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**

\

Soutenu publiquement le ……. devant le jury composé de :

|  |  |  |  |
| --- | --- | --- | --- |
| Président |  |  |  |
| Examinateurs |  |  |  |
|  |  |  |
| Encadrant |  |  |  |
|  |  |  |

**PRÉFACE**

**DEDICACES**

………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

**REMERCIEMENTS**

………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

**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 connait 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, mais 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.**

**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 boss 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.**

**TABLE DES MATIERES**

**Résumé**

**Abstract**

**Liste des Figures**

**Liste des Tableaux**

**Liste des Sigles et Abréviations**

**Introduction générale**

1. Présentation de la structure d’accueil …………………………………………………
   1. Structure d’accueil ………………………………………………………….
   2. Contexte de l’immersion ……………………………………………………………
2. Problématique …………………………………………………………………
3. Objectifs du projet ……………………………………………………………………
4. Annonce du plan ……………………………………………………………………

**Chapitre 1 : Généralités sur la détection d’anomalies**

1. Mise en contexte ……………………………………………………………
   1. Apprentissage automatique ………………………………………………….
   2. Apprentissage supervisé ………………………………………………
   3. Apprentissage non supervisé ………………………………………………….
   4. Clustering …………………………………………………….
   5. Logs …………………………………………………………………….
   6. Anomalies ……………………………………………………………………
   7. Isolation Forest …………………………………………………….
   8. One Class SVM …………………………………………………….
   9. K-Means …………………………………………………….
   10. Principal Component Analysis (PCA) …………………………………
   11. Accuracy …………………………………………………….
   12. Precision …………………………………………………….
   13. Recall …………………………………………………….
   14. F1Score …………………………………………………….
2. Les types d’anomalies ……………………………………………………………
   1. Les anomalies ponctuelles…………………………………………………
   2. Les anomalies contextuelles …………………………………………………
   3. Les anomalies collectives ………………………………………………………
3. Les challenges
   1. Données non structurées ………………………………………………………
   2. Informations d'exécution redondantes ………………………………………………
   3. Données déséquilibrées …………………………………………………………….

**Chapitre 2 : État de l’art**

1. Évolution des techniques de détection d’anomalies ………………………………………
   1. Détection d'anomalies basée sur la statistique ………………………………………
   2. Détection d'anomalies basée sur la profondeur ……………………………………
   3. Détection d'anomalies basée sur le regroupement (apprentissage non supervisé)
   4. Détection d'anomalies basée sur la distance ……………………………………
   5. Détection d'anomalies basée sur la densité (apprentissage non supervisé) ………
   6. Décomposition spectrale …………………………………………………………
   7. Détection d'anomalies basée sur l'apprentissage automatique supervisé …………
   8. Détection d'anomalies par réseau neuronal classique ………………………
2. Résumé de l’état actuel des connaissances ………………………………………

**Chapitre 3 : Étude conceptuelle de la solution**

1. Analyse des besoins ………………………………………
   1. Les besoins fonctionnels ………………………………………
   2. Les besoins non fonctionnels ………………………………………………………
2. Modélisation UML ………………………………………………………………
3. Diagramme des cas d’utilisation (Uses Cases) ……………………………………
4. Diagramme des classes ……………………………………………………………
5. Architecture du projet ………………………………………………………………
6. Collecte de données ………………………………………………………………
7. Description du jeu de données ………………………………………………………
8. Prétraitement des données ………………………………………………………
9. Algorithme …………………………………………………………………

**Chapitre 4 : Implémentation et déploiement**

1. Environnement de développement et technologies ………………………………………
   1. Docker ………………………………………………………………………
   2. Containers and Virtual Machines ………………………………………
   3. Images …………………………………………………………………
   4. Containers Lifecycle …………………………………………………
   5. Networking ……………………………………………………………
   6. Docker Compose …………………………………………………………
2. Apache Kafka …………………………………………………
   1. Kafka architecture …………………………………………………
   2. Topic structure …………………………………………………
   3. Clustered deployment …………………………………………………
   4. Parallel processing of the topic ……………………………………
3. Apache Zookeeper …………………………………………………
4. Apache Spark ………………………………………………
   1. Challenges in big data processing …………………………………………
   2. Spark cluster …………………………………………
   3. Mleap …………………………………………
5. Logstash …………………………………………………
   1. Data extraction …………………………………………
   2. Data transformation …………………………………………………
   3. Data load …………………………………………………
   4. Parallel processing …………………………………………………
   5. Grok (talk about it) …….
6. Elasticsearch ………………………………………………
   1. Elasticsearch architecture …………………………………………………
   2. Difference from a database ………………………………………………
   3. Clustered deployment …………………………………………………
   4. Search and aggregation ………………………………………………
7. Kibana ………………………………………………
   1. Widget and Dashboard …………………………………………………
   2. Discover ………………………………………
   3. ELK stack monitor and management ………………………………………
8. Beats ………………………………………………
   1. Filebeat and Metricbeat…………………………………………
9. Jupyter Notebook ………………………………………………
10. Summary ……………………………………………
11. Présentation de la solution ……………………………………………………………
12. Analyse et évaluation des résultats obtenus ………………………………………………
13. Déploiement de la solution ………………………………………………………………

**Chapitre 5: Conclusion Générale**

1. Synthèse des travaux
2. Apport du stage
3. Difficultés rencontrées
4. Perspectives

**Références**

## 

## LISTE DES FIGURES

**FIGURES**

## LISTE DES TABLEAUX

**TABLEAU**

## LISTES DES ABBRÉVIATIONS

**ABBREV**

## INTRODUCTION GÉNÉRALE

La détection des anomalies est l'une des tâches les plus importantes pour construire un système fiable et sûr. L'objectif de la détection d'anomalies est de détecter une déviation significative du comportement du système par rapport au comportement normal. La plupart des systèmes nécessitent une détection en temps réel des anomalies dans le but spécifique de réduire les dommages qui peuvent être causés par l'ignorance d'une anomalie ou d'une détection ultérieure. Un exemple d’anomalie qu’on pourrait citer dans une application pourrait être une faille de sécurité, ou bien une panne de service fournit par un tiers. Les implémentations récentes de la détection des anomalies sont principalement basées sur des méthodes d'auto-apprentissage.

L'application des méthodologies d'apprentissage machine ML aux opérations informatiques (AIOps), grâce au nombre croissant de frameworks disponibles, souvent open-source, devient omniprésente dans de nombreux domaines. De plus, l'énorme 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. La diffusion de l'apprentissage automatique correspond à un nombre croissant d'utilisateurs et de systèmes qui accèdent aux prédictions fournies par le modèle d'apprentissage automatique. Cela pose de nouveaux défis, car le modèle ML développé doit être placé dans un environnement dit opérationnel. Les environnements opérationnels ne sont généralement pas sous le contrôle des développeurs, contrairement aux environnements de développement, qui sont entièrement sous contrôle. La principale différence est qu'il est difficile de prévoir l'ampleur des données que le système pourrait être amené à traiter et que ces valeurs peuvent changer à tout moment. En outre, les systèmes matériels et logiciels peuvent tomber en panne à tout moment, mais aucun temps d'arrêt ne peut être toléré dans les environnements opérationnels, car les utilisateurs et les systèmes réels dépendent des services fournis par l'infrastructure ML. Pour résoudre ces deux problèmes, l'infrastructure qui alimente le modèle ML doit être mise à l'échelle et répliquée en conséquence.

Un autre défi des environnements opérationnels est que nous sommes soumis à des exigences temporelles strictes, souvent en temps réel, pour éviter les goulots d'étranglement dans le système supporté par le modèle ML. La plupart du temps, l'application des modèles ML n'est qu'un élément d'une chaîne de traitement plus large, appelée pipeline logiciel. Les pipelines sont utilisés pour traiter les flux de données en continu, l'échelle de l'infrastructure résout en partie ces problèmes, il convient également d'exécuter les modèles d'apprentissage automatique sur des logiciels spécialisés qui permettent une prédiction distribuée à faible latence sur de grandes quantités de données. La surveillance des performances et de l'état des logiciels dans le pipeline devient fondamentale pour répondre aux exigences strictes imposées par la nature de l'environnement opérationnel : les administrateurs doivent être en mesure de détecter les applications défectueuses et de les remplacer rapidement, de reconnaître rapidement les éventuels goulets d'étranglement et de les résoudre dès que possible. La surveillance nécessite une application spécialisée et un stockage dédié pour prendre en charge la collecte des métriques [1], ainsi qu'une interface dédiée pour analyser les données collectées.

Un autre aspect important des environnements opérationnels est la gestion de la sécurité, un problème souvent négligé dans les phases de développement. Les attaques de pirates sur les systèmes opérationnels sont de plus en plus fréquentes, ce qui fait de la sécurité l'une des préoccupations les plus importantes pour les environnements opérationnels où des informations sensibles sont traitées. Nous devrions être en mesure de garantir ce que l'on appelle la triade de la CIA : La confidentialité, c'est-à-dire que toutes les communications entre les étapes du traitement ne doivent pas être lues ou mises à la disposition d'entités non autorisées, l'intégrité, les données doivent être conservées et ne pas être modifiées par des utilisateurs malveillants et la disponibilité, c'est-à-dire la capacité du système à fonctionner en permanence. Il est essentiel de garantir cette propriété dans un environnement opérationnel afin d'éviter les fuites de données potentielles ou la mise hors service du système par un attaquant. La sécurisation de la pile d'applications doit donc être effectuée dès le début du processus de développement en appliquant les meilleures pratiques de sécurité telles que le cryptage et l'authentification entre les parties. Le logiciel choisi doit être bien connues pour avoir été développées avec la sécurité comme première préoccupation, ce que l'on appelle la sécurité dès la conception. Les applications à code source ouvert ont un plus grand avantage que les applications à code source fermé en ce qui concerne la sécurité : étant gratuites, elles sont largement adoptées et testées contre les vulnérabilités et les bogues, et comme leur code est accessible au public, il est généralement examiné par des experts en sécurité.

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 quatre 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 le projet DER actuellement en développement à TeamX et utilisons un modèle ML prédéveloppé qui va effectuer la prédiction d’une panne potentielle dans l’application.

Ainsi, nous avons structuré le document en quatre grandes parties. La première partie de ce mémoire portera sur la présentation la structure d’accueil. Ensuite, la deuxième partie va décrire la réalisation d’une étude de l’état de l’art technique dans ce domaine. Dans la troisième on va la proposer d’une architecture appropriée pour le système. Ensuite la quatrième partie traitera 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.

1. **Présentation de la structure d’accueil**

A remplir later

1. **Problématique**

L’entreprise TeamX est une start-up créée récemment en 2018. Elle évolue dans le domaine de la prestation de service avec des projets tournant autour de domaines assez diversifiés (Fintech, Healthtech, E-commerce, …).

Dans le cadre de la création des projets, les technologies utilisées sont à peu près similaires avec un système de logger permettant de logger les **erreurs** qui surviennent au niveau des apps.

Cette tâche de détection étant effectuée manuellement, il n’existe pas un système de monitoring pour notifier l’équipe en cas de soucis au niveau des applications:

1. un endpoint (api) qui ne marche plus normalement comme l’api Wizzal (cas réel rencontré en entreprise)
2. un service tiers qui n’est plus valide faute de certificat
3. bugs en continu suite au déploiement de nouvelles fonctionnalités

Du coup souvent les développeurs sont obligés d’aller consulter les logs au niveau du serveur pour tracer certaines erreurs relancer par les clients.

1. **Objectifs du projet**

En réponse à cela, l’idée a été émise de créer un système d’alerte basé sur l’IA (Machine Learning) qui fait des prédictions sur les logs pour anticiper les pannes de services sur les applications en production.

1. **Annonce du plan**

Annoncer le plan

## GÉNÉRALITÉ SUR LA DÉTECTION D’ANOMALIES

* 1. **Mise en contexte**

1. **Apprentissage automatique**

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. Elle 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. **Apprentissage supervisé**

L'apprentissage supervisé nécessite la disponibilité de données étiquetées, car les algorithmes relevant de cette catégorie peuvent appliquer des connaissances antérieures à de nouvelles données.

1. **Apprentissage non supervisé**

D'autre part, 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.

1. **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.

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 :

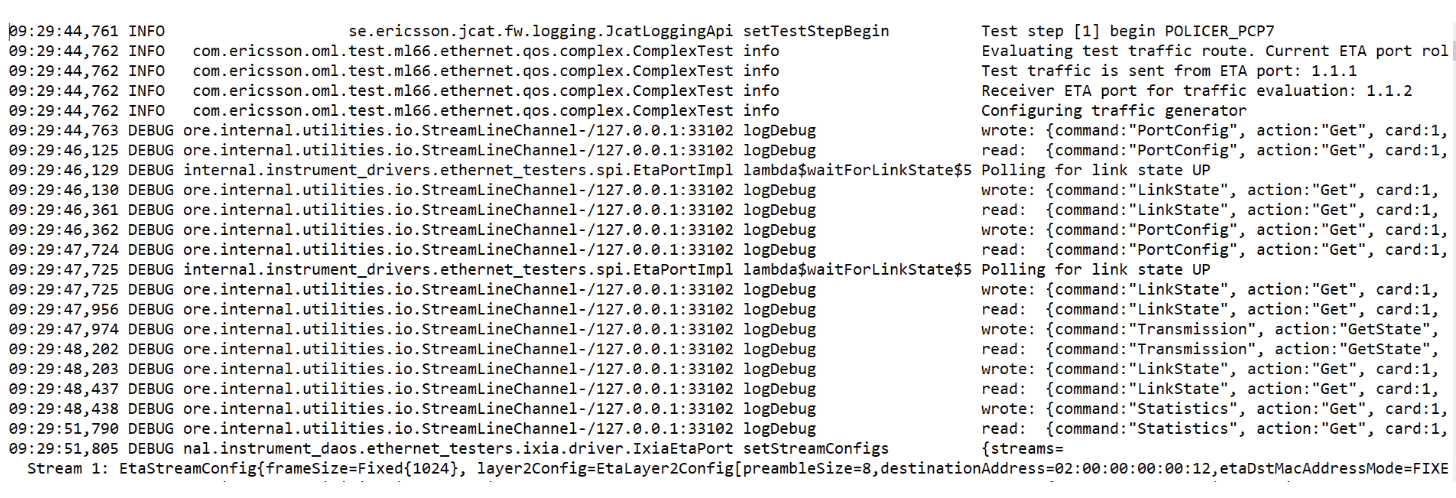
****

Figure 1.1: 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.

1. **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.

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.

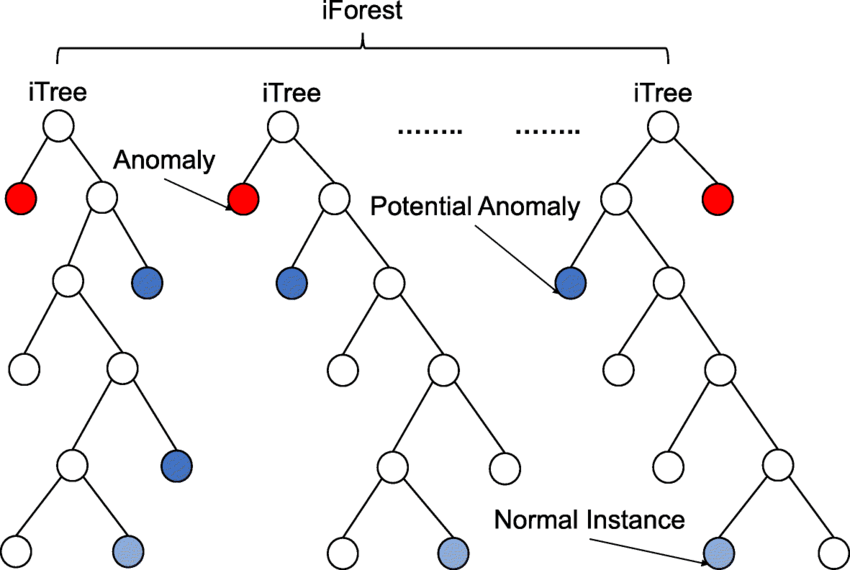


Figure 1.2 : Construction d'iForest pour l'ensemble des données

1. **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.

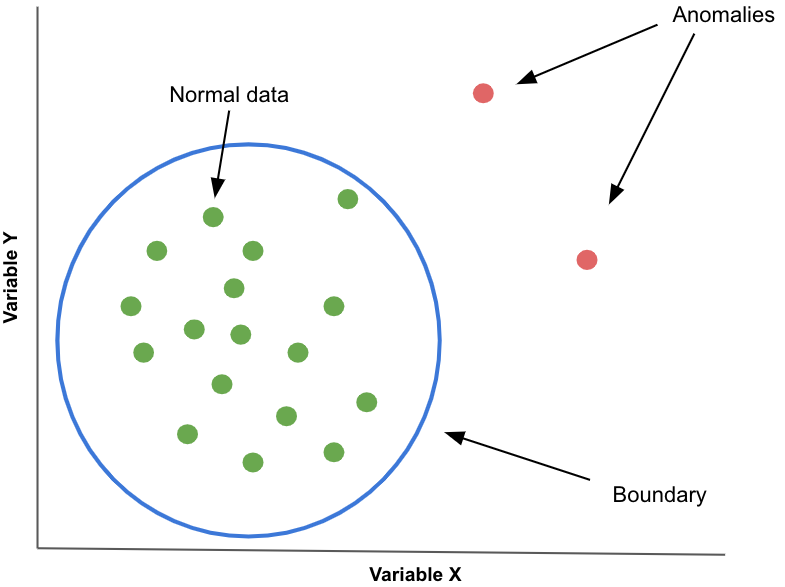


Figure 1.3 : Un exemple visuel d'un simple SVM à une classe

1. **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.

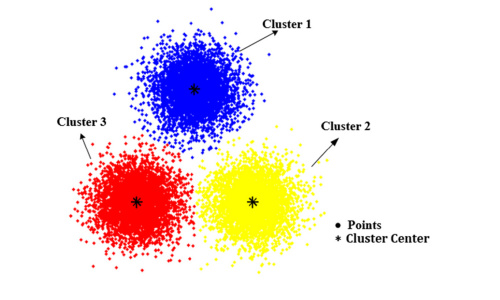


Figure 1.4 : Schéma conceptuel du clustering k-means.

1. **Principal Component Analysis (PCA)**

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 :

1. **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.

Accuracy = T P + T N/(TP + F P + F N + T N)

1. **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.

Precision = TP/TP + FP

1. **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.

Recall = TP/TP + FN

1. **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.

F 1Score = 2 ∗ (Recall ∗ P recision)/(Recall + Precision)

* 1. **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 1.5 montre des exemples d'anomalies ponctuelles.

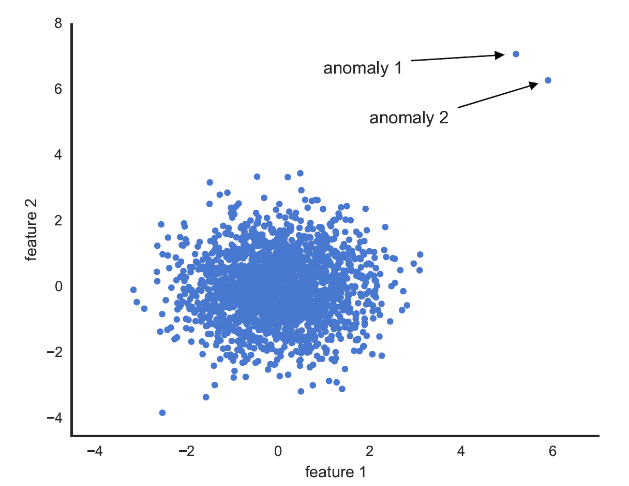


Figure 1.5 : Anomalie ponctuelle

1. **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 6 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.

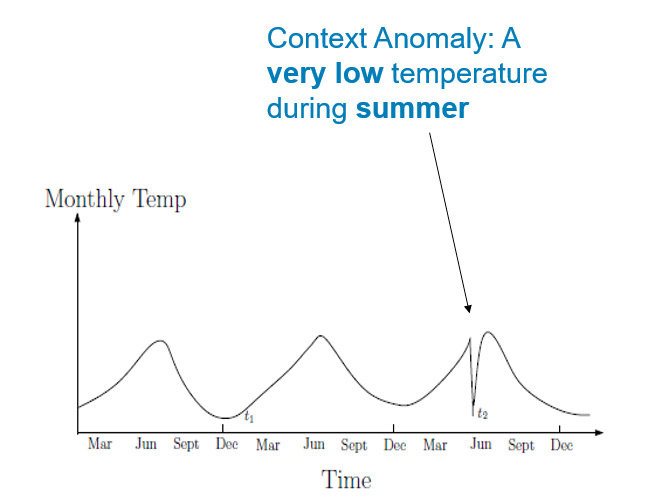


Figure 1.6 : Anomalie contextuelle

1. **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 1.7 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 1.7 : Électrocardiogramme

* 1. **Les challenges**

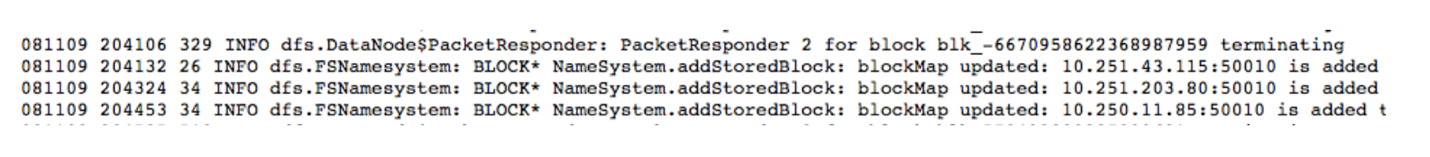
Les anomalies dans les données du journal des applications sont considérées comme des modèles ou des caractéristiques qui ne suivent pas la moyenne ou le comportement normal lors d'un fonctionnement parfait. Comme le décrit Grubbs [20], "une observation aberrante, est une observation qui semble dévier 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**

Le journal d'une application, comme le montre la figure 2, n'est pas structuré et est stocké en texte brut. Ce manque de structure complique l'analyse des données, qui est encore exacerbée par les formats de journalisation qui varient complètement entre les applications.

1. **Informations d’exécution redondantes**

Comme on peut le voir sur la Fig. 1.8, 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 1.8 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.

Figure 1.8 : Journal avec des données redondantes

1. **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 anormale. Par exemple, pour 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 donc le processus de détection des anomalies.

**ÉTAT DE L’ART**

1. **Évolution des techniques de détection d’anomalies**

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 densité, de regroupement, de distance, d'apprentissage automatique et de réseaux neuronaux.

* 1. **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.

* 1. **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 nécessite 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.

* 1. **Détection d'anomalies basée sur le regroupement (apprentissage non supervisé)**

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.

* 1. **Détection d'anomalies basée sur 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.

* 1. **Détection d'anomalies basée sur la densité (apprentissage non supervisé)**

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.

* 1. **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.

* 1. **Détection d’anomalies basée sur l’apprentissage automatique 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 surpasse 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.

* 1. **Détection d'anomalies par réseau neuronal classique**

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.

1. **Résumé de l’état actuel des connaissances**

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].

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.

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.

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.

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.

1. **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
  1. **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.

1. **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, de séquence et d’activités 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 3.1 représente le diagramme de cas d’utilisation du système global.

* 1. **Diagramme des classes**

Les diagrammes des classes

1. **Architecture du projet**
2. **Collecte de données**
3. **Description du jeu de données**
4. **Prétraitement des données**
5. **Algorithme**

## (Discussion expliquer le pourquoi on choisit telle méthode d’apprentissage par rapport à un autre ect regarder dans tes notes)

**IMPLÉMENTATION ET DÉPLOIEMENT**

1. **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. **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 4.1 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 4.1 : Architecture de Docker en comparaison avec la virtualisation classique

* 1. **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. **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 4.2 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.

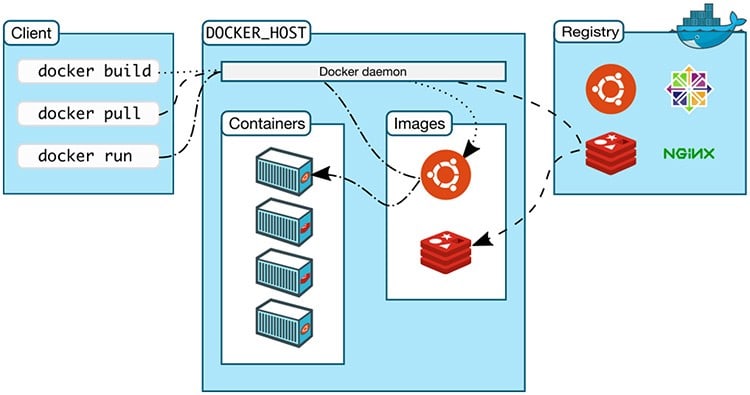


Figure 4.2 : Cycle de vie des conteneurs Docker. Source de l'image : [26]

* 1. **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. **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. 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.

* 1. **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].

* 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. Deuxièmement, 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'équilibreur de charge entre plusieurs travailleurs en exploitant sa fonction de partition des sujets. La figure 2.3 montre l'architecture d'un courtier Kafka.

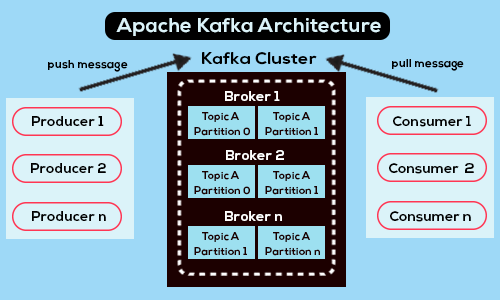


Figure 4.3: Architecture of Apache Kafka

* 1. **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 se soit écoulé. Cela permet à de nombreux abonnés différents 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.

* 1. **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.

* 1. **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.

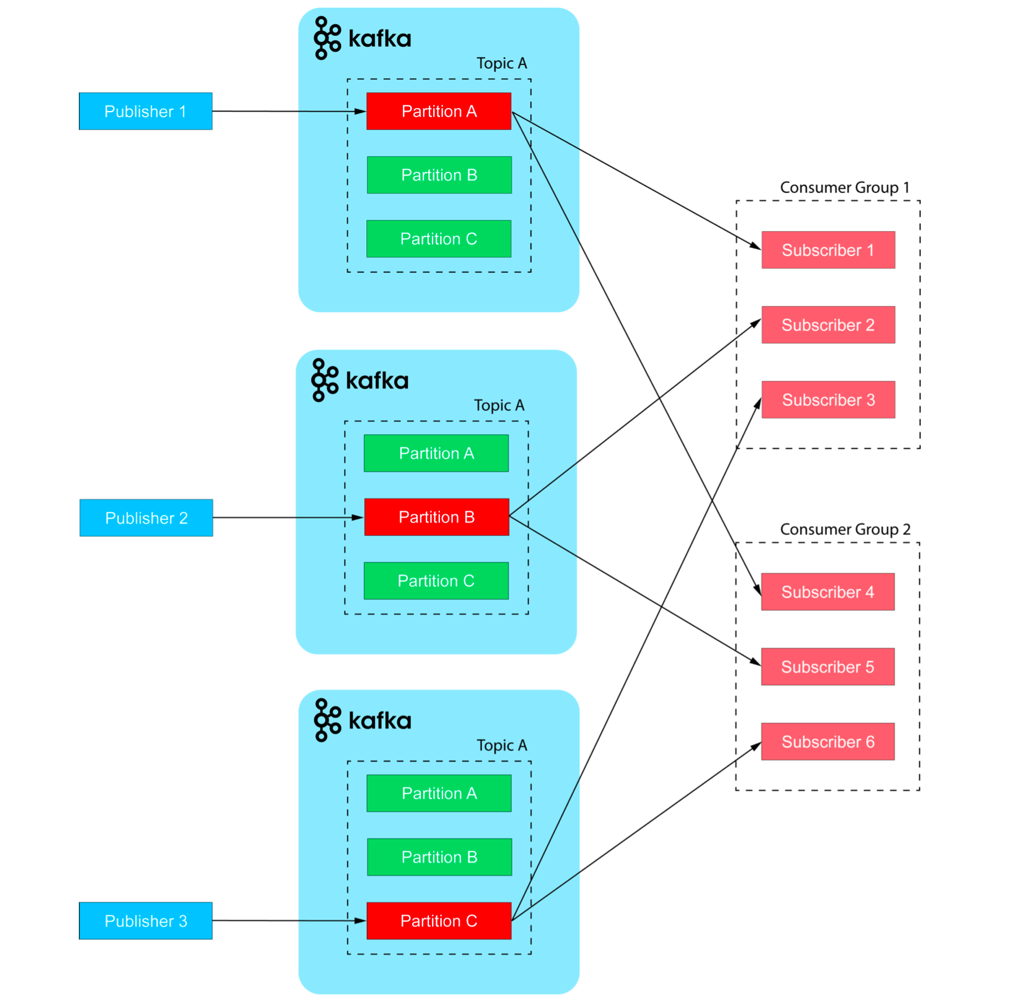


Figure 4.4: Déploiement en cluster d'Apache Kafka. Les partitions surlignées en rouge sont celles du leader.

* 1. **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.

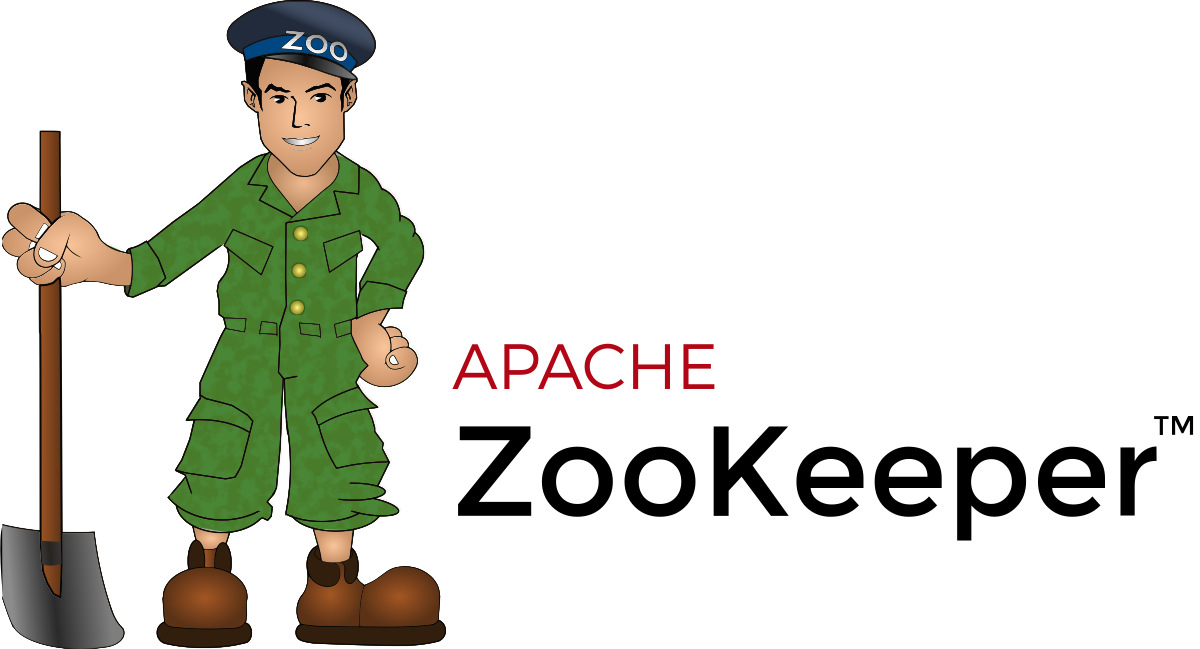


Figure 4.5: Logo Apache ZooKeeper

* 1. **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].

* 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 surgiront 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. Sparks 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.
  1. **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'étincelle, 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 4.5 montre les composants du cluster et les deux phases de l'allocation des tâches aux exécutants.

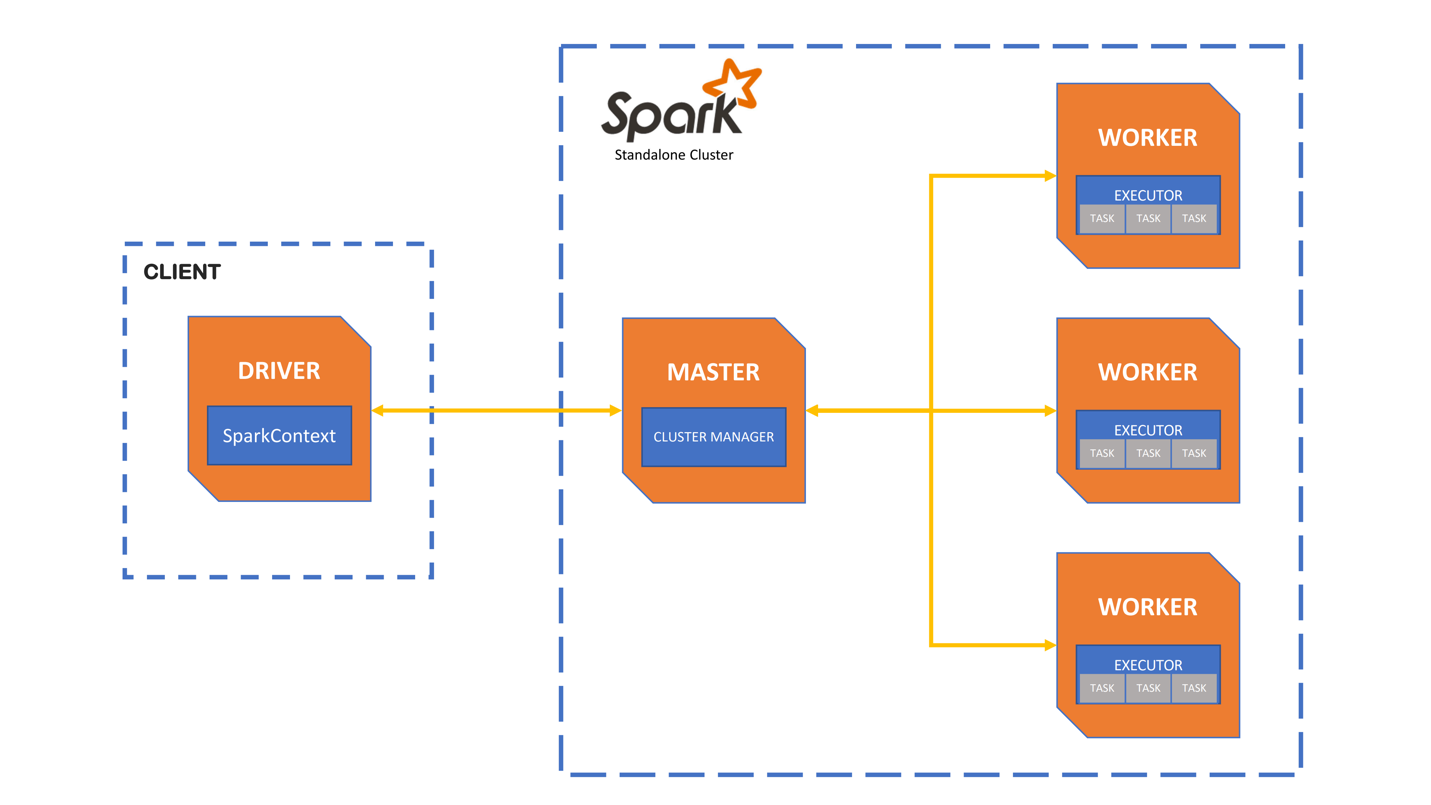


Figure 4.6: Architecture du cluster Spark

* 1. **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 désérialisés 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.

A picture containing graphical user interface

Description automatically generated

Figure 4.7: Spark, Scikit-learn et Tensorflow: One Runtime

* 1. **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 con- figuration du pipeline où les trois phases ETL sont définies. La principale ressource pour cette section est [33].

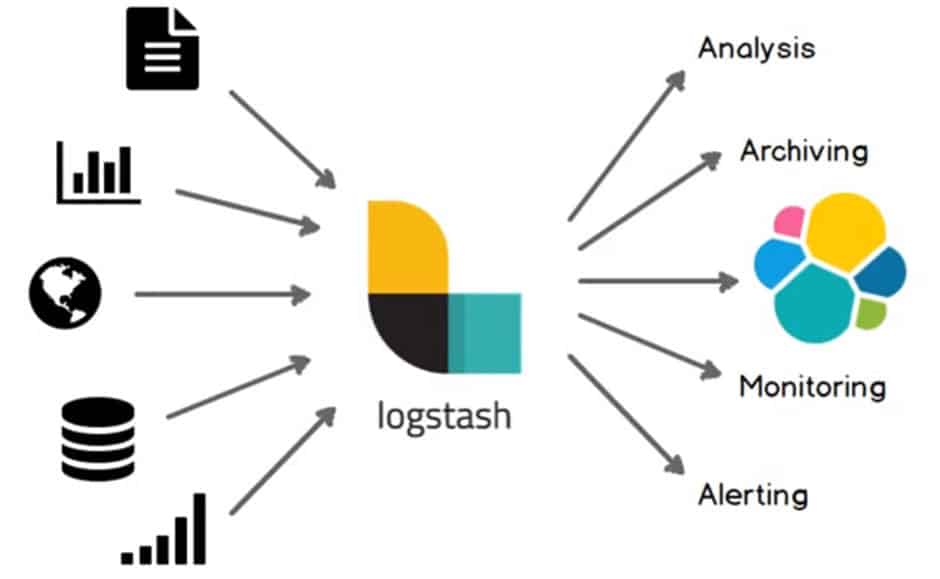


Figure 4.8: Logstash architecture. Image source: [33]

* 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'emblée 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.

* 1. **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.
* 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 structurés et non structurés, 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, appelés filtres. Les exemples vont d'opérations simples comme l'ajout de champs comme 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.

* 1. **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.

* 1. **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 4.

* 1. **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 4.9: Logo Logstash Grok

* 1. **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].

* 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.

* 1. **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.

* 1. **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 :

* Shard primaire : cette tenue 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.
* Shard 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 principaux, 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.

Graphical user interface, diagram, application

Description automatically generated

Figure 4.10: Elasticsearch Cluster

* 1. **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.

* 1. **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].

* 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.

* 1. **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.

* 1. **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 privilèges des utilisateurs, les index Elastcisearch, 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.

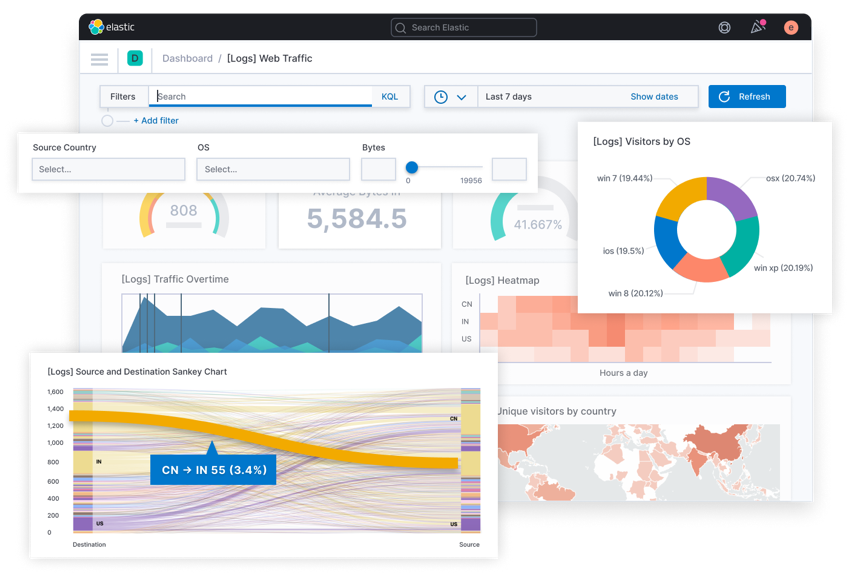


Figure 4.11: Exemple Dashboard Kibana

* 1. **Beats**

Beats est une famille d'expéditeurs de données légers, 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].

* 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.

* 1. **Jupyter Notebook**
  2. **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.

**CONCLUSION GÉNÉRALE**

1. Synthèse des travaux
2. Apport du stage
3. Difficultés rencontrées
4. Perspectives

## 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 MLib [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. OReilly, 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>