HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Département : Technologies de l'information et de la communication
Filière : Informatique et systèmes de communication
Orientation : Informatique logicielle

Travail de Bachelor

Analyse et implémentation d'un système de logging multi-niveau pour une plateforme Smart Grid

Étudiant : Jael Dubey

Travail proposé par : Jonathan Bischof
DEPsys SA
Route du Verney 20B
1070 Puidoux

Enseignant responsable : Nastaran Fatemi

Année académique : 2019-2020

Yverdon-les-Bains, le *JOUR MOIS* 2020

1. Cahier des charges

1.1. Résumé du problème

DEPsys est une entreprise Suisse leader technologique du marché énergétique. Fondée en 2012 et basée à Puidoux, elle fournit des solutions évolutives basées sur sa plateforme GridEye permettant aux réseaux de distribution d'énergie traditionnels de faire face aux nouvelles contraintes de la production décentralisée des sources d'énergie renouvelable, tels que les systèmes photovoltaïques et les technologies de mobilité électrique.

La plateforme GridEye offre une solution technologique innovatrice pour les gestionnaires de réseau de distribution (GRD, p. ex. Romande Énergie). Positionné de manière unique avec sa simplicité de déploiement, c'est la seule solution réellement Plug & Play qui évite tous les problèmes d'installation. La surveillance du réseau électrique en temps réel et les statistiques fournissent des informations détaillées sur les conditions du réseau. Les algorithmes de contrôle et de gestion garantissent la qualité et la stabilité du réseau.

Ce Travail de Bachelor consiste en une évaluation de différentes technologies de logging et de l'analyse des performances, pour ensuite mettre en place la meilleure solution pour l'infrastructure GridEye.

1.2. Étapes de réalisation du projet

- Étude des systèmes de gestion de logs actuels.

  Choix des systèmes à évaluer.
- Évaluation théorique des systèmes.
- Prise en main et configuration des outils pour une comparaison théorique & pratique.

  Choix d'un système.
- Implémentation d'un cas d'utilisation concret fourni par DEPsys.

  Avec un programme de simulation minimaliste de la plateforme GridEye.
- Implémentation d'une démonstration de faisabilité (Proof of Concept, PoC).
- Implémentation de librairies (SDK) pour l'interfaçage avec le Back-End de la solution GridEye (puis Front-End, puis outils externes).

1.3. Informations diverses et technologies

Les logs sont constitués de plusieurs types (System, User Action, Notification, ...) avec différents niveaux de priorités (debug, info, ...). Les logs utilisateurs doivent pouvoir être consultés depuis le Front-End, alors que les logs systèmes peuvent être accessible depuis une dashboard interne. Les notifications, dépendant du niveau de priorité, doivent pouvoir être transmises en temps réel aux gestionnaire de réseau de distribution électrique par e-mail, notification push, etc. ou sous forme de rapport journalier/hebdomadaire. La mise en place de librairies (SDK) est nécessaire pour communiquer avec les différents composants de l'infrastructure. Le but étant de pouvoir s'interfacer avec l'infrastructure actuelle afin de pouvoir remplacer la solution de logging actuelle.

Technologies :

- Base de données : SQL, JSON, ... (dépendant de la solution choisie).
- Back-End : Java
- Front-End : Javascript
- Outils externes : Python 3

Table des matières

2. Introduction

La plateforme GridEye de DEPsys [3] est une solution permettant d'optimiser les réseaux électriques. Elle s'adresse aux gestionnaires de réseau de distribution, et permet de faire face aux contraintes de la production décentralisée, qui correspond par exemple aux panneaux photovoltaïques, ou encore aux batteries de stockage. Autant d'équipements que de plus en plus de particulier se procurent, tout comme des entreprises. Cette transition énergétique est accélérée en Suisse avec la « Stratégie Energétique 2050 », un texte de loi accepté par le peuple en 2017, qui promeut les énergies renouvelables.

GridEye permet une optimisation efficace du réseau grâce à plusieurs applications, ayant principalement pour but d'assurer une grande qualité et une haute fiabilité. Ces qualités sont atteintes avec des outils de surveillance du réseau, d'analyses des événements via des algorithmes sophistiqués, et d'autres fonctionnalités diverses. Le service de GridEye fourni également l'accès à tout un support hautement qualifié. La figure 1 montre les différentes applications de GridEye, ainsi que ses composants et services.

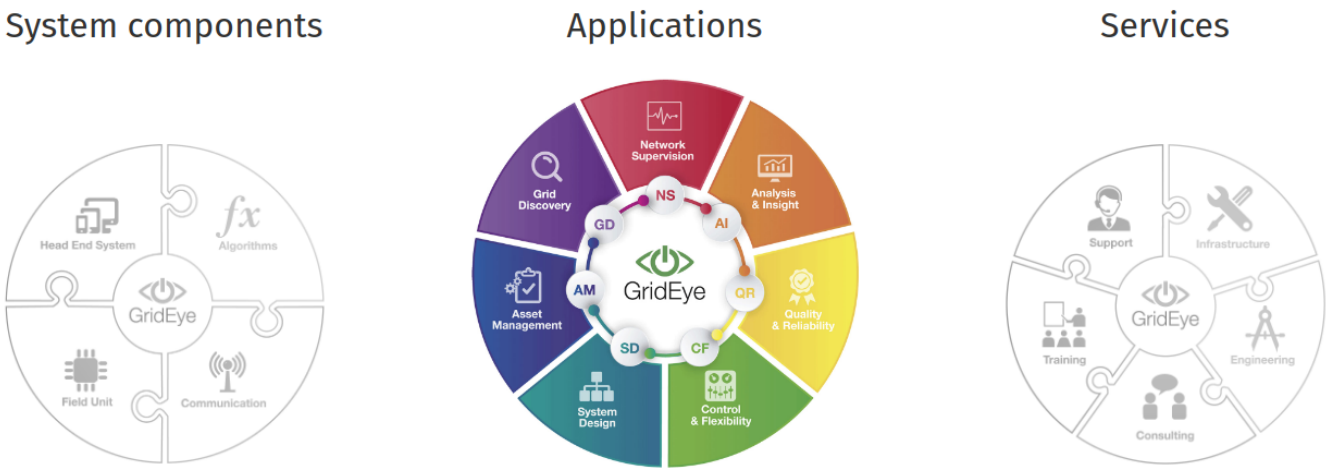

FIGURE 1 – Description de la plateforme GridEye [3]

Ce Travail de Bachelor se positionne dans les applications d'analyse et de surveillance du réseau électrique. Le but est, à terme, de remplacer la solution actuelle par un système de gestion de logs.

3. Évaluation

3.1. Critères d'évaluation

Pour réaliser une évaluation de différents système de gestion de log, il faut obligatoirement choisir des critères d'évaluation. Ces critères se basent sur deux sources. La première étant les demandes formulées par DEPsys, et la deuxième provient des différentes fonctionnalités nécessaires à un système de gestion de logs [12] [4]. Voici les critères retenus :

- La collecte des logs
  Approche minimaliste ou maximaliste, est-ce que le système installe un agent sur le dispositif émetteur de log qui lui envoie uniquement les informations les plus importantes (approche minimaliste, méthode PUSH), ou est-ce que le système reçoit tous les logs et les enregistre tous (approche maximaliste, méthode PULL) ?

- L'agrégation centralisée des logs
  L'agrégation des logs est un défi, car après avoir collecté les logs, il faut tous les regrouper dans un même endroit, alors qu'ils peuvent avoir des formats différents. De plus, ils peuvent être généré très rapidement (s'exprime en EPS, Event Per Second), il faut donc être capable de traiter et regrouper ces logs de manière efficace.

- Le stockage à long terme et la durée de rétention des logs
  Après avoir agrégé ces informations, il faut maintenant faire des choix quant à leur stockage. L'idéal serait de garder tous les logs indéfiniment, mais chaque information stockée a un coût. Il faut donc avoir une stratégie de rétention qui permette de supprimer ou garder tel type de log.

- La rotation des fichiers de logs
  La rotation consiste à rendre automatique la stratégie de rétention et/ou de stockage des logs.

- L'analyse des logs (en temps réel et en vrac après une période de stockage)
  L'analyse des logs est, en quelque sorte, le but de tout le système de gestion de logs. En effet, il ne sert à rien de stocker de l'information sur un système si l'on en fait rien. L'analyse est donc la pour synthétiser les informations contenues dans les logs.

- Les rapports
  Il doit être possible pour un système de gestion des logs d'effectuer des recherches sur les informations stockées et de rédiger des rapports.

- Visionnage et gestion des alertes
  Un des buts d'un système de gestion de logs est de pouvoir réagir très vite à un problème, voire même de l'anticiper. Ceci passe par une émission d'alerte et d'une possibilité de visionnage des données, si possible en temps réel.

- Popularité
  Voici un critère qui change en permanence, mais qui a son importance lors d'un choix d'outil informatique. En effet, un logiciel sur le déclin sera de plus en plus dur à supporter, alors qu'un outil trop jeune n'a souvent que trop peu d'utilisateurs qui pourraient partager leurs connaissances sur les forums. L'idéal étant donc un logiciel populaire et qui est en pleine croissance.

En plus de ces critères, les coûts d'utilisation seront, si possible, évalués.

3.2. Choix des différents systèmes à évaluer

Comme pour le choix des critères d'évaluation, les différents systèmes qui vont être analysés ont été définis soit par DEPsys, soit par des recherches dans les nombreux classement de « Log Management Tools »disponibles sur internet.
Quatre classements différents ont été sélectionnés afin d'avoir plusieurs avis différents, tout en restant dans une quantité raisonnable d'analyses. Les classements qui allaient être utilisés devaient être neutres. On entend par là que le site réalisant le classement ne doit pas proposer, par exemple, une solution cloud utilisant un certain outil de gestion de log, auquel cas son classement serait forcément biaisé. Il fallait également que le classement soit récent, étant donné la vitesse d'évolution générale des outils informatiques. Une période d'un an maximum a été définie.

Les classements suivants ont été choisis :

- Top 8 BEST Log Management Software [11]

  Classement datant de mars 2020, et publié par *SoftwareTestingHelp*.

- Best Log Manager & Monitoring Software & Tools [2]

  Classement datant de janvier 2020, et publié par *iTT Systems*.

- 6 Best Log Management Tools [1]

  Classement datant de août 2019, et publié par *AddictiveTips*.

- 13 Best Log Management & Analysis Tools [9]

  Classement datant de août 2019, et publié par *Comparitech*.

Les quatre sites internet retenus sont de simples portails informatiques, publiant des articles sur divers sujets informatiques. Lorsque l'on effectue une recherche Google afin de trouver des classements de système de gestion de logs, on tombe régulièrement sur des articles de *DNSStuff* et *Sematext*. Ceux-ci n'ont pas été utilisé car le premier appartient à SolarWinds, une entreprise proposant plusieurs logiciels, dont des systèmes de gestion de logs, et le second propose des solutions cloud utilisant des outils de gestion de logs. Ils ont donc été jugés non-neutre.

Afin de réaliser un classement regroupant le contenu de ces 4 tops, une note a été attribuée à chaque position de chaque classement. Deux critères ont été intégré dans le calcul de la note : la position (mieux on est positionné dans son propre classement, plus on aura de points), ainsi que le nombre total d'outils cités. Il est en effet plus facile d'être bien classé dans un classement contenant 5 outils que dans un classement en contenant 10. Pour finir, afin de faciliter la lecture, la note attribuée se situe entre 0 (le plus mauvais), et 100 (la meilleure note possible). La formule suivante a donc été appliquée :

$$\frac{i * 100}{n}$$

i étant la position dans le classement, et n le nombre total d'outils dans le classement.

Les tableaux 1 et 2 montrent les points obtenus par chaque système.

Et le tableau 3 montre le classement global, calculé d'après l'addition des points obtenus dans les différents classements.

Après une étude légèrement plus approfondie sur ces différents systèmes, il a été décidé de garder les suivants pour l'évaluation :

1. Elastic Stack

   Choisi par DEPsys, probablement la plus populaire.

SoftwareTestingHelp	
Système	Points
SolarWinds Log Analyzer	100
Sematext Logs	89
Splunk	78
ManageEngine EventLog Analyzer	67
LogDNA	56
Fluentd	44
Logalyze	33
Graylog	22
Netwrix Auditor	11

(a) Classement de SoftwareTestingHelp

Comparitech	
Système	Points
ManageEngine EventLog Analyzer	100
SolarWinds Papertrail	92
Loggly	83
PRTG Network Monitor	75
Splunk	67
Fluentd	58
Logstash	50
Kibana	43
Graylog	33
XpoLog	25
ManageEngine SyslogForwarder	17
TekWire Managelogs	8

(b) Classement de Comparitech

TABLE 1 – Classements SoftwareTestingHelp et Comparitech

AddictiveTips	
Système	Points
SolarWinds Papertrail	100
SolarWinds Loggly	83
Splunk	67
Nagios Log Server	50
Graylog	33
ManageEngine EventLog Analyzer	17

(a) Classement de AddictiveTips

iTT Systems	
Système	Points
SolarWinds Log & Event Manager	100
PRTG Network Monitor	83
Lepide	67
McAfee Enterprise Log Manager	50
Veriato	33
Splunk	17

(b) Classement de iTT Systems

TABLE 2 – Classements AddictiveTips et iTT Systems

	Système	Points
1	Splunk	229
2	SolarWinds Papertrail	192
3	ManageEngine EventLog Analyzer	184
4	SolarWinds Loggly	166
5	PRTG Network Monitor	158
6	Fluentd	102
7	SolarWinds Log Analyzer	100
8	SolarWinds Log & Event Manager	100
9	ELK Stack (Kibana + Logstash)	93
10	Sematext Logs	89
11	Graylog	88
12	Lepide	67
13	LogDNA	56
14	Nagios Log Server	50
15	McAfee Enterprise Log Manager	50
16	Logalyze	33
17	Veriato	33
18	XpoLog	25
19	ManageEngine Syslog Forwarder	17
20	Netwrix Auditor	11
21	TekWire Managelogs	8

TABLE 3 – Classement global des systèmes de gestion de log

2. Splunk

   1er du classement.

3. SolarWinds Loggly

   De tous les outils appartenant à SolarWinds, je voulais n'en choisir qu'un. Loggly me paraissaît le plus approprié au cas d'utilisation de ce travail de Bachelor.

4. Graylog

   Suggéré par DEPsys, et semble avoir une bonne documentation.

3.3. Comparaison théorique

3.3.1. Elastic Stack

La « Elastic Stack », ou « Suite Elastic »en français, anciennement appelée « ELK Stack »est composée de plusieurs outils :

- Elasticsearch

  Un moteur de recherche RESTful.

- Kibana

  Un outil de visualisation.

- Logstash

  Un pipeline d'ingestion de log.

- Beat

Une famille d'agent dédié au transfert de données.

La table 4 montre un résumé des fonctionnalités disponible ainsi que des coûts lié à la Suite Elastic.

Elastic Stack	
Collecte	La collecte des logs se fait en approche minimaliste. La suite Elastic contient l'outil Beat, qui est donc une famille d'agent léger. On installe un agent beat (p. ex. Filebeat, Metricbeat, etc.) sur le système générant les logs, et cet agent envoie les données vers le serveur.
Agrégation centralisée	Se fait via Logstash. Peux supporter beaucoup d'événements par seconde ($>$ 10'000 EPS). Compatible avec énormément de type de logs. Permet d'analyser et transformer les logs en temps réel. Logstash dispose d'une API permettant de créer nos propres plug-in, si les sources de données ne sont pas compatible nativement.
Stockage et rétention	Le stockage se fait avec Elasticsearch. Il n'y a pas de rétention des données de bases avec Elasticsearch. Il est cependant possible de le faire avec Elastic-Curator, qui est un outil permettant de gérer un cluster Elasticsearch.
Rotation	La rotation se fait avec Elastic-Curator
Analyse	Elasticsearch et ses requêtes poussées permettent de faire des recherches avancées.
Rapport	Les rapports peuvent être générés depuis Kibana.
Visionnage et alertes	La visualisation des données en temps réels peut se faire avec Kibana. La gestion des alertes se fait également via Kibana. Il est possible de paramétrer des alertes classiques, qui se déclenchent suivant des règles précises. Et il est également possible de paramétrer des alertes suivant un algorithme d'apprentissage automatique, qui détectera des événements inhabituels.
Popularité	La suite Elastic est très certainement la plus populaire actuellement. Elle bénéficie d'une grande communauté active. Au niveau de la tendance, on peut voir une grande croissance entre les années 2016 et 2019. Ces derniers mois, cela semble se stabiliser.
Coûts	La suite Elastic propose différents abonnements. Il y a une offre gratuite, mais celle-ci ne contient pas de gestion d'alerte et de création de rapport. Les prix pour les offres payantes ne sont pas publics. Il faut contacter Elastic et les prix varient en fonction de la taille du système à implémenter.

TABLE 4 – Résumé théorique de Elastic Stack

3.3.2. Graylog

Graylog un outil de gestion de logs. Il dispose de deux versions : Open Source et Enterprise. La table 5 montre les fonctionnalités du systèmes Graylog.

Graylog	
Collecte	La collecte des logs se fait en approche minimaliste. Graylog possède un outil appelé « Sidecar »qui permet de gérer plusieurs type d'agent, y compris l'outil Beat de la suite Elastic.
Agrégation centralisée	Graylog permet de gérer « d'énormes »jeux de donnée et de les traiter selon des règles définies par l'utilisateur. En plus des règles d'agrégation classiques, comme l'origine géographique, le niveau d'alerte, etc., Graylog permet de faire des listes noires de logs.
Stockage et rétention	Le stockage se fait avec MongoDB et Elasticsearch. Graylog offre une solution (Graylog Archive) de rétention des données, disponible avec la version Enterprise, qui peut être paramétrée.
Rotation	La rotation se fait avec Graylog Archive.
Analyse	Graylog utilisant Elasticsearch, il dispose de ses requêtes poussées permettant de faire des recherches avancées. Graylog possède également son langage de requête, basé sur Apache Lucene.
Rapport	La création de rapport est une fonctionnalité de Graylog Enterprise. Il est possible de les configurer depuis l'interface de Graylog.
Visionnage et alertes	La visualisation des données en temps réels se fait avec l'interface de Graylog. La gestion des alertes est incluse dans le Graylog de base. Elle permet de définir des alertes selon des règles. Graylog possède également un « Store »proposant, entre autre, des fonctionnalités liées aux alertes, développées par la communauté.
Popularité	Graylog n'est pas arrivé très haut de manière générale dans les tops, mais il est en revanche souvent cité. La courbe de tendance de Graylog est en croissance régulière depuis 2008.
Coûts	Graylog propose deux versions : « Open Source »et « Enterprise ». La première est gratuite mais propose quelques fonctionnalités en moins, comme les rapports programmés ou le support technique. Les coûts de la version « Enterprise »ne sont pas disponible, il faut contacter Graylog. À noter que la version « Enterprise »est gratuite jusqu'à une utilisation de 5 GB par jour.

TABLE 5 – Résumé théorique de Graylog

3.3.3. SolarWinds Loggly

SolarWinds Loggly un outil de gestion de logs SaaS (Software-as-a-Service, dans le cloud). Il dispose de plusieurs versions, dont une gratuite. La table 6 montre les fonctionnalités et des informations sur les coûts de SolarWinds Loggly.

SolarWinds Loggly	
Collecte	L'envoi de données à Loggly est relativement simple. La seule contrainte est qu'il s'agisse de texte. Il n'y a pas besoin d'avoir d'agent (envoi de log via un endpoint), mais il est également possible d'en utiliser.
Agrégation centralisée	Loggly permet de parser les logs des formats les plus connus, comme les logs MySQL ou Java, mais ne possède pas d'outil permettant de réaliser un filtrage et une agrégation de logs personnalisés. Pour ce faire, il préconise d'utiliser un logiciel comme Logstash.
Stockage et rétention	La rétention des données est plutôt courte (7 jours dans la version gratuite), et ensuite, les logs peuvent être sauvegardés dans une instance S3 de AWS.
Rotation	La rétention se fait automatiquement après un certain nombre de jour.
Analyse	Loggly possède son propre langage de requête, qui est basé sur Apache Lucene. Il est également possible d'analyser les logs en temps réel via l'interface de Loggly.
Rapport	Il est possible de réaliser des rapport dans plusieurs format depuis l'interface graphique.
Visionnage et alertes	Il est possible de configurer des alertes selon des règles classiques, et également sur des événements inhabituels. Le visionnage des données en direct se fait via l'interface graphique de Loggly.
Popularité	Loggly était en croissance entre 2008 et 2016, mais depuis cette année-ci, sa courbe de tendance est en décroissance.
Coûts	Loggly propose une version gratuite et trois versions payantes. La version gratuite est limitée à un volume de 200 MB par jour, un seul utilisateur pouvant se connecter sur une instance, et ne contient pas certaines fonctionnalités comme la gestion des alertes. Les versions payantes varient entre 79 et 279 USD par mois, et proposent chacune quelques fonctionnalités en plus, et un volume de moins en moins limité.

TABLE 6 – Résumé théorique de SolarWinds Loggly

3.3.4. Splunk

Splunk est un outil de gestion de logs. Il est séparé en trois « parties », chacune étant responsable de plusieurs choses :

- Universal Forwarder

  Effectue la collecte et envoie les données à l'indexer.

- Indexer

  Effectue le stockage des logs.

- Search Head

  Permet de lire dans l'index.

Chacune de ces fonctions peut être transformé en cluster si de grand volume de données sont à traiter. Splunk peut être disponible en version « sur site »ou « cloud ». La table 7 montre les fonctionnalités et les coûts relatif à Splunk.

Splunk	
Collecte	La collecte des logs se fait en approche minimaliste. Ceci via les « Universal Forwarder »qui sont des agents à installer sur le système générant les logs. Il envoie ensuite les données vers l'indexer.
Agrégation centralisée	Pour Splunk, l'agrégation des logs ne se fait pas dans le pipeline lors de l'ingestion des logs, mais après coup. Cela se fait donc une fois que les logs ont été indexés. Il est possible de faire des filtres personnalisés ainsi que d'utiliser des filtres pour les formats courants, et ceci avec une fonctionnalité s'appelant « Field Extraction ».
Stockage et rétention	Le stockage des logs se fait avec l'indexer. Concernant la rétention des données, Splunk la gère avec des « buckets ». Concrètement, un bucket possède une durée qui détermine le temps qu'une donnée va passer dedans. P. ex., on peut avoir un bucket de 30 jours où les logs seront accessible et analysable librement. Puis, passé ces 30 jours, ils iront dans un autre bucket où ils seront compressé pour le stockage à long terme.
Rotation	Se fait via les buckets.
Analyse	L'analyse est possible via l'interface de Splunk. Celle-ci permet de trier et filtrer les logs selon de nombreux critères.
Rapport	La génération de rapport est possible depuis l'interface de Splunk.
Visionnage et alertes	Le visionnage et la gestion des alertes est également possible depuis l'interface de Splunk. Pour les alertes, elles sont définissables selon des règles classiques (p. ex. logs provenant d'une adresse IP particulière, etc.).
Popularité	D'après les courbes de Google Trends, le système Splunk est en constante croissance depuis 2010. Dans le classement des différents tops consultés, Splunk arrive en bonne position. Splunk bénéficie d'une documentation relativement grande et d'un forum.
Coûts	Les coûts sont en « infrastructure-based pricing »et ne sont donc pas fixe. Mais Splunk Enterprise commence à 150$ par mois. Splunk ne propose pas de version gratuite (mise à part l'essai de 60 jours).

TABLE 7 – Résumé théorique de Splunk

3.3.5. Popularité des systèmes

La comparaison de popularité des différents systèmes permet de se faire un idée sur les tendances actuelles et passées d'utilisation de ces systèmes. Elle a également pour but de pouvoir éventuellement anticiper les futures tendances. Ces données sont importantes lors du choix d'un logiciel, afin de pouvoir compter sur un support de la communauté lors des problèmes qui arriveront pendant le développement.

Conditions de tests

- Les courbes de tendances seront tirées de Google Trends [5].
- Les courbes porteront sur les 10 dernières années.

  Du 23 avril 2010 au 23 avril 2020
- Les courbes porteront sur les recherches dans tous les pays du monde.
- Il n'y aura pas d'autre restriction de recherche (catégorie, outil Google)

Résultats

La figure 2 montre les courbes de tendances des quatre systèmes. Les courbes de Loggly et Graylog étant particulièrement basses et donc difficilement comparable, la figure 3 permet de les comparer entre eux.

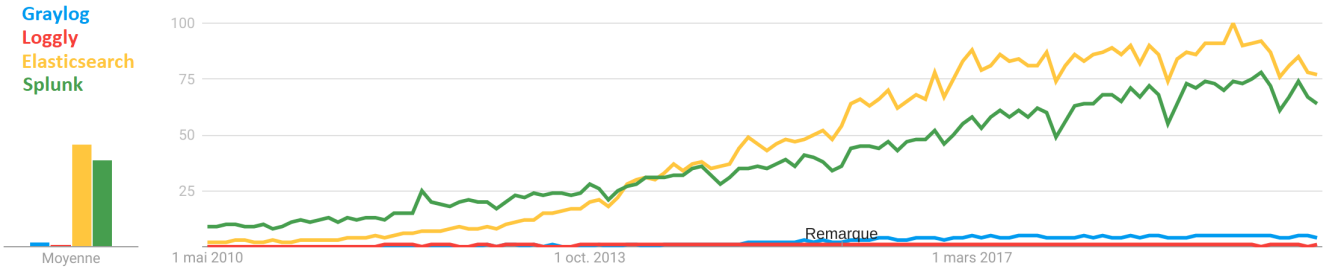

FIGURE 2 – Courbes Google Trends des quatre systèmes

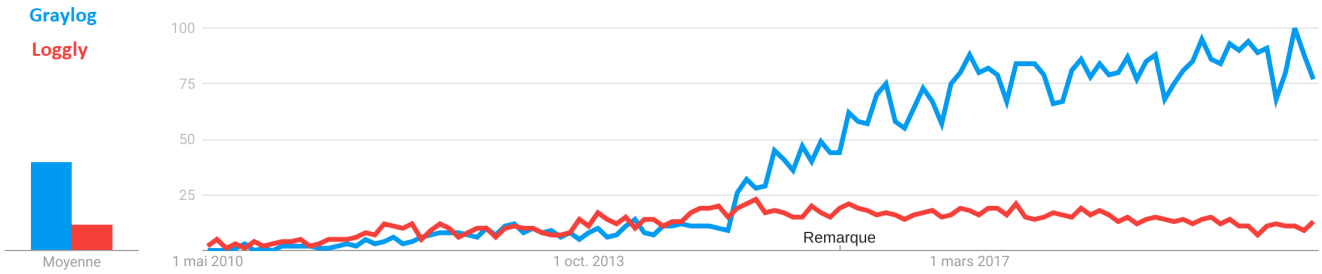

FIGURE 3 – Courbes Google Trends de Graylog et Loggly

La Suite Elastic contenant 4 logiciels séparés, le choix ici a été d'inclure uniquement « Elasticsearch », car c'est la plus connue des quatre applications, et que les résultats sont meilleurs qu'avec le terme « Elastic Stack »(il faut également savoir que la suite à changé de nom en 2016, passant de « ELK Stack »à « Elastic Stack », ce qui ne favorise pas la recherche dans Google Trends).

Conclusion
On distingue deux groupes dans cette analyse : Elasticsearch et Splunk d'un côté, Graylog et Loggly de l'autre. Le premier est très populaire et en croissance, alors que le deuxième n'est que peu tendance, et Loggly n'est pas en croissance. Au sein même du groupe Graylog-Loggly, on peut encore constater une grande différence, Graylog étant beaucoup plus recherché sur Google que Loggly. On peut donc affirmer que la Suite Elastic est le système le plus populaire, devant Splunk. Loin derrière vient Graylog et encore plus loin, on trouve Loggly.

3.4. Tests réels et prise en main des logiciels

Plusieurs tests ont été effectués afin de pouvoir effectuer une comparaison de ces systèmes. Comme la plupart de ces tests obligent une prise en main concrète des logiciels, ceux-ci ont également permis d'avoir une idée de la facilité d'utilisation, de la flexibilité ainsi que de l'utilisabilité (« user-friendliness ») des systèmes.

Voici les versions des logicielles utilisées pour effectuer ces tests :

- Elastic Stack

  Elasticsearch 7.6.2

  Filebeat 7.6.2

  Logstash 7.6.2

  Kibana 7.6.2

- Graylog

  Graylog 3.2.4-1 (Virtual Appliance)

  Filebeat 7.6.2

- Splunk

  Splunk 8.0.2.1

  Splunk Universal Forwarder 8.0.2.1

- Loggly

  Loggly Lite (Cloud)

3.4.1. Test de débit d'ingestion de log

Le premier test de cette comparaison consiste en un test de débit. Cela permet d'avoir une première idée de la performance des applications.

Conditions de tests

- Chargement d'un fichier de logs.

  Le fichier contient 98'305 logs identiques.

  Log : 2020-02-27 09 :06 :24.596 INFO o.s.s.c.ThreadPoolTaskScheduler -> Shutting down ExecutorService 'taskScheduler'

- On effectue un test d'ingestion simple (on stocke le log brut).
- On effectue un test d'ingestion avec filtre (on parse les différentes parties du log).

  Date et heure : 2020-02-27 09 :06 :24.596

  Niveau du log : INFO

  Classe Java : o.s.s.c.ThreadPoolTaskScheduler

  Message : Shutting down ExecutorService 'taskScheduler'

- Le résultat est un débit exprimé en EPS (Event Per Second), qui correspond aux nombres de logs que le système aura pu ingéré en une seconde.

La table 8 montre **les résultats** des tests d'ingestion.

	Elastic Stack	Graylog	Splunk	Loggly
Sans filtrage	4'898	14'044	-	46'295 (*)
Avec filtrage	9'181	7'868	2'628	-

TABLE 8 – Résultats des tests de débit d'ingestion

(*) Loggly n'acceptant que des fichiers de taille inférieure à 5 MB, le fichier à été réduit à 3'999 logs.

Splunk favorise l'extraction d'informations (parsing) après avoir stocké les logs dans le système. Il est donc compliqué d'ajouter un filtrage avant le stockage et ça n'a donc pas été réalisé. Par contre, Splunk effectue automatiquement une extraction de la date des logs. Son débit à donc été classé dans la catégorie « avec filtrage ». Loggly ayant été écarté assez tôt de l'évaluation, la partie « avec filtrage »n'a pas été testée.

Conclusion
Sans filtrage, on constate que Graylog possède un débit supérieur à Elastic Stack. Loggly n'acceptant pas des fichiers de grande taille, il est difficilement comparable. Avec filtrage, étonnemment, la Suite Elastic gagne en débit. Elle possède donc un meilleur débit que Graylog et Splunk. Après ce test, outre les résultats comptables, il se dessine surtout une tendance vers Graylog et Elastic Stack, car ces deux systèmes sont plus « ouvert »que les autres. Il n'y a pas eu de difficultés particulières lors de l'utilisation, pas de contraintes comme la taille du fichier ou encore le moment du filtrage au sein du pipeline.

3.4.2. Test d'ingestion d'une grande quantité de logs

Ce test a pour but de mettre en évidence les limites de certains systèmes lors de l'ingestion de gros fichier de logs. Il permet également d'avoir une brève vue sur la constance du débit mesuré lors du test précédent.

Conditions de tests

- Chargement d'un fichier de logs.

  Le fichier contient 999'999 logs identiques.

  Log : 2020-02-27 09 :06 :24.596 INFO o.s.s.c.ThreadPoolTaskScheduler -> Shutting down ExecutorService 'taskScheduler'
- On n'effectue aucun traitement sur le log.
- Le résultat est soit une limite supérieure, soit « > 1'000'000 »

La table 9 montre **les résultats** du test d'ingestion d'une grande quantité de logs.

Elastic Stack	Graylog	Splunk	Loggly
> 1'000'000	> 1'000'000	> 1'000'000	~4'000 (5 MB)

TABLE 9 – Résultats des tests d'ingestion de grande quantité de logs

Conclusion
Ce test permet simplement de montrer que les systèmes travaillant avec un agent, soit selon la méthode PUSH, n'ont pas de problème de taille de fichier, car l'agent lit et envoie les logs au fur et à mesure. Loggly est le seul ici utilisant la méthode PULL. Au niveau des débits, ceux d'Elastic Stack et de Graylog ont été divisés par 2, alors que celui de Splunk est resté stable. Pour Loggly, il n'y a pas eu de différence, étant donné qu'il est limité en taille de fichier.

3.4.3. Test de consommation CPU

Ce test-ci a pour but de pouvoir comparer les systèmes de gestion de logs Elastic Stack et Graylog en terme de performance.

Conditions de test

- Chargement du même fichier de log que pour le test de débit.
- L'utilisation du CPU sera réduite au minimum hors système à tester.
- Ordinateur de test : ASUS UX360UAK
- Processeur : Intel Core i7-7500U CPU @ 2.70GHz x 4
- Lors de la réception, l'agent Filebeat se situera sur une autre machine.

Résultats

La figure 4 montre la consommation CPU et l'historique du trafic réseau lors de la réception des logs pour le système Elastic Stack. La figure 5 montre les mêmes informations, mais pour le système Graylog. Enfin, la figure 6 montre ces informations-ci avec l'agent Filebeat.

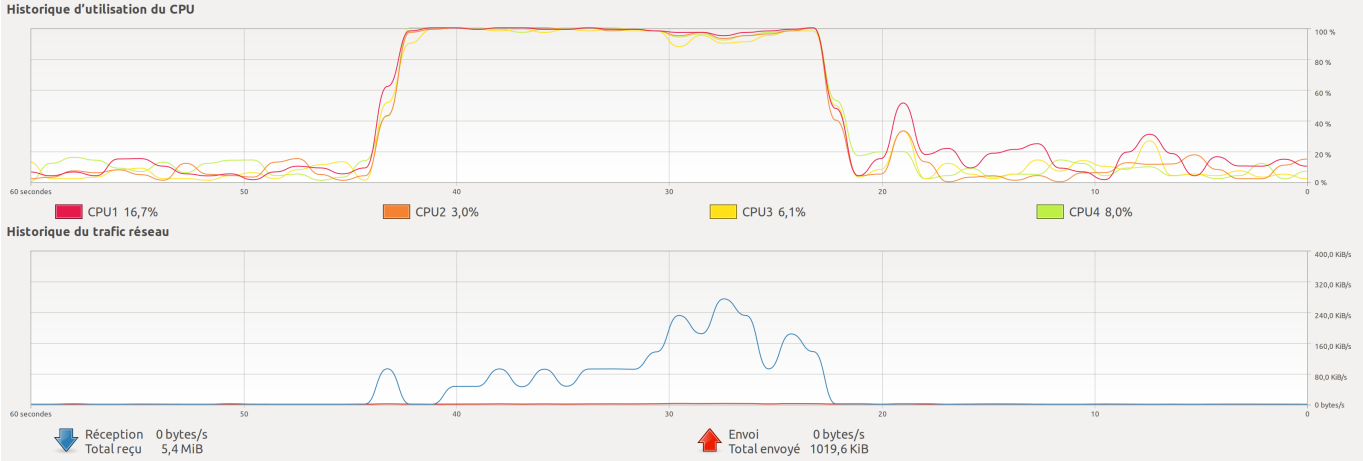

FIGURE 4 – Consommation CPU Elastic Stack

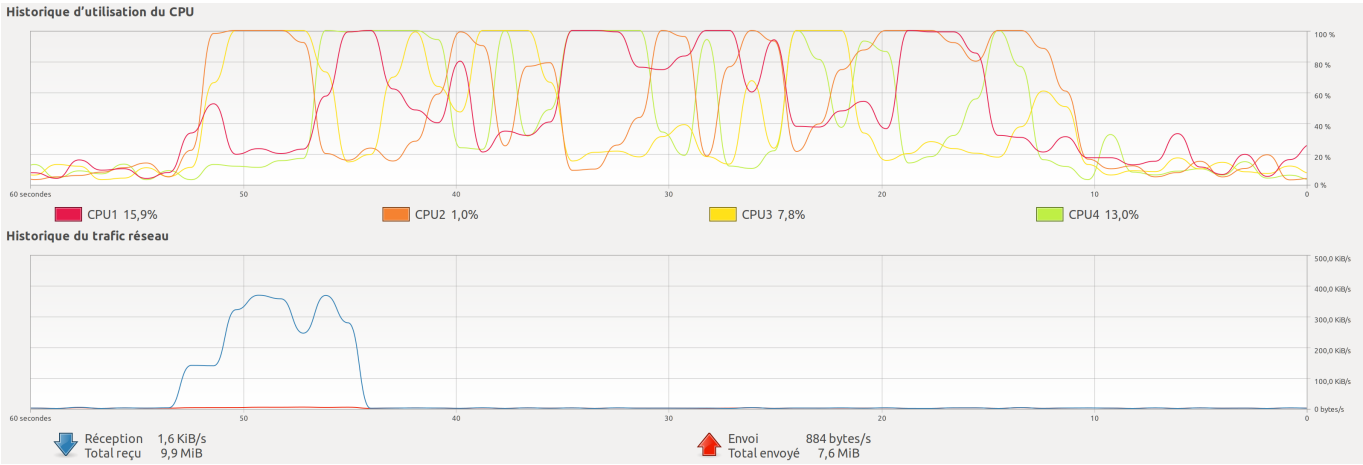

FIGURE 5 – Consommation CPU Graylog

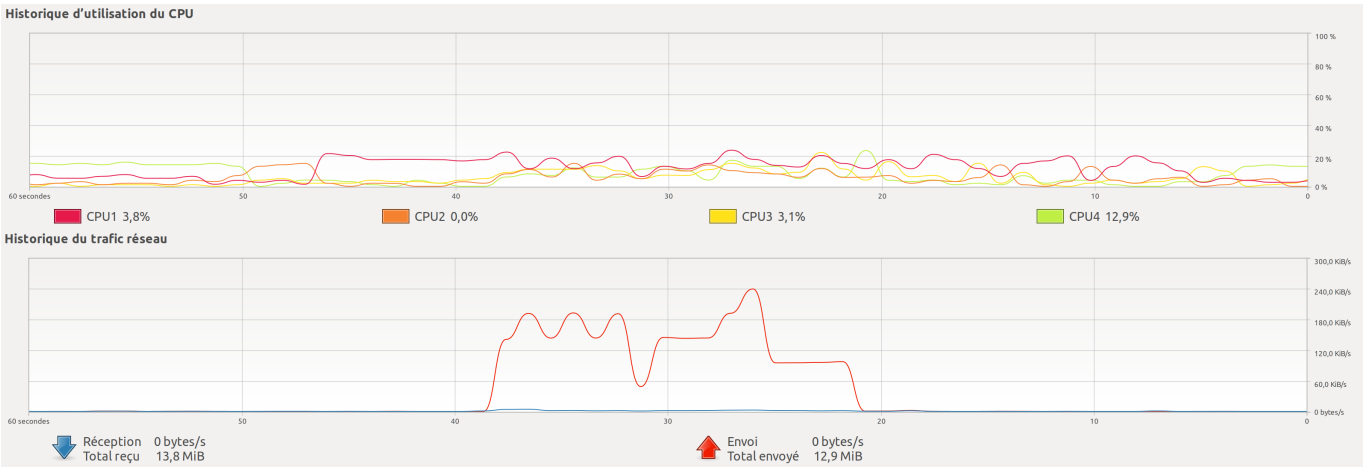

FIGURE 6 – Consommation CPU Filebeat

Conclusion
Premièrement, on constate que l'agent Filebeat est un agent très léger puisque son action ne se fait quasiment pas ressentir sur le taux d'utilisation du CPU. Pour ce qui est du taux d'utilisation CPU par la Suite Elastic, on remarque que lors de la réception et le stockage des logs, tous les CPUs tournaient à 100% de leur capacité, et la charge dure environ 25 secondes. Pour Graylog, les CPUs ne sont pas autant sollicité, ou du moins pas tous en même temps. Ils varient entre 20% et 100% et la distribution est plutôt égale entre les CPUs. On peut donc estimer un taux d'utilisation moyen à 60-70%. La charge dure cette fois-ci environ 45 sec. À noter que Graylog ingère les logs en env. 7 secondes, puis travaille encore pendant env. 35 secondes, alors que la Suite Elastic fait tout en même temps. On peut constater ceci grâce à la courbe de l'historique du trafic réseau.

3.5. Conclusion de l'évaluation

3.5.1. Analyse théorique des systèmes

Au niveau fonctionnel, les quatre systèmes de gestion de logs ne se différencient pas énormément. Ils suivent à peu près tous le même schéma et permettent tous de faire les mêmes actions (reporting, alerting, analyse, etc.). Loggly est légèrement différent dans le sens où son approche de collecte de log par défaut est maximaliste (approche PULL). Il met en effet un endpoint à disposition, sur lequel on peut envoyer des données. Il s'agit la d'une différence mineure car bien que les autres systèmes privilégient et facilitent l'approche minimaliste, il est également possible de configurer un endpoint semblable pour les systèmes Elastic Stack, Graylog et Splunk. Pour ce qui est du non-fonctionnel, il y a quelques points où les systèmes se démarquent. Premièrement, Loggly de SolarWinds est uniquement disponible dans une version Cloud. La Suite Elastic et Splunk sont eux disponibles tant dans une version Cloud que dans une version sur site. Graylog lui est uniquement disponible dans une version locale. Il y a également l'architecture des systèmes. La Suite Elastic est formée de 3 applications indépendantes, et en utilise une $4^{\text{ème}}$ pour l'agent Beat. Graylog est composé d'un système, et utilise des agents ne lui appartenant pas, comme Beat. Splunk suit la même architecture que Graylog, mais possède son propre agent, le Splunk Forwarder. Et Loggly est une infrastructure cloud sans agent. Reste la comparaison de la popularité des applications. Celle-ci a montré une nette avance pour Splunk et Elastic Stack, devant Graylog, puis Loggly qui est vraiment peu populaire.
Globalement, l'analyse théorique des systèmes de gestion de logs ne permet pas de faire un choix concret sur un système ou un autre pour la réalisation d'une solution destinée au système GridEye. Elle permettrait tout au plus de se détacher de Loggly, qui n'est vraiment pas recommandable du point de vue de sa courbe de tendance.

3.5.2. Analyse pratique des systèmes

Le but des tests réels était de pouvoir confronter les systèmes de gestion de log au niveau de leur performance, et également de réaliser une prise en main des différents logiciels, avec tout ce que cela implique (installation, configuration, etc.). Cette deuxième partie avait également pour but d'évaluer l'utilisabilité des divers systèmes. Au niveau des performances, on a pu voir que les systèmes Graylog et Elastic Stack se distinguaient légèrement avec un meilleur débit d'ingestion. Que tous les systèmes, excepté Loggly, pouvaient, grâce à leur approche minimaliste, recevoir des grandes quantités de logs.
À ce point-ci de l'évaluation, un premier bilan de l'« user-friendliness »des systèmes à également pu être tiré :

- SolarWinds Loggly est relativement simple d'utilisation. Sa caractéristique Cloud facilite également l'installation (il n'y en a pas, il suffit de se créer un compte et nous recevons un lien vers notre version d'essai).
- Graylog est un système ayant une interface plutôt complète et efficace. L'interface graphique permet directement de configurer le système de manière simple (elle permet d'ajouter des règles d'extraction d'informations dans les logs, de configurer les entrées ouvert aux agents, etc.).

- Elastic Stack est une « suite »de logiciels, ce qui la rend très ouverte. On peut en effet choisir d'utiliser ces software de base, mais également utiliser d'autres applications. Par exemple, on peu prendre un agent de la famille Beat, mais on pourrait aussi choisir un agent Prometheus. Chaque logiciel est configurable via des fichiers de configuration. On peut également choisir d'utiliser les 4 logiciels, ou pas. Par exemple enlever Logstash si l'on a pas besoin de ces filtres. Cette ouverture le rend agréable à utiliser.
- Splunk est comparable à Graylog et Elastic Stack dans tous les tests, mais l'expérience utilisateur est de son côté moins bonne. Le système est plus une sorte de « boîte noire »dans laquelle il n'est pas possible de tout configurer. L'extraction des informations des logs n'a par exemple pas pu avoir lieu avant l'indexage, car le système est prévu pour le faire après.

Avec ces premiers tests, nous avons déjà pu écarter la solution SolarWinds Loggly, qui ne correspondait pas aux attentes et dont les contraintes liées au Cloud ne sont pas souhaitées. Splunk n'est également pas recommandé à cause de son manque d'ouverture. Ensuite, il y a encore eu le test d'utilisation du CPU entre la Suite Elastic et Graylog. Avec celui-ci, on pourrait donner un très léger avantage à Graylog.

3.5.3. Recommandation

En conclusion, il est difficile de faire un choix entre les deux systèmes Graylog et la Suite Elastic. Mais de part sa très grande popularité, amenant son lot de réponse dans les forums et autres articles permettant d'accélérer le développement d'un logiciel, ainsi que de son caractère très ouvert lié à son « architecture suite », la **Suite Elastic** est le meilleur choix dans l'optique de mise en place d'une solution de gestion de logs pour l'infrastructure GridEye.

4. Implémentation du cas d'utilisation

Un cas d'utilisation à implémenter à été fourni par DEPsys et demande les éléments suivants :

Le système générant les logs sera composé d'une base de donnée PostgreSQL et d'une application Java simulant différents comportements, comme une surcharge du CPU, de la RAM ou autres problèmes.

Le système Elastic Stack devra mesurer plusieurs métriques :

- Redémarrage de conteneur.
- Taux d'utilisation de la mémoire.
- Occupation de la base de données (PostgreSQL).
- Taux d'utilisation du CPU.
- Les crashs Java
- Et éventuellement d'autres métriques comme les logs de l'application Java.

Plusieurs instances peuvent être lancées en même temps, ce qui permet de simuler au mieux la réalité (il y a plusieurs « field device »).

Il doit être possible de savoir si le système est fonctionnel, et à quel point.

Pour répondre à toutes ces demandes, on va commencer par installer et configurer tous les outils de la Suite Elastic qui seront utilisés, et ce avec Docker. Ensuite, nous allons créer des visualisations avec Kibana. Elles permettront de connaître l'état du système en temps réel, et de retrouver la cause des problèmes, s'il y en a.

4.1. La Suite Elastic avec Docker

Afin de faciliter les étapes d'installation, de configuration et de lancement de la Suite Elastic, une version « dockerisée »des différents logiciels est une solution idéale. Elle permet également d'uniformiser ces étapes-ci. En effet, l'installation des applications de la Suite Elastic sans conteneur fonctionnera de la même manière sur des systèmes Ubuntu, voire Linux, mais sera différente sur un système d'exploitation Microsoft Windows par exemple. L'utilisation de conteneurs comble ces problèmes.

L'installation à été séparée en deux, car deux machines sont nécessaires pour simuler l'interface GridEye : une machine serveur, recevant les logs, et une machine client, envoyant les logs.

4.1.1. Machine hôte de « ELK »

Par « ELK », qui est l'ancien nom de la Suite Elastic, on parle des trois outils Logstash, Elasticsearch et Kibana. C'est la machine qui ne sera pas sur le terrain et qui permettra l'agrégation, le stockage et l'analyse des logs, entre autres. La figure 7 montre un schéma de la topologie Docker prévue permettant de faire fonctionner ensemble les différents outils de la Suite Elastic.

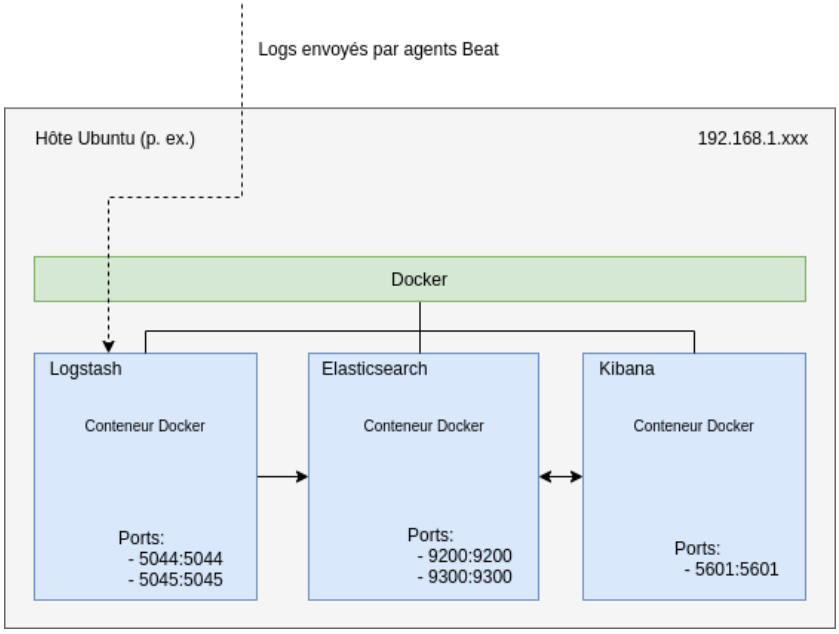

FIGURE 7 – Architecture de l'implémentation du use-case - côté « ELK »

Comme il y a plusieurs conteneur Docker devant être créés, nous allons utiliser l'outil `docker-compose` qui permet de gérer et d'orchestrer de multiple conteneurs. Nous aurons trois images à aller télécharger : Logstash, Elasticsearch et Kibana. L'agent Beat se situant sur un ordinateur distant, il n'est pas inclus dans cette topologie. La figure 8 montre l'arborescence des dossiers et fichiers nécessaires.

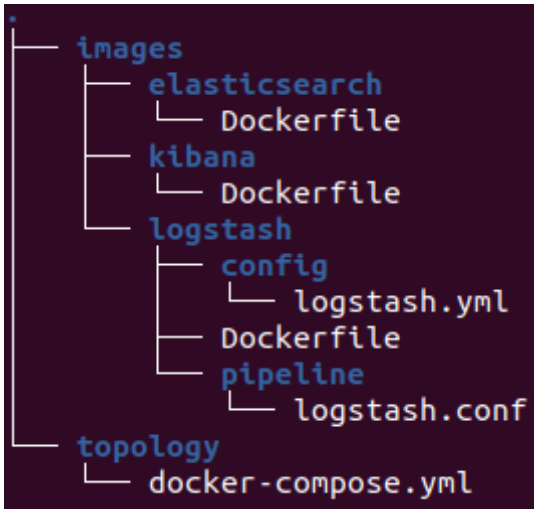

FIGURE 8 – Arborescence de la topologie Docker - côté ELK

Les `Dockerfile` des images Elasticsearch et Kibana sont très simples, ils vont simplement télécharger les images correspondantes depuis le `Elastic Docker registry`.
`Dockerfile` d'Elasticsearch :

```
1 FROM docker.elastic.co/elasticsearch/elasticsearch:7.6.2
```

`Dockerfile` de Kibana :

```
1 FROM docker.elastic.co/kibana/kibana:7.6.2
```

Pour l'image de Logstash, il y aura deux fichiers de configurations. Un pour le « pipeline », qui contiendra les configurations d'entrées, filtres et sorties, et un pour les paramètres de l'outil.
Fichier de configuration des paramètres (`logstash.yml`) :

```
http.host: "0.0.0.0"
xpack.monitoring.elasticsearch.hosts: [ "http://elasticsearch:9200" ]
```

La configuration du pipeline se fait dans un fichier `.conf` nommé `logstash.conf` et qui ira se copier à l'adresse `/usr/share/logstash/pipeline/` du système de fichier du conteneur. Le fichier peut avoir un nom différent, tant qu'il porte l'extension `.conf`.

Le fichier de configuration de Logstash est divisé en trois parties :

1. input

   Cette première partie permet de configurer l'arrivée des logs. Il est possible d'y configurer un agent Beat, ce qui nous concernera ici, mais également la lecture directe d'un fichier, ou de beaucoup d'autres sources, comme Twitter, JDBC, etc.

2. filter

   La partie filtre permet quant à elle de parser les logs. Il existe là aussi beaucoup de filtres différents. Nous utiliserons une structure `grok` qui permet de parser des événements non structurés.

3. output

   La sortie sert à choisir où Logstash enverra ses logs traités. En général, il s'agit d'Elasticsearch.

Pour la configuration de l'entrée, nous allons simplement écouter deux agents Beat, chacun sur un port donné. Nous allons également ajouté un tag `filebeat` qui sera utilisé pour la partie « filtre ».

```
input {
    beats {
        port => 5044
        tags => ["filebeat"]
    }

    beats {
        port => 5045
    }
}
```

En ce qui concerne le filtre, nous allons parser les logs envoyés par le logiciel de simulation de GridEye. Le format des logs est défini par la ligne suivante :

```
%d{yyyy-MM-dd HH:mm:ss.SSS} %-5level %logger{100} -> %msg%n
```

Le site `grokconstructor` [10] permet de créer des `grok pattern` de façon incrémentale. Il suffit de lui donner quelques lignes de logs générés par notre application et de le configurer. Il va ensuite nous proposer des motifs qu'il détecte dans les lignes de logs données. Voici le résultats après lui avoir donné quelques lignes générés par le simulateur :

```
\A%{TIMESTAMP_ISO8601}%{SPACE}%{LOGLEVEL}%{SPACE}%{JAVACLASS} -> %{JAVALOGMESSAGE}
```

À savoir que le tout premier champ, qui a été défini comme un `TIMESTAMP_ISO8601`, n'en est pas vraiment un [6]. Mais c'est le motif qui correspond le mieux à notre format de date-heure. Il reste à définir des labels décrivant les différents motifs. Ces labels seront utilisés par Kibana.

```
\A%{TIMESTAMP_ISO8601:date_hour}%{SPACE}%{LOGLEVEL:log_level}%{SPACE}%{JAVACLASS:class} ->
    %{JAVALOGMESSAGE:message}
```

Comme ce `grok pattern` permet de parser uniquement nos logs applicatifs, il faut ajouter une condition, qui indique que si les logs viennent d'ailleurs (p. ex. de la base de données PostgreSQL), ils ne doivent pas être parsés par ce filtre. Nous allons utiliser le tag `filebeat`.

```
filter {
    if ("filebeat" in [tags]) {
  grok {
      match => {"message" => "%{TIMESTAMP_ISO8601:date_hour}%{SPACE}%{LOGLEVEL:log_level
    }%{SPACE}%{JAVACLASS:class} -> %{JAVALOGMESSAGE:log_message}"}
  }
}
```

Pour finir, la partie sortie défini uniquement l'adresse et le port de notre instance Elasticsearch.

```
output {
    elasticsearch {
        hosts => ["elasticsearch:9200"]
    }
}
```

Le fichier complet `logstash.conf` est disponible dans l'annexe A.1.

Le `Dockerfile` de Logstash télécharge l'image, puis ajoute les fichiers de configuration au système de fichier du conteneur.

```
FROM docker.elastic.co/logstash/logstash:7.6.2
RUN rm -f /usr/share/logstash/pipeline/logstash.conf
ADD pipeline/ /usr/share/logstash/pipeline/
ADD config/ /usr/share/logstash/config/
```

Pour finir, le fichier `docker-compose.yml` organise toute la topologie. Il lancera d'abord Elasticsearch, puis Kibana, et enfin Logstash. Les ports par défaut des différents logiciels seront exposés, afin d'être accessible depuis l'extérieur. Ceci est nécessaire pour permettre la communication entre les différents conteneurs.

```
version: '3'

services:

  elasticsearch:
    container_name: elsearch
    build: ../images/elasticsearch
    environment:
            discovery.type: "single-node"
    ports:
      - 9200:9200
      - 9300:9300

  kibana:
    container_name: kib
    build: ../images/kibana
    links:
      - "elasticsearch"
    ports:
     - 5601:5601
    depends_on:
      - elasticsearch

  logstash:
    container_name: logst
    build: ../images/logstash
    ports:
      - 5044:5044
    depends_on:
      - elasticsearch
      - kibana
```

Un répertoire GitHub [7] contient toute l'architecture décrite.

Pour lancer le déploiement de la Suite Elastic, il suffit de lancer la commande `docker-compose up --build -d` à l'endroit où se situe le fichier `docker-compose.yml`. Dans ce cas-ci, le dossier `topology`. Pour tester si tout le déploiement s'est bien déroulé, une vérification d'Elasticsearch et de Kibana s'impose.

Premièrement, avec un navigateur, la visite de l'adresse `http://localhost:9200/_cat/indices` permet de constater le fonctionnement d'Elasticsearch et de Logstash. En effet, si tout se passe bien, on devrait voir les différents indices contenu dans la base de données, dont un s'appelant `logstash...`. La figure 9 montre cette configuration réussie.

```
green  open .kibana_task_manager_1     jxlhPGoPQCmLbgD3fOHxvA 1 0 0 0  230b  230b
green  open .apm-agent-configuration   koTkBCISQMmF3dqPn1iMew 1 0 0 0  230b  230b
green  open ilm-history-1-000001       RCUFq0GrTW2QdtbQE5k8Rw 1 0 3 0 7.9kb 7.9kb
yellow open logstash-2020.06.17-000001 jNW-irlITVyl9-jIt4z__g 1 1 0 0  230b  230b
green  open .kibana_1                  J-NO7wASRV-JStVtpsvWvw 1 0 1 0  230b  230b
```

FIGURE 9 – Indices contenus dans la base de données Elasticsearch, avec Logstash

Deuxièmement, toujours dans un navigateur, à l'adresse `localhost:5601`, il devrait y avoir l'application Kibana qui est lancée. La figure 10 montre ceci.

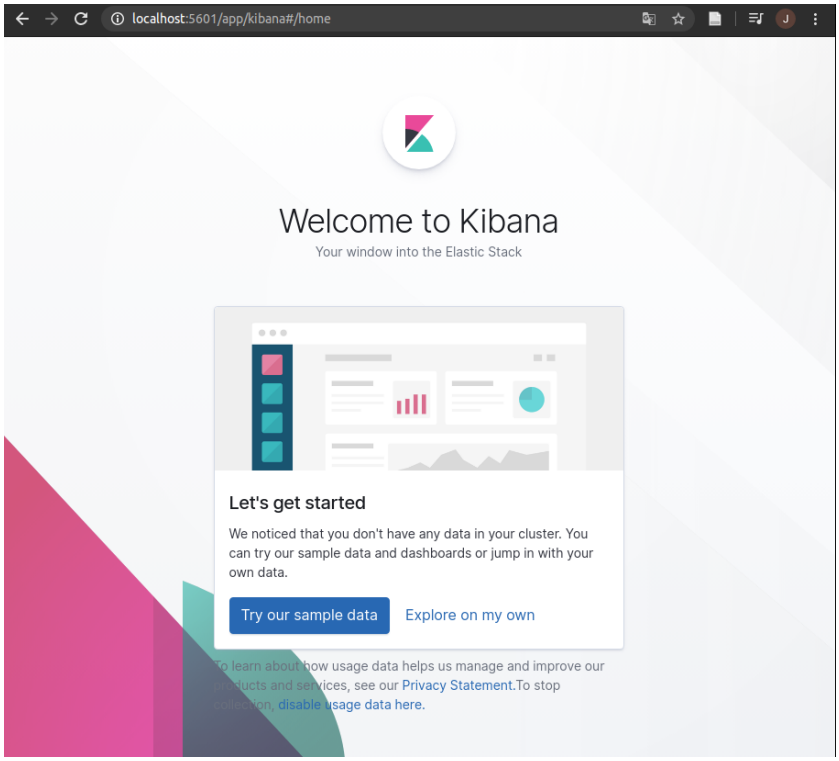

FIGURE 10 – Application Kibana lancée

4.1.2. Machine « terrain »

La machine sur le terrain contiendra les agents Beat nécessaires, ainsi que les programmes et applications générant les logs. Il y aura également une base de données contenant des données arbitraires. La figure 11 montre l'architecture définie pour l'implémentation du use-case sur la machine « terrain ».

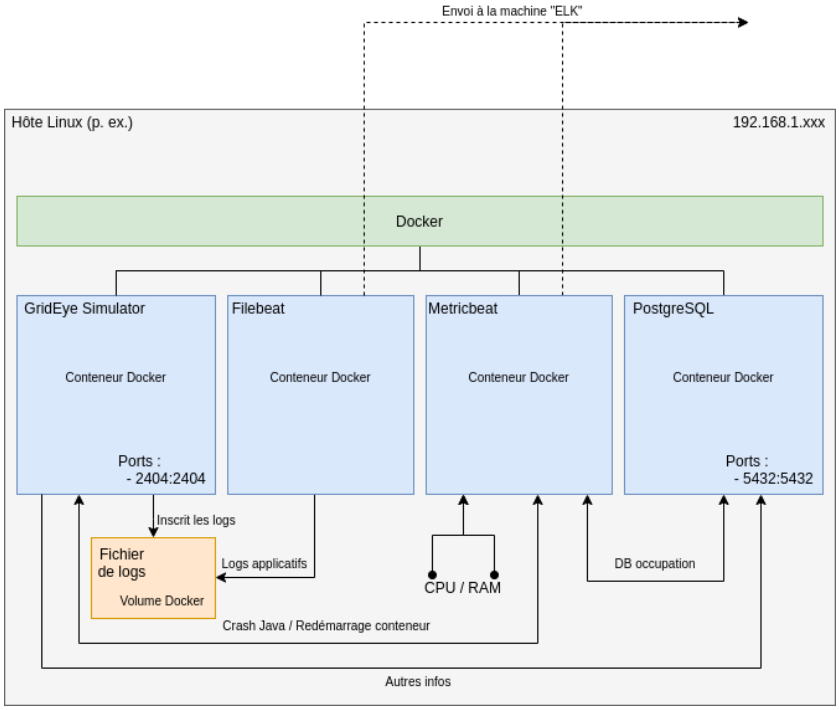

FIGURE 11 – Architecture de l'implémentation du use-case - côté terrain

Il y aura donc le simulateur GridEye générant les logs, un agent Filebeat qui va lire les logs applicatifs et les envoyer à la Suite Elastic (les logs seront partagés entre les deux conteneur via un Volume Docker, qui sera lié à un chemin sur la machine hôte), un agent Metricbeat qui va surveiller le système hôte ainsi que la base de données. L'agent Metricbeat contiendra également le module de monitoring de Docker, qui nécessite un Volume Docker permettant l'accès au « Daemon Docker ». Il enverra finalement les logs vers la machine hôte ELK. Pour finir, il y aura la base de donnée, qui sera un système de gestion de bases de données relationnelle (SGBDR) PostgreSQL. Comme pour la machine contenant les outils « ELK », nous allons travailler avec un `docker-compose`, qui permettra d'orchestrer tous les conteneurs. La figure 12 montre l'arborescence des dossiers et fichiers contenant les images Docker ainsi que le fichier `docker-compose.yml`. Cette arborescence reprend les mêmes bases que celle du côté ELK.

FIGURE 12 – Arborescence de la topologie Docker - côté terrain

Les images `filebeat` et `metricbeat` ont la même structure. Un `Dockerfile` qui télécharge l'image correspondante depuis le `Elastic Docker registry`, puis qui copie le fichier de configuration de l'agent, respectivement `filebeat.yml`, et `metricbeat.yml`.

`Dockerfile` de filebeat

```
FROM docker.elastic.co/beats/filebeat:7.6.2
COPY config/filebeat.yml /usr/share/filebeat/filebeat.yml
USER root
RUN chown root:filebeat /usr/share/filebeat/filebeat.yml
USER filebeat
```

`Dockerfile` de metricbeat

```
FROM docker.elastic.co/beats/metricbeat:7.6.2
COPY config/metricbeat.yml /usr/share/metricbeat/metricbeat.yml
USER root
RUN chown root:metricbeat /usr/share/metricbeat/metricbeat.yml
```

La configuration de Filebeat se fait dans un fichier `YAML`, qui se nomme `filebeat.yml`, et qui sera copié dans le répertoire `/usr/share/filebeat/` du conteneur. Il y a un grand nombre de paramètres de configuration disponibles. Dans le cas de notre utilisation, il n'y a que deux paramètres à configurer : l'entrée à lire ainsi que la sortie. Pour l'entrée, il suffit de faire pointer le paramètre vers le fichier dans lequel le simulateur GridEye écrit ses logs.

```
filebeat.inputs:
  - type: log
    enabled: true
    paths:
            - /usr/share/filebeat/logs/GridEye_sim.log
```

Pour la sortie, on va informer l'adresse de l'instance de Logstash configuré plus tôt.

```
output.logstash:
  hosts: ["<IP>:5044"]
```

Il faut remplacer `<IP>` par l'adresse IP de la machine hébergeant Logstash.

Pour Metricbeat, la configuration est similaire. Elle se fait également dans un fichier `YAML`, qui se nomme `metricbeat.yml`, et qui sera copié dans le répertoire `/usr/share/metricbeat/` du conteneur.

Il y a deux paramètres à configurer : les modules de métriques à surveiller, ainsi que la sortie.

Pour les modules de métriques, nous allons utiliser `system`, qui permettra d'obtenir les informations sur la mémoire et le CPU. Le module `postgresql` nous permettra lui d'obtenir les métriques de la base de données. Et pour finir, le module `docker` enverra lui les métriques concernant les redémarrage de conteneur, et les crashs Java.

La configuration de sortie est la même que pour filebeat, mais avec le port 5045.

Les fichiers de configuration des deux agents sont disponibles aux annexes A.3 (filebeat) et A.4 (metricbeat).

L'image du simulateur de la plateforme GridEye contient elle un `Dockerfile` qui va lancer une application Java depuis un fichier `jar`. Cette manière de lancer une application Java dockerisée est la plus simple pour ce cas présent (le fichier compilé ne change jamais car il n'y a pas de mise à jour du programme). Il y a également un dossiers `logs` qui contiendra le fichier de log et sera donc la base du Volume Docker partagé avec l'agent filebeat. Le code, ainsi qu'une description de l'application sont disponible dans un répertoire GitHub [8].

`Dockerfile` du simulateur GridEye

```
FROM openjdk:8-jdk-alpine

COPY app/grideye-service-log-generator-1.0.0-SNAPSHOT.jar grideye_sim.jar
CMD java -jar grideye_sim.jar
```

Finalement, le dossier de l'image de la base de données PostgreSQL est également simple. On va simplement télécharger l'image, puis copier le fichier `sql` qui permettra de peupler la base de données avec des informations arbitraires (en l'occurrence, des informations sur des pays et des villes).

`Dockerfile` de la base de données

```
FROM postgres:12.1
COPY data/world.sql /docker-entrypoint-initdb.d/10-init.sql
```

Pour ce qui est du docker-compose, disponible à l'annexe A.5, il contient les quatre services correspondant aux quatre conteneurs Docker. Les services `filebeat` et `grideye_simulator` partagent un Volume Docker, alors que `metricbeat` possède également un Volume Docker qui pointe sur le fichier `docker.sock`, permettant de gérer Docker depuis un conteneur.

Afin de tester l'installation de la partie terrain, il suffit de lancer la commande `docker-compose up --build -d` à l'endroit où se situe le fichier `docker-compose.yml`. Dans ce cas-ci, le dossier `topology`. Si tout s'est bien installé, des logs devraient arriver rapidement à l'adresse indiqué dans les fichiers de configurations, si la Suite Elastic y est bien lancée. On y retrouvera des logs applicatifs via filebeat, ainsi que des logs contenant les métriques via metricbeat.

Pour voir ces logs, on va utiliser Kibana. Il faut tout d'abord créer un `Index Pattern`. Cet outil est disponible dans `Management` → `Kibana` → `Index Patterns` → `Create index pattern`. Le nom du motif doit commencer par les mêmes lettres que l'index stocké dans Elasticsearch. Dans notre cas, on peut donc l'appeler simplement `logstash`. À l'étape 2, il est utile de choisir un `timefilter`, comme `@timestamp`, pour pouvoir effectuer des recherches et des graphes avec la notion de temps.

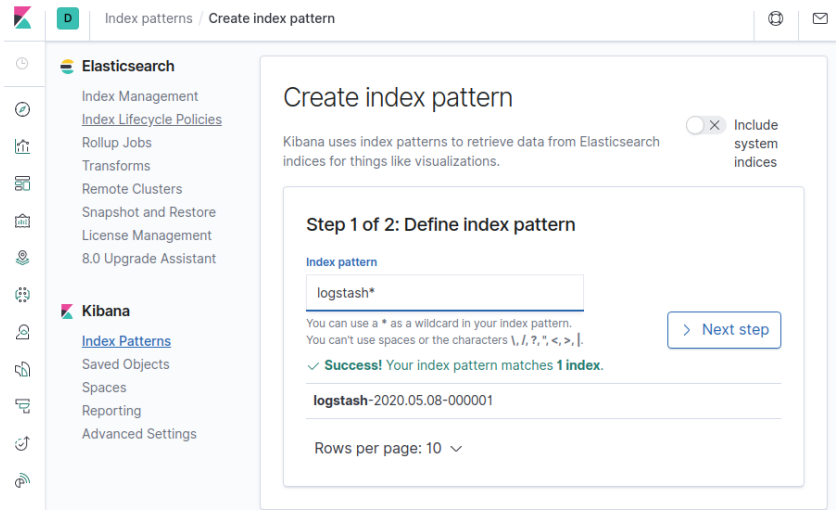

FIGURE 13 – Création de l'index pattern dans Kibana

Une fois créé, on peut voir les logs dans l'onglet `Discover` de Kibana. Il est également possible d'afficher uniquement les champs qui nous intéressent, comme ceux créés dans le filtre.

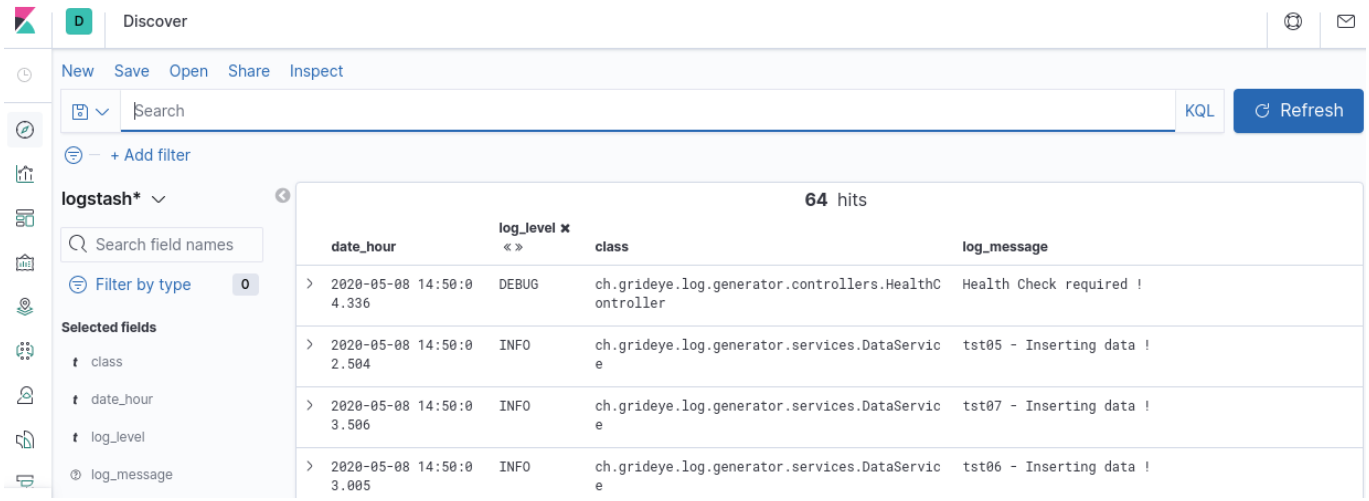

FIGURE 14 – Affichage des logs dans Kibana

4.2. Visualisation avec Kibana

Une fois tous les outils installés, il ne reste plus qu'à créer des visualisations permettant de suivre l'état du système en temps réel, ou encore de retrouver l'origine d'un problème sur le système.

Pour ce faire, nous allons créer plusieurs visualisations, elles-même contenues dans des tableaux de bord.

Un premier tableau de bord montrera tous les problèmes de l'infrastructure. Ainsi, un opérateur pourra voir immédiatement s'il y a eu un problème dernièrement, ou au contraire si le tableau est vide.

Pour ce cas d'utilisation, il y aura ensuite un autre tableau de bord mettant en relation les taux d'utilisations CPU de chaque conteneur docker. Il permettra, au cas où l'opérateur remarque une surcharge CPU sur le tableau de bord des problèmes, de déterminer quelle application pose problème.

4.2.1. Tableau de bord des problèmes

Pour ce tableau de bord, nous avons inséré quatre informations susceptible d'être considéré comme des « problèmes » :

- Les logs de crash du simulateur GridEye
- Les redémarrages de conteneur Docker
- Les sur-exploitation du processeur
- Les sur-exploitation de la mémoire

La figure 15 montre ce tableau de bord. Les trois premières visualisations, concernant les crashs du simulateur ainsi que les sur-exploitations sont de simples recherches dans les logs. Pour les crashs, on affiche tous les logs ayant le champ `class` indiquant le service de crash. Pour le CPU, tous les logs de pourcentage d'utilisation CPU avec une valeur plus grande que 3 (les pourcentages des 4 coeurs sont additionnés). Et pour l'utilisation de la mémoire, une valeur plus grande que 0.8. Pour la dernière visualisation, on a un tableau affichant le nombre de logs contenant l'information de Docker `container status` à `Up 2 minutes`.

Sur la capture d'écran de la figure 15, on peut voir un crash du simulateur, plusieurs logs de CPU sur-exploité, aucun pour la mémoire, et quelques redémarrage de conteneur. Pour les redémarrage conteneur, les premiers histogrammes correspondent au premier lancement de l'application, donc les redémarrages qui doivent être interprétés comme des erreurs sont seulement les suivants.

À la vue de ce tableau de bord, on peut donc rapidement conclure que dans l'heure qui précède, on a eu un crash, des problèmes de CPU et un redémarrage de conteneur.

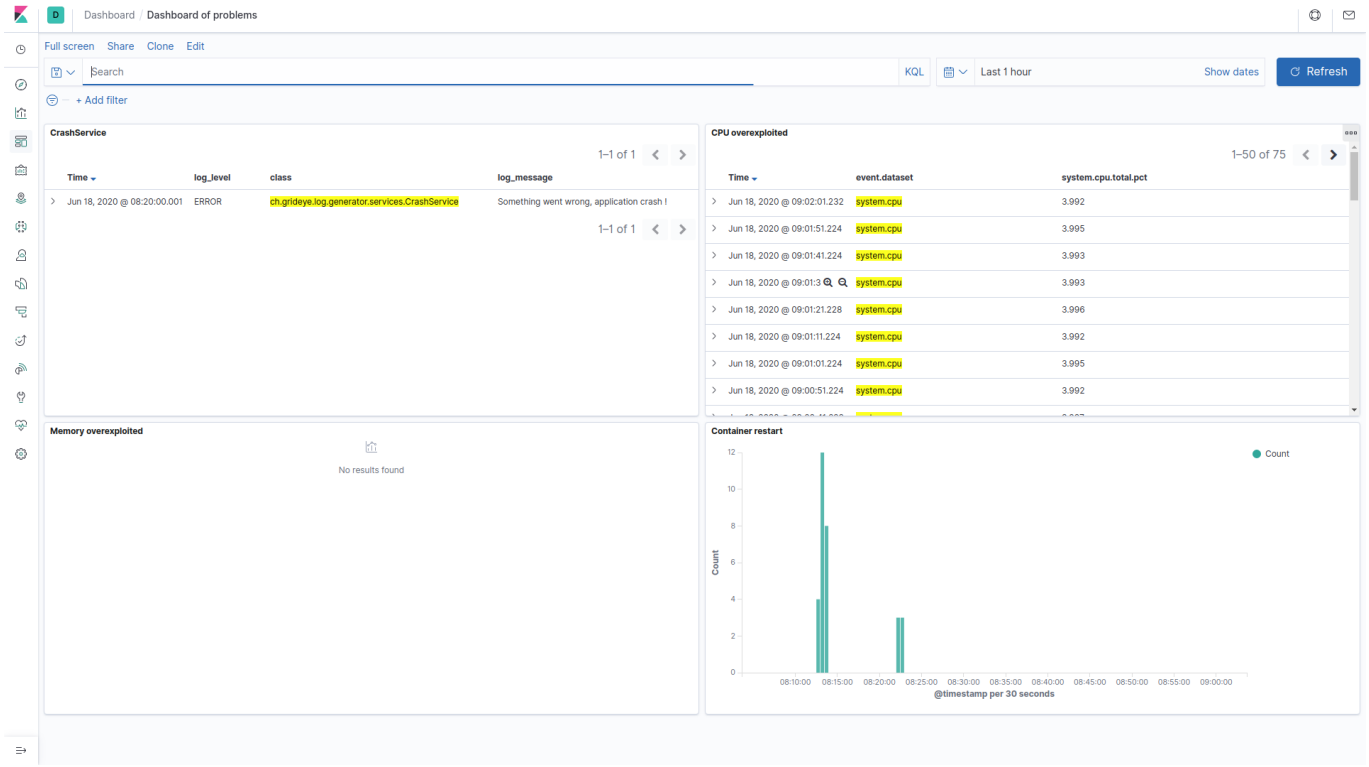

FIGURE 15 – Tableau de bord des problèmes - Kibana

4.2.2. Tableau de bord du CPU

Après avoir constaté ces différents problèmes, on pourrait imaginer que l'opérateur veuille s'attaquer à la surcharge du processeur, et voudrait en savoir un peu plus sur l'origine de cette anomalie. Un tableau de bord contenant les informations détaillées des consommations de CPU a donc été créé.

La figure 16 montre ce tableau de bord.

Les valeurs d'utilisation du CPU sont les mêmes que dans le tableau des problèmes : les pourcentages des 4 coeurs sont additionnés. Elles varient donc de 0 à 4. Dans cette capture d'écran, on remarque que le problème vient du simulateur GridEye, car il utilise périodiquement les pleines capacités du CPU, alors que les trois autres conteneurs ne dépassent pas les 3% sur la globalité des 4 coeurs. L'opérateur pourra encore pousser ces recherches en regardant les logs de GridEye aux heures où il consomme tout le processeur.

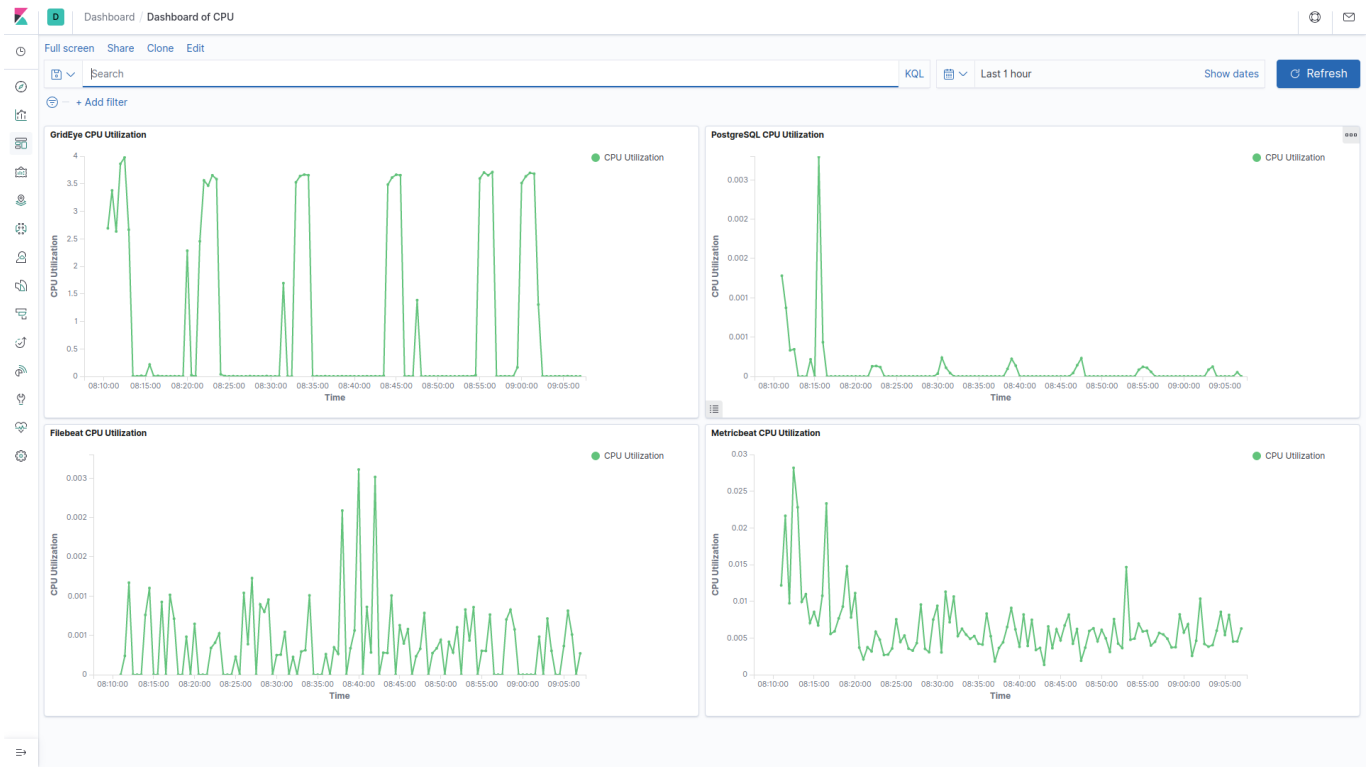

FIGURE 16 – Tableau de bord du CPU - Kibana

ANNEXES

Liste des tableaux

Table des figures

Références

[1] *6 Best Log Management Tools For Linux in 2020*. AddictiveTips. Library Catalog : www.addictivetips.com Section : Network Admin. 23 août 2019. URL : https://www.addictivetips.com/net-admin/linux-log-management-tools/ (visité le 08/05/2020).

[2] *Best Log Manager Software & Tools for Log Monitoring & Events for 2020*. ITT Systems. URL : https://www.ittsystems.com/log-manager-software-and-tools/ (visité le 21/03/2020).

[3] *DEPsys solution : GridEye*. DEPsys. URL : https://www.depsys.ch/solutions/ (visité le 16/06/2020).

[4] *Gestion des logs*. In : *Wikipédia*. Page Version ID : 160258700. 19 juin 2019. URL : https://fr.wikipedia.org/w/index.php?title=Gestion_des_logs&oldid=160258700 (visité le 08/05/2020).

[5] *Google Trends*. Google Trends. Library Catalog : trends.google.fr. URL : https://trends.google.fr/trends/explore?date=2010-04-23%202020-04-23&q=graylog,Loggly,Elasticsearch,Splunk (visité le 08/05/2020).

[6] *ISO 8601*. In : *Wikipédia*. Page Version ID : 169232031. 6 avr. 2020. URL : https://fr.wikipedia.org/w/index.php?title=ISO_8601&oldid=169232031 (visité le 08/05/2020).

[7] Jael24. *Jael24/TB_ElasticStack*. original-date : 2020-04-15T08 :28 :17Z. 13 mai 2020. URL : https://github.com/Jael24/TB_ElasticStack (visité le 14/05/2020).

[8] Jael24. *Jael24/TB_GridEye_Simulator*. original-date : 2020-05-08T10 :12 :58Z. 13 mai 2020. URL : https://github.com/Jael24/TB_GridEye_Simulator (visité le 18/06/2020).

[9] Ayushi Sharma SAYS. *Ten alternatives to Cronolog*. Comparitech. Library Catalog : www.comparitech.com Section : Net Admin. 1er mar. 2019. URL : https://www.comparitech.com/net-admin/log-management-tools/ (visité le 08/05/2020).

[10] *Test grok patterns*. URL : http://grokconstructor.appspot.com/do/match#result (visité le 09/04/2020).

[11] *Top 8 BEST Log Management Software | Log Analysis Tool Review 2020*. URL : https://www.softwaretestinghelp.com/log-management-software/ (visité le 08/05/2020).

[12] *What Is Log Management ? A Complete Logging Guide | The Graylog Blog*. URL : https://www.graylog.org/post/what-is-log-management-a-complete-logging-guide (visité le 08/05/2020).

A. Fichiers

A.1. logstash.conf

```
input {
    beats {
        port => 5044
        tags => ["filebeat"]
    }

    beats {
        port => 5045
    }
}

filter {
    if ("filebeat" in [tags]) {
  grok {
      match => {"message" => "%{TIMESTAMP_ISO8601:date_hour}%{SPACE}%{LOGLEVEL:log_level
    }%{SPACE}%{JAVACLASS:class} -> %{JAVALOGMESSAGE:log_message}"}
  }
    }
}

output {
    elasticsearch {
        hosts => ["elasticsearch:9200"]
    }
}
```

A.2. filebeat.yml

```
filebeat.inputs:
- type: log
  enabled: true
  paths:
    - <REPOSITORY>/GridEye_Simulator.log


output.logstash:
  hosts: ["<IP>:5044"]
```

Remplacer `<REPOSITORY>` par la localisation du répertoire contenant le fichier de logs.
Remplacer `<IP>` par l'adresse IP de la machine sur laquelle Logstash tourne.

A.3. filebeat.yml de l'image Docker pour l'implémentation du cas d'utilisation

```
filebeat.inputs:
  - type: log
    enabled: true
    paths:
            - /usr/share/filebeat/logs/GridEye_sim.log


output.logstash:
  hosts: ["192.168.1.119:5044"]
```

A.4. metricbeat.yml de l'image Docker pour l'implémentation du cas d'utilisation

```
metricbeat.modules:
- module: system
  metricsets: ["cpu","memory"]
  enabled: false
  period: 10s
  processes: ['.*']
```

```
7
8     cpu.metrics:  ["percentages","normalized_percentages"]
9     core.metrics: ["percentages"]
10
11  - module: postgresql
12    enabled: false
13    period: 10s
14    metricsets: ["database"]
15    hosts: ["postgres://postgres:5432?sslmode=disable"]
16    username: postgres
17    password: postgres
18
19  metricbeat.modules:
20  - module: docker
21    metricsets:
22      - "container"
23      - "cpu"
24      #- "diskio"
25      - "event"
26      #- "healthcheck"
27      - "info"
28      #- "image"
29      - "memory"
30      - "network"
31    hosts: ["unix:///var/run/docker.sock"]
32    period: 10s
33    enabled: true
34
35  output.logstash:
36    hosts: ["192.168.1.119:5045"]
```

A.5. docker-compose.yml du côté terrain pour l'implémentation du cas d'utilisation

```
1  version: '3'
2
3  services:
4
5    grideye_simulator:
6      container_name: grideye_sim
7      build: ../images/GridEye_Simulator
8      restart: on-failure
9      ports:
10       - 2404:2404
11     volumes:
12       - applicativeLogs:/logs
13
14   filebeat:
15     container_name: fbeat
16     build: ../images/filebeat
17     volumes:
18       - applicativeLogs:/usr/share/filebeat/logs
19
20   metricbeat:
21     container_name: mbeat
22     build: ../images/metricbeat
23     depends_on:
24       - db
25     volumes:
26       - /var/run/docker.sock:/var/run/docker.sock
27
28   # Non-persistant DB
29   db:
30     container_name: postgres
31     build: ../images/postgresql
32     ports:
```

```
      - 5432:5432


volumes:
  applicativeLogs:
    driver: local
    driver_opts:
      type: 'none'
      o: 'bind'
      device: '<DOCKER_FIELD_PATH>/images/GridEye_Simulator/logs'
```

Remplacer `<DOCKER_FIELD_PATH>` par la localisation du répertoire contenant l'arborescence de la machine « terrain ».

B. Détails des tests

B.1. Tests de débit d'ingestion

Graylog (3.2.4 Virtual appliance) avec Filebeat :

Sans extractor (module de Graylog permettant le parsing des logs) :

Début du chargement : 10 :09 :17
Fin du chargement : 10 :09 :24
Temps de chargement : 7 secondes
Débit moyen : ~14'043,47 EPS

Avec extractor :

Début du chargement : 07 :46 :47.153
Fin du chargement : 07 :46 :59.648
Temps de chargement : 12.495 secondes
Débit moyen : ~7'867,55 EPS

Elastic Stack sans Logstash :

Début du chargement : 10 :34 :34.200
Fin du chargement : 10 :35 :03.622
Temps de chargement : 29.422 secondes
Débit moyen : ~3'341,21 EPS

Elastic Stack complet (sans filtrage du log) :

Début du chargement : 10 :44 :11.851
Fin du chargement : 10 :44 :31.920
Temps de chargement : 20.069 secondes
Débit moyen : ~4'898,35 EPS

Elastic Stack complet (avec filtrage du log) :

Début du chargement : 10 :26 :37.308
Fin du chargement : 10 :26 :48.015
Temps de chargement : 10.707 secondes
Débit moyen : ~9'181,38 EPS

Splunk :

Avec filtrage :
Donné (metrics.log de Splunk) : 2'628 EPS

Solarwinds Loggly avec endpoint bulk : Avec Loggly, il est impossible de charger un fichier de plus de 5 MB. Il a donc été réduit à 39'999 lignes.

Début du chargement : 08 :42 :11.360
Fin du chargement : 18 :42 :12.224
Temps de chargement : 0.864 seconde
Débit moyen : ~46'295,14 EPS

C. Journal de travail

C.1. Jeudi 13 février

Première réunion avec Nastaran, Jonathan et Pascal. Jonathan et Pascal ont expliqué leur vision du TB à travers une présentation, puis nous avons planifié le travail de Bachelor. Notamment les dates de fin d'évaluation (avec la présentation à DEPsys), et de fin de développement du use-case.

C.2. Mercredi 19 février

Début du Travail de Bachelor. J'ai commencé par suivre un tuto afin de maîtriser les bases du langage LaTeX, ce qui me sera utile pour tout ce qui est rédactionnel. Ensuite, j'ai commencé à revoir la présentation de Pascal afin de bien comprendre (notamment les technologies que je ne connais pas).

C.3. Jeudi 20 février

J'ai regardé plusieurs vidéos qui présentent les différentes technologies que je dois évaluer.

C.4. Mercredi 26 février

J'ai décidé de commencer à évaluer plus en profondeur Elasticsearch en premier, car Prometheus à comme contrainte de ne pas gérer les logs textuels, mais uniquement des métriques numériques. Cependant, d'après plusieurs lectures, je pense qu'il pourrait être intéressant de mixer les deux solutions. J'ai donc installé les outils de la suite ELK, et suivi des tutos plus concret en ce qui concerne Elasticsearch (insertion de donnée, recherches, etc.).

C.5. Jeudi 27 février

Deuxième réunion avec Nastaran. Elle me propose de recentrer mes recherches sur la partie « Log Analysis », donc rechercher directement l'intégration de l'analyse de logs avec ELK par exemple. Après la réunion, j'ai donc continuer mes recherches dans ce sens et ai suivi la vidéo de elastic qui concerne l'analyse de logs.

C.6. Mercredi 04 mars

J'ai commencé à écrire ce journal afin de mieux me rappeler de ce que j'ai fais, ainsi que d'être plus structuré. Je commence également à utiliser Zotero, qui permet d'enregistrer tous les liens que je trouve intéressant, ainsi que de créer une bibliographie. J'ai également décidé de m'intéresser à Graylog en plus de ELK.

C.7. Jeudi 05 mars

J'ai exploré plus en profondeur les articles de type « Elastic Stack versus Graylog », et je vais donc inclure la stack « Graylog server, MongoDB et Elasticsearch »dans le comparatif. Cette suite-là me semble très appropriée au traitement et à l'analyse de logs. J'ai été à la réunion avec Nastaran à 11h30. Suite à cette réunion, nous avons décidé qu'il fallait que je fasse une synthèse des mes recherches et que je la présente en quelques slides le jeudi 12 mars.

C.8. Lundi 09 mars

J'ai commencé à faire la synthèse de mes recherches. Je vais donc la faire en 3 étapes :

1. Choix des critères d'évaluations
    a) Selon des recherches au sujet des caractéristiques d'un « Log Management Tool »

2. Choix des outils à évaluer
    a) Pour cette étape, je vais consulter plusieurs classements de système de gestion de logs et choisir ceux qui sont le plus souvent cités. Je vais probablement en prendre 5 ou 6.
3. Synthèse et rédaction des slides

Ce lundi, j'ai défini les critères d'évaluation, selon les demandes de DEPsys ainsi que les critères lu lors de mes recherches.

C.9. Mercredi 11 mars

J'ai fais un tableau pour le choix des outils à tester. J'ai donc effectué un classement selon 4 tops de système de gestion de logs. J'ai également commencé l'évaluation à proprement parler, en particulier sur Elastic Stack et Graylog. J'ai également eu un problème de stockage de la base de donnée Zotero et j'ai perdu toute ma bibliographie.

C.10. Jeudi 12 mars

J'ai continué l'évaluation avec Loggly, j'ai créé la présentation de synthèse pour la réunion avec Nastaran, puis je l'ai présentée.

C.11. Mercredi 18 mars

J'ai remis en place Zotero, cette fois avec un synchronisation en ligne de ma bibliographie. J'ai analysé les différents systèmes de gestion de logs que j'hésitais à inclure dans l'évaluation. J'ai donc écarté ManageEngine EventLog Analyzer pour sa popularité vraiment faible et son manque de documentation, et PRTG Network Monitor, qui est très axé sur l'analyse d'un réseau, comme son nom l'indique. Je vais donc évaluer Splunk.

C.12. Jeudi 19 mars

J'ai fais l'évaluation de Splunk. J'ai également eu la réunion hebdomadaire avec Nastaran.

C.13. Samedi 21 mars

J'ai reformaté mon rapport avec le template LaTeXécrit par Mateo Tutic. J'ai également développé la partie Choix des différents systèmes à évaluer.

C.14. Dimanche 22 mars

J'ai continué la partie Choix des différents systèmes à évaluer. J'ai également télécharger la suite Elastic et testé avec les logs systèmes de Ubuntu. Cela fonctionne normalement.

C.15. Lundi 23 mars

J'ai mis en place les systèmes de gestion de logs Elastic Stack, Graylog, Splunk et Loggly. J'ai testé (en insérant des logs et regardant le débit) les 3 premiers. Encore quelques problèmes pour Loggly (pour l'instant, il sauvegarde 1 log avec n lignes dans le message plutôt que n logs avec 1 ligne). J'ai également terminé les tableaux récapitulatif de l'évaluation de chaque système.

C.16. Mardi 24 mars

J'ai terminé les tests d'ingestions de logs pour les 4 systèmes. J'ai commencé à faire mes slides pour la présentation du 25 mars.

C.17. Mercredi 25 mars

J'ai terminé les slides de la présentation. J'ai fais la présentation du travail de Bachelor à l'entreprise DEPsys. S'en est suivi une discussion avec Pascal, Jonathan, Nastaran et moi au sujet de la suite de l'évaluation de mon TB, puis un ajustement du cas d'utilisation à implémenter fourni par Pascal.

C.18. Jeudi 26 mars

J'ai commencé à refaire des tests d'ingestion de log. Cette fois-ci avec tout le pipeline. Je commence avec Elastic Suite, en y intégrant logstash afin qu'il filtre les données.

C.19. Mercredi 01 avril

J'ai continué les tests d'ingestion avec Elastic Suite et rencontré beaucoup de problème. J'arrive à faire fonctionner un pipeline Logstash-Elasticsearch-Kibana, et un pipeline Filebeat-Elasticsearch-Kibana, mais pas un contenant les 4 logiciels de la suite.

C.20. Jeudi 02 avril

J'ai continué les tests en tentant plusieurs tutoriaux trouvé sur internet. Mais je rencontre toujours des problèmes. Ils sont probablement liés à la communication entre Filebeat et Logstash. J'ai également suivi les tuto officiels de la Suite Elastic, mais ça n'a pas fonctionné non plus. Ceci est peut-être dû à mes fichiers de configurations des logiciels Filebeat et Logstash. J'ai ensuite eu une réunion avec Nastaran.

C.21. Mercredi 08 avril

J'ai continué les tests d'ingestion. En suivant les guides du site d'elastic, j'ai remarqué qu'il y avait toute une section expliquant l'utilisation de la Suite avec Docker. Je me suis dit qu'il y avait plus de chance que cela fonctionne étant donné l'uniformité que propose Docker. Malheureusement, j'ai toujours les mêmes problèmes. Même en prenant un git public sensé fonctionner. Je me dit alors que le problème vient peut-être de mes fichiers de logs de tests (ils contiennent le même log multiplié n fois).

C.22. Jeudi 09 avril

Je suis repassé sur une version non dockerisée de la suite Elastic. J'ai téléchargé un fichier de log d'un serveur Apache afin de tester la Suite avec un fichier de log réel, et fait d'autres modifications, notamment sur les fichiers de configuration (j'ai créé la partie « filtrage »de Logstash avec un site internet permettant de créer ces filtres de manière incrémentale). Et ça à fonctionné. J'ai ensuite eu la réunion avec Nastaran.

C.23. Lundi 13 avril

J'ai effectué les tests de performances avec filtrage de la Suite Elastic. Après ceci, je me suis lancé dans les tests avec filtrage de Graylog. Cette fois-ci, ce n'est pas avec un logiciel intermédiaire comme Logstash, mais avec une fonctionnalité intégrée à Graylog : les Graylog Extractors.

C.24. Mercredi 15 avril

J'ai commencé à cherché une façon d'effectuer le filtrage avec le système de gestion de logs Splunk. Malheureusement, j'ai l'impression que cela va être plus compliqué car Splunk favorise l'extraction des informations après l'indexage. Je vais encore chercher une journée, et si ce n'est pas concluant, je passerai outre.

C.25. Jeudi 16 avril

J'ai tenté d'effectuer l'extraction d'informations dans les logs durant la phase de "parsing" du pipeline de Splunk. J'ai vu qu'il devait être possible de le faire en modifiant des fichiers de configuration dans le répertoire de Splunk, mais cela n'a pas fonctionné.

C.26. Lundi 20 avril

J'ai rédigé les résultats des tests d'ingestion. J'ai ensuite commencé à étudié les manières de faire des tests de consommation CPU. Sachant qu'avec Amazon Web Service (AWS), comme je l'avais vu quelques semaines plus tôt dans un cours de Cloud Computing, il est possible de monitorer différentes métriques d'une instance, entre autres l'utilisation du CPU, je me suis lancé dans une installation de la Suite Elastic sur une instance t2.micro d'AWS. Malheureusement, ces instances sont trop petites et ne supportent pas simplement Elasticsearch. Ne voulant pas payer pour des instances plus grosses, je me suis rabattu sur la solution locale. Je vais donc simplement stopper le maximum de processus et monitorer l'utilisation de mon CPU avec l'outil natif d'Ubuntu. Je dois aussi installer Filebeat et Graylog sur un autre ordinateur afin de pouvoir effectué ces tests (dans la réalité, le serveur et le client ne seront pas sur la même machine).

C.27. Mercredi 22 avril

J'ai effectué les tests de consommation CPU de la Suite Elastic, de Graylog, ainsi que de Filebeat. J'ai ensuite commencé à rédiger la synthèse de ces tests.

C.28. Jeudi 23 avril

J'ai continué la rédaction, j'ai ajouté le test d'ingestion d'un grand nombre de logs, la comparaison de popularité.

C.29. Vendredi 24 avril

J'ai rédigé le cahier des charges, puis ai eu une réunion avec Nastaran. Nous avons discuté du cahier des charges, des améliorations à y apporter. Ensuite, nous avons un peu regarder l'état du rapport actuel. Il faut entre autre pense a y ajouter les références (tableaux, image, bibliographie). Nous avons ensuite convenu de fixer une réunion avec Jonathan et Pascal afin de leur présenter les résultats de l'évaluation. J'ai donc fixé cette réunion à jeudi 30 avril.

C.30. Mercredi 29 avril

J'ai modifié le rapport pour y ajouter les références sur les tableaux et sur les figures. J'ai également amélioré les annexes avec la table des tableaux et la table des figures. J'ai également commencé la conclusion finale de l'évaluation, en rédigeant le paragraphe concernant l'analyse théorique.

C.31. Jeudi 30 avril

J'ai continué le rapport. J'y ai ajouté une conclusion finale à l'évaluation. J'ai ensuite eu une réunion avec Pascal, Jonathan et Nastaran. Je leur ai présenté les résultats de mon évaluation et nous avons convenu ensemble d'effectuer l'implémentation du cas d'utilisation avec le système Elastic Stack.

C.32. Mercredi 6 mai

J'ai pris en main le programme de Jonathan qui simule l'infrastructure GridEye de façon minimale. Je l'ai modifié légérement pour qu'il stocke les logs dans un fichier, plutôt que de les afficher dans la console. J'ai ensuite commencé à rédiger la partie implémentation du cas d'utilisation dans le rapport, tout en testant les commandes que je met dans le rapport sur une installation neuve d'Ubuntu.

C.33. Jeudi 7 mai

J'ai continué la rédaction et là réalisation de l'implémentation du use case. J'ai rédigé les parties installations de la Suite Elastic, Configuration de la Suite Elastic.

C.34. Vendredi 8 mai

J'ai continué la rédaction et la réalisation de l'implémentation du use case. J'ai maintenant un pipeline entier qui fonctionne avec le simulateur GridEye. J'ai également décrit tout cela dans le rapport. J'ai complété la bibliographie.

C.35. Mercredi 13 mai

Je me suis lancé dans la dockerisation de la Suite Elastic. J'ai commencé par le faire avec 3 images Docker séparée, que je lançais avec un docker run, comme indiqué dans la documentation. Cela fonctionnait bien avec Elasticsearch et Kibana, mais Logstash n'arrivait pas à communiquer avec Elasticsearch. J'ai remarqué que ceci était dû à la séparation des conteneurs effectuée par Docker. En effet, il n'était pas possible pour Logstash d'atteindre Elasticsearch avec l'adresse localhost, car pour le conteneur Logstash, il n'y a que lui sur « son »localhost. J'ai donc décidé de passer à une topologie avec un docker-compose. Ayant plus d'expérience avec les configurations docker-compose que les simples commandes docker, je pouvais plus facilement créer les liens entre les différentes applications. J'ai donc créé une topologie faisant communiquer toutes les applications de la Suite Elastic. J'ai ensuite rédigé mon rapport et j'y ai inclus un schéma montrant cette topologie.

C.36. Jeudi 14 mai

J'ai continué le rapport avec la partie ELK et l'installation avec Docker et docker-compose. J'ai ensuite commencé à faire un schéma similaire pour la partie terrain, qui est un peu plus complexe. J'ai envoyé une première idée du schéma à Depsys et Nastaran ainsi que mes questions, comme le rôle de la base de donnée PostgreSQL dans cette architecture. J'ai également eu une réunion avec Nastaran.

C.37. Mercredi 20 mai

Après avoir reçu des réponses sur mes questions ainsi que l'approbation de Pascal quant à mon schéma. Je me suis lancé dans la réalisation de la topologie du côté terrain. J'ai décidé de construire mon conteneur docker contenant l'application Java du simulateur GridEye à partir d'un fichier `.jar`. J'ai eu quelques problèmes avec le fichier de logs dans lequel le simulateur doit écrire ses logs, et l'agent filebeat lire. Je rapidement compris qu'il me fallait là un Volume Docker partagé entre les deux conteneurs.

C.38. Jeudi 21 mai

Je me suis renseigné sur les Volume Docker et leur intégration avec docker-compose. J'en ai donc intégré dans mon architecture pour partager le fichier de logs entre le simulateur GridEye et Filebeat. J'ai continué la réalisation de la topologie côté terrain avec l'intégration de Metricbeat et PostgreSQL.

C.39. Mercredi 27 mai

En vue de la réunion du 28 mai, qui a pour but de présenter mes avancées dans l'implémentation du cas d'utilisation, j'ai terminé la topologie docker du côté terrain, et ait tout fait fonctionné ensemble (terrain et ELK). J'ai aussi commencer à créer mes premières visualisations et tableau de bord (dashboard) dans Kibana. J'ai également avancé le rapport.

C.40. Jeudi 28 mai

J'ai tenté de lancer un deuxième « device terrain »à l'aide d'une machine virtuelle. Le test s'est bien passé. J'ai ensuite préparé ma présentation, et fait ma présenation à Jonathan, Pascal et Nastaran. Lors de cette réunion, DEPsys était globalement content de l'avancée et de l'architecture mise en place, qui correspondait à leurs attentes. Les prochaines étapes pour le cas d'utilisation étaient principalement situées dans le dashboard de Kibana, afin de créer, par exemple, un dashboard des problèmes, permettant à un opérateur de voir rapidement si une anomalie est présente, ou a été présente, dans le système.

C.41. Jeudi 4 juin

J'ai avancé le rapport, dans lequel j'avais pris un peu de retard suite aux nombreuses implémentations pour le cas d'utilisation.