



**UNIVERSITÉ IBN ZOHR**  
**FACULTÉ DES SCIENCES**  
**DÉPARTEMENT D'INFORMATIQUE**

**Master Spécialisé**  
Systèmes Informatiques Distribués & Big Data

---

**Analyse en temps réel des données de trafic aérien à l'aide  
d'Apache Spark, Kafka Framework, Elasticsearch et Kibana**

---

**Réaliser par :**

Boujarfaoui Ayman

En-nahel Aissam

Ayoub Ellaouzi

El-adarissi Abdelaziz

**Encadré par :**

Professeur Mr. Karim Afdel

Année universitaire 2022/2023



# Table des matières

<b>1</b>	<b>Apache Kafka Architecture &amp; Fundamentals</b>	<b>2</b>
1.1	Introduction . . . . .	2
1.2	Kafka Concepts . . . . .	3
1.3	Architecture . . . . .	5
<b>2</b>	<b>Spark Streaming</b>	<b>7</b>
2.1	Introduction . . . . .	7
2.2	Architecture de Spark Streaming . . . . .	8
2.2.1	Stream Processing Model de Spark Streaming . . . . .	9
2.2.2	Comparaison avec le traitement par lots (Batch Processing) . . . . .	9
2.2.3	Spark Streaming Concepts . . . . .	10
<b>3</b>	<b>Configuration d'environnement</b>	<b>11</b>
3.1	Téléchargement . . . . .	11
3.2	Installation . . . . .	14
3.3	Variables d'environnement . . . . .	19
3.4	Kibana et Elasticsearch . . . . .	24
<b>4</b>	<b>Test d'Application</b>	<b>26</b>

# Table des figures

# Introduction

Notre projet, intitulé "Analyse en temps réel des données de trafic aérien à l'aide d'Apache Spark et Kafka Framework", vise à mettre en œuvre un système de traitement en temps réel des données de trafic aérien en utilisant deux technologies : Apache Kafka et Apache Spark.

L'objectif principal est de collecter les données de trafic aérien en temps réel à partir de l'API "Flight Tracker API and Data - Aviation database and API" et de les analyser en continu pour fournir des informations en temps réel sur le trafic aérien.

Apache Kafka joue un rôle crucial en tant que système de messagerie distribuée, facilitant ainsi la collecte et la gestion des flux de données en temps réel. Les données de trafic aérien collectées seront acheminées vers les topics correspondants, où elles seront prêtes à être traitées par Apache Spark. D'autre part, Apache Spark est un framework de traitement de données en temps réel offrant des fonctionnalités avancées de traitement distribué et d'analyse. En configurant Apache Spark Streaming, nous pourrions mettre en place des tâches de traitement en temps réel pour analyser les données de trafic aérien provenant de Kafka.

La relation étroite entre Apache Kafka et Apache Spark est cruciale dans ce projet, car Kafka agit en tant que fournisseur de données en temps réel pour Spark. En utilisant Kafka comme source de données pour Spark Streaming, nous assurons un flux continu de données pour une analyse en temps réel efficace.

Ce projet offre une opportunité passionnante de mettre en pratique nos connaissances sur Apache Spark et Kafka Framework, en développant un système de traitement en temps réel fonctionnel pour l'analyse des données de trafic aérien.

# Chapitre 1

## Apache Kafka Architecture & Fundamentals

### 1.1 Introduction

Apache Kafka est une plate-forme logicielle de traitement de flux open source développée par LinkedIn et donnée à Apache Software Foundation, écrite en Scala et Java. Le projet vise à fournir une plate-forme unifiée, à haut débit et à faible latence pour la gestion des flux de données en temps réel. Son concept architectural de base est un journal immuable de messages qui peuvent être organisés en rubriques pour être consommés par plusieurs utilisateurs ou applications. Un journal de validation du système de fichiers ou de la base de données conserve un enregistrement permanent de tous les messages afin que Kafka puisse les relire pour maintenir un état système cohérent.

Kafka stocke les données de manière durable, de manière sérialisée. Il distribue les données sur un cluster de nœuds, offrant des performances à grande échelle qui résistent aux pannes.

Kafka reçoit et envoie des données sous forme d'événements appelés messages qui sont organisés en rubriques qui sont « publiées » par les producteurs de données et « souscrites » par les consommateurs de données, c'est pourquoi Kafka est parfois appelé un système « pub/sub ». Un événement est simplement une déclaration selon laquelle quelque chose s'est produit dans le monde réel. Un événement peut également être appelé enregistrement ou message.

Apache Kafka est une plateforme de streaming qui bénéficie de trois fonctionnalités :

1. Elle vous permet de publier et souscrire à un flux d'enregistrements. Elle ressemble ainsi à une file de message ou un système de messaging d'entreprise.

2. Elle permet de stocker des flux d'enregistrements d'une façon tolérante aux pannes.
3. Elle vous permet de traiter (au besoin) les enregistrements au fur et à mesure qu'ils arrivent.

Les principaux avantages de Kafka sont :

1. **La fiabilité** : Kafka est distribué, partitionné, répliqué et tolèrent aux fautes.
2. **La scalabilité** : Kafka se met à l'échelle facilement et sans temps d'arrêt.
3. **La durabilité** : Kafka utilise un commit log distribué, ce qui permet de stocker les messages sur le disque le plus vite possible.
4. **La performance** : Kafka a un débit élevé pour la publication et l'abonnement.

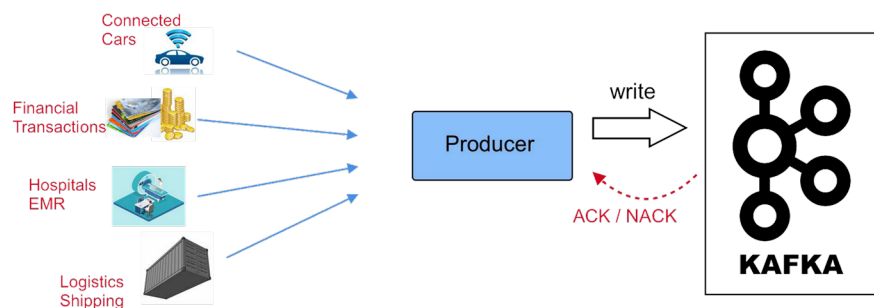
## 1.2 Kafka Concepts

Pour comprendre le fonctionnement de Kafka, il faut d'abord se familiariser avec le vocabulaire suivant :

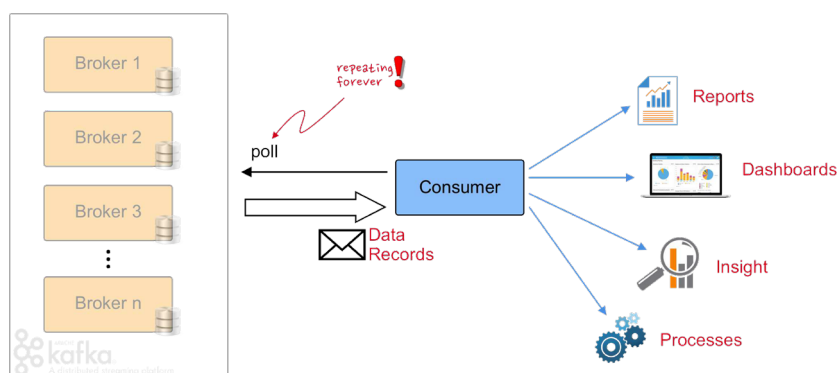
- **Topic** : Un flux de messages appartenant à une catégorie particulière. Les données sont stockées dans des topics.
- **Partitions** : Chaque topic est divisé en partitions. Pour chaque topic, Kafka conserve un minimum d'une partition. Chaque partition contient des messages dans une séquence ordonnée immuable. Une partition est implémentée comme un ensemble de segments de tailles égales.
- **Offset** : Les enregistrements d'une partition ont chacun un identifiant séquentiel appelé offset, qui permet de l'identifier de manière unique dans la partition.
- **Replicas** : Les répliques sont des Backups d'une partition. Elles ne sont jamais lues ni modifiées par les acteurs externes, elles servent uniquement à prévenir la perte de données.
- **Brokers** : Les brokers (ou courtiers) sont de simples systèmes responsables de maintenir les données publiées. Chaque courtier peut avoir zéro ou plusieurs partitions par topic. Si

un topic admet N partitions et N courtiers, chaque courtier va avoir une seule partition. Si le nombre de courtiers est plus grand que celui des partitions, certains n'auront aucune partition de ce topic.

- **Cluster** : Un système Kafka ayant plus qu'un seul Broker est appelé cluster Kafka. L'ajout de nouveau broker est fait de manière transparente sans temps d'arrêt.
- **Producers** : Les producteurs sont les éditeurs de messages à un ou plusieurs topics Kafka. Ils envoient des données aux courtiers Kafka. Chaque fois qu'un producteur publie un message à un courtier, ce dernier rattache le message au dernier segment, ajouté ainsi à une partition. Un producteur peut également envoyer un message à une partition particulière.



- **Consumers** : Les consommateurs lisent les données à partir des brokers. Ils souscrivent à un ou plusieurs topics, et consomment les messages publiés en extrayant les données à partir des brokers.



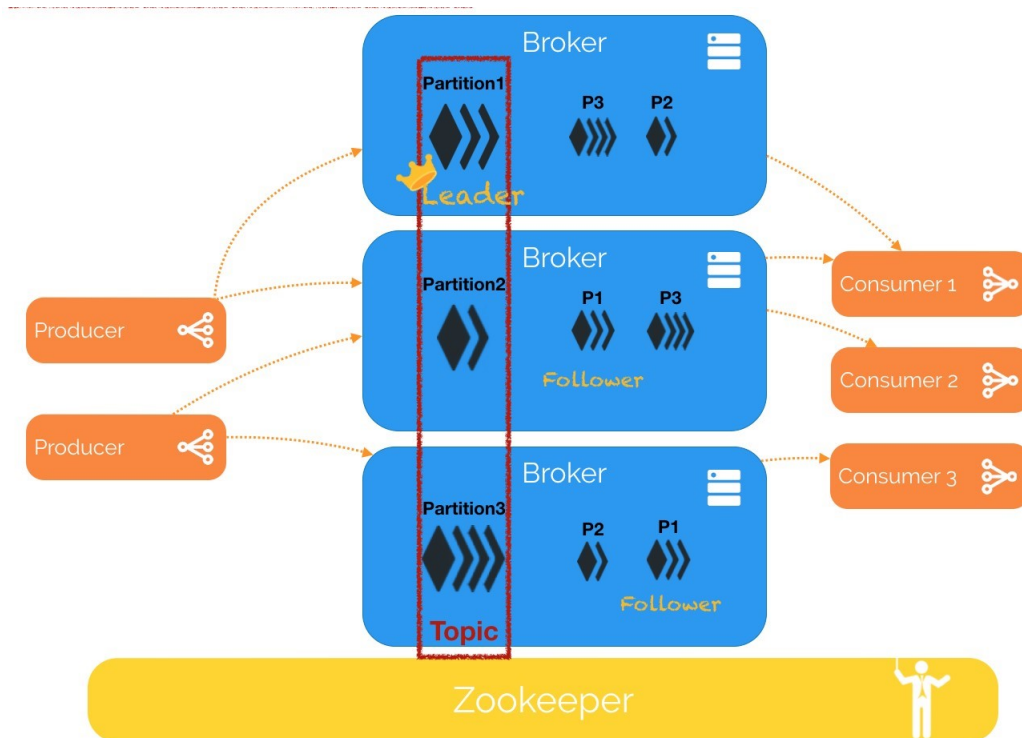
- **Leaders** : Le leader est le nœud responsable de toutes les lectures et écritures d'une partition donnée. Chaque partition a un serveur jouant le rôle de leader.



- **Followers** : C'est un nœud qui suit les instructions du leader. Si le leader tombe en panne, l'un des followers deviendra automatiquement le nouveau leader.

## 1.3 Architecture

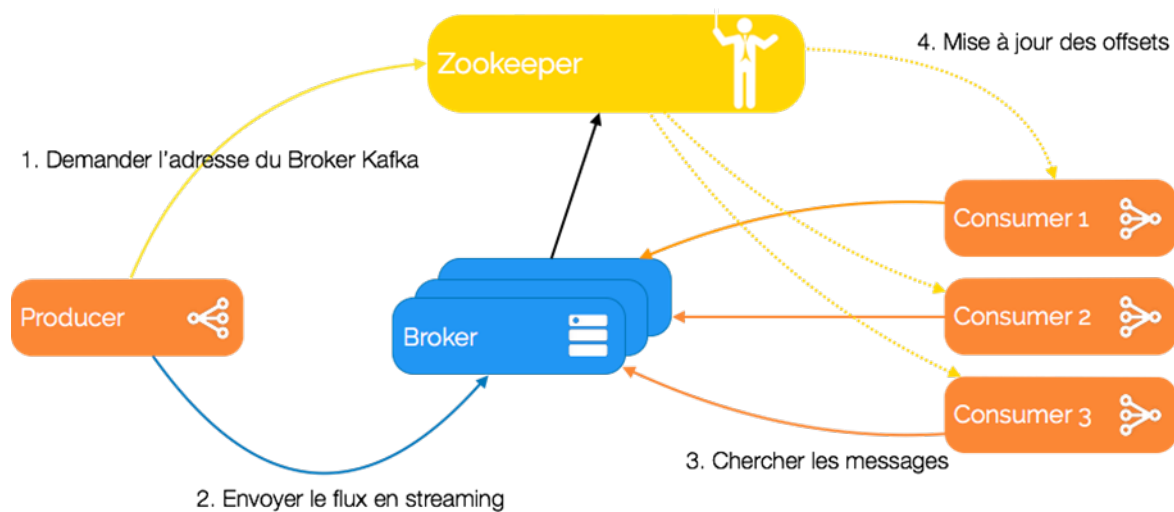
La figure suivante montre un exemple de flux entre les différentes parties d'un système Kafka :



Dans cet exemple, un topic est configuré en trois partitions. En supposant que, si le facteur de réplication du topic est de 3, alors Kafka va créer trois répliques identiques de chaque partition et les placer dans le cluster pour les rendre disponibles pour toutes les opérations. L'identifiant de la réplique est le même que l'identifiant du serveur qui l'héberge. Pour équilibrer la charge dans le cluster, chaque broker stocke une ou plusieurs de ces partitions. Plusieurs producteurs et consommateurs peuvent publier et extraire les messages au même moment.

## Kafka et Zookeeper

Zookeeper est un service centralisé permettant de maintenir l'information de configuration, de nommage, de synchronisation et de services de groupe. Ces services sont utilisés par les applications distribuées en général, et par Kafka en particulier. Pour éviter la complexité et difficulté de leur implémentation manuelle, Zookeeper est utilisé.



Un cluster Kafka consiste typiquement en plusieurs courtiers (Brokers) pour maintenir la répartition de charge. Ces courtiers sont stateless, c'est pour cela qu'ils utilisent Zookeeper pour maintenir l'état du cluster. Un courtier peut gérer des centaines de milliers de lectures et écritures par seconde, et chaque courtier peut gérer des téra-octets de messages sans impact sur la performance. Zookeeper est utilisé pour gérer et coordonner les courtiers Kafka. Il permet de notifier les producteurs et consommateurs de messages de la présence de tout nouveau courtier, ou de l'échec d'un courtier dans le cluster.

# Chapitre 2

## Spark Streaming

### 2.1 Introduction

Spark Streaming est un composant essentiel de l'écosystème Apache Spark, conçu pour le traitement des données en temps réel et la gestion de flux de données à grande échelle. Contrairement au traitement par lots traditionnel où les données sont traitées en blocs, Spark Streaming permet de traiter continuellement les données en flux, ce qui le rend idéal pour les applications nécessitant des analyses en temps réel.

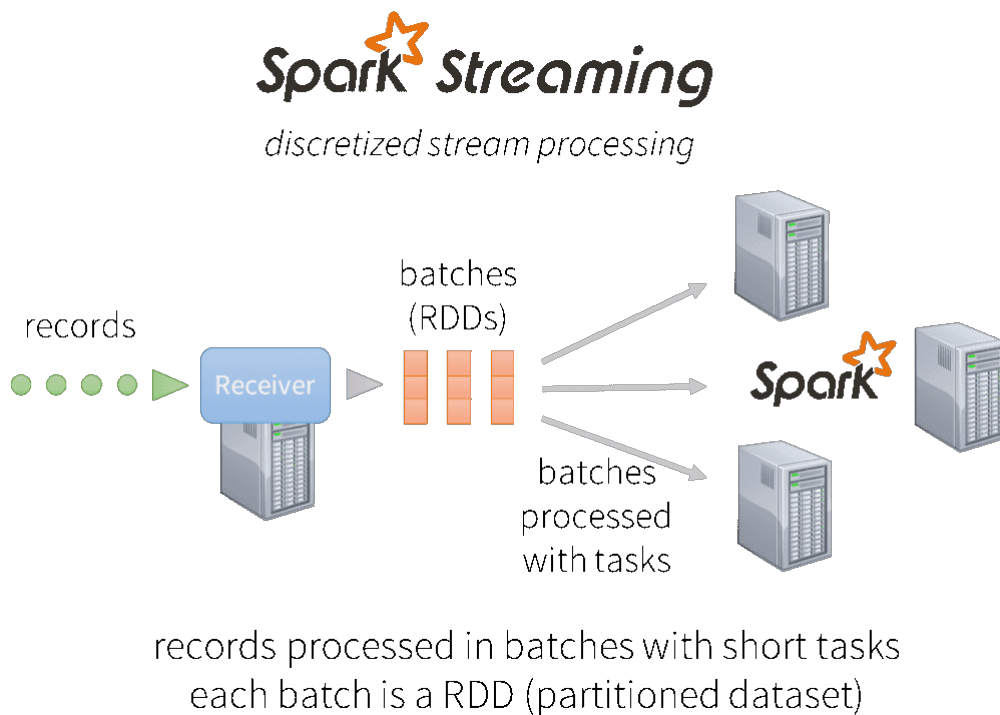
Les principaux objectifs de l'utilisation de Spark Streaming étaient les suivants :

- **Temps réel** : Notre projet exigeait une analyse en temps réel des données pour prendre des décisions rapides et réagir rapidement aux changements dans les données d'entrée. Spark Streaming nous permettait de traiter les données en continu, assurant ainsi une faible latence dans le traitement.
- **Évolutivité** : Étant donné que les sources de données génèrent un grand volume de flux de données en continu, il était essentiel d'avoir une architecture hautement évolutive. Spark Streaming peut facilement s'adapter à l'augmentation du débit en ajoutant simplement des nœuds au cluster.
- **Tolérance aux pannes** : Dans un environnement de traitement en temps réel, la tolérance aux pannes est cruciale pour garantir la continuité du service. Spark Streaming assure la reprise après une défaillance en répliquant les données et en maintenant un état cohérent entre les nœuds du cluster.
- **Intégration avec Apache Spark** : Étant intégré à Apache Spark, Spark Streaming peut profiter de toutes les fonctionnalités et optimisations du framework Spark. Cela inclut

l'utilisation de RDDs (Resilient Distributed Datasets) pour la manipulation des données et l'intégration aisée avec d'autres composants de Spark tels que Spark SQL, Spark MLlib, et Spark GraphX.

Le traitement des données en temps réel est essentiel dans de nombreuses applications, telles que la surveillance de l'Internet des Objets (IoT), l'analyse de données de suivi en temps réel, les analyses de médias sociaux, la détection d'anomalies, etc. Spark Streaming répond à ce besoin en offrant une approche distribuée et évolutive pour traiter ces flux de données en temps réel.

### 2.2 Architecture de Spark Streaming



Spark Streaming est un module de traitement en temps réel intégré à Apache Spark qui permet de traiter des flux de données en continu. Il offre un modèle de traitement en flux qui se distingue du traitement par lots traditionnel en permettant le traitement des données à mesure qu'elles arrivent plutôt que de les traiter par groupes (blocs) à intervalles réguliers. Voici une explication du modèle de traitement des données en flux de Spark Streaming, suivi d'une comparaison avec le traitement par lots, ainsi qu'une introduction aux concepts de base de Spark Streaming

### 2.2.1 Stream Processing Model de Spark Streaming

- **Micro-batch** : Le modèle de traitement des données en flux de Spark Streaming repose sur le concept de micro-batch. Plutôt que de traiter chaque enregistrement de manière individuelle, Spark Streaming divise le flux continu de données en petits lots ou fenêtres de temps appelés micro-batch. Chaque micro-batch est traité comme une mini-batch par le moteur de traitement distribué de Spark.
- **Ingestion des Données** : Le flux de données continu est ingéré par Spark Streaming à partir de sources telles que Kafka, Kinesis, Flume ou d'autres systèmes de collecte de données en temps réel. Ces flux de données sont ensuite divisés en micro-batch de durée fixe, par exemple, chaque seconde ou chaque quelques secondes, en fonction de la configuration définie par l'utilisateur.
- **Transformation des Données** : Une fois que les micro-batch sont créés, Spark Streaming applique les transformations et les opérations de traitement spécifiées par l'utilisateur sur chaque micro-batch. Le résultat de chaque micro-batch peut être stocké dans un état intermédiaire ou renvoyé à une destination finale, comme une base de données, un système de stockage distribué, ou un autre système tiers.
- **Analyse en Temps Réel** : Le modèle de traitement en flux permet ainsi à Spark Streaming de fournir des analyses en temps réel avec une latence faible, ce qui le rend adapté à un large éventail d'applications allant de l'IoT à l'analyse de données en temps réel.

### 2.2.2 Comparaison avec le traitement par lots (Batch Processing)

Le traitement par lots est le mode de traitement traditionnel où les données sont collectées et traitées par groupes (blocs) à intervalles réguliers, généralement sur des données historiques. Cela peut être efficace pour des tâches d'analyse rétrospective, mais il présente des inconvénients pour le traitement en temps réel :

- **Latence élevée** : Le traitement par lots nécessite d'attendre la fin de la période de collecte pour commencer le traitement, ce qui entraîne une latence élevée entre la collecte des données et la disponibilité des résultats.
- **Difficulté à réagir en temps réel** : Dans certaines applications, il est essentiel de réagir rapidement aux changements dans les données, ce qui n'est pas possible avec le traitement

par lots en raison de la latence.

- **Non adapté aux flux continus** : Le traitement par lots est conçu pour des données statiques, tandis que Spark Streaming est conçu spécifiquement pour le traitement des flux de données continus.

### 2.2.3 Spark Streaming Concepts

- **DStream (Discretized Stream)** : C'est l'abstraction de base de Spark Streaming. Un DStream est une séquence de RDDs (Resilient Distributed Datasets) représentant le flux de données continu. Chaque RDD du DStream représente un micro-batch.
- **Transformations** : Comme dans Spark RDD, Spark Streaming prend en charge des transformations telles que map, filter, reduceByKey, join, etc., pour traiter les données dans chaque micro-batch.
- **Actions** : Les actions sont des opérations qui déclenchent le traitement des données dans Spark Streaming. Cela peut être le stockage des résultats dans un système tiers, l'affichage des résultats à la console, etc.
- **Fenêtres de traitement** : Spark Streaming prend en charge les fenêtres de traitement qui permettent de regrouper les données sur des périodes spécifiques plutôt que sur un seul micro-batch. Cela permet d'effectuer des analyses basées sur des fenêtres de temps glissantes ou des fenêtres d'une durée fixe.

En combinant ces concepts, Spark Streaming fournit un modèle de traitement en temps réel hautement évolutif et tolérant aux pannes pour le traitement efficace de flux de données continus.

# Chapitre 3

## Configuration d'environnement

### 3.1 Téléchargement

- Télécharger JDK 1.8 dans le site officiel d'oracle  
<https://www.oracle.com/java/technologies/downloads/>

Java 8   Java 8 Enterprise Performance Pack   **Java 11**

---

**Java SE Development Kit 11.0.20**

Java downloads   Tools and resources   Java archive

Commercial license and support are available for a low cost with [Java SE Universal Subscription](#).

JDK 11 software is licensed under the [Oracle Technology Network License Agreement for Oracle Java SE](#).

JDK 11.0.20 [checksums](#)

Linux   macOS   Solaris   **Windows**

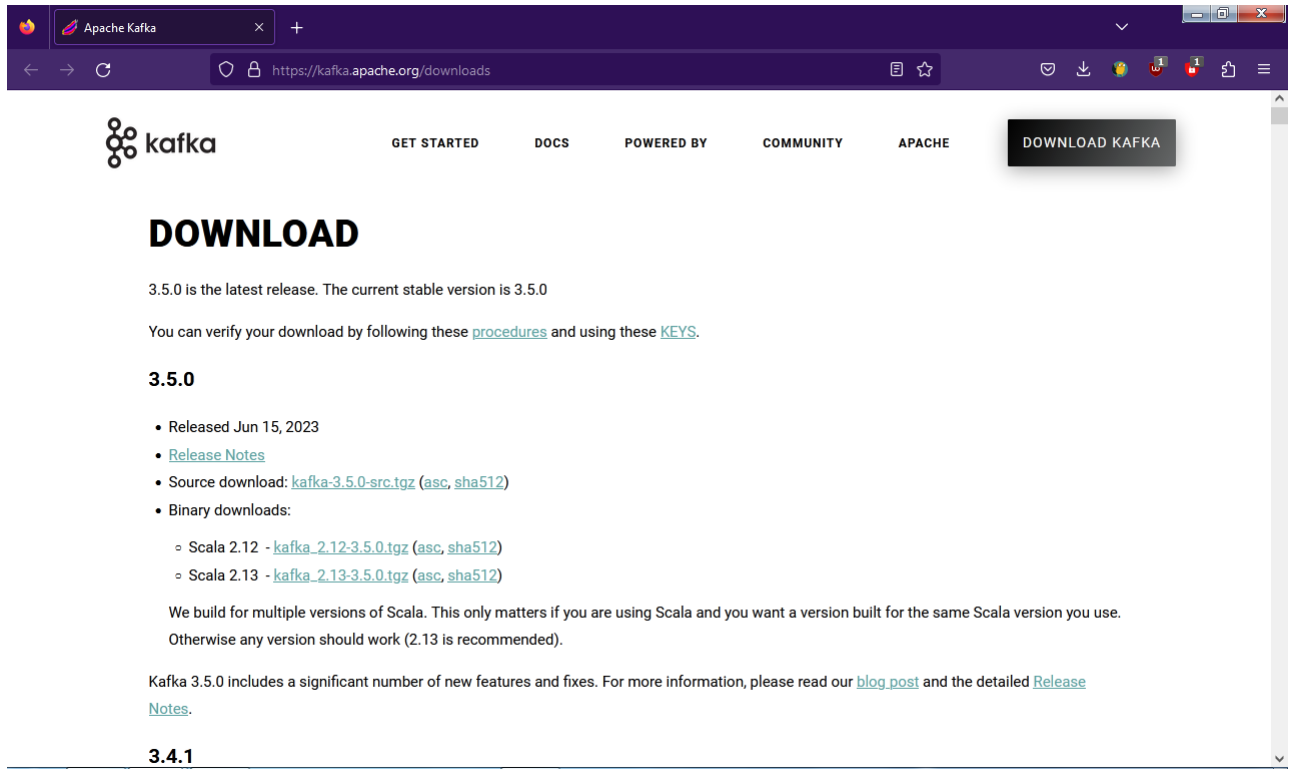
---

Product/file description	File size	Download
x64 Installer	141.39 MB	<a href="#">jdk-11.0.20_windows-x64_bin.exe</a>
x64 Compressed Archive	159.14 MB	<a href="#">jdk-11.0.20_windows-x64_bin.zip</a>

[Documentation Download](#)

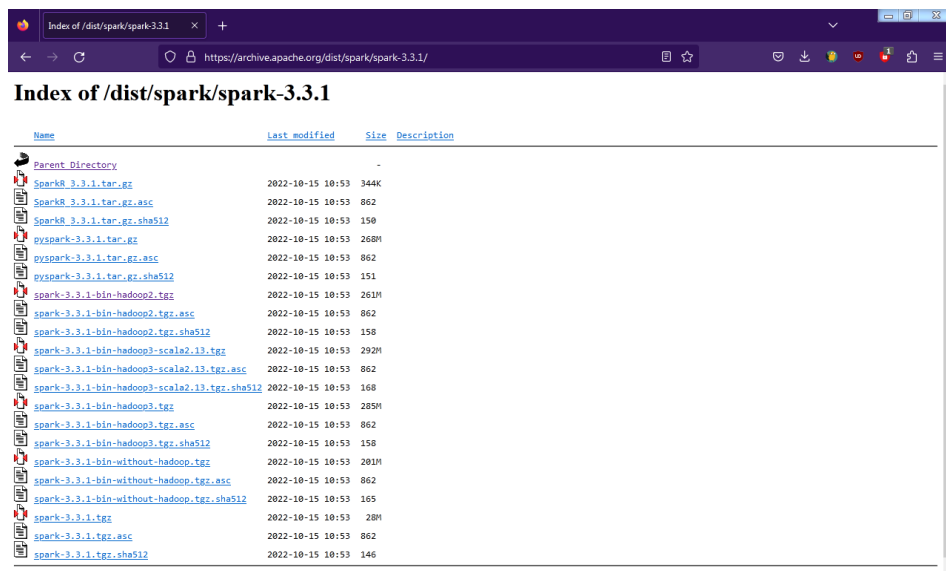
- Téléchargement d'Apache Kafka  
<https://kafka.apache.org/downloads>

## Chapitre 3 Configuration d'environnement



- Téléchargement d'Apache Spark

<https://archive.apache.org/dist/spark/spark-3.3.1/>

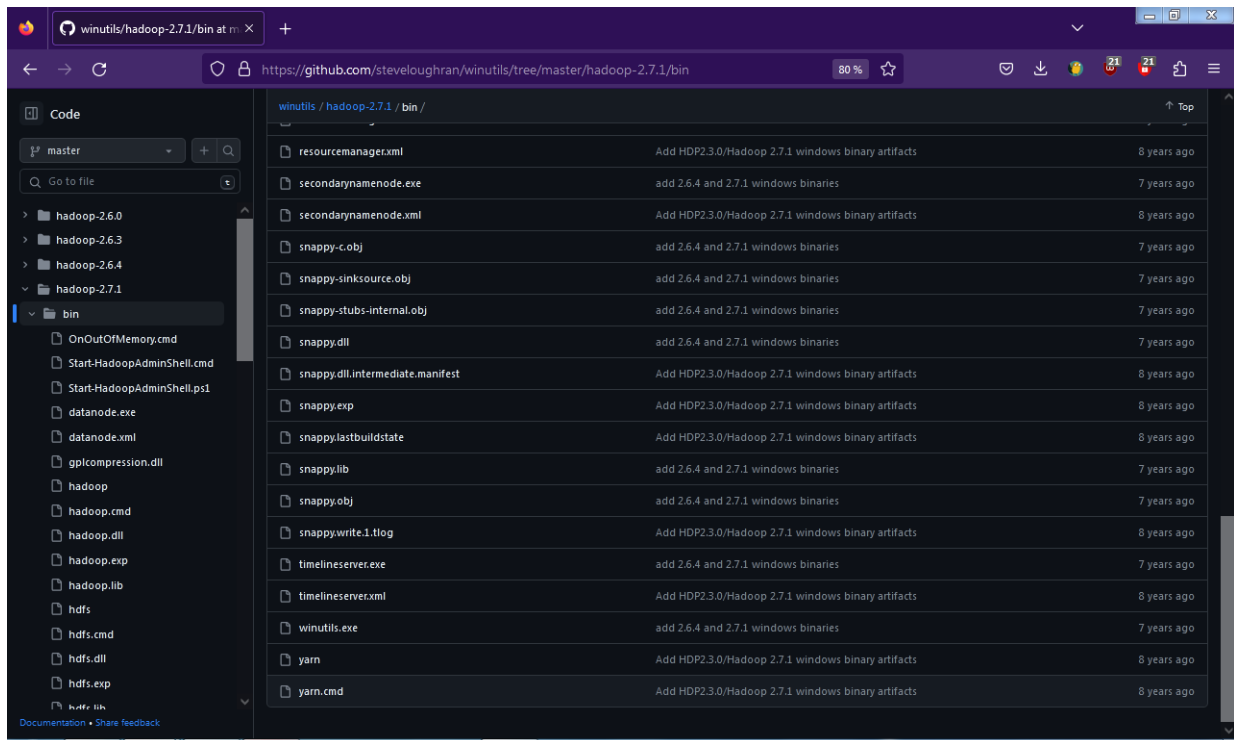


- Téléchargement de la version appropriée de winutils.exe pour Hadoop

<https://github.com/stevcloughran/winutils/blob/master/hadoop-2.7.1/bin/winutils.exe>

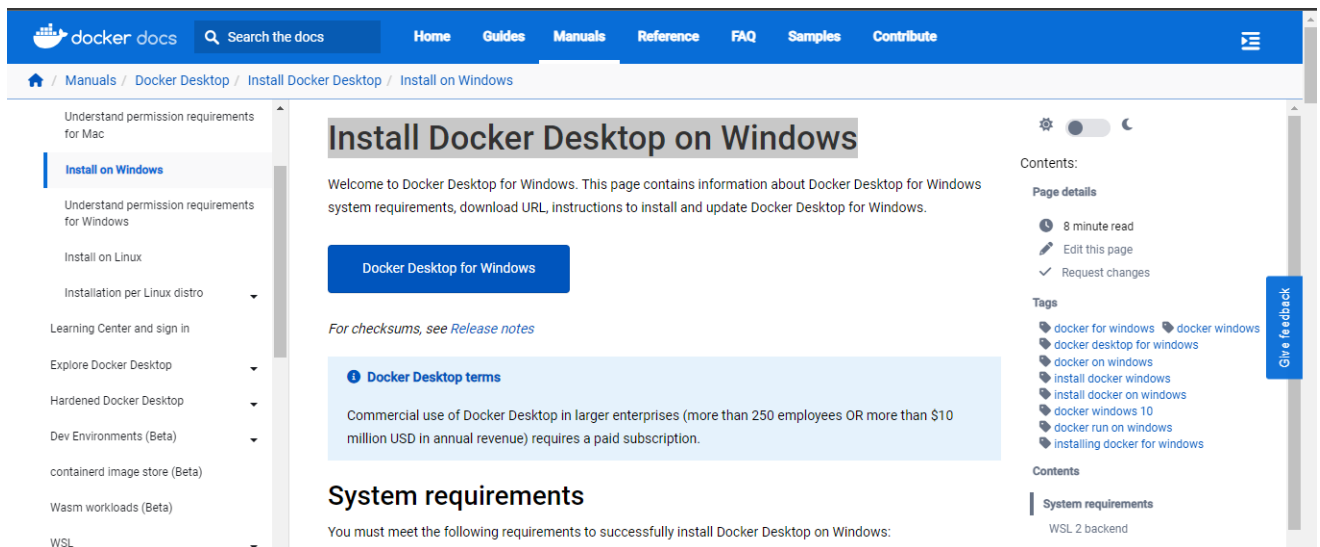


## Chapitre 3 Configuration d'environnement



- Install Docker Desktop on Windows

<https://docs.docker.com/desktop/install/windows-install/>



- Télécharger le package de mise à jour du noyau Linux WSL

<https://learn.microsoft.com/fr-fr/windows/wsl/install-manual#step-4---/download-the-linux-kernel-update-package>

## Chapitre 3 Configuration d'environnement

Filter by title

Documentation WSL

> Vue d'ensemble

> Installer

Install WSL 2

Étapes d'installation manuelle pour les anciennes versions

Installation sur Windows Server

> Tutoriels

> Concepts

> Procédures

Forum Aux Questions (FAQ)

Résolution des problèmes

> Notes de publication

Télécharger le PDF

dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /nores

Redémarrez votre ordinateur pour terminer l'installation de WSL et mettre à jour vers WSL 2.

Étape 4 : télécharger le package de mise à jour du noyau Linux

1. Téléchargez le dernier package :

- Package de mise à jour du noyau Linux WSL2 pour machines x64

Notes

Si vous utilisez une machine ARM64, téléchargez le package ARM64 à la place. Si vous n'êtes pas sûr du type de machine que vous avez, ouvrez l'invite de commandes ou PowerShell et entrez : `systeminfo | find "System Type"`.  
**Avertissement** : Sur les versions non anglaises de Windows, vous devrez probablement modifier le texte recherché et traduire la chaîne « System Type ». Vous devrez peut-être aussi échapper les guillemets pour la commande find. Par exemple, en allemand `systeminfo | find "Systemtyp"`.

Ressources supplémentaires

Documentation

Configurer un environnement de développement WSL

Configurez un environnement de développement WSL à l'aide des meilleures pratiques de ce guide d'ensemble par étape....

Résolution des problèmes liés au sous-système Windows pour Linux

Cet article fournit des informations détaillées sur les erreurs et problèmes courants auxquels peuvent être confrontés les utilisateurs...

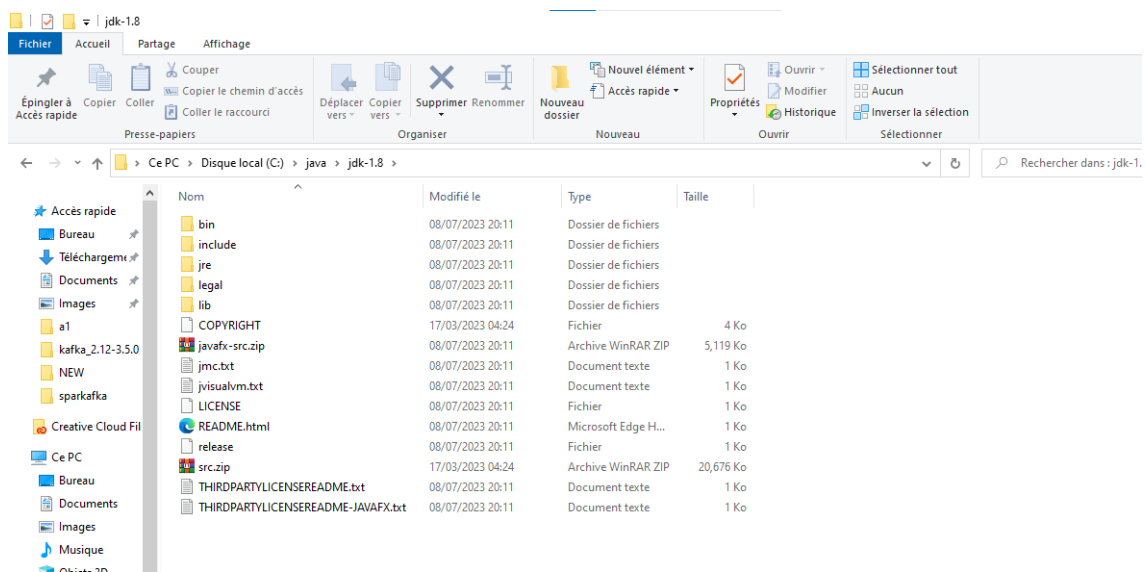
Comparaison des versions WSL

WSL 2 offre les avantages de WSL 1, mais utilise un noyau Linux réel, au lieu d'une couche de traduction comme WSL 1, ce qui...

Afficher 5 de plus

## 3.2 Installation

- Après l'installation du Java `jdk-8u371-windows-x64.exe` dans `C :/Java/`



## Chapitre 3 Configuration d'environnement

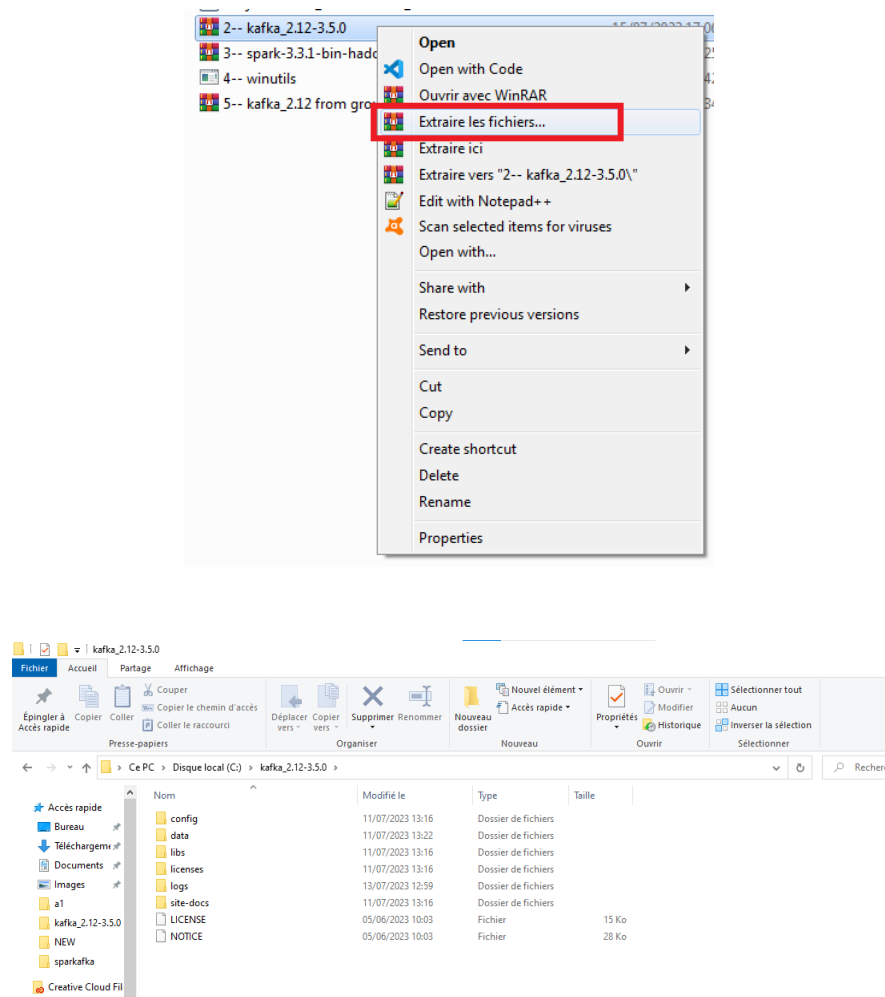
- Test JDK Version

```
C:\Windows\system32\CMD.exe

C:\Users\Paname>java -version
java version "1.8.0_371"
Java(TM) SE Runtime Environment (build 1.8.0_371-b11)
Java HotSpot(TM) 64-Bit Server VM (build 25.371-b11, mixed mode)

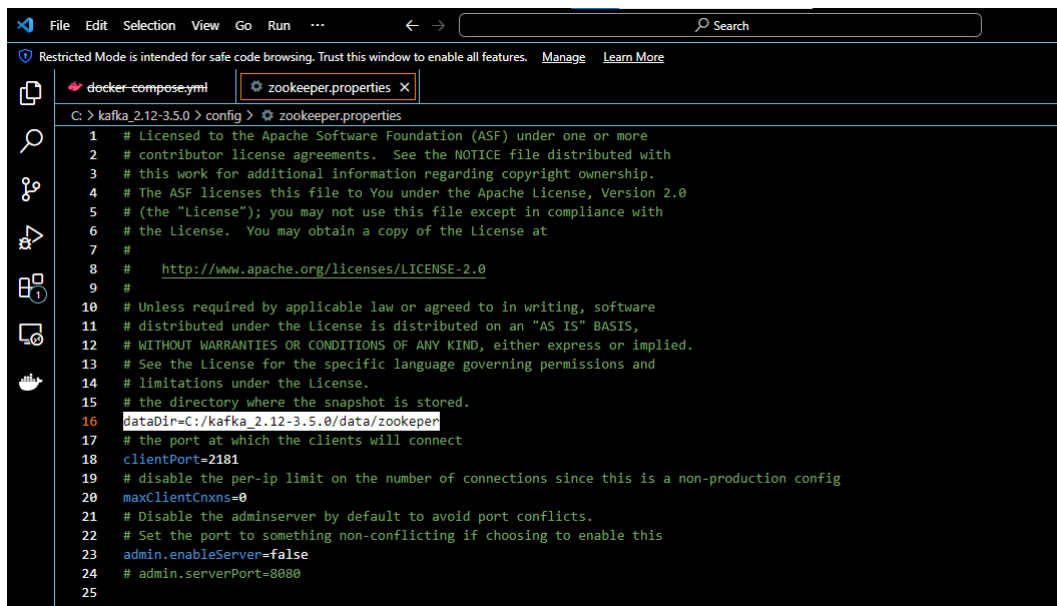
C:\Users\Paname>
```

- Après On va décompresser l'archive kafka\_2.12-3.5.0.tgz dans le répertoire **C:/**



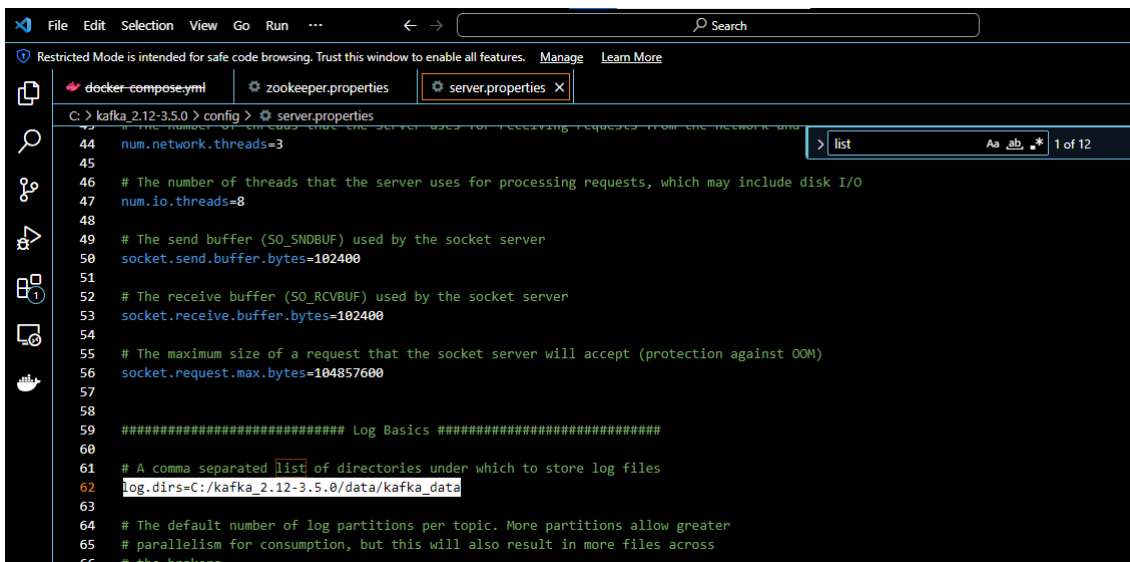
## Chapitre 3 Configuration d'environnement

- On va modifier une ligne dataDir dans `C:\kafka\_2.12-3.5.0\config\zookeeper.properties` par `dataDir=C:/kafka_2.12-3.5.0/data/zookeeper`



```
1 # Licensed to the Apache Software Foundation (ASF) under one or more
2 # contributor license agreements. See the NOTICE file distributed with
3 # this work for additional information regarding copyright ownership.
4 # The ASF licenses this file to You under the Apache License, Version 2.0
5 # (the "License"); you may not use this file except in compliance with
6 # the license. You may obtain a copy of the License at
7 #
8 # http://www.apache.org/licenses/LICENSE-2.0
9 #
10 # Unless required by applicable law or agreed to in writing, software
11 # distributed under the license is distributed on an "AS IS" BASIS,
12 # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
13 # See the license for the specific language governing permissions and
14 # limitations under the license.
15 # the directory where the snapshot is stored.
16 dataDir=C:/kafka_2.12-3.5.0/data/zookeeper
17 # the port at which the clients will connect
18 clientPort=2181
19 # disable the per-ip limit on the number of connections since this is a non-production config
20 maxClientCnxns=0
21 # Disable the adminserver by default to avoid port conflicts.
22 # Set the port to something non-conflicting if choosing to enable this
23 admin.enableServer=false
24 # admin.serverPort=8080
25
```

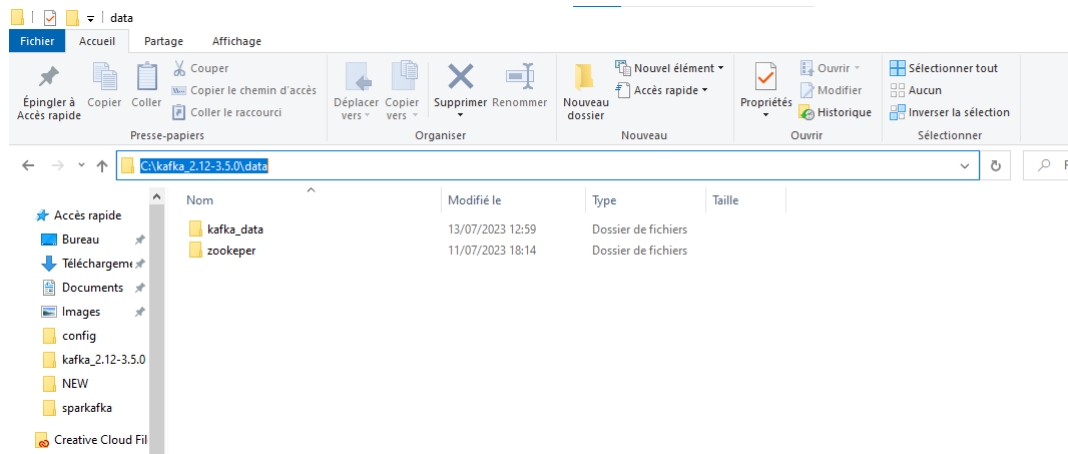
- Et On va modifier une ligne dans `C:/kafka/_2.12-3.5.0/config/server.properties` `log.dirs=C:/kafka_2.12-3.5.0/data/kafka_data`



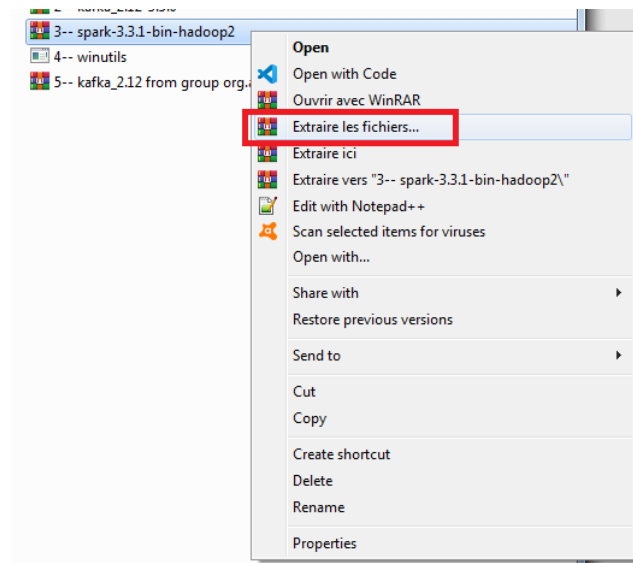
```
44 num.network.threads=3
45
46 # The number of threads that the server uses for processing requests, which may include disk I/O
47 num.io.threads=8
48
49 # The send buffer (SO_SNDBUF) used by the socket server
50 socket.send.buffer.bytes=102400
51
52 # The receive buffer (SO_RCVBUF) used by the socket server
53 socket.receive.buffer.bytes=102400
54
55 # The maximum size of a request that the socket server will accept (protection against OOM)
56 socket.request.max.bytes=104857600
57
58 ##### Log Basics #####
59
60 # A comma separated list of directories under which to store log files
61 log.dirs=C:/kafka_2.12-3.5.0/data/kafka_data
62
63 # The default number of log partitions per topic. More partitions allow greater
64 # parallelism for consumption, but this will also result in more files across
65 # the brokers.
66
```

## Chapitre 3 Configuration d'environnement

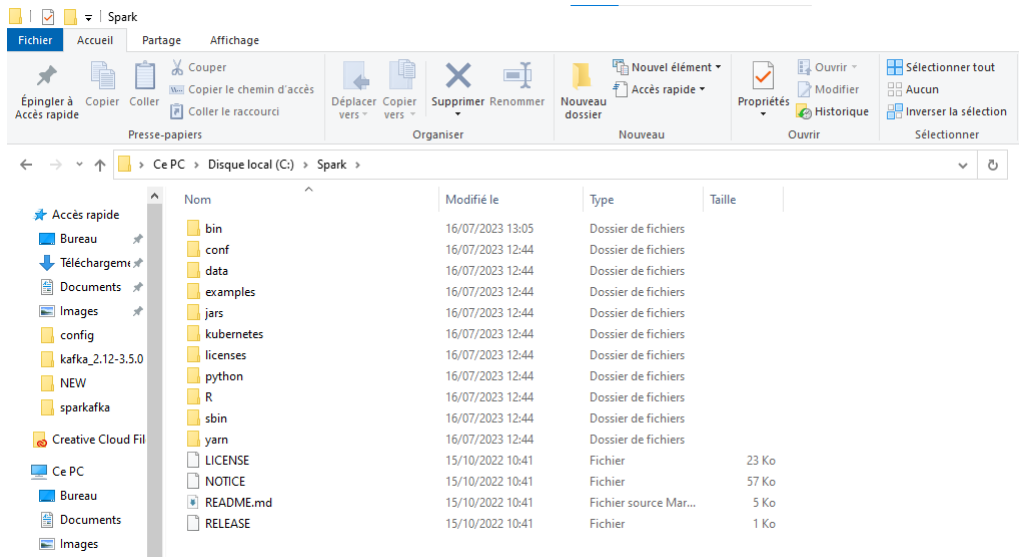
- Donc il faut créer les 2 répertoires `C :/kafka_2.12-3.5.0/data/kafka_data` et `C :/kafka_2.12-3.5.0/data/zookeeper`



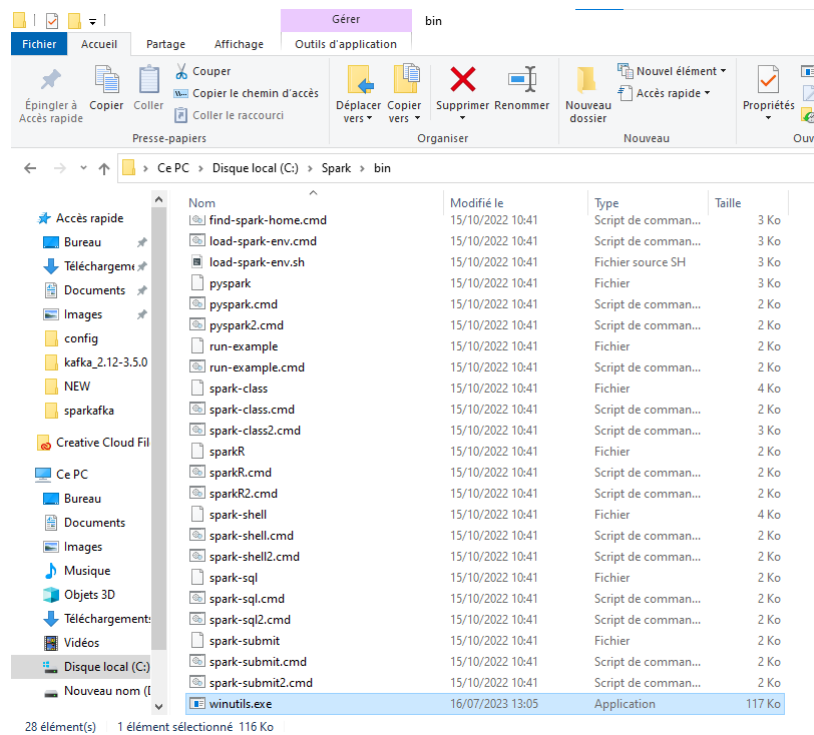
- l'étape suivante est de décompresser l'archive `spark-3.3.1-bin-hadoop2.tgz` dans le répertoire `C :/spark`



## Chapitre 3 Configuration d'environnement

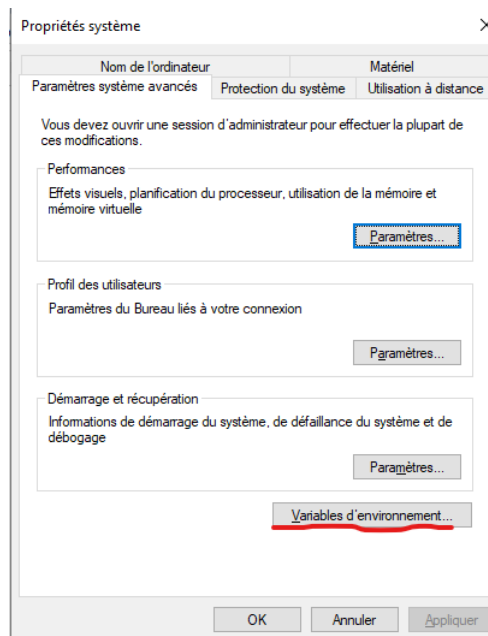
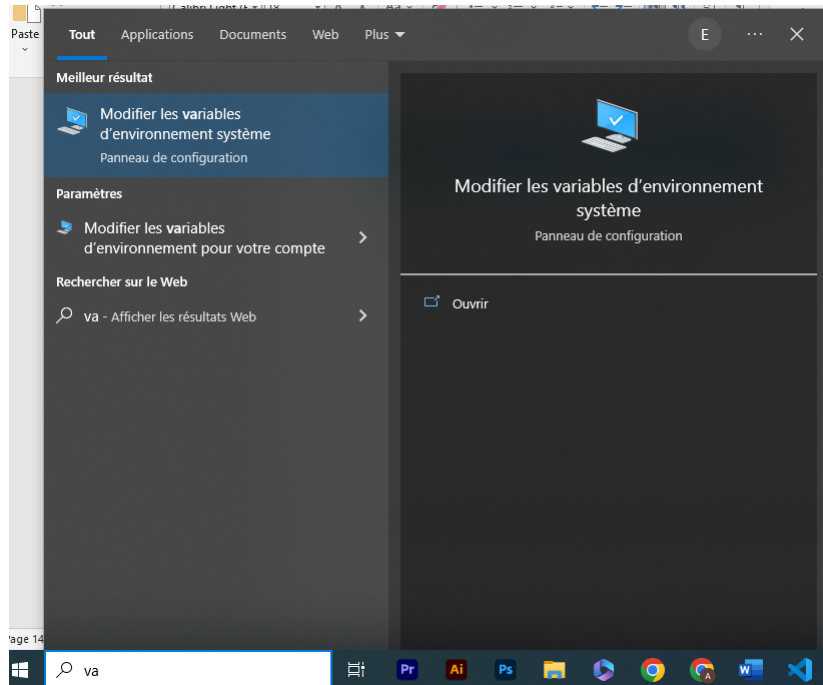


- Puis on va déplacer winutils.exe dans C :/spark/bin/



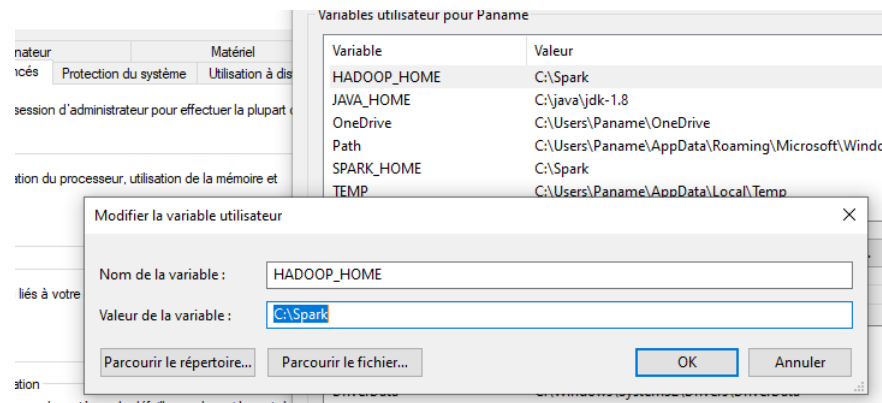
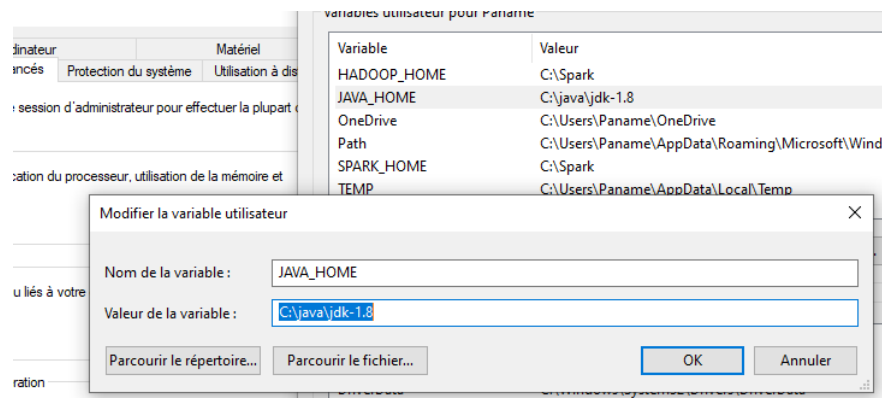
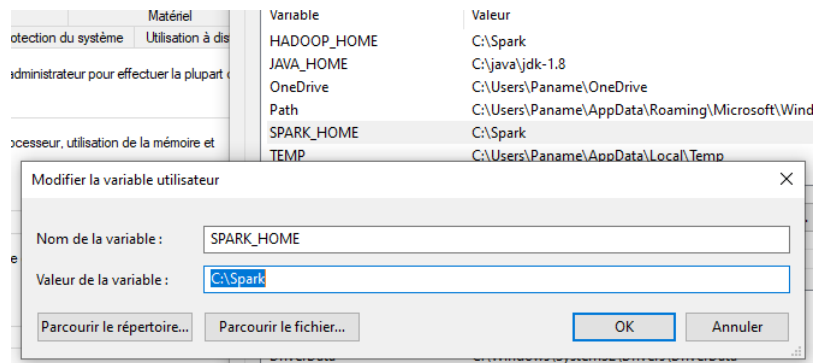
### 3.3 Variables d'environnement

- Dans la barre de recherche taper variable d'environnement



### Chapitre 3 Configuration d'environnement

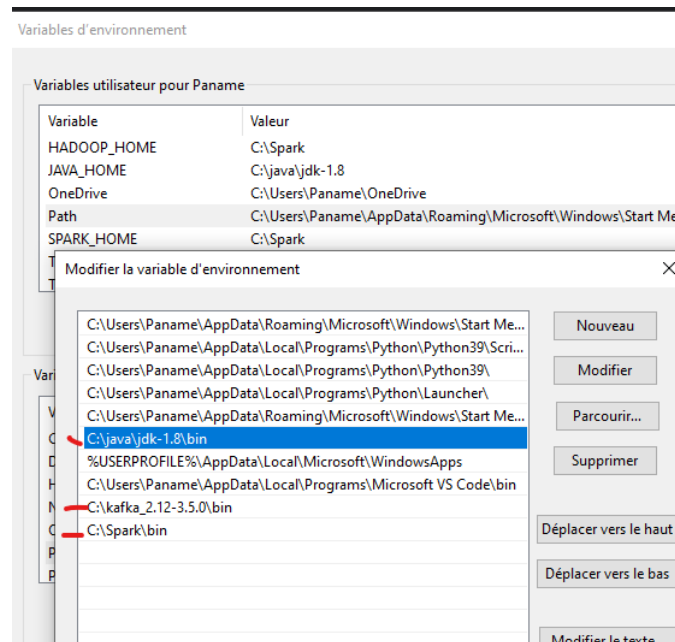
- Ajouter les paths suivants : SPARK\_HOME : C :/spark et HADOOP\_HOME : C :/spark et C :/java/jdk-1.8



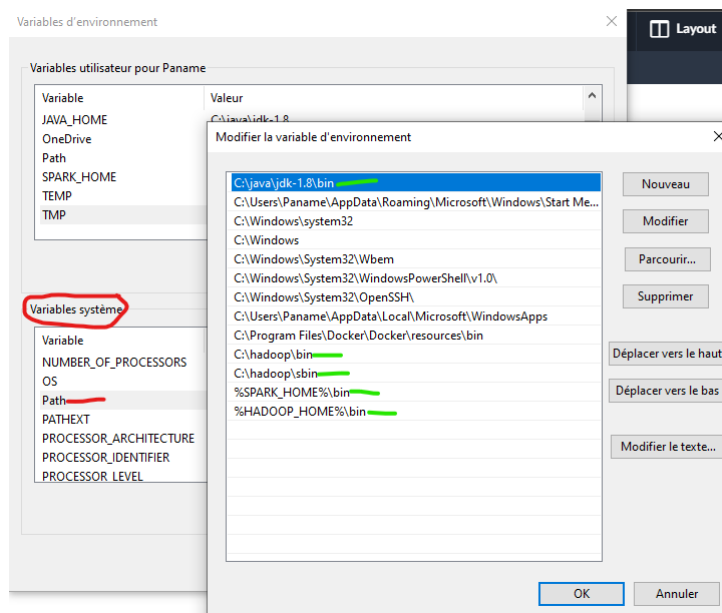


## Chapitre 3 Configuration d'environnement

- Ajouter aussi les pathes suivants dans path



- Ajouter aussi les pathes suivants dans path de système



### 1 Configuration de Docker Compose

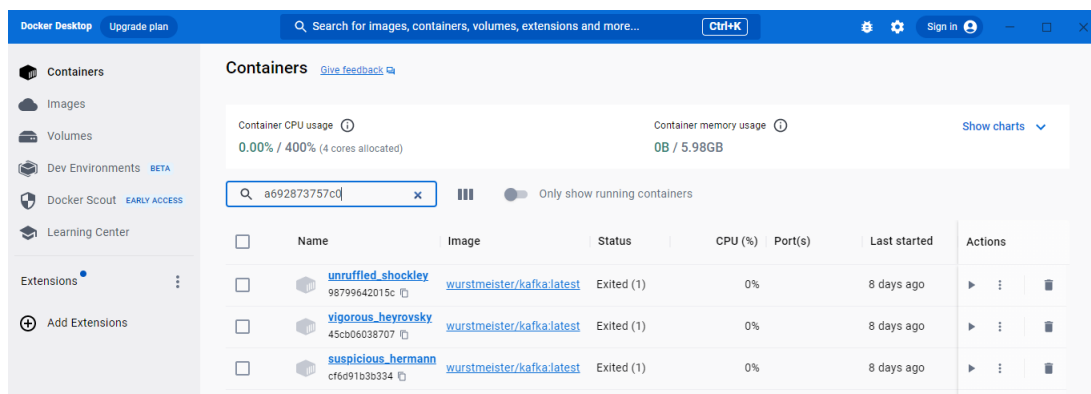
- Créer un fichier docker-compose.yml qui contient des images(containers) Zookeeper, Kafka server, Kibana et Elasticsearch

```
1 version: '3.7'
2
3 services:
4     zookeeper:
5         image: wurstmeister/zookeeper
6         ulimits:
7             nofile:
8                 soft: 65536
9                 hard: 65536
10        container_name: zookeeper
11        ports:
12            - "2181:2181"
13    kafka:
14        image: wurstmeister/kafka
15        container_name: kafka
16        ports:
17            - "9092:9092"
18        environment:
19            KAFKA_ADVERTISED_HOST_NAME: localhost
20            KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
21    elasticsearch:
22        image: docker.elastic.co/elasticsearch/elasticsearch:7.4.0
23        container_name: elasticsearch
24        restart: always
25        environment:
26            - xpack.security.enabled=false
27            - discovery.type=single-node
28        ulimits:
29            memlock:
30                soft: -1
31                hard: -1
32            nofile:
33                soft: 65536
34                hard: 65536
35        cap_add:
36            - IPC_LOCK
37        volumes:
38            - elasticsearch-data-volume:/usr/share/elasticsearch/data
39        ports:
40            - 9200:9200
```

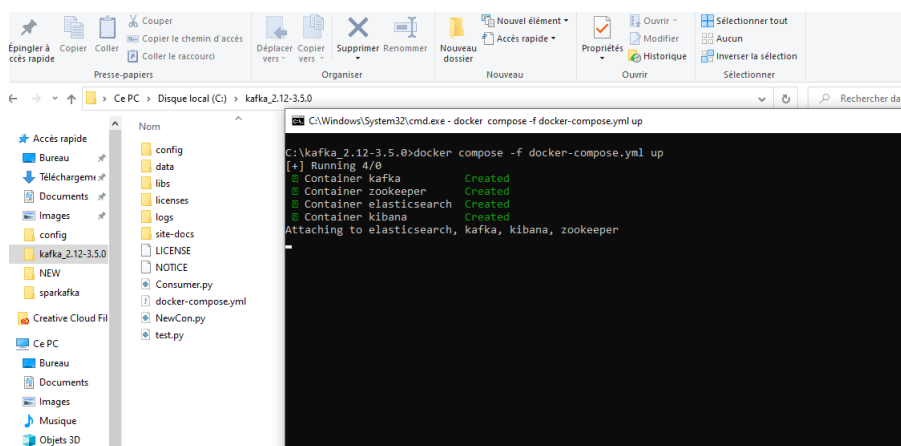
## Chapitre 3 Configuration d'environnement

```
41     - 9300:9300
42
43     kibana:
44         container_name: kibana
45         image: docker.elastic.co/kibana/kibana:7.4.0
46         restart: always
47         environment:
48             - ELASTICSEARCH_HOSTS=http://elasticsearch:9200
49         ports:
50             - 5601:5601
51         depends_on:
52             - elasticsearch
53 volumes:
54     elasticsearch-data-volume:
55         driver: local
```

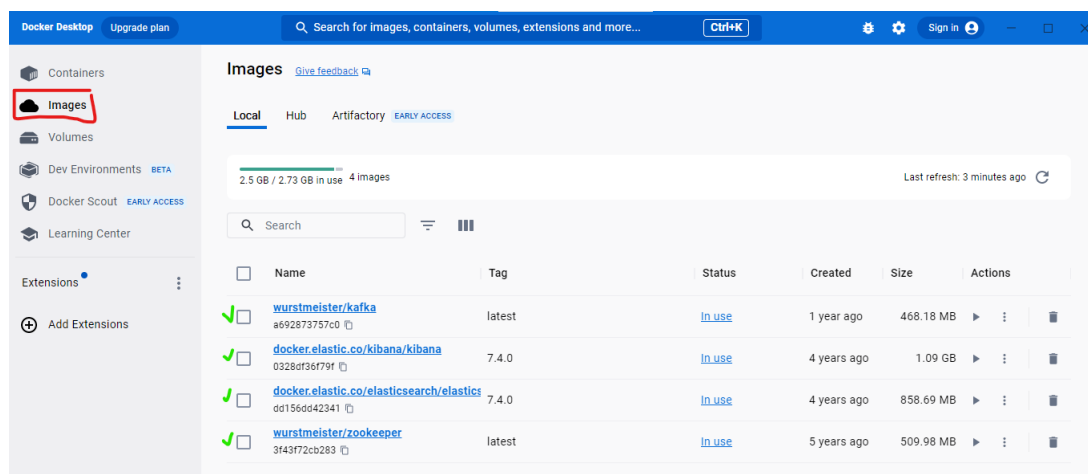
puis Lancer l'Application de Docker déjà installer



Et apres Taper la commande suivante



Finalement les containers seront bien installés



### 3.4 Kibana et Elasticsearch

#### Elasticsearch

En quelques mots, nous aidons les utilisateurs à trouver plus rapidement ce qu'ils recherchent, qu'il s'agisse de collaborateurs ayant besoin de documents sur votre intranet ou de clients sur internet en quête de la paire de chaussures idéale. Pour aller un peu plus loin sur le plan technique, voici ce qu'on pourrait dire :

Elasticsearch est un moteur de recherche et d'analyse distribué gratuit et ouvert pour toutes les données (textuelles, numériques, géospatiales, structurées et non structurées). Elasticsearch a été conçu à partir d'Apache Lucene et a été lancé en 2010 par Elasticsearch N. V. (maintenant appelé Elastic). Réputé pour ses API REST simples, sa nature distribuée, sa vitesse et sa scalabilité, Elasticsearch est le composant principal de la Suite Elastic, un ensemble d'outils gratuits et ouverts d'ingestion de données, d'enrichissement, de stockage, d'analyse et de visualisation. Couramment appelée la Suite ELK (pour Elasticsearch, Logstash et Kibana), la Suite Elastic comprend désormais une riche collection d'agents de transfert légers, appelés les agents Beats, pour envoyer des données à Elasticsearch.

#### Kibana

Kibana est une application frontend gratuite et ouverte qui s'appuie sur la Suite Elastic. Elle permet de rechercher et de visualiser les données indexées dans Elasticsearch. Si Kibana est connue pour être l'outil de représentation graphique de la Suite Elastic (précédemment appelée "la Suite ELK", acronyme d'Elasticsearch, Logstash et Kibana), elle sert aussi d'interface utilisateur pour le monitoring, la gestion et la sécurité des clusters de la Suite Elastic. Sans oublier qu'elle joue aussi le rôle de hub centralisé pour des solutions intégrées, développées sur

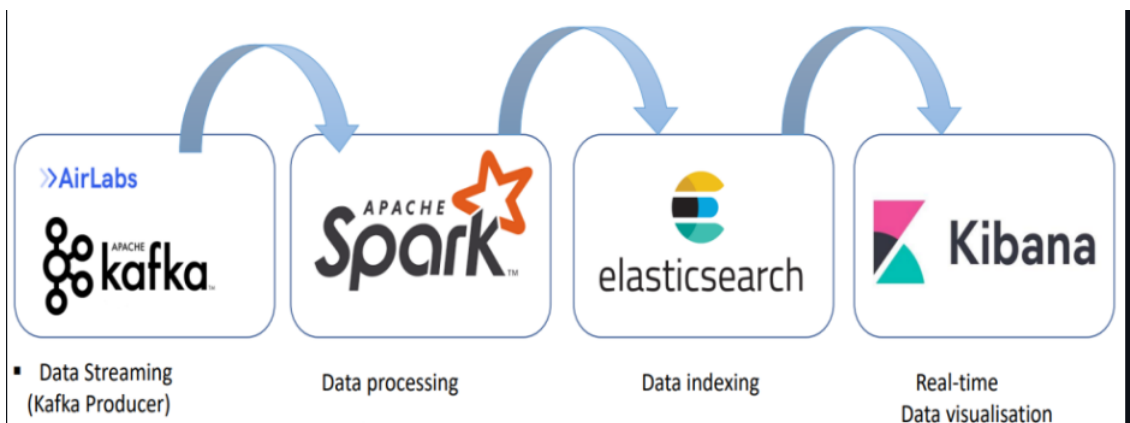
### *Chapitre 3 Configuration d'environnement*

la Suite Elastic. Créée en 2013 au sein de la communauté Elasticsearch, Kibana s'est développée pour offrir une vue à 360° sur la Suite Elastic, devenant ainsi un véritable portail pour les utilisateurs et les entreprises.

# Chapitre 4

## Test d'Application

Dans ce projet, nous utiliserons une API de suivi de vols en temps réel, Apache Kafka, Elasticsearch et Kibana pour créer un pipeline de données d'informations de vol en temps réel et suivre les vols en direct. Nous utiliserons une architecture de haut niveau ainsi que les configurations correspondantes qui nous permettront de créer ce pipeline de données. Le résultat final sera un tableau de bord Kibana récupérant des données en temps réel à partir d'Elasticsearch.



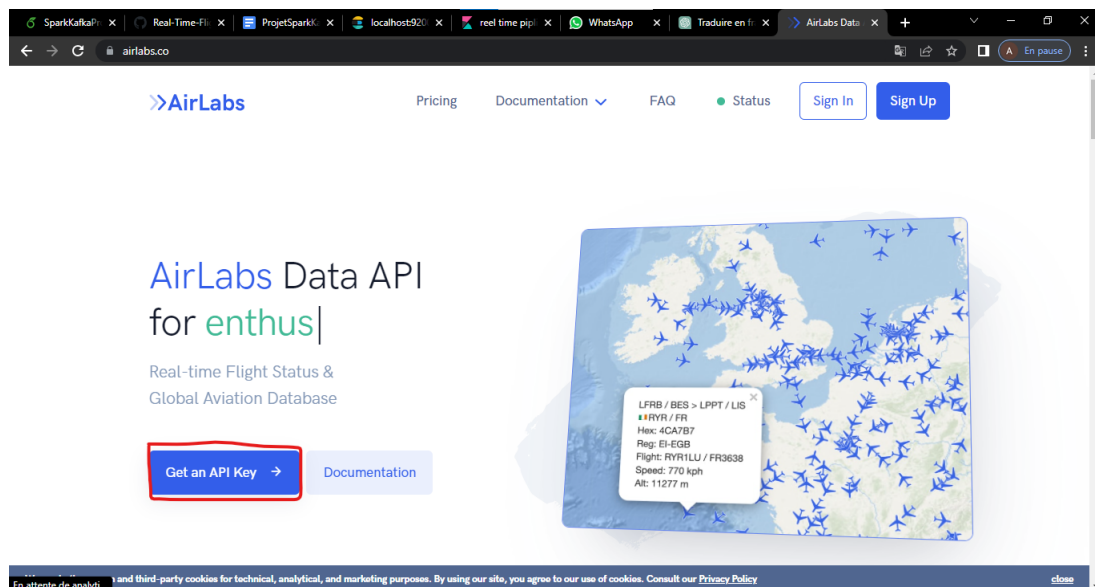
## Chapitre 4 Test d'Application

Nous avons commencé par collecter en temps réel des informations de vol "Flight Tracker API and Data - Aviation database and API"(numéro d'immatriculation de l'aéronef, latitude géographique de l'aéronef, longitude géographique de l'aéronef, élévation de l'aéronef, numéro de vol, etc.) et nous les avons ensuite envoyées à Kafka pour l'analyse.

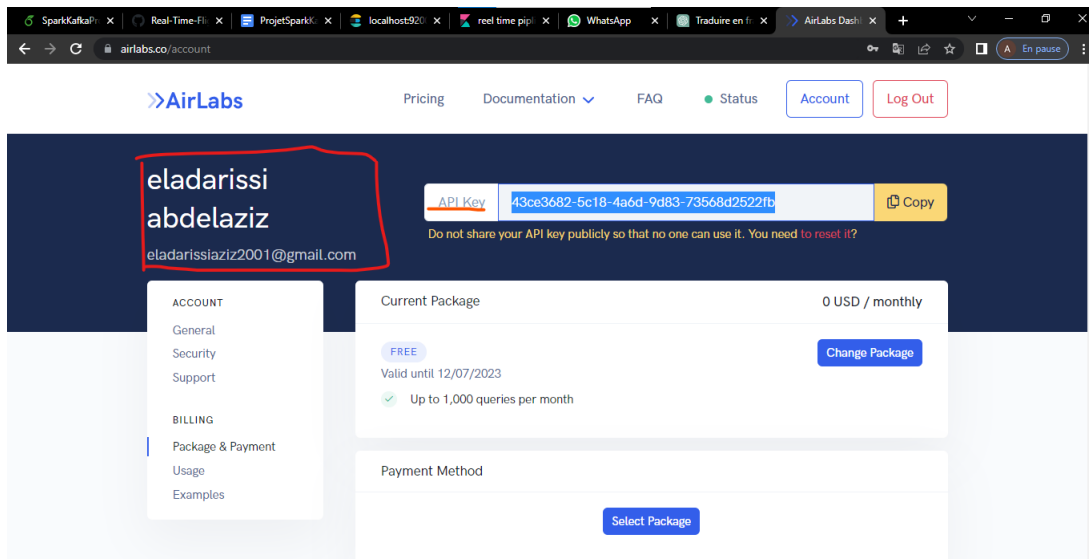
```
> $ curl https://airlabs.co/api/v9/flights

[{"hex": "76CCCD",
  "reg_number": "9V-SFM",
  "flight_iata": "SQ7371",
  "flight_icao": "SIA7371",
  "lat": 21.366613, "lng": 66.61321,
  "alt": 10021, "dir": 121,
  "speed": 966, "v_speed": -0.3,
  "airline_iata": "SQ", "airline_icao": "SIA",
  "dep_iata": "AMS", "dep_icao": "EHAM",
  "arr_iata": "SIN", "arr_icao": "WSSS",
  "flag": "SG", "aircraft_icao": "B744",
  "squawk": "3420", "updated": 1611366621
}]
```

il faut créer une compte dans ce site <https://airlabs.co/> pour avoir le KEY API et le clé secret



## Chapitre 4 Test d'Application



L'étape suivante consiste à créer un topic (test-topic) pour stocker les informations qui seront produites par le producteur.(Programme Java)

```
1 kafka-topics.bat --create --topic test-topic --bootstrap-server localhost:9092 --partitions 2 --replication-factor 2
```

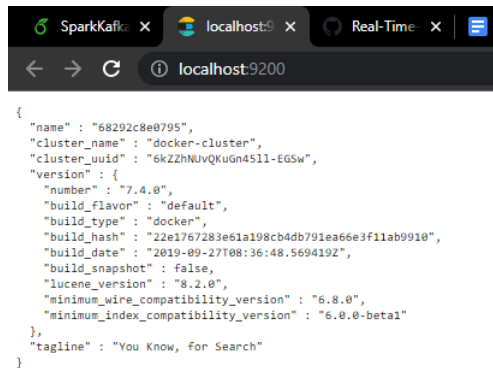
Puis Démarrer les serveurs kafka,zookeeper,Kibana et ElasticSearch

```
C:\Windows\System32\cmd.exe - docker compose up
C:\kafka_2.12-3.5.0>docker compose up
[+] Running 4/0
  Container elasticsearch Created 0.0s
  Container zookeeper Created 0.0s
  Container kafka Created 0.0s
  Container kibana Created 0.0s
Attaching to elasticsearch, kafka, kibana, zookeeper
kafka | [Configuring] 'port' in '/opt/kafka/config/server.properties'
kafka | [Configuring] 'advertised.host.name' in '/opt/kafka/config/server.properties'
kafka | Excluding KAFKA_HOME from broker config
zookeeper | ZooKeeper JMX enabled by default
zookeeper | Using config: /opt/zookeeper-3.4.13/bin/./conf/zoo.cfg
kafka | [Configuring] 'log.dirs' in '/opt/kafka/config/server.properties'
kafka | Excluding KAFKA_VERSION from broker config
kafka | [Configuring] 'zookeeper.connect' in '/opt/kafka/config/server.properties'
kafka | [Configuring] 'broker.id' in '/opt/kafka/config/server.properties'
zookeeper | 2023-07-21 22:10:49,913 [myid:] - INFO [main:QuorumPeerConfig@136] - Reading configuration from: /opt/zookeeper-3.4.13/bin/./conf/zoo.cfg
zookeeper | 2023-07-21 22:10:49,925 [myid:] - INFO [main:DatadirCleanupManager@78] - autopurge.snapRetainCount set to 3
zookeeper | 2023-07-21 22:10:49,925 [myid:] - INFO [main:DatadirCleanupManager@79] - autopurge.purgeInterval set to 1
zookeeper | 2023-07-21 22:10:49,928 [myid:] - WARN [main:QuorumPeerMain@116] - Either no config or no quorum defined in config, running in standalone mode
zookeeper | 2023-07-21 22:10:49,928 [myid:] - INFO [PurgeTask:DatadirCleanupManager$PurgeTask@138] - Purge task started.
zookeeper | 2023-07-21 22:10:49,956 [myid:] - INFO [PurgeTask:DatadirCleanupManager$PurgeTask@144] - Purge task completed.
zookeeper | 2023-07-21 22:10:49,976 [myid:] - INFO [main:QuorumPeerConfig@136] - Reading configuration from: /opt/zookeeper-3.4.13/bin/./conf/zoo.cfg
zookeeper | 2023-07-21 22:10:49,998 [myid:] - INFO [main:ZooKeeperServerMain@98] - Starting server
zookeeper | 2023-07-21 22:10:50,012 [myid:] - INFO [main:Environment@100] - Server environment:zookeeper.version=3.4.13-2d71af4dbe22557fda74f9a9b4309b15a7487f03,
built on 06/29/2018 04:05 GMT
zookeeper | 2023-07-21 22:10:50,012 [myid:] - INFO [main:Environment@100] - Server environment:host.name=6264158797fd
zookeeper | 2023-07-21 22:10:50,013 [myid:] - INFO [main:Environment@100] - Server environment:java.version=1.7.0_65
zookeeper | 2023-07-21 22:10:50,015 [myid:] - INFO [main:Environment@100] - Server environment:java.vendor=Oracle Corporation
zookeeper | 2023-07-21 22:10:50,015 [myid:] - INFO [main:Environment@100] - Server environment:java.home=/usr/lib/jvm/java-7-openjdk-amd64/jre
zookeeper | 2023-07-21 22:10:50,016 [myid:] - INFO [main:Environment@100] - Server environment:java.class.path=/opt/zookeeper-3.4.13/bin/./build/classes:/opt/zoo
keeper-3.4.13/bin/./build/lib/*:jar:/opt/zookeeper-3.4.13/bin/./lib/slf4j-log4j12-1.7.25.jar:/opt/zookeeper-3.4.13/bin/./lib/slf4j-api-1.7.25.jar:/opt/zookeeper-3.4.
13/bin/./lib/netty-3.10.6.Final.jar:/opt/zookeeper-3.4.13/bin/./lib/log4j-1.2.17.jar:/opt/zookeeper-3.4.13/bin/./lib/jline-0.9.94.jar:/opt/zookeeper-3.4.13/bin/./li
b/audience-annotations-0.5.0.jar:/opt/zookeeper-3.4.13/bin/./zookeeper-3.4.13.jar:/opt/zookeeper-3.4.13/bin/./src/java/lib/*:jar:/opt/zookeeper-3.4.13/bin/./conf:
```

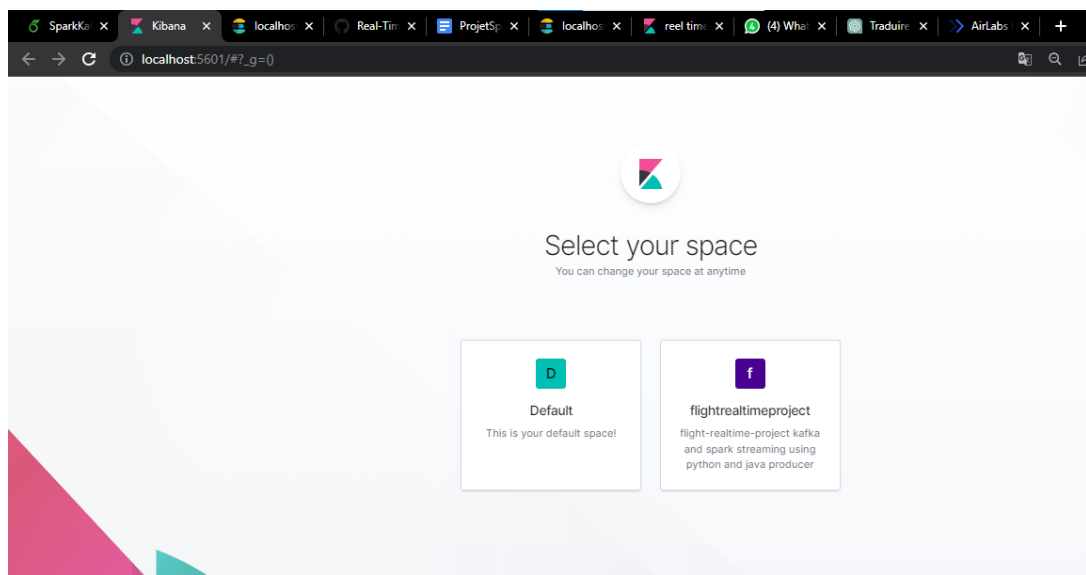


## Chapitre 4 Test d'Application

Pour ElasticSearch Vous pouvez accéder à <http://localhost:9200> pour vérifier s'il est actif et opérationnel. et Kibana <http://localhost:5601>

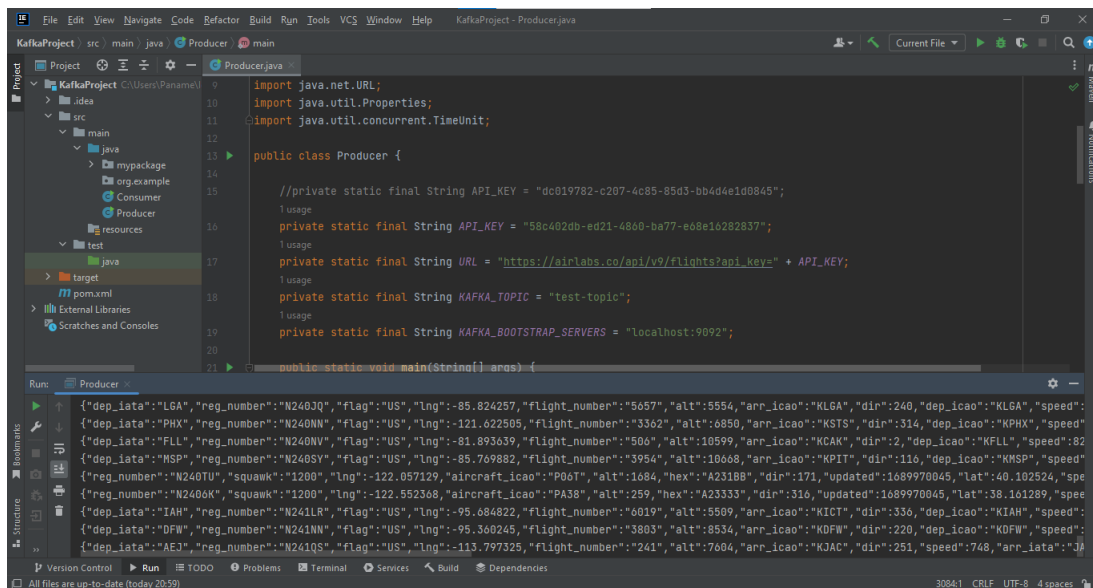


```
{
  "name" : "68292c8e0795",
  "cluster_name" : "docker-cluster",
  "cluster_uuid" : "6kZzhNuvQKuGn4511-EGSw",
  "version" : {
    "number" : "7.4.0",
    "build_flavor" : "default",
    "build_type" : "docker",
    "build_hash" : "22e1767283e61a198cb4db791ea66e3f11ab9910",
    "build_date" : "2019-09-27T08:36:48.569419Z",
    "build_snapshot" : false,
    "lucene_version" : "8.2.0",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```



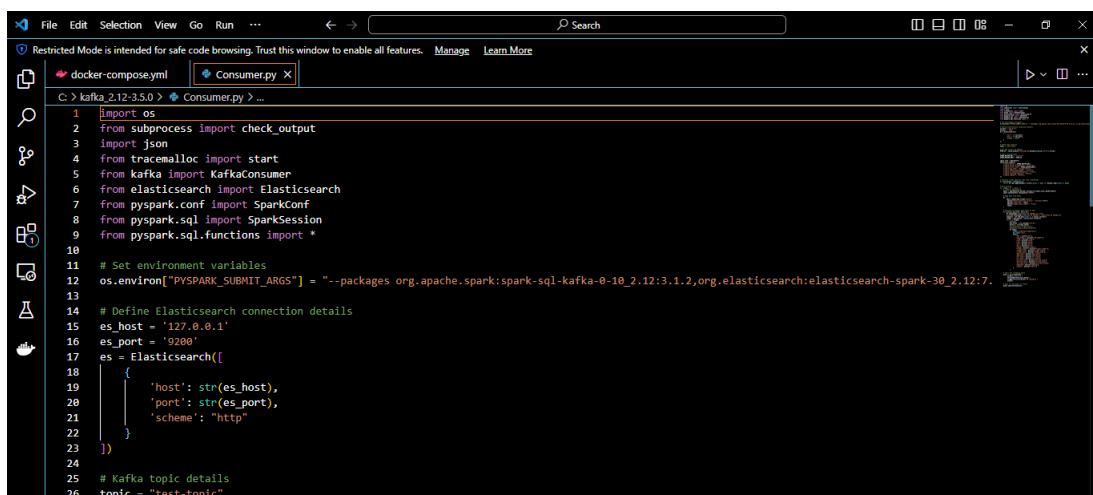
## Chapitre 4 Test d'Application

Les données sont récupérées à partir de l'API de flux de données de vol et envoyées vers un topic Kafka à l'aide de Programme Java(Producer.java) qui Nous avons créer.



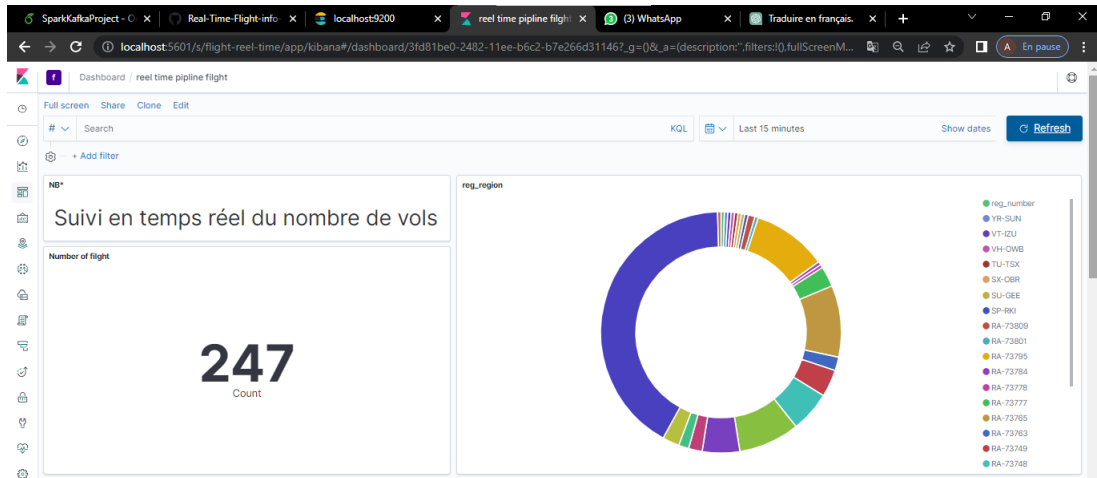
En fin On va démarrer le programme Python (Consumer.py) pour collecter périodiquement des données en temps réel à partir du topic Kafka et les envoyer dans un index Elasticsearch. EN Utilisant la commande suivante

```
1 spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.12:3.1.2,
    org.elasticsearch:elasticsearch-spark-30_2.12:7.14.2 Consumer.py
```



## Chapitre 4 Test d'Application

### Dashboard Final



The dashboard displays a table of flight information. The table has the following columns: reg\_number, hex, \_id, status, arr\_lata, dep\_lata, airline\_lata, alt, dir, lat, lng, and speed. The table contains 247 rows of data, showing flight details such as registration number, hex code, status, arrival and departure locations, airline, altitude, direction, latitude, longitude, and speed.

reg_number	hex	_id	status	arr_lata	dep_lata	airline_lata	alt	dir	lat	lng	speed
RA-09607	142587	wrggkBCOV74RKHUj	en-route	LED	RGK	4G	12192	258	56.545029	46.745495	833
RA-09607	142587	xLgglkBCOV74RKYErj	en-route	LED	RGK	4G	12192	258	56.545029	46.745495	833
RA-73734	152006	xbgfkBCOV74RKHx4	en-route	OMS	SVO	SU	11285	83	56.050106	43.6619	905
RA-73734	152006	xrgLokBCOV74RKYU9	en-route	OMS	SVO	SU	11285	83	56.050106	43.6619	905
RA-73734	152006	yLgMeokBCOV74RKHU9	en-route	OMS	SVO	SU	11285	83	56.050106	43.6619	905
RA-73734	152006	ynlgMeokBCOV74RKHUpe	en-route	OMS	SVO	SU	11285	83	56.050106	43.6619	905
N331DN	A390DE	x7gMeokBCOV74RKHGkx	en-route	ATL	LGA	DL	10332	237	39.710632	-76.563957	740
SU-GEE	010163	w7ggkBCOV74RKHDEpr	en-route	BRU	CAI	MS	11582	334	42.811984	22.852872	772
NS04NN	AC7F41	ybgMeokBCOV74RKHYE13	en-route	DFW	DFW	AA	2819	24	33.167084	-96.97599	546
OE-LNX	440BCE	07gMeokBCOV74RKHUj2	en-route	EMA	DUB	QV	6393	104	53.364487	-6.070025	750
RA-73734	152006	tLgDeokBCOV74RKHUIN	en-route	OMS	SVO	SU	11277	86	56.110249	44.633076	894
RA-73734	152006	z7gMeokBCOV74RKHUu	en-route	OMS	SVO	SU	11269	83	56.096287	44.387512	896
RA-73734	152006	OLgMeokBCOV74RKHUqF	en-route	OMS	SVO	SU	11269	83	56.101265	44.469223	896
RA-73734	152006	0hgMeokBCOV74RKH4E/S	en-route	OMS	SVO	SU	11277	86	56.110249	44.633076	894
RA-73734	152006	zLgMeokBCOV74RKHkzPZ	en-route	OMS	SVO	SU	11269	83	56.096287	44.387512	896
RA-73734	152006	zrghMeokBCOV74RKHFcjB	en-route	OMS	SVO	SU	11269	83	56.096287	44.387512	896
N912WN	ACA01A	y7gMeokBCOV74RKHqzqH	en-route	DEN	BNA	WN	10736	269	38.844955	-102.205383	764

