에서 Apache Iceberg 사용 AWS

AWS 권장 가이드

AWS 권장 가이드: 에서 Apache Iceberg 사용 AWS

Table of Contents

에서 Apache Iceberg 사용 AWS

Amazon Web Services(기여자)

2025년 11월(문서 기록)

Apache Iceberg는 성능을 개선하면서 테이블 관리를 간소화하는 오픈 소스 테이블 형식입니다.Amazon EMR AWS Glue, Amazon Athena, Amazon Redshift와 같은 AWS 분석 서비스에는 Iceberg에 대한 기본 지원이 포함되어 있으므로 Amazon Simple Storage Service(Amazon S3)를 기반으로 트랜잭션 데이터 레이크를 쉽게 빌드할 수 있습니다 AWS.

또한 차세대 Amazon SageMaker는 데이터 레이크, 데이터 웨어하우스, 타사 및 페더레이션 소스 간의 데이터 액세스를 통합하는 오픈 레이크하우스 아키텍처를 기반으로 구축되었습니다. AWS 레이크하우스는 Iceberg와 완벽하게 호환되며 Iceberg REST API를 사용하여 제자리에 있는 데이터에 액세스하고 쿼리할 수 있는 유연성을 제공합니다.

이 기술 가이드는 다양한에서 Iceberg를 시작하는 방법에 대한 지침을 제공하며 비용과 성능을 최적화하면서 AWS 대규모로 Iceberg를 실행하기 위한 모범 사례와 권장 사항을 AWS 서비스포함합니다.

Iceberg로 막 시작하든 기존 Iceberg 워크로드를 최적화하려는 숙련된 사용자이든 AWS이 가이드는 프로젝트의 모든 단계에 대한 귀중한 인사이트를 제공합니다.

이 가이드에는 다음 내용이 포함됩니다.

- 최신 데이터 레이크
- Athena SQL에서 Iceberg 테이블 시작하기
- Amazon EMR에서 Iceberg 작업
- 에서 Iceberg 작업 AWS Glue
- Spark를 사용하여 Iceberg 테이블 작업
- Trino를 사용하여 Iceberg 테이블 작업
- Amazon Data Firehose를 사용하여 Iceberg 테이블 작업
- Athena SQL을 사용하여 Iceberg 테이블 작업
- PyIceberg 테이블 작업
- Iceberg 테이블 형식 사양 버전 3 작업
- 기존 테이블을 Iceberg로 마이그레이션
- Iceberg 워크로드 최적화 모범 사례

최신 데이터 레이크

최신 데이터 레이크의 고급 사용 사례

데이터 스토리지의 진화는 데이터베이스에서 각 기술이 고유한 비즈니스 및 데이터 요구 사항을 해결하는 데이터 웨어하우스 및 데이터 레이크로 진행되었습니다. 기존 데이터베이스는 구조화된 데이터 및 트랜잭션 워크로드를 처리하는 데 능숙했지만 데이터 볼륨이 증가함에 따라 성능 문제가 발생했습니다. 데이터 웨어하우스는 성능 및 확장성 문제를 해결하기 위해 등장했지만 데이터베이스와 마찬가지로 수직 통합 시스템 내의 독점 형식에 의존했습니다.

데이터 레이크는 비용, 확장성 및 유연성 측면에서 데이터를 저장하는 가장 좋은 옵션 중 하나를 제공합니다. 데이터 레이크를 사용하면 저렴한 비용으로 대량의 정형 및 비정형 데이터를 유지할 수 있으며, 비즈니스 인텔리전스 보고부터 빅 데이터 처리, 실시간 분석, 기계 학습 및 생성형 인공 지능(AI)에 이르기까지 다양한 유형의 분석 워크로드에이 데이터를 사용하여 더 나은 결정을 내릴 수 있습니다.

이러한 이점에도 불구하고 데이터 레이크는 처음에는 데이터베이스와 유사한 기능으로 설계되지 않았습니다. 데이터 레이크는 원자성, 일관성, 격리 및 내구성(ACID) 처리 시맨틱을 지원하지 않습니다.이 시맨틱은 다양한 기술을 사용하여 수백 또는 수천 명의 사용자에 걸쳐 데이터를 대규모로 효과적으로 최적화하고 관리하는 데 필요할 수 있습니다. 데이터 레이크는 다음 기능에 대한 기본 지원을 제공하지 않습니다.

- 비즈니스의 데이터 변경에 따라 효율적인 레코드 수준 업데이트 및 삭제 수행
- 테이블이 수백만 개의 파일과 수십만 개의 파티션으로 증가함에 따라 쿼리 성능 관리
- 여러 동시 라이터 및 리더에서 데이터 일관성 보장
- 작업 도중 쓰기 작업이 실패할 때 데이터 손상 방지
- 데이터 세트를 (부분적으로) 다시 작성하지 않고 시간 경과에 따른 테이블 스키마 개선

이러한 문제는 변경 데이터 캡처(CDC) 처리 또는 개인 정보 보호와 관련된 사용 사례, 데이터 삭제, 스트리밍 데이터 수집과 같은 사용 사례에서 특히 보편화되어 최적이 아닌 테이블이 될 수 있습니다.

기존 Hive 형식 테이블을 사용하는 데이터 레이크는 전체 파일에 대해서만 쓰기 작업을 지원합니다. 따라서 업데이트 및 삭제를 구현하기 어렵고 시간이 많이 걸리며 비용이 많이 듭니다. 또한 데이터 무결성과 일관성을 보장하려면 ACID 준수 시스템에서 제공되는 동시성 제어 및 보장이 필요합니다.

이러한 과제로 인해 사용자는 완전히 통합되었지만 독점적인 플랫폼 중에서 선택하거나 잠재적 가치를 실현하기 위해 지속적인 유지 관리 및 마이그레이션이 필요한 공급업체 중립적이지만 리소스 집약적인 자체 구축 데이터 레이크를 선택할 수 있습니다.

이러한 문제를 해결하는 데 도움이 되도록 Iceberg는 Amazon S3와 같은 비용 효율적인 시스템에서 스토리지를 지원하면서 데이터 레이크의 최적화 및 관리 오버헤드를 간소화하는 추가 데이터베이스와 유사한 기능을 제공합니다.

Apache Iceberg 소개

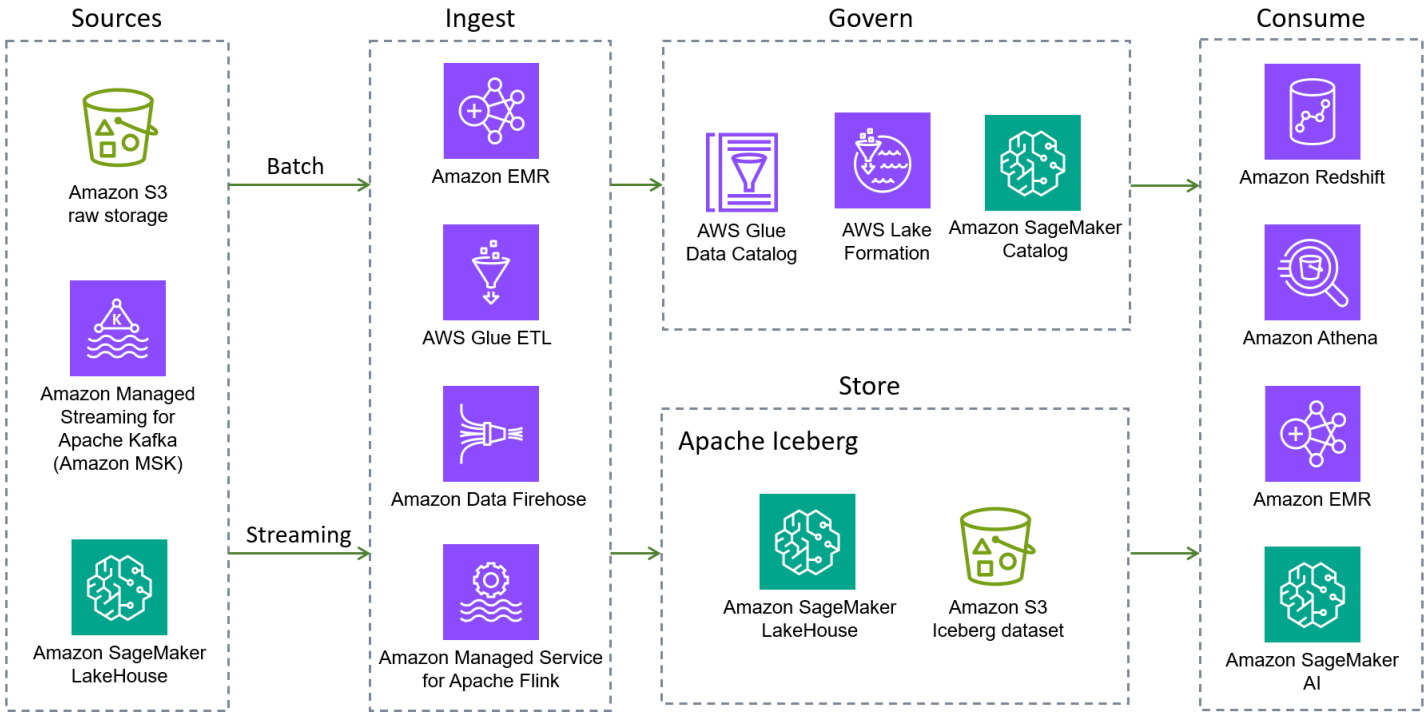
Apache Iceberg는 데이터베이스 또는 데이터 웨어하우스에서만 사용할 수 있었던 데이터 레이크 테이블의 기능을 제공하는 오픈 소스 테이블 형식입니다. 확장 및 성능을 위해 설계되었으며 수백 기가바이트가 넘는 테이블을 관리하는 데 적합합니다. Iceberg 테이블의 몇 가지 주요 기능은 다음과 같습니다.

- 삭제, 업데이트 및 병합. Iceberg는 데이터 레이크 테이블에 사용할 데이터 웨어하우징을 위한 표준 SQL 명령을 지원합니다.
- 빠른 스캔 계획 및 고급 필터링. Iceberg는 엔진에서 쿼리 계획 및 실행 속도를 높이는 데 사용할 수 있는 파티션 및 열 수준 통계와 같은 메타데이터를 저장합니다.
- 전체 스키마 진화. Iceberg는 부작용 없이 열 추가, 삭제, 업데이트 또는 이름 변경을 지원합니다.
- 파티션 진화. 데이터 볼륨 또는 쿼리 패턴이 변경되면 테이블의 파티션 레이아웃을 업데이트할 수 있습니다. Iceberg는 테이블이 분할된 열을 변경하거나, 복합 파티션에 열을 추가하거나, 복합 파티션에서 열을 제거할 수 있도록 지원합니다.
- 숨겨진 파티셔닝. 이 기능은 불필요한 파티션을 자동으로 읽는 것을 방지합니다. 따라서 사용자가 테이블의 파티셔닝 세부 정보를 이해하거나 쿼리에 추가 필터를 추가할 필요가 없습니다.
- 버전 롤백. 사용자는 트랜잭션 전 상태로 되돌려 문제를 신속하게 해결할 수 있습니다.
- 시간 이동. 사용자는 테이블의 특정 이전 버전을 쿼리할 수 있습니다.
- 직렬화 가능한 격리. 테이블 변경은 원자성이므로 독자는 부분 또는 커밋되지 않은 변경 사항을 볼 수 없습니다.
- 동시 라이터. Iceberg는 낙관적 동시성을 사용하여 여러 트랜잭션이 성공할 수 있도록 합니다. 충돌이 발생할 경우 작성자 중 한 명이 트랜잭션을 다시 시도해야 합니다.
- 파일 형식을 엽니다. Iceberg는 Apache Parquet, Apache Avro 및 Apache ORC를 비롯한 여러 오픈 소스 파일 형식을 지원합니다.

요약하면 Iceberg 형식을 사용하는 데이터 레이크는 트랜잭션 일관성, 속도, 규모 및 스키마 진화의 이점을 누릴 수 있습니다. 이러한 기능 및 기타 Iceberg 기능에 대한 자세한 내용은 Apache Iceberg 설명서를 참조하세요.

AWS Apache Iceberg에 대한 지원

Apache Iceberg는 Amazon EMR, Amazon Amazon Athena Redshift, AWS Glue및 Amazon SageMaker AWS 서비스 와 같은에서 지원됩니다. 다음 다이어그램은 Iceberg를 기반으로 하는 데이터 레이크의 간소화된 참조 아키텍처를 보여줍니다.



다음은 네이티브 Iceberg 통합을 AWS 서비스 제공합니다. 간접적으로 또는 Iceberg 라이브러리를 패키징하여 Iceberg와 상호 작용할 수 AWS 서비스 있는 추가 기능이 있습니다.

- Amazon S3는 내구성, 가용성, 확장성, 보안, 규정 준수 및 감사 기능으로 인해 데이터 레이크를 빌드하기에 가장 적합합니다. Iceberg는 Amazon S3와 원활하게 상호 작용하도록 설계 및 구축되었으며 Iceberg 설명서에 나열된 많은 Amazon S3 기능을 지원합니다. 또한 Amazon S3 Tables는 Iceberg 지원이 내장된 첫 번째 클라우드 객체 스토어를 제공하고 대규모 테이블 형식 데이터 저장을 간소화합니다. Iceberg에 대한 S3 Tables 지원을 사용하면 인기 AWS 있는 타사 쿼리 엔진을 사용하여 테이블 형식 데이터를 쉽게 쿼리할 수 있습니다.
- 차세대 SageMaker는 Amazon S3 데이터 레이크, Amazon Redshift 데이터 웨어하우스, 타사 및 페더레이션 데이터 소스 전반의 데이터 액세스를 통합하는 오픈 레이크하우스 아키텍처를 기반으로 구축되었습니다. 이러한 기능을 사용하면 단일 데이터 사본에서 강력한 분석 및 AI/ML 애플리케이션을 구축할 수 있습니다. 레이크하우스는 Iceberg와 완벽하게 호환되므로 Iceberg REST API를 사용하여 현재 위치에 있는 데이터에 액세스하고 쿼리할 수 있는 유연성이 있습니다.

- Amazon EMR은 Apache Spark, Flink, Trino 및 Hive와 같은 오픈 소스 프레임워크를 사용하여 페타바이트 규모의 데이터 처리, 대화형 분석 및 기계 학습을 위한 빅 데이터 솔루션입니다. Amazon EMR은 사용자 지정 Amazon Elastic Compute Cloud(Amazon EC2) 클러스터, Amazon Elastic Kubernetes Service(Amazon EKS), AWS Outposts또는 Amazon EMR Serverless에서 실행할 수 있습니다.
- Amazon Athena는 오픈 소스 프레임워크를 기반으로 구축된 서버리스 대화형 분석 서비스입니다. 오픈 테이블 및 파일 형식을 지원하며 페타바이트 단위의 데이터를 분석하는 간단하고 유연한 방법을 제공합니다. Athena는 Iceberg에 대한 읽기, 시간 이동, 쓰기 및 DDL 쿼리를 기본적으로 지원하고 Iceberg 메타스토어 AWS Glue Data Catalog 에를 사용합니다.
- Amazon Redshift는 클러스터 기반 및 서버리스 배포 옵션을 모두 지원하는 페타바이트 규모의 클라우드 데이터 웨어하우스입니다. Amazon Redshift Spectrum은에 등록 AWS Glue Data Catalog 되고 Amazon S3에 저장된 외부 테이블을 쿼리할 수 있습니다. Redshift Spectrum은 Iceberg 스토리지 형식도 지원합니다.
- AWS Glue는 분석, 기계 학습(ML) 및 애플리케이션 개발을 위해 여러 소스에서 데이터를 더 쉽게 검색, 준비, 이동 및 통합할 수 있는 서버리스 데이터 통합 서비스입니다. Iceberg와 완전히 통합됩니다. 특히 AWS Glue 작업을 사용하여 Iceberg 테이블에서 읽기 및 쓰기 작업을 수행하고, AWS Glue Data Catalog (Hive 메타스토어 호환)를 통해 테이블을 관리하고, AWS Glue 크롤러를 사용하여 테이블을 자동으로 검색 및 등록하고, Data Quality 기능을 통해 Iceberg 테이블의 AWS Glue 데이터 품질을 평가할 수 있습니다. AWS Glue Data Catalog 또한는 열 통계 수집, Iceberg 테이블의 각 열에 대한 고유 값(NDVs) 수 계산 및 업데이트, 자동 테이블 최적화(압축, 스냅샷 보존, 분리된 파일 삭제)를 지원합니다.는 AWS 서비스 및 타사 애플리케이션 목록에서 Iceberg 테이블로의 제로 ETL 통합 AWS Glue 도 지원합니다.
- Amazon Data Firehose는 Amazon S3, Amazon Redshift, Amazon OpenSearch Service, Amazon OpenSearch Serverless, Splunk, Apache Iceberg 테이블과 같은 대상과 Datadog, Dynatrace, LogicMonitor, MongoDB, New Relic, Coralogix, Elastic 등 지원되는 타사 서비스 공급자가 소유한 모든 사용자 지정 HTTP 또는 HTTP 엔드포인트에 실시간 스트리밍 데이터를 제공하기 위한 완전 관리형 서비스입니다. Firehose를 사용하면 애플리케이션을 작성하거나 리소스를 관리할 필요가 없습니다. 데이터 생산자가 데이터를 Firehose로 보내도록 구성하면 지정한 대상으로 데이터를 자동 전송합니다. 전송 전에 데이터를 변환하도록 Firehose를 구성할 수도 있습니다.
- Amazon Managed Service for Apache Flink는 Apache Flink 애플리케이션을 사용하여 스트리밍 데이터를 처리할 수 있는 완전 관리형 Amazon 서비스입니다. Iceberg 테이블에서 읽고 쓸 수 있으며 실시간 데이터 처리 및 분석을 지원합니다.
- Amazon SageMaker AI는 Iceberg 형식을 사용하여 Amazon SageMaker AI 특성 저장소에 특성 세트를 저장할 수 있도록 지원합니다.

- AWS Lake Formation는 Athena 또는 Amazon Redshift에서 사용하는 Iceberg 테이블을 포함하여 데이터에 액세스할 수 있는 거칠고 세분화된 액세스 제어 권한을 제공합니다. Iceberg 테이블의 권한 지원에 대한 자세한 내용은 Lake Formation 설명서를 참조하세요.

AWS 에는 Iceberg를 지원하는 다양한 서비스가 있지만 이러한 모든 서비스를 다루는 것은이 가이드의 범위를 벗어납니다. 다음 섹션에서는 Amazon EMR 및 Athena SQL의 Spark(배치 및 구조화된 스트리밍) AWS Glue에 대해 설명합니다. 다음 섹션에서는 Athena SQL의 Iceberg 지원을 간략하게 살펴봅니다.

Amazon Athena SQL에서 Iceberg 테이블 시작하기

Amazon Athena는 Iceberg에 대한 기본 지원을 제공합니다. Athena 설명서의 시작하기 섹션에 설명된 서비스 사전 조건을 설정하는 경우를 제외하고 추가 단계 또는 구성 없이 Iceberg를 사용할 수 있습니다. 이 섹션에서는 Athena에서 테이블을 생성하는 방법을 간략하게 소개합니다. 자세한 내용은이 안내서 뒷부분의 Athena SQL을 사용하여 Iceberg 테이블 작업을 참조하세요.

다른 엔진을 사용하여에서 Iceberg 테이블 AWS 을 생성할 수 있습니다. 이러한 테이블은 원활하게 작동합니다 AWS 서비스. Athena SQL을 사용하여 첫 번째 Iceberg 테이블을 생성하려면 다음 표준 문안 코드를 사용할 수 있습니다.

```
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'
)
```

다음 섹션에서는 Athena에서 파티셔닝된 Iceberg 테이블과 파티셔닝되지 않은 Iceberg 테이블을 생성하는 예를 제공합니다. 자세한 내용은 Athena 설명서에 자세히 설명된 Iceberg 구문을 참조하세요.

파티셔닝되지 않은 테이블 생성

다음 예제 문은 Athena에서 분할되지 않은 Iceberg 테이블을 생성하도록 표준 문 SQL 코드를 사용자 지정합니다. Athena 콘솔의 쿼리 편집기에이 문을 추가하여 테이블을 생성할 수 있습니다.

```
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'
)
```

쿼리 편집기 사용에 대한 step-by-step 지침은 Athena 설명서의 시작하기를 참조하세요.

파티셔닝된 테이블 생성

다음 문은 Iceberg의 숨겨진 파티셔닝 개념을 사용하여 날짜를 기반으로 파티셔닝된 테이블을 생성합니다. `day()` 변환을 사용하여 타임스탬프 열에서 `dd-mm-yyyy` 형식을 사용하여 일일 파티션을 추출합니다. Iceberg는이 값을 데이터 세트의 새 열로 저장하지 않습니다. 대신 데이터를 작성하거나 쿼리할 때 값이 즉시 파생됩니다.

```
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'
)
```

단일 CTAS 문을 사용하여 테이블 생성 및 데이터 로드

이전 섹션의 분할된 예제와 분할되지 않은 예제에서 Iceberg 테이블은 빈 테이블로 생성됩니다. `INSERT` 또는 `MERGE` 문을 사용하여 테이블에 데이터를 로드할 수 있습니다. 또는 `CREATE TABLE AS SELECT (CTAS)` 문을 사용하여 한 번에 Iceberg 테이블로 데이터를 생성하고 로드할 수 있습니다.

CTAS는 Athena에서 테이블을 생성하고 단일 문에 데이터를 로드하는 가장 좋은 방법입니다. 다음 예제에서는 CTAS를 사용하여 Athena의 기존 Hive/Parquet 테이블(`iceberg_ctas_table`)에서 Iceberg 테이블(`hive_table`)을 생성하는 방법을 보여줍니다.

```
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
```

CTAS에 대한 자세한 내용은 Athena CTAS 설명서를 참조하세요.

데이터 삽입, 업데이트 및 삭제

Athena는 INSERT INTO, UPDATE, MERGE INTO및 DELETE FROM 문을 사용하여 Iceberg 테이블에 데이터를 쓰는 다양한 방법을 지원합니다.

> **Note**
>
> Athena SQL은 현재 copy-on-write 접근 방식을 지원하지 않습니다. UPDATE, MERGE INTO및 DELETE FROM 작업은 지정된 테이블 속성에 관계없이 위치 삭제와 함께 merge-on-read 접근 방식을 항상 사용합니다. copy-on-writewrite.delete.mode를 사용하도록 write.merge.mode, 및 write.update.mode와 같은 테이블 속성을 설정하면 쿼리가 실패하지 않지만 Athena는 쿼리를 무시하고 merge-on-read을 계속 사용합니다.

다음 문은 INSERT INTO를 사용하여 Iceberg 테이블에 데이터를 추가합니다.

```
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;
```

샘플 출력:

자세한 내용은 Athena 설명서를 참조하세요.

Iceberg 테이블 쿼리

이전 예제와 같이 Athena SQL을 사용하여 Iceberg 테이블에 대해 일반 SQL 쿼리를 실행할 수 있습니다.

Athena는 일반적인 쿼리 외에도 Iceberg 테이블에 대한 시간 이동 쿼리도 지원합니다. 앞서 설명한 것처럼 Iceberg 테이블의 업데이트 또는 삭제를 통해 기존 레코드를 변경할 수 있으므로 시간 이동 쿼리를 사용하여 타임스탬프 또는 스냅샷 ID를 기반으로 테이블의 이전 버전을 다시 살펴보는 것이 편리합니다.

예를 들어 다음 문은의 색상 값을 업데이트한 다음 `Person5`2023년 1월 4일부터 이전 값을 표시합니다.

```
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'
```

샘플 출력:

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

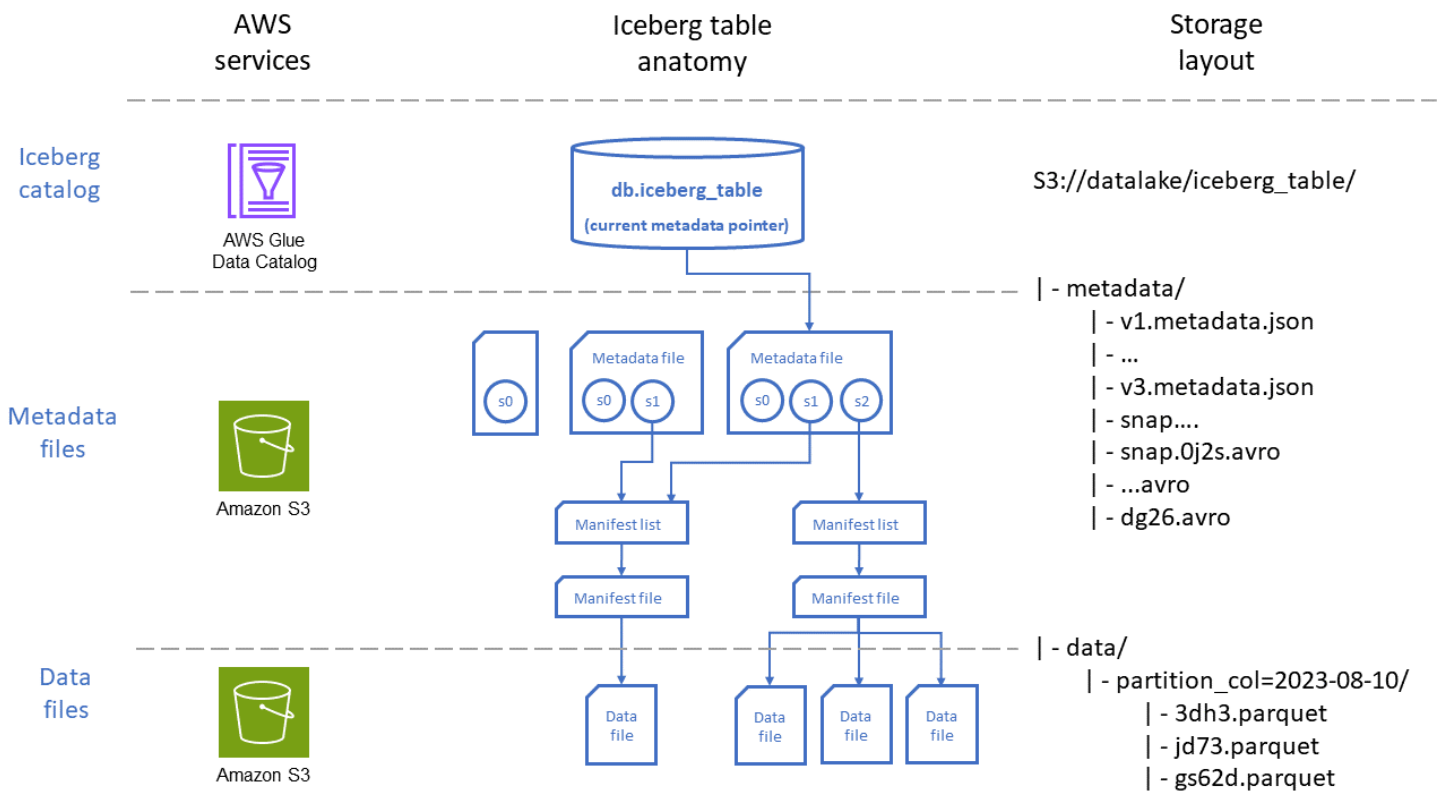
시간 이동 쿼리의 구문과 추가 예제는 Athena 설명서를 참조하세요.

Iceberg 테이블 구조

이제 Iceberg 테이블 작업의 기본 단계를 살펴보았으므로 Iceberg 테이블의 복잡한 세부 정보와 설계에 대해 자세히 살펴보겠습니다.

이 가이드의 앞부분에서 설명한 기능을 활성화하기 위해 Iceberg는 데이터 및 메타데이터 파일의 계층적 계층으로 설계되었습니다. 이러한 계층은 메타데이터를 지능적으로 관리하여 쿼리 계획 및 실행을 최적화합니다.

다음 다이어그램은 테이블을 저장하는 데 AWS 서비스 사용되는와 Amazon S3에 파일 배치라는 두 가지 관점을 통해 Iceberg 테이블의 구성을 보여줍니다.



다이어그램과 같이 Iceberg 테이블은 세 가지 기본 계층으로 구성됩니다.

- Iceberg 카탈로그: AWS Glue Data Catalog integrate는 기본적으로 Iceberg와 통합되며 대부분의 사용 사례에서 실행되는 워크로드에 가장 적합한 옵션입니다 AWS. Iceberg 테이블과 상호 작용하는 서비스(예: Athena)는 카탈로그를 사용하여 데이터를 읽거나 쓸 테이블의 현재 스냅샷 버전을 찾습니다.
- 메타데이터 계층: 메타데이터 파일, 즉 매니페스트 파일 및 매니페스트 목록 파일은 테이블의 스키마, 파티션 전략, 데이터 파일의 위치와 같은 정보와 각 데이터 파일에 저장된 레코드의 최소 및 최대 범위와 같은 열 수준 통계를 추적합니다. 이러한 메타데이터 파일은 테이블 경로 내의 Amazon S3에 저장됩니다.
  - 매니페스트 파일에는 위치, 형식, 크기, 체크섬 및 기타 관련 정보를 포함하여 각 데이터 파일에 대한 레코드가 포함됩니다.
  - 매니페스트 목록은 매니페스트 파일의 인덱스를 제공합니다. 테이블에서 매니페스트 파일 수가 증가함에 따라 해당 정보를 더 작은 하위 섹션으로 나누면 쿼리로 스캔해야 하는 매니페스트 파일 수를 줄이는 데 도움이 됩니다.
  - 메타데이터 파일에는 매니페스트 목록, 스키마, 파티션 메타데이터, 스냅샷 파일 및 테이블의 메타데이터를 관리하는 데 사용되는 기타 파일을 포함하여 전체 Iceberg 테이블에 대한 정보가 포함됩니다.

- 데이터 계층:이 계층에는 쿼리가 실행될 데이터 레코드가 있는 파일이 포함됩니다. 이러한 파일은 Apache Parquet, Apache Avro, Apache ORC 등 다양한 형식으로 저장할 수 있습니다.
  - 데이터 파일에는 테이블에 대한 데이터 레코드가 포함됩니다.
  - Iceberg 테이블에서 행 수준 삭제 및 업데이트 작업을 인코딩하는 파일을 삭제합니다. Iceberg 설명서에 설명된 대로 Iceberg에는 두 가지 유형의 삭제 파일이 있습니다. 이러한 파일은 merge-on-read 모드를 사용하여 작업에 의해 생성됩니다.

Amazon EMR에서 Iceberg 작업

Amazon EMR은 Apache Spark, Apache Hive, Flink, Trino와 같은 오픈 소스 프레임워크를 사용하여 클라우드에서 페타바이트 규모의 데이터 처리, 대화형 분석 및 기계 학습을 제공합니다.

> **Note**
>
> 이 가이드에서는 Apache Spark를 예제로 사용합니다.

Amazon EMR은 Amazon EMR on EC2, Amazon EMR on EKS, Amazon EMR Serverless, Amazon EMR on 등 여러 배포 옵션을 지원합니다 AWS Outposts. 워크로드에 대한 배포 옵션을 선택하려면 Amazon EMR FAQ를 참조하세요.

버전 및 기능 호환성

Amazon EMR 버전 6.5.0 이상은 기본적으로 Apache Iceberg를 지원합니다. 각 Amazon EMR 릴리스에 대해 지원되는 Iceberg 버전 목록은 Amazon EMR 설명서의 Iceberg 릴리스 기록을 참조하세요. 또한 다양한 프레임워크의 Amazon EMR에서 지원되는 Iceberg 기능을 확인하려면 Iceberg에서 클러스터 사용 섹션을 검토하세요.

지원되는 최신 Iceberg 버전을 활용하려면 최신 Amazon EMR 버전을 사용하는 것이 좋습니다. 이 섹션의 코드 예제 및 구성에서는 Amazon EMR 릴리스 emr-7.8.0을 사용한다고 가정합니다.

Iceberg를 사용하여 Amazon EMR 클러스터 생성

Iceberg가 설치된 Amazon EC2에서 Amazon EMR 클러스터를 생성하려면 Amazon EMR 설명서의 지침을 따르세요.

특히 클러스터는 다음 분류로 구성되어야 합니다.

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

Amazon EMR 6.6.0부터 Amazon EMR Serverless 또는 Amazon EMR on EKS를 Iceberg 워크로드의 배포 옵션으로 사용하도록 선택할 수도 있습니다.

Amazon EMR에서 Iceberg 애플리케이션 개발

Iceberg 애플리케이션을 위한 Spark 코드를 개발하려면 Amazon EMR 클러스터에서 실행되는 완전 관리형 Jupyter 노트북을 위한 웹 기반 통합 개발 환경(IDE)인 Amazon EMR Studio를 사용할 수 있습니다.

Amazon EMR Studio 노트북 사용

Amazon EMR Studio Workspace 노트북에서 Spark 애플리케이션을 대화형으로 개발하고 해당 노트북을 Amazon EMR on EC2 클러스터 또는 Amazon EMR on EKS 관리형 엔드포인트에 연결할 수 있습니다. Amazon EMR on EC2 및 Amazon EMR on EKS용 EMR Studio 설정에 대한 지침은 AWS 서비스 설명서를 참조하세요. https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-studio-create-eks-cluster.html

EMR Studio에서 Iceberg를 사용하려면 다음 단계를 따르세요.

1. Iceberg가 설치된 클러스터 사용의 지침에 따라 Iceberg가 활성화된 Amazon EMR 클러스터를 시작합니다.
2. EMR Studio를 설정합니다. 지침은 Amazon EMR Studio 설정을 참조하세요.
3. EMR Studio Workspace 노트북을 열고 노트북의 첫 번째 셀로 다음 코드를 실행하여 Iceberg를 사용하도록 Spark 세션을 구성합니다.

```
%%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"
    }
}
```

여기서 각 항목은 다음과 같습니다.

- `<catalog_name>`는 Iceberg Spark 세션 카탈로그 이름입니다. 선택한 이름으로 바꾸고이 카탈로그와 연결된 모든 구성에서 참조를 변경해야 합니다. 코드에서 다음과 같이 Spark 세션 카탈로그 이름을 포함하여 정규화된 테이블 이름을 사용하여 Iceberg 테이블을 참조할 수 있습니다.

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

  또는를 카탈로그 이름으로 설정하여 기본 카탈로그를 정의한 Iceberg 카탈로그`spark.sql.defaultCatalog`로 변경할 수 있습니다. 이 두 번째 접근 방식을 사용하면 카탈로그 접두사가 없는 테이블을 참조할 수 있으므로 쿼리를 간소화할 수 있습니다.

- `<catalog_name>.warehouse`는 데이터와 메타데이터를 저장할 Amazon S3 경로를 가리킵니다.
- 카탈로그를 로 만들려면를 `spark.sql.catalog.<catalog_name>.type`로 AWS Glue Data Catalog설정합니다`glue`. 이 키는 사용자 지정 카탈로그 구현을 위한 구현 클래스를 가리키는 데 필요합니다. 이 가이드의 뒷부분에 있는 일반 모범 사례 섹션에서는 다양한 Iceberg 지원 카탈로그에 대해 설명합니다.

4. 이제 다른 Spark 애플리케이션과 마찬가지로 노트북에서 Iceberg용 Spark 애플리케이션을 대화형으로 개발할 수 있습니다.

Amazon EMR Studio를 사용하여 Apache Iceberg용 Spark를 구성하는 방법에 대한 자세한 내용은 블로그 게시물 Amazon EMR에서 Apache Iceberg를 사용하여 진화하는 고성능 ACID 호환 데이터 레이크 구축을 참조하세요.

Amazon EMR에서 Iceberg 작업 실행

Iceberg 워크로드에 대한 Spark 애플리케이션 코드를 개발한 후 Iceberg를 지원하는 모든 Amazon EMR 배포 옵션에서 실행할 수 있습니다(Amazon EMR FAQ 참조).

다른 Spark 작업과 마찬가지로 단계를 추가하거나 Spark 작업을 마스터 노드에 대화형으로 제출하여 Amazon EMR on EC2 클러스터에 작업을 제출할 수 있습니다. Spark 작업을 실행하려면 다음 Amazon EMR 설명서 페이지를 참조하세요.

- Amazon EMR on EC2 클러스터에 작업을 제출하는 다양한 옵션에 대한 개요와 각 옵션에 대한 자세한 지침은 클러스터에 작업 제출을 참조하세요.
- Amazon EMR on EKS의 경우 StartJobRun을 사용하여 Spark 작업 실행을 참조하세요.
- EMR Serverless의 경우 작업 실행을 참조하세요.

다음 섹션에서는 각 Amazon EMR 배포 옵션의 예를 제공합니다.

Amazon EMR on EC2

다음 단계를 사용하여 Iceberg Spark 작업을 제출할 수 있습니다.

1. 워크스테이션에 다음 콘텐츠가 `emr_step_iceberg.json` 포함된 파일을 생성합니다.

```
[{
    "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. 굵게 강조 표시된 Iceberg 구성 옵션을 사용자 지정하여 특정 Spark 작업의 구성 파일을 수정합니다.
3. AWS Command Line Interface ()를 사용하여 단계를 제출합니다AWS CLI. `emr_step_iceberg.json` 파일이 있는 디렉터리에서 명령을 실행합니다.

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

Amazon EMR Serverless

를 사용하여 EMR Serverless에 Iceberg Spark 작업을 제출하려면 AWS CLI:

1. 워크스테이션에 다음 콘텐츠`emr_serverless_iceberg.json`가 포함된 파일을 생성합니다.

```
{
    "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. 굵게 강조 표시된 Iceberg 구성 옵션을 사용자 지정하여 특정 Spark 작업의 구성 파일을 수정합니다.

3. 를 사용하여 작업을 제출합니다 AWS CLI. `emr_serverless_iceberg.json` 파일이 있는 디렉터리에서 명령을 실행합니다.

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

EMR Studio 콘솔을 사용하여 EMR Serverless에 Iceberg Spark 작업을 제출하려면:

1. EMR Serverless 설명서의 지침을 따릅니다.
2. 작업 구성의 경우에 제공된 Spark에 대한 Iceberg 구성을 사용하고 Iceberg에 대해 강조 표시된 필드를 AWS CLI 사용자 지정합니다. 자세한 지침은 Amazon EMR 설명서의 EMR Serverless에서 Apache Iceberg 사용을 참조하세요.

Amazon EMR on EKS

AWS CLI다음을 사용하여 Amazon EMR on EKS에 Iceberg Spark 작업을 제출하려면

1. 워크스테이션에 다음 콘텐츠가 `emr_eks_iceberg.json`포함된 파일을 생성합니다.

```
{
    "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. 굵게 강조 표시된 Iceberg 구성 옵션을 사용자 지정하여 Spark 작업의 구성 파일을 수정합니다.
3. 를 사용하여 작업을 제출합니다 AWS CLI. `emr_eks_iceberg.json` 파일이 있는 디렉터리에서 다음 명령을 실행합니다.

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

자세한 지침은 Amazon EMR on EKS 설명서의 Amazon EMR on EKS에서 Apache Iceberg 사용을 참조하세요.

Amazon EMR 모범 사례

이 섹션에서는 Iceberg 테이블에 대한 데이터 읽기 및 쓰기를 최적화하기 위해 Amazon EMR에서 Spark 작업을 튜닝하기 위한 일반 지침을 제공합니다. Iceberg 관련 모범 사례는이 가이드 뒷부분의 모범 사례 섹션을 참조하세요.

- 최신 버전의 Amazon EMR 사용 - Amazon EMR은 Amazon EMR Spark 런타임을 통해 즉시 Spark 최적화를 제공합니다. AWS 는 새 릴리스마다 Spark 런타임 엔진의 성능을 개선합니다.
- Spark 워크로드에 대한 최적의 인프라 결정 - Spark 워크로드는 최적의 성능을 보장하기 위해 다양한 작업 특성에 대해 다양한 유형의 하드웨어가 필요할 수 있습니다. Amazon EMR은 다양한 인스턴스 유형(예: 컴퓨팅 최적화, 메모리 최적화, 범용 및 스토리지 최적화)을 지원하여 모든 유형의 처리 요구 사항을 충족합니다. 새 워크로드를 온보딩할 때는 M5 또는 M6g와 같은 일반 인스턴스 유형으로 벤치마킹하는 것이 좋습니다. Ganglia 및 Amazon CloudWatch의 운영 체제(OS) 및 YARN 지표를 모니터링하여 최대 부하 시 시스템 병목 현상(CPU, 메모리, 스토리지 및 I/O)을 확인하고 적절한 하드웨어를 선택합니다.
- 튜닝 `spark.sql.shuffle.partitions` - `spark.sql.shuffle.partitions` 속성을 클러스터의 총 가상 코어(vCores) 수 또는 해당 값의 배수(일반적으로 총 vCores 수의 1~2배)로 설정합니다. 이 설정은 해시 및 범위 파티셔닝을 쓰기 배포 모드로 사용할 때 Spark의 병렬 처리에 영향을 줍니다. 데이터를 구성하기 위해 쓰기 전에 셔플을 요청하므로 파티션 정렬이 보장됩니다.

- 관리형 조정 활성화 - 거의 모든 사용 사례에서 관리형 조정 및 동적 할당을 활성화하는 것이 좋습니다. 그러나 예측 가능한 패턴이 있는 워크로드가 있는 경우 자동 조정 및 동적 할당을 비활성화하는 것이 좋습니다. 관리형 조정이 활성화된 경우 스팟 인스턴스를 사용하여 비용을 줄이는 것이 좋습니다. 코어 또는 마스터 노드 대신 태스크 노드에 스팟 인스턴스를 사용합니다. 스팟 인스턴스를 사용하는 경우 플릿당 여러 인스턴스 유형이 있는 인스턴스 플릿을 사용하여 스팟 가용성을 보장합니다.
- 가능한 경우 브로드캐스트 조인 사용 - 브로드캐스트(맵사이드) 조인이 가장 최적의 조인입니다. 단, 테이블 중 하나가 가장 작은 노드의 메모리(MBs)에 맞을 만큼 작고 등식(=) 조인을 수행하는 경우에 한합니다. 전체 외부 조인을 제외한 모든 조인 유형이 지원됩니다. 브로드캐스트 조인은 메모리의 모든 작업자 노드에서 더 작은 테이블을 해시 테이블로 브로드캐스트합니다. 작은 테이블이 브로드캐스트된 후에는 변경할 수 없습니다. 해시 테이블은 Java 가상 머신(JVM)에 로컬로 있으므로 해시 조인을 사용하여 조인 조건에 따라 큰 테이블과 쉽게 병합할 수 있습니다. 브로드캐스트 조인은 셔플 오버헤드를 최소화하므로 고성능을 제공합니다.
- 가비지 수집기 튜닝 - 가비지 수집(GC) 주기가 느린 경우 성능 향상을 위해 기본 병렬 가비지 수집기에서 G1GC로 전환하는 것이 좋습니다. GC 성능을 최적화하기 위해 GC 파라미터를 미세 조정할 수 있습니다. GC 성능을 추적하려면 Spark UI를 사용하여 모니터링할 수 있습니다. GC 시간은 총 작업 런타임의 1% 이하여야 합니다.

에서 Iceberg 작업 AWS Glue

AWS Glue는 분석, 기계 학습(ML) 및 애플리케이션 개발을 위해 여러 소스에서 데이터를 더 쉽게 검색, 준비, 이동 및 통합할 수 있는 서버리스 데이터 통합 서비스입니다. 의 핵심 기능 중 하나는 간단하고 비용 효율적인 방식으로 추출, 변환 및 로드(ETL) 작업을 수행하는 기능 AWS Glue 입니다. 이렇게 하면 데이터를 분류하고, 정리하고, 보강하고, 다양한 데이터 스토어와 데이터 스트림 간에 안정적으로 이동할 수 있습니다.

AWS Glue 작업은 Apache Sparkor Python runtime. AWS Glue jobs를 사용하여 변환 로직을 정의하는 스크립트를 캡슐화합니다.

에서 Iceberg 작업을 생성할 때 버전에 AWS Glue따라 AWS Glue네이티브 Iceberg 통합 또는 사용자 지정 Iceberg 버전을 사용하여 Iceberg 종속성을 작업에 연결할 수 있습니다.

네이티브 Iceberg 통합 사용

AWS Glue 버전 3.0, 4.0 및 5.0은 기본적으로의 Apache Iceberg, Apache Hudi 및 Linux Foundation Delta Lake AWS Glue for Spark와 같은 트랜잭션 데이터 레이크 형식을 지원합니다. 이 통합 기능은에서 이러한 프레임워크 사용을 시작하는 데 필요한 구성 단계를 간소화합니다 AWS Glue.

AWS Glue 작업에 대한 Iceberg 지원을 활성화하려면 작업을 설정합니다. 작업에 대한 작업 세부 정보 탭을 선택하고 고급 속성에서 작업 파라미터로 AWS Glue 스크롤한 다음 키를 `--datalake-formats` 로, 해당 값을 로 설정합니다`iceberg`.

노트북을 사용하여 작업을 작성하는 경우 다음과 같이 `%%configure` 매직을 사용하여 첫 번째 노트북 셀에서 파라미터를 구성할 수 있습니다.

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

`--datalake-formats`의 `iceberg` AWS Glue 구성은 버전에 따라 특정 Iceberg AWS Glue 버전에 해당합니다.

AWS Glue 버전	기본 Iceberg 버전
5.0	1.7.1

AWS Glue 버전	기본 Iceberg 버전
4.0	1.0.0
3.0	0.13.1

사용자 지정 Iceberg 버전 사용

경우에 따라 작업에 대한 Iceberg 버전을 제어하고 원하는 속도로 업그레이드해야 할 수 있습니다. 예를 들어, 최신 버전으로 업그레이드하면 새로운 기능 및 성능 향상에 대한 액세스가 잠금 해제될 수 있습니다. 에서 특정 Iceberg 버전을 사용하려면 자체 JAR 파일을 제공할 AWS Glue수 있습니다.

사용자 지정 Iceberg 버전을 구현하기 전에 설명서의 AWS Glue 버전 섹션을 확인하여 환경과의 AWS Glue 호환성을 AWS Glue 확인합니다. 예를 들어, AWS Glue 5.0은 Spark 3.5.4와의 호환성이 필요합니다.

예를 들어 Iceberg 버전 1.9.1을 사용하는 AWS Glue 작업을 실행하려면 다음 단계를 따릅니다.

1. 필요한 JAR 파일을 획득하여 Amazon S3에 업로드합니다.
   a. Apache Maven 리포지토리에서 iceberg-spark-runtime-3.5_2.12-1.9.1.jar 및 iceberg-aws-bundle-1.9.1.jar을 다운로드합니다.
   b. 이러한 파일을 지정된 S3 버킷 위치(예: `s3://your-bucket-name/jars/`)에 업로드합니다.
2. 다음과 같이 작업에 대한 AWS Glue 작업 파라미터를 설정합니다.
   a. `--extra-jars` 파라미터의 두 JAR 파일에 대한 전체 S3 경로를 쉼표로 구분하여 지정합니다(예: `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. `--datalake-formats` 파라미터의 값으로 `iceberg`를 포함하지 마세요.
   c. AWS Glue 5.0을 사용하는 경우 `--user-jars-first` 파라미터를 로 설정해야 합니다`true`.

의 Iceberg에 대한 Spark 구성 AWS Glue

이 섹션에서는 Iceberg 데이터 세트에 대한 AWS Glue ETL 작업을 작성하는 데 필요한 Spark 구성에 대해 설명합니다. 모든 `--conf` Spark 구성 키 및 값의 쉼표로 구분된 목록과 함께 Spark 키를 사용하여 이러한 구성을 설정할 수 있습니다. 노트북에서 `%%configure` 매직을 사용하거나 AWS Glue Studio 콘솔의 작업 파라미터 섹션을 사용할 수 있습니다.

```
%glue_version 5.0

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

다음 속성을 사용하여 Spark 세션을 구성합니다.

- <catalog_name>는 Iceberg Spark 세션 카탈로그 이름의 이름입니다. 선택한 이름으로 바꾸고이 카탈로그와 연결된 모든 구성에서 참조를 변경해야 합니다. 코드에서 다음과 같이 Spark 세션 카탈로그 이름을 포함하여 정규화된 테이블 이름을 사용하여 Iceberg 테이블을 참조할 수 있습니다.

  <catalog_name>.<database_name>.<table_name>

  또는를 카탈로그 이름으로 설정하여 기본 카탈로그를 정의한 Iceberg 카탈로그spark.sql.defaultCatalog로 변경할 수 있습니다. 이 두 번째 접근 방식을 사용하여 카탈로그 접두사가 없는 테이블을 참조할 수 있으므로 쿼리를 간소화할 수 있습니다.
- <catalog_name>.<warehouse>는 데이터와 메타데이터를 저장할 Amazon S3 경로를 가리킵니다.
- 카탈로그를 로 만들려면를 spark.sql.catalog.<catalog_name>.type로 AWS Glue Data Catalog설정합니다glue. 이 키는 사용자 지정 카탈로그 구현을 위한 구현 클래스를 가리키는 데 필요합니다. Iceberg에서 지원하는 카탈로그는이 가이드 뒷부분의 일반 모범 사례 섹션을 참조하세요.

예를 들어 라는 카탈로그가 있는 경우 다음과 같이 여러 --conf 키를 사용하여 작업을 구성할 glue_iceberg수 있습니다.

```
%%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"
}
```

또는 코드를 사용하여 다음과 같이 Spark 스크립트에 위의 구성을 추가할 수 있습니다.

```
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 작업 모범 사례

이 섹션에서는 Iceberg 테이블에 대한 데이터 읽기 및 쓰기를 최적화 AWS Glue 하기 위해에서 Spark 작업을 튜닝하기 위한 일반 지침을 제공합니다. Iceberg 관련 모범 사례는이 가이드 뒷부분의 모범 사례 섹션을 참조하세요.

- 가능한 경우 최신 버전의 AWS Glue 및 업그레이드 사용 -의 새 버전은 성능 개선, 시작 시간 단축 및 새 기능을 AWS Glue 제공합니다. 또한 최신 Iceberg 버전에 필요할 수 있는 최신 Spark 버전을 지원합니다. 사용 가능한 AWS Glue 버전 및 지원하는 Spark 버전 목록은 AWS Glue 설명서를 참조하세요.
- AWS Glue 작업 메모리 최적화 - AWS 블로그 게시물의 메모리 관리 최적화의 AWS Glue권장 사항을 따릅니다.
- AWS Glue Auto Scaling 사용 - Auto Scaling을 활성화하면가 워크로드에 따라 동적으로 AWS Glue 작업자 수를 AWS Glue 자동으로 조정합니다. 이렇게 하면 워크로드가 작고 작업자가 유휴 상태일 때가 작업자 수를 스케일 다운하므로 AWS Glue 최대 부하 AWS Glue 중에 작업 비용을 줄일 수 있습니다. AWS Glue Auto Scaling을 사용하려면 작업을 확장할 수 있는 최대 작업자 수를 AWS Glue 지정합니다. 자세한 내용은 설명서의 에 Auto Scaling 사용을 AWS Glue 참조하세요 AWS Glue .
- 원하는 Iceberg 버전 사용 - Iceberg의 AWS Glue 기본 통합은 Iceberg를 시작하는 데 가장 좋습니다. 그러나 프로덕션 워크로드의 경우 Iceberg 버전을 완전히 제어하려면 라이브러리 종속성(이 가이드의 앞부분에서 설명한 대로)을 추가하는 것이 좋습니다. 이 접근 방식은 작업의 최신 Iceberg 기능 및 성능 개선을 활용하는 데 도움이 됩니다 AWS Glue .
- 모니터링 및 디버깅을 위한 Spark UI 활성화 - 의 Spark UI AWS Glue를 사용하여 방향성 비순환 그래프(DAG)에서 Spark 작업의 다양한 단계를 시각화하고 작업을 세부적으로 모니터링하여 Iceberg 작업을 검사할 수도 있습니다. Spark UI는 Iceberg 작업의 문제를 해결하고 최적화하는 효과적인 방법을 제공합니다. 예를 들어, 큰 셔플 또는 디스크 유출이 있는 병목 현상 단계를 식별하여 튜닝 기회를 식별할 수 있습니다. 자세한 내용은 설명서의 Apache Spark 웹 UI를 사용하여 작업 모니터링을 참조하세요 AWS Glue .

Apache Spark를 사용하여 Iceberg 테이블 작업

이 섹션에서는 Apache Spark를 사용하여 Iceberg 테이블과 상호 작용하는 방법에 대한 개요를 제공합니다. 이 예제는 Amazon EMR 또는에서 실행할 수 있는 표준 문안 코드입니다 AWS Glue.

참고: Iceberg 테이블과 상호 작용하기 위한 기본 인터페이스는 SQL이므로 대부분의 예제는 Spark SQL을 DataFrames API와 결합합니다.

Iceberg 테이블 생성 및 작성

Spark SQL 및 Spark DataFrames를 사용하여 Iceberg 테이블에 데이터를 생성하고 추가할 수 있습니다.

Spark SQL 사용

Iceberg 데이터 세트를 작성하려면 `CREATE TABLE` 및와 같은 표준 Spark SQL 문을 사용합니다`INSERT INTO`.

파티셔닝되지 않은 테이블

다음은 Spark SQL을 사용하여 파티셔닝되지 않은 Iceberg 테이블을 생성하는 예입니다.

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

파티셔닝되지 않은 테이블에 데이터를 삽입하려면 표준 `INSERT INTO` 문을 사용합니다.

```
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
""")
```

분할된 테이블

다음은 Spark SQL을 사용하여 파티셔닝된 Iceberg 테이블을 생성하는 예입니다.

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

Spark SQL을 사용하여 파티셔닝된 Iceberg 테이블에 데이터를 삽입하려면 표준 `INSERT INTO` 문을 사용합니다.

```
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**
>
> Iceberg 1.5.0부터는 분할된 테이블에 데이터를 삽입할 때 hash 쓰기 배포 모드가 기본값입니다. 자세한 내용은 Iceberg 설명서의 배포 모드 작성을 참조하세요.

DataFrames API 사용

Iceberg 데이터 세트를 작성하려면 `DataFrameWriterV2` API를 사용할 수 있습니다.

Iceberg 테이블을 생성하고 여기에 데이터를 쓰려면 df.writeTo(t) 함수를 사용합니다. 테이블이 있는 경우 .append() 함수를 사용합니다. 그렇지 않은 경우를 사용합니다. 다음 예제.create().에서는와 .create() 동일한 변형.createOrReplace()인를 사용합니다CREATE OR REPLACE TABLE AS SELECT.

파티셔닝되지 않은 테이블

DataFrameWriterV2 API를 사용하여 파티셔닝되지 않은 Iceberg 테이블을 생성하고 채우려면:

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

DataFrameWriterV2 API를 사용하여 파티셔닝되지 않은 기존 Iceberg 테이블에 데이터를 삽입하려면:

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

분할된 테이블

DataFrameWriterV2 API를 사용하여 파티셔닝된 Iceberg 테이블을 생성하고 채우려면:

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

DataFrameWriterV2 API를 사용하여 파티셔닝된 Iceberg 테이블에 데이터를 삽입하려면:

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

Iceberg 테이블의 데이터 업데이트

다음 예제에서는 Iceberg 테이블의 데이터를 업데이트하는 방법을 보여줍니다. 이 예제에서는 c_customer_sk 열에 짝수가 있는 모든 행을 수정합니다.

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

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

이 작업은 기본 copy-on-write 전략을 사용하므로 영향을 받는 모든 데이터 파일을 다시 씁니다.

Iceberg 테이블의 데이터 업서스팅

데이터 업서팅은 단일 트랜잭션에서 새 데이터 레코드를 삽입하고 기존 데이터 레코드를 업데이트하는 것을 의미합니다. Iceberg 테이블로 데이터를 업서트하려면 SQL MERGE INTO 문을 사용합니다.

다음 예제에서는 테이블 내 테이블 {UPSERT_TABLE_NAME}의 콘텐츠를 업서트합니다. {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 *
""")
```

- 에 있는 고객 레코드가 {UPSERT_TABLE_NAME} 이미 동일한 {TABLE_NAME}로에 있는 경우 c_customer_id {UPSERT_TABLE_NAME} 레코드 c_email_address 값은 기존 값(업데이트 작업)을 재정의합니다.
- 에 있는 고객 레코드가에 {UPSERT_TABLE_NAME} 없는 경우 {TABLE_NAME} {UPSERT_TABLE_NAME} 레코드가 {TABLE_NAME} (작업 삽입)에 추가됩니다.

Iceberg 테이블에서 데이터 삭제

Iceberg 테이블에서 데이터를 삭제하려면 DELETE FROM 표현식을 사용하고 삭제할 행과 일치하는 필터를 지정합니다.

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

필터가 전체 파티션과 일치하는 경우 Iceberg는 메타데이터 전용 삭제를 수행하고 데이터 파일을 그대로 둡니다. 그렇지 않으면 영향을 받는 데이터 파일만 다시 씁니다.

삭제 메서드는 WHERE 절의 영향을 받는 데이터 파일을 가져와 삭제된 레코드 없이 복사본을 생성합니다. 그런 다음 새 데이터 파일을 가리키는 새 테이블 스냅샷을 생성합니다. 따라서 삭제된 레코드는 테이블의 이전 스냅샷에 여전히 존재합니다. 예를 들어 테이블의 이전 스냅샷을 검색하면 방금 삭제한 데이터가 표시됩니다. 정리를 위해 관련 데이터 파일을 사용하여 불필요한 이전 스냅샷을 제거하는 방법에 대한 자세한 내용은이 가이드 뒷부분의 압축을 사용하여 파일 유지 관리를 참조하세요.

데이터 읽기

Spark SQL 및 DataFrames를 사용하여 Spark에서 Iceberg 테이블의 최신 상태를 읽을 수 있습니다. DataFrames

Spark SQL 사용 예제:

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

DataFrames API 사용 예제:

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

시간 이동 사용

Iceberg 테이블의 각 쓰기 작업(삽입, 업데이트, 업서트, 삭제)은 새 스냅샷을 생성합니다. 그런 다음 이러한 스냅샷을 시간 이동에 사용하여 과거로 돌아가서 과거의 테이블 상태를 확인할 수 있습니다.

snapshot-id 및 타이밍 값을 사용하여 테이블의 스냅샷 기록을 검색하는 방법에 대한 자세한 내용은 이 가이드 뒷부분의 메타데이터 액세스 섹션을 참조하세요.

다음 시간 이동 쿼리는 특정를 기반으로 테이블의 상태를 표시합니다 snapshot-id.

Spark SQL 사용:

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

DataFrames API 사용:

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

다음 시간 이동 쿼리는 특정 타임스탬프 이전에 생성된 마지막 스냅샷을 기반으로 테이블의 상태를 밀리초() 단위로 표시합니다`as-of-timestamp`.

Spark SQL 사용:

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

DataFrames API 사용:

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

증분 쿼리 사용

Iceberg 스냅샷을 사용하여 추가된 데이터를 점진적으로 읽을 수도 있습니다.

참고: 현재이 작업은 append 스냅샷에서 데이터 읽기를 지원합니다. `replace`, `overwrite`또는와 같은 작업에서 데이터 가져오기를 지원하지 않습니다`delete`. 또한 Spark SQL 구문에서는 증분 읽기 작업이 지원되지 않습니다.

다음 예시에서는 스냅샷`start-snapshot-id`(제외)과 `end-snapshot-id` (포함) 사이의 Iceberg 테이블에 추가된 모든 레코드를 검색합니다.

```
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}")
)
```

메타데이터 액세스

Iceberg는 SQL을 통해 메타데이터에 대한 액세스를 제공합니다. 네임스페이스를 쿼리하여 지정된 테이블(<table_name>)의 메타데이터에 액세스할 수 있습니다<table_name>.<metadata_table>. 메타데이터 테이블의 전체 목록은 Iceberg 설명서의 테이블 검사를 참조하세요.

다음 예제에서는 Iceberg 테이블의 커밋(변경) 기록을 보여주는 Iceberg 기록 메타데이터 테이블에 액세스하는 방법을 보여줍니다.

Amazon EMR Studio 노트북에서 Spark SQL(%%sql매직 포함) 사용:

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

DataFrames API 사용:

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

샘플 출력:

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

Trino를 사용하여 Iceberg 테이블 작업

이 섹션에서는 Amazon EMR에서 Trino를 사용하여 Iceberg 테이블을 설정하고 운영하는 방법을 설명합니다. 이 예제는 Amazon EMR on EC2 클러스터에서 실행할 수 있는 표준 문안 코드입니다. 이 섹션의 코드 예제 및 구성에서는 Amazon EMR 릴리스 emr-7.9.0을 사용한다고 가정합니다.

Amazon EMR on EC2 설정

1. 다음과 같은 콘텐츠로 `iceberg.properties` 파일을 생성합니다. `CREATE TABLE` 문에 형식이 명시적으로 지정되지 않은 경우 `iceberg.file-format=parquet` 설정에 따라 새 테이블의 기본 스토리지 형식이 결정됩니다.

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

2. `iceberg.properties` 파일을 S3 버킷에 업로드합니다.
3. S3 버킷에서 `iceberg.properties` 파일을 복사하여 생성할 Amazon EMR 클러스터에 Trino 구성 파일로 저장하는 부트스트랩 작업을 생성합니다.를 S3 버킷 이름으로 `<S3-bucket-name>` 바꿔 야 합니다.

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

4. Trino가 설치된 Amazon EMR 클러스터를 생성하고 이전 스크립트의 실행을 부트스트랩 작업으로 지정합니다. 다음은 클러스터를 생성하기 위한 sample AWS Command Line Interface (AWS CLI) 명령입니다.

```
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>"}'
```

여기서 다음을 바꿉니다.

- <S3-bucket-name> S3 버킷 이름 사용
- <region> 특정를 사용하는 AWS 리전
- <key-name>를 키 페어와 함께 사용합니다. 키 페어가 없는 경우 키 페어가 생성됩니다.
- <IAM-service-role> 최소 권한 원칙을 따르는 Amazon EMR 서비스 역할을 사용합니다.
- <EMR-EC2-instance-profile>를 인스턴스 프로파일과 함께 사용합니다.

5. Amazon EMR 클러스터가 초기화되면 다음 명령을 실행하여 Trino 세션을 초기화할 수 있습니다.

```
trino-cli
```

6. Trino CLI에서 다음을 실행하여 카탈로그를 볼 수 있습니다.

```
SHOW CATALOGS;
```

Iceberg 테이블 생성

Iceberg 테이블을 생성하려면 CREATE TABLE 문을 사용할 수 있습니다. 다음은 Iceberg 숨겨진 파티셔닝을 사용하는 파티셔닝된 테이블을 생성하는 예입니다.

```
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**
>
> 형식을 지정하지 않으면 이전 섹션에서 구성한 `iceberg.file-format` 값이 사용됩니다.

데이터를 삽입하려면 `INSERT INTO` 명령을 사용합니다. 다음은 그 예입니다.

```
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');
```

Iceberg 테이블에서 읽기

다음과 같이 `SELECT` 문을 사용하여 Iceberg 테이블의 최신 상태를 읽을 수 있습니다.

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

Iceberg 테이블로 데이터 업서스팅

`MERGE INTO` 문을 사용하여 업서트 작업을 수행할 수 있습니다(새 레코드를 동시에 삽입하고 기존 레코드를 업데이트). 다음은 그 예입니다.

```
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);
```

Iceberg 테이블에서 레코드 삭제

Iceberg 테이블에서 데이터를 삭제하려면 `DELETE FROM` 표현식을 사용하고 삭제할 행과 일치하는 필터를 지정합니다. 다음은 그 예입니다.

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

Iceberg 테이블 메타데이터 쿼리

Iceberg는 SQL을 통해 메타데이터에 대한 액세스를 제공합니다. 네임스페이스를 쿼리하여 지정된 테이블(`<table_name>`)의 메타데이터에 액세스할 수 있습니다"`<table_name>.$<metadata_table>`". 메타데이터 테이블의 전체 목록은 Iceberg 설명서의 테이블 검사를 참조하세요.

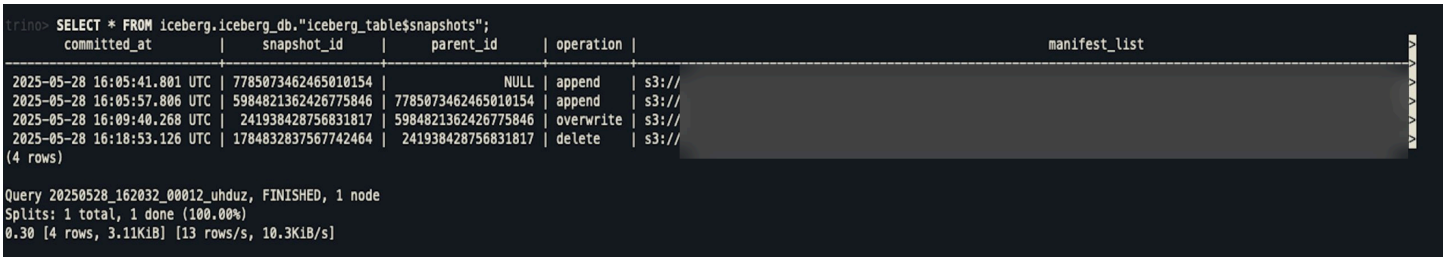
다음은 Iceberg 메타데이터를 검사하기 위한 쿼리 목록의 예입니다.

```
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";
```

예를 들어, 쿼리는 다음과 같습니다.

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

는 출력을 제공합니다.



시간 이동 사용

Iceberg 테이블의 각 쓰기 작업(삽입, 업데이트, 업서트 또는 삭제)은 새 스냅샷을 생성합니다. 그런 다음 이러한 스냅샷을 시간 이동에 사용하여 과거로 돌아가서 과거의 테이블 상태를 확인할 수 있습니다.

다음 시간 이동 쿼리는 특정를 기반으로 테이블의 상태를 표시합니다. `snapshot_id`

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

다음 시간 이동 쿼리는 특정 타임스탬프를 기반으로 테이블의 상태를 표시합니다.

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

Iceberg를 Trino와 함께 사용할 때 고려 사항

Iceberg 테이블의 Trino 쓰기 작업은 merge-on-read 설계를 따르므로 업데이트 또는 삭제의 영향을 받는 전체 데이터 파일을 다시 작성하는 대신 위치 삭제 파일을 생성합니다. copy-on-write 접근 방식을 사용하려면 쓰기 작업에 Spark를 사용하는 것이 좋습니다.

Amazon Data Firehose를 사용하여 Iceberg 테이블 작업

Amazon Data Firehose는 AWS WAF 로그, Amazon CloudWatch Logs, Amazon Amazon Kinesis Data Streams AWS IoT, Amazon Managed Streaming for Apache Kafka(Amazon MSK)와 같은 20개 이상의 소스에서 Amazon S3, Amazon Redshift, Snowflake, Splunk와 같은 대상으로 데이터 스트림을 전송하기 위한 서버리스 노코드 서비스입니다.

Firehose를 사용하여 스트리밍 데이터를 Amazon S3의 Apache Iceberg 테이블로 직접 전송할 수 있습니다. Firehose를 사용하면 단일 스트림의 레코드를 다른 Apache Iceberg 테이블로 라우팅하고 테이블의 레코드에 삽입, 업데이트 및 삭제 작업을 자동으로 적용할 수 있습니다. Firehose는 Iceberg 테이블로의 정확히 한 번 전송을 보장합니다. 이 기능을 사용하려면 AWS Glue Data Catalog를 사용해야 합니다.

Firehose는 스트리밍 데이터를 Amazon S3 테이블에 직접 전송할 수도 있습니다. 이러한 테이블은 대규모 분석 워크로드에 최적화된 스토리지를 제공하며 쿼리 성능을 지속적으로 개선하고 테이블 형식 데이터의 스토리지 비용을 줄이는 기능을 포함합니다.

데이터를 Apache Iceberg 테이블로 전송하도록 Firehose 스트림을 설정하는 방법에 대한 자세한 내용은 Firehose 설명서의 Firehose 스트림 설정 또는 블로그 게시물 Amazon Data Firehose를 사용하여 Amazon S3의 Apache Iceberg 테이블로 실시간 데이터 스트리밍을 참조하세요.

Athena SQL을 사용하여 Iceberg 테이블 작업

Amazon Athena는 Apache Iceberg에 대한 기본 지원을 제공하며 추가 단계 또는 구성이 필요하지 않습니다. 이 섹션에서는 지원되는 기능에 대한 자세한 개요와 Athena를 사용하여 Iceberg 테이블과 상호 작용하는 방법에 대한 개략적인 지침을 제공합니다.

버전 및 기능 호환성

Iceberg 테이블 사양 지원

Apache Iceberg 테이블 사양은 Iceberg 테이블의 동작 방식을 지정합니다. Athena는 테이블 형식 버전 2를 지원하므로 콘솔, CLI 또는 SDK로 생성한 모든 Iceberg 테이블은 기본적으로 해당 버전을 사용합니다.

Amazon EMR의 Apache Spark와 같은 다른 엔진으로 생성된 Iceberg 테이블을 사용하는 경우 테이블 속성을 사용하여 테이블 형식 버전을 설정해야 AWS Glue합니다. 참고로이 가이드 앞부분의 Iceberg 테이블 생성 및 작성 섹션을 참조하세요.

Iceberg 기능 지원

Athena를 사용하여 Iceberg 테이블에서 읽고 쓸 수 있습니다. `UPDATE`, `MERGE INTO`및 `DELETE FROM` 문을 사용하여 데이터를 변경하는 경우 Athena는 merge-on-read 모드만 지원합니다. 이 속성은 변경할 수 없습니다. copy-on-write로 데이터를 업데이트하거나 삭제하려면 Amazon EMR 또는의 Apache Spark와 같은 다른 엔진을 사용해야 합니다 AWS Glue. 다음 표에는 Athena에서의 Iceberg 기능 지원이 요약되어 있습니다.

		DDL 지원		DML 지원		AWS Lake Formation 보안을 위해(선택 사항)
	테이블 형식	테이블 생성	스키마 진화	데이터 읽기	데이터 쓰기	행/열 액세스 제어
Amazon Athena	버전 2	✓	✓	✓	X Copy-on-write	✓

		DDL 지원	DML 지원		AWS Lake Formation 보안을 위해(선택 사항)
				✓ Merge-on-read	✓

> **Note**
>
> - Athena는 증분 쿼리를 지원하지 않습니다.
> - Athena에서 업데이트, 삭제 및 병합 작업은 CoW가 지원되지 않으므로 테이블 속성의 쓰기 시 복사(CoW) 설정에 관계없이 항상 읽기 시 병합(MoR)으로 기본 설정됩니다.

Iceberg 테이블 작업

Athena에서 Iceberg 사용을 빠르게 시작하려면이 가이드 앞부분의 Athena SQL에서 Iceberg 테이블 시작하기 섹션을 참조하세요.

다음 표에는 제한 사항 및 권장 사항이 나열되어 있습니다.

시나리오	제한	권장 사항
테이블 DDL 생성	다른 엔진으로 생성된 Iceberg 테이블에는 Athena에 노출되지 않는 속성이 있을 수 있습니다. 이러한 테이블의 경우 DDL을 생성할 수 없습니다.	테이블을 생성한 엔진에서 동등한 문(예: Spark `SHOW CREATE TABLE` 문)을 사용합니다.
Iceberg 테이블에 기록된 객체의 무작위 Amazon S3 접두사	기본적으로 Athena로 생성된 Iceberg 테이블에는 `write.object-storage.enabled` 속성이 활성화되어 있습니다.	이 동작을 비활성화하고 Iceberg 테이블 속성을 완전히 제어하려면 Amazon EMR 또는의 Spark와 같은 다른 엔진을

시나리오	제한	권장 사항
		사용하여 Iceberg 테이블을 생성합니다 AWS Glue.
증분 쿼리	현재 Athena에서는 지원되지 않습니다.	증분 쿼리를 사용하여 증분 데이터 수집 파이프라인을 활성화하려면 Amazon EMR 또는 에서 Spark를 사용합니다 AWS Glue.

PyIceberg 테이블 작업

이 섹션에서는 PyIceberg를 사용하여 Iceberg 테이블과 상호 작용하는 방법을 설명합니다. 제공된 예제는 Amazon Linux 2023 EC2 인스턴스, AWS Lambda 함수 또는 자격 AWS 증명이 올바르게 구성된 Python 환경에서 실행할 수 있는 표준 문안 코드입니다.

사전 조건

> **Note**
>
> 이 예제에서는 PyIceberg 1.9.1을 사용합니다.

PyIceberg를 사용하려면 PyIceberg 및가 AWS SDK for Python (Boto3) 설치되어 있어야 합니다. 다음은 PyIceberg 및와 함께 작동하도록 Python 가상 환경을 설정하는 방법의 예입니다. AWS Glue Data Catalog

1. pip python 패키지 설치 관리자를 사용하여 PyIceberg를 다운로드합니다. 또한와 상호 작용하려면 Boto3가 필요합니다 AWS 서비스. 다음 명령을 사용하여 테스트하도록 로컬 Python 가상 환경을 구성할 수 있습니다.

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

2. 를 실행`python`하여 Python 셸을 열고 명령을 테스트합니다.

데이터 카탈로그에 연결

에서 Iceberg 테이블 작업을 시작하려면 AWS Glue먼저에 연결해야 합니다 AWS Glue Data Catalog.

`load_catalog`함수는 모든 Iceberg 작업에 대한 기본 인터페이스 역할을 하는 카탈로그 객체를 생성하여 데이터 카탈로그에 대한 연결을 초기화합니다.

```
from pyiceberg.catalog import load_catalog
```

```
region = "us-east-1"

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

데이터베이스 나열 및 생성

기존 데이터베이스를 나열하려면 `list_namespaces` 함수를 사용합니다.

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

새 데이터베이스를 생성하려면 `create_namespace` 함수를 사용합니다.

```
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})
```

Iceberg 테이블 생성 및 작성

분할되지 않은 테이블

다음은 `create_table` 함수를 사용하여 파티셔닝되지 않은 Iceberg 테이블을 생성하는 예제입니다.

```
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)
```

`list_tables` 함수를 사용하여 데이터베이스 내의 테이블 목록을 확인할 수 있습니다.

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

함수와 PyArrow를append 사용하여 Iceberg 테이블 내에 데이터를 삽입할 수 있습니다.

```
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)
```

분할된 테이블

다음은 `create_table` 함수 및를 사용하여 숨겨진 파티셔닝이 있는 파티셔닝된 Iceberg 테이블을 생성하는 예제입니다`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
)
```

파티셔닝되지 않은 테이블과 동일한 방식으로 파티셔닝된 테이블에 데이터를 삽입할 수 있습니다. 파티셔닝은 자동으로 처리됩니다.

```
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)
```

데이터 읽기

PyIceberg `scan` 함수를 사용하여 Iceberg 테이블에서 데이터를 읽을 수 있습니다. 행을 필터링하고, 특정 열을 선택하고, 반환된 레코드 수를 제한할 수 있습니다.

```
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)
```

데이터 삭제

PyIceberg `delete` 함수를 사용하면 `delete_filter`를 사용하여 테이블에서 레코드를 제거할 수 있습니다.

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

메타데이터 액세스

PyIceberg는 테이블 메타데이터에 액세스하는 여러 함수를 제공합니다. 테이블 스냅샷에 대한 정보를 보는 방법은 다음과 같습니다.

```
#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}")
```

사용 가능한 메타데이터의 자세한 목록은 PyIceberg 설명서의 메타데이터 코드 참조 섹션을 참조하세요.

시간 이동 사용

테이블 스냅샷을 시간 이동에 사용하여 테이블의 이전 상태에 액세스할 수 있습니다. 다음은 마지막 작업 이전의 테이블 상태를 보는 방법입니다.

```
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)
```

사용 가능한 함수의 전체 목록은 PyIcebergPython API 설명서를 참조하세요.

Iceberg 테이블 형식 사양 버전 3 작업

Apache Iceberg 테이블 형식 사양의 최신 버전은 버전 3입니다. 이 버전은 성능을 개선하고 운영 오버헤드를 줄이면서 페타바이트 규모의 데이터 레이크를 빌드하는 고급 기능을 소개합니다. 특히 배치 업데이트 및 규정 준수 삭제 작업과 관련하여 버전 2에서 발생하는 일반적인 성능 병목 현상을 해결합니다.

AWS 는 Iceberg 버전 3 사양에 정의된 대로 삭제 벡터 및 행 계보에 대한 지원을 제공합니다. 이러한 기능은 다음과 같은 Apache Spark에서 사용할 수 있습니다 AWS 서비스.

AWS 서비스	버전 3 지원
Amazon EMR for Apache Spark	Amazon EMR 릴리스 7.12 이상
AWS Glue	예
AWS Glue: Iceberg REST API, 테이블 유지 관리	예
Amazon SageMaker Unified Studio 노트북	예
Amazon S3 테이블: Iceberg REST API, 테이블 유지 관리	예
Amazon Athena(트리노)	아니요

버전 3의 주요 기능

삭제 벡터는 버전 2에서 사용된 위치 삭제 파일을 Puffin 파일로 저장된 효율적인 바이너리 형식으로 대체합니다. 이렇게 하면 임의 배치 업데이트 및 일반 데이터 보호 규정(GDPR) 규정 준수 삭제로 인한 쓰기 증폭이 제거되고 새로운 데이터 유지 관리 오버헤드가 크게 줄어듭니다. 고빈도 업데이트를 처리하는 조직은 쓰기 성능이 즉시 향상되고 작은 파일 수로 인해 스토리지 비용이 절감됩니다.

행 계보를 사용하면 행 수준에서 정확한 변경 내용 추적이 가능합니다. 다운스트림 시스템은 변경 사항을 점진적으로 처리하여 데이터 파이프라인 속도를 높이고 변경 데이터 캡처(CDC) 워크플로의 컴퓨팅 비용을 절감할 수 있습니다. 이 내장 기능을 사용하면 사용자 지정 변경 추적 구현이 필요하지 않습니다.

버전 호환성

버전 3은 버전 2 테이블과의 이전 버전과의 호환성을 유지합니다. AWS 서비스는 버전 2 및 버전 3 테이블을 동시에 지원하므로 다음을 수행할 수 있습니다.

- 버전 2 및 버전 3 테이블 모두에서 쿼리를 실행합니다.
- 데이터 재작성 없이 기존 버전 2 테이블을 버전 3으로 업그레이드합니다.
- 버전 2 및 버전 3 스냅샷에 적용되는 런타임 이동 쿼리입니다.
- 테이블 버전 간에 스키마 진화와 숨겨진 파티셔닝을 사용합니다.

버전 3 시작하기

사전 조건

버전 3 테이블로 작업하기 전에 다음이 있는지 확인합니다.

- 적절한 AWS Identity and Access Management (IAM) 권한이 AWS 계정 있는 .
- 하나 이상의 AWS 분석 서비스(Amazon EMR, AWS Glue Amazon SageMaker Unified Studio 노트북 또는 Amazon S3 Tables)에 대한 액세스.
- 테이블 데이터 및 메타데이터를 저장하기 위한 S3 버킷입니다.
- 자체 Iceberg 인프라를 구축하는 경우 Amazon S3 Tables 또는 범용 S3 버킷을 시작할 테이블 버킷입니다.
- 구성된 AWS Glue 카탈로그입니다.

버전 3 테이블 생성

새 테이블 생성

새 Iceberg 버전 3 테이블을 생성하려면 `format-version` 테이블 속성을 3으로 설정합니다.

Spark SQL 사용:

```
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'
)
```

버전 2 테이블을 버전 3으로 업그레이드

데이터를 다시 작성 하지 않고도 기존 버전 2 테이블을 버전 3으로 원자적으로 업그레이드할 수 있습니다.

Spark SQL 사용:

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

> **Important**
>
> 버전 3은 단방향 업그레이드입니다. 테이블을 버전 2에서 버전 3으로 업그레이드한 후에는 표준 작업을 통해 다시 버전 2로 다운그레이드할 수 없습니다.

업그레이드 중에 발생하는 일:

- 새 메타데이터 스냅샷은 원자적으로 생성됩니다.
- 기존 Parquet 데이터 파일은 재사용됩니다.
- 행 계보 필드가 테이블 메타데이터에 추가됩니다.

업그레이드 후:

- 다음 압축을 수행하면 이전 버전 2 삭제 파일이 제거됩니다.
- 새 수정 사항은 버전 3 삭제 벡터 파일을 사용합니다.

업그레이드는 행 계보 변경 추적 레코드의 과거 채우기를 수행하지 않습니다.

삭제 벡터 활성화

업데이트, 삭제 및 병합을 위해 삭제 벡터를 활용하려면 쓰기 모드를 구성합니다.

Spark SQL 사용:

```
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'
                  )
```

이러한 설정을 통해 업데이트, 삭제 및 병합 작업이 전체 데이터 파일을 다시 쓰는 대신 삭제 벡터 파일을 생성할 수 있습니다.

변경 추적에 행 계보 사용

버전 3은 행 계보 메타데이터 필드를 자동으로 추가하여 변경 사항을 추적합니다.

Spark SQL 사용:

```
# 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` 필드는 각 행을 고유하게 식별하고 행이 마지막으로 수정된 시기를 `_last_updated_sequence_number` 추적합니다. 다음 필드를 사용하여 다음을 수행할 수 있습니다.

- 증분 처리를 위해 변경된 행을 식별합니다.
- 규정 준수를 위해 데이터 계보를 추적합니다.
- CDC 파이프라인을 최적화합니다.

- 변경 사항만 처리하여 컴퓨팅 비용을 줄입니다.

버전 3 모범 사례

버전 3을 사용해야 하는 경우

다음과 같은 경우 버전 3으로 업그레이드하거나 버전 3으로 시작하는 것이 좋습니다.

- 배치 업데이트 또는 삭제를 자주 수행합니다.
- GDPR 또는 규정 준수 삭제 요구 사항을 충족해야 합니다.
- 워크로드에는 고주파 업서트가 포함됩니다.
- 효율적인 CDC 워크플로가 필요합니다.
- 작은 파일에서 스토리지 비용을 줄이려고 합니다.
- 더 나은 변경 추적 기능이 필요합니다.

쓰기 성능 최적화

- 업데이트가 많은 워크로드에 대해 삭제 벡터를 활성화합니다.

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

- 적절한 파일 크기를 구성합니다.

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

읽기 성능 최적화

- 증분 처리에 행 계보를 사용합니다.
- 복사하지 않고 시간 이동을 사용하여 기록 데이터에 액세스합니다.
- 더 나은 쿼리 계획을 위해 통계 수집을 활성화합니다.

마이그레이션 전략

버전 2에서 버전 3으로 마이그레이션할 때는 다음 모범 사례를 따르세요.

- 먼저 비프로덕션 환경에서 테스트하여 업그레이드 프로세스와 성능을 검증합니다.
- 활동이 적은 기간 동안 업그레이드하여 동시 작업에 미치는 영향을 최소화합니다.
- 초기 성능을 모니터링하고 업그레이드 후 지표를 추적합니다.
- 업그레이드 후 압축을 실행하여 삭제 파일을 통합합니다.
- 버전 3 기능을 반영하도록 팀 설명서를 업데이트합니다.

호환성 고려 사항

- 엔진 버전 - 테이블에 액세스하는 모든 엔진이 버전 3을 지원하는지 확인합니다.
- 타사 도구 - 업그레이드하기 전에 도구의 버전 3 호환성을 확인합니다.
- 백업 전략 - 스냅샷 기반 복구 절차를 테스트합니다.
- 모니터링 - 버전 3별 지표에 대한 모니터링 대시보드를 업데이트합니다.

문제 해결

일반적인 문제

오류: "format-version 3 is not supported"

- 엔진 버전이 버전 3을 지원하는지 확인합니다. 자세한 내용은이 섹션의 시작 부분에 있는 표를 참조하세요.
- 카탈로그 호환성을 확인합니다.
- 의 최신 버전을 사용하고 있는지 확인합니다 AWS 서비스.

업그레이드 후 성능 저하

- 압축 압축 실패가 없는지 확인합니다. 자세한 내용은 Amazon S3 S3 설명서의 S3 테이블에 대한 로깅 및 모니터링을 참조하세요.
- 삭제 벡터가 활성화되어 있는지 확인합니다. 다음 속성을 설정해야 합니다.

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

다음 코드를 사용하여 테이블 속성을 확인할 수 있습니다.

```
DESCRIBE FORMATTED myns.orders_v3
```

- 파티션 전략을 검토합니다. 과도하게 파티셔닝하면 작은 파일이 생성될 수 있습니다. 다음 쿼리를 실행하여 테이블의 평균 파일 크기를 가져옵니다.

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

타사 도구와의 비호환성

- 도구가 버전 3 사양을 지원하는지 확인합니다.
- 지원되지 않는 도구에 대해 버전 2 테이블을 유지 관리하는 것이 좋습니다.
- 버전 3 지원 타임라인은 도구 공급업체에 문의하세요.

도움말 가져오기

- 특정 문제는 AWS 서비스에 문의하세요AWS Support.
- Iceberg 커뮤니티의 도움을 받으려면 Iceberg Slack 채널을 사용합니다.
- 를 사용하여 분석 워크로드 AWS 서비스 를 관리하는 방법에 대한 자세한 내용은 의 분석을 AWS참조하세요.

가격 책정

- Amazon EMR 컴퓨팅 및 스토리지 요금
- Amazon SageMaker 요금
- AWS Glue 작업 실행 및 데이터 카탈로그 요금

- S3 Tables 스토리지 및 요청 요금

가용성

Iceberg 테이블 형식 사양 버전 3 지원은 Amazon EMR AWS Glue, AWS Glue Data Catalog및 S3 Tables가 작동하는 모든 AWS 리전 에서 사용할 수 있습니다. 리전 가용성은 리전별AWS 서비스 섹션을 참조하세요.

추가 리소스

- Apache Iceberg 설명서
- Apache Iceberg 테이블 사양
- Amazon S3에서 S3 테이블로 테이블 형식 데이터를 마이그레이션하기 위한 지침
- 자습서: S3 Tables 시작하기

기존 테이블을 Iceberg로 마이그레이션

이 섹션에서는 기존 Hive 스타일 테이블을 Iceberg 형식으로 마이그레이션하는 데 중점을 둡니다. Apache Parquet 또는 Apache ORC와 같은 기존 Hive 호환 형식을 사용하는 테이블에 적용됩니다. 이 정보는 Linux Foundation Delta Lake 또는 Apache Hudi와 같은 최신 테이블 형식을 이미 사용하는 테이블에는 적용되지 않습니다.

현재 Hive 스타일 테이블을 Iceberg 형식으로 마이그레이션하려면 현재 위치 또는 전체 데이터 마이그레이션을 사용할 수 있습니다.

- 현재 위치 마이그레이션은 기존 데이터 파일 위에 Iceberg의 메타데이터 파일을 생성하는 프로세스입니다.
- 전체 데이터 마이그레이션은 Iceberg 메타데이터 계층을 생성하고 원본 테이블의 기존 데이터 파일을 새 Iceberg 테이블로 다시 씁니다.

다음 섹션에서는 구현을 위한 step-by-step 지침 및 고려 사항을 포함하여 각 마이그레이션 방법에 대한 자세한 개요를 제공합니다. 이러한 마이그레이션 전략에 대한 자세한 내용은 Iceberg 설명서의 테이블 마이그레이션 섹션을 참조하세요.

현재 위치 및 전체 데이터 마이그레이션 방법에 대한 세부 정보를 검토한 후 의사 결정 프로세스에 도움이 되는 다음 두 가지 주요 섹션을 참조하세요.

- 마이그레이션 전략을 선택하면 특정 요구 사항 및 사용 사례에 따라 가장 적합한 마이그레이션 접근 방식을 결정하는 데 도움이 되는 일련의 질문 및 시나리오에 대한 지침이 제공됩니다.
- 마이그레이션 옵션 요약은 다양한 마이그레이션 옵션에서 주요 특성과 고려 사항을 비교하는 포괄적인 표를 제공합니다. 이 표는 빠른 참조 가이드 역할을 하며 메서드 간의 기술적 장단점을 이해하는 데 도움이 되는 기능 비교를 제공합니다.

현재 위치 마이그레이션

인플레이스 마이그레이션을 사용하면 모든 데이터 파일을 다시 쓸 필요가 없습니다. 대신 Iceberg 메타데이터 파일이 생성되어 기존 데이터 파일에 연결됩니다. 이 방법은 일반적으로 더 빠르고 비용 효율적이며, 특히 Parquet, Avro 및 ORC와 같은 호환되는 파일 형식이 있는 대규모 데이터 세트 또는 테이블의 경우 더욱 효과적입니다.

> **Note**
>
> Amazon S3 Tables로 마이그레이션할 때는 현재 위치 마이그레이션을 사용할 수 없습니다.

Iceberg는 인플레이스 마이그레이션을 구현하기 위한 두 가지 주요 옵션을 제공합니다.

- 스냅샷 절차를 사용하여 소스 테이블을 변경하지 않고 새 Iceberg 테이블을 생성합니다. 자세한 내용은 Iceberg 설명서의 스냅샷 테이블을 참조하세요.
- 마이그레이션 절차를 사용하여 소스 테이블을 대체하는 새 Iceberg 테이블을 생성합니다. 자세한 내용은 Iceberg 설명서의 테이블 마이그레이션을 참조하세요. 이 절차는 Hive 메타스토어(HMS)에서 작동하지만 현재와 호환되지 않습니다 AWS Glue Data Catalog. 이 가이드의 뒷부분에 있는 섹션의 테이블 마이그레이션 복제 절차는 AWS Glue Data Catalog 데이터 카탈로그로 유사한 결과를 달성하기 위한 해결 방법을 제공합니다.

`snapshot` 또는를 사용하여 현재 위치 마이그레이션을 수행한 후 `migrate`일부 데이터 파일은 마이그레이션되지 않은 상태로 유지될 수 있습니다. 이는 일반적으로 마이그레이션 도중 또는 이후에 라이터가 소스 테이블에 계속 쓸 때 발생합니다. 이러한 나머지 파일을 Iceberg 테이블에 통합하려면 add_files 프로시저를 사용할 수 있습니다. 자세한 내용은 Iceberg 설명서의 파일 추가를 참조하세요.

다음과 같이 Athena에서 생성되고 채워진 Parquet 기반 `products` 테이블이 있다고 가정해 보겠습니다.

```
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');
```

다음 섹션에서는이 테이블에서 `snapshot` 및 `migrate` 프로시저를 사용하는 방법을 설명합니다.

옵션 1: 스냅샷 프로시저

이 snapshot 절차에서는 이름이 다른 새 Iceberg 테이블을 생성하지만 소스 테이블의 스키마와 파티셔닝을 복제합니다. 이 작업은 작업 중과 작업 후에 소스 테이블을 완전히 변경하지 않습니다. 테이블의 경량 복사본을 효과적으로 생성하므로 원래 데이터 소스를 수정하지 않고도 시나리오 또는 데이터 탐색을 테스트하는 데 특히 유용합니다. 이 접근 방식을 사용하면 원래 테이블과 Iceberg 테이블을 모두 사용할 수 있는 전환 기간이 활성화됩니다(이 섹션의 끝에 있는 참고 사항 참조). 테스트가 완료되면 모든 라이터와 리더를 새 테이블로 전환하여 새 Iceberg 테이블을 프로덕션 테이블로 이동할 수 있습니다.

모든 Amazon EMR 배포 모델(예: Amazon EMR on EC2, Amazon EMR on EKS, EMR Serverless) 및 에서 Spark를 사용하여 snapshot 프로시저를 실행할 수 있습니다 AWS Glue.

snapshot Spark 절차를 사용하여 인플레이스 마이그레이션을 테스트하려면 다음 단계를 따르세요.

1. Spark 애플리케이션을 시작하고 다음 설정으로 Spark 세션을 구성합니다.
   - `"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. snapshot 프로시저를 실행하여 원래 테이블 데이터 파일을 가리키는 새 Iceberg 테이블을 생성합니다.

```
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)
```

   출력 데이터 프레임에는 imported_files_count (추가된 파일 수)가 포함됩니다.

3. 쿼리를 통해 새 테이블을 검증합니다.

```
spark.sql(f"""
SELECT * FROM mydb.products_iceberg LIMIT 10
"""
```

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

참고

- 프로시저를 실행한 후 소스 테이블에서 데이터 파일을 수정하면 생성된 테이블이 동기화되지 않습니다. 추가하는 새 파일은 Iceberg 테이블에 표시되지 않으며 제거한 파일은 Iceberg 테이블의 쿼리 기능에 영향을 미칩니다. 동기화 문제를 방지하려면:
  - 새 Iceberg 테이블이 프로덕션용인 경우 원본 테이블에 쓰는 모든 프로세스를 중지하고 새 테이블로 리디렉션합니다.
  - 전환 기간이 필요하거나 새 Iceberg 테이블이 테스트용인 경우 테이블 동기화 유지에 대한 지침은이 섹션 뒷부분의 인플레이스 마이그레이션 후 Iceberg 테이블 동기화 유지를 참조하세요.
- snapshot 프로시저를 사용하면 생성된 Iceberg 테이블의 테이블 gc.enabled 속성false에서 속성이 로 설정됩니다. 이 설정은 옵션을 DROP TABLE 사용하여 데이터 파일을 물리적으로 삭제PURGE하는 expire_snapshotsremove_orphan_files, 또는 등의 작업을 금지합니다. 소스 파일에 직접적인 영향을 주지 않는 Iceberg 삭제 또는 병합 작업은 여전히 허용됩니다.
- 데이터 파일을 물리적으로 삭제하는 작업에 대한 제한 없이 새 Iceberg 테이블이 완전히 작동하도록 하려면 gc.enabled 테이블 속성을 로 변경할 수 있습니다true. 그러나이 설정은 원본 테이블에 대한 액세스를 손상시킬 수 있는 소스 데이터 파일에 영향을 미치는 작업을 허용합니다. 따라서 원래 테이블의 기능을 더 이상 유지할 필요가 없는 경우에만 gc.enabled 속성을 변경합니다. 예제:

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

옵션 2: 마이그레이션 절차

이 migrate 절차에서는 소스 테이블과 이름, 스키마 및 파티셔닝이 동일한 새 Iceberg 테이블을 생성합니다. 이 프로시저가 실행되면 소스 테이블을 잠그고 이름을 <table_name>_BACKUP_ (또는 backup_table_name 프로시저 파라미터로 지정된 사용자 지정 이름)으로 바꿉니다.

> **Note**
>
> drop_backup 프로시저 파라미터를 로 설정하면 true원래 테이블이 백업으로 유지되지 않습니다.

따라서 migrate 테이블 절차에서는 작업을 수행하기 전에 소스 테이블에 영향을 미치는 모든 수정 사항을 중지해야 합니다. migrate 프로시저를 실행하기 전에:

- 소스 테이블과 상호 작용하는 모든 라이터를 중지합니다.
- 기본적으로 Iceberg를 지원하지 않는 리더와 라이터를 수정하여 Iceberg 지원을 활성화합니다.

예제:

- Athena는 수정 없이 계속 작동합니다.
- Spark에는 다음이 필요합니다.
  - 클래스 경로에 포함될 Iceberg Java Archive(JAR) 파일(이 가이드 앞부분의 섹션에서 Amazon EMR에서 Iceberg 작업 및 Iceberg 작업 AWS Glue 참조).
  - 다음 Spark 세션 카탈로그 구성(SparkSessionCatalog를 사용하여 Iceberg가 아닌 테이블의 기본 제공 카탈로그 기능을 유지하면서 Iceberg 지원을 추가):
    - "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

프로시저를 실행한 후 새 Iceberg 구성으로 라이터를 다시 시작할 수 있습니다.

현재는 데이터 카탈로그가 RENAME 작업을 지원하지 AWS Glue Data Catalog않으므로 migrate 프로시저가와 호환되지 않습니다. 따라서 Hive 메타스토어로 작업하는 경우에만이 절차를 사용하는 것이 좋습니다. 데이터 카탈로그를 사용하는 경우 대체 접근 방식은 다음 섹션을 참조하세요.

모든 Amazon EMR 배포 모델(EC2의 Amazon EMR, EKS의 Amazon EMR, EMR Serverless) 및에서 migrate 프로시저를 실행할 수 AWS Glue있지만 Hive 메타스토어에 대해 구성된 연결이 필요합니다. Amazon EMR on EC2는 설정 복잡성을 최소화하는 기본 제공 Hive 메타스토어 구성을 제공하므로 권장되는 선택입니다.

Hive 메타스토어로 구성된 EC2 클러스터의 Amazon EMR에서 `migrate` Spark 절차를 사용하여 인플레이스 마이그레이션을 테스트하려면 다음 단계를 따르세요.

1. Spark 애플리케이션을 시작하고 Iceberg Hive 카탈로그 구현을 사용하도록 Spark 세션을 구성합니다. 예를 들어 `pyspark` CLI를 사용하는 경우:

```
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. Hive 메타스토어에서 `products` 테이블을 생성합니다. 이는 일반적인 마이그레이션에 이미 존재하는 소스 테이블입니다.

   a. Hive 메타스토어에서 `products` 외부 Hive 테이블을 생성하여 Amazon S3의 기존 데이터를 가리킵니다.

```
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. `MSCK REPAIR TABLE` 명령을 사용하여 기존 파티션을 추가합니다.

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

   c. `SELECT` 쿼리를 실행하여 테이블에 데이터가 포함되어 있는지 확인합니다.

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

샘플 출력:

```
>>> 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. Iceberg `migrate` 절차를 사용합니다.

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

df_res.show()
```

출력 데이터 프레임에는 `migrated_files_count` (Iceberg 테이블에 추가된 파일 수)가 포함됩니다.

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

4. 백업 테이블이 생성되었는지 확인합니다.

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

샘플 출력:

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

5. Iceberg 테이블을 쿼리하여 작업을 검증합니다.

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

참고

- 절차를 실행한 후 Iceberg 지원으로 제대로 구성되지 않은 경우 소스 테이블에 쿼리하거나 쓰는 모든 현재 프로세스에 영향을 미칩니다. 따라서 다음 단계를 따르는 것이 좋습니다.
  1. 마이그레이션하기 전에 소스 테이블을 사용하여 모든 프로세스를 중지합니다.
  2. 마이그레이션을 수행합니다.
  3. 적절한 Iceberg 설정을 사용하여 프로세스를 다시 활성화합니다.
- 마이그레이션 프로세스 중에 데이터 파일 수정이 발생하면(새 파일이 추가되거나 파일이 제거됨) 생성된 테이블이 동기화되지 않습니다. 동기화 옵션은이 섹션 뒷부분의 인플레이스 마이그레이션 후 Iceberg 테이블 동기화 유지를 참조하세요.

에서 테이블 마이그레이션 절차 복제 AWS Glue Data Catalog

다음 단계에 따라에서 마이그레이션 절차의 결과를 복제할 수 있습니다 AWS Glue Data Catalog (원본 테이블 백업 및 Iceberg 테이블로 교체).

1. 스냅샷 절차를 사용하여 원본 테이블의 데이터 파일을 가리키는 새 Iceberg 테이블을 생성합니다.
2. 데이터 카탈로그에서 원본 테이블 메타데이터를 백업합니다.
   a. GetTable API를 사용하여 소스 테이블 정의를 검색합니다.
   b. GetPartitions API를 사용하여 소스 테이블 파티션 정의를 검색합니다.

c. CreateTable API를 사용하여 데이터 카탈로그에 백업 테이블을 생성합니다.

d. CreatePartition 또는 BatchCreatePartition API를 사용하여 데이터 카탈로그의 백업 테이블에 파티션을 등록합니다.

3. gc.enabled Iceberg 테이블 속성을 로 변경false하여 전체 테이블 작업을 활성화합니다.
4. 원본 테이블을 삭제합니다.
5. 테이블 루트 위치의 메타데이터 폴더에서 Iceberg 테이블 메타데이터 JSON 파일을 찾습니다.
6. register_table 프로시저를 원래 테이블 이름과 snapshot 프로시저로 생성된 metadata.json 파일의 위치와 함께 사용하여 데이터 카탈로그에 새 테이블을 등록합니다.

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

인플레이스 마이그레이션 후 Iceberg 테이블 동기화 유지

이 add_files 절차는 기존 데이터를 Iceberg 테이블에 통합하는 유연한 방법을 제공합니다. 특히 Iceberg의 메타데이터 계층에서 절대 경로를 참조하여 기존 데이터 파일(예: Parquet 파일)을 등록합니다. 기본적으로 프로시저는 모든 테이블 파티션의 파일을 Iceberg 테이블에 추가하지만 특정 파티션의 파일을 선택적으로 추가할 수 있습니다. 이 선택적 접근 방식은 여러 시나리오에서 특히 유용합니다.

- 초기 마이그레이션 후 새 파티션이 소스 테이블에 추가되는 경우.
- 초기 마이그레이션 후 데이터 파일이 기존 파티션에 추가되거나 기존 파티션에서 제거되는 경우. 그러나 수정된 파티션을 다시 추가하려면 먼저 파티션을 삭제해야 합니다. 이에 대한 자세한 내용은이 섹션의 뒷부분에 나와 있습니다.

다음은 새 Iceberg 테이블을 소스 데이터 파일과 동기화된 상태로 유지하기 위해 인플레이스 마이그레이션(snapshot 또는 migrate)을 수행한 후 add_file 절차를 사용할 때 고려해야 할 몇 가지 사항입니다.

- 소스 테이블의 새 파티션에 새 데이터가 추가되면 옵션과 함께 add_files 프로시저를 partition_filter 사용하여 이러한 추가 사항을 Iceberg 테이블에 선택적으로 통합합니다.

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

또는:

```
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)
```

- 이 add_files 프로시저는 partition_filter 옵션을 지정할 때 전체 소스 테이블 또는 특정 파티션의 파일을 스캔하고 찾은 모든 파일을 Iceberg 테이블에 추가하려고 시도합니다. 기본적으로 check_duplicate_files 프로시저 속성은 로 설정true되므로 Iceberg 테이블에 파일이 이미 있는 경우 프로시저가 실행되지 않습니다. 이는 이전에 추가한 파일을 건너뛰는 기본 제공 옵션이 없고 비활성화check_duplicate_files하면 파일이 두 번 추가되어 중복이 생성되므로 중요합니다. 소스 테이블에 새 파일이 추가되면 다음 단계를 따릅니다.
  1. 새 파티션의 경우와 add_files 함께 partition_filter를 사용하여 새 파티션에서 파일만 가져옵니다.
  2. 기존 파티션의 경우 먼저 Iceberg 테이블에서 파티션을 삭제한 다음를 지정하여 해당 파티션에 add_files 대해 다시 실행합니다partition_filter. 예제:

```
# 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**
>
> 모든 `add_files` 작업은 추가된 데이터가 포함된 새 Iceberg 테이블 스냅샷을 생성합니다.

올바른 인플레이스 마이그레이션 전략 선택

최적의 인플레이스 마이그레이션 전략을 선택하려면 다음 표의 질문을 고려하세요.

질문	권장 사항	설명
테스트 또는 점진적 전환을 위해 Hive 테이블과 Iceberg 테이블 모두에 액세스할 수 있도록 유지하면서 데이터를 다시 작성하지 않고 빠르게 마이그레이션하고 싶으신가요?	`snapshot` 프로시저 다음에 `add_files` 프로시저	`snapshot` 절차를 사용하여 소스 테이블을 수정하지 않고 스키마를 복제하고 데이터 파일을 참조하여 새 Iceberg 테이블을 생성합니다. `add_files` 절차를 사용하여 마이그레이션 후 추가되거나 수정된 파티션을 통합합니다. 수정된 파티션

질문	권장 사항	설명
		을 다시 추가하려면 먼저 파티션 삭제가 필요합니다.
Hive 메타스토어를 사용 중이며 데이터를 다시 작성하지 않고 Hive 테이블을 즉시 Iceberg 테이블로 바꾸고 싶으신가요?	`migrate` 프로시저 다음에 `add_files` 프로시저	`migrate` 절차를 사용하여 Iceberg 테이블을 생성하고, 소스 테이블을 백업하고, 원래 테이블을 Iceberg 버전으로 바꿉니다. 참고:이 옵션은 Hive Metastore와 호환되지만 와는 호환되지 않습니다 AWS Glue Data Catalog. `add_files` 절차를 사용하여 마이그레이션 후 추가되거나 수정된 파티션을 통합합니다. 수정된 파티션을 다시 추가하려면 먼저 파티션 삭제가 필요합니다.

질문	권장 사항	설명
를 사용 중이며 데이터를 다시 작성하지 않고 Hive 테이블을 즉시 Iceberg 테이블로 바꾸고 싶으 AWS Glue Data Catalog 신가요?	`migrate` 프로시저 조정 후 `add_files` 프로시저	복제 `migrate` 프로시저 동작: 1. `snapshot`를 사용하여 Iceberg 테이블을 생성합니다. 2. API를 사용하여 AWS Glue 원래 테이블 메타데이터를 백업합니다. APIs 3. Iceberg 테이블 속성`gc.enabled` 에서를 활성화합니다. 4. 원본 테이블을 삭제합니다. 5. `register_table` 를 사용하여 원래 이름으로 새 테이블 항목을 생성합니다. 참고: 이 옵션을 사용하려면 메타데이터 백업을 위한 AWS Glue API 호출을 수동으로 처리해야 합니다. `add_files` 절차를 사용하여 마이그레이션 후 추가되거나 수정된 파티션을 통합합니다. 수정된 파티션을 다시 추가하려면 먼저 파티션 삭제가 필요합니다.

전체 데이터 마이그레이션

전체 데이터 마이그레이션은 데이터 파일과 메타데이터를 다시 생성합니다. 이 접근 방식은 인플레이스 마이그레이션에 비해 시간이 더 오래 걸리고 추가 컴퓨팅 리소스가 필요합니다. 그러나 전체 데이터

마이그레이션은 테이블 품질을 개선하고 데이터 스토리지 및 액세스 패턴을 최적화할 수 있는 상당한 기회를 제공합니다.

전체 데이터 마이그레이션 중에 무결성과 정확성을 보장하기 위한 데이터 검증, 현재 요구 사항을 더 잘 충족하기 위한 스키마 수정, 쿼리 성능 개선을 위한 파티션 전략 조정과 같은 몇 가지 유용한 작업을 수행할 수 있습니다. 또한 데이터를 재정렬하여 일반적인 액세스 패턴을 최적화하고, 쿼리 효율성을 높이기 위해 Iceberg 숨겨진 파티셔닝을 구현하고, 원하는 경우 파일 형식 변환(예: CSV에서 Parquet으로)을 수행할 수 있습니다.

이러한 기능을 통해 전체 데이터 마이그레이션은 Iceberg 형식으로 전환하고 데이터 스토리지 전략을 포괄적으로 구체화하고 최적화하는 데 이상적입니다. 전체 데이터 마이그레이션에는 더 많은 시간과 리소스가 필요하지만 데이터 품질, 조직 및 쿼리 성능이 개선되면 장기적인 이점을 얻을 수 있습니다. 전체 데이터 마이그레이션을 구현하려면 다음 옵션 중 하나를 사용합니다.

- Spark(Amazon EMR 또는) 또는 Athena에서 `CREATE TABLE ... AS SELECT` (CTAS AWS Glue) 문을 사용합니다. 및 `TBLPROPERTIES` 절을 사용하여 `PARTITIONED BY` 새 Iceberg 테이블에 대한 파티션 사양 및 테이블 속성을 설정할 수 있습니다. 소스 테이블에서 스키마 및 파티셔닝을 상속하는 대신 필요에 따라 새 테이블의 스키마 및 파티셔닝을 변경할 수 있습니다.
- 소스 테이블에서 읽고 Amazon EMR 또는의 Spark를 사용하여 데이터를 새 Iceberg 테이블로 작성합니다 AWS Glue. 자세한 내용은 Iceberg 설명서의 테이블 생성을 참조하세요.

마이그레이션 전략 선택

Iceberg 형식으로 전환할 때는 현재 위치 마이그레이션과 전체 마이그레이션 중에서 선택하는 것이 중요합니다. 특정 요구 사항에 가장 적합한 접근 방식을 결정하려면 다음 질문과 권장 사항을 고려하세요.

질문	권장 사항
데이터 파일 형식(예: CSV 또는 Apache Parquet)은 무엇입니까?	• 테이블 파일 형식이 Parquet, ORC 또는 Avro인 경우 현재 위치 마이그레이션을 고려하세요. • CSV, JSON 등과 같은 다른 형식의 경우 전체 데이터 마이그레이션을 사용합니다.
테이블 스키마를 업데이트하거나 통합하시겠습니까?	• Iceberg 네이티브 기능을 사용하여 테이블 스키마를 발전시키려면 인플레이스 마이그레이션을 고려하세요. 예를 들어 마이

질문	권장 사항
	그레이션 후 열의 이름을 바꿀 수 있습니다. (스키마는 Iceberg 메타데이터 계층에서 변경할 수 있습니다.) • 더 이상 필요하지 않기 때문에 전체 열을 제거하려면 전체 데이터 마이그레이션을 사용하는 것이 좋습니다.
파티션 전략을 변경하면 테이블이 이점을 얻을 수 있나요?	• Iceberg의 파티셔닝 접근 방식이 요구 사항을 충족하는 경우(예: 기존 파티션이 그대로 유지되는 동안 새 파티션 레이아웃을 사용하여 새 데이터가 저장됨) 인플레이스 마이그레이션을 고려하세요. • 테이블에 숨겨진 파티션을 사용하려면 전체 데이터 마이그레이션을 고려하세요. 숨겨진 파티션에 대한 자세한 내용은 모범 사례 섹션을 참조하세요.
테이블이 정렬 순서 전략을 추가하거나 변경하면 도움이 됩니까?	• 데이터의 정렬 순서를 추가하거나 변경하려면 데이터 세트를 다시 작성해야 합니다. 이 경우 전체 데이터 마이그레이션을 사용하는 것이 좋습니다. • 모든 테이블 파티션을 다시 작성하는 데 엄청난 비용이 드는 대형 테이블의 경우, 가장 자주 액세스하는 파티션에 대해 인플레이스 마이그레이션을 사용하고 압축(정렬이 활성화된 상태)을 실행하는 것이 좋습니다.
테이블에 작은 파일이 많이 있습니까?	• 작은 파일을 더 큰 파일로 병합하려면 데이터 세트를 다시 작성해야 합니다. 이 경우 전체 데이터 마이그레이션을 사용하는 것이 좋습니다. • 모든 테이블 파티션을 다시 작성하는 데 엄청난 비용이 드는 대형 테이블의 경우, 가장 자주 액세스하는 파티션에 대해 인플레이스 마이그레이션을 사용하고 압축(정렬이 활성화된 상태)을 실행하는 것이 좋습니다.

마이그레이션 옵션 요약

이 표에는 각 마이그레이션 옵션의 주요 특성과 고려 사항이 요약되어 있습니다.

Feature	현재 위치 마이그레이션 스냅샷	현재 위치 마이그레이션 마이그레이션	전체 데이터 마이그레이션 CTAS 또는 (CREATE TABLE + INSERT)
마이그레이션 프로세스의 일환으로 데이터 레이아웃 개선			
• 데이터 재정렬	 아니요	 아니요	 예
• 파티셔닝 변경(예: Iceber 숨겨진 파티셔닝 사용)	 아니요	 아니요	 예
• 테이블 스키	 아니요	 아니요	 예

Feature	현재 위치 마이그레이션 스냅샷	현재 위치 마이그레이션 마이그레이션	전체 데이터 마이그레이션 CTAS 또는 (CREATE TABLE + INSERT)
마 변경			
• 파일 크기 최적화	 아니요	 아니요	 예
• 데이터를 추가하기 전에 기존 데이터의 스키마를 검증합니다.	 아니요	 아니요	 예
지원되는 파일 형식	Parquet, Avro, ORC	Parquet, Avro, ORC	Parquet, Avro, ORC, JSON, CSV

Feature	현재 위치 마이그레이션 스냅샷	현재 위치 마이그레이션 마이그레이션	전체 데이터 마이그레이션 CTAS 또는 (CREATE TABLE + INSERT)
Iceberg 테이블로 소스 테이블 교체	 아니요 (가 새 테이블을 생성 하지만 추가 단계를 통해 소스 테이블을 교체할 수 있음)	 예 (백업 테이블을 생성하고 소스 테이블을 Iceberg 테이블로 대체)	 아니요 (새 테이블 생성)
소스 테이블 영향			
• Iceberg 테이블의 파일 삭제 작업 (`expireSnapshots` 작업, 제거와 함께 테이블 삭제)	소스 테이블을 손상시킵니다.	백업 테이블을 손상시킵니다.	안전하고 영향을 받지 않는 소스

Feature	현재 위치 마이그레이션 스냅샷	현재 위치 마이그레이션 마이그레이션	전체 데이터 마이그레이션 CTAS 또는 (CREATE TABLE + INSERT)
Iceberg 테이블 영향			
• 소스 테이블 파일이 제거될 경우의 영향	Iceberg 테이블 손상	Iceberg 테이블 손상	Iceberg 테이블에 미치는 영향 없음
• 소스 테이블 위치에 새 파일이 추가될 경우의 영향	새 테이블에 표시되지 않음 (파티션을 와 통합해야 함add_files)	새 테이블에 표시되지 않음 (파티션을와 통합해야 함add_files)	새 테이블에 표시되지 않음 (INSERT INTO새 테이블 필요)
비용	낮음	낮음	더 높음(전체 데이터 재작성)

Feature	현재 위치 마이그레이션 스냅샷	현재 위치 마이그레이션 마이그레이션	전체 데이터 마이그레이션 CTAS 또는 (CREATE TABLE + INSERT)
마이그레이션 속도	빠른	빠른	느림
Amazon S3 Tables로 마이그레이션하는 데 사용할 수 있습니다.	 아니요	 아니요	 예
수동 DDL 필요	 아니요 (스키마 및 파티션은 소스 테이블에서 복사됨)	 아니요 (스키마 및 파티션은 소스 테이블에서 복사됨)	CTAS를 사용하는 경우 파티셔닝만 지정하면 됩니다.

Feature	현재 위치 마이그레이션 스냅샷	현재 위치 마이그레이션 마이그레이션	전체 데이터 마이그레이션 CTAS 또는 (CREATE TABLE + INSERT)
최적 사용	데이터를 다시 작성하지 않고도 빠르게 마이그레이션할 수 있으므로 테스트 또는 점진적 전환을 위해 Hive와 Iceberg를 side-by-side 사용할 수 있습니다.	즉각적인 전환이 허용되는 경우 데이터를 다시 작성하지 않고 Hive 테이블을 제자리에 교체하세요.	데이터 재작성을 통한 전체 Iceberg 최적화. 파티션 또는 스키마를 재설계하거나 레이아웃 및 성능을 개선할 때 적합합니다. 가능하면 항상 권장됩니다.

Apache Iceberg 워크로드 최적화 모범 사례

Iceberg는 데이터 레이크 관리를 간소화하고 워크로드 성능을 향상하도록 설계된 테이블 형식입니다. 사용 사례마다 비용, 읽기 성능, 쓰기 성능 또는 데이터 보존과 같은 다양한 측면을 우선시할 수 있으므로 Iceberg는 이러한 장단점을 관리하기 위한 구성 옵션을 제공합니다. 이 섹션에서는 요구 사항에 맞게 Iceberg 워크로드를 최적화하고 미세 조정하기 위한 인사이트를 제공합니다.

주제

일반 모범 사례

사용 사례에 관계없이에서 Apache Iceberg를 사용하는 경우 다음 일반 모범 사례를 따르는 AWS것이 좋습니다.

- Iceberg 형식 버전 2를 사용합니다.

  Athena는 기본적으로 Iceberg 형식 버전 2를 사용합니다.

  Amazon EMR 또는에서 Spark AWS Glue 를 사용하여 Iceberg 테이블을 생성하는 경우 Iceberg 설명서에 설명된 대로 형식 버전을 지정합니다.
- 를 데이터 카탈로그 AWS Glue Data Catalog 로 사용합니다.

  Athena는 AWS Glue Data Catalog 기본적으로를 사용합니다.

  Amazon EMR 또는에서 Spark AWS Glue 를 사용하여 Iceberg로 작업하는 경우 Spark 세션에 다음 구성을 추가하여를 사용합니다 AWS Glue Data Catalog. 자세한 내용은이 가이드 앞부분의 Iceberg에 대한 Spark 구성 AWS Glue 섹션을 참조하세요.

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

- 를 잠금 관리자 AWS Glue Data Catalog 로 사용합니다.

  Athena는 기본적으로 Iceberg 테이블에 대해를 잠금 관리자 AWS Glue Data Catalog 로 사용합니다.

  Amazon EMR 또는에서 Spark AWS Glue 를 사용하여 Iceberg로 작업하는 경우를 잠금 관리자 AWS Glue Data Catalog 로 사용하도록 Spark 세션 구성을 구성해야 합니다. 자세한 내용은 Iceberg 설명서의 낙관적 잠금을 참조하세요.

- Zstandard(ZSTD) 압축을 사용합니다.

  Iceberg의 기본 압축 코덱은 gzip이며 테이블 속성를 사용하여 수정할 수 있습니다`write.<file_type>.compression-codec`. Athena는 이미 Iceberg 테이블의 기본 압축 코덱으로 ZSTD를 사용합니다.

  일반적으로 ZSTD 압축 코덱을 사용하는 것이 좋습니다. GZIP과 Snappy 간의 균형을 맞추고 압축 비율을 손상시키지 않으면서 우수한 읽기/쓰기 성능을 제공하기 때문입니다. 또한 필요에 맞게 압축 수준을 조정할 수 있습니다. 자세한 내용은 Athena 설명서의 Athena의 ZSTD 압축 수준을 참조하세요.

  Snappy는 최상의 전체 읽기 및 쓰기 성능을 제공할 수 있지만 GZIP 및 ZSTD보다 압축률이 낮습니다. Amazon S3에 더 큰 데이터 볼륨을 저장하더라도 성능에 우선순위를 두는 경우 Snappy가 최적의 선택일 수 있습니다.

읽기 성능 최적화

이 섹션에서는 엔진과 관계없이 읽기 성능을 최적화하기 위해 조정할 수 있는 테이블 속성에 대해 설명합니다.

분할

Hive 테이블과 마찬가지로 Iceberg는 파티션을 인덱싱의 기본 계층으로 사용하여 불필요한 메타데이터 파일과 데이터 파일을 읽지 않도록 합니다. 또한 쿼리 계획을 더욱 개선하기 위해 열 통계를 인덱싱의 보조 계층으로 고려하여 전체 실행 시간을 개선합니다.

데이터 파티셔닝

Iceberg 테이블을 쿼리할 때 스캔되는 데이터의 양을 줄이려면 예상 읽기 패턴과 일치하는 균형 잡힌 파티션 전략을 선택합니다.

- 쿼리에 자주 사용되는 열을 식별합니다. 이들은 이상적인 파티셔닝 후보입니다. 예를 들어 일반적으로 특정 날짜의 데이터를 쿼리하는 경우 파티션 열의 자연스러운 예는 날짜 열입니다.
- 파티션 수가 너무 많지 않도록 하려면 카디널리티가 낮은 파티션 열을 선택합니다. 파티션이 너무 많으면 테이블의 파일 수가 증가하여 쿼리 성능에 부정적인 영향을 미칠 수 있습니다. 일반적으로 "파티션이 너무 많음"은 대다수 파티션의 데이터 크기가에서 설정한 값의 2~5배 미만인 시나리오로 정의할 수 있습니다`target-file-size-bytes`.

> **Note**
>
> 일반적으로 카디널리티가 높은 열(예: 수천 개의 값을 가질 수 있는 `id` 열)에서 필터를 사용하여 쿼리하는 경우 다음 섹션에 설명된 대로 버킷 변환과 함께 Iceberg의 숨겨진 파티셔닝 기능을 사용합니다.

숨겨진 파티셔닝 사용

쿼리가 일반적으로 테이블 열의 파생물을 기준으로 필터링하는 경우 파티션으로 사용할 새 열을 명시적으로 생성하는 대신 숨겨진 파티션을 사용합니다. 이 기능에 대한 자세한 내용은 Iceberg 설명서를 참조하세요.

예를 들어 타임스탬프 열이 있는 데이터 세트(예: `2023-01-01 09:00:00`)에서는 구문 분석된 날짜(예: `2023-01-01`)로 새 열을 생성하는 대신 파티션 변환을 사용하여 타임스탬프에서 날짜 부분을 추출하고 즉시 이러한 파티션을 생성합니다.

숨겨진 파티셔닝의 가장 일반적인 사용 사례는 다음과 같습니다.

- 데이터에 타임스탬프 열이 있는 경우 날짜 또는 시간에 대한 파티셔닝. Iceberg는 타임스탬프의 날짜 또는 시간 부분을 추출하기 위해 여러 변환을 제공합니다.
- 분할 열의 카디널리티가 높아 파티션이 너무 많은 경우 열의 해시 함수에서 분할합니다. Iceberg의 버킷 변환은 파티셔닝 열에서 해시 함수를 사용하여 여러 파티션 값을 더 적은 수의 숨겨진(버킷) 파티션으로 그룹화합니다.

사용 가능한 모든 파티션 변환에 대한 개요는 Iceberg 설명서의 파티션 변환을 참조하세요.

숨겨진 파티셔닝에 사용되는 열은 `year()` 및와 같은 일반 SQL 함수를 사용하여 쿼리 조건자의 일부가 될 수 있습니다`month()`. 조건자는 `BETWEEN` 및와 같은 연산자와 결합할 수도 있습니다`AND`.

> **Note**
>
> Iceberg는와 같이 다른 데이터 유형을 생성하는 함수에 대해 파티션 정리를 수행할 수 없습니다`substring(event_time, 1, 10) = '2022-01-01'`.

파티션 진화 사용

기존 파티션 전략이 최적이 아닌 경우 Iceberg의 파티션 진화를 사용합니다. 예를 들어 너무 작은(각각 몇 메가바이트) 시간당 파티션을 선택하는 경우 일별 또는 월별 파티션으로 전환하는 것이 좋습니다.

이 접근 방식은 테이블에 대한 최상의 파티션 전략이 처음에는 명확하지 않고 더 많은 인사이트를 얻으면서 파티셔닝 전략을 구체화하려는 경우에 사용할 수 있습니다. 파티션 진화의 또 다른 효과적인 사용은 데이터 볼륨이 변경되고 현재 파티셔닝 전략이 시간이 지남에 따라 덜 효과적이 되는 경우입니다.

파티션을 발전시키는 방법에 대한 지침은 Iceberg 설명서의 ALTER TABLE SQL 확장을 참조하세요.

파일 크기 조정

쿼리 성능을 최적화하려면 테이블의 작은 파일 수를 최소화해야 합니다. 쿼리 성능을 높이려면 일반적으로 Parquet 및 ORC 파일을 100MB보다 크게 유지하는 것이 좋습니다.

파일 크기는 Iceberg 테이블의 쿼리 계획에도 영향을 미칩니다. 테이블의 파일 수가 증가하면 메타데이터 파일의 크기도 증가합니다. 메타데이터 파일이 클수록 쿼리 계획이 느려질 수 있습니다. 따라서 테이블 크기가 커지면 파일 크기를 늘려 메타데이터의 지수 확장을 완화 합니다.

다음 모범 사례를 사용하여 Iceberg 테이블에 적절한 크기의 파일을 생성합니다.

대상 파일 및 행 그룹 크기 설정

Iceberg는 데이터 파일 레이아웃을 튜닝하기 위해 다음과 같은 주요 구성 파라미터를 제공합니다. 이러한 파라미터를 사용하여 대상 파일 크기와 행 그룹 또는 스트라이크 크기를 설정하는 것이 좋습니다.

파라미터	기본값	Comment
`write.target-file-size-bytes`	512MB	이 파라미터는 Iceberg가 생성할 최대 파일 크기를 지정합니다. 그러나 특정 파일은이 제한보다 작은 크기로 작성될 수 있습니다.

파라미터	기본값	Comment
`write.parquet.row-group-size-bytes`	128MB	Parquet과 ORC 모두 데이터를 청크로 저장하므로 엔진이 일부 작업에 대해 전체 파일을 읽지 않아도 됩니다.
`write.orc.stripe-size-bytes`	64MB	
`write.distribution-mode`	없음, Iceberg 버전 1.1 이하의 경우 해시, Iceberg 버전 1.2부터	Iceberg는 스토리지에 쓰기 전에 작업 간에 데이터를 정렬하도록 Spark에 요청합니다.

- 예상 테이블 크기에 따라 다음 일반 지침을 따릅니다.
  - 작은 테이블(최대 몇 기가바이트) - 대상 파일 크기를 128MB로 줄입니다. 또한 행 그룹 또는 스트라이프 크기를 줄입니다(예: 8MB 또는 16MB로).
  - 중간에서 큰 테이블(몇 기가바이트에서 수백 기가바이트) - 기본값은 이러한 테이블의 좋은 시작점입니다. 쿼리가 매우 선택적인 경우 행 그룹 또는 스트라이프 크기를 조정합니다(예: 16MB).
  - 매우 큰 테이블(수백 기가바이트 또는 테라바이트) - 대상 파일 크기를 1,024MB 이상으로 늘리고 쿼리가 일반적으로 대용량 데이터 세트를 가져오는 경우 행 그룹 또는 스트라이프 크기를 늘리는 것이 좋습니다.

- Iceberg 테이블에 쓰는 Spark 애플리케이션이 적절한 크기의 파일을 생성하도록 하려면 `write.distribution-mode` 속성을 `hash` 또는 로 설정합니다`range`. 이러한 모드 간의 차이점에 대한 자세한 설명은 Iceberg 설명서의 배포 모드 작성을 참조하세요.

다음은 일반적인 지침입니다. 테스트를 실행하여 특정 테이블 및 워크로드에 가장 적합한 값을 식별하는 것이 좋습니다.

정기적인 압축 실행

이전 테이블의 구성은 쓰기 작업에서 생성할 수 있는 최대 파일 크기를 설정하지만 파일의 크기가 해당 크기인지 보장하지는 않습니다. 적절한 파일 크기를 보장하려면 정기적으로 압축을 실행하여 작은 파

일을 더 큰 파일로 결합합니다. 압축 실행에 대한 자세한 지침은이 가이드 뒷부분의 Iceberg 압축을 참조하세요.

열 통계 최적화

Iceberg는 열 통계를 사용하여 파일 정리를 수행하므로 쿼리에서 스캔하는 데이터의 양을 줄여 쿼리 성능이 향상됩니다. 열 통계를 활용하려면 Iceberg가 쿼리 필터에 자주 사용되는 모든 열에 대한 통계를 수집해야 합니다.

기본적으로 Iceberg는 테이블 속성에 정의된 대로 각 테이블의 처음 100개 열에 대한 통계만 수집합니다`write.metadata.metrics.max-inferred-column-defaults`. 테이블에 100개 이상의 열이 있고 쿼리가 처음 100개 열 외부의 열을 자주 참조하는 경우(예: 열 132를 필터링하는 쿼리가 있을 수 있음) Iceberg가 해당 열에 대한 통계를 수집하는지 확인합니다. 이를 달성하기 위한 두 가지 옵션이 있습니다.

- Iceberg 테이블을 생성할 때 쿼리에 필요한 열이에서 설정한 열 범위`write.metadata.metrics.max-inferred-column-defaults`(기본값은 100) 내에 있도록 열을 재정렬합니다.

  참고: 100개 열에 대한 통계가 필요하지 않은 경우 `write.metadata.metrics.max-inferred-column-defaults` 구성을 원하는 값(예: 20개)으로 조정하고 쿼리를 읽고 쓰는 데 필요한 열이 데이터 세트 왼쪽의 처음 20개 열에 속하도록 열을 재정렬할 수 있습니다.
- 쿼리 필터에 몇 개의 열만 사용하는 경우 다음 예제와 같이 지표 수집에 대한 전체 속성을 비활성화하고 통계를 수집할 개별 열을 선택적으로 선택할 수 있습니다.

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

참고: 열 통계는 해당 열에서 데이터를 정렬할 때 가장 효과적입니다. 자세한 내용은이 가이드 뒷부분의 정렬 순서 설정 섹션을 참조하세요.

올바른 업데이트 전략 선택

느린 copy-on-write 전략을 사용하여 읽기 성능을 최적화합니다. Iceberg에서 사용하는 기본 전략입니다.

파일이 읽기 최적화 방식으로 스토리지에 직접 기록되므로 Copy-on-write를 통해 읽기 성능이 향상됩니다. 그러나 merge-on-read에 비해 각 쓰기 작업은 더 오래 걸리고 더 많은 컴퓨팅 리소스를 소비합니

다. 이는 읽기 및 쓰기 지연 시간 간의 클래식 절충점을 제공합니다. 일반적으로 copy-on-write는 대부분의 업데이트가 동일한 테이블 파티션에 공동 배치되는 사용 사례에 적합합니다(예: 일일 배치 로드의 경우).

Copy-on-write 구성(`write.update.mode`, `write.delete.mode`및 `write.merge.mode`)은 테이블 수준에서 설정하거나 애플리케이션 측에서 독립적으로 설정할 수 있습니다.

ZSTD 압축 사용

테이블 속성를 사용하여 Iceberg에서 사용하는 압축 코덱을 수정할 수 있습니다`write.<file_type>.compression-codec`. ZSTD 압축 코덱을 사용하여 테이블의 전반적인 성능을 개선하는 것이 좋습니다.

기본적으로 Iceberg 버전 1.3 이하에서는 ZSTD에 비해 읽기/쓰기 성능이 느린 GZIP 압축을 사용합니다.

참고: 일부 엔진은 다른 기본값을 사용할 수 있습니다. 이는 Athena 또는 Amazon EMR 버전 7.x로 생성된 Iceberg 테이블의 경우입니다.

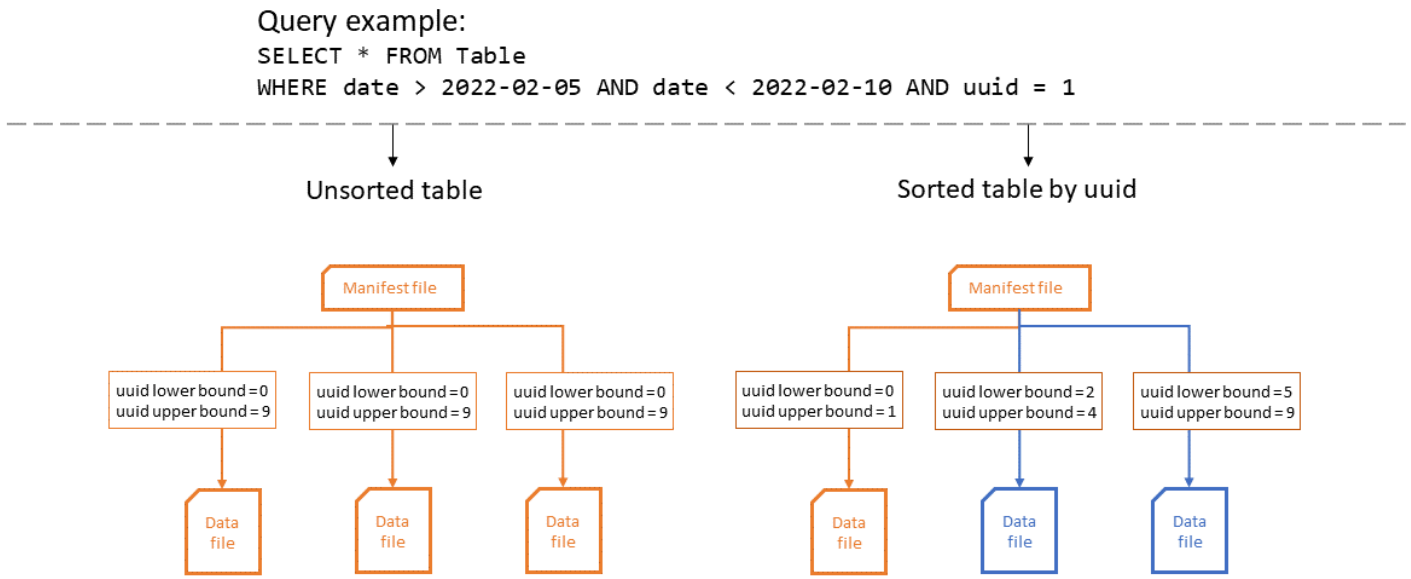
정렬 순서 설정

Iceberg 테이블의 읽기 성능을 개선하려면 쿼리 필터에 자주 사용되는 하나 이상의 열을 기준으로 테이블을 정렬하는 것이 좋습니다. Iceberg의 열 통계와 함께 정렬하면 파일 정리가 훨씬 더 효율적이 되어 읽기 작업이 빨라질 수 있습니다. 또한 정렬은 쿼리 필터에서 정렬 열을 사용하는 쿼리에 대한 Amazon S3 요청 수를 줄입니다.

Spark에서 데이터 정의 언어(DDL) 문을 실행하여 테이블 수준에서 계층적 정렬 순서를 설정할 수 있습니다. 사용 가능한 옵션은 Iceberg 설명서를 참조하세요. 정렬 순서를 설정한 후 라이터는 Iceberg 테이블의 후속 데이터 쓰기 작업에이 정렬을 적용합니다.

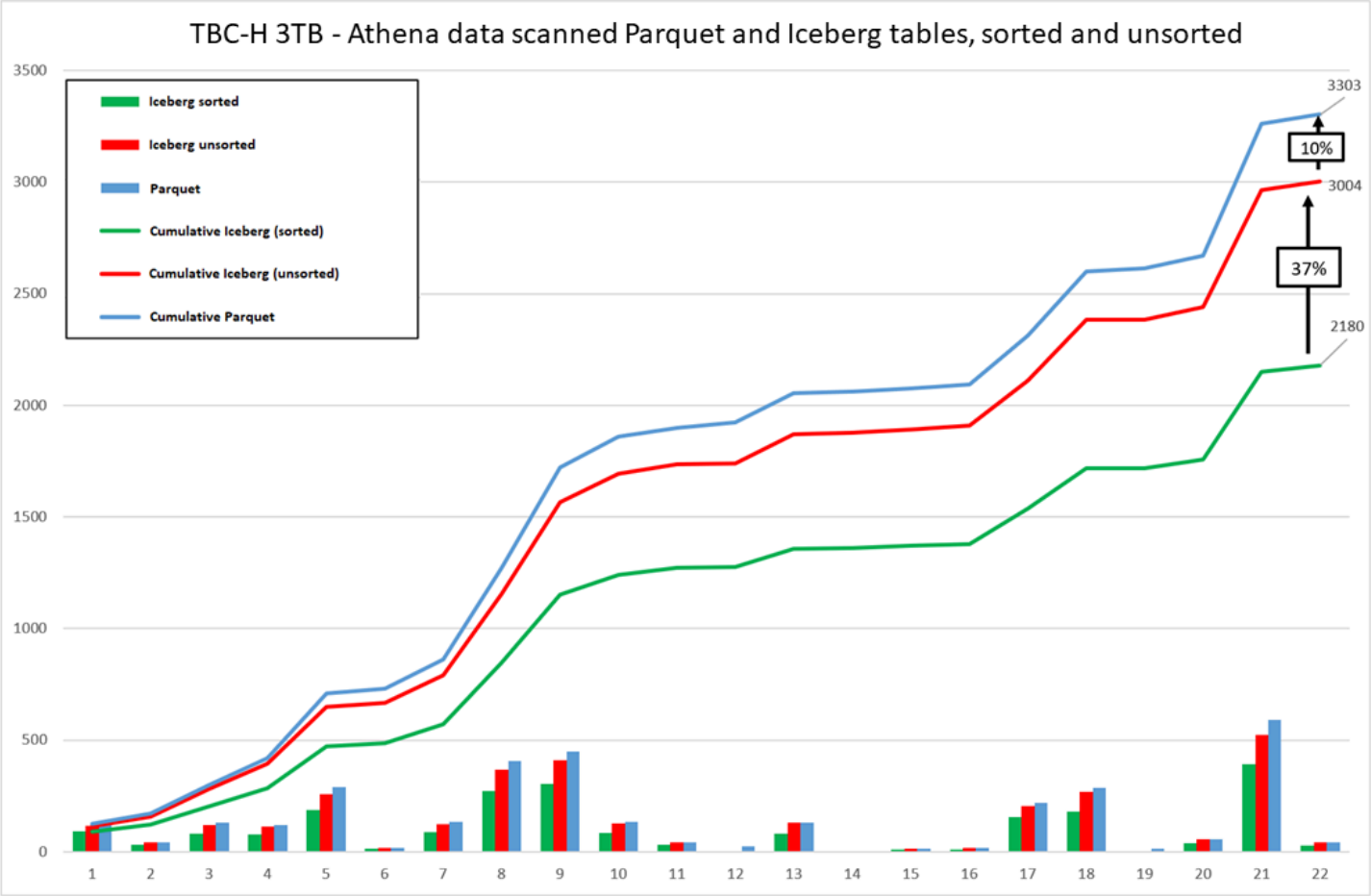
예를 들어 대부분의 쿼리가를 기준으로 필터링하는 날짜(`yyyy-mm-dd`)로 분할된 테이블에서 DDL 옵션을 사용하여 Spark가 겹치지 않는 범위의 파일을 쓰도록 `Write Distributed By Partition Locally Ordered`할 `uuid`수 있습니다.

다음 다이어그램은 테이블이 정렬될 때 열 통계의 효율성이 어떻게 향상되는지 보여줍니다. 이 예제에서 정렬된 테이블은 단일 파일만 열어야 하며 Iceberg의 파티션과 파일의 이점을 최대한 활용할 수 있습니다. 정렬되지 않은 테이블에서는 모든 데이터 파일에가 존재할 `uuid` 수 있으므로 쿼리는 모든 데이터 파일을 열어야 합니다.

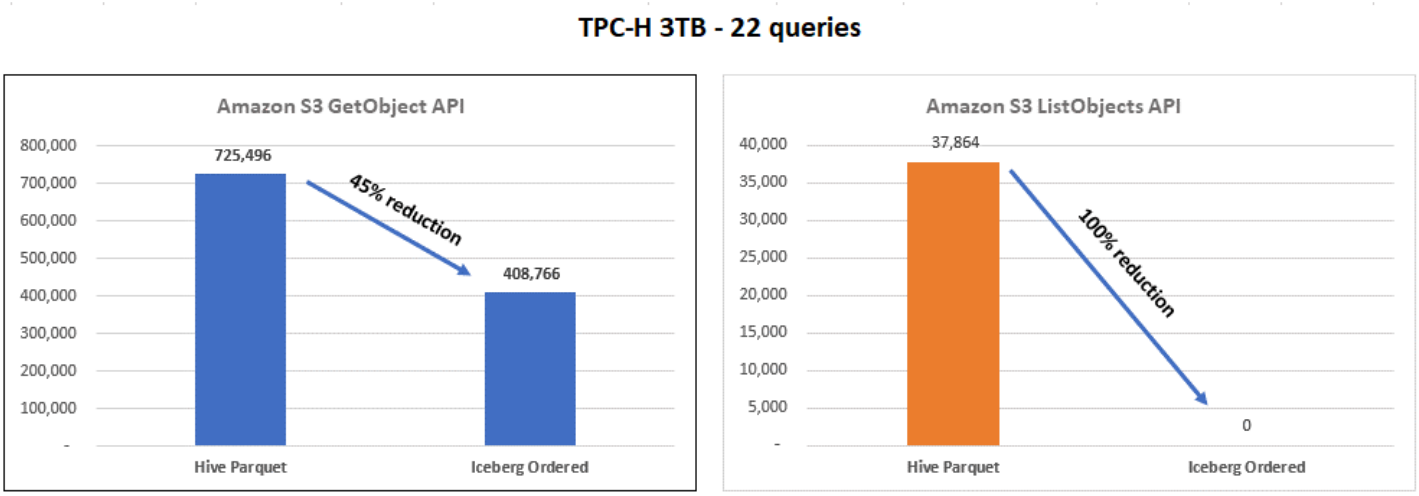


정렬 순서를 변경해도 기존 데이터 파일에는 영향을 주지 않습니다. Iceberg 압축을 사용하여 정렬 순서를 적용할 수 있습니다.

다음 그래프와 같이 Iceberg 정렬 테이블을 사용하면 워크로드 비용이 절감될 수 있습니다.



이 그래프에는 Iceberg 정렬 테이블과 비교하여 Hive(Parquet) 테이블에 대한 TPC-H 벤치마크를 실행한 결과가 요약되어 있습니다. 그러나 결과는 다른 데이터 세트 또는 워크로드에 따라 다를 수 있습니다.



쓰기 성능 최적화

이 섹션에서는 엔진과 관계없이 Iceberg 테이블의 쓰기 성능을 최적화하기 위해 조정할 수 있는 테이블 속성에 대해 설명합니다.

테이블 배포 모드 설정

Iceberg는 Spark 작업 전체에 쓰기 데이터가 배포되는 방식을 정의하는 여러 쓰기 배포 모드를 제공합니다. 사용 가능한 모드에 대한 개요는 Iceberg 설명서의 배포 모드 작성을 참조하세요.

특히 스트리밍 워크로드에서 쓰기 속도를 우선시하는 사용 사례의 경우를 `write.distribution-mode`로 설정합니다`none`. 이렇게 하면 Iceberg가 추가 Spark 셔플링을 요청하지 않고 Spark 작업에서 사용할 수 있게 되면 데이터가 기록됩니다. 이 모드는 Spark Structured Streaming 애플리케이션에 특히 적합합니다.

> **Note**
>
> 쓰기 배포 모드를 none 로 설정하면 작은 파일이 많이 생성되어 읽기 성능이 저하되는 경향이 있습니다. 쿼리 성능을 위해 이러한 작은 파일을 적절한 크기의 파일로 통합하려면 정기적으로 압축하는 것이 좋습니다.

올바른 업데이트 전략 선택

최신 데이터에 대한 느린 읽기 작업이 사용 사례에 적합한 경우 읽기 merge-on-read 전략을 사용하여 쓰기 성능을 최적화합니다.

merge-on-read을 사용하는 경우 Iceberg는 업데이트를 쓰고 스토리지에 별도의 작은 파일로 삭제합니다. 테이블을 읽을 때 리더는 이러한 변경 사항을 기본 파일과 병합하여 데이터의 최신 보기를 반환해야 합니다. 이로 인해 읽기 작업에 성능 저하가 발생하지만 업데이트 및 삭제 쓰기 속도가 빨라집니다. 일반적으로 merge-on-read은 업데이트가 있는 스트리밍 워크로드 또는 여러 테이블 파티션에 분산된 업데이트가 거의 없는 작업에 적합합니다.

테이블 수준에서 또는 애플리케이션 측에서 merge-on-read 구성(`write.update.mode`, `write.delete.mode`, 및 `write.merge.mode`)을 독립적으로 설정할 수 있습니다.

merge-on-read 사용하려면 시간이 지남에 따라 읽기 성능이 저하되지 않도록 정기적인 압축을 실행해야 합니다. 압축은 기존 데이터 파일과 업데이트 및 삭제를 조정하여 새 데이터 파일 세트를 생

성하므로 읽기 측에서 발생하는 성능 저하를 제거합니다. 기본적으로 Iceberg의 압축은 `delete-file-threshold` 속성의 기본값을 더 작은 값으로 변경하지 않는 한 삭제 파일을 병합하지 않습니다 (Iceberg 설명서 참조). 압축에 대해 자세히 알아보려면이 가이드 뒷부분의 Iceberg 압축 섹션을 참조하세요.

올바른 파일 형식 선택

Iceberg는 Parquet, ORC 및 Avro 형식의 데이터 쓰기를 지원합니다. Parquet은 기본 형식입니다. Parquet 및 ORC는 우수한 읽기 성능을 제공하지만 일반적으로 쓰기 속도가 느린 열 형식입니다. 이는 읽기 및 쓰기 성능 간의 일반적인 장단점을 나타냅니다.

스트리밍 워크로드와 같이 사용 사례에 쓰기 속도가 중요한 경우 `Avro`라이터의 옵션에서를 로 설정하여 Avro 형식으로 작성하는 `write-format` 것이 좋습니다. Avro는 행 기반 형식이므로 읽기 성능이 느려지는 대신 더 빠른 쓰기 시간을 제공합니다.

읽기 성능을 개선하려면 정기적인 압축을 실행하여 작은 Avro 파일을 병합하고 더 큰 Parquet 파일로 변환합니다. 압축 프로세스의 결과는 `write.format.default` 테이블 설정에 의해 관리됩니다. Iceberg의 기본 형식은 Parquet이므로 Avro로 작성한 다음 압축을 실행하면 Iceberg가 Avro 파일을 Parquet 파일로 변환합니다. 다음은 그 예입니다.

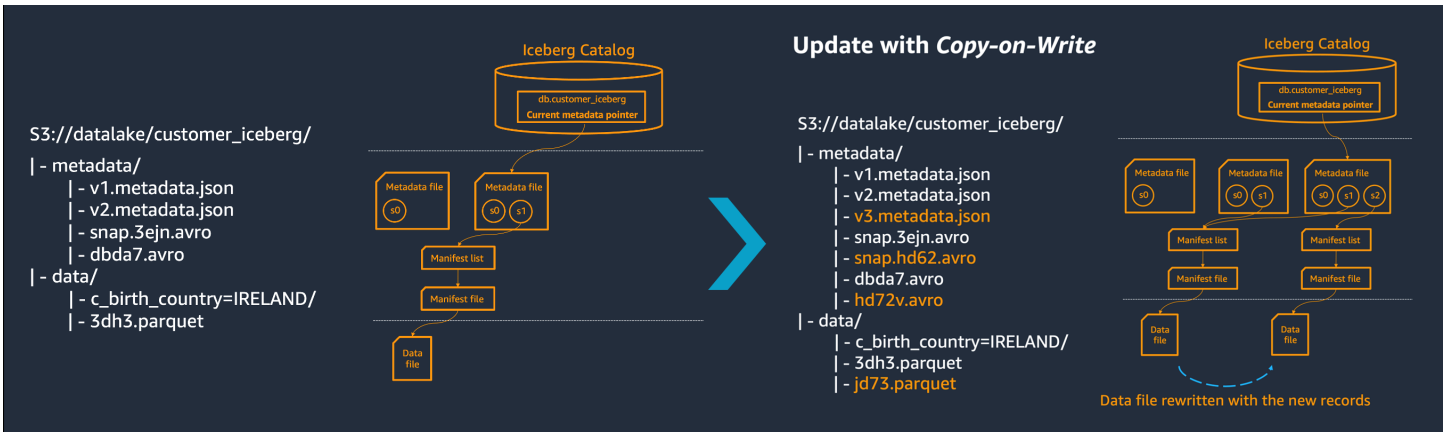
```
spark.sql(f"""
    CREATE TABLE IF NOT EXISTS glue_catalog.{DB_NAME}.{TABLE_NAME} (
        Col_1 float,
        <<<…other columns…>>
        ts timestamp)
    USING iceberg
    PARTITIONED BY (days(ts))
    OPTIONS (
      'format-version'='2',
      write.format.default'=parquet)
""")

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()
```

스토리지 최적화

Iceberg 테이블에서 데이터를 업데이트하거나 삭제하면 다음 다이어그램과 같이 데이터의 복사본 수가 증가합니다. 압축을 실행하는 경우에도 마찬가지입니다. 즉, Amazon S3의 데이터 복사본 수가 증가합니다. Iceberg는 모든 테이블의 기본 파일을 변경할 수 없는 것으로 취급하기 때문입니다.



이 섹션의 모범 사례에 따라 스토리지 비용을 관리합니다.

S3 Intelligent-Tiering 활성화

Amazon S3 Intelligent-Tiering 스토리지 클래스를 사용하면 액세스 패턴이 변경될 때 가장 비용 효율적인 액세스 계층으로 데이터를 자동으로 이동할 수 있습니다. 이 옵션은 운영 오버헤드나 성능에 영향을 주지 않습니다.

참고: Iceberg 테이블을 사용한 S3 Intelligent-Tiering에서 선택적 계층(예: Archive Access 및 Deep Archive Access)을 사용하지 마세요. 데이터를 아카이브하려면 다음 섹션의 지침을 참조하세요.

Amazon S3 수명 주기 규칙을 사용하여 객체를 S3 Standard-IA 또는 Amazon S3 One Zone-IA와 같은 다른 Amazon S3 스토리지 클래스로 이동하기 위한 자체 규칙을 설정할 수도 있습니다(Amazon S3 설명서의 지원되는 전환 및 관련 제약 조건 참조).

기록 스냅샷 보관 또는 삭제

Iceberg 테이블에 커밋된 모든 트랜잭션(삽입, 업데이트, 병합, 압축)에 대해 테이블의 새 버전 또는 스냅샷이 생성됩니다. 시간이 지남에 따라 Amazon S3의 버전 수와 메타데이터 파일 수가 누적됩니다.

스냅샷 격리, 테이블 롤백, 시간 이동 쿼리와 같은 기능에는 테이블의 스냅샷을 유지해야 합니다. 그러나 스토리지 비용은 보존하는 버전 수에 따라 증가합니다.

다음 표에서는 데이터 보존 요구 사항에 따라 비용을 관리하기 위해 구현할 수 있는 설계 패턴을 설명합니다.

디자인 패턴	솔루션	사용 사례
이전 스냅샷 삭제	• Athena의 VACUUM 문을 사용하여 이전 스냅샷을 제거합니다. 이 작업에는 컴퓨팅 비용이 발생하지 않습니다. • 또는 Amazon EMR에서 Spark를 사용하거나 스냅샷 AWS Glue 을 제거할 수 있습니다. 자세한 내용은 Iceberg 설명서의 expire_snapshots를 참조하세요.	이 접근 방식은 스토리지 비용을 줄이는 데 더 이상 필요하지 않은 스냅샷을 삭제합니다. 데이터 보존 요구 사항에 따라 보존해야 하는 스냅샷 수 또는 기간을 구성할 수 있습니다. 이 옵션은 스냅샷을 하드 삭제합니다. 만료된 스냅샷으로 롤백하거나 시간 이동을 할 수 없습니다.
특정 스냅샷에 대한 보존 정책 설정	1. 태그를 사용하여 특정 스냅샷을 표시하고 Iceberg에서 보존 정책을 정의합니다. 자세한 내용은 Iceberg 설명서의 기록 태그를 참조하세요. 예를 들어 Amazon EMR의 Spark에서 다음 SQL 문을 사용하여 1년 동안 매월 스냅샷 하나를 유지할 수 있습니다. `ALTER TABLE glue_catalog.db.table CREATE TAG 'EOM-01' AS OF VERSION 30 RETAIN 365 DAYS` 2. Amazon EMR 또는에서 Spark AWS Glue 를 사용하여 태그가 지정되지 않은 나	이 패턴은 과거의 특정 시점에서 테이블의 상태를 표시해야 하는 비즈니스 또는 법적 요구 사항을 준수하는 데 유용합니다. 태그가 지정된 특정 스냅샷에 보존 정책을 배치하면 생성된 다른(태그가 지정되지 않은) 스냅샷을 제거할 수 있습니다. 이렇게 하면 생성된 모든 스냅샷을 유지하지 않고도 데이터 보존 요구 사항을 충족할 수 있습니다.

디자인 패턴	솔루션	사용 사례
	머지 중간 스냅샷을 제거합니다.	

디자인 패턴	솔루션	사용 사례
이전 스냅샷 아카이브	1. Amazon S3 태그를 사용하여 Spark로 객체를 표시합니다. (Amazon S3 태그는 Iceberg 태그와 다릅니다. 자세한 내용은 Iceberg 설명서를 참조하세요.) 예제: `spark.sql.catalog.my_catalog.s3.delete-enabled=false and \ spark.sql.catalog.my_catalog.s3.delete.tags.my_key=to_archive` 2. Amazon EMR 또는에서 Spark AWS Glue 를 사용하여 스냅샷을 제거합니다. 예제의 설정을 사용하면이 절차에서는 객체에 태그를 지정하고 Amazon S3에서 객체를 삭제하는 대신 Iceberg 테이블 메타데이터에서 객체를 분리합니다. 3. S3 수명 주기 규칙을 사용하여 로 태그가 지정된 객체`to_archive` 를 S3 Glacier 스토리지 클래스 중 하나로 전환합니다. 4. 아카이브된 데이터를 쿼리하려면: • 아카이브된 객체를 복원합니다(객체가 Amazon Glacier Instant Retrieval 스토리지 클래스로 전환	이 패턴을 사용하면 모든 테이블 버전과 스냅샷을 더 저렴한 비용으로 유지할 수 있습니다. 먼저 해당 버전을 새 테이블로 복원하지 않으면 시간을 이동하거나 아카이브된 스냅샷으로 롤백할 수 없습니다. 이는 일반적으로 감사 목적으로 허용됩니다. 이 접근 방식을 이전 설계 패턴과 결합하여 특정 스냅샷에 대한 보존 정책을 설정할 수 있습니다.

디자인 패턴	솔루션	사용 사례
	된 경우에는이 단계가 필요하지 않음). • Iceberg의 register_table 프로시저를 사용하여 스냅샷을 카탈로그의 테이블로 등록합니다. 자세한 지침은 AWS 블로그 게시물 Amazon S3 데이터 레이크에 구축된 Apache Iceberg 테이블의 운영 효율성 개선을 참조하세요.	

분리된 파일 삭제

특정 상황에서는 트랜잭션을 커밋하기 전에 Iceberg 애플리케이션이 실패할 수 있습니다. 이렇게 하면 Amazon S3에 데이터 파일이 남습니다. 커밋이 없으므로 이러한 파일은 테이블과 연결되지 않으므로 비동기적으로 정리해야 할 수 있습니다.

이러한 삭제를 처리하기 위해 Amazon Athena에서 VACUUM 문을 사용할 수 있습니다. 이 문은 스냅샷을 제거하고 분리된 파일도 삭제합니다. Athena는이 작업의 컴퓨팅 비용을 청구하지 않으므로 매우 비용 효율적입니다. 또한 VACUUM 문을 사용할 때 추가 작업을 예약할 필요가 없습니다.

또는 Amazon EMR에서 Spark를 사용하거나 `remove_orphan_files` 프로시저 AWS Glue 를 실행할 수 있습니다. 이 작업에는 컴퓨팅 비용이 있으며 독립적으로 예약해야 합니다. 자세한 내용은 Iceberg 설명서를 참조하세요.

압축을 사용하여 테이블 유지 관리

Iceberg에는 테이블에 데이터를 쓴 후 테이블 유지 관리 작업을 수행할 수 있는 기능이 포함되어 있습니다. 일부 유지 관리 작업은 메타데이터 파일을 간소화하는 데 중점을 두는 반면, 다른 유지 관리 작업은 쿼리 엔진이 사용자 요청에 응답하는 데 필요한 정보를 효율적으로 찾을 수 있도록 데이터가 파일에 클러스터링되는 방법을 개선합니다. 이 섹션에서는 압축 관련 최적화에 중점을 둡니다.

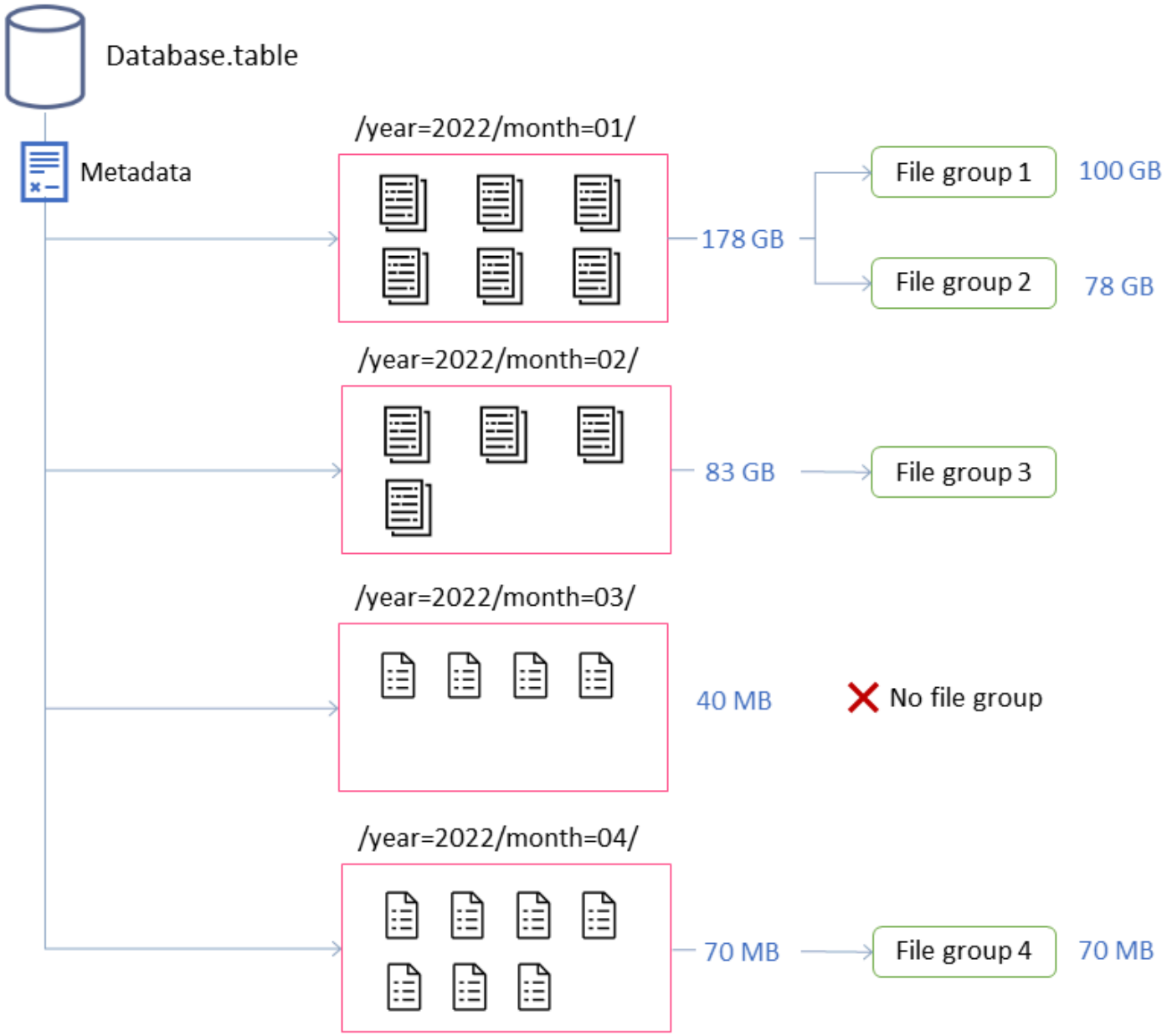
Iceberg 압축

Iceberg에서는 압축을 사용하여 다음 4가지 작업을 수행할 수 있습니다.

- 작은 파일을 일반적으로 크기가 100MB를 초과하는 더 큰 파일로 결합합니다. 이 기술을 빈 패킹이라고 합니다.
- 삭제 파일을 데이터 파일과 병합합니다. 삭제 파일은 merge-on-read 접근 방식을 사용하는 업데이트 또는 삭제에 의해 생성됩니다.
- (Re)쿼리 패턴에 따라 데이터 정렬. 데이터는 정렬 순서 없이 또는 쓰기 및 업데이트에 적합한 정렬 순서로 작성할 수 있습니다.
- 공간 채우기 곡선을 사용하여 데이터를 클러스터링하여 고유한 쿼리 패턴, 특히 z 순서 정렬을 최적화합니다.

에서 Amazon Athena를 통해 또는 Amazon EMR 또는에서 Spark를 사용하여 Iceberg에 대한 테이블 압축 및 유지 관리 작업을 실행할 AWS수 있습니다 AWS Glue.

rewrite_data_files 프로시저를 사용하여 압축을 실행할 때 여러 개의 노브를 조정하여 압축 동작을 제어할 수 있습니다. 다음 다이어그램은 빈 패킹의 기본 동작을 보여줍니다. 빈 패킹 압축을 이해하는 것은 계층적 정렬 및 Z 순서 정렬 구현을 이해하는 데 중요합니다. 빈 패킹 인터페이스의 확장이며 유사한 방식으로 작동하기 때문입니다. 주요 차이점은 데이터를 정렬하거나 클러스터링하는 데 필요한 추가 단계입니다.



이 예제에서 Iceberg 테이블은 4개의 파티션으로 구성됩니다. 파티션마다 크기가 다르고 파일 수가 다릅니다. Spark 애플리케이션을 시작하여 압축을 실행하면 애플리케이션은 처리할 총 4개의 파일 그룹을 생성합니다. 파일 그룹은 단일 Spark 작업에서 처리할 파일 모음을 나타내는 Iceberg 추상화입니다. 즉, 압축을 실행하는 Spark 애플리케이션은 데이터를 처리하는 4개의 Spark 작업을 생성합니다.

압축 동작 튜닝

다음 키 속성은 압축을 위해 데이터 파일을 선택하는 방법을 제어합니다.

- MAX_FILE_GROUP_SIZE_BYTES는 기본적으로 단일 파일 그룹(Spark 작업)의 데이터 제한을 100GB로 설정합니다. 이 속성은 파티션이 없는 테이블 또는 수백 기가바이트에 걸친 파티션이 있는 테이블에서 특히 중요합니다. 이 제한을 설정하면 작업을 계획하고 진행하면서 클러스터의 리소스 소진을 방지하는 작업을 세분화할 수 있습니다.

  참고: 각 파일 그룹은 별도로 정렬됩니다. 따라서 파티션 수준 정렬을 수행하려면 파티션 크기에 맞게이 제한을 조정해야 합니다.

- MIN_FILE_SIZE_BYTES 또는 MIN_FILE_SIZE_DEFAULT_RATIO는 기본적으로 테이블 수준에서 설정된 대상 파일 크기의 75%로 설정됩니다. 예를 들어 테이블의 대상 크기가 512MB인 경우 384MB보다 작은 모든 파일이 압축될 파일 세트에 포함됩니다.
- MAX_FILE_SIZE_BYTES 또는 MAX_FILE_SIZE_DEFAULT_RATIO는 기본적으로 대상 파일 크기의 180%로 설정됩니다. 최소 파일 크기를 설정하는 두 가지 속성과 마찬가지로 이러한 속성은 압축 작업의 후보 파일을 식별하는 데 사용됩니다.
- MIN_INPUT_FILES는 테이블 파티션 크기가 대상 파일 크기보다 작은 경우 압축할 최소 파일 수를 지정합니다. 이 속성의 값은 파일 수(기본값은 5)를 기준으로 파일을 압축할 가치가 있는지 여부를 결정하는 데 사용됩니다.
- DELETE_FILE_THRESHOLD는 압축에 포함되기 전에 파일에 대한 최소 삭제 작업 수를 지정합니다. 달리 지정하지 않는 한 압축은 삭제 파일을 데이터 파일과 결합하지 않습니다. 이 기능을 활성화하려면이 속성을 사용하여 임계값을 설정해야 합니다. 이 임계값은 개별 데이터 파일에 고유하므로 3으로 설정하면 데이터 파일을 참조하는 삭제 파일이 3개 이상 있는 경우에만 데이터 파일이 다시 작성됩니다.

이러한 속성은 이전 다이어그램에서 파일 그룹의 형성에 대한 인사이트를 제공합니다.

예를 들어 레이블이 지정된 파티션에는 최대 크기 제약 조건인 100GB를 초과하기 때문에 두 개의 파일 그룹이 `month=01` 포함됩니다. 반면 파티션은 100GB 미만이므로 단일 파일 그룹을 `month=02` 포함합니다. `month=03` 파티션이 파일 5개의 기본 최소 입력 파일 요구 사항을 충족하지 않습니다. 따라서 압축되지 않습니다. 마지막으로 `month=04` 파티션에 원하는 크기의 단일 파일을 형성하기에 충분한 데이터가 포함되어 있지 않지만 파티션에 5개 이상의 작은 파일이 포함되어 있기 때문에 파일이 압축됩니다.

Amazon EMR 또는에서 실행되는 Spark에 대해 이러한 파라미터를 설정할 수 있습니다 AWS Glue. Amazon Athena의 경우 접두사)로 시작하는 테이블 속성을 사용하여 유사한 속성을 관리할 수 있습니다`optimize_`.

Amazon EMR 또는에서 Spark로 압축 실행 AWS Glue

이 섹션에서는 Iceberg의 압축 유틸리티를 실행하기 위해 Spark 클러스터의 크기를 적절하게 조정하는 방법을 설명합니다. 다음 예제에서는 Amazon EMR Serverless를 사용하지만 Amazon EMR on EC2 또는 EKS 또는에서 동일한 방법론을 사용할 수 있습니다 AWS Glue.

파일 그룹과 Spark 작업 간의 상관 관계를 활용하여 클러스터 리소스를 계획할 수 있습니다. 파일 그룹당 최대 크기인 100GB를 고려하여 파일 그룹을 순차적으로 처리하려면 다음 Spark 속성을 설정할 수 있습니다.

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

압축 속도를 높이려면 병렬로 압축되는 파일 그룹 수를 늘려 수평적으로 확장할 수 있습니다. 수동 또는 동적 조정을 사용하여 Amazon EMR을 조정할 수도 있습니다.

- 수동 조정(예: 4배)
  - `MAX_CONCURRENT_FILE_GROUP_REWRITES = 4` (계수)
  - `spark.executor.instances = 5` (예제에 사용된 값) x `4` (계수) = `20`
  - `spark.dynamicAllocation.enabled = FALSE`
- 동적 조정
  - `spark.dynamicAllocation.enabled = TRUE` (기본값, 작업 필요 없음)
  - MAX_CONCURRENT_FILE_GROUP_REWRITES = `N` (이 값을 기본적으로 `100`spark.dynamicAllocation.maxExecutors인와 정렬합니다. 예제의 실행기 구성에 따라 를 `20N`으로 설정할 수 있습니다.)

  다음은 클러스터 크기를 조정하는 데 도움이 되는 지침입니다. 하지만 워크로드에 가장 적합한 설정을 찾으려면 Spark 작업의 성능도 모니터링해야 합니다.

Amazon Athena를 사용하여 압축 실행

Athena는 OPTIMIZE 문을 통해 Iceberg의 압축 유틸리티를 관리형 기능으로 구현합니다. 인프라를 평가할 필요 없이이 문을 사용하여 압축을 실행할 수 있습니다.

이 문은 빈 패킹 알고리즘을 사용하여 작은 파일을 더 큰 파일로 그룹화하고 삭제 파일을 기존 데이터 파일과 병합합니다. 계층적 정렬 또는 z 순서 정렬을 사용하여 데이터를 클러스터링하려면 Amazon EMR에서 Spark 또는를 사용합니다 AWS Glue.

테이블 생성 시 `OPTIMIZE` 문에 테이블 속성을 전달하거나 테이블 생성 후 `CREATE TABLE` 문을 사용하여 문의 기본 동작을 변경할 수 `ALTER TABLE` 있습니다. 기본값은 Athena 설명서를 참조하세요.

압축 실행을 위한 권장 사항

사용 사례:	권장 사항
일정에 따라 빈 패킹 압축 실행	• 테이블에 포함된 작은 파일의 수를 모르는 경우 Athena에서 `OPTIMIZE` 문을 사용합니다. Athena 요금 모델은 스캔한 데이터를 기반으로 하므로 압축할 파일이 없는 경우 이러한 작업과 관련된 비용은 없습니다. Athena 테이블에서 제한 시간이 발생하지 않도록 하려면 per-table-partition `OPTIMIZE`를 실행합니다. • 대용량의 작은 파일이 압축될 것으로 예상되는 경우 Amazon EMR 또는를 동적 조정과 AWS Glue 함께 사용합니다.
이벤트를 기반으로 빈 패킹 압축 실행	• 대용량의 작은 파일이 압축될 것으로 예상되는 경우 Amazon EMR 또는를 동적 조정과 AWS Glue 함께 사용합니다.
압축을 실행하여 데이터 정렬	• 정렬은 비용이 많이 들고 디스크에 데이터를 유출해야 할 수 AWS Glue있으므로 Amazon EMR 또는를 사용합니다.
압축을 실행하여 z 순서 정렬을 사용하여 데이터 클러스터링	• z 순서 정렬은 매우 비용이 많이 들고 데이터를 디스크에 유출해야 할 수 AWS Glue있으므로 Amazon EMR 또는를 사용합니다.
지연 도착 데이터로 인해 다른 애플리케이션에서 업데이트할 수 있는 파티션에서 압축 실행	• Amazon EMR 또는를 사용합니다 AWS Glue. Iceberg PARTIAL_PROGRESS_ENABLED 속성을 활성화합니다. 이 옵션을 사용하면 Iceberg는 압축 출력을 여러 커밋으로 분할합

사용 사례:	권장 사항
	니다. 충돌이 있는 경우(즉, 압축이 실행되는 동안 데이터 파일이 업데이트되는 경우)이 설정은 영향을 받는 파일이 포함된 커밋으로 제한하여 재시도 비용을 줄입니다. 그렇지 않으면 모든 파일을 다시 압축해야 할 수 있습니다.
콜드 파티션(더 이상 활성 쓰기를 수신하지 않는 데이터 파티션)에서 압축 실행	• Amazon EMR 또는를 사용합니다 AWS Glue. `rewrite_data_files` 절차에서 적극적으로 작성된 파티션을 제외하는 `where` 조건자를 지정합니다. 이 전략은 작성자와 압축 작업 간의 데이터 충돌을 방지하고 Iceberg가 자동으로 해결할 수 있는 메타데이터 충돌만 남깁니다.

Amazon S3에서 Iceberg 워크로드 사용

이 섹션에서는 Iceberg와 Amazon S3의 상호 작용을 최적화하는 데 사용할 수 있는 Iceberg 속성에 대해 설명합니다.

핫 파티셔닝 방지(HTTP 503 오류)

Amazon S3에서 실행되는 일부 데이터 레이크 애플리케이션은 수백만 또는 수십억 개의 객체를 처리하고 페타바이트의 데이터를 처리합니다. 이로 인해 일반적으로 HTTP 503(서비스 사용 불가) 오류를 통해 감지되는 대량의 트래픽을 수신하는 접두사가 발생할 수 있습니다. 이 문제를 방지하려면 다음 Iceberg 속성을 사용합니다.

- Iceberg가 대용량 파일을 쓰`range`도록 `hash` 또는 `write.distribution-mode`로 설정하면 Amazon S3 요청이 줄어듭니다. 이는 기본 구성이며 대부분의 사례를 해결해야 합니다.
- 워크로드에 방대한 양의 데이터로 인해 503 오류가 계속 발생하는 경우 Icebergtrue에서를 `write.object-storage.enabled`로 설정할 수 있습니다. 이렇게 하면 Iceberg가 객체 이름을 해시하고 여러 개의 무작위 Amazon S3 접두사에 로드를 분산하도록 지시합니다.

이러한 속성에 대한 자세한 내용은 Iceberg 설명서의 쓰기 속성을 참조하세요.

Iceberg 유지 관리 작업을 사용하여 미사용 데이터 릴리스

Iceberg 테이블을 관리하기 위해 Iceberg 코어 API, Iceberg 클라이언트(예: Spark) 또는 Amazon Athena와 같은 관리형 서비스를 사용할 수 있습니다. Amazon S3에서 이전 파일 또는 미사용 파일을 삭제하려면 Iceberg 기본 APIs만 사용하여 스냅샷을 제거하고 이전 메타데이터 파일을 제거하며 분리된 파일을 삭제하는 것이 좋습니다.

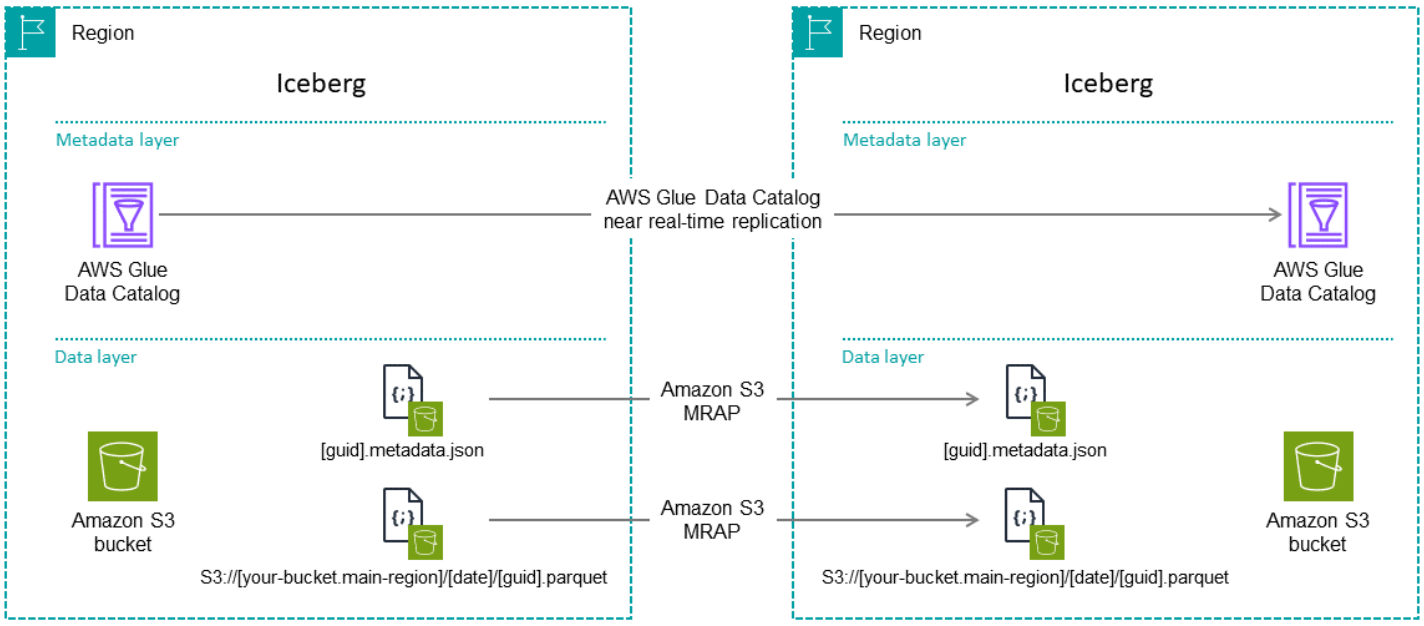
Boto3, Amazon S3 SDK 또는 AWS Command Line Interface (AWS CLI)를 통해 Amazon S3 APIs를 사용하거나 다른 비 Iceberg 메서드를 사용하여 Iceberg 테이블에 대한 Amazon S3 파일을 덮어쓰거나 제거하면 테이블 손상 및 쿼리 실패가 발생합니다.

에서 데이터 복제 AWS 리전

Amazon S3에 Iceberg 테이블을 저장할 때 교차 리전 복제(CRR) 및 다중 리전 액세스 포인트(MRAP)와 같은 Amazon S3의 기본 제공 기능을 사용하여 여러에 데이터를 복제할 수 있습니다 AWS 리전. MRAP는 애플리케이션이 여러에 있는 S3 버킷에 액세스할 수 있는 글로벌 엔드포인트를 제공합니다 AWS 리전. Iceberg는 상대 경로를 지원하지 않지만, MRAP를 사용하여 버킷을 액세스 포인트에 매핑하여 Amazon S3 작업을 수행할 수 있습니다. 또한 MRAP는 Amazon S3 교차 리전 복제 프로세스와 원활하게 통합되어 최대 15분의 지연 시간을 제공합니다. 데이터와 메타데이터 파일을 모두 복제해야 합니다.

> **Important**
>
> 현재 MRAP와의 Iceberg 통합은 Apache Spark에서만 작동합니다. 보조 로 장애 조치해야 하는 경우 사용자 쿼리 AWS 리전를 장애 조치 리전의 Spark SQL 환경(예: Amazon EMR)으로 리디렉션해야 합니다.

CRR 및 MRAP 기능은 다음 다이어그램과 같이 Iceberg 테이블에 대한 리전 간 복제 솔루션을 구축하는 데 도움이 됩니다.



이 교차 리전 복제 아키텍처를 설정하려면:

1. MRAP 위치를 사용하여 테이블을 생성합니다. 이렇게 하면 Iceberg 메타데이터 파일이 물리적 버킷 위치 대신 MRAP 위치를 가리킬 수 있습니다.
2. Amazon S3 MRAP를 사용하여 Iceberg 파일을 복제합니다. MRAP는 15분의 서비스 수준 계약(SLA)으로 데이터 복제를 지원합니다. Iceberg는 읽기 작업에서 복제 중에 불일치가 발생하는 것을 방지합니다.
3. 보조 리전의 AWS Glue Data Catalog 에서 테이블을 사용할 수 있도록 설정합니다. 다음 두 가지 옵션 중에서 선택할 수 있습니다.
   - AWS Glue Data Catalog 복제를 사용하여 Iceberg 테이블 메타데이터를 복제하기 위한 파이프라인을 설정합니다. 이 유틸리티는 GitHub Glue 카탈로그 및 Lake Formation Permissions 복제 리포지토리에서 사용할 수 있습니다. 이 이벤트 기반 메커니즘은 이벤트 로그를 기반으로 대상 리전의 테이블을 복제합니다.
   - 장애 조치가 필요한 경우 보조 리전에 테이블을 등록합니다. 이 옵션의 경우 이전 유틸리티 또는 Iceberg register_table 프로시저를 사용하여 최신 `metadata.json` 파일을 가리킬 수 있습니다.

Apache Iceberg 워크로드 모니터링

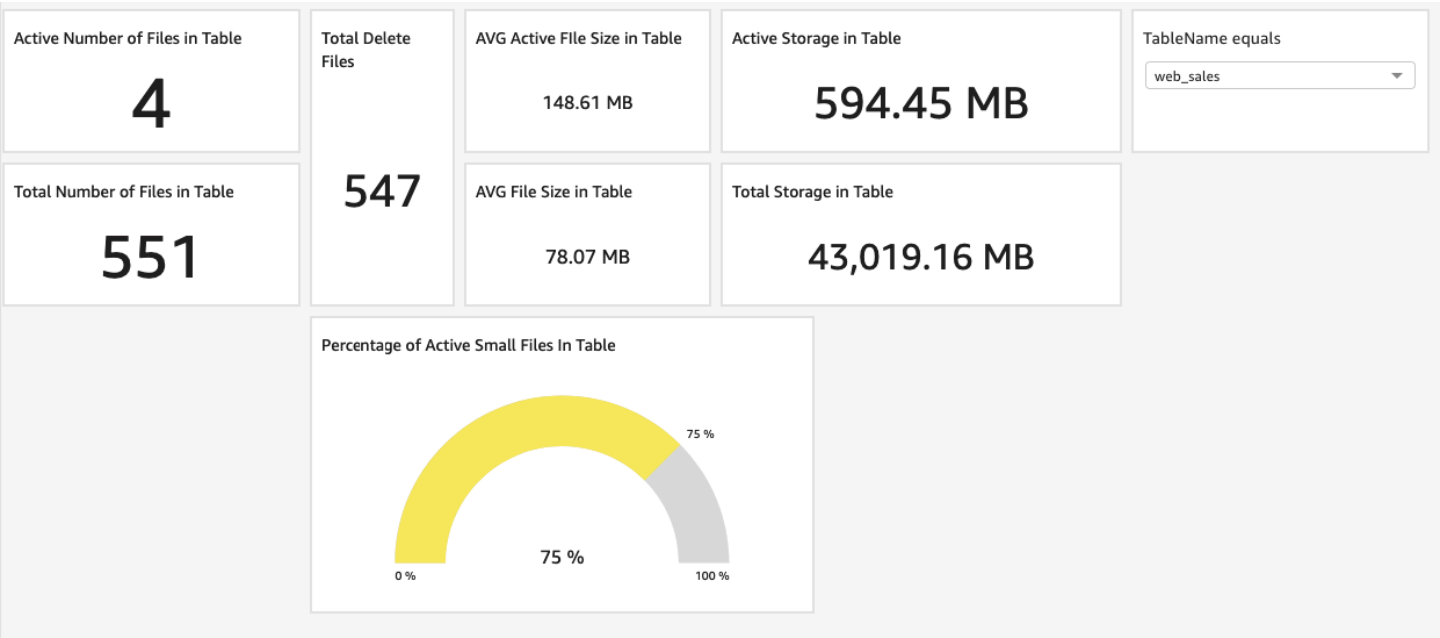
Iceberg 워크로드를 모니터링하려면 메타데이터 테이블 분석 또는 지표 리포터 사용이라는 두 가지 옵션이 있습니다. 지표 리포터는 Iceberg 버전 1.2에 도입되었으며 REST 및 JDBC 카탈로그에만 사용할 수 있습니다.

를 사용하는 경우 Iceberg가 노출하는 메타데이터 테이블 위에 모니터링을 설정하여 Iceberg 테이블의 상태에 대한 인사이트를 얻을 AWS Glue Data Catalog수 있습니다.

모니터링은 성능 관리 및 문제 해결에 매우 중요합니다. 예를 들어 Iceberg 테이블의 파티션이 특정 비율의 작은 파일에 도달하면 워크로드가 압축 작업을 시작하여 파일을 더 큰 파일로 통합할 수 있습니다. 이렇게 하면 쿼리가 허용 수준 이상으로 느려지는 것을 방지할 수 있습니다.

테이블 수준 모니터링

다음 화면은 Amazon Quick Sight에서 생성된 테이블 모니터링 대시보드를 보여줍니다. 이 대시보드는 Spark SQL을 사용하여 Iceberg 메타데이터 테이블을 쿼리하고 활성 파일 수 및 총 스토리지와 같은 세부 지표를 캡처합니다. 그런 다음이 정보는 운영 목적으로 AWS Glue 테이블에 저장됩니다. 마지막으로 다음 그림과 같이 Amazon Athena를 사용하여 Quick Sight 대시보드를 생성합니다. 이 정보는 시스템의 특정 문제를 식별하고 해결하는 데 도움이 됩니다.



Quick Sight 대시보드 예제는 Iceberg 테이블에 대해 다음과 같은 핵심 성과 지표(KPIs)를 수집합니다.

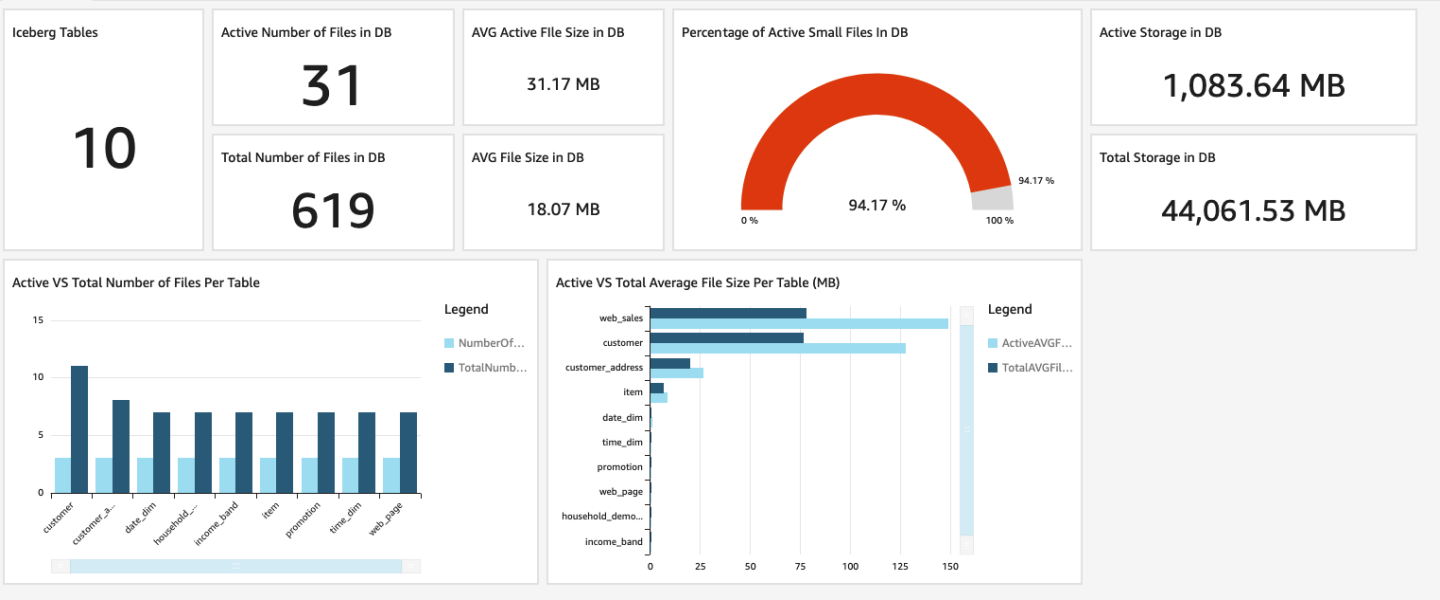
KPI	설명	Query
파일 수	Iceberg 테이블의 파일 수(모든 스냅샷용)	`select count(*) from <catalog.database.table_name>.all_files`
활성 파일 수	Iceberg 테이블의 마지막 스냅샷에 있는 활성 파일 수	`select count(*) from <catalog.database.table_name>.files`
평균 파일 크기	Iceberg 테이블의 모든 파일에 대한 메가바이트 단위의 평균 파일 크기	`select avg(file_size_in_bytes)/1000000 from <catalog.database.table_name>.all_files`
평균 활성 파일 크기	Iceberg 테이블의 활성 파일에 대한 메가바이트 단위의 평균 파일 크기	`select avg(file_size_in_bytes)/1000000 from <catalog.database.table_name>.files`
작은 파일의 백분율	100MB보다 작은 활성 파일의 백분율	`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`
총 스토리지 크기	분리된 파일 및 Amazon S3 객체 버전을 제외한 테이블에 있는 모든 파일의 총 크기(활성화된 경우)	`select sum(file_size_in_bytes)/1000000 from <catalog.database.table_name>.all_files`

KPI	설명	Query
총 활성 스토리지 크기	지정된 테이블의 현재 스냅샷에 있는 모든 파일의 총 크기	`select sum(file_size_in_bytes)/1000000 from <catalog.database.table_name>.files`

대시보드 생성에 대한 자세한 내용은 Quick Sight 설명서를 참조하세요.

데이터베이스 수준 모니터링

다음 예제는 Iceberg 테이블 컬렉션의 데이터베이스 수준 KPIs에 대한 개요를 제공하기 위해 Quick Sight에서 생성된 모니터링 대시보드를 보여줍니다.



이 대시보드는 다음 KPIs를 수집합니다.

KPI	설명	Query
파일 수	Iceberg 데이터베이스의 파일 수(모든 스냅샷용)	이 대시보드는 이전 섹션에 제공된 테이블 수준 쿼리를 사용하고 결과를 통합합니다.

KPI	설명	Query
활성 파일 수	Iceberg 데이터베이스의 활성 파일 수(Iceberg 테이블의 마지막 스냅샷 기준)	
평균 파일 크기	Iceberg 데이터베이스의 모든 파일에 대한 메가바이트 단위의 평균 파일 크기	
평균 활성 파일 크기	Iceberg 데이터베이스의 모든 활성 파일에 대한 메가바이트 단위의 평균 파일 크기	
작은 파일의 백분율	Iceberg 데이터베이스에서 100MB보다 작은 활성 파일의 백분율	
총 스토리지 크기	분리된 파일 및 Amazon S3 객체 버전을 제외한 데이터베이스에 있는 모든 파일의 총 크기(활성화된 경우)	
총 활성 스토리지 크기	데이터베이스에 있는 모든 테이블의 현재 스냅샷에 있는 모든 파일의 총 크기	

예방적 유지 관리

이전 섹션에서 설명한 모니터링 기능을 설정하면 사후 대응 각도 대신 예방 각도에서 테이블 유지 관리에 접근할 수 있습니다. 예를 들어 테이블 수준 및 데이터베이스 수준 지표를 사용하여 다음과 같은 작업을 예약할 수 있습니다.

- 테이블이 N개의 작은 파일에 도달하면 빈 패킹 압축을 사용하여 작은 파일을 그룹화합니다.
- 테이블이 지정된 파티션의 N 삭제 파일에 도달하면 빈 패킹 압축을 사용하여 삭제 파일을 병합합니다.

- 총 스토리지가 활성 스토리지보다 X배 높을 때 스냅샷을 제거하여 이미 압축된 작은 파일을 제거합니다.

아파치 아이스버그의 거버넌스 및 액세스 제어 켜기 AWS

Apache Iceberg는 와 AWS Lake Formation 통합되어 데이터 거버넌스를 단순화합니다. 이 통합을 통해 데이터 레이크 관리자는 Iceberg 테이블에 셀 수준의 액세스 권한을 할당할 수 있습니다. Amazon Athena를 사용하여 Iceberg 테이블을 쿼리하는 예제는 Amazon Athena를 사용하여 Apache Iceberg 테이블과 AWS Lake Formation상호 작용하고 사용하는 교차 계정 세분화된 권한을 사용하는 AWS 블로그 게시물을 참조하십시오. AWS Lake Formation

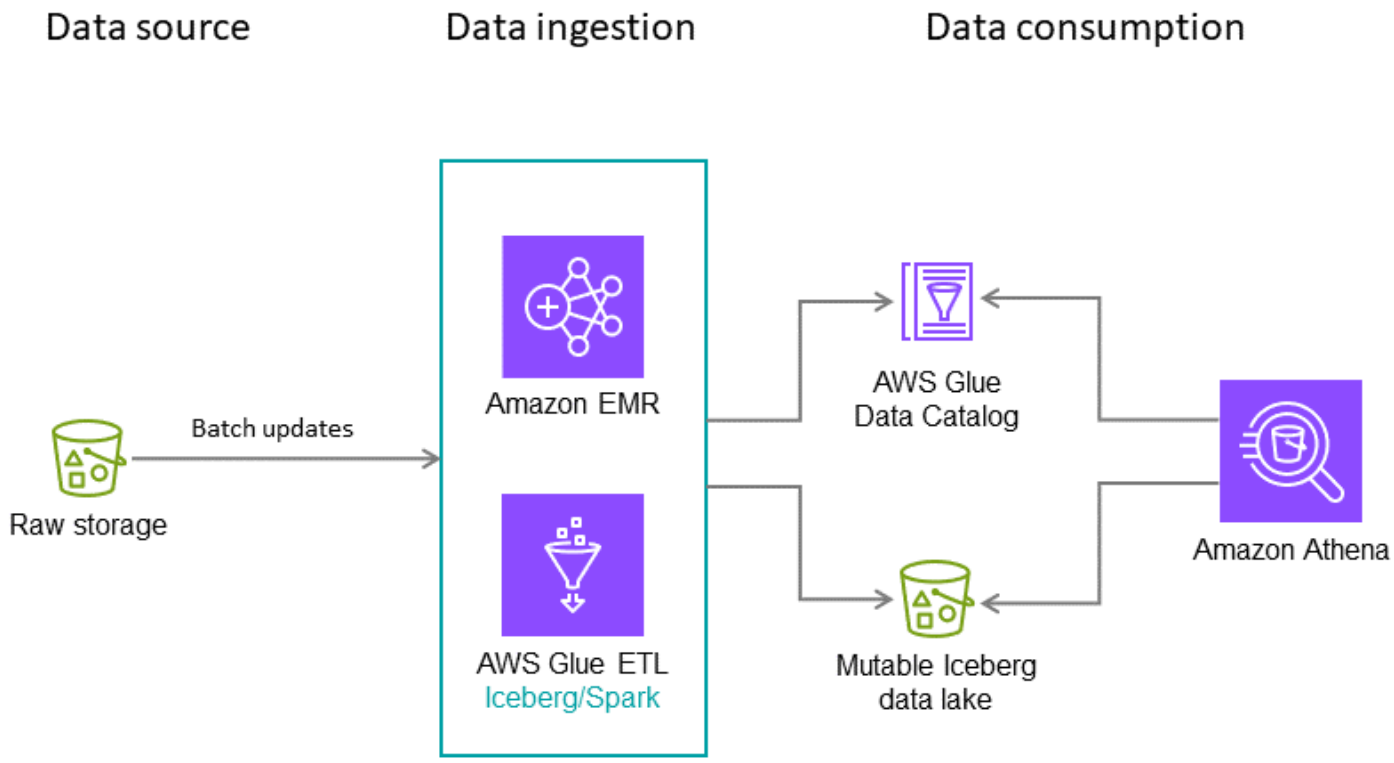
의 Apache Iceberg용 참조 아키텍처 AWS

이 섹션에서는 배치 수집 및 배치와 스트리밍 데이터 수집을 결합하는 데이터 레이크와 같은 다양한 사용 사례에 모범 사례를 적용하는 방법의 예를 제공합니다.

야간 배치 수집

이 가상 사용 사례의 경우 Iceberg 테이블이 매일 밤 신용 카드 거래를 수집한다고 가정해 보겠습니다. 각 배치에는 증분 업데이트만 포함되며, 대상 테이블에 병합해야 합니다. 매년 여러 번 전체 기록 데이터가 수신됩니다. 이 시나리오에서는 다음 아키텍처 및 구성을 사용하는 것이 좋습니다.

참고: 이것은 예시일 뿐입니다. 최적의 구성은 데이터 및 요구 사항에 따라 달라집니다.



권장 사항:

- 파일 크기: 128MB, Apache Spark 태스크는 128MB 청크로 데이터를 처리하기 때문입니다.
- 쓰기 유형: copy-on-write. 이 가이드의 앞부분에서 설명한 대로이 접근 방식은 데이터가 읽기 최적화 방식으로 작성되도록 하는 데 도움이 됩니다.

- 파티션 변수: year/month/day. 가상 사용 사례에서는 최근 데이터를 가장 자주 쿼리하지만, 지난 2년 간의 데이터에 대해 전체 테이블 스캔을 실행하는 경우가 있습니다. 파티셔닝의 목표는 사용 사례의 요구 사항에 따라 빠른 읽기 작업을 구동하는 것입니다.
- 정렬 순서: 타임스탬프
- 데이터 카탈로그: AWS Glue Data Catalog

배치 수집과 실시간에 가까운 수집을 결합한 데이터 레이크

계정 및 리전 간에 배치 및 스트리밍 데이터를 공유하는 Amazon S3의 데이터 레이크를 프로비저닝할 수 있습니다. 아키텍처 다이어그램 및 세부 정보는 AWS 블로그 게시물 Build a transactional data lake using Apache Iceberg AWS Glue, and AWS Lake Formation Amazon Athena를 사용한 교차 계정 데이터 공유를 참조하세요.

리소스

- 에서 Iceberg 프레임워크 사용 AWS Glue(AWS Glue 문서)
- Iceberg(Amazon EMR 설명서)
- Apache Iceberg 테이블 사용(Amazon Athena 설명서)
- Amazon S3 설명서
- 빠른 설명서
- Glue 카탈로그 및 Lake Formation Permissions 복제(GitHub 리포지토리)
- Apache Iceberg 설명서
- Apache Spark 설명서

기여자

의 다음 사람들이이 가이드를 AWS 작성, 공동 작성 및 검토했습니다.

기여자

- Stefano Sandona, 빅 데이터 솔루션 아키텍트
- Imtiaz(Taz) Sayed, 분석 솔루션 아키텍트 기술 리더
- Shana Schipers, 빅 데이터 솔루션 아키텍트
- Prashant Singh, Amazon EMR의 소프트웨어 개발 엔지니어
- Arun A K, 솔루션 아키텍트, 빅 데이터 및 ETL
- Francisco Morillo, 스트리밍 솔루션 아키텍트
- Suthan Phillips, Amazon EMR의 분석 아키텍트
- Sercan Karaoglu, 솔루션 아키텍트
- Yonatan Dolan, 분석 전문가
- Guy Bachar, 솔루션 아키텍트
- Sofia Zilberman, 스트리밍 솔루션 아키텍트
- Dan Stair, 전문가 솔루션 아키텍트
- Sakti Mishra, 솔루션 아키텍트
- Ron Ortloff, Amazon S3의 보안 주체 제품 관리자
- David Zhang, 솔루션 아키텍트, 분석

검토자

- Rick Sears, Amazon EMR 총지배인
- Linda OConnor, Amazon EMR 전문가
- Ian Meyers, Amazon EMR 디렉터
- Vinita Ananth, Amazon EMR 제품 관리 책임자
- Jason Berkowitz, 제품 관리자, AWS Lake Formation
- Mahesh Mishra, Amazon Redshift의 제품 관리자
- Vladimir Zlatkin, 빅 데이터 솔루션 아키텍처 관리자
- Karthik Prabhakar, Amazon EMR의 분석 아키텍트

- Vijay Jain, 제품 관리자
- Anupriti Warade, 제품 관리자, Amazon S3
- Ajit Tandale, 데이터 솔루션 아키텍트
- Gwen Chen, 제품 마케팅 관리자
- Noritaka Sekiyama, 빅 데이터 수석 아키텍트

문서 기록

아래 표에 이 가이드의 주요 변경 사항이 설명되어 있습니다. 향후 업데이트에 대한 알림을 받으려면 RSS 피드를 구독하십시오.

변경 사항	설명	날짜
새로운 섹션	Iceberg 테이블 형식 사양 버전 3 작업에 대한 정보가 추가되었습니다.	2025년 11월 26일
수정	스냅샷 절차 섹션의 `gc.enabled` 설정에 대한 정보가 수정되었습니다.	2025년 10월 24일
추가	Trino 및 PyIceberg를 사용하여 Iceberg 테이블 작업에 대한 새 단원을 추가하고 테이블을 Iceberg로 마이그레이션하는 단원을 확장했습니다.	2025년 8월 12일
업데이트	최신 버전의 AWS Glue, Amazon EMR 및 Apache Iceberg를 반영하도록 가이드 전체에서 정보를 개선하고 명확히 했습니다.	2025년 7월 14일
추가	Amazon Data Firehose를 사용하여 Iceberg 테이블 작업에 대한 새 섹션을 추가했습니다.	2025년 2월 20일
최초 게시	—	2024년 4월 30일

AWS 권장 가이드 용어집

다음은 AWS 권장 가이드에서 제공하는 전략, 가이드 및 패턴에서 일반적으로 사용되는 용어입니다. 용어집 항목을 제안하려면 용어집 끝에 있는 피드백 제공 링크를 사용하십시오.

숫자

7가지 전략

애플리케이션을 클라우드로 이전하기 위한 7가지 일반적인 마이그레이션 전략 이러한 전략은 Gartner가 2011년에 파악한 5가지 전략을 기반으로 하며 다음으로 구성됩니다.

- 리팩터링/리아키텍트 - 클라우드 네이티브 기능을 최대한 활용하여 애플리케이션을 이동하고 해당 아키텍처를 수정함으로써 민첩성, 성능 및 확장성을 개선합니다. 여기에는 일반적으로 운영 체제와 데이터베이스 이식이 포함됩니다. 예: 온프레미스 Oracle 데이터베이스를 Amazon Aurora PostgreSQL 호환 에디션으로 마이그레이션합니다.
- 리플랫포밍(리프트 앤드 리셰이프) - 애플리케이션을 클라우드로 이동하고 일정 수준의 최적화를 도입하여 클라우드 기능을 활용합니다. 예: 온프레미스 Oracle 데이터베이스를 AWS 클라우드의 Amazon Relational Database Service(Amazon RDS) for Oracle로 마이그레이션합니다.
- 재구매(드롭 앤드 숍) - 일반적으로 기존 라이선스에서 SaaS 모델로 전환하여 다른 제품으로 전환합니다. 예: 고객 관계 관리(CRM) 시스템을 Salesforce.com으로 마이그레이션합니다.
- 리호스팅(리프트 앤드 시프트) - 애플리케이션을 변경하지 않고 클라우드로 이동하여 클라우드 기능을 활용합니다. 예: 온프레미스 Oracle 데이터베이스를 AWS 클라우드클라우드의 EC2 인스턴스에 있는 Oracle로 마이그레이션합니다.
- 재배치(하이퍼바이저 수준의 리프트 앤 시프트) - 새 하드웨어를 구매하거나, 애플리케이션을 다시 작성하거나, 기존 운영을 수정하지 않고도 인프라를 클라우드로 이동합니다. 온프레미스 플랫폼에서 동일한 플랫폼의 클라우드 서비스로 서버를 마이그레이션합니다. 예: Microsoft Hyper-V 애플리케이션을 로 마이그레이션합니다 AWS.
- 유지(보관) - 소스 환경에 애플리케이션을 유지합니다. 대규모 리팩터링이 필요하고 해당 작업을 나중으로 연기하려는 애플리케이션과 비즈니스 차원에서 마이그레이션할 이유가 없어 유지하려는 레거시 애플리케이션이 여기에 포함될 수 있습니다.
- 사용 중지 - 소스 환경에서 더 이상 필요하지 않은 애플리케이션을 폐기하거나 제거합니다.

A

ABAC

속성 기반 액세스 제어를 참조하세요.

추상화된 서비스

관리형 서비스를 참조하세요.

ACID

원자성, 일관성, 격리성, 내구성을 참조하세요.

능동-능동 마이그레이션

양방향 복제 도구 또는 이중 쓰기 작업을 사용하여 소스 데이터베이스와 대상 데이터베이스가 동기화된 상태로 유지되고, 두 데이터베이스 모두 마이그레이션 중 연결 애플리케이션의 트랜잭션을 처리하는 데이터베이스 마이그레이션 방법입니다. 이 방법은 일회성 전환이 필요한 대신 소규모의 제어된 배치로 마이그레이션을 지원합니다. 더 유연하지만 액티브 패시브 마이그레이션보다 더 많은 작업이 필요합니다.

능동-수동 마이그레이션

소스 데이터베이스와 대상 데이터베이스가 동기화된 상태로 유지되지만 소스 데이터베이스만 연결 애플리케이션의 트랜잭션을 처리하고 데이터는 대상 데이터베이스로 복제되는 데이터베이스 마이그레이션 방법입니다. 대상 데이터베이스는 마이그레이션 중 어떤 트랜잭션도 허용하지 않습니다.

집계 함수

행 그룹에서 작동하고 그룹에 대한 단일 반환 값을 계산하는 SQL 함수입니다. 집계 함수의 예로 `SUM` 및 `MAX`가 있습니다.

AI

인공 지능을 참조하세요.

AIOps

인공 지능 운영을 참조하세요.

익명화

데이터세트에서 개인 정보를 영구적으로 삭제하는 프로세스입니다. 익명화는 개인 정보 보호에 도움이 될 수 있습니다. 익명화된 데이터는 더 이상 개인 데이터로 간주되지 않습니다.

안티 패턴

솔루션이 다른 솔루션보다 비생산적이거나 비효율적이거나 덜 효과적이어서 반복되는 문제에 자주 사용되는 솔루션입니다.

애플리케이션 제어

맬웨어로부터 시스템을 보호하기 위해 승인된 애플리케이션만 사용하도록 허용하는 보안 접근 방식입니다.

애플리케이션 포트폴리오

애플리케이션 구축 및 유지 관리 비용과 애플리케이션의 비즈니스 가치를 비롯하여 조직에서 사용하는 각 애플리케이션에 대한 세부 정보 모음입니다. 이 정보는 포트폴리오 탐색 및 분석 프로세스의 핵심이며 마이그레이션, 현대화 및 최적화할 애플리케이션을 식별하고 우선순위를 정하는 데 도움이 됩니다.

인공 지능

컴퓨터 기술을 사용하여 학습, 문제 해결, 패턴 인식 등 일반적으로 인간과 관련된 인지 기능을 수행하는 것을 전문으로 하는 컴퓨터 과학 분야입니다. 자세한 내용은 What is Artificial Intelligence?를 참조하십시오.

인공 지능 운영(AIOps)

기계 학습 기법을 사용하여 운영 문제를 해결하고, 운영 인시던트 및 사용자 개입을 줄이고, 서비스 품질을 높이는 프로세스입니다. AWS 마이그레이션 전략에서 AIOps가 사용되는 방법에 대한 자세한 내용은 운영 통합 가이드를 참조하십시오.

비대칭 암호화

한 쌍의 키, 즉 암호화를 위한 퍼블릭 키와 복호화를 위한 프라이빗 키를 사용하는 암호화 알고리즘입니다. 퍼블릭 키는 복호화에 사용되지 않으므로 공유할 수 있지만 프라이빗 키에 대한 액세스는 엄격히 제한되어야 합니다.

원자성, 일관성, 격리성, 내구성(ACID)

오류, 정전 또는 기타 문제가 발생한 경우에도 데이터베이스의 데이터 유효성과 운영 신뢰성을 보장하는 소프트웨어 속성 세트입니다.

ABAC(속성 기반 액세스 제어)

부서, 직무, 팀 이름 등의 사용자 속성을 기반으로 세분화된 권한을 생성하는 방식입니다. 자세한 내용은 AWS Identity and Access Management (IAM) 설명서의 용 ABAC AWS를 참조하세요.

신뢰할 수 있는 데이터 소스

가장 신뢰할 수 있는 정보 소스로 간주되는 기본 버전의 데이터를 저장하는 위치입니다. 익명화, 편집 또는 가명화와 같은 데이터 처리 또는 수정의 목적으로 신뢰할 수 있는 데이터 소스의 데이터를 다른 위치로 복사할 수 있습니다.

가용 영역

다른 가용 영역의 장애로부터 격리 AWS 리전 되고 동일한 리전의 다른 가용 영역에 저렴하고 지연 시간이 짧은 네트워크 연결을 제공하는 내의 고유한 위치입니다.

AWS 클라우드 채택 프레임워크(AWS CAF)

조직이 클라우드로 성공적으로 전환 AWS 하기 위한 효율적이고 효과적인 계획을 개발하는 데 도움이 되는의 지침 및 모범 사례 프레임워크입니다. AWS CAF는 지침을 비즈니스, 사람, 거버넌스, 플랫폼, 보안 및 운영이라는 6가지 중점 영역으로 구성합니다. 비즈니스, 사람 및 거버넌스 관점은 비즈니스 기술과 프로세스에 초점을 맞추고, 플랫폼, 보안 및 운영 관점은 전문 기술과 프로세스에 중점을 둡니다. 예를 들어, 사람 관점은 인사(HR), 직원 배치 기능 및 인력 관리를 담당하는 이해관계자를 대상으로 합니다. 이러한 관점에서 AWS CAF는 성공적인 클라우드 채택을 위해 조직을 준비하는 데 도움이 되는 인력 개발, 교육 및 커뮤니케이션에 대한 지침을 제공합니다. 자세한 내용은 AWS CAF 웹사이트와 AWS CAF 백서를 참조하세요.

AWS 워크로드 검증 프레임워크(AWS WQF)

데이터베이스 마이그레이션 워크로드를 평가하고, 마이그레이션 전략을 권장하고, 작업 견적을 제공하는 도구입니다. AWS WQF는 AWS Schema Conversion Tool (AWS SCT)에 포함되어 있습니다. 데이터베이스 스키마 및 코드 객체, 애플리케이션 코드, 종속성 및 성능 특성을 분석하고 평가 보고서를 제공합니다.

B

악성 봇

개인 또는 조직을 방해하거나 해를 입히기 위한 봇입니다.

BCP

비즈니스 연속성 계획을 참조하세요.

동작 그래프

리소스 동작과 시간 경과에 따른 상호 작용에 대한 통합된 대화형 뷰입니다. Amazon Detective에서 동작 그래프를 사용하여 실패한 로그온 시도, 의심스러운 API 직접 호출 및 유사한 작업을 검사할 수 있습니다. 자세한 내용은 Detective 설명서의 Data in a behavior graph를 참조하십시오.

빅 엔디안 시스템

가장 중요한 바이트를 먼저 저장하는 시스템입니다. 엔디안도 참조하세요.

바이너리 분류

바이너리 결과(가능한 두 클래스 중 하나)를 예측하는 프로세스입니다. 예를 들어, ML 모델이 "이 이메일이 스팸인가요, 스팸이 아닌가요?", '이 제품은 책인가요, 자동차인가요?' 등의 문제를 예측해야 할 수 있습니다.

블룸 필터

요소가 세트의 멤버인지 여부를 테스트하는 데 사용되는 메모리 효율성이 높은 확률론적 데이터 구조입니다.

블루/그린(Blue/Green) 배포

동일하지만 별개의 두 환경을 생성하는 배포 전략입니다. 하나의 환경(파란색)에서 현재 애플리케이션 버전을 실행하고 새 애플리케이션 버전은 다른 환경(녹색)에서 실행합니다. 이 전략을 사용하면 영향을 최소화하면서 신속하게 롤백할 수 있습니다.

bot

인터넷을 통해 자동화된 태스크를 실행하고 인적 활동이나 상호 작용을 시뮬레이션하는 소프트웨어 애플리케이션입니다. 인터넷에서 정보를 인덱싱하는 웹 크롤러와 같이 유용하거나 이로운 봇도 있습니다. 악성 봇이라고 하는 다른 일부 봇은 개인 또는 조직을 방해하거나 해를 입히기 위한 봇입니다.

봇넷

맬웨어에 감염되고 봇 허더 또는 봇 운영자와 같은 단일 당사자가 제어하는 봇 네트워크입니다. 봇넷은 봇의 규모와 봇의 영향 범위를 확대하는 가장 잘 알려진 메커니즘입니다.

브랜치

코드 리포지토리의 포함된 영역입니다. 리포지토리에 생성되는 첫 번째 브랜치가 기본 브랜치입니다. 기존 브랜치에서 새 브랜치를 생성한 다음 새 브랜치에서 기능을 개발하거나 버그를 수정할 수 있습니다. 기능을 구축하기 위해 생성하는 브랜치를 일반적으로 기능 브랜치라고 합니다. 기능을 출시할 준비가 되면 기능 브랜치를 기본 브랜치에 다시 병합합니다. 자세한 내용은 About branches(GitHub 설명서)를 참조하십시오.

긴급 액세스 권한

예외적인 상황에서 승인된 프로세스를 통해 사용자가 일반적으로 액세스할 권한이 없는에 액세스할 수 AWS 계정 있는 빠른 방법입니다. 자세한 내용은 AWS Well-Architected 지침의 Implement break-glass procedures 지표를 참조하세요.

브라운필드 전략

사용자 환경의 기존 인프라 시스템 아키텍처에 브라운필드 전략을 채택할 때는 현재 시스템 및 인프라의 제약 조건을 중심으로 아키텍처를 설계합니다. 기존 인프라를 확장하는 경우 브라운필드 전략과 그린필드 전략을 혼합할 수 있습니다.

버퍼 캐시

가장 자주 액세스하는 데이터가 저장되는 메모리 영역입니다.

사업 역량

기업이 가치를 창출하기 위해 하는 일(예: 영업, 고객 서비스 또는 마케팅)입니다. 마이크로서비스 아키텍처 및 개발 결정은 비즈니스 역량에 따라 이루어질 수 있습니다. 자세한 내용은 백서의 AWS에서 컨테이너화된 마이크로서비스 실행의 비즈니스 역량 중심의 구성화 섹션을 참조하십시오.

비즈니스 연속성 계획(BCP)

대규모 마이그레이션과 같은 중단 이벤트가 운영에 미치는 잠재적 영향을 해결하고 비즈니스가 신속하게 운영을 재개할 수 있도록 지원하는 계획입니다.

C

CAF

AWS Cloud Adoption Framework를 참조하세요.

카나리 배포

최종 사용자에게 제공하는 느린 증분 릴리스 버전입니다. 확신이 들면 새 버전을 배포하고 현재 버전을 완전히 교체합니다.

CCoE

클라우드 혁신 센터를 참조하세요.

CDC

데이터 캡처 변경을 참조하세요.

변경 데이터 캡처(CDC)

데이터베이스 테이블과 같은 데이터 소스의 변경 내용을 추적하고 변경 사항에 대한 메타데이터를 기록하는 프로세스입니다. 대상 시스템의 변경 내용을 감사하거나 복제하여 동기화를 유지하는 등의 다양한 용도로 CDC를 사용할 수 있습니다.

카오스 엔지니어링

시스템의 복원력을 테스트하기 위해 의도적으로 장애나 중단 이벤트를 도입합니다. AWS Fault Injection Service (AWS FIS)를 사용하여 AWS 워크로드에 스트레스를 주고 응답을 평가하는 실험을 수행할 수 있습니다.

CI/CD

지속적 통합 및 지속적 전송을 참조하세요.

분류

예측을 생성하는 데 도움이 되는 분류 프로세스입니다. 분류 문제에 대한 ML 모델은 이산 값을 예측합니다. 이산 값은 항상 서로 다릅니다. 예를 들어, 모델이 이미지에 자동차가 있는지 여부를 평가해야 할 수 있습니다.

클라이언트측 암호화

대상이 데이터를 AWS 서비스 수신하기 전에 로컬에서 데이터를 암호화합니다.

클라우드 혁신 센터(CCoE)

클라우드 모범 사례 개발, 리소스 동원, 마이그레이션 타임라인 설정, 대규모 혁신을 통한 조직 선도 등 조직 전체에서 클라우드 채택 노력을 추진하는 다분야 팀입니다. 자세한 내용은 AWS 클라우드 엔터프라이즈 전략 블로그의 CCoE 게시물을 참조하세요.

클라우드 컴퓨팅

원격 데이터 스토리지와 IoT 디바이스 관리에 일반적으로 사용되는 클라우드 기술 클라우드 컴퓨팅은 일반적으로 엣지 컴퓨팅 기술에 연결되어 있습니다.

클라우드 운영 모델

IT 조직에서 하나 이상의 클라우드 환경을 구축, 성숙화 및 최적화하는 데 사용되는 운영 모델입니다. 자세한 내용은 클라우드 운영 모델 구축을 참조하십시오.

클라우드 채택 단계

조직이 AWS 클라우드로 마이그레이션할 때 일반적으로 거치는 4단계는 다음과 같습니다.

- 프로젝트 - 개념 증명 및 학습 목적으로 몇 가지 클라우드 관련 프로젝트 실행
- 기반 – 클라우드 채택 확장을 위한 기초 투자(예: 랜딩 존 생성, CCoE 정의, 운영 모델 구축)
- 마이그레이션 - 개별 애플리케이션 마이그레이션
- Re-invention - 제품 및 서비스 최적화와 클라우드 혁신

이러한 단계는 Stephen Orban이 블로그 게시물 The Journey Toward Cloud-First and the Stages of Adoption on the AWS 클라우드 Enterprise Strategy 블로그에서 정의했습니다. AWS 마이그레이션 전략과 어떤 관련이 있는지에 대한 자세한 내용은 마이그레이션 준비 가이드를 참조하세요.

CMDB

구성 관리 데이터베이스를 참조하세요.

코드 리포지토리

소스 코드와 설명서, 샘플, 스크립트 등의 기타 자산이 버전 관리 프로세스를 통해 저장되고 업데이트되는 위치입니다. 일반적인 클라우드 리포지토리로 GitHub 또는 Bitbucket Cloud가 포함됩니다. 코드의 각 버전을 브랜치라고 합니다. 마이크로서비스 구조에서 각 리포지토리는 단일 기능 전용입니다. 단일 CI/CD 파이프라인은 여러 리포지토리를 사용할 수 있습니다.

콜드 캐시

비어 있거나, 제대로 채워지지 않았거나, 오래되었거나 관련 없는 데이터를 포함하는 버퍼 캐시입니다. 주 메모리나 디스크에서 데이터베이스 인스턴스를 읽어야 하기 때문에 성능에 영향을 미치며, 이는 버퍼 캐시에서 읽는 것보다 느립니다.

콜드 데이터

거의 액세스되지 않고 일반적으로 과거 데이터인 데이터. 이런 종류의 데이터를 쿼리할 때는 일반적으로 느린 쿼리가 허용됩니다. 이 데이터를 성능이 낮고 비용이 저렴한 스토리지 계층 또는 클래스로 옮기면 비용을 절감할 수 있습니다.

컴퓨터 비전(CV)

기계 학습을 사용하여 디지털 이미지 및 비디오와 같은 시각적 형식에서 정보를 분석하고 추출하는 AI 필드입니다. 예를 들어 Amazon SageMaker AI는 CV에 대한 이미지 처리 알고리즘을 제공합니다.

구성 드리프트

워크로드의 경우 구성이 예상되는 상태에서 변경됩니다. 이로 인해 워크로드가 규정을 준수하지 않을 수 있으며, 이는 일반적으로 점진적이고 의도되지 않은 작업입니다.

구성 관리 데이터베이스(CMDB)

하드웨어 및 소프트웨어 구성 요소와 해당 구성을 포함하여 데이터베이스와 해당 IT 환경에 대한 정보를 저장하고 관리하는 리포지토리입니다. 일반적으로 마이그레이션의 포트폴리오 탐색 및 분석 단계에서 CMDB의 데이터를 사용합니다.

규정 준수 팩

규정 준수 및 보안 검사를 사용자 지정하기 위해 조합할 수 있는 AWS Config 규칙 및 수정 작업 모음입니다. YAML 템플릿을 사용하여 적합성 팩을 AWS 계정 및 리전 또는 조직 전체에 단일 엔터티로 배포할 수 있습니다. 자세한 내용은 AWS Config 설명서의 적합성 팩을 참조하세요.

지속적 통합 및 지속적 전달(CI/CD)

소프트웨어 릴리스 프로세스의 소스, 빌드, 테스트, 스테이징 및 프로덕션 단계를 자동화하는 프로세스입니다. CI/CD는 일반적으로 파이프라인으로 설명됩니다. CI/CD를 통해 프로세스를 자동화하고, 생산성을 높이고, 코드 품질을 개선하고, 더 빠르게 제공할 수 있습니다. 자세한 내용은 지속적 전달의 이점을 참조하십시오. CD는 지속적 배포를 의미하기도 합니다. 자세한 내용은 지속적 전달(Continuous Delivery)과 지속적인 개발을 참조하십시오.

CV

컴퓨터 비전을 참조하세요.

D

저장 데이터

스토리지에 있는 데이터와 같이 네트워크에 고정되어 있는 데이터입니다.

데이터 분류

중요도와 민감도를 기준으로 네트워크의 데이터를 식별하고 분류하는 프로세스입니다. 이 프로세스는 데이터에 대한 적절한 보호 및 보존 제어를 결정하는 데 도움이 되므로 사이버 보안 위험 관리 전략의 중요한 구성 요소입니다. 데이터 분류는 AWS Well-Architected Framework의 보안 원칙 구성 요소입니다. 자세한 내용은 데이터 분류를 참조하십시오.

데이터 드리프트

프로덕션 데이터와 ML 모델 학습에 사용된 데이터 간의 상당한 차이 또는 시간 경과에 따른 입력 데이터의 의미 있는 변화. 데이터 드리프트는 ML 모델 예측의 전반적인 품질, 정확성 및 공정성을 저하시킬 수 있습니다.

전송 중 데이터

네트워크를 통과하고 있는 데이터입니다. 네트워크 리소스 사이를 이동 중인 데이터를 예로 들 수 있습니다.

데이터 메시

중앙 집중식 관리 및 거버넌스를 통해 분산되고 탈중앙화된 데이터 소유권을 제공하는 아키텍처 프레임워크입니다.

데이터 최소화

꼭 필요한 데이터만 수집하고 처리하는 원칙입니다. 에서 데이터를 최소화하면 개인 정보 보호 위험, 비용 및 분석 탄소 발자국을 줄일 AWS 클라우드 수 있습니다.

데이터 경계

신뢰할 수 있는 자격 증명만 예상 네트워크에서 신뢰할 수 있는 리소스에 액세스하도록 하는 데 도움이 되는 AWS 환경의 예방 가드레일 세트입니다. 자세한 내용은 데이터 경계 구축을 참조하세요 AWS.

데이터 사전 처리

원시 데이터를 ML 모델이 쉽게 구문 분석할 수 있는 형식으로 변환하는 것입니다. 데이터를 사전 처리한다는 것은 특정 열이나 행을 제거하고 누락된 값, 일관성이 없는 값 또는 중복 값을 처리함을 의미할 수 있습니다.

데이터 출처

라이프사이클 전반에 걸쳐 데이터의 출처와 기록을 추적하는 프로세스(예: 데이터 생성, 전송, 저장 방법).

데이터 주체

데이터를 수집 및 처리하는 개인입니다.

데이터 웨어하우스

분석과 같은 비즈니스 인텔리전스를 지원하는 데이터 관리 시스템입니다. 데이터 웨어하우스에는 보통 많은 양의 기록 데이터가 포함되며 일반적으로 쿼리 및 분석에 사용됩니다.

데이터 정의 언어(DDL)

데이터베이스에서 테이블 및 객체의 구조를 만들거나 수정하기 위한 명령문 또는 명령입니다.

데이터베이스 조작 언어(DML)

데이터베이스에서 정보를 수정(삽입, 업데이트 및 삭제)하기 위한 명령문 또는 명령입니다.

DDL

데이터 정의 언어를 참조하세요.

딥 앙상블

예측을 위해 여러 딥 러닝 모델을 결합하는 것입니다. 딥 앙상블을 사용하여 더 정확한 예측을 얻거나 예측의 불확실성을 추정할 수 있습니다.

딥 러닝

여러 계층의 인공 신경망을 사용하여 입력 데이터와 관심 대상 변수 간의 매핑을 식별하는 ML 하위 분야입니다.

심층 방어

네트워크와 그 안의 데이터 기밀성, 무결성 및 가용성을 보호하기 위해 컴퓨터 네트워크 전체에 일련의 보안 메커니즘과 제어를 신중하게 계층화하는 정보 보안 접근 방식입니다. 이 전략을 채택하면 AWS Organizations 구조의 여러 계층에 여러 제어를 AWS추가하여 리소스를 보호할 수 있습니다. 예를 들어, 심층 방어 접근 방식은 다단계 인증, 네트워크 세분화 및 암호화를 결합할 수 있습니다.

위임된 관리자

에서 AWS Organizations호환되는 서비스는 AWS 멤버 계정을 등록하여 조직의 계정을 관리하고 해당 서비스에 대한 권한을 관리할 수 있습니다. 이러한 계정을 해당 서비스의 위임된 관리자라고 합니다. 자세한 내용과 호환되는 서비스 목록은 AWS Organizations 설명서의 AWS Organizations와 함께 사용할 수 있는 AWS 서비스를 참조하십시오.

배포

대상 환경에서 애플리케이션, 새 기능 또는 코드 수정 사항을 사용할 수 있도록 하는 프로세스입니다. 배포에는 코드 베이스의 변경 사항을 구현한 다음 애플리케이션 환경에서 해당 코드베이스를 구축하고 실행하는 작업이 포함됩니다.

개발 환경

환경을 참조하세요.

탐지 제어

이벤트 발생 후 탐지, 기록 및 알림을 수행하도록 설계된 보안 제어입니다. 이러한 제어는 기존의 예방적 제어를 우회한 보안 이벤트를 알리는 2차 방어선입니다. 자세한 내용은 AWS에서 보안 제어 구현의 탐지 제어를 참조하세요.

개발 가치 흐름 매핑 (DVSM)

소프트웨어 개발 라이프사이클에서 속도와 품질에 부정적인 영향을 미치는 제약 조건을 식별하고 우선 순위를 지정하는 데 사용되는 프로세스입니다. DVSM은 원래 린 제조 방식을 위해 설계된 가치 흐름 매핑 프로세스를 확장합니다. 소프트웨어 개발 프로세스를 통해 가치를 창출하고 이동하는 데 필요한 단계와 팀에 중점을 둡니다.

디지털 트윈

건물, 공장, 산업 장비 또는 생산 라인과 같은 실제 시스템을 가상으로 표현한 것입니다. 디지털 트윈은 예측 유지 보수, 원격 모니터링, 생산 최적화를 지원합니다.

차원 테이블

스타 스키마에서 팩트 테이블의 정량적 데이터에 대한 데이터 속성을 포함하는 더 작은 테이블을 말합니다. 차원 테이블 속성은 일반적으로 텍스트 필드나 텍스트처럼 동작하는 개별 숫자입니다. 이러한 속성은 보통 쿼리 제약, 필터링 및 결과 세트 레이블 지정에 사용됩니다.

재해

워크로드 또는 시스템이 기본 배포 위치에서 비즈니스 목표를 달성하지 못하게 방해하는 이벤트입니다. 이러한 이벤트는 자연재해, 기술적 오류, 의도하지 않은 구성 오류 또는 멀웨어 공격과 같은 사람의 행동으로 인한 결과일 수 있습니다.

재해 복구(DR)

재해로 인한 가동 중지 시간 및 데이터 손실을 최소화하기 위해 사용하는 전략 및 프로세스입니다. 자세한 내용은 AWS Well-Architected Framework의 Disaster Recovery of Workloads on AWS: Recovery in the Cloud를 참조하세요.

DML

데이터베이스 조작 언어를 참조하세요.

도메인 기반 설계

구성 요소를 각 구성 요소가 제공하는 진화하는 도메인 또는 핵심 비즈니스 목표에 연결하여 복잡한 소프트웨어 시스템을 개발하는 접근 방식입니다. 이 개념은 에릭 에반스에 의해 그의 저서인 도메인 기반 디자인: 소프트웨어 중심의 복잡성 해결(Boston: Addison-Wesley Professional, 2003)에서 소개되었습니다. Strangler Fig 패턴과 함께 도메인 기반 설계를 사용하는 방법에 대한 자세한 내용은 컨테이너 및 Amazon API Gateway를 사용하여 기존의 Microsoft ASP.NET(ASMX) 웹 서비스를 점진적으로 현대화하는 방법을 참조하십시오.

DR

재해 복구를 참조하세요.

드리프트 감지

기준이 되는 구성과의 편차 추적을 말합니다. 예를 들어 AWS CloudFormation 를 사용하여 시스템 리소스의 드리프트를 감지하거나를 사용하여 AWS Control Tower 거버넌스 요구 사항 준수에 영향을 미칠 수 있는 랜딩 존의 변경 사항을 감지할 수 있습니다.

DVSM

개발 가치 흐름 매핑을 참조하세요.

E

EDA

탐색 데이터 분석을 참조하세요.

EDI

전자 데이터 교환을 참조하세요.

엣지 컴퓨팅

IoT 네트워크의 엣지에서 스마트 디바이스의 컴퓨팅 성능을 개선하는 기술 엣지 컴퓨팅은 클라우드 컴퓨팅에 비해 보다 통신 지연 시간을 줄이고 응답 시간을 개선할 수 있습니다.

전자 데이터 교환(EDI)

조직 간 비즈니스 문서의 자동화된 교환을 나타냅니다. 자세한 내용은 전자 데이터 교환(EDI)이란 무엇인가요?를 참조하세요.

암호화

사람이 읽을 수 있는 일반 텍스트 데이터를 사이퍼텍스트로 변환하는 컴퓨팅 프로세스입니다.

암호화 키

암호화 알고리즘에 의해 생성되는 무작위 비트의 암호화 문자열입니다. 키의 길이는 다양할 수 있으며 각 키는 예측할 수 없고 고유하게 설계되었습니다.

엔디안

컴퓨터 메모리에 바이트가 저장되는 순서입니다. 빅 엔디안 시스템은 가장 중요한 바이트를 먼저 저장합니다. 리틀 엔디안 시스템은 가장 덜 중요한 바이트를 먼저 저장합니다.

엔드포인트

서비스 엔드포인트를 참조하세요.

엔드포인트 서비스

Virtual Private Cloud(VPC)에서 호스팅하여 다른 사용자와 공유할 수 있는 서비스입니다. 를 사용하여 엔드포인트 서비스를 생성하고 다른 AWS 계정 또는 AWS Identity and Access Management (IAM) 보안 주체에 권한을 AWS PrivateLink 부여할 수 있습니다. 이러한 계정 또는 보안 주체는 인터페이스 VPC 엔드포인트를 생성하여 엔드포인트 서비스에 비공개로 연결할 수 있습니다. 자세한 내용은 Amazon Virtual Private Cloud(VPC) 설명서의 엔드포인트 서비스 생성을 참조하십시오.

엔터프라이즈 리소스 계획(ERP)

엔터프라이즈의 주요 비즈니스 프로세스(예: 회계, MES, 프로젝트 관리)를 자동화하고 관리하는 시스템입니다.

봉투 암호화

암호화 키를 다른 암호화 키로 암호화하는 프로세스입니다. 자세한 내용은 AWS Key Management Service (AWS KMS) 설명서의 봉투 암호화를 참조하세요.

환경

실행 중인 애플리케이션의 인스턴스입니다. 다음은 클라우드 컴퓨팅의 일반적인 환경 유형입니다.

- 개발 환경 - 애플리케이션 유지 관리를 담당하는 핵심 팀만 사용할 수 있는 실행 중인 애플리케이션의 인스턴스입니다. 개발 환경은 변경 사항을 상위 환경으로 승격하기 전에 테스트하는 데 사용됩니다. 이러한 유형의 환경을 테스트 환경이라고도 합니다.
- 하위 환경 - 초기 빌드 및 테스트에 사용되는 환경을 비롯한 애플리케이션의 모든 개발 환경입니다.
- 프로덕션 환경 - 최종 사용자가 액세스할 수 있는 실행 중인 애플리케이션의 인스턴스입니다. CI/CD 파이프라인에서 프로덕션 환경이 마지막 배포 환경입니다.
- 상위 환경 - 핵심 개발 팀 이외의 사용자가 액세스할 수 있는 모든 환경입니다. 프로덕션 환경, 프로덕션 이전 환경 및 사용자 수용 테스트를 위한 환경이 여기에 포함될 수 있습니다.

에픽

애자일 방법론에서 작업을 구성하고 우선순위를 정하는 데 도움이 되는 기능적 범주입니다. 에픽은 요구 사항 및 구현 작업에 대한 개괄적인 설명을 제공합니다. 예를 들어, AWS CAF 보안 에픽에는 ID 및 액세스 관리, 탐지 제어, 인프라 보안, 데이터 보호 및 인시던트 대응이 포함됩니다. AWS 마이그레이션 전략의 에픽에 대한 자세한 내용은 프로그램 구현 가이드를 참조하십시오.

ERP

엔터프라이즈 리소스 계획을 참조하세요.

탐색 데이터 분석(EDA)

데이터 세트를 분석하여 주요 특성을 파악하는 프로세스입니다. 데이터를 수집 또는 집계한 다음 초기 조사를 수행하여 패턴을 찾고, 이상을 탐지하고, 가정을 확인합니다. EDA는 요약 통계를 계산하고 데이터 시각화를 생성하여 수행됩니다.

F

팩트 테이블

스타 스키마의 중앙 테이블입니다. 비즈니스 운영에 대한 정량적 데이터를 저장합니다. 일반적으로 팩트 테이블은 측정값이 있는 열 및 차원 테이블에 대한 외래 키가 있는 열과 같이 두 가지 열 유형을 포함합니다.

빠른 실패

개발 수명 주기를 줄이기 위해 빈번한 증분 테스트를 사용하는 철학입니다. 애자일 접근 방식의 핵심입니다.

장애 격리 경계

에서 장애의 영향을 제한하고 워크로드의 복원력을 개선하는 데 도움이 되는 가용 영역, AWS 리전 컨트롤 플레인 또는 데이터 플레인과 같은 AWS 클라우드경계입니다. 자세한 내용은 AWS 장애 격리 경계를 참조하세요.

기능 브랜치

브랜치를 참조하세요.

기능

예측에 사용하는 입력 데이터입니다. 예를 들어, 제조 환경에서 기능은 제조 라인에서 주기적으로 캡처되는 이미지일 수 있습니다.

기능 중요도

모델의 예측에 특성이 얼마나 중요한지를 나타냅니다. 이는 일반적으로 SHAP(Shapley Additive Descriptions) 및 통합 그레디언트와 같은 다양한 기법을 통해 계산할 수 있는 수치 점수로 표현됩니다. 자세한 내용은 기계 학습 모델 해석 가능성을 참조하세요 AWS.

기능 변환

추가 소스로 데이터를 보강하거나, 값을 조정하거나, 단일 데이터 필드에서 여러 정보 세트를 추출하는 등 ML 프로세스를 위해 데이터를 최적화하는 것입니다. 이를 통해 ML 모델이 데이터를 활용

할 수 있습니다. 예를 들어, 날짜 '2021-05-27 00:15:37'을 '2021년', '5월', '목', '15일'로 분류하면 학습 알고리즘이 다양한 데이터 구성 요소와 관련된 미묘한 패턴을 학습하는 데 도움이 됩니다.

퓨샷 프롬프팅

유사한 태스크를 수행하도록 요청하기 전에 LLM에 태스크와 원하는 출력을 보여주는 몇 가지 예제를 제공합니다. 이 기법은 모델이 프롬프트에 포함된 예제(샷)에서 학습하는 컨텍스트 내 학습을 적용합니다. 퓨샷 프롬프팅은 특정 형식 지정, 추론 또는 분야별 지식이 필요한 태스크에 효과적일 수 있습니다. 제로샷 프롬프팅도 참조하세요.

FGAC

세분화된 액세스 제어를 참조하세요.

세분화된 액세스 제어(FGAC)

여러 조건을 사용하여 액세스 요청을 허용하거나 거부합니다.

플래시컷 마이그레이션

단계적 접근 방식을 사용하는 대신 변경 데이터 캡처를 통해 지속적 데이터 복제를 사용하여 최단 시간에 데이터를 마이그레이션하는 데이터베이스 마이그레이션 방법입니다. 목표는 가동 중지 시간을 최소화하는 것입니다.

FM

파운데이션 모델을 참조하세요.

파운데이션 모델(FM)

일반화되고 레이블이 지정되지 않은 데이터의 대규모 데이터세트에서 훈련된 대규모 딥 러닝 신경망입니다. FM은 언어 이해, 텍스트 및 이미지 생성, 자연어 대화와 같은 다양한 일반 태스크를 수행할 수 있습니다. 자세한 내용은 파운데이션 모델이란?을 참조하세요.

G

생성형 AI

대량의 데이터에서 훈련되었으며 간단한 텍스트 프롬프트를 사용하여 이미지, 비디오, 텍스트, 오디오와 같은 새 콘텐츠와 아티팩트를 생성할 수 있는 AI 모델의 하위 세트입니다. 자세한 내용은 생성형 AI란 무엇인가요?를 참조하세요.

지리적 차단

지리적 제한을 참조하세요.

지리적 제한(지리적 차단)

Amazon CloudFront에서 특정 국가의 사용자가 콘텐츠 배포에 액세스하지 못하도록 하는 옵션입니다. 허용 목록 또는 차단 목록을 사용하여 승인된 국가와 차단된 국가를 지정할 수 있습니다. 자세한 내용은 CloudFront 설명서의 콘텐츠의 지리적 배포 제한을 참조하십시오.

Gitflow 워크플로

하위 환경과 상위 환경이 소스 코드 리포지토리의 서로 다른 브랜치를 사용하는 방식입니다. Gitflow 워크플로는 레거시로 간주되며 트렁크 기반 워크플로는 선호되는 현대적 접근 방식입니다.

골든 이미지

시스템 또는 소프트웨어의 새 인스턴스를 배포하기 위한 템플릿으로 사용되는 해당 시스템 또는 소프트웨어의 스냅샷입니다. 예를 들어 제조 분야에서는 골든 이미지를 사용하여 여러 디바이스에서 소프트웨어를 프로비저닝할 수 있으며 이를 통해 딥이스 제조 작업의 속도, 확장성 및 생산성을 개선할 수 있습니다.

브라운필드 전략

새로운 환경에서 기존 인프라의 부재 시스템 아키텍처에 대한 그린필드 전략을 채택할 때 브라운필드라고도 하는 기존 인프라와의 호환성 제한 없이 모든 새로운 기술을 선택할 수 있습니다. 기존 인프라를 확장하는 경우 브라운필드 전략과 그린필드 전략을 혼합할 수 있습니다.

가드레일

조직 단위(OU) 전체에서 리소스, 정책 및 규정 준수를 관리하는 데 도움이 되는 중요 규칙입니다. 예방 가드레일은 규정 준수 표준에 부합하도록 정책을 시행하며, 서비스 제어 정책과 IAM 권한 경계를 사용하여 구현됩니다. 탐지 가드레일은 정책 위반 및 규정 준수 문제를 감지하고 해결을 위한 알림을 생성하며, 이는 AWS Config Amazon GuardDuty AWS Security Hub CSPM, , AWS Trusted Advisor Amazon Inspector 및 사용자 지정 AWS Lambda 검사를 사용하여 구현됩니다.

H

HA

고가용성을 참조하세요.

이기종 데이터베이스 마이그레이션

다른 데이터베이스 엔진을 사용하는 대상 데이터베이스로 소스 데이터베이스 마이그레이션(예: Oracle에서 Amazon Aurora로) 이기종 마이그레이션은 일반적으로 리아키텍트 작업의 일부이며 스

키마를 변환하는 것은 복잡한 작업일 수 있습니다.AWS 는 스키마 변환에 도움이 되는 AWS SCT를 제공합니다.

높은 가용성(HA)

문제나 재해 발생 시 개입 없이 지속적으로 운영할 수 있는 워크로드의 능력. HA 시스템은 자동으로 장애 조치되고, 지속적으로 고품질 성능을 제공하고, 성능에 미치는 영향을 최소화하면서 다양한 부하와 장애를 처리하도록 설계되었습니다.

히스토리언 현대화

제조 산업의 요구 사항을 더 잘 충족하도록 운영 기술(OT) 시스템을 현대화하고 업그레이드하는 데 사용되는 접근 방식입니다. 히스토리언은 공장의 다양한 출처에서 데이터를 수집하고 저장하는 데 사용되는 일종의 데이터베이스입니다.

홀드아웃 데이터

기계 학습 모델을 훈련하는 데 사용되는 데이터세트에서 보류되는 레이블이 지정된 기록 데이터의 일부입니다. 홀드아웃 데이터를 사용하여 모델 예측을 홀드아웃 데이터와 비교해 모델 성능을 평가할 수 있습니다.

동종 데이터베이스 마이그레이션

동일한 데이터베이스 엔진을 공유하는 대상 데이터베이스로 소스 데이터베이스 마이그레이션(예: Microsoft SQL Server에서 Amazon RDS for SQL Server로) 동종 마이그레이션은 일반적으로 리호스팅 또는 리플랫포밍 작업의 일부입니다. 네이티브 데이터베이스 유틸리티를 사용하여 스키마를 마이그레이션할 수 있습니다.

핫 데이터

자주 액세스하는 데이터(예: 실시간 데이터 또는 최근 번역 데이터). 일반적으로 이 데이터에는 빠른 쿼리 응답을 제공하기 위한 고성능 스토리지 계층 또는 클래스가 필요합니다.

핫픽스

프로덕션 환경의 중요한 문제를 해결하기 위한 긴급 수정입니다. 핫픽스는 긴급하기 때문에 일반적인 DevOps 릴리스 워크플로 외부에서 실행됩니다.

하이퍼케어 기간

전환 직후 마이그레이션 팀이 문제를 해결하기 위해 클라우드에서 마이그레이션된 애플리케이션을 관리하고 모니터링하는 기간입니다. 일반적으로 이 기간은 1~4일입니다. 하이퍼케어 기간이 끝나면 마이그레이션 팀은 일반적으로 애플리케이션에 대한 책임을 클라우드 운영 팀에 넘깁니다.

I

IaC

코드형 인프라를 참조하세요.

자격 증명 기반 정책

AWS 클라우드 환경 내에서 권한을 정의하는 하나 이상의 IAM 보안 주체에 연결된 정책입니다.

유휴 애플리케이션

90일 동안 평균 CPU 및 메모리 사용량이 5~20%인 애플리케이션입니다. 마이그레이션 프로젝트에서는 이러한 애플리케이션을 사용 중지하거나 온프레미스에 유지하는 것이 일반적입니다.

IIoT

산업용 사물 인터넷을 참조하세요.

변경 불가능한 인프라

기존 인프라를 업데이트, 패치 또는 수정하는 대신 프로덕션 워크로드에 대한 새 인프라를 배포하는 모델입니다. 변경 불가능한 인프라는 변경 가능한 인프라보다 본질적으로 더 일관되고 안정적이며 예측 가능합니다. 자세한 내용은 AWS Well-Architected Framework의 변경 불가능한 인프라를 사용하여 배포 모범 사례를 참조하세요.

인바운드(수신) VPC

AWS 다중 계정 아키텍처에서 애플리케이션 외부에서 네트워크 연결을 수락, 검사 및 라우팅하는 VPC입니다. AWS Security Reference Architecture에서는 애플리케이션과 더 넓은 인터넷 간의 양방향 인터페이스를 보호하기 위해 인바운드, 아웃바운드 및 검사 VPC로 네트워크 계정을 설정할 것을 권장합니다.

증분 마이그레이션

한 번에 전체 전환을 수행하는 대신 애플리케이션을 조금씩 마이그레이션하는 전환 전략입니다. 예를 들어, 처음에는 소수의 마이크로서비스나 사용자만 새 시스템으로 이동할 수 있습니다. 모든 것이 제대로 작동하는지 확인한 후에는 레거시 시스템을 폐기할 수 있을 때까지 추가 마이크로서비스 또는 사용자를 점진적으로 이동할 수 있습니다. 이 전략을 사용하면 대규모 마이그레이션과 관련된 위험을 줄일 수 있습니다.

Industry 4.0

연결성, 실시간 데이터, 자동화, 분석 및 AI/ML의 발전을 통해 제조 프로세스의 현대화를 나타내기 위해 2016년에 Klaus Schwab에서 도입한 용어입니다.

인프라

애플리케이션의 환경 내에 포함된 모든 리소스와 자산입니다.

코드형 인프라(IaC)

구성 파일 세트를 통해 애플리케이션의 인프라를 프로비저닝하고 관리하는 프로세스입니다. IaC는 새로운 환경의 반복 가능성, 신뢰성 및 일관성을 위해 인프라 관리를 중앙 집중화하고, 리소스를 표준화하고, 빠르게 확장할 수 있도록 설계되었습니다.

산업용 사물 인터넷(IIoT)

제조, 에너지, 자동차, 의료, 생명과학, 농업 등의 산업 부문에서 인터넷에 연결된 센서 및 디바이스의 사용 자세한 내용은 산업용 사물 인터넷(IoT) 디지털 트랜스포메이션 전략 구축을 참조하십시오.

검사 VPC

AWS 다중 계정 아키텍처에서는 VPC(동일하거나 다른 AWS 리전), 인터넷 및 온프레미스 네트워크 간의 네트워크 트래픽 검사를 관리하는 중앙 집중식 VPCs입니다. AWS Security Reference Architecture에서는 애플리케이션과 더 넓은 인터넷 간의 양방향 인터페이스를 보호하기 위해 인바운드, 아웃바운드 및 검사 VPC로 네트워크 계정을 설정할 것을 권장합니다.

사물 인터넷(IoT)

인터넷이나 로컬 통신 네트워크를 통해 다른 디바이스 및 시스템과 통신하는 센서 또는 프로세서가 내장된 연결된 물리적 객체의 네트워크 자세한 내용은 IoT란?을 참조하십시오.

해석력

모델의 예측이 입력에 따라 어떻게 달라지는지를 사람이 이해할 수 있는 정도를 설명하는 기계 학습 모델의 특성입니다. 자세한 내용은 기계 학습 모델 해석 가능성을 참조하세요 AWS.

IoT

사물 인터넷을 참조하세요.

IT 정보 라이브러리(ITIL)

IT 서비스를 제공하고 이러한 서비스를 비즈니스 요구 사항에 맞게 조정하기 위한 일련의 모범 사례 ITIL은 ITSM의 기반을 제공합니다.

IT 서비스 관리(ITSM)

조직의 IT 서비스 설계, 구현, 관리 및 지원과 관련된 활동 클라우드 운영을 ITSM 도구와 통합하는 방법에 대한 자세한 내용은 운영 통합 가이드를 참조하십시오.

ITIL

IT 정보 라이브러리를 참조하세요.

ITSM

IT 서비스 관리를 참조하세요.

L

레이블 기반 액세스 제어(LBAC)

사용자 및 데이터 자체에 각각 보안 레이블 값을 명시적으로 할당하는 필수 액세스 제어(MAC)를 구현한 것입니다. 사용자 보안 레이블과 데이터 보안 레이블 간의 교차 부분에 따라 사용자가 볼 수 있는 행과 열이 결정됩니다.

랜딩 존

랜딩 존은 확장 가능하고 안전한 잘 설계된 다중 계정 AWS 환경입니다. 조직은 여기에서부터 보안 및 인프라 환경에 대한 확신을 가지고 워크로드와 애플리케이션을 신속하게 시작하고 배포할 수 있습니다. 랜딩 존에 대한 자세한 내용은 안전하고 확장 가능한 다중 계정 AWS 환경 설정을 참조하십시오.

대규모 언어 모델(LLM)

방대한 양의 데이터에서 사전 훈련된 딥 러닝 AI 모델입니다. LLM은 질문에 대한 답변, 문서 요약, 텍스트를 다른 언어로 번역, 문장 완성과 같은 여러 태스크를 수행할 수 있습니다. 자세한 내용은 대규모 언어 모델(LLM)이란 무엇인가요?를 참조하세요.

대규모 마이그레이션

300대 이상의 서버 마이그레이션입니다.

LBAC

레이블 기반 액세스 제어를 참조하세요.

최소 권한

작업을 수행하는 데 필요한 최소 권한을 부여하는 보안 모범 사례입니다. 자세한 내용은 IAM 설명서의 최소 권한 적용을 참조하십시오.

리프트 앤드 시프트

7R을 참조하세요.

리틀 엔디안 시스템

가장 덜 중요한 바이트를 먼저 저장하는 시스템입니다. 엔디안도 참조하세요.

LLM

대규모 언어 모델을 참조하세요.

하위 환경

환경을 참조하세요.

M

기계 학습(ML)

패턴 인식 및 학습에 알고리즘과 기법을 사용하는 인공 지능의 한 유형입니다. ML은 사물 인터넷(IoT) 데이터와 같은 기록된 데이터를 분석하고 학습하여 패턴을 기반으로 통계 모델을 생성합니다. 자세한 내용은 기계 학습을 참조하십시오.

기본 브랜치

브랜치를 참조하세요.

맬웨어

컴퓨터 보안 또는 프라이버시를 위협하도록 설계된 소프트웨어입니다. 맬웨어는 컴퓨터 시스템을 방해하거나 민감한 정보를 유출하거나 무단 액세스 권한을 확보할 수 있습니다. 맬웨어의 예로 바이러스, 웜, 랜섬웨어, 트로이 목마, 스파이웨어, 키로거 등이 있습니다.

관리형 서비스

AWS 서비스 는 인프라 계층, 운영 체제 및 플랫폼을 AWS 운영하고, 사용자는 엔드포인트에 액세스하여 데이터를 저장하고 검색합니다. 관리형 서비스의 예로 Amazon Simple Storage Service(Amazon S3) 및 Amazon DynamoDB가 있습니다. 이를 추상화된 서비스라고도 합니다.

제조 실행 시스템(MES)

원자재를 생산 현장에서 완제품으로 변환하는 생산 프로세스를 추적, 모니터링, 문서화 및 제어하기 위한 소프트웨어 시스템입니다.

MAP

Migration Acceleration Program을 참조하세요.

메커니즘

도구를 생성하고 도구 채택을 유도한 다음 조정을 위해 결과를 검사하는 전체 프로세스입니다. 메커니즘은 작동 시 자체적으로 강화하고 개선하는 주기입니다. 자세한 내용은 AWS Well-Architected Framework의 메커니즘 구축을 참조하세요.

멤버 계정

조직의 일부인 관리 계정을 AWS 계정 제외한 모든 계정. AWS Organizations하나의 계정은 한 번에 하나의 조직 멤버만 될 수 있습니다.

MES

제조 실행 시스템을 참조하세요.

메시지 큐 원격 분석 전송(MQTT)

리소스 제약이 있는 IoT 디바이스에 대한 게시 및 구독 패턴을 기반으로 하는 경량 Machine-to-Machine(M2M) 통신 프로토콜입니다.

마이크로서비스

잘 정의된 API를 통해 통신하고 일반적으로 소규모 자체 팀이 소유하는 소규모 독립 서비스입니다. 예를 들어, 보험 시스템에는 영업, 마케팅 등의 비즈니스 역량이나 구매, 청구, 분석 등의 하위 영역에 매핑되는 마이크로 서비스가 포함될 수 있습니다. 마이크로서비스의 이점으로 민첩성, 유연한 확장, 손쉬운 배포, 재사용 가능한 코드, 복원력 등이 있습니다. 자세한 내용은 AWS 서버리스 서비스를 사용하여 마이크로서비스 통합을 참조하세요.

마이크로서비스 아키텍처

각 애플리케이션 프로세스를 마이크로서비스로 실행하는 독립 구성 요소를 사용하여 애플리케이션을 구축하는 접근 방식입니다. 이러한 마이크로서비스는 경량 API를 사용하여 잘 정의된 인터페이스를 통해 통신합니다. 애플리케이션의 특정 기능에 대한 수요에 맞게 이 아키텍처의 각 마이크로서비스를 업데이트, 배포 및 조정할 수 있습니다. 자세한 내용은 에서 마이크로서비스 구현을 참조하세요 AWS.

Migration Acceleration Program(MAP)

조직이 클라우드로 전환하기 위한 강력한 운영 기반을 구축하고 초기 마이그레이션 비용을 상쇄하는 데 도움이 되는 컨설팅 지원, 교육 및 서비스를 제공하는 AWS 프로그램입니다. MAP에는 레거시 마이그레이션을 체계적인 방식으로 실행하기 위한 마이그레이션 방법론과 일반적인 마이그레이션 시나리오를 자동화하고 가속화하는 도구 세트가 포함되어 있습니다.

대규모 마이그레이션

애플리케이션 포트폴리오의 대다수를 웨이브를 통해 클라우드로 이동하는 프로세스로, 각 웨이브에서 더 많은 애플리케이션이 더 빠른 속도로 이동합니다. 이 단계에서는 이전 단계에서 배운 모범 사례와 교훈을 사용하여 팀, 도구 및 프로세스의 마이그레이션 팩토리를 구현하여 자동화 및 민첩한 제공을 통해 워크로드 마이그레이션을 간소화합니다. 이것은 AWS 마이그레이션 전략의 세 번째 단계입니다.

마이그레이션 팩토리

자동화되고 민첩한 접근 방식을 통해 워크로드 마이그레이션을 간소화하는 다기능 팀입니다. 마이그레이션 팩토리 팀에는 일반적으로 스프린트에서 일하는 운영, 비즈니스 분석가 및 소유자, 마이그레이션 엔지니어, 개발자, DevOps 전문가가 포함됩니다. 엔터프라이즈 애플리케이션 포트폴리오의 20~50%는 공장 접근 방식으로 최적화할 수 있는 반복되는 패턴으로 구성되어 있습니다. 자세한 내용은 이 콘텐츠 세트의 클라우드 마이그레이션 팩토리 가이드와 마이그레이션 팩토리에 대한 설명을 참조하십시오.

마이그레이션 메타데이터

마이그레이션을 완료하는 데 필요한 애플리케이션 및 서버에 대한 정보 각 마이그레이션 패턴에는 서로 다른 마이그레이션 메타데이터 세트가 필요합니다. 마이그레이션 메타데이터의 예로는 대상 서브넷, 보안 그룹 및 AWS 계정이 있습니다.

마이그레이션 패턴

사용되는 마이그레이션 전략, 마이그레이션 대상, 마이그레이션 애플리케이션 또는 서비스를 자세히 설명하는 반복 가능한 마이그레이션 작업입니다. 예: AWS Application Migration Service를 사용하여 Amazon EC2로 마이그레이션을 리호스팅합니다.

Migration Portfolio Assessment(MPA)

AWS 클라우드로 마이그레이션하는 비즈니스 사례를 검증하기 위한 정보를 제공하는 온라인 도구입니다. MPA는 상세한 포트폴리오 평가(서버 적정 규모 조정, 가격 책정, TCO 비교, 마이그레이션 비용 분석)와 마이그레이션 계획(애플리케이션 데이터 분석 및 데이터 수집, 애플리케이션 그룹화, 마이그레이션 우선순위 지정, 웨이브 계획)을 제공합니다. MPA 도구(로그인 필요)는 모든 AWS 컨설턴트와 APN 파트너 컨설턴트가 무료로 사용할 수 있습니다.

마이그레이션 준비 상태 평가(MRA)

AWS CAF를 사용하여 조직의 클라우드 준비 상태에 대한 인사이트를 얻고, 강점과 약점을 식별하고, 식별된 격차를 해소하기 위한 행동 계획을 수립하는 프로세스입니다. 자세한 내용은 마이그레이션 준비 가이드를 참조하십시오. MRA는 AWS 마이그레이션 전략의 첫 번째 단계입니다.

마이그레이션 전략

워크로드를 AWS 클라우드로 마이그레이션하는 데 사용되는 접근 방식입니다. 자세한 내용은 이 용어집의 7R 항목과 조직을 동원하여 대규모 마이그레이션 가속화를 참조하세요.

ML

기계 학습을 참조하세요.

현대화

비용을 절감하고 효율성을 높이고 혁신을 활용하기 위해 구식(레거시 또는 모놀리식) 애플리케이션과 해당 인프라를 클라우드의 민첩하고 탄력적이고 가용성이 높은 시스템으로 전환하는 것입니다. 자세한 내용은 AWS 클라우드에서 애플리케이션을 현대화하기 위한 전략을 참조하세요.

현대화 준비 상태 평가

조직 애플리케이션의 현대화 준비 상태를 파악하고, 이점, 위험 및 종속성을 식별하고, 조직이 해당 애플리케이션의 향후 상태를 얼마나 잘 지원할 수 있는지를 확인하는 데 도움이 되는 평가입니다. 평가 결과는 대상 아키텍처의 청사진, 현대화 프로세스의 개발 단계와 마일스톤을 자세히 설명하는 로드맵 및 파악된 격차를 해소하기 위한 실행 계획입니다. 자세한 내용은 AWS 클라우드에서 애플리케이션의 현대화 준비 상태 평가를 참조하세요.

모놀리식 애플리케이션(모놀리식 유형)

긴밀하게 연결된 프로세스를 사용하여 단일 서비스로 실행되는 애플리케이션입니다. 모놀리식 애플리케이션에는 몇 가지 단점이 있습니다. 한 애플리케이션 기능에 대한 수요가 급증하면 전체 아키텍처 규모를 조정해야 합니다. 코드 베이스가 커지면 모놀리식 애플리케이션의 기능을 추가하거나 개선하는 것도 더 복잡해집니다. 이러한 문제를 해결하기 위해 마이크로서비스 아키텍처를 사용할 수 있습니다. 자세한 내용은 마이크로서비스로 모놀리식 유형 분해를 참조하십시오.

MPA

Migration Portfolio Assessment를 참조하세요.

MQTT

메시지 큐 원격 분석 전송을 참조하세요.

멀티클래스 분류

여러 클래스에 대한 예측(2개 이상의 결과 중 하나 예측)을 생성하는 데 도움이 되는 프로세스입니다. 예를 들어, ML 모델이 '이 제품은 책인가요, 자동차인가요, 휴대폰인가요?' 또는 '이 고객이 가장 관심을 갖는 제품 범주는 무엇인가요?'라고 물을 수 있습니다.

변경 가능한 인프라

프로덕션 워크로드에 대한 기존 인프라를 업데이트하고 수정하는 모델입니다. 일관성, 신뢰성 및 예측 가능성을 높이기 위해 AWS Well-Architected Framework에서는 변경 불가능한 인프라를 모범 사례로 사용할 것을 권장합니다.

O

OAC

오리진 액세스 제어를 참조하세요.

OAI

오리진 액세스 ID를 참조하세요.

OCM

조직 변경 관리를 참조하세요.

오프라인 마이그레이션

마이그레이션 프로세스 중 소스 워크로드가 중단되는 마이그레이션 방법입니다. 이 방법은 가동 중지 증가를 수반하며 일반적으로 작고 중요하지 않은 워크로드에 사용됩니다.

OI

운영 통합을 참조하세요.

OLA

운영 수준 계약을 참조하세요.

온라인 마이그레이션

소스 워크로드를 오프라인 상태로 전환하지 않고 대상 시스템에 복사하는 마이그레이션 방법입니다. 워크로드에 연결된 애플리케이션은 마이그레이션 중에도 계속 작동할 수 있습니다. 이 방법은 가동 중지 차단 또는 최소화를 수반하며 일반적으로 중요한 프로덕션 워크로드에 사용됩니다.

OPC-UA

Open Process Communications - Unified Architecture(OPC-UA)를 참조하세요.

Open Process Communications - Unified Architecture(OPC-UA)

산업 자동화를 위한 Machine-to-Machine(M2M) 통신 프로토콜입니다. OPC-UA는 데이터 암호화, 인증 및 권한 부여 체계에 관한 상호 운용성 표준을 제공합니다.

운영 수준 협약(OLA)

서비스 수준에 관한 계약(SLA)을 지원하기 위해 직무 IT 그룹이 서로에게 제공하기로 약속한 내용을 명확히 하는 계약입니다.

운영 준비 상태 검토(ORR)

인시던트 및 잠재적 장애의 범위를 이해, 평가 또는 예방하거나 줄이는 데 도움이 되는 질문 체크리스트 및 관련 모범 사례입니다. 자세한 내용은 AWS Well-Architected Framework의 운영 준비 상태 검토(ORR)를 참조하세요.

운영 기술(OT)

물리적 환경에서 작동하여 산업 운영, 장비 및 인프라를 제어하는 하드웨어 및 소프트웨어 시스템입니다. 제조 분야에서 OT 및 정보 기술(IT) 시스템의 통합은 Industry 4.0 트랜스포메이션의 주요 중점 사항입니다.

운영 통합(OI)

클라우드에서 운영을 현대화하는 프로세스로 준비 계획, 자동화 및 통합을 수반합니다. 자세한 내용은 운영 통합 가이드를 참조하십시오.

조직 트레일

조직 AWS 계정 내 모든에 대한 모든 이벤트를 로깅 AWS CloudTrail 하는에서 생성된 추적입니다 AWS Organizations. 이 트레일은 조직에 속한 각 AWS 계정 에 생성되고 각 계정의 활동을 추적합니다. 자세한 내용은 CloudTrail 설명서의 Creating a trail for an organization을 참조하십시오.

조직 변경 관리(OCM)

사람, 문화 및 리더십 관점에서 중대하고 파괴적인 비즈니스 혁신을 관리하기 위한 프레임워크입니다. OCM은 변화 채택을 가속화하고, 과도기적 문제를 해결하고, 문화 및 조직적 변화를 주도함으로써 조직이 새로운 시스템 및 전략을 준비하고 전환할 수 있도록 지원합니다. AWS 마이그레이션 전략에서는 클라우드 채택 프로젝트에 필요한 변경 속도 때문에이 프레임워크를 인력 가속화라고 합니다. 자세한 내용은 사용 가이드를 참조하십시오.

오리진 액세스 제어(OAC)

CloudFront에서 Amazon Simple Storage Service(S3) 콘텐츠를 보호하기 위해 액세스를 제한하는 고급 옵션입니다. OAC는 AWS KMS (SSE-KMS)를 사용한 모든 서버 측 암호화 AWS 리전와 S3 버킷에 대한 동적 PUT 및 DELETE 요청에서 모든 S3 버킷을 지원합니다.

오리진 액세스 ID(OAI)

CloudFront에서 Amazon S3 콘텐츠를 보호하기 위해 액세스를 제한하는 옵션입니다. OAI를 사용하면 CloudFront는 Amazon S3가 인증할 수 있는 보안 주체를 생성합니다. 인증된 보안 주체는 특

정 CloudFront 배포를 통해서만 S3 버킷의 콘텐츠에 액세스할 수 있습니다. 더 세분화되고 향상된 액세스 제어를 제공하는 OAC도 참조하십시오.

ORR

운영 준비 상태 검토를 참조하세요.

OT

운영 기술을 참조하세요.

아웃바운드(송신) VPC

AWS 다중 계정 아키텍처에서 애플리케이션 내에서 시작된 네트워크 연결을 처리하는 VPC입니다. AWS Security Reference Architecture에서는 애플리케이션과 더 넓은 인터넷 간의 양방향 인터페이스를 보호하기 위해 인바운드, 아웃바운드 및 검사 VPC로 네트워크 계정을 설정할 것을 권장합니다.

P

권한 경계

사용자나 역할이 가질 수 있는 최대 권한을 설정하기 위해 IAM 보안 주체에 연결되는 IAM 관리 정책입니다. 자세한 내용은 IAM 설명서의 권한 경계를 참조하십시오.

개인 식별 정보(PII)

직접 보거나 다른 관련 데이터와 함께 짝을 지을 때 개인의 신원을 합리적으로 추론하는 데 사용할 수 있는 정보입니다. PII의 예로는 이름, 주소, 연락처 정보 등이 있습니다.

PII

개인 식별 정보를 참조하세요.

플레이북

클라우드에서 핵심 운영 기능을 제공하는 등 마이그레이션과 관련된 작업을 캡처하는 일련의 사전 정의된 단계입니다. 플레이북은 스크립트, 자동화된 런북 또는 현대화된 환경을 운영하는 데 필요한 프로세스나 단계 요약의 형태를 취할 수 있습니다.

PLC

프로그래밍 가능 로직 컨트롤러를 참조하세요.

PLM

제품 수명 주기 관리를 참조하세요.

정책

권한 정의(ID 기반 정책 참조), 액세스 조건 지정(리소스 기반 정책 참조), AWS Organizations 내 조직의 모든 계정에 대한 최대 권한 정의(서비스 제어 정책 참조)와 같은 작업을 수행할 수 있는 객체입니다.

다국어 지속성

데이터 액세스 패턴 및 기타 요구 사항을 기반으로 독립적으로 마이크로서비스의 데이터 스토리지 기술 선택. 마이크로서비스가 동일한 데이터 스토리지 기술을 사용하는 경우 구현 문제가 발생하거나 성능이 저하될 수 있습니다. 요구 사항에 가장 적합한 데이터 저장소를 사용하면 마이크로서비스를 더 쉽게 구현하고 성능과 확장성을 높일 수 있습니다.

포트폴리오 평가

마이그레이션을 계획하기 위해 애플리케이션 포트폴리오를 검색 및 분석하고 우선순위를 정하는 프로세스입니다. 자세한 내용은 마이그레이션 준비 상태 평가를 참조하십시오.

조건자

보통 `WHERE` 절에 있는 `true` 또는 `false`를 반환하는 쿼리 조건입니다.

푸시다운 조건자

전송 전에 쿼리의 데이터를 필터링하는 데이터베이스 쿼리 최적화 기법입니다. 이렇게 하면 관계형 데이터베이스에서 검색하고 처리해야 하는 데이터의 양이 줄고 쿼리 성능이 향상됩니다.

예방적 제어

이벤트 발생을 방지하도록 설계된 보안 제어입니다. 이 제어는 네트워크에 대한 무단 액세스나 원치 않는 변경을 방지하는 데 도움이 되는 1차 방어선입니다. 자세한 내용은 Implementing security controls on AWS의 Preventative controls를 참조하십시오.

보안 주체

작업을 수행하고 리소스에 액세스할 수 AWS 있는의 엔터티입니다. 이 엔터티는 일반적으로 , AWS 계정 IAM 역할 또는 사용자의 루트 사용자입니다. 자세한 내용은 IAM 설명서의 역할 용어 및 개념의 보안 주체를 참조하십시오.

개인 정보 보호 중심 설계

전체 개발 프로세스에서 개인 정보를 고려하는 시스템 엔지니어링에서의 접근 방식입니다.

프라이빗 호스팅 영역

Amazon Route 53에서 하나 이상의 VPC 내 도메인과 하위 도메인에 대한 DNS 쿼리에 응답하는 방법에 대한 정보가 담긴 컨테이너입니다. 자세한 내용은 Route 53 설명서의 프라이빗 호스팅 영역 작업을 참조하십시오.

선제적 제어

규정 미준수 리소스의 배포를 방지하도록 설계된 보안 제어입니다. 이러한 제어는 리소스를 프로비저닝하기 전에 리소스를 스캔합니다. 리소스가 제어를 준수하지 않으면 프로비저닝되지 않습니다. 자세한 내용은 AWS Control Tower 설명서의 제어 참조 가이드를 참조하고 보안 제어 구현의 사전 예방적 제어를 참조하세요. AWS

제품 수명 주기 관리(PLM)

설계, 개발 및 출시부터 성장 및 성숙도를 거쳐 거부 및 제거에 이르기까지 전체 수명 주기 동안 제품의 데이터 및 프로세스 관리를 나타냅니다.

프로덕션 환경

환경을 참조하세요.

프로그래밍 가능 로직 컨트롤러(PLC)

제조 분야에서 기계를 모니터링하고 제조 프로세스를 자동화하는 매우 안정적이고 적응력이 뛰어난 컴퓨터입니다.

프롬프트 체이닝

한 LLM 프롬프트의 출력을 다음 프롬프트의 입력으로 사용하여 더 나은 응답을 생성합니다. 이 기법은 복잡한 태스크를 하위 태스크로 나누거나 예비 응답을 반복적으로 세부 조정하거나 확장하는 데 사용됩니다. 이를 통해 모델 응답의 정확성과 관련성을 개선하고 보다 세분화되고 개인화된 결과를 얻을 수 있습니다.

가명화

데이터세트의 개인 식별자를 자리 표시자 값으로 바꾸는 프로세스입니다. 가명화는 개인 정보를 보호하는 데 도움이 될 수 있습니다. 가명화된 데이터는 여전히 개인 데이터로 간주됩니다.

게시/구독(pub/sub)

여러 마이크로서비스에서 비동기 통신을 지원하여 확장성과 응답성을 개선하는 패턴입니다. 예를 들어 마이크로서비스 기반 MES에서 마이크로서비스는 다른 마이크로서비스가 구독할 수 있는 채널에 이벤트 메시지를 게시할 수 있습니다. 시스템은 게시 서비스를 변경하지 않고도 새 마이크로서비스를 추가할 수 있습니다.

Q

쿼리 계획

SQL 관계형 데이터베이스 시스템의 데이터에 액세스하는 데 사용되는 명령어와 같은 일련의 단계입니다.

쿼리 계획 회귀

데이터베이스 서비스 최적화 프로그램이 데이터베이스 환경을 변경하기 전보다 덜 최적의 계획을 선택하는 경우입니다. 통계, 제한 사항, 환경 설정, 쿼리 파라미터 바인딩 및 데이터베이스 엔진 업데이트의 변경으로 인해 발생할 수 있습니다.

R

RACI 매트릭스

Responsible, Accountable, Consulted, Informed(RACI)를 참조하세요.

RAG

검색 증강 생성을 참조하세요.

랜섬웨어

결제가 완료될 때까지 컴퓨터 시스템이나 데이터에 대한 액세스를 차단하도록 설계된 악성 소프트웨어입니다.

RASCI 매트릭스

Responsible, Accountable, Consulted, Informed(RACI)를 참조하세요.

RCAC

행 및 열 액세스 제어를 참조하세요.

읽기 전용 복제본

읽기 전용 용도로 사용되는 데이터베이스의 사본입니다. 쿼리를 읽기 전용 복제본으로 라우팅하여 기본 데이터베이스의 로드를 줄일 수 있습니다.

리아키텍팅

7R을 참조하세요.

Recovery Point Objective(RPO)

마지막 데이터 복구 시점 이후 허용되는 최대 시간입니다. 이에 따라 마지막 복구 시점과 서비스 중단 사이에 허용되는 데이터 손실로 간주되는 범위가 결정됩니다.

Recovery Time Objective(RTO)

서비스 중단과 서비스 복원 사이의 허용 가능한 지연 시간입니다.

리팩터링

7R을 참조하세요.

리전

지리적 영역의 AWS 리소스 모음입니다. 각 AWS 리전 는 내결함성, 안정성 및 복원력을 제공하기 위해 서로 격리되고 독립적입니다. 자세한 내용은 계정에서 사용할 수 있는 AWS 리전 지정을 참조하세요.

회귀

숫자 값을 예측하는 ML 기법입니다. 예를 들어, '이 집은 얼마에 팔릴까?'라는 문제를 풀기 위해 ML 모델은 선형 회귀 모델을 사용하여 주택에 대해 알려진 사실(예: 면적)을 기반으로 주택의 매매 가격을 예측할 수 있습니다.

리호스팅

7R을 참조하세요.

릴리스

배포 프로세스에서 변경 사항을 프로덕션 환경으로 승격시키는 행위입니다.

재배치

7R을 참조하세요.

리플랫포밍

7R을 참조하세요.

재구매

7R을 참조하세요.

복원력

중단에 저항하거나 중단을 복구할 수 있는 애플리케이션의 기능입니다. 고가용성 및 재해 복구는 AWS 클라우드에서 복원력을 계획할 때 일반적인 고려 사항입니다. 자세한 내용은 AWS 클라우드 복원력을 참조하세요.

리소스 기반 정책

Amazon S3 버킷, 엔드포인트, 암호화 키 등의 리소스에 연결된 정책입니다. 이 유형의 정책은 액세스가 허용된 보안 주체, 지원되는 작업 및 충족해야 하는 기타 조건을 지정합니다.

RACI(Responsible, Accountable, Consulted, Informed) 매트릭스

마이그레이션 활동 및 클라우드 운영에 참여하는 모든 당사자의 역할과 책임을 정의하는 매트릭스입니다. 매트릭스 이름은 매트릭스에 정의된 책임 유형에서 파생됩니다. 실무 담당자 (R), 의사 결정권자 (A), 업무 수행 조언자 (C), 결과 통보 대상자 (I). 지원자는 (S) 선택사항입니다. 지원자를 포함하면 매트릭스를 RASCI 매트릭스라고 하고, 지원자를 제외하면 RACI 매트릭스라고 합니다.

대응 제어

보안 기준에서 벗어나거나 부정적인 이벤트를 해결하도록 설계된 보안 제어입니다. 자세한 내용은 AWS에서 보안 제어 구현의 대응 제어를 참조하세요.

retain

7R을 참조하세요.

사용 중지

7R을 참조하세요.

검색 증강 세대(RAG)

응답을 생성하기 전에 LLM이 훈련 데이터 소스 외부에 있는 신뢰할 수 있는 데이터 소스를 참조하는 생성형 AI 기술입니다. 예를 들어 RAG 모델은 조직의 지식 기반 또는 사용자 지정 데이터에 대한 시맨틱 검색을 수행할 수 있습니다. 자세한 내용은 검색 증강 생성(RAG)이란 무엇인가요?를 참조하세요.

교체

공격자가 자격 증명에 액세스하는 것을 더욱 어렵게 만들기 위해 보안 암호를 주기적으로 업데이트하는 프로세스입니다.

행 및 열 액세스 제어(RCAC)

액세스 규칙이 정의된 기본적이고 유연한 SQL 표현식을 사용합니다. RCAC는 행 권한과 열 마스크로 구성됩니다.

RPO

목표 복구 시점(RPO)을 참조하세요.

RTO

목표 복구 시간(RTO)을 참조하세요.

런북

특정 작업을 수행하는 데 필요한 일련의 수동 또는 자동 절차입니다. 일반적으로 오류율이 높은 반복 작업이나 절차를 간소화하기 위해 런북을 만듭니다.

S

SAML 2.0

많은 ID 제공업체(idP)에서 사용하는 개방형 표준입니다. 이 기능을 사용하면 연동 SSO(Single Sign-On)를 AWS Management Console 사용할 수 있으므로 사용자는 조직의 모든 사용자에 대해 IAM에서 사용자를 생성하지 않고도에 로그인하거나 AWS API 작업을 호출할 수 있습니다. SAML 2.0 기반 페더레이션에 대한 자세한 내용은 IAM 설명서의 SAML 2.0 기반 페더레이션 정보를 참조하십시오.

SCADA

감독 제어 및 데이터 획득을 참조하세요.

SCP

서비스 제어 정책을 참조하세요.

보안 암호

에는 암호화된 형식으로 저장하는 암호 또는 사용자 자격 증명과 같은 AWS Secrets Manager기밀 또는 제한된 정보가 있습니다. 보안 암호 값과 메타데이터로 구성됩니다. 보안 암호 값은 바이너리, 단일 문자열 또는 여러 문자열일 수 있습니다. 자세한 내용은 AWS Secrets Manager 설명서의 Secrets Manager 보안 암호란 무엇인가요?를 참조하세요.

보안 중심 설계

전체 개발 프로세스에서 보안을 고려하는 시스템 엔지니어링에서의 접근 방식입니다.

보안 제어

위협 행위자가 보안 취약성을 악용하는 능력을 방지, 탐지 또는 감소시키는 기술적 또는 관리적 가드레일입니다. 보안 제어는 예방, 감지, 대응, 선제적과 같은 기본적인 네 가지 보안 제어 유형으로 구분됩니다.

보안 강화

공격 표면을 줄여 공격에 대한 저항력을 높이는 프로세스입니다. 더 이상 필요하지 않은 리소스 제거, 최소 권한 부여의 보안 모범 사례 구현, 구성 파일의 불필요한 기능 비활성화 등의 작업이 여기에 포함될 수 있습니다.

보안 정보 및 이벤트 관리(SIEM) 시스템

보안 정보 관리(SIM)와 보안 이벤트 관리(SEM) 시스템을 결합하는 도구 및 서비스입니다. SIEM 시스템은 서버, 네트워크, 디바이스 및 기타 소스에서 데이터를 수집, 모니터링 및 분석하여 위협과 보안 침해를 탐지하고 알림을 생성합니다.

보안 응답 자동화

보안 이벤트에 자동으로 응답하거나 이를 해결하도록 설계된 사전 정의되고 프로그래밍된 작업입니다. 이러한 자동화는 보안 모범 사례를 구현하는 데 도움이 되는 탐지 또는 대응 AWS 보안 제어 역할을 합니다. 자동화된 응답 작업의 예로 VPC 보안 그룹 수정, Amazon EC2 인스턴스 패치 적용 또는 자격 증명 교체 등이 있습니다.

서버 측 암호화

대상에서 데이터를 수신하는 AWS 서비스 에 의한 데이터 암호화.

서비스 제어 정책(SCP)

AWS Organizations에 속한 조직의 모든 계정에 대한 권한을 중앙 집중식으로 제어하는 정책입니다. SCP는 관리자가 사용자 또는 역할에 위임할 수 있는 작업에 대해 제한을 설정하거나 가드레일을 정의합니다. SCP를 허용 목록 또는 거부 목록으로 사용하여 허용하거나 금지할 서비스 또는 작업을 지정할 수 있습니다. 자세한 내용은 AWS Organizations 설명서의 서비스 제어 정책을 참조하세요.

서비스 엔드포인트

에 대한 진입점의 URL입니다 AWS 서비스. 엔드포인트를 사용하여 대상 서비스에 프로그래밍 방식으로 연결할 수 있습니다. 자세한 내용은 AWS 일반 참조의 AWS 서비스 엔드포인트를 참조하십시오.

서비스 수준에 관한 계약(SLA)

IT 팀이 고객에게 제공하기로 약속한 내용(예: 서비스 가동 시간 및 성능)을 명시한 계약입니다.

서비스 수준 지표(SLI)

오류 발생률, 가용성 또는 처리량과 같은 서비스의 성능 측면에 대한 측정값입니다.

서비스 수준 목표(SLO)

서비스 수준 지표로 측정되는 서비스의 상태를 나타내는 목표 지표입니다.

공동 책임 모델

클라우드 보안 및 규정 준수를 AWS 위해와 공유하는 책임을 설명하는 모델입니다. AWS 는 클라우드의 보안을 담당하는 반면, 사용자는 클라우드의 보안을 담당합니다. 자세한 내용은 공동 책임 모델을 참조하십시오.

SIEM

보안 정보 및 이벤트 관리 시스템을 참조하세요.

단일 장애점(SPOF)

애플리케이션을 중단시킬 수 있는 애플리케이션의 중요한 단일 구성 요소에서 발생하는 장애입니다.

SLA

서비스 수준 계약을 참조하세요.

SLI

서비스 수준 지표를 참조하세요.

SLO

서비스 수준 목표를 참조하세요.

분할 앤 시드 모델

현대화 프로젝트를 확장하고 가속화하기 위한 패턴입니다. 새로운 기능과 제품 릴리스가 정의되면 핵심 팀이 분할되어 새로운 제품 팀이 만들어집니다. 이를 통해 조직의 역량과 서비스 규모를 조정하고, 개발자 생산성을 개선하고, 신속한 혁신을 지원할 수 있습니다. 자세한 내용은 AWS 클라우드에서 애플리케이션을 현대화하기 위한 단계별 접근 방식을 참조하세요.

SPOF

단일 장애점을 참조하세요.

스타 스키마

하나의 큰 팩트 테이블을 사용하여 트랜잭션 또는 측정된 데이터를 저장하고 하나 이상의 더 작은 차원 테이블을 사용하여 데이터 속성을 저장하는 데이터베이스 조직 구조입니다. 이 구조는 데이터 웨어하우스에서 또는 비즈니스 인텔리전스 목적으로 사용하도록 설계되었습니다.

Strangler Fig 패턴

레거시 시스템을 폐기할 수 있을 때까지 시스템 기능을 점진적으로 다시 작성하고 교체하여 모놀리식 시스템을 현대화하기 위한 접근 방식. 이 패턴은 무화과 덩굴이 나무로 자라 결국 숙주를 압도하고 대체하는 것과 비슷합니다. Martin Fowler가 모놀리식 시스템을 다시 작성할 때 위험을 관리하는 방법으로 이 패턴을 도입했습니다. 이 패턴을 적용하는 방법의 예는 컨테이너 및 Amazon API Gateway를 사용하여 기존의 Microsoft ASP.NET(ASMX) 웹 서비스를 점진적으로 현대화하는 방법을 참조하십시오.

서브넷

VPC의 IP 주소 범위입니다. 서브넷은 단일 가용 영역에 상주해야 합니다.

감독 제어 및 데이터 획득(SCADA)

제조 분야에서 하드웨어와 소프트웨어를 사용하여 물리적 자산과 프로덕션 작업을 모니터링하는 시스템입니다.

대칭 암호화

동일한 키를 사용하여 데이터를 암호화하고 복호화하는 암호화 알고리즘입니다.

합성 테스트

사용자 상호 작용을 시뮬레이션하여 잠재적 문제를 감지하거나 성능을 모니터링하는 방식으로 진행되는 시스템 테스트입니다. Amazon CloudWatch Synthetics를 사용하여 이러한 테스트를 생성할 수 있습니다.

시스템 프롬프트

LLM에 컨텍스트, 명령 또는 지침을 제공하여 동작을 지시하는 기법입니다. 시스템 프롬프트는 컨텍스트를 설정하고 사용자와의 상호 작용을 위한 규칙을 설정하는 데 도움이 됩니다.

T

tags

AWS 리소스를 구성하기 위한 메타데이터 역할을 하는 키-값 페어입니다. 태그를 사용하면 리소스를 손쉽게 관리, 식별, 정리, 검색, 필터링할 수 있습니다. 자세한 내용은 AWS 리소스에 태그 지정을 참조하십시오.

대상 변수

지도 ML에서 예측하려는 값으로, 결과 변수라고도 합니다. 예를 들어, 제조 설정에서 대상 변수는 제품 결함일 수 있습니다.

작업 목록

런북을 통해 진행 상황을 추적하는 데 사용되는 도구입니다. 작업 목록에는 런북의 개요와 완료해야 할 일반 작업 목록이 포함되어 있습니다. 각 일반 작업에 대한 예상 소요 시간, 소유자 및 진행 상황이 작업 목록에 포함됩니다.

테스트 환경

환경을 참조하세요.

훈련

ML 모델이 학습할 수 있는 데이터를 제공하는 것입니다. 훈련 데이터에는 정답이 포함되어야 합니다. 학습 알고리즘은 훈련 데이터에서 대상(예측하려는 답)에 입력 데이터 속성을 매핑하는 패턴을 찾고, 이러한 패턴을 캡처하는 ML 모델을 출력합니다. 그런 다음 ML 모델을 사용하여 대상을 모르는 새 데이터에 대한 예측을 할 수 있습니다.

Transit Gateway

VPC와 온프레미스 네트워크를 상호 연결하는 데 사용할 수 있는 네트워크 전송 허브입니다. 자세한 내용은 AWS Transit Gateway 설명서의 전송 게이트웨이란 무엇입니까?를 참조하세요.

트렁크 기반 워크플로

개발자가 기능 브랜치에서 로컬로 기능을 구축하고 테스트한 다음 해당 변경 사항을 기본 브랜치에 병합하는 접근 방식입니다. 이후 기본 브랜치는 개발, 프로덕션 이전 및 프로덕션 환경에 순차적으로 구축됩니다.

신뢰할 수 있는 액세스

사용자를 대신하여 AWS Organizations 및 해당 계정에서 조직에서 작업을 수행하도록 지정하는 서비스에 대한 권한 부여. 신뢰할 수 있는 서비스는 필요할 때 각 계정에 서비스 연결 역할을 생성하여 관리 작업을 수행합니다. 자세한 내용은 설명서의 다른 AWS 서비스와 AWS Organizations 함께 사용을 참조하세요 AWS Organizations .

튜닝

ML 모델의 정확도를 높이기 위해 훈련 프로세스의 측면을 여러 변경하는 것입니다. 예를 들어, 레이블링 세트를 생성하고 레이블을 추가한 다음 다양한 설정에서 이러한 단계를 여러 번 반복하여 모델을 최적화하는 방식으로 ML 모델을 훈련할 수 있습니다.

피자 두 판 팀

피자 두 판이면 충분한 소규모 DevOps 팀. 피자 두 판 팀 규모는 소프트웨어 개발에 있어 가능한 최상의 공동 작업 기회를 보장합니다.

U

불확실성

예측 ML 모델의 신뢰성을 저해할 수 있는 부정확하거나 불완전하거나 알려지지 않은 정보를 나타내는 개념입니다. 불확실성에는 두 가지 유형이 있습니다. 인식론적 불확실성은 제한적이고 불완전한 데이터에 의해 발생하는 반면, 우연한 불확실성은 데이터에 내재된 노이즈와 무작위성에 의해 발생합니다. 자세한 내용은 Quantifying uncertainty in deep learning systems 가이드를 참조하십시오.

차별화되지 않은 작업

애플리케이션을 만들고 운영하는 데 필요하지만 최종 사용자에게 직접적인 가치를 제공하거나 경쟁 우위를 제공하지 못하는 작업을 헤비 리프팅이라고도 합니다. 차별화되지 않은 작업의 예로는 조달, 유지보수, 용량 계획 등이 있습니다.

상위 환경

환경을 참조하세요.

V

정리

스토리지를 회수하고 성능을 향상시키기 위해 증분 업데이트 후 정리 작업을 수반하는 데이터베이스 유지 관리 작업입니다.

버전 제어

리포지토리의 소스 코드 변경과 같은 변경 사항을 추적하는 프로세스 및 도구입니다.

VPC 피어링

프라이빗 IP 주소를 사용하여 트래픽을 라우팅할 수 있게 하는 두 VPC 간의 연결입니다. 자세한 내용은 Amazon VPC 설명서의 VPC 피어링이란?을 참조하십시오.

취약성

시스템 보안을 손상시키는 소프트웨어 또는 하드웨어 결함입니다.

W

웜 캐시

자주 액세스하는 최신 관련 데이터를 포함하는 버퍼 캐시입니다. 버퍼 캐시에서 데이터베이스 인스턴스를 읽을 수 있기 때문에 주 메모리나 디스크에서 읽는 것보다 빠릅니다.

웜 데이터

자주 액세스하지 않는 데이터입니다. 이런 종류의 데이터를 쿼리할 때는 일반적으로 적절히 느린 쿼리가 허용됩니다.

창 함수

현재 레코드와 어떤 식으로든 관련된 행 그룹에서 계산을 수행하는 SQL 함수입니다. 창 함수는 이동 평균을 계산하거나 현재 행의 상대적 위치를 기반으로 행 값에 액세스하는 등의 태스크를 처리하는 데 유용합니다.

워크로드

고객 대면 애플리케이션이나 백엔드 프로세스 같이 비즈니스 가치를 창출하는 리소스 및 코드 모음입니다.

워크스트림

마이그레이션 프로젝트에서 특정 작업 세트를 담당하는 직무 그룹입니다. 각 워크스트림은 독립적이지만 프로젝트의 다른 워크스트림을 지원합니다. 예를 들어, 포트폴리오 워크스트림은 애플리케이션 우선순위 지정, 웨이브 계획, 마이그레이션 메타데이터 수집을 담당합니다. 포트폴리오 워크스트림은 이러한 자산을 마이그레이션 워크스트림에 전달하고, 마이그레이션 워크스트림은 서버와 애플리케이션을 마이그레이션합니다.

WORM

Write Once, Read Many(WORM)를 참조하세요.

WQF

AWS Workload Qualification Framework를 참조하세요.

Write Once Read Many(WORM)

데이터를 한 번 쓰고 데이터가 삭제되거나 수정되지 않도록 하는 스토리지 모델입니다. 권한 있는 사용자는 필요한 만큼 여러 번 데이터를 읽을 수 있지만 데이터를 변경할 수는 없습니다. 이 데이터 스토리지 인프라는 변경 불가능한 항목으로 간주됩니다.

Z

제로데이 익스플로잇

제로데이 취약성을 악용하는 공격(일반적으로 맬웨어)입니다.

제로데이 취약성

프로덕션 시스템의 명백한 결함 또는 취약성입니다. 위협 행위자는 이러한 유형의 취약성을 사용하여 시스템을 공격할 수 있습니다. 개발자는 공격의 결과로 취약성을 인지하는 경우가 많습니다.

제로샷 프롬프팅

태스크를 수행하기 위해 LLM에 명령을 제공하지만 안내에 도움이 되는 예제(샷)는 제공하지 않습니다. LLM은 사전 훈련된 지식을 사용하여 태스크를 처리해야 합니다. 제로샷 프롬프팅의 효과는 태스크의 복잡성과 프롬프트의 품질에 따라 달라집니다. 퓨샷 프롬프팅도 참조하세요.

좀비 애플리케이션

평균 CPU 및 메모리 사용량이 5% 미만인 애플리케이션입니다. 마이그레이션 프로젝트에서는 이러한 애플리케이션을 사용 중지하는 것이 일반적입니다.

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.