



---

Exécuter Spark  
sur Dataproc

# Programme

---

## L'écosystème Hadoop

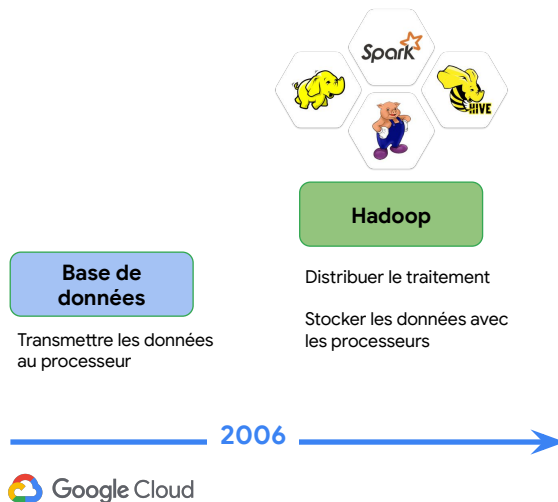
Exécuter Hadoop dans  
Cloud Dataproc

GCS au lieu de HDFS

Optimiser Dataproc

Atelier

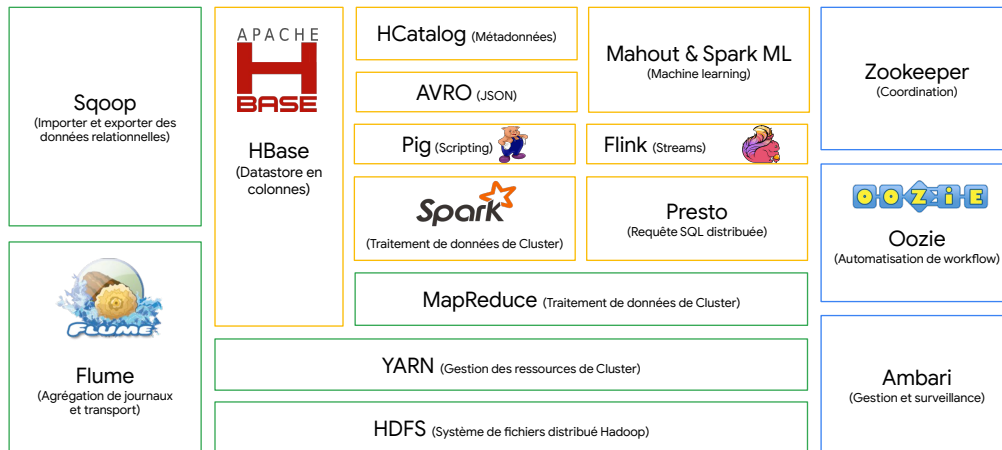
# L'écosystème Hadoop s'est développé en raison de la nécessité d'analyser de grands ensembles de données



Il permet de replacer les services que vous allez découvrir dans leur contexte historique. Avant 2006, les big data étaient synonymes de grandes bases de données. La conception des bases de données est née à une époque où le stockage était relativement bon marché et le traitement coûteux. Il était donc logique de copier les données de leur lieu de stockage vers le processeur pour effectuer le traitement des données. Le résultat était ensuite recopié sur le stockage.

Vers 2006, le traitement distribué de données volumineuses est devenu plus pratique grâce à Hadoop. L'idée qui sous-tend Hadoop est de créer un cluster d'ordinateurs et de tirer parti du traitement distribué. HDFS - le système de fichiers distribués de Hadoop stockait les données sur les machines du cluster, et Map Reduce assurait le traitement distribué des données. Tout un écosystème de logiciels liés à Hadoop s'est développé autour de Hadoop, dont Hive, Pig et Spark.

# L'écosystème Hadoop est très populaire pour les gros volumes de données



Les organisations utilisent Hadoop pour les charges de travail big data sur site. Ils utilisent une série d'applications qui s'exécutent sur des clusters Hadoop, comme Presto, mais beaucoup de clients utilisent Spark.

Apache Hadoop est un projet de logiciel open source qui développe un cadre pour le traitement distribué de grands ensembles de données sur des clusters d'ordinateurs en utilisant des modèles de programmation simples. HDFS est le principal système de fichiers utilisé par Hadoop pour distribuer le travail aux nœuds du cluster.

Apache Spark est un projet de logiciel open source qui fournit un moteur d'analyse de haute performance pour le traitement de données par lots et en continu. Spark peut être jusqu'à 100 fois plus rapide que les tâches équivalentes de Hadoop car il exploite le traitement en mémoire. Spark fournit également quelques abstractions pour le traitement des données, notamment des ensembles de données et des cadres de données distribués et résilients. Spark, en particulier, est très puissant et expressif et est utilisé pour de nombreuses charges de travail.

Une grande partie de la complexité et de la surcharge de travail d'OSS Hadoop est liée aux hypothèses de conception qui existaient dans le centre de données. Libéré de ces limitations, le traitement des données devient une solution beaucoup plus riche avec beaucoup plus d'options.

OSS Hadoop présente deux problèmes courants : le réglage et l'utilisation. Une entreprise aura généralement plusieurs clusters Hadoop qui sont partagés par

plusieurs organisations et qui gèrent une grande variété de tâches. Les experts de Hadoop doivent ajuster de nombreux paramètres de configuration dans la collection de logiciels de projet open source sous-jacents afin d'optimiser le cluster pour les différents types de travail qu'il est chargé d'effectuer. Il s'agit d'un problème de réglage. Les clusters Hadoop ont tendance à nécessiter beaucoup de matériel dédié, ce qui les rend coûteux lorsqu'ils ne sont pas utilisés. Il s'agit du problème de l'utilisation. Les administrateurs de Hadoop peuvent trouver qu'ils cherchent dans les organisations des tâches de traitement, afin de pouvoir augmenter l'utilisation des clusters. S'ils réussissent, la capacité commencera à être consommée et il sera temps de commander plus de matériel. Ce cycle de réglage, de sous-utilisation, de sur-utilisation, d'expansion crée des frais généraux importants pour Hadoop.

Dans de nombreux cas, l'utilisation de Cloud Dataproc tel qu'il a été conçu permettra de surmonter ces limitations.

## Les clusters Hadoop sur site présentent un certain nombre de limites



Pas évolutifs



Difficiles à faire évoluer rapidement



Présentent des limites de capacité



Ne séparent pas les ressources de stockage et de calcul

# Cloud Dataproc simplifie la charge de travail de Hadoop sur les GCP



Support intégré pour Hadoop



Matériel et configuration gérés



Gestion simplifiée des versions



Configuration flexible des tâches



Il existe de nombreuses façons d'utiliser GCP qui peuvent vous faire économiser du temps, de l'argent et des efforts par rapport à l'utilisation d'une solution Hadoop sur site. Dans de nombreux cas, l'adoption d'une approche basée sur le Cloud peut rendre votre solution globale plus simple et plus facile à gérer.

## **Support intégré pour Hadoop**

GCP inclut Cloud Dataproc, qui est un environnement géré Hadoop et Spark. Vous pouvez utiliser Cloud Dataproc pour exécuter la plupart de vos tâches existantes avec un minimum de modifications, vous n'avez donc pas besoin de vous éloigner de tous les outils Hadoop que vous connaissez déjà.

## **Matériel et configuration gérés**

Lorsque vous exécutez Hadoop sur GCP, vous n'avez jamais besoin de vous soucier du matériel physique. Vous spécifiez la configuration de votre cluster, et Cloud Dataproc vous alloue des ressources. Vous pouvez faire évoluer votre cluster à tout moment.

## **Gestion simplifiée des versions**

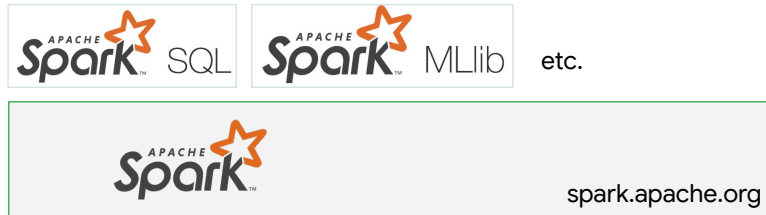
Maintenir les outils open source à jour et les faire collaborer est l'une des parties les plus complexes de la gestion d'un cluster Hadoop. Lorsque vous utilisez Cloud Dataproc, une grande partie de ce travail est gérée pour vous par la gestion de version de Cloud Dataproc.

## **Configuration flexible des tâches**

Une installation Hadoop typique sur site utilise un seul cluster qui sert à de nombreuses fins. Lorsque vous passez à GCP, vous pouvez vous concentrer sur des tâches individuelles, en créant autant de clusters que nécessaire. Cela vous permet d'éliminer une grande partie de la complexité liée à la maintenance d'un seul cluster avec des dépendances et des interactions de configuration logicielle croissantes.



# Apache Spark est un moyen populaire, flexible et puissant de traiter de grands ensembles de données



Exécuter MapReduce directement au dessus de Hadoop est très utile. Mais il est compliqué par le fait que le système Hadoop doit être « réglé » pour le type de travail à effectuer afin d'utiliser efficacement les ressources sous-jacentes. Imaginons une tâche s'exécutant sur des millions de données de capteurs provenant d'une application de l'Internet des objets. Et imaginons une tâche s'exécutant sur ces énormes photos de l'exemple précédent. Essayer de faire les deux choses en même temps de manière efficace est compliqué. C'est là que Spark, une innovation importante intervient. Pour faire simple, Spark est capable de mélanger différents types d'applications et d'ajuster la façon dont il utilise les ressources disponibles.

Vous devez apprendre à programmer Spark différemment de la programmation traditionnelle. Car vous ne pouvez pas lui dire comment faire les choses. Pour conférer à Spark la flexibilité dont il a besoin pour déterminer comment utiliser les ressources disponibles, vous devez décrire ce que vous voulez et laisser Spark déterminer comment y parvenir. C'est ce qu'on appelle la programmation déclarative par opposition à la programmation impérative. Dans la programmation impérative, vous dites au système ce qu'il doit faire et comment il doit le faire. Dans la programmation déclarative, vous dites au système ce que vous voulez et il détermine comment le mettre en œuvre. Dans ce cours, vous apprendrez à travailler avec Spark lors des ateliers.

Il existe une implémentation complète de SQL en plus de Spark. Il existe un modèle DataFrame commun qui fonctionne avec Scala, Java, Python, SQL et R. Et il existe une bibliothèque de machine learning distribuée appelée Spark ML-lib.

### **Notes des étudiants :**

Vous connaissez peut-être Spark en tant qu'outil de traitement de données à grande échelle, open source et général, comme Apache Hadoop MapReduce, dont il s'agit. Mais de nombreuses couches ont été construites par-dessus.

Une implémentation SQL complète a été écrite au-dessus, qui fournit un modèle de données DataFrame commun à Scala, Java, SQL, R et Python.

Et en plus de cela, il existe une bibliothèque de machine learning distribuée de Spark MLlib Spark.

# Programme

---

L'écosystème Hadoop

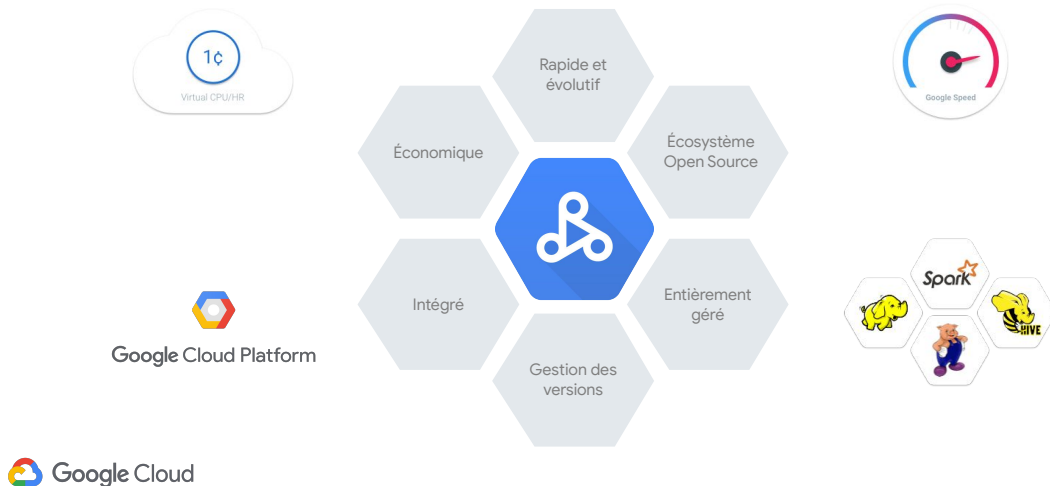
Exécuter Hadoop dans  
Cloud Dataproc

GCS au lieu de HDFS

Optimiser Dataproc

Atelier

# Cloud Dataproc est un service géré pour gérer les charges de travail de traitement des données Hadoop et Spark



Cloud Dataproc est un service de Spark et Hadoop géré, qui vous permet de profiter d'outils de données open source pour le traitement par lots, la soumission de requêtes, le streaming et le machine learning. L'automatisation de Cloud Dataproc permet de créer des clusters rapidement, de les gérer facilement et de réaliser des économies en désactivant les clusters inutilisés.

Par rapport aux produits traditionnels sur site et aux services Cloud concurrents, Cloud Dataproc présente des avantages uniques pour les clusters de trois à plusieurs centaines de nœuds.

Il n'est pas nécessaire d'apprendre de nouveaux outils ou API pour utiliser Cloud Dataproc, ce qui permet de transférer facilement des projets existants dans Cloud Dataproc sans avoir à les redévelopper. Spark, Hadoop, Pig et Hive sont fréquemment mis à jour.

Voici quelques-unes des principales caractéristiques de Cloud Dataproc :

- **Économique** : Le prix de Cloud Dataproc est de 1 cent par processeur virtuel et par cluster par heure, en plus des autres ressources GCP que vous utilisez. En outre, les clusters Cloud Dataproc peuvent inclure des instances préemptives dont le prix de traitement est inférieur. Vous n'utilisez et ne payez les éléments que lorsque vous en avez besoin, c'est pourquoi Cloud Dataproc facture à la seconde près avec une période de facturation minimale d'une minute.
- **Super rapide** : Le lancement, le dimensionnement et la fermeture des

- clusters Cloud Dataproc sont bien plus rapides, chacune de ces opérations prenant en moyenne 90 secondes ou moins.
- **Clusters redimensionnables** : Les clusters peuvent être créés et mis à l'échelle rapidement grâce à une variété de types de machines virtuelles, de tailles de disques, de nombre de nœuds et d'options de mise en réseau.
- **Écosystème open source** : Vous pouvez utiliser les outils, les bibliothèques et la documentation Spark et Hadoop avec Cloud Dataproc. Cloud Dataproc fournit des mises à jour fréquentes des versions natives de Spark, Hadoop, Pig et Hive. Il n'est donc pas nécessaire d'apprendre de nouveaux outils ou API, et il est possible de déplacer des projets existants ou des pipelines ETL sans redéveloppement.
- **Intégré** : L'intégration complète avec Cloud Storage, BigQuery et Cloud Bigtable garantit que les données ne seront pas perdues. Cette solution, associée à Stackdriver Logging et Stackdriver Monitoring, fournit une plateforme de données complète et pas seulement un cluster Spark ou Hadoop. Par exemple, vous pouvez utiliser Cloud Dataproc pour extraire, transformer et charger sans effort des téraoctets de données de journaux brutes directement dans BigQuery pour les rapports d'activité.
- **Géré** : Interagissez facilement avec les clusters et les travaux Spark ou Hadoop, sans l'aide d'un administrateur ou d'un logiciel spécial, grâce à la console GCP, au SDK Cloud ou à l'API REST de Cloud Dataproc. Lorsque vous avez terminé un cluster, il suffit de le désactiver pour éviter de dépenser de l'argent dans un cluster inutilisé.
- **Gestion des versions** : Le gestion des versions des images vous permet de passer d'une version à l'autre d'Apache Spark, d'Apache Hadoop et d'autres outils.

En outre :

- **Disponibilité élevée** : Exécutez des clusters avec plusieurs nœuds maîtres et configurez les tâches pour qu'elles redémarrent en cas d'échec afin de garantir la haute disponibilité de vos clusters et de vos tâches.
- **Outils de développeur** : Plusieurs façons de gérer un cluster, y compris la console GCP, le SDK Cloud, les API RESTful et l'accès SSH.
- **Actions d'initialisation** : Exécutez des actions d'initialisation pour installer ou personnaliser les paramètres et les bibliothèques dont vous avez besoin lors de la création de votre cluster.
- **Configuration manuelle ou automatique** : Cloud Dataproc configure automatiquement pour vous le matériel et les logiciels des clusters tout en permettant un contrôle manuel.

## Il existe d'autres options OSS disponibles dans Cloud Dataproc

|                        |                   |                   |
|------------------------|-------------------|-------------------|
| Spark (par défaut)     | Hive (par défaut) | HDFS (par défaut) |
| Pig (par défaut)       | Zeppelin          | Zookeeper         |
| Kafka                  | Hue               | Tez               |
| Presto                 | Anaconda          | Cloud SQL Proxy   |
| Jupyter                | Apache Flink      | Cloud Datalab     |
| IPython                | Oozie             | Sqoop             |
| Et bien plus encore... |                   |                   |



Cloud Dataproc propose deux façons de personnaliser les clusters : des composants optionnels et des actions d'initialisation. Les composants optionnels préconfigurés peuvent être sélectionnés lors du déploiement depuis la console ou via la ligne de commande et comprennent : Anaconda, Hive WebHCat, Jupyter Notebook, Zeppelin Notebook, Druid, Presto et Zookeeper.

Les Actions d'initialisation vous permettent de personnaliser votre cluster en spécifiant les exécutable ou les scripts que Cloud Dataproc exécutera sur tous les nœuds de votre cluster Cloud Dataproc immédiatement après la mise en place du cluster. Vous pouvez définir vos propres scripts d'initialisation ou choisir parmi un large éventail d'actions d'initialisation fréquemment utilisées et d'autres exemples, comme indiqué ici :

[https://cloud.google.com/dataproc/docs/concepts/configuring-clusters/init-actions#examplewzxhjdk15staging\\_binaries/?hl=fr](https://cloud.google.com/dataproc/docs/concepts/configuring-clusters/init-actions#examplewzxhjdk15staging_binaries/?hl=fr)

# Utiliser les actions d'initialisation pour ajouter d'autres logiciels au cluster au démarrage

Utilisez **des actions d'initialisation** pour ajouter des composants supplémentaire sur le cluster.



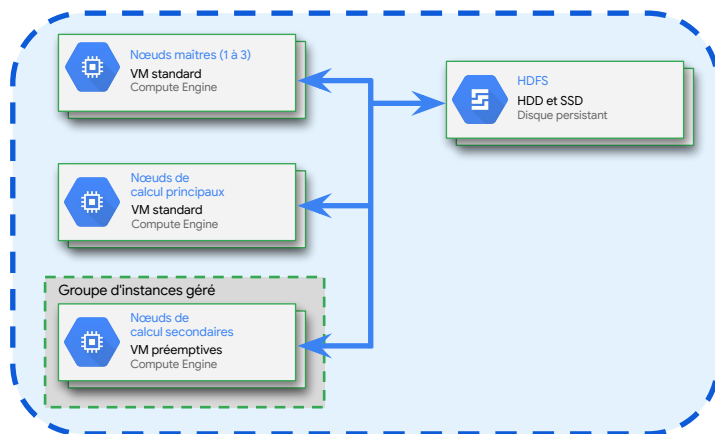
```
Les clusters gcloud dataproc créent <CLUSTER_NAME> \  
--initialization-actions gs://$MY_BUCKET/hbase/hbase.sh \  
--num-masters 3 --num-workers 2
```



<https://github.com/GoogleCloudPlatform/dataproc-initialization-actions> (Flink, Jupyter, Oozie, Presto, Tez, HBase, etc.)

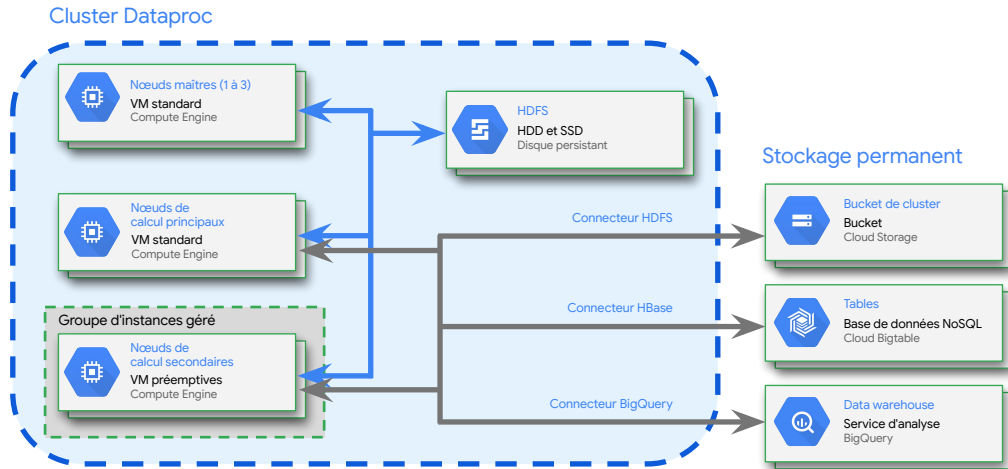
# Un cluster Dataproc a des nœuds maîtres, des travailleurs et HDFS

Cluster Dataproc





# Le cluster Dataproc peut lire/écrire sur les produits de stockage GCP



# Utilisation de Cloud Dataproc



L'utilisation de Cloud Dataproc implique cette séquence d'événements : Installation, configuration, optimisation, utilisation et surveillance.

L'installation implique la création d'un cluster. Et vous pouvez exécuter cette opération via la console, depuis la ligne de commande en utilisant la commande `gcloud`. Vous pouvez également exporter un fichier YAML à partir d'un cluster existant ou créer un cluster à partir d'un fichier YAML. Vous pouvez créer un cluster à partir d'un modèle de Deployment Manager, ou utiliser l'API REST.

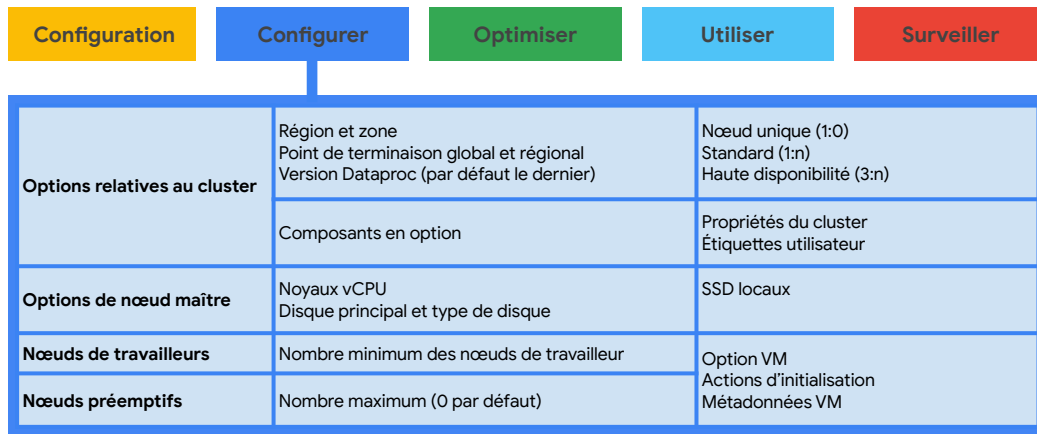
La documentation sur la création d'un cluster se trouve ici :

<https://cloud.google.com/dataproc/docs/guides/create-cluster/?hl=fr>

Les propriétés du cluster peuvent être utilisées pour modifier les fichiers de configuration OSS courants.

<https://cloud.google.com/dataproc/docs/concepts/configuring-clusters/cluster-properties/?hl=fr>

# Configurer



Le cluster peut être défini comme une VM unique, ce qui permet généralement de réduire les coûts de développement et d'expérimentation. La configuration standard est avec un seul nœud maître, et la haute disponibilité a trois nœuds maîtres. Vous pouvez choisir une région et une zone, ou sélectionner une « région globale » et permettre au service de choisir la zone pour vous. Par défaut, le cluster est global, mais la définition d'un point d'extrémité régional peut offrir un isolement accru et, dans certains cas, une latence plus faible.

Le nœud maître est l'endroit où fonctionne le HDFS Namenode, ainsi que le nœud YARN et les pilotes de tâches. La réplication du HDFS a pour valeur par défaut 2 dans Cloud Dataproc.

Les composants en option de l'écosystème Hadoop sont les suivants : Anaconda (distribution Python et gestionnaire de package), Hive Webcat, Jupyter Notebook, Zeppelin Notebook

Les propriétés des clusters sont des valeurs d'exécution qui peuvent être utilisées par les fichiers de configuration pour des options de démarrage plus dynamiques. Et les étiquettes utilisateur peuvent être utilisées pour marquer le cluster pour vos propres solutions ou à des fins de génération de rapports.

Les nœuds de travailleur du nœud maître et les nœuds de travailleur préemptifs, s'ils sont activés, disposent d'options VM distinctes, telles que vCPU, mémoire et stockage.

Les nœuds préemptifs incluent YARN NodeManager mais n'exécutent pas HDFS.

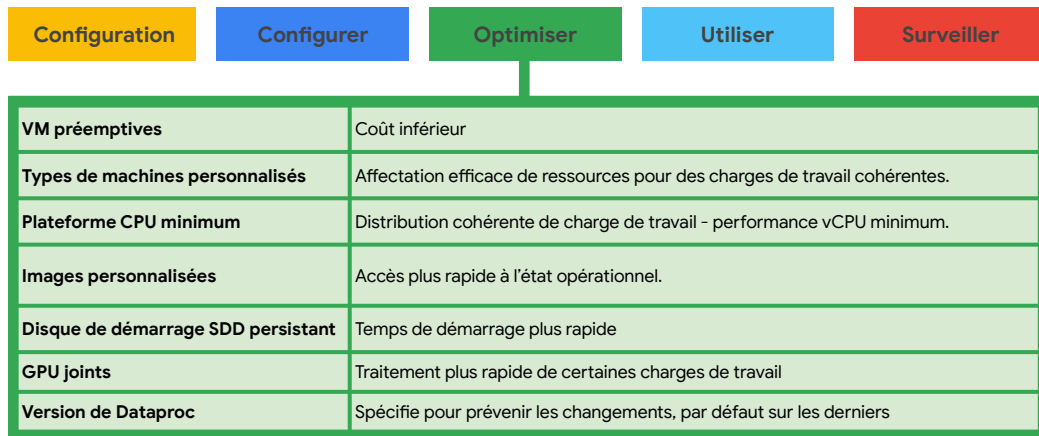
Il existe un nombre minimum de nœuds de travailleurs, la valeur par défaut est 2. Le nombre maximum de nœuds de travailleurs est déterminé par un quota et le nombre de SSD attachés à chaque travailleur.

Vous pouvez également spécifier des actions d'initialisation, telles que des scripts d'initialisation qui peuvent personnaliser davantage les nœuds de travailleur. Et des métadonnées peuvent être définies pour que les VM puissent partager des informations d'état.

Les propriétés du cluster modifient les valeurs communes des fichiers de configuration des OSS.

<https://cloud.google.com/dataproc/docs/concepts/configuring-clusters/cluster-properties/?hl=fr>

# Optimiser



Les VM préemptives peuvent être utilisées pour réduire les coûts. N'oubliez pas qu'elles peuvent être retirées du service à tout moment et dans les 24 heures. Votre application doit donc être conçue de manière à être résiliente afin d'éviter toute perte de données.

Les types de machine personnalisés vous permettent de spécifier l'équilibre entre la mémoire et le processeur pour adapter la VM à la charge, afin de ne pas gaspiller de ressources.

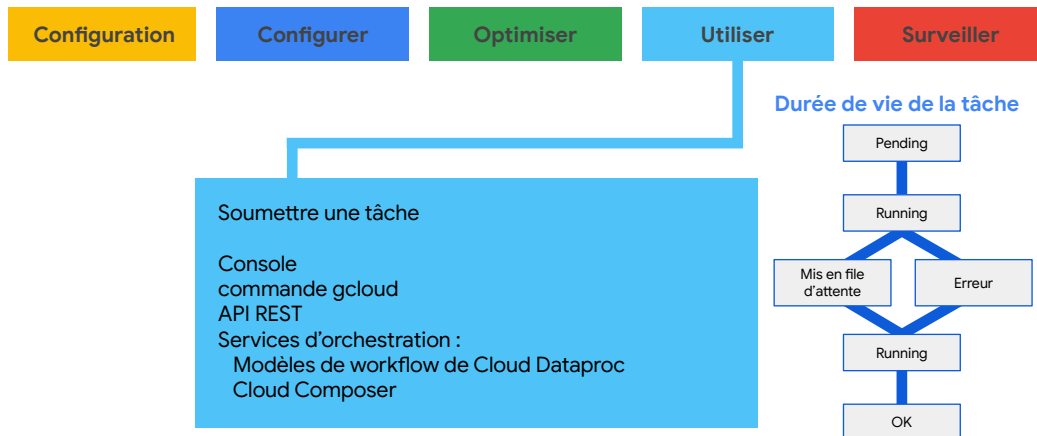
Une image personnalisée peut être utilisée pour préinstaller un logiciel de sorte qu'il faut moins de temps pour que le nœud personnalisé devienne opérationnel que si vous aviez installé le logiciel sont l'utilisation du temps de démarrage et le script d'initialisation.

Création et utilisation d'une image personnalisée :

<https://cloud.google.com/dataproc/docs/guides/dataproc-images/?hl=fr>

Vous pouvez obtenir un disque de démarrage SSD persistant pour accélérer le démarrage du cluster.

## Utiliser : Soumission de tâche



Les tâches peuvent être soumises via la console, la commande gcloud ou l'API REST. Elles peuvent également être lancées par des services d'orchestration tels que Cloud Dataproc Workflow et Cloud Composer.

N'utilisez pas les interfaces directes de Hadoop pour soumettre des tâches, car les métadonnées ne seront pas disponibles pour Cloud Dataproc pour la gestion des tâches et des clusters, et pour des raisons de sécurité, elles sont désactivées par défaut.

Durée de vie de la tâche :

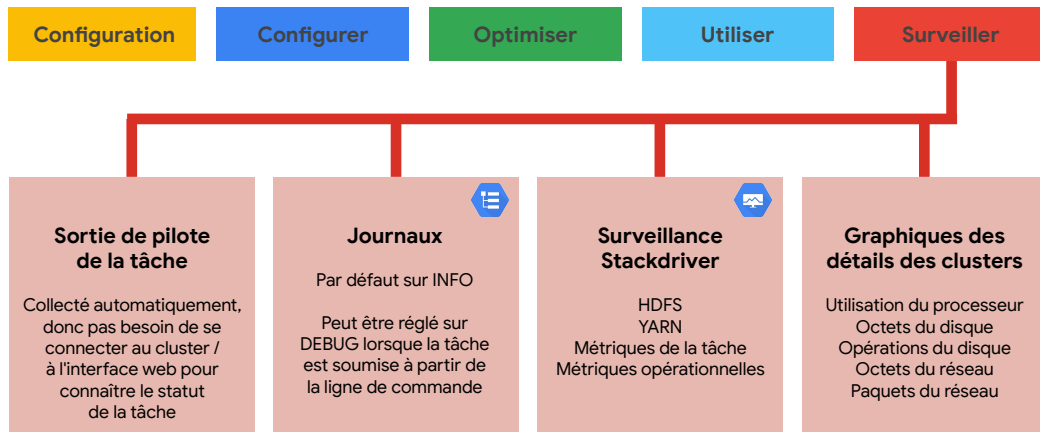
<https://cloud.google.com/dataproc/docs/concepts/jobs/life-of-a-job/?hl=fr>

Par défaut, il est impossible de redémarrer les tâches. Cependant, vous pouvez créer des tâches redémarrables via la ligne de commande ou l'API REST. Les tâches redémarrables doivent être conçues pour être idempotents et pour détecter les successeurs et restaurer l'état.

Tâches redémarrables :

<https://cloud.google.com/dataproc/docs/concepts/jobs/restartable-jobs/?hl=fr>

# Surveiller à travers la console et Stackdriver



À l'aide de Stackdriver, vous pouvez créer un tableau de bord personnalisé avec des graphiques et définir une politique d'alerte de surveillance pour signaler les incidents.

<https://cloud.google.com/dataproc/docs/guides/stackdriver-monitoring/?hl=fr>

## Page Cluster

État : comme « en cours d'exécution »

Région

Date et heure de création

Nombre total de nœuds de travailleur

La page Détails sur le cluster comporte des graphiques de la charge sur le cluster.

Graphiques :

Utilisation du processeur

Octets du disque

Opérations du disque

Octets du réseau

Paquets du réseau

# Programme

---

L'écosystème Hadoop

Exécuter Hadoop dans  
Cloud Dataproc

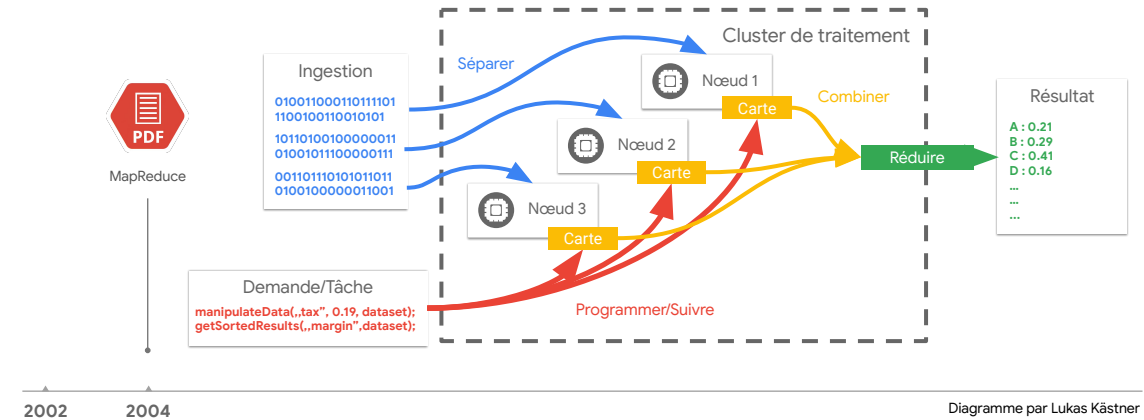
GCS au lieu de HDFS

Optimiser Dataproc

Atelier



Le papier MapReduce original a été conçu pour un monde où les données restaient dans la machine de calcul



Les vitesses du réseau étaient lentes au départ, c'est la raison pour laquelle nous avons gardé les données aussi près que possible du processeur. Maintenant, avec le réseau pétabit, vous pouvez traiter le stockage et calculer indépendamment et déplacer rapidement le trafic sur le réseau.

# HDFS dans le Cloud est une solution de sous-partie

## Taille des blocs

Par défaut sur 64 Mo  
(souvent augmenté à 128 Mo)

Détermine le parallélisme  
de l'exécution

Echelles d'E/S avec taille de disque et  
noyaux VM (jusqu'à 2 To et 8 noyaux)

Accessible uniquement à partir  
d'un nœud unique (en mode RW)

Le calcul et le stockage  
ne sont pas indépendants,  
ce qui augmente les coûts

## Localité

HDFS répartir les blocs

La plupart des moteurs d'exécution  
sur les HDFS connaissent la localité

Si vous utilisez des disques  
persistants, la localité des  
données ne tient plus

## Réplication

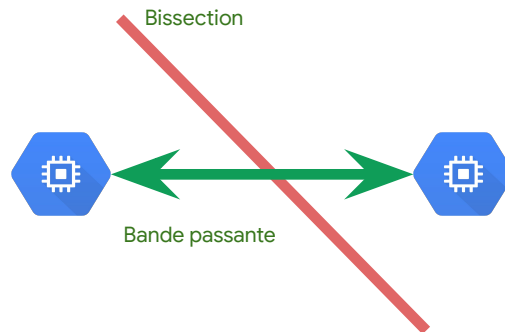
Par défaut sur 3 copies de chaque  
bloc ( $r=3$ )

Toujours besoin de  $r=2$  sur HDFS,  
pour la disponibilité

- Les serveurs Cloud Dataproc  
doivent transmettre  $2 \times 3 = 6$   
copies des blocs HDFS à Colossus.

La réplication de nombreuses  
données rend cette opération  
coûteuse

## La bande passante en petabits change la donne pour big data



Traitement des données là où elles se trouvent sans les copier

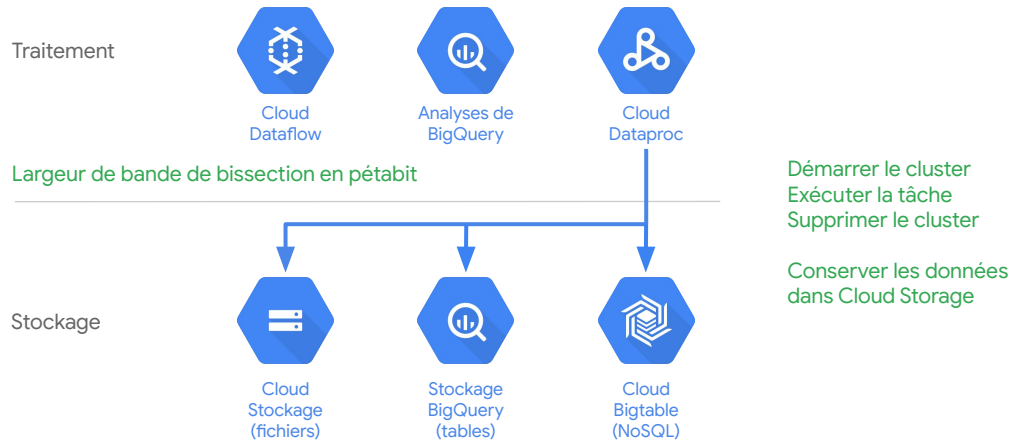


Le réseau de Google permet de nouvelles solutions pour Big Data. La structure de réseau Jupiter au sein d'un centre de données Google offre plus de 1 Po/s de bande passante.

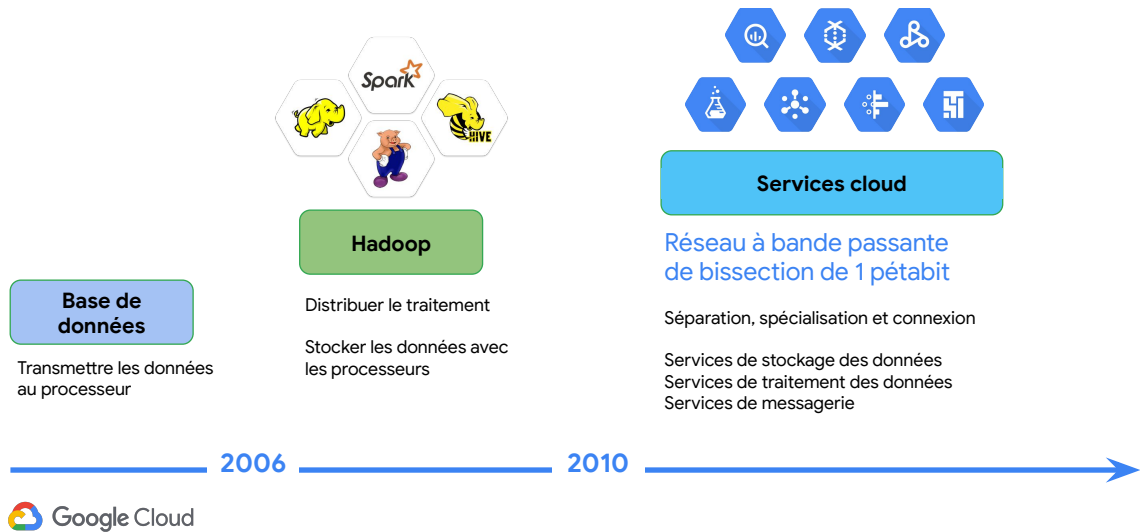
<https://cloudplatform.googleblog.com/2015/06/A-Look-Inside-Googles-Data-Center-Networks.html>. Pour mettre cela en perspective, cela représente environ deux fois la quantité de trafic échangé sur l'ensemble de l'Internet public. (voir l'estimation annuelle de Cisco sur l'ensemble du trafic Internet)

Si vous tracez une ligne à un emplacement quelconque d'un réseau, la bande passante de bissection désigne la vitesse à laquelle les serveurs situés d'un côté de cette ligne peuvent communiquer avec les serveurs placés de l'autre côté. Une bande passante de bissection suffisamment élevée permet à un serveur quelconque de communiquer avec tout autre serveur en bénéficiant du débit réseau maximal. Avec une bande passante de bissection d'un pétabit/s, la communication est si rapide qu'il n'y a plus lieu de transférer les fichiers et de les stocker localement. Il est préférable d'utiliser les données directement à l'endroit où elles sont stockées.

# Sur GCP, Jupyter et Colossus rendent possible la séparation du calcul et du stockage



## La séparation du calcul et du stockage offre de meilleures options



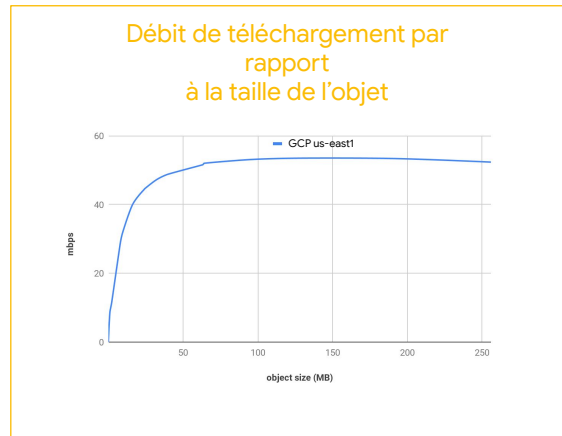
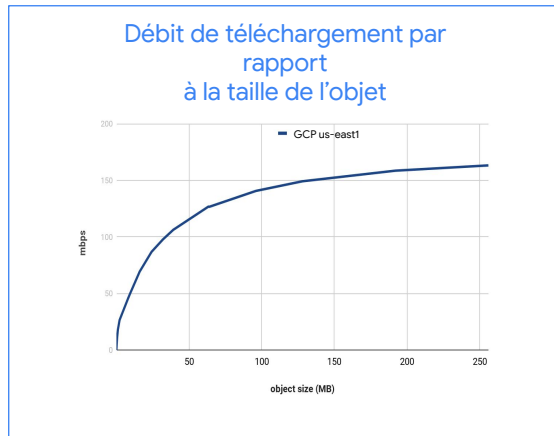
Il permet de replacer les services que vous allez découvrir dans leur contexte historique. Avant 2006, les big data étaient synonymes de grandes bases de données. La conception des bases de données est née à une époque où le stockage était relativement bon marché et le traitement coûteux. Il était donc logique de copier les données de leur lieu de stockage vers le processeur pour effectuer le traitement des données. Le résultat était ensuite recopié sur le stockage.

Vers 2006, le traitement distribué de données volumineuses est devenu plus pratique grâce à Hadoop. L'idée qui sous-tend Hadoop est de créer un cluster d'ordinateurs et de tirer parti du traitement distribué. HDFS - le système de fichiers distribués de Hadoop stockait les données sur les machines du cluster, et Map Reduce assurait le traitement distribué des données. Tout un écosystème de logiciels liés à Hadoop s'est développé autour de Hadoop, dont Hive, Pig et Spark.

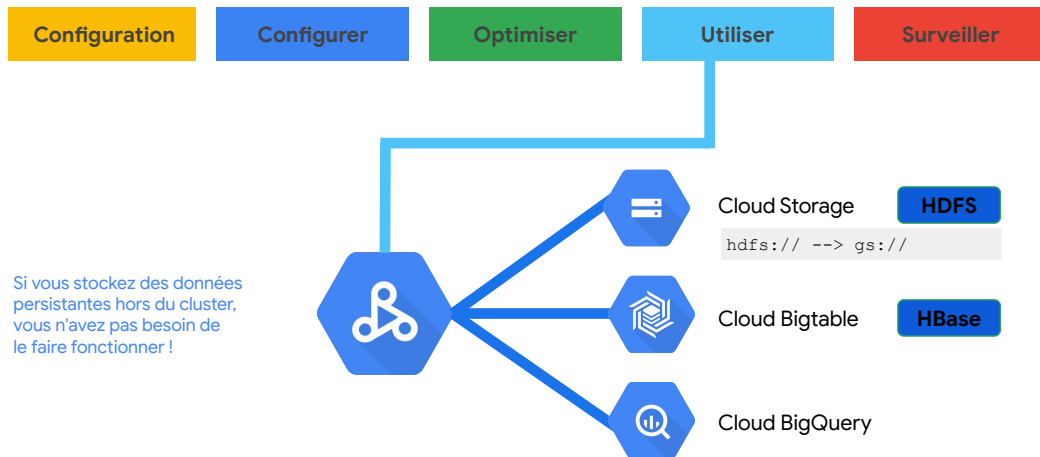
BigQuery, le premier des nombreux services de big data conçus par Google, est apparu sur le marché aux alentours de 2010. Vers 2015, Google a lancé Cloud Dataproc, un service géré permettant de créer des clusters Hadoop et Spark et de gérer des charges de travail de traitement des données.

2003, 2004 -- Système de fichiers Google  
2006, 2008 -- Hadoop  
2015 -- Cloud Dataproc  
BigQuery -- 2010  
Cloud Dataflow -- 2014

## Performances de débit GCP (à un point donné du temps)



# Le stockage à l'extérieur du cluster est un gage d'efficacité



Il suffit d'utiliser les HDFS sur le cluster pour le stockage de travail pendant le traitement.

Cela permet de créer le cluster pour une seule tâche ou un seul type de charge de travail et de l'arrêter lorsqu'il n'est pas utilisé.

Il existe un connecteur HDFS pour Cloud Storage. Vous pouvez remplacer hdfs:// par gs://

Il existe un connecteur HDFS pour Cloud Bigtable.

Le connecteur BigQuery est utilisé pour traiter les données stockées dans un entrepôt de données dans BigQuery.

# Utilisez Cloud Storage au lieu de HDFS avec Cloud Dataproc

Configuration

Configurer

Optimiser

Utiliser

Surveiller

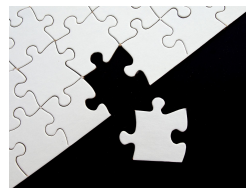


Cloud Storage est  
un service distribué

Élimine les goulets  
d'étranglement traditionnels  
et les points d'échec uniques



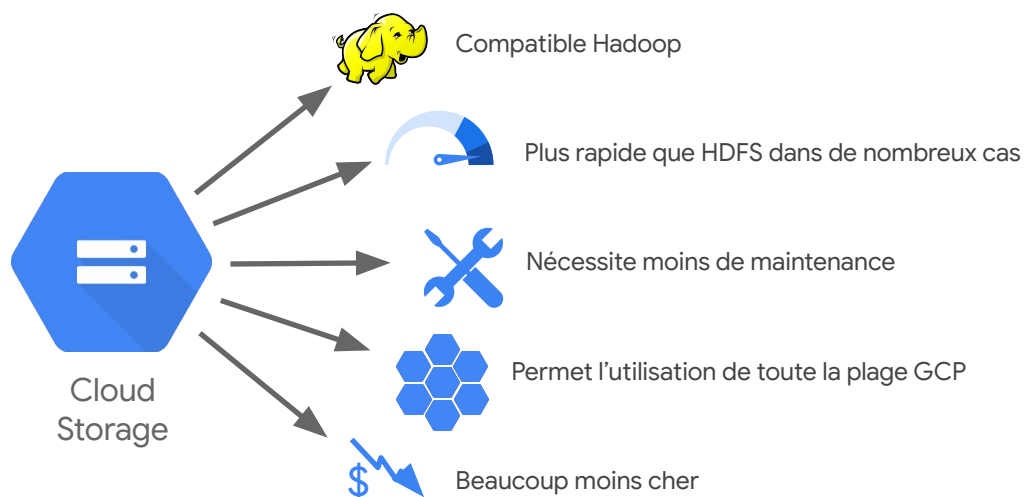
Les répertoires sont simulés,  
donc renommer  
un répertoire implique de  
renommer tous les objets\*



Les objets ne prennent  
pas en charge "append"



## Utiliser Cloud Storage comme stockage de données persistantes



# Cloud Storage est une solution de remplacement des HDFS



Interfaces du système de fichiers Hadoop - compatible "HCFS" (Hadoop Compatible File System)  
File[Input|Output]Format, SparkContext.textFile, etc., fonctionne



Cloud Dataflow : Déduplication, ordre et fenêtrage



HDFS - vous devez surprovisionner les données actuelles, mais aussi les données que vous pourriez avoir, et vous devez utiliser des disques persistants tout au long du processus.

Cloud Storage - payez exactement ce dont vous avez besoin, quand vous l'utilisez.

# Bonnes pratiques des performances

## Optimiser pour les opérations en vrac/parallèles

- ✓ Évitez les petites lectures ; utilisez si possible des blocs de grande taille
- ✓ Évitez d'itérer successivement sur plusieurs répertoires imbriqués dans une seule tâche



## Le renommage des répertoires dans les HDFS n'est pas le même que dans Cloud Storage

Cloud Storage ne connaît pas le concept des répertoires !

```
mv gs://foo/bar/ gs://foo/bar2
```

- `list(gs://foo/bar/)`
- `copy({gs://foo/bar/baz1, gs://foo/bar/baz2}, {gs://foo/bar2/baz1, gs://foo/bar2/baz2})`
- `delete({gs://foo/bar/baz1, gs://foo/bar/baz2})`

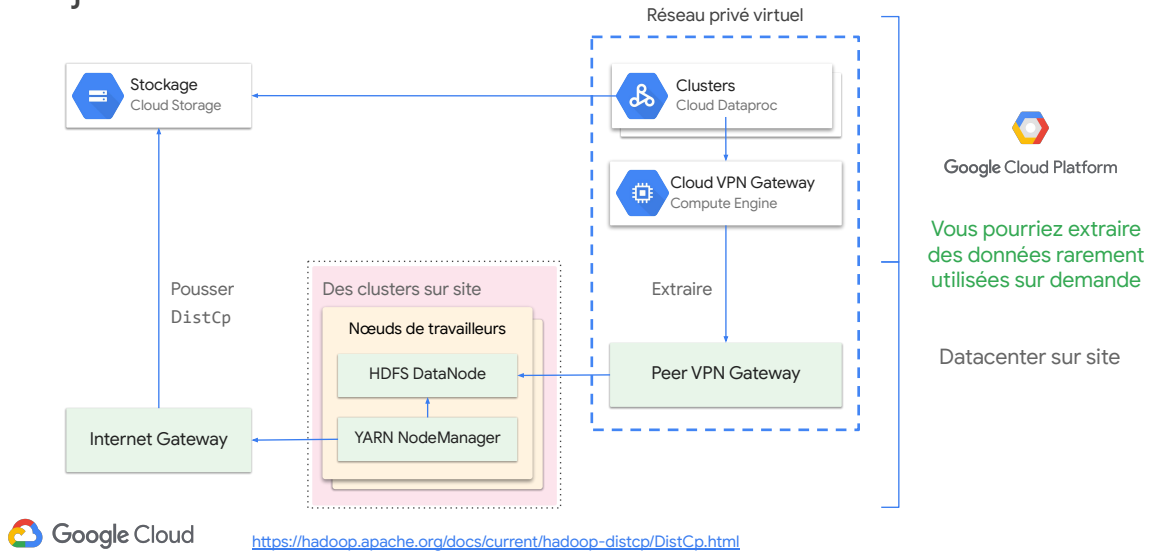
Le code migré devrait permettre de gérer les incohérences de la liste lors du renommage !

- Les committers de formats de sortie modernes gèrent correctement les magasins d'objets



Le renommage des répertoires dans hdfs n'est pas le même que dans Cloud Storage,  
Les nouveaux « committers » de sortie orientés vers le stockage d'objets atténuent ce problème.

## Les données DistCp on-prem dont vous aurez toujours besoin



Mentionnez le distcp comme un outil clé pour le déplacement des données. En général, vous souhaitez utiliser un modèle basé sur le push pour toutes les données nécessaires, tandis que le pull peut être un modèle utile si vous avez beaucoup de données que vous n'aurez peut-être jamais besoin de migrer.

# Programme

---

L'écosystème Hadoop

Exécuter Hadoop dans  
Cloud Dataproc

GCS au lieu de HDFS

Optimiser Dataproc

Atelier

# Questions sur les performances de Hadoop et Spark pour toutes les architectures de clusters, y compris Cloud Dataproc

- 1 Où sont vos données et où se trouve votre cluster ?
- 2 Le trafic de votre réseau est-il filtré ?
- 3 Combien de fichiers d'entrée et de partitions Hadoop essayez-vous de traiter ?
- 4 La taille de votre disque persistant limite-t-elle votre débit ?
- 5 Avez-vous alloué suffisamment de machines virtuelles (VM) à votre grappe ?



## Où sont vos données et où se trouve votre cluster ?

Connaître la localisation de vos données peut avoir un impact majeur sur vos performances. Vous voulez être sûr que la région de vos données et la zone de votre cluster sont physiquement proches en distance.

Lorsque vous utilisez Cloud Dataproc, vous pouvez omettre la zone et demander à la fonction Cloud Dataproc Auto Zone de sélectionner une zone pour vous dans la région de votre choix. Bien que cette fonction pratique puisse optimiser l'emplacement de votre cluster, elle ne sait pas comment anticiper l'emplacement des données auxquelles votre cluster aura accès. Assurez-vous que le bucket Cloud Storage se trouve dans le même emplacement régional que votre région Dataproc.

## Le trafic de votre réseau est-il filtré ?

Assurez-vous que vous n'avez pas de règles de réseau ou de routes qui canalisent le trafic Cloud Storage à travers un petit nombre de passerelles VPN avant qu'il n'atteigne votre cluster. Il existe de grands tuyaux de réseau entre Cloud Storage et le moteur de calcul. Vous ne voulez pas réduire votre bande passante en envoyant le trafic dans un goulot d'étranglement dans votre configuration de réseau GCP.

## Combien de fichiers d'entrée et de partitions Hadoop essayez-vous de traiter ?

Assurez-vous que vous ne traitez pas plus de 10 000 fichiers d'entrée environ. Si vous vous trouvez dans cette situation, essayez de combiner ou de « syndiquer » les données dans des fichiers de plus grande taille. Si cette réduction du volume des fichiers signifie que vous travaillez maintenant avec des ensembles de données plus importants (plus de ~50k partitions Hadoop), vous devriez envisager d'ajuster le

paramètre `fs.gs.block.size` à une valeur plus importante en conséquence.

« `fs.gs.block.size` » est un paramètre de configuration qui aide les tâches à effectuer des fractionnements. Lorsqu'il est utilisé en conjonction avec des données Cloud Storage, il devient une « fausse » valeur puisque Cloud Storage n'expose pas les tailles de bloc réelles. La valeur par défaut de 64 Mo sert uniquement à aider Hadoop à décider comment effectuer des fractionnements. Par conséquent, si les fichiers sont supérieurs à 512 Mo, vous pouvez constater que vous pouvez obtenir de meilleures performances en augmentant manuellement cette valeur jusqu'à 1 Go ou même 2 Go. Contrairement aux disques persistants standard, les performances IOPS des disques persistants SSD dépendent du nombre de vCPU dans l'instance.

### **La taille de votre disque persistant limite-t-elle votre débit ?**

Souvent, lorsque vous commencez à utiliser Google Cloud, vous n'avez qu'un petit tableau à comparer. Il s'agit généralement d'une bonne approche, tant que vous ne choisissez pas un disque persistant dimensionné pour une si petite quantité de données ; cela limitera très probablement vos performances. Les disques persistants standard évoluent de manière linéaire en fonction de la taille du volume.

### **Avez-vous alloué suffisamment de machines virtuelles (VM) à votre grappe ?**

Une question qui se pose souvent lors de la migration d'un matériel sur site vers Google Cloud est de savoir comment dimensionner avec précision le nombre de machines virtuelles nécessaires. Pour déterminer la taille d'un cluster, il est essentiel de comprendre vos charges de travail. L'exécution de prototypes et l'analyse comparative avec des données et des tâches réelles sont essentielles pour prendre une décision éclairée sur l'attribution des machines virtuelles. Heureusement, la nature éphémère du cloud permet de « dimensionner » facilement les clusters pour la tâche spécifique à accomplir au lieu d'essayer d'acheter du matériel à l'avance. Ainsi, vous pouvez facilement redimensionner votre cluster selon vos besoins. L'utilisation de clusters adaptés aux tâches est une stratégie courante pour les clusters Dataproc.

Même si nous savons que les clusters peuvent facilement être agrandis ou réduits, il peut être utile de disposer de calculs rapides lorsque nous envisageons la taille de nos clusters. Pour un exemple de calcul, disons que nous migrons à partir de 50 nœuds physiques, chacun avec 12 noyaux physiques et 2 hyperthreads par noyau.

Il est essentiel de comprendre que, sur Compute Engine, chaque processeur virtuel (vCPU) est mis en œuvre sous la forme d'une technologie Hyperthread matérielle unique sur l'une des plates-formes de processeur disponibles. Dans notre exemple de 50 nœuds comportant chacun 12 cœurs physiques, vous auriez deux options que vous pourriez configurer sur Compute Engine :

- Option 1 : 1,200 4-vCPU VM (300 n1-standard-4)
- Option 2 : 600 8-vCPU VM (150 n1-standard-8)

Lorsque vous choisissez une option, il est également important de prendre en



compte les implications de stockage associées à ce choix. Il y a des limites à la quantité totale de disque persistant que vous pouvez ajouter à chaque VM. La plupart des types d'instance ont une limite de 64 To, ce qui signifie que l'option 2 limiterait les données de votre cluster à 225 To. Dans notre exemple, nous devons nous demander si cela est suffisant ou si nous préférons avoir plus de VM et donc, augmenter notre taille de stockage. Étant donné que les clients déplacent généralement la grande majorité de leur stockage de données à long terme du HDFS vers Cloud Storage lorsqu'ils migrent vers le cloud, les limites de disques persistants sont généralement plus que suffisantes. Cependant, vous pouvez toujours prendre en compte vos charges de travail spécifiques. Voici quelques besoins de stockage qui méritent d'être étudiés :

- Données HBase, y compris la réplication
- Déversements temporaires non répétés
- Ensembles de données intra-pipeline générés au milieu de requêtes Hive, de scripts Pig, de tâches Mahout, ou dans d'autres scénarios

## Les HDFS locaux sont parfois nécessaires

- Les HDFS locaux sont une bonne option si :
- Votre travail nécessite de nombreuses opérations sur les métadonnées
- Vous modifiez fréquemment les données du HDFS ou vous renommez des répertoires.
- Vous utilisez beaucoup l'opération append sur les fichiers HDFS.
- Vous avez des charges de travail qui impliquent de lourdes entrées/sorties.
- Vous avez des charges de travail d'E/S qui sont particulièrement sensibles à la latence.

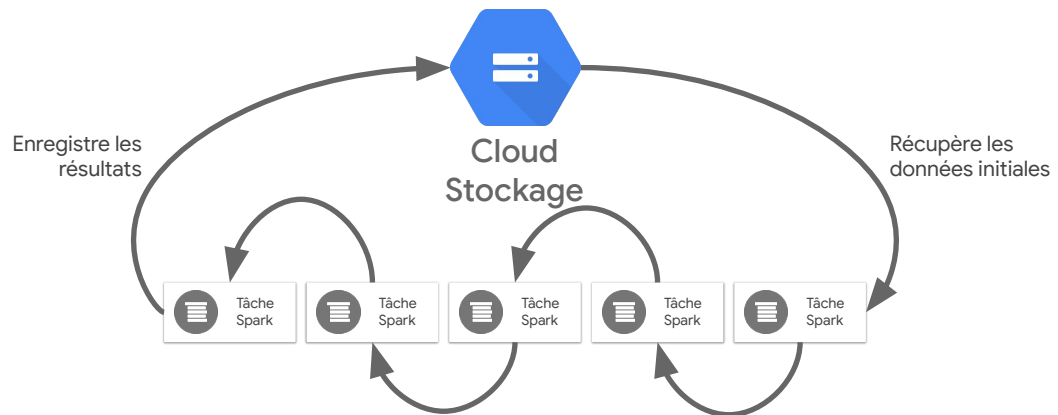


Les HDFS locaux sont une bonne option si :

- Vos tâches nécessitent de nombreuses opérations sur les métadonnées - par exemple, vous avez des milliers de partitions et de répertoires, et la taille de chaque fichier est relativement petite.
- Vous modifiez fréquemment les données du HDFS ou vous renommez des répertoires. (Les objets du Cloud Storage sont immuables, de sorte que renommer un répertoire est une opération coûteuse car elle consiste à copier tous les objets sur une nouvelle clé et à les supprimer ensuite)
- Vous utilisez beaucoup l'opération append sur les fichiers HDFS.
- Vous avez des charges de travail qui impliquent de lourdes entrées/sorties. Par exemple, vous avez beaucoup d'écritures partitionnées, telles que les suivantes :  

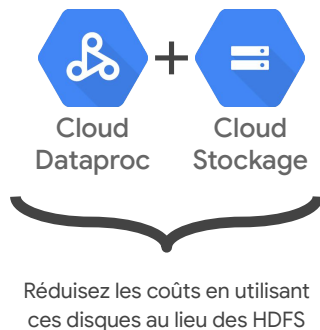
```
spark.read().write.partitionBy(...).parquet("gs://")
```
- Vous avez des charges de travail d'E/S qui sont particulièrement sensibles à la latence. Par exemple, vous avez besoin d'une latence à un chiffre en millisecondes par opération de stockage.

Cloud Storage fonctionne bien en tant que source initiale et finale de données dans un pipeline de données big-data



En général, nous recommandons d'utiliser Cloud Storage en tant que source initiale et finale de données dans un pipeline de données big-data. Par exemple, si un flux de travail contient cinq tâches Spark en série, la première tâche récupère les données initiales de Cloud Storage, puis écrit les données aléatoires et la sortie des tâches intermédiaires dans le HDFS. La dernière tâche Spark écrit ses résultats dans Cloud Storage.

## Utiliser Cloud Dataproc avec Cloud Storage vous permet de réduire les besoins en disques et de réaliser des économies



Voici quelques options pour ajuster la taille du HDFS local :

- Diminuer la taille totale du HDFS local en diminuant la taille des disques persistants principal pour le maître et les travailleurs.
- Augmenter la taille totale du HDFS local en augmentant la taille des disques persistants principal pour le maître et les travailleurs.
- Attachez jusqu'à huit disques SSD (375 Go chacun) à chaque travailleur et utilisez ces disques pour le HDFS.
- Utilisez les disques persistants SSD pour le maître ou les travailleurs comme disque principal.



L'utilisation de Cloud Dataproc avec Cloud Storage vous permet de réduire les besoins en disques et de réaliser des économies en plaçant les données dessus au lieu de les placer dans le HDFS. Lorsque vous conservez vos données sur Cloud Storage et que vous ne les stockez pas sur le HDFS local, vous pouvez utiliser des disques plus petits pour votre cluster. En rendant votre cluster véritablement à la demande, vous pouvez également séparer le stockage et le calcul, comme indiqué précédemment, ce qui vous permet de réduire les coûts de manière significative.

Même si vous stockez toutes vos données dans Cloud Storage, votre cluster Cloud Dataproc a besoin du HDFS pour certaines opérations telles que le stockage des fichiers de contrôle et de récupération, ou l'agrégation des journaux. Il a également besoin d'espace disque local non HDFS pour le brassage. Vous pouvez réduire la taille du disque par travailleur si vous n'utilisez pas intensivement le HDFS local.

Voici quelques options pour ajuster la taille du HDFS local :

Diminuer la taille totale du HDFS local en diminuant la taille des disques persistants principal pour le maître et les travailleurs. Le disque persistant principal contient également le volume de démarrage et les bibliothèques système, allouez donc au moins 100 Go.

Augmenter la taille totale du HDFS local en augmentant la taille des disques persistants principal pour le maître et les travailleurs. Considérez cette option avec soin : il est rare d'avoir des charges de travail qui obtiennent de meilleures performances en utilisant le HDFS avec des disques persistants standard par rapport

à l'utilisation de Cloud Storage ou du HDFS local avec le SSD.

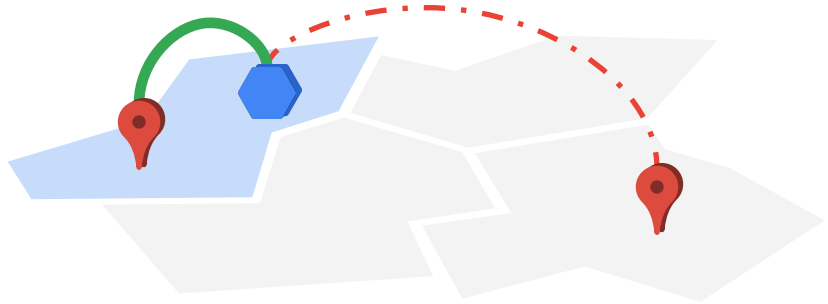
Attachez jusqu'à huit disques SSD (375 Go chacun) à chaque travailleur et utilisez ces disques pour le HDFS. C'est une bonne option si vous devez utiliser le HDFS pour des charges de travail à forte intensité d'E/S et que vous avez besoin d'une latence à un chiffre en millisecondes. Assurez-vous que vous utilisez un type de machine qui dispose de suffisamment de CPU et de mémoire sur le travailleur pour prendre en charge ces disques.

Utilisez les disques persistants SSD (PD\_SSD, actuellement en version bêta) pour le maître ou les travailleurs comme disque principal.

# Les régions géographiques peuvent avoir un impact sur l'efficacité de votre solution

Les régions peuvent avoir des répercussions sur vos tâches, par exemple :

- Demander la latence
- Prolifération des données
- Performance



Vous devez comprendre les répercussions de la géographie et des régions avant de configurer vos données et vos tâches. De nombreux services de GCP vous demandent de spécifier des régions ou des zones dans lesquelles allouer des ressources. La latence des demandes peut augmenter lorsque les demandes proviennent d'une région différente de celle où les ressources sont stockées. En outre, si les ressources du service et vos données persistantes sont situées dans des régions différentes, certains appels aux services GCP peuvent copier toutes les données requises d'une zone à l'autre avant le traitement. Cela peut avoir un impact important sur les performances.

## Les GCP offrent différentes options de stockage pour différentes tâches



Cloud  
Stockage

- Datastore principal pour GCP
- Données non structurées



Cloud  
Bigtable

- Grande quantité de données éparpillées
- Compatible avec HBase
- Faible latence
- Évolutivité élevée



BigQuery

- Entreposage des données
- L'API de stockage permet d'accélérer le processus
- Pourrait pousser les requêtes vers BigQuery, en remaniant la tâche




Cloud Storage est le principal moyen de stocker des données non structurées dans GCP, mais ce n'est pas la seule option de stockage. Certaines de vos données pourraient être mieux adaptées au stockage dans des produits conçus explicitement pour des données volumineuses.

Vous pouvez utiliser Cloud Bigtable pour stocker de grandes quantités de données éparses. Le Cloud Bigtable est une API compatible HBase- qui offre une faible latence et une grande évolutivité pour s'adapter à vos tâches.

Pour l'entreposage de données, vous pouvez utiliser BigQuery.

# La reproduction de votre configuration permanente sur site présente quelques inconvénients


Les clusters persistants  
sont coûteux



Vos outils open-source  
peuvent être inefficaces



Les clusters persistants  
sont difficiles à gérer



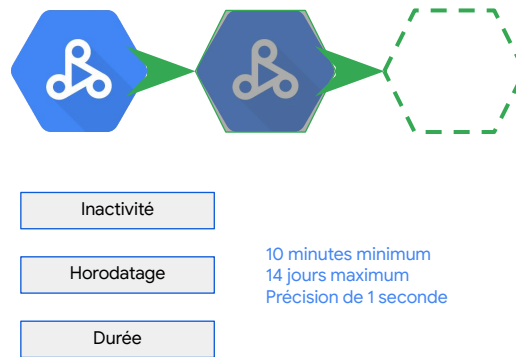
Comme Cloud Dataproc exécute Hadoop sur GCP, utiliser un cluster Cloud Dataproc persistant pour reproduire votre installation sur site peut sembler la solution la plus simple. Toutefois, cette approche présente certaines limites :

- Conserver vos données dans un cluster HDFS persistant en utilisant Cloud Dataproc est plus coûteux que de les stocker dans Cloud Storage, ce que nous recommandons. Le fait de conserver les données dans un cluster HDFS limite également votre capacité à utiliser vos données avec d'autres produits GCP.
- Augmenter ou remplacer certains de vos outils open-source par d'autres services GCP associés peut être plus efficace ou plus économique pour des cas d'utilisation particuliers.
- Il est plus difficile de gérer l'utilisation d'un cluster unique et permanent de Cloud Dataproc pour vos tâches que de passer à des clusters ciblés qui servent des tâches individuelles ou des domaines de tâches.

La manière la plus rentable et la plus souple de faire migrer votre système Hadoop vers GCP est de ne plus penser en termes de grands clusters persistants et polyvalents, mais plutôt en termes de petits clusters de courte durée conçus pour des tâches spécifiques. Vous stockez vos données dans Cloud Storage pour prendre en charge de multiples clusters de traitement temporaires. Ce modèle est souvent appelé le modèle éphémère, car les clusters que vous utilisez pour les tâches de traitement sont attribués selon les besoins et sont libérés lorsque les tâches se terminent.



## Suppression programmée du cluster



Pour une utilisation efficace, ne payez pas pour des ressources que vous n'utilisez pas.

Une durée fixe après l'entrée du cluster dans l'état d'inactivité.

Définissez une minuterie. Vous lui attribuez un horodatage. Le comptage commence immédiatement une fois que l'expiration a été fixée.

Définissez une durée. Temps en secondes à attendre avant de supprimer le cluster.

La plage est comprise entre 10 minutes minimum et 14 jours maximum, avec une granularité de 1 seconde.

Actuellement disponible en ligne de commande et via l'API REST, mais pas via la console.

<https://cloud.google.com/dataproc/docs/concepts/configuring-clusters/scheduled-deletion/?hl=fr>

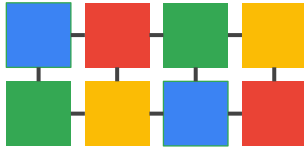
RFC 3339 UTC format « Zulu », précis à la nanoseconde près

Spécifiez une durée en secondes avec un maximum de neuf chiffres fractionnaires, terminés par un « s ». Exemple : "3.5s".

## Avec les clusters éphémères, vous ne payez que ce que vous utilisez



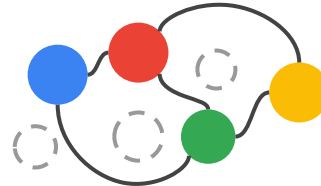
Clusters persistants



Les ressources sont tout le temps actives. Vous payez constamment l'intégralité des clusters.



Clusters éphémères



Les ressources dont vous avez besoin sont actives uniquement quand elles sont utilisées. Vous payez uniquement ce que vous utilisez.



Le plus grand changement dans votre approche entre l'exécution d'un flux de travail Hadoop sur site et l'exécution du même flux de travail sur GCP est l'abandon des clusters monolithiques et persistants au profit de clusters spécialisés et éphémères. Vous faites tourner un cluster lorsque vous devez exécuter une tâche et vous le supprimez lorsque la tâche est terminée. Les ressources nécessaires à vos tâches ne sont actives que lorsqu'elles sont utilisées, vous ne payez donc que pour ce que vous utilisez. Cette approche vous permet d'adapter la configuration des clusters à chaque tâche. Comme vous n'entretenez et ne configurez pas un cluster persistant, vous réduisez les coûts d'utilisation des ressources et d'administration du cluster. Cette section décrit comment déplacer votre infrastructure Hadoop existante vers un modèle éphémère.

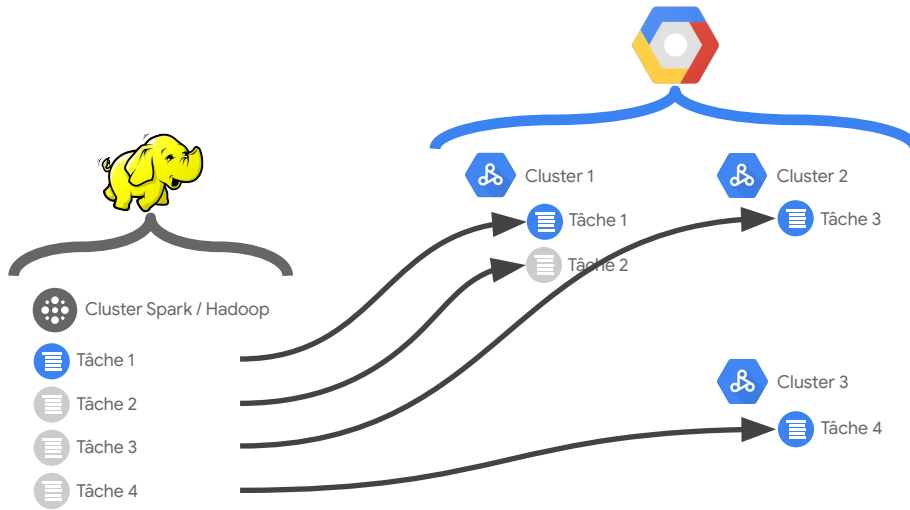
Pour tirer le meilleur parti de Cloud Dataproc, les clients doivent passer à un modèle « éphémère » consistant à n'utiliser les clusters que lorsqu'ils en ont besoin. Cela peut être effrayant car un cluster persistant est confortable. Cependant, avec la persistance des données GCS et le démarrage rapide de Cloud Dataproc, un cluster persistant est un gaspillage de ressources. Si un cluster persistant est nécessaire, créez-le de petite taille. Les clusters sont redimensionnables à tout moment.

Le modèle éphémère est la voie recommandée, mais il nécessite que le stockage soit découplé du calcul.

Dunnhumby a donné une excellente conférence sur le passage aux clusters

éphémères à l'adresse <https://youtu.be/2ksD7udWFys>

## Séparer les clusters des tâches

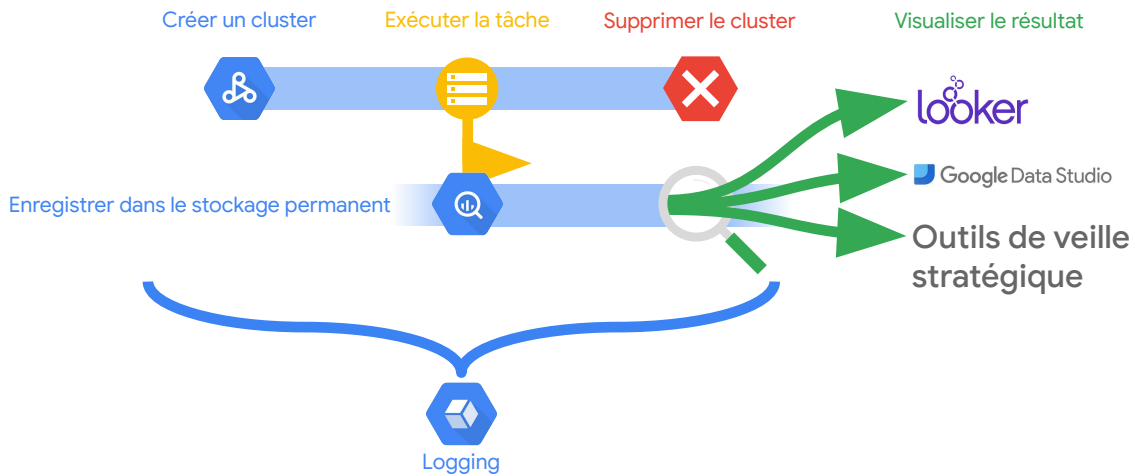


 Google Cloud

Formes de tâches séparées -> clusters séparés. Décomposez encore plus avec des clusters par tâches.

Isolez les environnements de développement, de « staging » et de production en utilisant des clusters séparés. Lisez à partir de la même source de données sous-jacente sur GCS. Ajoutez les ACL appropriées aux comptes de service pour protéger les données.

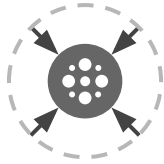
# Utiliser des clusters éphémères pour la durée de vie d'une tâche



L'intérêt des clusters éphémères est de ne les utiliser que pour la durée de vie des tâches. Lorsqu'il est temps d'exécuter une tâche, suivez ce processus :

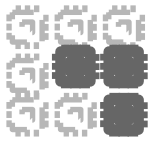
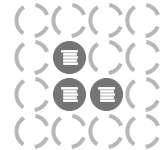
1. Créez un cluster correctement configuré.
2. Exécutez votre travail, en envoyant la sortie vers Cloud Storage ou un autre emplacement persistant.
3. Supprimez le cluster.
4. Utilisez les résultats de votre tâche comme vous le souhaitez.
5. Consultez les journaux dans Stackdriver ou Cloud Storage.

## Points à retenir si vous avez besoin d'un cluster persistant



Créez le cluster le plus petit possible, à l'aide de **VM préemptives** en fonction du budget temps

Étendez votre travail sur un cluster persistant au plus petit nombre de tâches possible



Dimensionnez le cluster au nombre minimum de nœuds utilisables. Ajoutez des nœuds à la demande plus dynamiquement (**auto-scaling**).

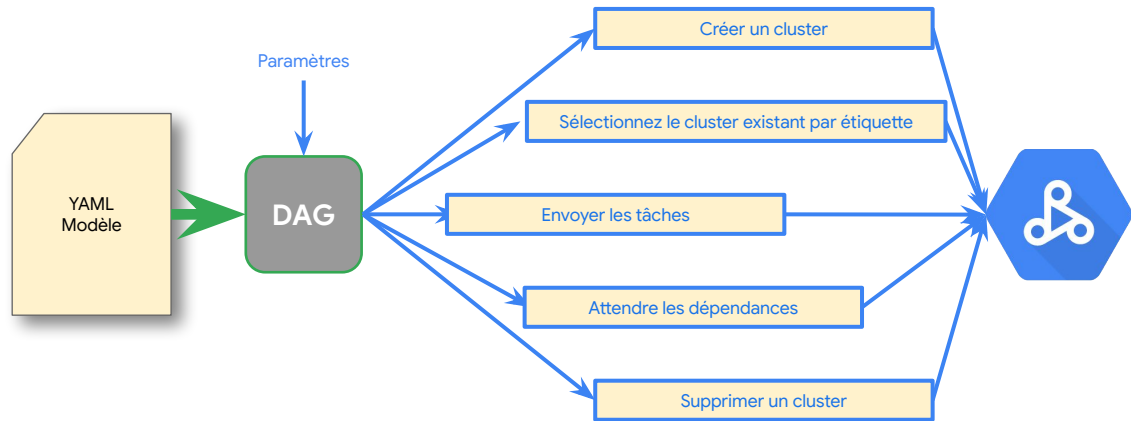


Si vous ne pouvez pas accomplir votre travail sans un cluster persistant, vous pouvez en créer un. Cette option peut être coûteuse et n'est pas recommandée s'il existe un moyen d'accomplir votre travail sur des clusters éphémères.

Vous pouvez minimiser le coût d'un cluster persistant en :

- En créant le cluster le plus petit possible.
- Étendez votre travail sur ce cluster au plus petit nombre de tâches possible.
- En augmentant le nombre de nœuds du cluster jusqu'au nombre minimal de nœuds utilisables, en ajoutant d'autres nœuds de manière dynamique pour répondre à la demande.

## Modèles de workflow de Cloud Dataproc



Le modèle de flux de travail de Cloud Dataproc est un fichier YAML qui est traité par un Directed Acyclique Graph (DAG). Il permet de créer un nouveau cluster, de sélectionner un cluster existant, de soumettre des tâches, de conserver des tâches à soumettre jusqu'à ce que les dépendances soient terminées et de supprimer un cluster lorsque la tâche est terminée.

Il est actuellement disponible via la commande `gcloud` et l'API REST, mais pas via la console.

Le modèle de flux de travail devient actif lorsqu'il est instancié dans le DAG. Le modèle peut être soumis plusieurs fois avec différentes valeurs de paramètres. Vous pouvez également écrire un modèle imbriqué dans la commande `gcloud`, et vous pouvez lister les flux de travail et les métadonnées de flux de travail pour aider à diagnostiquer les problèmes.

<https://cloud.google.com/dataproc/docs/concepts/workflows/overview/?hl=fr>

<https://cloud.google.com/dataproc/docs/concepts/labels/?hl=fr>

# Modèles de workflow de Cloud Dataproc

```
# the things we need pip-installed on the cluster
STARTUP_SCRIPT=gs://${BUCKET}/sparktobq/startup_script.sh
echo "pip install --upgrade --quiet google-compute-engine google-cloud-storage matplotlib" >
/tmp/startup_script.sh
gsutil cp /tmp/startup_script.sh $STARTUP_SCRIPT

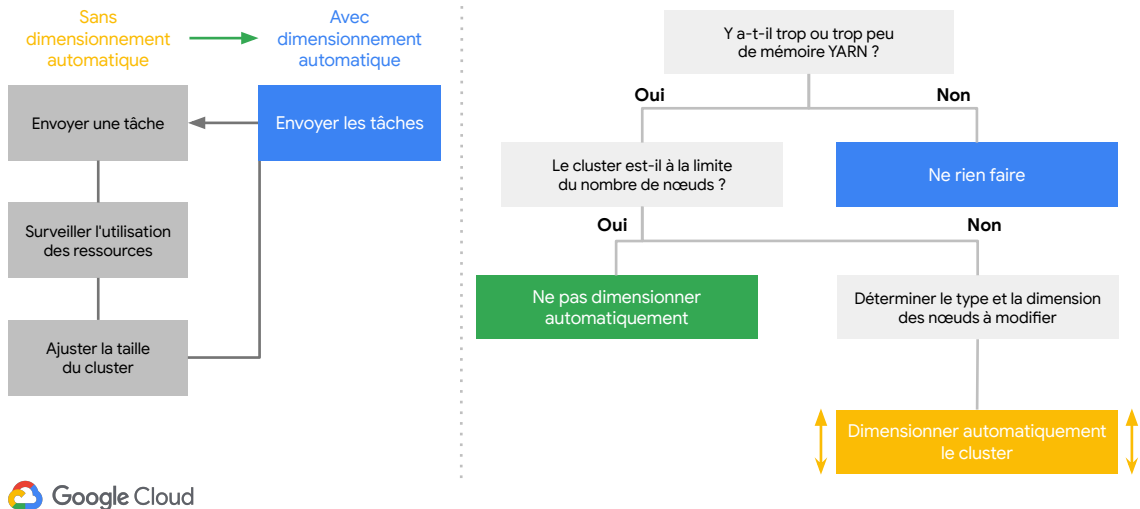
# create new cluster for job
gcloud dataproc workflow-templates set-managed-cluster $TEMPLATE \
  --master-machine-type $MACHINE_TYPE \
  --worker-machine-type $MACHINE_TYPE \
  --initialization-actions $STARTUP_SCRIPT \
  --num-workers 2 \
  --image-version 1.4 \
  --cluster-name $CLUSTER

# steps in job
gcloud dataproc workflow-templates add-job \
  pyspark gs://$BUCKET/spark_analysis.py \
  --step-id create-report \
  --workflow-template $TEMPLATE \
  -- --bucket=$BUCKET

# submit workflow template
gcloud beta dataproc workflow-templates instantiate $TEMPLATE
```



# Workflow de dimensionnement automatique de Cloud Dataproc



La mise à l'échelle automatique de Cloud Dataproc fournit des clusters qui se dimensionnent en fonction des besoins de l'entreprise. Ses principales caractéristiques sont les suivantes :

- Les tâches sont « déclenchées puis oubliées »
- Il n'est pas nécessaire d'intervenir manuellement lorsqu'un cluster est en surcapacité ou en sous-capacité
- Vous pouvez choisir entre les travailleurs standard et les travailleurs préemptibles, et
- Vous pouvez économiser des ressources (quota et coût) à tout moment

Les politiques d'Autoscaling permettent un contrôle fin. Cela repose sur la différence entre la mémoire en attente de YARN et la mémoire disponible. Si vous avez besoin de plus de mémoire, vous augmentez les dimensions. S'il y a un excès de mémoire, vous diminuez. Respectez les limites de la VM et les dimensions reposant sur le facteur d'échelle.

Les améliorations liées à l'autoscaling peuvent être résumées comme suit :

Des contrôles encore plus fins :

- Les politiques d'autoscaling peuvent être mises à jour ou supprimées à tout moment
- L'intervalle minimum de mise à l'échelle a été réduit de 10 min à 2 min
- Les politiques d'autoscaling peuvent être partagées entre plusieurs clusters

Plus facile à comprendre :

- Tableaux de bord YARN et HDFS dans la page des clusters
- L'historique des décisions d'autoscaling est disponible dans Stackdriver Logging

Stabilité de la tâche :

- Dimensionnez MapReduce et créez des tâches sans perdre de vue les progrès

# La fonctionnalité Autoscaling de Cloud Dataproc permet d'ajuster la capacité

Cluster avec beaucoup de tâches  
ou une seule grande tâche

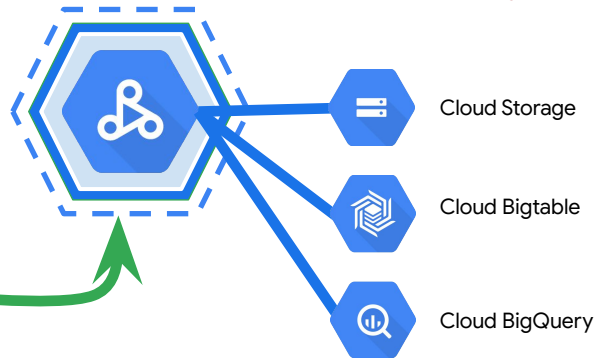
Pas pour les clusters inactifs ou mettre  
à zéro

L'autoscaling repose sur les  
métriques YARN Hadoop

Données externes

Pas pour les HDFS sur les clusters

Pas pour le streaming structuré Spark



La fonctionnalité Autoscaling de Cloud Dataproc offre une capacité flexible pour une utilisation plus efficace.

Elle base ses décisions en matière de scaling sur les métriques YARN Hadoop.

Il est conçu pour être utilisé uniquement avec des données persistantes hors cluster, et non avec des données HDFS ou HBase sur cluster.

Il fonctionne mieux avec un cluster qui traite beaucoup de tâches ou qui traite une seule grosse tâche.

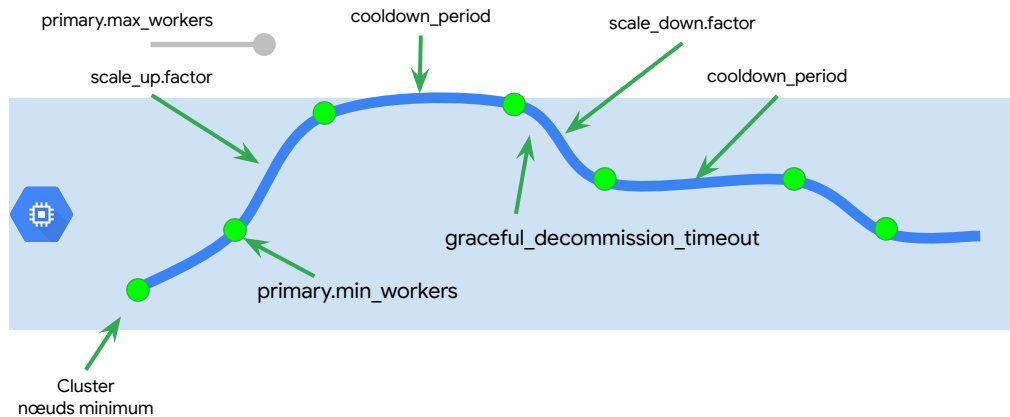
Il ne prend pas en charge le Spark Structured Streaming (un service de streaming construit sur Spark SQL).

Il n'est pas conçu pour être mis à l'échelle zéro. Il n'est donc pas le mieux adapté pour les clusters peu utilisés ou inactifs.

Dans ces cas-là, il est tout aussi rapide de mettre fin à un cluster inactif et d'en créer un nouveau lorsque c'est nécessaire.

Pour ce faire, vous pouvez envisager d'utiliser Cloud Dataproc Workflows ou Cloud Composer, et Cluster Scheduled Deletion.

# Fonctionnement du dimensionnement automatique de Cloud Dataproc



**Travailleurs initiaux** -- Le nombre de travailleurs initiaux est défini à partir de Nœuds de travailleurs > nœuds minimum. Le réglage de cette valeur garantit que le cluster atteindra sa capacité de base plus rapidement que si vous laissez l'autoscaling le gérer. En effet, l'autoscaling peut nécessiter plusieurs périodes de mise à l'échelle automatique.

Le nombre minimal de travailleurs principaux peut être le même que le nombre minimal de nœuds du cluster. Il existe un maximum qui limite le nombre de nœuds de travailleurs.

Le cluster est alors fortement sollicité. Et l'autoscaling détermine qu'il est temps de passer à l'échelle supérieure. Le facteur `scale_up.factor` détermine le nombre de nœuds à lancer. Il s'agit généralement d'un seul nœud. Mais si vous saviez qu'il y aurait beaucoup de demande simultanément, peut-être voudriez-vous passer à l'échelle plus rapidement.

Après l'action, il y a une période de pause pour laisser les choses se calmer avant que l'évaluation d'autoscaling ne se produise à nouveau. La période de pause réduit les chances que le cluster démarre et termine des nœuds simultanément.

Dans cet exemple, la capacité supplémentaire n'est pas nécessaire. Et il y a un délai de mise hors service gracieux pour donner aux tâches en cours une chance de se terminer avant que le nœud ne soit mis hors service. Notez qu'il y a un facteur de réduction d'échelle. Dans ce cas, il s'agit de réduire la capacité d'un nœud à la fois

pour une réduction plus souple.

Après l'action, il y a une autre période de pause.

Et une deuxième réduction d'échelle, qui se traduit par un retour au nombre minimum de travailleurs.

Un deuxième `min_workers` et `max_workers` contrôle l'échelle des travailleurs préemptibles.

Vous pouvez en savoir plus sur l'algorithme de mise à l'échelle ici :

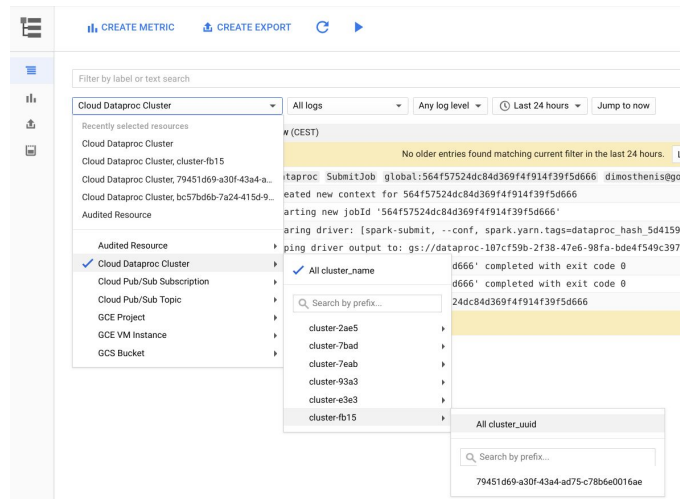
<https://cloud.google.com/dataproc/docs/concepts/configuring-clusters/autoscaling/?hl=fr>

`scale_up.factor` -- le nombre de nœuds à ajouter lors d'un événement de mise à l'échelle.

`scale_down.factor` -- le nombre de nœuds à supprimer lors d'un événement de réduction d'échelle.

`graceful_decommission_timeout` -- combien de temps il faut attendre la fin d'une tâche avant de fermer le nœud.

# Utiliser la journalisation et la surveillance des performances de Stackdriver



Dans GCP, vous pouvez utiliser Stackdriver Logging et Stackdriver Monitoring pour afficher et personnaliser les journaux, et pour surveiller les tâches et les ressources. La meilleure façon de trouver quelle erreur a causé l'échec d'une tâche Spark est d'examiner la sortie du pilote et les journaux générés par les fichiers exécutables Spark.

**Remarque :** Si vous soumettez la tâche Spark en vous connectant directement au nœud maître en utilisant SSH, il n'est pas possible d'obtenir la sortie du pilote.

Vous pouvez récupérer la sortie du programme du pilote en utilisant la console de la plateforme Google Cloud ou en utilisant une commande `gcloud`. La sortie est également stockée dans le bucket « Cloud Storage » du cluster Cloud Dataproc.

**Remarque :** Lorsque vous visualisez la sortie du programme du pilote dans la console GCP, la barre de progression est stockée sous la forme d'une longue ligne sans nouvelle ligne à la fin. Par conséquent, la première chose que le programme pilote imprime apparaîtra à la fin de cette ligne. Pour visualiser cette sortie plus facilement, vous pouvez cliquer sur Line Wrapping.

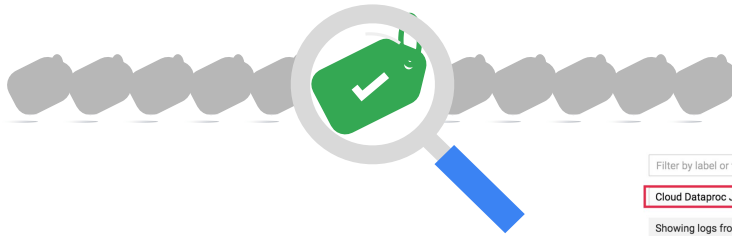
Tous les autres journaux sont situés dans différents fichiers à l'intérieur des machines du cluster. Il est possible de voir les journaux pour chaque conteneur à partir de l'interface Web de l'application Spark (ou du serveur d'historique après la fin du programme) dans l'onglet des fichiers exécutables. Vous devez naviguer dans chaque conteneur Spark pour visualiser chaque journal. Si vous écrivez des journaux ou les imprimez sur `stdout` ou `stderr` dans votre code d'application, les journaux sont enregistrés dans la redirection de `stdout` ou `stderr`.

Dans un cluster Cloud Dataproc, YARN est configuré pour collecter tous ces journaux par défaut, et ils sont disponibles dans Stackdriver Logging. La journalisation offre une vue consolidée et concise de tous les journaux afin que vous n'ayez pas besoin de passer du temps à naviguer parmi les journaux de conteneurs pour trouver des erreurs.

Cet écran affiche la page Logging dans la console GCP. Vous pouvez consulter tous les journaux de votre cluster Cloud Dataproc en sélectionnant le nom du cluster dans le menu de sélection. N'oubliez pas d'étendre la durée dans le sélecteur de plage de temps.

Vous pouvez obtenir les journaux d'une application Spark en les filtrant par son ID. Vous pouvez obtenir l'ID de l'application à partir de la sortie du pilote.

# Créer des étiquettes sur les clusters et les tâches pour retrouver plus rapidement les journaux



```
Filter by label or text search
Cloud Dataproc Job [dropdown] dataproc.job.driver
Showing logs from all time (PDT)
2019-04-03 09:34:16.478 PDT Pi is roughly 3.1417569
{
  insertId: "1e81240nizpl88ay9"
  labels: {}
  logName: "projects/google.com:hadoop-cloud-dev/log"
  receiverTimestamp: "2019-04-03T16:34:19.778423350Z"
  resource: {}
  textPayload: "Pi is roughly 3.1417569514175696"
  timestamp: "2019-04-03T16:34:16.478380936Z"
}
```



Pour trouver les journaux plus rapidement, vous pouvez créer et utiliser vos propres étiquettes pour chaque cluster ou pour chaque tâche de Cloud Dataproc. Par exemple, vous pouvez créer une étiquette avec la clé `env` et l'exploration des valeurs et l'utiliser pour votre travail d'exploration des données. Vous pouvez ensuite obtenir des journaux pour toutes les créations de tâches d'exploration en filtrant avec `label:env:exploration` dans Logging. Notez que ce filtre ne renvoie pas tous les journaux pour cette tâche, mais seulement les journaux de création de ressources.



## Régler le niveau de journal

Vous pouvez définir le niveau du journal du pilote en utilisant la commande gcloud suivante :

```
gcloud dataproc jobs submit hadoop --driver-log-levels
```

Vous définissez le niveau de journalisation pour le reste de l'application à partir du contexte Spark.

Par exemple :

```
spark.sparkContext.setLogLevel("DEBUG")
```



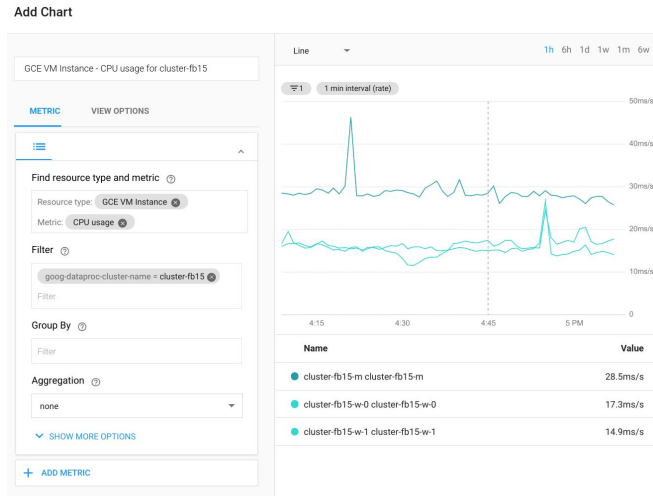
Vous pouvez définir le niveau du journal du pilote en utilisant la commande gcloud suivante :

```
gcloud dataproc jobs submit hadoop --driver-log-levels
```

Vous définissez le niveau de journalisation pour le reste de l'application à partir du contexte Spark. Par exemple :

```
spark.sparkContext.setLogLevel("DEBUG")
```

# Surveillez vos tâches



Stackdriver Monitoring peut surveiller le processeur, le disque, l'utilisation du réseau et les ressources YARN du cluster. Vous pouvez créer un tableau de bord personnalisé pour obtenir des graphiques à jour pour ces métriques et d'autres. Cloud Dataproc fonctionne sur Compute Engine. Si vous souhaitez visualiser l'utilisation du CPU, les entrées/sorties de disque ou les métriques de réseau dans un tableau, vous devez sélectionner une instance de la VM Engine Compute comme type de ressource, puis filtrer par le nom du cluster. Ce diagramme montre un exemple de la sortie.

Pour visualiser les métriques des requêtes, des jobs, des étapes ou des tâches Spark, connectez-vous à l'interface Web de l'application Spark.

# Programme

---

L'écosystème Hadoop

Exécuter Hadoop dans  
Cloud Dataproc

GCS au lieu de HDFS

Optimiser Dataproc

Atelier



---

## Exécuter des tâches Apache Spark dans Cloud Dataproc

### Objectifs

- Migrer les tâches Spark existantes vers Cloud Dataproc
- Modifiez les tâches Spark pour utiliser Cloud Storage au lieu de HDFS
- Optimiser les tâches Spark pour qu'elles fonctionnent sur des clusters spécifiques

# Objectifs de l'atelier

---

L'écosystème Hadoop

Exécuter Hadoop dans  
Cloud Dataproc

GCS au lieu de HDFS

Optimiser Dataproc