



Union - Discipline - Travail

Ministère de l'enseignement Supérieur et de la  
Recherche Scientifique

-----

International Data Science Institute



Ministère de l'Economie Numérique  
et de la Poste

-----



# MÉMOIRE DE FIN DE CYCLE

pour l'obtention du

Master en Data Science - Big Data - Intelligence Artificielle

## THEME

Analyse des Réseaux Sociaux pour renforcer la lutte  
contre le Covid-19 - ARSCO

Présenté par :

**FONDIO Kimba**

**Encadreur pédagogique**

MONSAN Vincent

*Maitre de Conférences à l'UFR*

*Mathématiques et Informatique à*

*l'Université Félix Houphouët Boigny*

**Maître de stage**

Fabrice ZAPFACK

*Co-fondateur et CTO à data354*

Année académique 2020-2021



## DEDICACES

*À ma famille, amis et connaissances qui m'ont apporté le soutien moral et financier afin que je puisse réussir cette formation aux métiers de la data science et du big data.*

## REMERCIEMENTS

Nous remercions tout d'abord l'Eternel Dieu pour le souffle de vie et la force nécessaire qu'il nous a accordé pour l'effectivité de ce travail. Nos remerciements vont aussi à l'endroit d' Orange Côte d'Ivoire, la fondation X Polytechnique, l'école X Polytechnique, l'INP-HB de Yamoussoukro et l'ENSEA d'Abidjan qui ont tout mis en œuvre pour la réussite de cette formation.

Ensuite au Directeur Technique (CTO) de data354, **M. Fabrice ZAPFACK**, pour l'accueil et l'assistance qu'il nous a apporté durant ces six mois de stage. Également nous remercions les membres de l'équipe de data354, pour leur sympathie, disponibilité, conseils et bonne humeur, qui ont contribué à faciliter notre intégration.

Un remerciement particulier à l'endroit de **M. Bio Mikaila TOKO WOROU** qui nous apportera son expertise tout au long de ce stage.

Nous témoignons aussi notre profonde gratitude à l'administration de l'INP-HB, en particulier à Monsieur **KOFFI N'Guéssan**, ancien Directeur de l'INP-HB, à Monsieur **DIABY Moussa**, Directeur Général actuel de l'INP-HB, à Monsieur **TANOH Tanoh Lambert**, Directeur de l'IDSI, et à ses collaborateurs.

Nos remerciements les plus distingués vont également à l'endroit du Professeur **MONSAN Vincent** pour l'encadrement durant toute la période de stage.

Mes sincères remerciements à ma tante **DOUMBIA Seingo**, à ma mère **DOUMBIA Mariam**, à mes oncles **DOUMBIA Bouaké**, **DOUMBIA Baba Gaoussou** et **DOUMBIA Siaka Laciné**, à tous mes frères et soeurs, à mes amis **DIARRASOUBA Daouda**, **DIARRASSOUBA Brahima**, **BAMBA Daouda**, **KONE Wardjouma Adama**, à tous mes condisciples de la **3ème promotion de l'IDSI** qui m'ont accompagné moralement et financièrement tout au long de cette formation, sans oublier tout ceux ou celles qui ont pris de leur temps pour lire et corriger ce mémoire.

---

## SOMMAIRE

REMERCIEMENTS	4
SOMMAIRE	5
LISTE DES FIGURES	6
LISTE DES ABREVIATIONS	7
GLOSSAIRE	8
AVANT-PROPOS	9
RESUME	10
INTRODUCTION GENERALE	11
PARTIE I : ENVIRONNEMENT DE TRAVAIL	12
I - PRÉSENTATION DE LA STRUCTURE D'ACCUEIL	13
II - PRESENTATION DU PROJET	15
PARTIE II: PRÉSENTATION DES CONCEPTS ET TYPE D'ARCHITECTURE BIG DATA	18
I- PRESENTATION DES CONCEPTS	19
II- TYPE D'ARCHITECTURE BIG DATA	37
PARTIE III: ARCHITECTURE BIG DATA DE LA PLATEFORME D'ANALYSE DES RÉSEAUX SOCIAUX	41
I- DESCRIPTION DÉTAILLÉE DE L'ARCHITECTURE DE LA SOLUTION	42
II- ETUDE COMPARATIVE, CHOIX DES OUTILS TECHNIQUES ET DESCRIPTION DE L'APPROCHE	46
PARTIE IV: MISE EN PLACE ET DÉPLOIEMENT DE LA SOLUTION BIG DATA/CLOUD	64
I- PRÉPARATION DE L'ENVIRONNEMENT	65
II- DÉPLOIEMENT DES SERVICES DANS AZURE	69
CONCLUSION GENERALE	78
BIBLIOGRAPHIE & WEBOGRAPHIE	79
TABLE DES MATIERES	80

## LISTE DES FIGURES

Figure 1: Schéma du processus ETL	24
Figure 3 : Différence entre conteneurisation (conteneur) et virtualisation (machine virtuelle)	28
Figure 4 : Cycle de développement avec la méthodologie DevOps	31
Figure 5: Les différents modèles de services cloud	34
Figure 6 : Services Cloud: AWS vs Azure vs GCP	37
Figure 7: Schéma de l'architecture lambda	39
Figure 8: Schéma de l'architecture Kappa	40
Figure 9: Schéma de l'architecture fonctionnelle de la plateforme d'analyse de données	42
Figure 10 : Schéma de l'architecture de l'application de collecte de données	49
Figure 11: Schéma du système d'ingestion de données	51
Figure 12: Schéma de l'outil d'ETL (Apache Spark)	54
Figure 13: Schéma du fonctionnement de Spark dans sur le cluster Kubernetes (AKS)	56
Figure 14: Schéma du système d'orchestration dans Data Factory	60
Figure 15 : Schéma du synapse de data warehouse avec synapse et ADLS	62
Figure 16: Schéma final de l'architecture de la plateforme Big data	63
Figure 17: Interface de l'application de collecte de données dans App Services	72
Figure 19 : Interface de contrôle du Data Lake (ADLS)	73
Figure 20: Interface de contrôle du cluster Kubernetes (AKS)	74
Figure 21 : Interface de Azure Container Registry présentant les images Docker.	74
Figure 22 : Interface de Azure Function App	75
Figure 23: Schéma de la pipeline de traitement des données dans Data Factory	75
Figure 24: Interface de monitoring de Data Factory	75
Figure 25: Interface de Azure Synapse Analytics (Synapse Studio)	76
Figure 26: Interface de visualisation avec Power BI	77

---

## LISTE DES ABREVIATIONS

**ACR:** Azure Container Registry

**AD:** Active Directory

**ADF:** Azure Data Factory

**ADLS:** Azure Data Lake Storage

**AKS:** Azure Kubernetes Services

**ARM:** Azure Resources manager

**AWS :** Amazon Web Services

**CAPEX:** Capital Expenditure

**CEO :** Chief Executive Officer

**CI/CD:** Continuous Integration/ Continuous Delivery

**CLI:** Commande-Line Interface

**CTO :** Chief Technical Officer

**ENSEA :** École Nationale Supérieure de Statistique et d'Economie Appliquée

**ETL :** Extract Transform Load

**GCP :** Google Cloud Platform

**HTTP:** Hyper Text Markup language

**IaaS:** Infrastructure as a Service

**IDSi :** International Data Science Institute

**INP-HB :** Institut National Polytechnique Félix Houphouët Boigny

**OPEX:** Operating Expense

**PaaS:** Platform as a Service

**RGPD:** Règlement Général sur la Protection des Données

**SaaS:** Software as a Service

**SQL:** Structured Query Language

**SSIS:** Microsoft SQL Server Integration Services

**VM(s) :** Virtual Machines (Machine virtuelles)

---

## GLOSSAIRE

- **Batch:** Il désigne les traitements effectués par lots, c'est-à-dire de manière périodique.
- **Cron:** cron (Command Run On) est un système de planification propre aux systèmes UNIX. C'est une expression de cinq ou six champs séparés par des espaces représentant des périodes de temps.
- **CXP:** c'est un cabinet indépendant internationalement reconnu de conseil et d'analyse de l'écosystème du numérique.
- **Intelligence artificielle :** elle désigne l'ensemble des théories et des techniques mises en œuvre en vue de réaliser des machines capables de simuler l'intelligence humaine.
- **Graph API :** C'est l'API officiel de Facebook mise à la disposition des utilisateurs.
- **Pods :** Un pod est la plus petite unité d'exécution de Kubernetes. Les pods sont éphémères par nature, si un pod (ou le nœud sur lequel il s'exécute) échoue, Kubernetes peut automatiquement créer une nouvelle réplique de ce pod pour continuer les opérations. Les pods incluent un ou plusieurs conteneurs (tels que les conteneurs Docker).
- **Serverless:** Traduit par architecture sans serveur, serverless est un modèle de développement du cloud computing qui permet au développeur de créer et exécuter des applications sans avoir à gérer de serveurs. Cependant, il existe toujours des serveurs mais ils sont distants et ne sont pas à la charge du développeur.
- **Spark Operator:** Spark Operator est un contrôleur personnalisé Kubernetes qui utilise des ressources personnalisées pour la spécification déclarative des applications Spark
- **Streaming:** le streaming dans le contexte du big data, désigne les traitements effectués au fil de l'eau : en temps réel.
- **Tweepy:** librairie python qui permet d'interagir avec l'API officielle de Twitter.



## AVANT-PROPOS

L'**International Data Science Institute** est une chaire internationale de formation en Data Science et Big Data, issu d'un partenariat entre l'opérateur de téléphonie Orange, l'Institut National Polytechnique Félix Houphouët-Boigny (INP-HB), l'École Nationale Supérieure de Statistique et d'Economie Appliquée (ENSEA), l'École polytechnique (X) et la Fondation de l'École polytechnique (FX).

Homologuée par l'État de Côte d'Ivoire, la chaire a ouvert ses portes à la toute première promotion en 2017 et elle a pour objectif de former des experts dans le domaine de la statistique, de l'intelligence artificielle et du Big Data.

D'une durée de deux ans, l'IDSI propose un master d'excellence de niveau international à destination d'étudiants qui s'approprient des connaissances en ingénierie informatique basée sur les nouvelles évolutions en matière de stockage et de traitement des données. Les cours sont dispensés au sein du Data Science Institute de l'INP-HB par une équipe pédagogique composée de professeurs de l'INP-HB, de l'ENSEA, de l'X, ainsi que des professionnels experts dans leur domaine.

Pour parfaire la formation et se rapprocher du monde professionnel, il est prévu un projet de fin d'études sanctionné par la rédaction d'un mémoire puis une soutenance devant un jury d'enseignants de l'INP-HB de l'ENSEA de l'X et de responsables d'entreprises.

C'est dans ce cadre que l'IDSI initie en fin de cycle, des stages pratiques en entreprise en vue d'amener ses élèves à confronter leurs connaissances théoriques acquises durant leur parcours académique aux réalités du monde professionnel. C'est ainsi que nous avons été accueilli par l'entreprise data354 du 13 avril 2021 au 15 octobre 2021 pour un stage de 6 mois dans ses locaux.

---

## RESUME

Les réseaux sociaux fournissent une très grande quantité de données. L'analyse de cette masse de données peut permettre aux organisations et entreprises de mieux gérer la relation avec les utilisateurs. Une des applications concrètes de ces analyses peut aider le gouvernement à évaluer les impacts des mesures et politiques qu'il a mises en place dans le cadre de la lutte contre le COVID-19. De ce fait, nous nous sommes intéressés à ce domaine afin de concevoir un outil d'analyse mieux adapté au contexte africain pour aider les acteurs publics et privés à prendre des décisions éclairées.

Le travail exposé dans ce mémoire est un projet en partenariat avec l'INP-HB qui vise à mettre en place **un outil d'analyse des réseaux sociaux pour renforcer la lutte contre le COVID-19 (Projet ARSCO)**.

Nous exposons les techniques et outils utilisés pour atteindre nos objectifs. Pour se faire, nous procédons tout d'abord à la présentation de l'architecture fonctionnelle de la plateforme d'analyse, par la suite nous présentons de manière détaillée le fonctionnement de l'outil d'analyse. Nous terminons par l'implémentation de la solution.

Dans l'intention de minimiser les coûts de déploiement de la plateforme, nous avons été amenés à mettre en place toute l'architecture dans le Cloud, plus précisément Microsoft Azure avec des approches originales.

### Mots clés :

Réseaux sociaux; Covid-19; Big data; Cloud Computing; Microsoft Azure; Intelligence artificielle; Conteneurisation; Architecture .

## INTRODUCTION GENERALE

L'analyse des réseaux sociaux consiste à analyser les conversations et tendances qui se produisent autour d'une marque et de son secteur d'activité. Les informations recueillies permettent de prendre des décisions plus éclairées car cette analyse permet non seulement de recueillir l'opinion des utilisateurs mais aussi de comprendre les raisons qui motivent ceux-ci. L'analyse des réseaux sociaux est idéale pour mesurer les impacts des campagnes marketing, des politiques publiques et pour recueillir le ressenti des utilisateurs en toute liberté. Les outils d'analyse des réseaux permettent de centraliser les messages et commentaires provenant de toutes les plateformes en intégrant de nombreuses analyses. Sur le marché, de nombreux outils d'analyse des réseaux sociaux existent. Cependant, les analyses proposées par ces outils ne sont pas très adaptées au contexte africain car la manière dont s'exprime un français n'est pas identique à celle d'un ivoirien. De plus, les coûts de ces outils sont parfois très élevés.

Aujourd'hui, avec la crise sanitaire qui sévit dans le monde, l'analyse des réseaux sociaux peut permettre de connaître l'opinion publique sur les mesures et politiques mises en place par les autorités publiques. Elle peut donc contribuer au renforcement de la lutte contre le COVID-19 en anticipant ses effets potentiels.

Dans sa volonté de contribuer au développement de l'Afrique, **data354** souhaiterait mettre en place un outil d'analyse des réseaux sociaux adaptés aux contextes des organisations et entreprises en Afrique. Dans le cadre de cette ambition, l'entreprise a décidé de démarrer par une application plus sociale. C'est dans ce cadre que le projet suivant : **"Analyse des Réseaux Sociaux pour renforcer la lutte contre le COVID19 (ARSCO)"** a été initié. Nous nous sommes vu confier ce projet lors de notre stage dans ses locaux.

Ceci étant, comment mettre en place un système d'écoute temps réel des réseaux sociaux plus adapté au contexte ivoirien? Quelles architectures Big data utilisées pour implémenter une telle solution à moindre coût? Pour répondre à ces questions, nous commencerons, après avoir présenté notre structure d'accueil ainsi que le projet, à présenter les différents concepts que nous aborderons et les types d'architecture Big data. Par la suite, nous allons décrire de manière détaillée notre approche et enfin nous procéderons à la mise en place et le déploiement de cette solution.

## PARTIE I : ENVIRONNEMENT DE TRAVAIL

---

*Nous présenterons dans cette partie, la structure d'accueil ainsi que le cahier de charges du projet et les objectifs à atteindre ainsi que le planning à observer.*

## I - PRÉSENTATION DE LA STRUCTURE D'ACCUEIL

### 1- Structure d'accueil

[data354](#) est une société de consulting technique sur les produits et services d'ingénierie et d'analyse des données. La société accompagne aujourd'hui les acteurs du secteur public et privé à la maîtrise des données et au développement de solutions d'analyse prédictive. Fondée par une équipe expérimentée, pluridisciplinaire et complémentaire, tous animés d'une vision commune: celle de mettre en place des outils analytiques et prédictifs de pointe au service des organisations évoluant en Afrique; elle accompagne aujourd'hui ses clients tout au long de leur transformation data grâce à son **expertise Technologique** et à sa capacité de **conseil en Stratégie et Management**.

### 2- Domaines de compétences

Partenaire privilégié des cabinets de conseil en transformation digitale en Afrique et en Europe, **data354** propose une large gamme de prestations externalisées tels que :

- Cartographie de l'infrastructure technique et des données existantes;
- Collecte, stockage, sécurisation, traitement, visualisation et valorisation de données internes et externes;
- Mise en place d'outils d'aide à la décision;
- Intelligence artificielle : machine learning / deep learning (entraînement des modèles, prédictions et visualisation).

Elle accompagne également les entreprises technologiques internationales à l'adaptation de leurs solutions aux marchés africains.

Les services proposés par l'entreprise sont regroupés en trois grands groupes de services à savoir :

Data Strategy	Data Management	Data Intelligence
<ul style="list-style-type: none"><li>• Évaluation maturité Data</li></ul>		<ul style="list-style-type: none"><li>• Data Science</li></ul>

<ul style="list-style-type: none"> <li>● Stratégie &amp; plan gouvernance Data &amp; IA</li> <li>● Acculturation Data</li> </ul>	<ul style="list-style-type: none"> <li>● Infrastructure Big Data et temps-réel</li> <li>● Qualité des données</li> <li>● Data Catalog<sup>1</sup></li> </ul>	<ul style="list-style-type: none"> <li>● Business Intelligence</li> <li>● RPA<sup>2</sup></li> </ul>
--	--	--

### 3- Equipe

Avec un mode de gestion agile, **data354** est composée de plusieurs équipes qui collaborent entre elles afin de satisfaire la clientèle.

- **Equipe dirigeante**

Il s'agit de l'équipe fondatrice, composée du **Chief Executive Officer (CEO)** et du **Chief Technology Officer (CTO)**. C'est une équipe très expérimentée désireuse de mettre leur savoir à la disposition des entreprises africaines.

- **Equipe de Management**

Cette équipe est chargée de la gestion administrative de l'entreprise.

- **Équipe de Marketing et vente**

Composée essentiellement de Consultants et Commerciaux très qualifiés, cette équipe est responsable de la stratégie de l'entreprise. Elle est directement rattachée au CEO.

- **Equipe technique**

Dirigée par le CTO, cette équipe pluridisciplinaire est responsable de la réalisation des missions confiées à l'entreprise. Elle se compose de Data Scientists, Data Engineers, MLOps, Data Architects, Data Analysts, spécialistes BI et de développeurs.

<sup>1</sup> **Data catalog** : un Data Catalog est un catalogue de données. Plus précisément, il s'agit d'un emplacement centralisé où sont regroupées les informations sur les données contenues dans une base de données : les métadonnées.

<sup>2</sup> **RPA** : La RPA, ou Robotic Process Automation, est une technologie permettant d'automatiser des processus métier qui nécessitent, jusque-là, l'intervention d'un humain. c'est -à-dire traiter des tâches répétitives et chronophages grâce à l'utilisation de logiciels d'intelligence artificielle capables d'imiter un travailleur humain.

---

## II - PRESENTATION DU PROJET

### 1- Contexte du projet

En Mars 2020, la Côte d'Ivoire enregistrait son premier cas de COVID-19. Depuis lors, le nombre de cas n'a cessé d'évoluer plongeant tout le pays dans une grave crise sanitaire. Cette crise a provoqué de nombreux problèmes aussi bien au plan social (décès) qu'au plan économique. D'après **le rapport de la banque mondiale**, cette crise pourrait avoir des **effets néfastes** sur le pays tout entier avec une **baisse de la croissance à 1,8% au lieu de 7%**, **des milliards de francs en pertes de production**, **plus de 70% des ménages en incapacité de faire face à leur dépenses**, **plus d'un million de personnes qui basculeraient dans la pauvreté**, **des milliers d'emplois menacés dans les secteurs formels et informels**, **des milliers d'enfants non scolarisés** et une **baisse des dépenses de santé publiques et privées en 2020**, etc.

Cette situation de crise **nécessite des actions urgentes afin de contrer** ses effets néfastes potentiels. Ainsi, le gouvernement tente de freiner cette crise à travers de nombreuses actions. Cependant, il peine à mettre en place des politiques adaptées à cette crise. Le gouvernement rencontre des **difficultés à trouver les bonnes mesures** et aussi à **évaluer les impacts de ces mesures**. De plus, cette situation renforce la **défiance des populations** envers les pouvoirs publics à cause de l'inaction ou la mauvaise gestion de la communication de ceux-ci.

De ce fait, **data354**, dans l'optique d'apporter sa contribution dans la lutte contre le COVID-19, en partenariat avec **l'Institut National Polytechnique - Houphouët Boigny**, souhaite fournir aux acteurs publics un outil d'analyse et de surveillance des réseaux sociaux afin de **renforcer la lutte contre le COVID-19 en Côte d'Ivoire**.

### 2- Objectif général

L'objectif de ce projet est de fournir aux acteurs publics **un outil d'analyse et de surveillance des réseaux sociaux afin de renforcer la lutte contre la Covid-19 en Côte d'Ivoire à l'aide des technologies du big data, du cloud computing, et d'algorithmes d'intelligence artificielle**.

### 3- Objectifs spécifiques

Pour atteindre l'objectif général, nous identifions trois objectifs spécifiques à ce projet à savoir:

- Collecter et stocker les données des réseaux sociaux à propos du Covid19 à l'aide d'outils big data et cloud computing;
- Nettoyer, Analyser et traiter les données recueillies afin de ressortir les informations pertinentes grâce aux algorithmes d'intelligence artificielle;
- Développer une interface utilisateur pour faciliter l'interaction avec la solution.

### 4- Planning d'exécution

La clé principale de la réussite d'un projet est un bon planning. En effet, le planning aide à bien subdiviser le travail et séparer les tâches à réaliser, il offre une meilleure estimation et gestion de temps nécessaire pour chaque tâche. De plus, il donne assez de visibilité permettant d'estimer approximativement la date d'achèvement de chaque tâche. Dans notre projet, le planning prévisionnel se présente comme suit.

Durée	Tâches	Livrables	Outils
1 semaine	Setup environnement de travail		Linux, AWS, GCP, Azure
5 semaines	Collecte et ingestion des données des réseaux sociaux	Tweets et commentaires stockées dans un Data Lake	Azure App Service,s, Event Hub, ADLS
10 semaines	Traitement des données	Données enrichies stockées dans un Data Warehouse	Python, Rest API, NLP Spark, Kubernetes
4 semaines	Manipulation et visualisation	Dashboard, APIs	Rest APIs, Power BI
4 semaines	Finalisation du projet	Documentation détaillée Rapport de stage	Google Docs, Slides



---

## 5- Equipe Projet

### a- Équipe de pilotage

Composée d'experts de l'INPHB et de data354, cette équipe est responsable de la planification et du pilotage de tout le projet.

### b- Équipe de recherche

Il s'agit d'enseignants chercheurs dans les domaines de la linguistique, sociologie, statistique et économie. Ces chercheurs participent à l'analyse socio-économique et l'exploitation des données recueillies. Ils sont aussi responsables de l'interprétation des résultats obtenus en les confrontant avec les réalités socio-économiques.

### c- Équipe de développement

Formée, de Data Scientists, Data Engineers, Data Analysts et spécialistes DevOps, cette équipe est chargée du développement de l'outil d'analyse avec les dernières technologies sur le marché.

*Notre contribution dans le cadre de ce projet, a été la mise en place de toute l'infrastructure de la chaîne de traitement (Data Engineering, DevOps).*

***Le contenu de ce document ne prend en compte que cette contribution.***

## PARTIE II: PRÉSENTATION DES CONCEPTS ET TYPE D'ARCHITECTURE BIG DATA

---

*Notre projet, pour sa mise en place fait appel à un certain nombre de concepts: le Big data, les traitements ETL, les clusters de serveurs, la conteneurisation, le DevOps et le Cloud Computing. Nous allons d'abord présenter ces concepts, ensuite nous présenterons les principaux types d'architectures Big data.*

---

## I- PRESENTATION DES CONCEPTS

### 1- Le Big Data

#### a - Définition

Selon la définition du **CXP** , Le Big data est un ensemble de méthodes et technologies utilisées pour charger, stocker et analyser des volumes importants de données poly structurées de manière évolutive.

Le Big data désigne donc une collection de données d'un volume énorme qui croît de manière exponentielle avec le temps. La taille et la complexité de ces données sont tellement grandes qu'elles ne peuvent être stockées ou traitées efficacement par les outils de gestion de données classiques. Les données sont analysées grâce aux **techniques de data science** et peuvent être structurées, semi-structurées ou non structurées. Elles sont caractérisées par les trois V ou même les cinq V.

#### b- les caractéristiques du big data

Les données du big data se caractérisent par les trois **V**:

- **Le Volume**

il s'agit de la très grande taille des données du big data;

- **La Variété**

les données sont de sources et natures hétérogènes;

- **La Vitesse**

Ce terme fait référence à la vitesse à laquelle les données sont générées et doivent être traitées.

D'autres caractéristiques sont parfois utilisées. À savoir:

- **La Vérité**

Elle désigne le fait de pouvoir tirer des informations vraies et fiables dans cette grande masse de données.

- **La Valeur**

Il s'agit de la capacité à faire ressortir de la valeur dans cette infobésité .

### **c- Intérêts du big data**

L'immense masse de données qu'est le big data peut impacter fortement les activités de l'entreprise lorsqu'il est bien exploité. L'utilisation du big data permet aux entreprises de:

- **Éclairer et guider leur prise de décision;**
- **Accroître les performances des système opérationnels;**
- **Améliorer la réactivité à l'égard des utilisateurs.**

### **d- Enjeux du big data**

Considéré comme le nouveau pétrole, l'exploitation des données du big data est l'un des plus grands défis de ce siècle. Les enjeux du big data touchent plusieurs secteurs.

- **Enjeux techniques**

L'utilisation des données du big data est un véritable challenge technologique. La volumétrie, la complexité et la vélocité de ces données dépassent les outils de stockage et de traitement classiques. Les entreprises doivent donc mettre en place des data centers plus adaptés, avec des outils capables de traiter ces données. Les données reçues étant de formats divers, un contrôle de qualité est nécessaire afin de ressortir des informations pertinentes.

- **Enjeux Economiques**

Les données du big data sont stockées et traitées grâce à d'immenses Data centers qui sont généralement contrôlés par les plus grands acteurs économiques. Cette demande croissante de puissance de calcul et de capacité de stockage a favorisé la

démocratisation du cloud computing qui génère aujourd'hui des milliards de dollars par an (Synergy Research Group). Le marché du big data se chiffre aujourd'hui à des centaines de milliards de dollars.

- **Enjeux juridiques**

Les données big data sont dans la grande majorité des données à caractère personnel provenant des comptes des utilisateurs. Ainsi la question de la protection de ces données devient une priorité. Les entreprises sont donc contraintes de respecter un ensemble de règles établies par des organismes spécialisés (ARTCI pour la Côte d'Ivoire) afin d'assurer la protection de ces données.

***Dans le cadre de ce projet, les données, étant recueillies sur les réseaux sociaux et en temps réel (vélocité), elles ne sont donc pas très structurées (variété) et sont assez volumineuses (volumétrie). Leur traitement et leur stockage nécessitent des outils du big data. Les techniques du big data nous permettront de mettre en place cette solution.***

## **2- Traitements ETL (extract transform load)**

### **a -Définition**

ETL est un processus d'intégration de données faisant référence à trois étapes: Extraction, Transformation et Chargement, utilisé pour traiter les données provenant de plusieurs sources.

### **b- Fonctionnement des processus ETL et ELT**

Durant, le processus ETL, les données sont d'abord extraites depuis des systèmes sources avant d'être converties dans un format exploitable: c'est la transformation. Enfin les données transformées sont chargées dans un data warehouse.

l'ELT (Extract, Load, Transform) est une approche alternative mais convexe conçue pour pousser les données vers des Data lake avant de les traiter. Cette approche améliore les performances.

### c- Types d'outils ETL

Les outils ETL sont utilisés depuis plus de 30 ans. Au cours de leurs évolutions, de nombreux outils ont fait leur apparition sur le marché. De nombreux outils ETL sont offerts par les concepteurs de logiciels comme IBM, Oracle, Informatica, SAP, etc... Plus récemment, des outils ETL open source et des ETL du cloud ont commencé à émerger.

- **Logiciels ETL propriétaires(d'entreprise)**

De nombreuses entreprises conçoivent et supportent des outils ETL qui sont très appréciés des entreprises. Ce sont entre autres:

- **Informatica Powercenter;**
- **IBM InfoSphere DataStage;**
- **Oracle Data integrator (ODI);**
- **Microsoft SQL Server Integration Services (SSIS);**
- **SAP Data Services.**

Malgré les belles performances de ces outils, leurs coûts de licence et de maintenance poussent certaines entreprises vers les outils ETL open source.

- **ETL Open source**

Ces outils ETL sont de belles alternatives aux logiciels payants car ils sont gratuits et parfois plus performants. Les leaders dans cette catégorie sont :

- **Talend Open Studio;**
- **Pentaho Data Integration (PDI);**
- **Hadoop/Spark.**

Ces outils open source aident les entreprises à réaliser leur processus ETL. Cependant certaines, ayant plus d'expérience préfèrent réaliser des outils ETL personnalisés qui répondent parfaitement à leurs besoins.

---

- **Outils ETL personnalisés**

A la recherche de flexibilité, les entreprises utilisent les langages de programmation pour concevoir leur propre ETL. Cette approche nécessite une maintenance régulière et une documentation solide afin de rester compétitive sur le marché. Ces outils sont faits pour répondre à des besoins spécifiques donc ont du mal à servir d'autres personnes en dehors de l'entreprise. Les langages les plus utilisés à cet effet sont:

- **SQL;**
- **Python;**
- **Java;**
- **Scala;**
- **Spark & Hadoop MapReduce.**

- **Services Cloud ETL**

Avec l'émergence du cloud computing, certains fournisseurs conçoivent leurs propres services ETL qu'ils mettent à disposition de leurs usagers. Cette approche offre une grande flexibilité et permet aux entreprises de faire des économies sans se soucier de la maintenance. Les plus célèbres sont:

- **Elastic MapReduce (EMR) d' Amazon Web Services (AWS);**
- **AWS Glue;**
- **AWS Data Pipeline;**
- **Azure Data Factory de Microsoft Azure;**
- **Google Cloud Dataflow de Google Cloud Platform.**

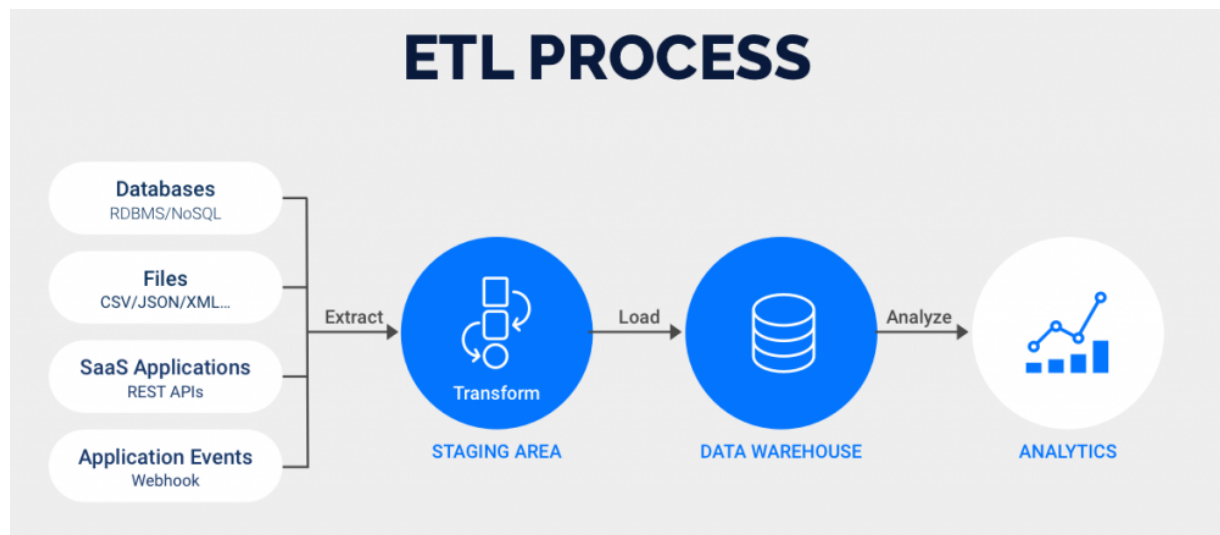


Figure 1: Schéma du processus ETL

**Le concept d'ETL est très important dans le cadre de notre solution car il nous permettra de mettre en place toute la chaîne de traitement nécessaire pour valoriser les données collectées.**

### 3- Cluster de serveurs

#### a- Définition

Un cluster de serveurs ou grappe de serveurs désigne un système informatique composé de plusieurs serveurs indépendants (connectés en réseau) qui fonctionnent comme une seule machine. Chaque serveur est un nœud du cluster et dispose de ses propres disques, mémoire, processeurs, etc. Dans la plupart des configurations des clusters, Les différents nœuds sont connectés à un nœud central à travers le réseau appelé nœud maître. Les autres noeuds sont appelés noeuds esclaves.

#### b- Avantage d'un cluster

L'utilisation d'un cluster est plus fiable, scalable et efficace qu'une seule machine. Les clusters sont faits principalement pour exécuter les applications qui doivent



tourner pendant de très longues périodes et qui nécessitent une très grande puissance de calcul et de capacité de stockage parce que le cluster mutualise les ressources des nœuds la composant.

Ainsi les principaux avantages d'un cluster sont :

- **La scalabilité**

Elle permet au cluster de s'adapter à la charge de travail. Lorsqu'on a une petite charge de travail, certains nœuds esclaves peuvent être retirés du cluster sans que cela ne cause un problème au système. De même lorsque la charge de travail augmente, de nouveaux nœuds peuvent être ajoutés afin de supporter cette charge.

- **La tolérances aux pannes**

La tolérance au pannes désigne le fait qu'un cluster soit capable de fonctionner avec des problème tels que:

- Les défaillances du logiciel d'application et les défaillances des services associées;
- Les défaillances dans les systèmes matériels tels que les processeurs, la mémoire, les alimentations, etc;
- Les défaillances du site Web causées par des revers naturels, des pannes de courant, etc.

- **Haute disponibilité**

Elle réside dans le fait que les nœuds du clusters sont **répliqués : la redondance**. Ainsi, un cluster peut continuer à fonctionner même si l'un des serveurs principaux tombe en panne, car la charge de travail du serveur défaillant est automatiquement transférée sur le serveur secondaire du cluster.

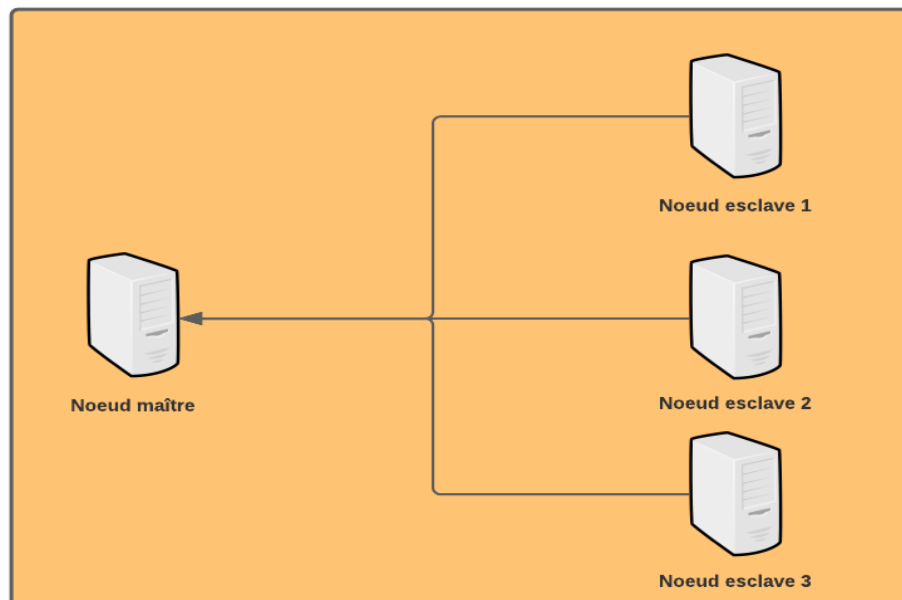


Figure 2 : Schéma d'un cluster de serveurs

*Notre solution a besoin d'une forte scalabilité car les volumes de données reçues sont assez importants et variables. De plus, pour traiter ces données, il est nécessaire d'avoir une grande puissance de calcul, une grande capacité de stockage et une forte tolérance aux pannes. D'où le choix de se tourner vers les clusters de serveurs.*

## 4- Conteneurisation

### a- Définition

Selon Red Hat, La conteneurisation consiste à rassembler le code du logiciel et tous ses composants (bibliothèques, frameworks et autres dépendances) de manière à les isoler dans leur propre « **conteneur** ». En d'autres termes c'est le fait de faire fonctionner une application indépendamment du système d'exploitation. On embarque l'application et toutes ses dépendances dans un mini système d'exploitation (OS) appelé conteneur.

Cette isolation de l'application facilite son déplacement entre les plateformes et infrastructures car l'application dispose de tout ce dont elle a besoin dans le conteneur. Le système de conteneurisation le plus célèbre aujourd'hui est Docker avec Docker Engine.

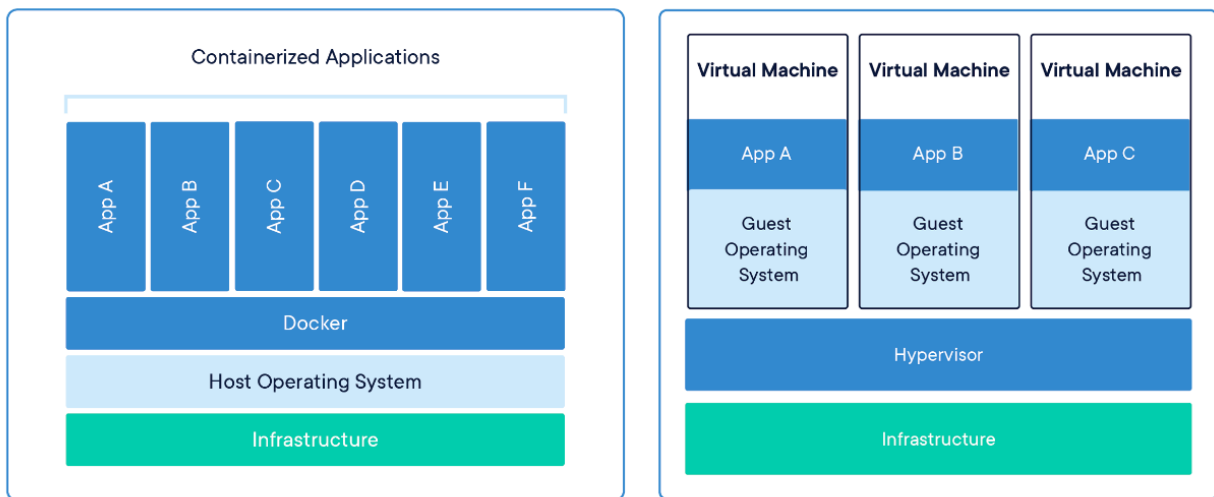
### b- Avantages

Les conteneurs simplifient le processus de création, de test, et de déploiement des applications sur différents types de plateformes depuis un poste de travail local vers un data center sur site ou même dans le cloud. Ces principaux avantages sont:

- **Moins de ressources nécessaires;**
- **Une portabilité accrue;**
- **Fonctionnement plus cohérent;**
- **Meilleure efficacité et développement d'applications.**

### c- Conteneurs et machine virtuelles

Les **machines virtuelles** (VMs) sont des environnements virtuels qui permettent de créer un système informatique virtuel disposant de ses propres ressources (processeur, mémoire, disque, interface réseaux) sur un système matériel physique. Les machines virtuelles permettent d'isoler aussi les applications comme les conteneurs. Cependant, leurs principales différences résident dans le poids et la portabilité. Les VMs sont plus lourdes et plus difficiles à porter contrairement aux conteneurs. La conteneurisation est la "**virtualisation**" au niveau applicatif.



**Figure 3 : Différence entre conteneurisation (conteneur) et virtualisation (machine virtuelle)**

#### d- Orchestration des conteneurs

L'orchestration des containers désigne le fait d'automatiser le déploiement, la gestion, la mise à l'échelle et la mise en réseau des conteneurs. Les outils d'orchestration des conteneurs Docker les plus connus sont Docker Swarm et Kubernetes.

- **Docker Swarm**

Intégré à **Docker Engine** depuis sa version 1.12 (2016), **Docker Swarm** est l'outil d'orchestration natif de Docker qui permet de gérer une multitude de conteneurs déployés sur une machine hôte.

Il offre une haute disponibilité aux applications à travers ces nombreux nœuds "workers" gérés par un nœud maître. Docker swarm est très efficace dans la gestion des ressources pour les conteneurs.

- **Kubernetes (K8s)**

K8s est une plateforme open source d'orchestration de conteneurs permettant d'automatiser le déploiement, la mise à échelle et la gestion des applications conteneurisées. Il est initialement développé par Google à l'image de son propre

système d'orchestration en interne (Borg). Il a été offert à la **Cloud Native Computing Foundation (CNCF)** en 2015.

K8s offre un environnement adapté pour le déploiement des applications distribuées grâce au cluster qui est mis à disposition.

*Les orchestrateurs de conteneur comme Kubernetes ou Docker Swarm permettent aujourd'hui d'exécuter tout type d'applications de façon distribuée et scalable sur des clusters de serveurs. Ceci permet de mieux exploiter les ressources disponibles, d'où notre intérêt pour ce concept. Il va nous permettre de gérer nos ressources efficacement tout en réduisant nos coûts.*

## 5- Devops

### a- Définition

**DevOps** est une approche qui concilie deux équipes antérieurement opposées: les développeurs (Dev) et les administrateurs de système et d'architecture (Ops). C'est donc une combinaison de philosophies, de pratiques et d'outils qui augmente la capacité de l'entreprise à livrer des applications et services à grande vitesse. Cette vitesse de livraison améliore la relation avec les clients sur un marché de plus en plus compétitif. Initialement utilisé dans le développement de logiciel, le DevOps est aujourd'hui adopté aussi pour la **Data science** et le **Machine Learning** à travers d'autres dénominations tels que **MLOps**, **AIOps**, **DataOps**.

### b- Avantages du DevOps

Les principaux avantages de cette approche sont:

- **La vitesse de développement;**
- **La livraison rapide des applications et services;**
- **La fiabilité du processus;**
- **La scalabilité;**
- **L'amélioration de la collaboration entre des équipes autrefois opposées.**

### c- Pratiques

Pour mener à bien ses objectifs, le DevOps fait usage d'un certain nombre de pratiques:

- Une des pratiques fondamentales du DevOps consiste à faire de “petites” mises à jour des applications qui permettent à l'entreprise d'innover plus rapidement et de limiter les risques de bugs pendant les mises en production. Cette pratique permet aussi d'identifier les mises à jour qui créent des bugs.
- L'usage des architectures microservices permet de réaliser des applications plus flexibles et innovantes. Cette architecture permet de découper les grosses et complexes applications en de plus petits projets simples et indépendants. Ces petites applications sont appelées microservices.

Pour mettre en œuvre ces pratiques, le DevOps fait usage de certaines bonnes pratiques.

### d- Bonnes pratiques DevOps

- **Intégration continue (CI)**

C'est une pratique de développement de logiciel dans laquelle les développeurs joignent les différentes modifications de leur code dans un dépôt central. Après cette opération les différents tests et déploiements automatisés s'exécutent.

- **Livraison continue (CD)**

Cette autre pratique du monde de développement logiciel permet l'automatisation de la génération, des tests et la préparation des codes pour la mise en production. C'est la suite de l'intégration continue en déployant l'application dans un environnement de tests ou de production.

- **Microservices**

Il s'agit d'une architecture de conception de logiciel qui permet de découper les grosses et complexes applications en de plus petits projets simples et indépendants. Ces petites applications sont appelées microservices.

- **Infrastructure as Code**

Cette approche consiste à faire usage des techniques de développement pour approvisionner et gérer l'infrastructure informatique (vm, stockages, serveur) comme les contrôles de version et l'intégration continue.

- **Monitoring et Journalisation**

Ils consistent pour les entreprises à surveiller les métriques et journaux d'événements des applications et infrastructures afin de mesurer l'impact de leur performance sur l'expérience de leurs clients. Elles récoltent des données afin de les analyser pour déterminer les impacts des changements et mises à jour.

- **Communication et Collaboration**

En DevOps, la communication et la collaboration sont au cœur des activités. L'utilisation des outils DevOps et l'automatisation des processus établissent une collaboration entre les équipes de développement et celles de l'exploitation.

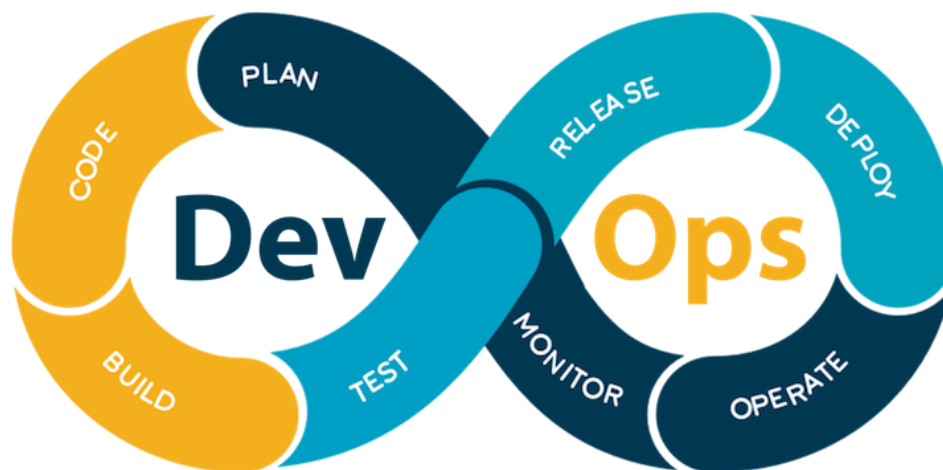


Figure 4 : Cycle de développement avec la méthodologie DevOps

*La solution mise en place dans le cadre de ce projet nécessite des équipes de développement et d'exploitation. L'usage du concept DevOps va augmenter la productivité et le niveau de collaboration entre ces équipes.*

## 6- Cloud computing

### a- Définition

Selon le [NIST](#) (National Institute of Standards and Technology), le **Cloud Computing** est un modèle permettant un accès facile et à la demande, via le réseau, à un pool partagé de ressources informatiques configurables, c'est-à-dire réseaux, firewalls, load balancers, serveurs, stockages, applications et services. On peut donc résumer que le Cloud Computing est le fait de présenter un ensemble de ressources informatiques en tant que services.

Ce modèle est doté de caractéristiques suivantes:

- **Elasticité;**
- **Accès simple;**
- **Mutualisation des ressources;**
- **Agilité;**
- **Facturation à la demande.**

### b- Différent type de cloud

On peut classer les services du cloud computing en trois catégories:

- **Cloud privé**

Il s'agit d'une infrastructure dédiée uniquement à une entreprise dont la gestion et l'hébergement peuvent être internes ou externes. Ce modèle de cloud permet à l'entreprise de contrôler la gestion et la sécurité de ses infrastructures. Les équipes informatiques peuvent ainsi approvisionner, allouer et livrer des ressources à la demande.

- **Cloud public**



Il s'agit de services fournis par un tiers via internet. Ces fournisseurs vendent des services généralement à la demande et les ressources fournies sont partagées et utilisables par tous. Ainsi les usagers ne payent que ce qu'ils consomment comme puissance de calcul, stockage ou bande passante.

- **Cloud hybride**

Dans ce modèle de cloud, l'entreprise veut créer un environnement unifié en tirant les avantages des cloud privés et publics. Un cas classique est que l'entreprise peut héberger leurs applications critiques ou sensibles sur leur cloud privé et utiliser le cloud public pour assurer le fonctionnement des applications nécessitant une forte scalabilité des ressources.

### **c- Différents modèles (niveau) de services cloud**

Malgré la perpétuelle évolution des services du cloud computing, ils peuvent être toujours subdivisés en trois grandes catégories:

- **Infrastructure as a service (IaaS)**

Les IaaS sont des services cloud qui offrent des ressources de calcul, de mise en réseau, de stockage à la demande. Il s'agit donc d'une infrastructure informatique virtualisée fournie aux utilisateurs via les APIs. Avec les IaaS, l'entreprise peut construire son data centers sans se soucier des coûts de maintenances matérielles telles que la maintenance des serveurs physiques, des matériels réseaux et les coûts d'entretien du data center (électricité, propreté). Ainsi elle peut bien garder la gestion de son parc informatique tout en maîtrisant ses coûts. Ce niveau de service Cloud est fourni grâce aux gestionnaires de VMs et stockages.

- **Platform as a service (PaaS)**

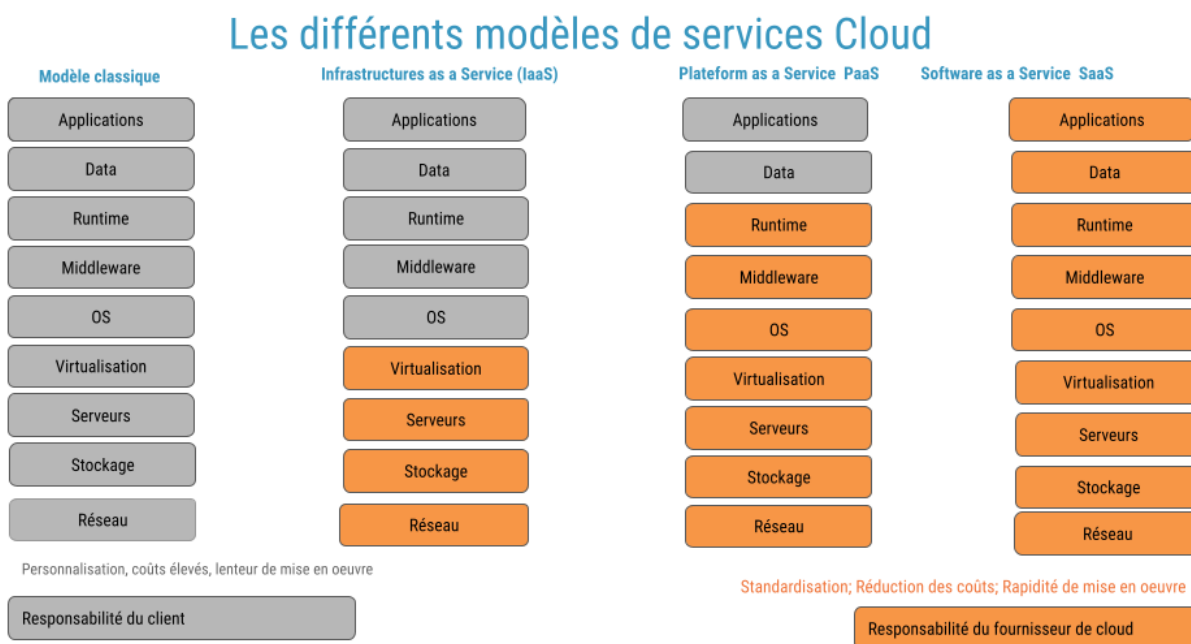
Il s'agit d'une plateforme de développement et de déploiement dans le cloud. Ce type de service permet aux utilisateurs de développer, exécuter et gérer leurs applications sans se soucier de la mise en place de l'infrastructure. Cette flexibilité permet aux

développeurs de mieux se concentrer sur leur code, d'où un gain énorme en productivité et une maîtrise des coûts en termes de gestion de licence. Ce niveau de cloud est possible grâce aux gestionnaires de conteneurs.

- **Software as a service (SaaS)**

Les SaaS constituent des logiciels qu'offrent des fournisseurs via internet. Dans ce modèle, le fournisseur héberge l'application et se charge de sa maintenance (mise à jour, correction, installation dans le data center). Quant aux utilisateurs, il ne font qu'utiliser l'application selon leur besoin sans se soucier de son installation, sa disponibilité, et des mises à jour. Grâce à cette flexibilité, les entreprises peuvent mieux maîtriser leurs coûts sans impacter leur productivité.

Ce schéma résume les services cloud en situant pour chaque catégorie les responsabilités qui incombent aux fournisseurs et aux utilisateurs:



**Figure 5: Les différents modèles de services cloud**

Pour le choix du type de service adapté à ses besoins, il est primordial que l'entreprise puisse déterminer ce qu'elle sait faire clairement et ce qu'elle veut faire faire (Au fournisseur)

---

## d- Avantages du cloud computing

L'utilisation des services procure de nombreux bénéfices qui permettent à l'entreprise d'éviter le gaspillage et de faire des économies de masse.

- **La flexibilité des coûts**

L'utilisation des services cloud offre une grande flexibilité aux niveaux des coûts dont la transformation des **CAPEX** (dépense d'investissement) en **OPEX** (dépenses d'exploitation) et l'option de facturation à la demande et à l'usage

- **La scalabilité du business**

Cette scalabilité permet à l'entreprise d'ajouter et de retirer des ressources et de la puissance en fonction de la croissance de ses activités.

- **L'agilité du business**

Avec les services cloud, l'entreprise réduit drastiquement son Time to market, ce qui lui permet de s'adapter rapidement aux changements.

- **La facilité de consommation**

Les services fournis via le cloud favorisent le self-service et une possibilité de consommer des ressources sans limite.

- **La stratégie de différenciation et spécialisation**

Les entreprises utilisant le cloud ont une facilité à s'adapter au contexte du moment et peuvent définir une meilleure stratégie grâce à une expérience définie par l'utilisateur lui-même.

- **Une meilleure relation client**

La relation client étant au cœur de la stratégie des entreprises, il est primordial de l'adapter rapidement aux changements. Les entreprises utilisant le cloud peuvent facilement mettre en place de nouveaux services qui vont améliorer la relation client.

---

## e- Principaux fournisseurs

Le marché du cloud étant en perpétuelle évolution, tous les géants du web, veulent prendre une bonne part de ce marché. Ce sont donc de multitudes services qu'ils offrent avec des adaptations afin d'être plus performant sur le marché.

Il existe deux grandes catégories de fournisseurs: les fournisseurs de cloud public et les fournisseurs de cloud privé.

Dans la première catégorie nous avons des géants suivants:

- **Amazon Web Service (AWS)**

En tant que précurseur dans ce secteur, AWS est le leader mondial avec plus de 32% (2021) de part de marché. Avec une qualité de service appréciée par les utilisateurs, AWS ne cesse d'innover et continue de créer des data centers dans presque toutes les régions du monde afin d'assurer la haute disponibilité.

- **Microsoft Azure**

Introduit par Microsoft en 2010, Azure détient la deuxième plus grande part de marché avec 19%, soit 50% avec un taux de croissance de 50% en 2021. Il propose plus de 200 services répartis entre les différentes catégories (IaaS, PaaS, SaaS)

- **Google Cloud Platform**

Fournie par Google, GCP propose l'hébergement sur la même infrastructure que celle qu'utilise Google pour ses produits internes tels que son moteur de recherche, G-Drive, GMap, G-mail, etc... GCP constitue la troisième plus grosse part de marché avec 7%. Google reste en constante évolution avec 56% de taux de croissance en 2021.

**Source: Synergy Research group**

Il est important de noter que ces trois fournisseurs proposent des services équivalents avec des performances sensiblement égales.

PRODUCT	aws	Microsoft Azure	Google Cloud Platform
Virtual Servers	Instances	VMs	VM Instances
Platform-as-a-Service	Elastic Beanstalk	Cloud Services	App Engine
Serverless Computing	Lambda	Azure Functions	Cloud Functions
Docker Management	ECS	Container Service	Container Engine
Kubernetes Management	EKS	Kubernetes Service	Kubernetes Engine
Object Storage	S3	Block Blob	Cloud Storage
Archive Storage	Glacier	Archive Storage	Coldline
File Storage	EFS	Azure Files	ZFS / Avere
Global Content Delivery	CloudFront	Delivery Network	Cloud CDN
Managed Data Warehouse	Redshift	SQL Warehouse	Big Query

Figure 6 : Services Cloud: AWS vs Azure vs GCP

Il existe d'autres fournisseurs tels que **Alibaba Cloud**, **Oracle Cloud**, etc...

Dans la deuxième catégorie, c'est-à-dire les fournisseurs de cloud privé nous avons entre autres **OpenStack**, **VMware**, **HP**, **Microsoft System Center**, etc.

*Dans l'optique de minimiser le coût de projet, le cloud computing s'avère être un bel environnement pour l'hébergement de notre infrastructure. Il offre une belle opportunité pour nous en termes de flexibilité et d'agilité.*

## II- TYPE D'ARCHITECTURE BIG DATA

Il existe aujourd'hui un nombre important d'architectures big data, l'architecture Lambda, l'architecture Kappa ou l'architecture Zeta, regroupées sous le nom de traitement polyglotte (Polyglot Processing).

## 1- Architecture Lambda

**Nathan Marz** a proposé le terme **Architecture Lambda** pour une architecture de traitement de données générique, évolutive et tolérante aux pannes, sur la base de son expérience de travail sur des systèmes de traitement de données distribués chez Backtype et Twitter. Il s'agit d'un moyen de traiter des quantités massives de données (**Big Data**) qui permet d'accéder à des méthodes de traitement par lots (**batch**) et de traitement temps réel (**streaming**) avec une approche hybride. Cette architecture est composée de plusieurs couches qui la rendent polyvalente.

### a- Couche Batch

Les données arrivent en continu depuis les sources de données. Les couches batch et streaming sont alimentées simultanément. La couche batch traite toutes les données et corrige éventuellement les données traitées par la couche streaming. De nombreux outils ETL et Data warehouse classique peuvent être utilisés dans cette couche. Les traitements batch sont programmés selon des horaires bien définis, généralement une ou deux fois par jour car la fréquence des traitements ne doit pas être trop importante afin de minimiser les tâches de fusion des résultats pour constituer les vues.

Les principales fonctions de cette couche sont:

- Stockage et gestion de l'ensemble des données
- Pré-calcul des vues batch.

### b- Couche Streaming

Cette couche traite les données qui ne sont pas prises en compte dans les vues batch déjà livrées. De plus, il ne traite que les données récentes afin de fournir une vue complète des données à l'utilisateur en créant **des vues temps réel**.

### c- Couche de Service

Les données sortant des couches batch sous formes de **vues batch** et celles provenant de la couche streaming sous forme de **vues temps quasi réel** sont

transmises au serveur. Cette couche, adaptée à tout type de base de données NoSQL, permet de stocker donc les vues créées par les couches batch et streaming. Les vues batch sont indexées afin qu'elles puissent être interrogées depuis cette base NoSQL ad hoc avec une **faible latence**.

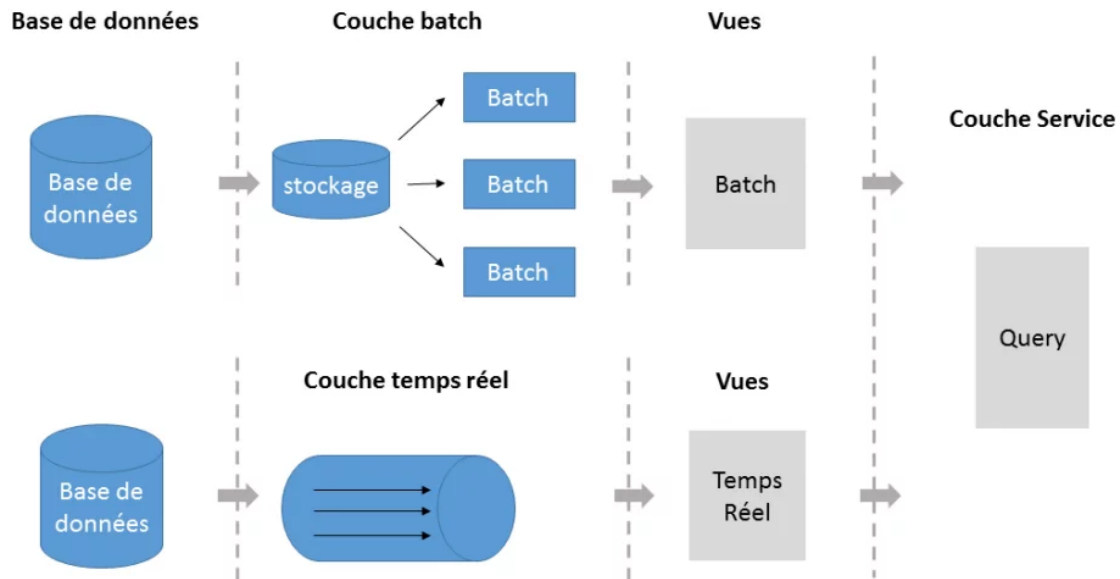


Figure 7: Schéma de l'architecture lambda

## 2- Architecture Kappa

Décrite pour la première fois par **Jay Kreps**, l'**architecture Kappa** est une architecture logicielle conçue pour le traitement des données temps réel (streaming). L'idée principale derrière cette architecture est d'effectuer à la fois les traitements temps réels et par lots avec les mêmes outils technologiques. Cette architecture permet de pallier la complexité de l'architecture Lambda car elle **fusionne les couches batch et streaming**. Elle se compose aussi de différentes couches.

### a- Couche de stockage temps réel

Ne permettant pas le **stockage permanent**, cette architecture Kappa est faite uniquement pour le traitement des données. Cette couche de stockage est donc

temporaire, elle permet de conserver les données sur une durée limitée (7 jours maximum pour certains outils). Pour implémenter cette couche, les moteurs de messagerie comme **Apache Kafka** sont utilisés. Ces outils permettent d'ingérer les données en temps réel tout en les conservant pour une période de temps. Cela permet à la couche de traitement de pouvoir repasser dans les données en cas d'éventuelles erreurs lors du traitement.

### b- Couche de traitement

Cette couche de traitement réalisée avec des outils comme Apache Spark ou Storm, traite à la fois les données streaming et batch. Cette polyvalence diminue la complexité de l'architecture. Les données sont essentiellement traitées en temps réel avant d'être déversées dans la couche de service.

### c- Couche de service

Tout comme l'architecture lambda, la couche de services stocke les vues traitées et les expose aux utilisateurs pour les requêtes. On y retrouve des technologies comme **Elasticsearch, Cassandra, Hive**, etc...

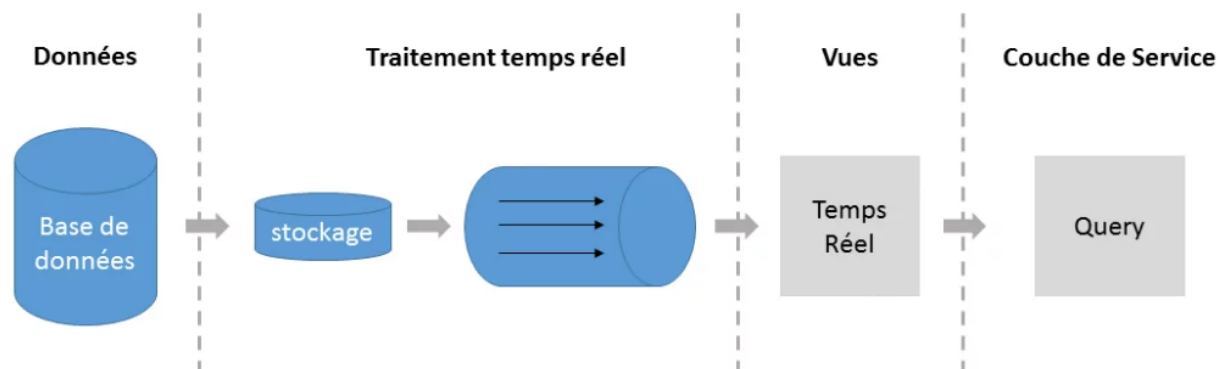


Figure 8: Schéma de l'architecture Kappa



## PARTIE III: ARCHITECTURE BIG DATA DE LA PLATEFORME D'ANALYSE DES RÉSEAUX SOCIAUX

---

*Nous présenterons dans cette partie, la description détaillée de l'architecture de la solution , ensuite l'étude comparative permettant de choisir les outils techniques en décrivant notre approche.*

## I- DESCRIPTION DÉTAILLÉE DE L'ARCHITECTURE DE LA SOLUTION

Notre solution permettra l'ingestion, le traitement et l'analyse des données des réseaux sociaux afin de mieux guider et renforcer l'action des acteurs publics.

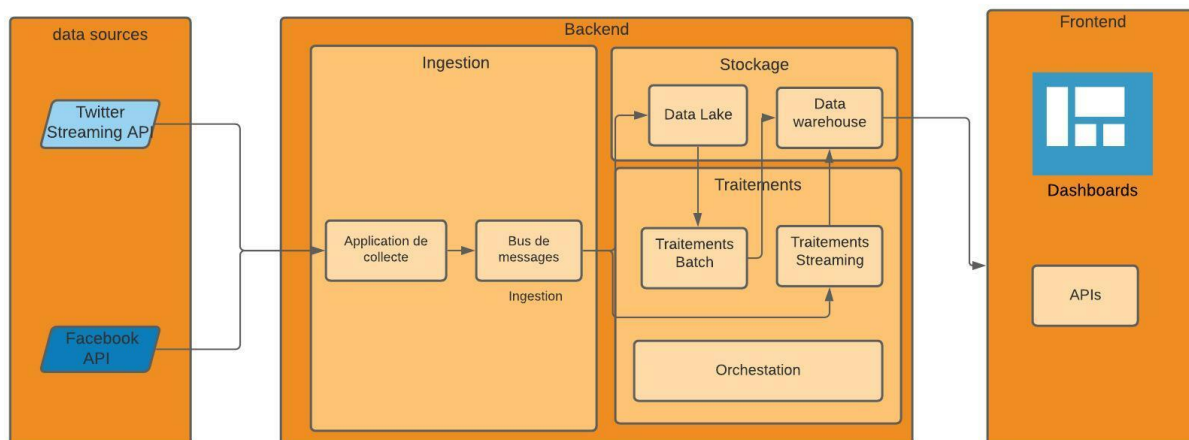


Figure 9: Schéma de l'architecture fonctionnelle de la plateforme d'analyse de données

### 1- Collecte des données depuis les réseaux sociaux

Cette couche nous permettra de récupérer les données depuis les sources en temps réel avant de les stocker.

#### a- Sources des données

Notre système va ingérer les données depuis plusieurs sources. Nous avons fait le choix des réseaux sociaux car ce sont des sources gratuites d'information. Les sources retenues sont:

- **Facebook**

Ce réseau social à travers ses milliards d'abonnés, constitue une immense source de données. Des millions d'ivoiriens expriment chaque jour leur opinion à travers ce canal. Il constitue donc pour nous une très grande base de données à exploiter.

- **Twitter**

Avec ce réseau social, de nombreuses institutions et personnalités véhiculent leurs messages avec une grande instantanéité. Cette instantanéité permet d'avoir les réactions et opinions beaucoup plus en temps réel.

### **b- Spécificités des données collectées**

Les données collectées depuis ces sources possèdent les spécificités suivantes:

- **Contenu**

Les données recueillies depuis ces sources sont les publications, messages et commentaires sur les pages des organes gouvernementaux pour Facebook.

Quant à Twitter, il s'agit des tweets liés à la crise Covid19 en Côte d'Ivoire. En effet, ces tweets sont de courts textes représentant des prises de parole spontanées, avec des informations de géolocalisation, des dates etc...

Nous nous intéresserons qu'aux tweets publiés en Côte d'Ivoire en filtrant par les mots clés liés à Covid (**vaccin, épidémie, coronavirus, covid**, etc.).

- **Fréquence**

Les utilisateurs de Facebook et Twitter publient des messages et tweets par milliers en une minute. Ces données sont donc d'une grande vélocité, d'où la nécessité de les traiter en temps réel avec les outils Big data.

- **Volume**

Les données produites par les utilisateurs ivoiriens sur ces réseaux sociaux sont des dizaines de milliers par jour et évoluent très rapidement.

Par ailleurs, Il est important de noter que l'utilisation des données mises à disposition par les réseaux sociaux respecte toutes les réglementations en matière de protection des données personnelles (**RGPD, loi n°2013-450**). Pour ce faire, nous utiliserons donc une application validée par Facebook et Twitter et leur APIs officielles.

Ces exigences montrent clairement que nos données doivent être collectées avec un système d'ingestion big data. De plus, le système doit être hautement disponible et suffisamment scalable afin de garantir la sécurité des données.

## 2- Analyse et traitements des données

Les données récupérées depuis les réseaux sociaux ne sont pas dans des formats exploitables. Il est donc nécessaire d'effectuer un certain nombre de traitements et d'analyses afin de faciliter leur exploitation.

### a- les étapes d'analyse et de traitements des données

Les données passeront par les étapes suivantes afin de les rendre exploitables:

- **Nettoyage des données**

Il va consister à :

- Formater et normaliser les données;
- Identifier et supprimer les doublons;
- Supprimer les données à caractère personnel.

- **Prétraitement**

Cette phase va garantir le traitement légal et éthique des données. Il va s'agir de :

- Anonymiser les données afin de respecter la vie privée;
- Détecter et retirer les contenus malveillants, c'est-à-dire fake news, contenus violents, obscénité, etc.

- **Analyse des données**

A cette phase, les données subiront des analyses techniques et socio-économiques.

Pour les analyses techniques, il s'agit de l'utilisation de l'Intelligence Artificielle pour détecter la nature et les sujets évoqués dans les messages traités. Quant aux analyses socio-économiques, elles consisteront à intégrer des expertises métier afin de proposer des analyses plus adaptées pour les acteurs publics.

### **b- Architecture de la couche de traitement.**

Nous avons fait le choix d'une architecture Lambda afin d'être plus flexible entre traitement temps réel (**streaming**) et traitements par lots (**batch**).

- **Couche de Streaming**

La couche de streaming permet de traiter les données en temps réel au fur et à mesure qu'elles sont ingérées. Elle calcule des vues incrémentales qui vont compléter les vues batch afin de fournir des données plus récentes.

- **Couche Batch**

Cette couche permet, tout comme la précédente, de traiter les données mais cette fois les traitements sont massifs, réguliers et effectués par lots. Le système de batch processing est capable de traiter les données aussi bien manuellement que automatiquement.

- **Orchestration**

Un système d'orchestration doit être mis en place pour assurer le déclenchement automatique des traitements. Ce système doit être aussi capable de lancer des traitements manuels en cas de panne du système afin que les données soient toujours à jour.

## **3- Stockage des données**

Le stockage est très important dans notre système qui se veut suffisamment robuste et sécurisé pour résister aux pannes et garantir la disponibilité des données. Ainsi deux systèmes de stockage sont nécessaires:

- **Data lake**

Cette couche de stockage est liée à la couche de collecte de données car elle permet de stocker les données brutes ingérées à chaud. Elle sert aussi de source pour la couche de traitements par lots. Ce type de stockage est appelé Data lake et doit être compatible et accessible depuis nos applications de collecte et de traitement.

- **Data warehousing**

Le data warehouse est un outil qui stocke les données déjà traitées et analysées pour les mettre à la disposition des équipes analytiques. Cette couche est liée à la couche de traitement car elle constitue la destination. De plus, le data warehouse permet de séparer les données selon les besoins des équipes. Il facilite ainsi l'exploitation et l'analyse des données.

#### **4- Couche de Visualisation**

Cette couche est d'une importance capitale dans notre architecture car elle permet de faire ressortir les KPIs et informations pertinentes des données. Il s'agit de tableaux de bords, d'APIs, et d'applications web pour faciliter la visualisation des données aux utilisateurs. La couche de visualisation prend ses données soit directement dans le data warehouse ou via un middleware (API) connecté au data warehouse. Ces interfaces doivent être:

- Ergonomiques
- Accessibles
- Personnalisables
- Réactives

## **II- ETUDE COMPARATIVE, CHOIX DES OUTILS TECHNIQUES ET DESCRIPTION DE L'APPROCHE**

Après avoir schématisé une architecture qui répond à nos objectifs, nous devons choisir les technologies capables d'implémenter ces visions en tenant compte des performances, des coûts et des spécificités du projet.

### **1- Environnement**

#### **a- Data Center Local vs Cloud Computing**

Cette plateforme peut être réalisée avec Data Center local ou dans le cloud. Pour la première option, elle nécessite beaucoup d'investissement et va nécessiter beaucoup d'interventions humaines pour la maintenance de la plateforme. Ainsi, nous avons fait le choix d'un environnement Cloud, qui va permettre d'avoir la puissance nécessaire sans coûts d'investissement massifs. De plus, certains services proposés déjà dans le cloud répondent parfaitement à certaines de nos exigences telles que la flexibilité, la tolérance aux pannes et la haute disponibilité.

Les coûts de maintenance de l'infrastructure seront évités.

### **b- Fournisseur Cloud (Cloud Provider)**

Comme vu dans la précédente partie, il existe une multitude de fournisseurs de services cloud (**AWS, Azure, GCP**, etc.). Nous devons effectuer un choix judicieux qui s'inscrit dans notre vision. Après de longues séances de comparaison, nous avons choisi **Microsoft Azure** pour la qualité de leur service et surtout parce que les services qu'ils proposent peuvent nous aider à atteindre nos objectifs en réduisant nos coûts.

## **2- Application de collecte des données**

### **a- Application web Flask (Python)**

Notre application de collecte doit être fonctionnelle à tout moment et suffisamment tolérante aux pannes. Nous avons donc opté pour une application web faite avec le **Framework Flask** de **Python** pour garder de la simplicité. Les données sont récupérées depuis les sources grâce aux clients Python officiels de l'API de Twitter (**Tweepy**) et de Facebook (**Graph API**).

Pour faciliter le monitoring, une interface a été développée pour **contrôler le démarrage et l'arrêt du processus** de collecte des données. Ce processus est basé sur le système de multi-threading qui permet au client de Twitter de rester actif en arrière-plan sans entraver l'affichage de l'état du système dans la page de monitoring. Pour rendre encore plus robuste cette application, un module de **relance automatique** a été intégré. Ce système de relance permet de relancer automatiquement le processus de collecte de données lorsqu'une erreur se produit, ce qui nous évite les pertes de

données. A côté de ces modules, un **système d'alerte** a été mis en place à travers la messagerie **Slack** afin d'informer l'équipe en temps réel des erreurs qui peuvent survenir et surtout quand le système de relance n'arrive pas à relancer le système.

### **b- Mode de filtrage des données (Twitter)**

Les données sur twitter en particulier, ont un champ de géolocalisation. Cependant, seulement 1% des tweets publiés sur Twitter ont ce champ renseigné, ce qui ne nous permet pas de récupérer suffisamment de données liées à la Côte d'Ivoire. Pour éviter cette perte de données, nous avons ciblé tous les tweets covid en **français** dans le but d'appliquer certaines **heuristiques** dans la phase de traitement pour détecter la localisation réelle.

### **c- Hébergement de l'application web**

L'application web est hébergée dans **Azure Apps Service**, un service d'hébergement d'application et de site web proposés par Azure. Ce choix s'explique par la haute disponibilité de ce service et son coût qui est l'un des plus bas sur le marché.

Ainsi, notre application de collecte remplit parfaitement les exigences car elle est suffisamment robuste pour collecter le maximum de données disponibles dans nos sources.



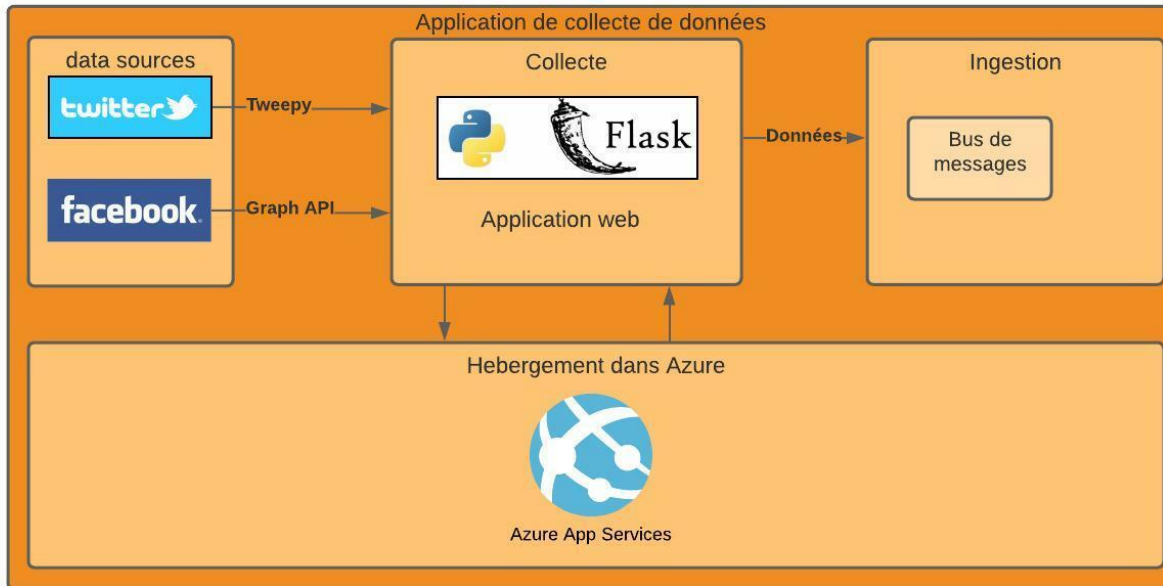


Figure 10 : Schéma de l'architecture de l'application de collecte de données

### 3- Système d'ingestion de données

#### a- Event Hub

Les données collectées depuis **Twitter**, et **Facebook** en temps réel grâce à notre **application de collecte** doivent être ingérées au fil de l'eau. De nombreuses solutions open source sont disponibles sur le marché comme **Apache Kafka**, qui fait partie des meilleurs dans le domaine. Cependant, l'utilisation de cette solution va nécessiter la mobilisation de serveurs supplémentaires pour son hébergement. Pour éviter ces coûts supplémentaires tout en gardant le niveau de performance recherché, nous avons fait le choix de l'équivalent proposé par Microsoft via son service cloud: **Event Hub**. Ce service d'ingestion de données en temps réel et entièrement managé par Azure garde les mêmes concepts et protocoles compatibles avec Apache Kafka tout en épargnant aux utilisateurs la maintenance des serveurs. Il est capable d'ingérer et diffuser plus d'un **million d'événements par seconde** depuis n'importe quelle source.

De plus, avec Event Hub, Azure nous assure un **Service Layer Agreement (SLA)** de plus de **99,99%** de disponibilité avec une capacité de rétention de données

temporaire allant jusqu'à une semaine. Ce service est connecté à notre application de collecte de données en tant que bus de messages; c'est-à-dire que les données collectées sont poussées directement dans cet outil qui les retient temporairement avant leur stockage dans le Data Lake. Cette opération dépend du réseau, ce qui peut engendrer des délais d'attente lors des requêtes. Pour éviter cette attente et ne pas perdre de données, elle est réalisée en **mode asynchrone**, ce qui permet à notre système de collecter le maximum de tweets.

### b- Event Hub Capture Event

Les données stockées temporairement dans Event Hub, doivent être récupérées et poussées dans le Data lake à l'aide d'un autre outil de traitement (Spark, SQL, K-SQL).

Cependant, Event Hub, grâce à sa fonctionnalité **Capture Event**, nous permet de stocker les données ingérées de manière permanente dans notre Data Lake sans passer par un outil supplémentaire de traitement de données. Avec Capture Event, nous stockons nos données dans un format **avro** avec le partitionnement suivant:

**{Year}/{Month}/{Day}/{Hour}/{Minute}/**

Ce partitionnement nous permet de stocker nos données par **année, mois, jour, heure et minute** avec une fenêtre de **cinq (5) minutes**. La définition d'une fenêtre de quelques minutes nous évite d'avoir un trop grand nombre de petits fichiers dont la lecture dans notre ETL serait trop **chronophage**.

Cette approche nous évite les nombreuses écritures dans le Data lake qui sont très coûteuses en termes de ressource et de temps.

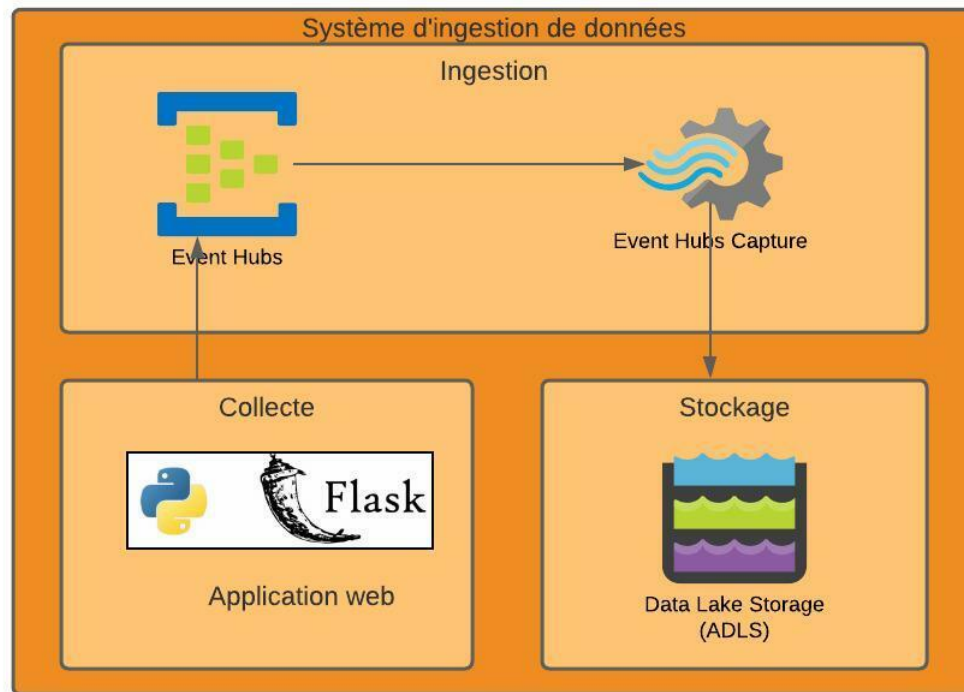


Figure 11: Schéma du système d'ingestion de données

## 4- Outils d'ETL

### a- Apache Spark

Comme vu précédemment, de nombreux outils ETL sont présents sur le marché. Certains sont proposés par des concepteurs de logiciels avec de très bonnes performances. Cependant, pour éviter les coûts supplémentaires de licence, nous avons choisi un outil du monde open source qui va nous permettre de concevoir notre système ETL personnalisé: **Apache Spark**. Spark se présente comme l'outil par excellence des calculs distribués et des traitements des données big data. Spark n'est pas le seul outil open source capable de faire ce traitement, il y a aussi **dask**. Mais Spark se démarque par sa compatibilité avec la plupart des systèmes de gestion de cluster que nous avons expérimentés, notamment les cluster **Mesos**, **Yarn** et le tout récent **Kubernetes** avec les conteneurs.

Avec Spark, nous chargeons les données depuis le Data Lake. Spark est capable de lire dans la plupart des Data Lake grâce à sa couche **Hadoop (HDFS)**. Les données chargées vont subir toutes les transformations et analyses prévues (nettoyage, prétraitement, analyse avec les modèles d'IA, etc..) avant d'être stockées dans notre Data warehouse. Tous les traitements effectués dans Spark sont faits de manière distribuée; c'est-à-dire sur un cluster de plusieurs machines afin d'avoir une grande puissance de calcul. Les données traitées sont **partitionnées par jour** pour le stockage.

### b- Traitements du job Spark

Les analyses et traitements réalisés dans Spark sont:

- **Détection et filtrage** des tweets par localisation (country, user location)

Les tweets récupérés depuis Twitter ne sont pas tous de la Côte d'Ivoire et très peu ont leur champ de localisation renseigné (**champ correspondant au lieu de publication du tweet**). Nous avons donc mis en place cette opération de filtrage qui permet de retrouver les tweets des utilisateurs de Côte d'Ivoire. A l'aide d'**expressions régulières**, nous détectons tous les tweets dont les utilisateurs ont dans leur champ de localisation (**champ correspondant au lieu de résidence de l'utilisateur**), des villes ou communes de Côte d'Ivoire. Pour affiner encore plus la recherche, sachant que certains tweets n'ont aucun de ces deux champs, nous recherchons dans le texte du tweet certains mots clés liés à la Côte d'Ivoire (les hashtags populaires et les mots clés propres au pays). Ces heuristiques ont permis d'augmenter le volume de tweets de **1000 %** par rapport à l'approche basée sur le **lieu de publication du tweet**.

- **Récupération des hashtags, mentions, urls**

Cette étape permet de récupérer des hashtags, mentions et les urls des tweets pour les mettre dans des formats tabulaires. Ces champs permettent de faire ressortir des statistiques, les mots et personnalités tendance.

- **Récupération du texte entier des tweets (full text)**

Les textes des tweets sont tronqués lorsqu'ils excèdent un certain nombre de caractères fixés par Twitter. Cette limite peut causer d'énormes biais dans nos

analyses. Cependant, il est possible de récupérer le texte en entier à l'aide de certaines heuristiques. Nous avons donc remplacé tous les textes tronqués par leur version complète.

- **Nettoyage du texte** (cleaned text)

Les textes des tweets contiennent certains mots qui permettent de ressortir les ressentis des utilisateurs. Nous voulons afficher ces mots dans un **word cloud**. Les textes contiennent cependant de nombreux mots sans valeur ajoutée: articles, adverbes, préposition, etc. De plus, des symboles et des chiffres existent dans ces textes. Un nettoyage est donc nécessaire afin de supprimer ces mots vides (stops words), symboles et chiffres. Cette opération permet d'avoir des textes plus propres et propice à la visualisation dans le word cloud.

- **Anonymisation des noms d'utilisateurs** (username)

Ce traitement est effectué par mesure de sécurité afin de ne pas dévoiler les identités des utilisateurs (**protection des données à caractère personnel**). Techniquement, nous utilisons une fonction de hachage qui attribue une signature numérique à chaque nom d'utilisateur. Cette opération est irréversible, ce qui nous assure un certain niveau de protection des données des utilisateurs.

- **Détection de sentiment** (sentiment)

Il s'agit d'une analyse basée sur les modèles de machine learning afin de détecter les sentiments ressentis dans les tweets (**positif, négatif, neutre**). Notre modèle de détection de sentiment est un modèle basé sur les transformers qui a été **fin-tuné (Transfert learning)** sur un large volume de tweets et a une **précision d'environ 71%**.

- **Détection des thématiques** (topic)

Tout comme l'analyse de sentiment, cette analyse de topic modeling a été réalisée grâce aux réseaux de neurones de type transformer. Elle permet de détecter avec une **précision de 66%** les thématiques évoquées dans les tweets. Les thématiques sont :

- **mesures** : tweets évoquant les mesures covids;
- **opinion** : tweets exprimant l'opinion des utilisateurs;
- **chiffres** : tweets contenant des statistiques liées à covid;

- **divers** : pour les autres thématiques, économiques, actualités, etc).

Pour construire les modèles classification des sentiments et le modèle de classification des thèmes des tweets nous avons utilisé l'apprentissage par transfert. L'apprentissage par transfert consiste à exploiter les connaissances acquises (état du réseau de neurones) lors de la résolution d'une tâche source pour améliorer l'apprentissage dans la résolution d'une nouvelle tâche. Dans notre cas la tâche source est un modèle de langue et les tâche cible sont les modèles de classification des sentiments et des thèmes. Le modèle de langue que nous utilisons est le modèle introduit (Louis Martin, et al. 2019), pré-entraîné sur 138 GB de textes en français.

Tous les modèles de deep learning ont été hébergés sur **Hugging Face**, une plateforme d'hébergement de modèle de types transformers. Cette approche apporte plusieurs avantages. Il permet de:

- **Mettre à la disposition** de la communauté de nos modèles
- **Versionner** le modèles depuis un dépôt distant;
- **Faciliter d'accès** aux modèles depuis n'importe quel code python, sans persistance;
- **Séparer les mises à jour** des modèles par l'équipe de Data Science et celles du job Spark par les Data Engineer;
- **Éviter le stockage des modèles** dans nos images Docker.

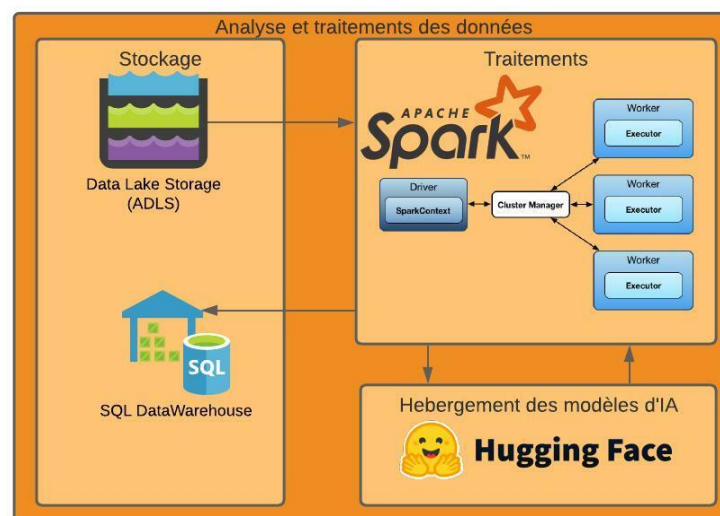


Figure 12: Schéma de l'outil d'ETL (Apache Spark)

## 5- Type de Cluster

### a- Kubernetes

Notre système ETL, Spark doit être déployé dans un cluster qui se doit être hautement scalable, disponible et tolérant aux pannes. Concevoir un cluster avec **Yarn** ou **Mesos** va nécessiter la création de machines virtuelles comme nœuds. Sur ce plan, **Kubernetes** propose un cluster avec des nœuds dont chacun peut héberger plusieurs **workers** et **drivers** de Spark à travers le système de **pods**. Nous choisissons donc Kubernetes comme système de gestion de cluster afin de disposer de plus de ressources et de flexibilité. Cependant, pour l'hébergement du cluster Kubernetes lui-même, nous avons opté pour le service managé d'Azure à savoir **Azure Kubernetes Services (AKS)**. AKS est une plateforme managée offerte par Azure afin de faire tourner des applications dans un cluster Kubernetes sans se soucier des nombreuses configurations de machine et de réseaux. Avec AKS, il est possible d'ajouter et de supprimer des nœuds de notre cluster selon la charge de travail.

Dans cette approche,

- Un service est déployé dans AKS (**Spark Operator**) afin de soumettre les jobs Spark dans le cluster;
- Après soumission du job, un pod Kubernetes est créé en tant que **Driver Spark**;
- Le Driver Spark crée des exécuteurs Spark (**Workers**), en leur distribuant les données à traiter;
- Chacun des workers, traite donc une partie des données avant de la renvoyer au driver qui les agrège et les stocke.

### b- Hébergement des images Docker pour Kubernetes

Spark Operator, les drivers et workers de Spark, sont des conteneurs Docker dont les images sont hébergées dans le dépôt privé d'images de conteneurs **Azure Container Registry (ACR)**. Ce service est connecté à AKS qui y récupère toutes les

images dont il a besoin. ACR offre un stockage sécurisé et garantit la haute disponibilité des images Docker.

Par ailleurs, AKS peut utiliser les **Spot VMs** comme nœuds. Ces Spot VMs sont des machines virtuelles Azure dont la tarification permet **d'économiser plus de 85% des coûts normaux** des machines virtuelles. Utiliser de telles VMs pour exécuter nos applications sans perdre en performance est un véritable défi technique que nous avons réussi à régler grâce à la mise en place d'un système **d'auto-scaling** qui remplace automatiquement les nœuds qui tombent.

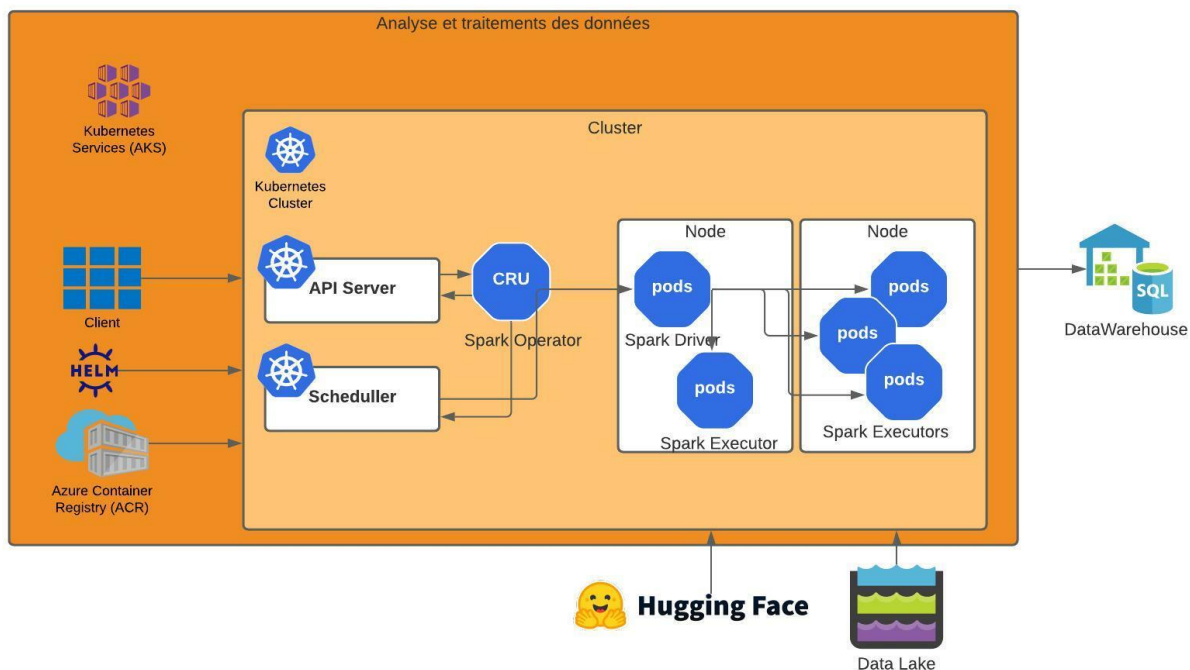


Figure 13: Schéma du fonctionnement de Spark dans sur le cluster Kubernetes (AKS)

## 6- Système d'orchestration

### a- Azure Function App

Pour lancer nos jobs de traitement, nous avons conçu une application web qui permet de:



- Allumer et Éteindre notre cluster AKS quand cela est nécessaire afin d'éviter les coûts supplémentaires lorsque qu'aucun jobs de traitements ne tournent;
- Lancer le job spark dans le cluster Kubernetes via le Spark Operator;
- Vérifier l'état de progression des jobs lancés dans le cluster.

Cette application n'est pas exécutée 24h/24h, et lorsqu'elle est lancée, elle ne dure que quelques minutes. Nous avons donc choisi de l'héberger grâce à **Azure Function App**. Azure Function est une **infrastructure sans serveur (serverless)** d'Azure permettant d'héberger et exécuter uniquement du code. Ce service est facturé à l'exécution. Avec les Function App, vous pouvez héberger votre code écrit avec de nombreux langages de programmation disponibles sur le marché (Python, Java, C#, JavaScript, F#, ...). Les applications disponibles dans ce service peuvent être déclenchées par des requêtes HTTP dans lesquelles on peut passer des paramètres ou par des exécutions programmées avec les **crons**. La limite de la durée d'exécution de ces applications est de dix (10) minutes. L'utilisation de ce service nous permet donc de rester dans notre objectif et de ne payer que ce que nous utilisons comme ressource à chaque exécution.

### b- Azure Data Factory

Pour compléter cette couche, nous avons fait le choix d'un autre service d'Azure , **Data Factory (ADF)**, comme orchestrateur de tout notre système. Data Factory permet de programmer toutes étapes de notre processus ETL via les pipelines. Nous avons donc mis en place la pipeline avec les étapes suivantes:

- **Start Cluster**

Cette étape est le point d'entrée de notre pipeline. Si elle échoue, aucune autre tâche n'est exécutée. En effet, il s'agit d'une requête HTTP avec pour paramètres le type d'opération "**start-cluster**" qui ordonne le démarrage du cluster AKS via le client Python d'Azure (**azure-cli**). Les instructions sont exécutées dans la Function App. Cette opération étant critique pour notre système, elle nécessite l'authentification avec un compte Azure autorisé. Une fois cette tâche est réalisée avec succès, l'utilisateur est automatiquement déconnecté pour éviter tout risque de sécurité et la pipeline passe aux autres étapes.

- **Start time et End time**

Il s'agit de deux tâches qui s'exécutent simultanément afin de déterminer de manière dynamique les dates de début et de fin de traitement. Ces deux tâches sont configurées en fonction de la fréquence de nos traitements.

- **Start Submit**

Cette tâche consiste à soumettre le job Spark de traitement dans le cluster AKS. Tout comme la tâche Start Cluster, c'est aussi une requête HTTP avec cette fois comme type d'opération "**spark-submit**" ainsi que les informations d'authentification. Cette opération est possible grâce au client officiel Kubernetes de Python (**kubernetes**) et aussi **azure-cli** pour l'authentification et la connexion au cluster AKS. Pour traiter les données de la période spécifiée, les dates de début et de fin obtenues dans la précédente tâche sont passées en paramètre dans cette requête. Une fois que le job est soumis correctement, l'identifiant du job est renvoyé dans la réponse de la requête pour suivre sa progression.

- **Job Pod Id**

Cette tâche est une étape intermédiaire permettant de stocker l'identifiant du job Spark dans une variable afin de faciliter son utilisation dans les tâches ultérieures.

- **Check Job Status**

Cette tâche permet de suivre l'état d'avancement du job dans le cluster. Techniquement, il s'agit d'une boucle "**tant que**" dans ADF qui est constituée de trois activités:

- **Running**

Cette tâche permet de marquer une pause de 60 secondes avant la vérification de l'état. Ceci permet d'éviter à notre système de lancer des requêtes successivement pour vérifier l'état du job. Ainsi nous évitons des coûts d'exécutions inutiles sur Function App.

- **Job Status**

Il s'agit aussi d'une requête HTTP avec le type d'opération "**job-status**" et l'identifiant du job concerné qui récupère l'état du pod responsable de ce job. Le pod peut afficher les états **Pending**, **Running**, **Succeed**, **Fail**, **Error** et **Unknow**. Le pod passe par d'autres états intermédiaires entre Pending et Running.

- **Job State**

Cette tâche stocke l'état du job en cours dans une variable.

Ce processus est répété tant que l'état du job n'est pas à **Succeed**, **Fail**, **Error** ou **Unknow**. Une fois que l'un de ces états est atteint, la boucle s'arrête et la pipeline passe l'étape suivante.

- **Stop Cluster**

Cette tâche est analogue à la tâche Start Cluster susmentionnée avec le type d'opération "**stop-cluster**". Elle est évoquée selon deux conditions:

- En cas d'échec de n'importe quelle étape de notre pipeline
- Lorsque l'état du job passe à **Succeed**, **Fail**, **Error** et **Unknow**.

Chaque évocation de cette tâche est accompagnée d'une alerte envoyée à l'équipe en spécifiant la raison. Ainsi l'équipe est informée de l'état du système et peut donc prendre les dispositions nécessaires en cas d'échec ou d'erreur.

Ce processus que nous avons imaginé, nous permet de traiter efficacement nos données sans coût supplémentaire sur notre facture Azure et de réagir très rapidement lorsqu'un problème survient afin de garantir la haute disponibilité de la plateforme.

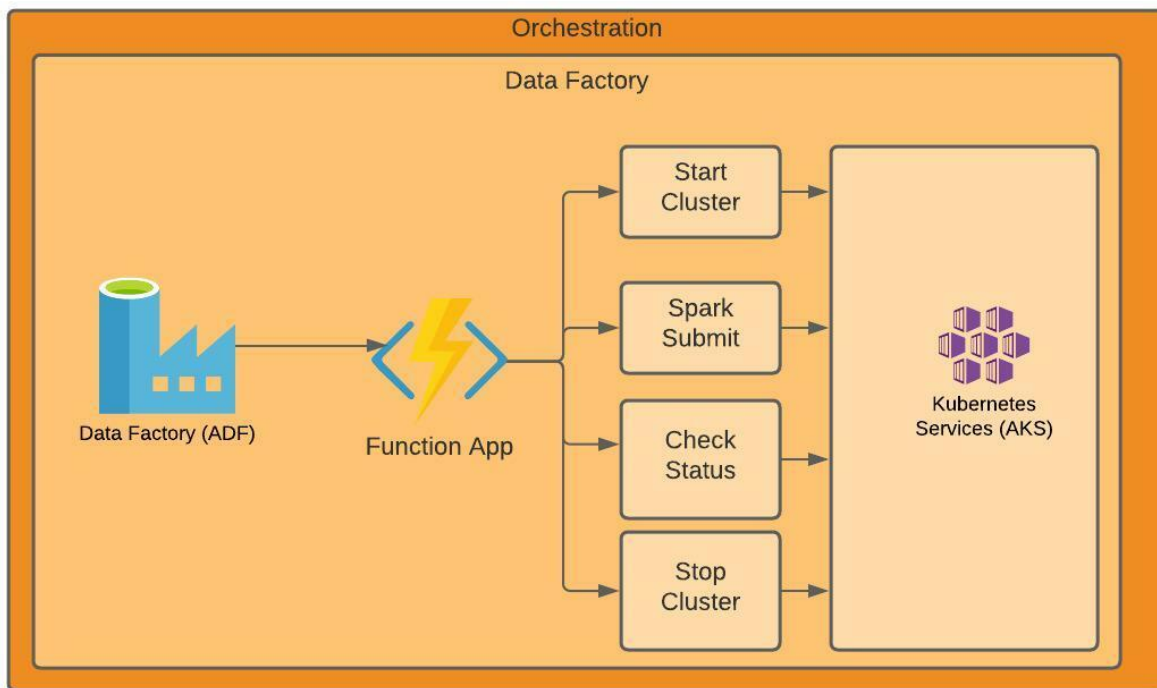


Figure 14: Schéma du système d'orchestration dans Data Factory

## 7- Data Lake, Data Warehouse

Pour le stockage des données brutes collectées, nous avons fait le choix d'un système de stockage objet: **Azure Data Lake Storage (ADLS)**. Ce stockage d'Azure est compatible avec Hadoop qui est la couche de stockage utilisée par notre outil ETL Spark. De plus, ADLS est un service hautement sécurisé et disponible pouvant contenir une très grande quantité de données et à moindre coût.

Et comme Data warehouse, **Azure Synapse Analytics** a été choisi pour sa facilité de connexion avec ADLS et son coût à l'usage qui est très compétitif sur le marché.

En résumé, pour le stockage des données brutes (Data Lake), nous avons ADLS et pour le stockage des données traitées (Data warehouse), nous avons Azure Synapse Analytics.

---

### a- Azure Data Lake Storage (ADLS)

ADLS est directement connecté à Event Hub, notre système d'ingestion, qui y stocke les données qu'il récupère depuis l'application de collecte. Les données stockées sont lues par notre ETL (Spark sur AKS) pour le traitement avant de les déverser de nouveau dans un autre dossier de ADLS.

### b- Azure Synapse Analytics

Synapse est un Data warehouse hybride car il dispose d'un système de traitement **Transact-SQL (T-SQL)** et de stockage (**Microsoft SQL Server**). Cependant l'utilisation du système de stockage dans Synapse a un coût très élevé. Notre approche a donc consisté à stocker les données physiquement dans ADLS (ce qui ne coûte quasiment rien), et créer des vues à partir de ces données dans Synapse avec T-SQL. Ainsi notre système de visualisation peut aisément exploiter les données fournies à travers ces vues ayant une structure tabulaire. Cette approche s'avère intéressante car le coût du traitement dans Synapse est de 1 Tera octets de données pour seulement 5 dollars, ce qui est extrêmement avantageux pour nous.

Notre système de stockage répond parfaitement à nos objectifs, c'est-à-dire le stockage sécurisé d'une grande quantité de données à moindre coût.

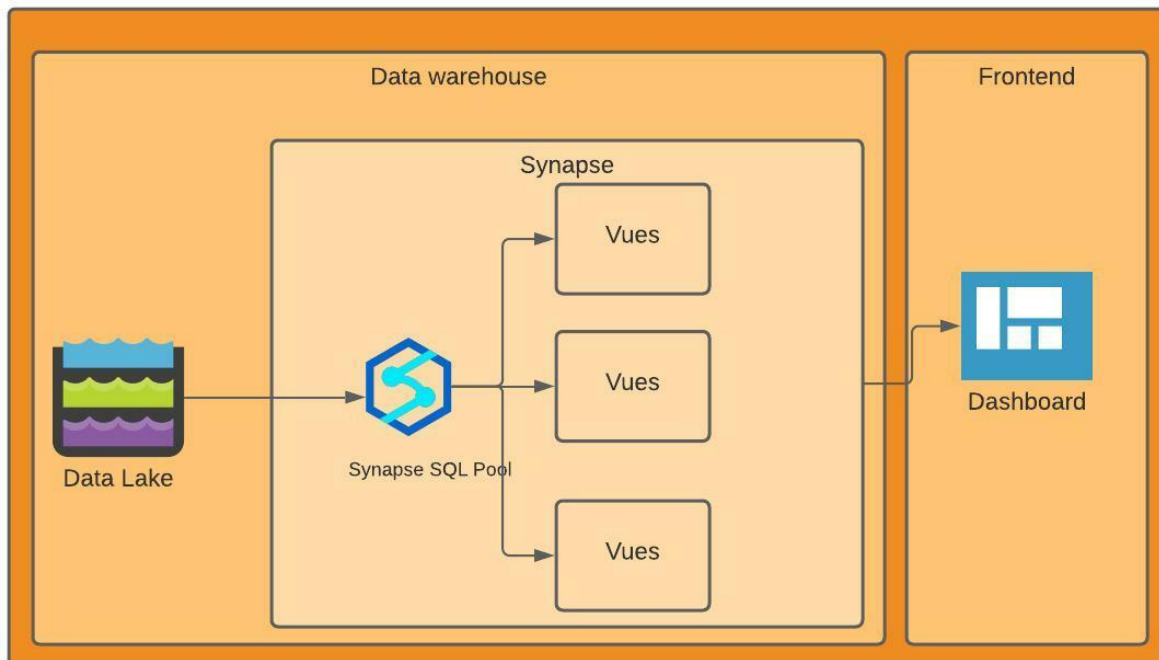


Figure 15 : Schéma du synapse de data warehouse avec synapse et ADLS

## 8- Outils de visualisation

De nombreux outils de visualisation sont disponibles sur le marché comme **Tableau**, **Qlikview**, **Dash (Python)** etc. Cependant **Power BI** a été retenue pour son ergonomie, sa facilité d'utilisation, sa grande interactivité et le coût moindre de sa licence. Power BI propose un service en ligne qui permet d'héberger nos tableaux et rapports facilitant son accès aux utilisateurs. De plus Power BI offre des tableaux de bords très interactifs et une facilité de connexion avec les systèmes de stockage d'Azure.

Power BI va lire les vues créées dans Synapse (Data warehouse), avant de les afficher dans le tableau de bord qui a été conçu. Cette opération s'effectue régulièrement afin d'avoir nos rapports toujours à jour.

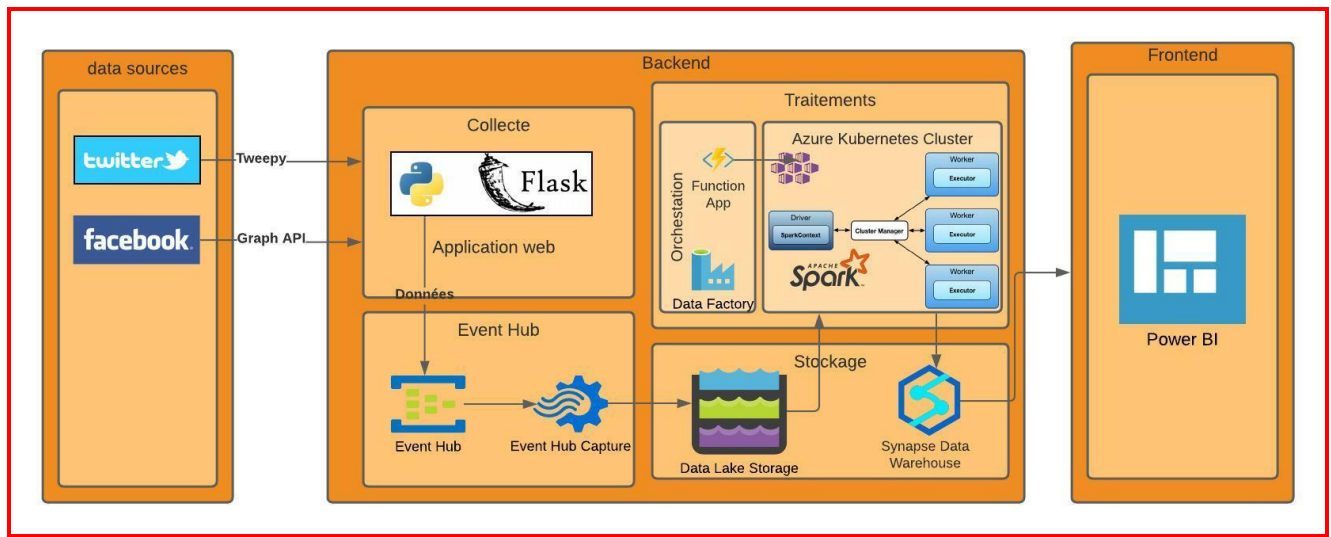


Figure 16: Schéma final de l'architecture de la plateforme Big data

## PARTIE IV: MISE EN PLACE ET DÉPLOIEMENT DE LA SOLUTION BIG DATA/CLOUD

---

*Nous présenterons dans cette partie la mise en place de l'architecture et le déploiement des différents composants*



## I- PRÉPARATION DE L'ENVIRONNEMENT

Avant de mettre en place notre plateforme, il est nécessaire de préparer notre environnement de travail. Deux environnements doivent être configurés: Environnement Azure et notre environnement local.

### 1- Environnement Azure

L'envergure de notre projet exige une organisation rigoureuse de notre environnement. Azure offre un ensemble de solutions pour la gestion de projets et de l'organisation.

#### a- Azure Active Directory (Azure AD)

Notre solution est composée d'un ensemble d'applications et services critiques dont les accès doivent être réservés à des personnes spécifiques selon leur rôle. Azure AD est le service LDAP d'Azure qui va nous permettre d'organiser et de gérer les accès aux ressources. Nous avons donc créé un ensemble de rôle sur Azure AD en spécifiant pour chaque rôle un ensemble de droits.

- **Rôle Admin or Owner**

Les membres de l'équipe ayant ce rôle disposent de tous les droits dans Azure. Certains de leurs droits sont la gestion de comptes utilisateur, la gestion des abonnements Azure, la gestion des factures, la gestion quotas des ressources, définition des règles de création des ressources, etc. Une personne type ayant ce droit est le CTO de l'entreprise.

- **Rôle Developer or Contributor**

Ce rôle est attribué au Data Engineer, Data Architect et Devops du projet. Ce rôle leur permet de créer des ressources dont ils ont besoin selon les règles définies. Ils ont aussi le droit de gérer les permissions d'accès aux ressources qu'ils créent.

L'organisation de l'équipe avec Azure AD permet d'identifier les activités des membres de l'équipe et de situer les responsabilités facilement. Tout ceci s'inscrit dans notre vision qui est de concevoir une plateforme sécurisée qui va gérer des données en réalité sensibles.

## **b- Abonnements Azure**

Les abonnements Azure sont des conteneurs logiques de ressources. Il s'agit d'un dossier virtuel dans lequel les ressources sont créées. Le paiement de facture se fait par abonnement.

Dans notre contexte, l'organisation des abonnements est très importante. Les membres de l'équipe sont liés à des abonnements selon leurs tâches. Nous avons organisé les ressources en deux grandes souscriptions:

- **Abonnement de développement**

Cet abonnement contient les ressources créées pendant la phase de développement de tests. Ces ressources étant utilisées pour de l'exploration, un certain nombre de limites sont imposées sur celles-ci afin d'éviter les coûts exorbitants en phase de développement.

- **Abonnement de production**

Dans cet abonnement, sont créées les ressources de production, c'est-à-dire qui vont servir effectivement pour la plateforme. Les membres de l'équipe qui ont accès à ces ressources sont les CTO, les chefs d'équipes. Ces ressources sont critiques et doivent avoir des configurations strictes qui permettent de les sécuriser et assurer la haute disponibilité de la plateforme.

## **c- Groupes de ressources**

Les groupes de ressources permettent de mieux organiser les ressources afin de mieux les gérer. Un groupe de ressource va contenir des ressources ayant un lien particulier. Par exemple, les ressources de la couche de traitement de notre plateforme

peuvent être mises dans un ressources groupe. Cela va faciliter leur gestion. Les ressources sont contenues dans les abonnements.

Dans notre cas, nous avons organisé les ressources comme suit:

- **Groupe de ressources manuelles**

Il s'agit des ressources créées pendant l'exploration à la main. Elles sont créées rapidement afin de tester des approches. Ces ressources seront supprimées après la mise en production.

- **Groupe de ressources DevOps**

Ce groupe est constitué des ressources de développement ou tests ayant été déployés grâce aux méthodes DevOps, c'est-à-dire CI/CD.

- **Groupe de ressources officielles**

Ce sont les ressources de mise en production. Elles sont critiques et ont une gestion plus rigoureuse.

#### **d- Azure DevOps**

Azure DevOps est une plateforme qui fournit les services de supports aux équipes de développement afin de planifier et collaborer dans la mise en place d'applications. Cet outil de collaboration va permettre de concilier nos équipes constituées de Data engineer, Data Scientist, chefs de projets, et responsables systèmes pour mener à bien le projet. Ce service est composé d'une multitude de fonctionnalités(**Repos, Pipelines, Board, Artifacts, Tests Plans**) .

Dans le cadre du projet, nous avons organisé les codes des différentes applications et services grâce au **dépôt de code (Repos)** d'Azure DevOps. Les processus de tests et déploiements ont été automatisés grâce aux **Pipelines** Azure DevOps. Pour réaliser cette automatisation, il est nécessaire de configurer un **Agent** Azure DevOps (une VM ubuntu) qui va exécuter l'ensemble des instructions de nos pipelines de déploiement.

### e- Key Vault

La plateforme mise en place dans le cadre du projet, contient un grand nombre d'applications et services qui doivent communiquer entre eux. Cette communication doit être sécurisée avec la gestion des accès faite depuis Azure AD. Cependant, les informations d'authentification et d'autorisation ne doivent pas figurer en clair dans le code afin d'éviter le risque de divulgation. Key Vault, nous permet de stocker et chiffrer de manière sécurisée toutes les informations d'authentification, d'autorisation et de configuration. Ainsi, nos applications peuvent accéder à ces données nécessaires pour leur fonctionnement sans qu'elles n'apparaissent dans le code ou soient à la disposition de personnes non autorisées.

Notre Key Vault contient tous nos secrets, configuration, clés, et certificats de sécurité.

## 2- Environnement local

Pour interagir avec nos ressources dans le cloud et écrire nos codes pour le traitement des données, nous avons besoin de configurer certains outils sur notre machine locale.

Les principaux outils utilisés sont :

- **Python et ses librairies ( tweepy, Fask, kubernetes, azure-cli, pyspark, etc...);**
- **Visual Studio Code avec Extensions Azure;**
- **Kubectl (client kubernetes);**
- **Spark;**
- **Git, Docker, Azure CLI.**

Avec ces outils, nous allons interagir avec nos ressources dans Azure et écrire nos pipelines de traitement de données.

## II- DÉPLOIEMENT DES SERVICES DANS AZURE

### 1- Préparation des Azure Resource Manager (ARM) Templates

**ARM** est le service gestion et de déploiement des ressources d'Azure. Il propose un ensemble d'outils et méthodes qui facilitent le déploiement automatisé des ressources (Infrastructure as Code). Avec les ARM templates, nous spécifions pour chacune des ressources que nous devons déployer les différentes caractéristiques et configurations qu'elle doit avoir. Les templates sont des fichiers **JavaScript Object Notation (JSON)** qui définissent de manière déclarative l'infrastructure et les configurations de notre projet. L'un des plus gros avantages de cette approche, c'est que nous pouvons supprimer et redéployer toute notre infrastructure et être sûr d'obtenir exactement le même résultat, ce qui facilite l'automatisation de tout le processus. Avec ARM, nous pouvons:

- **Gérer notre infrastructure** à l'aide de modèles déclaratifs plutôt que de scripts;
- **Déployer**, gérer et superviser toutes les ressources de notre solution sous forme de groupe, plutôt qu'individuellement;
- **Redéployer votre solution** tout au long du cycle de vie de développement et vérifier que nos ressources sont déployées dans un état cohérent;
- **Définir les dépendances** entre les ressources pour qu'elles soient déployées dans le bon ordre;
- **Appliquer le contrôle d'accès** à tous les services, car le contrôle d'accès en fonction du rôle Azure (**Azure RBAC**) est intégré de manière native à la plateforme de gestion;
- **Appliquer des étiquettes aux ressources** pour organiser de manière logique toutes les ressources de notre abonnement.

Par ailleurs Microsoft Azure a introduit une nouvelle alternative à **ARM** baptisé **BICEP**. Cette nouvelle fonctionnalité sera plus avancée que ARM avec de nombreux avantages. Nous comptons migrer vers cette version lorsque la version finale sera disponible.

## 2 - Préparation des pipelines d'industrialisation

Les pipelines sont des chaînes d'instructions qui décrivent les étapes de construction, de test et de déploiement de nos codes dans notre infrastructure Azure.

Ces pipelines sont construits dans Azure DevOps Pipeline à travers des fichiers YAML. De manière déclarative nous indiquons toutes les étapes et instructions de déploiement pour nos applications et services. Les pipelines sont déclenchés par des événements que nous spécifions afin d'automatiser leur exécution. Un exemple typique est de déclencher un pipeline lorsqu'il y a de nouveaux changements dans nos codes. De manière générale, nos pipelines sont structurés comme suit:

- **Etape de configuration**

L'étape de configuration dans nos pipelines consiste à définir toutes configurations liées aux services concernés telles que la définition du nom de la ressource dans laquelle le code doit être déployé, la récupération des configurations spécifiques et des secrets dans Key vault, etc...

- **Etape de récupération du code source**

À cette étape, l'Agent Azure Devops chargé de l'exécution de la pipeline, récupère le code source depuis les dépôts spécifiés à l'étape précédente.

- **Etape de Tests**

Dans chacun de nos pipelines, nos codes passent par une batterie de tests fonctionnels et de qualité afin de s'assurer du bon fonctionnement de nos services. L'un de nos tests classiques est par exemple le test de conformité avec pylint qui implémente les recommandations du PEP8 (Standard python). De nombreux autres tests fonctionnels sont effectués. Si le code échoue au test alors tout le processus est annulé, rien n'est soumis à notre plateforme et les équipes sont informées. Cette pratique nous permet d'éviter de nombreux problèmes.

Une fois que ces problèmes détectés sont corrigés, le processus reprend. Lorsque les tests sont effectués avec succès, la pipeline passe à l'étape suivante.

- **Étape de déploiement du code dans les services**

Les tests ayant été effectués avec succès, le processus de déploiement est lancé. Cette étape contient des instructions précises du déploiement du code. Elle varie selon la nature des services. Les instructions de déploiements s'exécutent de manière séquentielle et tiennent compte de toutes les configurations faites à la première étape. Une fois l'étape de déploiement effectuée, l'application ou le service est alors disponible et prêt à exécuter les tâches qui lui incombent.

Les pipelines Azure offrent une grande flexibilité dans le déploiement de nos applications et services. La cohérence entre ces étapes fait que si l'une des étapes échoue, toute la chaîne s'arrête, ce qui garantit l'intégrité de notre infrastructure.

### **3- Déploiement de la plateforme**

Tout le processus de déploiement ayant été automatisé depuis Azure DevOps, les différents composants de notre infrastructure peuvent être ainsi déployés aisément. Nous allons donc présenter l'interface des différents composants.

#### **a - Azure App service**

L'application de collecte d'ingestion a été hébergée dans App Service. L'interface de contrôle se présente comme suit:

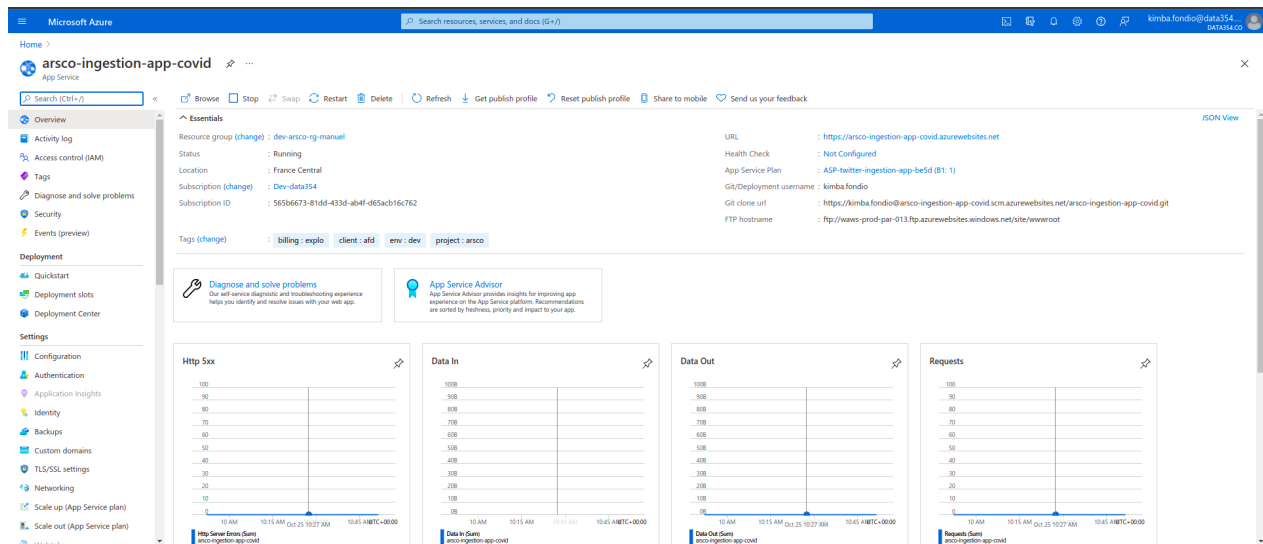


Figure 17: Interface de l'application de collecte de données dans App Services

## b- Azure Event Hub

Après le déploiement, Event Hub reçoit les données depuis l'application de collecte et stocke dans le Data Lake. Sur cette image, on observe les activités de Event Hub

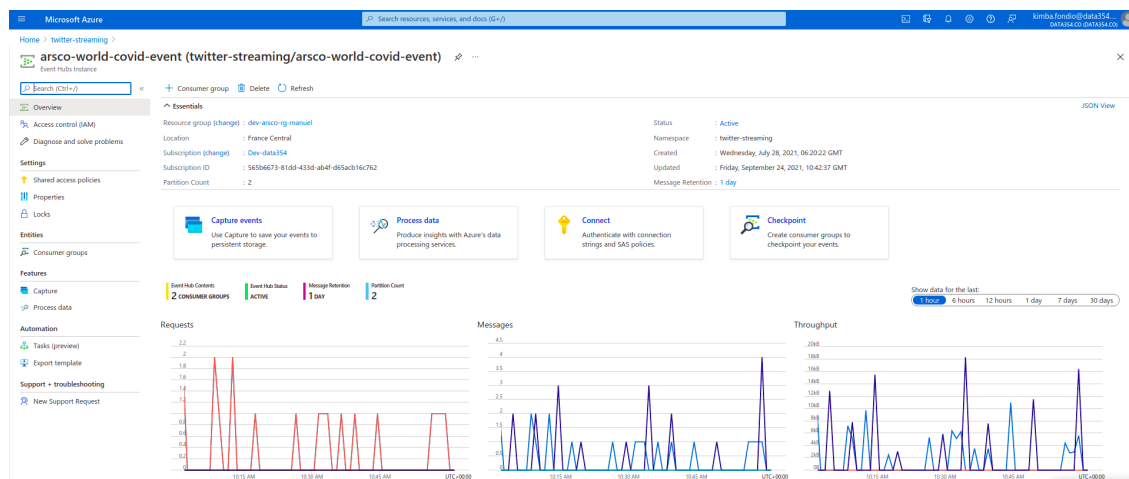
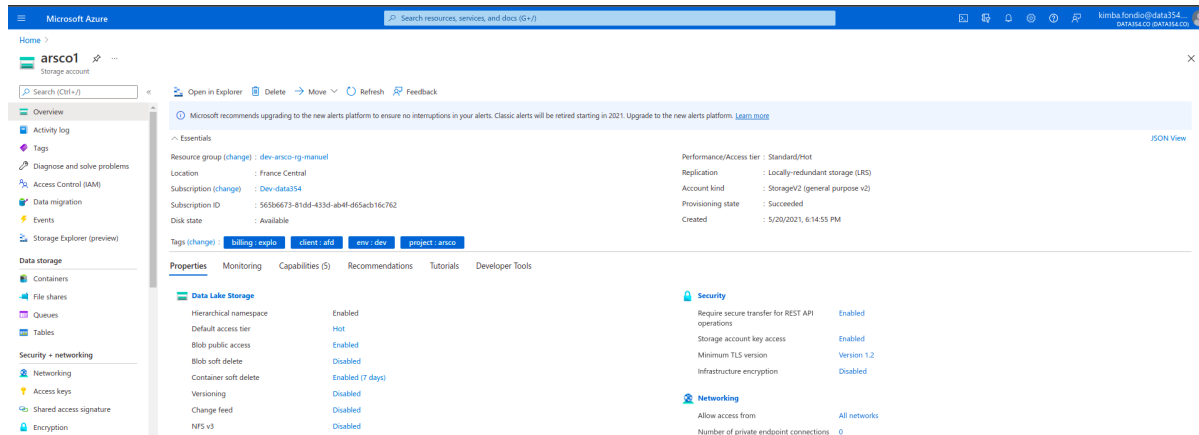


Figure 18 : Interface de contrôle et de monitoring de Event Hub

## c- Azure Data Lake Storage Gen2 (ADLS)



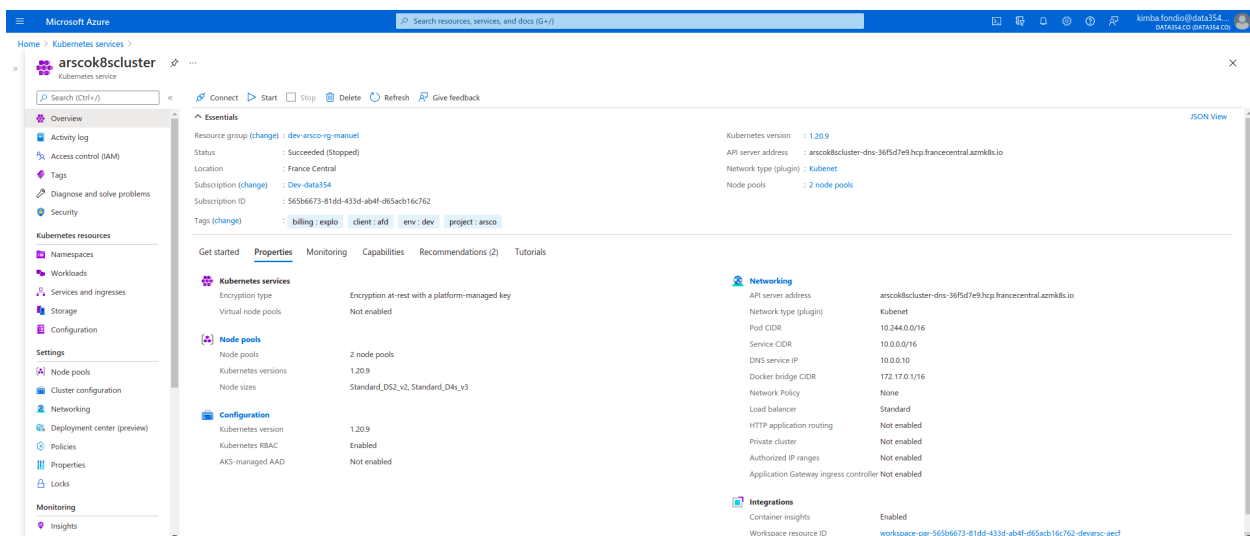
Les données ingérées par Event Hub sont stockées dans Azure Data Lake Storage (ADLS), notre Data Lake. L'image suivante présente l'interface de contrôle de ADLS, qui nous permet de configurer les autorisations de ce service avec d'autres composants de l'infrastructure.



**Figure 19 : Interface de contrôle du Data Lake (ADLS)**

## d- Azure Kubernetes Services

Apache Spark, notre ETL a été correctement déployé dans le Cluster Kubernetes d'Azure (AKS). Le nombre de nœuds, les systèmes d'authentification, ainsi que tous les services nécessaires ont été configurés. Voici la capture de l'interface de AKS qui permet de monitorer l'état de ce service ainsi que des applications qui y sont déployées.



## e- Azure Container registry (ACR)

ACR, notre dépôt d'images Docker, est connecté à AKS afin que ce dernier puisse récupérer toutes les images nécessaires pour la création des pods et services.



## f - Azure Function App

Notre chaîne de traitement est déclenchée dans Data factory à travers une web app qui est une Function App. L'interface de cette application présente les statistiques sur le nombre d'exécutions ou d'appels.

**aks\_master\_function** | Monitor

Search (Ctrl+F)

Overview

Developer

- Code + Test
- Integration
- Monitor**

Function Keys

### Invocations

Success Count: 195 (Last 30 Days) | Error Count: 1893 (Last 30 Days)

### Invocation Traces

The twenty most recent function invocation traces. For more advanced analysis, run the query in Application Insights.

Run query in Application Insights | Refresh

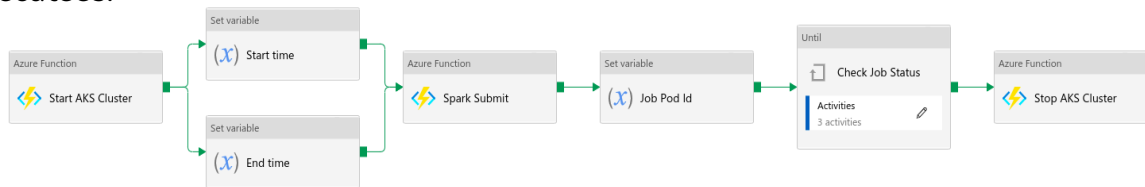
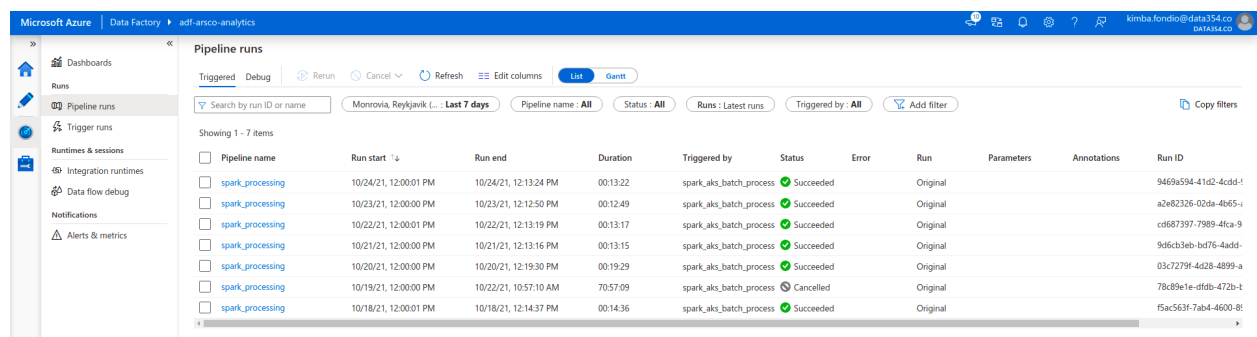
Filter invocations

Date (UTC)	Success	Result Code	Duration (ms)	Operation Id
2021-10-24 12:11:19.599	✓ Success	200	121560	e9da2cd3af01c141a75f585f12cd2ad1be
2021-10-24 12:11:08.232	✓ Success	200	88	42adae594aa1094ae95342d8baec0e06c
2021-10-24 12:10:00.146	✓ Success	200	93	9a7a042bf9ef9d99eeaff895775c511
2021-10-24 12:08:54.162	✓ Success	200	94	288ab2119ef9940a14c3953de9d26d7
2021-10-24 12:07:48.536	✓ Success	200	99	310488db090824986db0face4023635
2021-10-24 12:06:42.019	✓ Success	200	100	034506e47c5fe489c2af2cf4988986
2021-10-24 12:05:35.913	✓ Success	200	108	39154640653d043095e5c22858ed00e
2021-10-24 12:04:30.055	✓ Success	200	94	c204709721c5b0480dea6c20917f53
2021-10-24 12:03:21.686	✓ Success	200	2070	c33066eb06374593dd5e4610789056
2021-10-24 12:00:06.472	✓ Success	200	190098	2252cb4022ae1ea187db3ef2b08a69f5
2021-10-23 12:10:16.132	✓ Success	200	151176	197c3286645ab43b4bd799aa2df14164
2021-10-23 12:10:05.768	✓ Success	200	87	8830606afe4848493530a87d164177fb

Figure 22 : Interface de Azure Function App

### g- Azure Data Factory (ADF)

Notre système d'orchestration ADF présente une interface très riche avec des sections qui permettent de concevoir les pipelines et de monitorer toutes tâches exécutées.

Pipeline name	Run start	Run end	Duration	Triggered by	Status	Error	Run	Parameters	Annotations	Run ID
spark_processing	10/24/21, 12:00:01 PM	10/24/21, 12:13:24 PM	00:13:22	spark_aks_batch_process	Succeeded		Original			9469a594-41d2-4cdd-1
spark_processing	10/23/21, 12:00:00 PM	10/23/21, 12:12:50 PM	00:12:49	spark_aks_batch_process	Succeeded		Original			a2e82326-02da-4b65-i
spark_processing	10/22/21, 12:00:01 PM	10/22/21, 12:13:19 PM	00:13:17	spark_aks_batch_process	Succeeded		Original			cd687397-7989-4fca-9
spark_processing	10/21/21, 12:00:00 PM	10/21/21, 12:13:16 PM	00:13:15	spark_aks_batch_process	Succeeded		Original			9d6cb3eb-bd76-4add-
spark_processing	10/20/21, 12:00:00 PM	10/20/21, 12:19:30 PM	00:19:29	spark_aks_batch_process	Succeeded		Original			03c7279f-4d28-4899-a
spark_processing	10/19/21, 12:00:00 PM	10/22/21, 10:57:10 AM	70:57:09	spark_aks_batch_process	Cancelled		Original			78c89e1e-dfdb-472b-t
spark_processing	10/18/21, 12:00:01 PM	10/18/21, 12:14:37 PM	00:14:36	spark_aks_batch_process	Succeeded		Original			f5ac563f-7ab4-4600-8

Figure 24: Interface de monitoring de Data Factory

### h- Azure Synapse Analytics

Les données enrichies sont stockées sous forme de vues dans notre Data Warehouse, Azure Synapse. Ce service a une interface graphique appelée Synapse Studio, qui est une forme d'Interface de Développement Intégré en ligne. Cette interface permet d'écrire des requêtes Transact-SQL.

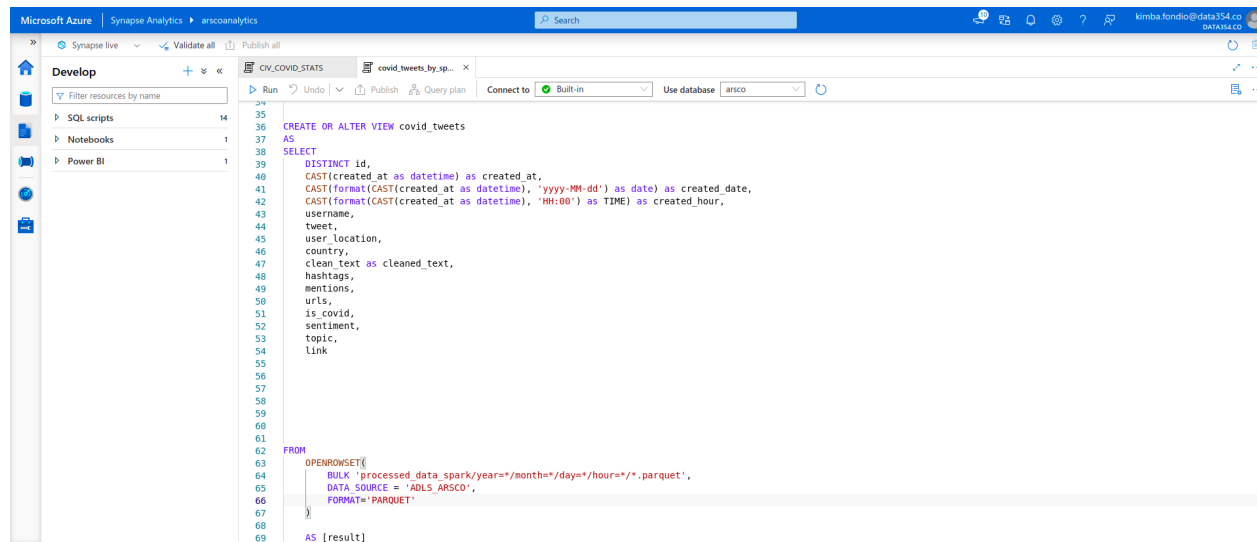


Figure 25: Interface de Azure Synapse Analytics (Synapse Studio)

## i- Power BI

Pour faciliter les analyses et visualisation des résultats des traitements, un tableau de bord a été créé dans Power BI. Les résultats sont affichés à l'aide de visuels permettant de faciliter la lecture par les décideurs.

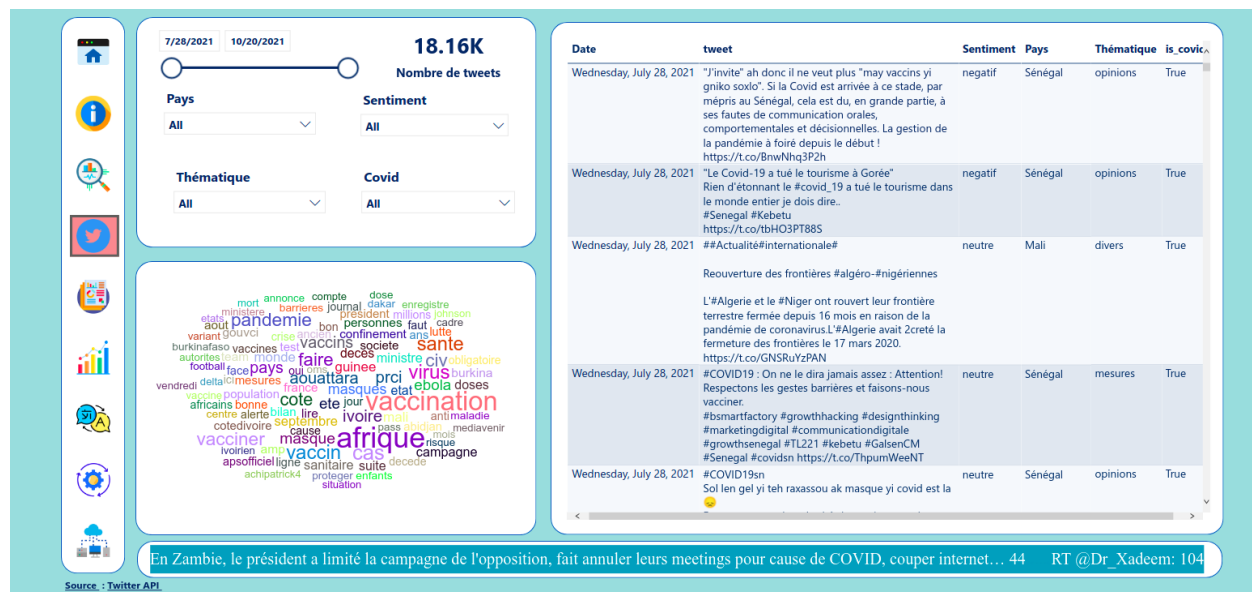


Figure 26: Interface de visualisation avec Power BI

## CONCLUSION GENERALE

Au terme de notre projet, nous avons mis en place un outil d'analyse des réseaux sociaux qui va permettre au gouvernement de mieux suivre et évaluer les impacts des politiques qu'il met en place dans le cadre de la lutte contre le COVID-19.

La conception d'un tel outil nous a permis d'explorer de nombreux concepts étudiés pendant notre formation. La mise en pratique de ces concepts dans un environnement réel a permis de consolider les connaissances acquises.

L'analyse des réseaux sociaux est une innovation de taille car elle permet aux organisations et entreprises de suivre les tendances autour de leurs activités en temps réel sur ces nouvelles plateformes où les utilisateurs n'hésitent pas à se prononcer sur les différents sujets qui minent nos sociétés.

Si la première version de notre outil est focalisée sur le COVID-19, sa portée ne se limite cependant pas à cette thématique. Cet outil peut être étendu à d'autres problématiques telles que la cherté de la vie, la prévention de conflits, les périodes électorales, le suivi et l'évaluation des politiques publiques. Sous la responsabilité d'un directeur technique dynamique, spécialiste des big data, le stage s'est déroulé dans de très bonnes conditions. Le projet faisant l'objet de notre thème s'est quasiment achevé, A présent il nous reste à le parfaire et le déployer pour utilisation. Comme perspectives, nous comptons intégrer de nouvelles possibilités d'analyses et de généraliser l'utilisation de cet outil sur d'autres problématiques quotidiennes ou à venir.

---

## BIBLIOGRAPHIE & WEBOGRAPHIE

### OUVRAGES

- [1] **James Warren and Nathan Marz**,  
*Big Data: Principles and Best Practices of Scalable Realtime*, MANNING, 2015.
- [2] **Uma N. Dulhare**,  
*Machine Learning and Big Data: Concepts, Algorithms, Tools and Applications*, Wiley, 2020
- [3] **Priyanka Gotter, Kiranbir Kaur, Tanveer Kaur**  
*Enhancing Availability for NoSQL Database Systems using Failover Techniques Paperback*, Kindle Edition, June 15, 2019.

### MEMOIRES

- [4] **TANO Assandé Jacob**,  
“Traduction automatisée des langues africaines: cas du lingala, mémoire de fin d’étude pour l’obtention du Master en Data Science - Big Data, IDSI, 61 pages, 2019-2020.

### SITES WEB

- [5] **Building and Deploying Microservices with Azure Kubernetes Service (AKS) and Azure DevOps**,  
<https://kishore1021.wordpress.com/2018/11/08/building-micro-services-with-azure-container-service-aks-and-azure-devops-part-1/>
- [6] **Architecture big data**,  
<https://www.cyres.fr/blog/architecture-lambda-big-data/>,
- [7] **Documentation Microsoft Azure**,  
<https://azure.microsoft.com/en-us/>
- [8] **Le plan d'un mémoire : quels éléments intégrer ?**  
<https://www.scribbr.fr/category/plan-memoire/>,

## TABLE DES MATIERES

REMERCIEMENTS	4
SOMMAIRE	5
LISTE DES FIGURES	6
LISTE DES ABREVIATIONS	7
GLOSSAIRE	8
AVANT-PROPOS	9
RESUME	10
INTRODUCTION GENERALE	11
PARTIE I : ENVIRONNEMENT DE TRAVAIL	12
I - PRÉSENTATION DE LA STRUCTURE D'ACCUEIL	13
1- Structure d'accueil	13
2- Domaines de compétences	13
3- Equipes	14
II - PRESENTATION DU PROJET	15
1- Contexte du projet	15
2- Objectif général	15
3- Objectifs spécifiques	16
4- Planning d'exécution	16
5- Equipe Projet	17
PARTIE II: PRÉSENTATION DES CONCEPTS ET TYPE D'ARCHITECTURE BIG DATA	18
I- PRESENTATION DES CONCEPTS	19
1- Le Big Data	19
2- Traitements ETL (extract transform load)	21
3- Cluster de serveurs	24
4- Conteneurisation	26
5- Devops	29
6- Cloud computing	32



II- TYPE D'ARCHITECTURE BIG DATA	37
1- Architecture Lambda	38
2- Architecture Kappa	39
PARTIE III: ARCHITECTURE BIG DATA DE LA PLATEFORME D'ANALYSE DES RÉSEAUX SOCIAUX	41
I- DESCRIPTION DÉTAILLÉE DE L'ARCHITECTURE DE LA SOLUTION	42
1- Collecte des données depuis les réseaux sociaux	42
2- Analyse et traitements des données	44
4- Couche de Visualisation	46
II- ETUDE COMPARATIVE, CHOIX DES OUTILS TECHNIQUES ET DESCRIPTION DE L'APPROCHE	46
1- Environnement	46
2- Application de collecte des données	47
3- Système d'ingestion de données	49
4- Outils d'ETL	51
5- Type de Cluster	54
6- Système d'orchestration	56
7- Data Lake, Data Warehouse	60
8- Outils de visualisation	62
PARTIE IV: MISE EN PLACE ET DÉPLOIEMENT DE LA SOLUTION BIG DATA/CLOUD	64
I- PRÉPARATION DE L'ENVIRONNEMENT	65
1- Environnement Azure	65
2- Environnement local	68
II- DÉPLOIEMENT DES SERVICES DANS AZURE	69
1- Préparation des Azure Resource Manager (ARM) Templates	69
2 - Préparation des pipelines d'industrialisation	70
3- Déploiement de la plateforme	71
CONCLUSION GENERALE	78
BIBLIOGRAPHIE & WEBOGRAPHIE	79
TABLE DES MATIERES	80

