在 上使用 Apache Iceberg AWS

AWS 方案指引

AWS 方案指引: 在 上使用 Apache Iceberg AWS

Table of Contents

在 上使用 Apache Iceberg AWS

Amazon Web Services （ 貢獻者)

2025 年 11 月 (文件歷史記錄)

Apache Iceberg 是一種開放原始碼資料表格式，可簡化資料表管理，同時改善效能。Amazon EMR AWS Glue、Amazon Athena 和 Amazon Redshift 等 AWS 分析服務包含 Iceberg 的原生支援，因此您可以在 Amazon Simple Storage Service (Amazon S3) 上輕鬆建置交易資料湖 AWS。

此外，新一代 Amazon SageMaker 建置在開放式湖群架構上，該架構整合了 AWS 跨資料湖、資料倉儲以及第三方和聯合來源的資料存取。湖房與 Iceberg 完全相容，可讓您使用 Iceberg REST API 彈性存取和查詢資料。

本技術指南提供在不同 上開始使用 Iceberg 的指引 AWS 服務，並包含 AWS 大規模執行 Iceberg 的最佳實務和建議，同時最佳化成本和效能。

無論您是剛開始使用 Iceberg，還是想要最佳化現有 Iceberg 工作負載的資深使用者 AWS，本指南都會為專案的每個階段提供寶貴的洞見

在本指南中：

- 現代資料湖
- Athena SQL 中的 Iceberg 資料表入門
- 在 Amazon EMR 中使用 Iceberg
- 在 中使用 Iceberg AWS Glue
- 使用 Spark 處理 Iceberg 資料表
- 使用 Trino 處理 Iceberg 資料表
- 使用 Amazon Data Firehose 處理 Iceberg 資料表
- 使用 Athena SQL 處理 Iceberg 資料表
- 使用 PyIceberg 處理 Iceberg 資料表
- 使用 Iceberg 資料表格式規格第 3 版
- 將現有資料表遷移至 Iceberg
- 最佳化 Iceberg 工作負載的最佳實務
- 監控 Iceberg 工作負載
- 控管與存取控制

現代資料湖

現代資料湖中的進階使用案例

資料儲存的演變已從資料庫進展到資料倉儲和資料湖，其中每項技術都處理獨特的業務和資料需求。傳統資料庫擅長處理結構化資料和交易工作負載，但隨著資料量的增加，他們面臨效能挑戰。資料倉儲出現來解決效能和可擴展性問題，但與資料庫一樣，它們依賴垂直整合系統中的專屬格式。

資料湖提供在成本、可擴展性和彈性方面儲存資料的最佳選項之一。您可以使用資料湖以低成本保留大量結構化和非結構化資料，並將此資料用於不同類型的分析工作負載，從商業智慧報告到大數據處理、即時分析、機器學習和生成式人工智慧 (AI)，以協助引導更好的決策。

儘管有這些優點，但資料湖最初並非使用類似資料庫的功能進行設計。資料湖不提供原子性、一致性、隔離性和耐久性 (ACID) 處理語意的支援，您可能需要這些語意，才能透過使用許多不同的技術，在數百或數千個使用者之間大規模最佳化和管理資料。資料湖不提供下列功能的原生支援：

- 當業務中的資料變更時，執行有效的記錄層級更新和刪除
- 隨著資料表成長到數百萬個檔案和數十萬個分割區，管理查詢效能
- 確保多個並行寫入器和讀取器之間的資料一致性
- 防止寫入操作在操作中途失敗時資料損毀
- 隨著時間演進的資料表結構描述，無需 （ 部分 ） 重寫資料集

這些挑戰在處理變更資料擷取 (CDC) 等使用案例中變得特別普遍，或與隱私權、刪除資料和串流資料擷取相關的使用案例，這可能會導致次佳資料表。

使用傳統 Hive 格式資料表的資料湖僅支援整個檔案的寫入操作。這使得更新和刪除難以實作、耗時且成本高昂。此外，需要 ACID 相容系統中提供的並行控制和保證，以確保資料完整性和一致性。

這些挑戰讓使用者面臨兩難：選擇完全整合的專屬平台，或選擇廠商中立但資源密集的自建資料湖，需要持續維護和遷移才能實現其潛在價值。

為了協助克服這些挑戰，Iceberg 提供額外的類似資料庫的功能，可簡化資料湖的最佳化和管理負荷，同時仍然支援 Amazon S3 等經濟實惠系統的儲存。

Apache Iceberg 簡介

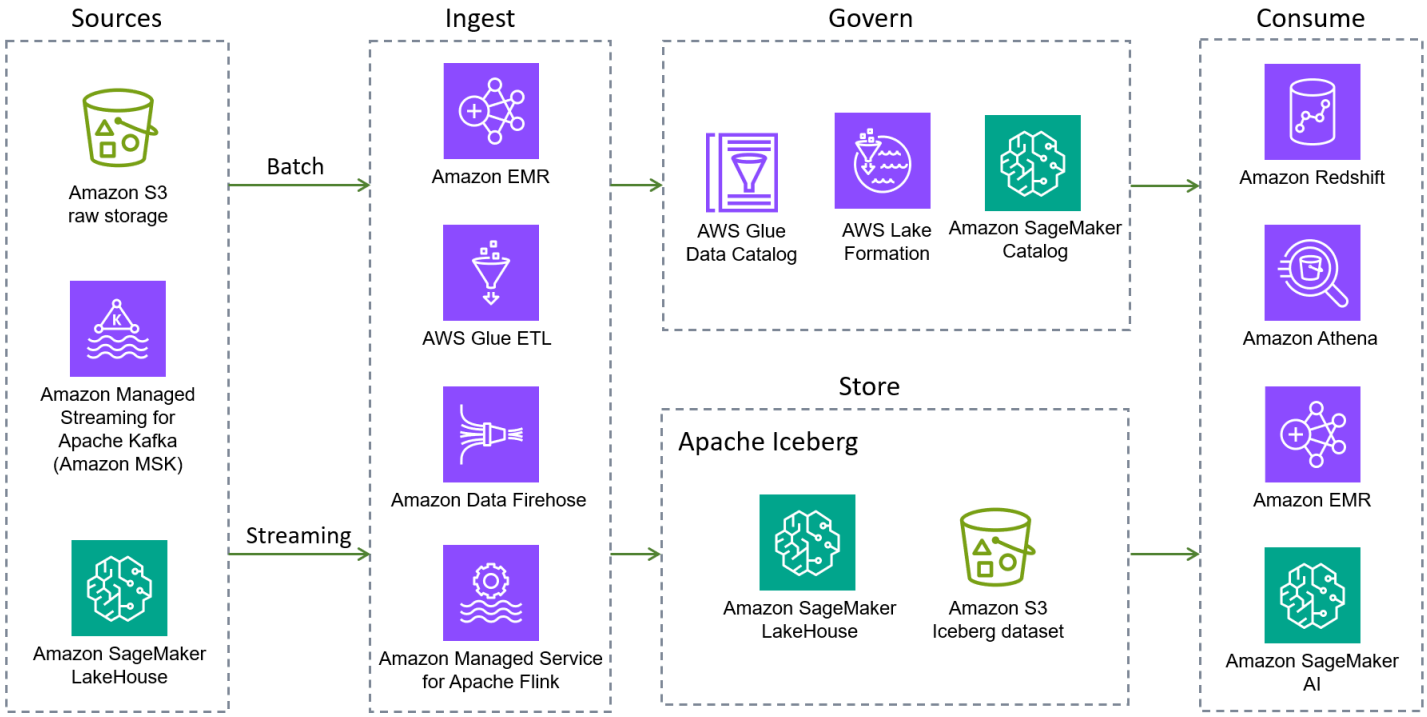
Apache Iceberg 是一種開放原始碼資料表格式，可在資料湖資料表中提供過去只能在資料庫或資料倉儲中使用的功能。它專為擴展和效能而設計，非常適合用於管理超過數百 GB 的資料表。Iceberg 資料表的一些主要功能包括：

- 刪除、更新和合併。 Iceberg 支援資料倉儲的標準 SQL 命令，可與資料湖資料表搭配使用。
- 快速掃描規劃和進階篩選。Iceberg 會存放中繼資料，例如分割區和資料欄層級統計資料，可供引擎用來加速規劃和執行查詢。
- 完整的結構描述演變。Iceberg 支援新增、捨棄、更新或重新命名資料欄，而不會產生副作用。
- 分割區演變。您可以在資料磁碟區或查詢模式變更時更新資料表的分割區配置。Iceberg 支援變更資料表分割所在的資料欄，或將資料欄新增至複合分割區或從中移除資料欄。
- 隱藏分割。 此功能可防止自動讀取不必要的分割區。這可讓使用者不必了解資料表的分割詳細資訊，或將額外的篩選條件新增至其查詢。
- 版本轉返。使用者可以還原至交易前狀態，以快速修正問題。
- 時間歷程。使用者可以查詢資料表的特定先前版本。
- 可序列化隔離。資料表變更是原子的，因此讀者永遠不會看到部分或未遞交的變更。
- 並行寫入器。Iceberg 使用樂觀並行來允許多個交易成功。如果發生衝突，其中一個寫入器必須重試交易。
- 開啟檔案格式。Iceberg 支援多種開放原始碼檔案格式，包括 Apache Parquet、Apache Avro 和 Apache ORC。

總之，使用 Iceberg 格式的資料湖受益於交易一致性、速度、擴展和結構描述演變。如需這些和其他 Iceberg 功能的詳細資訊，請參閱 Apache Iceberg 文件。

AWS 支援 Apache Iceberg

Amazon EMR、Amazon Athena、Amazon Redshift、 AWS Glue和 Amazon SageMaker AWS 服務等 支援 Apache Iceberg。下圖說明以 Iceberg 為基礎的資料湖的簡化參考架構。



以下 AWS 服務 提供原生 Iceberg 整合。還有其他 AWS 服務 可以間接或透過封裝 Iceberg 程式庫來與 Iceberg 互動。

- 由於其耐用性、可用性、可擴展性、安全性、合規性和稽核功能，Amazon S3 是建置資料湖的最佳位置。Iceberg 的設計和建置旨在與 Amazon S3 無縫互動，並支援 Iceberg 文件中列出的許多 Amazon S3 功能。此外，Amazon S3 Tables 提供第一個具有內建 Iceberg 支援的雲端物件存放區，並簡化大規模儲存表格式資料。透過支援 Iceberg 的 S3 Tables，您可以使用熱門 AWS 和第三方查詢引擎輕鬆查詢表格式資料。
- 新一代 SageMaker 建置在開放式湖群架構上，可統一跨 Amazon S3 資料湖、Amazon Redshift 資料倉儲以及第三方和聯合資料來源的資料存取。這些功能可協助您在單一資料複本上建置強大的分析和 AI/ML 應用程式。湖房與 Iceberg 完全相容，因此您可以使用 Iceberg REST API 靈活地存取和查詢資料。
- Amazon EMR 是一種大數據解決方案，可透過使用 Apache Spark、Flink、Trino 和 Hive 等開放原始碼架構來處理 PB 級資料處理、互動式分析和機器學習。Amazon EMR 可以在自訂的 Amazon Elastic Compute Cloud (Amazon EC2) 叢集、Amazon Elastic Kubernetes Service (Amazon EKS) AWS Outposts、 或 Amazon EMR Serverless 上執行。
- Amazon Athena 是一種以開放原始碼架構為基礎的無伺服器互動式分析服務。它支援開放資料表和檔案格式，並提供簡化、靈活的方法來分析其居住地的 PB 資料。Athena 為 Iceberg 的讀取、時間歷程、寫入和 DDL 查詢提供原生支援，並使用 AWS Glue Data Catalog Iceberg 中繼存放區的 。

- Amazon Redshift 是 PB 級雲端資料倉儲，支援叢集型和無伺服器部署選項。Amazon Redshift Spectrum 可以查詢向 註冊 AWS Glue Data Catalog 並存放在 Amazon S3 上的外部資料表。Redshift Spectrum 也支援 Iceberg 儲存格式。
- AWS Glue 是一種無伺服器資料整合服務，可讓您更輕鬆地探索、準備、移動和整合來自多個來源的資料，以進行分析、機器學習 (ML) 和應用程式開發。它與 Iceberg 完全整合。具體而言，您可以使用 AWS Glue 任務在 Iceberg 資料表上執行讀取和寫入操作、透過 管理資料表 AWS Glue Data Catalog(Hive 中繼存放區相容）、使用 AWS Glue 爬蟲程式自動探索和註冊資料表，以及透過 Data Quality 功能評估 Iceberg 資料表中的 AWS Glue 資料品質。 AWS Glue Data Catalog 也支援收集資料欄統計資料、計算和更新 Iceberg 資料表中每個資料欄的不同值 (NDVs) 數量，以及自動資料表最佳化 （壓縮、快照保留和孤立檔案刪除）。 AWS Glue 也支援從 AWS 服務 和第三方應用程式清單將零 ETL 整合到 Iceberg 資料表。
- Amazon Data Firehose 是一項全受管服務，可將即時串流資料交付至目的地，例如 Amazon S3、Amazon Redshift、Amazon OpenSearch Service、Amazon OpenSearch Serverless、Splunk、Apache Iceberg 資料表，以及支援的第三方服務提供者擁有的任何自訂 HTTP 或 HTTP 端點，包括 Datadog、Dynatrace、LogicMonitor、MongoDB、New Relic、Coralogix 和 Elastic。使用 Firehose，您不需要撰寫應用程式或管理資源。將您的資料產生來源設定為把資料傳送至 Firehose，它就會將資料自動交付至您指定的目的地。您也可以將 Firehose 設定為在交付資料前轉換您的資料。
- Amazon Managed Service for Apache Flink 是一種全受管的 Amazon 服務，可讓您使用 Apache Flink 應用程式來處理串流資料。它支援從 Iceberg 資料表讀取和寫入，並啟用即時資料處理和分析。
- Amazon SageMaker AI 支援使用 Iceberg 格式在 Amazon SageMaker AI Feature Store 中儲存功能集。
- AWS Lake Formation 提供存取資料的粗略和精細存取控制許可，包括 Athena 或 Amazon Redshift 耗用的 Iceberg 資料表。若要進一步了解 Iceberg 資料表的許可支援，請參閱 Lake Formation 文件。

AWS 有各種支援 Iceberg 的服務，但涵蓋所有這些服務超出本指南的範圍。下列各節涵蓋 Amazon EMR 和 AWS Glue以及 Athena SQL 上的 Spark （批次和結構化串流）。下一節提供 Athena SQL 中 Iceberg 支援的快速說明。

Amazon Athena SQL 中的 Iceberg 資料表入門

Amazon Athena 為 Iceberg 提供內建支援。您可以使用 Iceberg，無需任何其他步驟或組態，但設定 Athena 文件入門一節中詳述的服務先決條件除外。本節提供在 Athena 中建立資料表的簡介。如需詳細資訊，請參閱本指南稍後的使用 Athena SQL 來使用 Iceberg 資料表。

您可以使用不同的引擎 AWS 在 上建立 Iceberg 資料表。這些資料表可順暢運作 AWS 服務。若要使用 Athena SQL 建立第一個 Iceberg 資料表，您可以使用下列樣板程式碼。

```
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'
)
```

下列各節提供在 Athena 中建立分割和未分割 Iceberg 資料表的範例。如需詳細資訊，請參閱 Athena 文件中詳述的 Iceberg 語法。

建立未分割的資料表

下列範例陳述式會自訂樣板 SQL 程式碼，以在 Athena 中建立未分割的 Iceberg 資料表。您可以將此陳述式新增至 Athena 主控台中的查詢編輯器，以建立資料表。

```
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'
)
```

如需使用查詢編輯器的step-by-step說明，請參閱 Athena 文件中的入門。

建立分割的資料表

下列陳述式會使用 Iceberg 的隱藏分割概念，根據日期建立分割資料表。它會使用day()轉換，從時間戳記資料欄使用 dd-mm-yyyy 格式衍生每日分割區。Iceberg 不會將此值儲存為資料集中的新資料欄。相反地，當您寫入或查詢資料時， 值會即時衍生。

```
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'
)
```

使用單一 CTAS 陳述式建立資料表並載入資料

在前幾節的分割和未分割範例中，Iceberg 資料表會建立為 空白資料表。您可以使用 INSERT 或 MERGE陳述式將資料載入資料表。或者，您可以使用 CREATE TABLE AS SELECT (CTAS)陳述式，在單一步驟中建立資料並將其載入 Iceberg 資料表。

CTAS 是 Athena 在單一陳述式中建立資料表和載入資料的最佳方式。下列範例說明如何使用 CTAS 從 Athena 中現有的 Hive/Parquet 資料表 (iceberg_ctas_table) 建立 Iceberg 資料表 (hive_table)。

```
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
```

若要進一步了解 CTAS，請參閱 Athena CTAS 文件。

插入、更新和刪除資料

Athena 支援使用 INSERT INTO、MERGE INTO、 和 DELETE FROM 陳述式將資料寫入 Iceberg UPDATE資料表的不同方式。

> **Note**
>
> Athena SQL 目前不支援copy-on-write方法。UPDATE、 MERGE INTO和 DELETE FROM操作一律使用具有位置刪除的merge-on-read方法，無論指定的資料表屬性為何。如果您設定 write.update.mode、 write.merge.mode和 等資料表屬性write.delete.mode來使用copy-on-write，您的查詢將不會失敗，但 Athena 會忽略它們並繼續使用merge-on-read。

下列陳述式使用 INSERT INTO將資料新增至 Iceberg 資料表：

```
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;
```

輸出範例：

如需詳細資訊，請參閱 Athena 文件。

查詢 Iceberg 資料表

您可以使用 Athena SQL 對 Iceberg 資料表執行定期 SQL 查詢，如先前範例所示。

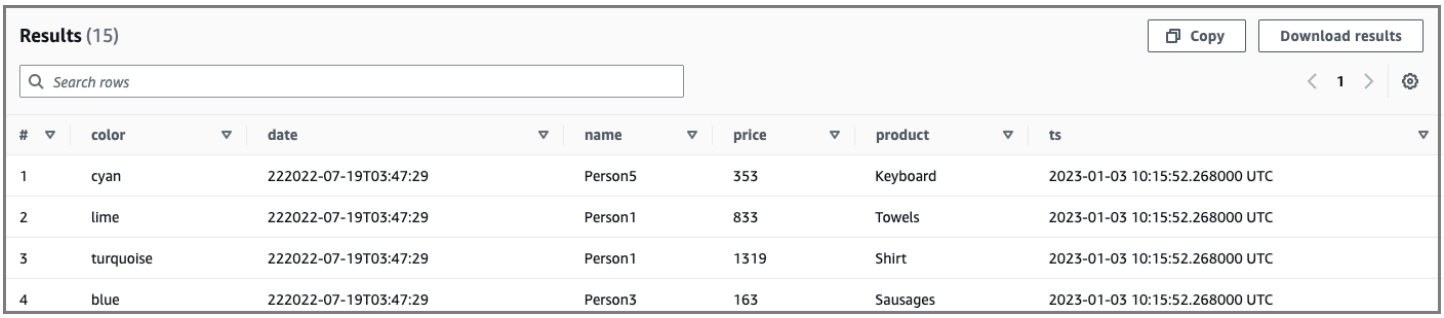
除了一般查詢之外，Athena 還支援 Iceberg 資料表的時間歷程查詢。如前所述，您可以透過 Iceberg 資料表中的更新或刪除來變更現有記錄，因此根據時間戳記或快照 ID，使用時間歷程查詢來查看資料表的較舊版本相當方便。

例如，下列陳述式會更新 的顏色值Person5，然後顯示 2023 年 1 月 4 日起的舊值：

```
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'
```

輸出範例：



Results (15)

Copy | Download results

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

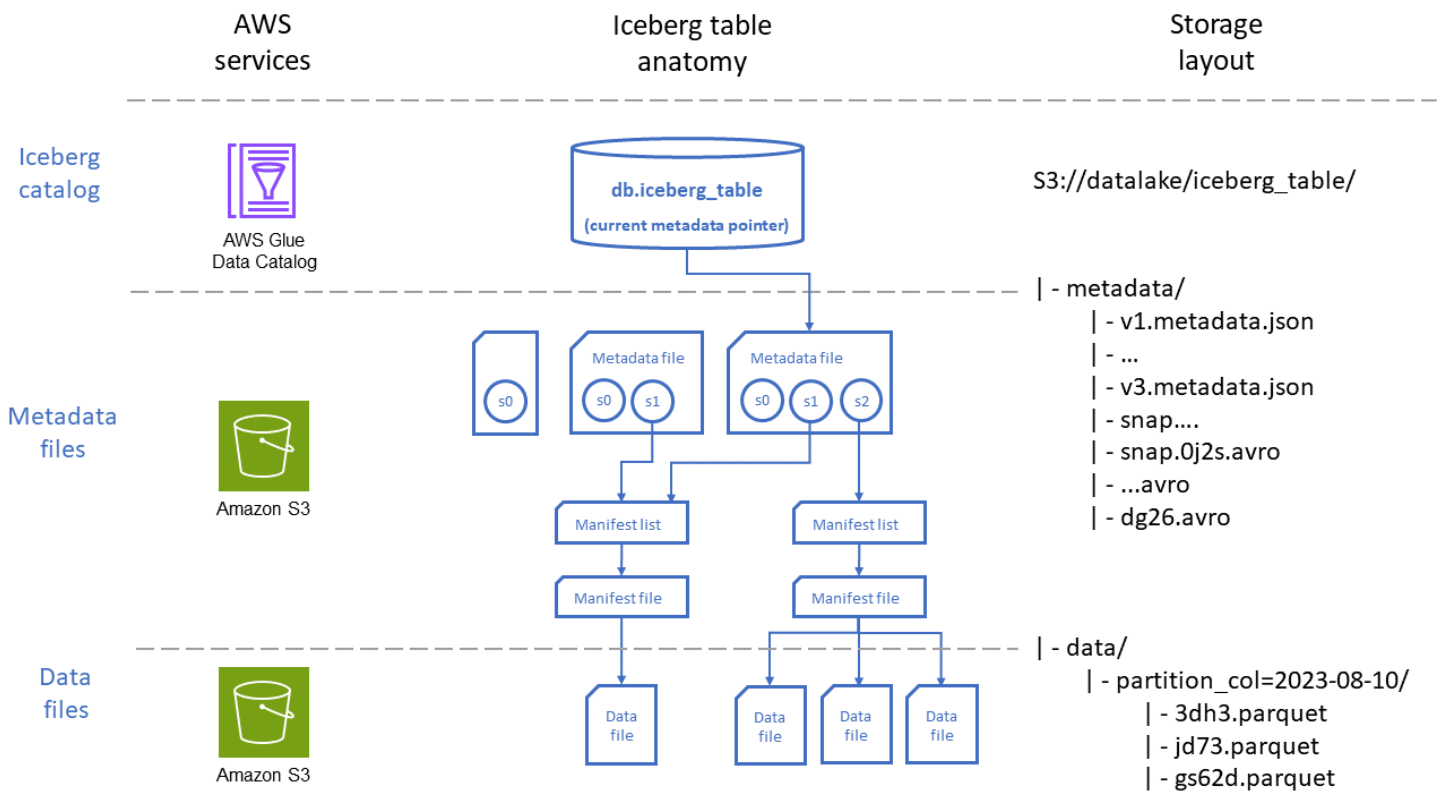
如需時間歷程查詢的語法和其他範例，請參閱 Athena 文件。

Iceberg 資料表結構

現在我們已經介紹了使用 Iceberg 資料表的基本步驟，讓我們深入了解 Iceberg 資料表的複雜細節和設計。

為了啟用本指南先前所述的功能，Iceberg 的設計採用階層式資料層和中繼資料檔案。這些層以智慧方式管理中繼資料，以最佳化查詢規劃和執行。

下圖透過兩個角度描繪 Iceberg 資料表的組織： AWS 服務 用於在 Amazon S3 中存放資料表和檔案放置的 。



如圖所示，Iceberg 資料表由三個主要層組成：

- Iceberg 目錄：與 Iceberg 原生 AWS Glue Data Catalog 整合，對於大多數使用案例而言， 是執行之工作負載的最佳選項 AWS。與 Iceberg 資料表 （ 例如 Athena) 互動的服務會使用 目錄來尋找資料表的目前快照版本，以讀取或寫入資料。
- 中繼資料層：中繼資料檔案，即資訊清單檔案和資訊清單檔案，追蹤資訊，例如資料表的結構描述、分割區策略和資料檔案的位置，以及資料欄層級統計資料，例如儲存在每個資料檔案中記錄的最小和最大範圍。這些中繼資料檔案存放在資料表路徑中的 Amazon S3 中。
  - 資訊清單檔案包含每個資料檔案的記錄，包括其位置、格式、大小、檢查總和和其他相關資訊。
  - 資訊清單清單提供資訊清單檔案的索引。隨著資訊清單檔案的數量在資料表中成長，將該資訊分成較小的子區段有助於減少查詢需要掃描的資訊清單檔案數量。
  - 中繼資料檔案包含整個 Iceberg 資料表的相關資訊，包括資訊清單清單、結構描述、分割區中繼資料、快照檔案，以及用於管理資料表中繼資料的其他檔案。
- 資料層：此層包含具有將執行查詢之資料記錄的檔案。這些檔案可以以不同的格式儲存，包括 Apache Parquet、Apache Avro 和 Apache ORC。
  - 資料檔案包含資料表的資料記錄。
  - 刪除在 Iceberg 資料表中編碼資料列層級刪除和更新操作的檔案。Iceberg 有兩種類型的刪除檔案，如 Iceberg 文件所述。這些檔案是由 操作使用merge-on-read模式建立。

在 Amazon EMR 中使用 Iceberg

Amazon EMR 使用 Apache Spark、Apache Hive、Flink 和 Trino 等開放原始碼架構，在雲端提供 PB 級資料處理、互動式分析和機器學習。

> **Note**
>
> 本指南使用 Apache Spark 做為範例。

Amazon EMR 支援多個部署選項：Amazon EMR on EC2、Amazon EMR on EKS、Amazon EMR Serverless 和 Amazon EMR on AWS Outposts。若要為您的工作負載選擇部署選項，請參閱 Amazon EMR 常見問答集。

版本和功能相容性

Amazon EMR 6.5.0 版和更新版本原生支援 Apache Iceberg。如需每個 Amazon EMR 發行版本的支援 Iceberg 版本清單，請參閱 Amazon EMR 文件中的 Iceberg 發行歷史記錄。另請參閱使用叢集搭配 Iceberg 下的章節，了解 Amazon EMR 在不同架構上支援哪些 Iceberg 功能。

我們建議您使用最新的 Amazon EMR 版本，以受益於最新的支援 Iceberg 版本。本節中的程式碼範例和組態假設您使用的是 Amazon EMR 發行版本 emr-7.8.0。

使用 Iceberg 建立 Amazon EMR 叢集

若要在已安裝 Iceberg 的 Amazon EC2 上建立 Amazon EMR 叢集，請遵循 Amazon EMR 文件中的指示。

具體而言，您的叢集應該設定下列分類：

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

您也可以選擇使用 Amazon EMR Serverless 或 Amazon EMR on EKS 做為 Iceberg 工作負載的部署選項，從 Amazon EMR 6.6.0 開始。

在 Amazon EMR 中開發 Iceberg 應用程式

若要為您的 Iceberg 應用程式開發 Spark 程式碼，您可以使用 Amazon EMR Studio，這是適用於在 Amazon EMR 叢集上執行之全受管 Jupyter 筆記本的 Web 整合開發環境 (IDE)。

使用 Amazon EMR Studio 筆記本

您可以在 Amazon EMR Studio 工作區筆記本中以互動方式開發 Spark 應用程式，並將這些筆記本連接到 EC2 叢集上的 Amazon EMR 或 Amazon EMR on EKS 受管端點。如需設定適用於 Amazon EMR on EC2 和 Amazon EMR on EKS 的 EMR Studio 的說明，請參閱 AWS 服務 文件。

若要在 EMR Studio 中使用 Iceberg，請遵循下列步驟：

1. 啟動已啟用 Iceberg 的 Amazon EMR 叢集，如使用已安裝 Iceberg 的叢集所述。
2. 設定 EMR Studio。如需說明，請參閱設定 Amazon EMR Studio。
3. 開啟 EMR Studio 工作區筆記本，並執行下列程式碼做為筆記本中的第一個儲存格，以設定 Spark 工作階段以使用 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"
    }
}
```

其中：

- `<catalog_name>` 是您的 Iceberg Spark 工作階段目錄名稱。將其取代為您選擇的名稱，並記得在所有與此目錄相關聯的組態中變更參考。在程式碼中，您可以使用完整資料表名稱來參考 Iceberg 資料表，包括 Spark 工作階段目錄名稱，如下所示：

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

或者，您可以將預設目錄變更為您透過將 設定為目錄名稱定義的 Iceberg `spark.sql.defaultCatalog` 目錄。這種第二種方法可讓您參考沒有目錄字首的資料表，這可以簡化您的查詢。

- `<catalog_name>.warehouse` 會指向您要存放資料和中繼資料的 Amazon S3 路徑。
- 若要將目錄設為 AWS Glue Data Catalog，請將 `spark.sql.catalog.<catalog_name>.type`設定為 `glue`。需要此金鑰才能指向任何自訂目錄實作的實作類別。本指南稍後的一般最佳實務章節說明不同的 Iceberg 支援的目錄。

4. 您現在可以開始在筆記本中以互動方式開發適用於 Iceberg 的 Spark 應用程式，如同任何其他 Spark 應用程式一樣。

如需使用 Amazon EMR Studio 設定 Spark for Apache Iceberg 的詳細資訊，請參閱部落格文章在 Amazon EMR 上使用 Apache Iceberg 建置高效能、符合 ACID 規範、不斷發展的資料湖。

在 Amazon EMR 中執行 Iceberg 任務

開發 Iceberg 工作負載的 Spark 應用程式程式碼後，您可以在支援 Iceberg 的任何 Amazon EMR 部署選項上執行它（請參閱 Amazon EMR 常見問答集)。

如同其他 Spark 任務，您可以透過新增步驟或以互動方式將 Spark 任務提交至主節點，將工作提交至 EC2 叢集上的 Amazon EMR。若要執行 Spark 任務，請參閱下列 Amazon EMR 文件頁面：

- 如需將工作提交至 EC2 叢集上 Amazon EMR 的不同選項概觀，以及每個選項的詳細指示，請參閱將工作提交至叢集。
- 對於 Amazon EMR on EKS，請參閱使用 StartJobRun 執行 Spark 任務。
- 對於 EMR Serverless，請參閱執行中任務。

以下各節提供每個 Amazon EMR 部署選項的範例。

EC2 上的 Amazon EMR

您可以使用下列步驟來提交 Iceberg Spark 任務：

1. 在工作站上使用`emr_step_iceberg.json`下列內容建立 檔案：

```
[{
    "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. 透過自訂以粗體反白顯示的 Iceberg 組態選項，修改特定 Spark 任務的組態檔案。
3. 使用 AWS Command Line Interface (AWS CLI) 提交步驟。在 emr_step_iceberg.json 檔案所在的目錄中執行 命令。

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

Amazon EMR Serverless

若要使用 將 Iceberg Spark 任務提交至 EMR Serverless AWS CLI：

1. 在工作站上使用emr_serverless_iceberg.json下列內容建立 檔案：

```
{
    "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.AWS
            }
        }],
        "monitoringConfiguration": {
            "s3MonitoringConfiguration": {
                "logUri": "s3://YOUR-BUCKET-NAME/emr-serverless/logs/"
            }
        }
    }
}
```

2. 透過自訂以粗體反白顯示的 Iceberg 組態選項，修改特定 Spark 任務的組態檔案。
3. 使用 提交任務 AWS CLI。在 `emr_serverless_iceberg.json` 檔案所在的目錄中執行 命令：

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

若要使用 EMR Studio 主控台將 Iceberg Spark 任務提交至 EMR Serverless：

1. 請遵循 EMR Serverless 文件中的指示。
2. 對於任務組態，請使用為 提供的 Spark 的 Iceberg 組態， AWS CLI 並自訂 Iceberg 的反白欄位。如需詳細說明，請參閱 Amazon EMR 文件中的將 Apache Iceberg 與 EMR Serverless 搭配使用。

Amazon EMR on EKS

若要使用 將 Iceberg Spark 任務提交至 Amazon EMR on EKS AWS CLI：

1. 在工作站上使用`emr_eks_iceberg.json`下列內容建立 檔案：

```
{
    "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. 透過自訂以粗體反白顯示的 Iceberg 組態選項，修改 Spark 任務的組態檔案。

3. 使用 提交任務 AWS CLI。在 `emr_eks_iceberg.json` 檔案所在的目錄中執行下列命令：

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

如需詳細說明，請參閱《Amazon EMR on EKS 文件》中的將 Apache Iceberg 與 Amazon EMR on EKS 搭配使用。

Amazon EMR 的最佳實務

本節提供在 Amazon EMR 中調校 Spark 任務的一般準則，以最佳化讀取和寫入資料至 Iceberg 資料表。如需 Iceberg 特定的最佳實務，請參閱本指南稍後的最佳實務一節。

- 使用最新版本的 Amazon EMR – Amazon EMR 隨 Amazon EMR Spark 執行期立即提供 Spark 最佳化。 AWS 會在每次新版本中改善 Spark 執行期引擎的效能。
- 判斷 Spark 工作負載的最佳基礎設施 – Spark 工作負載可能需要不同類型的硬體，才能確保最佳效能。Amazon EMR 支援多種執行個體類型 （例如運算最佳化、記憶體最佳化、一般用途和儲存最佳化），以涵蓋所有類型的處理需求。當您加入新的工作負載時，我們建議您使用 M5 或 M6g 等一般執行個體類型進行基準測試。從 Ganglia 和 Amazon CloudWatch 監控作業系統 (OS) 和 YARN 指標，以判斷尖峰負載時的系統瓶頸 (CPU、記憶體、儲存體和 I/O)，並選擇適當的硬體。
- 調校 `spark.sql.shuffle.partitions` – 將 `spark.sql.shuffle.partitions` 屬性設定為叢集中的虛擬核心 (vCores) 總數或該值的倍數 （通常為 vCores總數的 1 到 2 倍）。當您使用雜湊和範圍分割做為寫入分佈模式時，此設定會影響 Spark 的平行處理。它會在寫入之前請求隨機播放來組織資料，以確保分割區對齊。
- 啟用受管擴展 – 對於幾乎所有使用案例，我們建議您啟用受管擴展和動態配置。不過，如果您的工作負載具有可預測的模式，我們建議您停用自動擴展和動態配置。啟用受管擴展時，建議您使用 Spot 執行個體來降低成本。針對任務節點使用 Spot 執行個體，而非核心節點或主節點。當您使用 Spot 執行個體時，請使用每個機群具有多個執行個體類型的執行個體機群，以確保 Spot 可用性。
- 盡可能使用廣播聯結 – 廣播 (mapside) 聯結是最理想的聯結，只要其中一個資料表夠小，足以容納您最小節點的記憶體 （依 MBs順序），而且您正在執行等式 (=) 聯結。支援除了完整外部聯結以外的所有聯結類型。廣播聯結會將較小的資料表廣播為記憶體中所有工作者節點的雜湊資料表。小型資料表廣播之後，您就無法對其進行變更。由於雜湊資料表位於 Java 虛擬機器 (JVM) 中的本機，因此可以使用雜湊聯結，根據聯結條件輕鬆與大型資料表合併。廣播聯結可提供高效能，因為隨機播放額外負荷最少。
- 調校垃圾收集器 – 如果垃圾收集 (GC) 週期緩慢，請考慮從預設平行垃圾收集器切換到 G1GC，以獲得更好的效能。若要最佳化 GC 效能，您可以微調 GC 參數。若要追蹤 GC 效能，您可以使用 Spark UI 進行監控。理想情況下，GC 時間應小於或等於總任務執行時間的 1%。

在 中使用 Iceberg AWS Glue

AWS Glue 是一種無伺服器資料整合服務，可讓您更輕鬆地探索、準備、移動和整合來自多個來源的資料，以進行分析、機器學習 (ML) 和應用程式開發。的核心功能之一 AWS Glue 是能夠以簡單且符合成本效益的方式執行擷取、轉換和載入 (ETL) 操作。這有助於分類資料、清理資料、充實資料，並在各種資料存放區和資料串流之間可靠地移動資料。

AWS Glue 任務使用 Apache Sparkor Python 執行期來封裝定義轉換邏輯 AWS Glue 的指令碼。任務可以在批次和串流模式下執行。

當您在 中建立 Iceberg 任務時 AWS Glue，根據 的版本 AWS Glue，您可以使用原生 Iceberg 整合或自訂 Iceberg 版本將 Iceberg 相依性連接到任務。

使用原生 Iceberg 整合

AWS Glue 3.0、4.0 和 5.0 版原生支援 AWS Glue 適用於 Spark 的 Apache Iceberg、Apache Hudi 和 Linux Foundation Delta Lake 等交易資料湖格式。此整合功能可簡化在 中開始使用這些架構所需的組態步驟 AWS Glue。

若要為您的 AWS Glue 任務啟用 Iceberg 支援，請設定任務：選擇 AWS Glue 任務的任務詳細資訊索引標籤、捲動至進階屬性下的任務參數，並將金鑰設定為 `--datalake-formats` ，並將值設定為 `iceberg`。

如果您使用筆記本撰寫任務，則可以使用`%%configure`魔法在第一個筆記本儲存格中設定 參數，如下所示：

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

`--datalake-formats` 中的 AWS Glue `iceberg`組態會根據您的版本對應至特定 Iceberg AWS Glue 版本：

AWS Glue 版本	預設 Iceberg 版本
5.0	1.7.1

AWS Glue 版本	預設 Iceberg 版本
4.0	1.0.0
3.0	0.13.1

使用自訂 Iceberg 版本

在某些情況下，您可能想要保留對任務 Iceberg 版本的控制權，並依照自己的步調進行升級。例如，升級至更新版本可以解鎖對新功能和效能增強功能的存取。若要搭配 使用特定的 Iceberg 版本 AWS Glue，您可以提供自己的 JAR 檔案。

實作自訂 Iceberg 版本之前，請先檢查文件的 AWS Glue AWS Glue 版本區段，以確認與 AWS Glue 環境的相容性。例如， AWS Glue 5.0 需要與 Spark 3.5.4 相容。

例如，若要執行使用 Iceberg 1.9.1 版 AWS Glue 的任務，請遵循下列步驟：

1. 取得必要的 JAR 檔案並將其上傳至 Amazon S3：
   a. 從 Apache Maven 儲存庫下載 iceberg-spark-runtime-3.5_2.12-1.9.1.jar 和 iceberg-aws-bundle-1.9.1.jar。
   b. 將這些檔案上傳至您指定的 S3 儲存貯體位置 （ 例如 `s3://your-bucket-name/jars/`）。
2. 為您的任務設定 AWS Glue 任務參數，如下所示：
   a. 在 `--extra-jars` 參數中指定兩個 JAR 檔案的完整 S3 路徑，以逗號分隔 （ 例如，`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. 請勿包括 `iceberg` 作為 `--datalake-formats` 參數的值。
   c. 如果您使用 AWS Glue 5.0，則必須將 `--user-jars-first` 參數設定為 `true`。

中 Iceberg 的 Spark 組態 AWS Glue

本節討論為 Iceberg 資料集撰寫 AWS Glue ETL 任務所需的 Spark 組態。您可以使用 `--conf` Spark 金鑰搭配所有 Spark 組態金鑰和值的逗號分隔清單來設定這些組態。您可以在筆記本或 AWS Glue Studio 主控台的任務參數區段中使用`%%configure`魔術。

```
%glue_version 5.0
```

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

使用下列屬性設定 Spark 工作階段：

- <catalog_name> 是 Iceberg Spark 工作階段目錄名稱的名稱。將其取代為您選擇的名稱，並記得在所有與此目錄相關聯的組態中變更參考。在程式碼中，您可以使用完整資料表名稱來參考 Iceberg 資料表，包括 Spark 工作階段目錄名稱，如下所示：

  <catalog_name>.<database_name>.<table_name>

  或者，您可以將 設定為目錄名稱，將預設目錄變更為您定義的 Iceberg spark.sql.defaultCatalog 目錄。您可以使用此第二個方法參考沒有目錄字首的資料表，這可以簡化您的查詢。
- <catalog_name>.<warehouse> 會指向您要存放資料和中繼資料的 Amazon S3 路徑。
- 若要將目錄設為 AWS Glue Data Catalog，請將 spark.sql.catalog.<catalog_name>.type設定為 glue。需要此金鑰才能指向任何自訂目錄實作的實作類別。如需 Iceberg 支援的目錄，請參閱本指南稍後的一般最佳實務一節。

例如，如果您有名為 的目錄glue_iceberg，您可以使用多個--conf金鑰來設定任務，如下所示：

```
%%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"
}
```

或者，您可以使用程式碼將上述組態新增至 Spark 指令碼，如下所示：

```
spark = SparkSession.builder\

 .config("spark.sql.extensions","org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensi
```

```
                    .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()
```

AWS Glue 任務的最佳實務

本節提供在 中調校 Spark 任務的一般準則 AWS Glue ，以最佳化讀取和寫入資料至 Iceberg 資料表。如需 Iceberg 特定的最佳實務，請參閱本指南稍後的最佳實務一節。

- 盡可能使用最新版本的 AWS Glue 和升級 – 新版本的 AWS Glue 提供效能改善、縮短啟動時間和新功能。它們也支援較新的 Spark 版本，這是最新 Iceberg 版本可能需要的版本。如需可用 AWS Glue 版本及其支援的 Spark 版本清單，請參閱 AWS Glue 文件。
- 最佳化 AWS Glue 任務記憶體 – 遵循 AWS 部落格文章中的建議 在 中最佳化記憶體管理 AWS Glue。
- Use AWS Glue Auto Scaling – 當您啟用 Auto Scaling 時， 會根據工作負載自動動態 AWS Glue 調整 AWS Glue 工作者數量。這有助於降低尖峰負載期間 AWS Glue 的任務成本，因為 AWS Glue 當工作負載很小且工作者閒置時， 會縮減工作者數量。若要使用 AWS Glue Auto Scaling，您可以指定任務 AWS Glue 可以擴展的工作者數量上限。如需詳細資訊，請參閱 文件中的 AWS Glue 使用 的自動擴展AWS Glue 。
- 使用所需的 Iceberg 版本 – Iceberg 的 AWS Glue 原生整合最適合開始使用 Iceberg。不過，對於生產工作負載，我們建議您新增程式庫相依性（如本指南先前所述），以完全控制 Iceberg 版本。此方法可協助您從任務中的最新 Iceberg AWS Glue 功能和效能改進中獲益。
- 啟用 Spark UI 進行監控和偵錯 – 您也可以在 中使用 Spark UI AWS Glue 來檢查 Iceberg 任務，方法是將 Spark 任務的不同階段視覺化為導向無環圖 (DAG)，並詳細監控任務。Spark UI 提供有效的方法來疑難排解和最佳化 Iceberg 任務。例如，您可以識別具有大型隨機播放或磁碟溢出的瓶頸階段，以識別調校機會。如需詳細資訊，請參閱 文件中的 AWS Glue 使用 Apache Spark Web UI 監控任務。

使用 Apache Spark 處理 Iceberg 資料表

本節提供使用 Apache Spark 與 Iceberg 資料表互動的概觀。這些範例是可在 Amazon EMR 或 上執行的樣板程式碼 AWS Glue。

注意：與 Iceberg 資料表互動的主要界面是 SQL，因此大多數範例會將 Spark SQL 與 DataFrames API 結合。

建立和寫入 Iceberg 資料表

您可以使用 Spark SQL 和 Spark DataFrames 來建立資料，並將資料新增至 Iceberg 資料表。

使用 Spark SQL

若要寫入 Iceberg 資料集，請使用標準 Spark SQL 陳述式，例如 `CREATE TABLE`和 `INSERT INTO`。

未分割的資料表

以下是使用 Spark SQL 建立未分割 Iceberg 資料表的範例：

```
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')
""")
```

若要將資料插入至未分割的資料表，請使用標準`INSERT INTO`陳述式：

```
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
```

```
""")
```

分割的資料表

以下是使用 Spark SQL 建立分割 Iceberg 資料表的範例：

```
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')
""")
```

若要使用 Spark SQL 將資料插入分割的 Iceberg 資料表，請使用標準`INSERT INTO`陳述式：

```
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**
>
> 從 Iceberg 1.5.0 開始，當您將資料插入分割資料表時，`hash`寫入分佈模式為預設值。如需詳細資訊，請參閱 Iceberg 文件中的撰寫分佈模式。

使用 DataFrames API

若要寫入 Iceberg 資料集，您可以使用 `DataFrameWriterV2` API。

若要建立 Iceberg 資料表並將資料寫入其中，請使用 `df.writeTo(t)` 函數。如果資料表存在，請使用 `.append()`函數。如果沒有，請使用`.create()`.下列範例使用 `.createOrReplace()`，這是`.create()`相當於 的變體`CREATE OR REPLACE TABLE AS SELECT`。

未分割的資料表

若要使用 DataFrameWriterV2 API 建立和填入未分割的 Iceberg 資料表：

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

若要使用 DataFrameWriterV2 API 將資料插入 現有的未分割 Iceberg 資料表：

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

分割的資料表

若要使用 DataFrameWriterV2 API 建立和填入分割的 Iceberg 資料表：

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

若要使用 DataFrameWriterV2 API 將資料插入分割的 Iceberg 資料表：

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

更新 Iceberg 資料表中的資料

下列範例顯示如何更新 Iceberg 資料表中的資料。此範例會修改c_customer_sk資料欄中具有偶數的所有資料列。

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

此操作使用預設copy-on-write策略，因此會重寫所有受影響的資料檔案。

在 Iceberg 資料表中備份資料

支援資料是指在單一交易中插入新的資料記錄和更新現有的資料記錄。若要將資料升級到 Iceberg 資料表，請使用 SQL MERGE INTO陳述式。

下列範例會在資料表 內 upsert 資料表 {UPSERT_TABLE_NAME} 的內容{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 *
""")
```

- 如果 中{UPSERT_TABLE_NAME}已存在{TABLE_NAME}具有相同 的客戶記錄c_customer_id，則{UPSERT_TABLE_NAME}記錄c_email_address值會覆寫現有的值 （更新操作）。
- 如果 中的客戶記錄{UPSERT_TABLE_NAME}不存在於 中{TABLE_NAME}，則{UPSERT_TABLE_NAME}記錄會新增至 {TABLE_NAME}（插入操作）。

刪除 Iceberg 資料表中的資料

若要從 Iceberg 資料表刪除資料，請使用 DELETE FROM運算式，並指定符合要刪除資料列的篩選條件。

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

如果篩選條件符合整個分割區，Iceberg 會執行僅限中繼資料的刪除，並保留資料檔案。否則，它只會重寫受影響的資料檔案。

刪除方法會取得受 WHERE子句影響的資料檔案，並建立不含已刪除記錄的複本。然後，它會建立新的資料表快照，指向新的資料檔案。因此，已刪除的記錄仍然存在於資料表的較舊快照中。例如，如果您擷取資料表的上一個快照，您會看到您剛刪除的資料。如需有關使用相關資料檔案移除不需要的舊快照以進行清除的資訊，請參閱本指南稍後使用壓縮來維護檔案一節。

讀取資料

您可以使用 Spark SQL 和 DataFrames 在 Spark 中讀取 Iceberg 資料表的最新狀態。

使用 Spark SQL 的範例：

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

使用 DataFrames API 的範例：

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

使用時間歷程

Iceberg 資料表中的每個寫入操作（插入、更新、upsert、刪除）都會建立新的快照。然後，您可以將這些快照用於時間歷程，以返回時間並檢查過去資料表的狀態。

如需如何使用 `snapshot-id`和計時值擷取資料表快照歷史記錄的資訊，請參閱本指南稍後的存取中繼資料一節。

下列時間歷程查詢會根據特定 顯示資料表的狀態 `snapshot-id`。

使用 Spark SQL：

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

使用 DataFrames API：

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

下列時間歷程查詢會根據在特定時間戳記之前建立的最後一個快照顯示資料表的狀態，以毫秒為單位 (`as-of-timestamp`)。

使用 Spark SQL：

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

使用 DataFrames API：

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

使用增量查詢

您也可以使用 Iceberg 快照逐步讀取附加的資料。

注意： 目前，此操作支援從append快照讀取資料。它不支援從 replace、 overwrite或 等操作擷取資料delete。 此外，Spark SQL 語法不支援增量讀取操作。

下列範例會擷取附加至快照 start-snapshot-id（獨佔） 和 end-snapshot-id（包含） 之間 Iceberg 資料表的所有記錄。

```
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}")
)
```

存取中繼資料

Iceberg 可透過 SQL 存取其中繼資料。您可以查詢命名空間 來存取任何指定資料表 (<table_name>) 的中繼資料<table_name>.<metadata_table>。 如需中繼資料資料表的完整清單，請參閱 Iceberg 文件中的檢查資料表。

下列範例顯示如何存取 Iceberg 歷史記錄中繼資料表，其中顯示 Iceberg 資料表的遞交（變更）歷史記錄。

從 Amazon EMR Studio 筆記本使用 Spark SQL（搭配 %%sql魔術）：

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

使用 DataFrames API：

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

輸出範例：

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

使用 Trino 處理 Iceberg 資料表

本節說明如何在 Amazon EMR 上使用 Trino 來設定和操作 Iceberg 資料表。這些範例是樣板程式碼，您可以在 Amazon EMR on EC2 叢集上執行。本節中的程式碼範例和組態假設您使用的是 Amazon EMR 發行版本 emr-7.9.0。

EC2 上的 Amazon EMR 設定

1. 使用下列內容建立 `iceberg.properties` 檔案。如果未在 `CREATE TABLE`陳述式中明確指定格式，此`iceberg.file-format=parquet`設定會決定新資料表的預設儲存格式。

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

2. 將 `iceberg.properties` 檔案上傳至 S3 儲存貯體。
3. 建立從 S3 儲存貯體複製`iceberg.properties` 檔案的引導操作，並將其儲存為您要建立的 Amazon EMR 叢集上的 Trino 組態檔案。 請務必<S3-bucket-name> 將 取代為您的 S3 儲存貯體名稱。

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

4. 建立已安裝 Trino 的 Amazon EMR 叢集，並將先前指令碼的執行指定為引導操作。以下是用於建立叢集的 sample AWS Command Line Interface (AWS CLI) 命令：

```
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>"}'
```

您取代的位置：

- <S3-bucket-name> 使用您的 S3 儲存貯體名稱
- <region> 您的特定 AWS 區域
- <key-name> 您的金鑰對。如果金鑰對不存在，則會建立金鑰對。
- <IAM-service-role> 搭配遵循最低權限原則的 Amazon EMR 服務角色。
- <EMR-EC2-instance-profile> 您的執行個體描述檔。

5. 初始化 Amazon EMR 叢集後，您可以執行下列命令來初始化 Trino 工作階段：

```
trino-cli
```

6. 在 Trino CLI 中，您可以執行下列動作來檢視目錄：

```
SHOW CATALOGS;
```

建立 Iceberg 資料表

若要建立 Iceberg 資料表，您可以使用 CREATE TABLE陳述式。 以下是建立使用 Iceberg 隱藏分割的分割資料表的範例：

```
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**
>
> 如果您未指定格式，則會使用您在上一節中設定iceberg.file-format的值。

若要插入資料，請使用 INSERT INTO命令。範例如下：

```
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');
```

從 Iceberg 資料表讀取

您可以使用 SELECT陳述式讀取 Iceberg 資料表的最新狀態，如下所示：

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

將資料升級至 Iceberg 資料表

您可以使用 MERGE INTO陳述式執行 upsert 操作（同時插入新記錄並更新現有記錄）。範例如下：

```
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);
```

從 Iceberg 資料表刪除記錄

若要從 Iceberg 資料表刪除資料，請使用 DELETE FROM運算式，並指定符合要刪除資料列的篩選條件。範例如下：

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

查詢 Iceberg 資料表中繼資料

Iceberg 可透過 SQL 存取其中繼資料。您可以透過查詢命名空間 來存取任何指定資料表 (<table_name>) 的中繼資料"<table_name>.$<metadata_table>"。如需中繼資料資料表的完整清單，請參閱 Iceberg 文件中的檢查資料表。

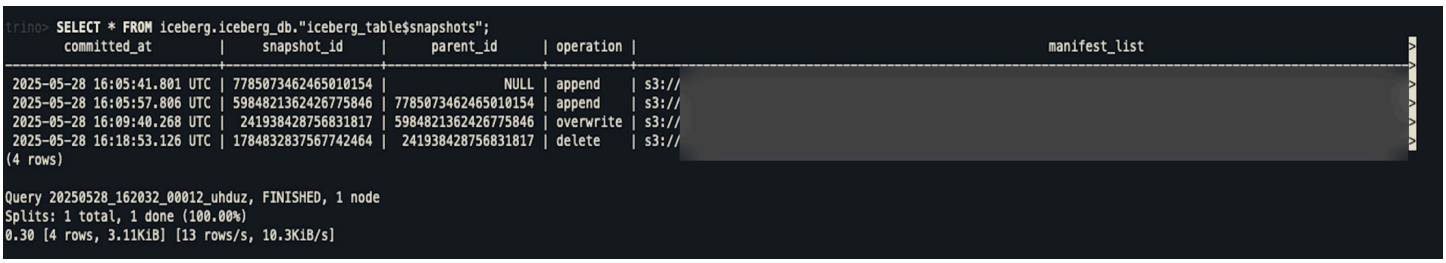
以下是檢查 Iceberg 中繼資料的查詢清單範例：

```
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";
```

例如，這個查詢：

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

提供輸出：



使用時間歷程

Iceberg 資料表中的每個寫入操作 （ 插入、更新、 upsert 或刪除 ） 都會建立新的快照。然後，您可以將這些快照用於時間歷程，以返回時間並檢查過去資料表的狀態。

下列時間歷程查詢會根據特定 顯示資料表的狀態snapshot_id：

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

下列時間歷程查詢會根據特定時間戳記顯示資料表的狀態：

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

搭配 Trino 使用 Iceberg 時的考量事項

Iceberg 資料表上的 Trino 寫入操作遵循merge-on-read 設計，因此它們會建立位置刪除檔案，而不是重寫受更新或刪除影響的整個資料檔案。如果您想要使用copy-on-write方法，請考慮使用 Spark 進行寫入操作。

使用 Amazon Data Firehose 處理 Iceberg 資料表

Amazon Data Firehose 是一種無伺服器、無程式碼服務，可將超過 20 個來源的資料串流，例如 AWS WAF 日誌、Amazon CloudWatch Logs AWS IoT、Amazon Kinesis Data Streams 和 Amazon Managed Streaming for Apache Kafka (Amazon MSK)，交付至 Amazon S3、Amazon Redshift、Snowflake 和 Splunk 等目的地。

您可以使用 Firehose 將串流資料直接交付至 Amazon S3 中的 Apache Iceberg 資料表。使用 Firehose，您可以將記錄從單一串流路由到不同的 Apache Iceberg 資料表，並自動將插入、更新和刪除操作套用至資料表中的記錄。Firehose 保證準確交付至 Iceberg 資料表一次。此功能需要使用 AWS Glue Data Catalog。

Firehose 也可以直接將串流資料交付至 Amazon S3 資料表。這些資料表提供針對大規模分析工作負載最佳化的儲存體，並包含可持續改善查詢效能並降低表格式資料儲存成本的功能。

如需有關如何設定 Firehose 串流以將資料交付至 Apache Iceberg 資料表的資訊，請參閱 Firehose 文件中的設定 Firehose 串流，或部落格文章使用 Amazon Data Firehose 將即時資料串流至 Amazon S3 中的 Apache Iceberg 資料表。

使用 Athena SQL 處理 Iceberg 資料表

Amazon Athena 提供 Apache Iceberg 的內建支援，不需要額外的步驟或組態。本節提供使用 Athena 與 Iceberg 資料表互動的支援功能和高階指引的詳細概觀。

版本和功能相容性

Iceberg 資料表規格支援

Apache Iceberg 資料表規格指定 Iceberg 資料表的行為。Athena 支援資料表格式第 2 版，因此您使用主控台、CLI 或 SDK 建立的任何 Iceberg 資料表本質上都會使用該版本。

如果您使用與另一個引擎建立的 Iceberg 資料表，例如 Amazon EMR 上的 Apache Spark，或者 AWS Glue，請務必使用資料表屬性來設定資料表格式版本。做為參考，請參閱本指南稍早的建立和撰寫 Iceberg 資料表一節。

Iceberg 功能支援

您可以使用 Athena 從 Iceberg 資料表讀取和寫入。當您使用 `UPDATE`、 `MERGE INTO`和 `DELETE FROM`陳述式變更資料時，Athena 僅支援merge-on-read模式。此屬性無法變更。若要使用copy-on-write更新或刪除資料，您必須使用其他引擎，例如 Amazon EMR 上的 Apache Spark 或 AWS Glue。下表摘要說明 Athena 中的 Iceberg 功能支援。

		DDL 支援		DML 支援		AWS Lake Formation 安全 （選用）
	資料表格式	建立資料表	結構描述演進	讀取資料	寫入資料	資料列/資料欄存取控制
Amazon Athena	2 版	✓	✓	✓	X Copy-on-write	✓
					✓ Merge-on-read	✓

Note

- Athena 不支援增量查詢。
- 在 Athena 中，更新、刪除和合併操作一律預設為在讀取時合併 (MoR)，無論資料表屬性中的任何寫入時複製 (CoW) 設定為何，因為不支援 CoW。

使用 Iceberg 資料表

如需在 Athena 中使用 Iceberg 的快速入門，請參閱本指南稍早的 Athena SQL 中的 Iceberg 資料表入門一節。

下表列出限制和建議。

情況	限制	建議
資料表 DDL 產生	使用其他引擎建立的 Iceberg 資料表可以具有未在 Athena 中公開的屬性。對於這些資料表，無法產生 DDL。	在建立資料表的引擎中使用對等陳述式（例如 Spark 的 `SHOW CREATE TABLE`陳述式）。
寫入 Iceberg 資料表之物件中的隨機 Amazon S3 字首	根據預設，使用 Athena 建立的 Iceberg 資料表已啟用 `write.object-storage.enabled` 屬性。	若要停用此行為並完全控制 Iceberg 資料表屬性，請使用其他引擎建立 Iceberg 資料表，例如 Amazon EMR 上的 Spark 或 AWS Glue。
增量查詢	Athena 目前不支援。	若要使用增量查詢來啟用增量資料擷取管道，請在 Amazon EMR 或 上使用 Spark AWS Glue。

使用 PyIceberg 處理 Iceberg 資料表

本節說明如何使用 PyIceberg 與 Iceberg 資料表互動。提供的範例是樣板程式碼，您可以在 Amazon Linux 2023 EC2 執行個體、AWS Lambda函數或任何具有正確設定AWS 憑證的 Python 環境中執行。

先決條件

> **Note**
>
> 這些範例使用 PyIceberg 1.9.1。

若要使用 PyIceberg，您需要 適用於 Python (Boto3) 的 AWS SDK 安裝 PyIceberg 和 。以下是如何設定 Python 虛擬環境以使用 PyIceberg 和 的範例 AWS Glue Data Catalog：

1. 使用 pip python 套件安裝程式下載 PyIceberg。您也需要 Boto3 才能與 互動 AWS 服務。您可以使用下列命令，設定要測試的本機 Python 虛擬環境：

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

2. 執行python 以開啟 Python shell 並測試命令。

連線至 Data Catalog

若要開始在 中使用 Iceberg 資料表 AWS Glue，您必須先連線到 AWS Glue Data Catalog。

load_catalog函數會建立目錄物件，做為所有 Iceberg 操作的主要界面，以初始化與 Data Catalog 的連線：

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

glue_catalog = load_catalog(
    'default',
```

```
    **{
        'client.region': region
    },
    type='glue'
)
```

列出和建立資料庫

若要列出現有的資料庫，請使用 `list_namespaces`函數：

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

若要建立新的資料庫，請使用 `create_namespace`函數：

```
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})
```

建立和寫入 Iceberg 資料表

未分割的資料表

以下是使用 `create_table`函數建立未分割 Iceberg 資料表的範例：

```
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)
```

您可以使用 list_tables函數來檢查資料庫內的資料表清單：

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

您可以使用 append函數和 PyArrow 在 Iceberg 資料表中插入資料：

```
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)
```

分割的資料表

以下是使用 create_table函數和 建立具有隱藏分割的分割 Iceberg 資料表的範例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
)
```

您可以將資料插入分割資料表的方式與未分割資料表相同。分割會自動處理。

```
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)
```

讀取資料

您可以使用 PyIceberg `scan`函數從 Iceberg 資料表讀取資料。您可以篩選資料列、選取特定資料欄，以及限制傳回的記錄數目。

```
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)
```

刪除資料

Pyiceberg `delete`函數可讓您使用 從資料表中移除記錄`delete_filter`：

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

存取中繼資料

Pyiceberg 提供多種函數來存取資料表中繼資料。以下是您可以檢視資料表快照相關資訊的方式：

```
#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}")
```

如需可用中繼資料的詳細清單，請參閱 Pyiceberg 文件的中繼資料程式碼參考區段。

使用時間歷程

您可以使用資料表快照進行時間歷程，以存取資料表的先前狀態。以下是在上次操作之前檢視資料表狀態的方法：

```
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)
```

如需可用函數的完整清單，請參閱 PyIcebergPython API 文件。

使用 Iceberg 資料表格式規格第 3 版

Apache Iceberg 資料表格式規格的最新版本是第 3 版。此版本推出進階功能，可建置 PB 級資料湖，以改善效能並降低營運開銷。它解決了第 2 版遇到的常見效能瓶頸，特別是在批次更新和合規刪除操作方面。

AWS 支援 Iceberg 第 3 版規格中定義的刪除向量和資料列歷程。這些功能可在下列 Apache Spark 上使用 AWS 服務。

AWS 服務	第 3 版支援
Amazon EMR for Apache Spark	Amazon EMR 7.12 版及更新版本
AWS Glue	是
AWS Glue：Iceberg REST API、資料表維護	是
Amazon SageMaker Unified Studio 筆記本	是
Amazon S3 資料表：Iceberg REST API、資料表維護	是
Amazon Athena (Trino)	否

第 3 版的主要功能

刪除向量會以儲存為 Puffin 檔案的有效二進位格式取代第 2 版中使用的位置刪除檔案。這可消除隨機批次更新和一般資料保護法規 (GDPR) 合規刪除的寫入放大，並大幅降低維護新資料的額外負荷。處理高頻率更新的組織將立即改善寫入效能，並從較少的小型檔案降低儲存成本。

資料列譜系可在資料列層級啟用精確的變更追蹤。您的下游系統可以逐步處理變更，加速資料管道並降低變更資料擷取 (CDC) 工作流程的運算成本。此內建功能不需要自訂變更追蹤實作。

版本相容性

第 3 版維持與第 2 版資料表的回溯相容性。AWS 服務同時支援第 2 版和第 3 版資料表，因此您可以：

- 跨第 2 版和第 3 版資料表執行查詢。

- 將現有的第 2 版資料表升級至第 3 版，無需重新寫入資料。
- 跨越第 2 版和第 3 版快照的執行時間歷程查詢。
- 跨資料表版本使用結構描述演變和隱藏分割。

第 3 版入門

先決條件

使用第 3 版資料表之前，請確定您已：

- AWS 帳戶 具有適當 AWS Identity and Access Management (IAM) 許可的 。
- 存取一或多個 AWS 分析服務 (Amazon EMR AWS Glue、Amazon SageMaker Unified Studio 筆記本或 Amazon S3 Tables)。
- 用於儲存資料表資料和中繼資料的 S3 儲存貯體。
- 如果您要建置自己的 Iceberg 基礎設施，可使用資料表儲存貯體開始使用 Amazon S3 Tables 或一般用途 S3 儲存貯體。
- 已設定的 AWS Glue 目錄。

建立第 3 版資料表

建立新的資料表

若要建立新的 Iceberg 第 3 版資料表，請將`format-version`資料表屬性設定為 3。

使用 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'
```

```
)
```

將第 2 版資料表升級至第 3 版

您可以將現有的第 2 版資料表以原子方式升級至第 3 版， 而無需重寫資料。

使用 Spark SQL：

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

> **Important**
>
> 第 3 版是單向升級。資料表從第 2 版升級至第 3 版之後，就無法透過標準操作降級回第 2 版。

升級期間會發生什麼情況：

- 會以原子方式建立新的中繼資料快照。
- 重複使用現有的 Parquet 資料檔案。
- 資料列譜系欄位會新增至資料表中繼資料。

升級後：

- 下一個壓縮動作會移除舊版本 2 的刪除檔案。
- 新的修改將使用第 3 版刪除向量檔案。

升級不會執行資料列歷程變更追蹤記錄的歷史回填。

啟用刪除向量

若要利用刪除向量進行更新、刪除和合併，請設定您的寫入模式。

使用 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'
)
```

這些設定可確保更新、刪除和合併操作會建立刪除向量檔案，而不是重寫整個資料檔案。

使用資料列歷程記錄進行變更追蹤

第 3 版會自動新增資料列歷程中繼資料欄位以追蹤變更。

使用 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
```

`_row_id` 欄位可唯一識別每一列，並 `_last_updated_sequence_number`追蹤列上次修改的時間。使用這些欄位來：

- 識別變更的資料列以進行增量處理。
- 追蹤資料歷程以確保合規。
- 最佳化 CDC 管道。
- 僅處理變更以降低運算成本。

第 3 版的最佳實務

何時使用第 3 版

在下列情況下，請考慮升級到第 3 版或從第 3 版開始：

- 您經常執行批次更新或刪除。
- 您需要符合 GDPR 或合規刪除要求。

- 您的工作負載涉及高頻率 upsert。
- 您需要有效率的 CDC 工作流程。
- 您想要降低小型檔案的儲存成本。
- 您需要更好的變更追蹤功能。

最佳化寫入效能

- 為繁重的更新工作負載啟用刪除向量：

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

- 設定適當的檔案大小：

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

最佳化讀取效能

- 使用資料列譜系進行增量處理。
- 使用時間歷程來存取歷史資料，無需複製。
- 啟用統計資料收集，以便更好地規劃查詢。

遷移策略

當您從第 2 版遷移到第 3 版時，請遵循下列最佳實務：

- 首先在非生產環境中進行測試，以驗證升級程序和效能。
- 在低活動期間升級，將對並行操作的影響降至最低。
- 監控初始效能，並在升級後追蹤指標。
- 執行壓縮以在升級後合併刪除檔案。

- 更新您的團隊文件以反映第 3 版功能。

相容性考量

- 引擎版本 – 確定存取資料表的所有引擎都支援第 3 版。
- 第三方工具 – 在升級之前，請確認工具的第 3 版相容性。
- 備份策略 – 測試快照型復原程序。
- 監控 – 更新第 3 版特定指標的監控儀表板。

疑難排解

常見問題

錯誤：「不支援 格式版本 3」

- 確認您的引擎版本支援第 3 版。 如需詳細資訊，請參閱本節開頭的表格。
- 檢查目錄相容性。
- 請確定您使用的是最新版本的 AWS 服務。

升級後效能降低

- 確認沒有壓縮壓縮失敗。如需詳細資訊，請參閱 Amazon S3 文件中的記錄和監控 S3 資料表。Amazon S3
- 確認已啟用刪除向量。應設定下列屬性：

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

  您可以使用下列程式碼來驗證資料表屬性：

```
DESCRIBE FORMATTED myns.orders_v3
```

- 檢閱您的分割區策略。過度分割可能會導致小型檔案。執行下列查詢以取得資料表的平均檔案大小：

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

與第三方工具不相容

- 確認工具支援第 3 版規格。
- 考慮維護不支援工具的第 2 版資料表。
- 請聯絡工具廠商以取得其第 3 版支援時間表。

取得說明

- 如需 AWS 服務了解特定問題，請聯絡 AWS 支援。
- 若要從 Iceberg 社群取得協助，請使用 Iceberg Slack 頻道。
- 如需有關使用 AWS 服務 管理分析工作負載的資訊，請參閱 上的分析 AWS。

定價

- Amazon EMR 運算和儲存定價
- Amazon SageMaker 定價
- AWS Glue 任務執行和 Data Catalog 定價
- S3 Tables 儲存和請求定價

可用性

Iceberg 資料表格式規格第 3 版支援適用於 Amazon EMR AWS Glue AWS Glue Data Catalog、 和 S3 Tables 運作的所有 AWS 區域 。如需區域可用性，請參閱AWS 服務 依區域。

其他資源

- Apache Iceberg 文件
- Apache Iceberg 資料表規格
- 將表格式資料從 Amazon S3 遷移至 S3 資料表的指引

- 教學課程：S3 資料表入門

將現有資料表遷移至 Iceberg

本節著重於將您現有的 Hive 樣式資料表遷移至 Iceberg 格式。它適用於使用傳統 Hive 相容格式的資料表，例如 Apache Parquet 或 Apache ORC。此資訊不適用於已使用現代資料表格式的資料表，例如 Linux Foundation Delta Lake 或 Apache Hudi。

若要將目前的 Hive 樣式資料表遷移至 Iceberg 格式，您可以使用就地或完整資料遷移：

- 就地遷移是在現有資料檔案上產生 Iceberg 中繼資料檔案的程序。
- 完整資料遷移會建立 Iceberg 中繼資料層，也會將現有資料檔案從原始資料表重寫至新的 Iceberg 資料表。

以下各節提供每個遷移方法的詳細概觀，包括step-by-step說明和實作的考量。如需這些遷移策略的詳細資訊，請參閱 Iceberg 文件的資料表遷移一節。

在您檢閱就地和完整資料遷移方法的詳細資訊後，請參閱下列兩個重要章節，以協助您的決策程序：

- 選擇遷移策略可透過一系列問題和案例提供指引，協助您根據您的特定需求和使用案例來判斷最適合的遷移方法。
- 遷移選項摘要提供全面的資料表，可比較不同遷移選項的關鍵特性和考量事項。此資料表可做為快速參考指南，並提供功能比較，協助您了解方法之間的技術取捨。

就地遷移

就地遷移不需要重寫所有資料檔案。反之，會產生 Iceberg 中繼資料檔案，並連結到您現有的資料檔案。此方法通常更快且更具成本效益，尤其是具有相容檔案格式的大型資料集或資料表，例如 Parquet、Avro 和 ORC。

> **Note**
>
> 遷移至 Amazon S3 Tables 時，無法使用就地遷移。

Iceberg 提供實作就地遷移的兩個主要選項：

- 使用快照程序建立新的 Iceberg 資料表，同時保持來源資料表不變。如需詳細資訊，請參閱 Iceberg 文件中的快照表。

- 使用遷移程序建立新的 Iceberg 資料表，以取代來源資料表。如需詳細資訊，請參閱 Iceberg 文件中的遷移資料表。雖然此程序適用於 Hive Metastore (HMS)，但目前與 不相容 AWS Glue Data Catalog。本指南稍後章節中的複寫資料表遷移程序 AWS Glue Data Catalog提供了使用 Data Catalog 實現類似結果的解決方法。

使用 snapshot或 執行就地遷移後migrate，某些資料檔案可能會保持未遷移狀態。這通常發生在寫入器在遷移期間或之後繼續寫入來源資料表時。若要將這些剩餘檔案納入 Iceberg 資料表，您可以使用 add_files 程序。如需詳細資訊，請參閱 Iceberg 文件中的新增檔案。

假設您有一個在 Athena 中建立並填入的 Parquet 型products資料表，如下所示：

```
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');
```

下列各節說明如何搭配此資料表使用 snapshot和 migrate 程序。

選項 1：快照程序

snapshot 程序會建立新的 Iceberg 資料表，其名稱不同，但會複寫來源資料表的結構描述和分割。此操作會在動作期間和之後讓來源資料表完全保持不變。它有效地建立資料表的輕量型副本，這對於測試案例或資料探勘特別有用，而不會危及對原始資料來源的修改。此方法可啟用轉換期間，其中原始資料表和 Iceberg 資料表都會保持可用 （請參閱本節結尾的備註）。測試完成後，您可以將所有寫入器和讀取器轉換為新資料表，藉此將新的 Iceberg 資料表移至生產環境。

您可以在任何 Amazon EMR 部署模型 （例如，Amazon EMR on EC2、Amazon EMR on EKS、EMR Serverless) 和 中使用 Spark 來執行snapshot程序 AWS Glue。

若要使用 snapshot Spark 程序測試就地遷移，請遵循下列步驟：

1. 啟動 Spark 應用程式並使用下列設定設定 Spark 工作階段：
   - "spark.sql.extensions":"org.apache.iceberg.spark.extensions.IcebergSparkSessi
   - "spark.sql.catalog.spark_catalog":"org.apache.iceberg.spark.SparkSessionCatal
   - "spark.sql.catalog.spark_catalog.type":"glue"
   - "spark.hadoop.hive.metastore.client.factory.class":"com.amazonaws.glue.catalo
2. 執行 snapshot 程序以建立新的 Iceberg 資料表，指向原始資料表資料檔案：

```
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)
```

   輸出資料框架包含 imported_files_count（已新增的檔案數目）。
3. 透過查詢來驗證新資料表：

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

備註

- 執行程序後，來源資料表上的任何資料檔案修改都會將產生的資料表擲出不同步。您新增的新檔案不會出現在 Iceberg 資料表中，而您移除的檔案會影響 Iceberg 資料表中的查詢功能。若要避免同步問題：
  - 如果新的 Iceberg 資料表適用於生產用途，請停止寫入原始資料表的所有程序，並將其重新導向至新資料表。
  - 如果您需要轉換期間或新的 Iceberg 資料表用於測試目的，請參閱本節稍後的就地遷移後保持 Iceberg 資料表同步，以取得維護資料表同步的指引。
- 當您使用 snapshot 程序時， gc.enabled 屬性會在建立的 Iceberg 資料表的資料表屬性false中設定為 。此設定禁止 expire_snapshots、 remove_orphan_files或 等動

作DROP TABLE搭配 PURGE選項，這會實際刪除資料檔案。仍然允許不直接影響來源檔案的 Iceberg 刪除或合併操作。

- 若要讓您的新 Iceberg 資料表完全正常運作，對實際刪除資料檔案的動作沒有限制，您可以將gc.enabled資料表屬性變更為 true。不過，此設定將允許影響來源資料檔案的動作，這可能會損毀對原始資料表的存取。因此，只有在您不再需要維護原始資料表的功能時，才變更 gc.enabled 屬性。例如：

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

選項 2：遷移程序

程序migrate會建立新的 Iceberg 資料表，其名稱、結構描述和分割與來源資料表相同。當此程序執行時，它會鎖定來源資料表並將其重新命名為 <table_name>_BACKUP_（或 backup_table_name 程序參數指定的自訂名稱）。

> **Note**
>
> 如果您將drop_backup程序參數設定為 true，則原始資料表將不會保留為備份。

因此，migrate資料表程序要求在執行動作之前停止影響來源資料表的所有修改。執行migrate程序之前：

- 停止與來源資料表互動的所有寫入器。
- 修改原生不支援 Iceberg 的讀取器和寫入器，以啟用 Iceberg 支援。

例如：

- Athena 可在不修改的情況下繼續運作。
- Spark 需要：
  - 要包含在 classpath 中的 Iceberg Java Archive (JAR) 檔案（請參閱本指南稍早章節中的在 Amazon EMR 中使用 Iceberg 和使用 Iceberg AWS Glue）。

- 下列 Spark 工作階段目錄組態（使用 SparkSessionCatalog 新增 Iceberg 支援，同時維護非 Iceberg 資料表的內建目錄功能）：
  - "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

執行程序後，您可以使用新的 Iceberg 組態重新啟動寫入器。

目前， migrate 程序與 不相容 AWS Glue Data Catalog，因為 Data Catalog 不支援 RENAME操作。因此，建議您只在使用 Hive Metastore 時使用此程序。如果您使用的是 Data Catalog，請參閱下一節以取得替代方法。

您可以在所有 Amazon EMR 部署模型 (Amazon EMR on EC2、Amazon EMR on EKS、EMR Serverless) 中執行migrate此程序 AWS Glue，但需要已設定的 Hive Metastore 連線。建議選擇 Amazon EMR on EC2，因為它提供內建的 Hive 中繼存放區組態，可將設定複雜性降至最低。

若要從使用 Hive Metastore 設定的 EC2 叢集上的 Amazon EMR 使用 migrate Spark 程序測試就地遷移，請遵循下列步驟：

1. 啟動 Spark 應用程式並設定 Spark 工作階段以使用 Iceberg Hive 目錄實作。例如，如果您使用的是 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. 在 Hive 中繼存放區中建立products資料表。這是來源資料表，已存在於典型的遷移中。
   a. 在 Hive 中繼存放區中建立products外部 Hive 資料表，以指向 Amazon S3 中的現有資料：

```
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. 使用 MSCK REPAIR TABLE命令新增現有的分割區：

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

c. 透過執行SELECT查詢來確認資料表包含資料：

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

輸出範例：

```
>>> 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. 使用 Iceberg `migrate` 程序：

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

df_res.show()
```

輸出資料框架包含 migrated_files_count （新增至 Iceberg 資料表的檔案數目）：

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

4. 確認已建立備份資料表：

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

輸出範例：

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

5. 透過查詢 Iceberg 資料表來驗證操作：

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

備註

- 執行程序後，如果未正確設定 Iceberg 支援，查詢或寫入來源資料表的所有目前程序都會受到影響。因此，我們建議您遵循下列步驟：
  1. 在遷移之前使用來源資料表停止所有程序。
  2. 執行遷移。
  3. 使用適當的 Iceberg 設定重新啟用程序。
- 如果在遷移程序期間發生資料檔案修改（新增檔案或移除檔案），產生的資料表將會不同步。如需同步選項，請參閱本節稍後的在就地遷移之後保持 Iceberg 資料表同步。

在 中複寫資料表遷移程序 AWS Glue Data Catalog

您可以在 AWS Glue Data Catalog （備份原始資料表並以 Iceberg 資料表取代） 中複寫遷移程序的結果，方法如下：

1. 使用 快照程序來建立新的 Iceberg 資料表，以指向原始資料表的資料檔案。
2. 在 Data Catalog 中備份原始資料表中繼資料：
   a. 使用 GetTable API 擷取來源資料表定義。
   b. 使用 GetPartitions API 擷取來源資料表分割區定義。
   c. 使用 CreateTable API 在 Data Catalog 中建立備份資料表。
   d. 使用 CreatePartition 或 BatchCreatePartition API，將分割區註冊至 Data Catalog 中的備份資料表。
3. 將 `gc.enabled` Iceberg 資料表屬性變更為 `false`，以啟用完整資料表操作。
4. 捨棄原始資料表。
5. 在資料表根位置的中繼資料資料夾中找到 Iceberg 資料表中繼資料 JSON 檔案。
6. 使用 register_table 程序，搭配原始資料表名稱和`snapshot`程序所建立`metadata.json`檔案的位置，在 Data Catalog 中註冊新資料表：

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

就地遷移後保持 Iceberg 資料表同步

`add_files` 此程序提供靈活的方式來將現有資料整合到 Iceberg 資料表中。具體而言，它會在 Iceberg 的中繼資料層中參考其絕對路徑，以註冊現有的資料檔案 （例如 Parquet 檔案）。根據預設，程序會從所有資料表分割區將檔案新增至 Iceberg 資料表，但您可以選擇性地從特定分割區新增檔案。這種選擇性方法在幾個案例中特別有用：

- 在初始遷移之後，將新的分割區新增至來源資料表時。
- 資料檔案在初始遷移後新增至現有分割區或從現有分割區中移除時。不過，重新新增修改過的分割區需要先刪除分割區。本節稍後將提供有關此項目的詳細資訊。

以下是在執行就地遷移 (snapshot 或 migrate) 之後使用 add_file 程序的一些考量，以保持新的 Iceberg 資料表與來源資料檔案同步：

- 將新資料新增至來源資料表中的新分割區時，請使用 add_files 程序搭配 partition_filter選項，以選擇性地將這些新增項目納入 Iceberg 資料表：

```
spark.sql(f"""
CALL system.add_files(
source_table => 'mydb.products',
table => 'mydb.products_iceberg',
partition_filter => map('category', 'electronics')
).show(truncate=False)
```

  或：

```
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)
```

- 當您指定 partition_filter選項時，add_files程序會掃描整個來源資料表或特定分割區中的檔案，並嘗試將找到的所有檔案新增至 Iceberg 資料表。根據預設，check_duplicate_files程序屬性會設定為 true，如果 Iceberg 資料表中已有檔案，則會防止程序執行。這很重要，因為沒有內建選項可略過先前新增的檔案，而停用check_duplicate_files會導致新增檔案兩次，進而建立重複項目。將新檔案新增至來源資料表時，請遵循下列步驟：

  1. 對於新分割區，使用 add_files搭配 partition_filter ，僅從新分割區匯入檔案。
  2. 對於現有的分割區，請先從 Iceberg 資料表刪除分割區，然後add_files為該分割區重新執行，並指定 partition_filter。例如：

```
# 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**
>
> 每個add_files操作都會產生具有附加資料的新 Iceberg 資料表快照。

選擇正確的就地遷移策略

若要選擇最佳的就地遷移策略，請考慮下表中的問題。

問題	建議	說明
您是否要在不重寫資料的情況下快速遷移，同時讓 Hive 和 Iceberg 資料表可供測試或逐步轉換使用？	snapshot 程序後接add_files 程序	使用 snapshot 程序透過複製結構描述和參考資料檔案來建立新的 Iceberg 資料表，而無需修改來源資料表。使用 add_files 程序來合併遷移後新增或修改的分割區，請注

問題	建議	說明
		意，重新新增修改的分割區需要先刪除分割區。
您是否使用 Hive 中繼存放區，並且想要立即將 Hive 資料表取代為 Iceberg 資料表，而無需重寫資料？	`migrate` 程序後接`add_files` 程序	使用 `migrate` 程序來建立 Iceberg 資料表、備份來源資料表，並將原始資料表取代為 Iceberg 版本。 注意：此選項與 Hive Metastore 相容，但與 不相容 AWS Glue Data Catalog。 使用 `add_files` 程序來合併遷移後新增或修改的分割區，請注意，重新新增修改的分割區需要先刪除分割區。

問題	建議	說明
您是否正在使用 AWS Glue Data Catalog ，並且想要立即將 Hive 資料表取代為 Iceberg 資料表，而無需重寫資料？	調整migrate程序，接著是add_files 程序	複寫migrate程序行為： 1. 使用 snapshot建立 Iceberg 資料表。 2. 使用 AWS Glue APIs 備份原始資料表中繼資料。 3. 在 Iceberg 資料表屬性gc.enabled 上啟用 。 4. 捨棄原始資料表。 5. 使用 register_table 建立具有原始名稱的新資料表項目。 注意：此選項需要手動處理中繼資料備份的 AWS Glue API 呼叫。 使用 add_files 程序來合併遷移後新增或修改的分割區，請注意，重新新增修改的分割區需要先刪除分割區。

完整資料遷移

完整資料遷移會重新建立資料檔案和中繼資料。相較於就地遷移，此方法需要更長的時間，且需要額外的運算資源。不過，完整資料遷移提供許多改善資料表品質的機會，並最佳化資料儲存和存取模式。

在完整資料遷移期間，您可以執行數個有益的操作，例如確保完整性和正確性的資料驗證、更符合目前需求的結構描述修改，以及改善查詢效能的分割區策略調整。您也可以重新排序資料，以最佳化常見存取模式、實作 Iceberg 隱藏分割以提高查詢效率，並視需要執行檔案格式轉換 （例如，從 CSV 轉換到 Parquet)。

這些功能使完整資料遷移非常適合轉換為 Iceberg 格式，以及全面精簡和最佳化資料儲存策略。雖然完整資料遷移需要更多的時間和資源，但資料品質、組織和查詢效能的改善可以提供長期利益。若要實作完整資料遷移，請使用下列其中一個選項：

- 在 Spark `CREATE TABLE ... AS SELECT` (Amazon EMR 或） 或 Athena 中使用 (CTAS AWS Glue) 陳述式。您可以使用 和 子句來設定新 Iceberg `PARTITIONED BY`資料表的分割區規格和 `TBLPROPERTIES`資料表屬性。您可以根據您的需求變更新資料表的結構描述和分割，而不是從來源資料表繼承它們。
- 從來源資料表讀取，並在 Amazon EMR 或 上使用 Spark 將資料寫入為新的 Iceberg 資料表 AWS Glue。如需詳細資訊，請參閱 Iceberg 文件中的建立資料表。

選擇移轉策略

轉換為 Iceberg 格式時，就地遷移和完全遷移之間的選擇至關重要。若要判斷最適合您特定需求的方法，請考慮下列問題和建議：

問題	建議
什麼是資料檔案格式（例如 CSV 或 Apache Parquet)？	• 如果您的資料表檔案格式是 Parquet、ORC 或 Avro，請考慮就地遷移。 • 對於其他格式，例如 CSV、JSON 等，請使用完整資料遷移。
您要更新或合併資料表結構描述嗎？	• 如果您想要使用 Iceberg 原生功能來發展資料表結構描述，請考慮就地遷移。例如，您可以在遷移後重新命名資料欄。（結構描述可以在 Iceberg 中繼資料層中變更。） • 如果您想要移除整個資料欄，因為不再需要它們，我們建議您使用完整資料遷移。
資料表是否會受益於變更分割區策略？	• 如果 Iceberg 的分割方法符合您的需求（例如，使用新的分割區配置存放新資料，同時現有分割區保持不變），請考慮就地遷移。 • 如果您想要在資料表中使用隱藏的分割區，請考慮完整資料遷移。如需隱藏分割區的詳細資訊，請參閱最佳實務一節。
資料表是否會受益於新增或變更排序順序策略？	• 新增或變更資料的排序順序需要重寫資料集。在這種情況下，請考慮使用完整資料遷移。

問題	建議
	• 對於重寫所有資料表分割區的成本過高的大型資料表，請考慮使用就地遷移，並對最常存取的分割區執行壓縮（啟用排序）。
資料表是否有許多小型檔案？	• 將小型檔案合併為較大的檔案需要重寫資料集。在這種情況下，請考慮使用完整資料遷移。 • 對於重寫所有資料表分割區的成本過高的大型資料表，請考慮使用就地遷移，並對最常存取的分割區執行壓縮（啟用排序）。

遷移選項摘要

此資料表摘要說明每個遷移選項的主要特性和考量事項。

功能	就地遷移 快照	就地遷移 migrate	完整資料遷移 CTAS 或 (CREATE TABLE + INSERT)
資料配置改善是遷移程序的一部分			
• 重新排序資料	⊗否	⊗否	⊘ 是
• 變更分割（例如，使用 Iceber	⊗否	⊗否	⊘ 是

功能	就地遷移 快照	就地遷移 migrate	完整資料遷移 CTAS 或 (CREATE TABLE + INSERT)
隱藏分割）			
• 變更資料表結構描述	⊗否	⊗否	⊘ 是
• 最佳化檔案大小	⊗否	⊗否	⊘ 是
• 在新增資料之前驗證現有資料的結構描述	⊗否	⊗否	⊘ 是
支援的檔案格式	Parquet、Avro、ORC	Parquet、Avro、ORC	Parquet、Avro、ORC、JSON、CSV

功能	就地遷移 快照	就地遷移 migrate	完整資料遷移 CTAS 或 (CREATE TABLE + INSERT)
Iceberg 資料表取代來源資料表	⊗否 （建立新的資料表，但您可以使用其他步驟取代來源資料表）	⊘ 是 （建立備份資料表並以 Iceberg 資料表取代來源資料表）	⊗否 （建立新的資料表）
來源資料表影響			
• Iceber 資料表上的檔案刪除操作 (expi apsho 操作、捨棄具有清除的資料表）	損毀來源資料表	Corrupts 備份資料表	安全、來源不受影響
Iceberg 資料表影響			

功能	就地遷移 快照	就地遷移 migrate	完整資料遷移 CTAS 或 (CREATE TABLE + INSERT)
• 來源資料表檔案移除時的影響	Corrupts Iceberg 資料表	Corrupts Iceberg 資料表	不會影響 Iceberg 資料表
• 如果在來源資料表位置上新增了新檔案，會影響。	在新資料表上看不到 （需要將分割區與 合併add_files ）	在新資料表上看不到 （需要將分割區與 合併add_files ）	在新資料表上看不到 （需要INSERT INTO新資料表）
成本	低	低	較高（完整資料重寫）
遷移速度	快速	快速	較慢
可用於遷移至 Amazon S3 Tables	⊗否	⊗否	⊘ 是

功能	就地遷移 快照	就地遷移 migrate	完整資料遷移 CTAS 或 (CREATE TABLE + INSERT)
需要手動 DDL	⊗否 （從來源資料表複製結構描述和分割區）	⊗否 （從來源資料表複製結構描述和分割區）	如果使用 CTAS， 只需要指定分割
最佳使用	無需重寫資料的快速遷移，允許 side-by-side使用 Hive 和 Iceberg 進行測試或逐步轉換。	可接受立即切換時，在不重寫資料的情況下更換 Hive 資料表。	完整 Iceberg 最佳化與資料重寫。重新設計分割區或結構描述，或改善配置和效能時的理想選擇。如果可能，一律建議。

最佳化 Apache Iceberg 工作負載的最佳實務

Iceberg 是一種資料表格式，旨在簡化資料湖管理並增強工作負載效能。不同的使用案例可能會優先考慮不同的層面，例如成本、讀取效能、寫入效能或資料保留，因此 Iceberg 提供組態選項來管理這些權衡。本節提供最佳化和微調 Iceberg 工作負載以符合需求的洞見。

主題

一般最佳實務

無論您的使用案例為何，當您在 上使用 Apache Iceberg 時 AWS，我們建議您遵循這些一般最佳實務。

- 使用 Iceberg 格式第 2 版。

  Athena 預設使用 Iceberg 格式第 2 版。

  當您在 Amazon EMR 上使用 Spark 或 AWS Glue 建立 Iceberg 資料表時，請指定 Iceberg 文件中所述的格式版本。
- 使用 AWS Glue Data Catalog 做為您的資料目錄。

  Athena AWS Glue Data Catalog 預設會使用 。

  當您在 Amazon EMR 上使用 Spark 或使用 AWS Glue Iceberg 時，請將下列組態新增至 Spark 工作階段以使用 AWS Glue Data Catalog。如需詳細資訊，請參閱本指南稍早中 Iceberg 的 Spark 組態 AWS Glue一節。

  ```
  "spark.sql.catalog.<your_catalog_name>.type": "glue"
  ```
- 使用 AWS Glue Data Catalog 做為鎖定管理員。

根據預設，Athena 會使用 AWS Glue Data Catalog 做為 Iceberg 資料表的鎖定管理員。

當您在 Amazon EMR 上使用 Spark 或使用 AWS Glue Iceberg 時，請務必將 Spark 工作階段組態設定為使用 AWS Glue Data Catalog 做為鎖定管理員。如需詳細資訊，請參閱 Iceberg 文件中的樂觀鎖定。

- 使用 Zstandard (ZSTD) 壓縮。

  Iceberg 的預設壓縮轉碼器是 gzip，可以使用資料表屬性 修改`write.<file_type>.compression-codec`。Athena 已使用 ZSTD 做為 Iceberg 資料表的預設壓縮轉碼器。

  一般而言，我們建議您使用 ZSTD 壓縮轉碼器，因為它在 GZIP 和 Snappy 之間取得平衡，並提供良好的讀取/寫入效能，而不會影響壓縮率。此外，可以調整壓縮層級以符合您的需求。如需詳細資訊，請參閱 Athena 文件中的 Athena 中的 ZSTD 壓縮層級。

  Snappy 可能會提供最佳的整體讀取和寫入效能，但壓縮比率低於 GZIP 和 ZSTD。如果您優先考慮效能，即使這意味著在 Amazon S3 中存放較大的資料磁碟區，Snappy 仍可能是最佳選擇。

最佳化讀取效能

本節討論您可以調校的資料表屬性，以最佳化讀取效能，不受引擎影響。

分割

如同 Hive 資料表，Iceberg 使用分割區做為索引的主要層，以避免讀取不必要的中繼資料檔案和資料檔案。資料欄統計資料也會納入考量，做為索引的輔助層，以進一步改善查詢規劃，進而改善整體執行時間。

分割您的資料

若要減少查詢 Iceberg 資料表時掃描的資料量，請選擇符合您預期讀取模式的平衡分割區策略：

- 識別經常用於查詢的資料欄。這些是理想的分割候選項目。例如，如果您通常查詢特定日期的資料，分割區資料欄的自然範例會是日期資料欄。
- 選擇低基數分割區資料欄，以避免建立過多分割區。太多分割區可能會增加資料表中的檔案數量，這可能會對查詢效能產生負面影響。根據經驗法則，「太多分割區」可定義為大多數分割區中的資料大小小於 所設定值的 2-5 倍的情況`target-file-size-bytes`。

> **Note**
>
> 如果您通常會在高基數資料欄 （例如，可以有數千個值的資料id欄） 上使用篩選條件進行查詢，請使用 Iceberg 的隱藏分割功能與儲存貯體轉換，如下一節所述。

使用隱藏的分割

如果您的查詢通常會篩選資料表資料欄的衍生項目，請使用隱藏的分割區，而不是明確建立新的資料欄以做為分割區。如需此功能的詳細資訊，請參閱 Iceberg 文件。

例如，在具有時間戳記資料欄 （例如，2023-01-01 09:00:00) 的資料集中，使用分割區轉換從時間戳記擷取日期部分，並即時建立這些分割區，而不是建立具有剖析日期的新資料欄 （例如，2023-01-01)。

隱藏分割最常見的使用案例為：

- 當資料具有時間戳記資料欄時，依日期或時間進行分割。Iceberg 提供多種轉換來擷取時間戳記的日期或時間部分。
- 當分割資料欄具有高基數且會導致太多分割區時，對資料欄的雜湊函數進行分割。Iceberg 的儲存貯體轉換會使用分割區資料欄上的雜湊函數，將多個分割區值分組為較少的隱藏 （儲存貯體） 分割區。

如需所有可用分割區轉換的概觀，請參閱 Iceberg 文件中的分割區轉換。

用於隱藏分割的資料欄可透過使用 year()和 等一般 SQL 函數，成為查詢述詞的一部分month()。述詞也可以與 BETWEEN和 等運算子結合AND。

> **Note**
>
> Iceberg 無法對產生不同資料類型的函數執行分割區剔除；例如，substring(event_time, 1, 10) = '2022-01-01'。

使用分割區演變

當現有的分割區策略不最佳時，請使用 Iceberg 的分割區演變。例如，如果您選擇每小時分割區太小（每個分割區只有幾個 MB)，請考慮轉換為每日或每月分割區。

當資料表的最佳分割區策略最初不清楚，而且您想要在獲得更多洞見時精簡分割區策略時，您可以使用此方法。分割區演變的另一個有效用途是資料磁碟區變更，且目前的分割區策略會隨著時間而變得較不有效。

如需如何發展分割區的說明，請參閱 Iceberg 文件中的 ALTER TABLE SQL 延伸。

調校檔案大小

最佳化查詢效能需要將資料表中的小型檔案數量降至最低。為了獲得良好的查詢效能，我們通常建議將 Parquet 和 ORC 檔案保留大於 100 MB。

檔案大小也會影響 Iceberg 資料表的查詢規劃。隨著資料表中的檔案數量增加，中繼資料檔案的大小也會增加。較大的中繼資料檔案可能會導致較慢的查詢規劃。因此，當資料表大小增加時，請增加檔案大小 以減輕中繼資料的指數擴展。

使用下列最佳實務，在 Iceberg 資料表中建立適當大小的檔案。

設定目標檔案和資料列群組大小

Iceberg 提供下列用於調校資料檔案配置的關鍵組態參數。我們建議您使用這些參數來設定目標檔案大小和資料列群組或執行大小。

Parameter (參數)	預設值	註解
`write.target-file-size-bytes`	512 MB	此參數指定 Iceberg 將建立的檔案大小上限。不過，某些檔案的寫入大小可能會小於此限制。
`write.parquet.row-group-size-bytes`	128 MB	Parquet 和 ORC 都會將資料存放在區塊中，以便引擎可以避免讀取某些操作的整個檔案。
`write.orc.stripe-size-bytes`	64 MB	
`write.distribution-mode`	無，適用於 Iceberg 1.1 版及更低版本	Iceberg 要求 Spark 在寫入儲存體之前，先排序其任務之間的資料。

Parameter (參數)	預設值	註解
	雜湊，從 Iceberg 1.2 版開始	

- 根據您預期的資料表大小，請遵循下列一般準則：
  - 小型資料表（最多幾個 GB) – 將目標檔案大小縮減至 128 MB。同時減少資料列群組或條紋大小（例如 8 或 16 MB)。
  - 中大型資料表（從數 GB 到數百 GB) – 預設值是這些資料表的良好起點。如果您的查詢非常選擇性，請調整資料列群組或條紋大小（例如，調整為 16 MB)。
  - 非常大的資料表（數百 GB 或 TB) – 將目標檔案大小增加到 1024 MB 或更高，如果您的查詢通常提取大量資料集，請考慮增加資料列群組或條紋大小。

- 若要確保寫入 Iceberg 資料表的 Spark 應用程式建立適當大小的檔案，請將 `write.distribution-mode` 屬性設定為 `hash`或 `range`。如需這些模式之間差異的詳細說明，請參閱 Iceberg 文件中的撰寫分佈模式。

這些是一般準則。我們建議您執行測試，以識別特定資料表和工作負載最合適的值。

執行定期壓縮

上表中的組態會設定寫入任務可建立的檔案大小上限，但不保證檔案會有該大小。為了確保適當的檔案大小，請定期執行壓縮，將小型檔案合併成較大的檔案。如需執行壓縮的詳細指引，請參閱本指南稍後的 Iceberg 壓縮。

最佳化資料欄統計資料

Iceberg 使用資料欄統計資料來執行檔案刪除，透過減少查詢掃描的資料量來改善查詢效能。若要受益於資料欄統計資料，請確定 Iceberg 會收集查詢篩選條件中常用的所有資料欄統計資料。

根據預設，Iceberg 只會收集每個資料表中前 100 個資料欄的統計資料，如資料表屬性 所定義`write.metadata.metrics.max-inferred-column-defaults`。如果您的資料表有超過 100 個資料欄，而且您的查詢經常參考前 100 個資料欄以外的資料欄（例如，您可能有篩選資料欄 132 的 查詢），請確定 Iceberg 收集這些資料欄的統計資料。有兩種選項可達成此目標：

- 當您建立 Iceberg 資料表時，請重新排序資料欄，以便查詢所需的資料欄落在 設定的資料欄範圍內 `write.metadata.metrics.max-inferred-column-defaults`（預設為 100)。

注意：如果您不需要 100 個資料欄的統計資料，您可以將`write.metadata.metrics.max-inferred-column-defaults`組態調整為所需的值（例如 20) 並重新排序資料欄，以便您需要讀取和寫入查詢的資料欄落在資料集左側的前 20 個資料欄內。

- 如果您在查詢篩選條件中只使用幾個資料欄，您可以停用指標收集的整體屬性，並選擇性地選擇要收集統計資料的個別資料欄，如本範例所示：

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

注意：資料欄統計資料在這些資料欄上排序時最有效。如需詳細資訊，請參閱本指南稍後的設定排序順序一節。

選擇正確的更新策略

當您的使用案例接受較慢的寫入操作時，請使用copy-on-write策略來最佳化讀取效能。這是 Iceberg 使用的預設策略。

Copy-on-write可提高讀取效能，因為檔案是以讀取最佳化的方式直接寫入儲存體。不過，相較於merge-on-read，每個寫入操作需要更長的時間，並耗用更多運算資源。這提供了讀取和寫入延遲之間的傳統取捨。一般而言，copy-on-write非常適合大多數更新共置於相同資料表分割區（例如，用於每日批次載入）的使用案例。

Copy-on-write組態 (`write.update.mode`、 和 `write.merge.mode`) 可以在資料表層級設定`write.delete.mode`，或在應用程式端獨立設定。

使用 ZSTD 壓縮

您可以使用資料表屬性 修改 Iceberg 使用的壓縮轉碼器`write.<file_type>.compression-codec`。我們建議您使用 ZSTD 壓縮轉碼器來改善資料表的整體效能。

根據預設，Iceberg 1.3 版和更早版本使用 GZIP 壓縮，相較於 ZSTD，可提供較慢的讀取/寫入效能。

注意：某些引擎可能會使用不同的預設值。對於使用 Athena 或 Amazon EMR 7.x 版建立的 Iceberg 資料表，這是這種情況。

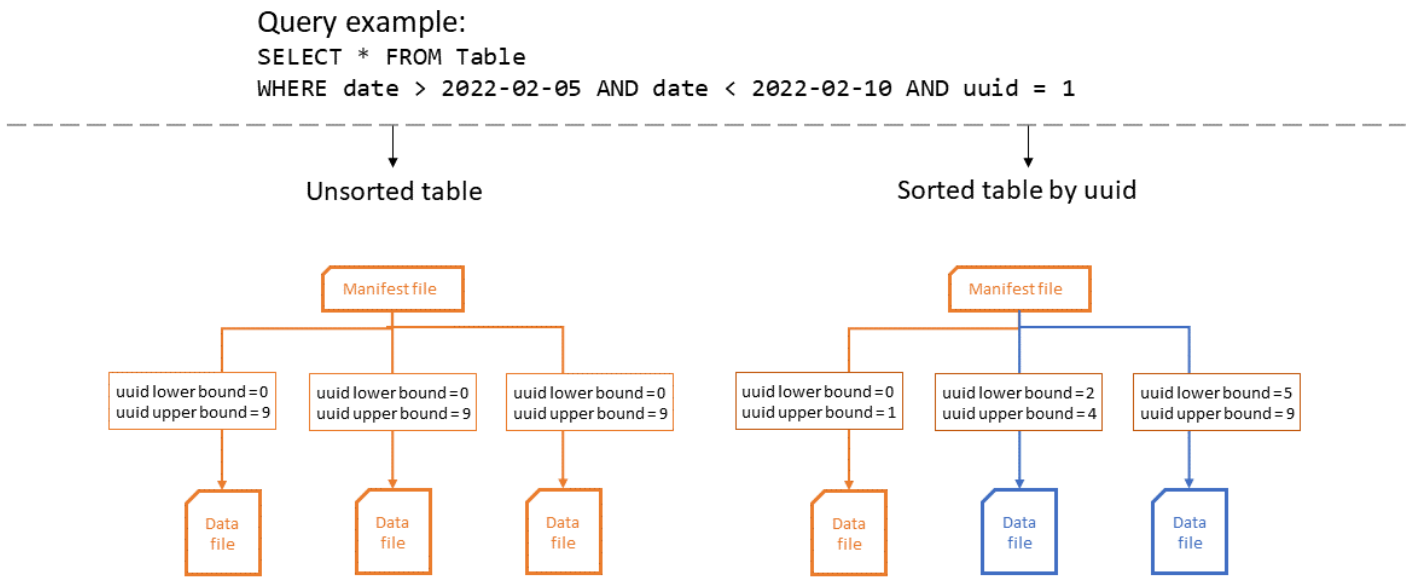
設定排序順序

為了改善 Iceberg 資料表的讀取效能，建議您根據查詢篩選條件中常用的一或多個資料欄來排序資料表。排序結合 Iceberg 的資料欄統計資料，可以使檔案剔除更有效率，進而加快讀取操作速度。排序也會減少使用查詢篩選條件中排序資料欄之查詢的 Amazon S3 請求數量。

您可以使用 Spark 執行資料定義語言 (DDL) 陳述式，在資料表層級設定階層排序順序。如需可用選項，請參閱 Iceberg 文件。設定排序順序後，寫入器會將此排序套用至 Iceberg 資料表中的後續資料寫入操作。

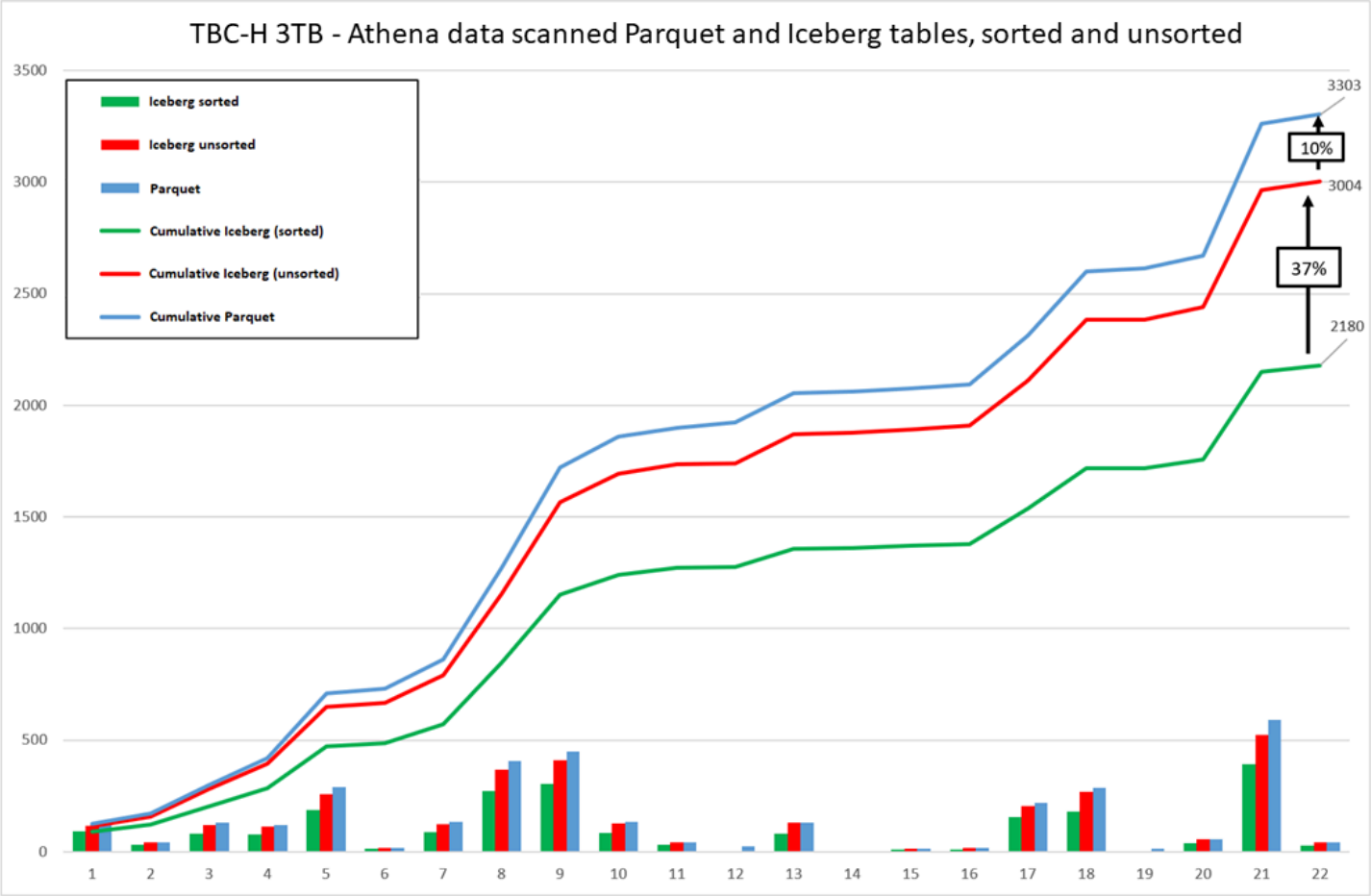
例如，在大部分查詢依 篩選的日期 (yyyy-mm-dd) 分割的資料表中uuid，您可以使用 DDL 選項來Write Distributed By Partition Locally Ordered確保 Spark 寫入具有非重疊範圍的檔案。

下圖說明排序資料表時，資料欄統計資料的效率如何改善。在此範例中，排序的資料表只需要開啟單一檔案，並從 Iceberg 的分割區和檔案獲得最大效益。在未排序的資料表中，任何 uuid 都可能存在於任何資料檔案中，因此查詢必須開啟所有資料檔案。

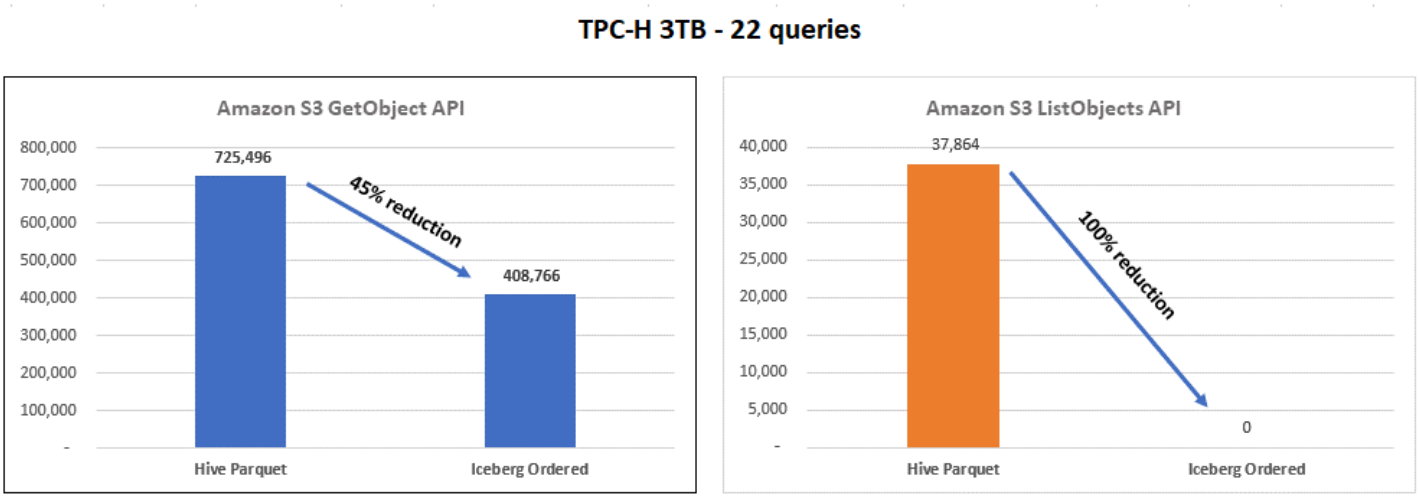


變更排序順序不會影響現有的資料檔案。您可以使用 Iceberg 壓縮來套用排序順序。

使用 Iceberg 排序資料表可能會降低工作負載的成本，如下圖所示。



這些圖表摘要了執行 Hive (Parquet) 資料表與 Iceberg 排序資料表的 TPC-H 基準測試的結果。不過，其他資料集或工作負載的結果可能不同。



最佳化寫入效能

本節討論您可以調校的資料表屬性，以最佳化 Iceberg 資料表上的寫入效能，與引擎無關。

設定資料表分佈模式

Iceberg 提供多種寫入分佈模式，可定義寫入資料在 Spark 任務中的分佈方式。如需可用模式的概觀，請參閱 Iceberg 文件中的撰寫分佈模式。

對於優先考慮寫入速度的使用案例，特別是在串流工作負載中，請將 `write.distribution-mode`設定為 none。這可確保 Iceberg 不會請求額外的 Spark 隨機播放，而且資料會在 Spark 任務中可用時寫入。此模式特別適合 Spark 結構化串流應用程式。

> **Note**
>
> 將寫入分佈模式設定為 none 往往會產生許多小型檔案，進而降低讀取效能。我們建議定期壓縮，將這些小型檔案合併為適當大小的檔案，以實現查詢效能。

選擇正確的更新策略

當您的使用案例接受最新資料的較慢讀取操作時，請使用merge-on-read 策略來最佳化寫入效能。

當您使用merge-on-read，Iceberg 會將更新寫入儲存，並刪除為個別的小型檔案。讀取資料表時，讀取器必須將這些變更與基本檔案合併，以傳回資料的最新檢視。這會導致讀取操作的效能懲罰，但可加快更新和刪除的寫入速度。一般而言，merge-on-read非常適合具有更新或任務的串流工作負載，這些更新較少，且分散在許多資料表分割區中。

您可以在資料表層級或應用程式端獨立設定merge-on-read組態 (`write.delete.mode`、`write.update.mode`和 `write.merge.mode`)。

使用merge-on-read需要執行定期壓縮，以防止讀取效能隨著時間降低。壓縮會與現有的資料檔案協調更新和刪除，以建立新的一組資料檔案，從而消除讀取端產生的效能損失。根據預設，除非您將 `delete-file-threshold` 屬性的預設值變更為較小的值，否則 Iceberg 的壓縮不會合併刪除檔案（請參閱 Iceberg 文件)。若要進一步了解壓縮，請參閱本指南稍後的 Iceberg 壓縮一節。

選擇正確的檔案格式

Iceberg 支援以 Parquet、ORC 和 Avro 格式寫入資料。Parquet 是預設格式。Parquet 和 ORC 是單欄式格式，可提供卓越的讀取效能，但寫入速度通常較慢。這代表讀取和寫入效能之間的典型取捨。

如果寫入速度對您的使用案例很重要，例如在串流工作負載中，請考慮在寫入器的選項Avro中write-format將 設定為 ，以 Avro 格式寫入。由於 Avro 是以資料列為基礎的格式，因此能以較慢的讀取效能提供更快的寫入時間。

為了改善讀取效能，請執行定期壓縮，將小型 Avro 檔案合併並轉換為較大的 Parquet 檔案。壓縮程序的結果由write.format.default資料表設定管理。Iceberg 的預設格式為 Parquet，因此如果您在 Avro 中寫入 ，然後執行壓縮，Iceberg 會將 Avro 檔案轉換為 Parquet 檔案。範例如下：

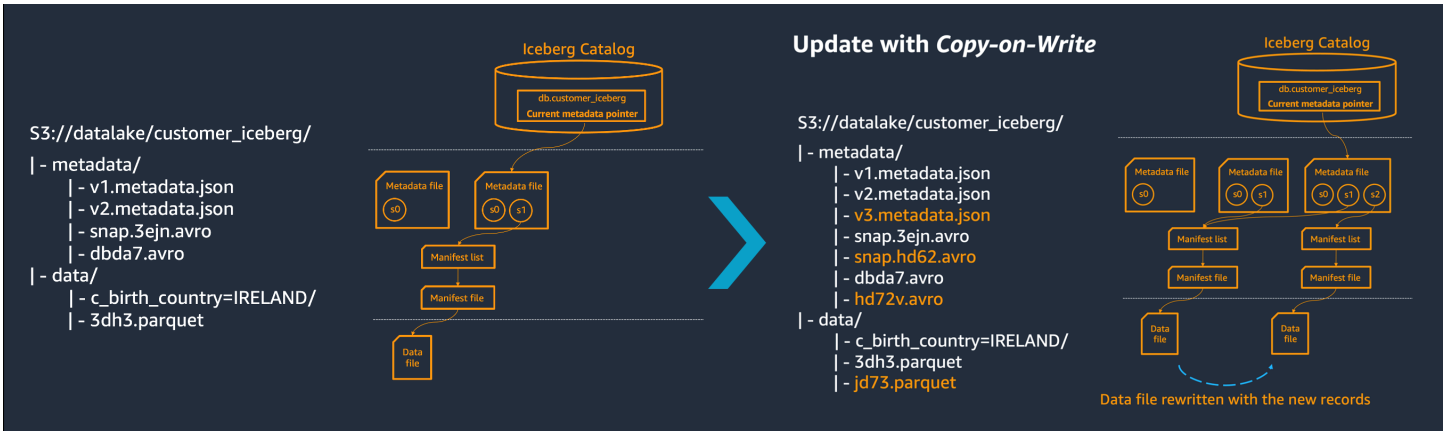
```
spark.sql(f"""
    CREATE TABLE IF NOT EXISTS glue_catalog.{DB_NAME}.{TABLE_NAME} (
        Col_1 float,
        <<<…other columns…>>
        ts timestamp)
    USING iceberg
    PARTITIONED BY (days(ts))
    OPTIONS (
      'format-version'='2',
      write.format.default'=parquet)
""")

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()
```

最佳化儲存體

更新或刪除 Iceberg 資料表中的資料會增加資料的複本數量，如下圖所示。執行壓縮也是如此：它會增加 Amazon S3 中的資料副本數量。這是因為 Iceberg 會將所有資料表的基礎檔案視為不可變。



遵循本節中的最佳實務來管理儲存成本。

啟用 S3 Intelligent-Tiering

使用 Amazon S3 Intelligent-Tiering 儲存類別，在存取模式變更時，自動將資料移至最具成本效益的存取層。此選項不會對效能造成操作額外負荷或影響。

注意：請勿在 S3 Intelligent-Tiering 搭配 Iceberg 資料表 中使用選用層 （例如 Archive Access 和 Deep Archive Access)。若要封存資料，請參閱下一節中的準則。

您也可以使用 Amazon S3 生命週期規則來設定自己的規則，將物件移至另一個 Amazon S3 儲存類別，例如 S3 Standard-IA 或 S3 One Zone-IA （請參閱 Amazon S3 文件中的支援的轉換和相關限制)。

封存或刪除歷史快照

對於 Iceberg 資料表的每個遞交交易 （插入、更新、合併、壓縮），會建立新的資料表版本或快照。隨著時間的推移，Amazon S3 中的版本數量和中繼資料檔案數量會累積。

快照隔離、資料表復原和時間歷程查詢等功能需要保留資料表的快照。不過，儲存成本會隨著您保留的版本數量而增加。

下表說明您可以實作的設計模式，以根據您的資料保留需求來管理成本。

設計模式	解決方案	使用案例
刪除舊快照	• 使用 Athena 中的 VACUUM 陳述式來移除舊快照。此操作不會產生任何運算成本。	此方法會刪除不再需要的快照，以降低儲存成本。您可以

設計模式	解決方案	使用案例
	• 或者，您可以在 Amazon EMR 上使用 Spark 或 AWS Glue 移除快照。如需詳細資訊，請參閱 Iceberg 文件中的 expire_snapshots。	根據您的資料保留需求，設定應保留多少快照或保留多久。 此選項會執行快照的硬刪除。您無法復原或將時間歷程轉返至過期的快照。
設定特定快照的保留政策	1. 使用標籤來標記特定快照，並在 Iceberg 中定義保留政策。如需詳細資訊，請參閱 Iceberg 文件中的歷史標籤。 例如，您可以在 Amazon EMR 上的 Spark 中使用下列 SQL 陳述式，每月保留一個快照，為期一年： `ALTER TABLE glue_catalog.db.table CREATE TAG 'EOM-01' AS OF VERSION 30 RETAIN 365 DAYS` 2. 在 Amazon EMR 上使用 Spark 或 AWS Glue 移除剩餘的未標記中繼快照。	此模式有助於遵守業務或法律要求，這些要求要求您在過去的特定時間點顯示資料表的狀態。透過在特定標記的快照上放置保留政策，您可以移除已建立的其他（未標記）快照。如此一來，您可以滿足資料保留需求，而無需保留建立的每個快照。

設計模式	解決方案	使用案例
封存舊快照	1. 使用 Amazon S3 標籤以 Spark 標記物件。(Amazon S3 標籤與 Iceberg 標籤不同；如需詳細資訊，請參閱 Iceberg 文件。) 例如： `spark.sql.catalog.my_catalog.s3.delete-enabled=false and \ spark.sql.catalog.my_catalog.s3.delete.tags.my_key=to_archive` 2. 在 Amazon EMR 上使用 Spark 或 AWS Glue 來移除快照。當您使用範例中的設定時，此程序會標記物件，並將其從 Iceberg 資料表中繼資料分離，而不是從 Amazon S3 刪除它們。 3. 使用 S3 生命週期規則，將標記為 的物件轉換為`to_archive` 其中一個 S3 Glacier 儲存類別。 4. 若要查詢封存的資料： • 還原封存的物件 （如果物件已轉換為 Amazon Glacier Instant Retrieval 儲存類別，則不需要此步驟）。 • 使用 Iceberg 中的 register_table 程序，將	此模式可讓您以較低的成本保留所有資料表版本和快照。 在未先將這些版本還原為新資料表的情況下，您無法安排時間或轉返至封存的快照。這通常適用於稽核目的。 您可以結合此方法與先前的設計模式，為特定快照設定保留政策。

設計模式	解決方案	使用案例
	快照註冊為目錄中的資料表。 如需詳細說明，請參閱 AWS 部落格文章 改善 Apache Iceberg 資料表建置在 Amazon S3 資料湖上的操作效率。	

刪除孤立檔案

在某些情況下，Iceberg 應用程式可能會在您遞交交易之前失敗。這會在 Amazon S3 中留下資料檔案。由於沒有遞交，因此這些檔案不會與任何資料表相關聯，因此您可能必須以非同步方式清除它們。

若要處理這些刪除，您可以在 Amazon Athena 中使用 VACUUM 陳述式。此陳述式會移除快照，也會刪除孤立的檔案。這非常符合成本效益，因為 Athena 不會收取此操作的運算成本。此外，使用 VACUUM陳述式時，您不需要排程任何其他操作。

或者，您可以在 Amazon EMR 上使用 Spark 或 AWS Glue 來執行`remove_orphan_files`程序。此操作具有運算成本，必須獨立排程。如需詳細資訊，請參閱 Iceberg 文件。

使用壓縮來維護資料表

Iceberg 包含的功能可讓您在將資料寫入資料表後執行資料表維護操作。有些維護操作著重於簡化中繼資料檔案，而其他維護操作則強化資料在檔案中的叢集方式，以便查詢引擎可以有效地找到必要資訊來回應使用者請求。本節著重於壓縮相關最佳化。

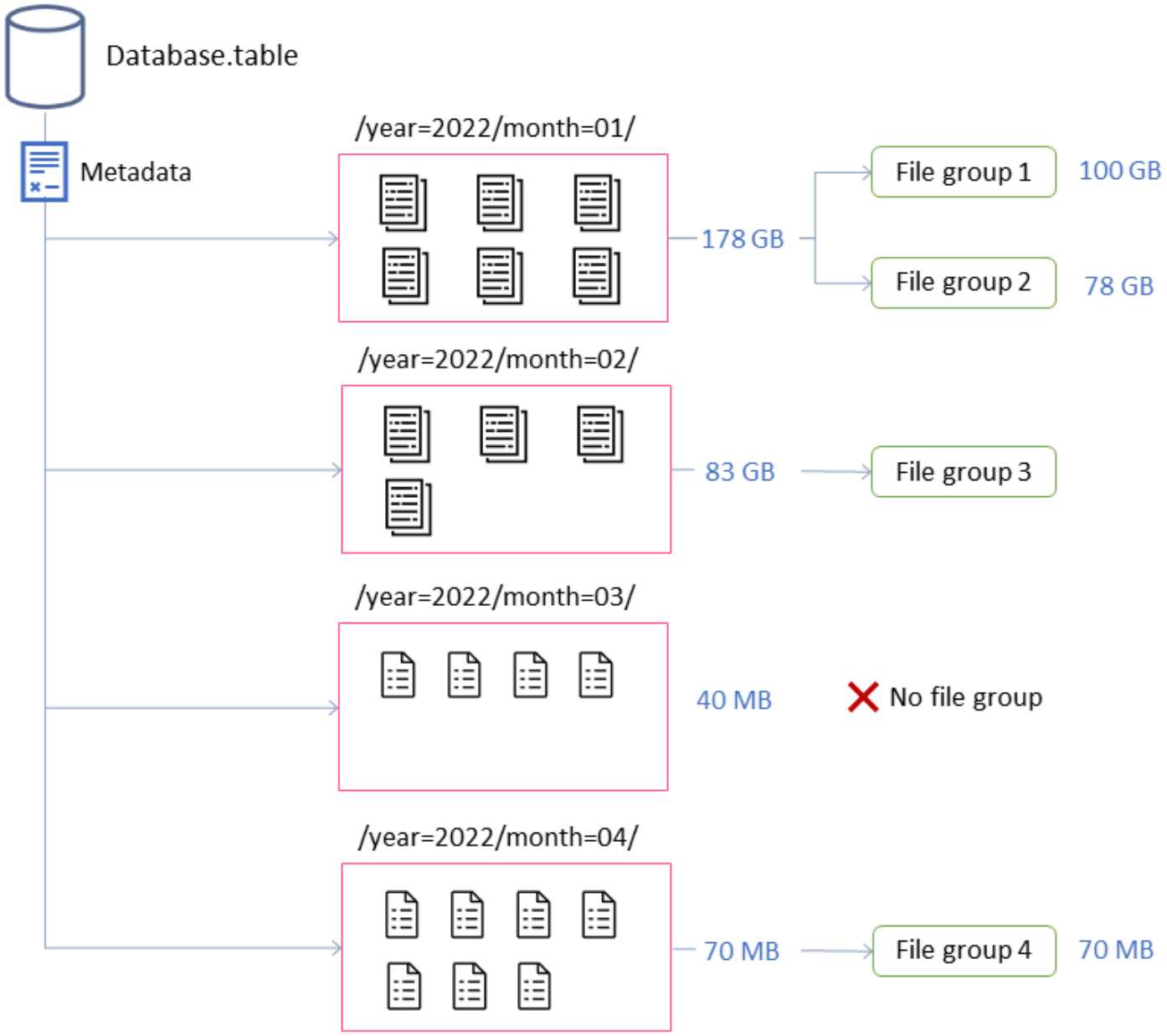
Iceberg 壓縮

在 Iceberg 中，您可以使用壓縮來執行四個任務：

- 將小型檔案合併成大小通常超過 100 MB 的大型檔案。此技術稱為 bin packing。
- 將刪除檔案與資料檔案合併。使用merge-on-read方法的更新或刪除會產生刪除檔案。
- （重新）根據查詢模式排序資料。無需任何排序順序或適合寫入和更新的排序順序即可寫入資料。
- 使用空間填入曲線來叢集資料，以最佳化不同的查詢模式，特別是 z 順序排序。

在 上 AWS，您可以透過 Amazon Athena 或在 Amazon EMR 或 中使用 Spark 來執行 Iceberg 的資料表壓縮和維護操作 AWS Glue。

當您使用 rewrite_data_files 程序執行壓縮時，您可以調整數個旋鈕來控制壓縮行為。下圖顯示 bin 封裝的預設行為。了解 bin 封裝壓縮是了解階層排序和 Z 順序排序實作的關鍵，因為它們是 bin 封裝界面的延伸，並以類似方式操作。主要區別是排序或叢集資料所需的額外步驟。



在此範例中，Iceberg 資料表包含四個分割區。每個分割區都有不同的大小和不同的檔案數量。如果您啟動 Spark 應用程式來執行壓縮，應用程式會建立總共四個要處理的檔案群組。檔案群組是 Iceberg

抽象，代表將由單一 Spark 任務處理的檔案集合。也就是說，執行壓縮的 Spark 應用程式將建立四個 Spark 任務來處理資料。

調校壓縮行為

下列金鑰屬性會控制如何選取資料檔案進行壓縮：

- MAX_FILE_GROUP_SIZE_BYTES 預設會將單一檔案群組 (Spark 任務） 的資料限制設為 100 GB。此屬性對於沒有分割區的資料表或分割區跨越數百 GB 的資料表特別重要。透過設定此限制，您可以細分操作來規劃工作並取得進展，同時防止叢集上的資源耗盡。

  注意：每個檔案群組都會個別排序。因此，如果您想要執行分割區層級排序，您必須調整此限制以符合分割區大小。

- MIN_FILE_SIZE_BYTES 或 MIN_FILE_SIZE_DEFAULT_RATIO 預設為資料表層級所設定目標檔案大小的 75%。例如，如果資料表的目標大小為 512 MB，則任何小於 384 MB 的檔案都會包含在將壓縮的一組檔案中。
- MAX_FILE_SIZE_BYTES 或 MAX_FILE_SIZE_DEFAULT_RATIO 預設為目標檔案大小的 180%。與設定最小檔案大小的兩個屬性一樣，這些屬性用於識別壓縮任務的候選檔案。
- 如果資料表分割區大小小於目標檔案大小，MIN_INPUT_FILES 會指定要壓縮的檔案數目下限。此屬性的值用於判斷根據檔案數量壓縮檔案是否值得 （ 預設為 5)。
- DELETE_FILE_THRESHOLD 會指定檔案在壓縮前的刪除操作數目下限。除非您另有指定，否則壓縮不會將刪除檔案與資料檔案合併。若要啟用此功能，您必須使用此屬性來設定閾值。此閾值專屬於個別資料檔案，因此如果您將其設定為 3，只有在有三個或多個參考該檔案的刪除檔案時，才會重新寫入資料檔案。

這些屬性可讓您深入了解上圖中檔案群組的形成。

例如，標示的分割區`month=01`包含兩個檔案群組，因為它超過 100 GB 的大小限制上限。相反地，`month=02`分割區包含單一檔案群組，因為它小於 100 GB。`month=03` 分割區不符合 5 個檔案的預設最低輸入檔案需求。因此，它不會壓縮。最後，雖然`month=04`分割區的資料不足以形成所需大小的單一檔案，但檔案會壓縮，因為分割區包含五個以上的小型檔案。

您可以為在 Amazon EMR 或 上執行的 Spark 設定這些參數 AWS Glue。對於 Amazon Athena，您可以使用以字首 開頭的資料表屬性來管理類似的屬性`optimize_`)。

在 Amazon EMR 或 上使用 Spark 執行壓縮 AWS Glue

本節說明如何正確調整 Spark 叢集的大小，以執行 Iceberg 的壓縮公用程式。下列範例使用 Amazon EMR Serverless，但您可以在 Amazon EMR on EC2 或 EKS 或 中使用相同的方法 AWS Glue。

您可以利用檔案群組與 Spark 任務之間的相互關聯來規劃叢集資源。若要循序處理檔案群組，請考慮每個檔案群組的大小上限為 100 GB，您可以設定下列 Spark 屬性：

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

如果您想要加速壓縮，您可以透過增加平行壓縮的檔案群組數量來水平擴展。您也可以使用手動或動態擴展來擴展 Amazon EMR。

- 手動擴展 （例如，因數為 4)
  - `MAX_CONCURRENT_FILE_GROUP_REWRITES = 4`（我們的因素）
  - `spark.executor.instances = 5`（範例中使用的值）x 4（我們的因素）= 20
  - `spark.dynamicAllocation.enabled = FALSE`
- 動態擴展
  - `spark.dynamicAllocation.enabled = TRUE` （預設，無需採取任何動作）
  - `MAX_CONCURRENT_FILE_GROUP_REWRITES = N` （將此值與 對齊`spark.dynamicAllocation.maxExecutors`，預設為 100；根據範例中的執行器組態，您可以將 N 設定為 20)

  這些是協助調整叢集大小的指導方針。不過，您也應該監控 Spark 任務的效能，以尋找工作負載的最佳設定。

使用 Amazon Athena 執行壓縮

Athena 透過 OPTIMIZE 陳述式，將 Iceberg 的壓縮公用程式實作為受管功能。您可以使用此陳述式來執行壓縮，而不必評估基礎設施。

此陳述式使用 bin 封裝演算法將小型檔案分組為較大的檔案，並將刪除檔案與現有的資料檔案合併。若要使用階層排序或 z 順序排序來叢集資料，請在 Amazon EMR 或 上使用 Spark AWS Glue。

您可以在建立資料表時，透過傳遞 CREATE TABLE OPTIMIZE陳述式中的資料表屬性，或使用 陳述式在建立資料表之後，變更 ALTER TABLE陳述式的預設行為。如需預設值，請參閱 Athena 文件。

執行壓縮的建議

使用案例	建議
根據排程執行 bin 封裝壓縮	• 如果您不知道資料表包含多少小型檔案，請在 Athena 中使用 OPTIMIZE陳述式。Athena 定價模型是以掃描的資料為基礎，因此，如果沒有要壓縮的檔案，則沒有與這些操作相關聯的成本。為了避免在 Athena 資料表上遇到逾時，OPTIMIZE請per-table-partition執行 。 • 當您預期壓縮大量小型檔案時，請使用 Amazon EMR 或 AWS Glue 搭配動態擴展。
根據事件執行 bin 封裝壓縮	• 當您預期壓縮大量小型檔案時，請使用 Amazon EMR 或 AWS Glue 搭配動態擴展。
執行壓縮以排序資料	• 使用 Amazon EMR 或 AWS Glue，因為排序是一項昂貴的操作，可能需要將資料溢出到磁碟。
執行壓縮以使用 z 順序排序來叢集資料	• 使用 Amazon EMR 或 AWS Glue，因為 z 順序排序是非常昂貴的操作，可能需要將資料溢出到磁碟。
在可能因資料延遲抵達而由其他應用程式更新的分割區上執行壓縮	• 使用 Amazon EMR 或 AWS Glue。啟用 Iceberg PARTIAL_PROGRESS_ENABLED 屬性。當您使用此選項時，Iceberg 會將壓縮輸出分割成多個遞交。如果有碰撞 （亦即，如果資料檔案在壓縮執行時更新），則此設定會將其限制為包含受影響檔案的遞交，以降低重試成本。否則，您可能需要重新編譯所有檔案。
在冷分割區上執行壓縮 （不再接收作用中寫入的資料分割區）	• 使用 Amazon EMR 或 AWS Glue。在rewrite_data_files 程序中，指定排

使用案例	建議
	除主動寫入分割區的`where`述詞。此策略可防止寫入器和壓縮任務之間的資料衝突，並只留下 Iceberg 可以自動解決的中繼資料衝突。

在 Amazon S3 中使用 Iceberg 工作負載

本節討論 Iceberg 屬性，您可以用來最佳化 Iceberg 與 Amazon S3 的互動。

防止熱分割 (HTTP 503 錯誤)

在 Amazon S3 上執行的某些資料湖應用程式會處理數百萬或數十億個物件，並處理 PB 的資料。這可能會導致接收大量流量的字首，這通常透過 HTTP 503 （服務無法使用） 錯誤偵測到。若要避免此問題，請使用下列 Iceberg 屬性：

- 將 `write.distribution-mode`設定為 `hash`或 ，`range`讓 Iceberg 寫入大型檔案，這會導致較少的 Amazon S3 請求。這是偏好的組態，應處理大多數案例。
- 如果您因為工作負載中的大量資料而持續遇到 503 錯誤，您可以在 Iceberg `true`中`write.object-storage.enabled`將 設定為 。這會指示 Iceberg 雜湊物件名稱，並將負載分散到多個隨機的 Amazon S3 字首。

如需這些屬性的詳細資訊，請參閱 Iceberg 文件中的寫入屬性。

使用 Iceberg 維護操作來釋出未使用的資料

若要管理 Iceberg 資料表，您可以使用 Iceberg 核心 API、Iceberg 用戶端 （例如 Spark) 或 Amazon Athena 等受管服務。若要從 Amazon S3 刪除舊或未使用的檔案，建議您僅使用 Iceberg 原生 APIs 來移除快照、移除舊中繼資料檔案，以及刪除孤立檔案。

透過 Boto3、Amazon S3 SDK 或 AWS Command Line Interface (AWS CLI) 使用 Amazon S3 APIs，或使用任何其他非 Iceberg 方法來覆寫或移除 Iceberg 資料表的 Amazon S3 檔案，會導致資料表損毀和查詢失敗。

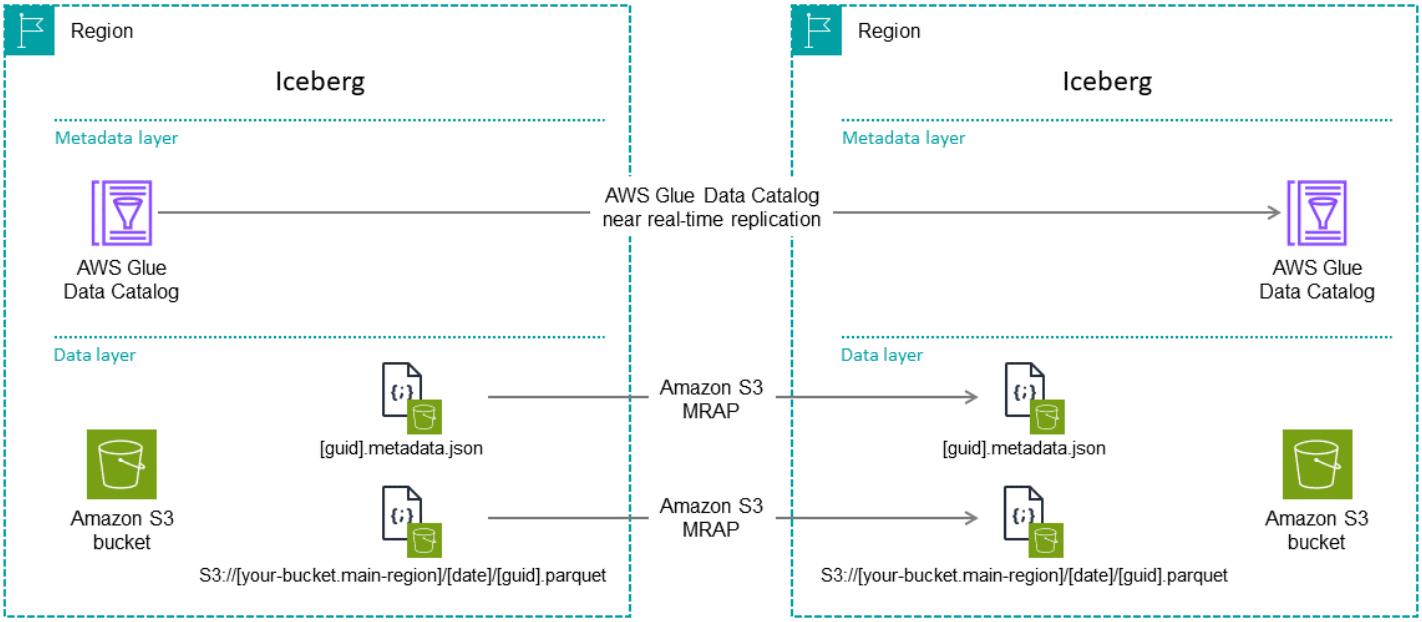
跨 複寫資料 AWS 區域

在 Amazon S3 中存放 Iceberg 資料表時，您可以使用 Amazon S3 中的內建功能，例如跨區域複寫 (CRR) 和多區域存取點 (MRAP)，跨多個區域複寫資料 AWS 區域。MRAP 為應用程式提供全域端點，

以存取位於多個 中的 S3 儲存貯體 AWS 區域。Iceberg 不支援相對路徑，但您可以透過將儲存貯體映射至存取點，使用 MRAP 來執行 Amazon S3 操作。MRAP 也會與 Amazon S3 跨區域複寫程序無縫整合，並引入高達 15 分鐘的延遲。您必須複寫資料和中繼資料檔案。

> **Important**
>
> 目前，與 MRAP 的 Iceberg 整合僅適用於 Apache Spark。如果您需要容錯移轉至次要 AWS 區域，您必須計劃將使用者查詢重新導向至容錯移轉區域中的 Spark SQL 環境 （ 例如 Amazon EMR)。

CRR 和 MRAP 功能可協助您為 Iceberg 資料表建置跨區域複寫解決方案，如下圖所示。



若要設定此跨區域複寫架構：

1. 使用 MRAP 位置建立資料表。這可確保 Iceberg 中繼資料檔案指向 MRAP 位置，而不是實體儲存貯體位置。
2. 使用 Amazon S3 MRAP 複寫 Iceberg 檔案。 MRAP 支援服務層級協議 (SLA) 為 15 分鐘的資料複寫。Iceberg 可防止讀取操作在複寫期間引入不一致。
3. 在次要區域中，讓 AWS Glue Data Catalog 中的資料表可用。您可以從兩個選項中選擇：
   - 設定管道以使用複寫來 AWS Glue Data Catalog 複寫 Iceberg 資料表中繼資料。此公用程式可在 GitHub Glue Catalog 和 Lake Formation Permissions 複寫儲存庫中使用。此事件驅動機制會根據事件日誌複寫目標區域中的資料表。

- 當您需要容錯移轉時，在次要區域中註冊資料表。對於此選項，您可以使用先前的公用程式或 Iceberg register_table 程序，並將其指向最新的`metadata.json`檔案。

監控 Apache Iceberg 工作負載

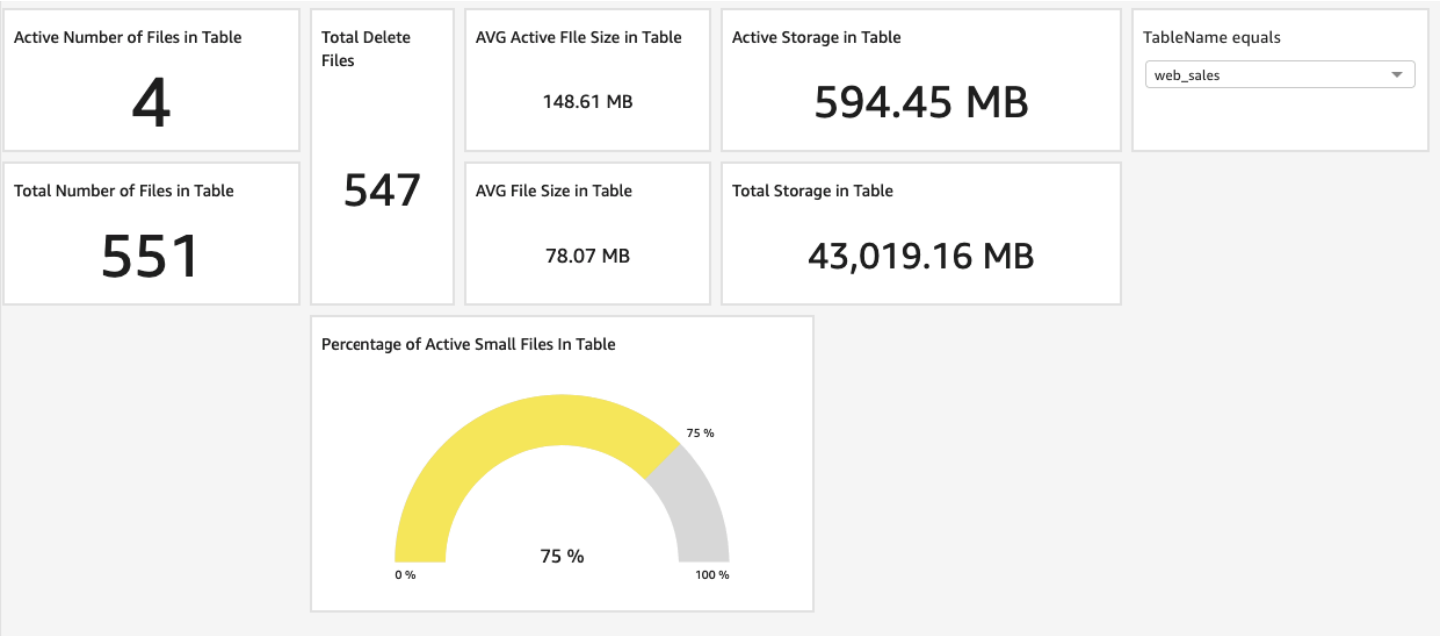
若要監控 Iceberg 工作負載，您有兩個選項：分析中繼資料表或使用指標報告程式。指標報告程式在 Iceberg 1.2 版中推出，僅適用於 REST 和 JDBC 目錄。

如果您使用的是 AWS Glue Data Catalog，您可以在 Iceberg 公開的中繼資料資料表上設定監控，以深入了解 Iceberg 資料表的運作狀態。

監控對於效能管理和疑難排解至關重要。例如，當 Iceberg 資料表中的分割區達到特定百分比的小型檔案時，您的工作負載可以啟動壓縮任務，將檔案合併成較大的檔案。這可防止查詢減速超過可接受的層級。

資料表層級監控

下列畫面顯示在 Amazon Quick Sight 中建立的資料表監控儀表板。此儀表板會使用 Spark SQL 查詢 Iceberg 中繼資料表，並擷取詳細的指標，例如作用中檔案的數量和總儲存體。然後，此資訊會存放在 AWS Glue 資料表中，以供操作之用。最後，使用 Amazon Athena 建立 Quick Sight 儀表板，如下圖所示。此資訊可協助您識別和解決系統中的特定問題。



Quick Sight 儀表板範例會收集 Iceberg 資料表的下列關鍵效能指標 (KPIs)：

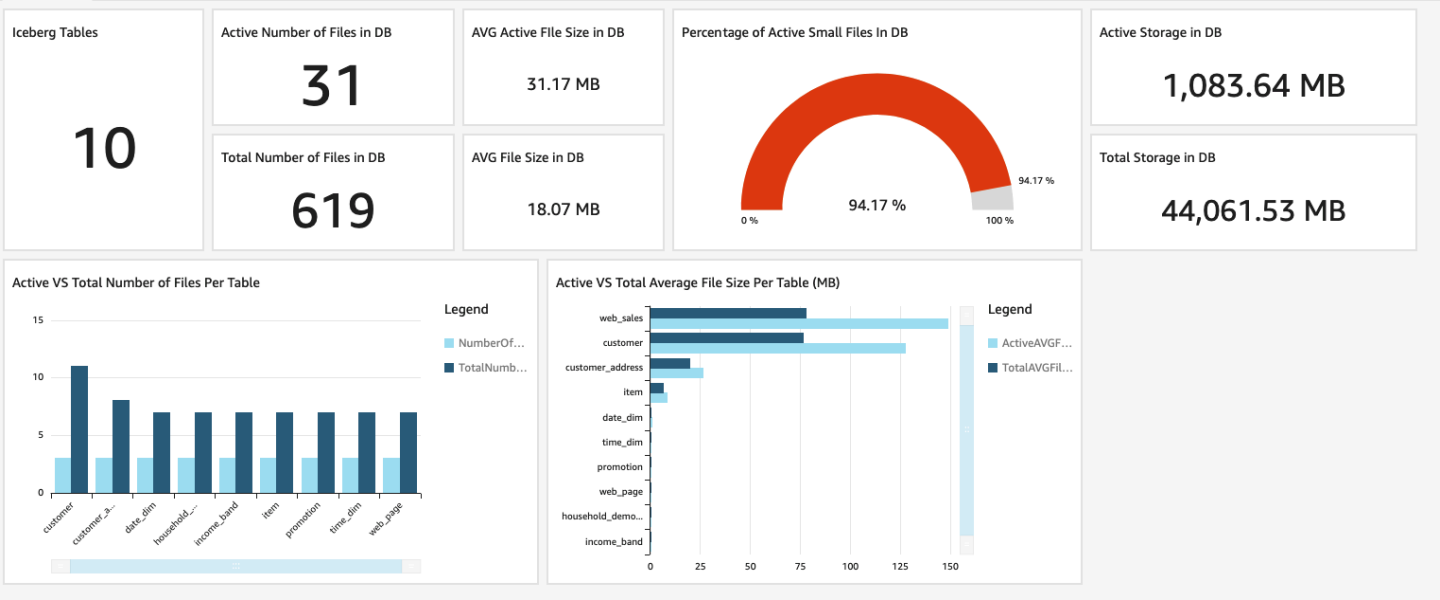
KPI	Description	Query
檔案數量	Iceberg 資料表中的檔案數目（適用於所有快照）	`select count(*) from <catalog.database.table_name>.all_files`
作用中檔案的數量	Iceberg 資料表最後一個快照中的作用中檔案數目	`select count(*) from <catalog.database.table_name>.files`
平均檔案大小	Iceberg 資料表中所有檔案的平均檔案大小，以 MB 為單位	`select avg(file_size_in_bytes)/1000000 from <catalog.database.table_name>.all_files`
平均作用中檔案大小	Iceberg 資料表中作用中檔案的平均檔案大小，以 MB 為單位	`select avg(file_size_in_bytes)/1000000 from <catalog.database.table_name>.files`
小型檔案的百分比	小於 100 MB 的作用中檔案百分比	`select cast(sum( case when file_size_in_bytes < 100000000 then 1 else 0 end)*100/count(*) as decimal(10,2)) from <catalog.database.table_name>.files`
總儲存體大小	資料表中所有檔案的總大小，不包括孤立檔案和 Amazon S3 物件版本（如果啟用）	`select sum(file_size_in_bytes)/1000000 from <catalog.database.table_name>.all_files`

KPI	Description	Query
作用中儲存體大小總計	指定資料表目前快照中所有檔案的總大小	`select sum(file_size_in_bytes)/1000000 from <catalog.database.table_name>.files`

如需建立儀表板的詳細資訊，請參閱 Quick Sight 文件。

資料庫層級監控

下列範例顯示在 Quick Sight 中建立的監控儀表板，以提供 Iceberg 資料表集合的資料庫層級 KPIs概觀。



此儀表板會收集下列 KPIs：

KPI	Description	Query
檔案數量	Iceberg 資料庫中的檔案數量（適用於所有快照）	此儀表板使用上一節提供的資料表層級查詢，並合併結果。

KPI	Description	Query
作用中檔案的數量	Iceberg 資料庫中作用中檔案的數量 （根據 Iceberg 資料表的最後一個快照）	
平均檔案大小	Iceberg 資料庫中所有檔案的平均檔案大小，以 MB 為單位	
平均作用中檔案大小	Iceberg 資料庫中所有作用中檔案的平均檔案大小，以 MB 為單位	
小型檔案的百分比	Iceberg 資料庫中小於 100 MB 的作用中檔案百分比	
總儲存體大小	資料庫中所有檔案的總大小，不包括孤立檔案和 Amazon S3 物件版本 （如果啟用）	
作用中儲存體大小總計	資料庫中所有資料表目前快照中所有檔案的總大小	

預防性維護

透過設定先前章節中討論的監控功能，您可以從預防性而非被動角度接近資料表維護。例如，您可以使用資料表層級和資料庫層級指標來排程動作，如下所示：

- 當資料表達到 N 個小型檔案時，使用 bin 封裝壓縮來分組小型檔案。
- 當資料表達到指定分割區中的 N 個刪除檔案時，使用 bin 封裝壓縮來合併刪除檔案。
- 當總儲存體比作用中儲存體高 X 倍時，移除快照，移除已壓縮的小型檔案。

阿帕奇冰山上的治理和訪問控制 AWS

阿帕奇冰山與整合 AWS Lake Formation 以簡化資料控管。此整合可讓資料湖管理員將儲存格層級存取權限指派給 Iceberg 表格。如需使用 Amazon Athena 查詢冰山資料表的範例 AWS Lake Formation，請參閱部 AWS 落格文章使用 Amazon Athena 與 Apache 冰山表互動，以及使用. AWS Lake Formation

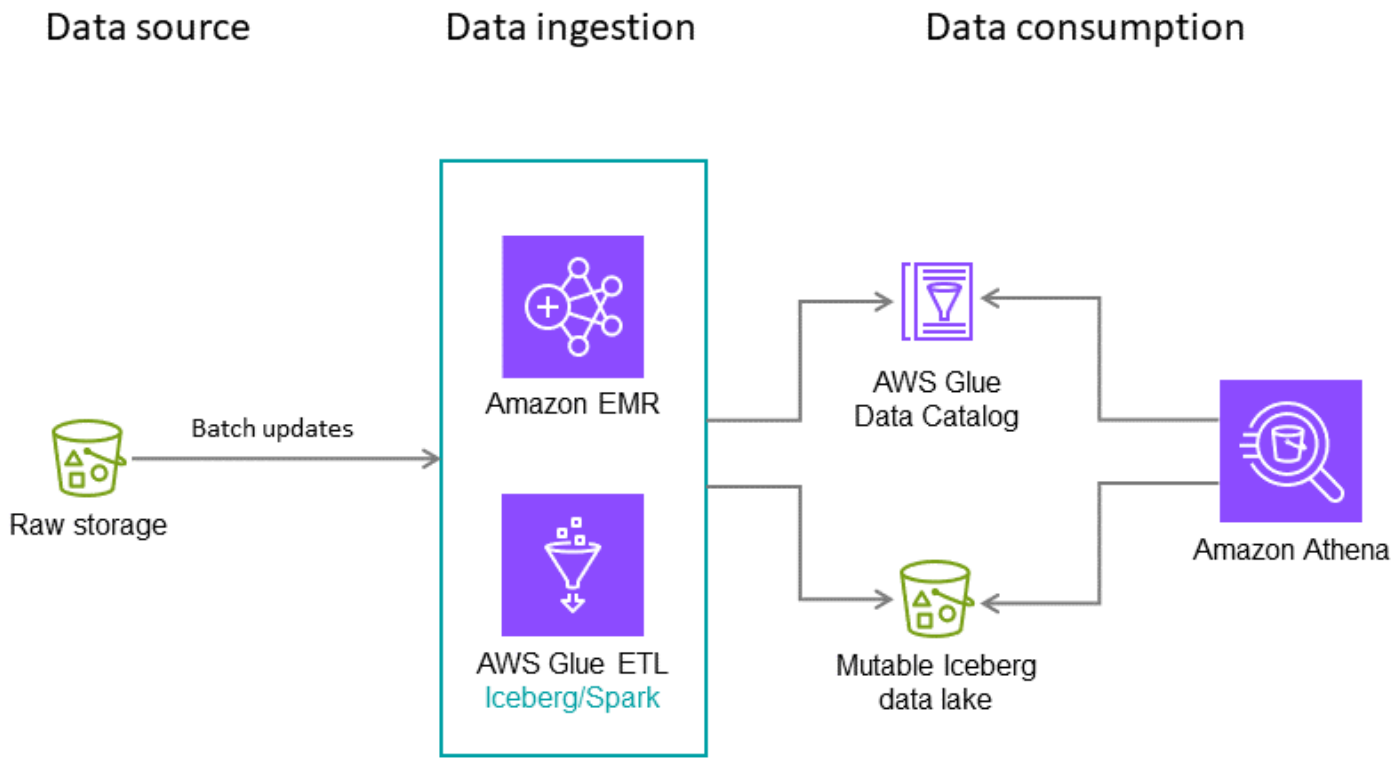
上的 Apache Iceberg 參考架構 AWS

本節提供在不同使用案例中套用最佳實務的範例，例如批次擷取，以及結合批次和串流資料擷取的資料湖。

每天批次擷取

針對此假設性使用案例，假設您的 Iceberg 資料表每晚擷取信用卡交易。每個批次只包含增量更新，這些更新必須合併到目標資料表中。每年收到幾次完整的歷史資料。在此案例中，我們建議使用下列架構和組態。

注意：這只是範例。最佳組態取決於您的資料和需求。



建議：

- 檔案大小：128 MB，因為 Apache Spark 任務會以 128 MB 區塊處理資料。
- 寫入類型：copy-on-write。如本指南先前所述，此方法有助於確保以讀取最佳化的方式寫入資料。
- 分割區變數：year/month/day。在我們的假設使用案例中，我們最常查詢最近的資料，雖然我們偶爾會執行過去兩年資料的完整資料表掃描。分割的目標是根據使用案例的需求來推動快速讀取操作。

- 排序順序：時間戳記
- 資料目錄： AWS Glue Data Catalog

結合批次和近乎即時擷取的資料湖

您可以在 Amazon S3 上佈建資料湖，以跨帳戶和區域共用批次和串流資料。如需架構圖表和詳細資訊，請參閱 AWS 部落格文章使用 Apache Iceberg 建置交易資料湖 AWS Glue，以及使用 AWS Lake Formation 和 Amazon Athena 進行跨帳戶資料共用。

Resources

- 在 中使用 Iceberg 架構 AWS Glue (AWS Glue 文件)
- Iceberg (Amazon EMR 文件)
- 使用 Apache Iceberg 資料表 (Amazon Athena 文件)
- Amazon S3 文件
- 快速文件
- Glue Catalog 和 Lake Formation 許可複寫 (GitHub 儲存庫)
- Apache Iceberg 文件
- Apache Spark 文件

貢獻者

AWS 撰寫、共同撰寫和檢閱本指南的下列人員。

貢獻者

- Stefano Sandona，大數據解決方案架構師
- Imtiaz (Taz) Sayed，解決方案架構師技術負責人，分析
- Shana Schipers，大數據解決方案架構師
- Prashant Singh，Amazon EMR 軟體開發工程師
- Arun A K，大數據和 ETL 解決方案架構師
- Francisco Morillo， 串流解決方案架構師
- Suthan Phillips，Amazon EMR 分析架構師
- Sercan Karaoglu，解決方案架構師
- Yonatan Dolan，分析專家
- Guy Bachar，解決方案架構師
- Sofia Zilberman， 串流解決方案架構師
- Dan Stair，專家解決方案架構師
- Sakti Mishra，解決方案架構師
- Ron Ortloff，Amazon S3 首席產品經理
- David Zhang，解決方案架構師，分析

檢閱者

- Rick Sears，Amazon EMR 總經理
- Linda OConnor，Amazon EMR 專家
- Ian Meyers，Amazon EMR 總監
- Vinita Ananth，Amazon EMR 產品管理總監
- Jason Berkowitz，產品經理 AWS Lake Formation
- Mahesh Mishra，Amazon Redshift 產品經理
- Vladimir Zlatkin，大數據解決方案架構經理
- Karthik Prabhakar，Amazon EMR 分析架構師

- Vijay Jain，產品經理
- Anupriti Warade，Amazon S3 產品經理
- Ajit Tandale，解決方案架構師，資料
- Gwen Chen，產品行銷經理
- Noritaka Sekiyama，大數據首席架構師

文件歷史紀錄

下表描述了本指南的重大變更。如果您想收到有關未來更新的通知，可以訂閱 RSS 摘要。

變更	描述	日期
新增章節	新增使用 Iceberg 資料表格式規格第 3 版的相關資訊。	2025 年 11 月 26 日
更正	更正快照程序區段中`gc.enabled` 設定的相關資訊。	2025 年 10 月 24 日
新增項目	新增了使用 Trino 和 PyIceberg 來使用 Iceberg 資料表的新章節，並擴展了將資料表遷移至 Iceberg 的章節。	2025 年 8 月 12 日
更新	強化並釐清指南中的資訊，以反映 AWS Glue、Amazon EMR 和 Apache Iceberg 的最新版本。	2025 年 7 月 14 日
新增項目	新增使用 Amazon Data Firehose 處理 Iceberg 資料表的新章節。	2025 年 2 月 20 日
初次出版	—	2024 年 4 月 30 日

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。