在 上使用 Apache Iceberg AWS

AWS 方案指引

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS 方案指引: 在 上使用 Apache Iceberg AWS

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon 的商標和商業外觀不得用於任何非 Amazon 的產品或服務，也不能以任何可能造成客戶混淆、任何貶低或使 Amazon 名譽受損的方式使用 Amazon 的商標和商業外觀。所有其他非 Amazon 擁有的商標均為其各自擁有者的財產，這些擁有者可能附屬於 Amazon，或與 Amazon 有合作關係，亦或受到 Amazon 贊助。

Table of Contents

簡介 ... 1
現代資料湖 ... 2
現代資料湖中的進階使用案例 ... 2
Apache Iceberg 簡介 ... 2
AWS 支援 Apache Iceberg ... 3
開始使用 Athena SQL 中的冰山表 ... 6
建立未分割資料表 ... 6
建立分割的資料表 ... 7
使用單一 CTAS 陳述式建立資料表並載入資料 ... 7
插入、更新和刪除資料 ... 8
查詢冰山表 ... 8
冰山, 表, 解剖學 ... 9
在 Amazon EMR 與冰山合作 ... 11
版本和功能兼容性 ... 11
使用冰山建立 Amazon EMR 叢集 ... 11
在 Amazon EMR 中開發冰山應用 ... 12
使用 Amazon EMR 工作室筆記本 ... 12
在 Amazon EMR 中執行冰山工作 ... 13
Amazon EMR 的最佳實踐 ... 17
與冰山合作 AWS Glue ... 19
使用原生冰山整合 ... 19
使用自定義的冰山版本 ... 19
使用自訂連接器 ... 20
使用您自己的 JAR 檔案 ... 21
冰山的火花配置 AWS Glue ... 21
AWS Glue 工作的最佳做法 ... 23
使用 Spark 處理 Iceberg 資料表 ... 24
建立和寫入 Iceberg 資料表 ... 24
使用 Spark SQL ... 24
使用 DataFrames API ... 25
更新 Iceberg 資料表中的資料 ... 26
在 Iceberg 資料表中備份資料 ... 27
刪除 Iceberg 資料表中的資料 ... 27
讀取資料 ... 28

使用時間歷程 28
使用增量查詢 29
存取中繼資料 29
使用 Firehose 處理 Iceberg 資料表 31
使用 Athena SQL 處理 Iceberg 資料表 32
版本和功能相容性 32
Iceberg 資料表規格支援 32
Iceberg 功能支援 32
使用 Iceberg 資料表 33
將現有資料表遷移至 Iceberg 34
就地遷移 34
完整資料遷移 39
選擇遷移策略 40
最佳化 Iceberg 工作負載的最佳實務 42
一般最佳實務 42
最佳化讀取效能 43
分割 43
調校檔案大小 45
最佳化資料欄統計資料 46
選擇正確的更新策略 47
使用 ZSTD 壓縮 47
設定排序順序 48
最佳化寫入效能 50
設定資料表分佈模式 50
選擇正確的更新策略 50
選擇正確的檔案格式 50
最佳化儲存體 51
啟用 S3 Intelligent-Tiering 52
封存或刪除歷史快照 52
刪除孤立檔案 55
使用壓縮來維護資料表 55
Iceberg 壓縮 55
調校壓縮行為 57
在 Amazon EMR 或 上使用 Spark 執行壓縮 AWS Glue 57
使用 Amazon Athena 執行壓縮 58
執行壓縮的建議 59

在 Amazon S3 中使用 Iceberg 工作負載 59
防止熱分割 (HTTP 503 錯誤) 60
使用 Iceberg 維護操作來釋出未使用的資料 60
跨 複寫資料 AWS 區域 60
監控 Iceberg 工作負載 62
資料表層級監控 62
資料庫層級監控 64
預防性維護 65
控管和存取權控制 66
參考架構 67
每晚批次擷取 67
結合批次和近乎即時擷取的資料湖 68
資源 69
貢獻者 70
文件歷史紀錄 72
詞彙表 73
..... 73
A 73
B 76
C 77
D 80
E 83
F 85
G 86
H 87
I 88
L 90
M 91
O 95
P 97
Q 99
R 99
S 102
T 105
U 106
V 107

W 107
Z 108
..... cix

在 上使用 Apache Iceberg AWS

Amazon Web Services （貢獻者）

2025 年 2 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 Apache 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。

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

本指南適用於在 上使用 Apache Iceberg 的任何人 AWS，從想要快速開始使用 Apache Iceberg 的新手使用者，到想要在其上最佳化和調整現有 Apache Iceberg 工作負載的進階使用者 AWS。

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 Amazon Data Firehose 處理 Iceberg 資料表
- 使用 Athena SQL 處理 Iceberg 資料表
- 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
- 參考架構
- 資源
- 貢獻者

現代資料湖

現代資料湖中的進階使用案例

資料湖提供在成本、可擴展性和彈性方面存放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、一致性、隔離和耐久性 (ACID) 處理語意的支援，您可能需要這些語意，才能透過使用多種不同的技術，在數百或數千名使用者之間大規模有效地最佳化和管理資料。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
- 隨著時間演進資料表結構描述，而不需要（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，這可能會導致資料表欠佳。

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。

為了協助克服這些挑戰，Apache 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 Simple Storage Service (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如下：

- 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，例如可供引擎用來加速規劃和執行查詢的分割區和資料欄層級統計資料。

- 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。
- 隱藏的分割。 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，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

Apache Iceberg 受熱門的開放原始碼資料處理架構和 支援， AWS 服務 例如 Amazon EMR、Amazon Athena、Amazon Redshift 和 AWS Glue。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 可以間接或封裝 Iceberg 程式庫的方式與 Iceberg 互動。

- Amazon S3 因其耐用性、可用性、可擴展性、安全性、合規性和稽核功能，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。
- Amazon EMRis 是一種大數據解決方案，適用於使用 Apache Spark、Flink、Trino 和 Hive 等開放原始碼架構的 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 Athenais 是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、靈活的方式來分析其所在位置的 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 是一種 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。 AWS Glue 3.0 和更新版本支援資料湖的 Iceberg 架構。您可以使用 AWS Glue 在 Amazon S3 中的 Iceberg 資料表上執行讀取和寫入操作，

或使用 來使用 Iceberg 資料表 AWS Glue Data Catalog。也支援其他操作，例如插入、更新、Spark 查詢和 Spark 寫入。

- AWS Glue Data Catalog提供支援 Iceberg 資料表的 Hive 中繼存放區相容資料目錄服務。
- AWS Glue 編目程式提供自動化以在 中註冊 Iceberg 資料表 AWS Glue Data Catalog。
- 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供應商擁有的任何自訂 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 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 具有支援 Iceberg 的廣泛服務，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 以及 Amazon Athena SQL 上的 Spark AWS Glue（批次和結構化串流）。下一節簡要介紹 Athena SQL 中的 Iceberg 支援。

在 Amazon Athena SQL 中開始使用阿帕奇冰山表

Amazon Athena 為 Apache 冰山提供內置的支持。除了設定 Athena 說明文件的入門一節中詳述的服務必要條件外，您無需任何其他步驟或組態即可使用 Iceberg。本節提供在 Athena 中建立資料表的簡要介紹。如需詳細資訊，請參閱本指南稍後的使用 Athena SQL 使用 Apache 冰山資料表。

您可以 AWS 通過使用不同的引擎創建冰山表。這些表跨無縫工作 AWS 服務。若要使用 Athena SQL 建立您的第一個冰山資料表，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'
)
```

以下各節提供在 Athena 中建立分割和未分割的 Iceberg 資料表的範例。如需詳細資訊，請參閱 Athena 文件中詳細說明的冰山語法。

建立未分割資料表

下列範例陳述式會自訂樣板 SQL 程式碼，以便在 Athena 建立未分割的 Iceberg 資料表。您可以將此陳述式新增至 Athenaconsole 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 指示，請參閱 Athena 文件中的入門指示。

建立分割的資料表

下列陳述式會使用 Iceberg 隱藏分割的概念，根據日期建立資料分割資料表。它會使用`day()`轉換，以時間戳記資料行的`dd-mm-yyyy`格式衍生每日分割區。Iceberg 不會將此值存儲為數據集中的新列。相反地，當您撰寫或查詢資料時，會即時衍生值。

```
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

在前幾節的已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` 或陳述式將資料載入資料`MERGE`表。或者，您可以使用`CREATE TABLE AS SELECT (CTAS)`語句在一個步驟中創建數據並將其加載到 Iceberg 表中。

CTAS 是 Athena 在單一陳述式中建立資料表並載入資料的最佳方式。下列範例說明如何使用 CTAS，從 Athena 現有的隱藏/鑲木地板資料表 (`iceberg_ctas_table`) 建立冰山資料表 (`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`、`UPDATE`、和 `DELETE FRO M` 陳述式，將資料寫入冰山資料表的不同方式。`MERGE INTO`

注意：`UPDATE`，`MERGE INTO`，並`DELETE FROM`使用位置刪除的 merge-on-read 方法。Athena SQL 目前不支援此 copy-on-write 方法。

例如，下列陳述式會使用將資料新增`INSERT INTO`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：

Results (1)

Copy | Download results

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

如需詳細資訊，請參閱 A thena 文件。

查詢冰山表

您可以使用 Athena SQL 對您的冰山資料表執行一般 SQL 查詢，如前面的範例所示。

除了常見的查詢之外，Athena 也支援冰山表格的時間旅行查詢。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，因此您可以方便地使用時間移動查詢根據時間戳記或快照 ID 回溯舊版資料表。

例如，下列陳述式會更新的顏色值`Person5`，然後顯示 2023 年 1 月 4 日起較早的值：

```
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

#	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

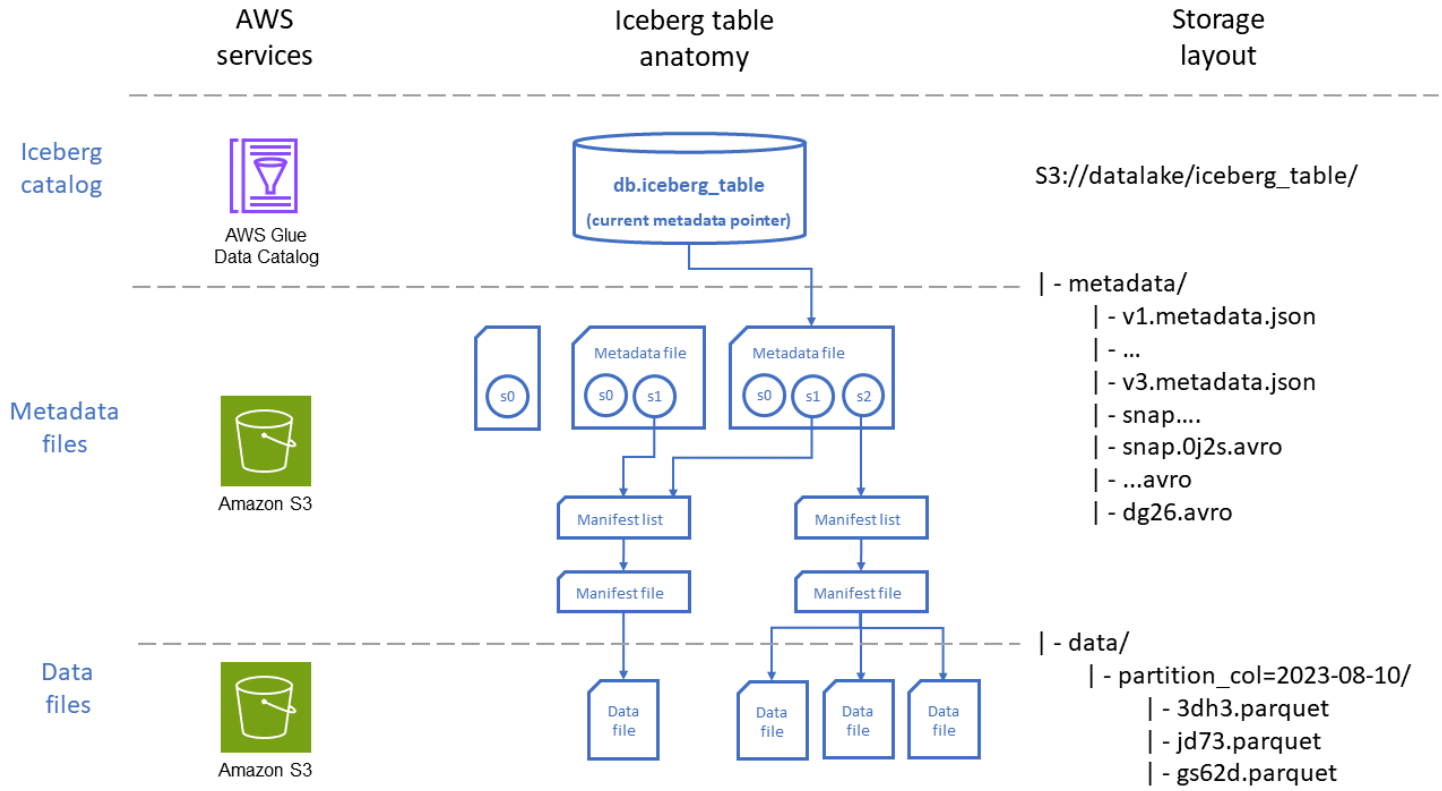
如需時間旅行查詢的語法和其他範例，請參閱 A thena 文件。

冰山, 表, 解剖學

現在，我們已經介紹了使用 Iceberg 桌子的基本步驟，讓我們深入了解冰山餐桌的複雜細節和設計。

為了啟用本指南前面描述的功能，Iceberg 採用了數據和元數據文件的分層層設計。這些層可以智慧地管理中繼資料，以最佳化查詢規劃和執行。

下圖透過兩個方面描繪了 Iceberg 表的組織： AWS 服務 用於在 Amazon S3 中存放資料表和檔案放置。



如圖所示，冰山表由三個主要層組成：

- Iceberg 目錄： AWS Glue Data Catalog 與 Iceberg 原生整合，對於大多數使用案例而言，是執行在. AWS與 Iceberg 資料表互動的服務 (例如 Athena) 會使用目錄尋找資料表的目前快照版本，以便讀取或寫入資料。
- 中繼資料層：中繼資料檔案 (亦即資訊清單檔案和資訊清單清單檔案) 會追蹤資訊，例如資料表的結構描述、資料分割策略和資料檔案位置，以及資料行層級統計資料 (例如儲存在每個資料檔案中之記錄的最小和最大範圍)。這些中繼資料檔案存放在表格路徑內的 Amazon S3 中。
  - 清單文件包含每個數據文件的記錄，包括其位置，格式，大小，校驗和和其他相關信息。
  - 資訊清單清單提供資訊清單檔案的索引。隨著資料表中的資訊清單檔案數量增加，將該資訊分解為較小的子區段有助於減少需要由查詢掃描的資訊清單檔案數目。
  - 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、分割中繼資料、快照檔案，以及用來管理資料表中繼資料的其他檔案。
- 數據層：該層包含具有查詢將運行的數據記錄的文件。這些文件可以存儲在不同的格式，包括阿帕奇鑲木地板，阿帕奇阿夫羅和 Apache 的 OR C。
  - 數據文件包含一個表的數據記錄。
  - 刪除文件對 Iceberg 表中的行級刪除和更新操作進行編碼。Iceberg 有兩種類型的刪除文件，如冰山文檔中所述。這些文件是通過操作通過使用 merge-on-read 模式創建的。

與阿帕奇冰山在 Amazon EMR 工作

Amazon EMR 使用開放原始碼架構 (例如 Apache Spark、Apache Hive、Flink 和 Trito)，在雲端提供 PB 規模的資料處理、互動式分析和機器學習。

> **Note**
>
> 本指南使用 Apache 星火為例。

Amazon EMR 支持多種部署選項：Amazon EC2 上的 Amazon EMR，Amazon EKS 上的 Amazon EMR，Amazon EMR 無服務器和亞馬遜 EMR。 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。

版本和功能兼容性

Amazon EMR 版本 6.5.0 及更高版本本地支持阿帕奇冰山。如需每個 Amazon EMR 版本支援的冰山版本清單，請參閱 Amazon EMR 文件中的冰山版本歷史記錄。 同時檢閱在 Amazon EMR 上使用 Iceberg 的考量和限制，以瞭解 Amazon EMR 在不同架構上支援哪些冰山功能。

我們建議您使用最新的 Amazon EMR 版本，以享受最新支援的冰山版本。本節中的程式碼範例和組態假設您使用的是 Amazon EMR 版本 em r-6.9.0。

使用冰山建立 Amazon EMR 叢集

若要在已安裝冰山的 Amazon EC2 上建立 Amazon EMR 叢集，請遵循亞馬遜 EM R 文件中的指示。

具體而言，您的叢集應設定下列分類：

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

您也可以選擇使用 Amazon EMR 無伺服器或 Amazon EKS 上的 Amazon EMR 作為您的冰山工作負載的部署選項，從 Amazon EMR 6.6.0 開始。

在 Amazon EMR 中開發冰山應用

若要為您的 Iceberg 應用程式開發 Spark 程式碼，您可以使用 Amazon EMR Studio，這是一個以網路為基礎的整合式開發環境 (IDE)，適用於在 Amazon EMR 叢集上執行的全受管 Jupyter 筆記本電腦。

使用 Amazon EMR 工作室筆記本

您可以在 Amazon EMR 工作室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 Amazon EC2 叢集上的 Amazon EMR 或 Amazon EKS 受管端點上的 Amazon EMR。有關在 Amazon Amazon EC2 和 Amazon EKS 上的 Amazon EMR 上設置 EMR 工作室的說明，請參閱 AWS 服務 文檔。

要在 EMR 工作室中使用冰山，請按照以下步驟操作：

1. 按照使用已安裝冰山的叢集中的指示，啟動已啟用冰山的 Amazon EMR 叢集。
2. 建立 EMR 工作室。如需指示，請參閱設定 Amazon EMR 工作室。
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>.catalog-impl":
 "org.apache.iceberg.aws.glue.GlueCatalog",
        "spark.sql.catalog.<catalog_name>.io-impl":
 "org.apache.iceberg.aws.s3.S3FileIO",
        "spark.sql.extensions":
 "org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions"
    }
}
```

其中：

- `<catalog_name>`是您的冰山星火工作階段目錄名稱。將其取代為目錄的名稱，並記得在與此型錄相關聯的所有模型組態中變更參考。然後，在您的程式碼中，您應該使用完整資料表名稱 (包括 Spark 工作階段目錄名稱) 來參考 Iceberg 資料表，如下所示：

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

- `<catalog_name>.warehouse`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`<catalog_name>.catalog-impl` 將設定為`org.apache.iceberg.aws.glue.GlueCatalog`。若要指向任何自訂目錄實作的實作類別，都需要此金鑰。本指南後面的「一般最佳做法」一節說明了不同的 Iceberg 支援的目錄。
- 用`org.apache.iceberg.aws.s3.S3FileIO`作，以便`<catalog_name>.io-impl`利用 Amazon S3 多部分上傳以實現高平行性。

4. 現在，您可以開始在筆記本中以互動方式開發 Iceberg 的 Spark 應用程序，就像您對任何其他 Spark 應用程序一樣。

如需使用 Amazon EMR Studio 為 Apache 冰山設定 Spark 的詳細資訊，請參閱部落格文章在 Amazon EMR 上使用 Apache 冰山建置高效能、ACID 合規且不斷發展的資料湖。

在 Amazon EMR 中執行冰山工作

開發適用於 Iceberg 工作負載的 Spark 應用程式程式碼後，您可以在任何支援 Iceberg 的 Amazon EMR 部署選項上執行該程式碼 (請參閱 Amazon EM R 常見問答集)。

與其他 Spark 任務一樣，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 Amazon EC2 叢集上的 Amazon EMR。若要執行星火任務，請參閱下列 Amazon EMR 文件頁面：

- 如需將工作提交至 Amazon EC2 叢集上 Amazon EMR 的不同選項概觀，以及每個選項的詳細指示，請參閱將工作提交到叢集。
- 對於 Amazon EKS 上的 Amazon EMR，請參閱使用運行 Spark 任務。 StartJobRun
- 如需 Amazon EMR 無伺服器資訊，請參閱執行任務。

以下各節提供每個 Amazon EMR 部署選項的範例。

Amazon 在亞馬 Amazon EC2 上的 EMR

您可以使用以下步驟提交冰山星火工作：

1. 在工作站上建立`emr_step_iceberg.json`包含下列內容的檔案：

```
[{
    "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>.catalog-
impl=org.apache.iceberg.aws.glue.GlueCatalog",
        "--conf",
        "spark.sql.catalog.<catalog_name>.warehouse=s3://YOUR-BUCKET-NAME/YOUR-
FOLDER-NAME/",
        "--conf",
        "spark.sql.catalog.<catalog_name>.io-
impl=org.apache.iceberg.aws.s3.S3FileIO",
        "s3://YOUR-BUCKET-NAME/code/iceberg-job.py"
    ]
}]
```

2. 通過自定義以粗體突出顯示的 Iceberg 配置選項來修改特定 Spark 作業的配置文件。
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

若要使用以下命令將冰山星火任務提交給 Amazon EMR 無伺服器： AWS CLI

1. 在工作站上建立`emr_serverless_iceberg.json`包含下列內容的檔案：

```
{
    "applicationId": "<APPLICATION_ID>",
    "executionRoleArn": "<ROLE_ARN>",
    "jobDriver": {
        "sparkSubmit": {
            "entryPoint": "s3://YOUR-BUCKET-NAME/code/iceberg-job.py",
```

```
            "entryPointArguments": [],
            "sparkSubmitParameters": "--jars /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>.catalog-impl":
 "org.apache.iceberg.aws.glue.GlueCatalog",
                "spark.sql.catalog.<catalog_name>.warehouse": "s3://YOUR-BUCKET-NAME/
YOUR-FOLDER-NAME/",
                "spark.sql.catalog.<catalog_name>.io-impl":
 "org.apache.iceberg.aws.s3.S3FileIO",
                "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. 通過自定義以粗體突出顯示的 Iceberg 配置選項來修改特定 Spark 作業的配置文件。
3. 使用提交工作 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 工作室主控台將冰山星火任務提交給 Amazon EMR 無伺服器：

1. 請遵循 Amazon EMR 無伺服器文件中的指示進行。

2. 對於 Job 配置，請使用為 Spark 提供的冰山配置， AWS CLI 並自定義冰山突出顯示的字段。如需詳細指示，請參閱 Amazon EMR 文件中的將 Apache 冰山與無伺服器搭配使用。

Amazon EMR 在 Amazon EKS

要通過使用以下命令向 Amazon EKS 上的 Amazon EMR 提交冰山火花任務： AWS CLI

1. 在工作站上建立`emr_eks_iceberg.json`包含下列內容的檔案：

```
{
    "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>.catalog-impl":
 "org.apache.iceberg.aws.glue.GlueCatalog",
                "spark.sql.catalog.<catalog_name>.warehouse": "s3://YOUR-BUCKET-NAME/
YOUR-FOLDER-NAME/",
                "spark.sql.catalog.<catalog_name>.io-impl":
 "org.apache.iceberg.aws.s3.S3FileIO",
                "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. 通過自定義以粗體突出顯示的 Iceberg 配置選項來修改 Spark 作業的配置文件。
3. 使用提交工作 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
```

有關詳細說明，請參閱在 EKS 上的 Amazon EMR 文檔中使用阿帕奇冰山與 Amazon EMR 在 EKS 上。

Amazon EMR 的最佳實踐

本節提供在 Amazon EMR 中調整 Spark 任務的一般準則，以最佳化資料讀取和寫入冰山資料表。有關特定於 Iceberg 的最佳實踐，請參閱本指南後面的最佳實踐部分。

- 使用最新版本的 Amazon EMR — Amazon EMR 通過 Amazon EMR 星火運行時提供開箱即用的 Spark 優化。 AWS 改善每個新版本的 Spark 執行階段引擎效能。
- 決定 Spark 工作負載的最佳基礎架構 — Spark 工作負載可能需要不同類型的硬體來處理不同的工作特性，以確保最佳效能。Amazon EMR 支援多種執行個體類型 (例如運算優化、記憶體優化、一般用途和儲存最佳化)，以涵蓋所有類型的處理需求。當您上架新的工作負載時，建議您使用一般執行個體類型 (例如 M5 或 M6g) 進行基準測試。監控 Ganglia 和 Amazon 的作業系統 (OS) 和 YARN 指標， CloudWatch 以判斷尖峰負載時的系統瓶頸 (CPU、記憶體、儲存和 I/O)，並選擇適當的硬體。
- 微調 `spark.sql.shuffle.partitions` — 將內`spark.sql.shuffle.partitions`容設定為叢集中虛擬核心 (vCore) 總數，或設定為該值的倍數 (通常是 vCore 總數的 1 到 2 倍)。當您使用雜湊和範圍分割做為寫入分配模式時，此設定會影響 Spark 的平行處理原則。它在寫入組織數據之前請求洗牌，以確保分區對齊。
- 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模式，建議您停用自動調整規模和動態配置。啟用受管擴展後，建議您使用 Spot 執行個體降低成本。針對任務節點使用 Spot 執行個體，而非核心或主節點。使用 Spot 執行個體時，請使用具有每個叢集多個執行個體類型的執行個體叢集，以確保 Spot 可用性
- 盡可能使用廣播聯接-廣播（mapside）聯接是最佳的聯接，只要您的其中一個表足夠小以適應最小節點的內存（按 MB 的順序），並且您正在執行 equi（=）連接。支援除完整外部聯結以外的所有聯

結類型。廣播聯接將較小的表作為哈希表廣播，跨內存中的所有工作節點。小桌子廣播後，您將無法對其進行更改。由於哈希表位於 Java 虛擬機（JVM）中的本地，因此可以通過使用散列連接，根據連接條件輕鬆地將其與大表合併。由於最小的隨機播放額外負荷，廣播連接可提供高效能。

- 調整記憶體回收器 — 如果記憶體回收 (GC) 週期很慢，請考慮從預設的 parallel 記憶體回收器切換至 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。要跟踪 GC 性能，您可以使用 Spark UI 對其進行監視。理想情況下，GC 時間應小於或等於總工作執行階段的 1%。

與阿帕奇冰山一起工作 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) 作業。這有助於分類您的資料、清理資料、豐富資料，以及在各種資料存放區和資料串流之間可靠地移動資料。

AWS Glue 工作封裝使用 Apache 星火或 Python 執行階段定義轉換邏輯的指令碼。 AWS Glue 工作可以在批次和串流模式下執行。

當您在中建立 Iceberg 工作時 AWS Glue，視版本而定 AWS Glue，您可以使用原生 Iceberg 整合或自訂 Iceberg 版本，將 Iceberg 相依性附加至工作。

使用原生冰山整合

AWS Glue 3.0 和 4.0 版本本地支持事務數據湖格式，如阿帕奇冰山，阿帕奇胡迪，和 Linux 基金會三角洲湖在 AWS Glue 星火。此整合功能可簡化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 Job 啟用 Iceberg 支援，請設定 Job：選擇工作的 [工作詳細資料] 索引標籤，捲動至 [進階屬性] 下的 [工作參數]，然後將金鑰設定為`--datalake-formats`及其值為`iceberg`。AWS Glue

如果您使用筆記本編寫工作，您可以使用下列`%%configure`魔法設定第一個筆記本儲存格中的參數：

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

使用自定義的冰山版本

在某些情況下，您可能希望保留對工作的 Iceberg 版本的控制權，並按照自己的步調對其進行升級。例如，升級到更新版本可以解除對新功能的存取權限和效能增強功能。若要搭配使用特定的 Iceberg 版本 AWS Glue，您可以使用自訂連接器或您自己的 JAR 檔案。

使用自訂連接器

AWS Glue 支援連接器，這是可選的程式碼套件，可協助存取 AWS Glue Studio. 您可以在中訂閱連接器 AWS Marketplace，也可以建立自訂連接器。

> **Note**
>
> AWS Marketplace 提供了阿帕奇冰山連接器 AWS Glue. 但是，我們建議您改用自訂連接器來保留對 Iceberg 版本的控制權。

例如，若要為 Iceberg 版本 0.13.1 建立客戶連接器，請依照下列步驟執行：

1. 將文件`iceberg-spark-runtime-3.1_2.12-0.13.1.jar`上`bundle-2.17.161.jar`傳`url-connection-client-2.17.161.jar`到 Amazon S3 存儲桶。您可以從各自的 Apache Maven 存儲庫中下載這些文件。
2. 在AWS Glue Studio 主控台上，建立自訂 Spark 連接器：
   a. 在導覽窗格中，選擇 [資料連線]。(如果您使用的是舊版導覽，請選擇 [連接器]、[建立自訂連接器])。
   b. 在 [連接器] 方塊中選擇 [建立自訂連接器]。
   c. 在 [建立自訂連接器] 頁面上：
      - 在 Amazon S3 中指定 JAR 檔案的路徑。
      - 輸入連接器的名稱。
      - 選擇星火作為連接器類型。
      - 對於 [類別名稱]，指定使用`format`運算子載入 Spark 資料來源時所使用的完整資料來源類別名稱 (或其別名)。
      - (選擇性) 提供連接器的描述。
3. 選擇 Create connector (建立連接器)。

在中使用連接器時 AWS Glue，您必須建立連接器的連接。連接包含連接至特定資料倉庫所需的性質。您在 ETL 任務中將連線用於資料來源和資料目標。連接器和連線搭配運作，以方便存取資料存放區。

若要使用您建立的自訂 Iceberg 連接器來建立連線：

1. 在AWS Glue Studio 主控台上，選取您的自訂 Iceberg 連接器。

2. 依照提示提供詳細資料 (例如 VPC 和工作所需的其他網路組態)，然後選擇 [建立連線]。

您現在可以在 AWS Glue ETL 工作中使用連線。視您建立工作的方式而定，有不同的方式可將連線附加至工作：

- 如果您使用建立視覺化工作 AWS Glue Studio，則可以從 [資料來源內容 — 連接器] 索引標籤上的 [連線] 清單中選取連線。
- 如果您在筆記本中開發工作，請使用`%connections`魔術來設置連接名稱：

```
%glue_version 3.0

%connections <name-of-the iceberg-connection>

%%configure
{
  "--conf" : "job-specific Spark configurations, to be discussed later",
  "--datalake-formats" : "iceberg"
}
```

- 如果您使用指令碼編輯器來編寫 Job，請在 [工作詳細資料] 索引標籤的 [進階內容]、[其他網路連線] 下方指定連線。

如需本節中程序的詳細資訊，請參閱 AWS Glue 文件 AWS Glue Studio中的使用連接器和連線。

使用您自己的 JAR 檔案

在中 AWS Glue，您也可以使用 Iceberg，而無需使用連接器。當您想要保留對 Iceberg 版本的控制權並快速更新時，此方法非常有用。若要使用此選項，請將所需的 Iceberg JAR 檔案上傳到您選擇的 S3 儲存貯體，並參考 AWS Glue 工作中的檔案。例如，如果您正在使用冰山 1.0.0，則所需的 JAR 文件是`iceberg-spark-runtime-3.0_2.12-1.0.0.jar``url-connection-client-2.15.40.jar`、和。`bundle-2.15.40.jar`您也可以將工作的`--user-jars-first`參數設定`true`為，來排定類別路徑中其他 JAR 檔案的優先順序。

冰山的火花配置 AWS Glue

本節討論為冰山資料集編寫 AWS Glue ETL 工作所需的 Spark 組態。 您可以通過使用 Spark 鍵與所有 S `--conf` park 配置鍵和值的逗號分隔列表來設置這些配置。您可以在筆記本或 AWS Glue Studio 控制台的 Job 參數部分中使用`%%configure`魔術。

```
%glue_version 3.0

%connections <name-of-the iceberg-connection>

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

使用以下屬性配置 Spark 會話：

- `<catalog_name>`是您的冰山星火工作階段目錄名稱。將其取代為目錄的名稱，並記得在與此型錄相關聯的所有模型組態中變更參考。然後，在您的代碼中，您應該使用完全限定的表名稱（包括 Spark 會話目錄名稱）引用您的 Iceberg 表，如下所示：`<catalog_name>.<database_name>.<table_name>`。
- `<catalog_name>.<warehouse>`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`<catalog_name>.catalog-impl`將設定為`org.apache.iceberg.aws.glue.GlueCatalog`。需要此金鑰才能指向任何自訂目錄實作的實作類別。有關 Iceberg 支援的目錄，請參閱本指南後???面的「一般最佳做法」一節。
- 用`org.apache.iceberg.aws.s3.S3FileIO`作以`<catalog_name>.io-impl`利用 Amazon S3 多部分上傳以實現高平行性。

例如，如果您有名為的目錄`glue_iceberg`，您可以使用多個`--conf`金鑰來設定工作，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.catalog-
impl=org.apache.iceberg.aws.glue.GlueCatalog ",
  "--conf" : " spark.sql.catalog.glue_iceberg.io-
impl=org.apache.iceberg.aws.s3.S3FileIO
}
```

或者，您可以使用代碼將上述配置添加到 Spark 腳本中，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.catalog-impl",
 "org.apache.iceberg.aws.glue.GlueCatalog") \
                    .config("spark.sql.catalog.glue_iceberg.io-impl",
 "org.apache.iceberg.aws.s3.S3FileIO") \
                    .getOrCreate()
```

AWS Glue 工作的最佳做法

本節提供調整 Spark 工作的一般準則， AWS Glue 以最佳化 Iceberg 資料表的讀取和寫入資料。有關特定於 Iceberg 的最佳實踐，請參閱本指南後面的最佳實踐部分。

- 使用最新版本的 AWS Glue 並儘可能升級 — 新版本 AWS Glue 提供效能改善、縮短啟動時間和新功能。它們還支持最新的冰山版本可能需要的較新 Spark 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 工作記憶體 — 請遵循中「最佳化記憶體管理」 AWS 部落格文章中的建議 AWS Glue。
- 使用 AWS Glue Auto Scaling 整 — 當您啟用「 AWS Glue 自動調整 Auto Scaling 例」時，會根據您的 AWS Glue 工作負載動態調整工作程式的數量。這有助於降低尖峰負載期間的工 AWS Glue 作成本，因為當工作量很小且工作者閒置時，可 AWS Glue 縮減工作人員的數量。若要使用「 AWS Glue Auto Scaling」，您可以指定工 AWS Glue 作可擴充至的最大 Worker 數目。如需詳細資訊，請參閱 AWS Glue 文件AWS Glue 中的〈使用 auto 調整比例〉。
- 使用自訂連接器或新增程式庫相依性-Iceberg 的 AWS Glue 原生整合最適合開始使用 Iceberg。但是，對於生產工作負載，我們建議您使用自訂容器或新增程式庫相依性 (如本指南前面所述)，以完全控制 Iceberg 版本。這種方法可幫助您從最新的 Iceberg 功能和 AWS Glue 工作的性能改進中受益。
- 啟用 Spark UI 以進行監視和除錯 — 您也可以在中使用 Spark UI AWS Glue 來檢查您的 Iceberg 工作，方法是在有向非循環圖 (DAG) 中將 Spark 工作的不同階段視覺化，並詳細監控工作。Spark UI 提供了一種有效的方法來排除故障和優化冰山工作。例如，您可以識別擁有大量洗牌或磁碟溢出的瓶頸階段，以識別調整機會。如需詳細資訊，請參閱 AWS Glue 文件中的使用 Apache Spark 網路使用者介面監視工作。

使用 Apache Spark 處理 Apache 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，請執行全域排序，然後寫入資料：

```
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
ORDER BY c_birth_country
""")
```

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，您可以使用本機排序來擷取資料：

```
input_data.sortWithinPartitions("c_birth_country") \
    .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，您可以使用全域排序來擷取資料：

```
input_data.orderBy("c_birth_country") \
    .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

上傳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 內更新資料表 {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 中的客戶記錄{UPSERT_TABLE_NAME}已存在{TABLE_NAME}於具有相同 的中c_customer_id，則{UPSERT_TABLE_NAME}記錄c_email_address值會覆寫現有值（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 快照來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。

下列範例會擷取快照 `start-snapshot-id`（獨佔） 和 `end-snapshot-id`（包含） 之間附加到 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。

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 Amazon Data Firehose 處理 Apache 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。

使用 Amazon Athena SQL 處理 Apache 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	✓	✓	✓	XCopy-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，請參閱本指南稍早在 Athena SQL 中開始使用 Iceberg 資料表一節。

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，請使用另一個引擎建立 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。

將現有資料表遷移至 Apache Iceberg

若要將目前的 Athena 或 AWS Glue 資料表（也稱為 Hive 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之上產生 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。

下列各節提供可用於遷移資料表APIs 概觀，以及選擇遷移策略的指引。如需這兩種策略的詳細資訊，請參閱 Iceberg 文件中的資料表遷移一節。

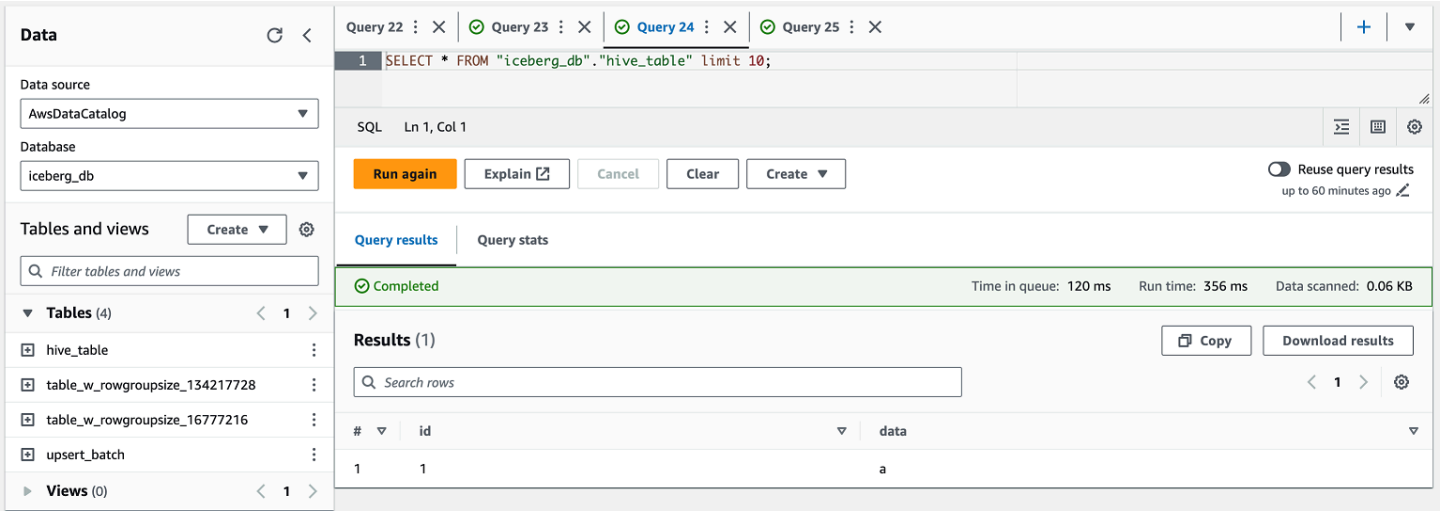
就地遷移

就地遷移不需要重寫所有資料檔案。相反地，會產生 Iceberg 中繼資料檔案，並將其連結到現有的資料檔案。Iceberg 提供三種實作就地遷移的選項：

- 使用 `snapshot` 程序，如 Iceberg 文件中的快照表和 Spark 程序：快照一節所述。
- 使用 `add_files` 程序，如 Iceberg 文件中的新增檔案和 Spark 程序：Add_files 一節所述。
- 使用 `migrate` 程序，如 Iceberg 文件中的遷移資料表和 Spark 程序：遷移一節所述。

目前，遷移程序無法直接與 搭配使用 AWS Glue Data Catalog，只能與 Hive 中繼存放區搭配使用。如果您需要使用 `migrate` 程序，而非 `snapshot`或 `add_files`，您可以搭配 Hive 中繼存放區 (HMS) 使用暫時 Amazon EMR 叢集。此方法需要 Iceberg 1.2 版或更新版本。

假設您想要建立下列 Hive 資料表：



您可以在 Athena 主控台中執行此程式碼來建立此 Hive 資料表：

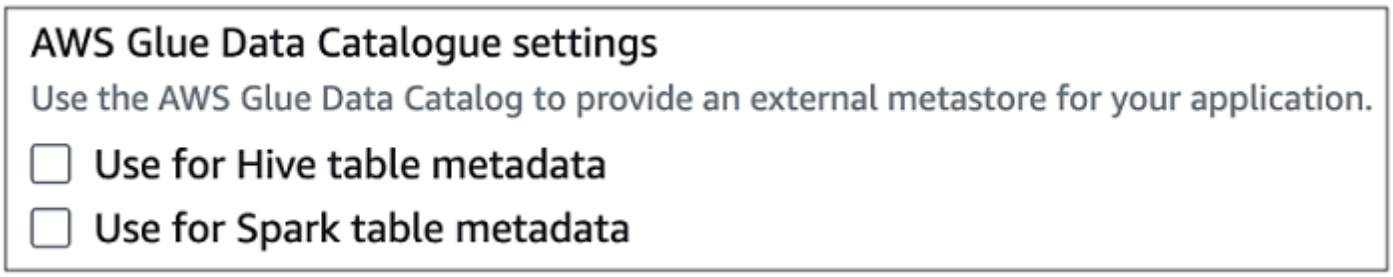
```
CREATE EXTERNAL TABLE 'hive_table'(
  'id' bigint,
  'data' string)
USING parquet
LOCATION
  's3://datalake-xxxx/aws_workshop/iceberg_db/hive_table'

INSERT INTO iceberg_db.hive_table VALUES (1, 'a')
```

如果您的 Hive 資料表已分割，請包含分割區陳述式，並根據 Hive 需求新增分割區。

```
ALTER TABLE default.placeholder_table_for_migration ADD
    PARTITION (date = '2023-10-10')
```

步驟：

1. 在未啟用 AWS Glue Data Catalog 整合的情況下建立 Amazon EMR 叢集，也就是說，請勿選取 Hive 或 Spark 資料表中繼資料的核取方塊。這是因為您將使用叢集中可用的原生 Hive 中繼存放區 (HMS) 進行此解決方法。



2. 設定 Spark 工作階段以使用 Iceberg Hive 目錄實作。

```
"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": "hive",
```

3. AWS Glue Data Catalog 透過執行 `show databases`或 來驗證您的 Amazon EMR 叢集未連線至 `show tables`。

```
[2]: %%sql
     show databases
```

Last executed at 2023-07-05 12:24:26 in 35.03s

Starting Spark application

ID	YARN Application ID	Kind	State	Spark UI	Driver log	User	Current session?
1	application_1686667730124_0002	pyspark	idle	Link	Link	None	✓

SparkSession available as 'spark'.

Type: Table Pie

namespace
default

4. 在 Amazon EMR 叢集的 Hive 中繼存放區中註冊 Hive 資料表，然後使用 Iceberg `migrate` 程序。

Register the Hive table in this local HMS catalog pointing to the S3 location where the files from the original Hive tables exist

```
%%sql -q
CREATE TABLE default.placeholder_hive_table (id bigint NOT NULL, data string)
USING parquet
LOCATION 's3://datalake-743490154766/aws_workshop/iceberg_db/hive_table/'
```

Last executed at 2023-07-05 12:55:19 in 3.25s

```
%%sql
select * from default.placeholder_hive_table limit 5
```

Last executed at 2023-07-05 12:57:13 in 7.43s

Type: Table Pie Scatter Line Area Bar

id	data
1	a

Once the Hive table is registered in this local HMS catalog, you can use Iceberg's migrate procedure.

```
spark.sql("CALL spark_catalog.system.migrate('default.placeholder_hive_table')")
```

Last executed at 2023-07-05 13:00:06 in 3.27s

▸ Spark Job Progress

```
DataFrame[migrated_files_count: bigint]
```

```
%%sql
show tables from default
```

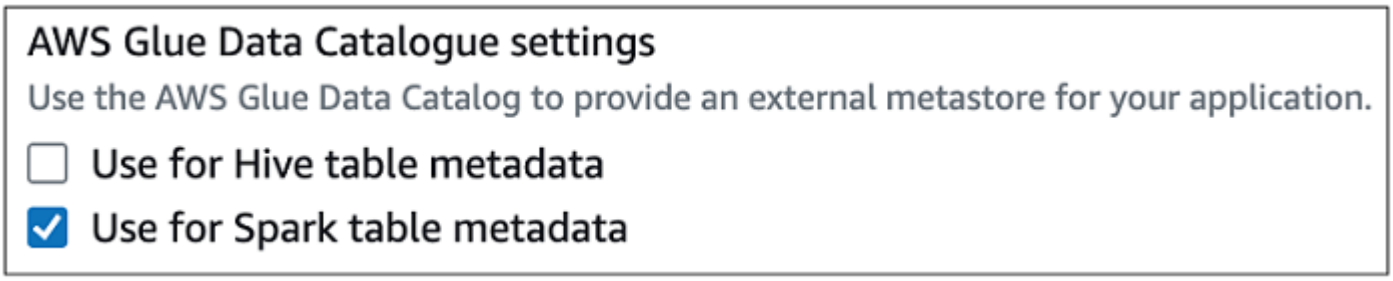
Last executed at 2023-07-05 13:00:49 in 7.42s

Type: Table Pie Scatter Line Area Bar

namespace	tableName	isTemporary
default	placeholder_hive_table	False
default	placeholder_hive_table_backup_	False

此程序會在與 Hive 資料表相同的位置建立 Iceberg 中繼資料檔案。

5. 在 中註冊遷移的 Iceberg 資料表 AWS Glue Data Catalog。
6. 切換回已啟用 AWS Glue Data Catalog 整合的 Amazon EMR 叢集。



7. 在 Spark 工作階段中使用下列 Iceberg 組態。

```
"spark.sql.extensions":"org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions",
        "spark.sql.catalog.glue_catalog": "org.apache.iceberg.spark.SparkCatalog",
        "spark.sql.catalog.glue_catalog.warehouse": "s3://datalake-xxxx/
aws_workshop",
        "spark.sql.catalog.glue_catalog.catalog-impl":
 "org.apache.iceberg.aws.glue.GlueCatalog",
        "spark.sql.catalog.glue_catalog.io-impl":
 "org.apache.iceberg.aws.s3.S3FileIO",
```

您現在可以從 Amazon EMR AWS Glue或 Athena 查詢此資料表。

```
%%sql
show tables from iceberg_db
```

Last executed at 2023-07-05 13:10:50 in 7.44s

Type: Table Pie Scatter Line Area Bar

namespace	tableName	isTemporary
iceberg_db	hive_table	False
iceberg_db	table_w_rowgroupsize_134217728	False
iceberg_db	table_w_rowgroupsize_16777216	False
iceberg_db	upsert_batch	False
iceberg_db	ws_webpage_pk_partitioned_140gb_trino	False

```
%%bash
aws s3 ls s3://datalake-743490154766/aws_workshop/iceberg_db/hive_table/metadata/
```

Last executed at 2023-07-05 13:10:20 in 488ms

```
2023-07-05 12:00:07       2239 00000-12a20051-6a3f-4b46-bae1-985f6df254db.metadata.json
2023-07-05 12:00:07       5802 b3d40480-0cb9-4cea-a4af-94c40a123689-m0.avro
2023-07-05 12:00:07       3781 snap-6104693268717769849-1-b3d40480-0cb9-4cea-a4af-94c40a123689.avro
```

```
metadata_file = "s3://datalake-743490154766/aws_workshop/iceberg_db/hive_table/metadata/00000-12a20051-6a3f-4b46-bae1-985f6df254db.metadata.json"
```

Last executed at 2023-07-05 13:11:46 in 49ms

```
spark.sql(f"CALL glue_catalog.system.register_table('iceberg_db.migrated_iceberg_table','{metadata_file}')")
```

Last executed at 2023-07-05 13:12:27 in 3.32s

Spark Job Progress

```
DataFrame[current_snapshot_id: bigint, total_records_count: bigint, total_data_files_count: bigint]
```

```
%%sql -q
alter table glue_catalog.iceberg_db.migrated_iceberg_table SET TBLPROPERTIES('format-version'='2')
```

Last executed at 2023-07-05 13:12:33 in 2.24s

```
%%sql
select * from glue_catalog.iceberg_db.migrated_iceberg_table limit 5
```

Last executed at 2023-07-05 13:12:44 in 7.42s

Type: Table Pie Scatter Line Area Bar

id	data
1	a

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，此選項有助於改善資料表品質：您可以驗證資料、進行結構描述和分割區變更、排序資料等。若要實作完整資料遷移，請使用下列其中一個選項：

- 在 Amazon EMR 或 AWS Glue Athena 上的 Spark 中使用 `CREATE TABLE ... AS SELECT`(CTAS) 陳述式。 您可以使用 和 `TBLPROPERTIES`子句來設定新 Iceberg `PARTITIONED BY`資料表的分割區規格和資料表屬性。您可以根據您的需求微調新資料表的結構描述和分割，而不只是從來源資料表繼承它們。

- 從來源資料表讀取，並使用 Amazon EMR 上的 Spark 或 AWS Glue（請參閱 Iceberg 文件中的建立資料表)，將資料寫入為新的 Iceberg 資料表。

選擇遷移策略

若要選擇最佳遷移策略，請考慮下表中的問題。

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。） • 如果您想要從資料檔案刪除整個資料欄，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。在此情況下，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。在此情況下，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）。

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
- 最佳化讀取效能
- 最佳化寫入效能
- 最佳化儲存體
- 使用壓縮來維護資料表
- 在 Amazon S3 中使用 Iceberg 工作負載

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 AWS Glue 中 Iceberg 的 Spark 組態一節。

```
"spark.sql.catalog.<your_catalog_name>.catalog-impl":
 "org.apache.iceberg.aws.glue.GlueCatalog"
```

- 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：

- 識別查詢中經常使用的資料欄。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 的隱藏分割功能搭配儲存貯體轉換，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，以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一開始不清楚，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

最佳化查詢效能涉及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刪除大小。

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 會請求 Spark 在寫入儲存體之前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（從幾 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 或更多，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，但不保證檔案會具有該大小。為了確保適當的檔案大小，請定期執行壓縮，將小型檔案合併為較大的檔案。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 會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僅使用幾個資料欄，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：當資料對這些資料欄進行排序時，資料欄統計資料最有效。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在相同資料表分割區中的使用案例（例如，用於每日批次載入）。

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，其提供較慢的讀取/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 資料表而言是如此。

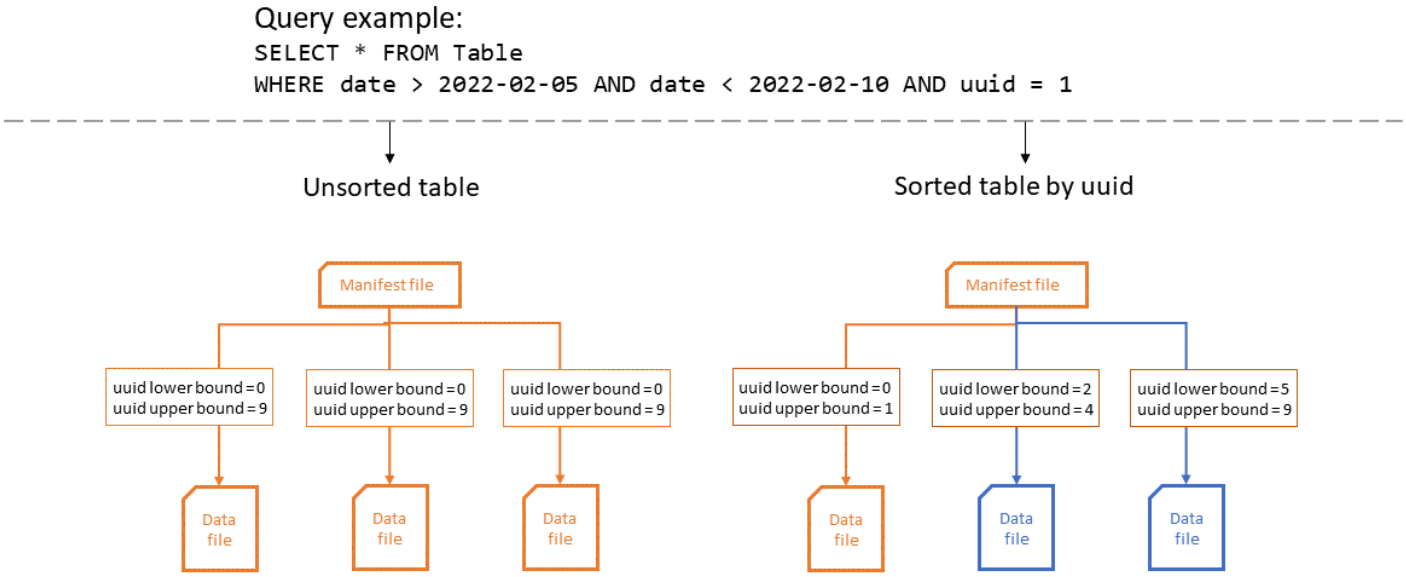
設定排序順序

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，可大幅提高檔案剔除的效率，進而加快讀取操作。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。

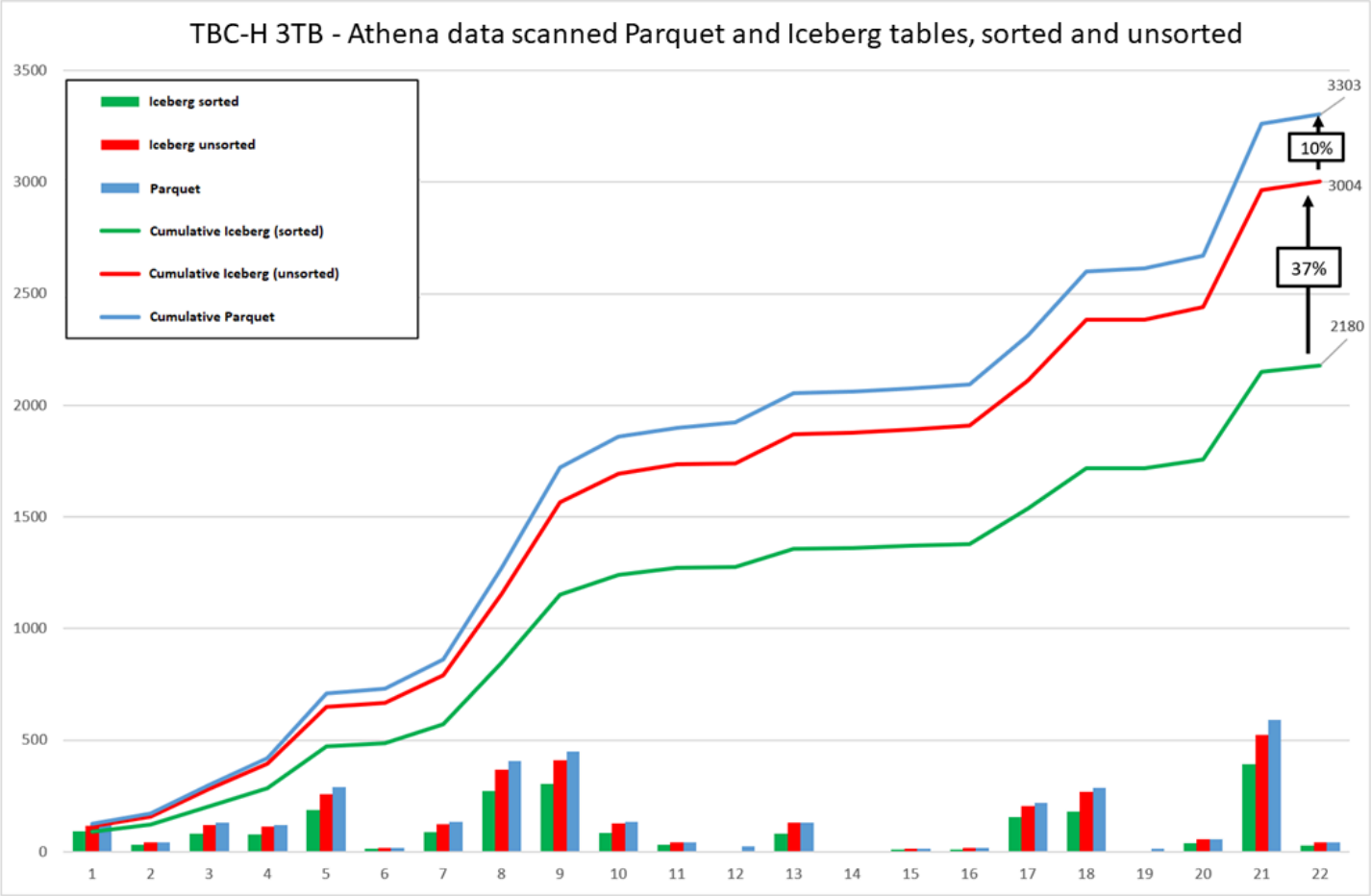
例如，在大部分查詢依 篩選的日期 (`yyyy-mm-dd`) 分割的資料表中`uuid`，您可以使用 DDL 選項來`Write Distributed By Partition Locally Ordered`確保 Spark 寫入具有非重疊範圍的檔案。

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。

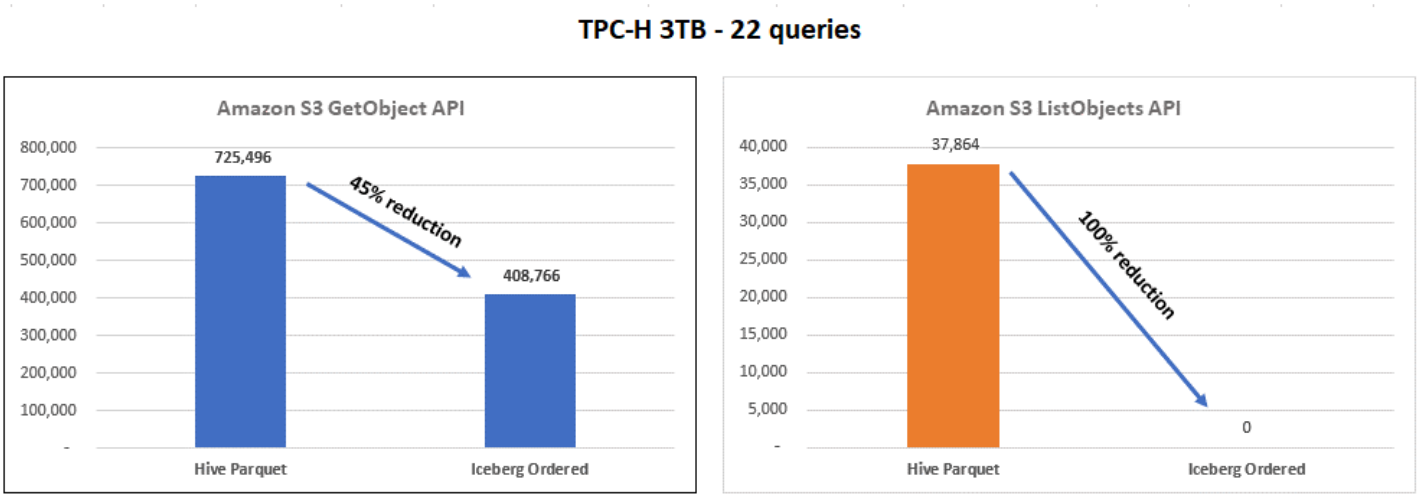


變更排序順序不會影響現有的資料檔案。您可以使用 Iceberg 壓縮來套用排序順序。

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 任務之間的分佈方式。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，以獲得查詢效能。

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。這表示讀取和寫入效能之間的典型取捨。

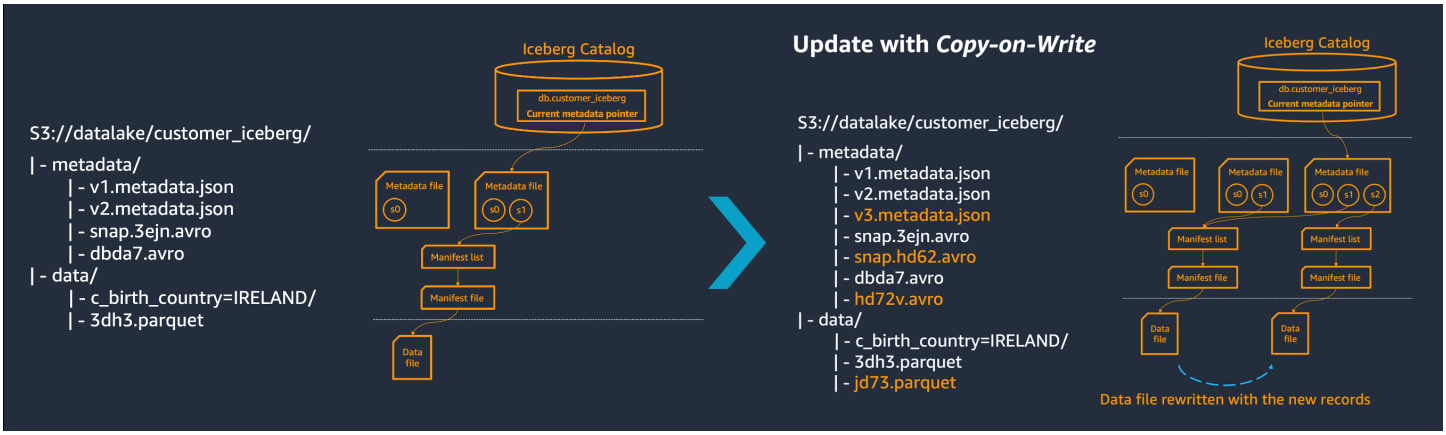
如果寫入速度對您的使用案例很重要，例如在串流工作負載中，請考慮在寫入器的選項`Avro`中`write-format`將 設定為 ，以 Avro 格式寫入。由於 Avro 是以資料列為基礎的格式，因此能以較慢的讀取效能提供更快的寫入時間。

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 將所有資料表基礎的檔案視為不可變。



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。您無法復原或時間歷程至過期的快照。
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單一快照。

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 移除快照。 https://iceberg.apache.org/docs/latest/spark-procedures/#expire_snapshots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。 • 使用 Iceberg 中的 register_table 程序，將快照註冊為目錄中的資料表。	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，您無法安排行程或轉返至封存快照的時間。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
	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叢集的方式，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。

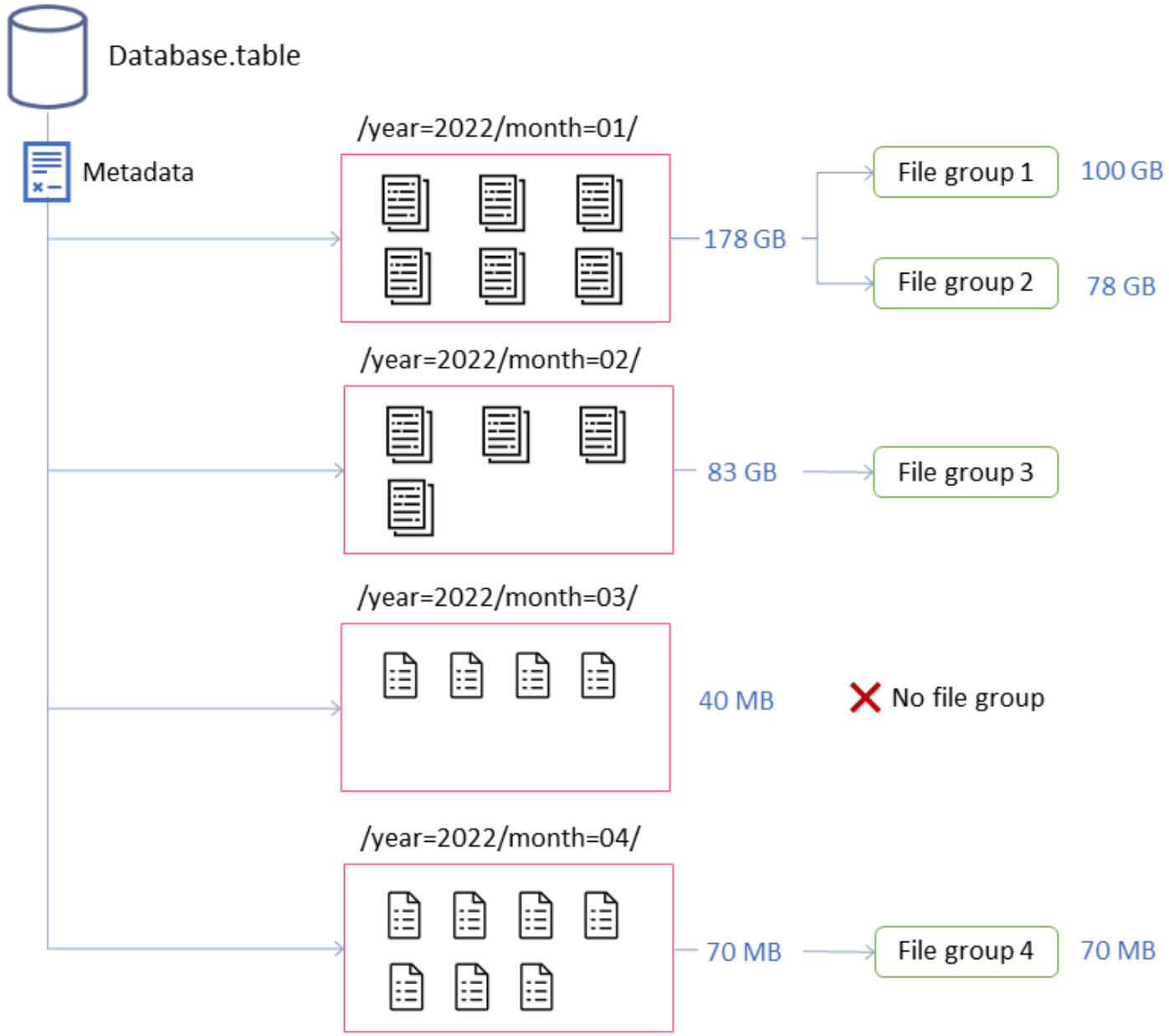
Iceberg 壓縮

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，只有在有三個或更多個參考該檔案的刪除檔案時，才會重寫資料檔案。

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，因為它低於 100 GB。`month=03` 分割區不符合五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 EC2 或 Amazon EKS 上的 Amazon EMR 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，透過在 `OPTIMIZE` 陳述式中傳遞資料表屬性，或使用 陳述式在建立資料表之後，變更 `CREATE TABLE` `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，則不會產生與這些操作相關的成本。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。
在可能因延遲抵達資料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。如果發生碰撞（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 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 個錯誤，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，並將負載分配到多個隨機的 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。

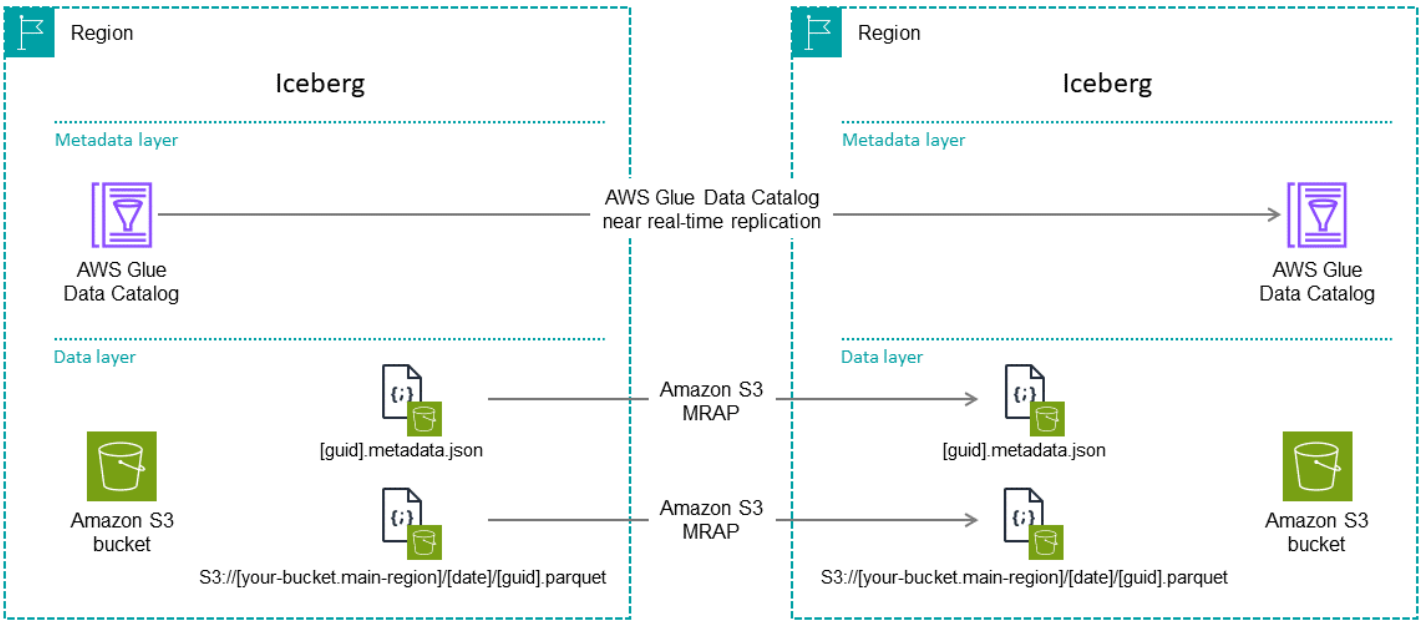
透過 Boto3、Amazon S3 SDK 或 AWS Command Line Interface (AWS CLI) 使用 Amazon S3 APIs，或使用任何其他非 Iceberg 方法來覆寫或移除 Iceberg 資料表的 Amazon S3 檔案，會導致資料表損毀和查詢失敗。

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 AWS 區域複寫資料。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，但您可以使用 MRAP 透過將儲存貯體映射至存取點來執行 Amazon S3 操作。MRAP 也與 Amazon S3 跨區域複寫程序無縫整合，其引入高達 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：
   - 使用複寫設定用於 AWS Glue Data Catalog 複寫 Iceberg 資料表中繼資料的管道。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

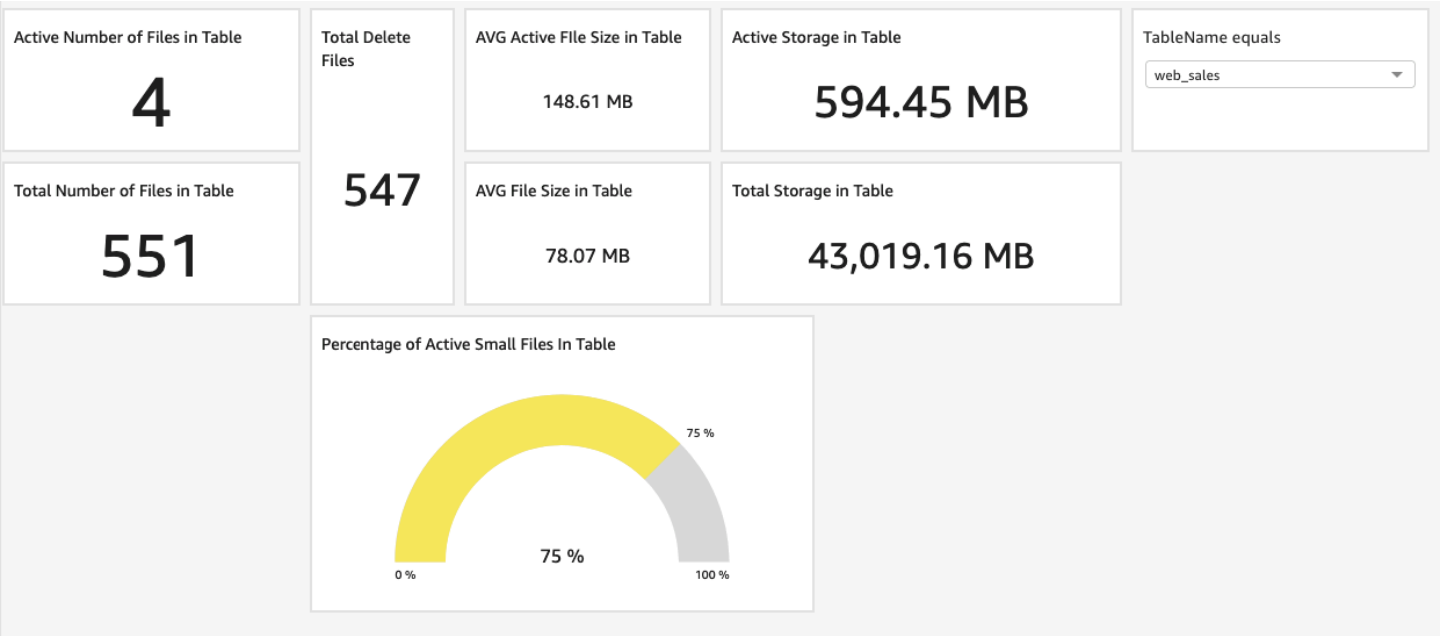
若要監控 Iceberg 工作負載，您有兩個選項：分析中繼資料表或使用指標報告程式。指標報告程式在 Iceberg 1.2 版中推出，僅適用於 REST 和 JDBC 目錄。

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 QuickSight 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 資料表中以用於操作。最後，使用 Amazon Athena 建立 QuickSight 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。



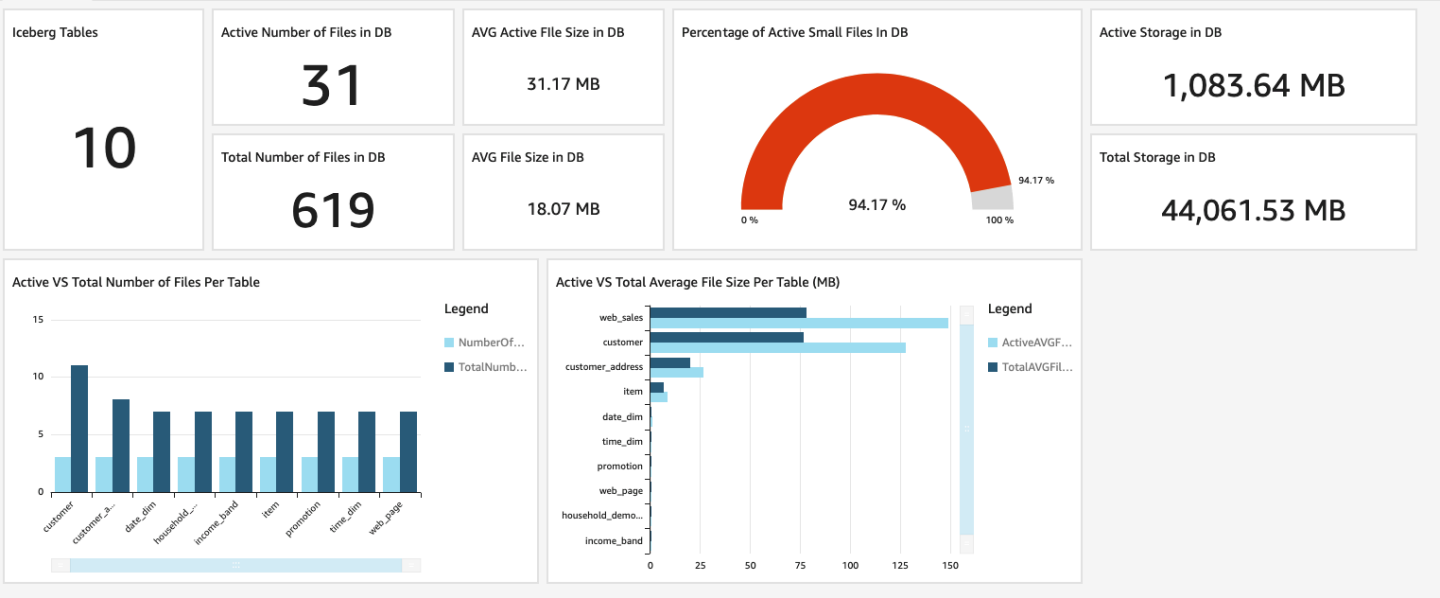
QuickSight 儀表板範例會收集 Iceberg 資料表的下列關鍵效能指標 KPIs)：

KPI	Description	查詢
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	查詢
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`

資料庫層級監控

下列範例顯示在 QuickSight 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	查詢
檔案數量	Iceberg 資料庫中的檔案數目（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。
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）	

KPI	Description	查詢
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處理資料表維護。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

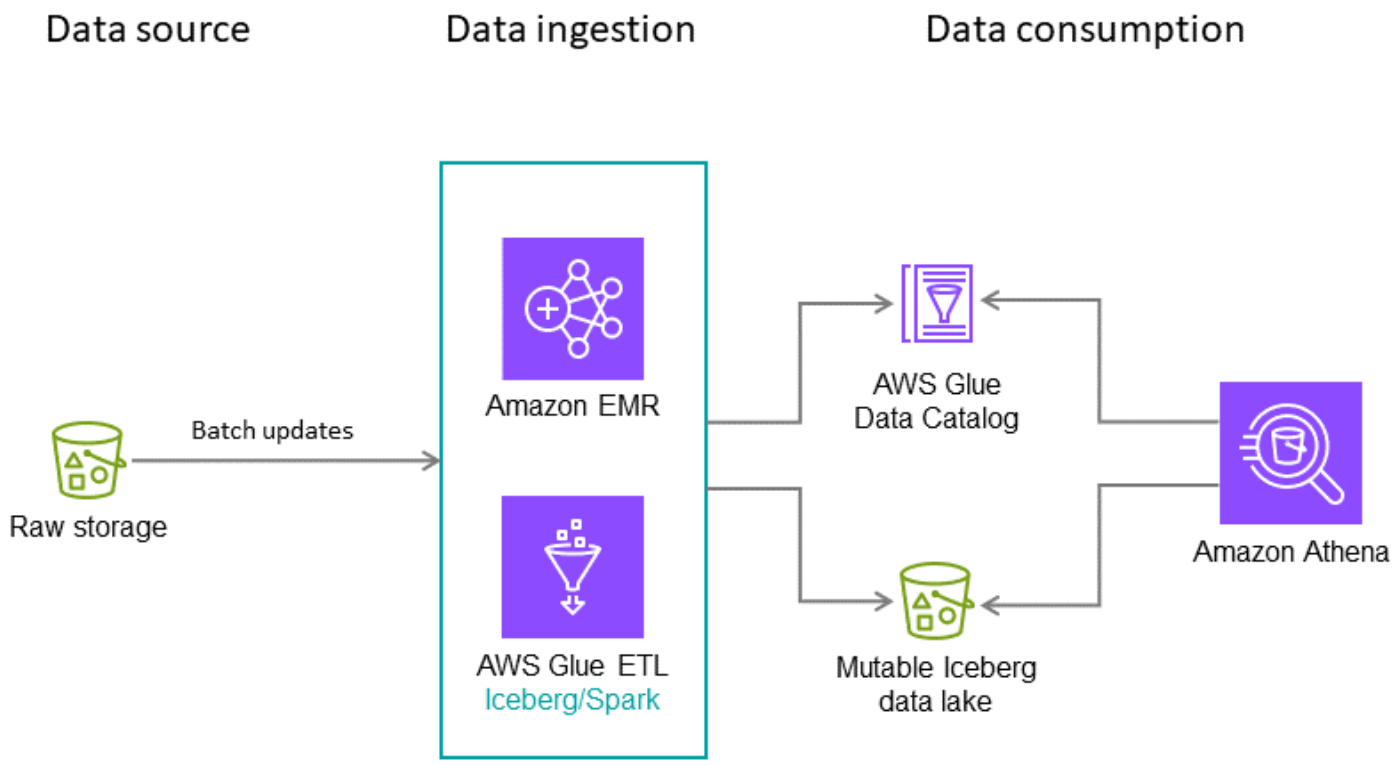
阿帕奇冰山上的參考架構 AWS

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

每晚批次擷取

對於這個假設的用例，假設您的 Iceberg 表每晚會導入信用卡交易。每個批次只包含增量更新，必須將這些更新合併到目標資料表中。每年接收數次完整的歷史資料。對於這種情況，我們建議使用以下架構和配置。

注意：這只是一個例子。最佳配置取決於您的數據和需求。



建議：

- 文件大小：128 MB，因為阿帕奇星火任務處理 128 MB 塊的數據。
- 寫入類型： copy-on-write。如本指南前面所詳述，此方法有助於確保資料以讀取最佳化的方式撰寫。

- 分區變量：年/月/日。在我們假設的使用案例中，我們最頻繁地查詢最近的資料，雖然我們偶爾會針對過去兩年的資料執行完整的資料表掃描。磁碟分割的目標是根據使用案例的需求來驅動快速讀取作業。
- 排序順序：時間戳
- 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。如需架構圖和詳細資訊，請參閱 AWS 部落格文章使用 Apache Iceberg 建立交易資料湖 AWS Glue，以及使用 AWS Lake Formation 和 Amazon Athena 的跨帳戶資料共用。

資源

- 在（文件）中使用 Iceberg 架構 AWS GlueAWS Glue
- Iceberg (Amazon EMR 文件)
- 使用 Apache Iceberg 資料表 (Amazon Athena 文件)
- Amazon S3 文件
- QuickSight 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

以下人員在 AWS 撰寫、共同撰寫和審閱本指南。

貢獻者

- 卡洛斯·羅德里格斯, 解決方案架構師, 大
- Imtiaz（Taz）Sayd，解決方案架構師技術領導者，分析
- 夏娜施珀斯，解決方案架構師，大數據
- 普拉珊特·辛格，軟件開發工程師，Amazon EMR
- 斯蒂法諾·桑多納，解決方案架構師，大數據
- 阿倫 A K，解決方案架構師，大數據和 ETL
- 弗朗西斯科·莫里略，解決方案架構師
- 蘇森·菲利普斯，分析架構師，Amazon EMR
- 卡拉 OK 系列, 解決方案架構師
- 與那坦·多蘭, 分析專家
- 蓋·巴查爾，解決方案架構師
- 索菲亞·齊伯曼, 解決方案架構師, 流
- 伊斯梅爾·馬赫魯夫，解決方案架構師，分析
- 丹樓梯, 專業解決方案建築師
- 薩克蒂米什拉, 解決方案架構師

審稿

- 里克·西爾斯, 總經理, Amazon EMR
- 琳達·奧康納, Amazon EMR 專家
- 伊恩·邁耶斯, 主任, Amazon EMR
- 維妮塔·安南斯，Amazon EMR 產品管理總監
- 傑森·伯科維茨，產品經理， AWS Lake Formation
- 馬赫什米什拉, 產品經理, Amazon Redshift
- 弗拉基米爾·茲拉特金, 經理, 解決方案架構, 大數據
- 卡蒂克·普拉巴卡爾，分析建築師，Amazon EMR

- Amazon EMR 軟體開發工程師葉傑
- 維傑 · 耆那教, 產品經理
- 評論, Amazon S3 產品經理
- 莫莉·布朗, 總經理, AWS Lake Formation
- 設計師, 解決方案架構師, 數據
- 陳格文 , 產品行銷經理

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

AWS 規範性指引詞彙表

以下是 AWS Prescriptive Guidance 提供的策略、指南和模式中常用的術語。若要建議項目，請使用詞彙表末尾的提供意見回饋連結。

數字

7 R

將應用程式移至雲端的七種常見遷移策略。這些策略以 Gartner 在 2011 年確定的 5 R 為基礎，包括以下內容：

- 重構/重新架構 – 充分利用雲端原生功能來移動應用程式並修改其架構，以提高敏捷性、效能和可擴展性。這通常涉及移植作業系統和資料庫。範例：將您的內部部署 Oracle 資料庫遷移至 Amazon Aurora PostgreSQL 相容版本。
- 平台轉換 (隨即重塑) – 將應用程式移至雲端，並引入一定程度的優化以利用雲端功能。範例：將您的現場部署 Oracle 資料庫遷移至 中的 Amazon Relational Database Service (Amazon RDS) for Oracle AWS 雲端。
- 重新購買 (捨棄再購買) – 切換至不同的產品，通常從傳統授權移至 SaaS 模型。範例：將您的客戶關係管理 (CRM) 系統遷移至 Salesforce.com。
- 主機轉換 (隨即轉移) – 將應用程式移至雲端，而不進行任何變更以利用雲端功能。範例：將您的現場部署 Oracle 資料庫遷移至 中 EC2 執行個體上的 Oracle AWS 雲端。
- 重新放置 (虛擬機器監視器等級隨即轉移) – 將基礎設施移至雲端，無需購買新硬體、重寫應用程式或修改現有操作。您可以將伺服器從內部部署平台遷移到相同平台的雲端服務。範例：將 Microsoft Hyper-V應用程式遷移至 AWS。
- 保留 (重新檢視) – 將應用程式保留在來源環境中。其中可能包括需要重要重構的應用程式，且您希望將該工作延遲到以後，以及您想要保留的舊版應用程式，因為沒有業務理由來進行遷移。
- 淘汰 – 解除委任或移除來源環境中不再需要的應用程式。

A

ABAC

請參閱屬性型存取控制。

抽象服務

請參閱 受管服務。

ACID

請參閱原子性、一致性、隔離性、耐久性。

主動-主動式遷移

一種資料庫遷移方法，其中來源和目標資料庫保持同步 (透過使用雙向複寫工具或雙重寫入操作)，且兩個資料庫都在遷移期間處理來自連接應用程式的交易。此方法支援小型、受控制批次的遷移，而不需要一次性切換。它更靈活，但需要比主動-被動遷移更多的工作。

主動-被動式遷移

一種資料庫遷移方法，其中來源和目標資料庫保持同步，但只有來源資料庫處理來自連接應用程式的交易，同時將資料複寫至目標資料庫。目標資料庫在遷移期間不接受任何交易。

彙總函數

在一組資料列上運作的 SQL 函數，會計算群組的單一傳回值。彙總函數的範例包括 SUM和 MAX。

AI

請參閱人工智慧。

AIOps

請參閱人工智慧操作。

匿名化

永久刪除資料集中個人資訊的程序。匿名化有助於保護個人隱私權。匿名資料不再被視為個人資料。

反模式

經常用於經常性問題的解決方案，其中解決方案具有反生產力、無效或比替代解決方案更有效。

應用程式控制

一種安全方法，僅允許使用核准的應用程式，以協助保護系統免受惡意軟體攻擊。

應用程式組合

有關組織使用的每個應用程式的詳細資訊的集合，包括建置和維護應用程式的成本及其商業價值。此資訊是產品組合探索和分析程序的關鍵，有助於識別要遷移、現代化和優化的應用程式並排定其優先順序。

人工智慧 (AI)

電腦科學領域，致力於使用運算技術來執行通常與人類相關的認知功能，例如學習、解決問題和識別模式。如需詳細資訊，請參閱什麼是人工智慧？

人工智慧操作 (AIOps)

使用機器學習技術解決操作問題、減少操作事件和人工干預以及提高服務品質的程序。如需有關如何在 AWS 遷移策略中使用 AIOps 的詳細資訊，請參閱操作整合指南。

非對稱加密

一種加密演算法，它使用一對金鑰：一個用於加密的公有金鑰和一個用於解密的私有金鑰。您可以共用公有金鑰，因為它不用於解密，但對私有金鑰存取應受到高度限制。

原子性、一致性、隔離性、耐久性 (ACID)

一組軟體屬性，即使在出現錯誤、電源故障或其他問題的情況下，也能確保資料庫的資料有效性和操作可靠性。

屬性型存取控制 (ABAC)

根據使用者屬性 (例如部門、工作職責和團隊名稱) 建立精細許可的實務。如需詳細資訊，請參閱《 AWS Identity and Access Management (IAM) 文件》中的 ABAC for AWS。

授權資料來源

存放主要版本資料的位置，被視為最可靠的資訊來源。您可以將授權資料來源中的資料複製到其他位置，以處理或修改資料，例如匿名、修訂或假名化資料。

可用區域

中的不同位置 AWS 區域 ，可隔離其他可用區域中的故障，並提供相同區域中其他可用區域的低成本、低延遲網路連線。

AWS 雲端採用架構 (AWS CAF)

的指導方針和最佳實務架構 AWS ，可協助組織制定高效且有效的計劃，以成功地移至雲端。 AWS CAF 將指導方針組織到六個重點領域：業務、人員、治理、平台、安全和營運。業務、人員和控管層面著重於業務技能和程序；平台、安全和操作層面著重於技術技能和程序。例如，人員層面針對處理人力資源 (HR)、人員配備功能和人員管理的利害關係人。因此， AWS CAF 為人員開發、訓練和通訊提供指引，協助組織做好成功採用雲端的準備。如需詳細資訊，請參閱 AWS CAF 網站和 AWS CAF 白皮書。

AWS 工作負載資格架構 (AWS WQF)

一種工具，可評估資料庫遷移工作負載、建議遷移策略，並提供工作預估值。 AWS WQF 隨附於 AWS Schema Conversion Tool (AWS SCT)。它會分析資料庫結構描述和程式碼物件、應用程式程式碼、相依性和效能特性，並提供評估報告。

B

錯誤的機器人

旨在中斷或傷害個人或組織的機器人。

BCP

請參閱業務持續性規劃。

行為圖

資源行為的統一互動式檢視，以及一段時間後的互動。您可以將行為圖與 Amazon Detective 搭配使用來檢查失敗的登入嘗試、可疑的 API 呼叫和類似動作。如需詳細資訊，請參閱偵測文件中的行為圖中的資料。

大端序系統

首先儲存最高有效位元組的系統。另請參閱 Endianness。

二進制分類

預測二進制結果的過程 (兩個可能的類別之一)。例如，ML 模型可能需要預測諸如「此電子郵件是否是垃圾郵件？」等問題 或「產品是書還是汽車？」

Bloom 篩選條件

一種機率性、記憶體高效的資料結構，用於測試元素是否為集的成員。

藍/綠部署

一種部署策略，您可以在其中建立兩個不同但相同的環境。您可以在一個環境 （藍色） 中執行目前的應用程式版本，並在另一個環境 （綠色） 中執行新的應用程式版本。此策略可協助您快速復原，並將影響降至最低。

機器人

透過網際網路執行自動化任務並模擬人類活動或互動的軟體應用程式。有些機器人有用或有益，例如在網際網路上為資訊編製索引的 Web 爬蟲程式。某些其他機器人稱為惡意機器人，旨在中斷或傷害個人或組織。

殭屍網路

受到惡意軟體感染且受單一方控制之機器人的網路，稱為機器人繼承器或機器人運算子。殭屍網路是擴展機器人及其影響的最佳已知機制。

分支

程式碼儲存庫包含的區域。儲存庫中建立的第一個分支是主要分支。您可以從現有分支建立新分支，然後在新分支中開發功能或修正錯誤。您建立用來建立功能的分支通常稱為功能分支。當準備好發佈功能時，可以將功能分支合併回主要分支。如需詳細資訊，請參閱關於分支 (GitHub 文件)。

碎片存取

在特殊情況下，以及透過核准的程序，讓使用者快速取得他們通常無權存取 AWS 帳戶 之 的存取權。如需詳細資訊，請參閱 Well-Architected 指南中的 AWS 實作打破玻璃程序指標。

棕地策略

環境中的現有基礎設施。對系統架構採用棕地策略時，可以根據目前系統和基礎設施的限制來設計架構。如果正在擴展現有基礎設施，則可能會混合棕地和綠地策略。

緩衝快取

儲存最常存取資料的記憶體區域。

業務能力

業務如何創造價值 (例如，銷售、客戶服務或營銷)。業務能力可驅動微服務架構和開發決策。如需詳細資訊，請參閱在 AWS上執行容器化微服務白皮書的圍繞業務能力進行組織部分。

業務連續性規劃 (BCP)

一種解決破壞性事件 (如大規模遷移) 對營運的潛在影響並使業務能夠快速恢復營運的計畫。

C

CAF

請參閱AWS 雲端採用架構。

Canary 部署

版本對最終使用者的緩慢和增量版本。當您有信心時，您可以部署新版本，並完全取代目前的版本。

CCoE

請參閱 Cloud Center of Excellence。

CDC

請參閱變更資料擷取。

變更資料擷取 (CDC)

追蹤對資料來源 (例如資料庫表格) 的變更並記錄有關變更的中繼資料的程序。您可以將 CDC 用於各種用途，例如稽核或複寫目標系統中的變更以保持同步。

混沌工程

故意引入故障或破壞性事件，以測試系統的彈性。您可以使用 AWS Fault Injection Service (AWS FIS) 執行實驗，為您的 AWS 工作負載帶來壓力，並評估其回應。

CI/CD

請參閱持續整合和持續交付。

分類

有助於產生預測的分類程序。用於分類問題的 ML 模型可預測離散值。離散值永遠彼此不同。例如，模型可能需要評估影像中是否有汽車。

用戶端加密

在目標 AWS 服務 接收資料之前，在本機加密資料。

雲端卓越中心 (CCoE)

一個多學科團隊，可推動整個組織的雲端採用工作，包括開發雲端最佳實務、調動資源、制定遷移時間表以及領導組織進行大規模轉型。如需詳細資訊，請參閱 AWS 雲端 企業策略部落格上的 CCoE 文章。

雲端運算

通常用於遠端資料儲存和 IoT 裝置管理的雲端技術。雲端運算通常連接到邊緣運算技術。

雲端操作模型

在 IT 組織中，用於建置、成熟和最佳化一或多個雲端環境的操作模型。如需詳細資訊，請參閱建置您的雲端操作模型。

採用雲端階段

組織在遷移至 時通常會經歷的四個階段 AWS 雲端：

- 專案 – 執行一些與雲端相關的專案以進行概念驗證和學習用途
- 基礎 – 進行基礎投資以擴展雲端採用 (例如，建立登陸區域、定義 CCoE、建立營運模型)
- 遷移 – 遷移個別應用程式
- 重塑 – 優化產品和服務，並在雲端中創新

這些階段由 Stephen Orban 在部落格文章 The Journey Toward Cloud-First 和 Enterprise Strategy 部落格上的採用階段中定義。 AWS 雲端 如需有關它們如何與 AWS 遷移策略相關的詳細資訊，請參閱遷移整備指南。

CMDB

請參閱組態管理資料庫。

程式碼儲存庫

透過版本控制程序來儲存及更新原始程式碼和其他資產 (例如文件、範例和指令碼) 的位置。常見的雲端儲存庫包括 GitHub或 Bitbucket Cloud。程式碼的每個版本都稱為分支。在微服務結構中，每個儲存庫都專用於單個功能。單一 CI/CD 管道可以使用多個儲存庫。

冷快取

一種緩衝快取，它是空的、未填充的，或者包含過時或不相關的資料。這會影響效能，因為資料庫執行個體必須從主記憶體或磁碟讀取，這比從緩衝快取讀取更慢。

冷資料

很少存取且通常是歷史資料的資料。查詢這類資料時，通常可接受慢查詢。將此資料移至效能較低且成本較低的儲存層或類別，可以降低成本。

電腦視覺 (CV)

AI 欄位???，使用機器學習從數位影像和影片等視覺化格式分析和擷取資訊。例如，Amazon SageMaker AI 提供 CV 的影像處理演算法。

組態偏離

對於工作負載，組態會從預期狀態變更。這可能會導致工作負載不合規，而且通常是漸進和無意的。

組態管理資料庫 (CMDB)

儲存和管理有關資料庫及其 IT 環境的資訊的儲存庫，同時包括硬體和軟體元件及其組態。您通常在遷移的產品組合探索和分析階段使用 CMDB 中的資料。

一致性套件

您可以組合的 AWS Config 規則和修補動作集合，以自訂您的合規和安全檢查。您可以使用 YAML 範本，將一致性套件部署為 AWS 帳戶 和 區域中或整個組織的單一實體。如需詳細資訊，請參閱 AWS Config 文件中的一致性套件。

持續整合和持續交付 (CI/CD)

自動化軟體發行程序的來源、建置、測試、暫存和生產階段的程序。CI/CD 通常被描述為管道。CI/CD 可協助您將程序自動化、提升生產力、改善程式碼品質以及加快交付速度。如需詳細資訊，請參閱持續交付的優點。CD 也可表示持續部署。如需詳細資訊，請參閱持續交付與持續部署。

CV

請參閱電腦視覺。

D

靜態資料

網路中靜止的資料，例如儲存中的資料。

資料分類

根據重要性和敏感性來識別和分類網路資料的程序。它是所有網路安全風險管理策略的關鍵組成部分，因為它可以協助您確定適當的資料保護和保留控制。資料分類是 AWS Well-Architected Framework 中安全支柱的元件。如需詳細資訊，請參閱資料分類。

資料偏離

生產資料與用於訓練 ML 模型的資料之間有意義的變化，或輸入資料隨時間有意義的變更。資料偏離可以降低 ML 模型預測的整體品質、準確性和公平性。

傳輸中的資料

在您的網路中主動移動的資料，例如在網路資源之間移動。

資料網格

架構架構，提供分散式、分散式資料擁有權與集中式管理。

資料最小化

僅收集和處理嚴格必要資料的原則。在 中實作資料最小化 AWS 雲端 可以降低隱私權風險、成本和分析碳足跡。

資料周邊

AWS 環境中的一組預防性防護機制，可協助確保只有信任的身分才能從預期的網路存取信任的資源。如需詳細資訊，請參閱在 上建置資料周邊 AWS。

資料預先處理

將原始資料轉換成 ML 模型可輕鬆剖析的格式。預處理資料可能意味著移除某些欄或列，並解決遺失、不一致或重複的值。

資料來源

在整個生命週期中追蹤資料的原始伺服器和歷史記錄的程序，例如資料的產生、傳輸和儲存方式。

資料主體

正在收集和處理其資料的個人。

資料倉儲

支援商業智慧的資料管理系統，例如 分析。資料倉儲通常包含大量歷史資料，通常用於查詢和分析。

資料庫定義語言 (DDL)

用於建立或修改資料庫中資料表和物件之結構的陳述式或命令。

資料庫處理語言 (DML)

用於修改 (插入、更新和刪除) 資料庫中資訊的陳述式或命令。

DDL

請參閱資料庫定義語言。

深度整體

結合多個深度學習模型進行預測。可以使用深度整體來獲得更準確的預測或估計預測中的不確定性。

深度學習

一個機器學習子領域，它使用多層人工神經網路來識別感興趣的輸入資料與目標變數之間的對應關係。

深度防禦

這是一種資訊安全方法，其中一系列的安全機制和控制項會在整個電腦網路中精心分層，以保護網路和其中資料的機密性、完整性和可用性。當您在 上採用此策略時 AWS，您可以在 AWS

Organizations 結構的不同層新增多個控制項，以協助保護資源。例如，defense-in-depth方法可能會結合多重要素驗證、網路分割和加密。

委派的管理員

在 中 AWS Organizations，相容的服務可以註冊 AWS 成員帳戶來管理組織的帳戶，並管理該服務的許可。此帳戶稱為該服務的委派管理員。如需詳細資訊和相容服務清單，請參閱 AWS Organizations 文件中的可搭配 AWS Organizations運作的服務。

部署

在目標環境中提供應用程式、新功能或程式碼修正的程序。部署涉及在程式碼庫中實作變更，然後在應用程式環境中建置和執行該程式碼庫。

開發環境

請參閱 環境。

偵測性控制

一種安全控制，用於在事件發生後偵測、記錄和提醒。這些控制是第二道防線，提醒您注意繞過現有預防性控制的安全事件。如需詳細資訊，請參閱在 AWS上實作安全控制中的偵測性控制。

開發值串流映射 (DVSM)

一種程序，用於識別對軟體開發生命週期中的速度和品質造成負面影響的限制並排定優先順序。DVSM 擴展了最初專為精簡製造實務設計的價值串流映射程序。它著重於透過軟體開發程序建立和移動價值所需的步驟和團隊。

數位分身

真實世界系統的虛擬呈現，例如建築物、工廠、工業設備或生產線。數位分身支援預測性維護、遠端監控和生產最佳化。

維度資料表

在星星結構描述中，較小的資料表包含有關事實資料表中量化資料的資料屬性。維度資料表屬性通常是文字欄位或離散數字，其行為類似於文字。這些屬性通常用於查詢限制、篩選和結果集標記。

災難

防止工作負載或系統在其主要部署位置實現其業務目標的事件。這些事件可能是自然災難、技術故障或人為動作的結果，例如意外設定錯誤或惡意軟體攻擊。

災難復原 (DR)

您用來將災難造成的停機時間和資料遺失降至最低的策略和程序。如需詳細資訊，請參閱 AWS Well-Architected Framework 中的 上工作負載災難復原 AWS：雲端中的復原。

DML

請參閱資料庫處理語言。

領域驅動的設計

一種開發複雜軟體系統的方法，它會將其元件與每個元件所服務的不斷發展的領域或核心業務目標相關聯。Eric Evans 在其著作 Domain-Driven Design: Tackling Complexity in the Heart of Software (Boston: Addison-Wesley Professional, 2003) 中介紹了這一概念。如需有關如何將領域驅動的設計與 strangler fig 模式搭配使用的資訊，請參閱使用容器和 Amazon API Gateway 逐步現代化舊版 Microsoft ASP.NET (ASMX) Web 服務。

DR

請參閱災難復原。

偏離偵測

追蹤與基準組態的偏差。例如，您可以使用 AWS CloudFormation 來偵測系統資源中的偏離，也可以使用 AWS Control Tower 來偵測登陸區域中可能影響控管要求合規性的變更。 https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/using-cfn-stack-drift.html

DVSM

請參閱開發值串流映射。

E

EDA

請參閱探索性資料分析。

EDI

請參閱電子資料交換。

邊緣運算

提升 IoT 網路邊緣智慧型裝置運算能力的技術。與雲端運算相比，邊緣運算可以減少通訊延遲並改善回應時間。

電子資料交換 (EDI)

組織之間商業文件的自動交換。如需詳細資訊，請參閱什麼是電子資料交換。

加密

一種運算程序，可將人類可讀取的純文字資料轉換為加密文字。

加密金鑰

由加密演算法產生的隨機位元的加密字串。金鑰長度可能有所不同，每個金鑰的設計都是不可預測且唯一的。

端序

位元組在電腦記憶體中的儲存順序。大端序系統首先儲存最高有效位元組。小端序系統首先儲存最低有效位元組。

端點

請參閱 服務端點。

端點服務

您可以在虛擬私有雲端 (VPC) 中託管以與其他使用者共用的服務。您可以使用 建立端點服務，AWS PrivateLink 並將許可授予其他 AWS 帳戶 或 AWS Identity and Access Management (IAM) 委託人。這些帳戶或主體可以透過建立介面 VPC 端點私下連接至您的端點服務。如需詳細資訊，請參閱 Amazon Virtual Private Cloud (Amazon VPC) 文件中的建立端點服務。

企業資源規劃 (ERP)

一種系統，可自動化和管理企業的關鍵業務流程（例如會計、MES 和專案管理）。

信封加密

使用另一個加密金鑰對某個加密金鑰進行加密的程序。如需詳細資訊，請參閱 AWS Key Management Service (AWS KMS) 文件中的信封加密。

環境

執行中應用程式的執行個體。以下是雲端運算中常見的環境類型：

- 開發環境 – 執行中應用程式的執行個體，只有負責維護應用程式的核心團隊才能使用。開發環境用來測試變更，然後再將開發環境提升到較高的環境。此類型的環境有時稱為測試環境。
- 較低的環境 – 應用程式的所有開發環境，例如用於初始建置和測試的開發環境。
- 生產環境 – 最終使用者可以存取的執行中應用程式的執行個體。在 CI/CD 管道中，生產環境是最後一個部署環境。
- 較高的環境 – 核心開發團隊以外的使用者可存取的所有環境。這可能包括生產環境、生產前環境以及用於使用者接受度測試的環境。

epic

在敏捷方法中，有助於組織工作並排定工作優先順序的功能類別。epic 提供要求和實作任務的高層級描述。例如， AWS CAF 安全概念包括身分和存取管理、偵測控制、基礎設施安全、資料保護和事件回應。如需有關 AWS 遷移策略中的 Epic 的詳細資訊，請參閱計畫實作指南。

ERP

請參閱企業資源規劃。

探索性資料分析 (EDA)

分析資料集以了解其主要特性的過程。您收集或彙總資料，然後執行初步調查以尋找模式、偵測異常並檢查假設。透過計算摘要統計並建立資料可視化來執行 EDA。

F

事實資料表

星狀結構描述中的中央資料表。它存放有關業務操作的量化資料。一般而言，事實資料表包含兩種類型的資料欄：包含度量的資料，以及包含維度資料表外部索引鍵的資料欄。

快速失敗

一種使用頻繁和增量測試來縮短開發生命週期的理念。這是敏捷方法的關鍵部分。

故障隔離界限

在 中 AWS 雲端，像是可用區域 AWS 區域、控制平面或資料平面等界限會限制故障的影響，並有助於改善工作負載的彈性。如需詳細資訊，請參閱AWS 故障隔離界限。

功能分支

請參閱分支。

特徵

用來進行預測的輸入資料。例如，在製造環境中，特徵可能是定期從製造生產線擷取的影像。

功能重要性

特徵對於模型的預測有多重要。這通常表示為可以透過各種技術來計算的數值得分，例如 Shapley Additive Explanations (SHAP) 和積分梯度。如需詳細資訊，請參閱 的機器學習模型可解譯性 AWS。

特徵轉換

優化 ML 程序的資料，包括使用其他來源豐富資料、調整值、或從單一資料欄位擷取多組資訊。這可讓 ML 模型從資料中受益。例如，如果將「2021-05-27 00:15:37」日期劃分為「2021」、「五月」、「週四」和「15」，則可以協助學習演算法學習與不同資料元件相關聯的細微模式。

少量擷取提示

在要求 LLM 執行類似的任務之前，提供少量示範任務和所需輸出的範例。此技術是內容內學習的應用程式，其中模型會從內嵌在提示中的範例 (快照) 中學習。對於需要特定格式、推理或網域知識的任務，少量的提示可以有效。另請參閱零鏡頭提示。

FGAC

請參閱精細存取控制。

精細存取控制 (FGAC)

使用多個條件來允許或拒絕存取請求。

閃切遷移

一種資料庫遷移方法，透過變更資料擷取使用連續資料複寫，以盡可能在最短的時間內遷移資料，而不是使用分階段方法。目標是將停機時間降至最低。

FM

請參閱基礎模型。

基礎模型 (FM)

大型深度學習神經網路，已針對廣義和未標記資料的大量資料集進行訓練。FMs 能夠執行各種一般任務，例如了解語言、產生文字和影像，以及自然語言的交談。如需詳細資訊，請參閱什麼是基礎模型。

G

生成式 AI

已針對大量資料進行訓練的 AI 模型子集，可使用簡單的文字提示建立新的內容和成品，例如影像、影片、文字和音訊。如需詳細資訊，請參閱什麼是生成式 AI。

地理封鎖

請參閱地理限制。

地理限制 (地理封鎖)

Amazon CloudFront 中的選項，可防止特定國家/地區的使用者存取內容分發。您可以使用允許清單或封鎖清單來指定核准和禁止的國家/地區。如需詳細資訊，請參閱 CloudFront 文件中的限制內容的地理分佈。

Gitflow 工作流程

這是一種方法，其中較低和較高環境在原始碼儲存庫中使用不同分支。Gitflow 工作流程會被視為舊版，而以幹線為基礎的工作流程是現代、偏好的方法。

黃金影像

系統或軟體的快照，做為部署該系統或軟體新執行個體的範本。例如，在製造中，黃金映像可用於在多個裝置上佈建軟體，並有助於提高裝置製造操作的速度、可擴展性和生產力。

綠地策略

新環境中缺乏現有基礎設施。對系統架構採用綠地策略時，可以選擇所有新技術，而不會限制與現有基礎設施的相容性，也稱為棕地。如果正在擴展現有基礎設施，則可能會混合棕地和綠地策略。

防護機制

有助於跨組織單位 (OU) 來管控資源、政策和合規的高層級規則。預防性防護機制會強制執行政策，以確保符合合規標準。透過使用服務控制政策和 IAM 許可界限來將其實作。偵測性防護機制可偵測政策違規和合規問題，並產生提醒以便修正。它們是透過使用 AWS Config AWS Security Hub、Amazon GuardDuty、、Amazon Inspector AWS Trusted Advisor和自訂 AWS Lambda 檢查來實作。

H

HA

請參閱高可用性。

異質資料庫遷移

將來源資料庫遷移至使用不同資料庫引擎的目標資料庫 (例如，Oracle 至 Amazon Aurora)。異質遷移通常是重新架構工作的一部分，而轉換結構描述可能是一項複雜任務。AWS 提供有助於結構描述轉換的 AWS SCT。

高可用性 (HA)

在遇到挑戰或災難時，工作負載能夠在不介入的情況下持續運作。HA 系統的設計目的是自動容錯移轉、持續提供高品質的效能，並處理不同的負載和故障，並將效能影響降至最低。

歷史現代化

一種方法，用於現代化和升級操作技術 (OT) 系統，以更好地滿足製造業的需求。歷史資料是一種資料庫，用於從工廠中的各種來源收集和存放資料。

保留資料

從用於訓練機器學習模型的資料集中保留的部分歷史標記資料。您可以使用保留資料，透過比較模型預測與保留資料來評估模型效能。

異質資料庫遷移

將您的來源資料庫遷移至共用相同資料庫引擎的目標資料庫 (例如，Microsoft SQL Server 至 Amazon RDS for SQL Server)。同質遷移通常是主機轉換或平台轉換工作的一部分。您可以使用原生資料庫公用程式來遷移結構描述。

熱資料

經常存取的資料，例如即時資料或最近的轉譯資料。此資料通常需要高效能儲存層或類別，才能提供快速的查詢回應。

修補程序

緊急修正生產環境中的關鍵問題。由於其緊迫性，通常會在典型 DevOps 發行工作流程之外執行修補程式。

超級護理期間

在切換後，遷移團隊在雲端管理和監控遷移的應用程式以解決任何問題的時段。通常，此期間的長度為 1-4 天。在超級護理期間結束時，遷移團隊通常會將應用程式的責任轉移給雲端營運團隊。

I

IaC

將基礎設施視為程式碼。

身分型政策

連接至一或多個 IAM 主體的政策，可定義其在 AWS 雲端 環境中的許可。

閒置應用程式

90 天期間 CPU 和記憶體平均使用率在 5% 至 20% 之間的應用程式。在遷移專案中，通常會淘汰這些應用程式或將其保留在內部部署。

IIoT

請參閱工業物聯網。

不可變的基礎設施

為生產工作負載部署新基礎設施的模型，而不是更新、修補或修改現有的基礎設施。不可變基礎設施本質上比可變基礎設施更一致、可靠且可預測。如需詳細資訊，請參閱 AWS Well-Architected Framework 中的使用不可變基礎設施部署最佳實務。

傳入 (輸入) VPC

在 AWS 多帳戶架構中，接受、檢查和路由來自應用程式外部之網路連線的 VPC。AWS 安全參考架構建議您使用傳入、傳出和檢查 VPC 來設定網路帳戶，以保護應用程式與更廣泛的網際網路之間的雙向介面。

增量遷移

一種切換策略，您可以在其中將應用程式分成小部分遷移，而不是執行單一、完整的切換。例如，您最初可能只將一些微服務或使用者移至新系統。確認所有項目都正常運作之後，您可以逐步移動其他微服務或使用者，直到可以解除委任舊式系統。此策略可降低與大型遷移關聯的風險。

工業 4.0

2016 年 Klaus Schwab 推出的術語，透過連線能力、即時資料、自動化、分析和 AI/ML 的進展，指製造程序的現代化。

基礎設施

應用程式環境中包含的所有資源和資產。

基礎設施即程式碼 (IaC)

透過一組組態檔案來佈建和管理應用程式基礎設施的程序。IaC 旨在協助您集中管理基礎設施，標準化資源並快速擴展，以便新環境可重複、可靠且一致。

工業物聯網 (IIoT)

在製造業、能源、汽車、醫療保健、生命科學和農業等產業領域使用網際網路連線的感測器和裝置。如需詳細資訊，請參閱建立工業物聯網 (IIoT) 數位轉型策略。

檢查 VPC

在 AWS 多帳戶架構中，集中式 VPC 可管理 VPCs 之間（在相同或不同的 中 AWS 區域)、網際網路和內部部署網路之間的網路流量檢查。AWS 安全參考架構建議您使用傳入、傳出和檢查 VPC 來設定網路帳戶，以保護應用程式與更廣泛的網際網路之間的雙向介面。

物聯網 (IoT)

具有內嵌式感測器或處理器的相連實體物體網路，其透過網際網路或本地通訊網路與其他裝置和系統進行通訊。如需詳細資訊，請參閱什麼是 IoT？

可解釋性

機器學習模型的一個特徵，描述了人類能夠理解模型的預測如何依賴於其輸入的程度。如需詳細資訊，請參閱 的機器學習模型可解譯性 AWS。

IoT

請參閱物聯網。

IT 資訊庫 (ITIL)

一組用於交付 IT 服務並使這些服務與業務需求保持一致的最佳實務。ITIL 為 ITSM 提供了基礎。

IT 服務管理 (ITSM)

與組織的設計、實作、管理和支援 IT 服務關聯的活動。如需有關將雲端操作與 ITSM 工具整合的資訊，請參閱操作整合指南。

ITIL

請參閱 IT 資訊庫。

ITSM

請參閱 IT 服務管理。

L

標籤型存取控制 (LBAC)

強制存取控制 (MAC) 的實作，其中使用者和資料本身都會獲得明確指派的安全標籤值。使用者安全標籤和資料安全標籤之間的交集會決定使用者可以看到哪些資料列和資料欄。

登陸區域

登陸區域是架構良好的多帳戶 AWS 環境，可擴展且安全。這是一個起點，您的組織可以從此起點快速啟動和部署工作負載與應用程式，並對其安全和基礎設施環境充滿信心。如需有關登陸區域的詳細資訊，請參閱設定安全且可擴展的多帳戶 AWS 環境。

大型語言模型 (LLM)

預先訓練大量資料的深度學習 AI 模型。LLM 可以執行多個任務，例如回答問題、摘要文件、將文字翻譯成其他語言，以及完成句子。如需詳細資訊，請參閱什麼是 LLMs。

大型遷移

遷移 300 部或更多伺服器。

LBAC

請參閱標籤型存取控制。

最低權限

授予執行任務所需之最低許可的安全最佳實務。如需詳細資訊，請參閱 IAM 文件中的套用最低權限許可。

隨即轉移

請參閱 7 個 R。

小端序系統

首先儲存最低有效位元組的系統。另請參閱 Endianness。

LLM

請參閱大型語言模型。

較低的環境

請參閱 環境。

M

機器學習 (ML)

一種使用演算法和技術進行模式識別和學習的人工智慧。機器學習會進行分析並從記錄的資料 (例如物聯網 (IoT) 資料) 中學習，以根據模式產生統計模型。如需詳細資訊，請參閱機器學習。

主要分支

請參閱分支。

惡意軟體

旨在危及電腦安全或隱私權的軟體。惡意軟體可能會中斷電腦系統、洩露敏感資訊，或取得未經授權的存取。惡意軟體的範例包括病毒、蠕蟲、勒索軟體、特洛伊木馬、間諜軟體和鍵盤記錄器。

受管服務

AWS 服務 會 AWS 操作基礎設施層、作業系統和平台，而您會存取端點來存放和擷取資料。Amazon Simple Storage Service (Amazon S3) 和 Amazon DynamoDB 是受管服務的範例。這些也稱為抽象服務。

製造執行系統 (MES)

一種軟體系統，用於追蹤、監控、記錄和控制生產程序，將原物料轉換為現場成品。

MAP

請參閱遷移加速計劃。

機制

建立工具、推動工具採用，然後檢查結果以進行調整的完整程序。機制是在操作時強化和改善自身的循環。如需詳細資訊，請參閱 AWS Well-Architected Framework 中的建置機制。

成員帳戶

除了屬於組織一部分的管理帳戶 AWS 帳戶 之外的所有 AWS Organizations。一個帳戶一次只能是一個組織的成員。

製造執行系統

請參閱製造執行系統。

訊息佇列遙測傳輸 (MQTT)

根據發佈/訂閱模式的輕量型machine-to-machine(M2M) 通訊協定，適用於資源受限的 IoT 裝置。

微服務

一種小型的獨立服務，它可透過定義明確的 API 進行通訊，通常由小型獨立團隊擁有。例如，保險系統可能包含對應至業務能力 (例如銷售或行銷) 或子領域 (例如購買、索賠或分析) 的微服務。微服務的優點包括靈活性、彈性擴展、輕鬆部署、可重複使用的程式碼和適應力。如需詳細資訊，請參閱使用無 AWS 伺服器服務整合微服務。

微服務架構

一種使用獨立元件來建置應用程式的方法，這些元件會以微服務形式執行每個應用程式程序。這些微服務會使用輕量型 API，透過明確定義的介面進行通訊。此架構中的每個微服務都可以進行

更新、部署和擴展，以滿足應用程式特定功能的需求。如需詳細資訊，請參閱在 上實作微服務 AWS。

Migration Acceleration Program (MAP)

一種 AWS 計畫，提供諮詢支援、訓練和服務，協助組織建立強大的營運基礎，以移至雲端，並協助抵銷遷移的初始成本。MAP 包括用於有條不紊地執行舊式遷移的遷移方法以及一組用於自動化和加速常見遷移案例的工具。

大規模遷移

將大部分應用程式組合依波次移至雲端的程序，在每個波次中，都會以更快的速度移動更多應用程式。此階段使用從早期階段學到的最佳實務和經驗教訓來實作團隊、工具和流程的遷移工廠，以透過自動化和敏捷交付簡化工作負載的遷移。這是 AWS 遷移策略的第三階段。

遷移工廠

可透過自動化、敏捷的方法簡化工作負載遷移的跨職能團隊。遷移工廠團隊通常包括營運、業務分析師和擁有者、遷移工程師、開發人員以及從事 Sprint 工作的 DevOps 專業人員。20% 至 50% 之間的企業應用程式組合包含可透過工廠方法優化的重複模式。如需詳細資訊，請參閱此內容集中的遷移工廠的討論和雲端遷移工廠指南。

遷移中繼資料

有關完成遷移所需的應用程式和伺服器的資訊。每種遷移模式都需要一組不同的遷移中繼資料。遷移中繼資料的範例包括目標子網路、安全群組和 AWS 帳戶。

遷移模式

可重複的遷移任務，詳細描述遷移策略、遷移目的地以及所使用的遷移應用程式或服務。範例：使用 AWS Application Migration Service 重新託管遷移至 Amazon EC2。

遷移組合評定 (MPA)

線上工具，提供驗證商業案例以遷移至 的資訊 AWS 雲端。MPA 提供詳細的組合評定 (伺服器適當規模、定價、總體擁有成本比較、遷移成本分析) 以及遷移規劃 (應用程式資料分析和資料收集、應用程式分組、遷移優先順序，以及波次規劃)。MPA 工具 （需要登入） 可供所有 AWS 顧問和 APN 合作夥伴顧問免費使用。

遷移準備程度評定 (MRA)

使用 AWS CAF 取得組織雲端整備狀態的洞見、識別優缺點，以及建立行動計劃以消除已識別差距的程序。如需詳細資訊，請參閱遷移準備程度指南。MRA 是 AWS 遷移策略的第一階段。

遷移策略

用來將工作負載遷移至 的方法 AWS 雲端。如需詳細資訊，請參閱此詞彙表中的 7 個 Rs 項目，並請參閱動員您的組織以加速大規模遷移。

機器學習 (ML)

請參閱機器學習。

現代化

將過時的 (舊版或單一) 應用程式及其基礎架構轉換為雲端中靈活、富有彈性且高度可用的系統，以降低成本、提高效率並充分利用創新。如需詳細資訊，請參閱《》中的現代化應用程式的策略 AWS 雲端。

現代化準備程度評定

這項評估可協助判斷組織應用程式的現代化準備程度；識別優點、風險和相依性；並確定組織能夠在多大程度上支援這些應用程式的未來狀態。評定的結果就是目標架構的藍圖、詳細說明現代化程序的開發階段和里程碑的路線圖、以及解決已發現的差距之行動計畫。如需詳細資訊，請參閱《》中的評估應用程式的現代化準備 AWS 雲端程度。

單一應用程式 (單一)

透過緊密結合的程序作為單一服務執行的應用程式。單一應用程式有幾個缺點。如果一個應用程式功能遇到需求激增，則必須擴展整個架構。當程式碼庫增長時，新增或改進單一應用程式的功能也會變得更加複雜。若要解決這些問題，可以使用微服務架構。如需詳細資訊，請參閱將單一體系分解為微服務。

MPA

請參閱遷移產品組合評估。

MQTT

請參閱訊息佇列遙測傳輸。

多類別分類

一個有助於產生多類別預測的過程 (預測兩個以上的結果之一)。例如，機器學習模型可能會詢問「此產品是書籍、汽車還是電話？」 或者「這個客戶對哪種產品類別最感興趣？」

可變基礎設施

更新和修改生產工作負載現有基礎設施的模型。為了提高一致性、可靠性和可預測性， AWS Well-Architected Framework 建議使用不可變基礎設施做為最佳實務。

O

OAC

請參閱原始存取控制。

OAI

請參閱原始存取身分。

OCM

請參閱組織變更管理。

離線遷移

一種遷移方法，可在遷移過程中刪除來源工作負載。此方法涉及延長停機時間，通常用於小型非關鍵工作負載。

OI

請參閱 操作整合。

OLA

請參閱操作層級協議。

線上遷移

一種遷移方法，無需離線即可將來源工作負載複製到目標系統。連接至工作負載的應用程式可在遷移期間繼續運作。此方法涉及零至最短停機時間，通常用於關鍵的生產工作負載。

OPC-UA

請參閱開放程序通訊 - 統一架構。

開放程序通訊 - 統一架構 (OPC-UA)

用於工業自動化的machine-to-machine(M2M) 通訊協定。OPC-UA 提供資料加密、身分驗證和授權機制的互通性標準。

操作水準協議 (OLA)

一份協議，闡明 IT 職能群組承諾向彼此提供的內容，以支援服務水準協議 (SLA)。

操作整備審查 (ORR)

問題和相關最佳實務的檢查清單，可協助您了解、評估、預防或減少事件和可能失敗的範圍。如需詳細資訊，請參閱 AWS Well-Architected Framework 中的操作準備度審查 (ORR)。

操作技術 (OT)

使用實體環境控制工業操作、設備和基礎設施的硬體和軟體系統。在製造業中，整合 OT 和資訊技術 (IT) 系統是工業 4.0 轉型的關鍵重點。

操作整合 (OI)

在雲端中將操作現代化的程序，其中包括準備程度規劃、自動化和整合。如需詳細資訊，請參閱操作整合指南。

組織追蹤

由 建立的線索 AWS CloudTrail 會記錄 AWS 帳戶 組織中所有 的所有事件 AWS Organizations。在屬於組織的每個 AWS 帳戶 中建立此追蹤，它會跟蹤每個帳戶中的活動。如需詳細資訊，請參閱 CloudTrail 文件中的建立組織追蹤。

組織變更管理 (OCM)

用於從人員、文化和領導力層面管理重大、顛覆性業務轉型的架構。OCM 透過加速變更採用、解決過渡問題，以及推動文化和組織變更，協助組織為新系統和策略做好準備，並轉移至新系統和策略。在 AWS 遷移策略中，此架構稱為人員加速，因為雲端採用專案所需的變更速度。如需詳細資訊，請參閱 OCM 指南。

原始存取控制 (OAC)

CloudFront 中的增強型選項，用於限制存取以保護 Amazon Simple Storage Service (Amazon S3) 內容。OAC 支援所有 S3 儲存貯體中的所有伺服器端加密 AWS KMS (SSE-KMS) AWS 區域，以及對 S3 儲存貯體的動態`PUT`和`DELETE`請求。

原始存取身分 (OAI)

CloudFront 中的一個選項，用於限制存取以保護 Amazon S3 內容。當您使用 OAI 時，CloudFront 會建立一個可供 Amazon S3 進行驗證的主體。經驗證的主體只能透過特定 CloudFront 分發來存取 S3 儲存貯體中的內容。另請參閱 OAC，它可提供更精細且增強的存取控制。

ORR

請參閱操作整備審核。

OT

請參閱操作技術。

傳出 (輸出) VPC

在 AWS 多帳戶架構中，處理從應用程式內啟動之網路連線的 VPC。AWS 安全參考架構建議您使用傳入、傳出和檢查 VPC 來設定網路帳戶，以保護應用程式與更廣泛的網際網路之間的雙向介面。

P

許可界限

附接至 IAM 主體的 IAM 管理政策，可設定使用者或角色擁有的最大許可。如需詳細資訊，請參閱 IAM 文件中的許可界限。

個人身分識別資訊 (PII)

直接檢視或與其他相關資料配對時，可用來合理推斷個人身分的資訊。PII 的範例包括名稱、地址和聯絡資訊。

PII

請參閱個人身分識別資訊。

手冊

一組預先定義的步驟，可擷取與遷移關聯的工作，例如在雲端中提供核心操作功能。手冊可以採用指令碼、自動化執行手冊或操作現代化環境所需的程序或步驟摘要的形式。

PLC

請參閱可程式設計邏輯控制器。

PLM

請參閱產品生命週期管理。

政策

可定義許可的物件（請參閱身分型政策)、指定存取條件（請參閱資源型政策)，或定義組織中所有帳戶的最大許可 AWS Organizations（請參閱服務控制政策)。

混合持久性

根據資料存取模式和其他需求，獨立選擇微服務的資料儲存技術。如果您的微服務具有相同的資料儲存技術，則其可能會遇到實作挑戰或效能不佳。如果微服務使用最適合其需求的資料儲存，則

可以更輕鬆地實作並達到更好的效能和可擴展性。如需詳細資訊，請參閱在微服務中啟用資料持久性。

組合評定

探索、分析應用程式組合並排定其優先順序以規劃遷移的程序。如需詳細資訊，請參閱評估遷移準備程度。

述詞

傳回 `true`或 的查詢條件`false`，通常位於 `WHERE`子句中。

述詞下推

一種資料庫查詢最佳化技術，可在傳輸前篩選查詢中的資料。這可減少必須從關聯式資料庫擷取和處理的資料量，並改善查詢效能。

預防性控制

旨在防止事件發生的安全控制。這些控制是第一道防線，可協助防止對網路的未經授權存取或不必要變更。如需詳細資訊，請參閱在 AWS上實作安全控制中的預防性控制。

委託人

中可執行動作和存取資源 AWS 的實體。此實體通常是 AWS 帳戶、IAM 角色或使用者的根使用者。如需詳細資訊，請參閱 IAM 文件中角色術語和概念中的主體。

依設計的隱私權

透過整個開發程序將隱私權納入考量的系統工程方法。

私有託管區域

一種容器，它包含有關您希望 Amazon Route 53 如何回應一個或多個 VPC 內的域及其子域之 DNS 查詢的資訊。如需詳細資訊，請參閱 Route 53 文件中的使用私有託管區域。

主動控制

旨在防止部署不合規資源的安全控制。這些控制項會在佈建資源之前對其進行掃描。如果資源不符合控制項，則不會佈建。如需詳細資訊，請參閱 AWS Control Tower 文件中的控制項參考指南，並參閱實作安全控制項中的主動控制項。 AWS

產品生命週期管理 (PLM)

管理產品整個生命週期的資料和程序，從設計、開發和啟動，到成長和成熟，再到拒絕和移除。

生產環境

請參閱 環境。

可程式設計邏輯控制器 (PLC)

在製造中，高度可靠、可調整的電腦，可監控機器並自動化製造程序。

提示鏈結

使用一個 LLM 提示的輸出做為下一個提示的輸入，以產生更好的回應。此技術用於將複雜任務分解為子任務，或反覆精簡或展開初步回應。它有助於提高模型回應的準確性和相關性，並允許更精細、個人化的結果。

擬匿名化

將資料集中的個人識別符取代為預留位置值的程序。假名化有助於保護個人隱私權。假名化資料仍被視為個人資料。

發佈/訂閱 (pub/sub)

一種模式，可啟用微服務之間的非同步通訊，以提高可擴展性和回應能力。例如，在微服務型 MES 中，微服務可以將事件訊息發佈到其他微服務可訂閱的頻道。系統可以新增新的微服務，而無需變更發佈服務。

Q

查詢計劃

一系列步驟，如指示，用於存取 SQL 關聯式資料庫系統中的資料。

查詢計劃迴歸

在資料庫服務優化工具選擇的計畫比對資料庫環境進行指定的變更之前的計畫不太理想時。這可能因為對統計資料、限制條件、環境設定、查詢參數繫結的變更以及資料庫引擎的更新所導致。

R

RACI 矩陣

請參閱負責、負責、諮詢、告知 (RACI)。

RAG

請參閱擷取增強產生。

勒索軟體

一種惡意軟體，旨在阻止對計算機系統或資料的存取，直到付款為止。

RASCI 矩陣

請參閱負責、負責、諮詢、告知 (RACI)。

RCAC

請參閱資料列和資料欄存取控制。

僅供讀取複本

用於唯讀用途的資料庫複本。您可以將查詢路由至僅供讀取複本以減少主資料庫的負載。

重新架構師

請參閱 7 個 R。

復原點目標 (RPO)

自上次資料復原點以來可接受的時間上限。這會決定最後一個復原點與服務中斷之間可接受的資料遺失。

復原時間目標 (RTO)

服務中斷與服務還原之間的可接受延遲上限。

重構

請參閱 7 個 R。

區域

地理區域中的 AWS 資源集合。每個 AWS 區域 都獨立於其他 ，以提供容錯能力、穩定性和彈性。如需詳細資訊，請參閱指定 AWS 區域 您的帳戶可以使用哪些 。

迴歸

預測數值的 ML 技術。例如，為了解決「這房子會賣什麼價格？」的問題 ML 模型可以使用線性迴歸模型，根據已知的房屋事實 (例如，平方英尺) 來預測房屋的銷售價格。

重新託管

請參閱 7 個 R。

版本

在部署程序中，它是將變更提升至生產環境的動作。

重新定位

請參閱 7 個 R。

Replatform

請參閱 7 個 R。

回購

請參閱 7 個 R。

彈性

應用程式抵禦中斷或從中斷中復原的能力。在 中規劃彈性時，高可用性和災難復原是常見的考量 AWS 雲端。如需詳細資訊，請參閱AWS 雲端 彈性。

資源型政策

附接至資源的政策，例如 Amazon S3 儲存貯體、端點或加密金鑰。這種類型的政策會指定允許存取哪些主體、支援的動作以及必須滿足的任何其他條件。

負責者、當責者、事先諮詢者和事後告知者 (RACI) 矩陣

矩陣，定義所有涉及遷移活動和雲端操作之各方的角色和責任。矩陣名稱衍生自矩陣中定義的責任類型：負責人 (R)、責任 (A)、已諮詢 (C) 和知情 (I)。支援 (S) 類型為選用。如果您包含支援，則矩陣稱為 RASCI 矩陣，如果您排除它，則稱為 RACI 矩陣。

回應性控制

一種安全控制，旨在驅動不良事件或偏離安全基準的補救措施。如需詳細資訊，請參閱在 AWS上實作安全控制中的回應性控制。

保留

請參閱 7 個 R。

淘汰

請參閱 7 Rs。

檢索增強生成 (RAG)

一種生成式 AI 技術，其中 LLM 會在產生回應之前參考訓練資料來源以外的授權資料來源。例如，RAG 模型可能會對組織的知識庫或自訂資料執行語意搜尋。如需詳細資訊，請參閱什麼是 RAG。

輪換

定期更新秘密的程序，讓攻擊者更難存取登入資料。

資料列和資料欄存取控制 (RCAC)

使用已定義存取規則的基本、彈性 SQL 表達式。RCAC 包含資料列許可和資料欄遮罩。

RPO

請參閱復原點目標。

RTO

請參閱復原時間目標。

執行手冊

執行特定任務所需的一組手動或自動程序。這些通常是為了簡化重複性操作或錯誤率較高的程序而建置。

S

SAML 2.0

許多身分提供者 (IdP) 使用的開放標準。此功能會啟用聯合單一登入 (SSO)，讓使用者可以登入 AWS Management Console 或呼叫 AWS API 操作，而不必為您組織中的每個人在 IAM 中建立使用者。如需有關以 SAML 2.0 為基礎的聯合詳細資訊，請參閱 IAM 文件中的關於以 SAML 2.0 為基礎的聯合。

SCADA

請參閱監督控制和資料擷取。

SCP

請參閱服務控制政策。

秘密

您以加密形式存放的 AWS Secrets Manager機密或限制資訊，例如密碼或使用者登入資料。它由秘密值及其中繼資料組成。秘密值可以是二進位、單一字串或多個字串。如需詳細資訊，請參閱 Secrets Manager 文件中的 Secrets Manager 秘密中的什麼內容？。

設計安全性

透過整個開發程序將安全性納入考量的系統工程方法。

安全控制

一種技術或管理防護機制，它可預防、偵測或降低威脅行為者利用安全漏洞的能力。安全控制有四種主要類型：預防性、偵測性、回應性和主動性。

安全強化

減少受攻擊面以使其更能抵抗攻擊的過程。這可能包括一些動作，例如移除不再需要的資源、實作授予最低權限的安全最佳實務、或停用組態檔案中不必要的功能。

安全資訊與事件管理 (SIEM) 系統

結合安全資訊管理 (SIM) 和安全事件管理 (SEM) 系統的工具與服務。SIEM 系統會收集、監控和分析來自伺服器、網路、裝置和其他來源的資料，以偵測威脅和安全漏洞，並產生提醒。

安全回應自動化

預先定義和程式設計的動作，旨在自動回應或修復安全事件。這些自動化可做為偵測或回應式安全控制，協助您實作 AWS 安全最佳實務。自動化回應動作的範例包括修改 VPC 安全群組、修補 Amazon EC2 執行個體或輪換登入資料。

伺服器端加密

由 AWS 服務 接收資料的 在其目的地加密資料。

服務控制政策 (SCP)

為 AWS Organizations中的組織的所有帳戶提供集中控制許可的政策。SCP 會定義防護機制或設定管理員可委派給使用者或角色的動作限制。您可以使用 SCP 作為允許清單或拒絕清單，以指定允許或禁止哪些服務或動作。如需詳細資訊，請參閱 AWS Organizations 文件中的服務控制政策。

服務端點

的進入點 URL AWS 服務。您可以使用端點，透過程式設計方式連接至目標服務。如需詳細資訊，請參閱 AWS 一般參考 中的 AWS 服務 端點。

服務水準協議 (SLA)

一份協議，闡明 IT 團隊承諾向客戶提供的服務，例如服務正常執行時間和效能。

服務層級指標 (SLI)

服務效能層面的測量，例如其錯誤率、可用性或輸送量。

服務層級目標 (SLO)

代表服務運作狀態的目標指標，由服務層級指標測量。

共同責任模式

描述您與 共同 AWS 承擔雲端安全與合規責任的模型。 AWS 負責雲端的安全，而 負責雲端的安全。如需詳細資訊，請參閱共同責任模式。

SIEM

請參閱安全資訊和事件管理系統。

單一故障點 (SPOF)

應用程式的單一關鍵元件故障，可能會中斷系統。

SLA

請參閱服務層級協議。

SLI

請參閱服務層級指標。

SLO

請參閱服務層級目標。

先拆分後播種模型

擴展和加速現代化專案的模式。定義新功能和產品版本時，核心團隊會進行拆分以建立新的產品團隊。這有助於擴展組織的能力和服務，提高開發人員生產力，並支援快速創新。如需詳細資訊，請參閱 中的階段式應用程式現代化方法 AWS 雲端。

SPOF

請參閱單一故障點。

星狀結構描述

使用一個大型事實資料表來存放交易或測量資料的資料庫組織結構，並使用一或多個較小的維度資料表來存放資料屬性。此結構旨在用於資料倉儲或商業智慧用途。

Strangler Fig 模式

一種現代化單一系統的方法，它會逐步重寫和取代系統功能，直到舊式系統停止使用為止。此模式源自無花果藤，它長成一棵馴化樹並最終戰勝且取代了其宿主。該模式由 Martin Fowler 引入，作

為重寫單一系統時管理風險的方式。如需有關如何套用此模式的範例，請參閱使用容器和 Amazon API Gateway 逐步現代化舊版 Microsoft ASP.NET (ASMX) Web 服務。

子網

您 VPC 中的 IP 地址範圍。子網必須位於單一可用區域。

監控控制和資料擷取 (SCADA)

在製造中，使用硬體和軟體來監控實體資產和生產操作的系統。

對稱加密

使用相同金鑰來加密及解密資料的加密演算法。

合成測試

以模擬使用者互動的方式測試系統，以偵測潛在問題或監控效能。您可以使用 Amazon CloudWatch Synthetics 來建立這些測試。

系統提示

一種向 LLM 提供內容、指示或指導方針以指示其行為的技術。系統提示有助於設定內容，並建立與使用者互動的規則。

T

標籤

做為中繼資料以組織 AWS 資源的鍵值對。標籤可協助您管理、識別、組織、搜尋及篩選資源。如需詳細資訊，請參閱標記您的 AWS 資源。

目標變數

您嘗試在受監督的 ML 中預測的值。這也被稱為結果變數。例如，在製造設定中，目標變數可能是產品瑕疵。

任務清單

用於透過執行手冊追蹤進度的工具。任務清單包含執行手冊的概觀以及要完成的一般任務清單。對於每個一般任務，它包括所需的預估時間量、擁有者和進度。

測試環境

請參閱 環境。

訓練

為 ML 模型提供資料以供學習。訓練資料必須包含正確答案。學習演算法會在訓練資料中尋找將輸入資料屬性映射至目標的模式 (您想要預測的答案)。它會輸出擷取這些模式的 ML 模型。可以使用 ML 模型，來預測您不知道的目標新資料。

傳輸閘道

可以用於互連 VPC 和內部部署網路的網路傳輸中樞。如需詳細資訊，請參閱 AWS Transit Gateway 文件中的什麼是傳輸閘道。

主幹型工作流程

這是一種方法，開發人員可在功能分支中本地建置和測試功能，然後將這些變更合併到主要分支中。然後，主要分支會依序建置到開發環境、生產前環境和生產環境中。

受信任的存取權

將許可授予您指定的服務，以代表您在組織中 AWS Organizations 及其帳戶中執行任務。受信任的服務會在需要該角色時，在每個帳戶中建立服務連結角色，以便為您執行管理工作。如需詳細資訊，請參閱 文件中的 AWS Organizations 搭配使用 AWS Organizations 與其他 AWS 服務。

調校

變更訓練程序的各個層面，以提高 ML 模型的準確性。例如，可以透過產生標籤集、新增標籤、然後在不同的設定下多次重複這些步驟來訓練 ML 模型，以優化模型。

雙比薩團隊

兩個比薩就能吃飽的小型 DevOps 團隊。雙披薩團隊規模可確保軟體開發中的最佳協作。

U

不確定性

這是一個概念，指的是不精確、不完整或未知的資訊，其可能會破壞預測性 ML 模型的可靠性。有兩種類型的不確定性：認知不確定性是由有限的、不完整的資料引起的，而隨機不確定性是由資料中固有的噪聲和隨機性引起的。如需詳細資訊，請參閱量化深度學習系統的不確定性指南。

未區分的任務

也稱為繁重工作，這是建立和操作應用程式的必要工作，但不為最終使用者提供直接價值或提供競爭優勢。未區分任務的範例包括採購、維護和容量規劃。

較高的環境

請參閱 環境。

V

清空

一種資料庫維護操作，涉及增量更新後的清理工作，以回收儲存並提升效能。

版本控制

追蹤變更的程序和工具，例如儲存庫中原始程式碼的變更。

VPC 對等互連

兩個 VPC 之間的連線，可讓您使用私有 IP 地址路由流量。如需詳細資訊，請參閱 Amazon VPC 文件中的什麼是 VPC 對等互連。

漏洞

危害系統安全性的軟體或硬體瑕疵。

W

暖快取

包含經常存取的目前相關資料的緩衝快取。資料庫執行個體可以從緩衝快取讀取，這比從主記憶體或磁碟讀取更快。

暖資料

不常存取的資料。查詢這類資料時，通常可接受中等速度的查詢。

視窗函數

SQL 函數，對與目前記錄有某種程度關聯的資料列群組執行計算。視窗函數適用於處理任務，例如根據目前資料列的相對位置計算移動平均值或存取資料列的值。

工作負載

提供商業價值的資源和程式碼集合，例如面向客戶的應用程式或後端流程。

工作串流

遷移專案中負責一組特定任務的功能群組。每個工作串流都是獨立的，但支援專案中的其他工作串流。例如，組合工作串流負責排定應用程式、波次規劃和收集遷移中繼資料的優先順序。組合工作串流將這些資產交付至遷移工作串流，然後再遷移伺服器和應用程式。

WORM

請參閱寫入一次，讀取許多。

WQF

請參閱AWS 工作負載資格架構。

寫入一次，讀取許多 (WORM)

儲存模型，可一次性寫入資料，並防止刪除或修改資料。授權使用者可以視需要多次讀取資料，但無法變更資料。此資料儲存基礎設施被視為不可變。

Z

零時差入侵

利用零時差漏洞的攻擊，通常是惡意軟體。

零時差漏洞

生產系統中未緩解的缺陷或漏洞。威脅行為者可以使用這種類型的漏洞來攻擊系統。開發人員經常因為攻擊而意識到漏洞。

零鏡頭提示

提供 LLM 執行任務的指示，但沒有可協助引導任務的範例 (快照)。LLM 必須使用其預先訓練的知識來處理任務。零鏡頭提示的有效性取決於任務的複雜性和提示的品質。另請參閱少量擷取提示。

殭屍應用程式

CPU 和記憶體平均使用率低於 5% 的應用程式。在遷移專案中，通常會淘汰這些應用程式。

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。