



Soit via le cluster des ressources associé pour notre script :

The screenshot shows the Databricks interface with the 'Calculer' section selected. The sidebar on the left has a 'Calculer' tab highlighted with a red box and a red circle containing the number '1'. In the main area, a table lists clusters, with 'Cluster1' highlighted by a red box and a red circle containing the number '2'. Below the table, the 'Cluster1' details page is shown, with the 'Interface utilisateur Spark' tab highlighted by a red box and a red circle containing the number '3'.

Voici une explication détaillée des résultats observés dans les différents sections de SparkUI :

## ⇒ Executors

The screenshot shows the SparkUI Executors section. At the top, there is a summary table with columns for Active, Dead, and Total executors, along with various metrics like Storage Memory, Disk Used, and Task Times. Below this is a detailed table for each executor, showing more granular metrics such as Address, Status, and specific task counts.

	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write	Excluded
<b>Active(1)</b>	0	855.2 Kib / 3.3 GiB	0.0 B	4	0	0	1240	1240	59.5 h (11 s)	0.0 B	282 Kib	282 Kib	0
<b>Dead(0)</b>	0	0.0 B / 0.0 B	0.0 B	0	0	0	0	0	0.0 ms (0.0 ms)	0.0 B	0.0 B	0.0 B	0
<b>Total(1)</b>	0	855.2 Kib / 3.3 GiB	0.0 B	4	0	0	1240	1240	59.5 h (11 s)	0.0 B	282 Kib	282 Kib	0

Executor ID	Address	Status	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write	Thread Dump	Heap Histogram	Add Time	Remove Time
driver	10.139.64.4:45489	Active	0	855.2 Kib / 3.3 GiB	0.0 B	4	0	0	1240	1240	59.5 h (11 s)	0.0 B	282 Kib	282 Kib	Thread Dump	Heap Histogram	2024-12-27 01:46:16	-

Cette section affiche les ressources utilisées par Spark :

- **Active Tasks** : Nombre de tâches actuellement exécutées.
- **Task Time** : Temps total passé à exécuter les tâches sur chaque exécuteur.
- **Shuffle Read/Write** : Quantité de données lues/écrites lors des opérations de partitionnement.
- **Memory Usage** : Mémoire utilisée par les exécuteurs.

Dans notre script :

- **Nombre total d'exéuteurs actifs** : On n'utilise qu'un seul exécuteur (le driver) pour exécuter les tâches. Cela est dû à l'abonnement Databricks Standard qui permet d'avoir un cluster avec un seul **Exécuteur = Driver** (pratiquement comme si un cluster avec une seule machine). Malheureusement, on n'a pas pu travailler avec l'abonnement **Premium**, qui aurait permis une gestion plus avancée des ressources, en raison de notre budget limité sur Azure (moins de 50 \$ pour réaliser l'ensemble de ce TP).
- **Shuffle Write et Read** : Une partie de notre script (agrégations avec fenêtre glissante) entraîne des opérations de partitionnement, expliquant les volumes modestes observés dans les shuffles.

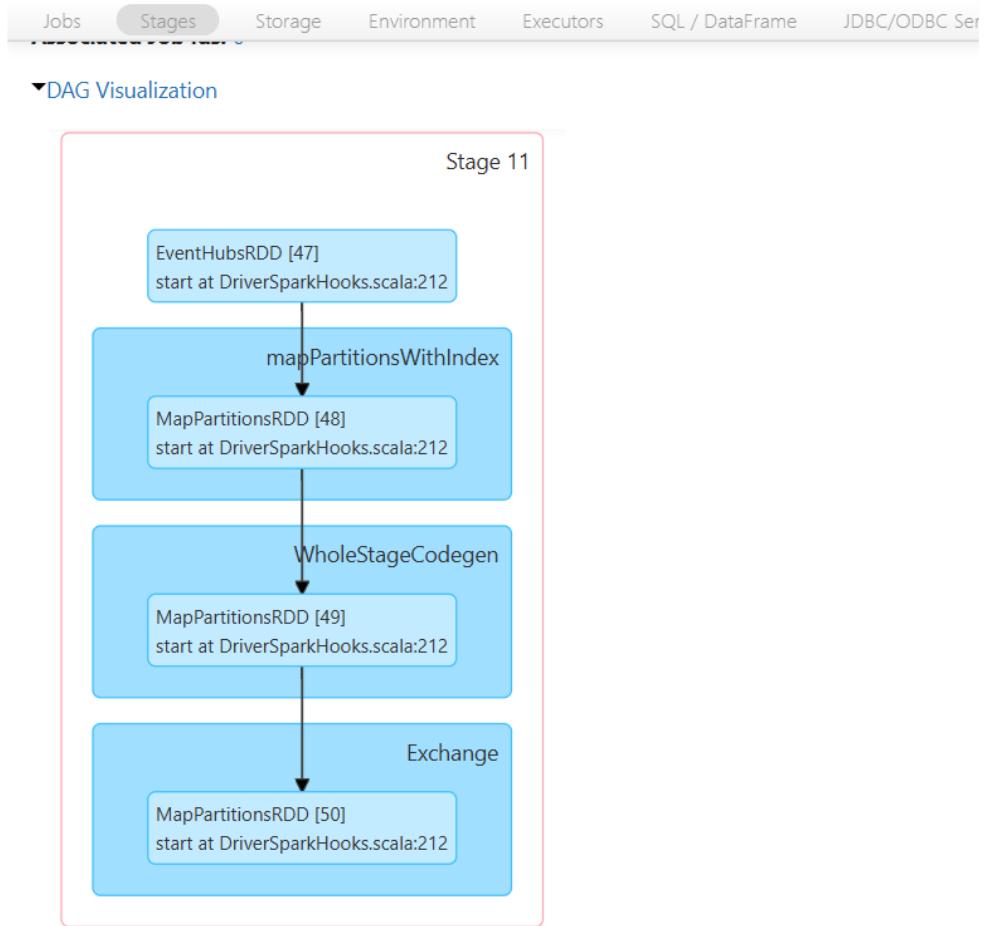
⇒ **Jobs :**

Jobs						
Job Id (Job Group) ▾	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total	
7 (d45cc4a0-d149-4741-b39a-95461515d39e)	display_query_12 id = c1c9f919-d684-4bb1-bf80-9e2e8c0192a5 runId = d45cc4a0-d149-4741... start at DriverSparkHooks.scala:212	2024/12/29 12:16:34	9 s	3/3	204/204	
6 (14cb8294-8dce-4158-ba2c-8729812d63f8)	display_query_11 id = 846a2c55-6319-4c33-b8f6-818c3f20398d runId = 14cb8294-8dce-4158... start at DriverSparkHooks.scala:212	2024/12/29 12:16:34	0.6 s	2/2	4/4	
5 (d45cc4a0-d149-4741-b39a-95461515d39e)	display_query_12 id = c1c9f919-d684-4bb1-bf80-9e2e8c0192a5 runId = d45cc4a0-d149-4741... start at DriverSparkHooks.scala:212	2024/12/29 12:15:33	11 s	3/3	204/204	
4 (14cb8294-8dce-4158-ba2c-8729812d63f8)	display_query_11 id = 846a2c55-6319-4c33-b8f6-818c3f20398d runId = 14cb8294-8dce-4158... start at DriverSparkHooks.scala:212	2024/12/29 12:15:33	2 s	2/2	4/4	
3 (d45cc4a0-d149-4741-b39a-95461515d39e)	display_query_12 id = c1c9f919-d684-4bb1-bf80-9e2e8c0192a5 runId = d45cc4a0-d149-4741... start at DriverSparkHooks.scala:212	2024/12/29 12:15:15	12 s	3/3	204/204	
2 (14cb8294-8dce-4158-ba2c-8729812d63f8)	display_query_11 id = 846a2c55-6319-4c33-b8f6-818c3f20398d runId = 14cb8294-8dce-4158... start at DriverSparkHooks.scala:212	2024/12/29 12:15:14	0.3 s	2/2	4/4	
1 (9fb0e4b-94bd-4b57-ab3d-7308687e97ec)	display_query_5 id = 9f537bc8-97a6-4c74-a05c-1eaccf0ff77d runId = 9fb0e4b-94bd-4b57-a... start at DriverSparkHooks.scala:212	2024/12/29 10:08:59	28 s	3/3	204/204	
0 (f7c8a66e-ba93-480a-8260-75f1529485bb)	display_query_6 id = 4c386e3e-faf7-4231-9045-0717db56ed6c runId = f7c8a66e-ba93-480a-8... start at DriverSparkHooks.scala:212	2024/12/29 10:08:59	16 s	3/3	204/204	

Cette section représente la liste des jobs déclenchés par notre application Spark, incluant des détails tels que la date de soumission, la durée d'exécution, le nombre d'étapes réussies et le nombre de tâches.

- **Une tâche** correspond à une unité d'exécution d'une partition sur un nœud de calcul. Le nombre de tâches est directement lié au partitionnement, qui consiste à diviser les données en morceaux pour les traiter en parallèle.
- **Le partitionnement** dépend de la source et des transformations effectuées : par exemple, notre source Event Hubs a un paramètre de partitionnement fixé à 3, donc Spark répartit initialement les données en 3 partitions. Cependant, pour des traitements plus complexes, comme agrégation par fenêtre glissante, Spark ajoute des partitions (par exemple, 240 partitions) pour optimiser l'exécution et gérer la charge.

⇒ **Stages :**



L'interface **Stages** de SparkUI nous permet d'explorer en détail les étapes d'exécution d'un job Spark via son DAG (Directed Acyclic Graph). Ce DAG représente les différentes transformations appliquées aux données pour produire le résultat final, en suivant un enchaînement précis des étapes.

Dans notre cas, le DAG du **Stage 11** illustre les transformations spécifiques sur les données streamées depuis Event Hubs. Chaque bloc du DAG correspond à une étape clé :

- **EventHubsRDD [47]** : C'est la source brute des données, directement extraite des flux Event Hubs. Cette étape initiale indique comment Spark lit et partitionne les données pour les traitements suivants.
- **mapPartitionsWithIndex** : Une transformation appliquée sur chaque partition, permettant de préparer les données (par exemple, parsing JSON ou ajout de métadonnées).
- **WholeStageCodegen** : Une optimisation où Spark regroupe plusieurs transformations simples (comme map ou filter) dans un pipeline unique pour améliorer les performances.
- **Exchange** : C'est une étape de shuffle, où les données sont redistribuées entre partitions pour permettre des calculs dépendant d'un regroupement, comme les agrégations ou fenêtres temporelles.

- **mapPartitionsRDD [50]** : Une transformation finale appliquée après le regroupement, où les calculs finaux sont exécutés (comme les sommes ou moyennes) avant de rendre les résultats exploitables.

Ce DAG met en évidence comment Spark optimise les calculs tout en gérant efficacement les flux de données, en minimisant les coûts des étapes coûteuses comme les shuffles et en maximisant le parallélisme via le partitionnement.

## ⇒ SQL/DataFrame

The screenshot shows the SparkUI interface with the 'SQL / DataFrame' tab selected. It displays two main sections: 'Running Queries' and 'Completed Queries'.

**Running Queries:**

ID	Description	Submitted	Duration	Running Job IDs	Succeeded Job IDs	Failed Job IDs	Sub Execution IDs
178	display_query_12 id = c1c9f919-d684-4bb1-bf80-9e2e8c0192a5 runId = d45cc4a0-d149-4741... +details	2024/12/29 12:19:38	6 s				[179] +details

**Completed Queries:**

ID	Description	Submitted	Duration	Job IDs	Sub Execution IDs
180	collectResult at OutputAggregator.scala:340 +details	2024/12/29 12:19:39	26 ms		
176	display_query_11 id = 846a2c55-6319-4c33-b8f6-818c3f20398d runId = 14cb8294-8dce-4158-ba2c-8729812d63f8 ba... +details	2024/12/29 12:19:38	0.7 s	[177]	+details
175	collectResult at OutputAggregator.scala:340	2024/12/29 12:19:31	20 ms		

L’interface **SQL/DataFrame** de SparkUI affiche les requêtes SQL ou DataFrame exécutées dans Spark, en deux catégories :

- **Running Queries** : Liste des requêtes actuellement en cours d’exécution.
- **Completed Queries** : Historique des requêtes terminées, avec des détails comme la durée et les stages associés.

Dans notre cas :

- Les requêtes actives incluent celles en streaming, comme display\_query\_5, qui effectue des agrégations ou transformations en temps réel.
- Les requêtes terminées, quant à elles, regroupent souvent des tests intermédiaires, tels que des appels collect, utilisés pour vérifier les résultats ou les étapes de transformations.

Cette section nous aide à surveiller en temps réel l’état des calculs SQL/DataFrame, à analyser leurs performances et à repérer les éventuels problèmes dans leur exécution.

## ⇒ Streaming Query

The screenshot shows the Apache Spark Streaming UI interface. At the top, there's a navigation bar with links like Jobs, Stages, Storage, Environment, Executors, SQL / DataFrame, JDBC/ODBC Server, Structured Streaming, and Connect. Below the navigation bar, the title "Streaming Query" is displayed. There are two sections: "Active Streaming Queries (2)" and "Completed Streaming Queries (10)".

**Active Streaming Queries (2):**

Name	Status	ID	Run ID	Start Time	Duration	Avg Input /sec	Avg Process /sec	Latest Batch
display_query_12	RUNNING	c1c9f919-d684-4bb1-bf80-9e2e80c192a5	d45cc4a0-d149-4741-b39a-95461515d39e	2024/12/29 12:15:13	4 minutes 54 seconds	679.38	64.36	5
display_query_11	RUNNING	846a2c55-6319-4c33-b8f6-818c3f20398d	14cb8294-8dce-4158-ba2c-8729812d63f8	2024/12/29 12:15:12	4 minutes 55 seconds	678.83	583.04	5

**Completed Streaming Queries (10):**

Name	Status	ID	Run ID	Start Time	Duration	Avg Input /sec	Avg Process /sec	Latest Batch	Error
display_query_10	FINISHED	7de4895a-7674-416e-b013-e14bd7fc0b98	b6d0e4aa-aea6-48ea-8d5c-e8b78f501ad2	2024/12/29 12:14:23	0 ms	NaN	NaN	NaN	-
display_query_9	FAILED	c91a135d-8fe1-4cf2-bb5e-4970dbffaa7	00cebedd-7bc4-4ccf-a0e1-	2024/12/29 12:14:22	0 ms	NaN	NaN	NaN	com.microsoft.azure.eventhubs.CommunicationException: A communication error has occurred. This may be due to an

L'interface **Streaming Query** est dédiée aux requêtes en streaming, avec des informations clés :

- **Active Streaming Queries** : Liste des requêtes de streaming en cours.
- **Avg Input** : Nombre moyen d'enregistrements entrants par seconde.
- **Avg Process** : Vitesse moyenne de traitement des enregistrements.
- **Completed Streaming Queries** : Historique des requêtes terminées, y compris les éventuelles erreurs rencontrées.

Dans notre cas :

- Deux requêtes actives, display\_query\_2\_1 et display\_query\_5, traitent les données en temps réel depuis Event Hubs.
- ✓ Un **Avg Input** élevé (678 et 583 événements par seconde) montre que le flux de données est constant.
- ✓ Un **Avg Process** efficace indique que Spark gère bien le traitement sans ralentissement notable.
- Une erreur sur une requête terminée révèle un problème de connexion à Event Hubs, probablement lié à des délais ou interruptions réseau.

Cette section est cruciale pour diagnostiquer les performances et la fiabilité des pipelines de streaming.

### 3. Stream Analytics :

Alors, en venant à notre deuxième méthode, qui est **Stream Analytics**, on constate qu'elle est très facile à manipuler. On commence par :

- ✓ Crée une tâche **Stream Analytics**

**Screenshot 1:** Accueil > Place de marché > Nouvelle tâche Stream Analytics. Le filtre recherche "stream-analytics".

**Screenshot 2:** Place de marché > Tâche Stream Analytics. Un nouveau tâche nommé "SAOpenSky" est créé.

**Screenshot 3:** Nouvelle tâche Stream Analytics. Configuration de l'instance avec "Azure for Students" et "tpgroup" comme groupe de ressources.

**Screenshot 4:** Nouvelle tâche Stream Analytics. Clique sur le bouton "Créer".

**Screenshot 5:** StreamAnalyticsJob | Vue d'ensemble. Confirmation du déploiement réussi.

- ✓ Configurer les entrées et les sorties :

- L'entrée est **Event Hub**, qui reçoit les flux de données en temps réel :

**Screenshot 1:** SAOpenSky | Entrées. Sélectionné "Entrées".

**Screenshot 2:** Ajouter une entrée. Sélectionné "Hub d'événements".

**Screenshot 3:** Entrée de flux. Liste des entrées disponibles : Hub d'événements (cercle rouge), IoT Hub, Kafka, Stockage blob/ADLS Gen2.

## Hub d'événements

Nouvelle entrée

**Alias de l'entrée \***

Fournir les paramètres de Hub d'événements manuellement  
 Sélectionner Hub d'événements dans vos abonnements

**Abonnement**

**Espace de noms Event Hub \***

**Nom de l'Event Hub \***

 Créer  Utiliser existant

**Groupe de consommateurs Event Hub \***

 Créer  Utiliser existant

**Mode d'authentification**

Le rôle Récepteur de données Azure Event Hubs est accordé à l'identité managée pour cette tâche Stream Analytics lorsque vous cliquez sur Enregistrer. Si l'octroi échoue, suivez les étapes d'octroi manuel [ici](#).

**Clé de partition**

**Format de sérialisation de l'événement \***

**Codage**

**Type de compression d'événement**

**Registre de schémas (préversion)**

Enregistrer

Accueil > StreamAnalyticsJob | Vue d'ensemble > SAOpenSky

**SAOpenSky | Entrées**

+ Ajouter une entrée Actualiser

Alias	Type de source	Type	M..	R..
demo2	Stream	Hub d'événements	Ident	  

**Le test de la connexion a réussi**  
La connexion à l'entrée 'demo2' a été établie.

**Vue d'ensemble**

- Journal d'activité
- Contrôle d'accès (IAM)
- Étiquettes
- Diagnostiquer et résoudre les problèmes
- Topologie de la tâche
- Entrées
- Fonctions
- Requête**
- Sorties
- Éditeur sans code (préversion)

5

8 **Tester la requête sélectionnée** 7 **Enregistrer la requête** X Ignorer les modifications

```

1 /* 8
2 Voici des lignes qui vous aideront à écrire avec le langage de requête de Stream Analytics :
3 Modèles de requête courants - https://go.microsoft.com/fwlink/?linkID=619153
4 Langage de requête - https://docs.microsoft.com/stream-analytics-query/query-language-elements-azure-stream-analytics
5 */
6
7
6
    
```

Aperçu de l'entrée Résultats des tests Simulation de travail (prévision)

Affichage des exemples d'événements de « demo2 ».

Table	Brute	Actualiser	Sélectionner un intervalle de temps	Charger l'échantillon d'entrée	Envoyer des événements	Télécharger
icao24	callsign	origin_country	time_position	last_contact	longitude	
string	string	string	bigint	bigint	float	
“4b1816”	“SWR7DC”	“Switzerland”	1735154769	1735154770	6.6349	
“344445”	“PVG2024”	“Spain”	1735154769	1735154769	1.8325	

While sampling data, no data was received from '1' partitions.

- La sortie est **Blob Storage**, où les résultats sont stockés pour un usage ultérieur :

Nouvelle sortie

Format de sérialisation de l'événement \* ⓘ

JSON

Format ⓘ

Tableau

Codage ⓘ

UTF-8

Mode écriture \*

Une fois, lorsque tous les résultats de la partition de temps sont disponibles. Assure une livraison une seule fois (préversion)

Sélectionner Stockage blob/ADLS Gen2 dans vos abonnements

Abonnement

Azure for Students

Compte de stockage \*

tp2cloud

Conteneur \*

Créer Utiliser existant

mycontainer

Mode d'authentification

Identité managée : Système attribué

Le rôle Contributeur aux données Blob du stockage est accordé à l'identité managée pour cette tâche Stream Analytics lorsque vous cliquez sur Enregistrer. Si l'octroi échoue, suivez les étapes d'octroi manuel [ici](#).

Modèle de chemin d'accès

Format de date

YYYY/MM/DD

Format d'heure

HH

Nombre minimal de lignes

Durée maximale

Heures Minutes Secondes

Enregistrer

Commencer le travail

Ouvrir dans VS Code

Paramètres de diagnostic

Actualiser

Documentation du langage

Entrées (2)

Sorties (1)

Fonctions (0)

Résultats des tests

Simulation de travail (prévision)

Conteneurs de blobs > mycontainer

Méthode d'authentification : Clé d'accès (Bascuer vers le compte d'utilisateur Microsoft Entra)

Rechercher les objets blobs par préfixe (respecte la casse)

Affichage de tous les éléments

Nom Dernière modification Niveau d'accès Type de blob Taille État du bail

input

seed\_input

0\_4e1f9c1314d463e89db489d54c23c21\_1.json

12/25/2024, 9:50:00 PM Élevé (déduit)

Objet blob de ... 27.81 KiB Disponible

Contenu de l'Event Hub

GROUPE DE CONSOMMATEURS

Etat d'Event Hub ACTIVE

Stratégie de nettoyage SUPPRIMER

Nombre de partitions 2

Demandes

Messages

Débit

1 heure 6 heures 12 heures 1 jour 7 jours 30 jours

Incoming Requests (Somme), openskydemo | 194k

Successful Requests (Somme), openskydemo | 1,94k

Outgoing Messages (Somme), openskydemo | 15,6k

Outgoing Bytes (Somme), openskydemo | 4,63Mo

Server Errors. (Somme), openskydemo | 0

Captured Messages. (Somme), openskydemo | 0

Captured Bytes. (Somme), openskydemo | 0

Accéder au blob storage pour contrôler le stockage

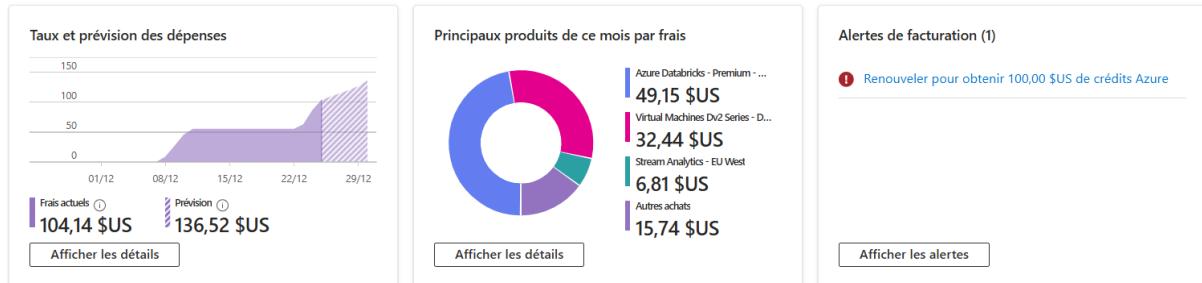
Accéder au EventHub pour contrôler le flux

Et voilà, on a terminé l'étape 1 pour configurer un environnement permettant de capturer, traiter et stocker des données en temps réel en utilisant deux méthodes très puissantes. Il est important de noter qu'on peut remarquer des différences dans les paramétrages par rapport à ce qu'on avait fait lors de la création des ressources dans le travail précédent ou celui qui suivra. Ces écarts sont dus à la terminaison de l'abonnement du compte Azure avec lequel on travaille et l'enchainement avec un autre compte.

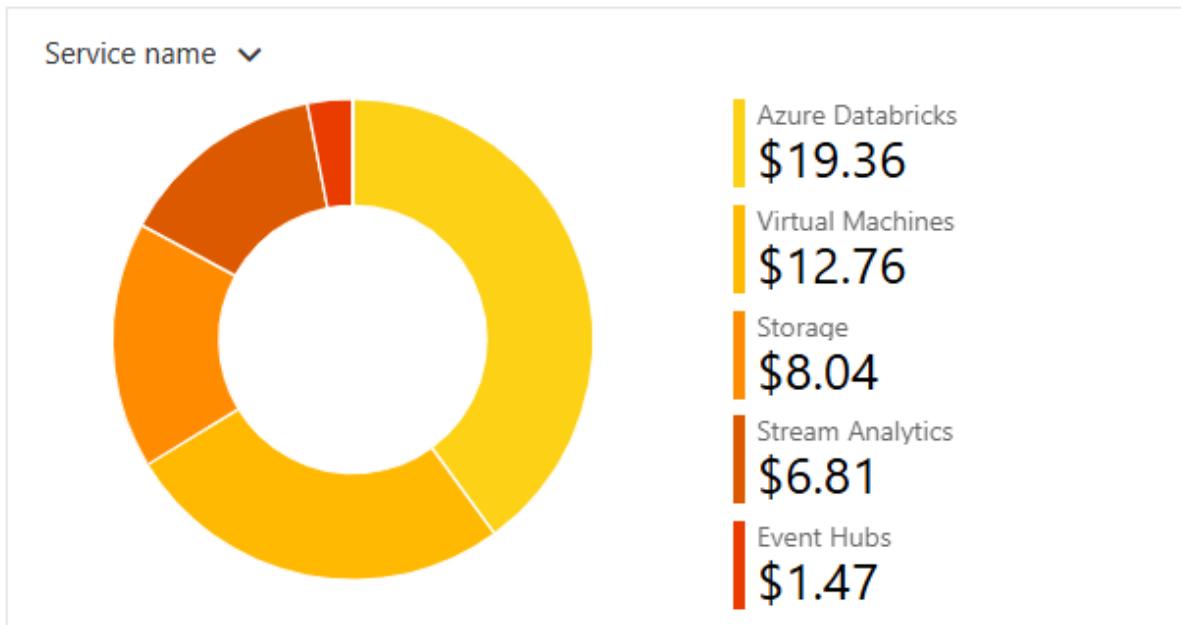
En tout, l'ensemble du travail n'a pas dépassé un budget de 50 \$. On a, bien sûr, perdu beaucoup de temps à gérer des erreurs et à créer plusieurs ressources, mais si on applique

directement ce travail après avoir identifié toutes les erreurs possibles et appris à les éviter (en reproduisant exactement ce qu'on a fait ici), le coût ne dépasserait même pas 25 \$.

Pour donner une idée plus précise, voici une description du budget dépensé dans ce TP (ainsi qu'un projet déjà réalisé avant ce TP). Cela permettra de mieux gérer nos ressources tout en maîtrisant les coûts liés à ce type de projet.



### Voici les dépenses pour ce TP :



### Étape 2 : Optimisation des performances

- Objectif : Identifier les goulets d'étranglement et optimiser le traitement des données.**

#### 1. Configuration de partitions pour un traitement parallèle.

- ⇒ Pour la configuration des partitions pour un traitement parallèle, on peut intervenir directement dans **Event Hubs** lors de la création du **namespace**. C'est là qu'on définit le nombre de **partitions** des données. Lors de cette création, on configure aussi les **unités de débit**, qui détermineront la capacité de consommation des partitions. Ce paramètre est important pour le parallélisme, car plus on a de partitions et d'unités de débit, plus le traitement des données peut être parallélisé.

- ⇒ On peut également définir une **configuration initiale** pour les unités de débit et spécifier le **maximum** qu'on pourrait atteindre, en fonction des besoins de l'application. De plus, Event Hubs offre une option pour l'**augmentation automatique** des unités de débit. Cela permet à la plateforme d'ajuster automatiquement les ressources en fonction du volume de données traité, ce qui est utile pour gérer les pics de charge sans avoir à ajuster manuellement les paramètres.

The screenshot shows the configuration page for a Stream Analytics job. It includes fields for:

- Nombre de partitions**: A slider set to 3.
- Niveau tarifaire \***: Set to "Standard (~\$22 USD par TU par mois)".
- Unités de débit \***: A slider set to 4.
- Activer l'augmentation automatique**: Checked.
- Augmentation automatique des unités de débit maximales**: A slider set to 15.

At the bottom are buttons for "Vérifier + créer" (Review + Create), "< Précédent" (Previous), and "Suivant : Avancé >" (Next : Advanced).

2. **Test de performances sur des flux à grande échelle.**
3. **Ajustement des paramètres de scalabilité automatique (auto-scaling) sur la plateforme cloud.**

- ⇒ Dans cette tâche, on va activer l'**autoscaling** des **unités de streaming** dans **Azure Stream Analytics**. L'autoscaling permet d'ajuster automatiquement le nombre d'unités de streaming en fonction du volume de données à traiter. Cela permet de gérer efficacement les pics de données tout en optimisant les coûts, car Azure ajoute ou réduit dynamiquement les unités de streaming selon les besoins. On va configurer les paramètres nécessaires pour que l'autoscaling s'active en fonction de la charge, garantissant ainsi une performance optimale sans nécessiter d'ajustements manuels constants.

**(!) NB :** L'**unité de streaming** dans **Azure Stream Analytics** correspond à une mesure des ressources de calcul nécessaires pour traiter les données en temps réel. Chaque unité de streaming représente un certain nombre de **ressources de calcul** et de **mémoire** allouées pour exécuter les tâches de traitement des flux de données. Cela inclut les opérations telles que les agrégations, les filtrages, et autres transformations effectuées sur les données en continu.

Accueil > StreamAnalyticsJob | Vue d'ensemble > SAOpenSky

## SAOpenSky | Mettre à l'échelle ⋮

Tâche Stream Analytics

Rechercher
Enregistrer
Abandonner
Actualiser
Journaux
Commentaires

✖ Diagnostiquer et résoudre les problèmes

- ▽ Topologie de la tâche
  - Entrées
  - Fonctions
  - Réquête
  - Sorties
  - Éditeur sans code (préversion)
- ▽ Paramètres
  - Environnement
  - Paramètres du compte de stockage
  - Mettre à l'échelle (sélectionné)
  - Paramètres régionaux
  - Ordre des événements
  - Mise en réseau

**Configurer** Histor. exéc. JSON Notifier Paramètres de diagnostic

La mise à l'échelle automatique permet à votre tâche de modifier dynamiquement le nombre d'unités de streaming (SU) allouées en fonction de la charge, des métriques et/ou de l'évolution de la planification. Définissez des conditions basées sur les métriques de votre tâche pour déterminer le moment où les opérations de mise à l'échelle doivent avoir lieu. Cette méthode est idéale si votre tâche n'a pas de schéma prévisible et que des ajustements dynamiques sont nécessaires pour fonctionner de manière optimale. Si le modèle de votre tâche est prévisible, utilisez la condition de planification pour organiser automatiquement les opérations de mise à l'échelle à votre place. Consultez [Mise à l'échelle automatique des unités de streaming Azure Stream Analytics | Microsoft Docs](#).

**Choisir comment mettre à l'échelle votre ressource**

✖ **Mise à l'échelle manuelle**  
 Ajuster manuellement les unités de streaming fixes de votre travail

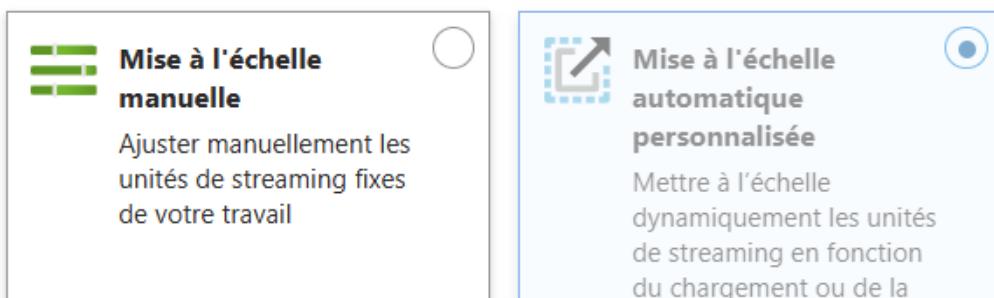
✖ **Mise à l'échelle automatique personnalisée**  
 Mettre à l'échelle dynamiquement les unités de streaming en fonction du chargement ou de la

Mise à l'échelle manuelle

Remplacer la condition

Unité de diffusion en continu

### Choisir comment mettre à l'échelle votre ressource



Après avoir changé le mode à "**Mise à l'échelle automatique personnalisée**", on choisit l'option 'selon une mesure' de mode de mise à l'échelle, puis on définit les limites minimales et maximales des unités de diffusion en continu. Ici, on a fixé un minimum à 1 et un maximum à 3.

Puis, la condition de mise à l'échelle est basée sur la métrique "Input Event Bytes" : si la moyenne des octets entrants dépasse 70 pendant 10 minutes consécutives, une unité de diffusion est ajoutée, avec une période de refroidissement de 5 minutes(une pause de 5 minutes avant d'exécuter une autre action pour éviter des ajustements trop fréquents). Les étapes configurées sont détaillées dans l'interface.

**Par défaut\*** Condition de mise à l'échelle par défaut créée automatiquement  

Supprimer l'avertissement  La toute dernière règle de périodicité ou celle par défaut ne peuvent pas être supprimées. Vous pouvez désactiver l'échelle automatique pour l'arrêter.

Mode de mise à l'échelle  Mettre à l'échelle selon une mesure  Effectuer une mise à l'échelle vers un unité de diffusion en continu spécifique

Règles  [Ajouter une règle](#) pour ajuster les unités de streaming (SUS) en fonction des règles. Par exemple : « Ajouter une règle qui augmente les unités de sécurité à la valeur disponible suivante lorsque le pourcentage d'UC est supérieur à 70 % ». Si vous enregistrez le paramètre sans aucune règle définie, aucune mise à l'échelle ne se produit.

Unité de diffusion en continu Minimum \* Maximum \* Par défaut \*  
1/3 1 1

Planification **Cette condition de mise à l'échelle est exécutée quand aucune des autres conditions de mise à l'échelle ne correspond**

## Règle de mise à l'échelle

Source de la mesure  
Ressource actuelle (SAOpenSky)

Type de ressource  Ressource   
Travaux Stream Analytics SAOpenSky

Critères

Espace de noms de métrique *	Nom de la métrique
Métriques standard	Input Event Bytes

Fragments de temps de 1 minute

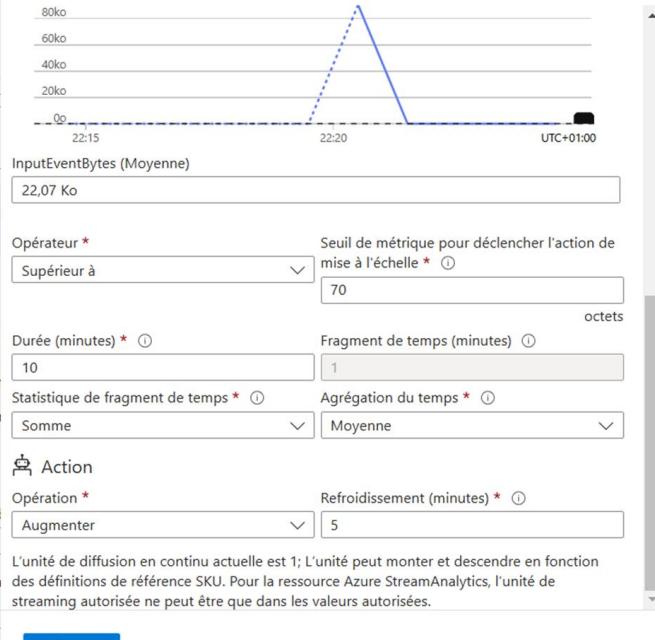
Nom de dimension	Opérateur	Valeurs de la dimension	Aj...
Logical Name	=	Toutes les valeurs	+ 
Logical Name Instance	=	Toutes les valeurs	+ 
Node Name	=	Toutes les valeurs	+ 
Partition ID	=	Toutes les valeurs	+ 

Si vous sélectionnez plusieurs valeurs pour une dimension, la mise à l'échelle automatique agrège la métrique sur les valeurs sélectionnées au lieu d'évaluer la métrique pour chaque valeur séparément.

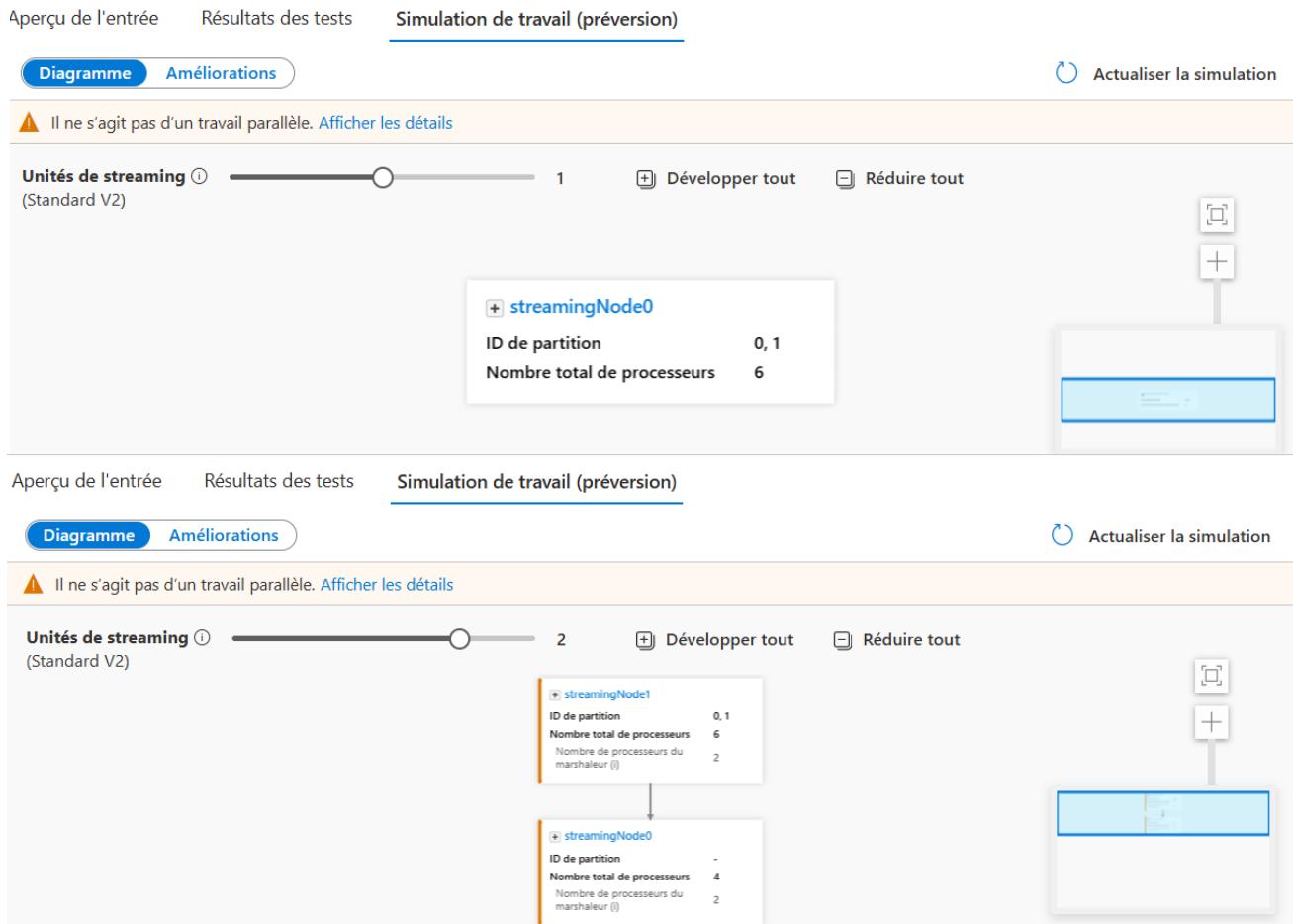


**Ajouter**

## Règle de mise à l'échelle



**( ! ) NB :** On peut modifier les unités de streaming manuellement dans l'interface de 'Requête' dans notre tâche Stream Analytics via :



### Étape 3 : Sécurisation des données

- **Objectif : Appliquer les bonnes pratiques de sécurité dans un environnement cloud.**
  1. **Configuration des règles IAM (Identity and Access Management) pour limiter l'accès aux ressources.**
    - ⇒ Par exemple on peut donner un accès de lecture seulement des evenement à un autre utilisateur , via :

Accueil > ProjectEventHubs | Contrôle d'accès (IAM) ...

**ProjectEventHubs | Contrôle d'accès (IAM)**

Rechercher Ajouter Télécharger les attributions de rôles Modifier les attributions de rôles

Vue d'ensemble Journal d'activité Contrôle d'accès (IAM) 1 Étiquettes Diagnostiquer et résoudre les problèmes Data Explorer Événements Paramètres Stratégies d'accès partagé Mettre à l'échelle Géo-récupération Réseau Chiffrement Identité Configuration

Vérifier l'accès Attributions de rôles Rôles Affections de rôles

Mon accès Afficher mon niveau d'accès à cette ressource. Vérifier mon accès

Vérifier l'accès Passez en revue le niveau d'accès d'un utilisateur, d'un groupe, d'un principal Vérifier l'accès

Accorder l'accès à cette ressource Accordez l'accès aux ressources en attribuant un rôle. En savoir plus 2 Ajouter une attribution de rôle

Afficher l'accès Affichez les attributs autorisant l'accès à d'autres ressources En savoir plus Afficher

Rôle Membres Conditions Vérifier + attribuer

Une définition de rôle est un ensemble d'autorisations. Vous pouvez utiliser les rôles intégrés ou vous pouvez créer vos propres rôles personnels.

Rôles de fonction de travail Rôles d'administrateur privilégié

Accordez l'accès aux ressources Azure en fonction de la fonction de travail, comme la possibilité de créer des machines virtuelles.

Rechercher par nom de rôle, description, autorisation... Type : Tout Catégorie : Tout

Nom ↑ Description ↑

Lecteur Affiche toutes les ressources, mais ne vous autorise pas à apporter des modifications App Compliance Automation Administrator Create, read, download, modify and delete reports objects and related other resource

Accueil > ProjectDatarciks | Contrôle d'accès (IAM) > Ajouter une attribution de rôle ...

Rôle Membres 4 Conditions Vérifier + attribuer

Role sélectionné Lecteur

Attribuer l'accès à Utilisateur, groupe ou principal de service 5 Identité managée

Membres + Sélectionner des membres 6

Description Facultatif

Vérifier + attribuer Précédent Suivant

Sélectionner des membres

Hamdi.Belanezi@isima.u... MINISTÈRE DE L'ENSEIGNEMENT...

wafa.jbali wafa.jbali@isima.u-monastir.tn 7

Membres sélectionnés :

wafa.jbali wafa.jbali@isima.u-monastir.tn

Sélectionner Fermer

Accueil > ProjectDatarciks | Contrôle d'accès (IAM) > Ajouter une attribution de rôle ...

Rôle Membres 8 Conditions Vérifier + attribuer

Rôle sélectionné Lecteur

Attribuer l'accès à Utilisateur, groupe ou principal de service Identité managée

Membres + Sélectionner des membres

Nom	ID d'objet	Type
wafa.jbali	6fafc547c	Utilisateur

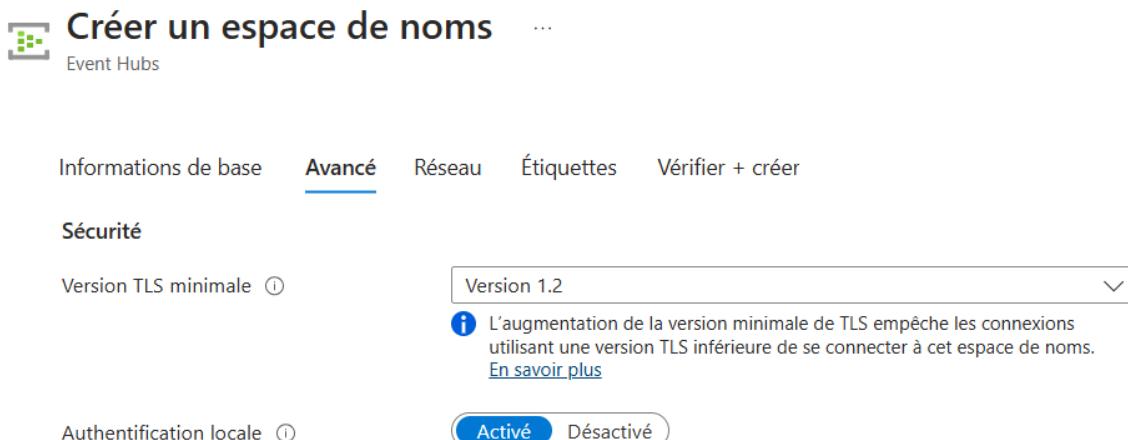
Vérifier + attribuer Précédent Suivant

## 2. Mise en place d'un chiffrement des données au repos et en transit.

⇒ **Azure Event Hubs** fournit par défaut le chiffrement des données **en transit** et **au repos** pour garantir une sécurité renforcée. Voici comment cela fonctionne :

### ✓ Chiffrement des données en transit

- **Par défaut** : Toutes les communications entre les producteurs, Event Hubs, et les consommateurs utilisent **TLS (Transport Layer Security)**, qui chiffre les données en transit.
- On a la possibilité de renforcer cela en configurant la **version minimale de TLS** (par exemple, TLS 1.2 ou supérieur), comme dans la configuration ci-dessous lors de création du eventHub Namespace



The screenshot shows the Azure portal interface for creating a new Event Hub namespace. The top navigation bar includes 'Créer un espace de noms', 'Event Hubs', and a '...' button. Below the navigation, there are tabs: 'Informations de base', 'Avancé' (which is underlined, indicating it's the active tab), 'Réseau', 'Étiquettes', and 'Vérifier + créer'. The 'Avancé' tab is currently selected. In the 'Sécurité' section, there are two settings: 'Version TLS minimale' (set to 'Version 1.2') and 'Authentification locale' (set to 'Activé'). A tooltip for the TLS version setting provides information about preventing lower TLS versions from connecting.

(!) **NB** : On peut même chiffrer la clé d'accès à EventHub comme faite dans le script du consommateur (Spark Streaming) vu précédemment.

### ✓ Chiffrement des données au repos

- Azure chiffre automatiquement les données au repos dans Event Hubs à l'aide de **Microsoft-managed keys** (clés gérées par Microsoft) dans le cadre du **chiffrement côté serveur (Server-Side Encryption)**.

- Les messages, logs, et toutes les données stockées dans Event Hubs sont protégés par défaut. (Malheureusement cette option n'est possible que pour l'abonnement Premium)

The screenshot shows the Azure portal interface for managing an Event Hub. The top navigation bar includes 'Accès' and 'ProjectEventHubs'. The main title is 'ProjectEventHubs | Chiffrement' with a subtitle 'Espace de noms Event Hubs'. Below the title are buttons for 'Rechercher', 'Enregistrer', and 'Abandonner'. A sidebar on the left lists several options: Événements, Paramètres (selected), Stratégies d'accès partagé, Mettre à l'échelle, Géo-récupération, Réseau, Chiffrement (selected), and Identité. The main content area discusses client-managed encryption (Bring Your Own Key) and notes that it is only available on Premium namespaces. It also mentions that Microsoft-managed keys are used for encrypting data in the Event Hub namespace. A note at the bottom indicates that client-managed encryption can only be activated on empty namespaces.

### 3. Définition de règles de journalisation pour suivre les accès et activités.

⇒ Pour assurer un suivi efficace des accès et des activités dans un EventHub par exemple, on commence par accéder à l'espace de noms EventHub concerné. Dans le menu, il suffit de naviguer vers l'onglet Supervision puis de sélectionner Plans de diagnostic. À cet endroit, on peut configurer des règles de journalisation pour capturer différents types de logs, comme les connexions entrantes, les erreurs, ou les métriques liées au traitement des messages. Par exemple, on peut activer la journalisation des logs opérationnels pour analyser les requêtes effectuées sur l'Event Hub et s'assurer d'un suivi détaillé des accès. Ces journaux peuvent ensuite être envoyés vers une destination comme Log Analytics, Azure Storage, ou un Event Hub secondaire, permettant ainsi une analyse centralisée et sécurisée des données collectées.

Accéder au EventHub Namespace  
Dans le menu > Supervision

Paramètres de diagnostic

Actualiser Commentaires

Les paramètres de diagnostic sont utilisés pour configurer l'exportation en streaming des métriques et/ou paramètres de diagnostic pour envoyer différents journaux et métriques à des destinations indépendantes

Paramètres de diagnostic

Nom	Compte de stockage	Hub d'événements
Aucun paramètre de diagnostic défini		

+ Ajouter un paramètre de diagnostic

Paramètre de diagnostic

Enregistrer Abandonner Supprimer Commentaires

Un paramètre de diagnostic spécifie une liste de catégories de métriques et/ou journaux de plateforme à collecter sur une ressource, et une ou plusieurs destinations auxquelles les envoyer en streaming. Des frais d'utilisation normaux vous sont facturés pour la destination. [En savoir plus sur les différentes catégories de journal et le contenu de ces journaux](#)

Nom du paramètre de diagnostic \*

Journaux

Groupes de catégories

- allLogs
- audit

Catégories

- Diagnostic Error Logs
- Archive Logs
- Operational Logs
- Auto Scale Logs
- Kafka Coordinator Logs
- Kafka Topic Error Logs

Détails de la destination

- Envoyer à l'espace de travail Log Analytics
- Archiver dans un compte de stockage
- Diffuser vers Event Hub
- Envoyer à la solution partenaire

## Étape 4 : Intégration et visualisation des données

- **Objectif : Exploiter les données issues du pipeline pour des analyses en temps réel.**
  1. **Connecter une plateforme de visualisation (Power BI, Tableau, Grafana) aux données en temps réel.**

⇒ On va utiliser PowerBI pour la visualisation, on peut le connecter en suivant ces étapes :

Accéder au website PowerBI et s'authentifier en utilisant les mêmes données d'accès du Compte Microsoft

Créer un espace de travail

Nom \* workspaceOpenSky

Description Décrire cet espace de travail

Domaine Affecter à un domaine (facultatif)

Image d'espace de travail Charger Réinitialiser

Options avancées

Liste de contacts \* Hamdi.Belanez (Propriétaire)

Modèle de licence

- Pro Select Pro to use basic Power BI features and collaborate on reports, dashboards, and scorecards. To access a Pro workspace, users need Pro per-user licenses. [En savoir plus](#)
- Essai Select the free trial per-user license to try all the new features and experiences in Microsoft Fabric for 60 days. A Microsoft Fabric trial license allows users to create Microsoft Fabric items and collaborate with others in a Microsoft Fabric trial capacity. Explore new capabilities in Power BI, Data Factory, Data Engineering, and Real-Time Intelligence, among others. [En savoir plus](#)
- Premium par utilisateur Select Premium per-user to collaborate using Power BI Premium features, including paginated reports, dataflows, and datamarts. To collaborate and share content in a Premium per-user workspace, users need Premium per-user licenses. [En savoir plus](#)
- Capacité Premium

Applications modèles

Les applications modèles sont développées pour être partagées en dehors de l'organisation. Un espace de travail d'application modèle est créé pour développer et publier l'application. [En savoir plus sur les applications modèles](#)

Développer des applications modèles

Appliquer Annuler

inputstream

Sorties (1)

output

Fonctions (0)

+

Base de données PostgreSQL

Cosmos DB

Data Lake Storage Gen1

File d'attente Service Bus

Fonction Azure

Hub d'événements

Kafka

Power BI

Rubrique Service Bus

SQL Database

le nom de workspace crée dans PowerBI(va être automatiquement listé dans les choix)

le nom du dataset que va être affiché dans le workspace PowerBI dès qu'on lance le travail

et le nom du table dans ce dataset

Power BI

Nouvelle sortie

⚠ Stream Analytics Power BI output is deprecating. Existing jobs will continue to run till Oct 31 2027. To learn more about the Power BI output deprecation click here.

Alias de sortie \* FlightInfosDashboard

Fournir les paramètres de Power BI manuellement

Sélectionner Power BI dans vos abonnements

Espace de travail de groupe \* workspaceOpenSky

Mode d'authentification Identité managée : Système attribué

Le rôle Collaborateur est accordé à l'identité managée pour cette tâche Stream Analytics lorsque vous cliquez sur Enregistrer. Si l'octroi échoue, suivez les étapes d'octroi manuel [ici](#).

Nom du jeu de données \* FlightsDataset

Nom de la table \* FlightsTable

Enregistrer

⇒ Dans ce job Stream Analytics, on a traité les données en sélectionnant les colonnes principales, ajouté deux nouvelles colonnes calculées (aircompany et flightstatus), et

filtré les lignes avec des valeurs nulles dans les champs critiques (icao24, callsign, origin\_country). Contrairement aux agrégations précédentes.

```
SELECT
    icao24,
    callsign,
    origin_country,
    time_position,
    last_contact,
    longitude,
    latitude,
    baro_altitude,
    geo_altitude,
    velocity,
    true_track,
    vertical_rate,
    -- Nouvelle colonne : aircompany (les 3 premiers caractères de callsign)
    LEFT(callsign, 3) AS aircompany,
    -- Nouvelle colonne : flightstatus (basé sur la colonne on_ground)
    CASE
        WHEN on_ground = TRUE THEN 'on ground'
        ELSE 'in air'
    END AS flightstatus
INTO
    [FlightInfosDashboard]
FROM
    [OpenSkyProducer] TIMESTAMP BY time
-- Eliminer les valeurs nulles dans les colonnes critiques
WHERE icao24 IS NOT NULL
    AND callsign IS NOT NULL
    AND origin_country IS NOT NULL
```

**Test de la requête**

```

1 SELECT
2   icao24,
3   callsign,
4   origin_country,
5   time_position,
6   last_contact,
7   longitude,
8   latitude,
9   baro_altitude,
10  geo_altitude

```

Aperçu de l'entrée Résultats des tests Simulation de travail (prévision)

Affichage des exemples d'événements de « OpenSkyProducer ».

Table	Brute	Actualiser	Sélectionner un intervalle de temps	Charger l'échantillon d'entrée	Envoyer des événements	Télécharger
icao24 string	callsign string	origin_country string	time_position bigint	last_contact bigint	longitude float	
"511172"	"MBU2AT"	"Estonia"	1735260928	1735260928	0.9406	
"407794"	"EZY41DG"	"United Kingdom"	1735260928	1735260928	6.1911	
"49d418"	"TVS54147"	"Czech Republic"	1735260929	1735260929	2.0844	

While sampling data, no data was received from '1' partitions.

**Test de la requête**

```

1 SELECT
2   icao24,
3   callsign,
4   origin_country,
5   time_position,
6   last_contact,
7   longitude,
8   latitude,
9   baro_altitude,
10  geo_altitude

```

Aperçu de l'entrée Résultats des tests Simulation de travail (prévision)

Télécharger les résultats

Table	Brute	Actualiser	Sélectionner un intervalle de temps	Charger l'échantillon d'entrée	Envoyer des événements	Télécharger
icao24 string	callsign string	origin_country string	time_position bigint	last_contact bigint	longitude float	
"511172"	"MBU2AT"	"Estonia"	1735260928	1735260928	0.9406	
"407794"	"EZY41DG"	"United Kingdom"	1735260928	1735260928	6.1911	
"49d418"	"TVS54147"	"Czech Republic"	1735260929	1735260929	2.0844	
"a7ef06"	"N610CX"	"United States"	1735260928	1735260928	7.6698	

Affichage de 0 lignes à partir de « FlightInfosDashboard ».

## Démarrer le travail

SAOpenSky

Vous n'avez pas encore configuré les paramètres de diagnostic pour ce travail.  
[Ajoutez des paramètres de diagnostic dans le volet Paramètres de diagnostic.](#)

Unités de streaming

2

Environnement

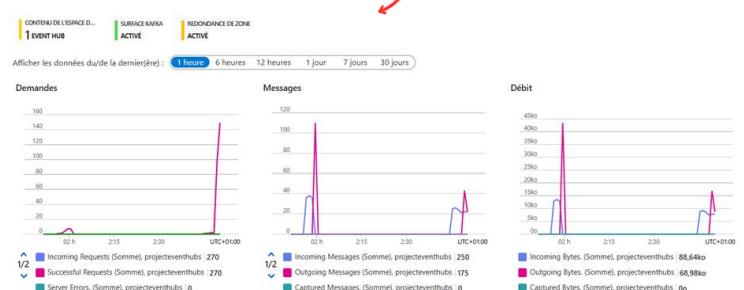
Standard

Heure de début de la sortie de la tâche

Maintenant

Personnalisé

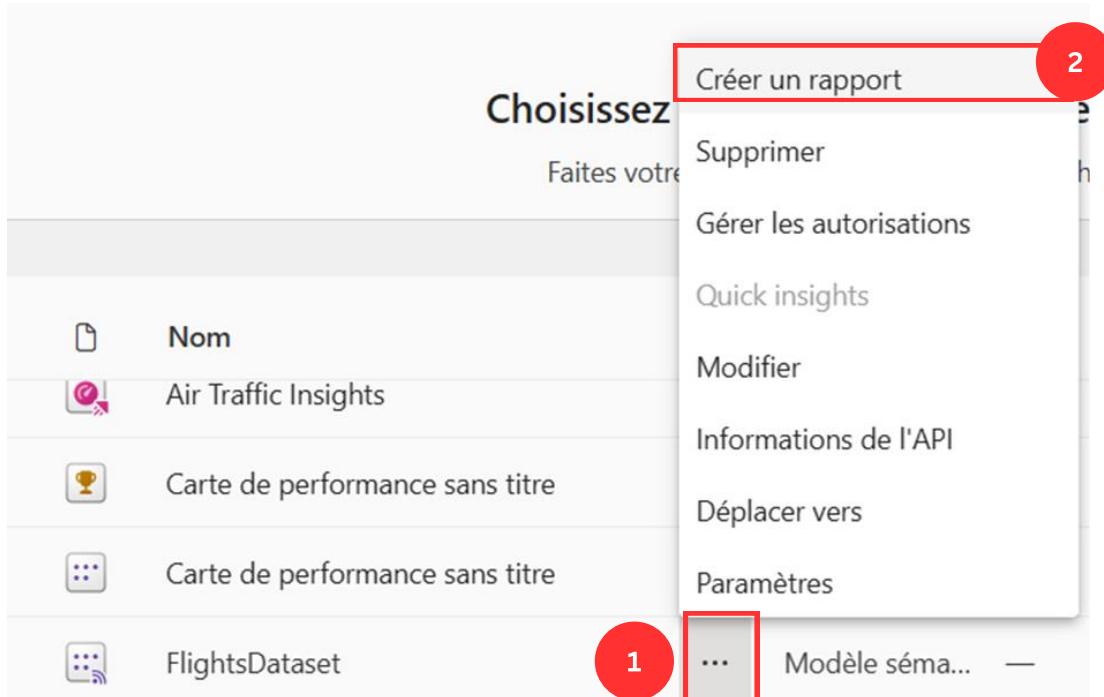
Accéder au EventHub pour contrôler le flux



Démarrer

Accéder au workspace créé dans PowerBI pour contrôler les données

## 2. Créer des graphiques pour surveiller les performances ou détecter des anomalies.



- ⇒ Dans ce rapport Power BI créé intitulé "**AIR TRAFFIC INSIGHTS**", on a créé plusieurs visualisations pour faciliter la surveillance des performances et la détection de tendances importantes dans les données de trafic aérien.
- **Total Flights et Air Companies** : Deux cartes de KPI affichent le nombre total de vols et le nombre de compagnies aériennes distinctes , ce qui permet de suivre rapidement l'activité globale.
- **Mean, Min, and Max Values of Velocity** : Une jauge affiche la valeur moyenne actuelle de la vitesse avec une plage allant de minimum au maximum, pour visualiser les vitesses enregistrées.
- **Number of Flights By Origin Country** : Un graphique en colonnes représente le nombre de vols par pays d'origine, ce qui permet d'identifier les principaux contributeurs en termes de trafic aérien.
- **Latitude et Longitude** : Une carte géographique localise les vols en fonction des coordonnées de latitude et longitude, offrant une vue visuelle claire de leur répartition géographique.
- **Temps** : Un curseur qui permet non seulement de visualiser les données en temps réel, mais aussi de contrôler les anciennes versions des données tout en affichant les nouvelles en temps réel.

AirTrafficInsights | Données mises à jour le 27/12/24

Fichier Affichage Mode Lecture Disposition pour mobile Ouvrir le modèle de données

Rechercher Copilot Essai : 56 jour Responsable de compte pour Hamdi Belanez

**AIR TRAFFIC INSIGHTS**

Total Flights: 77 Air Companies: 36 Mean,Min and Max values of Velocity: 146.88 Time: 274,89

Number of Flights By origin\_country:

origin_country	Nombre de origin_country
Germany	180
Austria	160
United Kingdom	140
Denmark	120
Spain	100
United States	90
Belgium	80
Switzerland	70
Ireland	60
Egypt	50
France	40
San Marino	30
Canada	20
Tunisia	15
Bulgaria	10
Czech Republic	5
United Arab Emirates	2

latitude and longitude map of France and surrounding regions.

Visualisations: Générer un élément visuel. Données: FlightsTable. Filters: Latitude (checked), Longitude (checked). Other filters include: aircompany, baro\_altitude, callsign, flightstatus, geo\_altitude, icao24, last\_contact, latitude, longitude, origin\_country, time\_position, true\_track, velocity, vertical\_rate.

- Une version selon les données actuelle :

## AIR TRAFFIC INSIGHTS



- Une version selon les données anciennes :

# AIR TRAFFIC INSIGHTS

Total Flights

44

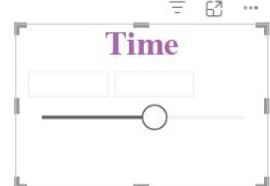
Air Companies

25

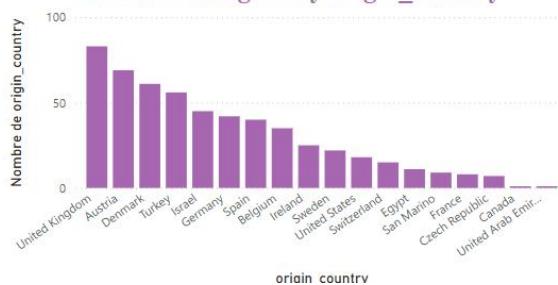
Mean,Min and Max values of Velocity



Time



Number of Flights By origin\_country



latitude and longitude



Et voilà, avec cela, nous avons terminé ce TP en utilisant plusieurs outils Azure et en dépassant 127,71 \$ entre les deux comptes (nous avons dû recréer toutes les ressources sur un nouveau compte pour finaliser certaines étapes). Cependant, sans les erreurs rencontrées et les répétitions, les dépenses auraient pu être réduites à un maximum de 50 \$.

**(!) NB :** il ne faut pas laisser le cluster Databricks actif trop longtemps, car c'est la ressource la plus coûteuse. Contrairement à Event Hub, Blob Storage ou Stream Analytics, Databricks génère des coûts élevés si les clusters restent actifs inutilement. Nous aurions dû les désactiver dès la fin des exécutions et les réactiver uniquement en cas de besoin.

Enfin, pour clôturer ce TP, n'oublions pas d'arrêter également notre travail Stream Analytics, car sinon, ce service, tout comme Databricks, pourrait aussi générer des coûts importants. :D

Arrêter le travail Ouvrir dans VS Code Paramètres de diagnostic Actualiser Document

## Arrêter le travail Stream Analytics

Voulez-vous vraiment arrêter le travail « SAOpenSky » ?