



Ministère de l'Enseignement Supérieur, de la Recherche et de l'Innovation
ECOLE POLYTECHNIQUE DE THIES



DEPARTEMENT GENIE INFORMATIQUE ET TÉLÉCOMMUNICATIONS

Projet de Fin d'Études en vue de l'obtention du Diplôme d'Ingénieur de Conception

Présenté par : Papa Omar DIOP

Sujet:

Conception et Réalisation d'un système de détection et de gestion des anomalies liées aux dysfonctionnements des applications en production

Soutenu publiquement le 12 Novembre 2022 devant le jury composé de :

| Président | Dr Ahmed M. WADE | Maître Assistant | PER GIT-EPT |
|--------------|-----------------------|------------------------------------|-------------------|
| Examinateurs | M. Mame Demba CISSE | PhD, Chef Recherche & Innovation | Modelsis |
| | Mme Hawa DIA | Ingénieur, IT Consultant | FTI, France |
| | M. Mouhammad DIAKHATE | Ingénieur, Data Engineer | Capgemini, France |
| Encadreurs | M. Sadibou SOW | Ingénieur, Chief Operating Officer | TeamX |
| | Dr Michel SECK | Assistant | PER GIT-EPT |

PRÉFACE

DEDICACES

Par la grâce de Dieu le tout puissant, le Miséricordieux, je dédie ce modeste travail à :

À mes très chers parents

Quoi que je fasse ou que je dise, je ne saurai point les remercier comme il se doit. Votre affection me couvre, votre bienveillance me guide et votre présence à mes côtés a toujours été ma source de force pour affronter les différents obstacles.

À ma très chère grande mère

Tu as toujours été à mes côtés pour me soutenir et m'encourager.
Que ce travail traduit ma gratitude et mon affection.

À Serigne Ndiaye DIOP, Momar DIOP et Mouhamed DIOP

Puisse Dieu vous donner santé, bonheur, courage et surtout réussite

À toute ma famille,

Pour leur soutien et leur aide durant tout mon cursus scolaire.

À tous mes amis,

En souvenir des meilleurs moments que nous avons passés ensemble.

À toute la famille polytechnicienne,

Pour ces beaux moments durant ces cinq années et plus particulièrement à mes promotionnaires

REMERCIEMENTS

Je commence par rendre grâce à Allah (swt) pour tous ses bienfaits qui m'ont permis d'entamer et de terminer ce mémoire et je prie sur son Prophète Mouhammed (psl) ainsi que sur sa famille et ses compagnons.

S'il faut beaucoup de motivation, de rigueur et d'enthousiasme pour mener à bien ce mémoire, alors, ce travail de recherche a eu besoin de la contribution de plusieurs personnes, que je tiens à remercier.

Je commence par l'ensemble des professeurs, en particulier ceux du département Génie Informatique et Télécommunications (GIT) ainsi que le cadre administratif de l'École Polytechnique de Thiès, qui durant ma formation m'ont transmis toutes les connaissances nécessaires à la réussite de mes études.

Je remercie plus spécialement, mon encadrant académique Monsieur Michel SECK, professeur à l'École Polytechnique de Thiès, qui a guidé mon mémoire, je le remercie aussi pour son soutien constant, sa confiance et surtout ses judicieux conseils, qui ont contribué à alimenter ma réflexion.

Monsieur Aboul Aziz CISS, une référence vraiment, pour son aide, sa patience et sa disponibilité.

Je remercie également toute l'équipe professionnelle de TeamX, les intervenants responsables de ma formation qui ont assuré la partie pratique de celle-ci et contribuant au succès de mon stage.

Monsieur Abass SARR qui m'a beaucoup appris sur les défis à relever dans le monde de la recherche et du développement (R&D). Il a partagé ses connaissances et expériences dans ce milieu, tout en m'accordant sa confiance et une large indépendance dans l'exécution de missions valorisantes.

Messieurs Papa Birame SENE, Sadibou SOW et Babacar TOURÉ, pour m'avoir accordé des entretiens et avoir répondu à mes questions sur la culture du monde des affaires ainsi que leur expérience personnelle. Ils ont été d'un grand soutien dans l'élaboration de ce mémoire.

Mon promotionnaire Papa Modoulaye DIOP, pour avoir relu et corrigé mon mémoire. Ses conseils de rédaction ont été très précieux.

Je ne saurais terminer sans remercier tous les membres de ma famille, mes amis et proches ainsi que toutes les personnes qui m'ont soutenu.

RÉSUMÉ

Quand il s'agit de la détection d'anomalies, combien de fois on a pu entendre un collègue dire si tu rencontres tel ou tel bug fais ceci ou bien il y a seulement le lead qui connaît comment résoudre le problème. Ne pas documenter, normaliser ou partager la façon d'analyser les données pour détecter les intrusions potentielles dans un réseau est plus courant qu'on ne le pense, surtout lorsqu'une équipe est très diverse du point de vue technique et de l'expertise. Cela n'affecte pas seulement les stratégies de détection mais aussi la dynamique de l'équipe ainsi que le bon avancement des projets.

Maintenant, combien de fois avons-nous également pensé à un moyen plus efficace, intuitif ou créatif d'analyser les événements de sécurité que notre organisation collecte, et que nous nous sentons limité aux capacités d'une barre de recherche dépendant d'une seule langue ?

Dans le développement de logiciels, il est absolument nécessaire de s'assurer qu'un système, une fois développé, fonctionne au mieux pendant toute sa durée de vie. Avec le temps, il génère des ressources de données à grande échelle, impliquant une grande quantité d'informations de logs dont le contenu est très significatif. Les données des journaux d'applications sont essentielles au maintien des performances des applications. Les techniques d'analyse, de compréhension et de détection des anomalies dans les journaux d'applications sont donc essentielles pour garantir l'efficacité du développement logiciel. Bien qu'initialement entravées par un matériel limité et un manque d'ensembles de données de qualité, les techniques de détection d'anomalies ont récemment bénéficié d'un regain d'intérêt grâce aux progrès de la technologie d'apprentissage automatique et notamment des réseaux neuronaux.

Dans cette étude, nous explorons la détection d'anomalies, les techniques historiques de détection d'anomalies et les progrès récents des méthodes d'apprentissage qui promettent de révolutionner la détection d'anomalies dans les données des journaux d'applications. En outre, nous analysons les techniques de détection d'anomalies les plus prometteuses et proposons un modèle basé principalement sur les méthodes d'apprentissage automatique, qui améliore les techniques existantes.

Nous verrons par la suite que les résultats de l'évaluation sont implémentés à travers un système d'alertes qui répond aux besoins de nos applications actuellement en production. Ce qui améliore considérablement la précision de la fiabilité en temps réel et fournit ainsi une base de données précise et une base de localisation des anomalies pour l'intelligence artificielle pour les opérations informatiques. L'ensemble du processus a été établi depuis le journal original jusqu'à la mesure de la fiabilité en temps réel. De nouvelles méthodes ont été adoptées pour améliorer la précision, le rappel et la valeur F1 de la détection des anomalies. En outre, comme il s'agit d'une approche axée sur l'apprentissage, il est possible de mettre à jour le modèle de manière incrémentielle afin qu'il puisse s'adapter aux nouveaux modèles de journaux qui apparaissent au fil du temps.

Termes clés – AIOps, Détection d'anomalies, journaux d'applications, apprentissage automatique, Big Data.

ABSTRACT

When it comes to anomaly detection, how many times have we heard a colleague say that if you encounter such-and-such a bug, do this or that, or that only the leader knows how to fix the problem. Not documenting, standardizing, or sharing how to analyze data to detect potential network intrusions is more common than you might think, especially when a team is very diverse in terms of technical expertise. This affects not only detection strategies but also team dynamics and project progress.

How many times have we thought of a more efficient, intuitive, or creative way to analyze the security events collected by our organization, but we feel limited to the capabilities of a single language-dependent search bar?

In software development, it is absolutely necessary to ensure that a system, once developed, works at its best throughout its lifetime. Over time, it generates large-scale data resources involving a large amount of log information, the content of which is very important. Application log data is critical to maintaining application performance. Therefore, techniques for analyzing, understanding, and detecting anomalies in application logs are critical to effective software development. Although initially hampered by limited hardware and a lack of quality datasets, anomaly detection techniques have recently received renewed interest due to advances in machine learning technology, particularly neural networks.

In this paper, we explore anomaly detection, historical anomaly detection techniques, and recent advances in learning methods, which promise to revolutionize anomaly detection in application log data. In addition, we analyze the most promising anomaly detection techniques and propose a model based primarily on machine learning methods that improve on existing techniques.

We will then see that the evaluation results are implemented through an alerting system that meets the needs of our applications currently in production. This significantly improves the accuracy of real-time reliability and thus provides an accurate database and anomaly location basis for artificial intelligence for IT operations. The entire process has been established from the original log to the real-time reliability measurement. New methods were adopted to improve the accuracy, recall, and F1 value of anomaly detection. In addition, because this is a learning approach, the model can be incrementally updated to account for new log patterns that emerge over time.

Index Terms – AIOps, Anomaly Detection, Application logs, Machine Learning, Big Data.

TABLE DES MATIERES

Résumé

Abstract

Liste des Figures

Liste des Tableaux

Liste des Sigles et Abréviations

Chapitre 1 : Introduction générale

| | | |
|-------------|--|----|
| I. | Présentation de la structure d'accueil | 14 |
| 1. | Contexte de l'immersion | 14 |
| 2. | Structure d'accueil..... | 14 |
| 3. | Concept de strartup studio | 15 |
| 4. | Les quelques partenaires..... | 16 |
| II. | Contexte et Problématique | 17 |
| III. | Objectifs du projet | 18 |
| IV. | Annonce du plan..... | 18 |

Chapitre 2 : Généralités sur la détection d'anomalies

| | | |
|-------------|--|----|
| I. | Mise en contexte..... | 19 |
| 1. | Logs | 19 |
| 2. | Anomalies..... | 19 |
| II. | Les types d'anomalies | 19 |
| 1. | Les anomalies ponctuelles | 19 |
| 2. | Les anomalies contextuelles | 20 |
| 3. | Les anomalies collectives | 21 |
| III. | Les challenges | 22 |
| 1. | Données non structurées..... | 22 |
| 2. | Informations d'exécution redondantes | 22 |
| 3. | Données déséquilibrées | 22 |

Chapitre 3 : État de l'art

| | | |
|------------|--|----|
| I. | Détection d'anomalies basée sur la statistique | 23 |
| II. | Détection d'anomalies basée sur la profondeur..... | 23 |

| | | |
|-------------|--|----|
| III. | Détection d'anomalies basée sur l'apprentissage automatique..... | 23 |
| 1. | Les algorithmes de détection | 23 |
| 2. | Méthode basée sur la décomposition spectrale..... | 27 |
| 3. | Méthode basée sur l'apprentissage supervisé | 27 |
| 4. | Méthode basée sur l'apprentissage non supervisé | 27 |
| 4.1 | La distance..... | 27 |
| 4.2 | La densité..... | 27 |
| 4.3 | Le regroupement ou clustering | 28 |
| 4.4 | Travaux antérieurs..... | 28 |
| 5. | Modèles basés sur les réseaux neuronaux | 29 |
| 5.1 | Détection d'anomalies avec le réseau de neurones LSTM | 30 |
| 5.2 | Détection d'anomalies avec l'encodeur automatique | 30 |
| 5.3 | Travaux antérieurs..... | 31 |
| IV. | Synthèse | 31 |

Chapitre 4 : Étude conceptuelle de la solution

| | | |
|--------------|---|----|
| I. | Analyse des besoins..... | 33 |
| 1. | Les besoins fonctionnels..... | 33 |
| 2. | Les besoins non fonctionnels..... | 33 |
| II. | Modélisation UML | 33 |
| 1. | Diagramme des cas d'utilisation (Uses Cases)..... | 34 |
| 2. | Diagramme des classes..... | 34 |
| III. | Architecture du projet..... | 35 |
| IV. | Collecte de données | 37 |
| V. | Description du jeu de données..... | 38 |
| VI. | Prétraitement des données | 39 |
| VII. | Algorithme | 42 |
| VIII. | Analyse et évaluation des résultats..... | 47 |

Chapitre 5 : Implémentation et présentation de la solution

| | | |
|-----------|--|----|
| I. | Environnement de développement et technologies | 49 |
| 1. | Docker | 49 |
| 1.1 | Containers and Virtual Machines | 49 |
| 1.2 | Images | 50 |
| 1.3 | Containers Lifecycle..... | 51 |
| 1.4 | Networking | 52 |
| 1.5 | Docker Compose | 52 |

| | | |
|------------|---|-----------|
| 2. | Apache Kafka | 53 |
| 2.1 | Kafka architecture | 53 |
| 2.2 | Topic structure | 54 |
| 2.3 | Clustered deployment | 54 |
| 2.4 | Parallel processing of the topic..... | 55 |
| 3. | Apache Zookeeper..... | 56 |
| 4. | Apache Spark | 56 |
| 4.1 | Challenges in big data processing..... | 57 |
| 4.2 | Spark cluster | 57 |
| 4.3 | Mleap | 58 |
| 5. | Logstash | 59 |
| 5.1 | Data extraction..... | 60 |
| 5.2 | Data transformation..... | 60 |
| 5.3 | Data load..... | 61 |
| 5.4 | Parallel processing..... | 61 |
| 5.5 | Grok | 61 |
| 6. | Elasticsearch..... | 62 |
| 6.1 | Elasticsearch architecture..... | 62 |
| 6.2 | Difference from a database | 62 |
| 6.3 | Clustered deployment | 63 |
| 6.4 | Search and aggregation..... | 64 |
| 7. | Kibana | 64 |
| 7.1 | Widget and Dashboard | 64 |
| 7.2 | Discover..... | 65 |
| 7.3 | ELK stack monitor and management..... | 65 |
| 8. | Beats | 66 |
| 8.1 | Filebeat and Metricbeat | 66 |
| 9. | Lucidchart..... | 66 |
| 10. | Technologies | 67 |
| 11. | Summary | 69 |
| II. | Présentation de la solution..... | 69 |

Chapitre 6 : Conclusion Générale

| | | |
|-------------|-----------------------------------|-----------|
| I. | Synthèse des travaux | 75 |
| II. | Apport du stage | 75 |
| III. | Perspectives | 76 |

Références

LISTE DES FIGURES

| | |
|---|----|
| Figure 1.1 – Logo de l'entreprise | 15 |
| Figure 1.2 – Concept de startup studio | 15 |
| Figure 1.3 – Logo de Logidoo..... | 16 |
| Figure 1.4 – Logo de Xaalys..... | 16 |
| Figure 1.5 – Logo de DER..... | 17 |
| Figure 2.1 – Un journal de test | 19 |
| Figure 2.2 – Anomalie ponctuelle | 20 |
| Figure 2.3 – Anomalie contextuelle | 21 |
| Figure 2.4 – Électrocardiogramme | 21 |
| Figure 2.5 – Journal avec des données redondantes | 22 |
| Figure 3.1 – Construction d'iForest pour l'ensemble des données | 24 |
| Figure 3.2 – Un exemple visuel d'un simple SVM à une classe | 24 |
| Figure 3.3 – Schéma conceptuel du clustering k-means | 25 |
| Figure 3.4 – Détection d'anomalies basée sur le clustering | 28 |
| Figure 3.5 – Comparaison de l'architecture d'un RNN et d'un réseau neuronal direct | 29 |
| Figure 3.6 – Architecture LSTM | 30 |
| Figure 3.7 – Apprentissage de l'encodeur automatique | 31 |
| Figure 3.8 – Techniques pour la détection d'anomalies | 32 |
| Figure 4.1 – Diagramme des cas d'utilisation | 34 |
| Figure 4.2 – Diagramme des classes pour l'application streaming | 34 |
| Figure 4.3 – Architecture du pipeline | 36 |
| Figure 4.4 – Format donné pour les logs | 38 |
| Figure 4.5 – Récupération des données sur Elasticsearch | 38 |
| Figure 4.6 – Flux de mise en œuvre de l'étude | 40 |
| Figure 4.7 – Première étape du prétraitement | 40 |
| Figure 4.8 – Inputs pour l'algorithme de prédiction | 42 |
| Figure 4.9 – Workflow Algorithme ML | 43 |
| Figure 4.10 – Variation du coût en fonction du nombre de clusters k | 44 |
| Figure 4.11 – Décompte des anomalies obtenues avec le k-means | 44 |
| Figure 4.12 – Résultats obtenus avec le k-means | 44 |
| Figure 4.13 – Cross validation avec le one class svm | 45 |
| Figure 4.14 – Décompte des anomalies obtenues avec le one class svm | 45 |
| Figure 4.15 – Résultats obtenus avec le one class svm | 45 |
| Figure 4.16 – Représentation graphique des anomalies obtenues avec le one class svm | 46 |

| | |
|---|----|
| Figure 4.17 – Cross validation avec l’isolation forest | 46 |
| Figure 4.18 – Décompte des anomalies obtenues avec l’isolation forest | 46 |
| Figure 4.19 – Résultats obtenus avec l’isolation forest | 47 |
| Figure 4.20 – Performance des algos | 47 |
| Figure 5.1 – Architecture de Docker en comparaison avec la virtualisation classique | 50 |
| Figure 5.2 – Cycle de vie des conteneurs Docker | 51 |
| Figure 5.3 – Architecture de Apache Kafka | 54 |
| Figure 5.4 – Déploiement en cluster d’Apache Kafka..... | 55 |
| Figure 5.5 – Logo Apache ZooKeeper | 56 |
| Figure 5.6 – Architecture du cluster Spark | 58 |
| Figure 5.7 – Spark, Scikit-learn, and TensorFlow: One Runtime | 59 |
| Figure 5.8 – Logstash architecture | 60 |
| Figure 5.9 – Logo Logstash Grok | 61 |
| Figure 5.10 – Elasticsearch Cluster | 63 |
| Figure 5.11 – Exemple Dashboard Kibana | 65 |
| Figure 5.12 – Quelques graphes modélisés avec l’outil | 67 |
| Figure 5.13 – Logo Jupyter Notebook..... | 67 |
| Figure 5.14 – Logo Python | 68 |
| Figure 5.15 – Logo Scala | 68 |
| Figure 5.16 – Logo Maven | 69 |
| Figure 5.17 – Logo Sbt | 69 |
| Figure 5.18 – Filebeat Configuration | 70 |
| Figure 5.19 – Page de gestion des index..... | 70 |
| Figure 5.20 – Données ingérées depuis le serveur | 71 |
| Figure 5.21 – Workflow Application Streaming..... | 72 |
| Figure 5.22 – Modèle bundle généré depuis Spark | 72 |
| Figure 5.23 – Application Streaming | 73 |
| Figure 5.24 – Outputs algorithme de détection | 74 |
| Figure 5.25 – Output des résultats sur Slack..... | 74 |

LISTE DES TABLEAUX

Tableau 4.1 – Description du jeu de données l’entreprise 39

LISTES DES ABBRÉVIATIONS

| | |
|--------------|---|
| EPT | École Polytechnique de Thiès |
| GIT | Génie Informatique et Télécommunications |
| DER | Délégation Générale à l'Entreprenariat Rapide |
| SI | Systèmes d'Informations |
| ML | Machine Learning |
| AIOps | Artificial Intelligence Operations System |
| ECG | Électrocardiogramme |
| LSTM | Long Term Surveillance and Maintenance |
| RNN | Réseaux Neuronaux Récurrents |
| API | Application Programming Interface |
| PCA | Principal Component Analysis |
| SVM | Support Vector Machine |
| UML | Unified Modeling Language |
| VM | Virtual Machine |
| IP | Internet Protocol |
| NAT | Network Address Translation |
| SQL | Structured Query Language |
| XML | Extensible Markup Language |
| JSON | JavaScript Object Notation |
| HTTP | Hypertext Transfer Protocol |
| REST | Representational State Transfer |
| ETL | Extract, Transform, Load |

INTRODUCTION GÉNÉRALE

Pendant longtemps, dans le secteur de l'informatique et des télécommunications, la quasi-totalité des tâches dans les entreprises se faisaient de manière manuelle. En effet, grâce à une ressource humaine diversifiée et qualifiée, les entreprises réussissaient à réaliser ces tâches et proposer des produits et services de qualité à leurs clients. Cela leur permettait non seulement de pouvoir détecter des événements indésirables dans leurs systèmes, mais aussi de pouvoir maintenir la qualité de leurs services afin de satisfaire leurs clients, rester en vie et faire face à la concurrence. Cependant, la plupart du temps, réaliser ces tâches nécessitait beaucoup de temps et d'énergie. Beaucoup de temps s'écoulait entre la détection d'un incident dans un système et sa résolution. Pour résoudre un incident, il faut faire beaucoup d'efforts de recherches, mobiliser beaucoup de ressources humaines et matérielles et cela peut avoir un impact négatif sur la productivité, mais aussi sur le ressenti du client. De plus, avec l'explosion des données depuis quelques temps (Big Data), l'un des défis majeurs des entreprises est de savoir comment exploiter ces données et les gérer afin de pouvoir les traiter, les utiliser et d'en tirer le maximum. Ils se sont alors tournés vers un nouvel outil qui leur permet, avec les quantités de données énormes qu'ils récupèrent combinées aux nouvelles branches de l'informatique, de réaliser ces tâches beaucoup plus efficacement afin d'attirer le maximum de clients. C'est là le début de la naissance d'une nouvelle branche de l'informatique : AIOps. Il s'agit d'une combinaison du Big Data et de l'intelligence artificielle, plus particulièrement de l'apprentissage automatique, afin d'automatiser certaines tâches au niveau des SI (systèmes d'information).

Ainsi, cette première partie constitue une introduction générale où nous allons dans un premier temps définir le contexte du stage, son importance et ses finalités. Dans un second temps dégager la problématique, présenter le projet ainsi que ses objectifs. Enfin le point final de ce chapitre nous donnera l'occasion de nous appesantir sur le plan structuré de ce rapport.

I. Présentation de la structure d'accueil

1. Contexte de l'immersion

La formation au sein du génie informatique de l'E.P.T. allie les côtés théorique et pratique. Ce qui exige aux étudiants de faire des immersions dans les entreprises pour consolider les connaissances acquises en classe. Ainsi, la dernière année de formation est surtout remplie par des stages d'une durée minimum de six (5) mois. Ces stages permettront aux étudiants de travailler sur un sujet proposé par l'entreprise hôte en concertation avec le service académique. Ce projet, dont l'objectif est l'obtention du diplôme d'ingénieur de conception en Informatique et Télécommunication, s'inscrit dans le cadre de ces immersions pédagogiques qui ont eu lieu dans la période du 31 janvier 2022 au 15 juin 2022.

2. Structure d'accueil

TeamX Group est une startup créée en 2018 par Papa Birame SENE. Elle s'est fixée comme objectifs : la création et le développement de startups sur la base d'idées générées en interne, mais aussi l'accompagnement des entrepreneurs dans la concrétisation de leur projets à travers un apport technique et parfois même financier. TeamX a comme vision de mettre l'Afrique au cœur de la révolution technologique grâce au succès de ses startups.



Figure 1.1 : Logo de l'entreprise

2.1 Concept de startup studio

Dans le monde des structures qui accompagnent les startups, il y a les accélérateurs et les incubateurs qui sont les plus connus. Mais le problème avec ces deniers c'est qu'ils ne font qu'office de guide et de conseiller pour les startupeurs. Alors que le souci majeur des startups est que leur chance de survie au démarrage des activités est très faible dû à un manque de partenaire stratégique et opérationnel mais aussi de financement. Parties de ce constat, deux startups, Rocket Internet à Berlin et Betaworks à New-York, décidèrent de créer un nouveau model d'accompagnement des startups. On assiste à la naissance du concept «startup studio». Un startup studio est une structure dont le but est de créer des startups destinés à devenir des entreprises. Grâce à leurs infrastructures et à leurs ressources mis à disposition, les startups studio augmentent les chances de réussite d'une startup et optimisent sa création et sa croissance. Le startup studio devient donc un partenaire stratégique, opérationnel et financier de ses startups partenaires. Ci dessous un schéma illustratif du degré d'implication du startup studio par rapport aux autres accompagnateurs :

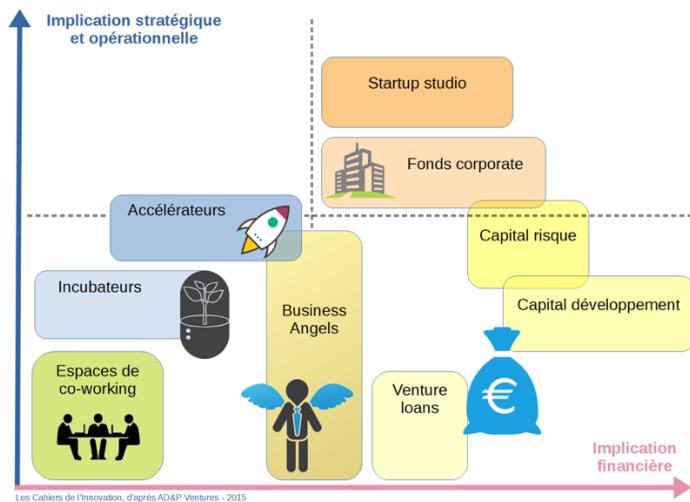


Figure 1.2: Concept de startup studio

Créer des entreprises à travers un startup studio entraîne de nombreux avantages provenant de l'infrastructure du studio parmi lesquels:

- Développement rapide : grâce à des processus judicieux et à une équipe dévouée d'experts qui travaillent au quotidien pour aider l'entreprise à croître.
- Mieux créer : l'équipe du studio est composée d'experts dans leurs domaines (design, marketing, code, etc.) qui travaillent ensemble. Le recrutement de profils élevés est facilité par la bonne réputation du studio et l'assistance de l'équipe. Le réseau du studio dans son ensemble est un réel avantage pour aider une startup à se développer.

Si un startup studio veut survivre, dans l'écosystème des accompagnateurs de startups il lui faut des partenaires. TeamX Group, en seulement 4 ans d'existence a noué plusieurs partenariats. Nous ferons une brève présentation de ces partenaires dans la sous-section suivante.

2.2 Les quelques partenaires

La startup TeamX Group, qui se veut studio technologique, accompagne principalement ses partenaires sur le plan technologique. Ci-dessous la liste des startups avec lesquelles, TeamX a noué un partenariat :

- **Logidoo**

Logidoo est une marketplace de prestation logistiques regroupant des bourses de fret et un ensemble de prestations rattachées.



Figure 1.3: Logo de Logidoo

- **Xaalys**

Xaalys est une néo-banque française dédiée aux adolescents de 12 à 17 ans créée en avril 2017 par Diana Brondel. Selon la fondatrice : « Xaalys se trouve à la frontière des Fintech (startups de la finance) et des Edtech (startups de l'éducation). La société opère entre Paris et Dakar avec une équipe technique de développement basée à Dakar (TeamX Group) et une équipe opérationnelle à Paris pour accompagner les clients.



Figure 1.4: Logo de Xaalys

▪ DER

La Délégation générale à l'Entreprenariat Rapide des Femmes et des Jeunes est une structure mise en place par le Président de la République. Crée dans la perspective d'asseoir de façon durable un cadre catalyseur pour la réalisation de performances économiques signifiantes et ressenties pleinement par les populations, la DER inscrit son intervention dans le cadre des efforts consentis et actions déployées par l'Etat du Sénégal et ses différents partenaires. Sa mission principale consiste à contribuer à dynamiser l'entreprenariat qui constitue la principale occupation de la population sénégalaise.



Figure 1.5: Logo de la DER

II. Contexte et Problématique

L'application des méthodologies d'apprentissage machine et de réseaux neuronaux aux opérations informatiques (AIOps), grâce au nombre croissant de frameworks disponibles, souvent open-source, devient omniprésente dans de nombreux domaines. De plus, lénorme disponibilité de données d'entraînement, due à la grande capacité de stockage des centres de données et aux systèmes modernes de collecte de données, permet de développer des modèles d'apprentissage automatique plus précis et moins sujets aux erreurs.

Au sein de TeamX, les technologies utilisées dans les applications sont à peu près similaires et ces dernières génèrent des quantités massives de données de journal, des données horodatées produites automatiquement qui représentent chaque événement système et utilisateur pour tous les utilisateurs de l'application. Ces données sont générées tout au long de la durée de vie d'une application et ont tendance à être des séries temporelles, non structurées, textuelles, mal formatées et sont générées à un rythme incroyable à mesure que l'application évolue et ajoute des utilisateurs. Tout le processus de débogage nécessite donc qu'un développeur ou un ingénieur de support analyse ces données manuellement, en les lisant ligne par ligne jusqu'à ce qu'une anomalie soit localisée. Les détails et les horodatages de l'anomalie offrent un point de départ pour découvrir quand, comment et où les erreurs dans l'application se sont produites.

En raison des demandes croissantes, l'entreprise est obligée de déployer un nombre croissant d'ingénieurs pour gérer les données des journaux, et même dans ce cas, un grand nombre d'anomalies telles que les messages d'erreur, les notifications d'avertissement et les tentatives d'intrusion dans le réseau ne sont pas détectées. Une solution possible pour aborder la détection des anomalies dans les journaux d'applications est de détecter de manière autonome les anomalies dans les données des journaux d'applications.

III. Objectifs du projet

Pour incorporer les exigences susmentionnées dans notre cadre, nous proposons une architecture logicielle en pipeline supportant l'application de modèles ML dans un environnement opérationnel d'une manière évolutive et tolérante aux pannes et permettant le traitement en temps quasi réel de données en continu. La conception proposée peut prendre en charge jusqu'à des milliers de sources de données dans un environnement réel et imprévisible et est construite à l'aide de modules logiciels de pointe et de sources ouvertes qui sont enchaînés pour former un pipeline logiciel. Tous les composants prennent en charge le déploiement distribué et sont conformes aux meilleures pratiques de sécurité : toutes les applications sont conçues pour fonctionner dans un cluster afin d'effectuer la distribution du travail et le traitement parallèle et elles incluent le cryptage des données échangées et l'authentification entre les parties. L'architecture de pipeline proposée est composée de quatre parties :

- Ingestion de données : Applications qui prennent en charge l'ingestion des données dans le système. Elle comporte trois phases : mise en mémoire tampon, prétraitement et stockage. C'est le point d'entrée des sources de données en continu.
- Application de modèle d'apprentissage automatique : les données stockées sont traitées par lots avec le modèle d'apprentissage automatique et les résultats sont stockés en retour.
- Visualisation des données : permet d'examiner les données d'entrée et les résultats du traitement des données ainsi que la prédiction dans une interface.
- Surveillance de la santé : surveille l'état des applications du pipeline afin de détecter les défaillances logicielles et d'y réagir rapidement.

Afin de réaliser une implémentation de référence et d'avoir un cas d'utilisation réel pour tester le pipeline développé, nous considérons l'application de la ML pour la gestion des anomalies dans les projets de la DER actuellement en développement à TeamX et utilisons un modèle ML qui va effectuer la prédiction d'une panne potentielle dans l'application.

IV. Annonce du plan

Nous avons structuré le document en cinq grandes parties. Après cette introduction, la deuxième partie de ce mémoire portera sur les généralités de la détection d'anomalies. Ensuite, le troisième partie va décrire la réalisation d'une étude de l'état de l'art technique dans ce domaine. Dans la quatrième partie on va proposer une architecture appropriée pour le système. Il s'agira d'apporter une lumière sur tout ce qui est conception, à savoir la définition des besoins, la modélisation, l'architecture choisie et les algorithmes. Le point final de ce chapitre nous donnera l'occasion de, nous appesantir sur la mise en œuvre agile de la solution allant de l'implémentation jusqu'au déploiement et enfin proposer des pistes d'évolution pour la solution.

GÉNÉRALITÉ SUR LA DÉTECTION D'ANOMALIES

I. Mise en contexte

1. Logs

Les journaux des grands systèmes de données sont généralement des données non structurées imprimées en séquence temporelle. Normalement, chaque entrée de journal (ligne) peut être divisée en deux parties différentes : constante et variable. La partie constante est constituée des messages imprimés directement par les instructions du code source. Les clés de journal sont souvent extraites de ces parties constantes, où les clés de journal sont les messages constants communs à toutes les entrées de journal similaires. Un journal de test ressemble à ceci :

```
09:29:44,761 INFO          se.ericsson.jcat.fw.logging.JcatLoggingApi setTestStepBegin
09:29:44,762 INFO  com.ericsson.oml.test.ml66.ethernet.qos.complex.ComplexTest info
09:29:44,762 INFO  com.ericsson.oml.test.ml66.ethernet.qos.complex.ComplexTest info
09:29:44,762 INFO  com.ericsson.oml.test.ml66.ethernet.qos.complex.ComplexTest info
09:29:44,762 INFO  com.ericsson.oml.test.ml66.ethernet.qos.complex.ComplexTest info
09:29:44,763 DEBUG ore.internal.utilities.io.StreamlineChannel-/127.0.0.1:33102 logDebug
09:29:46,125 DEBUG ore.internal.utilities.io.StreamlineChannel-/127.0.0.1:33102 logDebug
09:29:46,129 DEBUG internal.instrument_drivers.ethernet_testers.spi.EtaPortImpl lambda$waitForLinkState$5
09:29:46,130 DEBUG ore.internal.utilities.io.StreamlineChannel-/127.0.0.1:33102 logDebug
09:29:46,361 DEBUG ore.internal.utilities.io.StreamlineChannel-/127.0.0.1:33102 logDebug
09:29:46,362 DEBUG ore.internal.utilities.io.StreamlineChannel-/127.0.0.1:33102 logDebug
09:29:47,724 DEBUG ore.internal.utilities.io.StreamlineChannel-/127.0.0.1:33102 logDebug
09:29:47,725 DEBUG internal.instrument_drivers.ethernet_testers.spi.EtaPortImpl lambda$waitForLinkState$5
09:29:47,725 DEBUG ore.internal.utilities.io.StreamlineChannel-/127.0.0.1:33102 logDebug
09:29:47,956 DEBUG ore.internal.utilities.io.StreamlineChannel-/127.0.0.1:33102 logDebug
09:29:47,974 DEBUG ore.internal.utilities.io.StreamlineChannel-/127.0.0.1:33102 logDebug
09:29:48,202 DEBUG ore.internal.utilities.io.StreamlineChannel-/127.0.0.1:33102 logDebug
09:29:48,203 DEBUG ore.internal.utilities.io.StreamlineChannel-/127.0.0.1:33102 logDebug
09:29:48,437 DEBUG ore.internal.utilities.io.StreamlineChannel-/127.0.0.1:33102 logDebug
09:29:48,438 DEBUG ore.internal.utilities.io.StreamlineChannel-/127.0.0.1:33102 logDebug
09:29:51,790 DEBUG ore.internal.utilities.io.StreamlineChannel-/127.0.0.1:33102 logDebug
09:29:51,805 DEBUG nai.instrument_daos.ethernet_testers.ixia.driver.IxiaEtaPort setStreamConfigs
    Stream 1: EtaStreamConfig{frameSize=Fixed{1024}, layer2Config=EtaLayer2Config[preambleSize=8,destinationAddress=02:00:00:00:00:12,etaDstMacAddressMode=FIXE
Test step [1] begin POLICER_PCP7
Evaluating test traffic route. Current ETA port rol
Test traffic is sent from ETA port: 1.1.1
Receiver ETA port for traffic evaluation: 1.1.2
Configuring traffic generator
wrote: {command:"PortConfig", action:"Get", card:1,
read: {command:"PortConfig", action:"Get", card:1,
read: {command:"LinkState", action:"Get", card:1,
read: {command:"LinkState", action:"Get", card:1,
wrote: {command:"PortConfig", action:"Get", card:1,
read: {command:"PortConfig", action:"Get", card:1,
read: {command:"LinkState", action:"Get", card:1,
read: {command:"LinkState", action:"Get", card:1,
wrote: {command:"Transmission", action:"GetState",
read: {command:"Transmission", action:"GetState",
wrote: {command:"LinkState", action:"Get", card:1,
read: {command:"LinkState", action:"Get", card:1,
wrote: {command:"Statistics", action:"Get", card:1,
read: {command:"Statistics", action:"Get", card:1,
read: {command:"Statistics", action:"Get", card:1,
{streams=
```

Figure 2.1: Un journal de test

Dans la figure ci-dessus, le journal commence par un horodatage suivi d'un événement de type INFO ou DEBUG. Ceci indique le type de message que l'événement génère dans le fichier journal. Après l'événement, une description de l'événement est générée, suivie des commandes et des étapes de test effectuées lors de l'exécution des tests.

2. Anomalies

Une anomalie ne cadre pas avec le reste du modèle. Le mot anomalie vient du mot grec "anomolia" qui signifie inégal ou irrégulier. Lorsque quelque chose est inhabituel par rapport aux choses qui l'entourent, on parle d'anomalie.

II. Les types d'anomalies

1. Les anomalies ponctuelles

Une anomalie ponctuelle est une donnée qui dévie de manière significative de la distribution moyenne ou normale du reste des données. Ces données sont souvent générées par un système, et la déviation significative est limitée à des points de données spécifiques et partage peu de contexte avec le reste des données moyennes ou normales.

Les anomalies ponctuelles sont les plus simples à détecter et de nombreuses techniques existent pour automatiser la détection des anomalies ponctuelles. Les anomalies ponctuelles peuvent être rapidement découvertes et corrigées, et ont donc rarement un effet néfaste significatif sur les applications. La figure 2.2 montre des exemples d'anomalies ponctuelles.

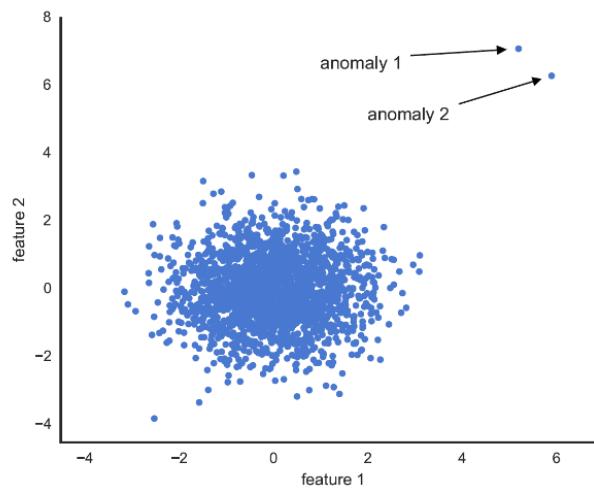


Figure 2.2 : Anomalie ponctuelle

2. Les anomalies contextuelles

Une anomalie contextuelle est identifiée comme un comportement anormal restreint à un contexte spécifique, et normal selon d'autres contextes. Ce type d'anomalie, également appelé anomalie conditionnelle, est souvent difficile à détecter car il nécessite une connaissance approfondie du domaine pour comprendre le contexte dans lequel l'anomalie se produit. La figure 2.3 montre une anomalie contextuelle dans un relevé de température. Les valeurs t1 et t2 sont identiques, mais t2 devient une anomalie une fois replacée dans son contexte : l'été.

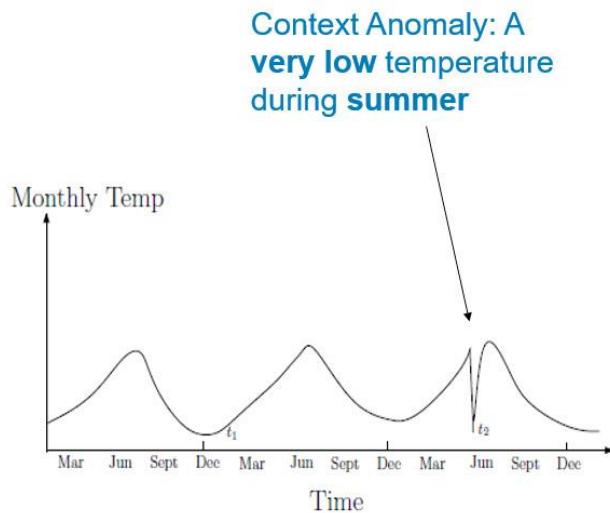


Figure 2.3 : Anomalie contextuelle

3. Les anomalies collectives

Contrairement aux anomalies contextuelles et ponctuelles, les anomalies collectives apparaissent comme un groupe de valeurs anormales dans les données. Les anomalies collectives sont des comportements anormaux d'une collection d'instances de données par rapport à l'ensemble des données. Les instances de données individuelles peuvent ne pas représenter une anomalie, mais la présence d'instances de données dans l'anomalie collective est un indicateur de comportement anormal. Il convient toutefois de noter qu'à elle seule, une instance de données ne représente pas une anomalie collective et qu'elle doit apparaître dans une collection de données pour être collectivement anormale. La figure 2.4 montre un électrocardiogramme. L'absence brève d'activité entre deux pulsations du cœur est normale. Si cette valeur se répète sur une plus longue période en revanche, c'est une anomalie.

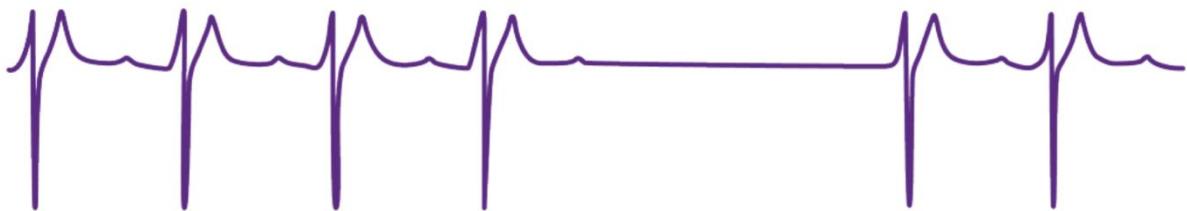


Figure 2.4 : Électrocardiogramme

III. Les challenges

Les anomalies dans les données du journal des applications sont considérées comme des modèles ou des caractéristiques qui ne suivent pas la moyenne ou le comportement normal lors d'un fonctionnement parfait. Comme le décrit Grubbs [20], "une observation aberrante, est une observation qui semble dévier de façon marquée des autres membres de l'échantillon dans lequel elle se produit". De telles anomalies peuvent être initiées par des acteurs malveillants, des bugs au niveau du système ou une utilisation incorrecte de la part de l'utilisateur et sont souvent des symptômes d'une panne ou d'une violation imminente du système. La détection d'anomalies dans les logs est particulièrement difficile, qu'elle soit automatisée ou manuelle, pour les raisons suivantes : des données non structurées, des informations d'exécution redondantes ou bien des données déséquilibrées.

1. Données non structurées

Les données non structurées sont des données qui manquent de structure ou d'architecture identifiable. Cela signifie qu'elles ne se conforment pas à un modèle de données prédefini et que, par conséquent, elles ne sont pas adaptées à un journal classique. Ce manque de structure complique l'analyse des données, qui est encore exacerbée par les formats de journalisation qui varient complètement entre les applications.

2. Informations d'exécution redondantes

Comme on peut le voir sur la Fig. 2.5, les journaux d'application contiennent des informations d'exécution telles que l'adresse IP des serveurs. Ces données changent pendant l'exécution et varient d'un serveur à l'autre ; elles sont donc redondantes pour la détection des anomalies. Comme le montre également la figure 2.5 les données des journaux d'application contiennent des données spécifiques au domaine, telles que "blockMap updated" pour les journaux qui, combinées aux informations d'exécution redondantes, augmentent la complexité de la détection des anomalies.

```
081109 204106 329 INFO dfs.DataNode$PacketResponder: PacketResponder 2 for block blk_-6670958622368987959 terminating
081109 204132 26 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.43.115:50010 is added
081109 204324 34 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.203.80:50010 is added
081109 204453 34 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.250.11.85:50010 is added t
```

Figure 2.5 : Journal avec des données redondantes

3. Données déséquilibrées

Les données du journal des applications sont conçues pour enregistrer tous les changements apportés à une application et contiennent donc des données fortement déséquilibrées en faveur d'une exécution non normale. Par exemple, des données générées par une application peut contenir moins de 5% par rapport aux événements de journal. La taille et la nature déséquilibrée des données de journal compliquent ainsi le processus de détection des anomalies.

ÉTAT DE L'ART

Les techniques de détection des anomalies ont évolué avec l'avènement du big data et de l'apprentissage automatique. Initialement abordée à l'aide de techniques statistiques, la détection d'anomalies est rapidement devenue un domaine à part entière englobant des approches statistiques, de profondeur, de distance, d'apprentissage automatique et de réseaux neuronaux.

I. Détection d'anomalies basée sur la statistique

Afin de tirer parti de la détection statistique des anomalies, l'ensemble des données du journal est organisé en fonction de sa distribution statistique globale et les points de données qui se distinguent ou ne sont pas conformes à cette distribution sont supprimés ou examinés. Ces approches sont simples à mettre en œuvre mais sont compliquées par des définitions des anomalies qui changent constamment dans différents domaines. Une transaction d'un million de dollars serait anormale pour les applications de finance personnelle mais pas pour les applications de banque d'investissement. Ainsi, ces approches nécessitent une connaissance préalable de l'ensemble de données sans laquelle la détection d'anomalies contextuelles ou collectives peut être incroyablement difficile, en particulier pour les données de journal d'application qui varient d'une application à l'autre.

II. Détection d'anomalies basée sur la profondeur

L'approche basée sur la profondeur contourne l'obligation d'organiser les données selon leur distribution statistique et exploite plutôt les coques convexes et les objets drapeaux pour calculer les anomalies dans les couches les plus externes. Cette approche nécessitant toutefois des calculs lourds, est incapable de détecter les anomalies contextuelles et ne convient pas aux ensembles de données volumineux et rapides tels que les journaux d'application.

III. Détection d'anomalies basée sur l'apprentissage automatique (ML)

L'apprentissage automatique est une forme d'intelligence artificielle qui donne aux ordinateurs la capacité d'apprendre sans être spécifiquement programmés. Il se concentre sur la création de programmes informatiques qui sont susceptibles de changer lorsqu'ils sont exposés à de nouvelles données. Il peut être classé comme supervisé ou non supervisé. Il consiste à utiliser les bonnes caractéristiques pour construire les bons modèles qui accomplissent les bonnes tâches. Ces tâches comprennent la classification binaire et multi-classes, le regroupement par régression et la modélisation descriptive.

1. Les algorithmes de détection

1.1 Isolation forest

L'isolation forest est une méthode non supervisée basée sur la construction d'arbres. L'idée derrière cet algorithme c'est qu'une donnée atypique sera plus facile à isoler qu'une donnée standard. Cet algorithme calcule, pour chaque donnée du jeu, un score d'anomalie, c'est-à-dire une mesure qui reflète à quel point la donnée en question est atypique. Afin de calculer ce score,

l'algorithme isole la donnée en question de manière récursive : il choisit un descripteur et un “seuil de coupure” au hasard, puis il évalue si cela permet d’isoler la donnée en question ; si tel est le cas, l'algorithme s'arrête, sinon il choisit un autre descripteur et un autre point de coupure au hasard, et ainsi de suite jusqu'à ce que la donnée soit isolée du reste. Au final, l'algorithme classe les données suivant leur score d'anomalie. Un score > 0.5 témoigne d'une anomalie.

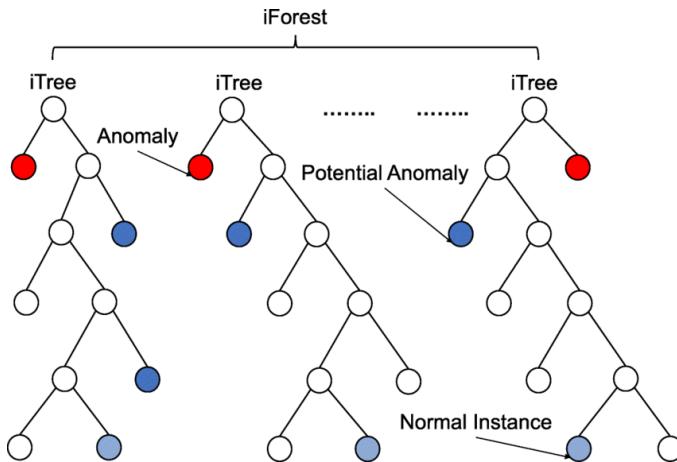


Figure 3.1 : Construction d'iForest pour l'ensemble des données

1.2 One Class SVM

Le SVM à classe unique est une variante du SVM qui peut être utilisée dans un cadre non supervisé pour la détection des anomalies. Lors de la modélisation d'une classe, l'algorithme capture la densité de la classe majoritaire et classe les exemples situés aux extrêmes de la fonction de densité comme des valeurs aberrantes. Cette modification du SVM est appelée SVM à une classe.

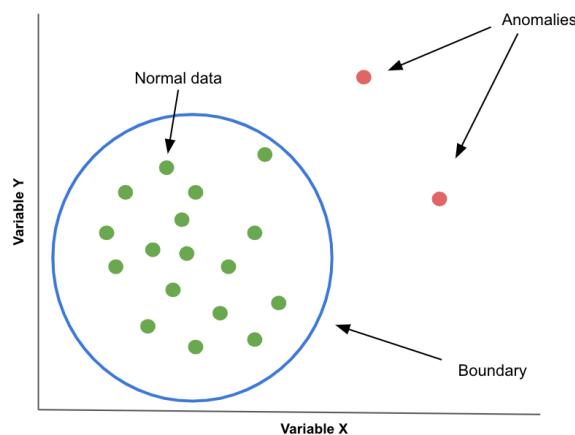


Figure 3.2 : Un exemple visuel d'un simple SVM à une classe

1.3 K-Means

Le clustering K-means peut être une méthode de quantification vectorielle, issue initialement du traitement du signal, qui vise à partitionner n observations en k clusters au cours desquels chaque observation appartient au cluster dont la moyenne est la plus proche, servant de prototype du cluster. Cet algorithme regroupe les données en essayant de séparer les échantillons en n groupes de variances égales, en minimisant un critère appelé somme des carrés d'inertie ou intra-groupe. Il nécessite que le nombre de clusters soit spécifié. Cet algorithme divise un ensemble de N échantillons X en K grappes disjointes C , chacune décrite par la moyenne des échantillons de la grappe. Les moyennes sont communément appelées les "centroïdes" des clusters. Cet algorithme vise à choisir des centroïdes qui minimisent l'inertie, ou le critère de la somme des carrés à l'intérieur d'une grappe. L'inertie est la mesure de la cohérence interne des clusters.

K-means est souvent appelé l'algorithme de Lloyd. En termes simples, l'algorithme comporte trois étapes. La première étape choisit les centroïdes initiaux, la méthode la plus basique étant de strier sur k échantillons de l'ensemble de données X . Après l'initialisation, K-means consiste à boucler entre les deux autres étapes. La première étape assigne chaque échantillon à son centroïde le plus proche. La deuxième étape crée de nouveaux centroïdes en prenant la moyenne de tous les échantillons affectés à chaque centroïde précédent. La différence entre l'ancien et le nouveau centroïde est calculée, et l'algorithme répète ces deux dernières étapes jusqu'à ce que cette valeur soit inférieure à un seuil. En d'autres termes, il répète jusqu'à ce que les centroïdes ne se déplacent plus de manière significative.

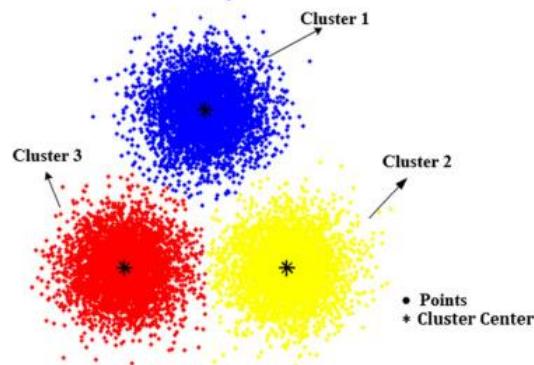


Figure 3.3 : Schéma conceptuel du clustering k-means

1.4 PCA (Principal Component Analysis)

L'ACP est une méthode statistique qui permet de saisir des modèles dans des données hautement dimensionnelles en choisissant automatiquement un ensemble de coordonnées les composantes principales qui reflètent la covariation entre les coordonnées originales. Nous utilisons l'ACP pour séparer les événements répétitifs dans les vecteurs de caractéristiques, ce qui facilite la détection des modèles de messages anormaux. L'ACP a un temps d'exécution linéaire par rapport au nombre de vecteurs de caractéristiques, de sorte que la détection d'anomalies peut s'étendre à des fichiers journaux volumineux.

L'ACP est l'une des techniques les plus couramment utilisées en analyse multivariée pour la réduction de la dimension et l'extraction de caractéristiques, et elle est particulièrement bien adaptée aux cas où les données sont hautement dimensionnelles.

L'ACP a un large éventail d'applications allant de la compression des données au regroupement. Les métriques utilisées pour évaluer les performances des approches utilisées sont mentionnées ci-dessous :

- ***Accuracy***

L'accuracy est la mesure de performance la plus intuitive. Il s'agit simplement d'un rapport entre les observations correctement prédites et le total des observations. Si nous avons une précision élevée pour un modèle particulier, alors ce modèle est le meilleur.

$$\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{FP} + \text{FN} + \text{TN})$$

- ***Precision***

La précision est le rapport entre les observations positives correctement prédites et le total des observations positives prédites. Une précision élevée est liée à un faible taux de faux positifs.

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$$

- ***Recall***

Le rappel est le rapport entre les observations positives correctement prédites et l'ensemble des observations de la classe actuelle. Il a des valeurs comprises entre 0 et 1. Plus le rappel est élevé, plus le modèle est performant.

$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$$

- ***F1Score***

Le score F1 est la moyenne pondérée de la Précision et du Rappel. Ce score prend donc en compte les faux positifs et les faux négatifs. Il a des valeurs entre 0 et 1, plus le score F1 est élevé, plus le modèle est performant.

$$\text{F1Score} = 2 * (\text{Recall} * \text{Precision}) / (\text{Recall} + \text{Precision})$$

2. Méthode basée sur décomposition spectrale

La décomposition spectrale est une technique mathématique permettant de réduire artificiellement la dimensionnalité de l'ensemble de données. Les techniques de décomposition spectrale basées sur l'analyse en composantes principales (ACP) fonctionnent en divisant l'espace de l'ensemble de données en sous-espaces normal, de bruit et d'anomalie, ce qui permet une détection plus simple et plus efficace des anomalies.

3. Méthode basée sur l'apprentissage supervisé

Afin de résoudre le problème par classification, le problème est redéfini comme un problème d'identification où l'ensemble des données est classé en données anormales ou non anormales. Ce processus se déroule en deux parties, en commençant par l'entraînement d'un modèle sur un sous-ensemble de données et en utilisant ce modèle entraîné pour tester le reste des données. Comme les données des journaux sont incroyablement verbeuses et très déséquilibrées, l'apprentissage de la classification surpassé le modèle. Si les ensembles de données ne sont pas bien équilibrés, l'apprentissage par classification a du mal à généraliser et à classer les anomalies avec précision.

4. Méthode basée sur l'apprentissage non supervisé

L'apprentissage à partir de données non étiquetées est appelé apprentissage non supervisé. Par exemple, pour évaluer des données particulières en clusters, on peut calculer la distance moyenne des centres de clusters. D'autres formes d'apprentissage non supervisé comprennent l'apprentissage d'associations et l'identification de variables cachées telles que les genres de films. La suradaptation est un problème dans l'apprentissage supervisé ; par exemple, l'attribution d'un cluster à chaque point de données réduira la distance moyenne au centre du cluster à zéro, mais ne sera pas très utile. Ces algorithmes tirent des conclusions à partir des ensembles de données.

4.1 La distance

Cette catégorie de méthode de détection des anomalies détecte la distance d'un élément par rapport à un sous-ensemble le plus proche de lui. Bien que cette méthode fonctionne bien dans de nombreuses situations, elle échoue lorsqu'elle est appliquée à des ensembles de données présentant une distribution imprévisible avec des régions à la fois clairsemées et denses. C'est ce qu'on appelle le problème de la multi-densité, qui exclut la détection des anomalies collectives.

4.2 La densité

Les techniques de détection des anomalies basées sur la densité sont explicitement conçues pour contourner les problèmes de multi-densité dont souffrent les méthodes basées sur la distance. Les méthodes basées sur la densité tirent parti du facteur d'aberration local (LOF). Le LOF est une quantification de la mesure dans laquelle chaque ensemble de données se situe en dehors des comportements normaux, qui dépend elle-même de la densité locale de son voisinage. Comme les méthodes basées sur la densité incluent la densité en plus de la distance, les méthodes basées sur la densité peuvent fonctionner beaucoup mieux avec des distributions imprévisibles de régions clairsemées et denses.

4.3 Le regroupement ou clustering

La tâche consistant à regrouper des données sans information préalable sur les groupes est appelée "clustering". Un algorithme de clustering typique fonctionne en évaluant la similarité entre les instances et en plaçant les instances similaires dans le même cluster et les instances dissemblables dans des clusters différents.

Le clustering, considéré comme le couteau suisse de la modélisation statistique, génère des clusters à partir de similitudes dans les ensembles de données, éliminant ainsi les points de données qui ne sont pas conformes à ces clusters comme des anomalies. La technique des moyennes K est la plus populaire et peut être efficace pour détecter les anomalies contextuelles et collectives.

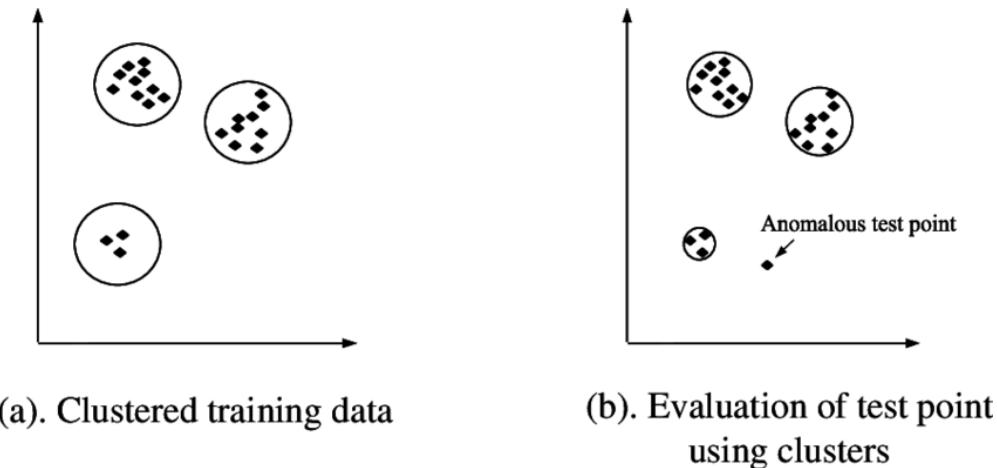


Figure 3.4 : Détection d'anomalies basée sur le clustering

4.4 Travaux antérieurs

Parmi les travaux déjà réalisés dans le domaine on peut citer :

Kumari et al tirent parti du regroupement non supervisé K-Means sur des données de trafic réseau pour détecter avec succès des anomalies avec une précision bien supérieure à celle des approches existantes [22].

Olsson et al [23] ont développé une approche d'apprentissage non supervisée pour détecter les anomalies collectives en dérivant un "score d'anomalie" pour chaque anomalie. Ils évaluent leur modèle en utilisant un ensemble de données artificielles ainsi que deux ensembles de données industrielles et détectent avec succès les anomalies dans les données de grues mobiles ainsi que dans les données de consommation de carburant au fil du temps.

5. Modèles basés sur les réseaux neuronaux

Les techniques classiques de détection des anomalies par réseaux neuronaux se comportent de la même manière que les approches d'apprentissage automatique et nécessitent donc des ensembles de données bien équilibrés. Les progrès récents des réseaux neuronaux liés aux réseaux neuronaux récurrents, aux réseaux neuronaux à mémoire à long terme et aux encodeurs automatiques ont été

largement utilisés pour résoudre une myriade de problèmes liés à la détection d'anomalies, tels que la détection d'intrusions dans les réseaux, l'analyse d'anomalies dans les données de capteurs, la détection d'anomalies dans les séries temporelles d'ECG ainsi que de nombreux autres domaines. Ces avancées plus récentes sont capables de gérer les anomalies contextuelles et collectives particulièrement bien grâce aux mécanismes de mémoire à couche cachée, ce qui permet une détection précise et généralisable des anomalies, même avec des ensembles de données déséquilibrés.

Les réseaux neuronaux récurrents (RNN) modifient les réseaux neuronaux standard en permettant aux sorties des couches cachées de neurones de servir d'entrées à la couche suivante [46]. Cela permet d'incorporer un retour d'information à chaque étape et le réseau est donc capable de tirer parti de l'historique pour prendre des décisions de classification [46]. Cette approche rend les RNN souhaitables pour la détection des anomalies, car la conservation du contexte dans les événements du journal est essentielle pour découvrir les anomalies contextuelles.

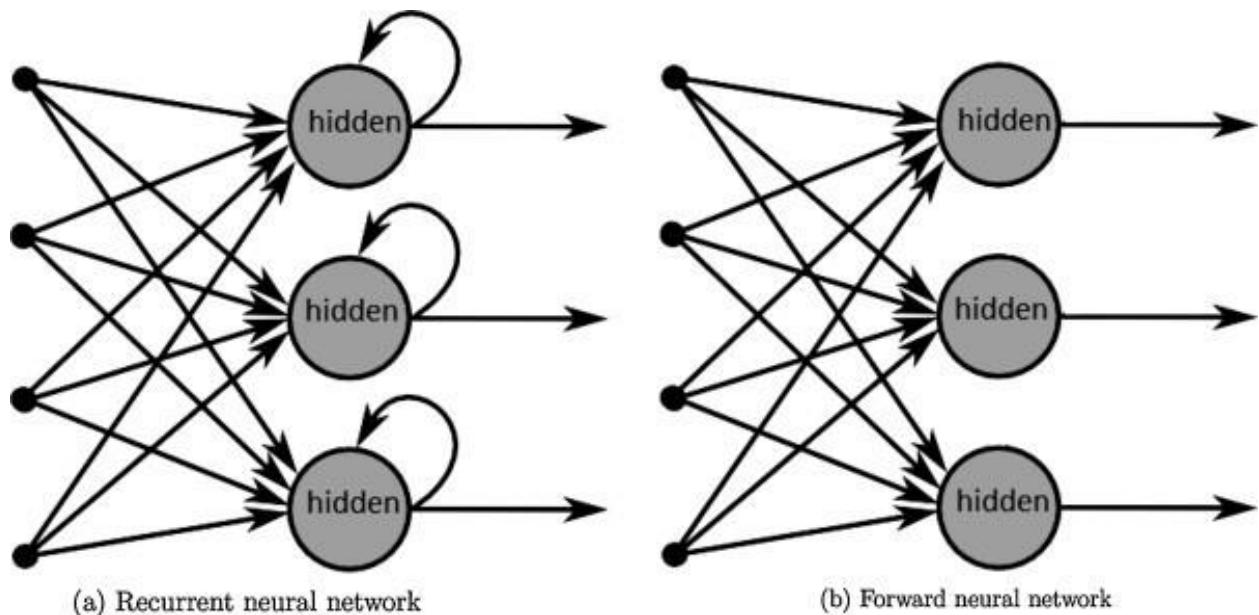


Figure 3.5 : Comparaison de l'architecture d'un RNN et d'un réseau neuronal direct

Les réseaux neuronaux récurrents sont largement utilisés pour la détection des anomalies, avec des approches allant du regroupement à la reconstruction, en passant par la classification [46]. Les recherches récentes sur la détection d'anomalies à l'aide de réseaux neuronaux ont porté sur les réseaux neuronaux à mémoire à long terme et les encodeurs automatiques, qui utilisent souvent des réseaux neuronaux récurrents [46].

5.1 Détection d'anomalies avec le réseau de neurones LSTM

Les réseaux neuronaux à mémoire à long terme (LSTM) sont une sous-catégorie de réseaux neuronaux récurrents particulièrement adaptés à l'apprentissage des dépendances à long terme entre les données d'entrée. L'architecture LSTM est représentée par des blocs de mémoire qui sont eux-mêmes essentiellement des structures connectées de manière récurrente.

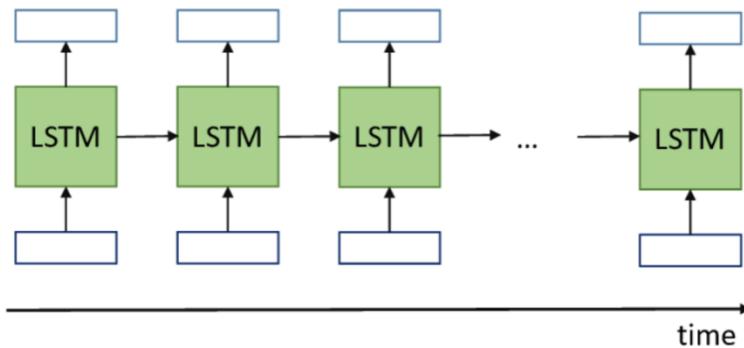


Figure 3.6: Architecture LSTM

5.2 Détection d'anomalies avec l'encodeur automatique

Les autoencodeurs sont des réseaux de neurones artificiels conçus pour induire une représentation pour des ensembles de données en apprenant des approximations de la fonction d'identité de l'ensemble de données. Ils sont généralement associés à un décodeur qui est utilisé pour recréer l'ensemble de données initial en utilisant la représentation définie par les autoencodeurs. En termes d'architecture, un auto-codeur peut être aussi simple qu'un réseau neuronal à action directe, qui peut être récurrent ou non.

Les couches d'entrée et de sortie sont connectées entre elles par des couches cachées. L'approche de base de l'utilisation des auto-codeurs consiste à utiliser un codeur pour cartographier une approximation de l'ensemble de données qui est ensuite reconstruite par le décodeur. Cette représentation reconstruite contient les caractéristiques les plus importantes et une erreur de reconstruction est alors calculée pour déchiffrer les anomalies.

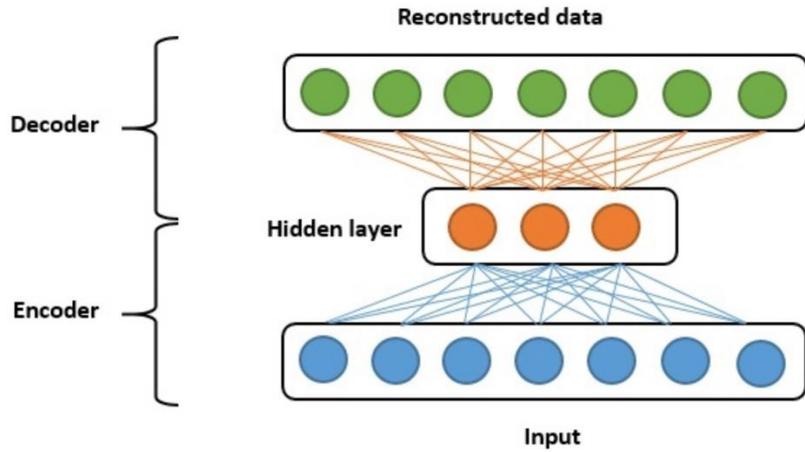


Figure 3.7: Apprentissage de l'encodeur automatique

5.3 Travaux antérieurs

Parmi les travaux déjà réalisés dans le domaine on peut citer :

Sakurada et al. [25] utilisent des encodeurs automatiques comme outils de réduction de la dimensionnalité pour détecter des anomalies dans des ensembles de données réelles et artificielles. Les ensembles de données utilisés sont basés sur des séries temporelles et les résultats de leur architecture d'encodeur automatique sont comparés en détail aux techniques de décomposition spectrale existantes telles que l'ACP. Sakurada et al. [25] ont également comparé leurs résultats avec un codeur automatique de débruitage et ont comparé les résultats avec le codeur automatique standard et les techniques d'ACP. Dans leurs expériences, les autoencodeurs ont réussi à détecter des anomalies subtiles qui n'avaient pas été détectées par l'ACP. En outre, le débruitage des encodeurs automatiques a permis d'obtenir une plus grande précision et de détecter des anomalies subtiles. Les auteurs ont également noté que, contrairement aux techniques PCA à noyau, les Auto-Encoders nécessitent moins de cycles de calcul.

Dans [24], Malhotra et al. exploitent les réseaux neuronaux pour résoudre la détection des anomalies dans les séries temporelles. Ce modèle a été comparé aux données réelles pour les n prochaines étapes temporelles afin de déduire avec succès les anomalies.

IV. Synthèse

L'apprentissage automatique, les réseaux neuronaux et les encodeurs automatiques sont conçus pour être utilisés avec des données non équilibrées, non structurées, non étiquetées et spécifiques à un domaine. Les recherches intensives dans ces domaines, combinées à l'omniprésence du matériel de base, ont rendu ces techniques accessibles, puissantes et très efficaces pour la détection des anomalies [21].

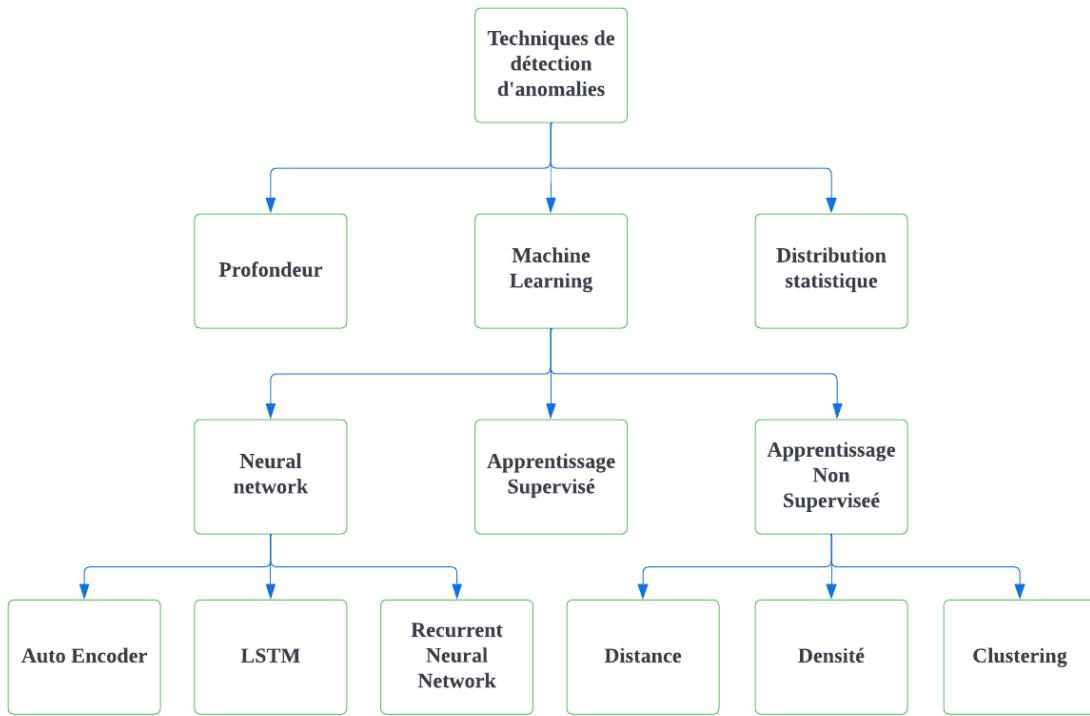


Figure 3.8 : Techniques pour la détection d'anomalies

Comme le montre notre revue de la littérature, l'apprentissage automatique non supervisé, les réseaux neuronaux et les Auto-Encoders sont bien adaptés à la détection d'anomalies ponctuelles, collectives et contextuelles dans les données du journal des applications. De plus, ces techniques de pointe sont utilisées indépendamment et dans différents domaines pour détecter avec succès des anomalies dans des ensembles de données qui partagent des caractéristiques avec les données des journaux d'applications.

ÉTUDE CONCEPTUELLE DE LA SOLUTION

Dans ce chapitre d'une grande importance, il sera question de faire une étude de premier niveau et une évaluation préliminaire du projet. Il s'agira alors de traiter des éléments conceptuels de notre projet, des bases sur lesquelles sont bâties notre solution.

Ainsi, nous y avons fait d'abord une analyse complète des besoins, une étude détaillée des différents cas d'utilisation et aussi expliqué le fonctionnement du travail accompli par des schémas explicites et une modélisation par différents diagrammes UML. Ensuite, nous y décrivons de façon plus explicite l'architecture du projet, la phase allant de la collecte jusqu'au traitement du jeu de données passant par la description de ces dernières et pour conclure apporter plus de justifications sur le choix des algorithmes et certaines technologies.

I. Analyse des besoins

1. Les besoins fonctionnels

La solution proposée face aux limites que montre le système actuel de TeamX offre entre autre plusieurs fonctionnalités :

- ⇒ visualiser les données d'opérations
- ⇒ détecter les anomalies en temps réel
- ⇒ alerter en cas de détection d'anomalie
- ⇒ faire du monitoring

2. Les besoins non fonctionnels

Il s'agit des besoins complémentaires dont l'importance est confirmée car permettant le bon fonctionnement du système. Ainsi, le système proposé dans notre solution fera en sorte que les données soient agrégées suivant des critères bien définis afin de faciliter leur visualisation. Les anomalies détectées seront classées selon leur degré de sévérité pour faire en sorte que les alertes ne concernent que les plus sévères d'entre elles. De même, le système d'alerte sera accompagné d'une notification via le canal de travail (Slack) pour mieux se pencher vers une approche plus réactive. Enfin, tous ces éléments seront mis au point avec la sécurité nécessaire des données et des services.

II. Modélisation UML

La modélisation permet de faire l'analyse et la conception de l'information contenue dans un système afin de faire une description visuelle et graphique des besoins. Dans le cadre de ce projet, nous avons utilisé le langage UML (Unified Modeling Language, ou langage de modélisation unifié) pour la conception de la solution proposée. Ainsi, nous avons utilisé les diagrammes de classe, de cas d'utilisation pour représenter les interactions et les exigences du système.

1. Diagramme des cas d'utilisation (Uses Cases)

Les diagrammes de cas d'utilisation sont des diagrammes UML qui permettent de cartographier la sphère de fonctionnalités d'un système et comment ces dernières interagissent entre elles ou avec les acteurs. La figure 4.1 représente le diagramme de cas d'utilisation du système global.

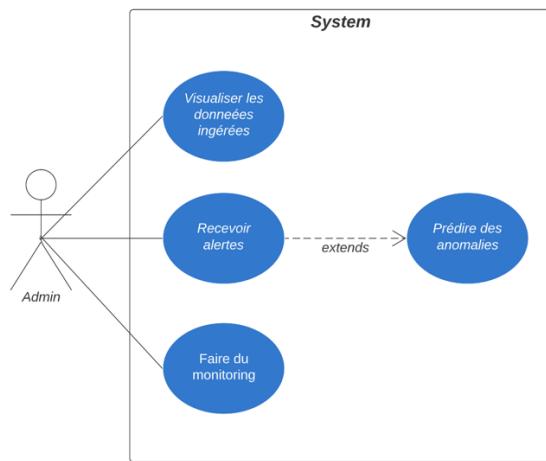


Figure 4.1 : Diagramme des cas d'utilisation

2. Diagramme des classes

Les diagrammes des classes permettent de représenter chaque classe ainsi que ses attributs et ses méthodes ou fonctions. Il permet aussi de représenter les relations entre les classes. À partir de ce diagramme on en déduit une implémentation de notre application streaming.

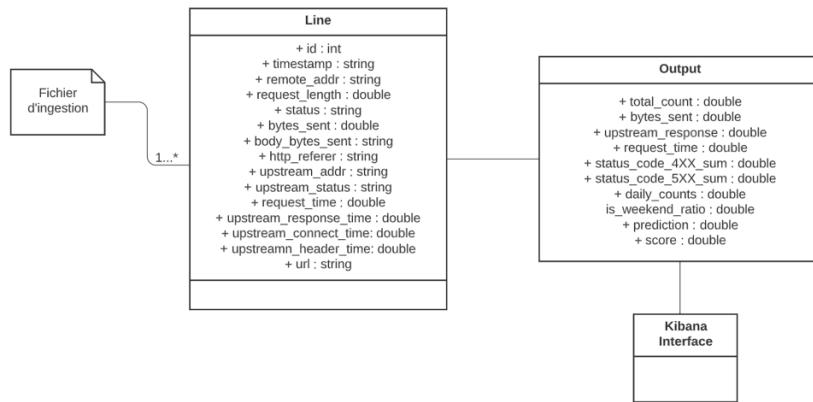


Figure 4.2 : Diagramme des classes pour l'application streaming

III. Architecture du projet

Dans cette section nous présentons d'abord une vue d'ensemble des pipelines logiciels et des données en continu, puis nous décrivons l'architecture de pipeline proposée.

1. Pipeline logiciel et données en continu

Un pipeline logiciel est une chaîne d'applications qui traite un flux de données de manière séquentielle, c'est-à-dire que la sortie d'une application est l'entrée de la suivante. Cette structure est particulièrement adaptée au traitement des flux de données. Le flux de données fait référence à des quantités limitées de données, également appelées événements, généralement de la taille de quelques kilo-octets, qui sont traitées par les différentes applications du pipeline dès que la source de données les recueille. Les données en flux sont particulièrement difficiles à traiter, car elles n'ont ni début ni fin et leur génération peut être imprévisible. S'il existe plusieurs sources de données, il peut être impossible de savoir à l'avance combien de nouvelles données atteindront l'entrée du pipeline à un instant donné. Parmi les exemples de données en continu, on peut citer les données IoT, l'interaction des utilisateurs avec un site web, les transactions bancaires ou les journaux de travail des applications dans un réseau qui est le cas d'utilisation considéré dans notre travail.

Les opérations les plus courantes qui sont effectuées sur les données en continu sont le traitement, le stockage et la visualisation. Cependant, en raison de l'intérêt croissant pour le traitement des données en ligne rendu possible par des algorithmes d'apprentissage automatique et des plateformes de traitement efficaces, comme dans le cas de l'analyse des données des dispositifs IoT, de la détection des fraudes de paiement en ligne, du suivi des utilisateurs en ligne, souvent l'étape de traitement des données ML est incluse dans les pipelines, ce qui ajoute une complexité supplémentaire à l'infrastructure. Dans ce partie, nous proposons un pipeline logiciel générique qui peut être utilisé pour traiter les flux de données, ce qui inclut les opérations classiques du pipeline ainsi que les applications d'apprentissage automatique sur les flux de données. Nous divisons le pipeline proposé en quatre groupes, qui sont analysés séparément et sont énumérés ci-dessous :

- Ingestion de données : Applications qui prennent en charge l'ingestion des données dans le système. Elle comporte trois étapes séquentielles : mise en mémoire tampon, prétraitement et stockage.
- Application d'apprentissage automatique : application d'un modèle d'apprentissage automatique pré-entraîné au flux de données.
- Visualisation des données : permet à l'examineur d'examiner les données et de connaître les résultats de l'apprentissage automatique.
- Surveillance de la santé : surveille l'état des applications du pipeline pour détecter les défaillances logicielles et y réagir.

La figure 4.3 présente une vue d'ensemble du pipeline proposé, avec l'interaction entre les composants.

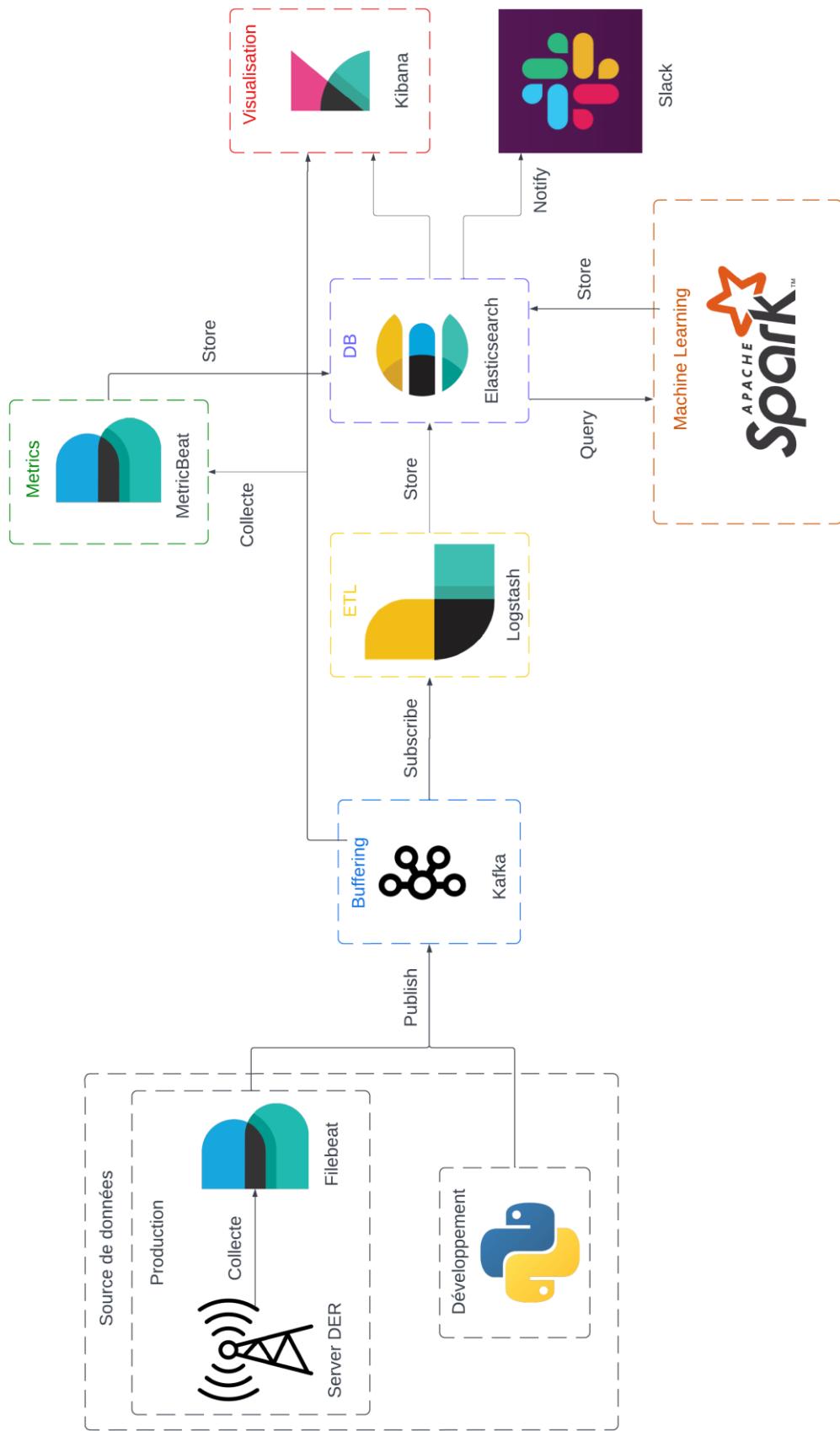


Figure 4.3: Architecture du pipeline

2. Description de l'architecture proposée

La toute première tâche dans le pipeline d'analyse des données de logs est la collecte des logs pour l'analyse. Il s'agit d'un processus continu qui doit être réalisé en temps réel. Pour ce faire, on utilise Filebeat et Logstash. Les logs du serveur DER sont poussés dans le pipeline en ayant une instance Filebeat fonctionnant sur le serveur configurée pour envoyer les logs dans Logstash. Le Logstash au début du pipeline accepte ces journaux. Les journaux collectés par le Logstash sont analysés, mis en correspondance avec le format prédéfini pour en vérifier la validité. Les logs sont agrégés pour obtenir le nombre de logs avec un niveau d'information et d'erreur dans une fenêtre d'une minute. Ces agrégations sont transmises au module d'apprentissage automatique pour d'autres analyses.

Les données prétraitées sont analysées par le module d'apprentissage automatique. Le modèle ML est entraîné avec les nouvelles données à chaque intervalle de réentraînement prédéfini afin qu'il puisse s'adapter aux nouveaux modèles de journaux qui apparaissent au fil du temps. Un score d'anomalie est calculé, et les valeurs prédites sont ensuite envoyées à la pile élastique. Tous ces processus sont exécutés en tant que processus Spark afin de traiter la quantité massive de données. Le transfert de données entre chaque sous-module est effectué par Kafka afin d'obtenir un décalage minimal dans le flux de données.

Les prédictions faites par le module d'apprentissage automatique ainsi que le score d'anomalie sont indexés dans Elasticsearch. Les données stockées dans Elasticsearch sont interrogées et visualisées sous forme de graphiques sur le tableau de bord Kibana en temps réel. L'URL du tableau de bord peut être partagée avec l'utilisateur du pipeline à des fins de surveillance. Un message d'alerte est envoyé à l'utilisateur lorsque le score d'anomalie [N1] dépasse la valeur seuil définie. L'alerte peut être un courrier envoyé à l'adresse électronique de l'utilisateur ou un message sur le compte Slack de l'utilisateur.

IV. Collecte de données

Dans cette partie on va commencer par décrire un peu les données qui ont pu être récoltées dans les logs. Notons d'abord que la boite ne stockait pas ces données de manières continues. Les logs étaient enregistrés de manière temporaire (hebdomadaire) au niveau du serveur pour prévenir les soucis de stockage. Aussi étant donné que les données de journal d'application en direct peuvent être difficiles à obtenir en raison d'une multitude de problèmes de confidentialité et de sécurité, il a fallu les récolter de manière périodique sur une période de trois mois et plus environ depuis le serveur cal (pre-prod) de l'entreprise. L'application principalement ciblé dans cette étude était à sa première version de développement V1. Les données récupérées nous ont permis de prendre en compte un nombre assez diversifié d'erreurs survenues et assez inhabituelles. Ci-dessous on montre un exemple de logs généré par l'application:

[N1] Les données qui arrive au niveau de la pile elastic se font de façon séquentielle ce qui fait qu'une anomalie peut être détectée à maintes reprises suivant le flux du coup arrivé à seuil l'alerte sera directement envoyé.

```
"21/Apr/2022:13:18:58 +0000" client=41.82.163.126 method=GET request="GET /socket.io/?EIO=3&transport=websocket HTTP/1.1" request_length=343 status=101 bytes_sent=414 body_bytes_sent=120 referer=- user_agent="Dart/2.16 (dart:io)" upstream_addr=unix:/tmp/passenger.X1hKtzl/agents.s/core upstream_status=101 request_time=0.003 upstream_response_time=0.004 upstream_connect_time=0.000 upstream_header_time=0.004
```

```
In [2]: LOG_FORMAT = '$time_local' client=$remote_addr +'method=$request_method request="$request" +' + 'request_length=$request_length +' + 'status=$status bytes_sent=$bytes_sent +' + 'body_bytes_sent=$body_bytes_sent +' + 'referer=$http_referer +' + 'user_agent="$http_user_agent"-' + 'upstream_addr=$upstream_addr +' + 'upstream_status=$upstream_status +' + 'request_time=$request_time +' + 'upstream_response_time=$upstream_response_time +' + 'upstream_connect_time=$upstream_connect_time +' + 'upstream_header_time=$upstream_header_time'
```

Figure 4.4 : Format donné pour les logs

Le pipeline est exécuté avec des données dynamiques générées dans les fichiers journaux. Le modèle d'apprentissage est entraîné en continu à un intervalle spécifique. Environ 100 Mo de données sont générés dans un fichier journal sur un mois. Le pipeline est exécuté avec un intervalle de réentraînement d'une semaine afin que le modèle apprenne le comportement du système. La figure suivante (figure 4.5) montre le processus assez rapide utilisé pour collecter les données.

```
: # Loads data.
der_accesslog_es = spark.read.format("es") \
    .option("es.port", "9200") \
    .option("es.nodes", "http://18.222.56.201") \
    .option("es.nodes.discovery", "false") \
    .option("es.nodes.wan.only", "true") \
    .option("es.net.http.auth.user", "elastic") \
    .option("es.http.timeout", "10s") \
    .option("es.net.http.auth.pass", "ianwY00bkckt13TfJvd6" ) \
    .load("der_accesslog")
```

Figure 4.5: Récupération des données sur Elasticsearch

V. Description du jeu de données

Il s'agit d'un dataset de **67946 lignes** distincts et **14 colonnes**. Ces données font une description globale des requêtes adressées à l'application, de sa performance, des temps de réponse, etc. Le tableau ci-dessous donne assez d'informations sur le jeu de données :

Tableau 4.1: Description du jeu de données

| Labels | Types | Descriptions |
|------------------------|----------|---|
| @timestamp | datetime | ce champ sauvegarde la date pendant laquelle l'évènement a été enregistré. l'enregistrement se fait jusqu'à la seconde près, ce qui donne une excellente précision. |
| remote_addr | object | fait référence à l'adresse IP du client |
| request_length | object | request length (y compris le request line, le header et le body request) |
| status | int64 | response status |
| bytes_sent | int64 | nombre de bytes envoyé par le client |
| body_bytes_sent | object | longueur du request body |
| http_referer | object | http request header contient une adresse absolue ou partielle de la page qui effectue la requête. |
| upstream_addr | object | adresse du cluster auquel vous pouvez adresser les demandes par proxy |
| upstream_status | object | response upstream status |
| request_time | float64 | total time passé à traiter une demande |
| upstream_response_time | object | le temps passé entre l'établissement d'une connexion et la réception du dernier octet du corps de la réponse du serveur en amont |
| upstream_connect_time | object | le temps passé à établir une connexion avec un serveur en amont |
| upstream_header_time | object | le temps entre l'établissement d'une connexion et la réception du premier octet de l'en-tête de réponse du serveur en amont |
| url | object | l'url demandée par un client |

VI. Prétraitement des données

La première étape de la mise en œuvre de notre étude est le prétraitement des données, où les données brutes du journal sont transformées en caractéristiques qui peuvent être ingérées par nos

algorithmes de détection des anomalies. Les données brutes sont utilisées comme entrée dans la phase d'analyse des journaux. L'analyse des journaux élimine les détails superflus ou spécifiques à l'exécution des données brutes et la sortie est utilisée comme entrée pour la phase d'extraction des caractéristiques. À ce stade, les données de journal analysées sont converties en caractéristiques numériques qui sont utilisées comme entrée dans notre algorithme de détection d'anomalies qui identifie ensuite les anomalies en fonction de la technique utilisée.

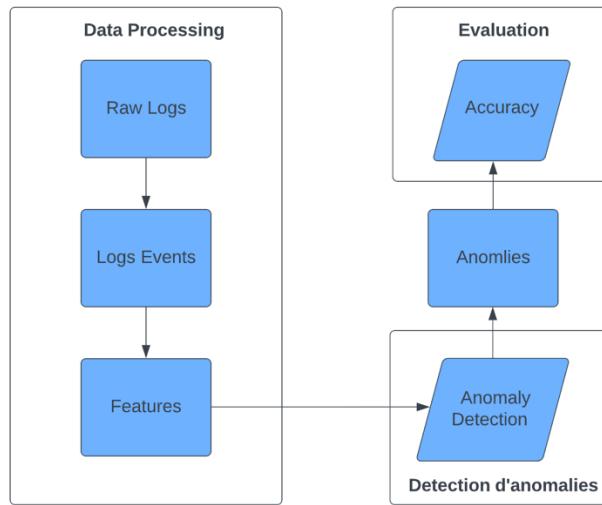


Figure 4.6: Flux de mise en œuvre de l'étude

Dans cette partie on va expliquer les différentes features prises en compte pour l'établissement de notre modèle d'apprentissage machine.

```

size of dataset 67946

{'time_local': datetime.datetime(2022, 4, 14, 2, 43, 14),
 'remote_addr': '102.164.170.249',
 'bytes_sent': 371.0,
 'request_time': 0.001,
 'upstream_response_time': 0.0,
 'status_code_4XX': 0,
 'status_code_5XX': 0,
 'url': '/api/users?numPage=1&peerPage=10',
 'country': 'SN',
 'browser_family': 'Chrome',
 'os_family': 'Linux'}

```

Figure 4.7: Première étape du prétraitement

Les données récupérées sont d'abord formattées afin d'y ajouter les colonnes suivantes pour le prétraitement :

⇒ *status_code_4XX* qui sera une colonne type booléenne pour déterminer si c'est une erreur niveau client.

- ⇒ *status_code_5XX* qui sera une colonne type booléenne pour déterminer si c'est une erreur niveau server.
- ⇒ *country* qui est directement tiré au niveau de la colonne requête et détermine sa source.
- ⇒ *url* qui est directement tiré au niveau de la colonne requête et détermine le lien demandé.
- ⇒ *browser_family* qui est directement tiré au niveau de la colonne requête et détermine le navigateur utilisé.
- ⇒ *os_family* qui est directement tiré au niveau de la colonne requête et détermine le système d'exploitation de la source

Par la suite, les données sont regroupées en fonction de leurs valeurs (ressemblances) afin d'éviter les duplicitas. On passe ainsi à la partie prétraitement proprement dite où l'on spécifie les colonnes ou features qui seront prises en compte pour l'application de notre modèle de prédiction :

bytes_sent:

correspond à la variance du nombre de bytes envoyés par rapport à la normale. Ceci doit rester une valeur constante pour chaque type de service donnée sauf altéré

upstream_response:

correspond à la variance par rapport à la normale du temps pour l'établissement de la connexion avec le serveur. Obtenir cette valeur pour les différents services nous permettaient en même temps de repérer les valeurs aberrantes.

request_time:

correspondra à la variance par rapport à la normale du temps de réponse pour un service donné. Celui-ci reste une information très pertinente pour repérer un problème au niveau du serveur.

status_code_4XX_sum:

correspond au nombre total d'erreurs **internes** pour un service donné c'est à dire destinée aux situations dans lesquelles l'erreur semble avoir été causée par le client.

status_code_5XX_sum:

correspond au nombre total d'erreurs **externes** pour un service donné qui indiquent les cas dans lesquels le serveur est conscient d'avoir rencontré une erreur ou d'être incapable d'exécuter la demande.

total_count:

permet de retracer un peu l'activité des clients de manière globale journalière (*daily_counts*) aussi hebdomadaire (*is_weekend_ratio*) et repéré les services les plus sollicités dans les applications

daily_counts:

permet de retracer l'activité journalière (nombre de sollicitations par service) comme une tentative de connexion par exemple *login*

is_weekend_ratio:

permet de retracer l'activité dans la semaine

tidy_url:

cette colonne répertorie les mots clés dans chaque erreur de service rencontrée et pourrait être utile pour établir un système de messaging. Cette colonne ne sera pas pris en compte pour l'application du modèle.

Le tableau ci-dessus répertorie les données prétraitées et normalisées et prêtes à être appliquer à notre modèle.

| bytes_sent | upstream_response | status_code_4XX_sum | status_code_5XX_sum | total_count | request_time | daily_counts | is_weekend_ratio | tidy_url |
|------------|-------------------|---------------------|---------------------|-------------|--------------|--------------|------------------|--|
| 0.000102 | 0.000000 | 0.404147 | | 0.0 | 0.096582 | 0.002213 | 0.089041 | 0.016153 |
| 0.000000 | 0.000000 | 0.000461 | | 0.0 | 0.000000 | 0.000000 | 0.000000 | 27https agent api wizall com subscribers woyof... |
| 0.000000 | 0.000000 | 0.108756 | | 0.0 | 0.025910 | 0.000013 | 0.020548 | 0.011718 |
| 0.003133 | 0.000002 | 0.971429 | | 0.0 | 0.525358 | 0.000312 | 0.568493 | 0.013291 |
| 0.000000 | 0.000000 | 0.000461 | | 0.0 | 0.000000 | 0.000000 | 0.000000 | 23a 40org apache commons IOUtils 40toString 40... |

Figure 4.8: Inputs pour l'algorithme de prédiction

VII. Algorithme

Dans cette recherche, deux méthodes sont utilisées pour répondre aux questions de recherche : L'analyse documentaire et l'expérimentation. La conception de ce projet est d'analyser les données et de considérer les méthodes appropriées qui peuvent être utilisées pour trouver les anomalies à partir des fichiers journaux. Ainsi, une revue de la littérature est menée pour acquérir des connaissances sur les techniques d'apprentissage automatique utilisées précédemment pour ce genre de questions. Cela nous a aidé à sélectionner les algorithmes qui peuvent être utilisés pour développer le modèle. La sélection des algorithmes dépend de la précision avec laquelle ils vont

prédir les anomalies, ce qui signifie qu'il faut sélectionner les algorithmes qui montrent les meilleurs résultats de performance en utilisant des métriques de mesure statistiques classiques. Une revue de la littérature (enquête) des algorithmes d'apprentissage automatique existants dans le domaine qui pourrait potentiellement fonctionner le mieux pour ce problème (c'est-à-dire, trouver les anomalies dans le fichier journal) donne une réponse à la Q1 (analyse documentaire).

La Q2 ou l'expérimentation est répondue par l'expérience menée. L'expérience est choisie par rapport à d'autres méthodes de recherche car elle implique la manipulation de variables [43] qui est nécessaire pour obtenir les résultats de cette étude. Le type de problème abordé est traité par le clustering car l'algorithme doit classer une anomalie à partir du fichier journal. Les données collectées sont sous leur forme brute et ne sont pas entièrement structurées ni même étiquetées. Le fichier journal est donc formatté lors de l'analyse des données, puis seules les données nécessaires pour alimenter les algorithmes sont prises en compte. Il s'agit donc de données non étiquetées, c'est pourquoi nous avons utilisé des techniques d'apprentissage automatique non supervisées dans l'étude. En revanche, des techniques supervisées sont utilisées lorsque cela est nécessaire, ce qui est abordé dans la suite.

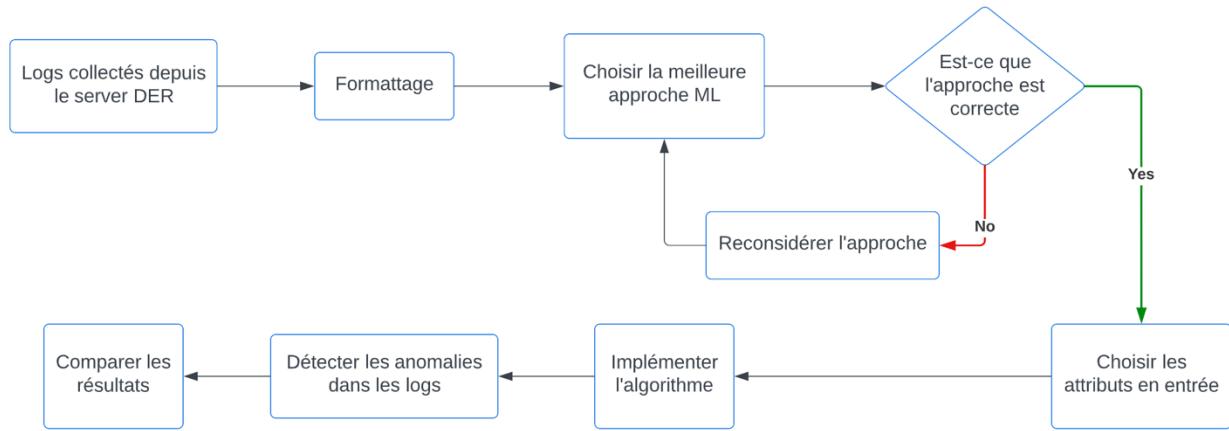


Figure 4.9: Workflow Algorithme ML

Expérience 1: Unsupervised Machine Learning (K-Means Clustering)

Puisqu'il s'agit d'un processus non supervisé, nous avons étiqueté manuellement les données et voir la performance des algos individuels. Les critères pour les étiquettes manuelles sont strictement basés sur les décomptes d'occurrences des erreurs (4xx et 5xx) ainsi que le temps de réponse car nous voulons que les algos soient robustes sur les anomalies de décomptes. Le même process sera utilisé dans la suite afin d'en tirer le meilleur modèle avec les autres algorithmes.

L'algorithme démarre le clustering K-Means avec K égal à deux. Il augmente ensuite itérativement K (clusters) jusqu'à ce que le K souhaité soit atteint. À chaque itération, nous comparons les anomalies identifiées avec les anomalies réelles dans l'ensemble de données. Le nombre de clusters est augmenté jusqu'à ce que les mesures de précision atteignent leur maximum. La valeur K choisie représente donc la taille de cluster à laquelle les mesures du coût sont les plus faibles et commencent à augmenter à mesure que la taille du cluster augmente.

- *choix du nombre de clusters*

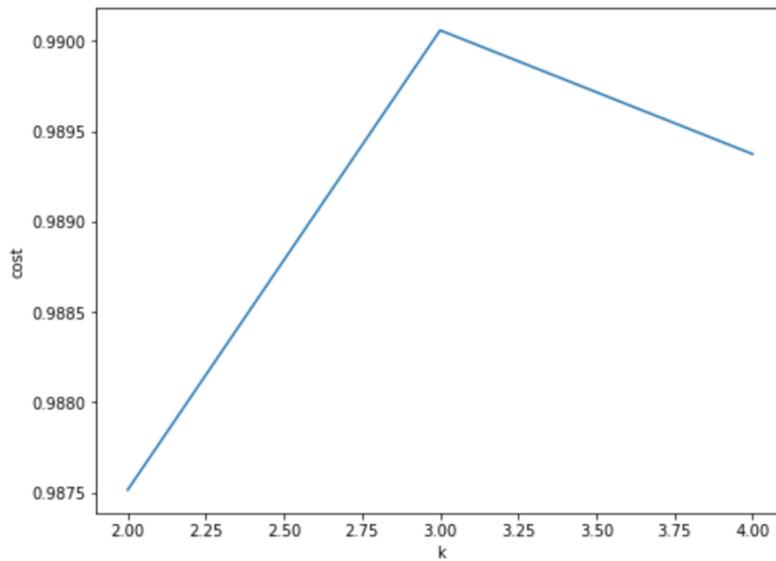


Figure 4.10: Variation du coût en fonction du nombre de clusters k

- *après prédiction sur le nombre d'erreurs globale obtenu*

```
data_new[ 'anomaly_kmeans' ].value_counts()

0      2020
1         4
Name: anomaly_kmeans, dtype: int64
```

Figure 4.11: Décompte des anomalies obtenues avec le k-means

- *résultat (output)*

| | url | bytes_sent_mean | upstream_response_mean | status_code_4XX_sum | status_code_5XX_sum | total_count | request_time_mean | da... |
|-----|---|-----------------|------------------------|---------------------|---------------------|-------------|-------------------|-------|
| 639 | /api/contenu | 0.000337 | 0.000793 | 0.006452 | 0.000503 | 0.056009 | 0.000796 | |
| 682 | /api/endpoints/2/docker/tasks? filters=%7B%7D | 0.010123 | 0.000015 | 0.000000 | 0.000000 | 0.016759 | 0.000016 | |
| 888 | /api/wizall/transaction-type/transactionType?n... | 0.000888 | 0.001260 | 0.004147 | 0.000503 | 0.093495 | 0.001263 | |
| 895 | /api/wizall/transactions | 0.000367 | 0.000815 | 0.064977 | 0.000252 | 0.031092 | 0.000817 | |

Figure 4.12: Résultats obtenus avec le k-means

Expérience 2: Unsupervised Machine Learning (One Class SVM)

- application du grid search (modèle de validation) pour le choix des hyperparamètres

```
param_grid = {'degree': [1, 2, 5],
              'nu': ['auto', 0.001, 0.02, 0.1],
              'kernel': ['rbf', 'linear'],
             }

data_new_clean = data_new[num_features].loc[data_new[num_features].total_count <= data_new.total_count]

grid_search = model_selection.GridSearchCV(model_svm,
                                             param_grid,
                                             scoring='accuracy',
                                             refit=True,
                                             cv=10,
                                             return_train_score=True)
```

Figure 4.13: Cross validation avec le one class svm

- après prédiction sur le nombre d'erreurs globale obtenu

```
data_new['anomaly_svm'].value_counts()

0    1098
1     926
Name: anomaly_svm, dtype: int64
```

Figure 4.14: Décompte des anomalies obtenues avec le one class svm

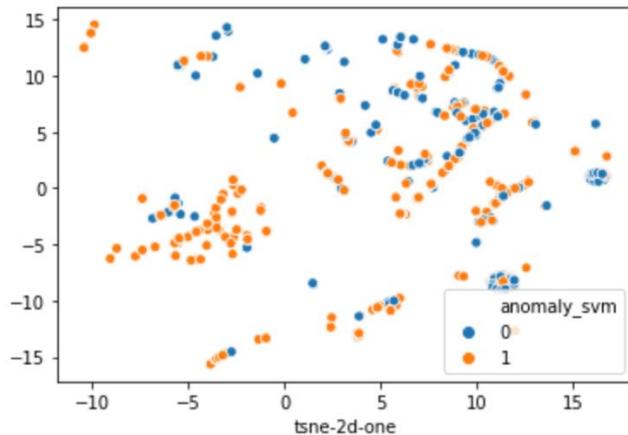


Figure 4.15: Représentation graphique des anomalies obtenues avec le one class svm

- *résultat (output)*

| | | | | |
|--|---|----------|-----|----------|
| 32 | ./well-known/acme-challenge/9sg5nbkyA6i-kPZDNR... | 0.000049 | 0.0 | 0.000000 |
| to scroll output; double click to hide | // | 0.000330 | 0.0 | 0.000000 |
| 54 | //html5shim.googlecode.com/svn/trunk/html5.js | 0.000136 | 0.0 | 0.000461 |
| 57 | //MyAdmin/scripts/setup.php | 0.000113 | 0.0 | 0.000461 |
| 60 | //admin/.env | 0.000116 | 0.0 | 0.000461 |
| 61 | //admin/config.php | 0.000116 | 0.0 | 0.000461 |
| 62 | //admin/vendor/phpunit/phpunit/src/Util/PHP/ev... | 0.000116 | 0.0 | 0.000461 |

Figure 4.16: Résultats obtenus avec le one class svm

Expérience 3: Unsupervised Machine Learning (Isolation Forest)

- *application du grid search (modèle de validation) pour le choix des hyperparamètres*

```
num_features = ['total_count', 'request_time_mean', 'daily_counts', 'bytes_sent_mean',
                 'upstream_response_mean', 'status_code_4XX_sum', 'status_code_5XX_sum', 'is_weekend_ratio']

model = IsolationForest(random_state=RANDOM_STATE)

param_grid = {'n_estimators': [100, 200, 500],
              'max_samples': [5, 10],
              'contamination': ['auto', 0.1, 0.0002, 0.02],
              'max_features': [8],
              'bootstrap': [True],
              'n_jobs': [-1]}

grid_search = model_selection.GridSearchCV(model,
                                             param_grid,
                                             scoring="neg_mean_squared_error",
                                             refit=True,
                                             cv=10,
                                             return_train_score=True)
```

Figure 4.17: Cross validation avec l'isolation-forest

- *après choix des best parameters et prédiction sur le nombre d'erreurs obtenu*

```
data_new['anomaly_isolated'].value_counts()

0    2022
1      2
Name: anomaly_isolated, dtype: int64
```

Figure 4.18: Décompte des anomalies obtenues avec l'isolation-forest

- *résultat (output)*

| Statistiques des requêtes | | | | | | | | | |
|---------------------------|--------------------------|------------|-------------------|---------------------|---------------------|-------------|--------------|--------------|--|
| id | url | bytes_sent | upstream_response | status_code_4XX_sum | status_code_5XX_sum | total_count | request_time | daily_counts | |
| 601 | /api/auth/signin | 0.001466 | 0.004086 | 0.075576 | 0.005286 | 0.077508 | 0.004086 | 0.047945 | |
| 895 | /api/wizall/transactions | 0.001350 | 0.005761 | 0.064977 | 0.000252 | 0.031092 | 0.005761 | 0.123288 | |

Figure 4.19: Résultats obtenus avec l'isolation forest

VIII. Analyse et évaluation des résultats

Dans cette section, nous discutons des résultats obtenus à partir des approches 1, 2 et 3 qui ont été mises en œuvre dans la section précédente. Nous interprétons également en quoi cette étude est différente des autres études.

- **Comparaison des performances**

Le tableau ci-dessus décrit un peu le score obtenu pour chaque modèle avec le taux d'erreurs, ainsi que la sensibilité, ...

| | kmeans | isolated_forest | svm |
|-------------|----------|-----------------|----------|
| accuracy | 0.943676 | 0.950593 | 0.336462 |
| roc_score | 0.496878 | 0.509804 | 0.177159 |
| sensitivity | 0.949304 | 0.950544 | 0.869732 |
| specificity | 0.000000 | 1.000000 | 0.000000 |

Figure 4.20: Performance des algos

Dans l'approche 1 (K-Means), ce sont les anomalies possibles dans la figure 4.12 qui causent l'interruption du système ou l'échec d'une suite de tests particulière. Il renvoie à des problèmes de wallet rencontré avec les utilisateurs, les transactions aussi liés à l'api, les événements se produisent très rarement et restent différents du reste des événements sont présentés dans la figure.

Dans l'approche 2 (One Class SVM), nous avons montré quelques résultats par le biais de graphiques. Les points aberrants ont également été montrés dans la figure 4.15. L'algorithme a permis de trouver les points aberrants calculés à l'aide de la mesure de distance d'accessibilité, qui est utilisée comme mesure de distance pour le facteur aberrant local. Néanmoins il présente une précision de 33,6 % ce qui reste très faible et montre que le modèle a un nombre très élevé de faux positifs. Cela signifie que le modèle prédit des valeurs comme des anomalies qui ne le sont pas en réalité. En se basant aussi sur la figure 4.16 on peut aussi constater que ces résultats restent quasi inexploitables car non pertinents comparés aux failles/bugs généralement rencontrés. Il est donc moins probable qu'il s'agisse de la meilleure approche pour cette recherche.

Dans l'approche 3 (Isolation Forest), le modèle prédit les séquences rares dans le fichier journal. Ce modèle montre une précision de 95%, ce qui constitue un très bon choix d'algorithme. Aussi comparé à l'algo de k-means le temps d'exécution reste plus court, il a une précision encore plus élevée. Le modèle prédit des anomalies réelles rencontrées en entreprise et qui nécessitaient des corrections :

- des tentatives de connexion importantes pour des utilisateurs non enregistrés (ceci peut être un soucis provenant de l'utilisateur qui a renseigné les mauvaises informations à maintes reprises, ou bien qu'il soit un problème avec l'api lors de la récupération des données tellement l'id pour l'identifier, ...).
- le soucis avec la récupération du wallet d'un client qui était un problème majeur rencontré durant un moment avec leur api qui marchait pas normalement.

Cela aiderait les développeurs à trouver où l'événement s'est produit et ils n'ont pas besoin de parcourir manuellement le journal pour trouver la raison de l'interruption du système ou des mauvaises connexions. Ce modèle fonctionne mieux pour ce projet car il est capable de distinguer et de mentionner clairement où le problème se produit sans avoir à consulter manuellement le fichier journal. Et la plupart des développeurs ne seront pas en mesure de parcourir le fichier journal à chaque fois qu'il y a un problème. Cela facilitera donc leur travail. Il montre également de très bonnes performances, ce qui en fait le meilleur outil pour cette recherche et donne une réponse à Q2.

- **Valeur ajoutée par rapport à la solution**

Dans l'étude on a utilisé plusieurs approches et comparé les résultats pour trouver la solution optimale parmi elles. Seul l'événement du journal est pris en compte dans mon approche pour trouver les anomalies, ce qui rend beaucoup plus facile l'implémentation des modèles qui ont moins de temps d'exécution. Le problème des quantités plutôt élevées de faux positifs dont souffrent toutes les techniques de détection d'anomalies reste non résolu. [44]

L'approche 3 que j'ai mentionnée dans cette étude y contribuerait car le modèle a un bon pourcentage de vrais positifs, ce qui réduit le problème du nombre élevé de faux positifs.

En plus de cela, plusieurs paramètres ont été prises en compte pour l'établissement de notre modèle et reste donc beaucoup plus vraisemblable par rapport au contexte. Par exemple avec la stack ELK (Elasticsearch, Kibana, Logstash) il existe bien des algorithmes pré-developpés de détection d'anomalies se basant sur les même types de données (logs) pour faire l'étude cependant le modèle peut être trop généraliste par rapport au jeu de données d'où l'intérêt d'entraîner notre propre modèle avec notre propre dataset.

IMPLÉMENTATION ET DÉPLOIEMENT

I. Environnement de développement et technologies

Ce chapitre décrit les principales applications utilisées pour construire le pipeline. Nous soulignons l'objectif principal et les caractéristiques de conception de chaque logiciel choisi pour faire partie de l'architecture du pipeline.

1. Docker

La gestion et la maintenance d'un cluster composé de nombreuses applications constituent un défi majeur dans le déploiement de systèmes distribués. Il y a quelques années, la seule option pour exécuter plusieurs instances en parallèle sur un matériel limité était la virtualisation. Les machines virtuelles ont le grand avantage de permettre le partage du matériel avec une affectation fine des ressources disponibles, mais cela a un coût. Le principal inconvénient des machines virtuelles est qu'elles sont difficiles à maintenir puisqu'il faut s'occuper du système d'exploitation en plus des logiciels qu'elles sont censées exécuter. L'exécution d'un système d'exploitation complet nécessite une quantité énorme de ressources, et dans le cas où nous avons besoin d'une autre instance d'un composant existant pour effectuer une extension horizontale, une autre VM doit être installée et configurée pour gérer les nouvelles exigences. Toutes ces limitations font qu'un cluster basé sur une VM n'est pas adapté à un environnement de développement, où les spécifications peuvent changer à tout moment et où les choses ont tendance à se casser, mais aussi à un environnement opérationnel, où une mise à l'échelle rapide est nécessaire pour faire face aux pics de travail ou réagir à la défaillance d'un composant.

Docker apporte une solution à toutes ces limitations et ajoute quelques fonctionnalités supplémentaires qui ne sont pas disponibles dans les machines virtuelles traditionnelles, telles que le déploiement basé sur les images, les registres d'images et le déploiement automatique des piles, tout en garantissant une isolation complète entre les différentes applications, comme le font les machines virtuelles. Dans les sections suivantes, l'architecture et les fonctionnalités principales de Docker sont présentées, en utilisant comme référence la documentation officielle [26].

1.1 Containers and Virtual Machines

La solution adoptée par Docker pour surmonter les limites des machines virtuelles consiste à ajouter un autre niveau d'abstraction : alors qu'un hyperviseur traditionnel fait abstraction du matériel physique sous-jacent, l'abstraction de Docker se situe au niveau du système d'exploitation. Pour ce faire, on introduit des conteneurs, un moyen d'empaqueter une application avec ses dépendances requises, telles que des binaires ou des bibliothèques supplémentaires : par exemple, une application Java peut être empaquetée dans un conteneur qui contient également la JVM nécessaire à son exécution. Chaque conteneur est doté d'une interface avec un système d'exploitation Linux virtuel, ce qui permet au logiciel conteneurisé de croire qu'il fonctionne au-dessus d'un véritable système d'exploitation : cela garantit que toute application existante peut être exécutée dans un conteneur sans nécessiter de modification du code source. Les conteneurs offrent

la même isolation que celle permise par les VM, puisque chacun d'entre eux s'exécute dans un environnement virtuel différent, complètement disjoint des autres. Le moteur Docker est chargé de gérer les appels au système d'exploitation de l'application conteneurisée et de les transmettre au système d'exploitation hôte. La figure 5.1 met en évidence la principale différence entre Docker et la virtualisation classique [27] : l'avantage le plus évident est que, bien que le nombre d'applications en cours d'exécution soit le même, Docker ne nécessite qu'un seul système d'exploitation, ce qui signifie qu'il n'y a qu'un seul système à gérer et à maintenir à jour et fonctionnel et que l'on économise beaucoup d'espace disque. Il est ainsi possible de regrouper chaque composant d'un cluster dans un conteneur différent, ce qui serait trop coûteux pour un environnement de machines virtuelles, ce qui permet ce que l'on appelle l'architecture de microserveurs [28] : ce degré d'isolation plus élevé rend le système beaucoup plus facile à gérer et garantit qu'aucun conflit ne survient entre les logiciels installés sur le même environnement.

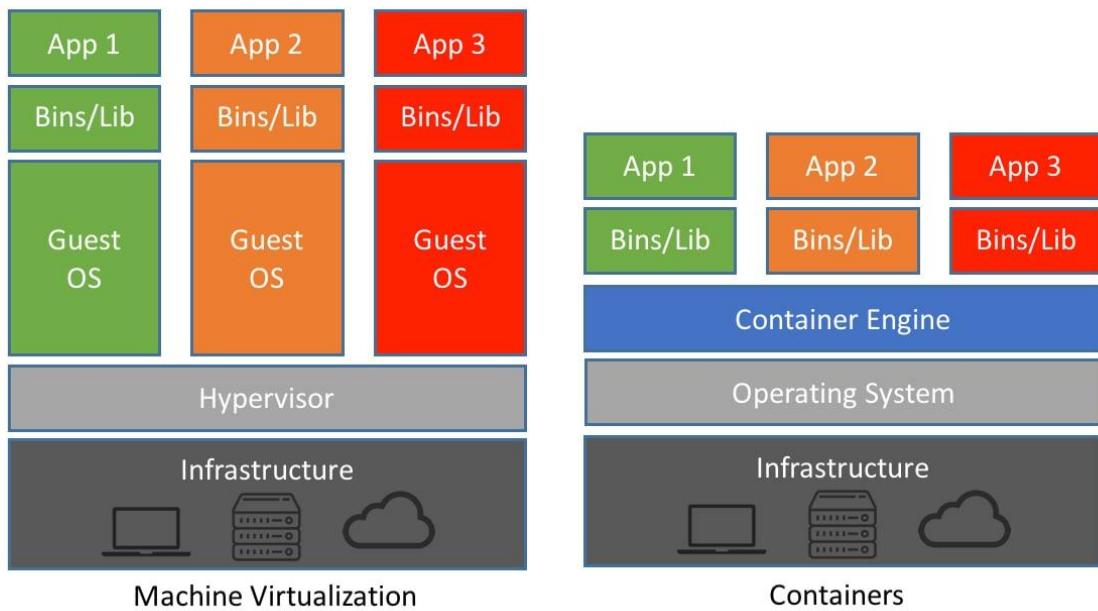


Figure 5.1 : Architecture de Docker en comparaison avec la virtualisation classique

1.2 Images

Le mécanisme des conteneurs présente un autre grand avantage par rapport aux machines virtuelles : l'application et ses dépendances étant complètement abstraites par le système d'exploitation, pour définir un conteneur, la seule chose à faire est de spécifier l'application à exécuter et ses dépendances. Docker met en œuvre une solution simple pour définir la façon dont les nouveaux conteneurs doivent être créés en utilisant des modèles de conteneurs, appelés images. Les images peuvent être construites en définissant un dockerfile, qui déclare comment installer l'application et les dépendances qui doivent être conteneurisées avec une syntaxe de style bash. Mais la véritable puissance des images réside dans le registre public : les images de presque toutes les applications disponibles peuvent y être trouvées et facilement instanciées en tant que conteneurs en quelques secondes. Grâce au système d'images, la mise à l'échelle de toute application par l'ajout d'instances

supplémentaires ne nécessite aucun effort : il suffit de créer un nouveau conteneur à partir de l'image requise.

1.3 Containers Lifecycle

Pour exécuter un conteneur, une image de celui-ci doit être disponible. Les images peuvent être créées en exécutant la commande docker build avec un dockerfile en argument ou elles peuvent être téléchargées à partir d'un registre si elles sont déjà disponibles en utilisant docker pull. Une fois l'image disponible, elle peut être instanciée en tant que conteneur avec la commande docker run. La figure 5.2 illustre le cycle de vie des conteneurs. Plusieurs conteneurs peuvent être créés à partir d'une seule image, ce qui permet une réplication rapide et une mise à l'échelle horizontale de l'application. Chaque fois qu'un conteneur est créé, il démarre dans un état "propre" qui est celui défini par l'image. Peu après le démarrage, le conteneur conserve un état qui reflète les calculs effectués par l'application et les éventuelles modifications du système de fichiers, comme la création de fichiers. Lorsqu'un conteneur est supprimé, tout ce qui diffère de l'état propre est perdu, et de nouveaux conteneurs sont créés dans l'état propre. Cependant, Docker permet la persistance du système de fichiers en définissant des volumes qui sont montés à l'intérieur du conteneur dans un répertoire particulier. Par exemple, une opération courante consiste à monter un répertoire de configuration dans tous les conteneurs créés en tant qu'instances d'une image particulière : de cette façon, tous peuvent lire et exécuter la configuration fournie.

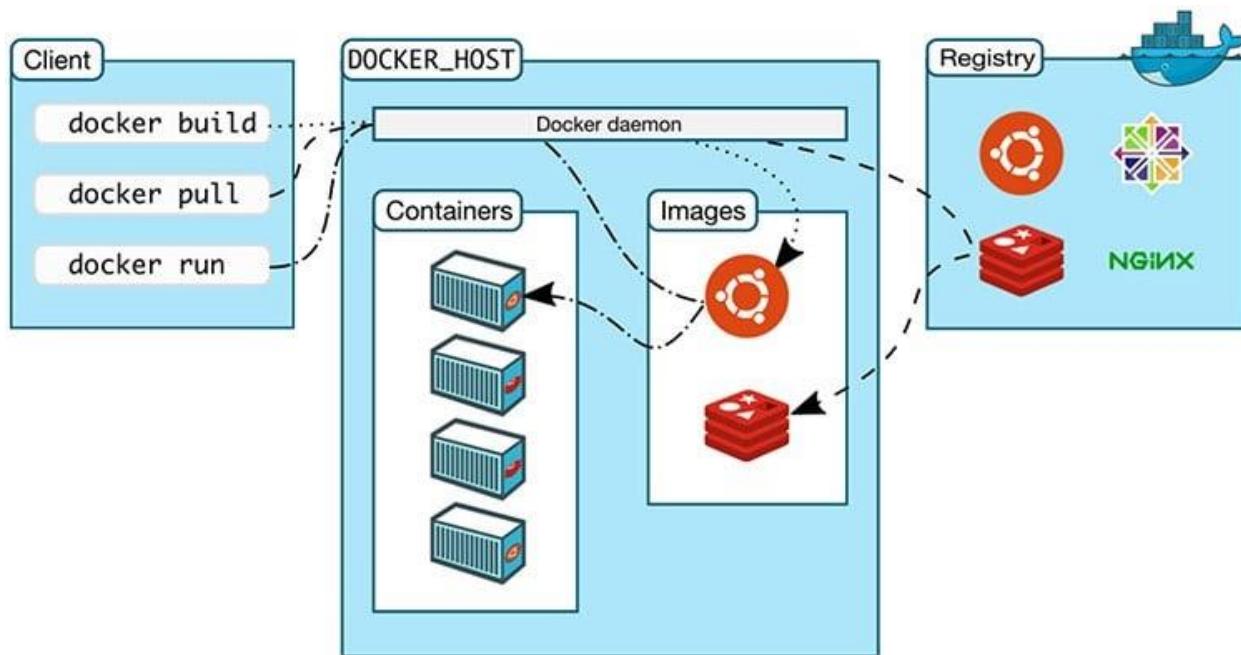


Figure 5.2 : Cycle de vie des conteneurs Docker. Source de l'image : [26]

1.4 Networking

Lorsqu'un conteneur est créé, le comportement par défaut de Docker consiste à le placer dans le réseau Docker par défaut, un sous-réseau relié à celui de la machine hôte qui assure l'isolation avec le reste du réseau et les autres applications exécutées sur la machine. L'utilisateur peut créer un réseau Docker personnalisé pour mieux segmenter les conteneurs. Pour des raisons de sécurité, le moteur Docker a mis en place un pare-feu strict sur les ponts du réseau Docker afin de bloquer toutes les demandes externes, de sorte que seuls les conteneurs qui s'exécutent à l'intérieur du même réseau puissent communiquer entre eux. Pour permettre l'accès externe à un réseau, Docker exploite un mécanisme similaire aux liaisons de port NAT entre le conteneur auquel il faut accéder et l'adresse IP de la machine hôte. Par exemple, considérons un conteneur qui expose un service sur le port 80 : la liaison de port exige que la machine hôte lie l'un de ses ports libres au port 80 du conteneur. Si le port choisi par l'hôte est 8080, un client qui n'est pas sur le réseau Docker envoie ses demandes à l'adresse IP de la machine hôte sur le port 8080, puis le moteur Docker les transmet au bon conteneur sur le port 80.

1.5 Docker Compose

Grâce à Docker, l'exécution d'une pile complète d'applications est aussi simple que le téléchargement et l'exécution des images requises en tant que conteneurs. Pour rendre ce processus encore plus simple, Docker Compose a été introduit pour automatiser la tâche fastidieuse consistant à exécuter trop de commandes docker run avec les bons paramètres. Docker Compose permet de définir les conteneurs à exécuter ainsi que leurs paramètres, les paramètres réseau et les volumes à monter dans un seul fichier qui définit l'ensemble du cluster. Il suffit ensuite d'exécuter la commande docker compose en spécifiant le fichier à exécuter et docker se chargera d'exécuter toutes les opérations nécessaires au démarrage du cluster. Cela simplifie énormément le déploiement dans les environnements opérationnels de piles d'applications : il suffit de maintenir une machine Docker polyvalente, de copier sur celle-ci le fichier docker-compose et d'exécuter la commande docker compose. Le moteur Docker effectue alors automatiquement les opérations suivantes :

1. Télécharge les images des conteneurs à partir du registre fourni.
2. Crée un réseau avec les paramètres spécifiés : cela permet d'isoler les autres piles d'applications fonctionnant sur le même hôte.
3. Crée des volumes pour la persistance des conteneurs si nécessaire
4. Instancie autant de conteneurs que spécifié pour chaque image, puis les connecte au réseau, lie les ports avec l'hôte et monte les volumes si nécessaire.

Autant de clusters que nécessaire peuvent être créés sur le même hôte, Docker Engine se charge de la gestion des ressources physiques.

2. Apache Kafka

Apache Kafka est une plateforme open-source de streaming d'événements distribués, développée à l'origine chez LinkedIn. L'objectif principal de Kafka est de fournir un courtier en messages qui interconnecte différentes applications mais il peut également être utilisé pour fournir des files d'attente ou des tampons distribués. Le streaming de données fait référence à un cas d'utilisation où les données à traiter arrivent au système en temps réel, organisées en morceaux, également appelés messages ou événements. Les flux sont difficiles à gérer car le rythme des nouveaux événements peut changer rapidement et de manière imprévisible. Cela rend l'interconnexion directe entre différentes applications impossible à réaliser de manière classique, par exemple en utilisant un mécanisme tel qu'une API. Cela est nécessaire car si la charge sur l'une des destinations est trop élevée, certains messages peuvent être rejetés car l'application n'a pas les ressources nécessaires pour les recevoir et les traiter immédiatement. De même, dans un environnement distribué où le travail peut être partagé par plusieurs instances (ou nœuds) d'un même logiciel, il est nécessaire d'équilibrer la charge des données à traiter entre les nœuds et d'assurer une certaine coordination afin que le même message ne soit pas traité deux fois par des instances différentes. Apache Kafka a été développé pour fournir tous ces types de fonctionnalités. La principale ressource pour cette section est [29].

2.1 Kafka architecture

L'architecture de Kafka est basée sur le modèle de publication et d'abonnement : lorsqu'un nouveau message doit être envoyé à une autre application, la source, appelée éditeur, publie le message sur un sujet spécifique [30]. Un topic est un nom unique qui identifie une file d'attente indépendante dans Kafka. L'application de destination, l'abonné, s'abonne au sujet qui l'intéresse et est notifié lorsque de nouvelles données sont disponibles dans la file d'attente. De cette façon, la source et la destination sont complètement découplées et abstraites. Cela présente trois avantages principaux : tout d'abord, il existe un nœud central qui fait office de routeur et de point d'entrée pour tous les messages envoyés à n'importe quel endroit du système, ce qui permet à quiconque a besoin d'un flux spécifique, même s'il provient de plusieurs sources différentes, de l'obtenir simplement en s'abonnant à ce sujet. Puis le courtier en messages agit comme un tampon : les messages sont toujours conservés dans la file d'attente, et le souscripteur peut les lire lorsqu'il dispose des ressources nécessaires pour les traiter. Enfin, Kafka peut également servir d'équilibrer de charge entre plusieurs travailleurs en exploitant sa fonction de partition des sujets. La figure 5.3 montre l'architecture d'un courtier Kafka.

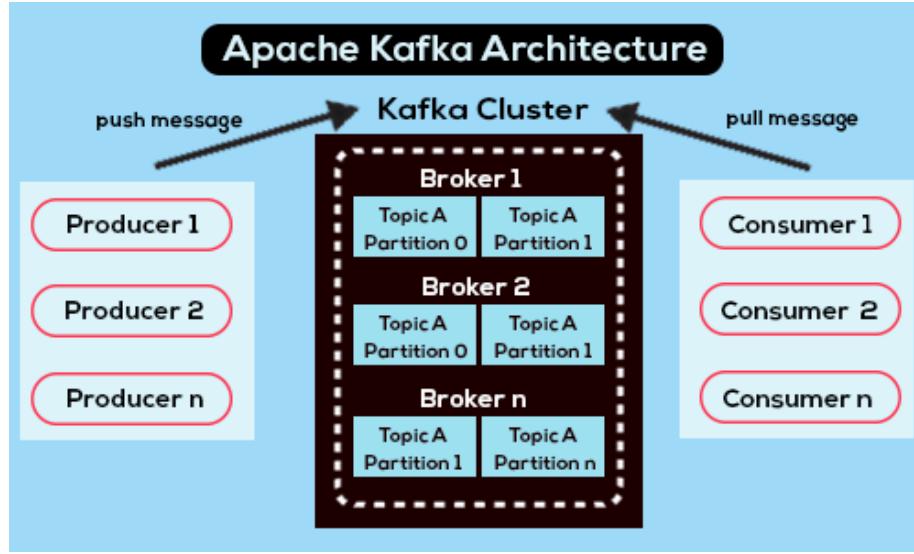


Figure 5.3: Architecture de Apache Kafka

2.2 Topic structure

Les sujets (topics) Kafka sont organisés comme des files d'attente mais ils diffèrent des files classiques car, une fois consommés, les messages ne sont pas rejetés. Au lieu de cela, ils sont conservés en mémoire jusqu'à ce qu'un temps spécifié par l'utilisateur soit écoulé. Cela permet à de nombreux abonnés de consommer le même message dans l'ordre de leur choix. Kafka attribue à chaque message de la file d'attente un indice progressif unique, appelé offset pour l'identifier. Le consommateur doit fournir à Kafka l'index du message qu'il veut lire dans chaque requête. Les topics sont tolérants aux pannes de par leur conception : les messages sont également conservés sur le disque, de sorte que si un plantage appose, ils peuvent être récupérés au prochain démarrage. Les sujets prennent également en charge un mécanisme de division de plusieurs courtiers pour assurer la tolérance aux pannes appelé partitionnement, qui est abordé dans la section suivante dans le cadre du déploiement en cluster de Kafka.

2.3 Clustered deployment

Kafka permet la distribution dans un environnement en cluster, où plusieurs instances du broker permettent de répartir la charge de travail sur les nœuds. La répartition de la charge est basée sur la division des sujets sur plusieurs partitions, chacune contenant un sous-ensemble des messages qui sont publiés sur les sujets.

Pour chaque partition, un courtier différent est élu comme chef de partition et conserve la partition primaire tandis que les autres nœuds du cluster en conservent une copie. Chaque fois qu'un nouvel éditeur rejoint le sujet, il est affecté à une partition primaire du sujet sur le courtier chef de partition comme destination de ses messages. Le mécanisme de partition permet à la fois la tolérance aux pannes et la répartition de la charge : lorsqu'un courtier tombe en panne, un nouveau courtier est élu chef de file des partitions défectueuses et comme il a déjà une copie des messages, cette

opération est très rapide. La répartition de la charge est garantie du côté de l'éditeur en assignant à chaque producteur une partition primaire différente, du côté de l'abonné en lisant les sujets de plusieurs courtiers pour obtenir la file d'attente complète des messages. Le nombre de partitions d'un sujet est un paramètre configurable, non lié au nombre de courtiers dans le cluster : même dans un déploiement à instance unique, il pourrait y avoir plusieurs partitions pour un sujet afin d'exploiter les groupes de consommateurs, discuté dans la prochaine session.

2.4 Parallel processing of the topic

Les partitions de sujets Kafka permettent le traitement d'une file d'attente distribuée par plusieurs abonnés qui peuvent consommer différents sous-ensembles de messages en parallèle. Cela multiplie le débit du système mais nécessite une coordination supplémentaire entre l'abonné et les courtiers puisque chaque message doit être traité exactement une fois par un seul consommateur. Pour ce faire, Kafka organise les abonnés en groupes de consommateurs. Chaque membre du groupe de consommateurs lit une partition du sujet mais l'ensemble du groupe consomme la totalité des messages. Le fait d'avoir plusieurs groupes de consommateurs permet à plusieurs groupes de lire des copies de la file d'attente, mais en même temps de s'assurer que chaque membre du groupe en lit un sous-ensemble, ce qui permet un traitement parallèle des messages.

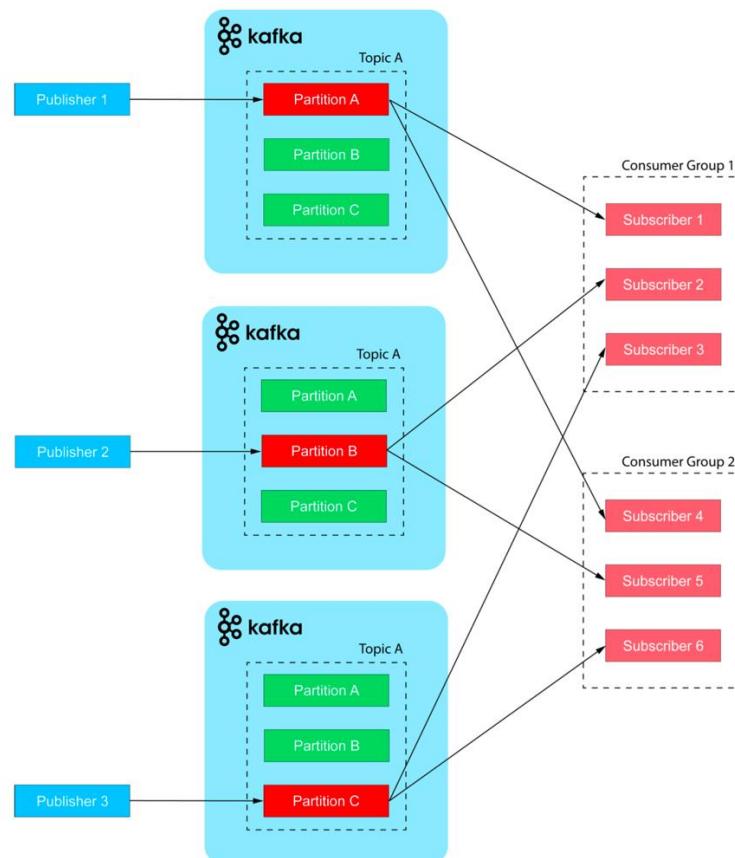


Figure 5.4: Déploiement en cluster d'Apache Kafka. Les partitions surlignées en rouge sont celles du leader.

3. Apache Zookeeper

Apache ZooKeeper est un projet bénévole open source sous l'égide de l'Apache Software Foundation. ZooKeeper est un service centralisé pour maintenir les informations de configuration, le nommage, la synchronisation distribuée et les services de groupe. Tous ces types de services sont utilisés sous une forme ou une autre par les applications distribuées. Chaque fois qu'ils sont implémentés, il y a beaucoup de travail pour corriger les bogues et les conditions de course qui sont inévitables. En raison de la difficulté d'implémentation de ces types de services, les applications en font généralement l'économie au départ, ce qui les rend fragiles en cas de changement et difficiles à gérer. Même lorsqu'elles sont réalisées correctement, les différentes implémentations de ces services entraînent une complexité de gestion lorsque les applications sont déployées.



Figure 5.5: Logo Apache ZooKeeper

4. Apache Spark

Apache Spark est un moteur de traitement de données distribué en mémoire utilisé pour effectuer des calculs généraux sur de grandes quantités de données, communément appelées Big Data. Il fournit une API de programmation pour accéder à ses fonctionnalités depuis Java, Scala et Python. Les calculs peuvent être effectués sur des données non structurées en utilisant une API de bas niveau ou sur des données structurées en utilisant une API de type SQL, appelée SparkSQL. Les principales sources pour cette section sont [31] et [32].

4.1 Challenges in big data processing

Le développement d'applications qui traitent de grandes quantités de données est un défi. Il est difficile de les construire à partir de zéro et il est préférable d'utiliser un cadre dédié comme Spark, car certains problèmes apparaîtront rapidement :

- Sources de données hétérogènes : les données peuvent provenir d'une pléthore de sources différentes, chacune ayant une interface différente pour y accéder, par exemple des bases de données SQL ou No-SQL, des sources non structurées comme les systèmes de stockage d'objets, des flux de données en continu et bien d'autres encore. Spark fournit une interface unifiée pour gérer tous les types de sources de données.
- Taille des données : le terme "big data" fait référence au fait que les données à traiter ne tiennent pas dans la mémoire d'une seule machine, elles doivent donc être réparties sur plusieurs nœuds. Spark fournit une structure de données de base appelée RDD (Resilient Distributed Dataset) qui est conçue pour distribuer de grandes quantités de données sur plusieurs instances Spark de manière tolérante aux pannes. Le grand avantage de cette structure de données est que les données sont conservées en mémoire : cela rend le fonctionnement de Spark très rapide par rapport aux autres structures qui conservent les données sur disque.
- Traitement parallèle des données : Le traitement des données et pas seulement leur stockage, doit également être effectué en parallèle sur plusieurs machines afin d'exploiter toute la puissance disponible du cluster. Spark dispose d'une API qui demande au programmeur de définir les opérations à effectuer sur l'ensemble des données comme si une seule machine traitait les données. Ensuite, le moteur d'optimisation divise automatiquement le travail en petites tâches et détermine quelles opérations peuvent être effectuées en parallèle et gère la distribution des tâches sur les nœuds du cluster.

4.2 Spark cluster

Les calculs Spark ont lieu sur plusieurs nœuds. À l'intérieur du cluster, on peut distinguer trois rôles principaux :

- Driver ou pilote : Cette machine exécute le programme qui est développé par l'utilisateur. Le programmeur peut accéder aux fonctions Spark par le biais d'un objet appelé Spark Session qui fournit un point d'entrée pour toutes les API disponibles et la façon d'accéder au cluster. Le pilote est chargé de planifier la façon d'exécuter le calcul parallèle en divisant les tâches en tâches atomiques qui peuvent être exécutées par les exécuteurs sur les travailleurs. Il contacte le maître pour demander l'allocation de ressources sur les travailleurs afin d'exécuter le calcul. Le pilote peut être le PC du programmeur au stade du développement mais dans les environnements opérationnels, il s'agit généralement d'une machine dédiée.

- Master : Le master est responsable de la coordination du cluster, de la gestion de l'adhésion des travailleurs et de la gestion de leur défaillance éventuelle. Le pilote demande au master d'allouer des ressources aux travailleurs pour effectuer des calculs.
- Workers : Les workers sont responsables de l'exécution des tâches demandées par le pilote. Le pilote demande des ressources au master qui demande aux workers de les exécuter. Un exécuteur est un processus qui peut exécuter des tâches d'étalement, un worker a un exécuteur pour chaque noyau. Une fois les exécuteurs alloués, le pilote les contacte directement pour soumettre les tâches. Une fois le calcul effectué, les résultats sont renvoyés au pilote qui peut mettre fin à la session ou envoyer d'autres tâches.

La figure 5.6 montre les composants du cluster et les deux phases de l'allocation des tâches aux exécutants.

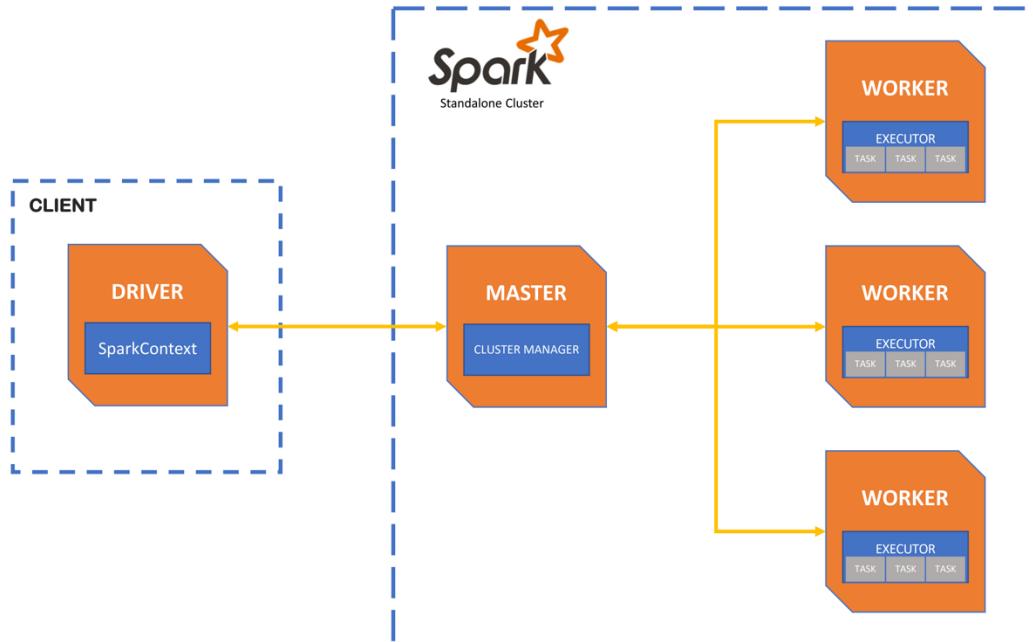


Figure 5.6: Architecture du cluster Spark

4.3 MLeap

MLeap est un format de sérialisation commun et un moteur d'exécution pour les pipelines d'apprentissage automatique. Il prend en charge Spark, Scikit-learn et Tensorflow pour former des pipelines et les exporter vers un Bundle MLeap. Les pipelines sérialisés (bundles) peuvent être déserialisés dans Spark pour une évaluation en mode batch ou dans le runtime MLeap pour alimenter des services API en temps réel.

Mélanger et assortir les technologies ML devient une tâche simple. Au lieu de demander à une équipe entière de développeurs de rendre les pipelines de recherche prêts pour la production, il

suffit d'exporter vers un Bundle MLeap et d'exécuter votre pipeline là où il est nécessaire. Autres avantages d'un runtime unifié :

- Entraînez différents éléments de votre pipeline à l'aide de Spark, Scikit-learn ou Tensorflow, puis exportez-les vers un fichier MLeap Bundle et déployez-le partout.
- Si vous utilisez Scikit pour la R&D mais que Spark propose un meilleur algorithme, vous pouvez exporter votre pipeline Scikit ML vers Spark, entraîner le nouveau modèle dans Spark puis le déployer en production à l'aide du runtime MLeap.

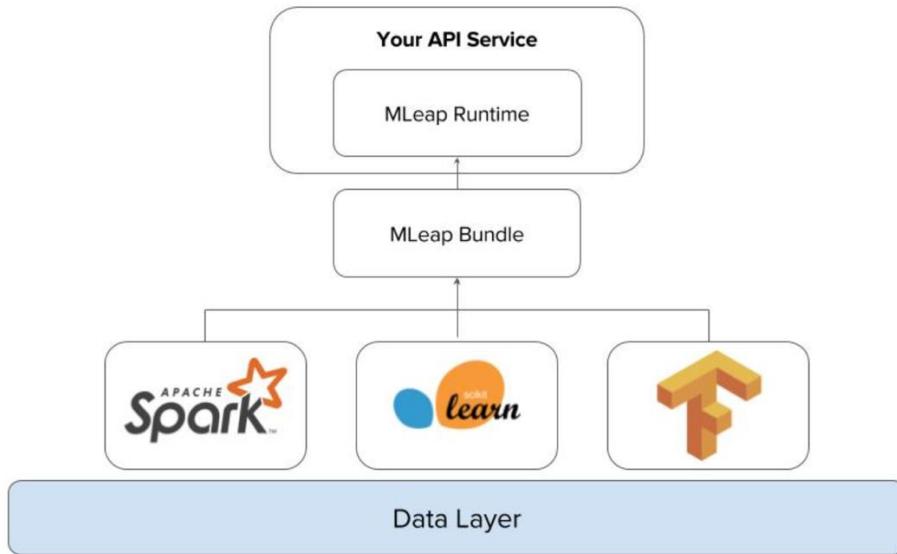


Figure 5.7: Spark, Scikit-learn et Tensorflow: One Runtime

5. Logstash

Logstash est un outil ETL (Extract, Transform and Load), conçu pour faire partie d'un pipeline logiciel car il effectue une étape de transformation intermédiaire dans le traitement des données en continu. L'objectif principal de Logstash est d'extraire les données d'une source, d'appliquer tout type de transformation et enfin de les charger dans le stockage de données persistant prévu. Il est profondément intégré à Elasticsearch pour le stockage et à Kibana pour la visualisation des données. Ces trois logiciels sont maintenus par Elastic sous le nom de pile ELK. Logstash est facilement configurable en définissant un fichier de configuration du pipeline où les trois phases ETL sont définies. La principale ressource pour cette section est [33].

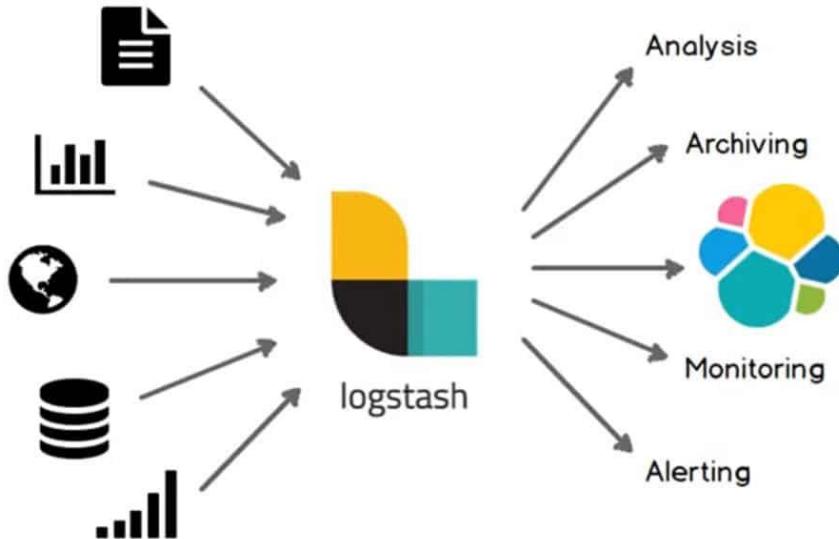


Figure 5.8: Logstash architecture. Image source: [33]

5.1 Data extraction

La première phase du pipeline ETL est la définition des entrées pour l'extraction des données. Logstash est conçu pour travailler avec des sources de données en continu où les données circulent en temps réel dans l'application. Il supporte d'embrée de nombreuses sources de données courantes, comme Apache Kafka, les bases de données SQL, les requêtes HTTP ou les protocoles personnalisés sur TCP/UDP. À partir de chacune de ces sources, il s'attend à avoir un flux d'événements en continu qui seront transmis à la phase de transformation.

5.2 Data transformation

Après la définition des entrées, les événements de streaming commencent à circuler dans Logstash. Les données entrantes peuvent être divisées en deux catégories [34] :

- Les données structurées : les données sont organisées en champs qui peuvent être interprétés par une machine. C'est le cas lorsque les sources sont des bases de données SQL, qui sont organisées en tables, ou JSON et XML.
- Les données non structurées : dans ce cas, les données n'ont aucune structure et bien qu'elles puissent être facilement comprises par un humain, elles sont des chaînes de caractères dénués de sens pour une machine qui ne sait pas comment les analyser. C'est le cas des journaux ou des requêtes HTTP.

Pour utiliser les données pour des tâches telles que l'analyse, l'agrégation ou l'application d'apprentissage automatique, elles doivent être structurées. Pour gérer les sources non structurées, Logstash inclut un moteur d'analyse syntaxique, appelé Grok, qui est utilisé pour dériver une structure à partir de données non structurées. Grok comprend des modèles prêts à l'emploi qui correspondent à la plupart des formats courants, mais il est également configurable manuellement.

en définissant les expressions régulières qui correspondent aux champs à dériver [35]. Pour les cas structurées et non structurées, les données entrantes sont ensuite converties en JSON avant de passer aux phases suivantes. Si aucune analyse syntaxique n'a été appliquée à une source non structurée, un JSON à un champ contenant le contenu de l'événement est émis. Une fois structurées, les données peuvent être manipulées en appliquant l'un des nombreux plugins disponibles pour la phase de transformation, appelé filtres. Les exemples vont d'opérations simples comme l'ajout de champs, l'horodatage ou la suppression de données sensibles, à des transformations plus complexes comme l'anonymisation des données, où un identifiant unique remplace l'identifiant de l'utilisateur, ou la géolocalisation IP, où les adresses IP sont remplacées par la localisation géographique.

5.3 Data load

Après la transformation des données, Logstash les envoie au stockage de données désigné, appelé stash, sous forme de documents JSON. Bien qu'Elasticsearch soit le stash le plus utilisé car il est conçu pour fonctionner sans problème avec Logstash, les données peuvent être envoyées vers un large éventail de destinations, comme des bases de données SQL, des buckets S3 ou même repoussées vers une autre file d'attente Kafka.

5.4 Parallel processing

La spécification du pipeline ETL est complètement apatride : chaque événement est une menace indépendante des précédents et des suivants et aucune information en dehors de celle contenue dans l'événement lui-même n'est requise pour appliquer la phase de transformation. Cela permet de traiter le flux de données en parallèle par plusieurs instances de Logstash sans exiger aucun type de mécanisme de coordination entre les répliques. Pour cette raison, Logstash n'inclut aucun mécanisme pour former et maintenir un cluster. Pour exploiter le parallélisme, la source de données doit veiller à ce que chaque réplique reçoive des sous-ensembles différents de l'événement afin d'éviter le traitement en double en utilisant un mécanisme comme les partitions Kafka, abordées à la section 2.

5.5 Grok

Logstash grok n'est qu'un type de filtre qui peut être appliqué à vos journaux avant qu'ils ne soient transmis à Elasticsearch. Comme il joue un rôle crucial dans le pipeline de journalisation, grok est également l'un des filtres les plus utilisés.



Figure 5.9: Logo Logstash Grok

6. Elasticsearch

Elasticsearch est un moteur de recherche distribué, conçu pour stocker des documents JSON et optimisé pour effectuer des recherches plein texte et des regroupements complexes sur les données stockées. Ses principaux cas d'utilisation sont l'analyse des journaux, la recherche sur le Web et la surveillance des infrastructures. La principale ressource pour cette section est [36].

6.1 Elasticsearch architecture

L'architecture d'Elasticsearch est similaire à celle d'une base de données No-SQL, même si elle ne fournit pas toutes les fonctions d'une base de données, comme les transactions. Les entrées dans Elasticsearch sont des documents JSON sans schéma, qui sont chargés déjà analysés et organisés en champs à partir d'applications d'ingestion comme Logstash, présentées dans les sections précédentes. Les documents JSON ont un grand avantage expressif par rapport à l'organisation classique en champs tabulaires : ils permettent de stocker n'importe quel type de structure de données, comme des tableaux ou des objets imbriqués, sans avoir besoin de les normaliser en une structure à une colonne et une valeur. Puisque l'objectif principal d'Elasticsearch est de fournir une recherche sur des documents, les champs des documents sont des index de texte pour fournir des capacités de recherche en temps quasi réel. Les documents ayant quelque chose en commun sont organisés en index, comme les lignes SQL sont organisées en tables. Les index sont plus flexibles que les tables, car Elasticsearch peut en gérer des milliers sans perdre en performance : il est courant dans les applications de journalisation d'avoir un index séparé pour chaque jour, car la recherche peut être effectuée entre plusieurs index. L'ingestion de données et l'interrogation à partir d'applications externes sont rendues possibles par une API HTTP REST.

6.2 Difference from a database

Malgré sa structure de type No-SQL, Elasticsearch n'est pas destiné à être une base de données car il lui manque l'un des principaux avenirs d'une base de données : les transactions. Une transaction est une opération sur les données stockées dans une base de données, comme une insertion ou une mise à jour, qui doit avoir certaines propriétés précises, résumées dans l'acronyme ACID (Atomicité, Cohérence, Intégrité, Durabilité) [37]. Ces propriétés permettent de s'assurer que les opérations concurrentes sur les documents ne créent pas d'incohérence dans la base de données. La motivation principale de ce manque est qu'Elasticsearch n'est pas destiné à être une base de données primaire avec des enregistrements qui sont mis à jour fréquemment, donc avoir un mécanisme de transaction complet sur un cluster qui est conçu pour évoluer horizontalement sur des centaines de nœuds rendra les opérations lentes puisque les transactions distribuées nécessitent une quantité énorme de communication inter-nœuds. Au lieu de cela, Elasticsearch est optimisé comme un moteur de recherche en temps quasi réel et l'accent est donc mis sur la vitesse de lecture, une opération qui ne pose aucun problème de cohérence lorsqu'elle est exécutée simultanément par de nombreux clients. Une autre fonctionnalité manquante est celle des relations mais même si cette fonctionnalité est absente de la plupart des bases de données No-SQL, les jointures peuvent être implémentées par l'utilisateur dans la plupart d'entre elles. Sur Elasticsearch, il n'y a aucun moyen d'avoir quelque chose de similaire, car les jointures sont les opérations les plus coûteuses sur une base de données, et cela aurait un impact trop important sur la vitesse de recherche. Étant donné qu'Elasticsearch est souvent utilisé en conjonction avec une base de données classique pour

disposer de capacités de recherche et d'agrégation rapides sur les données stockées [38], pour émuler une véritable jointure, les relations entre les tables sont dénormalisées : cela entraînera des données en double mais permettra une vitesse maximale. Elasticsearch peut également être utilisé comme stockage primaire dans le cas de charges de travail principalement en lecture seule, où la vitesse de récupération et d'agrégation est critique, comme dans le cas de l'analyse des journaux ou de la surveillance d'une autre application.

6.3 Clustered deployment

Bien qu'Elasticsearch puisse être utilisé en mode autonome, il est surtout utilisé dans le cadre d'un déploiement en cluster : il est courant qu'une configuration distribuée contienne des pétaoctets de données dans des centaines de nœuds. Le mécanisme de réPLICATION est basé sur les "shards", qui sont des sous-ensembles autonomes de documents appartenant au même index logique. Un shard peut être :

- Primaire : lorsque le shard est la copie principale des documents. Lorsqu'une recherche est effectuée, seuls les shards primaires sont ciblés. Pour chaque shard, il n'y en a qu'un seul qui est primaire.
- Secondaire : ce sont des copies de sauvegarde des shards primaires, qui sont conservées sur différents nœuds du cluster. Chaque shard est indexé indépendamment des autres parties du même index, ce qui fait que l'index n'est qu'un regroupement logique abstrait de shards. Les coordinateurs de cluster Elasticsearch appelés maîtres, sont chargés d'équilibrer les shards sur les nœuds du cluster afin d'assurer une disponibilité et une résilience maximales en cas de perturbation des primaires, ainsi que de gérer la création d'index et l'adhésion de nouveaux nœuds. En cas de défaillance d'un maître, un autre maître est élu entre les nœuds et le shard secondaire est promu au rang de primaire et équilibré sur le cluster. Elasticsearch prend en charge un autre système de réPLICATION, appelé réPLICATION inter-clusters. Dans ce cas, un site secondaire dédié réplique le site primaire : en cas d'indisponibilité du centre de données principal, le site de secours prend sa place.



Figure 5.10: Elasticsearch Cluster

6.4 Search and aggregation

Lorsque de nouveaux documents sont ajoutés à Elasticsearch, ils sont automatiquement indexés. Les index sont une structure de données qui permet de récupérer rapidement des documents spécifiques sans avoir à effectuer un balayage complet de l'ensemble du magasin pour trouver les données souhaitées. Les index d'Elasticsearch sont optimisés pour effectuer des recherches en texte intégral sur les champs des documents, contrairement aux bases de données où les index sont optimisés pour récupérer la clé numérique afin de réaliser des jointures rapides entre les tables. Le texte intégral est le type de recherche le plus courant effectué par les humains lorsqu'ils traitent des données : l'utilisateur fournit une liste de mots clés et le moteur fournit les documents qui y correspondent dans un ou plusieurs champs indexés. La structure de données utilisée par Elasticsearch pour indexer les documents stockés est appelée index inversé [39] : chaque mot qui apparaît dans un index particulier est associé aux identifiants uniques des documents où ce mot apparaît. Lorsque l'utilisateur fournit un mot-clé, il suffit de trouver l'entrée correcte dans la liste de mots, puis de récupérer les documents associés. Elasticsearch est également optimisé pour effectuer l'agrégation, une opération cruciale dans l'analyse des données, sur les documents correspondants : cela signifie appliquer des fonctions telles que la somme, le compte ou la moyenne pour obtenir une valeur à partir de plusieurs documents.

7. Kibana

Kibana est un cadre de visualisation de données faisant partie de la pile ELK. Il présente une interface Web permettant de visualiser des tableaux de bord qui peuvent être enrichis de widgets tels que des graphiques, des tableaux ou même des cartes pour afficher des données localisées. Il nécessite Elasticsearch comme source et moteur pour récupérer et agréger les données à afficher. Kibana est également utilisé comme interface graphique pour surveiller et gérer tous les composants de la pile ELK. La principale ressource pour cette section est [40].

7.1 Widget and Dashboard

Kibana facilite l'exécution de requêtes et d'agrégations complexes à partir d'une interface web graphique contre des documents Elasticsearch. La fonction Lens permet de créer tout type de visualisation de données en sélectionnant l'index à visualiser, en faisant glisser le champ sur le graphique et en sélectionnant la fonction d'agrégation à utiliser. Une fois créé, le widget doit être ajouté à un tableau de bord pour être utilisé. Les tableaux de bord sont des collections de widgets qui peuvent être personnalisés pour répondre à des cas d'utilisation particuliers ou aux besoins de différents utilisateurs. Kibana peut répondre à la fois aux besoins d'utilisateurs experts, ayant une formation en science des données, et d'utilisateurs non experts, qui peuvent n'avoir besoin que d'informations de base : plusieurs tableaux de bord peuvent être créés pour différencier les utilisateurs, à la fois par expertise et par rôle au sein de l'organisation. Par exemple, l'équipe informatique peut accéder aux données relatives aux performances de l'infrastructure, tandis que les analystes peuvent accéder aux informations sur les flux.

7.2 Discover

Outre la vue du tableau de bord, Kibana permet aux utilisateurs experts d'effectuer une analyse libre des données grâce à la fonction Discover : les analystes de données peuvent sélectionner l'indice à visualiser et effectuer une agrégation dans le temps avec des fonctions comme la somme, la moyenne, la variance, la corrélation et bien d'autres. Cela peut s'avérer particulièrement utile pour effectuer une analyse préliminaire en vue du développement de modèles d'apprentissage automatique, directement sur l'ensemble des données et en temps réel. En outre, les requêtes et les agrégations conçues dans Discover peuvent être facilement converties en un widget qui peut être mis à la disposition des utilisateurs non experts.

7.3 ELK stack monitor and management

Kibana sert également d'interface web centralisée pour surveiller et configurer l'ensemble de la pile élastique. Deux sections principales sont disponibles :

- Gestion de la pile : Permet de gérer tous les principaux éléments à l'intérieur de la pile ELK, comme les rôles et priviléges des utilisateurs, les index Elasticsearch, les pipelines Logstash, les paramètres de sécurité et de sauvegarde, le tout dans une seule interface web.
- Surveillance de la pile : Permet de surveiller l'ensemble de la pile ELK à partir d'une seule interface Web. Les données comprennent des mesures courantes telles que l'utilisation de la mémoire et du disque, la charge du processeur et des mesures personnalisées spécifiques aux applications surveillées. En outre, une ventilation détaillée des pipelines Logstash est disponible, afin de détecter les problèmes dans l'application des filtres aux données. La collecte des métriques est effectuée par Metricbeat, présenté dans la section suivante, et les données acquises sont stockées dans un index Elasticsearch dédié.

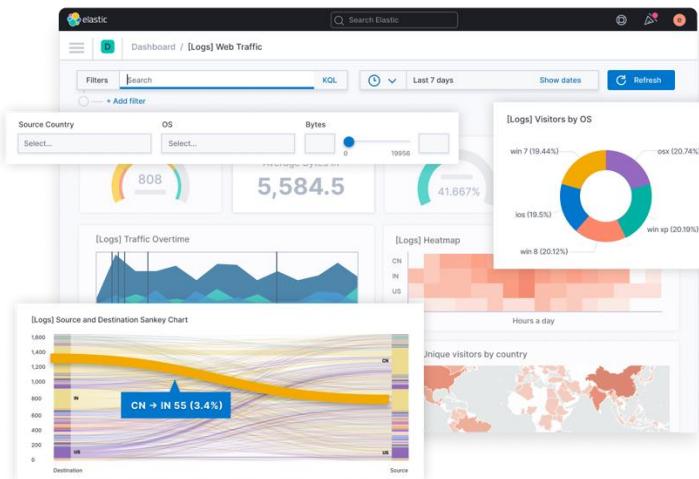


Figure 5.11: Exemple Dashboard Kibana

8. Beats

Beats est une famille d'expéditeurs de données légères, développée par Elastic en tant que module complémentaire de la pile ELK. Un expéditeur de données est un agent qui peut être installé sur un serveur qui exécute une autre application pour surveiller le système d'exploitation et les programmes qui s'exécutent au-dessus. Les données collectées peuvent être envoyées directement à Elasticsearch ou publiées sur un courtier de messages comme Apache Kafka, puis traitées avec un outil ETL comme Logstash. Pour notre travail, deux applications, Filebeat et Metricbeat, sont utilisées. La principale ressource pour cette section est [41].

8.1 Filebeat and Metricbeat

Filebeat et Metricbeat sont des agents qui collectent les données produites par d'autres applications sur le même serveur où ils sont installés. Ils collectent deux types de données : Filebeats collecte les journaux, qui sont des enregistrements des opérations effectuées par une application comme par exemple l'accès à un système, tandis que Metricbeat collecte les métriques des applications, qui sont des statistiques d'utilisation comme la RAM ou le CPU utilisés, l'espace disque disponible, etc.

Les deux systèmes peuvent être facilement configurés en spécifiant la destination des données collectées et le format des données à collecter. Filebeat et Metricbeat possèdent des dizaines de modules qui permettent de collecter et d'analyser de nombreuses sources de données courantes : des applications de serveur web comme Apache ou Nginx, des bases de données comme MySQL, des moteurs de cloud computing comme AWS ou Google Cloud. La pile ELK elle-même peut être surveillée par une pile de surveillance ELK dédiée : un déploiement courant de ces deux applications les utilise en combinaison avec Elasticsearch pour le stockage et Kibana pour la visualisation. Si les serveurs à surveiller sont nombreux ou disposent de peu de ressources, un courtier comme Kafka est employé pour la mise en file d'attente et l'analyse et l'injection sont ensuite déchargées vers Logstash.

9. Lucidchart

Lucidchart est un outil en ligne très complet pour réaliser facilement des diagrammes, des organigrammes, des schémas et autres cartes mentales. Il est simple d'utilisation et présente des fonctionnalités très poussées.

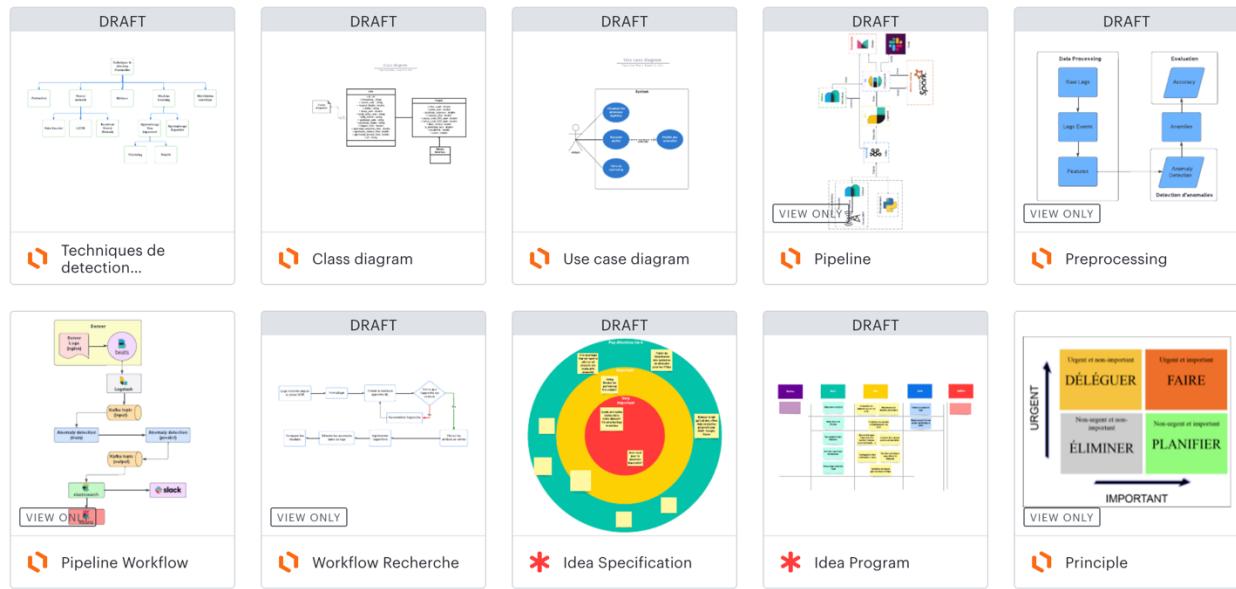


Figure 5.12: Quelques graphes modélisés avec l'outil

10. Technologies

10.1 Jupyter Notebook

L'application Jupyter Notebook est une application serveur-client qui permet d'éditer et d'exécuter des documents notebook via un navigateur web. L'application Jupyter Notebook peut être exécutée sur un bureau local ne nécessitant aucun accès à Internet ou peut être installée sur un serveur distant et accessible par Internet.

C'est un outil de préférence des data scientists car, il offre une bonne présentation des données, un codage collaboratif mais aussi une visualisation assez complète pour passer une première analyse des résultats. Cependant, dans le cadre des traitements de données, l'application devient très lente si le jeu de données est volumineux car étant directement reliée aux ressources de l'ordinateur hôte.



Figure 5.13: Logo Jupyter Notebook

10.2 Python

C'est est un langage de programmation puissant et facile à utiliser. Il dispose de structures de données de haut niveau et permet une approche simple mais efficace de la programmation orientée objet. Il est très simple d'utilisation et est très adapté pour le traitement de données.



Figure 5.14: Logo Python

10.3 Scala

C'est un langage de programmation à plusieurs paradigmes sortis en 2004, scala profite d'une syntaxe ultra-épurée qui permet de faire énormément de choses en très peu de lignes de code. Son avantage est qu'il combine POO (programmation orientée objet) et PF (programmation fonctionnelle) de manière parfaitement harmonieuse. Aussi, il est la base de création de plusieurs outils dont apache spark.



Figure 5.15: Logo Scala

10.4 Maven

Maven est un outil open source de la communauté Apache entièrement écrit en Java. Il permet d'automatiser la gestion et la construction d'un projet Java ; ce que l'on appelle communément un outil de build. Il permet notamment d'automatiser certaines tâches : compilation, tests unitaires et déploiement des applications qui composent le projet, de gérer des dépendances vis-à-vis des bibliothèques nécessaires au projet et aussi de générer des documentations concernant le projet.



Figure 5.16: Logo Maven

10.5 SBT

À l'instar de maven, SBT (Simple Build Tool) est aussi un outil de build. Il se veut plus simple et son objectif est de fournir des fonctionnalités basiques et avancées très simples à implémenter sur un projet Scala ou Java.



Figure 5.17: Logo Sbt

11. Summary

Dans cette section, nous avons décrit les applications qui sont utilisées dans notre travail. Dans le chapitre suivant, nous décrivons comment les applications présentées sont utilisées dans l'architecture de pipeline d'apprentissage automatique proposée.

II. Présentation du pipeline logiciel pour l'apprentissage machine

Le pipeline est entièrement développé en utilisant Docker comme outil de virtualisation, pour permettre un développement rapide dans un environnement aux ressources limitées. Chaque application du pipeline est déjà disponible sous forme d'image sur le registre public, de sorte qu'aucun effort d'installation ne soit nécessaire. Pour un démarrage rapide et facile de l'ensemble du cluster, un fichier docker-compose est fourni avec les spécifications complètes du pipeline. Des détails sur Docker peuvent être trouvés dans la section 5.1.

Ci-dessous on a une première image qui spécifie un peu notre fichier de configuration Filebeat pour l'ingestion des données depuis le serveur DER.

Chapitre 5 : Implémentation et présentation de la solution

```
GNU nano 4.8 log-elastic.conf
input {
  beats {
    # The port to listen on for filebeat connections.
    port => 5044
    # The IP address to listen for filebeat connections.
  }
}

filter {
  grok {
    match => { "message" => "\'%{HTTPDATE:timestamp}\'" %{WORD}=%{IPORHOST:client} %{WORD}=(?:%{WORD:method}|-) %{WORD}=(?:(%{WORD:methodUrl}|-) %{NOTSPACE:u
  }
}

output {
  elasticsearch {
    hosts => ["http://18.222.56.201:9200"]
    user => "elastic"
    password => "ianwyY00bkck13TFJvd6"
    manage_template => false
    index => "log_ingest"
  }
}
```

Figure 5.18: Filebeat Configuration

Ces données sont par la suite indexées ce qui permet de les identifier et y appliquer les traitements qui vont suivre.

The screenshot shows the Elasticsearch Index Management interface. On the left, there is a sidebar with navigation links for Management, Ingest, Data, Alerts and Insights, and Security. The main area is titled 'Index Management' and contains tabs for Indices, Data Streams, Index Templates, and Component Templates. Under the Indices tab, there is a search bar and filters for Lifecycle status and Lifecycle phase. A table lists various indices with columns for Name, Health, Status, Primaries, Replicas, Docs count, Storage size, and Data stream. The indices listed include elastalert_status, elastalert_status_silence, elastalert_status_error, log_ingest, elastalert_status_status, elastalert_status_past, metrics-endpoint.metadata_current_de fault, and der_accesslog. Most indices are in a yellow 'warning' state, except for metrics-endpoint.metadata_current_de fault which is green. The table has 10 rows per page.

| Name | Health | Status | Primaries | Replicas | Docs count | Storage size | Data stream |
|--|----------|--------|-----------|----------|------------|--------------|-------------|
| elastalert_status | ● yellow | open | 1 | 1 | 3 | 93kb | |
| elastalert_status_silence | ● yellow | open | 1 | 1 | 3 | 12.3kb | |
| elastalert_status_error | ● yellow | open | 1 | 1 | 19 | 18kb | |
| log_ingest | ● yellow | open | 1 | 1 | 14972 | 10.3mb | |
| elastalert_status_status | ● yellow | open | 1 | 1 | 9 | 34.7kb | |
| elastalert_status_past | ● yellow | open | 1 | 1 | 0 | 226b | |
| metrics-endpoint.metadata_current_de fault | ● green | open | 1 | 0 | 0 | 226b | |
| der_accesslog | ● yellow | open | 1 | 1 | 2024 | 429.8kb | |
| | | | | | | | |

Figure 5.19: Page de gestion des index

Avec kibana, une visualisation sous forme de diagramme et de graphe est faite d'une manière assez simple offrant ainsi une capacité d'analyse avancée des données.

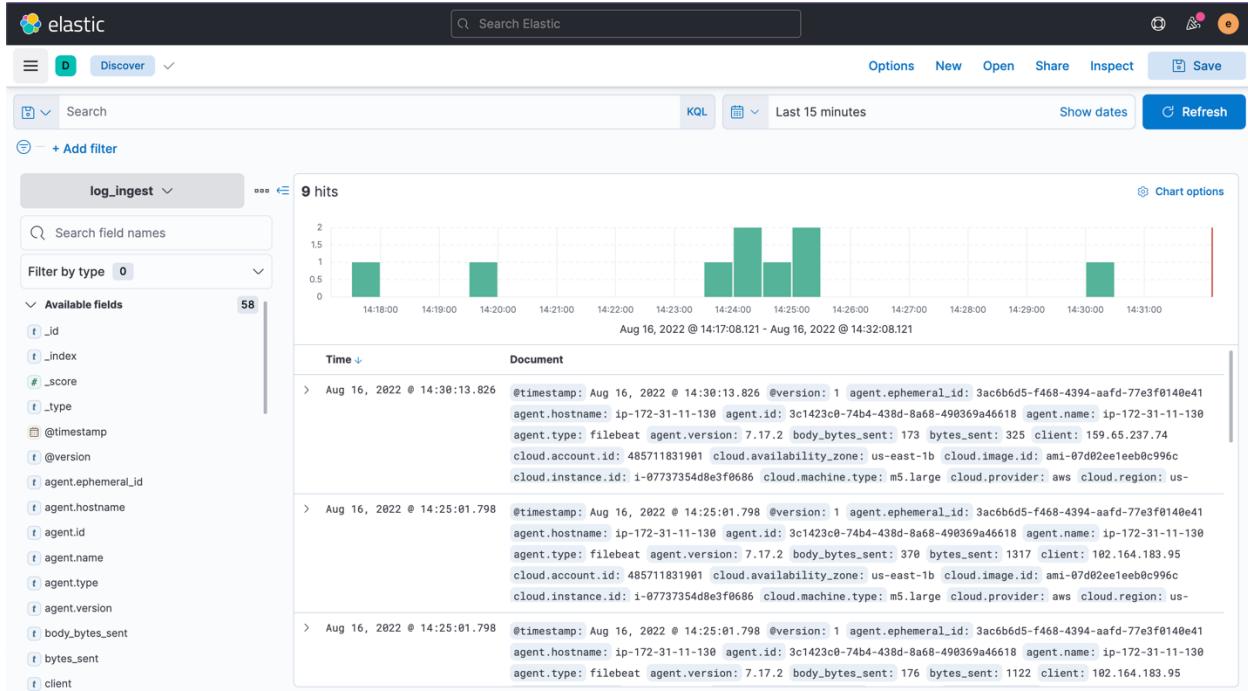


Figure 5.20: Données ingérées depuis le serveur

La construction d'un modèle semble être la partie la plus simple de la plupart des projets de science des données. Le déploiement des modèles en production est toutefois un défi en raison des exigences des systèmes de production des entreprises. Le déploiement des modèles doit prendre en compte la mise à l'échelle et l'équilibrage de la charge afin de respecter les accords de niveau de service (SLA) en fonction de la demande et doit s'intégrer aux systèmes de données en amont et aux consommateurs de modèles en aval. Bien que la gestion d'un modèle distribué soit une tâche d'ingénierie extrêmement complexe

- gérer un modèle distribué,
- noter les données en temps réel
- avec des garanties telles que l'évaluation de chaque requête exactement une fois
- y compris la tolérance aux pannes et la récupération

restent des exigences très courantes dans les déploiements d'entreprise. En intégrant notre modèle dans une application Kafka Streams, nous répondons à toutes ces exigences en héritant simplement des fonctionnalités d'entreprise de Kafka. Le workflow suivant montre comment intégrer un modèle de science des données construit et entraîné dans Spark en tant qu'application de streaming Kafka pour un scoring en temps réel.

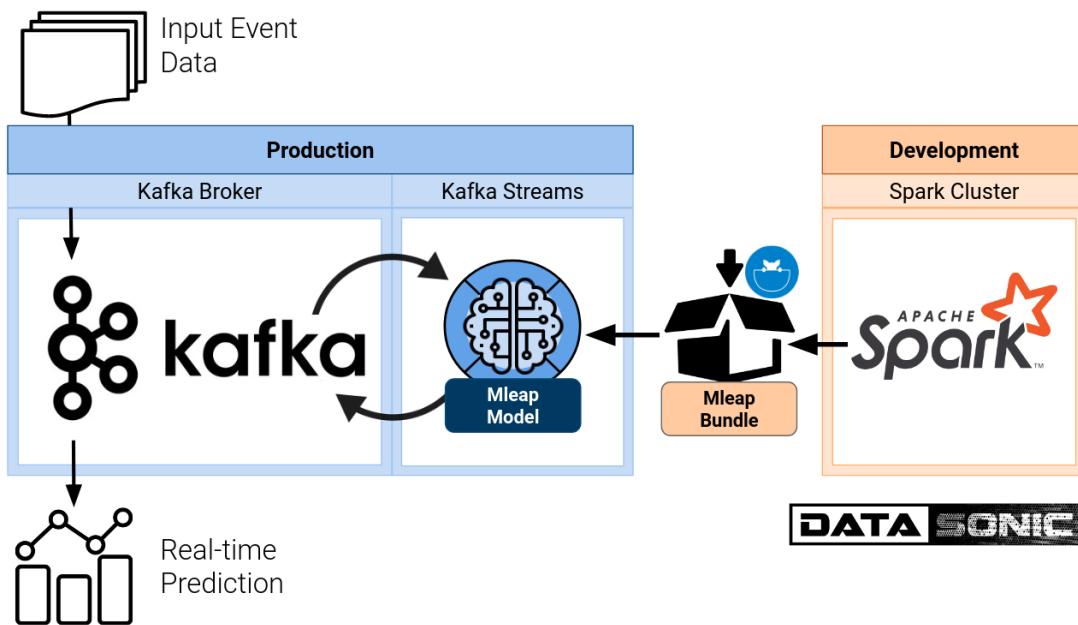


Figure 5.21: Workflow Application Streaming

Pour déployer un pipeline Spark en tant qu'application de streaming Kafka, nous utilisons le projet Mleap pour sérialiser notre pipeline Spark sans avoir besoin d'un contexte Spark. Cela nous génère ainsi notre modèle bundle avec l'architecture suivante :

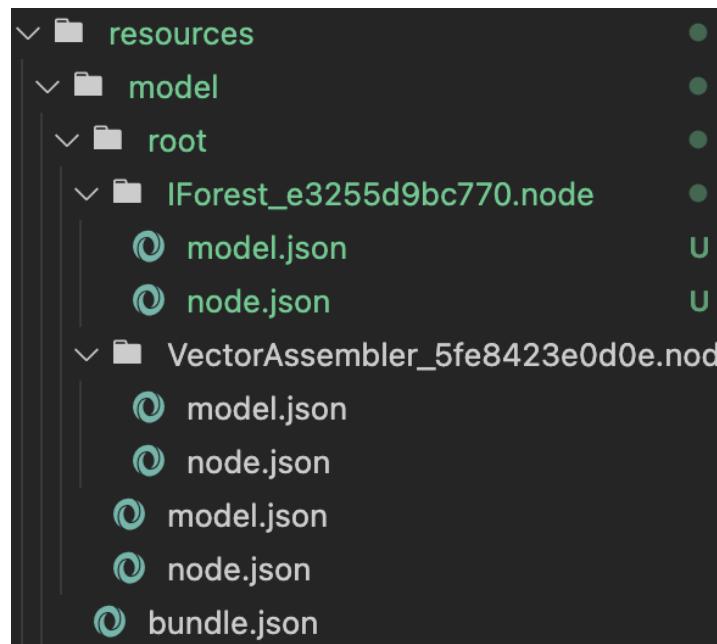
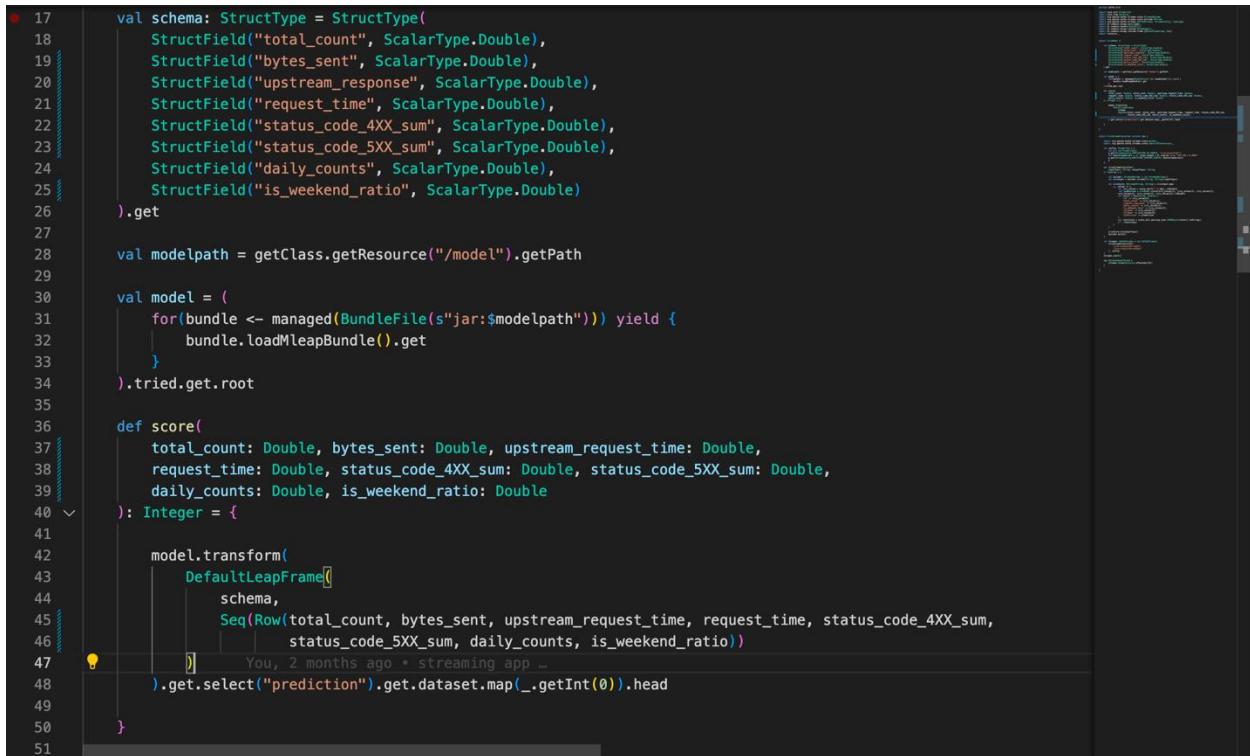


Figure 5.22: Modèle bundle généré depuis Spark

Dans la figure 5.23, on montre un petit aperçu de notre application streaming fait avec scala.



```

17 val schema: StructType = StructType(
18   StructField("total_count", ScalarType.Double),
19   StructField("bytes_sent", ScalarType.Double),
20   StructField("upstream_response", ScalarType.Double),
21   StructField("request_time", ScalarType.Double),
22   StructField("status_code_4XX_sum", ScalarType.Double),
23   StructField("status_code_5XX_sum", ScalarType.Double),
24   StructField("daily_counts", ScalarType.Double),
25   StructField("is_weekend_ratio", ScalarType.Double)
26 ).get
27
28 val modelpath = getClass.getResource("/model").getPath
29
30 val model = (
31   for(bundle <- managed(BundleFile(s"jar:$modelpath")))
32     yield {
33     bundle.loadMLeapBundle().get
34   }
35 ).tried.get.root
36
37 def score(
38   total_count: Double, bytes_sent: Double, upstream_request_time: Double,
39   request_time: Double, status_code_4XX_sum: Double, status_code_5XX_sum: Double,
40   daily_counts: Double, is_weekend_ratio: Double
41 ): Integer = {
42
43   model.transform(
44     DefaultLeapFrame(
45       schema,
46       Seq(Row(total_count, bytes_sent, upstream_request_time, request_time, status_code_4XX_sum,
47                 status_code_5XX_sum, daily_counts, is_weekend_ratio))
48     )
49   ).get.select("prediction").get.dataset.map(_.getInt(0)).head
50 }
51

```

Figure 5.23: Application Streaming

Les données prédites sont par la suite sauvegardées sur Elasticsearch et c'est en moment que notre système d'alerte entre en jeu en prenant les inputs avec les scores les plus élevés, faire une correspondance de chaque id et envoyer les alertes sur notre canal Slack.

Chapitre 5 : Implémentation et présentation de la solution

The screenshot shows the Elasticsearch Stack Management interface. The left sidebar includes sections for Rollup Jobs, Transforms, Remote Clusters, Alerts and Insights, Security, Kibana, Index Patterns, Stack, and a general navigation bar with Stack Management, Index patterns, and der_accesslog.

The main area displays the 'der_accesslog' index pattern. It shows 18 fields: _id, _index, _score, _source, _type, bytes_sent, daily_counts, id, is_weekend_ratio, and remote_addr. The table includes columns for Name, Type, Format, Searchable, Aggregatable, and Excluded.

| Name | Type | Format | Searchable | Aggregatable | Excluded |
|------------------|---------|--------|------------|--------------|----------|
| _id | _id | | ● | ● | ○ |
| _index | _index | | ● | ● | ○ |
| _score | | | | | ○ |
| _source | _source | | | | ○ |
| _type | _type | | ● | ● | ○ |
| bytes_sent | double | | ● | ● | ○ |
| daily_counts | double | | ● | ● | ○ |
| id | long | | ● | ● | ○ |
| is_weekend_ratio | double | | ● | ● | ○ |
| remote_addr | text | | ● | | ○ |

Figure 5.24: Outputs algorithme de détection

The screenshot shows a Slack channel named 'anomalyalert'. The left sidebar lists channels like PFE, TX, and N, along with Direct messages and Add teammates. The main area shows two messages from the 'elastalert' bot:

upstream_addr= upstream_status= request_time=0.000 upstream_time=123 bbb
At least 2 events occurred between 2022-08-20 01:49 UTC and 2022-08-20 01:54 UTC
@timestamp: 2022-08-20T01:54:21.18Z
@version: 1
_id: QKf1ulBb-HP2xEZVVVTZ
_index: log_ingest
_type: _doc
Show more

elastalert APP 2:16 AM
Anomaly Detected
DER Alert
At least 2 events occurred between 2022-08-20 02:10 UTC and 2022-08-20 02:15 UTC
@timestamp: 2022-08-20T02:15:55.26Z
@version: 1
_id: ZazJYlBb-HP2xEZGVQe
_index: log_ingest
_type: _doc
Show more

Figure 5.25: Output des résultats sur Slack

CONCLUSION GÉNÉRALE

I. Synthèse des travaux

Cette étude permet de trouver une solution optimale au problème de la détection d'anomalies dans des fichiers journaux variés et volumineux. Bien qu'il y ait eu de nombreuses recherches et études réalisées sur ce sujet, ce projet a étudié la plupart des moyens possibles pour résoudre ce problème. La Q1 a été répondue en faisant une revue de la littérature, la Q2 a été faite en conduisant une expérience sur les différents modèles d'apprentissage automatique qui pourraient s'adapter le mieux et donner une meilleure solution à ce problème. L'algorithme d'Isolation Forest semble être une solution optimale pour ce projet. Même si cela nous a donné de bons résultats, le déploiement de cette méthode pour un usage réel a été un peu difficile. En effet, lors de la réalisation du projet, beaucoup de travail manuel a été effectué.

Tout d'abord, l'application de la méthode d'analyse appropriée afin qu'il n'y ait aucune perte d'informations utiles dans les fichiers journaux. Deuxièmement, les journaux sont étudiés pour ne prendre que la partie du journal qui est réellement utile pour cela, car les données du journal contiennent toutes les informations parfois qui pourraient ne pas être utiles ou qui ne contiennent aucune information concernant les anomalies. La prise en compte de ces informations ne donne que de fausses prédictions sur les données normales, ce qui est encore plus désastreux. Cependant, la technique d'analyse syntaxique appropriée et les modèles d'apprentissage automatique ont été choisis après une étude approfondie de la littérature et sous la direction des superviseurs internes et externes.

II. Appart du stage

Cette dernière immersion de fin d'étude a été très enrichissante. Elle nous a permis de découvrir en profondeur le domaine de l'AIOps, un domaine très ouvert et en plein essor. Ce stage nous a fait découvrir la Recherche & Développement à travers les expérimentations, les articles et thèses consultés avant de passer au développement.

Au terme de ce stage, la montée en compétence était bien perceptible au travers du travail réalisé. Nous avons d'abord compris les enjeux autour des SI avant de se pencher dans l'automatisation du processus de gestion des dysfonctionnements en nous imprégnant des concepts tels que l'AIOps. Nous avons également acquis des connaissances poussées sur l'apprentissage automatique ainsi que les technologies du Big Data telles que Spark, Apache Kafka aussi avons exploré la stack ELK.

En somme, ce stage a été d'une grande aide pour l'aboutissement de notre formation car elle a permis l'acquisition d'une expérience solide dans les nouvelles technologies et pourra faciliter notre insertion dans le monde de l'entreprise.

III. Perspectives

Comme travail futur, développer une méthode ou un outil qui pourrait effectivement prendre la partie utile des fichiers journaux et ensuite donner la priorité en fonction du fichier journal qui est généré après l'exécution réussie des tests matériels ou logiciels. Fournir une solution sur la façon d'éviter les anomalies survenues après avoir trouvé où elles ont été causées serait une meilleure amélioration pour ce travail.

Par ailleurs, notre solution apporte des améliorations satisfaisantes sur la gestion des dysfonctionnements. En effet, elle propose non seulement un espace pour la visualisation et l'analyse des données mais permet également d'identifier une anomalie en temps quasi réel en faisant passer le temps d'identification de la cause d'un dysfonctionnement de quelques heures à quelques secondes.

RÉFÉRENCES

1. Anomaly Detection [En Ligne] Disponible :
<https://www.gavstech.com/anomaly-detection-in-aiops/>
2. What is Elasticsearch? [En Ligne] Disponible:
<https://www.elastic.co/guide/en/elasticsearch/reference/8.1/elasticsearch-intro.html>
3. Elastic Machine Learning [En Ligne] Disponible:
<https://www.elastic.co/guide/en/machine-learning/current/machine-learning-intro.html#machine-learning-intro>
4. Nginx [En Ligne]. Disponible:
http://nginx.org/en/docs/http/ngx_http_log_module.html#log_format
5. Mohiuddin Solaimani, “Anomaly Detection for Application Log Data” (2015). Thèse de doctorat, University of Texas at Dallas, Disponible:
https://scholarworks.sjsu.edu/cgi/viewcontent.cgi?article=1640&context=etd_projects
6. Machine learning on Elastic Search using Apache Spark and ES-Hadoop [En Ligne]. Disponible: <https://excelerate.systems/machine-learning-elasticsearch/>
7. Apache Spark Tutorial [En ligne]. Disponible:
<https://www.tutorialkart.com/apache-spark>
8. User Agent Anomaly Detection [En Ligne] Disponible:
<https://slideplayer.com/slide/13693768/>
9. <https://posts.specterops.io/threat-hunting-with-jupyter-notebooks-part-1-your-first-notebook-9a99a781fde7>
10. <https://opensource.com/article/19/5/log-data-apache-spark>
11. Spark MLlib [En Ligne] Disponible: <https://spark.apache.org/docs/latest/ml-features>
12. Machine Learning Pipeline [En Ligne] Disponible: <https://valohai.com/machine-learning-pipeline/>

13. Machine Learning Model [En Ligne] Disponible:
<https://www.youtube.com/watch?v=Om7G2qRBSVw>
14. Embed a Spark ML Model [En Ligne] Disponible:
<https://towardsdatascience.com/how-to-embed-a-spark-ml-model-as-a-kafka-real-time-streaming-application-for-production-deployment-933aecb79f3f>
15. Complete Data Science Project Template (Tres utile pour la presentation) [En Ligne] Disponible:
<https://towardsdatascience.com/complete-data-science-project-template-with-mlflow-for-non-dummies-d082165559eb>
16. XGBoost [En Ligne] Disponible:
<https://towardsdatascience.com/pyspark-and-xgboost-integration-tested-on-the-kaggle-titanic-dataset-4e75a568bdb>
17. Reason to embed model in kafka instead of remote [En Ligne] Disponible:
<https://www.kai-waehner.de/blog/2020/10/27/streaming-machine-learning-kafka-native-model-server-deployment-rpc-embedded-streams/>
18. Creating Apache Kafka Connector [En Ligne] Disponible:
<https://www.confluent.io/blog/create-dynamic-kafka-connect-source-connectors/>
19. Send data from kafka to elasticsearch [En Ligne] Disponible:
<https://selectfrom.dev/how-to-send-data-from-a-kafka-topic-to-elasticsearch-c623fba84a84>
20. Grubbs, Frank E. "Procedures for detecting outlying observations in samples." *Technometrics* 11, no. 1 (1969): 1-21.
21. Andrews, Jerone TA, Edward J. Morton, and Lewis D. Griffin. "Detecting anomalous data using auto-encoders." *International Journal of Machine Learning and Computing* 6, no. 1 (2016): 21.
22. Kumari, R., M. K. Singh, R. Jha, and N. K. Singh. "Anomaly detection in network traffic using K-mean clustering." In *Recent Advances in Information Technology (RAIT), 2016 3rd International Conference on*, pp. 387-393. IEEE, 2016.
23. Olsson, T., Holst, A.: A probabilistic approach to aggregating anomalies for unsupervised anomaly detection with industrial applications. In: *FLAIRS Conference*. pp. 434–439 (2015)

24. Malhotra, P., Vig, L., Shroff, G., Agarwal, P.: Long short-term memory networks for anomaly detection in time series. In: Proceedings. p. 89. Presses universitaires de Louvain (2015)
25. Sakurada, Mayu, and Takehisa Yairi. "Anomaly detection using autoencoders with nonlinear dimensionality reduction." In Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis, p. 4. ACM, 2014.
26. Docker Documentation [En Ligne] Disponible: <https://docs.docker.com/>
27. Rajdeep Dua, A Reddy Raja, and Dharmesh Kakadia. "Virtualization vs Containerization to Support PaaS". In: 2014 IEEE International Conference on Cloud Engineering. 2014, pp. 610– 614. doi: 10.1109/IC2E.2014.41.
28. Eberhard Wolff. Microservices: Flexible software architecture. Addison- Wesley Professional, 2017. isbn: 0134602412.
29. Apache Kafka Documentation [En Ligne] Disponible:
<https://kafka.apache.org/documentation/>
30. OGC. Publish/Subscribe Interface Standard. [En Ligne] Disponible:
<https://docs.opengeospatial.org/is/13-133r1/13-133r1.html>
31. Spark. Spark Documentation. [En Ligne] Disponible:
<https://spark.apache.org/docs/latest/>
32. J. Damji et al. Learning Spark: Lightning-Fast Data Analytics. O'Reilly, 2020. isbn: 978-1492050049.
33. Elastic. Logstash Reference [En Ligne] Disponible:
<https://www.elastic.co/guide/en/logstash/master/index.html>
34. W.H. Inmon and Daniel L. Data Architecture: A Primer for the Data Scientist. Elsevier, 2015. isbn: 9780128020449
35. Elastic. Grok parser reference. [En Ligne] Disponible:
<https://www.elastic.co/guide/en/logstash/current/plugins-filters-grok.html>
36. Elastic. Elastic Stack and Product Documentation. [En Ligne] Disponible:
<https://www.elastic.co/guide/index.html>

37. P. Atzeni et al. Database Systems: concepts, languages and architectures. McGraw-Hill, 1990. isbn: 0077095006
38. Elastic. Elasticsearch synchronization with a relational database. [En Ligne] Disponible: <https://www.elastic.co/blog/how-to-keep-elasticsearch-synchronized-with-a-relational-database-using-logstash>
39. S. Pranav and Sharath Kumar M. N. Learning Elastic Stack 6.0. OReilly, 2017. isbn: 9781787281868
40. Elastic. Kibana Guide. [En Ligne] Disponible: <https://www.elastic.co/guide/en/kibana/index.html>
41. Elastic. Beats Platform Reference. [En Ligne] Disponible: <https://www.elastic.co/guide/en/beats/libbeat/current/beats-reference.html>
42. P. Atzeni et al. Database Systems: concepts, languages and architectures. McGraw-Hill, 1990. isbn: 0077095006.
43. C. Wohlin, A. von Mayrhauser, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, and A. Wesslén, Experimentation in Software Engineering: An Introduction, ser. International Series in Software Engineering. Springer US, 2012. [En Ligne] Disponible: <https://books.google.se/books?id=3aPwBwAAQBAJ>
44. M. Landauer, M. Wurzenberger, F. Skopik, G. Settanni, and P. Filzmoser, "Dynamic log file analysis: An unsupervised cluster evolution approach for anomaly detection," Computers Security, vol. 79, pp. 94 – 116, 2018. [En Ligne] Disponible: <https://www.sciencedirect.com/science/article/pii/S0167404818306333>
45. Read the docs ElasticAlert [En Ligne] Disponible : <https://elastalert.readthedocs.io/en/latest/ruletypes.html#slack>
46. Kozma, R., M. Kitamura, M. Sakuma, and Y. Yokoyama. "Anomaly detection by neural network models and statistical time series analysis." In Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference on, vol. 5, pp. 3207-3210. IEEE, 1994.