Layanan Terkelola untuk Panduan Pengembang Apache Flink

Layanan Terkelola untuk Apache Flink

Layanan Terkelola untuk Apache Flink: Layanan Terkelola untuk Panduan Pengembang Apache Flink

Table of Contents

Amazon Managed Service untuk Apache Flink sebelumnya dikenal sebagai Amazon Kinesis Data Analytics untuk Apache Flink.

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

Apa itu Amazon Managed Service untuk Apache Flink?

Dengan Amazon Managed Service untuk Apache Flink, Anda dapat menggunakan Java, Scala, Python, atau SQL untuk memproses dan menganalisis data streaming. Layanan ini memungkinkan Anda untuk membuat dan menjalankan kode terhadap sumber streaming dan sumber statis untuk melakukan analitik deret waktu, memberi umpan dasbor waktu nyata, dan metrik.

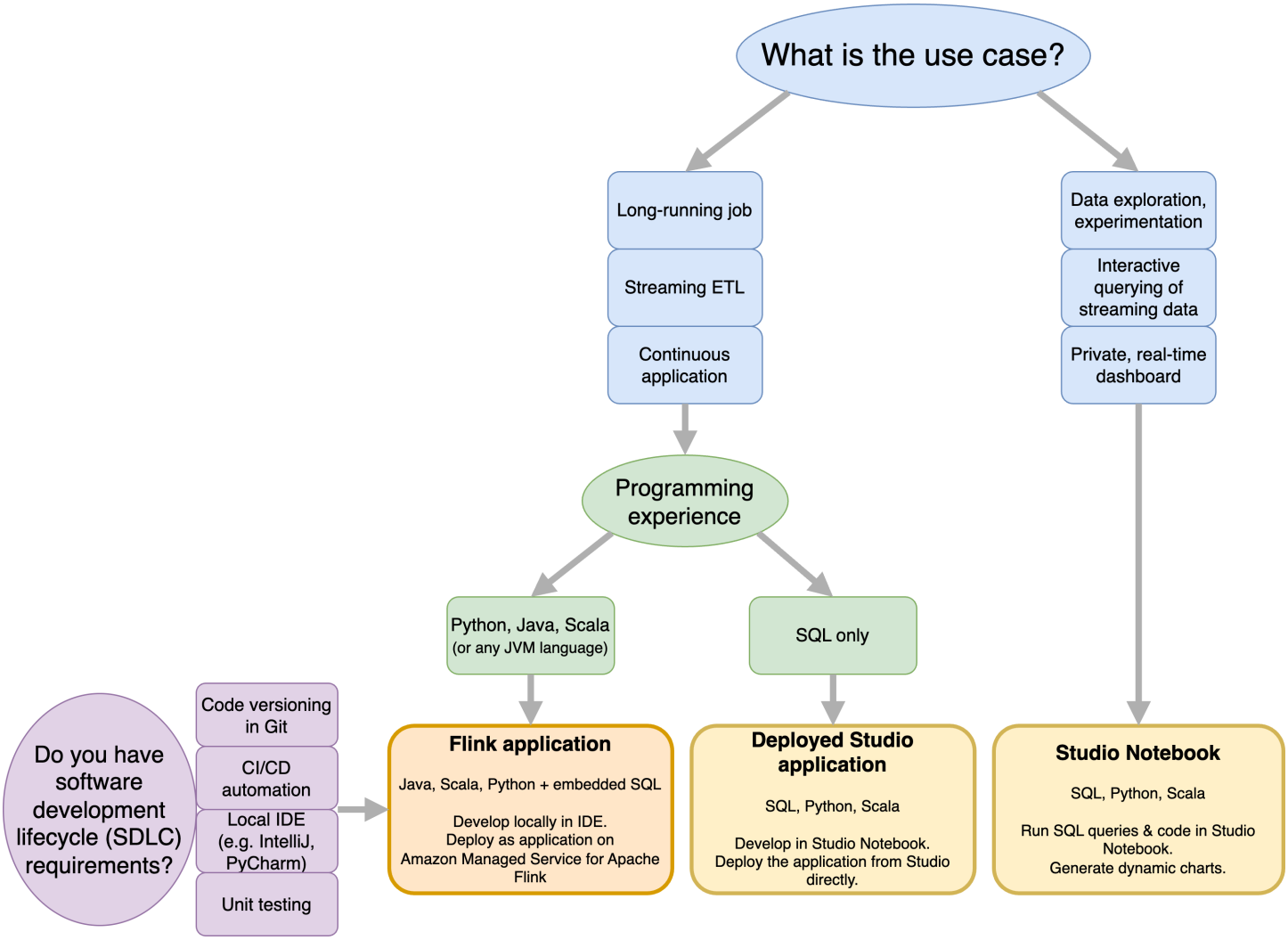
Anda dapat membangun aplikasi dengan bahasa pilihan Anda di Managed Service for Apache Flink menggunakan pustaka open-source berdasarkan Apache Flink. Apache Flink adalah kerangka kerja dan mesin populer untuk memproses aliran data.

Managed Service for Apache Flink menyediakan infrastruktur dasar untuk aplikasi Apache Flink Anda. Ini menangani kemampuan inti seperti penyediaan sumber daya komputasi, ketahanan failover AZ, komputasi paralel, penskalaan otomatis, dan pencadangan aplikasi (diimplementasikan sebagai pos pemeriksaan dan snapshot). Anda dapat menggunakan fitur pemrograman Flink tingkat tinggi (seperti operator, fungsi, sumber, dan sink) dengan cara yang sama seperti Anda menggunakannya ketika meng-host infrastruktur Flink Anda sendiri.

Memilih Managed Service untuk Apache Flink atau Managed Service untuk Apache Flink Studio

Anda memiliki dua opsi untuk menjalankan pekerjaan Flink Anda dengan Amazon Managed Service untuk Apache Flink. Dengan Managed Service for Apache Flink, Anda membangun aplikasi Flink di Java, Scala, atau Python (dan tertanam SQL) menggunakan IDE pilihan Anda dan Apache Flink Datastream atau Table API. Dengan Managed Service for Apache Flink Studio, Anda dapat secara interaktif menanyakan aliran data secara real time dan dengan mudah membangun dan menjalankan aplikasi pemrosesan aliran menggunakan SQL, Python, dan Scala standar.

Anda dapat memilih metode mana yang paling sesuai dengan kasus penggunaan Anda. Jika Anda tidak yakin, bagian ini akan menawarkan panduan tingkat tinggi untuk membantu Anda.



Sebelum memutuskan apakah akan menggunakan Amazon Managed Service untuk Apache Flink atau Amazon Managed Service untuk Apache Flink Studio Anda harus mempertimbangkan kasus penggunaan Anda.

Jika Anda berencana untuk mengoperasikan aplikasi yang berjalan lama yang akan melakukan beban kerja seperti Streaming ETL atau Aplikasi Berkelanjutan, Anda harus mempertimbangkan untuk menggunakan Layanan Terkelola untuk Apache Flink. Ini karena Anda dapat membuat aplikasi Flink Anda menggunakan API Flink langsung di IDE pilihan Anda. Mengembangkan secara lokal dengan IDE Anda juga memastikan Anda dapat memanfaatkan proses dan perkakas umum siklus hidup pengembangan perangkat lunak (SDLC) seperti pembuatan versi kode di Git, otomatisasi CI/CD, atau pengujian unit.

Jika Anda tertarik dengan eksplorasi data ad-hoc, ingin menanyakan data streaming secara interaktif, atau membuat dasbor real-time pribadi, Managed Service for Apache Flink Studio akan membantu

Anda memenuhi tujuan ini hanya dengan beberapa klik. Pengguna yang akrab dengan SQL dapat mempertimbangkan untuk menerapkan aplikasi yang berjalan lama dari Studio secara langsung.

> **Note**
>
> Anda dapat mempromosikan notebook Studio Anda ke aplikasi yang sudah berjalan lama. Namun, jika Anda ingin mengintegrasikan dengan alat SDLC Anda seperti pembuatan versi kode pada Git dan otomatisasi CI/CD, atau teknik seperti pengujian unit, kami merekomendasikan Layanan Terkelola untuk Apache Flink menggunakan IDE pilihan Anda.

Memilih Apache Flink API mana yang akan digunakan dalam Managed Service untuk Apache Flink

Anda dapat membangun aplikasi menggunakan Java, Python, dan Scala di Managed Service untuk Apache Flink menggunakan Apache Flink API dalam IDE pilihan Anda. Anda dapat menemukan panduan tentang cara membangun aplikasi menggunakan Flink Datastream dan Table API dalam dokumentasi. Anda dapat memilih bahasa tempat Anda membuat aplikasi Flink dan API yang Anda gunakan untuk memenuhi kebutuhan aplikasi dan operasi Anda. Jika Anda tidak yakin, bagian ini memberikan panduan tingkat tinggi untuk membantu Anda.

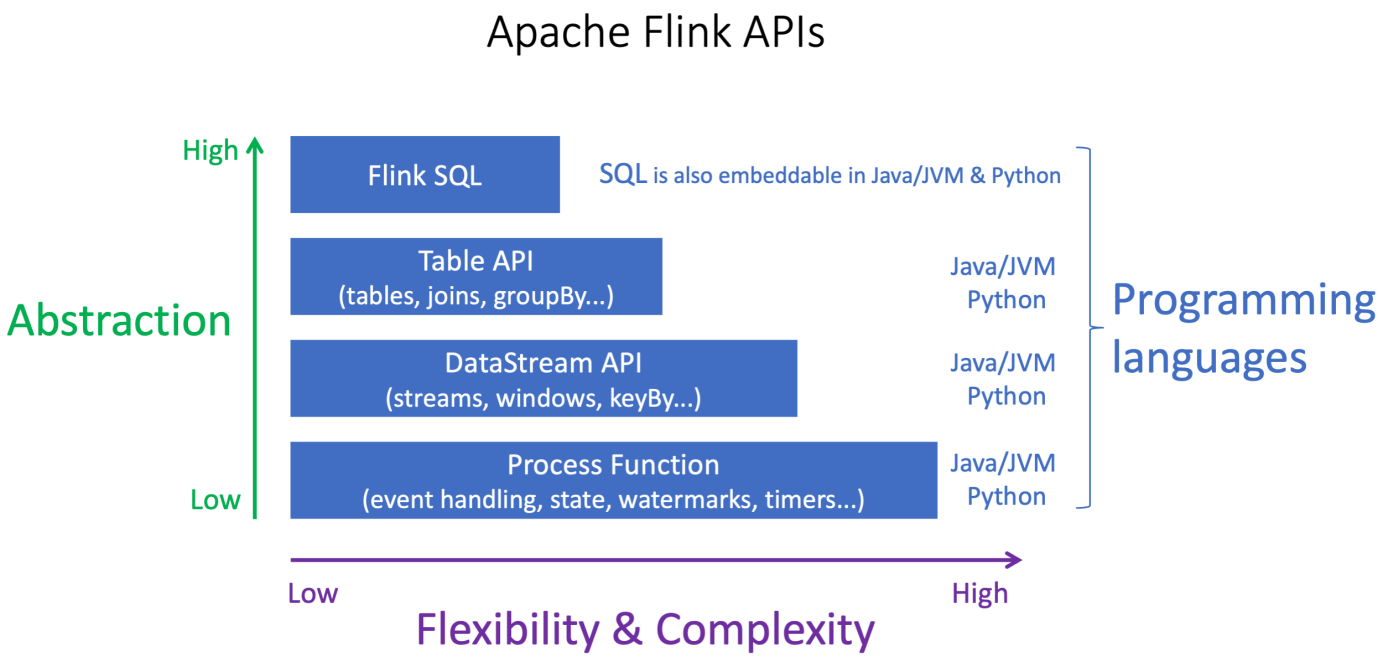
Memilih API Flink

Apache Flink API memiliki tingkat abstraksi yang berbeda yang dapat mempengaruhi bagaimana Anda memutuskan untuk membangun aplikasi Anda. Mereka ekspresif dan fleksibel dan dapat digunakan bersama untuk membangun aplikasi Anda. Anda tidak harus menggunakan hanya satu Flink API. Anda dapat mempelajari lebih lanjut tentang API Flink di dokumentasi Apache Flink.

Flink menawarkan empat tingkat abstraksi API: Flink SQL, Table API, DataStream API, dan Process Function, yang digunakan bersama dengan API. DataStream Ini semua didukung di Amazon Managed Service untuk Apache Flink. Dianjurkan untuk memulai dengan tingkat abstraksi yang lebih tinggi jika memungkinkan, namun beberapa fitur Flink hanya tersedia dengan Datastream API di mana Anda dapat membuat aplikasi Anda di Java, Python, atau Scala. Anda harus mempertimbangkan untuk menggunakan Datastream API jika:

- Anda memerlukan kontrol berbutir halus atas negara
- Anda ingin memanfaatkan kemampuan untuk memanggil database eksternal atau titik akhir secara asinkron (misalnya untuk inferensi)

- Anda ingin menggunakan pengatur waktu khusus



> **Note**
>
> Memilih bahasa dengan Datastream API:
>
> - SQL dapat disematkan dalam aplikasi Flink apa pun, terlepas dari bahasa pemrograman yang dipilih.
> - Jika Anda berencana untuk menggunakan DataStream API, tidak semua konektor didukung dengan Python.
> - Jika Anda membutuhkan latensi rendah/throughput tinggi, Anda harus mempertimbangkan Java/Scala terlepas dari API.
> - Jika Anda berencana untuk menggunakan Async IO di Process Functions API, Anda harus menggunakan Java.

Memulai

Anda dapat memulai dengan membuat Layanan Terkelola untuk aplikasi Apache Flink yang terus membaca dan memproses data streaming. Selanjutnya, tulis kode Anda menggunakan IDE pilihan

Anda, dan uji dengan data streaming langsung. Anda juga dapat mengonfigurasi tujuan di mana Anda ingin Layanan Terkelola untuk Apache Flink untuk mengirim hasilnya.

Sebaiknya mulai dengan membaca bagian berikut:

- Layanan Terkelola untuk Apache Flink: Cara Kerjanya
- Memulai dengan Amazon Managed Service untuk Apache Flink (API) DataStream

Secara altenatif, Anda dapat memulai dengan membuat Managed Service for Apache Flink Studio notebook yang memungkinkan Anda untuk secara interaktif menanyakan aliran data secara real time, dan dengan mudah membangun dan menjalankan aplikasi pemrosesan aliran menggunakan SQL standar, Python, dan Scala. Dengan beberapa klikAWS Management Console, Anda dapat meluncurkan notebook tanpa server untuk menanyakan aliran data dan mendapatkan hasil dalam hitungan detik. Sebaiknya mulai dengan membaca bagian berikut:

- Menggunakan notebook Studio dengan Managed Service untuk Apache Flink
- Membuat notebook Studio

Layanan Terkelola untuk Apache Flink: Cara Kerjanya

Managed Service for Apache Flink adalah layanan Amazon yang dikelola sepenuhnya yang memungkinkan Anda menggunakan aplikasi Apache Flink untuk memproses data streaming.

Memprogram Aplikasi Apache Flink Anda

Aplikasi Apache Flink adalah aplikasi Java atau Scala yang dibuat dengan kerangka kerja Apache Flink. Anda menulis dan membangun aplikasi Apache Flink Anda secara lokal.

Aplikasi terutama menggunakan salah satuDataStreamAPIatauTabel API. API Apache Flink lainnya juga tersedia untuk Anda gunakan, tetapi API tersebut kurang umum digunakan dalam membangun aplikasi streaming.

Fitur dari dua API adalah sebagai berikut:

API DataStream

Apache FlinkDataStreamModel pemrograman API didasarkan pada dua komponen:

- Aliran data: Representasi terstruktur dari aliran catatan data yang berkelanjutan.
- Operator transformasi: Membawa satu atau beberapa aliran data sebagai input, dan menghasilkan satu atau beberapa aliran data sebagai output.

Aplikasi yang dibuat denganDataStreamAPI lakukan hal berikut:

- Baca data dari Sumber Data (seperti aliran Kinesis atau topik Amazon MSK).
- Terapkan transformasi ke data, seperti penyaringan, agregasi, atau pengayaan.
- Tulis data yang diubah ke Sink Data.

Aplikasi yang menggunakanDataStreamAPI dapat ditulis dalam Java atau Scala, dan dapat dibaca dari aliran data Kinesis, topik MSK Amazon, atau sumber kustom.

Aplikasi Anda memproses data menggunakan konektor. Apache Flink menggunakan tipe konektor berikut:

- Source (Sumber) : Konektor yang digunakan untuk membaca data eksternal.

- Sink: Konektor yang digunakan untuk menulis ke lokasi eksternal.
- Operator: Konektor yang digunakan untuk memproses data dalam aplikasi.

Aplikasi yang khas terdiri dari setidaknya satu aliran data dengan sumber, aliran data dengan satu atau beberapa operator, dan setidaknya satu data sink.

Untuk informasi lebih lanjut tentang menggunakanDataStreamAPI, lihatAPI DataStream.

Tabel API

Model pemrograman API Tabel Apache Flink didasarkan pada komponen berikut:

- Lingkungan Tabel: Antarmuka untuk data yang mendasari yang Anda gunakan untuk membuat dan meng-host satu atau beberapa tabel.
- Tabel: Objek yang menyediakan akses ke tabel atau tampilan SQL.
- Sumber Tabel: Digunakan untuk membaca data dari sumber eksternal, seperti topik Amazon MSK.
- Fungsi Tabel: Kueri SQL atau panggilan API yang digunakan untuk mengubah data.
- Sink Tabel: Digunakan untuk menulis data ke lokasi eksternal, seperti bucket Amazon S3.

Aplikasi yang dibuat dengan API Tabel melakukan hal berikut:

- Buat `TableEnvironment` dengan menghubungkan ke `Table Source`.
- Buat tabel di `TableEnvironment` menggunakan kueri SQL atau fungsi API Tabel.
- Jalankan kueri pada tabel menggunakan API Tabel atau SQL
- Terapkan transformasi pada hasil kueri menggunakan Fungsi Tabel atau kueri SQL.
- Tulis hasil kueri atau fungsi ke `Table Sink`.

Aplikasi yang menggunakan API Tabel dapat ditulis di Java atau Scala, dan dapat mengkueri data menggunakan panggilan API atau kueri SQL.

Untuk informasi selengkapnya tentang penggunaan API Tabel, lihat Tabel API.

Membuat Layanan Terkelola Anda untuk Aplikasi Apache Flink

Managed Service untuk Apache Flink adalahAWSlayanan yang menciptakan lingkungan untuk hosting aplikasi Apache Flink Anda dan menyediakannya dengan pengaturan berikut:

- Properti Runtime: Parameter yang dapat Anda berikan ke aplikasi Anda. Anda dapat mengubah parameter ini tanpa mengompilasi ulang kode aplikasi Anda.
- Toleransi Kesalahan: Cara aplikasi Anda pulih dari gangguan dan mulai ulang.
- Pencatatan dan Pemantauan: Bagaimana aplikasi Anda mencatat peristiwa keCloudWatchLog.
- Penskalaan: Cara aplikasi Anda menyediakan sumber daya komputasi.

Anda membuat Layanan Terkelola untuk aplikasi Apache Flink menggunakan konsol atauAWS CLI. Untuk mulai membuat Layanan Terkelola untuk aplikasi Apache Flink, lihatMemulai (DataStream API).

Membuat Layanan Terkelola untuk Apache Flink Application

Topik ini berisi informasi tentang membuat Layanan Terkelola untuk Apache Flink.

Topik ini berisi bagian-bagian berikut:

- Membangun Layanan Terkelola Anda untuk Kode Aplikasi Apache Flink
- Membuat Layanan Terkelola Anda untuk Apache Flink Application
- Memulai Layanan Terkelola Anda untuk Aplikasi Apache Flink
- Memverifikasi Layanan Terkelola Anda untuk Aplikasi Apache Flink

Membangun Layanan Terkelola Anda untuk Kode Aplikasi Apache Flink

Bagian ini menjelaskan komponen yang Anda gunakan untuk membangun kode aplikasi untuk Layanan Terkelola untuk aplikasi Apache Flink Anda.

Sebaiknya gunakan versi terbaru Apache Flink yang didukung untuk kode aplikasi Anda. Versi terbaru Apache Flink yang Managed Service untuk Apache Flink mendukung adalah 1.15.2. Untuk informasi tentang memutakhirkan Layanan Terkelola untuk aplikasi Apache Flink, lihat. Meningkatkan Aplikasi

Anda membangun kode aplikasi Anda menggunakan Apache Maven. Proyek Apache Maven menggunakan file `pom.xml` untuk menentukan versi komponen yang digunakan.

> **Note**
>
> Layanan Terkelola untuk Apache Flink mendukung file JAR hingga ukuran 512 MB. Jika Anda menggunakan file JAR lebih besar dari ini, aplikasi Anda akan gagal dimulai.

Gunakan versi komponen berikut untuk Managed Service untuk aplikasi Apache Flink:

Komponen	Versi
Java	11 (direkomendasikan)
Skala	Lihat catatan decoupling Scala di bawah ini
Layanan Terkelola untuk Apache Flink Runtime () aws-kinesisanalytics-runtime	1.2.0
AWSKonektor Kinesis () flink-connector-kinesis	1.15.2
Apache Beam (Khusus Aplikasi Beam)	2.33.0, dengan Jackson versi 2.12.2

Dimulai dengan versi 1.15, Flink gratis Scala. Aplikasi sekarang dapat menggunakan Java API dari versi Scala apa pun. Anda perlu menggabungkan pustaka standar Scala pilihan Anda ke dalam aplikasi Scala Anda.

Untuk contoh `pom.xml` file untuk aplikasi Managed Service for Apache Flink yang menggunakan Apache Flink versi 1.15.2, lihat Managed Service for Apache Flink Getting Started Application.

Untuk informasi tentang membuat Layanan Terkelola untuk aplikasi Apache Flink yang menggunakan Apache Beam, lihat. Menggunakan Apache Beam

Menentukan Versi Apache Flink Aplikasi Anda

Saat menggunakan Managed Service for Apache Flink Runtime versi 1.1.0 dan yang lebih baru, Anda menentukan versi Apache Flink yang digunakan aplikasi Anda saat Anda mengkompilasi aplikasi Anda. Anda menyediakan versi Apache Flink dengan parameter `-Dflink.version` sebagai berikut:

```
mvn package -Dflink.version=1.15.3
```

Untuk membangun aplikasi dengan versi Apache Flink yang lebih lama, lihat Versi Sebelumnya.

Membuat Layanan Terkelola Anda untuk Apache Flink Application

Setelah Anda membangun kode aplikasi Anda, Anda melakukan hal berikut untuk membuat Layanan Terkelola untuk aplikasi Apache Flink Anda:

- Unggah kode Aplikasi Anda: Unggah kode aplikasi Anda ke bucket Amazon S3. Anda menentukan nama bucket S3 dan nama objek kode aplikasi Anda ketika membuat aplikasi Anda. Untuk tutorial yang menunjukkan cara mengunggah kode aplikasi Anda, lihat the section called “Unggah Kode Java Streaming Apache Flink” di tutorial Memulai (DataStream API).
- Buat Layanan Terkelola untuk aplikasi Apache Flink Anda: Gunakan salah satu metode berikut untuk membuat Layanan Terkelola untuk aplikasi Apache Flink Anda:
  - Buat Layanan Terkelola untuk aplikasi Apache Flink menggunakan AWS konsol: Anda dapat membuat dan mengonfigurasi aplikasi menggunakan konsol. AWS

    Saat Anda membuat aplikasi menggunakan konsol, sumber daya dependen aplikasi Anda (seperti aliran CloudWatch Log, peran IAM, dan kebijakan IAM) akan dibuat untuk Anda.

    Saat Anda membuat aplikasi menggunakan konsol, Anda menentukan versi Apache Flink yang digunakan aplikasi Anda dengan memilihnya dari pull-down pada halaman Managed Service for Apache Flink - Create application.

    Untuk tutorial tentang cara menggunakan konsol untuk membuat aplikasi, lihat the section called “Buat dan Jalankan Aplikasi (Konsol)” di tutorial Memulai (DataStream API).
  - Buat Layanan Terkelola Anda untuk aplikasi Apache Flink menggunakan AWS CLI: Anda dapat membuat dan mengkonfigurasi aplikasi Anda menggunakan CLI. AWS

    Saat Anda membuat aplikasi menggunakan CLI, Anda juga harus membuat sumber daya dependen aplikasi Anda (seperti aliran CloudWatch Log, peran IAM, dan kebijakan IAM) secara manual.

    Ketika Anda membuat aplikasi Anda menggunakan CLI, Anda menentukan versi Apache Flink yang digunakan aplikasi Anda menggunakan parameter `RuntimeEnvironment` dari tindakan `CreateApplication`.

    Untuk tutorial tentang cara menggunakan CLI untuk membuat aplikasi, lihat the section called “Buat dan Jalankan Aplikasi Menggunakan CLI” di tutorial Memulai (DataStream API).

> **Note**
>
> Anda tidak dapat mengubah `RuntimeEnvironment` dari aplikasi yang sudah ada. Jika Anda perlu mengubah `RuntimeEnvironment` dari aplikasi yang ada, Anda harus menghapus aplikasi dan membuatnya lagi.

Memulai Layanan Terkelola Anda untuk Aplikasi Apache Flink

Setelah Anda membangun kode aplikasi Anda, mengunggahnya ke S3, dan membuat Layanan Terkelola untuk aplikasi Apache Flink, Anda kemudian memulai aplikasi Anda. Memulai Layanan Terkelola untuk aplikasi Apache Flink biasanya memakan waktu beberapa menit.

Gunakan salah satu metode berikut untuk memulai aplikasi Anda:

- Mulai Layanan Terkelola untuk aplikasi Apache Flink menggunakan AWS konsol: Anda dapat menjalankan aplikasi Anda dengan memilih Jalankan pada halaman aplikasi Anda di AWS konsol.
- Mulai Layanan Terkelola untuk aplikasi Apache Flink menggunakan AWS API: Anda dapat menjalankan aplikasi Anda menggunakan tindakan. StartApplication

Memverifikasi Layanan Terkelola Anda untuk Aplikasi Apache Flink

Anda dapat memverifikasi bahwa aplikasi Anda bekerja dengan cara berikut:

- Menggunakan CloudWatch Log: Anda dapat menggunakan Wawasan CloudWatch CloudWatch Log dan Log untuk memverifikasi bahwa aplikasi Anda berjalan dengan benar. Untuk informasi tentang menggunakan CloudWatch Log dengan Layanan Terkelola untuk aplikasi Apache Flink, lihat. Pencatatan dan Pemantauan
- Menggunakan CloudWatch Metrik: Anda dapat menggunakan CloudWatch Metrik untuk memantau aktivitas aplikasi, atau aktivitas dalam sumber daya yang digunakan aplikasi untuk input atau output (seperti aliran Kinesis, aliran Firehose Data Kinesis, atau bucket Amazon S3.) Untuk informasi selengkapnya tentang CloudWatch metrik, lihat Bekerja dengan Metrik di CloudWatch Panduan Pengguna Amazon.
- Pemantauan Lokasi Output: Jika aplikasi Anda menulis output ke lokasi (seperti bucket atau basis data Amazon S3), Anda dapat memantau lokasi tersebut untuk data tertulis.

Menjalankan Layanan Terkelola untuk Apache Flink Application

Topik ini berisi informasi tentang menjalankan Managed Service untuk Apache Flink.

Saat Anda menjalankan aplikasi Managed Service for Apache Flink, layanan akan membuat pekerjaan Apache Flink. Pekerjaan Apache Flink adalah siklus hidup eksekusi Layanan Terkelola untuk aplikasi Apache Flink Anda. Eksekusi tugas, dan sumber daya yang digunakannya, dikelola oleh Manajer Tugas. Manajer Tugas memisahkan eksekusi aplikasi ke dalam tugas-tugas. Setiap tugas dikelola oleh Manajer Tugas. Ketika Anda memantau performa aplikasi, Anda dapat memeriksa performa masing-masing Manajer Tugas, atau Manajer Tugas secara keseluruhan.

Untuk informasi selengkapnya tentang tugas Apache Flink, lihat Tugas dan Penjadwalan di Dokumentasi Apache Flink.

Aplikasi dan Status Tugas

Aplikasi Anda dan tugas aplikasi memiliki status eksekusi saat ini:

- Status aplikasi: Aplikasi Anda memiliki status saat ini yang menggambarkan fase eksekusi. Status aplikasi mencakup hal-hal berikut:
  - Status aplikasi stabil: Aplikasi Anda biasanya tetap berada dalam status ini hingga Anda membuat perubahan status:
    - READY (SIAP): Aplikasi baru atau yang dihentikan berada dalam status READY (SIAP) hingga Anda menjalankannya.
    - RUNNING (BERJALAN): Aplikasi yang telah berhasil dimulai berada dalam status RUNNING (BERJALAN).
  - Status aplikasi sementara: Aplikasi dalam status ini biasanya dalam proses transisi ke status lain. Jika aplikasi tetap dalam status sementara untuk jangka waktu yang lama, Anda dapat menghentikan aplikasi menggunakanStopApplicationtindakan dengan`Force`parameter diatur `ketrue`. Status ini mencakup hal berikut:
    - `STARTING:`Terjadi setelahStartApplicationtindakan. Aplikasi ini bertransisi dari status `READY` ke `RUNNING`.
    - `STOPPING:`Terjadi setelahStopApplicationtindakan. Aplikasi ini bertransisi dari status `RUNNING` ke `READY`.
    - `DELETING:`Terjadi setelahDeleteApplicationtindakan. Aplikasi sedang dalam proses penghapusan.

- `UPDATING`: Terjadi setelah UpdateApplication tindakan. Aplikasi memperbarui, dan akan bertransisi kembali ke status `RUNNING` atau `READY`.
- `AUTOSCALING`: Aplikasi ini memiliki `AutoScalingEnabled` properti dari ParallelismConfiguration diatur `ke true`, dan layanan ini meningkatkan paralelisme aplikasi. Ketika aplikasi dalam status ini, satu-satunya tindakan API valid yang dapat Anda gunakan adalah StopApplication tindakan dengan `Force` parameter diatur `ke true`. Untuk informasi tentang penskalaan otomatis, lihat Penskalaan Otomatis.
- `FORCE_STOPPING`: Terjadi setelah StopApplication Tindakan disebut dengan `Force` parameter diatur `ke true`. Aplikasi sedang dalam proses penghentian paksa. Aplikasi bertransisi dari status `STARTING`, `UPDATING`, `STOPPING`, atau `AUTOSCALING` ke status `READY`.
- `ROLLING_BACK`: Terjadi setelah RollbackApplication tindakan disebut. Aplikasi sedang dalam proses dikembalikan ke versi sebelumnya. Aplikasi bertransisi dari status `UPDATING` atau `AUTOSCALING` ke status `RUNNING`.
- `ROLLED_BACK`: Ketika Anda berhasil mengembalikan aplikasi, ini menjadi status versi yang Anda kembalikan. Untuk informasi tentang memutar kembali aplikasi, lihat RollbackApplication.
- `MAINTENANCE`: Terjadi saat Managed Service for Apache Flink menerapkan patch ke aplikasi Anda. Untuk informasi selengkapnya, lihat Pemeliharaan.

Anda dapat memeriksa status aplikasi Anda menggunakan konsol, atau dengan menggunakan DescribeApplication tindakan.

- Job status (Status tugas): Saat aplikasi Anda berada dalam status `RUNNING`, tugas Anda memiliki status yang menggambarkan fase eksekusi saat ini. Tugas dimulai dalam status `CREATED`, lalu meneruskan ke status `RUNNING` ketika sudah dimulai. Jika kondisi kesalahan terjadi, aplikasi Anda memasuki status berikut:
  - Untuk aplikasi yang menggunakan Apache Flink 1.11 dan yang lebih baru, aplikasi Anda memasuki status `RESTARTING`.
  - Untuk aplikasi yang menggunakan Apache Flink 1.8 dan sebelumnya, aplikasi Anda memasuki status `FAILING`.

  Aplikasi selanjutnya meneruskan ke status `RESTARTING` atau `FAILED`, bergantung pada apakah tugas dapat dimulai ulang.

  Anda dapat memeriksa status pekerjaan dengan memeriksa aplikasi Anda CloudWatch log untuk perubahan status.

Beban kerja batch

Layanan Terkelola untuk Apache Flink mendukung menjalankan beban kerja batch Apache Flink. Dalam pekerjaan batch, ketika pekerjaan Apache Flink sampai keSELESAIstatus, Layanan Terkelola untuk status aplikasi Apache Flink diatur keSEDIA. Untuk informasi selengkapnya tentang status pekerjaan Flink, lihatPekerjaan dan Penjadwalan.

Sumber Daya Aplikasi

Bagian ini menjelaskan sumber daya sistem yang digunakan aplikasi Anda. Memahami bagaimana Layanan Terkelola untuk penyediaan dan penggunaan sumber daya Apache Flink akan membantu Anda merancang, membuat, dan mempertahankan Layanan Terkelola yang berkinerja dan stabil untuk aplikasi Apache Flink.

Layanan Terkelola untuk Sumber Daya Aplikasi Apache Flink

Managed Service untuk Apache Flink adalahAWSlayanan yang menciptakan lingkungan untuk hosting aplikasi Apache Flink Anda. Layanan Terkelola untuk layanan Apache Flink menyediakan sumber daya menggunakan unit yang disebutUnit Pengolahan Kinesis (KPU).

Satu KPU mewakili sumber daya sistem berikut:

- Satu inti CPU
- 4 GB memori, dengan satu GB adalah memori asli dan tiga GB adalah memori timbunan
- 50 GB ruang disk

KPU menjalankan aplikasi dalam unit eksekusi berbeda yang disebut tugas dan subtugas. Anda bisa menganggap subtugas seperti utas.

Jumlah KPU yang tersedia untuk aplikasi sama dengan pengaturan `Parallelism` aplikasi, dibagi dengan pengaturan `ParallelismPerKPU` aplikasi.

Untuk informasi selengkapnya tentang paralelisme aplikasi, lihat Penskalaan.

Sumber Daya Aplikasi Apache Flink

Lingkungan Apache Flink mengalokasikan sumber daya untuk aplikasi Anda menggunakan unit yang disebut slot tugas. Ketika Managed Service untuk Apache Flink mengalokasikan sumber daya untuk aplikasi Anda, ia menetapkan satu atau lebih slot tugas Apache Flink ke satu KPU. Jumlah slot

yang ditetapkan ke satu KPU sama dengan pengaturan `ParallelismPerKPU` aplikasi Anda. Untuk informasi selengkapnya tentang slot tugas, lihat Penjadwalan Tugas di Dokumentasi Apache Flink.

Paralelisme Operator

Anda dapat mengatur jumlah maksimum subtugas yang dapat digunakan operator. Nilai ini disebut Paralelisme Operator. Secara default, paralelisme dari setiap operator dalam aplikasi Anda adalah sama dengan paralelisme aplikasi. Artinya, setiap operator dalam aplikasi Anda secara default dapat menggunakan semua subtugas yang tersedia dalam aplikasi jika diperlukan.

Anda dapat mengatur paralelisme operator dalam aplikasi Anda menggunakan metode `setParallelism`. Dengan menggunakan metode ini, Anda dapat mengontrol jumlah subtugas yang dapat digunakan setiap operator pada satu waktu.

Untuk informasi selengkapnya tentang jaringan operator, lihat Rantai tugas dan grup sumber daya di Dokumentasi Apache Flink.

Rantai Operator

Biasanya, setiap operator menggunakan subtugas terpisah untuk mengeksekusi, tetapi jika beberapa operator selalu mengeksekusi secara berurutan, runtime dapat menetapkannya ke tugas yang sama. Proses ini disebut Rantai Operator.

Beberapa operator berurutan dapat dirantai menjadi satu tugas jika semuanya beroperasi pada data yang sama. Berikut adalah beberapa kriteria yang diperlukan agar hal ini benar:

- Operator melakukan penerusan sederhana 1 ke 1.
- Operator semuanya memiliki paralelisme operator yang sama.

Ketika aplikasi Anda merantai operator menjadi satu subtugas, hal ini menghemat sumber daya sistem, karena layanan tidak perlu melakukan operasi jaringan dan mengalokasikan subtugas untuk setiap operator. Untuk menentukan apakah aplikasi Anda menggunakan rantai operator, lihat grafik pekerjaan di konsol Managed Service for Apache Flink. Setiap simpul dalam aplikasi mewakili satu atau beberapa operator. Grafik menunjukkan operator yang sudah dirantai sebagai satu simpul.

API DataStream

Aplikasi Apache Flink Anda menggunakanApache FlinkDataStreamAPIuntuk mengubah data dalam aliran data.

Bagian ini berisi topik berikut:

- Menggunakan Konektor untuk Memindahkan Data dalam Layanan Terkelola untuk Apache Flink Dengan API DataStream : Komponen ini memindahkan data antara aplikasi Anda dan sumber serta tujuan data eksternal.
- Mengubah Data Menggunakan Operator di Managed Service untuk Apache Flink Dengan API DataStream : Komponen ini mengubah atau mengelompokkan elemen data dalam aplikasi Anda.
- Melacak Acara di Layanan Terkelola untuk Apache Flink MenggunakanDataStreamAPI: Topik ini menjelaskan bagaimana Managed Service for Apache Flink melacak peristiwa saat menggunakanDataStreamAPI.

Menggunakan Konektor untuk Memindahkan Data dalam Layanan Terkelola untuk Apache Flink Dengan API DataStream

Di Amazon Managed Service for Apache Flink DataStream API, konektor adalah komponen perangkat lunak yang memindahkan data masuk dan keluar dari Layanan Terkelola untuk aplikasi Apache Flink. Konektor adalah integrasi fleksibel yang memungkinkan Anda membaca dari file dan direktori. Konektor terdiri dari modul lengkap untuk berinteraksi dengan layanan Amazon dan sistem pihak ketiga.

Tipe konektor termasuk berikut ini:

- Sumber: Berikan data ke aplikasi Anda dari Kinesis data stream, file, atau sumber data lainnya.
- Sink: Kirim data dari aplikasi Anda ke aliran data Kinesis, aliran Kinesis Data Firehose, atau tujuan data lainnya.
- I/O Asinkron: Menyediakan akses asinkron ke sumber data (seperti basis data) untuk memperkaya peristiwa aliran.

Konektor yang Tersedia

Kerangka kerja Apache Flink berisi konektor untuk mengakses data dari berbagai sumber. Untuk informasi tentang konektor yang tersedia di kerangka kerja Apache Flink, lihat Konektor di Dokumentasi Apache Flink.

Warning

Jika Anda memiliki aplikasi yang berjalan di Flink 1.6, 1.8, 1.11 atau 1.13 dan ingin berjalan di Timur Tengah (UEA), Asia Pasifik (Hyderabad), Israel (Tel Aviv), Eropa (Zurich), Timur Tengah (UEA), Asia Pasifik (Melbourne) atau Asia Pasifik (Jakarta), Anda mungkin perlu membangun kembali arsip aplikasi Anda dengan konektor yang diperbarui atau meningkatkan ke Flink 1.15. Berikut ini adalah pedoman yang direkomendasikan:

Peningkatan konektor

V Fl	Konektor yang digunakan	Penyelesaian
1, - 1,	Firehose	Aplikasi Anda bergantung pada konektor Firehose versi usang yang tidak mengetahui Wilayah yang lebih baru. AWS Bangun kembali arsip aplikasi Anda dengan konektor Firehose versi 2.1.0. v2.1.0

V Fl	Konektor yang digunakan	Penyelesaian
1.	Kinesis	Aplikasi Anda bergantung pada versi konektor Kinesis Flink yang sudah ketinggalan zaman yang tidak mengetahui Wilayah yang lebih baru. AWS Bangun kembali arsip aplikasi Anda dengan konektor Flink Kinesis

V Fl	Konektor yang digunakan	Penyelesaian
		versi 1.6.1. https://github.com/awslabs/amazon-kinesis-connector-flink / pohon/1.6.1

V Fl	Konektor yang digunakan	Penyelesaian
1.	Kinesis	Aplikasi Anda bergantung pada versi konektor Kinesis Flink yang sudah ketinggalan zaman yang tidak mengetahui Wilayah yang lebih baru. AWS Membangu kembali arsip aplikasi Anda dengan konektor Flink Kinesis

V Fl	Konektor yang digunakan	Penyelesaian
		versi 2.4.1. https://github.com/awslabs/amazon-kinesis-connector-flink / pohon/2.4.1

V Fl	Konektor yang digunakan	Penyelesa ian
1. da 1.	Kinesis	Aplikasi Anda bergantun g pada versi konektor Kinesis Flink yang sudah ketinggal an zaman yang tidak mengetahu i Wilayah yang lebih baru. AWS Sayangnya , Flink tidak lagi merilis patch atau perbaikan bug

V Fl	Konektor yang digunakan	Penyelesaian
		untuk konektor 1.6/1.13. Kami menyarankan memperbaui ke Flink 1.15 dengan membangu kembali arsip aplikasi Anda dengan Flink 1.15.

Menambahkan Sumber Data Streaming ke Layanan Terkelola untuk Apache Flink

Apache Flink menyediakan konektor untuk membaca dari file, soket, koleksi, dan sumber kustom. Dalam kode aplikasi Anda, Anda menggunakan sumber Apache Flink untuk menerima data dari aliran. Bagian ini menjelaskan sumber yang tersedia untuk layanan Amazon.

Kinesis Data Streams

Sumber `FlinkKinesisConsumer` menyediakan data streaming ke aplikasi Anda dari Amazon Kinesis data stream.

Membuat**`FlinkKinesisConsumer`**

Contoh kode berikut mendemonstrasikan pembuatan `FlinkKinesisConsumer`:

```
Properties inputProperties = new Properties();
inputProperties.setProperty(ConsumerConfigConstants.AWS_REGION, region);
inputProperties.setProperty(ConsumerConfigConstants.STREAM_INITIAL_POSITION, "LATEST");

DataStream<string> input = env.addSource(new FlinkKinesisConsumer<>(inputStreamName,
 new SimpleStringSchema(), inputProperties));
```

Untuk informasi selengkapnya tentang cara menggunakan `FlinkKinesisConsumer`, lihat Unduh dan Periksa Kode Java Streaming Apache Flink.

Membuat **`FlinkKinesisConsumer`** yang menggunakan konsumen EFO

FlinkKinesisConsumer Sekarang mendukung Enhanced Fan-Out (EFO).

Jika konsumen Kinesis menggunakan EFO, layanan Kinesis Data Streams memberikan bandwidth khusus miliknya sendiri, bukan meminta konsumen berbagi bandwidth aliran tetap dengan konsumen lain yang membaca dari aliran.

Untuk informasi selengkapnya tentang penggunaan EFO dengan konsumen Kinesis, lihat FLIP-128: Fan Out yang Disempurnakan untuk Konsumen Kinesis AWS.

Anda mengaktifkan konsumen EFO dengan mengatur parameter berikut pada konsumen Kinesis:

- RECORD_PUBLISHER_TYPE: Atur parameter ini ke EFO untuk aplikasi Anda agar dapat menggunakan konsumen EFO untuk mengakses data Kinesis Data Stream.
- EFO_CONSUMER_NAME: Atur parameter ini ke nilai string yang unik di antara konsumen aliran ini. Menggunakan kembali nama konsumen di Kinesis Data Stream yang sama akan menyebabkan konsumen sebelumnya yang menggunakan nama tersebut dihentikan.

Untuk mengonfigurasi `FlinkKinesisConsumer` untuk menggunakan EFO, tambahkan parameter berikut ke konsumen:

```
consumerConfig.putIfAbsent(RECORD_PUBLISHER_TYPE, "EFO");
consumerConfig.putIfAbsent(EFO_CONSUMER_NAME, "basic-efo-flink-app");
```

Untuk contoh aplikasi Managed Service for Apache Flink yang menggunakan konsumen EFO, lihat. Konsumen EFO

Amazon MSK

Sumber `KafkaSource` menyediakan data streaming ke aplikasi Anda dari topik Amazon MSK.

Membuat**`KafkaSource`**

Contoh kode berikut mendemonstrasikan pembuatan `KafkaSource`:

```
KafkaSource<String> source = KafkaSource.<String>builder()
    .setBootstrapServers(brokers)
    .setTopics("input-topic")
    .setGroupId("my-group")
    .setStartingOffsets(OffsetsInitializer.earliest())
    .setValueOnlyDeserializer(new SimpleStringSchema())
    .build();

env.fromSource(source, WatermarkStrategy.noWatermarks(), "Kafka Source");
```

Untuk informasi selengkapnya tentang cara menggunakan `KafkaSource`, lihat Replikasi MSK.

Menulis Data Menggunakan Sinks di Managed Service untuk Apache Flink

Dalam kode aplikasi Anda, Anda menggunakan sink Apache Flink untuk menulis data dari aliran Apache Flink ke layanan AWS, seperti Kinesis Data Streams.

Apache Flink menyediakan sink untuk file, soket, dan sink kustom. Sink berikut tersedia untuk AWS:

Kinesis Data Streams

Apache Flink memberikan informasi tentang Konektor Kinesis Data Streams di dokumentasi Apache Flink.

Untuk contoh aplikasi yang menggunakan Kinesis data stream untuk input dan output, lihat Memulai (DataStream API).

Amazon S3

Anda dapat menggunakan Apache Flink `StreamingFileSink` untuk menulis objek ke bucket Amazon S3.

Untuk contoh tentang cara menulis objek ke S3, lihat the section called “Sink S3”.

Kinesis Data Firehose

`FlinkKinesisFirehoseProducer` adalah sink Apache Flink yang andal dan dapat diskalakan untuk menyimpan output aplikasi menggunakan layanan Kinesis Data Firehose. Bagian ini menjelaskan cara menyiapkan proyek Maven untuk membuat dan menggunakan `FlinkKinesisFirehoseProducer`.

Topik

- MembuatFlinkKinesisFirehoseProducer
- Contoh Kode FlinkKinesisFirehoseProducer

Membuat**`FlinkKinesisFirehoseProducer`**

Contoh kode berikut mendemonstrasikan pembuatan `FlinkKinesisFirehoseProducer`:

```
Properties outputProperties = new Properties();
outputProperties.setProperty(ConsumerConfigConstants.AWS_REGION, region);

FlinkKinesisFirehoseProducer<String> sink = new
 FlinkKinesisFirehoseProducer<>(outputStreamName, new SimpleStringSchema(),
 outputProperties);
```

Contoh Kode **`FlinkKinesisFirehoseProducer`**

Contoh kode berikut mendemonstrasikan cara membuat dan mengonfigurasi `FlinkKinesisFirehoseProducer` serta mengirim data dari aliran data Apache Flink ke layanan Kinesis Data Firehose.

```
package com.amazonaws.services.kinesisanalytics;

import
 com.amazonaws.services.kinesisanalytics.flink.connectors.config.ProducerConfigConstants;
import
 com.amazonaws.services.kinesisanalytics.flink.connectors.producer.FlinkKinesisFirehoseProducer
import com.amazonaws.services.kinesisanalytics.runtime.KinesisAnalyticsRuntime;
import org.apache.flink.api.common.serialization.SimpleStringSchema;
import org.apache.flink.streaming.api.datastream.DataStream;
import org.apache.flink.streaming.api.environment.StreamExecutionEnvironment;
import org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer;
import org.apache.flink.streaming.connectors.kinesis.FlinkKinesisProducer;
```

```
import org.apache.flink.streaming.connectors.kinesis.config.ConsumerConfigConstants;

import java.io.IOException;
import java.util.Map;
import java.util.Properties;

public class StreamingJob {

 private static final String region = "us-east-1";
 private static final String inputStreamName = "ExampleInputStream";
 private static final String outputStreamName = "ExampleOutputStream";

 private static DataStream<String>
 createSourceFromStaticConfig(StreamExecutionEnvironment env) {
  Properties inputProperties = new Properties();
  inputProperties.setProperty(ConsumerConfigConstants.AWS_REGION, region);
  inputProperties.setProperty(ConsumerConfigConstants.STREAM_INITIAL_POSITION,
 "LATEST");

  return env.addSource(new FlinkKinesisConsumer<>(inputStreamName, new
 SimpleStringSchema(), inputProperties));
 }

 private static DataStream<String>
 createSourceFromApplicationProperties(StreamExecutionEnvironment env)
   throws IOException {
  Map<String, Properties> applicationProperties =
 KinesisAnalyticsRuntime.getApplicationProperties();
  return env.addSource(new FlinkKinesisConsumer<>(inputStreamName, new
 SimpleStringSchema(),
    applicationProperties.get("ConsumerConfigProperties")));
 }

 private static FlinkKinesisFirehoseProducer<String>
 createFirehoseSinkFromStaticConfig() {
  /*
   * com.amazonaws.services.kinesisanalytics.flink.connectors.config.
   * ProducerConfigConstants
   * lists of all of the properties that firehose sink can be configured with.
   */

  Properties outputProperties = new Properties();
  outputProperties.setProperty(ConsumerConfigConstants.AWS_REGION, region);
```

```
 FlinkKinesisFirehoseProducer<String> sink = new
FlinkKinesisFirehoseProducer<>(outputStreamName,
   new SimpleStringSchema(), outputProperties);
 ProducerConfigConstants config = new ProducerConfigConstants();
 return sink;
}

private static FlinkKinesisFirehoseProducer<String>
createFirehoseSinkFromApplicationProperties() throws IOException {
 /*
  * com.amazonaws.services.kinesisanalytics.flink.connectors.config.
  * ProducerConfigConstants
  * lists of all of the properties that firehose sink can be configured with.
  */

 Map<String, Properties> applicationProperties =
KinesisAnalyticsRuntime.getApplicationProperties();
 FlinkKinesisFirehoseProducer<String> sink = new
FlinkKinesisFirehoseProducer<>(outputStreamName,
   new SimpleStringSchema(),
   applicationProperties.get("ProducerConfigProperties"));
 return sink;
}

public static void main(String[] args) throws Exception {
 // set up the streaming execution environment
 final StreamExecutionEnvironment env =
StreamExecutionEnvironment.getExecutionEnvironment();

 /*
  * if you would like to use runtime configuration properties, uncomment the
  * lines below
  * DataStream<String> input = createSourceFromApplicationProperties(env);
  */

 DataStream<String> input = createSourceFromStaticConfig(env);

 // Kinesis Firehose sink
 input.addSink(createFirehoseSinkFromStaticConfig());

 // If you would like to use runtime configuration properties, uncomment the
 // lines below
 // input.addSink(createFirehoseSinkFromApplicationProperties());
```

```
  env.execute("Flink Streaming Java API Skeleton");
 }
}
```

Untuk tutorial lengkap tentang cara menggunakan sink Kinesis Data Firehose, lihat the section called "Sink Kinesis Data Firehose".

Menggunakan Asynchronous I/O di Managed Service untuk Apache Flink

Operator I/O Asinkron memperkaya aliran data menggunakan sumber data eksternal seperti basis data. Layanan Terkelola untuk Apache Flink memperkaya peristiwa streaming secara asinkron sehingga permintaan dapat dikumpulkan untuk efisiensi yang lebih besar.

Untuk informasi selengkapnya, lihat I/O Asinkron di Dokumentasi Apache Flink.

Mengubah Data Menggunakan Operator di Managed Service untuk Apache Flink Dengan API DataStream

Untuk mengubah data masuk dalam Layanan Terkelola untuk Apache Flink, Anda menggunakan operator Apache Flink. Operator Apache Flink mengubah satu atau beberapa aliran data menjadi aliran data baru. Aliran data baru berisi data yang dimodifikasi dari aliran data asli. Apache Flink menyediakan lebih dari 25 operator pemrosesan aliran yang dibangun sebelumnya. Untuk informasi selengkapnya, lihat Operator di Dokumentasi Apache Flink.

Topik ini berisi bagian-bagian berikut:

- Ubah Operator
- Operator Agregasi

Ubah Operator

Berikut adalah contoh transformasi teks sederhana pada salah satu bidang aliran data JSON.

Kode ini membuat aliran data yang diubah. Aliran data baru memiliki data yang sama dengan aliran asli, dengan string " Company" yang ditambahkan ke isi bidang TICKER.

```
DataStream<ObjectNode> output = input.map(
    new MapFunction<ObjectNode, ObjectNode>() {
        @Override
```

```
        public ObjectNode map(ObjectNode value) throws Exception {
            return value.put("TICKER", value.get("TICKER").asText() + " Company");
        }
    }
);
```

Operator Agregasi

Berikut adalah contoh operator agregasi. Kode membuat aliran data agregat. Operator membuat jendela tumbling 5 detik dan menampilkan jumlah dari nilai `PRICE` untuk catatan di jendela dengan nilai `TICKER` yang sama.

```
DataStream<ObjectNode> output = input.keyBy(node -> node.get("TICKER").asText())
    .window(TumblingProcessingTimeWindows.of(Time.seconds(5)))
    .reduce((node1, node2) -> {
        double priceTotal = node1.get("PRICE").asDouble() +
 node2.get("PRICE").asDouble();
        node1.replace("PRICE", JsonNodeFactory.instance.numberNode(priceTotal));
    return node1;
});
```

Untuk contoh kode lengkap yang menggunakan operator, lihat Memulai (DataStream API). Kode sumber untuk aplikasi Memulai tersedia di Memulai di Repositori Managed Service for Apache Flink Java Examples GitHub .

Melacak Acara di Layanan Terkelola untuk Apache Flink MenggunakanDataStreamAPI

Layanan Terkelola untuk Apache Flink melacak peristiwa menggunakan stempel waktu berikut:

- Processing Time (Waktu Pemrosesan): Mengacu pada waktu sistem mesin yang menjalankan operasi masing-masing.
- Event Time (Waktu Peristiwa): Mengacu pada waktu setiap peristiwa individu terjadi pada perangkat produksinya.
- Waktu konsumsi:Mengacu pada waktu peristiwa memasuki Layanan Terkelola untuk layanan Apache Flink.

Anda mengatur waktu yang digunakan oleh lingkungan streaming menggunakan `setStreamTimeCharacteristic`:

```
env.setStreamTimeCharacteristic(TimeCharacteristic.ProcessingTime);
env.setStreamTimeCharacteristic(TimeCharacteristic.IngestionTime);
env.setStreamTimeCharacteristic(TimeCharacteristic.EventTime);
```

Untuk informasi selengkapnya tentang stempel waktu, lihat Waktu Peristiwa di Dokumentasi Apache Flink.

Tabel API

Aplikasi Apache Flink Anda menggunakan API Tabel Apache Flink untuk berinteraksi dengan data dalam aliran menggunakan model relasional. Anda menggunakan API Tabel untuk mengakses data menggunakan sumber Tabel, lalu menggunakan fungsi Tabel untuk mengubah dan memfilter data tabel. Anda dapat mengubah dan memfilter data tabel menggunakan fungsi API atau perintah SQL.

Bagian ini berisi topik berikut:

- Konektor API Tabel: Komponen ini memindahkan data antara aplikasi Anda dan sumber serta tujuan data eksternal.
- Atribut Waktu API Tabel: Topik ini menjelaskan bagaimana Managed Service for Apache Flink melacak peristiwa saat menggunakan Table API.

Konektor API Tabel

Dalam model pemrograman Apache Flink, konektor adalah komponen yang digunakan aplikasi Anda untuk membaca atau menulis data dari sumber eksternal, seperti layanan AWS lainnya.

Dengan API Tabel Apache Flink, Anda dapat menggunakan tipe konektor berikut:

- Sumber API Tabel: Anda menggunakan konektor sumber API Tabel untuk membuat tabel dalam `TableEnvironment` Anda menggunakan panggilan API atau kueri SQL.
- Sink API Tabel: Anda menggunakan perintah SQL untuk menulis data tabel ke sumber eksternal seperti topik Amazon MSK atau bucket Amazon S3.

Sumber API Tabel

Anda membuat sumber tabel dari aliran data. Kode berikut membuat tabel dari topik Amazon MSK:

```
//create the table
    final FlinkKafkaConsumer<StockRecord> consumer = new
 FlinkKafkaConsumer<StockRecord>(kafkaTopic, new KafkaEventDeserializationSchema(),
 kafkaProperties);
    consumer.setStartFromEarliest();
    //Obtain stream
    DataStream<StockRecord> events = env.addSource(consumer);

    Table table = streamTableEnvironment.fromDataStream(events);
```

Untuk informasi selengkapnya tentang sumber tabel, lihat Tabel & Konektor di Dokumentasi Apache Flink.

Sink API Tabel

Untuk menulis data tabel ke sink, Anda membuat sink di SQL, lalu jalankan sink berbasis SQL di objek `StreamTableEnvironment`.

Contoh kode berikut mendemonstrasikan cara menulis data tabel ke sink Amazon S3:

```
final String s3Sink = "CREATE TABLE sink_table (" +
    "event_time TIMESTAMP," +
    "ticker STRING," +
    "price DOUBLE," +
    "dt STRING," +
    "hr STRING" +
    ")" +
    " PARTITIONED BY (ticker,dt,hr)" +
    " WITH" +
    "(" +
    " 'connector' = 'filesystem'," +
    " 'path' = '" + s3Path + "'," +
    " 'format' = 'json'" +
    ") ";

    //send to s3
    streamTableEnvironment.executeSql(s3Sink);
    filteredTable.executeInsert("sink_table");
```

Anda dapat menggunakan`format`parameter untuk mengontrol format Managed Service untuk Apache Flink yang digunakan untuk menulis output ke wastafel. Untuk informasi tentang format, lihat Format di Dokumentasi Apache Flink.

Untuk informasi selengkapnya tentang sink tabel, lihat Tabel & Konektor di Dokumentasi Apache Flink.

Sumber dan Sink yang Ditetapkan Pengguna

Anda dapat menggunakan konektor Apache Kafka yang ada untuk mengirim data ke dan dari layanan AWS lainnya, seperti Amazon MSK dan Amazon S3. Untuk berinteraksi dengan sumber data dan tujuan lainnya, Anda dapat menentukan sumber dan sink Anda sendiri. Untuk informasi selengkapnya, lihat Sumber dan Sink yang Ditetapkan Pengguna di Dokumentasi Apache Flink.

Atribut Waktu API Tabel

Setiap catatan dalam aliran data memiliki beberapa stempel waktu yang menentukan kapan peristiwa yang terkait dengan catatan terjadi:

- Event Time (Waktu Peristiwa): Stempel waktu yang ditetapkan pengguna yang menentukan kapan peristiwa yang dibuat catatan terjadi.
- Ingestion Time (Waktu Penyerapan): Waktu ketika aplikasi Anda mengambil catatan dari aliran data.
- Processing Time (Waktu pemrosesan): Waktu ketika aplikasi Anda memproses catatan.

Saat Apache Flink Table API membuat jendela berdasarkan waktu rekaman, Anda menentukan stempel waktu mana yang digunakannya dengan menggunakansetStreamTimeKarakteristikmetode.

Untuk informasi selengkapnya tentang penggunaan stempel waktu dengan API Tabel, lihat Atribut Waktu di Dokumentasi Apache Flink.

Menggunakan Python dengan Managed Service untuk Apache Flink

> **Note**
>
> Jika Anda mengembangkan aplikasi Python Flink pada Mac baru dengan chip Apple Silicon, Anda mungkin mengalami beberapa masalah yang diketahui dengan dependensi Python 1,15. PyFlink Dalam hal ini kami sarankan menjalankan interpreter Python di Docker. Untuk step-by-step petunjuk, lihat PyFlink 1,15 pengembangan di Apple Silicon Mac.

Apache Flink versi 1.15.2 mencakup dukungan untuk membuat aplikasi menggunakan Python versi 3.8, menggunakan perpustakaan. PyFlink Anda membuat Managed Service untuk aplikasi Apache Flink menggunakan Python dengan melakukan hal berikut:

- Buat kode aplikasi Python Anda sebagai file teks dengan metode `main`.
- Gabungkan file kode aplikasi Anda dan dependensi Python atau Java apa pun ke dalam file zip, dan unggah ke bucket Amazon S3.
- Buat Layanan Terkelola untuk aplikasi Apache Flink Anda, tentukan lokasi kode Amazon S3, properti aplikasi, dan pengaturan aplikasi Anda.

Pada tingkat tinggi, API Tabel Python adalah wrapper di sekitar API Tabel Java. Untuk informasi tentang API Tabel Python, lihat Pengantar API Tabel Python di Dokumentasi Apache Flink.

Pemrograman Layanan Terkelola Anda untuk Apache Flink untuk Aplikasi Python

Anda kode Layanan Terkelola untuk Apache Flink untuk aplikasi Python menggunakan Apache Flink Python Table API. Mesin Apache Flink menerjemahkan pernyataan API Tabel Python (berjalan di VM Python) menjadi pernyataan API Tabel Java (berjalan di VM Java).

Anda menggunakan API Tabel Python dengan melakukan hal berikut:

- Buat referensi ke`StreamTableEnvironment`.
- Buat objek `table` dari data streaming sumber Anda dengan menjalankan query pada referensi `StreamTableEnvironment`.
- Jalankan kueri di objek `table` untuk membuat tabel output.
- Tulis tabel output Anda ke tujuan Anda menggunakan `StatementSet`.

Untuk mulai menggunakan Python Table API di Managed Service for Apache Flink, lihat. Memulai dengan Amazon Managed Service untuk Apache Flink untuk Python

Membaca dan Menulis Data Streaming

Untuk membaca dan menulis data streaming, Anda menjalankan kueri SQL pada lingkungan tabel.

Membuat Tabel

Contoh kode berikut menunjukkan fungsi yang ditetapkan pengguna yang membuat kueri SQL. Kueri SQL membuat tabel yang berinteraksi dengan aliran Kinesis:

```
def create_table(table_name, stream_name, region, stream_initpos):
   return """ CREATE TABLE {0} (
                `record_id` VARCHAR(64) NOT NULL,
                `event_time` BIGINT NOT NULL,
                `record_number` BIGINT NOT NULL,
                `num_retries` BIGINT NOT NULL,
                `verified` BOOLEAN NOT NULL
              )
              PARTITIONED BY (record_id)
              WITH (
                'connector' = 'kinesis',
                'stream' = '{1}',
                'aws.region' = '{2}',
                'scan.stream.initpos' = '{3}',
                'sink.partitioner-field-delimiter' = ';',
                'sink.producer.collection-max-count' = '100',
                'format' = 'json',
                'json.timestamp-format.standard' = 'ISO-8601'
              ) """.format(table_name, stream_name, region, stream_initpos)
```

Membaca Data Streaming

Contoh kode berikut menunjukkan cara menggunakan kueri SQL `CreateTable` sebelumnya di referensi lingkungan tabel untuk membaca data:

```
   table_env.execute_sql(create_table(input_table, input_stream, input_region,
 stream_initpos))
```

Menulis Data Streaming

Contoh kode berikut menunjukkan cara menggunakan kueri SQL dari contoh `CreateTable` untuk membuat referensi tabel output, dan cara menggunakan `StatementSet` untuk berinteraksi dengan tabel untuk menulis data ke aliran Kinesis tujuan:

```
   table_result = table_env.execute_sql("INSERT INTO {0} SELECT * FROM {1}"
                        .format(output_table_name, input_table_name))
```

Membaca Properti Runtime

Anda dapat menggunakan properti runtime untuk mengonfigurasi aplikasi Anda tanpa mengubah kode aplikasi Anda.

Anda menentukan properti aplikasi untuk aplikasi Anda dengan cara yang sama seperti dengan Managed Service untuk Apache Flink untuk aplikasi Java. Anda dapat menentukan properti runtime dengan cara berikut:

- Menggunakan CreateApplicationtindakan.
- Menggunakan UpdateApplicationtindakan.
- Mengonfigurasi aplikasi Anda menggunakan konsol.

Anda mengambil properti aplikasi dalam kode dengan membaca file json yang disebut `application_properties.json` bahwa runtime Layanan Terkelola untuk Apache Flink dibuat.

Contoh kode berikut menunjukkan properti aplikasi membaca dari file `application_properties.json`:

```
file_path = '/etc/flink/application_properties.json'
   if os.path.isfile(file_path):
       with open(file_path, 'r') as file:
           contents = file.read()
           properties = json.loads(contents)
```

Contoh kode fungsi yang ditetapkan pengguna berikut menunjukkan membaca grup properti dari objek properti aplikasi: mengambil:

```
def property_map(properties, property_group_id):
   for prop in props:
       if prop["PropertyGroupId"] == property_group_id:
           return prop["PropertyMap"]
```

Contoh kode berikut menunjukkan membaca properti yang disebut INPUT_STREAM_KEY dari grup properti yang dikembalikan contoh sebelumnya:

```
input_stream = input_property_map[INPUT_STREAM_KEY]
```

Membuat paket kode aplikasi Anda

Setelah Anda membuat aplikasi Python, Anda menggabungkan file kode Anda dan dependensi ke dalam file zip.

File zip Anda harus berisi script python dengan metode `main`, dan secara opsional dapat berisi berikut ini:

- File kode Python tambahan
- Kode Java yang ditetapkan pengguna dalam file JAR
- Pustaka Java dalam file JAR

> **Note**
>
> File zip aplikasi Anda harus berisi semua dependensi untuk aplikasi Anda. Anda tidak dapat merujuk pustaka dari sumber lainnya untuk aplikasi Anda.

Membuat Layanan Terkelola Anda untuk Aplikasi Apache Flink Python

Menentukan File Kode Anda

Setelah Anda telah membuat paket kode aplikasi, Anda mengunggahnya ke bucket Amazon S3. Anda kemudian membuat aplikasi Anda menggunakan konsol atau CreateApplicationtindakan.

Ketika Anda membuat aplikasi Anda menggunakan CreateApplicationtindakan, Anda menentukan file kode dan arsip dalam file zip Anda menggunakan grup properti aplikasi khusus yang disebut`kinesis.analytics.flink.run.options`. Anda dapat menentukan file tipe berikut:

- python: File teks yang berisi metode utama Python.
- jarfile: File JAR Java yang berisi fungsi yang ditetapkan pengguna Java.
- pyFiles: File sumber daya Python yang berisi sumber daya yang akan digunakan oleh aplikasi.
- pyArchives: File zip yang berisi file sumber daya untuk aplikasi.

Untuk informasi selengkapnya tentang tipe file kode Python Apache Flink, lihat Penggunaan Baris Perintah di Dokumentasi Apache Flink.

> **Note**
>
> Layanan Terkelola untuk Apache Flink tidak mendukung`pyModule`,`pyExecutable`, atau jenis `pyRequirements` file. Semua kode, persyaratan, dan dependensi harus dalam file zip Anda. Anda tidak dapat menentukan dependensi yang akan diinstal menggunakan pip.

Cuplikan json contoh berikut menunjukkan cara menentukan lokasi file dalam file zip aplikasi Anda:

```
"ApplicationConfiguration": {
    "EnvironmentProperties": {
      "PropertyGroups": [
        {
          "PropertyGroupId": "kinesis.analytics.flink.run.options",
          "PropertyMap": {
            "python": "MyApplication/main.py",
            "jarfile": "MyApplication/lib/myJarFile.jar",
            "pyFiles": "MyApplication/lib/myDependentFile.py",
            "pyArchives": "MyApplication/lib/myArchive.zip"
          }
        },
```

Memantau Layanan Terkelola Python Anda untuk Aplikasi Apache Flink

Anda menggunakan CloudWatch log aplikasi Anda untuk memantau Layanan Terkelola Anda untuk aplikasi Apache Flink Python.

Layanan Terkelola untuk Apache Flink mencatat pesan berikut untuk aplikasi Python:

- Pesan yang ditulis ke konsol menggunakan `print()` di metode `main` aplikasi.
- Pesan yang dikirim dalam fungsi yang ditetapkan pengguna menggunakan paket `logging`. Contoh kode berikut menunjukkan menulis ke log aplikasi dari fungsi yang ditetapkan pengguna:

  ```
  import logging

  @udf(input_types=[DataTypes.BIGINT()], result_type=DataTypes.BIGINT())
  def doNothingUdf(i):
      logging.info("Got {} in the doNothingUdf".format(str(i)))
      return i
  ```

- Pesan kesalahan yang dilemparkan oleh aplikasi.

Jika aplikasi melemparkan pengecualian di fungsi `main`, pengecualian akan muncul di log aplikasi Anda.

Contoh berikut menunjukkan entri log untuk pengecualian yang dilemparkan dari kode Python:

```
2021-03-15 16:21:20.000   -------------------------- Python Process Started
 --------------------------
2021-03-15 16:21:21.000   Traceback (most recent call last):
2021-03-15 16:21:21.000   "  File ""/tmp/flink-
web-6118109b-1cd2-439c-9dcd-218874197fa9/flink-web-upload/4390b233-75cb-4205-
a532-441a2de83db3_code/PythonKinesisSink/PythonUdfUndeclared.py"", line 101, in
 <module>"
2021-03-15 16:21:21.000       main()
2021-03-15 16:21:21.000   "  File ""/tmp/flink-
web-6118109b-1cd2-439c-9dcd-218874197fa9/flink-web-upload/4390b233-75cb-4205-
a532-441a2de83db3_code/PythonKinesisSink/PythonUdfUndeclared.py"", line 54, in main"
2021-03-15 16:21:21.000   "    table_env.register_function(""doNothingUdf"",
 doNothingUdf)"
2021-03-15 16:21:21.000   NameError: name 'doNothingUdf' is not defined
2021-03-15 16:21:21.000   -------------------------- Python Process Exited
 --------------------------
2021-03-15 16:21:21.000   Run python process failed
2021-03-15 16:21:21.000   Error occurred when trying to start the job
```

> **Note**
>
> Karena masalah performa, sebaiknya hanya gunakan pesan log kustom selama pengembangan aplikasi.

Menanyakan Log dengan CloudWatch Wawasan

Kueri CloudWatch Insights berikut mencari log yang dibuat oleh entrypoint Python saat menjalankan fungsi utama aplikasi Anda:

```
fields @timestamp, message
| sort @timestamp asc
| filter logger like /PythonDriver/
| limit 1000
```

Properti Runtime di Layanan Terkelola untuk Apache Flink

Anda dapat menggunakan properti runtime untuk mengonfigurasi aplikasi Anda tanpa mengompilasi ulang kode aplikasi Anda.

Topik ini berisi bagian-bagian berikut:

Bekerja dengan Properti Runtime di Konsol

Anda dapat menambahkan, memperbarui, atau menghapus properti runtime dari Layanan Terkelola untuk aplikasi Apache Flink menggunakan konsol.

> **Note**
>
> Anda tidak dapat menambahkan properti runtime saat membuat aplikasi di konsol Managed Service for Apache Flink.

Perbarui Properti Runtime untuk Layanan Terkelola untuk aplikasi Apache Flink

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/flink
2. Pilih Layanan Terkelola Anda untuk aplikasi Apache Flink. Pilih Application details (Detail aplikasi).
3. Di halaman untuk aplikasi Anda, pilih Configure (Konfigurasikan).
4. Perluas bagian Properties (Properti).
5. Gunakan kontrol di bagian Properti untuk menentukan grup properti dengan pasangan nilai kunci. Gunakan kontrol ini untuk menambah, memperbarui, atau menghapus grup properti dan properti runtime.
6. Pilih Update (Perbarui).

Bekerja dengan Properti Runtime di CLI

Anda dapat menambahkan, memperbarui, atau menghapus properti runtime menggunakan AWS CLI.

Bagian ini mencakup permintaan contoh tindakan API untuk mengonfigurasi properti runtime aplikasi. Untuk informasi tentang cara menggunakan file JSON untuk input tindakan API, lihat Layanan Terkelola untuk Kode Contoh API Apache Flink.

> **Note**
>
> Ganti ID akun sampel (*012345678901*) dalam contoh berikut dengan ID akun Anda.

Menambahkan Properti Runtime saat Membuat Aplikasi

Permintaan contoh berikut untuk tindakan `CreateApplication` menambahkan dua grup properti runtime (`ProducerConfigProperties` dan `ConsumerConfigProperties`) saat Anda membuat aplikasi:

```
{
    "ApplicationName": "MyApplication",
    "ApplicationDescription": "my java test app",
    "RuntimeEnvironment": "FLINK-1_15",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-stream-rw-role",
    "ApplicationConfiguration": {
        "ApplicationCodeConfiguration": {
            "CodeContent": {
                "S3ContentLocation": {
                    "BucketARN": "arn:aws:s3:::ka-app-code-username",
                    "FileKey": "java-getting-started-1.0.jar"
                }
            },
            "CodeContentType": "ZIPFILE"
        },
        "EnvironmentProperties":  {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "flink.stream.initpos" : "LATEST",
                    "aws.region" : "us-west-2",
                    "AggregationEnabled" : "false"
               }
            },
            {
               "PropertyGroupId": "ConsumerConfigProperties",
```

```
                "PropertyMap" : {
                     "aws.region" : "us-west-2"
                }
            }
         ]
      }
   }
}
```

Menambahkan dan Memperbarui Properti Runtime dalam Aplikasi yang Ada

Permintaan contoh berikut untuk tindakan `UpdateApplication` menambahkan atau memperbarui properti runtime untuk aplikasi yang ada:

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 2,
  "ApplicationConfigurationUpdate": {
    "EnvironmentPropertyUpdates": {
      "PropertyGroups": [
        {
          "PropertyGroupId": "ProducerConfigProperties",
          "PropertyMap" : {
            "flink.stream.initpos" : "LATEST",
            "aws.region" : "us-west-2",
            "AggregationEnabled" : "false"
          }
        },
        {
          "PropertyGroupId": "ConsumerConfigProperties",
          "PropertyMap" : {
            "aws.region" : "us-west-2"
          }
        }
      ]
    }
  }
}
```

> **Note**
>
> Jika Anda menggunakan kunci yang tidak memiliki properti runtime yang sesuai dalam grup properti, Managed Service for Apache Flink menambahkan pasangan kunci-nilai sebagai properti baru. Jika Anda menggunakan kunci untuk properti runtime yang ada di grup properti, Managed Service for Apache Flink akan memperbarui nilai properti.

Menghapus Properti Runtime

Permintaan contoh berikut untuk tindakan `UpdateApplication` menghapus semua properti runtime dan grup properti dari aplikasi yang ada:

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 3,
  "ApplicationConfigurationUpdate": {
    "EnvironmentPropertyUpdates": {
      "PropertyGroups": []
    }
  }
}
```

> **Important**
>
> Jika Anda menghilangkan grup properti yang ada atau kunci properti yang ada di grup properti, grup properti atau properti akan dihapus.

Mengakses Properti Runtime dalam Layanan Terkelola untuk Apache Flink Application

Anda mengambil properti runtime dalam kode aplikasi Java Anda menggunakan metode `KinesisAnalyticsRuntime.getApplicationProperties()` statis, yang mengembalikan objek `Map<String, Properties>`.

Contoh kode Java berikut mengambil properti runtime untuk aplikasi Anda:

```
 Map<String, Properties> applicationProperties =
 KinesisAnalyticsRuntime.getApplicationProperties();
```

Anda mengambil grup properti (sebagai objek `Java.Util.Properties`) sebagai berikut:

```
Properties consumerProperties = applicationProperties.get("ConsumerConfigProperties");
```

Anda biasanya mengonfigurasi sumber atau sink Apache Flinkdengan meneruskan di objek `Properties` tanpa perlu mengambil properti individu. Contoh kode berikut menunjukkan cara membuat sumber Flink dengan meneruskan di objek `Properties` yang diambil dari properti runtime:

```
private static FlinkKinesisProducer<String> createSinkFromApplicationProperties()
 throws IOException {
  Map<String, Properties> applicationProperties =
 KinesisAnalyticsRuntime.getApplicationProperties();
  FlinkKinesisProducer<String> sink = new FlinkKinesisProducer<String>(new
 SimpleStringSchema(),
    applicationProperties.get("ProducerConfigProperties"));

  sink.setDefaultStream(outputStreamName);
  sink.setDefaultPartition("0");
  return sink;
}
```

Untuk contoh kode lengkap yang menggunakan properti runtime, lihat Memulai (DataStream API). Kode sumber untuk aplikasi Memulai tersedia di Memulai di Repositori Managed Service for Apache Flink Java Examples GitHub .

Menerapkan Toleransi Kesalahan dalam Layanan Terkelola untuk Apache Flink

Checkpointing adalah metode yang digunakan untuk menerapkan toleransi kesalahan di Amazon Managed Service untuk Apache Flink. Pos pemeriksaan adalah up-to-date cadangan dari aplikasi yang sedang berjalan yang digunakan untuk memulihkan segera dari gangguan atau kegagalan aplikasi yang tidak terduga.

Untuk detail tentang checkpointing dalam aplikasi Apache Flink, lihat Titik pemeriksaan di Dokumentasi Apache Flink.

Snapshot adalah cadangan status aplikasi yang dibuat dan dikelola secara manual. Snapshot memungkinkan Anda memulihkan aplikasi Anda ke status sebelumnya dengan memanggil `UpdateApplication`. Untuk informasi selengkapnya, lihat Mengelola Cadangan Aplikasi Menggunakan Snapshot.

Jika checkpointing diaktifkan untuk aplikasi Anda, layanan menyediakan toleransi kesalahan dengan membuat dan memuat cadangan data aplikasi jika terjadi mulai ulang aplikasi tak terduga. Mulai ulang aplikasi tak terduga ini dapat disebabkan oleh mulai ulang tugas tak terduga, kegagalan instans, dll. Ini memberi aplikasi semantik yang sama seperti eksekusi bebas kegagalan selama mulai ulang ini.

Jika snapshot diaktifkan untuk aplikasi, dan dikonfigurasi menggunakan aplikasi ApplicationRestoreConfiguration, maka layanan menyediakan semantik pemrosesan tepat sekali selama pembaruan aplikasi, atau selama penskalaan atau pemeliharaan terkait layanan.

Mengkonfigurasi Checkpointing di Managed Service untuk Apache Flink

Anda dapat mengonfigurasi perilaku checkpointing aplikasi Anda. Anda dapat menentukan apakah ini mempertahankan status checkpointing, seberapa sering status untuk titik pemeriksaan disimpan, dan interval minimum antara akhir dari satu operasi titik pemeriksaan dan awal dari operasi lainnya.

Anda mengonfigurasi pengaturan berikut menggunakan operasi API `CreateApplication` atau `UpdateApplication`:

- `CheckpointingEnabled` — Menunjukkan apakah checkpointing diaktifkan dalam aplikasi.
- `CheckpointInterval` — Berisi waktu dalam milidetik di antara operasi titik pemeriksaan (persistensi).
- `ConfigurationType` — Atur nilai ini ke `DEFAULT` untuk menggunakan perilaku checkpointing default. Atur nilai ini ke `CUSTOM` untuk mengonfigurasi nilai lainnya.

> **Note**
>
> Perilaku titik pemeriksaan default adalah sebagai berikut:
>
> - CheckpointingEnabled: benar
> - CheckpointInterval: 60000
> - MinPauseBetweenCheckpoints: 5000

Jika ConfigurationTypediatur ke`DEFAULT`, nilai sebelumnya akan digunakan, bahkan jika mereka diatur ke nilai lain menggunakan baik menggunakanAWS Command Line Interface, atau dengan menetapkan nilai-nilai dalam kode aplikasi.

> **Note**
>
> Untuk Flink 1.15 dan seterusnya, Layanan Terkelola untuk Apache Flink akan digunakan `stop-with-savepoint` selama Pembuatan Snapshot Otomatis, yaitu pembaruan aplikasi, penskalaan, atau penghentian.

- `MinPauseBetweenCheckpoints` — Waktu minimum dalam milidetik antara akhir dari satu operasi titik pemeriksaan dan awal dari operasi lainnya. Mengatur nilai ini mencegah aplikasi dari melakukan checkpointing terus-menerus ketika operasi titik pemeriksaan memakan waktu lebih lama dari `CheckpointInterval`.

Contoh API Checkpointing

Bagian ini mencakup contoh permintaan tindakan API untuk mengonfigurasi checkpointing untuk aplikasi. Untuk informasi tentang cara menggunakan file JSON untuk input tindakan API, lihat Layanan Terkelola untuk Kode Contoh API Apache Flink.

Konfigurasikan Checkpointing untuk Aplikasi Baru

Contoh permintaan untuk tindakan `CreateApplication` berikut mengonfigurasi checkpointing saat Anda membuat aplikasi:

```
{
   "ApplicationName": "MyApplication",
   "RuntimeEnvironment":"FLINK-1_15",
   "ServiceExecutionRole":"arn:aws:iam::123456789123:role/myrole",
   "ApplicationConfiguration": {
      "ApplicationCodeConfiguration":{
      "CodeContent":{
        "S3ContentLocation":{
          "BucketARN":"arn:aws:s3:::mybucket",
          "FileKey":"myflink.jar",
          "ObjectVersion":"AbCdEfGhIjKlMnOpQrStUvWxYz12345"
        }
```

```
      },
      "FlinkApplicationConfiguration": {
         "CheckpointConfiguration": {
            "CheckpointingEnabled": "true",
            "CheckpointInterval": 20000,
            "ConfigurationType": "CUSTOM",
            "MinPauseBetweenCheckpoints": 10000
         }
      }
}
```

Nonaktifkan Checkpointing untuk Aplikasi Baru

Contoh permintaan untuk tindakan `CreateApplication` berikut menonaktifkan checkpointing saat Anda membuat aplikasi:

```
{
   "ApplicationName": "MyApplication",
   "RuntimeEnvironment":"FLINK-1_15",
   "ServiceExecutionRole":"arn:aws:iam::123456789123:role/myrole",
   "ApplicationConfiguration": {
      "ApplicationCodeConfiguration":{
      "CodeContent":{
        "S3ContentLocation":{
          "BucketARN":"arn:aws:s3:::mybucket",
          "FileKey":"myflink.jar",
          "ObjectVersion":"AbCdEfGhIjKlMnOpQrStUvWxYz12345"
        }
      },
      "FlinkApplicationConfiguration": {
         "CheckpointConfiguration": {
            "CheckpointingEnabled": "false"
         }
      }
}
```

Konfigurasikan Checkpointing untuk Aplikasi yang Ada

Contoh permintaan untuk tindakan `UpdateApplication` berikut mengonfigurasi checkpointing untuk aplikasi yang ada:

```
{
```

```
   "ApplicationName": "MyApplication",
   "ApplicationConfigurationUpdate": {
      "FlinkApplicationConfigurationUpdate": {
         "CheckpointConfigurationUpdate": {
            "CheckpointingEnabledUpdate": true,
            "CheckpointIntervalUpdate": 20000,
            "ConfigurationTypeUpdate": "CUSTOM",
            "MinPauseBetweenCheckpointsUpdate": 10000
         }
      }
   }
}
```

Nonaktifkan Checkpointing untuk Aplikasi yang Ada

Contoh permintaan untuk tindakan `UpdateApplication` berikut menonaktifkan checkpointing untuk aplikasi yang ada:

```
{
   "ApplicationName": "MyApplication",
   "ApplicationConfigurationUpdate": {
      "FlinkApplicationConfigurationUpdate": {
         "CheckpointConfigurationUpdate": {
            "CheckpointingEnabledUpdate": false,
            "CheckpointIntervalUpdate": 20000,
            "ConfigurationTypeUpdate": "CUSTOM",
            "MinPauseBetweenCheckpointsUpdate": 10000
         }
      }
   }
}
```

Mengelola Cadangan Aplikasi Menggunakan Snapshot

Snapshot adalah Managed Service untuk implementasi Apache Flink dari Apache Flink Savepoint. Snapshot adalah cadangan status aplikasi yang dipicu, dibuat, dan dikelola pengguna atau layanan. Untuk informasi tentang Titik Simpan Apache Flink, lihat Titik Simpan di Dokumentasi Apache Flink. Dengan menggunakan snapshot, Anda dapat memulai ulang aplikasi dari snapshot status aplikasi tertentu.

> **Note**
>
> Sebaiknya aplikasi Anda membuat snapshot beberapa kali sehari untuk memulai ulang dengan benar menggunakan data status yang benar. Frekuensi yang benar untuk snapshot Anda bergantung pada logika bisnis aplikasi Anda. Sering mengambil snapshot memungkinkan Anda memulihkan data yang lebih baru, tetapi meningkatkan biaya dan membutuhkan lebih banyak sumber daya sistem.

Di Managed Service for Apache Flink, Anda mengelola snapshot menggunakan tindakan API berikut:

- `CreateApplicationSnapshot`
- `DeleteApplicationSnapshot`
- `DescribeApplicationSnapshot`
- `ListApplicationSnapshots`

Untuk batas per aplikasi pada jumlah snapshot, lihat Kuota. Jika aplikasi Anda mencapai batas pada snapshot, lalu secara manual membuat snapshot gagal dengan `LimitExceededException`.

Layanan Terkelola untuk Apache Flink tidak pernah menghapus snapshot. Anda harus secara manual menghapus snapshot menggunakan tindakan `DeleteApplicationSnapshot`.

Untuk memuat snapshot status aplikasi tersimpan saat memulai aplikasi, gunakan parameter `ApplicationRestoreConfiguration` dari `StartApplication` atau tindakan `UpdateApplication`.

Topik ini berisi bagian-bagian berikut:

- Pembuatan Snapshot Otomatis
- Memulihkan dari Snapshot yang Berisi Data Status yang Tidak Kompatibel
- Contoh API Snapshot

Pembuatan Snapshot Otomatis

Jika `SnapshotsEnabled` diatur ke `true` dalam untuk aplikasi, Managed Service ApplicationSnapshotConfigurationfor Apache Flink secara otomatis membuat dan menggunakan snapshot saat aplikasi diperbarui, diskalakan, atau dihentikan untuk menyediakan semantik pemrosesan yang tepat sekali.

> **Note**
>
> Mengatur `ApplicationSnapshotConfiguration::SnapshotsEnabled` ke `false` akan menyebabkan kehilangan data selama pembaruan aplikasi.

> **Note**
>
> Layanan Terkelola untuk Apache Flink memicu savepoint perantara selama pembuatan snapshot. Untuk Flink versi 1.15 atau lebih besar, savepoint menengah tidak lagi melakukan efek samping apa pun. Lihat Memicu savepoint

Snapshot yang dibuat secara otomatis memiliki kualitas berikut:

- Snapshot dikelola oleh layanan, tetapi Anda dapat melihat snapshot menggunakan tindakan. ListApplicationSnapshots Snapshot yang dibuat secara otomatis menghitung batas snapshot Anda.
- Jika aplikasi Anda melebihi batas snapshot, snapshot yang dibuat secara manual akan gagal, tetapi Layanan Terkelola untuk layanan Apache Flink akan tetap berhasil membuat snapshot saat aplikasi diperbarui, diskalakan, atau dihentikan. Anda harus menghapus snapshot secara manual menggunakan DeleteApplicationSnapshottindakan sebelum membuat lebih banyak snapshot secara manual.

Memulihkan dari Snapshot yang Berisi Data Status yang Tidak Kompatibel

Karena snapshot berisi informasi tentang operator, memulihkan data status dari snapshot untuk operator yang telah berubah sejak versi aplikasi sebelumnya mungkin memiliki hasil yang tak terduga. Aplikasi akan gagal jika mencoba memulihkan data status dari snapshot yang tidak sesuai dengan operator saat ini. Aplikasi yang gagal akan terhenti di status `STOPPING` atau `UPDATING`.

Untuk memungkinkan aplikasi memulihkan dari snapshot yang berisi data status yang tidak kompatibel, atur `AllowNonRestoredState` parameter FlinkRunConfigurationuntuk `true` menggunakan tindakan. UpdateApplication

Anda akan melihat perilaku berikut ketika aplikasi dipulihkan dari snapshot usang:

- Operator ditambahkan: Jika operator baru ditambahkan, titik simpan tidak memiliki data status untuk operator baru. Tidak ada kesalahan yang akan terjadi, dan tidak perlu untuk mengatur `AllowNonRestoredState`.
- Operator dihapus: Jika operator yang ada dihapus, titik simpan memiliki data status untuk operator yang hilang. Kesalahan akan terjadi kecuali `AllowNonRestoredState` diatur ke `true`.
- Operator dimodifikasi: Jika perubahan yang kompatibel dibuat, seperti mengubah tipe parameter ke tipe yang kompatibel, aplikasi dapat memulihkan dari snapshot usang. Untuk informasi selengkapnya tentang pemulihan dari snapshot, lihat Titik Simpan di Dokumentasi Apache Flink. Aplikasi yang menggunakan Apache Flink versi 1.8 atau yang lebih baru mungkin dapat dipulihkan dari snapshot dengan skema yang berbeda. Aplikasi yang menggunakan Apache Flink versi 1.6 tidak dapat dipulihkan. Untuk two-phase-commit sink, sebaiknya gunakan snapshot sistem (SWs) alih-alih snapshot () yang dibuat pengguna. CreateApplicationSnapshot

  Untuk Flink, Layanan Terkelola untuk Apache Flink memicu savepoint perantara selama pembuatan snapshot. Untuk Flink 1.15 dan seterusnya, savepoint menengah tidak lagi melakukan efek samping apa pun. Lihat Memicu Savepoint.

Jika Anda perlu melanjutkan aplikasi yang tidak kompatibel dengan data savepoint yang ada, sebaiknya Anda melewatkan pemulihan dari snapshot dengan menyetel `ApplicationRestoreType` parameter tindakan ke. StartApplication`SKIP_RESTORE_FROM_SNAPSHOT`

Untuk informasi selengkapnya tentang cara Apache Flink menangai data status yang tidak kompatibel, lihat Evolusi Skema Status di Dokumentasi Apache Flink.

Contoh API Snapshot

Bagian ini mencakup permintaan contoh tindakan API untuk menggunakan snapshot dengan aplikasi. Untuk informasi tentang cara menggunakan file JSON untuk input tindakan API, lihat Layanan Terkelola untuk Kode Contoh API Apache Flink.

Aktifkan Snapshot untuk Aplikasi

Contoh permintaan untuk tindakan `UpdateApplication` berikut mengaktifkan snapshot untuk aplikasi:

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 1,
```

```
   "ApplicationConfigurationUpdate": {
      "ApplicationSnapshotConfigurationUpdate": {
         "SnapshotsEnabledUpdate": "true"
       }
    }
}
```

Membuat Snapshot

Contoh permintaan untuk tindakan `CreateApplicationSnapshot` berikut membuat snapshot dari status aplikasi saat ini:

```
{
   "ApplicationName": "MyApplication",
   "SnapshotName": "MyCustomSnapshot"
}
```

Cantumkan Snapshot untuk Aplikasi

Contoh permintaan untuk tindakan `ListApplicationSnapshots` berikut mencantumkan 50 snapshot pertama untuk status aplikasi saat ini:

```
{
   "ApplicationName": "MyApplication",
   "Limit": 50
}
```

Cantumkan Detail untuk Snapshot Aplikasi

Contoh permintaan berikut untuk tindakan `DescribeApplicationSnapshot` mencantumkan detail untuk snapshot aplikasi tertentu:

```
{
   "ApplicationName": "MyApplication",
   "SnapshotName": "MyCustomSnapshot"
}
```

Hapus Snapshot

Contoh permintaan berikut untuk tindakan `DeleteApplicationSnapshot` menghapus snapshot yang disimpan sebelumnya. Anda bisa mendapatkan nilai `SnapshotCreationTimestamp` menggunakan `ListApplicationSnapshots` atau `DeleteApplicationSnapshot`:

```
{
   "ApplicationName": "MyApplication",
   "SnapshotName": "MyCustomSnapshot",
   "SnapshotCreationTimestamp": 12345678901.0,
}
```

Mulai Ulang Aplikasi Menggunakan Snapshot yang Diberi Nama

Contoh permintaan berikut untuk tindakan StartApplication memulai aplikasi menggunakan status yang disimpan dari snapshot tertentu:

```
{
   "ApplicationName": "MyApplication",
   "RunConfiguration": {
      "ApplicationRestoreConfiguration": {
         "ApplicationRestoreType": "RESTORE_FROM_CUSTOM_SNAPSHOT",
         "SnapshotName": "MyCustomSnapshot"
      }
   }
}
```

Mulai Ulang Aplikasi Menggunakan Snapshot Terbaru

Contoh permintaan berikut untuk tindakan StartApplication memulai aplikasi menggunakan snapshot terbaru:

```
{
   "ApplicationName": "MyApplication",
   "RunConfiguration": {
      "ApplicationRestoreConfiguration": {
         "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"
      }
   }
}
```

Mulai Ulang Aplikasi Tanpa Menggunakan Snapshot

Contoh permintaan berikut untuk tindakan StartApplication memulai aplikasi tanpa memuat status aplikasi, bahkan jika snapshot tersedia:

```
{
```

```
   "ApplicationName": "MyApplication",
   "RunConfiguration": {
      "ApplicationRestoreConfiguration": {
         "ApplicationRestoreType": "SKIP_RESTORE_FROM_SNAPSHOT"
      }
   }
}
```

Penskalaan Aplikasi dalam Layanan Terkelola untuk Apache Flink

Anda dapat mengonfigurasi eksekusi tugas secara paralel dan alokasi sumber daya untuk Amazon Managed Service untuk Apache Flink untuk mengimplementasikan penskalaan. Untuk informasi tentang cara Apache Flink menjadwalkan instans tugas paralel, lihat Eksekusi Paralel di Dokumentasi Apache Flink.

Topik

- Konfigurasi Paralelisme Aplikasi dan KPU ParallelismPer
- Mengalokasikan Unit Pemrosesan Kinesis
- Memperbarui Paralelisme Aplikasi Anda
- Penskalaan Otomatis

Konfigurasi Paralelisme Aplikasi dan KPU ParallelismPer

Anda mengonfigurasi eksekusi paralel untuk tugas aplikasi Managed Service for Apache Flink (seperti membaca dari sumber atau mengeksekusi operator) menggunakan properti berikut: `ParallelismConfiguration`

- `Parallelism` — Gunakan properti ini untuk mengatur paralelisme aplikasi Apache Flink default. Semua operator, sumber, dan sink mengeksekusi dengan paralelisme ini kecuali ditimpa dalam kode aplikasi. Default-nya adalah `1`, dan maksimum default adalah `256`.
- `ParallelismPerKPU` — Gunakan properti ini untuk mengatur jumlah tugas paralel yang dapat dijadwalkan per Unit Pemrosesan Kinesis (KPU) aplikasi Anda. Default-nya adalah `1`, dan maksimumnya adalah `8`. Untuk aplikasi yang memiliki operasi pemblokiran (misalnya, I/O), nilai `ParallelismPerKPU` yang lebih tinggi mengarah pada penggunaan penuh sumber daya KPU.

> **Note**
>
> Batas untuk `Parallelism` sama dengan `ParallelismPerKPU` kali batas untuk KPU (yang memiliki default 64). Batas KPU dapat ditingkatkan dengan meminta peningkatan batas. Untuk petunjuk tentang cara meminta peningkatan batas ini, lihat "Untuk meminta peningkatan batas" di Service Quotas.

Untuk informasi tentang pengaturan paralelisme tugas, lihat Mengatur Paralelisme: Operator di Dokumentasi Apache Flink.

Mengalokasikan Unit Pemrosesan Kinesis

Managed Service untuk Apache Flink menyediakan kapasitas sebagai KPU. Satu KPU memberi Anda 1 vCPU dan memori 4 GB. Untuk setiap KPU yang dialokasikan, penyimpanan aplikasi berjalan sebesar 50 GB juga disediakan.

Managed Service for Apache Flink menghitung KPU yang diperlukan untuk menjalankan aplikasi Anda menggunakan `Parallelism` dan `ParallelismPerKPU` properti, sebagai berikut:

```
Allocated KPUs for the application = Parallelism/ParallelismPerKPU
```

Layanan Terkelola untuk Apache Flink dengan cepat memberikan sumber daya aplikasi Anda sebagai respons terhadap lonjakan throughput atau aktivitas pemrosesan. Ini akan menghapus sumber daya dari aplikasi Anda secara bertahap setelah lonjakan aktivitas telah berlalu. Untuk menonaktifkan alokasi otomatis sumber daya, atur nilai `AutoScalingEnabled` ke `false`, seperti yang dijelaskan nanti di Memperbarui Paralelisme Aplikasi Anda.

Batas default untuk KPU untuk aplikasi Anda adalah 64. Untuk petunjuk tentang cara meminta peningkatan batas ini, lihat "Untuk meminta peningkatan batas" di Service Quotas.

> **Note**
>
> KPU tambahan dikenakan biaya untuk keperluan orkestrasi. Untuk informasi selengkapnya, lihat Layanan Terkelola untuk harga Apache Flink.

Memperbarui Paralelisme Aplikasi Anda

Bagian ini berisi permintaan sampel untuk tindakan API yang mengatur paralelisme aplikasi. Untuk contoh dan petunjuk selengkapnya tentang cara menggunakan blok permintaan dengan tindakan API, lihat Layanan Terkelola untuk Kode Contoh API Apache Flink.

Permintaan contoh berikut untuk tindakan `CreateApplication` mengatur paralelisme saat Anda membuat aplikasi:

```
{
   "ApplicationName": "string",
   "RuntimeEnvironment":"FLINK-1_15",
   "ServiceExecutionRole":"arn:aws:iam::123456789123:role/myrole",
   "ApplicationConfiguration": {
      "ApplicationCodeConfiguration":{
      "CodeContent":{
         "S3ContentLocation":{
            "BucketARN":"arn:aws:s3:::mybucket",
            "FileKey":"myflink.jar",
            "ObjectVersion":"AbCdEfGhIjKlMnOpQrStUvWxYz12345"
            }
         },
      "CodeContentType":"ZIPFILE"
   },
      "FlinkApplicationConfiguration": {
         "ParallelismConfiguration": {
            "AutoScalingEnabled": "true",
            "ConfigurationType": "CUSTOM",
            "Parallelism": 4,
            "ParallelismPerKPU": 4
         }
      }
   }
}
```

Permintaan contoh berikut untuk tindakan `UpdateApplication` mengatur paralelisme untuk aplikasi yang sudah ada:

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 4,
   "ApplicationConfigurationUpdate": {
```

```
      "FlinkApplicationConfigurationUpdate": {
         "ParallelismConfigurationUpdate": {
            "AutoScalingEnabledUpdate": "true",
            "ConfigurationTypeUpdate": "CUSTOM",
            "ParallelismPerKPUUpdate": 4,
            "ParallelismUpdate": 4
         }
      }
   }
}
```

Permintaan contoh berikut untuk tindakan `UpdateApplication` menonaktifkan paralelisme untuk aplikasi yang sudah ada:

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 4,
   "ApplicationConfigurationUpdate": {
      "FlinkApplicationConfigurationUpdate": {
         "ParallelismConfigurationUpdate": {
            "AutoScalingEnabledUpdate": "false"
         }
      }
   }
}
```

Penskalaan Otomatis

Layanan Terkelola untuk Apache Flink secara elastis menskalakan paralelisme aplikasi Anda untuk mengakomodasi throughput data sumber Anda dan kompleksitas operator Anda untuk sebagian besar skenario. Layanan Terkelola untuk Apache Flink memantau penggunaan sumber daya (CPU) aplikasi Anda, dan secara elastis menskalakan paralelisme aplikasi Anda ke atas atau ke bawah sesuai dengan itu:

- Aplikasi Anda meningkatkan skala (meningkatkan paralelisme) jika CloudWatch metrik `containerCPUUtilization` lebih besar dari 75 persen atau lebih selama 15 menit. Itu berarti `ScaleUp` tindakan dipicu ketika ada 15 titik data berturut-turut dengan periode 1 menit sama dengan atau lebih dari 75 persen.

- Aplikasi Anda mengurangi (mengurangi paralelisme) ketika penggunaan CPU Anda tetap di bawah 10 persen selama enam jam. Itu berarti `ScaleDown` tindakan dipicu ketika ada 360 titik data berturut-turut dengan periode 1 menit kurang dari 10 persen.

> **Note**
>
> Maks periode `containerCPUUtilization` lebih dari 1 menit dapat direferensikan untuk menemukan korelasi dengan titik data yang digunakan untuk tindakan Penskalaan, tetapi tidak perlu untuk mencerminkan momen yang tepat ketika tindakan dipicu.

Layanan Terkelola untuk Apache Flink tidak akan mengurangi `CurrentParallelism` nilai aplikasi Anda menjadi kurang dari pengaturan aplikasi Anda. `Parallelism`

Ketika Layanan Terkelola untuk layanan Apache Flink menskalakan aplikasi Anda, itu akan berada dalam status. `AUTOSCALING` Anda dapat memeriksa status aplikasi Anda saat ini menggunakan ListApplicationstindakan DescribeApplicationatau. Saat layanan menskalakan aplikasi Anda, satu-satunya tindakan API valid yang dapat Anda gunakan adalah StopApplicationdengan `Force` parameter yang disetel `ketrue`.

Anda dapat menggunakan properti `AutoScalingEnabled` (bagian dari `FlinkApplicationConfiguration`) untuk mengaktifkan atau menonaktifkan perilaku penskalaan otomatis. AWSAkun Anda dikenakan biaya untuk KPU yang Managed Service untuk ketentuan Apache Flink yang merupakan fungsi dari aplikasi dan pengaturan Anda. `parallelism` `parallelismPerKPU` Lonjakan aktivitas meningkatkan Layanan Terkelola Anda untuk biaya Apache Flink.

Untuk informasi tentang harga, lihat Amazon Managed Service untuk harga Apache Flink.

Perhatikan hal tentang penskalaan aplikasi berikut:

- Penskalaan otomatis diaktifkan secara default.
- Penskalaan tidak berlaku untuk notebook Studio. Namun, jika Anda men-deploy notebook Studio sebagai aplikasi dengan status tahan lama, penskalaan akan diterapkan ke aplikasi yang di-deploy.
- Aplikasi Anda memiliki batas default 64 KPU. Untuk informasi selengkapnya, lihat Kuota.
- Saat penskalaan otomatis memperbarui paralelisme aplikasi, aplikasi mengalami waktu henti. Untuk menghindari waktu henti ini, lakukan hal berikut:
  - Nonaktifkan penskalaan otomatis

- Konfigurasikan aplikasi Anda `parallelism` dan `parallelismPerKPU` dengan UpdateApplicationtindakan. Untuk informasi selengkapnya tentang menetapkan pengaturan paralelisme aplikasi Anda, lihat the section called “Memperbarui Paralelisme Aplikasi Anda” di bawah ini.
- Pantau penggunaan sumber daya aplikasi Anda secara berkala untuk memverifikasi bahwa aplikasi Anda memiliki pengaturan paralelisme yang benar untuk beban kerjanya. Untuk informasi tentang pemantauan penggunaan sumber daya alokasi, lihat the section called “Metrik dan Dimensi dalam Layanan Terkelola untuk Apache Flink”.

Pertimbangan MaxParallelism

- Logika skala otomatis akan mencegah penskalaan pekerjaan Flink ke paralelisme yang akan menyebabkan gangguan pada pekerjaan dan operator. `maxParallelism` Misalnya, jika pekerjaan sederhana dengan hanya sumber dan wastafel di mana sumbernya memiliki `maxParallelism` 16 dan `sink` memiliki 8, kami tidak akan melakukan penskalaan otomatis pekerjaan ke atas 8.
- Jika tidak `maxParallelism` disetel untuk pekerjaan, Flink akan default ke 128. Oleh karena itu, jika Anda berpikir bahwa suatu pekerjaan perlu berjalan pada paralelisme yang lebih tinggi dari 128, Anda harus menetapkan nomor itu untuk aplikasi Anda.
- Jika Anda berharap untuk melihat skala otomatis pekerjaan Anda tetapi tidak melihatnya, pastikan `maxParallelism` nilai-nilai Anda memungkinkan untuk itu.

Untuk informasi tambahan, lihat Pemantauan yang ditingkatkan dan penskalaan otomatis untuk Apache Flink

Sebagai contoh, lihat kda-flink-app-autoscaling.

Menggunakan Penandaan

Bagian ini menjelaskan cara menambahkan tag metadata nilai kunci ke Layanan Terkelola untuk aplikasi Apache Flink. Tanda ini dapat digunakan untuk tujuan berikut:

- Menentukan penagihan untuk Layanan Terkelola individual untuk aplikasi Apache Flink. Untuk informasi selengkapnya, lihat Menggunakan Tanda Alokasi Biaya dalam Panduan Manajemen Penagihan dan Biaya.

- Mengontrol akses ke sumber daya aplikasi berdasarkan tanda. Untuk informasi selengkapnya, lihat Mengontrol Akses Menggunakan Tanda di Panduan Pengguna AWS Identity and Access Management.
- Tujuan yang ditentukan pengguna. Anda dapat menentukan fungsi aplikasi berdasarkan adanya tanda pengguna.

Catat informasi berikut tentang penandaan:

- Jumlah maksimum tanda aplikasi termasuk tanda sistem. Jumlah maksimum tanda aplikasi yang ditentukan pengguna adalah 50.
- Jika tindakan berisi daftar tanda yang memiliki nilai `Key` duplikat, layanan melempar `InvalidArgumentException`.

Topik ini berisi bagian-bagian berikut:

Menambahkan Tanda saat Aplikasi dibuat

Anda menambahkan tag saat membuat aplikasi menggunakan `tags` parameter CreateApplicationtindakan.

Contoh permintaan berikut menunjukkan simpul `Tags` untuk permintaan `CreateApplication`:

```
"Tags": [
    {
        "Key": "Key1",
        "Value": "Value1"
    },
    {
        "Key": "Key2",
        "Value": "Value2"
    }
]
```

Menambahkan atau Memperbarui Tanda untuk Aplikasi yang Ada

Anda menambahkan tag ke aplikasi menggunakan TagResourcetindakan. Anda tidak dapat menambahkan tag ke aplikasi menggunakan UpdateApplicationtindakan.

Untuk memperbarui tanda yang ada, tambahkan tanda dengan kunci yang sama dengan tanda yang ada.

Contoh permintaan berikut untuk tindakan `TagResource` menambahkan tanda baru atau memperbarui tanda yang ada:

```
{
   "ResourceARN": "string",
   "Tags": [
      {
         "Key": "NewTagKey",
         "Value": "NewTagValue"
      },
      {
         "Key": "ExistingKeyOfTagToUpdate",
         "Value": "NewValueForExistingTag"
      }
   ]
}
```

Mencantumkan Tanda untuk Aplikasi

Untuk mencantumkan tag yang ada, Anda menggunakan ListTagsForResourcetindakan.

Contoh permintaan berikut untuk tindakan `ListTagsForResource` mencantumkan tanda untuk aplikasi:

```
{
   "ResourceARN": "arn:aws:kinesisanalyticsus-west-2:012345678901:application/
MyApplication"
}
```

Menghapus Tanda dari Aplikasi

Untuk menghapus tag dari aplikasi, Anda menggunakan UntagResourcetindakan.

Contoh permintaan berikut untuk tindakan `UntagResource` menghapus tanda dari aplikasi:

```
{
   "ResourceARN": "arn:aws:kinesisanalyticsus-west-2:012345678901:application/
MyApplication",
   "TagKeys": [ "KeyOfFirstTagToRemove", "KeyOfSecondTagToRemove" ]
}
```

Menggunakan CloudFormation dengan Managed Service untuk Apache Flink

Latihan berikut menunjukkan cara memulai aplikasi Flink yang dibuat melalui AWS CloudFormation menggunakan fungsi Lambda di tumpukan yang sama.

Sebelum Anda memulai

Sebelum Anda memulai latihan ini, ikuti langkah-langkah untuk membuat aplikasi Flink menggunakan AWS CloudFormation at AWS::KinesisAnalytics::Application.

Menulis fungsi Lambda

Untuk memulai aplikasi Flink setelah pembuatan atau pembaruan, kami menggunakan kinesisanalyticsv2 start-application API. Panggilan akan dipicu oleh AWS CloudFormation peristiwa setelah pembuatan aplikasi Flink. Kita akan membahas cara mengatur tumpukan untuk memicu fungsi Lambda nanti dalam latihan ini, tetapi pertama-tama kita fokus pada deklarasi fungsi Lambda dan kodenya. Kami menggunakan `Python3.8` runtime dalam contoh ini.

```
StartApplicationLambda:
    Type: AWS::Lambda::Function
    DependsOn: StartApplicationLambdaRole
    Properties:
      Description: Starts an application when invoked.
      Runtime: python3.8
      Role: !GetAtt StartApplicationLambdaRole.Arn
      Handler: index.lambda_handler
      Timeout: 30
      Code:
        ZipFile: |
          import logging
          import cfnresponse
          import boto3
```

```
          logger = logging.getLogger()
          logger.setLevel(logging.INFO)

          def lambda_handler(event, context):
            logger.info('Incoming CFN event {}'.format(event))

            try:
              application_name = event['ResourceProperties']['ApplicationName']

              # filter out events other than Create or Update,
              # you can also omit Update in order to start an application on Create
 only.
              if event['RequestType'] not in ["Create", "Update"]:
                logger.info('No-op for Application {} because CFN RequestType {} is
 filtered'.format(application_name, event['RequestType']))
                cfnresponse.send(event, context, cfnresponse.SUCCESS, {})

                return

              # use kinesisanalyticsv2 API to start an application.
              client_kda = boto3.client('kinesisanalyticsv2',
 region_name=event['ResourceProperties']['Region'])

              # get application status.
              describe_response =
 client_kda.describe_application(ApplicationName=application_name)
              application_status = describe_response['ApplicationDetail']
['ApplicationStatus']

              # an application can be started from 'READY' status only.
              if application_status != 'READY':
                logger.info('No-op for Application {} because ApplicationStatus {} is
 filtered'.format(application_name, application_status))
                cfnresponse.send(event, context, cfnresponse.SUCCESS, {})

                return

              # create RunConfiguration.
              run_configuration = {
                'ApplicationRestoreConfiguration': {
                  'ApplicationRestoreType': 'RESTORE_FROM_LATEST_SNAPSHOT',
                }
              }
```

```
            logger.info('RunConfiguration for Application {}:
{}'.format(application_name, run_configuration))

            # this call doesn't wait for an application to transfer to 'RUNNING'
state.
            client_kda.start_application(ApplicationName=application_name,
RunConfiguration=run_configuration)

            logger.info('Started Application: {}'.format(application_name))
            cfnresponse.send(event, context, cfnresponse.SUCCESS, {})
          except Exception as err:
            logger.error(err)
            cfnresponse.send(event,context, cfnresponse.FAILED, {"Data": str(err)})
```

Dalam kode sebelumnya, Lambda akan memproses AWS CloudFormation peristiwa yang masuk, menyaring semuanya selain `Create` dan`Update`, mendapatkan status aplikasi dan memulainya jika statusnya. `READY` Untuk mendapatkan status aplikasi, Anda perlu membuat peran Lambda, seperti yang ditunjukkan berikut:

Membuat peran Lambda

Anda membuat peran agar Lambda berhasil “berbicara” dengan aplikasi dan menulis log. Peran ini akan menggunakan kebijakan terkelola default, tetapi Anda mungkin ingin mempersempitnya menggunakan kebijakan khusus.

```
StartApplicationLambdaRole:
    Type: AWS::IAM::Role
    DependsOn: TestFlinkApplication
    Properties:
      Description: A role for lambda to use while interacting with an application.
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - lambda.amazonaws.com
            Action:
              - sts:AssumeRole
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/Amazonmanaged-flinkFullAccess
        - arn:aws:iam::aws:policy/CloudWatchLogsFullAccess
```

```
      Path: /
```

Perhatikan bahwa sumber daya Lambda akan dibuat setelah pembuatan aplikasi Flink di tumpukan yang sama karena mereka bergantung padanya.

Memanggil fungsi Lambda

Sekarang yang tersisa hanyalah memanggil fungsi Lambda. Ini dilakukan dengan menggunakan sumber daya khusus.

```
StartApplicationLambdaInvoke:
    Description: Invokes StartApplicationLambda to start an application.
    Type: AWS::CloudFormation::CustomResource
    DependsOn: StartApplicationLambda
    Version: "1.0"
    Properties:
      ServiceToken: !GetAtt StartApplicationLambda.Arn
      Region: !Ref AWS::Region
      ApplicationName: !Ref TestFlinkApplication
```

Ini semua yang Anda butuhkan untuk memulai aplikasi Flink Anda menggunakan Lambda. Anda sekarang siap untuk membuat tumpukan Anda sendiri atau menggunakan contoh lengkap di bawah ini untuk melihat bagaimana semua langkah tersebut bekerja dalam praktik.

Contoh lengkap

Contoh berikut adalah versi yang sedikit diperluas dari langkah-langkah di atas dengan `RunConfiguration` penyesuaian tambahan yang dilakukan melalui parameter template. Ini adalah tumpukan kerja untuk Anda coba. Pastikan untuk membaca catatan yang menyertainya:

tumpukan.yaml

```
Description: 'kinesisanalyticsv2 CloudFormation Test Application'
Parameters:
  ApplicationRestoreType:
    Description: ApplicationRestoreConfiguration option, can
 be SKIP_RESTORE_FROM_SNAPSHOT, RESTORE_FROM_LATEST_SNAPSHOT or
 RESTORE_FROM_CUSTOM_SNAPSHOT.
    Type: String
    Default: SKIP_RESTORE_FROM_SNAPSHOT
    AllowedValues: [ SKIP_RESTORE_FROM_SNAPSHOT, RESTORE_FROM_LATEST_SNAPSHOT,
 RESTORE_FROM_CUSTOM_SNAPSHOT ]
```

```
  SnapshotName:
    Description: ApplicationRestoreConfiguration option, name of a snapshot to restore
 to, used with RESTORE_FROM_CUSTOM_SNAPSHOT ApplicationRestoreType.
    Type: String
    Default: ''
  AllowNonRestoredState:
    Description: FlinkRunConfiguration option, can be true or false.
    Default: true
    Type: String
    AllowedValues: [ true, false ]
  CodeContentBucketArn:
    Description: ARN of a bucket with application code.
    Type: String
  CodeContentFileKey:
    Description: A jar filename with an application code inside a bucket.
    Type: String
Conditions:
  IsSnapshotNameEmpty: !Equals [ !Ref SnapshotName, '' ]
Resources:
  TestServiceExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - kinesisanlaytics.amazonaws.com
            Action: sts:AssumeRole
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/AmazonKinesisFullAccess
        - arn:aws:iam::aws:policy/AmazonS3FullAccess
      Path: /
  InputKinesisStream:
    Type: AWS::Kinesis::Stream
    Properties:
      ShardCount: 1
  OutputKinesisStream:
    Type: AWS::Kinesis::Stream
    Properties:
      ShardCount: 1
  TestFlinkApplication:
    Type: 'AWS::kinesisanalyticsv2::Application'
```

```
    Properties:
      ApplicationName: 'CFNTestFlinkApplication'
      ApplicationDescription: 'Test Flink Application'
      RuntimeEnvironment: 'FLINK-1_15'
      ServiceExecutionRole: !GetAtt TestServiceExecutionRole.Arn
      ApplicationConfiguration:
        EnvironmentProperties:
          PropertyGroups:
            - PropertyGroupId: 'KinesisStreams'
              PropertyMap:
                INPUT_STREAM_NAME: !Ref InputKinesisStream
                OUTPUT_STREAM_NAME: !Ref OutputKinesisStream
                AWS_REGION: !Ref AWS::Region
        FlinkApplicationConfiguration:
          CheckpointConfiguration:
            ConfigurationType: 'CUSTOM'
            CheckpointingEnabled: True
            CheckpointInterval: 1500
            MinPauseBetweenCheckpoints: 500
          MonitoringConfiguration:
            ConfigurationType: 'CUSTOM'
            MetricsLevel: 'APPLICATION'
            LogLevel: 'INFO'
          ParallelismConfiguration:
            ConfigurationType: 'CUSTOM'
            Parallelism: 1
            ParallelismPerKPU: 1
            AutoScalingEnabled: True
        ApplicationSnapshotConfiguration:
          SnapshotsEnabled: True
        ApplicationCodeConfiguration:
          CodeContent:
            S3ContentLocation:
              BucketARN: !Ref CodeContentBucketArn
              FileKey: !Ref CodeContentFileKey
          CodeContentType: 'ZIPFILE'
  StartApplicationLambdaRole:
    Type: AWS::IAM::Role
    DependsOn: TestFlinkApplication
    Properties:
      Description: A role for lambda to use while interacting with an application.
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
```

```
          - Effect: Allow
            Principal:
              Service:
                - lambda.amazonaws.com
            Action:
              - sts:AssumeRole
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/Amazonmanaged-flinkFullAccess
        - arn:aws:iam::aws:policy/CloudWatchLogsFullAccess
      Path: /
  StartApplicationLambda:
    Type: AWS::Lambda::Function
    DependsOn: StartApplicationLambdaRole
    Properties:
      Description: Starts an application when invoked.
      Runtime: python3.8
      Role: !GetAtt StartApplicationLambdaRole.Arn
      Handler: index.lambda_handler
      Timeout: 30
      Code:
        ZipFile: |
          import logging
          import cfnresponse
          import boto3

          logger = logging.getLogger()
          logger.setLevel(logging.INFO)

          def lambda_handler(event, context):
            logger.info('Incoming CFN event {}'.format(event))

            try:
              application_name = event['ResourceProperties']['ApplicationName']

              # filter out events other than Create or Update,
              # you can also omit Update in order to start an application on Create
only.
              if event['RequestType'] not in ["Create", "Update"]:
                logger.info('No-op for Application {} because CFN RequestType {} is
filtered'.format(application_name, event['RequestType']))
                cfnresponse.send(event, context, cfnresponse.SUCCESS, {})

                return
```

```
            # use kinesisanalyticsv2 API to start an application.
            client_kda = boto3.client('kinesisanalyticsv2',
 region_name=event['ResourceProperties']['Region'])

            # get application status.
            describe_response =
 client_kda.describe_application(ApplicationName=application_name)
            application_status = describe_response['ApplicationDetail']
['ApplicationStatus']

            # an application can be started from 'READY' status only.
            if application_status != 'READY':
              logger.info('No-op for Application {} because ApplicationStatus {} is
 filtered'.format(application_name, application_status))
              cfnresponse.send(event, context, cfnresponse.SUCCESS, {})

              return

            # create RunConfiguration from passed parameters.
            run_configuration = {
              'FlinkRunConfiguration': {
                'AllowNonRestoredState': event['ResourceProperties']
['AllowNonRestoredState'] == 'true'
              },
              'ApplicationRestoreConfiguration': {
                'ApplicationRestoreType': event['ResourceProperties']
['ApplicationRestoreType'],
              }
            }

            # add SnapshotName to RunConfiguration if specified.
            if event['ResourceProperties']['SnapshotName'] != '':
              run_configuration['ApplicationRestoreConfiguration']['SnapshotName'] =
 event['ResourceProperties']['SnapshotName']

            logger.info('RunConfiguration for Application {}:
 {}'.format(application_name, run_configuration))

            # this call doesn't wait for an application to transfer to 'RUNNING'
 state.
            client_kda.start_application(ApplicationName=application_name,
 RunConfiguration=run_configuration)

            logger.info('Started Application: {}'.format(application_name))
```

```
            cfnresponse.send(event, context, cfnresponse.SUCCESS, {})
          except Exception as err:
            logger.error(err)
            cfnresponse.send(event,context, cfnresponse.FAILED, {"Data": str(err)})
  StartApplicationLambdaInvoke:
    Description: Invokes StartApplicationLambda to start an application.
    Type: AWS::CloudFormation::CustomResource
    DependsOn: StartApplicationLambda
    Version: "1.0"
    Properties:
      ServiceToken: !GetAtt StartApplicationLambda.Arn
      Region: !Ref AWS::Region
      ApplicationName: !Ref TestFlinkApplication
      ApplicationRestoreType: !Ref ApplicationRestoreType
      SnapshotName: !Ref SnapshotName
      AllowNonRestoredState: !Ref AllowNonRestoredState
```

Sekali lagi, Anda mungkin ingin menyesuaikan peran untuk Lambda serta aplikasi itu sendiri.

Sebelum membuat tumpukan di atas, jangan lupa untuk menentukan parameter Anda.

parameters.json

```
[
  {
    "ParameterKey": "CodeContentBucketArn",
    "ParameterValue": "YOUR_BUCKET_ARN"
  },
  {
    "ParameterKey": "CodeContentFileKey",
    "ParameterValue": "YOUR_JAR"
  },
  {
    "ParameterKey": "ApplicationRestoreType",
    "ParameterValue": "SKIP_RESTORE_FROM_SNAPSHOT"
  },
  {
    "ParameterKey": "AllowNonRestoredState",
    "ParameterValue": "true"
  }
]
```

Ganti YOUR_BUCKET_ARN dan YOUR_JAR dengan kebutuhan spesifik Anda. Anda dapat mengikuti panduan ini untuk membuat ember Amazon S3 dan toples aplikasi.

Sekarang buat tumpukan (ganti YOUR_REGION dengan wilayah pilihan Anda, misalnya us-east-1):

```
aws cloudformation create-stack --region YOUR_REGION --template-body "file://
stack.yaml" --parameters "file://parameters.json" --stack-name "TestManaged Service for
 Apache FlinkStack" --capabilities CAPABILITY_NAMED_IAM
```

Anda sekarang dapat menavigasi ke https://console.aws.amazon.com/cloudformation dan melihat kemajuannya. Setelah dibuat, Anda akan melihat aplikasi Flink Anda dalam `Starting` keadaan. Mungkin perlu beberapa menit sampai dimulai`Running`.

Untuk informasi selengkapnya, lihat yang berikut:

- Empat cara untuk mengambil properti AWS layanan apa pun menggunakan AWS CloudFormation (Bagian 1 dari 3).
- Panduan: Mencari ID Gambar Mesin Amazon.

Menggunakan Apache Flink Dashboard dengan Managed Service untuk Apache Flink

Anda dapat menggunakan Apache Flink Dashboard aplikasi Anda untuk memantau Layanan Terkelola Anda untuk kesehatan aplikasi Apache Flink. Dasbor aplikasi Anda menunjukkan informasi berikut:

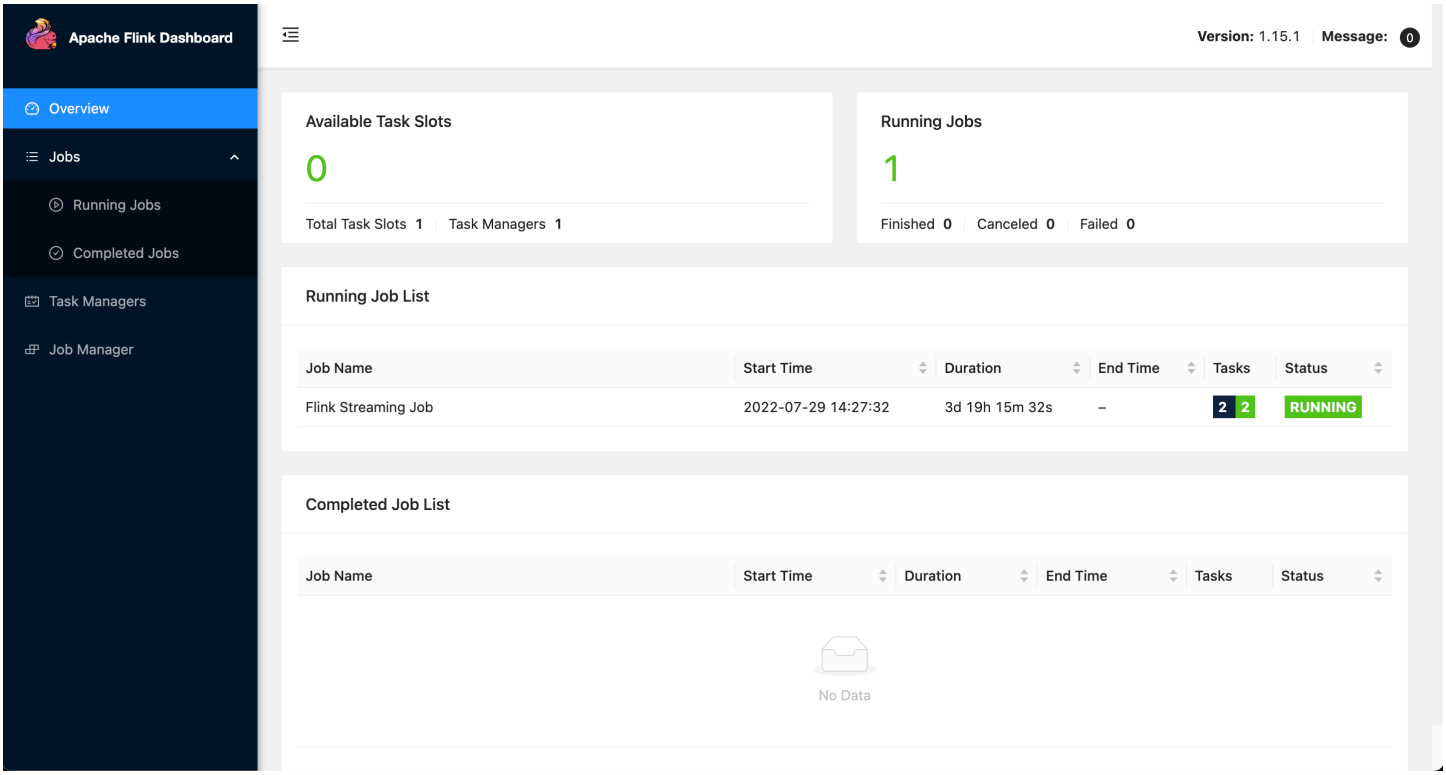
- Sumber daya yang digunakan, termasuk Manajer Tugas dan Slot Tugas.
- Informasi tentang Tugas, termasuk yang berjalan, selesai, dibatalkan, dan gagal.

Untuk informasi tentang Manajer Tugas, Slot Tugas, dan Tugas Apache Flink, lihat Arsitektur Apache Flink di situs web Apache Flink.

Perhatikan hal berikut tentang menggunakan Apache Flink Dashboard dengan Managed Service untuk aplikasi Apache Flink:

- Dasbor Apache Flink untuk Layanan Terkelola untuk aplikasi Apache Flink hanya bisa dibaca. Anda tidak dapat membuat perubahan pada aplikasi Managed Service for Apache Flink menggunakan Apache Flink Dashboard.

- Dasbor Apache Flink tidak kompatibel dengan Microsoft Internet Explorer.



Mengakses Dasbor Apache Flink Aplikasi Anda

Anda dapat mengakses Apache Flink Dashboard aplikasi Anda baik melalui Managed Service for Apache Flink console, atau dengan meminta endpoint URL aman menggunakan CLI.

Mengakses Dasbor Apache Flink Aplikasi Anda Menggunakan Layanan Terkelola untuk Apache Flink Console

Untuk mengakses Dasbor Apache Flink aplikasi Anda dari konsol, pilih Apache Flink Dashboard (Dasbor Apache Flink) di halaman aplikasi Anda.

> **Note**
>
> Saat Anda membuka dasbor dari Layanan Terkelola untuk konsol Apache Flink, URL yang dihasilkan konsol akan berlaku selama 12 jam.

Mengakses Apache Flink Dashboard aplikasi Anda menggunakan Managed Service for Apache Flink CLI

Anda dapat menggunakan Managed Service for Apache Flink CLI untuk menghasilkan URL untuk mengakses dasbor aplikasi Anda. URL yang Anda buat berlaku selama waktu tertentu.

> **Note**
>
> Jika Anda tidak mengakses URL yang dibuat dalam waktu tiga menit, URL tersebut tidak akan berlaku lagi.

Anda membuat URL dasbor Anda menggunakan CreateApplicationPresignedUrltindakan. Anda menentukan parameter berikut untuk tindakan:

- Nama aplikasi
- Waktu dalam detik ketika URL akan valid
- Anda menentukan `FLINK_DASHBOARD_URL` sebagai tipe URL.

Versi rilis

Topik ini berisi informasi tentang fitur yang didukung dan versi komponen yang direkomendasikan untuk setiap rilis Layanan Terkelola untuk Apache Flink.

Amazon Managed Service untuk rilis Apache Flink 1.15.2

Managed Service untuk Apache Flink mendukung fitur baru berikut di Apache 1.15.2

Fitur	Deskripsi	Referensi Apache FLIP
Wastafel Async	SebuahAWSkerangka kerja yang disumbangkan untuk membangun tujuan asinkron yang memungkin kan pengembang membuat kustomAWSkonektor dengan kurang dari setengah upaya sebelumnya. Untuk informasi lebih lanjut, lihatWastafel Dasar Asinkron Generik.	FLIP-171: Wastafel Asinkron.
Sink Kinesis Data Firehose	AWStelah menyumbangkan Amazon Kinesis Firehose Sink baru menggunakan kerangka kerja Async.	Wastafel Firehose Data Amazon Kinesis.
Berhenti dengan Savepoint	Berhenti dengan Savepoint memastikan operasi berhenti bersih, yang paling penting mendukung semantik yang tepat sekali untuk pelanggan yang bergantung pada mereka.	FLIP-34: Hentikan/Tangguhka n Pekerjaan dengan Savepoint .

Fitur	Deskripsi	Referensi Apache FLIP
Pemisahan Scala	Pengguna sekarang dapat memanfaatkan Java API dari versi Scala apa pun, termasuk Scala 3. Pelanggan perlu menggabungkan pustaka standar Scala pilihan mereka dalam aplikasi Scala mereka.	FLIP-28: Tujuan jangka panjang membuat flink-table bebas Scala-free.
Skala	Lihat decoupling Scala di atas	FLIP-28: Tujuan jangka panjang membuat flink-table bebas Scala-free.
Metrik Konektor Terpadu	Flink memilikimetrik standar yang ditentukan untuk pekerjaan, tugas, dan operator. Layanan Terkelola untuk Apache Flink akan terus mendukung sink dan metrik sumber dan di 1.15 memperkenalkan `numRestarts` secara paralel dengan `fullRestarts` untuk Metrik Ketersediaan.	FLIP-33: Standarisasi Metrik KonektordanFLIP-179: Mengekspos Metrik Operator Standar.
Checkpointing tugas selesai	Fitur ini diaktifkan secara default di Flink 1.15 dan memungkinkan untuk terus melakukan pos pemeriksaan bahkan jika bagian dari grafik pekerjaan telah selesai memproses semua data, yang mungkin terjadi jika berisi sumber (batch) terbatas.	FLIP-147: Mendukung Pos Pemeriksaan Setelah Tugas Selesai.

Perubahan Amazon Managed Service untuk Apache Flink dengan Apache Flink 1.15

Notebook studio

Layanan Terkelola untuk Apache Flink Studio sekarang mendukung Apache Flink 1.15. Managed Service untuk Apache Flink Studio menggunakan notebook Apache Zeppelin untuk memberikan pengalaman pengembangan antarmuka tunggal untuk mengembangkan, men-debug kode, dan menjalankan aplikasi pemrosesan aliran Apache Flink. Anda dapat mempelajari lebih lanjut tentang Managed Service untuk Apache Flink Studio dan cara memulainyaMenggunakan notebook Studio dengan Managed Service untuk Apache Flink.

Konektor EFO

Saat memutakhirkan ke Layanan Terkelola untuk Apache Flink versi 1.15, pastikan Anda menggunakan Konektor EFO terbaru, yaitu versi 1.15.3 atau yang lebih baru. Untuk informasi lebih lanjut tentang alasannya, lihatBATU API-29324.

Pemisahan Scala

Dimulai dengan Flink 1.15.2, Anda perlu menggabungkan pustaka standar Scala pilihan Anda dalam aplikasi Scala Anda.

Wastafel Firehose Data Kinesis

Saat memutakhirkan ke Layanan Terkelola untuk Apache Flink versi 1.15, pastikan Anda menggunakan yang terbaruWastafel Firehose Data Amazon Kinesis.

Konektor Kafka

Saat memutakhirkan ke Amazon Managed Service untuk Apache Flink untuk Apache Flink versi 1.15, pastikan Anda menggunakan API konektor Kafka terbaru. Apache Flink tidak digunakan lagiFlinkKafkaConsumerdanFlinkKafkaProducerAPI untuk sink Kafka ini tidak dapat berkomitmen ke Kafka untuk Flink 1.15. Pastikan Anda menggunakanKafkaSourcedanKafkaSink.

Komponen

Komponen	Versi
Java	11 (direkomendasikan)

Komponen	Versi
Skala	2.12
Layanan Terkelola untuk Apache Flink Flink Runtime (aws-kinesisanalytics-runtime)	1.2.0
AWSKonektor Kinesis (flink-connector-kinesis)	1.15.4
Apache Beam (hanya aplikasi Beam)	2.33.0, dengan Jackson versi 2.12.2

Menggunakan notebook Studio dengan Managed Service untuk Apache Flink

Notebook studio untuk Layanan Terkelola untuk Apache Flink memungkinkan Anda untuk secara interaktif menanyakan aliran data secara real time, dan dengan mudah membangun dan menjalankan aplikasi pemrosesan aliran menggunakan SQL, Python, dan Scala standar. Dengan beberapa klik di konsol Manajemen AWS, Anda dapat meluncurkan notebook nirserver untuk mengkueri aliran data dan mendapatkan hasil dalam hitungan detik.

Notebook adalah lingkungan pengembangan berbasis web. Dengan notebook, Anda mendapatkan pengalaman pengembangan interaktif sederhana yang dikombinasikan dengan kemampuan lanjutan yang disediakan oleh Apache Flink. Notebook studio menggunakan notebook yang didukung Apache Zeppelin, dan menggunakan Apache Flink sebagai mesin pemrosesan aliran. Notebook Studio menggabungkan teknologi ini dengan lancar untuk membuat analitik lanjutan pada aliran data yang dapat diakses oleh developer dari semua keahlian.

Apache Zeppelin memberi notebook Studio Anda dengan rangkaian alat analitik lengkap, termasuk yang berikut:

- Visualisasi Data
- Mengekspor data ke file
- Mengontrol format output untuk analisis yang lebih mudah

Untuk mulai menggunakan Managed Service untuk Apache Flink dan Apache Zeppelin, lihat. Membuat Tutorial notebook Studio Untuk informasi selengkapnya tentang Apache Zeppelin, lihat Dokumentasi Apache Zeppelin.

Dengan notebook, Anda memodelkan kueri menggunakan Apache Flink Table API & SQL di SQL, Python, atau Scala, atau API di Scala. DataStream Dengan beberapa klik, Anda kemudian dapat mempromosikan notebook Studio ke aplikasi pemrosesan aliran Apache Flink yang terus berjalan, non-interaktif, untuk beban kerja produksi Anda.

Topik ini berisi bagian-bagian berikut:

- Membuat notebook Studio
- Analisis interaktif data streaming

Membuat notebook Studio

Notebook Studio berisi kueri atau program yang ditulis dalam SQL, Python, atau Scala yang berjalan di data streaming dan mengembalikan hasil analitik. Anda membuat aplikasi menggunakan konsol atau CLI, dan menyediakan kueri untuk menganalisis data dari sumber data Anda.

Aplikasi Anda memiliki komponen berikut:

- Sumber data, seperti klaster Amazon MSK, Kinesis data stream, atau bucket Amazon S3.
- Basis data AWS Glue. Basis data ini berisi tabel, yang menyimpan sumber data dan tujuan serta skema titik akhir tujuan Anda. Untuk informasi selengkapnya, lihat Bekerja dengan AWS Glue.
- Kode aplikasi Anda. Kode Anda menerapkan kueri atau program analitik Anda.
- Pengaturan aplikasi dan properti runtime Anda. Untuk informasi tentang pengaturan aplikasi dan properti runtime, lihat topik berikut di Panduan Developer untuk Aplikasi Apache Flink:
  - Paralelisme dan Penskalaan Aplikasi: Anda menggunakan pengaturan Paralelisme aplikasi untuk mengontrol jumlah kueri yang dapat dijalankan aplikasi Anda secara bersamaan. Kueri Anda juga dapat mengambil keuntungan dari peningkatan paralelisme jika kueri tersebut memiliki beberapa jalur eksekusi, seperti dalam keadaan berikut:
    - Saat memproses beberapa serpihan Kinesis data stream
    - Ketika membuat partisi data menggunakan operator `KeyBy`.
    - Saat menggunakan beberapa operator jendela

    Untuk informasi selengkapnya tentang penskalaan aplikasi, lihat Penskalaan Aplikasi di Layanan Terkelola untuk Apache Flink untuk Apache Flink.

- Logging dan Monitoring: Untuk informasi tentang pencatatan dan pemantauan aplikasi, lihat Logging dan Monitoring di Amazon Managed Service untuk Apache Flink untuk Apache Flink.
- Aplikasi Anda menggunakan titik pemeriksaan dan titik simpan untuk toleransi kesalahan. Titik pemeriksaan dan titik simpan tidak diaktifkan secara default untuk notebook Studio.

Anda dapat membuat notebook Studio menggunakan AWS Management Console atau AWS CLI.

Saat membuat aplikasi dari konsol, Anda memiliki opsi berikut:

- Di konsol Amazon MSK, pilih klaster Anda, lalu pilih Process data in real time (Proses data secara langsung).
- Di konsol Kinesis Data Streams, pilih aliran data Anda, lalu di tab Application (Aplikasi), pilih Process data in real time (Proses data secara langsung).
- Di konsol Managed Service for Apache Flink pilih tab Studio, lalu pilih Buat notebook Studio.

Untuk tutorial, lihat Deteksi Peristiwa dengan Layanan Terkelola untuk Apache Flink.

Untuk contoh solusi notebook Studio yang lebih canggih, lihat Apache Flink di Amazon Managed Service for Apache Flink Studio.

Analisis interaktif data streaming

Anda menggunakan notebook nirserver yang didukung Apache Zeppelin untuk berinteraksi dengan data streaming Anda. Notebook Anda dapat memiliki beberapa catatan, dan setiap catatan dapat memiliki satu atau beberapa paragraf tempat Anda dapat menulis kode Anda.

Contoh kueri SQL berikut menunjukkan cara mengambil data dari sumber data:

```
%flink.ssql(type=update)
select * from stock;
```

Untuk contoh kueri SQL Flink Streaming selengkapnya, lihat Contoh dan tutorial berikut, dan Kueri di Dokumentasi Apache Flink.

Anda dapat menggunakan kueri SQL Flink di notebook Studio untuk mengkueri data streaming. Anda juga dapat menggunakan Python (API Tabel) dan Scala (API Tabel dan DataStream) untuk menulis program guna mengkueri data streaming Anda secara interaktif. Anda dapat melihat hasil kueri atau

program, memperbaruinya dalam hitungan detik, dan menjalankannya kembali untuk melihat hasil yang diperbarui.

Interpreter Flink

Anda menentukan bahasa Managed Service untuk Apache Flink yang digunakan untuk menjalankan aplikasi Anda dengan menggunakan interpreter. Anda dapat menggunakan interpreter berikut dengan Managed Service untuk Apache Flink:

Nama	Kelas	Deskripsi
%flink	FlinkInterpreter	Creates ExecutionEnvironment/StreamExecutionEnvironment/BatchTableEnvironment/StreamTableEnvironment and provides a Scala environment
%flink.pyflink	PyFlinkInterpreter	Provides a python environment
%flink.ipyflink	IPyFlinkInterpreter	Provides an ipython environment
%flink.ssql	FlinkStreamSqlInterpreter	Provides a stream sql environment
%flink.bsql	FlinkBatchSqlInterpreter	Provides a batch sql environment

Untuk informasi selengkapnya tentang interpreter Flink, lihat Interpreter Flink untuk Apache Zeppelin.

Jika Anda menggunakan `%flink.pyflink` atau `%flink.ipyflink` sebagai penerjemah Anda, Anda harus menggunakan `ZeppelinContext` untuk memvisualisasikan hasil dalam buku catatan.

Untuk contoh yang lebih PyFlink spesifik, lihat Kueri aliran data Anda secara interaktif menggunakan Layanan Terkelola untuk Apache Flink Studio dan Python.

Variabel lingkungan tabel Apache Flink

Apache Zeppelin menyediakan akses ke sumber daya lingkungan tabel menggunakan variabel lingkungan.

Anda mengakses sumber daya lingkungan tabel Scala dengan variabel berikut:

Variabel	Sumber daya
`senv`	`StreamExecutionEnvironment`
`stenv`	`StreamTableEnvironment untuk perencana kedip`

Anda mengakses sumber daya lingkungan tabel Python dengan variabel berikut:

Variabel	Sumber daya
`s_env`	`StreamExecutionEnvironment`
`st_env`	`StreamTableEnvironment untuk perencana kedip`

Untuk informasi selengkapnya tentang menggunakan lingkungan tabel, lihat Membuat TableEnvironment dokumentasi Apache Flink.

Menyebarkan sebagai aplikasi dengan status tahan lama

Anda dapat membangun kode Anda dan mengekspornya ke Amazon S3. Anda dapat mempromosikan kode yang Anda tulis dalam catatan Anda ke aplikasi pemrosesan streaming yang terus berjalan. Ada dua mode menjalankan aplikasi Apache Flink pada Managed Service untuk Apache Flink: Dengan notebook Studio, Anda memiliki kemampuan untuk mengembangkan kode Anda secara interaktif, melihat hasil kode Anda secara real time, dan memvisualisasikannya dalam catatan Anda. Setelah Anda menerapkan catatan untuk dijalankan dalam mode streaming, Managed Service for Apache Flink membuat aplikasi untuk Anda yang berjalan terus menerus, membaca data dari sumber Anda, menulis ke tujuan Anda, mempertahankan status aplikasi yang berjalan lama, dan skala otomatis secara otomatis berdasarkan throughput aliran sumber Anda.

> **Note**
>
> Bucket S3 tempat Anda mengekspor kode aplikasi harus berada dalam Wilayah yang sama dengan notebook Studio Anda.

Anda hanya dapat men-deploy catatan dari notebook Studio jika memenuhi kriteria berikut:

- Paragraf harus disusun secara berurutan. Saat Anda menerapkan aplikasi Anda, semua paragraf dalam catatan akan dieksekusi secara berurutan (left-to-right, top-to-bottom) seperti yang muncul di catatan Anda. Anda dapat memeriksa urutan ini dengan memilih Run All Paragraphs (Jalankan Semua Paragraf) di catatan Anda.
- Kode Anda adalah kombinasi Python dan SQL atau Scala dan SQL. Kami tidak mendukung Python dan Scala bersama saat ini untuk. deploy-as-application
- Catatan Anda sebaiknya hanya memiliki interpreter berikut: `%flink`, `%flink.ssql`, `%flink.pyflink`, `%flink.ipyflink`, `%md`.
- Penggunaan objek konteks Zeppelin z tidak didukung. Metode yang tidak mengembalikan apa pun tidak akan melakukan apa pun kecuali mencatat peringatan. Metode lain akan meningkatkan pengecualian Python atau gagal untuk mengompilasi di Scala.
- Catatan harus menghasilkan satu tugas Apache Flink.
- Catatan dengan formulir dinamis tidak didukung untuk men-deploy sebagai aplikasi.
- %md (Markdown) akan dilewati dalam deployment sebagai aplikasi, karena ini diprediksi berisi dokumentasi yang dapat dibaca manusia yang tidak cocok untuk dijalankan sebagai bagian dari aplikasi yang dihasilkan.
- Paragraf yang dinonaktifkan untuk berjalan dalam Zeppelin akan dilewati dalam deployment sebagai aplikasi. Bahkan jika paragraf yang dinonaktifkan menggunakan interpreter yang tidak kompatibel, misalnya, `%flink.ipyflink` dalam catatan dengan interpreter `%flink and %flink.ssql`, paragraf akan dilewati saat men-deploy catatan sebagai aplikasi, dan tidak akan mengakibatkan kesalahan.
- Harus ada setidaknya satu paragraf yang hadir dengan kode sumber (Flink SQL, PyFlink atau Flink Scala) yang diaktifkan untuk berjalan agar penerapan aplikasi berhasil.
- Mengatur paralelisme di direktif interpreter dalam paragraf (misalnya `%flink.ssql(parallelism=32)`) akan diabaikan dalam aplikasi yang di-deploy dari catatan. Sebagai gantinya, Anda dapat memperbarui aplikasi yang digunakan melaluiAWS Management Console, AWS Command Line Interface atau AWS API untuk mengubah pengaturan Paralelisme

dan/atau ParallelismPer KPU sesuai dengan tingkat paralelisme yang dibutuhkan aplikasi Anda, atau Anda dapat mengaktifkan penskalaan otomatis untuk aplikasi yang Anda gunakan.

- Jika Anda menerapkan sebagai aplikasi dengan status tahan lama VPC Anda harus memiliki akses internet. Jika VPC Anda tidak memiliki akses internet, lihat. Menyebarkan sebagai aplikasi dengan status tahan lama di VPC tanpa akses internet

Kriteria Scala/Python

- Dalam kode Scala atau Python Anda, gunakan Perencana Blink (`senv`, `stenv` untuk Scala; `s_env`, `st_env` untuk Python) dan bukan perencana "Flink" yang lebih lama (`stenv_2` untuk Scala, `st_env_2` untuk Python). Proyek Apache Flink merekomendasikan penggunaan perencana Blink untuk kasus penggunaan produksi, dan ini adalah perencana default di Zeppelin dan di Flink.
- Paragraf Python Anda tidak boleh menggunakan invokasi/tugas shell menggunakan `!` atau perintah magic IPython seperti `%timeit` atau `%conda` dalam catatan yang dimaksudkan untuk di-deploy sebagai aplikasi.
- Anda tidak dapat menggunakan kelas kasus Scala sebagai parameter fungsi yang diteruskan ke operator aliran data susunan yang lebih tinggi seperti `map` dan `filter`. Untuk informasi tentang kelas kasus Scala, lihat KELAS KASUS dalam dokumentasi Scala.

Kriteria SQL

- Pernyataan SELECT sederhana tidak diizinkan, karena tidak ada yang setara dengan bagian output paragraf tempat data dapat dikirim.
- Dalam setiap paragraf yang diberikan, pernyataan DDL (`USE`, `CREATE`, `ALTER`, `DROP`, `SET`, `RESET`) harus mendahului pernyataan DML (`INSERT`). Ini karena pernyataan DML dalam paragraf harus dikirimkan bersama-sama sebagai satu tugas Flink.
- Harus ada maksimal satu paragraf yang memiliki pernyataan DML di dalamnya. Ini karena, untuk deploy-as-application fitur tersebut, kami hanya mendukung pengiriman satu pekerjaan ke Flink.

Untuk informasi selengkapnya dan contoh, lihat Menerjemahkan, menyunting, dan menganalisis data streaming menggunakan fungsi SQL dengan Amazon Managed Service untuk Apache Flink, Amazon Translate, dan Amazon Comprehend.

Izin IAM untuk notebook Studio

Layanan Terkelola untuk Apache Flink membuat peran IAM untuk Anda saat Anda membuat buku catatan Studio melalui. AWS Management Console Ini juga berhubungan dengan peran kebijakan yang memungkinkan akses berikut:

Layanan	Akses
CloudWatch Log	Daftar
Amazon EC2	Daftar
AWS Glue	Baca, Tulis
Layanan Terkelola untuk Apache Flink	Baca
Layanan Terkelola untuk Apache Flink V2	Baca
Amazon S3	Baca, Tulis

Konektor dan dependensi

Konektor memungkinkan Anda membaca dan menulis data di berbagai teknologi. Layanan Terkelola untuk Apache Flink menggabungkan tiga konektor default dengan notebook Studio Anda. Anda juga dapat menggunakan konektor kustom. Untuk informasi selengkapnya tentang konektor, lihat Konektor Tabel & SQL di dokumentasi Apache Flink.

Konektor default

Jika Anda menggunakan AWS Management Console untuk membuat buku catatan Studio, Managed Service for Apache Flink menyertakan konektor kustom berikut secara default:`flink-sql-connector-flink`, `flink-connector-kafka_2.12` dan. `aws-msk-iam-auth` Untuk membuat notebook Studio melalui konsol tanpa konektor khusus ini, pilih opsi Buat dengan pengaturan khusus. Selanjutnya, ketika Anda sampai di halaman Konfigurasi, hapus kotak centang di sebelah dua konektor.

Jika Anda menggunakan CreateApplicationAPI untuk membuat notebook Studio, `flink-connector-kafka` konektor `flink-sql-connector-flink` dan konektor tidak disertakan

secara default. Untuk menambahkannya, tentukan konektor sebagai `MavenRefernce` di tipe data `CustomArtifactsConfiguration` seperti yang ditunjukkan dalam contoh berikut.

`aws-msk-iam-auth`Konektor adalah konektor yang akan digunakan dengan Amazon MSK yang menyertakan fitur untuk mengautentikasi secara otomatis dengan IAM.

> **Note**
>
> Versi konektor yang ditunjukkan dalam contoh berikut adalah satu-satunya versi yang kami dukung.

```
For the Kinesis connector:

"CustomArtifactsConfiguration": [{
"ArtifactType": "DEPENDENCY_JAR",
   "MavenReference": {
"GroupId": "org.apache.flink",

      "ArtifactId": "flink-sql-connector-kinesis",
      "Version": "1.15.4"

   }
}]

For authenticating with AWS MSK through AWS IAM:

"CustomArtifactsConfiguration": [{
"ArtifactType": "DEPENDENCY_JAR",
   "MavenReference": {
"GroupId": "software.amazon.msk",
      "ArtifactId": "aws-msk-iam-auth",
      "Version": "1.1.6"
   }
}]

For the Apache Kafka connector:

"CustomArtifactsConfiguration": [{
"ArtifactType": "DEPENDENCY_JAR",
   "MavenReference": {
"GroupId": "org.apache.flink",
```

```
      "ArtifactId": "flink-connector-kafka",
      "Version": "1.15.4"

   }
}]
```

Untuk menambahkan konektor ini ke notebook yang ada, gunakan operasi UpdateApplicationAPI dan tentukan sebagai `MavenReference` tipe `CustomArtifactsConfigurationUpdate` data.

> **Note**
>
> Anda dapat mengatur `failOnError` ke true untuk `flink-sql-connector-kinesis` konektor di API tabel.

Dependensi dan konektor kustom

Untuk menggunakan AWS Management Console untuk menambahkan dependensi atau konektor kustom ke notebook Studio, ikuti langkah-langkah berikut:

1. Unggah file konektor kustom Anda ke Amazon S3.
2. Di AWS Management Console, pilih opsi Custom create (Buat kustom) untuk membuat notebook Studio Anda.
3. Ikuti alur kerja pembuatan notebook Studio hingga Anda sampai di langkah Konfigurasi.
4. Di bagian Custom connectors (Konektor kustom), pilih Add custom connector (Tambahkan konektor kustom).
5. Tentukan lokasi Amazon S3 dari dependensi atau konektor kustom.
6. Pilih Save changes (Simpan perubahan).

Untuk menambahkan JAR dependensi atau konektor kustom saat Anda membuat notebook Studio baru menggunakan CreateApplicationAPI, tentukan lokasi Amazon S3 dari JAR dependensi atau konektor kustom dalam `CustomArtifactsConfiguration` tipe data. Untuk menambahkan dependensi atau konektor kustom ke notebook Studio yang ada, jalankan operasi UpdateApplicationAPI dan tentukan lokasi Amazon S3 dari JAR dependensi atau konektor khusus dalam tipe data. `CustomArtifactsConfigurationUpdate`

> **Note**
>
> Ketika Anda menyertakan dependensi atau konektor kustom, Anda juga harus menyertakan semua dependensi transitif yang tidak digabungkan di dalamnya.

Fungsi yang ditetapkan pengguna

Fungsi yang ditetapkan pengguna (UDFs) adalah titik ekstensi yang memungkinkan Anda memanggil logika yang sering digunakan atau logika kustom yang tidak dapat dinyatakan lain dalam kueri. Anda dapat menggunakan Python atau bahasa JVM seperti Java atau Scala untuk menerapkan UDFs Anda dalam paragraf di dalam notebook Studio Anda. Anda juga dapat menambahkan ke notebook Studio file JAR eksternal yang berisi UDFs yang diimplementasikan dalam bahasa JVM.

Saat menerapkan JAR yang mendaftarkan kelas abstrak yang subclass `UserDefinedFunction` (atau kelas abstrak Anda sendiri), gunakan cakupan yang disediakan di Apache Maven, deklarasi `compileOnly` dependensi di Gradle, cakupan yang disediakan di SBT, atau direktif yang setara dalam konfigurasi build proyek UDF Anda. Ini memungkinkan kode sumber UDF untuk dikompilasi terhadap API Flink, tetapi kelas Flink API tidak termasuk dalam artefak build. Lihat pom ini dari contoh toples UDF yang mematuhi prasyarat tersebut pada proyek Maven.

> **Note**
>
> Untuk contoh penyiapan, lihat Menerjemahkan, menyunting, dan menganalisis data streaming menggunakan fungsi SQL dengan Amazon Managed Service untuk Apache Flink, Amazon Translate, dan Amazon Comprehend di Blog Machine Learning. AWS

Untuk menggunakan konsol untuk menambahkan file JAR UDF ke notebook Studio Anda, ikuti langkah-langkah berikut:

1. Upload file JAR UDF Anda ke Amazon S3.
2. Di AWS Management Console, pilih opsi Custom create (Buat kustom) untuk membuat notebook Studio Anda.
3. Ikuti alur kerja pembuatan notebook Studio hingga Anda sampai di langkah Konfigurasi.
4. Di bagian User-defined functions (Fungsi yang ditetapkan pengguna), pilih Add user-defined function (Tambahkan fungsi yang ditetapkan pengguna).

5. Tentukan lokasi Amazon S3 dari file JAR atau file ZIP yang memiliki implementasi UDF Anda.
6. Pilih Simpan perubahan.

Untuk menambahkan UDF JAR saat membuat notebook Studio baru menggunakan CreateApplicationAPI, tentukan lokasi JAR dalam tipe `CustomArtifactConfiguration` data. Untuk menambahkan UDF JAR ke notebook Studio yang ada, jalankan operasi UpdateApplicationAPI dan tentukan lokasi JAR dalam tipe `CustomArtifactsConfigurationUpdate` data. Atau, Anda dapat menggunakan AWS Management Console untuk menambahkan file JAR UDF ke notebook Studio Anda.

Pertimbangan dengan fungsi yang ditentukan pengguna

- Managed Service untuk Apache Flink Studio menggunakan terminologi Apache Zeppelin dimana notebook adalah contoh Zeppelin yang dapat berisi beberapa catatan. Setiap catatan kemudian dapat berisi beberapa paragraf. Dengan Managed Service for Apache Flink Studio, proses interpreter dibagikan di semua catatan di buku catatan. Jadi jika Anda melakukan registrasi fungsi eksplisit menggunakan createTemporarySystemFungsi dalam satu catatan, hal yang sama dapat direferensikan apa adanya di catatan lain dari buku catatan yang sama.

  Namun, Deploy sebagai operasi aplikasi berfungsi pada catatan individual dan tidak semua catatan di buku catatan. Saat Anda melakukan penerapan sebagai aplikasi, hanya konten catatan aktif yang digunakan untuk menghasilkan aplikasi. Registrasi fungsi eksplisit apa pun yang dilakukan di notebook lain bukan merupakan bagian dari dependensi aplikasi yang dihasilkan. Selain itu, selama Deploy sebagai opsi aplikasi pendaftaran fungsi implisit terjadi dengan mengubah nama kelas utama JAR ke string huruf kecil.

  Misalnya, jika `TextAnalyticsUDF` adalah kelas utama untuk UDF JAR, maka registrasi implisit akan menghasilkan nama fungsi. `textanalyticsudf` Jadi jika pendaftaran fungsi eksplisit di catatan 1 Studio terjadi seperti berikut ini, maka semua catatan lain di buku catatan itu (katakanlah catatan 2) dapat merujuk fungsi dengan nama `myNewFuncNameForClass` karena penerjemah bersama:

  ```
  stenv.createTemporarySystemFunction("myNewFuncNameForClass", new
  TextAnalyticsUDF())
  ```

  Namun selama penerapan sebagai operasi aplikasi pada catatan 2, pendaftaran eksplisit ini tidak akan disertakan dalam dependensi dan karenanya aplikasi yang diterapkan tidak akan berfungsi

seperti yang diharapkan. Karena pendaftaran implisit, secara default semua referensi ke fungsi ini diharapkan bersama `textanalyticsudf` dan tidak`myNewFuncNameForClass`.

Jika ada kebutuhan untuk pendaftaran nama fungsi kustom maka catatan 2 itu sendiri diharapkan berisi paragraf lain untuk melakukan pendaftaran eksplisit lainnya sebagai berikut:

```
%flink(parallelism=1)
import com.amazonaws.kinesis.udf.textanalytics.TextAnalyticsUDF
# re-register the JAR for UDF with custom name
stenv.createTemporarySystemFunction("myNewFuncNameForClass", new TextAnalyticsUDF())
```

```
%flink. ssql(type=update, parallelism=1)
INSERT INTO
    table2
SELECT
    myNewFuncNameForClass(column_name)
FROM
    table1
;
```

- Jika UDF JAR Anda menyertakan Flink SDK, maka konfigurasikan proyek Java Anda sehingga kode sumber UDF dapat dikompilasi terhadap SDK Flink, tetapi kelas Flink SDK tidak termasuk dalam artefak build, misalnya JAR.

  Anda dapat menggunakan `provided` cakupan di Apache Maven, deklarasi `compileOnly` dependensi di Gradle, `provided` cakupan di SBT, atau direktif yang setara dalam konfigurasi build proyek UDF mereka. Anda dapat merujuk ke pom ini dari contoh toples UDF, yang mematuhi prasyarat seperti itu pada proyek maven. Untuk step-by-step tutorial selengkapnya, lihat Terjemahkan, edit, dan analisis data streaming menggunakan fungsi SQL dengan Amazon Managed Service untuk Apache Flink, Amazon Translate, dan Amazon Comprehend.

Mengaktifkan Checkpointing

Anda mengaktifkan checkpointing menggunakan pengaturan lingkungan. Untuk informasi tentang checkpointing, lihat Toleransi Kesalahan di Managed Service for Apache Flink Developer Guide.

Mengatur interval checkpointing

Contoh kode Scala berikut mengatur interval titik pemeriksaan aplikasi Anda ke satu menit:

```
// start a checkpoint every 1 minute
stenv.enableCheckpointing(60000)
```

Contoh kode Phyton berikut mengatur interval titik pemeriksaan aplikasi Anda ke satu menit:

```
st_env.get_config().get_configuration().set_string(
    "execution.checkpointing.interval", "1min"
)
```

Mengatur tipe checkpointing

Contoh kode Scala berikut mengatur mode titik pemeriksaan aplikasi Anda ke `EXACTLY_ONCE` (default):

```
// set mode to exactly-once (this is the default)
stenv.getCheckpointConfig.setCheckpointingMode(CheckpointingMode.EXACTLY_ONCE)
```

Contoh kode Phyton berikut mengatur mode titik pemeriksaan aplikasi Anda ke `EXACTLY_ONCE` (default):

```
st_env.get_config().get_configuration().set_string(
    "execution.checkpointing.mode", "EXACTLY_ONCE"
)
```

Bekerja dengan AWS Glue

Notebook Studio Anda menyimpan dan mendapatkan informasi tentang sumber data dan sink dari AWS Glue. Saat Anda membuat notebook Studio, Anda menentukan basis data AWS Glue yang berisi informasi koneksi Anda. Saat Anda mengakses sumber data dan sink, Anda menentukan tabel AWS Glue yang terdapat dalam basis data. Tabel AWS Glue menyediakan akses ke koneksi AWS Glue yang menentukan lokasi, skema, dan parameter sumber data serta tujuan Anda.

Notebook Studio menggunakan properti tabel untuk menyimpan data khusus aplikasi. Untuk informasi selengkapnya, lihat Properti tabel.

Untuk contoh cara menyiapkan koneksi AWS Glue, basis data, dan tabel untuk digunakan dengan notebook Studio, lihat Buat Basis Data AWS Glue di tutorial Membuat Tutorial notebook Studio.

Properti tabel

Selain bidang data, tabel AWS Glue Anda memberikan informasi lain ke notebook Studio Anda menggunakan properti tabel. Managed Service untuk Apache Flink menggunakan properti AWS Glue tabel berikut:

- Menggunakan nilai waktu Apache Flink: Properti ini menentukan bagaimana Managed Service untuk Apache Flink memancarkan nilai waktu pemrosesan data internal Apache Flink.
- Menggunakan Konektor Flink dan properti format: Properti ini memberikan informasi tentang aliran data Anda.

Untuk menambahkan properti ke tabel AWS Glue, lakukan hal berikut:

1. Masuk ke AWS Management Console, lalu buka konsol AWS Glue di https://console.aws.amazon.com/glue/.
2. Dari daftar tabel, pilih tabel yang digunakan aplikasi Anda untuk menyimpan informasi koneksi datanya. Pilih Action (Tindakan), Edit table details (Edit detail tabel).
3. Di bawah Table Properties (Properti Tabel), masukkan **`managed-flink.proctime`** untuk key (kunci) dan **`user_action_time`** untuk Value (Nilai).

Menggunakan nilai waktu Apache Flink

Apache Flink memberikan nilai waktu yang menjelaskan kapan peristiwa pemrosesan aliran terjadi, seperti Processing Time (Waktu Pemrosesan) dan Event Time (Waktu Peristiwa). Untuk menyertakan nilai-nilai ini dalam keluaran aplikasi Anda, Anda menentukan properti pada AWS Glue tabel yang memberi tahu runtime Managed Service for Apache Flink untuk memancarkan nilai-nilai ini ke dalam bidang yang ditentukan.

Kunci dan nilai yang Anda gunakan dalam properti tabel Anda adalah sebagai berikut:

Tipe Stempel Waktu	Kunci	Nilai
Waktu Pemrosesan	managed-flink.proctime	The column name that AWS Glue will use to expose the value. This column name does not correspond to an existing table column.

Tipe Stempel Waktu	Kunci	Nilai
Waktu Peristiwa	managed-flink.rowtime	The column name that AWS Glue will use to expose the value. This column name corresponds to an existing table column.
	dikelola flink.watermark. *column_name* .milidetik	The watermark interval in milliseconds

Menggunakan Konektor Flink dan properti format

Anda memberikan informasi tentang sumber data Anda ke konektor Flink aplikasi Anda menggunakan properti tabel AWS Glue. Beberapa contoh properti yang Managed Service untuk Apache Flink gunakan untuk konektor adalah sebagai berikut:

Tipe Konektor	Kunci	Nilai
Kafka	`format`	The format used to deserialize and serialize Kafka messages, e.g. `json` or `csv`.
	`scan.startup.mode`	The startup mode for the Kafka consumer, e.g. `earliest-offset` or `timestamp` .
Kinesis	`format`	The format used to deserialize and serialize Kinesis data stream records, e.g. `json` or `csv`.
	`aws.region`	The AWS region where the stream is defined.

Tipe Konektor	Kunci	Nilai
S3 (Filesystem)	format	The format used to deserialize and serialize files, e.g. `json` or `csv`.
	`path`	The Amazon S3 path, e.g. `s3://mybucket/ .`

Untuk informasi selengkapnya tentang konektor lainnya selain Kinesis dan Apache Kafka, lihat dokumentasi konektor Anda.

Contoh dan tutorial

Topik

- Tutorial: Membuat notebook Studio di Managed Service untuk Apache Flink
- Tutorial: Men-deploy sebagai aplikasi dengan status tahan lama
- Contoh-contoh

Tutorial: Membuat notebook Studio di Managed Service untuk Apache Flink

Tutorial berikut menunjukkan cara membuat notebook Studio yang membaca data dari Kinesis Data Stream atau klaster Amazon MSK.

Tutorial ini berisi bagian-bagian berikut:

- Pengaturan
- Buat Basis Data AWS Glue
- Langkah Selanjutnya
- Membuat notebook Studio dengan Kinesis Data Streams
- Membuat notebook Studio dengan Amazon MSK
- Membersihkan aplikasi Anda dan sumber daya dependen

Pengaturan

Pastikan AWS CLI Anda adalah versi 2 atau yang lebih baru. Untuk menginstal AWS CLI terbaru, lihat Menginstal, memperbarui, dan menghapus instalan AWS CLI versi 2.

Buat Basis Data AWS Glue

Notebook Studio Anda menggunakan basis data AWS Glue untuk metadata tentang sumber data Amazon MSK Anda.

Buat Basis Data AWS Glue

1. Buka konsol AWS Glue di https://console.aws.amazon.com/glue/.
2. Pilih Add database (Tambahkan basis data). Di jendela Add database (Tambahkan basis data), masukkan **`default`** untuk Database name (Nama basis data). Pilih Create (Buat).

Langkah Selanjutnya

Dengan tutorial ini, Anda dapat membuat notebook Studio yang menggunakan Kinesis Data Streams atau Amazon MSK:

- Kinesis Data Streams: Dengan Kinesis Data Streams, Anda dengan cepat membuat aplikasi yang menggunakan aliran data Kinesis sebagai sumber. Anda hanya perlu membuat Kinesis data stream sebagai sumber daya dependen.
- Amazon MSK: Dengan Amazon MSK, Anda membuat aplikasi yang menggunakan klaster Amazon MSK sebagai sumber. Anda perlu membuat Amazon VPC, instans klien Amazon EC2, dan klaster Amazon MSK sebagai sumber daya dependen.

Membuat notebook Studio dengan Kinesis Data Streams

Tutorial ini menjelaskan cara membuat notebook Studio yang menggunakan Kinesis data stream sebagai sumber.

Tutorial ini berisi bagian-bagian berikut:

- Pengaturan
- Buat tabel AWS Glue
- Buat notebook Studio dengan Kinesis Data Streams

- Kirim data ke Kinesis data stream Anda
- Uji notebook Studio Anda

Pengaturan

Sebelum Anda membuat notebook Studio, buat Kinesis data stream (`ExampleInputStream`). Aplikasi Anda menggunakan aliran ini untuk sumber aplikasi.

Anda dapat membuat aliran ini menggunakan konsol Amazon Kinesis atau perintah AWS CLI. Untuk instruksi konsol, lihat Membuat dan Memperbarui Aliran Data di Panduan Developer Amazon Kinesis Data Streams. Beri nama aliran **`ExampleInputStream`** dan atur Number of open shards (Jumlah serpihan terbuka) ke **`1`**.

Untuk membuat aliran (`ExampleInputStream`) menggunakan AWS CLI, gunakan perintah `create-stream` AWS CLI Amazon Kinesis berikut.

```
$ aws kinesis create-stream \
--stream-name ExampleInputStream \
--shard-count 1 \
--region us-east-1 \
--profile adminuser
```

Buat tabel AWS Glue

Notebook Studio Anda menggunakan basis data AWS Glue untuk metadata tentang sumber data Kinesis Data Streams Anda.

> **Note**
>
> Anda dapat membuat database secara manual terlebih dahulu atau Anda dapat membiarkan Managed Service for Apache Flink membuatnya untuk Anda saat Anda membuat buku catatan. Demikian pula, Anda dapat membuat tabel secara manual seperti yang dijelaskan di bagian ini, atau Anda dapat menggunakan kode konektor buat tabel untuk Layanan Terkelola untuk Apache Flink di buku catatan Anda dalam Apache Zeppelin untuk membuat tabel Anda melalui pernyataan DDL. Anda selanjutnya dapat masuk AWS Glue untuk memastikan tabel dibuat dengan benar.

Buat Tabel

1. Masuk ke AWS Management Console, lalu buka konsol AWS Glue di https://console.aws.amazon.com/glue/.
2. Jika Anda belum memiliki basis data AWS Glue, pilih Databases (Basis Data) dari bilah navigasi sebelah kiri. Pilih Add database (Tambahkan basis data). Di jendela Add database (Tambahkan basis data), masukkan **`default`** untuk Database name (Nama basis data). Pilih Create (Buat).
3. Di bilah navigasi sebelah kiri, pilih Tables (Tabel). Di halaman Tabel, pilih Add tables (Tambahkan tabel), Add table manually (Tambahkan tabel secara manual).
4. Di halaman Set up your table's properties (Siapkan properti tabel Anda), masukkan **`stock`** untuk Table name (Nama tabel). Pastikan Anda memilih basis data yang Anda buat sebelumnya. Pilih Berikutnya.
5. Di halaman Tambahkan penyimpanan data, pilih Kinesis. Untuk Stream name (Nama aliran), masukkan **`ExampleInputStream`**. untuk Kinesis source URL (URL sumber Kinesis), pilih masukkan **`https://kinesis.us-east-1.amazonaws.com`**. Jika Anda menyalin dan menempel URL sumber Kinesis, pastikan untuk menghapus spasi awal atau akhir. Pilih Berikutnya.
6. Di halaman Klasifikasi, pilih JSON. Pilih Berikutnya.
7. Di halaman Tentukan skema, pilih Add Column (Tambahkan kolom) untuk menambahkan kolom. Tambahkan kolom dengan properti berikut:

Nama kolom	Jenis data
`ticker`	**`string`**
`price`	**`double`**

   Pilih Berikutnya.
8. Di halaman berikutnya, verifikasi pengaturan Anda, dan pilih Finish (Selesai).
9. Pilih tabel yang baru dibuat dari daftar tabel.
10. Pilih Edit table (Edit tabel) dan tambahkan properti dengan kunci `managed-flink.proctime` dan nilai `proctime`.
11. Pilih Apply (Terapkan).

Buat notebook Studio dengan Kinesis Data Streams

Sekarang Anda sudah membuat sumber daya yang digunakan aplikasi Anda, Anda membuat notebook Studio Anda.

Untuk membuat aplikasi Anda, Anda dapat menggunakan AWS Management Console atau AWS CLI.

- Buat notebook Studio menggunakan AWS Management Console
- Buat notebook Studio menggunakan AWS CLI

Buat notebook Studio menggunakan AWS Management Console

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/managed-flink/home?region=us-east-1#/applications/dashboard.
2. Di halaman Managed Service for Apache Flink Apache Applications, pilih tab Studio. Pilih Create Studio notebook (Buat notebook Studio).

   > **Note**
   >
   > Anda juga dapat membuat notebook Studio dari konsol Amazon MSK atau Kinesis Data Streams dengan memilih klaster Amazon MSK input atau Kinesis data stream, dan memilih Process data in real time (Proses data secara langsung).

3. Di halaman Buat notebook Studio, berikan informasi berikut:

   - Masukkan **`MyNotebook`** untuk nama notebook.
   - Pilih default untuk Basis data AWS Glue.

   Pilih Create Studio notebook (Buat notebook Studio).
4. Di MyNotebookhalaman, pilih Jalankan. Tunggu Status hingga menampilkan Running (Berjalan). Biaya berlaku saat notebook berjalan.

Buat notebook Studio menggunakan AWS CLI

Untuk membuat notebook Studio Anda menggunakan AWS CLI, lakukan hal berikut:

1. Verifikasi ID akun Anda. Anda memerlukan nilai ini untuk membuat aplikasi Anda.

2. Buat peran `arn:aws:iam::AccountID:role/ZeppelinRole` dan tambahkan izin berikut ke peran yang dibuat secara otomatis oleh konsol.

```
"kinesis:GetShardIterator",

"kinesis:GetRecords",

"kinesis:ListShards"
```

3. Buat file bernama `create.json` dengan konten berikut. Ganti nilai placeholder dengan informasi Anda.

```
{
    "ApplicationName": "MyNotebook",
    "RuntimeEnvironment": "ZEPPELIN-FLINK-3_0",
    "ApplicationMode": "INTERACTIVE",
    "ServiceExecutionRole": "arn:aws:iam::AccountID:role/ZeppelinRole",
    "ApplicationConfiguration": {
        "ApplicationSnapshotConfiguration": {
            "SnapshotsEnabled": false
        },
        "ZeppelinApplicationConfiguration": {
            "CatalogConfiguration": {
                "GlueDataCatalogConfiguration": {
                    "DatabaseARN": "arn:aws:glue:us-east-1:AccountID:database/
default"
                }
            }
        }
    }
}
```

4. Jalankan perintah berikut untuk membuat aplikasi Anda.

```
aws kinesisanalyticsv2 create-application --cli-input-json file://create.json
```

5. Setelah perintah selesai, Anda melihat output yang menampilkan detail untuk notebook Studio baru Anda. Berikut adalah contoh output.

```
{
    "ApplicationDetail": {
        "ApplicationARN": "arn:aws:kinesisanalyticsus-
east-1:012345678901:application/MyNotebook",
```

```
        "ApplicationName": "MyNotebook",
        "RuntimeEnvironment": "ZEPPELIN-FLINK-3_0",
        "ApplicationMode": "INTERACTIVE",
        "ServiceExecutionRole": "arn:aws:iam::012345678901:role/ZeppelinRole",
...
```

6. Jalankan perintah berikut untuk memulai aplikasi Anda. Ganti nilai sampel dengan ID akun Anda.

```
aws kinesisanalyticsv2 start-application --application-arn
 arn:aws:kinesisanalyticsus-east-1:012345678901:application/MyNotebook\
```

Kirim data ke Kinesis data stream Anda

Untuk mengirim data uji ke Kinesis data stream, lakukan hal berikut:

1. Buka Kinesis Data Generator.
2. Pilih Buat Pengguna Cognito dengan. CloudFormation
3. Konsol AWS CloudFormation terbuka dengan templat Kinesis Data Generator. Pilih Berikutnya.
4. Di halaman Tentukan detail tumpukan, masukkan nama pengguna dan kata sandi pengguna Cognito Anda. Pilih Berikutnya.
5. Di halaman Konfigurasikan opsi tumpukan, pilih Next (Berikutnya).
6. Di halaman Review Kinesis-Data-Generator-Cognito-User, pilih yang saya akui yang mungkin membuat sumber daya IAM. AWS CloudFormation kotak centang. Pilih Create Stack (Buat Tumpukan).
7. Tunggu tumpukan AWS CloudFormation selesai dibuat. Setelah tumpukan selesai, buka tumpukan Kinesis-Data-Generator-Cognito-User di konsol AWS CloudFormation, dan pilih tab Outputs (Output). Buka URL yang terdaftar untuk nilai KinesisDataGeneratorUrloutput.
8. Di halaman Amazon Kinesis Data Generator, masuk dengan kredensial yang Anda buat di langkah 4.
9. Di halaman berikutnya, berikan nilai berikut:

Region	**us-east-1**
Aliran/Kinesis Data Aliran Firehose	**ExampleInputStream**
Catatan per detik	**1**

Untuk Record Template (Templat Catatan), tempel kode berikut:

```
{
    "ticker": "{{random.arrayElement(
        ["AMZN","MSFT","GOOG"]
    )}}",
    "price": {{random.number(
        {
            "min":10,
            "max":150
        }
    )}}
}
```

10. Pilih Send data (Kirim data).
11. Generator akan mengirimkan data ke Kinesis data stream Anda.

    Biarkan generator berjalan sewaktu Anda menyelesaikan bagian berikutnya.

Uji notebook Studio Anda

Di bagian ini, Anda menggunakan notebook Studio untuk mengkueri data dari Kinesis data stream Anda.

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/managed-flink/home?region=us-east-1#/applications/dashboard.
2. Pada halaman Managed Service for Apache Flink Apache Applications, pilih tab notebook Studio. Pilih MyNotebook.
3. Di MyNotebookhalaman, pilih Buka di Apache Zeppelin.

   Antarmuka Apache Zeppelin terbuka di tab baru.
4. Di halaman Selamat Datang di Zeppelin!, pilih Zeppelin Note (Catatan Zeppelin).
5. Di halaman Zeppelin Note (Catatan Zeppelin), masukkan kueri berikut ke dalam catatan baru:

   ```
   %flink.ssql(type=update)
   select * from stock
   ```

   Pilih ikon jalankan.

Setelah beberapa saat, catatan menampilkan data dari Kinesis data stream.

Untuk membuka Dasbor Apache Flink untuk aplikasi Anda agar dapat melihat aspek operasional, pilih FLINK JOB (TUGAS FLINK). Untuk informasi selengkapnya tentang Dasbor Flink, lihat Dasbor Apache Flink di Managed Service for Apache Flink Developer Guide.

Untuk contoh kueri SQL Flink Streaming selengkapnya, lihat Kueri di Dokumentasi Apache Flink.

Membuat notebook Studio dengan Amazon MSK

Tutorial ini menjelaskan cara membuat notebook Studio yang menggunakan klaster Amazon MSK sebagai sumber.

Tutorial ini berisi bagian-bagian berikut:

- Penyiapan
- Tambahkan Gateway NAT ke VPC Anda
- Buat Koneksi AWS Glue dan Tabel
- Buat notebook Studio dengan Amazon MSK
- Kirim data ke klaster Amazon MSK Anda
- Uji notebook Studio Anda

Penyiapan

Untuk tutorial ini, Anda memerlukan klaster Amazon MSK yang memungkinkan akses plaintext. Jika Anda belum menyiapkan klaster Amazon MSK, ikuti tutorial Memulai Menggunakan Amazon MSK untuk membuat Amazon VPC, klaster Amazon MSK, topik, dan instans klien Amazon EC2.

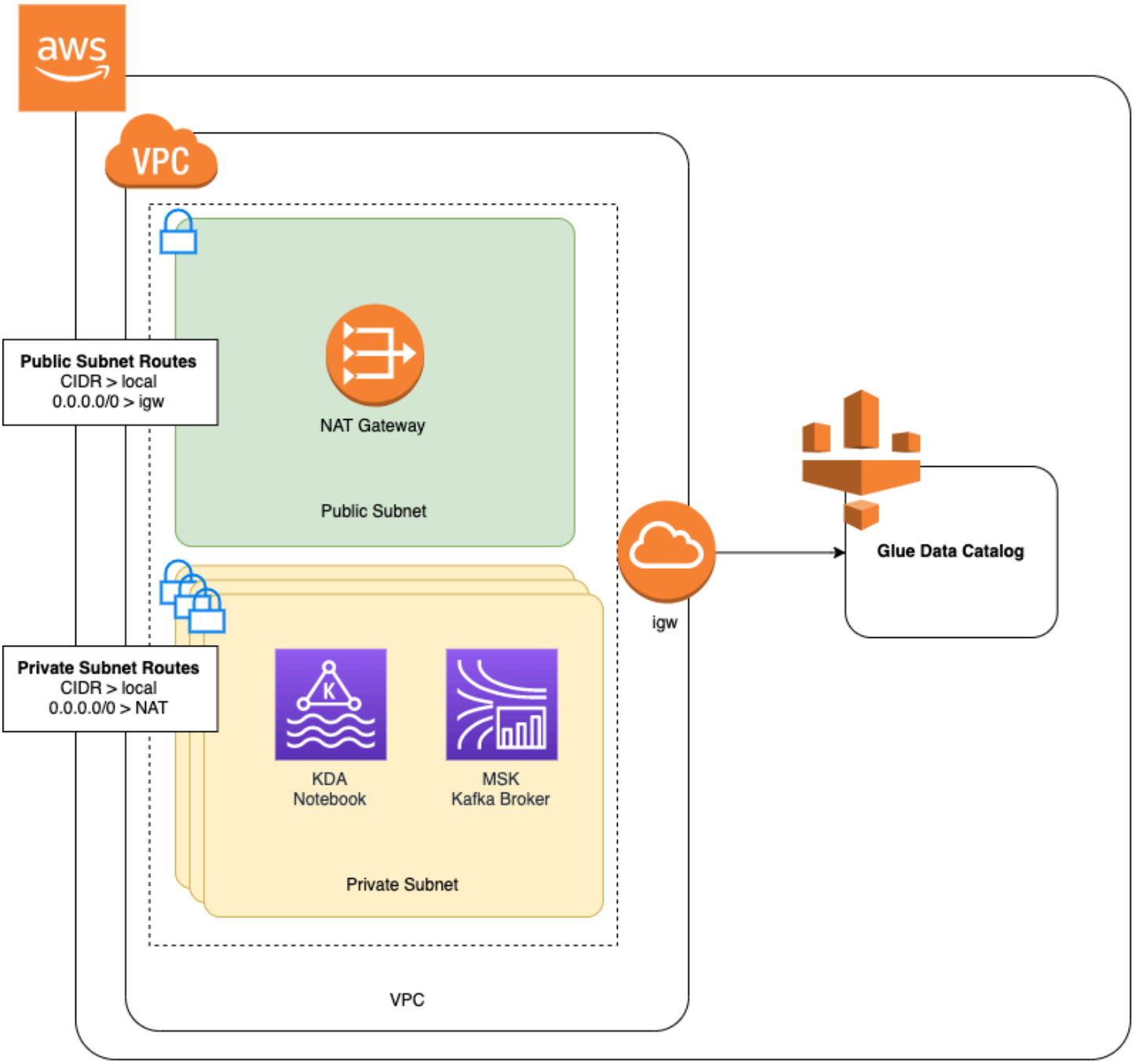
Saat mengikuti tutorial, lakukan hal berikut:

- Di Langkah 3: Buat Klaster Amazon MSK, di langkah 4, ubah nilai `ClientBroker` dari `TLS` ke **`PLAINTEXT`**.

Tambahkan Gateway NAT ke VPC Anda

Jika Anda membuat klaster Amazon MSK dengan mengikuti tutorial Memulai Menggunakan Amazon MSK, atau jika Amazon VPC Anda yang sudah ada tidak memiliki gateway NAT untuk subnet

privatnya, Anda harus menambahkan Gateway NAT ke Amazon VPC Anda. Diagram berikut menunjukkan arsitektur.



Untuk membuat Gateway NAT untuk Amazon VPC Anda, lakukan hal berikut:

1. Buka konsol Amazon VPC di https://console.aws.amazon.com/vpc/.
2. Pilih NAT Gateways (Gateway NAT) dari bilah navigasi sebelah kiri.
3. Di halaman Gateway NAT, pilih Create NAT Gateway (Buat Gateway NAT).

4. Di halaman Buat Gateway NAT, berikan nilai berikut:

Nama - opsional	**`ZeppelinGateway`**
Subnet	AWSKafkaTutorialSubnet1
ID alokasi IP elastis	Choose an available Elastic IP. If there are no Elastic IPs available, choose Alokasikan IP elastis, and then choose the Elasic IP that the console creates.

   Pilih Create NAT Gateway (Buat Gateway NAT).
5. Di bilah navigasi sebelah kiri, pilih Route Tables (Tabel Rute).
6. Pilih Create Route Table (Buat Tabel Rute).
7. Di halaman Create route table (Buat tabel rute), berikan informasi berikut:
   - Name tag (Tanda nama): **`ZeppelinRouteTable`**
   - VPC: Pilih VPC Anda (misalnya VPC). AWS KafkaTutorial

   Pilih Create (Buat).
8. Dalam daftar tabel rute, pilih ZeppelinRouteTable. Pilih tab Routes (Rute), dan pilih Edit routes (Edit rute).
9. Di halaman Edit Rute, pilih Add route (Tambahkan rute).
10. Di Untuk Tujuan, masukkan **`0.0.0.0/0`**. Untuk Target, pilih NAT Gateway, ZeppelinGateway. Pilih Save Routes (Simpan Rute). Pilih Close (Tutup).
11. Pada halaman Tabel Rute, dengan ZeppelinRouteTabledipilih, pilih tab Asosiasi Subnet. Pilih Edit subnet associations (Edit asosiasi subnet).
12. Di halaman Edit asosiasi subnet, pilih AWSKafkaTutorialSubnet2 dan AWSKafkaTutorialSubnet3. Pilih Save (Simpan).

Buat Koneksi AWS Glue dan Tabel

Notebook Studio Anda menggunakan basis data AWS Glue untuk metadata tentang sumber data Amazon MSK Anda. Di bagian ini, Anda membuat koneksi AWS Glue yang menjelaskan cara

mengakses klaster Amazon MSK Anda, dan tabel AWS Glue yang menjelaskan cara menyajikan data dalam sumber data Anda ke klien seperti notebook Studio Anda.

Buat Koneksi

1. Masuk ke AWS Management Console, lalu buka konsol AWS Glue di https://console.aws.amazon.com/glue/.
2. Jika Anda belum memiliki basis data AWS Glue, pilih Databases (Basis Data) dari bilah navigasi sebelah kiri. Pilih Add database (Tambahkan basis data). Di jendela Add database (Tambahkan basis data), masukkan **`default`** untuk Database name (Nama basis data). Pilih Create (Buat).
3. Pilih Connections (Koneksi) dari bilah navigasi sebelah kiri. Pilih Add Connection (Tambahkan Koneksi).
4. Di jendela Tambahkan Koneksi, berikan nilai berikut:

   - Untuk Connection name (Nama koneksi), masukkan **`ZeppelinConnection`**.
   - Untuk Connection type (Tipe koneksi), pilih Kafka.
   - Untuk Kafka bootstrap server URLs (URL server bootstrap Kafka, berikan string broker bootstrap untuk klaster Anda. Anda bisa mendapatkan broker bootstrap dari konsol MSK, atau dengan memasukkan perintah CLI berikut:

     ```
     aws kafka get-bootstrap-brokers --region us-east-1 --cluster-arn ClusterArn
     ```

   - Hapus centang di kotak centang Require SSL connection (Perlu koneksi SSL).

   Pilih Selanjutnya.
5. Di halaman VPC, berikan nilai berikut:

   - Untuk VPC, pilih nama VPC Anda (misalnya VPC.) AWS KafkaTutorial
   - Untuk Subnet, pilih AWSKafkaTutorialSubnet2.
   - Untuk Security groups (Grup keamanan), pilih semua grup yang tersedia.

   Pilih Selanjutnya.
6. Di halaman Properti koneksi / Akses koneksi, pilih Finish (Selesai).

Buat Tabel

> **Note**
>
> Anda dapat membuat tabel secara manual seperti yang dijelaskan dalam langkah-langkah berikut, atau Anda dapat menggunakan kode konektor buat tabel untuk Layanan Terkelola untuk Apache Flink di buku catatan Anda dalam Apache Zeppelin untuk membuat tabel Anda melalui pernyataan DDL. Anda selanjutnya dapat masuk AWS Glue untuk memastikan tabel dibuat dengan benar.

1. Di bilah navigasi sebelah kiri, pilih Tables (Tabel). Di halaman Tabel, pilih Add tables (Tambahkan tabel), Add table manually (Tambahkan tabel secara manual).
2. Di halaman Set up your table's properties (Siapkan properti tabel Anda), masukkan **`stock`** untuk Table name (Nama tabel). Pastikan Anda memilih basis data yang Anda buat sebelumnya. Pilih Selanjutnya.
3. Di halaman Tambahkan penyimpanan data, pilih Kafka. Untuk nama Topik, masukkan nama topik Anda (mis. AWSKafkaTutorialTopic). Untuk Koneksi, pilih ZeppelinConnection.
4. Di halaman Klasifikasi, pilih JSON. Pilih Selanjutnya.
5. Di halaman Tentukan skema, pilih Add Column (Tambahkan kolom) untuk menambahkan kolom. Tambahkan kolom dengan properti berikut:

Nama kolom	Tipe data
`ticker`	**`string`**
`price`	**`double`**

   Pilih Selanjutnya.
6. Di halaman berikutnya, verifikasi pengaturan Anda, dan pilih Finish (Selesai).
7. Pilih tabel yang baru dibuat dari daftar tabel.
8. Pilih Edit table (Edit tabel) dan tambahkan properti dengan kunci `managed-flink.proctime` dan nilai `proctime`.
9. Pilih Apply (Terapkan).

Buat notebook Studio dengan Amazon MSK

Sekarang Anda sudah membuat sumber daya yang digunakan aplikasi Anda, Anda membuat notebook Studio Anda.

Anda dapat membuat aplikasi menggunakan AWS Management Console atau AWS CLI.

- Buat notebook Studio menggunakan AWS Management Console
- Buat notebook Studio menggunakan AWS CLI

> **Note**
>
> Anda juga dapat membuat notebook Studio dari konsol Amazon MSK dengan memilih klaster yang sudah ada, lalu memilih Process data in real time (Proses data secara langsung).

Buat notebook Studio menggunakan AWS Management Console

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/managed-flink/home?region=us-east-1#/applications/dashboard.
2. Di halaman Managed Service for Apache Flink Apache Applications, pilih tab Studio. Pilih Create Studio notebook (Buat notebook Studio).

   > **Note**
   >
   > Untuk membuat notebook Studio dari konsol Amazon MSK atau Kinesis Data Streams pilih klaster Amazon MSK input atau Kinesis data stream Anda, lalu pilih Process data in real time (Proses data secara langsung).

3. Di halaman Buat notebook Studio, berikan informasi berikut:

   - Masukkan **`MyNotebook`** untuk Studio notebook Name (Nama notebook Studio).
   - Pilih default untuk Basis data AWS Glue.

   Pilih Create Studio notebook (Buat notebook Studio).
4. Di MyNotebookhalaman, pilih tab Konfigurasi. Di bagian Jaringan, pilih Edit.

5. Di MyNotebook halaman Edit jaringan untuk, pilih konfigurasi VPC berdasarkan kluster MSK Amazon. Pilih klaster Amazon MSK untuk Amazon MSK Cluster (Klaster Amazon MSK). Pilih Save changes (Simpan perubahan).
6. Di MyNotebookhalaman, pilih Jalankan. Tunggu Status hingga menampilkan Running (Berjalan).

Buat notebook Studio menggunakan AWS CLI

Untuk membuat notebook Studio Anda menggunakan AWS CLI, lakukan hal berikut:

1. Pastikan bahwa Anda memiliki informasi berikut. Anda perlu nilai-nilai ini untuk membuat aplikasi Anda.
   - ID akun Anda.
   - ID subnet dan ID grup keamanan untuk Amazon VPC yang berisi klaster Amazon MSK Anda.
2. Buat file bernama `create.json` dengan konten berikut. Ganti nilai placeholder dengan informasi Anda.

```
{
    "ApplicationName": "MyNotebook",
    "RuntimeEnvironment": "ZEPPELIN-FLINK-3_0",
    "ApplicationMode": "INTERACTIVE",
    "ServiceExecutionRole": "arn:aws:iam::AccountID:role/ZeppelinRole",
    "ApplicationConfiguration": {
        "ApplicationSnapshotConfiguration": {
            "SnapshotsEnabled": false
        },
        "VpcConfigurations": [
            {
                "SubnetIds": [
                    "SubnetID 1",
                    "SubnetID 2",
                    "SubnetID 3"
                ],
                "SecurityGroupIds": [
                    "VPC Security Group ID"
                ]
            }
        ],
        "ZeppelinApplicationConfiguration": {
            "CatalogConfiguration": {
                "GlueDataCatalogConfiguration": {
```

```
                    "DatabaseARN": "arn:aws:glue:us-east-1:AccountID:database/
default"
                }
            }
        }
    }
}
```

3. Jalankan perintah berikut untuk membuat aplikasi Anda.

```
aws kinesisanalyticsv2 create-application --cli-input-json file://create.json
```

4. Setelah perintah selesai, Anda akan melihat output yang serupa dengan yang berikut, yang menampilkan detail untuk notebook Studio baru Anda:

```
{
    "ApplicationDetail": {
        "ApplicationARN": "arn:aws:kinesisanalyticsus-
east-1:012345678901:application/MyNotebook",
        "ApplicationName": "MyNotebook",
        "RuntimeEnvironment": "ZEPPELIN-FLINK-3_0",
        "ApplicationMode": "INTERACTIVE",
        "ServiceExecutionRole": "arn:aws:iam::012345678901:role/ZeppelinRole",
...
```

5. Jalankan perintah berikut untuk memulai aplikasi Anda. Ganti nilai sampel dengan ID akun Anda.

```
aws kinesisanalyticsv2 start-application --application-arn
 arn:aws:kinesisanalyticsus-east-1:012345678901:application/MyNotebook\
```

Kirim data ke klaster Amazon MSK Anda

Di bagian ini, Anda menjalankan skrip Python di klien Amazon EC2 Anda untuk mengirim data ke sumber data Amazon MSK Anda.

1. Sambungkan ke klien Amazon EC2 Anda.
2. Jalankan perintah berikut untuk menginstal Python versi 3, Pip, dan Kafka untuk paket Python, dan mengonfirmasi tindakan:

```
sudo yum install python37
curl -O https://bootstrap.pypa.io/get-pip.py
```

```
python3 get-pip.py --user
pip install kafka-python
```

3. Konfigurasi AWS CLI di mesin klien Anda dengan memasukkan perintah berikut:

```
aws configure
```

Berikan kredensial akun Anda, dan **`us-east-1`** untuk `region`.

4. Buat file bernama `stock.py` dengan konten berikut. Ganti nilai sampel dengan string Bootstrap Brokers cluster Amazon MSK Anda, dan perbarui nama topik jika topik Anda bukan AWSKafkaTutorialTopic:

```
from kafka import KafkaProducer
import json
import random
from datetime import datetime

BROKERS = "<<Bootstrap Broker List>>"
producer = KafkaProducer(
    bootstrap_servers=BROKERS,
    value_serializer=lambda v: json.dumps(v).encode('utf-8'),
    retry_backoff_ms=500,
    request_timeout_ms=20000,
    security_protocol='PLAINTEXT')


def getStock():
    data = {}
    now = datetime.now()
    str_now = now.strftime("%Y-%m-%d %H:%M:%S")
    data['event_time'] = str_now
    data['ticker'] = random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV'])
    price = random.random() * 100
    data['price'] = round(price, 2)
    return data


while True:
    data =getStock()
    # print(data)
    try:
        future = producer.send("AWSKafkaTutorialTopic", value=data)
```

```
        producer.flush()
        record_metadata = future.get(timeout=10)
        print("sent event to Kafka! topic {} partition {} offset
{}".format(record_metadata.topic, record_metadata.partition,
record_metadata.offset))
    except Exception as e:
        print(e.with_traceback())
```

5. Jalankan skrip dengan perintah berikut:

```
$ python3 stock.py
```

6. Biarkan skrip berjalan saat Anda menyelesaikan bagian berikut.

Uji notebook Studio Anda

Di bagian ini, Anda menggunakan notebook Studio Anda untuk mengkueri data dari klaster Amazon MSK Anda.

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/managed-flink/home?region=us-east-1#/applications/dashboard.
2. Pada halaman Managed Service for Apache Flink Apache Applications, pilih tab notebook Studio. Pilih MyNotebook.
3. Di MyNotebookhalaman, pilih Buka di Apache Zeppelin.

   Antarmuka Apache Zeppelin terbuka di tab baru.
4. Di halaman Selamat Datang di Zeppelin!, pilih Zeppelin new note (Catatan baru Zeppelin).
5. Di halaman Zeppelin Note (Catatan Zeppelin), masukkan kueri berikut ke dalam catatan baru:

   ```
   %flink.ssql(type=update)
   select * from stock
   ```

   Pilih ikon jalankan.

   Aplikasi menampilkan data dari klaster Amazon MSK.

Untuk membuka Dasbor Apache Flink untuk aplikasi Anda agar dapat melihat aspek operasional, pilih FLINK JOB (TUGAS FLINK). Untuk informasi selengkapnya tentang Dasbor Flink, lihat Dasbor Apache Flink di Managed Service for Apache Flink Developer Guide.

Untuk contoh kueri SQL Flink Streaming selengkapnya, lihat Kueri di Dokumentasi Apache Flink.

Membersihkan aplikasi Anda dan sumber daya dependen

Hapus notebook Studio Anda

1. Buka Layanan Terkelola untuk konsol Apache Flink.
2. PilihMyNotebook.
3. Pilih Actions (Tindakan), lalu Delete (Hapus).

Hapus Basis Data dan Koneksi AWS Glue Anda

1. Buka konsol AWS Glue di https://console.aws.amazon.com/glue/.
2. Pilih Databases (Basis Data) dari bilah navigasi sebelah kiri. Centang kotak centang di sebelah Default untuk memilihnya. Pilih Action (Tindakan), Delete Database (Hapus Basis Data). Konfirmasikan pilihan Anda.
3. Pilih Connections (Koneksi) dari bilah navigasi sebelah kiri. Centang kotak di sebelahZeppelinConnectionuntuk memilihnya. Pilih Action (Tindakan), Delete Connection (Hapus Koneksi). Konfirmasikan pilihan Anda.

Hapus IAM role dan kebijakan IAM Anda

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pilih Roles (Peran) dari bilah navigasi sebelah kiri.
3. Gunakan bilah pencarian untuk mencariZeppelinRoleperan.
4. PilihZeppelinRoleperan. Pilih Delete Role (Hapus Peran). Konfirmasi penghapusan.

Hapus AndaCloudWatchgrup log

Konsol membuatCloudWatchGrup log dan aliran log untuk Anda saat membuat aplikasi menggunakan konsol. Anda tidak memiliki grup dan aliran log jika Anda membuat aplikasi menggunakan AWS CLI.

1. Buka konsol CloudWatch di https://console.aws.amazon.com/cloudwatch/.
2. Pilih Log groups (Grup log) dari bilah navigasi sebelah kiri.
3. Pilih/AWS/KinesisAnalytics/MyNotebookgrup log.

4. Pilih Actions (Tindakan), Delete log group(s) (Hapus grup log). Konfirmasi penghapusan.

Bersihkan Sumber Daya Kinesis Data Streams

Untuk menghapus aliran Kinesis, buka konsol Kinesis Data Streams, pilih aliran Kinesis, lalu pilih Actions (Tindakan), Delete (Hapus).

Bersihkan sumber daya MSK

Ikuti langkah-langkah di bagian ini jika Anda membuat klaster Amazon MSK untuk tutorial ini. Bagian ini berisi petunjuk untuk membersihkan instans klien Amazon EC2, Amazon VPC, dan klaster Amazon MSK Anda.

Hapus Klaster Amazon MSK Anda

Ikuti langkah-langkah ini jika Anda membuat klaster Amazon MSK untuk tutorial ini.

1. Buka konsol Amazon MSK di https://console.aws.amazon.com/msk/home?region=us-east-1#/home/.
2. PilihAWSKafkaTutorialCluster. Pilih Delete (Hapus). Masukkan **`delete`** di jendela yang muncul, dan konfirmasikan pilihan Anda.

Akhiri intans klien Anda

Ikuti langkah-langkah ini jika Anda membuat instans klien Amazon EC2 untuk tutorial ini.

1. Buka konsol Amazon EC2 di https://console.aws.amazon.com/ec2/.
2. Pilih Instances (Instans) dari panel navigasi sebelah kiri.
3. Pilih kotak centang di sebelahZeppelinClientuntuk memilihnya.
4. Pilih Instance State (Status Instans), Terminate Instance (Akhiri Instans).

Hapus Amazon VPC Anda

Ikuti langkah-langkah ini jika Anda membuat klaster Amazon VPC untuk tutorial ini.

1. Buka konsol Amazon EC2 di https://console.aws.amazon.com/ec2/.
2. Pilih Network Interfaces (Antarmuka Jaringan) dari bilah navigasi sebelah kiri.
3. Masukkan ID VPC Anda di bilah pencarian dan tekan enter untuk mencari.

4. Pilih kotak centang di header tabel untuk memilih semua antarmuka jaringan yang ditampilkan.
5. Pilih Actions (Tindakan), Detach (Lepaskan). Di jendela yang muncul, pilih Enable (Aktifkan) di bawah Force detachment (Lepas paksa). Pilih Detach (Lepaskan), dan tunggu hingga semua antarmuka jaringan mencapai status Available (Tersedia).
6. Pilih kotak centang di header tabel untuk memilih lagi semua antarmuka jaringan yang ditampilkan.
7. Pilih Actions (Tindakan), Delete (Hapus). Konfirmasikan tindakan.
8. Buka konsol Amazon VPC di https://console.aws.amazon.com/vpc/.
9. PilihAWSKafkaTutorialVPC. Pilih Actions (Tindakan), Delete VPC (Hapus VPC). Masukkan **`delete`** dan konfirmasikan penghapusan.

Tutorial: Men-deploy sebagai aplikasi dengan status tahan lama

Tutorial berikut menunjukkan cara menyebarkan notebook Studio sebagai Layanan Terkelola untuk aplikasi Apache Flink dengan status tahan lama.

Tutorial ini berisi bagian-bagian berikut:

- Pengaturan
- Deploy aplikasi dengan status tahan lama menggunakan AWS Management Console
- Deploy Aplikasi dengan Status Tahan Lama Menggunakan AWS CLI

Pengaturan

Buat notebook Studio baru dengan mengikuti Membuat Tutorial notebook Studio, menggunakan Kinesis Data Streams atau Amazon MSK. Beri nama notebook Studio `ExampleTestDeploy`.

Deploy aplikasi dengan status tahan lama menggunakan AWS Management Console

1. Tambahkan lokasi bucket S3 tempat Anda ingin kode yang dikemas disimpan di bawah Lokasi kode aplikasi - opsional di konsol. Ini mengaktifkan langkah-langkah untuk men-deploy dan menjalankan aplikasi Anda langsung dari notebook.
2. Tambahkan izin yang diperlukan ke peran aplikasi untuk mengaktifkan peran yang Anda gunakan untuk membaca dan menulis ke bucket Amazon S3, dan untuk meluncurkan Layanan Terkelola untuk aplikasi Apache Flink:

    - AmazonS3FullAccess

- Amazondikelola-flinkFullAccess
- Akses ke sumber, tujuan, dan VPC Anda sebagaimana berlaku. Untuk informasi selengkapnya, lihat Izin IAM untuk notebook Studio.

3. Gunakan kode sampel berikut:

```
%flink.ssql(type=update)
CREATE TABLE exampleoutput (
  'ticket' VARCHAR,
  'price' DOUBLE
)
WITH (
  'connector' = 'kinesis',
  'stream' = 'ExampleOutputStream',
  'aws.region' = 'us-east-1',
  'scan.stream.initpos' = 'LATEST',
  'format' = 'json'
);

INSERT INTO exampleoutput SELECT ticker, price FROM exampleinputstream
```

4. Dengan peluncuran fitur ini, Anda akan melihat menu menurun baru di sudut kanan atas setiap catatan di notebook Anda dengan nama notebook. Anda dapat melakukan hal berikut:

   - Lihat pengaturan notebook Studio di AWS Management Console.
   - Bangun Zeppelin Note dan ekspor ke Amazon S3. Di titik ini, beri nama aplikasi Anda dan pilih Build and Export (Bangun dan Ekspor). Anda akan mendapatkan notifikasi saat ekspor selesai.
   - Jika perlu, Anda dapat melihat dan menjalankan tes tambahan pada executable di Amazon S3.
   - Setelah selesai dibangun, Anda akan dapat men-deploy kode Anda sebagai aplikasi streaming Kinesis dengan status tahan lama dan penskalaan otomatis.
   - Gunakan menu menurun dan pilih Deploy Zeppelin Note as Kinesis streaming application (Deploy Zeppelin Note sebagai aplikasi streaming Kinesis). Tinjau nama aplikasi dan pilih Deploy via AWS Console (Deploy melalui Konsol).
   - Ini akan membawa Anda keAWS Management Consolehalaman untuk membuat Layanan Terkelola untuk aplikasi Apache Flink. Perhatikan bahwa nama aplikasi, paralelisme, lokasi kode, Glue DB default, VPC (jika berlaku) dan IAM role sudah diisi sebelumnya. Pastikan IAM role memiliki izin yang diperlukan untuk sumber dan tujuan Anda. Snapshot diaktifkan secara default untuk manajemen state aplikasi yang tahan lama.

- Pilih create application (buat aplikasi).
- Anda dapat memilih configure (konfigurasikan) dan mengubah pengaturan apa pun, lalu memilih Run (Jalankan) untuk memulai aplikasi streaming Anda.

Deploy Aplikasi dengan Status Tahan Lama Menggunakan AWS CLI

Untuk men-deploy aplikasi menggunakan AWS CLI, Anda harus memperbarui AWS CLI Anda untuk menggunakan model layanan yang disediakan dengan informasi Beta 2 Anda. Untuk informasi tentang cara menggunakan model layanan yang diperbarui, lihat Pengaturan.

Kode contoh berikut membuat notebook Studio baru:

```
aws kinesisanalyticsv2 create-application \
     --application-name <app-name> \
     --runtime-environment ZEPPELIN-FLINK-3_0 \
     --application-mode INTERACTIVE \
     --service-execution-role <iam-role>
     --application-configuration '{
       "ZeppelinApplicationConfiguration": {
         "CatalogConfiguration": {
           "GlueDataCatalogConfiguration": {
             "DatabaseARN": "arn:aws:glue:us-east-1:<account>:database/<glue-database-
name>"
           }
         }
       },
       "FlinkApplicationConfiguration": {
         "ParallelismConfiguration": {
           "ConfigurationType": "CUSTOM",
           "Parallelism": 4,
           "ParallelismPerKPU": 4
         }
       },
       "DeployAsApplicationConfiguration": {
            "S3ContentLocation": {
               "BucketARN": "arn:aws:s3:::<s3bucket>",
               "BasePath": "/something/"
            }
        },
       "VpcConfigurations": [
         {
           "SecurityGroupIds": [
```

```
            "<security-group>"
          ],
          "SubnetIds": [
            "<subnet-1>",
            "<subnet-2>"
          ]
        }
      ]
    }' \
    --region us-east-1
```

Contoh kode berikut memulai notebook Studio baru:

```
aws kinesisanalyticsv2 start-application \
    --application-name <app-name> \
    --region us-east-1 \
    --no-verify-ssl
```

Kode berikut mengembalikan URL untuk halaman notebook Apache Zeppelin aplikasi:

```
aws kinesisanalyticsv2 create-application-presigned-url \
    --application-name <app-name> \
    --url-type ZEPPELIN_UI_URL \

    --region us-east-1 \
    --no-verify-ssl
```

Contoh-contoh

Kueri contoh berikut menunjukkan cara menganalisis data menggunakan kueri jendela di notebook Studio.

- Membuat tabel dengan Amazon MSK/Apache Kafka
- Membuat tabel dengan Kinesis
- Jendela tumbling
- Jendela geser
- SQL Interaktif
- BlackHole Konektor SQL
- Generator data

- Scala Interaktif
- Python Interaktif
- Python, SQL, dan Scala Interaktif
- Kinesis data stream lintas akun

Untuk informasi tentang pengaturan kueri SQL Apache Flink, lihat Flink pada Notebook Zeppelin untuk Analisis Data Interaktif.

Untuk melihat aplikasi Anda di dasbor Apache Flink, pilih FLINK JOB (TUGAS FLINK) di halaman Zeppelin Note aplikasi Anda.

Untuk informasi selengkapnya tentang kueri jendela, lihat Windows (Jendela) di Dokumentasi Apache Flink.

Untuk contoh kueri SQL Apache Flink Streaming selengkapnya, lihat Kueri di Dokumentasi Apache Flink.

Membuat tabel dengan Amazon MSK/Apache Kafka

Anda dapat menggunakan konektor Amazon MSK Flink dengan Managed Service for Apache Flink Studio untuk mengautentikasi koneksi Anda dengan otentikasi Plaintext, SSL, atau IAM. Buat tabel Anda menggunakan properti spesifik sesuai kebutuhan Anda.

```
-- Plaintext connection

CREATE TABLE your_table (
  `column1` STRING,
  `column2` BIGINT
) WITH (
  'connector' = 'kafka',
  'topic' = 'your_topic',
  'properties.bootstrap.servers' = '<bootstrap servers>',
  'scan.startup.mode' = 'earliest-offset',
  'format' = 'json'
);

-- SSL connection

CREATE TABLE your_table (
  `column1` STRING,
  `column2` BIGINT
```

```
) WITH (
  'connector' = 'kafka',
   'topic' = 'your_topic',
  'properties.bootstrap.servers' = '<bootstrap servers>',
  'properties.security.protocol' = 'SSL',
  'properties.ssl.truststore.location' = '/usr/lib/jvm/java-11-amazon-corretto/lib/
security/cacerts',
  'properties.ssl.truststore.password' = 'changeit',
  'properties.group.id' = 'myGroup',
  'scan.startup.mode' = 'earliest-offset',
  'format' = 'json'
);

-- IAM connection (or for MSK Serverless)

CREATE TABLE your_table (
  `column1` STRING,
  `column2` BIGINT
) WITH (
  'connector' = 'kafka',
  'topic' = 'your_topic',
  'properties.bootstrap.servers' = '<bootstrap servers>',
  'properties.security.protocol' = 'SASL_SSL',
  'properties.sasl.mechanism' = 'AWS_MSK_IAM',
  'properties.sasl.jaas.config' = 'software.amazon.msk.auth.iam.IAMLoginModule
 required;',
  'properties.sasl.client.callback.handler.class' =
 'software.amazon.msk.auth.iam.IAMClientCallbackHandler',
  'properties.group.id' = 'myGroup',
  'scan.startup.mode' = 'earliest-offset',
  'format' = 'json'
);
```

Anda dapat menggabungkan ini dengan properti lain di Apache Kafka SQL Connector.

Membuat tabel dengan Kinesis

Dalam contoh berikut, Anda membuat tabel menggunakan Kinesis:

```
CREATE TABLE KinesisTable (
  `column1` BIGINT,
  `column2` BIGINT,
  `column3` BIGINT,
```

```
  `column4` STRING,
  `ts` TIMESTAMP(3)
)
PARTITIONED BY (column1, column2)
WITH (
  'connector' = 'kinesis',
  'stream' = 'test_stream',
  'aws.region' = '<region>',
  'scan.stream.initpos' = 'LATEST',
  'format' = 'csv'
);
```

Untuk informasi selengkapnya tentang properti lain yang dapat Anda gunakan, lihat Konektor SQL Amazon Kinesis Data Streams.

Jendela tumbling

Kueri SQL Flink Streaming berikut memilih harga tertinggi di setiap jendela tumbling lima detik dari tabel `ZeppelinTopic`:

```
%flink.ssql(type=update)
SELECT TUMBLE_END(event_time, INTERVAL '5' SECOND) as winend, MAX(price) as
 five_second_high, ticker
FROM ZeppelinTopic
GROUP BY ticker, TUMBLE(event_time, INTERVAL '5' SECOND)
```

Jendela geser

Kueri SQL Apache Flink Streaming berikut memilih harga tertinggi di setiap jendela geser lima detik dari tabel `ZeppelinTopic`:

```
%flink.ssql(type=update)
SELECT HOP_END(event_time, INTERVAL '3' SECOND, INTERVAL '5' SECOND) AS winend,
 MAX(price) AS sliding_five_second_max
FROM ZeppelinTopic//or your table name in AWS Glue
GROUP BY HOP(event_time, INTERVAL '3' SECOND, INTERVAL '5' SECOND)
```

SQL Interaktif

Contoh ini mencetak maks. waktu peristiwa dan waktu pemrosesan serta jumlah nilai dari tabel nilai kunci. Pastikan Anda memiliki skrip pembuatan data sampel dari the section called “Generator data”

yang berjalan. Untuk mencoba kueri SQL lainnya seperti filter dan gabung di notebook Studio Anda, lihat dokumentasi Apache Flink: Kueri di dokumentasi Apache Flink.

```
%flink.ssql(type=single, parallelism=4, refreshInterval=1000, template=<h1>{2}</h1>
 records seen until <h1>Processing Time: {1}</h1> and <h1>Event Time: {0}</h1>)

-- An interactive query prints how many records from the `key-value-stream` we have
 seen so far, along with the current processing and event time.
SELECT
  MAX(`et`) as `et`,
  MAX(`pt`) as `pt`,
  SUM(`value`) as `sum`
FROM
  `key-values`
```

```
%flink.ssql(type=update, parallelism=4, refreshInterval=1000)

-- An interactive tumbling window query that displays the number of records observed
 per (event time) second.
-- Browse through the chart views to see different visualizations of the streaming
 result.
SELECT
  TUMBLE_START(`et`, INTERVAL '1' SECONDS) as `window`,
  `key`,
  SUM(`value`) as `sum`
FROM
  `key-values`
GROUP BY
  TUMBLE(`et`, INTERVAL '1' SECONDS),
  `key`;
```

BlackHole Konektor SQL

Konektor BlackHole SQL tidak mengharuskan Anda membuat aliran data Kinesis atau kluster MSK Amazon untuk menguji kueri Anda. Untuk informasi tentang konektor BlackHole SQL, lihat Konektor BlackHole SQL dalam dokumentasi Apache Flink. Dalam contoh ini, katalog default adalah katalog dalam memori.

```
%flink.ssql

CREATE TABLE default_catalog.default_database.blackhole_table (
```

```
 `key` BIGINT,
 `value` BIGINT,
 `et` TIMESTAMP(3)
) WITH (
 'connector' = 'blackhole'
)
```

```
%flink.ssql(parallelism=1)

INSERT INTO `test-target`
SELECT
  `key`,
  `value`,
  `et`
FROM
  `test-source`
WHERE
  `key` > 3
```

```
%flink.ssql(parallelism=2)

INSERT INTO `default_catalog`.`default_database`.`blackhole_table`
SELECT
  `key`,
  `value`,
  `et`
FROM
  `test-target`
WHERE
  `key` > 7
```

Generator data

Contoh ini menggunakan Scala untuk menghasilkan data sampel. Anda dapat menggunakan data sampel ini untuk menguji berbagai kueri. Gunakan pernyataan buat tabel untuk membuat tabel nilai kunci.

```
import org.apache.flink.streaming.api.functions.source.datagen.DataGeneratorSource
import org.apache.flink.streaming.api.functions.source.datagen.RandomGenerator
import org.apache.flink.streaming.api.scala.DataStream
```

```
import java.sql.Timestamp

// ad-hoc convenience methods to be defined on Table
implicit class TableOps[T](table: DataStream[T]) {
    def asView(name: String): DataStream[T] = {
      if (stenv.listTemporaryViews.contains(name)) {
        stenv.dropTemporaryView("`" + name + "`")
      }
      stenv.createTemporaryView("`" + name + "`", table)
      return table;
    }
}
```

```
%flink(parallelism=4)
val stream = senv
 .addSource(new DataGeneratorSource(RandomGenerator.intGenerator(1, 10), 1000))
 .map(key => (key, 1, new Timestamp(System.currentTimeMillis)))
 .asView("key-values-data-generator")
```

```
%flink.ssql(parallelism=4)
-- no need to define the paragraph type with explicit parallelism (such as
 "%flink.ssql(parallelism=2)")
-- in this case the INSERT query will inherit the parallelism of the of the above
 paragraph
INSERT INTO `key-values`
SELECT
 `_1` as `key`,
 `_2` as `value`,
 `_3` as `et`
FROM
 `key-values-data-generator`
```

Scala Interaktif

Ini adalah terjemahan Scala dari the section called “SQL Interaktif”. Untuk contoh Scala lainnya, lihat Tabel API di dokumentasi Apache Flink.

```
%flink
import org.apache.flink.api.scala._
import org.apache.flink.table.api._
import org.apache.flink.table.api.bridge.scala._
```

```
// ad-hoc convenience methods to be defined on Table
implicit class TableOps(table: Table) {
    def asView(name: String): Table = {
      if (stenv.listTemporaryViews.contains(name)) {
        stenv.dropTemporaryView(name)
      }
      stenv.createTemporaryView(name, table)
      return table;
    }
}
```

```
%flink(parallelism=4)

// A view that computes many records from the `key-values` we have seen so far, along
 with the current processing and event time.
val query01 = stenv
  .from("`key-values`")
  .select(
    $"et".max().as("et"),
    $"pt".max().as("pt"),
    $"value".sum().as("sum")
  ).asView("query01")
```

```
%flink.ssql(type=single, parallelism=16, refreshInterval=1000, template=<h1>{2}</h1>
 records seen until <h1>Processing Time: {1}</h1> and <h1>Event Time: {0}</h1>)

-- An interactive query prints the query01 output.
SELECT * FROM query01
```

```
%flink(parallelism=4)

// An tumbling window view that displays the number of records observed per (event
 time) second.
val query02 = stenv
  .from("`key-values`")
  .window(Tumble over 1.seconds on $"et" as $"w")
  .groupBy($"w", $"key")
  .select(
    $"w".start.as("window"),
    $"key",
    $"value".sum().as("sum")
```

```
  ).asView("query02")
```

```
%flink.ssql(type=update, parallelism=4, refreshInterval=1000)

-- An interactive query prints the query02 output.
-- Browse through the chart views to see different visualizations of the streaming
 result.
SELECT * FROM `query02`
```

Python Interaktif

Ini adalah terjemahan Python dari the section called “SQL Interaktif”. Untuk contoh Python lainnya, lihat Tabel API di dokumentasi Apache Flink.

```
%flink.pyflink
from pyflink.table.table import Table

def as_view(table, name):
  if (name in st_env.list_temporary_views()):
    st_env.drop_temporary_view(name)
  st_env.create_temporary_view(name, table)
  return table

Table.as_view = as_view
```

```
%flink.pyflink(parallelism=16)

# A view that computes many records from the `key-values` we have seen so far, along
 with the current processing and event time
st_env \
  .from_path("`keyvalues`") \
  .select(", ".join([
    "max(et) as et",
    "max(pt) as pt",
    "sum(value) as sum"
  ])) \
  .as_view("query01")
```

```
%flink.ssql(type=single, parallelism=16, refreshInterval=1000, template=<h1>{2}</h1>
 records seen until <h1>Processing Time: {1}</h1> and <h1>Event Time: {0}</h1>)
```

```
-- An interactive query prints the query01 output.
SELECT * FROM query01
```

```
%flink.pyflink(parallelism=16)

# A view that computes many records from the `key-values` we have seen so far, along
 with the current processing and event time
st_env \
  .from_path("`key-values`") \
  .window(Tumble.over("1.seconds").on("et").alias("w")) \
  .group_by("w, key") \
  .select(", ".join([
    "w.start as window",
    "key",
    "sum(value) as sum"
  ])) \
  .as_view("query02")
```

```
%flink.ssql(type=update, parallelism=16, refreshInterval=1000)

-- An interactive query prints the query02 output.
-- Browse through the chart views to see different visualizations of the streaming
 result.
SELECT * FROM `query02`
```

Python, SQL, dan Scala Interaktif

Anda dapat menggunakan kombinasi SQL, Python, dan Scala apa pun di notebook Anda untuk analisis interaktif. Dalam notebook Studio yang Anda rencanakan untuk di-deploy sebagai aplikasi dengan status tahan lama, Anda dapat menggunakan kombinasi SQL dan Scala. Contoh ini menunjukkan bagian yang diabaikan dan bagian yang dapat digunakan dalam aplikasi dengan status tahan lama.

```
%flink.ssql
CREATE TABLE `default_catalog`.`default_database`.`my-test-source` (
  `key` BIGINT NOT NULL,
  `value` BIGINT NOT NULL,
  `et` TIMESTAMP(3) NOT NULL,
  `pt` AS PROCTIME(),
  WATERMARK FOR `et` AS `et` - INTERVAL '5' SECOND
)
```

```
WITH (
  'connector' = 'kinesis',
  'stream' = 'kda-notebook-example-test-source-stream',
  'aws.region' = 'eu-west-1',
  'scan.stream.initpos' = 'LATEST',
  'format' = 'json',
  'json.timestamp-format.standard' = 'ISO-8601'
)
```

```
%flink.ssql
CREATE TABLE `default_catalog`.`default_database`.`my-test-target` (
  `key` BIGINT NOT NULL,
  `value` BIGINT NOT NULL,
  `et` TIMESTAMP(3) NOT NULL,
  `pt` AS PROCTIME(),
  WATERMARK FOR `et` AS `et` - INTERVAL '5' SECOND
)
WITH (
  'connector' = 'kinesis',
  'stream' = 'kda-notebook-example-test-target-stream',
  'aws.region' = 'eu-west-1',
  'scan.stream.initpos' = 'LATEST',
  'format' = 'json',
  'json.timestamp-format.standard' = 'ISO-8601'
)
```

```
%flink()

// ad-hoc convenience methods to be defined on Table
implicit class TableOps(table: Table) {
  def asView(name: String): Table = {
    if (stenv.listTemporaryViews.contains(name)) {
      stenv.dropTemporaryView(name)
    }
    stenv.createTemporaryView(name, table)
    return table;
  }
}
```

```
%flink(parallelism=1)
val table = stenv
  .from("`default_catalog`.`default_database`.`my-test-source`")
```

```
  .select($"key", $"value", $"et")
  .filter($"key" > 10)
  .asView("query01")
```

```
%flink.ssql(parallelism=1)

-- forward data
INSERT INTO `default_catalog`.`default_database`.`my-test-target`
SELECT * FROM `query01`
```

```
%flink.ssql(type=update, parallelism=1, refreshInterval=1000)

-- forward data to local stream (ignored when deployed as application)
SELECT * FROM `query01`
```

```
%flink

// tell me the meaning of life (ignored when deployed as application!)
print("42!")
```

Kinesis data stream lintas akun

Untuk menggunakan Kinesis data stream yang ada di akun selain akun yang memiliki notebook Studio, buat peran eksekusi layanan di akun tempat notebook Studio Anda berjalan dan kebijakan kepercayaan peran di akun yang memiliki aliran data. Gunakan `aws.credentials.provider`, `aws.credentials.role.arn`, dan `aws.credentials.role.sessionName` di konektor Kinesis dalam pernyataan DDL buat tabel Anda untuk membuat tabel pada aliran data.

Gunakan peran eksekusi layanan berikut untuk akun notebook Studio.

```
{
 "Sid": "AllowNotebookToAssumeRole",
 "Effect": "Allow",
 "Action": "sts:AssumeRole"
 "Resource": "*"
}
```

Gunakan kebijakan `AmazonKinesisFullAccess` dan kebijakan kepercayaan peran berikut untuk akun aliran data.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Principal": {
 "AWS": "arn:aws:iam::<accountID>:root"
 },
 "Action": "sts:AssumeRole",
 "Condition": {}
 }
 ]
}
```

Gunakan paragraf berikut untuk membuat pernyataan tabel.

```
%flink.ssql
CREATE TABLE test1 (
name VARCHAR,
age BIGINT
) WITH (
'connector' = 'kinesis',
'stream' = 'stream-assume-role-test',
'aws.region' = 'us-east-1',
'aws.credentials.provider' = 'ASSUME_ROLE',
'aws.credentials.role.arn' = 'arn:aws:iam::<accountID>:role/stream-assume-role-test-
role',
'aws.credentials.role.sessionName' = 'stream-assume-role-test-session',
'scan.stream.initpos' = 'TRIM_HORIZON',
'format' = 'json'
)
```

Memecahkan masalah

Bagian ini berisi informasi pemecahan masalah untuk notebook Studio.

Menghentikan aplikasi yang tertahan

Untuk menghentikan aplikasi yang macet dalam keadaan transien, panggil StopApplicationtindakan dengan `Force` parameter yang disetel ke. `true` Untuk informasi selengkapnya, lihat Menjalankan Aplikasi di Managed Service for Apache Flink Developer Guide.

Menyebarkan sebagai aplikasi dengan status tahan lama di VPC tanpa akses internet

deploy-as-application Fungsi Managed Service for Apache Flink Studio tidak mendukung aplikasi VPC tanpa akses internet. Sebaiknya Anda membuat aplikasi di Studio, lalu gunakan Managed Service for Apache Flink untuk membuat aplikasi Flink secara manual dan memilih file zip yang Anda buat di Notebook Anda.

Langkah-langkah berikut menguraikan pendekatan ini:

1. Buat dan ekspor aplikasi Studio Anda ke Amazon S3. Ini harus berupa file zip.
2. Buat Layanan Terkelola untuk aplikasi Apache Flink secara manual dengan jalur kode yang mereferensikan lokasi file zip di Amazon S3. Selain itu, Anda perlu mengkonfigurasi aplikasi dengan `env` variabel berikut (total 2`groupID`, 3`var`):
3. kinesis.analytics.flink.run.options

    a. python: source/note.py

    b. jarfile: PythonApplicationDependencies lib/ .jar
4. terkelola.deploy_as_app.options

    - DatabaSearN: <glue database ARN (Amazon Resource Name) >
5. Anda mungkin perlu memberikan izin ke Layanan Terkelola untuk Apache Flink Studio dan Layanan Terkelola untuk peran IAM Apache Flink untuk layanan yang digunakan aplikasi Anda. Anda dapat menggunakan peran IAM yang sama untuk kedua aplikasi.

eploy-as-app Ukuran D dan pengurangan waktu pembuatan

Studio deploy-as-app untuk aplikasi Python mengemas semua yang tersedia di lingkungan Python karena kami tidak dapat menentukan pustaka mana yang Anda butuhkan. Ini dapat menghasilkan ukuran yang lebih besar dari yang diperlukan. deploy-as-app Prosedur berikut menunjukkan cara mengurangi ukuran ukuran aplikasi deploy-as-app Python dengan menghapus dependensi.

Jika Anda sedang membangun aplikasi Python dengan deploy-as-app fitur dari Studio, Anda dapat mempertimbangkan untuk menghapus paket Python yang sudah diinstal sebelumnya dari sistem jika aplikasi Anda tidak bergantung pada. Ini tidak hanya akan membantu mengurangi ukuran artefak akhir untuk menghindari pelanggaran batas layanan untuk ukuran aplikasi, tetapi juga meningkatkan waktu pembuatan aplikasi dengan fitur tersebut deploy-as-app .

Anda dapat menjalankan perintah berikut untuk mencantumkan semua paket Python yang diinstal dengan ukuran terinstal masing-masing dan secara selektif menghapus paket dengan ukuran yang signifikan.

```
%flink.pyflink

!pip list --format freeze | awk -F = {'print $1'} | xargs pip show | grep -E
 'Location:|Name:' | cut -d ' ' -f 2 | paste -d ' ' - - | awk '{gsub("-","_",$1); print
 $2 "/" tolower($1)}' | xargs du -sh 2> /dev/null | sort -hr
```

> **Note**
>
> `apache-beam`diperlukan oleh Flink Python untuk beroperasi. Anda tidak boleh menghapus paket ini dan dependensinya.

Berikut ini adalah daftar paket Python pra-instal di Studio V2 yang dapat dipertimbangkan untuk dihapus:

```
scipy
statsmodels
plotnine
seaborn
llvmlite
bokeh
pandas
matplotlib
botocore
boto3
numba
```

Untuk menghapus paket Python dari notebook Zeppelin:

1. Periksa apakah aplikasi Anda bergantung pada paket, atau paket konsumsinya, sebelum menghapusnya. Anda dapat mengidentifikasi dependan paket menggunakan pipdeptree.
2. Menjalankan perintah berikut untuk menghapus paket:

   ```
   %flink.pyflink
   !pip uninstall -y <package-to-remove>
   ```

3. Jika Anda perlu mengambil paket yang Anda hapus karena kesalahan, jalankan perintah berikut:

```
%flink.pyflink
!pip install <package-to-install>
```

Example Contoh: Hapus **`scipy`** paket sebelum menerapkan aplikasi deploy-as-app Python Anda dengan fitur.

1. Gunakan `pipdeptree` untuk menemukan semua `scipy` konsumen dan verifikasi apakah Anda dapat menghapus dengan aman`scipy`.

   - Instal alat melalui notebook:

   ```
   %flink.pyflink
   !pip install pipdeptree
   ```

   - Dapatkan pohon ketergantungan terbalik `scipy` dengan menjalankan:

   ```
   %flink.pyflink
   !pip -r -p scipy
   ```

   Anda akan melihat output yang mirip dengan berikut ini (diringkas untuk singkatnya):

   ```
   ...
   ------------------------------------------------------------------------
   scipy==1.8.0
   ### plotnine==0.5.1 [requires: scipy>=1.0.0]
   ### seaborn==0.9.0 [requires: scipy>=0.14.0]
   ### statsmodels==0.12.2 [requires: scipy>=1.1]
       ### plotnine==0.5.1 [requires: statsmodels>=0.8.0]
   ```

2. Hati-hati memeriksa penggunaan`seaborn`, `statsmodels` dan `plotnine` dalam aplikasi Anda. Jika aplikasi Anda tidak bergantung pada salah satu`scipy`,`seaborn`,`statemodels`, atau`plotnine`, Anda dapat menghapus semua paket ini, atau hanya paket yang tidak diperlukan aplikasi Anda.
3. Hapus paket dengan menjalankan:

```
!pip uninstall -y scipy plotnine seaborn statemodels
```

Membatalkan tugas

Bagian ini menunjukkan cara untuk membatalkan tugas Apache Flink yang tidak bisa Anda dapatkan dari Apache Zeppelin. Jika Anda ingin membatalkan tugas seperti itu, buka dasbor Apache Flink, salin ID tugas, lalu gunakan di salah satu contoh berikut.

Untuk membatalkan satu tugas:

```
%flink.pyflink
import requests

requests.patch("https://zeppelin-flink:8082/jobs/[job_id]", verify=False)
```

Untuk membatalkan semua tugas yang sedang berjalan:

```
%flink.pyflink
import requests

r = requests.get("https://zeppelin-flink:8082/jobs", verify=False)
jobs = r.json()['jobs']

for job in jobs:
    if (job["status"] == "RUNNING"):
        print(requests.patch("https://zeppelin-flink:8082/jobs/{}".format(job["id"]),
 verify=False))
```

Untuk membatalkan semua tugas:

```
%flink.pyflink
import requests

r = requests.get("https://zeppelin-flink:8082/jobs", verify=False)
jobs = r.json()['jobs']

for job in jobs:
    requests.patch("https://zeppelin-flink:8082/jobs/{}".format(job["id"]),
 verify=False)
```

Memulai ulang interpreter Apache Flink

Untuk memulai ulang interpreter Apache Flink dalam notebook Studio Anda

1. Pilih Configuration (Konfigurasi) di dekat sudut kanan atas layar.
2. Pilih Interpreter.
3. Pilih restart (mulai ulang), lalu OK.

Lampiran: Membuat kebijakan IAM kustom

Anda biasanya menggunakan kebijakan IAM terkelola untuk mengizinkan aplikasi Anda mengakses sumber daya dependen. Jika Anda memerlukan kontrol yang lebih baik atas izin aplikasi Anda, Anda dapat menggunakan kebijakan IAM kustom. Bagian ini berisi contoh kebijakan IAM kustom.

> **Note**
>
> Dalam contoh kebijakan berikut, ganti teks placeholder dengan nilai-nilai aplikasi Anda.

Topik ini berisi bagian-bagian berikut:

- AWS Glue
- CloudWatch Log
- Aliran Kinesis
- Klaster Amazon MSK

AWS Glue

Kebijakan contoh berikut memberikan izin untuk mengakses basis data AWS Glue.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "GlueTable",
            "Effect": "Allow",
            "Action": [
                "glue:GetConnection",
                "glue:GetTable",
                "glue:GetTables",
                "glue:GetDatabase",
                "glue:CreateTable",
```

```
                "glue:UpdateTable"
            ],
            "Resource": [
                "arn:aws:glue:<region>:<accountId>:connection/*",
                "arn:aws:glue:<region>:<accountId>:table/<database-name>/*",
                "arn:aws:glue:<region>:<accountId>:database/<database-name>",
                "arn:aws:glue:<region>:<accountId>:database/hive",
                "arn:aws:glue:<region>:<accountId>:catalog"
            ]
        },
        {
            "Sid": "GlueDatabase",
            "Effect": "Allow",
            "Action": "glue:GetDatabases",
            "Resource": "*"
        }
    ]
}
```

CloudWatch Log

Kebijakan berikut memberikan izin untuk mengakses CloudWatch Log:

```
{
      "Sid": "ListCloudwatchLogGroups",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "arn:aws:logs:<region>:<accountId>:log-group:*"
      ]
    },
    {
      "Sid": "ListCloudwatchLogStreams",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogStreams"
      ],
      "Resource": [
        "<logGroupArn>:log-stream:*"
      ]
    },
```

```
    {
      "Sid": "PutCloudwatchLogs",
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents"
      ],
      "Resource": [
        "<logStreamArn>"
      ]
    }
```

> **Note**
>
> Jika Anda membuat aplikasi menggunakan konsol, konsol akan menambahkan kebijakan yang diperlukan untuk mengakses CloudWatch Log ke peran aplikasi Anda.

Aliran Kinesis

Aplikasi Anda dapat menggunakan Aliran Kinesis untuk sumber atau tujuan. Aplikasi Anda memerlukan izin baca untuk membaca dari aliran sumber, dan izin tulis untuk menulis ke aliran tujuan.

Kebijakan berikut memberikan izin untuk membaca dari Aliran Kinesis yang digunakan sebagai sumber:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "KinesisShardDiscovery",
      "Effect": "Allow",
      "Action": "kinesis:ListShards",
      "Resource": "*"
    },
    {
      "Sid": "KinesisShardConsumption",
      "Effect": "Allow",
      "Action": [
        "kinesis:GetShardIterator",
        "kinesis:GetRecords",
```

```
        "kinesis:DescribeStream",
        "kinesis:DescribeStreamSummary",
        "kinesis:RegisterStreamConsumer",
        "kinesis:DeregisterStreamConsumer"
      ],
      "Resource": "arn:aws:kinesis:<region>:<accountId>:stream/<stream-name>"
    },
    {
      "Sid": "KinesisEfoConsumer",
      "Effect": "Allow",
      "Action": [
        "kinesis:DescribeStreamConsumer",
        "kinesis:SubscribeToShard"
      ],
      "Resource": "arn:aws:kinesis:<region>:<account>:stream/<stream-name>/consumer/*"
    }
  ]
}
```

Kebijakan berikut memberikan izin untuk menulis ke Aliran Kinesis yang digunakan sebagai tujuan:

```
{
    "Version": "2012-10-17",
    "Statement": [
         {
            "Sid": "KinesisStreamSink",
            "Effect": "Allow",
            "Action": [
                "kinesis:PutRecord",
                "kinesis:PutRecords",
                "kinesis:DescribeStreamSummary",
                "kinesis:DescribeStream"
            ],
            "Resource": "arn:aws:kinesis:<region>:<accountId>:stream/<stream-name>"
        }
    ]
}
```

Jika aplikasi Anda mengakses aliran Kinesis terenkripsi, Anda harus memberikan izin tambahan untuk mengakses aliran dan kunci enkripsi aliran.

Kebijakan berikut memberikan izin untuk mengakses aliran sumber terenkripsi dan kunci enkripsi aliran:

```
{
      "Sid": "ReadEncryptedKinesisStreamSource",
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt"
      ],
      "Resource": [
        "<inputStreamKeyArn>"
      ]
    }
    ,
```

Kebijakan berikut memberikan izin untuk mengakses aliran tujuan terenkripsi dan kunci enkripsi aliran:

```
{
      "Sid": "WriteEncryptedKinesisStreamSink",
      "Effect": "Allow",
      "Action": [
        "kms:GenerateDataKey"
      ],
      "Resource": [
        "<outputStreamKeyArn>"
      ]
    }
```

Klaster Amazon MSK

Untuk memberikan akses ke klaster Amazon MSK, Anda memberikan akses ke VPC klaster. Untuk contoh kebijakan untuk mengakses Amazon VPC, lihat Izin Aplikasi VPC.

Memulai dengan Amazon Managed Service untuk Apache Flink (API) DataStream

Bagian ini memperkenalkan Anda pada konsep dasar Managed Service untuk Apache Flink dan API. DataStream Ini menjelaskan opsi yang tersedia untuk membuat dan menguji aplikasi Anda. Ini juga memberikan petunjuk untuk menginstal alat yang diperlukan untuk menyelesaikan tutorial dalam panduan ini dan untuk membuat aplikasi pertama Anda.

Topik

Komponen Layanan Terkelola untuk Aplikasi Apache Flink

Untuk memproses data, Managed Service untuk aplikasi Apache Flink Anda menggunakan aplikasi Java/Apache Maven atau Scala yang memproses input dan menghasilkan output menggunakan runtime Apache Flink.

Layanan Terkelola untuk aplikasi Apache Flink memiliki komponen-komponen berikut:

- Properti runtime: Anda dapat menggunakan properti runtime untuk mengonfigurasi aplikasi Anda tanpa mengompilasi ulang kode aplikasi Anda.
- Source (Sumber): Aplikasi mengonsumsi data menggunakan sumber. Konektor sumber membaca data dari Kinesis data stream, bucket Amazon S3, dll. Untuk informasi selengkapnya, lihat Sumber.
- Operators (Operator): Aplikasi memproses data menggunakan satu atau beberapa operator. Operator dapat mengubah, memperkaya, atau menggabungkan data. Untuk informasi selengkapnya, lihat DataStream Operator API.

- Sink: Aplikasi menghasilkan data ke sumber eksternal menggunakan sink. Konektor sink menulis data ke aliran data Kinesis, aliran Firehose Data Kinesis, bucket Amazon S3, dll. Untuk informasi selengkapnya, lihat Sink.

Setelah Anda membuat, mengompilasi, dan mengemas kode aplikasi Anda, Anda mengunggah paket kode ke bucket Amazon Simple Storage Service (Amazon S3). Anda kemudian membuat Layanan Terkelola untuk aplikasi Apache Flink. Anda meneruskan di lokasi paket kode, Kinesis data stream sebagai sumber data streaming, dan biasanya lokasi streaming atau file yang menerima data yang diproses dari aplikasi.

Prasyarat untuk Menyelesaikan Latihan

Untuk menyelesaikan langkah-langkah di panduan ini, Anda harus memiliki hal-hal berikut:

- Java Development Kit (JDK) versi 11. Atur variabel lingkungan `JAVA_HOME` untuk menunjuk ke lokasi penginstalan JDK Anda.
- Sebaiknya gunakan lingkungan pengembangan (seperti Eclipse Java Neon atau IntelliJ Idea) untuk mengembangkan dan mengompilasi aplikasi Anda.
- Klien Git. Instal klien Git jika Anda belum menginstalnya.
- Plugin Compiler Apache Maven. Maven harus berada di jalur kerja Anda. Untuk menguji instalasi Apache Maven Anda, masukkan hal berikut:

```
$ mvn -version
```

Untuk memulai, buka Langkah 1: Siapkan Akun AWS dan Buat Pengguna Administrator.

Langkah 1: Siapkan Akun AWS dan Buat Pengguna Administrator

Sebelum Anda menggunakan Managed Service untuk Apache Flink untuk pertama kalinya, selesaikan tugas-tugas berikut:

Mendaftar Akun AWS

Jika Anda tidak memiliki Akun AWS, selesaikan langkah-langkah berikut untuk membuatnya.

Untuk mendaftar Akun AWS

1. Buka https://portal.aws.amazon.com/billing/signup.
2. Ikuti petunjuk secara online.

   Anda akan diminta untuk menerima panggilan telepon dan memasukkan kode verifikasi pada keypad telepon sebagai bagian dari prosedur pendaftaran.

   Saat Anda mendaftar Akun AWS, Pengguna root akun AWS akan dibuat. Pengguna root memiliki akses ke semua Layanan AWS dan sumber daya dalam akun. Sebagai praktik terbaik keamanan, tetapkan akses administratif ke pengguna administratif, dan hanya gunakan pengguna root untuk melakukan tugas yang memerlukan akses pengguna root.

AWS akan mengirimkan email konfirmasi kepada Anda setelah proses pendaftaran selesai. Anda dapat melihat aktivitas akun saat ini dan mengelola akun dengan mengunjungi https://aws.amazon.com/ dan memilih Akun Saya.

Membuat pengguna administratif

Setelah mendaftar Akun AWS, amankan Pengguna root akun AWS, aktifkan AWS IAM Identity Center, dan buat sebuah pengguna administratif sehingga Anda tidak menggunakan pengguna root untuk tugas sehari-hari.

Mengamankan Pengguna root akun AWS Anda

1. Masuk ke AWS Management Console sebagai pemilik akun dengan memilih Pengguna root dan memasukkan alamat email Akun AWS Anda. Di halaman berikutnya, masukkan kata sandi Anda.

   Untuk bantuan masuk menggunakan pengguna root, lihat Masuk sebagai pengguna root dalam Panduan Pengguna AWS Sign-In.
2. Aktifkan autentikasi multi-faktor (MFA) untuk pengguna root Anda.

   Untuk petunjuknya, silakan lihat Mengaktifkan perangkat MFA virtual untuk pengguna root Akun AWS Anda (konsol) dalam Panduan Pengguna IAM.

Membuat pengguna administratif

1. Aktifkan Pusat Identitas IAM.

Untuk mendapatkan petunjuk, silakan lihat Mengaktifkan AWS IAM Identity Center di Panduan Pengguna AWS IAM Identity Center.

2. Di Pusat Identitas IAM, berikan akses administratif ke sebuah pengguna administratif.

   Untuk mendapatkan tutorial tentang menggunakan Direktori Pusat Identitas IAM sebagai sumber identitas Anda, silakan lihat Mengonfigurasi akses pengguna dengan Direktori Pusat Identitas IAM default di Panduan Pengguna AWS IAM Identity Center.

Masuk sebagai pengguna administratif

- Untuk masuk dengan pengguna Pusat Identitas IAM, gunakan URL masuk yang dikirim ke alamat email Anda saat Anda membuat pengguna Pusat Identitas IAM.

  Untuk bantuan masuk menggunakan pengguna Pusat Identitas IAM, lihat Masuk ke portal akses AWS dalam Panduan Pengguna AWS Sign-In.

Memberikan akses terprogram

Pengguna membutuhkan akses terprogram jika mereka ingin berinteraksi dengan AWS luar dari AWS Management Console. Cara memberikan akses terprogram bergantung pada jenis pengguna yang mengakses AWS.

Untuk memberi pengguna akses terprogram, pilih salah satu opsi berikut.

Pengguna mana yang membutuhkan akses terprogram?	Untuk	Oleh
Identitas tenaga kerja (Pengguna yang dikelola di Pusat Identitas IAM)	Gunakan kredensial sementara untuk menandatangani permintaan terprogram ke AWS CLI, SDK AWS, atau API AWS.	Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan. • Untuk AWS CLI, lihat Mengonfigurasi AWS CLI untuk menggunakan AWS IAM Identity Center di

Pengguna mana yang membutuhkan akses terprogram?	Untuk	Oleh
		Panduan Pengguna AWS Command Line Interface. • Untuk SDK AWS, alat, dan API AWS, lihat Autentikasi Pusat Identitas IAM di Panduan Referensi SDK dan Alat AWS.
IAM	Gunakan kredensial sementara untuk menandatangani permintaan terprogram ke AWS CLI, SDK AWS, atau API AWS.	Mengikuti petunjuk dalam Menggunakan kredensial sementara dengan sumber daya AWS di Panduan Pengguna IAM.

Pengguna mana yang membutuhkan akses terprogram?	Untuk	Oleh
IAM	(Tidak direkomendasikan) Gunakan kredensial jangka panjang untuk menandatangani permintaan terprogram ke AWS CLI, SDK AWS, atau API AWS.	Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan. • Untuk AWS CLI, lihat Mengautentikasi menggunakan kredensial pengguna IAM di Panduan Pengguna AWS Command Line Interface. • Untuk SDK dan alat AWS, lihat Mengautentikasi menggunakan kredensial jangka panjang di Panduan Referensi SDK dan Alat AWS. • Untuk API AWS, lihat Mengelola kunci akses untuk pengguna IAM di Panduan Pengguna IAM.

Langkah Selanjutnya

Langkah 2: Siapkan AWS Command Line Interface (AWS CLI)

Langkah 2: Siapkan AWS Command Line Interface (AWS CLI)

Pada langkah ini, Anda mengunduh dan mengonfigurasi AWS CLI untuk digunakan dengan Managed Service for Apache Flink.

> **Note**
>
> Latihan memulai dalam panduan ini mengasumsikan Anda menggunakan kredensial administrator (`adminuser`) di akun Anda untuk melakukan operasi.

> **Note**
>
> Jika sudah menginstal AWS CLI, Anda mungkin perlu meningkatkan untuk mendapatkan fungsionalitas terbaru. Untuk informasi selengkapnya, lihat Menginstal AWS Command Line Interface dalam Panduan Pengguna AWS Command Line Interface. Untuk memeriksa versi AWS CLI, jalankan perintah berikut.
>
> ```
> aws --version
> ```
>
> Latihan dalam tutorial ini memerlukan versi AWS CLI berikut atau yang lebih baru:
>
> ```
> aws-cli/1.16.63
> ```

Untuk mengatur AWS CLI

1. Unduh dan konfigurasikan AWS CLI. Untuk instruksinya, lihat topik berikut di AWS Command Line InterfacePanduan Pengguna:
   - Menginstal AWS Command Line Interface
   - Mengonfigurasi AWS CLI
2. Tambahkan profil bernama untuk pengguna administrator dalam file AWS CLI `config`. Anda dapat menggunakan profil ini saat menjalankan perintah AWS CLI. Untuk informasi selengkapnya tentang profil yang diberi nama, lihat Profil yang Diberi Nama dalam Panduan Pengguna AWS Command Line Interface.

   ```
   [profile adminuser]
   aws_access_key_id = adminuser access key ID
   aws_secret_access_key = adminuser secret access key
   region = aws-region
   ```

Untuk daftar AWS Wilayah yang tersedia, lihat Wilayah dan Titik Akhir di. Referensi Umum Amazon Web Services

> **Note**
>
> Kode dan perintah contoh dalam tutorial ini menggunakan Wilayah US West (Oregon). Untuk menggunakan Wilayah yang berbeda, ubah Wilayah dalam kode dan perintah untuk tutorial ini ke Wilayah yang ingin Anda gunakan.

3. Verifikasikan penyiapan dengan memasukkan perintah bantuan berikut pada prompt perintah.

```
aws help
```

Setelah Anda mengatur AWS akun danAWS CLI, Anda dapat mencoba latihan berikutnya, di mana Anda mengkonfigurasi aplikasi sampel dan menguji end-to-end pengaturan.

Langkah Selanjutnya

Langkah 3: Buat dan Jalankan Layanan Terkelola untuk aplikasi Apache Flink

Langkah 3: Buat dan Jalankan Layanan Terkelola untuk aplikasi Apache Flink

Dalam latihan ini, Anda membuat Layanan Terkelola untuk aplikasi Apache Flink dengan aliran data sebagai sumber dan wastafel.

Bagian ini berisi langkah-langkah berikut.

- Buat Dua Amazon Kinesis Data Streams
- Tulis Catatan Sampel ke Aliran Input
- Unduh dan Periksa Kode Java Streaming Apache Flink
- Kompilasi Kode Aplikasi
- Unggah Kode Java Streaming Apache Flink
- Buat dan Jalankan Managed Service untuk Apache Flink Application
- Langkah Selanjutnya

Buat Dua Amazon Kinesis Data Streams

Sebelum Anda membuat Layanan Terkelola untuk aplikasi Apache Flink untuk latihan ini, buat dua aliran data Kinesis (dan). `ExampleInputStream` `ExampleOutputStream` Aplikasi Anda menggunakan aliran ini untuk sumber aplikasi dan aliran tujuan.

Anda dapat membuat aliran ini menggunakan konsol Amazon Kinesis atau perintah AWS CLI berikut. Untuk instruksi konsol, lihat Membuat dan Memperbarui Aliran Data di Panduan Developer Amazon Kinesis Data Streams.

Untuk membuat aliran data AWS CLI

1. Untuk membuat aliran pertama (`ExampleInputStream`), gunakan perintah `create-stream` AWS CLI Amazon Kinesis berikut.

```
$ aws kinesis create-stream \
--stream-name ExampleInputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

2. Untuk membuat aliran kedua yang digunakan aplikasi untuk menulis output, jalankan perintah yang sama, yang mengubah nama aliran menjadi `ExampleOutputStream`.

```
$ aws kinesis create-stream \
--stream-name ExampleOutputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

Tulis Catatan Sampel ke Aliran Input

Di bagian ini, Anda menggunakan script Python untuk menulis catatan sampel ke aliran untuk diproses aplikasi.

> **Note**
>
> Bagian ini memerlukan AWS SDK for Python (Boto).

1. Buat file bernama `stock.py` dengan konten berikut:

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        'event_time': datetime.datetime.now().isoformat(),
        'ticker': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'price': round(random.random() * 100, 2)}


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(data),
            PartitionKey="partitionkey")


if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis', region_name='us-west-2'))
```

2. Selanjutnya dalam tutorial ini, Anda menjalankan skrip `stock.py` untuk mengirim data ke aplikasi.

```
$ python stock.py
```

Unduh dan Periksa Kode Java Streaming Apache Flink

Kode aplikasi Java untuk contoh ini tersedia dari GitHub. Untuk mengunduh kode aplikasi, lakukan hal berikut:

1. Kloning repositori jarak jauh menggunakan perintah berikut:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

2. Buka direktori `amazon-kinesis-data-analytics-java-examples/GettingStarted` tersebut.

Perhatikan hal tentang kode aplikasi berikut:

- File Project Object Model (pom.xml) berisi informasi tentang konfigurasi dan dependensi aplikasi, termasuk Layanan Terkelola untuk pustaka Apache Flink.
- File `BasicStreamingJob.java` berisi metode `main` yang menentukan fungsionalitas aplikasi.
- Aplikasi menggunakan sumber Kinesis untuk membaca dari aliran sumber. Cuplikan berikut ini membuat sumber Kinesis:

```
return env.addSource(new FlinkKinesisConsumer<>(inputStreamName,
                new SimpleStringSchema(), inputProperties));
```

- Aplikasi Anda membuat konektor sumber dan sink untuk mengakses sumber daya eksternal menggunakan objek `StreamExecutionEnvironment`.
- Aplikasi membuat konektor sumber dan sink menggunakan properti statis. Untuk menggunakan properti aplikasi dinamis, gunakan metode `createSourceFromApplicationProperties` dan `createSinkFromApplicationProperties` untuk membuat konektor. Metode ini membaca properti aplikasi untuk mengonfigurasi konektor.

  Untuk informasi selengkapnya tentang properti runtime, lihat Properti Runtime.

Kompilasi Kode Aplikasi

Di bagian ini, Anda menggunakan compiler Apache Maven untuk membuat kode Java untuk aplikasi. Untuk informasi tentang menginstal Apache Maven dan Java Development Kit (JDK), lihat Prasyarat untuk Menyelesaikan Latihan.

Untuk mengompikasi kode aplikasi

1. Untuk menggunakan kode aplikasi Anda, Anda mengompilasi dan mengemasnya ke dalam file JAR. Anda dapat mengompilasi dan mengemas kode Anda dengan salah satu dari dua cara:

- Gunakan alat Maven baris perintah. Buat file JAR Anda dengan menjalankan perintah berikut di direktori yang berisi file `pom.xml`:

```
mvn package -Dflink.version=1.15.3
```

- Menyiapkan lingkungan pengembangan Anda. Lihat dokumentasi lingkungan pengembangan Anda untuk detail.

> **Note**
>
> Kode sumber yang disediakan bergantung pada pustaka dari Java 11.

Anda dapat mengunggah paket Anda sebagai file JAR, atau Anda dapat mengompresi paket Anda dan mengunggahnya sebagai file ZIP. Jika Anda membuat aplikasi Anda menggunakan AWS CLI, Anda menentukan tipe konten kode Anda (JAR atau ZIP).

2. Jika ada kesalahan saat mengompilasi, pastikan variabel lingkungan `JAVA_HOME`Anda diatur dengan benar.

Jika aplikasi berhasil mengompilasi, file berikut dibuat:

`target/aws-kinesis-analytics-java-apps-1.0.jar`

Unggah Kode Java Streaming Apache Flink

Pada bagian ini, Anda membuat bucket Amazon Simple Storage Service (Amazon S3) dan mengunggah kode aplikasi Anda.

Untuk mengunggah kode aplikasi

1. Buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.
2. Pilih Buat bucket.
3. Masukkan **`ka-app-code-<username>`** di bidang Bucket name (Nama bucket). Tambahkan sufiks ke nama bucket, seperti nama pengguna Anda, untuk membuatnya unik secara global. Pilih Next (Selanjutnya).
4. Di langkah Konfigurasi opsi, jangan ubah pengaturan, dan pilih Next (Selanjutnya).
5. Di langkah Atur izin, jangan ubah pengaturan, dan pilih Next (Selanjutnya).

6. Pilih Create bucket (Buat bucket).
7. Di konsol Amazon S3, pilih bucket ka-app-code -, dan pilih Unggah. <username>
8. Di langkah Pilih file, pilih Add files (Tambahkan berkas). Navigasikan ke file `aws-kinesis-analytics-java-apps-1.0.jar` yang Anda buat di langkah sebelumnya. Pilih Berikutnya.
9. Anda tidak perlu mengubah pengaturan apa pun untuk objek, jadi pilih Upload (Unggah).

Kode aplikasi Anda sekarang disimpan di bucket Amazon S3 yang dapat diakses aplikasi Anda.

Buat dan Jalankan Managed Service untuk Apache Flink Application

Anda dapat membuat dan menjalankan Layanan Terkelola untuk aplikasi Apache Flink menggunakan konsol atau aplikasi. AWS CLI

> **Note**
>
> Saat Anda membuat aplikasi menggunakan konsol, sumber daya AWS Identity and Access Management (IAM) dan Amazon CloudWatch Logs dibuat untuk Anda. Saat Anda membuat aplikasi menggunakan AWS CLI, Anda membuat sumber daya ini secara terpisah.

Topik

- Buat dan Jalankan Aplikasi (Konsol)
- Buat dan Jalankan Aplikasi (AWS CLI)

Buat dan Jalankan Aplikasi (Konsol)

Ikuti langkah-langkah ini untuk membuat, mengonfigurasi, memperbarui, dan menjalankan aplikasi menggunakan konsol.

Buat Aplikasi

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/flink
2. Pada dashboard Managed Service for Apache Flink, pilih Create Analytics Application.
3. Pada Layanan Terkelola untuk Apache Flink - Buat halaman aplikasi, berikan detail aplikasi sebagai berikut:

   - Untuk Application name (Nama aplikasi), masukkan **`MyApplication`**.

- Untuk Description (Deskripsi), masukkan **`My java test app`**.
- Untuk Runtime, pilih Apache Flink.
- Biarkan versi pulldown sebagai Apache Flink versi 1.15.2 (Versi yang disarankan).

4. Untuk Access permissions (Izin akses), pilih Create / update IAM role **`kinesis-analytics-MyApplication-us-west-2`** (Buat/perbarui IAM role).
5. Pilih Create application (Buat aplikasi).

> **Note**
>
> Saat membuat aplikasi Managed Service for Apache Flink menggunakan konsol, Anda memiliki opsi untuk membuat peran dan kebijakan IAM untuk aplikasi Anda. Aplikasi Anda menggunakan peran dan kebijakan ini untuk mengakses sumber daya dependen. Sumber daya IAM ini diberi nama menggunakan nama aplikasi dan Wilayah sebagai berikut:
>
> - Kebijakan: `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Peran: `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Edit Kebijakan IAM

Edit kebijakan IAM untuk menambahkan izin mengakses Kinesis data streams.

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pilih Policies (Kebijakan). Pilih kebijakan **`kinesis-analytics-service-MyApplication-us-west-2`** yang dibuat konsol untuk Anda di bagian sebelumnya.
3. Di halaman Ringkasan, pilih Edit policy (Edit kebijakan). Pilih tab JSON.
4. Tambahkan bagian yang disorot dari contoh kebijakan berikut ke kebijakan. Ganti ID akun sampel (*`012345678901`*) dengan ID akun Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
```

```
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ka-app-code-username/aws-kinesis-analytics-java-
apps-1.0.jar"
            ]
        },
        {
            "Sid": "DescribeLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
            ]
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
```

```
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

Konfigurasikan Aplikasi

1. Pada MyApplicationhalaman, pilih Konfigurasi.
2. Di halaman Konfigurasikan aplikasi, berikan Code location (Lokasi kode):
   - Untuk Bucket Amazon S3, masukkan **ka-app-code-*<username>***.
   - Untuk Jalur ke objek Amazon S3, masukkan **aws-kinesis-analytics-java-apps-1.0.jar**.
3. Di bawah Akses ke sumber daya aplikasi, untuk Access permissions (Izin akses), pilih Create / update IAM role **kinesis-analytics-MyApplication-us-west-2** (Pilih/perbarui IAM role).
4. Di bawah Properties (Properti), untuk Group ID (ID Grup), masukkan **ProducerConfigProperties**.
5. Masukkan properti dan nilai aplikasi berikut:

ID Grup	Kunci	Nilai
ProducerConfigProperties	**flink.inputstream.initpos**	**LATEST**
ProducerConfigProperties	**aws.region**	**us-west-2**

ID Grup	Kunci	Nilai
ProducerConfigProperties	**AggregationEnabled**	**false**

6. Di bawah Pemantauan, pastikan Memantau tingkat metrik diatur ke Aplikasi.
7. Untuk CloudWatch logging, pilih kotak centang Aktifkan.
8. Pilih Perbarui.

> **Note**
>
> Saat Anda memilih untuk mengaktifkan CloudWatch pencatatan Amazon, Layanan Terkelola untuk Apache Flink membuat grup log dan aliran log untuk Anda. Nama-nama sumber daya ini adalah sebagai berikut:
>
> - Grup log: `/aws/kinesis-analytics/MyApplication`
> - Aliran log: `kinesis-analytics-log-stream`

Jalankan Aplikasi

Grafik pekerjaan Flink dapat dilihat dengan menjalankan aplikasi, membuka dasbor Apache Flink, dan memilih pekerjaan Flink yang diinginkan.

Hentikan Aplikasi

Pada MyApplicationhalaman, pilih Berhenti. Konfirmasikan tindakan.

Perbarui Aplikasi

Dengan menggunakan konsol, Anda dapat memperbarui pengaturan aplikasi seperti properti aplikasi, pengaturan pemantauan, dan lokasi atau nama file dari JAR aplikasi. Anda juga dapat memuat ulang aplikasi JAR dari bucket Amazon S3 jika Anda perlu memperbarui kode aplikasi.

Pada MyApplicationhalaman, pilih Konfigurasi. Perbarui pengaturan aplikasi dan pilih Update (Perbarui).

Buat dan Jalankan Aplikasi (AWS CLI)

Di bagian ini, Anda menggunakan AWS CLI untuk membuat dan menjalankan aplikasi Managed Service for Apache Flink. Managed Service untuk Apache Flink menggunakan `kinesisanalyticsv2` AWS CLI perintah untuk membuat dan berinteraksi dengan Managed Service untuk aplikasi Apache Flink.

Membuat Kebijakan Izin

> **Note**
>
> Anda harus membuat kebijakan izin dan peran untuk aplikasi Anda. Jika Anda tidak membuat sumber daya IAM ini, aplikasi Anda tidak dapat mengakses data dan aliran log.

Pertama, Anda membuat kebijakan izin dengan dua pernyataan: satu yang memberikan izin untuk tindakan `read` di aliran sumber, dan lainnya yang memberikan izin untuk tindakan `write` di aliran sink. Anda selanjutnya melampirkan kebijakan ke IAM role (yang Anda buat di bagian berikutnya). Jadi, ketika Layanan Terkelola untuk Apache Flink mengasumsikan peran tersebut, layanan memiliki izin yang diperlukan untuk membaca dari aliran sumber dan menulis ke aliran wastafel.

Gunakan kode berikut untuk membuat kebijakan izin `AKReadSourceStreamWriteSinkStream`. Ganti *`username`* dengan nama pengguna yang Anda gunakan untuk membuat bucket Amazon S3 untuk menyimpan kode aplikasi. Ganti ID akun di Amazon Resource Names (ARN) (*`012345678901`*) dengan ID akun Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": ["arn:aws:s3:::ka-app-code-username",
                "arn:aws:s3:::ka-app-code-username/*"
            ]
        },
```

```
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

Untuk step-by-step petunjuk membuat kebijakan izin, lihat Tutorial: Membuat dan Melampirkan Kebijakan Terkelola Pelanggan Pertama Anda di Panduan Pengguna IAM.

> **Note**
>
> Untuk mengakses layanan Amazon lainnya, Anda dapat menggunakan AWS SDK for Java. Layanan Terkelola untuk Apache Flink secara otomatis menyetel kredensyal yang diperlukan oleh SDK ke peran IAM eksekusi layanan yang terkait dengan aplikasi Anda. Tidak ada langkah tambahan yang diperlukan.

Buat IAM Role

Di bagian ini, Anda membuat peran IAM yang dapat diasumsikan oleh aplikasi Managed Service for Apache Flink untuk membaca aliran sumber dan menulis ke aliran sink.

Layanan Terkelola untuk Apache Flink tidak dapat mengakses aliran Anda tanpa izin. Anda memberikan izin ini melalui IAM role. Setiap IAM role memiliki dua kebijakan yang dilampirkan. Kebijakan kepercayaan memberikan izin Layanan Terkelola untuk Apache Flink untuk mengambil peran, dan kebijakan izin menentukan apa yang dapat dilakukan Layanan Terkelola untuk Apache Flink setelah mengambil peran.

Anda melampirkan kebijakan izin yang Anda buat di bagian sebelumnya ke peran ini.

Untuk membuat IAM role

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Dalam panel navigasi, pilih Roles (Peran), Create role (Buat Peran).
3. Di bawah Pilih tipe identitas tepercaya, pilih Layanan AWS. Di bawah Pilih layanan yang akan menggunakan peran ini, pilih Kinesis. Di bawah Pilih kasus penggunaan Anda, pilih Analitik Kinesis.

   Pilih Selanjutnya: Izin.
4. Di halaman Lampirkan kebijakan izin, pilih Next: Review (Selanjutnya: Tinjau). Anda melampirkan kebijakan izin setelah Anda membuat peran tersebut.
5. Di halaman Buat peran, masukkan **`MF-stream-rw-role`** untuk Role name (Nama peran). Pilih Create role (Buat peran).

   Sekarang Anda sudah membuat IAM role baru yang disebut `MF-stream-rw-role`. Berikutnya, Anda memperbarui kebijakan kepercayaan dan izin untuk peran.
6. Lampirkan kebijakan izin ke peran tersebut.

   > **Note**
   >
   > Untuk latihan ini, Managed Service for Apache Flink mengasumsikan peran ini untuk membaca data dari aliran data Kinesis (sumber) dan menulis output ke aliran data Kinesis lain. Jadi, Anda melampirkan kebijakan yang Anda buat di langkah sebelumnya, the section called “Membuat Kebijakan Izin”.

   a. Di halaman Ringkasan, pilih tab Permissions (Izin).

   b. Pilih Attach Policies (Lampirkan Kebijakan).

   c. Di kotak pencarian, masukkan **`AKReadSourceStreamWriteSinkStream`** (kebijakan yang Anda buat bagian sebelumnya).

   d. Pilih ReadSourceStreamWriteSinkStream kebijakan AK, dan pilih Lampirkan kebijakan.

Anda sekarang telah membuat peran eksekusi layanan yang digunakan aplikasi Anda untuk mengakses sumber daya. Buat catatan ARN peran baru.

Untuk step-by-step petunjuk cara membuat peran, lihat Membuat Peran IAM (Konsol) di Panduan Pengguna IAM.

Buat Managed Service untuk Apache Flink Application

1. Simpan kode JSON berikut ke file bernama `create_request.json`. Ganti ARN peran sampel dengan ARN untuk peran yang Anda buat sebelumnya. Ganti sufiks ARN bucket (*username*) dengan sufiks yang Anda pilih di bagian sebelumnya. Ganti ID akun sampel (*012345678901*) di peran eksekusi layanan dengan ID akun Anda.

```
{
    "ApplicationName": "test",
    "ApplicationDescription": "my java test app",
    "RuntimeEnvironment": "FLINK-1_15",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-stream-rw-role",
    "ApplicationConfiguration": {
        "ApplicationCodeConfiguration": {
            "CodeContent": {
                "S3ContentLocation": {
                    "BucketARN": "arn:aws:s3:::ka-app-code-username",
                    "FileKey": "aws-kinesis-analytics-java-apps-1.0.jar"
                }
            },
            "CodeContentType": "ZIPFILE"
        },
        "EnvironmentProperties":  {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "flink.stream.initpos" : "LATEST",
                    "aws.region" : "us-west-2",
                    "AggregationEnabled" : "false"
               }
            },
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2"
               }
            }
         ]
      }
```

```
    }
}
```

2. Jalankan tindakan `CreateApplication` dengan permintaan sebelumnya untuk membuat aplikasi:

```
aws kinesisanalyticsv2 create-application --cli-input-json file://
create_request.json
```

Aplikasi ini sekarang dibuat. Anda memulai aplikasi di langkah berikutnya.

Mulai Aplikasi

Di bagian ini, Anda menggunakan tindakan `StartApplication` untuk memulai aplikasi.

Untuk memulai aplikasi

1. Simpan kode JSON berikut ke file bernama `start_request.json`.

```
{
    "ApplicationName": "test",
    "RunConfiguration": {
        "ApplicationRestoreConfiguration": {
         "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"
         }
    }
}
```

2. Jalankan tindakan `StartApplication` dengan permintaan sebelumnya untuk memulai aplikasi:

```
aws kinesisanalyticsv2 start-application --cli-input-json file://start_request.json
```

Aplikasi sekarang berjalan. Anda dapat memeriksa metrik Layanan Terkelola untuk Apache Flink di CloudWatch konsol Amazon untuk memverifikasi bahwa aplikasi berfungsi.

Hentikan Aplikasi

Di bagian ini, Anda menggunakan tindakan `StopApplication` untuk menghentikan aplikasi.

Untuk menghentikan aplikasi

1. Simpan kode JSON berikut ke file bernama `stop_request.json`.

```
{
   "ApplicationName": "test"
}
```

2. Jalankan tindakan `StopApplication` dengan permintaan berikut untuk menghentikan aplikasi:

```
aws kinesisanalyticsv2 stop-application --cli-input-json file://stop_request.json
```

Aplikasi sekarang dihentikan.

Tambahkan Opsi CloudWatch Logging

Anda dapat menggunakan AWS CLI untuk menambahkan aliran CloudWatch log Amazon ke aplikasi Anda. Untuk informasi tentang menggunakan CloudWatch Log dengan aplikasi Anda, lihatthe section called “Menyiapkan Pencatatan”.

Perbarui Properti Lingkungan

Di bagian ini, Anda menggunakan tindakan `UpdateApplication` untuk mengubah properti lingkungan untuk aplikasi tanpa mengompilasi ulang kode aplikasi. Dalam contoh ini, Anda mengubah Wilayah aliran sumber dan tujuan.

Untuk memperbarui properti lingkungan untuk aplikasi

1. Simpan kode JSON berikut ke file bernama `update_properties_request.json`.

```
{"ApplicationName": "test",
   "CurrentApplicationVersionId": 1,
   "ApplicationConfigurationUpdate": {
      "EnvironmentPropertyUpdates": {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "flink.stream.initpos" : "LATEST",
                    "aws.region" : "us-west-2",
                    "AggregationEnabled" : "false"
```

```
                }
            },
            {
                "PropertyGroupId": "ConsumerConfigProperties",
                "PropertyMap" : {
                     "aws.region" : "us-west-2"
                }
            }
         ]
      }
   }
}
```

2. Jalankan tindakan UpdateApplication dengan permintaan sebelumnya untuk memperbarui properti lingkungan:

```
aws kinesisanalyticsv2 update-application --cli-input-json file://
update_properties_request.json
```

Perbarui Kode Aplikasi

Ketika Anda perlu memperbarui kode aplikasi Anda dengan versi baru paket kode Anda, Anda menggunakan tindakan UpdateApplication AWS CLI.

> **Note**
>
> Untuk memuat versi baru kode aplikasi dengan nama file yang sama, Anda harus menentukan versi objek baru. Untuk informasi selengkapnya tentang menggunakan versi objek Amazon S3, lihat Mengaktifkan dan Menonaktifkan Versioning.

Untuk menggunakan AWS CLI, hapus paket kode Anda sebelumnya dari bucket Amazon S3 Anda, unggah versi baru, dan panggil `UpdateApplication`, yang menentukan bucket Amazon S3 dan nama objek yang sama, serta versi objek baru. Aplikasi akan memulai ulang dengan paket kode baru.

Permintaan sampel berikut untuk tindakan `UpdateApplication` memuat ulang kode aplikasi dan memulai ulang aplikasi. Perbarui `CurrentApplicationVersionId` ke versi aplikasi saat ini. Anda dapat memeriksa versi aplikasi saat ini menggunakan tindakan `ListApplications` atau `DescribeApplication`. Perbarui sufiks nama bucket (*<username>*) dengan sufiks yang Anda pilih di bagian the section called “Buat Dua Amazon Kinesis Data Streams”.

```
{
    "ApplicationName": "test",
    "CurrentApplicationVersionId": 1,
    "ApplicationConfigurationUpdate": {
        "ApplicationCodeConfigurationUpdate": {
            "CodeContentUpdate": {
                "S3ContentLocationUpdate": {
                    "BucketARNUpdate": "arn:aws:s3:::ka-app-code-username",
                    "FileKeyUpdate": "aws-kinesis-analytics-java-apps-1.0.jar",
                    "ObjectVersionUpdate": "SAMPLEUehYngP87ex1nzYIGYgfhypvDU"
                }
            }
        }
    }
}
```

Langkah Selanjutnya

[Langkah 4: Bersihkan Sumber Daya AWS]

Langkah 4: Bersihkan Sumber Daya AWS

Bagian ini mencakup prosedur untuk membersihkan sumber daya AWS yang dibuat dalam tutorial Memulai.

Topik ini berisi bagian-bagian berikut:

- Hapus Layanan Terkelola Anda untuk Aplikasi Apache Flink
- Hapus Kinesis Data Streams Anda
- Hapus Objek dan Bucket Amazon S3 Anda
- Hapus Sumber Daya IAM Anda
- Hapus CloudWatch Sumber Daya Anda
- Langkah Selanjutnya

Hapus Layanan Terkelola Anda untuk Aplikasi Apache Flink

1. Buka konsol Kinesis di https://console.aws.amazon.com/kinesis.
2. Di panel Managed Service for Apache Flink, pilih. MyApplication

3. Di halaman aplikasi, pilih Delete (Hapus), lalu konfirmasikan penghapusan.

Hapus Kinesis Data Streams Anda

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/flink
2. Di panel Kinesis Data Streams, pilih. ExampleInputStream
3. Di ExampleInputStreamhalaman, pilih Hapus Stream Kinesis dan kemudian konfirmasikan penghapusan.
4. Di halaman Kinesis streams, pilih, pilih Tindakan ExampleOutputStream, pilih Hapus, lalu konfirmasikan penghapusan.

Hapus Objek dan Bucket Amazon S3 Anda

1. Buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.
2. Pilih ka-app-code- ember. <username>
3. Pilih Delete (Hapus), lalu masukkan nama bucket untuk mengonfirmasi penghapusan.

Hapus Sumber Daya IAM Anda

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di bilah navigasi, pilih Policies (Kebijakan).
3. Di kontrol filter, masukkan kinesis.
4. Pilih kebijakan kinesis-analytics-service- MyApplication -us-west-2.
5. Pilih Policy Actions (Tindakan Kebijakan), lalu pilih Delete (Hapus).
6. Di bilah navigasi, pilih Roles (Peran).
7. Pilih peran kinesis-analytics- -us-west-2. MyApplication
8. Pilih Delete role (Hapus peran), lalu konfirmasi penghapusan.

Hapus CloudWatch Sumber Daya Anda

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di bilah navigasi, pilih Logs.

3. Pilih grup log MyApplication/aws/kinesis-analytics/.
4. Pilih Delete Log Group (Hapus Grup Log), lalu konfirmasi penghapusan.

Langkah Selanjutnya

Langkah 5: Langkah selanjutnya

Langkah 5: Langkah selanjutnya

Sekarang setelah Anda membuat dan menjalankan Layanan Terkelola dasar untuk aplikasi Apache Flink, lihat sumber daya berikut untuk solusi Managed Service for Apache Flink yang lebih canggih.

- Solusi Data Streaming AWS untuk Amazon Kinesis: Solusi Data Streaming AWS untuk Amazon Kinesis secara otomatis mengonfigurasi layanan AWS yang diperlukan untuk dengan mudah menangkap, menyimpan, memproses, dan mengirimkan data streaming. Solusi ini menyediakan beberapa opsi untuk memecahkan kasus penggunaan data streaming. Layanan Terkelola untuk opsi Apache Flink menyediakan contoh ETL end-to-end streaming yang menunjukkan aplikasi dunia nyata yang menjalankan operasi analitis pada data taksi New York yang disimulasikan. Solusinya menyiapkan semua AWS sumber daya yang diperlukan seperti peran dan kebijakan IAM, CloudWatch dasbor, dan CloudWatch alarm.
- AWSSolusi Data Streaming untuk Amazon MSK: Solusi Data AWS Streaming untuk Amazon MSK menyediakan AWS CloudFormation template tempat data mengalir melalui produsen, penyimpanan streaming, konsumen, dan tujuan.
- Clickstream Lab dengan Apache Flink dan Apache Kafka: Lab ujung ke ujung untuk kasus penggunaan clickstream menggunakan Amazon Managed Streaming untuk Apache Kafka Kafka untuk penyimpanan streaming dan Layanan Terkelola untuk Apache Flink untuk aplikasi Apache Flink untuk pemrosesan streaming.
- Amazon Managed Service untuk Apache Flink Workshop: Dalam lokakarya ini, Anda membangun arsitektur end-to-end streaming untuk menelan, menganalisis, dan memvisualisasikan data streaming dalam waktu dekat. Anda mulai meningkatkan operasi perusahaan taksi di Kota New York. Anda menganalisis data telemetri armada taksi di Kota New York hampir secara langsung untuk mengoptimalkan operasi armada mereka.
- Layanan Terkelola untuk Apache Flink: Contoh: Bagian Panduan Pengembang ini memberikan contoh membuat dan bekerja dengan aplikasi di Managed Service untuk Apache Flink. Mereka menyertakan kode contoh dan step-by-step instruksi untuk membantu Anda membuat Layanan Terkelola untuk aplikasi Apache Flink dan menguji hasil Anda.

- Belajar Flink: Pelatihan Langsung: Pelatihan pengenalan resmi Apache Flink yang membuat Anda mulai menulis ETL streaming yang dapat diskalakan, analitik, dan aplikasi yang didorong peristiwa.

> **Note**
>
> Ketahuilah bahwa Managed Service for Apache Flink tidak mendukung versi Apache Flink (1.12) yang digunakan dalam pelatihan ini. Anda dapat menggunakan Flink 1.15.2 di Flink Managed Service untuk Apache Flink.

Memulai dengan Amazon Managed Service untuk Apache Flink (Tabel API)

Bagian ini memperkenalkan Anda pada konsep dasar Managed Service untuk Apache Flink dan Table API. Ini menjelaskan opsi yang tersedia untuk membuat dan menguji aplikasi Anda. Ini juga memberikan petunjuk untuk menginstal alat yang diperlukan untuk menyelesaikan tutorial dalam panduan ini dan untuk membuat aplikasi pertama Anda.

Topik

Komponen Layanan Terkelola untuk Aplikasi Apache Flink

Untuk memproses data, Managed Service untuk aplikasi Apache Flink Anda menggunakan aplikasi Java/Apache Maven atau Scala yang memproses input dan menghasilkan output menggunakan runtime Apache Flink.

Managed Service untuk aplikasi Apache Flink memiliki komponen-komponen berikut:

- Properti runtime: Anda dapat menggunakan properti runtime untuk mengonfigurasi aplikasi Anda tanpa mengompilasi ulang kode aplikasi Anda.
- Sumber Tabel: Aplikasi mengonsumsi data menggunakan sumber. Konektor sumber membaca data dari Kinesis data stream, topik Amazon MSK, atau yang serupa. Untuk informasi selengkapnya, lihat Sumber API Tabel.
- Fungsi: Aplikasi memproses data menggunakan satu atau beberapa fungsi. Fungsi dapat mengubah, memperkaya, atau menggabungkan data.
- Sink: Aplikasi menghasilkan data ke sumber eksternal menggunakan sink. Konektor sink menulis data ke aliran data Kinesis, aliran Firehose Data Kinesis Data Firehose Kinesis, topik MSK Amazon, bucket Amazon S3, dan sebagainya. Untuk informasi selengkapnya, lihat Sink API Tabel.

Setelah Anda membuat, mengompilasi, dan mengemas kode aplikasi Anda, Anda mengunggah paket kode ke bucket Amazon S3. Anda kemudian membuat Layanan Terkelola untuk aplikasi Apache Flink. Anda meneruskan di lokasi paket kode, topik Amazon MSK sebagai sumber data streaming, dan biasanya lokasi streaming atau file yang menerima data yang diproses dari aplikasi.

Prasyarat

Sebelum memulai tutorial ini, selesaikan dua langkah pertama dari Memulai dengan Amazon Managed Service untuk Apache Flink (API) DataStream :

- Langkah 1: Siapkan Akun AWS dan Buat Pengguna Administrator
- Langkah 2: Siapkan AWS Command Line Interface (AWS CLI)

Untuk memulai, lihat Buat Aplikasi.

Buat dan Jalankan Layanan Terkelola untuk aplikasi Apache Flink

Dalam latihan ini, Anda membuat Layanan Terkelola untuk aplikasi Apache Flink dengan topik MSK Amazon sebagai sumber dan ember Amazon S3 sebagai wastafel.

Bagian ini berisi langkah-langkah berikut.

- Buat Sumber Daya Dependen
- Tulis Catatan Sampel ke Aliran Input
- Unduh dan Periksa Kode Java Streaming Apache Flink
- Kompilasi Kode Aplikasi
- Unggah Kode Java Streaming Apache Flink
- Buat dan Jalankan Managed Service untuk Apache Flink Application
- Langkah Selanjutnya

Buat Sumber Daya Dependen

Sebelum Anda membuat Layanan Terkelola untuk Apache Flink untuk latihan ini, Anda membuat sumber daya dependen berikut:

- Virtual private cloud (VPC) berdasarkan Amazon VPC dan klaster Amazon MSK

- Bucket Amazon S3 untuk menyimpan kode dan output aplikasi (`ka-app-code-<username>`)

Buat VPC dan Klaster Amazon MSK

Untuk membuat klaster MSK VPC dan Amazon untuk mengakses dari aplikasi Managed Service for Apache Flink Anda, ikuti tutorial Memulai Menggunakan Amazon MSK.

Saat menyelesaikan tutorial, perhatikan hal berikut:

- Catat daftar server bootstrap untuk klaster Anda. Anda bisa mendapatkan daftar server bootstrap dengan perintah berikut, mengganti *ClusterArn* dengan Amazon Resource Name (ARN) klaster MSK Anda:

```
aws kafka get-bootstrap-brokers --region us-west-2 --cluster-arn ClusterArn
{...
    "BootstrapBrokerStringTls": "b-2.awskafkatutorialcluste.t79r6y.c4.kafka.us-
west-2.amazonaws.com:9094,b-1.awskafkatutorialcluste.t79r6y.c4.kafka.us-
west-2.amazonaws.com:9094,b-3.awskafkatutorialcluste.t79r6y.c4.kafka.us-
west-2.amazonaws.com:9094"
}
```

- Saat mengikuti langkah-langkah dalam tutorial, pastikan untuk menggunakan Wilayah AWS yang Anda pilih dalam kode, perintah, dan entri konsol Anda.

Buat Bucket Amazon S3

Anda dapat membuat bucket Amazon S3 menggunakan konsol. Untuk petunjuk pembuatan sumber daya ini, lihat topik berikut:

- Bagaimana Cara Membuat Bucket S3? di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon. Beri bucket Amazon S3 nama yang unik secara global dengan menambahkan nama login Anda, seperti **`ka-app-code-<username>`**.

Sumber Daya Lainnya

Saat Anda membuat aplikasi, Managed Service for Apache Flink akan membuat CloudWatch resource Amazon berikut jika belum ada:

- Grup log yang disebut `/AWS/KinesisAnalytics-java/MyApplication`.

- Aliran log yang disebut `kinesis-analytics-log-stream`.

Tulis Catatan Sampel ke Aliran Input

Di bagian ini, Anda menggunakan skrip Python untuk menulis catatan sampel ke topik Amazon MSK untuk diproses aplikasi.

1. Sambungkan ke instans klien yang Anda buat di Langkah 4: Buat Mesin Klien dari tutorial Memulai Menggunakan Amazon MSK.
2. Instal pustaka Python3, Pip, dan Kafka Python:

```
$ sudo yum install python37
$ curl -O https://bootstrap.pypa.io/get-pip.py
$ python3 get-pip.py --user
$ pip install kafka-python
```

3. Buat file bernama `stock.py` dengan isi berikut ini. Ganti nilai BROKERS dengan daftar broker bootstrap Anda yang Anda catat sebelumnya.

```
import datetime
    import json
    import random
    import boto3

    STREAM_NAME = "ExampleInputStream"


    def get_data():
        return {
            'event_time': datetime.datetime.now().isoformat(),
            'ticker': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
            'price': round(random.random() * 100, 2)}


    def generate(stream_name, kinesis_client):
        while True:
            data = get_data()
            print(data)
            kinesis_client.put_record(
                StreamName=stream_name,
                Data=json.dumps(data),
```

```
                PartitionKey="partitionkey")


    if __name__ == '__main__':
        generate(STREAM_NAME, boto3.client('kinesis', region_name='us-west-2'))
```

4. Selanjutnya dalam tutorial ini, Anda menjalankan skrip `stock.py` untuk mengirim data ke aplikasi.

```
$ python3 stock.py
```

Unduh dan Periksa Kode Java Streaming Apache Flink

Kode aplikasi Java untuk contoh ini tersedia dari GitHub.

Untuk mengunduh kode aplikasi Java

1. Kloning repositori jarak jauh menggunakan perintah berikut:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

2. Buka direktori `amazon-kinesis-data-analytics-java-examples/GettingStartedTable` tersebut.

Perhatikan hal tentang kode aplikasi berikut:

- File Project Object Model (pom.xml) berisi informasi tentang konfigurasi dan dependensi aplikasi, termasuk Layanan Terkelola untuk pustaka Apache Flink.
- File `StreamingJob.java` berisi metode `main` yang menentukan fungsionalitas aplikasi.
- Aplikasi ini menggunakan `FlinkKafkaConsumer` untuk membaca dari topik Amazon MSK. Cuplikan berikut membuat objek `FlinkKafkaConsumer`:

```
final FlinkKafkaConsumer<StockRecord> consumer = new
 FlinkKafkaConsumer<StockRecord>(kafkaTopic, new KafkaEventDeserializationSchema(),
 kafkaProps);
```

- Aplikasi Anda membuat konektor sumber dan sink untuk mengakses sumber daya eksternal menggunakan objek `StreamExecutionEnvironment` dan `TableEnvironment`.

- Aplikasi membuat konektor sumber dan sink menggunakan properti aplikasi dinamis, agar Anda dapat menentukan parameter aplikasi Anda (seperti bucket S3 Anda) tanpa mengompilasi ulang kode.

```
//read the parameters from the Managed Service for Apache Flink environment
Map<String, Properties> applicationProperties =
 KinesisAnalyticsRuntime.getApplicationProperties();
Properties flinkProperties = null;

String kafkaTopic = parameter.get("kafka-topic", "AWSKafkaTutorialTopic");
String brokers = parameter.get("brokers", "");
String s3Path = parameter.get("s3Path", "");

if (applicationProperties != null) {
    flinkProperties = applicationProperties.get("FlinkApplicationProperties");
}

if (flinkProperties != null) {
    kafkaTopic = flinkProperties.get("kafka-topic").toString();
    brokers = flinkProperties.get("brokers").toString();
    s3Path = flinkProperties.get("s3Path").toString();
}
```

Untuk informasi selengkapnya tentang properti runtime, lihat Properti Runtime.

> **Note**
>
> Saat membuat aplikasi Anda, kami sangat menyarankan untuk membuat dan menjalankan Layanan Terkelola untuk aplikasi Apache Flink di Wilayah yang sama dengan kluster MSK Amazon. Ini karena konektor Flink Kafka secara default dioptimalkan untuk lingkungan latensi rendah. Jika Anda perlu mengkonsumsi dari cluster Kafka lintas Wilayah, pertimbangkan untuk meningkatkan nilai konfigurasi untuk`receive.buffer.byte`, seperti 2097152. Untuk informasi selengkapnya, lihat Konfigurasi MSK khusus.

Kompilasi Kode Aplikasi

Di bagian ini, Anda menggunakan compiler Apache Maven untuk membuat kode Java untuk aplikasi. Untuk informasi tentang menginstal Apache Maven dan Java Development Kit (JDK), lihat Prasyarat untuk Menyelesaikan Latihan.

Untuk mengompikasi kode aplikasi

1. Untuk menggunakan kode aplikasi Anda, Anda mengompilasi dan mengemasnya ke dalam file JAR. Anda dapat mengompilasi dan mengemas kode Anda dengan salah satu dari dua cara:

   - Gunakan alat Maven baris perintah. Buat file JAR Anda dengan menjalankan perintah berikut di direktori yang berisi file `pom.xml`:

     ```
     mvn package -Dflink.version=1.15.3
     ```

   - Menyiapkan lingkungan pengembangan Anda. Lihat dokumentasi lingkungan pengembangan Anda untuk detail.

     > **Note**
     >
     > Kode sumber yang disediakan bergantung pada pustaka dari Java 11.

   Anda dapat mengunggah paket Anda sebagai file JAR, atau Anda dapat mengompresi paket Anda dan mengunggahnya sebagai file ZIP. Jika Anda membuat aplikasi Anda menggunakan AWS CLI, Anda menentukan tipe konten kode Anda (JAR atau ZIP).

2. Jika ada kesalahan saat mengompilasi, pastikan variabel lingkungan `JAVA_HOME`Anda diatur dengan benar.

Jika aplikasi berhasil mengompilasi, file berikut dibuat:

`target/aws-kinesis-analytics-java-apps-1.0.jar`

Unggah Kode Java Streaming Apache Flink

Di bagian ini, Anda membuat bucket Amazon S3 dan mengunggah kode aplikasi Anda.

Untuk mengunggah kode aplikasi

1. Buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.
2. Pilih Buat bucket.
3. Masukkan **ka-app-code-*<username>*** di bidang Bucket name (Nama bucket). Tambahkan sufiks ke nama bucket, seperti nama pengguna Anda, untuk membuatnya unik secara global. Pilih Next (Selanjutnya).
4. Di langkah Konfigurasi opsi, jangan ubah pengaturan, dan pilih Next (Selanjutnya).
5. Di langkah Atur izin, jangan ubah pengaturan, dan pilih Next (Selanjutnya).
6. Pilih Create bucket (Buat bucket).
7. Di konsol Amazon S3, pilih bucket ka-app-code -, dan pilih Unggah. <username>
8. Di langkah Pilih file, pilih Add files (Tambahkan berkas). Navigasikan ke file `aws-kinesis-analytics-java-apps-1.0.jar` yang Anda buat di langkah sebelumnya. Pilih Berikutnya.
9. Anda tidak perlu mengubah pengaturan apa pun untuk objek, jadi pilih Upload (Unggah).

Kode aplikasi Anda sekarang disimpan di bucket Amazon S3 yang dapat diakses aplikasi Anda.

Buat dan Jalankan Managed Service untuk Apache Flink Application

Ikuti langkah-langkah ini untuk membuat, mengonfigurasi, memperbarui, dan menjalankan aplikasi menggunakan konsol.

Buat Aplikasi

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/flink
2. Pada dashboard Managed Service for Apache Flink, pilih Create Analytics Application.
3. Pada Layanan Terkelola untuk Apache Flink - Buat halaman aplikasi, berikan detail aplikasi sebagai berikut:
   - Untuk Application name (Nama aplikasi), masukkan **`MyApplication`**.
   - Untuk Description (Deskripsi), masukkan **`My java test app`**.
   - Untuk Runtime, pilih Apache Flink.
   - Tinggalkan versi sebagai Apache Flink versi 1.15.2 (Versi yang disarankan).
4. Untuk Access permissions (Izin akses), pilih Create / update IAM role **`kinesis-analytics-MyApplication-us-west-2`** (Buat/perbarui IAM role).

5. Pilih Create application (Buat aplikasi).

> **Note**
>
> Saat membuat aplikasi Managed Service for Apache Flink menggunakan konsol, Anda memiliki opsi untuk membuat peran dan kebijakan IAM untuk aplikasi Anda. Aplikasi Anda menggunakan peran dan kebijakan ini untuk mengakses sumber daya dependen. Sumber daya IAM ini diberi nama menggunakan nama aplikasi dan Wilayah sebagai berikut:
>
> - Kebijakan: `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Peran: `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Edit Kebijakan IAM

Edit kebijakan IAM untuk menambahkan izin agar dapat mengakses bucket Amazon S3.

Untuk mengedit kebijakan IAM agar dapat menambahkan izin bucket S3

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pilih Policies (Kebijakan). Pilih kebijakan **`kinesis-analytics-service-MyApplication-us-west-2`** yang dibuat konsol untuk Anda di bagian sebelumnya.
3. Di halaman Ringkasan, pilih Edit policy (Edit kebijakan). Pilih tab JSON.
4. Tambahkan bagian yang disorot dari contoh kebijakan berikut ke kebijakan.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
        "Sid": "S3",
        "Effect": "Allow",
        "Action": [
            "s3:Abort*",
            "s3:DeleteObject*",
            "s3:GetObject*",
            "s3:GetBucket*",
            "s3:List*",
            "s3:ListBucket",
            "s3:PutObject"
```

```
        ],
        "Resource": [
            "arn:aws:s3:::ka-app-code-<username>",
            "arn:aws:s3:::ka-app-code-<username>/*"
        ]
        },
        {
            "Sid": "DescribeLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
            ]
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
            ]
        }
    ]
}
```

Konfigurasikan Aplikasi

Gunakan prosedur berikut untuk mengonfigurasi aplikasi.

Untuk mengonfigurasi aplikasi

1. Pada MyApplicationhalaman, pilih Konfigurasi.
2. Di halaman Konfigurasikan aplikasi, berikan Code location (Lokasi kode):
   - Untuk Bucket Amazon S3, masukkan **ka-app-code-*<username>***.
   - Untuk Jalur ke objek Amazon S3, masukkan **aws-kinesis-analytics-java-apps-1.0.jar**.
3. Di bawah Akses ke sumber daya aplikasi, untuk Access permissions (Izin akses), pilih Create / update IAM role **kinesis-analytics-MyApplication-us-west-2** (Pilih/perbarui IAM role).
4. Di bawah Properties (Properti), pilih Create group (Buat grup).
5. Masukkan yang berikut ini:

ID Grup	Kunci	Nilai
FlinkApplicationProperties	**kafka-topic**	**AWSKafkaTutorialTopic**
FlinkApplicationProperties	**brokers**	***Your Amazon MSK cluster's Bootstrap Brokers list***
FlinkApplicationProperties	**s3Path**	**ka-app-code- *<username>***
FlinkApplicationProperties	**security.protocol**	**SSL**
FlinkApplicationProperties	**ssl.truststore.location**	**/usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts**

ID Grup	Kunci	Nilai
FlinkApplicationProperties	**ssl.truststore.password**	**changeit**

6. Di bawah Pemantauan, pastikan Memantau tingkat metrik diatur ke Aplikasi.
7. Untuk CloudWatch logging, pilih kotak centang Aktifkan.
8. Di bagian Virtual Private Cloud (VPC), pilih Konfigurasi VPC berdasarkan klaster Amazon MSK. Pilih AWSKafkaTutorialCluster.
9. Pilih Perbarui.

> **Note**
>
> Saat Anda memilih untuk mengaktifkan CloudWatch pencatatan Amazon, Layanan Terkelola untuk Apache Flink membuat grup log dan aliran log untuk Anda. Nama-nama sumber daya ini adalah sebagai berikut:
>
> - Grup log: `/aws/kinesis-analytics/MyApplication`
> - Aliran log: `kinesis-analytics-log-stream`

Jalankan Aplikasi

Gunakan prosedur berikut untuk menjalankan aplikasi.

Untuk menjalankan aplikasi

1. Pada MyApplicationhalaman, pilih Jalankan. Konfirmasikan tindakan.
2. Ketika aplikasi berjalan, refresh halaman. Konsol menunjukkan Grafik aplikasi.
3. Dari klien Amazon EC2 Anda, jalankan skrip Python yang Anda buat sebelumnya untuk menulis catatan ke klaster Amazon MSK untuk diproses aplikasi Anda:

```
$ python3 stock.py
```

Hentikan Aplikasi

Untuk menghentikan aplikasi, pada MyApplicationhalaman, pilih Berhenti. Konfirmasikan tindakan.

Langkah Selanjutnya

Pembersihan Sumber Daya AWS

Pembersihan Sumber Daya AWS

Bagian ini mencakup prosedur untuk membersihkan sumber daya AWS yang dibuat dalam tutorial Memulai (API Tabel).

Topik ini berisi bagian-bagian berikut.

- Hapus Layanan Terkelola Anda untuk Aplikasi Apache Flink
- Hapus Klaster Amazon MSK Anda
- Hapus VPC Anda
- Hapus Objek dan Bucket Amazon S3 Anda
- Hapus Sumber Daya IAM Anda
- Hapus CloudWatch Sumber Daya Anda
- Langkah Selanjutnya

Hapus Layanan Terkelola Anda untuk Aplikasi Apache Flink

Gunakan prosedur berikut untuk menghapus aplikasi.

Untuk menghapus aplikasi

1. Buka konsol Kinesis di https://console.aws.amazon.com/kinesis.
2. Di panel Managed Service for Apache Flink, pilih. MyApplication
3. Di halaman aplikasi, pilih Delete (Hapus), lalu konfirmasikan penghapusan.

Hapus Klaster Amazon MSK Anda

Untuk menghapus klaster Amazon MSK Anda, ikuti Langkah 8: Hapus Klaster Amazon MSK di Panduan Developer Amazon Managed Streaming for Apache Kafka.

Hapus VPC Anda

Untuk menghapus Amazon VPC Anda, lakukan hal berikut:

- Buka konsol Amazon VPC.
- Pilih VPC Anda.
- Untuk Tindakan, pilih Delete VPC (Hapus VPC).

Hapus Objek dan Bucket Amazon S3 Anda

Gunakan prosedur berikut untuk menghapus objek dan bucket S3 Anda.

Untuk menghapus objek dan bucket S3 Anda

1. Buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.
2. Pilih ka-app-code- <username>ember.
3. Pilih Delete (Hapus), lalu masukkan nama bucket untuk mengonfirmasi penghapusan.

Hapus Sumber Daya IAM Anda

Gunakan prosedur berikut untuk menghapus sumber daya IAM Anda.

Untuk menghapus sumber daya IAM Anda

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di bilah navigasi, pilih Policies (Kebijakan).
3. Di kontrol filter, masukkan kinesis.
4. Pilih kebijakan kinesis-analytics-service- MyApplication -us-west-2.
5. Pilih Policy Actions (Tindakan Kebijakan), lalu pilih Delete (Hapus).
6. Di bilah navigasi, pilih Roles (Peran).
7. Pilih peran kinesis-analytics- -us-west-2. MyApplication
8. Pilih Delete role (Hapus peran), lalu konfirmasi penghapusan.

Hapus CloudWatch Sumber Daya Anda

Gunakan prosedur berikut untuk menghapus CloudWatch sumber daya Anda.

Untuk menghapus CloudWatch sumber daya Anda

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di bilah navigasi, pilih Logs.
3. Pilih grup log MyApplication/aws/kinesis-analytics/.
4. Pilih Delete Log Group (Hapus Grup Log), lalu konfirmasi penghapusan.

Langkah Selanjutnya

Langkah Berikutnya

Langkah Berikutnya

Sekarang setelah Anda membuat dan menjalankan Managed Service untuk aplikasi Apache Flink yang menggunakan Table API, lihat Langkah 5: Langkah selanjutnya di file. Memulai dengan Amazon Managed Service untuk Apache Flink (API) DataStream

Memulai dengan Amazon Managed Service untuk Apache Flink untuk Python

Bagian ini memperkenalkan Anda pada konsep dasar Managed Service untuk Apache Flink menggunakan Python dan Table API. Ini menjelaskan opsi yang tersedia untuk membuat dan menguji aplikasi Anda. Ini juga memberikan petunjuk untuk menginstal alat yang diperlukan untuk menyelesaikan tutorial dalam panduan ini dan untuk membuat aplikasi pertama Anda.

Topik

> **Note**
>
> Jika Anda mengembangkan aplikasi Python Flink pada Mac baru dengan chip Apple Silicon, Anda mungkin mengalami beberapa masalah yang diketahui dengan dependensi Python 1,15. PyFlink Dalam hal ini kami sarankan menjalankan interpreter Python di Docker. Untuk step-by-step petunjuk, lihat PyFlink 1,15 pengembangan di Apple Silicon Mac.

Memulai Pyflink - Penerjemah Python untuk Apache | Amazon Web Services

Sebelum Anda mulai, kami mendorong Anda untuk menonton video berikut:

Memulai Pyflink - Penerjemah Python untuk Apache | Amazon Web Services

Komponen Layanan Terkelola untuk Aplikasi Apache Flink

Untuk memproses data, Managed Service untuk aplikasi Apache Flink Anda menggunakan aplikasi Python yang memproses input dan menghasilkan output menggunakan runtime Apache Flink.

Managed Service untuk aplikasi Apache Flink memiliki komponen-komponen berikut:

- Properti runtime: Anda dapat menggunakan properti runtime untuk mengonfigurasi aplikasi Anda tanpa mengompilasi ulang kode aplikasi Anda.
- Sumber Tabel: Aplikasi mengonsumsi data menggunakan sumber. Konektor sumber membaca data dari Kinesis data stream, topik Amazon MSK, atau yang serupa. Untuk informasi selengkapnya, lihat Sumber API Tabel.
- Fungsi: Aplikasi memproses data menggunakan satu atau beberapa fungsi. Fungsi dapat mengubah, memperkaya, atau menggabungkan data.
- Sink: Aplikasi menghasilkan data ke sumber eksternal menggunakan sink. Konektor sink menulis data ke aliran data Kinesis, aliran Firehose Data Kinesis Data Firehose Kinesis, topik MSK Amazon, bucket Amazon S3, dan sebagainya. Untuk informasi selengkapnya, lihat Sink API Tabel.

Setelah Anda membuat dan mengemas kode aplikasi Anda, Anda mengunggah paket kode ke bucket Amazon S3. Anda kemudian membuat Layanan Terkelola untuk aplikasi Apache Flink. Anda meneruskan di lokasi paket kode, sumber data streaming, dan biasanya lokasi streaming atau file yang menerima data yang diproses dari aplikasi.

Prasyarat

Sebelum memulai tutorial ini, selesaikan dua langkah pertama dari Memulai dengan Amazon Managed Service untuk Apache Flink (API) DataStream :

- Langkah 1: Siapkan Akun AWS dan Buat Pengguna Administrator
- Langkah 2: Siapkan AWS Command Line Interface (AWS CLI)

Untuk memulai, lihat Buat Aplikasi.

Buat dan Jalankan Layanan Terkelola untuk Apache Flink untuk Aplikasi Python

Dalam latihan ini, Anda membuat Layanan Terkelola untuk aplikasi Apache Flink untuk aplikasi Python dengan aliran Kinesis sebagai sumber dan wastafel.

Bagian ini berisi langkah-langkah berikut.

Buat Sumber Daya Dependen

Sebelum Anda membuat Layanan Terkelola untuk Apache Flink untuk latihan ini, Anda membuat sumber daya dependen berikut:

- Dua aliran Kinesis untuk input dan output.
- Bucket Amazon S3 untuk menyimpan kode dan output aplikasi (`ka-app-code-<username>`)

Buat Dua Aliran Kinesis

Sebelum Anda membuat Layanan Terkelola untuk aplikasi Apache Flink untuk latihan ini, buat dua aliran data Kinesis (dan). `ExampleInputStream` `ExampleOutputStream` Aplikasi Anda menggunakan aliran ini untuk sumber aplikasi dan aliran tujuan.

Anda dapat membuat aliran ini menggunakan konsol Amazon Kinesis atau perintah AWS CLI berikut. Untuk instruksi konsol, lihat Membuat dan Memperbarui Aliran Data di Panduan Developer Amazon Kinesis Data Streams.

Untuk membuat aliran data AWS CLI

1. Untuk membuat aliran pertama (`ExampleInputStream`), gunakan perintah `create-stream` AWS CLI Amazon Kinesis berikut.

```
$ aws kinesis create-stream \
--stream-name ExampleInputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

2. Untuk membuat aliran kedua yang digunakan aplikasi untuk menulis output, jalankan perintah yang sama, yang mengubah nama aliran menjadi `ExampleOutputStream`.

```
$ aws kinesis create-stream \
--stream-name ExampleOutputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

Buat Bucket Amazon S3

Anda dapat membuat bucket Amazon S3 menggunakan konsol. Untuk petunjuk pembuatan sumber daya ini, lihat topik berikut:

- Bagaimana Cara Membuat Bucket S3? di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon. Beri bucket Amazon S3 nama yang unik secara global dengan menambahkan nama login Anda, seperti **`ka-app-code-<username>`**.

Sumber Daya Lainnya

Saat Anda membuat aplikasi, Managed Service for Apache Flink akan membuat CloudWatch resource Amazon berikut jika belum ada:

- Grup log yang disebut `/AWS/KinesisAnalytics-java/MyApplication`.
- Aliran log yang disebut `kinesis-analytics-log-stream`.

Tulis Catatan Sampel ke Aliran Input

Di bagian ini, Anda menggunakan script Python untuk menulis catatan sampel ke aliran untuk diproses aplikasi.

> **Note**
>
> Bagian ini memerlukan AWS SDK for Python (Boto).

> **Note**
>
> Skrip Python di bagian ini menggunakan AWS CLI. Anda harus mengonfigurasi AWS CLI untuk menggunakan kredensial akun dan wilayah default Anda. Untuk mengonfigurasi AWS CLI Anda, masukkan berikut ini:
>
> ```
> aws configure
> ```

1. Buat file bernama `stock.py` dengan konten berikut:

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        'event_time': datetime.datetime.now().isoformat(),
        'ticker': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'price': round(random.random() * 100, 2)}


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(data),
            PartitionKey="partitionkey")


if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis', region_name='us-west-2'))
```

2. Jalankan skrip `stock.py`.

```
$ python stock.py
```

Biarkan skrip tetap berjalan saat menyelesaikan sisa tutorial.

Buat dan Periksa Kode Python Streaming Apache Flink

Kode aplikasi Python untuk contoh ini tersedia dari. GitHub Untuk mengunduh kode aplikasi, lakukan hal berikut:

1. Instal klien Git jika Anda belum menginstalnya. Untuk informasi selengkapnya, lihat Menginstal Git.
2. Klon repositori jarak jauh dengan perintah berikut:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Buka direktori `amazon-kinesis-data-analytics-java-examples/python/GettingStarted` tersebut.

Kode aplikasi terletak di file `getting_started.py`. Perhatikan hal tentang kode aplikasi berikut:

- Aplikasi menggunakan sumber tabel Kinesis untuk membaca dari aliran sumber. Cuplikan berikut memanggil fungsi `create_table` untuk membuat sumber tabel Kinesis:

```
table_env.execute_sql(
        create_table(output_table_name, output_stream, output_region)
```

Fungsi `create_table` menggunakan perintah SQL untuk membuat tabel yang didukung oleh sumber streaming:

```
def create_table(table_name, stream_name, region, stream_initpos = None):
    init_pos = "\n'scan.stream.initpos' = '{0}',".format(stream_initpos) if
 stream_initpos is not None else ''

    return """ CREATE TABLE {0} (
                ticker VARCHAR(6),
                price DOUBLE,
                event_time TIMESTAMP(3),
                WATERMARK FOR event_time AS event_time - INTERVAL '5' SECOND
```

```
              )
              PARTITIONED BY (ticker)
              WITH (
                'connector' = 'kinesis',
                'stream' = '{1}',
                'aws.region' = '{2}',{3}
                'format' = 'json',
                'json.timestamp-format.standard' = 'ISO-8601'
              ) """.format(table_name, stream_name, region, init_pos)
}
```

- Aplikasi ini membuat dua tabel, lalu menulis konten dari satu tabel ke yang lainnya.

```
    # 2. Creates a source table from a Kinesis Data Stream
    table_env.execute_sql(
        create_table(input_table_name, input_stream, input_region)
    )

    # 3. Creates a sink table writing to a Kinesis Data Stream
    table_env.execute_sql(
        create_table(output_table_name, output_stream, output_region, stream_initpos)
    )

    # 4. Inserts the source table data into the sink table
    table_result = table_env.execute_sql("INSERT INTO {0} SELECT * FROM {1}"
                         .format(output_table_name, input_table_name))
```

- Aplikasi ini menggunakan konektor Flink, dari file flink- sql-connector-kinesis _2.12/1.15.2.

Menambahkan dependensi pihak ketiga ke aplikasi Python

Saat menggunakan paket python pihak ketiga (seperti boto3), Anda perlu menambahkan dependensi transitifnya dan properti yang diperlukan untuk menargetkan dependensi ini. Pada tingkat tinggi, untuk PyPi dependensi, Anda dapat menyalin file dan folder yang terletak di dalam `site-packages` folder lingkungan python Anda untuk membuat struktur direktori seperti di bawah ini:

```
PythonPackages
#   README.md
#   python-packages.py
#
```

```
####my_deps
       ####boto3
       #   #   session.py
       #   #   utils.py
       #   #   ...
       #
       ####botocore
       #   #   args.py
       #   #   auth.py
       #    ...
       ####mynonpypimodule
       #   #   mymodulefile1.py
       #   #   mymodulefile2.py
            ...
####lib
#    # flink-sql-connector-kinesis-1.15.2.jar
#    # ...
...
```

Untuk menambahkan boto3 sebagai dependensi pihak ketiga:

1. Buat lingkungan Python mandiri (conda atau serupa) di mesin lokal Anda dengan dependensi yang diperlukan.
2. Perhatikan daftar awal paket di `site_packages` folder lingkungan itu.
3. `pip-install`semua dependensi yang diperlukan untuk aplikasi Anda.
4. Perhatikan paket yang ditambahkan ke `site_packages` folder setelah langkah 3 di atas. Ini adalah folder yang perlu Anda sertakan dalam paket Anda (di bawah `my_deps` folder), diatur seperti yang ditunjukkan di atas. Ini akan memungkinkan Anda untuk menangkap perbedaan paket antara langkah 2 dan 3 untuk mengidentifikasi dependensi paket yang tepat untuk aplikasi Anda.
5. Menyediakan `my_deps/` sebagai argumen untuk `pyFiles` properti dalam grup `kinesis.analytics.flink.run.options` properti seperti yang dijelaskan di bawah ini untuk `jarfiles` properti. Flink juga memungkinkan Anda untuk menentukan dependensi Python menggunakan fungsi add_python_file, tetapi penting untuk diingat bahwa Anda hanya perlu menentukan satu atau yang lain - bukan keduanya.

> **Note**
>
> Anda tidak perlu memberi nama folder`my_deps`. Bagian yang penting adalah mendaftarkan dependensi menggunakan salah satu ataupy`Files`. `add_python_file` Contoh dapat ditemukan di Cara menggunakan boto3 dalam PyfLink.

Unggah Kode Python Streaming Apache Flink

Di bagian ini, Anda membuat bucket Amazon S3 dan mengunggah kode aplikasi Anda.

Untuk mengunggah kode aplikasi menggunakan konsol:

1. Gunakan aplikasi kompresi pilihan Anda untuk mengompres file `getting-started.py` dan https://mvnrepository.com/artifact/org.apache.flink/flink - sql-connector-kinesis _2.12/1.15.2. Beri nama arsip `myapp.zip`. Jika Anda menyertakan folder luar dalam arsip Anda, Anda harus menyertakan ini di jalur dengan kode di file konfigurasi Anda:`GettingStarted/getting-started.py`.
2. Buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.
3. Pilih Buat bucket.
4. Masukkan **`ka-app-code-<username>`** di bidang Bucket name (Nama bucket). Tambahkan sufiks ke nama bucket, seperti nama pengguna Anda, untuk membuatnya unik secara global. Pilih Next (Selanjutnya).
5. Di langkah Konfigurasi opsi, jangan ubah pengaturan, dan pilih Next (Selanjutnya).
6. Di langkah Atur izin, jangan ubah pengaturan, dan pilih Next (Selanjutnya).
7. Pilih Create bucket (Buat bucket).
8. Di konsol Amazon S3, pilih bucket ka-app-code -, dan pilih Unggah. <username>
9. Di langkah Pilih file, pilih Add files (Tambahkan berkas). Navigasikan ke file `myapp.zip` yang Anda buat di langkah sebelumnya. Pilih Berikutnya.
10. Anda tidak perlu mengubah pengaturan apa pun untuk objek, jadi pilih Upload (Unggah).

Untuk mengunggah kode aplikasi menggunakanAWS CLI:

> **Note**
>
> Jangan gunakan fitur kompres di Finder (macOS) atau Windows Explorer (Windows) untuk membuat arsip. `myapp.zip` Hal ini dapat mengakibatkan kode aplikasi tidak valid.

1. Gunakan aplikasi kompresi pilihan Anda untuk mengompres file `streaming-file-sink.py` dan https://mvnrepository.com/artifact/org.apache.flink/flink - sql-connector-kinesis _2.12/1.15.2.

   > **Note**
   >
   > Jangan gunakan fitur kompres di Finder (macOS) atau Windows Explorer (Windows) untuk membuat arsip myapp.zip. Hal ini dapat mengakibatkan kode aplikasi tidak valid.

2. Gunakan aplikasi kompresi pilihan Anda untuk mengompres file `getting-started.py` dan https://mvnrepository.com/artifact/org.apache.flink/ flink-sql-connector-kinesis /1.15.2. Beri nama arsip `myapp.zip`. Jika Anda menyertakan folder luar dalam arsip Anda, Anda harus menyertakan ini di jalur dengan kode di file konfigurasi Anda:`GettingStarted/getting-started.py`.
3. Jalankan perintah berikut:

   ```
   $ aws s3 --region aws region cp myapp.zip s3://ka-app-code-<username>
   ```

Kode aplikasi Anda sekarang disimpan di bucket Amazon S3 yang dapat diakses aplikasi Anda.

Buat dan Jalankan Managed Service untuk Apache Flink Application

Ikuti langkah-langkah ini untuk membuat, mengonfigurasi, memperbarui, dan menjalankan aplikasi menggunakan konsol.

Buat Aplikasi

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/flink
2. Pada dashboard Managed Service for Apache Flink, pilih Create Analytics Application.
3. Pada Layanan Terkelola untuk Apache Flink - Buat halaman aplikasi, berikan detail aplikasi sebagai berikut:

- Untuk Application name (Nama aplikasi), masukkan **MyApplication**.
- Untuk Description (Deskripsi), masukkan **My java test app**.
- Untuk Runtime, pilih Apache Flink.
- Tinggalkan versi sebagai Apache Flink versi 1.15.2 (Versi yang disarankan).

4. Untuk Access permissions (Izin akses), pilih Create / update IAM role **kinesis-analytics-MyApplication-us-west-2** (Buat/perbarui IAM role).
5. Pilih Create application (Buat aplikasi).

> **Note**
>
> Saat membuat aplikasi Managed Service for Apache Flink menggunakan konsol, Anda memiliki opsi untuk membuat peran dan kebijakan IAM untuk aplikasi Anda. Aplikasi Anda menggunakan peran dan kebijakan ini untuk mengakses sumber daya dependen. Sumber daya IAM ini diberi nama menggunakan nama aplikasi dan Wilayah sebagai berikut:
>
> - Kebijakan: `kinesis-analytics-service-MyApplication-us-west-2`
> - Peran: `kinesisanalytics-MyApplication-us-west-2`

Konfigurasikan Aplikasi

Gunakan prosedur berikut untuk mengonfigurasi aplikasi.

Untuk mengonfigurasi aplikasi

1. Pada MyApplicationhalaman, pilih Konfigurasi.
2. Di halaman Konfigurasikan aplikasi, berikan Code location (Lokasi kode):
   - Untuk Bucket Amazon S3, masukkan **ka-app-code-*<username>***.
   - Untuk Jalur ke objek Amazon S3, masukkan **myapp.zip**.
3. Di bawah Akses ke sumber daya aplikasi, untuk Access permissions (Izin akses), pilih Create / update IAM role **kinesis-analytics-MyApplication-us-west-2** (Pilih/perbarui IAM role).
4. Di bawah Properties (Properti), pilih Add group (Tambahkan grup).
5. Masukkan yang berikut ini:

ID Grup	Kunci	Nilai
consumer.config.0	**input.stream.name**	**ExampleInputStream**
consumer.config.0	**aws.region**	**us-west-2**
consumer.config.0	**scan.stream.initpos**	**LATEST**

Pilih Simpan.

6. Di bawah Properties (Properti), pilih Add group (Tambahkan grup) lagi.
7. Masukkan yang berikut ini:

ID Grup	Kunci	Nilai
producer.config.0	**output.stream.name**	**ExampleOutputStream**
producer.config.0	**aws.region**	**us-west-2**
producer.config.0	**shard.count**	**1**

8. Di bawah Properties (Properti), pilih Add group (Tambahkan grup) lagi. Untuk ID Grup, masukkan **kinesis.analytics.flink.run.options**. Grup properti khusus ini memberi tahu aplikasi Anda tempat untuk menemukan sumber daya kodenya. Untuk informasi selengkapnya, lihat Menentukan File Kode Anda.
9. Masukkan yang berikut ini:

ID Grup	Kunci	Nilai
kinesis.analytics.flink.run.options	**python**	**getting-started.py**
kinesis.analytics.flink.run.options	**jarfile**	**flink-sql-connector-kinesis-1.15.2.jar**

10. Di bawah Pemantauan, pastikan Memantau tingkat metrik diatur ke Aplikasi.

11. Untuk CloudWatch logging, pilih kotak centang Aktifkan.
12. Pilih Perbarui.

> **Note**
>
> Saat Anda memilih untuk mengaktifkan CloudWatch pencatatan Amazon, Layanan Terkelola untuk Apache Flink membuat grup log dan aliran log untuk Anda. Nama-nama sumber daya ini adalah sebagai berikut:
>
> - Grup log: `/aws/kinesis-analytics/MyApplication`
> - Aliran log: `kinesis-analytics-log-stream`

Edit Kebijakan IAM

Edit kebijakan IAM untuk menambahkan izin agar dapat mengakses bucket Amazon S3.

Untuk mengedit kebijakan IAM agar dapat menambahkan izin bucket S3

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pilih Policies (Kebijakan). Pilih kebijakan **`kinesis-analytics-service-MyApplication-us-west-2`** yang dibuat konsol untuk Anda di bagian sebelumnya.
3. Di halaman Ringkasan, pilih Edit policy (Edit kebijakan). Pilih tab JSON.
4. Tambahkan bagian yang disorot dari contoh kebijakan berikut ke kebijakan. Ganti ID akun sampel (*012345678901*) dengan ID akun Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ka-app-code-username/myapp.zip"
            ]
```

```
        },
        {
            "Sid": "DescribeLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
            ]
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
```

```
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

Jalankan Aplikasi

Grafik pekerjaan Flink dapat dilihat dengan menjalankan aplikasi, membuka dasbor Apache Flink, dan memilih pekerjaan Flink yang diinginkan.

Hentikan Aplikasi

Untuk menghentikan aplikasi, pada MyApplicationhalaman, pilih Berhenti. Konfirmasikan tindakan.

Langkah Selanjutnya

Pembersihan Sumber Daya AWS

Pembersihan Sumber Daya AWS

Bagian ini mencakup prosedur untuk membersihkan sumber daya AWS yang dibuat dalam tutorial Memulai (Python).

Topik ini berisi bagian-bagian berikut.

- Hapus Layanan Terkelola Anda untuk Aplikasi Apache Flink
- Hapus Kinesis Data Streams Anda
- Hapus Objek dan Bucket Amazon S3 Anda
- Hapus Sumber Daya IAM Anda
- Hapus CloudWatch Sumber Daya Anda

Hapus Layanan Terkelola Anda untuk Aplikasi Apache Flink

Gunakan prosedur berikut untuk menghapus aplikasi.

Untuk menghapus aplikasi

1. Buka konsol Kinesis di https://console.aws.amazon.com/kinesis.

2. Di panel Managed Service for Apache Flink, pilih. MyApplication
3. Di halaman aplikasi, pilih Delete (Hapus), lalu konfirmasikan penghapusan.

Hapus Kinesis Data Streams Anda

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/flink
2. Di panel Kinesis Data Streams, pilih. ExampleInputStream
3. Di ExampleInputStreamhalaman, pilih Hapus Stream Kinesis dan kemudian konfirmasikan penghapusan.
4. Di halaman Kinesis streams, pilih, pilih Tindakan ExampleOutputStream, pilih Hapus, lalu konfirmasikan penghapusan.

Hapus Objek dan Bucket Amazon S3 Anda

Gunakan prosedur berikut untuk menghapus objek dan bucket S3 Anda.

Untuk menghapus objek dan bucket S3 Anda

1. Buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.
2. Pilih ka-app-code- <username>ember.
3. Pilih Delete (Hapus), lalu masukkan nama bucket untuk mengonfirmasi penghapusan.

Hapus Sumber Daya IAM Anda

Gunakan prosedur berikut untuk menghapus sumber daya IAM Anda.

Untuk menghapus sumber daya IAM Anda

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di bilah navigasi, pilih Policies (Kebijakan).
3. Di kontrol filter, masukkan kinesis.
4. Pilih kebijakan kinesis-analytics-service- MyApplication -us-west-2.
5. Pilih Policy Actions (Tindakan Kebijakan), lalu pilih Delete (Hapus).
6. Di bilah navigasi, pilih Roles (Peran).
7. Pilih peran kinesis-analytics- -us-west-2. MyApplication

8. Pilih Delete role (Hapus peran), lalu konfirmasi penghapusan.

Hapus CloudWatch Sumber Daya Anda

Gunakan prosedur berikut untuk menghapus CloudWatch sumber daya Anda.

Untuk menghapus CloudWatch sumber daya Anda

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di bilah navigasi, pilih Logs.
3. Pilih grup log MyApplication/aws/kinesis-analytics/.
4. Pilih Delete Log Group (Hapus Grup Log), lalu konfirmasi penghapusan.

Getting Started (Memulai)

> **Note**
>
> Mulai dari versi 1.15 Flink gratis Scala. Aplikasi sekarang dapat menggunakan Java API dari versi Scala apa pun. Flink masih menggunakan Scala di beberapa komponen kunci secara internal tetapi tidak mengekspos Scala ke dalam classloader kode pengguna. Karena itu, pengguna perlu menambahkan dependensi Scala ke dalam arsip jar mereka.
> Untuk informasi selengkapnya tentang perubahan Scala di Flink 1.15, lihat Scala Free in One Fifteen.

Dalam latihan ini, Anda membuat Layanan Terkelola untuk aplikasi Apache Flink untuk Scala dengan aliran Kinesis sebagai sumber dan wastafel.

Topik ini berisi bagian-bagian berikut:

- Buat Sumber Daya Dependen
- Tulis Catatan Sampel ke Aliran Input
- Unduh dan Periksa Kode Aplikasi
- Kompilasi dan unggah kode aplikasi
- Buat dan jalankan Aplikasi (konsol)
- Membuat dan menjalankan aplikasi (CLI)
- Pembersihan Sumber Daya AWS

Buat Sumber Daya Dependen

Sebelum Anda membuat Layanan Terkelola untuk aplikasi Apache Flink untuk latihan ini, Anda membuat sumber daya dependen berikut:

- Dua aliran Kinesis untuk input dan output.
- Bucket Amazon S3 untuk menyimpan kode aplikasi (`ka-app-code-<username>`)

Anda dapat membuat aliran Kinesis dan bucket Amazon S3 menggunakan konsol. Untuk petunjuk membuat sumber daya ini, lihat topik berikut:

- Membuat dan Memperbarui Aliran Data di Panduan Developer Amazon Kinesis Data Streams. Beri nama aliran data **`ExampleInputStream`** dan **`ExampleOutputStream`** Anda.

  Untuk membuat aliran data AWS CLI

  - Untuk membuat stream (`ExampleInputStream`) pertama, gunakan perintah Amazon Kinesis AWS CLI create-stream berikut.

    ```
    aws kinesis create-stream \
        --stream-name ExampleInputStream \
        --shard-count 1 \
        --region us-west-2 \
        --profile adminuser
    ```

  - Untuk membuat aliran kedua yang digunakan aplikasi untuk menulis output, jalankan perintah yang sama, yang mengubah nama aliran menjadi `ExampleOutputStream`.

    ```
    aws kinesis create-stream \
        --stream-name ExampleOutputStream \
        --shard-count 1 \
        --region us-west-2 \
        --profile adminuser
    ```

- Bagaimana Cara Membuat Bucket S3? di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon. Beri bucket Amazon S3 nama yang unik secara global dengan menambahkan nama login Anda, seperti **`ka-app-code-<username>`**.

Sumber Daya Lainnya

Saat Anda membuat aplikasi, Managed Service for Apache Flink akan membuat CloudWatch resource Amazon berikut jika belum ada:

- Sebuah grup log yang disebut `/AWS/KinesisAnalytics-java/MyApplication`
- Aliran log yang disebut `kinesis-analytics-log-stream`

Tulis Catatan Sampel ke Aliran Input

Di bagian ini, Anda menggunakan script Python untuk menulis catatan sampel ke aliran untuk diproses aplikasi.

> **Note**
>
> Bagian ini memerlukan AWS SDK for Python (Boto).

> **Note**
>
> Skrip Python di bagian ini menggunakan AWS CLI. Anda harus mengonfigurasi AWS CLI untuk menggunakan kredensial akun dan wilayah default Anda. Untuk mengonfigurasi AWS CLI Anda, masukkan berikut ini:
>
> ```
> aws configure
> ```

1. Buat file bernama `stock.py` dengan konten berikut:

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        'event_time': datetime.datetime.now().isoformat(),
        'ticker': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'price': round(random.random() * 100, 2)}


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(data),
            PartitionKey="partitionkey")
```

```
if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis', region_name='us-west-2'))
```

2. Jalankan skrip `stock.py`.

```
$ python stock.py
```

Biarkan skrip tetap berjalan saat menyelesaikan sisa tutorial.

Unduh dan Periksa Kode Aplikasi

Kode aplikasi Python untuk contoh ini tersedia dari. GitHub Untuk mengunduh kode aplikasi, lakukan hal berikut:

1. Instal klien Git jika Anda belum menginstalnya. Untuk informasi selengkapnya, lihat Menginstal Git.
2. Klon repositori jarak jauh dengan perintah berikut:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Buka direktori `amazon-kinesis-data-analytics-java-examples/scala/GettingStarted` tersebut.

Perhatikan hal tentang kode aplikasi berikut:

- `build.sbt`File berisi informasi tentang konfigurasi dan dependensi aplikasi, termasuk Layanan Terkelola untuk pustaka Apache Flink.
- `BasicStreamingJob.scala`File berisi metode utama yang mendefinisikan fungsionalitas aplikasi.
- Aplikasi menggunakan sumber Kinesis untuk membaca dari aliran sumber. Cuplikan berikut ini membuat sumber Kinesis:

```
private def createSource: FlinkKinesisConsumer[String] = {
  val applicationProperties = KinesisAnalyticsRuntime.getApplicationProperties
  val inputProperties = applicationProperties.get("ConsumerConfigProperties")

  new FlinkKinesisConsumer[String](inputProperties.getProperty(streamNameKey,
 defaultInputStreamName),
```

```
    new SimpleStringSchema, inputProperties)
}
```

Aplikasi ini juga menggunakan sink Kinesis untuk menulis ke dalam aliran hasil. Cuplikan berikut membuat sink Kinesis:

```
private def createSink: KinesisStreamsSink[String] = {
  val applicationProperties = KinesisAnalyticsRuntime.getApplicationProperties
  val outputProperties = applicationProperties.get("ProducerConfigProperties")

  KinesisStreamsSink.builder[String]
    .setKinesisClientProperties(outputProperties)
    .setSerializationSchema(new SimpleStringSchema)
    .setStreamName(outputProperties.getProperty(streamNameKey,
 defaultOutputStreamName))
    .setPartitionKeyGenerator((element: String) => String.valueOf(element.hashCode))
    .build
}
```

- Aplikasi ini membuat konektor sumber dan wastafel untuk mengakses sumber daya eksternal menggunakan StreamExecutionEnvironment objek.
- Aplikasi ini membuat konektor sumber dan wastafel menggunakan properti aplikasi dinamis. Properti aplikasi runtime dibaca untuk mengkonfigurasi konektor. Untuk informasi selengkapnya tentang properti runtime, lihat Properti Runtime.

Kompilasi dan unggah kode aplikasi

Di bagian ini, Anda mengkompilasi dan mengunggah kode aplikasi Anda ke bucket Amazon S3 yang Anda buat di Buat Sumber Daya Dependen bagian tersebut.

Kompilasi Kode Aplikasi

Di bagian ini, Anda menggunakan alat build SBT untuk membangun kode Scala untuk aplikasi. Untuk menginstal SBT, lihat Menginstal sbt dengan pengaturan cs. Anda juga perlu menginstal Java Development Kit (JDK). Lihat Prasyarat untuk Menyelesaikan Latihan.

1. Untuk menggunakan kode aplikasi Anda, Anda mengompilasi dan mengemasnya ke dalam file JAR. Anda dapat mengkompilasi dan mengemas kode Anda dengan SBT:

```
sbt assembly
```

2. Jika aplikasi berhasil mengompilasi, file berikut dibuat:

```
target/scala-3.2.0/getting-started-scala-1.0.jar
```

Unggah Kode Scala Streaming Apache Flink

Di bagian ini, Anda membuat bucket Amazon S3 dan mengunggah kode aplikasi Anda.

1. Buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.
2. Pilih Buat ember
3. Masukkan `ka-app-code-<username>` di bidang Bucket name (Nama bucket). Tambahkan sufiks ke nama bucket, seperti nama pengguna Anda, untuk membuatnya unik secara global. Pilih Next (Selanjutnya).
4. Di opsi Konfigurasi, pertahankan pengaturan apa adanya, dan pilih Berikutnya.
5. Di Setel izin, pertahankan pengaturan apa adanya, dan pilih Berikutnya.
6. Pilih Create bucket (Buat bucket).
7. Pilih `ka-app-code-<username>` bucket, lalu pilih Unggah.
8. Di langkah Pilih file, pilih Add files (Tambahkan berkas). Navigasikan ke file `getting-started-scala-1.0.jar` yang Anda buat di langkah sebelumnya.
9. Anda tidak perlu mengubah pengaturan apa pun untuk objek, jadi pilih Upload (Unggah).

Kode aplikasi Anda sekarang disimpan di bucket Amazon S3 yang dapat diakses aplikasi Anda.

Buat dan jalankan Aplikasi (konsol)

Ikuti langkah-langkah ini untuk membuat, mengonfigurasi, memperbarui, dan menjalankan aplikasi menggunakan konsol.

Buat Aplikasi

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/flink
2. Pada dashboard Managed Service for Apache Flink, pilih Create Analytics Application.

3. Pada Layanan Terkelola untuk Apache Flink - Buat halaman aplikasi, berikan detail aplikasi sebagai berikut:
   - Untuk Application name (Nama aplikasi), masukkan **`MyApplication`**.
   - Untuk Description (Deskripsi), masukkan **`My scala test app`**.
   - Untuk Runtime, pilih Apache Flink.
   - Tinggalkan versi sebagai Apache Flink versi 1.15.2 (Versi yang disarankan).
4. Untuk Access permissions (Izin akses), pilih Create / update IAM role **`kinesis-analytics-MyApplication-us-west-2`** (Buat/perbarui IAM role).
5. Pilih Create application (Buat aplikasi).

> **Note**
>
> Saat membuat aplikasi Managed Service for Apache Flink menggunakan konsol, Anda memiliki opsi untuk membuat peran dan kebijakan IAM untuk aplikasi Anda. Aplikasi Anda menggunakan peran dan kebijakan ini untuk mengakses sumber daya dependen. Sumber daya IAM ini diberi nama menggunakan nama aplikasi dan Wilayah sebagai berikut:
>
> - Kebijakan: `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Peran: `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Konfigurasikan Aplikasi

Gunakan prosedur berikut untuk mengonfigurasi aplikasi.

Untuk mengonfigurasi aplikasi

1. Pada MyApplicationhalaman, pilih Konfigurasi.
2. Di halaman Konfigurasikan aplikasi, berikan Code location (Lokasi kode):
   - Untuk Bucket Amazon S3, masukkan **`ka-app-code-`*`<username>`***.
   - Untuk Jalur ke objek Amazon S3, masukkan **`getting-started-scala-1.0.jar.`**.
3. Di bawah Akses ke sumber daya aplikasi, untuk Access permissions (Izin akses), pilih Create / update IAM role **`kinesis-analytics-MyApplication-us-west-2`** (Pilih/perbarui IAM role).

4. Di bawah Properties (Properti), pilih Add group (Tambahkan grup).
5. Masukkan yang berikut ini:

ID Grup	Kunci	Nilai
ConsumerConfigProperties	**input.stream.name**	**ExampleInputStream**
ConsumerConfigProperties	**aws.region**	**us-west-2**
ConsumerConfigProperties	**flink.stream.initpos**	**LATEST**

   Pilih Save (Simpan).
6. Di bawah Properties (Properti), pilih Add group (Tambahkan grup) lagi.
7. Masukkan yang berikut ini:

ID Grup	Kunci	Nilai
ProducerConfigProperties	**output.stream.name**	**ExampleOutputStream**
ProducerConfigProperties	**aws.region**	**us-west-2**

8. Di bawah Pemantauan, pastikan Memantau tingkat metrik diatur ke Aplikasi.
9. Untuk CloudWatch logging, pilih kotak centang Aktifkan.
10. Pilih Update (Perbarui).

> **Note**
>
> Saat Anda memilih untuk mengaktifkan CloudWatch pencatatan Amazon, Layanan Terkelola untuk Apache Flink membuat grup log dan aliran log untuk Anda. Nama-nama sumber daya ini adalah sebagai berikut:

- Grup log: `/aws/kinesis-analytics/MyApplication`
- Aliran log: `kinesis-analytics-log-stream`

Edit Kebijakan IAM

Edit kebijakan IAM untuk menambahkan izin agar dapat mengakses bucket Amazon S3.

Untuk mengedit kebijakan IAM agar dapat menambahkan izin bucket S3

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pilih Policies (Kebijakan). Pilih kebijakan **`kinesis-analytics-service-MyApplication-us-west-2`** yang dibuat konsol untuk Anda di bagian sebelumnya.
3. Di halaman Ringkasan, pilih Edit policy (Edit kebijakan). Pilih tab JSON.
4. Tambahkan bagian yang disorot dari contoh kebijakan berikut ke kebijakan. Ganti ID akun sampel (*012345678901*) dengan ID akun Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ka-app-code-username/getting-started-scala-1.0.jar"
            ]
        },
        {
            "Sid": "DescribeLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
```

```
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
            ]
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

Jalankan Aplikasi

Grafik pekerjaan Flink dapat dilihat dengan menjalankan aplikasi, membuka dasbor Apache Flink, dan memilih pekerjaan Flink yang diinginkan.

Hentikan Aplikasi

Untuk menghentikan aplikasi, pada MyApplicationhalaman, pilih Berhenti. Konfirmasikan tindakan.

Membuat dan menjalankan aplikasi (CLI)

Di bagian ini, Anda menggunakan AWS Command Line Interface untuk membuat dan menjalankan aplikasi Managed Service for Apache Flink. Gunakan AWS CLI perintah kinesisanalyticsv2 untuk membuat dan berinteraksi dengan Managed Service untuk aplikasi Apache Flink.

Membuat kebijakan izin

> **Note**
>
> Anda harus membuat kebijakan izin dan peran untuk aplikasi Anda. Jika Anda tidak membuat sumber daya IAM ini, aplikasi Anda tidak dapat mengakses data dan aliran log.

Pertama, Anda membuat kebijakan izin dengan dua pernyataan: satu yang memberikan izin untuk tindakan baca di aliran sumber, dan satu lagi yang memberikan izin untuk tindakan tulis di aliran sink. Anda selanjutnya melampirkan kebijakan ke IAM role (yang Anda buat di bagian berikutnya). Jadi, ketika Layanan Terkelola untuk Apache Flink mengasumsikan peran tersebut, layanan memiliki izin yang diperlukan untuk membaca dari aliran sumber dan menulis ke aliran wastafel.

Gunakan kode berikut untuk membuat kebijakan izin `AKReadSourceStreamWriteSinkStream`. Ganti **`username`** dengan nama pengguna yang Anda gunakan untuk membuat bucket Amazon S3 untuk menyimpan kode aplikasi. Ganti ID akun di Amazon Resource Names (ARN) **`(012345678901)`** dengan ID akun Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
```

```
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ka-app-code-username/getting-started-scala-1.0.jar"
            ]
        },
        {
            "Sid": "DescribeLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-analytics/
MyApplication:log-stream:*"
            ]
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-analytics/
MyApplication:log-stream:kinesis-analytics-log-stream"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
```

```
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

Untuk step-by-step petunjuk membuat kebijakan izin, lihat Tutorial: Membuat dan Melampirkan Kebijakan Terkelola Pelanggan Pertama Anda di Panduan Pengguna IAM.

Membuat kebijakan IAM

Di bagian ini, Anda membuat peran IAM yang dapat diasumsikan oleh aplikasi Managed Service for Apache Flink untuk membaca aliran sumber dan menulis ke aliran sink.

Layanan Terkelola untuk Apache Flink tidak dapat mengakses aliran Anda tanpa izin. Anda memberikan izin ini melalui IAM role. Setiap IAM role memiliki dua kebijakan yang dilampirkan. Kebijakan kepercayaan memberikan izin Layanan Terkelola untuk Apache Flink untuk mengambil peran, dan kebijakan izin menentukan apa yang dapat dilakukan Layanan Terkelola untuk Apache Flink setelah mengambil peran.

Anda melampirkan kebijakan izin yang Anda buat di bagian sebelumnya ke peran ini.

Untuk membuat IAM role

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi, pilih Peran dan kemudian Buat Peran.
3. Di bawah Pilih jenis identitas tepercaya, pilih AWSLayanan
4. Di bawah Pilih layanan yang akan menggunakan peran ini, pilih Kinesis.
5. Di bawah Pilih kasus penggunaan Anda, pilih Layanan Terkelola untuk Apache Flink.
6. Pilih Next: Permissions (Selanjutnya: Izin).
7. Di halaman Lampirkan kebijakan izin, pilih Next: Review (Selanjutnya: Tinjau). Anda melampirkan kebijakan izin setelah Anda membuat peran tersebut.

8. Di halaman Buat peran, masukkan **`MF-stream-rw-role`** untuk Role name (Nama peran). Pilih Create role (Buat peran).

   Sekarang Anda sudah membuat IAM role baru yang disebut `MF-stream-rw-role`. Selanjutnya, Anda memperbarui kebijakan kepercayaan dan izin untuk peran tersebut

9. Lampirkan kebijakan izin ke peran tersebut.

   > **Note**
   >
   > Untuk latihan ini, Managed Service for Apache Flink mengasumsikan peran ini untuk membaca data dari aliran data Kinesis (sumber) dan menulis output ke aliran data Kinesis lain. Jadi Anda melampirkan kebijakan yang Anda buat di langkah sebelumnya, Buat Kebijakan Izin.

   a. Di halaman Ringkasan, pilih tab Permissions (Izin).

   b. Pilih Attach Policies (Lampirkan Kebijakan).

   c. Di kotak pencarian, masukkan **`AKReadSourceStreamWriteSinkStream`** (kebijakan yang Anda buat bagian sebelumnya).

   d. Pilih `AKReadSourceStreamWriteSinkStream` kebijakan, lalu pilih Lampirkan kebijakan.

Anda sekarang telah membuat peran eksekusi layanan yang digunakan aplikasi Anda untuk mengakses sumber daya. Buat catatan ARN peran baru.

Untuk step-by-step petunjuk cara membuat peran, lihat Membuat Peran IAM (Konsol) di Panduan Pengguna IAM.

Buat aplikasi

Simpan kode JSON berikut ke file bernama `create_request.json`. Ganti ARN peran sampel dengan ARN untuk peran yang Anda buat sebelumnya. Ganti bucket akhiran ARN (nama pengguna) dengan akhiran yang Anda pilih di bagian sebelumnya. Ganti ID akun sampel (012345678901) di peran eksekusi layanan dengan ID akun Anda.

```
{
    "ApplicationName": "getting_started",
    "ApplicationDescription": "Scala getting started application",
```

```
    "RuntimeEnvironment": "FLINK-1_15",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-stream-rw-role",
    "ApplicationConfiguration": {
        "ApplicationCodeConfiguration": {
            "CodeContent": {
                "S3ContentLocation": {
                    "BucketARN": "arn:aws:s3:::ka-app-code-username",
                    "FileKey": "getting-started-scala-1.0.jar"
                }
            },
            "CodeContentType": "ZIPFILE"
        },
        "EnvironmentProperties":  {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleInputStream",
                    "flink.stream.initpos" : "LATEST"
               }
            },
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleOutputStream"
               }
            }
         ]
      }
    },
    "CloudWatchLoggingOptions": [
      {
         "LogStreamARN": "arn:aws:logs:us-west-2:012345678901:log-
group:MyApplication:log-stream:kinesis-analytics-log-stream"
      }
   ]
}
```

Jalankan CreateApplicationdengan permintaan berikut untuk membuat aplikasi:

```
aws kinesisanalyticsv2 create-application --cli-input-json file://create_request.json
```

Aplikasi ini sekarang dibuat. Anda memulai aplikasi di langkah berikutnya.

Mulai Aplikasi

Di bagian ini, Anda menggunakan tindakan StartApplication untuk memulai aplikasi.

Untuk memulai aplikasi

1. Simpan kode JSON berikut ke file bernama `start_request.json`.

```
{
    "ApplicationName": "getting_started",
    "RunConfiguration": {
        "ApplicationRestoreConfiguration": {
         "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"
         }
    }
}
```

2. Jalankan tindakan `StartApplication` dengan permintaan sebelumnya untuk memulai aplikasi:

```
aws kinesisanalyticsv2 start-application --cli-input-json file://start_request.json
```

Aplikasi sekarang berjalan. Anda dapat memeriksa metrik Layanan Terkelola untuk Apache Flink di CloudWatch konsol Amazon untuk memverifikasi bahwa aplikasi berfungsi.

Hentikan Aplikasi

Di bagian ini, Anda menggunakan tindakan StopApplication untuk menghentikan aplikasi.

Untuk menghentikan aplikasi

1. Simpan kode JSON berikut ke file bernama `stop_request.json`.

```
{
   "ApplicationName": "s3_sink"
}
```

2. Jalankan `StopApplication` tindakan dengan permintaan sebelumnya untuk menghentikan aplikasi:

```
aws kinesisanalyticsv2 stop-application --cli-input-json file://stop_request.json
```

Aplikasi sekarang dihentikan.

Tambahkan Opsi CloudWatch Logging

Anda dapat menggunakan AWS CLI untuk menambahkan aliran CloudWatch log Amazon ke aplikasi Anda. Untuk informasi tentang menggunakan CloudWatch Log dengan aplikasi Anda, lihat Menyiapkan Pencatatan Aplikasi.

Perbarui Properti Lingkungan

Di bagian ini, Anda menggunakan tindakan UpdateApplication untuk mengubah properti lingkungan untuk aplikasi tanpa mengompilasi ulang kode aplikasi. Dalam contoh ini, Anda mengubah Wilayah aliran sumber dan tujuan.

Untuk memperbarui properti lingkungan untuk aplikasi

1. Simpan kode JSON berikut ke file bernama `update_properties_request.json`.

```
{
      "ApplicationName": "getting_started",
       "CurrentApplicationVersionId": 1,
       "ApplicationConfigurationUpdate": {
        "EnvironmentPropertyUpdates": {
           "PropertyGroups": [
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleInputStream",
                    "flink.stream.initpos" : "LATEST"
               }
            },
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleOutputStream"
               }
```

```
            }
           ]
        }
    }
```

2. Jalankan tindakan `UpdateApplication` dengan permintaan sebelumnya untuk memperbarui properti lingkungan:

```
aws kinesisanalyticsv2 update-application --cli-input-json file://
update_properties_request.json
```

Perbarui kode aplikasi

Ketika Anda perlu memperbarui kode aplikasi Anda dengan versi baru dari paket kode Anda, Anda menggunakan tindakan UpdateApplicationCLI.

> **Note**
>
> Untuk memuat versi baru kode aplikasi dengan nama file yang sama, Anda harus menentukan versi objek baru. Untuk informasi selengkapnya tentang menggunakan versi objek Amazon S3, lihat Mengaktifkan dan Menonaktifkan Versioning.

Untuk menggunakan AWS CLI, hapus paket kode Anda sebelumnya dari bucket Amazon S3 Anda, unggah versi baru, dan panggil `UpdateApplication`, yang menentukan bucket Amazon S3 dan nama objek yang sama, serta versi objek baru. Aplikasi akan memulai ulang dengan paket kode baru.

Permintaan sampel berikut untuk tindakan `UpdateApplication` memuat ulang kode aplikasi dan memulai ulang aplikasi. Perbarui `CurrentApplicationVersionId` ke versi aplikasi saat ini. Anda dapat memeriksa versi aplikasi saat ini menggunakan tindakan `ListApplications` atau `DescribeApplication`. Perbarui akhiran nama bucket (<username>) dengan akhiran yang Anda pilih di bagian. Buat Sumber Daya Dependen

```
{{
    "ApplicationName": "getting_started",
    "CurrentApplicationVersionId": 1,
    "ApplicationConfigurationUpdate": {
        "ApplicationCodeConfigurationUpdate": {
            "CodeContentUpdate": {
```

```
                "S3ContentLocationUpdate": {
                    "BucketARNUpdate": "arn:aws:s3:::ka-app-code-<username>",
                    "FileKeyUpdate": "getting-started-scala-1.0.jar",
                    "ObjectVersionUpdate": "SAMPLEUehYngP87ex1nzYIGYgfhypvDU"
                }
            }
        }
    }
}
```

Pembersihan Sumber Daya AWS

Bagian ini mencakup prosedur untuk membersihkan sumber daya AWS yang dibuat dalam tutorial Jendela Tumbling.

Topik ini berisi bagian-bagian berikut:

- Hapus Layanan Terkelola Anda untuk Aplikasi Apache Flink
- Hapus Kinesis Data Streams Anda
- Hapus Objek dan Bucket Amazon S3 Anda
- Hapus Sumber Daya IAM Anda
- Hapus CloudWatch Sumber Daya Anda

Hapus Layanan Terkelola Anda untuk Aplikasi Apache Flink

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/flink
2. di panel Managed Service for Apache Flink, pilih. MyApplication
3. Di halaman aplikasi, pilih Delete (Hapus), lalu konfirmasikan penghapusan.

Hapus Kinesis Data Streams Anda

1. Buka konsol Kinesis di https://console.aws.amazon.com/kinesis.
2. Di panel Kinesis Data Streams, pilih. ExampleInputStream
3. Di ExampleInputStreamhalaman, pilih Hapus Stream Kinesis dan kemudian konfirmasikan penghapusan.

4. Di halaman Kinesis streams, pilih, pilih Tindakan ExampleOutputStream, pilih Hapus, lalu konfirmasikan penghapusan.

Hapus Objek dan Bucket Amazon S3 Anda

1. Buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.
2. Pilih ka-app-code- ember. <username>
3. Pilih Delete (Hapus), lalu masukkan nama bucket untuk mengonfirmasi penghapusan.

Hapus Sumber Daya IAM Anda

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di bilah navigasi, pilih Policies (Kebijakan).
3. Di kontrol filter, masukkan kinesis.
4. Pilih kebijakan kinesis-analytics-service- MyApplication -us-west-2.
5. Pilih Policy Actions (Tindakan Kebijakan), lalu pilih Delete (Hapus).
6. Di bilah navigasi, pilih Roles (Peran).
7. Pilih peran kinesis-analytics- -us-west-2. MyApplication
8. Pilih Delete role (Hapus peran), lalu konfirmasi penghapusan.

Hapus CloudWatch Sumber Daya Anda

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di bilah navigasi, pilih Logs.
3. Pilih grup log MyApplication/aws/kinesis-analytics/.
4. Pilih Delete Log Group (Hapus Grup Log), lalu konfirmasi penghapusan.

Membuat Managed Service untuk aplikasi Apache Flink dengan Apache Beam

Anda dapat menggunakan kerangka Apache Beam dengan Managed Service untuk aplikasi Apache Flink Anda untuk memproses data streaming. Managed Service untuk aplikasi Apache Flink yang menggunakan Apache Beam menggunakan Apache Flink runner untuk mengeksekusi pipeline Beam.

Untuk tutorial tentang cara menggunakan Apache Beam dalam Layanan Terkelola untuk aplikasi Apache Flink, lihat. Menggunakan CloudFormation dengan Managed Service untuk Apache Flink

Topik ini berisi bagian-bagian berikut:

- Menggunakan Apache Beam dengan Managed Service untuk Apache Flink
- Kemampuan Beam
- Membuat aplikasi menggunakan Apache Beam

Menggunakan Apache Beam dengan Managed Service untuk Apache Flink

Perhatikan hal berikut tentang penggunaan runner Apache Flink dengan Managed Service for Apache Flink:

- Metrik Apache Beam tidak dapat dilihat di konsol Managed Service for Apache Flink.
- Apache Beam hanya didukung dengan Managed Service untuk aplikasi Apache Flink yang menggunakan Apache Flink versi 1.8 ke atas. Apache Beam tidak didukung dengan Managed Service untuk aplikasi Apache Flink yang menggunakan Apache Flink versi 1.6.

Kemampuan Beam

Managed Service untuk Apache Flink mendukung kemampuan Apache Beam yang sama dengan pelari Apache Flink. Untuk informasi tentang fitur apa yang didukung dengan runner Apache Flink, lihat Matriks Kemampuan Beam.

Kami menyarankan Anda menguji aplikasi Apache Flink Anda di Layanan Terkelola untuk layanan Apache Flink untuk memverifikasi bahwa kami mendukung semua fitur yang dibutuhkan aplikasi Anda.

Membuat aplikasi menggunakan Apache Beam

Dalam latihan ini, Anda membuat Layanan Terkelola untuk aplikasi Apache Flink yang mengubah data menggunakan Apache Beam. Apache Beam adalah model pemrograman untuk memproses data streaming. Untuk informasi tentang menggunakan Apache Beam dengan Managed Service untuk Apache Flink, lihat. Menggunakan Apache Beam

> **Note**
>
> Untuk menyiapkan prasyarat yang diperlukan untuk latihan ini, selesaikan latihan Memulai (DataStream API) terlebih dulu.

Topik ini berisi bagian-bagian berikut:

- Buat Sumber Daya Dependen
- Tulis Catatan Sampel ke Aliran Input
- Unduh dan Periksa Kode Aplikasi
- Kompilasi Kode Aplikasi
- Unggah Kode Java Streaming Apache Flink
- Buat dan Jalankan Managed Service untuk Apache Flink Application
- Pembersihan Sumber Daya AWS
- Langkah Selanjutnya

Buat Sumber Daya Dependen

Sebelum Anda membuat Layanan Terkelola untuk aplikasi Apache Flink untuk latihan ini, Anda membuat sumber daya dependen berikut:

- Dua Kinesis data streams (`ExampleInputStream` dan `ExampleOutputStream`)
- Bucket Amazon S3 untuk menyimpan kode aplikasi (`ka-app-code-<username>`)

Anda dapat membuat aliran Kinesis dan bucket Amazon S3 menggunakan konsol. Untuk petunjuk membuat sumber daya ini, lihat topik berikut:

- Membuat dan Memperbarui Aliran Data di Panduan Developer Amazon Kinesis Data Streams. Beri nama aliran data **`ExampleInputStream`** dan **`ExampleOutputStream`** Anda.
- Bagaimana Cara Membuat Bucket S3? di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon. Beri bucket Amazon S3 nama yang unik secara global dengan menambahkan nama login Anda, seperti **`ka-app-code-<username>`**.

Tulis Catatan Sampel ke Aliran Input

Di bagian ini, Anda menggunakan script Python untuk menulis string acak ke aliran untuk diproses aplikasi.

> **Note**
>
> Bagian ini memerlukan AWS SDK for Python (Boto).

1. Buat file bernama `ping.py` dengan konten berikut:

```
import json
import boto3
import random

kinesis = boto3.client('kinesis')

while True:
        data = random.choice(['ping', 'telnet', 'ftp', 'tracert', 'netstat'])
        print(data)
        kinesis.put_record(
                StreamName="ExampleInputStream",
                Data=data,
                PartitionKey="partitionkey")
```

2. Jalankan skrip `ping.py`.

```
$ python ping.py
```

Biarkan skrip tetap berjalan saat menyelesaikan sisa tutorial.

Unduh dan Periksa Kode Aplikasi

Kode aplikasi Java untuk contoh ini tersedia dari GitHub. Untuk mengunduh kode aplikasi, lakukan hal berikut:

1. Instal klien Git jika Anda belum menginstalnya. Untuk informasi selengkapnya, lihat Menginstal Git.
2. Klon repositori jarak jauh dengan perintah berikut:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Buka direktori `amazon-kinesis-data-analytics-java-examples/Beam` tersebut.

Kode aplikasi terletak di file `BasicBeamStreamingJob.java`. Perhatikan hal tentang kode aplikasi berikut:

- Aplikasi ini menggunakan Apache Beam ParDountuk memproses catatan masuk dengan menjalankan fungsi transformasi kustom yang disebut. `PingPongFn`

  Kode untuk memanggil fungsi `PingPongFn` adalah sebagai berikut:

```
.apply("Pong transform",
    ParDo.of(new PingPongFn())
```

- Managed Service untuk aplikasi Apache Flink yang menggunakan Apache Beam memerlukan komponen-komponen berikut. Jika Anda tidak menyertakan komponen dan versi ini di `pom.xml` Anda, aplikasi Anda memuat versi yang salah dari dependensi lingkungan, dan karena versi tidak cocok, aplikasi Anda mengalahi crash saat runtime.

```
<jackson.version>2.10.2</jackson.version>
...
<dependency>
    <groupId>com.fasterxml.jackson.module</groupId>
    <artifactId>jackson-module-jaxb-annotations</artifactId>
    <version>2.10.2</version>
</dependency>
```

- Fungsi ubah `PingPongFn` meneruskan data input ke aliran output, kecuali data input adalah ping, yang dalam hal ini memancarkan string pong\n ke aliran output.

  Kode fungsi ubah adalah sebagai berikut:

```
    private static class PingPongFn extends DoFn<KinesisRecord, byte[]> {
    private static final Logger LOG = LoggerFactory.getLogger(PingPongFn.class);

    @ProcessElement
    public void processElement(ProcessContext c) {
        String content = new String(c.element().getDataAsBytes(),
 StandardCharsets.UTF_8);
        if (content.trim().equalsIgnoreCase("ping")) {
            LOG.info("Ponged!");
            c.output("pong\n".getBytes(StandardCharsets.UTF_8));
        } else {
            LOG.info("No action for: " + content);
            c.output(c.element().getDataAsBytes());
        }
    }
}
```

Kompilasi Kode Aplikasi

Untuk mengompilasi aplikasi, lakukan hal berikut:

1. Instal Java dan Maven jika Anda belum menginstalnya. Untuk informasi selengkapnya, lihat Prasyarat di tutorial Memulai (DataStream API).
2. Susun aplikasi dengan perintah berikut:

```
mvn package -Dflink.version=1.15.3 -Dflink.version.minor=1.8
```

> **Note**
>
> Kode sumber yang disediakan bergantung pada pustaka dari Java 11.

Mengkompilasi aplikasi membuat file JAR aplikasi (`target/basic-beam-app-1.0.jar`).

Unggah Kode Java Streaming Apache Flink

Di bagian ini, Anda mengunggah kode aplikasi ke bucket Amazon S3 yang Anda buat di bagian Buat Sumber Daya Dependen.

1. Di konsol Amazon S3, pilih bucket ka-app-code -, dan pilih Unggah. <username>
2. Di langkah Pilih file, pilih Add files (Tambahkan berkas). Navigasikan ke file `basic-beam-app-1.0.jar` yang Anda buat di langkah sebelumnya.
3. Anda tidak perlu mengubah pengaturan apa pun untuk objek, jadi pilih Upload (Unggah).

Kode aplikasi Anda sekarang disimpan di bucket Amazon S3 yang dapat diakses aplikasi Anda.

Buat dan Jalankan Managed Service untuk Apache Flink Application

Ikuti langkah-langkah ini untuk membuat, mengonfigurasi, memperbarui, dan menjalankan aplikasi menggunakan konsol.

Buat Aplikasi

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/flink
2. Pada dashboard Managed Service for Apache Flink, pilih Create Analytics Application.
3. Pada Layanan Terkelola untuk Apache Flink - Buat halaman aplikasi, berikan detail aplikasi sebagai berikut:
   - Untuk Application name (Nama aplikasi), masukkan **`MyApplication`**.
   - Untuk Runtime, pilih Apache Flink.

     > **Note**
     >
     > Managed Service untuk Apache Flink menggunakan Apache Flink versi 1.15.2.

   - Biarkan versi pulldown sebagai Apache Flink versi 1.15.2 (Versi yang disarankan).
4. Untuk Access permissions (Izin akses), pilih Create / update IAM role **`kinesis-analytics-MyApplication-us-west-2`** (Buat/perbarui IAM role).
5. Pilih Create application (Buat aplikasi).

> **Note**
>
> Saat membuat aplikasi Managed Service for Apache Flink menggunakan konsol, Anda memiliki opsi untuk membuat peran dan kebijakan IAM untuk aplikasi Anda. Aplikasi Anda menggunakan peran dan kebijakan ini untuk mengakses sumber daya dependen. Sumber daya IAM ini diberi nama menggunakan nama aplikasi dan Wilayah sebagai berikut:
>
> - Kebijakan: `kinesis-analytics-service-MyApplication-us-west-2`
> - Peran: `kinesis-analytics-MyApplication-us-west-2`

Edit Kebijakan IAM

Edit kebijakan IAM untuk menambahkan izin mengakses Kinesis data streams.

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pilih Policies (Kebijakan). Pilih kebijakan **kinesis-analytics-service-MyApplication-us-west-2** yang dibuat konsol untuk Anda di bagian sebelumnya.
3. Di halaman Ringkasan, pilih Edit policy (Edit kebijakan). Pilih tab JSON.
4. Tambahkan bagian yang disorot dari contoh kebijakan berikut ke kebijakan. Ganti ID akun sampel (*012345678901*) dengan ID akun Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "logs:DescribeLogGroups",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*",
                "arn:aws:s3:::ka-app-code-<username>/basic-beam-app-1.0.jar"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
```

```
            "Effect": "Allow",
            "Action": "logs:DescribeLogStreams",
            "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/
kinesis-analytics/MyApplication:log-stream:*"
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": "logs:PutLogEvents",
            "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/
kinesis-analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
        },
        {
            "Sid": "ListCloudwatchLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

Konfigurasikan Aplikasi

1. Pada MyApplicationhalaman, pilih Konfigurasi.

2. Di halaman Konfigurasikan aplikasi, berikan Code location (Lokasi kode):
   - Untuk Bucket Amazon S3, masukkan **`ka-app-code-<username>`**.
   - Untuk Jalur ke objek Amazon S3, masukkan **`basic-beam-app-1.0.jar`**.
3. Di bawah Akses ke sumber daya aplikasi, untuk Access permissions (Izin akses), pilih Create / update IAM role **`kinesis-analytics-MyApplication-us-west-2`** (Pilih/perbarui IAM role).
4. Masukkan yang berikut ini:

ID Grup	Kunci	Nilai
`BeamApplicationProperties`	**`InputStreamName`**	**`ExampleInputStream`**
`BeamApplicationProperties`	**`OutputStreamName`**	**`ExampleOutputStream`**
`BeamApplicationProperties`	**`AwsRegion`**	**`us-west-2`**

5. Di bawah Pemantauan, pastikan Memantau tingkat metrik diatur ke Aplikasi.
6. Untuk CloudWatch logging, pilih kotak centang Aktifkan.
7. Pilih Update (Perbarui).

> **Note**
>
> Saat Anda memilih untuk mengaktifkan CloudWatch logging, Managed Service for Apache Flink membuat grup log dan aliran log untuk Anda. Nama-nama sumber daya ini adalah sebagai berikut:
>
> - Grup log: `/aws/kinesis-analytics/MyApplication`
> - Aliran log: `kinesis-analytics-log-stream`
>
> Aliran log ini digunakan untuk memantau aplikasi. Ini bukan aliran log yang sama dengan yang digunakan aplikasi untuk mengirim hasil.

Jalankan Aplikasi

Grafik pekerjaan Flink dapat dilihat dengan menjalankan aplikasi, membuka dasbor Apache Flink, dan memilih pekerjaan Flink yang diinginkan.

Anda dapat memeriksa metrik Managed Service for Apache Flink di CloudWatch konsol untuk memverifikasi bahwa aplikasi berfungsi.

Pembersihan Sumber Daya AWS

Bagian ini mencakup prosedur untuk membersihkan sumber daya AWS yang dibuat dalam tutorial Jendela Tumbling.

Topik ini berisi bagian-bagian berikut:

- Hapus Layanan Terkelola Anda untuk Aplikasi Apache Flink
- Hapus Kinesis Data Streams Anda
- Hapus Objek dan Bucket Amazon S3 Anda
- Hapus Sumber Daya IAM Anda
- Hapus CloudWatch Sumber Daya Anda

Hapus Layanan Terkelola Anda untuk Aplikasi Apache Flink

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/flink
2. di panel Managed Service for Apache Flink, pilih. MyApplication
3. Di halaman aplikasi, pilih Delete (Hapus), lalu konfirmasikan penghapusan.

Hapus Kinesis Data Streams Anda

1. Buka konsol Kinesis di https://console.aws.amazon.com/kinesis.
2. Di panel Kinesis Data Streams, pilih. ExampleInputStream
3. Di ExampleInputStreamhalaman, pilih Hapus Stream Kinesis dan kemudian konfirmasikan penghapusan.
4. Di halaman Kinesis streams, pilih, pilih Tindakan ExampleOutputStream, pilih Hapus, lalu konfirmasikan penghapusan.

Hapus Objek dan Bucket Amazon S3 Anda

1. Buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.
2. Pilih ka-app-code- ember. <username>
3. Pilih Delete (Hapus), lalu masukkan nama bucket untuk mengonfirmasi penghapusan.

Hapus Sumber Daya IAM Anda

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di bilah navigasi, pilih Policies (Kebijakan).
3. Di kontrol filter, masukkan kinesis.
4. Pilih kebijakan kinesis-analytics-service- MyApplication -us-west-2.
5. Pilih Policy Actions (Tindakan Kebijakan), lalu pilih Delete (Hapus).
6. Di bilah navigasi, pilih Roles (Peran).
7. Pilih peran kinesis-analytics- -us-west-2. MyApplication
8. Pilih Delete role (Hapus peran), lalu konfirmasi penghapusan.

Hapus CloudWatch Sumber Daya Anda

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di bilah navigasi, pilih Logs.
3. Pilih grup log MyApplication/aws/kinesis-analytics/.
4. Pilih Delete Log Group (Hapus Grup Log), lalu konfirmasi penghapusan.

Langkah Selanjutnya

Sekarang setelah Anda membuat dan menjalankan Managed Service dasar untuk aplikasi Apache Flink yang mengubah data menggunakan Apache Beam, lihat aplikasi berikut untuk contoh Managed Service yang lebih canggih untuk solusi Apache Flink.

- Beam on Managed Service untuk Apache Flink Streaming Workshop: Dalam lokakarya ini, kami mengeksplorasi contoh ujung ke ujung yang menggabungkan aspek batch dan streaming dalam satu pipa Apache Beam yang seragam.

Lokakarya Pelatihan, Lab, dan Implementasi Solusi

end-to-end Contoh berikut menunjukkan Layanan Terkelola tingkat lanjut untuk solusi Apache Flink.

Topik

- Mengembangkan aplikasi Apache Flink secara lokal sebelum menerapkan ke Managed Service untuk Apache Flink untuk Apache Flink
- Deteksi peristiwa dengan Managed Service untuk Apache Flink Studio
- Solusi Data Streaming AWS untuk Amazon Kinesis
- Lab Clickstream dengan Apache Flink dan Apache Kafka
- Penskalaan Kustom menggunakan Application Auto Scaling
- CloudWatch Dasbor Amazon
- Solusi Data Streaming AWS untuk Amazon MSK
- Lebih Banyak Layanan Terkelola untuk Solusi Apache Flink di GitHub

Mengembangkan aplikasi Apache Flink secara lokal sebelum menerapkan ke Managed Service untuk Apache Flink untuk Apache Flink

Lokakarya ini akan menunjukkan kepada Anda dasar-dasar bangun dan mulai mengembangkan aplikasi Apache Flink secara lokal dengan tujuan jangka panjang menyebarkan ke Managed Service untuk Apache Flink untuk Apache Flink.

Solusinya dapat ditemukan di sini: Panduan Pemula untuk Pengembangan Lokal dengan Apache Flink

Deteksi peristiwa dengan Managed Service untuk Apache Flink Studio

Lokakarya ini menjelaskan deteksi peristiwa dengan Managed Service untuk Apache Flink Studio dan menerapkannya sebagai Managed Service untuk aplikasi Apache Flink

Solusinya dapat ditemukan di sini: Deteksi Acara dengan Layanan Terkelola untuk Apache Flink untuk Apache Flink

Solusi Data Streaming AWS untuk Amazon Kinesis

Solusi Data Streaming AWS untuk Amazon Kinesis secara otomatis mengonfigurasi layanan AWS yang diperlukan untuk dengan mudah menangkap, menyimpan, memproses, dan mengirimkan data streaming. Solusi ini menyediakan beberapa opsi untuk memecahkan kasus penggunaan data streaming. Layanan Terkelola untuk opsi Apache Flink menyediakan contoh ETL end-to-end streaming yang menunjukkan aplikasi dunia nyata yang menjalankan operasi analitis pada data taksi New York yang disimulasikan.

Setiap solusi mencakup komponen berikut:

- AWS CloudFormationPaket untuk menyebarkan contoh lengkap.
- CloudWatch Dasbor untuk menampilkan metrik aplikasi.
- CloudWatch alarm pada metrik aplikasi yang paling relevan.
- Semua IAM role dan kebijakan IAM yang diperlukan

Solusinya dapat ditemukan di sini: Solusi Data Streaming untuk Amazon Kinesis

Lab Clickstream dengan Apache Flink dan Apache Kafka

Lab ujung ke ujung untuk kasus penggunaan clickstream menggunakan Amazon Managed Streaming for Apache Kafka untuk penyimpanan streaming dan Layanan Terkelola untuk Apache Flink untuk aplikasi Apache Flink untuk pemrosesan streaming.

Solusinya dapat ditemukan di sini: Lab Clickstream

Penskalaan Kustom menggunakan Application Auto Scaling

Contoh yang membantu pengguna secara otomatis menskalakan Layanan Terkelola mereka untuk aplikasi Apache Flink menggunakan Application Auto Scaling. Ini memungkinkan pengguna mengatur kebijakan penskalaan kustom dan atribut penskalaan kustom.

Solusinya dapat ditemukan di sini:

- Layanan Terkelola untuk Apache Flink App Autoscaling
- Penskalaan Terjadwal

Untuk informasi selengkapnya tentang Anda dapat melakukan penskalaan khusus, lihat Mengaktifkan penskalaan berbasis metrik dan terjadwal untuk Amazon Managed Service for Apache Flink.

CloudWatch Dasbor Amazon

CloudWatch Dasbor sampel untuk memantau Layanan Terkelola untuk aplikasi Apache Flink. Dasbor sampel juga mencakup aplikasi demo untuk membantu mendemonstrasikan fungsionalitas dasbor.

Solusinya dapat ditemukan di sini: Layanan Terkelola untuk Dasbor Metrik Apache Flink

Solusi Data Streaming AWS untuk Amazon MSK

Solusi Data Streaming AWS untuk Amazon MSK menyediakan templat AWS CloudFormation tempat data mengalir melalui produsen, penyimpanan streaming, konsumen, dan tujuan.

Solusinya dapat ditemukan di sini: Solusi Data AWS Streaming untuk Amazon MSK

Lebih Banyak Layanan Terkelola untuk Solusi Apache Flink di GitHub

end-to-end Contoh berikut menunjukkan Layanan Terkelola tingkat lanjut untuk solusi Apache Flink dan tersedia di: GitHub

- Layanan Dikelola Amazon untuk Apache Flink Flink - Utilitas Benchmarking
- Snapshot Manager - Amazon Managed Service untuk Apache Flink untuk Apache Flink
- Streaming ETL dengan Apache Flink dan Amazon Managed Service untuk Apache Flink
- Analisis sentimen waktu nyata pada umpan balik pelanggan

Utilitas

Utilitas berikut dapat membuat penggunaan Layanan Terkelola untuk layanan Apache Flink lebih mudah digunakan:

Topik

Manajer snapshot

Ini adalah praktik terbaik bagi Aplikasi Flink untuk secara teratur memicu savepoint/snapshot untuk memungkinkan pemulihan kegagalan yang lebih mulus. Manajer snapshot mengotomatiskan tugas ini dan menawarkan manfaat berikut:

- mengambil snapshot baru dari Managed Service yang sedang berjalan untuk Apache Flink untuk Apache Flink Application
- mendapat hitungan snapshot aplikasi
- memeriksa apakah hitungannya lebih dari jumlah snapshot yang diperlukan
- menghapus snapshot lama yang lebih tua dari nomor yang diperlukan

Sebagai contoh, lihat Manajer snapshot.

Benchmarking

Managed Service untuk Apache Flink Flink Benchmarking Utility membantu perencanaan kapasitas, pengujian integrasi, dan benchmarking Managed Service untuk Apache Flink untuk aplikasi Apache Flink.

Sebagai contoh, lihat Benchmarking

Layanan Terkelola untuk Apache Flink: Contoh

Bagian ini memberikan contoh membuat dan bekerja dengan aplikasi di Managed Service untuk Apache Flink. Mereka menyertakan contoh kode dan step-by-step instruksi untuk membantu Anda membuat Layanan Terkelola untuk aplikasi Apache Flink dan menguji hasil Anda.

Sebelum Anda menjelajahi contoh-contoh ini, sebaiknya tinjau hal berikut terlebih dulu:

- Cara Kerjanya
- Memulai (DataStream API)

> **Note**
>
> Contoh ini menganggap Anda menggunakan Wilayah US West (Oregon) (`us-west-2`). Jika Anda menggunakan Wilayah yang berbeda, perbarui kode aplikasi, perintah, dan IAM role Anda dengan tepat.

Topik

- DataStream Contoh API
- Contoh Python
- Contoh Scala

DataStream Contoh API

Contoh berikut menunjukkan cara membuat aplikasi menggunakan Apache Flink API DataStream .

Topik

- Contoh: Jendela Tumbling
- Contoh: Jendela Geser
- Contoh: Menulis ke Bucket Amazon S3
- Tutorial: Menggunakan Managed Service untuk aplikasi Apache Flink untuk Mereplikasi Data dari Satu Topik di MSK Cluster ke Lainnya dalam VPC
- Contoh: Gunakan Konsumen EFO dengan Kinesis Data Stream

- Contoh: Menulis ke Kinesis Data Firehose
- Contoh: Baca dari Aliran Kinesis di Akun Berbeda.
- Tutorial: Menggunakan Truststore Kustom dengan Amazon MSK

Contoh: Jendela Tumbling

Dalam latihan ini, Anda membuat Layanan Terkelola untuk aplikasi Apache Flink yang mengumpulkan data menggunakan jendela tumbling. Agregrasi diaktifkan secara default di Flink. Untuk menonaktifkannya, gunakan yang berikut ini:

```
sink.producer.aggregation-enabled' = 'false'
```

> **Note**
>
> Untuk menyiapkan prasyarat yang diperlukan untuk latihan ini, selesaikan latihan Memulai (DataStream API) terlebih dulu.

Topik ini berisi bagian-bagian berikut:

- Buat Sumber Daya Dependen
- Tulis Catatan Sampel ke Aliran Input
- Unduh dan Periksa Kode Aplikasi
- Kompilasi Kode Aplikasi
- Unggah Kode Java Streaming Apache Flink
- Buat dan Jalankan Managed Service untuk Apache Flink Application
- Pembersihan Sumber Daya AWS

Buat Sumber Daya Dependen

Sebelum Anda membuat Layanan Terkelola untuk aplikasi Apache Flink untuk latihan ini, Anda membuat sumber daya dependen berikut:

- Dua Kinesis data streams (`ExampleInputStream` dan `ExampleOutputStream`)
- Bucket Amazon S3 untuk menyimpan kode aplikasi (`ka-app-code-<username>`)

Anda dapat membuat aliran Kinesis dan bucket Amazon S3 menggunakan konsol. Untuk petunjuk membuat sumber daya ini, lihat topik berikut:

- Membuat dan Memperbarui Aliran Data di Panduan Developer Amazon Kinesis Data Streams. Beri nama aliran data **`ExampleInputStream`** dan **`ExampleOutputStream`** Anda.
- Bagaimana Cara Membuat Bucket S3? di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon. Beri bucket Amazon S3 nama yang unik secara global dengan menambahkan nama login Anda, seperti **`ka-app-code-<username>`**.

Tulis Catatan Sampel ke Aliran Input

Di bagian ini, Anda menggunakan script Python untuk menulis catatan sampel ke aliran untuk diproses aplikasi.

> **Note**
>
> Bagian ini memerlukan AWS SDK for Python (Boto).

1. Buat file bernama `stock.py` dengan konten berikut:

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        'event_time': datetime.datetime.now().isoformat(),
        'ticker': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'price': round(random.random() * 100, 2)}


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
```

```
            print(data)
            kinesis_client.put_record(
                StreamName=stream_name,
                Data=json.dumps(data),
                PartitionKey="partitionkey")


    if __name__ == '__main__':
        generate(STREAM_NAME, boto3.client('kinesis', region_name='us-west-2'))
```

2. Jalankan skrip `stock.py`.

```
$ python stock.py
```

Biarkan skrip tetap berjalan saat menyelesaikan sisa tutorial.

Unduh dan Periksa Kode Aplikasi

Kode aplikasi Java untuk contoh ini tersedia dari GitHub. Untuk mengunduh kode aplikasi, lakukan hal berikut:

1. Instal klien Git jika Anda belum menginstalnya. Untuk informasi selengkapnya, lihat Menginstal Git.
2. Klon repositori jarak jauh dengan perintah berikut:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Buka direktori `amazon-kinesis-data-analytics-java-examples/TumblingWindow` tersebut.

Kode aplikasi terletak di file `TumblingWindowStreamingJob.java`. Perhatikan hal tentang kode aplikasi berikut:

- Aplikasi menggunakan sumber Kinesis untuk membaca dari aliran sumber. Cuplikan berikut ini membuat sumber Kinesis:

```
return env.addSource(new FlinkKinesisConsumer<>(inputStreamName,
                new SimpleStringSchema(), inputProperties));
```

- Tambahkan pernyataan impor berikut:

```
import
 org.apache.flink.streaming.api.windowing.assigners.TumblingProcessingTimeWindows; //
flink 1.13 onward
```

- Aplikasi menggunakan operator `timeWindow` untuk mencari hitungan nilai untuk setiap simbol saham melalui jendela tumbling 5 detik. Kode berikut membuat operator dan mengirimkan data agregat ke sink Kinesis Data Streams baru:

```
input.flatMap(new Tokenizer()) // Tokenizer for generating words
                    .keyBy(0) // Logically partition the stream for each word

                    .window(TumblingProcessingTimeWindows.of(Time.seconds(5))) //
Flink 1.13 onward
                    .sum(1) // Sum the number of words per partition
                    .map(value -> value.f0 + "," + value.f1.toString() + "\n")
                    .addSink(createSinkFromStaticConfig());
```

Kompilasi Kode Aplikasi

Untuk mengompilasi aplikasi, lakukan hal berikut:

1. Instal Java dan Maven jika Anda belum menginstalnya. Untuk informasi selengkapnya, lihat Prasyarat di tutorial Memulai (DataStream API).
2. Susun aplikasi dengan perintah berikut:

```
mvn package -Dflink.version=1.15.3
```

> **Note**
>
> Kode sumber yang disediakan bergantung pada pustaka dari Java 11.

Mengkompilasi aplikasi membuat file JAR aplikasi (`target/aws-kinesis-analytics-java-apps-1.0.jar`).

Unggah Kode Java Streaming Apache Flink

Di bagian ini, Anda mengunggah kode aplikasi ke bucket Amazon S3 yang Anda buat di bagian Buat Sumber Daya Dependen.

1. Di konsol Amazon S3, pilih bucket ka-app-code -, dan pilih Unggah. <username>
2. Di langkah Pilih file, pilih Add files (Tambahkan berkas). Navigasikan ke file `aws-kinesis-analytics-java-apps-1.0.jar` yang Anda buat di langkah sebelumnya.
3. Anda tidak perlu mengubah pengaturan apa pun untuk objek, jadi pilih Upload (Unggah).

Kode aplikasi Anda sekarang disimpan di bucket Amazon S3 yang dapat diakses aplikasi Anda.

Buat dan Jalankan Managed Service untuk Apache Flink Application

Ikuti langkah-langkah ini untuk membuat, mengonfigurasi, memperbarui, dan menjalankan aplikasi menggunakan konsol.

Buat Aplikasi

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/flink
2. Pada dashboard Managed Service for Apache Flink, pilih Create Analytics Application.
3. Pada Layanan Terkelola untuk Apache Flink - Buat halaman aplikasi, berikan detail aplikasi sebagai berikut:
   - Untuk Application name (Nama aplikasi), masukkan **`MyApplication`**.
   - Untuk Runtime, pilih Apache Flink.

     > **Note**
     >
     > Managed Service untuk Apache Flink menggunakan Apache Flink versi 1.15.2.

   - Biarkan versi pulldown sebagai Apache Flink versi 1.15.2 (Versi yang disarankan).
4. Untuk Access permissions (Izin akses), pilih Create / update IAM role **`kinesis-analytics-MyApplication-us-west-2`** (Buat/perbarui IAM role).
5. Pilih Create application (Buat aplikasi).

> **Note**
>
> Saat membuat aplikasi Managed Service for Apache Flink menggunakan konsol, Anda memiliki opsi untuk membuat peran dan kebijakan IAM untuk aplikasi Anda. Aplikasi Anda menggunakan peran dan kebijakan ini untuk mengakses sumber daya dependen. Sumber daya IAM ini diberi nama menggunakan nama aplikasi dan Wilayah sebagai berikut:
>
> - Kebijakan: `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Peran: `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Edit Kebijakan IAM

Edit kebijakan IAM untuk menambahkan izin mengakses Kinesis data streams.

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pilih Policies (Kebijakan). Pilih kebijakan **`kinesis-analytics-service-MyApplication-us-west-2`** yang dibuat konsol untuk Anda di bagian sebelumnya.
3. Di halaman Ringkasan, pilih Edit policy (Edit kebijakan). Pilih tab JSON.
4. Tambahkan bagian yang disorot dari contoh kebijakan berikut ke kebijakan. Ganti ID akun sampel (*`012345678901`*) dengan ID akun Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "logs:DescribeLogGroups",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*",
                "arn:aws:s3:::ka-app-code-<username>/aws-kinesis-analytics-java-
apps-1.0.jar"
            ]
        },
        {
```

```
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": "logs:DescribeLogStreams",
            "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/
kinesis-analytics/MyApplication:log-stream:*"
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": "logs:PutLogEvents",
            "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/
kinesis-analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
        },
        {
            "Sid": "ListCloudwatchLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

Konfigurasikan Aplikasi

1. Pada MyApplicationhalaman, pilih Konfigurasi.

2. Di halaman Konfigurasikan aplikasi, berikan Code location (Lokasi kode):
   - Untuk Bucket Amazon S3, masukkan **`ka-app-code-<username>`**.
   - Untuk Jalur ke objek Amazon S3, masukkan **`aws-kinesis-analytics-java-apps-1.0.jar`**.
3. Di bawah Akses ke sumber daya aplikasi, untuk Access permissions (Izin akses), pilih Create / update IAM role **`kinesis-analytics-MyApplication-us-west-2`** (Pilih/perbarui IAM role).
4. Di bawah Pemantauan, pastikan Memantau tingkat metrik diatur ke Aplikasi.
5. Untuk CloudWatch logging, pilih kotak centang Aktifkan.
6. Pilih Perbarui.

> **Note**
>
> Saat Anda memilih untuk mengaktifkan CloudWatch logging, Managed Service for Apache Flink membuat grup log dan aliran log untuk Anda. Nama-nama sumber daya ini adalah sebagai berikut:
>
> - Grup log: `/aws/kinesis-analytics/MyApplication`
> - Aliran log: `kinesis-analytics-log-stream`
>
> Aliran log ini digunakan untuk memantau aplikasi. Ini bukan aliran log yang sama dengan yang digunakan aplikasi untuk mengirim hasil.

Jalankan Aplikasi

1. Pada MyApplicationhalaman, pilih Jalankan. Biarkan opsi Run without snapshot (Jalankan tanpa snapshot) dipilih, dan konfirmasikan tindakan.
2. Ketika aplikasi berjalan, refresh halaman. Konsol menunjukkan Grafik aplikasi.

Anda dapat memeriksa metrik Managed Service for Apache Flink di CloudWatch konsol untuk memverifikasi bahwa aplikasi berfungsi.

Pembersihan Sumber Daya AWS

Bagian ini mencakup prosedur untuk membersihkan sumber daya AWS yang dibuat dalam tutorial Jendela Tumbling.

Topik ini berisi bagian-bagian berikut:

- Hapus Layanan Terkelola Anda untuk Aplikasi Apache Flink
- Hapus Kinesis Data Streams Anda
- Hapus Objek dan Bucket Amazon S3 Anda
- Hapus Sumber Daya IAM Anda
- Hapus CloudWatch Sumber Daya Anda

Hapus Layanan Terkelola Anda untuk Aplikasi Apache Flink

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/flink
2. di panel Managed Service for Apache Flink, pilih. MyApplication
3. Di halaman aplikasi, pilih Delete (Hapus), lalu konfirmasikan penghapusan.

Hapus Kinesis Data Streams Anda

1. Buka konsol Kinesis di https://console.aws.amazon.com/kinesis.
2. Di panel Kinesis Data Streams, pilih. ExampleInputStream
3. Di ExampleInputStreamhalaman, pilih Hapus Stream Kinesis dan kemudian konfirmasikan penghapusan.
4. Di halaman Kinesis streams, pilih, pilih Tindakan ExampleOutputStream, pilih Hapus, lalu konfirmasikan penghapusan.

Hapus Objek dan Bucket Amazon S3 Anda

1. Buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.
2. Pilih ka-app-code- ember. <username>
3. Pilih Delete (Hapus), lalu masukkan nama bucket untuk mengonfirmasi penghapusan.

Hapus Sumber Daya IAM Anda

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di bilah navigasi, pilih Policies (Kebijakan).
3. Di kontrol filter, masukkan kinesis.
4. Pilih kebijakan kinesis-analytics-service- MyApplication -us-west-2.
5. Pilih Policy Actions (Tindakan Kebijakan), lalu pilih Delete (Hapus).
6. Di bilah navigasi, pilih Roles (Peran).
7. Pilih peran kinesis-analytics- -us-west-2. MyApplication
8. Pilih Delete role (Hapus peran), lalu konfirmasi penghapusan.

Hapus CloudWatch Sumber Daya Anda

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di bilah navigasi, pilih Logs.
3. Pilih grup log MyApplication/aws/kinesis-analytics/.
4. Pilih Delete Log Group (Hapus Grup Log), lalu konfirmasi penghapusan.

Contoh: Jendela Geser

> **Note**
>
> Untuk menyiapkan prasyarat yang diperlukan untuk latihan ini, selesaikan latihan Memulai (DataStream API) terlebih dulu.

Topik ini berisi bagian-bagian berikut:

- Buat Sumber Daya Dependen
- Tulis Catatan Sampel ke Aliran Input
- Unduh dan Periksa Kode Aplikasi
- Kompilasi Kode Aplikasi
- Unggah Kode Java Streaming Apache Flink

- Buat dan Jalankan Managed Service untuk Apache Flink Application
- Pembersihan Sumber Daya AWS

Buat Sumber Daya Dependen

Sebelum Anda membuat Layanan Terkelola untuk aplikasi Apache Flink untuk latihan ini, Anda membuat sumber daya dependen berikut:

- Dua Kinesis data streams (`ExampleInputStream` dan `ExampleOutputStream`)
- Bucket Amazon S3 untuk menyimpan kode aplikasi (`ka-app-code-<username>`)

Anda dapat membuat aliran Kinesis dan bucket Amazon S3 menggunakan konsol. Untuk petunjuk membuat sumber daya ini, lihat topik berikut:

- Membuat dan Memperbarui Aliran Data di Panduan Developer Amazon Kinesis Data Streams. Beri nama aliran data **`ExampleInputStream`** dan **`ExampleOutputStream`** Anda.
- Bagaimana Cara Membuat Bucket S3? di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon. Beri bucket Amazon S3 nama yang unik secara global dengan menambahkan nama login Anda, seperti **`ka-app-code-<username>`**.

Tulis Catatan Sampel ke Aliran Input

Di bagian ini, Anda menggunakan script Python untuk menulis catatan sampel ke aliran untuk diproses aplikasi.

> **Note**
>
> Bagian ini memerlukan AWS SDK for Python (Boto).

1. Buat file bernama `stock.py` dengan konten berikut:

```
import datetime
import json
import random
import boto3
```

```
STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        "EVENT_TIME": datetime.datetime.now().isoformat(),
        "TICKER": random.choice(["AAPL", "AMZN", "MSFT", "INTC", "TBV"]),
        "PRICE": round(random.random() * 100, 2),
    }


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(data),
 PartitionKey="partitionkey"
        )


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

2. Jalankan skrip `stock.py`.

```
$ python stock.py
```

Biarkan skrip tetap berjalan saat menyelesaikan sisa tutorial.

Unduh dan Periksa Kode Aplikasi

Kode aplikasi Java untuk contoh ini tersedia dari GitHub. Untuk mengunduh kode aplikasi, lakukan hal berikut:

1. Instal klien Git jika Anda belum menginstalnya. Untuk informasi selengkapnya, lihat Menginstal Git.
2. Klon repositori jarak jauh dengan perintah berikut:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Buka direktori `amazon-kinesis-data-analytics-java-examples/SlidingWindow` tersebut.

Kode aplikasi terletak di file `SlidingWindowStreamingJobWithParallelism.java`. Perhatikan hal tentang kode aplikasi berikut:

- Aplikasi menggunakan sumber Kinesis untuk membaca dari aliran sumber. Cuplikan berikut ini membuat sumber Kinesis:

```
return env.addSource(new FlinkKinesisConsumer<>(inputStreamName,
                new SimpleStringSchema(), inputProperties));
```

- Aplikasi menggunakan operator `timeWindow` untuk menemukan nilai minimum untuk setiap simbol saham melalui jendela 10 detik yang bergeser 5 detik. Kode berikut membuat operator dan mengirimkan data agregat ke sink Kinesis Data Streams baru:
- Tambahkan pernyataan impor berikut:

```
import
 org.apache.flink.streaming.api.windowing.assigners.TumblingProcessingTimeWindows; //
flink 1.13 onward
```

- Aplikasi menggunakan operator `timeWindow` untuk mencari hitungan nilai untuk setiap simbol saham melalui jendela tumbling 5 detik. Kode berikut membuat operator dan mengirimkan data agregat ke sink Kinesis Data Streams baru:

```
input.flatMap(new Tokenizer()) // Tokenizer for generating words
                .keyBy(0) // Logically partition the stream for each word

  .window(TumblingProcessingTimeWindows.of(Time.seconds(5))) //Flink 1.13 onward
                .sum(1) // Sum the number of words per partition
                .map(value -> value.f0 + "," + value.f1.toString() + "\n")
                .addSink(createSinkFromStaticConfig());
```

Kompilasi Kode Aplikasi

Untuk mengompilasi aplikasi, lakukan hal berikut:

1. Instal Java dan Maven jika Anda belum menginstalnya. Untuk informasi selengkapnya, lihat Prasyarat di tutorial Memulai (DataStream API).

2. Susun aplikasi dengan perintah berikut:

```
mvn package -Dflink.version=1.15.3
```

> **Note**
>
> Kode sumber yang disediakan bergantung pada pustaka dari Java 11.

Mengkompilasi aplikasi membuat file JAR aplikasi (`target/aws-kinesis-analytics-java-apps-1.0.jar`).

Unggah Kode Java Streaming Apache Flink

Di bagian ini, Anda mengunggah kode aplikasi ke bucket Amazon S3 yang Anda buat di bagian Buat Sumber Daya Dependen.

1. Di konsol Amazon S3, pilih bucket ka-app-code -, lalu pilih Unggah. <username>
2. Di langkah Pilih file, pilih Add files (Tambahkan berkas). Navigasikan ke file `aws-kinesis-analytics-java-apps-1.0.jar` yang Anda buat di langkah sebelumnya.
3. Anda tidak perlu mengubah pengaturan apa pun untuk objek, jadi pilih Upload (Unggah).

Kode aplikasi Anda sekarang disimpan di bucket Amazon S3 yang dapat diakses aplikasi Anda.

Buat dan Jalankan Managed Service untuk Apache Flink Application

Ikuti langkah-langkah ini untuk membuat, mengonfigurasi, memperbarui, dan menjalankan aplikasi menggunakan konsol.

Buat Aplikasi

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/flink
2. Pada dashboard Managed Service for Apache Flink, pilih Create Analytics Application.
3. Pada Layanan Terkelola untuk Apache Flink - Buat halaman aplikasi, berikan detail aplikasi sebagai berikut:
   - Untuk Application name (Nama aplikasi), masukkan **`MyApplication`**.
   - Untuk Runtime, pilih Apache Flink.

- Biarkan versi pulldown sebagai Apache Flink versi 1.15.2 (Versi yang disarankan).

4. Untuk Access permissions (Izin akses), pilih Create / update IAM role **kinesis-analytics-MyApplication-us-west-2** (Buat/perbarui IAM role).
5. Pilih Create application (Buat aplikasi).

> **Note**
>
> Saat membuat aplikasi Managed Service for Apache Flink menggunakan konsol, Anda memiliki opsi untuk membuat peran dan kebijakan IAM untuk aplikasi Anda. Aplikasi Anda menggunakan peran dan kebijakan ini untuk mengakses sumber daya dependen. Sumber daya IAM ini diberi nama menggunakan nama aplikasi dan Wilayah sebagai berikut:
>
> - Kebijakan: `kinesis-analytics-service-MyApplication-us-west-2`
> - Peran: `kinesisanalytics-MyApplication-us-west-2`

Edit Kebijakan IAM

Edit kebijakan IAM untuk menambahkan izin mengakses Kinesis data streams.

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pilih Policies (Kebijakan). Pilih kebijakan **kinesis-analytics-service-MyApplication-us-west-2** yang dibuat konsol untuk Anda di bagian sebelumnya.
3. Di halaman Ringkasan, pilih Edit policy (Edit kebijakan). Pilih tab JSON.
4. Tambahkan bagian yang disorot dari contoh kebijakan berikut ke kebijakan. Ganti ID akun sampel (*012345678901*) dengan ID akun Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "logs:DescribeLogGroups",
                "s3:GetObjectVersion"
            ],
```

```
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*",
                "arn:aws:s3:::ka-app-code-<username>/aws-kinesis-analytics-java-
apps-1.0.jar"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": "logs:DescribeLogStreams",
            "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/
kinesis-analytics/MyApplication:log-stream:*"
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": "logs:PutLogEvents",
            "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/
kinesis-analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
        },
        {
            "Sid": "ListCloudwatchLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
```

```
    ]
}
```

Konfigurasikan Aplikasi

1. Pada MyApplicationhalaman, pilih Konfigurasi.
2. Di halaman Konfigurasikan aplikasi, berikan Code location (Lokasi kode):
   - Untuk Bucket Amazon S3, masukkan **`ka-app-code-<username>`**.
   - Untuk Jalur ke objek Amazon S3, masukkan **`aws-kinesis-analytics-java-apps-1.0.jar`**.
3. Di bawah Akses ke sumber daya aplikasi, untuk Access permissions (Izin akses), pilih Create / update IAM role **`kinesis-analytics-MyApplication-us-west-2`** (Pilih/perbarui IAM role).
4. Di bawah Pemantauan, pastikan Memantau tingkat metrik diatur ke Aplikasi.
5. Untuk CloudWatch logging, pilih kotak centang Aktifkan.
6. Pilih Perbarui.

> **Note**
>
> Saat Anda memilih untuk mengaktifkan CloudWatch pencatatan Amazon, Layanan Terkelola untuk Apache Flink membuat grup log dan aliran log untuk Anda. Nama-nama sumber daya ini adalah sebagai berikut:
>
> - Grup log: `/aws/kinesis-analytics/MyApplication`
> - Aliran log: `kinesis-analytics-log-stream`
>
> Aliran log ini digunakan untuk memantau aplikasi. Ini bukan aliran log yang sama dengan yang digunakan aplikasi untuk mengirim hasil.

Konfigurasikan Paralelisme Aplikasi

Contoh aplikasi ini menggunakan eksekusi tugas paralel. Kode aplikasi berikut mengatur paralelisme operator `min`:

```
.setParallelism(3) // Set parallelism for the min operator
```

Aplikasi paralelisme tidak boleh lebih besar dari paralelisme yang disediakan, yang memiliki default sama dengan 1. Untuk meningkatkan paralelisme aplikasi Anda, gunakan tindakan AWS CLI berikut:

```
aws kinesisanalyticsv2 update-application
      --application-name MyApplication
      --current-application-version-id <VersionId>
      --application-configuration-update "{\"FlinkApplicationConfigurationUpdate
\": { \"ParallelismConfigurationUpdate\": {\"ParallelismUpdate\": 5,
 \"ConfigurationTypeUpdate\": \"CUSTOM\" }}}"
```

Anda dapat mengambil ID versi aplikasi saat ini menggunakan ListApplicationstindakan DescribeApplicationatau.

Jalankan Aplikasi

Grafik pekerjaan Flink dapat dilihat dengan menjalankan aplikasi, membuka dasbor Apache Flink, dan memilih pekerjaan Flink yang diinginkan.

Anda dapat memeriksa metrik Managed Service for Apache Flink di CloudWatch konsol untuk memverifikasi bahwa aplikasi berfungsi.

Pembersihan Sumber Daya AWS

Bagian ini mencakup prosedur untuk membersihkan sumber daya AWS yang dibuat dalam tutorial Jendela Geser.

Topik ini berisi bagian-bagian berikut:

- Hapus Layanan Terkelola Anda untuk Aplikasi Apache Flink
- Hapus Kinesis Data Streams Anda
- Hapus Objek dan Bucket Amazon S3 Anda
- Hapus Sumber Daya IAM Anda
- Hapus CloudWatch Sumber Daya Anda

Hapus Layanan Terkelola Anda untuk Aplikasi Apache Flink

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/flink

2. Di panel Managed Service for Apache Flink, pilih. MyApplication
3. Di halaman aplikasi, pilih Delete (Hapus), lalu konfirmasikan penghapusan.

Hapus Kinesis Data Streams Anda

1. Buka konsol Kinesis di https://console.aws.amazon.com/kinesis.
2. Di panel Kinesis Data Streams, pilih. ExampleInputStream
3. Di ExampleInputStreamhalaman, pilih Hapus Stream Kinesis dan kemudian konfirmasikan penghapusan.
4. Di halaman Kinesis streams, pilih, pilih Tindakan ExampleOutputStream, pilih Hapus, lalu konfirmasikan penghapusan.

Hapus Objek dan Bucket Amazon S3 Anda

1. Buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.
2. Pilih ka-app-code- ember. <username>
3. Pilih Delete (Hapus), lalu masukkan nama bucket untuk mengonfirmasi penghapusan.

Hapus Sumber Daya IAM Anda

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di bilah navigasi, pilih Policies (Kebijakan).
3. Di kontrol filter, masukkan kinesis.
4. Pilih kebijakan kinesis-analytics-service- MyApplication -us-west-2.
5. Pilih Policy Actions (Tindakan Kebijakan), lalu pilih Delete (Hapus).
6. Di bilah navigasi, pilih Roles (Peran).
7. Pilih peran kinesis-analytics- -us-west-2. MyApplication
8. Pilih Delete role (Hapus peran), lalu konfirmasi penghapusan.

Hapus CloudWatch Sumber Daya Anda

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di bilah navigasi, pilih Logs.

3. Pilih grup log MyApplication/aws/kinesis-analytics/.
4. Pilih Delete Log Group (Hapus Grup Log), lalu konfirmasi penghapusan.

Contoh: Menulis ke Bucket Amazon S3

Dalam latihan ini, Anda membuat Layanan Terkelola untuk Apache Flink yang memiliki aliran data Kinesis sebagai sumber dan bucket Amazon S3 sebagai wastafel. Dengan menggunakan sink, Anda dapat memverifikasi output aplikasi di konsol Amazon S3.

> **Note**
>
> Untuk menyiapkan prasyarat yang diperlukan untuk latihan ini, selesaikan latihan Memulai (DataStream API) terlebih dulu.

Topik ini berisi bagian-bagian berikut:

- Buat Sumber Daya Dependen
- Tulis Catatan Sampel ke Aliran Input
- Unduh dan Periksa Kode Aplikasi
- Ubah Kode Aplikasi
- Kompilasi Kode Aplikasi
- Unggah Kode Java Streaming Apache Flink
- Buat dan Jalankan Managed Service untuk Apache Flink Application
- Verifikasi Output Aplikasi
- Opsional: Sesuaikan Sumber dan Sink
- Pembersihan Sumber Daya AWS

Buat Sumber Daya Dependen

Sebelum Anda membuat Layanan Terkelola untuk Apache Flink untuk latihan ini, Anda membuat sumber daya dependen berikut:

- Kinesis data stream (`ExampleInputStream`).
- Bucket Amazon S3 untuk menyimpan kode dan output aplikasi (`ka-app-code-<username>`)

> **Note**
>
> Layanan Terkelola untuk Apache Flink tidak dapat menulis data ke Amazon S3 dengan enkripsi sisi server diaktifkan pada Layanan Terkelola untuk Apache Flink.

Anda dapat membuat aliran Kinesis dan bucket Amazon S3 menggunakan konsol. Untuk petunjuk membuat sumber daya ini, lihat topik berikut:

- Membuat dan Memperbarui Aliran Data di Panduan Developer Amazon Kinesis Data Streams. Beri nama aliran data **`ExampleInputStream`** Anda.
- Bagaimana Cara Membuat Bucket S3? di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon. Beri bucket Amazon S3 nama yang unik secara global dengan menambahkan nama login Anda, seperti **`ka-app-code-`*`<username>`***. Buat dua folder (**`code`** dan **`data`**) dalam bucket Amazon S3.

Aplikasi membuat CloudWatch sumber daya berikut jika belum ada:

- Grup log yang disebut `/AWS/KinesisAnalytics-java/MyApplication`.
- Aliran log yang disebut `kinesis-analytics-log-stream`.

Tulis Catatan Sampel ke Aliran Input

Di bagian ini, Anda menggunakan script Python untuk menulis catatan sampel ke aliran untuk diproses aplikasi.

> **Note**
>
> Bagian ini memerlukan AWS SDK for Python (Boto).

1. Buat file bernama `stock.py` dengan konten berikut:

```
import datetime
import json
import random
import boto3
```

```
    STREAM_NAME = "ExampleInputStream"


    def get_data():
        return {
            'event_time': datetime.datetime.now().isoformat(),
            'ticker': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
            'price': round(random.random() * 100, 2)}


    def generate(stream_name, kinesis_client):
        while True:
            data = get_data()
            print(data)
            kinesis_client.put_record(
                StreamName=stream_name,
                Data=json.dumps(data),
                PartitionKey="partitionkey")


    if __name__ == '__main__':
        generate(STREAM_NAME, boto3.client('kinesis', region_name='us-west-2'))
```

2. Jalankan skrip `stock.py`.

```
$ python stock.py
```

Biarkan skrip tetap berjalan saat menyelesaikan sisa tutorial.

Unduh dan Periksa Kode Aplikasi

Kode aplikasi Java untuk contoh ini tersedia dari GitHub. Untuk mengunduh kode aplikasi, lakukan hal berikut:

1. Instal klien Git jika Anda belum menginstalnya. Untuk informasi selengkapnya, lihat Menginstal Git.
2. Klon repositori jarak jauh dengan perintah berikut:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Buka direktori `amazon-kinesis-data-analytics-java-examples/S3Sink` tersebut.

Kode aplikasi terletak di file `S3StreamingSinkJob.java`. Perhatikan hal tentang kode aplikasi berikut:

- Aplikasi menggunakan sumber Kinesis untuk membaca dari aliran sumber. Cuplikan berikut ini membuat sumber Kinesis:

```
return env.addSource(new FlinkKinesisConsumer<>(inputStreamName,
                new SimpleStringSchema(), inputProperties));
```

- Anda perlu menambahkan pernyataan impor berikut:

```
import
 org.apache.flink.streaming.api.windowing.assigners.TumblingProcessingTimeWindows;
```

- Aplikasi ini menggunakan sink S3 Apache Flink untuk menulis ke Amazon S3.

  Sink membaca pesan di jendela tumbling, mengenkodekan pesan ke objek bucket S3, dan mengirimkan objek yang dienkodekan ke sink S3. Kode berikut mengenkodekan objek untuk mengirim ke Amazon S3:

```
input.map(value -> { // Parse the JSON
                JsonNode jsonNode = jsonParser.readValue(value, JsonNode.class);
                return new Tuple2<>(jsonNode.get("ticker").toString(), 1);
            }).returns(Types.TUPLE(Types.STRING, Types.INT))
            .keyBy(v -> v.f0) // Logically partition the stream for each word
            .window(TumblingProcessingTimeWindows.of(Time.minutes(1)))
            .sum(1) // Count the appearances by ticker per partition
            .map(value -> value.f0 + " count: " + value.f1.toString() + "\n")
            .addSink(createS3SinkFromStaticConfig());
```

> **Note**
>
> Aplikasi ini menggunakan objek `StreamingFileSink` Flink untuk menulis ke Amazon S3. Untuk informasi lebih lanjut tentang`StreamingFileSink`, lihat StreamingFileSinkdi dokumentasi Apache Flink.

Ubah Kode Aplikasi

Di bagian ini, Anda mengubah kode aplikasi untuk menulis output ke bucket Amazon S3 Anda.

Perbarui baris berikut dengan nama pengguna Anda untuk menentukan lokasi output aplikasi:

```
private static final String s3SinkPath = "s3a://ka-app-code-<username>/data";
```

Kompilasi Kode Aplikasi

Untuk mengompilasi aplikasi, lakukan hal berikut:

1. Instal Java dan Maven jika Anda belum menginstalnya. Untuk informasi selengkapnya, lihat Prasyarat di tutorial Memulai (DataStream API).
2. Susun aplikasi dengan perintah berikut:

   ```
   mvn package -Dflink.version=1.15.3
   ```

Mengkompilasi aplikasi membuat file JAR aplikasi (`target/aws-kinesis-analytics-java-apps-1.0.jar`).

> **Note**
>
> Kode sumber yang disediakan bergantung pada pustaka dari Java 11.

Unggah Kode Java Streaming Apache Flink

Di bagian ini, Anda mengunggah kode aplikasi ke bucket Amazon S3 yang Anda buat di bagian Buat Sumber Daya Dependen.

1. Di konsol Amazon S3, pilih bucket ka-app-code -, navigasikan ke folder kode, dan pilih Unggah. <username>
2. Di langkah Pilih file, pilih Add files (Tambahkan berkas). Navigasikan ke file `aws-kinesis-analytics-java-apps-1.0.jar` yang Anda buat di langkah sebelumnya.
3. Anda tidak perlu mengubah pengaturan apa pun untuk objek, jadi pilih Upload (Unggah).

Kode aplikasi Anda sekarang disimpan di bucket Amazon S3 yang dapat diakses aplikasi Anda.

Buat dan Jalankan Managed Service untuk Apache Flink Application

Ikuti langkah-langkah ini untuk membuat, mengonfigurasi, memperbarui, dan menjalankan aplikasi menggunakan konsol.

Buat Aplikasi

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/flink
2. Pada dashboard Managed Service for Apache Flink, pilih Create Analytics Application.
3. Pada Layanan Terkelola untuk Apache Flink - Buat halaman aplikasi, berikan detail aplikasi sebagai berikut:
   - Untuk Application name (Nama aplikasi), masukkan **`MyApplication`**.
   - Untuk Runtime, pilih Apache Flink.
   - Biarkan versi pulldown sebagai Apache Flink versi 1.15.2 (Versi yang disarankan).
4. Untuk Access permissions (Izin akses), pilih Create / update IAM role **`kinesis-analytics-MyApplication-us-west-2`** (Buat/perbarui IAM role).
5. Pilih Create application (Buat aplikasi).

> **Note**
>
> Saat membuat aplikasi Managed Service for Apache Flink menggunakan konsol, Anda memiliki opsi untuk membuat peran dan kebijakan IAM untuk aplikasi Anda. Aplikasi Anda menggunakan peran dan kebijakan ini untuk mengakses sumber daya dependen. Sumber daya IAM ini diberi nama menggunakan nama aplikasi dan Wilayah sebagai berikut:
>
> - Untuk Application name (Nama aplikasi), masukkan **`MyApplication`**.
> - Untuk Runtime, pilih Apache Flink.
> - Tinggalkan versi sebagai Apache Flink versi 1.15.2 (Versi yang disarankan).

6. Untuk Access permissions (Izin akses), pilih Create / update IAM role **`kinesis-analytics-MyApplication-us-west-2`** (Buat/perbarui IAM role).
7. Pilih Create application (Buat aplikasi).

> **Note**
>
> Saat Anda membuat Layanan Terkelola untuk Apache Flink menggunakan konsol, Anda memiliki opsi untuk membuat peran dan kebijakan IAM untuk aplikasi Anda. Aplikasi Anda menggunakan peran dan kebijakan ini untuk mengakses sumber daya dependen. Sumber daya IAM ini diberi nama menggunakan nama aplikasi dan Wilayah sebagai berikut:
>
> - Kebijakan: `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Peran: `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Edit Kebijakan IAM

Edit kebijakan IAM untuk menambahkan izin mengakses Kinesis data stream.

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pilih Policies (Kebijakan). Pilih kebijakan **`kinesis-analytics-service-MyApplication-us-west-2`** yang dibuat konsol untuk Anda di bagian sebelumnya.
3. Di halaman Ringkasan, pilih Edit policy (Edit kebijakan). Pilih tab JSON.
4. Tambahkan bagian yang disorot dari contoh kebijakan berikut ke kebijakan. Ganti ID akun sampel (*`012345678901`*) dengan ID akun Anda. Ganti <username> dengan nama pengguna Anda.

```
{
            "Sid": "S3",
            "Effect": "Allow",
            "Action": [
                "s3:Abort*",
                "s3:DeleteObject*",
                "s3:GetObject*",
                "s3:GetBucket*",
                "s3:List*",
                "s3:ListBucket",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::ka-app-code-<username>",
                "arn:aws:s3:::ka-app-code-<username>/*"
            ]
        },
```

```
        {
            "Sid": "ListCloudwatchLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:region:account-id:log-group:*"
            ]
        },
        {
            "Sid": "ListCloudwatchLogStreams",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:region:account-id:log-group:%LOG_GROUP_PLACEHOLDER
%:log-stream:*"
            ]
        },
        {
            "Sid": "PutCloudwatchLogs",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:region:account-id:log-group:%LOG_GROUP_PLACEHOLDER
%:log-stream:%LOG_STREAM_PLACEHOLDER%"
            ]
        }
        ,
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },

    ]
}
```

Konfigurasikan Aplikasi

1. Pada MyApplicationhalaman, pilih Konfigurasi.
2. Di halaman Konfigurasikan aplikasi, berikan Code location (Lokasi kode):
   - Untuk Bucket Amazon S3, masukkan **`ka-app-code-<username>`**.
   - Untuk Jalur ke objek Amazon S3, masukkan **`code/aws-kinesis-analytics-java-apps-1.0.jar`**.
3. Di bawah Akses ke sumber daya aplikasi, untuk Access permissions (Izin akses), pilih Create / update IAM role **`kinesis-analytics-MyApplication-us-west-2`** (Pilih/perbarui IAM role).
4. Di bawah Pemantauan, pastikan Memantau tingkat metrik diatur ke Aplikasi.
5. Untuk CloudWatch logging, pilih kotak centang Aktifkan.
6. Pilih Perbarui.

> **Note**
>
> Saat Anda memilih untuk mengaktifkan CloudWatch logging, Managed Service for Apache Flink membuat grup log dan aliran log untuk Anda. Nama-nama sumber daya ini adalah sebagai berikut:
>
> - Grup log: `/aws/kinesis-analytics/MyApplication`
> - Aliran log: `kinesis-analytics-log-stream`
>
> Aliran log ini digunakan untuk memantau aplikasi. Ini bukan aliran log yang sama dengan yang digunakan aplikasi untuk mengirim hasil.

Jalankan Aplikasi

1. Pada MyApplicationhalaman, pilih Jalankan. Biarkan opsi Run without snapshot (Jalankan tanpa snapshot) dipilih, dan konfirmasikan tindakan.
2. Ketika aplikasi berjalan, refresh halaman. Konsol menunjukkan Grafik aplikasi.

Verifikasi Output Aplikasi

Di konsol Amazon S3, buka folder data di bucket S3 Anda.

Setelah beberapa menit, objek yang berisi data agregat dari aplikasi akan muncul.

> **Note**
>
> Agregrasi diaktifkan secara default di Flink. Untuk menonaktifkannya, gunakan yang berikut ini:
>
> ```
> sink.producer.aggregation-enabled' = 'false'
> ```

Opsional: Sesuaikan Sumber dan Sink

Di bagian ini, Anda menyesuaikan pengaturan pada objek sumber dan sink.

> **Note**
>
> Setelah mengubah bagian kode yang dijelaskan di bagian berikut, lakukan hal berikut untuk memuat ulang kode aplikasi:
>
> - Ulangi langkah-langkah di bagian the section called “Kompilasi Kode Aplikasi” untuk mengompilasi kode aplikasi yang diperbarui.
> - Ulangi langkah-langkah di bagian the section called “Unggah Kode Java Streaming Apache Flink” untuk mengunggah kode aplikasi yang diperbarui.
> - Di halaman aplikasi di konsol, pilih Configure (Konfigurasikan), lalu pilih Update (Perbarui) untuk memuat ulang kode aplikasi yang diperbarui ke dalam aplikasi Anda.

Bagian ini berisi bagian-bagian berikut:

- Konfigurasikan Partisi Data
- Konfigurasikan Frekuensi Baca
- Konfigurasikan Buffering Tulis

Konfigurasikan Partisi Data

Di bagian ini, Anda mengkonfigurasi nama folder yang dibuat sink file streaming di bucket S3. Anda melakukan ini dengan menambahkan pemberi tugas bucket ke sink file streaming.

Untuk menyesuaikan nama folder yang dibuat dalam bucket S3, lakukan hal berikut:

1. Tambahkan pernyataan impor berikut ke bagian depan file `S3StreamingSinkJob.java`:

```
import
 org.apache.flink.streaming.api.functions.sink.filesystem.rollingpolicies.DefaultRollingPol
import
 org.apache.flink.streaming.api.functions.sink.filesystem.bucketassigners.DateTimeBucketAss
```

2. Perbarui metode `createS3SinkFromStaticConfig()` dalam kode agar terlihat seperti berikut ini:

```
private static StreamingFileSink<String> createS3SinkFromStaticConfig() {

    final StreamingFileSink<String> sink = StreamingFileSink
        .forRowFormat(new Path(s3SinkPath), new
 SimpleStringEncoder<String>("UTF-8"))
        .withBucketAssigner(new DateTimeBucketAssigner("yyyy-MM-dd--HH"))
        .withRollingPolicy(DefaultRollingPolicy.create().build())
        .build();
    return sink;
}
```

Contoh kode sebelumnya menggunakan `DateTimeBucketAssigner` dengan format tanggal kustom untuk membuat folder dalam bucket S3. `DateTimeBucketAssigner` menggunakan waktu sistem saat ini untuk membuat nama bucket. Jika Anda ingin membuat pendaftar bucket khusus untuk menyesuaikan nama folder yang dibuat lebih lanjut, Anda dapat membuat kelas yang mengimplementasikannya. BucketAssigner Anda menerapkan logika kustom Anda menggunakan metode `getBucketId`.

Implementasi kustom dari `BucketAssigner` dapat menggunakan parameter Context untuk mendapatkan informasi selengkapnya tentang catatan untuk menentukan folder tujuannya.

Konfigurasikan Frekuensi Baca

Di bagian ini, Anda mengonfigurasi frekuensi membaca pada aliran sumber.

Konsumen Aliran Kinesis membaca dari sumber aliran lima kali per detik secara default. Frekuensi ini akan menyebabkan masalah jika ada lebih dari satu klien yang membaca dari aliran, atau jika aplikasi perlu mencoba lagi pembacaan catatan. Anda dapat menghindari masalah ini dengan mengatur frekuensi baca konsumen.

Untuk mengatur frekuensi baca konsumen Kinesis, Anda menetapkan pengaturan `SHARD_GETRECORDS_INTERVAL_MILLIS`.

Contoh kode berikut menetapkan pengaturan `SHARD_GETRECORDS_INTERVAL_MILLIS` ke satu detik:

```
kinesisConsumerConfig.setProperty(ConsumerConfigConstants.SHARD_GETRECORDS_INTERVAL_MILLIS,
 "1000");
```

Konfigurasikan Buffering Tulis

Di bagian ini, Anda mengonfigurasi frekuensi tulis dan pengaturan sink lainnya.

Secara default, aplikasi menulis ke bucket tujuan setiap menit. Anda dapat mengubah interval ini dan pengaturan lainnya dengan mengonfigurasi objek `DefaultRollingPolicy`.

> **Note**
>
> Sink file streaming Apache Flink menulis ke bucket output setiap kali aplikasi membuat titik pemeriksaan. Aplikasi ini membuat titik pemeriksaan setiap menit secara default. Untuk meningkatkan interval tulis sink S3, Anda juga harus meningkatkan interval titik pemeriksaan.

Untuk mengonfigurasi objek `DefaultRollingPolicy`, lakukan hal berikut:

1. Tingkatkan pengaturan `CheckpointInterval` aplikasi. Input berikut untuk UpdateApplicationtindakan menetapkan interval pos pemeriksaan menjadi 10 menit:

   ```
   {
      "ApplicationConfigurationUpdate": {
   ```

```
      "FlinkApplicationConfigurationUpdate": {
         "CheckpointConfigurationUpdate": {
            "ConfigurationTypeUpdate" : "CUSTOM",
            "CheckpointIntervalUpdate": 600000
         }
      }
   },
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 5
}
```

Untuk menggunakan kode sebelumnya, tentukan versi aplikasi saat ini. Anda dapat mengambil versi aplikasi dengan menggunakan ListApplicationstindakan.

2. Tambahkan pernyataan impor berikut ke bagian depan file `S3StreamingSinkJob.java`:

```
import java.util.concurrent.TimeUnit;
```

3. Perbarui metode `createS3SinkFromStaticConfig` dalam file `S3StreamingSinkJob.java` agar terlihat seperti berikut ini:

```
private static StreamingFileSink<String> createS3SinkFromStaticConfig() {

        final StreamingFileSink<String> sink = StreamingFileSink
                .forRowFormat(new Path(s3SinkPath), new
 SimpleStringEncoder<String>("UTF-8"))
                .withBucketAssigner(new DateTimeBucketAssigner("yyyy-MM-dd--HH"))
                .withRollingPolicy(
                        DefaultRollingPolicy.create()
                            .withRolloverInterval(TimeUnit.MINUTES.toMillis(8))
                            .withInactivityInterval(TimeUnit.MINUTES.toMillis(5))
                            .withMaxPartSize(1024 * 1024 * 1024)
                            .build())
                .build();
        return sink;
    }
```

Contoh kode sebelumnya menetapkan frekuensi tulis ke bucket Amazon S3 ke 8 menit.

Untuk informasi selengkapnya tentang sink file streaming Apache Flink, lihat Format yang Dienkode Baris di Dokumentasi Apache Flink.

Pembersihan Sumber Daya AWS

Bagian ini mencakup prosedur untuk membersihkan sumber daya AWS yang dibuat dalam tutorial Amazon S3.

Topik ini berisi bagian-bagian berikut:

- Hapus Layanan Terkelola Anda untuk Aplikasi Apache Flink
- Hapus Kinesis Data Stream Anda
- Hapus Objek dan Bucket Amazon S3 Anda
- Hapus Sumber Daya IAM Anda
- Hapus CloudWatch Sumber Daya Anda

Hapus Layanan Terkelola Anda untuk Aplikasi Apache Flink

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/flink
2. Di panel Managed Service for Apache Flink, pilih. MyApplication
3. Di halaman aplikasi, pilih Delete (Hapus), lalu konfirmasikan penghapusan.

Hapus Kinesis Data Stream Anda

1. Buka konsol Kinesis di https://console.aws.amazon.com/kinesis.
2. Di panel Kinesis Data Streams, pilih. ExampleInputStream
3. Pada ExampleInputStreamhalaman, pilih Hapus Kinesis Stream dan kemudian konfirmasikan penghapusan.

Hapus Objek dan Bucket Amazon S3 Anda

1. Buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.
2. Pilih ka-app-code- ember. <username>
3. Pilih Delete (Hapus), lalu masukkan nama bucket untuk mengonfirmasi penghapusan.

Hapus Sumber Daya IAM Anda

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di bilah navigasi, pilih Policies (Kebijakan).

3. Di kontrol filter, masukkan kinesis.
4. Pilih kebijakan kinesis-analytics-service- MyApplication -us-west-2.
5. Pilih Policy Actions (Tindakan Kebijakan), lalu pilih Delete (Hapus).
6. Di bilah navigasi, pilih Roles (Peran).
7. Pilih peran kinesis-analytics- -us-west-2. MyApplication
8. Pilih Delete role (Hapus peran), lalu konfirmasi penghapusan.

Hapus CloudWatch Sumber Daya Anda

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di bilah navigasi, pilih Logs.
3. Pilih grup log MyApplication/aws/kinesis-analytics/.
4. Pilih Delete Log Group (Hapus Grup Log), lalu konfirmasi penghapusan.

Tutorial: Menggunakan Managed Service untuk aplikasi Apache Flink untuk Mereplikasi Data dari Satu Topik di MSK Cluster ke Lainnya dalam VPC

Tutorial berikut menunjukkan cara membuat VPC Amazon dengan cluster MSK Amazon dan dua topik, dan cara membuat Layanan Terkelola untuk aplikasi Apache Flink yang membaca dari satu topik MSK Amazon dan menulis ke yang lain.

> **Note**
>
> Untuk menyiapkan prasyarat yang diperlukan untuk latihan ini, selesaikan latihan Memulai (DataStream API) terlebih dulu.

Tutorial ini berisi bagian-bagian berikut:

- Buat Amazon VPC dengan klaster Amazon MSK
- Buat Kode Aplikasi
- Unggah Kode Java Streaming Apache Flink
- Buat Aplikasi
- Konfigurasikan Aplikasi
- Jalankan Aplikasi

- Uji Aplikasi

Buat Amazon VPC dengan klaster Amazon MSK

Untuk membuat contoh klaster VPC dan Amazon MSK untuk mengakses dari aplikasi Managed Service for Apache Flink, ikuti tutorial Memulai Menggunakan Amazon MSK.

Saat menyelesaikan tutorial, perhatikan hal berikut:

- Pada Langkah 3: Buat Topik, ulangi `kafka-topics.sh --create` perintah untuk membuat topik tujuan bernama`AWSKafkaTutorialTopicDestination`:

```
bin/kafka-topics.sh --create --zookeeper ZooKeeperConnectionString --replication-
factor 3 --partitions 1 --topic AWSKafkaTutorialTopicDestination
```

- Catat daftar server bootstrap untuk klaster Anda. Anda bisa mendapatkan daftar server bootstrap dengan perintah berikut (ganti *`ClusterArn`*dengan ARN cluster MSK Anda):

```
aws kafka get-bootstrap-brokers --region us-west-2 --cluster-arn ClusterArn
{...
    "BootstrapBrokerStringTls": "b-2.awskafkatutorialcluste.t79r6y.c4.kafka.us-
west-2.amazonaws.com:9094,b-1.awskafkatutorialcluste.t79r6y.c4.kafka.us-
west-2.amazonaws.com:9094,b-3.awskafkatutorialcluste.t79r6y.c4.kafka.us-
west-2.amazonaws.com:9094"
}
```

- Saat mengikuti langkah-langkah dalam tutorial, pastikan untuk menggunakan Wilayah AWS yang Anda pilih dalam kode, perintah, dan entri konsol Anda.

Buat Kode Aplikasi

Di bagian ini, Anda akan mengunduh dan mengompilasi file JAR aplikasi. Kami merekomendasikan menggunakan Java 11.

Kode aplikasi Java untuk contoh ini tersedia dari GitHub. Untuk mengunduh kode aplikasi, lakukan hal berikut:

1. Instal klien Git jika Anda belum menginstalnya. Untuk informasi selengkapnya, lihat Menginstal Git.
2. Klon repositori jarak jauh dengan perintah berikut:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Kode aplikasi terletak di file `amazon-kinesis-data-analytics-java-examples/KafkaConnectors/KafkaGettingStartedJob.java`. Anda dapat memeriksa kode untuk membiasakan diri dengan struktur Layanan Terkelola untuk kode aplikasi Apache Flink.
4. Gunakan salah satu alat Maven baris perintah atau lingkungan pengembangan pilihan Anda untuk membuat file JAR. Untuk mengompilasi file JAR menggunakan alat Maven baris perintah, masukkan berikut ini:

   ```
   mvn package -Dflink.version=1.15.3
   ```

   Jika berhasil membangun, file berikut dibuat:

   ```
   target/KafkaGettingStartedJob-1.0.jar
   ```

   > **Note**
   >
   > Kode sumber yang disediakan bergantung pada pustaka dari Java 11. Jika Anda menggunakan lingkungan pengembangan,

Unggah Kode Java Streaming Apache Flink

Di bagian ini, Anda mengunggah kode aplikasi ke bucket Amazon S3 yang Anda buat di tutorial Memulai (DataStream API).

> **Note**
>
> Jika Anda menghapus bucket Amazon S3 dari tutorial Memulai, ikuti lagi langkah the section called “Unggah Kode Java Streaming Apache Flink”.

1. Di konsol Amazon S3, pilih bucket ka-app-code -, dan pilih Unggah. <username>
2. Di langkah Pilih file, pilih Add files (Tambahkan berkas). Navigasikan ke file `KafkaGettingStartedJob-1.0.jar` yang Anda buat di langkah sebelumnya.
3. Anda tidak perlu mengubah pengaturan apa pun untuk objek, jadi pilih Upload (Unggah).

Kode aplikasi Anda sekarang disimpan di bucket Amazon S3 yang dapat diakses aplikasi Anda.

Buat Aplikasi

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/flink
2. Pada dashboard Managed Service for Apache Flink, pilih Create Analytics Application.
3. Pada Layanan Terkelola untuk Apache Flink - Buat halaman aplikasi, berikan detail aplikasi sebagai berikut:
   - Untuk Application name (Nama aplikasi), masukkan **`MyApplication`**.
   - Untuk Runtime, pilih Apache Flink versi 1.15.2.
4. Untuk Access permissions (Izin akses), pilih Create / update IAM role **`kinesis-analytics-MyApplication-us-west-2`** (Buat/perbarui IAM role).
5. Pilih Create application (Buat aplikasi).

> **Note**
>
> Saat membuat aplikasi Managed Service for Apache Flink menggunakan konsol, Anda memiliki opsi untuk membuat peran dan kebijakan IAM untuk aplikasi Anda. Aplikasi Anda menggunakan peran dan kebijakan ini untuk mengakses sumber daya dependen. Sumber daya IAM ini diberi nama menggunakan nama aplikasi dan Wilayah sebagai berikut:
>
> - Kebijakan: `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Peran: `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Konfigurasikan Aplikasi

1. Pada MyApplicationhalaman, pilih Konfigurasi.
2. Di halaman Konfigurasikan aplikasi, berikan Code location (Lokasi kode):
   - Untuk Bucket Amazon S3, masukkan **`ka-app-code-`***`<username>`*.
   - Untuk Jalur ke objek Amazon S3, masukkan **`KafkaGettingStartedJob-1.0.jar`**.
3. Di bawah Akses ke sumber daya aplikasi, untuk Access permissions (Izin akses), pilih Create / update IAM role **`kinesis-analytics-MyApplication-us-west-2`** (Pilih/perbarui IAM role).

> **Note**
>
> Saat Anda menentukan sumber daya aplikasi menggunakan konsol (seperti CloudWatch Log atau VPC Amazon), konsol akan mengubah peran eksekusi aplikasi Anda untuk memberikan izin untuk mengakses sumber daya tersebut.

4. Di bawah Properties (Properti), pilih Add Group (Tambahkan Grup). Masukkan properti berikut:

ID Grup	Kunci	Nilai
KafkaSource	topik	AWSKafkaTutorialTopic
KafkaSource	bootstrap.servers	*Daftar server bootstrap yang Anda simpan sebelumnya*
KafkaSource	security.protocol	SSL
KafkaSource	ssl.truststore.location	/usr/lib/jvm/java-11-amazon-corretto/lib/keamanan/cacerts
KafkaSource	ssl.truststore.password	changeit

> **Note**
>
> ssl.truststore.password untuk sertifikat default adalah "changeit"; Anda tidak perlu mengubah nilai ini jika Anda menggunakan sertifikat default.

Pilih Add Group (Tambahkan Grup) lagi. Masukkan properti berikut:

ID Grup	Kunci	Nilai
KafkaSink	topik	AWSKafkaTutorialTopicDestination

ID Grup	Kunci	Nilai
KafkaSink	bootstrap.servers	*Daftar server bootstrap yang Anda simpan sebelumnya*
KafkaSink	security.protocol	SSL
KafkaSink	ssl.truststore.location	/usr/lib/jvm/java-11-amazon-corretto/lib/keamanan/cacerts
KafkaSink	ssl.truststore.password	changeit
KafkaSink	transaction.timeout.ms	1000

Kode aplikasi membaca properti aplikasi di atas untuk mengonfigurasi sumber dan sink yang digunakan untuk berinteraksi dengan VPC dan klaster Amazon MSK Anda. Untuk informasi selengkapnya tentang penggunaan runtime, lihat Properti Runtime.

5. Di bawah Snapshots, pilih Disable (Nonaktifkan). Tindakan ini akan memudahkan pembaruan aplikasi tanpa memuat data status aplikasi yang tidak valid.
6. Di bawah Pemantauan, pastikan Memantau tingkat metrik diatur ke Aplikasi.
7. Untuk CloudWatch logging, pilih kotak centang Aktifkan.
8. Di bagian Virtual Private Cloud (VPC), pilih VPC untuk dikaitkan dengan aplikasi Anda. Pilih subnet dan grup keamanan yang terkait dengan VPC Anda yang ingin digunakan aplikasi untuk mengakses sumber daya VPC.
9. Pilih Update (Perbarui).

> **Note**
>
> Saat Anda memilih untuk mengaktifkan CloudWatch logging, Managed Service for Apache Flink membuat grup log dan aliran log untuk Anda. Nama-nama sumber daya ini adalah sebagai berikut:
>
> - Grup log: `/aws/kinesis-analytics/MyApplication`
> - Aliran log: `kinesis-analytics-log-stream`

Aliran log ini digunakan untuk memantau aplikasi.

Jalankan Aplikasi

Grafik pekerjaan Flink dapat dilihat dengan menjalankan aplikasi, membuka dasbor Apache Flink, dan memilih pekerjaan Flink yang diinginkan.

Uji Aplikasi

Di bagian ini, Anda menulis catatan ke topik sumber. Aplikasi membaca catatan dari topik sumber dan menuliskannya ke topik tujuan. Anda memverifikasi aplikasi bekerja dengan menulis catatan ke topik sumber dan membaca catatan dari topik tujuan.

Untuk menulis dan membaca catatan dari topik, ikuti langkah-langkah di Langkah 6: Buat dan Gunakan Data di tutorial Memulai Menggunakan Amazon MSK.

Untuk membaca dari topik tujuan, gunakan nama topik tujuan bukan topik sumber dalam koneksi kedua Anda ke kluster:

```
bin/kafka-console-consumer.sh --bootstrap-server BootstrapBrokerString --
consumer.config client.properties --topic AWSKafkaTutorialTopicDestination --from-
beginning
```

Jika tidak ada catatan yang muncul dalam topik tujuan, lihat bagian Tidak Dapat Mengakses Sumber Daya dalam VPC di topik Pemecahan Masalah.

Contoh: Gunakan Konsumen EFO dengan Kinesis Data Stream

Dalam latihan ini, Anda membuat Layanan Terkelola untuk aplikasi Apache Flink yang membaca dari Kinesis Data Stream menggunakan konsumen Enhanced Fan-Out (EFO). Jika konsumen Kinesis menggunakan EFO, layanan Kinesis Data Streams memberikan bandwidth khusus miliknya sendiri, bukan meminta konsumen berbagi bandwidth aliran tetap dengan konsumen lain yang membaca dari aliran.

Untuk informasi selengkapnya tentang penggunaan EFO dengan konsumen Kinesis, lihat FLIP-128: Fan Out yang Disempurnakan untuk Konsumen Kinesis.

Aplikasi yang Anda buat dalam contoh ini menggunakan AWS Kinesis Connector (flink-connector-kinesis) 1.15.3.

> **Note**
>
> Untuk menyiapkan prasyarat yang diperlukan untuk latihan ini, selesaikan latihan Memulai (DataStream API) terlebih dulu.

Topik ini berisi bagian-bagian berikut:

- Buat Sumber Daya Dependen
- Tulis Catatan Sampel ke Aliran Input
- Unduh dan Periksa Kode Aplikasi
- Kompilasi Kode Aplikasi
- Unggah Kode Java Streaming Apache Flink
- Buat dan Jalankan Managed Service untuk Apache Flink Application
- Pembersihan Sumber Daya AWS

Buat Sumber Daya Dependen

Sebelum Anda membuat Layanan Terkelola untuk aplikasi Apache Flink untuk latihan ini, Anda membuat sumber daya dependen berikut:

- Dua Kinesis data streams (`ExampleInputStream` dan `ExampleOutputStream`)
- Bucket Amazon S3 untuk menyimpan kode aplikasi (`ka-app-code-<username>`)

Anda dapat membuat aliran Kinesis dan bucket Amazon S3 menggunakan konsol. Untuk petunjuk membuat sumber daya ini, lihat topik berikut:

- Membuat dan Memperbarui Aliran Data di Panduan Developer Amazon Kinesis Data Streams. Beri nama aliran data **`ExampleInputStream`** dan **`ExampleOutputStream`** Anda.
- Bagaimana Cara Membuat Bucket S3? di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon. Beri bucket Amazon S3 nama yang unik secara global dengan menambahkan nama login Anda, seperti **`ka-app-code-<username>`**.

Tulis Catatan Sampel ke Aliran Input

Di bagian ini, Anda menggunakan script Python untuk menulis catatan sampel ke aliran untuk diproses aplikasi.

> **Note**
>
> Bagian ini memerlukan AWS SDK for Python (Boto).

1. Buat file bernama `stock.py` dengan konten berikut:

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        'event_time': datetime.datetime.now().isoformat(),
        'ticker': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'price': round(random.random() * 100, 2)}


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(data),
            PartitionKey="partitionkey")


if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis', region_name='us-west-2'))
```

2. Jalankan skrip `stock.py`.

```
$ python stock.py
```

Biarkan skrip tetap berjalan saat menyelesaikan sisa tutorial.

Unduh dan Periksa Kode Aplikasi

Kode aplikasi Java untuk contoh ini tersedia dari GitHub. Untuk mengunduh kode aplikasi, lakukan hal berikut:

1. Instal klien Git jika Anda belum menginstalnya. Untuk informasi selengkapnya, lihat Menginstal Git.
2. Klon repositori jarak jauh dengan perintah berikut:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Buka direktori `amazon-kinesis-data-analytics-java-examples/EfoConsumer` tersebut.

Kode aplikasi terletak di file `EfoApplication.java`. Perhatikan hal tentang kode aplikasi berikut:

- Anda mengaktifkan konsumen EFO dengan mengatur parameter berikut pada konsumen Kinesis:
  - RECORD_PUBLISHER_TYPE: Atur parameter ini ke EFO untuk aplikasi Anda agar dapat menggunakan konsumen EFO untuk mengakses data Kinesis Data Stream.
  - EFO_CONSUMER_NAME: Atur parameter ini ke nilai string yang unik di antara konsumen aliran ini. Menggunakan kembali nama konsumen di Kinesis Data Stream yang sama akan menyebabkan konsumen sebelumnya yang menggunakan nama tersebut dihentikan.
- Contoh kode berikut menunjukkan cara menetapkan nilai ke properti konfigurasi konsumen untuk menggunakan konsumen EFO agar dapat membaca dari aliran sumber:

```
consumerConfig.putIfAbsent(RECORD_PUBLISHER_TYPE, "EFO");
consumerConfig.putIfAbsent(EFO_CONSUMER_NAME, "basic-efo-flink-app");
```

Kompilasi Kode Aplikasi

Untuk mengompilasi aplikasi, lakukan hal berikut:

1. Instal Java dan Maven jika Anda belum menginstalnya. Untuk informasi selengkapnya, lihat Prasyarat di tutorial Memulai (DataStream API).
2. Susun aplikasi dengan perintah berikut:

```
mvn package -Dflink.version=1.15.3
```

> **Note**
>
> Kode sumber yang disediakan bergantung pada pustaka dari Java 11.

Mengkompilasi aplikasi membuat file JAR aplikasi (`target/aws-kinesis-analytics-java-apps-1.0.jar`).

Unggah Kode Java Streaming Apache Flink

Di bagian ini, Anda mengunggah kode aplikasi ke bucket Amazon S3 yang Anda buat di bagian Buat Sumber Daya Dependen.

1. Di konsol Amazon S3, pilih bucket ka-app-code -, dan pilih Unggah. <username>
2. Di langkah Pilih file, pilih Add files (Tambahkan berkas). Navigasikan ke file `aws-kinesis-analytics-java-apps-1.0.jar` yang Anda buat di langkah sebelumnya.
3. Anda tidak perlu mengubah pengaturan apa pun untuk objek, jadi pilih Upload (Unggah).

Kode aplikasi Anda sekarang disimpan di bucket Amazon S3 yang dapat diakses aplikasi Anda.

Buat dan Jalankan Managed Service untuk Apache Flink Application

Ikuti langkah-langkah ini untuk membuat, mengonfigurasi, memperbarui, dan menjalankan aplikasi menggunakan konsol.

Buat Aplikasi

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/flink
2. Pada dashboard Managed Service for Apache Flink, pilih Create Analytics Application.
3. Pada Layanan Terkelola untuk Apache Flink - Buat halaman aplikasi, berikan detail aplikasi sebagai berikut:

- Untuk Application name (Nama aplikasi), masukkan **`MyApplication`**.
- Untuk Runtime, pilih Apache Flink.

> **Note**
>
> Managed Service untuk Apache Flink menggunakan Apache Flink versi 1.15.2.

- Biarkan versi pulldown sebagai Apache Flink versi 1.15.2 (Versi yang disarankan).

4. Untuk Access permissions (Izin akses), pilih Create / update IAM role **`kinesis-analytics-MyApplication-us-west-2`** (Buat/perbarui IAM role).
5. Pilih Create application (Buat aplikasi).

> **Note**
>
> Saat membuat aplikasi Managed Service for Apache Flink menggunakan konsol, Anda memiliki opsi untuk membuat peran dan kebijakan IAM untuk aplikasi Anda. Aplikasi Anda menggunakan peran dan kebijakan ini untuk mengakses sumber daya dependen. Sumber daya IAM ini diberi nama menggunakan nama aplikasi dan Wilayah sebagai berikut:
>
> - Kebijakan: `kinesis-analytics-service-MyApplication-us-west-2`
> - Peran: `kinesisanalytics-MyApplication-us-west-2`

Edit Kebijakan IAM

Edit kebijakan IAM untuk menambahkan izin mengakses Kinesis data streams.

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pilih Policies (Kebijakan). Pilih kebijakan **`kinesis-analytics-service-MyApplication-us-west-2`** yang dibuat konsol untuk Anda di bagian sebelumnya.
3. Di halaman Ringkasan, pilih Edit policy (Edit kebijakan). Pilih tab JSON.
4. Tambahkan bagian yang disorot dari contoh kebijakan berikut ke kebijakan. Ganti ID akun sampel (*`012345678901`*) dengan ID akun Anda.

> **Note**
>
> Izin ini memberi aplikasi kemampuan untuk mengakses konsumen EFO.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "logs:DescribeLogGroups",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*",
                "arn:aws:s3:::ka-app-code-<username>/aws-kinesis-analytics-java-
apps-1.0.jar"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": "logs:DescribeLogStreams",
            "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/
kinesis-analytics/MyApplication:log-stream:*"
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": "logs:PutLogEvents",
            "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/
kinesis-analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
        },
        {
            "Sid": "ListCloudwatchLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
```

```
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "AllStreams",
            "Effect": "Allow",
            "Action": [
                "kinesis:ListShards",
                "kinesis:ListStreamConsumers",
                "kinesis:DescribeStreamSummary"
            ],
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/*"
        },
        {
            "Sid": "Stream",
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStream",
                "kinesis:RegisterStreamConsumer",
                "kinesis:DeregisterStreamConsumer"
            ],
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        },
        {
            "Sid": "Consumer",
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStreamConsumer",
                "kinesis:SubscribeToShard"
            ],
            "Resource": [
                "arn:aws:kinesis:us-west-2:012345678901:stream/ExampleInputStream/
consumer/my-efo-flink-app",
                "arn:aws:kinesis:us-west-2:012345678901:stream/ExampleInputStream/
consumer/my-efo-flink-app:*"
```

```
            ]
        }
    ]
}
```

Konfigurasikan Aplikasi

1. Pada MyApplicationhalaman, pilih Konfigurasi.
2. Di halaman Konfigurasikan aplikasi, berikan Code location (Lokasi kode):
   - Untuk Bucket Amazon S3, masukkan **ka-app-code-*<username>***.
   - Untuk Jalur ke objek Amazon S3, masukkan **aws-kinesis-analytics-java-apps-1.0.jar**.
3. Di bawah Akses ke sumber daya aplikasi, untuk Access permissions (Izin akses), pilih Create / update IAM role **kinesis-analytics-MyApplication-us-west-2** (Pilih/perbarui IAM role).
4. Di bawah Properties (Properti), pilih Create group (Buat grup).
5. Masukkan properti dan nilai aplikasi berikut:

ID Grup	Kunci	Nilai
ConsumerConfigProperties	**flink.stream.recordpublisher**	**EFO**
ConsumerConfigProperties	**flink.stream.efo.consumername**	**basic-efo-flink-app**
ConsumerConfigProperties	**INPUT_STREAM**	**ExampleInputStream**
ConsumerConfigProperties	**flink.inputstream.initpos**	**LATEST**
ConsumerConfigProperties	**AWS_REGION**	**us-west-2**

6. Di bawah Properties (Properti), pilih Create group (Buat grup).

7. Masukkan properti dan nilai aplikasi berikut:

ID Grup	Kunci	Nilai
ProducerConfigProperties	**OUTPUT_STREAM**	**ExampleOutputStream**
ProducerConfigProperties	**AWS_REGION**	**us-west-2**
ProducerConfigProperties	**AggregationEnabled**	**false**

8. Di bawah Pemantauan, pastikan Memantau tingkat metrik diatur ke Aplikasi.
9. Untuk CloudWatch logging, pilih kotak centang Aktifkan.
10. Pilih Perbarui.

> **Note**
>
> Saat Anda memilih untuk mengaktifkan CloudWatch logging, Managed Service for Apache Flink membuat grup log dan aliran log untuk Anda. Nama-nama sumber daya ini adalah sebagai berikut:
>
> - Grup log: `/aws/kinesis-analytics/MyApplication`
> - Aliran log: `kinesis-analytics-log-stream`
>
> Aliran log ini digunakan untuk memantau aplikasi. Ini bukan aliran log yang sama dengan yang digunakan aplikasi untuk mengirim hasil.

Jalankan Aplikasi

Grafik pekerjaan Flink dapat dilihat dengan menjalankan aplikasi, membuka dasbor Apache Flink, dan memilih pekerjaan Flink yang diinginkan.

Anda dapat memeriksa metrik Managed Service for Apache Flink di CloudWatch konsol untuk memverifikasi bahwa aplikasi berfungsi.

Anda juga dapat memeriksa konsol Kinesis Data Streams, di tab Penggemar yang Ditingkatkan aliran data, untuk mengetahui nama konsumen Anda (). basic-efo-flink-app

Pembersihan Sumber Daya AWS

Bagian ini mencakup prosedur untuk membersihkan sumber daya AWS yang dibuat dalam tutorial Jendela efo.

Topik ini berisi bagian-bagian berikut:

- Hapus Layanan Terkelola Anda untuk Aplikasi Apache Flink
- Hapus Kinesis Data Streams Anda
- Hapus Objek dan Bucket Amazon S3 Anda
- Hapus Sumber Daya IAM Anda
- Hapus CloudWatch Sumber Daya Anda

Hapus Layanan Terkelola Anda untuk Aplikasi Apache Flink

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/flink
2. di panel Managed Service for Apache Flink, pilih. MyApplication
3. Di halaman aplikasi, pilih Delete (Hapus), lalu konfirmasikan penghapusan.

Hapus Kinesis Data Streams Anda

1. Buka konsol Kinesis di https://console.aws.amazon.com/kinesis.
2. Di panel Kinesis Data Streams, pilih. ExampleInputStream
3. Di ExampleInputStreamhalaman, pilih Hapus Stream Kinesis dan kemudian konfirmasikan penghapusan.
4. Di halaman Kinesis streams, pilih, pilih Tindakan ExampleOutputStream, pilih Hapus, lalu konfirmasikan penghapusan.

Hapus Objek dan Bucket Amazon S3 Anda

1. Buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.
2. Pilih ka-app-code- ember. <username>
3. Pilih Delete (Hapus), lalu masukkan nama bucket untuk mengonfirmasi penghapusan.

Hapus Sumber Daya IAM Anda

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di bilah navigasi, pilih Policies (Kebijakan).
3. Di kontrol filter, masukkan kinesis.
4. Pilih kebijakan kinesis-analytics-service- MyApplication -us-west-2.
5. Pilih Policy Actions (Tindakan Kebijakan), lalu pilih Delete (Hapus).
6. Di bilah navigasi, pilih Roles (Peran).
7. Pilih peran kinesis-analytics- -us-west-2. MyApplication
8. Pilih Delete role (Hapus peran), lalu konfirmasi penghapusan.

Hapus CloudWatch Sumber Daya Anda

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di bilah navigasi, pilih Logs.
3. Pilih grup log MyApplication/aws/kinesis-analytics/.
4. Pilih Delete Log Group (Hapus Grup Log), lalu konfirmasi penghapusan.

Contoh: Menulis ke Kinesis Data Firehose

Dalam latihan ini, Anda membuat Layanan Terkelola untuk aplikasi Apache Flink yang memiliki aliran data Kinesis sebagai sumber dan aliran Kinesis Data Firehose sebagai wastafel. Dengan menggunakan sink, Anda dapat memverifikasi output aplikasi di bucket Amazon S3.

> **Note**
>
> Untuk menyiapkan prasyarat yang diperlukan untuk latihan ini, selesaikan latihan Memulai (DataStream API) terlebih dulu.

Bagian ini berisi langkah-langkah berikut.

- Buat Sumber Daya Dependen
- Tulis Catatan Sampel ke Aliran Input
- Unduh dan Periksa Kode Java Streaming Apache Flink

- Kompilasi Kode Aplikasi
- Unggah Kode Java Streaming Apache Flink
- Buat dan Jalankan Managed Service untuk Apache Flink Application
- Pembersihan Sumber Daya AWS

Buat Sumber Daya Dependen

Sebelum Anda membuat Layanan Terkelola untuk Apache Flink untuk latihan ini, Anda membuat sumber daya dependen berikut:

- Kinesis data stream (`ExampleInputStream`)
- Aliran Kinesis Data Firehose dimana aplikasi menulis output `ExampleDeliveryStream` ke ().
- Bucket Amazon S3 untuk menyimpan kode aplikasi (`ka-app-code-<username>`)

Anda dapat membuat aliran Kinesis, bucket Amazon S3, dan aliran Kinesis Data Firehose menggunakan konsol. Untuk petunjuk membuat sumber daya ini, lihat topik berikut:

- Membuat dan Memperbarui Aliran Data di Panduan Developer Amazon Kinesis Data Streams. Beri nama aliran data **`ExampleInputStream`** Anda.
- Membuat Aliran Pengiriman Amazon Kinesis Data Firehose dalam Panduan Developer Amazon Kinesis Data Firehose.. Beri nama aliran Kinesis Data **`ExampleDeliveryStream`** Firehose Anda. Saat Anda membuat aliran Kinesis Data Firehose, buat juga peran tujuan S3 dan IAM aliran Kinesis Data Firehose.
- Bagaimana Cara Membuat Bucket S3? di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon. Beri bucket Amazon S3 nama yang unik secara global dengan menambahkan nama login Anda, seperti **`ka-app-code-<username>`**.

Tulis Catatan Sampel ke Aliran Input

Di bagian ini, Anda menggunakan script Python untuk menulis catatan sampel ke aliran untuk diproses aplikasi.

> **Note**
>
> Bagian ini memerlukan AWS SDK for Python (Boto).

1. Buat file bernama `stock.py` dengan konten berikut:

```
import datetime
   import json
   import random
   import boto3

   STREAM_NAME = "ExampleInputStream"


   def get_data():
       return {
           'event_time': datetime.datetime.now().isoformat(),
           'ticker': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
           'price': round(random.random() * 100, 2)}


   def generate(stream_name, kinesis_client):
       while True:
           data = get_data()
           print(data)
           kinesis_client.put_record(
               StreamName=stream_name,
               Data=json.dumps(data),
               PartitionKey="partitionkey")


   if __name__ == '__main__':
       generate(STREAM_NAME, boto3.client('kinesis', region_name='us-west-2'))
```

2. Jalankan skrip `stock.py`.

```
$ python stock.py
```

   Biarkan skrip tetap berjalan saat menyelesaikan sisa tutorial.

Unduh dan Periksa Kode Java Streaming Apache Flink

Kode aplikasi Java untuk contoh ini tersedia dari GitHub. Untuk mengunduh kode aplikasi, lakukan hal berikut:

1. Klon repositori jarak jauh dengan perintah berikut:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

2. Buka direktori `amazon-kinesis-data-analytics-java-examples/FirehoseSink` tersebut.

Kode aplikasi terletak di file `FirehoseSinkStreamingJob.java`. Perhatikan hal tentang kode aplikasi berikut:

- Aplikasi menggunakan sumber Kinesis untuk membaca dari aliran sumber. Cuplikan berikut ini membuat sumber Kinesis:

```
return env.addSource(new FlinkKinesisConsumer<>(inputStreamName,
                new SimpleStringSchema(), inputProperties));
```

- Aplikasi ini menggunakan sink Kinesis Data Firehose untuk menulis data ke aliran Kinesis Data Firehose. Cuplikan berikut membuat sink Kinesis Data Firehose:

```
private static KinesisFirehoseSink<String> createFirehoseSinkFromStaticConfig() {
        Properties sinkProperties = new Properties();
        sinkProperties.setProperty(AWS_REGION, region);

        return KinesisFirehoseSink.<String>builder()
                .setFirehoseClientProperties(sinkProperties)
                .setSerializationSchema(new SimpleStringSchema())
                .setDeliveryStreamName(outputDeliveryStreamName)
                .build();
    }
```

Kompilasi Kode Aplikasi

Untuk mengompilasi aplikasi, lakukan hal berikut:

1. Instal Java dan Maven jika Anda belum menginstalnya. Untuk informasi selengkapnya, lihat Prasyarat di tutorial Memulai (DataStream API).
2. Untuk menggunakan konektor Kinesis untuk aplikasi berikut, Anda perlu mengunduh, membangun, dan menginstal Apache Maven. Lihat informasi yang lebih lengkap di the section called “Menggunakan Konektor Apache Flink Kinesis Streams dengan versi Apache Flink sebelumnya”.

3. Susun aplikasi dengan perintah berikut:

```
mvn package -Dflink.version=1.15.3
```

> **Note**
>
> Kode sumber yang disediakan bergantung pada pustaka dari Java 11.

Mengkompilasi aplikasi membuat file JAR aplikasi (`target/aws-kinesis-analytics-java-apps-1.0.jar`).

Unggah Kode Java Streaming Apache Flink

Di bagian ini, Anda mengunggah kode aplikasi ke bucket Amazon S3 yang Anda buat di bagian Buat Sumber Daya Dependen.

Untuk mengunggah kode aplikasi

1. Buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.
2. Di konsol, pilih <username>ember ka-app-code-, lalu pilih Unggah.
3. Di langkah Pilih file, pilih Add files (Tambahkan berkas). Navigasikan ke file `java-getting-started-1.0.jar` yang Anda buat di langkah sebelumnya.
4. Anda tidak perlu mengubah pengaturan apa pun untuk objek, jadi pilih Upload (Unggah).

Kode aplikasi Anda sekarang disimpan di bucket Amazon S3 yang dapat diakses aplikasi Anda.

Buat dan Jalankan Managed Service untuk Apache Flink Application

Anda dapat membuat dan menjalankan Layanan Terkelola untuk aplikasi Apache Flink menggunakan konsol atau aplikasi. AWS CLI

> **Note**
>
> Saat Anda membuat aplikasi menggunakan konsol, sumber daya AWS Identity and Access Management (IAM) dan Amazon CloudWatch Logs dibuat untuk Anda. Saat Anda membuat aplikasi menggunakan AWS CLI, Anda membuat sumber daya ini secara terpisah.

Topik

Buat dan Jalankan Aplikasi (Konsol)

Ikuti langkah-langkah ini untuk membuat, mengonfigurasi, memperbarui, dan menjalankan aplikasi menggunakan konsol.

Buat Aplikasi

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/flink
2. Pada dashboard Managed Service for Apache Flink, pilih Create Analytics Application.
3. Pada Layanan Terkelola untuk Apache Flink - Buat halaman aplikasi, berikan detail aplikasi sebagai berikut:
   - Untuk Application name (Nama aplikasi), masukkan **`MyApplication`**.
   - Untuk Description (Deskripsi), masukkan **`My java test app`**.
   - Untuk Runtime, pilih Apache Flink.

     > **Note**
     >
     > Managed Service untuk Apache Flink menggunakan Apache Flink versi 1.15.2.

   - Biarkan versi pulldown sebagai Apache Flink versi 1.15.2 (Versi yang disarankan).
4. Untuk Access permissions (Izin akses), pilih Create / update IAM role **`kinesis-analytics-MyApplication-us-west-2`** (Buat/perbarui IAM role).
5. Pilih Create application (Buat aplikasi).

> **Note**
>
> Saat membuat aplikasi menggunakan konsol, Anda memiliki opsi untuk memiliki IAM role dan kebijakan IAM yang dibuat untuk aplikasi Anda. Aplikasi menggunakan peran dan kebijakan ini untuk mengakses sumber daya dependen. Sumber daya IAM ini diberi nama menggunakan nama aplikasi dan Wilayah sebagai berikut:
>
> - Kebijakan: `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*

- Peran: `kinesisanalytics-MyApplication-us-west-2`

Edit Kebijakan IAM

Edit kebijakan IAM untuk menambahkan izin untuk mengakses aliran data Kinesis dan aliran Firehose Data Kinesis.

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pilih Policies (Kebijakan). Pilih kebijakan **kinesis-analytics-service-MyApplication-us-west-2** yang dibuat konsol untuk Anda di bagian sebelumnya.
3. Di halaman Ringkasan, pilih Edit policy (Edit kebijakan). Pilih tab JSON.
4. Tambahkan bagian yang disorot dari contoh kebijakan berikut ke kebijakan. Ganti semua instans ID akun sampel (*012345678901*) dengan ID akun Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ka-app-code-username/java-getting-started-1.0.jar"
            ]
        },
        {
            "Sid": "DescribeLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
```

```
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
            ]
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteDeliveryStream",
            "Effect": "Allow",
            "Action": "firehose:*",
            "Resource": "arn:aws:firehose:us-west-2:012345678901:deliverystream/
ExampleDeliveryStream"
        }
    ]
}
```

Konfigurasikan Aplikasi

1. Pada MyApplicationhalaman, pilih Konfigurasi.
2. Di halaman Konfigurasikan aplikasi, berikan Code location (Lokasi kode):

- Untuk Bucket Amazon S3, masukkan **ka-app-code-*<username>***.
- Untuk Jalur ke objek Amazon S3, masukkan **java-getting-started-1.0.jar**.

3. Di bawah Akses ke sumber daya aplikasi, untuk Access permissions (Izin akses), pilih Create / update IAM role **kinesis-analytics-MyApplication-us-west-2** (Pilih/perbarui IAM role).
4. Di bawah Pemantauan, pastikan Memantau tingkat metrik diatur ke Aplikasi.
5. Untuk CloudWatch logging, pilih kotak centang Aktifkan.
6. Pilih Perbarui.

> **Note**
>
> Saat Anda memilih untuk mengaktifkan CloudWatch logging, Managed Service for Apache Flink membuat grup log dan aliran log untuk Anda. Nama-nama sumber daya ini adalah sebagai berikut:
>
> - Grup log: `/aws/kinesis-analytics/MyApplication`
> - Aliran log: `kinesis-analytics-log-stream`

Jalankan Aplikasi

Grafik pekerjaan Flink dapat dilihat dengan menjalankan aplikasi, membuka dasbor Apache Flink, dan memilih pekerjaan Flink yang diinginkan.

Hentikan Aplikasi

Pada MyApplicationhalaman, pilih Berhenti. Konfirmasikan tindakan.

Perbarui Aplikasi

Dengan menggunakan konsol, Anda dapat memperbarui pengaturan aplikasi seperti properti aplikasi, pengaturan pemantauan, dan lokasi atau nama file dari JAR aplikasi.

Pada MyApplicationhalaman, pilih Konfigurasi. Perbarui pengaturan aplikasi dan pilih Update (Perbarui).

> **Note**
>
> Untuk memperbarui kode aplikasi pada konsol, Anda harus mengubah nama objek JAR, menggunakan bucket S3 yang berbeda, atau menggunakan AWS CLI seperti yang dijelaskan di bagian the section called “Perbarui Kode Aplikasi”. Jika nama file atau bucket tidak berubah, kode aplikasi tidak dimuat ulang ketika Anda memilih Update (Perbarui) di halaman Konfigurasi.

Buat dan Jalankan Aplikasi (AWS CLI)

Di bagian ini, Anda menggunakan AWS CLI untuk membuat dan menjalankan aplikasi Managed Service for Apache Flink.

Membuat Kebijakan Izin

Pertama, Anda membuat kebijakan izin dengan dua pernyataan: satu yang memberikan izin untuk tindakan `read` di aliran sumber, dan lainnya yang memberikan izin untuk tindakan `write` di aliran sink. Anda selanjutnya melampirkan kebijakan ke IAM role (yang Anda buat di bagian berikutnya). Jadi, ketika Layanan Terkelola untuk Apache Flink mengasumsikan peran tersebut, layanan memiliki izin yang diperlukan untuk membaca dari aliran sumber dan menulis ke aliran wastafel.

Gunakan kode berikut untuk membuat kebijakan izin `AKReadSourceStreamWriteSinkStream`. Ganti *`username`* (nama pengguna) dengan nama pengguna yang akan Anda gunakan untuk membuat bucket Amazon S3 untuk menyimpan kode aplikasi. Ganti ID akun di Amazon Resource Names (ARN) (*`012345678901`*) dengan ID akun Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": ["arn:aws:s3:::ka-app-code-username",
                "arn:aws:s3:::ka-app-code-username/*"
            ]
```

```
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteDeliveryStream",
            "Effect": "Allow",
            "Action": "firehose:*",
            "Resource": "arn:aws:firehose:us-west-2:012345678901:deliverystream/
ExampleDeliveryStream"
        }
    ]
}
```

Untuk step-by-step petunjuk membuat kebijakan izin, lihat Tutorial: Membuat dan Melampirkan Kebijakan Terkelola Pelanggan Pertama Anda di Panduan Pengguna IAM.

> **Note**
>
> Untuk mengakses layanan Amazon lainnya, Anda dapat menggunakan AWS SDK for Java. Layanan Terkelola untuk Apache Flink secara otomatis menetapkan kredensil yang diperlukan oleh SDK ke peran IAM eksekusi layanan yang terkait dengan aplikasi Anda. Tidak ada langkah tambahan yang diperlukan.

Buat IAM Role

Di bagian ini, Anda membuat peran IAM yang dapat diasumsikan oleh aplikasi Managed Service for Apache Flink untuk membaca aliran sumber dan menulis ke aliran sink.

Layanan Terkelola untuk Apache Flink tidak dapat mengakses aliran Anda jika tidak memiliki izin. Anda memberikan izin ini melalui IAM role. Setiap IAM role memiliki dua kebijakan yang dilampirkan. Kebijakan kepercayaan memberikan izin Layanan Terkelola untuk Apache Flink untuk mengambil peran tersebut. Kebijakan izin menentukan apa yang dapat dilakukan Layanan Terkelola untuk Apache Flink setelah mengambil peran.

Anda melampirkan kebijakan izin yang Anda buat di bagian sebelumnya ke peran ini.

Untuk membuat IAM role

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Dalam panel navigasi, pilih Roles (Peran), Create role (Buat Peran).
3. Di bawah Pilih tipe identitas tepercaya, pilih Layanan AWS. Di bawah Pilih layanan yang akan menggunakan peran ini, pilih Kinesis. Di bawah Pilih kasus penggunaan Anda, pilih Analitik Kinesis.

   Pilih Selanjutnya: Izin.
4. Di halaman Lampirkan kebijakan izin, pilih Next: Review (Selanjutnya: Tinjau). Anda melampirkan kebijakan izin setelah Anda membuat peran tersebut.
5. Di halaman Buat peran, masukkan **`MF-stream-rw-role`** untuk Role name (Nama peran). Pilih Create role (Buat peran).

   Sekarang Anda sudah membuat IAM role baru yang disebut `MF-stream-rw-role`. Berikutnya, Anda memperbarui kebijakan kepercayaan dan izin untuk peran.
6. Lampirkan kebijakan izin ke peran tersebut.

   > **Note**
   >
   > Untuk latihan ini, Managed Service for Apache Flink mengasumsikan peran ini untuk membaca data dari aliran data Kinesis (sumber) dan menulis output ke aliran data Kinesis lain. Jadi, Anda melampirkan kebijakan yang Anda buat di langkah sebelumnya, the section called “Membuat Kebijakan Izin”.

   a. Di halaman Ringkasan, pilih tab Permissions (Izin).

   b. Pilih Attach Policies (Lampirkan Kebijakan).

   c. Di kotak pencarian, masukkan **`AKReadSourceStreamWriteSinkStream`** (kebijakan yang Anda buat bagian sebelumnya).

   d. Pilih ReadSourceStreamWriteSinkStream kebijakan AK, dan pilih Lampirkan kebijakan.

Anda sekarang telah membuat peran eksekusi layanan yang akan digunakan aplikasi Anda untuk mengakses sumber daya. Buat catatan ARN peran baru.

Untuk step-by-step petunjuk cara membuat peran, lihat Membuat Peran IAM (Konsol) di Panduan Pengguna IAM.

Buat Managed Service untuk Apache Flink Application

1. Simpan kode JSON berikut ke file bernama `create_request.json`. Ganti ARN peran sampel dengan ARN untuk peran yang Anda buat sebelumnya. Ganti sufiks ARN bucket dengan sufiks yang Anda pilih di bagian the section called “Buat Sumber Daya Dependen” (`ka-app-code-<username>`.) Ganti ID akun sampel (*012345678901*) di peran eksekusi layanan dengan ID akun Anda.

```
{
    "ApplicationName": "test",
    "ApplicationDescription": "my java test app",
    "RuntimeEnvironment": "FLINK-1_15",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-stream-rw-role",
    "ApplicationConfiguration": {
        "ApplicationCodeConfiguration": {
            "CodeContent": {
                "S3ContentLocation": {
                    "BucketARN": "arn:aws:s3:::ka-app-code-username",
                    "FileKey": "java-getting-started-1.0.jar"
                }
            },
            "CodeContentType": "ZIPFILE"
        }
      }
    }
}
```

2. Jalankan tindakan `CreateApplication` dengan permintaan sebelumnya untuk membuat aplikasi:

```
aws kinesisanalyticsv2 create-application --cli-input-json file://
create_request.json
```

Aplikasi ini sekarang dibuat. Anda memulai aplikasi di langkah berikutnya.

Mulai Aplikasi

Di bagian ini, Anda menggunakan tindakan `StartApplication` untuk memulai aplikasi.

Untuk memulai aplikasi

1. Simpan kode JSON berikut ke file bernama `start_request.json`.

```
{
    "ApplicationName": "test",
    "RunConfiguration": {
        "ApplicationRestoreConfiguration": {
         "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"
         }
    }
}
```

2. Jalankan tindakan `StartApplication` dengan permintaan sebelumnya untuk memulai aplikasi:

```
aws kinesisanalyticsv2 start-application --cli-input-json file://start_request.json
```

Aplikasi sekarang berjalan. Anda dapat memeriksa metrik Layanan Terkelola untuk Apache Flink di CloudWatch konsol Amazon untuk memverifikasi bahwa aplikasi berfungsi.

Hentikan Aplikasi

Di bagian ini, Anda menggunakan tindakan `StopApplication` untuk menghentikan aplikasi.

Untuk menghentikan aplikasi

1. Simpan kode JSON berikut ke file bernama `stop_request.json`.

```
{
    "ApplicationName": "test"
}
```

2. Jalankan tindakan `StopApplication` dengan permintaan berikut untuk menghentikan aplikasi:

```
aws kinesisanalyticsv2 stop-application --cli-input-json file://stop_request.json
```

Aplikasi sekarang dihentikan.

Tambahkan Opsi CloudWatch Logging

Anda dapat menggunakan AWS CLI untuk menambahkan aliran CloudWatch log Amazon ke aplikasi Anda. Untuk informasi tentang menggunakan CloudWatch Log dengan aplikasi Anda, lihatthe section called “Menyiapkan Pencatatan”.

Perbarui Kode Aplikasi

Ketika Anda perlu memperbarui kode aplikasi Anda dengan versi baru paket kode Anda, Anda menggunakan tindakan `UpdateApplication` AWS CLI.

Untuk menggunakan AWS CLI, hapus paket kode Anda sebelumnya dari bucket Amazon S3 Anda, unggah versi baru, dan panggil `UpdateApplication`, yang menentukan bucket Amazon S3 dan nama objek yang sama.

Permintaan sampel berikut untuk tindakan `UpdateApplication` memuat ulang kode aplikasi dan memulai ulang aplikasi. Perbarui `CurrentApplicationVersionId` ke versi aplikasi saat ini. Anda dapat memeriksa versi aplikasi saat ini menggunakan tindakan `ListApplications` atau `DescribeApplication`. Perbarui sufiks nama bucket (<*username*>) dengan sufiks yang Anda pilih di bagian the section called “Buat Sumber Daya Dependen”.

```
{
    "ApplicationName": "test",
    "CurrentApplicationVersionId": 1,
    "ApplicationConfigurationUpdate": {
        "ApplicationCodeConfigurationUpdate": {
            "CodeContentUpdate": {
                "S3ContentLocationUpdate": {
                    "BucketARNUpdate": "arn:aws:s3:::ka-app-code-username",
                    "FileKeyUpdate": "java-getting-started-1.0.jar"
                }
            }
        }
    }
}
```

Pembersihan Sumber Daya AWS

Bagian ini mencakup prosedur untuk membersihkan sumber daya AWS yang dibuat dalam tutorial Memulai.

Topik ini berisi bagian-bagian berikut:

- Hapus Layanan Terkelola Anda untuk Aplikasi Apache Flink
- Hapus Kinesis Data Stream Anda
- Hapus aliran Firehose Data Kinesis Anda
- Hapus Objek dan Bucket Amazon S3 Anda
- Hapus Sumber Daya IAM Anda
- Hapus CloudWatch Sumber Daya Anda

Hapus Layanan Terkelola Anda untuk Aplikasi Apache Flink

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/flink
2. Di panel Managed Service for Apache Flink, pilih. MyApplication
3. Pilih Configure (Konfigurasikan).
4. Di bagian Snapshots, pilih Disable (Nonaktifkan), lalu pilih Update (Perbarui).
5. Di halaman aplikasi, pilih Delete (Hapus), lalu konfirmasikan penghapusan.

Hapus Kinesis Data Stream Anda

1. Buka konsol Kinesis di https://console.aws.amazon.com/kinesis.
2. Di panel Kinesis Data Streams, pilih. ExampleInputStream
3. Di ExampleInputStreamhalaman, pilih Hapus Stream Kinesis dan kemudian konfirmasikan penghapusan.

Hapus aliran Firehose Data Kinesis Anda

1. Buka konsol Kinesis di https://console.aws.amazon.com/kinesis.
2. Di panel Kinesis Data Firehose, pilih. ExampleDeliveryStream
3. Di ExampleDeliveryStreamhalaman, pilih Hapus aliran Firehose Data Kinesis dan kemudian konfirmasikan penghapusan.

Hapus Objek dan Bucket Amazon S3 Anda

1. Buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.

2. Pilih ka-app-code- ember. <username>
3. Pilih Delete (Hapus), lalu masukkan nama bucket untuk mengonfirmasi penghapusan.
4. Jika Anda membuat bucket Amazon S3 untuk tujuan aliran Kinesis Data Firehose, hapus juga bucket tersebut.

Hapus Sumber Daya IAM Anda

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di bilah navigasi, pilih Policies (Kebijakan).
3. Di kontrol filter, masukkan kinesis.
4. Pilih kebijakan kinesis-analytics-service- MyApplication -us-west-2.
5. Pilih Policy Actions (Tindakan Kebijakan), lalu pilih Delete (Hapus).
6. Jika Anda membuat kebijakan baru untuk aliran Firehose Data Kinesis, hapus kebijakan tersebut juga.
7. Di bilah navigasi, pilih Roles (Peran).
8. Pilih peran kinesis-analytics- -us-west-2. MyApplication
9. Pilih Delete role (Hapus peran), lalu konfirmasi penghapusan.
10. Jika Anda membuat peran baru untuk aliran Kinesis Data Firehose, hapus peran tersebut juga.

Hapus CloudWatch Sumber Daya Anda

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di bilah navigasi, pilih Logs.
3. Pilih grup log MyApplication/aws/kinesis-analytics/.
4. Pilih Delete Log Group (Hapus Grup Log), lalu konfirmasi penghapusan.

Contoh: Baca dari Aliran Kinesis di Akun Berbeda.

Contoh ini menunjukkan cara membuat Layanan Terkelola untuk aplikasi Apache Flink yang membaca data dari aliran Kinesis di akun yang berbeda. Dalam contoh ini, Anda akan menggunakan satu akun untuk sumber Kinesis stream, dan akun kedua untuk Managed Service untuk aplikasi Apache Flink dan tenggelam Kinesis stream.

Topik ini berisi bagian-bagian berikut:

Prasyarat

- Dalam tutorial ini, Anda mengubah contoh Memulai untuk membaca data dari aliran Kinesis di akun yang berbeda. Selesaikan tutorial Memulai (DataStream API) sebelum melanjutkan.
- Anda memerlukan dua akun AWS untuk menyelesaikan tutorial ini: satu untuk aliran sumber, dan satu untuk aplikasi serta aliran sink. Gunakan akun AWS yang Anda gunakan untuk tutorial Memulai untuk aliran aplikasi dan sink. Gunakan akun AWS yang berbeda untuk aliran sumber.

Pengaturan

Anda akan mengakses dua akun AWS Anda menggunakan profil yang diberi nama. Ubah kredensial AWS dan file konfigurasi Anda untuk menyertakan dua profil yang berisi wilayah dan informasi koneksi untuk dua akun Anda.

File kredensial contoh berikut berisi dua profil yang diberi nama, `ka-source-stream-account-profile` dan `ka-sink-stream-account-profile`. Gunakan akun yang Anda gunakan untuk tutorial Memulai untuk akun aliran sink.

```
[ka-source-stream-account-profile]
aws_access_key_id=AKIAIOSFODNN7EXAMPLE
aws_secret_access_key=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

[ka-sink-stream-account-profile]
aws_access_key_id=AKIAI44QH8DHBEXAMPLE
aws_secret_access_key=je7MtGbClwBF/2Zp9Utk/h3yCo8nvbEXAMPLEKEY
```

File konfigurasi contoh berikut berisi profil bernama sama dengan informasi wilayah dan format output.

```
[profile ka-source-stream-account-profile]
region=us-west-2
output=json

[profile ka-sink-stream-account-profile]
region=us-west-2
output=json
```

> **Note**
>
> Tutorial ini tidak menggunakan `ka-sink-stream-account-profile`. Ini disertakan sebagai contoh cara mengakses dua akun AWS yang berbeda menggunakan profil.

Untuk informasi selengkapnya tentang menggunakan profil bernama dengan AWS CLI, lihat Profil Bernama di dokumentasi AWS Command Line Interface.

Buat Aliran Kinesis Sumber

Di bagian ini, Anda akan membuat aliran Kinesis di akun sumber.

Masukkan perintah berikut untuk membuat aliran Kinesis yang akan digunakan aplikasi untuk input. Perhatikan bahwa parameter `--profile` menentukan profil akun yang akan digunakan.

```
$ aws kinesis create-stream \
--stream-name SourceAccountExampleInputStream \
--shard-count 1 \
--profile ka-source-stream-account-profile
```

Buat dan Perbarui IAM Role dan Kebijakan IAM

Untuk mengizinkan akses objek di seluruh akun AWS Anda, Anda membuat IAM role dan kebijakan IAM di akun sumber. Selanjutnya, Anda mengubah kebijakan IAM di akun sink. Untuk informasi tentang membuat IAM role dan kebijakan IAM, lihat topik berikut di bagian Panduan Pengguna AWS Identity and Access Management:

- Membuat Peran IAM
- Membuat Kebijakan IAM

Peran dan Kebijakan Akun Sink

1. Edit kebijakan `kinesis-analytics-service-MyApplication-us-west-2` dari tutorial Memulai. Kebijakan ini memungkinkan peran dalam akun sumber diasumsikan agar dapat membaca aliran sumber.

> **Note**
>
> Saat Anda menggunakan konsol untuk membuat aplikasi Anda, konsol membuat kebijakan yang disebut `kinesis-analytics-service-<application name>-<application region>`, dan peran yang disebut `kinesisanalytics-<application name>-<application region>`.

Tambahkan bagian yang disorot di bawah ini ke kebijakan. Ganti ID akun sampel (*SOURCE01234567*) dengan ID akun yang akan Anda gunakan untuk aliran sumber.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AssumeRoleInSourceAccount",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::SOURCE01234567:role/KA-Source-Stream-Role"
        },
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ka-app-code-username/aws-kinesis-analytics-java-
apps-1.0.jar"
            ]
        },
        {
            "Sid": "ListCloudwatchLogGroups",
            "Effect": "Allow",
```

```
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:SINK012345678:log-group:*"
            ]
        },
        {
            "Sid": "ListCloudwatchLogStreams",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:SINK012345678:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
            ]
        },
        {
            "Sid": "PutCloudwatchLogs",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:SINK012345678:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
            ]
        }
    ]
}
```

2. Buka peran `kinesis-analytics-MyApplication-us-west-2`, dan buat catatan Amazon Resource Name (ARN). Anda akan membutuhkannya di bagian selanjutnya. ARN peran terlihat seperti berikut.

```
arn:aws:iam::SINK012345678:role/service-role/kinesis-analytics-MyApplication-us-
west-2
```

Peran dan Kebijakan Akun Sumber

1. Buat kebijakan di akun sumber yang disebut `KA-Source-Stream-Policy`. Gunakan JSON berikut untuk kebijakan. Ganti nomor akun sampel dengan nomor akun dari akun sumber.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStream",
                "kinesis:GetRecords",
                "kinesis:GetShardIterator",
                "kinesis:ListShards"
            ],
            "Resource":
               "arn:aws:kinesis:us-west-2:SOURCE123456784:stream/
SourceAccountExampleInputStream"
        }
    ]
}
```

2. Buat peran di akun sumber yang disebut `MF-Source-Stream-Role`. Lakukan hal berikut untuk membuat peran menggunakan kasus penggunaan Managed Flink:

    1. Di Konsol Manajemen IAM, pilih Create Role (Buat Peran).
    2. Di halaman Buat Peran, pilih Layanan AWS. Dalam daftar layanan, pilih Kinesis.
    3. Di bagian Pilih kasus penggunaan Anda, pilih Layanan Terkelola untuk Apache Flink.
    4. Pilih Selanjutnya: Izin.
    5. Tambahkan kebijakan izin `KA-Source-Stream-Policy` yang Anda buat di langkah sebelumnya. Pilih Next:Tags.
    6. Pilih Next: Review (Selanjutnya: Tinjauan).
    7. Beri nama peran `KA-Source-Stream-Role`. Aplikasi Anda akan menggunakan peran ini untuk mengakses aliran sumber.

3. Tambahkan ARN `kinesis-analytics-MyApplication-us-west-2` dari akun sink ke hubungan kepercayaan dari peran `KA-Source-Stream-Role` dalam akun sumber:

1. Buka `KA-Source-Stream-Role` di konsol IAM.
2. Pilih tab Trust Relationships (Hubungan Kepercayaan).
3. Pilih Edit trust relationship (Edit Hubungan Kepercayaan).
4. Gunakan kode berikut untuk hubungan kepercayaan. Ganti ID akun sampel (*SINK012345678*) dengan ID akun sink Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Principal": {
          "AWS": "arn:aws:iam::SINK012345678:role/service-role/kinesis-analytics-
MyApplication-us-west-2"
        },
        "Action": "sts:AssumeRole"
      }
    ]
}
```

Perbarui Skrip Python

Di bagian ini, Anda memperbarui skrip Python yang menghasilkan data sampel untuk menggunakan profil akun sumber.

Perbarui skrip `stock.py` dengan perubahan yang disorot berikut.

```
import json
import boto3
import random
import datetime
import os

os.environ['AWS_PROFILE'] ='ka-source-stream-account-profile'
os.environ['AWS_DEFAULT_REGION'] = 'us-west-2'


kinesis = boto3.client('kinesis')
def getReferrer():
    data = {}
```

```
    now = datetime.datetime.now()
    str_now = now.isoformat()
    data['event_time'] = str_now
    data['ticker'] = random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV'])
    price = random.random() * 100
    data['price'] = round(price, 2)
    return data

while True:
        data = json.dumps(getReferrer())
        print(data)
        kinesis.put_record(
                StreamName="SourceAccountExampleInputStream",
                Data=data,
                PartitionKey="partitionkey")
```

Perbarui Aplikasi Java

Di bagian ini, Anda memperbarui kode aplikasi Java untuk mengasumsikan peran akun sumber saat membaca dari aliran sumber.

Buat perubahan berikut ke file `BasicStreamingJob.java`. Ganti nomor akun sumber contoh (*SOURCE01234567*) dengan nomor akun sumber Anda.

```
package com.amazonaws.services.managed-flink;

import com.amazonaws.services.managed-flink.runtime.KinesisAnalyticsRuntime;
import org.apache.flink.api.common.serialization.SimpleStringSchema;
import org.apache.flink.streaming.api.datastream.DataStream;
import org.apache.flink.streaming.api.environment.StreamExecutionEnvironment;
import org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer;
import org.apache.flink.streaming.connectors.kinesis.FlinkKinesisProducer;
import org.apache.flink.streaming.connectors.kinesis.config.ConsumerConfigConstants;
import org.apache.flink.streaming.connectors.kinesis.config.AWSConfigConstants;

import java.io.IOException;
import java.util.Map;
import java.util.Properties;

 /**
 * A basic Managed Service for Apache Flink for Java application with Kinesis data
 streams
 * as source and sink.
```

```
 */
public class BasicStreamingJob {
    private static final String region = "us-west-2";
    private static final String inputStreamName = "SourceAccountExampleInputStream";
    private static final String outputStreamName = ExampleOutputStream;
    private static final String roleArn = "arn:aws:iam::SOURCE01234567:role/KA-Source-
Stream-Role";
    private static final String roleSessionName = "ksassumedrolesession";

    private static DataStream<String>
 createSourceFromStaticConfig(StreamExecutionEnvironment env) {
        Properties inputProperties = new Properties();
        inputProperties.setProperty(AWSConfigConstants.AWS_CREDENTIALS_PROVIDER,
 "ASSUME_ROLE");
        inputProperties.setProperty(AWSConfigConstants.AWS_ROLE_ARN, roleArn);
        inputProperties.setProperty(AWSConfigConstants.AWS_ROLE_SESSION_NAME,
 roleSessionName);
        inputProperties.setProperty(ConsumerConfigConstants.AWS_REGION, region);
        inputProperties.setProperty(ConsumerConfigConstants.STREAM_INITIAL_POSITION,
 "LATEST");

        return env.addSource(new FlinkKinesisConsumer<>(inputStreamName, new
 SimpleStringSchema(), inputProperties));
    }

    private static KinesisStreamsSink<String> createSinkFromStaticConfig() {
        Properties outputProperties = new Properties();
        outputProperties.setProperty(AWSConfigConstants.AWS_REGION, region);

        return KinesisStreamsSink.<String>builder()
                .setKinesisClientProperties(outputProperties)
                .setSerializationSchema(new SimpleStringSchema())
                .setStreamName(outputProperties.getProperty("OUTPUT_STREAM",
 "ExampleOutputStream"))
                .setPartitionKeyGenerator(element ->
 String.valueOf(element.hashCode()))
                .build();
    }

    public static void main(String[] args) throws Exception {
        // set up the streaming execution environment
        final StreamExecutionEnvironment env =
 StreamExecutionEnvironment.getExecutionEnvironment();
```

```
        DataStream<String> input = createSourceFromStaticConfig(env);

        input.addSink(createSinkFromStaticConfig());

        env.execute("Flink Streaming Java API Skeleton");
    }
}
```

Bangun, Unggah, dan Jalankan Aplikasi

Lakukan hal berikut untuk memperbarui dan menjalankan aplikasi:

1. Bangun lagi aplikasi dengan menjalankan perintah berikut di direktori dengan file `pom.xml`.

   ```
   mvn package -Dflink.version=1.15.3
   ```

2. Hapus file JAR sebelumnya dari bucket Amazon Simple Storage Service (Amazon S3) Anda, lalu unggah file `aws-kinesis-analytics-java-apps-1.0.jar` baru ke bucket S3.
3. Di halaman aplikasi di Managed Service for Apache Flink console, pilih Configure, Update untuk memuat ulang file JAR aplikasi.
4. Jalankan skrip `stock.py` untuk mengirim data ke aliran sumber.

   ```
   python stock.py
   ```

Aplikasi sekarang membaca data dari aliran Kinesis di akun lainnya.

Anda dapat memverifikasi bahwa aplikasi berfungsi dengan memeriksa metrik `PutRecords.Bytes` dari aliran `ExampleOutputStream`. Jika ada aktivitas dalam aliran output, aplikasi berfungsi dengan baik.

Tutorial: Menggunakan Truststore Kustom dengan Amazon MSK

API sumber data saat ini

Jika Anda menggunakan API sumber data saat ini, aplikasi Anda dapat memanfaatkan utilitas Penyedia Konfigurasi MSK yang dijelaskan di sini. Ini memungkinkan KafkaSource fungsi Anda untuk mengakses keystore dan truststore Anda untuk TLS bersama di Amazon S3.

```
...
```

```
// define names of config providers:
builder.setProperty("config.providers", "secretsmanager,s3import");

// provide implementation classes for each provider:
builder.setProperty("config.providers.secretsmanager.class",
 "com.amazonaws.kafka.config.providers.SecretsManagerConfigProvider");
builder.setProperty("config.providers.s3import.class",
 "com.amazonaws.kafka.config.providers.S3ImportConfigProvider");

String region = appProperties.get(Helpers.S3_BUCKET_REGION_KEY).toString();
String keystoreS3Bucket = appProperties.get(Helpers.KEYSTORE_S3_BUCKET_KEY).toString();
String keystoreS3Path = appProperties.get(Helpers.KEYSTORE_S3_PATH_KEY).toString();
String truststoreS3Bucket =
 appProperties.get(Helpers.TRUSTSTORE_S3_BUCKET_KEY).toString();
String truststoreS3Path = appProperties.get(Helpers.TRUSTSTORE_S3_PATH_KEY).toString();
String keystorePassSecret =
 appProperties.get(Helpers.KEYSTORE_PASS_SECRET_KEY).toString();
String keystorePassSecretField =
 appProperties.get(Helpers.KEYSTORE_PASS_SECRET_FIELD_KEY).toString();

// region, etc..
builder.setProperty("config.providers.s3import.param.region", region);

// properties
builder.setProperty("ssl.truststore.location", "${s3import:" + region + ":" +
 truststoreS3Bucket + "/" + truststoreS3Path + "}");
builder.setProperty("ssl.keystore.type", "PKCS12");
builder.setProperty("ssl.keystore.location", "${s3import:" + region + ":" +
 keystoreS3Bucket + "/" + keystoreS3Path + "}");
builder.setProperty("ssl.keystore.password", "${secretsmanager:" + keystorePassSecret +
 ":" + keystorePassSecretField + "}");
builder.setProperty("ssl.key.password", "${secretsmanager:" + keystorePassSecret + ":"
 + keystorePassSecretField + "}");
...
```

Detail lebih lanjut dan panduan dapat ditemukan di sini.

API Legacy SourceFunction

Jika Anda menggunakan SourceFunction API lama, aplikasi Anda akan menggunakan skema serialisasi dan deserialisasi khusus yang mengganti `open` metode untuk memuat truststore kustom. Hal ini membuat truststore tersedia untuk aplikasi setelah aplikasi restart atau menggantikan thread.

Truststore kustom diambil dan disimpan menggunakan kode berikut:

```
public static void initializeKafkaTruststore() {
    ClassLoader classLoader = Thread.currentThread().getContextClassLoader();
    URL inputUrl = classLoader.getResource("kafka.client.truststore.jks");
    File dest = new File("/tmp/kafka.client.truststore.jks");

    try {
        FileUtils.copyURLToFile(inputUrl, dest);
    } catch (Exception ex) {
        throw new FlinkRuntimeException("Failed to initialize Kakfa truststore", ex);
    }
}
```

> **Note**
>
> Apache Flink mengharuskan truststore dalam format JKS.

> **Note**
>
> Untuk menyiapkan prasyarat yang diperlukan untuk latihan ini, selesaikan latihan Memulai (DataStream API) terlebih dulu.

Tutorial berikut menunjukkan cara menghubungkan dengan aman (enkripsi dalam perjalanan) ke Kafka Cluster yang menggunakan sertifikat server yang dikeluarkan oleh Certificate Authority (CA) kustom, pribadi, atau bahkan yang di-host sendiri.

Untuk menghubungkan Klien Kafka dengan aman melalui TLS ke Kafka Cluster, Klien Kafka (seperti contoh Aplikasi Flink) harus mempercayai rantai kepercayaan lengkap yang diberikan oleh sertifikat server Kafka Cluster (dari CA Penerbitan hingga CA Tingkat Root). Sebagai contoh untuk Truststore kustom, kami akan menggunakan kluster MSK Amazon dengan Otentikasi Mutual TLS (MTLS) diaktifkan. Ini menyiratkan bahwa node cluster MSK menggunakan sertifikat server yang dikeluarkan oleh Certificate Manager Private AWS Certificate Authority (ACM Private CA) yang bersifat pribadi untuk akun dan Wilayah Anda dan oleh karena itu tidak dipercaya oleh Truststore default Java Virtual Machine (JVM) yang menjalankan Aplikasi Flink.

> **Note**
>
> - Keystore digunakan untuk menyimpan kunci pribadi dan sertifikat identitas aplikasi harus hadir ke server atau klien untuk verifikasi.
> - Truststore digunakan untuk menyimpan sertifikat dari Otoritas Bersertifikat (CA) yang memverifikasi sertifikat yang disajikan oleh server dalam koneksi SSL.

Anda juga dapat menggunakan teknik dalam tutorial ini untuk interaksi antara Managed Service untuk aplikasi Apache Flink dan sumber Apache Kafka lainnya, seperti:

- Klaster Apache Kafka kustom yang di-hosting di AWS (Amazon EC2 atau Amazon EKS)
- Klaster Confluent Kafka yang di-hosting di AWS
- Klaster Kafka on-premise yang diakses melalui AWS Direct Connect atau VPN

Tutorial ini berisi bagian-bagian berikut:

- Buat VPC dengan Klaster Amazon MSK
- Buat Truststore Kustom dan Terapkan ke Cluster Anda
- Buat Kode Aplikasi
- Unggah Kode Java Streaming Apache Flink
- Buat Aplikasi
- Konfigurasikan Aplikasi
- Jalankan Aplikasi
- Uji Aplikasi

Buat VPC dengan Klaster Amazon MSK

Untuk membuat contoh klaster VPC dan Amazon MSK untuk mengakses dari aplikasi Managed Service for Apache Flink, ikuti tutorial Memulai Menggunakan Amazon MSK.

Saat menyelesaikan tutorial, juga lakukan hal berikut:

- Pada Langkah 3: Buat Topik, ulangi `kafka-topics.sh --create` perintah untuk membuat topik tujuan bernama`AWSKafkaTutorialTopicDestination`:

```
bin/kafka-topics.sh --create --bootstrap-server ZooKeeperConnectionString --
replication-factor 3 --partitions 1 --topic AWSKafkaTutorialTopicDestination
```

> **Note**
>
> Jika perintah `kafka-topics.sh` menampilkan `ZooKeeperClientTimeoutException`, verifikasi bahwa grup keamanan klaster Kafka memiliki aturan inbound untuk mengizinkan semua lalu lintas dari alamat IP privat instans klien.

- Catat daftar server bootstrap untuk klaster Anda. Anda bisa mendapatkan daftar server bootstrap dengan perintah berikut (ganti *ClusterArn*dengan ARN cluster MSK Anda):

```
aws kafka get-bootstrap-brokers --region us-west-2 --cluster-arn ClusterArn
{...
    "BootstrapBrokerStringTls": "b-2.awskafkatutorialcluste.t79r6y.c4.kafka.us-
west-2.amazonaws.com:9094,b-1.awskafkatutorialcluste.t79r6y.c4.kafka.us-
west-2.amazonaws.com:9094,b-3.awskafkatutorialcluste.t79r6y.c4.kafka.us-
west-2.amazonaws.com:9094"
}
```

- Saat mengikuti langkah-langkah dalam tutorial dan tutorial prasyarat, pastikan untuk menggunakan Wilayah AWS yang Anda pilih dalam kode, perintah, dan entri konsol Anda.

Buat Truststore Kustom dan Terapkan ke Cluster Anda

Di bagian ini, Anda membuat otoritas sertifikat kustom (CA), menggunakannya untuk menghasilkan truststore kustom, dan menerapkannya ke cluster MSK Anda.

Untuk membuat dan menerapkan truststore kustom Anda, ikuti tutorial Otentikasi Klien di Amazon Managed Streaming for Apache Kafka Developer Guide.

Buat Kode Aplikasi

Di bagian ini, Anda mengunduh dan mengompilasi file JAR aplikasi.

Kode aplikasi Java untuk contoh ini tersedia dari GitHub. Untuk mengunduh kode aplikasi, lakukan hal berikut:

1. Instal klien Git jika Anda belum menginstalnya. Untuk informasi selengkapnya, lihat Menginstal Git.
2. Klon repositori jarak jauh dengan perintah berikut:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Kode aplikasi terletak di`amazon-kinesis-data-analytics-java-examples/CustomKeystore`. Anda dapat memeriksa kode untuk membiasakan diri dengan struktur Managed Service untuk kode Apache Flink.
4. Gunakan salah satu alat Maven baris perintah atau lingkungan pengembangan pilihan Anda untuk membuat file JAR. Untuk mengompilasi file JAR menggunakan alat Maven baris perintah, masukkan berikut ini:

```
mvn package -Dflink.version=1.15.3
```

Jika berhasil membangun, file berikut dibuat:

```
target/flink-app-1.0-SNAPSHOT.jar
```

> **Note**
>
> Kode sumber yang disediakan bergantung pada pustaka dari Java 11.

Unggah Kode Java Streaming Apache Flink

Di bagian ini, Anda mengunggah kode aplikasi ke bucket Amazon S3 yang Anda buat di tutorial Memulai (DataStream API).

> **Note**
>
> Jika Anda menghapus bucket Amazon S3 dari tutorial Memulai, ikuti lagi langkah the section called “Unggah Kode Java Streaming Apache Flink”.

1. Di konsol Amazon S3, pilih bucket ka-app-code -, dan pilih Unggah. <username>

2. Di langkah Pilih file, pilih Add files (Tambahkan berkas). Navigasikan ke file `flink-app-1.0-SNAPSHOT.jar` yang Anda buat di langkah sebelumnya.
3. Anda tidak perlu mengubah pengaturan apa pun untuk objek, jadi pilih Upload (Unggah).

Kode aplikasi Anda sekarang disimpan di bucket Amazon S3 yang dapat diakses aplikasi Anda.

Buat Aplikasi

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/flink
2. Pada dashboard Managed Service for Apache Flink, pilih Create Analytics Application.
3. Pada Layanan Terkelola untuk Apache Flink - Buat halaman aplikasi, berikan detail aplikasi sebagai berikut:
   - Untuk Application name (Nama aplikasi), masukkan **`MyApplication`**.
   - Untuk Runtime, pilih Apache Flink versi 1.15.2.
4. Untuk Access permissions (Izin akses), pilih Create / update IAM role **`kinesis-analytics-MyApplication-us-west-2`** (Buat/perbarui IAM role).
5. Pilih Create application (Buat aplikasi).

> **Note**
>
> Saat Anda membuat Layanan Terkelola untuk Apache Flink menggunakan konsol, Anda memiliki opsi untuk membuat peran dan kebijakan IAM untuk aplikasi Anda. Aplikasi Anda menggunakan peran dan kebijakan ini untuk mengakses sumber daya dependen. Sumber daya IAM ini diberi nama menggunakan nama aplikasi dan Wilayah sebagai berikut:
>
> - Kebijakan: `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Peran: `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Konfigurasikan Aplikasi

1. Pada MyApplicationhalaman, pilih Konfigurasi.
2. Di halaman Konfigurasikan aplikasi, berikan Code location (Lokasi kode):
   - Untuk Bucket Amazon S3, masukkan **`ka-app-code-`*`<username>`***.

- Untuk Jalur ke objek Amazon S3, masukkan **`flink-app-1.0-SNAPSHOT.jar`**.

3. Di bawah Akses ke sumber daya aplikasi, untuk Access permissions (Izin akses), pilih Create / update IAM role **`kinesis-analytics-MyApplication-us-west-2`** (Pilih/perbarui IAM role).

> **Note**
>
> Jika Anda menentukan sumber daya aplikasi menggunakan konsol (seperti logd atau VPC), konsol tersebut akan mengubah peran eksekusi aplikasi Anda untuk memberikan izin mengakses sumber daya tersebut.

4. Di bawah Properties (Properti), pilih Add Group (Tambahkan Grup). Masukkan properti berikut:

ID Grup	Kunci	Nilai
`KafkaSource`	topik	AWSKafkaTutorialTopic
`KafkaSource`	bootstrap.servers	*`Daftar server bootstrap yang Anda simpan sebelumnya`*
`KafkaSource`	security.protocol	SSL
`KafkaSource`	ssl.truststore.location	/usr/lib/jvm/java-11-amazon-corretto/lib/keamanan/cacerts
`KafkaSource`	ssl.truststore.password	changeit

> **Note**
>
> ssl.truststore.password untuk sertifikat default adalah "changeit"—Anda tidak perlu mengubah nilai ini jika menggunakan sertifikat default.

Pilih Add Group (Tambahkan Grup) lagi. Masukkan properti berikut:

ID Grup	Kunci	Nilai
KafkaSink	topik	AWSKafkaTutorialTopicDestination
KafkaSink	bootstrap.servers	*Daftar server bootstrap yang Anda simpan sebelumnya*
KafkaSink	security.protocol	SSL
KafkaSink	ssl.truststore.location	/usr/lib/jvm/java-11-amazon-corretto/lib/keamanan/cacerts
KafkaSink	ssl.truststore.password	changeit
KafkaSink	transaction.timeout.ms	1000

Kode aplikasi membaca properti aplikasi di atas untuk mengonfigurasi sumber dan sink yang digunakan untuk berinteraksi dengan VPC dan klaster Amazon MSK Anda. Untuk informasi selengkapnya tentang penggunaan runtime, lihat Properti Runtime.

5. Di bawah Snapshots, pilih Disable (Nonaktifkan). Tindakan ini akan memudahkan pembaruan aplikasi tanpa memuat data status aplikasi yang tidak valid.
6. Di bawah Pemantauan, pastikan Memantau tingkat metrik diatur ke Aplikasi.
7. Untuk CloudWatch logging, pilih kotak centang Aktifkan.
8. Di bagian Virtual Private Cloud (VPC), pilih VPC untuk dikaitkan dengan aplikasi Anda. Pilih subnet dan grup keamanan yang terkait dengan VPC Anda yang ingin digunakan aplikasi untuk mengakses sumber daya VPC.
9. Pilih Update (Perbarui).

> **Note**
>
> Saat Anda memilih untuk mengaktifkan CloudWatch logging, Managed Service for Apache Flink membuat grup log dan aliran log untuk Anda. Nama-nama sumber daya ini adalah sebagai berikut:
>
> - Grup log: `/aws/kinesis-analytics/MyApplication`
> - Aliran log: `kinesis-analytics-log-stream`
>
> Aliran log ini digunakan untuk memantau aplikasi.

Jalankan Aplikasi

Grafik pekerjaan Flink dapat dilihat dengan menjalankan aplikasi, membuka dasbor Apache Flink, dan memilih pekerjaan Flink yang diinginkan.

Uji Aplikasi

Di bagian ini, Anda menulis catatan ke topik sumber. Aplikasi membaca catatan dari topik sumber dan menuliskannya ke topik tujuan. Anda memverifikasi bahwa aplikasi berfungsi dengan menulis catatan ke topik sumber dan membaca catatan dari topik tujuan.

Untuk menulis dan membaca catatan dari topik, ikuti langkah-langkah di Langkah 6: Buat dan Gunakan Data di tutorial Memulai Menggunakan Amazon MSK.

Untuk membaca dari topik tujuan, gunakan nama topik tujuan bukan topik sumber dalam koneksi kedua Anda ke kluster:

```
bin/kafka-console-consumer.sh --bootstrap-server BootstrapBrokerString --
consumer.config client.properties --topic AWSKafkaTutorialTopicDestination --from-
beginning
```

Jika tidak ada catatan yang muncul dalam topik tujuan, lihat bagian Tidak Dapat Mengakses Sumber Daya dalam VPC di topik Pemecahan Masalah.

Contoh Python

Contoh berikut menunjukkan cara membuat aplikasi menggunakan Python dengan API Tabel Apache Flink.

Topik

Contoh: Membuat Jendela Tumbling di Python

Dalam latihan ini, Anda membuat Layanan Terkelola Python untuk aplikasi Apache Flink yang mengumpulkan data menggunakan jendela tumbling.

> **Note**
>
> Untuk menyiapkan prasyarat yang diperlukan untuk latihan ini, selesaikan latihan Memulai (Python) terlebih dulu.

Topik ini berisi bagian-bagian berikut:

Buat Sumber Daya Dependen

Sebelum Anda membuat Layanan Terkelola untuk aplikasi Apache Flink untuk latihan ini, Anda membuat sumber daya dependen berikut:

- Dua Kinesis data streams (`ExampleInputStream` dan `ExampleOutputStream`)

- Bucket Amazon S3 untuk menyimpan kode aplikasi (`ka-app-code-<username>`)

Anda dapat membuat aliran Kinesis dan bucket Amazon S3 menggunakan konsol. Untuk petunjuk membuat sumber daya ini, lihat topik berikut:

- Membuat dan Memperbarui Aliran Data di Panduan Developer Amazon Kinesis Data Streams. Beri nama aliran data **`ExampleInputStream`** dan **`ExampleOutputStream`** Anda.
- Bagaimana Cara Membuat Bucket S3? di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon. Beri bucket Amazon S3 nama yang unik secara global dengan menambahkan nama login Anda, seperti **`ka-app-code-<username>`**.

Tulis Catatan Sampel ke Aliran Input

Di bagian ini, Anda menggunakan script Python untuk menulis catatan sampel ke aliran untuk diproses aplikasi.

> **Note**
>
> Bagian ini memerlukan AWS SDK for Python (Boto).

> **Note**
>
> Skrip Python di bagian ini menggunakan AWS CLI. Anda harus mengonfigurasi AWS CLI untuk menggunakan kredensial akun dan wilayah default Anda. Untuk mengonfigurasi AWS CLI Anda, masukkan berikut ini:
>
> ```
> aws configure
> ```

1. Buat file bernama `stock.py` dengan konten berikut:

```
import datetime
import json
import random
import boto3
```

```
    STREAM_NAME = "ExampleInputStream"


    def get_data():
        return {
            'event_time': datetime.datetime.now().isoformat(),
            'ticker': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
            'price': round(random.random() * 100, 2)}


    def generate(stream_name, kinesis_client):
        while True:
            data = get_data()
            print(data)
            kinesis_client.put_record(
                StreamName=stream_name,
                Data=json.dumps(data),
                PartitionKey="partitionkey")


    if __name__ == '__main__':
        generate(STREAM_NAME, boto3.client('kinesis', region_name='us-west-2'))
```

2. Jalankan skrip `stock.py`.

```
$ python stock.py
```

Biarkan skrip tetap berjalan saat menyelesaikan sisa tutorial.

Unduh dan Periksa Kode Aplikasi

Kode aplikasi Python untuk contoh ini tersedia dari. GitHub Untuk mengunduh kode aplikasi, lakukan hal berikut:

1. Instal klien Git jika Anda belum menginstalnya. Untuk informasi selengkapnya, lihat Menginstal Git.
2. Klon repositori jarak jauh dengan perintah berikut:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Buka direktori `amazon-kinesis-data-analytics-java-examples/python/TumblingWindow` tersebut.

Kode aplikasi terletak di file `tumbling-windows.py`. Perhatikan hal tentang kode aplikasi berikut:

- Aplikasi menggunakan sumber tabel Kinesis untuk membaca dari aliran sumber. Cuplikan berikut memanggil fungsi `create_table` untuk membuat sumber tabel Kinesis:

```
table_env.execute_sql(
        create_input_table(input_table_name, input_stream, input_region,
 stream_initpos)
    )
```

  Fungsi `create_table` menggunakan perintah SQL untuk membuat tabel yang didukung oleh sumber streaming:

```
def create_input_table(table_name, stream_name, region, stream_initpos):
    return """ CREATE TABLE {0} (
                ticker VARCHAR(6),
                price DOUBLE,
                event_time TIMESTAMP(3),
                WATERMARK FOR event_time AS event_time - INTERVAL '5' SECOND
              )
              PARTITIONED BY (ticker)
              WITH (
                'connector' = 'kinesis',
                'stream' = '{1}',
                'aws.region' = '{2}',
                'scan.stream.initpos' = '{3}',
                'format' = 'json',
                'json.timestamp-format.standard' = 'ISO-8601'
              ) """.format(table_name, stream_name, region, stream_initpos)
```

- Aplikasi menggunakan operator `Tumble` untuk menggabungkan catatan dalam jendela tumbling tertentu, dan mengembalikan catatan agregat sebagai objek tabel:

```
tumbling_window_table = (
        input_table.window(
            Tumble.over("10.seconds").on("event_time").alias("ten_second_window")
        )
        .group_by("ticker, ten_second_window")
```

```
        .select("ticker, price.min as price, to_string(ten_second_window.end) as
event_time")
```

- Aplikasi ini menggunakan konektor Flink Kinesis, dari `flink-sql-connector-kinesis-1.15.2.jar`.

Kompres dan Unggah Kode Python Streaming Apache Flink

Di bagian ini, Anda mengunggah kode aplikasi ke bucket Amazon S3 yang Anda buat di bagian Buat Sumber Daya Dependen.

1. Gunakan aplikasi kompresi pilihan Anda untuk mengompresi file `tumbling-windows.py` dan `flink-sql-connector-kinesis-1.15.2.jar`. Beri nama arsip `myapp.zip`.
2. Di konsol Amazon S3, pilih bucket ka-app-code -, dan pilih Unggah. <username>
3. Di langkah Pilih file, pilih Add files (Tambahkan berkas). Navigasikan ke file `myapp.zip` yang Anda buat di langkah sebelumnya.
4. Anda tidak perlu mengubah pengaturan apa pun untuk objek, jadi pilih Upload (Unggah).

Kode aplikasi Anda sekarang disimpan di bucket Amazon S3 yang dapat diakses aplikasi Anda.

Buat dan Jalankan Managed Service untuk Apache Flink Application

Ikuti langkah-langkah ini untuk membuat, mengonfigurasi, memperbarui, dan menjalankan aplikasi menggunakan konsol.

Buat Aplikasi

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/flink
2. Pada dashboard Managed Service for Apache Flink, pilih Create Analytics Application.
3. Pada Layanan Terkelola untuk Apache Flink - Buat halaman aplikasi, berikan detail aplikasi sebagai berikut:
   - Untuk Application name (Nama aplikasi), masukkan **`MyApplication`**.
   - Untuk Runtime, pilih Apache Flink.

> **Note**
>
> Managed Service untuk Apache Flink menggunakan Apache Flink versi 1.15.2.

- Biarkan versi pulldown sebagai Apache Flink versi 1.15.2 (Versi yang disarankan).

4. Untuk Access permissions (Izin akses), pilih Create / update IAM role **`kinesis-analytics-MyApplication-us-west-2`** (Buat/perbarui IAM role).
5. Pilih Create application (Buat aplikasi).

> **Note**
>
> Saat membuat aplikasi Managed Service for Apache Flink menggunakan konsol, Anda memiliki opsi untuk membuat peran dan kebijakan IAM untuk aplikasi Anda. Aplikasi Anda menggunakan peran dan kebijakan ini untuk mengakses sumber daya dependen. Sumber daya IAM ini diberi nama menggunakan nama aplikasi dan Wilayah sebagai berikut:
>
> - Kebijakan: `kinesis-analytics-service-MyApplication-us-west-2`
> - Peran: `kinesisanalytics-MyApplication-us-west-2`

Konfigurasikan Aplikasi

1. Pada MyApplicationhalaman, pilih Konfigurasi.
2. Di halaman Konfigurasikan aplikasi, berikan Code location (Lokasi kode):
   - Untuk Bucket Amazon S3, masukkan **`ka-app-code-<username>`**.
   - Untuk Jalur ke objek Amazon S3, masukkan **`myapp.zip`**.
3. Di bawah Akses ke sumber daya aplikasi, untuk Access permissions (Izin akses), pilih Create / update IAM role **`kinesis-analytics-MyApplication-us-west-2`** (Pilih/perbarui IAM role).
4. Di bawah Properties (Properti), pilih Add group (Tambahkan grup).
5. Masukkan yang berikut ini:

ID Grup	Kunci	Nilai
consumer.config.0	**input.stream.name**	**ExampleInputStream**
consumer.config.0	**aws.region**	**us-west-2**
consumer.config.0	**scan.stream.initpos**	**LATEST**

Pilih Simpan.

6. Di bawah Properties (Properti), pilih Add group (Tambahkan grup) lagi.
7. Masukkan yang berikut ini:

ID Grup	Kunci	Nilai
producer.config.0	**output.stream.name**	**ExampleOutputStream**
producer.config.0	**aws.region**	**us-west-2**
producer.config.0	**shard.count**	**1**

8. Di bawah Properties (Properti), pilih Add group (Tambahkan grup) lagi. Untuk ID Grup, masukkan **kinesis.analytics.flink.run.options**. Grup properti khusus ini memberi tahu aplikasi Anda tempat untuk menemukan sumber daya kodenya. Untuk informasi selengkapnya, lihat Menentukan File Kode Anda.
9. Masukkan yang berikut ini:

ID Grup	Kunci	Nilai
kinesis.analytics.flink.run.options	**python**	**tumbling-windows.py**
kinesis.analytics.flink.run.options	**jarfile**	**flink-sql-connector-kinesis-1.15.2.jar**

10. Di bawah Pemantauan, pastikan Memantau tingkat metrik diatur ke Aplikasi.

11. Untuk CloudWatch logging, pilih kotak centang Aktifkan.
12. Pilih Perbarui.

> **Note**
>
> Saat Anda memilih untuk mengaktifkan CloudWatch logging, Managed Service for Apache Flink membuat grup log dan aliran log untuk Anda. Nama-nama sumber daya ini adalah sebagai berikut:
>
> - Grup log: `/aws/kinesis-analytics/MyApplication`
> - Aliran log: `kinesis-analytics-log-stream`
>
> Aliran log ini digunakan untuk memantau aplikasi. Ini bukan aliran log yang sama dengan yang digunakan aplikasi untuk mengirim hasil.

Edit Kebijakan IAM

Edit kebijakan IAM untuk menambahkan izin mengakses Kinesis data streams.

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pilih Policies (Kebijakan). Pilih kebijakan **`kinesis-analytics-service-MyApplication-us-west-2`** yang dibuat konsol untuk Anda di bagian sebelumnya.
3. Di halaman Ringkasan, pilih Edit policy (Edit kebijakan). Pilih tab JSON.
4. Tambahkan bagian yang disorot dari contoh kebijakan berikut ke kebijakan. Ganti ID akun sampel (*`012345678901`*) dengan ID akun Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "logs:DescribeLogGroups",
                "s3:GetObjectVersion"
            ],
```

```
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*",
                "arn:aws:s3:::ka-app-code-<username>/myapp.zip"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": "logs:DescribeLogStreams",
            "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/
kinesis-analytics/MyApplication:log-stream:*"
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": "logs:PutLogEvents",
            "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/
kinesis-analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
        },
        {
            "Sid": "ListCloudwatchLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
```

```
}
```

Jalankan Aplikasi

Grafik pekerjaan Flink dapat dilihat dengan menjalankan aplikasi, membuka dasbor Apache Flink, dan memilih pekerjaan Flink yang diinginkan.

Anda dapat memeriksa metrik Managed Service for Apache Flink di CloudWatch konsol untuk memverifikasi bahwa aplikasi berfungsi.

Pembersihan Sumber Daya AWS

Bagian ini mencakup prosedur untuk membersihkan sumber daya AWS yang dibuat dalam tutorial Jendela Tumbling.

Topik ini berisi bagian-bagian berikut:

- Hapus Layanan Terkelola Anda untuk Aplikasi Apache Flink
- Hapus Kinesis Data Streams Anda
- Hapus Objek dan Bucket Amazon S3 Anda
- Hapus Sumber Daya IAM Anda
- Hapus CloudWatch Sumber Daya Anda

Hapus Layanan Terkelola Anda untuk Aplikasi Apache Flink

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/flink
2. di panel Managed Service for Apache Flink, pilih. MyApplication
3. Di halaman aplikasi, pilih Delete (Hapus), lalu konfirmasikan penghapusan.

Hapus Kinesis Data Streams Anda

1. Buka konsol Kinesis di https://console.aws.amazon.com/kinesis.
2. Di panel Kinesis Data Streams, pilih. ExampleInputStream
3. Di ExampleInputStreamhalaman, pilih Hapus Stream Kinesis dan kemudian konfirmasikan penghapusan.
4. Di halaman Kinesis streams, pilih, pilih Tindakan ExampleOutputStream, pilih Hapus, lalu konfirmasikan penghapusan.

Hapus Objek dan Bucket Amazon S3 Anda

1. Buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.
2. Pilih ka-app-code- ember. <username>
3. Pilih Delete (Hapus), lalu masukkan nama bucket untuk mengonfirmasi penghapusan.

Hapus Sumber Daya IAM Anda

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di bilah navigasi, pilih Policies (Kebijakan).
3. Di kontrol filter, masukkan kinesis.
4. Pilih kebijakan kinesis-analytics-service- MyApplication -us-west-2.
5. Pilih Policy Actions (Tindakan Kebijakan), lalu pilih Delete (Hapus).
6. Di bilah navigasi, pilih Roles (Peran).
7. Pilih peran kinesis-analytics- -us-west-2. MyApplication
8. Pilih Delete role (Hapus peran), lalu konfirmasi penghapusan.

Hapus CloudWatch Sumber Daya Anda

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di bilah navigasi, pilih Logs.
3. Pilih grup log MyApplication/aws/kinesis-analytics/.
4. Pilih Delete Log Group (Hapus Grup Log), lalu konfirmasi penghapusan.

Contoh: Membuat Jendela Geser di Python

> **Note**
>
> Untuk menyiapkan prasyarat yang diperlukan untuk latihan ini, selesaikan latihan Memulai (Python) terlebih dulu.

Topik ini berisi bagian-bagian berikut:

- Buat Sumber Daya Dependen

- Tulis Catatan Sampel ke Aliran Input
- Unduh dan Periksa Kode Aplikasi
- Kompres dan Unggah Kode Python Streaming Apache Flink
- Buat dan Jalankan Managed Service untuk Apache Flink Application
- Pembersihan Sumber Daya AWS

Buat Sumber Daya Dependen

Sebelum Anda membuat Layanan Terkelola untuk aplikasi Apache Flink untuk latihan ini, Anda membuat sumber daya dependen berikut:

- Dua Kinesis data streams (`ExampleInputStream` dan `ExampleOutputStream`)
- Bucket Amazon S3 untuk menyimpan kode aplikasi (`ka-app-code-<username>`)

Anda dapat membuat aliran Kinesis dan bucket Amazon S3 menggunakan konsol. Untuk petunjuk membuat sumber daya ini, lihat topik berikut:

- Membuat dan Memperbarui Aliran Data di Panduan Developer Amazon Kinesis Data Streams. Beri nama aliran data **`ExampleInputStream`** dan **`ExampleOutputStream`** Anda.
- Bagaimana Cara Membuat Bucket S3? di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon. Beri bucket Amazon S3 nama yang unik secara global dengan menambahkan nama login Anda, seperti **`ka-app-code-<username>`**.

Tulis Catatan Sampel ke Aliran Input

Di bagian ini, Anda menggunakan script Python untuk menulis catatan sampel ke aliran untuk diproses aplikasi.

> **Note**
>
> Bagian ini memerlukan AWS SDK for Python (Boto).

> **Note**
>
> Skrip Python di bagian ini menggunakan AWS CLI. Anda harus mengonfigurasi AWS CLI untuk menggunakan kredensial akun dan wilayah default Anda. Untuk mengonfigurasi AWS CLI Anda, masukkan berikut ini:
>
> ```
> aws configure
> ```

1. Buat file bernama `stock.py` dengan konten berikut:

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        'event_time': datetime.datetime.now().isoformat(),
        'ticker': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'price': round(random.random() * 100, 2)}


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(data),
            PartitionKey="partitionkey")


if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis', region_name='us-west-2'))
```

2. Jalankan skrip `stock.py`.

```
$ python stock.py
```

Biarkan skrip tetap berjalan saat menyelesaikan sisa tutorial.

Unduh dan Periksa Kode Aplikasi

Kode aplikasi Python untuk contoh ini tersedia dari. GitHub Untuk mengunduh kode aplikasi, lakukan hal berikut:

1. Instal klien Git jika Anda belum menginstalnya. Untuk informasi selengkapnya, lihat Menginstal Git.
2. Klon repositori jarak jauh dengan perintah berikut:

```
git clone https://github.com/aws-samples/>amazon-kinesis-data-analytics-java-
examples
```

3. Buka direktori `amazon-kinesis-data-analytics-java-examples/python/SlidingWindow` tersebut.

Kode aplikasi terletak di file `sliding-windows.py`. Perhatikan hal tentang kode aplikasi berikut:

- Aplikasi menggunakan sumber tabel Kinesis untuk membaca dari aliran sumber. Cuplikan berikut memanggil fungsi `create_input_table` untuk membuat sumber tabel Kinesis:

```
table_env.execute_sql(
        create_input_table(input_table_name, input_stream, input_region,
 stream_initpos)
    )
```

  Fungsi `create_input_table` menggunakan perintah SQL untuk membuat tabel yang didukung oleh sumber streaming:

```
def create_input_table(table_name, stream_name, region, stream_initpos):
    return """ CREATE TABLE {0} (
                ticker VARCHAR(6),
                price DOUBLE,
                event_time TIMESTAMP(3),
                WATERMARK FOR event_time AS event_time - INTERVAL '5' SECOND
```

```
            )
            PARTITIONED BY (ticker)
            WITH (
              'connector' = 'kinesis',
              'stream' = '{1}',
              'aws.region' = '{2}',
              'scan.stream.initpos' = '{3}',
              'format' = 'json',
              'json.timestamp-format.standard' = 'ISO-8601'
            ) """.format(table_name, stream_name, region, stream_initpos)
}
```

- Aplikasi menggunakan operator `Slide` untuk menggabungkan catatan dalam jendela geser tertentu, dan mengembalikan catatan agregat sebagai objek tabel:

```
sliding_window_table = (
        input_table
            .window(
                Slide.over("10.seconds")
                .every("5.seconds")
                .on("event_time")
                .alias("ten_second_window")
            )
            .group_by("ticker, ten_second_window")
            .select("ticker, price.min as price, to_string(ten_second_window.end) as
 event_time")
    )
```

- Aplikasi ini menggunakan konektor Kinesis Flink, dari file -1.15.2.jar. flink-sql-connector-kinesis

Kompres dan Unggah Kode Python Streaming Apache Flink

Di bagian ini, Anda mengunggah kode aplikasi ke bucket Amazon S3 yang Anda buat di bagian Buat Sumber Daya Dependen.

Bagian ini menjelaskan cara mengemas aplikasi Python Anda.

1. Gunakan aplikasi kompresi pilihan Anda untuk mengompresi file `sliding-windows.py` dan `flink-sql-connector-kinesis-1.15.2.jar`. Beri nama arsip `myapp.zip`.
2. Di konsol Amazon S3, pilih bucket ka-app-code -, dan pilih Unggah. <username>

3. Di langkah Pilih file, pilih Add files (Tambahkan berkas). Navigasikan ke file `myapp.zip` yang Anda buat di langkah sebelumnya.
4. Anda tidak perlu mengubah pengaturan apa pun untuk objek, jadi pilih Upload (Unggah).

Kode aplikasi Anda sekarang disimpan di bucket Amazon S3 yang dapat diakses aplikasi Anda.

Buat dan Jalankan Managed Service untuk Apache Flink Application

Ikuti langkah-langkah ini untuk membuat, mengonfigurasi, memperbarui, dan menjalankan aplikasi menggunakan konsol.

Buat Aplikasi

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/flink
2. Pada dashboard Managed Service for Apache Flink, pilih Create Analytics Application.
3. Pada Layanan Terkelola untuk Apache Flink - Buat halaman aplikasi, berikan detail aplikasi sebagai berikut:
   - Untuk Application name (Nama aplikasi), masukkan **`MyApplication`**.
   - Untuk Runtime, pilih Apache Flink.

     > **Note**
     >
     > Managed Service untuk Apache Flink menggunakan Apache Flink versi 1.15.2.

   - Biarkan versi pulldown sebagai Apache Flink versi 1.15.2 (Versi yang disarankan).
4. Untuk Access permissions (Izin akses), pilih Create / update IAM role **`kinesis-analytics-MyApplication-us-west-2`** (Buat/perbarui IAM role).
5. Pilih Create application (Buat aplikasi).

> **Note**
>
> Saat membuat aplikasi Managed Service for Apache Flink menggunakan konsol, Anda memiliki opsi untuk membuat peran dan kebijakan IAM untuk aplikasi Anda. Aplikasi Anda menggunakan peran dan kebijakan ini untuk mengakses sumber daya dependen. Sumber daya IAM ini diberi nama menggunakan nama aplikasi dan Wilayah sebagai berikut:

- Kebijakan: `kinesis-analytics-service-MyApplication-us-west-2`
- Peran: `kinesisanalytics-MyApplication-us-west-2`

Konfigurasikan Aplikasi

1. Pada MyApplicationhalaman, pilih Konfigurasi.
2. Di halaman Konfigurasikan aplikasi, berikan Code location (Lokasi kode):
   - Untuk Bucket Amazon S3, masukkan **`ka-app-code-<username>`**.
   - Untuk Jalur ke objek Amazon S3, masukkan **`myapp.zip`**.
3. Di bawah Akses ke sumber daya aplikasi, untuk Access permissions (Izin akses), pilih Create / update IAM role **`kinesis-analytics-MyApplication-us-west-2`** (Pilih/perbarui IAM role).
4. Di bawah Properties (Properti), pilih Add group (Tambahkan grup).
5. Masukkan properti dan nilai aplikasi berikut:

ID Grup	Kunci	Nilai
`consumer.config.0`	**`input.stream.name`**	**`ExampleInputStream`**
`consumer.config.0`	**`aws.region`**	**`us-west-2`**
`consumer.config.0`	**`scan.stream.initpos`**	**`LATEST`**

   Pilih Simpan.
6. Di bawah Properties (Properti), pilih Add group (Tambahkan grup) lagi.
7. Masukkan properti dan nilai aplikasi berikut:

ID Grup	Kunci	Nilai
`producer.config.0`	**`output.stream.name`**	**`ExampleOutputStream`**
`producer.config.0`	**`aws.region`**	**`us-west-2`**

ID Grup	Kunci	Nilai
producer.config.0	**shard.count**	**1**

8. Di bawah Properties (Properti), pilih Add group (Tambahkan grup) lagi. Untuk ID Grup, masukkan **kinesis.analytics.flink.run.options**. Grup properti khusus ini memberi tahu aplikasi Anda tempat untuk menemukan sumber daya kodenya. Untuk informasi selengkapnya, lihat Menentukan File Kode Anda.
9. Masukkan properti dan nilai aplikasi berikut:

ID Grup	Kunci	Nilai
kinesis.analytics.flink.run.options	**python**	**sliding-windows.py**
kinesis.analytics.flink.run.options	**jarfile**	**flink-sql-connector-kinesis_1.15.2.jar**

10. Di bawah Pemantauan, pastikan Memantau tingkat metrik diatur ke Aplikasi.
11. Untuk CloudWatch logging, pilih kotak centang Aktifkan.
12. Pilih Perbarui.

> **Note**
>
> Saat Anda memilih untuk mengaktifkan CloudWatch logging, Managed Service for Apache Flink membuat grup log dan aliran log untuk Anda. Nama-nama sumber daya ini adalah sebagai berikut:
>
> - Grup log: `/aws/kinesis-analytics/MyApplication`
> - Aliran log: `kinesis-analytics-log-stream`
>
> Aliran log ini digunakan untuk memantau aplikasi. Ini bukan aliran log yang sama dengan yang digunakan aplikasi untuk mengirim hasil.

Edit Kebijakan IAM

Edit kebijakan IAM untuk menambahkan izin mengakses Kinesis data streams.

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pilih Policies (Kebijakan). Pilih kebijakan **kinesis-analytics-service-MyApplication-us-west-2** yang dibuat konsol untuk Anda di bagian sebelumnya.
3. Di halaman Ringkasan, pilih Edit policy (Edit kebijakan). Pilih tab JSON.
4. Tambahkan bagian yang disorot dari contoh kebijakan berikut ke kebijakan. Ganti ID akun sampel (*012345678901*) dengan ID akun Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "logs:DescribeLogGroups",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*",
                "arn:aws:s3:::ka-app-code-<username>/myapp.zip"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": "logs:DescribeLogStreams",
            "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/
kinesis-analytics/MyApplication:log-stream:*"
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": "logs:PutLogEvents",
            "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/
kinesis-analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
        },
        {
```

```
            "Sid": "ListCloudwatchLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

Jalankan Aplikasi

Grafik pekerjaan Flink dapat dilihat dengan menjalankan aplikasi, membuka dasbor Apache Flink, dan memilih pekerjaan Flink yang diinginkan.

Anda dapat memeriksa metrik Managed Service for Apache Flink di CloudWatch konsol untuk memverifikasi bahwa aplikasi berfungsi.

Pembersihan Sumber Daya AWS

Bagian ini mencakup prosedur untuk membersihkan sumber daya AWS yang dibuat dalam tutorial Jendela Geser.

Topik ini berisi bagian-bagian berikut:

- Hapus Layanan Terkelola Anda untuk Aplikasi Apache Flink

- Hapus Kinesis Data Streams Anda
- Hapus Objek dan Bucket Amazon S3 Anda
- Hapus Sumber Daya IAM Anda
- Hapus CloudWatch Sumber Daya Anda

Hapus Layanan Terkelola Anda untuk Aplikasi Apache Flink

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/flink
2. di panel Managed Service for Apache Flink, pilih. MyApplication
3. Di halaman aplikasi, pilih Delete (Hapus), lalu konfirmasikan penghapusan.

Hapus Kinesis Data Streams Anda

1. Buka konsol Kinesis di https://console.aws.amazon.com/kinesis.
2. Di panel Kinesis Data Streams, pilih. ExampleInputStream
3. Di ExampleInputStreamhalaman, pilih Hapus Stream Kinesis dan kemudian konfirmasikan penghapusan.
4. Di halaman Kinesis streams, pilih, pilih Tindakan ExampleOutputStream, pilih Hapus, lalu konfirmasikan penghapusan.

Hapus Objek dan Bucket Amazon S3 Anda

1. Buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.
2. Pilih ka-app-code- ember. <username>
3. Pilih Delete (Hapus), lalu masukkan nama bucket untuk mengonfirmasi penghapusan.

Hapus Sumber Daya IAM Anda

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di bilah navigasi, pilih Policies (Kebijakan).
3. Di kontrol filter, masukkan kinesis.
4. Pilih kebijakan kinesis-analytics-service- MyApplication -us-west-2.
5. Pilih Policy Actions (Tindakan Kebijakan), lalu pilih Delete (Hapus).

6. Di bilah navigasi, pilih Roles (Peran).
7. Pilih peran kinesis-analytics- -us-west-2. MyApplication
8. Pilih Delete role (Hapus peran), lalu konfirmasi penghapusan.

Hapus CloudWatch Sumber Daya Anda

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di bilah navigasi, pilih Logs.
3. Pilih grup log MyApplication/aws/kinesis-analytics/.
4. Pilih Delete Log Group (Hapus Grup Log), lalu konfirmasi penghapusan.

Contoh: Kirim Data Streaming ke Amazon S3 dengan Python

Dalam latihan ini, Anda membuat Layanan Terkelola Python untuk aplikasi Apache Flink yang mengalirkan data ke wastafel Amazon Simple Storage Service.

> **Note**
>
> Untuk menyiapkan prasyarat yang diperlukan untuk latihan ini, selesaikan latihan Memulai (Python) terlebih dulu.

Topik ini berisi bagian-bagian berikut:

- Buat Sumber Daya Dependen
- Tulis Catatan Sampel ke Aliran Input
- Unduh dan Periksa Kode Aplikasi
- Kompres dan Unggah Kode Python Streaming Apache Flink
- Buat dan Jalankan Managed Service untuk Apache Flink Application
- Pembersihan Sumber Daya AWS

Buat Sumber Daya Dependen

Sebelum Anda membuat Layanan Terkelola untuk aplikasi Apache Flink untuk latihan ini, Anda membuat sumber daya dependen berikut:

- Kinesis data stream (`ExampleInputStream`)
- Bucket Amazon S3 untuk menyimpan kode dan output aplikasi (`ka-app-code-<username>`)

> **Note**
>
> Layanan Terkelola untuk Apache Flink tidak dapat menulis data ke Amazon S3 dengan enkripsi sisi server diaktifkan pada Layanan Terkelola untuk Apache Flink.

Anda dapat membuat aliran Kinesis dan bucket Amazon S3 menggunakan konsol. Untuk petunjuk membuat sumber daya ini, lihat topik berikut:

- Membuat dan Memperbarui Aliran Data di Panduan Developer Amazon Kinesis Data Streams. Beri nama aliran data **`ExampleInputStream`** Anda.
- Bagaimana Cara Membuat Bucket S3? di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon. Beri bucket Amazon S3 nama yang unik secara global dengan menambahkan nama login Anda, seperti **`ka-app-code-<username>`**.

Tulis Catatan Sampel ke Aliran Input

Di bagian ini, Anda menggunakan script Python untuk menulis catatan sampel ke aliran untuk diproses aplikasi.

> **Note**
>
> Bagian ini memerlukan AWS SDK for Python (Boto).

> **Note**
>
> Skrip Python di bagian ini menggunakan AWS CLI. Anda harus mengonfigurasi AWS CLI untuk menggunakan kredensial akun dan wilayah default Anda. Untuk mengonfigurasi AWS CLI Anda, masukkan berikut ini:
>
> ```
> aws configure
> ```

1. Buat file bernama `stock.py` dengan konten berikut:

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        'event_time': datetime.datetime.now().isoformat(),
        'ticker': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'price': round(random.random() * 100, 2)}


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(data),
            PartitionKey="partitionkey")


if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis', region_name='us-west-2'))
```

2. Jalankan skrip `stock.py`.

```
$ python stock.py
```

   Biarkan skrip tetap berjalan saat menyelesaikan sisa tutorial.

Unduh dan Periksa Kode Aplikasi

Kode aplikasi Python untuk contoh ini tersedia dari. GitHub Untuk mengunduh kode aplikasi, lakukan hal berikut:

1. Instal klien Git jika Anda belum menginstalnya. Untuk informasi selengkapnya, lihat Menginstal Git.
2. Klon repositori jarak jauh dengan perintah berikut:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Buka direktori `amazon-kinesis-data-analytics-java-examples/python/S3Sink` tersebut.

Kode aplikasi terletak di file `streaming-file-sink.py`. Perhatikan hal tentang kode aplikasi berikut:

- Aplikasi menggunakan sumber tabel Kinesis untuk membaca dari aliran sumber. Cuplikan berikut memanggil fungsi `create_source_table` untuk membuat sumber tabel Kinesis:

```
table_env.execute_sql(
        create_source_table(input_table_name, input_stream, input_region,
 stream_initpos)
    )
```

`create_source_table`Fungsi ini menggunakan perintah SQL untuk membuat tabel yang didukung oleh sumber streaming

```
import datetime
    import json
    import random
    import boto3

    STREAM_NAME = "ExampleInputStream"


    def get_data():
        return {
            'event_time': datetime.datetime.now().isoformat(),
            'ticker': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
            'price': round(random.random() * 100, 2)}


    def generate(stream_name, kinesis_client):
        while True:
```

```
            data = get_data()
            print(data)
            kinesis_client.put_record(
                StreamName=stream_name,
                Data=json.dumps(data),
                PartitionKey="partitionkey")


    if __name__ == '__main__':
        generate(STREAM_NAME, boto3.client('kinesis', region_name='us-west-2'))
```

- Aplikasi menggunakan konektor `filesystem` untuk mengirim catatan ke bucket Amazon S3:

```
def create_sink_table(table_name, bucket_name):
    return """ CREATE TABLE {0} (
                ticker VARCHAR(6),
                price DOUBLE,
                event_time VARCHAR(64)
              )
              PARTITIONED BY (ticker)
              WITH (
                  'connector'='filesystem',
                  'path'='s3a://{1}/',
                  'format'='json',
                  'sink.partition-commit.policy.kind'='success-file',
                  'sink.partition-commit.delay' = '1 min'
              ) """.format(table_name, bucket_name)
```

- Aplikasi ini menggunakan konektor Kinesis Flink, dari file -1.15.2.jar. flink-sql-connector-kinesis

Kompres dan Unggah Kode Python Streaming Apache Flink

Di bagian ini, Anda mengunggah kode aplikasi ke bucket Amazon S3 yang Anda buat di bagian Buat Sumber Daya Dependen.

1. Gunakan aplikasi kompresi pilihan Anda untuk mengompres file `streaming-file-sink.py` dan flink-sql-connector-kinesis-1.15.2.jar. Beri nama arsip `myapp.zip`.
2. Di konsol Amazon S3, pilih bucket ka-app-code -, dan pilih Unggah. <username>
3. Di langkah Pilih file, pilih Add files (Tambahkan berkas). Navigasikan ke file `myapp.zip` yang Anda buat di langkah sebelumnya.
4. Anda tidak perlu mengubah pengaturan apa pun untuk objek, jadi pilih Upload (Unggah).

Kode aplikasi Anda sekarang disimpan di bucket Amazon S3 yang dapat diakses aplikasi Anda.

Buat dan Jalankan Managed Service untuk Apache Flink Application

Ikuti langkah-langkah ini untuk membuat, mengonfigurasi, memperbarui, dan menjalankan aplikasi menggunakan konsol.

Buat Aplikasi

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/flink
2. Pada dashboard Managed Service for Apache Flink, pilih Create Analytics Application.
3. Pada Layanan Terkelola untuk Apache Flink - Buat halaman aplikasi, berikan detail aplikasi sebagai berikut:
   - Untuk Application name (Nama aplikasi), masukkan **`MyApplication`**.
   - Untuk Runtime, pilih Apache Flink.

     > **Note**
     >
     > Managed Service untuk Apache Flink menggunakan Apache Flink versi 1.15.2.

   - Biarkan versi pulldown sebagai Apache Flink versi 1.15.2 (Versi yang disarankan).
4. Untuk Access permissions (Izin akses), pilih Create / update IAM role **`kinesis-analytics-MyApplication-us-west-2`** (Buat/perbarui IAM role).
5. Pilih Create application (Buat aplikasi).

> **Note**
>
> Saat membuat aplikasi Managed Service for Apache Flink menggunakan konsol, Anda memiliki opsi untuk membuat peran dan kebijakan IAM untuk aplikasi Anda. Aplikasi Anda menggunakan peran dan kebijakan ini untuk mengakses sumber daya dependen. Sumber daya IAM ini diberi nama menggunakan nama aplikasi dan Wilayah sebagai berikut:
>
> - Kebijakan: `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Peran: `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Konfigurasikan Aplikasi

1. Pada MyApplicationhalaman, pilih Konfigurasi.
2. Di halaman Konfigurasikan aplikasi, berikan Code location (Lokasi kode):
   - Untuk Bucket Amazon S3, masukkan **`ka-app-code-<username>`**.
   - Untuk Jalur ke objek Amazon S3, masukkan **`myapp.zip`**.
3. Di bawah Akses ke sumber daya aplikasi, untuk Access permissions (Izin akses), pilih Create / update IAM role **`kinesis-analytics-MyApplication-us-west-2`** (Pilih/perbarui IAM role).
4. Di bawah Properties (Properti), pilih Add group (Tambahkan grup).
5. Masukkan properti dan nilai aplikasi berikut:

ID Grup	Kunci	Nilai
`consumer.config.0`	**`input.stream.name`**	**`ExampleInputStream`**
`consumer.config.0`	**`aws.region`**	**`us-west-2`**
`consumer.config.0`	**`scan.stream.initpos`**	**`LATEST`**

   Pilih Simpan.
6. Di bawah Properties (Properti), pilih Add group (Tambahkan grup) lagi. Untuk ID Grup, masukkan **`kinesis.analytics.flink.run.options`**. Grup properti khusus ini memberi tahu aplikasi Anda tempat untuk menemukan sumber daya kodenya. Untuk informasi selengkapnya, lihat Menentukan File Kode Anda.
7. Masukkan properti dan nilai aplikasi berikut:

ID Grup	Kunci	Nilai
`kinesis.analytics.flink.run.options`	**`python`**	**`streaming-file-sink.py`**

ID Grup	Kunci	Nilai
kinesis.analytics.flink.run.options	**jarfile**	**S3Sink/lib/flink-sql-connector-kinesis-1.15.2.jar**

8. Di bawah Properties (Properti), pilih Add group (Tambahkan grup) lagi. Untuk ID Grup, masukkan **sink.config.0**. Grup properti khusus ini memberi tahu aplikasi Anda tempat untuk menemukan sumber daya kodenya. Untuk informasi selengkapnya, lihat Menentukan File Kode Anda.
9. Masukkan properti dan nilai aplikasi berikut: (ganti nama *ember dengan nama* sebenarnya dari bucket Amazon S3 Anda.)

ID Grup	Kunci	Nilai
sink.config.0	**output.bucket.name**	***bucket-name***

10. Di bawah Pemantauan, pastikan Memantau tingkat metrik diatur ke Aplikasi.
11. Untuk CloudWatch logging, pilih kotak centang Aktifkan.
12. Pilih Perbarui.

> **Note**
>
> Saat Anda memilih untuk mengaktifkan CloudWatch logging, Managed Service for Apache Flink membuat grup log dan aliran log untuk Anda. Nama-nama sumber daya ini adalah sebagai berikut:
>
> - Grup log: `/aws/kinesis-analytics/MyApplication`
> - Aliran log: `kinesis-analytics-log-stream`
>
> Aliran log ini digunakan untuk memantau aplikasi. Ini bukan aliran log yang sama dengan yang digunakan aplikasi untuk mengirim hasil.

Edit Kebijakan IAM

Edit kebijakan IAM untuk menambahkan izin mengakses Kinesis data streams.

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pilih Policies (Kebijakan). Pilih kebijakan **`kinesis-analytics-service-MyApplication-us-west-2`** yang dibuat konsol untuk Anda di bagian sebelumnya.
3. Di halaman Ringkasan, pilih Edit policy (Edit kebijakan). Pilih tab JSON.
4. Tambahkan bagian yang disorot dari contoh kebijakan berikut ke kebijakan. Ganti ID akun sampel (*012345678901*) dengan ID akun Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "logs:DescribeLogGroups",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*",
                "arn:aws:s3:::ka-app-code-<username>/myapp.zip"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": "logs:DescribeLogStreams",
            "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/
kinesis-analytics/MyApplication:log-stream:*"
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": "logs:PutLogEvents",
            "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/
kinesis-analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
        },
        {
            "Sid": "ListCloudwatchLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
```

```
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteObjects",
            "Effect": "Allow",
            "Action": [
                "s3:Abort*",
                "s3:DeleteObject*",
                "s3:GetObject*",
                "s3:GetBucket*",
                "s3:List*",
                "s3:ListBucket",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::ka-app-code-<username>",
                "arn:aws:s3:::ka-app-code-<username>/*"
            ]
        }
    ]
}
```

Jalankan Aplikasi

Grafik pekerjaan Flink dapat dilihat dengan menjalankan aplikasi, membuka dasbor Apache Flink, dan memilih pekerjaan Flink yang diinginkan.

Anda dapat memeriksa metrik Managed Service for Apache Flink di CloudWatch konsol untuk memverifikasi bahwa aplikasi berfungsi.

Pembersihan Sumber Daya AWS

Bagian ini mencakup prosedur untuk membersihkan sumber daya AWS yang dibuat dalam tutorial Jendela Geser.

Topik ini berisi bagian-bagian berikut:

- Hapus Layanan Terkelola Anda untuk Aplikasi Apache Flink
- Hapus Kinesis Data Stream Anda
- Hapus Objek dan Bucket Amazon S3 Anda
- Hapus Sumber Daya IAM Anda
- Hapus CloudWatch Sumber Daya Anda

Hapus Layanan Terkelola Anda untuk Aplikasi Apache Flink

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/flink
2. di panel Managed Service for Apache Flink, pilih. MyApplication
3. Di halaman aplikasi, pilih Delete (Hapus), lalu konfirmasikan penghapusan.

Hapus Kinesis Data Stream Anda

1. Buka konsol Kinesis di https://console.aws.amazon.com/kinesis.
2. Di panel Kinesis Data Streams, pilih. ExampleInputStream
3. Di ExampleInputStreamhalaman, pilih Hapus Stream Kinesis dan kemudian konfirmasikan penghapusan.

Hapus Objek dan Bucket Amazon S3 Anda

1. Buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.
2. Pilih ka-app-code- ember. <username>
3. Pilih Delete (Hapus), lalu masukkan nama bucket untuk mengonfirmasi penghapusan.

Hapus Sumber Daya IAM Anda

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di bilah navigasi, pilih Policies (Kebijakan).

3. Di kontrol filter, masukkan kinesis.
4. Pilih kebijakan kinesis-analytics-service- MyApplication -us-west-2.
5. Pilih Policy Actions (Tindakan Kebijakan), lalu pilih Delete (Hapus).
6. Di bilah navigasi, pilih Roles (Peran).
7. Pilih peran kinesis-analytics- -us-west-2. MyApplication
8. Pilih Delete role (Hapus peran), lalu konfirmasi penghapusan.

Hapus CloudWatch Sumber Daya Anda

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di bilah navigasi, pilih Logs.
3. Pilih grup log MyApplication/aws/kinesis-analytics/.
4. Pilih Delete Log Group (Hapus Grup Log), lalu konfirmasi penghapusan.

Contoh Scala

Contoh berikut menunjukkan cara membuat aplikasi menggunakan Scala dengan Apache Flink.

Topik

- Contoh: Membuat Jendela Tumbling di Scala
- Contoh: Membuat Jendela Geser di Scala
- Contoh: Kirim Data Streaming ke Amazon S3 di Scala

Contoh: Membuat Jendela Tumbling di Scala

> **Note**
>
> Mulai dari versi 1.15 Flink gratis Scala. Aplikasi sekarang dapat menggunakan Java API dari versi Scala apa pun. Flink masih menggunakan Scala di beberapa komponen kunci secara internal tetapi tidak mengekspos Scala ke dalam classloader kode pengguna. Karena itu, pengguna perlu menambahkan dependensi Scala ke dalam arsip jar mereka.
> Untuk informasi selengkapnya tentang perubahan Scala di Flink 1.15, lihat Scala Free in One Fifteen.

Dalam latihan ini, Anda akan membuat aplikasi streaming sederhana yang menggunakan Scala 3.2.0 dan Java API Flink. DataStream Aplikasi membaca data dari aliran Kinesis, menggabungkannya menggunakan jendela geser dan menulis hasil ke output Kinesis stream.

> **Note**
>
> Untuk mengatur prasyarat yang diperlukan untuk latihan ini, pertama-tama selesaikan latihan Memulai (Scala).

Topik ini berisi bagian-bagian berikut:

- Unduh dan Periksa Kode Aplikasi
- Kompilasi dan unggah kode aplikasi
- Buat dan jalankan Aplikasi (konsol)
- Membuat dan menjalankan aplikasi (CLI)
- Perbarui kode aplikasi
- Pembersihan Sumber Daya AWS

Unduh dan Periksa Kode Aplikasi

Kode aplikasi Python untuk contoh ini tersedia dari. GitHub Untuk mengunduh kode aplikasi, lakukan hal berikut:

1. Instal klien Git jika Anda belum menginstalnya. Untuk informasi selengkapnya, lihat Menginstal Git.
2. Klon repositori jarak jauh dengan perintah berikut:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Buka direktori `amazon-kinesis-data-analytics-java-examples/scala/TumblingWindow` tersebut.

Perhatikan hal tentang kode aplikasi berikut:

- `build.sbt`File berisi informasi tentang konfigurasi dan dependensi aplikasi, termasuk Layanan Terkelola untuk pustaka Apache Flink.

- `BasicStreamingJob.scala`File berisi metode utama yang mendefinisikan fungsionalitas aplikasi.
- Aplikasi menggunakan sumber Kinesis untuk membaca dari aliran sumber. Cuplikan berikut ini membuat sumber Kinesis:

```
private def createSource: FlinkKinesisConsumer[String] = {
  val applicationProperties = KinesisAnalyticsRuntime.getApplicationProperties
  val inputProperties = applicationProperties.get("ConsumerConfigProperties")

  new FlinkKinesisConsumer[String](inputProperties.getProperty(streamNameKey,
 defaultInputStreamName),
    new SimpleStringSchema, inputProperties)
}
```

  Aplikasi ini juga menggunakan sink Kinesis untuk menulis ke dalam aliran hasil. Cuplikan berikut membuat sink Kinesis:

```
private def createSink: KinesisStreamsSink[String] = {
  val applicationProperties = KinesisAnalyticsRuntime.getApplicationProperties
  val outputProperties = applicationProperties.get("ProducerConfigProperties")

  KinesisStreamsSink.builder[String]
    .setKinesisClientProperties(outputProperties)
    .setSerializationSchema(new SimpleStringSchema)
    .setStreamName(outputProperties.getProperty(streamNameKey,
 defaultOutputStreamName))
    .setPartitionKeyGenerator((element: String) => String.valueOf(element.hashCode))
    .build
}
```

- Aplikasi ini menggunakan operator jendela untuk menemukan jumlah nilai untuk setiap simbol saham selama 5 detik jatuh jendela. Kode berikut membuat operator dan mengirimkan data agregat ke sink Kinesis Data Streams baru:

```
environment.addSource(createSource)
  .map { value =>
    val jsonNode = jsonParser.readValue(value, classOf[JsonNode])
    new Tuple2[String, Int](jsonNode.get("ticker").toString, 1)
  }
  .returns(Types.TUPLE(Types.STRING, Types.INT))
  .keyBy(v => v.f0) // Logically partition the stream for each ticker
```

```
    .window(TumblingProcessingTimeWindows.of(Time.seconds(10)))
    .sum(1) // Sum the number of tickers per partition
    .map { value => value.f0 + "," + value.f1.toString + "\n" }
    .sinkTo(createSink)
```

- Aplikasi ini membuat konektor sumber dan wastafel untuk mengakses sumber daya eksternal menggunakan StreamExecutionEnvironment objek.
- Aplikasi ini membuat konektor sumber dan wastafel menggunakan properti aplikasi dinamis. Properti aplikasi runtime dibaca untuk mengkonfigurasi konektor. Untuk informasi selengkapnya tentang properti runtime, lihat Properti Runtime.

Kompilasi dan unggah kode aplikasi

Di bagian ini, Anda mengkompilasi dan mengunggah kode aplikasi ke bucket Amazon S3.

Kompilasi Kode Aplikasi

Gunakan alat build SBT untuk membangun kode Scala untuk aplikasi. Untuk menginstal SBT, lihat Menginstal sbt dengan pengaturan cs. Anda juga perlu menginstal Java Development Kit (JDK). Lihat Prasyarat untuk Menyelesaikan Latihan.

1. Untuk menggunakan kode aplikasi Anda, Anda mengompilasi dan mengemasnya ke dalam file JAR. Anda dapat mengkompilasi dan mengemas kode Anda dengan SBT:

   ```
   sbt assembly
   ```

2. Jika aplikasi berhasil mengompilasi, file berikut dibuat:

   ```
   target/scala-3.2.0/tumbling-window-scala-1.0.jar
   ```

Unggah Kode Scala Streaming Apache Flink

Di bagian ini, Anda membuat bucket Amazon S3 dan mengunggah kode aplikasi Anda.

1. Buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.
2. Pilih Buat ember
3. Masukkan `ka-app-code-<username>` di bidang Bucket name (Nama bucket). Tambahkan sufiks ke nama bucket, seperti nama pengguna Anda, untuk membuatnya unik secara global. Pilih Next (Selanjutnya).

4. Di opsi Konfigurasi, pertahankan pengaturan apa adanya, dan pilih Berikutnya.
5. Di Setel izin, pertahankan pengaturan apa adanya, dan pilih Berikutnya.
6. Pilih Buat bucket.
7. Pilih `ka-app-code-<username>` bucket, lalu pilih Unggah.
8. Di langkah Pilih file, pilih Add files (Tambahkan berkas). Navigasikan ke file `tumbling-window-scala-1.0.jar` yang Anda buat di langkah sebelumnya.
9. Anda tidak perlu mengubah pengaturan apa pun untuk objek, jadi pilih Upload (Unggah).

Kode aplikasi Anda sekarang disimpan di bucket Amazon S3 yang dapat diakses aplikasi Anda.

Buat dan jalankan Aplikasi (konsol)

Ikuti langkah-langkah ini untuk membuat, mengonfigurasi, memperbarui, dan menjalankan aplikasi menggunakan konsol.

Buat Aplikasi

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/flink
2. Pada dashboard Managed Service for Apache Flink, pilih Create Analytics Application.
3. Pada Layanan Terkelola untuk Apache Flink - Buat halaman aplikasi, berikan detail aplikasi sebagai berikut:
   - Untuk Application name (Nama aplikasi), masukkan **`MyApplication`**.
   - Untuk Description (Deskripsi), masukkan **`My Scala test app`**.
   - Untuk Runtime, pilih Apache Flink.
   - Tinggalkan versi sebagai Apache Flink versi 1.15.2 (Versi yang disarankan).
4. Untuk Access permissions (Izin akses), pilih Create / update IAM role **`kinesis-analytics-MyApplication-us-west-2`** (Buat/perbarui IAM role).
5. Pilih Create application (Buat aplikasi).

> **Note**
>
> Saat membuat aplikasi Managed Service for Apache Flink menggunakan konsol, Anda memiliki opsi untuk membuat peran dan kebijakan IAM untuk aplikasi Anda. Aplikasi Anda

menggunakan peran dan kebijakan ini untuk mengakses sumber daya dependen. Sumber daya IAM ini diberi nama menggunakan nama aplikasi dan Wilayah sebagai berikut:

- Kebijakan: `kinesis-analytics-service-MyApplication-us-west-2`
- Peran: `kinesisanalytics-MyApplication-us-west-2`

Konfigurasikan Aplikasi

Gunakan prosedur berikut untuk mengonfigurasi aplikasi.

Untuk mengonfigurasi aplikasi

1. Pada MyApplicationhalaman, pilih Konfigurasi.
2. Di halaman Konfigurasikan aplikasi, berikan Code location (Lokasi kode):
   - Untuk Bucket Amazon S3, masukkan **ka-app-code-*<username>***.
   - Untuk Jalur ke objek Amazon S3, masukkan **tumbling-window-scala-1.0.jar**.
3. Di bawah Akses ke sumber daya aplikasi, untuk Access permissions (Izin akses), pilih Create / update IAM role **kinesis-analytics-MyApplication-us-west-2** (Pilih/perbarui IAM role).
4. Di bawah Properties (Properti), pilih Add group (Tambahkan grup).
5. Masukkan yang berikut ini:

ID Grup	Kunci	Nilai
ConsumerConfigProperties	**input.stream.name**	**ExampleInputStream**
ConsumerConfigProperties	**aws.region**	**us-west-2**
ConsumerConfigProperties	**flink.stream.initpos**	**LATEST**

   Pilih Simpan.
6. Di bawah Properties (Properti), pilih Add group (Tambahkan grup) lagi.

7. Masukkan yang berikut ini:

ID Grup	Kunci	Nilai
ProducerConfigProperties	**output.stream.name**	**ExampleOutputStream**
ProducerConfigProperties	**aws.region**	**us-west-2**

8. Di bawah Pemantauan, pastikan Memantau tingkat metrik diatur ke Aplikasi.
9. Untuk CloudWatch logging, pilih kotak centang Aktifkan.
10. Pilih Perbarui.

> **Note**
>
> Saat Anda memilih untuk mengaktifkan CloudWatch pencatatan Amazon, Layanan Terkelola untuk Apache Flink membuat grup log dan aliran log untuk Anda. Nama-nama sumber daya ini adalah sebagai berikut:
>
> - Grup log: `/aws/kinesis-analytics/MyApplication`
> - Aliran log: `kinesis-analytics-log-stream`

Edit Kebijakan IAM

Edit kebijakan IAM untuk menambahkan izin agar dapat mengakses bucket Amazon S3.

Untuk mengedit kebijakan IAM agar dapat menambahkan izin bucket S3

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pilih Policies (Kebijakan). Pilih kebijakan **kinesis-analytics-service-MyApplication-us-west-2** yang dibuat konsol untuk Anda di bagian sebelumnya.
3. Di halaman Ringkasan, pilih Edit policy (Edit kebijakan). Pilih tab JSON.
4. Tambahkan bagian yang disorot dari contoh kebijakan berikut ke kebijakan. Ganti ID akun sampel (*012345678901*) dengan ID akun Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ka-app-code-username/tumbling-window-scala-1.0.jar"
            ]
        },
        {
            "Sid": "DescribeLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
            ]
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
```

```
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

Jalankan Aplikasi

Grafik pekerjaan Flink dapat dilihat dengan menjalankan aplikasi, membuka dasbor Apache Flink, dan memilih pekerjaan Flink yang diinginkan.

Hentikan Aplikasi

Untuk menghentikan aplikasi, pada MyApplicationhalaman, pilih Berhenti. Konfirmasikan tindakan.

Membuat dan menjalankan aplikasi (CLI)

Di bagian ini, Anda menggunakan AWS Command Line Interface untuk membuat dan menjalankan aplikasi Managed Service for Apache Flink. Gunakan AWS CLI perintah kinesisanalyticsv2 untuk membuat dan berinteraksi dengan Managed Service untuk aplikasi Apache Flink.

Membuat kebijakan izin

> **Note**
>
> Anda harus membuat kebijakan izin dan peran untuk aplikasi Anda. Jika Anda tidak membuat sumber daya IAM ini, aplikasi Anda tidak dapat mengakses data dan aliran log.

Pertama, Anda membuat kebijakan izin dengan dua pernyataan: satu yang memberikan izin untuk tindakan baca di aliran sumber, dan satu lagi yang memberikan izin untuk tindakan tulis di aliran sink. Anda selanjutnya melampirkan kebijakan ke IAM role (yang Anda buat di bagian berikutnya). Jadi, ketika Layanan Terkelola untuk Apache Flink mengasumsikan peran tersebut, layanan memiliki izin yang diperlukan untuk membaca dari aliran sumber dan menulis ke aliran wastafel.

Gunakan kode berikut untuk membuat kebijakan izin `AKReadSourceStreamWriteSinkStream`. Ganti **`username`** dengan nama pengguna yang Anda gunakan untuk membuat bucket Amazon S3 untuk menyimpan kode aplikasi. Ganti ID akun di Amazon Resource Names (ARN) **`(012345678901)`** dengan ID akun Anda. Peran eksekusi **`MF-stream-rw-role`** layanan harus disesuaikan dengan peran khusus pelanggan.

```
{
    "ApplicationName": "tumbling_window",
    "ApplicationDescription": "Scala tumbling window application",
    "RuntimeEnvironment": "FLINK-1_15",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-stream-rw-role",
    "ApplicationConfiguration": {
        "ApplicationCodeConfiguration": {
            "CodeContent": {
                "S3ContentLocation": {
                    "BucketARN": "arn:aws:s3:::ka-app-code-username",
                    "FileKey": "tumbling-window-scala-1.0.jar"
                }
            },
            "CodeContentType": "ZIPFILE"
        },
        "EnvironmentProperties":  {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2",
```

```
                    "stream.name" : "ExampleInputStream",
                    "flink.stream.initpos" : "LATEST"
                }
            },
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleOutputStream"
               }
            }
         ]
      }
    },
    "CloudWatchLoggingOptions": [
      {
         "LogStreamARN": "arn:aws:logs:us-west-2:012345678901:log-
group:MyApplication:log-stream:kinesis-analytics-log-stream"
      }
   ]
}
```

Untuk step-by-step petunjuk membuat kebijakan izin, lihat Tutorial: Membuat dan Melampirkan Kebijakan Terkelola Pelanggan Pertama Anda di Panduan Pengguna IAM.

Membuat peran IAM

Di bagian ini, Anda membuat peran IAM yang dapat diasumsikan oleh aplikasi Managed Service for Apache Flink untuk membaca aliran sumber dan menulis ke aliran sink.

Layanan Terkelola untuk Apache Flink tidak dapat mengakses aliran Anda tanpa izin. Anda memberikan izin ini melalui IAM role. Setiap IAM role memiliki dua kebijakan yang dilampirkan. Kebijakan kepercayaan memberikan izin Layanan Terkelola untuk Apache Flink untuk mengambil peran, dan kebijakan izin menentukan apa yang dapat dilakukan Layanan Terkelola untuk Apache Flink setelah mengambil peran.

Anda melampirkan kebijakan izin yang Anda buat di bagian sebelumnya ke peran ini.

Untuk membuat IAM role

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi, pilih Peran dan kemudian Buat Peran.

3. Di bawah Pilih jenis identitas tepercaya, pilih AWSLayanan
4. Di bawah Pilih layanan yang akan menggunakan peran ini, pilih Kinesis.
5. Di bawah Pilih kasus penggunaan Anda, pilih Layanan Terkelola untuk Apache Flink.
6. Pilih Selanjutnya: Izin.
7. Di halaman Lampirkan kebijakan izin, pilih Next: Review (Selanjutnya: Tinjau). Anda melampirkan kebijakan izin setelah Anda membuat peran tersebut.
8. Di halaman Buat peran, masukkan **`MF-stream-rw-role`** untuk Role name (Nama peran). Pilih Create role (Buat peran).

   Sekarang Anda sudah membuat IAM role baru yang disebut `MF-stream-rw-role`. Selanjutnya, Anda memperbarui kebijakan kepercayaan dan izin untuk peran tersebut
9. Lampirkan kebijakan izin ke peran tersebut.

   > **Note**
   >
   > Untuk latihan ini, Managed Service for Apache Flink mengasumsikan peran ini untuk membaca data dari aliran data Kinesis (sumber) dan menulis output ke aliran data Kinesis lain. Jadi Anda melampirkan kebijakan yang Anda buat di langkah sebelumnya, Buat Kebijakan Izin.

   a. Di halaman Ringkasan, pilih tab Permissions (Izin).
   b. Pilih Attach Policies (Lampirkan Kebijakan).
   c. Di kotak pencarian, masukkan **`AKReadSourceStreamWriteSinkStream`** (kebijakan yang Anda buat bagian sebelumnya).
   d. Pilih `AKReadSourceStreamWriteSinkStream` kebijakan, lalu pilih Lampirkan kebijakan.

Anda sekarang telah membuat peran eksekusi layanan yang digunakan aplikasi Anda untuk mengakses sumber daya. Buat catatan ARN peran baru.

Untuk step-by-step petunjuk cara membuat peran, lihat Membuat Peran IAM (Konsol) di Panduan Pengguna IAM.

Buat aplikasi

Simpan kode JSON berikut ke file bernama `create_request.json`. Ganti ARN peran sampel dengan ARN untuk peran yang Anda buat sebelumnya. Ganti bucket akhiran ARN (nama pengguna) dengan akhiran yang Anda pilih di bagian sebelumnya. Ganti ID akun sampel (012345678901) di peran eksekusi layanan dengan ID akun Anda. `ServiceExecutionRole`Harus menyertakan peran pengguna IAM yang Anda buat di bagian sebelumnya.

```
"ApplicationName": "tumbling_window",
    "ApplicationDescription": "Scala getting started application",
    "RuntimeEnvironment": "FLINK-1_15",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-stream-rw-role",
    "ApplicationConfiguration": {
        "ApplicationCodeConfiguration": {
            "CodeContent": {
                "S3ContentLocation": {
                    "BucketARN": "arn:aws:s3:::ka-app-code-username",
                    "FileKey": "tumbling-window-scala-1.0.jar"
                }
            },
            "CodeContentType": "ZIPFILE"
        },
        "EnvironmentProperties":  {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleInputStream",
                    "flink.stream.initpos" : "LATEST"
               }
            },
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleOutputStream"
               }
            }
         ]
      }
    },
    "CloudWatchLoggingOptions": [
```

```
      {
         "LogStreamARN": "arn:aws:logs:us-west-2:012345678901:log-
group:MyApplication:log-stream:kinesis-analytics-log-stream"
      }
   ]
}
```

Jalankan CreateApplicationdengan permintaan berikut untuk membuat aplikasi:

```
aws kinesisanalyticsv2 create-application --cli-input-json file://create_request.json
```

Aplikasi ini sekarang dibuat. Anda memulai aplikasi di langkah berikutnya.

Mulai Aplikasi

Di bagian ini, Anda menggunakan tindakan StartApplication untuk memulai aplikasi.

Untuk memulai aplikasi

1. Simpan kode JSON berikut ke file bernama `start_request.json`.

   ```
   {
       "ApplicationName": "tumbling_window",
       "RunConfiguration": {
           "ApplicationRestoreConfiguration": {
            "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"
            }
       }
   }
   ```

2. Jalankan tindakan `StartApplication` dengan permintaan sebelumnya untuk memulai aplikasi:

   ```
   aws kinesisanalyticsv2 start-application --cli-input-json file://start_request.json
   ```

Aplikasi sekarang berjalan. Anda dapat memeriksa metrik Layanan Terkelola untuk Apache Flink di CloudWatch konsol Amazon untuk memverifikasi bahwa aplikasi berfungsi.

Hentikan Aplikasi

Di bagian ini, Anda menggunakan tindakan StopApplication untuk menghentikan aplikasi.

Untuk menghentikan aplikasi

1. Simpan kode JSON berikut ke file bernama `stop_request.json`.

```
{
   "ApplicationName": "tumbling_window"
}
```

2. Jalankan `StopApplication` tindakan dengan permintaan sebelumnya untuk menghentikan aplikasi:

```
aws kinesisanalyticsv2 stop-application --cli-input-json file://stop_request.json
```

Aplikasi sekarang dihentikan.

Tambahkan Opsi CloudWatch Logging

Anda dapat menggunakan AWS CLI untuk menambahkan aliran CloudWatch log Amazon ke aplikasi Anda. Untuk informasi tentang menggunakan CloudWatch Log dengan aplikasi Anda, lihat Menyiapkan Pencatatan Aplikasi.

Perbarui Properti Lingkungan

Di bagian ini, Anda menggunakan tindakan UpdateApplication untuk mengubah properti lingkungan untuk aplikasi tanpa mengompilasi ulang kode aplikasi. Dalam contoh ini, Anda mengubah Wilayah aliran sumber dan tujuan.

Untuk memperbarui properti lingkungan untuk aplikasi

1. Simpan kode JSON berikut ke file bernama `update_properties_request.json`.

```
{"ApplicationName": "tumbling_window",
   "CurrentApplicationVersionId": 1,
   "ApplicationConfigurationUpdate": {
      "EnvironmentPropertyUpdates": {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleInputStream",
```

```
                    "flink.stream.initpos" : "LATEST"
                }
            },
            {
                "PropertyGroupId": "ProducerConfigProperties",
                "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleOutputStream"
                }
            }
        ]
      }
   }
}
```

2. Jalankan tindakan `UpdateApplication` dengan permintaan sebelumnya untuk memperbarui properti lingkungan:

```
aws kinesisanalyticsv2 update-application --cli-input-json file://
update_properties_request.json
```

Perbarui kode aplikasi

Ketika Anda perlu memperbarui kode aplikasi Anda dengan versi baru dari paket kode Anda, Anda menggunakan tindakan UpdateApplicationCLI.

> **Note**
>
> Untuk memuat versi baru kode aplikasi dengan nama file yang sama, Anda harus menentukan versi objek baru. Untuk informasi selengkapnya tentang menggunakan versi objek Amazon S3, lihat Mengaktifkan dan Menonaktifkan Versioning.

Untuk menggunakan AWS CLI, hapus paket kode Anda sebelumnya dari bucket Amazon S3 Anda, unggah versi baru, dan panggil `UpdateApplication`, yang menentukan bucket Amazon S3 dan nama objek yang sama, serta versi objek baru. Aplikasi akan memulai ulang dengan paket kode baru.

Permintaan sampel berikut untuk tindakan `UpdateApplication` memuat ulang kode aplikasi dan memulai ulang aplikasi. Perbarui `CurrentApplicationVersionId` ke versi aplikasi saat ini. Anda dapat memeriksa versi aplikasi saat ini menggunakan tindakan `ListApplications` atau

`DescribeApplication`. Perbarui akhiran nama bucket (<username>) dengan akhiran yang Anda pilih di bagian. Buat Sumber Daya Dependen

```
{
    "ApplicationName": "tumbling_window",
    "CurrentApplicationVersionId": 1,
    "ApplicationConfigurationUpdate": {
        "ApplicationCodeConfigurationUpdate": {
            "CodeContentUpdate": {
                "S3ContentLocationUpdate": {
                    "BucketARNUpdate": "arn:aws:s3:::ka-app-code-username",
                    "FileKeyUpdate": "tumbling-window-scala-1.0.jar",
                    "ObjectVersionUpdate": "SAMPLEUehYngP87ex1nzYIGYgfhypvDU"
                }
            }
        }
    }
}
```

Pembersihan Sumber Daya AWS

Bagian ini mencakup prosedur untuk membersihkan AWS sumber daya yang dibuat dalam tutorial Tumbling Window.

Topik ini berisi bagian-bagian berikut:

- Hapus Layanan Terkelola Anda untuk Aplikasi Apache Flink
- Hapus Kinesis Data Streams Anda
- Hapus Objek dan Bucket Amazon S3 Anda
- Hapus Sumber Daya IAM Anda
- Hapus CloudWatch Sumber Daya Anda

Hapus Layanan Terkelola Anda untuk Aplikasi Apache Flink

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/flink
2. di panel Managed Service for Apache Flink, pilih. MyApplication
3. Di halaman aplikasi, pilih Delete (Hapus), lalu konfirmasikan penghapusan.

Hapus Kinesis Data Streams Anda

1. Buka konsol Kinesis di https://console.aws.amazon.com/kinesis.
2. Di panel Kinesis Data Streams, pilih. ExampleInputStream
3. Di ExampleInputStreamhalaman, pilih Hapus Stream Kinesis dan kemudian konfirmasikan penghapusan.
4. Di halaman Kinesis streams, pilih, pilih Tindakan ExampleOutputStream, pilih Hapus, lalu konfirmasikan penghapusan.

Hapus Objek dan Bucket Amazon S3 Anda

1. Buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.
2. Pilih ka-app-code- ember. <username>
3. Pilih Delete (Hapus), lalu masukkan nama bucket untuk mengonfirmasi penghapusan.

Hapus Sumber Daya IAM Anda

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di bilah navigasi, pilih Policies (Kebijakan).
3. Di kontrol filter, masukkan kinesis.
4. Pilih kebijakan kinesis-analytics-service- MyApplication -us-west-2.
5. Pilih Policy Actions (Tindakan Kebijakan), lalu pilih Delete (Hapus).
6. Di bilah navigasi, pilih Roles (Peran).
7. Pilih peran kinesis-analytics- -us-west-2. MyApplication
8. Pilih Delete role (Hapus peran), lalu konfirmasi penghapusan.

Hapus CloudWatch Sumber Daya Anda

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di bilah navigasi, pilih Logs.
3. Pilih grup log MyApplication/aws/kinesis-analytics/.
4. Pilih Delete Log Group (Hapus Grup Log), lalu konfirmasi penghapusan.

Contoh: Membuat Jendela Geser di Scala

> **Note**
>
> Mulai dari versi 1.15 Flink gratis Scala. Aplikasi sekarang dapat menggunakan Java API dari versi Scala apa pun. Flink masih menggunakan Scala di beberapa komponen kunci secara internal tetapi tidak mengekspos Scala ke dalam classloader kode pengguna. Karena itu, pengguna perlu menambahkan dependensi Scala ke dalam arsip jar mereka.
> Untuk informasi selengkapnya tentang perubahan Scala di Flink 1.15, lihat Scala Free in One Fifteen.

Dalam latihan ini, Anda akan membuat aplikasi streaming sederhana yang menggunakan Scala 3.2.0 dan Java API Flink. DataStream Aplikasi membaca data dari aliran Kinesis, menggabungkannya menggunakan jendela geser dan menulis hasil ke output Kinesis stream.

> **Note**
>
> Untuk mengatur prasyarat yang diperlukan untuk latihan ini, pertama-tama selesaikan latihan Memulai (Scala).

Topik ini berisi bagian-bagian berikut:

- Unduh dan Periksa Kode Aplikasi
- Kompilasi dan unggah kode aplikasi
- Buat dan jalankan Aplikasi (konsol)
- Membuat dan menjalankan aplikasi (CLI)
- Perbarui kode aplikasi
- Pembersihan Sumber Daya AWS

Unduh dan Periksa Kode Aplikasi

Kode aplikasi Python untuk contoh ini tersedia dari. GitHub Untuk mengunduh kode aplikasi, lakukan hal berikut:

1. Instal klien Git jika Anda belum menginstalnya. Untuk informasi selengkapnya, lihat Menginstal Git.
2. Klon repositori jarak jauh dengan perintah berikut:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Buka direktori `amazon-kinesis-data-analytics-java-examples/scala/SlidingWindow` tersebut.

Perhatikan hal tentang kode aplikasi berikut:

- `build.sbt`File berisi informasi tentang konfigurasi dan dependensi aplikasi, termasuk Layanan Terkelola untuk pustaka Apache Flink.
- `BasicStreamingJob.scala`File berisi metode utama yang mendefinisikan fungsionalitas aplikasi.
- Aplikasi menggunakan sumber Kinesis untuk membaca dari aliran sumber. Cuplikan berikut ini membuat sumber Kinesis:

```
private def createSource: FlinkKinesisConsumer[String] = {
  val applicationProperties = KinesisAnalyticsRuntime.getApplicationProperties
  val inputProperties = applicationProperties.get("ConsumerConfigProperties")

  new FlinkKinesisConsumer[String](inputProperties.getProperty(streamNameKey,
 defaultInputStreamName),
    new SimpleStringSchema, inputProperties)
}
```

  Aplikasi ini juga menggunakan sink Kinesis untuk menulis ke dalam aliran hasil. Cuplikan berikut membuat sink Kinesis:

```
private def createSink: KinesisStreamsSink[String] = {
  val applicationProperties = KinesisAnalyticsRuntime.getApplicationProperties
  val outputProperties = applicationProperties.get("ProducerConfigProperties")

  KinesisStreamsSink.builder[String]
    .setKinesisClientProperties(outputProperties)
    .setSerializationSchema(new SimpleStringSchema)
    .setStreamName(outputProperties.getProperty(streamNameKey,
 defaultOutputStreamName))
```

```
    .setPartitionKeyGenerator((element: String) => String.valueOf(element.hashCode))
    .build
}
```

- Aplikasi ini menggunakan operator jendela untuk menemukan jumlah nilai untuk setiap simbol saham selama jendela 10 detik yang meluncur 5 detik. Kode berikut membuat operator dan mengirimkan data agregat ke sink Kinesis Data Streams baru:

```
environment.addSource(createSource)
    .map { value =>
      val jsonNode = jsonParser.readValue(value, classOf[JsonNode])
      new Tuple2[String, Double](jsonNode.get("ticker").toString,
 jsonNode.get("price").asDouble)
    }
    .returns(Types.TUPLE(Types.STRING, Types.DOUBLE))
    .keyBy(v => v.f0) // Logically partition the stream for each word
    .window(SlidingProcessingTimeWindows.of(Time.seconds(10), Time.seconds(5)))
    .min(1) // Calculate minimum price per ticker over the window
    .map { value => value.f0 + String.format(",%.2f", value.f1) + "\n" }
    .sinkTo(createSink)
```

- Aplikasi ini membuat konektor sumber dan wastafel untuk mengakses sumber daya eksternal menggunakan StreamExecutionEnvironment objek.
- Aplikasi ini membuat konektor sumber dan wastafel menggunakan properti aplikasi dinamis. Properti aplikasi runtime dibaca untuk mengkonfigurasi konektor. Untuk informasi selengkapnya tentang properti runtime, lihat Properti Runtime.

Kompilasi dan unggah kode aplikasi

Di bagian ini, Anda mengkompilasi dan mengunggah kode aplikasi ke bucket Amazon S3.

Kompilasi Kode Aplikasi

Gunakan alat build SBT untuk membangun kode Scala untuk aplikasi. Untuk menginstal SBT, lihat Menginstal sbt dengan pengaturan cs. Anda juga perlu menginstal Java Development Kit (JDK). Lihat Prasyarat untuk Menyelesaikan Latihan.

1. Untuk menggunakan kode aplikasi Anda, Anda mengompilasi dan mengemasnya ke dalam file JAR. Anda dapat mengkompilasi dan mengemas kode Anda dengan SBT:

```
sbt assembly
```

2. Jika aplikasi berhasil mengompilasi, file berikut dibuat:

```
target/scala-3.2.0/sliding-window-scala-1.0.jar
```

Unggah Kode Scala Streaming Apache Flink

Di bagian ini, Anda membuat bucket Amazon S3 dan mengunggah kode aplikasi Anda.

1. Buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.
2. Pilih Buat ember
3. Masukkan `ka-app-code-<username>` di bidang Bucket name (Nama bucket). Tambahkan sufiks ke nama bucket, seperti nama pengguna Anda, untuk membuatnya unik secara global. Pilih Next (Selanjutnya).
4. Di opsi Konfigurasi, pertahankan pengaturan apa adanya, dan pilih Berikutnya.
5. Di Setel izin, pertahankan pengaturan apa adanya, dan pilih Berikutnya.
6. Pilih Buat bucket.
7. Pilih `ka-app-code-<username>` bucket, lalu pilih Unggah.
8. Di langkah Pilih file, pilih Add files (Tambahkan berkas). Navigasikan ke file `sliding-window-scala-1.0.jar` yang Anda buat di langkah sebelumnya.
9. Anda tidak perlu mengubah pengaturan apa pun untuk objek, jadi pilih Upload (Unggah).

Kode aplikasi Anda sekarang disimpan di bucket Amazon S3 yang dapat diakses aplikasi Anda.

Buat dan jalankan Aplikasi (konsol)

Ikuti langkah-langkah ini untuk membuat, mengonfigurasi, memperbarui, dan menjalankan aplikasi menggunakan konsol.

Buat Aplikasi

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/flink
2. Pada dashboard Managed Service for Apache Flink, pilih Create Analytics Application.

3. Pada Layanan Terkelola untuk Apache Flink - Buat halaman aplikasi, berikan detail aplikasi sebagai berikut:
   - Untuk Application name (Nama aplikasi), masukkan **`MyApplication`**.
   - Untuk Description (Deskripsi), masukkan **`My Scala test app`**.
   - Untuk Runtime, pilih Apache Flink.
   - Tinggalkan versi sebagai Apache Flink versi 1.15.2 (Versi yang disarankan).
4. Untuk Access permissions (Izin akses), pilih Create / update IAM role **`kinesis-analytics-MyApplication-us-west-2`** (Buat/perbarui IAM role).
5. Pilih Create application (Buat aplikasi).

> **Note**
>
> Saat membuat aplikasi Managed Service for Apache Flink menggunakan konsol, Anda memiliki opsi untuk membuat peran dan kebijakan IAM untuk aplikasi Anda. Aplikasi Anda menggunakan peran dan kebijakan ini untuk mengakses sumber daya dependen. Sumber daya IAM ini diberi nama menggunakan nama aplikasi dan Wilayah sebagai berikut:
>
> - Kebijakan: `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Peran: `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Konfigurasikan Aplikasi

Gunakan prosedur berikut untuk mengonfigurasi aplikasi.

Untuk mengonfigurasi aplikasi

1. Pada MyApplicationhalaman, pilih Konfigurasi.
2. Di halaman Konfigurasikan aplikasi, berikan Code location (Lokasi kode):
   - Untuk Bucket Amazon S3, masukkan **`ka-app-code-`*`<username>`***.
   - Untuk Jalur ke objek Amazon S3, masukkan **`sliding-window-scala-1.0.jar.`**.
3. Di bawah Akses ke sumber daya aplikasi, untuk Access permissions (Izin akses), pilih Create / update IAM role **`kinesis-analytics-MyApplication-us-west-2`** (Pilih/perbarui IAM role).

4. Di bawah Properties (Properti), pilih Add group (Tambahkan grup).
5. Masukkan yang berikut ini:

ID Grup	Kunci	Nilai
ConsumerConfigProperties	**input.stream.name**	**ExampleInputStream**
ConsumerConfigProperties	**aws.region**	**us-west-2**
ConsumerConfigProperties	**flink.stream.initpos**	**LATEST**

   Pilih Simpan.
6. Di bawah Properties (Properti), pilih Add group (Tambahkan grup) lagi.
7. Masukkan yang berikut ini:

ID Grup	Kunci	Nilai
ProducerConfigProperties	**output.stream.name**	**ExampleOutputStream**
ProducerConfigProperties	**aws.region**	**us-west-2**

8. Di bawah Pemantauan, pastikan Memantau tingkat metrik diatur ke Aplikasi.
9. Untuk CloudWatch logging, pilih kotak centang Aktifkan.
10. Pilih Perbarui.

> **Note**
>
> Saat Anda memilih untuk mengaktifkan CloudWatch pencatatan Amazon, Layanan Terkelola untuk Apache Flink membuat grup log dan aliran log untuk Anda. Nama-nama sumber daya ini adalah sebagai berikut:

- Grup log: `/aws/kinesis-analytics/MyApplication`
- Aliran log: `kinesis-analytics-log-stream`

Edit Kebijakan IAM

Edit kebijakan IAM untuk menambahkan izin agar dapat mengakses bucket Amazon S3.

Untuk mengedit kebijakan IAM agar dapat menambahkan izin bucket S3

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pilih Policies (Kebijakan). Pilih kebijakan **`kinesis-analytics-service-MyApplication-us-west-2`** yang dibuat konsol untuk Anda di bagian sebelumnya.
3. Di halaman Ringkasan, pilih Edit policy (Edit kebijakan). Pilih tab JSON.
4. Tambahkan bagian yang disorot dari contoh kebijakan berikut ke kebijakan. Ganti ID akun sampel (*012345678901*) dengan ID akun Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ka-app-code-username/sliding-window-scala-1.0.jar"
            ]
        },
        {
            "Sid": "DescribeLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
```

```
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
            ]
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

Jalankan Aplikasi

Grafik pekerjaan Flink dapat dilihat dengan menjalankan aplikasi, membuka dasbor Apache Flink, dan memilih pekerjaan Flink yang diinginkan.

Hentikan Aplikasi

Untuk menghentikan aplikasi, pada MyApplicationhalaman, pilih Berhenti. Konfirmasikan tindakan.

Membuat dan menjalankan aplikasi (CLI)

Di bagian ini, Anda menggunakan AWS Command Line Interface untuk membuat dan menjalankan aplikasi Managed Service for Apache Flink. Gunakan AWS CLI perintah kinesisanalyticsv2 untuk membuat dan berinteraksi dengan Managed Service untuk aplikasi Apache Flink.

Membuat kebijakan izin

> **Note**
>
> Anda harus membuat kebijakan izin dan peran untuk aplikasi Anda. Jika Anda tidak membuat sumber daya IAM ini, aplikasi Anda tidak dapat mengakses data dan aliran log.

Pertama, Anda membuat kebijakan izin dengan dua pernyataan: satu yang memberikan izin untuk tindakan baca di aliran sumber, dan satu lagi yang memberikan izin untuk tindakan tulis di aliran sink. Anda selanjutnya melampirkan kebijakan ke IAM role (yang Anda buat di bagian berikutnya). Jadi, ketika Layanan Terkelola untuk Apache Flink mengasumsikan peran tersebut, layanan memiliki izin yang diperlukan untuk membaca dari aliran sumber dan menulis ke aliran wastafel.

Gunakan kode berikut untuk membuat kebijakan izin `AKReadSourceStreamWriteSinkStream`. Ganti **`username`** dengan nama pengguna yang Anda gunakan untuk membuat bucket Amazon S3 untuk menyimpan kode aplikasi. Ganti ID akun di Amazon Resource Names (ARN) **`(012345678901)`** dengan ID akun Anda.

```
{
    "ApplicationName": "sliding_window",
    "ApplicationDescription": "Scala sliding window application",
    "RuntimeEnvironment": "FLINK-1_15",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-stream-rw-role",
    "ApplicationConfiguration": {
        "ApplicationCodeConfiguration": {
```

```
            "CodeContent": {
                "S3ContentLocation": {
                    "BucketARN": "arn:aws:s3:::ka-app-code-username",
                    "FileKey": "sliding-window-scala-1.0.jar"
                }
            },
            "CodeContentType": "ZIPFILE"
        },
        "EnvironmentProperties":  {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleInputStream",
                    "flink.stream.initpos" : "LATEST"
               }
            },
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleOutputStream"
               }
            }
         ]
      }
    },
    "CloudWatchLoggingOptions": [
      {
         "LogStreamARN": "arn:aws:logs:us-west-2:012345678901:log-
group:MyApplication:log-stream:kinesis-analytics-log-stream"
      }
   ]
}
```

Untuk step-by-step petunjuk membuat kebijakan izin, lihat Tutorial: Membuat dan Melampirkan Kebijakan Terkelola Pelanggan Pertama Anda di Panduan Pengguna IAM.

Membuat peran IAM

Di bagian ini, Anda membuat peran IAM yang dapat diasumsikan oleh aplikasi Managed Service for Apache Flink untuk membaca aliran sumber dan menulis ke aliran sink.

Layanan Terkelola untuk Apache Flink tidak dapat mengakses aliran Anda tanpa izin. Anda memberikan izin ini melalui IAM role. Setiap IAM role memiliki dua kebijakan yang dilampirkan. Kebijakan kepercayaan memberikan izin Layanan Terkelola untuk Apache Flink untuk mengambil peran, dan kebijakan izin menentukan apa yang dapat dilakukan Layanan Terkelola untuk Apache Flink setelah mengambil peran.

Anda melampirkan kebijakan izin yang Anda buat di bagian sebelumnya ke peran ini.

Untuk membuat IAM role

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi, pilih Peran dan kemudian Buat Peran.
3. Di bawah Pilih jenis identitas tepercaya, pilih AWSLayanan
4. Di bawah Pilih layanan yang akan menggunakan peran ini, pilih Kinesis.
5. Di bawah Pilih kasus penggunaan Anda, pilih Layanan Terkelola untuk Apache Flink.
6. Pilih Selanjutnya: Izin.
7. Di halaman Lampirkan kebijakan izin, pilih Next: Review (Selanjutnya: Tinjau). Anda melampirkan kebijakan izin setelah Anda membuat peran tersebut.
8. Di halaman Buat peran, masukkan **`MF-stream-rw-role`** untuk Role name (Nama peran). Pilih Create role (Buat peran).

   Sekarang Anda sudah membuat IAM role baru yang disebut `MF-stream-rw-role`. Selanjutnya, Anda memperbarui kebijakan kepercayaan dan izin untuk peran tersebut
9. Lampirkan kebijakan izin ke peran tersebut.

   > **Note**
   >
   > Untuk latihan ini, Managed Service for Apache Flink mengasumsikan peran ini untuk membaca data dari aliran data Kinesis (sumber) dan menulis output ke aliran data Kinesis lain. Jadi Anda melampirkan kebijakan yang Anda buat di langkah sebelumnya, Buat Kebijakan Izin.

   a. Di halaman Ringkasan, pilih tab Permissions (Izin).

   b. Pilih Attach Policies (Lampirkan Kebijakan).

c. Di kotak pencarian, masukkan **`AKReadSourceStreamWriteSinkStream`** (kebijakan yang Anda buat bagian sebelumnya).

d. Pilih `AKReadSourceStreamWriteSinkStream` kebijakan, lalu pilih Lampirkan kebijakan.

Anda sekarang telah membuat peran eksekusi layanan yang digunakan aplikasi Anda untuk mengakses sumber daya. Buat catatan ARN peran baru.

Untuk step-by-step petunjuk cara membuat peran, lihat Membuat Peran IAM (Konsol) di Panduan Pengguna IAM.

Buat aplikasi

Simpan kode JSON berikut ke file bernama `create_request.json`. Ganti ARN peran sampel dengan ARN untuk peran yang Anda buat sebelumnya. Ganti bucket akhiran ARN (nama pengguna) dengan akhiran yang Anda pilih di bagian sebelumnya. Ganti ID akun sampel (012345678901) di peran eksekusi layanan dengan ID akun Anda.

```
{
    "ApplicationName": "sliding_window",
    "ApplicationDescription": "Scala sliding_window application",
    "RuntimeEnvironment": "FLINK-1_15",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-stream-rw-role",
    "ApplicationConfiguration": {
        "ApplicationCodeConfiguration": {
            "CodeContent": {
                "S3ContentLocation": {
                    "BucketARN": "arn:aws:s3:::ka-app-code-username",
                    "FileKey": "sliding-window-scala-1.0.jar"
                }
            },
            "CodeContentType": "ZIPFILE"
        },
        "EnvironmentProperties":  {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleInputStream",
                    "flink.stream.initpos" : "LATEST"
               }
```

```
            },
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleOutputStream"
               }
            }
         ]
      }
    },
    "CloudWatchLoggingOptions": [
      {
         "LogStreamARN": "arn:aws:logs:us-west-2:012345678901:log-
group:MyApplication:log-stream:kinesis-analytics-log-stream"
      }
   ]
}
```

Jalankan CreateApplicationdengan permintaan berikut untuk membuat aplikasi:

```
aws kinesisanalyticsv2 create-application --cli-input-json file://create_request.json
```

Aplikasi ini sekarang dibuat. Anda memulai aplikasi di langkah berikutnya.

Mulai Aplikasi

Di bagian ini, Anda menggunakan tindakan StartApplication untuk memulai aplikasi.

Untuk memulai aplikasi

1. Simpan kode JSON berikut ke file bernama `start_request.json`.

```
{
    "ApplicationName": "sliding_window",
    "RunConfiguration": {
        "ApplicationRestoreConfiguration": {
         "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"
         }
    }
}
```

2. Jalankan tindakan `StartApplication` dengan permintaan sebelumnya untuk memulai aplikasi:

```
aws kinesisanalyticsv2 start-application --cli-input-json file://start_request.json
```

Aplikasi sekarang berjalan. Anda dapat memeriksa metrik Layanan Terkelola untuk Apache Flink di CloudWatch konsol Amazon untuk memverifikasi bahwa aplikasi berfungsi.

Hentikan Aplikasi

Di bagian ini, Anda menggunakan tindakan StopApplication untuk menghentikan aplikasi.

Untuk menghentikan aplikasi

1. Simpan kode JSON berikut ke file bernama `stop_request.json`.

```
{
   "ApplicationName": "sliding_window"
}
```

2. Jalankan `StopApplication` tindakan dengan permintaan sebelumnya untuk menghentikan aplikasi:

```
aws kinesisanalyticsv2 stop-application --cli-input-json file://stop_request.json
```

Aplikasi sekarang dihentikan.

Tambahkan Opsi CloudWatch Logging

Anda dapat menggunakan AWS CLI untuk menambahkan aliran CloudWatch log Amazon ke aplikasi Anda. Untuk informasi tentang menggunakan CloudWatch Log dengan aplikasi Anda, lihat Menyiapkan Pencatatan Aplikasi.

Perbarui Properti Lingkungan

Di bagian ini, Anda menggunakan tindakan UpdateApplication untuk mengubah properti lingkungan untuk aplikasi tanpa mengompilasi ulang kode aplikasi. Dalam contoh ini, Anda mengubah Wilayah aliran sumber dan tujuan.

Untuk memperbarui properti lingkungan untuk aplikasi

1. Simpan kode JSON berikut ke file bernama `update_properties_request.json`.

```
{"ApplicationName": "sliding_window",
   "CurrentApplicationVersionId": 1,
   "ApplicationConfigurationUpdate": {
      "EnvironmentPropertyUpdates": {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleInputStream",
                    "flink.stream.initpos" : "LATEST"
               }
            },
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleOutputStream"
               }
            }
         ]
      }
   }
}
```

2. Jalankan tindakan `UpdateApplication` dengan permintaan sebelumnya untuk memperbarui properti lingkungan:

```
aws kinesisanalyticsv2 update-application --cli-input-json file://
update_properties_request.json
```

Perbarui kode aplikasi

Ketika Anda perlu memperbarui kode aplikasi Anda dengan versi baru dari paket kode Anda, Anda menggunakan tindakan UpdateApplicationCLI.

> **Note**
>
> Untuk memuat versi baru kode aplikasi dengan nama file yang sama, Anda harus menentukan versi objek baru. Untuk informasi selengkapnya tentang menggunakan versi objek Amazon S3, lihat Mengaktifkan dan Menonaktifkan Versioning.

Untuk menggunakan AWS CLI, hapus paket kode Anda sebelumnya dari bucket Amazon S3 Anda, unggah versi baru, dan panggil `UpdateApplication`, yang menentukan bucket Amazon S3 dan nama objek yang sama, serta versi objek baru. Aplikasi akan memulai ulang dengan paket kode baru.

Permintaan sampel berikut untuk tindakan `UpdateApplication` memuat ulang kode aplikasi dan memulai ulang aplikasi. Perbarui `CurrentApplicationVersionId` ke versi aplikasi saat ini. Anda dapat memeriksa versi aplikasi saat ini menggunakan tindakan `ListApplications` atau `DescribeApplication`. Perbarui akhiran nama bucket (<username>) dengan akhiran yang Anda pilih di bagian. Buat Sumber Daya Dependen

```
{
    "ApplicationName": "sliding_window",
    "CurrentApplicationVersionId": 1,
    "ApplicationConfigurationUpdate": {
        "ApplicationCodeConfigurationUpdate": {
            "CodeContentUpdate": {
                "S3ContentLocationUpdate": {
                    "BucketARNUpdate": "arn:aws:s3:::ka-app-code-username",
                    "FileKeyUpdate": "-1.0.jar",
                    "ObjectVersionUpdate": "SAMPLEUehYngP87ex1nzYIGYgfhypvDU"
                }
            }
        }
    }
}
```

Pembersihan Sumber Daya AWS

Bagian ini mencakup prosedur untuk membersihkan AWS sumber daya yang dibuat dalam tutorial Jendela geser.

Topik ini berisi bagian-bagian berikut:

- Hapus Layanan Terkelola Anda untuk Aplikasi Apache Flink

- Hapus Kinesis Data Streams Anda
- Hapus Objek dan Bucket Amazon S3 Anda
- Hapus Sumber Daya IAM Anda
- Hapus CloudWatch Sumber Daya Anda

Hapus Layanan Terkelola Anda untuk Aplikasi Apache Flink

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/flink
2. di panel Managed Service for Apache Flink, pilih. MyApplication
3. Di halaman aplikasi, pilih Delete (Hapus), lalu konfirmasikan penghapusan.

Hapus Kinesis Data Streams Anda

1. Buka konsol Kinesis di https://console.aws.amazon.com/kinesis.
2. Di panel Kinesis Data Streams, pilih. ExampleInputStream
3. Di ExampleInputStreamhalaman, pilih Hapus Stream Kinesis dan kemudian konfirmasikan penghapusan.
4. Di halaman Kinesis streams, pilih, pilih Tindakan ExampleOutputStream, pilih Hapus, lalu konfirmasikan penghapusan.

Hapus Objek dan Bucket Amazon S3 Anda

1. Buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.
2. Pilih ka-app-code- ember. <username>
3. Pilih Delete (Hapus), lalu masukkan nama bucket untuk mengonfirmasi penghapusan.

Hapus Sumber Daya IAM Anda

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di bilah navigasi, pilih Policies (Kebijakan).
3. Di kontrol filter, masukkan kinesis.
4. Pilih kebijakan kinesis-analytics-service- MyApplication -us-west-2.
5. Pilih Policy Actions (Tindakan Kebijakan), lalu pilih Delete (Hapus).

6. Di bilah navigasi, pilih Roles (Peran).
7. Pilih peran kinesis-analytics- -us-west-2. MyApplication
8. Pilih Delete role (Hapus peran), lalu konfirmasi penghapusan.

Hapus CloudWatch Sumber Daya Anda

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di bilah navigasi, pilih Logs.
3. Pilih grup log MyApplication/aws/kinesis-analytics/.
4. Pilih Delete Log Group (Hapus Grup Log), lalu konfirmasi penghapusan.

Contoh: Kirim Data Streaming ke Amazon S3 di Scala

> **Note**
>
> Mulai dari versi 1.15 Flink gratis Scala. Aplikasi sekarang dapat menggunakan Java API dari versi Scala apa pun. Flink masih menggunakan Scala di beberapa komponen kunci secara internal tetapi tidak mengekspos Scala ke dalam classloader kode pengguna. Karena itu, pengguna perlu menambahkan dependensi Scala ke dalam arsip jar mereka.
> Untuk informasi selengkapnya tentang perubahan Scala di Flink 1.15, lihat Scala Free in One Fifteen.

Dalam latihan ini, Anda akan membuat aplikasi streaming sederhana yang menggunakan Scala 3.2.0 dan Java API Flink. DataStream Aplikasi membaca data dari aliran Kinesis, menggabungkannya menggunakan jendela geser dan menulis hasil ke S3.

> **Note**
>
> Untuk mengatur prasyarat yang diperlukan untuk latihan ini, pertama-tama selesaikan latihan Memulai (Scala). Anda hanya perlu membuat folder tambahan ***data/*** di bucket ka-app-code Amazon S3 -. <username>

Topik ini berisi bagian-bagian berikut:

- Unduh dan Periksa Kode Aplikasi

- Kompilasi dan unggah kode aplikasi
- Buat dan jalankan Aplikasi (konsol)
- Membuat dan menjalankan aplikasi (CLI)
- Perbarui kode aplikasi
- Pembersihan Sumber Daya AWS

Unduh dan Periksa Kode Aplikasi

Kode aplikasi Python untuk contoh ini tersedia dari. GitHub Untuk mengunduh kode aplikasi, lakukan hal berikut:

1. Instal klien Git jika Anda belum menginstalnya. Untuk informasi selengkapnya, lihat Menginstal Git.
2. Klon repositori jarak jauh dengan perintah berikut:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Buka direktori `amazon-kinesis-data-analytics-java-examples/scala/S3Sink` tersebut.

Perhatikan hal tentang kode aplikasi berikut:

- `build.sbt`File berisi informasi tentang konfigurasi dan dependensi aplikasi, termasuk Layanan Terkelola untuk pustaka Apache Flink.
- `BasicStreamingJob.scala`File berisi metode utama yang mendefinisikan fungsionalitas aplikasi.
- Aplikasi menggunakan sumber Kinesis untuk membaca dari aliran sumber. Cuplikan berikut ini membuat sumber Kinesis:

```
private def createSource: FlinkKinesisConsumer[String] = {
  val applicationProperties = KinesisAnalyticsRuntime.getApplicationProperties
  val inputProperties = applicationProperties.get("ConsumerConfigProperties")

  new FlinkKinesisConsumer[String](inputProperties.getProperty(streamNameKey,
 defaultInputStreamName),
    new SimpleStringSchema, inputProperties)
}
```

Aplikasi ini juga menggunakan a StreamingFileSink untuk menulis ke ember Amazon S3: `

```
def createSink: StreamingFileSink[String] = {
  val applicationProperties = KinesisAnalyticsRuntime.getApplicationProperties
  val s3SinkPath =
 applicationProperties.get("ProducerConfigProperties").getProperty("s3.sink.path")

  StreamingFileSink
    .forRowFormat(new Path(s3SinkPath), new SimpleStringEncoder[String]("UTF-8"))
    .build()
}
```

- Aplikasi ini membuat konektor sumber dan wastafel untuk mengakses sumber daya eksternal menggunakan StreamExecutionEnvironment objek.
- Aplikasi ini membuat konektor sumber dan wastafel menggunakan properti aplikasi dinamis. Properti aplikasi runtime dibaca untuk mengkonfigurasi konektor. Untuk informasi selengkapnya tentang properti runtime, lihat Properti Runtime.

Kompilasi dan unggah kode aplikasi

Di bagian ini, Anda mengkompilasi dan mengunggah kode aplikasi ke bucket Amazon S3.

Kompilasi Kode Aplikasi

Gunakan alat build SBT untuk membangun kode Scala untuk aplikasi. Untuk menginstal SBT, lihat Menginstal sbt dengan pengaturan cs. Anda juga perlu menginstal Java Development Kit (JDK). Lihat Prasyarat untuk Menyelesaikan Latihan.

1. Untuk menggunakan kode aplikasi Anda, Anda mengompilasi dan mengemasnya ke dalam file JAR. Anda dapat mengkompilasi dan mengemas kode Anda dengan SBT:

   ```
   sbt assembly
   ```

2. Jika aplikasi berhasil mengompilasi, file berikut dibuat:

   ```
   target/scala-3.2.0/s3-sink-scala-1.0.jar
   ```

Unggah Kode Scala Streaming Apache Flink

Di bagian ini, Anda membuat bucket Amazon S3 dan mengunggah kode aplikasi Anda.

1. Buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.
2. Pilih Buat ember
3. Masukkan `ka-app-code-<username>` di bidang Bucket name (Nama bucket). Tambahkan sufiks ke nama bucket, seperti nama pengguna Anda, untuk membuatnya unik secara global. Pilih Next (Selanjutnya).
4. Di opsi Konfigurasi, pertahankan pengaturan apa adanya, dan pilih Berikutnya.
5. Di Setel izin, pertahankan pengaturan apa adanya, dan pilih Berikutnya.
6. Pilih Buat bucket.
7. Pilih `ka-app-code-<username>` bucket, lalu pilih Unggah.
8. Di langkah Pilih file, pilih Add files (Tambahkan berkas). Navigasikan ke file `s3-sink-scala-1.0.jar` yang Anda buat di langkah sebelumnya.
9. Anda tidak perlu mengubah pengaturan apa pun untuk objek, jadi pilih Upload (Unggah).

Kode aplikasi Anda sekarang disimpan di bucket Amazon S3 yang dapat diakses aplikasi Anda.

Buat dan jalankan Aplikasi (konsol)

Ikuti langkah-langkah ini untuk membuat, mengonfigurasi, memperbarui, dan menjalankan aplikasi menggunakan konsol.

Buat Aplikasi

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/flink
2. Pada dashboard Managed Service for Apache Flink, pilih Create Analytics Application.
3. Pada Layanan Terkelola untuk Apache Flink - Buat halaman aplikasi, berikan detail aplikasi sebagai berikut:
   - Untuk Application name (Nama aplikasi), masukkan **`MyApplication`**.
   - Untuk Description (Deskripsi), masukkan **`My java test app`**.
   - Untuk Runtime, pilih Apache Flink.
   - Tinggalkan versi sebagai Apache Flink versi 1.15.2 (Versi yang disarankan).

4. Untuk Access permissions (Izin akses), pilih Create / update IAM role **kinesis-analytics-MyApplication-us-west-2** (Buat/perbarui IAM role).
5. Pilih Create application (Buat aplikasi).

> **Note**
>
> Saat membuat aplikasi Managed Service for Apache Flink menggunakan konsol, Anda memiliki opsi untuk membuat peran dan kebijakan IAM untuk aplikasi Anda. Aplikasi Anda menggunakan peran dan kebijakan ini untuk mengakses sumber daya dependen. Sumber daya IAM ini diberi nama menggunakan nama aplikasi dan Wilayah sebagai berikut:
>
> - Kebijakan: `kinesis-analytics-service-MyApplication-us-west-2`
> - Peran: `kinesisanalytics-MyApplication-us-west-2`

Konfigurasikan Aplikasi

Gunakan prosedur berikut untuk mengonfigurasi aplikasi.

Untuk mengonfigurasi aplikasi

1. Pada MyApplicationhalaman, pilih Konfigurasi.
2. Di halaman Konfigurasikan aplikasi, berikan Code location (Lokasi kode):
   - Untuk Bucket Amazon S3, masukkan **ka-app-code-*<username>***.
   - Untuk Jalur ke objek Amazon S3, masukkan **s3-sink-scala-1.0.jar**.
3. Di bawah Akses ke sumber daya aplikasi, untuk Access permissions (Izin akses), pilih Create / update IAM role **kinesis-analytics-MyApplication-us-west-2** (Pilih/perbarui IAM role).
4. Di bawah Properties (Properti), pilih Add group (Tambahkan grup).
5. Masukkan yang berikut ini:

ID Grup	Kunci	Nilai
ConsumerConfigProperties	**input.stream.name**	**ExampleInputStream**

ID Grup	Kunci	Nilai
ConsumerConfigProperties	**aws.region**	**us-west-2**
ConsumerConfigProperties	**flink.stream.initpos**	**LATEST**

Pilih Simpan.

6. Di bawah Properties (Properti), pilih Add group (Tambahkan grup).
7. Masukkan yang berikut ini:

ID Grup	Kunci	Nilai
ProducerConfigProperties	**s3.sink.path**	**s3a://ka-app-code-*<user-name>* /data**

8. Di bawah Pemantauan, pastikan Memantau tingkat metrik diatur ke Aplikasi.
9. Untuk CloudWatch logging, pilih kotak centang Aktifkan.
10. Pilih Perbarui.

> **Note**
>
> Saat Anda memilih untuk mengaktifkan CloudWatch pencatatan Amazon, Layanan Terkelola untuk Apache Flink membuat grup log dan aliran log untuk Anda. Nama-nama sumber daya ini adalah sebagai berikut:
>
> - Grup log: `/aws/kinesis-analytics/MyApplication`
> - Aliran log: `kinesis-analytics-log-stream`

Edit Kebijakan IAM

Edit kebijakan IAM untuk menambahkan izin agar dapat mengakses bucket Amazon S3.

Untuk mengedit kebijakan IAM agar dapat menambahkan izin bucket S3

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pilih Policies (Kebijakan). Pilih kebijakan **`kinesis-analytics-service-MyApplication-us-west-2`** yang dibuat konsol untuk Anda di bagian sebelumnya.
3. Di halaman Ringkasan, pilih Edit policy (Edit kebijakan). Pilih tab JSON.
4. Tambahkan bagian yang disorot dari contoh kebijakan berikut ke kebijakan. Ganti ID akun sampel (*012345678901*) dengan ID akun Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:Abort*",
                "s3:DeleteObject*",
                "s3:GetObject*",
                "s3:GetBucket*",
                "s3:List*",
                "s3:ListBucket",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::ka-app-code-<username>",
                "arn:aws:s3:::ka-app-code-<username>/*"
            ]
        },
        {
            "Sid": "DescribeLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
```

```
            "Action": [
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
            ]
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        }
    ]
}
```

Jalankan Aplikasi

Grafik pekerjaan Flink dapat dilihat dengan menjalankan aplikasi, membuka dasbor Apache Flink, dan memilih pekerjaan Flink yang diinginkan.

Hentikan Aplikasi

Untuk menghentikan aplikasi, pada MyApplicationhalaman, pilih Berhenti. Konfirmasikan tindakan.

Membuat dan menjalankan aplikasi (CLI)

Di bagian ini, Anda menggunakan AWS Command Line Interface untuk membuat dan menjalankan aplikasi Managed Service for Apache Flink. Gunakan AWS CLI perintah kinesisanalyticsv2 untuk membuat dan berinteraksi dengan Managed Service untuk aplikasi Apache Flink.

Membuat kebijakan izin

> **Note**
>
> Anda harus membuat kebijakan izin dan peran untuk aplikasi Anda. Jika Anda tidak membuat sumber daya IAM ini, aplikasi Anda tidak dapat mengakses data dan aliran log.

Pertama, Anda membuat kebijakan izin dengan dua pernyataan: satu yang memberikan izin untuk tindakan baca di aliran sumber, dan satu lagi yang memberikan izin untuk tindakan tulis di aliran sink. Anda selanjutnya melampirkan kebijakan ke IAM role (yang Anda buat di bagian berikutnya). Jadi, ketika Layanan Terkelola untuk Apache Flink mengasumsikan peran tersebut, layanan memiliki izin yang diperlukan untuk membaca dari aliran sumber dan menulis ke aliran wastafel.

Gunakan kode berikut untuk membuat kebijakan izin `AKReadSourceStreamWriteSinkStream`. Ganti **`username`** dengan nama pengguna yang Anda gunakan untuk membuat bucket Amazon S3 untuk menyimpan kode aplikasi. Ganti ID akun di Amazon Resource Names (ARN) **`(012345678901)`** dengan ID akun Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ka-app-code-username/getting-started-scala-1.0.jar"
            ]
        },
        {
            "Sid": "DescribeLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
```

```
            ]
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-analytics/
MyApplication:log-stream:*"
            ]
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-analytics/
MyApplication:log-stream:kinesis-analytics-log-stream"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

Untuk step-by-step petunjuk membuat kebijakan izin, lihat Tutorial: Membuat dan Melampirkan Kebijakan Terkelola Pelanggan Pertama Anda di Panduan Pengguna IAM.

Membuat peran IAM

Di bagian ini, Anda membuat peran IAM yang dapat diasumsikan oleh aplikasi Managed Service for Apache Flink untuk membaca aliran sumber dan menulis ke aliran sink.

Layanan Terkelola untuk Apache Flink tidak dapat mengakses aliran Anda tanpa izin. Anda memberikan izin ini melalui IAM role. Setiap IAM role memiliki dua kebijakan yang dilampirkan. Kebijakan kepercayaan memberikan izin Layanan Terkelola untuk Apache Flink untuk mengambil peran, dan kebijakan izin menentukan apa yang dapat dilakukan Layanan Terkelola untuk Apache Flink setelah mengambil peran.

Anda melampirkan kebijakan izin yang Anda buat di bagian sebelumnya ke peran ini.

Untuk membuat IAM role

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi, pilih Peran dan kemudian Buat Peran.
3. Di bawah Pilih jenis identitas tepercaya, pilih AWSLayanan
4. Di bawah Pilih layanan yang akan menggunakan peran ini, pilih Kinesis.
5. Di bawah Pilih kasus penggunaan Anda, pilih Layanan Terkelola untuk Apache Flink.
6. Pilih Selanjutnya: Izin.
7. Di halaman Lampirkan kebijakan izin, pilih Next: Review (Selanjutnya: Tinjau). Anda melampirkan kebijakan izin setelah Anda membuat peran tersebut.
8. Di halaman Buat peran, masukkan **`MF-stream-rw-role`** untuk Role name (Nama peran). Pilih Create role (Buat peran).

   Sekarang Anda sudah membuat IAM role baru yang disebut `MF-stream-rw-role`. Selanjutnya, Anda memperbarui kebijakan kepercayaan dan izin untuk peran tersebut
9. Lampirkan kebijakan izin ke peran tersebut.

> **Note**
>
> Untuk latihan ini, Managed Service for Apache Flink mengasumsikan peran ini untuk membaca data dari aliran data Kinesis (sumber) dan menulis output ke aliran data Kinesis lain. Jadi Anda melampirkan kebijakan yang Anda buat di langkah sebelumnya, Buat Kebijakan Izin.

a. Di halaman Ringkasan, pilih tab Permissions (Izin).
b. Pilih Attach Policies (Lampirkan Kebijakan).
c. Di kotak pencarian, masukkan **`AKReadSourceStreamWriteSinkStream`** (kebijakan yang Anda buat bagian sebelumnya).
d. Pilih `AKReadSourceStreamWriteSinkStream` kebijakan, lalu pilih Lampirkan kebijakan.

Anda sekarang telah membuat peran eksekusi layanan yang digunakan aplikasi Anda untuk mengakses sumber daya. Buat catatan ARN peran baru.

Untuk step-by-step petunjuk cara membuat peran, lihat Membuat Peran IAM (Konsol) di Panduan Pengguna IAM.

Buat aplikasi

Simpan kode JSON berikut ke file bernama `create_request.json`. Ganti ARN peran sampel dengan ARN untuk peran yang Anda buat sebelumnya. Ganti bucket akhiran ARN (nama pengguna) dengan akhiran yang Anda pilih di bagian sebelumnya. Ganti ID akun sampel (012345678901) di peran eksekusi layanan dengan ID akun Anda.

```
{
    "ApplicationName": "s3_sink",
    "ApplicationDescription": "Scala tumbling window application",
    "RuntimeEnvironment": "FLINK-1_15",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-stream-rw-role",
    "ApplicationConfiguration": {
        "ApplicationCodeConfiguration": {
            "CodeContent": {
                "S3ContentLocation": {
                    "BucketARN": "arn:aws:s3:::ka-app-code-username",
                    "FileKey": "s3-sink-scala-1.0.jar"
                }
            },
            "CodeContentType": "ZIPFILE"
        },
        "EnvironmentProperties":  {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
```

```
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleInputStream",
                    "flink.stream.initpos" : "LATEST"
                }
            },
            {
                "PropertyGroupId": "ProducerConfigProperties",
                "PropertyMap" : {
                    "s3.sink.path" : "s3a://ka-app-code-<username>/data"
                }
            }
        ]
      }
    },
    "CloudWatchLoggingOptions": [
      {
         "LogStreamARN": "arn:aws:logs:us-west-2:012345678901:log-
group:MyApplication:log-stream:kinesis-analytics-log-stream"
      }
   ]
}
```

Jalankan CreateApplicationdengan permintaan berikut untuk membuat aplikasi:

```
aws kinesisanalyticsv2 create-application --cli-input-json file://create_request.json
```

Aplikasi ini sekarang dibuat. Anda memulai aplikasi di langkah berikutnya.

Mulai Aplikasi

Di bagian ini, Anda menggunakan tindakan StartApplication untuk memulai aplikasi.

Untuk memulai aplikasi

1. Simpan kode JSON berikut ke file bernama `start_request.json`.

```
{{
    "ApplicationName": "s3_sink",
    "RunConfiguration": {
        "ApplicationRestoreConfiguration": {
         "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"
         }
    }
```

```
}
```

2. Jalankan tindakan `StartApplication` dengan permintaan sebelumnya untuk memulai aplikasi:

```
aws kinesisanalyticsv2 start-application --cli-input-json file://start_request.json
```

Aplikasi sekarang berjalan. Anda dapat memeriksa metrik Layanan Terkelola untuk Apache Flink di CloudWatch konsol Amazon untuk memverifikasi bahwa aplikasi berfungsi.

Hentikan Aplikasi

Di bagian ini, Anda menggunakan tindakan StopApplication untuk menghentikan aplikasi.

Untuk menghentikan aplikasi

1. Simpan kode JSON berikut ke file bernama `stop_request.json`.

```
{
   "ApplicationName": "s3_sink"
}
```

2. Jalankan `StopApplication` tindakan dengan permintaan sebelumnya untuk menghentikan aplikasi:

```
aws kinesisanalyticsv2 stop-application --cli-input-json file://stop_request.json
```

Aplikasi sekarang dihentikan.

Tambahkan Opsi CloudWatch Logging

Anda dapat menggunakan AWS CLI untuk menambahkan aliran CloudWatch log Amazon ke aplikasi Anda. Untuk informasi tentang menggunakan CloudWatch Log dengan aplikasi Anda, lihat Menyiapkan Pencatatan Aplikasi.

Perbarui Properti Lingkungan

Di bagian ini, Anda menggunakan tindakan UpdateApplication untuk mengubah properti lingkungan untuk aplikasi tanpa mengompilasi ulang kode aplikasi. Dalam contoh ini, Anda mengubah Wilayah aliran sumber dan tujuan.

Untuk memperbarui properti lingkungan untuk aplikasi

1. Simpan kode JSON berikut ke file bernama `update_properties_request.json`.

```
{"ApplicationName": "s3_sink",
   "CurrentApplicationVersionId": 1,
   "ApplicationConfigurationUpdate": {
      "EnvironmentPropertyUpdates": {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleInputStream",
                    "flink.stream.initpos" : "LATEST"
               }
            },
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "s3.sink.path" : "s3a://ka-app-code-<username>/data"
               }
            }
         ]
      }
   }
}
```

2. Jalankan tindakan `UpdateApplication` dengan permintaan sebelumnya untuk memperbarui properti lingkungan:

```
aws kinesisanalyticsv2 update-application --cli-input-json file://
update_properties_request.json
```

Perbarui kode aplikasi

Ketika Anda perlu memperbarui kode aplikasi Anda dengan versi baru dari paket kode Anda, Anda menggunakan tindakan UpdateApplicationCLI.

> **Note**
>
> Untuk memuat versi baru kode aplikasi dengan nama file yang sama, Anda harus menentukan versi objek baru. Untuk informasi selengkapnya tentang menggunakan versi objek Amazon S3, lihat Mengaktifkan dan Menonaktifkan Versioning.

Untuk menggunakan AWS CLI, hapus paket kode Anda sebelumnya dari bucket Amazon S3 Anda, unggah versi baru, dan panggil `UpdateApplication`, yang menentukan bucket Amazon S3 dan nama objek yang sama, serta versi objek baru. Aplikasi akan memulai ulang dengan paket kode baru.

Permintaan sampel berikut untuk tindakan `UpdateApplication` memuat ulang kode aplikasi dan memulai ulang aplikasi. Perbarui `CurrentApplicationVersionId` ke versi aplikasi saat ini. Anda dapat memeriksa versi aplikasi saat ini menggunakan tindakan `ListApplications` atau `DescribeApplication`. Perbarui akhiran nama bucket (<username>) dengan akhiran yang Anda pilih di bagian. Buat Sumber Daya Dependen

```
{
    "ApplicationName": "s3_sink",
    "CurrentApplicationVersionId": 1,
    "ApplicationConfigurationUpdate": {
        "ApplicationCodeConfigurationUpdate": {
            "CodeContentUpdate": {
                "S3ContentLocationUpdate": {
                    "BucketARNUpdate": "arn:aws:s3:::ka-app-code-username",
                    "FileKeyUpdate": "s3-sink-scala-1.0.jar",
                    "ObjectVersionUpdate": "SAMPLEUehYngP87ex1nzYIGYgfhypvDU"
                }
            }
        }
    }
}
```

Pembersihan Sumber Daya AWS

Bagian ini mencakup prosedur untuk membersihkan sumber daya AWS yang dibuat dalam tutorial Jendela Tumbling.

Topik ini berisi bagian-bagian berikut:

- Hapus Layanan Terkelola Anda untuk Aplikasi Apache Flink

- Hapus Kinesis Data Streams Anda
- Hapus Objek dan Bucket Amazon S3 Anda
- Hapus Sumber Daya IAM Anda
- Hapus CloudWatch Sumber Daya Anda

Hapus Layanan Terkelola Anda untuk Aplikasi Apache Flink

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/flink
2. di panel Managed Service for Apache Flink, pilih. MyApplication
3. Di halaman aplikasi, pilih Delete (Hapus), lalu konfirmasikan penghapusan.

Hapus Kinesis Data Streams Anda

1. Buka konsol Kinesis di https://console.aws.amazon.com/kinesis.
2. Di panel Kinesis Data Streams, pilih. ExampleInputStream
3. Di ExampleInputStreamhalaman, pilih Hapus Stream Kinesis dan kemudian konfirmasikan penghapusan.
4. Di halaman Kinesis streams, pilih, pilih Tindakan ExampleOutputStream, pilih Hapus, lalu konfirmasikan penghapusan.

Hapus Objek dan Bucket Amazon S3 Anda

1. Buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.
2. Pilih ka-app-code- ember. <username>
3. Pilih Delete (Hapus), lalu masukkan nama bucket untuk mengonfirmasi penghapusan.

Hapus Sumber Daya IAM Anda

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di bilah navigasi, pilih Policies (Kebijakan).
3. Di kontrol filter, masukkan kinesis.
4. Pilih kebijakan kinesis-analytics-service- MyApplication -us-west-2.
5. Pilih Policy Actions (Tindakan Kebijakan), lalu pilih Delete (Hapus).

6. Di bilah navigasi, pilih Roles (Peran).
7. Pilih peran kinesis-analytics- -us-west-2. MyApplication
8. Pilih Delete role (Hapus peran), lalu konfirmasi penghapusan.

Hapus CloudWatch Sumber Daya Anda

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di bilah navigasi, pilih Logs.
3. Pilih grup log MyApplication/aws/kinesis-analytics/.
4. Pilih Delete Log Group (Hapus Grup Log), lalu konfirmasi penghapusan.

Keamanan di Amazon Managed Service untuk Apache Flink

Keamanan cloud di AWS adalah prioritas tertinggi. Sebagai pelanggan AWS, Anda akan mendapatkan manfaat dari pusat data dan arsitektur jaringan yang dibangun untuk memenuhi persyaratan organisasi yang paling sensitif terhadap keamanan.

Keamanan adalah tanggung jawab bersama antara AWS dan Anda. Model tanggung jawab bersama menjelaskan hal ini sebagai keamanan dari cloud dan keamanan di cloud:

- Keamanan cloud – AWS berfungsi melindungi infrastruktur yang menjalankan layanan AWS Cloud AWS. AWS juga menyediakan layanan yang dapat Anda gunakan dengan aman. Efektivitas keamanan kami diuji dan diverifikasi secara rutin oleh auditor pihak ketiga sebagai bagian dari program kepatuhan AWS. Untuk mempelajari tentang program kepatuhan yang berlaku untuk Layanan Terkelola untuk Apache Flink, lihat AWSLayanan dalam Lingkup berdasarkan Program Kepatuhan.
- Keamanan di cloud – Tanggung jawab Anda ditentukan oleh layanan AWS yang Anda gunakan. Anda juga bertanggung jawab atas faktor-faktor lain termasuk sensitivitas data Anda, persyaratan organisasi Anda, serta undang-undang dan peraturan yang berlaku.

Dokumentasi ini membantu Anda memahami cara menerapkan model tanggung jawab bersama saat menggunakan Layanan Terkelola untuk Apache Flink. Topik berikut menunjukkan cara mengonfigurasi Layanan Terkelola untuk Apache Flink untuk memenuhi tujuan keamanan dan kepatuhan Anda. Anda juga akan mempelajari cara menggunakan layanan Amazon lain yang dapat membantu Anda memantau dan mengamankan Layanan Terkelola untuk sumber daya Apache Flink.

Topik

Perlindungan Data di Amazon Managed Service untuk Apache Flink

Anda dapat melindungi data Anda menggunakan alat yang disediakan oleh AWS. Managed Service for Apache Flink dapat bekerja dengan layanan yang mendukung enkripsi data, termasuk Kinesis Data Firehose, dan Amazon S3.

Enkripsi Data dalam Layanan Terkelola untuk Apache Flink

Enkripsi at Rest

Perhatikan hal berikut tentang mengenkripsi data saat istirahat dengan Managed Service for Apache Flink:

- Anda dapat mengenkripsi data pada aliran data Kinesis yang masuk menggunakan. StartStreamEncryption Untuk informasi selengkapnya, lihat Apa Itu Enkripsi Sisi Server untuk Kinesis Data Streams?.
- Data output dapat dienkripsi saat istirahat menggunakan Kinesis Data Firehose untuk menyimpan data dalam bucket Amazon S3 terenkripsi. Anda dapat menentukan kunci enkripsi yang digunakan bucket Amazon S3 Anda. Untuk informasi selengkapnya, lihat Melindungi Data Menggunakan Enkripsi Sisi Server dengan Kunci yang Dikelola KMS (SSE-KMS).
- Layanan Terkelola untuk Apache Flink dapat membaca dari sumber streaming apa pun, dan menulis ke tujuan streaming atau database apa pun. Pastikan sumber dan tujuan Anda mengenkripsi semua data dalam transit dan data at rest.
- Kode aplikasi Anda dienkripsi saat istirahat.
- Penyimpanan aplikasi yang tahan lama dienkripsi saat istirahat.
- Menjalankan penyimpanan aplikasi dienkripsi saat istirahat.

Enkripsi dalam Transit

Layanan Terkelola untuk Apache Flink mengenkripsi semua data dalam perjalanan. Enkripsi dalam perjalanan diaktifkan untuk semua Layanan Terkelola untuk aplikasi Apache Flink dan tidak dapat dinonaktifkan.

Layanan Terkelola untuk Apache Flink mengenkripsi data dalam perjalanan dalam skenario berikut:

- Data dalam perjalanan dari Kinesis Data Streams ke Managed Service untuk Apache Flink.
- Data dalam transit antara komponen internal dalam Layanan Terkelola untuk Apache Flink.
- Data dalam perjalanan antara Managed Service untuk Apache Flink dan Kinesis Data Firehose.

Manajemen kunci

Enkripsi data dalam Layanan Terkelola untuk Apache Flink menggunakan kunci yang dikelola layanan. Kunci yang dikelola pelanggan tidak didukung.

Identity and Access Management untuk Amazon Managed Service untuk Apache Flink

AWS Identity and Access Management (IAM) adalah Layanan AWS yang membantu administrator mengontrol akses ke sumber daya AWS secara aman. Administrator IAM mengontrol siapa yang dapat diautentikasi (masuk) dan diberi wewenang (memiliki izin) untuk menggunakan Layanan Terkelola untuk sumber daya Apache Flink. IAM adalah layanan Layanan AWS yang dapat Anda gunakan tanpa dikenakan biaya tambahan.

Topik

Audiens

Cara Anda menggunakan AWS Identity and Access Management (IAM) berbeda, tergantung pada pekerjaan yang Anda lakukan di Managed Service untuk Apache Flink.

Pengguna layanan - Jika Anda menggunakan Layanan Terkelola untuk layanan Apache Flink untuk melakukan pekerjaan Anda, maka administrator Anda memberi Anda kredensi dan izin yang Anda butuhkan. Saat Anda menggunakan lebih banyak fitur Layanan Terkelola untuk Apache Flink untuk melakukan pekerjaan Anda, Anda mungkin memerlukan izin tambahan. Memahami cara akses dikelola dapat membantu Anda meminta izin yang tepat dari administrator Anda. Jika Anda tidak dapat mengakses fitur di Layanan Terkelola untuk Apache Flink, lihat. Memecahkan masalah Amazon Managed Service untuk identitas dan akses Apache Flink

Administrator layanan - Jika Anda bertanggung jawab atas Layanan Terkelola untuk sumber daya Apache Flink di perusahaan Anda, Anda mungkin memiliki akses penuh ke Layanan Terkelola untuk Apache Flink. Tugas Anda adalah menentukan fitur dan sumber daya Layanan Terkelola untuk Apache Flink mana yang harus diakses pengguna layanan Anda. Kemudian, Anda harus mengirimkan permintaan kepada administrator IAM Anda untuk mengubah izin pengguna layanan Anda. Tinjau informasi di halaman ini untuk memahami konsep Basic IAM. Untuk mempelajari lebih lanjut tentang bagaimana perusahaan Anda dapat menggunakan IAM dengan Managed Service for Apache Flink, lihat. Bagaimana Amazon Managed Service untuk Apache Flink bekerja dengan IAM

Administrator IAM - Jika Anda seorang administrator IAM, Anda mungkin ingin mempelajari detail tentang cara menulis kebijakan untuk mengelola akses ke Layanan Terkelola untuk Apache Flink. Untuk melihat contoh Layanan Terkelola untuk kebijakan berbasis identitas Apache Flink yang dapat Anda gunakan di IAM, lihat. Contoh kebijakan berbasis identitas untuk Amazon Managed Service untuk Apache Flink

Mengautentikasi dengan identitas

Autentikasi adalah cara Anda untuk masuk ke AWS menggunakan kredensial identitas Anda. Anda harus terautentikasi (masuk ke AWS) sebagai Pengguna root akun AWS, sebagai pengguna IAM, atau dengan mengambil peran IAM.

Anda dapat masuk ke AWS sebagai identitas terfederasi dengan menggunakan kredensial yang disediakan melalui sumber identitas. Pengguna AWS IAM Identity Center Pengguna (Pusat Identitas IAM), autentikasi Single Sign-On perusahaan Anda, dan kredensial Google atau Facebook Anda merupakan contoh identitas terfederasi. Saat Anda masuk sebagai identitas gabungan, administrator Anda sebelumnya menyiapkan federasi identitas menggunakan peran IAM. Ketika Anda mengakses AWS dengan menggunakan federasi, Anda secara tidak langsung mengambil suatu peran.

Bergantung pada jenis pengguna Anda, Anda dapat masuk ke AWS Management Console atau portal akses AWS. Untuk informasi selengkapnya tentang cara masuk ke AWS, lihat Cara masuk ke Akun AWS dalam Panduan Pengguna AWS Sign-In.

Jika Anda mengakses AWS secara terprogram, AWS memberikan Kit Pengembangan Perangkat Lunak (SDK) dan antarmuka baris perintah (CLI) untuk menandatangani permintaan Anda secara kriptografis dengan menggunakan kredensial Anda. Jika Anda tidak menggunakan peralatan AWS, Anda harus menandatangani permintaan sendiri. Untuk informasi selengkapnya tentang cara menggunakan metode yang disarankan untuk menandatangani permintaan sendiri, lihat Menandatangani permintaan API AWS dalam Panduan Pengguna IAM.

Apa pun metode autentikasi yang digunakan, Anda mungkin diminta untuk menyediakan informasi keamanan tambahan. Sebagai contoh, AWS menyarankan Anda menggunakan autentikasi multi-faktor (MFA) untuk meningkatkan keamanan akun Anda. Untuk mempelajari lebih lanjut, lihat Autentikasi multi-faktor dalam Panduan Pengguna AWS IAM Identity Center dan Menggunakan autentikasi multi-faktor (MFA) di AWS dalam Panduan Pengguna IAM.

Pengguna root Akun AWS

Ketika membuat Akun AWS, Anda memulai dengan satu identitas masuk yang memiliki akses penuh ke semua Layanan AWS dan sumber daya di akun tersebut. Identitas ini disebut pengguna root Akun AWS dan diakses dengan cara masuk menggunakan alamat email dan kata sandi yang Anda gunakan untuk membuat akun. Kami sangat menyarankan agar Anda tidak menggunakan pengguna root untuk tugas sehari-hari Anda. Lindungi kredensial pengguna root Anda dan gunakan kredensial tersebut untuk melakukan tugas yang hanya dapat dilakukan pengguna root. Untuk daftar tugas lengkap yang mengharuskan Anda masuk sebagai pengguna root, lihat Tugas yang memerlukan kredensial pengguna root dalam Panduan Pengguna IAM.

Identitas terfederasi

Praktik terbaiknya adalah mewajibkan pengguna manusia, termasuk pengguna yang memerlukan akses administrator, untuk menggunakan federasi dengan penyedia identitas untuk mengakses Layanan AWS dengan menggunakan kredensial temporer.

Identitas terfederasi adalah pengguna dari direktori pengguna perusahaan Anda, penyedia identitas web, AWS Directory Service, direktori Pusat Identitas, atau pengguna mana pun yang mengakses Layanan AWS dengan menggunakan kredensial yang disediakan melalui sumber identitas. Ketika identitas terfederasi mengakses Akun AWS, identitas tersebut mengambil peran, dan peran ini memberikan kredensial sementara.

Untuk pengelolaan akses terpusat, sebaiknya Anda menggunakan AWS IAM Identity Center. Anda dapat membuat pengguna dan grup di Pusat Identitas IAM, atau Anda dapat menghubungkan dan menyinkronkan ke sekumpulan pengguna dan grup di sumber identitas Anda sendiri untuk digunakan

di semua Akun AWS dan aplikasi Anda. Untuk informasi tentang Pusat Identitas IAM, lihat Apa yang dimaksud Pusat Identitas IAM? dalam Panduan Pengguna AWS IAM Identity Center.

Pengguna dan grup IAM

Pengguna IAM adalah identitas dalam Akun AWS Anda yang memiliki izin khusus untuk satu orang atau aplikasi. Jika memungkinkan, sebaiknya andalkan kredensial temporer, dan bukan membuat pengguna IAM yang memiliki kredensial jangka panjang seperti kata sandi dan kunci akses. Namun, jika Anda memiliki kasus penggunaan khusus yang memerlukan kredensial jangka panjang dengan pengguna IAM, sebaiknya rotasikan kunci akses. Untuk informasi selengkapnya, lihat Merotasi kunci akses secara teratur untuk kasus penggunaan yang memerlukan kredensial jangka panjang dalam Panduan Pengguna IAM.

Grup IAM adalah identitas yang menentukan kumpulan pengguna IAM. Anda tidak dapat masuk sebagai grup. Anda dapat menggunakan grup untuk menentukan izin untuk beberapa pengguna sekaligus. Grup membuat izin lebih mudah dikelola untuk sekelompok besar pengguna. Misalnya, Anda dapat memiliki grup yang bernama IAMAdmins dan memberikan izin kepada grup tersebut untuk mengelola sumber daya IAM.

Pengguna berbeda dari peran. Pengguna secara unik terkait dengan satu orang atau aplikasi, tetapi peran tersebut dimaksudkan untuk dapat diambil oleh siapa pun yang membutuhkannya. Pengguna memiliki kredensial jangka panjang permanen, tetapi peran memberikan kredensial sementara. Untuk mempelajari selengkapnya, silakan lihat Kapan harus membuat pengguna IAM (bukan peran) dalam Panduan Pengguna IAM.

Peran IAM

Peran IAM merupakan identitas dalam Akun AWS Anda yang memiliki izin khusus. Peran ini mirip dengan pengguna IAM, tetapi tidak terkait dengan orang tertentu. Anda dapat mengambil peran IAM untuk sementara dalam AWS Management Console dengan berganti peran. Anda dapat mengambil peran dengan cara memanggil operasi API AWS CLI atau AWS atau menggunakan URL kustom. Untuk informasi selengkapnya tentang metode untuk menggunakan peran, lihat Menggunakan peran IAM dalam Panduan Pengguna IAM.

Peran IAM dengan kredensial sementara berguna dalam situasi berikut:

- Akses pengguna gabungan – Untuk menetapkan izin ke sebuah identitas gabungan, Anda dapat membuat peran dan menentukan izin untuk peran tersebut. Saat identitas terfederasi diautentikasi, identitas tersebut dikaitkan dengan peran dan diberikan izin yang ditentukan oleh peran. Untuk informasi tentang peran untuk federasi, lihat Membuat peran untuk Penyedia Identitas pihak

ketiga dalam Panduan Pengguna IAM. Jika Anda menggunakan Pusat Identitas IAM, Anda mengonfigurasi sekumpulan izin. Untuk mengontrol apa yang dapat diakses identitas Anda setelah identitas tersebut diautentikasi, Pusat Identitas IAM mengaitkan izin yang ditetapkan ke peran dalam IAM. Untuk informasi tentang rangkaian izin, lihat Rangkaian izin dalam Panduan Pengguna AWS IAM Identity Center.

- Izin pengguna IAM sementara – Pengguna atau peran IAM dapat mengambil peran IAM guna mendapatkan berbagai izin secara sementara untuk tugas tertentu.
- Akses lintas akun – Anda dapat menggunakan peran IAM untuk mengizinkan seseorang (pengguna utama tepercaya) dengan akun berbeda untuk mengakses sumber daya yang ada di akun Anda. Peran adalah cara utama untuk memberikan akses lintas akun. Namun, pada beberapa Layanan AWS, Anda dapat menyertakan kebijakan secara langsung ke sumber daya (bukan menggunakan peran sebagai proksi). Untuk mempelajari perbedaan antara kebijakan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Bagaimana peran IAM berbeda dari kebijakan berbasis sumber daya dalam Panduan Pengguna IAM.
- Akses lintas layanan – Sebagian Layanan AWS menggunakan fitur di Layanan AWS lainnya. Contoh, ketika Anda melakukan panggilan dalam layanan, umumnya layanan tersebut menjalankan aplikasi di Amazon EC2 atau menyimpan objek di Amazon S3. Suatu layanan mungkin melakukan hal tersebut menggunakan izin pengguna utama panggilan, menggunakan peran layanan, atau peran terkait layanan.
  - Sesi akses maju (FAS) – Ketika Anda menggunakan pengguna IAM atau peran IAM untuk melakukan tindakan di AWS, Anda akan dianggap sebagai seorang pengguna utama. Saat menggunakan beberapa layanan, Anda mungkin melakukan tindakan yang kemudian dilanjutkan oleh tindakan lain pada layanan yang berbeda. FAS menggunakan izin dari pengguna utama untuk memanggil Layanan AWS, yang dikombinasikan dengan Layanan AWS yang diminta untuk membuat permintaan ke layanan hilir. Permintaan FAS hanya diajukan saat layanan menerima permintaan yang memerlukan interaksi dengan Layanan AWS lain atau sumber daya lain untuk diselesaikan. Dalam hal ini, Anda harus memiliki izin untuk melakukan kedua tindakan tersebut. Untuk detail kebijakan ketika mengajukan permintaan FAS, lihat Meneruskan sesi akses.
  - Peran IAM – Peran layanan adalah peran IAM yang diambil layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, memodifikasi, dan menghapus peran layanan dari dalam IAM. Untuk informasi selengkapnya, lihat Membuat peran untuk mendelegasikan izin ke Layanan AWS dalam Panduan pengguna IAM.
  - Peran terkait layanan – Peran terkait layanan adalah tipe peran layanan yang terkait dengan Layanan AWS. Layanan tersebut dapat mengambil peran untuk melakukan sebuah tindakan

atas nama Anda. Peran terkait layanan akan muncul di Akun AWS Anda dan dimiliki oleh layanan tersebut. Administrator IAM dapat melihat, tetapi tidak dapat mengedit izin untuk peran terkait layanan.

- Aplikasi yang berjalan di Amazon EC2 – Anda dapat menggunakan peran IAM untuk mengelola kredensial sementara untuk aplikasi yang berjalan di instans EC2 dan mengajukan permintaan API AWS CLI atau AWS. Cara ini lebih dianjurkan daripada menyimpan kunci akses dalam instans EC2. Untuk menetapkan peran AWS ke instans EC2 dan menyediakannya bagi semua aplikasinya, Anda dapat membuat profil instans yang dilampirkan ke instans tersebut. Profil instans berisi peran dan memungkinkan program yang berjalan di instans EC2 mendapatkan kredensial sementara. Untuk informasi selengkapnya, lihat Menggunakan peran IAM untuk memberikan izin ke aplikasi yang berjalan di instans Amazon EC2 dalam Panduan Pengguna IAM.

Untuk mempelajari apakah kita harus menggunakan peran IAM atau pengguna IAM, lihat Kapan harus membuat peran IAM (bukan pengguna) dalam Panduan Pengguna IAM.

Mengelola akses menggunakan kebijakan

Anda mengendalikan akses di AWS dengan membuat kebijakan dan melampirkannya ke identitas atau sumber daya AWS. Kebijakan adalah objek di AWS yang, ketika terkait dengan identitas atau sumber daya, akan menentukan izinnya. AWS mengevaluasi kebijakan-kebijakan tersebut ketika seorang pengguna utama (pengguna, pengguna root, atau sesi peran) mengajukan permintaan. Izin dalam kebijakan menentukan apakah permintaan diizinkan atau ditolak. Sebagian besar kebijakan disimpan di AWS sebagai dokumen JSON. Untuk informasi selengkapnya tentang struktur dan isi dokumen kebijakan JSON, silakan lihat Gambaran Umum kebijakan JSON dalam Panduan Pengguna IAM.

Administrator dapat menggunakan kebijakan JSON AWS untuk menentukan secara spesifik siapa yang memiliki akses terhadap apa. Artinya, pengguna utama manakah yang dapat melakukan tindakan pada sumber daya apa, dan dalam kondisi apa.

Secara default, pengguna dan peran tidak memiliki izin. Untuk memberikan izin kepada pengguna untuk melakukan tindakan pada sumber daya yang mereka perlukan, administrator IAM dapat membuat kebijakan IAM. Administrator kemudian dapat menambahkan kebijakan IAM ke peran, dan pengguna dapat menjalankan peran.

Kebijakan IAM mendefinisikan izin untuk suatu tindakan terlepas dari metode yang Anda gunakan untuk operasi. Sebagai contoh, anggap saja Anda memiliki kebijakan yang mengizinkan tindakan

`iam:GetRole`. Pengguna dengan kebijakan tersebut dapat memperoleh informasi peran dari AWS Management Console, AWS CLI, atau API AWS.

Kebijakan berbasis identitas

Kebijakan berbasis identitas adalah dokumen kebijakan izin JSON yang dapat Anda lampirkan ke sebuah identitas, seperti pengguna IAM, grup pengguna IAM, atau peran IAM. Kebijakan ini mengontrol jenis tindakan yang dapat dilakukan pengguna dan peran, di sumber daya mana, dan dengan ketentuan apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Membuat kebijakan IAM dalam Panduan Pengguna IAM.

Kebijakan berbasis identitas dapat dikategorikan lebih lanjut sebagai kebijakan inline atau kebijakan terkelola. Kebijakan inline disematkan langsung ke satu pengguna, grup, atau peran. Kebijakan terkelola adalah kebijakan mandiri yang dapat Anda lampirkan ke beberapa pengguna, grup, dan peran di Akun AWS Anda. Kebijakan terkelola meliputi kebijakan yang dikelola AWS dan kebijakan yang dikelola pelanggan. Untuk mempelajari cara memilih antara kebijakan terkelola atau kebijakan inline, lihat Memilih antara kebijakan terkelola dan kebijakan inline dalam Panduan Pengguna IAM.

Kebijakan berbasis sumber daya

Kebijakan berbasis sumber daya adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contoh kebijakan berbasis sumber daya adalah kebijakan kepercayaan peran IAM dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya, administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Untuk sumber daya yang dilampiri kebijakan tersebut, kebijakan ini menentukan jenis tindakan yang dapat dilakukan oleh pengguna utama tertentu di sumber daya tersebut dan apa ketentuannya. Anda harus menentukan pengguna utama dalam kebijakan berbasis sumber daya. Pengguna utama dapat mencakup akun, pengguna, peran, pengguna gabungan, atau Layanan AWS.

Kebijakan berbasis sumber daya merupakan kebijakan inline yang terletak di layanan tersebut. Anda tidak dapat menggunakan kebijakan yang dikelola AWS dari IAM dalam kebijakan berbasis sumber daya.

Daftar kontrol akses (ACL)

Daftar kontrol akses (ACL) mengendalikan pengguna utama mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACL serupa dengan kebijakan berbasis sumber daya, meskipun tidak menggunakan format dokumen kebijakan JSON.

Amazon S3, AWS WAF, dan Amazon VPC adalah contoh layanan yang mendukung ACL. Untuk mempelajari ACL selengkapnya, silakan lihat Gambaran umum daftar kontrol akses (ACL) di Panduan Developer Layanan Penyimpanan Ringkas Amazon.

Tipe kebijakan lain

AWS mendukung jenis kebijakan tambahan yang kurang umum. Tipe-tipe kebijakan ini dapat mengatur izin maksimum yang diberikan kepada Anda berdasarkan tipe kebijakan yang lebih umum.

- Batasan izin – Batasan izin adalah fitur lanjutan di mana Anda menetapkan izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas kepada entitas IAM (pengguna atau peran IAM). Anda dapat menetapkan batasan izin untuk suatu entitas. Izin yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas milik entitas dan batasan izinnya. Kebijakan berbasis sumber daya yang menentukan pengguna atau peran dalam bidang `Principal` tidak dibatasi oleh batasan izin. Penolakan secara eksplisit terhadap salah satu kebijakan ini akan mengesampingkan izin tersebut. Untuk informasi selengkapnya tentang batasan izin, lihat Batasan izin untuk entitas IAM dalam Panduan Pengguna IAM.
- Kebijakan kontrol layanan (SCP) – SCP adalah kebijakan JSON yang menentukan izin maksimum untuk sebuah organisasi atau unit organisasi (OU) di AWS Organizations. AWS Organizations adalah layanan untuk mengelompokkan dan mengelola beberapa akun Akun AWS yang dimiliki bisnis Anda secara terpusat. Jika Anda mengaktifkan semua fitur di organisasi, Anda dapat menerapkan kebijakan kontrol layanan (SCP) ke salah satu atau semua akun Anda. SCP membatasi izin untuk entitas dalam akun anggota, termasuk setiap Pengguna root akun AWS. Untuk informasi selengkapnya tentang Organisasi dan SCP, lihat Cara kerja SCP dalam Panduan Pengguna AWS Organizations.
- Kebijakan sesi – Kebijakan sesi adalah kebijakan lanjutan yang Anda teruskan sebagai parameter saat Anda membuat sesi sementara secara terprogram untuk peran atau pengguna gabungan. Izin sesi yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas pengguna atau peran dan kebijakan sesi. Izin juga bisa datang dari kebijakan berbasis sumber daya. Penolakan eksplisit di salah satu kebijakan ini akan membatalkan izin tersebut. Untuk informasi selengkapnya, lihat Kebijakan sesi dalam Panduan Pengguna IAM.

Berbagai jenis kebijakan

Jika beberapa jenis kebijakan diberlakukan untuk satu permintaan, izin yang dihasilkan lebih rumit untuk dipahami. Untuk mempelajari cara AWS menentukan apakah akan mengizinkan permintaan ketika ada beberapa jenis kebijakan, lihat Logika evaluasi kebijakan dalam Panduan Pengguna IAM.

Bagaimana Amazon Managed Service untuk Apache Flink bekerja dengan IAM

Sebelum Anda menggunakan IAM untuk mengelola akses ke Managed Service for Apache Flink, pelajari fitur IAM yang tersedia untuk digunakan dengan Managed Service for Apache Flink.

Fitur IAM yang dapat Anda gunakan dengan Amazon Managed Service untuk Apache Flink

Fitur IAM	Layanan Terkelola untuk dukungan Apache Flink
Kebijakan berbasis identitas	Ya
Kebijakan berbasis sumber daya	Tidak
Tindakan kebijakan	Ya
Sumber daya kebijakan	Ya
Kunci persyaratan kebijakan	Tidak
ACL	Tidak
ABAC (tanda dalam kebijakan)	Ya
Kredensial sementara	Ya
Izin pengguna utama	Ya
Peran layanan	Tidak
Peran terkait layanan	Tidak

Untuk mendapatkan tampilan tingkat tinggi tentang bagaimana Layanan Terkelola untuk Apache Flink dan AWS layanan lainnya bekerja dengan sebagian besar fitur IAM, lihat AWSlayanan yang bekerja dengan IAM di Panduan Pengguna IAM.

Kebijakan berbasis identitas untuk Layanan Terkelola untuk Apache Flink

Mendukung kebijakan berbasis identitas	Ya

Kebijakan berbasis identitas adalah dokumen kebijakan izin JSON yang dapat Anda lampirkan ke sebuah identitas, seperti pengguna IAM, grup pengguna IAM, atau peran IAM. Kebijakan ini mengontrol jenis tindakan yang dapat dilakukan pengguna dan peran, di sumber daya mana, dan dengan ketentuan apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Membuat kebijakan IAM dalam Panduan Pengguna IAM.

Dengan kebijakan berbasis identitas IAM, Anda dapat menentukan tindakan dan sumber daya yang diizinkan atau ditolak, serta ketentuan terkait jenis tindakan yang diizinkan atau ditolak. Anda tidak dapat menentukan pengguna utama dalam kebijakan berbasis identitas karena kebijakan ini berlaku untuk pengguna atau peran yang dilampiri kebijakan. Untuk mempelajari semua elemen yang dapat digunakan dalam kebijakan JSON, lihat Referensi elemen kebijakan JSON IAM dalam Panduan Pengguna IAM.

Contoh kebijakan berbasis identitas untuk Managed Service untuk Apache Flink

Untuk melihat contoh Layanan Terkelola untuk kebijakan berbasis identitas Apache Flink, lihat. Contoh kebijakan berbasis identitas untuk Amazon Managed Service untuk Apache Flink

Kebijakan berbasis sumber daya dalam Layanan Terkelola untuk Apache Flink

Mendukung kebijakan berbasis sumber daya	Ya

Kebijakan berbasis sumber daya adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contoh kebijakan berbasis sumber daya adalah kebijakan kepercayaan peran IAM dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya, administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Untuk sumber daya yang dilampiri kebijakan tersebut, kebijakan ini menentukan jenis tindakan yang dapat dilakukan oleh pengguna utama tertentu di sumber daya tersebut dan apa ketentuannya. Anda harus menentukan pengguna utama dalam kebijakan berbasis sumber daya. Pengguna utama dapat mencakup akun, pengguna, peran, pengguna gabungan, atau Layanan AWS.

Untuk mengaktifkan akses lintas akun, Anda dapat menentukan seluruh akun atau entitas IAM di akun lain sebagai pengguna utama dalam kebijakan berbasis sumber daya. Menambahkan pengguna utama lintas akun ke kebijakan berbasis sumber daya bagian dari membangun hubungan kepercayaan. Ketika pengguna utama dan sumber daya berada di Akun AWS yang berbeda, administrator IAM di akun tepercaya juga harus memberikan izin kepada entitas pengguna utama (pengguna atau peran) untuk mengakses sumber daya. Izin diberikan dengan melampirkan kebijakan berbasis identitas ke entitas tersebut. Namun, jika kebijakan berbasis sumber daya memberikan akses kepada pengguna utama dalam akun yang sama, kebijakan berbasis identitas lainnya tidak diperlukan. Untuk informasi selengkapnya, lihat Perbedaan peran IAM dengan kebijakan berbasis sumber daya di Panduan Pengguna IAM.

Tindakan kebijakan untuk Layanan Terkelola untuk Apache Flink

Mendukung tindakan kebijakan	Ya

Administrator dapat menggunakan kebijakan JSON AWS untuk menentukan siapa yang memiliki akses ke apa. Yaitu, pengguna utama mana yang dapat melakukan tindakan pada sumber daya apa, dan dalam kondisi apa.

Elemen `Action` dari kebijakan JSON menjelaskan tindakan yang dapat Anda gunakan untuk mengizinkan atau menolak akses dalam sebuah kebijakan. Tindakan kebijakan biasanya memiliki nama yang sama seperti operasi API AWS terkait. Ada beberapa pengecualian, misalnya tindakan hanya izin yang tidak memiliki operasi API yang cocok. Ada juga beberapa operasi yang memerlukan beberapa tindakan dalam suatu kebijakan. Tindakan tambahan ini disebut tindakan dependen.

Menyertakan tindakan dalam suatu kebijakan untuk memberikan izin melakukan operasi terkait.

Untuk melihat daftar Layanan Terkelola untuk tindakan Apache Flink, lihat Tindakan yang Ditetapkan oleh Amazon Managed Service untuk Apache Flink di Referensi Otorisasi Layanan.

Tindakan kebijakan di Layanan Terkelola untuk Apache Flink menggunakan awalan berikut sebelum tindakan:

```
Kinesis Analytics
```

Untuk menetapkan secara spesifik beberapa tindakan dalam satu pernyataan, pisahkan tindakan-tindakan tersebut dengan koma.

```
"Action": [
      "Kinesis Analytics:action1",
      "Kinesis Analytics:action2"
         ]
```

Anda juga dapat menentukan beberapa tindakan menggunakan wildcard (*). Sebagai contoh, untuk menentukan semua tindakan yang dimulai dengan kata `Describe`, sertakan tindakan berikut:

```
"Action": "Kinesis Analytics:Describe*"
```

Untuk melihat contoh Layanan Terkelola untuk kebijakan berbasis identitas Apache Flink, lihat. Contoh kebijakan berbasis identitas untuk Amazon Managed Service untuk Apache Flink

Sumber daya kebijakan untuk Layanan Terkelola untuk Apache Flink

Mendukung sumber daya kebijakan	Ya

Administrator dapat menggunakan kebijakan JSON AWS untuk menentukan siapa yang memiliki akses ke apa. Yaitu, pengguna utama mana yang dapat melakukan tindakan pada sumber daya apa, dan dalam kondisi apa.

Elemen kebijakan JSON `Resource` menentukan objek atau beberapa objek yang menjadi target penerapan tindakan. Pernyataan harus menyertakan elemen `Resource` atau `NotResource`. Praktik terbaiknya, tentukan sumber daya menggunakan Amazon Resource Name (ARN). Anda dapat melakukan ini untuk tindakan yang mendukung jenis sumber daya tertentu, yang dikenal sebagai izin tingkat sumber daya.

Untuk tindakan yang tidak mendukung izin di tingkat sumber daya, misalnya operasi pencantuman, gunakan wildcard (*) untuk mengindikasikan bahwa pernyataan tersebut berlaku untuk semua sumber daya.

```
"Resource": "*"
```

Untuk melihat daftar Layanan Terkelola untuk jenis sumber daya Apache Flink dan ARNnya, lihat Sumber Daya yang Ditetapkan oleh Amazon Managed Service untuk Apache Flink di Referensi

Otorisasi Layanan. Untuk mempelajari tindakan yang dapat Anda tentukan ARN dari setiap sumber daya, lihat Tindakan yang Ditentukan oleh Amazon Managed Service untuk Apache Flink.

Untuk melihat contoh Layanan Terkelola untuk kebijakan berbasis identitas Apache Flink, lihat. Contoh kebijakan berbasis identitas untuk Amazon Managed Service untuk Apache Flink

Kunci kondisi kebijakan untuk Layanan Terkelola untuk Apache Flink

Mendukung kunci kondisi kebijakan spesifik layanan	Ya

Administrator dapat menggunakan kebijakan JSON AWS untuk menentukan siapa yang memiliki akses ke apa. Yaitu, pengguna utama mana yang dapat melakukan tindakan pada sumber daya apa, dan dalam kondisi apa.

Elemen `Condition` (atau blok `Condition`) memungkinkan Anda menentukan kondisi di mana suatu pernyataan akan diterapkan. Elemen `Condition` bersifat opsional. Anda dapat membuat ekspresi kondisional yang menggunakan operator kondisi, misalnya sama dengan atau kurang dari, untuk mencocokkan kondisi dalam kebijakan dengan nilai-nilai yang diminta.

Jika Anda menentukan beberapa elemen `Condition` dalam satu pernyataan, atau beberapa kunci dalam satu elemen `Condition`, AWS akan mengevaluasinya dengan menggunakan operasi `AND` logis. Jika Anda menentukan beberapa niilai untuk satu kunci persyaratan, AWS akan mengevaluasi syarat tersebut menggunakan operasi `OR` yang logis. Semua kondisi harus dipenuhi sebelum izin pernyataan diberikan.

Anda juga dapat menggunakan variabel placeholder saat menentukan kondisi. Sebagai contoh, Anda dapat memberikan izin kepada pengguna IAM untuk mengakses sumber daya hanya jika izin tersebut mempunyai tanda yang sesuai dengan nama pengguna IAM mereka. Untuk informasi selengkapnya, silakan lihat Elemen kebijakan IAM: variabel dan tanda di Panduan Pengguna IAM.

AWS mendukung kunci kondisi global dan kunci kondisi spesifik layanan. Untuk melihat semua kunci kondisi global AWS, lihat kunci konteks kondisi global AWS dalam Panduan Pengguna IAM.

Untuk melihat daftar kunci kondisi Layanan Terkelola untuk Apache Flink, lihat Kunci Kondisi untuk Amazon Managed Service for Apache Flink di Referensi Otorisasi Layanan. Untuk mempelajari tindakan dan sumber daya yang dapat Anda gunakan kunci kondisi, lihat Tindakan yang Ditentukan oleh Amazon Managed Service untuk Apache Flink.

Untuk melihat contoh Layanan Terkelola untuk kebijakan berbasis identitas Apache Flink, lihat. Contoh kebijakan berbasis identitas untuk Amazon Managed Service untuk Apache Flink

Daftar kontrol akses (ACL) di Layanan Terkelola untuk Apache Flink

Mendukung ACL	Tidak

Daftar kontrol akses (ACL) mengontrol pengguna utama (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACL sama dengan kebijakan berbasis sumber daya, meskipun tidak menggunakan format dokumen kebijakan JSON.

Kontrol akses berbasis atribut (ABAC) dengan Managed Service untuk Apache Flink

Mendukung ABAC (tanda dalam kebijakan)	Ya

Kontrol akses berbasis atribut (ABAC) adalah strategi otorisasi yang mendefinisikan izin berdasarkan atribut. Di AWS, atribut ini disebut tag. Anda dapat melampirkan tanda ke entitas IAM (pengguna atau peran) dan ke banyak sumber daya AWS. Pemberian tanda ke entitas dan sumber daya adalah langkah pertama dari ABAC. Kemudian rancanglah kebijakan ABAC untuk mengizinkan operasi-operasi ketika tanda milik pengguna utama cocok dengan tanda yang ada di sumber daya yang ingin diakses.

ABAC sangat berguna di lingkungan yang berkembang dengan cepat dan berguna dalam situasi di mana pengelolaan kebijakan menjadi rumit.

Untuk mengendalikan akses berdasarkan tag, berikan informasi tentang tanda di elemen syarat dari sebuh kebijakan dengan menggunakan kunci-kunci persyaratan `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`*, atau `aws:TagKeys`.

Jika sebuah layanan mendukung ketiga kunci kondisi untuk setiap jenis sumber daya, nilainya adalah Ya untuk layanan tersebut. Jika suatu layanan mendukung ketiga kunci kondisi hanya untuk beberapa jenis sumber daya, nilainya adalah Parsial.

Untuk informasi selengkapnya tentang ABAC, lihat Apa itu ABAC? di Panduan Pengguna IAM. Untuk melihat tutorial terkait langkah-langkah penyiapan ABAC, lihat Menggunakan kontrol akses berbasis atribut (ABAC) di Panduan Pengguna IAM.

Menggunakan kredensyal Sementara dengan Managed Service untuk Apache Flink

Mendukung kredensial sementara	Ya

Beberapa Layanan AWS tidak berfungsi saat Anda masuk menggunakan kredensial sementara. Sebagai informasi tambahan, termasuk tentang Layanan AWS mana saja yang berfungsi dengan kredensial sementara, lihat Layanan AWS yang berfungsi dengan IAM di Panduan Pengguna IAM.

Anda menggunakan kredensial sementara jika Anda masuk ke AWS Management Console menggunakan metode apa pun kecuali nama pengguna dan kata sandi. Misalnya, ketika Anda mengakses AWS dengan menggunakan tautan masuk tunggal (SSO) milik perusahaan Anda, proses itu secara otomatis akan membuat kredensial temporer. Anda juga akan membuat kredensial sementara secara otomatis saat masuk ke konsol sebagai pengguna dan kemudian beralih peran. Untuk informasi selengkapnya tentang cara beralih peran, lihat Beralih peran (konsol) di Panduan Pengguna IAM.

Anda dapat membuat kredensial sementara secara manual menggunakan AWS CLI atau AWS API. Anda kemudian dapat menggunakan kredensial sementara untuk mengakses AWS. AWS menyarankan Anda membuat kredensial sementara secara dinamis, alih-alih menggunakan kunci akses jangka panjang. Untuk informasi selengkapnya, lihat Kredensial keamanan sementara di IAM.

Izin utama lintas layanan untuk Layanan Terkelola untuk Apache Flink

Mendukung sesi akses maju (FAS)	Ya

Jika menggunakan pengguna IAM atau peran IAM untuk melakukan tindakan di AWS, Anda akan dianggap sebagai pengguna utama. Jika menggunakan beberapa layanan, Anda mungkin melakukan tindakan yang kemudian dilanjutkan oleh tindakan lain di layanan yang berbeda. FAS menggunakan izin dari pengguna utama untuk memanggil Layanan AWS, yang dikombinasikan dengan Layanan AWS yang diminta untuk membuat permintaan ke layanan hilir. Permintaan FAS hanya diajukan saat layanan menerima permintaan yang memerlukan interaksi dengan Layanan AWS lain atau sumber daya lain untuk diselesaikan. Dalam hal ini, Anda harus memiliki izin untuk melakukan kedua tindakan tersebut. Untuk detail kebijakan ketika mengajukan permintaan FAS, lihat Meneruskan sesi akses.

Peran layanan untuk Layanan Terkelola untuk Apache Flink

Mendukung peran layanan	Ya

Peran layanan adalah peran IAM yang diambil oleh layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, mengubah, dan menghapus peran layanan dari dalam IAM. Untuk informasi selengkapnya, lihat Membuat peran untuk mendelegasikan izin ke Layanan AWS dalam Panduan pengguna IAM.

> **Warning**
>
> Mengubah izin untuk peran layanan dapat merusak fungsionalitas Layanan Terkelola untuk Apache Flink. Edit peran layanan hanya jika Layanan Terkelola untuk Apache Flink memberikan panduan untuk melakukannya.

Peran terkait layanan untuk Layanan Terkelola untuk Apache Flink

Mendukung peran yang terkait layanan	Ya

Peran yang terkait layanan adalah jenis peran layanan yang terkait dengan Layanan AWS. Layanan ini dapat menjalankan peran untuk melakukan tindakan atas nama Anda. Peran terkait layanan akan muncul di Akun AWS Anda dan dimiliki oleh layanan tersebut. Administrator IAM dapat melihat, tetapi tidak dapat mengedit izin untuk peran terkait layanan.

Untuk detail tentang pembuatan atau pengelolaan peran terkait layanan, lihat Layanan AWS yang berfungsi dengan IAM. Temukan sebuah layanan dalam tabel yang memiliki `Yes` di kolom Peran terkait layanan. Pilih tautan Ya untuk melihat dokumentasi peran terkait layanan untuk layanan tersebut.

Contoh kebijakan berbasis identitas untuk Amazon Managed Service untuk Apache Flink

Secara default, pengguna dan peran tidak memiliki izin untuk membuat atau memodifikasi Layanan Terkelola untuk sumber daya Apache Flink. Pengguan dan peran tersebut juga tidak dapat melakukan tugas dengan menggunakan AWS Management Console, AWS Command Line

Interface (AWS CLI), atau API AWS. Untuk memberikan izin kepada pengguna untuk melakukan tindakan pada sumber daya yang mereka perlukan, administrator IAM dapat membuat kebijakan IAM. Administrator kemudian dapat menambahkan kebijakan IAM ke peran, dan pengguna dapat menjalankan peran.

Untuk mempelajari cara membuat kebijakan berbasis identitas IAM menggunakan contoh dokumen kebijakan JSON ini, lihat Membuat kebijakan IAM dalam Panduan Pengguna IAM.

Untuk detail tentang tindakan dan jenis sumber daya yang ditentukan oleh Layanan Terkelola untuk Apache Flink, termasuk format ARN untuk setiap jenis sumber daya, lihat Tindakan, Sumber Daya, dan Kunci Kondisi untuk Amazon Managed Service for Apache Flink di Referensi Otorisasi Layanan.

Topik

- Praktik terbaik kebijakan
- Menggunakan Layanan Terkelola untuk konsol Apache Flink
- Izinkan pengguna melihat izin mereka sendiri

Praktik terbaik kebijakan

Kebijakan berbasis identitas menentukan apakah seseorang dapat membuat, mengakses, atau menghapus Layanan Terkelola untuk sumber daya Apache Flink di akun Anda. Tindakan ini dikenai biaya untuk Akun AWS Anda. Ketika Anda membuat atau mengedit kebijakan berbasis identitas, ikuti panduan dan rekomendasi ini:

- Mulai menggunakan kebijakan yang dikelola AWS dan beralih ke izin dengan hak akses paling rendah – Untuk mulai memberikan izin kepada pengguna dan beban kerja Anda, gunakan kebijakan yang dikelola AWS yang memberikan izin untuk banyak kasus penggunaan umum. Kebijakan ini ada di Akun AWS Anda. Sebaiknya Anda mengurangi izin lebih lanjut dengan menentukan kebijakan yang dikelola pelanggan AWS yang khusus untuk kasus penggunaan Anda. Untuk informasi selengkapnya, lihat kebijakan yang dikelola AWS atau kebijakan yang dikelola AWS untuk fungsi pekerjaan di Panduan Pengguna IAM.
- Menerapkan izin dengan hak akses paling rendah – Ketika Anda menetapkan izin dengan kebijakan IAM, hanya berikan izin yang diperlukan untuk melakukan tugas. Anda melakukan ini dengan menentukan tindakan yang dapat diambil pada sumber daya tertentu dalam kondisi tertentu, juga dikenal sebagai izin hak akses paling rendah. Untuk informasi selengkapnya tentang cara menggunakan IAM untuk menerapkan izin, lihat Kebijakan dan izin di IAM di Panduan Pengguna IAM.

- Gunakan kondisi dalam kebijakan IAM untuk membatasi akses lebih lanjut – Anda dapat menambahkan kondisi ke kebijakan Anda untuk membatasi akses ke tindakan dan sumber daya. Misalnya, Anda dapat menulis syarat kebijakan untuk menentukan bahwa semua pengajuan harus dikirim menggunakan SSL. Anda juga dapat menggunakan kondisi untuk memberi akses ke tindakan layanan jika digunakan melalui Layanan AWS yang spesifk, seperti AWS CloudFormation. Untuk informasi selengkapnya, lihat Elemen kebijakan JSON IAM: Syarat di Panduan Pengguna IAM.
- Menggunakan IAM Access Analyzer untuk memvalidasi kebijakan IAM Anda guna memastikan izin yang aman dan berfungsi – IAM Access Analyzer memvalidasi kebijakan baru dan yang sudah ada sehingga kebijakan tersebut mematuhi bahasa kebijakan IAM (JSON) dan praktik terbaik IAM. IAM Access Analyzer menyediakan lebih dari 100 pemeriksaan kebijakan dan rekomendasi yang dapat ditindaklanjuti untuk membantu Anda membuat kebijakan yang aman dan fungsional. Untuk informasi selengkapnya, lihat validasi kebijakan Analizer Akses IAM di Panduan Pengguna IAM.
- Wajibkan autentikasi multi-faktor (MFA) – Jika Anda memiliki skenario yang mengharuskan pengguna IAM atau pengguna root di Akun AWS Anda, aktifkan MFA untuk keamanan tambahan. Untuk mewajibkan MFA saat operasi API dipanggil, tambahkan kondisi MFA pada kebijakan Anda. Untuk informasi selengkapnya, lihat Mengonfigurasi akses API yang dilindungi MFA di Panduan Pengguna IAM.

Untuk informasi selengkapnya tentang praktik terbaik dalam IAM, lihat Praktik terbaik keamanan di IAM di Panduan Pengguna IAM.

Menggunakan Layanan Terkelola untuk konsol Apache Flink

Untuk mengakses Amazon Managed Service untuk konsol Apache Flink, Anda harus memiliki set izin minimum. Izin ini harus memungkinkan Anda untuk membuat daftar dan melihat detail tentang Layanan Terkelola untuk sumber daya Apache Flink di sumber daya Anda. Akun AWS Jika Anda membuat kebijakan berbasis identitas yang lebih ketat daripada izin minimum yang diperlukan, konsol tidak akan berfungsi sebagaimana mestinya untuk entitas (pengguna atau peran) dengan kebijakan tersebut.

Anda tidak perlu memberikan izin konsol minimum untuk pengguna yang melakukan panggilan hanya ke AWS CLI atau API AWS. Sebaliknya, izinkan akses hanya ke tindakan yang cocok dengan operasi API yang coba dilakukan.

Untuk memastikan bahwa pengguna dan peran masih dapat menggunakan Managed Service for Apache Flink console, lampirkan juga Managed Service for Apache Flink `ConsoleAccess` atau

kebijakan `ReadOnly` AWS terkelola ke entitas. Untuk informasi selengkapnya, lihat Menambahkan izin ke pengguna di Panduan Pengguna IAM.

Izinkan pengguna melihat izin mereka sendiri

Contoh ini menunjukkan cara membuat kebijakan yang mengizinkan para pengguna IAM melihat kebijakan inline dan terkelola yang dilampirkan ke identitas pengguna mereka. Kebijakan ini mencakup izin untuk menyelesaikan tindakan pada konsol atau menggunakan AWS CLI atau AWS API secara terprogram.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Memecahkan masalah Amazon Managed Service untuk identitas dan akses Apache Flink

Gunakan informasi berikut untuk membantu Anda mendiagnosis dan memperbaiki masalah umum yang mungkin Anda temui saat bekerja dengan Layanan Terkelola untuk Apache Flink dan IAM.

Topik

Saya tidak berwenang untuk melakukan tindakan dalam Layanan Terkelola untuk Apache Flink

Jika AWS Management Console memberi tahu bahwa Anda tidak diotorisasi untuk melakukan tindakan, Anda harus menghubungi administrator untuk mendapatkan bantuan. Administrator adalah orang yang memberikan nama pengguna dan kata sandi kepada Anda.

Contoh kesalahan berikut terjadi ketika pengguna `mateojackson` mencoba menggunakan konsol untuk melihat detail tentang suatu sumber daya *`my-example-widget`* fiktif, tetapi tidak memiliki izin `Kinesis Analytics:`*`GetWidget`* fiktif.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform: Kinesis
 Analytics:GetWidget on resource: my-example-widget
```

Dalam hal ini, Mateo meminta administratornya untuk memperbarui kebijakan miliknya agar dia dapat mengakses sumber daya *`my-example-widget`* dengan menggunakan tindakan `Kinesis Analytics:`*`GetWidget`*.

Saya tidak berwenang untuk melakukan iam: PassRole

Jika Anda menerima kesalahan bahwa Anda tidak diizinkan untuk melakukan `iam:PassRole` tindakan, kebijakan Anda harus diperbarui agar Anda dapat meneruskan peran ke Layanan Terkelola untuk Apache Flink.

Sebagian Layanan AWS mengizinkan Anda untuk memberikan peran yang sudah ada ke layanan tersebut alih-alih membuat peran layanan baru atau peran terkait-layanan. Untuk melakukan tindakan tersebut, Anda harus memiliki izin untuk memberikan peran pada layanan tersebut.

Contoh kesalahan berikut terjadi ketika pengguna IAM bernama `marymajor` mencoba menggunakan konsol untuk melakukan tindakan di Managed Service for Apache Flink. Namun, tindakan tersebut memerlukan layanan untuk mendapatkan izin yang diberikan oleh peran layanan. Mary tidak memiliki izin untuk meneruskan peran tersebut pada layanan.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Dalam kasus ini, kebijakan Mary harus diperbarui agar dia mendapatkan izin untuk melakukan tindakan `iam:PassRole` tersebut.

Jika Anda membutuhkan bantuan, hubungi administrator AWS Anda. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Saya ingin mengizinkan orang di luar AWS akun saya untuk mengakses Layanan Terkelola saya untuk sumber daya Apache Flink

Anda dapat membuat peran yang dapat digunakan pengguna di akun lain atau pengguna di luar organisasi Anda untuk mengakses sumber daya Anda. Anda dapat menentukan siapa saja yang dipercaya untuk mengambil peran tersebut. Untuk layanan yang mendukung kebijakan berbasis sumber daya atau daftar kontrol akses (ACL), Anda dapat menggunakan kebijakan tersebut untuk memberi pengguna akses ke sumber daya Anda.

Untuk mempelajari selengkapnya, periksa hal berikut:

- Untuk mengetahui apakah Managed Service for Apache Flink mendukung fitur-fitur ini, lihat. Bagaimana Amazon Managed Service untuk Apache Flink bekerja dengan IAM
- Untuk mempelajari cara memberikan akses ke sumber daya di seluruh Akun AWS yang Anda miliki, lihat Menyediakan akses ke pengguna IAM di Akun AWS lainnya yang Anda miliki dalam Panduan Pengguna IAM.
- Untuk mempelajari cara memberikan akses ke sumber daya Anda ke pihak ketiga Akun AWS, lihat Menyediakan akses ke akun Akun AWS yang dimiliki oleh pihak ketiga dalam Panduan Pengguna IAM.
- Untuk mempelajari cara memberikan akses melalui federasi identitas, lihat Menyediakan akses ke pengguna terautentikasi eksternal (gabungan identitas) dalam Panduan Pengguna IAM.

- Untuk mempelajari perbedaan antara penggunaan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Perbedaan antara peran IAM dan kebijakan berbasis sumber daya di Panduan Pengguna IAM.

Pencegahan confused deputy lintas layanan

DalamAWS, peniruan lintas layanan dapat terjadi ketika satu layanan (layanan panggilan) memanggil layanan lain (layanan yang disebut). Layanan panggilan dapat dimanipulasi untuk bertindak atas sumber daya pelanggan lain meskipun seharusnya tidak memiliki izin yang tepat, yang mengakibatkan masalah wakil yang membingungkan.

Untuk mencegah kebingungan deputi, AWS sediakan alat yang membantu Anda melindungi data Anda untuk semua layanan menggunakan prinsip layanan yang telah diberikan akses ke sumber daya di akun Anda. Bagian ini berfokus pada pencegahan wakil bingung lintas layanan khusus untuk Layanan Terkelola untuk Apache Flink namun, Anda dapat mempelajari lebih lanjut tentang topik ini di Bagian masalah wakil yang bingung dari Panduan Pengguna IAM.

Dalam konteks Layanan Terkelola untuk Apache Flink, sebaiknya gunakan kunci konteks kondisi SourceAccount global aws: SourceArn and aws: dalam kebijakan kepercayaan peran Anda untuk membatasi akses ke peran hanya pada permintaan yang dihasilkan oleh sumber daya yang diharapkan.

Gunakan `aws:SourceArn` jika Anda hanya ingin satu sumber daya dikaitkan dengan akses lintas layanan. Gunakan `aws:SourceAccount` jika Anda ingin mengizinkan sumber daya apa pun di akun tersebut dikaitkan dengan penggunaan lintas layanan.

Nilai `aws:SourceArn` harus ARN dari sumber daya yang digunakan oleh Managed Service untuk Apache Flink, yang ditentukan dengan format berikut:. `arn:aws:kinesisanalytics:region:account:resource`

Pendekatan yang direkomendasikan untuk masalah wakil yang membingungkan adalah dengan menggunakan kunci konteks kondisi `aws:SourceArn` global dengan ARN sumber daya penuh.

Jika Anda tidak mengetahui ARN lengkap dari sumber daya atau jika Anda menentukan beberapa sumber daya, gunakan `aws:SourceArn` kunci dengan karakter wildcard (*) untuk bagian ARN yang tidak diketahui. Misalnya: `arn:aws:kinesisanalytics::111122223333:*`.

Kebijakan peran yang Anda berikan ke Layanan Terkelola untuk Apache Flink serta kebijakan kepercayaan peran yang dihasilkan untuk Anda dapat menggunakan kunci ini.

Untuk melindungi dari masalah wakil yang membingungkan, lakukan langkah-langkah berikut:

Untuk melindungi dari masalah wakil yang membingungkan

1. Masuk ke Konsol Manajemen AWS dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pilih Peran dan kemudian pilih peran yang ingin Anda ubah.
3. Pilih Edit kebijakan kepercayaan.
4. Pada halaman Edit kebijakan kepercayaan, ganti kebijakan JSON default dengan kebijakan yang menggunakan salah satu atau kedua kunci konteks kondisi `aws:SourceAccount` global. `aws:SourceArn` Lihat contoh kebijakan berikut:
5. Pilih Perbarui kebijakan.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"kinesisanalytics.amazonaws.com"
         },
         "Action":"sts:AssumeRole",
         "Condition":{
            "StringEquals":{
               "aws:SourceAccount":"Account ID"
            },
            "ArnEquals":{
               "aws:SourceArn":"arn:aws:kinesisanalytics:us-
east-1:123456789012:application/my-app"
            }
         }
      }
   ]
}
```

Memantau Layanan Terkelola untuk Apache Flink

Layanan Terkelola untuk Apache Flink menyediakan fungsionalitas pemantauan untuk aplikasi Anda. Untuk informasi selengkapnya, lihat Pencatatan dan Pemantauan.

Validasi Kepatuhan untuk Amazon Managed Service untuk Apache Flink

Auditor pihak ketiga menilai keamanan dan kepatuhan Amazon Managed Service untuk Apache Flink sebagai bagian dari beberapa AWS program kepatuhan. Hal ini mencakup SOC, PCI, HIPAA, dan lainnya.

Untuk daftar AWS layanan dalam lingkup program kepatuhan tertentu, lihat. Untuk informasi umum, lihat Program Kepatuhan AWS.

Anda dapat mengunduh laporan audit pihak ketiga menggunakan AWS Artifact. Untuk informasi selengkapnya, lihat Mengunduh Laporan di AWS Artifact.

Tanggung jawab kepatuhan Anda saat menggunakan Layanan Terkelola untuk Apache Flink ditentukan oleh sensitivitas data Anda, tujuan kepatuhan perusahaan Anda, dan hukum dan peraturan yang berlaku. Jika penggunaan Layanan Terkelola untuk Apache Flink tunduk pada kepatuhan terhadap standar seperti HIPAA atau PCI, AWS sediakan sumber daya untuk membantu:

- Panduan Quick Start Keamanan dan Kepatuhan – Panduan deployment ini membahas pertimbangan arsitektur dan memberikan langkah untuk menerapkan lingkungan dasar yang berfokus pada keamanan dan kepatuhan di AWS.
- Arsitektur untuk Keamanan dan Kepatuhan HIPAA di Amazon Web Services. Whitepaper ini menjelaskan bagaimana perusahaan dapat menggunakan AWS untuk membuat aplikasi yang sesuai dengan HIPAA.
- AWS Sumber Daya Kepatuhan – Kumpulan buku kerja dan panduan ini mungkin berlaku untuk industri dan lokasi Anda.
- AWS Config – Layanan AWS ini menilai seberapa baik konfigurasi sumber daya Anda mematuhi praktik internal, pedoman industri, dan peraturan.
- AWS Security Hub – Layanan AWS ini menyediakan tampilan komprehensif status keamanan Anda dalam AWS yang membantu Anda memeriksa kepatuhan Anda terhadap standar dan praktik terbaik industri.

FedRAMP

Program Kepatuhan AWS FedRAMP mencakup Layanan Terkelola untuk Apache Flink sebagai layanan resmi FedRAMP. Jika Anda adalah pelanggan federal atau komersial, Anda dapat

menggunakan layanan ini untuk memproses dan menyimpan beban kerja sensitif di batas otorisasi Wilayah AWS GovCloud (AS) dengan data hingga tingkat dampak tinggi, serta Wilayah Timur AS (Virginia N.), Timur AS (Ohio), AS Barat (California N.), Wilayah AS Barat (Oregon) dengan data hingga tingkat sedang.

Anda dapat meminta akses ke Paket Keamanan AWS FedRAMP melalui FedRAMP PMO atau Manajer Akun Penjualan AWS Anda atau, mereka dapat diunduh melalui Artifact di Artifact. AWSAWS

Untuk informasi lebih lanjut, lihat FedRAMP.

Ketahanan dalam Layanan Terkelola Amazon untuk Apache Flink

Infrastruktur global AWS dibangun di seputar Kawasan dan Zona Ketersediaan AWS. AWS Kawasan menyediakan beberapa Zona Ketersediaan yang terpisah dan terisolasi secara fisik, yang tersambung dengan jejaring jaringan latensi rendah, throughput tinggi, dan sangat redundan. Dengan Zona Ketersediaan, Anda dapat merancang dan mengoperasikan aplikasi dan basis data yang melakukan secara otomatis pinda saat gagal/failover di antara zona-zona tanpa terputus. Zona Ketersediaan lebih sangat tersedia, lebih toleran kesalahan, dan lebih dapat diskalakan daripadai infrastruktur pusat data tunggal atau multi tradisional.

Untuk informasi selengkapnya tentang Wilayah AWS dan Zona Ketersediaan, lihat Infrastruktur Global AWS.

Selain infrastruktur AWS global, Layanan Terkelola untuk Apache Flink menawarkan beberapa fitur untuk membantu mendukung ketahanan data dan kebutuhan cadangan Anda.

Pemulihan Bencana

Layanan Terkelola untuk Apache Flink berjalan dalam mode tanpa server, dan menangani degradasi host, ketersediaan Zona Ketersediaan, dan masalah terkait infrastruktur lainnya dengan melakukan migrasi otomatis. Layanan Terkelola untuk Apache Flink mencapai ini melalui beberapa mekanisme yang berlebihan. Setiap Layanan Terkelola untuk aplikasi Apache Flink berjalan dalam cluster Apache Flink penyewa tunggal. Cluster Apache Flink dijalankan dengan mode ketersediaan tinggi menggunakan Zookeeper JobMananger di beberapa zona ketersediaan. Layanan Terkelola untuk Apache Flink menyebarkan Apache Flink menggunakan Amazon EKS. Beberapa pod Kubernetes digunakan di Amazon EKS untuk setiap wilayah AWS di seluruh availability zone. Jika terjadi kegagalan, Managed Service for Apache Flink pertama kali mencoba memulihkan aplikasi dalam

cluster Apache Flink yang sedang berjalan menggunakan pos pemeriksaan aplikasi Anda, jika tersedia.

Layanan Terkelola untuk Apache Flink mencadangkan status aplikasi menggunakan Checkpoints dan Snapshots:

- Checkpoint adalah backup dari status aplikasi yang Managed Service untuk Apache Flink secara otomatis membuat secara berkala dan menggunakan untuk memulihkan dari kesalahan.
- Snapshot adalah cadangan dari status aplikasi yang Anda buat dan pulihkan secara manual.

Untuk informasi selengkapnya tentang titik pemeriksaan dan snapshot, lihat Toleransi Kesalahan.

Versioning

Versi status aplikasi yang disimpan dibuat versi sebagai berikut:

- Versi titik pemeriksaan dibuat secara otomatis oleh layanan. Jika layanan menggunakan titik pemeriksaan untuk memulai ulang aplikasi, titik pemeriksaan terbaru akan digunakan.
- Savepoints diversi menggunakan SnapshotNameparameter tindakan. CreateApplicationSnapshot

Layanan Terkelola untuk Apache Flink mengenkripsi data yang disimpan di pos pemeriksaan dan savepoint.

Keamanan Infrastruktur dalam Layanan Terkelola untuk Apache Flink

Sebagai layanan terkelola, Managed Service for Apache Flink dilindungi oleh prosedur keamanan jaringan AWS global yang dijelaskan dalam whitepaper Amazon Web Services: Overview of Security Processes.

Anda menggunakan panggilan API yang AWS dipublikasikan untuk mengakses Layanan Terkelola untuk Apache Flink melalui jaringan. Semua panggilan API ke Managed Service untuk Apache Flink diamankan melalui Transport Layer Security (TLS) dan diautentikasi melalui IAM. Klien harus mendukung TLS 1.2 atau yang lebih baru. Klien juga harus mendukung cipher suite dengan perfect forward secrecy (PFS) seperti Ephemeral Diffie-Hellman (DHE) atau Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Sebagian besar sistem modern seperti Java 7 dan sistem yang lebih baru mendukung mode ini.

Selain itu, permintaan harus ditandatangani dengan menggunakan ID kunci akses dan kunci akses rahasia yang terkait dengan pengguna utama IAM. Atau Anda dapat menggunakan AWS Security Token Service (AWS STS) untuk menghasilkan kredensial keamanan sementara untuk menandatangani permintaan.

Praktik Terbaik Keamanan untuk Layanan Terkelola untuk Apache Flink

Amazon Managed Service untuk Apache Flink menyediakan sejumlah fitur keamanan untuk dipertimbangkan saat Anda mengembangkan dan menerapkan kebijakan keamanan Anda sendiri. Praktik terbaik berikut adalah pedoman umum dan tidak mewakili solusi keamanan yang lengkap. Karena praktik terbaik ini mungkin tidak sesuai atau tidak memadai untuk lingkungan Anda, perlakukan itu sebagai pertimbangan yang bermanfaat, bukan sebagai resep.

Terapkan akses hak akses paling rendah

Saat memberikan izin, Anda memutuskan siapa yang mendapatkan izin apa untuk Layanan Terkelola untuk sumber daya Apache Flink. Anda memungkinkan tindakan tertentu yang ingin Anda lakukan di sumber daya tersebut. Oleh karena itu, Anda harus memberikan hanya izin yang diperlukan untuk melaksanakan tugas. Menerapkan akses hak istimewa yang terkecil adalah hal mendasar dalam mengurangi risiko keamanan dan dampak yang dapat diakibatkan oleh kesalahan atau niat jahat.

Gunakan IAM role untuk mengakses layanan Amazon lainnya

Layanan Terkelola untuk aplikasi Apache Flink Anda harus memiliki kredensil yang valid untuk mengakses sumber daya di layanan lain, seperti aliran data Kinesis, aliran Firehose Data Kinesis, atau bucket Amazon S3. Anda tidak boleh menyimpan kredensial AWS secara langsung di aplikasi atau bucket Amazon S3. Ini adalah kredensial jangka panjang yang tidak dirotasi secara otomatis dan dapat menimbulkan dampak bisnis yang signifikan jika dibobol.

Sebaliknya, Anda harus menggunakan IAM role untuk mengelola kredensial sementara untuk aplikasi guna mengakses sumber daya lainnya. Ketika Anda menggunakan peran, Anda tidak perlu menggunakan kredensi jangka panjang untuk mengakses sumber daya lain.

Untuk informasi selengkapnya, lihat topik berikut di Panduan Pengguna IAM:

- Peran IAM
- Skenario Umum untuk Peran: Pengguna, Aplikasi, dan Layanan

Terapkan Enkripsi Sisi Server di Sumber Daya Dependen

Data saat istirahat dan data dalam perjalanan dienkripsi dalam Layanan Terkelola untuk Apache Flink, dan enkripsi ini tidak dapat dinonaktifkan. Anda harus menerapkan enkripsi sisi server di sumber daya dependen Anda, seperti aliran data Kinesis, aliran Firehose Data Kinesis, dan bucket Amazon S3. Untuk informasi selengkapnya tentang menerapkan enkripsi sisi server dalam sumber daya dependen, lihat Perlindungan Data.

Gunakan CloudTrail untuk Memantau Panggilan API

Layanan Terkelola untuk Apache Flink terintegrasi denganAWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau layanan Amazon di Layanan Terkelola untuk Apache Flink.

Dengan menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan permintaan yang dibuat untuk Layanan Terkelola untuk Apache Flink, alamat IP dari mana permintaan dibuat, siapa yang membuat permintaan, kapan dibuat, dan detail tambahan.

Untuk informasi selengkapnya, lihat the section called “Menggunakan AWS CloudTrail”.

Logging dan Monitoring di Amazon Managed Service untuk Apache Flink

Pemantauan adalah bagian penting dari menjaga keandalan, ketersediaan, dan kinerja Managed Service untuk aplikasi Apache Flink. Anda harus mengumpulkan data pemantauan dari semua bagian solusi AWS Anda sehingga Anda dapat dengan lebih mudah melakukan debug kegagalan multipoin jika ada yang terjadi.

Sebelum Anda mulai memantau Managed Service untuk Apache Flink, Anda harus membuat rencana pemantauan yang mencakup jawaban atas pertanyaan-pertanyaan berikut:

- Apa saja sasaran pemantauan Anda?
- Sumber daya apa yang akan Anda pantau?
- Seberapa sering Anda akan memantau sumber daya ini?
- Alat pemantauan apa yang akan Anda gunakan?
- Siapa yang akan melakukan tugas pemantauan?
- Siapa yang harus diberi tahu saat terjadi kesalahan?

Langkah selanjutnya adalah menetapkan garis dasar untuk Layanan Terkelola normal untuk kinerja Apache Flink di lingkungan Anda. Anda melakukan ini dengan mengukur performa pada berbagai waktu dan di bawah tingkat kesesuaian beban yang berbeda. Saat Anda memantau Layanan Terkelola untuk Apache Flink, Anda dapat menyimpan data pemantauan historis. Anda selanjutnya dapat membandingkannya dengan data performa saat ini, mengidentifikasi pola performa normal dan anomali performa, serta merancang metode untuk mengatasi masalah.

Topik

Pencatatan log

Logging penting bagi aplikasi produksi untuk memahami kesalahan dan kegagalan. Namun, subsistem logging perlu mengumpulkan dan meneruskan entri log ke Log Sementara beberapa logging baik-baik saja dan diinginkan, logging ekstensif dapat membebani layanan dan menyebabkan aplikasi Flink tertinggal. CloudWatch Pengecualian dan peringatan logging tentu merupakan ide yang bagus. Tetapi Anda tidak dapat membuat pesan log untuk setiap pesan yang diproses oleh aplikasi Flink. Flink dioptimalkan untuk seluruh latensi tinggi dan latensi rendah, subsistem logging tidak. Jika benar-benar diperlukan untuk menghasilkan output log untuk setiap pesan yang diproses, gunakan tambahan DataStream di dalam aplikasi Flink dan wastafel yang tepat untuk mengirim data ke Amazon CloudWatch S3 atau. Jangan gunakan sistem logging Java untuk tujuan ini. Selain itu, Managed Service untuk `Debug Monitoring Log Level` pengaturan Apache Flink menghasilkan sejumlah besar lalu lintas, yang dapat menciptakan tekanan balik. Anda hanya harus menggunakannya saat secara aktif menyelidiki masalah dengan aplikasi.

Menanyakan Log dengan Wawasan CloudWatch Log

CloudWatch Logs Insights adalah layanan yang ampuh untuk menanyakan log dalam skala besar. Pelanggan harus memanfaatkan kemampuannya untuk dengan cepat mencari melalui log untuk mengidentifikasi dan mengurangi kesalahan selama acara operasional.

Kueri berikut mencari pengecualian di semua log pengelola tugas dan memesannya sesuai dengan waktu terjadinya.

```
fields @timestamp, @message
| filter isPresent(throwableInformation.0) or isPresent(throwableInformation) or
 @message like /(Error|Exception)/
| sort @timestamp desc
```

Untuk kueri berguna lainnya, lihat Contoh Kueri.

Pemantauan

Saat menjalankan aplikasi streaming dalam produksi, Anda mulai menjalankan aplikasi secara terus menerus dan tanpa batas waktu. Sangat penting untuk menerapkan pemantauan dan pengkhawatiran yang tepat dari semua komponen tidak hanya aplikasi Flink. Jika tidak, Anda berisiko melewatkan masalah yang muncul sejak dini dan hanya menyadari peristiwa operasional setelah sepenuhnya terurai dan jauh lebih sulit untuk dikurangi. Hal-hal umum yang harus dipantau meliputi:

- Apakah sumbernya menelan data?
- Apakah data dibaca dari sumber (dari perspektif sumber)?
- Apakah aplikasi Flink menerima data?
- Apakah aplikasi Flink dapat mengikuti atau tertinggal?
- Apakah aplikasi Flink menyimpan data ke wastafel (dari perspektif aplikasi)?
- Apakah wastafel menerima data?

Metrik yang lebih spesifik kemudian harus dipertimbangkan untuk aplikasi Flink. CloudWatch Dasbor ini memberikan titik awal yang baik. Untuk informasi selengkapnya tentang metrik apa yang harus dipantau untuk aplikasi produksi, lihatMenggunakan CloudWatch Alarm dengan Amazon Managed Service untuk Apache Flink. Metrik ini meliputi:

- records_lag_max dan MillisBehindLatest — Jika aplikasi menggunakan Kinesis atau Kafka, metrik ini menunjukkan apakah aplikasi tertinggal dan perlu diskalakan untuk mengikuti beban saat ini. Ini adalah metrik generik yang baik yang mudah dilacak untuk semua jenis aplikasi. Tetapi itu hanya dapat digunakan untuk penskalaan reaktif, yaitu, ketika aplikasi sudah tertinggal.
- CPUutilization heapMemoryUtilizationdan — Metrik ini memberikan indikasi yang baik tentang pemanfaatan sumber daya keseluruhan aplikasi dan dapat digunakan untuk penskalaan proaktif kecuali aplikasi terikat I/O.
- downtime — Downtime yang lebih besar dari nol menunjukkan bahwa aplikasi telah gagal. Jika nilainya lebih besar dari 0, aplikasi tidak memproses data apa pun.
- lastCheckpointSizedan lastCheckpointDuration— Metrik ini memantau berapa banyak data yang disimpan dalam keadaan dan berapa lama waktu yang dibutuhkan untuk mengambil pos pemeriksaan. Jika pos pemeriksaan bertambah atau memakan waktu lama, aplikasi terus menghabiskan waktu untuk pos pemeriksaan dan memiliki lebih sedikit siklus untuk pemrosesan yang sebenarnya. Di beberapa titik, pos pemeriksaan mungkin tumbuh terlalu besar atau memakan waktu lama sehingga gagal. Selain memantau nilai absolut, pelanggan juga harus mempertimbangkan untuk memantau tingkat perubahan dengan `RATE(lastCheckpointSize)` dan`RATE(lastCheckpointDuration)`.
- numberOfFailedCheckpoints — Metrik ini menghitung jumlah pos pemeriksaan yang gagal sejak aplikasi dimulai. Tergantung pada aplikasinya, itu bisa ditoleransi jika pos pemeriksaan gagal sesekali. Tetapi jika pos pemeriksaan secara teratur gagal, aplikasi tersebut kemungkinan tidak sehat dan perlu perhatian lebih lanjut. Kami merekomendasikan pemantauan `RATE(numberOfFailedCheckpoints)` untuk alarm pada gradien dan bukan pada nilai absolut.

Menyiapkan Pencatatan Aplikasi

Dengan menambahkan opsi CloudWatch pencatatan Amazon ke Layanan Terkelola untuk aplikasi Apache Flink, Anda dapat memantau peristiwa aplikasi atau masalah konfigurasi.

Topik ini menjelaskan cara mengonfigurasi aplikasi Anda untuk menulis peristiwa aplikasi ke aliran CloudWatch Log. Opsi CloudWatch logging adalah kumpulan pengaturan aplikasi dan izin yang digunakan aplikasi Anda untuk mengonfigurasi cara menulis peristiwa aplikasi ke CloudWatch Log. Anda dapat menambahkan dan mengonfigurasi opsi CloudWatch logging menggunakan salah satu AWS Management Console atau AWS Command Line Interface (AWS CLI).

Perhatikan hal berikut tentang menambahkan opsi CloudWatch logging ke aplikasi Anda:

- Saat Anda menambahkan opsi CloudWatch logging menggunakan konsol, Managed Service for Apache Flink membuat grup CloudWatch log dan aliran log untuk Anda dan menambahkan izin yang perlu ditulis aplikasi Anda ke aliran log.
- Saat Anda menambahkan opsi CloudWatch logging menggunakan API, Anda juga harus membuat grup log aplikasi dan aliran log, dan menambahkan izin yang dibutuhkan aplikasi Anda untuk menulis ke aliran log.

Topik ini berisi bagian-bagian berikut:

Menyiapkan CloudWatch Logging Menggunakan Konsol

Saat Anda mengaktifkan CloudWatch pencatatan untuk aplikasi Anda di konsol, grup CloudWatch log dan aliran log dibuat untuk Anda. Selain itu, kebijakan izin aplikasi Anda diperbarui dengan izin untuk menulis ke aliran.

Layanan Terkelola untuk Apache Flink membuat grup log bernama menggunakan konvensi berikut, di *`ApplicationName`*mana nama aplikasi Anda.

```
/AWS/KinesisAnalytics/ApplicationName
```

Layanan Terkelola untuk Apache Flink membuat aliran log di grup log baru dengan nama berikut.

```
kinesis-analytics-log-stream
```

Anda menetapkan tingkat metrik pemantauan aplikasi dan tingkat log pemantauan menggunakan bagian Tingkat log pemantauan di halaman Konfigurasikan aplikasi. Untuk informasi tentang tingkat log aplikasi, lihat the section called “TingkatPemantauan Aplikasi”.

Menyiapkan CloudWatch Logging Menggunakan CLI

Untuk menambahkan opsi CloudWatch logging menggunakanAWS CLI, lakukan hal berikut:

- Buat grup CloudWatch log dan aliran log.
- Tambahkan opsi logging saat Anda membuat aplikasi dengan menggunakan CreateApplicationtindakan, atau tambahkan opsi logging ke aplikasi yang ada menggunakan AddApplicationCloudWatchLoggingOptiontindakan.
- Tambahkan izin ke kebijakan aplikasi Anda untuk menulis ke log.

Bagian ini berisi topik berikut:

- Membuat Grup CloudWatch Log dan Aliran Log
- Bekerja dengan Opsi CloudWatch Pencatatan Aplikasi
- Menambahkan Izin untuk Menulis ke Aliran CloudWatch Log

Membuat Grup CloudWatch Log dan Aliran Log

Anda membuat grup CloudWatch log dan melakukan streaming menggunakan konsol CloudWatch Log atau API. Untuk informasi tentang membuat grup CloudWatch log dan aliran log, lihat Bekerja dengan Grup Log dan Aliran Log.

Bekerja dengan Opsi CloudWatch Pencatatan Aplikasi

Gunakan tindakan API berikut untuk menambahkan opsi CloudWatch log ke aplikasi baru atau yang sudah ada atau ubah opsi log untuk aplikasi yang sudah ada. Untuk informasi tentang cara

menggunakan file JSON untuk input tindakan API, lihat Layanan Terkelola untuk Kode Contoh API Apache Flink.

Menambahkan Opsi CloudWatch Log Saat Membuat Aplikasi

Contoh berikut menunjukkan cara menggunakan `CreateApplication` tindakan untuk menambahkan opsi CloudWatch log saat Anda membuat aplikasi. Dalam contoh, ganti *Amazon Resource Name (ARN) dari aliran CloudWatch Log untuk menambahkan ke aplikasi baru dengan informasi* Anda sendiri. Untuk informasi selengkapnya tentang tindakan, lihat `CreateApplication`.

```
{
    "ApplicationName": "test",
    "ApplicationDescription": "test-application-description",
    "RuntimeEnvironment": "FLINK-1_15",
    "ServiceExecutionRole": "arn:aws:iam::123456789123:role/myrole",
    "ApplicationConfiguration": {
        "ApplicationCodeConfiguration": {
            "CodeContent": {
                "S3ContentLocation":{
                            "BucketARN": "arn:aws:s3:::mybucket",
                            "FileKey": "myflink.jar"
                }
            },
            "CodeContentType": "ZIPFILE"
        }
    },
    "CloudWatchLoggingOptions": [{
      "LogStreamARN": "<Amazon Resource Name (ARN) of the CloudWatch log stream to add
 to the new application>"
 }]
}
```

Menambahkan Opsi CloudWatch Log ke Aplikasi yang Ada

Contoh berikut menunjukkan cara menggunakan `AddApplicationCloudWatchLoggingOption` tindakan untuk menambahkan opsi CloudWatch log ke aplikasi yang ada. Dalam contoh, ganti setiap *placeholder input pengguna* dengan informasi Anda sendiri. Untuk informasi selengkapnya tentang tindakan, lihat `AddApplicationCloudWatchLoggingOption`.

```
{
   "ApplicationName": "<Name of the application to add the log option to>",
```

```
   "CloudWatchLoggingOption": {
      "LogStreamARN": "<ARN of the log stream to add to the application>"
   },
   "CurrentApplicationVersionId": <Version of the application to add the log to>
}
```

Memperbarui Opsi CloudWatch Log yang Ada

Contoh berikut menunjukkan bagaimana menggunakan `UpdateApplication` tindakan untuk memodifikasi opsi CloudWatch log yang ada. Dalam contoh, ganti setiap *`placeholder input pengguna`* dengan informasi Anda sendiri. Untuk informasi selengkapnya tentang tindakan, lihat `UpdateApplication`.

```
{
   "ApplicationName": "<Name of the application to update the log option for>",
   "CloudWatchLoggingOptionUpdates": [
         {
            "CloudWatchLoggingOptionId": "<ID of the logging option to modify>",
            "LogStreamARNUpdate": "<ARN of the new log stream to use>"
         }
      ],
   "CurrentApplicationVersionId": <ID of the application version to modify>
}
```

Menghapus Opsi CloudWatch Log dari Aplikasi

Contoh berikut menunjukkan cara menggunakan `DeleteApplicationCloudWatchLoggingOption` tindakan untuk menghapus opsi CloudWatch log yang ada. Dalam contoh, ganti setiap *`placeholder input pengguna`* dengan informasi Anda sendiri. Untuk informasi selengkapnya tentang tindakan, lihat `DeleteApplicationCloudWatchLoggingOption`.

```
{
   "ApplicationName": "<Name of application to delete log option from>",
   "CloudWatchLoggingOptionId": "<ID of the application log option to delete>",
   "CurrentApplicationVersionId": <Version of the application to delete the log option
 from>
}
```

Mengatur Tingkat Pencatatan Aplikasi

Untuk mengatur tingkat pencatatan aplikasi, gunakan parameter `MonitoringConfiguration` tindakan `CreateApplication` atau parameter `MonitoringConfigurationUpdate` tindakan `UpdateApplication`.

Untuk informasi tentang tingkat log aplikasi, lihat the section called “TingkatPemantauan Aplikasi”.

Atur Tingkat Pencatatan Aplikasi saat Membuat Aplikasi

Permintaan contoh berikut untuk tindakan `CreateApplication` menetapkan tingkat log aplikasi ke `INFO`.

```
{
   "ApplicationName": "MyApplication",
   "ApplicationDescription": "My Application Description",
   "ApplicationConfiguration": {
      "ApplicationCodeConfiguration":{
      "CodeContent":{
        "S3ContentLocation":{
          "BucketARN":"arn:aws:s3:::mybucket",
          "FileKey":"myflink.jar",
          "ObjectVersion":"AbCdEfGhIjKlMnOpQrStUvWxYz12345"
        }
      },
      "CodeContentType":"ZIPFILE"
      },
      "FlinkApplicationConfiguration":
         "MonitoringConfiguration": {
            "ConfigurationType": "CUSTOM",
            "LogLevel": "INFO"
         }
      },
   "RuntimeEnvironment": "FLINK-1_15",
   "ServiceExecutionRole": "arn:aws:iam::123456789123:role/myrole"
}
```

Perbarui Tingkat Pencatatan Aplikasi

Permintaan contoh berikut untuk tindakan `UpdateApplication` menetapkan tingkat log aplikasi ke `INFO`.

```
{
   "ApplicationConfigurationUpdate": {
      "FlinkApplicationConfigurationUpdate": {
         "MonitoringConfigurationUpdate": {
            "ConfigurationTypeUpdate": "CUSTOM",
            "LogLevelUpdate": "INFO"
         }
      }
   }
}
```

Menambahkan Izin untuk Menulis ke Aliran CloudWatch Log

Layanan Terkelola untuk Apache Flink memerlukan izin untuk menulis kesalahan konfigurasi. CloudWatch Anda dapat menambahkan izin ini ke peran AWS Identity and Access Management (IAM) yang diasumsikan oleh Managed Service for Apache Flink.

Untuk informasi selengkapnya tentang menggunakan peran IAM untuk Layanan Terkelola untuk Apache Flink, lihat. Identity and Access Management untuk Amazon Managed Service untuk Apache Flink

Kebijakan Kepercayaan

Untuk memberikan izin Layanan Terkelola untuk Apache Flink untuk mengambil peran IAM, Anda dapat melampirkan kebijakan kepercayaan berikut ke peran eksekusi layanan.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "kinesisanlaytics.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Kebijakan Izin

Untuk memberikan izin ke aplikasi untuk menulis peristiwa log CloudWatch dari Layanan Terkelola untuk sumber daya Apache Flink, Anda dapat menggunakan kebijakan izin IAM berikut. Sediakan Amazon Resource Names (ARN) yang benar untuk grup log dan aliran Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt0123456789000",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents",
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:123456789012:log-group:my-log-group:log-
stream:my-log-stream*",
                "arn:aws:logs:us-east-1:123456789012:log-group:my-log-group:*",
                "arn:aws:logs:us-east-1:123456789012:log-group:*",
            ]
        }
    ]
}
```

TingkatPemantauan Aplikasi

Anda mengontrol pembuatan pesan log aplikasi menggunakan Tingkat Metrik Pemantauan dan Tingkat Log Pemantauan aplikasi.

Tingkat metrik pemantauan aplikasi mengontrol granularitas pesan log. Memantau tingkat metrik ditentukan sebagai berikut:

- Aplikasi: Metrik dicakup untuk seluruh aplikasi.
- Tugas: Metrik dicakup untuk setiap tugas. Untuk informasi tentang tugas , lihat the section called “Penskalaan”.
- Operator: Metrik dicakup untuk setiap operator. Untuk informasi tentang operator, lihat the section called “DataStream Operator API”.

- Paralelisme: Metrik dicakup untuk paralelisme aplikasi. Anda hanya dapat mengatur level metrik ini menggunakan MonitoringConfigurationUpdateparameter UpdateApplicationAPI. Anda tidak dapat mengatur tingkat metrik ini menggunakan konsol. Untuk informasi tentang paralelisme, lihat the section called “Penskalaan”.

Tingkat log pemantauan aplikasi mengontrol verbositas log aplikasi. Memantau tingkat log ditentukan sebagai berikut:

- Kesalahan: Potensi peristiwa bencana aplikasi.
- Peringatan: Situasi aplikasi yang berpotensi berbahaya.
- Info: Informasi dan peristiwa kegagalan sementara aplikasi. Sebaiknya Anda menggunakan tingkat pencatatan ini.
- Debug: Peristiwa informasi terperinci yang paling berguna untuk melakukan debug aplikasi. Catatan: Hanya gunakan tingkat ini untuk tujuan debugging sementara.

Praktik Terbaik Pencatatan

Sebaiknya aplikasi Anda menggunakan tingkat pencatatan Info. Kami merekomendasikan tingkat ini untuk memastikan Anda melihat kesalahan Apache Flink, yang dicatat di tingkat Info bukan di tingkat Kesalahan.

Sebaiknya gunakan tingkat Debug hanya sementara saat menyelidiki masalah aplikasi. Alihkan kembali ke tingkat Info saat masalah teratasi. Menggunakan tingkat pencatatan Debug secara signifikan akan memengaruhi performa aplikasi Anda.

Pencatatan berlebihan juga dapat berdampak signifikan terhadap performa aplikasi. Sebaiknya jangan tulis entri log untuk setiap catatan yang diproses, misalnya. Pencatatan berlebihan dapat menyebabkan hambatan parah dalam pemrosesan data dan dapat menyebabkan tekanan balik dalam membaca data dari sumber.

Pemecahan Masalah Pencatatan

Jika log aplikasi tidak ditulis ke aliran log, verifikasi hal berikut:

- Verifikasi bahwa IAM role dan kebijakan IAM aplikasi Anda sudah benar. Kebijakan aplikasi Anda memerlukan izin berikut untuk mengakses aliran log Anda:
  - `logs:PutLogEvents`

- `logs:DescribeLogGroups`
- `logs:DescribeLogStreams`

  Untuk informasi selengkapnya, lihat the section called “Menambahkan Izin untuk Menulis ke Aliran CloudWatch Log”.
- Verifikasi bahwa aplikasi Anda sedang berjalan Untuk memeriksa status aplikasi Anda, lihat halaman aplikasi Anda di konsol, atau gunakan ListApplicationstindakan DescribeApplicationatau.
- Pantau CloudWatch metrik seperti `downtime` untuk mendiagnosis masalah aplikasi lainnya. Untuk informasi tentang membaca CloudWatch metrik, lihatMetrik dan Dimensi dalam Layanan Terkelola untuk Apache Flink.

Langkah Selanjutnya

Setelah Anda mengaktifkan CloudWatch login di aplikasi Anda, Anda dapat menggunakan Wawasan CloudWatch Log untuk menganalisis log aplikasi Anda. Untuk informasi selengkapnya, lihat the section called “Menganalisis Log”.

Menganalisis Log dengan Wawasan CloudWatch Log

Setelah menambahkan opsi CloudWatch pencatatan ke aplikasi seperti yang dijelaskan di bagian sebelumnya, Anda dapat menggunakan Wawasan CloudWatch Log untuk menanyakan aliran log Anda untuk peristiwa atau kesalahan tertentu.

CloudWatch Logs Insights memungkinkan Anda untuk secara interaktif mencari dan menganalisis data log Anda di CloudWatch Log.

Untuk informasi tentang memulai Wawasan CloudWatch Log, lihat Menganalisis Data Log dengan Wawasan CloudWatch Log.

Jalankan Kueri Sampel

Bagian ini menjelaskan cara menjalankan contoh kueri Wawasan CloudWatch Log.

Prasyarat

- Grup log dan aliran log yang ada disiapkan di CloudWatch Log.
- Log yang ada disimpan di CloudWatch Log.

Jika Anda menggunakan layanan sepertiAWS CloudTrail, Amazon Route 53, atau Amazon VPC, Anda mungkin sudah menyiapkan log dari layanan tersebut untuk masuk ke CloudWatch Log. Untuk informasi selengkapnya tentang mengirim CloudWatch log ke Log, lihat Memulai dengan CloudWatch Log.

Kueri dalam Wawasan CloudWatch Log mengembalikan sekumpulan bidang dari peristiwa log, atau hasil agregasi matematis atau operasi lain yang dilakukan pada peristiwa log. Bagian ini menunjukkan kueri yang mengembalikan daftar log acara.

Untuk menjalankan kueri sampel Wawasan CloudWatch Log

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi, pilih Insights (Wawasan).
3. Editor kueri di dekat bagian atas layar berisi kueri default yang mengembalikan 20 log acara terbaru. Di atas editor kueri, pilih satu grup log yang akan dikueri.

   Saat Anda memilih grup CloudWatch log, Wawasan Log secara otomatis mendeteksi bidang dalam data dalam grup log dan menampilkannya di bidang Ditemukan di panel kanan. Panel ini juga menampilkan grafik batang log acara dalam grup log ini dari waktu ke waktu. Grafik batang ini menunjukkan distribusi peristiwa dalam grup log yang sesuai dengan kueri dan rentang waktu Anda, bukan hanya peristiwa yang ditampilkan dalam tabel.
4. Pilih Run query (Jalankan kueri).

   Hasil kueri muncul. Dalam contoh ini, hasilnya adalah 20 log acara terbaru dari tipe apa pun.
5. Untuk melihat semua bidang untuk salah satu log acara yang ditampilkan, pilih panah di sebelah kiri log acara tersebut.

Untuk informasi selengkapnya tentang cara menjalankan dan memodifikasi kueri Wawasan CloudWatch Log, lihat Menjalankan dan Memodifikasi Kueri Contoh.

Kueri Sampel

Bagian ini berisi kueri contoh Wawasan CloudWatch Log untuk menganalisis Layanan Terkelola untuk log aplikasi Apache Flink. Kueri ini mencari beberapa contoh kondisi kesalahan, dan berfungsi sebagai templat untuk menulis kueri yang menemukan kondisi kesalahan lainnya.

> **Note**
>
> Ganti Region (*us-barat-2*), ID Akun (*012345678901*) dan nama aplikasi (*YourApplication*) dalam contoh kueri berikut dengan Region aplikasi dan ID Akun Anda.

Topik ini berisi bagian-bagian berikut:

- Analisis Operasi: Distribusi Tugas
- Analisis Operasi: Perubahan dalam Paralelisme
- Analisis Kesalahan: Akses Ditolak
- Analisis Kesalahan: Sumber atau Sink Tidak Ditemukan
- Analisis Kesalahan: Kegagalan Terkait Tugas Aplikasi

Analisis Operasi: Distribusi Tugas

Kueri CloudWatch Logs Insights berikut menampilkan jumlah tugas yang didistribusikan oleh Apache Flink Job Manager antar Task Manager. Anda perlu mengatur kerangka waktu kueri untuk mencocokkan satu tugas yang berjalan sehingga kueri tidak menampilkan tugas dari tugas sebelumnya. Untuk informasi selengkapnya tentang Paralelisme, lihat Penskalaan.

```
fields @timestamp, message
| filter message like /Deploying/
| parse message " to flink-taskmanager-*" as @tmid
| stats count(*) by @tmid
| sort @timestamp desc
| limit 2000
```

Kueri Wawasan CloudWatch Log berikut menampilkan subtugas yang ditetapkan ke setiap Task Manager. Jumlah total subtugas adalah jumlah paralelisme setiap tugas. Paralelisme tugas berasal dari paralelisme operator, dan sama dengan paralelisme aplikasi secara default, kecuali jika Anda mengubahnya dalam kode dengan menentukan `setParallelism`. Untuk informasi selengkapnya tentang pengaturan paralelisme operator, lihat Mengatur Paralelisme: Tingkat Operator di Dokumentasi Apache Flink.

```
fields @timestamp, @tmid, @subtask
| filter message like /Deploying/
```

```
| parse message "Deploying * to flink-taskmanager-*" as @subtask, @tmid
| sort @timestamp desc
| limit 2000
```

Untuk informasi selengkapnya tentang penjadwalan tugas, lihat Tugas dan Penjadwalan di Dokumentasi Apache Flink.

Analisis Operasi: Perubahan dalam Paralelisme

Kueri CloudWatch Logs Insights berikut mengembalikan perubahan pada paralelisme aplikasi (misalnya, karena penskalaan otomatis). Kueri ini juga menampilkan perubahan manual paralelisme aplikasi. Untuk informasi selengkapnya tentang penskalaan otomatis, lihat the section called "Penskalaan Otomatis".

```
fields @timestamp, @parallelism
| filter message like /property: parallelism.default, /
| parse message "default, *" as @parallelism
| sort @timestamp asc
```

Analisis Kesalahan: Akses Ditolak

Kueri CloudWatch Logs Insights berikut mengembalikan `Access Denied` log.

```
fields @timestamp, @message, @messageType
| filter applicationARN like /arn:aws:kinesisanalytics<us-
west-2>:<012345678901>:application\/<YourApplication>/
| filter @message like /AccessDenied/
| sort @timestamp desc
```

Analisis Kesalahan: Sumber atau Sink Tidak Ditemukan

Kueri CloudWatch Logs Insights berikut mengembalikan `ResourceNotFound` log. `ResourceNotFound`log dihasilkan jika sumber Kinesis atau wastafel tidak ditemukan.

```
fields @timestamp,@message
| filter applicationARN like /arn:aws:kinesisanalytics<us-
west-2>:<012345678901>:application\/<YourApplication>/
| filter @message like /ResourceNotFoundException/
| sort @timestamp desc
```

Analisis Kesalahan: Kegagalan Terkait Tugas Aplikasi

Kueri CloudWatch Logs Insights berikut menampilkan log kegagalan terkait tugas aplikasi. Ini mencatat hasil jika status aplikasi beralih dari `RUNNING` ke `RESTARTING`.

```
fields @timestamp,@message
| filter applicationARN like /arn:aws:kinesisanalytics*us-
west-2*:*012345678901*:application\/*YourApplication*/
| filter @message like /switched from RUNNING to RESTARTING/
| sort @timestamp desc
```

Untuk aplikasi yang menggunakan Apache Flink versi 1.8.2 dan sebelumnya, kegagalan terkait tugas akan mengakibatkan perubahan status aplikasi dari `RUNNING` ke `FAILED` sebagai gantinya. Ketika menggunakan Apache Flink 1.8.2 dan sebelumnya, gunakan kueri berikut untuk mencari kegagalan terkait tugas aplikasi:

```
fields @timestamp,@message
| filter applicationARN like /arn:aws:kinesisanalytics*us-
west-2*:*012345678901*:application\/*YourApplication*/
| filter @message like /switched from RUNNING to FAILED/
| sort @timestamp desc
```

Melihat Metrik dan Dimensi dalam Layanan Terkelola untuk Apache Flink

Topik ini berisi bagian-bagian berikut:

- Metrik Aplikasi
- Metrik Konektor Kinesis Data Streams
- Metrik Konektor Amazon MSK
- Metrik Apache Zeppelin
- Melihat CloudWatch Metrik
- Mengatur Tingkat Pelaporan CloudWatch Metrik
- Menggunakan Metrik Kustom dengan Amazon Managed Service untuk Apache Flink
- Menggunakan CloudWatch Alarm dengan Amazon Managed Service untuk Apache Flink

Saat Layanan Terkelola untuk Apache Flink memproses sumber data, Managed Service for Apache Flink melaporkan metrik dan dimensi berikut ke Amazon. CloudWatch

Metrik Aplikasi

Metrik	Unit	Deskripsi	Tingkat	Catatan Penggunaan	
`backPressuredTimeMsPerSecond*`	Milidetik	Waktu (dalam milidetik) tugas atau operator ini kembali ditekan per detik.	Tugas, Operator, Paralelisme	*Tersedia untuk Managed Service untuk aplikasi Apache Flink yang menjalankan Flink versi 1.13 saja. Metrik ini dapat berguna dalam mengidentifikasi kemacetan dalam suatu aplikasi.	
`busyTimeMsPerSecond*`	Milidetik	Waktu (dalam milidetik) tugas atau operator ini sibuk (tidak menganggur atau kembali ditekan) per detik. Bisa NaN, jika	Tugas, Operator, Paralelisme	*Tersedia untuk Managed Service untuk aplikasi Apache Flink yang menjalankan Flink versi 1.13 saja.	

Metrik	Unit	Deskripsi	Tingkat	Catatan Penggunaan	
		nilainya tidak bisa dihitung.		Metrik ini dapat berguna dalam mengident ifikasi kemacetan dalam suatu aplikasi.	
`cpuUtiliz ation`	Persentase	Keseluruhan persentase penggunaan CPU di seluruh manajer tugas. Misalnya, jika ada lima pengelola tugas, Managed Service for Apache Flink menerbitkan lima sampel metrik ini per interval pelaporan.	Aplikasi	Anda dapat menggunakan metrik ini untuk memantau penggunaan CPU minimum, rata-rata, dan maksimum dalam aplikasi Anda. `CPUUtiliz ation` Metrik hanya memperhit ungkan penggunaa n CPU dari proses TaskManag er JVM yang berjalan di dalam wadah.	

Metrik	Unit	Deskripsi	Tingkat	Catatan Penggunaan	
`containerCPUUtilization`	Persentase	Persentase keseluruhan pemanfaatan CPU di seluruh wadah task manager di cluster aplikasi Flink. Misalnya, jika ada lima pengelola tugas, maka ada lima TaskManager kontainer dan Layanan Terkelola untuk Apache Flink menerbitkan 2 * lima sampel metrik ini per interval pelaporan 1 menit.	Aplikasi	Itu dihitung per kontainer sebagai: Total waktu CPU (dalam detik) yang dikonsumsi oleh wadah * 100/batas CPU kontainer (dalam CPU/detik) `CPUUtilization` Metrik hanya memperhitungkan penggunaan CPU dari proses TaskManager JVM yang berjalan di dalam wadah. Ada komponen lain yang berjalan di luar JVM dalam wadah yang sama. `containerCPUUtiliz`	

Metrik	Unit	Deskripsi	Tingkat	Catatan Penggunaan	
				`ation` Metrik memberi Anda gambaran yang lebih lengkap, termasuk semua proses dalam hal kelelahan CPU di wadah dan kegagalan yang dihasilkan dari itu.	

Metrik	Unit	Deskripsi	Tingkat	Catatan Penggunaan	
`containerMemoryUtilization`	Persentase	Persentase keseluruhan pemanfaatan memori di seluruh wadah pengelola tugas di cluster aplikasi Flink. Misalnya, jika ada lima pengelola tugas, maka ada lima TaskManager kontainer dan Layanan Terkelola untuk Apache Flink menerbitkan 2 * lima sampel metrik ini per interval pelaporan 1 menit.	Aplikasi	Itu dihitung per kontainer sebagai: Penggunaan memori kontainer (byte) * 100/ Batas memori kontainer sesuai spesifikasi penerapan pod (dalam byte) Metrik `HeapMemoryUtilization` dan hanya memperhitungkan `ManagedMemoryUtilzations` metrik memori tertentu seperti Heap Memory Usage of TaskManager JVM atau Managed Memory (pengguna	

Metrik	Unit	Deskripsi	Tingkat	Catatan Penggunaan
				an memori di luar JVM untuk proses asli seperti RocksDB State Backend). `containerMemoryUtilization` Metrik memberi Anda gambaran yang lebih lengkap dengan memasukkan memori set kerja, yang merupakan pelacak yang lebih baik dari kelelahan memori total. Setelah kelelahan, itu akan menghasilkan pod`Out of Memory Error`. TaskManager

Metrik	Unit	Deskripsi	Tingkat	Catatan Penggunaan	
`containerDiskUtilization`	Persentase	Persentase keseluruhan pemanfaatan disk di seluruh wadah pengelola tugas di cluster aplikasi Flink. Misalnya, jika ada lima pengelola tugas, maka ada lima TaskManager kontainer dan Layanan Terkelola untuk Apache Flink menerbitkan 2 * lima sampel metrik ini per interval pelaporan 1 menit.	Aplikasi	Itu dihitung per kontainer sebagai: Penggunaan disk dalam byte* 100/ Batas Disk untuk wadah dalam byte Untuk kontainer, ini mewakili pemanfaatan sistem file tempat volume root wadah diatur.	

Metrik	Unit	Deskripsi	Tingkat	Catatan Penggunaan	
`currentInputWatermark`	Milidetik	Watermark terakhir yang diterima aplikasi/operator/tugas/ulir	Aplikasi, Operator, Tugas, Paralelisme	Catatan ini hanya dipancarkan untuk dimensi dengan dua input. Ini adalah nilai minimum dari watermark yang terakhir diterima.	
`currentOutputWatermark`	Milidetik	Watermark terakhir yang dipancarkan aplikasi/operator/tugas/ulir	Aplikasi, Operator, Tugas, Paralelisme		

Metrik	Unit	Deskripsi	Tingkat	Catatan Penggunaan	
`downtime`	Milidetik	Untuk tugas yang saat ini dalam situasi gagal/memulihkan, waktu berlalu selama penghentian ini.	Aplikasi	Metrik ini mengukur waktu berlalu saat tugas gagal atau memulihkan. Metrik ini menampilkan 0 untuk tugas yang berjalan dan -1 untuk tugas yang selesai. Jika metrik ini bukan 0 atau -1, ini menunjukkan tugas Apache Flink untuk aplikasi gagal dijalankan.	

Metrik	Unit	Deskripsi	Tingkat	Catatan Penggunaan	
`fullRestarts`	Hitungan	Total berapa kali tugas ini sepenuhnya dimulai kembali sejak dikirimkan. Metrik ini tidak mengukur mulai ulang secara detail.	Aplikasi	Anda dapat menggunakan metrik ini untuk mengevaluasi kesehatan aplikasi umum. Restart dapat terjadi selama pemeliharaan internal oleh Managed Service untuk Apache Flink. Mulai ulang yang lebih tinggi dari biasanya dapat menunjukkan masalah pada aplikasi.	

Metrik	Unit	Deskripsi	Tingkat	Catatan Penggunaan	
`heapMemoryUtilization`	Persentase	Keseluruhan pemanfaatan memori tumpukan di seluruh manajer tugas. Misalnya, jika ada lima pengelola tugas, Managed Service for Apache Flink menerbitkan lima sampel metrik ini per interval pelaporan.	Aplikasi	Anda dapat menggunakan metrik ini untuk memantau penggunaan memori tumpukan minimum, rata-rata, dan maksimum dalam aplikasi Anda. `HeapMemoryUtilization` Satu-satunya akun untuk metrik memori tertentu seperti Heap Memory Usage of TaskManager JVM.	

Metrik	Unit	Deskripsi	Tingkat	Catatan Penggunaan	
`idleTimeMsPerSecond*`	Milidetik	Waktu (dalam milidetik) tugas atau operator ini menganggur (tidak memiliki data untuk diproses) per detik. Waktu idle tidak termasuk waktu bertekanan kembali, jadi jika tugas kembali ditekan, itu tidak menganggur.	Tugas, Operator, Paralelisme	*Tersedia untuk Managed Service untuk aplikasi Apache Flink yang menjalankan Flink versi 1.13 saja. Metrik ini dapat berguna dalam mengidentifikasi kemacetan dalam suatu aplikasi.	

Metrik	Unit	Deskripsi	Tingkat	Catatan Penggunaan	
`lastCheckpointSize`	Byte	Total ukuran titik pemeriksaan terakhir	Aplikasi	Anda dapat menggunakan metrik ini untuk menentukan penggunaan penyimpanan aplikasi yang berjalan. Jika nilai metrik ini meningkat, ini mungkin menunjukkan adanya masalah pada aplikasi Anda, seperti kebocoran memori atau hambatan.	

Metrik	Unit	Deskripsi	Tingkat	Catatan Penggunaan	
`lastCheckpointDuration`	Milidetik	Waktu yang diperlukan untuk menyelesaikan titik pemeriksaan terakhir	Aplikasi	Metrik ini mengukur waktu yang diperlukan untuk menyelesaikan titik pemeriksaan terbaru. Jika nilai metrik ini meningkat, ini mungkin menunjukkan adanya masalah pada aplikasi Anda, seperti kebocoran memori atau hambatan. Dalam beberapa kasus, Anda dapat memecahkan masalah ini dengan menonaktifkan checkpointing.	

Metrik	Unit	Deskripsi	Tingkat	Catatan Penggunaan	
managedMemoryUsed*	Byte	Jumlah memori terkelola yang saat ini digunakan.	Aplikasi, Operator, Tugas, Paralelisme	*Tersedia untuk Managed Service untuk aplikasi Apache Flink yang menjalankan Flink versi 1.13 saja. Ini berkaitan dengan memori yang dikelola oleh Flink di luar tumpukan Java. Ini digunakan untuk backend status RocksDB, dan juga tersedia untuk aplikasi.	

Metrik	Unit	Deskripsi	Tingkat	Catatan Penggunaan	
managedMemoryTotal *	Byte	Jumlah total memori yang dikelola.	Aplikasi, Operator, Tugas, Paralelisme	*Tersedia untuk Managed Service untuk aplikasi Apache Flink yang menjalankan Flink versi 1.13 saja. Ini berkaitan dengan memori yang dikelola oleh Flink di luar tumpukan Java. Ini digunakan untuk backend status RocksDB, dan juga tersedia untuk aplikasi. managedMemoryUtilzations Metrik hanya memperhitungkan metrik memori tertentu seperti Memori Terkelola	

Metrik	Unit	Deskripsi	Tingkat	Catatan Penggunaan	
				(pengguna an memori di luar JVM untuk proses asli seperti RocksDB State Backend)	
`managedMemoryUtilization*`	Persentase	Diturunkan oleh managedMemoryUsed/ managedMemoryTotal	Aplikasi, Operator, Tugas, Paralelisme	*Tersedia untuk Managed Service untuk aplikasi Apache Flink yang menjalankan Flink versi 1.13 saja. Ini berkaitan dengan memori yang dikelola oleh Flink di luar tumpukan Java. Ini digunakan untuk backend status RocksDB, dan juga tersedia untuk aplikasi.	

Metrik	Unit	Deskripsi	Tingkat	Catatan Penggunaan	
`numberOfFailedCheckpoints`	Hitungan	Jumlah kegagalan checkpointing.	Aplikasi	Anda dapat menggunakan metrik ini untuk memantau kesehatan dan kemajuan aplikasi. Titik pemeriksaan mungkin gagal karena masalah aplikasi, seperti throughput atau masalah izin.	

Metrik	Unit	Deskripsi	Tingkat	Catatan Penggunaan
`numRecordsIn*`	Hitungan	Jumlah total catatan yang diterima aplikasi, operator, atau tugas.	Aplikasi, Operator, Tugas, Paralelisme	*Untuk menerapkan statistik SUM selama periode waktu (detik/menit): • Pilih metrik pada Level yang benar. Jika Anda melacak metrik untuk Operator, Anda harus memilih metrik operator yang sesuai. • Karena Layanan Terkelola untuk Apache Flink mengambil 4 snapshot metrik per menit, matematika metrik berikut harus digunakan: m1/4 di mana m1

Metrik	Unit	Deskripsi	Tingkat	Catatan Penggunaan	
				adalah statistik SUM selama periode (detik/menit) Tingkat metrik menentukan apakah metrik ini mengukur jumlah total catatan yang diterima seluruh aplikasi, operator tertentu, atau tugas tertentu.	

Metrik	Unit	Deskripsi	Tingkat	Catatan Penggunaan	
`numRecordsInPerSecond*`	Hitungan/Detik	Jumlah total catatan yang diterima aplikasi, operator, atau tugas per detik.	Aplikasi, Operator, Tugas, Paralelisme	*Untuk menerapkan statistik SUM selama periode waktu (detik/menit): • Pilih metrik pada Level yang benar. Jika Anda melacak metrik untuk Operator, Anda harus memilih metrik operator yang sesuai. • Karena Layanan Terkelola untuk Apache Flink mengambil 4 snapshot metrik per menit, matematika metrik berikut harus digunakan : m1/4 di mana m1	

Metrik	Unit	Deskripsi	Tingkat	Catatan Penggunaan	
				adalah statistik SUM selama periode (detik/menit) Tingkat metrik menentukan apakah metrik ini mengukur jumlah total catatan yang diterima seluruh aplikasi, operator tertentu, atau tugas tertentu per detik.	

Metrik	Unit	Deskripsi	Tingkat	Catatan Penggunaan
`numRecordsOut*`	Hitungan	Jumlah total catatan yang dipancarkan aplikasi, operator, atau tugas.	Aplikasi, Operator, Tugas, Paralelisme	*Untuk menerapkan statistik SUM selama periode waktu (detik/menit): • Pilih metrik pada Level yang benar. Jika Anda melacak metrik untuk Operator, Anda harus memilih metrik operator yang sesuai. • Karena Layanan Terkelola untuk Apache Flink mengambil 4 snapshot metrik per menit, matematika metrik berikut harus digunakan: m1/4 di mana m1

Metrik	Unit	Deskripsi	Tingkat	Catatan Penggunaan	
				adalah statistik SUM selama periode (detik/menit) Tingkat metrik menentukan apakah metrik ini mengukur jumlah total catatan yang dipancark an seluruh aplikasi, operator tertentu, atau tugas tertentu.	

Metrik	Unit	Deskripsi	Tingkat	Catatan Penggunaan	
`numLateRecordsDropped*`	Hitungan	Aplikasi, Operator, Tugas, Paralelisme		*Untuk menerapkan statistik SUM selama periode waktu (detik/menit): • Pilih metrik pada Level yang benar. Jika Anda melacak metrik untuk Operator, Anda harus memilih metrik operator yang sesuai. • Karena Layanan Terkelola untuk Apache Flink mengambil 4 snapshot metrik per menit, matematika metrik berikut harus digunakan : m1/4 di mana m1	

Metrik	Unit	Deskripsi	Tingkat	Catatan Penggunaan	
				adalah statistik SUM selama periode (detik/menit) Jumlah catatan yang dibuang operator atau tugas karena datang terlambat.	

Metrik	Unit	Deskripsi	Tingkat	Catatan Penggunaan	
`numRecordsOutPerSecond*`	Hitungan/Detik	Jumlah total catatan yang dipancark an aplikasi, operator, atau tugas per detik.	Aplikasi, Operator, Tugas, Paralelisme	*Untuk menerapkan statistik SUM selama periode waktu (detik/menit): • Pilih metrik pada Level yang benar. Jika Anda melacak metrik untuk Operator, Anda harus memilih metrik operator yang sesuai. • Karena Layanan Terkelola untuk Apache Flink mengambil 4 snapshot metrik per menit, matematik a metrik berikut harus digunakan : m1/4 di mana m1	

Metrik	Unit	Deskripsi	Tingkat	Catatan Penggunaan	
				adalah statistik SUM selama periode (detik/menit) Tingkat metrik menentukan apakah metrik ini mengukur jumlah total catatan yang dipancark an seluruh aplikasi, operator tertentu, atau tugas tertentu per detik.	
`oldGenera tionGCCou nt`	Hitungan	Jumlah total operasi pengumpulan sampah lama yang terjadi di semua manajer tugas.	Aplikasi		

Metrik	Unit	Deskripsi	Tingkat	Catatan Penggunaan
`oldGenerationGCTime`	Milidetik	Total waktu yang digunakan untuk melakukan operasi pengumpulan sampah lama.	Aplikasi	Anda dapat menggunakan metrik ini untuk memantau jumlah, rata-rata, dan waktu pengumpulan sampah maksimum.
`threadCount`	Hitungan	Jumlah total utas langsung yang digunakan aplikasi.	Aplikasi	Metrik ini mengukur jumlah utas yang digunakan kode aplikasi. Ini tidak sama dengan paralelisme aplikasi.
`uptime`	Milidetik	Waktu ketika tugas berjalan tanpa gangguan.	Aplikasi	Anda dapat menggunakan metrik ini untuk menentukan apakah tugas berhasil berjalan. Metrik ini menampilkan -1 untuk tugas yang selesai.

Metrik Konektor Kinesis Data Streams

AWS memancarkan semua catatan untuk Kinesis Data Streams selain yang berikut ini:

Metrik	Unit	Deskripsi	Tingkat	Catatan Penggunaan
`millisbehindLatest`	Milidetik	Jumlah milidetik konsumen berada di belakang bagian depan aliran, menunjukkan seberapa jauh di belakang waktu konsumen saat ini.	Aplikasi (untuk Stream), Paralelisme (untuk) ShardId	• Nilai 0 menunjukkan bahwa pemrosesan catatan sedang dilakukan, dan tidak ada catatan baru untuk diproses saat ini. Metrik serpihan tertentu dapat ditentukan oleh nama aliran dan id serpihan. • Nilai -1 menunjukkan layanan belum melaporkan nilai untuk metrik.
`bytesRequestedPerFetch`	Byte	Byte yang diminta dalam satu panggilan untuk `getRecords` .	Aplikasi (untuk Stream), Paralelisme (untuk) ShardId	

Metrik Konektor Amazon MSK

AWS memancarkan semua catatan untuk Amazon MSK selain yang berikut ini:

Metrik	Unit	Deskripsi	Tingkat	Catatan Penggunaan
`currentoffsets`	N/A	Offset baca konsumen saat ini, untuk setiap partisi. Metrik partisi tertentu dapat ditentukan berdasarkan nama topik dan id partisi.	Aplikasi (untuk Topik), Paralelisme (untuk) PartitionId	
`commitsFailed`	N/A	Jumlah total kegagalan commit offset ke Kafka, jika commit offset dan checkpointing diaktifkan.	Aplikasi, Operator, Tugas, Paralelisme	Melakukan commit offset kembali ke Kafka hanyalah sarana untuk mengungkapkan kemajuan konsumen, jadi kegagalan commit tidak memengaruhi integritas offset partisi titik pemeriksaan Flink.
`commitsSucceeded`	N/A	Jumlah total keberhasilan commit offset ke Kafka, jika	Aplikasi, Operator, Tugas, Paralelisme	

Metrik	Unit	Deskripsi	Tingkat	Catatan Penggunaan
		commit offset dan checkpoin ting diaktifkan.		
`committed offsets`	N/A	Offset komit yang berhasil terakhir ke Kafka, untuk setiap partisi. Metrik partisi tertentu dapat ditentukan berdasarkan nama topik dan id partisi.	Aplikasi (untuk Topik), Paralelis me (untuk) PartitionId	
`records_l ag_max`	Hitungan	Keterlambatan maksimum dalam hal jumlah catatan untuk setiap partisi di jendela ini	Aplikasi, Operator, Tugas, Paralelisme	
`bytes_con sumed_rate`	Byte	Jumlah rata-rata byte yang digunakan per detik untuk topik	Aplikasi, Operator, Tugas, Paralelisme	

Metrik Apache Zeppelin

Untuk notebook Studio, AWS memancarkan metrik berikut ini pada tingkat aplikasi: `KPUs`, `cpuUtilization`, `heapMemoryUtilization`, `oldGenerationGCTime`, `oldGenerationGCCount`, dan `threadCount`. Selain itu, ini memancarkan metrik yang ditunjukkan dalam tabel berikut, juga pada tingkat aplikasi.

Metrik	Unit	Deskripsi	Nama Prometheus
zeppelinCpuUtilization	Persentase	Persentase keseluruhan pemanfaatan CPU di server Apache Zeppelin.	process_cpu_usage
zeppelinHeapMemoryUtilization	Persentase	Persentase keseluruhan pemanfaatan memori tumpukan untuk server Apache Zeppelin.	jvm_memory_used_bytes
zeppelinThreadCount	Hitungan	Jumlah total utas langsung yang digunakan oleh server Apache Zeppelin.	jvm_threads_live_threads
zeppelinWaitingJobs	Hitungan	Jumlah antrian tugas Apache Zeppelin yang menunggu utas.	jetty_threads_jobs
zeppelinServerUptime	Detik	Total waktu server aktif dan berjalan.	process_uptime_seconds

Melihat CloudWatch Metrik

Anda dapat melihat CloudWatch metrik untuk aplikasi Anda menggunakan CloudWatch konsol Amazon atau. AWS CLI

Untuk melihat metrik menggunakan konsol CloudWatch

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi, pilih Metrik.
3. Di panel CloudWatch Metrik menurut Kategori untuk Layanan Terkelola untuk Apache Flink, pilih kategori metrik.
4. Di panel atas, gulir untuk melihat daftar lengkap metrik.

Untuk melihat metrik menggunakan konsol AWS CLI

- Pada prompt perintah, gunakan perintah berikut.

```
aws cloudwatch list-metrics --namespace "AWS/KinesisAnalytics" --region region
```

Mengatur Tingkat Pelaporan CloudWatch Metrik

Anda dapat mengontrol tingkat metrik aplikasi yang dibuat aplikasi Anda. Layanan Terkelola untuk Apache Flink mendukung tingkat metrik berikut:

- Application (Aplikasi): Aplikasi hanya melaporkan tingkat metrik tertinggi untuk setiap aplikasi. Layanan Terkelola untuk metrik Apache Flink dipublikasikan di tingkat Aplikasi secara default.
- Task (Tugas): Aplikasi ini melaporkan dimensi metrik khusus tugas untuk metrik yang ditentukan dengan tingkat pelaporan metrik Tugas, seperti jumlah catatan masuk dan keluar dari aplikasi per detik.
- Operator: Aplikasi ini melaporkan dimensi metrik khusus operator untuk metrik yang ditentukan dengan tingkat pelaporan metrik Operator, seperti metrik untuk setiap operasi filter atau peta.
- Paralelism (Paralelisme) Aplikasi melaporkan metrik tingkat `Task` dan `Operator` untuk setiap utas eksekusi. Tingkat pelaporan ini tidak disarankan untuk aplikasi dengan pengaturan Paralelisme di atas 64 karena biaya yang berlebihan.

> **Note**
>
> Anda hanya harus menggunakan tingkat metrik ini untuk pemecahan masalah karena jumlah data metrik yang dihasilkan layanan. Anda hanya dapat mengatur tingkat metrik ini menggunakan CLI. Tingkat metrik ini tidak tersedia di konsol.

Tingkat default adalah Application (Aplikasi). Aplikasi melaporkan metrik pada tingkat saat ini dan semua tingkat yang lebih tinggi. Misalnya, jika tingkat pelaporan diatur ke Operator, aplikasi melaporkan metrik Application (Aplikasi), Task (Tugas), dan Operator (Operator).

Anda menetapkan tingkat pelaporan CloudWatch metrik menggunakan `MonitoringConfiguration` parameter `CreateApplication`tindakan, atau `MonitoringConfigurationUpdate` parameter `UpdateApplication`tindakan. Contoh

permintaan UpdateApplicationtindakan berikut ini menetapkan tingkat pelaporan CloudWatch metrik ke Tugas:

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 4,
   "ApplicationConfigurationUpdate": {
      "FlinkApplicationConfigurationUpdate": {
         "MonitoringConfigurationUpdate": {
            "ConfigurationTypeUpdate": "CUSTOM",
            "MetricsLevelUpdate": "TASK"
         }
      }
   }
}
```

Anda juga dapat mengonfigurasi tingkat pencatatan menggunakan parameter `LogLevel` dari tindakan `CreateApplication` atau parameter `LogLevelUpdate` dari tindakan `UpdateApplication`. Anda dapat menggunakan tingkat log berikut:

- `ERROR`: Mencatat peristiwa kesalahan yang berpotensi dapat dipulihkan
- `WARN`: Mencatat peristiwa peringatan yang mungkin menyebabkan kesalahan.
- `INFO`: Mencatat peristiwa informasi.
- `DEBUG`: Mencatat peristiwa debugging umum.

Untuk informasi selengkapnya tentang tingkat pencatatan Log4j, lihat Tingkat Log Kustom di dokumentasi Apache Log4j.

Menggunakan Metrik Kustom dengan Amazon Managed Service untuk Apache Flink

Layanan Terkelola untuk Apache Flink memaparkan 19 metrik CloudWatch, termasuk metrik untuk penggunaan dan throughput sumber daya. Selain itu, Anda dapat membuat metrik sendiri untuk melacak data khusus aplikasi, seperti memproses peristiwa atau mengakses sumber daya eksternal.

Topik ini berisi bagian-bagian berikut:

- Cara Kerjanya
- Contoh-contoh

- Melihat Metrik Kustom

Cara Kerjanya

Metrik kustom dalam Layanan Terkelola untuk Apache Flink menggunakan sistem metrik Apache Flink. Metrik Apache Flink memiliki atribut berikut:

- Type (Tipe): Tipe metrik menjelaskan cara mengukur dan melaporkan data. Tipe metrik Apache Flink yang tersedia termasuk Count, Gauge, Histogram, dan Meter. Untuk informasi selengkapnya tentang tipe metrik Apache Flink, lihat Tipe Metrik.

  > **Note**
  >
  > AWS CloudWatch Metrik tidak mendukung jenis metrik Histogram Apache Flink. CloudWatch hanya dapat menampilkan metrik Apache Flink dari tipe Count, Gauge, dan Meter.

- Lingkup: Ruang lingkup metrik terdiri dari pengenal dan satu set pasangan nilai kunci yang menunjukkan bagaimana metrik akan dilaporkan. CloudWatch Pengidentifikasi metrik terdiri dari hal berikut:
  - Cakupan sistem, yang menunjukkan tingkat tempat metrik dilaporkan (misalnya Operator).
  - Cakupan pengguna, yang menentukan atribut seperti variabel pengguna atau nama grup metrik. Atribut ini didefinisikan menggunakan `MetricGroup.addGroup(key, value)`atau `MetricGroup.addGroup(name)`.

  Untuk informasi selengkapnya tentang cakupan metrik, lihat Cakupan.

Untuk informasi selengkapnya tentang metrik Apache Flink, lihat Metrik di Dokumentasi Apache Flink.

Untuk membuat metrik kustom di Managed Service for Apache Flink, Anda dapat mengakses sistem metrik Apache Flink dari fungsi pengguna apa pun yang diperluas dengan menelepon. `RichFunction` `GetMetricGroup` Metode ini mengembalikan MetricGroupobjek yang dapat Anda gunakan untuk membuat dan mendaftarkan metrik kustom. Layanan Terkelola untuk Apache Flink melaporkan semua metrik yang dibuat dengan kunci grup. `KinesisAnalytics` CloudWatch Metrik kustom yang Anda tentukan memiliki karakteristik sebagai berikut:

- Metrik kustom Anda memiliki nama metrik dan nama grup. Nama ini harus terdiri dari karakter alfanumerik.

- Atribut yang Anda tentukan dalam lingkup pengguna (kecuali untuk grup `KinesisAnalytics` metrik) diterbitkan sebagai CloudWatch dimensi.
- Metrik kustom dipublikasikan di tingkat `Application` secara default.
- Dimensi (Task/ Operator /Paralelism) ditambahkan ke metrik berdasarkan tingkat pemantauan aplikasi. Anda mengatur tingkat pemantauan aplikasi menggunakan MonitoringConfigurationparameter CreateApplicationtindakan, atau MonitoringConfigurationUpdateparameter UpdateApplicationtindakan.

Contoh-contoh

Contoh kode berikut menunjukkan cara membuat kelas pemetaan yang membuat dan menambahkan metrik kustom, dan cara menerapkan kelas pemetaan dalam aplikasi Anda dengan menambahkannya ke objek `DataStream`.

Metrik Kustom Hitungan Catatan

Contoh kode berikut menunjukkan cara membuat kelas pemetaan yang membuat metrik yang menghitung catatan dalam aliran data (fungsionalitas yang sama seperti metrik `numRecordsIn`):

```
    private static class NoOpMapperFunction extends RichMapFunction<String, String> {
        private transient int valueToExpose = 0;
        private final String customMetricName;

        public NoOpMapperFunction(final String customMetricName) {
            this.customMetricName = customMetricName;
        }

        @Override
        public void open(Configuration config) {
            getRuntimeContext().getMetricGroup()
                    .addGroup("KinesisAnalytics")
                    .addGroup("Program", "RecordCountApplication")
                    .addGroup("NoOpMapperFunction")
                    .gauge(customMetricName, (Gauge<Integer>) () -> valueToExpose);
        }

        @Override
        public String map(String value) throws Exception {
            valueToExpose++;
            return value;
        }
```

```
    }
```

Dalam contoh sebelumnya, variabel `valueToExpose` ditingkatkan untuk setiap catatan yang diproses aplikasi.

Setelah menentukan kelas pemetaan, Anda selanjutnya membuat aliran dalam aplikasi yang mengimplementasikan peta:

```
DataStream<String> noopMapperFunctionAfterFilter =
    kinesisProcessed.map(new NoOpMapperFunction("FilteredRecords"));
```

Untuk kode lengkap aplikasi ini, lihat Aplikasi Metrik Kustom Hitungan Catatan.

Metrik Kustom Hitungan Kata

Contoh kode berikut menunjukkan cara membuat kelas pemetaan yang membuat metrik yang menghitung kata dalam aliran data:

```
private static final class Tokenizer extends RichFlatMapFunction<String, Tuple2<String,
 Integer>> {

            private transient Counter counter;

            @Override
            public void open(Configuration config) {
                this.counter = getRuntimeContext().getMetricGroup()
                        .addGroup("KinesisAnalytics")
                        .addGroup("Service", "WordCountApplication")
                        .addGroup("Tokenizer")
                        .counter("TotalWords");
            }

            @Override
            public void flatMap(String value, Collector<Tuple2<String, Integer>>out) {
                // normalize and split the line
                String[] tokens = value.toLowerCase().split("\\W+");

                // emit the pairs
                for (String token : tokens) {
                    if (token.length() > 0) {
                        counter.inc();
                        out.collect(new Tuple2<>(token, 1));
```

```
                    }
                }
            }
        }
```

Dalam contoh sebelumnya, variabel `counter` ditingkatkan untuk setiap kata yang diproses aplikasi.

Setelah menentukan kelas pemetaan, Anda selanjutnya membuat aliran dalam aplikasi yang mengimplementasikan peta:

```
// Split up the lines in pairs (2-tuples) containing: (word,1), and
// group by the tuple field "0" and sum up tuple field "1"
DataStream<Tuple2<String, Integer>> wordCountStream = input.flatMap(new
 Tokenizer()).keyBy(0).sum(1);

// Serialize the tuple to string format, and publish the output to kinesis sink
wordCountStream.map(tuple -> tuple.toString()).addSink(createSinkFromStaticConfig());
```

Untuk kode lengkap aplikasi ini, lihat Aplikasi Metrik Kustom Hitungan Kata.

Melihat Metrik Kustom

Metrik khusus untuk aplikasi Anda muncul di konsol CloudWatch Metrik di AWS/ KinesisAnalyticsdasbor, di bawah grup metrik Aplikasi.

Menggunakan CloudWatch Alarm dengan Amazon Managed Service untuk Apache Flink

Menggunakan alarm CloudWatch metrik Amazon, Anda menonton CloudWatch metrik selama periode waktu yang Anda tentukan. Alarm tersebut melakukan satu atau beberapa tindakan berdasarkan pada nilai metrik atau ekspresi relatif terhadap ambang batas selama beberapa periode waktu. Contoh tindakan mengirim pemberitahuan ke topik Amazon Simple Notification Service (Amazon SNS).

Untuk informasi selengkapnya tentang CloudWatch alarm, lihat Menggunakan CloudWatch Alarm Amazon.

Alarm yang Direkomendasikan

Bagian ini berisi alarm yang direkomendasikan untuk memantau Layanan Terkelola untuk aplikasi Apache Flink.

Tabel menjelaskan alarm yang direkomendasikan dan memiliki kolom berikut:

- Metric Expression (Ekspresi Metrik): Metrik atau ekspresi metrik untuk menguji ambang.
- Statistic (Statistik): Statistik yang digunakan untuk memeriksa metrik—misalnya, Rata-rata.
- Threshold (Ambang): Menggunakan alarm ini mengharuskan Anda menentukan ambang yang menentukan batas performa aplikasi yang diharapkan. Anda perlu menentukan ambang ini dengan memantau aplikasi Anda dalam kondisi normal.
- Description (Deskripsi): Penyebab yang mungkin memicu alarm ini, dan kemungkinan solusi untuk kondisi.

Ekspresi Metrik	Statistik	Ambang	Deskripsi
`waktu henti > 0`	Average	0	A downtime greater than zero indicates that the applicati on has failed. If the value is larger than 0, the application is not processing any data. Recommended for all applications. The `Waktu henti` metric measures the duration of an outage. A downtime greater than zero indicates that the application has failed. For troubleshooting, see Aplikasi Dimulai Ulang.
`TARIF (numberOf FailedPos pemeriksaan) > 0`	Average	0	This metric counts the number of failed checkpoints since the application started.

Ekspresi Metrik	Statistik	Ambang	Deskripsi
			Depending on the application, it can be tolerable if checkpoints fail occasionally. But if checkpoints are regularly failing, the application is likely unhealthy and needs further attention. We recommend monitoring RATE(numberOfFailedCheckpoints) to alarm on the gradient and not on absolute values. Recommended for all applications. Use this metric to monitor application health and checkpointing progress. The application saves state data to checkpoints when it's healthy. Checkpointing can fail due to timeouts if the application isn't making progress in processing the input data. For troubleshooting, see Waktu titik checkpointing.

Ekspresi Metrik	Statistik	Ambang	Deskripsi
`Operator.numRecordsOutPerSecond` < ambang	Average	The minimum number of records emitted from the application during normal conditions.	Recommended for all applications. Falling below this threshold can indicate that the application isn't making expected progress on the input data. For troubleshooting, see Throughput Terlalu Lambat.

Ekspresi Metrik	Statistik	Ambang	Deskripsi
`Records_lag_max|MillisBehindLatest` > ambang	Maximum	The maximum expected latency during normal conditions.	If the application is consuming from Kinesis or Kafka, these metrics indicate if the application is falling behind and needs to be scaled in order to keep up with the current load. This is a good generic metric that is easy to track for all kinds of applications. But it can only be used for reactive scaling, i.e., when the application has already fallen behind. Recommended for all applications. Use the `records_lag_max` metric for a Kafka source, or the `MillisBehindTerbaru` for a Kinesis stream source. Rising above this threshold can indicate that the application isn't making expected progress on the input data. For troubleshooting, see

Ekspresi Metrik	Statistik	Ambang	Deskripsi
			Throughput Terlalu Lambat.

Ekspresi Metrik	Statistik	Ambang	Deskripsi
`lastCheckpointDuration` > ambang	Maximum	The maximum expected checkpoint duration during normal conditions.	Monitors how much data is stored in state and how long it takes to take a checkpoint. If checkpoints grow or take long, the application is continuously spending time on checkpointing and has less cycles for actual processing. At some points, checkpoints may grow too large or take so long that they fail. In addition to monitoring absolute values, customers should also considering monitoring the change rate with `TINGKAT(lastCheckpointSize)` and `TINGKAT(lastCheckpointDuration)` . If the `lastCheckpointDuration` continuously increases, rising above this threshold can indicate that the application isn't

Ekspresi Metrik	Statistik	Ambang	Deskripsi
			making expected progress on the input data, or that there are problems with application health such as backpressure. For troubleshooting, see Pertumbuhan Negara Tak Terbatas.

Ekspresi Metrik	Statistik	Ambang	Deskripsi
`lastCheckpointSize > ambang`	Maximum	The maximum expected checkpoint size during normal conditions.	Monitors how much data is stored in state and how long it takes to take a checkpoint. If checkpoints grow or take long, the application is continuously spending time on checkpointing and has less cycles for actual processing. At some points, checkpoints may grow too large or take so long that they fail. In addition to monitoring absolute values, customers should also considering monitoring the change rate with `TINGKAT(lastCheckpointSize)` and `TINGKAT(lastCheckpointDuration)` . If the `lastCheckpointSize` continuously increases, rising above this threshold can indicate that the application is

Ekspresi Metrik	Statistik	Ambang	Deskripsi
			accumulating state data. If the state data becomes too large, the application can run out of memory when recovering from a checkpoint, or recovering from a checkpoint might take too long. For troubleshooting, see Pertumbuhan Negara Tak Terbatas.
`heapMemoryUtilization` > ambang	Maximum	This gives a good indication of the overall resource utilization of the application and can be used for proactive scaling unless the application is I/O bound. The maximum expected `heapMemoryUtilization` size during normal conditions, with a recommended value of 90 percent.	You can use this metric to monitor the maximum memory utilization of task managers across the application. If the application reaches this threshold, you need to provision more resources . You do this by enabling automatic scaling or increasing the application parallelism. For more information about increasing resources, see Penskalaan.

Ekspresi Metrik	Statistik	Ambang	Deskripsi
`cpuUtilization` > ambang	Maximum	This gives a good indication of the overall resource utilization of the application and can be used for proactive scaling unless the application is I/O bound. The maximum expected `cpuUtilization` size during normal conditions, with a recommended value of 80 percent.	You can use this metric to monitor the maximum CPU utilization of task managers across the application. If the application reaches this threshold, you need to provision more resources You do this by enabling automatic scaling or increasing the application parallelism. For more information about increasing resources, see Penskalaan.
`ThreadsCount` > ambang	Maximum	The maximum expected `ThreadsCount` size during normal conditions.	You can use this metric to watch for thread leaks in task managers across the application. If this metric reaches this threshold, check your application code for threads being created without being closed.

Ekspresi Metrik	Statistik	Ambang	Deskripsi
`(oldGarbageCollectionWaktu* 100) /60_000 selama 1 menit periode ')` > ambang	Maximum	The maximum expected `oldGarbageCollectionWaktu` duration. We recommend setting a threshold such that typical garbage collection time is 60 percent of the specified threshold, but the correct threshold for your application will vary.	If this metric is continually increasing, this can indicate that there is a memory leak in task managers across the application.
`TINGKAT (oldGarbageCollectionHitung)` > ambang	Maximum	The maximum expected `oldGarbageCollectionHitung` under normal conditions. The correct threshold for your application will vary.	If this metric is continually increasing, this can indicate that there is a memory leak in task managers across the application.
`Operator.currentOutputWatermark - Operator.currentInputWatermark` > ambang	Minimum	The minimum expected watermark increment under normal conditions. The correct threshold for your application will vary.	If this metric is continually increasing, this can indicate that either the application is processing increasingly older events, or that an upstream subtask has not sent a watermark in an increasingly long time.

Menulis Pesan Kustom ke CloudWatch Log

Anda dapat menulis pesan khusus ke Layanan Terkelola untuk log aplikasi Apache Flink. CloudWatch Anda melakukannya menggunakan pustaka `log4j` Apache atau pustaka `Simple Logging Facade for Java (SLF4J)`.

Topik

Menulis ke CloudWatch Log Menggunakan Log4J

1. Tambahkan dependensi berikut ke file `pom.xml` aplikasi Anda:

```
<dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-api</artifactId>
    <version>2.6.1</version>
</dependency>
<dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-core</artifactId>
    <version>2.6.1</version>
</dependency>
```

2. Sertakan objek dari pustaka:

```
import org.apache.logging.log4j.Logger;
```

3. Beri contoh objek `Logger`, yang meneruskan di kelas aplikasi Anda:

```
private static final Logger log =
 LogManager.getLogger.getLogger(YourApplicationClass.class);
```

4. Tulis ke log menggunakan `log.info`. Sejumlah besar pesan ditulis ke log aplikasi. Untuk membuat pesan kustom Anda lebih mudah difilter, gunakan tingkat log aplikasi `INFO`.

```
log.info("This message will be written to the application's CloudWatch log");
```

Aplikasi ini menulis catatan ke log dengan pesan yang serupa dengan berikut ini:

```
{
  "locationInformation": "com.amazonaws.services.managed-
flink.StreamingJob.main(StreamingJob.java:95)",
  "logger": "com.amazonaws.services.managed-flink.StreamingJob",
  "message": "This message will be written to the application's CloudWatch log",
  "threadName": "Flink-DispatcherRestEndpoint-thread-2",
  "applicationARN": "arn:aws:kinesisanalyticsus-east-1:123456789012:application/test",
  "applicationVersionId": "1", "messageSchemaVersion": "1",
  "messageType": "INFO"
}
```

Menulis ke CloudWatch Log Menggunakan SLF4J

1. Tambahkan dependensi berikut ke file `pom.xml` aplikasi Anda:

   ```
   <dependency>
       <groupId>org.slf4j</groupId>
       <artifactId>slf4j-log4j12</artifactId>
       <version>1.7.7</version>
       <scope>runtime</scope>
   </dependency>
   ```

2. Sertakan objek dari pustaka:

   ```
   import org.slf4j.Logger;
   import org.slf4j.LoggerFactory;
   ```

3. Beri contoh objek `Logger`, yang meneruskan di kelas aplikasi Anda:

   ```
   private static final Logger log =
    LoggerFactory.getLogger(YourApplicationClass.class);
   ```

4. Tulis ke log menggunakan `log.info`. Sejumlah besar pesan ditulis ke log aplikasi. Untuk membuat pesan kustom Anda lebih mudah difilter, gunakan tingkat log aplikasi `INFO`.

   ```
   log.info("This message will be written to the application's CloudWatch log");
   ```

Aplikasi ini menulis catatan ke log dengan pesan yang serupa dengan berikut ini:

```
{
  "locationInformation": "com.amazonaws.services.managed-
flink.StreamingJob.main(StreamingJob.java:95)",
  "logger": "com.amazonaws.services.managed-flink.StreamingJob",
  "message": "This message will be written to the application's CloudWatch log",
  "threadName": "Flink-DispatcherRestEndpoint-thread-2",
  "applicationARN": "arn:aws:kinesisanalyticsus-east-1:123456789012:application/test",
  "applicationVersionId": "1", "messageSchemaVersion": "1",
  "messageType": "INFO"
}
```

Logging Layanan Terkelola untuk Panggilan API Apache Flink dengan AWS CloudTrail

Managed Service for Apache Flink terintegrasi denganAWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau AWS layanan di Managed Service untuk Apache Flink. CloudTrail menangkap semua panggilan API untuk Managed Service untuk Apache Flink sebagai event. Panggilan yang diambil termasuk panggilan dari Layanan Terkelola untuk konsol Apache Flink dan panggilan kode ke Layanan Terkelola untuk operasi API Apache Flink. Jika Anda membuat jejak, Anda dapat mengaktifkan pengiriman CloudTrail acara secara terus menerus ke bucket Amazon S3, termasuk peristiwa untuk Layanan Terkelola untuk Apache Flink. Jika Anda tidak mengonfigurasi jejak, Anda masih dapat melihat peristiwa terbaru di CloudTrail konsol dalam Riwayat acara. Dengan menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan permintaan yang dibuat untuk Layanan Terkelola untuk Apache Flink, alamat IP dari mana permintaan dibuat, siapa yang membuat permintaan, kapan dibuat, dan detail tambahan.

Untuk mempelajari selengkapnya CloudTrail, lihat Panduan AWS CloudTrail Pengguna.

Layanan Terkelola untuk Informasi Apache Flink di CloudTrail

CloudTrail diaktifkan di AWS akun Anda saat Anda membuat akun. Ketika aktivitas terjadi di Layanan Terkelola untuk Apache Flink, aktivitas tersebut direkam dalam suatu CloudTrail peristiwa bersama dengan peristiwa AWS layanan lainnya dalam riwayat Acara. Anda dapat melihat, mencari, dan mengunduh peristiwa terbaru di akun AWS. Untuk informasi selengkapnya, lihat Melihat Acara dengan Riwayat CloudTrail Acara.

Untuk catatan peristiwa yang sedang berlangsung di AWS akun Anda, termasuk acara untuk Layanan Terkelola untuk Apache Flink, buat jejak. Jejak memungkinkan CloudTrail untuk

mengirimkan file log ke bucket Amazon S3. Secara default, saat Anda membuat jejak di konsol tersebut, jejak diterapkan ke semua Wilayah AWS. Jejak mencatat peristiwa dari semua Wilayah di partisi AWS dan mengirimkan file log ke bucket Amazon S3 yang Anda tentukan. Selain itu, Anda dapat mengonfigurasi AWS layanan lain untuk menganalisis lebih lanjut dan menindaklanjuti data peristiwa yang dikumpulkan dalam CloudTrail log. Untuk informasi selengkapnya, lihat berikut:

- Gambaran umum untuk Membuat Jejak
- CloudTrail Layanan dan Integrasi yang Didukung
- Mengkonfigurasi Notifikasi Amazon SNS untuk CloudTrail
- Menerima File CloudTrail Log dari Beberapa Wilayah dan Menerima File CloudTrail Log dari Beberapa Akun

Semua Layanan Terkelola untuk tindakan Apache Flink dicatat oleh CloudTrail dan didokumentasikan dalam referensi API Managed Service for Apache Flink. Misalnya, panggilan ke `CreateApplication` dan `UpdateApplication` tindakan menghasilkan entri dalam file CloudTrail log.

Setiap peristiwa atau entri log berisi informasi tentang siapa yang membuat permintaan tersebut. Informasi identitas membantu Anda menentukan berikut ini:

- Bahwa permintaan tersebut dibuat dengan kredensial pengguna root atau pengguna (IAM) AWS Identity and Access Management.
- Baik permintaan tersebut dibuat dengan kredensial keamanan sementara untuk peran atau pengguna gabungan.
- Apakah permintaan dibuat oleh layanan AWS lain.

Untuk informasi lain, lihat Elemen userIdentity CloudTrail .

Memahami Layanan Terkelola untuk Entri File Log Apache Flink

Trail adalah konfigurasi yang memungkinkan pengiriman peristiwa sebagai file log ke bucket Amazon S3 yang Anda tentukan. CloudTrail file log berisi satu atau lebih entri log. Peristiwa mewakili permintaan tunggal dari sumber manapun dan mencakup informasi tentang tindakan yang diminta, tanggal dan waktu tindakan, parameter permintaan, dan sebagainya. CloudTrail file log bukanlah jejak tumpukan yang diurutkan dari panggilan API publik, sehingga file tersebut tidak muncul dalam urutan tertentu.

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan AddApplicationCloudWatchLoggingOptiondan DescribeApplicationtindakan.


```
{
    "Records": [
        {
            "eventVersion": "1.05",
            "userIdentity": {
                "type": "IAMUser",
                "principalId": "EX_PRINCIPAL_ID",
                "arn": "arn:aws:iam::012345678910:user/Alice",
                "accountId": "012345678910",
                "accessKeyId": "EXAMPLE_KEY_ID",
                "userName": "Alice"
            },
            "eventTime": "2019-03-07T01:19:47Z",
            "eventSource": "kinesisanlaytics.amazonaws.com",
            "eventName": "AddApplicationCloudWatchLoggingOption",
            "awsRegion": "us-east-1",
            "sourceIPAddress": "127.0.0.1",
            "userAgent": "aws-sdk-java/unknown-version Linux/x.xx",
            "requestParameters": {
                "applicationName": "cloudtrail-test",
                "currentApplicationVersionId": 1,
                "cloudWatchLoggingOption": {
                    "logStreamARN": "arn:aws:logs:us-east-1:012345678910:log-
group:cloudtrail-test:log-stream:flink-cloudwatch"
                }
            },
            "responseElements": {
                "cloudWatchLoggingOptionDescriptions": [
                    {
                        "cloudWatchLoggingOptionId": "2.1",
                        "logStreamARN": "arn:aws:logs:us-east-1:012345678910:log-
group:cloudtrail-test:log-stream:flink-cloudwatch"
                    }
                ],
                "applicationVersionId": 2,
                "applicationARN": "arn:aws:kinesisanalyticsus-
east-1:012345678910:application/cloudtrail-test"
            },
            "requestID": "18dfb315-4077-11e9-afd3-67f7af21e34f",
            "eventID": "d3c9e467-db1d-4cab-a628-c21258385124",
```

```
            "eventType": "AwsApiCall",
            "apiVersion": "2018-05-23",
            "recipientAccountId": "012345678910"
        },
        {
            "eventVersion": "1.05",
            "userIdentity": {
                "type": "IAMUser",
                "principalId": "EX_PRINCIPAL_ID",
                "arn": "arn:aws:iam::012345678910:user/Alice",
                "accountId": "012345678910",
                "accessKeyId": "EXAMPLE_KEY_ID",
                "userName": "Alice"
            },
            "eventTime": "2019-03-12T02:40:48Z",
            "eventSource": "kinesisanlaytics.amazonaws.com",
            "eventName": "DescribeApplication",
            "awsRegion": "us-east-1",
            "sourceIPAddress": "127.0.0.1",
            "userAgent": "aws-sdk-java/unknown-version Linux/x.xx",
            "requestParameters": {
                "applicationName": "sample-app"
            },
            "responseElements": null,
            "requestID": "3e82dc3e-4470-11e9-9d01-e789c4e9a3ca",
            "eventID": "90ffe8e4-9e47-48c9-84e1-4f2d427d98a5",
            "eventType": "AwsApiCall",
            "apiVersion": "2018-05-23",
            "recipientAccountId": "012345678910"
        }
    ]
}
```

Tuning Kinerja di Amazon Managed Service untuk Apache Flink

Topik ini menjelaskan teknik untuk memantau dan meningkatkan kinerja Layanan Terkelola Anda untuk aplikasi Apache Flink.

Topik

Memecahkan masalah performa

Bagian ini berisi daftar gejala yang dapat Anda periksa untuk mendiagnosis dan memperbaiki masalah performa.

Jika sumber data Anda adalah aliran Kinesis, masalah performa biasanya muncul sebagai metrik `millisbehindLatest` yang tinggi atau meningkat. Untuk sumber lainnya, Anda dapat memeriksa metrik serupa yang mewakili jeda dalam membaca dari sumbernya.

Jalur Data

Saat menyelidiki masalah performa dengan aplikasi Anda, pertimbangkan seluruh jalur yang dilalui data Anda. Komponen aplikasi berikut dapat menjadi hambatan performa dan membuat tekanan balik jika komponen tidak didesain atau ditetapkan dengan benar:

- Sumber data dan tujuan: Pastikan sumber daya eksternal yang berinteraksi dengan aplikasi Anda adalah properti yang disediakan untuk throughput yang akan dialami aplikasi Anda.
- Data status: Pastikan aplikasi Anda tidak terlalu sering berinteraksi dengan penyimpanan status.

  Anda dapat mengoptimalkan serializer yang digunakan aplikasi Anda. Serializer Kryo default dapat menangani tipe serialisasi apa pun, tetapi Anda dapat menggunakan serializer yang lebih berperforma jika aplikasi Anda hanya menyimpan data dalam tipe POJO. Untuk informasi tentang serializer Apache Flink, lihat Tipe Data & Serialisasi di Dokumentasi Apache Flink.

- Operator: Pastikan logika bisnis yang diterapkan oleh operator Anda tidak terlalu rumit, atau Anda tidak membuat atau menggunakan sumber daya dengan setiap catatan yang diproses. Juga pastikan aplikasi Anda tidak terlalu sering membuat jendela geser atau tumbling.

Solusi Pemecahan Masalah Performa

Bagian ini berisi solusi potensial untuk masalah performa.

Topik

CloudWatchTingkat Pemantauan

Verifikasi bahwaCloudWatchTingkat Pemantauan tidak diatur ke pengaturan yang terlalu bertele-tele.

Pengaturan Tingkat Log Pemantauan `Debug` menghasilkan sejumlah besar lalu lintas, yang dapat membuat tekanan balik. Anda hanya harus menggunakannya saat secara aktif menyelidiki masalah dengan aplikasi.

Jika aplikasi Anda memiliki pengaturan `Parallelism` tinggi, menggunakan Tingkat Metrik Pemantauan `Parallelism` juga akan menghasilkan sejumlah besar lalu lintas yang dapat menyebabkan tekanan balik. Gunakan hanya tingkat metrik ini saat `Parallelism` untuk aplikasi Anda rendah, atau saat menyelidiki masalah dengan aplikasi.

Untuk informasi selengkapnya, lihat TingkatPemantauan Aplikasi.

Metrik CPU Aplikasi

Periksa metrik `CPU` aplikasi. Jika metrik ini di atas 75 persen, Anda dapat mengizinkan aplikasi mengalokasikan lebih banyak sumber daya untuk dirinya sendiri dengan mengaktifkan penskalaan otomatis.

Jika penskalaan otomatis diaktifkan, aplikasi mengalokasikan lebih banyak sumber daya jika penggunaan CPU lebih dari 75 persen selama 15 menit. Untuk informasi tentang penskalaan, lihat bagian Mengelola penskalaan dengan benar berikut, dan Penskalaan.

> **Note**
>
> Aplikasi hanya akan menskalakan secara otomatis saat merespons penggunaan CPU. Aplikasi tidak akan menskalakan otomatis saat merespons metrik sistem lain, seperti `heapMemoryUtilization`. Jika aplikasi Anda memiliki tingkat penggunaan tinggi untuk metrik lain, tingkatkan paralelisme aplikasi Anda secara manual.

Paralelisme Aplikasi

Tingkatkan paralelisme aplikasi. Anda memperbarui paralelisme aplikasi menggunakan`ParallelismConfigurationUpdate`parameter dari`UpdateApplication`tindakan.

KPU maksimum untuk aplikasi adalah 64 secara default, dan dapat ditingkatkan dengan meminta peningkatan batas.

Ini juga penting untuk menetapkan paralelisme ke setiap operator berdasarkan beban kerjanya, bukan hanya meningkatkan paralelisme aplikasi itu sendiri. Lihat Paralelisme Operator berikut.

Pencatatan Aplikasi

Periksa apakah aplikasi mencatat entri untuk setiap catatan yang diproses. Menulis entri log untuk setiap catatan pada saat aplikasi memiliki throughput yang tinggi akan menyebabkan hambatan parah dalam pemrosesan data. Untuk memeriksa kondisi ini, kueri log Anda untuk entri log yang ditulis aplikasi Anda dengan setiap catatan yang diproses. Untuk informasi selengkapnya tentang membaca log aplikasi, lihat the section called “Menganalisis Log”.

Paralelisme Operator

Pastikan beban kerja aplikasi Anda didistribusikan secara merata di antara proses pekerja.

Untuk informasi tentang menyetel beban kerja operator aplikasi Anda, lihat Penskalaan operator.

Logika Aplikasi

Periksa logika aplikasi Anda untuk operasi yang tidak efisien atau yang tidak berfungsi, seperti mengakses dependensi eksternal (seperti basis data atau layanan web), mengakses status

aplikasi, dll. Dependensi eksternal juga dapat menghambat performa jika tidak berfungsi atau tidak dapat diakses dengan andal, yang dapat menyebabkan dependensi eksternal yang menampilkan kesalahan `HTTP 500`.

Jika aplikasi Anda menggunakan dependensi eksternal untuk memperkaya atau memproses data yang masuk, pertimbangkan untuk menggunakan IO asinkron sebagai gantinya. Untuk informasi selengkapnya, lihat I/O Asinkron di Dokumentasi Apache Flink.

Memori aplikasi

Periksa aplikasi Anda untuk kebocoran sumber daya. Jika aplikasi Anda tidak membuang utas atau memori dengan benar, Anda mungkin melihat metrik `millisbehindLatest`, `CheckpointSize`, dan `CheckpointDuration` yang meningkat atau meningkat secara bertahap. Kondisi ini juga dapat menyebabkan kegagalan manajer tugas atau manajer pekerjaan.

Praktik Terbaik Performa

Bagian ini menjelaskan pertimbangan khusus guna mendesain aplikasi untuk performa.

Mengelola penskalaan dengan benar

Bagian ini berisi informasi tentang mengelola penskalaan tingkat aplikasi dan tingkat operator.

Bagian ini berisi topik berikut:

- Kelola penskalaan aplikasi dengan benar
- Kelola penskalaan operator dengan benar

Kelola penskalaan aplikasi dengan benar

Anda dapat menggunakan penskalaan otomatis untuk menangani lonjakan tidak terduga dalam aktivitas aplikasi. KPU aplikasi Anda akan meningkat secara otomatis jika kriteria berikut terpenuhi:

- Penskalaan otomatis diaktifkan untuk aplikasi.
- Penggunaan CPU tetap di atas 75 persen selama 15 menit.

Jika penskalaan otomatis diaktifkan, tetapi penggunaan CPU tidak berada di ambang ini, aplikasi tidak akan meningkatkan skala CPU. Jika Anda mengalami lonjakan penggunaan CPU yang tidak memenuhi ambang ini, atau lonjakan metrik penggunaan yang berbeda seperti

`heapMemoryUtilization`, tingkatkan penskalaan secara manual agar aplikasi Anda dapat menangani lonjakan aktivitas.

> **Note**
>
> Jika aplikasi secara otomatis menambahkan lebih banyak sumber daya melalui penskalaan otomatis, aplikasi akan merilis sumber daya baru setelah periode tidak aktif. Penurunan skala sumber daya akan memengaruhi performa untuk sementara.

Untuk informasi selengkapnya tentang penskalaan, lihat Penskalaan.

Kelola penskalaan operator dengan benar

Anda dapat meningkatkan performa aplikasi Anda dengan memastikan beban kerja aplikasi Anda didistribusikan secara merata di antara proses pekerja, dan operator dalam aplikasi Anda memiliki sumber daya sistem yang mereka perlukan agar stabil dan berfungsi.

Anda dapat mengatur paralelisme untuk setiap operator dalam kode aplikasi Anda menggunakan pengaturan `parallelism`. Jika Anda tidak mengatur paralelisme untuk operator, pengaturan paralelisme tingkat aplikasi akan digunakan. Operator yang menggunakan pengaturan paralelisme tingkat aplikasi berpotensi menggunakan semua sumber daya sistem yang tersedia untuk aplikasi, membuat aplikasi tidak stabil.

Untuk menentukan paralelisme terbaik untuk setiap operator, pertimbangkan persyaratan sumber daya relatif operator yang dibandingkan dengan operator lain dalam aplikasi. Atur operator yang lebih intensif sumber daya untuk pengaturan paralelisme operator yang lebih tinggi dari operator yang kurang intensif sumber daya.

Total paralelisme operator untuk aplikasi adalah jumlah paralelisme untuk semua operator dalam aplikasi. Anda menyetel total paralelisme operator untuk aplikasi Anda dengan menentukan rasio terbaik di antaranya dan total slot tugas yang tersedia untuk aplikasi Anda. Rasio stabil khas dari total paralelisme operator untuk slot tugas adalah 4:1, yaitu, aplikasi memiliki satu slot tugas yang tersedia untuk setiap empat subtugas operator yang tersedia. Aplikasi dengan operator yang lebih intensif sumber daya mungkin memerlukan rasio 3:1 atau 2:1, sementara aplikasi dengan operator yang kurang intensif sumber daya mungkin stabil dengan rasio 10:1.

Anda dapat mengatur rasio untuk operator menggunakan Properti Runtime, sehingga Anda dapat menyetel paralelisme operator tanpa menyusun dan mengunggah kode aplikasi Anda.

Contoh kode berikut mendemonstrasikan cara mengatur paralelisme operator sebagai rasio yang dapat disetel dari paralelisme aplikasi saat ini:

```
Map<String, Properties> applicationProperties =
 KinesisAnalyticsRuntime.getApplicationProperties();
operatorParallelism =
    StreamExecutionEnvironment.getParallelism() /
    Integer.getInteger(

 applicationProperties.get("OperatorProperties").getProperty("MyOperatorParallelismRatio")
    );
```

Untuk informasi tentang subtugas, slot tugas, dan sumber daya aplikasi lainnya, lihat Sumber Daya Aplikasi.

Untuk mengontrol distribusi beban kerja di seluruh proses pekerja aplikasi Anda, gunakan pengaturan `Parallelism` dan metode partisi `KeyBy`. Untuk informasi selengkapnya, lihat topik berikut di Dokumentasi Apache Flink:

- Eksekusi Paralel
- DataStreamTransformasi

Pantau penggunaan sumber daya dependensi eksternal

Jika ada hambatan kinerja di suatu tujuan (seperti Kinesis Streams, Kinesis Data Firehose, DynamoDB atauOpenSearchLayanan), aplikasi Anda akan mengalami tekanan balik. Pastikan dependensi eksternal Anda disediakan dengan benar untuk throughput aplikasi Anda.

> **Note**
>
> Kegagalan dalam layanan lainnya dapat menyebabkan kegagalan dalam aplikasi Anda. Jika Anda melihat kegagalan dalam aplikasi Anda, periksaCloudWatchlog untuk layanan tujuan Anda untuk kegagalan.

Jalankan aplikasi Apache Flink Anda secara lokal

Untuk memecahkan masalah memori, Anda dapat menjalankan aplikasi Anda dalam instalasi Flink lokal. Ini akan memberi Anda akses ke alat debugging seperti stack trace dan heap dump yang tidak tersedia saat menjalankan aplikasi Anda di Managed Service for Apache Flink.

Untuk informasi tentang membuat instalasi Flink lokal, lihat Penyiapan Tutorial di Dokumentasi Apache Flink.

Memantau Performa

Bagian ini menjelaskan alat untuk memantau performa aplikasi.

Pemantauan Kinerja menggunakanCloudWatchMetrik

Anda memantau penggunaan sumber daya aplikasi, throughput, checkpointing, dan downtime menggunakanCloudWatchmetrik. Untuk informasi tentang penggunaanCloudWatchmetrik dengan Layanan Terkelola untuk aplikasi Apache Flink, lihatMetrik dan Dimensi dalam Layanan Terkelola untuk Apache Flink.

Pemantauan Kinerja menggunakanCloudWatchLog dan Alarm

Anda memantau kondisi kesalahan yang berpotensi menyebabkan masalah kinerja menggunakanCloudWatchLog.

Kondisi kesalahan muncul di entri log sebagai perubahan status pekerjaan Apache Flink dari status `RUNNING` ke status `FAILED`.

Anda menggunakanCloudWatchalarm untuk membuat notifikasi untuk masalah kinerja, seperti penggunaan sumber daya atau metrik pos pemeriksaan di atas ambang batas aman, atau perubahan status aplikasi yang tidak terduga.

Untuk informasi tentang membuatCloudWatchalarm untuk Layanan Terkelola untuk aplikasi Apache Flink, lihatAlarm.

Layanan Terkelola untuk Kuota Apache Flink dan Studio Notebook

Saat bekerja dengan Amazon Managed Service untuk Apache Flink, perhatikan kuota berikut:

- Anda dapat membuat hingga 50 Layanan Terkelola untuk aplikasi Apache Flink per Wilayah di akun Anda. Anda dapat membuat kasus untuk meminta aplikasi tambahan melalui formulir peningkatan kuota layanan. Untuk informasi selengkapnya, lihat Pusat AWS Support.

  Untuk daftar Wilayah yang mendukung Layanan Terkelola untuk Apache Flink, lihatLayanan Terkelola untuk Wilayah dan Titik Akhir Apache Flink.

- Jumlah unit pengolahan Kinesis (KPU) dibatasi hingga 64 secara default. Untuk petunjuk tentang cara meminta peningkatan kuota ini, lihat Untuk meminta peningkatan kuota di Service Quotas. Pastikan Anda menentukan awalan aplikasi yang perlu diterapkan batas KPU baru.

  Dengan Managed Service untuk Apache Flink, AndaAWSakun dikenakan biaya untuk sumber daya yang dialokasikan, bukan sumber daya yang digunakan aplikasi Anda. Anda akan ditagih tarif per jam berdasarkan jumlah maksimum KPU yang digunakan untuk menjalankan aplikasi pemrosesan aliran Anda. Satu KPU memberi Anda 1 vCPU dan memori 4 GiB. Untuk setiap KPU, layanan ini juga menyediakan 50 GiB penyimpanan aplikasi yang berjalan

- Anda dapat membuat hingga 1.000 Managed Service untuk Apache FlinkSnapshotper aplikasi.

- Anda dapat menetapkan hingga 50 tanda per aplikasi.

- Ukuran maksimum untuk file JAR aplikasi adalah 512 MiB. Jika melebihi kuota ini, aplikasi Anda akan gagal dimulai.

Untuk Studio notebook, kuota berikut berlaku. Untuk meminta kuota yang lebih tinggi, buat kasus dukungan.

- `websocketMessageSize` = 5 MiB
- `noteSize` = 5 MiB
- `noteCount`= 1000
- `Max cumulative UDF size`= 100 MiB
- `Max cumulative dependency jar size`= 300 MiB

Layanan Terkelola untuk Pemeliharaan Apache Flink

Layanan Terkelola untuk Apache Flink menambal aplikasi Anda secara berkala dengan pembaruan keamanan sistem operasi dan gambar kontainer untuk menjaga kepatuhan dan memenuhi tujuan keamanan. AWS Tabel berikut mencantumkan jendela waktu default selama Managed Service for Apache Flink melakukan jenis pemeliharaan ini. Pemeliharaan untuk aplikasi Anda mungkin terjadi kapan saja selama jendela waktu yang sesuai dengan Wilayah Anda. Aplikasi Anda mungkin mengalami waktu henti selama 10 hingga 30 detik selama proses pemeliharaan ini. Namun, durasi waktu henti sebenarnya bergantung pada status aplikasi. Untuk informasi tentang cara meminimalkan dampak waktu henti ini, lihat the section called “Toleransi kesalahan: titik pemeriksaan dan titik simpan”.

Untuk mengubah jendela waktu di mana Managed Service for Apache Flink melakukan pemeliharaan pada aplikasi Anda, gunakan API. UpdateApplicationMaintenanceConfiguration

Wilayah	Jendela waktu pemeliharaan
AWS GovCloud (AS-Barat)	06:00–14:00 UTC
AWS GovCloud (AS-Timur)	03:00–11:00 UTC
US East (N. Virginia)	03:00–11:00 UTC
US East (Ohio)	03:00–11:00 UTC
US West (N. California)	06:00–14:00 UTC
US West (Oregon)	06:00–14:00 UTC
Asia Pacific (Hong Kong)	13:00–21:00 UTC
Asia Pacific (Mumbai)	16:30–00:30 UTC
Asia Pacific (Hyderabad)	16:30–00:30 UTC
Asia Pacific (Seoul)	13:00–21:00 UTC
Asia Pacific (Singapore)	14:00–22:00 UTC

Wilayah	Jendela waktu pemeliharaan
Asia Pacific (Sydney)	12:00–20:00 UTC
Asia Pacific (Jakarta)	15:00 — 23:00 UTC
Asia Pacific (Tokyo)	13:00–21:00 UTC
Canada (Central)	03:00–11:00 UTC
China (Beijing)	13:00–21:00 UTC
China (Ningxia)	13:00–21:00 UTC
Europe (Frankfurt)	06:00–14:00 UTC
Europe (Zurich)	20.00–04:00 UTC
Europe (Ireland)	22:00–06:00 UTC
Europe (London)	22:00–06:00 UTC
Europe (Stockholm)	23:00–07:00 UTC
Europe (Milan)	21:00–05:00 UTC
Europe (Spain)	21:00–05:00 UTC
Africa (Cape Town)	20.00–04:00 UTC
Europe (Ireland)	22:00–06:00 UTC
Europe (London)	23:00–07:00 UTC
Europe (Paris)	23:00–07:00 UTC
Europe (Stockholm)	23:00–07:00 UTC
Middle East (Bahrain)	13:00–21:00 UTC
Middle East (UAE)	18:00 — 02:00 UTC

Wilayah	Jendela waktu pemeliharaan
South America (São Paulo)	19:00–03:00 UTC
Israel (Tel Aviv)	20.00–04:00 UTC

Tetapkan UUID untuk semua operator

Ketika Layanan Terkelola untuk Apache Flink memulai pekerjaan Flink untuk aplikasi dengan snapshot, pekerjaan Flink dapat gagal dimulai karena masalah tertentu. Salah satunya adalah ketidakcocokan ID operator. Flink mengharapkan ID operator yang eksplisit dan konsisten untuk operator grafik pekerjaan Flink. Jika tidak disetel secara eksplisit, Flink otomatis membuat ID untuk operator. Ini karena Flink menggunakan ID operator ini untuk mengidentifikasi operator secara unik dalam grafik pekerjaan dan menggunakannya untuk menyimpan status setiap operator di savepoint.

Masalah ketidakcocokan ID operator terjadi ketika Flink tidak menemukan pemetaan 1:1 antara ID operator grafik pekerjaan dan ID operator yang ditentukan dalam savepoint. Ini terjadi ketika ID operator konsisten eksplisit tidak disetel dan Flink secara otomatis menghasilkan ID operator yang mungkin tidak konsisten dengan setiap pembuatan grafik pekerjaan. Kemungkinan aplikasi mengalami masalah ini tinggi selama pemeliharaan berjalan. Untuk menghindari hal ini, kami menyarankan pelanggan mengatur UUID untuk semua operator dalam kode flink. Untuk informasi selengkapnya, lihat topik Menetapkan UUID untuk semua operator di bawah Kesiapan produksi.

Kesiapan produksi

Ini adalah kumpulan aspek penting dari menjalankan aplikasi produksi pada Managed Service untuk Apache Flink. Ini bukan daftar lengkap, melainkan minimum dari apa yang harus Anda perhatikan sebelum memasukkan aplikasi ke dalam produksi.

Memuat aplikasi pengujian

Beberapa masalah dengan aplikasi hanya bermanifestasi di bawah beban berat. Kami telah melihat kasus di mana aplikasi tampak sehat dan peristiwa operasional secara substansif memperkuat beban pada aplikasi. Ini dapat terjadi sepenuhnya independen dari aplikasi itu sendiri: Jika sumber data atau data sink tidak tersedia selama beberapa jam, aplikasi Flink tidak dapat membuat kemajuan. Setelah masalah itu diperbaiki, ada tumpukan data yang belum diproses telah terakumulasi yang dapat sepenuhnya menghabiskan sumber daya yang tersedia. Beban kemudian dapat memperkuat bug atau masalah kinerja yang belum pernah muncul sebelumnya.

Oleh karena itu penting untuk menjalankan tes beban yang tepat untuk aplikasi produksi. Pertanyaan yang harus dijawab selama tes beban tersebut meliputi:

- Apakah aplikasi stabil di bawah beban tinggi yang berkelanjutan?
- Bisakah aplikasi masih mengambil savepoint di bawah beban puncak?
- Berapa lama waktu yang dibutuhkan untuk memproses backlog 1 jam? Dan berapa lama selama 24 jam (tergantung pada retensi maksimal data dalam aliran)?
- Apakah throughput aplikasi meningkat saat aplikasi diskalakan?

Ketika mengkonsumsi dari aliran data, skenario ini dapat disimulasikan dengan memproduksi ke dalam aliran untuk beberapa waktu. Kemudian mulai aplikasi dan minta itu mengkonsumsi data dari awal waktu, misalnya, menggunakan posisi awal `TRIM_HORIZON` dalam kasus Kinesis Data Stream.

Paralelisme maks

Paralelisme maks mendefinisikan paralelisme maksimum yang dapat diskalakan oleh aplikasi stateful. Ini didefinisikan ketika status pertama kali dibuat dan tidak ada cara untuk menskalakan operator di luar maksimum ini tanpa membuang status.

Max Parallelism diatur saat status pertama kali dibuat.

Secara default, Max Parallelism diatur ke:

- 128, jika paralelisme <= 128
- `MIN(nextPowerOfTwo(parallelism + (parallelism / 2)), 2^15)`: jika paralelisme> 128

Jika Anda berencana untuk menskalakan paralelisme aplikasi > 128, Anda harus secara eksplisit mendefinisikan Paralelisme Maks.

Max Parallelism dapat didefinisikan pada tingkat aplikasi, dengan `env.setMaxParallelism(x)` atau operator tunggal. Kecuali ditentukan secara berbeda, semua operator mewarisi Paralelisme Maks aplikasi.

Untuk informasi selengkapnya, lihat Mengatur Paralelisme Maks Eksplisit dalam dokumentasi Flink.

Tetapkan UUID untuk semua operator

UUID digunakan dalam operasi di mana Flink memetakan savepoint kembali ke operator individu. Menyetel UUID tertentu untuk setiap operator memberikan pemetaan yang stabil untuk proses savepoint untuk dipulihkan.

```
.map(...).uid("my-map-function")
```

Untuk informasi selengkapnya, lihat Daftar Periksa Kesiapan Produksi.

Praktik Terbaik untuk Layanan Terkelola untuk Apache Flink

Bagian ini berisi informasi dan rekomendasi untuk mengembangkan Layanan Terkelola yang stabil dan berkinerja baik untuk aplikasi Apache Flink.

Topik

Toleransi kesalahan: titik pemeriksaan dan titik simpan

Gunakan pos pemeriksaan dan savepoint untuk menerapkan toleransi kesalahan dalam Layanan Terkelola untuk aplikasi Apache Flink Anda. Ingat hal berikut saat mengembangkan dan memelihara aplikasi Anda:

- Sebaiknya aktifkan checkpointing untuk aplikasi Anda. Checkpointing memberikan toleransi kesalahan untuk aplikasi Anda selama pemeliharaan terjadwal, dan dalam kasus kegagalan tak terduga karena masalah layanan, kegagalan dependensi aplikasi, dan masalah lainnya. Untuk informasi tentang pemeliharaan terjadwal, lihat Pemeliharaan.
- Set ApplicationSnapshotConfiguration:: SnapshotsEnabled ke `false` selama pengembangan aplikasi atau pemecahan masalah. Snapshot dibuat selama setiap aplikasi berhenti, yang dapat

menyebabkan masalah jika aplikasi dalam keadaan tidak sehat atau tidak berkinerja. Atur `SnapshotsEnabled` ke `true` setelah aplikasi dalam produksi dan stabil.

> **Note**
>
> Sebaiknya aplikasi Anda membuat snapshot beberapa kali sehari untuk memulai ulang dengan benar menggunakan data status yang benar. Frekuensi yang benar untuk snapshot Anda bergantung pada logika bisnis aplikasi Anda. Sering mengambil snapshot memungkinkan Anda memulihkan data yang lebih baru, tetapi meningkatkan biaya dan membutuhkan lebih banyak sumber daya sistem.

Untuk informasi tentang pemantauan waktu henti aplikasi, lihat Metrik dan Dimensi dalam Layanan Terkelola untuk Apache Flink.

Untuk informasi selengkapnya tentang penerapan toleransi kegagalan, lihat Toleransi Kesalahan.

Versi konektor yang tidak didukung

Layanan Terkelola untuk Apache Flink versi 1.15 akan secara otomatis mencegah aplikasi memulai atau memperbarui jika mereka menggunakan versi Konektor Kinesis yang tidak didukung (dibundel ke dalam JAR aplikasi). Saat meningkatkan ke Managed Service untuk Apache Flink versi 1.15 harap pastikan bahwa Anda menggunakan Konektor Kinesis terbaru. Ini adalah versi apa pun yang sama dengan atau lebih baru dari versi 1.15.2. Semua versi lain tidak akan didukung oleh Layanan Terkelola untuk Apache Flink karena dapat menyebabkan masalah konsistensi atau kegagalan dengan fitur Stop with Savepoint mencegah operasi berhenti/pembaruan bersih.

Performa dan paralelisme

Aplikasi Anda dapat diskalakan untuk memenuhi tingkat throughput apa pun dengan menyetel paralelisme aplikasi Anda, dan menghindari perangkap performa. Ingat hal berikut saat mengembangkan dan memelihara aplikasi Anda:

- Verifikasi bahwa semua sumber aplikasi dan sink Anda ditetapkan dengan cukup dan tidak dibatasi. Jika sumber dan wastafel adalah AWS layanan lain, pantau layanan tersebut menggunakan CloudWatch.

- Untuk aplikasi dengan paralelisme yang sangat tinggi, periksa apakah tingkat paralelisme yang tinggi diterapkan pada semua operator dalam aplikasi. Secara default, Apache Flink menerapkan paralelisme aplikasi yang sama untuk semua operator dalam grafik aplikasi. Ini dapat menyebabkan masalah penyediaan pada sumber atau sink, atau pun hambatan dalam pemrosesan data operator. Anda dapat mengubah paralelisme setiap operator dalam kode dengan setParallelism.
- Pahami arti pengaturan paralelisme untuk operatori dalam aplikasi Anda. Jika Anda mengubah paralelisme untuk operator, Anda mungkin tidak dapat memulihkan aplikasi dari snapshot yang dibuat ketika operator memiliki paralelisme yang tidak kompatibel dengan pengaturan saat ini. Untuk informasi selengkapnya tentang pengaturan paralelisme operator, lihat Mengatur paralelisme maksimum untuk operator secara eksplisit.

Untuk informasi selengkapnya tentang penerapan penskalaan, lihat Penskalaan.

Pengaturan paralelisme per operator

Secara default, semua operator memiliki paralelisme yang ditetapkan pada tingkat aplikasi. Anda dapat mengganti paralelisme dari satu operator menggunakan API. DataStream `.setParallelism(x)` Anda dapat mengatur paralelisme operator ke paralelisme apa pun yang sama atau lebih rendah dari paralelisme aplikasi.

Jika memungkinkan, tentukan paralelisme operator sebagai fungsi dari paralelisme aplikasi. Dengan cara ini, paralelisme operator akan bervariasi dengan paralelisme aplikasi. Jika Anda menggunakan penskalaan otomatis, misalnya, semua operator akan memvariasikan paralelisme mereka dalam proporsi yang sama:

```
int appParallelism = env.getParallelism();
...
...ops.setParalleism(appParallelism/2);
```

Dalam beberapa kasus, Anda mungkin ingin mengatur paralelisme operator ke konstanta. Misalnya, mengatur paralelisme sumber Aliran Kinesis ke jumlah pecahan. Dalam kasus ini, Anda harus mempertimbangkan untuk meneruskan paralelisme operator sebagai parameter konfigurasi aplikasi, untuk mengubahnya tanpa mengubah kode, jika Anda perlu, misalnya, untuk mengubah aliran sumber.

Pencatatan log

Anda dapat memantau kinerja dan kondisi kesalahan aplikasi Anda menggunakan CloudWatch Log. Ingat hal berikut saat mengonfigurasi pencatatan untuk aplikasi Anda:

- Aktifkan CloudWatch pencatatan untuk aplikasi sehingga masalah runtime apa pun dapat di-debug.
- Jangan buat entri log untuk setiap catatan yang diproses dalam aplikasi. Ini menyebabkan hambatan parah selama pemrosesan dan dapat menyebabkan tekanan balik dalam pemrosesan data.
- Buat CloudWatch alarm untuk memberi tahu Anda ketika aplikasi Anda tidak berjalan dengan benar. Untuk informasi selengkapnya, lihat Alarm

Untuk informasi selengkapnya tentang penerapan pencatatan, lihat Pencatatan dan Pemantauan.

Pengkodean

Anda dapat membuat aplikasi Anda berfungsi dan stabil menggunakan praktik pemrograman yang direkomendasikan. Ingat hal berikut saat menulis kode aplikasi:

- Jangan gunakan `system.exit()` dalam kode aplikasi Anda, baik dalam metode `main` aplikasi Anda atau dalam fungsi yang ditetapkan pengguna. Jika Anda ingin menonaktifkan aplikasi Anda dari dalam kode, lempar pengecualian yang berasal dari `Exception` atau `RuntimeException`, yang berisi pesan tentang apa yang salah dengan aplikasi.

  Catat hal berikut tentang bagaimana layanan menangani pengecualian ini:

  - Jika pengecualian dilemparkan dari metode `main` aplikasi Anda, layanan akan membungkusnya dalam `ProgramInvocationException` saat transisi aplikasi ke status `RUNNING`, dan manajer tugas akan gagal mengirimkan tugas.
  - Jika pengecualian dilemparkan dari fungsi yang ditetapkan pengguna, manajer tugas akan gagal tugas dan memulai ulang, serta detail pengecualian akan ditulis ke log pengecualian.
- Pertimbangkan bayangan file JAR aplikasi Anda dan dependensi yang disertakan. Bayangan direkomendasikan ketika ada potensi konflik dalam nama paket antara aplikasi Anda dan runtime Apache Flink. Jika terjadi konflik, log aplikasi Anda mungkin berisi pengecualian tipe `java.util.concurrent.ExecutionException`. Untuk informasi selengkapnya tentang bayangan file JAR aplikasi Anda, lihat Plugin Apache Maven Shade.

Mengelola kredensi

Anda tidak boleh memanggang kredensi jangka panjang apa pun ke dalam aplikasi produksi (atau lainnya). Kredensi jangka panjang kemungkinan diperiksa ke dalam sistem kontrol versi dan dapat dengan mudah hilang. Sebagai gantinya, Anda dapat mengaitkan peran ke Layanan Terkelola untuk aplikasi Apache Flink dan memberikan hak istimewa untuk peran tersebut. Aplikasi Flink yang sedang berjalan kemudian dapat mengambil kredensyal sementara dengan hak istimewa masing-masing dari lingkungan. Jika otentikasi diperlukan untuk layanan yang tidak terintegrasi secara native dengan IAM, misalnya database yang memerlukan nama pengguna dan kata sandi untuk otentikasi, Anda harus mempertimbangkan untuk menyimpan rahasia di Secrets Manager. AWS

Banyak layanan AWS asli mendukung otentikasi:

- Kinesis Data ProcessTaxiStream Streams — .java
- Amazon MSK - https://github.com/aws/ aws-msk-iam-auth using-the-amazon-msk /# - library-for-iam-authentication
- Amazon Elasticsearch Service — .java AmazonElasticsearchSink
- Amazon S3 - bekerja di luar kotak pada Layanan Terkelola untuk Apache Flink

Membaca dari sumber dengan sedikit pecahan/partisi

Saat membaca dari Apache Kafka atau Aliran Data Kinesis, mungkin ada ketidakcocokan antara paralelisme aliran (yaitu, jumlah partisi untuk Kafka dan jumlah pecahan untuk Kinesis) dan paralelisme aplikasi. Dengan desain yang naif, paralelisme aplikasi tidak dapat berskala melampaui paralelisme aliran: Setiap subtugas operator sumber hanya dapat membaca dari 1 atau lebih piringan/partisi. Itu berarti untuk aliran dengan hanya 2 pecahan dan aplikasi dengan paralelisme 8, bahwa hanya dua subtugas yang benar-benar memakan dari aliran dan 6 subtugas tetap menganggur. Ini secara substansional dapat membatasi throughput aplikasi, khususnya jika deserialisasi mahal dan dilakukan oleh sumber (yang merupakan default).

Untuk mengurangi efek ini, Anda dapat menskalakan aliran. Tapi itu mungkin tidak selalu diinginkan atau mungkin. Atau, Anda dapat merestrukturisasi sumber sehingga tidak melakukan serialisasi apa pun dan hanya meneruskan. `byte[]` Anda kemudian dapat menyeimbangkan kembali data untuk mendistribusikannya secara merata di semua tugas dan kemudian deserialisasi data di sana. Dengan cara ini, Anda dapat memanfaatkan semua subtugas untuk deserialisasi dan operasi yang berpotensi mahal ini tidak lagi terikat oleh jumlah shard/partisi aliran.

Interval refresh notebook Studio

Jika Anda mengubah interval refresh hasil paragraf, atur ke nilai yang setidaknya 1000 milidetik.

Performa optimum notebook Studio

Kami menguji dengan pernyataan berikut dan mendapat performa terbaik saat `events-per-second` yang dikalikan dengan `number-of-keys` berada di bawah 25.000.000. Ini adalah untuk `events-per-second` di bawah 150.000.

```
SELECT key, sum(value) FROM key-values GROUP BY key
```

Bagaimana strategi watermark dan pecahan idle memengaruhi jendela waktu

Saat membaca peristiwa dari Apache Kafka dan Kinesis Data Streams, sumber dapat mengatur waktu acara berdasarkan atribut aliran. Dalam kasus Kinesis, waktu acara sama dengan perkiraan waktu kedatangan peristiwa. Tetapi pengaturan waktu acara di sumber acara tidak cukup bagi aplikasi Flink untuk menggunakan waktu acara. Sumber juga harus menghasilkan tanda air yang menyebarkan informasi tentang waktu acara dari sumber ke semua operator lain. Dokumentasi Flink memiliki gambaran yang baik tentang cara kerja proses itu.

Secara default, stempel waktu peristiwa yang dibaca dari Kinesis diatur ke perkiraan waktu kedatangan yang ditentukan oleh Kinesis. Prasyarat tambahan untuk waktu acara untuk bekerja dalam aplikasi adalah strategi watermark.

```
WatermarkStrategy<String> s = WatermarkStrategy
    .<String>forMonotonousTimestamps()
    .withIdleness(Duration.ofSeconds(...));
```

Strategi watermark kemudian diterapkan ke a `DataStream` with the `assignTimestampsAndWatermarks` method. Ada beberapa strategi build-in yang berguna:

- `forMonotonousTimestamps()`hanya akan menggunakan waktu acara (perkiraan waktu kedatangan) dan secara berkala memancarkan nilai maksimum sebagai tanda air (untuk setiap subtugas tertentu)

- `forBoundedOutOfOrderness(Duration.ofSeconds(...))`mirip dengan strategi sebelumnya, tetapi akan menggunakan waktu acara - durasi untuk pembuatan tanda air.

Ini berhasil, tetapi ada beberapa peringatan yang harus diperhatikan. Tanda air dihasilkan pada tingkat subtugas dan mengalir melalui grafik operator.

Dari dokumentasi Flink:

Setiap subtugas paralel dari fungsi sumber biasanya menghasilkan tanda airnya secara independen. Tanda air ini menentukan waktu acara pada sumber paralel tertentu.

Saat tanda air mengalir melalui program streaming, mereka memajukan waktu acara di operator tempat mereka tiba. Setiap kali operator memajukan waktu acaranya, ia menghasilkan tanda air baru di hilir untuk operator penggantinya.

Beberapa operator menggunakan beberapa aliran input; serikat pekerja, misalnya, atau operator yang mengikuti fungsi KeyBy (...) atau partisi (...). Waktu kejadian operator saat ini adalah minimum waktu acara aliran inputnya. Karena aliran inputnya memperbarui waktu acara mereka, begitu juga operator.

Itu berarti, jika subtugas sumber mengkonsumsi dari pecahan siaga, operator hilir tidak menerima tanda air baru dari subtugas itu dan karenanya memproses stall untuk semua operator hilir yang menggunakan jendela waktu. Untuk menghindari hal ini, pelanggan dapat menambahkan `withIdleness` opsi ke strategi tanda air. Dengan opsi itu, operator mengecualikan tanda air dari subtugas upsteam idle saat menghitung waktu acara operator. Subtugas idle karenanya tidak lagi memblokir kemajuan waktu acara di operator hilir.

Namun, opsi kemalasan dengan strategi tanda air bawaan tidak akan memajukan waktu acara jika tidak ada subtugas yang membaca acara apa pun, yaitu, tidak ada acara dalam aliran. Ini menjadi sangat terlihat untuk kasus uji di mana serangkaian peristiwa terbatas dibaca dari aliran. Karena waktu acara tidak berlanjut setelah acara terakhir dibaca, jendela terakhir (berisi acara terakhir) tidak akan pernah ditutup.

Ringkasan

- `withIdleness`pengaturan tidak akan menghasilkan tanda air baru jika pecahan menganggur, itu hanya akan mengecualikan tanda air terakhir yang dikirim oleh subtugas idle dari perhitungan tanda air min di operator hilir

- dengan strategi tanda air bawaan, jendela terbuka terakhir tidak akan pernah ditutup (kecuali acara baru yang memajukan tanda air akan dikirim, tetapi itu menciptakan jendela baru yang kemudian tetap terbuka)
- bahkan ketika waktu diatur oleh aliran Kinesis, peristiwa kedatangan terlambat masih dapat terjadi jika satu pecahan dikonsumsi lebih cepat daripada yang lain (misalnya, selama inisialisasi aplikasi atau saat menggunakan `TRIM_HORIZON` di mana semua pecahan yang ada dikonsumsi secara paralel mengabaikan hubungan orang tua/anak mereka)
- `withIdleness`pengaturan strategi tanda air tampaknya menghentikan pengaturan khusus sumber Kinesis untuk pecahan siaga (`ConsumerConfigConstants.SHARD_IDLE_INTERVAL_MILLIS`

Contoh

Aplikasi berikut membaca dari aliran dan membuat jendela sesi berdasarkan waktu acara.

```
Properties consumerConfig = new Properties();
consumerConfig.put(AWSConfigConstants.AWS_REGION, "eu-west-1");
consumerConfig.put(ConsumerConfigConstants.STREAM_INITIAL_POSITION, "TRIM_HORIZON");

FlinkKinesisConsumer<String> consumer = new FlinkKinesisConsumer<>("...", new
 SimpleStringSchema(), consumerConfig);

WatermarkStrategy<String> s = WatermarkStrategy
    .<String>forMonotonousTimestamps()
    .withIdleness(Duration.ofSeconds(15));

env.addSource(consumer)
    .assignTimestampsAndWatermarks(s)
    .map(new MapFunction<String, Long>() {
        @Override
        public Long map(String s) throws Exception {
            return Long.parseLong(s);
        }
    })
    .keyBy(l -> 0l)
    .window(EventTimeSessionWindows.withGap(Time.seconds(10)))
    .process(new ProcessWindowFunction<Long, Object, Long, TimeWindow>() {
        @Override
```

```
        public void process(Long aLong, ProcessWindowFunction<Long, Object, Long,
 TimeWindow>.Context context, Iterable<Long>iterable, Collector<Object> collector)
 throws Exception {
            long count = StreamSupport.stream(iterable.spliterator(), false).count();
            long timestamp = context.currentWatermark();

            System.out.print("XXXXXXXXXXXXXX Window with " + count + " events");
            System.out.println("; Watermark: " + timestamp + ", " +
 Instant.ofEpochMilli(timestamp));


            for (Long l : iterable) {
                System.out.println(l);
            }
        }
    });
```

Dalam contoh berikut, 8 peristiwa ditulis ke aliran pecahan 16 (2 yang pertama dan peristiwa terakhir kebetulan mendarat di pecahan yang sama).

```
$ aws kinesis put-record --stream-name hp-16 --partition-key 1 --data MQ==
$ aws kinesis put-record --stream-name hp-16 --partition-key 2 --data Mg==
$ aws kinesis put-record --stream-name hp-16 --partition-key 3 --data Mw==
$ date

{
    "ShardId": "shardId-000000000012",
    "SequenceNumber": "49627894338614655560500811028721934184977530127978070210"
}
{
    "ShardId": "shardId-000000000012",
    "SequenceNumber": "49627894338614655560500811028795678659974022576354623682"
}
{
    "ShardId": "shardId-000000000014",
    "SequenceNumber": "49627894338659257050897872275134360684221592378842022114"
}
Wed Mar 23 11:19:57 CET 2022

$ sleep 10
$ aws kinesis put-record --stream-name hp-16 --partition-key 4 --data NA==
$ aws kinesis put-record --stream-name hp-16 --partition-key 5 --data NQ==
$ date
```

```
{
    "ShardId": "shardId-000000000010",
    "SequenceNumber": "49627894338570054070103749783042116732419934393936642210"
}
{
    "ShardId": "shardId-000000000014",
    "SequenceNumber": "49627894338659257050897872275659034489934342334017700066"
}
Wed Mar 23 11:20:10 CET 2022

$ sleep 10
$ aws kinesis put-record --stream-name hp-16 --partition-key 6 --data Ng==
$ date

{
    "ShardId": "shardId-000000000001",
    "SequenceNumber": "49627894338369347363316974173886988345467035365375213586"
}
Wed Mar 23 11:20:22 CET 2022

$ sleep 10
$ aws kinesis put-record --stream-name hp-16 --partition-key 7 --data Nw==
$ date

{
    "ShardId": "shardId-000000000008",
    "SequenceNumber": "49627894338525452579706688535878947299195189349725503618"
}
Wed Mar 23 11:20:34 CET 2022

$ sleep 60
$ aws kinesis put-record --stream-name hp-16 --partition-key 8 --data OA==
$ date

{
    "ShardId": "shardId-000000000012",
    "SequenceNumber": "49627894338614655560500811029600823255837371928900796610"
}
Wed Mar 23 11:21:27 CET 2022
```

Masukan ini akan menghasilkan jendela sesi 5: event 1,2,3; event 4,5; event 6; event 7; event 8. Namun, program ini hanya menghasilkan 4 jendela pertama.

```
11:59:21,529 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 5 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000006,HashKeyRange: {StartingHashKey:
 127605887595351923798765477786913079296,EndingHashKey:
 148873535527910577765226390751398592511},SequenceNumberRange: {StartingSequenceNumber:
 49627894338480851089309627289524549239292625588395704418,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,530 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 5 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000006,HashKeyRange: {StartingHashKey:
 127605887595351923798765477786913079296,EndingHashKey:
 148873535527910577765226390751398592511},SequenceNumberRange: {StartingSequenceNumber:
 49627894338480851089309627289524549239292625588395704418,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:21,530 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 6 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000007,HashKeyRange: {StartingHashKey:
 148873535527910577765226390751398592512,EndingHashKey:
 170141183460469231731687303715884105727},SequenceNumberRange: {StartingSequenceNumber:
 49627894338503151834508157912666084957565273949901684850,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,530 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 6 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000010,HashKeyRange: {StartingHashKey:
 212676479325586539664609129644855132160,EndingHashKey:
 233944127258145193631070042609340645375},SequenceNumberRange: {StartingSequenceNumber:
 49627894338570054070103749782090692112383219034419626146,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,530 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 6 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000007,HashKeyRange: {StartingHashKey:
 148873535527910577765226390751398592512,EndingHashKey:
 170141183460469231731687303715884105727},SequenceNumberRange: {StartingSequenceNumber:
 49627894338503151834508157912666084957565273949901684850,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:21,531 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 4 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000005,HashKeyRange: {StartingHashKey:
 106338239662793269832304564822427566080,EndingHashKey:
 127605887595351923798765477786913079295},SequenceNumberRange: {StartingSequenceNumber:
```

```
 496278943384585503441110966663830135210199772268897239 86,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,532 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 4 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000005,HashKeyRange: {StartingHashKey:
 106338239662793269832304564822427566080,EndingHashKey:
 127605887595351923798765477786913079295},SequenceNumberRange: {StartingSequenceNumber:
 496278943384585503441110966663830135210199772268897239 86,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:21,532 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 3 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000004,HashKeyRange: {StartingHashKey:
 850705917302346158658436518579420528 64,EndingHashKey:
 106338239662793269832304564822427566079},SequenceNumberRange: {StartingSequenceNumber:
 496278943384362495989125660432414778027473288653837435 54,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,532 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 2 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000003,HashKeyRange: {StartingHashKey:
 638029437976759618993827388934565396 48,EndingHashKey:
 850705917302346158658436518579420528 63},SequenceNumberRange: {StartingSequenceNumber:
 496278943384139488537140354200999420844746805038777631 22,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,532 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 3 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000015,HashKeyRange: {StartingHashKey:
 319014718988379809496913694467282698240,EndingHashKey:
 340282366920938463463374607431768211455},SequenceNumberRange: {StartingSequenceNumber:
 496278943386815577960964028977983707037464608419495283 06,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,532 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 2 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000014,HashKeyRange: {StartingHashKey:
 297747071055821155530452781502797185024,EndingHashKey:
 319014718988379809496913694467282698239},SequenceNumberRange: {StartingSequenceNumber:
 496278943386592570508978722746568349854738124804435478 74,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,532 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 3 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000004,HashKeyRange: {StartingHashKey:
 850705917302346158658436518579420528 64,EndingHashKey:
 106338239662793269832304564822427566079},SequenceNumberRange: {StartingSequenceNumber:
```

```
 4962789433843624959891256604324147780274732886538374355 4,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:21,532 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 2 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000003,HashKeyRange: {StartingHashKey:
 638029437976759618993827388934565396 48,EndingHashKey:
 850705917302346158658436518579420528 63},SequenceNumberRange: {StartingSequenceNumber:
 4962789433841394885371403542009994208447468050387776312 2,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:21,532 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 0 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000001,HashKeyRange: {StartingHashKey:
 212676479325586539664609129644855132 16,EndingHashKey:
 425352958651173079329218259289710264 31},SequenceNumberRange: {StartingSequenceNumber:
 4962789433836934736331697417381687064792938378086580225 8,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,532 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 0 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000009,HashKeyRange: {StartingHashKey:
 191408831393027885698148216680369618 944,EndingHashKey:
 212676479325586539664609129644855132 159},SequenceNumberRange: {StartingSequenceNumber:
 4962789433854775332490521915894915639411057067291364571 4,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,532 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 7 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000000,HashKeyRange: {StartingHashKey: 0,EndingHashKey:
 212676479325586539664609129644855132 15},SequenceNumberRange: {StartingSequenceNumber:
 4962789433834704661811844355067533492965673541935982182 6,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,533 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 0 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000012,HashKeyRange: {StartingHashKey:
 255211775190703847597530955573826158 592,EndingHashKey:
 276479423123262501563991868538311671 807},SequenceNumberRange: {StartingSequenceNumber:
 4962789433861465556050081102837376354892851575743158701 0,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,533 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 7 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000008,HashKeyRange: {StartingHashKey:
 170141183460469231731687303715884105 728,EndingHashKey:
 191408831393027885698148216680369618 943},SequenceNumberRange: {StartingSequenceNumber:
 4962789433852545257970668853580762067583792231140766528 2,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
```

```
11:59:21,533 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 0 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000001,HashKeyRange: {StartingHashKey:
 212676479325586539664609129644855132l6,EndingHashKey:
 42535295865117307932921825928971026431},SequenceNumberRange: {StartingSequenceNumber:
 49627894338369347363316974173816870647929383780865802258,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:21,533 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 7 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000011,HashKeyRange: {StartingHashKey:
 233944127258145193631070042609340645376,EndingHashKey:
 255211775190703847597530955573826158591},SequenceNumberRange: {StartingSequenceNumber:
 49627894338592354815302280405232227830655867395925606578,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,533 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 7 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000000,HashKeyRange: {StartingHashKey: 0,EndingHashKey:
 212676479325586539664609129644855132l5},SequenceNumberRange: {StartingSequenceNumber:
 49627894338347046618118443550675334929656735419359821826,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:21,568 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 1 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000002,HashKeyRange: {StartingHashKey:
 42535295865117307932921825928971026432,EndingHashKey:
 63802943797675961899382738893456539647},SequenceNumberRange: {StartingSequenceNumber:
 49627894338391648108515504796958406366202032142371782690,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,568 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 1 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000013,HashKeyRange: {StartingHashKey:
 27647942312326250156399186853831167l808,EndingHashKey:
 29774707105582115553045278150279718502 3},SequenceNumberRange: {StartingSequenceNumber:
 49627894338636956305699341651515299267201164118937567442,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,568 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 1 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000002,HashKeyRange: {StartingHashKey:
 42535295865117307932921825928971026432,EndingHashKey:
 63802943797675961899382738893456539647},SequenceNumberRange: {StartingSequenceNumber:
 49627894338391648108515504796958406366202032142371782690,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 0
```

```
11:59:23,209 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 0 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000009,HashKeyRange: {StartingHashKey:
 191408831393027885698148216680369618944,EndingHashKey:
 212676479325586539664609129644855132159},SequenceNumberRange: {StartingSequenceNumber:
 496278943385477533249052191589491563941105706729136457l4,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 1
11:59:23,244 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 6 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000010,HashKeyRange: {StartingHashKey:
 212676479325586539664609129644855132160,EndingHashKey:
 233944127258145193631070042609340645375},SequenceNumberRange: {StartingSequenceNumber:
 496278943385700540701037497820906921123832190344196261460,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 1
event: 6; timestamp: 1648030822428, 2022-03-23T10:20:22.428Z
11:59:23,377 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 3 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000015,HashKeyRange: {StartingHashKey:
 319014718988379809496913694467282698240,EndingHashKey:
 340282366920938463463374607431768211455},SequenceNumberRange: {StartingSequenceNumber:
 496278943386815577960964028977983707037464608419495283060,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 1
11:59:23,405 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 2 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000014,HashKeyRange: {StartingHashKey:
 297747071055821155530452781502797185024,EndingHashKey:
 319014718988379809496913694467282698239},SequenceNumberRange: {StartingSequenceNumber:
 496278943386592570508978722746568349854738124804435478l4,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 1
11:59:23,581 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 7 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000008,HashKeyRange: {StartingHashKey:
 170141183460469231731687303715884105728,EndingHashKey:
 191408831393027885698148216680369618943},SequenceNumberRange: {StartingSequenceNumber:
 496278943385254525797066885358076206758379223114076652820,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 1
11:59:23,586 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 1 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
```

```
 shard='{ShardId: shardId-000000000013,HashKeyRange: {StartingHashKey:
 276479423123262501563991868538311671808,EndingHashKey:
 297747071055821155530452781502797185023},SequenceNumberRange: {StartingSequenceNumber:
 496278943386369563056993416515152992672011641189375674 42,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 1
11:59:24,790 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 0 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000012,HashKeyRange: {StartingHashKey:
 255211775190703847597530955573826158592,EndingHashKey:
 276479423123262501563991868538311671807},SequenceNumberRange: {StartingSequenceNumber:
 496278943386146555605008110283737635489285157574315870 10,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 2
event: 4; timestamp: 1648030809282, 2022-03-23T10:20:09.282Z
event: 3; timestamp: 1648030797697, 2022-03-23T10:19:57.697Z
event: 5; timestamp: 1648030810871, 2022-03-23T10:20:10.871Z
11:59:24,907 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 7 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000011,HashKeyRange: {StartingHashKey:
 233944127258145193631070042609340645376,EndingHashKey:
 255211775190703847597530955573826158591},SequenceNumberRange: {StartingSequenceNumber:
 496278943385923548153022804052322278306558673959256065 78,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 2
event: 7; timestamp: 1648030834105, 2022-03-23T10:20:34.105Z
event: 1; timestamp: 1648030794441, 2022-03-23T10:19:54.441Z
event: 2; timestamp: 1648030796122, 2022-03-23T10:19:56.122Z
event: 8; timestamp: 1648030887171, 2022-03-23T10:21:27.171Z
XXXXXXXXXXXXXX Window with 3 events; Watermark: 1648030809281, 2022-03-23T10:20:09.281Z
3
1
2
XXXXXXXXXXXXXX Window with 2 events; Watermark: 1648030834104, 2022-03-23T10:20:34.104Z
4
5
XXXXXXXXXXXXXX Window with 1 events; Watermark: 1648030834104, 2022-03-23T10:20:34.104Z
6
XXXXXXXXXXXXXX Window with 1 events; Watermark: 1648030887170, 2022-03-23T10:21:27.170Z
7
```

Outputnya hanya menampilkan 4 jendela (tidak ada jendela terakhir yang berisi acara 8). Ini karena waktu acara dan strategi tanda air. Jendela terakhir tidak dapat ditutup karena dengan strategi watermark per built waktu tidak pernah maju melampaui waktu peristiwa terakhir yang telah dibaca

dari aliran. Tetapi agar jendela ditutup, waktu perlu maju lebih dari 10 detik setelah acara terakhir. Dalam hal ini tanda air terakhir adalah 2022-03-23T 10:21:27.170 Z tetapi agar jendela sesi ditutup, diperlukan tanda air 10 detik dan 1 ms kemudian.

Jika `withIdleness` opsi dihapus dari strategi tanda air, tidak ada jendela sesi yang akan ditutup, karena “tanda air global” dari operator jendela tidak dapat maju.

Perhatikan bahwa ketika aplikasi Flink dimulai (atau jika ada kemiringan data), beberapa pecahan dapat dikonsumsi lebih cepat daripada yang lain. Hal ini dapat menyebabkan beberapa tanda air dipancarkan terlalu dini dari subtugas (subtugas dapat memancarkan tanda air berdasarkan konten satu pecahan tanpa dikonsumsi dari pecahan lain yang dilangganannya). Cara untuk mengurangi adalah strategi watermarking yang berbeda yang menambahkan buffer keamanan `(forBoundedOutOfOrderness(Duration.ofSeconds(30))` atau secara eksplisit memungkinkan acara kedatangan terlambat. `(allowedLateness(Time.minutes(5))`

Tetapkan UUID untuk semua operator

Ketika Layanan Terkelola untuk Apache Flink memulai pekerjaan Flink untuk aplikasi dengan snapshot, pekerjaan Flink dapat gagal dimulai karena masalah tertentu. Salah satunya adalah ketidakcocokan ID operator. Flink mengharapkan ID operator yang eksplisit dan konsisten untuk operator grafik pekerjaan Flink. Jika tidak disetel secara eksplisit, Flink otomatis membuat ID untuk operator. Ini karena Flink menggunakan ID operator ini untuk mengidentifikasi operator secara unik dalam grafik pekerjaan dan menggunakannya untuk menyimpan status setiap operator di savepoint.

Masalah ketidakcocokan ID operator terjadi ketika Flink tidak menemukan pemetaan 1:1 antara ID operator grafik pekerjaan dan ID operator yang ditentukan dalam savepoint. Ini terjadi ketika ID operator konsisten eksplisit tidak disetel dan Flink secara otomatis menghasilkan ID operator yang mungkin tidak konsisten dengan setiap pembuatan grafik pekerjaan. Kemungkinan aplikasi mengalami masalah ini tinggi selama pemeliharaan berjalan. Untuk menghindari hal ini, kami menyarankan pelanggan mengatur UUID untuk semua operator dalam kode flink. Untuk informasi selengkapnya, lihat topik Menetapkan UUID untuk semua operator di bawah Kesiapan produksi.

Tambahkan ServiceResourceTransformer ke Plugin Maven Shade

Flink menggunakan Java Service Provider Interfaces (SPI) untuk memuat komponen seperti konektor dan format. Beberapa dependensi Flink menggunakan SPI dapat menyebabkan bentrokan di uber-jar dan perilaku aplikasi yang tidak terduga. Disarankan untuk menambahkan plugin maven shade, yang didefinisikan dalam pom.xml ServiceResourceTransformer

```
<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-shade-plugin</artifactId>
                <executions>
                    <execution>
                        <id>shade</id>
                        <phase>package</phase>
                        <goals>
                            <goal>shade</goal>
                        </goals>
                        <configuration>
                            <transformers combine.children="append">
                                <!-- The service transformer is needed to merge META-
INF/services files -->
                                <transformer
 implementation="org.apache.maven.plugins.shade.resource.ServicesResourceTransformer"/>
                                <!-- ... -->
                            </transformers>
                        </configuration>
                    </execution>
                </executions>
        </plugin>
```

Fungsi Stateful Apache Flink

Fungsi Statefuladalah API yang menyederhanakan pembuatan aplikasi stateful terdistribusi. Ini didasarkan pada fungsi dengan keadaan persisten yang dapat berinteraksi secara dinamis dengan jaminan konsistensi yang kuat.

Aplikasi Stateful Functions pada dasarnya hanyalah Aplikasi Apache Flink dan karenanya dapat digunakan untuk Managed Service untuk Apache Flink. Namun, ada beberapa perbedaan antara mengemas Stateful Functions untuk klaster Kubernetes dan Managed Service untuk Apache Flink. Aspek terpenting dari aplikasi Stateful Functions adalahkonfigurasi modulberisi semua informasi runtime yang diperlukan untuk mengonfigurasi runtime Stateful Functions. Konfigurasi ini biasanya dikemas ke dalam kontainer khusus Stateful Functions dan di-deploy pada Kubernetes. Tapi itu tidak mungkin dengan Managed Service untuk Apache Flink.

Berikut ini adalah adaptasi dariStateFunContoh Python untuk Managed Service untuk Apache Flink:

Templat Aplikasi Apache Flink

Alih-alih menggunakan wadah pelanggan untuk runtime Stateful Functions, pelanggan dapat mengkompilasi jar aplikasi Flink yang hanya memanggil runtime Stateful Functions dan berisi dependensi yang diperlukan. Untuk Flink 1.13, dependensi yang diperlukan terlihat mirip dengan ini:

```
<dependency>
 <groupId>org.apache.flink</groupId>
 <artifactId>statefun-flink-distribution</artifactId>
 <version>3.1.0</version>
 <exclusions>
  <exclusion>
   <groupId>org.slf4j</groupId>
   <artifactId>slf4j-log4j12</artifactId>
  </exclusion>
  <exclusion>
   <groupId>log4j</groupId>
   <artifactId>log4j</artifactId>
  </exclusion>
 </exclusions>
</dependency>
```

Dan metode utama aplikasi Flink untuk menjalankan runtime Stateful Function terlihat seperti ini:

```
public static void main(String[] args) throws Exception {
 final StreamExecutionEnvironment env =
 StreamExecutionEnvironment.getExecutionEnvironment();

 StatefulFunctionsConfig stateFunConfig = StatefulFunctionsConfig.fromEnvironment(env);

 stateFunConfig.setProvider((StatefulFunctionsUniverseProvider) (classLoader,
 statefulFunctionsConfig) -> {
  Modules modules = Modules.loadFromClassPath();
  return modules.createStatefulFunctionsUniverse(stateFunConfig);
 });

 StatefulFunctionsJob.main(env, stateFunConfig);
}
```

Perhatikan bahwa komponen-komponen ini bersifat generik dan independen dari logika yang diimplementasikan dalam Fungsi Stateful.

Lokasi konfigurasi modul

Konfigurasi modul Stateful Functions perlu disertakan dalam jalur kelas agar dapat ditemukan untuk runtime Stateful Functions. Yang terbaik adalah memasukkannya ke dalam folder sumber daya aplikasi Flink dan mengemasnya ke dalam file jar.

Mirip dengan aplikasi Apache Flink umum, Anda kemudian dapat menggunakan maven untuk membuat file jar uber dan menyebarkannya pada Managed Service untuk Apache Flink.

Informasi Versi Sebelumnya untuk Managed Service untuk Apache Flink

Topik ini berisi informasi tentang penggunaan Managed Service for Apache Flink dengan versi lama Apache Flink. Versi Apache Flink yang didukung Managed Service untuk Apache Flink adalah 1.15.2 (disarankan), 1.13.2, 1.11.1, 1.8.2 dan 1.6.2.

Kami menyarankan Anda menggunakan versi terbaru Apache Flink yang didukung dengan Layanan Terkelola untuk aplikasi Apache Flink Anda. Apache Flink versi 1.15.2 memiliki fitur berikut:

- Dukungan untuk API Tabel & SQL Apache Flink
- Dukungan untuk aplikasi Python.
- Support untuk Java versi 11 dan versi Scala
- Model memori yang ditingkatkan
- Optimasi RocksDB untuk stabilitas aplikasi yang ditingkatkan
- Dukungan untuk manajer tugas dan jejak tumpukan di Dasbor Apache Flink.

Topik ini berisi bagian-bagian berikut:

- Menggunakan Konektor Apache Flink Kinesis Streams dengan versi Apache Flink sebelumnya
- Membangun Aplikasi dengan Apache Flink 1.8.2
- Membangun Aplikasi dengan Apache Flink 1.6.2
- Meningkatkan Aplikasi
- Konektor yang Tersedia di Apache Flink 1.6.2 dan 1.8.2
- Memulai: Flink 1.13.2
- Memulai: Flink 1.11.1
- Memulai: Flink 1.8.2
- Memulai: Flink 1.6.2

Menggunakan Konektor Apache Flink Kinesis Streams dengan versi Apache Flink sebelumnya

Konektor Apache Flink Kinesis Stream tidak disertakan dalam Apache Flink sebelum versi 1.11. Agar aplikasi Anda dapat menggunakan konektor Apache Flink Kinesis dengan versi Apache Flink sebelumnya, Anda harus mengunduh, mengompilasi, dan menginstal versi Apache Flink yang digunakan aplikasi Anda. Konektor ini digunakan untuk mengonsumsi data dari aliran Kinesis yang digunakan sebagai sumber aplikasi, atau untuk menulis data ke aliran Kinesis yang digunakan untuk output aplikasi.

> **Note**
>
> Pastikan Anda membangun konektor dengan versi KPL 0.14.0 atau lebih tinggi.

Untuk mengunduh dan menginstal kode sumber Apache Flink versi 1.8.2, lakukan hal berikut:

1. Pastikan Anda menginstal Apache Maven, dan variabel lingkungan `JAVA_HOME` Anda merujuk ke JDK bukan JRE. Anda dapat menguji penginstalan Apache Maven Anda dengan perintah berikut:

   ```
   mvn -version
   ```

2. Unduh kode sumber Apache Flink versi 1.8.2:

   ```
   wget https://archive.apache.org/dist/flink/flink-1.8.2/flink-1.8.2-src.tgz
   ```

3. Batalkan kompresi kode sumber Apache Flink:

   ```
   tar -xvf flink-1.8.2-src.tgz
   ```

4. Ubah ke direktori kode sumber Apache Flink:

   ```
   cd flink-1.8.2
   ```

5. Komplikasi dan instal Apache Flink:

   ```
   mvn clean install -Pinclude-kinesis -DskipTests
   ```

> **Note**
>
> Jika Anda mengompilasi Flink di Microsoft Windows, Anda perlu menambahkan parameter `-Drat.skip=true`.

Membangun Aplikasi dengan Apache Flink 1.8.2

Bagian ini berisi informasi tentang komponen yang Anda gunakan untuk membangun Layanan Terkelola untuk aplikasi Apache Flink yang bekerja dengan Apache Flink 1.8.2.

Gunakan versi komponen berikut untuk Managed Service untuk aplikasi Apache Flink:

Komponen	Versi
Java	1.8 (direkomendasikan)
Apache Flink	1.8.2
Layanan Terkelola untuk Apache Flink untuk Flink Runtime () aws-kinesisanalytics-runtime	1.0.1
Layanan Terkelola untuk Konektor Apache Flink Flink () aws-kinesisanalytics-flink	1.0.1
Apache Maven	3.1

Untuk mengompilasi aplikasi menggunakan Apache Flink 1.8.2, jalankan Maven dengan parameter berikut:

```
mvn package -Dflink.version=1.8.2
```

Untuk contoh `pom.xml` file untuk aplikasi Managed Service for Apache Flink yang menggunakan Apache Flink versi 1.8.2, lihat Managed Service for Apache Flink 1.8.2 Memulai Aplikasi.

Untuk informasi tentang cara membuat dan menggunakan kode aplikasi untuk Layanan Terkelola untuk aplikasi Apache Flink, lihat. Membuat Aplikasi

Membangun Aplikasi dengan Apache Flink 1.6.2

Bagian ini berisi informasi tentang komponen yang Anda gunakan untuk membangun Layanan Terkelola untuk aplikasi Apache Flink yang bekerja dengan Apache Flink 1.6.2.

Gunakan versi komponen berikut untuk Managed Service untuk aplikasi Apache Flink:

Komponen	Versi
Java	1.8 (direkomendasikan)
SDK Java AWS	1.11.379
Apache Flink	1.6.2
Layanan Terkelola untuk Apache Flink untuk Flink Runtime () aws-kinesisanalytics-runtime	1.0.1
Layanan Terkelola untuk Konektor Apache Flink Flink () aws-kinesisanalytics-flink	1.0.1
Apache Maven	3.1
Apache Beam	Tidak didukung dengan Apache Flink 1.6.2.

> **Note**
>
> Saat menggunakan Managed Service for Apache Flink Runtime versi 1.0.1, Anda menentukan versi Apache Flink di `pom.xml` file Anda daripada menggunakan `-Dflink.version` parameter saat mengompilasi kode aplikasi Anda.

Untuk contoh `pom.xml` file untuk aplikasi Managed Service for Apache Flink yang menggunakan Apache Flink versi 1.6.2, lihat Managed Service for Apache Flink 1.6.2 Memulai Aplikasi.

Untuk informasi tentang cara membuat dan menggunakan kode aplikasi untuk Layanan Terkelola untuk aplikasi Apache Flink, lihat. Membuat Aplikasi

Meningkatkan Aplikasi

Untuk memutakhirkan versi Layanan Terkelola untuk aplikasi Apache Flink, Anda harus memperbarui kode aplikasi Anda, menghapus aplikasi sebelumnya, dan membuat aplikasi baru dengan kode yang diperbarui. Untuk melakukannya, lakukan hal berikut:

- Ubah versi Layanan Terkelola untuk Apache Flink Runtime dan Managed Service untuk konektor Apache Flink Flink (aws-kinesisanalytics-flink) di file aplikasi Anda menjadi 1.1.0. `pom.xml`
- Hapus properti `flink.version` dari file `pom.xml` aplikasi Anda. Anda akan memberikan parameter ini ketika Anda mengompilasi kode aplikasi di langkah berikutnya.
- Kompilasi ulang kode aplikasi Anda menggunakan perintah berikut:

```
mvn package -Dflink.version=1.15.3
```

- Hapus aplikasi yang ada. Buat aplikasi Anda lagi, dan pilih Apache Flink versi 1.15.2 (Versi yang disarankan) untuk Runtime aplikasi.

> **Note**
>
> Anda tidak dapat menggunakan snapshot dari versi aplikasi sebelumnya.

Konektor yang Tersedia di Apache Flink 1.6.2 dan 1.8.2

Kerangka kerja Apache Flink berisi konektor untuk mengakses data dari berbagai sumber.

- Untuk informasi tentang konektor yang tersedia di kerangka kerja Apache Flink 1.6.2, lihat Konektor (1.6.2) di Dokumentasi Apache Flink (1.6.2).
- Untuk informasi tentang konektor yang tersedia di kerangka kerja Apache Flink 1.8.2, lihat Konektor (1.8.2) di Dokumentasi Apache Flink (1.8.2).

Memulai: Flink 1.13.2

Bagian ini memperkenalkan Anda pada konsep dasar Managed Service untuk Apache Flink dan API. DataStream Ini menjelaskan opsi yang tersedia untuk membuat dan menguji aplikasi Anda. Ini juga

memberikan petunjuk untuk menginstal alat yang diperlukan untuk menyelesaikan tutorial dalam panduan ini dan untuk membuat aplikasi pertama Anda.

Topik

Komponen Layanan Terkelola untuk Aplikasi Apache Flink

Untuk memproses data, Managed Service untuk aplikasi Apache Flink Anda menggunakan aplikasi Java/Apache Maven atau Scala yang memproses input dan menghasilkan output menggunakan runtime Apache Flink.

Managed Service untuk aplikasi Apache Flink memiliki komponen-komponen berikut:

- Properti runtime: Anda dapat menggunakan properti runtime untuk mengonfigurasi aplikasi Anda tanpa mengompilasi ulang kode aplikasi Anda.
- Source (Sumber): Aplikasi mengonsumsi data menggunakan sumber. Konektor sumber membaca data dari Kinesis data stream, bucket Amazon S3, dll. Untuk informasi selengkapnya, lihat Sumber.
- Operators (Operator): Aplikasi memproses data menggunakan satu atau beberapa operator. Operator dapat mengubah, memperkaya, atau menggabungkan data. Untuk informasi selengkapnya, lihat DataStream Operator API.
- Sink: Aplikasi menghasilkan data ke sumber eksternal menggunakan sink. Konektor sink menulis data ke aliran data Kinesis, aliran Firehose Data Kinesis, bucket Amazon S3, dll. Untuk informasi selengkapnya, lihat Sink.

Setelah Anda membuat, mengompilasi, dan mengemas kode aplikasi Anda, Anda mengunggah paket kode ke bucket Amazon Simple Storage Service (Amazon S3). Anda kemudian membuat

Layanan Terkelola untuk aplikasi Apache Flink. Anda meneruskan di lokasi paket kode, Kinesis data stream sebagai sumber data streaming, dan biasanya lokasi streaming atau file yang menerima data yang diproses dari aplikasi.

Prasyarat untuk Menyelesaikan Latihan

Untuk menyelesaikan langkah-langkah di panduan ini, Anda harus memiliki hal-hal berikut:

- Java Development Kit (JDK) versi 11. Atur variabel lingkungan `JAVA_HOME` untuk menunjuk ke lokasi penginstalan JDK Anda.
- Sebaiknya gunakan lingkungan pengembangan (seperti Eclipse Java Neon atau IntelliJ Idea) untuk mengembangkan dan mengompilasi aplikasi Anda.
- Klien Git. Instal klien Git jika Anda belum menginstalnya.
- Plugin Compiler Apache Maven. Maven harus berada di jalur kerja Anda. Untuk menguji instalasi Apache Maven Anda, masukkan hal berikut:

```
$ mvn -version
```

Untuk memulai, buka Langkah 1: Siapkan Akun AWS dan Buat Pengguna Administrator.

Langkah 1: Siapkan Akun AWS dan Buat Pengguna Administrator

Mendaftar Akun AWS

Jika Anda tidak memiliki Akun AWS, selesaikan langkah-langkah berikut untuk membuatnya.

Untuk mendaftar Akun AWS

1. Buka https://portal.aws.amazon.com/billing/signup.
2. Ikuti petunjuk secara online.

   Anda akan diminta untuk menerima panggilan telepon dan memasukkan kode verifikasi pada keypad telepon sebagai bagian dari prosedur pendaftaran.

   Saat Anda mendaftar Akun AWS, Pengguna root akun AWS akan dibuat. Pengguna root memiliki akses ke semua Layanan AWS dan sumber daya dalam akun. Sebagai praktik terbaik keamanan, tetapkan akses administratif ke pengguna administratif, dan hanya gunakan pengguna root untuk melakukan tugas yang memerlukan akses pengguna root.

AWS akan mengirimkan email konfirmasi kepada Anda setelah proses pendaftaran selesai. Anda dapat melihat aktivitas akun saat ini dan mengelola akun dengan mengunjungi https://aws.amazon.com/ dan memilih Akun Saya.

Membuat pengguna administratif

Setelah mendaftar Akun AWS, amankan Pengguna root akun AWS, aktifkan AWS IAM Identity Center, dan buat sebuah pengguna administratif sehingga Anda tidak menggunakan pengguna root untuk tugas sehari-hari.

Mengamankan Pengguna root akun AWS Anda

1. Masuk ke AWS Management Console sebagai pemilik akun dengan memilih Pengguna root dan memasukkan alamat email Akun AWS Anda. Di halaman berikutnya, masukkan kata sandi Anda.

   Untuk bantuan masuk menggunakan pengguna root, lihat Masuk sebagai pengguna root dalam Panduan Pengguna AWS Sign-In.
2. Aktifkan autentikasi multi-faktor (MFA) untuk pengguna root Anda.

   Untuk petunjuknya, silakan lihat Mengaktifkan perangkat MFA virtual untuk pengguna root Akun AWS Anda (konsol) dalam Panduan Pengguna IAM.

Membuat pengguna administratif

1. Aktifkan Pusat Identitas IAM.

   Untuk mendapatkan petunjuk, silakan lihat Mengaktifkan AWS IAM Identity Center di Panduan Pengguna AWS IAM Identity Center.
2. Di Pusat Identitas IAM, berikan akses administratif ke sebuah pengguna administratif.

   Untuk mendapatkan tutorial tentang menggunakan Direktori Pusat Identitas IAM sebagai sumber identitas Anda, silakan lihat Mengonfigurasi akses pengguna dengan Direktori Pusat Identitas IAM default di Panduan Pengguna AWS IAM Identity Center.

Masuk sebagai pengguna administratif

- Untuk masuk dengan pengguna Pusat Identitas IAM, gunakan URL masuk yang dikirim ke alamat email Anda saat Anda membuat pengguna Pusat Identitas IAM.

Untuk bantuan masuk menggunakan pengguna Pusat Identitas IAM, lihat Masuk ke portal akses AWS dalam Panduan Pengguna AWS Sign-In.

Memberikan akses terprogram

Pengguna membutuhkan akses terprogram jika mereka ingin berinteraksi dengan AWS luar dari AWS Management Console. Cara memberikan akses terprogram bergantung pada jenis pengguna yang mengakses AWS.

Untuk memberi pengguna akses terprogram, pilih salah satu opsi berikut.

Pengguna mana yang membutuhkan akses terprogram?	Untuk	Oleh
Identitas tenaga kerja (Pengguna yang dikelola di Pusat Identitas IAM)	Gunakan kredensial sementara untuk menandatangani permintaan terprogram ke AWS CLI, SDK AWS, atau API AWS.	Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan. • Untuk AWS CLI, lihat Mengonfigurasi AWS CLI untuk menggunakan AWS IAM Identity Center di Panduan Pengguna AWS Command Line Interface. • Untuk SDK AWS, alat, dan API AWS, lihat Autentikasi Pusat Identitas IAM di Panduan Referensi SDK dan Alat AWS.
IAM	Gunakan kredensial sementara untuk menandatangani permintaan terprogram ke AWS CLI, SDK AWS, atau API AWS.	Mengikuti petunjuk dalam Menggunakan kredensial sementara dengan sumber daya AWS di Panduan Pengguna IAM.

Pengguna mana yang membutuhkan akses terprogram?	Untuk	Oleh
IAM	(Tidak direkomendasikan) Gunakan kredensial jangka panjang untuk menandatangani permintaan terprogram ke AWS CLI, SDK AWS, atau API AWS.	Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan. • Untuk AWS CLI, lihat Mengautentikasi menggunakan kredensial pengguna IAM di Panduan Pengguna AWS Command Line Interface. • Untuk SDK dan alat AWS, lihat Mengautentikasi menggunakan kredensial jangka panjang di Panduan Referensi SDK dan Alat AWS. • Untuk API AWS, lihat Mengelola kunci akses untuk pengguna IAM di Panduan Pengguna IAM.

Langkah Selanjutnya

Langkah 2: Siapkan AWS Command Line Interface (AWS CLI)

Langkah Selanjutnya

Langkah 2: Siapkan AWS Command Line Interface (AWS CLI)

Langkah 2: Siapkan AWS Command Line Interface (AWS CLI)

Pada langkah ini, Anda mengunduh dan mengonfigurasi AWS CLI untuk digunakan dengan Managed Service for Apache Flink.

> **Note**
>
> Latihan memulai dalam panduan ini mengasumsikan Anda menggunakan kredensial administrator (`adminuser`) di akun Anda untuk melakukan operasi.

> **Note**
>
> Jika sudah menginstal AWS CLI, Anda mungkin perlu meningkatkan untuk mendapatkan fungsionalitas terbaru. Untuk informasi selengkapnya, lihat Menginstal AWS Command Line Interface dalam Panduan Pengguna AWS Command Line Interface. Untuk memeriksa versi AWS CLI, jalankan perintah berikut.
>
> ```
> aws --version
> ```
>
> Latihan dalam tutorial ini memerlukan versi AWS CLI berikut atau yang lebih baru:
>
> ```
> aws-cli/1.16.63
> ```

Untuk mengatur AWS CLI

1. Unduh dan konfigurasikan AWS CLI. Untuk instruksinya, lihat topik berikut di AWS Command Line InterfacePanduan Pengguna:
   - Menginstal AWS Command Line Interface
   - Mengonfigurasi AWS CLI
2. Tambahkan profil bernama untuk pengguna administrator dalam file AWS CLI `config`. Anda dapat menggunakan profil ini saat menjalankan perintah AWS CLI. Untuk informasi selengkapnya tentang profil yang diberi nama, lihat Profil yang Diberi Nama dalam Panduan Pengguna AWS Command Line Interface.

   ```
   [profile adminuser]
   aws_access_key_id = adminuser access key ID
   aws_secret_access_key = adminuser secret access key
   region = aws-region
   ```

Untuk daftar AWS Wilayah yang tersedia, lihat Wilayah dan Titik Akhir di. Referensi Umum Amazon Web Services

> **Note**
>
> Kode dan perintah contoh dalam tutorial ini menggunakan Wilayah US West (Oregon). Untuk menggunakan Wilayah yang berbeda, ubah Wilayah dalam kode dan perintah untuk tutorial ini ke Wilayah yang ingin Anda gunakan.

3. Verifikasikan penyiapan dengan memasukkan perintah bantuan berikut pada prompt perintah.

```
aws help
```

Setelah Anda mengatur AWS akun danAWS CLI, Anda dapat mencoba latihan berikutnya, di mana Anda mengkonfigurasi aplikasi sampel dan menguji end-to-end pengaturan.

Langkah Selanjutnya

Langkah 3: Buat dan Jalankan Layanan Terkelola untuk aplikasi Apache Flink

Langkah 3: Buat dan Jalankan Layanan Terkelola untuk aplikasi Apache Flink

Dalam latihan ini, Anda membuat Layanan Terkelola untuk aplikasi Apache Flink dengan aliran data sebagai sumber dan wastafel.

Bagian ini berisi langkah-langkah berikut.

- Buat Dua Amazon Kinesis Data Streams
- Tulis Catatan Sampel ke Aliran Input
- Unduh dan Periksa Kode Java Streaming Apache Flink
- Kompilasi Kode Aplikasi
- Unggah Kode Java Streaming Apache Flink
- Buat dan Jalankan Managed Service untuk Apache Flink Application
- Langkah Selanjutnya

Buat Dua Amazon Kinesis Data Streams

Sebelum Anda membuat Layanan Terkelola untuk aplikasi Apache Flink untuk latihan ini, buat dua aliran data Kinesis (dan). `ExampleInputStream` `ExampleOutputStream` Aplikasi Anda menggunakan aliran ini untuk sumber aplikasi dan aliran tujuan.

Anda dapat membuat aliran ini menggunakan konsol Amazon Kinesis atau perintah AWS CLI berikut. Untuk instruksi konsol, lihat Membuat dan Memperbarui Aliran Data di Panduan Developer Amazon Kinesis Data Streams.

Untuk membuat aliran data AWS CLI

1. Untuk membuat aliran pertama (`ExampleInputStream`), gunakan perintah `create-stream` AWS CLI Amazon Kinesis berikut.

```
$ aws kinesis create-stream \
--stream-name ExampleInputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

2. Untuk membuat aliran kedua yang digunakan aplikasi untuk menulis output, jalankan perintah yang sama, yang mengubah nama aliran menjadi `ExampleOutputStream`.

```
$ aws kinesis create-stream \
--stream-name ExampleOutputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

Tulis Catatan Sampel ke Aliran Input

Di bagian ini, Anda menggunakan script Python untuk menulis catatan sampel ke aliran untuk diproses aplikasi.

> **Note**
>
> Bagian ini memerlukan AWS SDK for Python (Boto).

1. Buat file bernama `stock.py` dengan konten berikut:

```
import datetime
import json
import random
import boto3
STREAM_NAME = "ExampleInputStream"
def get_data():
    return {
        'event_time': datetime.datetime.now().isoformat(),
        'ticker': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'price': round(random.random() * 100, 2)}
def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(data),
            PartitionKey="partitionkey")
if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis', region_name='us-west-2'))
```

2. Selanjutnya dalam tutorial ini, Anda menjalankan skrip `stock.py` untuk mengirim data ke aplikasi.

```
$ python stock.py
```

Unduh dan Periksa Kode Java Streaming Apache Flink

Kode aplikasi Java untuk contoh ini tersedia dari GitHub. Untuk mengunduh kode aplikasi, lakukan hal berikut:

1. Kloning repositori jarak jauh menggunakan perintah berikut:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

2. Buka direktori `amazon-kinesis-data-analytics-java-examples/GettingStarted` tersebut.

Perhatikan hal tentang kode aplikasi berikut:

- File Project Object Model (pom.xml) berisi informasi tentang konfigurasi dan dependensi aplikasi, termasuk Layanan Terkelola untuk pustaka Apache Flink.
- File `BasicStreamingJob.java` berisi metode `main` yang menentukan fungsionalitas aplikasi.
- Aplikasi menggunakan sumber Kinesis untuk membaca dari aliran sumber. Cuplikan berikut ini membuat sumber Kinesis:

```
return env.addSource(new FlinkKinesisConsumer<>(inputStreamName,
                new SimpleStringSchema(), inputProperties));
```

- Aplikasi Anda membuat konektor sumber dan sink untuk mengakses sumber daya eksternal menggunakan objek `StreamExecutionEnvironment`.
- Aplikasi membuat konektor sumber dan sink menggunakan properti statis. Untuk menggunakan properti aplikasi dinamis, gunakan metode `createSourceFromApplicationProperties` dan `createSinkFromApplicationProperties` untuk membuat konektor. Metode ini membaca properti aplikasi untuk mengonfigurasi konektor.

  Untuk informasi selengkapnya tentang properti runtime, lihat Properti Runtime.

Kompilasi Kode Aplikasi

Di bagian ini, Anda menggunakan compiler Apache Maven untuk membuat kode Java untuk aplikasi. Untuk informasi tentang menginstal Apache Maven dan Java Development Kit (JDK), lihat Prasyarat untuk Menyelesaikan Latihan.

Untuk mengompikasi kode aplikasi

1. Untuk menggunakan kode aplikasi Anda, Anda mengompilasi dan mengemasnya ke dalam file JAR. Anda dapat mengompilasi dan mengemas kode Anda dengan salah satu dari dua cara:

   - Gunakan alat Maven baris perintah. Buat file JAR Anda dengan menjalankan perintah berikut di direktori yang berisi file `pom.xml`:

     ```
     mvn package -Dflink.version=1.13.2
     ```

   - Menyiapkan lingkungan pengembangan Anda. Lihat dokumentasi lingkungan pengembangan Anda untuk detail.

> **Note**
>
> Kode sumber yang disediakan bergantung pada pustaka dari Java 11.

Anda dapat mengunggah paket Anda sebagai file JAR, atau Anda dapat mengompresi paket Anda dan mengunggahnya sebagai file ZIP. Jika Anda membuat aplikasi Anda menggunakan AWS CLI, Anda menentukan tipe konten kode Anda (JAR atau ZIP).

2. Jika ada kesalahan saat mengompilasi, pastikan variabel lingkungan `JAVA_HOME`Anda diatur dengan benar.

Jika aplikasi berhasil mengompilasi, file berikut dibuat:

```
target/aws-kinesis-analytics-java-apps-1.0.jar
```

Unggah Kode Java Streaming Apache Flink

Pada bagian ini, Anda membuat bucket Amazon Simple Storage Service (Amazon S3) dan mengunggah kode aplikasi Anda.

Untuk mengunggah kode aplikasi

1. Buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.
2. Pilih Buat bucket.
3. Masukkan **`ka-app-code-<username>`** di bidang Bucket name (Nama bucket). Tambahkan sufiks ke nama bucket, seperti nama pengguna Anda, untuk membuatnya unik secara global. Pilih Next (Selanjutnya).
4. Di langkah Konfigurasi opsi, jangan ubah pengaturan, dan pilih Next (Selanjutnya).
5. Di langkah Atur izin, jangan ubah pengaturan, dan pilih Next (Selanjutnya).
6. Pilih Create bucket (Buat bucket).
7. Di konsol Amazon S3, pilih bucket ka-app-code -, dan pilih Unggah. <username>
8. Di langkah Pilih file, pilih Add files (Tambahkan berkas). Navigasikan ke file `aws-kinesis-analytics-java-apps-1.0.jar` yang Anda buat di langkah sebelumnya. Pilih Berikutnya.
9. Anda tidak perlu mengubah pengaturan apa pun untuk objek, jadi pilih Upload (Unggah).

Kode aplikasi Anda sekarang disimpan di bucket Amazon S3 yang dapat diakses aplikasi Anda.

Buat dan Jalankan Managed Service untuk Apache Flink Application

Anda dapat membuat dan menjalankan Layanan Terkelola untuk aplikasi Apache Flink menggunakan konsol atau aplikasi. AWS CLI

> **Note**
>
> Saat Anda membuat aplikasi menggunakan konsol, sumber daya AWS Identity and Access Management (IAM) dan Amazon CloudWatch Logs dibuat untuk Anda. Saat Anda membuat aplikasi menggunakan AWS CLI, Anda membuat sumber daya ini secara terpisah.

Topik

- Buat dan Jalankan Aplikasi (Konsol)
- Buat dan Jalankan Aplikasi (AWS CLI)

Buat dan Jalankan Aplikasi (Konsol)

Ikuti langkah-langkah ini untuk membuat, mengonfigurasi, memperbarui, dan menjalankan aplikasi menggunakan konsol.

Buat Aplikasi

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/flink
2. Pada dashboard Managed Service for Apache Flink, pilih Create Analytics Application.
3. Pada Layanan Terkelola untuk Apache Flink - Buat halaman aplikasi, berikan detail aplikasi sebagai berikut:
   - Untuk Application name (Nama aplikasi), masukkan **`MyApplication`**.
   - Untuk Description (Deskripsi), masukkan **`My java test app`**.
   - Untuk Runtime, pilih Apache Flink.
   - Biarkan versi pulldown sebagai Apache Flink versi 1.13.
4. Untuk Access permissions (Izin akses), pilih Create / update IAM role **`kinesis-analytics-MyApplication-us-west-2`** (Buat/perbarui IAM role).
5. Pilih Create application (Buat aplikasi).

> **Note**
>
> Saat membuat aplikasi Managed Service for Apache Flink menggunakan konsol, Anda memiliki opsi untuk membuat peran dan kebijakan IAM untuk aplikasi Anda. Aplikasi Anda menggunakan peran dan kebijakan ini untuk mengakses sumber daya dependen. Sumber daya IAM ini diberi nama menggunakan nama aplikasi dan Wilayah sebagai berikut:
>
> - Kebijakan: `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Peran: `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Edit Kebijakan IAM

Edit kebijakan IAM untuk menambahkan izin mengakses Kinesis data streams.

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pilih Policies (Kebijakan). Pilih kebijakan **`kinesis-analytics-service-MyApplication-us-west-2`** yang dibuat konsol untuk Anda di bagian sebelumnya.
3. Di halaman Ringkasan, pilih Edit policy (Edit kebijakan). Pilih tab JSON.
4. Tambahkan bagian yang disorot dari contoh kebijakan berikut ke kebijakan. Ganti ID akun sampel (*`012345678901`*) dengan ID akun Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ka-app-code-username/aws-kinesis-analytics-java-
apps-1.0.jar"
            ]
        },
        {
            "Sid": "DescribeLogGroups",
            "Effect": "Allow",
```

```
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
            ]
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
```

```
}
```

Konfigurasikan Aplikasi

1. Pada MyApplicationhalaman, pilih Konfigurasi.
2. Di halaman Konfigurasikan aplikasi, berikan Code location (Lokasi kode):
   - Untuk Bucket Amazon S3, masukkan **`ka-app-code-<username>`**.
   - Untuk Jalur ke objek Amazon S3, masukkan **`aws-kinesis-analytics-java-apps-1.0.jar`**.
3. Di bawah Akses ke sumber daya aplikasi, untuk Access permissions (Izin akses), pilih Create / update IAM role **`kinesis-analytics-MyApplication-us-west-2`** (Pilih/perbarui IAM role).
4. Masukkan yang berikut ini:

ID Grup	Kunci	Nilai
`ProducerConfigProperties`	**`flink.inputstream.initpos`**	**`LATEST`**
`ProducerConfigProperties`	**`aws.region`**	**`us-west-2`**
`ProducerConfigProperties`	**`AggregationEnabled`**	**`false`**

5. Di bawah Pemantauan, pastikan Memantau tingkat metrik diatur ke Aplikasi.
6. Untuk CloudWatch logging, pilih kotak centang Aktifkan.
7. Pilih Perbarui.

> **Note**
>
> Saat Anda memilih untuk mengaktifkan CloudWatch pencatatan Amazon, Layanan Terkelola untuk Apache Flink membuat grup log dan aliran log untuk Anda. Nama-nama sumber daya ini adalah sebagai berikut:

- Grup log: `/aws/kinesis-analytics/MyApplication`
- Aliran log: `kinesis-analytics-log-stream`

Jalankan Aplikasi

Grafik pekerjaan Flink dapat dilihat dengan menjalankan aplikasi, membuka dasbor Apache Flink, dan memilih pekerjaan Flink yang diinginkan.

Hentikan Aplikasi

Pada MyApplicationhalaman, pilih Berhenti. Konfirmasikan tindakan.

Perbarui Aplikasi

Dengan menggunakan konsol, Anda dapat memperbarui pengaturan aplikasi seperti properti aplikasi, pengaturan pemantauan, dan lokasi atau nama file dari JAR aplikasi. Anda juga dapat memuat ulang aplikasi JAR dari bucket Amazon S3 jika Anda perlu memperbarui kode aplikasi.

Pada MyApplicationhalaman, pilih Konfigurasi. Perbarui pengaturan aplikasi dan pilih Update (Perbarui).

Buat dan Jalankan Aplikasi (AWS CLI)

Di bagian ini, Anda menggunakan AWS CLI untuk membuat dan menjalankan aplikasi Managed Service for Apache Flink. Managed Service untuk Apache Flink menggunakan `kinesisanalyticsv2` AWS CLI perintah untuk membuat dan berinteraksi dengan Managed Service untuk aplikasi Apache Flink.

Membuat Kebijakan Izin

> **Note**
>
> Anda harus membuat kebijakan izin dan peran untuk aplikasi Anda. Jika Anda tidak membuat sumber daya IAM ini, aplikasi Anda tidak dapat mengakses data dan aliran log.

Pertama, Anda membuat kebijakan izin dengan dua pernyataan: satu yang memberikan izin untuk tindakan `read` di aliran sumber, dan lainnya yang memberikan izin untuk tindakan `write` di aliran

sink. Anda selanjutnya melampirkan kebijakan ke IAM role (yang Anda buat di bagian berikutnya). Jadi, ketika Layanan Terkelola untuk Apache Flink mengasumsikan peran tersebut, layanan memiliki izin yang diperlukan untuk membaca dari aliran sumber dan menulis ke aliran wastafel.

Gunakan kode berikut untuk membuat kebijakan izin `AKReadSourceStreamWriteSinkStream`. Ganti *`username`* dengan nama pengguna yang Anda gunakan untuk membuat bucket Amazon S3 untuk menyimpan kode aplikasi. Ganti ID akun di Amazon Resource Names (ARN) (*`012345678901`*) dengan ID akun Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": ["arn:aws:s3:::ka-app-code-username",
                "arn:aws:s3:::ka-app-code-username/*"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

Untuk step-by-step petunjuk membuat kebijakan izin, lihat Tutorial: Membuat dan Melampirkan Kebijakan Terkelola Pelanggan Pertama Anda di Panduan Pengguna IAM.

> **Note**
>
> Untuk mengakses layanan Amazon lainnya, Anda dapat menggunakan AWS SDK for Java. Layanan Terkelola untuk Apache Flink secara otomatis menyetel kredensyal yang diperlukan oleh SDK ke peran IAM eksekusi layanan yang terkait dengan aplikasi Anda. Tidak ada langkah tambahan yang diperlukan.

Buat IAM Role

Di bagian ini, Anda membuat peran IAM yang dapat diasumsikan oleh aplikasi Managed Service for Apache Flink untuk membaca aliran sumber dan menulis ke aliran sink.

Layanan Terkelola untuk Apache Flink tidak dapat mengakses aliran Anda tanpa izin. Anda memberikan izin ini melalui IAM role. Setiap IAM role memiliki dua kebijakan yang dilampirkan. Kebijakan kepercayaan memberikan izin Layanan Terkelola untuk Apache Flink untuk mengambil peran, dan kebijakan izin menentukan apa yang dapat dilakukan Layanan Terkelola untuk Apache Flink setelah mengambil peran.

Anda melampirkan kebijakan izin yang Anda buat di bagian sebelumnya ke peran ini.

Untuk membuat IAM role

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Dalam panel navigasi, pilih Roles (Peran), Create role (Buat Peran).
3. Di bawah Pilih tipe identitas tepercaya, pilih Layanan AWS. Di bawah Pilih layanan yang akan menggunakan peran ini, pilih Kinesis. Di bawah Pilih kasus penggunaan Anda, pilih Analitik Kinesis.

   Pilih Selanjutnya: Izin.
4. Di halaman Lampirkan kebijakan izin, pilih Next: Review (Selanjutnya: Tinjau). Anda melampirkan kebijakan izin setelah Anda membuat peran tersebut.
5. Di halaman Buat peran, masukkan **`MF-stream-rw-role`** untuk Role name (Nama peran). Pilih Create role (Buat peran).

   Sekarang Anda sudah membuat IAM role baru yang disebut `MF-stream-rw-role`. Berikutnya, Anda memperbarui kebijakan kepercayaan dan izin untuk peran.
6. Lampirkan kebijakan izin ke peran tersebut.

> **Note**
>
> Untuk latihan ini, Managed Service for Apache Flink mengasumsikan peran ini untuk membaca data dari aliran data Kinesis (sumber) dan menulis output ke aliran data Kinesis lain. Jadi, Anda melampirkan kebijakan yang Anda buat di langkah sebelumnya, the section called “Membuat Kebijakan Izin”.

a. Di halaman Ringkasan, pilih tab Permissions (Izin).

b. Pilih Attach Policies (Lampirkan Kebijakan).

c. Di kotak pencarian, masukkan **`AKReadSourceStreamWriteSinkStream`** (kebijakan yang Anda buat bagian sebelumnya).

d. Pilih ReadSourceStreamWriteSinkStream kebijakan AK, dan pilih Lampirkan kebijakan.

Anda sekarang telah membuat peran eksekusi layanan yang digunakan aplikasi Anda untuk mengakses sumber daya. Buat catatan ARN peran baru.

Untuk step-by-step petunjuk cara membuat peran, lihat Membuat Peran IAM (Konsol) di Panduan Pengguna IAM.

Buat Managed Service untuk Apache Flink Application

1. Simpan kode JSON berikut ke file bernama `create_request.json`. Ganti ARN peran sampel dengan ARN untuk peran yang Anda buat sebelumnya. Ganti sufiks ARN bucket (*`username`*) dengan sufiks yang Anda pilih di bagian sebelumnya. Ganti ID akun sampel (*`012345678901`*) di peran eksekusi layanan dengan ID akun Anda.

```
{
    "ApplicationName": "test",
    "ApplicationDescription": "my java test app",
    "RuntimeEnvironment": "FLINK-1_15",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-stream-rw-role",
    "ApplicationConfiguration": {
        "ApplicationCodeConfiguration": {
            "CodeContent": {
                "S3ContentLocation": {
                    "BucketARN": "arn:aws:s3:::ka-app-code-username",
                    "FileKey": "aws-kinesis-analytics-java-apps-1.0.jar"
```

```
                }
            },
            "CodeContentType": "ZIPFILE"
        },
        "EnvironmentProperties":  {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "flink.stream.initpos" : "LATEST",
                    "aws.region" : "us-west-2",
                    "AggregationEnabled" : "false"
               }
            },
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2"
               }
            }
         ]
      }
    }
}
```

2. Jalankan tindakan `CreateApplication` dengan permintaan sebelumnya untuk membuat aplikasi:

```
aws kinesisanalyticsv2 create-application --cli-input-json file://
create_request.json
```

Aplikasi ini sekarang dibuat. Anda memulai aplikasi di langkah berikutnya.

Mulai Aplikasi

Di bagian ini, Anda menggunakan tindakan `StartApplication` untuk memulai aplikasi.

Untuk memulai aplikasi

1. Simpan kode JSON berikut ke file bernama `start_request.json`.

```
{
    "ApplicationName": "test",
    "RunConfiguration": {
        "ApplicationRestoreConfiguration": {
         "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"
         }
    }
}
```

2. Jalankan tindakan StartApplication dengan permintaan sebelumnya untuk memulai aplikasi:

```
aws kinesisanalyticsv2 start-application --cli-input-json file://start_request.json
```

Aplikasi sekarang berjalan. Anda dapat memeriksa metrik Layanan Terkelola untuk Apache Flink di CloudWatch konsol Amazon untuk memverifikasi bahwa aplikasi berfungsi.

Hentikan Aplikasi

Di bagian ini, Anda menggunakan tindakan StopApplication untuk menghentikan aplikasi.

Untuk menghentikan aplikasi

1. Simpan kode JSON berikut ke file bernama `stop_request.json`.

```
{
   "ApplicationName": "test"
}
```

2. Jalankan tindakan StopApplication dengan permintaan berikut untuk menghentikan aplikasi:

```
aws kinesisanalyticsv2 stop-application --cli-input-json file://stop_request.json
```

Aplikasi sekarang dihentikan.

Tambahkan Opsi CloudWatch Logging

Anda dapat menggunakan AWS CLI untuk menambahkan aliran CloudWatch log Amazon ke aplikasi Anda. Untuk informasi tentang menggunakan CloudWatch Log dengan aplikasi Anda, lihatthe section called “Menyiapkan Pencatatan”.

Perbarui Properti Lingkungan

Di bagian ini, Anda menggunakan tindakan `UpdateApplication` untuk mengubah properti lingkungan untuk aplikasi tanpa mengompilasi ulang kode aplikasi. Dalam contoh ini, Anda mengubah Wilayah aliran sumber dan tujuan.

Untuk memperbarui properti lingkungan untuk aplikasi

1. Simpan kode JSON berikut ke file bernama `update_properties_request.json`.

```
{"ApplicationName": "test",
   "CurrentApplicationVersionId": 1,
   "ApplicationConfigurationUpdate": {
      "EnvironmentPropertyUpdates": {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "flink.stream.initpos" : "LATEST",
                    "aws.region" : "us-west-2",
                    "AggregationEnabled" : "false"
               }
            },
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2"
               }
            }
         ]
      }
   }
}
```

2. Jalankan tindakan `UpdateApplication` dengan permintaan sebelumnya untuk memperbarui properti lingkungan:

```
aws kinesisanalyticsv2 update-application --cli-input-json file://
update_properties_request.json
```

Perbarui Kode Aplikasi

Ketika Anda perlu memperbarui kode aplikasi Anda dengan versi baru paket kode Anda, Anda menggunakan tindakan `UpdateApplication` AWS CLI.

> **Note**
>
> Untuk memuat versi baru kode aplikasi dengan nama file yang sama, Anda harus menentukan versi objek baru. Untuk informasi selengkapnya tentang menggunakan versi objek Amazon S3, lihat Mengaktifkan dan Menonaktifkan Versioning.

Untuk menggunakan AWS CLI, hapus paket kode Anda sebelumnya dari bucket Amazon S3 Anda, unggah versi baru, dan panggil `UpdateApplication`, yang menentukan bucket Amazon S3 dan nama objek yang sama, serta versi objek baru. Aplikasi akan memulai ulang dengan paket kode baru.

Permintaan sampel berikut untuk tindakan `UpdateApplication` memuat ulang kode aplikasi dan memulai ulang aplikasi. Perbarui `CurrentApplicationVersionId` ke versi aplikasi saat ini. Anda dapat memeriksa versi aplikasi saat ini menggunakan tindakan `ListApplications` atau `DescribeApplication`. Perbarui sufiks nama bucket (*<username>*) dengan sufiks yang Anda pilih di bagian the section called “Buat Dua Amazon Kinesis Data Streams”.

```
{
    "ApplicationName": "test",
    "CurrentApplicationVersionId": 1,
    "ApplicationConfigurationUpdate": {
        "ApplicationCodeConfigurationUpdate": {
            "CodeContentUpdate": {
                "S3ContentLocationUpdate": {
                    "BucketARNUpdate": "arn:aws:s3:::ka-app-code-username",
                    "FileKeyUpdate": "aws-kinesis-analytics-java-apps-1.0.jar",
                    "ObjectVersionUpdate": "SAMPLEUehYngP87ex1nzYIGYgfhypvDU"
                }
            }
        }
```

```
    }
}
```

Langkah Selanjutnya

[Langkah 4: Bersihkan Sumber Daya AWS]

Langkah 4: Bersihkan Sumber Daya AWS

Bagian ini mencakup prosedur untuk membersihkan sumber daya AWS yang dibuat dalam tutorial Memulai.

Topik ini berisi bagian-bagian berikut:

- Hapus Layanan Terkelola Anda untuk Aplikasi Apache Flink
- Hapus Kinesis Data Streams Anda
- Hapus Objek dan Bucket Amazon S3 Anda
- Hapus Sumber Daya IAM Anda
- Hapus CloudWatch Sumber Daya Anda
- Langkah Selanjutnya

Hapus Layanan Terkelola Anda untuk Aplikasi Apache Flink

1. Buka konsol Kinesis di https://console.aws.amazon.com/kinesis.
2. Di panel Managed Service for Apache Flink, pilih. MyApplication
3. Di halaman aplikasi, pilih Delete (Hapus), lalu konfirmasikan penghapusan.

Hapus Kinesis Data Streams Anda

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/flink
2. Di panel Kinesis Data Streams, pilih. ExampleInputStream
3. Di ExampleInputStreamhalaman, pilih Hapus Stream Kinesis dan kemudian konfirmasikan penghapusan.
4. Di halaman Kinesis streams, pilih, pilih Tindakan ExampleOutputStream, pilih Hapus, lalu konfirmasikan penghapusan.

Hapus Objek dan Bucket Amazon S3 Anda

1. Buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.
2. Pilih ka-app-code- ember. <username>
3. Pilih Delete (Hapus), lalu masukkan nama bucket untuk mengonfirmasi penghapusan.

Hapus Sumber Daya IAM Anda

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di bilah navigasi, pilih Policies (Kebijakan).
3. Di kontrol filter, masukkan kinesis.
4. Pilih kebijakan kinesis-analytics-service- MyApplication -us-west-2.
5. Pilih Policy Actions (Tindakan Kebijakan), lalu pilih Delete (Hapus).
6. Di bilah navigasi, pilih Roles (Peran).
7. Pilih peran kinesis-analytics- -us-west-2. MyApplication
8. Pilih Delete role (Hapus peran), lalu konfirmasi penghapusan.

Hapus CloudWatch Sumber Daya Anda

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di bilah navigasi, pilih Logs.
3. Pilih grup log MyApplication/aws/kinesis-analytics/.
4. Pilih Delete Log Group (Hapus Grup Log), lalu konfirmasi penghapusan.

Langkah Selanjutnya

Langkah 5: Langkah selanjutnya

Langkah 5: Langkah selanjutnya

Setelah Anda membuat dan menjalankan Managed Service dasar untuk aplikasi Apache Flink, lihat sumber daya berikut untuk solusi Managed Service for Apache Flink yang lebih canggih.

- Solusi Data Streaming AWS untuk Amazon Kinesis: Solusi Data Streaming AWS untuk Amazon Kinesis secara otomatis mengonfigurasi layanan AWS yang diperlukan untuk dengan mudah

menangkap, menyimpan, memproses, dan mengirimkan data streaming. Solusi ini menyediakan beberapa opsi untuk memecahkan kasus penggunaan data streaming. Layanan Terkelola untuk opsi Apache Flink menyediakan contoh ETL end-to-end streaming yang menunjukkan aplikasi dunia nyata yang menjalankan operasi analitis pada data taksi New York yang disimulasikan. Solusinya menyiapkan semua AWS sumber daya yang diperlukan seperti peran dan kebijakan IAM, CloudWatch dasbor, dan CloudWatch alarm.

- AWSSolusi Data Streaming untuk Amazon MSK: Solusi Data AWS Streaming untuk Amazon MSK menyediakan AWS CloudFormation template tempat data mengalir melalui produsen, penyimpanan streaming, konsumen, dan tujuan.
- Clickstream Lab dengan Apache Flink dan Apache Kafka: Lab ujung ke ujung untuk kasus penggunaan clickstream menggunakan Amazon Managed Streaming untuk Apache Kafka Kafka untuk penyimpanan streaming dan Layanan Terkelola untuk Apache Flink untuk aplikasi Apache Flink untuk pemrosesan streaming.
- Amazon Managed Service untuk Apache Flink Workshop: Dalam lokakarya ini, Anda membangun arsitektur end-to-end streaming untuk menelan, menganalisis, dan memvisualisasikan data streaming dalam waktu dekat. Anda mulai meningkatkan operasi perusahaan taksi di Kota New York. Anda menganalisis data telemetri armada taksi di Kota New York hampir secara langsung untuk mengoptimalkan operasi armada mereka.
- Layanan Terkelola untuk Apache Flink: Contoh: Bagian Panduan Pengembang ini memberikan contoh membuat dan bekerja dengan aplikasi di Managed Service untuk Apache Flink. Mereka menyertakan kode contoh dan step-by-step instruksi untuk membantu Anda membuat Layanan Terkelola untuk aplikasi Apache Flink dan menguji hasil Anda.
- Belajar Flink: Pelatihan Langsung: Pelatihan pengenalan resmi Apache Flink yang membuat Anda mulai menulis ETL streaming yang dapat diskalakan, analitik, dan aplikasi yang didorong peristiwa.

> **Note**
>
> Ketahuilah bahwa Managed Service for Apache Flink tidak mendukung versi Apache Flink (1.12) yang digunakan dalam pelatihan ini. Anda dapat menggunakan Flink 1.15.2 di Flink Managed Service untuk Apache Flink.

Memulai: Flink 1.11.1

Topik ini berisi versi Memulai (DataStream API) Tutorial yang menggunakan Apache Flink 1.11.1.

Bagian ini memperkenalkan Anda pada konsep dasar Managed Service untuk Apache Flink dan API. DataStream Ini menjelaskan opsi yang tersedia untuk membuat dan menguji aplikasi Anda. Ini juga memberikan petunjuk untuk menginstal alat yang diperlukan untuk menyelesaikan tutorial dalam panduan ini dan untuk membuat aplikasi pertama Anda.

Topik

Komponen Layanan Terkelola untuk Aplikasi Apache Flink

Untuk memproses data, Managed Service untuk aplikasi Apache Flink Anda menggunakan aplikasi Java/Apache Maven atau Scala yang memproses input dan menghasilkan output menggunakan runtime Apache Flink.

Layanan Terkelola untuk aplikasi Apache Flink memiliki komponen-komponen berikut:

- Properti runtime: Anda dapat menggunakan properti runtime untuk mengonfigurasi aplikasi Anda tanpa mengompilasi ulang kode aplikasi Anda.
- Source (Sumber): Aplikasi mengonsumsi data menggunakan sumber. Konektor sumber membaca data dari Kinesis data stream, bucket Amazon S3, dll. Untuk informasi selengkapnya, lihat Sumber.
- Operators (Operator): Aplikasi memproses data menggunakan satu atau beberapa operator. Operator dapat mengubah, memperkaya, atau menggabungkan data. Untuk informasi selengkapnya, lihat DataStream Operator API.
- Sink: Aplikasi menghasilkan data ke sumber eksternal menggunakan sink. Konektor sink menulis data ke aliran data Kinesis, aliran Firehose Data Kinesis, bucket Amazon S3, dll. Untuk informasi selengkapnya, lihat Sink.

Setelah Anda membuat, mengompilasi, dan mengemas kode aplikasi Anda, Anda mengunggah paket kode ke bucket Amazon Simple Storage Service (Amazon S3). Anda kemudian membuat

Layanan Terkelola untuk aplikasi Apache Flink. Anda meneruskan di lokasi paket kode, Kinesis data stream sebagai sumber data streaming, dan biasanya lokasi streaming atau file yang menerima data yang diproses dari aplikasi.

Prasyarat untuk Menyelesaikan Latihan

Untuk menyelesaikan langkah-langkah di panduan ini, Anda harus memiliki hal-hal berikut:

- Java Development Kit (JDK) versi 11. Atur variabel lingkungan `JAVA_HOME` untuk menunjuk ke lokasi penginstalan JDK Anda.
- Sebaiknya gunakan lingkungan pengembangan (seperti Eclipse Java Neon atau IntelliJ Idea) untuk mengembangkan dan mengompilasi aplikasi Anda.
- Klien Git. Instal klien Git jika Anda belum menginstalnya.
- Plugin Compiler Apache Maven. Maven harus berada di jalur kerja Anda. Untuk menguji instalasi Apache Maven Anda, masukkan hal berikut:

```
$ mvn -version
```

Untuk memulai, buka Langkah 1: Siapkan Akun AWS dan Buat Pengguna Administrator.

Langkah 1: Siapkan Akun AWS dan Buat Pengguna Administrator

Mendaftar Akun AWS

Jika Anda tidak memiliki Akun AWS, selesaikan langkah-langkah berikut untuk membuatnya.

Untuk mendaftar Akun AWS

1. Buka https://portal.aws.amazon.com/billing/signup.
2. Ikuti petunjuk secara online.

   Anda akan diminta untuk menerima panggilan telepon dan memasukkan kode verifikasi pada keypad telepon sebagai bagian dari prosedur pendaftaran.

   Saat Anda mendaftar Akun AWS, Pengguna root akun AWS akan dibuat. Pengguna root memiliki akses ke semua Layanan AWS dan sumber daya dalam akun. Sebagai praktik terbaik keamanan, tetapkan akses administratif ke pengguna administratif, dan hanya gunakan pengguna root untuk melakukan tugas yang memerlukan akses pengguna root.

AWS akan mengirimkan email konfirmasi kepada Anda setelah proses pendaftaran selesai. Anda dapat melihat aktivitas akun saat ini dan mengelola akun dengan mengunjungi https://aws.amazon.com/ dan memilih Akun Saya.

Membuat pengguna administratif

Setelah mendaftar Akun AWS, amankan Pengguna root akun AWS, aktifkan AWS IAM Identity Center, dan buat sebuah pengguna administratif sehingga Anda tidak menggunakan pengguna root untuk tugas sehari-hari.

Mengamankan Pengguna root akun AWS Anda

1. Masuk ke AWS Management Console sebagai pemilik akun dengan memilih Pengguna root dan memasukkan alamat email Akun AWS Anda. Di halaman berikutnya, masukkan kata sandi Anda.

   Untuk bantuan masuk menggunakan pengguna root, lihat Masuk sebagai pengguna root dalam Panduan Pengguna AWS Sign-In.
2. Aktifkan autentikasi multi-faktor (MFA) untuk pengguna root Anda.

   Untuk petunjuknya, silakan lihat Mengaktifkan perangkat MFA virtual untuk pengguna root Akun AWS Anda (konsol) dalam Panduan Pengguna IAM.

Membuat pengguna administratif

1. Aktifkan Pusat Identitas IAM.

   Untuk mendapatkan petunjuk, silakan lihat Mengaktifkan AWS IAM Identity Center di Panduan Pengguna AWS IAM Identity Center.
2. Di Pusat Identitas IAM, berikan akses administratif ke sebuah pengguna administratif.

   Untuk mendapatkan tutorial tentang menggunakan Direktori Pusat Identitas IAM sebagai sumber identitas Anda, silakan lihat Mengonfigurasi akses pengguna dengan Direktori Pusat Identitas IAM default di Panduan Pengguna AWS IAM Identity Center.

Masuk sebagai pengguna administratif

- Untuk masuk dengan pengguna Pusat Identitas IAM, gunakan URL masuk yang dikirim ke alamat email Anda saat Anda membuat pengguna Pusat Identitas IAM.

Untuk bantuan masuk menggunakan pengguna Pusat Identitas IAM, lihat Masuk ke portal akses AWS dalam Panduan Pengguna AWS Sign-In.

Memberikan akses terprogram

Pengguna membutuhkan akses terprogram jika mereka ingin berinteraksi dengan AWS luar dari AWS Management Console. Cara memberikan akses terprogram bergantung pada jenis pengguna yang mengakses AWS.

Untuk memberi pengguna akses terprogram, pilih salah satu opsi berikut.

Pengguna mana yang membutuhkan akses terprogram?	Untuk	Oleh
Identitas tenaga kerja (Pengguna yang dikelola di Pusat Identitas IAM)	Gunakan kredensial sementara untuk menandatangani permintaan terprogram ke AWS CLI, SDK AWS, atau API AWS.	Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan. • Untuk AWS CLI, lihat Mengonfigurasi AWS CLI untuk menggunakan AWS IAM Identity Center di Panduan Pengguna AWS Command Line Interface. • Untuk SDK AWS, alat, dan API AWS, lihat Autentikasi Pusat Identitas IAM di Panduan Referensi SDK dan Alat AWS.
IAM	Gunakan kredensial sementara untuk menandatangani permintaan terprogram ke AWS CLI, SDK AWS, atau API AWS.	Mengikuti petunjuk dalam Menggunakan kredensial sementara dengan sumber daya AWS di Panduan Pengguna IAM.

Pengguna mana yang membutuhkan akses terprogram?	Untuk	Oleh
IAM	(Tidak direkomendasikan) Gunakan kredensial jangka panjang untuk menandatangani permintaan terprogram ke AWS CLI, SDK AWS, atau API AWS.	Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan. • Untuk AWS CLI, lihat Mengautentikasi menggunakan kredensial pengguna IAM di Panduan Pengguna AWS Command Line Interface. • Untuk SDK dan alat AWS, lihat Mengautentikasi menggunakan kredensial jangka panjang di Panduan Referensi SDK dan Alat AWS. • Untuk API AWS, lihat Mengelola kunci akses untuk pengguna IAM di Panduan Pengguna IAM.

Langkah Selanjutnya

Langkah 2: Siapkan AWS Command Line Interface (AWS CLI)

Langkah 2: Siapkan AWS Command Line Interface (AWS CLI)

Pada langkah ini, Anda mengunduh dan mengonfigurasi AWS CLI untuk digunakan dengan Managed Service for Apache Flink.

> **Note**
>
> Latihan memulai dalam panduan ini mengasumsikan Anda menggunakan kredensial administrator (`adminuser`) di akun Anda untuk melakukan operasi.

> **Note**
>
> Jika sudah menginstal AWS CLI, Anda mungkin perlu meningkatkan untuk mendapatkan fungsionalitas terbaru. Untuk informasi selengkapnya, lihat Menginstal AWS Command Line Interface dalam Panduan Pengguna AWS Command Line Interface. Untuk memeriksa versi AWS CLI, jalankan perintah berikut.
>
> ```
> aws --version
> ```
>
> Latihan dalam tutorial ini memerlukan versi AWS CLI berikut atau yang lebih baru:
>
> ```
> aws-cli/1.16.63
> ```

Untuk mengatur AWS CLI

1. Unduh dan konfigurasikan AWS CLI. Untuk instruksinya, lihat topik berikut di AWS Command Line InterfacePanduan Pengguna:
   - Menginstal AWS Command Line Interface
   - Mengonfigurasi AWS CLI
2. Tambahkan profil bernama untuk pengguna administrator dalam file AWS CLI `config`. Anda dapat menggunakan profil ini saat menjalankan perintah AWS CLI. Untuk informasi selengkapnya tentang profil yang diberi nama, lihat Profil yang Diberi Nama dalam Panduan Pengguna AWS Command Line Interface.

   ```
   [profile adminuser]
   aws_access_key_id = adminuser access key ID
   aws_secret_access_key = adminuser secret access key
   region = aws-region
   ```

Untuk daftar AWS Wilayah yang tersedia, lihat Wilayah dan Titik Akhir di. Referensi Umum Amazon Web Services

> **Note**
>
> Kode dan perintah contoh dalam tutorial ini menggunakan Wilayah US West (Oregon). Untuk menggunakan Wilayah yang berbeda, ubah Wilayah dalam kode dan perintah untuk tutorial ini ke Wilayah yang ingin Anda gunakan.

3. Verifikasikan penyiapan dengan memasukkan perintah bantuan berikut pada prompt perintah.

```
aws help
```

Setelah Anda mengatur AWS akun danAWS CLI, Anda dapat mencoba latihan berikutnya, di mana Anda mengkonfigurasi aplikasi sampel dan menguji end-to-end pengaturan.

Langkah Selanjutnya

Langkah 3: Buat dan Jalankan Layanan Terkelola untuk aplikasi Apache Flink

Langkah 3: Buat dan Jalankan Layanan Terkelola untuk aplikasi Apache Flink

Dalam latihan ini, Anda membuat Layanan Terkelola untuk aplikasi Apache Flink dengan aliran data sebagai sumber dan wastafel.

Bagian ini berisi langkah-langkah berikut.

- Buat Dua Amazon Kinesis Data Streams
- Tulis Catatan Sampel ke Aliran Input
- Unduh dan Periksa Kode Java Streaming Apache Flink
- Kompilasi Kode Aplikasi
- Unggah Kode Java Streaming Apache Flink
- Buat dan Jalankan Managed Service untuk Apache Flink Application
- Langkah Selanjutnya

Buat Dua Amazon Kinesis Data Streams

Sebelum Anda membuat Layanan Terkelola untuk aplikasi Apache Flink untuk latihan ini, buat dua aliran data Kinesis (dan). `ExampleInputStream` `ExampleOutputStream` Aplikasi Anda menggunakan aliran ini untuk sumber aplikasi dan aliran tujuan.

Anda dapat membuat aliran ini menggunakan konsol Amazon Kinesis atau perintah AWS CLI berikut. Untuk instruksi konsol, lihat Membuat dan Memperbarui Aliran Data di Panduan Developer Amazon Kinesis Data Streams.

Untuk membuat aliran data AWS CLI

1. Untuk membuat aliran pertama (`ExampleInputStream`), gunakan perintah `create-stream` AWS CLI Amazon Kinesis berikut.

```
$ aws kinesis create-stream \
--stream-name ExampleInputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

2. Untuk membuat aliran kedua yang digunakan aplikasi untuk menulis output, jalankan perintah yang sama, yang mengubah nama aliran menjadi `ExampleOutputStream`.

```
$ aws kinesis create-stream \
--stream-name ExampleOutputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

Tulis Catatan Sampel ke Aliran Input

Di bagian ini, Anda menggunakan script Python untuk menulis catatan sampel ke aliran untuk diproses aplikasi.

> **Note**
>
> Bagian ini memerlukan AWS SDK for Python (Boto).

1. Buat file bernama `stock.py` dengan konten berikut:

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        "EVENT_TIME": datetime.datetime.now().isoformat(),
        "TICKER": random.choice(["AAPL", "AMZN", "MSFT", "INTC", "TBV"]),
        "PRICE": round(random.random() * 100, 2),
    }


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(data),
 PartitionKey="partitionkey"
        )


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

2. Selanjutnya dalam tutorial ini, Anda menjalankan skrip `stock.py` untuk mengirim data ke aplikasi.

```
$ python stock.py
```

Unduh dan Periksa Kode Java Streaming Apache Flink

Kode aplikasi Java untuk contoh ini tersedia dari GitHub. Untuk mengunduh kode aplikasi, lakukan hal berikut:

1. Kloning repositori jarak jauh menggunakan perintah berikut:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

2. Buka direktori `amazon-kinesis-data-analytics-java-examples/GettingStarted` tersebut.

Perhatikan hal tentang kode aplikasi berikut:

- File Project Object Model (pom.xml) berisi informasi tentang konfigurasi dan dependensi aplikasi, termasuk Layanan Terkelola untuk pustaka Apache Flink.
- File `BasicStreamingJob.java` berisi metode `main` yang menentukan fungsionalitas aplikasi.
- Aplikasi menggunakan sumber Kinesis untuk membaca dari aliran sumber. Cuplikan berikut ini membuat sumber Kinesis:

```
return env.addSource(new FlinkKinesisConsumer<>(inputStreamName,
                new SimpleStringSchema(), inputProperties));
```

- Aplikasi Anda membuat konektor sumber dan sink untuk mengakses sumber daya eksternal menggunakan objek `StreamExecutionEnvironment`.
- Aplikasi membuat konektor sumber dan sink menggunakan properti statis. Untuk menggunakan properti aplikasi dinamis, gunakan metode `createSourceFromApplicationProperties` dan `createSinkFromApplicationProperties` untuk membuat konektor. Metode ini membaca properti aplikasi untuk mengonfigurasi konektor.

  Untuk informasi selengkapnya tentang properti runtime, lihat Properti Runtime.

Kompilasi Kode Aplikasi

Di bagian ini, Anda menggunakan compiler Apache Maven untuk membuat kode Java untuk aplikasi. Untuk informasi tentang menginstal Apache Maven dan Java Development Kit (JDK), lihat Prasyarat untuk Menyelesaikan Latihan.

Untuk mengompikasi kode aplikasi

1. Untuk menggunakan kode aplikasi Anda, Anda mengompilasi dan mengemasnya ke dalam file JAR. Anda dapat mengompilasi dan mengemas kode Anda dengan salah satu dari dua cara:

- Gunakan alat Maven baris perintah. Buat file JAR Anda dengan menjalankan perintah berikut di direktori yang berisi file `pom.xml`:

```
mvn package -Dflink.version=1.11.3
```

- Menyiapkan lingkungan pengembangan Anda. Lihat dokumentasi lingkungan pengembangan Anda untuk detail.

> **Note**
>
> Kode sumber yang disediakan bergantung pada pustaka dari Java 11. Pastikan versi Java proyek Anda adalah 11.

Anda dapat mengunggah paket Anda sebagai file JAR, atau Anda dapat mengompresi paket Anda dan mengunggahnya sebagai file ZIP. Jika Anda membuat aplikasi Anda menggunakan AWS CLI, Anda menentukan tipe konten kode Anda (JAR atau ZIP).

2. Jika ada kesalahan saat mengompilasi, pastikan variabel lingkungan `JAVA_HOME`Anda diatur dengan benar.

Jika aplikasi berhasil mengompilasi, file berikut dibuat:

`target/aws-kinesis-analytics-java-apps-1.0.jar`

Unggah Kode Java Streaming Apache Flink

Pada bagian ini, Anda membuat bucket Amazon Simple Storage Service (Amazon S3) dan mengunggah kode aplikasi Anda.

Untuk mengunggah kode aplikasi

1. Buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.
2. Pilih Buat bucket.
3. Masukkan **`ka-app-code-<username>`** di bidang Bucket name (Nama bucket). Tambahkan sufiks ke nama bucket, seperti nama pengguna Anda, untuk membuatnya unik secara global. Pilih Next (Selanjutnya).
4. Di langkah Konfigurasi opsi, jangan ubah pengaturan, dan pilih Next (Selanjutnya).

5. Di langkah Atur izin, jangan ubah pengaturan, dan pilih Next (Selanjutnya).
6. Pilih Create bucket (Buat bucket).
7. Di konsol Amazon S3, pilih bucket ka-app-code -, dan pilih Unggah. <username>
8. Di langkah Pilih file, pilih Add files (Tambahkan berkas). Navigasikan ke file `aws-kinesis-analytics-java-apps-1.0.jar` yang Anda buat di langkah sebelumnya. Pilih Berikutnya.
9. Anda tidak perlu mengubah pengaturan apa pun untuk objek, jadi pilih Upload (Unggah).

Kode aplikasi Anda sekarang disimpan di bucket Amazon S3 yang dapat diakses aplikasi Anda.

Buat dan Jalankan Managed Service untuk Apache Flink Application

Anda dapat membuat dan menjalankan Layanan Terkelola untuk aplikasi Apache Flink menggunakan konsol atau aplikasi. AWS CLI

> **Note**
>
> Saat Anda membuat aplikasi menggunakan konsol, sumber daya AWS Identity and Access Management (IAM) dan Amazon CloudWatch Logs dibuat untuk Anda. Saat Anda membuat aplikasi menggunakan AWS CLI, Anda membuat sumber daya ini secara terpisah.

Topik

- Buat dan Jalankan Aplikasi (Konsol)
- Buat dan Jalankan Aplikasi (AWS CLI)

Buat dan Jalankan Aplikasi (Konsol)

Ikuti langkah-langkah ini untuk membuat, mengonfigurasi, memperbarui, dan menjalankan aplikasi menggunakan konsol.

Buat Aplikasi

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/flink
2. Pada dashboard Managed Service for Apache Flink, pilih Create Analytics Application.
3. Pada Layanan Terkelola untuk Apache Flink - Buat halaman aplikasi, berikan detail aplikasi sebagai berikut:

- Untuk Application name (Nama aplikasi), masukkan **MyApplication**.
- Untuk Description (Deskripsi), masukkan **My java test app**.
- Untuk Runtime, pilih Apache Flink.
- Biarkan versi pulldown sebagai Apache Flink versi 1.11 (Versi yang disarankan).

4. Untuk Access permissions (Izin akses), pilih Create / update IAM role **kinesis-analytics-MyApplication-us-west-2** (Buat/perbarui IAM role).
5. Pilih Create application (Buat aplikasi).

> **Note**
>
> Saat membuat aplikasi Managed Service for Apache Flink menggunakan konsol, Anda memiliki opsi untuk membuat peran dan kebijakan IAM untuk aplikasi Anda. Aplikasi Anda menggunakan peran dan kebijakan ini untuk mengakses sumber daya dependen. Sumber daya IAM ini diberi nama menggunakan nama aplikasi dan Wilayah sebagai berikut:
>
> - Kebijakan: `kinesis-analytics-service-MyApplication-us-west-2`
> - Peran: `kinesisanalytics-MyApplication-us-west-2`

Edit Kebijakan IAM

Edit kebijakan IAM untuk menambahkan izin mengakses Kinesis data streams.

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pilih Policies (Kebijakan). Pilih kebijakan **kinesis-analytics-service-MyApplication-us-west-2** yang dibuat konsol untuk Anda di bagian sebelumnya.
3. Di halaman Ringkasan, pilih Edit policy (Edit kebijakan). Pilih tab JSON.
4. Tambahkan bagian yang disorot dari contoh kebijakan berikut ke kebijakan. Ganti ID akun sampel (*012345678901*) dengan ID akun Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
```

```
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ka-app-code-username/aws-kinesis-analytics-java-
apps-1.0.jar"
            ]
        },
        {
            "Sid": "DescribeLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
            ]
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
```

```
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

Konfigurasikan Aplikasi

1. Pada MyApplicationhalaman, pilih Konfigurasi.
2. Di halaman Konfigurasikan aplikasi, berikan Code location (Lokasi kode):
   - Untuk Bucket Amazon S3, masukkan **ka-app-code-*<username>***.
   - Untuk Jalur ke objek Amazon S3, masukkan **aws-kinesis-analytics-java-apps-1.0.jar**.
3. Di bawah Akses ke sumber daya aplikasi, untuk Access permissions (Izin akses), pilih Create / update IAM role **kinesis-analytics-MyApplication-us-west-2** (Pilih/perbarui IAM role).
4. Di bawah Properties (Properti), untuk Group ID (ID Grup), masukkan **ProducerConfigProperties**.
5. Masukkan properti dan nilai aplikasi berikut:

ID Grup	Kunci	Nilai
ProducerConfigProperties	**flink.inputstream.initpos**	**LATEST**
ProducerConfigProperties	**aws.region**	**us-west-2**

ID Grup	Kunci	Nilai
ProducerConfigProperties	**AggregationEnabled**	**false**

6. Di bawah Pemantauan, pastikan Memantau tingkat metrik diatur ke Aplikasi.
7. Untuk CloudWatch logging, pilih kotak centang Aktifkan.
8. Pilih Perbarui.

> **Note**
>
> Saat Anda memilih untuk mengaktifkan CloudWatch pencatatan Amazon, Layanan Terkelola untuk Apache Flink membuat grup log dan aliran log untuk Anda. Nama-nama sumber daya ini adalah sebagai berikut:
>
> - Grup log: `/aws/kinesis-analytics/MyApplication`
> - Aliran log: `kinesis-analytics-log-stream`

Jalankan Aplikasi

Grafik pekerjaan Flink dapat dilihat dengan menjalankan aplikasi, membuka dasbor Apache Flink, dan memilih pekerjaan Flink yang diinginkan.

Hentikan Aplikasi

Pada MyApplicationhalaman, pilih Berhenti. Konfirmasikan tindakan.

Perbarui Aplikasi

Dengan menggunakan konsol, Anda dapat memperbarui pengaturan aplikasi seperti properti aplikasi, pengaturan pemantauan, dan lokasi atau nama file dari JAR aplikasi. Anda juga dapat memuat ulang aplikasi JAR dari bucket Amazon S3 jika Anda perlu memperbarui kode aplikasi.

Pada MyApplicationhalaman, pilih Konfigurasi. Perbarui pengaturan aplikasi dan pilih Update (Perbarui).

Buat dan Jalankan Aplikasi (AWS CLI)

Di bagian ini, Anda menggunakan AWS CLI untuk membuat dan menjalankan aplikasi Managed Service for Apache Flink. Managed Service for Apache Flink menggunakan `kinesisanalyticsv2` AWS CLI perintah untuk membuat dan berinteraksi dengan Managed Service untuk aplikasi Apache Flink.

Membuat Kebijakan Izin

> **Note**
>
> Anda harus membuat kebijakan izin dan peran untuk aplikasi Anda. Jika Anda tidak membuat sumber daya IAM ini, aplikasi Anda tidak dapat mengakses data dan aliran log.

Pertama, Anda membuat kebijakan izin dengan dua pernyataan: satu yang memberikan izin untuk tindakan `read` di aliran sumber, dan lainnya yang memberikan izin untuk tindakan `write` di aliran sink. Anda selanjutnya melampirkan kebijakan ke IAM role (yang Anda buat di bagian berikutnya). Jadi, ketika Layanan Terkelola untuk Apache Flink mengasumsikan peran tersebut, layanan memiliki izin yang diperlukan untuk membaca dari aliran sumber dan menulis ke aliran wastafel.

Gunakan kode berikut untuk membuat kebijakan izin `AKReadSourceStreamWriteSinkStream`. Ganti *`username`* dengan nama pengguna yang Anda gunakan untuk membuat bucket Amazon S3 untuk menyimpan kode aplikasi. Ganti ID akun di Amazon Resource Names (ARN) (*`012345678901`*) dengan ID akun Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": ["arn:aws:s3:::ka-app-code-username",
                "arn:aws:s3:::ka-app-code-username/*"
            ]
        },
```

```
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

Untuk step-by-step petunjuk membuat kebijakan izin, lihat Tutorial: Membuat dan Melampirkan Kebijakan Terkelola Pelanggan Pertama Anda di Panduan Pengguna IAM.

> **Note**
>
> Untuk mengakses layanan Amazon lainnya, Anda dapat menggunakan AWS SDK for Java. Layanan Terkelola untuk Apache Flink secara otomatis menyetel kredensyal yang diperlukan oleh SDK ke peran IAM eksekusi layanan yang terkait dengan aplikasi Anda. Tidak ada langkah tambahan yang diperlukan.

Buat IAM Role

Di bagian ini, Anda membuat peran IAM yang dapat diasumsikan oleh aplikasi Managed Service for Apache Flink untuk membaca aliran sumber dan menulis ke aliran sink.

Layanan Terkelola untuk Apache Flink tidak dapat mengakses aliran Anda tanpa izin. Anda memberikan izin ini melalui IAM role. Setiap IAM role memiliki dua kebijakan yang dilampirkan. Kebijakan kepercayaan memberikan izin Layanan Terkelola untuk Apache Flink untuk mengambil peran, dan kebijakan izin menentukan apa yang dapat dilakukan Layanan Terkelola untuk Apache Flink setelah mengambil peran.

Anda melampirkan kebijakan izin yang Anda buat di bagian sebelumnya ke peran ini.

Untuk membuat IAM role

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Dalam panel navigasi, pilih Roles (Peran), Create role (Buat Peran).
3. Di bawah Pilih tipe identitas tepercaya, pilih Layanan AWS. Di bawah Pilih layanan yang akan menggunakan peran ini, pilih Kinesis. Di bawah Pilih kasus penggunaan Anda, pilih Analitik Kinesis.

   Pilih Selanjutnya: Izin.
4. Di halaman Lampirkan kebijakan izin, pilih Next: Review (Selanjutnya: Tinjau). Anda melampirkan kebijakan izin setelah Anda membuat peran tersebut.
5. Di halaman Buat peran, masukkan **`MF-stream-rw-role`** untuk Role name (Nama peran). Pilih Create role (Buat peran).

   Sekarang Anda sudah membuat IAM role baru yang disebut `MF-stream-rw-role`. Berikutnya, Anda memperbarui kebijakan kepercayaan dan izin untuk peran.
6. Lampirkan kebijakan izin ke peran tersebut.

   > **Note**
   >
   > Untuk latihan ini, Managed Service for Apache Flink mengasumsikan peran ini untuk membaca data dari aliran data Kinesis (sumber) dan menulis output ke aliran data Kinesis lain. Jadi, Anda melampirkan kebijakan yang Anda buat di langkah sebelumnya, the section called “Membuat Kebijakan Izin”.

   a. Di halaman Ringkasan, pilih tab Permissions (Izin).
   b. Pilih Attach Policies (Lampirkan Kebijakan).
   c. Di kotak pencarian, masukkan **`AKReadSourceStreamWriteSinkStream`** (kebijakan yang Anda buat bagian sebelumnya).
   d. Pilih ReadSourceStreamWriteSinkStream kebijakan AK, dan pilih Lampirkan kebijakan.

Anda sekarang telah membuat peran eksekusi layanan yang digunakan aplikasi Anda untuk mengakses sumber daya. Buat catatan ARN peran baru.

Untuk step-by-step petunjuk cara membuat peran, lihat Membuat Peran IAM (Konsol) di Panduan Pengguna IAM.

Buat Managed Service untuk Apache Flink Application

1. Simpan kode JSON berikut ke file bernama `create_request.json`. Ganti ARN peran sampel dengan ARN untuk peran yang Anda buat sebelumnya. Ganti sufiks ARN bucket (*username*) dengan sufiks yang Anda pilih di bagian sebelumnya. Ganti ID akun sampel (*012345678901*) di peran eksekusi layanan dengan ID akun Anda.

```
{
    "ApplicationName": "test",
    "ApplicationDescription": "my java test app",
    "RuntimeEnvironment": "FLINK-1_11",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-stream-rw-role",
    "ApplicationConfiguration": {
        "ApplicationCodeConfiguration": {
            "CodeContent": {
                "S3ContentLocation": {
                    "BucketARN": "arn:aws:s3:::ka-app-code-username",
                    "FileKey": "aws-kinesis-analytics-java-apps-1.0.jar"
                }
            },
            "CodeContentType": "ZIPFILE"
        },
        "EnvironmentProperties":  {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "flink.stream.initpos" : "LATEST",
                    "aws.region" : "us-west-2",
                    "AggregationEnabled" : "false"
               }
            },
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2"
               }
            }
         ]
      }
```

```
    }
}
```

2. Jalankan tindakan `CreateApplication` dengan permintaan sebelumnya untuk membuat aplikasi:

```
aws kinesisanalyticsv2 create-application --cli-input-json file://
create_request.json
```

Aplikasi ini sekarang dibuat. Anda memulai aplikasi di langkah berikutnya.

Mulai Aplikasi

Di bagian ini, Anda menggunakan tindakan `StartApplication` untuk memulai aplikasi.

Untuk memulai aplikasi

1. Simpan kode JSON berikut ke file bernama `start_request.json`.

```
{
    "ApplicationName": "test",
    "RunConfiguration": {
        "ApplicationRestoreConfiguration": {
         "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"
         }
    }
}
```

2. Jalankan tindakan `StartApplication` dengan permintaan sebelumnya untuk memulai aplikasi:

```
aws kinesisanalyticsv2 start-application --cli-input-json file://start_request.json
```

Aplikasi sekarang berjalan. Anda dapat memeriksa metrik Layanan Terkelola untuk Apache Flink di CloudWatch konsol Amazon untuk memverifikasi bahwa aplikasi berfungsi.

Hentikan Aplikasi

Di bagian ini, Anda menggunakan tindakan `StopApplication` untuk menghentikan aplikasi.

Untuk menghentikan aplikasi

1. Simpan kode JSON berikut ke file bernama `stop_request.json`.

```
{
    "ApplicationName": "test"
}
```

2. Jalankan tindakan `StopApplication` dengan permintaan berikut untuk menghentikan aplikasi:

```
aws kinesisanalyticsv2 stop-application --cli-input-json file://stop_request.json
```

Aplikasi sekarang dihentikan.

Tambahkan Opsi CloudWatch Logging

Anda dapat menggunakan AWS CLI untuk menambahkan aliran CloudWatch log Amazon ke aplikasi Anda. Untuk informasi tentang menggunakan CloudWatch Log dengan aplikasi Anda, lihatthe section called “Menyiapkan Pencatatan”.

Perbarui Properti Lingkungan

Di bagian ini, Anda menggunakan tindakan `UpdateApplication` untuk mengubah properti lingkungan untuk aplikasi tanpa mengompilasi ulang kode aplikasi. Dalam contoh ini, Anda mengubah Wilayah aliran sumber dan tujuan.

Untuk memperbarui properti lingkungan untuk aplikasi

1. Simpan kode JSON berikut ke file bernama `update_properties_request.json`.

```
{"ApplicationName": "test",
   "CurrentApplicationVersionId": 1,
   "ApplicationConfigurationUpdate": {
      "EnvironmentPropertyUpdates": {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "flink.stream.initpos" : "LATEST",
                    "aws.region" : "us-west-2",
```

```
                    "AggregationEnabled" : "false"
                }
            },
            {
                "PropertyGroupId": "ConsumerConfigProperties",
                "PropertyMap" : {
                    "aws.region" : "us-west-2"
                }
            }
         ]
      }
   }
}
```

2. Jalankan tindakan `UpdateApplication` dengan permintaan sebelumnya untuk memperbarui properti lingkungan:

```
aws kinesisanalyticsv2 update-application --cli-input-json file://
update_properties_request.json
```

Perbarui Kode Aplikasi

Ketika Anda perlu memperbarui kode aplikasi Anda dengan versi baru paket kode Anda, Anda menggunakan tindakan `UpdateApplication` AWS CLI.

> **Note**
>
> Untuk memuat versi baru kode aplikasi dengan nama file yang sama, Anda harus menentukan versi objek baru. Untuk informasi selengkapnya tentang menggunakan versi objek Amazon S3, lihat Mengaktifkan dan Menonaktifkan Versioning.

Untuk menggunakan AWS CLI, hapus paket kode Anda sebelumnya dari bucket Amazon S3 Anda, unggah versi baru, dan panggil `UpdateApplication`, yang menentukan bucket Amazon S3 dan nama objek yang sama, serta versi objek baru. Aplikasi akan memulai ulang dengan paket kode baru.

Permintaan sampel berikut untuk tindakan `UpdateApplication` memuat ulang kode aplikasi dan memulai ulang aplikasi. Perbarui `CurrentApplicationVersionId` ke versi aplikasi saat ini. Anda dapat memeriksa versi aplikasi saat ini menggunakan tindakan `ListApplications` atau

`DescribeApplication`. Perbarui sufiks nama bucket (*<username>*) dengan sufiks yang Anda pilih di bagian the section called “Buat Dua Amazon Kinesis Data Streams”.

```
{
    "ApplicationName": "test",
    "CurrentApplicationVersionId": 1,
    "ApplicationConfigurationUpdate": {
        "ApplicationCodeConfigurationUpdate": {
            "CodeContentUpdate": {
                "S3ContentLocationUpdate": {
                    "BucketARNUpdate": "arn:aws:s3:::ka-app-code-username",
                    "FileKeyUpdate": "aws-kinesis-analytics-java-apps-1.0.jar",
                    "ObjectVersionUpdate": "SAMPLEUehYngP87ex1nzYIGYgfhypvDU"
                }
            }
        }
    }
}
```

Langkah Selanjutnya

Langkah 4: Bersihkan Sumber Daya AWS

Langkah 4: Bersihkan Sumber Daya AWS

Bagian ini mencakup prosedur untuk membersihkan sumber daya AWS yang dibuat dalam tutorial Memulai.

Topik ini berisi bagian-bagian berikut:

- Hapus Layanan Terkelola Anda untuk Aplikasi Apache Flink
- Hapus Kinesis Data Streams Anda
- Hapus Objek dan Bucket Amazon S3 Anda
- Hapus Sumber Daya IAM Anda
- Hapus CloudWatch Sumber Daya Anda
- Langkah Selanjutnya

Hapus Layanan Terkelola Anda untuk Aplikasi Apache Flink

1. Buka konsol Kinesis di https://console.aws.amazon.com/kinesis.

2. Di panel Managed Service for Apache Flink, pilih. MyApplication
3. Di halaman aplikasi, pilih Delete (Hapus), lalu konfirmasikan penghapusan.

Hapus Kinesis Data Streams Anda

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/flink
2. Di panel Kinesis Data Streams, pilih. ExampleInputStream
3. Di ExampleInputStreamhalaman, pilih Hapus Stream Kinesis dan kemudian konfirmasikan penghapusan.
4. Di halaman Kinesis streams, pilih, pilih Tindakan ExampleOutputStream, pilih Hapus, lalu konfirmasikan penghapusan.

Hapus Objek dan Bucket Amazon S3 Anda

1. Buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.
2. Pilih ka-app-code- ember. <username>
3. Pilih Delete (Hapus), lalu masukkan nama bucket untuk mengonfirmasi penghapusan.

Hapus Sumber Daya IAM Anda

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di bilah navigasi, pilih Policies (Kebijakan).
3. Di kontrol filter, masukkan kinesis.
4. Pilih kebijakan kinesis-analytics-service- MyApplication -us-west-2.
5. Pilih Policy Actions (Tindakan Kebijakan), lalu pilih Delete (Hapus).
6. Di bilah navigasi, pilih Roles (Peran).
7. Pilih peran kinesis-analytics- -us-west-2. MyApplication
8. Pilih Delete role (Hapus peran), lalu konfirmasi penghapusan.

Hapus CloudWatch Sumber Daya Anda

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di bilah navigasi, pilih Logs.

3. Pilih grup log MyApplication/aws/kinesis-analytics/.
4. Pilih Delete Log Group (Hapus Grup Log), lalu konfirmasi penghapusan.

Langkah Selanjutnya

Langkah 5: Langkah selanjutnya

Langkah 5: Langkah selanjutnya

Setelah Anda membuat dan menjalankan Managed Service dasar untuk aplikasi Apache Flink, lihat sumber daya berikut untuk solusi Managed Service for Apache Flink yang lebih canggih.

- Solusi Data Streaming AWS untuk Amazon Kinesis: Solusi Data Streaming AWS untuk Amazon Kinesis secara otomatis mengonfigurasi layanan AWS yang diperlukan untuk dengan mudah menangkap, menyimpan, memproses, dan mengirimkan data streaming. Solusi ini menyediakan beberapa opsi untuk memecahkan kasus penggunaan data streaming. Layanan Terkelola untuk opsi Apache Flink menyediakan contoh ETL end-to-end streaming yang menunjukkan aplikasi dunia nyata yang menjalankan operasi analitis pada data taksi New York yang disimulasikan. Solusinya menyiapkan semua AWS sumber daya yang diperlukan seperti peran dan kebijakan IAM, CloudWatch dasbor, dan CloudWatch alarm.
- AWSSolusi Data Streaming untuk Amazon MSK: Solusi Data AWS Streaming untuk Amazon MSK menyediakan AWS CloudFormation template tempat data mengalir melalui produsen, penyimpanan streaming, konsumen, dan tujuan.
- Clickstream Lab dengan Apache Flink dan Apache Kafka: Lab ujung ke ujung untuk kasus penggunaan clickstream menggunakan Amazon Managed Streaming untuk Apache Kafka Kafka untuk penyimpanan streaming dan Layanan Terkelola untuk Apache Flink untuk aplikasi Apache Flink untuk pemrosesan streaming.
- Amazon Managed Service untuk Apache Flink Workshop: Dalam lokakarya ini, Anda membangun arsitektur end-to-end streaming untuk menelan, menganalisis, dan memvisualisasikan data streaming dalam waktu dekat. Anda mulai meningkatkan operasi perusahaan taksi di Kota New York. Anda menganalisis data telemetri armada taksi di Kota New York hampir secara langsung untuk mengoptimalkan operasi armada mereka.
- Layanan Terkelola untuk Apache Flink: Contoh: Bagian Panduan Pengembang ini memberikan contoh membuat dan bekerja dengan aplikasi di Managed Service untuk Apache Flink. Mereka menyertakan kode contoh dan step-by-step instruksi untuk membantu Anda membuat Layanan Terkelola untuk aplikasi Apache Flink dan menguji hasil Anda.

- Belajar Flink: Pelatihan Langsung: Pelatihan pengenalan resmi Apache Flink yang membuat Anda mulai menulis ETL streaming yang dapat diskalakan, analitik, dan aplikasi yang didorong peristiwa.

> **Note**
>
> Ketahuilah bahwa Managed Service for Apache Flink tidak mendukung versi Apache Flink (1.12) yang digunakan dalam pelatihan ini. Anda dapat menggunakan Flink 1.15.2 di Flink Managed Service untuk Apache Flink.

- Contoh Kode Apache Flink: Sebuah GitHub repositori dari berbagai macam contoh aplikasi Apache Flink.

Memulai: Flink 1.8.2

Topik ini berisi versi Memulai (DataStream API) Tutorial yang menggunakan Apache Flink 1.8.2.

Topik

- Komponen Layanan Terkelola untuk Aplikasi Apache Flink
- Prasyarat untuk Menyelesaikan Latihan
- Langkah 1: Siapkan Akun AWS dan Buat Pengguna Administrator
- Langkah 2: Siapkan AWS Command Line Interface (AWS CLI)
- Langkah 3: Buat dan Jalankan Layanan Terkelola untuk aplikasi Apache Flink
- Langkah 4: Bersihkan Sumber Daya AWS

Komponen Layanan Terkelola untuk Aplikasi Apache Flink

Untuk memproses data, Managed Service untuk aplikasi Apache Flink Anda menggunakan aplikasi Java/Apache Maven atau Scala yang memproses input dan menghasilkan output menggunakan runtime Apache Flink.

Layanan Terkelola untuk aplikasi Apache Flink memiliki komponen-komponen berikut:

- Properti runtime: Anda dapat menggunakan properti runtime untuk mengonfigurasi aplikasi Anda tanpa mengompilasi ulang kode aplikasi Anda.
- Source (Sumber): Aplikasi mengonsumsi data menggunakan sumber. Konektor sumber membaca data dari Kinesis data stream, bucket Amazon S3, dll. Untuk informasi selengkapnya, lihat Sumber.

- Operators (Operator): Aplikasi memproses data menggunakan satu atau beberapa operator. Operator dapat mengubah, memperkaya, atau menggabungkan data. Untuk informasi selengkapnya, lihat DataStream Operator API.
- Sink: Aplikasi menghasilkan data ke sumber eksternal menggunakan sink. Konektor sink menulis data ke aliran data Kinesis, aliran Firehose Data Kinesis, bucket Amazon S3, dll. Untuk informasi selengkapnya, lihat Sink.

Setelah Anda membuat, mengompilasi, dan mengemas kode aplikasi Anda, Anda mengunggah paket kode ke bucket Amazon Simple Storage Service (Amazon S3). Anda kemudian membuat Layanan Terkelola untuk aplikasi Apache Flink. Anda meneruskan di lokasi paket kode, Kinesis data stream sebagai sumber data streaming, dan biasanya lokasi streaming atau file yang menerima data yang diproses dari aplikasi.

Prasyarat untuk Menyelesaikan Latihan

Untuk menyelesaikan langkah-langkah di panduan ini, Anda harus memiliki hal-hal berikut:

- Java Development Kit (JDK) versi 8. Atur variabel lingkungan `JAVA_HOME` untuk menunjuk ke lokasi penginstalan JDK Anda.
- Untuk menggunakan konektor Apache Flink Kinesis dalam tutorial ini, Anda harus mengunduh dan menginstal Apache Flink. Untuk detailnya, lihat Menggunakan Konektor Apache Flink Kinesis Streams dengan versi Apache Flink sebelumnya.
- Sebaiknya gunakan lingkungan pengembangan (seperti Eclipse Java Neon atau IntelliJ Idea) untuk mengembangkan dan mengompilasi aplikasi Anda.
- Klien Git. Instal klien Git jika Anda belum menginstalnya.
- Plugin Compiler Apache Maven. Maven harus berada di jalur kerja Anda. Untuk menguji instalasi Apache Maven Anda, masukkan hal berikut:

```
$ mvn -version
```

Untuk memulai, buka Langkah 1: Siapkan Akun AWS dan Buat Pengguna Administrator.

Langkah 1: Siapkan Akun AWS dan Buat Pengguna Administrator

Mendaftar Akun AWS

Jika Anda tidak memiliki Akun AWS, selesaikan langkah-langkah berikut untuk membuatnya.

Untuk mendaftar Akun AWS

1. Buka https://portal.aws.amazon.com/billing/signup.
2. Ikuti petunjuk secara online.

   Anda akan diminta untuk menerima panggilan telepon dan memasukkan kode verifikasi pada keypad telepon sebagai bagian dari prosedur pendaftaran.

   Saat Anda mendaftar Akun AWS, Pengguna root akun AWS akan dibuat. Pengguna root memiliki akses ke semua Layanan AWS dan sumber daya dalam akun. Sebagai praktik terbaik keamanan, tetapkan akses administratif ke pengguna administratif, dan hanya gunakan pengguna root untuk melakukan tugas yang memerlukan akses pengguna root.

AWS akan mengirimkan email konfirmasi kepada Anda setelah proses pendaftaran selesai. Anda dapat melihat aktivitas akun saat ini dan mengelola akun dengan mengunjungi https://aws.amazon.com/ dan memilih Akun Saya.

Membuat pengguna administratif

Setelah mendaftar Akun AWS, amankan Pengguna root akun AWS, aktifkan AWS IAM Identity Center, dan buat sebuah pengguna administratif sehingga Anda tidak menggunakan pengguna root untuk tugas sehari-hari.

Mengamankan Pengguna root akun AWS Anda

1. Masuk ke AWS Management Console sebagai pemilik akun dengan memilih Pengguna root dan memasukkan alamat email Akun AWS Anda. Di halaman berikutnya, masukkan kata sandi Anda.

   Untuk bantuan masuk menggunakan pengguna root, lihat Masuk sebagai pengguna root dalam Panduan Pengguna AWS Sign-In.
2. Aktifkan autentikasi multi-faktor (MFA) untuk pengguna root Anda.

   Untuk petunjuknya, silakan lihat Mengaktifkan perangkat MFA virtual untuk pengguna root Akun AWS Anda (konsol) dalam Panduan Pengguna IAM.

Membuat pengguna administratif

1. Aktifkan Pusat Identitas IAM.

   Untuk mendapatkan petunjuk, silakan lihat Mengaktifkan AWS IAM Identity Center di Panduan Pengguna AWS IAM Identity Center.
2. Di Pusat Identitas IAM, berikan akses administratif ke sebuah pengguna administratif.

   Untuk mendapatkan tutorial tentang menggunakan Direktori Pusat Identitas IAM sebagai sumber identitas Anda, silakan lihat Mengonfigurasi akses pengguna dengan Direktori Pusat Identitas IAM default di Panduan Pengguna AWS IAM Identity Center.

Masuk sebagai pengguna administratif

- Untuk masuk dengan pengguna Pusat Identitas IAM, gunakan URL masuk yang dikirim ke alamat email Anda saat Anda membuat pengguna Pusat Identitas IAM.

  Untuk bantuan masuk menggunakan pengguna Pusat Identitas IAM, lihat Masuk ke portal akses AWS dalam Panduan Pengguna AWS Sign-In.

Memberikan akses terprogram

Pengguna membutuhkan akses terprogram jika mereka ingin berinteraksi dengan AWS luar dari AWS Management Console. Cara memberikan akses terprogram bergantung pada jenis pengguna yang mengakses AWS.

Untuk memberi pengguna akses terprogram, pilih salah satu opsi berikut.

Pengguna mana yang membutuhkan akses terprogram?	Untuk	Oleh
Identitas tenaga kerja (Pengguna yang dikelola di Pusat Identitas IAM)	Gunakan kredensial sementara untuk menandatangani permintaan terprogram ke AWS CLI, SDK AWS, atau API AWS.	Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan. • Untuk AWS CLI, lihat Mengonfigurasi AWS CLI untuk menggunakan AWS

Pengguna mana yang membutuhkan akses terprogram?	Untuk	Oleh
		IAM Identity Center di Panduan Pengguna AWS Command Line Interface. • Untuk SDK AWS, alat, dan API AWS, lihat Autentikasi Pusat Identitas IAM di Panduan Referensi SDK dan Alat AWS.
IAM	Gunakan kredensial sementara untuk menandatangani permintaan terprogram ke AWS CLI, SDK AWS, atau API AWS.	Mengikuti petunjuk dalam Menggunakan kredensial sementara dengan sumber daya AWS di Panduan Pengguna IAM.

Pengguna mana yang membutuhkan akses terprogram?	Untuk	Oleh
IAM	(Tidak direkomendasikan) Gunakan kredensial jangka panjang untuk menandatangani permintaan terprogram ke AWS CLI, SDK AWS, atau API AWS.	Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan. • Untuk AWS CLI, lihat Mengautentikasi menggunakan kredensial pengguna IAM di Panduan Pengguna AWS Command Line Interface. • Untuk SDK dan alat AWS, lihat Mengautentikasi menggunakan kredensial jangka panjang di Panduan Referensi SDK dan Alat AWS. • Untuk API AWS, lihat Mengelola kunci akses untuk pengguna IAM di Panduan Pengguna IAM.

Langkah 2: Siapkan AWS Command Line Interface (AWS CLI)

Pada langkah ini, Anda mengunduh dan mengonfigurasi AWS CLI untuk digunakan dengan Managed Service for Apache Flink.

> **Note**
>
> Latihan memulai dalam panduan ini mengasumsikan Anda menggunakan kredensial administrator (`adminuser`) di akun Anda untuk melakukan operasi.

> **Note**
>
> Jika sudah menginstal AWS CLI, Anda mungkin perlu meningkatkan untuk mendapatkan fungsionalitas terbaru. Untuk informasi selengkapnya, lihat Menginstal AWS Command Line Interface dalam Panduan Pengguna AWS Command Line Interface. Untuk memeriksa versi AWS CLI, jalankan perintah berikut.
>
> ```
> aws --version
> ```
>
> Latihan dalam tutorial ini memerlukan versi AWS CLI berikut atau yang lebih baru:
>
> ```
> aws-cli/1.16.63
> ```

Untuk mengatur AWS CLI

1. Unduh dan konfigurasikan AWS CLI. Untuk instruksinya, lihat topik berikut di AWS Command Line InterfacePanduan Pengguna:
   - Menginstal AWS Command Line Interface
   - Mengonfigurasi AWS CLI
2. Tambahkan profil bernama untuk pengguna administrator dalam file AWS CLI `config`. Anda dapat menggunakan profil ini saat menjalankan perintah AWS CLI. Untuk informasi selengkapnya tentang profil yang diberi nama, lihat Profil yang Diberi Nama dalam Panduan Pengguna AWS Command Line Interface.

   ```
   [profile adminuser]
   aws_access_key_id = adminuser access key ID
   aws_secret_access_key = adminuser secret access key
   region = aws-region
   ```

   Untuk daftar Wilayah yang tersedia, lihat Wilayah dan Titik Akhir di. Referensi Umum Amazon Web Services

> **Note**
>
> Kode dan perintah contoh dalam tutorial ini menggunakan Wilayah US West (Oregon). Untuk menggunakan Wilayah AWS yang berbeda, ubah Wilayah dalam kode dan perintah untuk tutorial ini ke Wilayah yang ingin Anda gunakan.

3. Verifikasikan penyiapan dengan memasukkan perintah bantuan berikut pada prompt perintah.

```
aws help
```

Setelah Anda mengatur AWS akun danAWS CLI, Anda dapat mencoba latihan berikutnya, di mana Anda mengkonfigurasi aplikasi sampel dan menguji end-to-end pengaturan.

Langkah Selanjutnya

Langkah 3: Buat dan Jalankan Layanan Terkelola untuk aplikasi Apache Flink

Langkah 3: Buat dan Jalankan Layanan Terkelola untuk aplikasi Apache Flink

Dalam latihan ini, Anda membuat Layanan Terkelola untuk aplikasi Apache Flink dengan aliran data sebagai sumber dan wastafel.

Bagian ini berisi langkah-langkah berikut.

- Buat Dua Amazon Kinesis Data Streams
- Tulis Catatan Sampel ke Aliran Input
- Unduh dan Periksa Kode Java Streaming Apache Flink
- Kompilasi Kode Aplikasi
- Unggah Kode Java Streaming Apache Flink
- Buat dan Jalankan Managed Service untuk Apache Flink Application
- Langkah Selanjutnya

Buat Dua Amazon Kinesis Data Streams

Sebelum Anda membuat Layanan Terkelola untuk aplikasi Apache Flink untuk latihan ini, buat dua aliran data Kinesis (dan). `ExampleInputStream` `ExampleOutputStream` Aplikasi Anda menggunakan aliran ini untuk sumber aplikasi dan aliran tujuan.

Anda dapat membuat aliran ini menggunakan konsol Amazon Kinesis atau perintah AWS CLI berikut. Untuk instruksi konsol, lihat Membuat dan Memperbarui Aliran Data di Panduan Developer Amazon Kinesis Data Streams.

Untuk membuat aliran data AWS CLI

1. Untuk membuat aliran pertama (`ExampleInputStream`), gunakan perintah `create-stream` AWS CLI Amazon Kinesis berikut.

```
$ aws kinesis create-stream \
--stream-name ExampleInputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

2. Untuk membuat aliran kedua yang digunakan aplikasi untuk menulis output, jalankan perintah yang sama, yang mengubah nama aliran menjadi `ExampleOutputStream`.

```
$ aws kinesis create-stream \
--stream-name ExampleOutputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

Tulis Catatan Sampel ke Aliran Input

Di bagian ini, Anda menggunakan script Python untuk menulis catatan sampel ke aliran untuk diproses aplikasi.

> **Note**
>
> Bagian ini memerlukan AWS SDK for Python (Boto).

1. Buat file bernama `stock.py` dengan konten berikut:

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        "EVENT_TIME": datetime.datetime.now().isoformat(),
        "TICKER": random.choice(["AAPL", "AMZN", "MSFT", "INTC", "TBV"]),
        "PRICE": round(random.random() * 100, 2),
    }


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(data),
 PartitionKey="partitionkey"
        )


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

2. Selanjutnya dalam tutorial ini, Anda menjalankan skrip `stock.py` untuk mengirim data ke aplikasi.

```
$ python stock.py
```

Unduh dan Periksa Kode Java Streaming Apache Flink

Kode aplikasi Java untuk contoh ini tersedia dari GitHub. Untuk mengunduh kode aplikasi, lakukan hal berikut:

1. Kloning repositori jarak jauh menggunakan perintah berikut:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

2. Buka direktori `amazon-kinesis-data-analytics-java-examples/GettingStarted_1_8` tersebut.

Perhatikan hal tentang kode aplikasi berikut:

- File Project Object Model (pom.xml) berisi informasi tentang konfigurasi dan dependensi aplikasi, termasuk Layanan Terkelola untuk pustaka Apache Flink.
- File `BasicStreamingJob.java` berisi metode `main` yang menentukan fungsionalitas aplikasi.
- Aplikasi menggunakan sumber Kinesis untuk membaca dari aliran sumber. Cuplikan berikut ini membuat sumber Kinesis:

```
return env.addSource(new FlinkKinesisConsumer<>(inputStreamName,
                new SimpleStringSchema(), inputProperties));
```

- Aplikasi Anda membuat konektor sumber dan sink untuk mengakses sumber daya eksternal menggunakan objek `StreamExecutionEnvironment`.
- Aplikasi membuat konektor sumber dan sink menggunakan properti statis. Untuk menggunakan properti aplikasi dinamis, gunakan metode `createSourceFromApplicationProperties` dan `createSinkFromApplicationProperties` untuk membuat konektor. Metode ini membaca properti aplikasi untuk mengonfigurasi konektor.

  Untuk informasi selengkapnya tentang properti runtime, lihat Properti Runtime.

Kompilasi Kode Aplikasi

Di bagian ini, Anda menggunakan compiler Apache Maven untuk membuat kode Java untuk aplikasi. Untuk informasi tentang menginstal Apache Maven dan Java Development Kit (JDK), lihat Prasyarat untuk Menyelesaikan Latihan.

> **Note**
>
> Untuk menggunakan konektor Kinesis dengan versi Apache Flink sebelum 1.11, Anda perlu mengunduh, membangun, dan menginstal Apache Maven. Lihat informasi yang lebih lengkap

di the section called “Menggunakan Konektor Apache Flink Kinesis Streams dengan versi Apache Flink sebelumnya”.

Untuk mengompikasi kode aplikasi

1. Untuk menggunakan kode aplikasi Anda, Anda mengompilasi dan mengemasnya ke dalam file JAR. Anda dapat mengompilasi dan mengemas kode Anda dengan salah satu dari dua cara:

   - Gunakan alat Maven baris perintah. Buat file JAR Anda dengan menjalankan perintah berikut di direktori yang berisi file `pom.xml`:

     ```
     mvn package -Dflink.version=1.8.2
     ```

   - Menyiapkan lingkungan pengembangan Anda. Lihat dokumentasi lingkungan pengembangan Anda untuk detail.

     > **Note**
     >
     > Kode sumber yang disediakan bergantung pada pustaka dari Java 1.8. Pastikan versi Java proyek Anda adalah 1.8.

   Anda dapat mengunggah paket Anda sebagai file JAR, atau Anda dapat mengompresi paket Anda dan mengunggahnya sebagai file ZIP. Jika Anda membuat aplikasi Anda menggunakan AWS CLI, Anda menentukan tipe konten kode Anda (JAR atau ZIP).

2. Jika ada kesalahan saat mengompilasi, pastikan variabel lingkungan `JAVA_HOME`Anda diatur dengan benar.

Jika aplikasi berhasil mengompilasi, file berikut dibuat:

`target/aws-kinesis-analytics-java-apps-1.0.jar`

Unggah Kode Java Streaming Apache Flink

Pada bagian ini, Anda membuat bucket Amazon Simple Storage Service (Amazon S3) dan mengunggah kode aplikasi Anda.

Untuk mengunggah kode aplikasi

1. Buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.
2. Pilih Buat bucket.
3. Masukkan **ka-app-code-*<username>*** di bidang Bucket name (Nama bucket). Tambahkan sufiks ke nama bucket, seperti nama pengguna Anda, untuk membuatnya unik secara global. Pilih Next (Selanjutnya).
4. Di langkah Konfigurasi opsi, jangan ubah pengaturan, dan pilih Next (Selanjutnya).
5. Di langkah Atur izin, jangan ubah pengaturan, dan pilih Next (Selanjutnya).
6. Pilih Create bucket (Buat bucket).
7. Di konsol Amazon S3, pilih bucket ka-app-code -, dan pilih Unggah. <username>
8. Di langkah Pilih file, pilih Add files (Tambahkan berkas). Navigasikan ke file `aws-kinesis-analytics-java-apps-1.0.jar` yang Anda buat di langkah sebelumnya. Pilih Berikutnya.
9. Anda tidak perlu mengubah pengaturan apa pun untuk objek, jadi pilih Upload (Unggah).

Kode aplikasi Anda sekarang disimpan di bucket Amazon S3 yang dapat diakses aplikasi Anda.

Buat dan Jalankan Managed Service untuk Apache Flink Application

Anda dapat membuat dan menjalankan Layanan Terkelola untuk aplikasi Apache Flink menggunakan konsol atau aplikasi. AWS CLI

> **Note**
>
> Saat Anda membuat aplikasi menggunakan konsol, sumber daya AWS Identity and Access Management (IAM) dan Amazon CloudWatch Logs dibuat untuk Anda. Saat Anda membuat aplikasi menggunakan AWS CLI, Anda membuat sumber daya ini secara terpisah.

Topik

- Buat dan Jalankan Aplikasi (Konsol)
- Buat dan Jalankan Aplikasi (AWS CLI)

Buat dan Jalankan Aplikasi (Konsol)

Ikuti langkah-langkah ini untuk membuat, mengonfigurasi, memperbarui, dan menjalankan aplikasi menggunakan konsol.

Buat Aplikasi

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/flink
2. Pada dashboard Managed Service for Apache Flink, pilih Create Analytics Application.
3. Pada Layanan Terkelola untuk Apache Flink - Buat halaman aplikasi, berikan detail aplikasi sebagai berikut:
   - Untuk Application name (Nama aplikasi), masukkan **`MyApplication`**.
   - Untuk Description (Deskripsi), masukkan **`My java test app`**.
   - Untuk Runtime, pilih Apache Flink.
   - Biarkan menu tarik turun versi sebagai Apache Flink 1.8 (Versi yang Direkomendasikan).
4. Untuk Access permissions (Izin akses), pilih Create / update IAM role **`kinesis-analytics-MyApplication-us-west-2`** (Buat/perbarui IAM role).
5. Pilih Create application (Buat aplikasi).

> **Note**
>
> Saat membuat aplikasi Managed Service for Apache Flink menggunakan konsol, Anda memiliki opsi untuk membuat peran dan kebijakan IAM untuk aplikasi Anda. Aplikasi Anda menggunakan peran dan kebijakan ini untuk mengakses sumber daya dependen. Sumber daya IAM ini diberi nama menggunakan nama aplikasi dan Wilayah sebagai berikut:
>
> - Kebijakan: `kinesis-analytics-service-MyApplication-us-west-2`
> - Peran: `kinesisanalytics-MyApplication-us-west-2`

Edit Kebijakan IAM

Edit kebijakan IAM untuk menambahkan izin mengakses Kinesis data streams.

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.

2. Pilih Policies (Kebijakan). Pilih kebijakan **`kinesis-analytics-service-MyApplication-us-west-2`** yang dibuat konsol untuk Anda di bagian sebelumnya.
3. Di halaman Ringkasan, pilih Edit policy (Edit kebijakan). Pilih tab JSON.
4. Tambahkan bagian yang disorot dari contoh kebijakan berikut ke kebijakan. Ganti ID akun sampel (*012345678901*) dengan ID akun Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ka-app-code-username/aws-kinesis-analytics-java-
apps-1.0.jar"
            ]
        },
        {
            "Sid": "DescribeLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
            ]
        },
```

```
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

Konfigurasikan Aplikasi

1. Pada MyApplicationhalaman, pilih Konfigurasi.
2. Di halaman Konfigurasikan aplikasi, berikan Code location (Lokasi kode):
   - Untuk Bucket Amazon S3, masukkan **ka-app-code-*<username>***.
   - Untuk Jalur ke objek Amazon S3, masukkan **aws-kinesis-analytics-java-apps-1.0.jar**.
3. Di bawah Akses ke sumber daya aplikasi, untuk Access permissions (Izin akses), pilih Create / update IAM role **kinesis-analytics-MyApplication-us-west-2** (Pilih/perbarui IAM role).
4. Masukkan properti dan nilai aplikasi berikut:

ID Grup	Kunci	Nilai
ProducerConfigProperties	**flink.inputstream.initpos**	**LATEST**
ProducerConfigProperties	**aws.region**	**us-west-2**
ProducerConfigProperties	**AggregationEnabled**	**false**

5. Di bawah Pemantauan, pastikan Memantau tingkat metrik diatur ke Aplikasi.
6. Untuk CloudWatch logging, pilih kotak centang Aktifkan.
7. Pilih Perbarui.

> **Note**
>
> Saat Anda memilih untuk mengaktifkan CloudWatch pencatatan Amazon, Layanan Terkelola untuk Apache Flink membuat grup log dan aliran log untuk Anda. Nama-nama sumber daya ini adalah sebagai berikut:
>
> - Grup log: `/aws/kinesis-analytics/MyApplication`
> - Aliran log: `kinesis-analytics-log-stream`

Jalankan Aplikasi

1. Pada MyApplicationhalaman, pilih Jalankan. Konfirmasikan tindakan.
2. Ketika aplikasi berjalan, refresh halaman. Konsol menunjukkan Grafik aplikasi.

Hentikan Aplikasi

Pada MyApplicationhalaman, pilih Berhenti. Konfirmasikan tindakan.

Perbarui Aplikasi

Dengan menggunakan konsol, Anda dapat memperbarui pengaturan aplikasi seperti properti aplikasi, pengaturan pemantauan, dan lokasi atau nama file dari JAR aplikasi. Anda juga dapat memuat ulang aplikasi JAR dari bucket Amazon S3 jika Anda perlu memperbarui kode aplikasi.

Pada MyApplicationhalaman, pilih Konfigurasi. Perbarui pengaturan aplikasi dan pilih Update (Perbarui).

Buat dan Jalankan Aplikasi (AWS CLI)

Di bagian ini, Anda menggunakan AWS CLI untuk membuat dan menjalankan aplikasi Managed Service for Apache Flink. Managed Service untuk Apache Flink menggunakan `kinesisanalyticsv2` AWS CLI perintah untuk membuat dan berinteraksi dengan Managed Service untuk aplikasi Apache Flink.

Membuat Kebijakan Izin

> **Note**
>
> Anda harus membuat kebijakan izin dan peran untuk aplikasi Anda. Jika Anda tidak membuat sumber daya IAM ini, aplikasi Anda tidak dapat mengakses data dan aliran log.

Pertama, Anda membuat kebijakan izin dengan dua pernyataan: satu yang memberikan izin untuk tindakan `read` di aliran sumber, dan lainnya yang memberikan izin untuk tindakan `write` di aliran sink. Anda selanjutnya melampirkan kebijakan ke IAM role (yang Anda buat di bagian berikutnya). Jadi, ketika Layanan Terkelola untuk Apache Flink mengasumsikan peran tersebut, layanan memiliki izin yang diperlukan untuk membaca dari aliran sumber dan menulis ke aliran wastafel.

Gunakan kode berikut untuk membuat kebijakan izin `AKReadSourceStreamWriteSinkStream`. Ganti *`username`* dengan nama pengguna yang Anda gunakan untuk membuat bucket Amazon S3 untuk menyimpan kode aplikasi. Ganti ID akun di Amazon Resource Names (ARN) (*`012345678901`*) dengan ID akun Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3",
```

```
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": ["arn:aws:s3:::ka-app-code-username",
                "arn:aws:s3:::ka-app-code-username/*"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

Untuk step-by-step petunjuk membuat kebijakan izin, lihat Tutorial: Membuat dan Melampirkan Kebijakan Terkelola Pelanggan Pertama Anda di Panduan Pengguna IAM.

> **Note**
>
> Untuk mengakses layanan Amazon lainnya, Anda dapat menggunakan AWS SDK for Java. Layanan Terkelola untuk Apache Flink secara otomatis menyetel kredensyal yang diperlukan oleh SDK ke peran IAM eksekusi layanan yang terkait dengan aplikasi Anda. Tidak ada langkah tambahan yang diperlukan.

Buat IAM Role

Di bagian ini, Anda membuat peran IAM yang dapat diasumsikan oleh aplikasi Managed Service for Apache Flink untuk membaca aliran sumber dan menulis ke aliran sink.

Layanan Terkelola untuk Apache Flink tidak dapat mengakses aliran Anda tanpa izin. Anda memberikan izin ini melalui IAM role. Setiap IAM role memiliki dua kebijakan yang dilampirkan. Kebijakan kepercayaan memberikan izin Layanan Terkelola untuk Apache Flink untuk mengambil peran, dan kebijakan izin menentukan apa yang dapat dilakukan Layanan Terkelola untuk Apache Flink setelah mengambil peran.

Anda melampirkan kebijakan izin yang Anda buat di bagian sebelumnya ke peran ini.

Untuk membuat IAM role

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Dalam panel navigasi, pilih Roles (Peran), Create role (Buat Peran).
3. Di bawah Pilih tipe identitas tepercaya, pilih Layanan AWS. Di bawah Pilih layanan yang akan menggunakan peran ini, pilih Kinesis. Di bawah Pilih kasus penggunaan Anda, pilih Analitik Kinesis.

   Pilih Selanjutnya: Izin.
4. Di halaman Lampirkan kebijakan izin, pilih Next: Review (Selanjutnya: Tinjau). Anda melampirkan kebijakan izin setelah Anda membuat peran tersebut.
5. Di halaman Buat peran, masukkan **`MF-stream-rw-role`** untuk Role name (Nama peran). Pilih Create role (Buat peran).

   Sekarang Anda sudah membuat IAM role baru yang disebut `MF-stream-rw-role`. Berikutnya, Anda memperbarui kebijakan kepercayaan dan izin untuk peran.
6. Lampirkan kebijakan izin ke peran tersebut.

   > **Note**
   >
   > Untuk latihan ini, Managed Service for Apache Flink mengasumsikan peran ini untuk membaca data dari aliran data Kinesis (sumber) dan menulis output ke aliran data Kinesis lain. Jadi, Anda melampirkan kebijakan yang Anda buat di langkah sebelumnya, the section called “Membuat Kebijakan Izin”.

   a. Di halaman Ringkasan, pilih tab Permissions (Izin).

   b. Pilih Attach Policies (Lampirkan Kebijakan).

c. Di kotak pencarian, masukkan **`AKReadSourceStreamWriteSinkStream`** (kebijakan yang Anda buat bagian sebelumnya).

d. Pilih ReadSourceStreamWriteSinkStream kebijakan AK, dan pilih Lampirkan kebijakan.

Anda sekarang telah membuat peran eksekusi layanan yang digunakan aplikasi Anda untuk mengakses sumber daya. Buat catatan ARN peran baru.

Untuk step-by-step petunjuk cara membuat peran, lihat Membuat Peran IAM (Konsol) di Panduan Pengguna IAM.

Buat Managed Service untuk Apache Flink Application

1. Simpan kode JSON berikut ke file bernama `create_request.json`. Ganti ARN peran sampel dengan ARN untuk peran yang Anda buat sebelumnya. Ganti sufiks ARN bucket (*`username`*) dengan sufiks yang Anda pilih di bagian sebelumnya. Ganti ID akun sampel (*`012345678901`*) di peran eksekusi layanan dengan ID akun Anda.

```
{
    "ApplicationName": "test",
    "ApplicationDescription": "my java test app",
    "RuntimeEnvironment": "FLINK-1_8",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-stream-rw-role",
    "ApplicationConfiguration": {
        "ApplicationCodeConfiguration": {
            "CodeContent": {
                "S3ContentLocation": {
                    "BucketARN": "arn:aws:s3:::ka-app-code-username",
                    "FileKey": "aws-kinesis-analytics-java-apps-1.0.jar"
                }
            },
            "CodeContentType": "ZIPFILE"
        },
        "EnvironmentProperties":  {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "flink.stream.initpos" : "LATEST",
                    "aws.region" : "us-west-2",
                    "AggregationEnabled" : "false"
               }
```

```
            },
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2"
               }
            }
         ]
      }
   }
}
```

2. Jalankan tindakan `CreateApplication` dengan permintaan sebelumnya untuk membuat aplikasi:

```
aws kinesisanalyticsv2 create-application --cli-input-json file://
create_request.json
```

Aplikasi ini sekarang dibuat. Anda memulai aplikasi di langkah berikutnya.

Mulai Aplikasi

Di bagian ini, Anda menggunakan tindakan `StartApplication` untuk memulai aplikasi.

Untuk memulai aplikasi

1. Simpan kode JSON berikut ke file bernama `start_request.json`.

```
{
    "ApplicationName": "test",
    "RunConfiguration": {
        "ApplicationRestoreConfiguration": {
         "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"
         }
    }
}
```

2. Jalankan tindakan `StartApplication` dengan permintaan sebelumnya untuk memulai aplikasi:

```
aws kinesisanalyticsv2 start-application --cli-input-json file://start_request.json
```

Aplikasi sekarang berjalan. Anda dapat memeriksa metrik Layanan Terkelola untuk Apache Flink di CloudWatch konsol Amazon untuk memverifikasi bahwa aplikasi berfungsi.

Hentikan Aplikasi

Di bagian ini, Anda menggunakan tindakan `StopApplication` untuk menghentikan aplikasi.

Untuk menghentikan aplikasi

1. Simpan kode JSON berikut ke file bernama `stop_request.json`.

```
{
   "ApplicationName": "test"
}
```

2. Jalankan tindakan `StopApplication` dengan permintaan berikut untuk menghentikan aplikasi:

```
aws kinesisanalyticsv2 stop-application --cli-input-json file://stop_request.json
```

Aplikasi sekarang dihentikan.

Tambahkan Opsi CloudWatch Logging

Anda dapat menggunakan AWS CLI untuk menambahkan aliran CloudWatch log Amazon ke aplikasi Anda. Untuk informasi tentang menggunakan CloudWatch Log dengan aplikasi Anda, lihatthe section called “Menyiapkan Pencatatan”.

Perbarui Properti Lingkungan

Di bagian ini, Anda menggunakan tindakan `UpdateApplication` untuk mengubah properti lingkungan untuk aplikasi tanpa mengompilasi ulang kode aplikasi. Dalam contoh ini, Anda mengubah Wilayah aliran sumber dan tujuan.

Untuk memperbarui properti lingkungan untuk aplikasi

1. Simpan kode JSON berikut ke file bernama `update_properties_request.json`.

```
{"ApplicationName": "test",
   "CurrentApplicationVersionId": 1,
   "ApplicationConfigurationUpdate": {
      "EnvironmentPropertyUpdates": {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "flink.stream.initpos" : "LATEST",
                    "aws.region" : "us-west-2",
                    "AggregationEnabled" : "false"
               }
            },
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2"
               }
            }
         ]
      }
   }
}
```

2. Jalankan tindakan UpdateApplication dengan permintaan sebelumnya untuk memperbarui properti lingkungan:

```
aws kinesisanalyticsv2 update-application --cli-input-json file://
update_properties_request.json
```

Perbarui Kode Aplikasi

Ketika Anda perlu memperbarui kode aplikasi Anda dengan versi baru paket kode Anda, Anda menggunakan tindakan UpdateApplication AWS CLI.

> **Note**
>
> Untuk memuat versi baru kode aplikasi dengan nama file yang sama, Anda harus menentukan versi objek baru. Untuk informasi selengkapnya tentang menggunakan versi objek Amazon S3, lihat Mengaktifkan dan Menonaktifkan Versioning.

Untuk menggunakan AWS CLI, hapus paket kode Anda sebelumnya dari bucket Amazon S3 Anda, unggah versi baru, dan panggil `UpdateApplication`, yang menentukan bucket Amazon S3 dan nama objek yang sama, serta versi objek baru. Aplikasi akan memulai ulang dengan paket kode baru.

Permintaan sampel berikut untuk tindakan `UpdateApplication` memuat ulang kode aplikasi dan memulai ulang aplikasi. Perbarui `CurrentApplicationVersionId` ke versi aplikasi saat ini. Anda dapat memeriksa versi aplikasi saat ini menggunakan tindakan `ListApplications` atau `DescribeApplication`. Perbarui sufiks nama bucket (*<username>*) dengan sufiks yang Anda pilih di bagian the section called “Buat Dua Amazon Kinesis Data Streams”.

```
{
    "ApplicationName": "test",
    "CurrentApplicationVersionId": 1,
    "ApplicationConfigurationUpdate": {
        "ApplicationCodeConfigurationUpdate": {
            "CodeContentUpdate": {
                "S3ContentLocationUpdate": {
                    "BucketARNUpdate": "arn:aws:s3:::ka-app-code-username",
                    "FileKeyUpdate": "aws-kinesis-analytics-java-apps-1.0.jar",
                    "ObjectVersionUpdate": "SAMPLEUehYngP87ex1nzYIGYgfhypvDU"
                }
            }
        }
    }
}
```

Langkah Selanjutnya

Langkah 4: Bersihkan Sumber Daya AWS

Langkah 4: Bersihkan Sumber Daya AWS

Bagian ini mencakup prosedur untuk membersihkan sumber daya AWS yang dibuat dalam tutorial Memulai.

Topik ini berisi bagian-bagian berikut:

- Hapus Layanan Terkelola Anda untuk Aplikasi Apache Flink
- Hapus Kinesis Data Streams Anda
- Hapus Objek dan Bucket Amazon S3 Anda
- Hapus Sumber Daya IAM Anda
- Hapus CloudWatch Sumber Daya Anda

Hapus Layanan Terkelola Anda untuk Aplikasi Apache Flink

1. Buka konsol Kinesis di https://console.aws.amazon.com/kinesis.
2. Di panel Managed Service for Apache Flink, pilih. MyApplication
3. Pilih Configure (Konfigurasikan).
4. Di bagian Snapshots, pilih Disable (Nonaktifkan), lalu pilih Update (Perbarui).
5. Di halaman aplikasi, pilih Delete (Hapus), lalu konfirmasikan penghapusan.

Hapus Kinesis Data Streams Anda

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/flink
2. Di panel Kinesis Data Streams, pilih. ExampleInputStream
3. Di ExampleInputStreamhalaman, pilih Hapus Stream Kinesis dan kemudian konfirmasikan penghapusan.
4. Di halaman Kinesis streams, pilih, pilih Tindakan ExampleOutputStream, pilih Hapus, lalu konfirmasikan penghapusan.

Hapus Objek dan Bucket Amazon S3 Anda

1. Buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.
2. Pilih ka-app-code- ember. <username>
3. Pilih Delete (Hapus), lalu masukkan nama bucket untuk mengonfirmasi penghapusan.

Hapus Sumber Daya IAM Anda

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.

2. Di bilah navigasi, pilih Policies (Kebijakan).
3. Di kontrol filter, masukkan kinesis.
4. Pilih kebijakan kinesis-analytics-service- MyApplication -us-west-2.
5. Pilih Policy Actions (Tindakan Kebijakan), lalu pilih Delete (Hapus).
6. Di bilah navigasi, pilih Roles (Peran).
7. Pilih peran kinesis-analytics- -us-west-2. MyApplication
8. Pilih Delete role (Hapus peran), lalu konfirmasi penghapusan.

Hapus CloudWatch Sumber Daya Anda

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di bilah navigasi, pilih Logs.
3. Pilih grup log MyApplication/aws/kinesis-analytics/.
4. Pilih Delete Log Group (Hapus Grup Log), lalu konfirmasi penghapusan.

Memulai: Flink 1.6.2

Topik ini berisi versi Memulai (DataStream API) Tutorial yang menggunakan Apache Flink 1.6.2.

Topik

Komponen Layanan Terkelola untuk Apache Flink

Untuk memproses data, Managed Service untuk aplikasi Apache Flink Anda menggunakan aplikasi Java/Apache Maven atau Scala yang memproses input dan menghasilkan output menggunakan runtime Apache Flink.

Managed Service untuk Apache Flink memiliki komponen-komponen berikut:

- Properti runtime: Anda dapat menggunakan properti runtime untuk mengonfigurasi aplikasi Anda tanpa mengompilasi ulang kode aplikasi Anda.
- Source (Sumber): Aplikasi mengonsumsi data menggunakan sumber. Konektor sumber membaca data dari Kinesis data stream, bucket Amazon S3, dll. Untuk informasi selengkapnya, lihat Sumber.
- Operators (Operator): Aplikasi memproses data menggunakan satu atau beberapa operator. Operator dapat mengubah, memperkaya, atau menggabungkan data. Untuk informasi selengkapnya, lihat DataStream Operator API.
- Sink: Aplikasi menghasilkan data ke sumber eksternal menggunakan sink. Konektor sink menulis data ke aliran data Kinesis, aliran Firehose Data Kinesis, bucket Amazon S3, dll. Untuk informasi selengkapnya, lihat Sink.

Setelah Anda membuat, mengompilasi, dan mengemas aplikasi Anda, Anda mengunggah paket kode ke bucket Amazon Simple Storage Service (Amazon S3). Anda kemudian membuat Layanan Terkelola untuk aplikasi Apache Flink. Anda meneruskan di lokasi paket kode, Kinesis data stream sebagai sumber data streaming, dan biasanya lokasi streaming atau file yang menerima data yang diproses dari aplikasi.

Prasyarat untuk Menyelesaikan Latihan

Untuk menyelesaikan langkah-langkah di panduan ini, Anda harus memiliki hal-hal berikut:

- Java Development Kit (JDK) versi 8. Atur variabel lingkungan `JAVA_HOME` untuk menunjuk ke lokasi penginstalan JDK Anda.
- Sebaiknya gunakan lingkungan pengembangan (seperti Eclipse Java Neon atau IntelliJ Idea) untuk mengembangkan dan mengompilasi aplikasi Anda.
- Klien Git. Instal klien Git jika Anda belum menginstalnya.
- Plugin Compiler Apache Maven. Maven harus berada di jalur kerja Anda. Untuk menguji instalasi Apache Maven Anda, masukkan hal berikut:

```
$ mvn -version
```

Untuk memulai, buka Langkah 1: Siapkan Akun AWS dan Buat Pengguna Administrator.

Langkah 1: Siapkan Akun AWS dan Buat Pengguna Administrator

Mendaftar Akun AWS

Jika Anda tidak memiliki Akun AWS, selesaikan langkah-langkah berikut untuk membuatnya.

Untuk mendaftar Akun AWS

1. Buka https://portal.aws.amazon.com/billing/signup.
2. Ikuti petunjuk secara online.

   Anda akan diminta untuk menerima panggilan telepon dan memasukkan kode verifikasi pada keypad telepon sebagai bagian dari prosedur pendaftaran.

   Saat Anda mendaftar Akun AWS, Pengguna root akun AWS akan dibuat. Pengguna root memiliki akses ke semua Layanan AWS dan sumber daya dalam akun. Sebagai praktik terbaik keamanan, tetapkan akses administratif ke pengguna administratif, dan hanya gunakan pengguna root untuk melakukan tugas yang memerlukan akses pengguna root.

AWS akan mengirimkan email konfirmasi kepada Anda setelah proses pendaftaran selesai. Anda dapat melihat aktivitas akun saat ini dan mengelola akun dengan mengunjungi https://aws.amazon.com/ dan memilih Akun Saya.

Membuat pengguna administratif

Setelah mendaftar Akun AWS, amankan Pengguna root akun AWS, aktifkan AWS IAM Identity Center, dan buat sebuah pengguna administratif sehingga Anda tidak menggunakan pengguna root untuk tugas sehari-hari.

Mengamankan Pengguna root akun AWS Anda

1. Masuk ke AWS Management Console sebagai pemilik akun dengan memilih Pengguna root dan memasukkan alamat email Akun AWS Anda. Di halaman berikutnya, masukkan kata sandi Anda.

   Untuk bantuan masuk menggunakan pengguna root, lihat Masuk sebagai pengguna root dalam Panduan Pengguna AWS Sign-In.
2. Aktifkan autentikasi multi-faktor (MFA) untuk pengguna root Anda.

   Untuk petunjuknya, silakan lihat Mengaktifkan perangkat MFA virtual untuk pengguna root Akun AWS Anda (konsol) dalam Panduan Pengguna IAM.

Membuat pengguna administratif

1. Aktifkan Pusat Identitas IAM.

   Untuk mendapatkan petunjuk, silakan lihat Mengaktifkan AWS IAM Identity Center di Panduan Pengguna AWS IAM Identity Center.
2. Di Pusat Identitas IAM, berikan akses administratif ke sebuah pengguna administratif.

   Untuk mendapatkan tutorial tentang menggunakan Direktori Pusat Identitas IAM sebagai sumber identitas Anda, silakan lihat Mengonfigurasi akses pengguna dengan Direktori Pusat Identitas IAM default di Panduan Pengguna AWS IAM Identity Center.

Masuk sebagai pengguna administratif

- Untuk masuk dengan pengguna Pusat Identitas IAM, gunakan URL masuk yang dikirim ke alamat email Anda saat Anda membuat pengguna Pusat Identitas IAM.

  Untuk bantuan masuk menggunakan pengguna Pusat Identitas IAM, lihat Masuk ke portal akses AWS dalam Panduan Pengguna AWS Sign-In.

Memberikan akses terprogram

Pengguna membutuhkan akses terprogram jika mereka ingin berinteraksi dengan AWS luar dari AWS Management Console. Cara memberikan akses terprogram bergantung pada jenis pengguna yang mengakses AWS.

Untuk memberi pengguna akses terprogram, pilih salah satu opsi berikut.

Pengguna mana yang membutuhkan akses terprogram?	Untuk	Oleh
Identitas tenaga kerja (Pengguna yang dikelola di Pusat Identitas IAM)	Gunakan kredensial sementara untuk menandatangani permintaan terprogram ke AWS CLI, SDK AWS, atau API AWS.	Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan. • Untuk AWS CLI, lihat Mengonfigurasi AWS CLI untuk menggunakan AWS

Pengguna mana yang membutuhkan akses terprogram?	Untuk	Oleh
		IAM Identity Center di Panduan Pengguna AWS Command Line Interface. • Untuk SDK AWS, alat, dan API AWS, lihat Autentikasi Pusat Identitas IAM di Panduan Referensi SDK dan Alat AWS.
IAM	Gunakan kredensial sementara untuk menandatangani permintaan terprogram ke AWS CLI, SDK AWS, atau API AWS.	Mengikuti petunjuk dalam Menggunakan kredensial sementara dengan sumber daya AWS di Panduan Pengguna IAM.

Pengguna mana yang membutuhkan akses terprogram?	Untuk	Oleh
IAM	(Tidak direkomendasikan) Gunakan kredensial jangka panjang untuk menandatangani permintaan terprogram ke AWS CLI, SDK AWS, atau API AWS.	Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan. • Untuk AWS CLI, lihat Mengautentikasi menggunakan kredensial pengguna IAM di Panduan Pengguna AWS Command Line Interface. • Untuk SDK dan alat AWS, lihat Mengautentikasi menggunakan kredensial jangka panjang di Panduan Referensi SDK dan Alat AWS. • Untuk API AWS, lihat Mengelola kunci akses untuk pengguna IAM di Panduan Pengguna IAM.

Langkah 2: Siapkan AWS Command Line Interface (AWS CLI)

Pada langkah ini, Anda mengunduh dan mengonfigurasi AWS CLI untuk digunakan dengan Managed Service for Apache Flink.

> **Note**
>
> Latihan memulai dalam panduan ini mengasumsikan Anda menggunakan kredensial administrator (`adminuser`) di akun Anda untuk melakukan operasi.

> **Note**
>
> Jika sudah menginstal AWS CLI, Anda mungkin perlu meningkatkan untuk mendapatkan fungsionalitas terbaru. Untuk informasi selengkapnya, lihat Menginstal AWS Command Line Interface dalam Panduan Pengguna AWS Command Line Interface. Untuk memeriksa versi AWS CLI, jalankan perintah berikut.
>
> ```
> aws --version
> ```
>
> Latihan dalam tutorial ini memerlukan versi AWS CLI berikut atau yang lebih baru:
>
> ```
> aws-cli/1.16.63
> ```

Untuk mengatur AWS CLI

1. Unduh dan konfigurasikan AWS CLI. Untuk instruksinya, lihat topik berikut di AWS Command Line InterfacePanduan Pengguna:
   - Menginstal AWS Command Line Interface
   - Mengonfigurasi AWS CLI
2. Tambahkan profil bernama untuk pengguna administrator dalam file AWS CLI `config`. Anda dapat menggunakan profil ini saat menjalankan perintah AWS CLI. Untuk informasi selengkapnya tentang profil yang diberi nama, lihat Profil yang Diberi Nama dalam Panduan Pengguna AWS Command Line Interface.

   ```
   [profile adminuser]
   aws_access_key_id = adminuser access key ID
   aws_secret_access_key = adminuser secret access key
   region = aws-region
   ```

   Untuk daftar AWS Wilayah yang tersedia, lihat Wilayah dan Titik Akhir di. Referensi Umum Amazon Web Services

> **Note**
>
> Kode dan perintah contoh dalam tutorial ini menggunakan Wilayah US West (Oregon). Untuk menggunakan Wilayah yang berbeda, ubah Wilayah dalam kode dan perintah untuk tutorial ini ke Wilayah yang ingin Anda gunakan.

3. Verifikasikan penyiapan dengan memasukkan perintah bantuan berikut pada prompt perintah.

```
aws help
```

Setelah Anda mengatur AWS akun danAWS CLI, Anda dapat mencoba latihan berikutnya, di mana Anda mengkonfigurasi aplikasi sampel dan menguji end-to-end pengaturan.

Langkah Selanjutnya

Langkah 3: Buat dan Jalankan Layanan Terkelola untuk aplikasi Apache Flink

Langkah 3: Buat dan Jalankan Layanan Terkelola untuk aplikasi Apache Flink

Dalam latihan ini, Anda membuat Layanan Terkelola untuk aplikasi Apache Flink dengan aliran data sebagai sumber dan wastafel.

Bagian ini berisi langkah-langkah berikut.

- Buat Dua Amazon Kinesis Data Streams
- Tulis Catatan Sampel ke Aliran Input
- Unduh dan Periksa Kode Java Streaming Apache Flink
- Kompilasi Kode Aplikasi
- Unggah Kode Java Streaming Apache Flink
- Buat dan Jalankan Managed Service untuk Apache Flink Application

Buat Dua Amazon Kinesis Data Streams

Sebelum Anda membuat Layanan Terkelola untuk aplikasi Apache Flink untuk latihan ini, buat dua aliran data Kinesis (dan). `ExampleInputStream` `ExampleOutputStream` Aplikasi Anda menggunakan aliran ini untuk sumber aplikasi dan aliran tujuan.

Anda dapat membuat aliran ini menggunakan konsol Amazon Kinesis atau perintah AWS CLI berikut. Untuk instruksi konsol, lihat Membuat dan Memperbarui Aliran Data di Panduan Developer Amazon Kinesis Data Streams.

Untuk membuat aliran data AWS CLI

1. Untuk membuat aliran pertama (`ExampleInputStream`), gunakan perintah `create-stream` AWS CLI Amazon Kinesis berikut.

```
$ aws kinesis create-stream \
--stream-name ExampleInputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

2. Untuk membuat aliran kedua yang digunakan aplikasi untuk menulis output, jalankan perintah yang sama, yang mengubah nama aliran menjadi `ExampleOutputStream`.

```
$ aws kinesis create-stream \
--stream-name ExampleOutputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

Tulis Catatan Sampel ke Aliran Input

Di bagian ini, Anda menggunakan script Python untuk menulis catatan sampel ke aliran untuk diproses aplikasi.

> **Note**
>
> Bagian ini memerlukan AWS SDK for Python (Boto).

1. Buat file bernama `stock.py` dengan konten berikut:

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        "EVENT_TIME": datetime.datetime.now().isoformat(),
        "TICKER": random.choice(["AAPL", "AMZN", "MSFT", "INTC", "TBV"]),
        "PRICE": round(random.random() * 100, 2),
    }


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(data),
 PartitionKey="partitionkey"
        )


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

2. Selanjutnya dalam tutorial ini, Anda menjalankan skrip `stock.py` untuk mengirim data ke aplikasi.

```
$ python stock.py
```

Unduh dan Periksa Kode Java Streaming Apache Flink

Kode aplikasi Java untuk contoh ini tersedia dari GitHub. Untuk mengunduh kode aplikasi, lakukan hal berikut:

1. Kloning repositori jarak jauh menggunakan perintah berikut:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

2. Buka direktori `amazon-kinesis-data-analytics-java-examples/GettingStarted_1_6` tersebut.

Perhatikan hal tentang kode aplikasi berikut:

- File Project Object Model (pom.xml) berisi informasi tentang konfigurasi dan dependensi aplikasi, termasuk Layanan Terkelola untuk pustaka Apache Flink.
- File `BasicStreamingJob.java` berisi metode `main` yang menentukan fungsionalitas aplikasi.
- Aplikasi menggunakan sumber Kinesis untuk membaca dari aliran sumber. Cuplikan berikut ini membuat sumber Kinesis:

```
return env.addSource(new FlinkKinesisConsumer<>(inputStreamName,
                new SimpleStringSchema(), inputProperties));
```

- Aplikasi Anda membuat konektor sumber dan sink untuk mengakses sumber daya eksternal menggunakan objek `StreamExecutionEnvironment`.
- Aplikasi membuat konektor sumber dan sink menggunakan properti statis. Untuk menggunakan properti aplikasi dinamis, gunakan metode `createSourceFromApplicationProperties` dan `createSinkFromApplicationProperties` untuk membuat konektor. Metode ini membaca properti aplikasi untuk mengonfigurasi konektor.

  Untuk informasi selengkapnya tentang properti runtime, lihat Properti Runtime.

Kompilasi Kode Aplikasi

Di bagian ini, Anda menggunakan compiler Apache Maven untuk membuat kode Java untuk aplikasi. Untuk informasi tentang menginstal Apache Maven dan Java Development Kit (JDK), lihat Prasyarat untuk Menyelesaikan Latihan.

> **Note**
>
> Untuk menggunakan konektor Kinesis dengan versi Apache Flink sebelum 1.11, Anda perlu mengunduh kode sumber untuk konektor dan membangunnya seperti yang dijelaskan dalam Dokumentasi Apache Flink.

Untuk mengompikasi kode aplikasi

1. Untuk menggunakan kode aplikasi Anda, Anda mengompilasi dan mengemasnya ke dalam file JAR. Anda dapat mengompilasi dan mengemas kode Anda dengan salah satu dari dua cara:

   - Gunakan alat Maven baris perintah. Buat file JAR Anda dengan menjalankan perintah berikut di direktori yang berisi file `pom.xml`:

     ```
     mvn package
     ```

     > **Note**
     >
     > Parameter -DFLink.version tidak diperlukan untuk Managed Service untuk Apache Flink Runtime versi 1.0.1; hanya diperlukan untuk versi 1.1.0 dan yang lebih baru. Untuk informasi selengkapnya, lihat the section called “Menentukan Versi Apache Flink Aplikasi Anda”.

   - Menyiapkan lingkungan pengembangan Anda. Lihat dokumentasi lingkungan pengembangan Anda untuk detail.

   Anda dapat mengunggah paket Anda sebagai file JAR, atau Anda dapat mengompresi paket Anda dan mengunggahnya sebagai file ZIP. Jika Anda membuat aplikasi Anda menggunakan AWS CLI, Anda menentukan tipe konten kode Anda (JAR atau ZIP).

2. Jika ada kesalahan saat mengompilasi, pastikan variabel lingkungan `JAVA_HOME`Anda diatur dengan benar.

Jika aplikasi berhasil mengompilasi, file berikut dibuat:

```
target/aws-kinesis-analytics-java-apps-1.0.jar
```

Unggah Kode Java Streaming Apache Flink

Pada bagian ini, Anda membuat bucket Amazon Simple Storage Service (Amazon S3) dan mengunggah kode aplikasi Anda.

Untuk mengunggah kode aplikasi

1. Buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.
2. Pilih Buat bucket.
3. Masukkan **ka-app-code-*<username>*** di bidang Bucket name (Nama bucket). Tambahkan sufiks ke nama bucket, seperti nama pengguna Anda, untuk membuatnya unik secara global. Pilih Next (Selanjutnya).
4. Di langkah Konfigurasi opsi, jangan ubah pengaturan, dan pilih Next (Selanjutnya).
5. Di langkah Atur izin, jangan ubah pengaturan, dan pilih Next (Selanjutnya).
6. Pilih Create bucket (Buat bucket).
7. Di konsol Amazon S3, pilih bucket ka-app-code -, dan pilih Unggah. <username>
8. Di langkah Pilih file, pilih Add files (Tambahkan berkas). Navigasikan ke file `aws-kinesis-analytics-java-apps-1.0.jar` yang Anda buat di langkah sebelumnya. Pilih Berikutnya.
9. Di langkah Atur izin, jangan ubah pengaturan. Pilih Berikutnya.
10. Di langkah Atur properti, jangan ubah pengaturan. Pilih Upload (Unggah).

Kode aplikasi Anda sekarang disimpan di bucket Amazon S3 yang dapat diakses aplikasi Anda.

Buat dan Jalankan Managed Service untuk Apache Flink Application

Anda dapat membuat dan menjalankan Layanan Terkelola untuk aplikasi Apache Flink menggunakan konsol atau aplikasi. AWS CLI

> **Note**
>
> Saat Anda membuat aplikasi menggunakan konsol, sumber daya AWS Identity and Access Management (IAM) dan Amazon CloudWatch Logs dibuat untuk Anda. Saat Anda membuat aplikasi menggunakan AWS CLI, Anda membuat sumber daya ini secara terpisah.

Topik

Buat dan Jalankan Aplikasi (Konsol)

Ikuti langkah-langkah ini untuk membuat, mengonfigurasi, memperbarui, dan menjalankan aplikasi menggunakan konsol.

Buat Aplikasi

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/flink
2. Pada dashboard Managed Service for Apache Flink, pilih Create Analytics Application.
3. Pada Layanan Terkelola untuk Apache Flink - Buat halaman aplikasi, berikan detail aplikasi sebagai berikut:
   - Untuk Application name (Nama aplikasi), masukkan **`MyApplication`**.
   - Untuk Description (Deskripsi), masukkan **`My java test app`**.
   - Untuk Runtime, pilih Apache Flink.

     > **Note**
     >
     > Managed Service untuk Apache Flink menggunakan Apache Flink versi 1.8.2 atau 1.6.2.

   - Ubah versi menu tarik turun ke Apache Flink 1.6.
4. Untuk Access permissions (Izin akses), pilih Create / update IAM role **`kinesis-analytics-MyApplication-us-west-2`** (Buat/perbarui IAM role).
5. Pilih Create application (Buat aplikasi).

> **Note**
>
> Saat membuat aplikasi Managed Service for Apache Flink menggunakan konsol, Anda memiliki opsi untuk membuat peran dan kebijakan IAM untuk aplikasi Anda. Aplikasi Anda menggunakan peran dan kebijakan ini untuk mengakses sumber daya dependen. Sumber daya IAM ini diberi nama menggunakan nama aplikasi dan Wilayah sebagai berikut:
>
> - Kebijakan: `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*

- Peran: `kinesisanalytics-MyApplication-us-west-2`

Edit Kebijakan IAM

Edit kebijakan IAM untuk menambahkan izin mengakses Kinesis data streams.

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pilih Policies (Kebijakan). Pilih kebijakan **`kinesis-analytics-service-MyApplication-us-west-2`** yang dibuat konsol untuk Anda di bagian sebelumnya.
3. Di halaman Ringkasan, pilih Edit policy (Edit kebijakan). Pilih tab JSON.
4. Tambahkan bagian yang disorot dari contoh kebijakan berikut ke kebijakan. Ganti ID akun sampel (*012345678901*) dengan ID akun Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ka-app-code-username/java-getting-started-1.0.jar"
            ]
        },
        {
            "Sid": "DescribeLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
```

```
            "Action": [
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
            ]
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

Konfigurasikan Aplikasi

1. Pada MyApplicationhalaman, pilih Konfigurasi.
2. Di halaman Konfigurasikan aplikasi, berikan Code location (Lokasi kode):
   - Untuk Bucket Amazon S3, masukkan **ka-app-code-*<username>***.

- Untuk Jalur ke objek Amazon S3, masukkan **`java-getting-started-1.0.jar`**.

3. Di bawah Akses ke sumber daya aplikasi, untuk Access permissions (Izin akses), pilih Create / update IAM role **`kinesis-analytics-MyApplication-us-west-2`** (Pilih/perbarui IAM role).
4. Masukkan properti dan nilai aplikasi berikut:

ID Grup	Kunci	Nilai
`ProducerConfigProperties`	**`flink.inputstream.initpos`**	**`LATEST`**
`ProducerConfigProperties`	**`aws.region`**	**`us-west-2`**
`ProducerConfigProperties`	**`AggregationEnabled`**	**`false`**

5. Di bawah Pemantauan, pastikan Memantau tingkat metrik diatur ke Aplikasi.
6. Untuk CloudWatch logging, pilih kotak centang Aktifkan.
7. Pilih Perbarui.

> **Note**
>
> Saat Anda memilih untuk mengaktifkan CloudWatch pencatatan Amazon, Layanan Terkelola untuk Apache Flink membuat grup log dan aliran log untuk Anda. Nama-nama sumber daya ini adalah sebagai berikut:
>
> - Grup log: `/aws/kinesis-analytics/MyApplication`
> - Aliran log: `kinesis-analytics-log-stream`

Jalankan Aplikasi

1. Pada MyApplicationhalaman, pilih Jalankan. Konfirmasikan tindakan.
2. Ketika aplikasi berjalan, refresh halaman. Konsol menunjukkan Grafik aplikasi.

Hentikan Aplikasi

Pada MyApplicationhalaman, pilih Berhenti. Konfirmasikan tindakan.

Perbarui Aplikasi

Dengan menggunakan konsol, Anda dapat memperbarui pengaturan aplikasi seperti properti aplikasi, pengaturan pemantauan, dan lokasi atau nama file dari JAR aplikasi. Anda juga dapat memuat ulang aplikasi JAR dari bucket Amazon S3 jika Anda perlu memperbarui kode aplikasi.

Pada MyApplicationhalaman, pilih Konfigurasi. Perbarui pengaturan aplikasi dan pilih Update (Perbarui).

Buat dan Jalankan Aplikasi (AWS CLI)

Di bagian ini, Anda menggunakan AWS CLI untuk membuat dan menjalankan aplikasi Managed Service for Apache Flink. Managed Service untuk Apache Flink menggunakan `kinesisanalyticsv2` AWS CLI perintah untuk membuat dan berinteraksi dengan Managed Service untuk aplikasi Apache Flink.

Membuat Kebijakan Izin

Pertama, Anda membuat kebijakan izin dengan dua pernyataan: satu yang memberikan izin untuk tindakan `read` di aliran sumber, dan lainnya yang memberikan izin untuk tindakan `write` di aliran sink. Anda selanjutnya melampirkan kebijakan ke IAM role (yang Anda buat di bagian berikutnya). Jadi, ketika Layanan Terkelola untuk Apache Flink mengasumsikan peran tersebut, layanan memiliki izin yang diperlukan untuk membaca dari aliran sumber dan menulis ke aliran wastafel.

Gunakan kode berikut untuk membuat kebijakan izin `AKReadSourceStreamWriteSinkStream`. Ganti *`username`* dengan nama pengguna yang Anda gunakan untuk membuat bucket Amazon S3 untuk menyimpan kode aplikasi. Ganti ID akun di Amazon Resource Names (ARN) (*`012345678901`*) dengan ID akun Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
```

```
                "s3:GetObjectVersion"
            ],
            "Resource": ["arn:aws:s3:::ka-app-code-username",
                "arn:aws:s3:::ka-app-code-username/*"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

Untuk step-by-step petunjuk membuat kebijakan izin, lihat Tutorial: Membuat dan Melampirkan Kebijakan Terkelola Pelanggan Pertama Anda di Panduan Pengguna IAM.

> **Note**
>
> Untuk mengakses layanan Amazon lainnya, Anda dapat menggunakan AWS SDK for Java. Layanan Terkelola untuk Apache Flink secara otomatis menyetel kredensyal yang diperlukan oleh SDK ke peran IAM eksekusi layanan yang terkait dengan aplikasi Anda. Tidak ada langkah tambahan yang diperlukan.

Buat IAM Role

Di bagian ini, Anda membuat peran IAM yang dapat diasumsikan oleh aplikasi Managed Service for Apache Flink untuk membaca aliran sumber dan menulis ke aliran sink.

Layanan Terkelola untuk Apache Flink tidak dapat mengakses aliran Anda tanpa izin. Anda memberikan izin ini melalui IAM role. Setiap IAM role memiliki dua kebijakan yang dilampirkan.

Kebijakan kepercayaan memberikan izin Layanan Terkelola untuk Apache Flink untuk mengambil peran, dan kebijakan izin menentukan apa yang dapat dilakukan Layanan Terkelola untuk Apache Flink setelah mengambil peran.

Anda melampirkan kebijakan izin yang Anda buat di bagian sebelumnya ke peran ini.

Untuk membuat IAM role

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Dalam panel navigasi, pilih Roles (Peran), Create role (Buat Peran).
3. Di bawah Pilih tipe identitas tepercaya, pilih Layanan AWS. Di bawah Pilih layanan yang akan menggunakan peran ini, pilih Kinesis. Di bawah Pilih kasus penggunaan Anda, pilih Analitik Kinesis.

   Pilih Selanjutnya: Izin.
4. Di halaman Lampirkan kebijakan izin, pilih Next: Review (Selanjutnya: Tinjau). Anda melampirkan kebijakan izin setelah Anda membuat peran tersebut.
5. Di halaman Buat peran, masukkan **`MF-stream-rw-role`** untuk Role name (Nama peran). Pilih Create role (Buat peran).

   Sekarang Anda sudah membuat IAM role baru yang disebut `MF-stream-rw-role`. Berikutnya, Anda memperbarui kebijakan kepercayaan dan izin untuk peran.
6. Lampirkan kebijakan izin ke peran tersebut.

   > **Note**
   >
   > Untuk latihan ini, Managed Service for Apache Flink mengasumsikan peran ini untuk membaca data dari aliran data Kinesis (sumber) dan menulis output ke aliran data Kinesis lain. Jadi, Anda melampirkan kebijakan yang Anda buat di langkah sebelumnya, the section called “Membuat Kebijakan Izin”.

   a. Di halaman Ringkasan, pilih tab Permissions (Izin).
   b. Pilih Attach Policies (Lampirkan Kebijakan).
   c. Di kotak pencarian, masukkan **`AKReadSourceStreamWriteSinkStream`** (kebijakan yang Anda buat bagian sebelumnya).
   d. Pilih ReadSourceStreamWriteSinkStream kebijakan AK, dan pilih Lampirkan kebijakan.

Anda sekarang telah membuat peran eksekusi layanan yang digunakan aplikasi Anda untuk mengakses sumber daya. Buat catatan ARN peran baru.

Untuk step-by-step petunjuk cara membuat peran, lihat Membuat Peran IAM (Konsol) di Panduan Pengguna IAM.

Buat Managed Service untuk Apache Flink Application

1. Simpan kode JSON berikut ke file bernama `create_request.json`. Ganti ARN peran sampel dengan ARN untuk peran yang Anda buat sebelumnya. Ganti sufiks ARN bucket (*username*) dengan sufiks yang Anda pilih di bagian sebelumnya. Ganti ID akun sampel (*012345678901*) di peran eksekusi layanan dengan ID akun Anda.

```
{
    "ApplicationName": "test",
    "ApplicationDescription": "my java test app",
    "RuntimeEnvironment": "FLINK-1_6",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-stream-rw-role",
    "ApplicationConfiguration": {
        "ApplicationCodeConfiguration": {
            "CodeContent": {
                "S3ContentLocation": {
                    "BucketARN": "arn:aws:s3:::ka-app-code-username",
                    "FileKey": "java-getting-started-1.0.jar"
                }
            },
            "CodeContentType": "ZIPFILE"
        },
        "EnvironmentProperties":  {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "flink.stream.initpos" : "LATEST",
                    "aws.region" : "us-west-2",
                    "AggregationEnabled" : "false"
               }
            },
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2"
               }
```

```
            }
         ]
      }
   }
}
```

2. Jalankan tindakan [CreateApplication](#) dengan permintaan sebelumnya untuk membuat aplikasi:

```
aws kinesisanalyticsv2 create-application --cli-input-json file://
create_request.json
```

Aplikasi ini sekarang dibuat. Anda memulai aplikasi di langkah berikutnya.

Mulai Aplikasi

Di bagian ini, Anda menggunakan tindakan [StartApplication](#) untuk memulai aplikasi.

Untuk memulai aplikasi

1. Simpan kode JSON berikut ke file bernama `start_request.json`.

```
{
    "ApplicationName": "test",
    "RunConfiguration": {
        "ApplicationRestoreConfiguration": {
         "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"
         }
    }
}
```

2. Jalankan tindakan [StartApplication](#) dengan permintaan sebelumnya untuk memulai aplikasi:

```
aws kinesisanalyticsv2 start-application --cli-input-json file://start_request.json
```

Aplikasi sekarang berjalan. Anda dapat memeriksa metrik Layanan Terkelola untuk Apache Flink di CloudWatch konsol Amazon untuk memverifikasi bahwa aplikasi berfungsi.

Hentikan Aplikasi

Di bagian ini, Anda menggunakan tindakan `StopApplication` untuk menghentikan aplikasi.

Untuk menghentikan aplikasi

1. Simpan kode JSON berikut ke file bernama `stop_request.json`.

```
{
   "ApplicationName": "test"
}
```

2. Jalankan tindakan `StopApplication` dengan permintaan berikut untuk menghentikan aplikasi:

```
aws kinesisanalyticsv2 stop-application --cli-input-json file://stop_request.json
```

Aplikasi sekarang dihentikan.

Tambahkan Opsi CloudWatch Logging

Anda dapat menggunakan AWS CLI untuk menambahkan aliran CloudWatch log Amazon ke aplikasi Anda. Untuk informasi tentang menggunakan CloudWatch Log dengan aplikasi Anda, lihatthe section called “Menyiapkan Pencatatan”.

Perbarui Properti Lingkungan

Di bagian ini, Anda menggunakan tindakan `UpdateApplication` untuk mengubah properti lingkungan untuk aplikasi tanpa mengompilasi ulang kode aplikasi. Dalam contoh ini, Anda mengubah Wilayah aliran sumber dan tujuan.

Untuk memperbarui properti lingkungan untuk aplikasi

1. Simpan kode JSON berikut ke file bernama `update_properties_request.json`.

```
{"ApplicationName": "test",
   "CurrentApplicationVersionId": 1,
   "ApplicationConfigurationUpdate": {
      "EnvironmentPropertyUpdates": {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ProducerConfigProperties",
```

```
                "PropertyMap" : {
                     "flink.stream.initpos" : "LATEST",
                     "aws.region" : "us-west-2",
                     "AggregationEnabled" : "false"
                }
            },
            {
                "PropertyGroupId": "ConsumerConfigProperties",
                "PropertyMap" : {
                     "aws.region" : "us-west-2"
                }
            }
         ]
      }
   }
}
```

2. Jalankan tindakan `UpdateApplication` dengan permintaan sebelumnya untuk memperbarui properti lingkungan:

```
aws kinesisanalyticsv2 update-application --cli-input-json file://
update_properties_request.json
```

Perbarui Kode Aplikasi

Ketika Anda perlu memperbarui kode aplikasi Anda dengan versi baru paket kode Anda, Anda menggunakan tindakan `UpdateApplication` AWS CLI.

Untuk menggunakan AWS CLI, hapus paket kode Anda sebelumnya dari bucket Amazon S3 Anda, unggah versi baru, dan panggil `UpdateApplication`, yang menentukan bucket Amazon S3 dan nama objek yang sama. Aplikasi akan memulai ulang dengan paket kode baru.

Permintaan sampel berikut untuk tindakan `UpdateApplication` memuat ulang kode aplikasi dan memulai ulang aplikasi. Perbarui `CurrentApplicationVersionId` ke versi aplikasi saat ini. Anda dapat memeriksa versi aplikasi saat ini menggunakan tindakan `ListApplications` atau `DescribeApplication`. Perbarui sufiks nama bucket (*<username>*) dengan sufiks yang Anda pilih di bagian the section called “Buat Dua Amazon Kinesis Data Streams”.

```
{
    "ApplicationName": "test",
    "CurrentApplicationVersionId": 1,
```

```
    "ApplicationConfigurationUpdate": {
        "ApplicationCodeConfigurationUpdate": {
            "CodeContentUpdate": {
                "S3ContentLocationUpdate": {
                    "BucketARNUpdate": "arn:aws:s3:::ka-app-code-username",
                    "FileKeyUpdate": "java-getting-started-1.0.jar"
                }
            }
        }
    }
}
```

Langkah 4: Bersihkan Sumber Daya AWS

Bagian ini mencakup prosedur untuk membersihkan sumber daya AWS yang dibuat dalam tutorial Memulai.

Topik ini berisi bagian-bagian berikut:

- Hapus Layanan Terkelola Anda untuk Aplikasi Apache Flink
- Hapus Kinesis Data Streams Anda
- Hapus Objek dan Bucket Amazon S3 Anda
- Hapus Sumber Daya IAM Anda
- Hapus CloudWatch Sumber Daya Anda

Hapus Layanan Terkelola Anda untuk Aplikasi Apache Flink

1. Buka konsol Kinesis di https://console.aws.amazon.com/kinesis.
2. Di panel Managed Service for Apache Flink, pilih. MyApplication
3. Pilih Configure (Konfigurasikan).
4. Di bagian Snapshots, pilih Disable (Nonaktifkan), lalu pilih Update (Perbarui).
5. Di halaman aplikasi, pilih Delete (Hapus), lalu konfirmasikan penghapusan.

Hapus Kinesis Data Streams Anda

1. Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/flink
2. Di panel Kinesis Data Streams, pilih. ExampleInputStream

3. Di ExampleInputStreamhalaman, pilih Hapus Stream Kinesis dan kemudian konfirmasikan penghapusan.
4. Di halaman Kinesis streams, pilih, pilih Tindakan ExampleOutputStream, pilih Hapus, lalu konfirmasikan penghapusan.

Hapus Objek dan Bucket Amazon S3 Anda

1. Buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.
2. Pilih ka-app-code- ember. <username>
3. Pilih Delete (Hapus), lalu masukkan nama bucket untuk mengonfirmasi penghapusan.

Hapus Sumber Daya IAM Anda

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di bilah navigasi, pilih Policies (Kebijakan).
3. Di kontrol filter, masukkan kinesis.
4. Pilih kebijakan kinesis-analytics-service- MyApplication -us-west-2.
5. Pilih Policy Actions (Tindakan Kebijakan), lalu pilih Delete (Hapus).
6. Di bilah navigasi, pilih Roles (Peran).
7. Pilih peran kinesis-analytics- -us-west-2. MyApplication
8. Pilih Delete role (Hapus peran), lalu konfirmasi penghapusan.

Hapus CloudWatch Sumber Daya Anda

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di bilah navigasi, pilih Logs.
3. Pilih grup log MyApplication/aws/kinesis-analytics/.
4. Pilih Delete Log Group (Hapus Grup Log), lalu konfirmasi penghapusan.

Pengaturan Apache Flink

Managed Service untuk Apache Flink adalah implementasi dari kerangka Apache Flink. Layanan Terkelola untuk Apache Flink menggunakan nilai default yang dijelaskan di bagian ini. Beberapa nilai ini dapat diatur oleh Managed Service untuk aplikasi Apache Flink dalam kode, dan lainnya tidak dapat diubah.

Topik ini berisi bagian-bagian berikut:

Apache Flink Konfigurasi

Managed Service for Apache Flink menyediakan konfigurasi Flink default yang terdiri dari nilai yang direkomendasikan Apache Flink untuk sebagian besar properti dan beberapa berdasarkan profil aplikasi umum. Untuk informasi selengkapnya tentang konfigurasi Flink, lihat Konfigurasi. Konfigurasi default yang disediakan layanan berfungsi untuk sebagian besar aplikasi. Namun, jika Anda perlu mengubah properti konfigurasi Flink untuk meningkatkan kinerja untuk aplikasi tertentu dengan paralelisme tinggi, memori tinggi dan penggunaan status, atau mengaktifkan fitur debugging baru di Apache Flink, Anda dapat mengubah properti tertentu dengan meminta kasus dukungan. Untuk informasi selengkapnya, lihat Pusat Dukungan AWS. Anda dapat memeriksa konfigurasi saat ini untuk aplikasi Anda menggunakan Apache Flink Dashboard.

Backend Status

Layanan Terkelola untuk Apache Flink menyimpan data sementara di backend status. Managed Service untuk Apache Flink menggunakan RocksDB. StateBackend Memanggil `setStateBackend` untuk mengatur backend yang berbeda tidak memiliki pengaruh.

Kami mengaktifkan fitur berikut pada backend status:

- Snapshot backend status tambahan
- Snapshot backend status asinkron
- Pemulihan lokal titik pemeriksaan

Di Managed Service for Apache Flink, `state.backend.rocksdb.ttl.compaction.filter.enabled` konfigurasi diaktifkan secara default. Dengan menggunakan filter ini, Anda dapat memperbarui kode aplikasi Anda untuk mengaktifkan strategi pembersihan pemadatan. Untuk informasi selengkapnya, lihat TTL Status di Flink 1.8.0 di Dokumentasi Apache Flink.

Untuk informasi selengkapnya tentang backend status, lihat Backend Status di Dokumentasi Apache Flink.

Checkpointing

Layanan Terkelola untuk Apache Flink menggunakan konfigurasi pos pemeriksaan default dengan nilai-nilai berikut. Beberapa nilai ini dapat diubah. Anda harus mengatur CheckpointConfiguration.ConfigurationType`CUSTOM`untuk Layanan Terkelola untuk Apache Flink untuk menggunakan nilai checkpointing yang dimodifikasi.

Pengaturan	Bisa dimodifikasi?	Bagaimana	nilai default
CheckpointingEnabled	Dapat diubah	Buat Aplikasi Perbarui Aplikasi AWS CloudFormation	Benar
CheckpointInterval	Dapat diubah	Buat Aplikasi Perbarui Aplikasi AWS CloudFormation	60000
MinPauseBetweenCheckpoints	Dapat diubah	Buat Aplikasi Perbarui Aplikasi	5000

Pengaturan	Bisa dimodifikasi?	Bagaimana	nilai default
		AWS CloudFormation	
Pos pemeriksaan tidak sejajar	Dapat diubah	Kasus Support	Salah
Jumlah Titik Pemeriksaan Konkuren	Tidak Dapat Dimodifikasi	T/A	1
Mode Checkpointing	Tidak Dapat Dimodifikasi	T/A	Tepat Satu Kali
Kebijakan Penyimpanan Titik Pemeriksaan	Tidak Dapat Dimodifikasi	T/A	Pada Kegagalan
Waktu Habis Titik Pemeriksaan	Tidak Dapat Dimodifikasi	T/A	60 menit
Titik Pemeriksaan Maks. yang Disimpan	Tidak Dapat Dimodifikasi	T/A	1
Mulai Ulang Strategi	Tidak Dapat Dimodifikasi	T/A	Penundaan Tetap, dengan pengulangan tidak terbatas setiap 10 detik.
Lokasi Titik Pemeriksaan dan Titik Simpan	Tidak Dapat Dimodifikasi	T/A	Kami menyimpan data titik pemeriksaan dan titik simpan yang tahan lama ke bucket S3 milik layanan.
Ambang Memori Backend Status	Tidak Dapat Dimodifikasi	T/A	1048576

Savepointing

Secara default, ketika memulihkan dari titik simpan, operasi lanjutkan akan mencoba memetakan semua status titik simpan kembali ke program yang Anda pulihkan. Jika Anda menghapus operator, secara default, memulihkan dari titik simpan yang memiliki data yang sesuai dengan operator yang hilang akan gagal. Anda dapat mengizinkan operasi berhasil dengan mengatur AllowNonRestoredStateparameter aplikasi FlinkRunConfigurationketrue. Ini akan memungkinkan operasi lanjutkan melewati status yang tidak dapat dipetakan ke program baru.

Untuk informasi selengkapnya, lihat Mengizinkan Status yang Tidak Dipulihkan di Dokumentasi Apache Flink.

Ukuran Tumpukan

Managed Service untuk Apache Flink mengalokasikan masing-masing KPU 3 GiB tumpukan JVM, dan cadangan 1 GiB untuk alokasi kode asli. Untuk informasi tentang meningkatkan kapasitas aplikasi Anda, lihat the section called “Penskalaan”.

Untuk informasi selengkapnya tentang ukuran tumpukan JVM, lihat Konfigurasi di Dokumentasi Apache Flink.

Buffer debloating

Buffer debloating dapat membantu aplikasi dengan tekanan balik tinggi. Jika aplikasi Anda mengalami pos pemeriksaan/savepoint yang gagal, mengaktifkan fitur ini bisa berguna. Untuk melakukan ini, mintalah kasus dukungan.

Untuk informasi selengkapnya, lihat Mekanisme Debloating Buffer di dokumentasi Apache Flink.

Properti konfigurasi Flink yang dapat dimodifikasi

Berikut ini adalah pengaturan konfigurasi Flink Anda dapat memodifikasi menggunakan kasus dukungan. Anda dapat memodifikasi lebih dari satu properti pada satu waktu, dan untuk beberapa aplikasi pada saat yang sama dengan menentukan awalan aplikasi. Jika ada properti konfigurasi Flink lain di luar daftar ini yang ingin Anda ubah, harap tentukan properti yang tepat dalam kasus Anda.

Toleransi Kesalahan

`restart-strategy:`

`restart-strategy.fixed-delay.delay:`

Pos pemeriksaan dan Backend Negara

`state.backend:`

`state.backend.fs.memory-threshold:`

`state.backend.incremental:`

Checkpointing

`execution.checkpointing.unaligned:`

Metrik Asli RocksDB

Metrik Asli RocksDB tidak dikirim ke. CloudWatch Setelah diaktifkan, metrik ini dapat diakses baik dari dasbor Flink atau Flink REST API dengan perkakas khusus.

Managed Service for Apache Flink memungkinkan pelanggan mengakses Flink REST API terbaru (atau versi yang didukung yang Anda gunakan) dalam mode read-only menggunakan API. CreateApplicationPresignedUrl API ini digunakan oleh dasbor Flink sendiri tetapi juga dapat digunakan oleh alat pemantauan khusus.

`state.backend.rocksdb.compaction.style:`

`state.backend.rocksdb.memory.partitioned-index-filters:`

`state.backend.rocksdb.metrics.actual-delayed-write-rate:`

`state.backend.rocksdb.metrics.background-errors:`

`state.backend.rocksdb.metrics.block-cache-capacity:`

`state.backend.rocksdb.metrics.block-cache-pinned-usage:`

`state.backend.rocksdb.metrics.block-cache-usage:`

```
state.backend.rocksdb.metrics.column-family-as-variable:

state.backend.rocksdb.metrics.compaction-pending:

state.backend.rocksdb.metrics.cur-size-active-mem-table:

state.backend.rocksdb.metrics.cur-size-all-mem-tables:

state.backend.rocksdb.metrics.estimate-live-data-size:

state.backend.rocksdb.metrics.estimate-num-keys:

state.backend.rocksdb.metrics.estimate-pending-compaction-bytes:

state.backend.rocksdb.metrics.estimate-table-readers-mem:

state.backend.rocksdb.metrics.is-write-stopped:

state.backend.rocksdb.metrics.mem-table-flush-pending:

state.backend.rocksdb.metrics.num-deletes-active-mem-table:

state.backend.rocksdb.metrics.num-deletes-imm-mem-tables:

state.backend.rocksdb.metrics.num-entries-active-mem-table:

state.backend.rocksdb.metrics.num-entries-imm-mem-tables:

state.backend.rocksdb.metrics.num-immutable-mem-table:

state.backend.rocksdb.metrics.num-live-versions:

state.backend.rocksdb.metrics.num-running-compactions:

state.backend.rocksdb.metrics.num-running-flushes:

state.backend.rocksdb.metrics.num-snapshots:

state.backend.rocksdb.metrics.size-all-mem-tables:

state.backend.rocksdb.thread.num:
```

Opsi Backend Status Tingkat Lanjut

```
state.storage.fs.memory-threshold:
```

TaskManager Opsi Lengkap

```
task.cancellation.timeout:

taskmanager.jvm-exit-on-oom:

taskmanager.numberOfTaskSlots:

taskmanager.slot.timeout:

taskmanager.network.memory.fraction:

taskmanager.network.memory.max:

taskmanager.network.request-backoff.initial:

taskmanager.network.request-backoff.max:

taskmanager.network.memory.buffer-debloat.enabled:

taskmanager.network.memory.buffer-debloat.period:

taskmanager.network.memory.buffer-debloat.samples:

taskmanager.network.memory.buffer-debloat.threshold-percentages:
```

Konfigurasi Memori

```
taskmanager.memory.jvm-metaspace.size:

taskmanager.memory.jvm-overhead.fraction:

taskmanager.memory.jvm-overhead.max:

taskmanager.memory.managed.consumer-weights:

taskmanager.memory.managed.fraction:

taskmanager.memory.network.fraction:

taskmanager.memory.network.max:

taskmanager.memory.segment-size:
```

```
taskmanager.memory.task.off-heap.size:
```

RPC/Akka

```
akka.ask.timeout:

akka.client.timeout:

akka.framesize:

akka.lookup.timeout:

akka.tcp.timeout:
```

Klien

```
client.timeout:
```

Opsi Cluster Tingkat Lanjut

```
cluster.intercept-user-system-exit:

cluster.processes.halt-on-fatal-error:
```

Konfigurasi Sistem File

```
fs.s3.connection.maximum:

fs.s3a.connection.maximum:

fs.s3a.threads.max:

s3.upload.max.concurrent.uploads:
```

Opsi Toleransi Kesalahan Tingkat Lanjut

```
heartbeat.timeout:

jobmanager.execution.failover-strategy:
```

Konfigurasi memori

```
jobmanager.memory.heap.size:
```

Metrik

`metrics.latency.interval:`

Opsi Lanjutan untuk titik akhir REST dan Klien

`rest.flamegraph.enabled:`

`rest.server.numThreads:`

Opsi Keamanan SSL Tingkat Lanjut

`security.ssl.internal.handshake-timeout:`

Opsi Penjadwalan Lanjutan

`slot.request.timeout:`

Opsi Lanjutan untuk UI Web Flink

`web.timeout:`

Melihat properti Flink yang dikonfigurasi

Anda dapat melihat properti Apache Flink yang telah Anda konfigurasikan sendiri atau diminta untuk dimodifikasi melalui kasus dukungan melalui Dasbor Apache Flink dan mengikuti langkah-langkah berikut:

1. Pergi ke Dasbor Flink
2. Pilih Job Manager di panel navigasi sisi kiri.
3. Pilih Konfigurasi untuk melihat daftar properti Flink.

Mengkonfigurasi Layanan Terkelola untuk Apache Flink untuk mengakses Sumber Daya di Amazon VPC

Anda dapat mengonfigurasi Layanan Terkelola untuk aplikasi Apache Flink untuk terhubung ke subnet pribadi di cloud pribadi virtual (VPC) di akun Anda. Gunakan Amazon Virtual Private Cloud (Amazon VPC) untuk membuat jaringan privat untuk sumber daya seperti basis data, instans cache, atau layanan internal. Sambungkan aplikasi Anda ke VPC untuk mengakses sumber daya privat selama eksekusi.

Topik ini berisi bagian-bagian berikut:

- Konsep Amazon VPC
- Izin Aplikasi VPC
- Akses Internet dan Layanan untuk Layanan Terkelola yang terhubung dengan VPC untuk aplikasi Apache Flink
- Layanan Terkelola untuk Apache Flink VPC API
- Contoh: Menggunakan VPC untuk Mengakses Data di Klaster Amazon MSK

Konsep Amazon VPC

Amazon VPC adalah lapisan jaringan untuk Amazon EC2. Jika Anda pengguna baru Amazon EC2, lihat Apa itu Amazon EC2? di Panduan Pengguna Amazon EC2 untuk Instans Linux untuk mendapatkan gambaran umumnya.

Berikut ini adalah konsep utama untuk VPC:

- Virtual private cloud (VPC) adalah jaringan virtual yang didedikasikan untuk akun AWS Anda.
- Sebuah subnet adalah rentang alamat IP di VPC Anda.
- Tabel rute berisi serangkaian aturan, yang disebut rute, yang digunakan untuk menentukan ke mana lalu lintas jaringan diarahkan.
- Gateway internet diskalakan secara horizontal, redundan, dan merupakan komponen VPC yang sangat tersedia yang mengizinkan komunikasi antara VPC Anda dan internet. Oleh karena itu, gateway internet tidak menimbulkan risiko ketersediaan atau kendala bandwidth pada lalu lintas jaringan Anda.

- SEBUAHTitik akhir VPCmemungkinkan Anda untuk menghubungkan VPC Anda secara pribadi ke yang didukungAWSlayanan dan layanan titik akhir VPC yang didukung olehPrivateLinktanpa memerlukan gateway internet, perangkat NAT, koneksi VPN, atauAWS Direct Connectkoneksi. Instans di VPC Anda tidak memerlukan alamat IP publik untuk berkomunikasi dengan sumber daya di layanan. Lalu lintas antara VPC Anda dan layanan lainnya tidak meninggalkan jaringan Amazon.

Untuk informasi selengkapnya tentang layanan Amazon VPC, lihat Panduan Pengguna Amazon Virtual Private Cloud.

Layanan Terkelola untuk Apache Flink menciptakanantarmuka jaringan elastisdi salah satu subnet yang disediakan dalam konfigurasi VPC Anda untuk aplikasi. Jumlah antarmuka jaringan elastis yang dibuat di subnet VPC Anda dapat bervariasi, bergantung pada paralelisme dan paralelisme per KPU aplikasi. Untuk informasi selengkapnya tentang penskalaan aplikasi, lihat Penskalaan.

> **Note**
>
> Konfigurasi VPC tidak didukung untuk aplikasi SQL.

> **Note**
>
> Layanan Terkelola untuk layanan Apache Flink mengelola pos pemeriksaan dan status snapshot untuk aplikasi yang memiliki konfigurasi VPC.

Izin Aplikasi VPC

Bagian ini menjelaskan kebijakan izin yang diperlukan aplikasi Anda untuk bekerja dengan VPC Anda. Untuk informasi selengkapnya tentang menggunakan kebijakan izin, lihat Identity and Access Management untuk Amazon Managed Service untuk Apache Flink.

Kebijakan izin berikut memberi aplikasi Anda izin yang diperlukan untuk berinteraksi dengan VPC. Untuk menggunakan kebijakan izin ini, tambahkan ke peran eksekusi aplikasi Anda.

Kebijakan Izin untuk Mengakses Amazon VPC

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VPCReadOnlyPermissions",
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeVpcs",
        "ec2:DescribeSubnets",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeDhcpOptions"
      ],
      "Resource": "*"
    },
    {
      "Sid": "ENIReadWritePermissions",
      "Effect": "Allow",
      "Action": [
        "ec2:CreateNetworkInterface",
        "ec2:CreateNetworkInterfacePermission",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DeleteNetworkInterface"
      ],
      "Resource": "*"
    }

    ]
}
```

> **Note**
>
> Saat Anda menentukan sumber daya aplikasi menggunakan konsol (sepertiCloudWatchLog atau Amazon VPC), konsol memodifikasi peran eksekusi aplikasi Anda untuk memberikan izin untuk mengakses sumber daya tersebut. Anda hanya perlu mengubah peran eksekusi aplikasi Anda secara manual jika Anda membuat aplikasi Anda tanpa menggunakan konsol tersebut.

Akses Internet dan Layanan untuk Layanan Terkelola yang terhubung dengan VPC untuk aplikasi Apache Flink

Secara default, ketika Anda menghubungkan Layanan Terkelola untuk aplikasi Apache Flink ke VPC di akun Anda, itu tidak memiliki akses ke internet kecuali VPC menyediakan akses. Jika aplikasi memerlukan akses internet, hal berikut harus benar:

- Layanan Terkelola untuk aplikasi Apache Flink seharusnya hanya dikonfigurasi dengan subnet pribadi.
- VPC harus berisi gateway NAT atau instans di subnet publik.
- Rute harus ada untuk lalu lintas keluar dari subnet privat ke gateway NAT di subnet publik.

> **Note**
>
> Beberapa layanan menawarkan VPC endpoint. Anda dapat menggunakan VPC endpoint untuk terhubung ke layanan Amazon dari dalam VPC tanpa akses internet.

Apakah subnet publik atau privat bergantung pada tabel rute. Setiap tabel rute memiliki rute default, yang menentukan hop berikutnya untuk paket yang memiliki tujuan publik.

- Untuk Subnet privat: Rute default menunjuk ke gateway NAT (nat-...) atau instans NAT (eni-...).
- Untuk Subnet publik: Rute default menunjuk ke gateway internet (igw-...).

Setelah Anda mengonfigurasi VPC Anda dengan subnet publik (dengan NAT) dan satu atau beberapa subnet privat, lakukan hal berikut untuk mengidentifikasi subnet privat dan publik Anda:

- Di konsol VPC, dari panel navigasi, pilih Subnets (Subnet).
- Pilih subnet, lalu pilih tab Route Table (Tabel Rute). Verifikasi rute default:
  - Subnet publik: Tujuan: 0.0.0.0/0, Target: igw-…
  - Subnet privat: Tujuan: 0.0.0.0/0, Target: nat-… atau eni-…

Untuk mengaitkan Layanan Terkelola untuk aplikasi Apache Flink dengan subnet pribadi:

- Buka Layanan Terkelola untuk konsol Apache Flink di https://console.aws.amazon.com/flink

- PadaLayanan Terkelola untuk aplikasi Apache Flinkhalaman, pilih aplikasi Anda, dan pilihDetail aplikasi.
- Di halaman untuk aplikasi Anda, pilih Configure (Konfigurasikan).
- Di bagian VPC Connectivity (Konektivitas VPC), pilih VPC yang akan dikaitkan dengan aplikasi Anda. Pilih subnet dan grup keamanan yang terkait dengan VPC Anda yang ingin digunakan aplikasi untuk mengakses sumber daya VPC.
- Pilih Update (Perbarui).

Informasi Terkait

Membuat VPC dengan Subnet Publik dan Pribadi

Dasar-dasar gateway NAT

Layanan Terkelola untuk Apache Flink VPC API

Gunakan Layanan Terkelola berikut untuk operasi Apache Flink API untuk mengelola VPC untuk aplikasi Anda. Untuk informasi tentang penggunaan Layanan Terkelola untuk Apache Flink API, lihatKode Contoh API.

CreateApplication

GunakanCreateApplicationtindakan untuk menambahkan konfigurasi VPC ke aplikasi Anda selama pembuatan.

Kode permintaan contoh untuk tindakan `CreateApplication` berikut mencakup konfigurasi VPC ketika aplikasi dibuat:

```
{
  "ApplicationName":"MyApplication",
  "ApplicationDescription":"My-Application-Description",
  "RuntimeEnvironment":"FLINK-1_15",
  "ServiceExecutionRole":"arn:aws:iam::123456789123:role/myrole",
  "ApplicationConfiguration": {
    "ApplicationCodeConfiguration":{
      "CodeContent":{
        "S3ContentLocation":{
```

```
          "BucketARN":"arn:aws:s3:::mybucket",
          "FileKey":"myflink.jar",
          "ObjectVersion":"AbCdEfGhIjKlMnOpQrStUvWxYz12345"
        }
      },
      "CodeContentType":"ZIPFILE"
    },
      "FlinkApplicationConfiguration":{
      "ParallelismConfiguration":{
        "ConfigurationType":"CUSTOM",
        "Parallelism":2,
        "ParallelismPerKPU":1,
        "AutoScalingEnabled":true
      }
    },
  "VpcConfigurations": [
        {
           "SecurityGroupIds": [ "sg-0123456789abcdef0" ],
           "SubnetIds": [ "subnet-0123456789abcdef0" ]
        }
      ]
  }
}
```

AddApplicationVpcConfiguration

GunakanAddApplicationVpcConfigurationtindakan untuk menambahkan konfigurasi VPC ke aplikasi Anda setelah dibuat.

Kode permintaan contoh untuk tindakan `AddApplicationVpcConfiguration` berikut menambahkan konfigurasi VPC ke aplikasi yang sudah ada.

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 9,
   "VpcConfiguration": {
      "SecurityGroupIds": [ "sg-0123456789abcdef0" ],
      "SubnetIds": [ "subnet-0123456789abcdef0" ]
   }
}
```

DeleteApplicationVpcConfiguration

GunakanDeleteApplicationVpcConfigurationtindakan untuk menghapus konfigurasi VPC dari aplikasi Anda.

Kode permintaan contoh untuk tindakan `AddApplicationVpcConfiguration` berikut menghapus konfigurasi VPC yang ada dari aplikasi.

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 9,
   "VpcConfigurationId": "1.1"
}
```

UpdateApplication

GunakanUpdateApplicationtindakan untuk memperbarui semua konfigurasi VPC aplikasi sekaligus.

Kode permintaan contoh untuk tindakan `UpdateApplication` berikut memperbarui semua konfigurasi VPC untuk aplikasi.

```
{
   "ApplicationConfigurationUpdate": {
      "VpcConfigurationUpdates": [
         {
            "SecurityGroupIdUpdates": [ "sg-0123456789abcdef0" ],
            "SubnetIdUpdates": [ "subnet-0123456789abcdef0" ],
            "VpcConfigurationId": "2.1"
         }
      ]
   },
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 9
}
```

Contoh: Menggunakan VPC untuk Mengakses Data di Klaster Amazon MSK

Untuk tutorial lengkap tentang cara mengakses data dari Klaster Amazon MSK di VPC, lihat Replikasi MSK.

Pemecahan Masalah Layanan Terkelola untuk Apache Flink

Berikut ini dapat membantu Anda memecahkan masalah yang mungkin Anda temui dengan Amazon Managed Service for Apache Flink.

Topik

Penyelesaian Masalah Pengembangan

Topik

Grafik Api Apache Flink

Grafik Flame diaktifkan secara default pada aplikasi di Managed Service untuk versi Apache Flink yang mendukungnya. Grafik Api dapat memengaruhi kinerja aplikasi jika Anda membiarkan grafik tetap terbuka, seperti yang disebutkan dalam dokumentasi Flink.

Jika Anda ingin menonaktifkan Flame Graphs untuk aplikasi Anda, buat case untuk memintanya dinonaktifkan untuk ARN aplikasi Anda. Untuk informasi selengkapnya, lihat Pusat AWS Dukungan.

Masalah Penyedia Kredensi dengan konektor EFO 1.15.2

Ada masalah yang diketahui dengan versi konektor EFO Kinesis Data Streams hingga 1.15.2 `FlinkKinesisConsumer` di mana konfigurasi tidak menghormati. `Credential Provider` Konfigurasi yang valid diabaikan karena masalah, yang mengakibatkan penyedia `AUTO` kredensi digunakan. Hal ini dapat menyebabkan masalah menggunakan akses lintas akun ke Kinesis menggunakan konektor EFO.

Untuk mengatasi kesalahan ini, gunakan konektor EFO versi 1.15.3 atau lebih tinggi.

Aplikasi dengan konektor Kinesis yang tidak didukung

Layanan Terkelola untuk Apache Flink untuk Apache Flink versi 1.15 akan secara otomatis menolak aplikasi dari memulai atau memperbarui jika mereka menggunakan versi Konektor Kinesis yang tidak didukung (pra-versi 1.15.2) yang dibundel ke dalam JAR atau arsip aplikasi (ZIP).

Kesalahan Penolakan

Anda akan melihat kesalahan berikut saat mengirimkan panggilan aplikasi buat/perbarui melalui:

```
An error occurred (InvalidArgumentException) when calling the CreateApplication
 operation: An unsupported Kinesis connector version has been detected in the
 application. Please update flink-connector-kinesis to any version equal to or newer
 than 1.15.2.
For more information refer to connector fix: https://issues.apache.org/jira/browse/
FLINK-23528
```

Langkah-langkah untuk memulihkan

- Perbarui ketergantungan aplikasi pada`flink-connector-kinesis`. Jika Anda menggunakan Maven sebagai alat pembuatan proyek Anda, ikuti. Perbarui ketergantungan Maven Jika Anda menggunakan Gradle, ikutiMemperbarui ketergantungan Gradle .
- Paket ulang aplikasi.
- Unggah ke bucket Amazon S3.
- Kirim ulang permintaan aplikasi buat/perbarui dengan aplikasi yang direvisi yang baru saja diunggah ke bucket Amazon S3.
- Jika Anda terus melihat pesan kesalahan yang sama, periksa kembali dependensi aplikasi Anda. Jika masalah berlanjut, silakan buat tiket dukungan.

Perbarui ketergantungan Maven

1. Buka proyek`pom.xml`.
2. Temukan dependensi proyek. Mereka terlihat seperti:

```
<project>

    ...

    <dependencies>

        ...

        <dependency>
            <groupId>org.apache.flink</groupId>
            <artifactId>flink-connector-kinesis</artifactId>
        </dependency>

        ...

    </dependencies>

    ...

</project>
```

3. Perbarui `flink-connector-kinesis` ke versi yang sama dengan atau lebih baru dari 1.15.2. Misalnya:

```
<project>

    ...

    <dependencies>

        ...

        <dependency>
            <groupId>org.apache.flink</groupId>
            <artifactId>flink-connector-kinesis</artifactId>
            <version>1.15.2</version>
        </dependency>
```

```
        ...

    </dependencies>

    ...

</project>
```

Memperbarui ketergantungan Gradle

1. Buka proyek `build.gradle` (atau `build.gradle.kts` untuk aplikasi Kotlin).
2. Temukan dependensi proyek. Mereka terlihat seperti:

```
...

dependencies {

    ...

    implementation("org.apache.flink:flink-connector-kinesis")

    ...

}

...
```

3. Perbarui `flink-connector-kinesis` ke versi yang sama dengan atau lebih baru dari 1.15.2. Misalnya:

```
...

dependencies {

    ...

    implementation("org.apache.flink:flink-connector-kinesis:1.15.2")

    ...

}
```

```
...
```

Kesalahan Kompilasi: "Could not resolve dependencies for project" ("Tidak dapat menyelesaikan dependensi untuk proyek")

Untuk mengkompilasi Layanan Terkelola untuk aplikasi sampel Apache Flink, Anda harus terlebih dahulu mengunduh dan mengkompilasi konektor Apache Flink Kinesis dan menambahkannya ke repositori Maven lokal Anda. Jika konektor belum ditambahkan ke repositori Anda, kesalahan kompilasi yang mirip dengan berikut akan muncul:

```
Could not resolve dependencies for project your project name: Failure to
 find org.apache.flink:flink-connector-kinesis_2.11:jar:1.8.2 in https://
repo.maven.apache.org/maven2 was cached in the local repository, resolution will not be
 reattempted until the update interval of central has elapsed or updates are forced
```

Untuk mengatasi kesalahan ini, Anda harus mengunduh kode sumber Apache Flink (versi 1.8.2 dari https://flink.apache.org/downloads.html) untuk konektor. Untuk petunjuk tentang cara mengunduh, mengumpulkan, dan menginstal kode sumber Apache Flink, lihat the section called “Menggunakan Konektor Apache Flink Kinesis Streams dengan versi Apache Flink sebelumnya”.

Pilihan Tidak Valid: "kinesisanalyticsv2"

Untuk menggunakan v2 dari Managed Service for Apache Flink API, Anda memerlukan versi terbaru dari AWS Command Line Interface ()AWS CLI.

Untuk informasi tentang meningkatkan AWS CLI, lihat Menginstal AWS Command Line Interface di Panduan Pengguna AWS Command Line Interface.

UpdateApplication Tindakan Tidak Memuat Ulang Kode Aplikasi

UpdateApplicationTindakan tidak akan memuat ulang kode aplikasi dengan nama file yang sama jika tidak ada versi objek S3 yang ditentukan. Untuk memuat ulang kode aplikasi dengan nama file yang sama, aktifkan versioning pada bucket S3 Anda, dan tentukan versi objek baru menggunakan parameter `ObjectVersionUpdate`. Untuk informasi selengkapnya tentang mengaktifkan versioning objek di bucket S3, lihat Mengaktifkan dan Menonaktifkan Versioning.

S3 StreamingFileSink FileNotFoundExceptions

Layanan Terkelola untuk aplikasi Apache Flink dapat berjalan ke file bagian dalam proses `FileNotFoundException` saat memulai dari snapshot jika file bagian dalam proses yang dirujuk oleh savepoint-nya tidak ada. Ketika mode kegagalan ini terjadi, status operator aplikasi Managed Service for Apache Flink biasanya tidak dapat dipulihkan dan harus dimulai ulang tanpa menggunakan snapshot. `SKIP_RESTORE_FROM_SNAPSHOT` Lihat contoh stacktrace berikut:

```
java.io.FileNotFoundException: No such file or directory: s3://your-s3-bucket/pathj/
INSERT/2023/4/19/7/_part-2-1234_tmp_12345678-1234-1234-1234-123456789012
        at
 org.apache.hadoop.fs.s3a.S3AFileSystem.s3GetFileStatus(S3AFileSystem.java:2231)
        at
 org.apache.hadoop.fs.s3a.S3AFileSystem.innerGetFileStatus(S3AFileSystem.java:2149)
        at
 org.apache.hadoop.fs.s3a.S3AFileSystem.getFileStatus(S3AFileSystem.java:2088)
        at org.apache.hadoop.fs.s3a.S3AFileSystem.open(S3AFileSystem.java:699)
        at org.apache.hadoop.fs.FileSystem.open(FileSystem.java:950)
        at
 org.apache.flink.fs.s3hadoop.HadoopS3AccessHelper.getObject(HadoopS3AccessHelper.java:98)
        at
 org.apache.flink.fs.s3.common.writer.S3RecoverableMultipartUploadFactory.recoverInProgressPart
...
```

Flink `StreamingFileSink` menulis catatan ke sistem file yang didukung oleh abstraksi Flink. FileSystem Mengingat bahwa aliran yang masuk dapat tidak dibatasi, data diatur ke dalam file bagian dengan ukuran terbatas dengan file baru ditambahkan saat data ditulis. Kebijakan siklus hidup dan rollover bagian menentukan waktu, ukuran, dan penamaan file bagian.

> **Note**
>
> Untuk informasi selengkapnya, lihat Siklus hidup file bagian.

Selama checkpointing dan savepointing (snapshotting), semua file yang Tertunda diganti namanya dan dikomit. Namun, file bagian dalam proses tidak dikomit tetapi diganti namanya dan referensi mereka disimpan dalam pos pemeriksaan atau metadata savepoint untuk digunakan saat memulihkan pekerjaan. File bagian dalam proses ini pada akhirnya akan bergulir ke Pending, diganti namanya, dan dilakukan oleh pos pemeriksaan atau savepoint berikutnya.

Berikut ini adalah akar penyebab dan mitigasi untuk file bagian dalam proses yang hilang:

- Snapshot basi digunakan untuk memulai Layanan Terkelola untuk aplikasi Apache Flink - hanya snapshot sistem terbaru yang diambil saat aplikasi dihentikan atau diperbarui yang dapat digunakan untuk memulai Layanan Terkelola untuk aplikasi Apache Flink dengan Amazon S3. StreamingFileSink Untuk menghindari kelas kegagalan ini, gunakan snapshot sistem terbaru.
  - Ini terjadi misalnya ketika Anda memilih snapshot yang dibuat menggunakan `CreateSnapshot` alih-alih Snapshot yang dipicu sistem selama berhenti atau memperbarui. Savepoint snapshot yang lebih lama menyimpan out-of-date referensi ke file bagian dalam proses yang telah diganti namanya dan dilakukan oleh pos pemeriksaan atau savepoint berikutnya.
  - Ini juga dapat terjadi ketika snapshot yang dipicu sistem dari acara Stop/Update non-terbaru dipilih. Contohnya adalah aplikasi dengan snapshot sistem dinonaktifkan tetapi telah `RESTORE_FROM_LATEST_SNAPSHOT` dikonfigurasi. Umumnya, Layanan Terkelola untuk aplikasi Apache Flink dengan Amazon StreamingFileSink S3 harus selalu mengaktifkan dan mengonfigurasi snapshot sistem. `RESTORE_FROM_LATEST_SNAPSHOT`
- File bagian dalam proses dihapus — Karena file bagian yang sedang berlangsung terletak di bucket S3, file tersebut dapat dihapus oleh komponen atau aktor lain yang memiliki akses ke bucket.
  - Hal ini dapat terjadi jika Anda telah menghentikan aplikasi terlalu lama dan file bagian dalam proses yang dirujuk oleh savepoint aplikasi Anda telah dihapus oleh kebijakan siklus hidup bucket S3. MultiPartUpload Untuk menghindari kelas kegagalan ini, pastikan bahwa kebijakan siklus hidup S3 Bucket MPU Anda mencakup periode yang cukup besar untuk kasus penggunaan Anda.
  - Ini juga dapat terjadi ketika file bagian dalam proses telah dihapus secara manual atau oleh salah satu komponen sistem Anda yang lain. Untuk menghindari kelas kegagalan ini, pastikan bahwa file bagian dalam proses tidak dihapus oleh aktor atau komponen lain.
- Kondisi balapan di mana pos pemeriksaan otomatis dipicu setelah savepoint - Ini memengaruhi Layanan Terkelola untuk versi Apache Flink hingga dan termasuk 1.13. Masalah ini diperbaiki di Layanan Terkelola untuk Apache Flink versi 1.15; memigrasikan aplikasi Anda ke Layanan Terkelola untuk Apache Flink versi 1.15 untuk mencegah kekambuhan. Kami juga menyarankan untuk bermigrasi dari StreamingFileSink ke FileSink.
  - Ketika aplikasi dihentikan atau diperbarui, Managed Service for Apache Flink memicu savepoint dan menghentikan aplikasi dalam dua langkah. Jika pos pemeriksaan otomatis terpicu di antara dua langkah, savepoint tidak akan dapat digunakan karena file bagian yang sedang berlangsung akan diganti namanya dan berpotensi dikomit.

FlinkKafkaConsumer masalah dengan berhenti dengan savepoint

Saat menggunakan legacy FlinkKafkaConsumer ada kemungkinan aplikasi Anda mungkin macet dalam UPDATE, STOPING atau SCALING, jika Anda mengaktifkan snapshot sistem. Tidak ada perbaikan yang dipublikasikan yang tersedia untuk masalah ini, oleh karena itu kami sarankan Anda meningkatkan ke yang baru KafkaSourceuntuk mengurangi masalah ini.

Jika Anda menggunakan snapshot `FlinkKafkaConsumer` with diaktifkan, ada kemungkinan saat pekerjaan Flink memproses stop dengan permintaan API savepoint, kesalahan `FlinkKafkaConsumer` dapat gagal dengan kesalahan runtime yang melaporkan file. `ClosedException` Dalam kondisi ini aplikasi Flink menjadi macet, bermanifestasi sebagai Pos Pemeriksaan Gagal.

Flink 1.15 Kebuntuan Wastafel Asinkron

Ada masalah yang diketahui dengan AWS konektor untuk antarmuka implementasi AsyncSink Apache Flink. Ini memengaruhi aplikasi yang menggunakan Flink 1.15 dengan konektor berikut:

- Untuk aplikasi Java:
  - KinesisStreamsSink – `org.apache.flink:flink-connector-kinesis`
  - KinesisStreamsSink – `org.apache.flink:flink-connector-aws-kinesis-streams`
  - KinesisFirehoseSink – `org.apache.flink:flink-connector-aws-kinesis-firehose`
  - DynamoDbSink – `org.apache.flink:flink-connector-dynamodb`

- Aplikasi Flink SQL/Tableapi/Python:
  - kinesis — `org.apache.flink:flink-sql-connector-kinesis`
  - kinesis — `org.apache.flink:flink-sql-connector-aws-kinesis-streams`
  - selang api — `org.apache.flink:flink-sql-connector-aws-kinesis-firehose`
  - dinamodb — `org.apache.flink:flink-sql-connector-dynamodb`

Aplikasi yang terpengaruh akan mengalami gejala berikut:

- Pekerjaan Flink dalam `RUNNING` keadaan, tetapi tidak memproses data;
- Tidak ada pekerjaan restart;
- Pos pemeriksaan sudah habis waktu.

Masalah ini disebabkan oleh bug di AWS SDK sehingga tidak memunculkan kesalahan tertentu ke pemanggil saat menggunakan klien HTTP async. Hal ini mengakibatkan wastafel menunggu tanpa batas waktu untuk “permintaan dalam penerbangan” selesai selama operasi flush pos pemeriksaan.

Masalah ini telah diperbaiki di AWS SDK mulai dari versi 2.20.144.

Berikut ini adalah petunjuk tentang cara memperbarui konektor yang terpengaruh untuk menggunakan versi baru AWS SDK di aplikasi Anda:

Topik

- Perbarui aplikasi Java
- Perbarui aplikasi Python

Perbarui aplikasi Java

Ikuti prosedur di bawah ini untuk memperbarui aplikasi Java:

flink-connector-kinesis

Jika aplikasi menggunakan`flink-connector-kinesis`:

Konektor Kinesis menggunakan shading untuk mengemas beberapa dependensi, termasuk AWS SDK, ke dalam stoples konektor. Untuk memperbarui versi AWS SDK, gunakan prosedur berikut untuk mengganti kelas berbayang ini:

Maven

1. Tambahkan konektor Kinesis dan modul AWS SDK yang diperlukan sebagai dependensi proyek.
2. Konfigurasikan`maven-shade-plugin`:

    a. Tambahkan filter untuk mengecualikan kelas AWS SDK yang diarsir saat menyalin konten jar konektor Kinesis.

    b. Tambahkan aturan relokasi untuk memindahkan kelas AWS SDK yang diperbarui ke paket, yang diharapkan oleh konektor Kinesis.

    pom.xml

```
<project>
```

```
...
<dependencies>
    ...
    <dependency>
        <groupId>org.apache.flink</groupId>
        <artifactId>flink-connector-kinesis</artifactId>
        <version>1.15.4</version>
    </dependency>

    <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>kinesis</artifactId>
        <version>2.20.144</version>
    </dependency>
    <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>netty-nio-client</artifactId>
        <version>2.20.144</version>
    </dependency>
    <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>sts</artifactId>
        <version>2.20.144</version>
    </dependency>
    ...
</dependencies>
...
<build>
    ...
    <plugins>
        ...
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-shade-plugin</artifactId>
            <version>3.1.1</version>
            <executions>
                <execution>
                    <phase>package</phase>
                    <goals>
                        <goal>shade</goal>
                    </goals>
                    <configuration>
                        ...
                        <filters>
```

```
                                ...
                                <filter>
                                    <artifact>org.apache.flink:flink-connector-
kinesis</artifact>
                                    <excludes>
                                        <exclude>org/apache/flink/kinesis/
shaded/software/amazon/awssdk/**</exclude>
                                        <exclude>org/apache/flink/kinesis/
shaded/org/reactivestreams/**</exclude>
                                        <exclude>org/apache/flink/kinesis/
shaded/io/netty/**</exclude>
                                        <exclude>org/apache/flink/kinesis/
shaded/com/typesafe/netty/**</exclude>
                                    </excludes>
                                </filter>
                                ...
                            </filters>
                            <relocations>
                                ...
                                <relocation>
                                    <pattern>software.amazon.awssdk</pattern>

 <shadedPattern>org.apache.flink.kinesis.shaded.software.amazon.awssdk</
shadedPattern>
                                </relocation>
                                <relocation>
                                    <pattern>org.reactivestreams</pattern>

 <shadedPattern>org.apache.flink.kinesis.shaded.org.reactivestreams</
shadedPattern>
                                </relocation>
                                <relocation>
                                    <pattern>io.netty</pattern>

 <shadedPattern>org.apache.flink.kinesis.shaded.io.netty</shadedPattern>
                                </relocation>
                                <relocation>
                                    <pattern>com.typesafe.netty</pattern>

 <shadedPattern>org.apache.flink.kinesis.shaded.com.typesafe.netty</
shadedPattern>
                                </relocation>
                                ...
                            </relocations>
```

```
                            ...
                        </configuration>
                    </execution>
                </executions>
            </plugin>
            ...
        </plugins>
        ...
    </build>
</project>
```

Gradle

1. Tambahkan konektor Kinesis dan modul AWS SDK yang diperlukan sebagai dependensi proyek.
2. Sesuaikan konfigurasi ShadowJar:

   a. Kecualikan kelas AWS SDK yang diarsir saat menyalin konten jar konektor Kinesis.

   b. Pindahkan kelas AWS SDK yang diperbarui ke paket yang diharapkan oleh konektor Kinesis.

   build.gradle

```
...
dependencies {
    ...
    flinkShadowJar("org.apache.flink:flink-connector-kinesis:1.15.4")

    flinkShadowJar("software.amazon.awssdk:kinesis:2.20.144")
    flinkShadowJar("software.amazon.awssdk:sts:2.20.144")
    flinkShadowJar("software.amazon.awssdk:netty-nio-client:2.20.144")
    ...
}
...
shadowJar {
    configurations = [project.configurations.flinkShadowJar]

    exclude("org/apache/flink/kinesis/shaded/software/amazon/awssdk/**/*.class")
    exclude("org/apache/flink/kinesis/shaded/org/reactivestreams/**/*.class")
    exclude("org/apache/flink/kinesis/shaded/io/netty/**/*.class")
```

```
    exclude("org/apache/flink/kinesis/shaded/com/typesafe/netty/**/*.class")

    relocate("software.amazon.awssdk",
 "org.apache.flink.kinesis.shaded.software.amazon.awssdk")
    relocate("org.reactivestreams",
 "org.apache.flink.kinesis.shaded.org.reactivestreams")
    relocate("io.netty", "org.apache.flink.kinesis.shaded.io.netty")
    relocate("com.typesafe.netty",
 "org.apache.flink.kinesis.shaded.com.typesafe.netty")
}
...
```

Konektor lain yang terpengaruh

Jika aplikasi menggunakan konektor lain yang terpengaruh:

Untuk memperbarui versi AWS SDK, versi SDK harus diterapkan dalam konfigurasi build proyek.

Maven

Tambahkan AWS SDK bill of materials (BOM) ke bagian manajemen dependensi `pom.xml` file untuk menerapkan versi SDK untuk proyek.

pom.xml

```
<project>
    ...
    <dependencyManagement>
        <dependencies>
            ...
            <dependency>
                <groupId>software.amazon.awssdk</groupId>
                <artifactId>bom</artifactId>
                <version>2.20.144</version>
                <scope>import</scope>
                <type>pom</type>
            </dependency>
            ...
        </dependencies>
    </dependencyManagement>
    ...
</project>
```

Gradle

Tambahkan ketergantungan platform pada bill of materials AWS SDK (BOM) untuk menerapkan versi SDK untuk proyek. Ini membutuhkan Gradle 5.0 atau yang lebih baru:

build.gradle

```
...
dependencies {
    ...
    flinkShadowJar(platform("software.amazon.awssdk:bom:2.20.144"))
    ...
}
...
```

Perbarui aplikasi Python

Aplikasi Python dapat menggunakan konektor dalam 2 cara berbeda: konektor pengemasan dan dependensi Java lainnya sebagai bagian dari tabung uber tunggal, atau menggunakan jar konektor secara langsung. Untuk memperbaiki aplikasi yang terpengaruh oleh kebuntuan Async Sink:

- Jika aplikasi menggunakan toples uber, ikuti instruksi untukPerbarui aplikasi Java .
- Untuk membangun kembali stoples konektor dari sumber, gunakan langkah-langkah berikut:

Membangun konektor dari sumber:

Prasyarat, mirip dengan persyaratan build Flink:

- Java 11
- Maven 3.2.5

flink-sql-connector-kinesis

1. Unduh kode sumber untuk Flink 1.15.4:

```
wget https://archive.apache.org/dist/flink/flink-1.15.4/flink-1.15.4-src.tgz
```

2. Buka kompres kode sumber:

```
tar -xvf flink-1.15.4-src.tgz
```

3. Arahkan ke direktori konektor kinesis

```
cd flink-1.15.4/flink-connectors/flink-connector-kinesis/
```

4. Kompilasi dan instal jar konektor, tentukan versi AWS SDK yang diperlukan. Untuk mempercepat penggunaan build `-DskipTests` untuk melewati eksekusi pengujian dan `-Dfast` melewati pemeriksaan kode sumber tambahan:

```
mvn clean install -DskipTests -Dfast -Daws.sdkv2.version=2.20.144
```

5. Arahkan ke direktori konektor kinesis

```
cd ../flink-sql-connector-kinesis
```

6. Kompilasi dan pasang jar konektor sql:

```
mvn clean install -DskipTests -Dfast
```

7. Jar yang dihasilkan akan tersedia di:

```
target/flink-sql-connector-kinesis-1.15.4.jar
```

flink-sql-connector-aws-kinesis-aliran

1. Unduh kode sumber untuk Flink 1.15.4:

```
wget https://archive.apache.org/dist/flink/flink-1.15.4/flink-1.15.4-src.tgz
```

2. Buka kompres kode sumber:

```
tar -xvf flink-1.15.4-src.tgz
```

3. Arahkan ke direktori konektor kinesis

```
cd flink-1.15.4/flink-connectors/flink-connector-aws-kinesis-streams/
```

4. Kompilasi dan instal jar konektor, tentukan versi AWS SDK yang diperlukan. Untuk mempercepat penggunaan build `-DskipTests` untuk melewati eksekusi pengujian dan `-Dfast` melewati pemeriksaan kode sumber tambahan:

```
mvn clean install -DskipTests -Dfast -Daws.sdk.version=2.20.144
```

5. Arahkan ke direktori konektor kinesis

```
cd ../flink-sql-connector-aws-kinesis-streams
```

6. Kompilasi dan pasang jar konektor sql:

```
mvn clean install -DskipTests -Dfast
```

7. Jar yang dihasilkan akan tersedia di:

```
target/flink-sql-connector-aws-kinesis-streams-1.15.4.jar
```

flink-sql-connector-aws-kinesis-firehose

1. Unduh kode sumber untuk Flink 1.15.4:

```
wget https://archive.apache.org/dist/flink/flink-1.15.4/flink-1.15.4-src.tgz
```

2. Buka kompres kode sumber:

```
tar -xvf flink-1.15.4-src.tgz
```

3. Arahkan ke direktori konektor

```
cd flink-1.15.4/flink-connectors/flink-connector-aws-kinesis-firehose/
```

4. Kompilasi dan instal jar konektor, tentukan versi AWS SDK yang diperlukan. Untuk mempercepat penggunaan build `-DskipTests` untuk melewati eksekusi pengujian dan `-Dfast` melewati pemeriksaan kode sumber tambahan:

```
mvn clean install -DskipTests -Dfast -Daws.sdk.version=2.20.144
```

5. Arahkan ke direktori konektor sql

```
cd ../flink-sql-connector-aws-kinesis-firehose
```

6. Kompilasi dan pasang jar konektor sql:

```
mvn clean install -DskipTests -Dfast
```

7. Jar yang dihasilkan akan tersedia di:

```
target/flink-sql-connector-aws-kinesis-firehose-1.15.4.jar
```

flink-sql-connector-dynamodb

1. Unduh kode sumber untuk Flink 1.15.4:

```
wget https://archive.apache.org/dist/flink/flink-connector-aws-3.0.0/flink-
connector-aws-3.0.0-src.tgz
```

2. Buka kompres kode sumber:

```
tar -xvf flink-connector-aws-3.0.0-src.tgz
```

3. Arahkan ke direktori konektor

```
cd flink-connector-aws-3.0.0
```

4. Kompilasi dan instal jar konektor, tentukan versi AWS SDK yang diperlukan. Untuk mempercepat penggunaan build `-DskipTests` untuk melewati eksekusi pengujian dan `-Dfast` melewati pemeriksaan kode sumber tambahan:

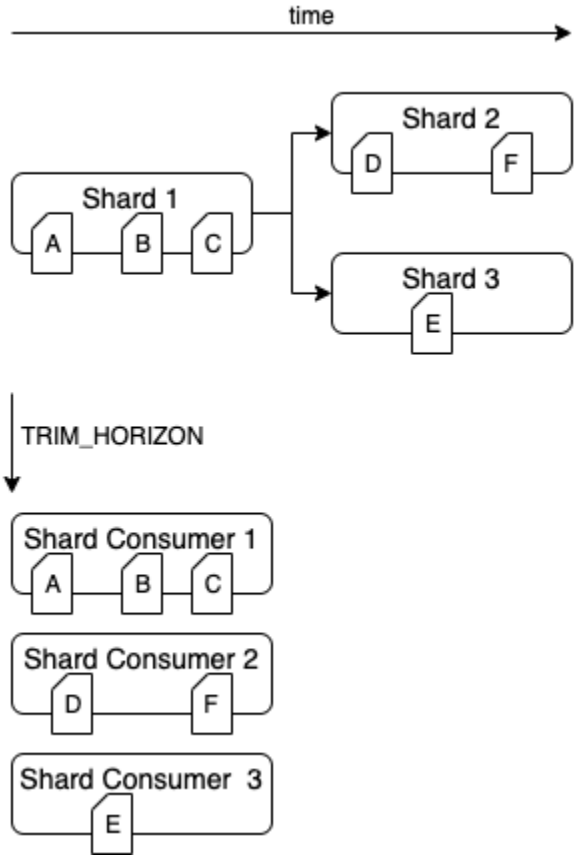
```
mvn clean install -DskipTests -Dfast -Dflink.version=1.15.4 -
Daws.sdk.version=2.20.144
```

5. Jar yang dihasilkan akan tersedia di:

```
flink-sql-connector-dynamodb/target/flink-sql-connector-dynamodb-3.0.0.jar
```

Amazon Kinesis Data Streams Pemrosesan sumber rusak selama re-sharding

FlinkKinesisConsumer Implementasi saat ini tidak memberikan jaminan pemesanan yang kuat antara pecahan Kinesis. Hal ini dapat menyebabkan out-of-order pemrosesan selama re-sharding Kinesis Stream, khususnya untuk aplikasi Flink yang mengalami kelambatan pemrosesan. Dalam beberapa keadaan, misalnya operator windows berdasarkan waktu acara, peristiwa mungkin dibuang karena keterlambatan yang dihasilkan.



Ini adalah masalah yang diketahui di Open Source Flink. Sampai perbaikan konektor tersedia, pastikan aplikasi Flink Anda tidak tertinggal di belakang Kinesis Data Streams selama partisi ulang. Dengan memastikan bahwa penundaan pemrosesan ditoleransi oleh aplikasi Flink Anda, Anda dapat meminimalkan dampak out-of-order pemrosesan dan risiko kehilangan data.

Penyelesaian Masalah Runtime

Bagian ini berisi informasi tentang mendiagnosis dan memperbaiki masalah runtime dengan aplikasi Managed Service for Apache Flink Anda.

Topik

Alat Pemecahan Masalah

Alat utama untuk mendeteksi masalah aplikasi adalahCloudWatchalarm. MenggunakanCloudWatchalarm, Anda dapat mengatur ambang batas untukCloudWatchmetrik yang menunjukkan kondisi error atau bottleneck dalam aplikasi Anda. Untuk informasi tentang rekomendasiCloudWatchalarm, lihatMenggunakan CloudWatch Alarm dengan Amazon Managed Service untuk Apache Flink.

Masalah Aplikasi

Bagian ini berisi solusi untuk kondisi kesalahan yang mungkin Anda temui dengan Layanan Terkelola untuk aplikasi Apache Flink Anda.

Topik

- Aplikasi dalam Status RUNNING, Tetapi Tidak Memproses Data
- Snapshot, Pembaruan Aplikasi, atau Kesalahan Penghentian Aplikasi:InvalidApplicationConfigurationException
- java.nio.file.NoSuchFileException: /usr/lokal/openjdk-8/lib/keamanan/cacerts

Aplikasi Terjebak dalam Status Sementara

Jika aplikasi Anda tetap dalam status sementara (`STARTING`,`UPDATING`,`STOPPING`, atau`AUTOSCALING`), Anda dapat menghentikan aplikasi Anda dengan menggunakanStopApplicationtindakan dengan`Force`parameter diatur `ketrue`. Anda tidak dapat menghentikan paksa aplikasi di status `DELETING`. Atau, jika aplikasi dalam status `UPDATING` atau `AUTOSCALING`, Anda dapat mengembalikannya ke versi berjalan sebelumnya. Ketika Anda mengembalikan aplikasi, data status dari snapshot terakhir yang berhasil akan dimuat. Jika aplikasi tidak memiliki snapshot, Managed Service for Apache Flink menolak permintaan rollback. Untuk informasi selengkapnya tentang memutar kembali aplikasi, lihatRollbackApplicationtindakan.

> **Note**
>
> Menghentikan paksa aplikasi Anda dapat menyebabkan kehilangan data atau duplikasi. Untuk mencegah kehilangan data atau menduplikasi pemrosesan data selama aplikasi dimulai ulang, sebaiknya ambil snapshot yang sering dari aplikasi Anda.

Penyebab aplikasi terhenti mencakup berikut ini:

- Status aplikasi terlalu besar: Memiliki status aplikasi yang terlalu besar atau terlalu persisten dapat menyebabkan aplikasi terhenti selama operasi titik pemeriksaan atau snapshot. Periksa metrik `lastCheckpointDuration` dan `lastCheckpointSize` aplikasi Anda untuk nilai yang terus meningkat atau nilai tinggi yang tidak normal.
- Kode aplikasi terlalu besar: Pastikan file JAR aplikasi Anda lebih kecil dari 512 MB. File JAR yang lebih besar dari 512 MB tidak didukung.
- Pembuatan snapshot aplikasi gagal:Layanan Terkelola untuk Apache Flink mengambil snapshot aplikasi selama`UpdateApplication`atau`StopApplication`permintaan. Layanan selanjutnya menggunakan status snapshot ini dan mengembalikan aplikasi menggunakan konfigurasi aplikasi yang diperbarui untuk memberikan semantik pemrosesan exactly-once. Jika pembuatan snapshot otomatis gagal, lihat Pembuatan Snapshot Gagal di bawah ini.

- Memulihkan dari snapshot gagal: Jika Anda menghapus atau mengubah operator dalam pembaruan aplikasi dan mencoba memulihkan dari snapshot, pemulihan akan gagal secara default jika snapshot berisi data status untuk operator yang hilang. Selain itu, aplikasi akan terhenti di status `STOPPED` atau `UPDATING`. Untuk mengubah perilaku ini dan memungkinkan pemulihan berhasil, ubahAllowNonRestoredStateparameter dari aplikasiFlinkRunConfigurationkepada`true`. Ini akan memungkinkan operasi lanjutkan melewati data status yang tidak dapat dipetakan ke program baru.
- Inisialisasi aplikasi memakan waktu lebih lama:Layanan Terkelola untuk Apache Flink menggunakan batas waktu internal 5 menit (pengaturan lunak) sambil menunggu pekerjaan Flink dimulai. Jika pekerjaan Anda gagal dimulai dalam batas waktu ini, Anda akan melihatCloudWatchlog sebagai berikut:

  ```
  Flink job did not start within a total timeout of 5 minutes for application: %s under
   account: %s
  ```

  Jika Anda menemukan kesalahan di atas, itu berarti operasi Anda ditentukan di bawah pekerjaan Flink`main`Metode memakan waktu lebih dari 5 menit, menyebabkan penciptaan pekerjaan Flink habis pada Layanan Terkelola untuk Apache Flink berakhir. Kami sarankan Anda memeriksa FlinkJobManagerlog serta kode aplikasi Anda untuk melihat apakah penundaan ini di`main`metode yang diharapkan. Jika tidak, Anda perlu mengambil langkah-langkah untuk mengatasi masalah ini sehingga selesai dalam waktu kurang dari 5 menit.

Anda dapat memeriksa status aplikasi Anda menggunakan tindakan `ListApplications` atau `DescribeApplication`.

Pembuatan Snapshot Gagal

Layanan Terkelola untuk layanan Apache Flink tidak dapat mengambil snapshot dalam keadaan berikut:

- Aplikasi melebihi batas snapshot. Batas untuk snapshot adalah 1.000. Untuk informasi selengkapnya, lihat Snapshot.
- Aplikasi tidak memiliki izin untuk mengakses sumber atau sink.
- Kode aplikasi tidak berfungsi dengan benar.
- Aplikasi mengalami masalah konfigurasi lainnya.

Jika Anda mendapatkan pengecualian saat mengambil snapshot selama pembaruan aplikasi atau saat menghentikan aplikasi, atur properti `SnapshotsEnabled` dari `ApplicationSnapshotConfiguration` aplikasi Anda ke `false` dan coba lagi permintaan.

Snapshot dapat gagal jika operator aplikasi Anda tidak disediakan dengan benar. Untuk informasi tentang penyetelan performa operator, lihat Penskalaan operator.

Setelah aplikasi kembali ke status sehat, sebaiknya atur properti `SnapshotsEnabled` aplikasi ke `true`.

Tidak Dapat Mengakses Sumber Daya dalam VPC

Jika aplikasi Anda menggunakan VPC yang berjalan di Amazon VPC, lakukan hal berikut untuk memastikan aplikasi Anda memiliki akses ke sumber dayanya:

- Periksa AndaCloudWatchlog untuk kesalahan berikut. Kesalahan ini menunjukkan aplikasi Anda tidak dapat mengakses sumber daya di VPC Anda:

  ```
  org.apache.kafka.common.errors.TimeoutException: Failed to update metadata after
   60000 ms.
  ```

  Jika Anda melihat kesalahan ini, pastikan tabel rute Anda diatur dengan benar, dan konektor Anda memiliki pengaturan koneksi yang benar.

  Untuk informasi tentang pengaturan dan analisisCloudWatchlog, lihatPencatatan dan Pemantauan.

Data Hilang Saat Menulis ke Bucket Amazon S3

Beberapa kehilangan data mungkin terjadi ketika menulis output ke bucket Amazon S3 menggunakan Apache Flink versi 1.6.2. Sebaiknya gunakan versi Apache Flink terbaru yang didukung ketika menggunakan Amazon S3 untuk output langsung. Untuk menulis ke bucket Amazon S3 menggunakan Apache Flink 1.6.2, sebaiknya gunakan Kinesis Data Firehose. Untuk informasi selengkapnya tentang penggunaan Kinesis Data Firehose dengan Managed Service for Apache Flink, lihatSink Kinesis Data Firehose.

Aplikasi dalam Status RUNNING, Tetapi Tidak Memproses Data

Anda dapat memeriksa status aplikasi Anda menggunakan tindakan `ListApplications` atau `DescribeApplication`. Jika aplikasi Anda memasukiRUNNINGstatus tetapi tidak menulis data ke wastafel Anda, Anda dapat memecahkan masalah dengan menambahkan AmazonCloudWatchlog

stream ke aplikasi Anda. Untuk informasi selengkapnya, lihat Bekerja dengan Opsi CloudWatch Pencatatan Aplikasi. Aliran log berisi pesan yang dapat Anda gunakan untuk memecahkan masalah aplikasi.

Snapshot, Pembaruan Aplikasi, atau Kesalahan Penghentian Aplikasi:InvalidApplicationConfigurationException

Kesalahan yang mirip dengan berikut ini mungkin terjadi selama operasi snapshot, atau selama operasi yang membuat snapshot, seperti memperbarui atau menghentikan aplikasi:

```
An error occurred (InvalidApplicationConfigurationException) when calling the
 UpdateApplication operation:

Failed to take snapshot for the application xxxx at this moment. The application is
 currently experiencing downtime.
Please check the application's CloudWatch metrics or CloudWatch logs for any possible
 errors and retry the request.
You can also retry the request after disabling the snapshots in the Managed Service for
 Apache Flink console or by updating
the ApplicationSnapshotConfiguration through the AWS SDK
```

Kesalahan ini terjadi ketika aplikasi tidak dapat membuat snapshot.

Jika Anda mengalami kesalahan ini selama operasi snapshot atau operasi yang membuat snapshot, lakukan hal berikut:

- Nonaktifkan snapshot untuk aplikasi Anda. Anda dapat melakukan ini baik di Managed Service for Apache Flink console, atau dengan menggunakan`SnapshotsEnabledUpdate`parameter dariUpdateApplicationtindakan.
- Selidiki alasan snapshot tidak dapat dibuat. Untuk informasi selengkapnya, lihat Aplikasi Terjebak dalam Status Sementara.
- Aktifkan kembali snapshot ketika aplikasi kembali ke status sehat.

java.nio.file.NoSuchFileException: /usr/lokal/openjdk-8/lib/keamanan/cacerts

Lokasi truststore SSL diperbarui di deployment sebelumnya. Sebagai gantinya, gunakan nilai berikut untuk parameter `ssl.truststore.location`:

```
/usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts
```

Aplikasi Dimulai Ulang

Jika aplikasi Anda tidak sehat, tugas Apache Flink terus gagal dan dimulai ulang. Bagian ini menjelaskan gejala dan langkah pemecahan masalah untuk kondisi ini.

Gejala

Kondisi ini dapat memiliki gejala berikut:

- Metrik `FullRestarts` tidak nol. Metrik ini menunjukkan berapa kali tugas aplikasi dimulai ulang sejak Anda memulai aplikasi.
- Metrik `Downtime` tidak nol. Metrik ini menunjukkan jumlah milidetik ketika aplikasi berada di status `FAILING` atau `RESTARTING`.
- Log aplikasi berisi perubahan status ke `RESTARTING` atau `FAILED`. Anda dapat menanyakan log aplikasi Anda untuk perubahan status ini menggunakan yang berikutCloudWatchKueri Wawasan Log:Analisis Kesalahan: Kegagalan Terkait Tugas Aplikasi.

Penyebab dan solusi

Kondisi berikut dapat menyebabkan aplikasi Anda menjadi tidak stabil dan dimulai ulang berulang kali:

- Operator Melempar Pengecualian: Jika pengecualian apa pun dalam operator di aplikasi Anda tidak ditangani, aplikasi gagal (dengan menafsirkan kegagalan tidak dapat ditangani oleh operator). Aplikasi dimulai ulang dari titik pemeriksaan terbaru untuk mempertahankan semantik pemrosesan "exactly-once". Akibatnya, `Downtime` tidak nol selama periode mulai ulang ini. Agar hal ini tidak terjadi, sebaiknya tangani pengecualian yang dapat dicoba lagi dalam kode aplikasi.

  Anda dapat menyelidiki penyebab kondisi ini dengan mengkueri log aplikasi Anda untuk perubahan dari status aplikasi Anda dari `RUNNING` ke `FAILED`. Untuk informasi selengkapnya, lihat the section called “Analisis Kesalahan: Kegagalan Terkait Tugas Aplikasi”.
- Kinesis Data Streams tidak disediakan dengan benar: Jika sumber atau sink untuk aplikasi Anda adalah Kinesis data stream, periksa metrik untuk aliran untuk kesalahan `ReadProvisionedThroughputExceeded` atau `WriteProvisionedThroughputExceeded`.

  Jika Anda melihat kesalahan ini, Anda dapat meningkatkan throughput yang tersedia untuk aliran Kinesis dengan meningkatkan jumlah serpihan aliran. Untuk informasi selengkapnya, lihat Bagaimana cara mengubah jumlah serpihan terbuka di Kinesis Data Streams?.

- Sumber atau sink lainnya tidak diberikan atau tersedia dengan benar: Pastikan aplikasi Anda menyediakan sumber dan sink dengan benar. Periksa apakah sumber atau sink apa pun yang digunakan dalam aplikasi (seperti layanan AWS lainnya, atau sumber atau pun tujuan eksternal) yang disediakan dengan baik, tidak mengalami throttling baca atau tulis, atau pun tidak tersedia secara berkala.

  Jika Anda mengalami masalah terkait throughput pada layanan dependen Anda, baik meningkatkan sumber daya yang tersedia untuk layanan tersebut, maupun menyelidiki penyebab kesalahan atau ketidaktersediaan apa pun.
- Operator tidak disediakan dengan benar: Jika beban kerja pada utas untuk salah satu operator dalam aplikasi Anda tidak didistribusikan dengan benar, operator dapat kelebihan beban dan aplikasi dapat crash. Untuk informasi tentang penyetelan paralelisme operator, lihat Kelola penskalaan operator dengan benar.
- Aplikasi gagal denganDaemonException:Kesalahan ini muncul di log aplikasi Anda jika Anda menggunakan versi Apache Flink sebelum 1.11. Anda mungkin perlu meng-upgrade ke versi Apache Flink yang lebih baru sehingga versi KPL 0,14 atau yang lebih baru digunakan.
- Aplikasi gagal denganTimeoutException,FlinkException, atauRemoteTransportException:Kesalahan ini mungkin muncul di log aplikasi Anda jika pengelola tugas Anda mogok. Jika aplikasi Anda kelebihan beban, manajer tugas Anda dapat mengalami tekanan sumber daya CPU atau memori, menyebabkannya gagal.

  Kesalahan ini mungkin terlihat seperti berikut:

  - `java.util.concurrent.TimeoutException: The heartbeat of JobManager with id xxx timed out`
  - `org.apache.flink.util.FlinkException: The assigned slot xxx was removed`
  - `org.apache.flink.runtime.io.network.netty.exception.RemoteTransportException: Connection unexpectedly closed by remote task manager`

  Untuk memecahkan masalah kondisi ini, periksa hal berikut:

  - PeriksaCloudWatchmetrik untuk lonjakan yang tidak biasa dalam penggunaan CPU atau memori.
  - Periksa aplikasi Anda untuk masalah throughput. Untuk informasi selengkapnya, lihat Memecahkan masalah performa.
  - Periksa log aplikasi Anda untuk pengecualian yang tidak tertangani yang ditimbulkan oleh kode aplikasi Anda.

- Aplikasi gagal denganJaxbAnnotationModuleKesalahan Tidak Ditemukan:Kesalahan ini terjadi jika aplikasi Anda menggunakan Apache Beam, tetapi tidak memiliki dependensi atau versi dependensi yang benar. Managed Service untuk aplikasi Apache Flink yang menggunakan Apache Beam harus menggunakan versi dependensi berikut:

```
<jackson.version>2.10.2</jackson.version>
...
<dependency>
    <groupId>com.fasterxml.jackson.module</groupId>
    <artifactId>jackson-module-jaxb-annotations</artifactId>
    <version>2.10.2</version>
</dependency>
```

  Jika Anda tidak menyediakan versi `jackson-module-jaxb-annotations` yang benar sebagai dependensi eksplisit, aplikasi Anda memuatnya dari dependensi lingkungan, dan karena versi tidak cocok, aplikasi mengalami crash saat runtime.

  Untuk informasi selengkapnya tentang penggunaan Apache Beam dengan Managed Service for Apache Flink, lihatMenggunakan CloudFormation dengan Managed Service untuk Apache Flink.

- Aplikasi gagal dengan java.io.IOException: Jumlah buffer jaringan tidak mencukupi

  Ini terjadi ketika aplikasi tidak memiliki cukup memori yang dialokasikan untuk buffer jaringan. Buffer jaringan memfasilitasi komunikasi antar subtugas. Mereka digunakan untuk menyimpan catatan sebelum transmisi melalui jaringan, dan untuk menyimpan data yang masuk sebelum membedahnya menjadi catatan dan menyerahkannya ke subtugas. Jumlah buffer jaringan diperlukan skala langsung dengan paralelisme dan kompleksitas grafik pekerjaan Anda. Ada sejumlah pendekatan untuk mengurangi masalah ini:

  - Anda dapat mengonfigurasi yang lebih rendah`parallelismPerKpu`sehingga ada lebih banyak memori yang dialokasikan per subtask dan buffer jaringan. Perhatikan bahwa menurunkan`parallelismPerKpu`akan meningkatkan KPU dan karenanya biaya. Untuk menghindari hal ini, Anda dapat menjaga jumlah KPU yang sama dengan menurunkan paralelisme dengan faktor yang sama.
  - Anda dapat menyederhanakan grafik pekerjaan Anda dengan mengurangi jumlah operator atau merantainya sehingga lebih sedikit buffer yang dibutuhkan.
  - Jika tidak, Anda dapat menghubungi https://aws.amazon.com/premiumsupport/ untuk konfigurasi buffer jaringan khusus.

Throughput Terlalu Lambat

Jika aplikasi Anda tidak memproses data streaming yang masuk dengan cukup cepat, aplikasi akan berperforma buruk dan menjadi tidak stabil. Bagian ini menjelaskan gejala dan langkah pemecahan masalah untuk kondisi ini.

Gejala

Kondisi ini dapat memiliki gejala berikut:

- Jika sumber data untuk aplikasi Anda adalah aliran Kinesis, metrik `millisbehindLatest` aliran terus meningkat.
- Jika sumber data untuk aplikasi Anda adalah klaster Amazon MSK, metrik lag konsumen klaster terus meningkat. Untuk informasi selengkapnya, lihat Pemantauan Lag Konsumen di Panduan Developer Amazon MSK.
- Jika sumber data untuk aplikasi Anda adalah layanan atau sumber yang berbeda, periksa metrik atau data lag konsumen yang tersedia.

Penyebab dan solusi

Ada banyak penyebab untuk throughput aplikasi yang lambat. Jika aplikasi Anda tidak mengikuti input, periksa hal berikut:

- Jika lag throughput melonjak, lalu menurun, periksa apakah aplikasi dimulai ulang. Aplikasi Anda akan berhenti memproses input saat dimulai ulang, menyebabkan lonjakan lag. Untuk informasi selengkapnya tentang kegagalan aplikasi, lihat Aplikasi Dimulai Ulang.
- Jika lag throughput konsisten, periksa untuk melihat apakah performa aplikasi Anda dioptimalkan. Untuk informasi tentang mengoptimalkan performa aplikasi, lihat Memecahkan masalah performa.
- Jika lag throughput tidak melonjak, tetapi terus meningkat, dan performa aplikasi Anda dioptimalkan, Anda harus meningkatkan sumber daya aplikasi Anda. Untuk informasi tentang peningkatan sumber daya aplikasi, lihat Penskalaan.
- Jika aplikasi Anda membaca dari cluster Kafka di Wilayah yang berbeda dan`FlinkKafkaConsumer`atau`KafkaSource`sebagian besar menganggur (tinggi`idleTimeMsPerSecond`atau rendah`CPUUtilization`) meskipun kelambatan konsumen tinggi, Anda dapat meningkatkan nilai`receive.buffer.byte`, seperti 2097152. Untuk informasi selengkapnya, lihat bagian lingkungan latensi tinggi diKonfigurasi MSK kustom.

Untuk langkah-langkah pemecahan masalah untuk throughput lambat atau lag konsumen yang meningkat di sumber aplikasi, lihat Memecahkan masalah performa.

Pertumbuhan Negara Tak Terbatas

Jika aplikasi Anda tidak membuang informasi status yang tidak berlaku dengan benar, informasi akan terus diakumulasi dan menyebabkan masalah performa atau stabilitas aplikasi. Bagian ini menjelaskan gejala dan langkah pemecahan masalah untuk kondisi ini.

Gejala

Kondisi ini dapat memiliki gejala berikut:

- Metrik `lastCheckpointDuration` meningkat atau melonjak secara bertahap.
- Metrik `lastCheckpointSize` meningkat atau melonjak secara bertahap.

Penyebab dan solusi

Kondisi berikut dapat menyebabkan aplikasi Anda mengakumulasi data status:

- Aplikasi Anda menyimpan data status lebih lama dari yang dibutuhkan.
- Aplikasi Anda menggunakan kueri jendela dengan durasi yang terlalu lama.
- Anda tidak menetapkan TTL untuk data status Anda. Untuk informasi selengkapnya, lihat Waktu Untuk Tayang (TTL) Status di Dokumentasi Apache Flink.
- Anda menjalankan aplikasi yang bergantung pada Apache Beam versi 2.25.0 atau yang lebih baru. Anda dapat memilih keluar dari versi baru transformasi baca dengan memperluas AndaBeamApplicationPropertiesdengan eksperimen dan nilai kunci `use_deprecated_read`. Untuk informasi lebih lanjut, lihatDokumentasi Apache Beam.

Terkadang aplikasi menghadapi pertumbuhan ukuran negara yang terus berkembang, yang tidak berkelanjutan dalam jangka panjang (bagaimanapun juga aplikasi Flink berjalan tanpa batas waktu). Terkadang, ini dapat ditelusuri kembali ke aplikasi yang menyimpan data dalam keadaan dan tidak menua informasi lama dengan benar. Tapi terkadang hanya ada harapan yang tidak masuk akal tentang apa yang bisa diberikan Flink. Aplikasi dapat menggunakan agregasi selama jendela waktu besar yang mencakup hari atau bahkan berminggu-minggu. KecualiAggregateFunctionsdigunakan, yang memungkinkan agregasi tambahan, Flink perlu menjaga peristiwa seluruh jendela dalam keadaan.

Selain itu, ketika menggunakan fungsi proses untuk mengimplementasikan operator khusus, aplikasi perlu menghapus data dari status yang tidak lagi diperlukan untuk logika bisnis. Dalam hal ini,negaratime-to-livedapat digunakan untuk secara otomatis menua data berdasarkan waktu pemrosesan. Layanan Terkelola untuk Apache Flink menggunakan pos pemeriksaan tambahan dan dengan demikian status ttl didasarkan padaPemadatan RocksDB. Anda hanya dapat mengamati pengurangan aktual dalam ukuran status (ditunjukkan oleh ukuran pos pemeriksaan) setelah operasi pemadatan terjadi. Khususnya untuk ukuran pos pemeriksaan di bawah 200 MB, kecil kemungkinan Anda mengamati pengurangan ukuran pos pemeriksaan sebagai akibat dari keadaan kedaluwarsa. Namun, savepoints didasarkan pada salinan bersih dari status yang tidak berisi data lama, sehingga Anda dapat memicu snapshot di Managed Service for Apache Flink untuk memaksa penghapusan status usang.

Untuk tujuan debugging, masuk akal untuk menonaktifkan pos pemeriksaan tambahan untuk memverifikasi lebih cepat bahwa ukuran pos pemeriksaan benar-benar berkurang atau stabil (dan menghindari efek pemadatan di RocksBS). Ini membutuhkan tiket ke tim layanan.

Operator terikat I/O

Yang terbaik adalah menghindari dependensi ke sistem eksternal pada jalur data. Seringkali jauh lebih berkinerja untuk menyimpan kumpulan data referensi dalam keadaan daripada menanyakan sistem eksternal untuk memperkaya peristiwa individu. Namun, terkadang ada dependensi yang tidak dapat dengan mudah dipindahkan ke status, misalnya, jika Anda ingin memperkaya peristiwa dengan model pembelajaran mesin yang di-host di Amazon Sagemaker.

Operator yang berinteraksi dengan sistem eksternal melalui jaringan dapat menjadi hambatan dan menyebabkan tekanan balik. Hal ini sangat dianjurkan untuk menggunakanAsynciountuk mengimplementasikan fungsionalitas, untuk mengurangi waktu tunggu untuk panggilan individu dan menghindari seluruh aplikasi melambat.

Selain itu, untuk aplikasi dengan operator terikat I/O juga masuk akal untuk meningkatkanParallelismPerKPUpengaturan Layanan Terkelola untuk aplikasi Apache Flink. Konfigurasi ini menjelaskan jumlah subtugas paralel yang dapat dilakukan aplikasi per Kinesis Processing Unit (KPU). Dengan meningkatkan nilai dari default 1 menjadi, katakanlah, 4, aplikasi memanfaatkan sumber daya yang sama (dan memiliki biaya yang sama) tetapi dapat menskalakan hingga 4 kali paralelisme. Ini berfungsi dengan baik untuk aplikasi terikat I/O, tetapi menyebabkan overhead tambahan untuk aplikasi yang tidak terikat I/O.

Pelambatan hulu atau sumber dari aliran data Kinesis

Gejala: Aplikasi sedang bertemu`LimitExceededExceptions`dari sumber hulu Kinesis aliran data mereka.

Penyebab Potensi: Pengaturan default untuk konektor Kinesis pustaka Apache Flink diatur untuk membaca dari sumber aliran data Kinesis dengan pengaturan default yang sangat agresif untuk jumlah maksimum catatan yang diambil per`GetRecords`panggilan. Apache Flink dikonfigurasi secara default untuk mengambil 10.000 catatan perGetRecordspanggilan (panggilan ini dibuat secara default setiap 200 ms), meskipun batas per pecahan hanya 1.000 catatan.

Perilaku default ini dapat menyebabkan pelambatan saat mencoba mengkonsumsi dari aliran data Kinesis, yang akan memengaruhi kinerja dan stabilitas aplikasi.

Hal ini dapat dikonfirmasi dengan memeriksaCloudWatch `ReadProvisionedThroughputExceeded`metrik dan melihat periode yang berkepanjangan atau berkelanjutan di mana metrik ini lebih besar dari nol.

Pelanggan juga akan dapat melihat ini diCloudWatchlog untuk Layanan Terkelola mereka untuk aplikasi Apache Flink dengan melihat lanjutan`LimitExceededException`kesalahan.

Resolusi: Pelanggan dapat melakukan salah satu dari dua hal untuk menyelesaikan skenario ini:

- Turunkan batas default untuk jumlah catatan yang diambil perGetRecordspanggilan
- Pelanggan dapat mengaktifkan Adaptive Reads di Managed Service untuk aplikasi Apache Flink mereka. Untuk informasi selengkapnya tentang fitur Bacaan Adaptif, lihatSHARD_USE_ADAPTIVE_READS

Titik pemeriksaan

Pos pemeriksaan adalah mekanisme Flink untuk memastikan bahwa status aplikasi toleran terhadap kesalahan. Mekanisme ini memungkinkan Flink untuk memulihkan status operator jika pekerjaan gagal dan memberikan aplikasi semantik yang sama dengan eksekusi bebas kegagalan. Dengan Managed Service for Apache Flink, status aplikasi disimpan di RocksDB, penyimpanan kunci/nilai tertanam yang menjaga status kerjanya pada disk. Ketika pos pemeriksaan diambil, status juga diunggah ke Amazon S3 sehingga meskipun disk hilang maka pos pemeriksaan dapat digunakan untuk mengembalikan status aplikasi.

Untuk informasi lebih lanjut, lihatBagaimana cara kerja snapshotting negara?.

Tahapan pemeriksaan

Untuk subtugas operator checkpointing di Flink ada 5 tahap utama:

- Menunggu [Mulai Penundaan] - Flink menggunakan penghalang pos pemeriksaan yang dimasukkan ke dalam aliran sehingga waktu dalam tahap ini adalah waktu operator menunggu penghalang pos pemeriksaan untuk mencapainya.
- Penjajaran [Durasi Penyelarasan] — Pada tahap ini subtugas telah mencapai satu penghalang tetapi menunggu hambatan dari aliran input lainnya.
- Sinkronkan pos pemeriksaan [Durasi Sinkronisasi] — Tahap ini adalah ketika subtugas benar-benar memotret status operator dan memblokir semua aktivitas lain pada subtugas.
- Pos pemeriksaan asinkron [Durasi Asinkron] - Mayoritas tahap ini adalah subtugas yang mengunggah status ke Amazon S3. Selama tahap ini, subtugas tidak lagi diblokir dan dapat memproses catatan.
- Mengakui — Ini biasanya merupakan tahap pendek dan hanyalah subtugas yang mengirimkan pengakuan keJobManagerdan juga melakukan pesan komit apa pun (misalnya dengan sink Kafka).

Masing-masing tahapan ini (selain dari Mengakui) memetakan ke metrik durasi untuk pos pemeriksaan yang tersedia dari WebUI Flink, yang dapat membantu mengisolasi penyebab pos pemeriksaan yang panjang.

Untuk melihat definisi yang tepat dari setiap metrik yang tersedia di pos pemeriksaan, bukaTab Sejarah.

Menyelidiki

Saat menyelidiki durasi pos pemeriksaan yang panjang, hal terpenting yang harus ditentukan adalah kemacetan untuk pos pemeriksaan, yaitu operator dan subtugas apa yang paling lama menuju pos pemeriksaan dan tahap mana dari subtugas itu yang membutuhkan waktu yang lama. Ini dapat ditentukan menggunakan Flink WebUI di bawah tugas pos pemeriksaan pekerjaan. Antarmuka Web Flink menyediakan data dan informasi yang membantu menyelidiki masalah pos pemeriksaan. Untuk rincian lengkap, lihatPemantauan Checkpointing.

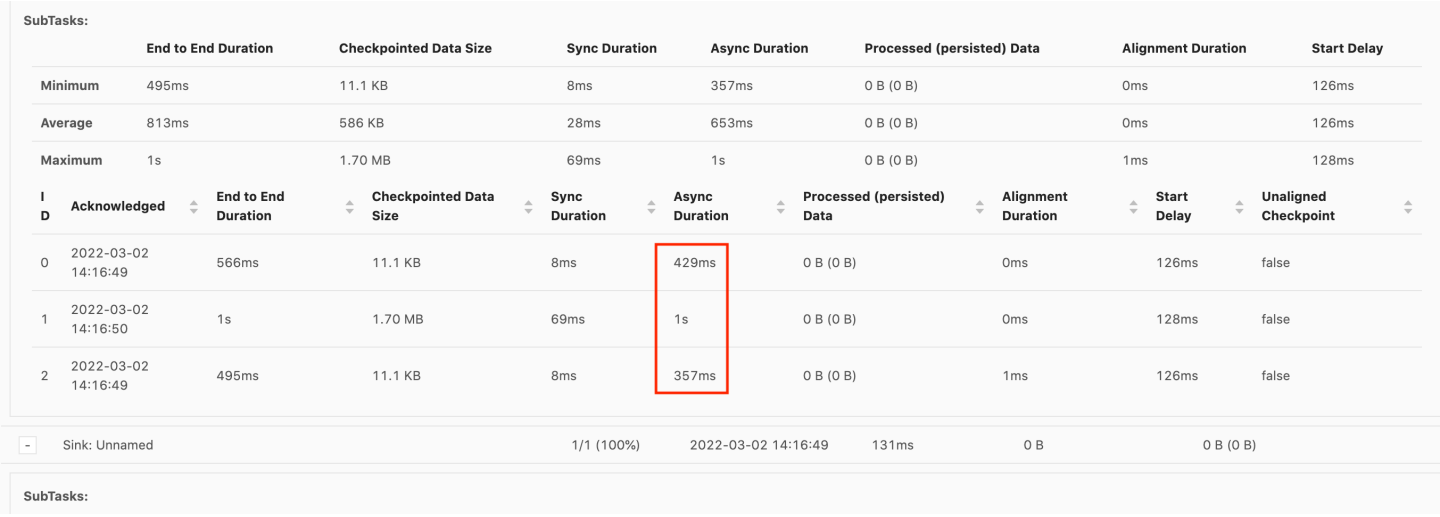
Hal pertama yang harus dilihat adalahDurasi Akhir ke Akhirdari setiap operator dalam grafik Pekerjaan untuk menentukan operator mana yang membutuhkan waktu lama untuk pos pemeriksaan dan menjamin penyelidikan lebih lanjut. Per dokumentasi Flink, definisi durasinya adalah:

Durasi dari stempel waktu pemicu hingga pengakuan terbaru (atau n/a jika belum ada pengakuan yang diterima). Durasi ujung ke akhir untuk pos pemeriksaan lengkap ditentukan oleh subtugas terakhir yang mengakui pos pemeriksaan. Waktu ini biasanya lebih besar dari subtugas tunggal yang perlu benar-benar memeriksa status.

Durasi lain untuk pos pemeriksaan juga memberikan informasi yang lebih halus tentang di mana waktu dihabiskan.

JikaDurasi Sinkronisasitinggi maka ini menunjukkan sesuatu sedang terjadi selama snapshotting. Selama tahap ini`snapshotState()`dipanggil untuk kelas yang mengimplementasikan antarmuka SnapshotState; ini bisa menjadi kode pengguna sehingga thread-dumps dapat berguna untuk menyelidiki ini.

PanjangDurasi Asinkronakan menyarankan bahwa banyak waktu dihabiskan untuk mengunggah negara bagian ke Amazon S3. Ini dapat terjadi jika statusnya besar atau jika ada banyak file status yang sedang diunggah. Jika ini masalahnya, perlu diselidiki bagaimana status digunakan oleh aplikasi dan memastikan bahwa struktur data asli Flink digunakan jika memungkinkan (Menggunakan Status Keyed). Layanan Terkelola untuk Apache Flink mengonfigurasi Flink sedemikian rupa untuk meminimalkan jumlah panggilan Amazon S3 untuk memastikan ini tidak terlalu lama. Berikut ini adalah contoh statistik checkpointing operator. Ini menunjukkan bahwaDurasi Asinkronrelatif panjang dibandingkan dengan statistik checkpointing operator sebelumnya.



SubTasks:

	End to End Duration	Checkpointed Data Size	Sync Duration	Async Duration	Processed (persisted) Data	Alignment Duration	Start Delay
Minimum	495ms	11.1 KB	8ms	357ms	0 B (0 B)	0ms	126ms
Average	813ms	586 KB	28ms	653ms	0 B (0 B)	0ms	126ms
Maximum	1s	1.70 MB	69ms	1s	0 B (0 B)	1ms	128ms

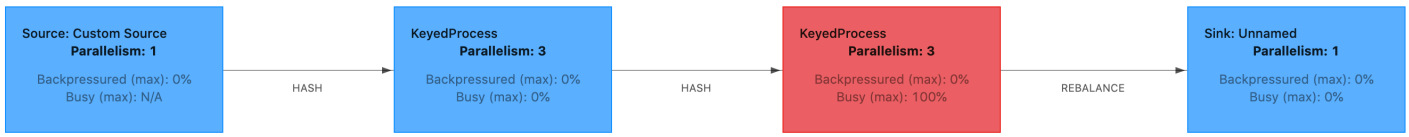
ID	Acknowledged	End to End Duration	Checkpointed Data Size	Sync Duration	Async Duration	Processed (persisted) Data	Alignment Duration	Start Delay	Unaligned Checkpoint
0	2022-03-02 14:16:49	566ms	11.1 KB	8ms	429ms	0 B (0 B)	0ms	126ms	false
1	2022-03-02 14:16:50	1s	1.70 MB	69ms	1s	0 B (0 B)	0ms	128ms	false
2	2022-03-02 14:16:49	495ms	11.1 KB	8ms	357ms	0 B (0 B)	1ms	126ms	false

-	Sink: Unnamed	1/1 (100%)	2022-03-02 14:16:49	131ms	0 B	0 B (0 B)

SubTasks:

TheMulai PenundaanMenjadi tinggi akan menunjukkan bahwa sebagian besar waktu dihabiskan untuk menunggu penghalang pos pemeriksaan untuk mencapai operator. Ini menunjukkan bahwa aplikasi membutuhkan waktu untuk memproses catatan, yang berarti penghalang mengalir melalui grafik pekerjaan secara perlahan. Ini biasanya terjadi jika Job mengalami

backpressure atau jika operator selalu sibuk. Berikut ini adalah contoh dariJobGraphdimana yang keduaKeyedProcessOperator sedang sibuk.



Anda dapat menyelidiki apa yang memakan waktu lama dengan menggunakan Flink Flame Graphs atauTaskManagertempat pembuangan benang. Setelah leher botol diidentifikasi, dapat diselidiki lebih lanjut menggunakan Flame-graphs atau thread-dumps.

Pembuangan benang

Thread dump adalah alat debugging lain yang berada pada tingkat yang sedikit lebih rendah dari grafik api. Thread dump menampilkan status eksekusi semua thread pada satu titik waktu. Flink mengambil dump thread JVM, yang merupakan status eksekusi dari semua thread dalam proses Flink. Keadaan utas disajikan oleh jejak tumpukan utas serta beberapa informasi tambahan. Grafik api sebenarnya dibuat menggunakan beberapa jejak tumpukan yang diambil secara berurutan. Grafik adalah visualisasi yang dibuat dari jejak ini yang membuatnya mudah untuk mengidentifikasi jalur kode umum.

```
"KeyedProcess (1/3)#0" prio=5 Id=1423 RUNNABLE
    at app//scala.collection.immutable.Range.foreach$mVc$sp(Range.scala:154)
    at $line33.$read$$iw$$iw$ExpensiveFunction.processElement(<console>:19)
    at $line33.$read$$iw$$iw$ExpensiveFunction.processElement(<console>:14)
    at app//
org.apache.flink.streaming.api.operators.KeyedProcessOperator.processElement(KeyedProcessOperat
    at app//org.apache.flink.streaming.runtime.tasks.OneInputStreamTask
$StreamTaskNetworkOutput.emitRecord(OneInputStreamTask.java:205)
    at app//
org.apache.flink.streaming.runtime.io.AbstractStreamTaskNetworkInput.processElement(AbstractStr
```

```
    at app//
org.apache.flink.streaming.runtime.io.AbstractStreamTaskNetworkInput.emitNext(AbstractStreamTas
    at app//
org.apache.flink.streaming.runtime.io.StreamOneInputProcessor.processInput(StreamOneInputProces
    ...
```

Di atas adalah cuplikan dump utas yang diambil dari UI Flink untuk satu utas. Baris pertama berisi beberapa informasi umum tentang utas ini termasuk:

- Nama threadKeyedProcess(1/3) #0
- Prioritas utasprio=5
- Id utas unikId=1423
- Status utasDAPAT DIJALANKAN

Nama utas biasanya memberikan informasi tentang tujuan umum utas. Utas operator dapat diidentifikasi dengan namanya karena utas operator memiliki nama yang sama dengan operator, serta indikasi subtugas mana yang terkait dengannya, misalnya,KeyedProcess(1/3) #0Thread berasal dariKeyedProcessoperator dan berasal dari subtugas 1 (dari 3).

Thread dapat berada di salah satu dari beberapa negara bagian:

- BARU - Thread telah dibuat tetapi belum diproses
- RUNNABLE — Thread adalah eksekusi pada CPU
- DIBLOKIR - Utas sedang menunggu utas lain untuk melepaskan kuncinya
- WAITING — Thread sedang menunggu dengan menggunakan`wait()`,`join()`, atau`park()`metode
- TIMED_WAITING — Thread sedang menunggu dengan menggunakan metode sleep, wait, join atau park, tetapi dengan waktu tunggu maksimum.

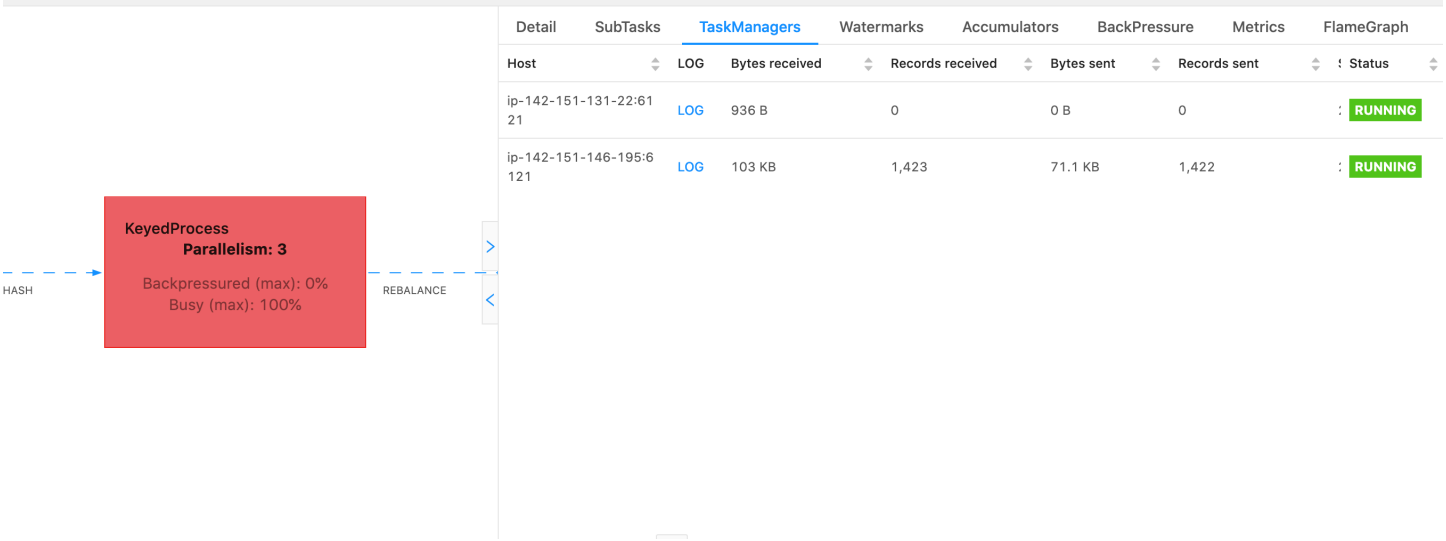
> **Note**
>
> Di Flink 1.13, kedalaman maksimum satu stacktrace di thread dump dibatasi hingga 8.

> **Note**
>
> Thread dump harus menjadi pilihan terakhir untuk men-debug masalah kinerja dalam aplikasi Flink karena dapat menantang untuk dibaca, memerlukan beberapa sampel untuk diambil dan dianalisis secara manual. Jika memungkinkan, lebih baik menggunakan grafik nyala api.

Pembuangan utas di Flink

Di Flink, dump utas dapat diambil dengan memilihManajer Tugasopsi di bilah navigasi kiri UI Flink, memilih pengelola tugas tertentu, dan kemudian menavigasi kePembuangan Benangtab. Thread dump dapat diunduh, disalin ke editor teks favorit Anda (atau thread dump analyzer), atau dianalisis langsung di dalam tampilan teks di UI Web Flink (namun, opsi terakhir ini bisa sedikit kikuk.

Untuk menentukan Task Manager mana yang akan mengambil thread dump dariTaskManagerstab dapat digunakan ketika operator tertentu dipilih. Ini menunjukkan bahwa operator berjalan pada subtugas yang berbeda dari operator dan dapat berjalan pada Manajer Tugas yang berbeda.



Dump akan terdiri dari beberapa jejak tumpukan. Namun ketika menyelidiki dump, yang terkait dengan operator adalah yang paling penting. Ini dapat dengan mudah ditemukan karena utas operator memiliki nama yang sama dengan operator, serta indikasi subtugas mana yang terkait dengannya. Misalnya jejak tumpukan berikut berasal dariKeyedProcessoperator dan merupakan subtugas pertama.

```
"KeyedProcess (1/3)#0" prio=5 Id=595 RUNNABLE
    at app//scala.collection.immutable.Range.foreach$mVc$sp(Range.scala:155)
```

```
    at $line360.$read$$iw$$iw$ExpensiveFunction.processElement(<console>:19)
    at $line360.$read$$iw$$iw$ExpensiveFunction.processElement(<console>:14)
    at app//
org.apache.flink.streaming.api.operators.KeyedProcessOperator.processElement(KeyedProcessOperat
    at app//org.apache.flink.streaming.runtime.tasks.OneInputStreamTask
$StreamTaskNetworkOutput.emitRecord(OneInputStreamTask.java:205)
    at app//
org.apache.flink.streaming.runtime.io.AbstractStreamTaskNetworkInput.processElement(AbstractStr
    at app//
org.apache.flink.streaming.runtime.io.AbstractStreamTaskNetworkInput.emitNext(AbstractStreamTas
    at app//
org.apache.flink.streaming.runtime.io.StreamOneInputProcessor.processInput(StreamOneInputProces
    ...
```

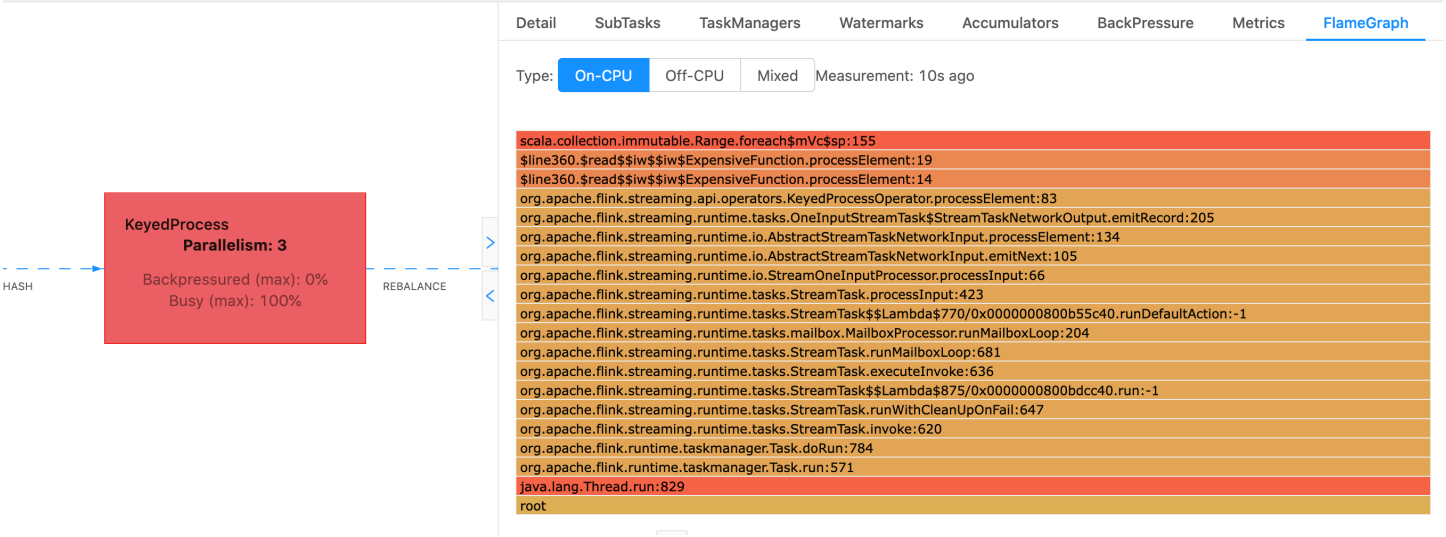
Ini bisa menjadi membingungkan jika ada beberapa operator dengan nama yang sama tetapi kita dapat memberi nama operator untuk menyiasatinya. Misalnya:

```
....
.process(new ExpensiveFunction).name("Expensive function")
```

Grafik api

Grafik api adalah alat debugging yang berguna yang memvisualisasikan jejak tumpukan kode yang ditargetkan, yang memungkinkan jalur kode yang paling sering diidentifikasi. Mereka dibuat dengan pengambilan sampel jejak tumpukan beberapa kali. Sumbu x dari grafik nyala menunjukkan profil tumpukan yang berbeda, sedangkan sumbu y menunjukkan kedalaman tumpukan, dan panggilan dalam jejak tumpukan. Sebuah persegi panjang tunggal dalam grafik nyala mewakili pada bingkai tumpukan, dan lebar bingkai menunjukkan seberapa sering muncul di tumpukan. Untuk detail selengkapnya tentang grafik api dan cara menggunakannya, lihatGrafik Api.

Di Flink, grafik api untuk operator dapat diakses melalui UI Web dengan memilih operator dan kemudian memilihFlameGraphtab. Setelah sampel yang cukup dikumpulkan, flamegraph akan ditampilkan. Berikut ini adalahFlameGraphuntukProcessFunctionItu membutuhkan banyak waktu untuk pos pemeriksaan.



Ini adalah grafik api yang sangat sederhana dan menunjukkan bahwa semua waktu CPU dihabiskan dalam tampilan foreach dalam`processElement`dariExpensiveFunctionoperator. Anda juga mendapatkan nomor baris untuk membantu menentukan di mana eksekusi kode berlangsung.

Waktu titik checkpointing

Jika aplikasi Anda tidak dioptimalkan atau disediakan dengan benar, titik pemeriksaan bisa gagal. Bagian ini menjelaskan gejala dan langkah pemecahan masalah untuk kondisi ini.

Gejala

Jika titik pemeriksaan gagal untuk aplikasi Anda, `numberOfFailedCheckpoints` akan lebih besar dari nol.

Titik pemeriksaan bisa gagal karena kegagalan langsung, seperti kesalahan aplikasi, atau karena kegagalan sementara, seperti kehabisan sumber daya aplikasi. Periksa log dan metrik aplikasi Anda untuk gejala berikut:

- Kesalahan dalam kode Anda.
- Kesalahan mengakses layanan dependen aplikasi Anda.
- Kesalahan serialisasi data. Jika serializer default tidak dapat membuat serialisasi data aplikasi Anda, aplikasi akan gagal. Untuk informasi tentang penggunaan serializer kustom dalam aplikasi Anda, lihat Serializer Kustom di Dokumentasi Apache Flink.
- Kesalahan Kehabisan Memori.
- Lonjakan atau peningkatan stabil dalam metrik berikut:

- `heapMemoryUtilization`
- `oldGenerationGCTime`
- `oldGenerationGCCount`
- `lastCheckpointSize`
- `lastCheckpointDuration`

Untuk informasi selengkapnya tentang memantau titik pemeriksaan, lihat Memantau Checkpointing di Dokumentasi Apache Flink.

Penyebab dan solusi

Pesan kesalahan log aplikasi Anda menunjukkan penyebab kegagalan langsung. Kegagalan sementara dapat disebabkan hal berikut:

- Persediaan KPU aplikasi Anda tidak cukup. Untuk informasi tentang meningkatkan persediaan aplikasi Anda, lihat Penskalaan.
- Ukuran status aplikasi Anda terlalu besar. Anda dapat memantau ukuran status aplikasi Anda menggunakan metrik `lastCheckpointSize`.
- Data status aplikasi Anda tidak didistribusikan secara merata di antara kunci. Jika aplikasi Anda menggunakan operator `KeyBy`, pastikan data yang masuk dibagi rata di antara kunci. Jika sebagian besar data ditetapkan ke satu kunci, ini membuat hambatan yang menyebabkan kegagalan.
- Aplikasi Anda mengalami tekanan balik memori atau pengumpulan sampah. Pantau `heapMemoryUtilization`, `oldGenerationGCTime`, dan `oldGenerationGCCount` aplikasi Anda untuk lonjakan atau nilai yang terus meningkat.

Kegagalan pos pemeriksaan untuk aplikasi Apache Beam

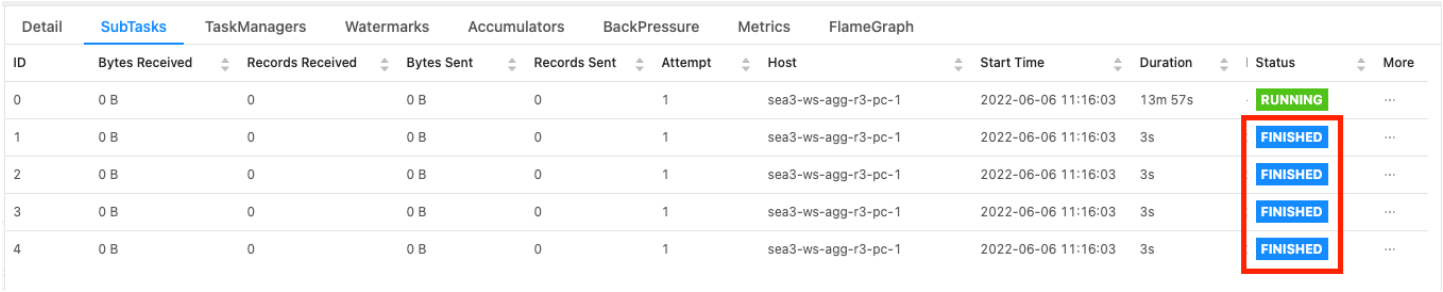
Jika aplikasi Beam Anda dikonfigurasi dengan`shutdownSourcesAfterIdleMs`disetel ke 0ms, pos pemeriksaan dapat gagal dipicu karena tugas dalam status “SELESAI”. Bagian ini menjelaskan gejala dan resolusi untuk kondisi ini.

Gejala

Buka Layanan Terkelola Anda untuk aplikasi Apache FlinkCloudWatchlog dan periksa apakah pesan log berikut telah dicatat. Pesan log berikut menunjukkan bahwa pos pemeriksaan gagal dipicu karena beberapa tugas telah selesai.

```
                {
                "locationInformation":
"org.apache.flink.runtime.checkpoint.CheckpointCoordinator.onTriggerFailure(CheckpointCoordina
                "logger": "org.apache.flink.runtime.checkpoint.CheckpointCoordinator",
                "message": "Failed to trigger checkpoint for job your job ID since some
tasks of job your job ID has been finished, abort the checkpoint Failure reason: Not
all required tasks are currently running.",
                "threadName": "Checkpoint Timer",
                "applicationARN": your application ARN,
                "applicationVersionId": "5",
                "messageSchemaVersion": "1",
                "messageType": "INFO"
                }
```

Ini juga dapat ditemukan di dasbor Flink di mana beberapa tugas telah memasuki status “SELESAI”, dan pos pemeriksaan tidak dimungkinkan lagi.



Penyebab

shutdownSourcesAfterIdleMsadalah variabel konfigurasi Beam yang mematikan sumber yang telah menganggur selama waktu milidetik yang dikonfigurasi. Setelah sumber dimatikan, pos pemeriksaan tidak dimungkinkan lagi. Hal ini dapat menyebabkankegagalan pos pemeriksaan.

Salah satu penyebab tugas memasuki status “SELESAI” adalah ketikashutdownSourcesAfterIdleMsdiatur ke 0ms, yang berarti bahwa tugas yang menganggur akan segera dimatikan.

Solusi

Untuk mencegah tugas memasuki status “SELESAI” segera, aturshutdownSourcesAfterIdleMske long.max_value. Ini dapat dilakukan dengan dua cara:

- Opsi 1: Jika konfigurasi balok diatur di halaman konfigurasi aplikasi Managed Service for Apache Flink, maka Anda dapat menambahkan pasangan nilai kunci baru untuk disetelshutdpwnSourcesAfteridleMs sebagai berikut:

- Opsi 2: Jika konfigurasi balok Anda diatur dalam file JAR Anda, maka Anda dapat mengaturshutdownSourcesAfterIdleMssebagai berikut:

```
                        FlinkPipelineOptions options =
PipelineOptionsFactory.create().as(FlinkPipelineOptions.class); // Initialize Beam
Options object

                        options.setShutdownSourcesAfterIdleMs(Long.MAX_VALUE); // set
shutdownSourcesAfterIdleMs to Long.MAX_VALUE
                        options.setRunner(FlinkRunner.class);

                        Pipeline p = Pipeline.create(options); // attach specified
options to Beam pipeline
```

Tekanan balik

Flink menggunakan tekanan balik untuk menyesuaikan kecepatan pemrosesan masing-masing operator.

Operator dapat berjuang untuk terus memproses volume pesan yang diterimanya karena berbagai alasan. Operasi mungkin memerlukan lebih banyak sumber daya CPU daripada yang tersedia operator, Operator mungkin menunggu operasi I/O selesai. Jika operator tidak dapat memproses peristiwa dengan cukup cepat, itu membangun tekanan balik di operator hulu yang masuk ke operator lambat. Hal ini menyebabkan operator hulu melambat, yang selanjutnya dapat menyebarkan

tekanan balik ke sumber dan menyebabkan sumber beradaptasi dengan keseluruhan throughput aplikasi dengan memperlambat juga. Anda dapat menemukan deskripsi yang lebih dalam tentang tekanan balik dan cara kerjanyaBagaimana Apache Flink™ menangani tekanan balik.

Mengetahui operator mana dalam aplikasi yang lambat memberi Anda informasi penting untuk memahami akar penyebab masalah kinerja dalam aplikasi. Informasi backpressure adalahdiekspos melalui Dasbor Flink. Untuk mengidentifikasi operator lambat, cari operator dengan nilai tekanan balik tinggi yang paling dekat dengan wastafel (operator B pada contoh berikut). Operator yang menyebabkan kelambatan kemudian menjadi salah satu operator hilir (operator C dalam contoh). B dapat memproses peristiwa lebih cepat, tetapi ditekan kembali karena tidak dapat meneruskan output ke operator lambat yang sebenarnya C.

```
A (backpressured 93%) -> B (backpressured 85%) -> C (backpressured 11%) -> D
 (backpressured 0%)
```

Setelah Anda mengidentifikasi operator yang lambat, cobalah untuk memahami mengapa itu lambat. Mungkin ada banyak alasan dan terkadang tidak jelas apa yang salah dan dapat memerlukan berhari-hari debugging dan pembuatan profil untuk menyelesaikannya. Berikut adalah beberapa alasan yang jelas dan lebih umum, beberapa di antaranya dijelaskan lebih lanjut di bawah ini:

- Operator melakukan I/O lambat, misalnya, panggilan jaringan (pertimbangkan untuk menggunakan AsyncIO sebagai gantinya).
- Ada kemiringan dalam data dan satu operator menerima lebih banyak peristiwa daripada yang lain (verifikasi dengan melihat jumlah pesan masuk/keluar dari subtugas individu (yaitu, contoh dari operator yang sama) di dasbor Flink.
- Ini adalah operasi intensif sumber daya (jika tidak ada kemiringan data, pertimbangkan penskalaan untuk pekerjaan terikat CPU/memori atau peningkatan`ParallelismPerKPU`untuk pekerjaan terikat I/O)
- Logging ekstensif di operator (kurangi logging seminimal mungkin untuk aplikasi produksi atau pertimbangkan untuk mengirim output debug ke aliran data sebagai gantinya).

Menguji Throughput dengan Discarding Sink

TheBuang Wastafelcukup mengabaikan semua peristiwa yang diterimanya saat masih menjalankan aplikasi (aplikasi tanpa sink gagal dijalankan). Ini sangat berguna untuk pengujian throughput, pembuatan profil, dan untuk memverifikasi apakah aplikasi melakukan penskalaan dengan benar. Ini juga merupakan pemeriksaan kewarasan yang sangat pragmatis untuk memverifikasi apakah sink

menyebabkan tekanan balik atau aplikasi (tetapi hanya memeriksa metrik tekanan balik seringkali lebih mudah dan lebih mudah).

Dengan mengganti semua sink aplikasi dengan wastafel pembuangan dan membuat sumber tiruan yang menghasilkan data yang r misalnya data produksi, Anda dapat mengukur throughput maksimum aplikasi untuk pengaturan paralelisme tertentu. Anda kemudian juga dapat meningkatkan paralelisme untuk memverifikasi bahwa aplikasi menskalakan dengan benar dan tidak memiliki hambatan yang hanya muncul pada throughput yang lebih tinggi (misalnya, karena kemiringan data).

Data miring

Aplikasi Flink dijalankan pada cluster dengan cara terdistribusi. Untuk skala ke beberapa node, Flink menggunakan konsep aliran yang dikunci, yang pada dasarnya berarti bahwa peristiwa aliran dipartisi sesuai dengan kunci tertentu, misalnya, id pelanggan, dan Flink kemudian dapat memproses partisi yang berbeda pada node yang berbeda. Banyak operator Flink kemudian dievaluasi berdasarkan partisi ini, misalnya,Jendela yang dikunci,Fungsi ProsesdanAsinkron I/O.

Memilih kunci partisi sering tergantung pada logika bisnis. Pada saat yang sama, banyak praktik terbaik untuk, misalnya,DynamoDBdan Spark, sama-sama berlaku untuk Flink, termasuk:

- memastikan kardinalitas kunci partisi yang tinggi
- menghindari kemiringan dalam volume acara antar partisi

Anda dapat mengidentifikasi kemiringan di partisi dengan membandingkan catatan yang diterima/dikirim dari subtugas (yaitu, contoh operator yang sama) di dasbor Flink. Selain itu, Managed Service untuk pemantauan Apache Flink dapat dikonfigurasi untuk mengekspos metrik`numRecordsIn/OutdannumRecordsInPerSecond/OutPerSecond`pada tingkat subtask juga.

Negara miring

Untuk operator stateful, yaitu, operator yang mempertahankan status untuk logika bisnis mereka seperti windows, kemiringan data selalu mengarah ke kemiringan status. Beberapa subtugas menerima lebih banyak peristiwa daripada yang lain karena kemiringan data dan karenanya juga mempertahankan lebih banyak data dalam keadaan. Namun, bahkan untuk aplikasi yang memiliki partisi seimbang secara merata, mungkin ada kemiringan dalam berapa banyak data yang disimpan dalam keadaan. Misalnya, untuk jendela sesi, beberapa pengguna dan sesi masing-masing mungkin jauh lebih lama daripada yang lain. Jika sesi yang lebih panjang kebetulan menjadi bagian dari partisi

yang sama, itu dapat menyebabkan ketidakseimbangan ukuran status yang disimpan oleh subtugas yang berbeda dari operator yang sama.

Kemiringan status tidak hanya meningkatkan lebih banyak memori dan sumber daya disk yang dibutuhkan oleh subtugas individu, tetapi juga dapat menurunkan kinerja aplikasi secara keseluruhan. Saat aplikasi mengambil pos pemeriksaan atau savepoint, status operator dipertahankan ke Amazon S3, untuk melindungi status terhadap kegagalan node atau cluster. Selama proses ini (terutama dengan semantik sekali yang diaktifkan secara default pada Managed Service for Apache Flink), pemrosesan berhenti dari perspektif eksternal hingga checkpoint/savepoint selesai. Jika ada kemiringan data, waktu untuk menyelesaikan operasi dapat diikat oleh satu subtugas yang telah mengumpulkan jumlah status yang sangat tinggi. Dalam kasus ekstrim, mengambil pos pemeriksaan/savepoint dapat gagal karena satu subtugas tidak dapat mempertahankan status.

Jadi mirip dengan kemiringan data, kemiringan status secara substansional dapat memperlambat aplikasi.

Untuk mengidentifikasi kemiringan status, Anda dapat memanfaatkan dasbor Flink. Temukan pos pemeriksaan atau savepoint terbaru dan bandingkan jumlah data yang telah disimpan untuk subtugas individual dalam detailnya.

Mengintegrasikan dengan sumber daya di berbagai wilayah

Anda dapat mengaktifkan menggunakan`StreamingFileSink`untuk menulis ke bucket Amazon S3 di Wilayah yang berbeda dari Layanan Terkelola untuk aplikasi Apache Flink Anda melalui pengaturan yang diperlukan untuk replikasi lintas Wilayah dalam konfigurasi Flink. Untuk melakukan ini, ajukan tiket dukungan diAWS SupportPusat.

Riwayat Dokumen untuk Amazon Managed Service untuk Apache Flink

Tabel berikut menjelaskan perubahan penting pada dokumentasi sejak rilis terakhir Layanan Terkelola untuk Apache Flink.

- Versi API: 2018-05-23
- Pembaruan dokumentasi terbaru: 30 Agustus 2023

Perubahan	Deskripsi	Tanggal
Kinesis Data Analytics sekarang dikenal sebagai Managed Service untuk Apache Flink	Tidak ada perubahan pada titik akhir layanan, API, Antarmuka Baris Perintah, kebijakan akses IAM, CloudWatch Metrik, atau dasbor AWS Penagihan . Aplikasi Anda yang ada akan terus berfungsi seperti sebelumnya. Untuk informasi selengkapnya, lihat Apa itu Managed Service for Apache Flink?	Agustus 30, 2023
Support untuk Apache Flink versi 1.15.2	Managed Service untuk Apache Flink sekarang mendukung aplikasi yang menggunakan Apache Flink versi 1.15.2. Buat aplikasi Kinesis Data Analytics menggunakan API Tabel Apache Flink. Untuk informasi selengkapnya, lihat Membuat Aplikasi.	22 November 2022

Perubahan	Deskripsi	Tanggal
Support untuk Apache Flink versi 1.13.2	Managed Service untuk Apache Flink sekarang mendukung aplikasi yang menggunakan Apache Flink versi 1.13.2. Buat aplikasi Kinesis Data Analytics menggunakan API Tabel Apache Flink. Untuk informasi selengkapnya, lihat Memulai: Flink 1.13.2.	13 Oktober 2021
Dukungan untuk Python	Managed Service untuk Apache Flink sekarang mendukung aplikasi yang menggunakan Python dengan Apache Flink Table API & SQL. Untuk informasi selengkapnya, lihat Menggunakan Python.	25 Maret 2021
Dukungan untuk Apache Flink 1.11.1	Managed Service untuk Apache Flink sekarang mendukung aplikasi yang menggunakan Apache Flink 1.11.1. Buat aplikasi Kinesis Data Analytics menggunakan API Tabel Apache Flink. Untuk informasi selengkapnya, lihat Membuat Aplikasi.	19 November 2020
Dasbor Apache Flink	Gunakan Dasbor Apache Flink untuk memantau kesehatan dan performa aplikasi. Untuk informasi selengkapnya, lihat Dasbor Apache Flink.	19 November 2020

Perubahan	Deskripsi	Tanggal
Konsumen EFO	Buat aplikasi yang menggunakan konsumen Fan-Out yang Ditingkatkan (EFO) untuk membaca dari Kinesis Data Stream. Untuk informasi selengkapnya, lihat Konsumen EFO.	6 Oktober 2020
Apache Beam	Buat aplikasi yang menggunakan Apache Beam untuk memproses data streaming. Untuk informasi selengkapnya, lihat Menggunakan CloudFormation dengan Managed Service untuk Apache Flink.	15 September 2020
Performa	Cara memecahkan masalah performa aplikasi, dan cara membuat aplikasi berfungsi. Untuk informasi selengkapnya, lihat Performa.	21 Juli 2020
Keystore Kustom	Cara mengakses klaster Amazon MSK yang menggunakan keystore kustom untuk enkripsi dalam transit. Untuk informasi selengkapnya, lihat Toko Perwalian Kustom.	10 Juni 2020
CloudWatch Alarm	Rekomendasi untuk membuat CloudWatch alarm dengan Managed Service untuk Apache Flink. Untuk informasi selengkapnya, lihat Alarm.	5 Juni 2020

Perubahan	Deskripsi	Tanggal
CloudWatch Metrik Baru	Layanan Terkelola untuk Apache Flink sekarang memancarkan 22 metrik ke Amazon Metrics. CloudWatch Untuk informasi selengkapnya, lihat Metrik dan Dimensi dalam Layanan Terkelola untuk Apache Flink.	12 Mei 2020
CloudWatch Metrik Kustom	Tentukan metrik khusus aplikasi dan pancarkan ke Metrik Amazon. CloudWatch Untuk informasi selengkapnya, lihat Metrik Kustom.	12 Mei 2020
Contoh: Baca dari Aliran Kinesis di Akun Berbeda.	Pelajari cara mengakses aliran Kinesis di AWS akun lain di aplikasi Managed Service for Apache Flink Anda. Untuk informasi selengkapnya, lihat Lintas Akun.	30 Maret 2020
Dukungan untuk Apache Flink 1.8.2	Managed Service untuk Apache Flink sekarang mendukung aplikasi yang menggunakan Apache Flink 1.8.2. Gunakan Streaming FileSink konektor Flink untuk menulis output langsung ke S3. Untuk informasi selengkapnya, lihat Membuat Aplikasi.	17 Desember 2019

Perubahan	Deskripsi	Tanggal
Layanan Terkelola untuk Apache Flink VPC	Konfigurasikan Layanan Terkelola untuk aplikasi Apache Flink untuk terhubung ke cloud pribadi virtual. Untuk informasi selengkapnya, lihat Menggunakan Amazon VPC .	25 November 2019
Layanan Terkelola untuk Praktik Terbaik Apache Flink	Praktik terbaik untuk membuat dan mengelola Layanan Terkelola untuk aplikasi Apache Flink. Untuk informasi selengkapnya, lihat Praktik Terbaik.	14 Oktober 2019
Menganalisis Layanan Terkelola untuk Apache Flink Application Logs	Gunakan Wawasan CloudWatch Log untuk memantau Layanan Terkelola Anda untuk aplikasi Apache Flink. Untuk informasi selengkapnya, lihat Menganalisis Log.	26 Juni 2019
Layanan Terkelola untuk Properti Runtime Aplikasi Apache Flink	Bekerja dengan Properti Runtime di Managed Service untuk Apache Flink. Untuk informasi selengkapnya, lihat Properti Runtime.	24 Juni 2019
Menandai Layanan Terkelola untuk Aplikasi Apache Flink	Gunakan penandaan aplikasi untuk menentukan biaya per aplikasi, kontrol akses, atau untuk tujuan yang ditetapkan pengguna. Untuk informasi selengkapnya, lihat Menggunakan Penandaan.	8 Mei 2019

Perubahan	Deskripsi	Tanggal
Layanan Terkelola untuk Apache Flink Contoh Aplikasi	Contoh aplikasi untuk Managed Service untuk Apache Flink mendemonstrasikan operator jendela dan menulis output ke Log. CloudWatch Untuk informasi selengkapnya, lihat Contoh-contoh.	1 Mei 2019
Logging Layanan Terkelola untuk Panggilan API Apache Flink dengan AWS CloudTrail	Managed Service for Apache Flink terintegrasi denganAWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau AWS layanan di Managed Service untuk Apache Flink. Untuk informasi selengkapnya, lihat Menggunakan AWS CloudTrail.	22 Maret 2019
Buat Aplikasi (Sink Kinesis Data Firehose)	Latihan untuk membuat Layanan Terkelola untuk Apache Flink dengan aliran data Amazon Kinesis sebagai sumber, dan aliran Amazon Kinesis Data Firehose sebagai wastafel. Untuk informasi selengkapnya, lihat Sink Kinesis Data Firehose.	13 Desember 2018
Rilis publik	Ini adalah rilis awal dari Managed Service for Apache Flink Developer Guide for Java Applications.	27 November 2018

Layanan Terkelola untuk Kode Contoh API Apache Flink

Topik ini berisi contoh blok permintaan untuk Layanan Terkelola untuk tindakan Apache Flink.

Untuk menggunakan JSON sebagai input untuk tindakan dengan AWS Command Line Interface (AWS CLI), simpan permintaan dalam file JSON. Selanjutnya teruskan nama file ke dalam tindakan menggunakan parameter `--cli-input-json`.

Contoh berikut menunjukkan cara menggunakan file JSON dengan tindakan.

```
$ aws kinesisanalyticsv2 start-application --cli-input-json file://start.json
```

Untuk informasi selengkapnya tentang penggunaan JSON dengan AWS CLI, lihat Membuat Parameter JSON Skeleton CLI dan Input CLI di Panduan Pengguna AWS Command Line Interface.

Topik

- AddApplicationCloudWatchLoggingOption
- AddApplicationInput
- AddApplicationInputProcessingConfiguration
- AddApplicationOutput
- AddApplicationReferenceDataSource
- AddApplicationVpcConfiguration
- CreateApplication
- CreateApplicationSnapshot
- DeleteApplication
- DeleteApplicationCloudWatchLoggingOption
- DeleteApplicationInputProcessingConfiguration
- DeleteApplicationOutput
- DeleteApplicationReferenceDataSource
- DeleteApplicationSnapshot
- DeleteApplicationVpcConfiguration
- DescribeApplication
- DescribeApplicationSnapshot

- DiscoverInputSchema
- ListApplications
- ListApplicationSnapshots
- StartApplication
- StopApplication
- UpdateApplication

AddApplicationCloudWatchLoggingOption

Berikut contoh kode permintaan untukAddApplicationCloudWatchLoggingOptiontindakan menambahkan AmazonCloudWatchopsi logging ke Layanan Terkelola untuk aplikasi Apache Flink:

```
{
    "ApplicationName": "MyApplication",
    "CloudWatchLoggingOption": {
        "LogStreamARN": "arn:aws:logs:us-east-1:123456789123:log-group:my-log-
group:log-stream:My-LogStream"
    },
    "CurrentApplicationVersionId": 2
}
```

AddApplicationInput

Berikut contoh kode permintaan untukAddApplicationInputtindakan menambahkan input aplikasi ke Layanan Terkelola untuk aplikasi Apache Flink:

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 2,
   "Input": {
      "InputParallelism": {
         "Count": 2
      },
      "InputSchema": {
         "RecordColumns": [
            {
               "Mapping": "$.TICKER",
               "Name": "TICKER_SYMBOL",
```

```
                "SqlType": "VARCHAR(50)"
            },
            {
                "SqlType": "REAL",
                "Name": "PRICE",
                "Mapping": "$.PRICE"
            }
         ],
         "RecordEncoding": "UTF-8",
         "RecordFormat": {
            "MappingParameters": {
               "JSONMappingParameters": {
                  "RecordRowPath": "$"
               }
            },
            "RecordFormatType": "JSON"
         }
      },
      "KinesisStreamsInput": {
         "ResourceARN": "arn:aws:kinesis:us-east-1:012345678901:stream/
ExampleInputStream"
      }
   }
}
```

AddApplicationInputProcessingConfiguration

Berikut contoh kode permintaan untukAddApplicationInputProcessingConfigurationaction menambahkan konfigurasi pemrosesan input aplikasi ke Layanan Terkelola untuk aplikasi Apache Flink:

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 2,
   "InputId": "2.1",
   "InputProcessingConfiguration": {
      "InputLambdaProcessor": {
         "ResourceARN": "arn:aws:lambda:us-
east-1:012345678901:function:MyLambdaFunction"
      }
   }
}
```

AddApplicationOutput

Berikut contoh kode permintaan untukAddApplicationOutputaction menambahkan aliran data Kinesis sebagai output aplikasi ke Managed Service untuk aplikasi Apache Flink:

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 2,
   "Output": {
      "DestinationSchema": {
         "RecordFormatType": "JSON"
      },
      "KinesisStreamsOutput": {
         "ResourceARN": "arn:aws:kinesis:us-east-1:012345678901:stream/
ExampleOutputStream"
      },
      "Name": "DESTINATION_SQL_STREAM"
   }
}
```

AddApplicationReferenceDataSource

Berikut contoh kode permintaan untukAddApplicationReferenceDataSourcetindakan menambahkan sumber data referensi aplikasi CSV ke Layanan Terkelola untuk aplikasi Apache Flink:

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 5,
   "ReferenceDataSource": {
      "ReferenceSchema": {
         "RecordColumns": [
            {
               "Mapping": "$.TICKER",
               "Name": "TICKER",
               "SqlType": "VARCHAR(4)"
            },
            {
               "Mapping": "$.COMPANYNAME",
               "Name": "COMPANY_NAME",
               "SqlType": "VARCHAR(40)"
            },
         ],
```

```
         "RecordEncoding": "UTF-8",
         "RecordFormat": {
            "MappingParameters": {
               "CSVMappingParameters": {
                  "RecordColumnDelimiter": " ",
                  "RecordRowDelimiter": "\r\n"
               }
            },
            "RecordFormatType": "CSV"
         }
      },
      "S3ReferenceDataSource": {
         "BucketARN": "arn:aws:s3:::MyS3Bucket",
         "FileKey": "TickerReference.csv"
      },
      "TableName": "string"
   }
}
```

AddApplicationVpcConfiguration

Kode permintaan contoh untuk tindakan AddApplicationVpcConfiguration berikut menambahkan konfigurasi VPC ke aplikasi yang sudah ada.

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 9,
   "VpcConfiguration": {
      "SecurityGroupIds": [ "sg-0123456789abcdef0" ],
      "SubnetIds": [ "subnet-0123456789abcdef0" ]
   }
}
```

CreateApplication

Berikut contoh kode permintaan untukCreateApplicationtindakan membuat Layanan Terkelola untuk aplikasi Apache Flink:

```
{
  "ApplicationName":"MyApplication",
```

```
  "ApplicationDescription":"My-Application-Description",
  "RuntimeEnvironment":"FLINK-1_15",
  "ServiceExecutionRole":"arn:aws:iam::123456789123:role/myrole",
  "CloudWatchLoggingOptions":[
    {
      "LogStreamARN":"arn:aws:logs:us-east-1:123456789123:log-group:my-log-group:log-
stream:My-LogStream"
    }
  ],
  "ApplicationConfiguration": {
    "EnvironmentProperties":
      {"PropertyGroups":
        [
          {"PropertyGroupId": "ConsumerConfigProperties",
            "PropertyMap":
              {"aws.region": "us-east-1",
              "flink.stream.initpos": "LATEST"}
          },
          {"PropertyGroupId": "ProducerConfigProperties",
            "PropertyMap":
              {"aws.region": "us-east-1"}
          },
        ]
      },
    "ApplicationCodeConfiguration":{
      "CodeContent":{
        "S3ContentLocation":{
          "BucketARN":"arn:aws:s3:::mybucket",
          "FileKey":"myflink.jar",
          "ObjectVersion":"AbCdEfGhIjKlMnOpQrStUvWxYz12345"
        }
      },
      "CodeContentType":"ZIPFILE"
    },
      "FlinkApplicationConfiguration":{
      "ParallelismConfiguration":{
        "ConfigurationType":"CUSTOM",
        "Parallelism":2,
        "ParallelismPerKPU":1,
        "AutoScalingEnabled":true
      }
    }
  }
```

```
}
```

CreateApplicationSnapshot

Berikut contoh kode permintaan untukCreateApplicationSnapshotaction membuat snapshot dari status aplikasi:

```
{
   "ApplicationName": "MyApplication",
   "SnapshotName": "MySnapshot"
}
```

DeleteApplication

Berikut contoh kode permintaan untukDeleteApplicationtindakan menghapus Layanan Terkelola untuk aplikasi Apache Flink:

```
{"ApplicationName": "MyApplication",
"CreateTimestamp": 12345678912}
```

DeleteApplicationCloudWatchLoggingOption

Berikut contoh kode permintaan untukDeleteApplicationCloudWatchLoggingOptiontindakan menghapus AmazonCloudWatchopsi logging dari Layanan Terkelola untuk aplikasi Apache Flink:

```
{
    "ApplicationName": "MyApplication",
    "CloudWatchLoggingOptionId": "3.1"
    "CurrentApplicationVersionId": 3
}
```

DeleteApplicationInputProcessingConfiguration

Berikut contoh kode permintaan untukDeleteApplicationInputProcessingConfigurationtindakan menghapus konfigurasi pemrosesan input dari Layanan Terkelola untuk aplikasi Apache Flink:

```
{
```

```
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 4,
   "InputId": "2.1"
}
```

DeleteApplicationOutput

Berikut contoh kode permintaan untukDeleteApplicationOutputtindakan menghapus output aplikasi dari Layanan Terkelola untuk aplikasi Apache Flink:

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 4,
   "OutputId": "4.1"
}
```

DeleteApplicationReferenceDataSource

Berikut contoh kode permintaan untukDeleteApplicationReferenceDataSourcetindakan menghapus sumber data referensi aplikasi dari Layanan Terkelola untuk aplikasi Apache Flink:

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 5,
   "ReferenceId": "5.1"
}
```

DeleteApplicationSnapshot

Berikut contoh kode permintaan untukDeleteApplicationSnapshottindakan menghapus snapshot dari status aplikasi:

```
{
   "ApplicationName": "MyApplication",
   "SnapshotCreationTimestamp": 12345678912,
   "SnapshotName": "MySnapshot"
}
```

DeleteApplicationVpcConfiguration

Kode permintaan contoh untuk tindakan DeleteApplicationVpcConfiguration berikut menghapus konfigurasi VPC yang ada dari aplikasi.

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 9,
   "VpcConfigurationId": "1.1"
}
```

DescribeApplication

Berikut contoh kode permintaan untukDescribeApplicationtindakan mengembalikan detail tentang Layanan Terkelola untuk aplikasi Apache Flink:

```
{"ApplicationName": "MyApplication"}
```

DescribeApplicationSnapshot

Berikut contoh kode permintaan untukDescribeApplicationSnapshottindakan mengembalikan rincian tentang snapshot dari status aplikasi:

```
{
   "ApplicationName": "MyApplication",
   "SnapshotName": "MySnapshot"
}
```

DiscoverInputSchema

Berikut contoh kode permintaan untukDiscoverInputSchematindakan menghasilkan skema dari sumber streaming:

```
{
   "InputProcessingConfiguration": {
      "InputLambdaProcessor": {
```

```
         "ResourceARN": "arn:aws:lambda:us-
east-1:012345678901:function:MyLambdaFunction"
      }
   },
   "InputStartingPositionConfiguration": {
      "InputStartingPosition": "NOW"
   },
   "ResourceARN": "arn:aws:kinesis:us-east-1:012345678901:stream/ExampleInputStream",
   "S3Configuration": {
      "BucketARN": "string",
      "FileKey": "string"
   },
   "ServiceExecutionRole": "string"
}
```

Berikut contoh kode permintaan untukDiscoverInputSchematindakan menghasilkan skema dari sumber referensi:

```
{
   "S3Configuration": {
      "BucketARN": "arn:aws:s3:::mybucket",
      "FileKey": "TickerReference.csv"
   },
   "ServiceExecutionRole": "arn:aws:iam::123456789123:role/myrole"
}
```

ListApplications

Berikut contoh kode permintaan untukListApplicationsaction mengembalikan daftar Managed Service untuk aplikasi Apache Flink di akun Anda:

```
{
   "ExclusiveStartApplicationName": "MyApplication",
   "Limit": 50
}
```

ListApplicationSnapshots

Berikut contoh kode permintaan untukListApplicationSnapshotsaction mengembalikan daftar snapshot dari status aplikasi:

```
{"ApplicationName": "MyApplication",
   "Limit": 50,
   "NextToken": "aBcDeFgHiJkLmNoPqRsTuVwXyZ0123"
}
```

StartApplication

Berikut contoh kode permintaan untukStartApplicationtindakan memulai Layanan Terkelola untuk aplikasi Apache Flink, dan memuat status aplikasi dari snapshot terbaru (jika ada):

```
{
    "ApplicationName": "MyApplication",
    "RunConfiguration": {
        "ApplicationRestoreConfiguration": {
         "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"
         }
    }
}
```

StopApplication

Berikut contoh kode permintaan untukAPI_StopApplicationtindakan menghentikan Layanan Terkelola untuk aplikasi Apache Flink:

```
{"ApplicationName": "MyApplication"}
```

UpdateApplication

Berikut contoh kode permintaan untukUpdateApplicationtindakan memperbarui Layanan Terkelola untuk aplikasi Apache Flink untuk mengubah lokasi kode aplikasi:

```
{"ApplicationName": "MyApplication",
"CurrentApplicationVersionId": 1,
"ApplicationConfigurationUpdate": {
      "ApplicationCodeConfigurationUpdate": {
         "CodeContentTypeUpdate": "ZIPFILE",
         "CodeContentUpdate": {
            "S3ContentLocationUpdate": {
```

```
                "BucketARNUpdate": "arn:aws:s3:::my_new_bucket",
                "FileKeyUpdate": "my_new_code.zip",
                "ObjectVersionUpdate": "2"
            }
        }
    }
}
```

Layanan Terkelola untuk Referensi API Apache Flink

Untuk informasi tentang API yang disediakan oleh Managed Service for Apache Flink, lihat Managed Service for Apache Flink API Reference.