



Utilisation d'Apache Iceberg sur AWS

# AWS Conseils prescriptifs



# AWS Conseils prescriptifs: Utilisation d'Apache Iceberg sur AWS

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Les marques et la présentation commerciale d'Amazon ne peuvent être utilisées en relation avec un produit ou un service qui n'est pas d'Amazon, d'une manière susceptible de créer une confusion parmi les clients, ou d'une manière qui dénigre ou discrédite Amazon. Toutes les autres marques commerciales qui ne sont pas la propriété d'Amazon appartiennent à leurs propriétaires respectifs, qui peuvent ou non être affiliés ou connectés à Amazon, ou sponsorisés par Amazon.

# Table of Contents

|                                                                                     |    |
|-------------------------------------------------------------------------------------|----|
| Introduction .....                                                                  | 1  |
| Lacs de données modernes .....                                                      | 3  |
| Cas d'utilisation avancés dans les lacs de données modernes .....                   | 3  |
| Présentation d'Apache Iceberg .....                                                 | 4  |
| AWS support pour Apache Iceberg .....                                               | 5  |
| Commencer à utiliser les tables Iceberg dans Athena SQL .....                       | 9  |
| Création d'une table non partitionnée .....                                         | 9  |
| Création d'une table partitionnée .....                                             | 10 |
| Création d'une table et chargement de données avec une seule instruction CTAS ..... | 10 |
| Insertion, mise à jour et suppression de données .....                              | 11 |
| Interrogation des tables Iceberg .....                                              | 12 |
| Anatomie de la table Iceberg .....                                                  | 13 |
| Utilisation d'Iceberg dans Amazon EMR .....                                         | 15 |
| Compatibilité des versions et des fonctionnalités .....                             | 15 |
| Création d'un cluster Amazon EMR avec Iceberg .....                                 | 15 |
| Développement d'applications Iceberg dans Amazon EMR .....                          | 16 |
| Utilisation des blocs-notes Amazon EMR Studio .....                                 | 16 |
| Exécution de tâches Iceberg dans Amazon EMR .....                                   | 17 |
| Bonnes pratiques pour Amazon EMR .....                                              | 21 |
| Travailler avec Iceberg dans AWS Glue .....                                         | 24 |
| Utilisation de l'intégration native d'Iceberg .....                                 | 24 |
| Utilisation d'une version personnalisée d'Iceberg .....                             | 25 |
| Configurations Spark pour Iceberg dans AWS Glue .....                               | 26 |
| Bonnes pratiques en matière AWS Glue d'emploi .....                                 | 27 |
| Utilisation des tables Iceberg à l'aide de Spark .....                              | 29 |
| Création et écriture de tables Iceberg .....                                        | 29 |
| Utilisation de Spark SQL .....                                                      | 29 |
| Utilisation de l' DataFrames API .....                                              | 30 |
| Mise à jour des données dans les tables Iceberg .....                               | 31 |
| Insertion de données dans des tables Iceberg .....                                  | 32 |
| Supprimer des données dans les tables Iceberg .....                                 | 32 |
| Lecture de données .....                                                            | 33 |
| Utiliser le voyage dans le temps .....                                              | 33 |
| Utilisation de requêtes incrémentielles .....                                       | 34 |

|                                                                                  |    |
|----------------------------------------------------------------------------------|----|
| Accès aux métadonnées .....                                                      | 35 |
| Utilisation des tables Iceberg à l'aide de Trino .....                           | 37 |
| Configuration d'Amazon EMR sur EC2 .....                                         | 37 |
| Création de tables Iceberg .....                                                 | 38 |
| Lecture depuis les tables Iceberg .....                                          | 39 |
| Insertion de données dans des tables Iceberg .....                               | 39 |
| Supprimer des enregistrements des tables Iceberg .....                           | 40 |
| Interrogation des métadonnées d'une table Iceberg .....                          | 40 |
| Utiliser le voyage dans le temps .....                                           | 41 |
| Considérations relatives à l'utilisation d'Iceberg avec Trino .....              | 41 |
| Utilisation des tables Iceberg à l'aide de Firehose .....                        | 42 |
| Utilisation de tables Iceberg à l'aide d'Athena SQL .....                        | 43 |
| Compatibilité des versions et des fonctionnalités .....                          | 43 |
| Support pour les spécifications des tables Iceberg .....                         | 43 |
| Support des fonctionnalités Iceberg .....                                        | 43 |
| Utilisation des tables Iceberg .....                                             | 44 |
| Utilisation des tables Iceberg en utilisant Pylceberg .....                      | 46 |
| Prérequis .....                                                                  | 46 |
| Connexion au catalogue de données .....                                          | 46 |
| Lister et créer des bases de données .....                                       | 47 |
| Création et écriture de tables Iceberg .....                                     | 47 |
| Tables non partitionnées .....                                                   | 47 |
| Tables partitionnées .....                                                       | 48 |
| Lecture de données .....                                                         | 51 |
| Suppression des données .....                                                    | 51 |
| Accès aux métadonnées .....                                                      | 51 |
| Utiliser le voyage dans le temps .....                                           | 52 |
| Utilisation de la version 3 de la spécification du format de table Iceberg ..... | 53 |
| Principales fonctionnalités de la version 3 .....                                | 53 |
| Compatibilité des versions .....                                                 | 54 |
| Commencer à utiliser la version 3 .....                                          | 54 |
| Conditions préalables .....                                                      | 54 |
| Création de tables de version 3 .....                                            | 55 |
| Activation des vecteurs de suppression .....                                     | 56 |
| Utilisation du lignage des lignes pour le suivi des modifications .....          | 56 |
| Bonnes pratiques pour la version 3 .....                                         | 57 |

|                                                                                      |    |
|--------------------------------------------------------------------------------------|----|
| Quand utiliser la version 3 .....                                                    | 57 |
| Optimisation des performances d'écriture .....                                       | 57 |
| Optimisation des performances de lecture .....                                       | 58 |
| Stratégie de migration .....                                                         | 58 |
| Considérations de compatibilité .....                                                | 58 |
| Résolution des problèmes .....                                                       | 59 |
| Problèmes courants .....                                                             | 59 |
| Obtenir de l'aide .....                                                              | 60 |
| Tarification .....                                                                   | 60 |
| Disponibilité .....                                                                  | 60 |
| Ressources supplémentaires .....                                                     | 60 |
| Migration de tables existantes vers Iceberg .....                                    | 61 |
| In-place migration .....                                                             | 61 |
| Option 1 : procédure de capture instantanée .....                                    | 63 |
| Option 2 : procédure de migration .....                                              | 65 |
| Réplication de la procédure de migration des tables dans AWS Glue Data Catalog ..... | 69 |
| Synchronisation des tables Iceberg après une migration sur place .....               | 70 |
| Choisir la bonne stratégie de migration sur place .....                              | 72 |
| Migration complète des données .....                                                 | 75 |
| Choix d'une stratégie de migration .....                                             | 75 |
| Résumé des options de migration .....                                                | 77 |
| Meilleures pratiques pour optimiser les charges de travail d'Iceberg .....           | 83 |
| Bonnes pratiques d'ordre général .....                                               | 83 |
| Optimisation des performances de lecture .....                                       | 84 |
| Partitioning .....                                                                   | 85 |
| Réglage de la taille des fichiers .....                                              | 87 |
| Optimisation des statistiques des colonnes .....                                     | 89 |
| Choisissez la bonne stratégie de mise à jour .....                                   | 90 |
| Utiliser la compression ZSTD .....                                                   | 90 |
| Définissez l'ordre de tri .....                                                      | 90 |
| Optimisation des performances d'écriture .....                                       | 93 |
| Définissez le mode de distribution des tables .....                                  | 93 |
| Choisissez la bonne stratégie de mise à jour .....                                   | 93 |
| Choisissez le bon format de fichier .....                                            | 94 |
| Optimisation du stockage .....                                                       | 95 |
| Activer S3 Intelligent-Tiering .....                                                 | 95 |

|                                                                                             |      |
|---------------------------------------------------------------------------------------------|------|
| Archiver ou supprimer des instantanés historiques .....                                     | 96   |
| Supprimer les fichiers orphelins .....                                                      | 99   |
| Maintien des tables en utilisant le compactage .....                                        | 100  |
| Compactage des icebergs .....                                                               | 100  |
| Réglage du comportement de compactage .....                                                 | 102  |
| Exécution du compactage avec Spark sur Amazon EMR ou AWS Glue .....                         | 103  |
| Exécution du compactage avec Amazon Athena .....                                            | 104  |
| Recommandations pour le compactage en cours .....                                           | 104  |
| Utilisation des charges de travail Iceberg dans Amazon S3 .....                             | 106  |
| Empêcher le partitionnement à chaud (erreurs HTTP 503) .....                                | 106  |
| Utilisez les opérations de maintenance d'Iceberg pour libérer les données inutilisées ..... | 107  |
| Répliquez les données sur Régions AWS .....                                                 | 107  |
| Surveillance des charges de travail d'Iceberg .....                                         | 109  |
| Table-level surveillance .....                                                              | 109  |
| Database-level surveillance .....                                                           | 111  |
| Maintenance préventive .....                                                                | 113  |
| Gouvernance et contrôle d'accès .....                                                       | 114  |
| Architectures de référence .....                                                            | 115  |
| Ingestion nocturne de lots .....                                                            | 115  |
| Lac de données combinant ingestion par lots et ingestion en temps quasi réel .....          | 116  |
| Ressources .....                                                                            | 117  |
| Collaborateurs .....                                                                        | 118  |
| Historique du document .....                                                                | 120  |
| .....                                                                                       | cxxi |

# Utilisation d'Apache Iceberg sur AWS

Amazon Web Services ([contributeurs](#))

Novembre 2025 ([historique du document](#))

Apache Iceberg est un format de table open source qui simplifie la gestion des tables tout en améliorant les performances. AWS les services d'analyse tels qu'Amazon EMR AWS Glue, Amazon Athena et Amazon Redshift incluent un support natif pour Iceberg, ce qui vous permet de créer facilement des lacs de données transactionnels sur Amazon Simple Storage Service (Amazon S3) sur AWS.

En outre, la prochaine génération d'Amazon SageMaker repose sur une [architecture Lakehouse ouverte](#) qui unifie l'accès aux données entre AWS les lacs de données, les entrepôts de données et les sources tierces et fédérées. Le Lakehouse est entièrement compatible avec Iceberg et vous donne la flexibilité d'accéder aux données sur place et de les interroger à l'aide de l'API REST d'Iceberg.

Ce guide technique fournit des conseils pour démarrer avec Iceberg sur différents points Services AWS, et inclut les meilleures pratiques et recommandations pour faire fonctionner Iceberg AWS à grande échelle tout en optimisant les coûts et les performances.

Que vous débutiez avec Iceberg ou que vous soyez un utilisateur expérimenté cherchant à optimiser vos charges de travail Iceberg existantes AWS, ce guide fournit des informations précieuses pour chaque étape de votre projet.

Dans ce guide :

- [Lacs de données modernes](#)
- [Commencer à utiliser les tables Iceberg dans Athena SQL](#)
- [Utilisation d'Iceberg dans Amazon EMR](#)
- [Travailler avec Iceberg dans AWS Glue](#)
- [Utilisation des tables Iceberg à l'aide de Spark](#)
- [Utilisation des tables Iceberg à l'aide de Trino](#)
- [Utilisation des tables Iceberg à l'aide d'Amazon Data Firehose](#)
- [Utilisation de tables Iceberg à l'aide d'Athena SQL](#)
- [Utilisation des tables Iceberg en utilisant Pylceberg](#)

- [Utilisation de la version 3 de la spécification du format de table Iceberg](#)
- [Migration de tables existantes vers Iceberg](#)
- [Meilleures pratiques pour optimiser les charges de travail d'Iceberg](#)
- [Surveillance des charges de travail d'Iceberg](#)
- [Gouvernance et contrôle d'accès](#)
- [Architectures de référence](#)
- [Ressources](#)
- [Collaborateurs](#)

# Lacs de données modernes

## Cas d'utilisation avancés dans les lacs de données modernes

L'évolution du stockage des données est passée des bases de données aux entrepôts de données et aux lacs de données, où chaque technologie répond à des exigences commerciales et de données uniques. Les bases de données traditionnelles excellaient dans la gestion des données structurées et des charges de travail transactionnelles, mais elles se heurtaient à des problèmes de performance à mesure que les volumes de données augmentaient. Les entrepôts de données sont apparus pour résoudre les problèmes de performance et d'évolutivité, mais comme les bases de données, ils reposaient sur des formats propriétaires au sein de systèmes intégrés verticalement.

Les lacs de données constituent l'une des meilleures options pour stocker des données en termes de coût, d'évolutivité et de flexibilité. Vous pouvez utiliser un lac de données pour conserver de gros volumes de données structurées et non structurées à moindre coût, et utiliser ces données pour différents types de charges de travail analytiques, qu'il s'agisse de rapports de business intelligence, de traitement de mégadonnées, d'analyses en temps réel, d'apprentissage automatique ou d'intelligence artificielle générative (IA), afin de prendre de meilleures décisions.

Malgré ces avantages, les lacs de données n'ont pas été initialement conçus avec des fonctionnalités similaires à celles des bases de données. Un lac de données ne prend pas en charge la sémantique de traitement ACID (atomicité, cohérence, isolation et durabilité), dont vous pourriez avoir besoin pour optimiser et gérer efficacement vos données à grande échelle auprès de centaines ou de milliers d'utilisateurs en utilisant de nombreuses technologies différentes. Les lacs de données ne fournissent pas de support natif pour les fonctionnalités suivantes :

- Effectuer des mises à jour et des suppressions efficaces au niveau des enregistrements à mesure que les données changent dans votre entreprise
- Gestion des performances des requêtes lorsque les tables se transforment en millions de fichiers et en centaines de milliers de partitions
- Garantir la cohérence des données entre plusieurs rédacteurs et lecteurs simultanés
- Empêcher la corruption des données lorsque les opérations d'écriture échouent en cours d'opération
- Évolution des schémas de table au fil du temps sans réécriture (partielle) des ensembles de données

Ces défis sont devenus particulièrement courants dans des cas d'utilisation tels que la gestion de la capture des données modifiées (CDC) ou les cas d'utilisation relatifs à la confidentialité, à la suppression de données et à l'ingestion de données en streaming, ce qui peut entraîner des tables sous-optimales.

Les lacs de données qui utilisent les Hive-format tables traditionnelles ne prennent en charge les opérations d'écriture que pour des fichiers entiers. Cela rend les mises à jour et les suppressions difficiles à mettre en œuvre, longues et coûteuses. En outre, les contrôles de simultanéité et les garanties proposés dans les ACID-compliant systèmes sont nécessaires pour garantir l'intégrité et la cohérence des données.

Ces défis placent les utilisateurs face à un dilemme : choisir entre une plate-forme entièrement intégrée mais propriétaire, ou opter pour un lac de données indépendant du fournisseur mais gourmand en ressources, construit par eux-mêmes et nécessitant une maintenance et une migration constantes pour tirer parti de sa valeur potentielle.

[Pour aider à relever ces défis, Iceberg fournit des fonctionnalités supplémentaires de type base de données qui simplifient l'optimisation et les frais de gestion des lacs de données, tout en prenant en charge le stockage sur des systèmes rentables tels qu'Amazon S3.](#)

## Présentation d'Apache Iceberg

Apache Iceberg est un format de table open source qui fournit des fonctionnalités dans les tables de lacs de données qui n'étaient auparavant disponibles que dans les bases de données ou les entrepôts de données. Il est conçu dans un souci d'évolutivité et de performance, et convient parfaitement à la gestion de tables de plus de centaines de gigaoctets. Voici quelques-unes des principales caractéristiques des tables Iceberg :

- Supprimez, mettez à jour et fusionnez. Iceberg prend en charge les commandes SQL standard pour l'entreposage de données à utiliser avec les tables de lacs de données.
- Planification rapide des scans et filtrage avancé. Iceberg stocke des métadonnées telles que les statistiques au niveau des partitions et des colonnes qui peuvent être utilisées par les moteurs pour accélérer la planification et l'exécution des requêtes.
- Évolution complète du schéma. Iceberg permet d'ajouter, de supprimer, de mettre à jour ou de renommer des colonnes sans effets secondaires.
- Évolution de la partition. Vous pouvez mettre à jour la disposition de partition d'une table à mesure que le volume de données ou les modèles de requête changent. Iceberg prend en charge la

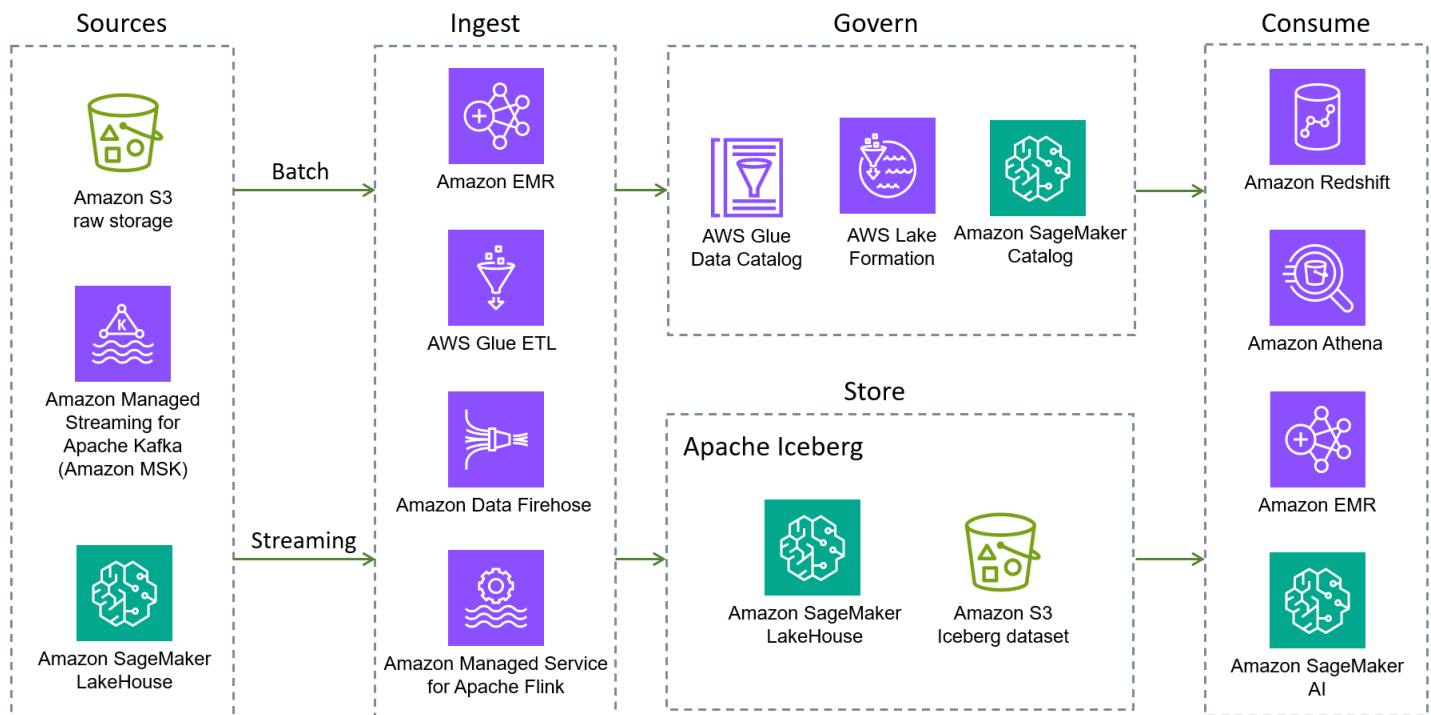
modification des colonnes sur lesquelles une table est partitionnée, l'ajout de colonnes ou la suppression de colonnes dans des partitions composites.

- **Partitionnement masqué.** Cette fonctionnalité empêche la lecture automatique de partitions inutiles. Les utilisateurs n'ont donc plus besoin de comprendre les détails du partitionnement de la table ou d'ajouter des filtres supplémentaires à leurs requêtes.
- **Annulation de la version.** Les utilisateurs peuvent rapidement corriger les problèmes en revenant à un état antérieur à la transaction.
- **Voyage dans le temps.** Les utilisateurs peuvent interroger une version précédente spécifique d'une table.
- **Isolation sérialisable.** Les modifications apportées aux tables sont atomiques, de sorte que les lecteurs ne voient jamais de modifications partielles ou non validées.
- **Rédacteurs concurrents.** Iceberg utilise une simultanéité optimiste pour permettre à plusieurs transactions de réussir. En cas de conflit, l'un des rédacteurs doit réessayer la transaction.
- **Formats de fichiers ouverts.** Iceberg prend en charge plusieurs formats de fichiers open source, notamment [Apache Parquet](#), [Apache Avro](#) et [Apache ORC](#).

En résumé, les lacs de données qui utilisent le format Iceberg bénéficient de la cohérence transactionnelle, de la rapidité, de l'échelle et de l'évolution des schémas. Pour plus d'informations sur ces fonctionnalités et sur d'autres fonctionnalités d'Iceberg, consultez la documentation d'[Apache Iceberg](#).

## AWS support pour Apache Iceberg

[Apache Iceberg est pris en charge par Services AWS Amazon EMR, Amazon Athena, Amazon AWS Glue <https://aws.amazon.com/glue/Redshift> et Amazon. SageMaker](#) Le schéma suivant illustre l'architecture de référence simplifiée d'un lac de données basé sur Iceberg.



Les éléments suivants Services AWS fournissent des intégrations natives d'Iceberg. D'autres Services AWS peuvent interagir avec Iceberg, soit indirectement, soit en empaquetant les bibliothèques Iceberg.

- [Amazon S3](#) est le meilleur endroit pour créer des lacs de données en raison de sa durabilité, de sa disponibilité, de son évolutivité, de sa sécurité, de sa conformité et de ses fonctionnalités d'audit. Iceberg a été conçu et construit pour interagir de manière fluide avec Amazon S3 et prend en charge de nombreuses fonctionnalités d'Amazon S3 répertoriées dans la documentation d'[Iceberg](#). En outre, [Amazon S3 Tables](#) fournit le premier magasin d'objets dans le cloud avec prise en charge intégrée d'Iceberg et rationalise le stockage des données tabulaires à grande échelle. Grâce à la prise en charge d'Iceberg par S3 Tables, vous pouvez facilement interroger vos données tabulaires à l'aide de moteurs de requêtes populaires AWS et tiers.
- [La prochaine génération SageMaker](#) repose sur une architecture Lakehouse ouverte qui unifie l'accès aux données entre les lacs de données Amazon S3, les entrepôts de données Amazon Redshift et les sources de données tierces et fédérées. Ces fonctionnalités vous aident à créer de puissantes analyses et AI/ML applications à partir d'une seule copie des données. Le lakehouse est entièrement compatible avec Iceberg. Vous avez donc la possibilité d'accéder aux données sur place et de les interroger en utilisant l'API REST d'Iceberg.
- [Amazon EMR](#) est une solution de mégadonnées destinée au traitement de données à l'échelle du pétaoctet, à l'analyse interactive et à l'apprentissage automatique en utilisant des frameworks open

source tels qu'Apache Spark, Flink, Trino et Hive. Amazon EMR peut s'exécuter sur des clusters Amazon Elastic Compute Cloud (Amazon EC2) personnalisés, Amazon Elastic Kubernetes Service (Amazon EKS) ou Amazon EMR Serverless. AWS Outposts

- [Amazon Athena](#) est un service d'analyse interactif sans serveur basé sur des frameworks open source. Il prend en charge les formats de table ouverte et de fichier et fournit un moyen simplifié et flexible d'analyser des pétaoctets de données là où elles se trouvent. Athena fournit un support natif pour les requêtes de lecture, de voyage dans le temps, d'écriture et DDL pour Iceberg et utilise le AWS Glue Data Catalog métastore for the Iceberg.
- [Amazon Redshift](#) est un entrepôt de données cloud de plusieurs pétaoctets qui prend en charge les options de déploiement basées sur des clusters et sans serveur. Amazon Redshift Spectrum peut interroger les tables externes enregistrées auprès AWS Glue Data Catalog d'Amazon S3 et stockées sur celui-ci. Redshift Spectrum prend également en charge le format de stockage Iceberg.
- [AWS Glue](#) est un service d'intégration de données sans serveur qui facilite la découverte, la préparation, le déplacement et l'intégration de données provenant de sources multiples à des fins d'analyse, d'apprentissage automatique (ML) et de développement d'applications. Il est totalement intégré à Iceberg. Plus précisément, vous pouvez effectuer des opérations de lecture et d'écriture sur des tables Iceberg à l'aide de AWS Glue tâches, gérer des tables via le [AWS Glue Data Catalog](#) (compatible avec le métastore Hive), découvrir et enregistrer des tables automatiquement à l'aide de robots d' AWS Glue exploration et évaluer la qualité des données dans les tables Iceberg grâce à la fonctionnalité Data Quality. AWS Glue AWS Glue Data Catalog Il prend également en charge la collecte de statistiques sur les colonnes, le calcul et la mise à jour du nombre de valeurs distinctes (NDV) pour chaque colonne dans les tables Iceberg, ainsi que les optimisations automatiques des tables (compactage, conservation des instantanés, suppression de fichiers orphelins). AWS Glue prend également en charge les intégrations zéro ETL à partir d'une liste d' Services AWS applications tierces dans les tables Iceberg.
- [Amazon Data Firehose](#) est un service entièrement géré qui fournit des données de streaming en temps réel à des destinations telles qu'Amazon S3, Amazon Redshift, Amazon Service, OpenSearch Amazon Serverless, OpenSearch Splunk, Apache Iceberg, ainsi qu'à tout point de terminaison HTTP ou HTTP personnalisé appartenant à des fournisseurs de services tiers pris en charge, notamment Datadog, Dynatrace LogicMonitor, MongoDB, New Relic, Coralogix et Elastic. Avec Firehose, vous n'avez pas besoin d'écrire d'applications ou de gérer des ressources. Vous configurez vos producteurs de données pour envoyer les données à Firehose, qui remet automatiquement les données à la destination que vous avez spécifiée. Vous pouvez également configurer Firehose pour transformer vos données avant de les remettre.

- [Amazon Managed Service pour Apache Flink](#) est un service Amazon entièrement géré qui vous permet d'utiliser une application Apache Flink pour traiter des données de streaming. Il prend en charge la lecture et l'écriture dans les tables Iceberg, et permet le traitement et l'analyse des données en temps réel.
- [Amazon SageMaker AI](#) prend en charge le stockage d'ensembles de fonctionnalités dans Amazon SageMaker AI Feature Store en utilisant le format Iceberg.
- [AWS Lake Formation](#) fournit des autorisations de contrôle d'accès grossières et détaillées pour accéder aux données, notamment aux tables Iceberg consommées par Athena ou Amazon Redshift. Pour en savoir plus sur la prise en charge des autorisations pour les tables Iceberg, consultez la [documentation de Lake Formation](#).

AWS propose une large gamme de services qui soutiennent Iceberg, mais la couverture de tous ces services dépasse le cadre de ce guide. Les sections suivantes traitent de Spark (streaming par lots et structuré) sur Amazon EMR ainsi AWS Glue que d'Athena SQL. La [section suivante](#) fournit un aperçu rapide du support d'Iceberg dans Athena SQL.

# Commencer à utiliser les tables Iceberg dans Amazon Athena SQL

Amazon Athena fournit un support intégré pour Iceberg. Vous pouvez utiliser Iceberg sans étapes ni configuration supplémentaires, à l'exception de la configuration des prérequis de service détaillés dans la section [Getting started](#) de la documentation d'Athena. Cette section fournit une brève introduction à la création de tables dans Athena. Pour plus d'informations, consultez la section [Utilisation des tables Iceberg à l'aide d'Athena](#) SQL plus loin dans ce guide.

Vous pouvez créer des tables Iceberg à l'aide AWS de différents moteurs. Ces tables fonctionnent parfaitement d'un bout à l'autre Services AWS. Pour créer vos premières tables Iceberg avec Athena SQL, vous pouvez utiliser le code standard suivant.

```
CREATE TABLE <table_name> (  
    col_1 string,  
    col_2 string,  
    col_3 bigint,  
    col_ts timestamp)  
PARTITIONED BY (col_1, <<<partition_transform>>>(col_ts))  
LOCATION 's3://<bucket>/<folder>/<table_name>/'  
TBLPROPERTIES (  
    'table_type' = 'ICEBERG'  
)
```

Les sections suivantes fournissent des exemples de création de tables Iceberg partitionnées et non partitionnées dans Athena. Pour plus d'informations, consultez la syntaxe Iceberg détaillée dans la documentation d'[Athena](#).

## Création d'une table non partitionnée

L'exemple d'instruction suivant personnalise le code SQL standard pour créer une table Iceberg non partitionnée dans Athena. Vous pouvez ajouter cette instruction à l'éditeur de requêtes de la [console Athena](#) pour créer la table.

```
CREATE TABLE athena_iceberg_table (  
    color string,  
    date string,  
    name string,
```

```
price bigint,  
product string,  
ts timestamp)  
LOCATION 's3://DOC_EXAMPLE_BUCKET/ice_warehouse/iceberg_db/athena_iceberg_table/'  
TBLPROPERTIES (  
    'table_type' = 'ICEBERG'  
)
```

Pour obtenir des instructions détaillées sur l'utilisation de l'éditeur de requêtes, voir [Getting started](#) dans la documentation d'Athena.

## Création d'une table partitionnée

L'instruction suivante crée une table partitionnée basée sur la date en utilisant le concept de partitionnement [caché](#) d'Iceberg. Il utilise la `day()` transformation pour dériver des partitions quotidiennes, en utilisant le `dd-mm-yyyy` format, à partir d'une colonne d'horodatage. Iceberg ne stocke pas cette valeur sous forme de nouvelle colonne dans le jeu de données. Au lieu de cela, la valeur est dérivée à la volée lorsque vous écrivez ou interrogez des données.

```
CREATE TABLE athena_iceberg_table_partitioned (  
    color string,  
    date string,  
    name string,  
    price bigint,  
    product string,  
    ts timestamp)  
PARTITIONED BY (day(ts))  
LOCATION 's3://DOC_EXAMPLE_BUCKET/ice_warehouse/iceberg_db/athena_iceberg_table/'  
TBLPROPERTIES (  
    'table_type' = 'ICEBERG'  
)
```

## Création d'une table et chargement de données avec une seule instruction CTAS

Dans les exemples partitionnés et non partitionnés des sections précédentes, les tables Iceberg sont créées sous forme de tables vides. Vous pouvez charger des données dans les tables à l'aide de l'`MERGE` instruction `INSERT` or. Vous pouvez également utiliser une `CREATE TABLE AS SELECT` (CTAS) instruction pour créer et charger des données dans une table Iceberg en une seule étape.

Le CTAS est le meilleur moyen dans Athena de créer une table et de charger des données dans une seule instruction. L'exemple suivant montre comment utiliser le CTAS pour créer une table Iceberg (`iceberg_ctas_table`) à partir d'une Hive/Parquet table existante (`hive_table`) dans Athena.

```
CREATE TABLE iceberg_ctas_table WITH (  
    table_type = 'ICEBERG',  
    is_external = false,  
    location = 's3://DOC_EXAMPLE_BUCKET/ice_warehouse/iceberg_db/iceberg_ctas_table/'  
) AS  
SELECT * FROM "iceberg_db"."hive_table" limit 20  
---  
SELECT * FROM "iceberg_db"."iceberg_ctas_table" limit 20
```

Pour en savoir plus sur le CTAS, consultez la documentation d'[Athena](#) CTAS.

## Insertion, mise à jour et suppression de données

Athena prend en charge différentes manières d'écrire des données dans une table Iceberg en utilisant les `INSERT INTO` instructions, `UPDATEMERGE INTO`, et `M. DELETE FROM`

### Note

Athena SQL ne prend actuellement pas en charge l'approche de copie sur écriture. `UPDATEMERGE INTO`, et les `DELETE FROM` opérations utilisent toujours l'approche de fusion lors de la lecture avec des suppressions positionnelles, quelles que soient les propriétés de table spécifiées. Si vous configurez des propriétés de table telles que `write.update.modewrite.merge.mode`, et si vous utilisez la `write.delete.mode` fonction `copy-on-write`, vos requêtes n'échoueront pas, mais Athena les ignorera et continuera à utiliser `merge-on-read`.

L'instruction suivante permet `INSERT INTO` d'ajouter des données à une table Iceberg :

```
INSERT INTO "iceberg_db"."ice_table" VALUES (  
    'red', '222022-07-19T03:47:29', 'PersonNew', 178, 'Tuna', now()  
)  
  
SELECT * FROM "iceberg_db"."ice_table"  
where color = 'red' limit 10;
```

## Exemple de sortie :

| Results (1)                              |       |                       |           |       |         |                                |  | Copy    | Download results |
|------------------------------------------|-------|-----------------------|-----------|-------|---------|--------------------------------|--|---------|------------------|
| <input type="text" value="Search rows"/> |       |                       |           |       |         |                                |  | < 1 > ⚙ |                  |
| #                                        | color | date                  | name      | price | product | ts                             |  |         |                  |
| 1                                        | red   | 222022-07-19T03:47:29 | PersonNew | 178   | Tuna    | 2023-10-11 11:35:01.298000 UTC |  |         |                  |

Pour plus d'informations, consultez la documentation d'[Athena](#).

## Interrogation des tables Iceberg

Vous pouvez exécuter des requêtes SQL régulières sur vos tables Iceberg à l'aide d'Athena SQL, comme illustré dans l'exemple précédent.

Outre les requêtes habituelles, Athena prend également en charge les requêtes de voyage dans le temps pour les tables Iceberg. Comme indiqué précédemment, vous pouvez modifier des enregistrements existants par le biais de mises à jour ou de suppressions dans une table Iceberg. Il est donc pratique d'utiliser des requêtes de voyage dans le temps pour consulter les anciennes versions de votre table en fonction d'un horodatage ou d'un identifiant d'instantané.

Par exemple, l'instruction suivante met à jour une valeur de couleur pour Person5, puis affiche une valeur antérieure datant du 4 janvier 2023 :

```
UPDATE ice_table SET color='new_color' WHERE name='Person5'

SELECT * FROM "iceberg_db"."ice_table" FOR TIMESTAMP AS OF TIMESTAMP '2023-01-04
12:00:00 UTC'
```

## Exemple de sortie :

| Results (15)                             |           |                       |         |       |          |                                |  | Copy    | Download results |
|------------------------------------------|-----------|-----------------------|---------|-------|----------|--------------------------------|--|---------|------------------|
| <input type="text" value="Search rows"/> |           |                       |         |       |          |                                |  | < 1 > ⚙ |                  |
| #                                        | color     | date                  | name    | price | product  | ts                             |  |         |                  |
| 1                                        | cyan      | 222022-07-19T03:47:29 | Person5 | 353   | Keyboard | 2023-01-03 10:15:52.268000 UTC |  |         |                  |
| 2                                        | lime      | 222022-07-19T03:47:29 | Person1 | 833   | Towels   | 2023-01-03 10:15:52.268000 UTC |  |         |                  |
| 3                                        | turquoise | 222022-07-19T03:47:29 | Person1 | 1319  | Shirt    | 2023-01-03 10:15:52.268000 UTC |  |         |                  |
| 4                                        | blue      | 222022-07-19T03:47:29 | Person3 | 163   | Sausages | 2023-01-03 10:15:52.268000 UTC |  |         |                  |

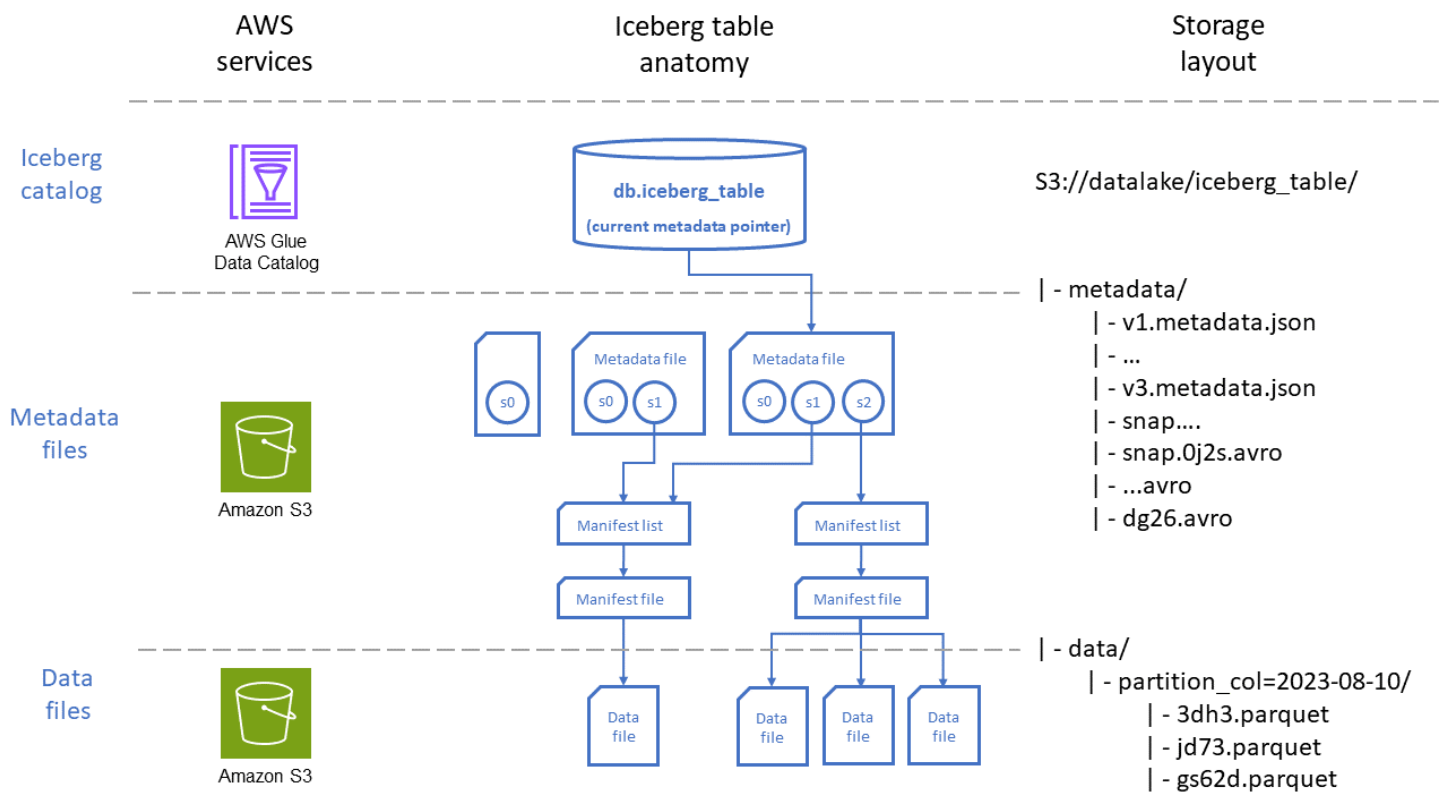
Pour la syntaxe et d'autres exemples de requêtes de voyage dans le temps, consultez la documentation d'[Athena](#).

# Anatomie de la table Iceberg

Maintenant que nous avons abordé les étapes de base de l'utilisation des tables Iceberg, examinons plus en détail les détails complexes et le design d'une table Iceberg.

Pour activer les fonctionnalités [décrites précédemment](#) dans ce guide, Iceberg est conçu avec des couches hiérarchiques de données et de fichiers de métadonnées. Ces couches gèrent les métadonnées de manière intelligente afin d'optimiser la planification et l'exécution des requêtes.

Le schéma suivant décrit l'organisation d'une table Iceberg selon deux perspectives : celle Services AWS utilisée pour stocker la table et le placement des fichiers dans Amazon S3.



Comme le montre le schéma, une table Iceberg se compose de trois couches principales :

- **Catalogue Iceberg :** AWS Glue Data Catalog s'intègre nativement à Iceberg et constitue, dans la plupart des cas d'utilisation, la meilleure option pour les charges de travail qui s'exécutent sur AWS. Les services qui interagissent avec les tables Iceberg (par exemple, Athena) utilisent le catalogue pour trouver la version instantanée actuelle de la table, soit pour lire soit pour écrire des données.
- **Couche de métadonnées :** les fichiers de métadonnées, à savoir les fichiers manifestes et les fichiers de listes de manifestes, gardent une trace des informations telles que le schéma des

tables, la stratégie de partition et l'emplacement des fichiers de données, ainsi que des statistiques au niveau des colonnes telles que les plages minimale et maximale pour les enregistrements stockés dans chaque fichier de données. Ces fichiers de métadonnées sont stockés dans Amazon S3 dans le chemin de la table.

- Les fichiers manifestes contiennent un enregistrement pour chaque fichier de données, y compris son emplacement, son format, sa taille, sa somme de contrôle et d'autres informations pertinentes.
- Les listes de manifestes fournissent un index des fichiers manifestes. À mesure que le nombre de fichiers manifestes augmente dans une table, la division de ces informations en sous-sections plus petites permet de réduire le nombre de fichiers manifestes devant être analysés par des requêtes.
- Les fichiers de métadonnées contiennent des informations sur l'ensemble de la table Iceberg, notamment les listes de manifestes, les schémas, les métadonnées de partition, les fichiers d'instantanés et les autres fichiers utilisés pour gérer les métadonnées de la table.
- Couche de données : cette couche contient les fichiers contenant les enregistrements de données sur lesquels les requêtes seront exécutées. Ces fichiers peuvent être stockés dans différents formats, notamment [Apache Parquet](#), [Apache Avro](#) et [Apache ORC](#).
- Les fichiers de données contiennent les enregistrements de données d'une table.
- Les fichiers de suppression encodent les opérations de suppression et de mise à jour au niveau des lignes dans une table Iceberg. Iceberg propose deux types de fichiers de suppression, comme décrit dans la documentation d'[Iceberg](#). Ces fichiers sont créés par des opérations en utilisant le mode fusion lors de la lecture.

# Utilisation d'Iceberg dans Amazon EMR

Amazon EMR fournit un traitement de données à l'échelle du pétaoctet, des analyses interactives et un apprentissage automatique dans le cloud en utilisant des frameworks open source tels qu'Apache Spark, Apache Hive, Flink et Trino.

## Note

Ce guide utilise Apache Spark à titre d'exemple.

Amazon EMR prend en charge plusieurs options de déploiement : Amazon EMR activé, EC2 Amazon EMR sur EKS, Amazon EMR sans serveur et Amazon EMR activé. AWS Outposts Pour choisir une option de déploiement adaptée à votre charge de travail, consultez la [FAQ Amazon EMR](#).

## Compatibilité des versions et des fonctionnalités

Amazon EMR version 6.5.0 et versions ultérieures prennent en charge Apache Iceberg de manière native. Pour obtenir la liste des versions d'Iceberg prises en charge pour chaque version d'Amazon EMR, [consultez l'historique des versions d'Iceberg](#) dans la documentation Amazon EMR. Consultez également les sections de la section [Utiliser un cluster avec Iceberg](#) pour voir quelles fonctionnalités d'Iceberg sont prises en charge dans Amazon EMR sur différents frameworks.

Nous vous recommandons d'utiliser la dernière version d'Amazon EMR pour bénéficier de la dernière version prise en charge d'Iceberg. Les exemples de code et les configurations présentés dans cette section supposent que vous utilisez la version emr-7.8.0 d'Amazon EMR.

## Création d'un cluster Amazon EMR avec Iceberg

Pour créer un cluster Amazon EMR sur Amazon EC2 avec Iceberg installé, suivez les instructions de la documentation Amazon [EMR](#).

Plus précisément, votre cluster doit être configuré selon la classification suivante :

```
[{
  "Classification": "iceberg-defaults",
  "Properties": {
    "iceberg.enabled": "true"
  }
}]
```

```
}]
```

Vous pouvez également choisir d'utiliser Amazon EMR Serverless ou Amazon EMR on EKS comme options de déploiement pour vos charges de travail Iceberg, à partir d'Amazon EMR 6.6.0.

## Développement d'applications Iceberg dans Amazon EMR

Pour développer le code Spark pour vos applications Iceberg, vous pouvez utiliser [Amazon EMR Studio](#), un environnement de développement intégré (IDE) basé sur le Web pour les blocs-notes Jupyter entièrement gérés qui s'exécutent sur des clusters Amazon EMR.

### Utilisation des blocs-notes Amazon EMR Studio

Vous pouvez développer des applications Spark de manière interactive dans des blocs-notes Amazon EMR Studio Workspace et connecter ces blocs-notes à votre Amazon EMR sur des EC2 clusters ou à Amazon EMR sur des points de terminaison gérés par EKS. Consultez Service AWS la documentation pour obtenir des instructions sur la configuration d'un studio EMR pour Amazon [EMR sur et EC2 Amazon EMR sur](#) EKS.

Pour utiliser Iceberg dans EMR Studio, procédez comme suit :

1. Lancez un cluster Amazon EMR avec Iceberg activé, comme indiqué dans [Utiliser un cluster avec Iceberg](#) installé.
2. Configurez un studio EMR. Pour obtenir des instructions, consultez [Configuration d'un studio Amazon EMR](#).
3. Ouvrez un bloc-notes EMR Studio Workspace et exécutez le code suivant dans la première cellule du bloc-notes pour configurer votre session Spark afin d'utiliser Iceberg :

```
%%configure -f
{
  "conf": {
    "spark.sql.catalog.<catalog_name>": "org.apache.iceberg.spark.SparkCatalog",
    "spark.sql.catalog.<catalog_name>.warehouse": "s3://YOUR-BUCKET-NAME/YOUR-
FOLDER-NAME/",
    "spark.sql.catalog.<catalog_name>.type": "glue",
    "spark.sql.extensions":
    "org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions"
  }
}
```

où :

- `<catalog_name>` est le nom de votre catalogue de sessions Iceberg Spark. Remplacez-le par le nom de votre choix et n'oubliez pas de modifier les références dans toutes les configurations associées à ce catalogue. Dans votre code, vous pouvez faire référence à vos tables Iceberg en utilisant le nom complet de la table, y compris le nom du catalogue de sessions Spark, comme suit :

```
<catalog_name>.<database_name>.<table_name>
```

Vous pouvez également remplacer le catalogue par défaut par le catalogue Iceberg que vous avez défini en définissant le nom `spark.sql.defaultCatalog` de votre catalogue. Cette seconde approche vous permet de faire référence à des tables sans le préfixe de catalogue, ce qui peut simplifier vos requêtes.

- `<catalog_name>.warehouse` pointe vers le chemin Amazon S3 où vous souhaitez stocker vos données et métadonnées.
  - Pour que le catalogue soit un AWS Glue Data Catalog, définissez `spark.sql.catalog.<catalog_name>.type` sur `glue`. Cette clé est requise pour pointer vers une classe d'implémentation pour toute implémentation de catalogue personnalisé. La section [Bonnes pratiques générales](#) plus loin dans ce guide décrit les différents catalogues pris en charge par Iceberg.
4. Vous pouvez désormais commencer à développer votre application Spark pour Iceberg de manière interactive dans le bloc-notes, comme vous le feriez pour n'importe quelle autre application Spark.

Pour plus d'informations sur la configuration de Spark pour Apache Iceberg à l'aide d'Amazon EMR Studio, consultez le [billet de blog Créez un lac de données évolutif à hautes performances, conforme à l'ACID et évolutif à l'aide d'Apache Iceberg sur Amazon EMR](#).

## Exécution de tâches Iceberg dans Amazon EMR

Après avoir développé le code d'application Spark pour votre charge de travail Iceberg, vous pouvez l'exécuter sur n'importe quelle option de déploiement Amazon EMR compatible avec Iceberg (voir la FAQ [Amazon EMR](#)).

Comme pour les autres tâches Spark, vous pouvez soumettre du travail à un Amazon EMR sur EC2 cluster en ajoutant des étapes ou en soumettant des tâches Spark de manière interactive au nœud

principal. Pour exécuter une tâche Spark, consultez les pages de documentation Amazon EMR suivantes :

- Pour un aperçu des différentes options de soumission de travaux à un Amazon EMR sur un EC2 cluster et des instructions détaillées pour chaque option, consultez [Soumettre un travail à un cluster](#).
- Pour Amazon EMR sur EKS, consultez [Exécuter des tâches Spark](#) avec StartJobRun
- [Pour EMR Serverless, voir Exécution de tâches](#).

Les sections suivantes fournissent un exemple pour chaque option de déploiement d'Amazon EMR.

## Amazon EMR sur EC2

Vous pouvez suivre les étapes suivantes pour soumettre le job Iceberg Spark :

1. Créez le fichier `emr_step_iceberg.json` avec le contenu suivant sur votre poste de travail :

```
[{
  "Name": "iceberg-test-job",
  "Type": "spark",
  "ActionOnFailure": "CONTINUE",
  "Args": [
    "--deploy-mode",
    "client",
    "--conf",

    "spark.sql.extensions=org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions",
    "--conf",
    "spark.sql.catalog.<catalog_name>=org.apache.iceberg.spark.SparkCatalog",
    "--conf",
    "spark.sql.catalog.<catalog_name>.type=glue",
    "--conf",
    "spark.sql.catalog.<catalog_name>.warehouse=s3://YOUR-BUCKET-NAME/YOUR-
FOLDER-NAME/",
    "s3://YOUR-BUCKET-NAME/code/iceberg-job.py"
  ]
}]
```

2. Modifiez le fichier de configuration pour votre tâche Spark spécifique en personnalisant les options de configuration d'Iceberg surlignées en gras.

3. Soumettez l'étape en utilisant le AWS Command Line Interface (AWS CLI). Exécutez la commande dans le répertoire où se trouve le `emr_step_iceberg.json` fichier.

```
aws emr add-steps --cluster-id <cluster_id> --steps file://emr_step_iceberg.json
```

## Amazon EMR sans serveur

Pour soumettre une tâche Iceberg Spark à EMR Serverless à l'aide de : AWS CLI

1. Créez le fichier `emr_serverless_iceberg.json` avec le contenu suivant sur votre poste de travail :

```
{
  "applicationId": "<APPLICATION_ID>",
  "executionRoleArn": "<ROLE_ARN>",
  "name": "iceberg-test-job",
  "jobDriver": {
    "sparkSubmit": {
      "entryPoint": "s3://YOUR-BUCKET-NAME/code/iceberg-job.py",
      "entryPointArguments": []
    }
  },
  "configurationOverrides": {
    "applicationConfiguration": [{
      "classification": "spark-defaults",
      "properties": {
        "spark.sql.extensions":
"org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions",
        "spark.sql.catalog.<catalog_name>":
"org.apache.iceberg.spark.SparkCatalog",
        "spark.sql.catalog.<catalog_name>.type": "glue",
        "spark.sql.catalog.<catalog_name>.warehouse": "s3://YOUR-BUCKET-NAME/
YOUR-FOLDER-NAME/",
        "spark.jars": "/usr/share/aws/iceberg/lib/iceberg-spark3-runtime.jar",
        "spark.hadoop.hive.metastore.client.factory.class": "com.amazonaws.glue.catalog.metastore.AWSGlueCatalogMetastoreClientFactory"
      }
    }],
    "monitoringConfiguration": {
      "s3MonitoringConfiguration": {
        "logUri": "s3://YOUR-BUCKET-NAME/emr-serverless/logs/"
      }
    }
  }
}
```

```

    }
  }
}

```

2. Modifiez le fichier de configuration pour votre tâche Spark spécifique en personnalisant les options de configuration d'Iceberg surlignées en gras.
3. Soumettez le travail à l'aide du AWS CLI. Exécutez la commande dans le répertoire où se trouve le `emr_serverless_iceberg.json` fichier :

```
aws emr-serverless start-job-run --cli-input-json file://emr_serverless_iceberg.json
```

Pour soumettre une tâche Iceberg Spark à EMR Serverless à l'aide de la console EMR Studio, procédez comme suit :

1. Suivez les instructions de la documentation [EMR Serverless](#).
2. Pour la configuration de Job, utilisez la configuration Iceberg pour Spark fournie pour Iceberg AWS CLI et personnalisez les champs surlignés pour Iceberg. Pour obtenir des instructions détaillées, consultez la section [Utilisation d'Apache Iceberg avec EMR](#) sans serveur dans la documentation Amazon EMR.

## Amazon EMR on EKS

Pour soumettre une tâche Iceberg Spark à Amazon EMR sur EKS à l'aide de : AWS CLI

1. Créez le fichier `emr_eks_iceberg.json` avec le contenu suivant sur votre poste de travail :

```

{
  "name": "iceberg-test-job",
  "virtualClusterId": "<VIRTUAL_CLUSTER_ID>",
  "executionRoleArn": "<ROLE_ARN>",
  "releaseLabel": "emr-6.9.0-latest",
  "jobDriver": {
    "sparkSubmitJobDriver": {
      "entryPoint": "s3://YOUR-BUCKET-NAME/code/iceberg-job.py",
      "entryPointArguments": [],
      "sparkSubmitParameters": "--jars local:///usr/share/aws/iceberg/lib/iceberg-spark3-runtime.jar"
    }
  }
}

```

```

    },
    "configurationOverrides": {
      "applicationConfiguration": [{
        "classification": "spark-defaults",
        "properties": {
          "spark.sql.extensions":
"org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions",
          "spark.sql.catalog.<catalog_name>":
"org.apache.iceberg.spark.SparkCatalog",
          "spark.sql.catalog.<catalog_name>.type": "glue",
          "spark.sql.catalog.<catalog_name>.warehouse": "s3://YOUR-BUCKET-NAME/
YOUR-FOLDER-NAME/",
          "spark.hadoop.hive.metastore.client.factory.class":
"com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory"
        }
      }],
      "monitoringConfiguration": {
        "persistentAppUI": "ENABLED",
        "s3MonitoringConfiguration": {
          "logUri": "s3://YOUR-BUCKET-NAME/emr-serverless/logs/"
        }
      }
    }
  }
}

```

2. Modifiez le fichier de configuration de votre tâche Spark en personnalisant les options de configuration d'Iceberg surlignées en gras.
3. Soumettez le travail à l'aide du AWS CLI. Exécutez la commande suivante dans le répertoire où se trouve le `emr_eks_iceberg.json` fichier :

```
aws emr-containers start-job-run --cli-input-json file://emr_eks_iceberg.json
```

Pour obtenir des instructions détaillées, consultez la section [Utilisation d'Apache Iceberg avec Amazon EMR sur EKS](#) dans la documentation Amazon EMR sur EKS.

## Bonnes pratiques pour Amazon EMR

Cette section fournit des directives générales pour le réglage des tâches Spark dans Amazon EMR afin d'optimiser la lecture et l'écriture des données dans les tables Iceberg. Pour connaître les

meilleures pratiques spécifiques à Iceberg, consultez la section [Meilleures pratiques](#) plus loin dans ce guide.

- Utilisez la dernière version d'Amazon EMR : Amazon EMR fournit des optimisations Spark prêtes à l'emploi avec le runtime Amazon EMR Spark. AWS améliore les performances du moteur d'exécution Spark à chaque nouvelle version.
- Déterminez l'infrastructure optimale pour vos charges de travail Spark : les charges de travail Spark peuvent nécessiter différents types de matériel pour différentes caractéristiques de travail afin de garantir des performances optimales. Amazon EMR [prend en charge plusieurs types d'instances](#) (optimisées pour le calcul, pour la mémoire, pour usage général et pour le stockage) afin de répondre à tous les types d'exigences de traitement. Lorsque vous intégrez de nouvelles charges de travail, nous vous recommandons de les comparer à des types d'instances généraux tels que M5 ou M6g. Surveillez le système d'exploitation (OS) et les métriques YARN de Ganglia et Amazon CloudWatch afin de déterminer les goulots d'étranglement du système (processeur, mémoire, stockage et E/S) en cas de charge maximale et choisissez le matériel approprié.
- **spark.sql.shuffle.partitions**Régler : définissez la `spark.sql.shuffle.partitions` propriété sur le nombre total de cœurs virtuels (vCores) de votre cluster ou sur un multiple de cette valeur (généralement, 1 à 2 fois le nombre total de vCores). Ce paramètre affecte le parallélisme de Spark lorsque vous utilisez le partitionnement par hachage et par plage comme mode de distribution d'écriture. Il demande un shuffle avant d'écrire pour organiser les données, ce qui garantit l'alignement des partitions.
- Activer le dimensionnement géré : dans presque tous les cas d'utilisation, nous vous recommandons d'activer le dimensionnement géré et l'allocation dynamique. Toutefois, si votre charge de travail présente un schéma prévisible, nous vous suggérons de désactiver le dimensionnement automatique et l'allocation dynamique. Lorsque le dimensionnement géré est activé, nous vous recommandons d'utiliser des instances Spot pour réduire les coûts. Utilisez des instances Spot pour les nœuds de tâches plutôt que pour les nœuds principaux ou principaux. Lorsque vous utilisez des instances Spot, utilisez des flottes d'instances avec plusieurs types d'instances par flotte pour garantir la disponibilité des instances ponctuelles.
- Utilisez la jointure par diffusion lorsque cela est possible : la jointure par diffusion (côté carte) est la meilleure jointure, à condition que l'une de vos tables soit suffisamment petite pour tenir dans la mémoire de votre plus petit nœud (dans l'ordre de MBs) et que vous effectuez une jointure equi (=). Tous les types de jointure, à l'exception des jointures externes complètes, sont pris en charge. Une jointure de diffusion diffuse la plus petite table sous forme de table de hachage sur tous les nœuds de travail en mémoire. Une fois que le petit tableau a été diffusé, vous ne pouvez pas y apporter de modifications. Comme la table de hachage se trouve localement dans la machine virtuelle

Java (JVM), elle peut être facilement fusionnée avec la grande table en fonction de la condition de jointure à l'aide d'une jointure par hachage. Les jointures de diffusion offrent des performances élevées en raison d'une surcharge minimale liée au shuffle.

- Régler le ramasse-miettes — Si les cycles de collecte des déchets (GC) sont lents, envisagez de passer du ramasse-miettes parallèle par défaut au G1GC pour de meilleures performances. Pour optimiser les performances du GC, vous pouvez affiner les paramètres du GC. Pour suivre les performances du GC, vous pouvez les surveiller à l'aide de l'interface utilisateur Spark. Idéalement, la durée du GC doit être inférieure ou égale à 1 % de l'exécution totale de la tâche.

# Travailler avec Iceberg dans AWS Glue

[AWS Glue](#) est un service d'intégration de données sans serveur qui facilite la découverte, la préparation, le déplacement et l'intégration de données provenant de sources multiples à des fins d'analyse, d'apprentissage automatique (ML) et de développement d'applications. L'une de ses principales fonctionnalités AWS Glue est sa capacité à effectuer des opérations d'extraction, de transformation et de chargement (ETL) de manière simple et rentable. Cela permet de classer vos données, de les nettoyer, de les enrichir et de les déplacer de manière fiable entre différents magasins de données et flux de données.

AWS Glue les [jobs](#) encapsulent des scripts qui définissent la logique de transformation à l'aide d'un environnement d'exécution [Apache Spark](#) ou Python. AWS Glue les tâches peuvent être exécutées à la fois en mode batch et en mode streaming.

Lorsque vous créez des tâches Iceberg dans AWS Glue, selon la version de AWS Glue, vous pouvez utiliser l'intégration native d'Iceberg ou une version personnalisée d'Iceberg pour associer des dépendances Iceberg à la tâche.

## Utilisation de l'intégration native d'Iceberg

AWS Glue les versions 3.0, 4.0 et 5.0 prennent en charge nativement les formats de lacs de données transactionnels tels qu'Apache Iceberg, Apache Hudi et Linux Foundation Delta Lake in for Spark. AWS Glue Cette fonctionnalité d'intégration simplifie les étapes de configuration requises pour commencer à utiliser ces frameworks dans AWS Glue.

Pour activer le support d'Iceberg pour votre AWS Glue tâche, définissez la tâche : choisissez l'onglet Détails de la tâche correspondant à votre AWS Glue tâche, accédez à Paramètres de la tâche sous Propriétés avancées, puis définissez la clé `--datalake-formats` et sa valeur sur `iceberg`.

Si vous créez une tâche à l'aide d'un bloc-notes, vous pouvez configurer le paramètre dans la première cellule du bloc-notes en utilisant la `%%configure` magie suivante :

```
%%configure
{
  "--conf" : <job-specific Spark configuration discussed later>,
  "--datalake-formats" : "iceberg"
}
```

La `iceberg` configuration de `--datalake-formats` in AWS Glue correspond à des versions spécifiques d'Iceberg en fonction de votre AWS Glue version :

| AWS Glue version | Version par défaut d'Iceberg |
|------------------|------------------------------|
| 5.0              | 1.7.1                        |
| 4.0              | 1.0.0                        |
| 3.0              | 0.13.1                       |

## Utilisation d'une version personnalisée d'Iceberg

Dans certains cas, vous souhaitez peut-être conserver le contrôle de la version d'Iceberg pour la tâche et la mettre à niveau à votre propre rythme. Par exemple, la mise à niveau vers une version ultérieure peut débloquer l'accès à de nouvelles fonctionnalités et à des améliorations de performances. Pour utiliser une version spécifique d'Iceberg AWS Glue, vous pouvez fournir vos propres fichiers JAR.

Avant d'implémenter une version personnalisée d'Iceberg, vérifiez la compatibilité avec votre AWS Glue environnement en consultant la section des [AWS Glue versions](#) de la AWS Glue documentation. Par exemple, la AWS Glue version 5.0 nécessite une compatibilité avec Spark 3.5.4.

Par exemple, pour exécuter des AWS Glue tâches utilisant la version 1.9.1 d'Iceberg, procédez comme suit :

1. Procurez-vous et téléchargez les fichiers JAR requis sur Amazon S3 :
  - a. [Téléchargez iceberg-spark-runtime-3.5\\_2.12-1.9.1.jar et -1.9.1.jar depuis le dépôt Apache Maven. iceberg-aws-bundle](#)
  - b. Téléchargez ces fichiers dans l'emplacement de compartiment S3 que vous avez désigné (par exemple, `s3://your-bucket-name/jars/`).
2. Configurez les paramètres de votre AWS Glue tâche comme suit :
  - a. Spécifiez le chemin S3 complet vers les deux fichiers JAR dans le `--extra-jars` paramètre, en les séparant par une virgule (par exemple, `s3://your-bucket-name/jars/iceberg-spark-runtime-3.5_2.12-1.9.1.jar,s3://your-bucket-name/jars/iceberg-aws-bundle-1.9.1.jar`).
  - b. N'incluez pas `iceberg` comme valeur du paramètre `--datalake-formats`.

- c. Si vous utilisez la AWS Glue version 5.0, vous devez définir le `--user-jars-first` paramètre sur `true`.

## Configurations Spark pour Iceberg dans AWS Glue

Cette section décrit les configurations Spark requises pour créer une tâche AWS Glue ETL pour un ensemble de données Iceberg. Vous pouvez définir ces configurations en utilisant la clé `--conf` Spark avec une liste séparée par des virgules de toutes les clés et valeurs de configuration Spark. Vous pouvez utiliser la `%%configure` magie d'un bloc-notes ou de la section Paramètres du Job de la AWS Glue Studio console.

```
%glue_version 5.0

%%configure
{
  "--conf" : "spark.sql.extensions=org.apache.iceberg.spark.extensions...",
  "--datalake-formats" : "iceberg"
}
```

Configurez la session Spark avec les propriétés suivantes :

- `<catalog_name>` est le nom du catalogue de votre session Iceberg Spark. Remplacez-le par le nom de votre choix et n'oubliez pas de modifier les références dans toutes les configurations associées à ce catalogue. Dans votre code, vous pouvez faire référence à vos tables Iceberg en utilisant le nom complet de la table, y compris le nom du catalogue de sessions Spark, comme suit :

`<catalog_name>.<database_name>.<table_name>`

Vous pouvez également remplacer le catalogue par défaut par le catalogue Iceberg que vous avez défini en définissant le nom `spark.sql.defaultCatalog` de votre catalogue. Vous pouvez utiliser cette seconde approche pour faire référence à des tables sans le préfixe de catalogue, ce qui peut simplifier vos requêtes.

- `<catalog_name>.<warehouse>` pointe vers le chemin Amazon S3 où vous souhaitez stocker vos données et métadonnées.
- Pour que le catalogue soit un AWS Glue Data Catalog, définissez `spark.sql.catalog.<catalog_name>.type` sur `glue`. Cette clé est requise pour pointer vers une classe d'implémentation pour toute implémentation de catalogue personnalisé. Pour les

catalogues pris en charge par Iceberg, consultez la section [Bonnes pratiques générales](#) plus loin dans ce guide.

Par exemple, si un catalogue est appelé `glue_iceberg`, vous pouvez configurer votre tâche à l'aide de plusieurs `--conf` clés comme suit :

```
%%configure
{
  "--datalake-formats" : "iceberg",
  "--conf" :
  "spark.sql.extensions=org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions
  --conf spark.sql.catalog.glue_iceberg=org.apache.iceberg.spark.SparkCatalog --
  conf spark.sql.catalog.glue_iceberg.warehouse=s3://<your-warehouse-dir>/ --conf
  spark.sql.catalog.glue_iceberg.type=glue"
}
```

Vous pouvez également utiliser du code pour ajouter les configurations ci-dessus à votre script Spark comme suit :

```
spark = SparkSession.builder\

  .config("spark.sql.extensions", "org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions") \
  .config("spark.sql.catalog.glue_iceberg",
  "org.apache.iceberg.spark.SparkCatalog")\
  .config("spark.sql.catalog.glue_iceberg.warehouse", "s3://<your-warehouse-dir>/")\
  .config("spark.sql.catalog.glue_iceberg.type", "glue") \
  .getOrCreate()
```

## Bonnes pratiques en matière AWS Glue d'emploi

Cette section fournit des directives générales pour ajuster les tâches Spark AWS Glue afin d'optimiser la lecture et l'écriture de données dans les tables Iceberg. Pour connaître les meilleures pratiques spécifiques à Iceberg, consultez la section [Meilleures pratiques](#) plus loin dans ce guide.

- Utilisez la dernière version de AWS Glue et mettez-la à niveau chaque fois que possible : les nouvelles versions de AWS Glue fournissent des améliorations de performances, des temps de démarrage réduits et de nouvelles fonctionnalités. Ils prennent également en charge les nouvelles versions de Spark qui peuvent être nécessaires pour les dernières versions d'Iceberg. Pour obtenir

la liste des AWS Glue versions disponibles et des versions de Spark qu'elles prennent en charge, consultez la [AWS Glue documentation](#).

- Optimisez la mémoire des AWS Glue tâches : suivez les recommandations du billet de AWS blog [Optimisez la gestion de la mémoire dans AWS Glue](#).
- Utiliser AWS Glue Auto Scaling : lorsque vous activez Auto Scaling, il ajuste AWS Glue automatiquement le nombre de AWS Glue travailleurs de manière dynamique en fonction de votre charge de travail. Cela permet de réduire le coût de votre AWS Glue travail pendant les périodes de pointe, car AWS Glue le nombre de travailleurs est réduit lorsque la charge de travail est faible et que les travailleurs restent inactifs. Pour utiliser AWS Glue Auto Scaling, vous devez spécifier un nombre maximum de travailleurs auxquels votre AWS Glue travail peut être adapté. Pour plus d'informations, consultez la section [Utilisation de la mise à l'échelle automatique pour](#) AWS Glue dans la AWS Glue documentation.
- Utilisez la version d'Iceberg souhaitée. L'intégration AWS Glue native pour Iceberg est la meilleure solution pour démarrer avec Iceberg. Toutefois, pour les charges de travail de production, nous vous recommandons d'ajouter des dépendances aux bibliothèques (comme indiqué [précédemment dans ce guide](#)) afin de contrôler totalement la version d'Iceberg. Cette approche vous permet de tirer parti des dernières fonctionnalités d'Iceberg et d'améliorer les performances dans le cadre de vos AWS Glue tâches.
- Activez l'interface utilisateur Spark pour la surveillance et le débogage : vous pouvez également utiliser l'[interface utilisateur Spark AWS Glue pour inspecter votre tâche Iceberg en](#) visualisant les différentes étapes d'une tâche Spark dans un graphe acyclique dirigé (DAG) et en surveillant les tâches en détail. L'interface utilisateur Spark constitue un moyen efficace de résoudre les problèmes et d'optimiser les tâches d'Iceberg. Par exemple, vous pouvez identifier les phases d'étranglement associées à de nombreux remaniements ou à des pertes de disque afin d'identifier les opportunités de réglage. Pour plus d'informations, consultez la section [Surveillance des tâches à l'aide de l'interface utilisateur Web d'Apache Spark](#) dans la AWS Glue documentation.

# Utilisation des tables Iceberg à l'aide d'Apache Spark

Cette section fournit une vue d'ensemble de l'utilisation d'Apache Spark pour interagir avec les tables Iceberg. Les exemples sont du code standard qui peut être exécuté sur Amazon EMR ou AWS Glue.

Remarque : L'interface principale pour interagir avec les tables Iceberg est SQL. La plupart des exemples associeront donc Spark SQL à l'API DataFrames.

## Création et écriture de tables Iceberg

Vous pouvez utiliser Spark SQL et Spark DataFrames pour créer et ajouter des données aux tables Iceberg.

### Utilisation de Spark SQL

Pour écrire un jeu de données Iceberg, utilisez des instructions SQL Spark standard telles que `CREATE TABLE` et `INSERT INTO`.

### Tables non partitionnées

Voici un exemple de création d'une table Iceberg non partitionnée avec Spark SQL :

```
spark.sql(f"""
    CREATE TABLE IF NOT EXISTS {CATALOG_NAME}.{DB_NAME}.{TABLE_NAME}_nopartitions (
        c_customer_sk          int,
        c_customer_id          string,
        c_first_name            string,
        c_last_name             string,
        c_birth_country         string,
        c_email_address         string)
    USING iceberg
    OPTIONS ('format-version'='2')
    """)
```

Pour insérer des données dans une table non partitionnée, utilisez une instruction standard `INSERT INTO` :

```
spark.sql(f"""
    INSERT INTO {CATALOG_NAME}.{DB_NAME}.{TABLE_NAME}_nopartitions
    SELECT c_customer_sk, c_customer_id, c_first_name, c_last_name, c_birth_country,
           c_email_address
```

```
FROM another_table  
""")
```

## Tables partitionnées

Voici un exemple de création d'une table Iceberg partitionnée avec Spark SQL :

```
spark.sql(f"""  
    CREATE TABLE IF NOT EXISTS {CATALOG_NAME}.{DB_NAME}.{TABLE_NAME}_withpartitions (  
        c_customer_sk            int,  
        c_customer_id            string,  
        c_first_name             string,  
        c_last_name              string,  
        c_birth_country          string,  
        c_email_address          string)  
    USING iceberg  
    PARTITIONED BY (c_birth_country)  
    OPTIONS ('format-version'='2')  
""")
```

Pour insérer des données dans une table Iceberg partitionnée avec Spark SQL, utilisez une instruction standard `INSERT INTO` :

```
spark.sql(f"""  
INSERT INTO {CATALOG_NAME}.{DB_NAME}.{TABLE_NAME}_withpartitions  
SELECT c_customer_sk, c_customer_id, c_first_name, c_last_name, c_birth_country,  
       c_email_address  
FROM another_table  
""")
```

### Note

À partir de la version 1.5.0 d'Iceberg, le mode de distribution d'hashécriture est le mode par défaut lorsque vous insérez des données dans des tables partitionnées. Pour plus d'informations, consultez la section [Writing Distribution Modes](#) dans la documentation d'Iceberg.

## Utilisation de l' DataFrames API

Pour écrire un jeu de données Iceberg, vous pouvez utiliser l'`DataFrameWriterV2API`.

Pour créer une table Iceberg et y écrire des données, utilisez la fonction `df.writeTo(t)`. Si la table existe, utilisez la `.append()` fonction. Si ce n'est pas le cas, utilisez `.create()`. Les exemples suivants utilisent `.createOrReplace()`, qui est une variante de `.create()` ce qui équivaut à `CREATE OR REPLACE TABLE AS SELECT`.

## Tables non partitionnées

Pour créer et remplir une table Iceberg non partitionnée à l'aide de l'API : `DataFrameWriterV2`

```
input_data.writeTo(f"{CATALOG_NAME}.{DB_NAME}.{TABLE_NAME}_nopartitions") \
    .tableProperty("format-version", "2") \
    .createOrReplace()
```

Pour insérer des données dans une table Iceberg non partitionnée existante à l'aide de l'API : `DataFrameWriterV2`

```
input_data.writeTo(f"{CATALOG_NAME}.{DB_NAME}.{TABLE_NAME}_nopartitions") \
    .append()
```

## Tables partitionnées

Pour créer et remplir une table Iceberg partitionnée à l'aide de l'API : `DataFrameWriterV2`

```
input_data.writeTo(f"{CATALOG_NAME}.{DB_NAME}.{TABLE_NAME}_withpartitions") \
    .tableProperty("format-version", "2") \
    .partitionedBy("c_birth_country") \
    .createOrReplace()
```

Pour insérer des données dans une table Iceberg partitionnée à l'aide de l'`DataFrameWriterV2API` :

```
input_data.writeTo(f"{CATALOG_NAME}.{DB_NAME}.{TABLE_NAME}_withpartitions") \
    .append()
```

## Mise à jour des données dans les tables Iceberg

L'exemple suivant montre comment mettre à jour les données d'une table Iceberg. Cet exemple modifie toutes les lignes dont la `c_customer_sk` colonne contient un nombre pair.

```
spark.sql(f"""
UPDATE {CATALOG_NAME}.{db.name}.{table.name}
SET c_email_address = 'even_row'
WHERE c_customer_sk % 2 == 0
""")
```

Cette opération utilise la stratégie de copie sur écriture par défaut, de sorte qu'elle réécrit tous les fichiers de données concernés.

## Insertion de données dans des tables Iceberg

La modification des données fait référence à l'insertion de nouveaux enregistrements de données et à la mise à jour des enregistrements de données existants en une seule transaction. Pour insérer des données dans une table Iceberg, vous devez utiliser l'SQL `MERGE INTO` instruction.

L'exemple suivant insère le contenu de la table `{UPSERT_TABLE_NAME}` à l'intérieur de la table : `{TABLE_NAME}`

```
spark.sql(f"""
MERGE INTO {CATALOG_NAME}.{DB_NAME}.{TABLE_NAME} t
USING {UPSERT_TABLE_NAME} s
ON t.c_customer_id = s.c_customer_id
WHEN MATCHED THEN UPDATE SET t.c_email_address = s.c_email_address
WHEN NOT MATCHED THEN INSERT *
""")
```

- Si un enregistrement client présent existe `{UPSERT_TABLE_NAME}` déjà dans le même dossier `c_customer_id`, la valeur de l'`{UPSERT_TABLE_NAME}` enregistrement remplace la `c_email_address` valeur existante (opération de mise à jour). `{TABLE_NAME}`
- Si un enregistrement client qui se trouve dans `{UPSERT_TABLE_NAME}` n'existe pas dans `{TABLE_NAME}`, l'`{UPSERT_TABLE_NAME}` enregistrement est ajouté `{TABLE_NAME}` (opération d'insertion).

## Supprimer des données dans les tables Iceberg

Pour supprimer des données d'une table Iceberg, utilisez l'`DELETE FROM` expression et spécifiez un filtre correspondant aux lignes à supprimer.

```
spark.sql(f"""
DELETE FROM {CATALOG_NAME}.{db.name}.{table.name}
WHERE c_customer_sk % 2 != 0
""")
```

Si le filtre correspond à une partition entière, Iceberg supprime uniquement les métadonnées et laisse les fichiers de données en place. Dans le cas contraire, il réécrit uniquement les fichiers de données concernés.

La méthode de suppression prend les fichiers de données concernés par la WHERE clause et en crée une copie sans les enregistrements supprimés. Il crée ensuite un nouvel instantané de table qui pointe vers les nouveaux fichiers de données. Par conséquent, les enregistrements supprimés sont toujours présents dans les anciens instantanés de la table. Par exemple, si vous récupérez l'instantané précédent de la table, vous verrez les données que vous venez de supprimer. Pour plus d'informations sur la suppression d'anciens instantanés inutiles avec les fichiers de données associés à des fins de nettoyage, consultez la section [Gestion des fichiers à l'aide du compactage](#) plus loin dans ce guide.

## Lecture de données

Vous pouvez consulter l'état le plus récent de vos tables Iceberg dans Spark à la fois avec Spark SQL et DataFrames.

Exemple d'utilisation de Spark SQL :

```
spark.sql(f"""
SELECT * FROM {CATALOG_NAME}.{db.name}.{table.name} LIMIT 5
""")
```

Exemple d'utilisation de l' DataFrames API :

```
df = spark.table(f"{CATALOG_NAME}.{DB_NAME}.{TABLE_NAME}").limit(5)
```

## Utiliser le voyage dans le temps

Chaque opération d'écriture (insertion, mise à jour, modification, suppression) dans une table Iceberg crée un nouvel instantané. Vous pouvez ensuite utiliser ces instantanés pour voyager dans le temps, pour revenir dans le temps et vérifier le statut d'un tableau dans le passé.

Pour plus d'informations sur la façon de récupérer l'historique des instantanés de tables en utilisant `snapshot-id` et en chronométrant des valeurs, consultez la section [Accès aux métadonnées](#) plus loin dans ce guide.

La requête de voyage dans le temps suivante affiche le statut d'une table en fonction d'une donnée spécifique `snapshot-id`.

À l'aide de Spark SQL :

```
spark.sql(f"""
SELECT * FROM {CATALOG_NAME}.{DB_NAME}.{TABLE_NAME} VERSION AS OF {snapshot_id}
""")
```

À l'aide de DataFrames l'API :

```
df_1st_snapshot_id = spark.read.option("snapshot-id", snapshot_id) \
    .format("iceberg") \
    .load(f"{CATALOG_NAME}.{DB_NAME}.{TABLE_NAME}") \
    .limit(5)
```

La requête de voyage dans le temps suivante affiche le statut d'une table en fonction du dernier instantané créé avant un horodatage spécifique, en millisecondes (). `as-of-timestamp`

À l'aide de Spark SQL :

```
spark.sql(f"""
SELECT * FROM dev.{db.name}.{table.name} TIMESTAMP AS OF '{snapshot_ts}'
""")
```

À l'aide de DataFrames l'API :

```
df_1st_snapshot_ts = spark.read.option("as-of-timestamp", snapshot_ts) \
    .format("iceberg") \
    .load(f"dev.{DB_NAME}.{TABLE_NAME}") \
    .limit(5)
```

## Utilisation de requêtes incrémentielles

Vous pouvez également utiliser les instantanés Iceberg pour lire les données ajoutées de manière incrémentielle.

Remarque : Actuellement, cette opération prend en charge la lecture de données à partir de append snapshots. Il ne prend pas en charge l'extraction de données à partir d'opérations telles que replaceoverwrite, ou delete. De plus, les opérations de lecture incrémentielles ne sont pas prises en charge dans la syntaxe SQL de Spark.

L'exemple suivant récupère tous les enregistrements ajoutés à une table Iceberg entre l'instantané start-snapshot-id (exclusif) et end-snapshot-id (inclus).

```
df_incremental = (spark.read.format("iceberg")
    .option("start-snapshot-id", snapshot_id_start)
    .option("end-snapshot-id", snapshot_id_end)
    .load(f"glue_catalog.{DB_NAME}.{TABLE_NAME}"))
)
```

## Accès aux métadonnées

Iceberg fournit un accès à ses métadonnées via SQL. Vous pouvez accéder aux métadonnées d'une table donnée (<table\_name>) en interrogeant l'espace de noms <table\_name>.<metadata\_table>. Pour obtenir la liste complète des tables de métadonnées, consultez la section [Inspection des tables](#) dans la documentation d'Iceberg.

L'exemple suivant montre comment accéder à la table de métadonnées d'historique d'Iceberg, qui présente l'historique des validations (modifications) d'une table Iceberg.

En utilisant Spark SQL (avec la %%sql magie) à partir d'un bloc-notes Amazon EMR Studio :

```
Spark.sql(f"""
SELECT * FROM {CATALOG_NAME}.{DB_NAME}.{TABLE_NAME}.history LIMIT 5
""")
```

À l'aide de DataFrames l'API :

```
spark.read.format("iceberg").load("{CATALOG_NAME}.{DB_NAME}."
{TABLE_NAME}.history").show(5, False)
```

Exemple de sortie :

|       |                                  |                     |                     |                     |      |     |
|-------|----------------------------------|---------------------|---------------------|---------------------|------|-----|
| Type: | Table                            | Pie                 | Scatter             | Line                | Area | Bar |
|       | made_current_at                  | snapshot_id         | parent_id           | is_current_ancestor |      |     |
|       | 2023-01-09 02:50:17.547000+00:00 | 7501027970051178613 | 6598755163776233735 | True                |      |     |
|       | 2023-01-12 05:39:29.567000+00:00 | 7069175828427777019 | 7501027970051178613 | True                |      |     |
|       | 2023-01-12 05:39:58.807000+00:00 | 5173022175861138222 | 7069175828427777019 | True                |      |     |
|       | 2023-01-12 05:40:18.499000+00:00 | 3703414997660223390 | 5173022175861138222 | True                |      |     |
|       | 2023-01-12 05:40:41.827000+00:00 | 3807904412292252460 | 3703414997660223390 | True                |      |     |

# Utilisation des tables Iceberg à l'aide de Trino

Cette section explique comment configurer et utiliser les tables Iceberg à l'aide de [Trino](#) sur Amazon [EMR](#). Les exemples sont du code standard que vous pouvez exécuter sur un cluster Amazon EMR sur EC2. Les exemples de code et les configurations présentés dans cette section supposent que vous utilisez la version emr-7.9.0 d'Amazon EMR.

## Configuration d'Amazon EMR sur EC2

1. Créez un `iceberg.properties` fichier avec le contenu suivant. Le `iceberg.file-format=parquet` paramètre détermine le format de stockage par défaut pour les nouvelles tables si le format n'est pas explicitement spécifié dans l'`CREATE TABLE` instruction.

```
connector.name=iceberg
iceberg.catalog.type=glue
iceberg.file-format=parquet
fs.native-s3.enabled=true
```

2. Chargez le fichier `iceberg.properties` dans votre compartiment S3.
3. Créez une action bootstrap qui copie le `iceberg.properties` fichier depuis votre compartiment S3 et le stocke en tant que fichier de configuration Trino sur le cluster Amazon EMR que vous allez créer. Assurez-vous de le remplacer `<S3-bucket-name>` par le nom de votre compartiment S3.

```
#!/bin/bash
set -ex
sudo aws s3 cp s3://<S3-bucket-name>/iceberg.properties /etc/trino/conf/catalog/
iceberg.properties
```

4. Créez un cluster Amazon EMR sur lequel Trino est installé et spécifiez l'exécution du script précédent sous forme d'action d'amorçage. Voici un exemple de commande AWS Command Line Interface (AWS CLI) pour créer le cluster :

```
aws emr create-cluster --release-label emr-7.9.0 \
--applications Name=Trino \
--region <region> \
--name Trino_Iceberg_Cluster \
--bootstrap-actions '[{"Path":"s3://<S3-bucket-name>/bootstrap.sh", "Name":"Add
iceberg.properties"}]' \
```

```
--instance-groups
' [{"InstanceGroupType": "MASTER", "InstanceCount": 1, "InstanceType": "m5.xlarge"},
{"InstanceGroupType": "CORE", "InstanceCount": 3, "InstanceType": "m5.xlarge"} ] ' \
--service-role "<IAM-service-role>" \
--ec2-attributes '{"KeyName": "<key-name>", "InstanceProfile": "<EMR-EC2-instance-
profile>"}'
```

où vous remplacez :

- <S3-bucket-name> avec le nom de votre compartiment S3
  - <region> avec votre spécifique Région AWS
  - <key-name> avec votre paire de clés. Si la paire de clés n'existe pas, elle sera créée.
  - <IAM-service-role> avec votre rôle de service Amazon EMR qui respecte le [principe du moindre privilège](#).
  - <EMR-EC2-instance-profile> avec votre [profil d'instance](#).
5. Lorsque le cluster Amazon EMR a été initialisé, vous pouvez initialiser une session Trino en exécutant la commande suivante :

```
trino-cli
```

6. Dans la CLI Trino, vous pouvez consulter les catalogues en exécutant :

```
SHOW CATALOGS;
```

## Création de tables Iceberg

Pour créer une table Iceberg, vous pouvez utiliser l'`CREATE TABLE` instruction. Voici un exemple de création d'une table partitionnée utilisant le partitionnement caché d'Iceberg :

```
CREATE TABLE iceberg.iceberg_db.iceberg_table (
    userid int,
    firstname varchar,
    city varchar)
WITH (
    format = 'PARQUET',
    partitioning = ARRAY['city', 'bucket(userid, 16)'],
    location = 's3://<S3-bucket>/<prefix>');
```

 Note

Si vous ne spécifiez pas le format, la `iceberg.file-format` valeur que vous avez configurée dans la section précédente sera utilisée.

Pour insérer des données, utilisez la `INSERT INTO` commande. Voici un exemple :

```
INSERT INTO iceberg.iceberg_db.iceberg_table (userid, firstname, city)
VALUES
  (1001, 'John', 'New York'),
  (1002, 'Mary', 'Los Angeles'),
  (1003, 'Mateo', 'Chicago'),
  (1004, 'Shirley', 'Houston'),
  (1005, 'Diego', 'Miami'),
  (1006, 'Nikki', 'Seattle'),
  (1007, 'Pat', 'Boston'),
  (1008, 'Terry', 'San Francisco'),
  (1009, 'Richard', 'Denver'),
  (1010, 'Pat', 'Phoenix');
```

## Lecture depuis les tables Iceberg

Vous pouvez consulter le dernier statut de votre table Iceberg à l'aide d'une `SELECT` instruction, comme suit :

```
SELECT * FROM iceberg.iceberg_db.iceberg_table;
```

## Insertion de données dans des tables Iceberg

Vous pouvez effectuer une opération upsert (insérer simultanément de nouveaux enregistrements et mettre à jour les enregistrements existants) à l'aide de l'`MERGE INTO` instruction. Voici un exemple :

```
MERGE INTO iceberg.iceberg_db.iceberg_table target
USING (
  VALUES
    (1001, 'John Updated', 'Boston'),      -- Update existing user
    (1002, 'Mary Updated', 'Seattle'),     -- Update existing user
    (1011, 'Martha', 'Portland'),          -- Insert new user
```

```
        (1012, 'Paulo', 'Austin')           -- Insert new user
    ) AS source (userid, firstname, city)
    ON target.userid = source.userid
    WHEN MATCHED THEN
        UPDATE SET
            firstname = source.firstname,
            city = source.city
    WHEN NOT MATCHED THEN
        INSERT (userid, firstname, city)
        VALUES (source.userid, source.firstname, source.city);
```

## Supprimer des enregistrements des tables Iceberg

Pour supprimer des données d'une table Iceberg, utilisez l'`DELETE FROM` expression et spécifiez un filtre correspondant aux lignes à supprimer. Voici un exemple :

```
DELETE FROM iceberg.iceberg_db.iceberg_table WHERE userid IN (1003, 1004);
```

## Interrogation des métadonnées d'une table Iceberg

Iceberg fournit un accès à ses métadonnées via SQL. Vous pouvez accéder aux métadonnées d'une table donnée (`<table_name>`) en interrogeant l'espace de noms "`<table_name> . $<metadata_table>`". Pour obtenir la liste complète des tables de métadonnées, consultez la section [Inspection des tables](#) dans la documentation d'Iceberg.

Voici un exemple de liste de requêtes permettant d'inspecter les métadonnées d'Iceberg :

```
SELECT FROM iceberg.iceberg_db."iceberg_table$snapshots";
SELECT FROM iceberg.iceberg_db."iceberg_table$history";
SELECT FROM iceberg.iceberg_db."iceberg_table$partitions";
SELECT FROM iceberg.iceberg_db."iceberg_table$files";
SELECT FROM iceberg.iceberg_db."iceberg_table$manifests";
SELECT FROM iceberg.iceberg_db."iceberg_table$refs";
SELECT * FROM iceberg.iceberg_db."iceberg_table$metadata_log_entries";
```

Par exemple, cette requête :

```
SELECT * FROM iceberg.iceberg_db."iceberg_table$snapshots";
```

fournit le résultat :

```
trino> SELECT * FROM iceberg.iceberg_db."iceberg_table$snapshots";
```

| committed_at                | snapshot_id         | parent_id           | operation | manifest_list |
|-----------------------------|---------------------|---------------------|-----------|---------------|
| 2025-05-28 16:05:41.801 UTC | 7785073462465010154 | NULL                | append    | s3://         |
| 2025-05-28 16:05:57.806 UTC | 5984821362426775846 | 7785073462465010154 | append    | s3://         |
| 2025-05-28 16:09:40.268 UTC | 241938428756831817  | 5984821362426775846 | overwrite | s3://         |
| 2025-05-28 16:18:53.126 UTC | 1784832837567742464 | 241938428756831817  | delete    | s3://         |

(4 rows)

Query 20250528\_162032\_00012\_uhduz, FINISHED, 1 node  
 Splits: 1 total, 1 done (100.00%)  
 0.30 [4 rows, 3.11KiB] [13 rows/s, 10.3KiB/s]

## Utiliser le voyage dans le temps

Chaque opération d'écriture (insertion, mise à jour, modification ou suppression) dans une table Iceberg crée un nouvel instantané. Vous pouvez ensuite utiliser ces instantanés pour voyager dans le temps, pour revenir dans le temps et vérifier le statut d'un tableau dans le passé.

La requête de voyage dans le temps suivante affiche le statut d'une table en fonction d'un paramètre spécifique `snapshot_id` :

```
SELECT *
FROM iceberg.iceberg_db.iceberg_table FOR VERSION AS OF 241938428756831817;
```

La requête de voyage dans le temps suivante affiche le statut d'une table en fonction d'un horodatage spécifique :

```
SELECT *
FROM iceberg.iceberg_db.iceberg_table FOR TIMESTAMP AS OF TIMESTAMP '2025-05-28
16:09:40.268 UTC'
```

## Considérations relatives à l'utilisation d'Iceberg avec Trino

Les opérations d'écriture de Trino sur les tables Iceberg suivent le modèle de [fusion lors de la lecture](#), de sorte qu'elles créent des fichiers de suppression positionnels au lieu de réécrire des fichiers de données entiers affectés par les mises à jour ou les suppressions. Si vous souhaitez utiliser l'approche copier-sur-écriture, pensez à utiliser Spark pour les opérations d'écriture.

# Utilisation des tables Iceberg à l'aide d'Amazon Data Firehose

Amazon Data Firehose est un service sans serveur et sans code permettant de diffuser des flux de données provenant de plus de 20 sources telles que les logs AWS WAF , Amazon Logs, Amazon CloudWatch Kinesis Data Streams et Amazon Managed Streaming for Apache Kafka (Amazon MSK) vers des destinations telles qu'Amazon S3, Amazon Redshift, Snowflake et Splunk. AWS IoT

Vous pouvez utiliser Firehose pour transmettre directement des données de streaming aux tables Apache Iceberg dans Amazon S3. À l'aide de Firehose, vous pouvez acheminer les enregistrements d'un seul flux vers différentes tables Apache Iceberg et appliquer automatiquement des opérations d'insertion, de mise à jour et de suppression aux enregistrements des tables. Firehose garantit une livraison en une seule fois sur les tables Iceberg. Cette fonctionnalité nécessite l'utilisation du AWS Glue Data Catalog.

Firehose peut également fournir directement des données de streaming aux tables Amazon S3. Ces tables fournissent un stockage optimisé pour les charges de travail analytiques à grande échelle et incluent des fonctionnalités qui améliorent en permanence les performances des requêtes et réduisent les coûts de stockage des données tabulaires.

Pour plus d'informations sur la configuration d'un flux Firehose afin de fournir des données aux tables Apache Iceberg, consultez [Configurer le flux Firehose dans la documentation Firehose ou le billet de blog Diffusez des données en temps réel dans des tables Apache Iceberg dans Amazon S3 à l'aide d'Amazon Data Firehose](#).

# Utilisation de tables Iceberg à l'aide d'Athena SQL

Amazon Athena fournit un support intégré pour Apache Iceberg et ne nécessite aucune étape ni configuration supplémentaires. Cette section fournit un aperçu détaillé des fonctionnalités prises en charge et des conseils de haut niveau sur l'utilisation d'Athena pour interagir avec les tables Iceberg.

## Compatibilité des versions et des fonctionnalités

### Support pour les spécifications des tables Iceberg

La spécification des tables Apache Iceberg indique comment les tables Iceberg doivent se comporter. Athena prend en charge le format de table version 2, de sorte que toute table Iceberg que vous créez avec la console, la CLI ou le SDK utilise intrinsèquement cette version.

Si vous utilisez une table Iceberg créée avec un autre moteur, tel qu'Apache Spark sur Amazon EMR, AWS Glue ou assurez-vous de définir la version du format de la table à l'[aide](#) des propriétés de la table. À titre de référence, consultez la section [Création et écriture de tables Iceberg](#) plus haut dans ce guide.

### Support des fonctionnalités Iceberg

Vous pouvez utiliser Athena pour lire et écrire dans des tables Iceberg. Lorsque vous modifiez des données à l'aide des DELETE FROM instructionsUPDATE,MERGE INTO, et, Athena prend uniquement en charge le merge-on-read mode. Cette propriété ne peut pas être modifiée. Pour mettre à jour ou supprimer des données avec copy-on-write, vous devez utiliser d'autres moteurs tels que Apache Spark sur Amazon EMR ou. AWS Glue Le tableau suivant résume la prise en charge des fonctionnalités d'Iceberg dans Athena.

|  | Support DDL       |              | Support DML         |                    | AWS Lake Formation pour des raisons de sécurité (facultatif) |                  |
|--|-------------------|--------------|---------------------|--------------------|--------------------------------------------------------------|------------------|
|  | Format de tableau | Create table | Évolution du schéma | Lecture de données | Écrire des données                                           | Contrôle d'accès |
|  |                   |              |                     |                    |                                                              |                  |

|               |           | Support DDL |   | Support DML |                 | AWS Lake Formation pour des raisons de sécurité (facultatif) |
|---------------|-----------|-------------|---|-------------|-----------------|--------------------------------------------------------------|
|               |           |             |   |             |                 | aux lignes/ colonnes                                         |
| Amazon Athena | Version 2 | ✓           | ✓ | ✓           | XC opy-on-write | ✓                                                            |
|               |           |             |   |             | ✓ Merge-on-read | ✓                                                            |

#### Note

- Athena ne prend pas en charge les requêtes incrémentielles.
- Dans Athena, les opérations de mise à jour, de suppression et de fusion utilisent toujours par défaut la fusion en lecture (MoR), quels que soient les paramètres de copie en écriture (CoW) dans les propriétés de la table, car CoW n'est pas pris en charge.

## Utilisation des tables Iceberg

Pour un démarrage rapide de l'utilisation d'Iceberg dans Athena, consultez la [section Commencer à utiliser les tables Iceberg dans Athena SQL plus haut dans](#) ce guide.

Le tableau suivant répertorie les limites et les recommandations.

| Scénario                 | Limitation                                                                                                 | Recommandation                                                                             |
|--------------------------|------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------|
| Génération de tables DDL | Les tables Iceberg créées avec d'autres moteurs peuvent avoir des propriétés qui ne sont pas exposées dans | Utilisez l'instruction équivalente dans le moteur qui a créé la table (par exemple, l'SHOW |

| Scénario                                                                    | Limitation                                                                                                                 | Recommandation                                                                                                                                                                    |
|-----------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                                             | Athena. Pour ces tables, il n'est pas possible de générer le DDL.                                                          | CREATE TABLE instruction pour Spark).                                                                                                                                             |
| Préfixes Amazon S3 aléatoires dans les objets écrits dans une table Iceberg | Par défaut, la propriété est activée pour les tables Iceberg créées avec Athena. <code>write.object-storage.enabled</code> | Pour désactiver ce comportement et contrôler totalement les propriétés des tables Iceberg, créez une table Iceberg avec un autre moteur tel que Spark sur Amazon EMR ou. AWS Glue |
| Requêtes progressives                                                       | Non pris en charge actuellement dans Athena.                                                                               | Pour utiliser des requêtes incrémentielles afin d'activer des pipelines d'ingestion de données incrémentiels, utilisez Spark sur Amazon EMR ou. AWS Glue                          |

# Utilisation des tables Iceberg en utilisant Pylceberg

Cette section explique comment vous pouvez interagir avec les tables Iceberg en utilisant [Pylceberg](#). [Les exemples fournis sont du code standard que vous pouvez exécuter sur des EC2 instances, des AWS Lambda fonctions Amazon Linux 2023 ou tout autre environnement Python avec des informations d'identification correctement configurées.](#)[AWS](#)

## Prérequis

### Note

Ces exemples utilisent le [Pylceberg 1.9.1](#).

Pour travailler avec Pylceberg, vous devez Pylceberg l' AWS SDK pour Python (Boto3) installer. Voici un exemple de la façon dont vous pouvez configurer un environnement virtuel Python pour travailler avec Pylceberg et AWS Glue Data Catalog :

1. Téléchargez [Pylceberg](#) en utilisant le programme d'[installation du package pip python](#). Vous avez également besoin de [Boto3](#) pour interagir avec. Services AWS Vous pouvez configurer un environnement virtuel Python local à tester à l'aide des commandes suivantes :

```
python3 -m venv my_env
cd my_env/bin/
source activate
pip install "pyiceberg[pyarrow,pandas,glue]"
pip install boto3
```

2. Exécutez python pour ouvrir le shell Python et tester les commandes.

## Connexion au catalogue de données

Pour commencer à utiliser les tables Iceberg dans AWS Glue, vous devez d'abord vous connecter au AWS Glue Data Catalog.

La `load_catalog` fonction initialise une connexion au catalogue de données en créant un objet de [catalogue](#) qui sert d'interface principale pour toutes les opérations Iceberg :

```
from pyiceberg.catalog import load_catalog
region = "us-east-1"

glue_catalog = load_catalog(
    'default',
    **{
        'client.region': region
    },
    type='glue'
)
```

## Lister et créer des bases de données

Pour répertorier les bases de données existantes, utilisez la `list_namespaces` fonction :

```
databases = glue_catalog.list_namespaces()
print(databases)
```

Pour créer une nouvelle base de données, utilisez la `create_namespace` fonction :

```
database_name="mydb"
s3_db_path=f"s3://amzn-s3-demo-bucket/{database_name}"

glue_catalog.create_namespace(database_name, properties={"location": s3_db_path})
```

## Création et écriture de tables Iceberg

### Tables non partitionnées

Voici un exemple de création d'une table Iceberg non partitionnée à l'aide de la fonction :

`create_table`

```
from pyiceberg.schema import Schema
from pyiceberg.types import NestedField, StringType, DoubleType

database_name="mydb"
table_name="pyiceberg_table"
s3_table_path=f"s3://amzn-s3-demo-bucket/{database_name}/{table_name}"
```

```
schema = Schema(
    NestedField(1, "city", StringType(), required=False),
    NestedField(2, "lat", DoubleType(), required=False),
    NestedField(3, "long", DoubleType(), required=False),
)

glue_catalog.create_table(f"{database_name}.{table_name}", schema=schema,
    location=s3_table_path)
```

Vous pouvez utiliser cette `list_tables` fonction pour vérifier la liste des tables d'une base de données :

```
tables = glue_catalog.list_tables(namespace=database_name)
print(tables)
```

Vous pouvez utiliser la `append` fonction et insérer PyArrow des données dans une table Iceberg :

```
import pyarrow as pa
df = pa.Table.from_pylist([
    {"city": "Amsterdam", "lat": 52.371807, "long": 4.896029},
    {"city": "San Francisco", "lat": 37.773972, "long": -122.431297},
    {"city": "Drachten", "lat": 53.11254, "long": 6.0989},
    {"city": "Paris", "lat": 48.864716, "long": 2.349014},
],
)

table = glue_catalog.load_table(f"{database_name}.{table_name}")
table.append(df)
```

## Tables partitionnées

Voici un exemple de création d'une table Iceberg [partitionnée](#) avec [partitionnement masqué](#) à l'aide de la `create_table` fonction et : `PartitionSpec`

```
from pyiceberg.schema import Schema
from pyiceberg.types import (
    NestedField,
    StringType,
    FloatType,
    DoubleType,
```

```

        TimestampType,
    )

# Define the schema
schema = Schema(
    NestedField(field_id=1, name="datetime", field_type=TimestampType(),
required=True),
    NestedField(field_id=2, name="drone_id", field_type=StringType(), required=True),
    NestedField(field_id=3, name="lat", field_type=DoubleType(), required=False),
    NestedField(field_id=4, name="lon", field_type=DoubleType(), required=False),
    NestedField(field_id=5, name="height", field_type=FloatType(), required=False),
)

from pyiceberg.partitioning import PartitionSpec, PartitionField
from pyiceberg.transforms import DayTransform

partition_spec = PartitionSpec(
    PartitionField(
        source_id=1, # Refers to "datetime"
        field_id=1000,
        transform=DayTransform(),
        name="datetime_day"
    )
)

database_name="mydb"
partitioned_table_name="pyiceberg_table_partitioned"
s3_table_path=f"s3://amzn-s3-demo-bucket/{database_name}/{partitioned_table_name}"

glue_catalog.create_table(
    identifier=f"{database_name}.{partitioned_table_name}",
    schema=schema,
    location=s3_table_path,
    partition_spec=partition_spec
)

```

Vous pouvez insérer des données dans une table partitionnée de la même manière que dans une table non partitionnée. Le partitionnement est géré automatiquement.

```

from datetime import datetime
arrow_schema = pa.schema([
    pa.field("datetime", pa.timestamp("us"), nullable=False),
    pa.field("drone_id", pa.string(), nullable=False),

```

```
pa.field("lat", pa.float64()),
pa.field("lon", pa.float64()),
pa.field("height", pa.float32()),
])

data = [
    {
        "datetime": datetime(2024, 6, 1, 12, 0, 0),
        "drone_id": "drone_001",
        "lat": 52.371807,
        "lon": 4.896029,
        "height": 120.5,
    },
    {
        "datetime": datetime(2024, 6, 1, 12, 5, 0),
        "drone_id": "drone_002",
        "lat": 37.773972,
        "lon": -122.431297,
        "height": 150.0,
    },
    {
        "datetime": datetime(2024, 6, 2, 9, 0, 0),
        "drone_id": "drone_001",
        "lat": 53.11254,
        "lon": 6.0989,
        "height": 110.2,
    },
    {
        "datetime": datetime(2024, 6, 2, 9, 30, 0),
        "drone_id": "drone_003",
        "lat": 48.864716,
        "lon": 2.349014,
        "height": 145.7,
    },
]

df = pa.Table.from_pylist(data, schema=arrow_schema)

table = glue_catalog.load_table(f"{database_name}.{partitioned_table_name}")
table.append(df)
```

## Lecture de données

Vous pouvez utiliser cette Pylceberg scan fonction pour lire les données de vos tables Iceberg. Vous pouvez filtrer les lignes, sélectionner des colonnes spécifiques et limiter le nombre d'enregistrements renvoyés.

```
table= glue_catalog.load_table(f"{database_name}.{table_name}")
scan_df = table.scan(
    row_filter=(
        f"city = 'Amsterdam'"
    ),
    selected_fields=("city", "lat"),
    limit=100,
).to_pandas()

print(scan_df)
```

## Suppression des données

La Pylceberg delete fonction vous permet de supprimer des enregistrements de votre table en utilisant `delete_filter` :

```
table = glue_catalog.load_table(f"{database_name}.{table_name}")
table.delete(delete_filter="city == 'Paris'")
```

## Accès aux métadonnées

Pylceberg fournit plusieurs fonctions pour accéder aux métadonnées des tables. Voici comment afficher les informations relatives aux instantanés de table :

```
#List of snapshots
table.snapshots()

#Current snapshot
table.current_snapshot()

#Take a previous snapshot
second_last_snapshot_id=table.snapshots()[-2].snapshot_id
print(f"Second last SnapshotID: {second_last_snapshot_id}")
```

Pour une liste détaillée des métadonnées disponibles, consultez la section de référence du code des [métadonnées](#) de la Pylceberg documentation.

## Utiliser le voyage dans le temps

Vous pouvez utiliser des instantanés de tableau pour voyager dans le temps afin d'accéder aux états précédents de votre tableau. Voici comment afficher l'état de la table avant la dernière opération :

```
second_last_snapshot_id=table.snapshots()[-2].snapshot_id

time_travel_df = table.scan(
    limit=100,
    snapshot_id=second_last_snapshot_id
).to_pandas()

print(time_travel_df)
```

Pour une liste complète des fonctions disponibles, consultez la documentation de l'[API Pylceberg Python](#).

# Utilisation de la version 3 de la spécification du format de table Iceberg

La dernière version de la spécification du format de table Apache Iceberg est la version 3. Cette version introduit des fonctionnalités avancées pour créer des lacs de données à l'échelle du pétaoctet avec des performances améliorées et une réduction des frais d'exploitation. Il résout les problèmes de performances courants rencontrés avec la version 2, en particulier en ce qui concerne les mises à jour par lots et les opérations de suppression de conformité.

AWS prend en charge les vecteurs de suppression et le lignage des lignes tels que définis dans la spécification Iceberg version 3. Ces fonctionnalités sont disponibles avec Apache Spark sur les sites suivants Services AWS.

| Service AWS                                                                                  | Support de la version 3                         |
|----------------------------------------------------------------------------------------------|-------------------------------------------------|
| <a href="#">Amazon EMR pour Apache Spark</a>                                                 | Amazon EMR version 7.12 et versions ultérieures |
| <a href="#">AWS Glue</a>                                                                     | Oui                                             |
| AWS Glue: <a href="#">API REST Iceberg</a> , maintenance des tables                          | Oui                                             |
| <a href="#">Carnets de notes Amazon SageMaker Unified Studio</a>                             | Oui                                             |
| Tables Amazon S3 : <a href="#">API REST Iceberg</a> , maintenance des <a href="#">tables</a> | Oui                                             |
| <a href="#">Amazon Athéna (Turin)</a>                                                        | Non                                             |

## Principales fonctionnalités de la version 3

Les vecteurs de suppression remplacent les fichiers de suppression positionnels utilisés dans la version 2 par un format binaire efficace stocké sous forme de fichiers Puffin. Cela élimine l'amplification de l'écriture due aux mises à jour aléatoires par lots et aux suppressions conformes

au règlement général sur la protection des données (RGPD), et réduit considérablement les frais de maintenance liés à la mise à jour des données. Organisations traitant des mises à jour fréquentes constateront des améliorations immédiates des performances d'écriture et une réduction des coûts de stockage grâce à la réduction du nombre de petits fichiers.

Le lignage des lignes permet un suivi précis des modifications au niveau des lignes. Vos systèmes en aval peuvent traiter les modifications de manière incrémentielle, accélérant ainsi les pipelines de données et réduisant les coûts de calcul pour les flux de travail de capture des données de modification (CDC). Cette fonctionnalité intégrée élimine le besoin d'implémentations personnalisées de suivi des modifications.

## Compatibilité des versions

La version 3 maintient la rétrocompatibilité avec les tables de la version 2. Les services AWS prennent en charge les tables des versions 2 et 3 simultanément, ce qui vous permet de :

- Exécutez des requêtes sur les tables des versions 2 et 3.
- Mettez à niveau les tables existantes de la version 2 vers la version 3 sans réécriture des données.
- Exécutez des requêtes de voyage dans le temps couvrant des instantanés des versions 2 et 3.
- Utilisez l'évolution du schéma et le partitionnement masqué entre les versions des tables.

## Commencer à utiliser la version 3

### Conditions préalables

Avant de travailler avec les tables de la version 3, assurez-vous que vous disposez des éléments suivants :

- Et Compte AWS avec les autorisations Gestion des identités et des accès AWS (IAM) appropriées.
- Accès à un ou plusieurs services AWS d'analyse (Amazon EMR, blocs-notes AWS Glue Amazon SageMaker Unified Studio ou Amazon S3 Tables).
- Un compartiment S3 pour stocker les données et les métadonnées des tables.
- Un compartiment de tables pour commencer à utiliser Amazon S3 Tables ou un compartiment S3 à usage général si vous créez votre propre infrastructure Iceberg.
- Un AWS Glue catalogue configuré.

## Création de tables de version 3

### Création de tables

Pour créer une nouvelle table Iceberg version 3, définissez la propriété de la `format-version` table sur 3.

À l'aide de Spark SQL :

```
CREATE TABLE IF NOT EXISTS myns.orders_v3 (  
  order_id bigint,  
  customer_id string,  
  order_date date,  
  total_amount decimal(10,2),  
  status string,  
  created_at timestamp  
)  
USING iceberg  
TBLPROPERTIES (  
  'format-version' = '3'  
)
```

### Mise à niveau des tables de la version 2 vers la version 3

Vous pouvez mettre à niveau les tables existantes de la version 2 vers la version 3 de manière atomique sans réécrire les données.

À l'aide de Spark SQL :

```
ALTER TABLE myns.existing_table  
SET TBLPROPERTIES ('format-version' = '3')
```

#### Important

La version 3 est une mise à niveau unidirectionnelle. Une fois qu'une table est mise à niveau de la version 2 vers la version 3, elle ne peut pas être rétrogradée à la version 2 par le biais d'opérations standard.

Que se passe-t-il lors de la mise à niveau :

- Un nouvel instantané de métadonnées est créé de manière atomique.
- Les fichiers de données Parquet existants sont réutilisés.
- Les champs de lignage des lignes sont ajoutés aux métadonnées de la table.

Après la mise à niveau :

- Le prochain compactage supprimera les fichiers de suppression de l'ancienne version 2.
- Les nouvelles modifications utiliseront les fichiers vectoriels de suppression de la version 3.

La mise à niveau n'effectue pas de remblayage historique des enregistrements de suivi des modifications du lignage des lignes.

## Activation des vecteurs de suppression

Pour tirer parti des vecteurs de suppression pour les mises à jour, les suppressions et les fusions, configurez votre mode d'écriture.

À l'aide de Spark SQL :

```
ALTER TABLE myns.orders_v3
SET TBLPROPERTIES ('format-version' = '3',
                  'write.delete.mode' = 'merge-on-read',
                  'write.update.mode' = 'merge-on-read',
                  'write.merge.mode' = 'merge-on-read'
                  )
```

Ces paramètres garantissent que les opérations de mise à jour, de suppression et de fusion créent des fichiers vectoriels de suppression au lieu de réécrire des fichiers de données entiers.

## Utilisation du lignage des lignes pour le suivi des modifications

La version 3 ajoute automatiquement des champs de métadonnées de lignage des lignes pour suivre les modifications.

À l'aide de Spark SQL :

```
# Query with parameter value provided
last_processed_sequence = 47
```

```
SELECT
    id,
    data,
    _row_id,
    _last_updated_sequence_number
FROM myns.orders_v3
WHERE _last_updated_sequence_number > :last_processed_sequence
```

Le `_row_id` champ identifie de manière unique chaque ligne et indique quand `_last_updated_sequence_number` la ligne a été modifiée pour la dernière fois. Utilisez ces champs pour :

- Identifiez les lignes modifiées pour un traitement incrémentiel.
- Suivez le lignage des données à des fins de conformité.
- Optimisez les pipelines CDC.
- Réduisez les coûts de calcul en traitant uniquement les modifications.

## Bonnes pratiques pour la version 3

### Quand utiliser la version 3

Envisagez de passer à la version 3 ou de commencer par celle-ci lorsque :

- Vous effectuez fréquemment des mises à jour ou des suppressions par lots.
- Vous devez respecter le RGPD ou les exigences de conformité en matière de suppression.
- Vos charges de travail impliquent des perturbations à haute fréquence.
- Vous avez besoin de flux de travail CDC efficaces.
- Vous souhaitez réduire les coûts de stockage liés aux petits fichiers.
- Vous avez besoin de meilleures capacités de suivi des modifications.

### Optimisation des performances d'écriture

- Activez les vecteurs de suppression pour les charges de travail gourmandes en mises à jour :

```
SET TBLPROPERTIES (
    'write.delete.mode' = 'merge-on-read',
```

```
'write.update.mode' = 'merge-on-read',  
'write.merge.mode' = 'merge-on-read'  
)
```

- Configurez les tailles de fichier appropriées :

```
SET TBLPROPERTIES (  
'write.target-file-size-bytes' = '536870912' -- 512 MB  
)
```

## Optimisation des performances de lecture

- Utilisez le lignage des lignes pour le traitement incrémentiel.
- Utilisez le voyage dans le temps pour accéder aux données historiques sans les copier.
- Activez la collecte de statistiques pour une meilleure planification des requêtes.

## Stratégie de migration

Lorsque vous migrez de la version 2 vers la version 3, suivez les meilleures pratiques suivantes :

- Effectuez d'abord un test dans un environnement hors production pour valider le processus de mise à niveau et les performances.
- Effectuez une mise à niveau pendant les périodes de faible activité afin de minimiser l'impact sur les opérations simultanées.
- Surveillez les performances initiales et suivez les indicateurs après la mise à niveau.
- Exécutez le compactage pour consolider les fichiers supprimés après la mise à niveau.
- Mettez à jour la documentation de votre équipe pour refléter les fonctionnalités de la version 3.

## Considérations de compatibilité

- Versions du moteur : assurez-vous que tous les moteurs accédant à la table sont compatibles avec la version 3.
- Outils tiers : vérifiez la compatibilité de votre outil avec la version 3 avant de procéder à la mise à niveau.
- Stratégie de sauvegarde : testez les procédures de restauration basées sur des snapshots.

- Surveillance — Mettez à jour les tableaux de bord de surveillance pour les métriques spécifiques à la version 3.

## Résolution des problèmes

### Problèmes courants

Erreur : « le format version 3 n'est pas pris en charge »

- Vérifiez que la version de votre moteur est compatible avec la version 3. Pour plus de détails, consultez le [tableau](#) au début de cette section.
- Vérifiez la compatibilité du catalogue.
- Assurez-vous que vous utilisez les dernières versions de Services AWS.

Dégradation des performances après mise à niveau

- Vérifiez qu'il n'y a aucun défaut de compactage. Pour plus d'informations, consultez la section [Journalisation et surveillance des tables S3](#) dans la documentation Amazon S3.
- Vérifiez que les vecteurs de suppression sont activés. Les propriétés suivantes doivent être définies :

```
SET TBLPROPERTIES (  
  'write.delete.mode' = 'merge-on-read',  
  'write.update.mode' = 'merge-on-read',  
  'write.merge.mode' = 'merge-on-read'  
)
```

Vous pouvez vérifier les propriétés de la table à l'aide du code suivant :

```
DESCRIBE FORMATTED myns.orders_v3
```

- Passez en revue votre stratégie de partition. Le partitionnement excessif peut entraîner la création de fichiers de petite taille. Exécutez la requête suivante pour obtenir la taille de fichier moyenne de votre table :

```
SELECT avg(file_size_in_bytes) as avg_file_size_bytes  
FROM myns.orders_v3.files
```

## Incompatibilité avec des outils tiers

- Vérifiez que l'outil prend en charge la spécification de la version 3.
- Envisagez de conserver les tables de version 2 pour les outils non pris en charge.
- Contactez le fournisseur de l'outil pour connaître son calendrier de support pour la version 3.

## Obtenir de l'aide

- Pour Service AWS des problèmes spécifiques, contactez [AWS Support](#).
- Pour obtenir de l'aide auprès de la communauté Iceberg, utilisez le canal [Slack d'Iceberg](#).
- Pour plus d'informations sur l'utilisation Services AWS pour gérer vos charges de travail d'analyse, consultez [Analytics sur AWS](#).

## Tarification

- [Tarification du calcul et du stockage Amazon EMR](#)
- [SageMakerTarification Amazon](#)
- [AWS Glue exécution des tâches et tarification du catalogue de données](#)
- [Tarification du stockage et des demandes dans S3 Tables](#)

## Disponibilité

La prise en charge de la version 3 de la spécification du format de table Iceberg est disponible partout Régions AWS où Amazon EMR AWS Glue, AWS Glue Data Catalog, et S3 Tables fonctionnent. Pour connaître la disponibilité par région, voir [Services AWS par région](#).

## Ressources supplémentaires

- [Documentation d'Apache Iceberg](#)
- [Spécifications de la table Apache Iceberg](#)
- [Conseils pour la migration de données tabulaires d'Amazon S3 vers S3 Tables](#)
- [Tutoriel : Débuter avec S3 Tables](#)

# Migration de tables existantes vers Iceberg

Cette section se concentre sur la migration de vos Hive-style tables existantes vers le format Iceberg. Cela s'applique aux tables qui utilisent des Hive-compatible formats traditionnels tels que [Apache Parquet](#) ou [Apache ORC](#). Ces informations ne s'appliquent pas aux tables qui utilisent déjà des formats de table modernes tels que Linux Foundation Delta Lake ou Apache Hudi.

Pour migrer vos Hive-style tableaux actuels au format Iceberg, vous pouvez utiliser la migration des données sur place ou complète :

- In-place la [migration](#) est le processus qui consiste à générer les fichiers de métadonnées d'Iceberg au-dessus des fichiers de données existants.
- La [migration complète des données](#) crée la couche de métadonnées Iceberg et réécrit également les fichiers de données existants de la table d'origine vers la nouvelle table Iceberg.

Les sections suivantes fournissent un aperçu détaillé de chaque méthode de migration, y compris des instructions détaillées et des considérations relatives à la mise en œuvre. Pour plus d'informations sur ces stratégies de migration, consultez la section [Migration des tables](#) de la documentation d'Iceberg.

Après avoir examiné les détails des méthodes de migration des données en place et complètes, consultez les deux sections clés suivantes pour faciliter votre processus de prise de décision :

- [Le choix d'une stratégie de migration](#) fournit des conseils à travers une série de questions et de scénarios, afin de vous aider à déterminer l'approche de migration la plus appropriée en fonction de vos besoins spécifiques et de vos cas d'utilisation.
- Le [résumé des options de migration](#) fournit un tableau complet qui compare les principales caractéristiques et considérations des différentes options de migration. Ce tableau sert de guide de référence rapide et propose une comparaison des fonctionnalités pour vous aider à comprendre les compromis techniques entre les méthodes.

## In-place migration

In-place la migration élimine le besoin de réécrire tous vos fichiers de données. Au lieu de cela, les fichiers de métadonnées Iceberg sont générés et liés à vos fichiers de données existants. Cette

méthode est généralement plus rapide et plus rentable, en particulier pour les grands ensembles de données ou les tables dont les formats de fichier sont compatibles, tels que Parquet, Avro et ORC.

 Note

In-place la migration ne peut pas être utilisée lors de la migration vers [Amazon S3 Tables](#).

Iceberg propose deux options principales pour mettre en œuvre la migration sur place :

- Utilisation de la procédure de [capture instantanée](#) pour créer une nouvelle table Iceberg tout en conservant la table source inchangée. Pour plus d'informations, consultez [Snapshot Table](#) dans la documentation d'Iceberg.
- Utilisation de la procédure de [migration](#) pour créer une nouvelle table Iceberg en remplacement de la table source. Pour plus d'informations, consultez la section [Migrate Table](#) dans la documentation d'Iceberg. Bien que cette procédure fonctionne avec Hive Metastore (HMS), elle n'est actuellement pas compatible avec le. AWS Glue Data Catalog La AWS Glue Data Catalog section [Réplication de la procédure de migration des tables](#) présentée plus loin dans ce guide fournit une solution permettant d'obtenir un résultat similaire avec le catalogue de données.

Une fois que vous avez effectué la migration sur place en utilisant l'une snapshot ou l'autre des options `migrate`, certains fichiers de données risquent de ne pas être migrés. Cela se produit généralement lorsque les rédacteurs continuent d'écrire dans la table source pendant ou après la migration. Pour intégrer ces fichiers restants dans votre table Iceberg, vous pouvez utiliser la procédure [add\\_files](#). Pour plus d'informations, consultez la section [Ajouter des fichiers](#) dans la documentation d'Iceberg.

Supposons que vous ayez une Parquet-based products table créée et remplie dans Athena comme suit :

```
CREATE EXTERNAL TABLE mydb.products (  
    product_id INT,  
    product_name STRING  
)  
PARTITIONED BY (category STRING)  
STORED AS PARQUET  
LOCATION 's3://amzn-s3-demo-bucket/products/';  
  
INSERT INTO mydb.products
```

## VALUES

```
(1001, 'Smartphone', 'electronics'),  
(1002, 'Laptop', 'electronics'),  
(2001, 'T-Shirt', 'clothing'),  
(2002, 'Jeans', 'clothing');
```

Les sections suivantes expliquent comment utiliser les migrate procédures snapshot et avec ce tableau.

## Option 1 : procédure de capture instantanée

La snapshot procédure crée une nouvelle table Iceberg portant un nom différent mais qui reproduit le schéma et le partitionnement de la table source. Cette opération laisse la table source complètement inchangée pendant et après l'action. Il crée efficacement une copie allégée de la table, ce qui est particulièrement utile pour tester des scénarios ou explorer des données sans risquer de modifier la source de données d'origine. Cette approche permet une période de transition pendant laquelle le tableau original et le tableau Iceberg restent disponibles (voir les notes à la fin de cette section). Une fois les tests terminés, vous pouvez mettre votre nouvelle table Iceberg en production en transférant tous les rédacteurs et lecteurs vers la nouvelle table.

Vous pouvez exécuter la snapshot procédure en utilisant Spark dans n'importe quel modèle de déploiement Amazon EMR (par exemple, Amazon EMR sur EC2, Amazon EMR sur EKS, EMR Serverless) et. AWS Glue

Pour tester la migration sur place à l'aide de la procédure snapshot Spark, procédez comme suit :

1. Lancez une application Spark et configurez la session Spark avec les paramètres suivants :

- "spark.sql.extensions":"org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions"
- "spark.sql.catalog.spark\_catalog":"org.apache.iceberg.spark.SparkSessionCatalog"
- "spark.sql.catalog.spark\_catalog.type":"glue"
- "spark.hadoop.hive.metastore.client.factory.class":"com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClient"

2. Exécutez la snapshot procédure pour créer une nouvelle table Iceberg pointant vers les fichiers de données de la table d'origine :

```
spark.sql(f"""  
CALL system.snapshot(  
  source_table => 'mydb.products',  
  table => 'mydb.products_iceberg',  
  location => 's3://amzn-s3-demo-bucket/products_iceberg/'  
)
```

```
)  
""""  
) .show(truncate=False)
```

La trame de données de sortie contient le `imported_files_count` (le nombre de fichiers ajoutés).

### 3. Validez la nouvelle table en l'interrogeant :

```
spark.sql(f"""  
SELECT * FROM mydb.products_iceberg LIMIT 10  
""")  
) .show(truncate=False)
```

#### Remarques

- Après avoir exécuté la procédure, toute modification du fichier de données sur la table source désynchronisera la table générée. Les nouveaux fichiers que vous ajoutez ne seront pas visibles dans la table Iceberg, et les fichiers que vous avez supprimés affecteront les capacités de requête de la table Iceberg. Pour éviter les problèmes de synchronisation :
- Si la nouvelle table Iceberg est destinée à une utilisation en production, arrêtez tous les processus qui écrivent dans la table d'origine et redirigez-les vers la nouvelle table.
- Si vous avez besoin d'une période de transition ou si la nouvelle table Iceberg est destinée à des fins de test, consultez la section [Maintenance de la synchronisation des tables Iceberg après une migration sur place](#) plus loin dans cette section pour obtenir des conseils sur la gestion de la synchronisation des tables.
- Lorsque vous utilisez snapshot cette procédure, la `gc.enabled` propriété est définie `false` dans les propriétés de la table Iceberg créée. Ce paramètre interdit les actions telles que `expire_snapshotsremove_orphan_files`, ou `DROP TABLE` avec l'`PURGEOption`, qui supprimeraient physiquement des fichiers de données. Les opérations de suppression ou de fusion d'Iceberg, qui n'ont pas d'impact direct sur les fichiers source, sont toujours autorisées.
- Pour que votre nouvelle table Iceberg soit pleinement fonctionnelle, sans aucune limite quant aux actions de suppression physique de fichiers de données, vous pouvez modifier la propriété de la `gc.enabled` table sur `true`. Toutefois, ce paramètre autorise les actions

qui ont un impact sur les fichiers de données source, ce qui pourrait altérer l'accès à la table d'origine. Par conséquent, modifiez la `gc.enabled` propriété uniquement si vous n'avez plus besoin de conserver les fonctionnalités de la table d'origine. Par exemple :

```
spark.sql(f"""
ALTER TABLE mydb.products_iceberg
SET TBLPROPERTIES ('gc.enabled' = 'true');
""")
```

## Option 2 : procédure de migration

La `migrate` procédure crée une nouvelle table Iceberg portant le même nom, le même schéma et le même partitionnement que la table source. Lorsque cette procédure s'exécute, elle verrouille la table source et la renomme en `<table_name>_BACKUP_` (ou en un nom personnalisé spécifié par le paramètre de `backup_table_name` procédure).

### Note

Si vous définissez le paramètre de `drop_backup` procédure sur `true`, la table d'origine ne sera pas conservée en tant que sauvegarde.

Par conséquent, la procédure de `migrate` table exige que toutes les modifications affectant la table source soient arrêtées avant que l'action ne soit exécutée. Avant d'exécuter la `migrate` procédure :

- Arrêtez tous les rédacteurs qui interagissent avec la table source.
- Modifiez les lecteurs et les rédacteurs qui ne supportent pas Iceberg de manière native pour activer le support Iceberg.

Par exemple :

- Athéna continue de travailler sans modification.
- Spark nécessite :
  - Fichiers Iceberg Java Archive (JAR) à inclure dans le chemin de classe (voir les sections [Travailler avec Iceberg dans Amazon EMR](#) et [Travailler avec Iceberg AWS Glue](#) dans les sections précédentes de ce guide).

- Les configurations de catalogue de session Spark suivantes (SparkSessionCatalog à utiliser pour ajouter la prise en charge d'Iceberg tout en conservant les fonctionnalités de catalogue intégrées pour les tables autres qu'Iceberg) :
  - "spark.sql.extensions":"org.apache.iceberg.spark.extensions.IcebergSparkSes
  - "spark.sql.catalog.spark\_catalog":"org.apache.iceberg.spark.SparkSessionCat
  - "spark.sql.catalog.spark\_catalog.type":"glue"
  - "spark.hadoop.hive.metastore.client.factory.class":"com.amazonaws.glue.cata

Après avoir exécuté la procédure, vous pouvez redémarrer vos rédacteurs avec leur nouvelle configuration Iceberg.

Actuellement, la migrate procédure n'est pas compatible avec le AWS Glue Data Catalog, car le catalogue de données ne prend pas en charge l'RENAME opération. Par conséquent, nous vous recommandons d'utiliser cette procédure uniquement lorsque vous travaillez avec Hive Metastore. Si vous utilisez le catalogue de données, consultez la [section suivante](#) pour une autre approche.

Vous pouvez exécuter la migrate procédure sur tous les modèles de déploiement Amazon EMR (Amazon EMR sur EC2, Amazon EMR sur EKS, EMR sans serveur), mais elle nécessite une connexion configurée à Hive Metastore AWS Glue. Amazon EMR sur EC2 est le choix recommandé car il fournit une configuration Hive Metastore intégrée, ce qui minimise la complexité de l'installation.

Pour tester la migration sur place à l'aide de la procédure migrate Spark à partir d'un cluster Amazon EMR sur EC2 configuré avec Hive Metastore, procédez comme suit :

1. Lancez une application Spark et configurez la session Spark pour utiliser l'implémentation du catalogue Iceberg Hive. Par exemple, si vous utilisez la pyspark CLI :

```
pyspark --conf
spark.sql.extensions=org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions
--conf spark.sql.catalog.spark_catalog=org.apache.iceberg.spark.SparkSessionCatalog
--conf spark.sql.catalog.spark_catalog.type=hive
```

2. Créez une products table dans Hive Metastore. Il s'agit de la table source, qui existe déjà dans une migration classique.
  - a. Créez la table Hive products externe dans Hive Metastore pour pointer vers les données existantes dans Amazon S3 :

```
spark.sql(f"""
```

```
CREATE EXTERNAL TABLE products (
  product_id INT,
  product_name STRING
)
PARTITIONED BY (category STRING)
STORED AS PARQUET
LOCATION 's3://amzn-s3-demo-bucket/products/';
""
)
```

b. Ajoutez les partitions existantes à l'aide de la `MSCK REPAIR TABLE` commande :

```
spark.sql(f"""
MSCK REPAIR TABLE products
""")
)
```

c. Vérifiez que la table contient des données en exécutant une `SELECT` requête :

```
spark.sql(f"""
SELECT * FROM products
""")
).show(truncate=False)
```

Exemple de sortie :

```
>>> spark.sql(f"""
... SELECT * FROM products
... """)
... ).show(truncate=False)
+-----+-----+-----+
|product_id|product_name|category|
+-----+-----+-----+
|1001      |Smartphone  |electronics|
|1002      |Laptop      |electronics|
|2001      |T-Shirt     |clothing   |
|2002      |Jeans       |clothing   |
+-----+-----+-----+
```

3. Utilisez la migrate procédure Iceberg :

```
df_res=spark.sql(f"""
CALL system.migrate(
table => 'default.products'
```

```
)
""
)

df_res.show()
```

La trame de données de sortie contient `migrated_files_count` (le nombre de fichiers ajoutés à la table Iceberg) :

```
>>> df_res.show()
+-----+
|migrated_files_count|
+-----+
|                    2|
+-----+
```

4. Vérifiez que la table de sauvegarde a été créée :

```
spark.sql("show tables").show()
```

Exemple de sortie :

```
>>> spark.sql("show tables").show()
+-----+-----+-----+
|namespace|tableName|isTemporary|
+-----+-----+-----+
| default|products|false|
| default|products_backup_|false|
+-----+-----+-----+
```

5. Validez l'opération en interrogeant la table Iceberg :

```
spark.sql(f"""
SELECT * FROM products
""")
).show(truncate=False)
```

### Remarques

- Une fois la procédure exécutée, tous les processus actuels qui interrogent ou écrivent dans la table source seront affectés s'ils ne sont pas correctement configurés avec le support d'Iceberg. Nous vous recommandons donc de suivre les étapes suivantes :
  1. Arrêtez tous les processus en utilisant la table source avant la migration.
  2. Effectuez la migration.
  3. Réactivez les processus en utilisant les paramètres Iceberg appropriés.
- Si des modifications de fichiers de données se produisent pendant le processus de migration (de nouveaux fichiers sont ajoutés ou des fichiers sont supprimés), la table générée sera désynchronisée. Pour les options de synchronisation, voir [Maintien de la synchronisation des tables Iceberg après une migration sur place](#) plus loin dans cette section.

## Réplication de la procédure de migration des tables dans AWS Glue Data Catalog

Vous pouvez reproduire le résultat de la procédure de migration dans AWS Glue Data Catalog (en sauvegardant la table d'origine et en la remplaçant par une table Iceberg) en procédant comme suit :

1. Utilisez la procédure de capture instantanée pour créer une nouvelle table Iceberg pointant vers les fichiers de données de la table d'origine.
2. Sauvegardez les métadonnées de la table d'origine dans le catalogue de données :
  - a. Utilisez l'[GetTable](#)API pour récupérer la définition de la table source.
  - b. Utilisez l'[GetPartitions](#)API pour récupérer la définition de la partition de table source.
  - c. Utilisez l'[CreateTable](#)API pour créer une table de sauvegarde dans le catalogue de données.
  - d. Utilisez l'[BatchCreatePartition](#)API [CreatePartition](#) pour enregistrer des partitions dans la table de sauvegarde du catalogue de données.
3. Modifiez la propriété de la table `gc.enabled Iceberg false` pour activer les opérations complètes de la table.
4. Supprimez la table d'origine.
5. Localisez le fichier JSON des métadonnées de la table Iceberg dans le dossier de métadonnées situé à la racine de la table.

6. Enregistrez la nouvelle table dans le catalogue de données en utilisant la procédure [register\\_table](#) avec le nom de table d'origine et l'emplacement du metadata .json fichier créé par la procédure : snapshot

```
spark.sql(f"""
CALL system.register_table(
    table => 'mydb.products',
    metadata_file => '{iceberg_metadata_file}'
)
""")
).show(truncate=False)
```

## Synchronisation des tables Iceberg après une migration sur place

La `add_files` procédure fournit un moyen flexible d'intégrer les données existantes dans les tables Iceberg. Plus précisément, il enregistre les fichiers de données existants (tels que les fichiers Parquet) en référençant leurs chemins absolus dans la couche de métadonnées d'Iceberg. Par défaut, la procédure ajoute des fichiers provenant de toutes les partitions de table à une table Iceberg, mais vous pouvez ajouter des fichiers provenant de partitions spécifiques de manière sélective. Cette approche sélective est particulièrement utile dans plusieurs scénarios :

- Lorsque de nouvelles partitions sont ajoutées à la table source après la migration initiale.
- Lorsque des fichiers de données sont ajoutés ou supprimés de partitions existantes après la migration initiale. Toutefois, pour ajouter à nouveau des partitions modifiées, il faut d'abord les supprimer. De plus amples informations à ce sujet sont fournies plus loin dans cette section.

Voici quelques considérations relatives à l'utilisation de la `add_file` procédure une fois la migration sur place (`snapshotumigrate`) effectuée, afin de maintenir la synchronisation de la nouvelle table Iceberg avec les fichiers de données sources :

- Lorsque de nouvelles données sont ajoutées à de nouvelles partitions de la table source, utilisez la `add_files` procédure avec la `partition_filter` possibilité d'intégrer de manière sélective ces ajouts dans la table Iceberg :

```
spark.sql(f"""
CALL system.add_files(
    source_table => 'mydb.products',
    table => 'mydb.products_iceberg',
```

```
partition_filter => map('category', 'electronics')
).show(truncate=False)
```

ou :

```
spark.sql(f"""
CALL system.add_files(
source_table => '`parquet`.`s3://amzn-s3-demo-bucket/products/`',
table => 'mydb.products_iceberg',
partition_filter => map('category', 'electronics')
).show(truncate=False)
```

- La `add_files` procédure recherche les fichiers dans l'ensemble de la table source ou dans des partitions spécifiques lorsque vous spécifiez l'`partition_filter` option, et tente d'ajouter tous les fichiers trouvés dans la table Iceberg. Par défaut, la propriété de `check_duplicate_files` procédure est définie sur `true`, ce qui empêche l'exécution de la procédure si des fichiers existent déjà dans la table Iceberg. Ceci est important car il n'existe aucune option intégrée permettant d'ignorer les fichiers ajoutés précédemment, et la désactivation `check_duplicate_files` entraînera l'ajout de deux fichiers, ce qui créera des doublons. Lorsque de nouveaux fichiers sont ajoutés à la table source, procédez comme suit :
1. Pour les nouvelles partitions, utilisez `add_files` with a `partition_filter` pour importer uniquement les fichiers de la nouvelle partition.
  2. Pour les partitions existantes, supprimez d'abord la partition de la table Iceberg, puis exécutez-la `add_files` à nouveau, en spécifiant le `partition_filter` Par exemple :

```
# We initially perform in-place migration with snapshot
spark.sql(f"""
CALL system.snapshot(
source_table => 'mydb.products',
table => 'mydb.products_iceberg',
location => 's3://amzn-s3-demo-bucket/products_iceberg/'
)
""")
).show(truncate=False)

# Then on the source table, some new files were generated under the
category='electronics' partition. Example:
spark.sql("""
INSERT INTO mydb.products
VALUES (1003, 'Tablet', 'electronics')
```

```

"""
# We delete the modified partition from the Iceberg table. Note this is a metadata
  operation only
spark.sql("""
DELETE FROM mydb.products_iceberg WHERE category = 'electronics'
""")

# We add_files from the modified partition
spark.sql("""
CALL system.add_files(
  source_table => 'mydb.products',
  table => 'mydb.products_iceberg',
  partition_filter => map('category', 'electronics')
)
""").show(truncate=False)

```

### Note

Chaque `add_files` opération génère un nouvel instantané de la table Iceberg avec des données ajoutées.

## Choisir la bonne stratégie de migration sur place

Pour choisir la meilleure stratégie de migration sur place, prenez en compte les questions du tableau suivant.

| Question                                                                                                                                                         | Recommandation                                                  | Explication                                                                                                                                                                                                                                              |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Voulez-vous migrer rapidement sans réécrire les données tout en gardant les tables Hive et Iceberg accessibles à des fins de test ou de transition progressive ? | snapshotprocédure suivie d'une <code>add_files</code> procédure | Utilisez snapshot cette procédure pour créer une nouvelle table Iceberg en clonant le schéma et en référençant les fichiers de données, sans modifier la table source. Utilisez la <code>add_files</code> procédure pour intégrer des partitions qui ont |

| Question                                                                                                                                   | Recommandation                                                               | Explication                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|--------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                                                                                                            |                                                                              | été ajoutées ou modifiées après la migration, en notant que le réajout de partitions modifiées nécessite d'abord la suppression des partitions.                                                                                                                                                                                                                                                                                                                                                                       |
| Utilisez-vous Hive Metastore et souhaitez-vous remplacer immédiatement votre table Hive par une table Iceberg, sans réécrire les données ? | <code>migrate</code> procédure suivie d'une <code>add_files</code> procédure | <p>Utilisez <code>migrate</code> cette procédure pour créer une table Iceberg, sauvegarder la table source et remplacer la table d'origine par la version Iceberg.</p> <p>Remarque : Cette option est compatible avec Hive Metastore mais pas avec AWS Glue Data Catalog</p> <p>Utilisez la <code>add_files</code> procédure pour intégrer des partitions qui ont été ajoutées ou modifiées après la migration, en notant que le réajout de partitions modifiées nécessite d'abord la suppression des partitions.</p> |

| Question                                                                                                                                          | Recommandation                                                                                 | Explication                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|---------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Utilisez-vous AWS Glue Data Catalog et souhaitez-vous remplacer immédiatement votre table Hive par une table Iceberg, sans réécrire les données ? | Adaptation de la <code>migrate</code> procédure, suivie de <code>add_files</code> la procédure | <p>Comportement de <code>migrate</code> procédure de réplication :</p> <ol style="list-style-type: none"><li>1. <code>snapshot</code> À utiliser pour créer une table Iceberg.</li><li>2. Sauvegardez les métadonnées de la table d'origine à l'aide d' AWS Glue API.</li><li>3. Activez <code>gc.enabled</code> les propriétés de la table Iceberg.</li><li>4. Supprimez la table d'origine .</li><li>5. <code>register_table</code> À utiliser pour créer une nouvelle entrée de table avec le nom d'origine.</li></ol> <p>Remarque : Cette option nécessite la gestion manuelle des appels d' AWS Glue API pour la sauvegarde des métadonnées.</p> <p>Utilisez la <code>add_files</code> procédure pour intégrer des partitions qui ont été ajoutées ou modifiées après la migration, en notant que le réajout de partitions modifiées nécessite d'abord la suppression des partitions.</p> |

# Migration complète des données

La migration complète des données recrée les fichiers de données ainsi que les métadonnées. Cette approche prend plus de temps et nécessite des ressources informatiques supplémentaires par rapport à la migration sur place. Cependant, la migration complète des données offre d'importantes opportunités pour améliorer la qualité des tables et optimiser le stockage des données et les modèles d'accès.

Au cours de la migration complète des données, vous pouvez effectuer plusieurs opérations bénéfiques, telles que la validation des données pour garantir leur intégrité et leur exactitude, les modifications du schéma pour mieux répondre aux exigences actuelles et les ajustements de la stratégie de partition pour améliorer les performances des requêtes. Vous pouvez également trier à nouveau les données pour optimiser les modèles d'accès courants, implémenter le partitionnement masqué Iceberg pour améliorer l'efficacité des requêtes et effectuer une conversion de format de fichier (par exemple, de CSV à Parquet) si vous le souhaitez.

Ces fonctionnalités font de la migration complète des données la solution idéale pour passer au format Iceberg et pour affiner et optimiser de manière globale votre stratégie de stockage de données. Bien que la migration complète des données nécessite plus de temps et de ressources dès le départ, les améliorations qui en résultent en termes de qualité des données, d'organisation et de performance des requêtes peuvent apporter des avantages à long terme. Pour implémenter la migration complète des données, utilisez l'une des options suivantes :

- Utilisez l'instruction `CREATE TABLE ... AS SELECT` ([CTAS](#)) dans Spark (sur Amazon EMR AWS Glue ou) ou dans Athena. Vous pouvez définir la spécification de partition et les propriétés de table pour la nouvelle table Iceberg à l'aide des `TBLPROPERTIES` clauses `PARTITIONED BY` et. Vous pouvez modifier le schéma et le partitionnement de la nouvelle table en fonction de vos besoins au lieu d'hériter de la table source.
- Lisez la table source et écrivez les données sous forme de nouvelle table Iceberg à l'aide de Spark sur Amazon AWS Glue EMR ou. Pour plus d'informations, consultez la section [Création d'une table](#) dans la documentation d'Iceberg.

## Choix d'une stratégie de migration

Lors de la transition vers le format Iceberg, le choix entre une migration sur place et une migration complète est crucial. Pour déterminer l'approche la mieux adaptée à vos besoins spécifiques, prenez en compte les questions et recommandations suivantes :

| Question                                                                                      | Recommandation                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|-----------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Quel est le format du fichier de données (par exemple, CSV ou Apache Parquet) ?               | <ul style="list-style-type: none"> <li>• Envisagez une migration sur place si le format de votre fichier de table est Parquet, ORC ou Avro.</li> <li>• Pour les autres formats tels que CSV, JSON, etc., utilisez la migration complète des données.</li> </ul>                                                                                                                                                                                                                                                                                                |
| Voulez-vous mettre à jour ou consolider le schéma de table ?                                  | <ul style="list-style-type: none"> <li>• Si vous souhaitez faire évoluer le schéma de table à l'aide des fonctionnalités natives d'Iceberg, envisagez une migration sur place. Par exemple, vous pouvez renommer les colonnes après la migration. (Le schéma peut être modifié dans la couche de métadonnées Iceberg.)</li> <li>• Si vous souhaitez supprimer des colonnes entières parce qu'elles ne sont plus nécessaires, nous vous recommandons d'utiliser la migration complète des données.</li> </ul>                                                   |
| La table bénéficierait-elle d'une modification de la stratégie de partition ?                 | <ul style="list-style-type: none"> <li>• Si l'approche de partitionnement d'Iceberg répond à vos exigences (par exemple, les nouvelles données sont stockées en utilisant le nouveau schéma de partition alors que les partitions existantes restent telles quelles), envisagez une migration sur place.</li> <li>• Si vous souhaitez utiliser des partitions masquées dans votre table, envisagez une migration complète des données. Pour plus d'informations sur les partitions masquées, consultez la section <a href="#">Bonnes pratiques</a>.</li> </ul> |
| Le tableau bénéficierait-il de l'ajout ou de la modification de la stratégie d'ordre de tri ? | <ul style="list-style-type: none"> <li>• L'ajout ou la modification de l'ordre de tri de vos données nécessite de réécrire le jeu de données. Dans ce cas, envisagez d'utiliser la migration complète des données.</li> <li>• Pour les grandes tables où le coût de réécriture de toutes les partitions de table est prohibitif, envisagez d'utiliser la migration sur place et d'exécuter le compactage (avec le tri activé) pour les partitions les plus fréquemment consultées.</li> </ul>                                                                  |
| Le tableau contient-il de nombreux petits fichiers ?                                          | <ul style="list-style-type: none"> <li>• La fusion de petits fichiers en fichiers plus volumineux nécessite de réécrire le jeu de données. Dans ce cas, envisagez d'utiliser la migration complète des données.</li> </ul>                                                                                                                                                                                                                                                                                                                                     |

| Question | Recommandation                                                                                                                                                                                                                                                                                         |
|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|          | <ul style="list-style-type: none"><li>• Pour les grandes tables où le coût de réécriture de toutes les partitions de table est prohibitif, envisagez d'utiliser la migration sur place et d'exécuter le compactage (avec le tri activé) pour les partitions les plus fréquemment consultées.</li></ul> |

## Résumé des options de migration

Ce tableau récapitule les principales caractéristiques et considérations relatives à chaque option de migration.

| Fonctionnalité                                                                       | In-place migration<br><a href="#">instantané</a> | In-place migration<br><a href="#">migrate</a> | Migration complète des données<br><a href="#">CTAS</a> ou <a href="#">(CRÉER UN TABLEAU + INSÉRER)</a> |
|--------------------------------------------------------------------------------------|--------------------------------------------------|-----------------------------------------------|--------------------------------------------------------------------------------------------------------|
| Améliorations de la mise en page des données dans le cadre du processus de migration |                                                  |                                               |                                                                                                        |
| • Re-sort données                                                                    | ⊗Non                                             | ⊗Non                                          | ⊙Oui                                                                                                   |

| Fonctionnalité                                                                               | In-place migration<br><u>instantané</u> | In-place migration<br><u>migrate</u> | Migration complète des données<br><u>CTAS ou (CRÉER UN TABLEAU + INSÉRER)</u> |
|----------------------------------------------------------------------------------------------|-----------------------------------------|--------------------------------------|-------------------------------------------------------------------------------|
| • Modifier le partitionnement (par exemple pour utiliser le partitionnement caché d'Iceberg) | ⊗ Non                                   | ⊗ Non                                | ☑ Oui                                                                         |
| • Modifier le schéma de table                                                                | ⊗ Non                                   | ⊗ Non                                | ☑ Oui                                                                         |
| • Optimisation de la taille du fichier                                                       | ⊗ Non                                   | ⊗ Non                                | ☑ Oui                                                                         |

| Fonctionnalité                                                                                                           | In-place migration<br><u>instantané</u>                                                                        | In-place migration<br><u>migrate</u>                                                     | Migration complète des données<br><u>CTAS ou (CRÉER UN TABLEAU + INSÉRER)</u> |
|--------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------|
| <ul style="list-style-type: none"> <li>• Valider le schéma des données existantes avant d'ajouter les données</li> </ul> | ⊗Non                                                                                                           | ⊗Non                                                                                     | ✔Oui                                                                          |
| Formats de fichiers pris en charge                                                                                       | Parquet, Avro, ORC                                                                                             | Parquet, Avro, ORC                                                                       | Parquet, Avro, ORC, JSON, CSV                                                 |
| Remplacement de la table source par une table Iceberg                                                                    | ⊗Non<br>(crée une nouvelle table, mais avec des étapes supplémentaires, vous pouvez remplacer la table source) | ✔Oui<br>(crée une table de sauvegarde et remplace la table source par une table Iceberg) | ⊗Non<br>(crée une nouvelle table)                                             |
| Impact sur le tableau source                                                                                             |                                                                                                                |                                                                                          |                                                                               |

| Fonctionnalité                                                                                                          | In-place migration<br><u>instantané</u> | In-place migration<br><u>migrate</u> | Migration complète des données<br><u>CTAS ou (CRÉER UN TABLEAU + INSÉRER)</u> |
|-------------------------------------------------------------------------------------------------------------------------|-----------------------------------------|--------------------------------------|-------------------------------------------------------------------------------|
| • Opérations de suppression de fichiers sur la table Iceberg (expiapshotopérations, suppression d'une table avec purge) | Corrige la table source                 | Corrige la table de sauvegarde       | Sûr, source non affectée                                                      |
| Impact de la table Iceberg                                                                                              |                                         |                                      |                                                                               |

| Fonctionnalité                                                                                                                   | In-place migration<br><u>instantané</u>                                                              | In-place migration<br><u>migrate</u>                                                                 | Migration complète des données<br><u>CTAS ou (CRÉER UN TABLEAU + INSÉRER)</u>                |
|----------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"> <li>Impact si les fichiers de la table source sont supprimés</li> </ul>                       | Corruption de la table Iceberg                                                                       | Corruption de la table Iceberg                                                                       | Aucun impact sur la table Iceberg                                                            |
| <ul style="list-style-type: none"> <li>Impact si de nouveaux fichiers sont ajoutés à l'emplacement de la table source</li> </ul> | Non visible sur le nouveau tableau<br>(besoin d'intégrer une partition avec <code>add_files</code> ) | Non visible sur le nouveau tableau<br>(besoin d'intégrer une partition avec <code>add_files</code> ) | Non visible sur le nouveau tableau<br>(besoin de <code>INSERT INTO</code> la nouvelle table) |
| Coût                                                                                                                             | Faible                                                                                               | Faible                                                                                               | Plus élevé (réécriture complète des données)                                                 |
| Vitesse de migration                                                                                                             | Rapide                                                                                               | Rapide                                                                                               | Plus lent                                                                                    |

| Fonctionnalité                                      | In-place migration<br><u>instantané</u>                                                                                                             | In-place migration<br><u>migrate</u>                                                                       | Migration complète des données<br><u>CTAS ou (CRÉER UN TABLEAU + INSÉRER)</u>                                                                                                                             |
|-----------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Peut être utilisé pour migrer vers Amazon S3 Tables | ⊗Non                                                                                                                                                | ⊗Non                                                                                                       | ☑Oui                                                                                                                                                                                                      |
| Nécessite un DDL manuel                             | ⊗Non<br>(le schéma et les partitions sont copiés depuis la table source)                                                                            | ⊗Non<br>(le schéma et les partitions sont copiés depuis la table source)                                   | Si vous utilisez le CTAS, il suffit de spécifier le partitionnement                                                                                                                                       |
| Meilleure utilisation                               | Migration rapide sans réécriture des données, ce qui permet d'utiliser Hive et Iceberg côte à côte à des fins de test ou de transition progressive. | Remplacer une table Hive en place sans réécrire les données, lorsqu'un changement immédiat est acceptable. | Optimisation complète d'Iceberg avec réécriture des données. Idéal pour redessiner des partitions ou des schémas, ou pour améliorer la mise en page et les performances. Toujours recommandé si possible. |

# Bonnes pratiques pour optimiser les charges de travail d'Apache Iceberg

Iceberg est un format de tableau conçu pour simplifier la gestion des lacs de données et améliorer les performances des charges de travail. Différents cas d'utilisation peuvent donner la priorité à différents aspects tels que le coût, les performances de lecture, les performances d'écriture ou la conservation des données. Iceberg propose donc des options de configuration pour gérer ces compromis. Cette section fournit des informations permettant d'optimiser et de peaufiner vos charges de travail Iceberg afin de répondre à vos besoins.

## Rubriques

- [Bonnes pratiques d'ordre général](#)
- [Optimisation des performances de lecture](#)
- [Optimisation des performances d'écriture](#)
- [Optimisation du stockage](#)
- [Maintenance des tables en utilisant le compactage](#)
- [Utilisation des charges de travail Iceberg dans Amazon S3](#)

## Bonnes pratiques d'ordre général

Quel que soit votre cas d'utilisation, lorsque vous utilisez Apache Iceberg sur Apache Iceberg AWS, nous vous recommandons de suivre ces bonnes pratiques générales.

- Utilisez le format Iceberg version 2.

Athena utilise le format Iceberg version 2 par défaut.

[Lorsque vous utilisez Spark sur Amazon EMR ou AWS Glue pour créer des tables Iceberg, spécifiez la version du format comme décrit dans la documentation d'Iceberg.](#)

- Utilisez-le AWS Glue Data Catalog comme catalogue de données.

Athéna utilise le AWS Glue Data Catalog par défaut.

Lorsque vous utilisez Spark sur Amazon EMR ou AWS Glue que vous travaillez avec Iceberg, ajoutez la configuration suivante à votre session Spark pour utiliser le. AWS Glue Data Catalog

Pour plus d'informations, consultez la section [Configurations de Spark pour Iceberg AWS Glue plus haut dans](#) ce guide.

```
"spark.sql.catalog.<your_catalog_name>.type": "glue"
```

- Utilisez le gestionnaire AWS Glue Data Catalog de verrous.

Athena utilise le gestionnaire de AWS Glue Data Catalog verrous par défaut pour les tables Iceberg.

Lorsque vous utilisez Spark sur Amazon EMR ou AWS Glue que vous travaillez avec Iceberg, assurez-vous de configurer la configuration de votre session Spark de manière à utiliser le AWS Glue Data Catalog gestionnaire de verrouillage. Pour plus d'informations, consultez [Optimistic Locking](#) dans la documentation d'Iceberg.

- Utilisez la compression Zstandard (ZSTD).

Le codec de compression par défaut d'Iceberg est gzip, qui peut être modifié à l'aide de la propriété `table.write.<file_type>.compression-codec`. Athena utilise déjà ZSTD comme codec de compression par défaut pour les tables Iceberg.

En général, nous recommandons d'utiliser le codec de compression ZSTD, car il établit un équilibre entre GZIP et Snappy et offre de bonnes read/write performances sans compromettre le taux de compression. De plus, les niveaux de compression peuvent être ajustés en fonction de vos besoins. Pour plus d'informations, consultez la section [Niveaux de compression ZSTD dans Athena dans la documentation d'Athena](#).

Snappy offre peut-être les meilleures performances globales de lecture et d'écriture, mais son taux de compression est inférieur à celui de GZIP et ZSTD. Si vous privilégiez les performances, même si cela implique de stocker des volumes de données plus importants dans Amazon S3, Snappy peut être le meilleur choix.

## Optimisation des performances de lecture

Cette section décrit les propriétés des tables que vous pouvez régler pour optimiser les performances de lecture, indépendamment du moteur.

## Partitioning

Comme pour les tables Hive, Iceberg utilise les partitions comme couche d'indexation principale afin d'éviter de lire des fichiers de métadonnées et des fichiers de données inutiles. Les statistiques des colonnes sont également prises en compte en tant que couche secondaire d'indexation afin d'améliorer encore la planification des requêtes, ce qui se traduit par un meilleur temps d'exécution global.

### Partitionner vos données

Pour réduire la quantité de données scannées lors de l'interrogation des tables Iceberg, choisissez une stratégie de partition équilibrée qui correspond aux modèles de lecture attendus :

- Identifiez les colonnes fréquemment utilisées dans les requêtes. Ce sont des candidats idéaux pour le partitionnement. Par exemple, si vous recherchez généralement des données relatives à un jour donné, un exemple naturel de colonne de partition serait une colonne de date.
- Choisissez une colonne de partition à faible cardinalité pour éviter de créer un nombre excessif de partitions. Un trop grand nombre de partitions peut augmenter le nombre de fichiers dans la table, ce qui peut avoir un impact négatif sur les performances des requêtes. En règle générale, « trop de partitions » peut être défini comme un scénario dans lequel la taille des données dans la majorité des partitions est inférieure à 2 à 5 fois la valeur définie par `target-file-size-bytes`.

#### Note

Si vous effectuez généralement une requête à l'aide de filtres sur une colonne à cardinalité élevée (par exemple, une `id` colonne pouvant contenir des milliers de valeurs), utilisez la fonction de partitionnement masquée d'Iceberg avec les transformations de compartiments, comme expliqué dans la section suivante.

### Utiliser le partitionnement masqué

Si vos requêtes filtrent généralement sur un dérivé d'une colonne de table, utilisez des partitions masquées au lieu de créer explicitement de nouvelles colonnes qui fonctionneront comme des partitions. Pour plus d'informations sur cette fonctionnalité, consultez la [documentation d'Iceberg](#).

Par exemple, dans un ensemble de données comportant une colonne d'horodatage (par exemple, `2023-01-01 09:00:00`), au lieu de créer une nouvelle colonne avec la date analysée

(par exemple, 2023-01-01), utilisez des transformations de partition pour extraire la partie date de l'horodatage et créer ces partitions à la volée.

Les cas d'utilisation les plus courants du partitionnement masqué sont les suivants :

- Partitionnement à la date ou à l'heure, lorsque les données comportent une colonne d'horodatage. Iceberg propose plusieurs transformations pour extraire la date ou l'heure d'un horodatage.
- Partitionnement sur une fonction de hachage d'une colonne, lorsque la colonne de partitionnement a une cardinalité élevée et entraînerait un trop grand nombre de partitions. La transformation des compartiments d'Iceberg regroupe plusieurs valeurs de partition en un nombre réduit de partitions cachées (compartiments) en utilisant des fonctions de hachage sur la colonne de partitionnement.

Voir les [transformations de partition](#) dans la documentation d'Iceberg pour un aperçu de toutes les transformations de partition disponibles.

Les colonnes utilisées pour le partitionnement masqué peuvent faire partie des prédicats de requête grâce à l'utilisation de fonctions SQL classiques telles que `year()` et `month()`. Les prédicats peuvent également être combinés avec des opérateurs tels que `BETWEEN` et `AND`.

#### Note

Iceberg ne peut pas effectuer d'élégage de partition pour les fonctions qui produisent un type de données différent, par exemple, `substring(event_time, 1, 10) = '2022-01-01'`

## Utiliser Partition Evolution

Utilisez l'[évolution des partitions d'Iceberg](#) lorsque la stratégie de partition existante n'est pas optimale. Par exemple, si vous choisissez des partitions horaires qui s'avèrent trop petites (quelques mégaoctets chacune), envisagez de passer à des partitions quotidiennes ou mensuelles.

Vous pouvez utiliser cette approche lorsque la meilleure stratégie de partition pour une table n'est initialement pas claire et que vous souhaitez affiner votre stratégie de partitionnement au fur et à mesure que vous en apprendrez davantage. Une autre utilisation efficace de l'évolution des partitions est lorsque les volumes de données changent et que la stratégie de partitionnement actuelle devient moins efficace au fil du temps.

Pour obtenir des instructions sur la façon de faire évoluer les partitions, consultez les [extensions SQL ALTER TABLE](#) dans la documentation d'Iceberg.

## Réglage de la taille des fichiers

L'optimisation des performances des requêtes implique de minimiser le nombre de petits fichiers dans vos tables. Pour de bonnes performances de requête, nous recommandons généralement de conserver les fichiers Parquet et ORC d'une taille supérieure à 100 Mo.

La taille du fichier a également un impact sur la planification des requêtes pour les tables Iceberg. À mesure que le nombre de fichiers d'une table augmente, la taille des fichiers de métadonnées augmente également. Les fichiers de métadonnées volumineux peuvent ralentir la planification des requêtes. Par conséquent, lorsque la taille de la table augmente, augmentez la taille du fichier pour limiter l'expansion exponentielle des métadonnées.

Suivez les bonnes pratiques suivantes pour créer des fichiers correctement dimensionnés dans les tables Iceberg.

### Définir la taille du fichier cible et du groupe de lignes

Iceberg propose les principaux paramètres de configuration suivants pour ajuster la mise en page des fichiers de données. Nous vous recommandons d'utiliser ces paramètres pour définir la taille du fichier cible, le groupe de lignes ou la taille de trait.

| Paramètre                                       | Valeur par défaut | Commentaire                                                                                                                                                       |
|-------------------------------------------------|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>write.target-file-size-bytes</code>       | 512 Mo            | Ce paramètre indique la taille de fichier maximale qu'Iceberg créera. Cependant, certains fichiers peuvent être écrits avec une taille inférieure à cette limite. |
| <code>write.parquet.row-group-size-bytes</code> | 128 Mo            | Parquet et ORC stockent les données par blocs afin que les moteurs puissent éviter de lire le fichier entier pour certaines opérations.                           |

| Paramètre                                | Valeur par défaut                                                                                       | Commentaire                                                                                         |
|------------------------------------------|---------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------|
| <code>write.orc.stripe-size-bytes</code> | 64 MO                                                                                                   |                                                                                                     |
| <code>write.distribution-mode</code>     | Aucun, pour les versions 1.1 et antérieures d'Iceberg<br><br>Hash, à partir de la version 1.2 d'Iceberg | Iceberg demande à Spark de trier les données entre ses tâches avant de les écrire dans le stockage. |

- En fonction de la taille de table prévue, suivez les directives générales suivantes :
  - Petites tables (jusqu'à quelques gigaoctets) : réduisez la taille du fichier cible à 128 Mo. Réduisez également le groupe de lignes ou la taille de bande (par exemple, à 8 ou 16 Mo).
  - Tables de taille moyenne à grande (de quelques gigaoctets à des centaines de gigaoctets) : les valeurs par défaut constituent un bon point de départ pour ces tables. Si vos requêtes sont très sélectives, ajustez le groupe de lignes ou la taille de bande (par exemple, à 16 Mo).
  - Tables très volumineuses (centaines de gigaoctets ou téraoctets) : augmentez la taille du fichier cible à 1 024 Mo ou plus, et envisagez d'augmenter le groupe de lignes ou la taille de bande si vos requêtes contiennent généralement de grands ensembles de données.
- Pour garantir que les applications Spark qui écrivent dans les tables Iceberg créent des fichiers de taille appropriée, définissez la `write.distribution-mode` propriété sur `hash` ou `range`. Pour une explication détaillée de la différence entre ces modes, consultez la section [Writing Distribution Modes](#) dans la documentation d'Iceberg.

Il s'agit de directives générales. Nous vous recommandons d'exécuter des tests afin d'identifier les valeurs les plus adaptées à vos tables et charges de travail spécifiques.

## Effectuez un compactage régulier

Les configurations du tableau précédent définissent une taille de fichier maximale que les tâches d'écriture peuvent créer, mais ne garantissent pas que les fichiers auront cette taille. Pour garantir des tailles de fichier appropriées, effectuez régulièrement le compactage pour combiner de petits fichiers en fichiers plus volumineux. Pour obtenir des conseils détaillés sur le compactage en cours, voir le [compactage d'icebergs](#) plus loin dans ce guide.

## Optimisation des statistiques des colonnes

Iceberg utilise les statistiques des colonnes pour élaguer les fichiers, ce qui améliore les performances des requêtes en réduisant la quantité de données analysées par les requêtes. Pour bénéficier des statistiques sur les colonnes, assurez-vous qu'Iceberg collecte des statistiques pour toutes les colonnes fréquemment utilisées dans les filtres de requêtes.

Par défaut, Iceberg collecte des statistiques uniquement pour les [100 premières colonnes de chaque table](#), comme défini par la propriété `write.metadata.metrics.max-inferred-column-defaults` de la table. Si votre table comporte plus de 100 colonnes et que vos requêtes font fréquemment référence à des colonnes situées en dehors des 100 premières colonnes (par exemple, vous pouvez avoir des requêtes qui filtrent sur la colonne 132), assurez-vous qu'Iceberg collecte des statistiques sur ces colonnes. Deux options s'offrent à vous pour y parvenir :

- Lorsque vous créez la table Iceberg, réorganisez les colonnes de manière à ce que les colonnes dont vous avez besoin pour les requêtes se situent dans la plage de colonnes définie par `write.metadata.metrics.max-inferred-column-defaults` (la valeur par défaut est 100).

Remarque : Si vous n'avez pas besoin de statistiques sur 100 colonnes, vous pouvez ajuster la `write.metadata.metrics.max-inferred-column-defaults` configuration à la valeur souhaitée (par exemple, 20) et réorganiser les colonnes afin que les colonnes dont vous avez besoin pour lire et écrire des requêtes se situent dans les 20 premières colonnes du côté gauche du jeu de données.

- Si vous n'utilisez que quelques colonnes dans les filtres de requête, vous pouvez désactiver la propriété globale pour la collecte des mesures et choisir de manière sélective les colonnes individuelles pour lesquelles collecter les statistiques, comme illustré dans cet exemple :

```
.tableProperty("write.metadata.metrics.default", "none")
.tableProperty("write.metadata.metrics.column.my_col_a", "full")
.tableProperty("write.metadata.metrics.column.my_col_b", "full")
```

Remarque : Les statistiques des colonnes sont plus efficaces lorsque les données sont triées sur ces colonnes. Pour plus d'informations, consultez la section [Définir l'ordre de tri](#) plus loin dans ce guide.

## Choisissez la bonne stratégie de mise à jour

Utilisez une stratégie de copie sur écriture pour optimiser les performances de lecture, lorsque des opérations d'écriture plus lentes sont acceptables pour votre cas d'utilisation. Il s'agit de la stratégie par défaut utilisée par Iceberg.

Copy-on-write se traduit par de meilleures performances de lecture, car les fichiers sont directement écrits dans le stockage de manière optimisée pour la lecture. Cependant, par rapport à la fusion en lecture, chaque opération d'écriture prend plus de temps et consomme davantage de ressources informatiques. Cela représente un compromis classique entre latence de lecture et d'écriture. Généralement, le copy-on-write est idéal pour les cas d'utilisation où la plupart des mises à jour sont colocalisées dans les mêmes partitions de table (par exemple, pour les chargements par lots quotidiens).

Copy-on-write les configurations (`write.update.mode`, `write.delete.mode`, `write.merge.mode`) peuvent être définies au niveau de la table ou indépendamment du côté de l'application.

## Utiliser la compression ZSTD

Vous pouvez modifier le codec de compression utilisé par Iceberg à l'aide de la propriété `table.write.<file_type>.compression-codec`. Nous vous recommandons d'utiliser le codec de compression ZSTD pour améliorer les performances globales des tables.

Par défaut, les versions 1.3 et antérieures d'Iceberg utilisent la compression GZIP, ce qui réduit les read/write performances par rapport à ZSTD.

Remarque : Certains moteurs peuvent utiliser des valeurs par défaut différentes. C'est le cas des [tables Iceberg créées avec Athena](#) ou Amazon EMR version 7.x.

## Définissez l'ordre de tri

Pour améliorer les performances de lecture des tables Iceberg, nous vous recommandons de trier votre table en fonction d'une ou de plusieurs colonnes fréquemment utilisées dans les filtres de requêtes. Le tri, combiné aux statistiques des colonnes d'Iceberg, peut rendre l'élagage des fichiers nettement plus efficace, ce qui se traduit par des opérations de lecture plus rapides. Le tri réduit également le nombre de demandes Amazon S3 pour les requêtes qui utilisent les colonnes de tri des filtres de requêtes.

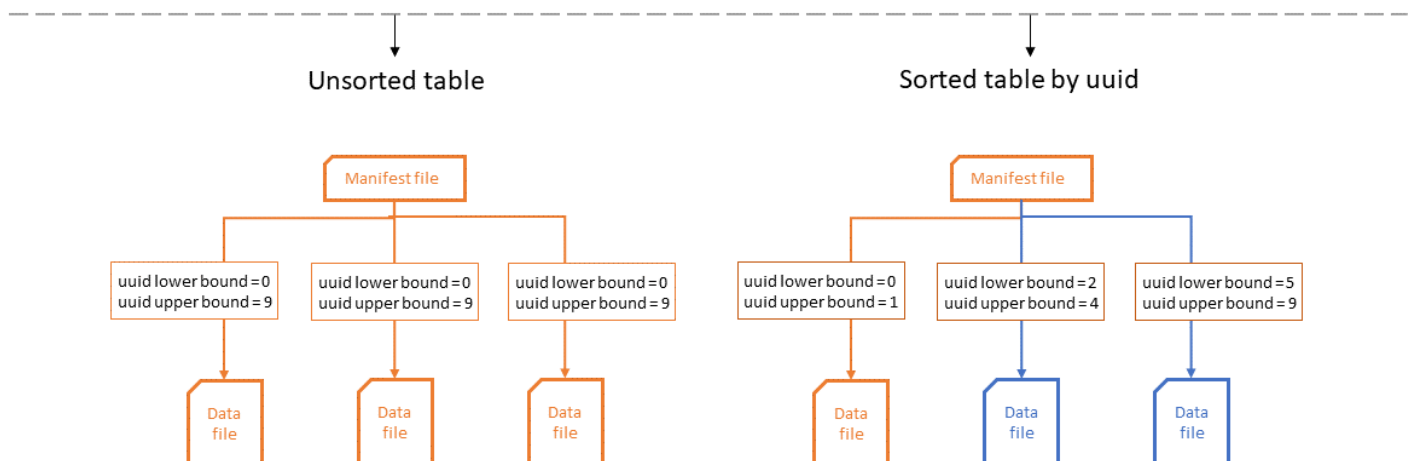
Vous pouvez définir un ordre de tri hiérarchique au niveau de la table en exécutant une instruction DDL (Data Definition Language) avec Spark. Pour connaître les options disponibles, consultez la [documentation d'Iceberg](#). Après avoir défini l'ordre de tri, les rédacteurs appliqueront ce tri aux opérations d'écriture de données suivantes dans la table Iceberg.

Par exemple, dans les tables partitionnées par date (yyyy-mm-dd) où la plupart des requêtes sont filtrées par `uuid`, vous pouvez utiliser l'option DDL `Write Distributed By Partition Locally Ordered` pour vous assurer que Spark écrit des fichiers dont les plages ne se chevauchent pas.

Le schéma suivant montre comment l'efficacité des statistiques de colonnes s'améliore lorsque les tables sont triées. Dans l'exemple, la table triée ne doit ouvrir qu'un seul fichier et tire le meilleur parti de la partition et du fichier d'Iceberg. Dans le tableau non trié, tout `uuid` peut potentiellement exister dans n'importe quel fichier de données. La requête doit donc ouvrir tous les fichiers de données.

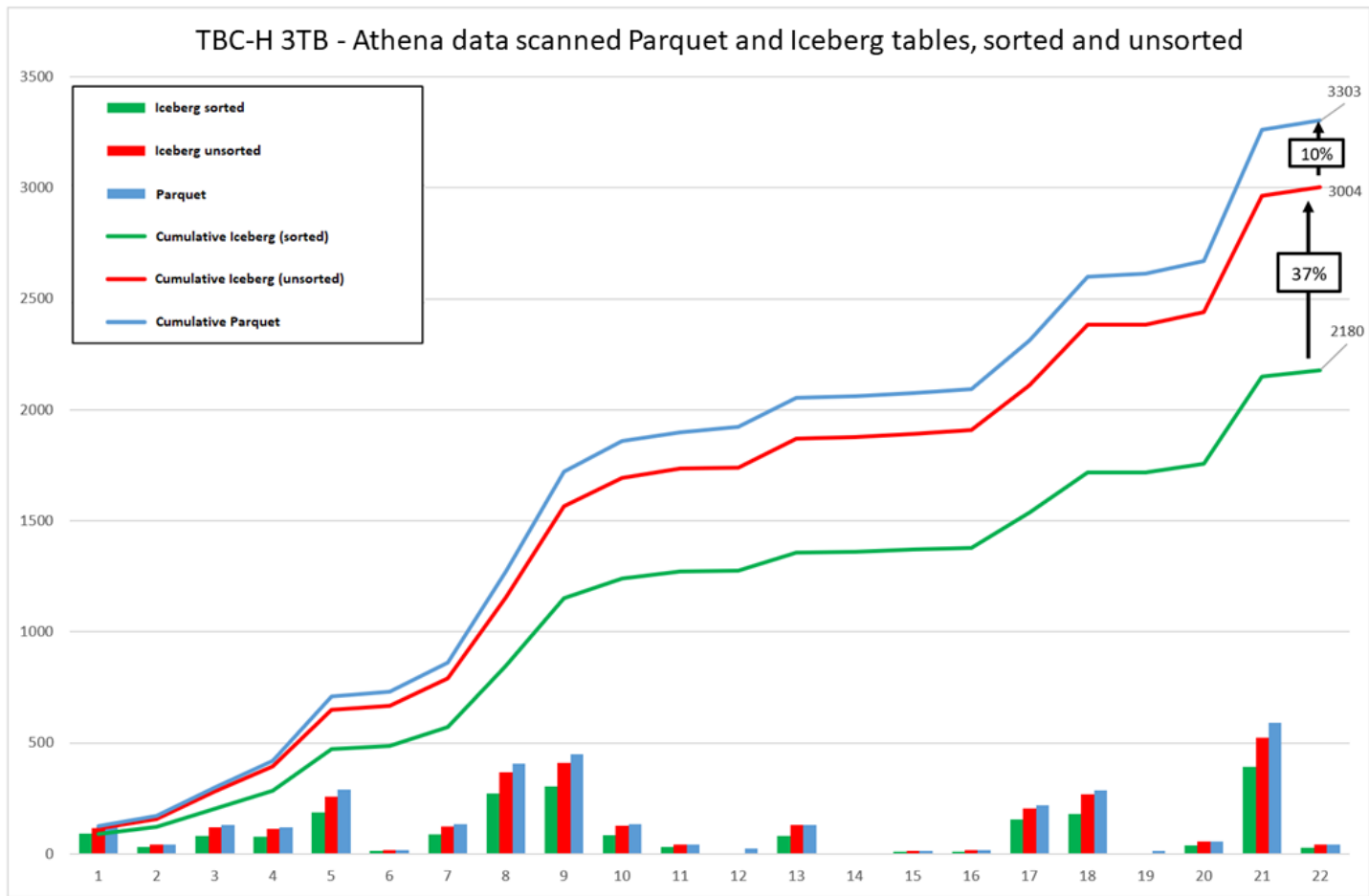
Query example:

```
SELECT * FROM Table
WHERE date > 2022-02-05 AND date < 2022-02-10 AND uuid = 1
```



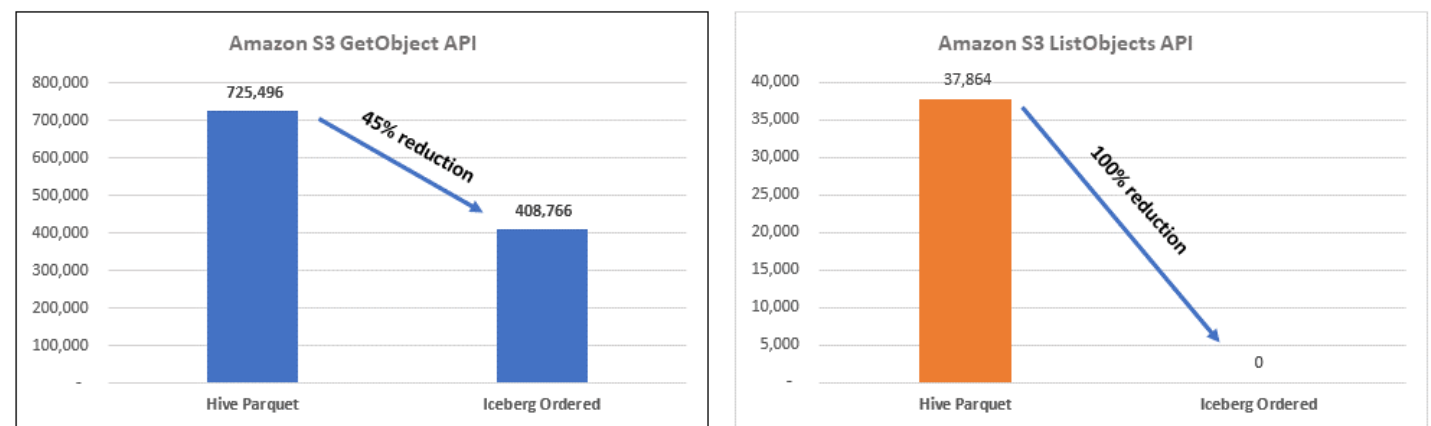
La modification de l'ordre de tri n'affecte pas les fichiers de données existants. Vous pouvez utiliser le compactage Iceberg pour leur appliquer l'ordre de tri.

L'utilisation de tables triées par Iceberg peut réduire les coûts liés à votre charge de travail, comme l'illustre le graphique suivant.



Ces graphiques résument les résultats de l'analyse TPC-H comparative entre les tables Hive (Parquet) et les tables triées par Iceberg. Toutefois, les résultats peuvent être différents pour d'autres ensembles de données ou charges de travail.

TPC-H 3TB - 22 queries



# Optimisation des performances d'écriture

Cette section décrit les propriétés des tables que vous pouvez ajuster pour optimiser les performances d'écriture sur les tables Iceberg, indépendamment du moteur.

## Définissez le mode de distribution des tables

Iceberg propose plusieurs modes de distribution d'écriture qui définissent la manière dont les données d'écriture sont distribuées entre les tâches Spark. Pour un aperçu des modes disponibles, consultez la section [Writing Distribution Modes](#) dans la documentation d'Iceberg.

Pour les cas d'utilisation qui privilégient la vitesse d'écriture, en particulier pour les charges de travail de streaming, définissez `surwrite.distribution-mode: none`. Cela garantit qu'Iceberg ne demande pas de remaniement Spark supplémentaire et que les données sont écrites dès qu'elles sont disponibles dans les tâches Spark. Ce mode est particulièrement adapté aux applications Spark Structured Streaming.

### Note

La définition du mode de distribution des écritures `none` a tendance à produire de nombreux petits fichiers, ce qui dégrade les performances de lecture. Nous recommandons un compactage régulier pour consolider ces petits fichiers en fichiers correctement dimensionnés afin d'améliorer les performances des requêtes.

## Choisissez la bonne stratégie de mise à jour

Utilisez une stratégie de fusion lors de la lecture pour optimiser les performances d'écriture, lorsque des opérations de lecture plus lentes sur les données les plus récentes sont acceptables pour votre cas d'utilisation.

Lorsque vous utilisez Merge-on-Read, Iceberg écrit les mises à jour et les suppressions dans le stockage sous forme de petits fichiers distincts. Lorsque le tableau est lu, le lecteur doit fusionner ces modifications avec les fichiers de base pour obtenir la dernière vue des données. Cela entraîne une baisse des performances pour les opérations de lecture, mais accélère l'écriture des mises à jour et des suppressions. La fusion en lecture est généralement idéale pour diffuser des charges de travail contenant des mises à jour ou des tâches comportant peu de mises à jour réparties sur de nombreuses partitions de table.

Vous pouvez définir des configurations de fusion lors de la lecture (`write.update.mode`, `write.delete.mode`, `etwrite.merge.mode`) au niveau de la table ou indépendamment du côté de l'application.

L'utilisation de la fusion lors de la lecture nécessite un compactage régulier pour éviter que les performances de lecture ne se dégradent au fil du temps. Le compactage concilie les mises à jour et les suppressions avec les fichiers de données existants afin de créer un nouvel ensemble de fichiers de données, éliminant ainsi la perte de performance encourue du côté lecture. Par défaut, le compactage d'Iceberg ne fusionne pas les fichiers de suppression, sauf si vous remplacez la valeur par défaut de la `delete-file-threshold` propriété par une valeur inférieure (voir la documentation d'[Iceberg](#)). Pour en savoir plus sur le compactage, consultez la section [Compaction des icebergs](#) plus loin dans ce guide.

## Choisissez le bon format de fichier

Iceberg prend en charge l'écriture de données aux formats Parquet, ORC et Avro. Le parquet est le format par défaut. Parquet et ORC sont des formats en colonnes qui offrent des performances de lecture supérieures mais sont généralement plus lents à écrire. Cela représente le compromis typique entre les performances de lecture et d'écriture.

Si la vitesse d'écriture est importante pour votre cas d'utilisation, par exemple pour les charges de travail en streaming, envisagez d'écrire au format Avro en la `write-format` configurant Avro dans les options du rédacteur. Avro étant un format basé sur des lignes, il permet d'accélérer les temps d'écriture au prix de performances de lecture plus lentes.

Pour améliorer les performances de lecture, effectuez un compactage régulier pour fusionner et transformer de petits fichiers Avro en fichiers Parquet plus volumineux. Le résultat du processus de compactage est régi par le réglage de la `write.format.default` table. Le format par défaut d'Iceberg est Parquet. Ainsi, si vous écrivez dans Avro puis exécutez le compactage, Iceberg transformera les fichiers Avro en fichiers Parquet. Voici un exemple :

```
spark.sql(f"""
  CREATE TABLE IF NOT EXISTS glue_catalog.{DB_NAME}.{TABLE_NAME} (
    Col_1 float,
    <<<...other columns...>>>
    ts timestamp)
  USING iceberg
  PARTITIONED BY (days(ts))
  OPTIONS (
```

```

    'format-version'='2',
    write.format.default='parquet')
    """)

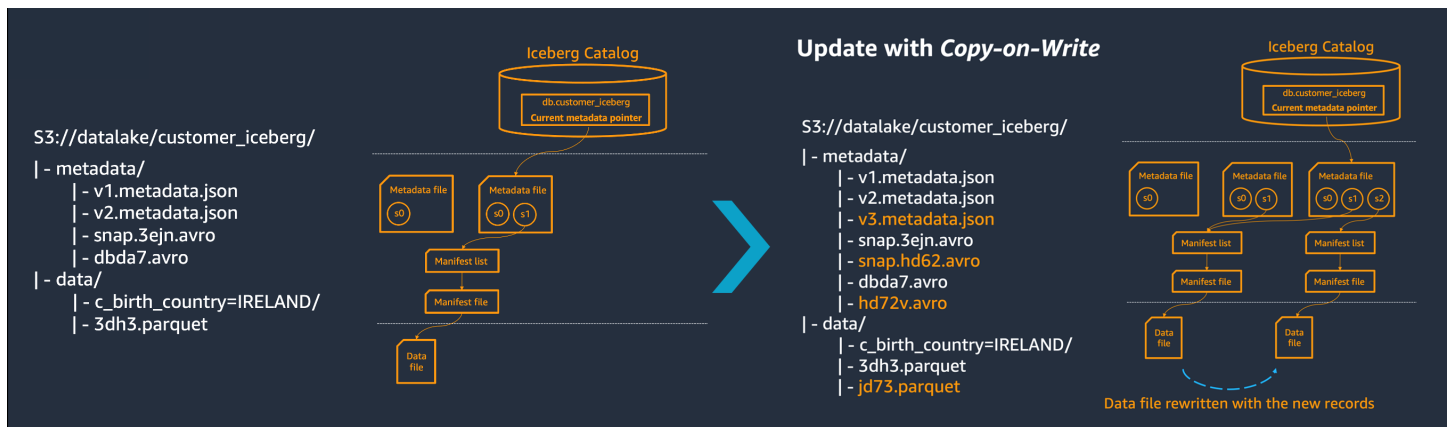
query = df \
    .writeStream \
    .format("iceberg") \
    .option("write-format", "avro") \
    .outputMode("append") \
    .trigger(processingTime='60 seconds') \
    .option("path", f"glue_catalog.{DB_NAME}.{TABLE_NAME}") \
    .option("checkpointLocation", f"s3://{BUCKET_NAME}/checkpoints/iceberg/")

    .start()

```

## Optimisation du stockage

La mise à jour ou la suppression de données dans une table Iceberg augmente le nombre de copies de vos données, comme illustré dans le schéma suivant. Il en va de même pour l'exécution du compactage : il augmente le nombre de copies de données dans Amazon S3. C'est parce qu'Iceberg considère les fichiers sous-jacents à toutes les tables comme immuables.



Suivez les meilleures pratiques décrites dans cette section pour gérer les coûts de stockage.

## Activer S3 Intelligent-Tiering

Utilisez la classe de Intelligent-Tiering stockage [Amazon S3](#) pour déplacer automatiquement les données vers le niveau d'accès le plus rentable lorsque les modèles d'accès changent. Cette option n'entraîne aucune surcharge opérationnelle ni aucun impact sur les performances.

Remarque : N'utilisez pas les niveaux facultatifs (tels que Archive Access et Deep Archive Access) dans S3 Intelligent-Tiering avec les tables Iceberg. Pour archiver des données, consultez les instructions de la section suivante.

Vous pouvez également utiliser les [règles du cycle de vie d'Amazon S3](#) pour définir vos propres règles de déplacement d'objets vers une autre classe de stockage Amazon S3, telle que S3 Standard-IA ou S3 One Zone-IA (voir [Transitions prises en charge et contraintes associées](#) dans la documentation Amazon S3).

## Archiver ou supprimer des instantanés historiques

Pour chaque transaction validée (insertion, mise à jour, fusion dans, compactage) dans une table Iceberg, une nouvelle version ou un instantané de la table est créé. Au fil du temps, le nombre de versions et le nombre de fichiers de métadonnées s'accumulent dans Amazon S3.

La conservation des instantanés d'une table est requise pour des fonctionnalités telles que l'isolation des instantanés, la restauration des tables et les requêtes de voyage dans le temps. Toutefois, les coûts de stockage augmentent en fonction du nombre de versions que vous conservez.

Le tableau suivant décrit les modèles de conception que vous pouvez mettre en œuvre pour gérer les coûts en fonction de vos exigences en matière de conservation des données.

| Motif de design                   | Solution                                                                                                                                                                                                                                                                                                                                                                              | Cas d'utilisation                                                                                                                                                                                                                                                                                                                                                                              |
|-----------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Supprimer les anciens instantanés | <ul style="list-style-type: none"><li>Utilisez l'<a href="#">instruction VACUUM</a> dans Athena pour supprimer les anciens instantanés. Cette opération n'entraîne aucun coût de calcul.</li><li><a href="#">Vous pouvez également utiliser Spark sur Amazon EMR ou AWS Glue pour supprimer des instantanés. Pour plus d'informations, consultez <code>expire_sn</code></a></li></ul> | <p>Cette approche supprime les instantanés qui ne sont plus nécessaires pour réduire les coûts de stockage. Vous pouvez configurer le nombre d'instantanés à conserver ou pendant combien de temps, en fonction de vos exigences en matière de conservation des données.</p> <p>Cette option effectue une suppression définitive des instantanés. Vous ne pouvez pas revenir en arrière ou</p> |

## Motif de design

## Solution

[apshots dans la documentation d'Iceberg.](#)

## Cas d'utilisation

voyager dans le temps pour retrouver des instantanés expirés.

Définissez des règles de conservation pour des instantanés spécifiques

1. Utilisez des balises pour marquer des instantanés spécifiques et définir une politique de rétention dans Iceberg. Pour plus d'informations, consultez la section [Balises historiques](#) dans la documentation d'Iceberg.

Par exemple, vous pouvez conserver un instantané par mois pendant un an en utilisant l'instruction SQL suivante dans Spark on Amazon EMR :

```
ALTER TABLE glue_catalog.db.table
CREATE TAG 'EOM-01' AS
OF VERSION 30 RETAIN
365 DAYS
```

2. Utilisez Spark sur Amazon EMR ou AWS Glue pour supprimer les instantanés intermédiaires non balisés restants.

Ce modèle est utile pour vous conformer aux exigences commerciales ou légales qui vous obligent à montrer l'état d'un tableau à un moment donné dans le passé. En appliquant des règles de conservation à des instantanés balisés spécifiques, vous pouvez supprimer d'autres instantanés (non balisés) créés. Ainsi, vous pouvez répondre aux exigences de conservation des données sans conserver chaque instantané créé.

| Motif de design                  | Solution                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Cas d'utilisation                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|----------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Archiver les anciens instantanés | <ol style="list-style-type: none"><li>Utilisez les balises Amazon S3 pour marquer des objets avec Spark. (Les balises Amazon S3 sont différentes des balises Iceberg ; pour plus d'informations, consultez la <a href="#">documentation d'Iceberg</a>.) Par exemple :<div><pre>spark.sql.catalog.<br/>my_catalog.s3.delete-enabled=false and<br/>\<br/>spark.sql.catalog.<br/>g.my_catalog.s3.delete.tags.my_key=to_archive</pre></div></li><li>Utilisez Spark sur Amazon EMR ou AWS Glue pour <a href="#">supprimer</a> des instantanés. Lorsque vous utilisez les paramètres de l'exemple , cette procédure balise les objets et les détache des métadonnées de la table Iceberg au lieu de les supprimer d'Amazon S3.</li><li>Utilisez les règles du cycle de vie de S3 pour transférer les objets marqués comme <code>to_archive</code> vers l'une des <a href="#">classes de stockage S3 Glacier</a>.</li></ol> | <p>Ce modèle vous permet de conserver toutes les versions des tables et les instantanés à moindre coût.</p> <p>Vous ne pouvez pas voyager dans le temps ou revenir à des instantanés archivés sans avoir d'abord restauré ces versions sous forme de nouvelles tables. Cela est généralement acceptable à des fins d'audit.</p> <p>Vous pouvez combiner cette approche avec le modèle de conception précédent, en définissant des politiques de conservation pour des instantanés spécifiques.</p> |

| Motif de design | Solution                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | Cas d'utilisation |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|
|                 | <p>4. Pour interroger les données archivées, procédez comme suit :</p> <ul style="list-style-type: none"><li>• <a href="#">Restaurez les objets archivés</a> (cette étape n'est pas obligatoire si les objets ont été transférés vers la classe de stockage Amazon Glacier Instant Retrieval).</li><li>• Utilisez la <a href="#">procédure register_table</a> dans Iceberg pour enregistrer l'instantané en tant que table dans le catalogue.</li></ul> <p>Pour obtenir des instructions détaillées, consultez le billet de AWS blog <a href="#">Améliorez l'efficacité opérationnelle des tables Apache Iceberg basées sur les lacs de données Amazon S3</a>.</p> |                   |

## Supprimer les fichiers orphelins

Dans certaines situations, les applications Iceberg peuvent échouer avant que vous n'ayez validé vos transactions. Cela laisse les fichiers de données dans Amazon S3. Comme il n'y a pas eu de validation, ces fichiers ne seront associés à aucune table. Vous devrez peut-être les nettoyer de manière asynchrone.

Pour gérer ces suppressions, vous pouvez utiliser l'[instruction VACUUM](#) dans Amazon Athena. Cette instruction supprime les instantanés et supprime également les fichiers orphelins. C'est très rentable, car Athena ne facture pas le coût de calcul de cette opération. De plus, il n'est pas nécessaire de planifier d'opérations supplémentaires lorsque vous utilisez l'`VACUUM` instruction.

Vous pouvez également utiliser Spark sur Amazon EMR ou AWS Glue exécuter la `remove_orphan_files` procédure. Cette opération a un coût de calcul et doit être planifiée indépendamment. Pour plus d'informations, consultez la [documentation d'Iceberg](#).

## Maintien des tables en utilisant le compactage

Iceberg inclut des fonctionnalités qui vous permettent d'effectuer des [opérations de maintenance de tables](#) après y avoir écrit des données. Certaines opérations de maintenance se concentrent sur la rationalisation des fichiers de métadonnées, tandis que d'autres améliorent la manière dont les données sont regroupées dans les fichiers afin que les moteurs de requêtes puissent localiser efficacement les informations nécessaires pour répondre aux demandes des utilisateurs. Cette section se concentre sur les optimisations liées au compactage.

### Compactage des icebergs

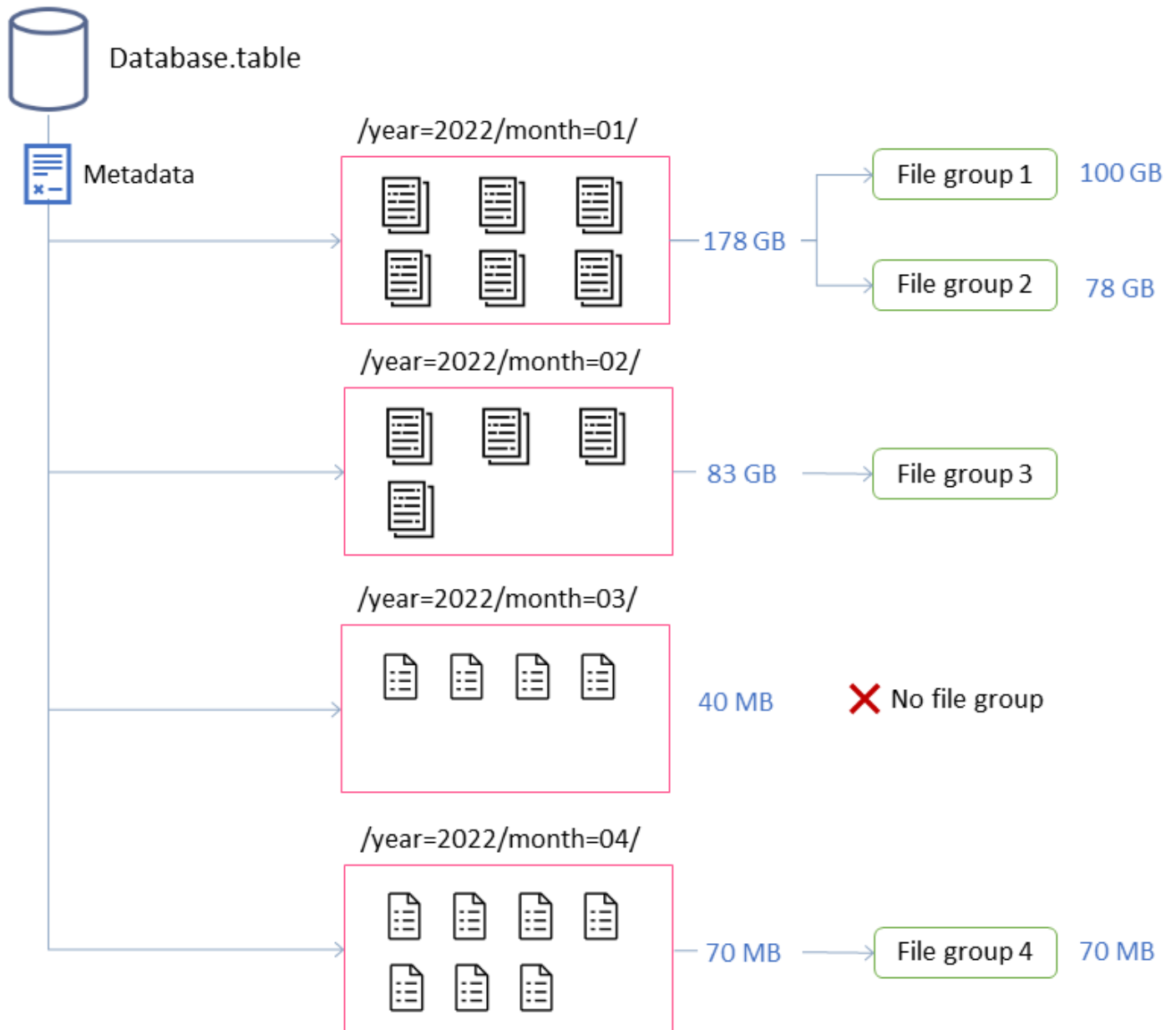
Dans Iceberg, vous pouvez utiliser le compactage pour effectuer quatre tâches :

- Combinaison de petits fichiers en fichiers plus volumineux dont la taille est généralement supérieure à 100 Mo. Cette technique est connue sous le nom de « bin packing ».
- Fusion de fichiers de suppression avec des fichiers de données. Les fichiers de suppression sont générés par des mises à jour ou des suppressions utilisant l'approche de fusion lors de la lecture.
- (Re) tri des données conformément aux modèles de requête. Les données peuvent être écrites sans ordre de tri ou avec un ordre de tri adapté aux écritures et aux mises à jour.
- Regrouper les données en utilisant des courbes de remplissage d'espace pour optimiser les modèles de requêtes distincts, en particulier le tri par ordre Z.

Vous pouvez exécuter des opérations de compactage et de maintenance de tables pour Iceberg via Amazon Athena ou en utilisant Spark dans Amazon EMR ou AWS Glue.

Lorsque vous exécutez le compactage à l'aide de la procédure [rewrite\\_data\\_files](#), vous pouvez régler plusieurs boutons pour contrôler le comportement de compactage. Le schéma suivant montre le comportement par défaut du bin packing. Comprendre le compactage de l'emballage en bacs

est essentiel pour comprendre le tri hiérarchique et les implémentations de Z-order tri, car il s'agit d'extensions de l'interface d'emballage en bacs et fonctionnent de manière similaire. La principale différence réside dans l'étape supplémentaire requise pour trier ou regrouper les données.



Dans cet exemple, la table Iceberg est composée de quatre partitions. Chaque partition a une taille et un nombre de fichiers différents. Si vous démarrez une application Spark pour exécuter le compactage, l'application crée au total quatre groupes de fichiers à traiter. Un groupe de fichiers est une abstraction d'Iceberg qui représente un ensemble de fichiers qui seront traités par une seule tâche Spark. En d'autres termes, l'application Spark qui exécute le compactage créera quatre tâches Spark pour traiter les données.

## Réglage du comportement de compactage

Les propriétés clés suivantes contrôlent la manière dont les fichiers de données sont sélectionnés pour le compactage :

- [MAX\\_FILE\\_GROUP\\_SIZE\\_BYTES](#) définit la limite de données pour un seul groupe de fichiers (tâche Spark) à 100 Go par défaut. Cette propriété est particulièrement importante pour les tables sans partitions ou les tables dont les partitions s'étendent sur des centaines de gigaoctets. En définissant cette limite, vous pouvez décomposer les opérations afin de planifier le travail et de progresser tout en évitant l'épuisement des ressources du cluster.

Remarque : Chaque groupe de fichiers est trié séparément. Par conséquent, si vous souhaitez effectuer un tri au niveau de la partition, vous devez ajuster cette limite en fonction de la taille de la partition.

- [MIN\\_FILE\\_SIZE\\_BYTES](#) ou [MIN\\_FILE\\_SIZE\\_DEFAULT\\_RATIO](#) utilise par défaut 75 % de la taille de fichier cible définie au niveau de la table. Par exemple, si la taille cible d'une table est de 512 Mo, tout fichier inférieur à 384 Mo est inclus dans l'ensemble de fichiers à compacter.
- [MAX\\_FILE\\_SIZE\\_BYTES](#) ou [MAX\\_FILE\\_SIZE\\_DEFAULT\\_RATIO](#) utilise par défaut 180 % de la taille du fichier cible. Comme pour les deux propriétés qui définissent les tailles de fichier minimales, ces propriétés sont utilisées pour identifier les fichiers candidats pour le travail de compactage.
- [MIN\\_INPUT\\_FILES](#) indique le nombre minimum de fichiers à compacter si la taille d'une partition de table est inférieure à la taille du fichier cible. La valeur de cette propriété est utilisée pour déterminer s'il est intéressant de compacter les fichiers en fonction du nombre de fichiers (5 par défaut).
- [DELETE\\_FILE\\_THRESHOLD](#) indique le nombre minimum d'opérations de suppression pour un fichier avant qu'il ne soit inclus dans le compactage. Sauf indication contraire, le compactage ne combine pas les fichiers de suppression avec les fichiers de données. Pour activer cette fonctionnalité, vous devez définir une valeur de seuil à l'aide de cette propriété. Ce seuil est spécifique à chaque fichier de données. Par conséquent, si vous le définissez sur 3, un fichier de données ne sera réécrit que s'il existe au moins trois fichiers de suppression qui y font référence.

Ces propriétés fournissent un aperçu de la formation des groupes de fichiers dans le schéma précédent.

Par exemple, la partition étiquetée month=01 inclut deux groupes de fichiers car elle dépasse la limite de taille maximale de 100 Go. En revanche, la month=02 partition contient un seul groupe

de fichiers car sa taille est inférieure à 100 Go. La `month=03` partition ne répond pas à l'exigence minimale de cinq fichiers d'entrée par défaut. Par conséquent, il ne sera pas compacté. Enfin, bien que la `month=04` partition ne contienne pas suffisamment de données pour former un seul fichier de la taille souhaitée, les fichiers seront compactés car la partition contient plus de cinq petits fichiers.

Vous pouvez définir ces paramètres pour que Spark s'exécute sur Amazon EMR ou. AWS Glue Pour Amazon Athena, vous pouvez gérer des propriétés similaires en utilisant les propriétés du [tableau](#) (qui commencent par le préfixe `optimize_`).

## Exécution du compactage avec Spark sur Amazon EMR ou AWS Glue

Cette section explique comment dimensionner correctement un cluster Spark pour exécuter l'utilitaire de compactage d'Iceberg. L'exemple suivant utilise Amazon EMR Serverless, mais vous pouvez utiliser la même méthodologie dans Amazon EMR sur EC2 ou EKS, ou dans. AWS Glue

Vous pouvez tirer parti de la corrélation entre les groupes de fichiers et les tâches Spark pour planifier les ressources du cluster. Pour traiter les groupes de fichiers de manière séquentielle, compte tenu de la taille maximale de 100 Go par groupe de fichiers, vous pouvez définir les [propriétés Spark](#) suivantes :

- `spark.dynamicAllocation.enabled = FALSE`
- `spark.executor.memory = 20 GB`
- `spark.executor.instances = 5`

Pour accélérer le compactage, vous pouvez effectuer une mise à l'échelle horizontale en augmentant le nombre de groupes de fichiers compactés en parallèle. Vous pouvez également redimensionner Amazon EMR en utilisant un dimensionnement manuel ou dynamique.

- Mise à l'échelle manuelle (par exemple, par un facteur 4)
  - `MAX_CONCURRENT_FILE_GROUP_REWRITES= 4` (notre facteur)
  - `spark.executor.instances= 5` (valeur utilisée dans l'exemple) x 4 (notre facteur) = 20
  - `spark.dynamicAllocation.enabled = FALSE`
- Mise à l'échelle dynamique
  - `spark.dynamicAllocation.enabled= TRUE` (par défaut, aucune action requise)
  - [MAX\\_CONCURRENT\\_FILE\\_GROUP\\_REWRITES](#) = N (aligne cette valeur sur 100 par défaut ; en fonction des `spark.dynamicAllocation.maxExecutors` configurations de l'exécuteur indiquées dans l'exemple, vous pouvez définir la valeur sur 20) N

Il s'agit de directives destinées à aider à dimensionner le cluster. Cependant, vous devez également surveiller les performances de vos tâches Spark afin de trouver les meilleurs paramètres pour vos charges de travail.

## Exécution du compactage avec Amazon Athena

[Athena propose une implémentation de l'utilitaire de compactage d'Iceberg en tant que fonctionnalité gérée via l'instruction OPTIMIZE.](#) Vous pouvez utiliser cette instruction pour exécuter le compactage sans avoir à évaluer l'infrastructure.

Cette instruction regroupe les petits fichiers en fichiers plus volumineux à l'aide de l'algorithme de bin packing et fusionne les fichiers de suppression avec les fichiers de données existants. Pour regrouper les données en utilisant le tri hiérarchique ou le tri par ordre Z, utilisez Spark sur Amazon AWS Glue EMR ou.

Vous pouvez modifier le comportement par défaut de l'OPTIMIZE instruction lors de la création de la table en transmettant les propriétés de la table dans l'CREATE TABLE instruction, ou après la création de la table en utilisant l'ALTER TABLE instruction. Pour les valeurs par défaut, consultez la documentation d'[Athena](#).

## Recommandations pour le compactage en cours

### Cas d'utilisation

Exécution du compactage des emballages dans les bacs selon un calendrier

### Recommandation

- Utilisez l'OPTIMIZE instruction d'Athena si vous ne savez pas combien de petits fichiers contient votre table. Le modèle de tarification d'Athena est basé sur les données numérisées. Ainsi, s'il n'y a aucun fichier à compacter, aucun coût n'est associé à ces opérations. Pour éviter de rencontrer des délais d'attente sur les tables Athena, OPTIMIZE exécutez-les par partition de table.
- Utilisez Amazon EMR ou AWS Glue le dimensionnement dynamique lorsque vous

## Cas d'utilisation

## Recommandation

vous attendez à ce que de gros volumes de petits fichiers soient compactés.

Exécution du compactage des bacs en fonction des événements

- Utilisez Amazon EMR ou AWS Glue le dimensionnement dynamique lorsque vous vous attendez à ce que de gros volumes de petits fichiers soient compactés.

Exécuter le compactage pour trier les données

- Utilisez Amazon EMR ou AWS Glue, parce que le tri est une opération coûteuse qui peut nécessiter le transfert de données sur le disque.

Exécution du compactage pour regrouper les données à l'aide du tri par ordre Z

- Utilisez Amazon EMR ou AWS Glue, car le tri par ordre Z est une opération très coûteuse qui peut nécessiter le transfert de données sur disque.

Exécution du compactage sur des partitions susceptibles d'être mises à jour par d'autres applications en raison de l'arrivée tardive de données

- Utilisez Amazon EMR ou. AWS Glue Activez la propriété Iceberg [PARTIAL\\_PROGRESS\\_ENABLED](#). Lorsque vous utilisez cette option, Iceberg divise la sortie de compactage en plusieurs validations. En cas de collision (c'est-à-dire si le fichier de données est mis à jour alors que le compactage est en cours d'exécution), ce paramètre réduit le coût d'une nouvelle tentative en le limitant à la validation qui inclut le fichier concerné. Dans le cas contraire, vous devrez peut-être recompacter tous les fichiers.

## Cas d'utilisation

Exécution du compactage sur des partitions froides (partitions de données qui ne reçoivent plus d'écritures actives)

## Recommandation

- Utilisez Amazon EMR ou AWS Glue. Dans la `rewrite_data_files` procédure, spécifiez un `where` prédicat qui exclut les partitions écrites activement. Cette stratégie évite les conflits de données entre les rédacteurs et les tâches de compactage, et ne laisse que les conflits de métadonnées qu'Iceberg peut résoudre automatiquement.

# Utilisation des charges de travail Iceberg dans Amazon S3

Cette section décrit les propriétés d'Iceberg que vous pouvez utiliser pour optimiser l'interaction d'Iceberg avec Amazon S3.

## Empêcher le partitionnement à chaud (erreurs HTTP 503)

Certaines applications de lac de données exécutées sur Amazon S3 gèrent des millions ou des milliards d'objets et traitent des pétaoctets de données. Cela peut entraîner des préfixes recevant un volume de trafic élevé, généralement détectés par le biais d'erreurs HTTP 503 (service non disponible). Pour éviter ce problème, utilisez les propriétés Iceberg suivantes :

- Réglez `write.distribution-mode` de `range` manière à `hash` ce qu'Iceberg écrive des fichiers volumineux, ce qui réduit le nombre de demandes Amazon S3. Il s'agit de la configuration préférée qui devrait convenir à la majorité des cas.
- Si vous continuez à rencontrer 503 erreurs en raison d'un énorme volume de données dans vos charges de travail, vous pouvez configurer `write.object-storage.enabled` cette option `true` dans Iceberg. Cela indique à Iceberg de hacher les noms des objets et de répartir la charge sur plusieurs préfixes Amazon S3 aléatoires.

Pour plus d'informations sur ces propriétés, consultez la section [Write properties](#) dans la documentation d'Iceberg.

## Utilisez les opérations de maintenance d'Iceberg pour libérer les données inutilisées

Pour gérer les tables Iceberg, vous pouvez utiliser l'API principale d'Iceberg, les clients Iceberg (tels que Spark) ou les services gérés tels qu'Amazon Athena. Pour supprimer des fichiers anciens ou inutilisés d'Amazon S3, nous vous recommandons de n'utiliser que les API natives d'Iceberg pour [supprimer les instantanés](#), [supprimer les anciens fichiers de métadonnées](#) et [supprimer les fichiers orphelins](#).

L'utilisation des API Amazon S3 via Boto3, le SDK Amazon S3 ou le AWS Command Line Interface (AWS CLI), ou l'utilisation de toute autre méthode autre que celle d'Iceberg pour remplacer ou supprimer des fichiers Amazon S3 pour une table Iceberg entraîne la corruption de la table et l'échec des requêtes.

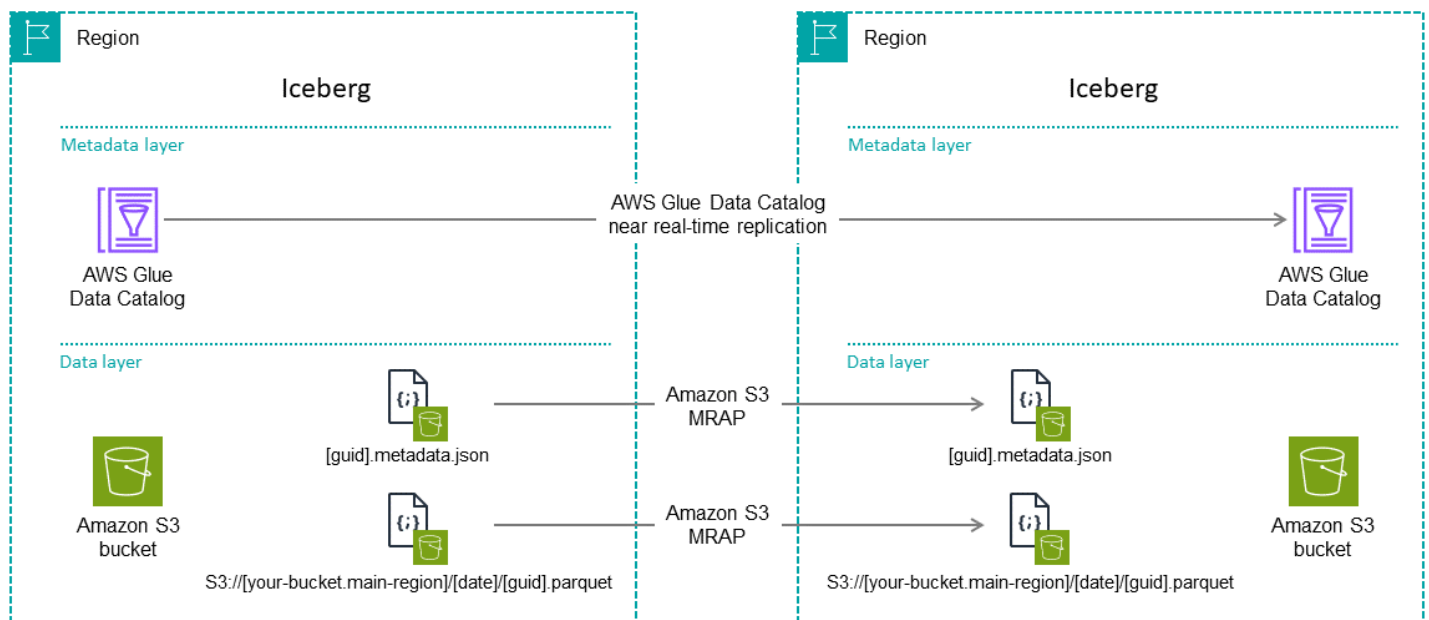
## Répliquez les données sur Régions AWS

Lorsque vous stockez des tables Iceberg dans Amazon S3, vous pouvez utiliser les fonctionnalités intégrées d'Amazon S3, telles que la [Cross-Region réplification \(CRR\)](#) et les [points d'Multi-Region accès \(MRAP\)](#), pour répliquer des données sur plusieurs Régions AWS. Le MRAP fournit un point de terminaison global permettant aux applications d'accéder aux compartiments S3 situés dans plusieurs compartiments. Régions AWS Iceberg ne prend pas en charge les chemins relatifs, mais vous pouvez utiliser le MRAP pour effectuer des opérations Amazon S3 en mappant des buckets à des points d'accès. Le MRAP s'intègre également parfaitement au processus de Cross-Region réplification Amazon S3, qui introduit un décalage pouvant atteindre 15 minutes. Vous devez répliquer les fichiers de données et de métadonnées.

### Important

Actuellement, l'intégration d'Iceberg au MRAP ne fonctionne qu'avec Apache Spark. Si vous devez passer au secondaire Région AWS, vous devez prévoir de rediriger les requêtes des utilisateurs vers un environnement Spark SQL (tel qu'Amazon EMR) dans la région de basculement.

Les fonctionnalités CRR et MRAP vous aident à créer une solution de réplification entre régions pour les tables Iceberg, comme illustré dans le schéma suivant.



Pour configurer cette architecture de réplication entre régions :

1. Créez des tables en utilisant l'emplacement MRAP. Cela garantit que les fichiers de métadonnées Iceberg pointent vers l'emplacement MRAP plutôt que vers l'emplacement physique du bucket.
2. Répliquez les fichiers Iceberg à l'aide d'Amazon S3 MRAP. Le MRAP prend en charge la réplication des données avec un accord de niveau de service (SLA) de 15 minutes. Iceberg empêche les opérations de lecture d'introduire des incohérences lors de la réplication.
3. Rendez les tables disponibles AWS Glue Data Catalog dans la région secondaire. Vous avez le choix entre deux options :
  - Configurez un pipeline pour répliquer les métadonnées des tables Iceberg à l'aide AWS Glue Data Catalog de la réplication. Cet utilitaire est disponible dans le référentiel de [réplication GitHub Glue Catalog et Lake Formation Permissions](#). Ce mécanisme piloté par les événements réplique les tables de la région cible en fonction des journaux d'événements.
  - Enregistrez les tables dans la région secondaire lorsque vous devez basculer. Pour cette option, vous pouvez utiliser l'utilitaire précédent ou la [procédure register\\_table](#) d'Iceberg et le faire pointer vers le dernier fichier .metadata.json

# Surveillance des charges de travail Apache Iceberg

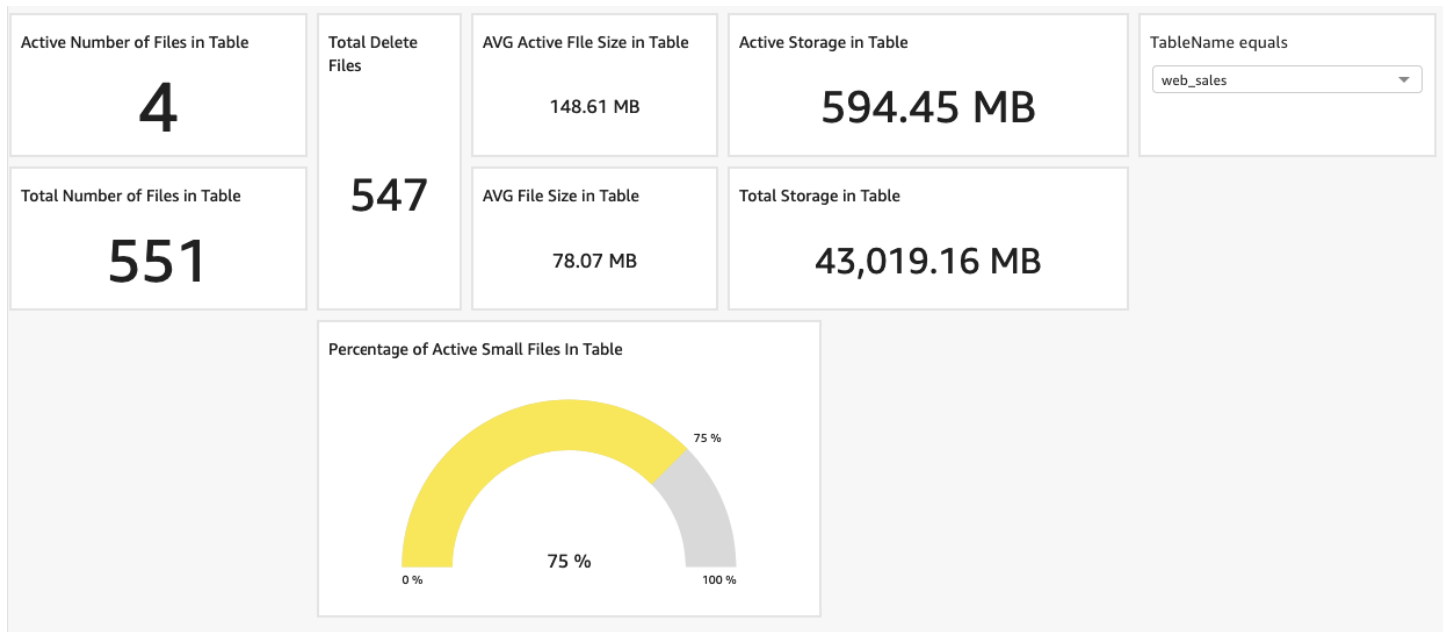
Pour surveiller les charges de travail d'Iceberg, deux options s'offrent à vous : analyser les [tables de métadonnées](#) ou utiliser des rapports de [mesures](#). Les reporters de métriques ont été introduits dans la version 1.2 d'Iceberg et ne sont disponibles que pour les catalogues REST et JDBC.

Si vous en utilisez AWS Glue Data Catalog, vous pouvez obtenir des informations sur l'état de vos tables Iceberg en configurant une surveillance au-dessus des tables de métadonnées exposées par Iceberg.

La surveillance est essentielle pour la gestion des performances et le dépannage. Par exemple, lorsqu'une partition d'une table Iceberg atteint un certain pourcentage de petits fichiers, votre charge de travail peut démarrer une tâche de compactage pour consolider les fichiers en fichiers plus volumineux. Cela empêche le ralentissement des requêtes au-delà d'un niveau acceptable.

## Table-level surveillance

L'écran suivant montre un tableau de bord de surveillance des tables créé dans Amazon Quick Sight. Ce tableau de bord interroge les tables de métadonnées d'Iceberg à l'aide de Spark SQL et capture des métriques détaillées telles que le nombre de fichiers actifs et le stockage total. Ces informations sont ensuite stockées dans AWS Glue des tables à des fins opérationnelles. Enfin, un tableau de bord Quick Sight, comme illustré dans l'illustration suivante, est créé à l'aide d'Amazon Athena. Ces informations vous aident à identifier et à résoudre les problèmes spécifiques de vos systèmes.



L'exemple de tableau de bord Quick Sight collecte les indicateurs de performance clés (KPI) suivants pour une table Iceberg :

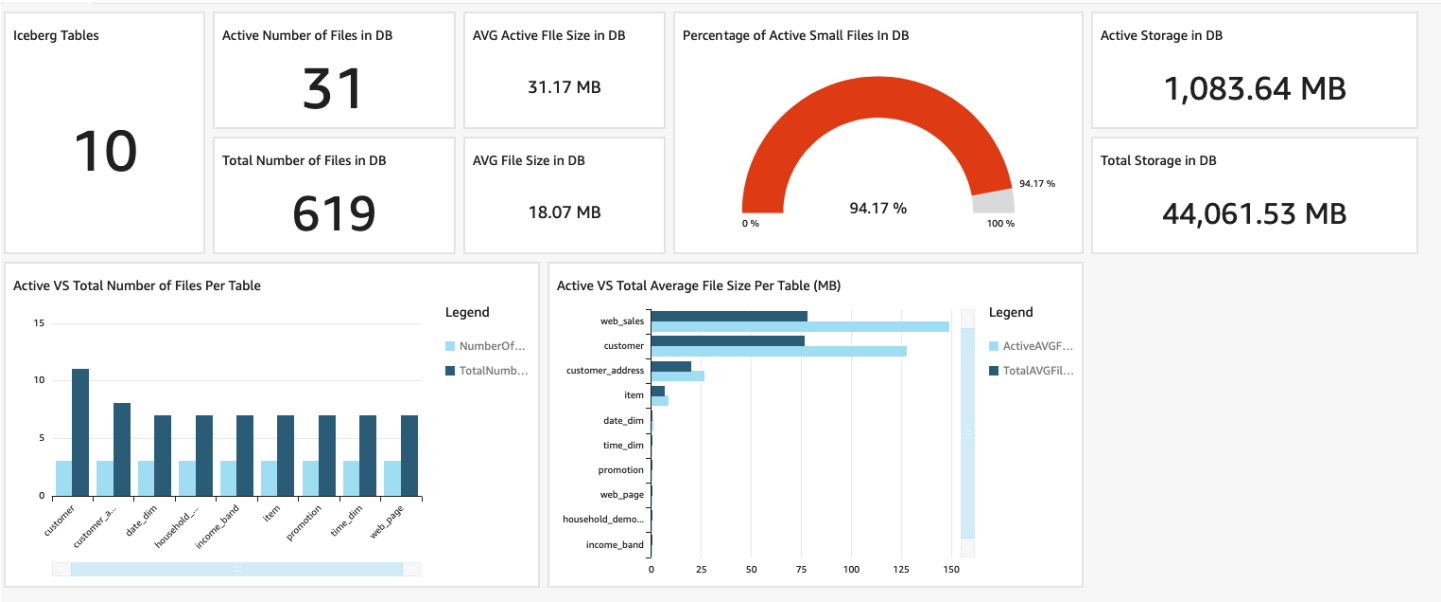
| KPI                             | Description                                                                                | Interrogation                                                                                                                                                                 |
|---------------------------------|--------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Nombre de fichiers              | Le nombre de fichiers dans la table Iceberg (pour tous les instantanés)                    | <pre>select count(*) from &lt;catalog.database. table_name&gt;.all_files</pre>                                                                                                |
| Nombre de fichiers actifs       | Le nombre de fichiers actifs dans le dernier instantané de la table Iceberg                | <pre>select count(*) from &lt;catalog.database. table_name&gt;.files</pre>                                                                                                    |
| Taille de fichier moyenne       | Taille moyenne des fichiers, en mégaoctets, pour tous les fichiers de la table Iceberg     | <pre>select avg(file_ size_in_bytes)/100 0000 from &lt;catalog.database. table_name&gt;.all_files</pre>                                                                       |
| Taille moyenne du fichier actif | Taille moyenne des fichiers, en mégaoctets, pour les fichiers actifs de la table Iceberg   | <pre>select avg(file_ size_in_bytes)/100 0000 from &lt;catalog.database. table_name&gt;.files</pre>                                                                           |
| Pourcentage de petits fichiers  | Pourcentage de fichiers actifs dont la taille est inférieure à 100 Mo                      | <pre>select cast(sum( case when file_size _in_bytes &lt; 1000000000 then 1 else 0 end)*100/ count(*) as decimal(1 0,2)) from &lt;catalog.database. table_name&gt;.files</pre> |
| Taille totale de stockage       | Taille totale de tous les fichiers du tableau, à l'exception des fichiers orphelins et des | <pre>select sum(file_ size_in_bytes)/100 0000</pre>                                                                                                                           |

| KPI                             | Description                                                                                 | Interrogation                                                                                                       |
|---------------------------------|---------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------|
|                                 | versions d'objets Amazon S3<br>(si activé)                                                  | <pre>from &lt;catalog.database.<br/>table_name&gt;.all_files</pre>                                                  |
| Taille totale du stockage actif | La taille totale de tous les<br>fichiers dans les instantanés<br>actuels d'une table donnée | <pre>select sum(file_<br/>size_in_bytes)/100<br/>0000<br/>from &lt;catalog.database.<br/>table_name&gt;.files</pre> |

Pour plus d'informations sur la création de tableaux de bord, consultez la [documentation de Quick Sight](#).

## Database-level surveillance

L'exemple suivant montre un tableau de bord de surveillance créé dans Quick Sight pour fournir une vue d'ensemble des KPI au niveau de la base de données pour une collection de tables Iceberg.



Ce tableau de bord collecte les KPI suivants :

| KPI                             | Description                                                                                                                                   | Interrogation                                                                                                                |
|---------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------|
| Nombre de fichiers              | Le nombre de fichiers dans la base de données Iceberg (pour tous les instantanés)                                                             | Ce tableau de bord utilise les requêtes au niveau des tables fournies dans la section précédente et consolide les résultats. |
| Nombre de fichiers actifs       | Le nombre de fichiers actifs dans la base de données Iceberg (basé sur les derniers instantanés des tables Iceberg)                           |                                                                                                                              |
| Taille de fichier moyenne       | Taille moyenne des fichiers, en mégaoctets, pour tous les fichiers de la base de données Iceberg                                              |                                                                                                                              |
| Taille moyenne du fichier actif | Taille moyenne des fichiers, en mégaoctets, pour tous les fichiers actifs de la base de données Iceberg                                       |                                                                                                                              |
| Pourcentage de petits fichiers  | Pourcentage de fichiers actifs dont la taille est inférieure à 100 Mo dans la base de données Iceberg                                         |                                                                                                                              |
| Taille totale du stockage       | Taille totale de tous les fichiers de la base de données, à l'exception des fichiers orphelins et des versions d'objets Amazon S3 (si activé) |                                                                                                                              |
| Taille totale du stockage actif | Taille totale de tous les fichiers dans les instantanés actuels de toutes les tables de la base de données                                    |                                                                                                                              |

## Maintenance préventive

En configurant les fonctionnalités de surveillance décrites dans les sections précédentes, vous pouvez aborder la maintenance des tables d'un point de vue préventif plutôt que réactif. Par exemple, vous pouvez utiliser les métriques au niveau de la table et au niveau de la base de données pour planifier des actions telles que les suivantes :

- Utilisez le compactage par bacs pour regrouper les petits fichiers lorsqu'une table atteint N petits fichiers.
- Utilisez le compactage par bacs pour fusionner les fichiers de suppression lorsqu'une table atteint N fichiers de suppression dans une partition donnée.
- Supprimez les petits fichiers déjà compactés en supprimant les instantanés lorsque le stockage total est X fois supérieur au stockage actif.

# Gouvernance et contrôle d'accès pour Apache Iceberg sur AWS

Apache Iceberg s'intègre AWS Lake Formation pour simplifier la gouvernance des données. Cette intégration permet aux administrateurs des lacs de données d'attribuer des autorisations d'accès au niveau des cellules aux tables Iceberg. Pour un exemple d'interrogation de tables Iceberg à l'aide d'Amazon Athena, consultez le [billet de blog Interagissez avec les tables Apache Iceberg à AWS l'aide d'Amazon Athena AWS Lake Formation et les autorisations affinées entre comptes à l'aide d'Amazon Athena à l'aide d'Amazon Athena](#). AWS Lake Formation

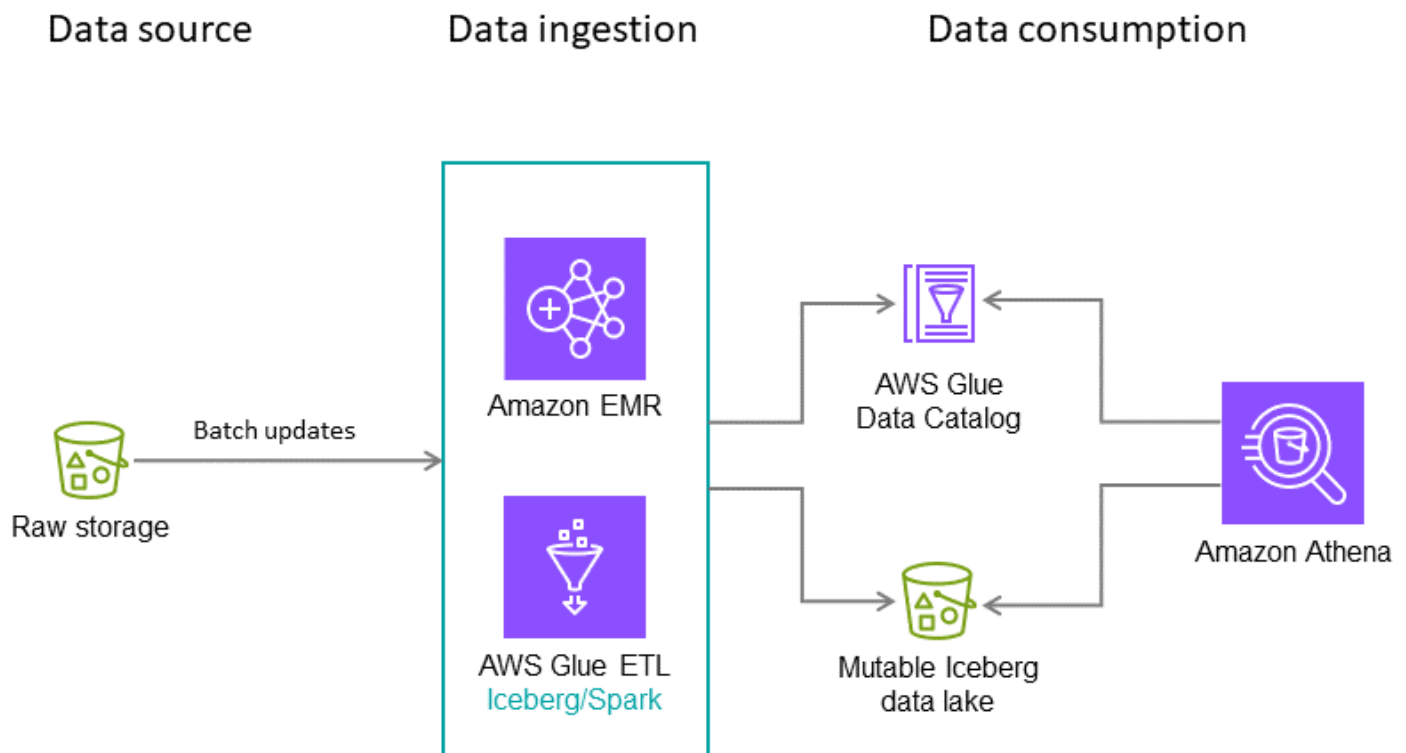
# Architectures de référence pour Apache Iceberg sur AWS

Cette section fournit des exemples de la manière d'appliquer les meilleures pratiques dans différents cas d'utilisation, tels que l'ingestion par lots et un lac de données qui combine l'ingestion de données par lots et par streaming.

## Ingestion nocturne de lots

Pour ce cas d'utilisation hypothétique, supposons que votre table Iceberg ingère les transactions par carte de crédit tous les soirs. Chaque lot contient uniquement des mises à jour incrémentielles, qui doivent être fusionnées dans la table cible. Plusieurs fois par an, des données historiques complètes sont reçues. Pour ce scénario, nous recommandons l'architecture et les configurations suivantes.

Remarque : Ceci n'est qu'un exemple. La configuration optimale dépend de vos données et de vos exigences.



### Recommandations :

- Taille du fichier : 128 Mo, car les tâches Apache Spark traitent les données par tranches de 128 Mo.

- Type d'écriture : copie sur écriture. Comme expliqué précédemment dans ce guide, cette approche permet de garantir que les données sont écrites de manière optimisée pour la lecture.
- Variables de partition : year/month /jour. Dans notre cas d'utilisation hypothétique, nous interrogeons le plus souvent les données récentes, bien que nous effectuions parfois des analyses complètes des tables des données des deux dernières années. L'objectif du partitionnement est de permettre des opérations de lecture rapides en fonction des exigences du cas d'utilisation.
- Ordre de tri : horodatage
- Catalogue de données : AWS Glue Data Catalog

## Lac de données combinant ingestion par lots et ingestion en temps quasi réel

Vous pouvez configurer un lac de données sur Amazon S3 qui partage des données par lots et en streaming entre les comptes et les régions. Pour un schéma d'architecture et des détails, consultez le billet de AWS blog [Construisez un lac de données transactionnel à l'aide d'Apache Iceberg et des partages de données entre comptes à l'aide d'Amazon AWS Lake Formation Athena](#). AWS Glue

# Ressources

- [Utilisation du framework Iceberg dans AWS Glue](#) (AWS Glue documentation)
- [Iceberg](#) (documentation Amazon EMR)
- [Utilisation des tables Apache Iceberg](#) (documentation Amazon Athena)
- [Documentation Amazon S3](#)
- [Documentation rapide](#)
- [Réplication du catalogue Glue et des autorisations de Lake Formation](#) (GitHub référentiel)
- [Documentation d'Apache Iceberg](#)
- [Documentation d'Apache Spark](#)

# Collaborateurs

Les personnes suivantes ont écrit, AWS co-écrit et révisé ce guide.

## Collaborateurs

- Stefano Sandona, architecte de solutions, Big Data
- Imtiaz (Taz) Sayed, architecte de solutions, responsable technique, analytique
- Shana Schipers, architecte de solutions, Big Data
- Prashant Singh, ingénieur en développement logiciel, Amazon EMR
- Arun A K, architecte de solutions, Big Data et ETL
- Francisco Morillo, architecte de solutions, Streaming
- Suthan Phillips, architecte d'analyse, Amazon EMR
- Sercan Karaoglu, architecte de solutions
- Yonatan Dolan, spécialiste de l'analyse
- Guy Bachar, architecte de solutions
- Sofia Zilberman, architecte de solutions, Streaming
- Dan Stair, architecte de solutions spécialisé
- Sakti Mishra, architecte de solutions
- Ron Ortloff, chef de produit principal, Amazon S3
- David Zhang, architecte de solutions, analytique

## Réviseurs

- Rick Sears, directeur général, Amazon EMR
- Linda OConnor, spécialiste des EMR sur Amazon
- Ian Meyers, directeur d'Amazon EMR
- Vinita Ananth, directrice de la gestion des produits Amazon EMR
- Jason Berkowitz, chef de produit, AWS Lake Formation
- Mahesh Mishra, chef de produit, Amazon Redshift
- Vladimir Zlatkin, responsable de l'architecture des solutions, Big Data
- Karthik Prabhakar, architecte d'analyse, Amazon EMR

- Vijay Jain, chef de produit
- Anupriti Warade, chef de produit, Amazon S3
- Ajit Tandale, architecte de solutions, données
- Gwen Chen, responsable du marketing des produits
- Noritaka Sekiyama, architecte principal, Big Data

# Historique du document

Le tableau suivant décrit les modifications importantes apportées à ce guide. Pour être averti des mises à jour à venir, abonnez-vous à un [fil RSS](#).

| Modification                         | Description                                                                                                                                                                                                       | Date             |
|--------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|
| <a href="#">Nouvelle section</a>     | Ajout d'informations sur l'utilisation <a href="#">de la version 3 de la spécification du format de table Iceberg</a> .                                                                                           | 26 novembre 2025 |
| <a href="#">Correctif</a>            | Informations corrigées concernant <code>gc.enabled</code> les paramètres dans la section relative à la <a href="#">procédure de capture instantanée</a> .                                                         | 24 octobre 2025  |
| <a href="#">Ajouts</a>               | Ajout de nouvelles sections sur l'utilisation des tables Iceberg à l'aide de <a href="#">Trino</a> et <a href="#">Pylceberg</a> extension de la section sur la <a href="#">migration des tables</a> vers Iceberg. | 12 août 2025     |
| <a href="#">Mises à jour</a>         | Informations améliorées et clarifiées tout au long du guide afin de refléter les dernières versions d' AWS Glue Amazon EMR et d'Apache Iceberg.                                                                   | 14 juillet 2025  |
| <a href="#">Ajouts</a>               | Ajout d'une <a href="#">nouvelle section</a> sur l'utilisation des tables Iceberg à l'aide d'Amazon Data Firehose.                                                                                                | 20 février 2025  |
| <a href="#">Publication initiale</a> | —                                                                                                                                                                                                                 | 30 avril 2024    |

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.