

YNOV Airlines

Flight tracker OpenSky

ABID Sabri
RAMDANI Rayan

Master 1 Data engineer & Data science
Ynov Paris 2021/2022

Sommaire

- ▶ Introduction OpenSky API
- ▶ Objectif du projet
- ▶ Technologies utilisées
- ▶ Notre architecture
- ▶ La démonstration
- ▶ Difficultés rencontrées
- ▶ Pistes explorées
- ▶ Conclusion

Introduction OpenSky API

- ❖ L'API nous permet de récupérer des informations sur l'espace aérien en direct à des fins de recherche et à des fins non commerciales.
- ❖ L'API ne fournit pas de données de vol commerciales telles que les horaires d'aéroport, les retards ou des informations similaires.
- ❖ Les informations sont généralement fournies sous la forme de *vecteurs d'état*.
- ❖ L'état d'un avion est un résumé de toutes les informations de suivi (principalement la position, la vitesse et l'identité) à un moment donné.
- ❖ Ces états d'avions peuvent être récupérés sous forme de vecteurs d'état sous la forme d'un objet JSON.

Introduction OpenSky API

- ❖ OpenSky propose plusieurs API : API REST, API JAVA et l'API PYTHON
- ❖ Pour notre projet nous avons opté pour l'API Python
- ❖ L'API REST était aussi intéressante car elle nous permettait de récupérer les données via une URL

Index	Property	Type	Description
0	<i>icao24</i>	string	Unique ICAO 24-bit address of the transponder in hex string representation.
1	<i>callsign</i>	string	Callsign of the vehicle (8 chars). Can be null if no callsign has been received.
2	<i>origin_country</i>	string	Country name inferred from the ICAO 24-bit address.
3	<i>time_position</i>	int	Unix timestamp (seconds) for the last position update. Can be null if no position report was received by OpenSky within the past 15s.
4	<i>last_contact</i>	int	Unix timestamp (seconds) for the last update in general. This field is updated for any new, valid message received from the transponder.
5	<i>longitude</i>	float	WGS-84 longitude in decimal degrees. Can be null.
6	<i>latitude</i>	float	WGS-84 latitude in decimal degrees. Can be null.
7	<i>baro_altitude</i>	float	Barometric altitude in meters. Can be null.
8	<i>on_ground</i>	boolean	Boolean value which indicates if the position was retrieved from a surface position report.
9	<i>velocity</i>	float	Velocity over ground in m/s. Can be null.
10	<i>true_track</i>	float	True track in decimal degrees clockwise from north (north=0°). Can be null.
11	<i>vertical_rate</i>	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.

- ❖ Voici certaines données qui sont disponible via l'API d'OpenSky, pour ce projet nous avons décidé de nous concentrer sur les données suivante : id, time, longitude ,latitude, vitesse, pays

Objectif du projet

- ❖ Le projet se divise en deux parties la première est de pouvoir collecter et effectuer des calculs grâce à Spark et Kafka. La deuxième partie consiste à utiliser ces données pour pouvoir les visualiser grâce à Streamlit et Folium.

Technologies utilisées

❖ Pour ce projet nous avons opté pour les technologies suivantes :

- 1) Docker.
- 2) Kafka.
- 3) Spark.
- 4) Streamlit

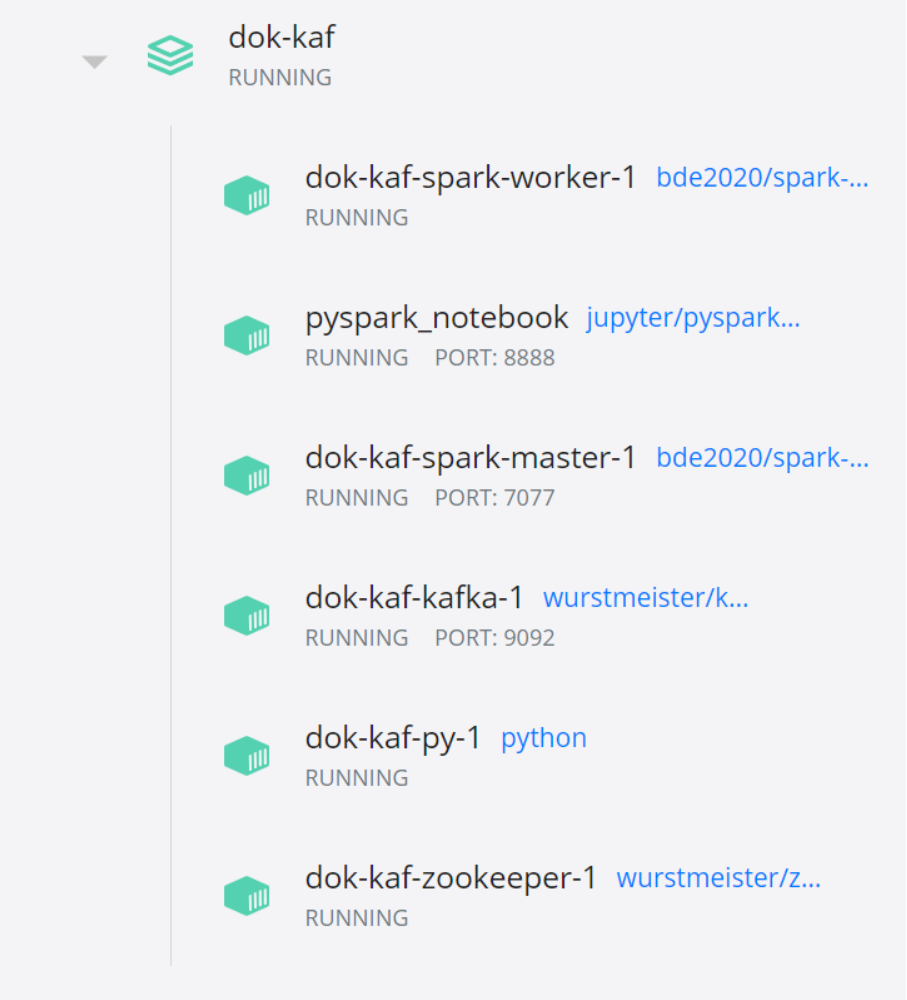
Technologies utilisées

❖ DOCKER :

- Docker est une plateforme permettant de lancer certaines applications dans des conteneurs logiciels.
- Docker est un outil qui peut emballer une application et ses dépendances dans un conteneur isolé, qui pourra être exécuté sur n'importe quel serveur.
- Un container s'agit d'un ensemble de processus logiciels léger et indépendant, regroupant tous les fichiers nécessaires à l'exécution des processus.

Technologies utilisées

❖ DOCKER :



The screenshot displays the Docker Desktop interface. At the top, a container named 'dok-kaf' is shown in a 'RUNNING' state. Below it, a list of six containers is displayed, each with a Docker icon, a name, and a status. The containers are: 'dok-kaf-spark-worker-1' (RUNNING), 'pyspark_notebook' (RUNNING, PORT: 8888), 'dok-kaf-spark-master-1' (RUNNING, PORT: 7077), 'dok-kaf-kafka-1' (RUNNING, PORT: 9092), 'dok-kaf-py-1' (RUNNING), and 'dok-kaf-zookeeper-1' (RUNNING). The interface is clean and modern, with a light gray background and blue accents.

Container Name	Status	Image	Port
dok-kaf	RUNNING		
dok-kaf-spark-worker-1	RUNNING	bde2020/spark-...	
pyspark_notebook	RUNNING	jupyter/pyspark...	8888
dok-kaf-spark-master-1	RUNNING	bde2020/spark-...	7077
dok-kaf-kafka-1	RUNNING	wurstmeister/k...	9092
dok-kaf-py-1	RUNNING	python	
dok-kaf-zookeeper-1	RUNNING	wurstmeister/z...	

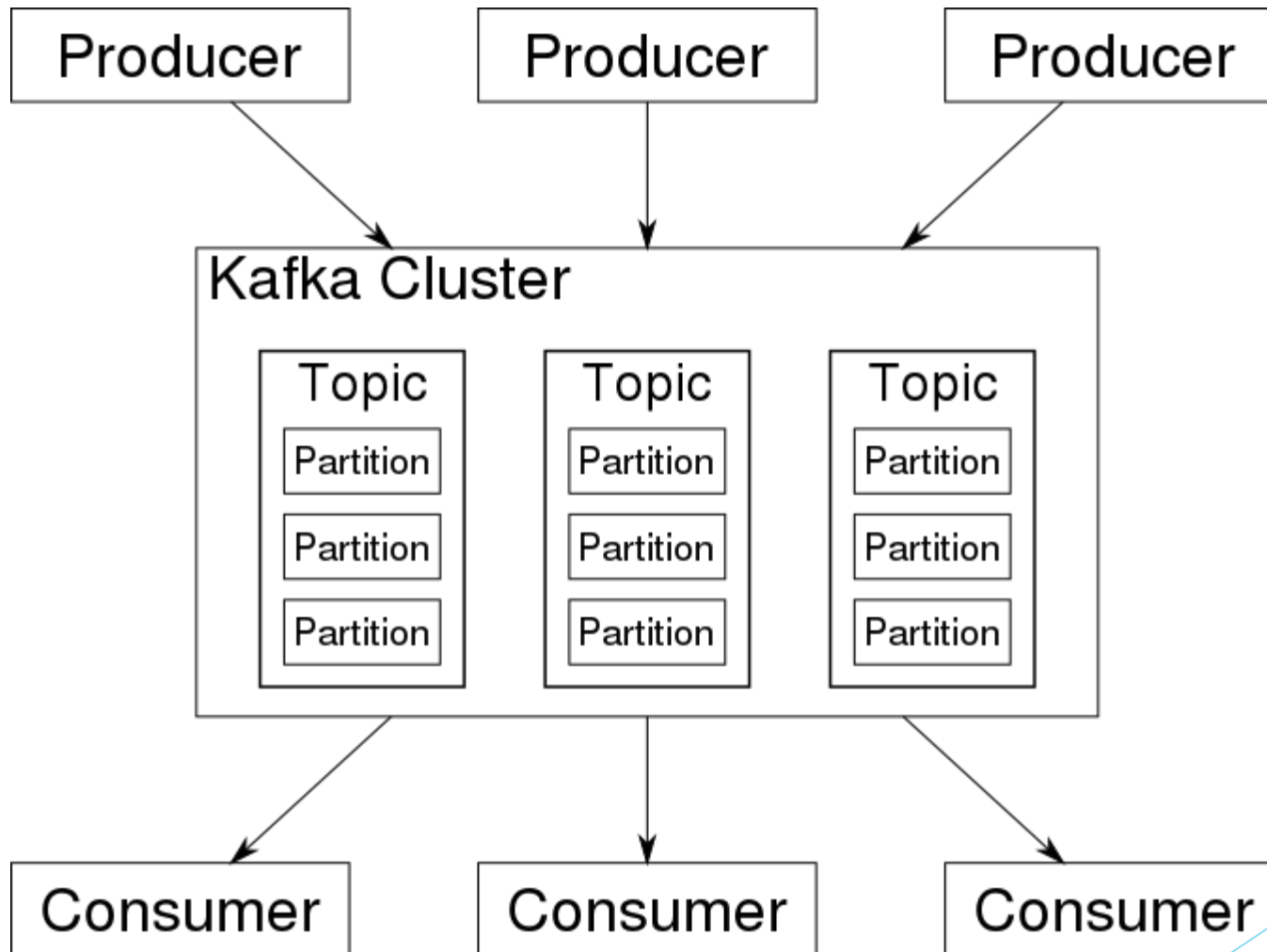
Technologies utilisées

❖ Kafka :

- Apache Kafka est une plateforme distribuée de diffusion de données en continu, capable de publier, stocker, traiter et souscrire à des flux d'enregistrement en temps réel.
- Kafka est conçue pour gérer des flux de données provenant de plusieurs sources et les fournir à plusieurs utilisateurs.

Technologies utilisées

❖ Kafka :



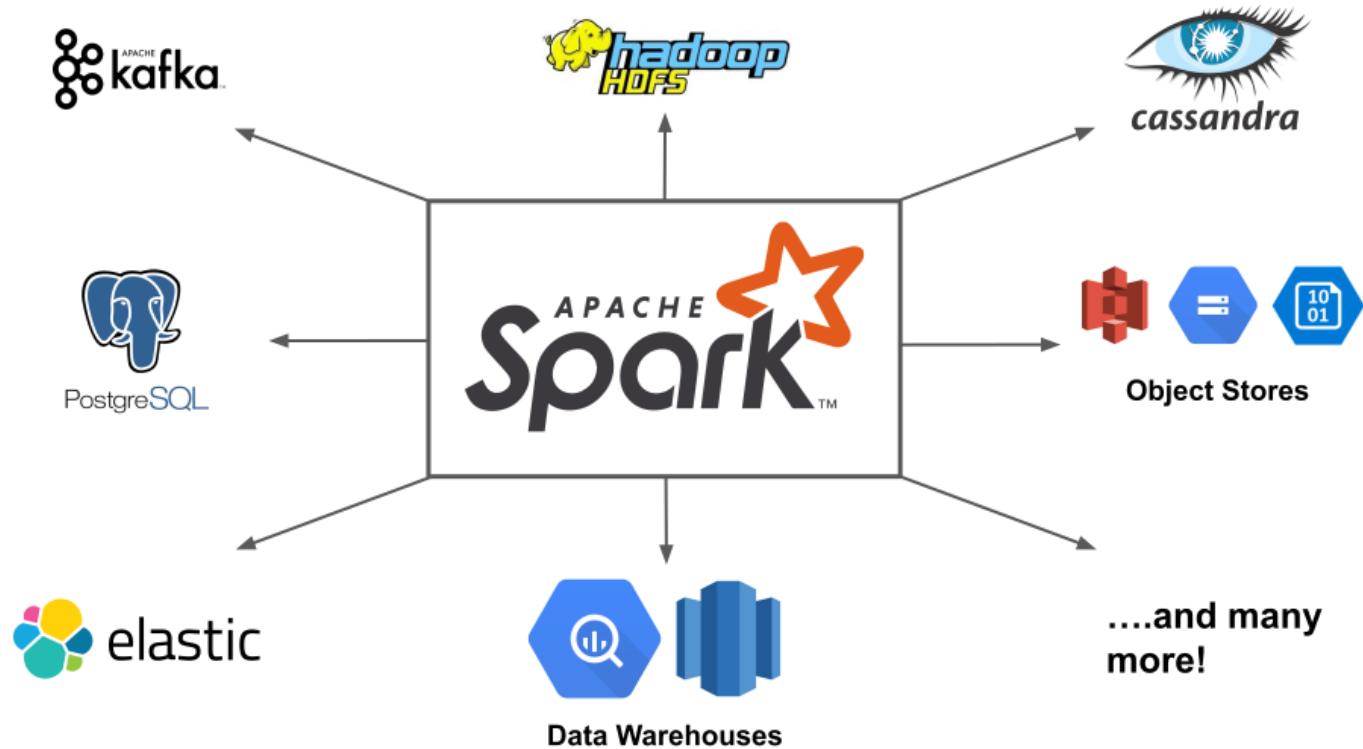
Technologies utilisées

❖ Spark:

- Spark est un Framework open source de calcul distribué. Il s'agit d'un ensemble d'outils et de composants logiciels structurés selon une architecture définie
- Spark réalise une lecture et écriture des données au niveau du cluster (grappe de serveurs sur un réseau).
- Spark peut travailler sur la totalité des données en même temps.
- Spark SQL permet d'exécuter des requêtes en langages **SQL** pour charger et transformer des données.
- Dans Spark, le langage SQL peut être utilisé pour traiter n'importe quelles données, quel que soit leur format d'origine.

Technologies utilisées

❖ Spark:



Technologies utilisées

❖ Streamlit:

Streamlit transforme les scripts de données en applications Web partageables en quelques minutes. Le tout en Python.

Technologies utilisées

Streamlit:

```
# Pour récupérer les données des avions dont les pays sont à l'intérieur
states = api.get_states(bbox=(35.332031,71.142435,-14.238281,34.648304))
l=[]

# On crée une liste qui va contenir toutes les informations dont nous a
for s in states.states:
    if s.callsign != (" ") and s.callsign != (''):
        if s.latitude != None and s.longitude != None:
            if s.velocity != None and s.velocity != 0.0000:
                l=l+[{"id":s.callsign,"time":states.time,"longitude":

# Transformation de la liste en DataFrame
df = pd.DataFrame(l)
st.write(df)

m = folium.Map(location=[20,0])

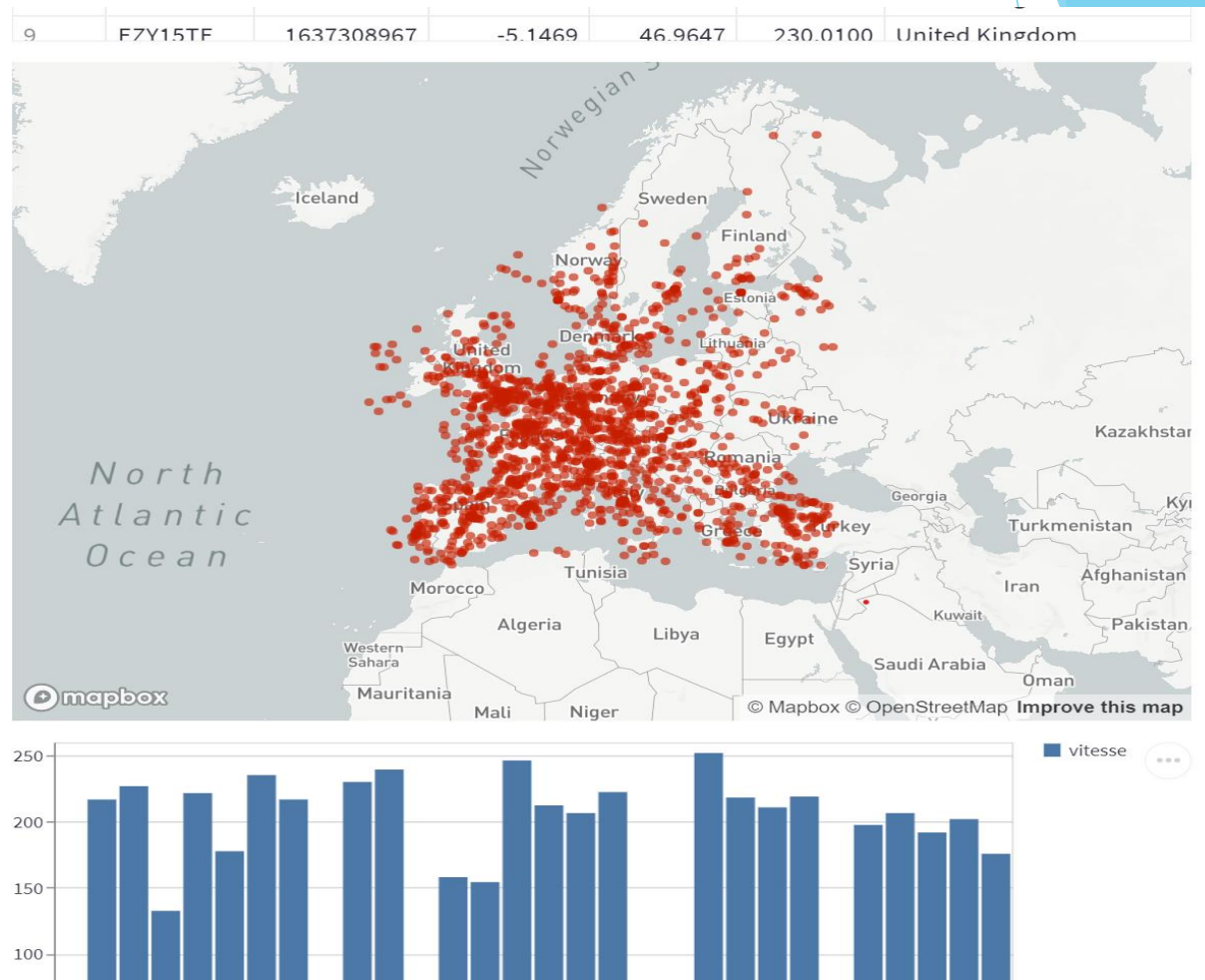
# Ici on affiche tous les avions en vol en fonction de leur latitude et
for i in range(0,len(l)):
    folium.Marker(location=[l[i]["latitude"], l[i]["longitude"]],popup=

l1 = [" "]
for i in range(0,30):
    l1.append(l[i]['vitesse'])

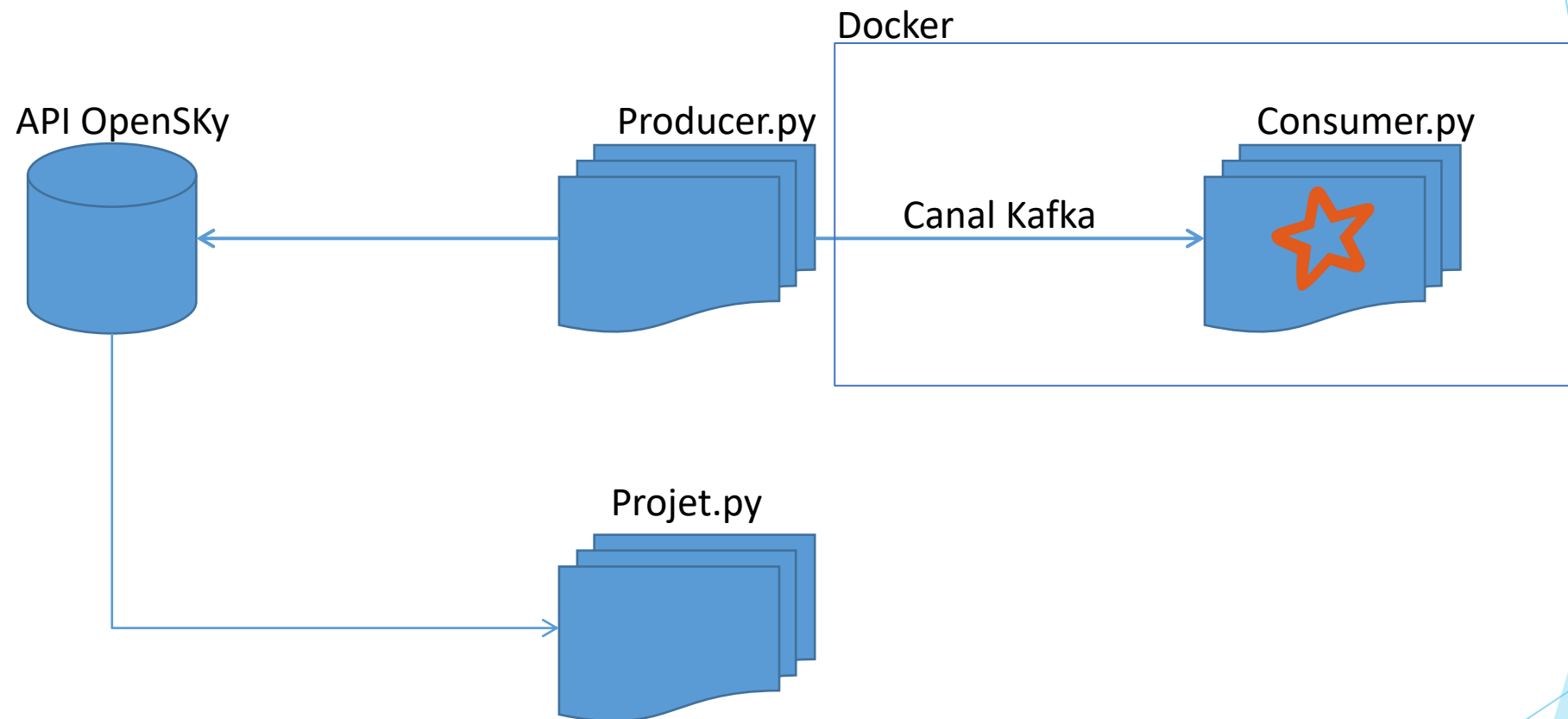
del l1[0]
for i in range(0,30):
    int(l1[i])
print(l1)

chart_data = pd.DataFrame(l1, columns=['vitesse'])

# Affichage de la map
st.map(df)
st.bar_chart(chart_data)
# Ce bouton sert à rafraichir la map
st.button("rerun")
```



Notre architecture :



Démonstration

Difficultés rencontrés

- ❖ Trouver une API de qualité gratuite
- ❖ Des difficultés sur les types de nos données (erreur json serializable)
- ❖ Récolte des données entre API REST et Python
- ❖ Connections entre Kafka et Spark pour le traitement en streaming

Les pistes potentielles

- ▶ Nous pouvons apporter d'autres calculs ou d'autres statistiques (données altitude par exemple)
- ▶ Faire des corrélations (altitude vitesses...)
- ▶ Tracer les avions sur la map



Conclusion

- ▶ Ce projet nous a permis de découvrir beaucoup de nouveaux outils pour nous aider à valoriser des données de manière rapide et continue, nous avons aussi appris à visualiser ses données et à effectuer des calculs dessus.
- ▶ De plus il nous a permis de nous familiariser avec le langage python, langage que nous n'avions jamais étudié auparavant.
- ▶ Nous l'avons trouvé très intéressant pour une première approche du monde de la data et nous espérons réutiliser ses technologies dans le futur.

Des questions ?