hurence

White Paper “Big Data series” # 1

Vers une plateforme Big Data nouvelle génération

19 Mars 2021

Remerciements

La rédaction de ce white paper n'aurait pas pu être possible sans la contribution exceptionnelle des membres de l'équipe d'architectes de Hurence. Mes remerciements vont donc à Thomas Bailet, ouvreur de voies en escalade comme en IT, spécialiste des infrastructures orientées industrie du futur et à l'initiative du Data Historian utilisé chez un nombre croissant de clients. Enrico Mano qui réalise avec passion (la passion d'un italien) nos architectures Saas à base de Kubernetes. Mathieu Rossignol qui est spécialiste de la sécurisation de ces infrastructures pour la tranquillité de nos clients et un joueur de tennis qui a dû interrompre sa carrière mais qui revient progressivement en forme sur les cours. Jérôme Arnou qui gère les déploiements et les migrations chez nos clients et donc sur le terrain (aussi sur le terrain de tennis) vérifie la qualité de nos propositions. Michael Soubra, vrai secouriste toujours prêt, pour les clusters Kubernetes de nos clients. Et sans tous les nommer merci à tous les membres de l'équipe. Par leur investissement et leur compétence, ils font de cette entreprise un véritable petit joyau où il fait bon vivre et bâtir, bâtir notre futur comme celui de nos clients appréciés.

Un autre chaleureux remerciement pour Pascal Roux de l'IFPEN, qui nous fait une belle confiance lorsque nous apportons nos évolutions dans ses clusters Big Data. Pour lui et les équipes de l'IFPEN nous voulons le meilleur. J'ai pensé à toi Pascal en rédigeant beaucoup de ces lignes.

Enfin un remerciement aux membres de l'équipe Hortonworks qui, avec leur stratégie open source ont permis à des petits acteurs comme Hurence et ses clients d'accéder au Big Data. Nous essayons à travers ce white paper de prendre modestement leur relais.

Table des matières

1. Introduction

Depuis 11 ans maintenant Hurence conçoit des infrastructures Big Data et de plus en plus, Big Data / IA / IoT / Blockchains mais pour faire court on dira “Big Data” pour englober tous ces sujets. Un peu précurseur et prescripteur en France, la société accompagne toujours nombre de clients dans les premiers pas d’une aventure Big Data ou dans son évolution (montées en version) ou dans ses migrations (dans le cloud Azure, Google ou Amazon). Nous bâtissons, détruisons, reconstruisons des infrastructures Big Data depuis 11 ans. C’est notre quotidien.

Nous avons eu envie à travers ce white paper de partager nos dernières aventures sur le sujet car beaucoup de nos clients et la communauté des utilisateurs d’infrastructures Big Data dans son ensemble se posent des questions sur ses prochaines étapes. Tous ont une infrastructure et veulent la faire évoluer et se posent la question de ce qu’il convient de faire, conscients que nous sommes un peu à un tournant.

Nous espérons que ce White paper vous aidera à choisir la meilleure voie pour votre contexte, sachant qu’il n’y a pas de bonne ou de mauvaise voie mais celle qui correspond à votre entreprise qu'elle soit de grande taille ou de petite taille.

De par sa nature, Hurence peut d’ailleurs présenter le biais de favoriser les petites entreprises. C’est dans notre ADN: nous essayons de faire accéder les petites structures à l’excellence du Big Data à des coûts possibles. Une prise de risque plus grande sera donc peut être acceptée par ces entreprises notamment pour s’affranchir de quelques contrats de supports coûteux. Donc vous devez garder ceci en tête si vous êtes le DSI d’un groupe du CAC 40 sachant que pour tout client, Hurence n’a jamais fait de compromission sur la sécurité des données et traitements.

2. Le Big Data entre 2008 et 2020: un peu d’histoire

L’histoire permet toujours de comprendre pourquoi on en arrive là où on en arrive. Pour le Big Data, comme pour l’histoire de l’humanité, il y a eu plusieurs ères. De la période Néandertal à la révolution industrielle...

Une infrastructure Big Data dans les années 2008/2011 était relativement simple en proportion de ce qu'elle est aujourd’hui. Basée uniquement sur Hadoop, la ré-implémentation des technologies de Google par Yahoo!, mises à disposition dans le monde Apache en 2008, nous avions un HDFS (Highly Distributed File System) pour stocker des fichiers sous forme de blocs, la base de données HBase (base de données orientée colonnes), du MapReduce, le célèbre algorithme pour traiter les données en parallèle et des configurations manuelles dans des fichiers XML. C’était notre ère Néandertal que l’on nommera l’ère **NoSQL**.

Mais rapidement les clients ont désiré faire du SQL (donc arrivée de Hive, du SQL distribué en MapReduce). Ils ont réclamé à raison de la sécurité, de l’audit et les outils se sont multipliés, venant du monde Apache, pas toujours cohérents les uns avec les autres. Les distributions Hadoop ont réalisé ce premier niveau de cohérence d’intégrations de ces nombreux outils Apache (les SolR, les Nifi, les Atlas, les Sentry/Ranger, les Sparks, Hive/Impala SQL et autres) et les performances avec le in-memory se sont améliorées sur des petits volumes de données. Sont apparues des consoles d’administration et de gestion comme les Cloudera Manager et les Ambari et les sécurisations nécessaires basées sur Kerberos et du ActiveDirectory. Le Big Data entrait dans son ère Homme moderne et **Enterprise ready**.

Mais... cela ne suffisait pas car chaque client voulait en sus son moteur de recherche Elasticsearch (le plus populaire), sa base graphe Neo4J, son infrastructure d’IA à base plutôt de Dask et que sais-je encore selon son métier… Et l’on se retrouvait à gérer de nombreux clusters dans des consoles d’administration différentes ou pire à colocaliser des services qui ne devaient pas l’être, faute de budgets serveurs au détriment d’une certaine stabilité des clusters. Les pires étant data scientists que l’on adore mais qui ont l’art de dégommer un cluster d’expérimentation en lançant leurs travaux de calculs de modèles d’IA sur Dask et en prenant toutes les ressources disponibles et non disponibles, asséchant en mémoire les clusters co-localisés et échappant en sus à la sécurisation Kerberos honnie. Le Big Data entrait dans son ère Guerres de religions ou l’ “**Enterprise chaos**” amplifié par la multiplications des initiatives dans le cloud ou chacun veut faire son expérience et découvre qu’il peut instancier sa technologie favorite en mode “Service managé” puis reculer quand il découvre - ou son management - sa facture à l’usage.

Chacun ayant maintenant expérimenté, on sait qu'il est illusoire de penser que l'infrastructure Big Data rassemblera tous les utilisateurs autour d'une seule et même technologie fut elle une superbe technologie comme Spark. Et même pour une technologie, on sait que tous les projets n'avancent pas à la même vitesse et qu'il faut gérer proprement plusieurs versions de certaines technologies, comme Spark.

Enfin les besoins en I(I)oT et en intégration digitale des entreprises ont généré des besoins pour de nouvelles technologies de captation, de stockage de données et d'analyses de séries temporelles et de **blockchains**. Voici l'ère du **Big Bang**. Du chaos, incluant la disparition de certains acteurs, émerge et se solidifie un tout nouvel univers. Ainsi est le cycle de la vie et le Big Data n'y échappe pas...

Aujourd'hui ce que l'on veut mettre dans une infrastructure Big Data nouvelle génération est très large ; ce doit être un catalogue **extensible facilement**, **évolutif**, **agile**, rendu **cohérent**, doit assurer **la bonne isolation des ressources**, doit être bien **sécurisé sans trop de contraintes**, et doit revisiter aussi ce qui est devenu obsolète. Nous allons voir que nous allons être aidés en cela par une belle technologie : **Kubernetes**.

Quelles solutions pour monter son infrastructure Big Data en 2021

Le panorama a soudainement changé sur le Big Data depuis disons 2 ans. Longtemps basées sur Hadoop et ses diverses distributions (Hortonworks Data Platform, Cloudera et autres MapR), notre stratégie doit maintenant être revisitée pour les raisons techniques évoquées précédemment mais aussi pour des raisons commerciales.

Nombre de ces distributions Big Data que nous qualifierons de premières générations ont disparu ou ont été rachetées : la MapR a été acquise par Hewlett Packard, puis récemment à travers le rachat par Cloudera la Hortonworks Data Platform (HDP). Seule résiste la CDP (Cloudera Data Platform) nouveau nom pour l'extension de la distribution pour le cloud, le "on premise" et l'hybride, mais sur le modèle business assez traditionnel d'un éditeur avec des coûts non négligeables. On parle de 10 000 dollars par nœud annuel et d'une partie variable si l'on dépasse les 16 cœurs et 128 gigas par nœud ou les 48 téraoctets de disque dans le cluster. Considérant qu'un cluster "Hadoop" a tellement d'administration que déjà 3 ou 4 nœuds sont nécessaires pour ne rien calculer, les clusters à plus de 10-12 nœuds sont classiques et la note vraiment salée pour une TPE, PME.

Heureusement, Hurence était là (et est toujours là) pour concevoir les mêmes infrastructures sans ces coûts, initialement à base de la distribution Hortonworks et aujourd'hui avec une toute nouvelle approche. Car nous n'avons plus le choix et nous avons dû innover au pied du mur.

En effet, les clients qui ont encore de vieilles Hortonworks HDP (qui marchent parfaitement pour leur usage en production) ne peuvent plus faire "scaler" ces infrastructures sans avoir une forme d'accord commercial avec Cloudera. Et les versions longtemps gratuites de Cloudera ne peuvent plus, elles non plus, scaler au-delà d'un certain nombre de nœuds.

Il en est donc fini de la belle époque où l'on pouvait donner accès à une distribution Hadoop "packagée" à des petits acteurs sans coûts autres que les serveurs, notre installation et notre gestion au quotidien des clusters de machines, sachant que les composants packagés étaient gérés par une communauté active Apache ce qui assurait une certaine qualité et un besoin en support assez minimaliste.

Alors on a fait le deuil de Hadoop, et finalement comme dans toute expérience de deuil on va vers une vie quelque fois meilleure que celle qui précédait le deuil. Une vie de plus grande liberté et d'indépendance pour nous et nos clients.

3. Qu'est ce qu'une infrastructure Big Data en 2021

Comme mentionné précédemment, une infrastructure Big Data en 2021 est tout ce qui permet de stocker et traiter des données en volumétries, en temps réel et batch, et cela couvre, sans que la liste ne soit exhaustive, les aspects stockages distribués, traitements parallèles temps réel ou non, SQL ou non, Intelligence artificielle (en python ou non), IoT (gateways pour capteurs), blockchains et moteurs de recherche. Bien entendu avec les catalogues de données et la

gestion des droits d'accès sur les données et services, l'auditing, la traçabilité et le monitoring. Enfin les aspects devops, dataops et les aspects CI/CD (continous integration et continuous deployment) font aussi maintenant partie de l'équation.

En réalité ce qu'on a fait pour nous et pour maintenant nos clients c'est choisir les meilleures technologies et de faire en sorte de les packager sur un socle commun et avec une sécurité commune.

Dès lors nous avons dû faire des choix car "être libre c'est choisir et faire les deuils de ce qui ne peut plus être pour aller, souvent, vers un meilleur":

- Le choix de s'affranchir des anciennes technologies considérées incontournables, par exemple HDFS, mais qui avaient des défauts notamment sur le traitement sur des petites volumétries au profit d'autres plus universelles, quitte à dans certains cas, accepter, à la marge, une légère perte en performances.
- Le choix de s'écarter des technologies dont la communauté ne nous semblait plus assez solide pour porter notre stratégie en l'absence d'acteurs qui les soutenait dans l'esprit d'un positionnement vraiment open source.
- Les choix de ne s'appuyer que sur les outils en licences libres, si possible Apache en priorité et en version communautaire uniquement (pas de solutions d'un éditeur) pour se mettre à l'abri de rachats ou de changements de stratégies. De notre point de vue d'ailleurs le "vrai" logiciel open source est par nature gratuit et sans limitation d'usage. Les logiciels en open source pour effet marketing, ne nous intéressent pas dans notre stratégie.
- Le choix de prendre des produits bien finis si possible dans leurs interfaces utilisateur car malgré les efforts des éditeurs de Hadoop, les interfaces n'étaient pas toujours de belle facture. Finalement on préfèrera, aussi par compromis, avoir de belles interfaces même si tout n'est pas intégré dans une seule application.
- Le choix d'avoir toujours des alternatives possibles dans les outils et dans leur assemblage grâce aux standards. Si notre choix actuel de Système de Fichier distribué doit évoluer alors cela ne doit pas remettre en cause les développements réalisés. Même chose pour le moteur SQL distribué.
- Le choix de privilégier des technologies ayant de larges catalogues de connecteurs (d'intégrations) avec les outils phares du Big Data (donc sélectionnés de facto).
- Le choix de sélectionner des technologies dont la sécurisation est possible via une seule technologie. La sécurisation via Kerberos a été un véritable frein à l'usage des infras, à leur démocratisation chez certaines communautés. Tant qu'on y était, autant essayer de résoudre ce problème.
- Le choix de s'affranchir de la co-localisation des données et des traitements car avec la multiplication des technologies cette promesse ne peut plus être tenue. De plus, l'amélioration des aspects in-memory et le fait que la plupart des traitements des clients, ne sont, en réalité, pas du Big Data (plus de 90% d'entre eux, voire 100%) font que cette contrainte, déjà levée dans le cloud, peut l'être on premises aussi. Sur nos volumétries on peut si nécessaire compenser par une meilleure bande passante réseau.
- Le choix de sélectionner dans notre architecture des composants qui ne nécessitent pas un gros effort de migration par rapport à ce que nos clients des anciennes distributions MapR, Hortonworks ou Cloudera peuvent avoir.

En relâchant, quand c'était nécessaire, certaines contraintes on aboutit à une architecture de référence que l'on va vous présenter dans la section 5. C'est celle d'Hurence mais des variations sur le thème sont possibles en fonction de vos cultures et existants. Mais avant regardons concrètement ce que nous avions de critique à gérer.

4. Le roi HDFS est mort, vive le roi !

Dans une infrastructure Big Data, le premier choix très structurant à faire est celui du système de fichiers distribué. Il va conditionner tous les autres composants logiciels qui devront être capables de se caler dessus.

Dans les infrastructures Big Data jusqu'à récemment, Apache HDFS était le système de fichiers distribué. C'est un stockage en blocs performant (par opposition à un stockage objet ou fichiers POSIX). Chaque fichier étant tronçonné en "relativement" gros blocs de généralement 512 ou 128 mégaoctets et ces blocs sont répartis et redondés (3 fois ou plus) sur les nœuds du cluster de stockage (disques différents en JBOD, racks différents, tout pour parer à nombre de pannes sauf à l'incendie de tout le data center !). Ensuite les traitements étaient automatiquement co-localisées sur les bons nœuds grâce à Yarn, le ressource manager et scheduler de Hadoop. Chacun des nœuds traitait localement la

donnée sans la transférer (ou en minimisant les échanges inter serveurs). Ce modèle est toujours viable pour des très très grosses volumétries mais il a aussi ses défauts. En effet, il faut faire “scaler” les nœuds de traitement de concert avec les nœuds de stockage (puisque ce sont les mêmes). Même si des ajustements ont été faits pour spécialiser les nœuds et en avoir plusieurs typologies, ce modèle était performant mais peu flexible et un peu coûteux.

Ceci dit, HDFS était un bon outil assez fiable: nous n’avons jamais été confrontés, en 11 ans, à des problèmes de corruption de données ou autres avec un usage dans les règles de l’art et en se calant sur les versions stables et gérant proprement les upgrade de versions.

4.1. Le roi ne bouge plus; le cloud l’a tué

Avec l’avènement de systèmes de fichiers à base du protocole S3, ce qu’utilisent tous les acteurs du cloud, HDFS et son système de stockage en blocs s’est déprécié au profit de stockages objets. N’ayant pas fait d’adaptation d’aucune sorte pour offrir d’autres paradigmes de stockage (objet, POSIX etc.), il est rattrapé par d’autres offres.

Indéniablement quand on va sur le site github de HDFS (voir figure section 5.2. Ceph: le système de fichiers distribué), on constate une moindre activité que par le passé. Les développeurs ont migré vers de meilleurs lendemains comme souvent dans les communautés libres. Avec le rachat de la société Hortonworks qui était derrière beaucoup d’activités de développement open source Hadoop, on imagine que beaucoup de collaborateurs ont quitté, accélérant une chute qui était peut être déjà écrite du fait d’une offre cloud en constante amélioration.

4.2. Le roi est mort… nous cherchons son successeur

Pour élire le nouveau roi, il faut revenir sur les besoins nouveaux que l’on a. Depuis l’avènement du cloud, avec en précurseur Amazon, des protocoles de stockage (S3) ont vu le jour. Ce sont des stockages objet avec des possibilités de définir des métadonnées riches sur les objets (fichiers), ce que ne permettait pas HDFS. Hors on cherche aujourd’hui à faire des développements relativement standards notamment dans une vue d’hybridation des traitements. En fonction des besoins en CPU ou GPU on voudra sans doute “déborder” chez un acteur du cloud pour calculer des modèles d’IA une fois par mois. Donc notre système de stockage (la manière de lire et écrire nos bytes et d’adresser nos fichiers par URL) doit être compatible avec ces environnements. D’autre part, on sait maintenant que notre roi n’est pas immortel et se caler sur un standard ou des standards est la meilleure manière de pouvoir le remplacer si l’on trouve un meilleur roi, et ce sans avoir à modifier les développements.

Ce comparatif pris et traduit du site https://cloudian.com/blog/object-storage-vs-block-storage/ donne un aperçu des comparatifs entre les types de systèmes de fichiers (nous avons corrigé quelques biais).

	Object Storage	Block Storage
Performance	Meilleure performance sur des gros contenus et sur des gros débits de données en streaming ndlr: les industriels avec les données de capteurs auront des bénéfices à utiliser ce stockage	Forte performance avec les bases de données et tout ce qui sera transactionnel
Géographie	Les données peuvent être stockées sur plusieurs régions ndlr: il faudra veiller à avoir une bonne bande passante réseau si c’est le cas	Plus la distance est grande entre la donnée et l’application et plus il y a de latence

Scalability	Peut scaler sur des pétaoctets et au delà	Il y a des limites à la scalabilité ndlr: l'infrastructure doit tenir sur un data center
Analytics	métadonnées customisables ; la donnée peut être facilement organisée et recherchée	pas de métadonnées ndlr: hormis les informations sur le fichier, son propriétaire et ses ACLs

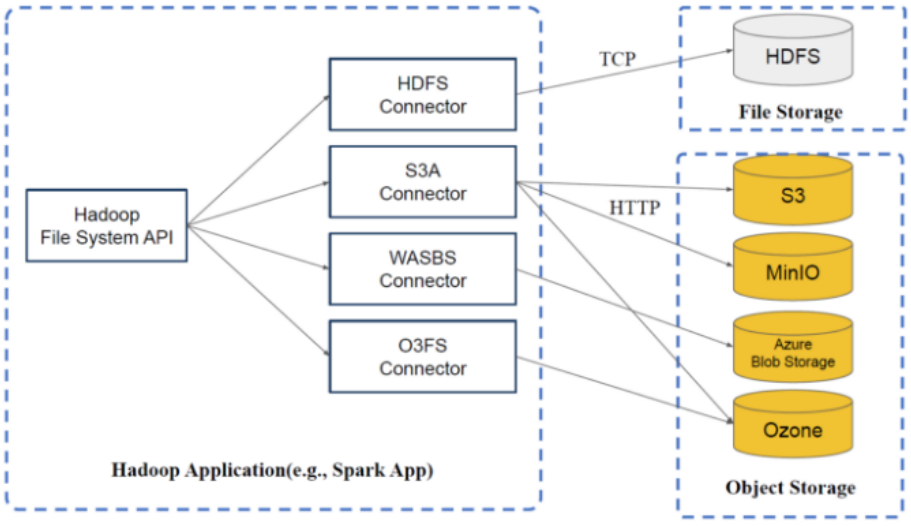
4.3. Alternatives à HDFS en open source

Si l'on regarde la compatibilité S3 en open source, l'offre est essentiellement **Minio**, **Ozone**, **Ceph**. Pour ce qui est non open source nous avons beaucoup d'offres donc celle des acteurs majeurs du cloud (AWS S3, Azure Gen2, Cloud storage).

Mais si l'on tente d'aller plus loin et d'avoir un système de fichiers multi paradigmes l'offre se restreint vite.

4.4. Une voie possible: juste remplacer HDFS

Nous aurions très bien pu faire une architecture de référence basée sur tous les composants Apache d'une distribution Hadoop classique dernier cri, Hadoop core et juste débrancher HDFS pour installer un nouveau système de fichiers (Ceph qui est S3, Minio ou Ozone). Nous aurions utilisé les connecteurs S3A vers l'une des solutions de remplacement comme dans la figure suivante. Ceux qui aiment leur Hadoop peuvent toujours choisir d'en installer un à base des composants Apache et faire cette connexion (ils pourront éventuellement être aidés de l'approche Apache Big Top et de Apache Ambari):



D'ailleurs Cloudera a choisi ce modèle et a remplacé HDFS par Ozone.

Cela nous a semblé trop problématique car nos clients s'étaient habitués à un certain confort d'utilisation (des interfaces utilisateur) et il aurait fallu réinvestir lourdement sur Ambari pour assurer les installations, les updates etc. Bref, gérer une stack en lieu et place de Hortonworks tout en sachant que cette infrastructure "packagée" a vécu et que nous en voulions une autre, plus moderne.

4.5. Une autre voie d'architecture plus moderne

Hurence n'a pas choisi la voie du remplacement du seul HDFS sur une installation Hadoop que l'on qualifie d'Apache bare metal (pour contourner les besoins en licences commerciales). La raison est liée au fait qu'on sait aujourd'hui que l'on doit intégrer de nombreuses technologies pas toujours du monde Apache/Hadoop, et que l'on aura besoin d'un seul scheduler et resource manager universel (Kubernetes). Aussi nous voulions une technologie intimement intégrée à Kubernetes et offrant maintes possibilités (un stockage POSIX, objet S3 et bloc). Nous ne voulions pas abandonner ce qui pourrait être intéressant pour conserver certaines bases de données comme HBase, comme le stockage en blocs.

4.6. Vive le roi Ceph

Nous avons donc choisi **Ceph** comme nouveau Roi pour sa capacité à nous proposer un stockage soit en POSIX (pour nos bases d'administration comme Postgresql mais aussi pour tout ce qui est stockage local, un stockage en S3 standard, un stockage en bloc pour conserver la possibilité de faire dessous clusters co-localisés en données et traitements.

5. Notre nouvelle architecture de référence

Dans les premiers temps du Big Data, Hortonworks avait donné une architecture de référence avec une sélection d'outils Apache qui étaient quelques fois peu fortement intégrés. Mais l'architecture était complète, innovante, intéressante à chaque nouvelle release. Progressivement, les outils se sont mieux intégrés et finalement d'autres acteurs ont fini par se rallier à certains choix de Hortonworks. C'est ce que nous avons tenté de faire ici en définissant la HDP new generation : Hurence Data Platform New Gen comme un clin d'œil à l'équipe Hortonworks que l'on se sent Hurence de remercier même si notre budget n'a jamais à l'époque permis de les aider.

Le premier choix structurant est donc celui du système de fichiers remplaçant de HDFS. Ceph n'est pas en licence Apache 2.0, mais LGPL mais ce n'est pas un problème dans notre cas car nous sommes sur des standards et remplacer cette brique ne serait pas un problème. Il est porté par la société Red Hat rachetée par IBM mais sa communauté est active malgré tout, ce qui est le gage qu'il ne disparaîtra pas. Ozone étant lui plus porté par Cloudera dont nous avons déjà apprécié la stratégie commerciale pour la HDP ancienne génération et la version gratuite de la Cloudera (plus personne ne peut faire scaler son infrastructure ni configurer un nœud).

Deuxième choix structurant, en fait Yarn n'est pas un scheduler et gestionnaire de ressources suffisamment universel. Dans le passé, lorsque l'on voulait faire co-localiser des technologies sur les nœuds d'un cluster Hadoop on remplaçait Yarn par Mésos: Mésos savait gérer les ressources d'un Cassandra au même titre que les ressources d'un Spark. Personne ne se marchait sur les pieds. Mais Mesos, populaire un temps, a été supplanté par un autre type de scheduler et gestionnaire de ressources : Kubernetes.

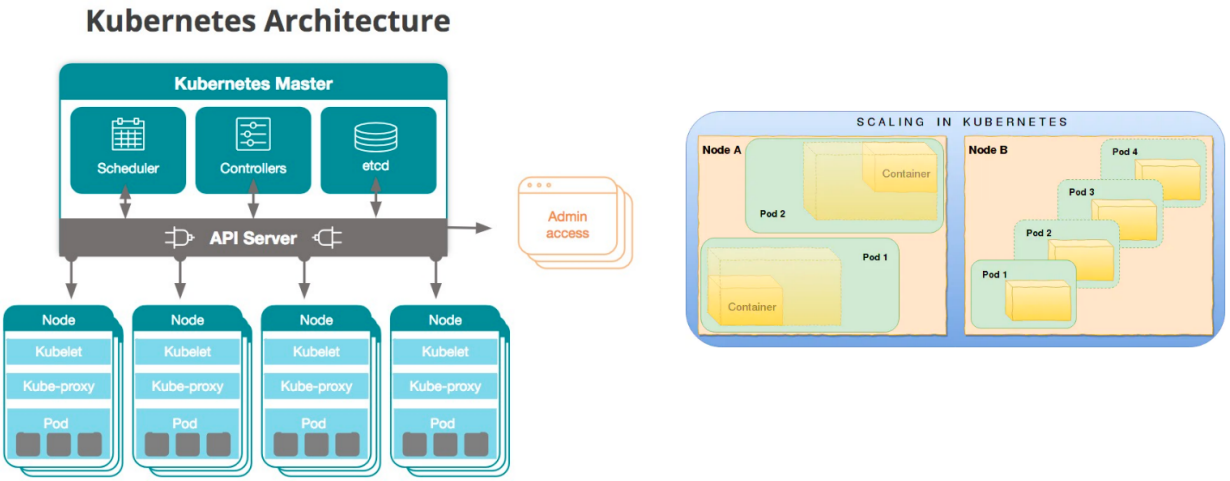
5.1. Kubernetes: notre socle

Kubernetes va constituer notre socle de gestion des ressources du cluster comme Hadoop Yarn l'était pas le passé. Avec Kubernetes on a la liberté de gérer les ressources de calcul ("compute") indépendamment des ressources de stockage ("storage") et le tout est bien orchestré.

Kubernetes est ce qui suit Docker dans la logique. Docker a permis de créer des "images" applicatives et de faire tourner ces images sans réellement se préoccuper d'installation (une image est créée avec tout le logiciel nécessaire et c'est elle qui est déployée toute installée sur tous les serveurs). Des points de montage sont créés pour les disques (données) et ensuite les images sont éventuellement connectées entre elles pour former "clusters". Kubernetes fonctionne donc sur ce principe rendu universel et pour former des clusters de machines.

Dans Kubernetes nous avons ce que l'on appelle des **Pods**. **Les Pods sont la plus petite unité de déploiement dans Kubernetes. Un Pod est un groupe d'un ou plusieurs conteneurs (au sens docker que la plupart d'entre vous connaissent déjà) qui partagent des ressources de stockage et de réseau ainsi qu'une spécification sur comment ces conteneurs doivent être lancés.** Des Pods de même nature peuvent être installés sur plusieurs nœuds physiques comme l'explique la figure suivante.

Ils pourront donc former des clusters si telle est la spécification de leur configuration.



Images extraites d'un blog de newrelic

Kubernetes va donc nous permettre de faire **scaler de manière indépendante la partie "stockage" et la partie "compute"**, permettant une réduction des coûts par rapport aux infrastructures Hadoop (ou les nœuds devaient scaler de "concert" pour éviter les déséquilibres dans les traitements).

De plus, les schedulers de Kubernetes pourront, si cela est demandé dans les spécifications, instancier automatiquement de nouveaux Pod pour typiquement supporter un pic de charge. L'éco système Kubernetes vient avec son load balancing (ingress controllers). Bien sûr, dans un cluster on premise, les ressources physiques sont plus réduites que dans le cloud et pas nécessairement provisionnables à la demande, malgré tout on peut faire varier les configurations des services pour donner plus à l'un et moins à l'autre en fonction des périodes d'utilisation, et cette flexibilité est offerte par Kubernetes même si il faut se préserver d'une complexification à outrance des infrastructures et les garder si possible aussi simples que possibles.

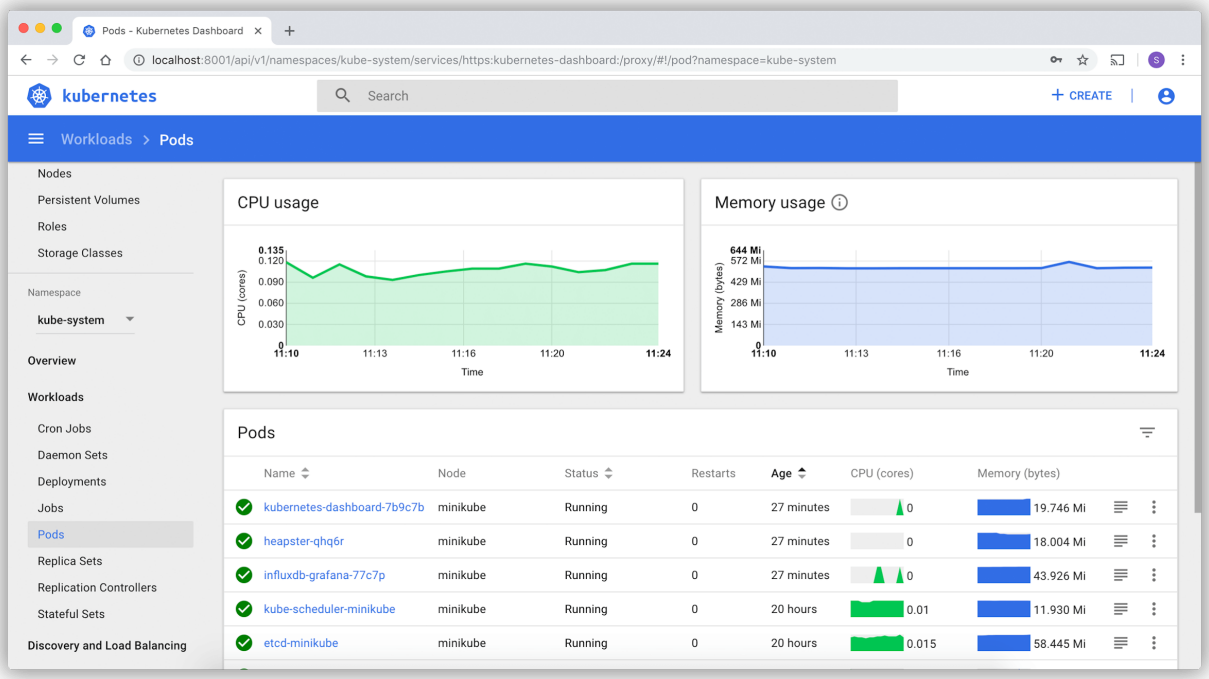
Kubernetes va aussi nous permettre de travailler avec ces images de logiciels en "isolation" avec la possibilité de faire cohabiter plusieurs images sur le même serveur physique (une image pour PostgresQL la base de donnée, et une image pour l'application Web qui l'utilise par exemple, ou une image pour un noeud Kafka et une image pour son stockage Ceph).

Kubernetes n'est que la gestion de toutes ces images qu'il permet d'instancier selon des demandes faites par l'intermédiaire de configurations. Et **ces configurations sont du code** et chaque changement de code fait que Kubernetes, au commit des changements de configurations va changer l'infrastructure, selon ce qui est maintenant spécifié. Les controllers de Kubernetes se chargeant de ces modifications. Par exemple, si l'on met à jour une image dans le repository gitops de Kubernetes, typiquement on vient de "comiter" la dernière version de l'application Web sous forme d'image, et bien Kubernetes selon les instructions (et seulement si on l'a configuré ainsi) va la déployer automatiquement. Ce sera potentiellement la même chose pour les jobs Hive, Impala, Spark ou Python modifiés. On entre dans un **univers de devops et d'intégration continue** qui demande un investissement initial (il faut décrire ces configurations) mais qui dans les phases de run de l'infrastructure en font quelque chose de plus agile et de plus à jour que les anciennes distributions Hadoop que l'on conservait 3 ans et faisait migrer tous les 3 ans.

Autres fonctionnalités intéressantes, il est possible plus facilement de faire déborder des travaux dans le cloud ou ailleurs et avoir des traitements sur plusieurs infrastructures. Dès lors, on peut parfaitement imaginer d'utiliser des outils d'un acteur du cloud pour des calculs de modèles d'IA ou autres sur des machines équipées de GPUs pour les utiliser ensuite sur son cluster "on-premises" avec toute la sécurisation nécessaire pour permettre cette ouverture sur l'extérieur puisque les interfaces réseaux et les firewalls, peuvent être ajustés pour se faire.

Kubernetes vient avec son monitoring (voir la figure empruntée à leur site) et sa propre sécurisation de qualité avec

notamment la **possibilité de mettre des polices d'accès sur les volumes de stockage de Ceph**. Une belle infrastructure qui de plus devient standard et se retrouve dans tous les clouds ce qui en fait un socle universel avec un investissement pérennisé.



En résumé un magnifique socle dont les possibilités d'automatisation peuvent vite rendre complexe une infrastructure. Ce sera le principal danger de cette superbe mécanique. Les experts Kubernetes devront réfréner leurs ardeurs technologiques pour que l'infrastructure reste facilement opérable par des gens formés mais non experts.

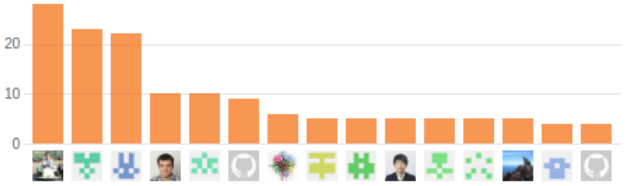
5.2. Ceph: le système de fichiers distribué

Nous avions plusieurs possibilités pour remplacer Hadoop HDFS. Nous voulions un outil open source (licence libre), se conformant à des standards (S3, POSIX), un outil bien intégré à Kubernetes, et un outil avec une communauté solide. Parmi les candidats nous avons trouvé Minio, Apache Ozone et Ceph. Alors nous aurions pu parier que HDFS s'adapterait mais, le problème de HDFS est en réalité sa communauté.

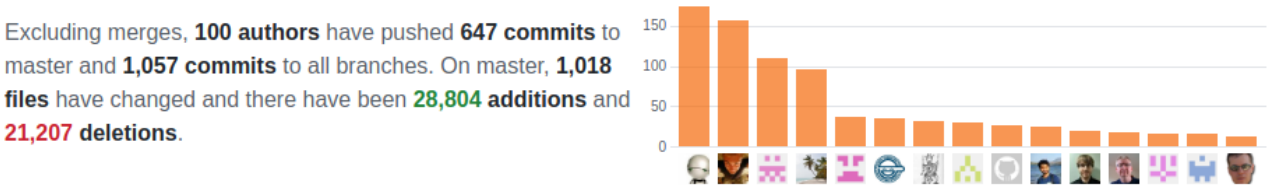
HDFS appartient au monde Hadoop, n'évolue plus ou quasiment plus. De notre point de vue c'est une technologie qui risque de mourir rapidement. Cloudera a d'ailleurs réalisé l'intégration de Apache Ozone pour la même raison.

Voici l'activité de développement sur Hadoop (HDFS inclus):

Excluding merges, **43 authors** have pushed **77 commits** to trunk and **197 commits** to all branches. On trunk, **297 files** have changed and there have been **7,209 additions** and **1,352 deletions**.



Voici l'activité de développement sur Ceph (qui est un autre ordre de grandeur):

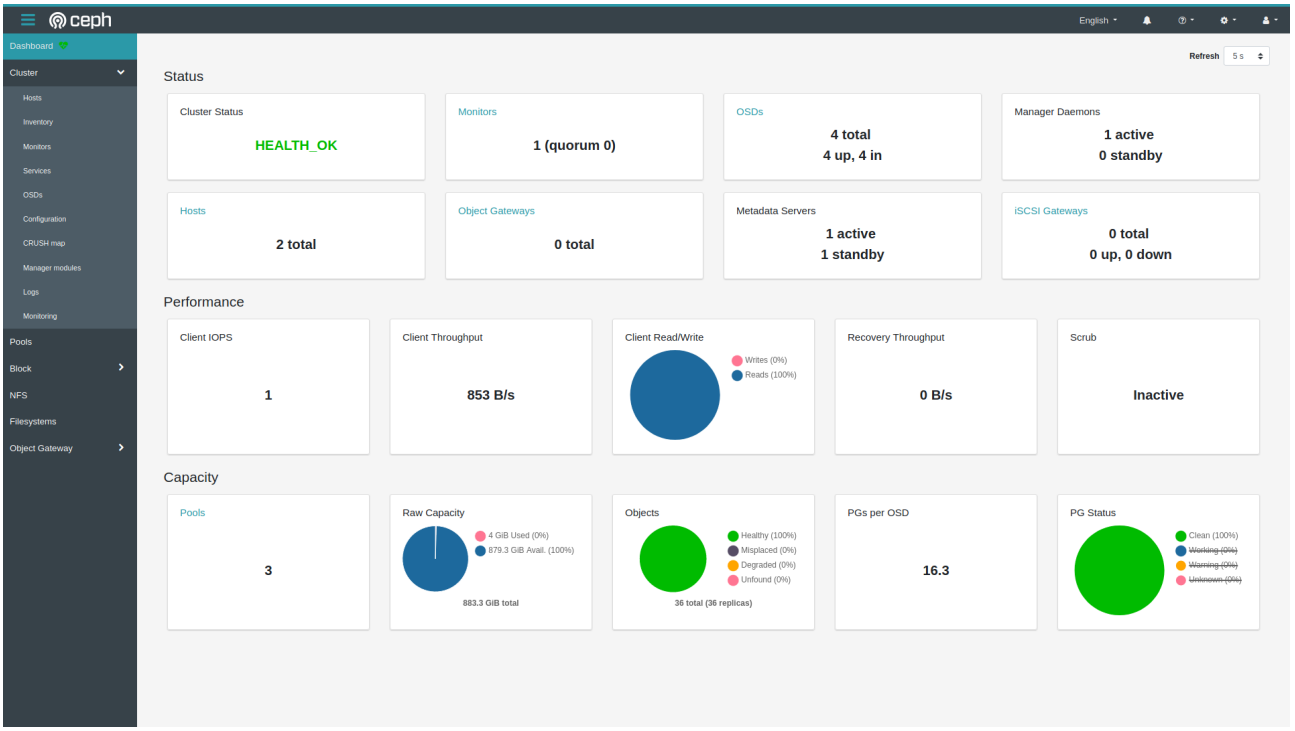


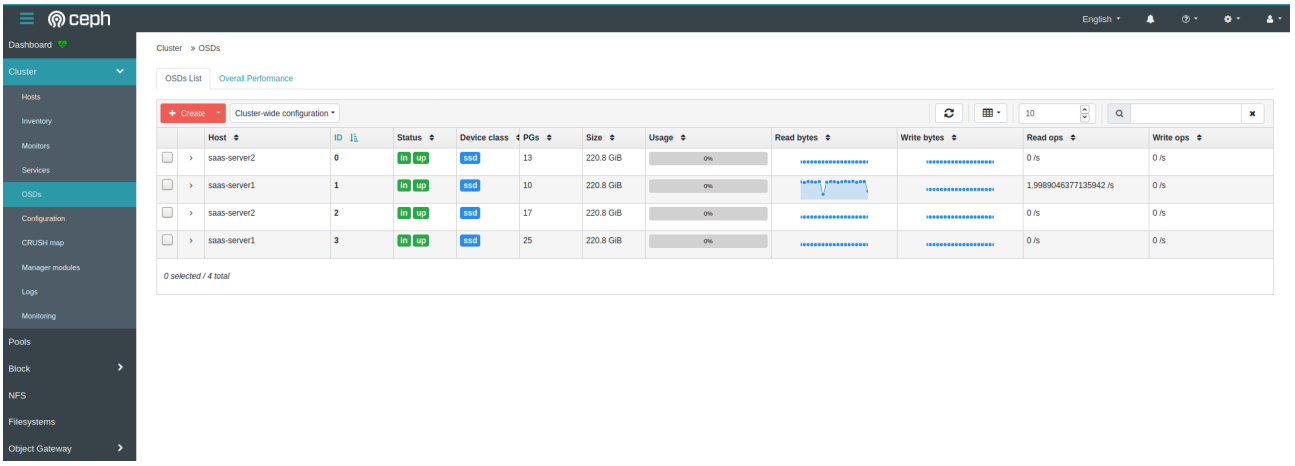
Ceph est un système de fichier distribué dont l'architecture est plus performante que celle de HDFS. De plus, il permet de créer des espaces de stockage multiples soit en mode POSIX, soit du stockage objet comme S3 d'Amazon, soit du stockage en bloc (comme HDFS). Les trois peuvent cohabiter pour divers usages. Dès lors, notre architecture de référence mettra d'activer du stockage POSIX pour réaliser le stockage des bases de données type Postgresql, du S3 pour les fichiers traditionnels des data lakes (fichiers parquets, ORC et autres). Ceph gère tout sous forme d'objets et les métadonnées sur les objets (fussent ils des fichiers) sont riches et peuvent être largement utilisées pour divers cas d'usages (y compris des cas d'usages RGPD). C'est le composant Rados qui est chargé du stockage de ces objets, quel que soit leur type. La couche RADOS assure la consistance des données, réalise la réplication, la détection de problèmes et la récupération des données (recovery). Il assure aussi la migration et le rééquilibrage des données sur l'ensemble des nœuds du cluster. C'est fiable et performant, ce que l'on demande à un bon système de fichiers distribués.

Ceph gère très bien les quotas de stockage aussi. Enfin il résout une grosse difficulté de HDFS: la gestion de quantités de petits fichiers. Pour ceux qui veulent plus d'information sur l'architecture interne de Ceph on peut conseiller le site de Ceph.

Ceph vient avec des capacités de compression de données via des plugins. Il vient aussi avec des capacités d'encryption de données.

Enfin, Ceph vient avec sa GUI pour notamment visualiser la santé du cluster. Elle n'a rien à envier à celle des GUIs dont nous avions l'habitude dans les distributions Hadoop.





Seul petit bémol sur Ceph, il est en licence LGPL et non Apache 2.0 mais ceci n'a pas d'incidence ni pour nos clients ni pour nous car toute modification que nous ferions à Ceph serait portée de toute façon à la communauté. Redhat, la société derrière la communauté a été rachetée récemment par IBM. Mais le commitment de IBM sur l'open source est plus fort que pour d'autres sociétés (avec la fondation Eclipse, IBM a une tradition open source). De plus, la communauté derrière Ceph est plus large que les seuls collaborateurs de Redhat. Le risque est donc minime et le fait que l'on s'appuie sur du standard POSIX ou S3 de toute façon assure que nos investissements seront pérennes dans tous les cas.

Une référence et non des moindre Ceph est le stockage derrière le système de collision Hadron au CERN (accélérateur de particules).

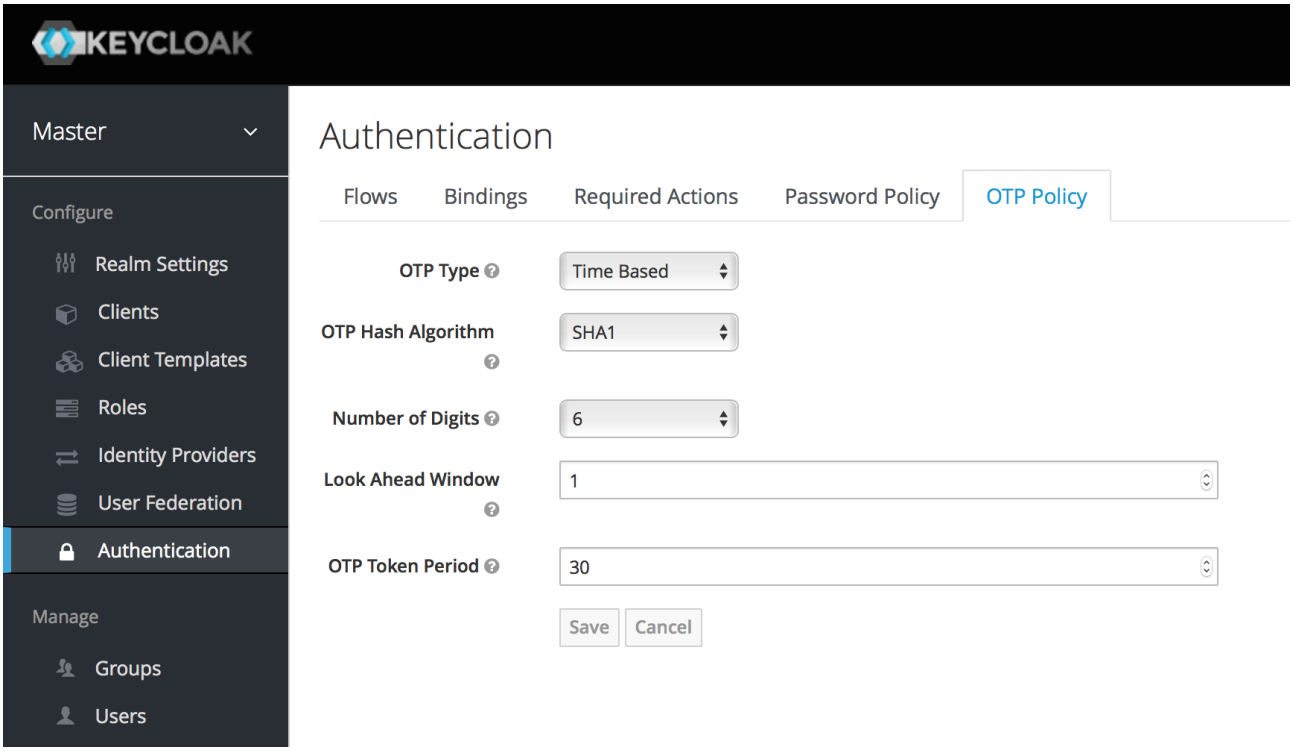
5.3. Keycloak: Gestion des utilisateurs, des identités, des droits

Kerberos, la technologie utilisée dans Hadoop pour conjointement avec une base LDAP (un Active Directory) ou autre pour sécuriser les clusters Hadoop a été disons le un réel frein à l'adoption et à l'opération de Hadoop. Forts de ce constat, c'est une erreur que nous ne voulons pas répliquer dans notre nouvelle architecture.

Quelques clients très en avance, qui se reconnaîtront ont expérimenté assez tôt l'utilisation de Keycloak pour sécuriser leur infrastructure Big Data revisitée (un peu différente de la nôtre sur le système de fichiers utilisé). Nous leur avons emboîté le pas dans notre architecture de référence sachant que nous utilisions Keycloak par ailleurs sur nos développements open source comme le data historian.

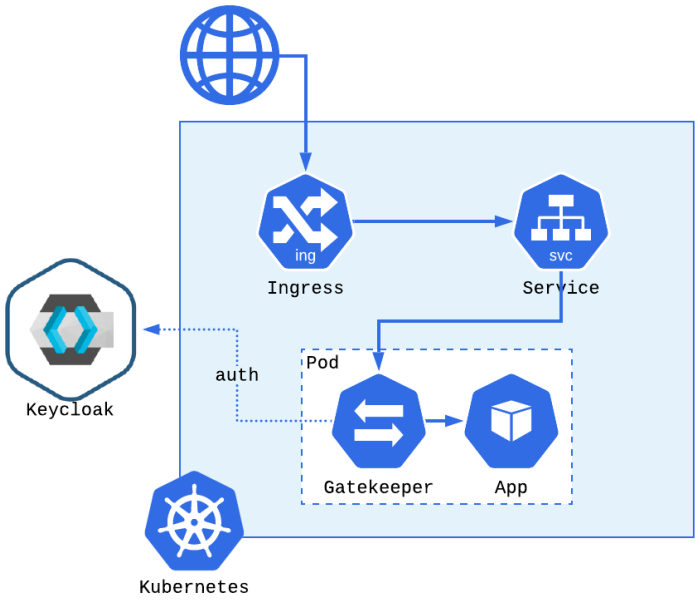
Il est possible de gérer les droits et rôles directement dans Ceph mais le mieux est de brancher Keycloak, si possible, et avoir une authentification et un **SSO (Single Sign On) sur tous nos outils grâce à Keycloak**. On pourra aussi utiliser Keycloak pour que les services s'authentifient aussi avant d'échanger. Keycloack nous servant de base LDAP (pour définir les rôles, les utilisateurs) et mécanisme "à la" Kerberos pour bien vérifier qui est qui dans les échanges.

Keycloack vient avec son interface utilisateur pour gérer tous les utilisateurs de tous les outils de la plateforme. Finalement c'est un composant qui nous permet de mettre de la cohérence sur tous nos outils.



Nous brancherons l'outil Keycloak comme système d'authentification et d'autorisation de manière à ce que tout soit cohérent de ce point de vue.

L'architecture donne donc de ce point de vue (image empruntée à Openshift de Red Hat), à savoir que nos applications seront protégées par un "Gatekeeper" (cela peut être un reverse proxy) qui s'occupera d'authentifier et d'autoriser les appels:



Avec ces trois outils, Kubernetes, Ceph et nos outils d'administration des droits, nous avons notre socle. Nous sommes maintenant libres de déployer les bons outils Big Data pour revisiter notre infrastructure.

5.4. Hive or not Hive, Impala or not Impala: une décision Presto s'impose...

Un des composants phares et très utilisé dans une distribution Hadoop est Hive et son alter ego Impala. Alors rien n'empêche de déployer Hive, son metastore et son thrift server. Mais bien sûr on ne veut pas de MapReduce ni de Tez (les moteurs d'exécution de Hive dans Hadoop). Heureusement, Hive permet l'utilisation de Spark comme moteur d'exécution de ses requêtes SQL donc une simple configuration de Hive avec:

```
set hive.execution.engine=spark;
```

fera le job. Et bien sûr Hive ira stocker en S3 ses données dans Ceph.

Donc on peut monter un Pod Hive bien configuré et l'on aura un Hive sans Hadoop.

Même chose pour Impala, on peut conserver un Impala en le configurant également pour utiliser Ceph comme stockage sachant que Impala vient avec son propre moteur d'exécution. Donc une installation à base de la version Apache de Impala est parfaitement possible (pour ceux qui ont utilisé les spécificités de Impala et ont un capital de code important dessus).

Après, quitte à migrer, on pourrait adopter Presto (renommé Trino récemment) qui permet d'avoir une certaine universalité: en effet depuis Trino (Presto) on peut requêter toute sorte de base de données en SQL - un SQL standard pour peu que l'on dispose d'un connecteur. Il existe des connecteurs pour Elasticsearch, mongoDB, Cassandra, Druid, Redis, Kafka etc. Faire des requêtes avec des jointures sur plusieurs tables externes quel que soit l'outil nous semble intéressant car dans bien des cas des données sont stockées dans divers back ends et l'on souhaite les croiser sans nécessairement avoir à les redumper toutes sous forme de fichiers dans le système de fichiers.

5.5. Quid de notre data catalog Apache Atlas

La Hortonworks Data Platform a fourni assez tôt un data catalogue permettant de recenser la plupart des données du data lake constitué (tables Hive, etc.) avec l'outil Apache Atlas. Beaucoup de nos clients se sont approprié ce data catalogue qui a toujours sa communauté et est le seul de sa catégorie. Nous avons donc choisi de l'inclure dans notre architecture de référence sachant qu'un plugin permet de le superposer à Ceph et que le Hive metastore marche aussi dans ce cas.

5.6. Quid de Apache Ranger

Nous avons également choisi de conserver Ranger pour la partie d'écriture des polices d'accès aux données et des possibilités d'audit de données. Il y a de la gestion de polices d'accès dans les trois outils Kubernetes, Ceph et Ranger. Kubernetes gère plus les polices d'accès aux services et aux disques alors que Ranger est lui au niveau des data sets. Ceph quant à lui reste au niveau des fichiers (ACLs). La positionnement Ranger / Ceph reste le même que celui de Ranger / HDFS en quelque sorte. Un plugin est utilisé pour connecter Ranger et Ceph. Les polices d'accès Kubernetes sont une surcouche qui permet une sécurité de plus bas niveau encore. On a donc les 3 niveaux, accès disques, accès fichiers et accès datasets.

Maintenant que l'on a fait le tour des gros points noirs de notre architecture à savoir les éléments sur lesquels nos clients avaient massivement investi dans les infrastructures Hadoop, voyons la table logique des remplacements (ou conservations) suivie d'une table d'ajouts, car on souhaitera faire des ajouts.

5.7. Table des remplacements

La table suivante nous présente les remplacements que nous proposons pour passer d'une Hortonworks Data Platform ou d'une Cloudera. En fait tout est possible mais on peut en profiter pour ne garder que le meilleur qui a fait ses preuves au fil des ans.

Alors bien sûr certaines sociétés utilisent des bases de données en sus des éléments de la distribution Hadoop qu'ils utilisaient. Ils peuvent choisir de laisser ces installations telles quelles ou choisir de les réintégrer dans Kubernetes aussi, formant ainsi une seule et même infrastructure Big Data.

Outil de la distribution Cloudera et Hortonworks	Fonction	Équivalent dans notre stack Kubernetes
Hadoop HDFS	Stockage distribué	Ceph https://ceph.io/ Ceph nous amène les 3 modes de stockage S3, bloc et POSIX alors que HDFS n'était que bloc.
Hadoop Yarn	Scheduler et gestionnaire des ressources	Kubernetes https://kubernetes.io/ Kubernetes amène tout un monde de scheduling, de gestion sophistiquée des ressources, d'automatisation des mises à jour et des déploiements avec le CI/CD etc. On gagne donc en sophistication et l'administration des clusters devient un travail de personnels légèrement plus expérimentés qu'avec Hadoop.
Ambari ou Cloudera Manager	Administration des noeuds, du stockage	Kubernetes GUIs https://kubernetes.io/ On perd le packaging sous la forme d'un repository unique où l'on va se sourcer mais tous les outils ont des Helm Charts à savoir des installateurs individuels. Helm est un package manager pour Kubernetes (l'équivalent d'un "apt-get install" ou "yum install" dans les OS). On perd un peu la possibilité de faire des mises à jour en quelques clics par GUI mais là encore les Helm charts prennent en compte les upgrade. Par contre, le CI/CD fait gagner en mise en production des applications. On gagne aussi en universalité pour la gestion de toutes les ressources de tous les outils/ bases de données utilisées.
Nifi	ETL	Nifi https://nifi.apache.org/ On conserve notre Nifi qui est compatible avec Ceph (comme avec Azure Gen2, AWS etc.)
Oozie (workflow)	Workflow de traitements	Nifi, Airflow, Kubeflow (selon le job) On gagne en qualité d'outillage car Oozie était assez primitif.
Hive	Moteur SQL	Hive https://hive.apache.org et TrIno (ex Presto) https://trino.io/ Hive gagne son indépendance de Hadoop (exécuté sur Spark) et avec Presto on obtient des possibilités de jointures SQL entre divers outils comme Elasticsearch, MongoDB
Impala	Moteur SQL	Impala https://impala.apache.org/ et Trino https://trino.io/ Impala s'affranchit aussi donc de Hadoop. Il survit

		du fait de ses spécificités SQL pour les clients qui se sont fait piéger à ne pas rester sur du SQL standard.
Spark	Moteur d'exécution distribué batch et temps réel fenêtré	Spark https://spark.apache.org/
Flink	Moteur d'exécution temps réel	Flink https://flink.apache.org/
Kafka	Broker de messages distribué	Kafka https://kafka.apache.org/ Ayant ses propres Pods de stockage natif, Kafka n'a pas de dépendance à Hadoop.
Zookeeper	Système de coordination inter outils	Zookeeper https://zookeeper.apache.org/ Ayant aussi ses propres Pods de stockage natif, Zookeeper n'a pas de dépendance sur Hadoop.
Hue	Outils d'accès aux composants	https://github.com/cloudera/hue On peut installer Hue dans sa version github et le superposer à nos outils. Il est en open source et en licence Apache 2.0. Comme pour la HDP ancienne génération (nous le faisions aussi), c'est un peu un artifice mais cela permettra aux clients de se retrouver en univers connu.
Zeppelin	Notebook pour développeurs Scala, SQL, R	Zeppelin https://zeppelin.apache.org/
Jupyter	Notebook pour développeurs python, R	Jupyter https://jupyter.org/
Solr	Moteur de recherche	Solr https://solr.apache.org/
Prometheus	Outil de monitoring	Prometheus https://prometheus.io/
Grafana	Outil de dashboarding spécialisé séries temporelles	Grafana https://grafana.com/
Apache Superset	Outil de dashboarding généraliste	Superset https://superset.apache.org/
Kerberos	Système d'autorisations pour utilisateurs et services	Keycloak https://www.keycloak.org/

Note: nous n'avons pas mis explicitement l'ancienne base de données HBase. Elle, comme Cassandra, MongoDB, Redis, Elasticsearch font partie des bases superposables à égalité dans notre architecture. Ces bases sont "en théorie" installables sur Ceph (grâce au connecteur Cephfs). Et dans le cas de HBase nous serions "en théorie" contents de bénéficier grâce à Ceph, de la possibilité d'instancier une partie de notre système de fichiers distribué en bloc pour HBase et donc de ne pas perdre en performance. Mais sans doute faudrait - il solidifier le connecteur Hadoop - Ceph. En pratique, si l'un de nos clients utilisait HBase, nous installerions plutôt un HBase en cluster sur des disques en JBOD (dans Kubernetes bien sûr). Comme on le ferait pour un Cassandra ou un MongoDB en l'absence d'une intégration forte et sûre avec Ceph.

5.8. Table des ajouts

Comme évoqué précédemment nous avons tenu à ajouter dans notre nouvelle référence des outils liés à des cas d'usages typiques Big Data que nous avons rencontrés dans l'industrie ou dans le commerce.

Le premier de ces outils est une plateforme de blockchain car les clients sont de plus en plus enclins à demander ce type d'outils qui nécessitent également de stocker potentiellement de grosses volumétries de données. Donc nous avons fait le choix de Hyperledger besu (des projets Linux foundations) https://www.hyperledger.org/use/besu et l'avons intégré à la stack. Ainsi les clients peuvent gérer des changements de propriétaires lors du transport de leurs containers, des acquittements de réceptions de produits etc. Une belle technologie qui a définitivement sa place dans une architecture Big Data moderne.

Un second outil important, lié à l'IoT ou l'IIoT (Iot industrielle) c'est la possibilité de se connecter à des données de capteurs et donc de référencer ces capteurs dans une forme de registre et d'activation/désactivation de la captation des données sur les devices. Cette fonctionnalité est donnée par Kapua de la fondation Eclipse https://www.eclipse.org/kapua/ . Le backend de Kapua est actuellement peu scalable donc il pourrait y avoir un besoin d'en déployer plusieurs. Hurence regarde dans quelle mesure il pourrait remplacer ce back end et lier plus profondément son data historian à cet outil (qui lui est scalable).

Le troisième outil est donc le data historian innovant et open source de Hurence (en licence Apache 2.0) https://github.com/Hurence/historian. C'est un bel outil permettant de stocker d'énormes quantités de données de capteurs et de les rechercher grâce à une indexation symbolique: Symbolic Aggregate Approximation (SAX). On peut rechercher des patterns de séries temporelles comme on chercherait des mots sur Google. Les séries temporelles sont visualisables dans Grafana (avec un plugin historian). Le plugin permet notamment du sampling - échantillonnage intelligent (nécessaire sur de grosses volumétries) pour grapher sur des années de points. C'est une belle alternative gratuite à InfluxDB (quel que soit le nombre de nœuds c'est gratuit). Enfin, une belle intégration avec Spark permet de faire du machine learning très simplement sur les données stockées.

Le quatrième outil est un plébiscite: c'est Elasticsearch mais dans sa version opendistro https://opendistro.github.io/for-elasticsearch/ . Elasticsearch est le moteur de recherche le plus populaire et donc l'ajouter alors qu'il faisait toujours partie de tous nos data lakes (mais sur le côté) est une excellente idée. Le voici partie intégrante de notre architecture de référence.

Un cinquième outil qui nous a semblé intéressant de rajouter c'est le plugin Loki de grafana qui permet d'agréger les logs, notamment les logs des Pods Kubernetes https://github.com/grafana/loki. C'est un bon compagnon à Kubernetes, facile à opérer.

Autre outil faisant son entrée dans l'architecture de référence: une base de données graphe. Pourquoi ? et bien parce qu'elle nous sert sur des cas d'usages de gestion de connaissances qui entrent aussi dans notre architecture de référence. Nous avons hésité entre JanusGraph et ArangoDB. Nous avons choisi ArangoDB https://www.arangodb.com/ (choix historique) mais ce choix pourrait à terme être revisité. Nous utilisons une surcouche pour créer nos graphes de sorte de pouvoir changer ce composant dans le cas d'un changement de stratégie commerciale de ArangoDB. Nous avons une forte demande sur les graphes de connaissance d'entreprises; nouvelle forme de représentation de données qui doit passer aussi à l'échelle.

Pour la partie MLOps nous avons fait le choix de faire entrer Kubeflow https://www.kubeflow.org/ qui permet de définir les pipelines de machine learning et automatiser (puisque c'est sur Kubernetes) le re-training des modèles aux changements de données et les redéploiements automatiques. Ce papier de Google explique un peu notre philosophie:
https://cloud.google.com/solutions/machine-learning/architecture-for-mlops-using-tfx-kubeflow-pipelines-and-cloud-build bien que nos pipelines ne tourneront pas nécessairement dans le cloud (mais pourrons y déborder si nécessaire). Nous avons aussi intégré Airflow https://airflow.apache.org/ en contre-proposition de Nifi car demandé par nos clients dans certains cas.

Toujours pour la partie machine learning et plus généralement travaux python nous avons choisi de complémenter Spark avec l'infrastructure Dask https://dask.org/ . Dask fait du nativement parallèle sur du code python. Il aide les data scientists à ne pas se préoccuper de la partie parallélisation de leurs algorithmes qui sont de facto parallèles dans les parties ou ils peuvent l'être. Un bon outil qui ravit cette population.

Pour les clients qui voudront sûrement déployer nombre d'applications ou APIs sous forme de **microservices** (serverless) ce qui est bien sûr possible dans Kubernetes ; pour cela nous avons intégré OpenFaas (https://www.openfaas.com/) dans l'architecture de référence.

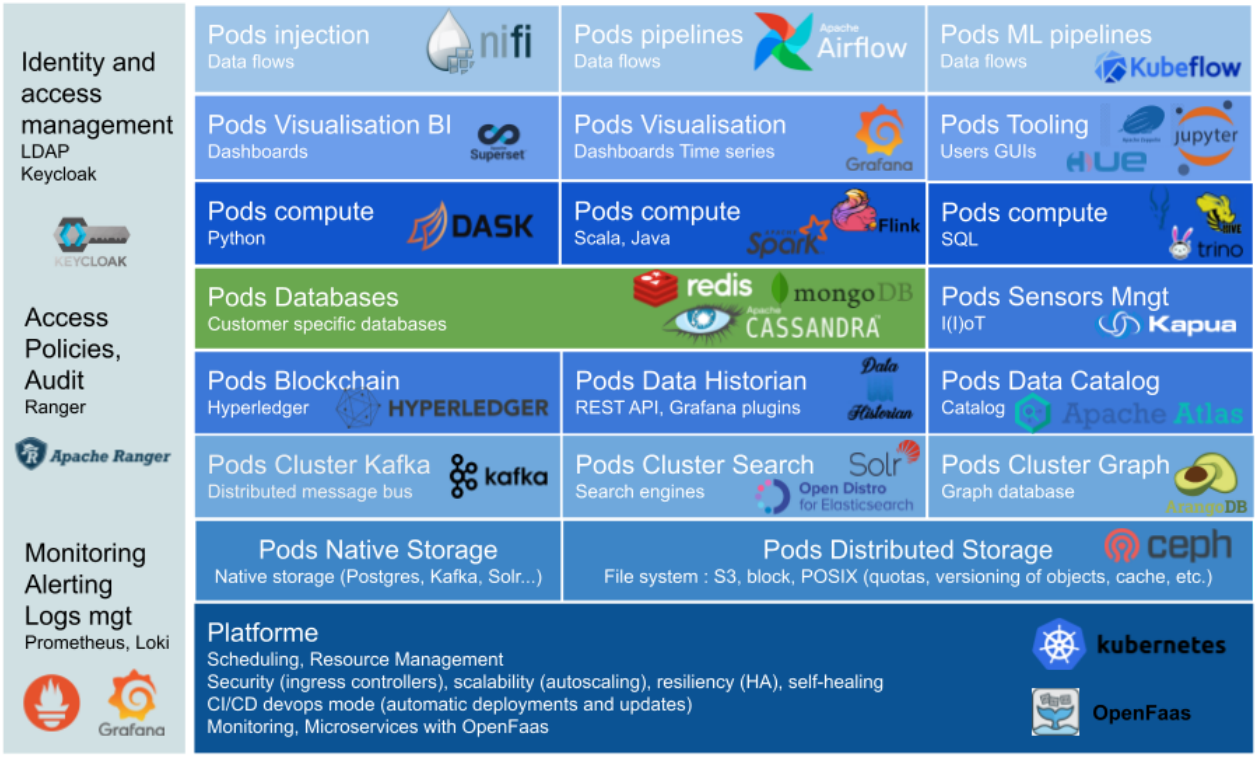
Enfin un dernier choix très structurant c'est Keycloak https://www.keycloak.org/ . C'est le système qui remplace Kerberos et permet les identifications (des utilisateurs comme des services) et la gestion des accès.

Notre table des ajouts est donc:

Outil	Fonction	URL
Hyperledger besu	Blockchain	https://www.hyperledger.org/use/besu
Kapua	Gestionnaire de devices de type IoT (capteurs d'équipements industriels ou non)	https://www.eclipse.org/kapua/
Hurence Data Historian	Time series data historian	https://github.com/Hurence/historian
Airflow	Pipelines divers (une alternative à Nifi dans certains cas)	https://airflow.apache.org/
Kubeflow	Pipelines Machine Learning et MLOps	https://www.kubeflow.org/
Open distro for Elasticsearch	Search engine	https://opendistro.github.io/for-elasticsearch/
ArangoDB	Support pour les graphes de connaissances	https://www.arangodb.com/
Dask	Framework de traitement parallélisé de code python	https://dask.org/ La licence est une licence libre BSD 3 ce qui en pratique nécessite de citer la société Anaconda, Inc.
Loki	Plugin Grafana d'agrégation de logs fonctionnant sur le modèle de Prometheus	https://github.com/grafana/loki
OpenFaas	Technologie de mise à disposition de microservices en mode serverless	https://www.openfaas.com/
Keycloak	Gestion des identités et des accès	https://www.keycloak.org/

6. Architecture de référence

Après tout ce suspense, mais une description fidèle de notre cheminement pour y arriver, nous obtenons donc l'architecture de référence suivante. Bien entendu, comme pour les distributions Hadoop, le client peut faire le choix de ne pas utiliser tel ou tel service mais nous avons fait une sélection éprouvée d'outils que l'on rencontre dans nos cas d'usages qu'ils soient industriels ou plus liés aux besoins commerce et marketing. Nous avons également tenu à ce que l'univers de chaque client habitué à sa distribution Hadoop ne soit pas dérouté. Finalement on n'a fait que remplacer HDFS et Yarn par Ceph et Kubernetes. Puis Kerberos par Keycloak. Le reste sont des outils déjà connus des habitués du Big Data. Seuls quelques ajouts, que le client pourra décider ou non de déployer.



7. Stratégie pour réaliser ce petit Big Bang

7.1. Quelques problématiques à gérer

La stratégie pour réaliser la migration vers cette nouvelle infrastructure sera l'objet d'un prochain blog sur le site de Hurence. Mais pour en donner une indication on a trois problématiques:

1) La première est que l'on veut dans l'idéal construire la nouvelle infrastructure avec les dernières versions stables de tous les outils. Mais cela veut dire des ajustements des codes existants et cela peut être coûteux et long à faire. Il va donc falloir définir une stratégie où les composants sont migrés avec des **priorités**. Possiblement il faudra réaliser des **synchronisations de données** pour ce qui manque entre le legacy existant et la nouvelle infrastructure. Enfin, dans certains cas, il faudra se résoudre à installer peut-être des versions Apache plus anciennes pour tourner les anciens codes, tout en développant les nouveaux dans les nouvelles versions. Kubernetes nous permettra cette souplesse.

2) La seconde est que les clients ne veulent pas arrêter leurs développements pendant la migration donc on a un souci potentiel de "moving target". On migre un job vers la nouvelle infrastructure mais des besoins impérieux du business ont déjà nécessité le changement de ce job et on doit recommencer: c'est la moving target.

3) La troisième est que les clients vont devoir être formés à cette nouvelle infrastructure qui est par nature moins simple à administrer pour une plus grande puissance et une harmonisation des outils sur un seul et même socle. Un challenge pour les équipes en général malgré tout ravie de ce nouveau challenge Kubernetes.

7.2. Stratégie pour un déploiement dans un cloud public

Autre aspect de la stratégie; certains voudront malgré tout déployer cette infrastructure dans le cloud Amazon, Azure ou Google platform. Hurence va plus naturellement sur des serveurs bare metal chez OVH avec une réplication à l'identique chez un autre hébergeur et ses grands clients ont tous des racks dédiés Big Data dans leurs data centers.

Dans ce cas c'est assez simple, il faudra a priori plutôt **remplacer Ceph par le stockage S3 natif de la plateforme** (AWS S3, Azure Gen2, Google Storage). La plupart des outils de l'architecture de référence y compris Kubernetes pourront être instanciés en service managé (Kafka, Spark, etc.).

Nous ne conseillons pas nécessairement de prendre Kubernetes en service managé chez l'acteur du cloud car c'est par cet outil que l'on a un petit risque de perdre la standardisation. Chaque acteur ayant sa propre version de Kubernetes et y opérant des changements, c'est par cet outil que l'on peut perdre potentiellement son indépendance. D'autres part les services managés induisent la plupart du temps des coûts de licences qu'il faudra accepter en sus avec le support à la clé. Dans la plupart des cas, on aura un important intérêt financier à installer le service dans sa version "bare-metal" apache, comme Hurence le fait souvent pour ses petits clients (et lui-même) et accepter de s'appuyer uniquement sur la communauté pour les fixes. C'est une réflexion d'entreprise à mener au cas par cas avec une bonne estimation détaillée de tous les coûts. N'oubliez pas qu'en facturation à l'usage, le Big Data coûte vite cher du fait des volumétries ou la nécessité d'infrastructures minimales pour les redondances ou le temps réel (ou juste l'administration).

8. Conclusion

Nous avons réussi à trouver une alternative open source et gratuite solide à nos anciennes distributions de Hadoop. Notre assemblage est complet et même bien plus large en couverture de fonctionnalités avec l'introduction d'une blockchain, de gestion des capteurs et d'une base "graphe" pour la gestion de connaissances. Bien évidemment nos clients ne vont pas nécessairement tout instancier mais ils ont un disposition un beau catalogue.

Ce faisant, nous avons augmenté, nous ne le cachons pas, le ticket d'entrée en termes de formation des équipes pour la gestion de l'infrastructure qui est aussi devenue plus "centrale", plus "puissante" mais plus "complexe" de par sa richesse. La partie CI/CD (continuous integration, continuous deployment) ajoute en professionnalisme et aussi en complexité. Elle sera également abordée dans un autre blog de Hurence.

A ce moment de notre aventure Kubernetes, nous avons choisi de tout mettre sur même déploiement Kubernetes mais peut être l'avenir nous fera revisiter ce choix pour des besoins en simplification d'administration ou d'autres raisons qui militent pour une séparation des outils dans des infrastructures Kubernetes séparées.

Nous espérons que ce white paper, fruit de notre travail passionné sur le Big Data et de la volonté d'aider nos clients à sortir de leurs blocages actuels, a pu vous éclairer un chemin possible.

N'hésitez pas à contacter l'équipe Hurence pour toute question complémentaire ou tout besoin concernant ces infrastructures à :

contact@hurence.com

(ou sales@hurence.com si vous ressentez le besoin d'une assistance ou d'un accompagnement sur ce sujet).