Fournier Arnaud
Delalande Mathieu

Rapport du projet IF27

La sécurité autour du Big Data

Semestre : Printemps 2016

Table des matières

Introduction

Aujourd'hui internet fait partie du quotidien de millions voire de milliards de personnes dans le monde. Les gens se retrouvent de plus en plus connectés et les grandes entreprises collectent un très grand nombre de données, Google détiendrait à lui seul 10 à 15 exaoctets de données soit environ 10^19 octets. C'est données sont souvent variées et il est souvent nécessaire d'obtenir un accès rapide. Les méthodes classiques de stockage incarnées par les bases de données relationnelles ne suffisent plus à les traiter efficacement. Il est donc obligatoire de repenser nos méthodes et outils d'analyses afin de garantir une utilisation optimale des données que l'on dispose. Le Big Data doit satisfaire trois critères principaux, que l'on regroupe sous le sigle des 3V pour volume, vitesse et variété. Il est une technologie qui ne s'arrête pas seulement au stockage des données, les Hommes ont aussi compris qu'ils pouvaient analyser cette quantité astronomique de données pour en soutirer des informations très importantes.

Le Big Data a donc trouvé un echo dans de nombreux secteurs, nous pouvons citer l'exemple du marketing pour ce qui résulte de l'analyse comportementale des acheteurs, mais aussi dans le domaine de la sécurité informatique pour détecter des comportements suspects ou des attaques informatiques. C'est ce dernier domaine qui nous intéresse, nous allons donc le détailler.

Nous réaliserons dans un premier temps un état de l'art des solutions Big Data, notamment les outils disponibles sur le marché et ceux consacrés à la sécurité. Ensuite nous expliquerons les tests que nous avons réalisé afin de tester la puissance de cette technologie. Pour ces expériences nous nous sommes mis dans deux cas de figure courants du monde professionnel.

I - Etat de l’art des solutions Big data

1.1. Outils pour le Big Data

Les grandes entreprises du web (Google, Facebook, Amazon, …) ont été de grands acteurs du développement du Big Data. Au fur et à mesure de l’accroissement de la quantité de données stockées sur leurs serveurs, ces sociétés ont dû trouver des moyens pour répondre à leurs besoins.

1.1.1. De nouvelles bases de données

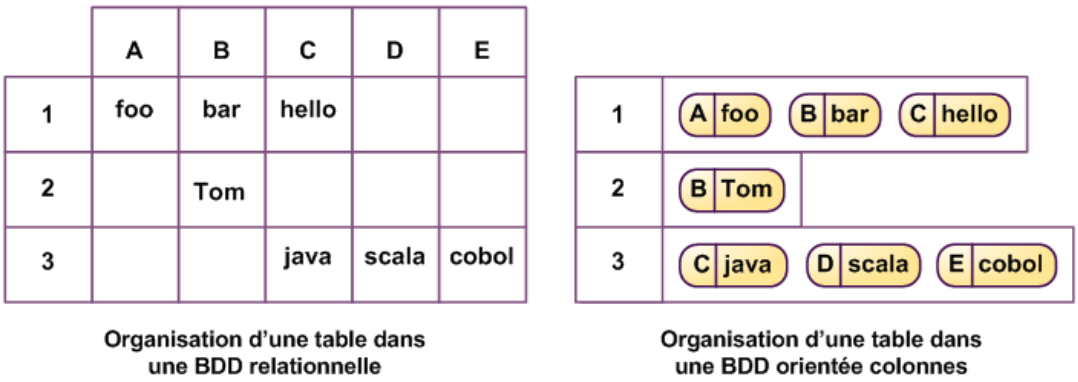
Les bases de données relationnelles ont montré leurs limites. Lorsqu’elles sont fortement sollicitées, par exemple par une multitude de connexions simultanées réalisées vers un site web, elles peuvent avoir du mal à gérer le traitement des flux de données en temps réel car le système produit plus de données qu’il ne peut en traiter. Il se trouve alors considérablement ralenti ce qui peut rendre le site inutilisable. Au-delà d’un certain seuil de données stockées, le modèle relationnel est également limité. En effet, compte tenu de l’organisation des données en lignes, lors de l’exécution d’une requête le système doit lire tous les attributs en mémoire avant d’appliquer une fonction d’analyse. Cela ralentit le temps de réponse et ne convient pas aux secteurs qui réalisent des opérations “instantanées” comme la finance ou les sites de e-commerce, ou encore ceux qui analysent les données en temps réel comme les moteurs de recherche. L’organisation de ces bases de données ne permet pas toujours de mettre en évidence les liens entre les données, par exemple les liens d’amitié entre deux utilisateurs d’un réseau social.

Pour répondre à ces problématiques de nouvelles formes de bases de données ont été créées. Elles sont réunies sous la notion de bases de données NoSQL puisque leur manipulation se fait avec différents langages de programmation et non plus seulement avec le langage SQL. Nous pouvons les séparer en plusieurs catégories :

- Entrepôts clé/valeurs : elles peuvent être envisagées comme des collections de couples clé-valeurs. On peut effectuer des opérations de récupération de valeurs pour une clé donnée, de mise à jour, de création ou de suppression d’une valeur pour une certaine clé. La base ne différencie pas la structure des données.

- Bases de données orientées documents : Ici chaque clé n’est plus associée à une valeur sous forme de bloc binaire mais à un document dont la structure reste libre. La base de données est consciente de la structure de la valeur qu’elle stocke. Elle sera ainsi à même d’offrir un certain nombre de fonctionnalités qui ne peuvent être accessibles avec une base de données clé-valeur. Cela concerne notamment l’ajout, la modification, la lecture ou la suppression de seulement certains champs dans un document.

- Bases de données orientées colonnes : Il s’agit ici de représenter les données sous la forme de blocs de colonnes stockés de manière triée sur le disque.



Organisation d'une table dans une BDD relationnelle

Organisation d'une table dans une BDD orientée colonnes

Comparaison de l'organisation d'une base de données relationnelle et orientée colonnes

On constate bien l'optimisation qui est faite lors du stockage. Les cases "vides" des tables de données relationnelles contiennent la valeur "null" ce qui n'est pas neutre d'un point de vu mémoire contrairement à la BDD orientée colonnes.

- Bases de données orientées graphes : Bases de données orientées objet utilisant la théorie des graphes, donc avec des nœuds et des arcs, permettant de représenter et stocker les données. Très présentes par exemple dans les réseaux sociaux, il s'agit de relier des entités par différentes relations.

Il existe plus d'une centaine d'implémentations de bases de données Nosql, nous avons choisi de lister les principales dans un tableau :

Nom	Type de données	Licence	développé par
Riak	clé-valeur	libre	Basho Technologies
Redis	clé-valeur	libre	Salvatore Sanfilippo
DynamoDB	clé-valeur	propriétaire	Amazon
CouchDB	documents JSON	libre	Fondation Apache
MongoDB	documents JSON	libre	MongoDB
Cassandra	colonnes	libre	Initialement par Facebook, désormais fondation Apache
Hbase	colonnes	libre	Fondation Apache
BigTable	colonnes	propriétaire	Google
Hypertable	colonnes	libre	Zvents Inc.
Neo4j	graphes	libre	Neo Technology
InfiniteGraph	graphes	propriétaire	Objectivity, Inc.

1.1.2. Un nouveau type de stockage de données

Des systèmes de fichiers distribués ont été développés dans le but de stocker de très gros volumes de données sur un grand nombre de machines équipées de disques durs banalisés. Les systèmes de fichiers distribués permettent l'abstraction de l'architecture physique de stockage, afin de manipuler un système de fichiers distribué comme s'il s'agissait d'un disque dur unique. Les systèmes de fichiers distribués permettent donc de scinder les données à enregistrer sur plusieurs serveurs tout en réussissant à les recoller par la suite lorsque les données auront besoin d'être extraites.

Google et son système de fichiers distribués Google File System a été l'une des premières technologies du genre. Hadoop Distributed File System s'inspire du système de fichier de Google et est aujourd'hui la solution la plus répandue.
Une architecture de machines HDFS (aussi appelée cluster HDFS) repose sur deux types de composants majeurs :
Le NameNode ou noeud de noms : ce composant gère l'espace de noms, l'arborescence du système de fichiers et les métadonnées des fichiers et des répertoires. Il centralise la localisation des blocs de données répartis dans le cluster. Il est unique mais dispose d'une instance secondaire qui gère l'historique des modifications dans le système de fichiers (rôle de backup). Ce NameNode secondaire permet la continuité du fonctionnement du cluster Hadoop en cas de panne du NameNode d'origine.
Le DataNode ou nœud de données : ce composant stocke et restitue les blocs de données. Lors du processus de lecture d'un fichier, le NameNode est interrogé pour localiser l'ensemble des blocs de données. Pour chacun d'entre eux, le NameNode renvoie l'adresse du DataNode le plus accessible, c'est-à-dire le DataNode qui dispose de la plus grande bande passante. Les DataNodes communiquent de manière périodique au NameNode la liste des blocs de données qu'ils hébergent.
Par défaut les données sont stockées sur trois nœuds : deux sur le même support et l'autre sur un support différent. Les DataNodes peuvent communiquer entre eux afin de rééquilibrer les données et garder un niveau de réplication des données élevé.

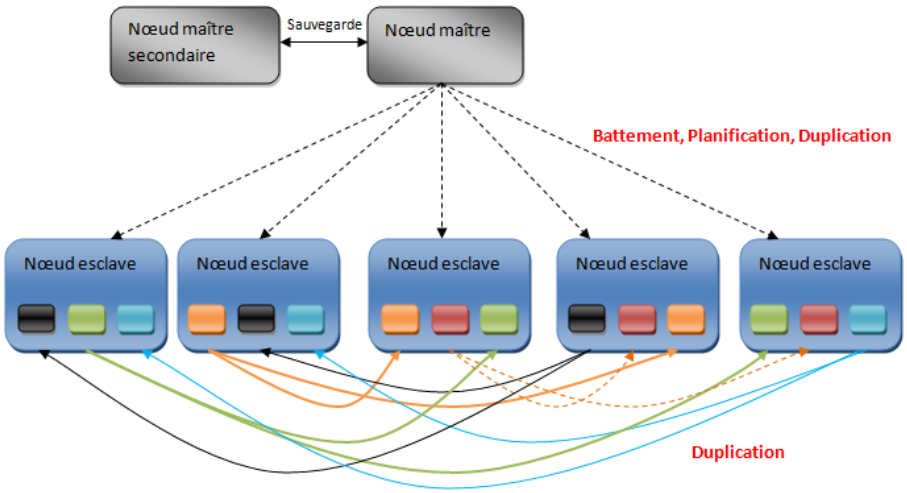


Schéma de fonctionnement de HDFS

Les NameNode sont les noeuds maitres tandis que les DataNodes sont les noeuds esclaves.

1.1.3. Une nouvelle méthode pour traiter les données

Des technologies de traitements rapides et parallélisés sur de vastes quantités de données, réparties sur plusieurs clusters ont été développées pour accompagner les systèmes de fichiers distribués. Nous parlerons ici du patron d'architecture MapReduce décrit pour la première fois par Google en 2004. Ce patron de conception est aujourd'hui largement utilisé pour automatiser le déploiement de traitements parallèles sur des clusters de centaines ou de milliers de serveurs.

Le principe de MapReduce est simple. Il s'agit de découper une tâche manipulant un gros volume de données en plusieurs tâches traitant chacune un sous-ensemble de ces données. Un cluster MapReduce utilise une architecture de type Maître-esclave où un nœud maître dirige tous les nœuds esclaves. MapReduce se déroule en deux étapes :

- Map : on prend en entrée un ensemble de « clé-valeurs », le nœud fait une analyse du problème et il va le séparer en sous-tâches, pour pouvoir les redistribuer sur les autres nœuds du cluster. Si nécessaire, les nœuds recevant les sous-tâches refont le même processus de manière récursive. Les sous-tâches des différents nœuds sont traitées chacune d'entre elles dans leur fonction map respective et vont retourner un résultat intermédiaire sous la forme d'une liste de clés-valeurs intermédiaire.

- Reduce : consiste à faire remonter tous les résultats à leur nœud parent respectif. Les résultats se remontent donc du nœud le plus bas jusqu'à la racine. Avec la fonction Reduce, chaque nœud va calculer un résultat partiel en associant toutes les valeurs correspondant à la même clé en un unique couple (clé–valeur). Une fois ce couple (clé-valeur) obtenu, le résultat est remonté au nœud parent, qui va refaire le même processus de manière récursive jusqu'au nœud racine.

Quand un nœud termine le traitement d'une tâche, un nouveau bloc de données est attribué à une tâche Map.

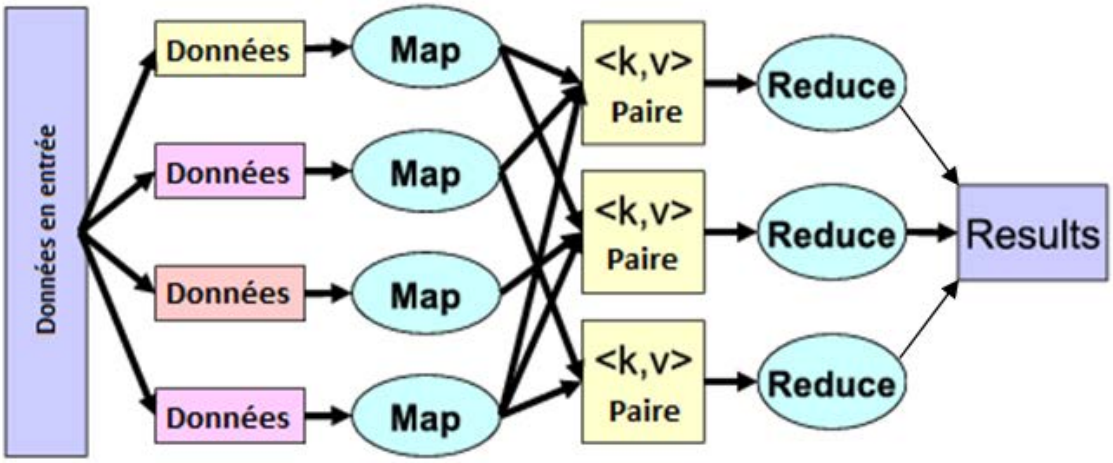


Schéma de fonctionnement du patron MapReduce

MapReduce ne serait pas d'une grande utilité dans un contexte Big Data sans une infrastructure logicielle prenant en charge la complexité d'implémentation liée à la distribution des traitements, la réplication des données et la tolérance aux pannes. C'est pourquoi des frameworks du pattern ont été développés.

1.1.4. Un framework pour rassembler ces outils

Le framework le plus utilisé à l'heure actuelle se nomme Hadoop. Hadoop est une plate-forme informatique open source créé par Doug Cutting d'après les travaux de Google, notamment ceux sur MapReduce, GoogleFS et BigTable. Hadoop prend en charge un grand nombre de tâches bas niveau, ce qui permet à l'utilisateur de se concentrer sur le traitement des données qui est fait avec des langages haut niveau comme le SQL, ou le Pig Latin. Tous les modules de Hadoop sont conçus dans l'idée fondamentale que les pannes matérielles sont fréquentes et qu'en conséquence elles doivent être gérées automatiquement par le framework.
Le noyau d'Hadoop est constitué du système de fichiers HDFS ainsi que de MapReduce pour le traitement.
Les autres composants principaux d'Hadoop sont :

- Hcatalog : Permet d'attaquer les données HDFS via des schémas de type tables de données en lecture/écriture. Permet également d'opérer sur des données issues de MapReduce, Pig ou Hive.

- ZooKeeper : Service centralisé pour gérer les informations de configuration, de nommage, et assurer la synchronisation des différents serveurs via un cluster. ZooKeeper est notamment utilisé pour l'implémentation de HBase.

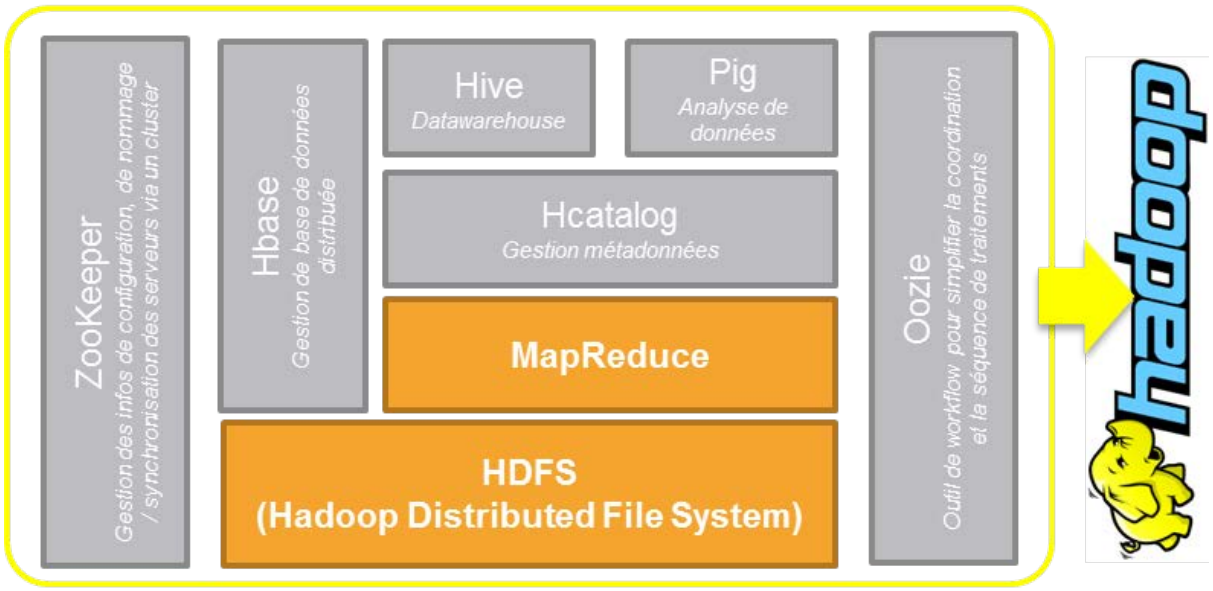
- HBase : base de données orientée colonnes.

- Oozie : Outil de workflow dont l'objectif est de simplifier la coordination et la séquence de différents traitements. Le système permet aux utilisateurs de définir des actions et des dépendances entre ces actions.

- Hive : logiciel d'analyse de données permettant d'utiliser Hadoop avec une syntaxe proche du SQL, nommée HiveQL.

- Pig : logiciel d'analyse de données comparable à Hive, qui comprend un langage de haut niveau gérant la parallélisation des traitements d'analyse.

Le terme Hadoop ne se limite pas seulement aux modules cités ci-dessus, il fait référence à tout un écosystème comprenant une multitude de composants logiciels venant s'y connecter: Spark, Flume, Sqoop, Storm, Zeppelin, YARN, etc.



Les principaux composants d'Apache Hadoop

Plusieurs éditeurs se sont livrés au développement de systèmes prêts à l'emploi, ajoutant aux principaux modules d'Hadoop des composants supplémentaires de son écosystème. Cloudera, Hortonworks, MapR, Amazon Elastic MapReduce, Microsoft HDInsight ou encore Oracle Big Data appliance en sont les principales distributions. Nous ne détaillerons pas les deux dernières bien qu'elles soient très utilisées.

- Cloudera (voir architecture Cloudera, annexe 1) ajoute au noyau d'Hadoop ses propres interfaces graphiques d'administration, de gestion de suivi, des outils de déploiement, des outils de gestion de la sécurité ainsi que des solutions d'intégration à un large éventail de produits. Cloudera est la distribution qui a le plus de déploiements référencés.

- A l'instar de Cloudera, Hortonworks reprend le noyau de Hadoop pour y ajouter de multiples composants. Cependant Hortonworks n'utilise que des composants open source Apache sans n'y ajouter de modifications (voir architecture Hortonworks, annexe 2).

- MapR est une distribution de Hadoop qui se veut particulièrement facile d'utilisation. Elle enrichit également le noyau de Hadoop mais par des logiciels propriétaires. Le système de fichier distribués est ainsi remplacé par MapR FS, tout comme MapReduce qui est implémenté sous le nom MapR MapReduce. (voir architecture MapR, annexe 3). MapR peut etre utilisée par l'intermédiaire d'Amazon Elastic MapReduce.

- Amazon Elastic MapReduce diffère des trois autres distributions puisque c'est une solution Hadoop basée le Cloud. Elle utilise son propre système de fichiers, EMRFS. C'est une solution idéale dans les situations où l'on cherche à disposer ponctuellement d'une importante puissance de traitement et de stockage. Nul besoin

de prévoir par avance la capacité de traitement et de stockage nécessaire, celles-ci peuvent être ajustées à la volée et la tarification se base sur les ressources effectivement consommées. Cette solution présente cependant les inconvénients du Cloud, c'est à dire potentiellement un temps de latence plus long que sur une installation Hadoop à domicile.

Nous avons choisi d'essayer la solution Hortonworks pour nos différents tests car elle supporte une utilisation sur machine virtuelle, contient exclusivement des logiciels open source, demande moins de ram que Cloudera (6Go contre 8), et dispose d'une interface graphique de gestion (ambari). Ces avantages font de la distribution Hortonworks un outil idéal pour débuter dans le Big Data et apprendre à utiliser les différents outils de l'écosystème Hadoop.

1.2. La sécurité autour du Big Data

La sécurité et le Big Data sont indissociables. Tout d'abord car il est évident qu'à l'heure du Big Data la sécurisation des données ainsi que des services permettant le stockage et l'administration de ces données est primordiale. Ensuite parce que l'exploitation de ces données massives grâce à des outils d'analyse du Big Data peut permettre d'augmenter considérablement la rapidité et l'efficacité dans la prévention et la gestion des incidents. Lorsque l'on sait que le nombre d'attaques malveillantes a doublé entre 2014 et 2015, passant de 4,2 milliards à 8,2 milliards d'unités, on se rend mieux compte de la nécessité d'améliorer les techniques de sécurité. Les « Black Hat » ont bien compris que le marché des données pouvait s'avérer très lucratif, ou encore compromettant pour certaines entreprises. En effet, au-delà du chiffre, les attaques deviennent de plus en plus sophistiquées et organisées, avec la multiplication des groupes de pirates qui unifient leurs forces. Du côté des entreprises, la mise en place d'un écosystème Big Data et le développement d'outils spécialisés sont coûteux et compliqués à réaliser, c'est pourquoi la majorité d'entre elles préfèrent opter pour des solutions déjà existantes basées sur le Cloud. Big Data et Cloud computing sont donc fortement liés, du fait de leur complémentarité. A tel point que l'Union Internationale des Télécommunications a proposé une normalisation de cette relation, qualifiée de "Big Data as a Service". Cela a permis de définir clairement quels sont les rôles de chacun des acteurs dans un écosystème Big Data, et les responsabilités de chacun en termes de sécurité.

Dans cette partie, nous allons donc voir dans un premier temps par quoi passe la sécurisation d'un environnement Big Data, en comparant certaines des solutions adoptées par les grands groupes. Nous discuterons ensuite des enjeux de l'analyse du Big Data à des fins de sécurité, et nous présenteront quelques-unes des solutions qui ont vu le jour sur ce marché en plein essor.

1.2.1. La sécurité dans un contexte de Big Data

Comme nous l'avons vu dans la première partie, Hadoop est un exemple brut d'environnement Big Data, dont les entreprises se servent pour créer leur environnement fonctionnel. Il faut dire que l'environnement Hadoop initial n'a pas été conçu de manière à être sécurisé. Ce n'est que lorsque le modèle a commencé à se répandre que les entreprises se sont intéressées à cet aspect, pour des raisons évidentes de confidentialité. Début 2013, Intel a lancé le projet open source Rhino, qui traduit la volonté d'un effort collectif pour sécuriser l'écosystème Hadoop le plus efficacement possible. L'objectif était de permettre à la plateforme de supporter des méthodes de sécurité plus complexes telles que le chiffrement des données ou encore la gestion de clés pour l'authentification. De nombreux modules complémentaires envahissent aujourd'hui le marché, preuve que la sécurité est devenue l'une des priorités en ce qui concerne le Big Data. Dans la suite nous allons voir quelques-unes des solutions développées afin de gérer la sécurisation d'un écosystème Big Data, autour de quatre axes que sont l'authentification, l'autorisation, la surveillance des données ainsi que la protection des données.

Authentification

Hadoop ne comporte pas de dispositif d'authentification par défaut, même s'il est conçu pour supporter le protocole Kerberos, qui est la méthode la plus sécurisée. Ce protocole est généralement utilisé pour l'authentification des clients, alors que LDAP est généralement utilisé pour l'authentification interne à l'entreprise, couplé avec Active Directory. Voici une liste des solutions utilisées par les principales distributions d'Hadoop :

- Hortonworks : Il faut configurer Kerberos manuellement sur les différents clusters, après avoir installé un KDC et différents logiciels. Il n'y a donc pas de réelle solution simplifiée pour l'instant. En revanche, **Apache Knox** est une passerelle qui permet de configurer le SSO et supporte Active Directory ainsi que LDAP. L'interface permet également de gérer l'intégration de nouvelles API dans l'écosystème.

- Cloudera : **Cloudera Manager** automatise la "Kerberization" des clusters et permet de configurer les services clients. **Hue** est également intégré et permet de faire du SSO. Comme Apache Knox, Cloudera Manager permet aussi de gérer l'intégration de nouveaux composants dans l'environnement.

- MapR : L'outil **maprlogin** gère l'authentification avec son propre système de tickets, tout en supportant Kerberos. MapR supporte aussi le SSO mais il ne semble pas y avoir de solution intégrée.

- Amazon EMR : Amazon profite de sa plateforme Cloud Amazon Web Services pour gérer sa base de compte clients, avec AWS Identity and Access Management (IAM). L'authentification multi-facteur (MFA) est supportée ainsi que la fédération d'identités via AD, Google ou Facebook. Le SSO est également supporté et configurable via **Hue**, comme Cloudera.

Contrôle des droits d'accès

La gestion des droits d'accès dans un ecosystème Hadoop devient vite problématique du fait de la multitude des composants et utilisateurs qui interagissent avec les données stockées dans les différents clusters. Il existe par ailleurs différentes méthodes de restriction d'accès aux données : le standard POSIX (Portable Operating System Interface), les ACLs (Access Control Lists) qui font partie de la famille des DAC (Discretionary Access Control), EBAC (Expression Based Access Control), RBAC (Role Based Access Control), ABAC (Attribute Based Access Control)., ou même MAC (Mandatory Access Control) lorsque les besoins en confidentialité sont très forts. Ces différents standards sont complémentaires donc une bonne politique de contrôle d'accès doit en intégrer au moins plusieurs d'entre eux. En effet, ils définissent différents types de relations entre le sujet, l'objet et la nature de l'opération effectuée. Comme pour l'authentification, nous allons faire un tour d'horizon des outils utilisés dans les principaux écosystèmes Big Data :

- Hortonworks : **Apache Ranger** est une interface centralisée qui permet la gestion des politiques de sécurité pour chaque composant de l'environnement. Elle est en plus flexible car on peut créer des plugins spécifiques pour chaque nouveau service que l'on voudrait intégrer en plus de ceux déjà présents. Apache Ranger supporte les RBAC, ABAC, ainsi que d'autres standards.

- Cloudera : Cloudera a participé activement au développement d'**Apache Sentry** qui unifie l'autorisation à partir de trois méthodes de contrôle d'accès : RBAC, ACL et un système de permission type POSIX. Sentry est également compatible avec Active Directory, avec la possibilité de synchroniser le rôle des utilisateurs avec leur groupe dans AD.

- MapR : MapR ne dispose pas d'interface centralisée pour la gestion des autorisations, mais il est maintenant possible d'y intégrer **Apache Sentry**. Les autorisations peuvent être gérées via des ACLs, ACEs et permissions type POSIX. Les rôles et privilèges sont également supportés étant donné la possibilité d'intégrer Sentry, mais doivent être gérés via **Hue.** Les accès aux sous-réseaux menant au service FileServer peuvent même être gérés via des listes blanches.

- Amazon EMR : Comme pour l'authentification, Amazon profite de **AWS IAM** pour gérer les autorisations, avec un système de permissions et de rôles. Comme EMR est basé sur le Cloud, Amazon offre également la possibilité de lancer son cluster dans **Amazon Virtual Private Cloud** (VPC) pour l'isoler du reste.

Surveillance de l'activité des données ou audit

Un des aspects importants de la sécurité est la journalisation des activités des données afin de pouvoir contrôler et vérifier les actions de chaque utilisateur et chaque service. En cas d'incident, c'est la première source d'informations pour remonter à l'origine du problème. Aussi, afin de pouvoir surveiller efficacement les données stockées, il faut

d'abord appliquer une stratégie de gouvernance des données. En effet, cela permet par exemple de bien définir quels sont les types de données dits sensibles (PII), et où elles sont situées sur les clusters, afin de détecter efficacement les opérations à risque. Les outils de gouvernance et d'audit sont donc fortement liés, c'est pourquoi nous listerons pour chaque distribution quelles sont les solutions utilisées pour ces deux aspects :

- Hortonworks : **Apache Atlas** et **Apache Ranger** sont les deux services intégrés à Hortonworks qui permettent respectivement la gouvernance et la surveillance des données. Atlas permet de classer les données en différents types et ainsi permettre d'appliquer une politique de sécurité spécifique sur Ranger. L'audit y est également centralisé ce qui facilite l'analyse des opérations et leur conformité avec les règles de sécurité.

- Cloudera : **Cloudera Navigator** est une solution centralisée qui joue deux rôles dans l'écosystème : la gestion de l'audit ainsi que la gestion des métadonnées. La gouvernance et la traçabilité sont donc assurées par ce même outil.

- MapR : MapR ne propose pas de service unifié pour la gouvernance des données. Quant à l'audit, la dernière version de MapR propose une interface centralisée, **MapR Control System**, qui permet de voir en temps réel l'état de l'écosystème, que ce soit au niveau des disques, des clusters, des nodes, etc.

- Amazon EMR : Comme pour MapR, Amazon EMR ne propose pas d'interface centralisée pour la gouvernance des données, bien que l'on puisse classer les clusters en différents groupes. Pour le monitoring, **CloudTrail** permet de garder une trace des différents accès aux données et les stocke dans un fichier log dont on peut spécifier le chemin.

Oracle a également développé sa propre solution, **Audit Vault and Database Firewall.** Tout comme Cloudera Navigator, il propose une interface centralisée pour la gouvernance et le traçage des accès aux données. Comme son nom l'indique, l'outil dispose d'un pare-feu qui fait office de première ligne de défense face aux attaques malicieuse cherchant à accéder à la base de données. Chaque requête SQL passe d'abord par le pare-feu, est analysée, puis est autorisée (ou non) à accéder à la base de données.

Protection des données

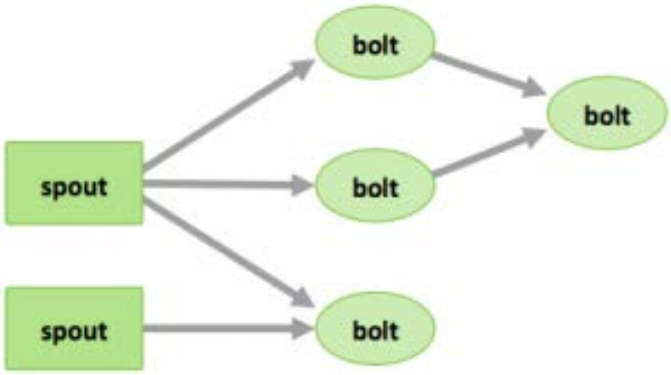
Le dernier aspect est donc la protection des données qui passe notamment par le chiffrement pour conserver la confidentialité. Dans ce cas, on peut distinguer deux catégories de données : celles au repos contre celles en transit. Pour chacune des catégories, les méthodes de chiffrement diffèrent. Pour les données au repos, Hadoop dispose par défaut d'un chiffrement sur le système de fichiers HDFS, que toutes les distributions utilisent. De plus tous supportent le chiffrement pour le transfert de données au niveau HTTP avec SSL/TSL. Pour le reste, chacune des distributions dispose d'outils de gestion des clés, qui permettent de choisir un type de chiffrement pour un type de transit. Hortonworks possède son **Key Management Store**, quand Amazon déploie son **Key**

Management Service. Sur Cloudera, c'est **Navigator Key Trustee** qui fait l'affaire. Enfin, MapR et son système de tickets particulier n'a pas d'instance de gestion de clés, mais utilise les algorithmes AES et GCM.

Pour finir, la redondance des données ainsi que la mise en place de backups sont bien évidemment nécessaires pour tout environnement Big Data, et c'est d'autant plus important pour les solutions de type BDaaS. Les entrepôts de données sont de plus en plus imposants et leur sécurisation est primordiale, bien que les techniques restent pour cela restent classiques.

1.2.2. La sécurité grâce à l'analyse du Big Data

Dans le domaine de la sécurité, l'analyse statique et le machine learning sont utilisés depuis longtemps dans les systèmes de prévention et détection d'intrusions (IDPS), dans les gestionnaires d'évènement et d'information de sécurité (SIEMs) ou encore pour faire de l'analyse de logs. Mais l'augmentation extraordinairement rapide du volume des données a fait ressortir les limites de ces méthodes. On parle en effet de centaines de teraoctets, voir de pétaoctets pour les plus grands groupes. Les données sont récoltées depuis de plus en plus de sources, allant des équipements internes de sécurité qui tracent tous les échanges de données jusqu'aux bases de données de malwares et d'autres informations partagées par les autres entreprises en temps-réel. Ces données déstructurées et bruitées ne sont pas analysables efficacement par les outils classiques, qui retournent un trop grand nombre de faux-positifs ce qui se traduit par une masse trop importante d'alertes à gérer dans les SIEMs.
Grâce aux plateformes Big Data, il est donc désormais possible de gérer corréler toutes ces données et de centraliser leur gestion dans des SOC. La visualisation de ces données est également un domaine où le Big Data vient apporter plus de cohérence et de clarté, ce qui facilite la prise de décision. Les métriques sont en effet énormément utilisées dans ce genre d'écosystèmes, pour avoir un grand nombre de statistiques en continu sur l'état des systèmes. L'analyse en temps réel permet donc de passer d'une défense statique et réactive à une défense proactive et préventive, ce qui est un avantage de taille pour les entreprises.
Pour ce faire, des outils comme apache Storm sur Hortonworks permettent d'analyser de grands volumes de données grâce au principe des "spouts and bolts". Les spouts ou robinets ont pour rôle de récupérer certaines données dans la base pour créer un flux continu jusqu'à des bolts. Les bolts se chargent ensuite d'appliquer un certain traitement aux données et envoient le flux résultant en sortie. En définissant des rôles spécifiques à chaque spout et bolt, on peut alors créer des topologies complexes qui réaliseront des traitements complexes et en continu. Voici un exemple de topologie :



Apache Storm est par exemple utilisé dans la plateforme Apache Metron développé par Hortonworks en partenariat avec ManTech et B23. Cette plateforme fonctionne comme un SOC, elle centralise tous les journaux, les alertes, les métriques sur les données et Storm peut être utilisé pour faire de la threat intelligence en temps réel.
De nombreuses autres solutions ont émergé sur le marché, qui s'appuient en général sur le principe MapReduce, et qui ont donné lieu à une nouvelle génération de SIEMs, qui font converger plusieurs outils dans le but de proposer une vision complète de l'état du SI. Cela concerne entre autres l'EDR (Endpoint Detection and Response), l'analyse du trafic réseau (NTA) ou même par l'analyse comportementale des entités et utilisateurs (UEBA).
Comme SIEMs, on peut citer Fortscale, RSA security analytics, QRadar d'IBM, ou encore Splunk, mais ce sont loin d'être les seules solutions. Toutes permettent de faire de la détection de menaces avancées et persistentes (APT) et de plus en plus font de l'UEBA pour détecter des comportements anormaux en temps réel. Splunk ou Fortscale sont parmi les précurseurs dans le domaine mais depuis beaucoup d'autres ont suivi le pas et appliquent désormais ce genre d'analyse sur leurs plateformes. Il existe également des fournisseurs indépendants comme LightCyber, Interset ou Gurucul. En général, toutes les sociétés tendent à proposer des offres de plus en plus complètes combinant un maximum d'aspects présentés plus haut.

Pour finir, le principal défi qu'il reste aujourd'hui à réaliser est certainement d'ordre éthique. En effet, l'analyse comportementale des utilisateurs est la dernière nouveauté en matière d'analyse et l'on pourrait d'ores et déjà se poser la question de savoir si les données récoltées ne pourraient pas également servir à d'autres fins, comme la surveillance des employés selon leur activité au travail. On peut s'attendre à ce que certaines entreprises dépassent certaines limites tant la puissance de l'analyse peut sembler absolue.
Par ailleurs, les attaquants vont certainement trouver des contournements à ces nouvelles méthodes, comme cela a toujours été le cas. On pourrait imaginer des programmes malveillants qui "injectent" de fausses données afin de fausser les résultats d'analyse, et empêcher la détection de certaines activités ou attaques étant en cours.

II - Analyse de logs de connexions

2.1. Attaque DDOS d'un serveur web

Pour cette première expérience nous avons voulu nous mettre dans la situation d'une entreprise ou d'une organisation qui souhaite détecter et analyser les attaques par déni de service orchestrées sur son serveur web. Nous avons donc suivi le tutoriel "REFINE AND VISUALIZE SERVER LOG DATA" disponible sur la plateforme des tutoriels Hortonworks.

Dans un premier temps, il nous fallait télécharger un script python qui permettait de générer une base de logs de plusieurs milliers de lignes. Chaque ligne était selon le format suivant :

Date (timestamp) | IP | Pays | Statut (SUCCES ou ERROR)

Le script est conçu pour simuler le déclenchement d'une attaque DDOS à partir d'un certain moment, en ajoutant un grand nombre de logs de connexions provenant d'une multitude de lieux. Les données sont stockées dans un fichier nommé "eventlog-demo.log". Pour une date et heure donnée, le script va générer un nombre aléatoire de logs qui va dépendre du fait que l'attaque DDOS a été lancée ou non. Cette fourchette va varier entre 1 et 1000 dans les conditions "légales" puis entre 1 et 10000 lorsque l'attaque DDOS sera déclenchée.

Dans Hive view sur la plateforme Ambari, nous avons ensuite exécuté la requête suivante pour créer la table "firewall_logs" à partir des données générées par le script python :

```
CREATE TABLE FIREWALL_LOGS(time STRING, ip STRING, country STRING, status STRING)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
LOCATION '/var/log/eventlog-demo.log';
```

Notre table ainsi sauvée dans la base de données, il ne nous restait plus qu'à visualiser les résultats de deux manières différentes, l'une en utilisant Microsoft Excel 2016 et l'autre avec Zeppelin.

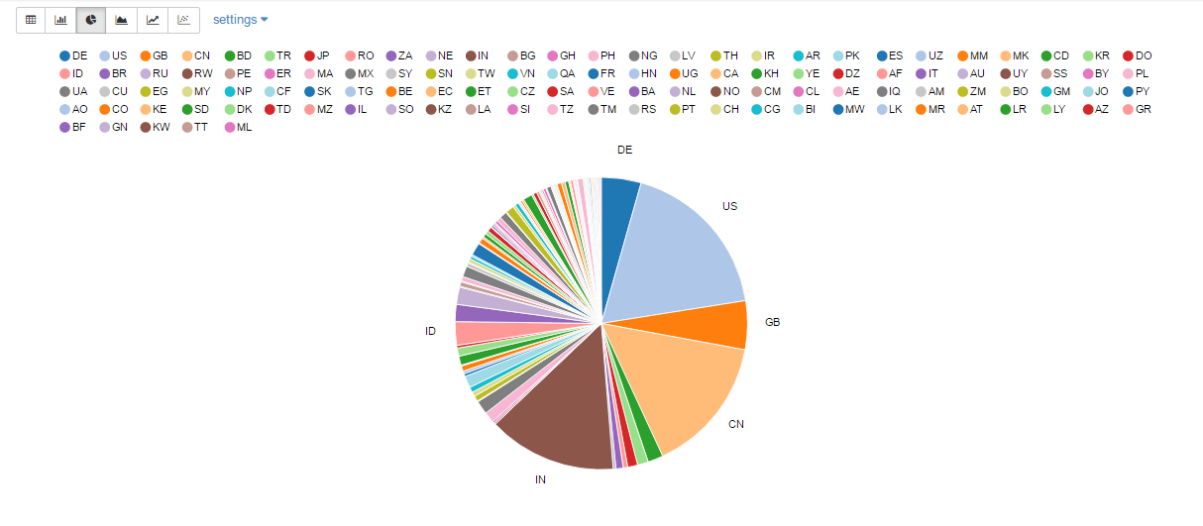
Pour importer les données dans Excel, il nous a fallu au préalable installer puis configurer le driver ODBC (Open DataBase Connectivity) d'Hortonworks. Pour ce faire, nous avons là encore suivi le tutoriel mis à disposition par Hortonworks. Nous avons configuré le service pour qu'il soit présent sur le port 10000 de notre sandbox. Dans Excel, nous sommes allés dans **Données->Données externes->Autres sources->Provenance: Microsoft Query**, avons sélectionné la base de donnée **Sample Hortonworks Hive DSN** disponible grâce à l'installation du driver ODBC. Une fois les données importées, nous avons utilisé l'outil PowerView d'excel en allant dans **Insertion->Carte 3D**. En ne conservant seulement les champs Pays et Statut, nous obtenions une carte interactive qui ressemblait à ceci :

Grâce à cette carte, on peut donc visualiser le trafic réseau de notre serveur et repérer en bleu toutes les connexions refusées, venant de sources non autorisées. En zoomant sur chaque lieu de connexion, on obtient l'adresse IP de la machine concernée ainsi que la date de connexion. On se rend compte très rapidement que l'attaque DDOS a probablement été orchestrée par un groupe organisé, qui a coordonné son attaque à partir de plusieurs pays. Dans un cadre professionnel, il aurait été simple de croiser ces données avec celles d'autres attaques subies antérieurement, ce qui aurait peut être permis de détecter un schéma s'attaque similaire par exemple (heure de déclenchement, adresses IP, lieux…).
A partir de ce moment, nous pourrions facilement extraire les adresses IP des sources non autorisées afin d'indiquer au pare-feu de refuser toute requête provenant de ces sources. Nous pourrions même, soit à partir d'un apprentissage antérieur, soit en utilisant une autre liste de données, éliminer les "faux-positifs" dans cette liste d'adresses à priori non autorisées, car il y a sûrement des connexions ayant été refusées simplement car le serveur était déjà trop encombré ou pour tout autre raison accidentelle.
La PowerView proposée par Excel est donc un outil puissant, simple d'utilisation et vraiment efficace lorsqu'il s'agit de visualiser des données géographiques.

Nous avons ensuite visualisé les données grâce à Zeppelin, auquel on accède par le port 9995 de notre machine virtuelle. Dans ce cas de figure, Zeppelin s'avère beaucoup moins puissant. En effet, il n'accepte que les requêtes ayant au plus 1000 résultats. Ici nous n'avons pas suivi le tutoriel car tout d'abord notre version de Zeppelin ne reconnaissait pas la commande "%hive", qui normalement veut dire que l'on écrit la requête en hiveQL, et puis car les visualisations proposées nous paraissaient un peu simplistes.

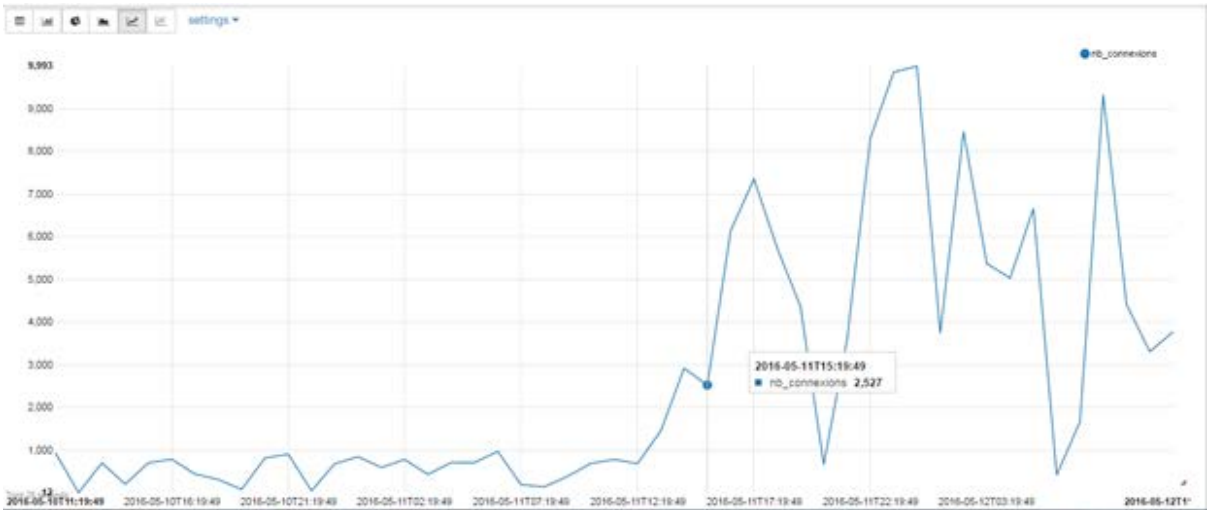
Par ailleurs, il est vrai que dans ce cas de figure Zeppelin offre une visualisation moins efficace que la carte d'Excel mais les graphiques restent très simples à générer, puisqu'il suffit d'une seule ligne de code (en sql) :

```
%sql select country, count(ip) from firewall_logs group by country, ip
```



Sur ce graphique on peut voir le nombre d'adresses ip différentes pour chaque pays. Là aussi on se rend compte assez vite que l'attaque vient d'une multitude de pays. On remarque aussi que la plupart des requêtes envoyées viennent de la Chine et de l'Inde, alors que ces pays n'ont aucune machine autorisée dans leurs frontières. Une étude approfondie des connexions provenant de ces deux pays pourrait peut-être être requise si nous voulions mener l'enquête.

```
%sql select time, count(ip) as nb_connexions from firewall_logs group by time order by time
```



Sur ce graphique, on peut voir l'évolution du nombre de connexions au cours du temps, et l'on remarque que l'attaque DDOS débute aux alentours du 11 mai 2016 à 15h19 (le serveur recense déjà plus de 2,500 demandes de connexions à ce moment là). Connaissant ces informations, on peut maintenant se concentrer sur les logs autour de cette heure, pour analyser les adresses IP ayant envoyé des requêtes, avant et pendant l'attaque.

Ce tutoriel nous a permis de mesurer la puissance d'analyse des services de Hadoop et de la plateforme Hortonworks.

2.2. Recherche de connexions suspectes au sein d'un réseau d'entreprise

Dans ce deuxième test nous nous sommes placé dans un écosystème d'entreprise et avons cherché à analyser les connexions des employés, dans le but de les analyser afin de repérer d'éventuels comportements suspects.

Nous sommes partis du script python de la première expérience et nous l'avons modifié pour que celui-ci nous génère des logs connexions avec des adresses ip de l'entreprise. Nous avons créé cinq services différents, contenant chacun 200 employés, et avec six plages horaires de travail différentes en tout. Ce script génère des connexions toutes les 40 secondes entre 7h et 10h, toutes les 8 minutes entre 21h et 5h et toutes les 13 minutes pour les autres plages horaires. Le format des données est comme suivi :

TIMESTAMP | IP | PAYS | ETAT

Nous avons ensuite créé un script python qui génère le profil des employés selon le format suivant :

nom | prénom | service | adresse ip | horaires de travail

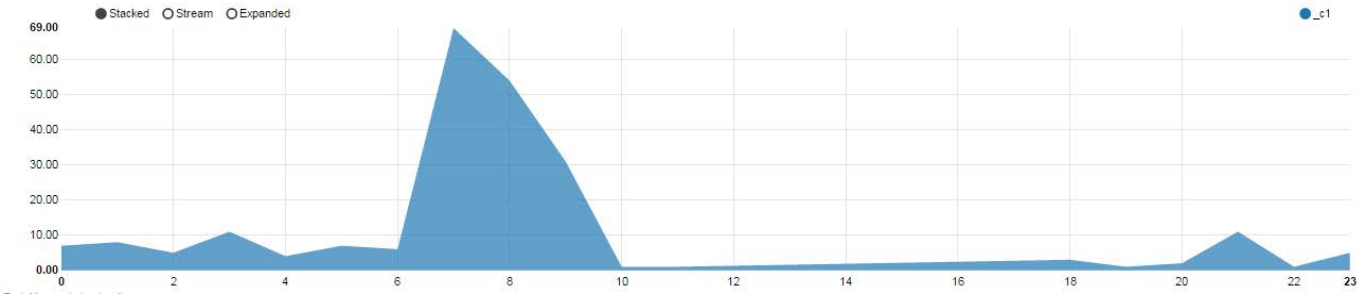
Nous avons également développé deux scripts bash pour supprimer les doubles des listes employés et supprimer les logs ne correspondant pas à des employés.

Dans Hive sur Ambari, nous avons donc créé trois tables : "eventlogs_entreprise", " employes_entreprise", contenant respectivement les logs de connexions sur le réseau de l'entreprise et la liste des employés, mais aussi une table vide "alert_detection", que nous avons rempli par la suite grâce à un script Pig. Elle est composée des attributs suivants : prenom, nom, addr_ip, hor_debut, hor_fin, heure_co, et etat. Ces tables devaient être stockées sous le format ORC (Optimized Row Columnar) afin de pouvoir être traitées par Pig. Voici le script Pig que nous avons écrit :

```
emp = LOAD 'employees_entreprise' using org.apache.hive.hcatalog.pig.HCatLoader;
logs = LOAD 'enventlogs_entreprise' using org.apache.hive.hcatalog.pig.HCatLoader;
emp_logs = join emp by addr_ip, logs by addr_ip;
res = foreach emp_logs {
       inf = (int) REGEX_EXTRACT($4, '(.*):(.*)', 1);
       sup = (int) REGEX_EXTRACT($4, '(.*):(.*)', 2);
       time = REGEX_EXTRACT($5, '(.*) (.*) (.*)', 3);
       heure = (int) REGEX_EXTRACT(time, '(.*):(.*):(.*)', 1);
generate prenom as prenom, nom as nom, $3 as addr_ip, inf as hor_debut:int, sup as
hor_fin:int, heure as heure_co:int, (
  CASE
    WHEN inf <= heure AND sup > heure THEN 'normal'
    WHEN inf > heure OR sup <= heure THEN 'suspect'
  END
) as etat; }
alert_detection = filter res by etat == 'suspect';
store alert_detection into 'alert_detection' using org.apache.hive.hcatalog.pig.HCatStorer;
```

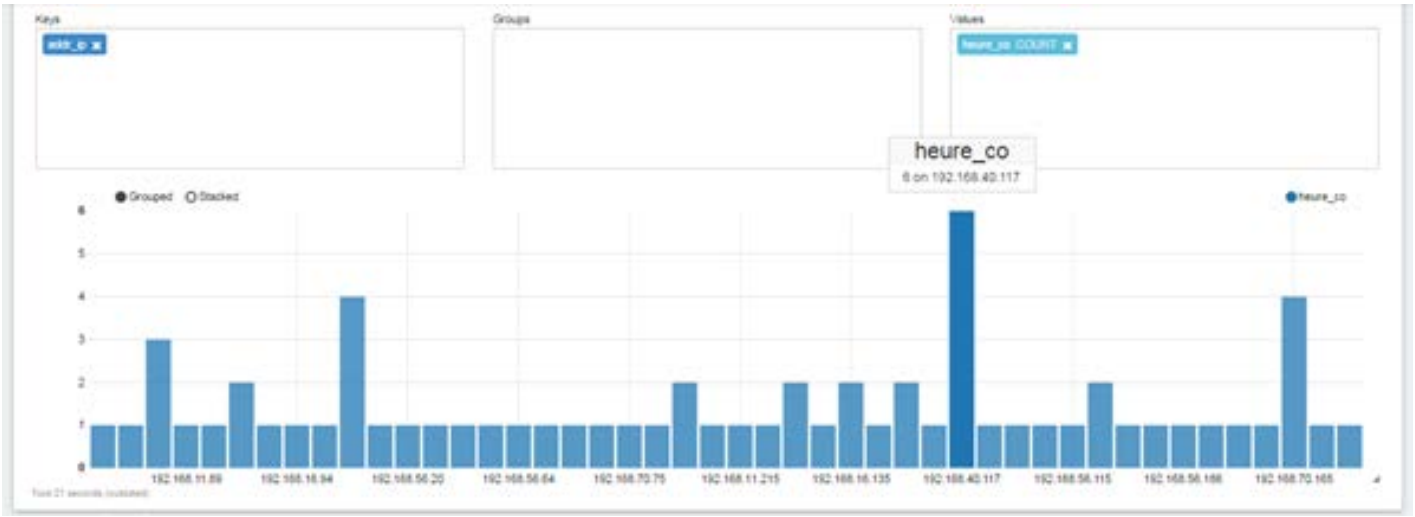
Nous voulions obtenir une table qui recensait chaque connexion suspecte de l'entreprise. Pour cela nous avons tout d'abord chargé nos tables grâce à la fonction **HCatLoader()** qui retrouve automatiquement la table dans la base de données. Puis nous avons fait une jointure entre les deux tables par l'attribut addr_ip. Ensuite, pour chaque log, nous avons extrait les horaires de début et fin de l'employé asoscié à la machine ainsi que l'heure de connexion. Si l'heure est hors de l'intervalle, la connexion est qualifiée de suspecte, sinon elle est normale. Enfin, nous avons filtré la table obtenue en ne conservant que les connexion suspectes, puis nous avons utilisé la fonction **HCatStorer()** pour stocker les résultats dans la table alert_detection.

Une fois cela fait, nous avons utilisé Zeppelin pour visualiser les données afin d'en tirer de l'information. Voici ce que nous avons obtenu :




```
%sql    select heure_co, count(*) from alert_detection group by heure_co
```

Ce graphique représente le nombre de connexions suspectes par heure de la journée. Les variations du nombre de connexions suspectes suivant les plages horaires correspondent à la différence des fréquences de génération de logs dans le script que nous avons écrit. Dans un cadre professionnel on aurait pu expliquer le nombre élevé de connexions suspectes entre 7h et 10h par le fait que les employés arrivent un peu en avance le matin, ou alors qu'ils ont des horaires flexibles et qu'ils n'arrivent pas forcément à l'heure indiquée dans la base de données. Cependant, les connexions suspectes durant la nuit mériteraient que l'on s'y attarde un peu plus.




```
%sql select addr_ip, heure_co from alert_detection where heure_co < 7 or heure_co > 20
```

Ce graphique montre le nombre de connexions suspectes entre 20h et 7h par adresse IP. On obtient finalement une quarantaine d'adresses qui présentent une menace relativement élevée, dont une en particulier : 192.168.40.117.

Au final, avec ces quelques lignes de code, nous sommes en mesure de détecter et repérer très rapidement des activités suspectes sur un réseau d'entreprises ainsi que les machines correspondantes. En utilisant Apache Storm et Apache Kafka, nous aurions pu faire cette analyse en temps réel. Malheureusement, nos ordinateurs respectifs n'avaient pas assez de mémoire vive pour supporter ce type de traitement de données, et nous n'avons pas pu trouver de solution alternative à temps. L'analyse en temps réel permettrait de créer un système d'alerte qui enverrait par exemple un sms au RSSI de l'entreprise dès qu'une activité suspecte serait détectée, voire qui déclencherait un mécanisme de défense automatiquement. Cela aurait donc permis d'augmenter considérablement la réactivité d'une entreprise face aux attaques. Nous aurions également pu faire du machine learning pour entrainer la machine et ainsi réduire le nombre de "faux-positifs" avec le temps, pour augmenter la fiabilité du système.

Conclusion

Le Big Data est aujourd'hui un élément clé du monde informatique, ces pratiques sont nombreuses puisqu'il permet de traiter une quantité astronomique de données dans des temps très courts. Le Big Data a ouvert de nouvelles portes dans le domaine de la sécurité informatique, on peut désormais analyser un grand ensemble de données dans le but d'extraire ou de trouver une ou plusieurs anomalies.

Le Big Data reste une technologie récente et propice à de nombreuses améliorations dans les années à venir. De nombreux progrès sont fait, et notamment dans son association avec le machine learning. Cette association permet de s'adapter rapidement à de nouvelles situations et de tenir compte de l'expérience du système, que ce soit pour détecter des intrusions dans un système d'information, pour détecter des comportements terroristes dans une gare, ou pour tout autre cas qui nécessite un système continuellement mis à jour.

Ce projet a été pour nous très enrichissant, le Big Data était une technologie que nous ne connaissions pas dans les détails et qui paraissait très vague. Nos différentes expériences et recherches nous ont permis de bien comprendre l'architecture d'un système Big Data et le rôle de chacun de ses composants. Nous nous sommes rendus compte de tout l'enjeu qu'il y avait derrière le Big Data autour de la sécurité pour l'avenir, car les attaques malveillantes n'ont pas fini de croître en nombre et en complexité. Nous avons pu réaliser deux expériences modestes, malgré les différents problèmes que nous avons rencontrés. Le principal était le fait que nos ordinateurs n'étaient pas toujours assez puissants, ce qui provoquait un dysfonctionnement de certains services. Ce projet, et principalement les tests effectués nous ont donné l'envie de continuer à s'intéresser au Big Data et à ses applications dans la sécurité.

Bibliographie

Pirmin Lemberger, Marc Batty, Médéric Morel, Jean-Luc Raffaëlli. Big Data et Machine Learning. 5 rue Laromiguière, 75005 Paris : Dunod, 2015. Management des systèmes d'information. 978-2-10-072074-3.

Jean-Charles Cointot, Yves Eychenne. La révolution Big Data. 5 rue Laromiguière, 75005 Paris : Dunod, 2014. Stratégie et management. 978-2-10-071142-0.

Christophe Brasseur. Enjeux et usages du Big Data, technologies, méthodes et mise en oeuvre. Paris : Lavoisier, 2013. Management et informatique. 978-2-7462-4520-4.

Adriano Girolamo PIAZZA, "NoSQL, Etat de l'art et benchmark", travail de Bachelor réalisé en vue de l'obtention du Bachelor HES présenté à la Haute École de Gestion de Genève (HEG-GE), Suisse, 2013.

Wikipedia. "Hadoop". In : Wikipédia [en ligne]. [Consulté le 12 avril 2016]. Disponible sur : https://fr.wikipedia.org/wiki/Hadoop.

HORTONWORKS.Hortonworks: Open and Connected Data Platforms [en ligne].[Consulté le 6 avril 2016]. Disponible sur : http://hortonworks.com/.

E. Ahlm et A. Litan, « Market Trends: User and Entity Behavior Analytics Expand Their Market Reach », 26-avr-2016. [En ligne]. Disponible sur: https://www.gartner.com/doc/reprints?id=1-370BP2V&ct=160518&st=sb.

Amazon, « Amazon Elastic Map Reduce ». Disponible sur : https://aws.amazon.com/fr/elasticmapreduce/.

A. Cárdenas, P. Manadhata, et S. Rajan, « Big Data Analytics for Security », *InfoQ*, février-2014. [En ligne]. Disponible sur: https://www.infoq.com/articles/bigdata-analytics-for-security.

A. Cesir, « ENABLING KERBEROS ON HDP AND INTEGRATING WITH ACTIVE DIRECTORY », *Hortonworks*, 07-nov-2014. [En ligne]. Disponible sur: http://fr.hortonworks.com/blog/enabling-kerberos-hdp-active-directory-integration/.

Cloudera, « Apache Sentry – The Open Standard for Unified Authorization in Hadoop », *Cloudera*, 26-janv-2015. [En ligne]. Disponible sur: https://vision.cloudera.com/apache-sentry-the-open-standard-for-unified-authorization-in-hadoop/.

C. Green, « Big security: big data and the end of SIEM », *Information Age*, 29-mai-2014. [En ligne]. Disponible sur: http://www.information-age.com/technology/security/123458055/big-security-big-data-and-end-siem.

Hortonworks, « SOLVING APACHE™ HADOOP® SECURITY ». Février-2016. [En ligne]. Disponible sur : http://info.hortonworks.com/rs/549-QAL-086/images/Security-White-Paper.pdf.

Hortonworks, « HOW STORM WORKS », *Hortonworks*. [En ligne]. Disponible sur: http://hortonworks.com/apache/storm/#section_2.

MapR, « MapR 5.1 Documentation ». [En ligne]. Disponible sur : http://maprdocs.mapr.com/51/index.html#c_51_home.html.

V. Marchive, « Fortscale combine Big Data et Machine Learning pour traquer les comportements anormaux », *LeMagIT*, 23-avr-2016. [En ligne]. Disponible sur: http://www.lemagit.fr/actualites/4500244892/Fortscale-combine-Big-Data-et-Machine-Learning-pour-traquer-les-comportements-anormaux.

J. Myerson, « Machine learning applications: Mitigating the risks », *TechTarget*, Avril-2016. [En ligne]. Disponible sur: http://searchsecurity.techtarget.com/tip/Machine-learning-applications-Mitigating-the-risks.

Oracle, « Oracle Audit Vault and Database Firewall ». Avril-2014. [En ligne] Disponible sur :

J. Paoli, « CYBERSÉCURITÉ : EN 2015, LES TENTATIVES D'ATTAQUES ONT DOUBLÉ, PASSANT DE 4,2 À 8,19 MILLIARDS », *Solutions numériques*, 23-févr-2016. [En ligne]. Disponible sur: http://www.solutions-numeriques.com/en-2015-les-tentatives-dattaques-ont-double-de-42-a-819-milliards/.

Y. Serra, « Hortonworks met plus de sécurité, de gouvernance et de Cloud dans Hadoop », *LeMagIT*, 19-avr-2016. [En ligne]. Disponible sur: http://www.lemagit.fr/actualites/450281607/Hortonworks-met-plus-de-securite-de-gouvernance-et-de-Cloud-dans-Hadoop.

S. Shea, « CSA top 10 big data security, privacy challenges and how to solve them », *TechTarget*, 11-avr-2013. [En ligne]. Disponible sur: http://searchcloudsecurity.techtarget.com/photostory/2240208464/CSA-top-10-big-data-security-privacy-challenges-and-how-to-solve-them/1/Big-data-security-big-data-privacy-issues-big-decisions.

J. Sirota, « LEVERAGING BIG DATA FOR SECURITY ANALYTICS », *Hortonworks*, 17-nov-2015. [En ligne]. Disponible sur: http://hortonworks.com/blog/leveraging-big-data-for-security-analytics/.

K. Smith, « Big Data Security: The Evolution of Hadoop's Security Model », *InfoQ*, 14-août-2013. [En ligne]. Disponible sur: https://www.infoq.com/articles/HadoopSecurityModel.

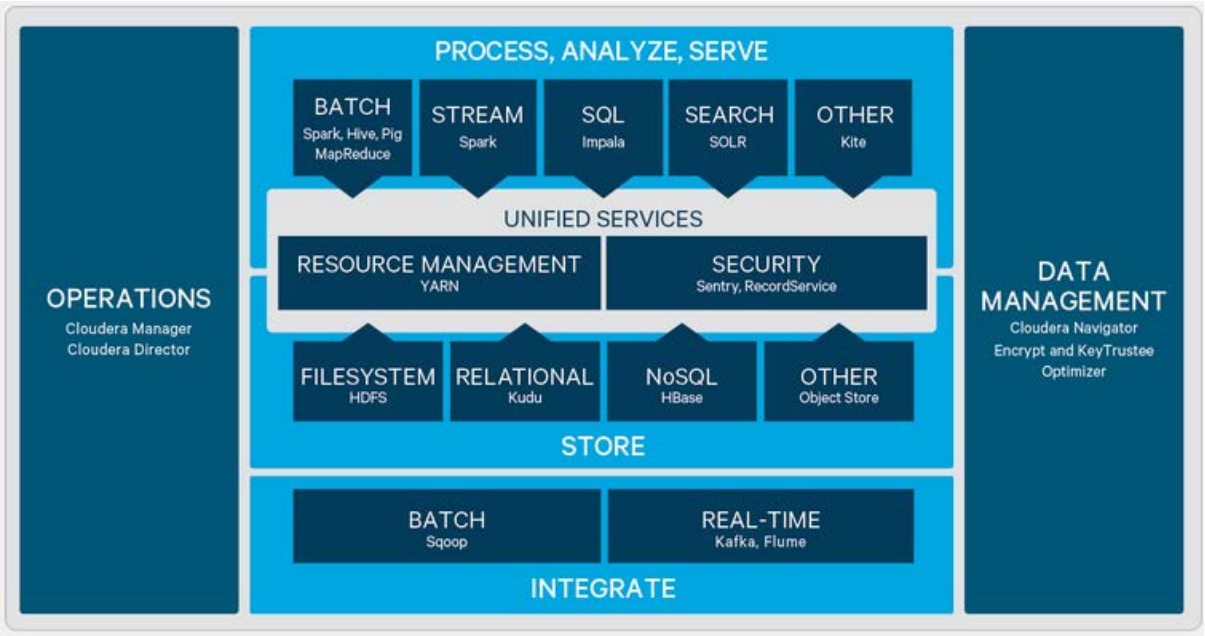
D. Sullivan, « Introduction to big data security analytics in the enterprise », *TechTarget*, nov-2015. [En ligne]. Disponible sur: http://searchsecurity.techtarget.com/feature/Introduction-to-big-data-security-analytics-in-the-enterprise

D. Sullivan, « Comparing the top big data security analytics tools », *TechTarget*, Février-2016. [En ligne]. Disponible sur: http://searchsecurity.techtarget.com/feature/Comparing-the-top-big-data-security-analytics-tools.

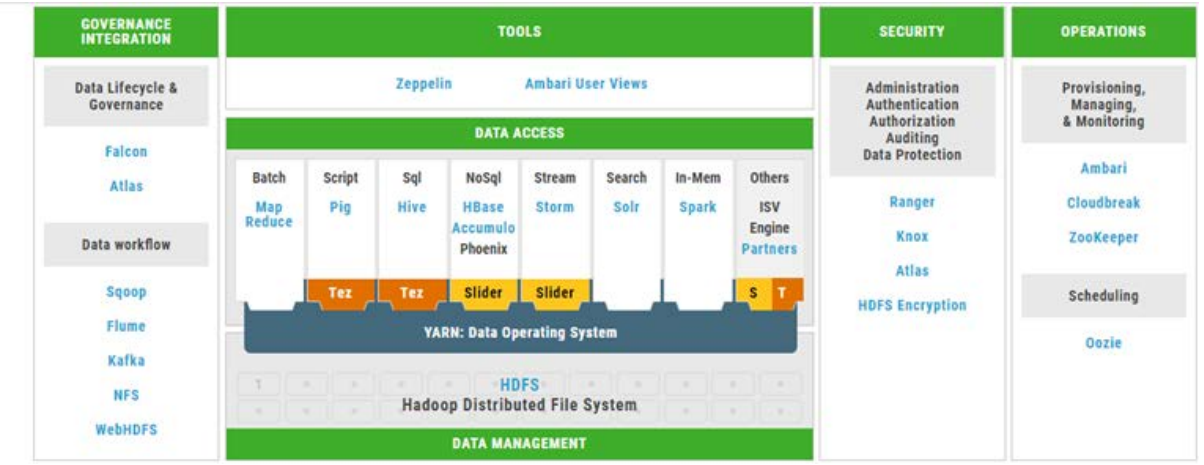
Cloudera, « Securing Your Enterprise Hadoop Ecosystem ». 2016. [En ligne]. Disponible sur : https://www.cloudera.com/content/dam/www/static/documents/whitepapers/securing-your-enterprise-hadoop-ecosystem.pdf.

Annexes

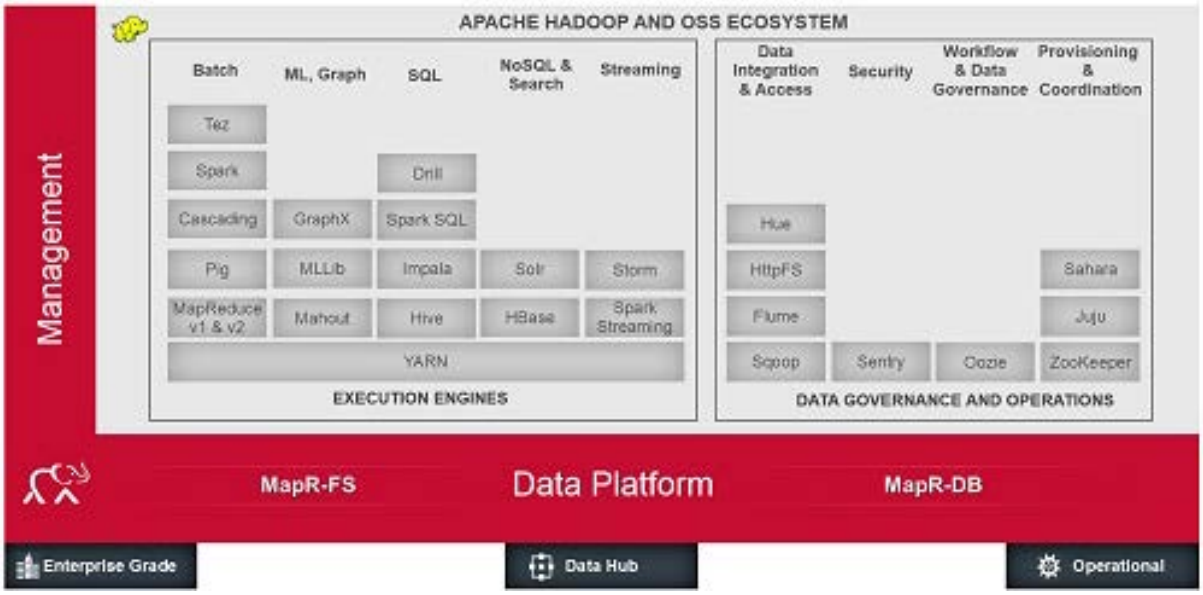
Annexe 1 : Architecture Cloudera



Annexe 2 : Architecture Hortonworks Data Platform (HDP)



Annexe 3 : Architecture MapR



Annexe 4 : Script python générant les connexions

```
#!/usr/bin/python
#
# Generate log files
# Log will be of the format :
# date : (SUCCESS|ERROR) : COUNTRY : IP

from random import randrange
import random
import logging
from optparse import OptionParser
from time import strftime
from datetime import timedelta, date, time, datetime

def read_options():
    parser = OptionParser()
    parser.add_option("-t", "--time", dest="duration", help="Generate logs for X Hours. Default=48H ( 2 days)", default="48", type="int")
    parser.add_option("-l", "--legit-connection", dest="legal", help="How many legal users are connecting. Default=1000", default="1000", type="int")
    parser.add_option("-s", "--start-ddos", dest="start", help="Start DDoS after X hours. Default=24H", default="24", type="int")
    parser.add_option("-d", "--ddos-connection", dest="ddos", help="How many DDoS connection. Default=10000", default="10000", type="int")
    parser.add_option("-i", "--increment", dest="increment", help="Generate logs every X minutes. Default=60min", default="60", type="int")
    parser.add_option("-f", "--logfile", dest="logfile", help="Specify a log file. Default=/var/log/eventlog-demo.log", default="/var/log/eventlog-demo.log", type="string")
    (options, args) = parser.parse_args()
    return options

def timestamp():
    timestamp = strftime("%Y-%m-%dT%H:%M:%S")
```

```
    print timestamp
    return timestamp


def generate_log(timestamp,users,status,logfile):
    logging.basicConfig(filename=logfile,
                        format='%(message)s',
                        level=logging.DEBUG)
    if status == 'SUCCESS':
        countries = weight_good_country(users)
    else :
        countries = weight_bad_country(users)
    for country, concurrent_user in countries.iteritems():
        i = 0
        while i < concurrent_user :
            logging.info('%s|%s|%s|%s'%(timestamp, random_ip(), country, status))
            i += 1

def WeightedPick(d):
    r = random.uniform(0, sum(d.itervalues()))
    s = 0.0
    for k, w in d.iteritems():
        s += w
        if r < s: return k
    return k

def weight_good_country(user):
    countries = {'US':60,'GB':20,'DE':15,'FR':5}
    results = {}
    for x in xrange(user):
        p = WeightedPick(countries)
        results[p] = results.get(p, 0) + 1
    return results

def weight_bad_country(user):
    countries     =     {'CN':1330044000,'IN':1173108018,     'US':610232863,'ID':242968342,
'BR':201103330,'PK':184404791,                              'BD':156118464,'NG':154000000,
'RU':140702000,'JP':127288000,  'MX':112468855,'PH':99900177,  'VN':89571130,'ET':88013491,
'DE':81802257,'EG':80471869,    'TR':77804122,'IR':76923300,    'CD':70916439,'TH':67089500,
'FR':64768389,'GB':62348447,    'IT':60340328,'MM':53414374,    'ZA':49000000,'KR':48422644,
'ES':46505963,'UA':45415596,    'CO':44205293,'TZ':41892895,    'AR':41343201,'KE':40046566,
'PL':38500000,'SD':35000000,    'DZ':34586184,'CA':33679000,    'UG':33398682,'MA':31627428,
'PE':29907003,'IQ':29671605,    'AF':29121286,'NP':28951852,    'MY':28274729,'UZ':27865738,
'VE':27223228,'SA':25731776,    'GH':24339838,'YE':23495361,    'KP':22912177,'TW':22894384,
'SY':22198110,'MZ':22061451,    'RO':21959278,'AU':21515754,    'LK':21513990,'MG':21281844,
'CI':21058798,'CM':19294149,    'CL':16746491,'NL':16645000,    'BF':16241811,'NE':15878271,
'MW':15447500,'KZ':15340000,    'EC':14790608,'KH':14453680,    'ML':13796354,'GT':13550440,
'ZM':13460305,'AO':13068161,    'SN':12323252,'ZW':11651858,    'CU':11423000,'RW':11055976,
'GR':11000000,'CS':10829175,    'PT':10676000,'TN':10589025,    'TD':10543464,'CZ':10476000,
'BE':10403000,'GN':10324025,     'SO':10112453,'BO':9947418,     'HU':9930000,'BI':9863117,
'DO':9823821,'BY':9685000,       'HT':9648924,'BJ':9056010,      'SE':9045000,'AZ':8303512,
'SS':8260490,'AT':8205000,       'HN':7989415,'CH':7581000,      'TJ':7487489,'IL':7353985,
'RS':7344847,'BG':7148785,       'RS':7344847,'BG':7148785,      'HK':6898686,'TG':6587239,
'LY':6461454,'JO':6407085,       'PY':6375830,'LA':6368162,      'PG':6064515,'SV':6052064,
'NI':5995928,'ER':5792984,       'KG':5508626,'DK':5484000,      'SK':5455000,'SL':5245695,
'FI':5244000,'NO':5009150,       'AE':4975593,'TM':4940916,      'CF':4844927,'SG':4701069,
```

```
'GE':4630000,'IE':4622917,      'BA':4590000,'CR':4516220,      'HR':4491000,'MD':4324000,
'NZ':4252277,'LB':4125247,      'PR':3916632,'PS':3800000,      'LR':3685076,'LT':3565000,
'UY':3477000,'PA':3410676,      'MR':3205060,'MN':3086918,      'CG':3039126,'AL':2986952,
'AM':2968000,'OM':2967717,      'JM':2847232,'KW':2789132,      'LV':2217969,'NA':2128471,
'MK':2061000,'BW':2029307,      'SI':2007000,'LS':1919552,      'XK':1800000,'GM':1593256,
'GW':1565126,'GA':1545255,      'SZ':1354051,'MU':1294104,      'EE':1291170,'TT':1228691,
'TL':1154625,'CY':1102677,       'GQ':1014999,'FJ':875983,       'QA':840926,'RE':776948,
'KM':773407,'GY':748486,         'DJ':740528,'BH':738004,        'BT':699847,'ME':666730,
'SB':559198,'CV':508659,         'LU':497538,'SR':492829,        'MO':449198,'GP':443000,
'MQ':432900,'MT':403000,         'MV':395650,'BN':395027,        'BZ':314522,'IS':308910,
'BS':301790,'BB':285653,         'EH':273008,'PF':270485,        'VU':221552,'NC':216494,
'GF':195506,'WS':192001,         'ST':175808,'LC':160922,        'GU':159358,'YT':159042,
'CW':141766,'AN':136197,         'TO':122580,'VI':108708,        'GD':107818,'FM':107708,
'VC':104217,'KI':92533,           'JE':90812,'SC':88340,          'AG':86754,'AD':84000,
'IM':75049,'DM':72813, 'AW':71566,'MH':65859, 'BM':65365,'GG':65228, 'AS':57881,'GL':56375,
'MP':53883,'KN':49898, 'FO':48228,'KY':44270, 'SX':37429,'MF':35925, 'LI':35000,'MC':32965,
'SM':31477,'GI':27884, 'AX':26711,'VG':21730, 'CK':21388,'TC':20556, 'PW':19907,'BQ':18012,
'WF':16025,'AI':13254,  'TV':10472,'NR':10065,  'MS':9341,'BL':8450,  'SH':7460,'PM':7012,
'IO':4000,'FK':2638,    'SJ':2550,'NU':2166,    'NF':1828,'CX':1500,    'TK':1466,'VA':921,
'CC':628,'TF':140, 'PN':46,'GS':30 }
    results = {}
    for x in xrange(user):
        p = WeightedPick(countries)
        results[p] = results.get(p, 0) + 1
    return results

def random_ip():
    not_valid = [10,127,169,172,192]

    first = randrange(1,256)
    while first in not_valid:
        first = randrange(1,256)

    ip = ".".join([str(first),str(randrange(1,256)),
    str(randrange(1,256)),str(randrange(1,256))])
    return ip

def concurrent_connection(users):
    return randrange(1,users)

def main():
    options = read_options()
    current_time = datetime.now()-timedelta(hours=options.duration)
    attack_time = current_time + timedelta(hours=options.start)
    print current_time.strftime("%Y-%m-%dT%H:%M:%S")
    while  datetime.now() > current_time:
        generate_log(current_time.strftime("%Y-%m-
%dT%H:%M:%S"),concurrent_connection(options.legal),"SUCCESS",options.logfile)
        if attack_time < current_time:
          generate_log(current_time.strftime("%Y-%m-
%dT%H:%M:%S"),concurrent_connection(options.ddos),"ERROR",options.logfile)
        current_time = current_time+timedelta(minutes=(options.increment))

if __name__ == "__main__":
    main()
```

Annexe 5 : Script python générant la liste des employés

```
#
# coding: utf8
#
# Generate employes file
# File will be of the format :
# first_name | last_name | IP
#

import random

def generate_employe(name_serv, working_h, third):

        with open("employe_win.txt", "a") as fich:
                for i in range (0, 200):
                      j = int(random.uniform(0,3))
                      fich.write(random_prenom()  +  '|'  +  random_patronyme()  +  '|'  +
name_serv + '|' + random_ip(third) + '|' + working_h[j] + '\n')

def random_prenom():

        with open("Prenoms.txt", "r") as f:
                x = random.randrange(0, 185)
                lines = f.readlines()
                prenom = unicode(lines[x].replace('\n', ''))

                f.close()

        return prenom

def random_patronyme():
        with open("Patronymes.txt", "r") as f:
                x = random.randrange(0, 400)
                lines = f.readlines()
                patronyme = unicode(lines[x].replace('\n', ''))

                f.close()

        return patronyme


def random_ip(third):
        ip = ".".join([unicode(192),unicode(168),
       unicode(third),unicode(random.randrange(1,256))])

        return ip

def generate_first():
    not_valid = [10,127,169,172,192]
    first = random.randrange(1,256)
```

```
    while first in not_valid:
        first = random.randrange(1,256)

    return first

def main():
    thirds = [11, 16, 40, 56, 70]
    name_service = ["Compta", "Informatique", "R&D", "RH", "Securite"]
    for i in range(0,5):
        if i == 4:
                working_hours = ["21:5", "5:13", "13:21"]
        else:
                working_hours = ["8:18", "9:19", "10:20"]
        generate_employe(name_service[i], working_hours, thirds[i])

if __name__ == "__main__":
    main()
```

Annexe 6 : Script bash supprimant les doubles dans la liste des employés

```
#!/bin/bash
nbLigne=$(wc -l employe_win.txt | cut -d " " -f 1)
i=1
echo $nbLigne
while read ligne
do
    adresse=$(echo $ligne | cut -d \| -f 4)
    echo $adresse
    grep $adresse employe_win.txt > temp
    cat temp
    sed -i "/$adresse/d" employe_win.txt
    head -n 1 temp >> employe.txt
done < employe_win.txt
```

Annexe 7 : Script bash supprimant les logs ne correspondant pas à des employés

```
#!/bin/bash
while read ligne
do
    adresse=$(echo $ligne | cut -d \| -f 2)
    grep $adresse employe.txt > temp
    if [ $(wc -l temp | cut -d " " -f 1) == 0 ] ; then
        sed -i "/$adresse/d" eventlog_entreprise.txt
    fi
done < eventlog_entreprise.txt
```