Menggunakan Apache Iceberg di AWS

AWS Bimbingan Preskriptif

AWS Bimbingan Preskriptif: Menggunakan Apache Iceberg di AWS

Table of Contents

Menggunakan Apache Iceberg di AWS

Amazon Web Services (kontributor)

November 2025 (riwayat dokumen)

Apache Iceberg adalah format tabel open-source yang menyederhanakan manajemen tabel sekaligus meningkatkan kinerja. AWS Layanan analitik seperti Amazon EMR, Amazon Athena AWS Glue, dan Amazon Redshift menyertakan dukungan asli untuk Iceberg, sehingga Anda dapat dengan mudah membangun danau data transaksional di atas Amazon Simple Storage Service (Amazon S3) on. AWS

Selain itu, Amazon SageMaker generasi berikutnya dibangun di atas arsitektur danau terbuka yang menyatukan akses data di seluruh danau data, gudang AWS data, dan sumber pihak ketiga dan federasi. Lakehouse sepenuhnya kompatibel dengan Iceberg dan memberi Anda fleksibilitas untuk mengakses dan meminta data di tempat dengan menggunakan Iceberg REST API.

Panduan teknis ini memberikan panduan untuk memulai dengan Iceberg pada berbagai hal Layanan AWS, dan mencakup praktik terbaik dan rekomendasi untuk menjalankan Iceberg AWS pada skala besar sambil mengoptimalkan biaya dan kinerja.

Apakah Anda baru memulai dengan Iceberg atau Anda adalah pengguna berpengalaman yang ingin mengoptimalkan beban kerja Iceberg yang ada AWS, panduan ini menawarkan wawasan berharga untuk setiap tahap proyek Anda

Dalam panduan ini:

Danau data modern

Kasus penggunaan lanjutan di danau data modern

Evolusi penyimpanan data telah berkembang dari database ke gudang data dan data lake, di mana setiap teknologi menangani kebutuhan bisnis dan data yang unik. Database tradisional unggul dalam menangani data terstruktur dan beban kerja transaksional, tetapi mereka menghadapi tantangan kinerja karena volume data meningkat. Gudang data muncul untuk mengatasi masalah kinerja dan skalabilitas, tetapi seperti database, mereka mengandalkan format eksklusif dalam sistem yang terintegrasi secara vertikal.

Data lake menawarkan salah satu opsi terbaik untuk menyimpan data dalam hal biaya, skalabilitas, dan fleksibilitas. Anda dapat menggunakan data lake untuk menyimpan volume besar data terstruktur dan tidak terstruktur dengan biaya rendah, dan menggunakan data ini untuk berbagai jenis beban kerja analitik, mulai dari pelaporan intelijen bisnis hingga pemrosesan data besar, analitik real-time, pembelajaran mesin, dan kecerdasan buatan generatif (AI), untuk membantu memandu keputusan yang lebih baik.

Terlepas dari manfaat ini, data lake pada awalnya tidak dirancang dengan kemampuan seperti database. Data lake tidak memberikan dukungan untuk semantik pemrosesan atomisitas, konsistensi, isolasi, dan daya tahan (ACID), yang mungkin Anda perlukan untuk mengoptimalkan dan mengelola data Anda secara efektif dalam skala di ratusan atau ribuan pengguna dengan menggunakan banyak teknologi berbeda. Data lake tidak menyediakan dukungan asli untuk fungsionalitas berikut:

- Melakukan pembaruan dan penghapusan tingkat rekor yang efisien saat data berubah dalam bisnis Anda
- Mengelola kinerja kueri saat tabel tumbuh menjadi jutaan file dan ratusan ribu partisi
- Memastikan konsistensi data di beberapa penulis dan pembaca bersamaan
- Mencegah korupsi data saat operasi penulisan gagal di tengah operasi
- Skema tabel yang berkembang dari waktu ke waktu tanpa (sebagian) menulis ulang kumpulan data

Tantangan ini telah menjadi sangat lazim dalam kasus penggunaan seperti penanganan pengambilan data perubahan (CDC) atau kasus penggunaan yang berkaitan dengan privasi, penghapusan data, dan streaming konsumsi data, yang dapat menghasilkan tabel yang kurang optimal.

Data lake yang menggunakan tabel HIVE-format tradisional mendukung operasi penulisan hanya untuk seluruh file. Ini membuat pembaruan dan penghapusan sulit diterapkan, memakan waktu, dan mahal. Selain itu, kontrol konkurensi dan jaminan yang ditawarkan dalam sistem yang sesuai dengan ACID diperlukan untuk memastikan integritas dan konsistensi data.

Tantangan-tantangan ini membuat pengguna mengalami dilema: memilih antara platform yang sepenuhnya terintegrasi tetapi berpemilik, atau memilih danau data mandiri yang netral tetapi intensif sumber daya, yang membutuhkan pemeliharaan dan migrasi konstan untuk mewujudkan nilai potensinya.

Untuk membantu mengatasi tantangan ini, Iceberg menyediakan fungsionalitas seperti database tambahan yang menyederhanakan pengoptimalan dan pengelolaan overhead data lake, sambil tetap mendukung penyimpanan pada sistem hemat biaya seperti Amazon S3.

Pengantar Apache Iceberg

Apache Iceberg adalah format tabel open-source yang menyediakan fitur dalam tabel data lake yang secara historis hanya tersedia di database atau gudang data. Ini dirancang untuk skala dan kinerja, dan sangat cocok untuk mengelola tabel yang lebih dari ratusan gigabyte. Beberapa fitur utama dari tabel Iceberg adalah:

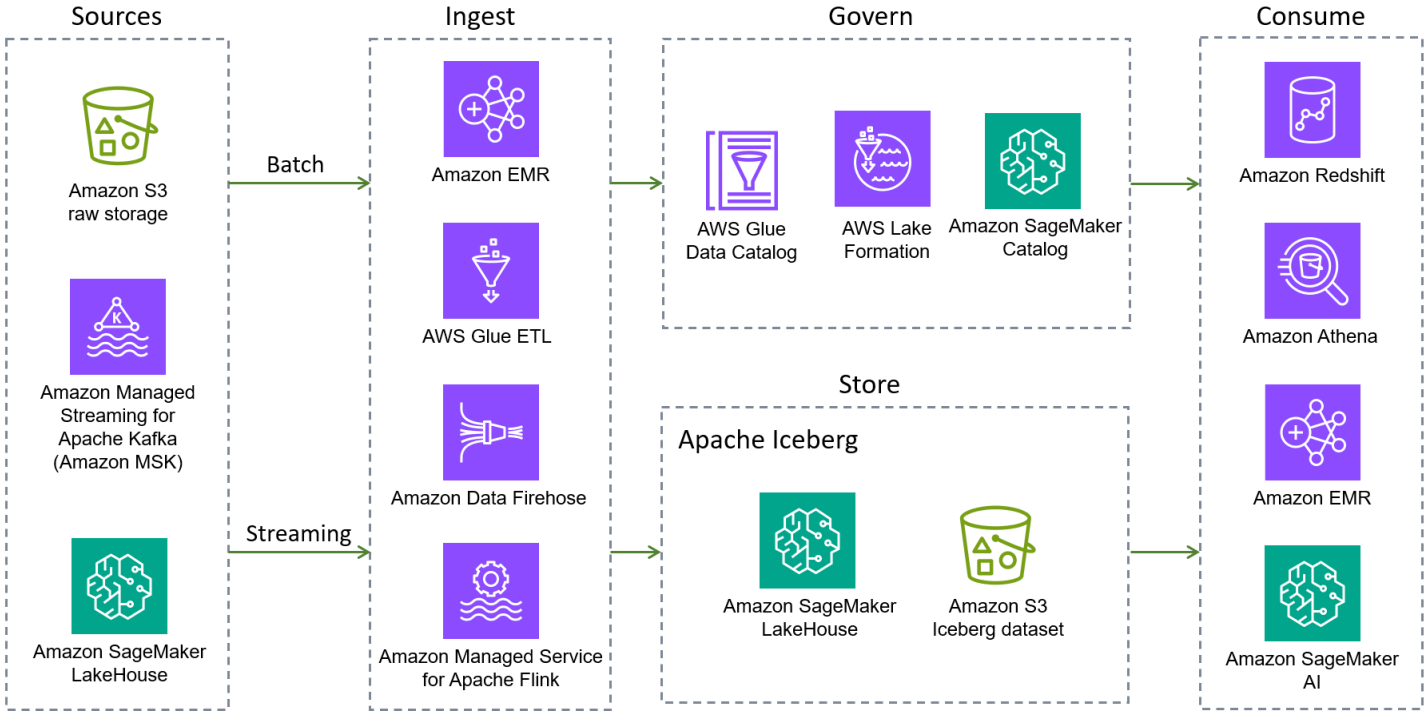
- Hapus, perbarui, dan gabungkan.Iceberg mendukung perintah SQL standar untuk pergudangan data untuk digunakan dengan tabel danau data.
- Perencanaan pemindaian cepat dan penyaringan lanjutan. Iceberg menyimpan metadata seperti partisi dan statistik tingkat kolom yang dapat digunakan oleh mesin untuk mempercepat perencanaan dan menjalankan kueri.
- Evolusi skema penuh. Iceberg mendukung penambahan, penurunan, pembaruan, atau penggantian nama kolom tanpa efek samping.
- Evolusi partisi. Anda dapat memperbarui tata letak partisi tabel saat volume data atau pola kueri berubah. Iceberg mendukung perubahan kolom tempat tabel dipartisi, atau menambahkan kolom ke, atau menghapus kolom dari, partisi komposit.
- Partisi tersembunyi.Fitur ini mencegah membaca partisi yang tidak perlu secara otomatis. Ini menghilangkan kebutuhan pengguna untuk memahami detail partisi tabel atau menambahkan filter tambahan ke kueri mereka.
- Versi rollback. Pengguna dapat dengan cepat memperbaiki masalah dengan kembali ke keadaan pra-transaksi.
- Perjalanan waktu. Pengguna dapat menanyakan versi tabel sebelumnya yang spesifik.

- Isolasi yang dapat diserialisasi. Perubahan tabel bersifat atomik, sehingga pembaca tidak pernah melihat perubahan sebagian atau tidak berkomitmen.
- Penulis bersamaan. Iceberg menggunakan konkurensi optimis untuk memungkinkan beberapa transaksi berhasil. Jika terjadi konflik, salah satu penulis harus mencoba kembali transaksi.
- Buka format file. Iceberg mendukung beberapa format file open source, termasuk Apache Parquet, ApacheAvro, dan Apache ORC.

Singkatnya, data lake yang menggunakan format Iceberg mendapat manfaat dari konsistensi transaksional, kecepatan, skala, dan evolusi skema. Untuk informasi lebih lanjut tentang ini dan fitur Iceberg lainnya, lihat dokumentasi Apache Iceberg.

AWS dukungan untuk Apache Iceberg

Apache Iceberg didukung oleh Layanan AWS Amazon EMR, Amazon Athena,Amazon Redshift, dan Amazon. AWS Glue SageMaker Diagram berikut menggambarkan arsitektur referensi yang disederhanakan dari danau data yang didasarkan pada Gunung Es.



Berikut ini Layanan AWS menyediakan integrasi Iceberg asli. Ada tambahan Layanan AWS yang dapat berinteraksi dengan Iceberg, baik secara tidak langsung atau dengan mengemas perpustakaan Iceberg.

- Amazon S3 adalah tempat terbaik untuk membangun data lake karena daya tahan, ketersediaan, skalabilitas, keamanan, kepatuhan, dan kemampuan auditnya. Iceberg dirancang dan dibangun untuk berinteraksi dengan Amazon S3 dengan mulus, dan memberikan dukungan untuk banyak fitur Amazon S3 seperti yang tercantum dalam dokumentasi Iceberg. Selain itu, Amazon S3 Tables menghadirkan penyimpanan objek cloud pertama dengan dukungan Iceberg bawaan dan merampingkan penyimpanan data tabular dalam skala besar. Dengan dukungan Tabel S3 untuk Iceberg, Anda dapat dengan mudah menanyakan data tabular Anda dengan menggunakan mesin kueri populer AWS dan pihak ketiga.
- Generasi SageMaker berikutnya dibangun di atas arsitektur danau terbuka yang menyatukan akses data di seluruh danau data Amazon S3, gudang data Amazon Redshift, dan sumber data pihak ketiga dan federasi. Kemampuan ini membantu Anda membangun analitik dan AI/ML aplikasi yang kuat pada satu salinan data. Lakehouse sepenuhnya kompatibel dengan Iceberg, sehingga Anda memiliki fleksibilitas untuk mengakses dan meminta data di tempat dengan menggunakan Iceberg REST API.
- Amazon EMR adalah solusi data besar untuk pemrosesan data skala petabyte, analitik interaktif, dan pembelajaran mesin dengan menggunakan kerangka kerja open source seperti Apache Spark, Flink, Trino, dan Hive. Amazon EMR dapat berjalan pada cluster Amazon Elastic Compute Cloud (Amazon EC2) yang disesuaikan, Amazon Elastic Kubernetes Service (Amazon EKS), atau Amazon EMR Tanpa Server. AWS Outposts
- Amazon Athena adalah layanan analitik interaktif tanpa server yang dibangun di atas kerangka kerja sumber terbuka. Ini mendukung format tabel terbuka dan file dan menyediakan cara yang disederhanakan dan fleksibel untuk menganalisis petabyte data di mana ia tinggal. Athena menyediakan dukungan asli untuk membaca, perjalanan waktu, menulis, dan kueri DDL untuk Iceberg dan menggunakan metastore for the Iceberg. AWS Glue Data Catalog
- Amazon Redshift adalah gudang data cloud skala petabyte yang mendukung opsi penerapan berbasis cluster dan tanpa server. Amazon Redshift Spectrum dapat menanyakan tabel eksternal yang terdaftar dengan AWS Glue Data Catalog dan disimpan di Amazon S3. Redshift Spectrum juga menyediakan dukungan untuk format penyimpanan Iceberg.
- AWS Glueadalah layanan integrasi data tanpa server yang memudahkan untuk menemukan, menyiapkan, memindahkan, dan mengintegrasikan data dari berbagai sumber untuk analitik, pembelajaran mesin (ML), dan pengembangan aplikasi. Ini sepenuhnya terintegrasi dengan Iceberg. Secara khusus, Anda dapat melakukan operasi baca dan tulis pada tabel Iceberg dengan menggunakan AWS Glue pekerjaan, mengelola tabel melalui AWS Glue Data Catalog(kompatibel dengan metastore-Hive), menemukan dan mendaftarkan tabel secara otomatis menggunakan AWS Glue crawler, dan mengevaluasi kualitas data dalam tabel Iceberg melalui fitur Kualitas Data.

AWS Glue Ini AWS Glue Data Catalog juga mendukung pengumpulan statistik kolom, menghitung dan memperbarui jumlah nilai yang berbeda (NDVs) untuk setiap kolom di tabel Iceberg, dan pengoptimalan tabel otomatis (pemadatan, retensi snapshot, penghapusan file yatim). AWS Glue juga mendukung integrasi nol-ETL dari daftar Layanan AWS dan aplikasi pihak ketiga ke dalam tabel Iceberg.

- Amazon Data Firehose adalah layanan yang dikelola sepenuhnya untuk mengirimkan data streaming waktu nyata ke tujuan seperti Amazon S3, Amazon Redshift, Layanan Amazon, Amazon Tanpa Server, Splunk, tabel Apache Iceberg, dan titik akhir HTTP atau HTTP kustom apa pun yang dimiliki oleh penyedia OpenSearch layanan pihak ketiga yang didukung, termasuk Datadog, Dynatrace,, MongoDB, New Relic, Coralogigie X, dan Elastis. OpenSearch LogicMonitor Dengan Firehose, Anda tidak perlu menulis aplikasi atau mengelola sumber daya. Anda mengonfigurasi produsen data Anda untuk mengirim data ke Firehose, dan secara otomatis mengirimkan data ke tujuan yang Anda tentukan. Anda juga dapat mengonfigurasi Firehose untuk mengubah data Anda sebelum mengirimkannya.
- Amazon Managed Service untuk Apache Flink adalah layanan Amazon yang dikelola sepenuhnya yang memungkinkan Anda menggunakan aplikasi Apache Flink untuk memproses data streaming. Ini mendukung membaca dari dan menulis ke tabel Iceberg, dan memungkinkan pemrosesan data dan analitik waktu nyata.
- Amazon SageMaker AI mendukung penyimpanan set fitur di Amazon SageMaker AI Feature Store dengan menggunakan format Iceberg.
- AWS Lake Formationmemberikan izin kontrol akses kasar dan halus untuk mengakses data, termasuk tabel Iceberg yang dikonsumsi oleh Athena atau Amazon Redshift. Untuk mempelajari lebih lanjut tentang dukungan izin untuk tabel Gunung Es, lihat dokumentasi Lake Formation.

AWS memiliki berbagai layanan yang mendukung Iceberg, tetapi mencakup semua layanan ini berada di luar cakupan panduan ini. Bagian berikut mencakup Spark (batch dan streaming terstruktur) di Amazon EMR AWS Glue dan, serta Athena SQL. Bagian berikut memberikan pandangan singkat pada dukungan Iceberg di Athena SQL.

Memulai dengan tabel Iceberg di Amazon Athena SQL

Amazon Athena menyediakan dukungan bawaan untuk Iceberg. Anda dapat menggunakan Iceberg tanpa langkah atau konfigurasi tambahan kecuali untuk menyiapkan prasyarat layanan yang dirinci di bagian Memulai dokumentasi Athena. Bagian ini memberikan pengantar singkat untuk membuat tabel di Athena. Untuk informasi lebih lanjut, lihat Bekerja dengan tabel Iceberg dengan menggunakan Athena SQL nanti dalam panduan ini.

Anda dapat membuat tabel Iceberg AWS dengan menggunakan mesin yang berbeda. Tabel-tabel itu bekerja dengan mulus. Layanan AWS Untuk membuat tabel Iceberg pertama Anda dengan Athena SQL, Anda dapat menggunakan kode boilerplate berikut.

```
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'
)
```

Bagian berikut memberikan contoh pembuatan tabel Gunung Es yang dipartisi dan tidak dipartisi di Athena. Untuk informasi selengkapnya, lihat sintaks Iceberg yang dirinci dalam dokumentasi Athena.

Membuat tabel yang tidak dipartisi

Contoh pernyataan berikut menyesuaikan kode SQL boilerplate untuk membuat tabel Iceberg yang tidak dipartisi di Athena. Anda dapat menambahkan pernyataan ini ke editor kueri di konsol Athena untuk membuat tabel.

```
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'
)
```

Untuk step-by-step petunjuk penggunaan editor kueri, lihat Memulai dokumentasi Athena.

Membuat tabel yang dipartisi

Pernyataan berikut membuat tabel dipartisi berdasarkan tanggal dengan menggunakan konsep Iceberg tentang partisi tersembunyi. Ini menggunakan `day()` transformasi untuk mendapatkan partisi harian, menggunakan `dd-mm-yyyy` format, dari kolom stempel waktu. Iceberg tidak menyimpan nilai ini sebagai kolom baru dalam kumpulan data. Sebagai gantinya, nilainya diturunkan dengan cepat saat Anda menulis atau menanyakan data.

```
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'
)
```

Membuat tabel dan memuat data dengan pernyataan CTAS tunggal

Dalam contoh yang dipartisi dan tidak dipartisi di bagian sebelumnya, tabel Iceberg dibuat sebagai tabel kosong. Anda dapat memuat data ke tabel dengan menggunakan `MERGE` pernyataan `INSERT` atau. Atau, Anda dapat menggunakan `CREATE TABLE AS SELECT (CTAS)` pernyataan untuk membuat dan memuat data ke dalam tabel Iceberg dalam satu langkah.

CTAS adalah cara terbaik di Athena untuk membuat tabel dan memuat data dalam satu pernyataan. Contoh berikut menggambarkan bagaimana menggunakan CTAS untuk membuat tabel Iceberg (`iceberg_ctas_table`) dari tabel yang ada () di Hive/Parquet Athena. `hive_table`

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

Untuk mempelajari lebih lanjut tentang CTAS, lihat dokumentasi Athena CTAS.

Memasukkan, memperbarui, dan menghapus data

Athena mendukung berbagai cara penulisan data ke tabel Iceberg dengan menggunakan pernyataan`INSERT INTO`,, `UPDATE``MERGE INTO`, dan M. `DELETE FRO`

> **Note**
>
> Athena SQL saat ini tidak mendukung pendekatan tersebut. copy-on-write `UPDATE`,`MERGE INTO`, dan `DELETE FROM` operasi selalu menggunakan merge-on-read pendekatan dengan penghapusan posisi, terlepas dari properti tabel yang ditentukan. Jika Anda mengatur properti tabel seperti`write.update.mode`,, dan `write.delete.mode` untuk digunakan `write.merge.mode` copy-on-write, kueri Anda tidak akan gagal, tetapi Athena akan mengabaikannya dan tetap menggunakannya. merge-on-read

Pernyataan berikut digunakan `INSERT INTO` untuk menambahkan data ke tabel Iceberg:

```
INSERT INTO "iceberg_db"."ice_table" VALUES (
    'red', '222022-07-19T03:47:29', 'PersonNew', 178, 'Tuna', now()
)

SELECT * FROM "iceberg_db"."ice_table"
where color = 'red' limit 10;
```

Contoh output:

Untuk informasi lebih lanjut, lihat dokumentasi Athena.

Menanyakan tabel Iceberg

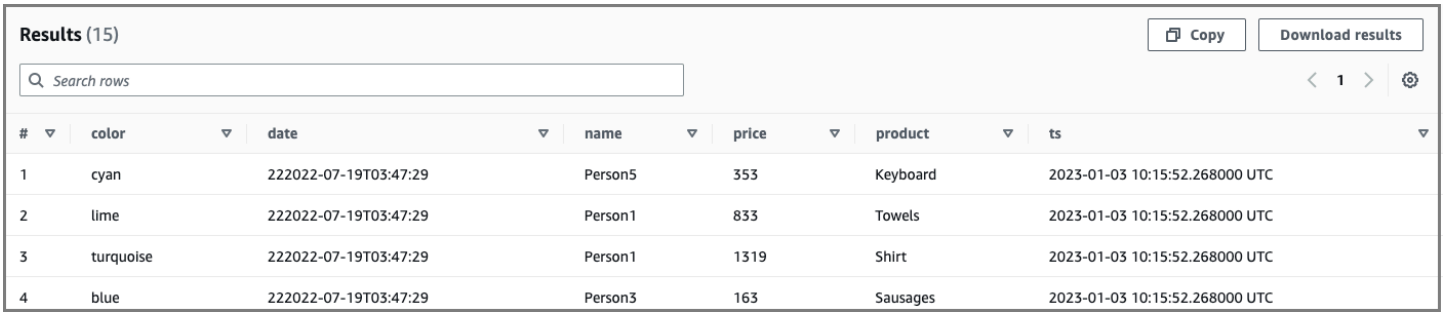
Anda dapat menjalankan query SQL reguler terhadap tabel Iceberg Anda dengan menggunakan Athena SQL, seperti yang diilustrasikan dalam contoh sebelumnya.

Selain pertanyaan biasa, Athena juga mendukung kueri perjalanan waktu untuk tabel Iceberg. Seperti yang telah dibahas sebelumnya, Anda dapat mengubah catatan yang ada melalui pembaruan atau penghapusan di tabel Iceberg, sehingga lebih mudah menggunakan kueri perjalanan waktu untuk melihat kembali ke versi tabel yang lebih lama berdasarkan stempel waktu atau ID snapshot.

Misalnya, pernyataan berikut memperbarui nilai warna untuk`Person5`, dan kemudian menampilkan nilai sebelumnya dari 4 Januari 2023:

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

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

Contoh output:



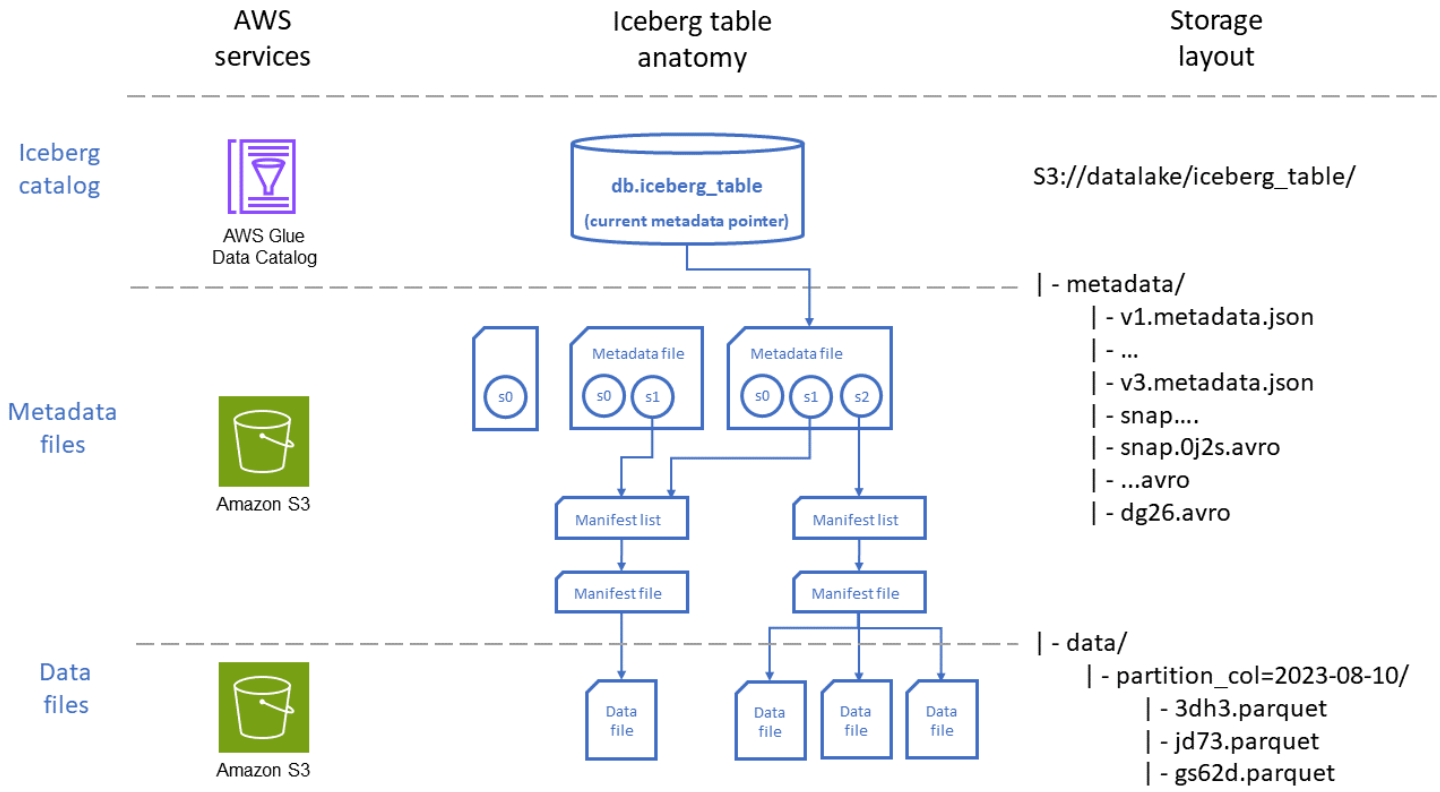
Untuk sintaks dan contoh tambahan kueri perjalanan waktu, lihat dokumentasi Athena.

Anatomi tabel gunung es

Sekarang setelah kita membahas langkah-langkah dasar bekerja dengan tabel Iceberg, mari selami lebih dalam detail dan desain meja Iceberg yang rumit.

Untuk mengaktifkan fitur yang dijelaskan sebelumnya dalam panduan ini, Iceberg dirancang dengan lapisan hierarkis data dan file metadata. Lapisan ini mengelola metadata secara cerdas untuk mengoptimalkan perencanaan dan eksekusi kueri.

Diagram berikut menggambarkan organisasi tabel Gunung Es melalui dua perspektif: yang Layanan AWS digunakan untuk menyimpan tabel dan penempatan file di Amazon S3.



Seperti yang ditunjukkan pada diagram, tabel Gunung Es terdiri dari tiga lapisan utama:

- Katalog Iceberg: AWS Glue Data Catalog terintegrasi secara native dengan Iceberg dan, untuk sebagian besar kasus penggunaan, merupakan opsi terbaik untuk beban kerja yang berjalan. AWS Layanan yang berinteraksi dengan tabel Iceberg (misalnya, Athena) menggunakan katalog untuk menemukan versi snapshot tabel saat ini, baik untuk membaca atau menulis data.
- Lapisan metadata: File metadata, yaitu file manifes dan file daftar manifes, melacak informasi seperti skema tabel, strategi partisi, dan lokasi file data, serta statistik tingkat kolom seperti

rentang minimum dan maksimum untuk catatan yang disimpan di setiap file data. File metadata ini disimpan di Amazon S3 dalam jalur tabel.

  - File manifes berisi catatan untuk setiap file data, termasuk lokasi, format, ukuran, checksum, dan informasi relevan lainnya.
  - Daftar manifes menyediakan indeks file manifes. Seiring bertambahnya jumlah file manifes dalam tabel, memecah informasi tersebut menjadi subbagian yang lebih kecil membantu mengurangi jumlah file manifes yang perlu dipindai oleh kueri.
  - File metadata berisi informasi tentang seluruh tabel Iceberg, termasuk daftar manifes, skema, metadata partisi, file snapshot, dan file lain yang digunakan untuk mengelola metadata tabel.
- Lapisan data: Lapisan ini berisi file yang memiliki catatan data yang akan dijalankan oleh kueri. File-file ini dapat disimpan dalam berbagai format, termasuk Apache Parquet, ApacheAvro, dan Apache ORC.
  - File data berisi catatan data untuk tabel.
  - Hapus file yang menyandikan penghapusan tingkat baris dan perbarui operasi dalam tabel Iceberg. Iceberg memiliki dua jenis file hapus, seperti yang dijelaskan dalam dokumentasi Iceberg. File-file ini dibuat oleh operasi dengan menggunakan merge-on-read mode.

Bekerja dengan Iceberg di Amazon EMR

Amazon EMR menyediakan pemrosesan data skala petabyte, analitik interaktif, dan pembelajaran mesin di cloud dengan menggunakan kerangka kerja open source seperti Apache Spark, Apache Hive, Flink, dan Trino.

> **Note**
>
> Panduan ini menggunakan Apache Spark sebagai contoh.

Amazon EMR mendukung beberapa opsi penerapan: Amazon EMR aktif, Amazon EMR di EKS, EC2 Amazon EMR Tanpa Server, dan Amazon EMR aktif. AWS Outposts Untuk memilih opsi penerapan beban kerja Anda, lihat FAQ EMR Amazon.

Versi dan kompatibilitas fitur

Amazon EMR versi 6.5.0 dan versi yang lebih baru mendukung Apache Iceberg secara asli. Untuk daftar versi Iceberg yang didukung untuk setiap rilis EMR Amazon, lihat Riwayat rilis Iceberg dalam dokumentasi Amazon EMR. Tinjau juga bagian di bawah Gunakan cluster dengan Iceberg untuk melihat fitur Iceberg mana yang didukung di Amazon EMR pada kerangka kerja yang berbeda.

Kami menyarankan Anda menggunakan versi EMR Amazon terbaru untuk mendapatkan keuntungan dari versi Iceberg terbaru yang didukung. Contoh kode dan konfigurasi di bagian ini mengasumsikan bahwa Anda menggunakan Amazon EMR rilis emr-7.8.0.

Membuat cluster EMR Amazon dengan Iceberg

Untuk membuat cluster EMR Amazon di Amazon EC2 dengan Iceberg diinstal, ikuti petunjuk dalam dokumentasi Amazon EMR.

Secara khusus, cluster Anda harus dikonfigurasi dengan klasifikasi berikut:

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

```
}]
```

Anda juga dapat memilih untuk menggunakan Amazon EMR Tanpa Server atau Amazon EMR di EKS sebagai opsi penerapan untuk beban kerja Iceberg Anda, mulai dari Amazon EMR 6.6.0.

Mengembangkan aplikasi Iceberg di Amazon EMR

Untuk mengembangkan kode Spark untuk aplikasi Iceberg Anda, Anda dapat menggunakan Amazon EMR Studio, yang merupakan lingkungan pengembangan terintegrasi berbasis web (IDE) untuk notebook Jupyter yang dikelola sepenuhnya yang berjalan di klaster EMR Amazon.

Menggunakan notebook Amazon EMR Studio

Anda dapat mengembangkan aplikasi Spark secara interaktif di notebook Amazon EMR Studio Workspace dan menghubungkan notebook tersebut ke EMR Amazon Anda di cluster EC2 atau Amazon EMR di titik akhir yang dikelola EKS. Lihat Layanan AWS dokumentasi untuk petunjuk cara menyiapkan EMR Studio untuk Amazon EMR dan Amazon EMR di EC2 EKS.

Untuk menggunakan Iceberg di EMR Studio, ikuti langkah-langkah berikut:

1. Luncurkan klaster EMR Amazon dengan Iceberg diaktifkan, seperti yang diinstruksikan dalam Gunakan klaster dengan Iceberg Installed.
2. Siapkan Studio EMR. Untuk petunjuk, lihat Menyiapkan Amazon EMR Studio.
3. Buka buku catatan EMR Studio Workspace dan jalankan kode berikut sebagai sel pertama di buku catatan untuk mengonfigurasi sesi Spark Anda untuk menggunakan Iceberg:

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

di mana:

- `<catalog_name>`adalah nama katalog sesi Iceberg Spark Anda. Ganti dengan nama pilihan Anda, dan ingat untuk mengubah referensi di semua konfigurasi yang terkait dengan katalog ini. Dalam kode Anda, Anda dapat merujuk ke tabel Iceberg Anda dengan menggunakan nama tabel yang sepenuhnya memenuhi syarat, termasuk nama katalog sesi Spark, sebagai berikut:

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

  Atau, Anda dapat mengubah katalog default ke katalog Iceberg yang Anda tentukan dengan menyetel `spark.sql.defaultCatalog` ke nama katalog Anda. Pendekatan kedua ini memungkinkan Anda untuk merujuk ke tabel tanpa awalan katalog, yang dapat menyederhanakan kueri Anda.
- `<catalog_name>.warehouse`menunjuk ke jalur Amazon S3 tempat Anda ingin menyimpan data dan metadata Anda.
- Untuk membuat katalog AWS Glue Data Catalog, atur `spark.sql.catalog.<catalog_name>.type` ke`glue`. Kunci ini diperlukan untuk menunjuk ke kelas implementasi untuk setiap implementasi katalog kustom. Bagian praktik terbaik umum nanti dalam panduan ini menjelaskan berbagai katalog yang didukung Iceberg.

4. Anda sekarang dapat mulai secara interaktif mengembangkan aplikasi Spark Anda untuk Iceberg di notebook, seperti yang Anda lakukan untuk aplikasi Spark lainnya.

Untuk informasi selengkapnya tentang mengonfigurasi Spark untuk Apache Iceberg dengan menggunakan Amazon EMR Studio, lihat posting blog Membangun data lake berperforma tinggi, sesuai ACID, dan berkembang menggunakan Apache Iceberg di Amazon EMR.

Pekerjaan Menjalankan Iceberg di Amazon EMR

Setelah Anda mengembangkan kode aplikasi Spark untuk beban kerja Iceberg Anda, Anda dapat menjalankannya di opsi penyebaran EMR Amazon apa pun yang mendukung Iceberg (lihat FAQ Amazon EMR).

Seperti pekerjaan Spark lainnya, Anda dapat mengirimkan pekerjaan ke EMR Amazon EC2 di klaster dengan menambahkan langkah-langkah atau dengan mengirimkan pekerjaan Spark secara interaktif ke node master. Untuk menjalankan pekerjaan Spark, lihat halaman dokumentasi Amazon EMR berikut:

- Untuk gambaran umum tentang berbagai opsi untuk mengirimkan karya ke EMR Amazon di EC2 klaster dan petunjuk terperinci untuk setiap opsi, lihat Mengirimkan pekerjaan ke klaster.

- Untuk Amazon EMR di EKS, lihat Menjalankan pekerjaan Spark dengan. StartJobRun
- Untuk EMR Tanpa Server, lihat Menjalankan pekerjaan.

Bagian berikut memberikan contoh untuk setiap opsi penyebaran EMR Amazon.

Amazon EMR aktif EC2

Anda dapat menggunakan langkah-langkah ini untuk mengirimkan pekerjaan Iceberg Spark:

1. Buat file `emr_step_iceberg.json` dengan konten berikut di workstation Anda:

```
[{
    "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. Ubah file konfigurasi untuk pekerjaan Spark spesifik Anda dengan menyesuaikan opsi konfigurasi Iceberg yang disorot dengan huruf tebal.
3. Kirim langkah dengan menggunakan AWS Command Line Interface (AWS CLI). Jalankan perintah di direktori tempat `emr_step_iceberg.json` file berada.

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

Amazon EMR Tanpa Server

Untuk mengirimkan pekerjaan Iceberg Spark ke EMR Tanpa Server dengan menggunakan: AWS CLI

1. Buat file `emr_serverless_iceberg.json` dengan konten berikut di workstation Anda:

```
{
    "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. Ubah file konfigurasi untuk pekerjaan Spark spesifik Anda dengan menyesuaikan opsi konfigurasi Iceberg yang disorot dengan huruf tebal.

3. Kirimkan pekerjaan dengan menggunakan AWS CLI. Jalankan perintah di direktori tempat `emr_serverless_iceberg.json` file berada:

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

Untuk mengirimkan pekerjaan Iceberg Spark ke EMR Tanpa Server dengan menggunakan konsol EMR Studio:

1. Ikuti petunjuk dalam dokumentasi EMR Tanpa Server.
2. Untuk konfigurasi Job, gunakan konfigurasi Iceberg untuk Spark yang disediakan untuk AWS CLI dan sesuaikan bidang yang disorot untuk Iceberg. Untuk petunjuk terperinci, lihat Menggunakan Apache Iceberg dengan EMR Tanpa Server di dokumentasi Amazon EMR.

Amazon EMR di EKS

Untuk mengirimkan pekerjaan Iceberg Spark ke Amazon EMR di EKS dengan menggunakan: AWS CLI

1. Buat file `emr_eks_iceberg.json` dengan konten berikut di workstation Anda:

```
{
    "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. Ubah file konfigurasi untuk pekerjaan Spark Anda dengan menyesuaikan opsi konfigurasi Iceberg yang disorot dengan huruf tebal.
3. Kirimkan pekerjaan dengan menggunakan AWS CLI. Jalankan perintah berikut di direktori tempat `emr_eks_iceberg.json` file berada:

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

Untuk petunjuk terperinci, lihat Menggunakan Apache Iceberg dengan Amazon EMR di EKS di Amazon EMR pada dokumentasi EKS.

Praktik terbaik untuk Amazon EMR

Bagian ini memberikan pedoman umum untuk menyetel pekerjaan Spark di Amazon EMR untuk mengoptimalkan membaca dan menulis data ke tabel Iceberg. Untuk praktik terbaik khusus Iceberg, lihat bagian Praktik terbaik nanti di panduan ini.

- Gunakan versi terbaru Amazon EMR - Amazon EMR menyediakan pengoptimalan Spark di luar kotak dengan runtime Amazon EMR Spark. AWS meningkatkan kinerja mesin runtime Spark dengan setiap rilis baru.
- Tentukan infrastruktur optimal untuk beban kerja Spark Anda — Beban kerja Spark mungkin memerlukan berbagai jenis perangkat keras untuk karakteristik pekerjaan yang berbeda guna

memastikan kinerja yang optimal. Amazon EMR mendukung beberapa jenis instans (seperti komputasi yang dioptimalkan, dioptimalkan memori, tujuan umum, dan penyimpanan yang dioptimalkan) untuk mencakup semua jenis persyaratan pemrosesan. Saat Anda melakukan onboard beban kerja baru, sebaiknya Anda melakukan benchmark dengan tipe instans umum seperti M5 atau M6g. Pantau metrik sistem operasi (OS) dan YARN dari Ganglia dan Amazon CloudWatch untuk menentukan kemacetan sistem (CPU, memori, penyimpanan, dan I/O) pada beban puncak dan pilih perangkat keras yang sesuai.

- Tune `spark.sql.shuffle.partitions` - Atur `spark.sql.shuffle.partitions` properti ke jumlah total inti virtual (vCores) di cluster Anda atau ke kelipatan dari nilai itu (biasanya, 1 hingga 2 kali jumlah total vCores). Pengaturan ini memengaruhi paralelisme Spark saat Anda menggunakan partisi hash dan rentang sebagai mode distribusi tulis. Ini meminta shuffle sebelum menulis untuk mengatur data, yang memastikan keselarasan partisi.
- Aktifkan penskalaan terkelola — Untuk hampir semua kasus penggunaan, sebaiknya aktifkan penskalaan terkelola dan alokasi dinamis. Namun, jika Anda memiliki beban kerja yang memiliki pola yang dapat diprediksi, kami sarankan Anda menonaktifkan penskalaan otomatis dan alokasi dinamis. Saat penskalaan terkelola diaktifkan, sebaiknya gunakan Instans Spot untuk mengurangi biaya. Gunakan Instans Spot untuk node tugas alih-alih node inti atau master. Saat Anda menggunakan Instans Spot, gunakan armada instans dengan beberapa jenis instans per armada untuk memastikan ketersediaan spot.
- Gunakan broadcast join bila memungkinkan — Broadcast (mapside) join adalah gabungan yang paling optimal, selama salah satu tabel Anda cukup kecil untuk muat dalam memori node terkecil Anda (dalam urutan MBs) dan Anda melakukan equi (=) join. Semua jenis gabungan kecuali gabungan luar penuh didukung. Sebuah broadcast join menyiarkan tabel yang lebih kecil sebagai tabel hash di semua node pekerja dalam memori. Setelah tabel kecil disiarkan, Anda tidak dapat mengubahnya. Karena tabel hash secara lokal di mesin virtual Java (JVM), maka dapat digabungkan dengan mudah dengan tabel besar berdasarkan kondisi gabungan dengan menggunakan gabungan hash. Broadcast join memberikan kinerja tinggi karena overhead shuffle minimal.
- Setel pengumpul sampah — Jika siklus pengumpulan sampah (GC) lambat, pertimbangkan untuk beralih dari pengumpul sampah paralel default ke G1GC untuk kinerja yang lebih baik. Untuk mengoptimalkan kinerja GC, Anda dapat menyempurnakan parameter GC. Untuk melacak kinerja GC, Anda dapat memantaunya dengan menggunakan UI Spark. Idealnya, waktu GC harus kurang dari atau sama dengan 1 persen dari total runtime tugas.

Bekerja dengan Iceberg di AWS Glue

AWS Glueadalah layanan integrasi data tanpa server yang memudahkan untuk menemukan, menyiapkan, memindahkan, dan mengintegrasikan data dari berbagai sumber untuk analitik, pembelajaran mesin (ML), dan pengembangan aplikasi. Salah satu kemampuan inti AWS Glue adalah kemampuannya untuk melakukan operasi ekstrak, transformasi, dan beban (ETL) dengan cara yang sederhana dan hemat biaya. Ini membantu mengkategorikan data Anda, membersihkannya, memperkayanya, dan memindahkannya dengan andal antara berbagai penyimpanan data dan aliran data.

AWS Glue pekerjaan merangkum skrip yang mendefinisikan logika transformasi dengan menggunakan runtime Apache Spark atau Python. AWS Glue pekerjaan dapat dijalankan dalam mode batch dan streaming.

Saat Anda membuat pekerjaan Iceberg di AWS Glue, tergantung pada versinya AWS Glue, Anda dapat menggunakan integrasi Iceberg asli atau versi Iceberg khusus untuk melampirkan dependensi Iceberg ke pekerjaan tersebut.

Menggunakan integrasi Iceberg asli

AWS Glue Versi 3.0, 4.0, dan 5.0 secara native mendukung format data lake transaksional seperti Apache Iceberg, Apache Hudi, dan Linux Foundation Delta Lake untuk Spark. AWS Glue Fitur integrasi ini menyederhanakan langkah-langkah konfigurasi yang diperlukan untuk mulai menggunakan kerangka kerja ini di. AWS Glue

Untuk mengaktifkan dukungan Iceberg untuk AWS Glue pekerjaan Anda, atur pekerjaan: Pilih tab Detail pekerjaan untuk AWS Glue pekerjaan Anda, gulir ke parameter Job di bawah Properti lanjutan, dan atur kunci ke `--datalake-formats` dan nilainya. `iceberg`

Jika Anda menulis pekerjaan dengan menggunakan buku catatan, Anda dapat mengonfigurasi parameter di sel notebook pertama dengan menggunakan `%%configure` sihir sebagai berikut:

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

`iceberg`Konfigurasi untuk `--datalake-formats` in AWS Glue sesuai dengan versi Iceberg tertentu berdasarkan versi Anda AWS Glue :

AWS Glue versi	Versi Iceberg default
5.0	1.7.1
4.0	1.0.0
3.0	0.13.1

Menggunakan versi Iceberg kustom

Dalam beberapa situasi, Anda mungkin ingin mempertahankan kendali atas versi Iceberg untuk pekerjaan itu dan meningkatkannya dengan kecepatan Anda sendiri. Misalnya, memutakhirkan ke versi yang lebih baru dapat membuka akses ke fitur baru dan peningkatan kinerja. Untuk menggunakan versi Iceberg tertentu AWS Glue, Anda dapat menyediakan file JAR Anda sendiri.

Sebelum Anda menerapkan versi Iceberg kustom, verifikasi kompatibilitas dengan AWS Glue lingkungan Anda dengan memeriksa bagian AWS Glue versi dokumentasi. AWS Glue Misalnya, AWS Glue 5.0 membutuhkan kompatibilitas dengan Spark 3.5.4.

Sebagai contoh, untuk menjalankan AWS Glue pekerjaan yang menggunakan Iceberg versi 1.9.1, ikuti langkah-langkah berikut:

1. Dapatkan dan unggah file JAR yang diperlukan ke Amazon S3:
   a. Unduh iceberg-spark-runtime-3.5_2.12-1.9.1.jar dan -1.9.1.jar dari repositori Apache Maven. iceberg-aws-bundle
   b. Unggah file-file ini ke lokasi bucket S3 yang Anda tentukan (misalnya,`s3://your-bucket-name/jars/`).
2. Siapkan parameter pekerjaan untuk AWS Glue pekerjaan Anda sebagai berikut:
   a. Tentukan jalur S3 lengkap ke kedua file JAR dalam `--extra-jars` parameter, pisahkan dengan koma (misalnya,). `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. Jangan sertakan `iceberg` sebagai nilai untuk `--datalake-formats` parameter.

c. Jika Anda menggunakan AWS Glue 5.0, Anda harus mengatur `--user-jars-first` parameter ke`true`.

Konfigurasi percikan untuk Iceberg di AWS Glue

Bagian ini membahas konfigurasi Spark yang diperlukan untuk membuat pekerjaan AWS Glue ETL untuk kumpulan data Iceberg. Anda dapat mengatur konfigurasi ini dengan menggunakan tombol `--conf` Spark dengan daftar dipisahkan koma dari semua kunci dan nilai konfigurasi Spark. Anda dapat menggunakan `%%configure` sihir di buku catatan, atau bagian Parameter Job di AWS Glue Studio konsol.

```
%glue_version 5.0

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

Konfigurasikan sesi Spark dengan properti berikut:

- `<catalog_name>`adalah nama nama katalog sesi Iceberg Spark Anda. Ganti dengan nama pilihan Anda, dan ingat untuk mengubah referensi di semua konfigurasi yang terkait dengan katalog ini. Dalam kode Anda, Anda dapat merujuk ke tabel Iceberg Anda dengan menggunakan nama tabel yang sepenuhnya memenuhi syarat, termasuk nama katalog sesi Spark, sebagai berikut:

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

  Atau, Anda dapat mengubah katalog default ke katalog Iceberg yang Anda tentukan dengan menyetel `spark.sql.defaultCatalog` ke nama katalog Anda. Anda dapat menggunakan pendekatan kedua ini untuk merujuk ke tabel tanpa awalan katalog, yang dapat menyederhanakan kueri Anda.
- `<catalog_name>.<warehouse>`menunjuk ke jalur Amazon S3 tempat Anda ingin menyimpan data dan metadata Anda.
- Untuk membuat katalog AWS Glue Data Catalog, atur `spark.sql.catalog.<catalog_name>.type` ke`glue`. Kunci ini diperlukan untuk menunjuk ke

kelas implementasi untuk setiap implementasi katalog kustom. Untuk katalog yang didukung oleh Iceberg, lihat bagian Praktik terbaik umum nanti dalam panduan ini.

Misalnya, jika Anda memiliki katalog yang dipanggil`glue_iceberg`, Anda dapat mengonfigurasi pekerjaan Anda dengan menggunakan beberapa `--conf` kunci sebagai berikut:

```
%%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"
}
```

Atau, Anda dapat menggunakan kode untuk menambahkan konfigurasi di atas ke skrip Spark Anda sebagai berikut:

```
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()
```

Praktik terbaik untuk AWS Glue pekerjaan

Bagian ini memberikan pedoman umum untuk menyetel pekerjaan Spark AWS Glue untuk mengoptimalkan membaca dan menulis data ke tabel Iceberg. Untuk praktik terbaik khusus Iceberg, lihat bagian Praktik terbaik nanti di panduan ini.

- Gunakan versi terbaru AWS Glue dan tingkatkan bila memungkinkan — Versi baru AWS Glue memberikan peningkatan kinerja, mengurangi waktu startup, dan fitur baru. Mereka juga mendukung versi Spark yang lebih baru yang mungkin diperlukan untuk versi Iceberg terbaru.

Untuk daftar versi yang tersedia dan AWS Glue versi Spark yang mereka dukung, lihat AWS Glue dokumentasi.

- Optimalkan memori AWS Glue pekerjaan - Ikuti rekomendasi di posting AWS blog Optimalkan manajemen memori di AWS Glue.
- Gunakan AWS Glue Auto Scaling — Saat Anda mengaktifkan Auto Scaling AWS Glue , secara otomatis menyesuaikan jumlah pekerja secara dinamis berdasarkan beban AWS Glue kerja Anda. Ini membantu mengurangi biaya AWS Glue pekerjaan Anda selama beban puncak, AWS Glue karena menurunkan jumlah pekerja ketika beban kerja kecil dan pekerja duduk diam. Untuk menggunakan AWS Glue Auto Scaling, Anda menentukan jumlah maksimum pekerja yang dapat diskalakan oleh AWS Glue pekerjaan Anda. Untuk informasi selengkapnya, lihat Menggunakan penskalaan otomatis untukAWS Glue AWS Glue dokumentasi.
- Gunakan versi Iceberg yang diinginkan - integrasi AWS Glue asli untuk Iceberg adalah yang terbaik untuk memulai dengan Iceberg. Namun, untuk beban kerja produksi, kami menyarankan Anda menambahkan dependensi pustaka (seperti yang dibahas sebelumnya dalam panduan ini) untuk mendapatkan kontrol penuh atas versi Iceberg. Pendekatan ini membantu Anda mendapatkan keuntungan dari fitur Iceberg terbaru dan peningkatan kinerja dalam pekerjaan Anda AWS Glue .
- Aktifkan UI Spark untuk pemantauan dan debugging — Anda juga dapat menggunakan UI Spark AWS Glue untuk memeriksa pekerjaan Iceberg Anda dengan memvisualisasikan berbagai tahapan pekerjaan Spark dalam grafik asiklik terarah (DAG) dan memantau pekerjaan secara detail. Spark UI menyediakan cara yang efektif untuk memecahkan masalah dan mengoptimalkan pekerjaan Iceberg. Misalnya, Anda dapat mengidentifikasi tahapan bottleneck yang memiliki pengocokan besar atau tumpahan disk untuk mengidentifikasi peluang penyetelan. Untuk informasi selengkapnya, lihat Memantau pekerjaan menggunakan UI web Apache Spark dalam dokumentasi. AWS Glue

Bekerja dengan tabel Iceberg dengan menggunakan Apache Spark

Bagian ini memberikan gambaran umum tentang penggunaan Apache Spark untuk berinteraksi dengan tabel Iceberg. Contohnya adalah kode boilerplate yang dapat berjalan di Amazon EMR atau. AWS Glue

Catatan: Antarmuka utama untuk berinteraksi dengan tabel Iceberg adalah SQL, sehingga sebagian besar contoh akan menggabungkan Spark SQL dengan API. DataFrames

Membuat dan menulis tabel Iceberg

Anda dapat menggunakan Spark SQL dan Spark DataFrames untuk membuat dan menambahkan data ke tabel Iceberg.

Menggunakan Spark SQL

Untuk menulis dataset Iceberg, gunakan pernyataan SQL Spark standar seperti dan. `CREATE TABLE` `INSERT INTO`

Tabel yang tidak dipartisi

Berikut adalah contoh membuat tabel Iceberg yang tidak dipartisi dengan Spark SQL:

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

Untuk menyisipkan data ke dalam tabel yang tidak dipartisi, gunakan pernyataan standar: `INSERT INTO`

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

Tabel yang dipartisi

Berikut adalah contoh membuat tabel Iceberg yang dipartisi dengan Spark SQL:

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

Untuk menyisipkan data ke dalam tabel Iceberg yang dipartisi dengan Spark SQL, gunakan pernyataan standar: `INSERT INTO`

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

> **Note**
>
> Dimulai dengan Iceberg 1.5.0, mode distribusi `hash` tulis adalah default saat Anda memasukkan data ke dalam tabel yang dipartisi. Untuk informasi selengkapnya, lihat Menulis Mode Distribusi dalam dokumentasi Gunung Es.

Menggunakan DataFrames API

Untuk menulis kumpulan data Iceberg, Anda dapat menggunakan API. `DataFrameWriterV2`

Untuk membuat tabel Iceberg dan menulis data untuk itu, gunakan fungsi `df.writeTo(` t). Jika tabel ada, gunakan `.append()` fungsi. Jika tidak, gunakan Contoh `.create().` berikut digunakan`.createOrReplace()`, yang merupakan variasi `.create()` yang setara dengan`CREATE OR REPLACE TABLE AS SELECT`.

Tabel yang tidak dipartisi

Untuk membuat dan mengisi tabel Iceberg yang tidak dipartisi dengan menggunakan API: `DataFrameWriterV2`

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

Untuk menyisipkan data ke dalam tabel Iceberg tak terpartisi yang sudah ada dengan menggunakan API: `DataFrameWriterV2`

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

Tabel yang dipartisi

Untuk membuat dan mengisi tabel Iceberg yang dipartisi dengan menggunakan API: `DataFrameWriterV2`

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

Untuk menyisipkan data ke dalam tabel Iceberg yang dipartisi dengan menggunakan API: `DataFrameWriterV2`

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

```
    .append()
```

Memperbarui data dalam tabel Iceberg

Contoh berikut menunjukkan cara memperbarui data dalam tabel Iceberg. Contoh ini memodifikasi semua baris yang memiliki nomor genap di `c_customer_sk` kolom.

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

Operasi ini menggunakan copy-on-write strategi default, sehingga menulis ulang semua file data yang terkena dampak.

Meningkatkan data dalam tabel Iceberg

Upserting data mengacu pada memasukkan catatan data baru dan memperbarui catatan data yang ada dalam satu transaksi. Untuk meningkatkan data ke dalam tabel Iceberg, Anda menggunakan pernyataan tersebut. `SQL MERGE INTO`

Contoh berikut meningkatkan isi tabel`{UPSERT_TABLE_NAME}` di dalam tabel: `{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 *
""")
```

- Jika catatan pelanggan yang `{UPSERT_TABLE_NAME}` sudah ada di dalam `{TABLE_NAME}` dengan yang sama`c_customer_id`, `c_email_address` nilai `{UPSERT_TABLE_NAME}` rekaman akan menggantikan nilai yang ada (operasi pembaruan).
- Jika catatan pelanggan yang masuk `{UPSERT_TABLE_NAME}` tidak ada di`{TABLE_NAME}`, `{UPSERT_TABLE_NAME}` catatan ditambahkan ke `{TABLE_NAME}` (operasi sisipkan).

Menghapus data dalam tabel Iceberg

Untuk menghapus data dari tabel Iceberg, gunakan `DELETE FROM` ekspresi dan tentukan filter yang cocok dengan baris yang akan dihapus.

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

Jika filter cocok dengan seluruh partisi, Iceberg melakukan penghapusan metadata saja dan membiarkan file data di tempatnya. Jika tidak, ia hanya menulis ulang file data yang terpengaruh.

Metode delete mengambil file data yang dipengaruhi oleh `WHERE` klausa dan membuat salinannya tanpa catatan yang dihapus. Kemudian membuat snapshot tabel baru yang menunjuk ke file data baru. Oleh karena itu, catatan yang dihapus masih ada di snapshot tabel yang lebih lama. Misalnya, jika Anda mengambil snapshot sebelumnya dari tabel, Anda akan melihat data yang baru saja Anda hapus. Untuk informasi tentang menghapus snapshot lama yang tidak dibutuhkan dengan file data terkait untuk tujuan pembersihan, lihat bagian Memelihara file dengan menggunakan pemadatan nanti dalam panduan ini.

Membaca data

Anda dapat membaca status terbaru dari tabel Iceberg Anda di Spark dengan Spark SQL dan. DataFrames

Contoh menggunakan Spark SQL:

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

Contoh menggunakan DataFrames API:

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

Menggunakan perjalanan waktu

Setiap operasi tulis (menyisipkan, memperbarui, meningkatkan, menghapus) dalam tabel Iceberg membuat snapshot baru. Anda kemudian dapat menggunakan snapshot ini untuk perjalanan waktu — untuk kembali ke masa lalu dan memeriksa status tabel di masa lalu.

Untuk informasi tentang cara mengambil riwayat snapshot untuk tabel dengan menggunakan `snapshot-id` dan nilai waktu, lihat bagian Mengakses metadata nanti dalam panduan ini.

Kueri perjalanan waktu berikut menampilkan status tabel berdasarkan spesifik`snapshot-id`.

Menggunakan Spark SQL:

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

Menggunakan DataFrames API:

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

Kueri perjalanan waktu berikut menampilkan status tabel berdasarkan snapshot terakhir yang dibuat sebelum stempel waktu tertentu, dalam milidetik (). `as-of-timestamp`

Menggunakan Spark SQL:

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

Menggunakan DataFrames API:

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

Menggunakan kueri inkremental

Anda juga dapat menggunakan snapshot Iceberg untuk membaca data yang ditambahkan secara bertahap.

Catatan: Saat ini, operasi ini mendukung pembacaan data dari `append` snapshot. Itu tidak mendukung pengambilan data dari operasi seperti`replace`,`overwrite`, atau`delete`. Selain itu, operasi baca tambahan tidak didukung dalam sintaks Spark SQL.

Contoh berikut mengambil semua catatan ditambahkan ke tabel Iceberg antara snapshot `start-snapshot-id` (eksklusif) dan (inklusif). `end-snapshot-id`

```
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}")
)
```

Mengakses metadata

Iceberg menyediakan akses ke metadata-nya melalui SQL. Anda dapat mengakses metadata untuk setiap tabel yang diberikan (`<table_name>`) dengan menanyakan namespace. `<table_name>.<metadata_table>` Untuk daftar lengkap tabel metadata, lihat Memeriksa tabel dalam dokumentasi Gunung Es.

Contoh berikut menunjukkan cara mengakses tabel metadata sejarah Gunung Es, yang menunjukkan riwayat komit (perubahan) untuk tabel Gunung Es.

Menggunakan Spark SQL (dengan `%%sql` keajaiban) dari notebook Amazon EMR Studio:

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

Menggunakan DataFrames API:

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

Contoh output:

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

Bekerja dengan tabel Iceberg dengan menggunakan Trino

Bagian ini menjelaskan cara mengatur dan mengoperasikan tabel Iceberg dengan menggunakan Trino di Amazon EMR. Contohnya adalah kode boilerplate yang dapat Anda jalankan di EMR Amazon di cluster EC2. Contoh kode dan konfigurasi di bagian ini mengasumsikan bahwa Anda menggunakan Amazon EMR rilis emr-7.9.0.

Amazon EMR pada pengaturan EC2

1. Buat `iceberg.properties` file dengan konten berikut. `iceberg.file-format=parquet`Pengaturan menentukan format penyimpanan default untuk tabel baru jika format tidak secara eksplisit ditentukan dalam pernyataan. `CREATE TABLE`

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

2. Unggah file `iceberg.properties` ke bucket S3 Anda.
3. Buat tindakan bootstrap yang menyalin `iceberg.properties` file dari bucket S3 Anda dan menyimpannya sebagai file konfigurasi Trino di cluster EMR Amazon yang akan Anda buat. Pastikan untuk mengganti `<S3-bucket-name>` dengan nama bucket S3 Anda.

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

4. Buat cluster EMR Amazon dengan Trino diinstal dan tentukan eksekusi skrip sebelumnya sebagai tindakan bootstrap. Berikut adalah contoh AWS Command Line Interface (AWS CLI) perintah untuk membuat cluster:

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

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

di mana Anda mengganti:

- `<S3-bucket-name>`dengan nama bucket S3 Anda
- `<region>`dengan spesifik Anda Wilayah AWS
- `<key-name>`dengan key pair Anda. Jika key pair tidak ada, maka akan dibuat.
- `<IAM-service-role>`dengan peran layanan EMR Amazon Anda yang mengikuti prinsip hak istimewa paling sedikit.
- `<EMR-EC2-instance-profile>`dengan profil instans Anda.

5. Ketika klaster EMR Amazon telah diinisialisasi, Anda dapat menginisialisasi sesi Trino dengan menjalankan perintah berikut:

```
trino-cli
```

6. Di Trino CLI, Anda dapat melihat katalog dengan menjalankan:

```
SHOW CATALOGS;
```

Membuat tabel Iceberg

Untuk membuat tabel Iceberg, Anda dapat menggunakan pernyataan tersebut. `CREATE TABLE` Berikut adalah contoh pembuatan tabel yang dipartisi yang menggunakan partisi tersembunyi Iceberg:

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

> **Note**
>
> Jika Anda tidak menentukan format, `iceberg.file-format` nilai yang Anda konfigurasikan di bagian sebelumnya akan digunakan.

Untuk memasukkan data, gunakan `INSERT INTO` perintah. Inilah contohnya:

```
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');
```

Membaca dari tabel Iceberg

Anda dapat membaca status terbaru dari tabel Iceberg Anda dengan menggunakan `SELECT` pernyataan, sebagai berikut:

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

Menambah data ke dalam tabel Iceberg

Anda dapat melakukan operasi upsert (secara bersamaan menyisipkan catatan baru dan memperbarui yang sudah ada) dengan menggunakan `MERGE INTO` pernyataan. Inilah contohnya:

```
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);
```

Menghapus catatan dari tabel Iceberg

Untuk menghapus data dari tabel Iceberg, gunakan `DELETE FROM` ekspresi dan tentukan filter yang cocok dengan baris yang akan dihapus. Inilah contohnya:

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

Menanyakan metadata tabel Iceberg

Iceberg menyediakan akses ke metadata-nya melalui SQL. Anda dapat mengakses metadata untuk setiap tabel yang diberikan (`<table_name>`) dengan menanyakan namespace. `"<table_name>.$<metadata_table>"` Untuk daftar lengkap tabel metadata, lihat Memeriksa tabel dalam dokumentasi Gunung Es.

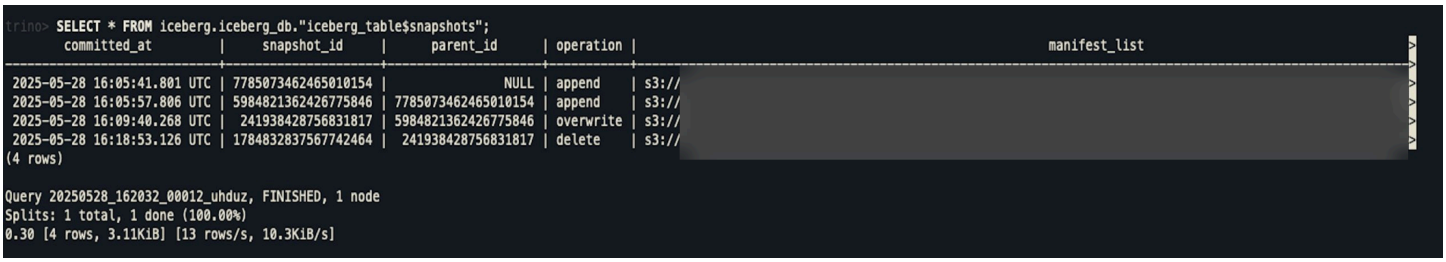
Berikut adalah contoh daftar kueri untuk memeriksa metadata Iceberg:

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

Misalnya, kueri ini:

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

menyediakan output:



Menggunakan perjalanan waktu

Setiap operasi tulis (menyisipkan, memperbarui, meningkatkan, atau menghapus) dalam tabel Iceberg membuat snapshot baru. Anda kemudian dapat menggunakan snapshot ini untuk perjalanan waktu — untuk kembali ke masa lalu dan memeriksa status tabel di masa lalu.

Kueri perjalanan waktu berikut menampilkan status tabel berdasarkan spesifik`snapshot_id`:

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

Kueri perjalanan waktu berikut menampilkan status tabel berdasarkan stempel waktu tertentu:

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

Pertimbangan saat menggunakan Iceberg dengan Trino

Operasi penulisan Trino pada tabel Iceberg mengikuti merge-on-readdesain, sehingga mereka membuat file hapus posisi alih-alih menulis ulang seluruh file data yang dipengaruhi oleh pembaruan atau penghapusan. Jika Anda ingin menggunakan copy-on-write pendekatan ini, pertimbangkan untuk menggunakan Spark untuk operasi tulis.

Bekerja dengan tabel Iceberg dengan menggunakan Amazon Data Firehose

Amazon Data Firehose adalah layanan tanpa server, tanpa kode untuk mengirimkan aliran data dari lebih dari 20 sumber seperti log, Log Amazon,, Amazon Kinesis AWS IoT Data Streams AWS WAF , dan Amazon CloudWatch Managed Streaming untuk Apache Kafka (Amazon MSK) ke tujuan seperti Amazon S3, Amazon Redshift, Snowflake, dan Split Unk.

Anda dapat menggunakan Firehose untuk langsung mengirimkan data streaming ke tabel Apache Iceberg di Amazon S3. Menggunakan Firehose, Anda dapat merutekan catatan dari satu aliran ke tabel Apache Iceberg yang berbeda, dan secara otomatis menerapkan operasi penyisipan, pembaruan, dan penghapusan ke catatan dalam tabel. Firehose menjamin pengiriman tepat sekali ke tabel Iceberg. Fitur ini membutuhkan penggunaan AWS Glue Data Catalog.

Firehose juga dapat langsung mengirimkan data streaming ke tabel Amazon S3. Tabel ini menyediakan penyimpanan yang dioptimalkan untuk beban kerja analitik skala besar, dan menyertakan fitur yang terus meningkatkan kinerja kueri dan mengurangi biaya penyimpanan untuk data tabular.

Untuk informasi tentang cara mengatur aliran Firehose untuk mengirimkan data ke tabel Apache Iceberg, lihat Mengatur aliran Firehose di dokumentasi Firehose atau posting blog Streaming data real-time ke tabel Apache Iceberg di Amazon S3 menggunakan Amazon Data Firehose.

Bekerja dengan tabel Iceberg dengan menggunakan Athena SQL

Amazon Athena menyediakan dukungan bawaan untuk Apache Iceberg, dan tidak memerlukan langkah atau konfigurasi tambahan. Bagian ini memberikan gambaran rinci tentang fitur yang didukung dan panduan tingkat tinggi untuk menggunakan Athena untuk berinteraksi dengan tabel Iceberg.

Versi dan kompatibilitas fitur

Dukungan spesifikasi tabel gunung es

Spesifikasi tabel Apache Iceberg menentukan bagaimana tabel Iceberg harus berperilaku. Athena mendukung format tabel versi 2, jadi tabel Iceberg apa pun yang Anda buat dengan konsol, CLI, atau SDK secara inheren menggunakan versi itu.

Jika Anda menggunakan tabel Iceberg yang dibuat dengan mesin lain, seperti Apache Spark di Amazon EMR AWS Glue atau, pastikan untuk mengatur versi format tabel dengan menggunakan properti tabel. Sebagai referensi, lihat bagian Membuat dan menulis tabel Gunung Es sebelumnya dalam panduan ini.

Dukungan fitur Iceberg

Anda dapat menggunakan Athena untuk membaca dan menulis ke tabel Iceberg. Saat Anda mengubah data dengan menggunakan`UPDATE`,`MERGE INTO`, dan `DELETE FROM` pernyataan, Athena hanya mendukung merge-on-read mode. Properti ini tidak dapat diubah. Untuk memperbarui atau menghapus data dengan copy-on-write, Anda harus menggunakan mesin lain seperti Apache Spark di Amazon EMR atau. AWS Glue Tabel berikut merangkum dukungan fitur Iceberg di Athena.

		Dukungan DDL		Dukungan DML		AWS Lake Formation untuk keamanan (opsional)
	Format tabel	Membuat tabel	Evolusi skema	Membaca data	Menulis data	Kontrol akses baris/kolom
Amazon Athena	Versi 2	✓	✓	✓	X X opy-on-write	✓
					✓ M erge-on-read	✓

> **Note**
>
> - Athena tidak mendukung kueri tambahan.
> - Di Athena, perbarui, hapus, dan gabungkan operasi selalu default untuk digabungkan saat baca (MoR), terlepas dari pengaturan copy on write (CoW) di properti tabel, karena CoW tidak didukung.

Bekerja dengan tabel Iceberg

Untuk memulai dengan cepat menggunakan Iceberg di Athena, lihat bagian Memulai dengan tabel Iceberg di Athena SQL sebelumnya dalam panduan ini.

Tabel berikut mencantumkan batasan dan rekomendasi.

Skenario	Batasan	Rekomendasi
Tabel generasi DDL	Tabel gunung es yang dibuat dengan mesin lain dapat memiliki properti yang tidak	Gunakan pernyataan yang setara di mesin yang membuat

Skenario	Batasan	Rekomendasi
	terpapar di Athena. Untuk tabel ini, tidak mungkin untuk menghasilkan DDL.	tabel (misalnya, `SHOW CREATE TABLE` pernyataan untuk Spark).
Awalan Amazon S3 acak dalam objek yang ditulis ke tabel Gunung Es	Secara default, tabel Iceberg yang dibuat dengan Athena mengaktifkan properti. `write.object-storage.enabled`	Untuk menonaktifkan perilaku ini dan mendapatkan kontrol penuh atas properti tabel Iceberg, buat tabel Iceberg dengan mesin lain seperti Spark di Amazon EMR atau. AWS Glue
Kueri tambahan	Saat ini tidak didukung di Athena.	Untuk menggunakan kueri inkremental untuk mengaktif kan pipeline konsumsi data tambahan, gunakan Spark di Amazon EMR atau. AWS Glue

Bekerja dengan tabel Iceberg dengan menggunakan PyIceberg

Bagian ini menjelaskan bagaimana Anda dapat berinteraksi dengan tabel Iceberg dengan menggunakan. PyIceberg Contoh yang diberikan adalah kode boilerplate yang dapat Anda jalankan di EC2 instans Amazon Linux 2023, AWS Lambdafungsi, atau lingkungan Python apa pun dengan kredensi yang dikonfigurasi dengan benar.AWS

Prasyarat

> **Note**
>
> Contoh-contoh ini menggunakan PyIceberg 1.9.1.

Untuk bekerja dengan PyIceberg, Anda perlu PyIceberg dan AWS SDK untuk Python (Boto3) menginstal. Berikut adalah contoh bagaimana Anda dapat mengatur lingkungan virtual Python untuk bekerja dengan PyIceberg dan: AWS Glue Data Catalog

1. Unduh PyIcebergdengan menggunakan penginstal paket pip python. Anda juga membutuhkan Boto3 untuk berinteraksi. Layanan AWS Anda dapat mengkonfigurasi lingkungan virtual Python lokal untuk menguji dengan menggunakan perintah ini:

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

2. Jalankan `python` untuk membuka shell Python dan menguji perintah.

Menghubungkan ke Katalog Data

Untuk mulai bekerja dengan tabel Iceberg di AWS Glue, Anda harus terlebih dahulu terhubung ke. AWS Glue Data Catalog

`load_catalog`Fungsi menginisialisasi koneksi ke Katalog Data dengan membuat objek katalog yang berfungsi sebagai antarmuka utama Anda untuk semua operasi Iceberg:

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

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

Membuat daftar dan membuat database

Untuk membuat daftar database yang ada, gunakan `list_namespaces` fungsi:

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

Untuk membuat database baru, gunakan `create_namespace` fungsi:

```
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})
```

Membuat dan menulis tabel Iceberg

Tabel yang tidak dipartisi

Berikut adalah contoh membuat tabel Iceberg yang tidak dipartisi dengan menggunakan fungsi: `create_table`

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

database_name="mydb"
```

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

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

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

Anda dapat menggunakan `list_tables` fungsi untuk memeriksa daftar tabel di dalam database:

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

Anda dapat menggunakan `append` fungsi dan PyArrow untuk menyisipkan data di dalam tabel Iceberg:

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

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

Tabel yang dipartisi

Berikut adalah contoh membuat tabel Iceberg yang dipartisi dengan partisi tersembunyi dengan menggunakan fungsi dan: `create_table` `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
)
```

Anda dapat menyisipkan data ke dalam tabel yang dipartisi dengan cara yang sama seperti untuk tabel yang tidak dipartisi. Partisi ditangani secara otomatis.

```
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)
```

Membaca data

Anda dapat menggunakan PyIceberg `scan` fungsi untuk membaca data dari tabel Iceberg Anda. Anda dapat memfilter baris, memilih kolom tertentu, dan membatasi jumlah catatan yang dikembalikan.

```
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)
```

Menghapus data

PyIceberg `delete`Fungsi ini memungkinkan Anda menghapus catatan dari tabel Anda dengan menggunakan`delete_filter`:

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

Mengakses metadata

PyIceberg menyediakan beberapa fungsi untuk mengakses metadata tabel. Berikut cara Anda dapat melihat informasi tentang snapshot tabel:

```
#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}")
```

Untuk daftar rinci metadata yang tersedia, lihat bagian referensi kode metadata dokumentasi. PyIceberg

Menggunakan perjalanan waktu

Anda dapat menggunakan snapshot tabel untuk perjalanan waktu untuk mengakses status sebelumnya dari tabel Anda. Berikut cara melihat status tabel sebelum operasi terakhir:

```
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)
```

Untuk daftar lengkap fungsi yang tersedia, lihat dokumentasi PyIceberg Python API.

Bekerja dengan spesifikasi format tabel Iceberg versi 3

Versi terbaru dari spesifikasi format tabel Apache Iceberg adalah versi 3. Versi ini memperkenalkan kemampuan canggih untuk membangun data lake berskala petabyte dengan peningkatan kinerja dan pengurangan biaya operasional. Ini mengatasi kemacetan kinerja umum yang dihadapi dengan versi 2, terutama seputar pembaruan batch dan operasi penghapusan kepatuhan.

AWS memberikan dukungan untuk vektor penghapusan dan garis keturunan baris seperti yang didefinisikan dalam spesifikasi Iceberg versi 3. Fitur-fitur ini tersedia dengan Apache Spark berikut ini. Layanan AWS

Layanan AWS	Dukungan versi 3
Amazon EMR untuk Apache Spark	Amazon EMR rilis 7.12 dan yang lebih baru
AWS Glue	Ya
AWS Glue: Iceberg REST API, pemeliharaan tabel	Ya
SageMaker Notebook Amazon Unified Studio	Ya
Tabel Amazon S3: Iceberg REST API, pemeliharaan tabel	Ya
Amazon Athena (Trino)	Tidak

Fitur utama dalam versi 3

Vektor penghapusan menggantikan file hapus posisi yang digunakan dalam versi 2 dengan format biner efisien yang disimpan sebagai file Puffin. Ini menghilangkan amplifikasi tulis dari pembaruan batch acak dan penghapusan kepatuhan Peraturan Perlindungan Data Umum (GDPR), dan secara signifikan mengurangi biaya pemeliharaan data baru. Organizations yang memproses pembaruan frekuensi tinggi akan melihat peningkatan langsung dalam kinerja penulisan dan mengurangi biaya penyimpanan dari lebih sedikit file kecil.

Garis keturunan baris memungkinkan pelacakan perubahan yang tepat di tingkat baris. Sistem hilir Anda dapat memproses perubahan secara bertahap, mempercepat jalur data, dan mengurangi

biaya komputasi untuk alur kerja pengambilan data perubahan (CDC). Kemampuan bawaan ini menghilangkan kebutuhan akan implementasi pelacakan perubahan khusus.

Kompatibilitas versi

Versi 3 mempertahankan kompatibilitas mundur dengan tabel versi 2. Layanan AWS mendukung tabel versi 2 dan versi 3 secara bersamaan, sehingga Anda dapat:

- Jalankan kueri di tabel versi 2 dan versi 3.
- Tingkatkan tabel versi 2 yang ada ke versi 3 tanpa penulisan ulang data.
- Jalankan kueri perjalanan waktu yang mencakup snapshot versi 2 dan versi 3.
- Gunakan evolusi skema dan partisi tersembunyi di seluruh versi tabel.

Memulai dengan versi 3

Prasyarat

Sebelum bekerja dengan tabel versi 3, pastikan Anda memiliki:

- An Akun AWS dengan izin AWS Identity and Access Management (IAM) yang sesuai.
- Akses ke satu atau beberapa layanan AWS analitik (Amazon EMR, notebook AWS Glue Amazon SageMaker Unified Studio, atau Amazon S3 Tables).
- Bucket S3 untuk menyimpan data tabel dan metadata.
- Bucket meja untuk memulai dengan Tabel Amazon S3 atau bucket S3 serba guna jika Anda membangun infrastruktur Gunung Es Anda sendiri.
- AWS Glue Katalog yang dikonfigurasi.

Membuat tabel versi 3

Membuat tabel baru

Untuk membuat tabel Iceberg versi 3 baru, atur properti `format-version` tabel ke 3.

Menggunakan Spark SQL:

```
CREATE TABLE IF NOT EXISTS myns.orders_v3 (
    order_id bigint,
```

```
    customer_id string,
    order_date date,
    total_amount decimal(10,2),
    status string,
    created_at timestamp
)
USING iceberg
TBLPROPERTIES (
    'format-version' = '3'
)
```

Memutakhirkan tabel versi 2 ke versi 3

Anda dapat memutakhirkan tabel versi 2 yang ada ke versi 3 secara atom tanpa menulis ulang data.

Menggunakan Spark SQL:

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

> **Important**
>
> Versi 3 adalah upgrade satu arah. Setelah tabel ditingkatkan dari versi 2 ke versi 3, itu tidak dapat diturunkan kembali ke versi 2 melalui operasi standar.

Apa yang terjadi selama upgrade:

- Cuplikan metadata baru dibuat secara atom.
- File data Parket yang ada digunakan kembali.
- Bidang garis keturunan baris ditambahkan ke metadata tabel.

Setelah peningkatan:

- Pemadatan berikutnya akan menghapus file hapus versi lama 2.
- Modifikasi baru akan menggunakan file vektor penghapusan versi 3.

Pemutakhiran tidak melakukan pengisian ulang historis dari catatan pelacakan perubahan garis keturunan baris.

Mengaktifkan vektor penghapusan

Untuk memanfaatkan vektor penghapusan untuk pembaruan, penghapusan, dan penggabungan, konfigurasikan mode tulis Anda.

Menggunakan Spark SQL:

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

Pengaturan ini memastikan bahwa operasi pembaruan, penghapusan, dan penggabungan membuat file vektor penghapusan alih-alih menulis ulang seluruh file data.

Menggunakan garis keturunan baris untuk melacak perubahan

Versi 3 secara otomatis menambahkan bidang metadata garis keturunan baris untuk melacak perubahan.

Menggunakan Spark SQL:

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

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

`_row_id`Bidang ini secara unik mengidentifikasi setiap baris, dan `_last_updated_sequence_number` melacak saat baris terakhir dimodifikasi. Gunakan bidang ini untuk:

- Identifikasi baris yang diubah untuk pemrosesan inkremental.

- Lacak garis keturunan data untuk kepatuhan.
- Optimalkan saluran pipa CDC.
- Kurangi biaya komputasi dengan hanya memproses perubahan.

Praktik terbaik untuk versi 3

Kapan menggunakan versi 3

Pertimbangkan untuk meningkatkan ke, atau memulai dengan, versi 3 saat:

- Anda sering melakukan pembaruan atau penghapusan batch.
- Anda harus memenuhi persyaratan penghapusan GDPR atau kepatuhan.
- Beban kerja Anda melibatkan peningkatan frekuensi tinggi.
- Anda memerlukan alur kerja CDC yang efisien.
- Anda ingin mengurangi biaya penyimpanan dari file kecil.
- Anda membutuhkan kemampuan pelacakan perubahan yang lebih baik.

Mengoptimalkan kinerja tulis

- Aktifkan vektor penghapusan untuk beban kerja pembaruan-berat:

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

- Konfigurasikan ukuran file yang sesuai:

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

Mengoptimalkan kinerja baca

- Gunakan garis keturunan baris untuk pemrosesan inkremental.

- Gunakan perjalanan waktu untuk mengakses data historis tanpa menyalin.
- Aktifkan pengumpulan statistik untuk perencanaan kueri yang lebih baik.

Strategi migrasi

Saat Anda bermigrasi dari versi 2 ke versi 3, ikuti praktik terbaik berikut ini:

- Uji di lingkungan non-produksi terlebih dahulu untuk memvalidasi proses dan kinerja peningkatan.
- Tingkatkan selama periode aktivitas rendah untuk meminimalkan dampak pada operasi bersamaan.
- Pantau kinerja awal, dan lacak metrik setelah peningkatan.
- Jalankan pemadatan untuk mengkonsolidasikan file hapus setelah peningkatan.
- Perbarui dokumentasi tim Anda untuk mencerminkan fitur versi 3.

Pertimbangan kompatibilitas

- Versi mesin — Pastikan bahwa semua mesin yang mengakses tabel mendukung versi 3.
- Alat pihak ketiga — Verifikasi kompatibilitas versi 3 alat Anda sebelum Anda meningkatkan.
- Strategi Backup — Uji prosedur pemulihan berbasis snapshot.
- Pemantauan - Perbarui dasbor pemantauan untuk metrik spesifik versi 3.

Pemecahan masalah

Masalah umum

Kesalahan: “format-versi 3 tidak didukung”

- Verifikasi bahwa versi mesin Anda mendukung versi 3. Untuk spesifik, lihat tabel di awal bagian ini.
- Periksa kompatibilitas katalog.
- Pastikan Anda menggunakan versi terbaru Layanan AWS.

Degradasi kinerja setelah peningkatan

- Verifikasi bahwa tidak ada kegagalan pemadatan pemadatan. Untuk informasi selengkapnya, lihat Pencatatan dan pemantauan untuk Tabel S3 di dokumentasi Amazon S3.
- Konfirmasikan bahwa vektor penghapusan diaktifkan. Properti berikut harus ditetapkan:

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

  Anda dapat memverifikasi properti tabel dengan kode berikut:

```
DESCRIBE FORMATTED myns.orders_v3
```

- Tinjau strategi partisi Anda. Over-partisi dapat menyebabkan file kecil. Jalankan kueri berikut untuk mendapatkan ukuran file rata-rata untuk tabel Anda:

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

Ketidakcocokan dengan alat pihak ketiga

- Verifikasi bahwa alat ini mendukung spesifikasi versi 3.
- Pertimbangkan untuk mempertahankan tabel versi 2 untuk alat yang tidak didukung.
- Hubungi vendor alat untuk garis waktu dukungan versi 3 mereka.

Mendapatkan bantuan

- Untuk masalah Layanan AWS-spesifik, hubungi AWS Dukungan.
- Untuk mendapatkan bantuan dari komunitas Iceberg, gunakan saluran Iceberg Slack.
- Untuk informasi tentang penggunaan Layanan AWS untuk mengelola beban kerja analitik Anda, lihat Analytics di AWS.

Harga

- Harga komputasi dan penyimpanan Amazon EMR

- SageMakerHarga Amazon
- AWS Glue job run dan harga Katalog Data
- Penyimpanan Tabel S3 dan permintaan harga

Ketersediaan

Dukungan spesifikasi format tabel Iceberg versi 3 tersedia di semua tempat Wilayah AWS Amazon EMR,, AWS Glue AWS Glue Data Catalog, dan Tabel S3 beroperasi. Untuk ketersediaan Wilayah, lihat Layanan AWS berdasarkan Wilayah.

Sumber daya tambahan

- Dokumentasi Apache Iceberg
- Spesifikasi tabel Apache Iceberg
- Panduan untuk memigrasi data tabular dari Amazon S3 ke Tabel S3
- Tutorial: Memulai dengan Tabel S3

Migrasi tabel yang ada ke Iceberg

Bagian ini berfokus pada migrasi tabel gaya HIVE yang ada ke format Iceberg. Ini berlaku untuk tabel yang menggunakan format tradisional yang kompatibel dengan HIVE seperti Apache Parquet atau Apache ORC. Informasi ini tidak berlaku untuk tabel yang sudah menggunakan format tabel modern seperti Linux Foundation Delta Lake atau Apache Hudi.

Untuk memigrasikan tabel gaya HIVE saat ini ke format Iceberg, Anda dapat menggunakan migrasi data di tempat atau penuh:

- Migrasi di tempat adalah proses menghasilkan file metadata Iceberg di atas file data yang ada.
- Migrasi data lengkap membuat lapisan metadata Iceberg dan juga menulis ulang file data yang ada dari tabel asli ke tabel Iceberg baru.

Bagian berikut memberikan gambaran rinci dari setiap metode migrasi, termasuk step-by-step instruksi dan pertimbangan untuk implementasi. Untuk informasi selengkapnya tentang strategi migrasi ini, lihat bagian Migrasi Tabel dari dokumentasi Gunung Es.

Setelah Anda meninjau detail metode migrasi data di tempat dan lengkap, lihat dua bagian utama berikut untuk membantu proses pengambilan keputusan Anda:

- Memilih strategi migrasi memberikan panduan melalui serangkaian pertanyaan dan skenario, untuk membantu Anda menentukan pendekatan migrasi yang paling sesuai berdasarkan persyaratan spesifik dan kasus penggunaan Anda.
- Ringkasan opsi migrasi menyediakan tabel komprehensif yang membandingkan karakteristik dan pertimbangan utama di berbagai opsi migrasi. Tabel ini berfungsi sebagai panduan referensi cepat dan menawarkan perbandingan fitur untuk membantu Anda memahami pertukaran teknis antar metode.

Migrasi di tempat

Migrasi di tempat menghilangkan kebutuhan untuk menulis ulang semua file data Anda. Sebagai gantinya, file metadata Iceberg dibuat dan ditautkan ke file data Anda yang ada. Metode ini biasanya lebih cepat dan lebih hemat biaya, terutama untuk kumpulan data besar atau tabel yang memiliki format file yang kompatibel seperti Parket, Avro, dan ORC.

> **Note**
>
> Migrasi di tempat tidak dapat digunakan saat bermigrasi ke Tabel Amazon S3.

Iceberg menawarkan dua opsi utama untuk menerapkan migrasi di tempat:

- Menggunakan prosedur snapshot untuk membuat tabel Iceberg baru sambil menjaga tabel sumber tidak berubah. Untuk informasi selengkapnya, lihat Tabel Snapshot dalam dokumentasi Gunung Es.
- Menggunakan prosedur migrasi untuk membuat tabel Iceberg baru sebagai substitusi untuk tabel sumber. Untuk informasi selengkapnya, lihat Memigrasi Tabel di dokumentasi Gunung Es. Meskipun prosedur ini bekerja dengan Hive Metastore (HMS), saat ini tidak kompatibel dengan. AWS Glue Data CatalogReplikasi prosedur migrasi tabel di AWS Glue Data Catalog bagian nanti dalam panduan ini memberikan solusi untuk mencapai hasil yang serupa dengan Katalog Data.

Setelah Anda melakukan migrasi di tempat dengan menggunakan salah satu `snapshot` atau`migrate`, beberapa file data mungkin tetap tidak bermigrasi. Ini biasanya terjadi ketika penulis terus menulis ke tabel sumber selama atau setelah migrasi. Untuk memasukkan file-file yang tersisa ini ke dalam tabel Iceberg Anda, Anda dapat menggunakan prosedur add_files. Untuk informasi selengkapnya, lihat Menambahkan File di dokumentasi Gunung Es.

Katakanlah Anda memiliki `products` tabel berbasis Parket yang dibuat dan dihuni di Athena sebagai berikut:

```
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');
```

Bagian berikut menjelaskan bagaimana Anda dapat menggunakan `snapshot` dan `migrate` prosedur dengan tabel ini.

Opsi 1: prosedur snapshot

`snapshot`Prosedur ini membuat tabel Iceberg baru yang memiliki nama berbeda tetapi mereplikasi skema dan partisi tabel sumber. Operasi ini meninggalkan tabel sumber sama sekali tidak berubah baik selama dan setelah tindakan. Ini secara efektif membuat salinan tabel yang ringan, yang sangat berguna untuk menguji skenario atau eksplorasi data tanpa risiko modifikasi pada sumber data asli. Pendekatan ini memungkinkan periode transisi di mana tabel asli dan tabel Gunung Es tetap tersedia (lihat catatan di akhir bagian ini). Saat pengujian selesai, Anda dapat memindahkan tabel Iceberg baru Anda ke produksi dengan mentransisikan semua penulis dan pembaca ke tabel baru.

Anda dapat menjalankan `snapshot` prosedur dengan menggunakan Spark di model penyebaran EMR Amazon apa pun (misalnya, Amazon EMR di EC2, Amazon EMR di EKS, EMR EMR Tanpa Server) dan. AWS Glue

Untuk menguji migrasi di tempat dengan prosedur `snapshot` Spark, ikuti langkah-langkah berikut:

1. Luncurkan aplikasi Spark dan konfigurasikan sesi Spark dengan pengaturan berikut:
   - `"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. Jalankan `snapshot` prosedur untuk membuat tabel Iceberg baru yang menunjuk ke file data tabel asli:

```
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)
```

   Rangka data keluaran berisi `imported_files_count` (jumlah file yang ditambahkan).
3. Validasi tabel baru dengan menanyakannya:

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

Catatan

- Setelah Anda menjalankan prosedur, modifikasi file data apa pun pada tabel sumber akan membuat tabel yang dihasilkan tidak sinkron. File baru yang Anda tambahkan tidak akan terlihat di tabel Iceberg, dan file yang Anda hapus akan memengaruhi kemampuan kueri di tabel Iceberg. Untuk menghindari masalah sinkronisasi:
  - Jika tabel Iceberg baru dimaksudkan untuk penggunaan produksi, hentikan semua proses yang menulis ke tabel asli dan arahkan ke tabel baru.
  - Jika Anda memerlukan periode transisi atau jika tabel Iceberg baru untuk tujuan pengujian, lihat Menyinkronkan tabel Iceberg setelah migrasi di tempat nanti di bagian ini untuk panduan tentang mempertahankan sinkronisasi tabel.
- Bila Anda menggunakan `snapshot` prosedur, `gc.enabled` properti diatur ke `false` dalam properti tabel tabel Iceberg dibuat. Pengaturan ini melarang tindakan seperti`expire_snapshots`,`remove_orphan_files`, atau `DROP TABLE` dengan `PURGE` opsi, yang secara fisik akan menghapus file data. Operasi penghapusan atau penggabungan gunung es, yang tidak berdampak langsung pada file sumber, masih diperbolehkan.
- Untuk membuat tabel Iceberg baru berfungsi penuh, tanpa batasan tindakan yang menghapus file data secara fisik, Anda dapat mengubah properti `gc.enabled` tabel menjadi. `true` Namun, pengaturan ini akan memungkinkan tindakan yang memengaruhi file data sumber, yang dapat merusak akses ke tabel asli. Oleh karena itu, ubah `gc.enabled` properti hanya jika Anda tidak perlu lagi mempertahankan fungsionalitas tabel asli. Contoh:

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

Opsi 2: prosedur migrasi

`migrate`Prosedur ini membuat tabel Iceberg baru yang memiliki nama, skema, dan partisi yang sama dengan tabel sumber. Ketika prosedur ini berjalan, ia mengunci tabel sumber dan mengganti namanya menjadi `<table_name>_BACKUP_` (atau nama khusus yang ditentukan oleh parameter `backup_table_name` prosedur).

> **Note**
>
> Jika Anda mengatur parameter `drop_backup` prosedur ke`true`, tabel asli tidak akan dipertahankan sebagai cadangan.

Akibatnya, prosedur `migrate` tabel mengharuskan semua modifikasi yang mempengaruhi tabel sumber untuk dihentikan sebelum tindakan dilakukan. Sebelum Anda menjalankan `migrate` prosedur:

- Hentikan semua penulis yang berinteraksi dengan tabel sumber.
- Ubah pembaca dan penulis yang tidak mendukung Iceberg secara asli untuk mengaktifkan dukungan Iceberg.

Contoh:

- Athena terus bekerja tanpa modifikasi.
- Spark membutuhkan:
  - File Iceberg Java Archive (JAR) untuk disertakan dalam classpath (lihat Bekerja dengan Gunung Es di Amazon EMR dan Bekerja dengan Gunung AWS Glue Es di bagian sebelumnya dalam panduan ini).
  - Konfigurasi katalog sesi Spark berikut (menggunakan `SparkSessionCatalog` untuk menambahkan dukungan Iceberg sambil mempertahankan fungsionalitas katalog bawaan untuk tabel non-Iceberg):
    - `"spark.sql.extensions":"org.apache.iceberg.spark.extensions.IcebergSparkSes`
    - `"spark.sql.catalog.spark_catalog":"org.apache.iceberg.spark.SparkSessionCat`
    - `"spark.sql.catalog.spark_catalog.type":"glue"`
    - `"spark.hadoop.hive.metastore.client.factory.class":"com.amazonaws.glue.cata`

Setelah Anda menjalankan prosedur, Anda dapat me-restart penulis Anda dengan konfigurasi Iceberg baru mereka.

Saat ini, `migrate` prosedur tidak kompatibel dengan AWS Glue Data Catalog, karena Katalog Data tidak mendukung `RENAME` operasi. Oleh karena itu, kami menyarankan Anda menggunakan prosedur ini hanya ketika Anda bekerja dengan Hive Metastore. Jika Anda menggunakan Katalog Data, lihat bagian selanjutnya untuk pendekatan alternatif.

Anda dapat menjalankan `migrate` prosedur di semua model penyebaran EMR Amazon (Amazon EMR di EC2, Amazon EMR di EKS, EMR EMR Tanpa Server AWS Glue) dan, tetapi memerlukan koneksi yang dikonfigurasi ke Hive Metastore. Amazon EMR di EC2 adalah pilihan yang disarankan karena menyediakan konfigurasi Hive Metastore bawaan, yang meminimalkan kompleksitas penyiapan.

Untuk menguji migrasi di tempat dengan prosedur `migrate` Spark dari EMR Amazon di kluster EC2 yang dikonfigurasi dengan Hive Metastore, ikuti langkah-langkah berikut:

1. Luncurkan aplikasi Spark dan konfigurasikan sesi Spark untuk menggunakan implementasi katalog Iceberg Hive. Misalnya, jika Anda menggunakan `pyspark` CLI:

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

2. Buat `products` tabel di Hive Metastore. Ini adalah tabel sumber, yang sudah ada dalam migrasi tipikal.
   a. Buat tabel Hive `products` eksternal di Hive Metastore untuk menunjuk ke data yang ada di Amazon S3:

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

b. Tambahkan partisi yang ada dengan menggunakan `MSCK REPAIR TABLE` perintah:

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

c. Konfirmasikan bahwa tabel berisi data dengan menjalankan `SELECT` kueri:

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

Contoh output:

```
>>> 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. Gunakan prosedur Icebergmigrate:

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

df_res.show()
```

Rangka data keluaran berisi `migrated_files_count` (jumlah file yang ditambahkan ke tabel Iceberg):

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

4. Konfirmasikan bahwa tabel cadangan telah dibuat:

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

Contoh output:

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

5. Validasi operasi dengan menanyakan tabel Iceberg:

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

> **Catatan**
>
> - Setelah Anda menjalankan prosedur, semua proses saat ini yang kueri atau tulis ke tabel sumber akan terpengaruh jika tidak dikonfigurasi dengan benar dengan dukungan Iceberg. Oleh karena itu, kami menyarankan Anda mengikuti langkah-langkah ini:
>
>   1. Hentikan semua proses dengan menggunakan tabel sumber sebelum migrasi.
>   2. Lakukan migrasi.
>   3. Aktifkan kembali proses dengan menggunakan pengaturan Iceberg yang tepat.

- Jika modifikasi file data terjadi selama proses migrasi (file baru ditambahkan atau file dihapus), tabel yang dihasilkan akan tidak sinkron. Untuk opsi sinkronisasi, lihat Menyinkronkan tabel Iceberg setelah migrasi di tempat nanti di bagian ini.

Mereplikasi prosedur migrasi tabel di AWS Glue Data Catalog

Anda dapat mereplikasi hasil prosedur migrasi di AWS Glue Data Catalog (mencadangkan tabel asli dan menggantinya dengan tabel Iceberg) dengan mengikuti langkah-langkah berikut:

1. Gunakan prosedur snapshot untuk membuat tabel Iceberg baru yang menunjuk ke file data tabel asli.
2. Cadangkan metadata tabel asli di Katalog Data:
   a. Gunakan GetTableAPI untuk mengambil definisi tabel sumber.
   b. Gunakan GetPartitionsAPI untuk mengambil definisi partisi tabel sumber.
   c. Gunakan CreateTableAPI untuk membuat tabel cadangan di Katalog Data.
   d. Gunakan BatchCreatePartitionAPI CreatePartitionatau untuk mendaftarkan partisi ke tabel cadangan di Katalog Data.
3. Ubah properti tabel `gc.enabled` Iceberg `false` untuk mengaktifkan operasi tabel lengkap.
4. Jatuhkan meja aslinya.
5. Temukan file JSON metadata tabel Iceberg di folder metadata lokasi root tabel.
6. Daftarkan tabel baru di Katalog Data dengan menggunakan prosedur register_table dengan nama tabel asli dan lokasi `metadata.json` file yang dibuat oleh prosedur: `snapshot`

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

Menyimpan tabel Iceberg tetap sinkron setelah migrasi di tempat

`add_files`Prosedur ini menyediakan cara yang fleksibel untuk memasukkan data yang ada ke dalam tabel Iceberg. Secara khusus, ia mendaftarkan file data yang ada (seperti file Parquet) dengan mereferensikan jalur absolutnya di lapisan metadata Iceberg. Secara default, prosedur menambahkan file dari semua partisi tabel ke tabel Iceberg, tetapi Anda dapat secara selektif menambahkan file dari partisi tertentu. Pendekatan selektif ini sangat berguna dalam beberapa skenario:

- Ketika partisi baru ditambahkan ke tabel sumber setelah migrasi awal.
- Ketika file data ditambahkan ke atau dihapus dari partisi yang ada setelah migrasi awal. Namun, menambahkan kembali partisi yang dimodifikasi membutuhkan penghapusan partisi terlebih dahulu. Informasi lebih lanjut tentang ini disediakan nanti di bagian ini.

Berikut adalah beberapa pertimbangan untuk menggunakan `add_file` prosedur setelah migrasi di tempat (`snapshot`atau`migrate`) dilakukan, agar tabel Iceberg baru tetap sinkron dengan file data sumber:

- Ketika data baru ditambahkan ke partisi baru di tabel sumber, gunakan `add_files` prosedur dengan `partition_filter` opsi untuk secara selektif memasukkan penambahan ini ke dalam tabel Iceberg:

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

atau:

```
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)
```

- `add_files`Prosedur memindai file baik di seluruh tabel sumber atau di partisi tertentu ketika Anda menentukan `partition_filter` opsi, dan mencoba untuk menambahkan semua file yang ditemukan ke tabel Iceberg. Secara default, properti `check_duplicate_files` prosedur diatur ke`true`, yang mencegah prosedur berjalan jika file sudah ada di tabel Iceberg. Ini penting karena tidak ada opsi bawaan untuk melewati file yang ditambahkan sebelumnya, dan menonaktifkan `check_duplicate_files` akan menyebabkan file ditambahkan dua kali, membuat duplikat. Saat file baru ditambahkan ke tabel sumber, ikuti langkah-langkah berikut:

  1. Untuk partisi baru, gunakan `add_files` dengan a `partition_filter` untuk mengimpor hanya file dari partisi baru.
  2. Untuk partisi yang ada, pertama-tama hapus partisi dari tabel Iceberg, dan kemudian jalankan kembali `add_files` untuk partisi itu, dengan menentukan. `partition_filter` Contoh:

```
# 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**
>
> Setiap `add_files` operasi menghasilkan snapshot tabel Iceberg baru dengan data yang ditambahkan.

Memilih strategi migrasi di tempat yang tepat

Untuk memilih strategi migrasi terbaik di tempat, pertimbangkan pertanyaan dalam tabel berikut.

Pertanyaan	Rekomendasi	Penjelasan
Apakah Anda ingin bermigrasi dengan cepat tanpa menulis ulang data sambil menjaga tabel Hive dan Iceberg dapat diakses untuk pengujian atau transisi bertahap?	`snapshot`prosedur diikuti dengan `add_files` prosedur	Gunakan `snapshot` prosedur untuk membuat tabel Iceberg baru dengan mengkloning skema dan referensi file data, tanpa memodifikasi tabel sumber. Gunakan `add_files` prosedur untuk memasukkan partisi yang ditambahkan atau dimodifikasi setelah migrasi, mencatat bahwa menambahkan kembali partisi yang dimodifikasi memerlukan penghapusan partisi terlebih dahulu.
Apakah Anda menggunakan Hive Metastore dan apakah Anda ingin segera mengganti tabel Hive Anda dengan tabel Iceberg, tanpa menulis ulang data?	`migrate`prosedur diikuti dengan `add_files` prosedur	Gunakan `migrate` prosedur untuk membuat tabel Iceberg, mencadangkan tabel sumber, dan mengganti tabel asli dengan versi Iceberg.

Pertanyaan	Rekomendasi	Penjelasan
		Catatan: Opsi ini kompatibel dengan Hive Metastore tetapi tidak dengan. AWS Glue Data Catalog Gunakan `add_files` prosedur untuk memasukkan partisi yang ditambahkan atau dimodifikasi setelah migrasi, mencatat bahwa menambahkan kembali partisi yang dimodifikasi memerlukan penghapusan partisi terlebih dahulu.

Pertanyaan	Rekomendasi	Penjelasan
Apakah Anda menggunakan AWS Glue Data Catalog dan apakah Anda ingin segera mengganti tabel Hive Anda dengan tabel Iceberg, tanpa menulis ulang data?	Adaptasi `migrate` prosedur, diikuti dengan `add_files` prosedur	Mereplikasi perilaku `migrate` prosedur: 1. Gunakan `snapshot` untuk membuat tabel Iceberg. 2. Cadangkan metadata tabel asli dengan menggunakan. AWS Glue APIs 3. Aktifkan `gc.enabled` pada properti tabel Iceberg. 4. Jatuhkan meja aslinya. 5. Gunakan `register_table` untuk membuat entri tabel baru dengan nama asli. Catatan: Opsi ini memerlukan penanganan manual panggilan AWS Glue API untuk pencadangan metadata. Gunakan `add_files` prosedur untuk memasukkan partisi yang ditambahkan atau dimodifikasi setelah migrasi, mencatat bahwa menambahkan kembali partisi yang dimodifikasi memerlukan penghapusan partisi terlebih dahulu.

Migrasi data lengkap

Migrasi data lengkap membuat ulang file data serta metadata. Pendekatan ini membutuhkan waktu lebih lama dan membutuhkan sumber daya komputasi tambahan dibandingkan dengan migrasi di tempat. Namun, migrasi data penuh menawarkan peluang signifikan untuk meningkatkan kualitas tabel dan mengoptimalkan penyimpanan data dan pola akses.

Selama migrasi data penuh, Anda dapat melakukan beberapa operasi yang bermanfaat, seperti validasi data untuk memastikan integritas dan kebenaran, modifikasi skema untuk memenuhi persyaratan saat ini dengan lebih baik, dan penyesuaian strategi partisi untuk meningkatkan kinerja kueri. Anda juga dapat mengurutkan ulang data untuk mengoptimalkan pola akses umum, menerapkan partisi tersembunyi Iceberg untuk meningkatkan efisiensi kueri, dan melakukan konversi format file (misalnya, dari CSV ke Parket) jika diinginkan.

Kemampuan ini membuat migrasi data lengkap ideal untuk transisi ke format Iceberg dan untuk menyempurnakan dan mengoptimalkan strategi penyimpanan data Anda secara komprehensif. Meskipun migrasi data penuh membutuhkan lebih banyak waktu dan sumber daya di muka, peningkatan yang dihasilkan dalam kualitas data, organisasi, dan kinerja kueri dapat memberikan manfaat jangka panjang. Untuk menerapkan migrasi data lengkap, gunakan salah satu opsi berikut:

- Gunakan pernyataan `CREATE TABLE ... AS SELECT` (CTAS) di Spark (di Amazon EMR atau) AWS Glue atau di Athena. Anda dapat mengatur spesifikasi partisi dan properti tabel untuk tabel Iceberg baru dengan menggunakan klausa `PARTITIONED BY` and`TBLPROPERTIES`. Anda dapat mengubah skema dan partisi untuk tabel baru sesuai dengan kebutuhan Anda alih-alih mewarisinya dari tabel sumber.
- Baca dari tabel sumber dan tulis data sebagai tabel Iceberg baru dengan menggunakan Spark di Amazon EMR atau. AWS Glue Untuk informasi selengkapnya, lihat Membuat tabel di dokumentasi Gunung Es.

Memilih strategi migrasi

Saat beralih ke format Iceberg, pilihan antara migrasi di tempat dan migrasi penuh sangat penting. Untuk menentukan pendekatan yang paling cocok untuk kebutuhan spesifik Anda, pertimbangkan pertanyaan dan rekomendasi berikut:

Pertanyaan	Rekomendasi
Apa format file data (misalnya, CSV atau Apache Parquet)?	• Pertimbangkan migrasi di tempat jika format file tabel Anda adalah Parket, ORC, atau Avro. • Untuk format lain seperti CSV, JSON, dan sebagainya, gunakan migrasi data lengkap.
Apakah Anda ingin memperbarui atau mengkonsolidasikan skema tabel?	• Jika Anda ingin mengembangkan skema tabel dengan menggunakan kemampuan asli Iceberg, pertimbangkan migrasi di tempat. Misalnya, Anda dapat mengganti nama kolom setelah migrasi. (Skema dapat diubah di lapisan metadata Iceberg.) • Jika Anda ingin menghapus seluruh kolom karena tidak lagi diperlukan, kami sarankan Anda menggunakan migrasi data lengkap.
Apakah tabel akan mendapat manfaat dari mengubah strategi partisi?	• Jika pendekatan partisi Iceberg memenuhi persyaratan Anda (misalnya, data baru disimpan dengan menggunakan tata letak partisi baru sementara partisi yang ada tetap apa adanya), pertimbangkan migrasi di tempat. • Jika Anda ingin menggunakan partisi tersembunyi di tabel Anda, pertimbangkan migrasi data lengkap. Untuk informasi selengkapnya tentang partisi tersembunyi, lihat bagian Praktik terbaik.
Apakah tabel akan mendapat manfaat dari menambahkan atau mengubah strategi urutan pengurutan?	• Menambahkan atau mengubah urutan data Anda memerlukan penulisan ulang kumpulan data. Dalam hal ini, pertimbangkan untuk menggunakan migrasi data lengkap. • Untuk tabel besar yang sangat mahal untuk menulis ulang semua partisi tabel, pertimbangkan untuk menggunakan migrasi di tempat dan menjalankan pemadatan (dengan pengurutan diaktifkan) untuk partisi yang paling sering diakses.
Apakah tabel memiliki banyak file kecil?	• Menggabungkan file kecil menjadi file yang lebih besar memerlukan penulisan ulang kumpulan data. Dalam hal ini, pertimbangkan untuk menggunakan migrasi data lengkap.

Pertanyaan	Rekomendasi
	• Untuk tabel besar yang sangat mahal untuk menulis ulang semua partisi tabel, pertimbangkan untuk menggunakan migrasi di tempat dan menjalankan pemadatan (dengan pengurutan diaktifkan) untuk partisi yang paling sering diakses.

Ringkasan opsi migrasi

Tabel ini merangkum karakteristik dan pertimbangan utama untuk setiap opsi migrasi.

Fitur	Migrasi di tempat snapshot	Migrasi di tempat bermigrasi	Migrasi data lengkap CTAS atau (BUAT TABELAN+SISIPKAN)
Perbaikata letak data sebagai bagian dari proses migrasi			
• Menguan ulang data	T	T	Ya
• Ubah partisi (misalr, untuk	T	T	Ya

Fitur	Migrasi di tempat snapshot	Migrasi di tempat bermigrasi	Migrasi data lengkap CTAS atau (BUAT TABELAN+SISIPKAN)
menggan partisi tersemyi Iceber			
• Ubah skema tabel	T	T	Ya
• Optiman ukurar file	T	T	Ya
• Validas skema data yang ada sebelu menar an data	T	T	Ya
Format file yang didukung	Parket, Avro, ORC	Parket, Avro, ORC	Parket, Avro, ORC, JSON, CSV

Fitur	Migrasi di tempat snapshot	Migrasi di tempat bermigrasi	Migrasi data lengkap CTAS atau (BUAT TABELAN+SISIPKAN)
Penggantian tabel sumber dengan tabel Iceberg	 T (membuat tabel baru, tetapi dengan langkah-langkah tambahan Anda dapat mengganti tabel sumber)	 Y (membuat tabel cadangan dan mengganti tabel sumber dengan tabel Iceberg)	 Tidak (membuat tabel baru)
Dampak tabel sumber			
• Operasi penghapusan file pada tabel Iceberg (`expireSnapshots` operasi menjatuhkan tabel dengan pembekuan)	Tabel sumber rusak	Merusak tabel cadangan	Aman, sumber tidak terpengaruh

Fitur	Migrasi di tempat snapshot	Migrasi di tempat bermigrasi	Migrasi data lengkap CTAS atau (BUAT TABELAN+SISIPKAN)
Dampak tabel gunung es			
• Dampa jika file tabel sumbe dihapu	Meja Gunung Es Corrupts	Meja Gunung Es Corrupts	Tidak berdampak pada tabel Iceberg
• Dampa jika file baru ditamb an pada lokasi tabel sumbe	Tidak terlihat di meja baru (perlu menggabungkan partisi dengan`add_files`)	Tidak terlihat di meja baru (perlu menggabungkan partisi dengan`add_files`)	Tidak terlihat di meja baru (perlu `INSERT INTO` ke tabel baru)
Biaya	Rendah	Rendah	Lebih tinggi (penulisan ulang data lengkap)
Kecepata migrasi	Cepat	Cepat	Lebih lambat

Fitur	Migrasi di tempat snapshot	Migrasi di tempat bermigrasi	Migrasi data lengkap CTAS atau (BUAT TABELAN+SISIPKAN)
Dapat digunaka untuk bermigrasi ke Tabel Amazon S3	 T	 T	 Ya
Membutu an DDL manual	 T (skema dan partisi disalin dari tabel sumber)	 T (skema dan partisi disalin dari tabel sumber)	Jika menggunakan CTAS, hanya memerlukan menentukan partisi
Penggun n terbaik	Migrasi cepat tanpa menulis ulang data, memungkinkan side-by-side penggunaan Hive dan Iceberg untuk pengujian atau transisi bertahap.	Mengganti tabel Hive di tempat tanpa menulis ulang data, ketika peralihan langsung dapat diterima.	Optimalisasi Gunung Es penuh dengan penulisan ulang data. Ideal saat mendesain ulang partisi atau skema, atau meningkatkan tata letak dan kinerja. Selalu direkomendasikan jika memungkinkan.

Praktik terbaik untuk mengoptimalkan beban kerja Apache Iceberg

Iceberg adalah format tabel yang dirancang untuk menyederhanakan pengelolaan data lake dan meningkatkan kinerja beban kerja. Kasus penggunaan yang berbeda mungkin memprioritaskan aspek yang berbeda seperti biaya, kinerja baca, kinerja tulis, atau retensi data, sehingga Iceberg menawarkan opsi konfigurasi untuk mengelola trade-off ini. Bagian ini memberikan wawasan untuk mengoptimalkan dan menyempurnakan beban kerja Iceberg Anda untuk memenuhi kebutuhan Anda.

Topik

Praktik terbaik umum

Terlepas dari kasus penggunaan Anda, saat Anda menggunakan Apache Iceberg AWS, kami sarankan Anda mengikuti praktik terbaik umum ini.

- Gunakan format Iceberg versi 2.

  Athena menggunakan format Iceberg versi 2 secara default.

  Bila Anda menggunakan Spark di Amazon EMR AWS Glue atau untuk membuat tabel Iceberg, tentukan versi format seperti yang dijelaskan dalam dokumentasi Iceberg.
- Gunakan AWS Glue Data Catalog sebagai katalog data Anda.

  Athena menggunakan secara default AWS Glue Data Catalog .

  Saat Anda menggunakan Spark di Amazon EMR AWS Glue atau untuk bekerja dengan Iceberg, tambahkan konfigurasi berikut ke sesi Spark Anda untuk menggunakan. AWS Glue Data Catalog

Untuk informasi selengkapnya, lihat bagian Konfigurasi percikan untuk Gunung Es di AWS Glue awal panduan ini.

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

- Gunakan AWS Glue Data Catalog sebagai manajer kunci.

  Athena menggunakan AWS Glue Data Catalog as lock manager secara default untuk tabel Iceberg.

  Saat Anda menggunakan Spark di Amazon EMR AWS Glue atau untuk bekerja dengan Iceberg, pastikan untuk mengonfigurasi konfigurasi sesi Spark Anda untuk menggunakan pengelola kunci as. AWS Glue Data Catalog Untuk informasi lebih lanjut, lihat Optimistic Locking dalam dokumentasi Iceberg.

- Gunakan kompresi Zstandard (ZSTD).

  Codec kompresi default dari Iceberg adalah gzip, yang dapat dimodifikasi dengan menggunakan properti tabel. `write.<file_type>.compression-codec` Athena sudah menggunakan ZSTD sebagai codec kompresi default untuk tabel Iceberg.

  Secara umum, kami merekomendasikan penggunaan codec kompresi ZSTD karena mencapai keseimbangan antara GZIP dan Snappy, dan menawarkan read/write kinerja yang baik tanpa mengorbankan rasio kompresi. Selain itu, tingkat kompresi dapat disesuaikan dengan kebutuhan Anda. Untuk informasi lebih lanjut, lihat tingkat kompresi ZSTD di Athena dalam dokumentasi Athena.

  Snappy mungkin memberikan kinerja baca dan tulis keseluruhan terbaik tetapi memiliki rasio kompresi yang lebih rendah daripada GZIP dan ZSTD. Jika Anda memprioritaskan kinerja—bahkan jika itu berarti menyimpan volume data yang lebih besar di Amazon S3—Snappy mungkin merupakan pilihan yang optimal.

Mengoptimalkan kinerja baca

Bagian ini membahas properti tabel yang dapat Anda atur untuk mengoptimalkan kinerja baca, terlepas dari mesin.

Partitioning

Seperti halnya tabel Hive, Iceberg menggunakan partisi sebagai lapisan utama pengindeksan untuk menghindari membaca file metadata dan file data yang tidak perlu. Statistik kolom juga dipertimbangkan sebagai lapisan sekunder pengindeksan untuk lebih meningkatkan perencanaan kueri, yang mengarah ke waktu eksekusi keseluruhan yang lebih baik.

Partisi data Anda

Untuk mengurangi jumlah data yang dipindai saat menanyakan tabel Iceberg, pilih strategi partisi seimbang yang selaras dengan pola baca yang Anda harapkan:

- Identifikasi kolom yang sering digunakan dalam kueri. Ini adalah kandidat partisi yang ideal. Misalnya, jika Anda biasanya meminta data dari hari tertentu, contoh alami dari kolom partisi akan menjadi kolom tanggal.
- Pilih kolom partisi kardinalitas rendah untuk menghindari pembuatan partisi dalam jumlah berlebihan. Terlalu banyak partisi dapat meningkatkan jumlah file dalam tabel, yang dapat berdampak negatif pada kinerja kueri. Sebagai aturan praktis, “terlalu banyak partisi” dapat didefinisikan sebagai skenario di mana ukuran data di sebagian besar partisi kurang dari 2-5 kali nilai yang ditetapkan oleh. `target-file-size-bytes`

> **Note**
>
> Jika Anda biasanya melakukan kueri dengan menggunakan filter pada kolom kardinalitas tinggi (misalnya, `id` kolom yang dapat memiliki ribuan nilai), gunakan fitur partisi tersembunyi Iceberg dengan transformasi bucket, seperti yang dijelaskan di bagian berikutnya.

Gunakan partisi tersembunyi

Jika kueri Anda biasanya memfilter turunan kolom tabel, gunakan partisi tersembunyi alih-alih secara eksplisit membuat kolom baru untuk berfungsi sebagai partisi. Untuk informasi selengkapnya tentang fitur ini, lihat dokumentasi Iceberg.

Misalnya, dalam kumpulan data yang memiliki kolom stempel waktu (misalnya,`2023-01-01 09:00:00`), alih-alih membuat kolom baru dengan tanggal yang diuraikan (misalnya,`2023-01-01`), gunakan transformasi partisi untuk mengekstrak bagian tanggal dari stempel waktu dan membuat partisi ini dengan cepat.

Kasus penggunaan yang paling umum untuk partisi tersembunyi adalah:

- Partisi pada tanggal atau waktu, ketika data memiliki kolom stempel waktu. Iceberg menawarkan beberapa transformasi untuk mengekstrak bagian tanggal atau waktu dari stempel waktu.
- Partisi pada fungsi hash kolom, ketika kolom partisi memiliki kardinalitas tinggi dan akan menghasilkan terlalu banyak partisi. Bucket transform Iceberg mengelompokkan beberapa nilai partisi menjadi lebih sedikit, partisi tersembunyi (bucket) dengan menggunakan fungsi hash pada kolom partisi.

Lihat transformasi partisi dalam dokumentasi Iceberg untuk ikhtisar semua transformasi partisi yang tersedia.

Kolom yang digunakan untuk partisi tersembunyi dapat menjadi bagian dari predikat kueri melalui penggunaan fungsi SQL biasa seperti dan. `year()` `month()` Predikat juga dapat dikombinasikan dengan operator seperti `BETWEEN` dan`AND`.

> **Note**
>
> Iceberg tidak dapat melakukan pemangkasan partisi untuk fungsi yang menghasilkan tipe data yang berbeda; misalnya,. `substring(event_time, 1, 10) = '2022-01-01'`

Gunakan evolusi partisi

Gunakan evolusi partisi Iceberg ketika strategi partisi yang ada tidak optimal. Misalnya, jika Anda memilih partisi per jam yang ternyata terlalu kecil (masing-masing hanya beberapa megabita), pertimbangkan untuk beralih ke partisi harian atau bulanan.

Anda dapat menggunakan pendekatan ini ketika strategi partisi terbaik untuk tabel awalnya tidak jelas, dan Anda ingin memperbaiki strategi partisi Anda saat Anda mendapatkan lebih banyak wawasan. Penggunaan lain yang efektif dari evolusi partisi adalah ketika volume data berubah dan strategi partisi saat ini menjadi kurang efektif dari waktu ke waktu.

Untuk petunjuk tentang cara mengembangkan partisi, lihat ALTER TABLE ekstensi SQL dalam dokumentasi Iceberg.

Ukuran file tuning

Mengoptimalkan kinerja kueri melibatkan meminimalkan jumlah file kecil di tabel Anda. Untuk kinerja kueri yang baik, kami biasanya merekomendasikan untuk menyimpan file Parket dan ORC lebih besar dari 100 MB.

Ukuran file juga memengaruhi perencanaan kueri untuk tabel Iceberg. Karena jumlah file dalam tabel meningkat, begitu juga ukuran file metadata. File metadata yang lebih besar dapat menghasilkan perencanaan kueri yang lebih lambat. Oleh karena itu, ketika ukuran tabel bertambah, tingkatkan ukuran file untuk mengurangi ekspansi eksponensial metadata.

Gunakan praktik terbaik berikut untuk membuat file berukuran benar di tabel Iceberg.

Tetapkan file target dan ukuran grup baris

Iceberg menawarkan parameter konfigurasi kunci berikut untuk menyetel tata letak file data. Kami menyarankan Anda menggunakan parameter ini untuk mengatur ukuran file target dan grup baris atau ukuran serangan.

Parameter	Nilai default	Komentar
`write.target-file-size-bytes`	512 MB	Parameter ini menentukan ukuran file maksimum yang akan dibuat Iceberg. Namun, file tertentu mungkin ditulis dengan ukuran yang lebih kecil dari batas ini.
`write.parquet.row-group-size-bytes`	128 MB	Baik Parket dan ORC menyimpan data dalam potongan sehingga mesin dapat menghindari membaca seluruh file untuk beberapa operasi.
`write.orc.stripe-size-bytes`	64 MB	

Parameter	Nilai default	Komentar
`write.distribution-mode`	Tidak ada, untuk Iceberg versi 1.1 dan lebih rendah Hash, dimulai dengan Iceberg versi 1.2	Iceberg meminta Spark untuk mengurutkan data di antara tugasnya sebelum menulis ke penyimpanan.

- Berdasarkan ukuran tabel yang Anda harapkan, ikuti panduan umum ini:
  - Tabel kecil (hingga beberapa gigabyte) — Kurangi ukuran file target menjadi 128 MB. Juga kurangi grup baris atau ukuran garis (misalnya, menjadi 8 atau 16 MB).
  - Tabel sedang hingga besar (dari beberapa gigabyte hingga ratusan gigabyte) — Nilai default adalah titik awal yang baik untuk tabel ini. Jika kueri Anda sangat selektif, sesuaikan grup baris atau ukuran garis (misalnya, menjadi 16 MB).
  - Tabel yang sangat besar (ratusan gigabyte atau terabyte) — Tingkatkan ukuran file target menjadi 1024 MB atau lebih, dan pertimbangkan untuk meningkatkan grup baris atau ukuran garis jika kueri Anda biasanya menarik kumpulan data yang besar.

- Untuk memastikan bahwa aplikasi Spark yang menulis ke tabel Iceberg membuat file berukuran tepat, atur `write.distribution-mode` properti ke salah satu atau. `hash` `range` Untuk penjelasan rinci tentang perbedaan antara mode ini, lihat Menulis Mode Distribusi dalam dokumentasi Gunung Es.

Ini adalah pedoman umum. Kami menyarankan Anda menjalankan pengujian untuk mengidentifikasi nilai yang paling sesuai untuk tabel dan beban kerja spesifik Anda.

Jalankan pemadatan reguler

Konfigurasi dalam tabel sebelumnya menetapkan ukuran file maksimum yang dapat dibuat oleh tugas tulis, tetapi tidak menjamin bahwa file akan memiliki ukuran itu. Untuk memastikan ukuran file yang tepat, jalankan pemadatan secara teratur untuk menggabungkan file kecil menjadi file yang lebih besar. Untuk panduan terperinci tentang menjalankan pemadatan, lihat Pemadatan gunung es nanti di panduan ini.

Optimalkan statistik kolom

Iceberg menggunakan statistik kolom untuk melakukan pemangkasan file, yang meningkatkan kinerja kueri dengan mengurangi jumlah data yang dipindai oleh kueri. Untuk mendapatkan keuntungan dari statistik kolom, pastikan bahwa Iceberg mengumpulkan statistik untuk semua kolom yang sering digunakan dalam filter kueri.

Secara default, Iceberg mengumpulkan statistik hanya untuk 100 kolom pertama di setiap tabel, seperti yang didefinisikan oleh properti tabel. `write.metadata.metrics.max-inferred-column-defaults` Jika tabel Anda memiliki lebih dari 100 kolom dan kueri Anda sering merujuk kolom di luar 100 kolom pertama (misalnya, Anda mungkin memiliki kueri yang memfilter pada kolom 132), pastikan bahwa Iceberg mengumpulkan statistik pada kolom tersebut. Ada dua opsi untuk mencapai ini:

- Saat Anda membuat tabel Iceberg, susun ulang kolom sehingga kolom yang Anda butuhkan untuk kueri termasuk dalam rentang kolom yang ditetapkan oleh `write.metadata.metrics.max-inferred-column-defaults` (default adalah 100).

  Catatan: Jika Anda tidak memerlukan statistik pada 100 kolom, Anda dapat menyesuaikan `write.metadata.metrics.max-inferred-column-defaults` konfigurasi ke nilai yang diinginkan (misalnya, 20) dan menyusun ulang kolom sehingga kolom yang perlu Anda baca dan tulis kueri termasuk dalam 20 kolom pertama di sisi kiri kumpulan data.
- Jika Anda hanya menggunakan beberapa kolom dalam filter kueri, Anda dapat menonaktifkan keseluruhan properti untuk koleksi metrik dan secara selektif memilih kolom individual untuk mengumpulkan statistik, seperti yang ditunjukkan dalam contoh ini:

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

Catatan: Statistik kolom paling efektif ketika data diurutkan pada kolom tersebut. Untuk informasi selengkapnya, lihat bagian Mengatur urutan pengurutan nanti dalam panduan ini.

Pilih strategi pembaruan yang tepat

Gunakan copy-on-write strategi untuk mengoptimalkan kinerja baca, ketika operasi penulisan yang lebih lambat dapat diterima untuk kasus penggunaan Anda. Ini adalah strategi default yang digunakan oleh Iceberg.

Copy-on-write menghasilkan kinerja baca yang lebih baik, karena file langsung ditulis ke penyimpanan dengan cara yang dioptimalkan untuk dibaca. Namun, dibandingkan dengan merge-on-read, setiap operasi penulisan membutuhkan waktu lebih lama dan mengkonsumsi lebih banyak sumber daya komputasi. Ini menyajikan trade-off klasik antara latensi baca dan tulis. Biasanya, copy-on-write sangat ideal untuk kasus penggunaan di mana sebagian besar pembaruan ditempatkan di partisi tabel yang sama (misalnya, untuk pemuatan batch harian).

Copy-on-write konfigurasi (`write.update.mode`,`write.delete.mode`, dan`write.merge.mode`) dapat diatur pada tingkat tabel atau secara independen di sisi aplikasi.

Gunakan kompresi ZSTD

Anda dapat memodifikasi codec kompresi yang digunakan oleh Iceberg dengan menggunakan properti tabel. `write.<file_type>.compression-codec` Kami menyarankan Anda menggunakan codec kompresi ZSTD untuk meningkatkan kinerja keseluruhan pada tabel.

Secara default, Iceberg versi 1.3 dan sebelumnya menggunakan kompresi GZIP, yang memberikan read/write kinerja lebih lambat dibandingkan dengan ZSTD.

Catatan: Beberapa mesin mungkin menggunakan nilai default yang berbeda. Ini adalah kasus untuk tabel Iceberg yang dibuat dengan Athena atau Amazon EMR versi 7.x.

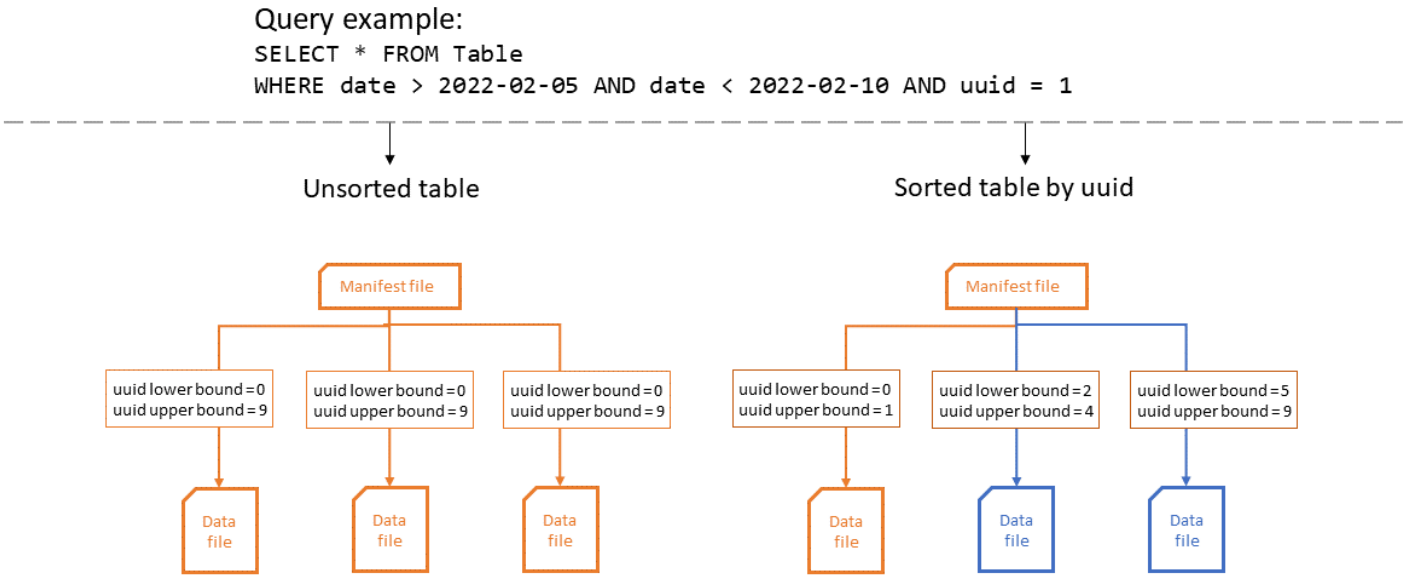
Mengatur urutan sortir

Untuk meningkatkan kinerja baca pada tabel Iceberg, sebaiknya Anda mengurutkan tabel berdasarkan satu atau beberapa kolom yang sering digunakan dalam filter kueri. Penyortiran, dikombinasikan dengan statistik kolom Iceberg, dapat membuat pemangkasan file secara signifikan lebih efisien, yang menghasilkan operasi pembacaan yang lebih cepat. Penyortiran juga mengurangi jumlah permintaan Amazon S3 untuk kueri yang menggunakan kolom pengurutan dalam filter kueri.

Anda dapat mengatur urutan hierarkis di tingkat tabel dengan menjalankan pernyataan data definition language (DDL) dengan Spark. Untuk opsi yang tersedia, lihat dokumentasi Iceberg. Setelah Anda mengatur urutan pengurutan, penulis akan menerapkan penyortiran ini ke operasi penulisan data berikutnya di tabel Iceberg.

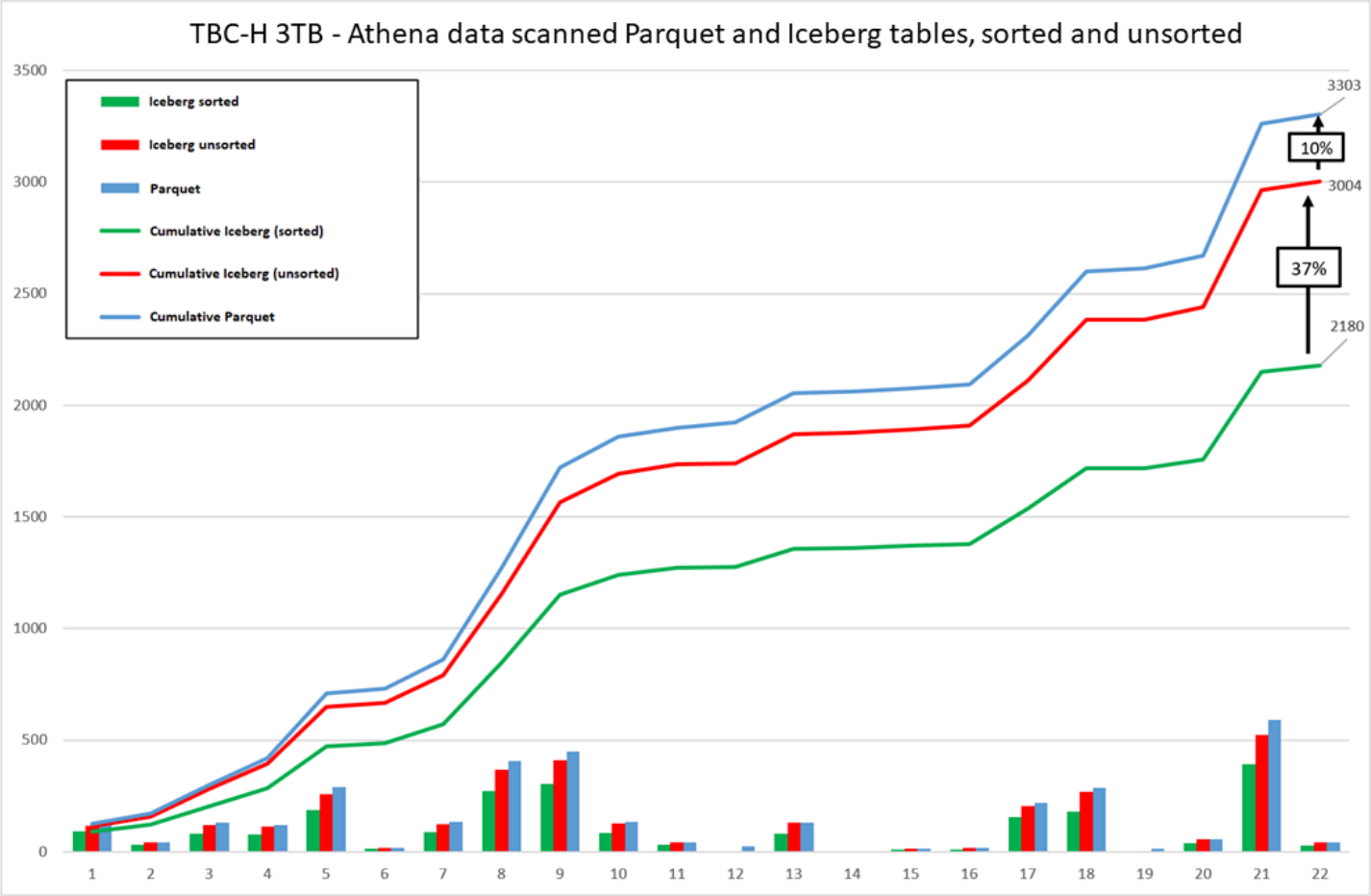
Misalnya, dalam tabel yang dipartisi berdasarkan date (`yyyy-mm-dd`) di mana sebagian besar kueri difilteruuid, Anda dapat menggunakan opsi DDL `Write Distributed By Partition Locally 0rdered` untuk memastikan bahwa Spark menulis file dengan rentang yang tidak tumpang tindih.

Diagram berikut menggambarkan bagaimana efisiensi statistik kolom meningkat ketika tabel diurutkan. Dalam contoh, tabel yang diurutkan hanya perlu membuka satu file, dan manfaat maksimal dari partisi dan file Iceberg. Dalam tabel yang tidak disortir, apa pun `uuid` berpotensi ada di file data apa pun, sehingga kueri harus membuka semua file data.

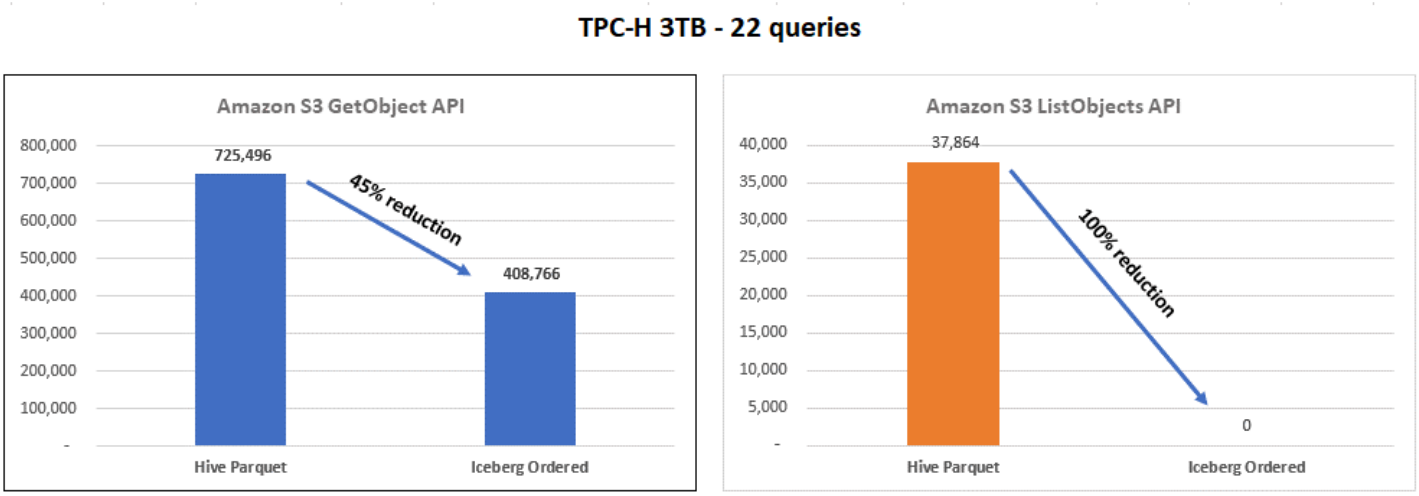


Mengubah urutan pengurutan tidak memengaruhi file data yang ada. Anda dapat menggunakan pemadatan Iceberg untuk menerapkan urutan pengurutan pada itu.

Menggunakan tabel yang diurutkan Iceberg dapat mengurangi biaya untuk beban kerja Anda, seperti yang diilustrasikan dalam grafik berikut.



Grafik ini merangkum hasil menjalankan benchmark TPC-H untuk tabel Hive (Parquet) dibandingkan dengan tabel yang diurutkan Iceberg. Namun, hasilnya mungkin berbeda untuk kumpulan data atau beban kerja lainnya.



Mengoptimalkan kinerja tulis

Bagian ini membahas properti tabel yang dapat Anda atur untuk mengoptimalkan kinerja tulis pada tabel Iceberg, terlepas dari mesin.

Mengatur modus distribusi tabel

Iceberg menawarkan beberapa mode distribusi tulis yang menentukan bagaimana data tulis didistribusikan di seluruh tugas Spark. Untuk gambaran umum tentang mode yang tersedia, lihat Menulis Mode Distribusi dalam dokumentasi Gunung Es.

Untuk kasus penggunaan yang memprioritaskan kecepatan tulis, terutama dalam beban kerja streaming, atur ke. `write.distribution-mode` `none` Ini memastikan bahwa Iceberg tidak meminta pengocokan Spark tambahan dan data tersebut ditulis saat tersedia dalam tugas Spark. Mode ini sangat cocok untuk aplikasi Streaming Terstruktur Spark.

> **Note**
>
> Mengatur mode distribusi tulis `none` cenderung menghasilkan banyak file kecil, yang menurunkan kinerja baca. Kami merekomendasikan pemadatan reguler untuk mengkonsolidasikan file-file kecil ini ke dalam file berukuran benar untuk kinerja kueri.

Pilih strategi pembaruan yang tepat

Gunakan merge-on-read strategi untuk mengoptimalkan kinerja penulisan, ketika operasi baca yang lebih lambat pada data terbaru dapat diterima untuk kasus penggunaan Anda.

Saat Anda menggunakan merge-on-read, Iceberg menulis pembaruan dan menghapus ke penyimpanan sebagai file kecil yang terpisah. Saat tabel dibaca, pembaca harus menggabungkan perubahan ini dengan file dasar untuk mengembalikan tampilan data terbaru. Ini menghasilkan penalti kinerja untuk operasi baca, tetapi mempercepat penulisan pembaruan dan penghapusan. Biasanya, merge-on-read sangat ideal untuk streaming beban kerja dengan pembaruan atau pekerjaan dengan beberapa pembaruan yang tersebar di banyak partisi tabel.

Anda dapat mengatur merge-on-read konfigurasi (`write.update.mode`,`write.delete.mode`, dan`write.merge.mode`) di tingkat tabel atau secara independen di sisi aplikasi.

Menggunakan merge-on-read membutuhkan menjalankan pemadatan reguler untuk mencegah kinerja baca menurun dari waktu ke waktu. Pemadatan merekonsiliasi pembaruan dan penghapusan

dengan file data yang ada untuk membuat kumpulan file data baru, sehingga menghilangkan penalti kinerja yang terjadi di sisi baca. Secara default, pemadatan Iceberg tidak menggabungkan file hapus kecuali Anda mengubah default `delete-file-threshold` properti ke nilai yang lebih kecil (lihat dokumentasi Iceberg). Untuk mempelajari lebih lanjut tentang pemadatan, lihat bagian Pemadatan gunung es nanti di panduan ini.

Pilih format file yang tepat

Iceberg mendukung penulisan data dalam format Parket, ORC, dan Avro. Parket adalah format default. Parket dan ORC adalah format kolom yang menawarkan kinerja baca yang unggul tetapi umumnya lebih lambat untuk ditulis. Ini mewakili trade-off khas antara kinerja baca dan tulis.

Jika kecepatan tulis penting untuk kasus penggunaan Anda, seperti dalam beban kerja streaming, pertimbangkan untuk menulis dalam format Avro dengan menyetel `write-format` ke `Avro` opsi penulis. Karena Avro adalah format berbasis baris, Avro memberikan waktu tulis yang lebih cepat dengan mengorbankan kinerja baca yang lebih lambat.

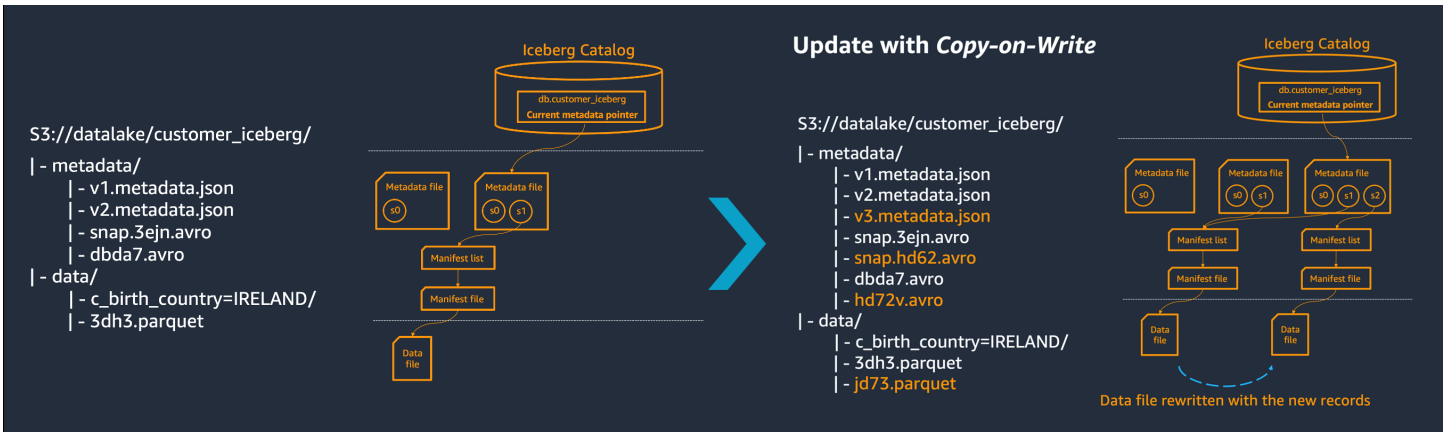
Untuk meningkatkan kinerja baca, jalankan pemadatan reguler untuk menggabungkan dan mengubah file Avro kecil menjadi file Parket yang lebih besar. Hasil dari proses pemadatan diatur oleh pengaturan `write.format.default` tabel. Format default untuk Iceberg adalah Parquet, jadi jika Anda menulis di Avro dan kemudian menjalankan pemadatan, Iceberg akan mengubah file Avro menjadi file Parket. Inilah contohnya:

```
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()
```

Mengoptimalkan penyimpanan

Memperbarui atau menghapus data dalam tabel Gunung Es meningkatkan jumlah salinan data Anda, seperti yang diilustrasikan dalam diagram berikut. Hal yang sama berlaku untuk menjalankan pemadatan: Ini meningkatkan jumlah salinan data di Amazon S3. Itu karena Iceberg memperlakukan file yang mendasari semua tabel sebagai tidak dapat diubah.



Ikuti praktik terbaik di bagian ini untuk mengelola biaya penyimpanan.

Aktifkan S3 Intelligent-Tiering

Gunakan kelas penyimpanan Amazon S3 Intelligent-Tiering untuk memindahkan data secara otomatis ke tingkat akses yang paling hemat biaya saat pola akses berubah. Opsi ini tidak memiliki overhead operasional atau berdampak pada kinerja.

Catatan: Jangan gunakan tingkatan opsional (seperti Akses Arsip dan Akses Arsip Dalam) di S3 Intelligent-Tiering with Iceberg tables. Untuk mengarsipkan data, lihat pedoman di bagian selanjutnya.

Anda juga dapat menggunakan aturan Siklus Hidup Amazon S3 untuk menetapkan aturan sendiri untuk memindahkan objek ke kelas penyimpanan Amazon S3 lainnya, seperti IA Standar S3 atau IA Zona Satu S3 (lihat Transisi yang didukung dan batasan terkait dalam dokumentasi Amazon S3).

Arsipkan atau hapus snapshot bersejarah

Untuk setiap transaksi yang dilakukan (sisipkan, perbarui, gabungkan, pemadatan) ke tabel Iceberg, versi baru atau snapshot tabel dibuat. Seiring waktu, jumlah versi dan jumlah file metadata di Amazon S3 terakumulasi.

Menyimpan snapshot tabel diperlukan untuk fitur seperti isolasi snapshot, rollback tabel, dan kueri perjalanan waktu. Namun, biaya penyimpanan tumbuh dengan jumlah versi yang Anda pertahankan.

Tabel berikut menjelaskan pola desain yang dapat Anda terapkan untuk mengelola biaya berdasarkan persyaratan retensi data Anda.

Pola desain	Solusi	Kasus penggunaan
Hapus snapshot lama	• Gunakan pernyataan VACUUM di Athena untuk menghapus snapshot lama. Operasi ini tidak dikenakan biaya komputasi apa pun. • Atau, Anda dapat menggunakan Spark di Amazon EMR AWS Glue atau untuk menghapus snapshot.Untuk informasi selengkapnya, lihat expire_snapshots dalam dokumentasi Iceberg.	Pendekatan ini menghapus snapshot yang tidak lagi diperlukan untuk mengurang i biaya penyimpanan. Anda dapat mengonfigurasi berapa banyak snapshot yang harus disimpan atau untuk berapa lama, berdasarkan persyarat an retensi data Anda. Opsi ini melakukan penghapus an snapshot dengan keras. Anda tidak dapat memutar kembali atau melakukan perjalanan waktu ke snapshot yang kedaluwarsa.
Tetapkan kebijakan retensi untuk snapshot tertentu	1. Gunakan tag untuk menandai snapshot tertentu dan menentukan kebijakan penyimpanan di Iceberg. Untuk informasi selengkap nya, lihat Tag Sejarah	Pola ini berguna untuk kepatuhan dengan persyarat an bisnis atau hukum yang mengharuskan Anda untuk menunjukkan keadaan tabel pada titik tertentu di masa lalu. Dengan menempatk

Pola desain	Solusi	Kasus penggunaan
	dalam dokumentasi Gunung Es. Misalnya, Anda dapat menyimpan satu snapshot per bulan selama satu tahun dengan menggunak an pernyataan SQL berikut di Spark di Amazon EMR: `ALTER TABLE glue_cata log.db.table CREATE TAG 'EOM-01' AS OF VERSION 30 RETAIN 365 DAYS` 2. Gunakan Spark di Amazon EMR AWS Glue atau untuk menghapus sisa snapshot perantara yang tidak ditandai.	an kebijakan penyimpanan pada snapshot yang diberi tag tertentu, Anda dapat menghapus snapshot lain (tidak ditandai) yang dibuat. Dengan cara ini, Anda dapat memenuhi persyaratan retensi data tanpa mempertahankan setiap snapshot yang dibuat.

Pola desain	Solusi	Kasus penggunaan
Arsipkan foto lama	1. Gunakan tag Amazon S3 untuk menandai objek dengan Spark. (Tag Amazon S3 berbeda dari tag Iceberg; untuk informasi selengkapnya, lihat dokumentasi Iceberg.) Contoh: `spark.sql.catalog.my_catalog.s3.delete-enabled=false and \ spark.sql.catalog.my_catalog.s3.delete.tags.my_key=to_archive` 2. Gunakan Spark di Amazon EMR AWS Glue atau untuk menghapus snapshot. Saat Anda menggunakan pengaturan dalam contoh, prosedur ini menandai objek dan melepaskannya dari metadata tabel Iceberg alih-alih menghapusnya dari Amazon S3. 3. Gunakan aturan Siklus Hidup S3 untuk mentransi sikan objek yang diberi tag `to_archive` ke salah satu kelas penyimpanan S3 Glacier. 4. Untuk menanyakan data yang diarsipkan:	Pola ini memungkinkan Anda menyimpan semua versi tabel dan snapshot dengan biaya lebih rendah. Anda tidak dapat melakukan perjalanan waktu atau memutar kembali ke snapshot yang diarsipkan tanpa terlebih dahulu memulihkan versi tersebut sebagai tabel baru. Ini biasanya dapat diterima untuk tujuan audit. Anda dapat menggabun gkan pendekatan ini dengan pola desain sebelumnya, menyetel kebijakan retensi untuk snapshot tertentu.

Pola desain	Solusi	Kasus penggunaan
	• Kembalikan objek yang diarsipkan (langkah ini tidak diperlukan jika objek dialihkan ke kelas penyimpanan Amazon Glacier Instant Retrieval). • Gunakan prosedur register_table di Iceberg untuk mendaftarkan snapshot sebagai tabel dalam katalog. Untuk petunjuk terperinci, lihat posting AWS blog Meningkatkan efisiensi operasional tabel Apache Iceberg yang dibangun di danau data Amazon S3.	

Hapus file yatim piatu

Dalam situasi tertentu, aplikasi Iceberg dapat gagal sebelum Anda melakukan transaksi Anda. Ini meninggalkan file data di Amazon S3. Karena tidak ada komit, file-file ini tidak akan dikaitkan dengan tabel apa pun, jadi Anda mungkin harus membersihkannya secara asinkron.

Untuk menangani penghapusan ini, Anda dapat menggunakan pernyataan VACUUM di Amazon Athena. Pernyataan ini menghapus snapshot dan juga menghapus file yatim piatu. Ini sangat hemat biaya, karena Athena tidak mengenakan biaya untuk biaya komputasi operasi ini. Selain itu, Anda tidak perlu menjadwalkan operasi tambahan apa pun saat menggunakan `VACUUM` pernyataan tersebut.

Atau, Anda dapat menggunakan Spark di Amazon EMR AWS Glue atau untuk menjalankan `remove_orphan_files` prosedur. Operasi ini memiliki biaya komputasi dan harus dijadwalkan secara independen. Untuk informasi lebih lanjut, lihat dokumentasi Iceberg.

Mempertahankan tabel dengan menggunakan pemadatan

Iceberg mencakup fitur yang memungkinkan Anda untuk melakukan operasi pemeliharaan tabel setelah menulis data ke tabel. Beberapa operasi pemeliharaan berfokus pada perampingan file metadata, sementara yang lain meningkatkan bagaimana data dikelompokkan dalam file sehingga mesin kueri dapat secara efisien menemukan informasi yang diperlukan untuk menanggapi permintaan pengguna. Bagian ini berfokus pada pengoptimalan terkait pemadatan.

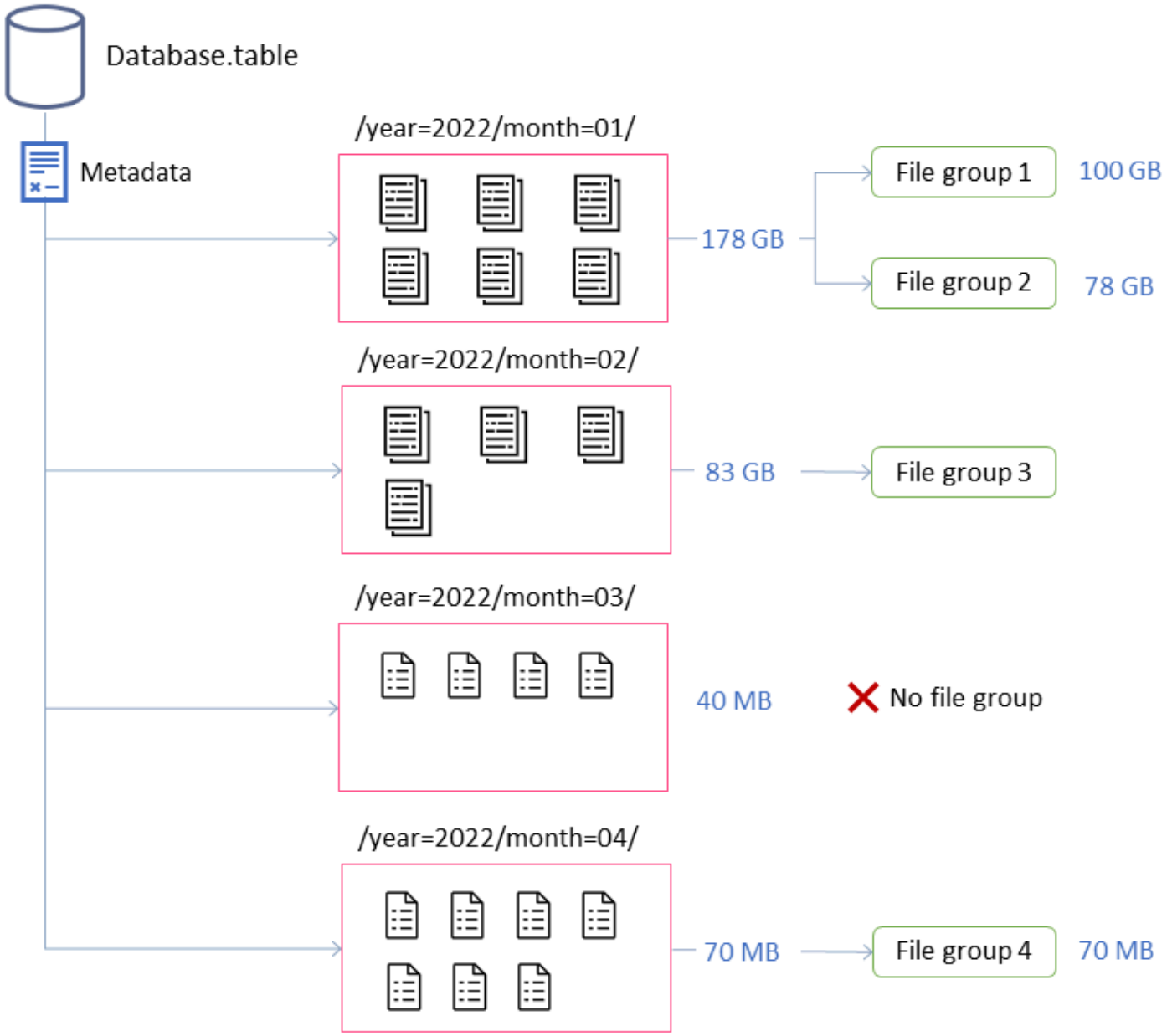
Pemadatan gunung es

Di Iceberg, Anda dapat menggunakan pemadatan untuk melakukan empat tugas:

- Menggabungkan file kecil menjadi file yang lebih besar yang umumnya berukuran lebih dari 100 MB. Teknik ini dikenal sebagai bin packing.
- Menggabungkan menghapus file dengan file data. Hapus file dihasilkan oleh pembaruan atau penghapusan yang menggunakan merge-on-read pendekatan.
- (Re) menyortir data sesuai dengan pola kueri. Data dapat ditulis tanpa urutan apapun atau dengan urutan sortir yang cocok untuk menulis dan update.
- Mengelompokkan data dengan menggunakan kurva pengisian ruang untuk mengoptimalkan pola kueri yang berbeda, terutama penyortiran urutan-z.

Pada AWS, Anda dapat menjalankan operasi pemadatan dan pemeliharaan tabel untuk Iceberg melalui Amazon Athena atau dengan menggunakan Spark di Amazon EMR atau. AWS Glue

Saat Anda menjalankan pemadatan dengan menggunakan prosedur rewrite_data_files, Anda dapat menyesuaikan beberapa kenop untuk mengontrol perilaku pemadatan. Diagram berikut menunjukkan perilaku default pengepakan bin. Memahami pemadatan kemasan bin adalah kunci untuk memahami penyortiran hierarkis dan implementasi penyortiran urutan-Z, karena mereka adalah ekstensi dari antarmuka pengemasan bin dan beroperasi dengan cara yang sama. Perbedaan utama adalah langkah tambahan yang diperlukan untuk menyortir atau mengelompokkan data.



Dalam contoh ini, tabel Iceberg terdiri dari empat partisi. Setiap partisi memiliki ukuran dan jumlah file yang berbeda. Jika Anda memulai aplikasi Spark untuk menjalankan pemadatan, aplikasi membuat total empat grup file untuk diproses. Grup file adalah abstraksi Gunung Es yang mewakili kumpulan file yang akan diproses oleh satu pekerjaan Spark. Artinya, aplikasi Spark yang menjalankan pemadatan akan menciptakan empat pekerjaan Spark untuk memproses data.

Menyetel perilaku pemadatan

Properti kunci berikut mengontrol bagaimana file data dipilih untuk pemadatan:

- MAX_FILE_GROUP_SIZE_BYTES menetapkan batas data untuk satu grup file (Spark job) pada 100 GB secara default. Properti ini sangat penting untuk tabel tanpa partisi atau tabel dengan partisi yang menjangkau ratusan gigabyte. Dengan menetapkan batas ini, Anda dapat memecah operasi untuk merencanakan pekerjaan dan membuat kemajuan sekaligus mencegah kehabisan sumber daya di cluster.

  Catatan: Setiap grup file diurutkan secara terpisah. Oleh karena itu, jika Anda ingin melakukan pengurutan tingkat partisi, Anda harus menyesuaikan batas ini agar sesuai dengan ukuran partisi.
- MIN_FILE_SIZE_BYTES atau MIN_FILE_SIZE_DEFAULT_RATIO default ke 75 persen dari ukuran file target yang ditetapkan pada tingkat tabel. Misalnya, jika tabel memiliki ukuran target 512 MB, file apa pun yang lebih kecil dari 384 MB disertakan dalam kumpulan file yang akan dipadatkan.
- MAX_FILE_SIZE_BYTES atau MAX_FILE_SIZE_DEFAULT_RATIO default 180 persen dari ukuran file target. Seperti dua properti yang mengatur ukuran file minimum, properti ini digunakan untuk mengidentifikasi file kandidat untuk pekerjaan pemadatan.
- MIN_INPUT_FILES menentukan jumlah minimum file yang akan dipadatkan jika ukuran partisi tabel lebih kecil dari ukuran file target. Nilai properti ini digunakan untuk menentukan apakah layak untuk memadatkan file berdasarkan jumlah file (default ke 5).
- DELETE_FILE_THRESHOLD menentukan jumlah minimum operasi penghapusan untuk file sebelum disertakan dalam pemadatan. Kecuali Anda menentukan sebaliknya, pemadatan tidak menggabungkan file hapus dengan file data. Untuk mengaktifkan fungsi ini, Anda harus menetapkan nilai ambang dengan menggunakan properti ini. Ambang batas ini khusus untuk file data individual, jadi jika Anda mengaturnya ke 3, file data akan ditulis ulang hanya jika ada tiga atau lebih file hapus yang mereferensikannya.

Properti ini memberikan wawasan tentang pembentukan kelompok file dalam diagram sebelumnya.

Misalnya, partisi berlabel `month=01` mencakup dua grup file karena melebihi batasan ukuran maksimum 100 GB. Sebaliknya, `month=02` partisi berisi satu grup file karena di bawah 100 GB. `month=03`Partisi tidak memenuhi persyaratan file input minimum default dari lima file. Akibatnya, itu tidak akan dipadatkan. Terakhir, meskipun `month=04` partisi tidak berisi data yang cukup untuk membentuk satu file dengan ukuran yang diinginkan, file akan dipadatkan karena partisi mencakup lebih dari lima file kecil.

Anda dapat mengatur parameter ini untuk Spark yang berjalan di Amazon AWS Glue EMR atau. Untuk Amazon Athena, Anda dapat mengelola properti serupa dengan menggunakan properti tabel yang dimulai dengan awalan`optimize_`).

Menjalankan pemadatan dengan Spark di Amazon EMR atau AWS Glue

Bagian ini menjelaskan cara mengukur cluster Spark dengan benar untuk menjalankan utilitas pemadatan Iceberg. Contoh berikut menggunakan Amazon EMR Tanpa Server, tetapi Anda dapat menggunakan metodologi yang sama di Amazon EMR di EC2 atau EKS, atau di. AWS Glue

Anda dapat memanfaatkan korelasi antara grup file dan pekerjaan Spark untuk merencanakan sumber daya cluster. Untuk memproses grup file secara berurutan, dengan mempertimbangkan ukuran maksimum 100 GB per grup file, Anda dapat mengatur properti Spark berikut:

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

Jika Anda ingin mempercepat pemadatan, Anda dapat menskalakan secara horizontal dengan meningkatkan jumlah grup file yang dipadatkan secara paralel. Anda juga dapat menskalakan EMR Amazon dengan menggunakan penskalaan manual atau dinamis.

- Penskalaan secara manual (misalnya, dengan faktor 4)
  - `MAX_CONCURRENT_FILE_GROUP_REWRITES`= `4` (faktor kami)
  - `spark.executor.instances`= `5` (nilai yang digunakan dalam contoh) x `4` (faktor kami) = `20`
  - `spark.dynamicAllocation.enabled` = `FALSE`
- Penskalaan dinamis
  - `spark.dynamicAllocation.enabled`= `TRUE` (default, tidak ada tindakan yang diperlukan)
  - MAX_CONCURRENT_FILE_GROUP_REWRITES = `N` (sejajarkan nilai ini dengan`spark.dynamicAllocation.maxExecutors`, yang 100 secara default; berdasarkan konfigurasi pelaksana dalam contoh, Anda dapat mengatur ke 20) `N`

  Ini adalah pedoman untuk membantu mengukur cluster. Namun, Anda juga harus memantau kinerja pekerjaan Spark Anda untuk menemukan pengaturan terbaik untuk beban kerja Anda.

Menjalankan pemadatan dengan Amazon Athena

Athena menawarkan implementasi utilitas pemadatan Iceberg sebagai fitur terkelola melalui pernyataan OPTIMIZE. Anda dapat menggunakan pernyataan ini untuk menjalankan pemadatan tanpa harus mengevaluasi infrastruktur.

Pernyataan ini mengelompokkan file kecil ke dalam file yang lebih besar dengan menggunakan algoritma pengemasan bin dan menggabungkan file hapus dengan file data yang ada. Untuk mengelompokkan data menggunakan pengurutan hierarkis atau pengurutan urutan z, gunakan Spark di Amazon EMR atau. AWS Glue

Anda dapat mengubah perilaku default `OPTIMIZE` pernyataan pada pembuatan tabel dengan meneruskan properti tabel dalam `CREATE TABLE` pernyataan, atau setelah pembuatan tabel dengan menggunakan `ALTER TABLE` pernyataan. Untuk nilai default, lihat dokumentasi Athena.

Rekomendasi untuk menjalankan pemadatan

Kasus penggunaan	Rekomendasi
Menjalankan pemadatan pengepakan bin berdasarkan jadwal	• Gunakan `OPTIMIZE` pernyataan di Athena jika Anda tidak tahu berapa banyak file kecil yang berisi tabel Anda. Model penetapan harga Athena didasarkan pada data yang dipindai, jadi jika tidak ada file yang akan dipadatkan, tidak ada biaya yang terkait dengan operasi ini. Untuk menghindari menemui batas waktu di tabel Athena, jalankan berdasarkan. `OPTIMIZE` per-table-partition • Gunakan Amazon EMR atau AWS Glue dengan penskalaan dinamis saat Anda mengharapkan volume besar file kecil dipadatkan.
Menjalankan pemadatan pengepakan bin berdasarkan peristiwa	• Gunakan Amazon EMR atau AWS Glue dengan penskalaan dinamis saat Anda mengharapkan volume besar file kecil dipadatkan.
Menjalankan pemadatan untuk mengurutkan data	• Gunakan Amazon EMR atau AWS Glue, karena penyortiran adalah operasi yang mahal dan mungkin perlu menumpahkan data ke disk.

Kasus penggunaan	Rekomendasi
Menjalankan pemadatan untuk mengelomp okkan data menggunakan pengurutan urutan z	• Gunakan Amazon EMR atau AWS Glue, karena penyortiran urutan z adalah operasi yang sangat mahal dan mungkin perlu menumpahkan data ke disk.
Menjalankan pemadatan pada partisi yang mungkin diperbarui oleh aplikasi lain karena data yang datang terlambat	• Gunakan Amazon EMR atau. AWS Glue Aktifkan properti Iceberg PARTIAL_P ROGRESS_ENABLED. Saat Anda menggunakan opsi ini, Iceberg membagi output pemadatan menjadi beberapa komit. Jika ada tabrakan (yaitu, jika file data diperbarui saat pemadatan sedang berjalan) , pengaturan ini mengurangi biaya percobaan ulang dengan membatasi ke komit yang menyertakan file yang terpengaruh. Jika tidak, Anda mungkin harus mengompres ulang semua file.
Menjalankan pemadatan pada partisi dingin (partisi data yang tidak lagi menerima penulisan aktif)	• Gunakan Amazon EMR atau. AWS Glue Dalam `rewrite_data_files` prosedur, tentukan `where` predikat yang mengecual ikan partisi yang ditulis secara aktif. Strategi ini mencegah konflik data antara penulis dan pekerjaan pemadatan, dan hanya menyisaka n konflik metadata yang dapat diselesaikan oleh Iceberg secara otomatis.

Menggunakan beban kerja Iceberg di Amazon S3

Bagian ini membahas properti Iceberg yang dapat Anda gunakan untuk mengoptimalkan interaksi Iceberg dengan Amazon S3.

Mencegah partisi panas (kesalahan HTTP 503)

Beberapa aplikasi data lake yang berjalan di Amazon S3 menangani jutaan atau miliaran objek dan memproses petabyte data. Hal ini dapat menyebabkan awalan yang menerima volume lalu lintas yang tinggi, yang biasanya terdeteksi melalui kesalahan HTTP 503 (layanan tidak tersedia). Untuk mencegah masalah ini, gunakan properti Iceberg berikut:

- Setel `write.distribution-mode` ke `hash` atau `range` lebih agar Iceberg menulis file besar, yang menghasilkan lebih sedikit permintaan Amazon S3. Ini adalah konfigurasi yang disukai dan harus mengatasi sebagian besar kasus.
- Jika Anda terus mengalami 503 kesalahan karena volume data yang sangat besar dalam beban kerja Anda, Anda dapat mengatur `write.object-storage.enabled` ke dalam Iceberg. `true` Ini menginstruksikan Iceberg untuk melakukan hash nama objek dan mendistribusikan beban di beberapa awalan Amazon S3 acak.

Untuk informasi selengkapnya tentang properti ini, lihat Menulis properti di dokumentasi Gunung Es.

Gunakan operasi pemeliharaan Iceberg untuk merilis data yang tidak digunakan

Untuk mengelola tabel Iceberg, Anda dapat menggunakan API inti Iceberg, klien Iceberg (seperti Spark), atau layanan terkelola seperti Amazon Athena. Untuk menghapus file lama atau yang tidak digunakan dari Amazon S3, kami sarankan Anda hanya menggunakan Iceberg APIs native untuk menghapus snapshot, menghapus file metadata lama, dan menghapusfile yatim piatu.

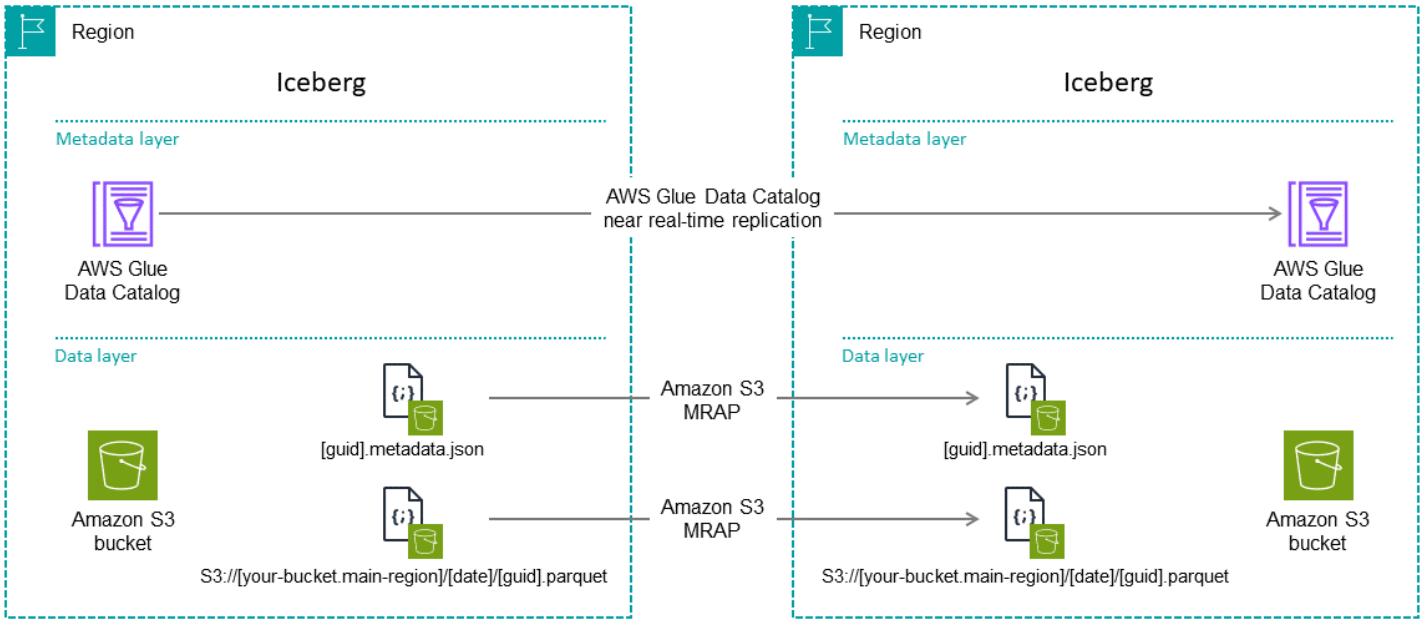
Menggunakan Amazon S3 APIs melalui Boto3, Amazon S3 SDK, atau AWS Command Line Interface (AWS CLI), atau menggunakan metode non-Iceberg lainnya untuk menimpa atau menghapus file Amazon S3 untuk tabel Iceberg menyebabkan kerusakan tabel dan kegagalan kueri.

Replikasi data di seluruh Wilayah AWS

Saat menyimpan tabel Iceberg di Amazon S3, Anda dapat menggunakan fitur bawaan di Amazon S3, seperti Replikasi Lintas Wilayah (CRR) dan Titik Akses Multi-Wilayah (MRAP), untuk mereplikasi data di beberapa. Wilayah AWS MRAP menyediakan titik akhir global untuk aplikasi untuk mengakses bucket S3 yang terletak di beberapa. Wilayah AWS Iceberg tidak mendukung jalur relatif, tetapi Anda dapat menggunakan MRAP untuk melakukan operasi Amazon S3 dengan memetakan bucket ke titik akses. MRAP juga terintegrasi secara mulus dengan proses Replikasi Lintas Wilayah Amazon S3, yang memperkenalkan jeda hingga 15 menit. Anda harus mereplikasi file data dan metadata.

> **Important**
>
> Saat ini, integrasi Iceberg dengan MRAP hanya berfungsi dengan Apache Spark. Jika Anda perlu gagal ke sekunder Wilayah AWS, Anda harus merencanakan untuk mengarahkan kueri pengguna ke lingkungan Spark SQL (seperti Amazon EMR) di Wilayah failover.

Fitur CRR dan MRAP membantu Anda membangun solusi replikasi lintas wilayah untuk tabel Iceberg, seperti yang diilustrasikan dalam diagram berikut.



Untuk menyiapkan arsitektur replikasi lintas wilayah ini:

1. Buat tabel dengan menggunakan lokasi MRAP. Ini memastikan bahwa file metadata Iceberg mengarah ke lokasi MRAP alih-alih lokasi bucket fisik.
2. Replikasi file Iceberg dengan menggunakan Amazon S3 MRAP. MRAP mendukung replikasi data dengan perjanjian tingkat layanan (SLA) selama 15 menit. Iceberg mencegah operasi baca dari memperkenalkan inkonsistensi selama replikasi.
3. Buat tabel tersedia AWS Glue Data Catalog di Wilayah sekunder. Anda dapat memilih dari dua opsi:
   - Siapkan pipeline untuk mereplikasi metadata tabel Iceberg dengan menggunakan replikasi. AWS Glue Data Catalog Utilitas ini tersedia di repositori replikasi GitHub Glue Catalog dan Lake Formation Permissions. Mekanisme berbasis peristiwa ini mereplikasi tabel di Wilayah target berdasarkan log peristiwa.

- Daftarkan tabel di Wilayah sekunder saat Anda harus gagal. Untuk opsi ini, Anda dapat menggunakan utilitas sebelumnya atau prosedur Iceberg register_table dan mengarahkannya ke file terbaru. `metadata.json`

Memantau beban kerja Apache Iceberg

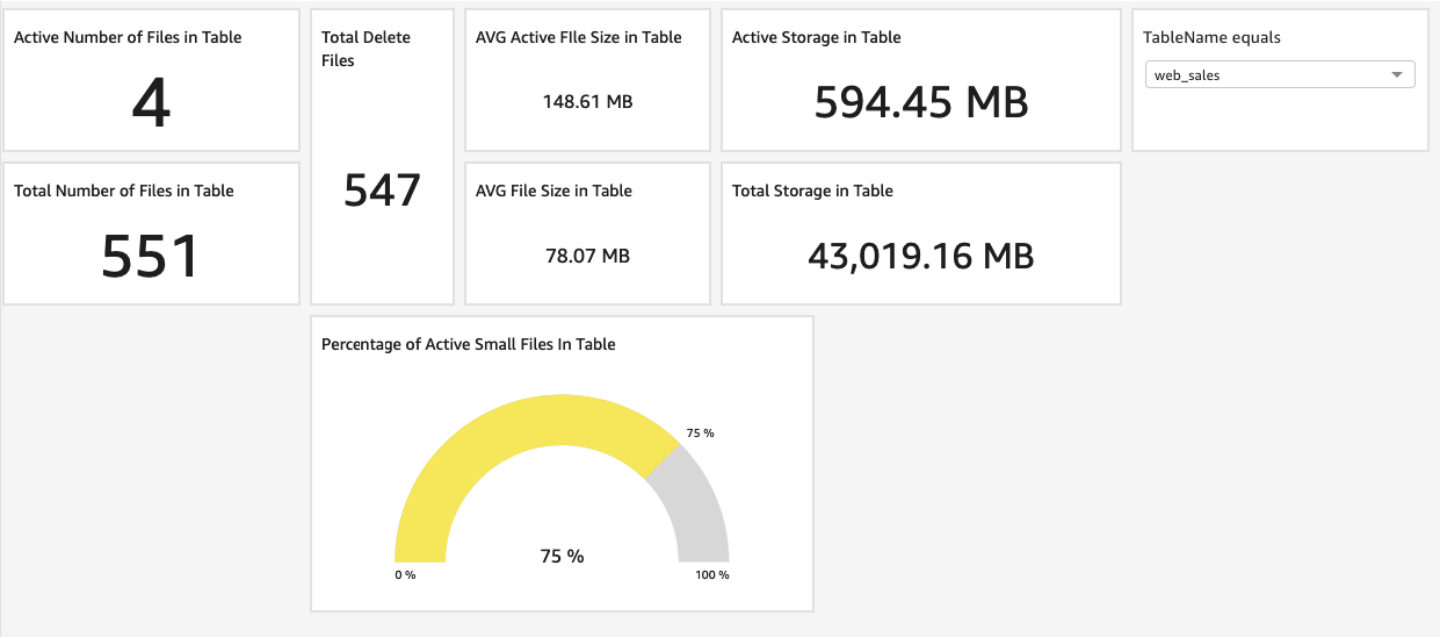
Untuk memantau beban kerja Iceberg, Anda memiliki dua opsi: menganalisis tabel metadata atau menggunakan reporter metrik. Reporter metrik diperkenalkan dalam Iceberg versi 1.2 dan hanya tersedia untuk katalog REST dan JDBC.

Jika Anda menggunakan AWS Glue Data Catalog, Anda dapat memperoleh wawasan tentang kesehatan tabel Gunung Es Anda dengan mengatur pemantauan di atas tabel metadata yang diekspos Iceberg.

Pemantauan sangat penting untuk manajemen kinerja dan pemecahan masalah. Misalnya, ketika partisi dalam tabel Iceberg mencapai persentase tertentu dari file kecil, beban kerja Anda dapat memulai pekerjaan pemadatan untuk mengkonsolidasikan file menjadi yang lebih besar. Ini mencegah kueri melambat di luar tingkat yang dapat diterima.

Pemantauan tingkat meja

Layar berikut menunjukkan dasbor pemantauan tabel yang dibuat di Amazon Quick Sight. Dasbor ini menanyakan tabel metadata Iceberg dengan menggunakan Spark SQL, dan menangkap metrik terperinci seperti jumlah file aktif dan total penyimpanan. Informasi ini kemudian disimpan dalam AWS Glue tabel untuk tujuan operasional. Akhirnya, dasbor Quick Sight, seperti yang ditunjukkan pada ilustrasi berikut, dibuat dengan menggunakan Amazon Athena. Informasi ini membantu Anda mengidentifikasi dan mengatasi masalah spesifik dalam sistem Anda.



Contoh dasbor Quick Sight mengumpulkan indikator kinerja utama berikut (KPIs) untuk tabel Iceberg:

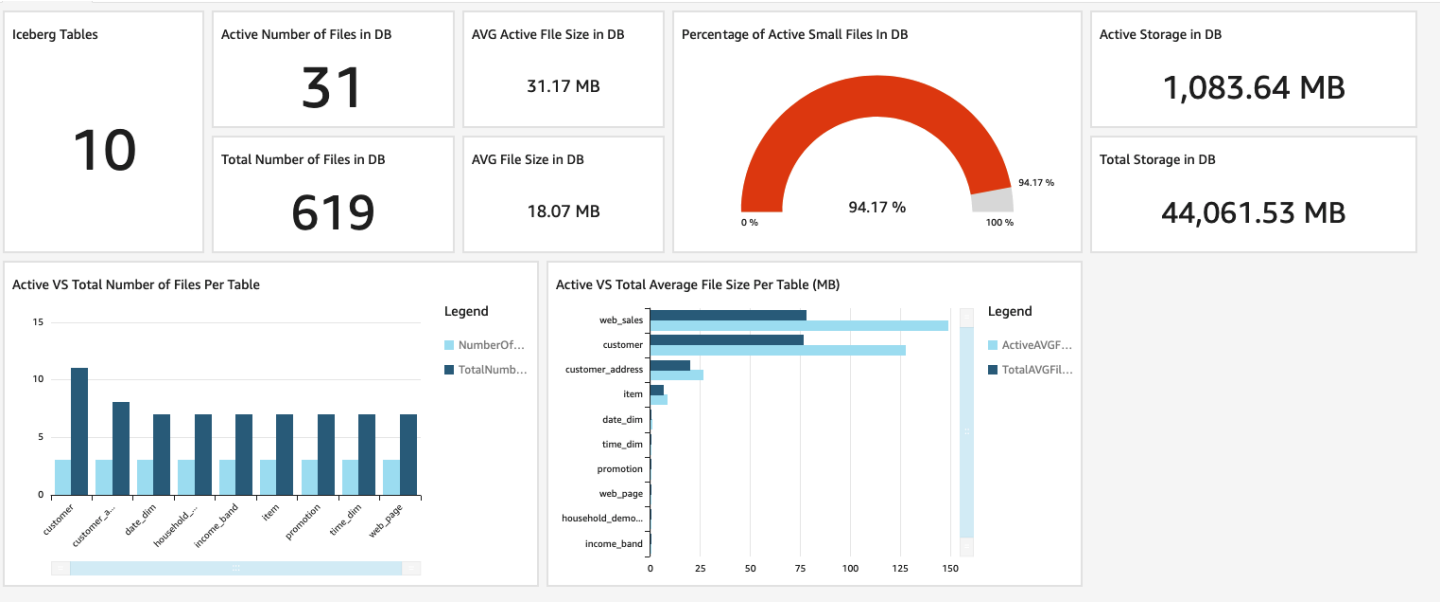
KPI	Deskripsi	Kueri
Jumlah file	Jumlah file dalam tabel Iceberg (untuk semua snapshot)	`select count(*) from <catalog.database.table_name>.all_files`
Jumlah file aktif	Jumlah file aktif dalam snapshot terakhir dari tabel Iceberg	`select count(*) from <catalog.database.table_name>.files`
Ukuran file rata-rata	Ukuran file rata-rata, dalam megabyte, untuk semua file di tabel Iceberg	`select avg(file_size_in_bytes)/1000000 from <catalog.database.table_name>.all_files`
Ukuran file aktif rata-rata	Ukuran file rata-rata, dalam megabyte, untuk file aktif dalam tabel Iceberg	`select avg(file_size_in_bytes)/1000000 from <catalog.database.table_name>.files`
Persentase file kecil	Persentase file aktif yang lebih kecil dari 100 MB	`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`
Ukuran penyimpanan total	Ukuran total semua file dalam tabel, tidak termasuk file yatim piatu dan versi objek Amazon S3 (jika diaktifkan)	`select sum(file_size_in_bytes)/1000000`

KPI	Deskripsi	Kueri
		`from <catalog.database.table_name>.all_files`
Total ukuran penyimpanan aktif	Ukuran total semua file dalam snapshot saat ini dari tabel yang diberikan	`select sum(file_size_in_bytes)/1000000 from <catalog.database.table_name>.files`

Untuk informasi selengkapnya tentang membuat dasbor, lihat dokumentasi Quick Sight.

Pemantauan tingkat database

Contoh berikut menunjukkan dasbor pemantauan yang dibuat di Quick Sight untuk memberikan gambaran tingkat database KPIs untuk kumpulan tabel Iceberg.



Dasbor ini mengumpulkan yang berikut: KPIs

KPI	Deskripsi	Kueri
Jumlah file	Jumlah file dalam database Iceberg (untuk semua snapshot)	Dasbor ini menggunakan kueri tingkat tabel yang disediaka n di bagian sebelumnya dan mengkonsolidasikan hasilnya.
Jumlah file aktif	Jumlah file aktif dalam database Iceberg (berdasar kan snapshot terakhir dari tabel Iceberg)	
Ukuran file rata-rata	Ukuran file rata-rata, dalam megabyte, untuk semua file dalam database Iceberg	
Ukuran file aktif rata-rata	Ukuran file rata-rata, dalam megabyte, untuk semua file aktif dalam database Iceberg	
Persentase file kecil	Persentase file aktif yang lebih kecil dari 100 MB dalam database Iceberg	
Ukuran Penyimpanan Total	Ukuran total semua file dalam database, tidak termasuk file yatim piatu dan versi objek Amazon S3 (jika diaktifkan)	
Total ukuran penyimpanan aktif	Ukuran total semua file dalam snapshot saat ini dari semua tabel dalam database	

Pemeliharaan preventif

Dengan menyiapkan kemampuan pemantauan yang dibahas di bagian sebelumnya, Anda dapat mendekati pemeliharaan tabel dari sudut preventif alih-alih reaktif. Misalnya, Anda dapat menggunakan metrik tingkat tabel dan tingkat database untuk menjadwalkan tindakan seperti berikut:

- Gunakan pemadatan kemasan bin untuk mengelompokkan file kecil saat tabel mencapai N file kecil.
- Gunakan pemadatan kemasan bin untuk menggabungkan file hapus saat tabel mencapai N menghapus file di partisi tertentu.
- Hapus file kecil yang sudah dipadatkan dengan menghapus snapshot saat total penyimpanan X kali lebih tinggi dari penyimpanan aktif.

Tata kelola dan kontrol akses untuk Apache Iceberg di AWS

Apache Iceberg terintegrasi dengan AWS Lake Formation menyederhanakan tata kelola data. Integrasi ini memungkinkan administrator data lake untuk menetapkan izin akses tingkat sel ke tabel Iceberg. Untuk contoh kueri tabel Iceberg dengan menggunakan Amazon Athena dan AWS Lake Formation, lihat posting AWS blog Berinteraksi dengan tabel Apache Iceberg menggunakan Amazon Athena dan lintas akun izin berbutir halus menggunakan. AWS Lake Formation

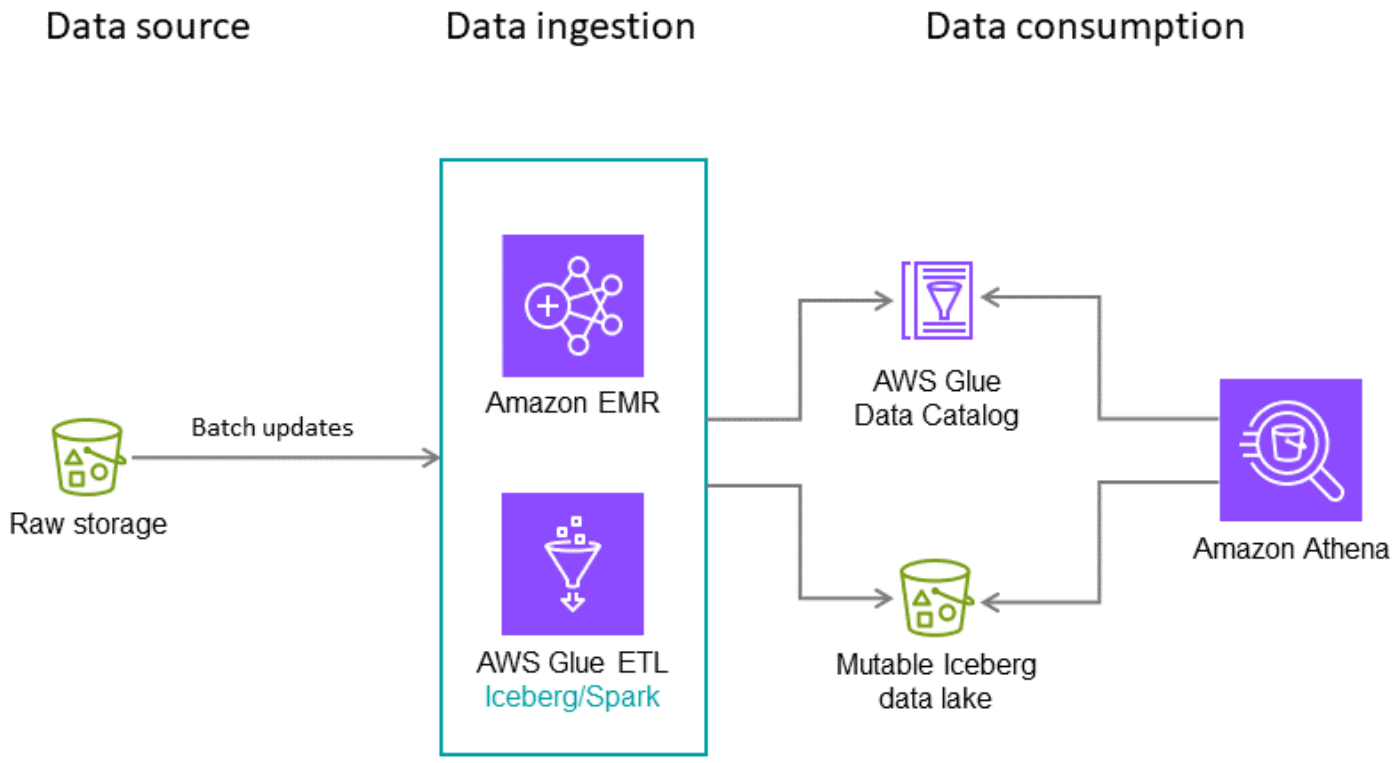
Arsitektur referensi untuk Apache Iceberg di AWS

Bagian ini memberikan contoh bagaimana menerapkan praktik terbaik dalam kasus penggunaan yang berbeda seperti konsumsi batch dan data lake yang menggabungkan konsumsi data batch dan streaming.

Konsumsi batch setiap malam

Untuk kasus penggunaan hipotetis ini, katakanlah tabel Iceberg Anda menelan transaksi kartu kredit setiap malam. Setiap batch hanya berisi pembaruan tambahan, yang harus digabungkan ke dalam tabel target. Beberapa kali per tahun, data historis lengkap diterima. Untuk skenario ini, kami merekomendasikan arsitektur dan konfigurasi berikut.

Catatan: Ini hanya sebuah contoh. Konfigurasi optimal tergantung pada data dan persyaratan Anda.



Rekomendasi:

- Ukuran file: 128 MB, karena tugas Apache Spark memproses data dalam potongan 128 MB.
- Jenis tulis: copy-on-write. Seperti yang dijelaskan sebelumnya dalam panduan ini, pendekatan ini membantu memastikan bahwa data ditulis dengan cara yang dioptimalkan untuk dibaca.

- Variabel partisi:year/month/day. Dalam kasus penggunaan hipotetis kami, kami paling sering menanyakan data terbaru, meskipun kami kadang-kadang menjalankan pemindaian tabel penuh selama dua tahun terakhir data. Tujuan partisi adalah untuk mendorong operasi baca cepat berdasarkan persyaratan kasus penggunaan.
- Urutkan urutan: stempel waktu
- Katalog data: AWS Glue Data Catalog

Danau data yang menggabungkan batch dan mendekati konsumsi waktu nyata

Anda dapat menyediakan data lake di Amazon S3 yang membagikan data batch dan streaming di seluruh akun dan Wilayah. Untuk diagram arsitektur dan detail, lihat posting AWS blog Membangun danau data transaksional menggunakan Apache Iceberg, AWS Glue, dan berbagi data lintas akun menggunakan dan Amazon Athena. AWS Lake Formation

Sumber daya

- Menggunakan kerangka Iceberg di AWS Glue(dokumentasi)AWS Glue
- Iceberg (dokumentasi Amazon EMR)
- Menggunakan tabel Apache Iceberg (dokumentasi Amazon Athena)
- Dokumentasi Amazon S3
- Dokumentasi cepat
- Katalog Glue dan Replikasi Izin Lake Formation (GitHub repositori)
- Dokumentasi Apache Iceberg
- Dokumentasi Apache Spark

Kontributor

Orang-orang berikut di AWS menulis, menulis bersama, dan meninjau panduan ini.

Kontributor

- Stefano Sandona, Arsitek Solusi, Big Data
- Imtiaz (Taz) Sayed, Pemimpin Teknologi Arsitek Solusi, Analisis
- Shana Schipers, Arsitek Solusi, Big Data
- Prashant Singh, Insinyur Pengembangan Perangkat Lunak, Amazon EMR
- Arun A K, Arsitek Solusi, Big Data dan ETL
- Francisco Morillo, Arsitek Solusi, Streaming
- Suthan Phillips, Arsitek Analitik, Amazon EMR
- Sercan Karaoglu, Arsitek Solusi
- Yonatan Dolan, Spesialis Analitik
- Guy Bachar, Arsitek Solusi
- Sofia Zilberman, Arsitek Solusi, Streaming
- Dan Stair, Arsitek Solusi Spesialis
- Sakti Mishra, Arsitek Solusi
- Ron Ortloff, Manajer Produk Utama, Amazon S3
- David Zhang, Arsitek Solusi, Analisis

Pengulas

- Rick Sears, Manajer Umum, Amazon EMR
- Linda OConnor, Spesialis EMR Amazon
- Ian Meyers, Direktur, Amazon EMR
- Vinita Ananth, Direktur, Manajemen Produk Amazon EMR
- Jason Berkowitz, Manajer Produk, AWS Lake Formation
- Mahesh Mishra, Manajer Produk, Amazon Redshift
- Vladimir Zlatkin, Manajer, Arsitektur Solusi, Big Data
- Karthik Prabhakar, Arsitek Analitik, Amazon EMR

- Vijay Jain, Manajer Produk
- Anupriti Warade, Manajer Produk, Amazon S3
- Ajit Tandale, Arsitek Solusi, Data
- Gwen Chen, Manajer Pemasaran Produk
- Noritaka Sekiyama, Arsitek Utama, Big Data

Riwayat dokumen

Tabel berikut menjelaskan perubahan signifikan pada panduan ini. Jika Anda ingin diberi tahu tentang pembaruan masa depan, Anda dapat berlangganan umpan RSS.

Perubahan	Deskripsi	Tanggal
Bagian baru	Menambahkan informasi tentang bekerja dengan spesifikasi format tabel Iceberg versi 3.	November 26, 2025
Koreksi	Informasi yang dikoreksi tentang `gc.enabled` pengaturan di bagian prosedur snapshot.	Oktober 24, 2025
Penambahan	Menambahkan bagian baru tentang bekerja dengan tabel Iceberg dengan menggunak an Trino dan PyIceberg, dan memperluas bagian pada tabel migrasi ke Iceberg.	Agustus 12, 2025
Update	Informasi yang disempurnakan dan diklarifikasi di seluruh panduan untuk mencermin kan versi terbaru AWS Glue, Amazon EMR, dan Apache Iceberg.	Juli 14, 2025
Penambahan	Menambahkan bagian baru tentang bekerja dengan tabel Iceberg dengan menggunakan Amazon Data Firehose.	Februari 20, 2025
Publikasi awal	—	April 30, 2024

AWS Glosarium Panduan Preskriptif

Berikut ini adalah istilah yang umum digunakan dalam strategi, panduan, dan pola yang disediakan oleh Panduan AWS Preskriptif. Untuk menyarankan entri, silakan gunakan tautan Berikan umpan balik di akhir glosarium.

Nomor

7 Rs

Tujuh strategi migrasi umum untuk memindahkan aplikasi ke cloud. Strategi ini dibangun di atas 5 Rs yang diidentifikasi Gartner pada tahun 2011 dan terdiri dari yang berikut:

- Refactor/Re-Architect — Memindahkan aplikasi dan memodifikasi arsitekturnya dengan memanfaatkan sepenuhnya fitur cloud-native untuk meningkatkan kelincahan, kinerja, dan skalabilitas. Ini biasanya melibatkan porting sistem operasi dan database. Contoh: Migrasikan database Oracle lokal Anda ke Amazon Aurora PostgreSQL Compatible Edition.
- Replatform (angkat dan bentuk ulang) — Pindahkan aplikasi ke cloud, dan perkenalkan beberapa tingkat pengoptimalan untuk memanfaatkan kemampuan cloud. Contoh: Migrasikan database Oracle lokal Anda ke Amazon Relational Database Service (Amazon RDS) untuk Oracle di. AWS Cloud
- Pembelian kembali (drop and shop) - Beralih ke produk yang berbeda, biasanya dengan beralih dari lisensi tradisional ke model SaaS. Contoh: Migrasikan sistem manajemen hubungan pelanggan (CRM) Anda ke Salesforce.com.
- Rehost (lift dan shift) — Pindahkan aplikasi ke cloud tanpa membuat perubahan apa pun untuk memanfaatkan kemampuan cloud. Contoh: Migrasikan database Oracle lokal Anda ke Oracle pada instans EC2 di. AWS Cloud
- Relokasi (hypervisor-level lift and shift) — Pindahkan infrastruktur ke cloud tanpa membeli perangkat keras baru, menulis ulang aplikasi, atau memodifikasi operasi yang ada. Anda memigrasikan server dari platform lokal ke layanan cloud untuk platform yang sama. Contoh: Migrasikan Microsoft Hyper-V aplikasi ke AWS.
- Pertahankan (kunjungi kembali) - Simpan aplikasi di lingkungan sumber Anda. Ini mungkin termasuk aplikasi yang memerlukan refactoring besar, dan Anda ingin menunda pekerjaan itu sampai nanti, dan aplikasi lama yang ingin Anda pertahankan, karena tidak ada pembenaran bisnis untuk memigrasikannya.

- Pensiun — Menonaktifkan atau menghapus aplikasi yang tidak lagi diperlukan di lingkungan sumber Anda.

A

ABAC

Lihat kontrol akses berbasis atribut.

layanan abstrak

Lihat layanan terkelola.

ASAM

Lihat atomisitas, konsistensi, isolasi, daya tahan.

migrasi aktif-aktif

Metode migrasi database di mana database sumber dan target tetap sinkron (dengan menggunakan alat replikasi dua arah atau operasi penulisan ganda), dan kedua database menangani transaksi dari menghubungkan aplikasi selama migrasi. Metode ini mendukung migrasi dalam batch kecil yang terkontrol alih-alih memerlukan pemotongan satu kali. Ini lebih fleksibel tetapi membutuhkan lebih banyak pekerjaan daripada migrasi aktif-pasif.

migrasi aktif-pasif

Metode migrasi database di mana database sumber dan target disimpan dalam sinkron, tetapi hanya database sumber yang menangani transaksi dari menghubungkan aplikasi sementara data direplikasi ke database target. Basis data target tidak menerima transaksi apa pun selama migrasi.

fungsi agregat

Fungsi SQL yang beroperasi pada sekelompok baris dan menghitung nilai pengembalian tunggal untuk grup. Contoh fungsi agregat meliputi `SUM` dan`MAX`.

AI

Lihat kecerdasan buatan.

AIOps

Lihat operasi kecerdasan buatan.

anonimisasi

Proses menghapus informasi pribadi secara permanen dalam kumpulan data. Anonimisasi dapat membantu melindungi privasi pribadi. Data anonim tidak lagi dianggap sebagai data pribadi.

anti-pola

Solusi yang sering digunakan untuk masalah berulang di mana solusinya kontra-produktif, tidak efektif, atau kurang efektif daripada alternatif.

kontrol aplikasi

Pendekatan keamanan yang memungkinkan penggunaan hanya aplikasi yang disetujui untuk membantu melindungi sistem dari malware.

portofolio aplikasi

Kumpulan informasi rinci tentang setiap aplikasi yang digunakan oleh organisasi, termasuk biaya untuk membangun dan memelihara aplikasi, dan nilai bisnisnya. Informasi ini adalah kunci untuk penemuan portofolio dan proses analisis dan membantu mengidentifikasi dan memprioritaskan aplikasi yang akan dimigrasi, dimodernisasi, dan dioptimalkan.

kecerdasan buatan (AI)

Bidang ilmu komputer yang didedikasikan untuk menggunakan teknologi komputasi untuk melakukan fungsi kognitif yang biasanya terkait dengan manusia, seperti belajar, memecahkan masalah, dan mengenali pola. Untuk informasi lebih lanjut, lihat Apa itu Kecerdasan Buatan?

operasi kecerdasan buatan (AIOps)

Proses menggunakan teknik pembelajaran mesin untuk memecahkan masalah operasional, mengurangi insiden operasional dan intervensi manusia, dan meningkatkan kualitas layanan. Untuk informasi selengkapnya tentang cara AIOps digunakan dalam strategi AWS migrasi, lihat panduan integrasi operasi.

enkripsi asimetris

Algoritma enkripsi yang menggunakan sepasang kunci, kunci publik untuk enkripsi dan kunci pribadi untuk dekripsi. Anda dapat berbagi kunci publik karena tidak digunakan untuk dekripsi, tetapi akses ke kunci pribadi harus sangat dibatasi.

atomisitas, konsistensi, isolasi, daya tahan (ACID)

Satu set properti perangkat lunak yang menjamin validitas data dan keandalan operasional database, bahkan dalam kasus kesalahan, kegagalan daya, atau masalah lainnya.

kontrol akses berbasis atribut (ABAC)

Praktik membuat izin berbutir halus berdasarkan atribut pengguna, seperti departemen, peran pekerjaan, dan nama tim. Untuk informasi selengkapnya, lihat ABAC untuk AWS dokumentasi AWS Identity and Access Management (IAM).

sumber data otoritatif

Lokasi di mana Anda menyimpan versi utama data, yang dianggap sebagai sumber informasi yang paling dapat diandalkan. Anda dapat menyalin data dari sumber data otoritatif ke lokasi lain untuk tujuan memproses atau memodifikasi data, seperti menganonimkan, menyunting, atau membuat nama samaran.

Zona Ketersediaan

Lokasi berbeda di dalam Wilayah AWS yang terisolasi dari kegagalan di Availability Zone lainnya dan menyediakan konektivitas jaringan latensi rendah yang murah ke Availability Zone lainnya di Wilayah yang sama.

AWS Kerangka Adopsi Cloud (AWS CAF)

Kerangka pedoman dan praktik terbaik AWS untuk membantu organisasi mengembangkan rencana yang efisien dan efektif untuk bergerak dengan sukses ke cloud. AWS CAF mengatur panduan ke dalam enam area fokus yang disebut perspektif: bisnis, orang, tata kelola, platform, keamanan, dan operasi. Perspektif bisnis, orang, dan tata kelola fokus pada keterampilan dan proses bisnis; perspektif platform, keamanan, dan operasi fokus pada keterampilan dan proses teknis. Misalnya, perspektif masyarakat menargetkan pemangku kepentingan yang menangani sumber daya manusia (SDM), fungsi kepegawaian, dan manajemen orang. Untuk perspektif ini, AWS CAF memberikan panduan untuk pengembangan, pelatihan, dan komunikasi orang untuk membantu mempersiapkan organisasi untuk adopsi cloud yang sukses. Untuk informasi lebih lanjut, lihat situs web AWS CAF dan whitepaper AWS CAF.

AWS Kerangka Kualifikasi Beban Kerja (AWS WQF)

Alat yang mengevaluasi beban kerja migrasi database, merekomendasikan strategi migrasi, dan memberikan perkiraan kerja. AWS WQF disertakan dengan AWS Schema Conversion Tool ()AWS SCT. Ini menganalisis skema database dan objek kode, kode aplikasi, dependensi, dan karakteristik kinerja, dan memberikan laporan penilaian.

B

bot buruk

Bot yang dimaksudkan untuk mengganggu atau menyebabkan kerugian bagi individu atau organisasi.

BCP

Lihat perencanaan kontinuitas bisnis.

grafik perilaku

Pandangan interaktif yang terpadu tentang perilaku dan interaksi sumber daya dari waktu ke waktu. Anda dapat menggunakan grafik perilaku dengan Amazon Detective untuk memeriksa upaya logon yang gagal, panggilan API yang mencurigakan, dan tindakan serupa. Untuk informasi selengkapnya, lihat Data dalam grafik perilaku di dokumentasi Detektif.

sistem big-endian

Sistem yang menyimpan byte paling signifikan terlebih dahulu. Lihat juga endianness.

klasifikasi biner

Sebuah proses yang memprediksi hasil biner (salah satu dari dua kelas yang mungkin). Misalnya, model ML Anda mungkin perlu memprediksi masalah seperti “Apakah email ini spam atau bukan spam?” atau “Apakah produk ini buku atau mobil?”

filter mekar

Struktur data probabilistik dan efisien memori yang digunakan untuk menguji apakah suatu elemen adalah anggota dari suatu himpunan.

deployment biru/hijau

Strategi penyebaran tempat Anda membuat dua lingkungan yang terpisah namun identik. Anda menjalankan versi aplikasi saat ini di satu lingkungan (biru) dan versi aplikasi baru di lingkungan lain (hijau). Strategi ini membantu Anda dengan cepat memutar kembali dengan dampak minimal.

bot

Aplikasi perangkat lunak yang menjalankan tugas otomatis melalui internet dan mensimulasikan aktivitas atau interaksi manusia. Beberapa bot berguna atau bermanfaat, seperti perayap web yang mengindeks informasi di internet. Beberapa bot lain, yang dikenal sebagai bot buruk, dimaksudkan untuk mengganggu atau membahayakan individu atau organisasi.

botnet

Jaringan bot yang terinfeksi oleh malware dan berada di bawah kendali satu pihak, yang dikenal sebagai bot herder atau operator bot. Botnet adalah mekanisme paling terkenal untuk skala bot dan dampaknya.

cabang

Area berisi repositori kode. Cabang pertama yang dibuat dalam repositori adalah cabang utama. Anda dapat membuat cabang baru dari cabang yang ada, dan Anda kemudian dapat mengembangkan fitur atau memperbaiki bug di cabang baru. Cabang yang Anda buat untuk membangun fitur biasanya disebut sebagai cabang fitur. Saat fitur siap dirilis, Anda menggabungkan cabang fitur kembali ke cabang utama. Untuk informasi selengkapnya, lihat Tentang cabang (GitHub dokumentasi).

akses break-glass

Dalam keadaan luar biasa dan melalui proses yang disetujui, cara cepat bagi pengguna untuk mendapatkan akses ke Akun AWS yang biasanya tidak memiliki izin untuk mengaksesnya. Untuk informasi lebih lanjut, lihat indikator Implementasikan prosedur break-glass dalam panduan Well-Architected AWS .

strategi brownfield

Infrastruktur yang ada di lingkungan Anda. Saat mengadopsi strategi brownfield untuk arsitektur sistem, Anda merancang arsitektur di sekitar kendala sistem dan infrastruktur saat ini. Jika Anda memperluas infrastruktur yang ada, Anda dapat memadukan strategi brownfield dan greenfield.

cache penyangga

Area memori tempat data yang paling sering diakses disimpan.

kemampuan bisnis

Apa yang dilakukan bisnis untuk menghasilkan nilai (misalnya, penjualan, layanan pelanggan, atau pemasaran). Arsitektur layanan mikro dan keputusan pengembangan dapat didorong oleh kemampuan bisnis. Untuk informasi selengkapnya, lihat bagian Terorganisir di sekitar kemampuan bisnis dari Menjalankan layanan mikro kontainer di whitepaper. AWS

perencanaan kelangsungan bisnis (BCP)

Rencana yang membahas dampak potensial dari peristiwa yang mengganggu, seperti migrasi skala besar, pada operasi dan memungkinkan bisnis untuk melanjutkan operasi dengan cepat.

C

KAFE

Lihat Kerangka Adopsi AWS Cloud.

penyebaran kenari

Rilis versi yang lambat dan bertahap untuk pengguna akhir. Ketika Anda yakin, Anda menyebarkan versi baru dan mengganti versi saat ini secara keseluruhan.

CCoE

Lihat Cloud Center of Excellence.

CDC

Lihat mengubah pengambilan data.

ubah pengambilan data (CDC)

Proses melacak perubahan ke sumber data, seperti tabel database, dan merekam metadata tentang perubahan tersebut. Anda dapat menggunakan CDC untuk berbagai tujuan, seperti mengaudit atau mereplikasi perubahan dalam sistem target untuk mempertahankan sinkronisasi.

rekayasa kekacauan

Sengaja memperkenalkan kegagalan atau peristiwa yang mengganggu untuk menguji ketahanan sistem. Anda dapat menggunakan AWS Fault Injection Service (AWS FIS) untuk melakukan eksperimen yang menekankan AWS beban kerja Anda dan mengevaluasi responsnya.

CI/CD

Lihat integrasi berkelanjutan dan pengiriman berkelanjutan.

klasifikasi

Proses kategorisasi yang membantu menghasilkan prediksi. Model ML untuk masalah klasifikasi memprediksi nilai diskrit. Nilai diskrit selalu berbeda satu sama lain. Misalnya, model mungkin perlu mengevaluasi apakah ada mobil dalam gambar atau tidak.

Enkripsi sisi klien

Enkripsi data secara lokal, sebelum target Layanan AWS menerimanya.

Pusat Keunggulan Cloud (CCoE)

Tim multi-disiplin yang mendorong upaya adopsi cloud di seluruh organisasi, termasuk mengembangkan praktik terbaik cloud, memobilisasi sumber daya, menetapkan jadwal migrasi, dan memimpin organisasi melalui transformasi skala besar. Untuk informasi selengkapnya, lihat posting CCo E di Blog Strategi AWS Cloud Perusahaan.

komputasi cloud

Teknologi cloud yang biasanya digunakan untuk penyimpanan data jarak jauh dan manajemen perangkat IoT. Cloud computing umumnya terhubung ke teknologi edge computing.

model operasi cloud

Dalam organisasi TI, model operasi yang digunakan untuk membangun, mematangkan, dan mengoptimalkan satu atau lebih lingkungan cloud. Untuk informasi selengkapnya, lihat Membangun Model Operasi Cloud Anda.

tahap adopsi cloud

Empat fase yang biasanya dilalui organisasi ketika mereka bermigrasi ke AWS Cloud:

- Proyek — Menjalankan beberapa proyek terkait cloud untuk bukti konsep dan tujuan pembelajaran
- Foundation — Melakukan investasi dasar untuk meningkatkan adopsi cloud Anda (misalnya, membuat landing zone, mendefinisikan CCo E, membuat model operasi)
- Migrasi — Migrasi aplikasi individual
- Re-invention — Mengoptimalkan produk dan layanan, dan berinovasi di cloud

Tahapan ini didefinisikan oleh Stephen Orban dalam posting blog The Journey Toward Cloud-First & the Stages of Adoption di blog Strategi Perusahaan. AWS Cloud Untuk informasi tentang bagaimana kaitannya dengan strategi AWS migrasi, lihat panduan kesiapan migrasi.

CMDB

Lihat database manajemen konfigurasi.

repositori kode

Lokasi di mana kode sumber dan aset lainnya, seperti dokumentasi, sampel, dan skrip, disimpan dan diperbarui melalui proses kontrol versi. Repositori cloud umum termasuk GitHub atau. Bitbucket Cloud Setiap versi kode disebut cabang. Dalam struktur layanan mikro, setiap repositori

dikhususkan untuk satu bagian fungsionalitas. Pipa CI/CD tunggal dapat menggunakan beberapa repositori.

cache dingin

Cache buffer yang kosong, tidak terisi dengan baik, atau berisi data basi atau tidak relevan. Ini mempengaruhi kinerja karena instance database harus membaca dari memori utama atau disk, yang lebih lambat daripada membaca dari cache buffer.

data dingin

Data yang jarang diakses dan biasanya historis. Saat menanyakan jenis data ini, kueri lambat biasanya dapat diterima. Memindahkan data ini ke tingkat penyimpanan atau kelas yang berkinerja lebih rendah dan lebih murah dapat mengurangi biaya.

visi komputer (CV)

Bidang AI yang menggunakan pembelajaran mesin untuk menganalisis dan mengekstrak informasi dari format visual seperti gambar dan video digital. Misalnya, Amazon SageMaker AI menyediakan algoritma pemrosesan gambar untuk CV.

konfigurasi drift

Untuk beban kerja, konfigurasi berubah dari status yang diharapkan. Ini dapat menyebabkan beban kerja menjadi tidak patuh, dan biasanya bertahap dan tidak disengaja.

database manajemen konfigurasi (CMDB)

Repositori yang menyimpan dan mengelola informasi tentang database dan lingkungan TI, termasuk komponen perangkat keras dan perangkat lunak dan konfigurasinya. Anda biasanya menggunakan data dari CMDB dalam penemuan portofolio dan tahap analisis migrasi.

paket kesesuaian

Kumpulan AWS Config aturan dan tindakan remediasi yang dapat Anda kumpulkan untuk menyesuaikan kepatuhan dan pemeriksaan keamanan Anda. Anda dapat menerapkan paket kesesuaian sebagai entitas tunggal di Akun AWS dan Region, atau di seluruh organisasi, dengan menggunakan templat YAMM. Untuk informasi selengkapnya, lihat Paket kesesuaian dalam dokumentasi. AWS Config

integrasi berkelanjutan dan pengiriman berkelanjutan (CI/CD)

Proses mengotomatiskan sumber, membangun, menguji, pementasan, dan tahap produksi dari proses rilis perangkat lunak. CI/CD biasanya digambarkan sebagai pipa. CI/CD dapat membantu

Anda mengotomatiskan proses, meningkatkan produktivitas, meningkatkan kualitas kode, dan memberikan lebih cepat. Untuk informasi lebih lanjut, lihat Manfaat pengiriman berkelanjutan. CD juga dapat berarti penerapan berkelanjutan. Untuk informasi selengkapnya, lihat Continuous Delivery vs Continuous Deployment.

CV

Lihat visi komputer.

D

data saat istirahat

Data yang stasioner di jaringan Anda, seperti data yang ada di penyimpanan.

klasifikasi data

Proses untuk mengidentifikasi dan mengkategorikan data dalam jaringan Anda berdasarkan kekritisan dan sensitivitasnya. Ini adalah komponen penting dari setiap strategi manajemen risiko keamanan siber karena membantu Anda menentukan perlindungan dan kontrol retensi yang tepat untuk data. Klasifikasi data adalah komponen pilar keamanan dalam AWS Well-Architected Framework. Untuk informasi selengkapnya, lihat Klasifikasi data.

penyimpangan data

Variasi yang berarti antara data produksi dan data yang digunakan untuk melatih model ML, atau perubahan yang berarti dalam data input dari waktu ke waktu. Penyimpangan data dapat mengurangi kualitas, akurasi, dan keadilan keseluruhan dalam prediksi model ML.

data dalam transit

Data yang aktif bergerak melalui jaringan Anda, seperti antara sumber daya jaringan.

jala data

Kerangka arsitektur yang menyediakan kepemilikan data terdistribusi dan terdesentralisasi dengan manajemen dan tata kelola terpusat.

minimalisasi data

Prinsip pengumpulan dan pemrosesan hanya data yang sangat diperlukan. Mempraktikkan minimalisasi data di dalamnya AWS Cloud dapat mengurangi risiko privasi, biaya, dan jejak karbon analitik Anda.

perimeter data

Satu set pagar pembatas pencegahan di AWS lingkungan Anda yang membantu memastikan bahwa hanya identitas tepercaya yang mengakses sumber daya tepercaya dari jaringan yang diharapkan. Untuk informasi selengkapnya, lihat Membangun perimeter data pada AWS.

prapemrosesan data

Untuk mengubah data mentah menjadi format yang mudah diuraikan oleh model ML Anda. Preprocessing data dapat berarti menghapus kolom atau baris tertentu dan menangani nilai yang hilang, tidak konsisten, atau duplikat.

asal data

Proses melacak asal dan riwayat data sepanjang siklus hidupnya, seperti bagaimana data dihasilkan, ditransmisikan, dan disimpan.

subjek data

Individu yang datanya dikumpulkan dan diproses.

gudang data

Sistem manajemen data yang mendukung intelijen bisnis, seperti analitik. Gudang data biasanya berisi sejumlah besar data historis, dan biasanya digunakan untuk kueri dan analisis.

bahasa definisi database (DDL)

Pernyataan atau perintah untuk membuat atau memodifikasi struktur tabel dan objek dalam database.

bahasa manipulasi basis data (DHTML)

Pernyataan atau perintah untuk memodifikasi (memasukkan, memperbarui, dan menghapus) informasi dalam database.

DDL

Lihat bahasa definisi database.

ansambel yang dalam

Untuk menggabungkan beberapa model pembelajaran mendalam untuk prediksi. Anda dapat menggunakan ansambel dalam untuk mendapatkan prediksi yang lebih akurat atau untuk memperkirakan ketidakpastian dalam prediksi.

pembelajaran mendalam

Subbidang ML yang menggunakan beberapa lapisan jaringan saraf tiruan untuk mengidentifikasi pemetaan antara data input dan variabel target yang diinginkan.

defense-in-depth

Pendekatan keamanan informasi di mana serangkaian mekanisme dan kontrol keamanan dilapisi dengan cermat di seluruh jaringan komputer untuk melindungi kerahasiaan, integritas, dan ketersediaan jaringan dan data di dalamnya. Saat Anda mengadopsi strategi ini AWS, Anda menambahkan beberapa kontrol pada lapisan AWS Organizations struktur yang berbeda untuk membantu mengamankan sumber daya. Misalnya, defense-in-depth pendekatan mungkin menggabungkan otentikasi multi-faktor, segmentasi jaringan, dan enkripsi.

administrator yang didelegasikan

Di AWS Organizations, layanan yang kompatibel dapat mendaftarkan akun AWS anggota untuk mengelola akun organisasi dan mengelola izin untuk layanan tersebut. Akun ini disebut administrator yang didelegasikan untuk layanan itu. Untuk informasi selengkapnya dan daftar layanan yang kompatibel, lihat Layanan yang berfungsi dengan AWS Organizations AWS Organizations dokumentasi.

deployment

Proses pembuatan aplikasi, fitur baru, atau perbaikan kode tersedia di lingkungan target. Deployment melibatkan penerapan perubahan dalam basis kode dan kemudian membangun dan menjalankan basis kode itu di lingkungan aplikasi.

lingkungan pengembangan

Lihat lingkungan.

kontrol detektif

Kontrol keamanan yang dirancang untuk mendeteksi, mencatat, dan memperingatkan setelah suatu peristiwa terjadi. Kontrol ini adalah garis pertahanan kedua, memperingatkan Anda tentang peristiwa keamanan yang melewati kontrol pencegahan yang ada. Untuk informasi selengkapnya, lihat Kontrol Detektif dalam Menerapkan kontrol keamanan pada. AWS

pemetaan aliran nilai pengembangan (DVSM)

Sebuah proses yang digunakan untuk mengidentifikasi dan memprioritaskan kendala yang mempengaruhi kecepatan dan kualitas dalam siklus hidup pengembangan perangkat lunak. DVSM memperluas proses pemetaan aliran nilai yang awalnya dirancang untuk praktik

manufaktur ramping. Ini berfokus pada langkah-langkah dan tim yang diperlukan untuk menciptakan dan memindahkan nilai melalui proses pengembangan perangkat lunak.

kembar digital

Representasi virtual dari sistem dunia nyata, seperti bangunan, pabrik, peralatan industri, atau jalur produksi. Kembar digital mendukung pemeliharaan prediktif, pemantauan jarak jauh, dan optimalisasi produksi.

tabel dimensi

Dalam skema bintang, tabel yang lebih kecil yang berisi atribut data tentang data kuantitatif dalam tabel fakta. Atribut tabel dimensi biasanya bidang teks atau angka diskrit yang berperilaku seperti teks. Atribut ini biasanya digunakan untuk pembatasan kueri, pemfilteran, dan pelabelan set hasil.

musibah

Peristiwa yang mencegah beban kerja atau sistem memenuhi tujuan bisnisnya di lokasi utama yang digunakan. Peristiwa ini dapat berupa bencana alam, kegagalan teknis, atau akibat dari tindakan manusia, seperti kesalahan konfigurasi yang tidak disengaja atau serangan malware.

pemulihan bencana (DR)

Strategi dan proses yang Anda gunakan untuk meminimalkan downtime dan kehilangan data yang disebabkan oleh bencana. Untuk informasi selengkapnya, lihat Disaster Recovery of Workloads on AWS: Recovery in the Cloud in the AWS Well-Architected Framework.

DML~

Lihat bahasa manipulasi basis data.

desain berbasis domain

Pendekatan untuk mengembangkan sistem perangkat lunak yang kompleks dengan menghubungkan komponennya ke domain yang berkembang, atau tujuan bisnis inti, yang dilayani oleh setiap komponen. Konsep ini diperkenalkan oleh Eric Evans dalam bukunya, Domain-Driven Design: Tackling Complexity in the Heart of Software (Boston: Addison-Wesley Professional, 2003). Untuk informasi tentang cara menggunakan desain berbasis domain dengan pola gambar pencekik, lihat Memodernisasi layanan web Microsoft ASP.NET (ASMX) lama secara bertahap menggunakan container dan Amazon API Gateway.

DR

Lihat pemulihan bencana.

deteksi drift

Melacak penyimpangan dari konfigurasi dasar. Misalnya, Anda dapat menggunakan AWS CloudFormation untuk mendeteksi penyimpangan dalam sumber daya sistem, atau Anda dapat menggunakannya AWS Control Tower untuk mendeteksi perubahan di landing zone yang mungkin memengaruhi kepatuhan terhadap persyaratan tata kelola.

DVSM

Lihat pemetaan aliran nilai pengembangan.

E

EDA

Lihat analisis data eksplorasi.

EDI

Lihat pertukaran data elektronik.

komputasi tepi

Teknologi yang meningkatkan daya komputasi untuk perangkat pintar di tepi jaringan IoT. Jika dibandingkan dengan komputasi awan, komputasi tepi dapat mengurangi latensi komunikasi dan meningkatkan waktu respons.

pertukaran data elektronik (EDI)

Pertukaran otomatis dokumen bisnis antar organisasi. Untuk informasi selengkapnya, lihat Apa itu Pertukaran Data Elektronik.

enkripsi

Proses komputasi yang mengubah data plaintext, yang dapat dibaca manusia, menjadi ciphertext.

kunci enkripsi

String kriptografi dari bit acak yang dihasilkan oleh algoritma enkripsi. Panjang kunci dapat bervariasi, dan setiap kunci dirancang agar tidak dapat diprediksi dan unik.

endianness

Urutan byte disimpan dalam memori komputer. Sistem big-endian menyimpan byte paling signifikan terlebih dahulu. Sistem little-endian menyimpan byte paling tidak signifikan terlebih dahulu.

titik akhir

Lihat titik akhir layanan.

layanan endpoint

Layanan yang dapat Anda host di cloud pribadi virtual (VPC) untuk dibagikan dengan pengguna lain. Anda dapat membuat layanan endpoint dengan AWS PrivateLink dan memberikan izin kepada prinsipal lain Akun AWS atau ke AWS Identity and Access Management (IAM). Akun atau prinsipal ini dapat terhubung ke layanan endpoint Anda secara pribadi dengan membuat titik akhir VPC antarmuka. Untuk informasi selengkapnya, lihat Membuat layanan titik akhir di dokumentasi Amazon Virtual Private Cloud (Amazon VPC).

perencanaan sumber daya perusahaan (ERP)

Sistem yang mengotomatiskan dan mengelola proses bisnis utama (seperti akuntansi, MES, dan manajemen proyek) untuk suatu perusahaan.

enkripsi amplop

Proses mengenkripsi kunci enkripsi dengan kunci enkripsi lain. Untuk informasi selengkapnya, lihat Enkripsi amplop dalam dokumentasi AWS Key Management Service (AWS KMS).

lingkungan

Sebuah contoh dari aplikasi yang sedang berjalan. Berikut ini adalah jenis lingkungan yang umum dalam komputasi awan:

- Development Environment — Sebuah contoh dari aplikasi yang berjalan yang hanya tersedia untuk tim inti yang bertanggung jawab untuk memelihara aplikasi. Lingkungan pengembangan digunakan untuk menguji perubahan sebelum mempromosikannya ke lingkungan atas. Jenis lingkungan ini kadang-kadang disebut sebagai lingkungan pengujian.
- lingkungan yang lebih rendah — Semua lingkungan pengembangan untuk aplikasi, seperti yang digunakan untuk build awal dan pengujian.
- lingkungan produksi — Sebuah contoh dari aplikasi yang berjalan yang dapat diakses oleh pengguna akhir. Dalam sebuah CI/CD pipeline, lingkungan produksi adalah lingkungan penyebaran terakhir.
- lingkungan atas — Semua lingkungan yang dapat diakses oleh pengguna selain tim pengembangan inti. Ini dapat mencakup lingkungan produksi, lingkungan praproduksi, dan lingkungan untuk pengujian penerimaan pengguna.

epik

Dalam metodologi tangkas, kategori fungsional yang membantu mengatur dan memprioritaskan pekerjaan Anda. Epik memberikan deskripsi tingkat tinggi tentang persyaratan dan tugas implementasi. Misalnya, epos keamanan AWS CAF mencakup manajemen identitas dan akses, kontrol detektif, keamanan infrastruktur, perlindungan data, dan respons insiden. Untuk informasi selengkapnya tentang epos dalam strategi AWS migrasi, lihat panduan implementasi program.

ERP

Lihat perencanaan sumber daya perusahaan.

analisis data eksplorasi (EDA)

Proses menganalisis dataset untuk memahami karakteristik utamanya. Anda mengumpulkan atau mengumpulkan data dan kemudian melakukan penyelidikan awal untuk menemukan pola, mendeteksi anomali, dan memeriksa asumsi. EDA dilakukan dengan menghitung statistik ringkasan dan membuat visualisasi data.

F

tabel fakta

Tabel tengah dalam skema bintang. Ini menyimpan data kuantitatif tentang operasi bisnis. Biasanya, tabel fakta berisi dua jenis kolom: kolom yang berisi ukuran dan yang berisi kunci asing ke tabel dimensi.

gagal cepat

Filosofi yang menggunakan pengujian yang sering dan bertahap untuk mengurangi siklus hidup pengembangan. Ini adalah bagian penting dari pendekatan tangkas.

batas isolasi kesalahan

Dalam AWS Cloud, batas seperti Availability Zone, Wilayah AWS, control plane, atau data plane yang membatasi efek kegagalan dan membantu meningkatkan ketahanan beban kerja. Untuk informasi selengkapnya, lihat Batas Isolasi AWS Kesalahan.

cabang fitur

Lihat cabang.

fitur

Data input yang Anda gunakan untuk membuat prediksi. Misalnya, dalam konteks manufaktur, fitur bisa berupa gambar yang diambil secara berkala dari lini manufaktur.

pentingnya fitur

Seberapa signifikan fitur untuk prediksi model. Ini biasanya dinyatakan sebagai skor numerik yang dapat dihitung melalui berbagai teknik, seperti Shapley Additive Explanations (SHAP) dan gradien terintegrasi. Untuk informasi lebih lanjut, lihat Interpretabilitas model pembelajaran mesin dengan. AWS

transformasi fitur

Untuk mengoptimalkan data untuk proses ML, termasuk memperkaya data dengan sumber tambahan, menskalakan nilai, atau mengekstrak beberapa set informasi dari satu bidang data. Hal ini memungkinkan model ML untuk mendapatkan keuntungan dari data. Misalnya, jika Anda memecah tanggal “2021-05-27 00:15:37" menjadi “2021", “Mei”, “Kamis”, dan “15", Anda dapat membantu algoritme pembelajaran mempelajari pola bernuansa yang terkait dengan komponen data yang berbeda.

beberapa tembakan mendorong

Menyediakan LLM dengan sejumlah kecil contoh yang menunjukkan tugas dan output yang diinginkan sebelum memintanya untuk melakukan tugas serupa. Teknik ini adalah aplikasi pembelajaran dalam konteks, di mana model belajar dari contoh (bidikan) yang tertanam dalam petunjuk. Beberapa bidikan dapat efektif untuk tugas-tugas yang memerlukan pemformatan, penalaran, atau pengetahuan domain tertentu. Lihat juga bidikan nol.

FGAC

Lihat kontrol akses berbutir halus.

kontrol akses berbutir halus (FGAC)

Penggunaan beberapa kondisi untuk mengizinkan atau menolak permintaan akses.

migrasi flash-cut

Metode migrasi database yang menggunakan replikasi data berkelanjutan melalui pengambilan data perubahan untuk memigrasikan data dalam waktu sesingkat mungkin, alih-alih menggunakan pendekatan bertahap. Tujuannya adalah untuk menjaga downtime seminimal mungkin.

FM

Lihat model pondasi.

model pondasi (FM)

Jaringan saraf pembelajaran mendalam yang besar yang telah melatih kumpulan data besar-besaran data umum dan tidak berlabel. FMs mampu melakukan berbagai tugas umum, seperti memahami bahasa, menghasilkan teks dan gambar, dan berbicara dalam bahasa alami. Untuk informasi selengkapnya, lihat Apa itu Model Foundation.

G

AI generatif

Subset model AI yang telah dilatih pada sejumlah besar data dan yang dapat menggunakan prompt teks sederhana untuk membuat konten dan artefak baru, seperti gambar, video, teks, dan audio. Untuk informasi lebih lanjut, lihat Apa itu AI Generatif.

pemblokiran geografis

Lihat pembatasan geografis.

pembatasan geografis (pemblokiran geografis)

Di Amazon CloudFront, opsi untuk mencegah pengguna di negara tertentu mengakses distribusi konten. Anda dapat menggunakan daftar izinkan atau daftar blokir untuk menentukan negara yang disetujui dan dilarang. Untuk informasi selengkapnya, lihat Membatasi distribusi geografis konten Anda dalam dokumentasi. CloudFront

Alur kerja Gitflow

Pendekatan di mana lingkungan bawah dan atas menggunakan cabang yang berbeda dalam repositori kode sumber. Alur kerja Gitflow dianggap warisan, dan alur kerja berbasis batang adalah pendekatan modern yang lebih disukai.

gambar emas

Sebuah snapshot dari sistem atau perangkat lunak yang digunakan sebagai template untuk menyebarkan instance baru dari sistem atau perangkat lunak itu. Misalnya, di bidang manufaktur, gambar emas dapat digunakan untuk menyediakan perangkat lunak pada beberapa perangkat dan membantu meningkatkan kecepatan, skalabilitas, dan produktivitas dalam operasi manufaktur perangkat.

strategi greenfield

Tidak adanya infrastruktur yang ada di lingkungan baru. Saat mengadopsi strategi greenfield untuk arsitektur sistem, Anda dapat memilih semua teknologi baru tanpa batasan kompatibilitas dengan infrastruktur yang ada, juga dikenal sebagai brownfield. Jika Anda memperluas infrastruktur yang ada, Anda dapat memadukan strategi brownfield dan greenfield.

pagar pembatas

Aturan tingkat tinggi yang membantu mengatur sumber daya, kebijakan, dan kepatuhan di seluruh unit organisasi ()OUs. Pagar pembatas preventif menegakkan kebijakan untuk memastikan keselarasan dengan standar kepatuhan. Mereka diimplementasikan dengan menggunakan kebijakan kontrol layanan dan batas izin IAM. Detective guardrails mendeteksi pelanggaran kebijakan dan masalah kepatuhan, dan menghasilkan peringatan untuk remediasi. Mereka diimplementasikan dengan menggunakan AWS Config, AWS Security Hub CSPM, Amazon GuardDuty AWS Trusted Advisor, Amazon Inspector, dan pemeriksaan khusus AWS Lambda .

H

HA

Lihat ketersediaan tinggi.

migrasi database heterogen

Memigrasi database sumber Anda ke database target yang menggunakan mesin database yang berbeda (misalnya, Oracle ke Amazon Aurora). Migrasi heterogen biasanya merupakan bagian dari upaya arsitektur ulang, dan mengubah skema dapat menjadi tugas yang kompleks. AWS menyediakan AWS SCT yang membantu dengan konversi skema.

ketersediaan tinggi (HA)

Kemampuan beban kerja untuk beroperasi terus menerus, tanpa intervensi, jika terjadi tantangan atau bencana. Sistem HA dirancang untuk gagal secara otomatis, secara konsisten memberikan kinerja berkualitas tinggi, dan menangani beban dan kegagalan yang berbeda dengan dampak kinerja minimal.

modernisasi sejarawan

Pendekatan yang digunakan untuk memodernisasi dan meningkatkan sistem teknologi operasional (OT) untuk melayani kebutuhan industri manufaktur dengan lebih baik. Sejarawan

adalah jenis database yang digunakan untuk mengumpulkan dan menyimpan data dari berbagai sumber di pabrik.

data penahanan

Sebagian dari data historis berlabel yang ditahan dari kumpulan data yang digunakan untuk melatih model pembelajaran mesin. Anda dapat menggunakan data penahanan untuk mengevaluasi kinerja model dengan membandingkan prediksi model dengan data penahanan.

migrasi database homogen

Memigrasi database sumber Anda ke database target yang berbagi mesin database yang sama (misalnya, Microsoft SQL Server ke Amazon RDS for SQL Server). Migrasi homogen biasanya merupakan bagian dari upaya rehosting atau replatforming. Anda dapat menggunakan utilitas database asli untuk memigrasi skema.

data panas

Data yang sering diakses, seperti data real-time atau data translasi terbaru. Data ini biasanya memerlukan tingkat atau kelas penyimpanan berkinerja tinggi untuk memberikan respons kueri yang cepat.

perbaikan terbaru

Perbaikan mendesak untuk masalah kritis dalam lingkungan produksi. Karena urgensinya, perbaikan terbaru biasanya dibuat di luar alur kerja DevOps rilis biasa.

periode hypercare

Segera setelah cutover, periode waktu ketika tim migrasi mengelola dan memantau aplikasi yang dimigrasi di cloud untuk mengatasi masalah apa pun. Biasanya, periode ini panjangnya 1-4 hari. Pada akhir periode hypercare, tim migrasi biasanya mentransfer tanggung jawab untuk aplikasi ke tim operasi cloud.

I

IAc

Lihat infrastruktur sebagai kode.

kebijakan berbasis identitas

Kebijakan yang dilampirkan pada satu atau beberapa prinsip IAM yang mendefinisikan izin mereka dalam lingkungan. AWS Cloud

aplikasi idle

Aplikasi yang memiliki penggunaan CPU dan memori rata-rata antara 5 dan 20 persen selama periode 90 hari. Dalam proyek migrasi, adalah umum untuk menghentikan aplikasi ini atau mempertahankannya di tempat.

IIoT

Lihat Internet of Things industri.

infrastruktur yang tidak dapat diubah

Model yang menyebarkan infrastruktur baru untuk beban kerja produksi alih-alih memperbarui, menambal, atau memodifikasi infrastruktur yang ada. Infrastruktur yang tidak dapat diubah secara inheren lebih konsisten, andal, dan dapat diprediksi daripada infrastruktur yang dapat berubah. Untuk informasi selengkapnya, lihat praktik terbaik Deploy using immutable infrastructure di AWS Well-Architected Framework.

masuk (masuknya) VPC

Dalam arsitektur AWS multi-akun, VPC yang menerima, memeriksa, dan merutekan koneksi jaringan dari luar aplikasi. Arsitektur Referensi AWS Keamanan merekomendasikan pengaturan akun Jaringan Anda dengan inbound, outbound, dan inspeksi VPCs untuk melindungi antarmuka dua arah antara aplikasi Anda dan internet yang lebih luas.

migrasi inkremental

Strategi cutover di mana Anda memigrasikan aplikasi Anda dalam bagian-bagian kecil alih-alih melakukan satu cutover penuh. Misalnya, Anda mungkin hanya memindahkan beberapa layanan mikro atau pengguna ke sistem baru pada awalnya. Setelah Anda memverifikasi bahwa semuanya berfungsi dengan baik, Anda dapat secara bertahap memindahkan layanan mikro atau pengguna tambahan hingga Anda dapat menonaktifkan sistem lama Anda. Strategi ini mengurangi risiko yang terkait dengan migrasi besar.

Industri 4.0

Sebuah istilah yang diperkenalkan oleh Klaus Schwab pada tahun 2016 untuk merujuk pada modernisasi proses manufaktur melalui kemajuan dalam konektivitas, data real-time, otomatisasi, analitik, dan AI/ML.

infrastruktur

Semua sumber daya dan aset yang terkandung dalam lingkungan aplikasi.

infrastruktur sebagai kode (IAc)

Proses penyediaan dan pengelolaan infrastruktur aplikasi melalui satu set file konfigurasi. IAc dirancang untuk membantu Anda memusatkan manajemen infrastruktur, menstandarisasi sumber daya, dan menskalakan dengan cepat sehingga lingkungan baru dapat diulang, andal, dan konsisten.

Internet of Things industri (IIoT)

Penggunaan sensor dan perangkat yang terhubung ke internet di sektor industri, seperti manufaktur, energi, otomotif, perawatan kesehatan, ilmu kehidupan, dan pertanian. Untuk informasi lebih lanjut, lihat Membangun strategi transformasi digital Internet of Things (IIoT) industri.

inspeksi VPC

Dalam arsitektur AWS multi-akun, VPC terpusat yang mengelola inspeksi lalu lintas jaringan antara VPCs (dalam yang sama atau berbeda Wilayah AWS), internet, dan jaringan lokal. Arsitektur Referensi AWS Keamanan merekomendasikan pengaturan akun Jaringan Anda dengan inbound, outbound, dan inspeksi VPCs untuk melindungi antarmuka dua arah antara aplikasi Anda dan internet yang lebih luas.

Internet of Things (IoT)

Jaringan objek fisik yang terhubung dengan sensor atau prosesor tertanam yang berkomunikasi dengan perangkat dan sistem lain melalui internet atau melalui jaringan komunikasi lokal. Untuk informasi selengkapnya, lihat Apa itu IoT?

interpretabilitas

Karakteristik model pembelajaran mesin yang menggambarkan sejauh mana manusia dapat memahami bagaimana prediksi model bergantung pada inputnya. Untuk informasi lebih lanjut, lihat Interpretabilitas model pembelajaran mesin dengan. AWS

IoT

Lihat Internet of Things.

Perpustakaan informasi TI (ITIL)

Serangkaian praktik terbaik untuk memberikan layanan TI dan menyelaraskan layanan ini dengan persyaratan bisnis. ITIL menyediakan dasar untuk ITSM.

Manajemen layanan TI (ITSM)

Kegiatan yang terkait dengan merancang, menerapkan, mengelola, dan mendukung layanan TI untuk suatu organisasi. Untuk informasi tentang mengintegrasikan operasi cloud dengan alat ITSM, lihat panduan integrasi operasi.

ITIL

Lihat perpustakaan informasi TI.

ITSM

Lihat manajemen layanan TI.

L

kontrol akses berbasis label (LBAC)

Implementasi kontrol akses wajib (MAC) di mana pengguna dan data itu sendiri masing-masing secara eksplisit diberi nilai label keamanan. Persimpangan antara label keamanan pengguna dan label keamanan data menentukan baris dan kolom mana yang dapat dilihat oleh pengguna.

landing zone

Landing zone adalah AWS lingkungan multi-akun yang dirancang dengan baik yang dapat diskalakan dan aman. Ini adalah titik awal dari mana organisasi Anda dapat dengan cepat meluncurkan dan menyebarkan beban kerja dan aplikasi dengan percaya diri dalam lingkungan keamanan dan infrastruktur mereka. Untuk informasi selengkapnya tentang zona pendaratan, lihat Menyiapkan lingkungan multi-akun AWS yang aman dan dapat diskalakan.

model bahasa besar (LLM)

Model AI pembelajaran mendalam yang dilatih sebelumnya pada sejumlah besar data. LLM dapat melakukan beberapa tugas, seperti menjawab pertanyaan, meringkas dokumen, menerjemahkan teks ke dalam bahasa lain, dan menyelesaikan kalimat. Untuk informasi lebih lanjut, lihat Apa itu LLMs.

migrasi besar

Migrasi 300 atau lebih server.

LBAC

Lihat kontrol akses berbasis label.

hak istimewa paling sedikit

Praktik keamanan terbaik untuk memberikan izin minimum yang diperlukan untuk melakukan tugas. Untuk informasi selengkapnya, lihat Menerapkan izin hak istimewa terkecil dalam dokumentasi IAM.

angkat dan geser

Lihat 7 Rs.

sistem endian kecil

Sebuah sistem yang menyimpan byte paling tidak signifikan terlebih dahulu. Lihat juga endianness.

LLM

Lihat model bahasa besar.

lingkungan yang lebih rendah

Lihat lingkungan.

M

pembelajaran mesin (ML)

Jenis kecerdasan buatan yang menggunakan algoritma dan teknik untuk pengenalan pola dan pembelajaran. ML menganalisis dan belajar dari data yang direkam, seperti data Internet of Things (IoT), untuk menghasilkan model statistik berdasarkan pola. Untuk informasi selengkapnya, lihat Machine Learning.

cabang utama

Lihat cabang.

malware

Perangkat lunak yang dirancang untuk membahayakan keamanan atau privasi komputer. Malware dapat mengganggu sistem komputer, membocorkan informasi sensitif, atau mendapatkan akses yang tidak sah. Contoh malware termasuk virus, worm, ransomware, Trojan horse, spyware, dan keyloggers.

layanan terkelola

Layanan AWS yang AWS mengoperasikan lapisan infrastruktur, sistem operasi, dan platform, dan Anda mengakses titik akhir untuk menyimpan dan mengambil data. Amazon Simple Storage Service (Amazon S3) dan Amazon DynamoDB adalah contoh layanan terkelola. Ini juga dikenal sebagai layanan abstrak.

sistem eksekusi manufaktur (MES)

Sistem perangkat lunak untuk melacak, memantau, mendokumentasikan, dan mengendalikan proses produksi yang mengubah bahan baku menjadi produk jadi di lantai toko.

PETA

Lihat Program Percepatan Migrasi.

mekanisme

Proses lengkap di mana Anda membuat alat, mendorong adopsi alat, dan kemudian memeriksa hasilnya untuk melakukan penyesuaian. Mekanisme adalah siklus yang memperkuat dan meningkatkan dirinya sendiri saat beroperasi. Untuk informasi lebih lanjut, lihat Membangun mekanisme di AWS Well-Architected Framework.

akun anggota

Semua Akun AWS selain akun manajemen yang merupakan bagian dari organisasi di AWS Organizations. Akun dapat menjadi anggota dari hanya satu organisasi pada suatu waktu.

MES

Lihat sistem eksekusi manufaktur.

Transportasi Telemetri Antrian Pesan (MQTT)

Protokol komunikasi ringan machine-to-machine (M2M), berdasarkan pola terbitkan/berlangganan, untuk perangkat IoT yang dibatasi sumber daya.

layanan mikro

Layanan kecil dan independen yang berkomunikasi dengan jelas APIs dan biasanya dimiliki oleh tim kecil yang mandiri. Misalnya, sistem asuransi mungkin mencakup layanan mikro yang memetakan kemampuan bisnis, seperti penjualan atau pemasaran, atau subdomain, seperti pembelian, klaim, atau analitik. Manfaat layanan mikro termasuk kelincahan, penskalaan yang fleksibel, penyebaran yang mudah, kode yang dapat digunakan kembali, dan ketahanan. Untuk

informasi selengkapnya, lihat Mengintegrasikan layanan mikro dengan menggunakan layanan tanpa AWS server.

arsitektur microservices

Pendekatan untuk membangun aplikasi dengan komponen independen yang menjalankan setiap proses aplikasi sebagai layanan mikro. Layanan mikro ini berkomunikasi melalui antarmuka yang terdefinisi dengan baik dengan menggunakan ringan. APIs Setiap layanan mikro dalam arsitektur ini dapat diperbarui, digunakan, dan diskalakan untuk memenuhi permintaan fungsi tertentu dari suatu aplikasi. Untuk informasi selengkapnya, lihat Menerapkan layanan mikro di AWS.

Program Percepatan Migrasi (MAP)

AWS Program yang menyediakan dukungan konsultasi, pelatihan, dan layanan untuk membantu organisasi membangun fondasi operasional yang kuat untuk pindah ke cloud, dan untuk membantu mengimbangi biaya awal migrasi. MAP mencakup metodologi migrasi untuk mengeksekusi migrasi lama dengan cara metodis dan seperangkat alat untuk mengotomatisasi dan mempercepat skenario migrasi umum.

migrasi dalam skala

Proses memindahkan sebagian besar portofolio aplikasi ke cloud dalam gelombang, dengan lebih banyak aplikasi bergerak pada tingkat yang lebih cepat di setiap gelombang. Fase ini menggunakan praktik dan pelajaran terbaik dari fase sebelumnya untuk mengimplementasikan pabrik migrasi tim, alat, dan proses untuk merampingkan migrasi beban kerja melalui otomatisasi dan pengiriman tangkas. Ini adalah fase ketiga dari strategi AWS migrasi.

pabrik migrasi

Tim lintas fungsi yang merampingkan migrasi beban kerja melalui pendekatan otomatis dan gesit. Tim pabrik migrasi biasanya mencakup operasi, analis dan pemilik bisnis, insinyur migrasi, pengembang, dan DevOps profesional yang bekerja di sprint. Antara 20 dan 50 persen portofolio aplikasi perusahaan terdiri dari pola berulang yang dapat dioptimalkan dengan pendekatan pabrik. Untuk informasi selengkapnya, lihat diskusi tentang pabrik migrasi dan panduan Pabrik Migrasi Cloud di kumpulan konten ini.

metadata migrasi

Informasi tentang aplikasi dan server yang diperlukan untuk menyelesaikan migrasi. Setiap pola migrasi memerlukan satu set metadata migrasi yang berbeda. Contoh metadata migrasi termasuk subnet target, grup keamanan, dan akun. AWS

pola migrasi

Tugas migrasi berulang yang merinci strategi migrasi, tujuan migrasi, dan aplikasi atau layanan migrasi yang digunakan. Contoh: Rehost migrasi ke Amazon EC2 AWS dengan Layanan Migrasi Aplikasi.

Penilaian Portofolio Migrasi (MPA)

Alat online yang menyediakan informasi untuk memvalidasi kasus bisnis untuk bermigrasi ke. AWS Cloud MPA menyediakan penilaian portofolio terperinci (ukuran kanan server, harga, perbandingan TCO, analisis biaya migrasi) serta perencanaan migrasi (analisis data aplikasi dan pengumpulan data, pengelompokan aplikasi, prioritas migrasi, dan perencanaan gelombang). Alat MPA (memerlukan login) tersedia gratis untuk semua AWS konsultan dan konsultan APN Partner.

Penilaian Kesiapan Migrasi (MRA)

Proses mendapatkan wawasan tentang status kesiapan cloud organisasi, mengidentifikasi kekuatan dan kelemahan, dan membangun rencana aksi untuk menutup kesenjangan yang diidentifikasi, menggunakan CAF. AWS Untuk informasi selengkapnya, lihat panduan kesiapan migrasi. MRA adalah tahap pertama dari strategi AWS migrasi.

strategi migrasi

Pendekatan yang digunakan untuk memigrasikan beban kerja ke. AWS Cloud Untuk informasi lebih lanjut, lihat entri 7 Rs di glosarium ini dan lihat Memobilisasi organisasi Anda untuk mempercepat migrasi skala besar.

ML

Lihat pembelajaran mesin.

modernisasi

Mengubah aplikasi usang (warisan atau monolitik) dan infrastrukturnya menjadi sistem yang gesit, elastis, dan sangat tersedia di cloud untuk mengurangi biaya, mendapatkan efisiensi, dan memanfaatkan inovasi. Untuk informasi selengkapnya, lihat Strategi untuk memodernisasi aplikasi di. AWS Cloud

penilaian kesiapan modernisasi

Evaluasi yang membantu menentukan kesiapan modernisasi aplikasi organisasi; mengidentifikasi manfaat, risiko, dan dependensi; dan menentukan seberapa baik organisasi dapat mendukung keadaan masa depan aplikasi tersebut. Hasil penilaian adalah cetak biru arsitektur target, peta

jalan yang merinci fase pengembangan dan tonggak untuk proses modernisasi, dan rencana aksi untuk mengatasi kesenjangan yang diidentifikasi. Untuk informasi lebih lanjut, lihat Mengevaluasi kesiapan modernisasi untuk aplikasi di. AWS Cloud

aplikasi monolitik (monolit)

Aplikasi yang berjalan sebagai layanan tunggal dengan proses yang digabungkan secara ketat. Aplikasi monolitik memiliki beberapa kelemahan. Jika satu fitur aplikasi mengalami lonjakan permintaan, seluruh arsitektur harus diskalakan. Menambahkan atau meningkatkan fitur aplikasi monolitik juga menjadi lebih kompleks ketika basis kode tumbuh. Untuk mengatasi masalah ini, Anda dapat menggunakan arsitektur microservices. Untuk informasi lebih lanjut, lihat Mengurai monolit menjadi layanan mikro.

MPA

Lihat Penilaian Portofolio Migrasi.

MQTT

Lihat Transportasi Telemetri Antrian Pesan.

klasifikasi multiclass

Sebuah proses yang membantu menghasilkan prediksi untuk beberapa kelas (memprediksi satu dari lebih dari dua hasil). Misalnya, model ML mungkin bertanya "Apakah produk ini buku, mobil, atau telepon?" atau "Kategori produk mana yang paling menarik bagi pelanggan ini?"

infrastruktur yang bisa berubah

Model yang memperbarui dan memodifikasi infrastruktur yang ada untuk beban kerja produksi. Untuk meningkatkan konsistensi, keandalan, dan prediktabilitas, AWS Well-Architected Framework merekomendasikan penggunaan infrastruktur yang tidak dapat diubah sebagai praktik terbaik.

O

OAC

Lihat kontrol akses asal.

OAI

Lihat identitas akses asal.

OCM

Lihat manajemen perubahan organisasi.

migrasi offline

Metode migrasi di mana beban kerja sumber diturunkan selama proses migrasi. Metode ini melibatkan waktu henti yang diperpanjang dan biasanya digunakan untuk beban kerja kecil dan tidak kritis.

OI

Lihat integrasi operasi.

OLA

Lihat perjanjian tingkat operasional.

migrasi online

Metode migrasi di mana beban kerja sumber disalin ke sistem target tanpa diambil offline. Aplikasi yang terhubung ke beban kerja dapat terus berfungsi selama migrasi. Metode ini melibatkan waktu henti nol hingga minimal dan biasanya digunakan untuk beban kerja produksi yang kritis.

OPC-UA

Lihat Komunikasi Proses Terbuka - Arsitektur Terpadu.

Komunikasi Proses Terbuka - Arsitektur Terpadu (OPC-UA)

Protokol komunikasi machine-to-machine (M2M) untuk otomasi industri. OPC-UA menyediakan standar interoperabilitas dengan enkripsi data, otentikasi, dan skema otorisasi.

perjanjian tingkat operasional (OLA)

Perjanjian yang menjelaskan apa yang dijanjikan kelompok TI fungsional untuk diberikan satu sama lain, untuk mendukung perjanjian tingkat layanan (SLA).

Tinjauan Kesiapan Operasional (ORR)

Daftar pertanyaan dan praktik terbaik terkait yang membantu Anda memahami, mengevaluasi, mencegah, atau mengurangi ruang lingkup insiden dan kemungkinan kegagalan. Untuk informasi lebih lanjut, lihat Ulasan Kesiapan Operasional (ORR) dalam Kerangka Kerja Well-Architected AWS .

teknologi operasional (OT)

Sistem perangkat keras dan perangkat lunak yang bekerja dengan lingkungan fisik untuk mengendalikan operasi industri, peralatan, dan infrastruktur. Di bidang manufaktur, integrasi sistem OT dan teknologi informasi (TI) adalah fokus utama untuk transformasi Industri 4.0.

integrasi operasi (OI)

Proses modernisasi operasi di cloud, yang melibatkan perencanaan kesiapan, otomatisasi, dan integrasi. Untuk informasi selengkapnya, lihat panduan integrasi operasi.

jejak organisasi

Jejak yang dibuat oleh AWS CloudTrail itu mencatat semua peristiwa untuk semua Akun AWS dalam organisasi di AWS Organizations. Jejak ini dibuat di setiap Akun AWS bagian organisasi dan melacak aktivitas di setiap akun. Untuk informasi selengkapnya, lihat Membuat jejak untuk organisasi dalam CloudTrail dokumentasi.

manajemen perubahan organisasi (OCM)

Kerangka kerja untuk mengelola transformasi bisnis utama yang mengganggu dari perspektif orang, budaya, dan kepemimpinan. OCM membantu organisasi mempersiapkan, dan transisi ke, sistem dan strategi baru dengan mempercepat adopsi perubahan, mengatasi masalah transisi, dan mendorong perubahan budaya dan organisasi. Dalam strategi AWS migrasi, kerangka kerja ini disebut percepatan orang, karena kecepatan perubahan yang diperlukan dalam proyek adopsi cloud. Untuk informasi lebih lanjut, lihat panduan OCM.

kontrol akses asal (OAC)

Di CloudFront, opsi yang disempurnakan untuk membatasi akses untuk mengamankan konten Amazon Simple Storage Service (Amazon S3) Anda. OAC mendukung semua bucket S3 di semua Wilayah AWS, enkripsi sisi server dengan AWS KMS (SSE-KMS), dan dinamis dan permintaan ke bucket S3. `PUT` `DELETE`

identitas akses asal (OAI)

Di CloudFront, opsi untuk membatasi akses untuk mengamankan konten Amazon S3 Anda. Saat Anda menggunakan OAI, CloudFront buat prinsipal yang dapat diautentikasi oleh Amazon S3. Prinsipal yang diautentikasi dapat mengakses konten dalam bucket S3 hanya melalui distribusi tertentu. CloudFront Lihat juga OAC, yang menyediakan kontrol akses yang lebih terperinci dan ditingkatkan.

ORR

Lihat tinjauan kesiapan operasional.

OT

Lihat teknologi operasional.

keluar (jalan keluar) VPC

Dalam arsitektur AWS multi-akun, VPC yang menangani koneksi jaringan yang dimulai dari dalam aplikasi. Arsitektur Referensi AWS Keamanan merekomendasikan pengaturan akun Jaringan Anda dengan inbound, outbound, dan inspeksi VPCs untuk melindungi antarmuka dua arah antara aplikasi Anda dan internet yang lebih luas.

P

batas izin

Kebijakan manajemen IAM yang dilampirkan pada prinsipal IAM untuk menetapkan izin maksimum yang dapat dimiliki pengguna atau peran. Untuk informasi selengkapnya, lihat Batas izin dalam dokumentasi IAM.

Informasi Identifikasi Pribadi (PII)

Informasi yang, jika dilihat secara langsung atau dipasangkan dengan data terkait lainnya, dapat digunakan untuk menyimpulkan identitas individu secara wajar. Contoh PII termasuk nama, alamat, dan informasi kontak.

PII

Lihat informasi yang dapat diidentifikasi secara pribadi.

buku pedoman

Serangkaian langkah yang telah ditentukan sebelumnya yang menangkap pekerjaan yang terkait dengan migrasi, seperti mengirimkan fungsi operasi inti di cloud. Buku pedoman dapat berupa skrip, runbook otomatis, atau ringkasan proses atau langkah-langkah yang diperlukan untuk mengoperasikan lingkungan modern Anda.

PLC

Lihat pengontrol logika yang dapat diprogram.

PLM

Lihat manajemen siklus hidup produk.

kebijakan

Objek yang dapat menentukan izin (lihat kebijakan berbasis identitas), menentukan kondisi akses (lihat kebijakan berbasis sumber daya), atau menentukan izin maksimum untuk semua akun di organisasi (lihat kebijakan kontrol layanan). AWS Organizations

ketekunan poliglot

Secara independen memilih teknologi penyimpanan data microservice berdasarkan pola akses data dan persyaratan lainnya. Jika layanan mikro Anda memiliki teknologi penyimpanan data yang sama, mereka dapat menghadapi tantangan implementasi atau mengalami kinerja yang buruk. Layanan mikro lebih mudah diimplementasikan dan mencapai kinerja dan skalabilitas yang lebih baik jika mereka menggunakan penyimpanan data yang paling sesuai dengan kebutuhan mereka.

penilaian portofolio

Proses menemukan, menganalisis, dan memprioritaskan portofolio aplikasi untuk merencanakan migrasi. Untuk informasi selengkapnya, lihat Mengevaluasi kesiapan migrasi.

predikat

Kondisi kueri yang mengembalikan `true` atau`false`, biasanya terletak di `WHERE` klausa.

predikat pushdown

Teknik pengoptimalan kueri database yang menyaring data dalam kueri sebelum transfer. Ini mengurangi jumlah data yang harus diambil dan diproses dari database relasional, dan meningkatkan kinerja kueri.

kontrol preventif

Kontrol keamanan yang dirancang untuk mencegah suatu peristiwa terjadi. Kontrol ini adalah garis pertahanan pertama untuk membantu mencegah akses tidak sah atau perubahan yang tidak diinginkan ke jaringan Anda. Untuk informasi selengkapnya, lihat Kontrol pencegahan dalam Menerapkan kontrol keamanan pada. AWS

principal

Entitas AWS yang dapat melakukan tindakan dan mengakses sumber daya. Entitas ini biasanya merupakan pengguna root untuk Akun AWS, peran IAM, atau pengguna. Untuk informasi selengkapnya, lihat Prinsip dalam istilah dan konsep Peran dalam dokumentasi IAM.

privasi berdasarkan desain

Pendekatan rekayasa sistem yang memperhitungkan privasi melalui seluruh proses pengembangan.

zona yang dihosting pribadi

Container yang menyimpan informasi tentang bagaimana Anda ingin Amazon Route 53 merespons kueri DNS untuk domain dan subdomainnya dalam satu atau lebih. VPCs Untuk informasi selengkapnya, lihat Bekerja dengan zona yang dihosting pribadi di dokumentasi Route 53.

kontrol proaktif

Kontrol keamanan yang dirancang untuk mencegah penyebaran sumber daya yang tidak sesuai. Kontrol ini memindai sumber daya sebelum disediakan. Jika sumber daya tidak sesuai dengan kontrol, maka itu tidak disediakan. Untuk informasi selengkapnya, lihat panduan referensi Kontrol dalam AWS Control Tower dokumentasi dan lihat Kontrol proaktif dalam Menerapkan kontrol keamanan pada AWS.

manajemen siklus hidup produk (PLM)

Manajemen data dan proses untuk suatu produk di seluruh siklus hidupnya, mulai dari desain, pengembangan, dan peluncuran, melalui pertumbuhan dan kematangan, hingga penurunan dan penghapusan.

lingkungan produksi

Lihat lingkungan.

pengontrol logika yang dapat diprogram (PLC)

Di bidang manufaktur, komputer yang sangat andal dan mudah beradaptasi yang memantau mesin dan mengotomatiskan proses manufaktur.

rantai cepat

Menggunakan output dari satu prompt LLM sebagai input untuk prompt berikutnya untuk menghasilkan respons yang lebih baik. Teknik ini digunakan untuk memecah tugas yang kompleks menjadi subtugas, atau untuk secara iteratif memperbaiki atau memperluas respons awal. Ini membantu meningkatkan akurasi dan relevansi respons model dan memungkinkan hasil yang lebih terperinci dan dipersonalisasi.

pseudonimisasi

Proses penggantian pengidentifikasi pribadi dalam kumpulan data dengan nilai placeholder. Pseudonimisasi dapat membantu melindungi privasi pribadi. Data pseudonim masih dianggap sebagai data pribadi.

publish/subscribe (pub/sub)

Pola yang memungkinkan komunikasi asinkron antara layanan mikro untuk meningkatkan skalabilitas dan daya tanggap. Misalnya, dalam MES berbasis layanan mikro, layanan mikro dapat mempublikasikan pesan peristiwa ke saluran yang dapat berlangganan layanan mikro lainnya. Sistem dapat menambahkan layanan mikro baru tanpa mengubah layanan penerbitan.

Q

rencana kueri

Serangkaian langkah, seperti instruksi, yang digunakan untuk mengakses data dalam sistem database relasional SQL.

regresi rencana kueri

Ketika pengoptimal layanan database memilih rencana yang kurang optimal daripada sebelum perubahan yang diberikan ke lingkungan database. Hal ini dapat disebabkan oleh perubahan statistik, kendala, pengaturan lingkungan, pengikatan parameter kueri, dan pembaruan ke mesin database.

R

Matriks RACI

Lihat bertanggung jawab, akuntabel, dikonsultasikan, diinformasikan (RACI).

LAP

Lihat Retrieval Augmented Generation.

ransomware

Perangkat lunak berbahaya yang dirancang untuk memblokir akses ke sistem komputer atau data sampai pembayaran dilakukan.

Matriks RASCI

Lihat bertanggung jawab, akuntabel, dikonsultasikan, diinformasikan (RACI).

RCAC

Lihat kontrol akses baris dan kolom.

replika baca

Salinan database yang digunakan untuk tujuan read-only. Anda dapat merutekan kueri ke replika baca untuk mengurangi beban pada database utama Anda.

arsitek ulang

Lihat 7 Rs.

tujuan titik pemulihan (RPO)

Jumlah waktu maksimum yang dapat diterima sejak titik pemulihan data terakhir. Ini menentukan apa yang dianggap sebagai kehilangan data yang dapat diterima antara titik pemulihan terakhir dan gangguan layanan.

tujuan waktu pemulihan (RTO)

Penundaan maksimum yang dapat diterima antara gangguan layanan dan pemulihan layanan.

refactor

Lihat 7 Rs.

Region

Kumpulan AWS sumber daya di wilayah geografis. Masing-masing Wilayah AWS terisolasi dan independen dari yang lain untuk memberikan toleransi kesalahan, stabilitas, dan ketahanan. Untuk informasi selengkapnya, lihat Menentukan Wilayah AWS akun yang dapat digunakan.

regresi

Teknik ML yang memprediksi nilai numerik. Misalnya, untuk memecahkan masalah “Berapa harga rumah ini akan dijual?” Model ML dapat menggunakan model regresi linier untuk memprediksi harga jual rumah berdasarkan fakta yang diketahui tentang rumah (misalnya, luas persegi).

rehost

Lihat 7 Rs.

melepaskan

Dalam proses penyebaran, tindakan mempromosikan perubahan pada lingkungan produksi.

memindahkan

Lihat 7 Rs.

memplatform ulang

Lihat 7 Rs.

pembelian kembali

Lihat 7 Rs.

ketahanan

Kemampuan aplikasi untuk melawan atau pulih dari gangguan. Ketersediaan tinggi dan pemulihan bencana adalah pertimbangan umum ketika merencanakan ketahanan di. AWS Cloud Untuk informasi lebih lanjut, lihat AWS Cloud Ketahanan.

kebijakan berbasis sumber daya

Kebijakan yang dilampirkan ke sumber daya, seperti bucket Amazon S3, titik akhir, atau kunci enkripsi. Jenis kebijakan ini menentukan prinsipal mana yang diizinkan mengakses, tindakan yang didukung, dan kondisi lain yang harus dipenuhi.

matriks yang bertanggung jawab, akuntabel, dikonsultasikan, diinformasikan (RACI)

Matriks yang mendefinisikan peran dan tanggung jawab untuk semua pihak yang terlibat dalam kegiatan migrasi dan operasi cloud. Nama matriks berasal dari jenis tanggung jawab yang didefinisikan dalam matriks: bertanggung jawab (R), akuntabel (A), dikonsultasikan (C), dan diinformasikan (I). Tipe dukungan (S) adalah opsional. Jika Anda menyertakan dukungan, matriks disebut matriks RASCI, dan jika Anda mengecualikannya, itu disebut matriks RACI.

kontrol responsif

Kontrol keamanan yang dirancang untuk mendorong remediasi efek samping atau penyimpangan dari garis dasar keamanan Anda. Untuk informasi selengkapnya, lihat Kontrol responsif dalam Menerapkan kontrol keamanan pada AWS.

melestarikan

Lihat 7 Rs.

pensiun

Lihat 7 Rs.

Retrieval Augmented Generation (RAG)

Teknologi AI generatif di mana LLM merujuk sumber data otoritatif yang berada di luar sumber data pelatihannya sebelum menghasilkan respons. Misalnya, model RAG mungkin melakukan

pencarian semantik dari basis pengetahuan organisasi atau data kustom. Untuk informasi lebih lanjut, lihat Apa itu RAG.

rotasi

Proses memperbarui rahasia secara berkala untuk membuatnya lebih sulit bagi penyerang untuk mengakses kredensil.

kontrol akses baris dan kolom (RCAC)

Penggunaan ekspresi SQL dasar dan fleksibel yang telah menetapkan aturan akses. RCAC terdiri dari izin baris dan topeng kolom.

RPO

Lihat tujuan titik pemulihan.

RTO

Lihat tujuan waktu pemulihan.

buku runbook

Satu set prosedur manual atau otomatis yang diperlukan untuk melakukan tugas tertentu. Ini biasanya dibangun untuk merampingkan operasi berulang atau prosedur dengan tingkat kesalahan yang tinggi.

D

SAML 2.0

Standar terbuka yang digunakan oleh banyak penyedia identitas (IdPs). Fitur ini memungkinkan sistem masuk tunggal gabungan (SSO), sehingga pengguna dapat masuk ke Konsol Manajemen AWS atau memanggil operasi AWS API tanpa Anda harus membuat pengguna di IAM untuk semua orang di organisasi Anda. Untuk informasi lebih lanjut tentang federasi berbasis SAMP 2.0, lihat Tentang federasi berbasis SAMP 2.0 dalam dokumentasi IAM.

SCADA

Lihat kontrol pengawasan dan akuisisi data.

SCP

Lihat kebijakan kontrol layanan.

Rahasia

Dalam AWS Secrets Manager, informasi rahasia atau terbatas, seperti kata sandi atau kredensil pengguna, yang Anda simpan dalam bentuk terenkripsi. Ini terdiri dari nilai rahasia dan metadata-nya. Nilai rahasia dapat berupa biner, string tunggal, atau beberapa string. Untuk informasi selengkapnya, lihat Apa yang ada di rahasia Secrets Manager? dalam dokumentasi Secrets Manager.

keamanan dengan desain

Pendekatan rekayasa sistem yang memperhitungkan keamanan melalui seluruh proses pengembangan.

kontrol keamanan

Pagar pembatas teknis atau administratif yang mencegah, mendeteksi, atau mengurangi kemampuan pelaku ancaman untuk mengeksploitasi kerentanan keamanan. Ada empat jenis kontrol keamanan utama: preventif, detektif, responsif, dan proaktif.

pengerasan keamanan

Proses mengurangi permukaan serangan untuk membuatnya lebih tahan terhadap serangan. Ini dapat mencakup tindakan seperti menghapus sumber daya yang tidak lagi diperlukan, menerapkan praktik keamanan terbaik untuk memberikan hak istimewa paling sedikit, atau menonaktifkan fitur yang tidak perlu dalam file konfigurasi.

sistem informasi keamanan dan manajemen acara (SIEM)

Alat dan layanan yang menggabungkan sistem manajemen informasi keamanan (SIM) dan manajemen acara keamanan (SEM). Sistem SIEM mengumpulkan, memantau, dan menganalisis data dari server, jaringan, perangkat, dan sumber lain untuk mendeteksi ancaman dan pelanggaran keamanan, dan untuk menghasilkan peringatan.

otomatisasi respons keamanan

Tindakan yang telah ditentukan dan diprogram yang dirancang untuk secara otomatis merespons atau memulihkan peristiwa keamanan. Otomatisasi ini berfungsi sebagai kontrol keamanan detektif atau responsif yang membantu Anda menerapkan praktik terbaik AWS keamanan. Contoh tindakan respons otomatis termasuk memodifikasi grup keamanan VPC, menambal instans Amazon EC2, atau memutar kredensil.

enkripsi sisi server

Enkripsi data di tujuannya, oleh Layanan AWS yang menerimanya.

kebijakan kontrol layanan (SCP)

Kebijakan yang menyediakan kontrol terpusat atas izin untuk semua akun di organisasi. AWS Organizations SCPs menentukan pagar pembatas atau menetapkan batasan pada tindakan yang dapat didelegasikan oleh administrator kepada pengguna atau peran. Anda dapat menggunakan SCPs daftar izin atau daftar penolakan, untuk menentukan layanan atau tindakan mana yang diizinkan atau dilarang. Untuk informasi selengkapnya, lihat Kebijakan kontrol layanan dalam AWS Organizations dokumentasi.

titik akhir layanan

URL titik masuk untuk file Layanan AWS. Anda dapat menggunakan endpoint untuk terhubung secara terprogram ke layanan target. Untuk informasi selengkapnya, lihat Layanan AWS titik akhir di Referensi Umum AWS.

perjanjian tingkat layanan (SLA)

Perjanjian yang menjelaskan apa yang dijanjikan tim TI untuk diberikan kepada pelanggan mereka, seperti waktu kerja dan kinerja layanan.

indikator tingkat layanan (SLI)

Pengukuran aspek kinerja layanan, seperti tingkat kesalahan, ketersediaan, atau throughputnya.

tujuan tingkat layanan (SLO)

Metrik target yang mewakili kesehatan layanan, yang diukur dengan indikator tingkat layanan.

model tanggung jawab bersama

Model yang menjelaskan tanggung jawab yang Anda bagikan AWS untuk keamanan dan kepatuhan cloud. AWS bertanggung jawab atas keamanan cloud, sedangkan Anda bertanggung jawab atas keamanan di cloud. Untuk informasi selengkapnya, lihat Model tanggung jawab bersama.

SIEM

Lihat informasi keamanan dan sistem manajemen acara.

titik kegagalan tunggal (SPOF)

Kegagalan dalam satu komponen penting dari aplikasi yang dapat mengganggu sistem.

SLA

Lihat perjanjian tingkat layanan.

SLI

Lihat indikator tingkat layanan.

SLO

Lihat tujuan tingkat layanan.

split-and-seed model

Pola untuk menskalakan dan mempercepat proyek modernisasi. Ketika fitur baru dan rilis produk didefinisikan, tim inti berpisah untuk membuat tim produk baru. Ini membantu meningkatkan kemampuan dan layanan organisasi Anda, meningkatkan produktivitas pengembang, dan mendukung inovasi yang cepat. Untuk informasi lebih lanjut, lihat Pendekatan bertahap untuk memodernisasi aplikasi di. AWS Cloud

SPOF

Lihat satu titik kegagalan.

skema bintang

Struktur organisasi database yang menggunakan satu tabel fakta besar untuk menyimpan data transaksional atau terukur dan menggunakan satu atau lebih tabel dimensi yang lebih kecil untuk menyimpan atribut data. Struktur ini dirancang untuk digunakan dalam gudang data atau untuk tujuan intelijen bisnis.

pola ara pencekik

Pendekatan untuk memodernisasi sistem monolitik dengan menulis ulang secara bertahap dan mengganti fungsionalitas sistem sampai sistem warisan dapat dinonaktifkan. Pola ini menggunakan analogi pohon ara yang tumbuh menjadi pohon yang sudah mapan dan akhirnya mengatasi dan menggantikan inangnya. Pola ini diperkenalkan oleh Martin Fowler sebagai cara untuk mengelola risiko saat menulis ulang sistem monolitik. Untuk contoh cara menerapkan pola ini, lihat Memodernisasi layanan web Microsoft ASP.NET (ASMX) lama secara bertahap menggunakan container dan Amazon API Gateway.

subnet

Rentang alamat IP dalam VPC Anda. Subnet harus berada di Availability Zone tunggal.

kontrol pengawasan dan akuisisi data (SCADA)

Di bidang manufaktur, sistem yang menggunakan perangkat keras dan perangkat lunak untuk memantau aset fisik dan operasi produksi.

enkripsi simetris

Algoritma enkripsi yang menggunakan kunci yang sama untuk mengenkripsi dan mendekripsi data.

pengujian sintetis

Menguji sistem dengan cara yang mensimulasikan interaksi pengguna untuk mendeteksi potensi masalah atau untuk memantau kinerja. Anda dapat menggunakan Amazon CloudWatch Synthetics untuk membuat tes ini.

sistem prompt

Teknik untuk memberikan konteks, instruksi, atau pedoman ke LLM untuk mengarahkan perilakunya. Permintaan sistem membantu mengatur konteks dan menetapkan aturan untuk interaksi dengan pengguna.

T

tag

Pasangan nilai kunci yang bertindak sebagai metadata untuk mengatur sumber daya Anda. AWS Tanda membantu Anda mengelola, mengidentifikasi, mengatur, dan memfilter sumber daya. Untuk informasi selengkapnya, lihat Menandai AWS sumber daya Anda.

variabel target

Nilai yang Anda coba prediksi dalam ML yang diawasi. Ini juga disebut sebagai variabel hasil. Misalnya, dalam pengaturan manufaktur, variabel target bisa menjadi cacat produk.

daftar tugas

Alat yang digunakan untuk melacak kemajuan melalui runbook. Daftar tugas berisi ikhtisar runbook dan daftar tugas umum yang harus diselesaikan. Untuk setiap tugas umum, itu termasuk perkiraan jumlah waktu yang dibutuhkan, pemilik, dan kemajuan.

lingkungan uji

Lihat lingkungan.

pelatihan

Untuk menyediakan data bagi model ML Anda untuk dipelajari. Data pelatihan harus berisi jawaban yang benar. Algoritma pembelajaran menemukan pola dalam data pelatihan yang

memetakan atribut data input ke target (jawaban yang ingin Anda prediksi). Ini menghasilkan model ML yang menangkap pola-pola ini. Anda kemudian dapat menggunakan model ML untuk membuat prediksi pada data baru yang Anda tidak tahu targetnya.

gerbang transit

Hub transit jaringan yang dapat Anda gunakan untuk menghubungkan jaringan Anda VPCs dan lokal. Untuk informasi selengkapnya, lihat Apa itu gateway transit dalam AWS Transit Gateway dokumentasi.

alur kerja berbasis batang

Pendekatan di mana pengembang membangun dan menguji fitur secara lokal di cabang fitur dan kemudian menggabungkan perubahan tersebut ke cabang utama. Cabang utama kemudian dibangun untuk pengembangan, praproduksi, dan lingkungan produksi, secara berurutan.

akses tepercaya

Memberikan izin ke layanan yang Anda tentukan untuk melakukan tugas di organisasi Anda di dalam AWS Organizations dan di akunnya atas nama Anda. Layanan tepercaya menciptakan peran terkait layanan di setiap akun, ketika peran itu diperlukan, untuk melakukan tugas manajemen untuk Anda. Untuk informasi selengkapnya, lihat Menggunakan AWS Organizations dengan AWS layanan lain dalam AWS Organizations dokumentasi.

penyetelan

Untuk mengubah aspek proses pelatihan Anda untuk meningkatkan akurasi model ML. Misalnya, Anda dapat melatih model ML dengan membuat set pelabelan, menambahkan label, dan kemudian mengulangi langkah-langkah ini beberapa kali di bawah pengaturan yang berbeda untuk mengoptimalkan model.

tim dua pizza

Sebuah DevOps tim kecil yang bisa Anda beri makan dengan dua pizza. Ukuran tim dua pizza memastikan peluang terbaik untuk berkolaborasi dalam pengembangan perangkat lunak.

U

waswas

Sebuah konsep yang mengacu pada informasi yang tidak tepat, tidak lengkap, atau tidak diketahui yang dapat merusak keandalan model ML prediktif. Ada dua jenis ketidakpastian: ketidakpastian epistemik disebabkan oleh data yang terbatas dan tidak lengkap, sedangkan

ketidakpastian aleatorik disebabkan oleh kebisingan dan keacakan yang melekat dalam data. Untuk informasi lebih lanjut, lihat panduan Mengukur ketidakpastian dalam sistem pembelajaran mendalam.

tugas yang tidak terdiferensiasi

Juga dikenal sebagai angkat berat, pekerjaan yang diperlukan untuk membuat dan mengoperasikan aplikasi tetapi itu tidak memberikan nilai langsung kepada pengguna akhir atau memberikan keunggulan kompetitif. Contoh tugas yang tidak terdiferensiasi termasuk pengadaan, pemeliharaan, dan perencanaan kapasitas.

lingkungan atas

Lihat lingkungan.

V

menyedot debu

Operasi pemeliharaan database yang melibatkan pembersihan setelah pembaruan tambahan untuk merebut kembali penyimpanan dan meningkatkan kinerja.

kendali versi

Proses dan alat yang melacak perubahan, seperti perubahan kode sumber dalam repositori.

Peering VPC

Koneksi antara dua VPCs yang memungkinkan Anda untuk merutekan lalu lintas dengan menggunakan alamat IP pribadi. Untuk informasi selengkapnya, lihat Apa itu peering VPC di dokumentasi VPC Amazon.

kerentanan

Kelemahan perangkat lunak atau perangkat keras yang membahayakan keamanan sistem.

W

cache hangat

Cache buffer yang berisi data terkini dan relevan yang sering diakses. Instance database dapat membaca dari cache buffer, yang lebih cepat daripada membaca dari memori utama atau disk.

data hangat

Data yang jarang diakses. Saat menanyakan jenis data ini, kueri yang cukup lambat biasanya dapat diterima.

fungsi jendela

Fungsi SQL yang melakukan perhitungan pada sekelompok baris yang berhubungan dengan catatan saat ini. Fungsi jendela berguna untuk memproses tugas, seperti menghitung rata-rata bergerak atau mengakses nilai baris berdasarkan posisi relatif dari baris saat ini.

beban kerja

Kumpulan sumber daya dan kode yang memberikan nilai bisnis, seperti aplikasi yang dihadapi pelanggan atau proses backend.

aliran kerja

Grup fungsional dalam proyek migrasi yang bertanggung jawab atas serangkaian tugas tertentu. Setiap alur kerja independen tetapi mendukung alur kerja lain dalam proyek. Misalnya, alur kerja portofolio bertanggung jawab untuk memprioritaskan aplikasi, perencanaan gelombang, dan mengumpulkan metadata migrasi. Alur kerja portofolio mengirimkan aset ini ke alur kerja migrasi, yang kemudian memigrasikan server dan aplikasi.

CACING

Lihat menulis sekali, baca banyak.

WQF

Lihat AWS Kerangka Kualifikasi Beban Kerja.

tulis sekali, baca banyak (WORM)

Model penyimpanan yang menulis data satu kali dan mencegah data dihapus atau dimodifikasi. Pengguna yang berwenang dapat membaca data sebanyak yang diperlukan, tetapi mereka tidak dapat mengubahnya. Infrastruktur penyimpanan data ini dianggap tidak dapat diubah.

Z

eksploitasi zero-day

Serangan, biasanya malware, yang memanfaatkan kerentanan zero-day.

kerentanan zero-day

Cacat atau kerentanan yang tak tanggung-tanggung dalam sistem produksi. Aktor ancaman dapat menggunakan jenis kerentanan ini untuk menyerang sistem. Pengembang sering menyadari kerentanan sebagai akibat dari serangan tersebut.

bisikan zero-shot

Memberikan LLM dengan instruksi untuk melakukan tugas tetapi tidak ada contoh (tembakan) yang dapat membantu membimbingnya. LLM harus menggunakan pengetahuan pra-terlatih untuk menangani tugas. Efektivitas bidikan nol tergantung pada kompleksitas tugas dan kualitas prompt. Lihat juga beberapa bidikan yang diminta.

aplikasi zombie

Aplikasi yang memiliki CPU rata-rata dan penggunaan memori di bawah 5 persen. Dalam proyek migrasi, adalah umum untuk menghentikan aplikasi ini.

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.