

MIAGE M2 SITN

CFA AFIA

Université Paris-Dauphine
SILCA - Groupe Crédit Agricole

Normalisation d'un environnement d'analyse Big Data

Septembre 2017

PAR : Alexis ANZIEU

Tuteur enseignant : Brice MAYAG

Maitre d'apprentissage : Nathalie GUYONNET

Remerciement

Je tiens tout d'abord à exprimer ma reconnaissance envers ma maitre d'apprentissage Madame Nathalie GUYONNET. Je la remercie de m'avoir aidé et conseillé durant ces années, toujours avec le sourire et la bonne humeur.

Je tiens aussi à remercier Monsieur Kevork SARAFYAN ainsi que Monsieur Thomas GILLET pour leur confiance en mes aptitudes face aux projets d'innovation qu'ils m'ont proposés tout au long de l'année.

Remerciements également à l'équipe pédagogique de l'Université Paris-Dauphine, plus particulièrement à mon tuteur Brice MAYAG pour sa disponibilité et pour ses qualités humaines sincères et bienveillantes durant ces semestres.

Je suis en outre ravi d'avoir eu l'opportunité de travailler avec mon collègue-alternant Maxime ANSEAUME et lui souhaite un épanouissement aussi bien dans la vie professionnelle que personnelle.

Mes remerciements les plus chaleureux vont à tous mes collègues du service pour l'ambiance agréable et les nombreux moments conviviaux partagés durant l'année.

Enfin, je souhaite remercier les membres du jury pour leur considération envers ce travail. Travail qui représente le témoignage de ma sincérité et de mon estime envers chacune des personnes que j'ai eu la chance de côtoyer tout au long de mes études et de ma vie professionnelle.

Table des matières

0	Remerciement	2
1	Problématique	5
1.1	Introduction	5
1.2	Contexte	6
1.2.1	SILCA	6
1.2.2	Direction Clients et mission personnelle	6
1.3	Sujet	7
2	Analyse de l'existant	9
2.1	À la recherche des caractéristiques	9
2.1.1	Volume	10
2.1.2	Variété	11
2.1.3	Vélocité	11
2.1.4	Valeur	11
2.1.5	Budget	12
2.1.6	Apprentissage	12
2.1.7	RTO / RPO	12
2.1.8	Sécurité	13
2.1.9	Portabilité	14
2.1.10	Interopérabilité	14
2.2	Banc d'essai	14
2.2.1	Environnement	14
2.2.2	Comparatif	15
2.2.3	Conversion textuelle	20
2.2.4	Pondération	20
2.2.5	Résultat	21
3	Au travail !	23
3.1	Norme-os	23

3.1.1	Mémoire Vive	23
3.1.2	Optimisons l'aléatoire	25
3.2	Norme-outil	26
3.2.1	Bouclons la	26
3.2.2	Protégeons l'interface	28
3.2.3	Darwinons	30
3.2.4	Snapshot, Backup & Restore	32
4	Évaluation	36
4.1	Notation	36
4.2	Outillage	37
5	Projection	40
6	Conclusion	42
A	Organigramme de SILCA	45
B	Capture d'écran de l'outil	47

Problématique

1.1 Introduction

Cette troisième année en alternance au sein de SILCA (production informatique du groupe Crédit Agricole) clôture mes études. Cette continuité m'a permis de découvrir de multiples problématiques et de processus qui ont naturellement émergé du fait de mon intégration dans l'entreprise. En outre, ces dernières années m'ont aussi apporté un épanouissement personnel conséquent ainsi qu'une mise en pratique des nombreuses théories enseignées tout au long de mon cursus scolaire.

Il apparaît que l'appartenance à un groupe d'individus possédant un objectif commun crée beaucoup de relations informelles auxquelles je ne m'étais pas attendu. Certains problèmes se résolvent d'ailleurs de cette façon, sans avoir besoin d'escalader la hiérarchie afin de trouver une issue. Cela propage un dynamisme et une cohésion plus forte dans un service. Ces aspects-là m'ont agréablement surpris pour une entreprise de cette taille, ne pensant voir ces échanges qu'au sein d'une "start-up".

Durant ces derniers mois, les différents objectifs qui m'ont été confiés m'ont confronté à des difficultés que je n'aurai pas pu surmonter sans l'aide de l'équipe avec laquelle je travaille. Bien entendu, cette aide se propage dans les deux sens et j'essaye de fusionner mes jeunes connaissances avec les approches un peu moins récentes déjà mises en place.

La curiosité est sans aucun doute un atout primordial dans l'univers informatique. Les outils évoluent en permanence en fonction des avancées produites par les chercheurs et les développeurs du monde entier. Certains sites web tels "Github" ou "Stackoverflow" permettent justement de mettre en relation toute cette synergie. Cet atout est en outre important quant à la capacité d'adaptation dans une entreprise. Cela correspond à une volonté de creuser le fond de chaque projet et d'y apporter sa contribution. C'est en partant de cette réflexion personnelle que je me suis attelé ces dernières années à tenter de comprendre et d'assimiler le concept d'analyse Big Data.

1.2 Contexte

1.2.1 SILCA

SILCA est une entité du Crédit Agricole, née en 2005 du rapprochement des productions informatiques du Crédit Agricole, du LCL et CACIB. Elle est composée de 617 salariés et gère l'imposante quantité de quatorze pétaoctets de données disposés sur 4000 serveurs. La grosse majorité de ses clients sont "internes", c'est-à-dire qu'ils dépendent directement du groupe Crédit Agricole (CAPS, LCL, Cariparma, etc.). Sa principale mission consiste à fournir des solutions conçues pour la gestion et le maintien de tous les services proposés par la banque (exploitation d'applications, réseau et bureautique, mises à disposition de postes de travail et outils collaboratifs pour les salariés).

Concernant l'organigramme de SILCA, il est décomposé en cinq groupes fonctionnels :

- Direction Support, Gestion et Transformation (risques, contrôle, performance, finance, juridique et secrétariat général).
- Direction Ressources Humaines,
- **Direction Clients, Développement et Services (support d'exploitation des banques de détail, des entités centrales et des entités spécialisées),**
- Direction Des Opérations (édification des standards techniques, plans d'amélioration et de sécurité des infrastructures, garantie de la disponibilité des composants techniques),
- Direction Générale.

1.2.2 Direction Clients et mission personnelle

Mon poste est situé au sein du service d'Exploitation de l'Entité Spécialisées (DES), imbriqué dans la Direction Client (organigramme disponible en annexe B.3). Ce service participe au maintien en condition opérationnelle des applications des entités spécialisées du groupe. Il aide aussi à l'intégration des nouvelles applications dans le cadre de projets ou d'évolutions. C'est justement sur ces dernières que mon domaine de compétence a été requis.

En effet, certains clients préféraient auparavant superviser eux-mêmes leur application en accédant directement et manuellement aux différents journaux de log¹. Cependant, au vu des dizaines de Gigaoctets générés quotidiennement, c'était une besogne non négligeable et très limitée en matière d'analyse. Il m'a alors été confié la tâche de trouver une alternative efficace et rapide à déployer.

C'est ainsi que mes recherches sur le concept d'analyse Big Data débutèrent. Je mis en place en début d'année un premier environnement dédié à ces analyses et le client fut ravi de cette nouvelle expérience utilisateur. Plusieurs applications vinrent alors à subir la même opération. Au fur et à mesure des implémentations, j'améliorais le processus et l'intégrité de l'environnement. Si bien qu'à la mi-année, j'avais en ma possession une liste conséquente de normes qu'il me suffisait de découler à chaque nouvelle installation afin d'en assurer la bonne fiabilité.

1.3 Sujet

Le terme *Big Data* est apparu en octobre 1997 dans l'article *Application-controlled demand paging for out-of-core visualization* co-écrit par Michael Cox et David Ellsworth. [3]

Depuis plusieurs décennies, d'importantes institutions emmagasinent des centaines de téraoctets de données brutes. Ceci en vue de s'en servir par la suite à des fins d'audit, de prospection, de calculs ou de toute autre activité nécessitant une masse d'information exhaustive. Depuis plusieurs années, le nombre d'organisations procédant ainsi croît de façon exponentielle. Cela est dû à deux facteurs principaux : le prix du stockage des données devenu moindre ainsi que la naissance d'algorithmes performants (on parle souvent *d'intelligence artificielle*) permettant de les exploiter. En outre, l'emmagasinement de données est devenu si courant qu'il est maintenant devenu source de discussion pour le grand public. Les capteurs se multiplient autour de nous et de plus en plus de particuliers souhaitent s'essayer à ces nouvelles technologies d'analyse. Il est par exemple possible d'acheter des stations météo² et de faire ses propres statistiques et prévisions. Cependant, poser le premier pas dans ce domaine est rude de par le manque de normalisation des outils à disposition. Pour

1. En informatique, le concept d'historique des événements ou de logging désigne l'enregistrement séquentiel dans un fichier ou une base de données de tous les événements affectant un processus particulier (application, activité d'un réseau informatique...). Le journal (en anglais log file ou plus simplement log), désigne alors le fichier contenant ces enregistrements. Source : Wikipédia

2. <https://www.netatmo.com/fr-FR/product/weather/>

faire court, on ne sait par où commencer.

Ce mémoire a pour finalité d'exposer les caractéristiques actuelles du *Big Data* afin de les normaliser pour permettre à n'importe quel particulier, start-up ou multinationale d'acquérir et de mettre en place un robuste environnement d'analyse de ses données.

Concernant son contenu, les schémas n'ayant pas de source ont été réalisés personnellement avec Microsoft Visio 2010. En outre, un bon nombre de définitions proviennent de "Wikipédia". Concernant sa fiabilité, il s'avère que selon plusieurs études les articles n'entraînant aucun conflit idéologique sont de la même qualité que ceux des encyclopédies traditionnelles payantes.³

Une partie de ce mémoire traite de développement logiciel et de scripting. Le code est intégralement disponible sous licence MIT à l'URL suivante : <https://github.com/AlexisAnzieu/normalizerBigData>

3. Source : <http://bit.ly/2eqIjHt>

Analyse de l'existant

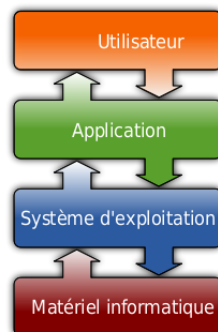
2.1 À la recherche des caractéristiques

Dans le vaste monde du logiciel, de nombreuses caractéristiques s'entremêlent. Du fait de l'open source¹, certains produits sont déclinés à l'infini. Il devient alors fastidieux de faire un choix parmi cette pluralité. En conséquence, il est primordial de commencer par rechercher les attributs les plus décisifs pour notre produit, en fonction de notre objectif final.

Dans notre cas, il s'agira donc dans un premier temps de déterminer les caractéristiques nécessaires à la mise en place d'un environnement d'analyse *Big Data*. Nous commencerons par celles de la plus haute couche, l'application *Big Data* en elle-même pour finir sur les plus basses, celles du système d'exploitation et du matériel.

FIGURE 2.1 – Couches d'un environnement informatique.

(source : *wikipedia.org*)



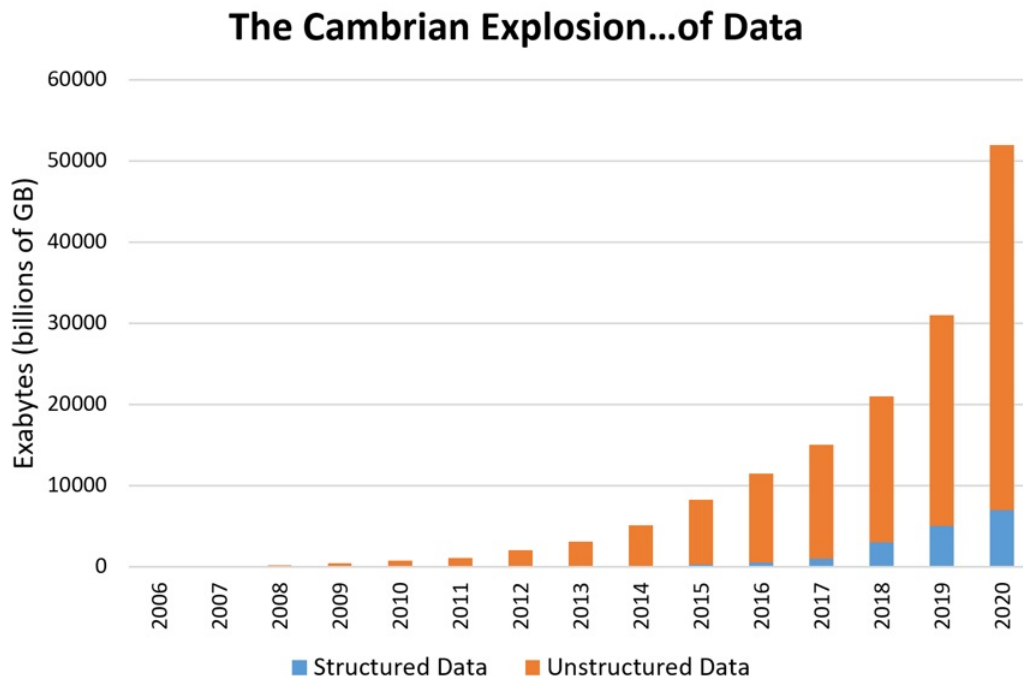
Nous reviendrons sur la couche utilisateur par la suite, car elle est tout aussi importante que les précédentes, mais débutons par les trois attributs principaux du *Big Data*, les "3 V".[9]

1. Un logiciel Open Source est un programme informatique dont le code source est distribué sous une licence permettant à quiconque de lire, modifier ou redistribuer ce logiciel. (source : *1min30.com*)

2.1.1 Volume

Cette première caractéristique n'est pas des moindres car elle est l'essence même du terme *Big Data*. 90% des données produites depuis le début de l'existence humaine l'ont été durant ces deux seules dernières années [4] et cette croissance ne va pas s'estomper puisqu'en 10 ans cette quantité va décupler [11]. Au vu de ces chiffres vertigineux, on peut aisément entériner le fait que notre environnement d'exploitation doit être à même de gérer ces capacités de données qui peuvent varier de quelques téraoctets² à plusieurs centaines de pétaoctets. Notons que si les besoins en volumétrie ne sont pas aussi importants, alors il n'est peut-être pas nécessaire de se pencher sur une solution *Big Data*.

FIGURE 2.2 – Évolution de la quantité de données.
(source : *eetimes.com*)



2. Octet : Unité de mémoire de l'informatique correspondant à une suite de huit chiffres binaires.

2.1.2 Variété

En partant de la définition pure de la donnée³ décrite comme étant un élément intangible dépourvue de raisonnement, il est aisé de constater qu'il n'existe pas de schéma capable de la généraliser. Une vidéo, une photo, une mesure provenant d'un capteur ou un texte sont chacune des données de type différent. Dans le *Big Data*, il n'y a pas de distinctions entre ces données. Tout est absorbé, stocké puis traité en aval. Cette pratique a pour conséquence l'engrangement d'une gigantesque variété de données, provenant de sources et de méthodes de traitement potentiellement différentes.

2.1.3 Vitesse

Dans le paragraphe précédent, nous avons formulé le fait que certaines données pouvaient être des mesures de capteurs. Ces mesures sont susceptibles d'être ingérées en temps réel par des algorithmes afin d'évoluer en indicateurs. Ces derniers permettent à des opérateurs ou à d'autres algorithmes de modifier leurs comportements. Il est donc nécessaire que tout ce processus soit effectué le plus rapidement possible afin de ne pas avoir de comportements déphasés par rapport aux données. En outre, même si ces données ne sont pas toutes des mesures, elles doivent demeurer accessibles le plus vite possible afin de satisfaire les besoins de l'utilisateur, notamment lors d'une recherche ou d'une création de graphiques en direct.

2.1.4 Valeur

La large quantité et variété de données au sein d'un environnement *Big Data* n'est pas synonyme de productivité. Certaines de ces données n'ont pas de valeur immédiate et ont été stockées en attente d'une utilité. Si cela n'a pas de conséquences à court terme sur un faible volume, ce n'est pas le cas sur du long terme où certaines études montrent que dans de nombreux environnements, 5% des données suffisent à satisfaire 95% des besoins d'analyse. [1]

Afin de limiter au mieux ce ratio, il est indispensable de fournir un travail en amont en séparant les informations cruciales de celles superflues. Ces dernières peuvent par

3. Une donnée est une description élémentaire d'une réalité. C'est par exemple une observation ou une mesure. La donnée est dépourvue de tout raisonnement, supposition, constatation, probabilité. Étant indiscutable ou indiscutée, elle sert de base à une recherche, à un examen quelconque. (*source : wikipedia.org*)

la suite être conservées, mais leur identification permettra une purge prioritaire et instantanée si un incident ou un manque de budget survient.

2.1.5 Budget

L'analyse de données *Big Data* nécessite non seulement un budget matériel proportionnel à la masse de données à exploiter, mais aussi un budget logiciel et humain. Toute organisation n'a pas les moyens de s'offrir les services d'une prestation externe ou d'acheter une licence d'exploitation onéreuse.

Concernant l'aspect logiciel, l'open source est ici un atout non négligeable, car de nombreux outils sont disponibles et révisables gratuitement sans coût légal supplémentaire. Néanmoins, il faut faire attention concernant la viabilité du produit, en préférant ceux maintenus fréquemment. Pour ce qui est de l'aspect humain, la courbe d'apprentissage se doit d'être aisée afin de s'y adapter rapidement.

2.1.6 Apprentissage

Se former sur un logiciel d'analyse *Big Data* est une priorité car cela permet de connaître immédiatement ses limites et les améliorations à apporter. La documentation doit être suffisamment claire, concise et au moins disponible en anglais à défaut de sa langue natale afin d'être compréhensible par le maximum de collaborateurs.

La communauté autour du produit se doit d'être active et conséquente. Ceci afin de rapidement trouver une solution à plusieurs et ne pas rester immobilisé sur une problématique. La plate-forme Github, outil d'hébergement de logiciels open-source, est un bon indicateur de ce dynamisme. Outre le fait d'avoir accès en temps réel aux différentes mises à jour et discussion concernant les fonctionnalités futures du produit, nous avons aussi accès à sa popularité grâce aux "stars", sorte de tampon que les développeurs peuvent apposer lorsque le projet leur semble intéressant.

2.1.7 RTO / RPO

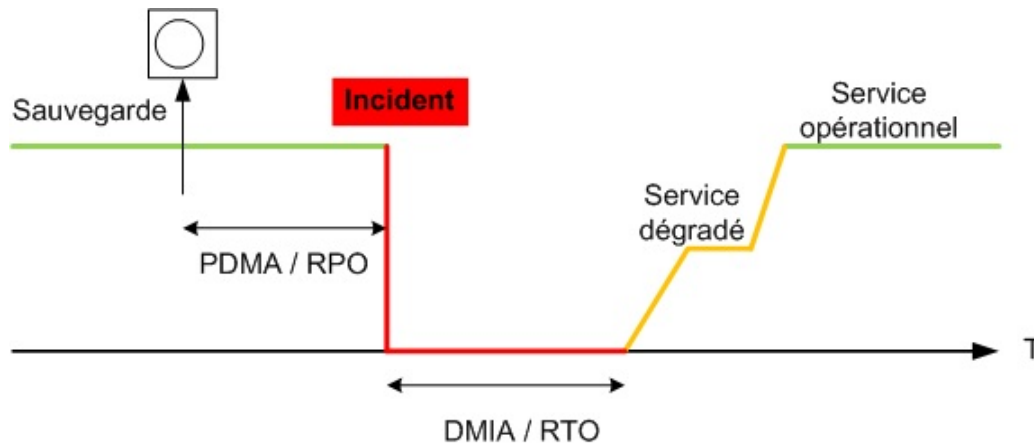
Le *Recovery Time Objective* ou Durée Maximale d'Interruption Admissible (DMIA) est cruciale sur un environnement de production, autant en matière d'image que de coût financier. Il s'agit du temps maximal acceptable durant lequel une ressource peut ne pas être fonctionnelle suite à une interruption majeure de service. Elle est définie en avance par le client ainsi que par le paramétrage possible de l'application.

Il peut tout aussi bien s'agir d'une simple sauvegarde de données que d'un redémarrage complet d'un parc de serveurs.

Concernant la sauvegarde de donnée, un terme a d'ailleurs été spécifiquement créé : le *Recovery Point Objective*. C'est un indicateur qui quantifie les données qu'un système d'information peut-être amené à perdre à la suite d'un incident. Plus la sauvegarde remonte dans le temps, plus le RPO sera important et donc plus la perte engendrée sera grande.

Il faudra donc s'assurer au préalable que l'environnement d'analyse *Big Data* respecte les normes SI propres à l'entreprise, en modifiant certains fichiers de configuration que nous verrons par la suite.

FIGURE 2.3 – Plan de Reprise d'Activité (PRA).
(source : wikipedia.org)



2.1.8 Sécurité

Présente aussi bien sur plan physique que sur le plan applicatif, la sécurité est une caractéristique primordiale. Nous ne nous pencherons ici cependant que sur l'applicatif.

Certaines données *Big Data* pouvant être confidentielles, nous devons cloisonner leur visualisation à des profils restreints et autorisés. Néanmoins, l'analyse étant au coeur de notre problématique, nous souhaitons avoir accès aux données en temps réel à partir de poste de travail distant. En outre, le paramétrage par défaut de certaines applications du système d'exploitation peut aussi faciliter les intentions de personnes malveillantes et est donc à proscrire.

2.1.9 Portabilité

Lors d’une migration ou d’une entrée dans un périmètre nécessitant de la redondance, l’environnement applicatif se doit d’être facile dupliquer et à migrer, on parle alors de portabilité. Cet attribut n’est cependant réalisable qu’avec un environnement stable et générique, suffisamment testé et surveillé pour pouvoir être copié à l’infini sans re-paramétrage fastidieux.

Cette caractéristique est fortement corrélée avec le RTO, lors d’un Plan de Reprise d’Activité, après une panne ou une inactivité d’un ou plusieurs serveurs.

2.1.10 Interopérabilité

Comme vu auparavant, les données *Big Data* peuvent être de types et de structures très différents. Cela est dû au fait que leur provenance est hétérogène. Elles sont générées par des applications qui peuvent ne rien avoir en commun, pas même le langage. Il est donc nécessaire d’avoir un processus capable de normaliser cette ingestion en provenance d’environnements diamétralement opposés. Qu’importe d’où proviennent les données, l’analyse doit rester identique.

2.2 Banc d’essai

2.2.1 Environnement

Ces différentes caractéristiques vont nous permettre de définir le meilleur outil d’analyse à notre disposition. Pour rappel, le but n’est pas de développer nous-même un outil d’analyse mais de définir un socle sur lequel nous pourrions en implémenter de façon générique, de telle sorte à établir une norme non seulement sur l’outil en lui-même mais aussi sur les aspects du système d’exploitation. Ces deux facettes seront néanmoins décorréées au maximum afin de pouvoir basculer d’un outil à l’autre sans devoir arranger les manipulations auparavant effectuées. Un axiome de départ va toutefois être imposé. Afin d’être le plus général possible et en concordance avec l’esprit open-source, nous nous baserons sur une solution auto-hébergée dans un environnement Linux tout au long de ce mémoire. Cela ne concerne évidemment que l’environnement de l’outil et non l’interopérabilité des données qui pourront être quant à elles agrégées à partir du plus grand nombre de systèmes d’exploitation possible.

Outre les contraintes éthiques et financières, nous allons aussi nous fixer des contraintes techniques initiales, toujours dans l'idée de créer une normalisation adaptée au grand public. Nous nous baserons sur les composants d'un ordinateur moyen actuellement disponible sur le marché [14] :

- 4 Go de mémoire vive ;
- 250 Go de disque dur ;
- 2 CPU.

Les différents tests seront réalisés sur un environnement identique réinitialisé à chaque fois grâce au principe de la virtualisation⁴. Ceci dans le but d'obtenir le résultat le plus impartial possible.

2.2.2 Comparatif

Venons en maintenant aux différents outils à notre disposition. De nombreux acteurs se sont rapidement introduits sur le marché[2], cependant ils sont pour la plupart payants ou accessibles seulement au travers d'Internet (et non auto-hébergé comme nous le souhaitons). Cela restreint donc drastiquement notre champ de recherche. En effet, seules trois applications respectent ces contraintes. Elles comportent toutes un ETL⁵, nécessaire au traitement *Big Data*. Il s'agit de Splunk, ELK (Elasticsearch pour la base de données NoSQL, Logstash pour l'ETL et Kibana pour l'interface graphique) et Graylog (un dérivé de ELK qui possède sa propre interface graphique mais avec une base de données MongoDB supplémentaire).

Afin de discerner lequel de ces trois produits est le plus adapté aux attentes actuelles du *Big Data*, nous allons les étudier en fonction de chacune des caractéristiques des paragraphes précédents. Nous résumerons ensuite nos analyses sous la forme d'un tableau. Cela nous permettra de visualiser directement l'outil sortant de cette recherche dont nous nous servirons pour établir notre norme-outil.

4. illusion d'un appareil informatique créée par un logiciel d'émulation. Le logiciel d'émulation simule la présence de ressources matérielles et logicielles telles que la mémoire, le processeur, le disque dur, voire le système d'exploitation et les pilotes, permettant d'exécuter des programmes dans les mêmes conditions que celles de la machine simulée. (*source : wikipedia.org*)

5. Il s'agit d'une technologie informatique intergicielle (Extract - Transform - Load) permettant d'effectuer des synchronisations massives d'information d'une source de données

Splunk

Volume/Variété/Vélocité : Étant la base d'un environnement *Big Data*, tous ces critères sont respectés. Ils ne sont donc pas un point où la comparaison va être significative.

Valeur : L'ETL en place filtre lui-même les données brutes en entrée selon les algorithmes de l'entreprise.

Budget : La licence est payante et liée au volume par année, si une sortie de log applicative est en erreur ou que l'environnement subit une attaque DOS, le budget explose. En outre, il est recommandé de posséder 12GO de RAM, ce qui n'est pas à la portée de toutes les bourses.

Apprentissage : L'installation est réalisée en 5 minutes mais le langage de recherche SPL est propriétaire et nécessite beaucoup de pratiques avant d'être assimilé.

RTO/RPO : La relance s'effectue avec la seule ligne de commande : "splunk start". La durée nécessaire à la relance n'excède donc pas les quelques minutes (pour peu que l'application soit correctement supervisée)

Sécurité : Le protocole HTTPS⁶ est implémentable et des sessions utilisateurs sont configurables.

Portabilité : Tout est compilé dans l'application, aucune librairie externe n'est nécessaire. Il est donc très aisé de transposer l'outil d'un système d'exploitation à un autre.

Interopérabilité : Beaucoup de connecteurs sont disponibles mais l'organisation en boîte noire⁷ mise en place sur ce traitement de données en entrées n'est pas forcément une bonne chose lors de débogages.

6. HTTPS permet aux sites web de prouver leur identité grâce à un certificat d'authentification émis par une autorité tierce, réputée fiable. Il garantit théoriquement la confidentialité et l'intégrité des données envoyées par l'utilisateur

7. Représentation théorique d'un système sans considérer son fonctionnement interne

ELK

Volume/Variété/Vélocité : Étant la base d'un environnement *Big Data*, tous ces critères sont respectés. Ils ne sont donc pas un point où la comparaison va être significative.

Valeur : L'ETL opensource fourni avec l'outil permet de filtrer aisément les données en entrée.

Budget : L'outil est gratuit et en partie opensource mais certains modules avancés (machine learning, sessions utilisateurs) sont payants.

Apprentissage : La configuration de l'ETL se fait manuellement sans interface utilisateur, ce qui rend difficile sa configuration par un utilisateur non technique. Le langage de recherche utilisé est Lucène, de la fondation Apache, mis à disposition en opensource.

RTO/RPO : Chacun des services composant l'outil (trois au total) doit être relancé séparément dans un ordre spécifique.

Sécurité : Le protocole HTTPS est implémentable et des sessions utilisateurs sont configurables monnayant le module correspondant.

Portabilité : Les outils étant développés en JAVA, l'intégration est facilitée par la popularité de ce langage (nécessite la version 8)

Interopérabilité : L'existence de plugins opensource pour l'ETL rendent les entrées/-sorties des différentes sources très adaptables. Une incompatibilité subsiste cependant avec le système d'exploitation AIX développé par IBM et le langage Go parfois nécessaire.

Graylog

Volume/Variété/Vélocité : Étant la base d'un environnement *Big Data*, tous ces critères sont respectés. Ils ne sont donc pas un point où la comparaison va être si-

gnificative. Valeur : L'ETL fourni avec l'outil permet de filtrer aisément les données en entrée.

Budget : L'outil est gratuit et opensource.

Apprentissage : La configuration de l'ETL se fait manuellement sans interface utilisateur, ce qui rend difficile sa configuration par un utilisateur non technique. En outre la communauté est moindre, ce qui rend difficile son implémentation en cas de blocage.

RTO/RPO : Chacun des services composant l'outil (quatre au total) doit être relancé séparément dans un ordre spécifique.

Sécurité : Le protocole https est implémentable et des sessions utilisateurs sont configurables.

Portabilité : Il est nécessaire d'installer une base de données supplémentaire (mongoDB), ce qui alourdit le processus de portabilité.

Interopérabilité : L'existence de plugins pour l'ETL rend les entrées/sorties très adaptables.

FIGURE 2.4 – Évolution des recherches des trois outils d'analyse corrélée à l'intérêt porté au Big Data.

(source : <https://trends.google.fr/trends/>)

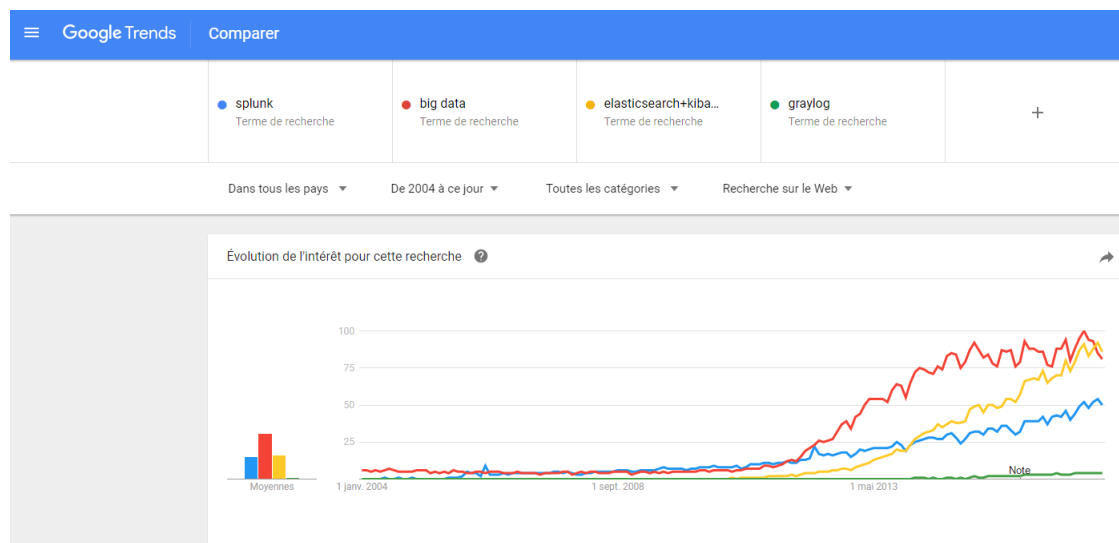


TABLE 2.1 – Tableau comparatif des solutions ETL

Outil	Les "3V"	Valeur	Budget	Sécurité
Splunk	Variations négligeables	Traitements internes	Licence très onéreuse	Intégrée
Graylog	Variations négligeables	ETL paramétrable	opensource	Peu intégrée
ELK	Variations négligeables	ETL paramétrable	opensource avec modules payants	Pas intégrée sans licence

Outil	Apprentissage	RTO/RPO	Portabilité	Interopérabilité
Splunk	long, licence fournie avec formations	Quelques minutes	Intégration en une étape	Connecteurs payants et gratuits
Graylog	Communauté peu nombreuse outil récent	Peu adapté	Environnement lourd à migrer	200 plugins ETL opensource
ELK	Mises à jour nombreuses	Redémarrage lourd	Nécessite JAVA 8	200 plugins ETL opensource

Malgré ce tableau et le graphique des tendances actuelles, il reste peu aisé de déterminer ici l'outil le plus adapté à nos besoins. En effet, certaines caractéristiques ont moins d'importances que d'autres et un jugement humain ne peut être rendu sur 8 caractéristiques et 24 résultats textuels différents.

Dans un premier temps, nous allons donc convertir les résultats écrits de notre tableau récapitulatif en matrice numérique. Ensuite, nous créerons une matrice de pondération pour chaque attribut. Enfin, nous multiplierons ces deux matrices ainsi formées pour d'obtenir un dénouement cohérent :

Choix final = résultats évalués * pondération des attributs

2.2.3 Conversion textuelle

Reprenons notre tableau textuel 2.1 de la page 19 et établissons-y les valeurs numériques qui nous semblent les plus adaptées. Le nombre d'outils étant de 3, nous allons attribuer la note de 1, 2 ou 3 à chaque case en fonction de son d'adaptation envers la caractéristique, 3 étant la meilleure note. Si ex aequo il y a entre 2 cases, la note sera de 2.

TABLE 2.2 – Conversion numéraire du comparatif des solutions ETL

Outil	"3V"	Valeur	Budget	Sécu.	Apprent.	RTO	Portab.	Interop.
Splunk	2	1	1	3	2	3	3	1
Graylog	2	2	3	2	1	1	1	2
ELK	2	2	2	1	3	2	2	2

Ce qui nous mène à un résultat de 15 pour Splunk, 15 pour Graylog et 16 pour ELK. Ce résultat n'est guère plus probant, ce qui nous dirige donc vers l'étape de pondération des caractéristiques.

2.2.4 Pondération

Débutons nos estimations avec le multiplicateur 1 (l'élément neutre de la multiplication). Ce coefficient pourra varier de 1 à 4 en fonction des différentes priorités, 4 étant le coefficient le plus important.

Les "3V" étant au même niveau pour tous les outils, il n'est pas nécessaire de modifier la valeur de notre coefficient ici. En revanche, la valeur a une part importante dans

un contexte *Big Data* car c'est elle qui permet de déterminer la pertinence de nos données et donc leur richesse. Le coefficient sera donc de 4. Il en va de même pour le budget qui est rapidement crucial pour tous, un 4 est donc là aussi de mise. En matière de sécurité, c'est le point noir pour toute grosse société, mais pas nécessairement apocalyptique lorsqu'il s'agit d'un petit outil d'analyse. Ce dernier sera donc placé sur 3. L'apprentissage est lié au budget et dépend surtout des collaborateurs impliqués dans le projet. Cet attribut sera donc lui aussi pondéré par 3. La reprise d'activité est intéressante mais il est possible de développer un outil chargé de relancer les outils automatiquement, il en va de même pour les plugins d'interopérabilité. Nous allons donc leur attribuer un 2. Enfin, la portabilité joue un rôle majeur car c'est cela qui va être le noyau de notre outil d'analyse. Nous terminerons donc notre matrice de pondération avec un 4.

TABLE 2.3 – Pondération des attributs

Attribut	"3V"	Valeur	Budget	Sécu.	Apprent.	RTO	Portab.	Interop.
Valeur	1	4	4	3	3	2	2	4

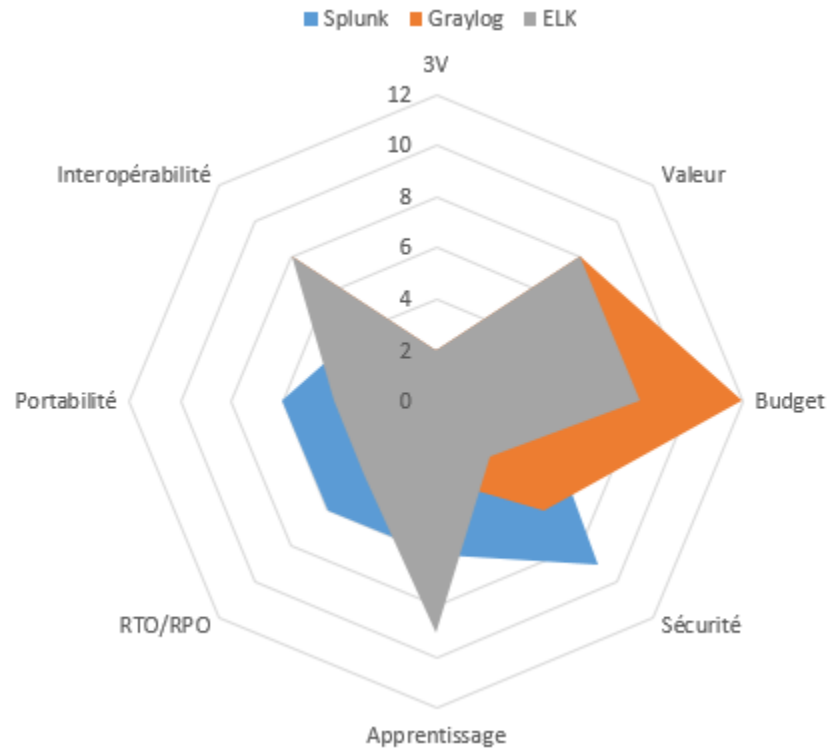
2.2.5 Résultat

Il convient donc maintenant de multiplier les deux matrices créées précédemment et d'en exploiter le résultat qui conduira au choix de l'outil le plus adapté à la création de notre environnement d'analyse Big Data.

$$\begin{pmatrix} 2 & 1 & 1 & 3 & 2 & 3 & 3 & 1 \\ 2 & 2 & 3 & 2 & 1 & 1 & 1 & 2 \\ 2 & 2 & 2 & 1 & 3 & 2 & 2 & 2 \end{pmatrix} * \begin{pmatrix} 1 \\ 4 \\ 4 \\ 3 \\ 3 \\ 2 \\ 2 \\ 4 \end{pmatrix} = \begin{pmatrix} 41 - Splunk \\ 43 - Graylog \\ 46 - ELK \end{pmatrix}$$

Grâce à nos données numériques et à notre graphe, nous remarquons que ELK est davantage régulier dans les domaines liés au large public (budget, apprentissage et interopérabilité). Splunk est plutôt dédié aux grosses organisations qui peuvent se permettre d'investir plusieurs milliers d'euros dans un environnement d'analyse. Enfin, Graylog permettrait de combler certaines déficiences de ELK mais c'est un

FIGURE 2.5 – Radar de caractéristiques des outils



outil encore jeune qui nécessite encore quelques améliorations en matière de fiabilité.

Pour conclure, nous allons normaliser un environnement d'analyse Big Data en nous basant sur l'outil ELK, tout en essayant de diminuer la quantité de ses défauts trouvés précédemment afin de l'adapter au plus grand nombre d'usages possible.

Au travail !

L'outil trouvé, nous pouvons dorénavant explorer plus en profondeur ses besoins d'architecture et de puissance matérielle. Nous appellerons cette étape la *Norme-OS* car elle s'appuiera exclusivement sur le socle de l'outil et non sur l'outil en lui-même. Par la suite, nous implémenterons ELK dessus en essayant de l'optimiser et de mettre en application les meilleures pratiques. Cette partie sera nommée *Norme-outil* et permettra de répondre pleinement à notre problématique en nous appuyant sur les défauts soulevés lors des différents tests précédents.

3.1 Norme-os

3.1.1 Mémoire Vive

La documentation officielle d'ELK préconise entre 16Go et 32Go de RAM [6]. Cependant cela correspond à un point de vue "entreprise" et il est théoriquement possible de faire fonctionner l'outil avec moins de puissance. Afin de déterminer au mieux la capacité requise pour un usage tout public, nous avons donc mis en place un "benchmark" (ou "test de performance") qui consiste en l'injection de plusieurs millions de données textuelles dans elasticsearch.

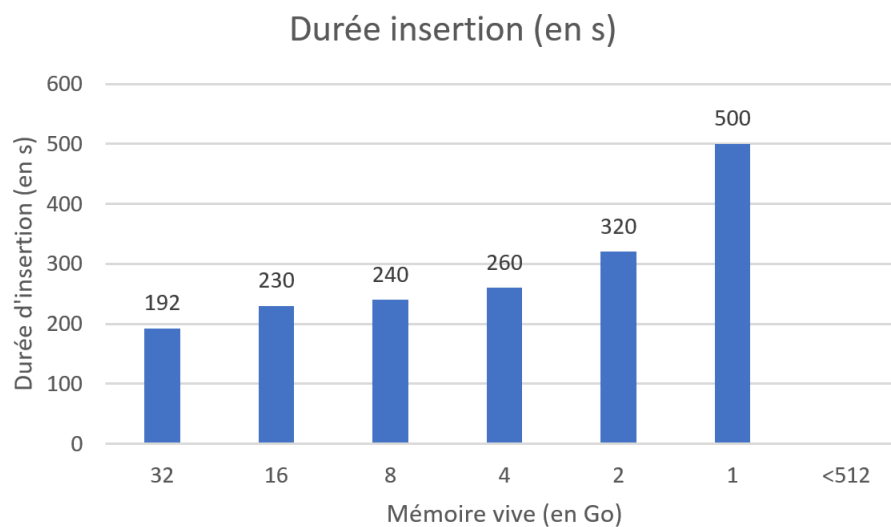
Nous savons que cet outil est développé en Java, la valeur de départ de notre test va donc se baser sur la configuration minimale requise pour exécuter le langage Java. D'après les spécifications [12], cette valeur est de 128Mo. Concernant la valeur finale du test mémoire et n'ayant à notre disposition qu'un dispositif doté de 16Go de mémoire, les résultats d'un système 32 Go ont été calqués sur les résultats d'Elastic [5]. Souhaitant être le plus précis possible, notre test de performance et nos données sont donc une extension de ce test.

La démarche d'Elastic est détaillée sur la plateforme de développement collaborative Github [7], nous avons donc reproduit les différentes étapes qui consistent princi-

pablement au téléchargement[10] de 11 540 072 points de géolocalisation à travers le monde puis en leur conversion au format JSON afin de les insérer dans la base de données.

L'axe des abscisses représente la durée nécessaire à l'injection de tous les points dans la base tandis que l'axe des ordonnées correspond à la mémoire correspondante allouée. Corrélée au système binaire[16], cette dernière est donc normée par toutes les puissances de 2, de 32Go à 512 Mo.

FIGURE 3.1 – Évolution de la durée d'insertion de données en fonction de la mémoire vive attribuée



Nous remarquons qu'elasticsearch ne fonctionne pas avec une allocation mémoire inférieure à 512 Mo. Nous avons donc regroupé ces valeurs dans la catégorie <512Mo.

Grâce à ce test de performance, nous sommes maintenant capables de déterminer la configuration minimale à adopter afin d'avoir un environnement d'analyse adéquat. En effet, il apparaît que la diminution de 2 à 1Go de mémoire vive fait drastiquement chuter les performances de 64%, ce qui n'est pas acceptable. En outre, l'attribution de 512Mo (ou moins) ne permet pas à l'outil de fonctionner. Nous allons donc promouvoir l'utilisation de 2Go de mémoire vive minimum et en recommander 4 pour des performances plus stables (pour un gain de 20 %)

3.1.2 Optimisons l'aléatoire

Lorsqu'une donnée est insérée dans une base de données, une clé primaire permettant de l'identifier¹ doit être définie. Il est possible de la définir manuellement, mais dans de nombreux cas cette dernière est une séquence de chiffres et de lettres générées de manière aléatoire et placée dans une colonne "id". Lorsque le nombre de données insérées est peu conséquent, la durée de génération de cette séquence est négligeable. Cependant nous travaillons ici sur un aspect Big Data, l'optimisation de l'algorithme permettant cette génération aléatoire (appelé aussi "RNG" pour "Random Number Generator") est donc primordiale.

Générer un nombre aléatoire est très laborieux car il est difficile de prouver qu'un algorithme ne suit pas un schéma non encore découvert. Dans les systèmes d'exploitation actuels, il existe un espace spécialement dédié à la génération de ces nombres. Cet espace contient l'agrégation de plusieurs événements aléatoire physique, l'évolution du déplacement de la souris en est un exemple [15]. Étant des phénomènes physiques et naturels, ils ne suivent pas une logique algorithmique et se rapprochent donc d'un réel processus aléatoire.

Cependant, certains ordinateurs ne possèdent pas assez d'éléments physiques (carte son, clavier, souris...) pour pouvoir générer suffisamment de séquences aléatoires et l'espace dédié est donc peu rempli. C'est un problème lorsqu'un programme comme le nôtre a besoin de récupérer des millions de séquence afin de pouvoir satisfaire sa propriété de clés primaires.

C'est pourquoi un algorithme appelé HAVEGE a été développé par l'IRISA (Institut de Recherche en Informatique et Systèmes Aléatoires) afin de palier ce problème. Il se base sur les variations du temps d'exécution d'un code sur un processeur. Étant donné qu'il est presque impossible pour un morceau de code de s'exécuter durant une période exactement identique (même dans un environnement avec du matériel similaire) cela convient pour une source aléatoire.

Cet algorithme est disponible gratuitement à travers la librairie haveged² mais n'est pas installé nativement sur tous les systèmes d'exploitation. Le gain de temps de génération d'une séquence aléatoire est impressionnant. Avec notre dispositif nous

1. une clé primaire est la donnée qui permet d'identifier de manière unique un enregistrement dans une table. Une clé primaire peut être composée d'un ou de plusieurs champs de la table. Deux lignes distinctes de la table ne peuvent pas avoir les mêmes valeurs pour les champs définis au niveau de la clé primaire.

2. <http://www.issihosts.com/haveged/>

avons réussi à multiplier par 30 le nombre de données aléatoires présentes dans l'espace dédié (/dev/random sur les environnements Linux). Nous préconisons donc dans notre norme l'installation de cette librairie afin de fluidifier et d'accélérer l'enregistrement de données dans elasticsearch.

Il est bon de savoir que par défaut le système d'exploitation Windows gère cette problématique.

3.2 Norme-outil

D'après notre radar de caractéristiques 2.5, l'outil ELK possède quelques lacunes au niveau de la portabilité, du RTO/RPO ainsi que de la sécurité. Notre norme va permettre de compenser en partie ceci grâce à quelques ajustements.

3.2.1 Bouclons la

Aussi appelée "localhost", la boucle locale est un procédé permettant de limiter la communication de certains processus avec le monde extérieur.

En effet, lorsqu'un programme souhaite accéder à une ressource située en dehors de son environnement, il fait appel à une interface réseau lui permettant d'établir la communication. Cette interface lui alloue alors un "port" à partir duquel les informations vont transiter. Ce port est par conséquent ouvert aux vulnérabilités du monde extérieur si il est mal configuré ou que le code du programme est mal implémenté. Elasticsearch a été développé en architecture restful³. C'est à dire que l'ajout ou la suppression d'une donnée se déroule à partir du protocole HTTP (GET pour accéder à la donnée, PUT pour en ajouter, DELETE pour en supprimer...)⁴. Cela a pour conséquence une communication avec l'interface réseau et donc une plausible vulnérabilité. Par défaut, le port alloué à elasticsearch est le 9200.

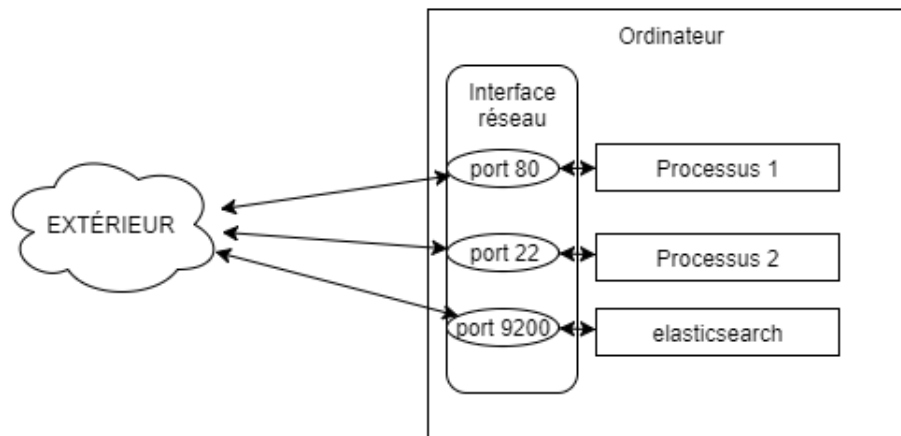
Afin de limiter ce problème de sécurité, deux solutions sont disponibles. La première consisterait à surcharger les fonctions de l'outil elasticsearch en y ajoutant des critères de sécurisation plus poussée, comme une identification par token⁵. Étant opensource, le code est en effet intégralement disponible et modifiable. Néanmoins cette solution n'est pas à conseiller car toute mise à jour de l'outil serait susceptible

3. https://fr.wikipedia.org/wiki/Representational_state_transfer

4. <https://developer.mozilla.org/fr/docs/HTTP/M%C3%A9thode>

5. https://fr.wikipedia.org/wiki/Jeton_d%27authentification

FIGURE 3.2 – Schéma d’une interface réseau et de ses ports



d’effacer nos modifications et rendrait donc nos efforts caducs. La seconde serait de couper la connexion du port 9200 avec l’extérieur et de la rediriger vers un de nos processus locaux qui vérifierait l’intégrité des communications entrantes avant d’interagir avec elasticsearch. Nous penchons plutôt vers ce compromis car d’une part le code initial reste intègre et d’autre part si nous souhaitons basculer d’elasticsearch vers une autre base de données nous pourrions récupérer une grosse partie du programme.

Cette problématique d’outil "d’interfaçage" a déjà été résolue de façon efficace par plusieurs projets opensource dont le plus connu (et récemment racheté par IBM) se nomme loopback.⁶ Le gain de temps est conséquent et de nombreuses fonctionnalités d’authentification comme le protocole OAuth2⁷ sont disponibles.

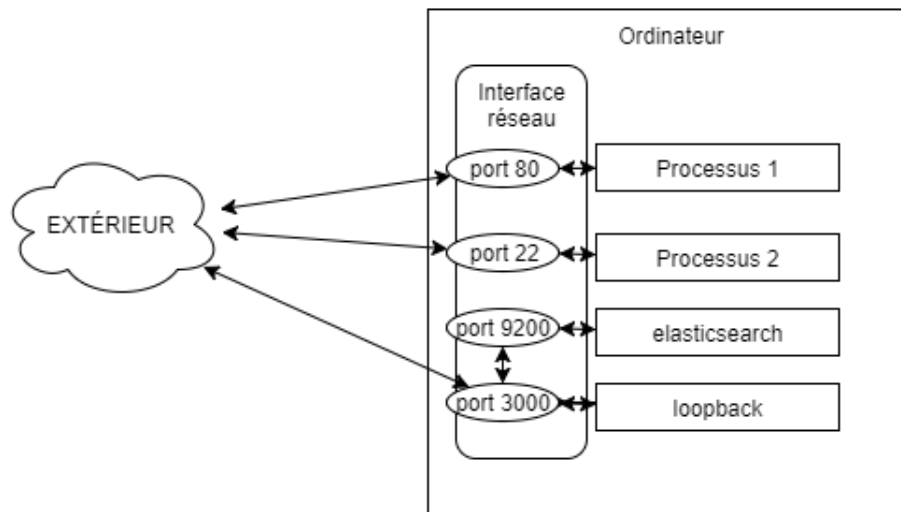
Explicité dans la figure ci-dessus, il est impossible de communiquer directement avec elasticsearch à partir d’une source extérieure. Tout est filtré par l’outil loopback situé sur un port différent. Ce filtrage est l’agrégation d’un ensemble de fonctions que nous pouvons paramétrer en fonction de nos besoins. Parmi elles, certaines concernent la sécurité, la documentation ou encore la génération d’un swagger⁸ qui est une norme de définition d’API (rappelons qu’elasticsearch est basé sur une architecture restful)

6. <https://github.com/strongloop/loopback>

7. <https://oauth.net/2/>

8. <https://swagger.io/specification/>

FIGURE 3.3 – Boucle locale sur le port elasticsearch 9200



3.2.2 Protégeons l'interface

Sécuriser notre base de données dans une boucle locale est une bonne chose, mais il est toujours possible d'y accéder à partir de l'interface graphique Kibana, présente dans le package ELK⁹. Cette interface est un affichage web permettant de créer des graphiques et d'observer en temps réel les analyses traitées par l'ETL logstash.

Là encore, cet affichage est directement accessible à partir de l'extérieur, sans authentification, par n'importe quel navigateur. Nous pourrions boucler localement cette visualisation comme précédemment. Cependant, l'intérêt de Kibana étant de pouvoir visualiser des données locales sur des postes distants, cela ne serait pas judicieux. Nous allons donc plutôt mettre en place un proxy inverse.¹⁰ Ce proxy sera protégé par un mot de passe et redirigera directement l'utilisateur vers le port correspondant à celui de kibana.

9. Pour rappel, ELK contient trois produits : Elasticsearch comme base de données, Logstash comme ETL et Kibana comme interface graphique

10. "un proxy inverse permet à un utilisateur d'Internet d'accéder à des serveurs internes, une des applications courantes du proxy inverse est la répartition de charge (load-balancing)". Source : wikipédia

FIGURE 3.4 – Capture d'écran de l'affichage web Kibana

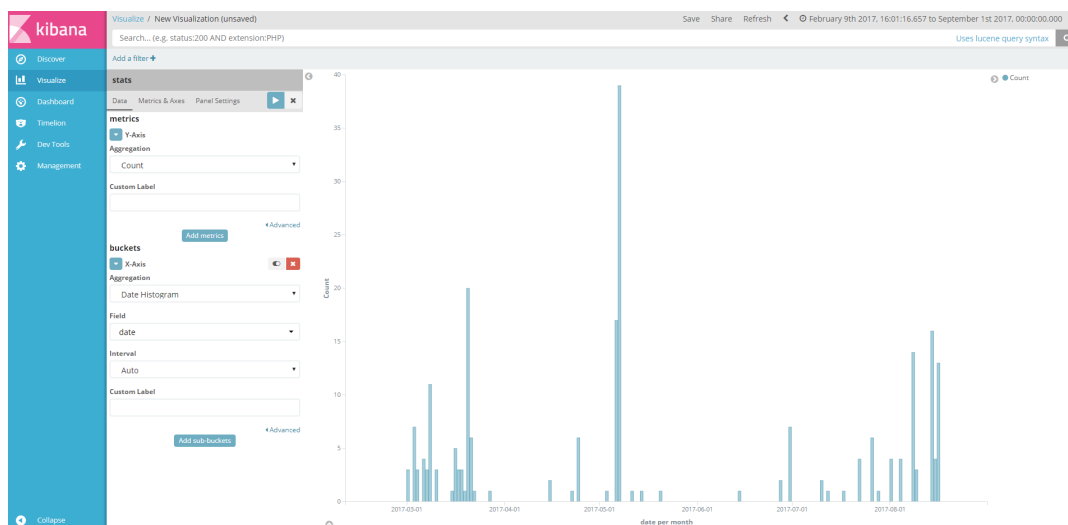
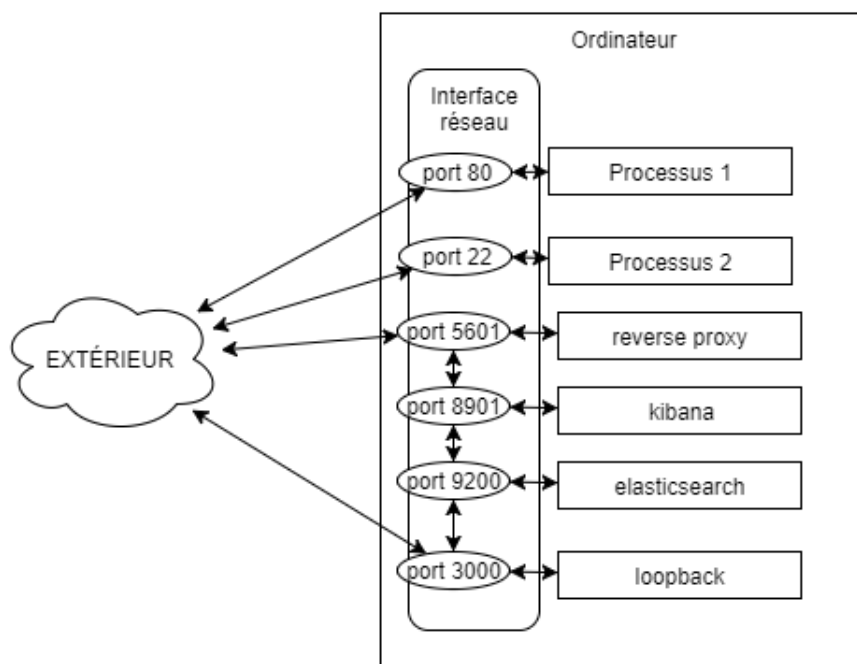


FIGURE 3.5 – Implémentation d'un reverse proxy sur Kibana



Le port 5601 est défini par défaut pour kibana, nous avons donc décidé de l'attribuer au reverse proxy afin de limiter au maximum la perte des normes Elastic. Le port 8901 est un port choisi aléatoirement lors de la modification du fichier de configuration de Kibana. Les accès sont définis directement dans un fichier texte

accessible seulement par l'administrateur. Ci-dessous, la création de notre proxy inverse grâce à l'outil Nginx.

```
server {  
    listen 5601;  
    location / {  
        auth_basic "Restricted Access";  
        auth_basic_user_file /etc/nginx/conf.d/kibana.htpasswd;  
        proxy_pass http://localhost:8910;  
    }  
}
```

Notre interface kibana ainsi que notre base de données elasticsearch sont maintenant pleinement sécurisées grâce à une surcouche logicielle qui les coupe des communications extérieures directes.

3.2.3 Darwinons

Pour peu qu'il soit maintenu, un programme est voué à évoluer en permanence. De nouveaux algorithmes sont découverts et de nouvelles libraires sont composées. Cela a pour conséquence le fractionnement du logiciel en différentes versions plus ou moins compatible entre elles mais surtout compatible avec l'environnement qui les encadre. En informatique, on parle de "portabilité".

L'environnement ELK nécessite Java 8 afin de fonctionner. Ce langage opensource est très populaire et justement réputé pour son aspect portable. Une simple vérification du numéro de version semble convenir. En revanche, il est indispensable d'édifier dans un premier temps l'arborescence la plus compréhensible possible pour pouvoir manipuler efficacement les trois outils ELK. Dans un second temps, d'assurer une compatibilité entre ces trois outils ou au minimum un avertissement si l'un d'eux venait à ne plus être mis à jour.

Afin de centraliser tous nos outils ELK, créons un répertoire nommé "ELK" dans lequel nous entreposerons les trois programmes : Kibana, Logstash et Elasticsearch. Il apparaît qu'une fois chaque outil téléchargé, ces derniers sont suffixés de leur numéro de version. C'est utile lorsque nous avons besoin de savoir s'ils sont à jour mais bien moins lorsque les scripts que nous développons à côté utilisent le nom exact du

dossier afin de fonctionner. Ils sont donc amenés à être remaniés régulièrement du fait de la modification du nom apporté par le changement de version.

Il n'est cependant pas nécessaire de tout renommer. Un lien symbolique¹¹ redirigeant vers le nom de l'outil dépourvu de son numéro de version est suffisant. Les scripts pointent vers ce lien et sont donc automatiquement conduits vers les nouvelles versions. Attention toutefois, le lien symbolique doit être réécrit à chaque fois qu'une mise à jour survient. Effectuons la première implémentation de cette norme, une fois après avoir téléchargé les programmes adéquats et les avoir placés au sein d'un même répertoire "elk".

```
ln -s /home/user/elk/elasticsearch-5.5.2
/home/user/elk/elasticsearch

ln -s /home/user/elk/kibana-5.5.2
/home/user/elk/kibana

ln -s /home/user/elk/logstash-5.5.2
/home/user/elk/logstash
```

Nos scripts n'ont maintenant plus besoin d'être modifiés.

La seconde tâche de portabilité consiste à vérifier la comptabilité de nos différents outils entre eux. Pour cela nous devons créer un outil permettant de vérifier automatiquement que les numéros suffixés soient bien en adéquation avec la matrice fournie par Elastic.¹²

Ces deux tâches réalisées, notre environnement ELK est maintenant davantage apte à être porté sur différents environnements

11. Un lien symbolique est un fichier qui pointe vers un autre fichier; si vous supprimez le fichier cible, les liens symboliques pointeront alors vers un fichier inexistant. Source : https://doc.ubuntu-fr.org/lien_physique_et_symbolique

12. https://www.elastic.co/fr/support/matrix#matrix_compatibility

3.2.4 Snapshot, Backup & Restore

La préconisation RPO/RTO¹³ est appréciée des entreprises cela correspond à des indicateurs de sécurité en cas de sinistre. Plus ce chiffre est bas, moins les données perdues et le temps d'inactivité seront conséquents. Notre objectif est donc ici de normer notre environnement ELK afin de satisfaire à ces attentes.

Elasticsearch est fourni avec des librairies permettant de créer des captures en temps réel de sa base [8].

Dans un premier temps, il est nécessaire de créer un répertoire dans lequel ces données seront archivées. Pour limiter les communications réseau, nous allons le créer localement et indiquer le type "fs", à contrario du type "url" qui le créerait sur un serveur distant. Afin de ne pas s'égarer dans l'arborescence du système, il est préférable de le placer dans le répertoire elasticsearch avec le nom "snapshots". Une option de compression est disponible, cela nous servira à optimiser l'espace disponible sur le disque.

```
$ curl -XPUT 'http://localhost:9200/_snapshot/memoire' -d '{
  "type": "fs",
  "settings": {

    "location": "/home/user/elk/elasticsearch/snapshots",
    "compress": true

  }
}'
```

Une fois ce répertoire créé, il nous faut développer un script fonctionnant de manière autonome et récurrente (aussi appelé daemon), qui aura pour objectif de démarrer des sauvegardes de la base. Commencer par une périodicité quotidienne est un bon compromis entre espace de stockage et RPO, mais cela dépend évidemment de l'importance des données stockées. Il incombe à chaque utilisateur de définir à l'avance ses besoins en cas de panne.

Ce script découlera d'une commande disponible dans la documentation Elastic, de-

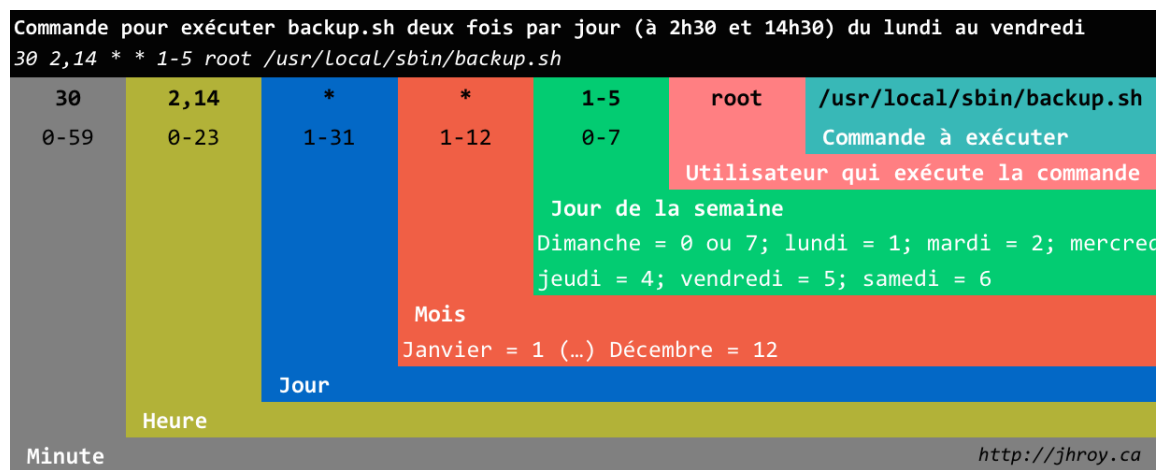
13. Le RTO ou Recovery Time Objective, peut se traduire par la durée maximale d'interruption admissible. RPO ou Recovery Point Objective, désigne la durée maximum d'enregistrement des données qu'il est acceptable de perdre lors d'une panne. Le fait de quantifier le RPO définit en fait les objectifs de sauvegarde. Source : journaldunet

vant contenir le nom de notre répertoire "memoire" créé précédemment ainsi qu'un nom de snapshot que nous appellerons first_snapshot

```
$ curl -XPUT 'localhost:9200/_snapshot/memoire/first_snapshot'
```

Afin de gérer la périodicité d'appel de notre script, il existe dans le système d'exploitation un programme dédié appelé crontab¹⁴. Nous allons ajouter dans cette dernière une ligne qui exécutera notre commande de sauvegarde à une périodicité précise. L'ajout d'une ligne doit cependant être structuré selon un modèle particulier.

FIGURE 3.6 – Exemple d'insertion de ligne dans une crontab. Source : Wikipedia.org



Pour notre part, nous souhaitons activer la commande quotidiennement, de préférence à une heure à laquelle le serveur est susceptible d'être le moins actif. Là encore cet horaire varie en fonction des données collectées. Ici, nous fixerons l'horaire à 4h du matin. Plaçons notre script dans un répertoire "scripts" placé dans le dossier "elasticsearch" et nommons le "snapshot_elastic.sh". Notre ligne crontab correspond donc à :

```
* 04 * * * root /home/user/elk/elasticsearch/scripts
/snapshots_elastic.sh
```

L'implémentation de la sauvegarde quotidienne étant faite, il convient d'en faire une seconde que nous appellerons "sauvegarde brute". Cette sauvegarde ne sera pas un snapshot, mais une copie pure du dossier contenant les données de la base. Dans

14. crontab est un programme qui permet aux utilisateurs des systèmes Unix d'exécuter automatiquement des scripts, des commandes ou des logiciels à une date et une heure spécifiées à l'avance, ou selon un cycle défini à l'avance. Source : wikipédia.org

le cas extrême où une mise à jour aurait corrompu l'intégrité d'un snapshot, nous serions quand même en mesure de récupérer des données, après un traitement un peu plus fastidieux néanmoins. Les données brutes sont situées dans le dossier /data à la racine du dossier elasticsearch. Nous allons donc rajouter dans notre script de sauvegarde deux commandes supplémentaires exécutant d'une part la copie du dossier data dans notre dossier snapshots et d'autre part la compression de ce dossier afin de limiter sa taille.

```
curl -XPUT 'localhost:9200/_snapshot/memoire/
first_snapshot '

cp -R /home/user/elk/elasticsearch/data/
/home/user/elk/elasticsearch/snapshots/archives/

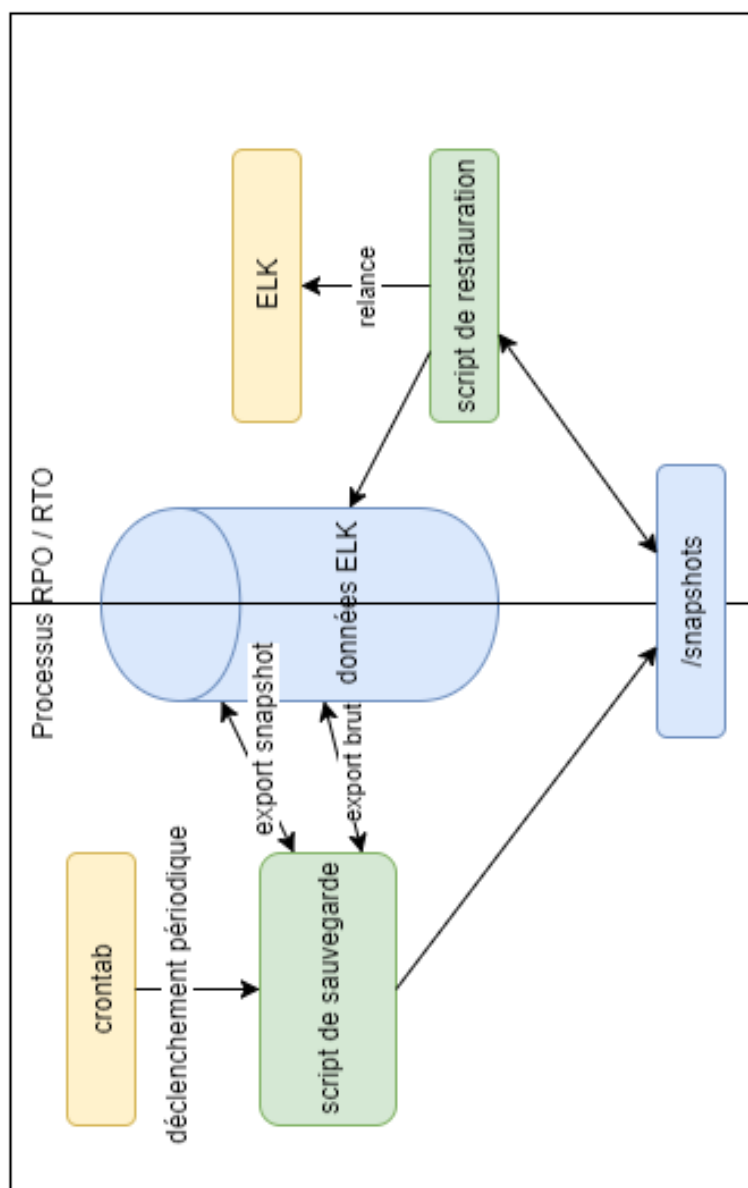
tar -czf /home/user/elk/elasticsearch/snapshots/archives
/data
/home/user/elk/elasticsearch/snapshots/archives/data.tgz
```

La périodicité n'est pas à redéfinir car le chemin du script initial reste inchangé. Une double sauvegarde nous permet donc d'avoir une redondance et une tolérance accrue face aux erreurs d'intégrité et aux pannes.

Cette étape réalisée, il nous reste toutefois à concevoir le mécanisme de relance de l'environnement ELK. Nous avons vu antérieurement qu'il était composé de trois outils indépendants les uns des autres. Lors d'un crash il est donc nécessaire de les redémarrer manuellement un par un, ce qui peut s'avérer laborieux et peu pratique. Le but du script est donc d'alléger cette tâche en proposant à l'utilisateur de relancer automatiquement les outils éteints nécessaires à l'environnement d'analyse. Ce script permettra aussi de restaurer les données si ces dernières ont été corrompues. Pour aller plus loin et afin de sécuriser encore davantage nos données, il est potentiellement envisageable d'exporter les snapshots sur un serveur distant.

Grâce à cette norme-outil RTO/RPO, la relance de l'activité d'analyse s'effectue désormais en une seule commande et la tolérance aux incidents est en outre accentuée.

FIGURE 3.7 – Schématisation du processus de relance RTO / RPO



Évaluation

4.1 Notation

Grâce à nos travaux précédents concernant la pondération des différentes caractéristiques, nous avons d'ores et déjà la plupart des éléments qui vont nous permettre d'établir une note d'évaluation de notre environnement d'analyse. Doté de 11 normes, nous allons établir un résultat sur une base 100. D'une part cela nous permettra plus d'agilité dans la notation et d'une seconde part le pourcentage sera plus parlant psychologiquement pour l'utilisateur. La caractéristique de fonctionnement est primordiale car nécessaire au bon fonctionnement de notre environnement.

TABLE 4.1 – Normes établies

Nom	Caractéristique	Note
Existence de Java 8	Fonctionnement	20
Mémoire vive disponible	Fonctionnement	15
Optimisation de l'aléatoire	Vélocité	10
Arborescence des dossiers respectée	Portabilité	5
Vérification des liens symboliques	Portabilité	5
Matrice de compatibilité	Portabilité	10
Boucle locale elasticsearch	Sécurité	10
Reverse proxy Kibana	Sécurité	10
Snapshots	RTO/RPO	5
Backups	RTO/RPO	5
Restore / Relance	RTO/RPO	5

4.2 Outillage

Afin d'être aisément compréhensible, l'affichage du résultat ne doit pas se faire de manière désagréable au sein d'une console noire, sans aide ni explication. Souhaitant faire adhérer le maximum d'utilisateur à notre norme, sa représentation doit être ergonomique et explicite. Pour cela, une interface accessible à partir de n'importe quel navigateur semble être la solution la plus efficace. (pas de logiciel supplémentaire à télécharger, chaque poste ayant un navigateur par défaut)

Non seulement cette interface permettra de visualiser en direct l'évolution de la normalisation de l'environnement mais aussi le lourd aspect de configuration de l'outil ELK (notamment à travers l'activation manuelle des scripts développés auparavant) à travers une console sera facilité par l'apport graphique. L'intérêt de cet affichage est donc double : permettre aux utilisateurs confirmés d'adapter leur outil à cette norme tout en facilitant le labeur de configuration aux utilisateurs plus "novices" démarrant de zéro, en implicitant ce travail de normalisation.

Cependant, il faut que ce surplus logiciel ne soit pas trop imposant et n'ait d'usage que celui que l'on souhaite. Cela signifie que nous avons besoin d'un langage modulaire, sans fonctions ni librairies tierces supplémentaires à l'initialisation. Le choix va donc être prioritairement porté ici sur le poids du code nécessaire à l'exécution du langage.

La lecture des documentations correspondantes nous a permis de vérifier que les langages exposés dans le tableau ci-dessous étaient suffisamment matures pour pouvoir y implémenter les fonctions de vérifications de nos normes. C'est à dire des exécutions de commandes, de lecture/écriture de fichiers ou de diagnostic des composants du système.

TABLE 4.2 – Poids des langages

Nom	Poids
Java JEE	130Mo
Ruby on rails	100Mo
PHP	70Mo
Python django	50Mo
NodeJS	25Mo

Après avoir téléchargé chaque environnement, identifié leur poids sur le disque et

trié par ordre décroissant, le choix s'est porté sur le langage NodeJS¹. Sa modularité en font en outre un atout indéniable.

Nous savons donc maintenant quel langage va nous permettre de créer l'application web qui répondra à notre problématique de normalisation. Il reste néanmoins une question à se poser : comment le programmer ? En effet, lorsqu'un programme est développé, il est très important de détailler le code dans son entièreté pour que les prochains développeurs puissent ajouter de nouvelles fonctionnalités sans trop de difficulté. Mais la façon de penser n'est pas à la même pour chaque individu, il est donc essentiel d'utiliser un modèle plus universel : un "framework".

L'utilisation d'un framework est très appréciée pour sa portabilité, sa réutilisation, sa maintenance et son évolutivité. Il permet de définir un squelette qui guidera notre code pour les futurs développeurs. Il existe des frameworks pour n'importe quel langage et certains sont plus reconnus que d'autres. Ici, nous allons nous intéresser à ceux employant le modèle MVC (Modèle-Vue-Contrôleur), car c'est le plus courant et il permet de découper les logiques techniques et métiers du programmeur (Contrôleur et Modèle) avec les logiques de présentation destinées aux designers (Vue).

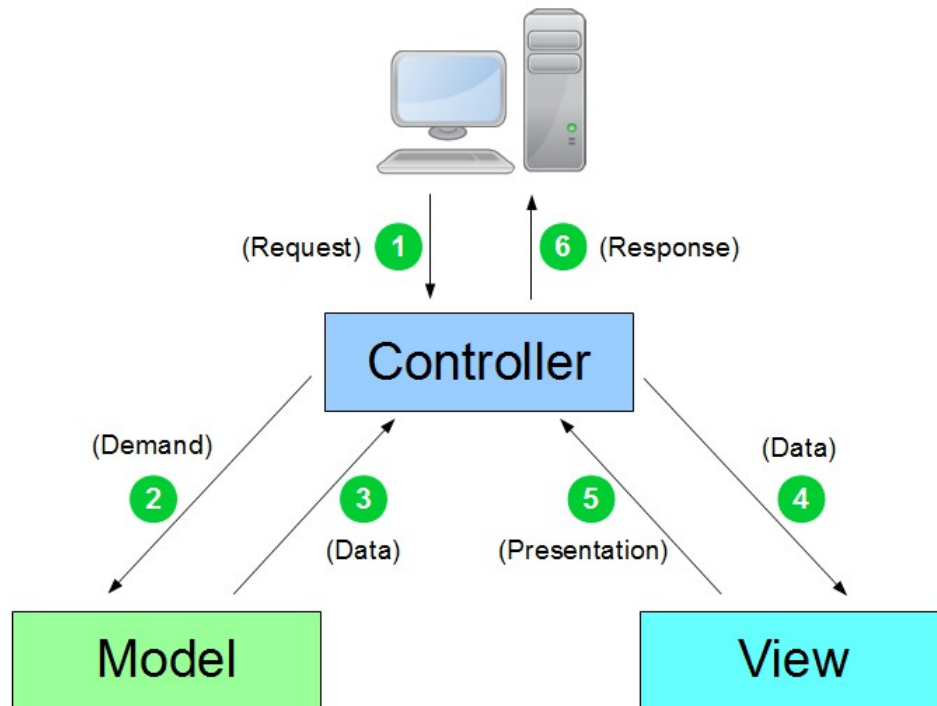
Le fonctionnement du modèle MVC peut se décomposer en six étapes :

- L'utilisateur envoie une requête HTTP vers le serveur (il appuie sur le bouton "rechercher" de Google par exemple)
- Le Contrôleur va intercepter cette requête et va demander au Modèle les données de résultats de cette requête (donc les résultats de la recherche de l'utilisateur)
- Le Modèle prépare ces résultats et les renvoie au Contrôleur.
- Le Contrôleur appelle enfin la Vue et lui transmet les données du Modèle
- La Vue va définir la façon dont cette page de réponse doit être affichée et va la renvoyer au Contrôleur (par exemple, afficher les données sous forme de liste)
- Le Contrôleur peut ainsi envoyer une réponse à l'utilisateur.

Afin d'assurer le suivi et la popularité de notre outil de normalisation, nous avons besoin du framework NodeJS open-source le plus populaire implémentant le modèle MVC. Une source fiable pour parvenir à le trouver est notamment l'inspection de la plateforme Github et de ses "stars". Plus le produit a de "stars", plus il est populaire

1. <https://nodejs.org/en/>

FIGURE 4.1 – Schéma d’une architecture MVC. Source : <http://www.adventy.org/>



au sein de la communauté du monde entier. Concernant les frameworks NodeJS, le produit en ayant le plus (et de loin) est ExpressJS.² Il ne faut pas non plus oublier vérifier sa dernière date de mise à jour, qui est ici convenable car datant de moins d’un mois.

Notre outil de normalisation et de notation est donc être écrit en NodeJS au travers le framework web ExpressJS. Il est intégralement disponible ici : <https://github.com/AlexisAnzieu/normalizerBigData>

2. <https://github.com/expressjs/express>

Projection

L'outil présenté n'aborde malheureusement pas toutes les caractéristiques de la norme, du fait de l'importante batterie de test et de contrainte à entreprendre afin de pouvoir gérer tous les environnements disponibles. Mais sa licence ouverte le laisse libre de modifications à qui le souhaite et il se peut qu'il soit pleinement opérationnel à terme.

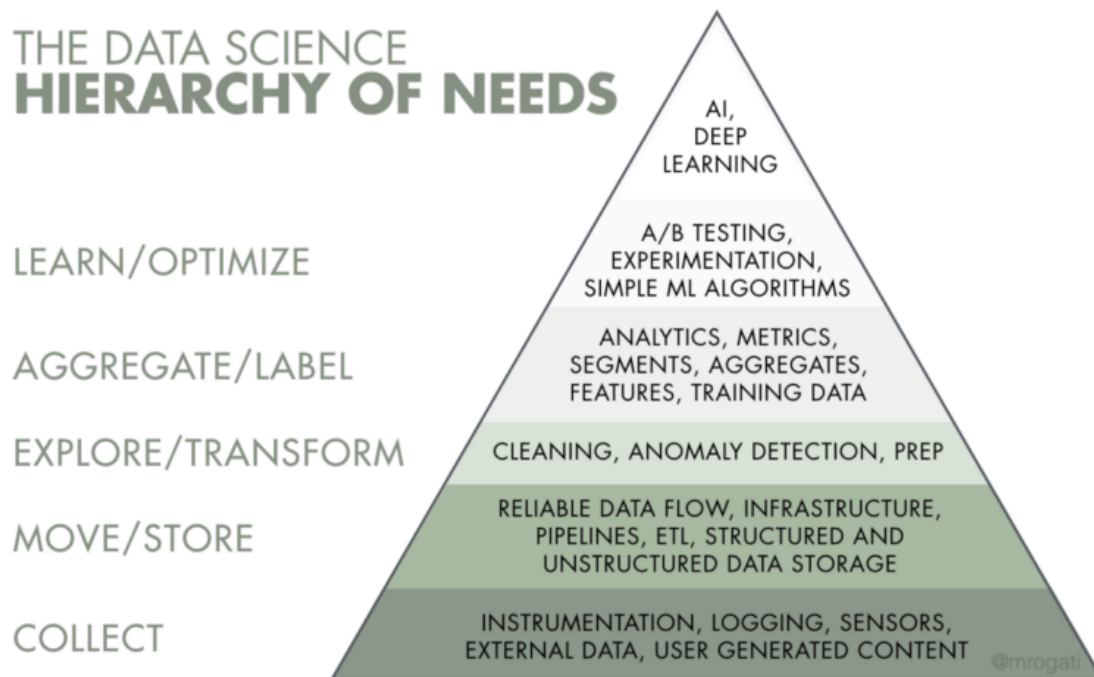
Concernant le proxy renverse de Kibana, nous avons vu qu'il suffisait d'écrire les identifiants dans un fichier texte pour pouvoir accéder à l'interface. En allant plus loin, il serait habile de créer des sessions différentes entre chaque utilisateur afin qu'ils aient accès à leurs propres graphiques. Une hypothèse serait de mettre en place une surcouche d'interface web dans lequel nous importerions les différents graphiques.

Par ailleurs, la course à l'analyse est en plein essor, mais encore faut-il savoir ce que l'on souhaite analyser et si cela se corrèle avec l'esprit du Big Data. Pourquoi ne pas mettre en place un didacticiel qui guiderait les utilisateurs (ou organisations) vers la manière la plus cohérente de traiter leurs données, avant même de plonger dans l'univers Big Data. En effet, de nombreux projets tombent dans l'oubli car par souci de modernité nous oublions le réel objectif à atteindre.

Enfin, et dans un ultime élan de projection, viendrait le déploiement d'un algorithme d'Intelligence Artificielle sur les données produites et stockées avec fiabilité. C'est effectivement l'étape qui adviendrait après ce processus de normalisation [13] . Il faut mieux passer plus de temps à se constituer un robuste socle de données plutôt que de se lancer directement dans son exploitation.

Transposée de la fameuse pyramide de Maslow, voici la pyramide des besoins du Big Data.

FIGURE 5.1 – Pyramide des besoins des sciences de données. Source : Hackernoon



Conclusion

Au terme de cette troisième année en alternance, j'ai eu le loisir de comprendre et de sonder les rouages d'un processus d'innovation Big Data à petite échelle sur tous les niveaux. Du cahier des charges du client, j'ai dû en déduire les besoins indirects, créer un environnement propice au bon fonctionnement de l'application (choix des technologies et des capacités matérielles) et choisir la meilleure façon de la mettre en place selon des objectifs actualisés en permanence.

Cette aventure m'a fait comprendre l'importance de la normalisation d'un processus itératif. La conception d'une norme est laborieuse mais une fois établie, le déroulement du processus est automatique et le gain de temps est conséquent. Cependant, l'adoption d'une norme par une communauté est longue et fastidieuse. Il faut donc faire en sorte qu'elle soit intégrée au fur et à mesure par des outils qui la mettent implicitement en place. Dans mon cas, il s'agit du programme développé : `normalizerBigData`¹.

De par cette automatisation de l'implémentation d'un environnement d'analyse Big Data, il est maintenant aisé pour n'importe quel particulier (ou professionnel) de stocker, de centraliser et de traiter ses données dans un environnement fiable.

1. <https://github.com/AlexisAnzieu/normalizerBigData>

Bibliographie

- [1] Neil Biehn. The missing v's in big data : Viability and value. <https://www.wired.com/insights/2013/05/the-missing-vs-in-big-data-viability-and-value>, may 2013.
- [2] Jackson Brian. Top 10+ log analysis tools – making data-driven decisions. <https://www.keycdn.com/blog/log-analysis-tools/>, november 2016.
- [3] Michael Cox and David Ellsworth. Application-controlled demand paging for out-of-core visualization. <https://www.nas.nasa.gov/assets/pdf/techreports/1997/nas-97-010.pdf>, july 1997.
- [4] Ase Dragland. Big data, for better or worse : 90% of world's data generated over last two years. <https://www.sciencedaily.com/releases/2013/05/130522085217.htm>, may 2013.
- [5] Elastic. Elasticsearch benchmarks. <https://elasticsearch-benchmarks.elastic.co/>, 2016.
- [6] Elastic. Hardware. <https://www.elastic.co/guide/en/elasticsearch/guide/current/hardware.html>, 2016.
- [7] Elastic. Github geonames track. <https://github.com/elastic/rally-tracks/tree/master/geonames>, 2017.
- [8] Elastic. Snapshot and restore. <https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-snapshots.html>, 2017.
- [9] Inc. Gartner. Press release. <http://www.gartner.com/newsroom/id/1731916>, june 2011.
- [10] Geonames.org. Geonames gazetteer extract files. <http://download.geonames.org/export/dump/>, 2017.
- [11] Mikal Khoso. How much data is produced every day ? <http://www.northeastern.edu/levelblog/2016/05/13/how-much-data-produced-every-day>, may 2016.
- [12] Oracle. Quelle est la configuration minimale requise pour java ? <https://www.java.com/fr/download/help/sysreq.xml>, 2016.

- [13] Monica Rogati. The ai hierarchy of needs. <https://hackernoon.com/the-ai-hierarchy-of-needs-18f111fcc007>, 2017.
- [14] Statista. Average selling price of desktop personal computers (pcs) worldwide from 2005 to 2015 (in u.s. dollars). <https://www.statista.com/statistics/203759/average-selling-price-of-desktop-pcs-worldwide/>, 2015.
- [15] Unix team. Linux man page - random. <https://linux.die.net/man/4/random>, 2017.
- [16] Wikipédia.org. Pourquoi un kilo-octet est égal à 1024 octets et pas 1000 ? https://fr.wikipedia.org/wiki/Syst%C3%A8me_binaire, 2017.

Organigramme de SILCA

FIGURE A.1 – Organigramme des différentes directions

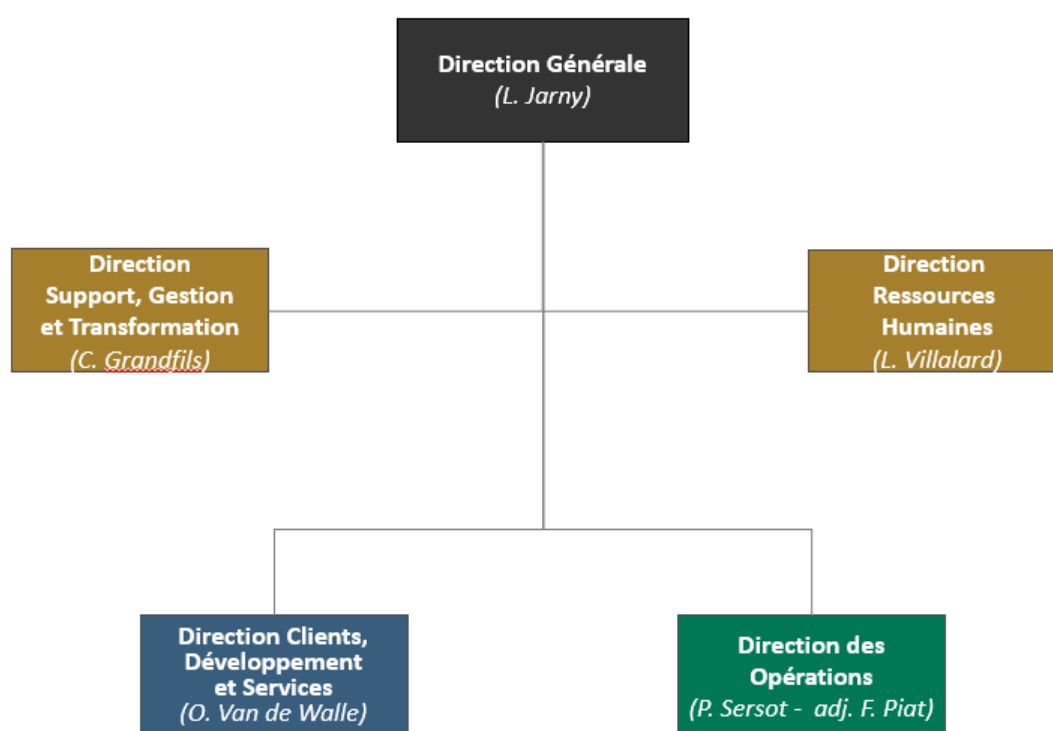
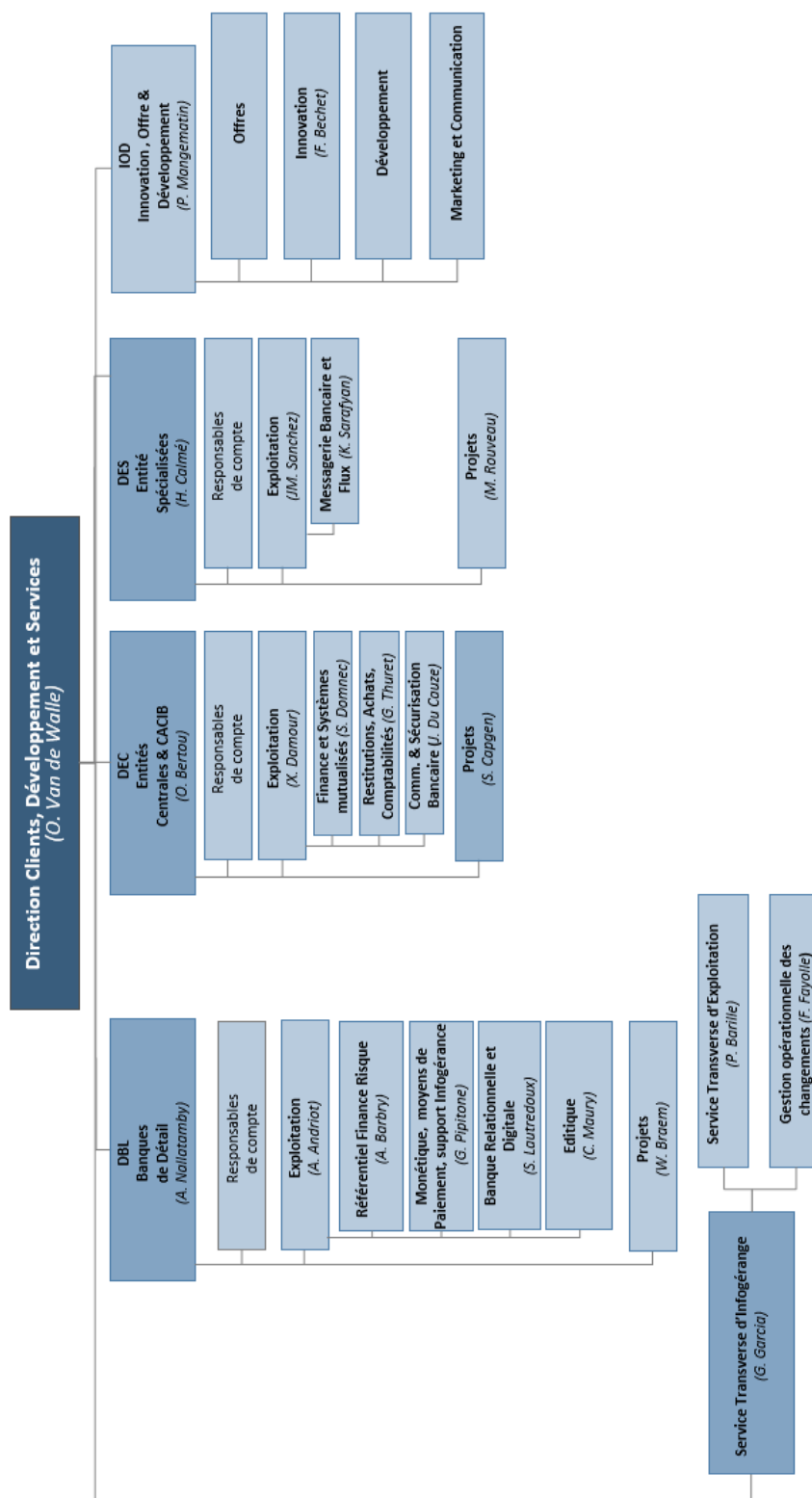


FIGURE A.2 – Organigramme de la direction client



Capture d'écran de l'outil

FIGURE B.1 – Capture d'écran de l'accueil de l'outil de normalisation opérationnel

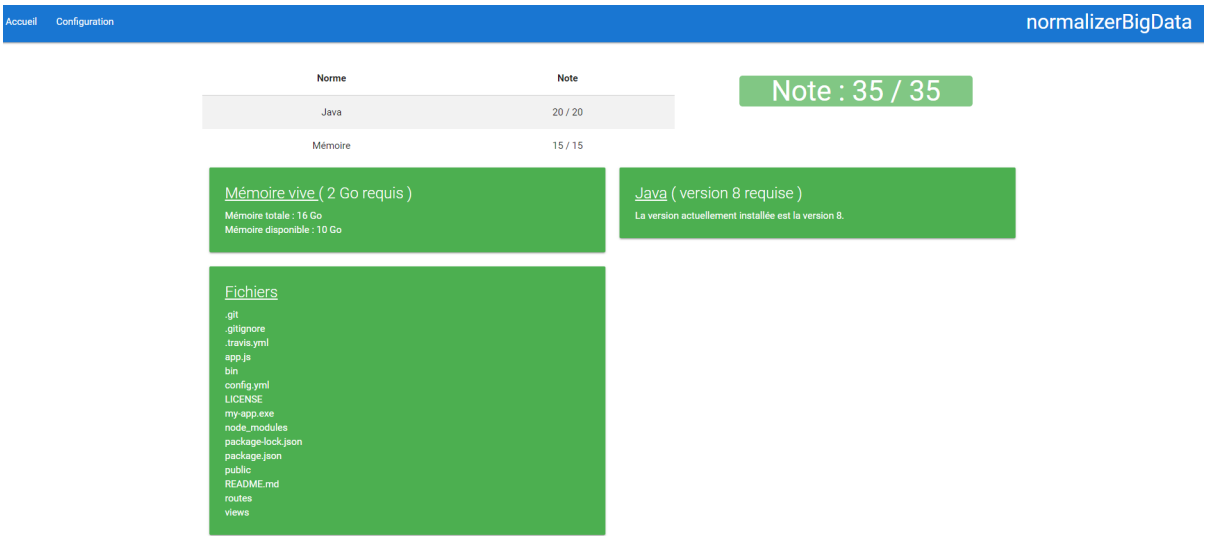


FIGURE B.2 – Capture d'écran de l'accueil de l'outil de normalisation en erreur

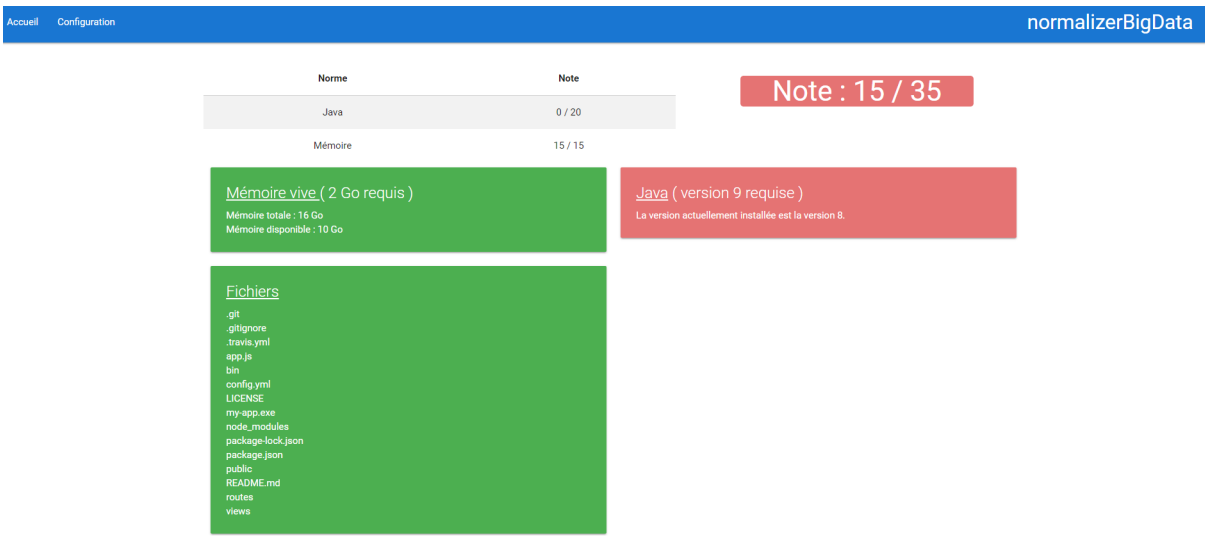


FIGURE B.3 – Capture d’écran de la configuration du fichier elasticsearch


Accueil

Configuration

normalizerBigData

Elasticsearch

cluster.name	node.name	node.attr.rack	path.data
haltevent	node-1	r5	/path/to/data
path.logs	bootstrap.memory_lock	network.host	http.port
/path/to/logs	true	193.70.37.84	9200
discovery.zen.ping.unicast.hosts	discovery.zen.minimum_master_nodes	gateway.recover_after_nodes	action.destructive_requires_name
host1,host2	3	3	false

 ENREGISTRER