aws

Panduan Developer

Amazon Data Firehose

Amazon Data Firehose: Panduan Developer

Table of Contents

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

Apa itu Amazon Data Firehose?

Amazon Data Firehose adalah layanan yang dikelola sepenuhnya untuk mengirimkan data streaming real-time ke tujuan seperti Amazon Simple Storage Service (Amazon S3), Amazon Redshift, Amazon Service, Amazon Serverless, Splunk, Apache Iceberg Tables, dan titik akhir HTTP kustom atau titik akhir HTTP apa pun yang dimiliki oleh penyedia layanan pihak ketiga yang didukung, termasuk LogicMonitor Datadog, Dynatrace, MongoDB, Relik Baru, Coralogix, dan Elastis. OpenSearch OpenSearch Dengan Amazon Data Firehose, Anda tidak perlu menulis aplikasi atau mengelola sumber daya. Anda mengonfigurasi produsen data Anda untuk mengirim data ke Amazon Data Firehose, dan secara otomatis mengirimkan data ke tujuan yang Anda tentukan. Anda juga dapat mengonfigurasi Amazon Data Firehose untuk mengubah data Anda sebelum mengirimkannya.

Untuk informasi selengkapnya tentang solusi AWS big data, lihat Big Data di AWS. Untuk informasi selengkapnya tentang solusi data streaming AWS , lihat Apa Itu Data Streaming?

Pelajari konsep-konsep kunci

Saat Anda memulai Amazon Data Firehose, Anda bisa mendapatkan keuntungan dari memahami konsep-konsep berikut.

Aliran Firehose

Entitas yang mendasari Amazon Data Firehose. Anda menggunakan Amazon Data Firehose dengan membuat aliran Firehose dan kemudian mengirim data ke sana. Untuk informasi selengkapnya, lihat Tutorial: Membuat aliran Firehose dari konsol dan Mengirim data ke aliran Firehose.

Rekam

Data yang menarik yang dikirim oleh produsen data Anda ke aliran Firehose. Sebuah catatan bisa berukuran sebesar 1.000 KB.

Penghasil data

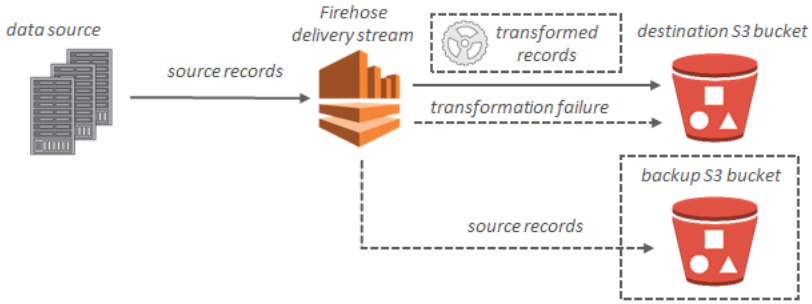
Produsen mengirim catatan ke aliran Firehose. Misalnya, server web yang mengirimkan data log ke aliran Firehose adalah produsen data. Anda juga dapat mengonfigurasi aliran Firehose untuk secara otomatis membaca data dari aliran data Kinesis yang ada, dan memuatnya ke tujuan. Untuk informasi selengkapnya, lihat Mengirim data ke aliran Firehose.

Ukuran buffer dan interval buffer

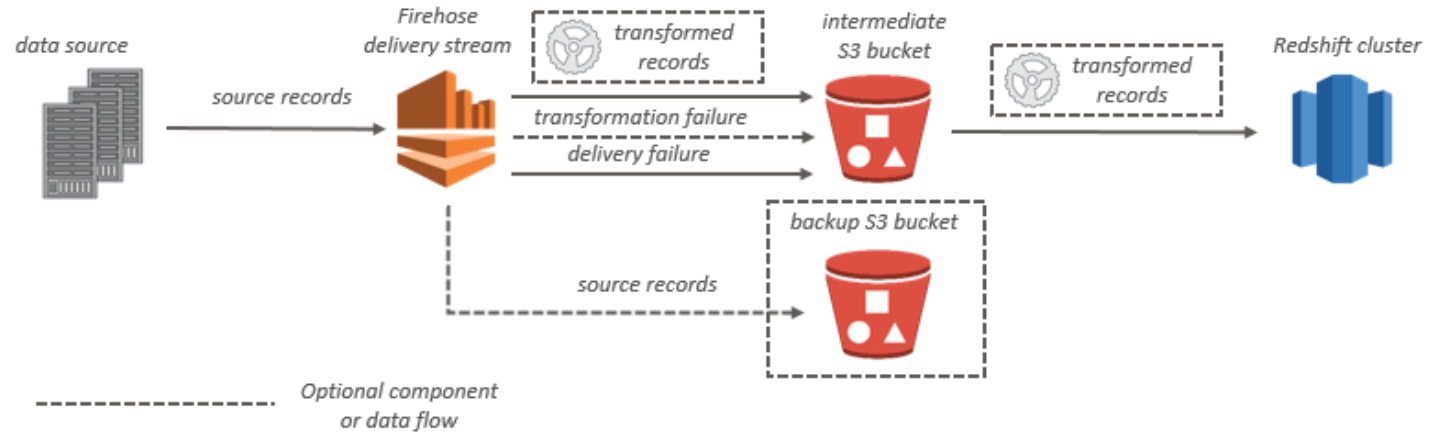
Amazon Data Firehose menyangga data streaming yang masuk ke ukuran tertentu atau untuk jangka waktu tertentu sebelum mengirimkannya ke tujuan. Buffer Sizemasuk MBs dan Buffer Interval dalam hitungan detik.

Memahami aliran data di Amazon Data Firehose

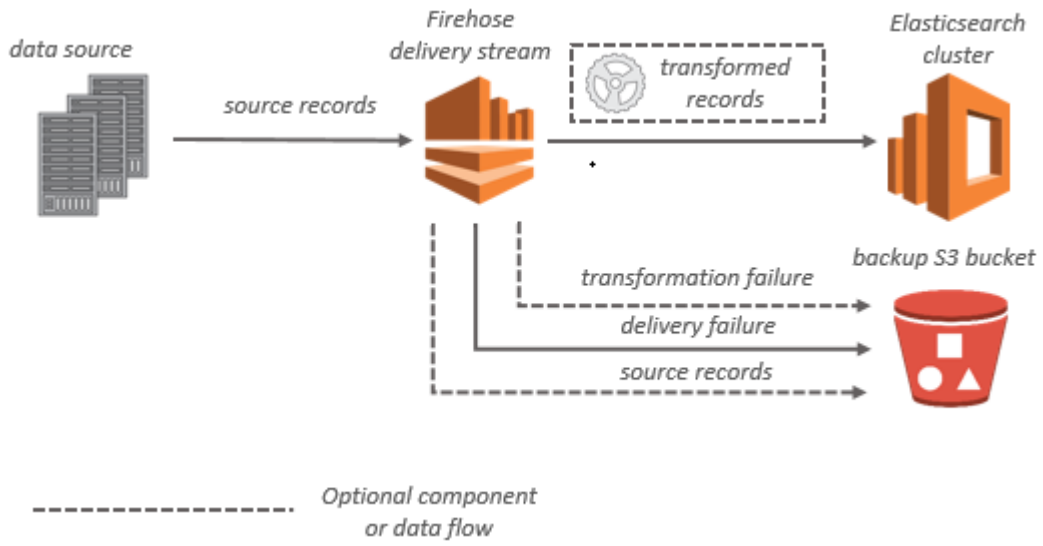
Untuk tujuan Amazon S3, data streaming dikirim ke bucket S3 Anda. Jika transformasi data diaktifkan, Anda dapat secara opsional mencadangkan data sumber ke bucket Amazon S3 lain.



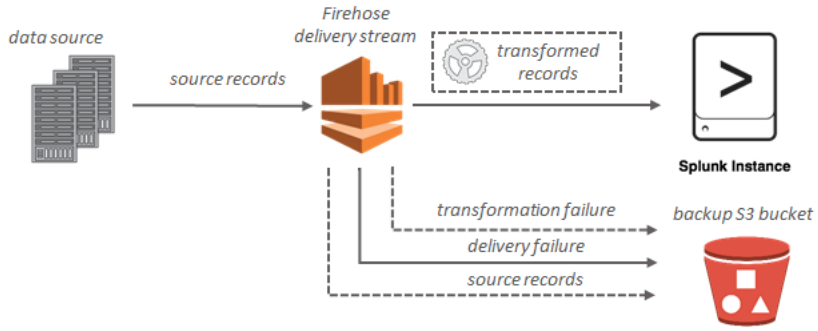
Untuk tujuan Amazon Redshift, data streaming dikirim ke bucket S3 Anda. Amazon Data Firehose kemudian mengeluarkan perintah Amazon COPY Redshift untuk memuat data dari bucket S3 Anda ke cluster Amazon Redshift Anda. Jika transformasi data diaktifkan, Anda dapat secara opsional mencadangkan data sumber ke bucket Amazon S3 lain.



Untuk tujuan OpenSearch Layanan, data streaming dikirimkan ke kluster OpenSearch Layanan Anda, dan secara opsional dapat dicadangkan ke bucket S3 Anda secara bersamaan.



Untuk tujuan Splunk, data streaming dikirim ke Splunk, dan secara opsional dapat dicadangkan ke bucket S3 Anda secara bersamaan.



Menggunakan Firehose dengan SDK AWS

AWS kit pengembangan perangkat lunak (SDKs) tersedia untuk banyak bahasa pemrograman populer. Setiap SDK menyediakan API, contoh kode, dan dokumentasi yang memudahkan developer untuk membangun aplikasi dalam bahasa pilihan mereka.

Dokumentasi SDK	Contoh kode
AWS SDK untuk C++	AWS SDK untuk C++ contoh kode
AWS CLI	AWS CLI contoh kode
AWS SDK untuk Go	AWS SDK untuk Go contoh kode

Dokumentasi SDK	Contoh kode
AWS SDK untuk Java	AWS SDK untuk Java contoh kode
AWS SDK untuk JavaScript	AWS SDK untuk JavaScript contoh kode
AWS SDK untuk Kotlin	AWS SDK untuk Kotlin contoh kode
AWS SDK untuk .NET	AWS SDK untuk .NET contoh kode
AWS SDK untuk PHP	AWS SDK untuk PHP contoh kode
Alat AWS untuk PowerShell	Alat AWS untuk PowerShell contoh kode
AWS SDK untuk Python (Boto3)	AWS SDK untuk Python (Boto3) contoh kode
AWS SDK untuk Ruby	AWS SDK untuk Ruby contoh kode
AWS SDK for Rust	AWS SDK for Rust contoh kode
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP contoh kode
AWS SDK for Swift	AWS SDK for Swift contoh kode

Ketersediaan contoh

Tidak dapat menemukan apa yang Anda butuhkan? Minta contoh kode menggunakan tautan Berikan umpan balik di bagian bawah halaman ini.

Prasyarat lengkap untuk menyiapkan Amazon Data Firehose

Sebelum Anda menggunakan Amazon Data Firehose untuk pertama kalinya, selesaikan tugas-tugas berikut.

Tugas

- Mendaftar untuk AWS
- (Opsional) Unduh perpustakaan dan alat

Mendaftar untuk AWS

Saat Anda mendaftar ke Amazon Web Services (AWS), AWS akun Anda secara otomatis mendaftar untuk semua layanan AWS, termasuk Amazon Data Firehose. Anda hanya membayar biaya layanan yang Anda gunakan.

Jika Anda sudah memiliki AWS akun, lompat ke tugas berikutnya. Jika Anda belum memiliki akun AWS , gunakan prosedur berikut untuk membuatnya.

Untuk mendaftar AWS akun

1. Buka https://portal.aws.amazon.com/billing/pendaftaran.
2. Ikuti petunjuk online.

   Bagian dari prosedur pendaftaran melibatkan menerima panggilan telepon atau pesan teks dan memasukkan kode verifikasi pada keypad telepon.

   Saat Anda mendaftar untuk sebuah Akun AWS, sebuah Pengguna root akun AWSdibuat. Pengguna root memiliki akses ke semua Layanan AWS dan sumber daya di akun. Sebagai praktik keamanan terbaik, tetapkan akses administratif ke pengguna, dan gunakan hanya pengguna root untuk melakukan tugas yang memerlukan akses pengguna root.

(Opsional) Unduh perpustakaan dan alat

Pustaka dan alat berikut akan membantu Anda bekerja dengan Amazon Data Firehose secara terprogram dan dari baris perintah:

- Operasi Firehose API adalah rangkaian operasi dasar yang didukung Amazon Data Firehose.

- AWS SDKs Untuk Go, Java, .NET, Node.js, Python, dan Ruby menyertakan dukungan Amazon Data Firehose dan sampel.

  Jika versi Anda AWS SDK untuk Java tidak menyertakan sampel untuk Amazon Data Firehose, Anda juga dapat mengunduh AWS SDK terbaru dari. GitHub
- AWS Command Line InterfaceMendukung Amazon Data Firehose. AWS CLI Ini memungkinkan Anda untuk mengontrol beberapa AWS layanan dari baris perintah dan mengotomatiskannya melalui skrip.

Tutorial: Membuat aliran Firehose dari konsol

Anda dapat menggunakan Konsol Manajemen AWS atau AWS SDK untuk membuat aliran Firehose ke tujuan yang Anda pilih.

Anda dapat memperbarui konfigurasi aliran Firehose kapan saja setelah dibuat, menggunakan konsol Amazon Data Firehose atau. UpdateDestination Aliran Firehose Anda tetap dalam `Active` status saat konfigurasi diperbarui, dan Anda dapat terus mengirim data. Konfigurasi yang diperbarui biasanya akan mulai berlaku dalam beberapa menit. Nomor versi aliran Firehose ditingkatkan dengan nilai 1 setelah Anda memperbarui konfigurasi. Hal ini tercermin dalam nama objek Amazon S3 yang dikirimkan. Untuk informasi selengkapnya, lihat Konfigurasikan format nama objek Amazon S3.

Lakukan langkah-langkah dalam topik berikut untuk membuat aliran Firehose.

Topik

Pilih sumber dan tujuan untuk aliran Firehose Anda

1. Buka konsol Firehose di. https://console.aws.amazon.com/firehose/
2. Pilih Buat aliran Firehose.
3. Pada halaman aliran Create Firehose, pilih sumber untuk aliran Firehose Anda dari salah satu opsi berikut.

    - Direct PUT - Pilih opsi ini untuk membuat aliran Firehose yang ditulis langsung oleh aplikasi produser. Berikut adalah daftar AWS layanan dan agen dan layanan open source yang terintegrasi dengan Direct PUT di Amazon Data Firehose. Daftar ini tidak lengkap, dan mungkin ada layanan tambahan yang dapat digunakan untuk mengirim data langsung ke Firehose.

- AWS SDK
- AWS Lambda
- AWS CloudWatch Log
- AWS CloudWatch Acara
- AWS Aliran Metrik Awan
- AWS IoT
- AWS Eventbridge
- Layanan Email Sederhana Amazon
- Amazon SNS
- AWS Log ACL web WAF
- Amazon API Gateway - Akses log
- Amazon Pinpoint
- Log Broker MSK Amazon
- Log kueri Amazon Route 53 Resolver
- AWS Log Peringatan Firewall Jaringan
- AWS Log Aliran Firewall Jaringan
- Amazon Elasticache Redis SLOWLOG
- Agen Kinesis (linux)
- Keran Kinesis (jendela)
- Fluentbit
- Lancar
- Apache Nifi
- Kepingan salju

- Amazon Kinesis Data Streams — Pilih opsi ini untuk mengonfigurasi aliran Firehose yang menggunakan aliran data Kinesis sebagai sumber data. Anda kemudian dapat menggunakan Firehose untuk membaca data dengan mudah dari aliran data Kinesis yang ada dan memuatnya ke tujuan. Untuk informasi selengkapnya tentang penggunaan Kinesis Data Streams sebagai sumber data, lihat Mengirim data ke aliran Firehose dengan Kinesis Data Streams.
- Amazon MSK — Pilih opsi ini untuk mengonfigurasi aliran Firehose yang menggunakan Amazon MSK sebagai sumber data. Anda kemudian dapat menggunakan Firehose untuk

membaca data dengan mudah dari kluster MSK Amazon yang ada dan memuatnya ke dalam bucket S3 tertentu. Untuk informasi selengkapnya, lihat Mengirim data ke aliran Firehose dengan Amazon MSK.

4. Pilih tujuan aliran Firehose Anda dari salah satu tujuan berikut yang didukung Firehose.

   - OpenSearch Layanan Amazon
   - Amazon Tanpa OpenSearch Server
   - Amazon Redshift
   - Amazon S3
   - Tabel Gunung Es Apache
   - Coralogix
   - Datadog
   - Dynatrace
   - Elastis
   - Titik Akhir HTTP
   - Honeycomb
   - Monitor Logika
   - Logz.io
   - Awan MongoDB
   - Relik Baru
   - Splunk
   - Cloud Observabilitas Splunk
   - Logika Sumo
   - Kepingan salju

5. Untuk nama aliran Firehose, Anda dapat menggunakan nama yang dihasilkan konsol untuk Anda atau menambahkan aliran Firehose pilihan Anda.

Konfigurasikan pengaturan sumber

Anda dapat mengonfigurasi pengaturan sumber berdasarkan sumber yang Anda pilih untuk mengirim informasi ke aliran Firehose dari konsol. Anda dapat mengonfigurasi pengaturan sumber untuk

Amazon MSK dan Amazon Kinesis Data Streams sebagai sumbernya. Tidak ada pengaturan sumber yang tersedia untuk Direct PUT sebagai sumbernya.

Konfigurasikan pengaturan sumber untuk Amazon MSK

Ketika Anda memilih Amazon MSK untuk mengirim informasi ke aliran Firehose, Anda dapat memilih antara kluster MSK yang disediakan dan MSK-Serverless. Anda kemudian dapat menggunakan Firehose untuk membaca data dengan mudah dari kluster dan topik MSK Amazon tertentu dan memuatnya ke tujuan S3 yang ditentukan.

Di bagian Pengaturan sumber halaman, berikan nilai untuk bidang berikut.

Konektivitas kluster MSK Amazon

Pilih salah satu broker bootstrap pribadi (disarankan) atau opsi pialang bootstrap publik berdasarkan konfigurasi cluster Anda. Broker Bootstrap adalah apa yang digunakan klien Apache Kafka sebagai titik awal untuk terhubung ke cluster. Broker bootstrap publik ditujukan untuk akses publik dari luar AWS, sedangkan broker bootstrap swasta dimaksudkan untuk akses dari dalam AWS. Untuk informasi selengkapnya tentang Amazon MSK, lihat Amazon Managed Streaming for Apache Kafka.

Untuk terhubung ke cluster MSK Amazon yang disediakan atau tanpa server melalui broker bootstrap pribadi, cluster harus memenuhi semua persyaratan berikut.

- Klaster harus aktif.
- Cluster harus memiliki IAM sebagai salah satu metode kontrol aksesnya.
- Konektivitas pribadi multi-VPC harus diaktifkan untuk metode kontrol akses IAM.
- Anda harus menambahkan ke klaster ini kebijakan berbasis sumber daya yang memberikan izin kepada kepala layanan Firehose untuk menjalankan operasi Amazon MSK API. `CreateVpcConnection`

Untuk terhubung ke cluster MSK Amazon yang disediakan melalui broker bootstrap publik, cluster harus memenuhi semua persyaratan berikut.

- Klaster harus aktif.
- Cluster harus memiliki IAM sebagai salah satu metode kontrol aksesnya.
- Cluster harus dapat diakses publik.

Akun klaster MSK

Anda dapat memilih akun tempat klaster MSK Amazon berada. Ini bisa menjadi salah satu dari berikut ini.

- Akun saat ini — Memungkinkan Anda untuk menelan data dari kluster MSK di akun saat ini AWS . Untuk ini, Anda harus menentukan ARN cluster MSK Amazon dari mana aliran Firehose Anda akan membaca data.
- Cross-account — Memungkinkan Anda untuk menelan data dari klaster MSK di akun lain. AWS Untuk informasi selengkapnya, lihat Cross-account pengiriman dari Amazon MSK.

Topik

Tentukan topik Apache Kafka dari mana Anda ingin aliran Firehose Anda untuk menelan data. Anda tidak dapat memperbarui topik ini setelah pembuatan aliran Firehose selesai.

> **Note**
>
> Firehose secara otomatis mendekompresi pesan Apache Kafka.

Konfigurasikan setelan sumber untuk Amazon Kinesis Data Streams

Konfigurasikan setelan sumber untuk Amazon Kinesis Data Streams untuk mengirim informasi ke aliran Firehose sebagai berikut.

> **Important**
>
> Jika Anda menggunakan Kinesis Producer Library (KPL) untuk menulis data ke aliran data Kinesis, Anda dapat menggunakan agregasi untuk menggabungkan catatan yang Anda tulis ke aliran data Kinesis tersebut. Jika Anda kemudian menggunakan aliran data tersebut sebagai sumber untuk aliran Firehose Anda, Amazon Data Firehose menghapus agregasi catatan sebelum mengirimkannya ke tujuan. Jika Anda mengonfigurasi aliran Firehose untuk mengubah data, Amazon Data Firehose melakukan de-agregasi catatan sebelum mengirimkannya. AWS Lambda Untuk informasi selengkapnya, lihat Mengembangkan Produsen Amazon Kinesis Data Streams Menggunakan Kinesis Producer Library dan Agregasi.

Di bawah pengaturan Sumber, pilih aliran yang ada di daftar aliran data Kinesis, atau masukkan ARN aliran data dalam format. `arn:aws:kinesis:[Region]:[AccountId]:stream/[StreamName]`

Jika Anda tidak memiliki aliran data yang ada, pilih Buat untuk membuat yang baru dari konsol Amazon Kinesis. Anda mungkin memerlukan peran IAM yang memiliki izin yang diperlukan pada aliran Kinesis. Untuk informasi selengkapnya, lihat ???. Setelah Anda membuat aliran baru, pilih ikon penyegaran untuk memperbarui daftar aliran Kinesis. Jika Anda memiliki sejumlah besar aliran, filter daftar menggunakan Filter berdasarkan nama.

> **Note**
>
> Saat Anda mengonfigurasi aliran data Kinesis sebagai sumber aliran Firehose, Firehose dan operasi Amazon Data akan dinonaktifkan. `PutRecord` `PutRecordBatch` Untuk menambahkan data ke aliran Firehose Anda dalam kasus ini, gunakan Kinesis Data Streams dan operasi. `PutRecord` `PutRecords`

Amazon Data Firehose mulai membaca data dari `LATEST` posisi aliran Kinesis Anda. Untuk informasi selengkapnya tentang posisi Kinesis Data Streams, lihat. GetShardIterator

Amazon Data Firehose memanggil operasi Kinesis Data GetRecordsStreams sekali per detik untuk setiap pecahan. Namun, ketika pencadangan penuh diaktifkan, Firehose memanggil operasi Kinesis Data `GetRecords` Streams dua kali per detik untuk setiap pecahan, satu untuk tujuan pengiriman utama dan satu lagi untuk pencadangan penuh.

Lebih dari satu aliran Firehose dapat membaca dari aliran Kinesis yang sama. Aplikasi Kinesis lainnya (konsumen) juga dapat membaca dari aliran yang sama. Setiap panggilan dari aliran Firehose atau aplikasi konsumen lainnya dihitung terhadap batas pelambatan keseluruhan untuk pecahan. Untuk menghindari throttling, rencanakan aplikasi Anda dengan hati-hati. Untuk informasi selengkapnya tentang batas Kinesis Data Streams, lihat Batas Amazon Kinesis Streams.

Lanjutkan ke langkah berikutnya untuk mengonfigurasi transformasi rekaman dan konversi format.

(Opsional) Konfigurasikan transformasi rekaman dan konversi format

Konfigurasikan Amazon Data Firehose untuk mengubah dan mengonversi data rekaman Anda.

Jika Anda memilih Amazon MSK sebagai sumber aliran Firehose Anda.

Di bagian Transform source records with AWS Lambda, berikan nilai untuk bidang berikut.

1. Transformasi data

   Untuk membuat aliran Firehose yang tidak mengubah data yang masuk, jangan centang kotak centang Aktifkan transformasi data.

   Untuk menentukan fungsi Lambda agar Firehose dipanggil dan digunakan untuk mengubah data yang masuk sebelum mengirimkannya, centang kotak centang Aktifkan transformasi data. Anda dapat mengonfigurasi fungsi Lambda baru menggunakan salah satu cetak biru Lambda atau memilih fungsi Lambda yang ada. Fungsi Lambda Anda harus berisi model status yang diperlukan oleh Firehose. Untuk informasi selengkapnya, lihat Mengubah data sumber di Amazon Data Firehose.

2. Di bagian Konversi format catatan, berikan nilai untuk bidang berikut:

   Rekam konversi format

   Untuk membuat aliran Firehose yang tidak mengonversi format rekaman data yang masuk, pilih Dinonaktifkan.

   Untuk mengonversi format catatan masuk, pilih Diaktifkan, lalu tentukan format output yang Anda inginkan. Anda perlu menentukan AWS Glue tabel yang menyimpan skema yang ingin Firehose gunakan untuk mengonversi format rekaman Anda. Untuk informasi selengkapnya, lihat Konversi format data masukan.

   Untuk contoh cara mengatur konversi format rekaman dengan CloudFormation, lihat AWS::KinesisFirehose: DeliveryStream.

Jika Anda memilih Amazon Kinesis Data Streams atau Direct PUT sebagai sumber aliran Firehose Anda

Di bagian Pengaturan sumber, berikan bidang berikut.

1. Di bawah Transform records, pilih salah satu dari berikut ini:

a. Jika tujuan Anda adalah Amazon S3 atau Splunk, di bagian Decompress source record Amazon CloudWatch Logs, pilih Aktifkan dekompresi.

b. Di bagian Transform source records with AWS Lambda, berikan nilai untuk bidang berikut:

Transformasi data

Untuk membuat aliran Firehose yang tidak mengubah data yang masuk, jangan centang kotak centang Aktifkan transformasi data.

Untuk menentukan fungsi Lambda agar Amazon Data Firehose dapat dipanggil dan digunakan untuk mengubah data yang masuk sebelum mengirimkannya, centang kotak centang Aktifkan transformasi data. Anda dapat mengonfigurasi fungsi Lambda baru menggunakan salah satu cetak biru Lambda atau memilih fungsi Lambda yang ada. Fungsi Lambda Anda harus berisi model status yang diperlukan oleh Amazon Data Firehose. Untuk informasi selengkapnya, lihat Mengubah data sumber di Amazon Data Firehose.

2. Di bagian Konversi format catatan, berikan nilai untuk bidang berikut:

Rekam konversi format

Untuk membuat aliran Firehose yang tidak mengonversi format rekaman data yang masuk, pilih Dinonaktifkan.

Untuk mengonversi format catatan masuk, pilih Diaktifkan, lalu tentukan format output yang Anda inginkan. Anda perlu menentukan AWS Glue tabel yang menyimpan skema yang ingin digunakan Amazon Data Firehose untuk mengonversi format rekaman Anda. Untuk informasi selengkapnya, lihat Konversi format data masukan.

Untuk contoh cara mengatur konversi format rekaman dengan CloudFormation, lihat AWS::KinesisFirehose: DeliveryStream.

Konfigurasikan pengaturan tujuan

Bagian ini menjelaskan pengaturan yang harus Anda konfigurasikan untuk aliran Firehose Anda berdasarkan tujuan yang Anda pilih.

Topik

- Konfigurasikan setelan tujuan untuk Amazon S3

- Konfigurasikan pengaturan tujuan untuk Apache Iceberg Tables
- Konfigurasikan setelan tujuan untuk Amazon Redshift
- Konfigurasikan setelan tujuan untuk OpenSearch Layanan
- Konfigurasikan pengaturan tujuan untuk Tanpa OpenSearch Server
- Konfigurasikan pengaturan tujuan untuk HTTP Endpoint
- Konfigurasikan pengaturan tujuan untuk Datadog
- Konfigurasikan pengaturan tujuan untuk Honeycomb
- Konfigurasikan pengaturan tujuan untuk Coralogix
- Konfigurasikan setelan tujuan untuk Dynatrace
- Konfigurasikan setelan tujuan untuk LogicMonitor
- Konfigurasikan pengaturan tujuan untuk Logz.io
- Konfigurasikan pengaturan tujuan untuk MongoDB Atlas
- Konfigurasikan pengaturan tujuan untuk New Relic
- Konfigurasikan pengaturan tujuan untuk Snowflake
- Konfigurasikan pengaturan tujuan untuk Splunk
- Konfigurasikan pengaturan tujuan untuk Splunk Observability Cloud
- Konfigurasikan pengaturan tujuan untuk Sumo Logic
- Konfigurasikan pengaturan tujuan untuk Elastic

Konfigurasikan setelan tujuan untuk Amazon S3

Anda harus menentukan pengaturan berikut agar dapat menggunakan Amazon S3 sebagai tujuan untuk aliran Firehose Anda.

- Masukkan nilai untuk bidang berikut.

  Ember S3

  Pilih bucket S3 yang Anda miliki tempat data streaming harus dikirim. Anda dapat membuat bucket S3 atau memilih yang sudah ada.

Pembatas baris baru

Anda dapat mengonfigurasi aliran Firehose untuk menambahkan pembatas baris baru di antara catatan dalam objek yang dikirimkan ke Amazon S3. Untuk melakukannya, pilih Diaktifkan. Untuk tidak menambahkan pembatas baris baru di antara catatan dalam objek yang dikirim ke Amazon S3, pilih Dinonaktifkan. Jika Anda berencana menggunakan Athena untuk menanyakan objek S3 dengan catatan agregat, aktifkan opsi ini.

Partisi dinamis

Pilih Diaktifkan untuk mengaktifkan dan mengonfigurasi partisi dinamis.

Deagregasi multi rekor

Ini adalah proses penguraian melalui catatan di aliran Firehose dan memisahkannya berdasarkan JSON yang valid atau pada pembatas baris baru yang ditentukan.

Jika Anda menggabungkan beberapa peristiwa, log, atau catatan ke dalam panggilan tunggal PutRecord dan PutRecordBatch API, Anda masih dapat mengaktifkan dan mengonfigurasi partisi dinamis. Dengan data agregat, saat Anda mengaktifkan partisi dinamis, Amazon Data Firehose mem-parsing catatan dan mencari beberapa objek JSON yang valid dalam setiap panggilan API. Ketika aliran Firehose dikonfigurasi dengan Kinesis Data Stream sebagai sumber, Anda juga dapat menggunakan agregasi bawaan di Kinesis Producer Library (KPL). Fungsionalitas partisi data dijalankan setelah data dideagregasi. Oleh karena itu, setiap rekaman di setiap panggilan API dapat dikirimkan ke awalan Amazon S3 yang berbeda. Anda juga dapat memanfaatkan integrasi fungsi Lambda untuk melakukan deagregasi lain atau transformasi lainnya sebelum fungsionalitas partisi data.

> **Important**
>
> Jika data Anda digabungkan, partisi dinamis hanya dapat diterapkan setelah deagregasi data dilakukan. Jadi, jika Anda mengaktifkan partisi dinamis ke data agregat Anda, Anda harus memilih Diaktifkan untuk mengaktifkan deagregasi multi rekaman.

Firehose stream melakukan langkah-langkah pemrosesan berikut dengan urutan sebagai berikut: deagregasi KPL (protobuf), deagregasi JSON atau pembatas, pemrosesan Lambda, partisi data, konversi format data, dan pengiriman Amazon S3.

Jenis deagregasi multi record

Jika Anda mengaktifkan deagregasi multi rekaman, Anda harus menentukan metode Firehose untuk mendeagregasi data Anda. Gunakan menu drop-down untuk memilih JSON atau Delimited.

Penguraian sebaris

Ini adalah salah satu mekanisme yang didukung untuk secara dinamis mempartisi data Anda yang terikat untuk Amazon S3. Untuk menggunakan penguraian inline untuk partisi dinamis data Anda, Anda harus menentukan parameter catatan data yang akan digunakan sebagai kunci partisi dan memberikan nilai untuk setiap kunci partisi yang ditentukan. Pilih Diaktifkan untuk mengaktifkan dan mengonfigurasi penguraian sebaris.

> **Important**
>
> Jika Anda menentukan fungsi AWS Lambda dalam langkah-langkah di atas untuk mengubah catatan sumber Anda, Anda dapat menggunakan fungsi ini untuk secara dinamis mempartisi data Anda yang terikat ke S3 dan Anda masih dapat membuat kunci partisi Anda dengan parsing inline. Dengan partisi dinamis, Anda dapat menggunakan penguraian sebaris atau fungsi AWS Lambda Anda untuk membuat kunci partisi Anda. Atau Anda dapat menggunakan penguraian inline dan fungsi AWS Lambda Anda secara bersamaan untuk membuat kunci partisi Anda.

Tombol partisi dinamis

Anda dapat menggunakan bidang Kunci dan Nilai untuk menentukan parameter catatan data yang akan digunakan sebagai kunci partisi dinamis dan kueri jq untuk menghasilkan nilai kunci partisi dinamis. Firehose hanya mendukung jq 1.6. Anda dapat menentukan hingga 50 tombol partisi dinamis. Anda harus memasukkan ekspresi jq yang valid untuk nilai kunci partisi dinamis agar berhasil mengonfigurasi partisi dinamis untuk aliran Firehose Anda.

Awalan ember S3

Saat mengaktifkan dan mengonfigurasi partisi dinamis, Anda harus menentukan awalan bucket S3 tempat Amazon Data Firehose akan mengirimkan data yang dipartisi.

Agar partisi dinamis dapat dikonfigurasi dengan benar, jumlah awalan bucket S3 harus identik dengan jumlah kunci partisi yang ditentukan.

Anda dapat mempartisi data sumber Anda dengan penguraian sebaris atau dengan fungsi Lambda yang Anda tentukan AWS . Jika Anda menetapkan fungsi AWS Lambda untuk membuat kunci partisi untuk data sumber Anda, Anda harus mengetikkan nilai awalan bucket S3 secara manual menggunakan format berikut: "Lambda:KeyID". partitionKeyFrom Jika Anda menggunakan penguraian inline untuk menentukan kunci partisi untuk data sumber Anda, Anda dapat mengetikkan nilai pratinjau bucket S3 secara manual menggunakan format berikut: "partitionKeyFromquery:keyID" atau Anda dapat memilih tombol Terapkan tombol partisi dinamis untuk menggunakan pasangan partisi dinamis Anda untuk menghasilkan awalan bucket S3 secara otomatis. key/value Saat mempartisi data dengan penguraian sebaris atau AWS Lambda, Anda juga dapat menggunakan formulir ekspresi berikut di awalan bucket S3:! {namespace:value}, di mana namespace dapat berupa Query atau Lambda. partitionKeyFrom partitionKeyFrom

Bucket S3 dan zona waktu awalan keluaran kesalahan S3

Pilih zona waktu yang ingin Anda gunakan untuk tanggal dan waktu di awalan khusus untuk objek Amazon S3. Secara default, Firehose menambahkan awalan waktu di UTC. Anda dapat mengubah zona waktu yang digunakan dalam awalan S3 jika Anda ingin menggunakan zona waktu yang berbeda.

Petunjuk penyangga

Firehose menyangga data yang masuk sebelum mengirimkannya ke tujuan yang ditentukan. Ukuran buffer yang direkomendasikan untuk tujuan bervariasi dari tiap-tiap penyedia layanan.

Kompresi S3

Pilih kompresi data GZIP, Snappy, Zip, atau Hadoop-Compatible Snappy, atau tanpa kompresi data. Kompresi Snappy, Zip, dan Snappy yang kompatibel dengan Hadoop tidak tersedia untuk aliran Firehose dengan tujuan Amazon Redshift.

Format ekstensi file S3 (opsional)

Tentukan format ekstensi file untuk objek yang dikirim ke bucket tujuan Amazon S3. Jika Anda mengaktifkan fitur ini, ekstensi file yang ditentukan akan mengganti ekstensi file default yang ditambahkan oleh Konversi Format Data atau fitur kompresi S3 seperti.parquet atau.gz. Pastikan jika Anda mengonfigurasi ekstensi file yang benar saat Anda menggunakan fitur ini dengan Konversi Format Data atau kompresi S3. Ekstensi file harus dimulai dengan titik (.) dan dapat berisi karakter yang diizinkan: 0-9a-z! -_.*' (). Ekstensi file tidak boleh melebihi 128 karakter.

Enkripsi S3

Firehose mendukung enkripsi sisi server Amazon S3 AWS Key Management Service dengan (SSE-KMS) untuk mengenkripsi data yang dikirimkan di Amazon S3. Anda dapat memilih untuk menggunakan jenis enkripsi default yang ditentukan dalam bucket S3 tujuan atau untuk mengenkripsi dengan kunci dari daftar AWS KMS kunci yang Anda miliki. Jika Anda mengenkripsi data dengan AWS KMS kunci, Anda dapat menggunakan kunci AWS terkelola default (aws/s3) atau kunci yang dikelola pelanggan. Untuk informasi selengkapnya, lihat Melindungi Data Menggunakan Enkripsi Sisi Server dengan Kunci yang Dikelola KMS (AWS SSE-KMS).

Konfigurasikan pengaturan tujuan untuk Apache Iceberg Tables

Firehose mendukung Apache Iceberg Tables sebagai tujuan di semua kecuali Wilayah AWSWilayah Tiongkok, Asia Pasifik (Taipei) AWS GovCloud (US) Regions, Asia Pasifik (Malaysia), Asia Pasifik (Selandia Baru), dan Meksiko (Tengah).

Untuk informasi selengkapnya tentang Apache Iceberg Tables sebagai tujuan Anda, lihat. Mengirimkan data ke Apache Iceberg Tables dengan Amazon Data Firehose

Konfigurasikan setelan tujuan untuk Amazon Redshift

Bagian ini menjelaskan setelan untuk menggunakan Amazon Redshift sebagai tujuan streaming Firehose Anda.

Pilih salah satu dari prosedur berikut berdasarkan apakah Anda memiliki klaster yang disediakan Amazon Redshift atau grup kerja Amazon Redshift Tanpa Server.

- Cluster Penyediaan Amazon Redshift
- Konfigurasikan setelan tujuan untuk workgroup Amazon Redshift Tanpa Server

> **Note**
>
> Firehose tidak dapat menulis ke cluster Amazon Redshift yang menggunakan perutean VPC yang disempurnakan.

Cluster Penyediaan Amazon Redshift

Bagian ini menjelaskan setelan untuk menggunakan klaster yang disediakan Amazon Redshift sebagai tujuan streaming Firehose Anda.

- Masukkan nilai untuk bidang berikut:

  Kluster

  Klaster Amazon Redshift tempat tujuan data bucket S3 disalin. Konfigurasikan klaster Amazon Redshift agar dapat diakses publik dan buka blokir alamat IP Amazon Data Firehose. Untuk informasi selengkapnya, lihat Berikan akses Firehose ke tujuan Amazon Redshift.

  Autentikasi

  Anda dapat memilih untuk memasukkan secara username/password langsung atau mengambil rahasia dari AWS Secrets Manager untuk mengakses cluster Amazon Redshift.

  - Nama pengguna

    Tentukan pengguna Amazon Redshift dengan izin untuk mengakses klaster Amazon Redshift. Pengguna ini harus memiliki izin `INSERT` Amazon Redshift untuk menyalin data dari bucket S3 ke klaster Amazon Redshift.

  - Kata Sandi

    Tentukan kata sandi untuk pengguna yang memiliki izin untuk mengakses cluster.

  - Rahasia

    Pilih rahasia dari AWS Secrets Manager yang berisi kredensil untuk cluster Amazon Redshift. Jika Anda tidak melihat rahasia Anda di daftar drop-down, buat satu AWS Secrets Manager untuk kredensi Amazon Redshift Anda. Untuk informasi selengkapnya, lihat Otentikasi dengan AWS Secrets Manager di Amazon Data Firehose.

  Basis Data

  Basis data Amazon Redshift tempat tujuan data disalin.

  Tabel

  Tabel Amazon Redshift tempat tujuan data disalin.

Kolom

(Opsional) Kolom tertentu dari tabel tempat tujuan data disalin. Gunakan opsi ini jika jumlah kolom yang didefinisikan di objek Amazon S3 Anda kurang dari jumlah kolom dalam tabel Amazon Redshift.

Tujuan S3 Menengah

Firehose mengirimkan data Anda ke bucket S3 Anda terlebih dahulu dan kemudian mengeluarkan COPY perintah Amazon Redshift untuk memuat data ke cluster Amazon Redshift Anda. Tentukan bucket S3 yang Anda miliki tempat data streaming harus dikirim. Anda dapat membuat bucket S3 atau memilih bucket yang Anda miliki.

Firehose tidak menghapus data dari bucket S3 Anda setelah memuatnya ke cluster Amazon Redshift Anda. Anda dapat mengelola data dalam bucket S3 Anda menggunakan konfigurasi siklus hidup. Untuk informasi selengkapnya, lihat Manajemen Siklus Hidup Objek di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Awalan S3 menengah

(Opsional) Untuk menggunakan prefiks default untuk objek Amazon S3, biarkan opsi ini kosong. Firehose secara otomatis menggunakan awalan dalam format waktu UTC "`YYYY/MM/dd/HH`" untuk objek Amazon S3 yang dikirimkan. Anda dapat menambahkan ke awal prefiks ini. Untuk informasi selengkapnya, lihat Konfigurasikan format nama objek Amazon S3.

Opsi COPY

Parameter yang dapat Anda tentukan di perintah COPY Amazon Redshift. Ini mungkin diperlukan untuk konfigurasi Anda. Misalnya, "`GZIP`" diperlukan jika kompresi data Amazon S3 diaktifkan. “`REGION`” diperlukan jika bucket S3 Anda tidak berada di AWS Wilayah yang sama dengan cluster Amazon Redshift Anda. Untuk informasi selengkapnya, lihat SALIN di Panduan Developer Basis Data Amazon Redshift.

perintah COPY

Perintah COPY Amazon Redshift. Untuk informasi selengkapnya, lihat SALIN di Panduan Developer Basis Data Amazon Redshift.

Coba lagi durasi

Durasi waktu (0—7200 detik) untuk Firehose mencoba lagi jika data ke klaster COPY Amazon Redshift Anda gagal. Firehose mencoba lagi setiap 5 menit sampai durasi coba lagi

berakhir. Jika Anda menyetel durasi coba lagi ke 0 (nol) detik, Firehose tidak akan mencoba lagi setelah COPY kegagalan perintah.

Petunjuk penyangga

Firehose menyangga data yang masuk sebelum mengirimkannya ke tujuan yang ditentukan. Ukuran buffer yang direkomendasikan untuk tujuan bervariasi dari tiap-tiap penyedia layanan.

Kompresi S3

Pilih kompresi data GZIP, Snappy, Zip, atau Hadoop-Compatible Snappy, atau tanpa kompresi data. Kompresi Snappy, Zip, dan Snappy yang kompatibel dengan Hadoop tidak tersedia untuk aliran Firehose dengan tujuan Amazon Redshift.

Format ekstensi file S3 (opsional)

Format ekstensi file S3 (opsional) — Tentukan format ekstensi file untuk objek yang dikirim ke bucket tujuan Amazon S3. Jika Anda mengaktifkan fitur ini, ekstensi file yang ditentukan akan mengganti ekstensi file default yang ditambahkan oleh Konversi Format Data atau fitur kompresi S3 seperti.parquet atau.gz. Pastikan jika Anda mengonfigurasi ekstensi file yang benar saat Anda menggunakan fitur ini dengan Konversi Format Data atau kompresi S3. Ekstensi file harus dimulai dengan titik (.) dan dapat berisi karakter yang diizinkan: 0-9a-z! -_.*' (). Ekstensi file tidak boleh melebihi 128 karakter.

Enkripsi S3

Firehose mendukung enkripsi sisi server Amazon S3 AWS Key Management Service dengan (SSE-KMS) untuk mengenkripsi data yang dikirimkan di Amazon S3. Anda dapat memilih untuk menggunakan jenis enkripsi default yang ditentukan dalam bucket S3 tujuan atau untuk mengenkripsi dengan kunci dari daftar AWS KMS kunci yang Anda miliki. Jika Anda mengenkripsi data dengan AWS KMS kunci, Anda dapat menggunakan kunci AWS terkelola default (aws/s3) atau kunci yang dikelola pelanggan. Untuk informasi selengkapnya, lihat Melindungi Data Menggunakan Enkripsi Sisi Server dengan Kunci yang Dikelola KMS (AWS SSE-KMS).

Konfigurasikan setelan tujuan untuk workgroup Amazon Redshift Tanpa Server

Bagian ini menjelaskan setelan untuk menggunakan workgroup Amazon Redshift Serverless sebagai tujuan streaming Firehose Anda.

- Masukkan nilai untuk bidang berikut:

Nama Workgroup

Workgroup Amazon Redshift Tanpa Server tempat data bucket S3 disalin. Konfigurasikan workgroup Amazon Redshift Tanpa Server agar dapat diakses publik dan buka blokir alamat IP Firehose. Untuk informasi selengkapnya, lihat bagian Connect ke instans Amazon Redshift Tanpa Server yang dapat diakses publik di Menyambung ke Amazon Redshift Tanpa Server dan juga. Berikan akses Firehose ke tujuan Amazon Redshift

Autentikasi

Anda dapat memilih untuk memasukkan secara username/password langsung atau mengambil rahasia dari AWS Secrets Manager untuk mengakses workgroup Amazon Redshift Tanpa Server.

- Nama pengguna

  Tentukan pengguna Amazon Redshift dengan izin untuk mengakses grup kerja Amazon Redshift Tanpa Server. Pengguna ini harus memiliki `INSERT` izin Amazon Redshift untuk menyalin data dari bucket S3 ke workgroup Amazon Redshift Serverless.

- Kata Sandi

  Tentukan kata sandi untuk pengguna yang memiliki izin untuk mengakses grup kerja Amazon Redshift Tanpa Server.

- Rahasia

  Pilih rahasia dari AWS Secrets Manager yang berisi kredensil untuk workgroup Amazon Redshift Serverless. Jika Anda tidak melihat rahasia Anda di daftar drop-down, buat satu AWS Secrets Manager untuk kredensi Amazon Redshift Anda. Untuk informasi selengkapnya, lihat Otentikasi dengan AWS Secrets Manager di Amazon Data Firehose.

Basis Data

Basis data Amazon Redshift tempat tujuan data disalin.

Tabel

Tabel Amazon Redshift tempat tujuan data disalin.

Kolom

(Opsional) Kolom tertentu dari tabel tempat tujuan data disalin. Gunakan opsi ini jika jumlah kolom yang didefinisikan di objek Amazon S3 Anda kurang dari jumlah kolom dalam tabel Amazon Redshift.

Tujuan S3 Menengah

Amazon Data Firehose mengirimkan data Anda ke bucket S3 Anda terlebih dahulu dan kemudian mengeluarkan COPY perintah Amazon Redshift untuk memuat data ke dalam grup kerja Amazon Redshift Tanpa Server Anda. Tentukan bucket S3 yang Anda miliki tempat data streaming harus dikirim. Anda dapat membuat bucket S3 atau memilih bucket yang Anda miliki.

Firehose tidak menghapus data dari bucket S3 Anda setelah memuatnya ke workgroup Amazon Redshift Tanpa Server Anda. Anda dapat mengelola data dalam bucket S3 Anda menggunakan konfigurasi siklus hidup. Untuk informasi selengkapnya, lihat Manajemen Siklus Hidup Objek di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Awalan S3 menengah

(Opsional) Untuk menggunakan prefiks default untuk objek Amazon S3, biarkan opsi ini kosong. Firehose secara otomatis menggunakan awalan dalam format waktu UTC "`YYYY/MM/dd/HH`" untuk objek Amazon S3 yang dikirimkan. Anda dapat menambahkan ke awal prefiks ini. Untuk informasi selengkapnya, lihat Konfigurasikan format nama objek Amazon S3.

Opsi COPY

Parameter yang dapat Anda tentukan di perintah COPY Amazon Redshift. Ini mungkin diperlukan untuk konfigurasi Anda. Misalnya, "`GZIP`" diperlukan jika kompresi data Amazon S3 diaktifkan. “`REGION`” diperlukan jika bucket S3 Anda tidak berada di AWS Wilayah yang sama dengan workgroup Amazon Redshift Tanpa Server Anda. Untuk informasi selengkapnya, lihat SALIN di Panduan Developer Basis Data Amazon Redshift.

perintah COPY

Perintah COPY Amazon Redshift. Untuk informasi selengkapnya, lihat SALIN di Panduan Developer Basis Data Amazon Redshift.

Coba lagi durasi

Durasi waktu (0—7200 detik) untuk Firehose mencoba lagi jika data ke grup kerja COPY Amazon Redshift Tanpa Server gagal. Firehose mencoba lagi setiap 5 menit sampai durasi coba lagi berakhir. Jika Anda menyetel durasi coba lagi ke 0 (nol) detik, Firehose tidak akan mencoba lagi setelah COPY kegagalan perintah.

Petunjuk penyangga

Firehose menyangga data yang masuk sebelum mengirimkannya ke tujuan yang ditentukan. Ukuran buffer yang direkomendasikan untuk tujuan bervariasi dari tiap-tiap penyedia layanan.

Kompresi S3

Pilih kompresi data GZIP, Snappy, Zip, atau Hadoop-Compatible Snappy, atau tanpa kompresi data. Kompresi Snappy, Zip, dan Snappy yang kompatibel dengan Hadoop tidak tersedia untuk aliran Firehose dengan tujuan Amazon Redshift.

Format ekstensi file S3 (opsional)

Format ekstensi file S3 (opsional) — Tentukan format ekstensi file untuk objek yang dikirim ke bucket tujuan Amazon S3. Jika Anda mengaktifkan fitur ini, ekstensi file yang ditentukan akan mengganti ekstensi file default yang ditambahkan oleh Konversi Format Data atau fitur kompresi S3 seperti.parquet atau.gz. Pastikan jika Anda mengonfigurasi ekstensi file yang benar saat Anda menggunakan fitur ini dengan Konversi Format Data atau kompresi S3. Ekstensi file harus dimulai dengan titik (.) dan dapat berisi karakter yang diizinkan: 0-9a-z! -_.*' (). Ekstensi file tidak boleh melebihi 128 karakter.

Enkripsi S3

Firehose mendukung enkripsi sisi server Amazon S3 AWS Key Management Service dengan (SSE-KMS) untuk mengenkripsi data yang dikirimkan di Amazon S3. Anda dapat memilih untuk menggunakan jenis enkripsi default yang ditentukan dalam bucket S3 tujuan atau untuk mengenkripsi dengan kunci dari daftar AWS KMS kunci yang Anda miliki. Jika Anda mengenkripsi data dengan AWS KMS kunci, Anda dapat menggunakan kunci AWS terkelola default (aws/s3) atau kunci yang dikelola pelanggan. Untuk informasi selengkapnya, lihat Melindungi Data Menggunakan Enkripsi Sisi Server dengan Kunci yang Dikelola KMS (AWS SSE-KMS).

Konfigurasikan setelan tujuan untuk OpenSearch Layanan

Firehose mendukung versi Elasticsearch — 1.5, 2.3, 5.1, 5.3, 5.5, 5.6, serta semua versi 6.*, 7.*, dan 8.*. Firehose mendukung Amazon OpenSearch Service 2.x dan 3.x.

Bagian ini menjelaskan opsi untuk menggunakan OpenSearch Layanan untuk tujuan Anda.

- Masukkan nilai untuk bidang berikut:

  OpenSearch Domain layanan

  Domain OpenSearch Layanan tempat data Anda dikirimkan.

  Indeks

  Nama indeks OpenSearch Layanan yang akan digunakan saat mengindeks data ke kluster OpenSearch Service Anda.

  Rotasi indeks

  Pilih apakah dan seberapa sering indeks OpenSearch Layanan harus diputar. Jika rotasi indeks diaktifkan, Amazon Data Firehose akan menambahkan tanda waktu yang sesuai ke nama indeks yang ditentukan dan dirotasi. Untuk informasi selengkapnya, lihat Konfigurasikan rotasi indeks untuk OpenSearch Layanan.

  Jenis

  Nama tipe OpenSearch Layanan yang akan digunakan saat mengindeks data ke kluster OpenSearch Service Anda. Untuk Elasticsearch 7.x dan OpenSearch 1.x, hanya ada satu jenis per indeks. Jika Anda mencoba menentukan jenis baru untuk indeks yang sudah ada yang sudah memiliki jenis lain, Firehose mengembalikan kesalahan saat runtime.

  Untuk Elasticsearch 7.x, biarkan bidang ini kosong.

  Coba lagi durasi

  Durasi waktu Firehose untuk mencoba lagi jika permintaan indeks gagal. OpenSearch Untuk durasi coba lagi, Anda dapat mengatur nilai apa pun antara 0-7200 detik. Durasi coba ulang default adalah 300 detik. Firehose akan mencoba lagi beberapa kali dengan mundur eksponensial hingga durasi percobaan ulang berakhir.

  Setelah durasi percobaan ulang berakhir, Firehose mengirimkan data ke Dead Letter Queue (DLQ), bucket kesalahan S3 yang dikonfigurasi. Untuk data yang dikirim ke DLQ, Anda harus

mengarahkan ulang data kembali dari bucket kesalahan S3 yang dikonfigurasi ke tujuan. OpenSearch

Jika Anda ingin memblokir aliran Firehose agar tidak mengirimkan data ke DLQ karena waktu henti atau pemeliharaan OpenSearch cluster, Anda dapat mengonfigurasi durasi coba lagi ke nilai yang lebih tinggi dalam hitungan detik. Anda dapat meningkatkan nilai durasi coba lagi di atas menjadi 7200 detik dengan menghubungi dukungan.AWS

Jenis DocumentID

Menunjukkan metode untuk mengatur ID dokumen. Metode yang didukung adalah ID dokumen buatan FireHose dan ID dokumen yang dihasilkan OpenSearch Layanan. ID dokumen yang dihasilkan Firehose adalah opsi default saat nilai ID dokumen tidak disetel. OpenSearch ID dokumen yang dihasilkan layanan adalah opsi yang direkomendasikan karena mendukung operasi berat tulis, termasuk analitik log dan observabilitas, mengkonsumsi lebih sedikit sumber daya CPU di domain OpenSearch Layanan dan dengan demikian, menghasilkan peningkatan kinerja.

Konektivitas VPC tujuan

Jika domain OpenSearch Layanan Anda berada dalam VPC pribadi, gunakan bagian ini untuk menentukan VPC tersebut. Tentukan juga subnet dan subgrup yang Anda inginkan Amazon Data Firehose untuk digunakan saat mengirimkan data ke domain Layanan Anda. OpenSearch Anda dapat menggunakan grup keamanan yang sama dengan yang digunakan domain OpenSearch Layanan. Jika Anda menentukan grup keamanan yang berbeda, pastikan bahwa mereka mengizinkan lalu lintas HTTPS keluar ke grup keamanan domain OpenSearch Layanan. Pastikan juga bahwa grup keamanan domain OpenSearch Layanan mengizinkan lalu lintas HTTPS dari grup keamanan yang Anda tentukan saat mengonfigurasi aliran Firehose. Jika Anda menggunakan grup keamanan yang sama untuk aliran Firehose dan domain OpenSearch Layanan, pastikan aturan masuk grup keamanan mengizinkan lalu lintas HTTPS. Untuk informasi lebih lanjut tentang aturan grup keamanan, lihat Aturan grup keamanan dalam Dokumentasi Amazon VPC.

> **Important**
>
> Saat Anda menentukan subnet untuk mengirimkan data ke tujuan dalam VPC pribadi, pastikan Anda memiliki cukup banyak alamat IP gratis di subnet yang dipilih. Jika tidak ada alamat IP gratis yang tersedia di subnet tertentu, Firehose tidak dapat

membuat atau ENIs menambahkan pengiriman data di VPC pribadi, dan pengiriman akan terdegradasi atau gagal.

Petunjuk penyangga

Amazon Data Firehose melakukan buffer pada data masuk sebelum mengirimkannya ke tujuan yang ditentukan. Ukuran buffer yang direkomendasikan untuk tujuan bervariasi dari tiap-tiap penyedia layanan.

Konfigurasikan pengaturan tujuan untuk Tanpa OpenSearch Server

Bagian ini menjelaskan opsi untuk menggunakan OpenSearch Tanpa Server untuk tujuan Anda.

- Masukkan nilai untuk bidang berikut:

  OpenSearch Koleksi tanpa server

  Titik akhir untuk sekelompok indeks OpenSearch Tanpa Server tempat data Anda dikirim.

  Indeks

  Nama indeks OpenSearch Tanpa Server yang akan digunakan saat mengindeks data ke koleksi Tanpa Server Anda OpenSearch .

  Konektivitas VPC tujuan

  Jika koleksi OpenSearch Tanpa Server Anda ada di VPC pribadi, gunakan bagian ini untuk menentukan VPC tersebut. Tentukan juga subnet dan subgrup yang Anda inginkan Amazon Data Firehose untuk digunakan saat mengirimkan data ke koleksi Tanpa Server Anda. OpenSearch

  Important

  Saat Anda menentukan subnet untuk mengirimkan data ke tujuan dalam VPC pribadi, pastikan Anda memiliki cukup banyak alamat IP gratis di subnet yang dipilih. Jika tidak ada alamat IP gratis yang tersedia di subnet tertentu, Firehose tidak dapat membuat atau ENIs menambahkan pengiriman data di VPC pribadi, dan pengiriman akan terdegradasi atau gagal.

Coba lagi durasi

Durasi waktu Firehose untuk mencoba lagi jika permintaan indeks ke Tanpa Server gagal OpenSearch . Untuk durasi coba lagi, Anda dapat mengatur nilai apa pun antara 0-7200 detik. Durasi coba ulang default adalah 300 detik. Firehose akan mencoba lagi beberapa kali dengan mundur eksponensial hingga durasi percobaan ulang berakhir.

Setelah durasi percobaan ulang berakhir, Firehose mengirimkan data ke Dead Letter Queue (DLQ), bucket kesalahan S3 yang dikonfigurasi. Untuk data yang dikirim ke DLQ, Anda harus mengarahkan ulang data kembali dari bucket kesalahan S3 yang dikonfigurasi ke tujuan Tanpa Server. OpenSearch

Jika Anda ingin memblokir aliran Firehose agar tidak mengirimkan data ke DLQ karena waktu henti atau pemeliharaan kluster OpenSearch Tanpa Server, Anda dapat mengonfigurasi durasi coba lagi ke nilai yang lebih tinggi dalam hitungan detik. Anda dapat meningkatkan nilai durasi coba lagi di atas menjadi 7200 detik dengan menghubungi dukungan.AWS

Petunjuk penyangga

Amazon Data Firehose melakukan buffer pada data masuk sebelum mengirimkannya ke tujuan yang ditentukan. Ukuran buffer yang direkomendasikan untuk tujuan bervariasi dari tiap-tiap penyedia layanan.

Konfigurasikan pengaturan tujuan untuk HTTP Endpoint

Bagian ini menjelaskan opsi untuk menggunakan Titik akhir HTTP untuk tujuan Anda.

> **Important**
>
> Jika Anda memilih titik akhir HTTP sebagai tujuan Anda, tinjau dan ikuti petunjuk di Memahami permintaan pengiriman titik akhir HTTP dan spesifikasi respons.

- Berikan nilai untuk bidang berikut:

Nama titik akhir HTTP - opsional

Tentukan nama yang ramah pengguna untuk titik akhir HTTP. Misalnya, `My HTTP Endpoint Destination`.

URL titik akhir HTTP

Tentukan URL untuk titik akhir HTTP dalam format berikut: `https://xyz.httpendpoint.com`. URL harus berupa URL HTTPS.

Autentikasi

Anda dapat memilih untuk memasukkan kunci akses secara langsung atau mengambil rahasia dari AWS Secrets Manager untuk mengakses titik akhir HTTP.

- (Opsional) Kunci akses

  Hubungi pemilik endpoint jika Anda perlu mendapatkan kunci akses untuk mengaktifkan pengiriman data ke titik akhir mereka dari Firehose.

- Rahasia

  Pilih rahasia dari AWS Secrets Manager yang berisi kunci akses untuk titik akhir HTTP. Jika Anda tidak melihat rahasia Anda di daftar drop-down, buat satu AWS Secrets Manager untuk kunci akses. Untuk informasi selengkapnya, lihat Otentikasi dengan AWS Secrets Manager di Amazon Data Firehose.

Pengkodean konten

Amazon Data Firehose menggunakan pengkodean konten untuk mengompres isi permintaan sebelum mengirimnya ke tujuan. Pilih GZIP atau Dinonaktifkan untuk pengkodean enable/disable konten permintaan Anda.

Coba lagi durasi

Tentukan berapa lama Amazon Data Firehose mencoba mengirim data ke titik akhir HTTP yang dipilih.

Setelah mengirim data, Amazon Data Firehose akan menunggu pengakuan dari titik akhir HTTP terlebih dahulu. Jika terjadi kesalahan atau pengakuan tidak diberikan dalam periode batas waktu pengakuan, Amazon Data Firehose akan memulai penghitung durasi percobaan ulang. Kinesis Data Firehose terus mencoba kembali sampai durasi coba lagi berakhir.

Setelah itu, Amazon Data Firehose menganggap pengiriman data gagal dan mencadangkan data ke bucket Amazon S3 Anda.

Setiap kali Amazon Data Firehose mengirimkan data ke titik akhir HTTP (baik upaya awal atau percobaan ulang), itu memulai ulang penghitung batas waktu pengakuan dan menunggu pengakuan dari titik akhir HTTP.

Bahkan jika durasi percobaan ulang berakhir, Amazon Data Firehose masih menunggu pengakuan sampai menerimanya atau periode batas waktu pengakuan tercapai. Jika waktu pengakuan habis, Amazon Data Firehose menentukan apakah ada waktu tersisa di penghitung coba lagi. Jika ada waktu yang tersisa, Kinesis Data Firehose akan mencoba lagi dan mengulangi logika sampai menerima pengakuan atau dapat menentukan bahwa waktu coba lagi telah berakhir.

Jika Anda tidak ingin Amazon Data Firehose mencoba lagi mengirim data, tetapkan nilai ini ke 0.

Parameter - opsional

Amazon Data Firehose menyertakan pasangan kunci-nilai ini di setiap panggilan HTTP. Parameter ini dapat membantu Anda mengidentifikasi dan mengatur tujuan.

Petunjuk penyangga

Amazon Data Firehose melakukan buffer pada data masuk sebelum mengirimkannya ke tujuan yang ditentukan. Ukuran buffer yang direkomendasikan untuk tujuan bervariasi dari tiap-tiap penyedia layanan.

> **Important**
>
> Untuk tujuan titik akhir HTTP, jika Anda melihat 413 kode respons dari titik akhir tujuan di CloudWatch Log, turunkan ukuran petunjuk buffering pada aliran Firehose Anda dan coba lagi.

Konfigurasikan pengaturan tujuan untuk Datadog

Bagian ini menjelaskan opsi untuk menggunakan Datadog untuk tujuan Anda. Untuk informasi selengkapnya tentang Datadog, lihat https://docs.datadoghq.com/integrations/amazon_web_services/.

- Berikan nilai untuk bidang berikut.

  URL titik akhir HTTP

  Pilih tempat Anda ingin mengirim data dari salah satu opsi berikut di menu tarik-turun.

  - Log Datadog - US1
  - Log Datadog - US3
  - Log Datadog - US5
  - Log Datadog - AP1
  - Log datadog - UE
  - Log datadog - GOV
  - Metrik Datadog - AS
  - Metrik Datadog - US5
  - Metrik Datadog - AP1
  - Metrik Datadog - UE
  - Konfigurasi Datadog - US1
  - Konfigurasi Datadog - US3
  - Konfigurasi Datadog - US5
  - Konfigurasi Datadog - AP1
  - Konfigurasi Datadog - EU
  - Konfigurasi Datadog - US GOV

  Autentikasi

  Anda dapat memilih untuk memasukkan kunci API secara langsung atau mengambil rahasia dari AWS Secrets Manager untuk mengakses Datadog.

  - Kunci API

    Hubungi Datadog untuk mendapatkan kunci API yang Anda perlukan untuk mengaktifkan pengiriman data ke titik akhir ini dari Firehose.

  - Rahasia

    Pilih rahasia dari AWS Secrets Manager yang berisi kunci API untuk Datadog. Jika Anda tidak melihat rahasia Anda di daftar drop-down, buat satu di AWS Secrets Manager. Untuk

informasi selengkapnya, lihat Otentikasi dengan AWS Secrets Manager di Amazon Data Firehose.

Pengkodean konten

Amazon Data Firehose menggunakan pengkodean konten untuk mengompres isi permintaan sebelum mengirimnya ke tujuan. Pilih GZIP atau Dinonaktifkan untuk pengkodean enable/disable konten permintaan Anda.

Coba lagi durasi

Tentukan berapa lama Amazon Data Firehose mencoba mengirim data ke titik akhir HTTP yang dipilih.

Setelah mengirim data, Amazon Data Firehose akan menunggu pengakuan dari titik akhir HTTP terlebih dahulu. Jika terjadi kesalahan atau pengakuan tidak diberikan dalam periode batas waktu pengakuan, Amazon Data Firehose akan memulai penghitung durasi percobaan ulang. Kinesis Data Firehose terus mencoba kembali sampai durasi coba lagi berakhir. Setelah itu, Amazon Data Firehose menganggap pengiriman data gagal dan mencadangkan data ke bucket Amazon S3 Anda.

Setiap kali Amazon Data Firehose mengirimkan data ke titik akhir HTTP (baik upaya awal atau percobaan ulang), itu memulai ulang penghitung batas waktu pengakuan dan menunggu pengakuan dari titik akhir HTTP.

Bahkan jika durasi percobaan ulang berakhir, Amazon Data Firehose masih menunggu pengakuan sampai menerimanya atau periode batas waktu pengakuan tercapai. Jika waktu pengakuan habis, Amazon Data Firehose menentukan apakah ada waktu tersisa di penghitung coba lagi. Jika ada waktu yang tersisa, Kinesis Data Firehose akan mencoba lagi dan mengulangi logika sampai menerima pengakuan atau dapat menentukan bahwa waktu coba lagi telah berakhir.

Jika Anda tidak ingin Amazon Data Firehose mencoba lagi mengirim data, tetapkan nilai ini ke 0.

Parameter - opsional

Amazon Data Firehose menyertakan pasangan kunci-nilai ini di setiap panggilan HTTP. Parameter ini dapat membantu Anda mengidentifikasi dan mengatur tujuan.

Petunjuk penyangga

Amazon Data Firehose melakukan buffer pada data masuk sebelum mengirimkannya ke tujuan yang ditentukan. Ukuran buffer yang direkomendasikan untuk tujuan bervariasi dari tiap-tiap penyedia layanan.

Konfigurasikan pengaturan tujuan untuk Honeycomb

Bagian ini menjelaskan opsi untuk menggunakan Honeycomb untuk tujuan Anda. Untuk informasi lebih lanjut tentang Honeycomb, lihat https://docs.honeycomb. io/getting-data-in/metrics/aws - cloudwatch-metrics/.

- Berikan nilai untuk bidang berikut:

  Titik akhir Kinesis Honeycomb

  Tentukan URL untuk titik akhir HTTP dalam format berikut: https://api.honeycom b.io/1/ kinesis_events/ {{dataset}}

  Autentikasi

  Anda dapat memilih untuk memasukkan kunci API secara langsung atau mengambil rahasia dari AWS Secrets Manager untuk mengakses Honeycomb.

  - Kunci API

    Hubungi Honeycomb untuk mendapatkan kunci API yang Anda perlukan untuk mengaktifkan pengiriman data ke titik akhir ini dari Firehose.

  - Rahasia

    Pilih rahasia dari AWS Secrets Manager yang berisi kunci API untuk Honeycomb. Jika Anda tidak melihat rahasia Anda di daftar drop-down, buat satu di AWS Secrets Manager. Untuk informasi selengkapnya, lihat Otentikasi dengan AWS Secrets Manager di Amazon Data Firehose.

  Pengkodean konten

  Amazon Data Firehose menggunakan pengkodean konten untuk mengompres isi permintaan sebelum mengirimnya ke tujuan. Pilih GZIP untuk mengaktifkan pengkodean konten permintaan Anda. Ini adalah opsi yang disarankan untuk tujuan Honeycomb.

Coba lagi durasi

Tentukan berapa lama Amazon Data Firehose mencoba mengirim data ke titik akhir HTTP yang dipilih.

Setelah mengirim data, Amazon Data Firehose akan menunggu pengakuan dari titik akhir HTTP terlebih dahulu. Jika terjadi kesalahan atau pengakuan tidak diberikan dalam periode batas waktu pengakuan, Amazon Data Firehose akan memulai penghitung durasi percobaan ulang. Kinesis Data Firehose terus mencoba kembali sampai durasi coba lagi berakhir. Setelah itu, Amazon Data Firehose menganggap pengiriman data gagal dan mencadangkan data ke bucket Amazon S3 Anda.

Setiap kali Amazon Data Firehose mengirimkan data ke titik akhir HTTP (baik upaya awal atau percobaan ulang), itu memulai ulang penghitung batas waktu pengakuan dan menunggu pengakuan dari titik akhir HTTP.

Bahkan jika durasi percobaan ulang berakhir, Amazon Data Firehose masih menunggu pengakuan sampai menerimanya atau periode batas waktu pengakuan tercapai. Jika waktu pengakuan habis, Amazon Data Firehose menentukan apakah ada waktu tersisa di penghitung coba lagi. Jika ada waktu yang tersisa, Kinesis Data Firehose akan mencoba lagi dan mengulangi logika sampai menerima pengakuan atau dapat menentukan bahwa waktu coba lagi telah berakhir.

Jika Anda tidak ingin Amazon Data Firehose mencoba lagi mengirim data, tetapkan nilai ini ke 0.

Parameter - opsional

Amazon Data Firehose menyertakan pasangan kunci-nilai ini di setiap panggilan HTTP. Parameter ini dapat membantu Anda mengidentifikasi dan mengatur tujuan.

Petunjuk penyangga

Amazon Data Firehose melakukan buffer pada data masuk sebelum mengirimkannya ke tujuan yang ditentukan. Ukuran buffer yang direkomendasikan untuk tujuan bervariasi dari tiap-tiap penyedia layanan.

Konfigurasikan pengaturan tujuan untuk Coralogix

Bagian ini menjelaskan opsi untuk menggunakan Coralogix untuk tujuan Anda. Untuk informasi selengkapnya tentang Coralogix, lihat Memulai Coralogix.

- Berikan nilai untuk bidang berikut:

  URL titik akhir HTTP

  Pilih URL titik akhir HTTP dari pilihan berikut di menu tarik-turun:

  - Coralogix - US
  - Coralogix - SINGAPORE
  - Coralogix - IRELAND
  - Coralogix - INDIA
  - Coralogix - STOCKHOLM

  Autentikasi

  Anda dapat memilih untuk memasukkan kunci pribadi secara langsung atau mengambil rahasia dari AWS Secrets Manager untuk mengakses Coralogix.

  - Kunci pribadi

    Hubungi Coralogix untuk mendapatkan kunci pribadi yang Anda perlukan untuk mengaktifkan pengiriman data ke titik akhir ini dari Firehose.

  - Rahasia

    Pilih rahasia dari AWS Secrets Manager yang berisi kunci pribadi untuk Coralogix. Jika Anda tidak melihat rahasia Anda di daftar drop-down, buat satu di AWS Secrets Manager. Untuk informasi selengkapnya, lihat Otentikasi dengan AWS Secrets Manager di Amazon Data Firehose.

  Pengkodean konten

  Amazon Data Firehose menggunakan pengkodean konten untuk mengompres isi permintaan sebelum mengirimnya ke tujuan. Pilih GZIP untuk mengaktifkan pengkodean konten permintaan Anda. Ini adalah opsi yang direkomendasikan untuk tujuan Coralogix.

Coba lagi durasi

Tentukan berapa lama Amazon Data Firehose mencoba mengirim data ke titik akhir HTTP yang dipilih.

Setelah mengirim data, Amazon Data Firehose akan menunggu pengakuan dari titik akhir HTTP terlebih dahulu. Jika terjadi kesalahan atau pengakuan tidak diberikan dalam periode batas waktu pengakuan, Amazon Data Firehose akan memulai penghitung durasi percobaan ulang. Kinesis Data Firehose terus mencoba kembali sampai durasi coba lagi berakhir. Setelah itu, Amazon Data Firehose menganggap pengiriman data gagal dan mencadangkan data ke bucket Amazon S3 Anda.

Setiap kali Amazon Data Firehose mengirimkan data ke titik akhir HTTP (baik upaya awal atau percobaan ulang), itu memulai ulang penghitung batas waktu pengakuan dan menunggu pengakuan dari titik akhir HTTP.

Bahkan jika durasi percobaan ulang berakhir, Amazon Data Firehose masih menunggu pengakuan sampai menerimanya atau periode batas waktu pengakuan tercapai. Jika waktu pengakuan habis, Amazon Data Firehose menentukan apakah ada waktu tersisa di penghitung coba lagi. Jika ada waktu yang tersisa, Kinesis Data Firehose akan mencoba lagi dan mengulangi logika sampai menerima pengakuan atau dapat menentukan bahwa waktu coba lagi telah berakhir.

Jika Anda tidak ingin Amazon Data Firehose mencoba lagi mengirim data, tetapkan nilai ini ke 0.

Parameter - opsional

Amazon Data Firehose menyertakan pasangan kunci-nilai ini di setiap panggilan HTTP. Parameter ini dapat membantu Anda mengidentifikasi dan mengatur tujuan.

- applicationName: lingkungan tempat Anda menjalankan Data Firehose
- subsystemName: nama integrasi Data Firehose
- computerName: nama aliran Firehose yang sedang digunakan

Petunjuk penyangga

Amazon Data Firehose melakukan buffer pada data masuk sebelum mengirimkannya ke tujuan yang ditentukan. Ukuran buffer yang disarankan untuk tujuan bervariasi berdasarkan penyedia layanan.

Konfigurasikan setelan tujuan untuk Dynatrace

Bagian ini menjelaskan opsi untuk menggunakan Dynatrace untuk tujuan Anda. Untuk informasi selengkapnya, lihat https://www.dynatrace.com/support/help/technology-support/cloud-platforms/amazon-web-services/integrations/cloudwatch-metric-streams/.

- Pilih opsi untuk menggunakan Dynatrace sebagai tujuan aliran Firehose Anda.

  Jenis konsumsi

  Pilih apakah Anda ingin mengirimkan Metrik atau Log (default) di Dynatrace untuk analisis dan pemrosesan lebih lanjut.

  URL titik akhir HTTP

  Pilih URL titik akhir HTTP (Dynatrace US, Dynatrace EU, atau Dynatrace Global) dari menu tarik-turun.

  Autentikasi

  Anda dapat memilih untuk memasukkan token API secara langsung atau mengambil rahasia dari AWS Secrets Manager untuk mengakses Dynatrace.

  - Token API

    Buat token API Dynatrace yang Anda perlukan untuk mengaktifkan pengiriman data ke titik akhir ini dari Firehose. Untuk informasi selengkapnya, lihat Dynatrace API - Token dan otentikasi.

  - Rahasia

    Pilih rahasia dari AWS Secrets Manager yang berisi token API untuk Dynatrace. Jika Anda tidak melihat rahasia Anda di daftar drop-down, buat satu di AWS Secrets Manager. Untuk informasi selengkapnya, lihat Otentikasi dengan AWS Secrets Manager di Amazon Data Firehose.

  URL API

  Berikan URL API lingkungan Dynatrace Anda.

Pengkodean konten

Pilih apakah Anda ingin mengaktifkan pengkodean konten untuk mengompres isi permintaan. Amazon Data Firehose menggunakan pengkodean konten untuk mengompres isi permintaan sebelum mengirimnya ke tujuan. Saat diaktifkan, konten yang dikompresi dalam format GZIP.

Coba lagi durasi

Tentukan berapa lama Firehose mencoba mengirim data ke titik akhir HTTP yang dipilih.

Setelah mengirim data, akan menunggu pengakuan dari titik akhir HTTP terlebih dahulu. Jika terjadi kesalahan atau pengakuan tidak diberikan dalam periode batas waktu pengakuan, Firehose akan memulai penghitung durasi percobaan ulang. Kinesis Data Firehose terus mencoba kembali sampai durasi coba lagi berakhir. Setelah itu, Firehose menganggap pengiriman data gagal dan mencadangkan data ke bucket Amazon S3 Anda.

Setiap kali Firehose mengirimkan data ke titik akhir HTTP, baik selama upaya awal atau setelah mencoba lagi, Firehose memulai ulang penghitung batas waktu pengakuan dan menunggu pengakuan dari titik akhir HTTP.

Bahkan jika durasi percobaan ulang berakhir, Firehose masih menunggu pengakuan sampai menerimanya atau periode batas waktu pengakuan tercapai. Jika waktu pengakuan habis, Firehose menentukan apakah ada waktu tersisa di penghitung coba lagi. Jika ada waktu yang tersisa, Kinesis Data Firehose akan mencoba lagi dan mengulangi logika sampai menerima pengakuan atau dapat menentukan bahwa waktu coba lagi telah berakhir.

Jika Anda tidak ingin Firehose mencoba mengirim data lagi, tetapkan nilai ini ke 0.

Parameter - opsional

Amazon Data Firehose menyertakan pasangan kunci-nilai ini di setiap panggilan HTTP. Parameter ini dapat membantu Anda mengidentifikasi dan mengatur tujuan.

Petunjuk penyangga

Amazon Data Firehose melakukan buffer pada data masuk sebelum mengirimkannya ke tujuan yang ditentukan. Petunjuk buffer mencakup ukuran buffer dan interval untuk aliran Anda. Ukuran buffer yang disarankan untuk tujuan bervariasi sesuai dengan penyedia layanan.

Konfigurasikan setelan tujuan untuk LogicMonitor

Bagian ini menjelaskan opsi untuk digunakan LogicMonitoruntuk tujuan Anda. Untuk informasi selengkapnya, lihat https://www.logicmonitor.com.

- Berikan nilai untuk bidang berikut:

  URL titik akhir HTTP

  Tentukan URL untuk titik akhir HTTP dalam format berikut.

  ```
  https://ACCOUNT.logicmonitor.com
  ```

  Autentikasi

  Anda dapat memilih untuk memasukkan kunci API secara langsung atau mengambil rahasia dari AWS Secrets Manager untuk mengakses LogicMonitor.

  - Kunci API

    Hubungi LogicMonitor untuk mendapatkan kunci API yang Anda perlukan untuk mengaktifkan pengiriman data ke titik akhir ini dari Firehose.

  - Rahasia

    Pilih rahasia dari AWS Secrets Manager yang berisi kunci API untuk LogicMonitor. Jika Anda tidak melihat rahasia Anda di daftar drop-down, buat satu di AWS Secrets Manager. Untuk informasi selengkapnya, lihat Otentikasi dengan AWS Secrets Manager di Amazon Data Firehose.

  Pengkodean konten

  Amazon Data Firehose menggunakan pengkodean konten untuk mengompres isi permintaan sebelum mengirimnya ke tujuan. Pilih GZIP atau Dinonaktifkan untuk pengkodean enable/disable konten permintaan Anda.

  Coba lagi durasi

  Tentukan berapa lama Amazon Data Firehose mencoba mengirim data ke titik akhir HTTP yang dipilih.

Setelah mengirim data, Amazon Data Firehose akan menunggu pengakuan dari titik akhir HTTP terlebih dahulu. Jika terjadi kesalahan atau pengakuan tidak diberikan dalam periode batas waktu pengakuan, Amazon Data Firehose akan memulai penghitung durasi percobaan ulang. Kinesis Data Firehose terus mencoba kembali sampai durasi coba lagi berakhir. Setelah itu, Amazon Data Firehose menganggap pengiriman data gagal dan mencadangkan data ke bucket Amazon S3 Anda.

Setiap kali Amazon Data Firehose mengirimkan data ke titik akhir HTTP (baik upaya awal atau percobaan ulang), itu memulai ulang penghitung batas waktu pengakuan dan menunggu pengakuan dari titik akhir HTTP.

Bahkan jika durasi percobaan ulang berakhir, Amazon Data Firehose masih menunggu pengakuan sampai menerimanya atau periode batas waktu pengakuan tercapai. Jika waktu pengakuan habis, Amazon Data Firehose menentukan apakah ada waktu tersisa di penghitung coba lagi. Jika ada waktu yang tersisa, Kinesis Data Firehose akan mencoba lagi dan mengulangi logika sampai menerima pengakuan atau dapat menentukan bahwa waktu coba lagi telah berakhir.

Jika Anda tidak ingin Amazon Data Firehose mencoba lagi mengirim data, tetapkan nilai ini ke 0.

Parameter - opsional

Amazon Data Firehose menyertakan pasangan kunci-nilai ini di setiap panggilan HTTP. Parameter ini dapat membantu Anda mengidentifikasi dan mengatur tujuan.

Petunjuk penyangga

Amazon Data Firehose melakukan buffer pada data masuk sebelum mengirimkannya ke tujuan yang ditentukan. Ukuran buffer yang direkomendasikan untuk tujuan bervariasi dari tiap-tiap penyedia layanan.

Konfigurasikan pengaturan tujuan untuk Logz.io

Bagian ini menjelaskan opsi untuk menggunakan Logz.io untuk tujuan Anda. Untuk informasi lebih lanjut, lihat https://logz.io/.

> **Note**
>
> Di wilayah Eropa (Milan), Logz.io tidak didukung sebagai tujuan Amazon Data Firehose.

- Berikan nilai untuk bidang berikut:

  URL titik akhir HTTP

  Tentukan URL untuk titik akhir HTTP dalam format berikut. URL harus berupa HTTPS URL.

  ```
  https://listener-aws-metrics-stream-<region>.logz.io/
  ```

  Sebagai contoh

  ```
  https://listener-aws-metrics-stream-us.logz.io/
  ```

  Autentikasi

  Anda dapat memilih untuk memasukkan token pengiriman secara langsung atau mengambil rahasia dari AWS Secrets Manager untuk mengakses Logz.io.

  - Token pengiriman

    Hubungi Logz.io untuk mendapatkan token pengiriman yang Anda perlukan untuk mengaktifkan pengiriman data ke titik akhir ini dari Firehose.

  - Rahasia

    Pilih rahasia dari AWS Secrets Manager yang berisi token pengiriman untuk Logz.io. Jika Anda tidak melihat rahasia Anda di daftar drop-down, buat satu di AWS Secrets Manager. Untuk informasi selengkapnya, lihat Otentikasi dengan AWS Secrets Manager di Amazon Data Firehose.

  Coba lagi durasi

  Tentukan berapa lama Amazon Data Firehose mencoba mengirim data ke Logz.io.

  Setelah mengirim data, Amazon Data Firehose akan menunggu pengakuan dari titik akhir HTTP terlebih dahulu. Jika terjadi kesalahan atau pengakuan tidak diberikan dalam periode batas waktu pengakuan, Amazon Data Firehose akan memulai penghitung durasi percobaan

ulang. Kinesis Data Firehose terus mencoba kembali sampai durasi coba lagi berakhir. Setelah itu, Amazon Data Firehose menganggap pengiriman data gagal dan mencadangkan data ke bucket Amazon S3 Anda.

Setiap kali Amazon Data Firehose mengirimkan data ke titik akhir HTTP (baik upaya awal atau percobaan ulang), itu memulai ulang penghitung batas waktu pengakuan dan menunggu pengakuan dari titik akhir HTTP.

Bahkan jika durasi percobaan ulang berakhir, Amazon Data Firehose masih menunggu pengakuan sampai menerimanya atau periode batas waktu pengakuan tercapai. Jika waktu pengakuan habis, Amazon Data Firehose menentukan apakah ada waktu tersisa di penghitung coba lagi. Jika ada waktu yang tersisa, Kinesis Data Firehose akan mencoba lagi dan mengulangi logika sampai menerima pengakuan atau dapat menentukan bahwa waktu coba lagi telah berakhir.

Jika Anda tidak ingin Amazon Data Firehose mencoba lagi mengirim data, tetapkan nilai ini ke 0.

Parameter - opsional

Amazon Data Firehose menyertakan pasangan kunci-nilai ini di setiap panggilan HTTP. Parameter ini dapat membantu Anda mengidentifikasi dan mengatur tujuan.

Petunjuk penyangga

Amazon Data Firehose melakukan buffer pada data masuk sebelum mengirimkannya ke tujuan yang ditentukan. Ukuran buffer yang direkomendasikan untuk tujuan bervariasi dari tiap-tiap penyedia layanan.

Konfigurasikan pengaturan tujuan untuk MongoDB Atlas

Bagian ini menjelaskan opsi untuk menggunakan MongoDB Atlas untuk tujuan Anda. Untuk informasi selengkapnya, lihat MongoDB Atlas di Amazon Web Services.

- Berikan nilai untuk bidang berikut:

  URL API Gateway

  Tentukan URL untuk titik akhir HTTP dalam format berikut.

```
https://xxxxx.execute-api.region.amazonaws.com/stage
```

URL harus berupa HTTPS URL.

Autentikasi

Anda dapat memilih untuk memasukkan kunci API secara langsung atau mengambil rahasia dari AWS Secrets Manager untuk mengakses MongoDB Atlas.

- Kunci API

  Ikuti petunjuk di MongoDB Atlas di Amazon Web Services untuk mendapatkan `APIKeyValue` yang Anda perlukan untuk mengaktifkan pengiriman data ke titik akhir ini dari Firehose.

- Rahasia

  Pilih rahasia dari AWS Secrets Manager yang berisi Kunci API untuk API Gateway yang didukung oleh Lambda yang berinteraksi dengan MongoDB Atlas. Jika Anda tidak melihat rahasia Anda di daftar drop-down, buat satu di AWS Secrets Manager. Untuk informasi selengkapnya, lihat Otentikasi dengan AWS Secrets Manager di Amazon Data Firehose.

Pengkodean konten

Amazon Data Firehose menggunakan pengkodean konten untuk mengompres isi permintaan sebelum mengirimnya ke tujuan. Pilih GZIP atau Dinonaktifkan untuk pengkodean enable/disable konten permintaan Anda.

Coba lagi durasi

Tentukan berapa lama Amazon Data Firehose mencoba mengirim data ke penyedia pihak ketiga yang dipilih.

Setelah mengirim data, Amazon Data Firehose akan menunggu pengakuan dari titik akhir HTTP terlebih dahulu. Jika terjadi kesalahan atau pengakuan tidak diberikan dalam periode batas waktu pengakuan, Amazon Data Firehose akan memulai penghitung durasi percobaan ulang. Kinesis Data Firehose terus mencoba kembali sampai durasi coba lagi berakhir. Setelah itu, Amazon Data Firehose menganggap pengiriman data gagal dan mencadangkan data ke bucket Amazon S3 Anda.

Setiap kali Amazon Data Firehose mengirimkan data ke titik akhir HTTP (baik upaya awal atau percobaan ulang), itu memulai ulang penghitung batas waktu pengakuan dan menunggu pengakuan dari titik akhir HTTP.

Bahkan jika durasi percobaan ulang berakhir, Amazon Data Firehose masih menunggu pengakuan sampai menerimanya atau periode batas waktu pengakuan tercapai. Jika waktu pengakuan habis, Amazon Data Firehose menentukan apakah ada waktu tersisa di penghitung coba lagi. Jika ada waktu yang tersisa, Kinesis Data Firehose akan mencoba lagi dan mengulangi logika sampai menerima pengakuan atau dapat menentukan bahwa waktu coba lagi telah berakhir.

Jika Anda tidak ingin Amazon Data Firehose mencoba lagi mengirim data, tetapkan nilai ini ke 0.

Petunjuk penyangga

Amazon Data Firehose melakukan buffer pada data masuk sebelum mengirimkannya ke tujuan yang ditentukan. Ukuran buffer yang direkomendasikan untuk tujuan bervariasi dari tiap-tiap penyedia layanan.

Parameter - opsional

Amazon Data Firehose menyertakan pasangan kunci-nilai ini di setiap panggilan HTTP. Parameter ini dapat membantu Anda mengidentifikasi dan mengatur tujuan.

Konfigurasikan pengaturan tujuan untuk New Relic

Bagian ini menjelaskan opsi untuk menggunakan New Relic untuk tujuan Anda. Untuk informasi selengkapnya, lihat https://newrelic.com.

- Berikan nilai untuk bidang berikut:

  URL titik akhir HTTP

  Pilih URL titik akhir HTTP dari opsi berikut dalam daftar drop-down.

  - Log Relik Baru - AS
  - Metrik Relik Baru - AS
  - Metrik Relik Baru - UE

Autentikasi

Anda dapat memilih untuk memasukkan kunci API secara langsung atau mengambil rahasia dari AWS Secrets Manager untuk mengakses New Relic.

- Kunci API

  Masukkan Kunci Lisensi Anda, yang merupakan string heksadesimal 40 karakter, dari pengaturan Akun New Relic One Anda. Anda memerlukan kunci API ini untuk mengaktifkan pengiriman data ke titik akhir ini dari Firehose.

- Rahasia

  Pilih rahasia dari AWS Secrets Manager yang berisi kunci API untuk New Relic. Jika Anda tidak melihat rahasia Anda di daftar drop-down, buat satu di AWS Secrets Manager. Untuk informasi selengkapnya, lihat Otentikasi dengan AWS Secrets Manager di Amazon Data Firehose.

Pengkodean konten

Amazon Data Firehose menggunakan pengkodean konten untuk mengompres isi permintaan sebelum mengirimnya ke tujuan. Pilih GZIP atau Dinonaktifkan untuk pengkodean enable/disable konten permintaan Anda.

Coba lagi durasi

Tentukan berapa lama Amazon Data Firehose mencoba mengirim data ke titik akhir HTTP Relic Baru.

Setelah mengirim data, Amazon Data Firehose akan menunggu pengakuan dari titik akhir HTTP terlebih dahulu. Jika terjadi kesalahan atau pengakuan tidak diberikan dalam periode batas waktu pengakuan, Amazon Data Firehose akan memulai penghitung durasi percobaan ulang. Kinesis Data Firehose terus mencoba kembali sampai durasi coba lagi berakhir. Setelah itu, Amazon Data Firehose menganggap pengiriman data gagal dan mencadangkan data ke bucket Amazon S3 Anda.

Setiap kali Amazon Data Firehose mengirimkan data ke titik akhir HTTP (baik upaya awal atau percobaan ulang), itu memulai ulang penghitung batas waktu pengakuan dan menunggu pengakuan dari titik akhir HTTP.

Bahkan jika durasi percobaan ulang berakhir, Amazon Data Firehose masih menunggu pengakuan sampai menerimanya atau periode batas waktu pengakuan tercapai. Jika

waktu pengakuan habis, Amazon Data Firehose menentukan apakah ada waktu tersisa di penghitung coba lagi. Jika ada waktu yang tersisa, Kinesis Data Firehose akan mencoba lagi dan mengulangi logika sampai menerima pengakuan atau dapat menentukan bahwa waktu coba lagi telah berakhir.

Jika Anda tidak ingin Amazon Data Firehose mencoba lagi mengirim data, tetapkan nilai ini ke 0.

Parameter - opsional

Amazon Data Firehose menyertakan pasangan kunci-nilai ini di setiap panggilan HTTP. Parameter ini dapat membantu Anda mengidentifikasi dan mengatur tujuan.

Petunjuk penyangga

Amazon Data Firehose melakukan buffer pada data masuk sebelum mengirimkannya ke tujuan yang ditentukan. Ukuran buffer yang direkomendasikan untuk tujuan bervariasi dari tiap-tiap penyedia layanan.

Konfigurasikan pengaturan tujuan untuk Snowflake

Bagian ini menjelaskan opsi untuk menggunakan Snowflake untuk tujuan Anda.

> **Note**
>
> Integrasi Firehose dengan Snowflake tersedia di AS Timur (Virginia N.), AS Barat (Oregon), Eropa (Irlandia), AS Timur (Ohio), Asia Pasifik (Tokyo), Eropa (Frankfurt), Asia Pasifik (Singapura), Asia Pasifik (Seoul), dan Asia Pasifik (Sydney), Asia Pasifik (Mumbai), Eropa (London), Amerika Selatan (Sao Paulo), Kanada (Tengah), Eropa (Paris), Asia Pasifik (Osaka), Eropa (Stockholm), Asia Pasifik (Jakarta). Wilayah AWS

Pengaturan koneksi

- Berikan nilai untuk bidang berikut:

  URL akun Snowflake

  Tentukan URL akun Snowflake. Sebagai contoh: `xy12345.us-east-1.aws.snowflakecomputing.com`. Lihat dokumentasi Snowflake tentang cara

menentukan URL akun Anda. Perhatikan bahwa Anda tidak harus menentukan nomor port, sedangkan protokol (https://) adalah opsional.

Autentikasi

Anda dapat memilih untuk memasukkan userlogin, kunci pribadi, dan frasa sandi secara manual atau mengambil rahasia dari untuk mengakses Snowflake. AWS Secrets Manager

- Login pengguna

  Tentukan pengguna Snowflake yang akan digunakan untuk memuat data. Pastikan pengguna memiliki akses untuk memasukkan data ke dalam tabel Snowflake.

- Kunci pribadi

  Tentukan kunci pribadi untuk otentikasi dengan Snowflake dalam format. PKCS8 Selain itu, jangan sertakan header dan footer PEM sebagai bagian dari kunci pribadi. Jika kunci dibagi menjadi beberapa baris, hapus jeda baris. Berikut ini adalah contoh dari apa kunci pribadi Anda harus terlihat seperti.

  ```
  -----BEGIN PRIVATE KEY-----
  KEY_CONTENT
  -----END PRIVATE KEY-----
  ```

  Hapus ruang KEY_CONTENT dan berikan itu ke Firehose. Tidak ada header/footer atau karakter baris baru yang diperlukan.

- Frasa sandi

  Tentukan frasa sandi untuk mendekripsi kunci privat yang dienkripsi. Anda dapat membiarkan bidang ini kosong jika kunci privat tidak dienkripsi. Untuk selengkapnya, lihat Menggunakan Otentikasi Pasangan Kunci & Rotasi Kunci.

- Rahasia

  Pilih rahasia dari AWS Secrets Manager yang berisi kredensil untuk Snowflake. Jika Anda tidak melihat rahasia Anda di daftar drop-down, buat satu di AWS Secrets Manager. Untuk informasi selengkapnya, lihat Otentikasi dengan AWS Secrets Manager di Amazon Data Firehose.

Konfigurasi peran

Gunakan peran Snowflake default - Jika opsi ini dipilih, Firehose tidak akan meneruskan peran apa pun ke Snowflake. Peran default diasumsikan untuk memuat data. Pastikan peran default memiliki izin untuk memasukkan data ke tabel Snowflake.

Gunakan peran Snowflake khusus - Masukkan peran Snowflake non-default yang akan diasumsikan oleh Firehose saat memuat data ke tabel Snowflake.

Konektivitas Snowflake

Pilihannya adalah Privat atau Publik.

ID VPCE Privat (opsional)

ID VPCE untuk Firehose untuk terhubung secara privat dengan Snowflake. Format ID adalah com.amazonaws.vpce. [wilayah] .vpce-svc-. *[id]* Untuk informasi lebih lanjut, lihat AWS PrivateLink & Kepingan Salju.

> **Note**
>
> Jika klaster Snowflake Anda mengaktifkan tautan pribadi, gunakan kebijakan jaringan `AwsVpceIds` berbasis untuk mengizinkan data Amazon Data Firehose. Firehose tidak mengharuskan Anda mengonfigurasi kebijakan jaringan berbasis IP di akun Snowflake Anda. Memiliki kebijakan jaringan berbasis IP yang diaktifkan dapat mengganggu konektivitas Firehose. Jika Anda memiliki kasus tepi yang memerlukan kebijakan berbasis IP, hubungi tim Firehose dengan mengirimkan tiket dukungan. Untuk daftar VPCE IDs yang dapat Anda gunakan, lihat. Mengakses Snowflake di VPC

Konfigurasi basis data

- Anda harus menentukan pengaturan berikut agar dapat menggunakan Snowflake sebagai tujuan untuk aliran Firehose Anda.
  - Basis data Snowflake – Semua data di Snowflake disimpan dalam basis data.
  - Skema Snowflake – Setiap basis data terdiri dari satu atau lebih skema, yang merupakan pengelompokan logis objek basis data, seperti tabel dan tampilan

- Tabel Snowflake – Semua data di Snowflake disimpan dalam tabel basis data, terstruktur secara logis sebagai kumpulan kolom dan baris.

Opsi pemuatan data untuk tabel Snowflake Anda

- Gunakan tombol JSON sebagai nama kolom
- Gunakan kolom VARIANT
  - Nama kolom konten - Tentukan nama kolom dalam tabel, di mana data mentah harus dimuat.
  - Nama kolom metadata (opsional) - Tentukan nama kolom dalam tabel, tempat informasi metadata harus dimuat. Saat Anda mengaktifkan bidang ini, Anda akan melihat kolom berikut di tabel Snowflake berdasarkan jenis sumber.

    Untuk Direct PUT sebagai sumber

    ```
    {
    "firehoseDeliveryStreamName" : "streamname",
    "IngestionTime" : "timestamp"
    }
    ```

    Untuk Kinesis Data Stream sebagai sumber

    ```
    {
    "kinesisStreamName" : "streamname",
    "kinesisShardId" : "Id",
    "kinesisPartitionKey" : "key",
    "kinesisSequenceNumber" : "1234",
    "subsequenceNumber" : "2334",
    "IngestionTime" : "timestamp"
    }
    ```

Coba lagi durasi

Durasi waktu (0-7200 detik) untuk Firehose untuk mencoba lagi jika membuka saluran atau pengiriman ke Snowflake gagal karena masalah layanan Snowflake. Firehose mencoba lagi dengan backoff eksponensial hingga durasi coba lagi berakhir. Jika Anda menyetel durasi coba lagi ke 0 (nol) detik, Firehose tidak akan mencoba lagi saat kegagalan Snowflake dan merutekan data ke bucket kesalahan Amazon S3.

Petunjuk penyangga

Amazon Data Firehose melakukan buffer pada data masuk sebelum mengirimkannya ke tujuan yang ditentukan. Ukuran buffer yang direkomendasikan untuk tujuan bervariasi dari tiap-tiap penyedia layanan. Untuk informasi selengkapnya, lihat Konfigurasikan petunjuk buffering.

Konfigurasikan pengaturan tujuan untuk Splunk

Bagian ini menjelaskan opsi untuk menggunakan Splunk untuk tujuan Anda.

> **Note**
>
> Firehose mengirimkan data ke cluster Splunk yang dikonfigurasi dengan Classic Load Balancer atau Application Load Balancer.

- Berikan nilai untuk bidang berikut:

  Titik akhir klaster splunk

  Untuk menentukan titik akhir, lihat Mengonfigurasi Amazon Data Firehose untuk Mengirim Data ke Platform Splunk dalam dokumentasi Splunk.

  Jenis titik akhir splunk

  Pilih `Raw endpoint` dalam banyak kasus. Pilih `Event endpoint` apakah Anda telah memproses data sebelumnya AWS Lambda untuk mengirim data ke indeks yang berbeda berdasarkan jenis acara. Untuk informasi tentang titik akhir yang akan digunakan, lihat Konfigurasikan Amazon Data Firehose untuk mengirim data ke platform Splunk dalam dokumentasi Splunk.

  Autentikasi

  Anda dapat memilih untuk memasukkan token otentikasi secara langsung atau mengambil rahasia dari AWS Secrets Manager untuk mengakses Splunk.

  - Token otentikasi

    Untuk menyiapkan endpoint Splunk yang dapat menerima data dari Amazon Data Firehose, lihat Ikhtisar instalasi dan konfigurasi untuk Add-on Splunk untuk Amazon Data Firehose dalam dokumentasi Splunk. Simpan token yang Anda dapatkan dari Splunk saat Anda mengatur titik akhir untuk aliran Firehose ini dan tambahkan di sini.

- Rahasia

  Pilih rahasia dari AWS Secrets Manager yang berisi token otentikasi untuk Splunk. Jika Anda tidak melihat rahasia Anda di daftar drop-down, buat satu di AWS Secrets Manager. Untuk informasi selengkapnya, lihat Otentikasi dengan AWS Secrets Manager di Amazon Data Firehose.

Batas waktu pengakuan HEC

Tentukan berapa lama Amazon Data Firehose menunggu pengakuan indeks dari Splunk. Jika Splunk tidak mengirimkan pengakuan sebelum batas waktu tercapai, Amazon Data Firehose menganggapnya sebagai kegagalan pengiriman data. Amazon Data Firehose kemudian akan mencoba kembali atau mencadangkan data ke bucket Amazon S3 Anda, tergantung pada nilai durasi percobaan ulang yang Anda tetapkan.

Coba lagi durasi

Tentukan berapa lama Amazon Data Firehose mencoba mengirim data ke Splunk.

Setelah mengirim data, Amazon Data Firehose menunggu pengakuan dari Splunk terlebih dahulu. Jika terjadi kesalahan atau pengakuan tidak diberikan dalam periode batas waktu pengakuan, Amazon Data Firehose akan memulai penghitung durasi percobaan ulang. Kinesis Data Firehose terus mencoba kembali sampai durasi coba lagi berakhir. Setelah itu, Amazon Data Firehose menganggap pengiriman data gagal dan mencadangkan data ke bucket Amazon S3 Anda.

Setiap kali Amazon Data Firehose mengirimkan data ke Splunk (baik upaya awal atau percobaan lagi), itu memulai ulang penghitung batas waktu pengakuan dan menunggu pengakuan dari Splunk.

Bahkan jika durasi percobaan ulang berakhir, Amazon Data Firehose masih menunggu pengakuan sampai menerimanya atau periode batas waktu pengakuan tercapai. Jika waktu pengakuan habis, Amazon Data Firehose menentukan apakah ada waktu tersisa di penghitung coba lagi. Jika ada waktu yang tersisa, Kinesis Data Firehose akan mencoba lagi dan mengulangi logika sampai menerima pengakuan atau dapat menentukan bahwa waktu coba lagi telah berakhir.

Jika Anda tidak ingin Amazon Data Firehose mencoba lagi mengirim data, tetapkan nilai ini ke 0.

Petunjuk penyangga

Amazon Data Firehose melakukan buffer pada data masuk sebelum mengirimkannya ke tujuan yang ditentukan. Ukuran buffer yang disarankan untuk tujuan bervariasi berdasarkan penyedia layanan.

Konfigurasikan pengaturan tujuan untuk Splunk Observability Cloud

Bagian ini menjelaskan opsi untuk menggunakan Splunk Observability Cloud untuk tujuan Anda. Untuk informasi selengkapnya, lihat https://docs.splunk.com/observability/en/gdi/get-data-in/connect/aws/awsconnect-to-aws-using-apiconfig.html# - -api. the-splunk-observability-cloud

- Berikan nilai untuk bidang berikut:

  URL Titik Akhir Cloud Ingest

  Anda dapat menemukan URL Penyerapan Data Real-time Splunk Observability Cloud di Profile > Organizations > Real-time Data Ingest Endpoint di konsol Splunk Observability.

  Autentikasi

  Anda dapat memilih untuk memasukkan token akses secara langsung atau mengambil rahasia dari AWS Secrets Manager untuk mengakses Splunk Observability Cloud.

  - Token Akses

    Salin token akses Observabilitas Splunk Anda dengan cakupan otorisasi PENYERAPAN dari Token Akses di bawah Pengaturan di konsol Observabilitas Splunk.

  - Rahasia

    Pilih rahasia dari AWS Secrets Manager yang berisi token akses untuk Splunk Observability Cloud. Jika Anda tidak melihat rahasia Anda di daftar drop-down, buat satu di AWS Secrets Manager. Untuk informasi selengkapnya, lihat Otentikasi dengan AWS Secrets Manager di Amazon Data Firehose.

  Pengkodean Konten

  Amazon Data Firehose menggunakan pengkodean konten untuk mengompres isi permintaan sebelum mengirimnya ke tujuan. Pilih GZIP atau Dinonaktifkan untuk pengkodean enable/disable konten permintaan Anda.

Coba lagi durasi

Tentukan berapa lama Amazon Data Firehose mencoba mengirim data ke titik akhir HTTP yang dipilih.

Setelah mengirim data, Amazon Data Firehose akan menunggu pengakuan dari titik akhir HTTP terlebih dahulu. Jika terjadi kesalahan atau pengakuan tidak diberikan dalam periode batas waktu pengakuan, Amazon Data Firehose akan memulai penghitung durasi percobaan ulang. Kinesis Data Firehose terus mencoba kembali sampai durasi coba lagi berakhir. Setelah itu, Amazon Data Firehose menganggap pengiriman data gagal dan mencadangkan data ke bucket Amazon S3 Anda.

Setiap kali Amazon Data Firehose mengirimkan data ke titik akhir HTTP (baik upaya awal atau percobaan ulang), itu memulai ulang penghitung batas waktu pengakuan dan menunggu pengakuan dari titik akhir HTTP.

Bahkan jika durasi percobaan ulang berakhir, Amazon Data Firehose masih menunggu pengakuan sampai menerimanya atau periode batas waktu pengakuan tercapai. Jika waktu pengakuan habis, Amazon Data Firehose menentukan apakah ada waktu tersisa di penghitung coba lagi. Jika ada waktu yang tersisa, Kinesis Data Firehose akan mencoba lagi dan mengulangi logika sampai menerima pengakuan atau dapat menentukan bahwa waktu coba lagi telah berakhir.

Jika Anda tidak ingin Amazon Data Firehose mencoba lagi mengirim data, tetapkan nilai ini ke 0.

Parameter - opsional

Amazon Data Firehose menyertakan pasangan kunci-nilai ini di setiap panggilan HTTP. Parameter ini dapat membantu Anda mengidentifikasi dan mengatur tujuan.

Petunjuk penyangga

Amazon Data Firehose melakukan buffer pada data masuk sebelum mengirimkannya ke tujuan yang ditentukan. Ukuran buffer yang direkomendasikan untuk tujuan bervariasi dari tiap-tiap penyedia layanan.

Konfigurasikan pengaturan tujuan untuk Sumo Logic

Bagian ini menjelaskan opsi untuk menggunakan Sumo Logic untuk tujuan Anda. Untuk informasi selengkapnya, lihat https://www.sumologic.com.

- Berikan nilai untuk bidang berikut:

  URL titik akhir HTTP

  Tentukan URL untuk titik akhir HTTP dalam format berikut: `https://deployment name.sumologic.net/receiver/v1/kinesis/dataType/access token`. URL harus berupa URL HTTPS.

  Pengkodean konten

  Amazon Data Firehose menggunakan pengkodean konten untuk mengompres isi permintaan sebelum mengirimnya ke tujuan. Pilih GZIP atau Dinonaktifkan untuk pengkodean enable/disable konten permintaan Anda.

  Coba lagi durasi

  Tentukan berapa lama Amazon Data Firehose mencoba mengirim data ke Sumo Logic.

  Setelah mengirim data, Amazon Data Firehose akan menunggu pengakuan dari titik akhir HTTP terlebih dahulu. Jika terjadi kesalahan atau pengakuan tidak diberikan dalam periode batas waktu pengakuan, Amazon Data Firehose akan memulai penghitung durasi percobaan ulang. Kinesis Data Firehose terus mencoba kembali sampai durasi coba lagi berakhir. Setelah itu, Amazon Data Firehose menganggap pengiriman data gagal dan mencadangkan data ke bucket Amazon S3 Anda.

  Setiap kali Amazon Data Firehose mengirimkan data ke titik akhir HTTP (baik upaya awal atau percobaan ulang), itu memulai ulang penghitung batas waktu pengakuan dan menunggu pengakuan dari titik akhir HTTP.

  Bahkan jika durasi percobaan ulang berakhir, Amazon Data Firehose masih menunggu pengakuan sampai menerimanya atau periode batas waktu pengakuan tercapai. Jika waktu pengakuan habis, Amazon Data Firehose menentukan apakah ada waktu tersisa di penghitung coba lagi. Jika ada waktu yang tersisa, Kinesis Data Firehose akan mencoba lagi dan mengulangi logika sampai menerima pengakuan atau dapat menentukan bahwa waktu coba lagi telah berakhir.

Jika Anda tidak ingin Amazon Data Firehose mencoba lagi mengirim data, tetapkan nilai ini ke 0.

Parameter - opsional

Amazon Data Firehose menyertakan pasangan kunci-nilai ini di setiap panggilan HTTP. Parameter ini dapat membantu Anda mengidentifikasi dan mengatur tujuan.

Petunjuk penyangga

Amazon Data Firehose melakukan buffer pada data masuk sebelum mengirimkannya ke tujuan yang ditentukan. Ukuran buffer yang direkomendasikan untuk tujuan Elastic bervariasi dari penyedia layanan ke penyedia layanan.

Konfigurasikan pengaturan tujuan untuk Elastic

Bagian ini menjelaskan opsi untuk menggunakan Elastic untuk tujuan Anda.

- Berikan nilai untuk bidang berikut:

  URL titik akhir elastis

  Tentukan URL untuk titik akhir HTTP dalam format berikut: `https://<cluster-id>.es.<region>.aws.elastic-cloud.com`. URL harus berupa URL HTTPS.

  Autentikasi

  Anda dapat memilih untuk memasukkan kunci API secara langsung atau mengambil rahasia dari AWS Secrets Manager untuk mengakses Elastic.

  - Kunci API

    Hubungi Elastic untuk mendapatkan kunci API yang Anda perlukan untuk mengaktifkan pengiriman data ke layanan mereka dari Firehose.

  - Rahasia

    Pilih rahasia dari AWS Secrets Manager yang berisi kunci API untuk Elastic. Jika Anda tidak melihat rahasia Anda di daftar drop-down, buat satu di AWS Secrets Manager. Untuk informasi selengkapnya, lihat Otentikasi dengan AWS Secrets Manager di Amazon Data Firehose.

Pengkodean konten

Amazon Data Firehose menggunakan pengkodean konten untuk mengompres isi permintaan sebelum mengirimnya ke tujuan. Pilih GZIP (yang dipilih secara default) atau Dinonaktifkan untuk pengkodean enable/disable konten permintaan Anda.

Coba lagi durasi

Tentukan berapa lama Amazon Data Firehose mencoba mengirim data ke Elastic.

Setelah mengirim data, Amazon Data Firehose akan menunggu pengakuan dari titik akhir HTTP terlebih dahulu. Jika terjadi kesalahan atau pengakuan tidak diberikan dalam periode batas waktu pengakuan, Amazon Data Firehose akan memulai penghitung durasi percobaan ulang. Kinesis Data Firehose terus mencoba kembali sampai durasi coba lagi berakhir. Setelah itu, Amazon Data Firehose menganggap pengiriman data gagal dan mencadangkan data ke bucket Amazon S3 Anda.

Setiap kali Amazon Data Firehose mengirimkan data ke titik akhir HTTP (baik upaya awal atau percobaan ulang), itu memulai ulang penghitung batas waktu pengakuan dan menunggu pengakuan dari titik akhir HTTP.

Bahkan jika durasi percobaan ulang berakhir, Amazon Data Firehose masih menunggu pengakuan sampai menerimanya atau periode batas waktu pengakuan tercapai. Jika waktu pengakuan habis, Amazon Data Firehose menentukan apakah ada waktu tersisa di penghitung coba lagi. Jika ada waktu yang tersisa, Kinesis Data Firehose akan mencoba lagi dan mengulangi logika sampai menerima pengakuan atau dapat menentukan bahwa waktu coba lagi telah berakhir.

Jika Anda tidak ingin Amazon Data Firehose mencoba lagi mengirim data, tetapkan nilai ini ke 0.

Parameter - opsional

Amazon Data Firehose menyertakan pasangan kunci-nilai ini di setiap panggilan HTTP. Parameter ini dapat membantu Anda mengidentifikasi dan mengatur tujuan.

Petunjuk penyangga

Amazon Data Firehose melakukan buffer pada data masuk sebelum mengirimkannya ke tujuan yang ditentukan. Ukuran buffer yang direkomendasikan untuk tujuan Elastic adalah 1 MiB.

Konfigurasikan pengaturan cadangan

Amazon Data Firehose menggunakan Amazon S3 untuk mencadangkan semua atau hanya data yang gagal yang coba dikirim ke tujuan yang Anda pilih.

> **Important**
>
> - Pengaturan Backup hanya didukung jika sumber aliran Firehose Anda adalah Direct PUT atau Kinesis Data Streams.
> - Fitur buffering nol hanya tersedia untuk tujuan aplikasi dan tidak tersedia untuk tujuan cadangan Amazon S3.

Anda dapat menentukan pengaturan cadangan S3 untuk aliran Firehose Anda jika Anda membuat salah satu pilihan berikut.

- Jika Anda menetapkan Amazon S3 sebagai tujuan aliran Firehose Anda dan Anda memilih untuk menentukan fungsi AWS Lambda untuk mengubah catatan data atau jika Anda memilih untuk mengonversi format rekaman data untuk aliran Firehose Anda.
- Jika Anda menetapkan Amazon Redshift sebagai tujuan aliran Firehose Anda dan Anda memilih untuk menentukan fungsi AWS Lambda untuk mengubah catatan data.
- Jika Anda menetapkan salah satu layanan berikut sebagai tujuan untuk aliran Firehose Anda — Amazon OpenSearch Service, Datadog, Dynatrace, HTTP Endpoint,, LogicMonitor MongoDB Cloud, New Relic, Splunk, atau Sumo Logic, Snowflake, Apache Iceberg Tables.

Berikut ini adalah pengaturan cadangan untuk aliran Firehose Anda.

- Pencadangan catatan sumber di Amazon S3 - jika S3 atau Amazon Redshift adalah tujuan yang Anda pilih, pengaturan ini menunjukkan apakah Anda ingin mengaktifkan cadangan data sumber atau menonaktifkannya. Jika layanan lain yang didukung (selain S3 atau Amazon Redshift) ditetapkan sebagai tujuan yang Anda pilih, maka pengaturan ini menunjukkan jika Anda ingin mencadangkan semua data sumber atau data yang gagal saja.
- Bucket cadangan S3 - ini adalah bucket S3 tempat Amazon Data Firehose mencadangkan data Anda.
- Awalan bucket cadangan S3 - ini adalah awalan tempat Amazon Data Firehose mencadangkan data Anda.

- Awalan keluaran kesalahan bucket cadangan S3 - semua data yang gagal dicadangkan dalam awalan keluaran kesalahan bucket S3 ini.
- Petunjuk penyangga, kompresi, dan enkripsi untuk pencadangan - Amazon Data Firehose menggunakan Amazon S3 untuk mencadangkan semua atau hanya gagal data yang coba dikirim ke tujuan yang Anda pilih. Amazon Data Firehose menyangga data yang masuk sebelum mengirimkannya (mencadangkannya) ke Amazon S3. Anda dapat memilih ukuran buffer 1—128 MiBs dan interval buffer 60—900 detik. Syarat pertama yang dipenuhi memicu pengiriman data ke Amazon S3. Jika Anda mengaktifkan transformasi data, interval buffer berlaku dari waktu data yang diubah diterima oleh Amazon Data Firehose hingga pengiriman data ke Amazon S3. Jika pengiriman data ke tujuan tertinggal dari penulisan data ke aliran Firehose, Amazon Data Firehose meningkatkan ukuran buffer secara dinamis untuk mengejar ketinggalan. Tindakan ini membantu memastikan bahwa semua data dikirim ke tujuan.
- Kompresi S3 - pilih kompresi data Snappy, Snappy, Zip, atau Hadoop-Compatible Snappy, atau tidak ada kompresi data. Kompresi Snappy Snappy, Zip, dan Hadoop-Compatible Snappy tidak tersedia untuk aliran Firehose dengan Amazon Redshift sebagai tujuannya.
- Format ekstensi file S3 (opsional) — Tentukan format ekstensi file untuk objek yang dikirim ke bucket tujuan Amazon S3. Jika Anda mengaktifkan fitur ini, ekstensi file yang ditentukan akan mengganti ekstensi file default yang ditambahkan oleh Konversi Format Data atau fitur kompresi S3 seperti.parquet atau.gz. Pastikan jika Anda mengonfigurasi ekstensi file yang benar saat Anda menggunakan fitur ini dengan Konversi Format Data atau kompresi S3. Ekstensi file harus dimulai dengan titik (.) dan dapat berisi karakter yang diizinkan: 0-9a-z! -_.*' (). Ekstensi file tidak boleh melebihi 128 karakter.
- Firehose mendukung enkripsi sisi server Amazon S3 AWS Key Management Service dengan (SSE-KMS) untuk mengenkripsi data yang dikirimkan di Amazon S3. Anda dapat memilih untuk menggunakan jenis enkripsi default yang ditentukan dalam bucket S3 tujuan atau untuk mengenkripsi dengan kunci dari daftar AWS KMS kunci yang Anda miliki. Jika Anda mengenkripsi data dengan AWS KMS kunci, Anda dapat menggunakan kunci AWS terkelola default (aws/s3) atau kunci yang dikelola pelanggan. Untuk informasi selengkapnya, lihat Melindungi Data Menggunakan Enkripsi Sisi Server dengan Kunci yang Dikelola KMS (AWS SSE-KMS).

Konfigurasikan petunjuk buffering

Amazon Data Firehose menyangga data streaming yang masuk dalam memori ke ukuran tertentu (ukuran buffering) dan untuk jangka waktu tertentu (interval buffering) sebelum mengirimkannya ke tujuan yang ditentukan. Anda akan menggunakan petunjuk buffering ketika Anda ingin mengirimkan

file berukuran optimal ke Amazon S3 dan mendapatkan kinerja yang lebih baik dari aplikasi pemrosesan data atau untuk menyesuaikan tingkat pengiriman Firehose agar sesuai dengan kecepatan tujuan.

Anda dapat mengonfigurasi ukuran buffering dan interval buffer sambil membuat aliran Firehose baru atau memperbarui ukuran buffering dan interval buffering pada aliran Firehose yang ada. Ukuran buffering diukur dalam MBs dan interval buffering diukur dalam hitungan detik. Akan tetapi, jika Anda menentukan nilai untuk salah satunya, Anda juga harus menyediakan nilai untuk yang lain. Kondisi buffer pertama yang puas memicu Firehose untuk mengirimkan data. Jika Anda tidak mengonfigurasi nilai buffering, maka nilai default akan digunakan.

Anda dapat mengonfigurasi petunjuk buffering Firehose melalui,, atau. Konsol Manajemen AWS AWS Command Line Interface AWS SDKs Untuk aliran yang ada, Anda dapat mengonfigurasi ulang petunjuk buffering dengan nilai yang sesuai dengan kasus penggunaan menggunakan opsi Edit di konsol atau menggunakan API. UpdateDestination Untuk aliran baru, Anda dapat mengonfigurasi petunjuk buffering sebagai bagian dari pembuatan aliran baru menggunakan konsol atau menggunakan API. CreateDeliveryStream Untuk menyesuaikan ukuran buffering, atur `SizeInMBs` dan `IntervalInSeconds` di `DestinationConfiguration` parameter spesifik tujuan CreateDeliveryStreamatau UpdateDestinationAPI.

> **Note**
>
> - Petunjuk buffer diterapkan pada tingkat pecahan atau partisi, sementara petunjuk buffer partisi dinamis diterapkan pada tingkat aliran atau topik.
> - Untuk memenuhi latensi yang lebih rendah dari kasus penggunaan waktu nyata, Anda dapat menggunakan petunjuk interval buffering nol. Saat Anda mengonfigurasi interval buffering sebagai nol detik, Firehose tidak akan menyangga data dan akan mengirimkan data dalam beberapa detik. Sebelum Anda mengubah petunjuk buffering ke nilai yang lebih rendah, tanyakan kepada vendor untuk petunjuk buffering Firehose yang direkomendasikan untuk tujuan mereka.
> - Fitur buffering nol hanya tersedia untuk tujuan aplikasi dan tidak tersedia untuk tujuan cadangan Amazon S3.
> - Fitur buffering nol tidak tersedia untuk partisi dinamis.
> - Firehose menggunakan unggahan multi-bagian untuk tujuan S3 saat Anda mengonfigurasi interval waktu buffer kurang dari 60 detik untuk menawarkan latensi yang lebih rendah.

Karena unggahan multi-bagian untuk tujuan S3, Anda akan melihat beberapa peningkatan biaya PUT API S3 jika Anda memilih interval waktu buffer kurang dari 60 detik.

Untuk rentang petunjuk buffering spesifik tujuan dan nilai default, lihat tabel berikut:

Destinasi	Ukuran buffering dalam MB (default dalam tanda kurung)	Interval buffering dalam hitungan detik (default dalam tanda kurung)
Amazon S3	1-128 (5)	0-900 (300)
Tabel Gunung Es Apache	1-128 (5)	0-900 (300)
Amazon Redshift	1-128 (5)	0-900 (300)
OpenSearch Tanpa server	1-100 (5)	0-900 (300)
OpenSearch	1-100 (5)	0-900 (300)
Splunk	1-5 (5)	0-60 (60)
Datadog	1-4 (4)	0-900 (60)
Coralogix	1-64 (6)	0-900 (60)
Dynatrace	1-64 (5)	0-900 (60)
Elastis	1	0-900 (60)
Honeycomb	1-64 (15)	0-900 (60)
Titik akhir HTTP	1-64 (5)	0-900 (60)
LogicMonitor	1-64 (5)	0-900 (60)

Destinasi	Ukuran buffering dalam MB (default dalam tanda kurung)	Interval buffering dalam hitungan detik (default dalam tanda kurung)
Logzio	1-64 (5)	0-900 (60)
MongoDB	1-16 (5)	0-900 (60)
NewRelic	1-64 (5)	0-900 (60)
SumoLogic	1-64 (1)	0-900 (60)
Cloud Observabilitas Splunk	1-64 (1)	0-900 (60)
Kepingan salju	1 - 128 (1)	0 - 900 (0)

Mengonfigurasi pengaturan lanjutan

Bagian berikut berisi detail tentang pengaturan lanjutan untuk aliran Firehose Anda.

- Enkripsi sisi server - Amazon Data Firehose mendukung enkripsi sisi server Amazon S3 AWS dengan Key Management Service (AWS KMS) untuk mengenkripsi data yang dikirimkan di Amazon S3. Untuk informasi selengkapnya, lihat Melindungi Data Menggunakan Enkripsi Sisi Server dengan Kunci Terkelola KMS (AWS SSE-KMS).
- Pencatatan kesalahan - Amazon Data Firehose mencatat kesalahan yang terkait dengan pemrosesan dan pengiriman. Selain itu, ketika transformasi data diaktifkan, ia dapat mencatat pemanggilan Lambda dan mengirim kesalahan pengiriman data ke Log. CloudWatch Untuk informasi selengkapnya, lihat Pantau Amazon Data Firehose Menggunakan Log CloudWatch.

> **Important**
>
> Meskipun opsional, mengaktifkan pencatatan kesalahan Amazon Data Firehose selama pembuatan aliran Firehose sangat disarankan. Praktik ini memastikan bahwa Anda dapat mengakses detail kesalahan jika terjadi pemrosesan catatan atau kegagalan pengiriman.

- Izin - Amazon Data Firehose menggunakan peran IAM untuk semua izin yang dibutuhkan aliran Firehose. Anda dapat memilih untuk membuat peran baru di mana izin yang diperlukan ditetapkan secara otomatis, atau memilih peran yang sudah ada yang dibuat untuk Amazon Data Firehose. Peran ini digunakan untuk memberikan akses Firehose ke berbagai layanan, termasuk bucket S3, kunci AWS KMS (jika enkripsi data diaktifkan), dan fungsi Lambda (jika transformasi data diaktifkan). Konsol dapat membuat peran dengan placeholder. Untuk informasi lebih lanjut, lihat Apa itu IAM? .

  > **Note**
  >
  > Peran IAM (termasuk placeholder) dibuat berdasarkan konfigurasi yang Anda pilih saat membuat aliran Firehose. Jika Anda membuat perubahan apa pun pada sumber atau tujuan aliran Firehose, Anda harus memperbarui peran IAM secara manual.

- Tag - Anda dapat menambahkan tag untuk mengatur AWS sumber daya Anda, melacak biaya, dan mengontrol akses.

  Jika Anda menentukan tag dalam `CreateDeliveryStream` tindakan, Amazon Data Firehose akan melakukan otorisasi tambahan pada `firehose:TagDeliveryStream` tindakan tersebut untuk memverifikasi apakah pengguna memiliki izin untuk membuat tag. Jika Anda tidak memberikan izin ini, permintaan untuk membuat aliran Firehose baru dengan tag sumber daya IAM akan gagal dengan hal seperti berikut. `AccessDeniedException`

  ```
  AccessDeniedException
  User: arn:aws:sts::x:assumed-role/x/x is not authorized to perform:
   firehose:TagDeliveryStream on resource: arn:aws:firehose:us-east-1:x:deliverystream/
  x with an explicit deny in an identity-based policy.
  ```

  Contoh berikut menunjukkan kebijakan yang memungkinkan pengguna membuat aliran Firehose dan menerapkan tag.

Setelah Anda memilih pengaturan cadangan dan lanjutan, tinjau pilihan Anda, lalu pilih Create Firehose stream.

Aliran Firehose baru membutuhkan beberapa saat dalam status Membuat sebelum tersedia. Setelah aliran Firehose Anda dalam status Aktif, Anda dapat mulai mengirim data ke sana dari produsen Anda.

Menguji aliran Firehose dengan data sampel

Anda dapat menggunakan data ticker saham simulasi Konsol Manajemen AWS untuk menelan. Konsol menjalankan skrip di browser Anda untuk menempatkan catatan sampel di aliran Firehose Anda. Ini memungkinkan Anda untuk menguji konfigurasi aliran Firehose Anda tanpa harus menghasilkan data pengujian Anda sendiri.

Berikut ini adalah contoh dari data simulasi:

```
{"TICKER_SYMBOL":"QXZ","SECTOR":"HEALTHCARE","CHANGE":-0.05,"PRICE":84.51}
```

Perhatikan bahwa biaya Firehose Data Amazon standar berlaku saat aliran Firehose Anda mentransmisikan data, tetapi tidak dikenakan biaya saat data dihasilkan. Untuk berhenti menimbulkan biaya ini, Anda dapat menghentikan aliran sampel dari konsol kapan saja.

Prasyarat

Sebelum Anda mulai, buat aliran Firehose. Untuk informasi selengkapnya, lihat Tutorial: Membuat aliran Firehose dari konsol.

Uji dengan Amazon S3

Gunakan prosedur berikut untuk menguji aliran Firehose Anda dengan Amazon Simple Storage Service (Amazon S3) sebagai tujuan.

Untuk menguji aliran Firehose menggunakan Amazon S3

1. Buka konsol Firehose di. https://console.aws.amazon.com/firehose/
2. Pilih aliran Firehose yang aktif. Aliran Firehose harus dalam status Aktif sebelum Anda dapat mulai mengirim data.
3. Di bawah Uji dengan data demo, pilih Mulai mengirim data demo untuk menghasilkan sampel data ticker saham.
4. Ikuti petunjuk di layar untuk memverifikasi bahwa data tersebut sedang dikirim ke bucket S3 Anda. Perhatikan bahwa mungkin diperlukan beberapa menit hingga objek baru muncul di bucket Anda, berdasarkan konfigurasi buffering bucket Anda.

5. Saat pengujian selesai, pilih Berhenti mengirim data demo untuk berhenti menimbulkan biaya penggunaan.

Uji dengan Amazon Redshift

Gunakan prosedur berikut untuk menguji aliran Firehose Anda dengan Amazon Redshift sebagai tujuannya.

Untuk menguji aliran Firehose menggunakan Amazon Redshift

1. Aliran Firehose Anda mengharapkan tabel hadir di klaster Amazon Redshift Anda. Hubungkan ke Amazon Redshift melalui antarmuka SQL dan jalankan pernyataan berikut untuk membuat tabel yang menerima data sampel.

```
create table firehose_test_table
(
 TICKER_SYMBOL varchar(4),
 SECTOR varchar(16),
 CHANGE float,
 PRICE float
);
```

2. Buka konsol Firehose di. https://console.aws.amazon.com/firehose/
3. Pilih aliran Firehose yang aktif. Aliran Firehose harus dalam status Aktif sebelum Anda dapat mulai mengirim data.
4. Edit detail tujuan aliran Firehose Anda untuk mengarah ke tabel yang baru dibuat`firehose_test_table`.
5. Di bawah Uji dengan data demo, pilih Mulai mengirim data demo untuk menghasilkan sampel data ticker saham.
6. Ikuti petunjuk di layar untuk memverifikasi bahwa data tersebut sedang dikirim ke tabel Anda. Perhatikan bahwa mungkin diperlukan beberapa menit hingga baris baru muncul di tabel Anda, berdasarkan konfigurasi buffering.
7. Saat pengujian selesai, pilih Berhenti mengirim data demo untuk berhenti menimbulkan biaya penggunaan.
8. Edit detail tujuan untuk aliran Firehose Anda untuk mengarah ke tabel lain.
9. (Opsional) Hapus tabel `firehose_test_table`.

Uji dengan OpenSearch Layanan

Gunakan prosedur berikut untuk menguji aliran Firehose Anda menggunakan OpenSearch Layanan Amazon sebagai tujuan.

Untuk menguji aliran Firehose menggunakan Service OpenSearch

1. Buka konsol Firehose di. https://console.aws.amazon.com/firehose/
2. Pilih aliran Firehose yang aktif. Aliran Firehose harus dalam status Aktif sebelum Anda dapat mulai mengirim data.
3. Di bawah Uji dengan data demo, pilih Mulai mengirim data demo untuk menghasilkan sampel data ticker saham.
4. Ikuti petunjuk di layar untuk memverifikasi bahwa data sedang dikirimkan ke domain OpenSearch Layanan Anda. Untuk informasi selengkapnya, lihat Mencari Dokumen di Domain OpenSearch Layanan di Panduan Pengembang OpenSearch Layanan Amazon.
5. Saat pengujian selesai, pilih Berhenti mengirim data demo untuk berhenti menimbulkan biaya penggunaan.

Uji dengan Splunk

Gunakan prosedur berikut untuk menguji aliran Firehose Anda menggunakan Splunk sebagai tujuan.

Untuk menguji aliran Firehose menggunakan Splunk

1. Buka konsol Firehose di. https://console.aws.amazon.com/firehose/
2. Pilih aliran Firehose yang aktif. Aliran Firehose harus dalam status Aktif sebelum Anda dapat mulai mengirim data.
3. Di bawah Uji dengan data demo, pilih Mulai mengirim data demo untuk menghasilkan sampel data ticker saham.
4. Periksa apakah data tersebut sedang dikirim ke indeks Splunk Anda. Contoh istilah pencarian di Splunk adalah `sourcetype="aws:firehose:json"` dan `index="name-of-your-splunk-index"`. Untuk informasi selengkapnya tentang cara mencari peristiwa di Splunk, lihat Cari Manual dalam dokumentasi Splunk.

   Jika data pengujian tidak muncul dalam indeks Splunk Anda, periksa peristiwa yang gagal di bucket Amazon S3 Anda. Lihat juga Data Tidak Terkirim ke Splunk.

5. Saat pengujian selesai, pilih Berhenti mengirim data demo untuk berhenti menimbulkan biaya penggunaan.

Tes dengan Apache Iceberg Tables

Gunakan prosedur berikut untuk menguji aliran Firehose Anda dengan Apache Iceberg Tables sebagai tujuan.

Untuk menguji aliran Firehose menggunakan Apache Iceberg Tables

1. Buka konsol Firehose di. https://console.aws.amazon.com/firehose/
2. Pilih aliran Firehose yang aktif. Aliran Firehose harus dalam status Aktif sebelum Anda dapat mulai mengirim data.
3. Di bawah Uji dengan data demo, pilih Mulai mengirim data demo untuk menghasilkan sampel data ticker saham.
4. Ikuti petunjuk di layar untuk memverifikasi bahwa data sedang dikirimkan ke Tabel Gunung Es Apache Anda. Perhatikan bahwa mungkin perlu beberapa menit agar objek baru muncul di bucket Anda, berdasarkan konfigurasi buffering-nya.
5. Jika data pengujian tidak muncul di Tabel Gunung Es Apache, periksa bucket Amazon S3 untuk mengetahui kejadian yang gagal.
6. Saat pengujian selesai, pilih Berhenti mengirim data demo untuk berhenti menimbulkan biaya penggunaan.

Mengirim data ke aliran Firehose

Bagian ini menjelaskan bagaimana Anda dapat menggunakan sumber data yang berbeda untuk mengirim data ke aliran Firehose Anda. Jika Anda baru mengenal Amazon Data Firehose, luangkan waktu untuk membiasakan diri dengan konsep dan terminologi yang disajikan. Apa itu Amazon Data Firehose?

> **Note**
>
> Beberapa AWS layanan hanya dapat mengirim pesan dan acara ke aliran Firehose yang berada di Wilayah yang sama. Jika aliran Firehose Anda tidak muncul sebagai opsi saat mengonfigurasi target untuk CloudWatch Log Amazon, CloudWatch Peristiwa, atau AWS IoT, verifikasi bahwa aliran Firehose Anda berada di Wilayah yang sama dengan layanan Anda yang lain. Untuk informasi tentang titik akhir layanan untuk setiap Wilayah, lihat titik akhir Amazon Data Firehose.

Anda dapat mengirim data ke aliran Firehose Anda dari sumber data berikut.

Topik

Konfigurasikan agen Kinesis untuk mengirim data

Agen Amazon Kinesis adalah aplikasi perangkat lunak Java mandiri yang berfungsi sebagai implementasi referensi untuk menunjukkan bagaimana Anda dapat mengumpulkan dan mengirim data ke Firehose. Agen terus memantau sekumpulan file dan mengirimkan data baru ke aliran Firehose Anda. Agen menunjukkan bagaimana Anda dapat menangani rotasi file, checkpointing, dan coba lagi setelah kegagalan. Ini menunjukkan bagaimana Anda dapat mengirimkan data Anda dengan cara yang andal, tepat waktu, dan sederhana. Ini juga menunjukkan bagaimana Anda dapat

memancarkan CloudWatch metrik untuk memantau dan memecahkan masalah proses streaming dengan lebih baik. Untuk mempelajari lebih lanjut, awslabs/ amazon-kinesis-agent.

Secara default, catatan diurai dari setiap file berdasarkan karakter baris baru (`'\n'`). Namun, agen juga dapat dikonfigurasi untuk mengurai catatan multi-baris (lihat Tentukan pengaturan konfigurasi agen).

Anda dapat menginstal agen di lingkungan server berbasis Linux seperti server web, server log, dan server basis data. Setelah menginstal agen, konfigurasikan dengan menentukan file yang akan dipantau dan aliran Firehose untuk data. Setelah agen dikonfigurasi, agen akan mengumpulkan data dari file dan mengirimkannya dengan andal ke aliran Firehose.

Prasyarat

Sebelum Anda mulai menggunakan Agen Kinesis, pastikan Anda memenuhi prasyarat berikut.

- Sistem operasi Anda harus Amazon Linux, atau Red Hat Enterprise Linux versi 7 atau yang lebih baru.
- Agen versi 2.0.0 atau yang lebih baru berjalan menggunakan JRE versi 1.8 atau yang lebih baru. Agen versi 1.1.x berjalan menggunakan JRE 1.7 atau yang lebih baru.
- Jika Anda menggunakan Amazon EC2 untuk menjalankan agen Anda, luncurkan instans EC2 Anda.
- Peran atau AWS kredensil IAM yang Anda tentukan harus memiliki izin untuk menjalankan PutRecordBatchoperasi Amazon Data Firehose agar agen dapat mengirim data ke aliran Firehose Anda. Jika Anda mengaktifkan CloudWatch pemantauan untuk agen, izin untuk melakukan CloudWatch PutMetricDataoperasi juga diperlukan. Untuk informasi selengkapnya, lihat Mengontrol akses dengan Amazon Data FirehosePantau kesehatan Agen Kinesis, dan Otentikasi dan Kontrol Akses untuk Amazon CloudWatch.

Mengelola AWS kredensyal

Kelola AWS kredensyal Anda menggunakan salah satu metode berikut:

- Buat penyedia kredensial khusus. Lihat perinciannya di the section called “Buat penyedia kredensi khusus”.
- Tentukan IAM role ketika Anda meluncurkan instans EC2 Anda.

- Tentukan AWS kredensil saat Anda mengonfigurasi agen (lihat entri untuk `awsAccessKeyId` dan `awsSecretAccessKey` di tabel konfigurasi di bawah). the section called “Tentukan pengaturan konfigurasi agen”
- Edit `/etc/sysconfig/aws-kinesis-agent` untuk menentukan AWS Region dan kunci AWS akses Anda.
- Jika instans EC2 Anda berada di AWS akun yang berbeda, buat peran IAM untuk menyediakan akses ke layanan Amazon Data Firehose. Tentukan peran tersebut saat Anda mengonfigurasi agen (lihat assumeRoleExternalAssumeroLearn dan Id). Gunakan salah satu metode sebelumnya untuk menentukan AWS kredensi pengguna di akun lain yang memiliki izin untuk mengambil peran ini.

Buat penyedia kredensi khusus

Anda dapat membuat penyedia kredensial khusus dan memberikan nama kelas dan jalur jarnya ke agen Kinesis dalam pengaturan konfigurasi berikut: `userDefinedCredentialsProvider.classname` dan `userDefinedCredentialsProvider.location`. Untuk deskripsi dari dua pengaturan konfigurasi ini, lihat the section called “Tentukan pengaturan konfigurasi agen”.

Untuk membuat penyedia kredensial khusus, tentukan kelas yang mengimplementasikan antarmuka `AWS CredentialsProvider`, seperti yang ada di contoh berikut.

```
import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.AWSCredentialsProvider;
import com.amazonaws.auth.BasicAWSCredentials;

public class YourClassName implements AWSCredentialsProvider {
    public YourClassName() {
    }

    public AWSCredentials getCredentials() {
        return new BasicAWSCredentials("key1", "key2");
    }

    public void refresh() {
    }
}
```

Kelas Anda harus memiliki konstruktor yang tidak mengambil argumen.

AWS memanggil metode penyegaran secara berkala untuk mendapatkan kredensyal yang diperbarui. Jika Anda ingin penyedia kredensial Anda memberikan kredensial yang berbeda sepanjang masa pakainya, sertakan kode untuk menyegarkan kredensial dalam metode ini. Atau, Anda dapat membiarkan metode ini kosong jika Anda ingin penyedia kredensial yang menyediakan kredensial statis (tidak berubah).

Unduh dan instal Agen

Pertama-tama, hubungkan ke instans Anda. Untuk informasi selengkapnya, lihat Connect to Your Instance di Panduan Pengguna Amazon EC2. Jika Anda mengalami masalah saat menyambung, lihat Pemecahan Masalah Menyambung ke Instans Anda di Panduan Pengguna Amazon EC2.

Selanjutnya, instal agen menggunakan salah satu metode berikut.

- Untuk mengatur agen dari repositori Amazon Linux

  Metode ini hanya berfungsi untuk instans Amazon Linux. Gunakan perintah berikut:

  ```
  sudo yum install -y aws-kinesis-agent
  ```

  Agen v 2.0.0 atau yang lebih baru diinstal pada komputer dengan sistem operasi Amazon Linux 2 ()AL2. Versi agen ini membutuhkan Java versi 1.8 atau yang lebih baru. Jika versi Java yang diperlukan belum ada, proses instalasi agen akan menginstalnya. Untuk informasi lebih lanjut tentang Amazon Linux 2 lihat https://aws.amazon.com/amazon-linux-2/.
- Untuk mengatur agen dari repositori Amazon S3

  Metode ini berfungsi untuk Red Hat Enterprise Linux, serta instans Amazon Linux 2 karena metode ini menginstal agen dari repositori yang tersedia untuk umum. Gunakan perintah berikut untuk mengunduh dan menginstal versi terbaru dari versi agen 2.x.x:

  ```
  sudo yum install -y https://s3.amazonaws.com/streaming-data-agent/aws-kinesis-agent-latest.amzn2.noarch.rpm
  ```

  Untuk menginstal versi tertentu dari agen, tentukan nomor versi dalam perintah. Sebagai contoh, perintah berikut menginstal agen v 2.0.1.

```
sudo yum install -y https://streaming-data-agent.s3.amazonaws.com/aws-kinesis-
agent-2.0.1-1.amzn1.noarch.rpm
```

Jika Anda memiliki Java 1.7 dan tidak ingin meng-upgrade-nya, Anda dapat mengunduh agen versi 1.x.x, yang kompatibel dengan Java 1.7. Misalnya, untuk mengunduh agen v1.1.6, Anda dapat menggunakan perintah berikut:

```
sudo yum install -y https://s3.amazonaws.com/streaming-data-agent/aws-kinesis-
agent-1.1.6-1.amzn1.noarch.rpm
```

Anda dapat mengunduh agen terbaru dengan perintah berikut

```
sudo yum install -y https://s3.amazonaws.com/streaming-data-agent/aws-kinesis-agent-
latest.amzn2.noarch.rpm
```

- Untuk mengatur agen dari GitHub repo

  1. Pertama, pastikan bahwa Anda menginstal versi Java yang diperlukan, tergantung pada versi agen.
  2. Unduh agen dari amazon-kinesis-agent GitHub awslabs/repo.
  3. Instal agen dengan menavigasi ke direktori unduhan dan menjalankan perintah berikut:

     ```
     sudo ./setup --install
     ```

- Untuk mengatur agen dalam wadah Docker

  Agen Kinesis dapat dijalankan dalam wadah juga melalui basis wadah amazonlinux. Gunakan Dockerfile berikut dan kemudian jalankan. `docker build`

```
FROM amazonlinux

RUN yum install -y aws-kinesis-agent which findutils
COPY agent.json /etc/aws-kinesis/agent.json
```

```
CMD ["start-aws-kinesis-agent"]
```

Konfigurasikan dan mulai Agen

Untuk mengonfigurasi dan memulai agen

1. Buka dan edit file konfigurasi (sebagai pengguna super jika menggunakan izin akses file default): `/etc/aws-kinesis/agent.json`

   Dalam file konfigurasi ini, tentukan file (`"filePattern"`) dari mana agen mengumpulkan data, dan nama Firehose stream `"deliveryStream"` () tempat agen mengirimkan data. Nama file merupakan pola, dan agen mengenali rotasi file. Anda dapat memutar file atau membuat file baru satu kali per detik. Agen menggunakan stempel waktu pembuatan file untuk menentukan file mana yang akan dilacak dan diekor ke aliran Firehose Anda. Membuat file baru atau memutar file lebih sering dari satu kali per detik membuat agen tidak dapat membedakan file-file tersebut dengan benar.

   ```
   {
      "flows": [
           {
               "filePattern": "/tmp/app.log*",
               "deliveryStream": "yourdeliverystream"
           }
      ]
   }
   ```

   AWS Wilayah default adalah`us-east-1`. Jika Anda menggunakan Wilayah yang berbeda, tambahkan pengaturan `firehose.endpoint` ke file konfigurasi, dan tentukan titik akhir untuk Wilayah Anda. Untuk informasi selengkapnya, lihat Tentukan pengaturan konfigurasi agen.

2. Mulailah agen secara manual:

   ```
   sudo service aws-kinesis-agent start
   ```

3. (Opsional) Konfigurasikan agen untuk memulai pada startup sistem:

   ```
   sudo chkconfig aws-kinesis-agent on
   ```

Agen sekarang berjalan sebagai layanan sistem di latar belakang. Ini terus memantau file yang ditentukan dan mengirim data ke aliran Firehose yang ditentukan. Aktivitas agen masuk di `/var/log/aws-kinesis-agent/aws-kinesis-agent.log`.

Tentukan pengaturan konfigurasi agen

Agen mendukung dua pengaturan konfigurasi wajib, `filePattern` dan `deliveryStream`, ditambah pengaturan konfigurasi opsional untuk fitur tambahan. Anda dapat menentukan pengaturan konfigurasi wajib dan opsional di `/etc/aws-kinesis/agent.json`.

Setiap kali mengubah file konfigurasi, Anda harus menghentikan dan memulai agen, menggunakan perintah berikut:

```
sudo service aws-kinesis-agent stop
sudo service aws-kinesis-agent start
```

Atau, Anda dapat menggunakan perintah berikut:

```
sudo service aws-kinesis-agent restart
```

Berikut ini adalah pengaturan konfigurasi umum.

Pengaturan Konfigurasi	Deskripsi
`assumeRoleARN`	Amazon Resource Name (ARN) dari peran yang diambil oleh pengguna. Untuk informasi selengkapnya, lihat Mendelegasikan Akses di Seluruh AWS Akun Menggunakan Peran IAM di Panduan Pengguna IAM.
`assumeRoleExternalId`	Pengidentifikasi opsional yang menentukan siapa yang dapat mengambil peran tersebut. Untuk informasi selengkapnya, lihat Cara Menggunakan ID Eksternal di Panduan Pengguna IAM.
`awsAccessKeyId`	AWS ID kunci akses yang mengesampingkan kredensyal default. Pengaturan ini diutamakan daripada semua penyedia kredensial lainnya.
`awsSecretAccessKey`	AWS kunci rahasia yang mengesampingkan kredensi default. Pengaturan ini diutamakan daripada semua penyedia kredensial lainnya.

Pengaturan Konfigurasi	Deskripsi
`cloudwatch.emitMetrics`	Memungkinkan agen untuk memancarkan metrik ke CloudWatch if set (true). Default: betul
`cloudwatch.endpoint`	Titik akhir regional untuk CloudWatch. Default: `monitoring.us-east-1.amazonaws.com`
`firehose.endpoint`	Titik akhir regional untuk Amazon Data Firehose. Default: `firehose.us-east-1.amazonaws.com`
`sts.endpoint`	Titik akhir regional untuk Layanan Token AWS Keamanan. Default: `https://sts.amazonaws.com`
`userDefinedCredentialsProvider.classname`	Jika Anda menentukan penyedia kredensial khusus, beri nama kelas yang sepenuhnya memenuhi syarat menggunakan pengaturan ini. Jangan sertakan `.class` pada akhir nama kelas.
`userDefinedCredentialsProvider.location`	Jika Anda menentukan penyedia kredensial khusus, gunakan pengaturan ini untuk menentukan jalur absolut dari jar yang berisi penyedia kredensial khusus. Agen juga mencari file jar di lokasi berikut: `/usr/share/aws-kinesis-agent/lib/` .

Berikut ini adalah pengaturan konfigurasi aliran.

Pengaturan Konfigurasi	Deskripsi
`aggregatedRecordSizeBytes`	Untuk membuat catatan agregat agen dan kemudian memasukkannya ke aliran Firehose dalam satu operasi, tentukan pengaturan ini. Setel ke ukuran yang Anda inginkan untuk memiliki catatan agregat sebelum agen memasukkannya ke aliran Firehose.

Pengaturan Konfigurasi	Deskripsi
	Default: 0 (tidak ada agregasi)
`dataProcessingOptions`	Daftar opsi pemrosesan diterapkan ke setiap catatan yang diuraikan sebelum dikirim ke aliran Firehose. Pilihan pemrosesan dilakukan dalam urutan yang ditentukan. Untuk informasi selengkapnya, lihat Pra-proses data dengan Agen.
`deliveryStream`	[Wajib] Nama aliran Firehose.
`filePattern`	[Diperlukan] Sebuah glob untuk file yang perlu dipantau oleh agen. Setiap file yang cocok dengan pola ini diambil oleh agen secara otomatis dan dipantau. Untuk semua file yang cocok dengan pola ini, berikan izin baca untuk `aws-kinesis-agent-user` . Untuk direktori yang berisi file, berikan izin baca dan eksekusi untuk `aws-kinesis-agent-user` . **Important** Agen mengambil file yang cocok dengan pola ini. Untuk memastikan bahwa agen tidak mengambil catatan yang tidak diinginkan, pilih pola ini dengan hati-hati.
`initialPosition`	Posisi awal dari mana file mulai diurai. Nilai yang valid adalah `START_OF_FILE` dan `END_OF_FILE` . Default: `END_OF_FILE`
`maxBufferAgeMillis`	Waktu maksimum, dalam milidetik, di mana agen buffer data sebelum mengirimnya ke aliran Firehose. Kisaran nilai: 1.000–900.000 (1 detik sampai 15 menit) Default: 60.000 (1 menit)

Pengaturan Konfigurasi	Deskripsi
`maxBufferSizeBytes`	Ukuran maksimum, dalam byte, dimana agen buffer data sebelum mengirimnya ke aliran Firehose. Kisaran nilai: 1–4.194.304 (4 MB) Default: 4.194.304 (4 MB)
`maxBufferSizeRecords`	Jumlah maksimum catatan yang agen buffer data sebelum mengirimnya ke aliran Firehose. Kisaran nilai: 1–500 Default: 500
`minTimeBetweenFilePollsMillis`	Interval waktu, dalam milidetik, saat agen melakukan polling dan mengurai file yang dipantau untuk data baru. Kisaran nilai: 1 atau lebih Default: 100
`multiLineStartPattern`	Pola untuk mengidentifikasi awal catatan. Catatan dibuat dari baris yang cocok dengan pola tersebut dan baris berikutnya yang tidak cocok dengan pola tersebut. Nilai-nilai yang benar adalah ekspresi reguler. Secara default, setiap baris baru dalam file log diurai sebagai satu catatan.
`skipHeaderLines`	Jumlah baris yang dilewati agen untuk diurai di awal file yang dipantau. Kisaran nilai: 0 atau lebih Default: 0 (nol)
`truncatedRecordTerminator`	String yang digunakan agen untuk memotong rekaman yang diuraikan saat ukuran rekaman melebihi batas ukuran rekaman Amazon Data Firehose. (1.000 KB) Default: `'\n'` (baris baru)

Konfigurasikan beberapa direktori dan aliran file

Dengan menentukan beberapa pengaturan konfigurasi aliran, Anda dapat mengonfigurasi agen untuk memantau beberapa direktori file dan mengirim data ke beberapa aliran. Dalam contoh konfigurasi berikut, agen memonitor dua direktori file dan mengirimkan data ke aliran data Kinesis dan aliran Firehose masing-masing. Anda dapat menentukan titik akhir yang berbeda untuk Kinesis Data Streams dan Amazon Data Firehose sehingga aliran data dan aliran Firehose Anda tidak perlu berada di Wilayah yang sama.

```
{
    "cloudwatch.emitMetrics": true,
    "kinesis.endpoint": "https://your/kinesis/endpoint",
    "firehose.endpoint": "https://your/firehose/endpoint",
    "flows": [
        {
            "filePattern": "/tmp/app1.log*",
            "kinesisStream": "yourkinesisstream"
        },
        {
            "filePattern": "/tmp/app2.log*",
            "deliveryStream": "yourfirehosedeliverystream"
        }
    ]
}
```

Untuk informasi lebih rinci tentang penggunaan agen dengan Amazon Kinesis Data Streams, lihat Menulis ke Amazon Kinesis Data Streams dengan Agen Kinesis.

Pra-proses data dengan Agen

Agen dapat memproses catatan yang diuraikan dari file yang dipantau sebelum mengirimnya ke aliran Firehose Anda. Anda dapat mengaktifkan fitur ini dengan menambahkan pengaturan konfigurasi `dataProcessingOptions` ke aliran file Anda. Satu atau lebih opsi pemrosesan dapat ditambahkan, dan dilakukan dalam urutan yang ditentukan.

Agen mendukung opsi pemrosesan berikut. Karena agen adalah sumber terbuka, Anda dapat lebih mengembangkan dan memperluas pilihan pemrosesannya. Anda dapat mengunduh agen dari Agen Kinesis.

Opsi Pemrosesan

`SINGLELINE`

Mengonversi catatan multi-baris untuk catatan baris tunggal dengan menghapus karakter baris baru, spasi di bagian paling depan, dan spasi di bagian paling belakang.

```
{
    "optionName": "SINGLELINE"
}
```

`CSVTOJSON`

Mengonversi catatan dari format yang dipisahkan pembatas ke format JSON.

```
{
    "optionName": "CSVTOJSON",
    "customFieldNames": [ "field1", "field2", ... ],
    "delimiter": "yourdelimiter"
}
```

`customFieldNames`

[Diperlukan] Nama-nama field yang digunakan sebagai kunci dalam setiap pasangan nilai kunci JSON. Misalnya, jika Anda menentukan `["f1", "f2"]`, catatan "v1, v2" dikonversi ke `{"f1":"v1","f2":"v2"}`.

`delimiter`

String yang digunakan sebagai pembatas dalam catatan. Default adalah koma (,).

`LOGTOJSON`

Mengonversi catatan dari format log ke format JSON. Format log yang didukung adalah Apache Common Log, Apache Combined Log, Apache Error Log, dan RFC3164 Syslog.

```
{
    "optionName": "LOGTOJSON",
    "logFormat": "logformat",
    "matchPattern": "yourregexpattern",
    "customFieldNames": [ "field1", "field2", … ]
}
```

`logFormat`

[Diperlukan] Format entri log. Berikut adalah nilai yang mungkin:

- `COMMONAPACHELOG` — Format Log Umum Apache. Setiap entri log memiliki pola berikut secara default: `"%{host} %{ident} %{authuser} [%{datetime}] \"%{request}\" %{response} %{bytes}"`.
- `COMBINEDAPACHELOG` — Format Log Gabungan Apache. Setiap entri log memiliki pola berikut secara default: `"%{host} %{ident} %{authuser} [%{datetime}] \"%{request}\" %{response} %{bytes} %{referrer} %{agent}"`.
- `APACHEERRORLOG` — Format Log Kesalahan Apache. Setiap entri log memiliki pola berikut secara default: `"[%{timestamp}] [%{module}:%{severity}] [pid %{processid}:tid %{threadid}] [client: %{client}] %{message}"`.
- `SYSLOG`— Format RFC3164 Syslog. Setiap entri log memiliki pola berikut secara default: `"%{timestamp} %{hostname} %{program}[%{processid}]: %{message}"`.

`matchPattern`

Menimpa pola default untuk format log tertentu. Gunakan pengaturan ini untuk mengekstraksi nilai dari entri log jika nilai menggunakan format khusus. Jika menentukan `matchPattern`, Anda juga harus menentukan `customFieldNames`.

`customFieldNames`

Nama bidang khusus digunakan sebagai kunci dalam setiap pasangan nilai kunci JSON. Anda dapat menggunakan pengaturan ini untuk menentukan nama bidang untuk nilai-nilai yang diekstraksi dari `matchPattern`, atau menimpa nama bidang default dari format log yang telah ditetapkan sebelumnya.

Example: Konfigurasi LOGTOJSON

Berikut adalah salah satu contoh konfigurasi `LOGTOJSON` untuk entri Log Umum Apache yang dikonversi ke format JSON:

```
{
    "optionName": "LOGTOJSON",
    "logFormat": "COMMONAPACHELOG"
}
```

Sebelum konversi:

```
64.242.88.10 - - [07/Mar/2004:16:10:02 -0800] "GET /mailman/listinfo/hsdivision
 HTTP/1.1" 200 6291
```

Setelah konversi:

```
{"host":"64.242.88.10","ident":null,"authuser":null,"datetime":"07/
Mar/2004:16:10:02 -0800","request":"GET /mailman/listinfo/hsdivision
 HTTP/1.1","response":"200","bytes":"6291"}
```

Example: Konfigurasi LOGTOJSON dengan Bidang Khusus

Berikut adalah contoh lain konfigurasi LOGTOJSON:

```
{
    "optionName": "LOGTOJSON",
    "logFormat": "COMMONAPACHELOG",
    "customFieldNames": ["f1", "f2", "f3", "f4", "f5", "f6", "f7"]
}
```

Dengan pengaturan konfigurasi ini, entri Log Umum Apache yang sama dari contoh sebelumnya dikonversi ke format JSON sebagai berikut:

```
{"f1":"64.242.88.10","f2":null,"f3":null,"f4":"07/Mar/2004:16:10:02 -0800","f5":"GET /
mailman/listinfo/hsdivision HTTP/1.1","f6":"200","f7":"6291"}
```

Example: Mengonversi Entri Log Umum Apache

Konfigurasi aliran berikut mengonversi entri Log Umum Apache ke catatan baris tunggal dalam format JSON:

```
{
    "flows": [
        {
            "filePattern": "/tmp/app.log*",
            "deliveryStream": "my-delivery-stream",
            "dataProcessingOptions": [
                {
                    "optionName": "LOGTOJSON",
                    "logFormat": "COMMONAPACHELOG"
```

```
                }
            ]
        }
    ]
}
```

Example: Mengonversi Catatan Multi-Baris

Konfigurasi aliran berikut mengurai catatan multi-baris yang baris pertamanya dimulai dengan "`[SEQUENCE=`". Setiap catatan dikonversi ke catatan baris tunggal terlebih dahulu. Kemudian, nilai-nilai diekstraksi dari catatan tersebut berdasarkan pembatas tab. Nilai yang diekstraksi dipetakan ke nilai `customFieldNames` yang ditentukan untuk membentuk catatan baris tunggal dalam format JSON.

```
{
    "flows": [
        {
            "filePattern": "/tmp/app.log*",
            "deliveryStream": "my-delivery-stream",
            "multiLineStartPattern": "\\[SEQUENCE=",
            "dataProcessingOptions": [
                {
                    "optionName": "SINGLELINE"
                },
                {
                    "optionName": "CSVTOJSON",
                    "customFieldNames": [ "field1", "field2", "field3" ],
                    "delimiter": "\\t"
                }
            ]
        }
    ]
}
```

Example: Konfigurasi LOGTOJSON dengan Pola Pencocokan

Berikut adalah salah satu contoh konfigurasi `LOGTOJSON` untuk entri Log Umum Apache yang dikonversi ke format JSON, dengan bidang terakhir (byte) dihilangkan:

```
{
    "optionName": "LOGTOJSON",
```

```
    "logFormat": "COMMONAPACHELOG",
    "matchPattern": "^([\\d.]+) (\\S+) (\\S+) \\[([\\w:/]+\\s[+\\-]\\d{4})\\] \"(.
+?)\" (\\d{3})",
    "customFieldNames": ["host", "ident", "authuser", "datetime", "request",
 "response"]
}
```

Sebelum konversi:

```
123.45.67.89 - - [27/Oct/2000:09:27:09 -0400] "GET /java/javaResources.html HTTP/1.0"
 200
```

Setelah konversi:

```
{"host":"123.45.67.89","ident":null,"authuser":null,"datetime":"27/Oct/2000:09:27:09
 -0400","request":"GET /java/javaResources.html HTTP/1.0","response":"200"}
```

Gunakan perintah Agen CLI yang umum

Tabel berikut menyediakan satu set kasus penggunaan umum dan perintah yang sesuai untuk bekerja dengan agen AWS Kinesis.

Kasus penggunaan	Perintah
Secara otomatis memulai agen pada sistem start up	**sudo chkconfig aws-kinesis-agent on**
Periksa status agen	**sudo service aws-kinesis-agent status**
Hentikan agen	**sudo service aws-kinesis-agent stop**
Baca file log agen dari lokasi ini	**/var/log/aws-kinesis-agent/aws-kinesis-agent.log**
Copot instalan agen	**sudo yum remove aws-kinesis-agent**

Memecahkan masalah saat mengirim dari Agen Kinesis

Tabel ini menyediakan informasi pemecahan masalah dan solusi untuk masalah umum yang dihadapi saat menggunakan Agen Amazon Kinesis.

Isu	Solusi
Mengapa Agen Kinesis tidak berfungsi di Windows?	Kinesis Agent untuk Windows adalah perangkat lunak yang berbeda dari Kinesis Agent untuk platform Linux.
Mengapa Agen Kinesis melambat dan/atau meningkat? `RecordSendErrors`	Ini biasanya karena pelambatan dari Kinesis. Periksa `WriteProvisionedThroughputExceeded` metrik untuk Kinesis Data Streams `ThrottledRecords` atau metrik untuk aliran Firehose. Setiap peningkatan dari 0 dalam metrik ini menunjukkan bahwa batas aliran perlu ditingkatkan. Untuk informasi selengkapnya, lihat batas Kinesis Data Stream dan aliran Firehose. Setelah Anda mengesampingkan pembatasan, lihat apakah Agen Kinesis dikonfigurasi untuk mengekor sejumlah besar file kecil. Ada penundaan ketika Agen Kinesis mengekor file baru, jadi Agen Kinesis harus membuntuti sejumlah kecil file yang lebih besar. Coba konsolidasikan file log Anda ke file yang lebih besar.
Bagaimana cara mengatasi `java.lang.OutOfMemoryError` pengecualian?	Ini terjadi ketika Agen Kinesis tidak memiliki cukup memori untuk menangani beban kerjanya saat ini. Cobalah meningkatkan `JAVA_START_HEAP` dan `JAVA_MAX_HEAP` masuk `/usr/bin/start-aws-kinesis-agent` dan memulai kembali agen.
Bagaimana cara mengatasi `IllegalStateException : connection pool shut down` pengecualian?	Agen Kinesis tidak memiliki koneksi yang cukup untuk menangani beban kerjanya saat ini. Coba tingkatkan `maxConnections` dan `maxSendingThreads` dalam pengaturan konfigurasi agen umum Anda di`/etc/aws-kinesis/agent.json` . Nilai default untuk bidang ini adalah 12 kali prosesor runtime yang tersedia. Lihat

Isu	Solusi
	AgentConfiguration.java untuk mengetahui selengkapnya tentang pengaturan konfigurasi agen lanjutan.
Bagaimana saya bisa men-debug masalah lain dengan Agen Kinesis?	`DEBUG`log level dapat diaktifkan di`/etc/aws-kinesis/log4j.xml` .
Bagaimana cara mengonfigurasi Agen Kinesis?	Semakin kecil`maxBufferSizeBytes` , semakin sering Agen Kinesis akan mengirim data. Ini bisa bagus karena mengurangi waktu pengiriman catatan, tetapi juga meningkatkan permintaan per detik ke Kinesis.
Mengapa Agen Kinesis mengirimkan catatan duplikat?	Ini terjadi karena kesalahan konfigurasi dalam file tailing. Pastikan masing-masing hanya `fileFlow's filePattern` cocok dengan satu file. Ini juga dapat terjadi jika `logrotate` mode yang digunakan dalam `copytruncate` mode. Coba ubah mode ke mode default atau buat untuk menghindari duplikasi. Untuk informasi selengkapnya tentang penanganan rekaman duplikat, lihat Menangani Rekaman Duplikat.

Kirim data dengan AWS SDK

Anda dapat menggunakan Amazon Data Firehose API untuk mengirim data ke aliran Firehose menggunakan SDK for AWS Java, .NET, Node.js, Python, atau Ruby. Jika Anda baru mengenal Amazon Data Firehose, luangkan waktu untuk membiasakan diri dengan konsep dan terminologi yang disajikan. Apa itu Amazon Data Firehose? Untuk informasi selengkapnya, lihat Mulai Pengembangan dengan Amazon Web Services.

Contoh-contoh ini tidak mewakili kode siap produksi, karena contoh ini tidak memeriksa semua kemungkinan pengecualian, atau memperhitungkan semua kemungkinan pertimbangan keamanan atau performa.

Amazon Data Firehose API menawarkan dua operasi untuk mengirim data ke aliran Firehose Anda: dan. PutRecordPutRecordBatch `PutRecord()`mengirim satu catatan data dalam satu panggilan dan `PutRecordBatch()` dapat mengirim beberapa catatan data dalam satu panggilan.

Operasi penulisan tunggal menggunakan PutRecord

Menempatkan data hanya membutuhkan nama aliran Firehose dan buffer byte (<=1000 KB). Karena Amazon Data Firehose mengumpulkan beberapa catatan sebelum memuat file ke Amazon S3, Anda mungkin ingin menambahkan pemisah rekaman. Untuk memasukkan data satu rekaman pada satu waktu ke aliran Firehose, gunakan kode berikut:

```
PutRecordRequest putRecordRequest = new PutRecordRequest();
putRecordRequest.setDeliveryStreamName(deliveryStreamName);

String data = line + "\n";

Record record = new Record().withData(ByteBuffer.wrap(data.getBytes()));
putRecordRequest.setRecord(record);

// Put record into the DeliveryStream
firehoseClient.putRecord(putRecordRequest);
```

Untuk konteks kode lainnya, lihat kode contoh yang disertakan dalam AWS SDK. Untuk informasi tentang sintaks permintaan dan respons, lihat topik yang relevan di Operasi API Firehose.

Operasi penulisan Batch menggunakan PutRecordBatch

Menempatkan data hanya memerlukan nama aliran Firehose dan daftar catatan. Karena Amazon Data Firehose mengumpulkan beberapa catatan sebelum memuat file ke Amazon S3, Anda mungkin ingin menambahkan pemisah rekaman. Untuk menempatkan catatan data dalam batch ke dalam aliran Firehose, gunakan kode berikut:

```
PutRecordBatchRequest putRecordBatchRequest = new PutRecordBatchRequest();
putRecordBatchRequest.setDeliveryStreamName(deliveryStreamName);
putRecordBatchRequest.setRecords(recordList);

// Put Record Batch records. Max No.Of Records we can put in a
// single put record batch request is 500
firehoseClient.putRecordBatch(putRecordBatchRequest);

recordList.clear();
```

Untuk konteks kode lainnya, lihat kode contoh yang disertakan dalam AWS SDK. Untuk informasi tentang sintaks permintaan dan respons, lihat topik yang relevan di Operasi API Firehose.

Kirim CloudWatch Log ke Firehose

CloudWatch Peristiwa log dapat dikirim ke Firehose menggunakan filter CloudWatch berlangganan. Untuk informasi selengkapnya, lihat Filter langganan dengan Amazon Data Firehose.

CloudWatch Peristiwa log dikirim ke Firehose dalam format gzip terkompresi. Jika Anda ingin mengirimkan peristiwa log yang didekompresi ke tujuan Firehose, Anda dapat menggunakan fitur dekompresi di Firehose untuk mendekompresi Log secara otomatis. CloudWatch

> **Important**
>
> Saat ini, Firehose tidak mendukung pengiriman CloudWatch Log ke tujuan OpenSearch Layanan Amazon karena Amazon CloudWatch menggabungkan beberapa peristiwa log ke dalam satu catatan Firehose dan OpenSearch Layanan Amazon tidak dapat menerima beberapa peristiwa log dalam satu catatan. Sebagai alternatif, Anda dapat mempertimbangkan Menggunakan filter langganan untuk OpenSearch Layanan Amazon di CloudWatch Log.

Dekompresi CloudWatch Log

Jika Anda menggunakan Firehose untuk mengirimkan CloudWatch Log dan ingin mengirimkan data yang didekompresi ke tujuan aliran Firehose Anda, gunakan Konversi Format Data Firehose (Parket, ORC) atau partisi Dinamis. Anda harus mengaktifkan dekompresi untuk aliran Firehose Anda.

Anda dapat mengaktifkan dekompresi menggunakan Konsol Manajemen AWS, AWS Command Line Interface atau AWS SDKs.

> **Note**
>
> Jika Anda mengaktifkan fitur dekompresi pada stream, gunakan streaming tersebut secara eksklusif untuk filter langganan CloudWatch Log, dan bukan untuk Vded Logs. Jika Anda mengaktifkan fitur dekompresi pada aliran yang digunakan untuk menelan CloudWatch Log dan Log Penjual, konsumsi Log Penjual ke Firehose gagal. Fitur dekompresi ini hanya untuk CloudWatch Log.

Ekstrak pesan setelah dekompresi Log CloudWatch

Saat Anda mengaktifkan dekompresi, Anda memiliki opsi untuk juga mengaktifkan ekstraksi pesan. Saat menggunakan ekstraksi pesan, Firehose menyaring semua metadata, seperti pemilik, loggroup, logstream, dan lainnya dari catatan CloudWatch Log yang didekompresi dan hanya mengirimkan konten di dalam bidang pesan. Jika Anda mengirimkan data ke tujuan Splunk, Anda harus mengaktifkan ekstraksi pesan untuk Splunk untuk mengurai data. Berikut ini adalah output sampel setelah dekompresi dengan dan tanpa ekstraksi pesan.

Gambar 1: Output sampel setelah dekompresi tanpa ekstraksi pesan:

```
{
 "owner": "111111111111",
 "logGroup": "CloudTrail/logs",
 "logStream": "111111111111_CloudTrail/logs_us-east-1",
 "subscriptionFilters": [
 "Destination"
 ],
 "messageType": "DATA_MESSAGE",
 "logEvents": [
 {
 "id": "31953106606966983378809025079804211143289615424298221568",
 "timestamp": 1432826855000,
 "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\"Root1\"}"
 },
 {
 "id": "31953106606966983378809025079804211143289615424298221569",
 "timestamp": 1432826855000,
 "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\"Root2\"}"
 },
 {
 "id": "31953106606966983378809025079804211143289615424298221570",
 "timestamp": 1432826855000,
 "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\"Root3\"}"
 }
 ]
}
```

Gambar 2: Output sampel setelah dekompresi dengan ekstraksi pesan:

```
{"eventVersion":"1.03","userIdentity":{"type":"Root1"}
{"eventVersion":"1.03","userIdentity":{"type":"Root2"}
```

```
{"eventVersion":"1.03","userIdentity":{"type":"Root3"}
```

Aktifkan dekompresi pada aliran Firehose baru dari konsol

Untuk mengaktifkan dekompresi pada aliran Firehose baru menggunakan Konsol Manajemen AWS

1. Masuk ke Konsol Manajemen AWS dan buka konsol Kinesis di /kinesis. https://console.aws.amazon.com
2. Pilih Amazon Data Firehose di panel navigasi.
3. Pilih Buat aliran Firehose.
4. Di bawah Pilih sumber dan tujuan

   Sumber

   Sumber aliran Firehose Anda. Pilih salah satu sumber berikut:

   - Direct PUT - Pilih opsi ini untuk membuat aliran Firehose yang ditulis langsung oleh aplikasi produser. Untuk daftar AWS layanan dan agen serta layanan open source yang terintegrasi dengan Direct PUT di Firehose, lihat bagian ini.
   - Aliran Kinesis: Pilih opsi ini untuk mengonfigurasi aliran Firehose yang menggunakan aliran data Kinesis sebagai sumber data. Anda kemudian dapat menggunakan Firehose untuk membaca data dengan mudah dari aliran data Kinesis yang ada dan memuatnya ke tujuan. Untuk informasi selengkapnya, lihat Menulis ke Firehose Menggunakan Kinesis Data Streams

   Destinasi

   Tujuan aliran Firehose Anda. Pilih salah satu cara berikut:

   - Amazon S3
   - Splunk
5. Di bawah nama aliran Firehose, masukkan nama untuk streaming Anda.
6. (Opsional) Di bawah Transform record:

   - Di bagian Decompress source records from Amazon CloudWatch Logs, pilih Aktifkan dekompresi.
   - Jika Anda ingin menggunakan ekstraksi pesan setelah dekompresi, pilih Aktifkan ekstraksi pesan.

Aktifkan dekompresi pada aliran Firehose yang ada

Bagian ini memberikan instruksi untuk mengaktifkan dekompresi pada aliran Firehose yang ada. Ini mencakup dua skenario - aliran dengan pemrosesan Lambda dinonaktifkan dan aliran dengan pemrosesan Lambda sudah diaktifkan. Bagian berikut menguraikan step-by-step prosedur untuk setiap kasus, termasuk pembuatan atau modifikasi fungsi Lambda, memperbarui pengaturan Firehose, dan metrik CloudWatch pemantauan untuk memastikan keberhasilan implementasi fitur dekompresi Firehose bawaan.

Mengaktifkan dekompresi saat pemrosesan Lambda dinonaktifkan

Untuk mengaktifkan dekompresi pada aliran Firehose yang ada dengan pemrosesan Lambda dinonaktifkan, Anda harus mengaktifkan pemrosesan Lambda terlebih dahulu. Kondisi ini hanya berlaku untuk aliran yang ada. Langkah-langkah berikut menunjukkan cara mengaktifkan dekompresi pada aliran yang ada yang tidak mengaktifkan pemrosesan Lambda.

1. Buat fungsi Lambda. Anda dapat membuat catatan dummy pass through atau dapat menggunakan cetak biru ini untuk membuat fungsi Lambda baru.
2. Perbarui aliran Firehose Anda saat ini untuk mengaktifkan pemrosesan Lambda dan menggunakan fungsi Lambda yang Anda buat untuk diproses.
3. Setelah Anda memperbarui aliran dengan fungsi Lambda baru, kembali ke konsol Firehose dan aktifkan dekompresi.
4. Nonaktifkan pemrosesan Lambda yang Anda aktifkan di langkah 1. Anda sekarang dapat menghapus fungsi yang Anda buat di langkah 1.

Mengaktifkan dekompresi saat pemrosesan Lambda diaktifkan

Jika Anda sudah memiliki aliran Firehose dengan fungsi Lambda, untuk melakukan dekompresi Anda dapat menggantinya dengan fitur dekompresi Firehose. Sebelum Anda melanjutkan, tinjau kode fungsi Lambda Anda untuk mengonfirmasi bahwa kode tersebut hanya melakukan dekompresi atau ekstraksi pesan. Output dari fungsi Lambda Anda akan terlihat mirip dengan contoh yang ditunjukkan pada Gambar 1 atau Gambar 2. Jika output terlihat serupa, Anda dapat mengganti fungsi Lambda menggunakan langkah-langkah berikut.

1. Ganti fungsi Lambda Anda saat ini dengan cetak biru ini. Fungsi Lambda cetak biru baru secara otomatis mendeteksi apakah data yang masuk dikompresi atau didekompresi. Ini hanya melakukan dekompresi jika data inputnya dikompresi.

2. Nyalakan dekompresi menggunakan opsi Firehose bawaan untuk dekompresi.
3. Aktifkan CloudWatch metrik untuk aliran Firehose Anda jika belum diaktifkan. Pantau metrik `CloudWatchProcessorLambda_IncomingCompressedData` dan tunggu hingga metrik ini berubah menjadi nol. Ini mengonfirmasi bahwa semua data input yang dikirim ke fungsi Lambda Anda didekompresi dan fungsi Lambda tidak lagi diperlukan.
4. Hapus transformasi data Lambda karena Anda tidak lagi membutuhkannya untuk mendekompresi aliran Anda.

Nonaktifkan dekompresi pada aliran Firehose

Untuk menonaktifkan dekompresi pada aliran data menggunakan Konsol Manajemen AWS

1. Masuk ke Konsol Manajemen AWS dan buka konsol Kinesis di /kinesis. https://console.aws.amazon.com
2. Pilih Amazon Data Firehose di panel navigasi.
3. Pilih aliran Firehose yang ingin Anda edit.
4. Pada halaman rincian aliran Firehose, pilih tab Konfigurasi.
5. Di bagian Transform and convert records, pilih Edit.
6. Di bawah Dekompresi catatan sumber dari Amazon CloudWatch Logs, hapus Aktifkan dekompresi, lalu pilih Simpan perubahan.

Memecahkan masalah dekompresi di Firehose

Tabel berikut menunjukkan cara Firehose menangani kesalahan selama dekompresi dan pemrosesan data, termasuk mengirimkan catatan ke bucket S3 kesalahan, kesalahan pencatatan, dan metrik pemancar. Ini juga menjelaskan pesan kesalahan yang dikembalikan untuk operasi penempatan data yang tidak sah.

Isu	Solusi
Apa yang terjadi pada data sumber jika terjadi kesalahan selama dekompresi?	Jika Amazon Data Firehose tidak dapat mendekomp resi rekaman, rekaman akan dikirimkan sebagaima na adanya (dalam format terkompresi) ke bucket S3 error yang Anda tentukan selama waktu pembuatan

Isu	Solusi
	aliran Firehose. Seiring dengan catatan, objek yang dikirimkan juga menyertakan kode kesalahan dan pesan kesalahan dan objek ini akan dikirim ke awalan bucket S3 yang disebut. `decompression-failed` Firehose akan terus memproses catatan lain setelah dekompresi rekaman yang gagal.
Apa yang terjadi pada data sumber jika terjadi kesalahan dalam pipa pemrosesan setelah dekompresi berhasil?	Jika Amazon Data Firehose terjadi kesalahan dalam langkah pemrosesan setelah dekompresi seperti Partisi Dinamis dan Konversi Format Data, rekaman akan dikirimkan dalam format terkompresi ke bucket S3 kesalahan yang Anda tentukan selama waktu pembuatan aliran Firehose. Seiring dengan catatan, objek yang dikirim juga mencakup kode kesalahan dan pesan kesalahan.
Bagaimana Anda diberitahu jika terjadi kesalahan atau pengecual ian?	Jika terjadi kesalahan atau pengecualian selama dekompresi, jika Anda mengonfigurasi CloudWatch Log, Firehose akan mencatat pesan CloudWatch kesalahan ke dalam Log. Selain itu, Firehose mengirimkan metrik ke CloudWatch metrik yang dapat Anda pantau. Anda juga dapat secara opsional membuat alarm berdasarkan metrik yang dipancarkan oleh Firehose.
Apa yang terjadi jika `put` operasi tidak berasal dari CloudWatch Log?	Ketika pelanggan `puts` tidak berasal dari CloudWatch Log, maka pesan kesalahan berikut dikembalikan: `Put to Firehose failed for AccountId: <accountID>, FirehoseName: <firehosename> because the request is not originating from allowed source types.`

Isu	Solusi
Metrik apa yang dipancarkan Firehose untuk fitur dekompresi?	Firehose memancarkan metrik untuk dekompresi setiap catatan. Anda harus memilih periode (1 menit), statistik (jumlah), rentang tanggal untuk mendapatkan jumlah `DecompressedRecords` gagal atau berhasil atau `DecompressedBytes` gagal atau berhasil. Lihat informasi yang lebih lengkap di CloudWatch Metrik Dekompresi Log.

Kirim CloudWatch Acara ke Firehose

Anda dapat mengonfigurasi Amazon CloudWatch untuk mengirim peristiwa ke aliran Firehose dengan menambahkan target ke aturan CloudWatch Acara.

Untuk membuat target untuk aturan CloudWatch Acara yang mengirimkan peristiwa ke aliran Firehose yang ada

1. Masuk ke Konsol Manajemen AWS dan buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pilih Buat aturan.
3. Pada halaman Langkah 1: Buat aturan, untuk Target, pilih Tambah target, lalu pilih Aliran Firehose.
4. Pilih aliran Firehose yang ada.

Untuk informasi selengkapnya tentang membuat aturan CloudWatch Acara, lihat Memulai CloudWatch Acara Amazon.

Konfigurasi AWS IoT untuk mengirim data ke Firehose

Anda dapat mengonfigurasi AWS IoT untuk mengirim informasi ke aliran Firehose dengan menambahkan tindakan.

Untuk membuat tindakan yang mengirimkan peristiwa ke aliran Firehose yang ada

1. Saat membuat aturan di AWS IoT konsol, pada halaman Buat aturan, di bawah Tetapkan satu atau beberapa tindakan, pilih Tambah tindakan.
2. Pilih Kirim pesan ke aliran Amazon Kinesis Firehose.
3. Pilih Konfigurasi tindakan.
4. Untuk nama Stream, pilih aliran Firehose yang ada.
5. Untuk Pemisah, pilih karakter pemisah untuk dimasukkan di antara tiap catatan.
6. Untuk Nama IAM role, pilih IAM role yang ada atau pilih Buat peran baru.
7. Pilih Tambahkan tindakan.

Untuk informasi selengkapnya tentang membuat AWS IoT aturan, lihat Tutorial Aturan AWS IoT.

Mengubah data sumber di Amazon Data Firehose

Amazon Data Firehose dapat menjalankan fungsi Lambda Anda untuk mengubah data sumber yang masuk dan mengirimkan data yang diubah ke tujuan. Anda dapat mengaktifkan transformasi data Amazon Data Firehose saat membuat aliran Firehose.

Memahami aliran transformasi data

Saat Anda mengaktifkan transformasi data Firehose, Firehose menyangga data yang masuk. Petunjuk ukuran buffering berkisar antara 0,2 MB dan 3MB. Petunjuk ukuran buffering Lambda default adalah 1 MB untuk semua tujuan, kecuali Splunk dan Snowflake. Untuk Splunk dan Snowflake, petunjuk buffering default adalah 256 KB. Petunjuk interval buffering Lambda berkisar antara 0 dan 900 detik. Petunjuk interval buffering Lambda default adalah enam puluh detik untuk semua tujuan kecuali Snowflake. Untuk Snowflake, interval petunjuk buffering default adalah 30 detik. Untuk menyesuaikan ukuran buffering, atur ProcessingConfigurationparameter CreateDeliveryStreamatau UpdateDestinationAPI dengan yang ProcessorParameterdipanggil `BufferSizeInMBs` dan`IntervalInSeconds`. Firehose kemudian memanggil fungsi Lambda yang ditentukan secara sinkron dengan setiap batch buffer menggunakan mode pemanggilan sinkron. AWS Lambda Data yang ditransformasikan dikirim dari Lambda ke Firehose. Firehose kemudian mengirimkannya ke tujuan ketika ukuran buffering tujuan yang ditentukan atau interval buffering tercapai, mana yang terjadi lebih dulu.

> **Important**
>
> Mode pemanggilan sinkron Lambda memiliki batas ukuran muatan 6 MB untuk permintaan dan respons. Pastikan bahwa ukuran buffering Anda untuk mengirim permintaan ke fungsi tersebut kurang dari atau sama dengan 6 MB. Juga pastikan bahwa respons yang dihasilkan fungsi tidak melebihi 6 MB.

Durasi pemanggilan Lambda

Amazon Data Firehose mendukung waktu pemanggilan Lambda hingga 5 menit. Jika fungsi Lambda membutuhkan waktu lebih dari 5 menit untuk diselesaikan, Anda mendapatkan kesalahan berikut: Firehose mengalami kesalahan batas waktu saat memanggil Lambda. AWS Waktu habis fungsi maksimum yang didukung adalah 5 menit.

Untuk informasi tentang apa yang Amazon Data Firehose lakukan jika terjadi kesalahan seperti itu, lihat. the section called “Menangani kegagalan dalam transformasi data”

Parameter yang diperlukan untuk transformasi data

Semua catatan yang diubah dari Lambda harus berisi parameter berikut, atau Amazon Data Firehose menolaknya dan memperlakukannya sebagai kegagalan transformasi data.

For Kinesis Data Streams and Direct PUT

Parameter berikut diperlukan untuk semua catatan yang diubah dari Lambda.

- `recordId`— ID rekaman diteruskan dari Amazon Data Firehose ke Lambda selama pemanggilan. Catatan yang ditransformasi harus berisi ID catatan yang sama. Ketidakcocokan apa pun antara ID catatan asli dan ID catatan yang ditransformasi dianggap sebagai kegagalan transformasi data.
- `result`— Status transformasi data catatan. Kemungkinan nilainya adalah: `Ok` (catatan berhasil ditransformasi), `Dropped` (catatan dihentikan dengan sengaja oleh logika pemrosesan Anda), dan `ProcessingFailed` (catatan tidak bisa ditransformasi). Jika rekaman memiliki status `Ok` atau`Dropped`, Amazon Data Firehose menganggapnya berhasil diproses. Jika tidak, Amazon Data Firehose menganggapnya tidak berhasil diproses.
- `data`— Payload data yang diubah, setelah pengkodean base64.

  Berikut ini adalah contoh hasil keluaran Lambda:

```
 {
    "recordId": "<recordId from the Lambda input>",
    "result": "Ok",
    "data": "<Base64 encoded Transformed data>"
}
```

For Amazon MSK

Parameter berikut diperlukan untuk semua catatan yang diubah dari Lambda.

- `recordId`— ID rekaman diteruskan dari Firehose ke Lambda selama pemanggilan. Catatan yang ditransformasi harus berisi ID catatan yang sama. Ketidakcocokan apa pun antara ID catatan asli dan ID catatan yang ditransformasi dianggap sebagai kegagalan transformasi data.

- `result`— Status transformasi data catatan. Kemungkinan nilainya adalah: `Ok` (catatan berhasil ditransformasi), `Dropped` (catatan dihentikan dengan sengaja oleh logika pemrosesan Anda), dan `ProcessingFailed` (catatan tidak bisa ditransformasi). Jika rekaman memiliki status `Ok` atau`Dropped`, Firehose menganggapnya berhasil diproses. Jika tidak, Firehose menganggapnya tidak berhasil diproses.
- `KafkaRecordValue`— Payload data yang diubah, setelah pengkodean base64.

  Berikut ini adalah contoh hasil keluaran Lambda:

```
 {
    "recordId": "<recordId from the Lambda input>",
    "result": "Ok",
    "kafkaRecordValue": "<Base64 encoded Transformed data>"
}
```

Cetak biru Lambda yang didukung

Cetak biru ini menunjukkan bagaimana Anda dapat membuat dan menggunakan fungsi AWS Lambda untuk mengubah data dalam aliran data Amazon Data Firehose Anda.

Untuk melihat cetak biru yang tersedia di konsol AWS Lambda

1. Masuk ke Konsol Manajemen AWS dan buka AWS Lambda konsol di https://console.aws.amazon.com/lambda/.
2. Pilih Buat fungsi, lalu pilih Gunakan cetak biru.
3. Di bidang Cetak Biru, cari kata kunci **`firehose`** untuk menemukan cetak biru Amazon Data Firehose Lambda.

Daftar cetak biru:

- Memproses catatan dikirim ke Amazon Data Firehose stream (Node.js, Python)

  Cetak biru ini menunjukkan contoh dasar cara memproses data dalam aliran data Firehose Anda menggunakan Lambda. AWS

  Tanggal rilis terbaru: November, 2016.

  Catatan rilis: tidak ada.

- CloudWatch Log Proses dikirim ke Firehose

  Cetak biru ini sudah usang. Jangan gunakan cetak biru ini. Ini mungkin dikenakan biaya tinggi ketika data CloudWatch Log didekompresi lebih dari 6MB (batas Lambda). Untuk informasi tentang pemrosesan CloudWatch Log yang dikirim ke Firehose, lihat Menulis ke Firehose Menggunakan Log. CloudWatch
- Mengonversi catatan aliran Amazon Data Firehose dalam format syslog ke JSON (Node.js)

  Cetak biru ini menunjukkan bagaimana Anda dapat mengonversi catatan input dalam format RFC3164 Syslog ke JSON.

  Tanggal rilis terbaru: Nov, 2016.

  Catatan rilis: tidak ada.

Untuk melihat cetak biru yang tersedia di AWS Serverless Application Repository

1. Kunjungi AWS Serverless Application Repository.
2. Pilih Jelajahi semua aplikasi.
3. Di bidang Aplikasi, cari dengan kata kunci `firehose`.

Anda juga dapat membuat fungsi Lambda tanpa menggunakan cetak biru. Lihat Memulai dengan AWS Lambda.

Menangani kegagalan dalam transformasi data

Jika pemanggilan fungsi Lambda Anda gagal karena batas waktu jaringan atau karena Anda telah mencapai batas pemanggilan Lambda, Amazon Data Firehose akan mencoba ulang pemanggilan tiga kali secara default. Jika pemanggilan tidak berhasil, Amazon Data Firehose kemudian melewatkan kumpulan catatan itu. Catatan yang dilewati dianggap sebagai catatan yang tidak berhasil diproses. Anda dapat menentukan atau mengganti opsi coba lagi menggunakan API CreateDeliveryStreamatau`UpdateDestination`. Untuk jenis kegagalan ini, Anda dapat mencatat kesalahan pemanggilan ke Amazon CloudWatch Logs. Untuk informasi selengkapnya, lihat Pantau Amazon Data Firehose Menggunakan Log CloudWatch.

Jika status transformasi data rekaman adalah`ProcessingFailed`, Amazon Data Firehose memperlakukan catatan sebagai tidak berhasil diproses. Untuk jenis kegagalan ini, Anda dapat

memancarkan log kesalahan ke Amazon CloudWatch Logs dari fungsi Lambda Anda. Untuk informasi selengkapnya, lihat Mengakses CloudWatch Log Amazon AWS Lambda di Panduan AWS Lambda Pengembang.

Jika transformasi data gagal, catatan yang tidak berhasil diproses akan dikirimkan ke bucket S3 Anda di `processing-failed` folder. Catatan tersebut memiliki format berikut:

```
{
    "attemptsMade": "count",
    "arrivalTimestamp": "timestamp",
    "errorCode": "code",
    "errorMessage": "message",
    "attemptEndingTimestamp": "timestamp",
    "rawData": "data",
    "lambdaArn": "arn"
}
```

`attemptsMade`

Jumlah permintaan pemanggilan yang dilakukan.

`arrivalTimestamp`

Waktu rekaman itu diterima oleh Amazon Data Firehose.

`errorCode`

Kode kesalahan HTTP dihasilkan oleh Lambda.

`errorMessage`

Pesan kesalahan yang dihasilkan oleh Lambda.

`attemptEndingTimestamp`

Waktu Amazon Data Firehose berhenti mencoba pemanggilan Lambda.

`rawData`

Data catatan berenkode base64.

`lambdaArn`

Amazon Resource Name (ARN) fungsi Lambda.

Cadangkan catatan sumber

Amazon Data Firehose dapat mencadangkan semua catatan yang belum diubah ke bucket S3 Anda secara bersamaan sambil mengirimkan catatan yang diubah ke tujuan. Anda dapat mengaktifkan cadangan catatan sumber saat membuat atau memperbarui aliran Firehose. Anda tidak dapat menonaktifkan pencadangan catatan sumber setelah Anda mengaktifkannya.

Partisi streaming data di Amazon Data Firehose

Partisi dinamis memungkinkan Anda untuk terus mempartisi data streaming di Firehose dengan menggunakan kunci dalam data (misalnya, `customer_id` atau`transaction_id`) dan kemudian mengirimkan data yang dikelompokkan berdasarkan kunci ini ke awalan Amazon Simple Storage Service (Amazon S3) yang sesuai. Ini membuatnya lebih mudah untuk menjalankan analitik berkinerja tinggi dan hemat biaya pada data streaming di Amazon S3 menggunakan berbagai layanan seperti Amazon Athena, Amazon EMR, Amazon Redshift Spectrum, dan Amazon. QuickSight Selain itu, AWS Glue dapat melakukan pekerjaan ekstrak, transformasi, dan pemuatan (ETL) yang lebih canggih setelah data streaming yang dipartisi secara dinamis dikirim ke Amazon S3, dalam kasus penggunaan di mana pemrosesan tambahan diperlukan.

Mempartisi data Anda meminimalkan jumlah data yang dipindai, mengoptimalkan kinerja, dan mengurangi biaya kueri analitik Anda di Amazon S3. Ini juga meningkatkan akses granular ke data Anda. Aliran Firehose secara tradisional digunakan untuk menangkap dan memuat data ke Amazon S3. Untuk mempartisi kumpulan data streaming untuk S3-based analitik Amazon, Anda perlu menjalankan aplikasi partisi antara bucket Amazon S3 sebelum membuat data tersedia untuk analisis, yang bisa menjadi rumit atau mahal.

Dengan partisi dinamis, Firehose terus mengelompokkan data dalam transit menggunakan kunci data yang ditentukan secara dinamis atau statis, dan mengirimkan data ke awalan Amazon S3 individual berdasarkan kunci. Ini mengurangi waktu-ke-wawasan dalam hitungan menit atau jam. Ini juga mengurangi biaya dan menyederhanakan arsitektur.

Topik

Aktifkan partisi dinamis di Amazon Data Firehose

Anda dapat mengonfigurasi partisi dinamis untuk aliran Firehose melalui Amazon Data Firehose Management Console, CLI, atau API.

> **Important**
>
> Anda hanya dapat mengaktifkan partisi dinamis saat membuat aliran Firehose baru. Anda tidak dapat mengaktifkan partisi dinamis untuk aliran Firehose yang sudah ada yang belum mengaktifkan partisi dinamis.

Untuk langkah-langkah mendetail tentang cara mengaktifkan dan mengonfigurasi partisi dinamis melalui konsol manajemen Firehose saat membuat aliran Firehose baru, lihat Membuat aliran Amazon Firehose. Saat Anda sampai pada tugas menentukan tujuan untuk aliran Firehose Anda, pastikan untuk mengikuti langkah-langkah di Konfigurasikan pengaturan tujuan bagian tersebut, karena saat ini, partisi dinamis hanya didukung untuk aliran Firehose yang menggunakan Amazon S3 sebagai tujuan.

Setelah partisi dinamis pada aliran Firehose aktif diaktifkan, Anda dapat memperbarui konfigurasi dengan menambahkan kunci partisi baru atau menghapus atau memperbarui kunci partisi yang ada dan ekspresi awalan S3. Setelah diperbarui, Firehose mulai menggunakan kunci baru dan ekspresi awalan S3 baru.

> **Important**
>
> Setelah Anda mengaktifkan partisi dinamis pada aliran Firehose, partisi tidak dapat dinonaktifkan di aliran Firehose ini.

Memahami kunci partisi

Dengan partisi dinamis, Anda membuat kumpulan data yang ditargetkan dari data streaming S3 dengan mempartisi data berdasarkan kunci partisi. Tombol partisi memungkinkan Anda memfilter data streaming berdasarkan nilai tertentu. Misalnya, jika Anda perlu memfilter data berdasarkan ID pelanggan dan negara, Anda dapat menentukan bidang data `customer_id` sebagai satu kunci partisi dan bidang data `country` sebagai kunci partisi lainnya. Kemudian, Anda menentukan ekspresi (menggunakan format yang didukung) untuk menentukan awalan bucket S3 yang akan dikirimkan rekaman data yang dipartisi secara dinamis.

Anda dapat membuat kunci partisi dengan metode berikut.

- Parsing inline - metode ini menggunakan mekanisme dukungan built-in Firehose, parser jq, untuk mengekstrak kunci untuk partisi dari catatan data yang dalam format JSON. Saat ini, kami hanya mendukung `jq 1.6` versi.
- AWS Fungsi Lambda — metode ini menggunakan AWS Lambda fungsi tertentu untuk mengekstrak dan mengembalikan bidang data yang diperlukan untuk partisi.

> **Important**
>
> Saat Anda mengaktifkan partisi dinamis, Anda harus mengonfigurasi setidaknya satu dari metode ini untuk mempartisi data Anda. Anda dapat mengonfigurasi salah satu metode ini untuk menentukan kunci partisi Anda atau keduanya secara bersamaan.

Buat kunci partisi dengan penguraian sebaris

Untuk mengonfigurasi penguraian sebaris sebagai metode partisi dinamis untuk data streaming Anda, Anda harus memilih parameter catatan data yang akan digunakan sebagai kunci partisi dan memberikan nilai untuk setiap kunci partisi yang ditentukan.

Catatan data sampel berikut menunjukkan bagaimana Anda dapat menentukan kunci partisi untuk itu dengan parsing inline. Perhatikan bahwa data harus dikodekan dalam format Base64. Anda juga dapat merujuk ke contoh CLI.

```
{
   "type": {
    "device": "mobile",
    "event": "user_clicked_submit_button"
  },
  "customer_id": "1234567890",
  "event_timestamp": 1565382027,   #epoch timestamp
  "region": "sample_region"
}
```

Misalnya, Anda dapat memilih untuk mempartisi data Anda berdasarkan `customer_id` parameter atau `event_timestamp` parameter. Ini berarti Anda ingin nilai `customer_id` parameter atau `event_timestamp` parameter di setiap catatan digunakan dalam menentukan awalan S3 yang akan dikirimkan catatan. Anda juga dapat memilih parameter bersarang, seperti `device` dengan ekspresi `.type.device`. Logika partisi dinamis Anda dapat bergantung pada beberapa parameter.

Setelah memilih parameter data untuk kunci partisi Anda, Anda kemudian memetakan setiap parameter ke ekspresi jq yang valid. Tabel berikut menunjukkan pemetaan parameter ke ekspresi jq:

Parameter	ekspresi jq
`customer_id`	.customer_id
`device`	.type.perangkat
`year`	.event_timestamp| strftime (“%Y”)
`month`	.event_timestamp| strftime (“%m”)
`day`	.event_timestamp| strftime (“%d”)
`hour`	.event_timestamp| strftime (“%H”)

Saat runtime, Firehose menggunakan kolom kanan di atas untuk mengevaluasi parameter berdasarkan data di setiap record.

Buat kunci partisi dengan AWS Lambda Fungsi

Untuk catatan data terkompresi atau terenkripsi, atau data yang dalam format file apa pun selain JSON, Anda dapat menggunakan AWS Lambda fungsi terintegrasi dengan kode kustom Anda sendiri untuk mendekompresi, mendekripsi, atau mengubah catatan untuk mengekstrak dan mengembalikan bidang data yang diperlukan untuk partisi. Ini adalah perluasan dari fungsi transformasi Lambda yang ada yang tersedia saat ini dengan Firehose. Anda dapat mengubah, mengurai, dan mengembalikan bidang data yang kemudian dapat Anda gunakan untuk partisi dinamis menggunakan fungsi Lambda yang sama.

Berikut ini adalah contoh Firehose stream processing fungsi Lambda di Python yang memutar ulang setiap catatan baca dari input ke output dan mengekstrak kunci partisi dari catatan.

```
from __future__ import print_function
import base64
import json
import datetime

# Signature for all Lambda functions that user must implement
```

```
def lambda_handler(firehose_records_input, context):
    print("Received records for processing from DeliveryStream: " +
 firehose_records_input['deliveryStreamArn']
          + ", Region: " + firehose_records_input['region']
          + ", and InvocationId: " + firehose_records_input['invocationId'])

    # Create return value.
    firehose_records_output = {'records': []}

    # Create result object.
    # Go through records and process them

    for firehose_record_input in firehose_records_input['records']:
        # Get user payload
        payload = base64.b64decode(firehose_record_input['data'])
        json_value = json.loads(payload)

        print("Record that was received")
        print(json_value)
        print("\n")
        # Create output Firehose record and add modified payload and record ID to it.
        firehose_record_output = {}
        event_timestamp = datetime.datetime.fromtimestamp(json_value['eventTimestamp'])
        partition_keys = {"customerId": json_value['customerId'],
                          "year": event_timestamp.strftime('%Y'),
                          "month": event_timestamp.strftime('%m'),
                          "day": event_timestamp.strftime('%d'),
                          "hour": event_timestamp.strftime('%H'),
                          "minute": event_timestamp.strftime('%M')
                          }

        # Create output Firehose record and add modified payload and record ID to it.
        firehose_record_output = {'recordId': firehose_record_input['recordId'],
                                  'data': firehose_record_input['data'],
                                  'result': 'Ok',
                                  'metadata': { 'partitionKeys': partition_keys }}

        # Must set proper record ID
        # Add the record to the list of output records.

        firehose_records_output['records'].append(firehose_record_output)

    # At the end return processed records
```

```
    return firehose_records_output
```

Berikut ini adalah contoh Firehose stream processing fungsi Lambda di Go yang memutar ulang setiap catatan baca dari input ke output dan mengekstrak kunci partisi dari catatan.

```
package main

import (
 "fmt"
 "encoding/json"
 "time"
 "strconv"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

type DataFirehoseEventRecordData struct {
 CustomerId string `json:"customerId"`
}

func handleRequest(evnt events.DataFirehoseEvent) (events.DataFirehoseResponse, error)
 {

 fmt.Printf("InvocationID: %s\n", evnt.InvocationID)
 fmt.Printf("DeliveryStreamArn: %s\n", evnt.DeliveryStreamArn)
 fmt.Printf("Region: %s\n", evnt.Region)

 var response events.DataFirehoseResponse

 for _, record := range evnt.Records {
  fmt.Printf("RecordID: %s\n", record.RecordID)
  fmt.Printf("ApproximateArrivalTimestamp: %s\n", record.ApproximateArrivalTimestamp)

  var transformedRecord events.DataFirehoseResponseRecord
  transformedRecord.RecordID = record.RecordID
  transformedRecord.Result = events.DataFirehoseTransformedStateOk
  transformedRecord.Data = record.Data

  var metaData events.DataFirehoseResponseRecordMetadata
  var recordData DataFirehoseEventRecordData
  partitionKeys := make(map[string]string)
```

```
  currentTime := time.Now()
  json.Unmarshal(record.Data, &recordData)
  partitionKeys["customerId"] = recordData.CustomerId
  partitionKeys["year"] = strconv.Itoa(currentTime.Year())
  partitionKeys["month"] = strconv.Itoa(int(currentTime.Month()))
  partitionKeys["date"] = strconv.Itoa(currentTime.Day())
  partitionKeys["hour"] = strconv.Itoa(currentTime.Hour())
  partitionKeys["minute"] = strconv.Itoa(currentTime.Minute())
  metaData.PartitionKeys = partitionKeys
  transformedRecord.Metadata = metaData

  response.Records = append(response.Records, transformedRecord)
 }

 return response, nil
}

func main() {
 lambda.Start(handleRequest)
}
```

Gunakan awalan bucket Amazon S3 untuk mengirimkan data

Saat membuat aliran Firehose yang menggunakan Amazon S3 sebagai tujuan, Anda harus menentukan bucket Amazon S3 tempat Firehose akan mengirimkan data Anda. Awalan bucket Amazon S3 digunakan untuk mengatur data yang Anda simpan di bucket S3. Awalan bucket Amazon S3 mirip dengan direktori yang memungkinkan Anda mengelompokkan objek serupa bersama-sama.

Dengan partisi dinamis, data yang dipartisi akan dikirimkan ke awalan Amazon S3 yang ditentukan. Jika Anda tidak mengaktifkan partisi dinamis, menentukan awalan bucket S3 untuk aliran Firehose Anda adalah opsional. Namun, jika Anda memilih untuk mengaktifkan partisi dinamis, Anda harus menentukan awalan bucket S3 tempat Firehose mengirimkan data yang dipartisi.

Di setiap aliran Firehose tempat Anda mengaktifkan partisi dinamis, nilai awalan bucket S3 terdiri dari ekspresi berdasarkan kunci partisi yang ditentukan untuk aliran Firehose tersebut. Menggunakan contoh catatan data di atas lagi, Anda dapat membangun nilai awalan S3 berikut yang terdiri dari ekspresi berdasarkan kunci partisi yang didefinisikan di atas:

```
"ExtendedS3DestinationConfiguration": {
"BucketARN": "arn:aws:s3:::my-logs-prod",
"Prefix": "customer_id=!{partitionKeyFromQuery:customer_id}/
    device=!{partitionKeyFromQuery:device}/
    year=!{partitionKeyFromQuery:year}/
    month=!{partitionKeyFromQuery:month}/
    day=!{partitionKeyFromQuery:day}/
    hour=!{partitionKeyFromQuery:hour}/"
}
```

Firehose mengevaluasi ekspresi di atas saat runtime. Ini mengelompokkan catatan yang cocok dengan ekspresi awalan S3 yang dievaluasi yang sama ke dalam satu set data. Firehose kemudian mengirimkan setiap kumpulan data ke awalan S3 yang dievaluasi. Frekuensi pengiriman kumpulan data ke S3 ditentukan oleh pengaturan buffer aliran Firehose. Akibatnya, catatan dalam contoh ini dikirim ke kunci objek S3 berikut:

```
s3://my-logs-prod/customer_id=1234567890/device=mobile/year=2019/month=08/day=09/
hour=20/my-delivery-stream-2019-08-09-23-55-09-a9fa96af-e4e4-409f-bac3-1f804714faaa
```

Untuk partisi dinamis, Anda harus menggunakan format ekspresi berikut di awalan bucket S3:`!{namespace:value}`, di mana namespace dapat berupa atau, atau keduanya. `partitionKeyFromQuery` `partitionKeyFromLambda` Jika Anda menggunakan penguraian inline untuk membuat kunci partisi untuk data sumber Anda, Anda harus menentukan nilai awalan bucket S3 yang terdiri dari ekspresi yang ditentukan dalam format berikut:. `"partitionKeyFromQuery:keyID"` Jika Anda menggunakan fungsi AWS Lambda untuk membuat kunci partisi untuk data sumber Anda, Anda harus menentukan nilai awalan bucket S3 yang terdiri dari ekspresi yang ditentukan dalam format berikut:. `"partitionKeyFromLambda:keyID"`

> **Note**
>
> Anda juga dapat menentukan nilai awalan bucket S3 menggunakan format gaya sarang, misalnya customer_id =! {partisi KeyFromQuery:customer _id}.

Untuk informasi selengkapnya, lihat “Pilih Amazon S3 untuk Tujuan Anda” di Membuat aliran Amazon Firehose dan Awalan Kustom untuk Objek Amazon S3.

Tambahkan pembatas baris baru saat mengirimkan data ke Amazon S3

Anda dapat mengaktifkan New Line Delimiter untuk menambahkan pembatas baris baru di antara catatan dalam objek yang dikirim ke Amazon S3. Hal ini dapat membantu penguraian objek di Amazon S3. Ini juga sangat berguna ketika partisi dinamis diterapkan ke data agregat karena deagregasi multi-rekaman (yang harus diterapkan pada data agregat sebelum dapat dipartisi secara dinamis) menghapus baris baru dari catatan sebagai bagian dari proses penguraian.

Terapkan partisi dinamis ke data agregat

Anda dapat menerapkan partisi dinamis ke data agregat (misalnya, beberapa peristiwa, log, atau catatan yang digabungkan menjadi satu `PutRecord` panggilan `PutRecordBatch` API) tetapi data ini harus dideagregasi terlebih dahulu. Anda dapat mendeagregasi data Anda dengan mengaktifkan deagregasi multi rekaman - proses penguraian melalui catatan di aliran Firehose dan memisahkannya.

Deagregasi multi record dapat berupa `JSON` tipe, artinya pemisahan catatan didasarkan pada objek JSON yang berurutan. Deagregasi juga dapat dari jenis`Delimited`, yang berarti bahwa pemisahan catatan dilakukan berdasarkan pembatas khusus yang ditentukan. Pembatas kustom ini harus berupa string yang dikodekan basis-64. Misalnya, jika Anda ingin menggunakan string berikut sebagai pembatas kustom Anda####, Anda harus menentukannya dalam format yang disandikan basis-64, yang menerjemahkannya ke. `IyMjIw==` Rekaman deagregasi oleh JSON atau dengan pembatas dibatasi pada 500 per catatan.

> **Note**
>
> Saat mendeagregasi catatan JSON, pastikan masukan Anda masih disajikan dalam format JSON yang didukung. Objek JSON harus berada pada satu baris tanpa pembatas atau newline-delimited (JSONL) saja. Array objek JSON bukan input yang valid.
> Ini adalah contoh masukan yang benar: `{"a":1}{"a":2} and {"a":1}\n{"a":2}`
> Ini adalah contoh dari input yang salah: `[{"a":1}, {"a":2}]`

Dengan data agregat, saat Anda mengaktifkan partisi dinamis, Firehose mem-parsing rekaman dan mencari objek JSON yang valid atau catatan yang dibatasi dalam setiap panggilan API berdasarkan jenis deagregasi multi rekaman yang ditentukan.

> **Important**
>
> Jika data Anda digabungkan, partisi dinamis hanya dapat diterapkan jika data Anda dideagregasi terlebih dahulu.

> **Important**
>
> Saat Anda menggunakan fitur Transformasi Data di Firehose, deagregasi akan diterapkan sebelum Transformasi Data. Data yang masuk ke Firehose akan diproses dengan urutan sebagai berikut: Deagregasi → Transformasi Data melalui Lambda → Kunci Partisi.

Memecahkan masalah kesalahan partisi dinamis

Jika Amazon Data Firehose tidak dapat mengurai catatan data dalam aliran Firehose Anda atau gagal mengekstrak kunci partisi yang ditentukan, atau untuk mengevaluasi ekspresi yang disertakan dalam nilai awalan S3, catatan data ini akan dikirimkan ke awalan bucket kesalahan S3 yang harus Anda tentukan saat membuat aliran Firehose tempat Anda mengaktifkan partisi dinamis. Awalan bucket kesalahan S3 berisi semua catatan yang Firehose tidak dapat kirimkan ke tujuan S3 yang ditentukan. Catatan ini diatur berdasarkan jenis kesalahan. Seiring dengan catatan, objek yang dikirim juga mencakup informasi tentang kesalahan untuk membantu memahami dan menyelesaikan kesalahan.

Anda harus menentukan awalan bucket kesalahan S3 untuk aliran Firehose jika Anda ingin mengaktifkan partisi dinamis untuk aliran Firehose ini. Jika Anda tidak ingin mengaktifkan partisi dinamis untuk aliran Firehose, menentukan awalan bucket kesalahan S3 adalah opsional.

Buffer data untuk partisi dinamis

Amazon Data Firehose menyangga data streaming yang masuk ke ukuran tertentu dan untuk jangka waktu tertentu sebelum mengirimkannya ke tujuan yang ditentukan. Anda dapat mengonfigurasi ukuran buffer dan interval buffer sambil membuat aliran Firehose baru atau memperbarui ukuran buffer dan interval buffer pada aliran Firehose yang ada. Ukuran buffer diukur dalam MB dan interval buffer diukur dalam hitungan detik.

> **Note**
>
> Fitur buffering nol tidak tersedia untuk partisi dinamis.

Saat partisi dinamis diaktifkan, Firehose secara internal menyangga catatan milik partisi tertentu berdasarkan petunjuk buffering (ukuran dan waktu) yang dikonfigurasi sebelum mengirimkan catatan ini ke bucket Amazon S3 Anda. Untuk memberikan objek ukuran maksimum, Firehose menggunakan buffering multi-tahap secara internal. Oleh karena itu, penundaan ujung ke ujung dari sekumpulan catatan mungkin 1,5 kali dari waktu petunjuk buffering yang dikonfigurasi. Hal ini mempengaruhi kesegaran data dari aliran Firehose.

Jumlah partisi aktif adalah jumlah total partisi aktif dalam buffer pengiriman. Misalnya, jika kueri partisi dinamis membangun 3 partisi per detik dan Anda memiliki konfigurasi petunjuk buffer yang memicu pengiriman setiap 60 detik, maka rata-rata Anda akan memiliki 180 partisi aktif. Jika Firehose tidak dapat mengirimkan data dalam partisi ke tujuan, partisi ini dihitung sebagai aktif dalam buffer pengiriman hingga dapat dikirim.

Partisi baru dibuat ketika awalan S3 dievaluasi ke nilai baru berdasarkan bidang data catatan dan ekspresi awalan S3. Buffer baru dibuat untuk setiap partisi aktif. Setiap catatan berikutnya dengan awalan S3 yang dievaluasi yang sama dikirim ke buffer itu.

Setelah buffer memenuhi batas ukuran buffer atau interval waktu buffer, Firehose membuat objek dengan data buffer dan mengirimkannya ke awalan Amazon S3 yang ditentukan. Setelah objek dikirim, buffer untuk partisi itu dan partisi itu sendiri dihapus dan dihapus dari jumlah partisi aktif.

Firehose mengirimkan setiap data buffer sebagai objek tunggal setelah ukuran buffer atau interval terpenuhi untuk setiap partisi secara terpisah. Setelah jumlah partisi aktif mencapai batas 500 per aliran Firehose, sisa catatan dalam aliran Firehose dikirim ke awalan bucket kesalahan S3 yang ditentukan (aktif). PartitionExceeded Anda dapat menggunakan formulir Amazon Data Firehose Limits untuk meminta peningkatan kuota ini hingga 2500 partisi aktif per aliran Firehose tertentu. Jika Anda membutuhkan lebih banyak partisi, Anda dapat membuat lebih banyak aliran Firehose dan mendistribusikan partisi aktif di dalamnya.

Mengkonversi format data masukan di Amazon Data Firehose

Amazon Data Firehose dapat mengonversi format data input Anda dari JSON ke Apache Parquet atau Apache ORC sebelum menyimpan data di Amazon S3. Parquet dan ORC adalah format data kolumnar yang menghemat ruang dan memungkinkan kueri yang lebih cepat dibandingkan dengan format berorientasi baris seperti JSON. Jika Anda ingin mengonversi format input selain JSON, seperti nilai yang dipisahkan koma (CSV) atau teks terstruktur, Anda dapat menggunakannya AWS Lambda untuk mengubahnya menjadi JSON terlebih dahulu. Untuk informasi selengkapnya, lihat Mengubah data sumber.

Anda dapat mengonversi format data Anda bahkan jika Anda menggabungkan catatan Anda sebelum mengirimnya ke Amazon Data Firehose.

Amazon Data Firehose memerlukan tiga elemen berikut untuk mengonversi format data rekaman Anda:

Deserializer

Amazon Data Firehose memerlukan deserializer untuk membaca JSON data input Anda. Anda dapat memilih salah satu dari dua jenis deserializer berikut.

Ketika menggabungkan beberapa dokumen JSON ke dalam catatan yang sama, pastikan bahwa input Anda masih disajikan dalam format JSON yang didukung. Array dokumen JSON bukan input yang valid.

Misalnya, ini adalah input yang benar: `{"a": 1}{"b": 1}` dan ini adalah input yang salah:`[{"a":1}, {"a":2}]`.

- Apache Sarang JSON SerDe
- OpenX JSON SerDe

Pilih deserializer JSON

Pilih OpenX JSON SerDe jika masukan JSON Anda berisi stempel waktu dalam format berikut:

- yyyy-MM-dd'T'HH:mm: SS [.S] 'Z', di mana fraksi dapat memiliki hingga 9 digit — Misalnya,. `2017-02-07T15:13:01.39256Z`

- yyyy-[M]M-[d]d HH:mm:ss[.S], dengan fraksi dapat terdiri atas hingga 9 digit – Sebagai contoh, `2017-02-07 15:13:01.14`.
- Detik jangka waktu – Sebagai contoh, `1518033528`.
- Milidetik jangka waktu – Sebagai contoh, `1518033528123`.
- Detik jangka waktu titik mengambang – Sebagai contoh, `1518033528.123`.

OpenX JSON SerDe dapat mengonversi periode (`.`) menjadi garis bawah (). _ OpenX JSON SerDE juga dapat mengonversi kunci JSON menjadi huruf kecil sebelum mendeserialisasinya. Untuk informasi selengkapnya tentang opsi yang tersedia dengan deserializer ini melalui Amazon Data Firehose, lihat Buka. XJson SerDe

Jika Anda tidak yakin deserializer mana yang harus dipilih, gunakan OpenX JSON SerDe, kecuali jika Anda memiliki stempel waktu yang tidak didukungnya.

Jika Anda memiliki stempel waktu dalam format selain yang tercantum sebelumnya, gunakan Apache Hive JSON. SerDe Bila Anda memilih deserializer ini, Anda dapat menentukan format stempel waktu yang akan digunakan. Untuk melakukannya, ikuti sintaks pola string format `DateTimeFormat` Joda-Time. Untuk informasi selengkapnya, lihat Kelas DateTimeFormat.

Anda juga dapat menggunakan nilai khusus `millis` untuk mengurai stempel waktu dalam milidetik jangka waktu. Jika Anda tidak menentukan format, Amazon Data Firehose menggunakan secara `java.sql.Timestamp::valueOf` default.

The Hive JSON SerDe tidak mengizinkan hal berikut:

- Periode (`.`) di nama kolom.
- Bidang yang jenisnya `uniontype`.
- Bidang yang memiliki jenis numerik dalam skema, tetapi berupa string di JSON. Misalnya, jika skema adalah (int), dan JSON adalah`{"a":"123"}`, Hive SerDe memberikan kesalahan.

The Hive SerDe tidak mengubah JSON bersarang menjadi string. Misalnya, jika Anda memiliki `{"a":{"inner":1}}`, itu tidak memperlakukan `{"inner":1}` sebagai string.

Skema

Amazon Data Firehose memerlukan skema untuk menentukan cara menafsirkan data tersebut. Gunakan AWS Glue untuk membuat skema di. AWS Glue Data Catalog Amazon Data Firehose

kemudian mereferensikan skema tersebut dan menggunakannya untuk menafsirkan data input Anda. Anda dapat menggunakan skema yang sama untuk mengonfigurasi Amazon Data Firehose dan perangkat lunak analitik Anda. Untuk informasi selengkapnya, lihat Mengisi Katalog Data AWS Glue di Panduan AWS Glue Pengembang.

> **Note**
>
> Skema yang dibuat dalam Katalog AWS Glue Data harus sesuai dengan struktur data input. Jika tidak, data yang dikonversi tidak akan berisi atribut yang tidak ditentukan dalam skema. Jika Anda menggunakan JSON bersarang, gunakan tipe STRUCT dalam skema yang mencerminkan struktur data JSON Anda. Lihat contoh ini untuk cara menangani JSON bersarang dengan tipe STRUCT.

> **Important**
>
> Untuk tipe data yang tidak menentukan batas ukuran, ada batas praktis 32 MBs untuk semua data dalam satu baris.
> Jika Anda menentukan panjang untuk `CHAR` or`VARCHAR`, Firehose memotong string pada panjang yang ditentukan saat membaca data input. Jika string data yang mendasarinya lebih panjang, itu tetap tidak berubah.

Serializer

Firehose memerlukan serializer untuk mengonversi data ke format penyimpanan kolom target (Parket atau ORC) - Anda dapat memilih salah satu dari dua jenis serializer berikut.

- ORC SerDe
- Parket SerDe

Pilih serializer

Serializer yang Anda pilih tergantung pada kebutuhan bisnis Anda. Untuk mempelajari lebih lanjut tentang dua opsi serializer, lihat ORC SerDe dan Parket. SerDe

Aktifkan konversi format rekaman

Jika Anda mengaktifkan konversi format rekaman, Anda tidak dapat menyetel tujuan Amazon Data Firehose menjadi Amazon OpenSearch Service, Amazon Redshift, atau Splunk. Dengan konversi format diaktifkan, Amazon S3 adalah satu-satunya tujuan yang dapat Anda gunakan untuk aliran Firehose Anda. Bagian berikut menunjukkan cara mengaktifkan konversi format rekaman dari operasi konsol dan Firehose API. Untuk contoh cara mengatur konversi format rekaman dengan CloudFormation, lihat AWS::DataFirehose: DeliveryStream.

Aktifkan konversi format rekaman dari konsol

Anda dapat mengaktifkan konversi format data di konsol saat membuat atau memperbarui aliran Firehose. Dengan konversi format data diaktifkan, Amazon S3 adalah satu-satunya tujuan yang dapat Anda konfigurasi untuk aliran Firehose. Selain itu, kompresi Amazon S3 akan dinonaktifkan ketika Anda mengaktifkan konversi format. Namun, kompresi Snappy terjadi secara otomatis sebagai bagian dari proses konversi. Format pembingkaian untuk Snappy yang digunakan Amazon Data Firehose dalam hal ini kompatibel dengan Hadoop. Ini berarti Anda dapat menggunakan hasil kompresi Snappy dan menjalankan kueri pada data ini di Athena. Untuk format pembingkaian Snappy yang diandalkan Hadoop, lihat .java. BlockCompressorStream

Untuk mengaktifkan konversi format data untuk aliran Firehose data

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon Data Firehose di. https://console.aws.amazon.com/firehose/
2. Pilih aliran Firehose untuk diperbarui, atau buat aliran Firehose baru dengan mengikuti langkah-langkahnya. Tutorial: Membuat aliran Firehose dari konsol
3. Di bawah Mengonversi format catatan, atur Konversi format catatan ke Diaktifkan.
4. Pilih format output yang Anda inginkan. Untuk informasi selengkapnya tentang kedua opsi, lihat Apache Parquet dan Apache ORC.
5. Pilih AWS Glue tabel untuk menentukan skema untuk catatan sumber Anda. Atur Wilayah, basis data, tabel, dan versi tabel.

Mengelola konversi format rekaman dari Firehose API

Jika Anda ingin Amazon Data Firehose mengonversi format data input Anda dari JSON ke Parket atau ORC, tentukan DataFormatConversionConfigurationelemen opsional di Extendeds3

atau di Extendeds3. DestinationConfiguration DestinationUpdate Jika Anda menentukan DataFormatConversionConfiguration, pembatasan berikut berlaku.

- Di BufferingHints, Anda tidak dapat mengatur `SizeInMBs` ke nilai kurang dari 64 jika Anda mengaktifkan konversi format rekaman. Selain itu, bila konversi format tidak diaktifkan, nilai defaultnya adalah 5. Nilai menjadi 128 saat Anda mengaktifkannya.
- Anda harus mengatur `CompressionFormat` di Extendeds3 DestinationConfiguration atau di Extendeds3 ke. DestinationUpdate `UNCOMPRESSED` Nilai default-nya `CompressionFormat` is `UNCOMPRESSED`. Oleh karena itu, Anda juga dapat membiarkannya tidak ditentukan di DestinationConfigurationExtendeds3. Data masih dikompresi sebagai bagian dari proses serialisasi, menggunakan kompresi Snappy secara default. Format pembingkaian untuk Snappy yang digunakan Amazon Data Firehose dalam hal ini kompatibel dengan Hadoop. Ini berarti Anda dapat menggunakan hasil kompresi Snappy dan menjalankan kueri pada data ini di Athena. Untuk format pembingkaian Snappy yang diandalkan Hadoop, lihat .java. BlockCompressorStream Bila Anda mengonfigurasi serializer, Anda dapat memilih jenis kompresi lainnya.

Menangani kesalahan untuk konversi format data

Saat Amazon Data Firehose tidak dapat mengurai atau mendeserialisasi catatan (misalnya, ketika data tidak cocok dengan skema), ia menuliskannya ke Amazon S3 dengan awalan kesalahan. Jika penulisan ini gagal, Amazon Data Firehose mencobanya lagi selamanya, memblokir pengiriman lebih lanjut. Untuk setiap catatan yang gagal, Amazon Data Firehose menulis dokumen JSON dengan skema berikut:

```
{
  "attemptsMade": long,
  "arrivalTimestamp": long,
  "ErrorCode": string,
  "ErrorMessage": string,
  "attemptEndingTimestamp": long,
  "rawData": string,
  "sequenceNumber": string,
  "subSequenceNumber": long,
  "dataCatalogTable": {
    "catalogId": string,
    "databaseName": string,
    "tableName": string,
    "region": string,
    "versionId": string,
```

```
    "catalogArn": string
  }
}
```

Memahami pengiriman data di Amazon Data Firehose

Saat Anda mengirim data ke aliran Firehose, data akan dikirim secara otomatis ke tujuan yang Anda pilih. Tabel berikut menjelaskan pengiriman data ke tujuan yang berbeda.

Destinasi	Detail
Amazon S3	Untuk pengiriman data ke Amazon S3, Firehose menggabungkan beberapa catatan masuk berdasarkan konfigurasi buffering aliran Firehose Anda. Kinesis Data Firehose kemudian memberikan catatan tersebut ke Amazon S3 sebagai objek Amazon S3. Secara default, Firehose menggabungkan data tanpa pembatas apa pun. Jika Anda ingin memiliki pembatas baris baru di antara catatan, Anda dapat menambahkan pembatas baris baru dengan mengaktifkan fitur dalam konfigurasi konsol Firehose atau parameter API. Pengiriman data antara Firehose dan tujuan Amazon S3 dienkripsi dengan TLS (HTTPS).
Amazon Redshift	Untuk pengiriman data ke Amazon Redshift, Firehose pertama-t ama mengirimkan data masuk ke bucket S3 Anda dalam format yang dijelaskan sebelumnya. Firehose kemudian mengeluarkan perintah Amazon COPY Redshift untuk memuat data dari bucket S3 ke cluster yang disediakan Amazon Redshift atau workgroup Amazon Redshift Serverless. Pastikan bahwa setelah Amazon Data Firehose menggabungkan beberapa catatan masuk ke objek Amazon S3, objek Amazon S3 dapat disalin ke klaster yang disediakan Amazon Redshift atau grup kerja Amazon Redshift Tanpa Server. Untuk informasi selengkapnya, lihat Parameter Format Data Perintah SALIN Amazon Redshift.
OpenSearch Layanan dan Tanpa OpenSearch Server	Untuk pengiriman data ke OpenSearch Layanan dan OpenSearch Tanpa Server, Amazon Data Firehose menyangga catatan masuk berdasarkan konfigurasi buffering aliran Firehose Anda. Kemudian menghasilkan permintaan massal OpenSearch Layanan atau OpenSearch Tanpa Server untuk mengindeks beberapa catatan ke kluster OpenSearch Layanan atau koleksi Tanpa OpenSearch Server Anda. Pastikan rekaman Anda dikodekan UTF-8 dan diratakan ke

Destinasi	Detail
	objek JSON satu baris sebelum Anda mengirimkannya ke Amazon Data Firehose. Selain itu, `rest.action.multi.allow_explicit_index` opsi untuk kluster OpenSearch Layanan Anda harus disetel ke true (default) untuk mengambil permintaan massal dengan indeks eksplisit yang ditetapkan per catatan. Untuk informasi selengkapnya, lihat Opsi Lanjutan Konfigurasi OpenSearch Layanan di Panduan Pengembang OpenSearch Layanan Amazon.
Splunk	Untuk pengiriman data ke Splunk, Amazon Data Firehose menggabungkan byte yang Anda kirim. Jika Anda ingin pembatas dalam data, seperti karakter baris baru, Anda harus memasukkannya sendiri. Pastikan bahwa Splunk dikonfigurasi untuk mengurai pembatas tersebut. Untuk mengarahkan ulang data yang dikirim ke bucket kesalahan S3 (cadangan S3) kembali ke Splunk, ikuti langkah-langkah yang disebutkan dalam dokumentasi Splunk.
Titik akhir HTTP	Untuk pengiriman data ke titik akhir HTTP yang dimiliki oleh penyedia layanan pihak ketiga yang didukung, Anda dapat menggunakan layanan Amazon Lambda terintegrasi untuk membuat fungsi untuk mengubah rekaman masuk ke format yang sesuai dengan format yang diharapkan integrasi penyedia layanan. Hubungi penyedia layanan pihak ketiga yang titik akhir HTTP-nya Anda pilih sebagai tujuan untuk mempelajari lebih lanjut tentang format catatan yang diterima.

Destinasi	Detail
Kepingan salju	Untuk pengiriman data ke Snowflake, Amazon Data Firehose secara internal menyangga data selama satu detik dan menggunak an operasi API streaming Snowflake untuk menyisipkan data ke Snowflake. Secara default, catatan yang Anda sisipkan dibilas dan dimasukkan ke tabel Snowflake setiap detik. Setelah Anda melakukan panggilan insert, Firehose memancarkan CloudWatch metrik yang mengukur berapa lama waktu yang dibutuhkan untuk data untuk berkomitmen ke Snowflake. Firehose saat ini hanya mendukung satu item JSON sebagai muatan rekaman dan tidak mendukung array JSON. Pastikan payload input Anda adalah objek JSON yang valid dan terbentuk dengan baik tanpa tanda kutip ganda, tanda kutip, atau karakter escape tambahan.

Setiap tujuan Firehose memiliki frekuensi pengiriman datanya sendiri. Untuk informasi selengkapnya, lihat Konfigurasikan petunjuk buffering.

Catatan duplikat

Amazon Data Firehose menggunakan at-least-once semantik untuk pengiriman data. Dalam beberapa keadaan, seperti ketika waktu pengiriman data habis, percobaan ulang pengiriman oleh Amazon Data Firehose mungkin memperkenalkan duplikat jika permintaan pengiriman data asli akhirnya berhasil. Ini berlaku untuk semua jenis tujuan yang didukung Amazon Data Firehose, kecuali untuk tujuan Amazon S3, Apache Iceberg Tables, dan tujuan Snowflake.

Topik

- Memahami pengiriman di seluruh AWS akun dan wilayah
- Memahami permintaan pengiriman titik akhir HTTP dan spesifikasi respons
- Menangani kegagalan pengiriman data
- Konfigurasikan format nama objek Amazon S3
- Konfigurasikan rotasi indeks untuk OpenSearch Layanan
- Jeda dan lanjutkan pengiriman data

Memahami pengiriman di seluruh AWS akun dan wilayah

Amazon Data Firehose mendukung pengiriman data ke tujuan titik akhir HTTP di seluruh akun. AWS Aliran Firehose dan titik akhir HTTP yang Anda pilih sebagai tujuan dapat menjadi milik akun yang berbeda. AWS

Amazon Data Firehose juga mendukung pengiriman data ke tujuan titik akhir HTTP di seluruh wilayah. AWS Anda dapat mengirimkan data dari aliran Firehose di satu AWS wilayah ke titik akhir HTTP di wilayah lain. AWS Anda juga dapat mengirimkan data dari aliran Firehose ke tujuan titik akhir HTTP di luar AWS wilayah, misalnya ke server lokal Anda sendiri dengan menyetel URL titik akhir HTTP ke tujuan yang Anda inginkan. Untuk skenario ini, biaya transfer data tambahan ditambahkan ke biaya pengiriman Anda. Untuk informasi selengkapnya tentang harga transfer data, lihat bagian Transfer Data di halaman "Harga Sesuai Permintaan".

Memahami permintaan pengiriman titik akhir HTTP dan spesifikasi respons

Agar Amazon Data Firehose berhasil mengirimkan data ke titik akhir HTTP kustom, titik akhir ini harus menerima permintaan dan mengirim tanggapan menggunakan format permintaan dan respons Amazon Data Firehose tertentu. Bagian ini menjelaskan spesifikasi format permintaan HTTP yang dikirim oleh layanan Amazon Data Firehose ke titik akhir HTTP kustom, serta spesifikasi format respons HTTP yang diharapkan oleh layanan Amazon Data Firehose. Titik akhir HTTP memiliki waktu 3 menit untuk menanggapi permintaan sebelum Amazon Data Firehose menghentikan permintaan tersebut. Amazon Data Firehose memperlakukan respons yang tidak sesuai dengan format yang tepat sebagai kegagalan pengiriman.

Format permintaan

Parameter Jalur dan URL

Ini dikonfigurasi secara langsung oleh Anda sebagai bagian dari bidang URL tunggal. Amazon Data Firehose mengirimkannya seperti yang dikonfigurasi tanpa modifikasi. Hanya tujuan https yang didukung. Pembatasan URL diterapkan selama konfigurasi aliran pengiriman.

> **Note**
>
> Saat ini, hanya port 443 yang didukung untuk pengiriman data titik akhir HTTP.

HTTP Header - -Versi X-Amz-Firehose-Protocol

Header ini digunakan untuk menunjukkan versi format permintaan/respons. Saru-satunya versi saat ini adalah 1.0.

HTTP Header - -Id X-Amz-Firehose-Request

Nilai header ini adalah GUID buram yang dapat digunakan untuk tujuan debugging dan deduplikasi. Implementasi titik akhir harus mencatat nilai header ini jika memungkinkan, baik permintaan sukses maupun gagal. ID permintaan tetap sama di antara beberapa upaya permintaan yang sama.

Header HTTP - Content-Type

Nilai header Content-Type selalu `application/json`.

Header HTTP - Content-Encoding

Aliran Firehose dapat dikonfigurasi untuk menggunakan GZIP untuk mengompres tubuh saat mengirim permintaan. Ketika kompresi ini diaktifkan, nilai header Content-Encoding diatur ke gzip, sesuai praktek standar. Jika kompresi tidak diaktifkan, header Content-Encoding tidak ada sama sekali.

Header HTTP - Content-Length

Ini digunakan dengan cara standar.

HTTP Header - X-Amz-Firehose-Source -Arn:

ARN dari aliran Firehose diwakili dalam format string ASCII. ARN mengkodekan wilayah, ID AWS akun, dan nama aliran. Misalnya, `arn:aws:firehose:us-east-1:123456789:deliverystream/testStream`.

HTTP Header - -Kunci X-Amz-Firehose-Access

Header ini membawa kunci API atau kredensial lainnya. Anda memiliki kemampuan untuk membuat atau memperbarui kunci API (alias token otorisasi) saat membuat atau memperbarui aliran pengiriman Anda. Amazon Data Firehose membatasi ukuran kunci akses ke 4096 byte. Amazon Data Firehose tidak mencoba menafsirkan kunci ini dengan cara apa pun. Kunci yang dikonfigurasi disalin secara verbatim ke nilai header ini. Namun, jika Anda menggunakan Secrets Manager untuk mengkonfigurasi kunci, maka rahasia harus mengikuti format objek JSON tertentu:`{"api_key": "..."}`.

Isi bisa berubah-ubah dan berpotensi mewakili token JWT atau ACCESS_KEY. Jika titik akhir memerlukan kredensial multi-bidang (misalnya, nama pengguna dan kata sandi), nilai semua

bidang harus disimpan bersama-sama dalam satu access key dalam format yang dimengerti titik akhir (JSON atau CSV). Bidang ini dapat berupa dikodekan basis-64 jika isi asli bersifat biner. Amazon Data Firehose tidak mengubah and/or encode nilai yang dikonfigurasi dan menggunakan konten apa adanya.

HTTP Header - -Atribut X-Amz-Firehose-Common

Header ini membawa atribut umum (metadata) yang berkaitan dengan seluruh permintaan, and/or ke semua catatan dalam permintaan. Ini dikonfigurasi langsung oleh Anda saat membuat aliran Firehose. Nilai atribut ini dikodekan sebagai objek JSON dengan skema berikut:

```
"$schema": http://json-schema.org/draft-07/schema#

properties:
  commonAttributes:
    type: object
    minProperties: 0
    maxProperties: 50
    patternProperties:
      "^.{1,256}$":
        type: string
        minLength: 0
        maxLength: 1024
```

Inilah contohnya:

```
"commonAttributes": {
    "deployment -context": "pre-prod-gamma",
    "device-types": ""
  }
```

Isi - Ukuran Maks

Ukuran isi maksimum dikonfigurasi oleh Anda, dan bisa sampai maksimal 64 MiB, sebelum kompresi.

Isi - Skema

Isi membawa satu dokumen JSON dengan Skema JSON berikut (ditulis dalam YAML):

```
"$schema": http://json-schema.org/draft-07/schema#

title: FirehoseCustomHttpsEndpointRequest
description: >
  The request body that the Firehose service sends to
  custom HTTPS endpoints.
type: object
properties:
  requestId:
    description: >
      Same as the value in the X-Amz-Firehose-Request-Id header,
      duplicated here for convenience.
    type: string
  timestamp:
    description: >
      The timestamp (milliseconds since epoch) at which the Firehose
      server generated this request.
    type: integer
  records:
    description: >
      The actual records of the Firehose stream, carrying
      the customer data.
    type: array
    minItems: 1
    maxItems: 10000
    items:
      type: object
      properties:
        data:
          description: >
            The data of this record, in Base64. Note that empty
            records are permitted in Firehose. The maximum allowed
            size of the data, before Base64 encoding, is 1024000
            bytes; the maximum length of this field is therefore
            1365336 chars.
          type: string
          minLength: 0
          maxLength: 1365336

required:
  - requestId
  - records
```

Inilah contohnya:

```
{
  "requestId": "ed4acda5-034f-9f42-bba1-f29aea6d7d8f",
  "timestamp": 1578090901599
  "records": [
    {
      "data": "aGVsbG8="
    },
    {
      "data": "aGVsbG8gd29ybGQ="
    }
  ]
}
```

Format respons

Perilaku Default pada Galat

Jika respons gagal memenuhi persyaratan di bawah ini, server Firehose memperlakukannya seolah-olah memiliki kode status 500 tanpa isi.

Kode Status

Kode status HTTP HARUS berada di kisaran 2XX, 4XX, atau 5XX.

Server Amazon Data Firehose TIDAK mengikuti pengalihan (kode status 3XX). Hanya kode respons 200 yang dianggap sebagai pengiriman catatan yang sukses untuk HTTP/EP. Kode respons 413 (ukuran melebihi) dianggap sebagai kegagalan permanen dan kumpulan catatan tidak dikirim ke bucket kesalahan jika dikonfigurasi. Semua kode respons lainnya dianggap sebagai kesalahan yang bisa dicoba ulang dan dikenakan algoritma mundur coba lagi yang akan dijelaskan kemudian.

Header HTTP - Jenis Konten

Satu-satunya jenis konten yang dapat diterima adalah aplikasi/json.

Header HTTP - Content-Encoding

Content-Encoding TIDAK HARUS digunakan. Isi HARUS tidak dikompresi.

Header HTTP - Content-Length

Header Content-Length HARUS ada jika respons memiliki isi.

Isi - Ukuran Maks

Isi respons harus berukuran 1 MiB atau kurang.

```
"$schema": http://json-schema.org/draft-07/schema#

title: FirehoseCustomHttpsEndpointResponse

description: >
  The response body that the Firehose service sends to
  custom HTTPS endpoints.
type: object
properties:
  requestId:
    description: >
      Must match the requestId in the request.
    type: string

  timestamp:
    description: >
      The timestamp (milliseconds since epoch) at which the
      server processed this request.
    type: integer

  errorMessage:
    description: >
      For failed requests, a message explaining the failure.
      If a request fails after exhausting all retries, the last
      Instance of the error message is copied to error output
      S3 bucket if configured.
    type: string
    minLength: 0
    maxLength: 8192
required:
  - requestId
  - timestamp
```

Inilah contohnya:

```
Failure Case (HTTP Response Code 4xx or 5xx)
{
  "requestId": "ed4acda5-034f-9f42-bba1-f29aea6d7d8f",
  "timestamp": "1578090903599",
  "errorMessage": "Unable to deliver records due to unknown error."
}
Success case (HTTP Response Code 200)
{
  "requestId": "ed4acda5-034f-9f42-bba1-f29aea6d7d8f",
  "timestamp": 1578090903599
}
```

Penanganan Respons Kesalahan

Dalam semua kasus kesalahan, server Amazon Data Firehose mencoba kembali pengiriman kumpulan catatan yang sama menggunakan algoritma back-off eksponensial. Percobaan ulang dibatalkan menggunakan waktu mundur awal (1 detik) dengan faktor jitter (15%) dan setiap percobaan ulang berikutnya dibatalkan menggunakan rumus (initial-backoff-time * (pengganda (2) ^ retry_count)) dengan jitter tambahan. Waktu mundur dibatasi oleh interval maksimum 2 menit. Misalnya pada percobaan ulang 'n'-th, waktu mundur adalah = MAX (120, 2^n) * acak (0,85, 1,15).

Parameter yang ditentukan dalam persamaan sebelumnya dapat berubah sewaktu-waktu. Lihat dokumentasi AWS Firehose untuk mengetahui waktu mundur awal yang tepat, waktu backoff maks, persentase pengganda, dan jitter yang digunakan dalam algoritma mundur eksponensial.

Dalam setiap percobaan ulang berikutnya, and/or tujuan kunci akses ke mana catatan dikirimkan mungkin berubah berdasarkan konfigurasi aliran Firehose yang diperbarui. Layanan Amazon Data Firehose menggunakan id permintaan yang sama di seluruh percobaan ulang dengan cara yang terbaik. Fitur terakhir ini dapat digunakan untuk tujuan deduplikasi oleh server titik akhir HTTP. Jika permintaan masih belum dikirimkan setelah waktu maksimum yang diizinkan (berdasarkan konfigurasi aliran Firehose), kumpulan rekaman secara opsional dapat dikirim ke bucket kesalahan berdasarkan konfigurasi aliran.

Contoh

Contoh permintaan CWLog bersumber.

```
{
  "requestId": "ed4acda5-034f-9f42-bba1-f29aea6d7d8f",
  "timestamp": 1578090901599,
  "records": [
   {
    "data": {
      "messageType": "DATA_MESSAGE",
      "owner": "123456789012",
      "logGroup": "log_group_name",
      "logStream": "log_stream_name",
      "subscriptionFilters": [
        "subscription_filter_name"
      ],
      "logEvents": [
        {
          "id": "01234567890123456789012345678901234567890123456789012345",
          "timestamp": 1510109208016,
          "message": "log message 1"
        },
        {
          "id": "01234567890123456789012345678901234567890123456789012345",
          "timestamp": 1510109208017,
          "message": "log message 2"
        }
      ]
    }
   }
  ]
}
```

Menangani kegagalan pengiriman data

Setiap tujuan Amazon Data Firehose memiliki penanganan kegagalan pengiriman datanya sendiri.

Saat menyiapkan aliran Firehose, untuk banyak tujuan seperti, Splunk OpenSearch, dan titik akhir HTTP, Anda juga menyiapkan bucket S3 tempat data yang gagal dikirimkan dapat dicadangkan.

Untuk informasi selengkapnya tentang cara Firehose mencadangkan data jika pengiriman gagal, lihat bagian tujuan yang relevan di halaman ini. Untuk informasi selengkapnya tentang cara memberikan akses ke bucket S3 tempat data yang gagal dikirimkan dapat dicadangkan, lihat Memberikan Akses Firehose ke Tujuan Amazon S3. Ketika Firehose (a) gagal mengirimkan data ke tujuan streaming, dan (b) gagal menulis data ke bucket S3 cadangan untuk pengiriman yang gagal, Firehose secara efektif menghentikan pengiriman streaming hingga saat data dapat dikirim ke tujuan atau ditulis ke lokasi S3 cadangan.

Amazon S3

Pengiriman data ke bucket S3 Anda mungkin gagal karena berbagai alasan. Misalnya, bucket mungkin tidak ada lagi, peran IAM yang diasumsikan Amazon Data Firehose mungkin tidak memiliki akses ke bucket, jaringan gagal, atau kejadian serupa. Dalam kondisi ini, Amazon Data Firehose terus mencoba lagi hingga 24 jam hingga pengiriman berhasil. Waktu penyimpanan data maksimum Amazon Data Firehose adalah 24 jam. Jika pengiriman data gagal selama lebih dari 24 jam, data Anda akan hilang.

Pengiriman data ke bucket S3 Anda dapat gagal karena berbagai alasan, seperti:

- Ember sudah tidak ada lagi.
- Peran IAM yang diasumsikan oleh Amazon Data Firehose tidak memiliki akses ke bucket.
- Masalah jaringan.
- Kesalahan S3, seperti HTTP 500s atau kegagalan API lainnya.

Dalam kasus ini, Amazon Data Firehose akan mencoba lagi pengiriman:

- DirectPut sumber: Percobaan ulang berlanjut hingga 24 jam.
- Kinesis Data Streams atau sumber MSK Amazon: Percobaan ulang berlanjut tanpa batas waktu, hingga kebijakan retensi yang ditentukan pada aliran.

Amazon Data Firehose mengirimkan catatan yang gagal ke bucket kesalahan S3 hanya jika pemrosesan Lambda atau konversi parket gagal. Skenario kegagalan lainnya akan menghasilkan upaya coba lagi terus menerus ke S3 hingga periode retensi tercapai. Ketika Firehose berhasil mengirimkan catatan ke S3, Firehose membuat file objek S3, dan dalam kasus kegagalan rekaman sebagian, secara otomatis mencoba ulang pengiriman dan memperbarui file objek S3 yang sama dengan catatan yang berhasil diproses.

Amazon Redshift

Untuk tujuan Amazon Redshift, Anda dapat menentukan durasi coba lagi (0—7200 detik) saat membuat aliran Firehose.

Pengiriman data ke klaster yang disediakan Amazon Redshift atau grup kerja Amazon Redshift Serverless mungkin gagal karena beberapa alasan. Misalnya, Anda mungkin memiliki konfigurasi klaster yang salah dari aliran Firehose, klaster atau grup kerja yang sedang dalam pemeliharaan, atau kegagalan jaringan. Dalam kondisi ini, Amazon Data Firehose mencoba ulang untuk durasi waktu yang ditentukan dan melewatkan kumpulan objek Amazon S3 tertentu. Informasi objek yang dilewati dikirim ke bucket S3 Anda sebagai file manifes di folder `errors/`, yang dapat Anda gunakan untuk backfill manual. Untuk informasi tentang cara MENYALIN data secara manual dengan file manifes, lihat Menggunakan Manifes untuk Menentukan File Data.

OpenSearch Layanan Amazon dan Tanpa OpenSearch Server

Untuk tujuan OpenSearch Layanan dan OpenSearch Tanpa Server, Anda dapat menentukan durasi coba lagi (0—7200 detik) selama pembuatan aliran Firehose.

Pengiriman data ke kluster OpenSearch Layanan atau pengumpulan OpenSearch Tanpa Server Anda mungkin gagal karena beberapa alasan. Misalnya, Anda mungkin memiliki klaster OpenSearch Layanan atau konfigurasi koleksi OpenSearch Tanpa Server yang salah dari aliran Firehose, kluster OpenSearch Layanan, atau koleksi OpenSearch Tanpa Server dalam pemeliharaan, kegagalan jaringan, atau peristiwa serupa. Dalam kondisi ini, Amazon Data Firehose mencoba ulang untuk durasi waktu yang ditentukan dan kemudian melewatkan permintaan indeks tertentu. Dokumen yang dilewati dikirim ke bucket S3 Anda di folder `AmazonOpenSearchService_failed/`, yang dapat Anda gunakan untuk backfill manual.

Untuk OpenSearch Layanan, setiap dokumen memiliki format JSON berikut:

```
{
    "attemptsMade": "(number of index requests attempted)",
    "arrivalTimestamp": "(the time when the document was received by Firehose)",
    "errorCode": "(http error code returned by OpenSearch Service)",
    "errorMessage": "(error message returned by OpenSearch Service)",
    "attemptEndingTimestamp": "(the time when Firehose stopped attempting index
 request)",
    "esDocumentId": "(intended OpenSearch Service document ID)",
    "esIndexName": "(intended OpenSearch Service index name)",
    "esTypeName": "(intended OpenSearch Service type name)",
```

```
    "rawData": "(base64-encoded document data)"
}
```

Untuk OpenSearch Tanpa Server, setiap dokumen memiliki format JSON berikut:

```
{
    "attemptsMade": "(number of index requests attempted)",
    "arrivalTimestamp": "(the time when the document was received by Firehose)",
    "errorCode": "(http error code returned by OpenSearch Serverless)",
    "errorMessage": "(error message returned by OpenSearch Serverless)",
    "attemptEndingTimestamp": "(the time when Firehose stopped attempting index
 request)",
    "osDocumentId": "(intended OpenSearch Serverless document ID)",
    "osIndexName": "(intended OpenSearch Serverless index name)",
    "rawData": "(base64-encoded document data)"
}
```

Splunk

Saat Amazon Data Firehose mengirim data ke Splunk, ia menunggu pengakuan dari Splunk. Jika terjadi kesalahan, atau pengakuan tidak tiba dalam periode batas waktu pengakuan, Amazon Data Firehose memulai penghitung durasi coba lagi. Kinesis Data Firehose terus mencoba kembali sampai durasi coba lagi berakhir. Setelah itu, Amazon Data Firehose menganggapnya sebagai kegagalan pengiriman data dan mencadangkan data ke bucket Amazon S3 Anda.

Setiap kali Amazon Data Firehose mengirimkan data ke Splunk, apakah itu upaya awal atau percobaan ulang, itu memulai ulang penghitung batas waktu pengakuan. Kinesis Data Firehose kemudian menunggu pengakuan tiba dari Splunk. Bahkan jika durasi percobaan ulang berakhir, Amazon Data Firehose masih menunggu pengakuan sampai menerimanya atau batas waktu pengakuan tercapai. Jika waktu pengakuan habis, Amazon Data Firehose memeriksa untuk menentukan apakah ada waktu tersisa di penghitung coba lagi. Jika ada waktu yang tersisa, Kinesis Data Firehose akan mencoba lagi dan mengulangi logika sampai menerima pengakuan atau sampai waktu coba lagi telah berakhir.

Kegagalan untuk menerima pengakuan bukan satu-satunya jenis kesalahan pengiriman data yang dapat terjadi. Untuk informasi tentang jenis kesalahan pengiriman data lainnya, lihat Kesalahan Pengiriman Data Splunk. Kesalahan pengiriman data memicu logika coba lagi jika durasi coba lagi lebih besar dari 0.

Berikut ini adalah contoh catatan kesalahan.


```
{
  "attemptsMade": 0,
  "arrivalTimestamp": 1506035354675,
  "errorCode": "Splunk.AckTimeout",
  "errorMessage": "Did not receive an acknowledgement from HEC before the HEC
 acknowledgement timeout expired. Despite the acknowledgement timeout, it's possible
 the data was indexed successfully in Splunk. Amazon Data Firehose backs up in Amazon
 S3 data for which the acknowledgement timeout expired.",
  "attemptEndingTimestamp": 13626284715507,
  "rawData":
 "MiAyNTE2MjAyNzIyMDkgZW5pLTA1ZjMyMmQ1IDIxOC45Mi4xODguMjE0IDE3Mi4xNi4xLjE2NyAyNTIzMyAxNDMzIDYgM
  "EventId": "49577193928114147339600778471082492393164139877200035842.0"
}
```


Tujuan titik akhir HTTP

Saat Amazon Data Firehose mengirimkan data ke tujuan titik akhir HTTP, Amazon Data Firehose menunggu respons dari tujuan ini. Jika terjadi kesalahan, atau respons tidak tiba dalam periode batas waktu respons, Amazon Data Firehose memulai penghitung durasi coba lagi. Kinesis Data Firehose terus mencoba kembali sampai durasi coba lagi berakhir. Setelah itu, Amazon Data Firehose menganggapnya sebagai kegagalan pengiriman data dan mencadangkan data ke bucket Amazon S3 Anda.

Setiap kali Amazon Data Firehose mengirimkan data ke tujuan titik akhir HTTP, apakah itu upaya awal atau percobaan ulang, itu memulai ulang penghitung batas waktu respons. Kinesis Data Firehose kemudian menunggu respons dari tujuan titik akhir HTTP. Bahkan jika durasi percobaan ulang berakhir, Amazon Data Firehose masih menunggu respons hingga menerimanya atau batas waktu respons tercapai. Jika waktu respons habis, Amazon Data Firehose memeriksa untuk menentukan apakah ada waktu tersisa di penghitung coba lagi. Jika ada waktu yang tersisa, Kinesis Data Firehose akan mencoba lagi dan mengulangi logika sampai menerima respons atau sampai waktu coba lagi telah berakhir.

Kegagalan untuk menerima respons bukan satu-satunya jenis kesalahan pengiriman data yang dapat terjadi. Untuk informasi tentang jenis kesalahan pengiriman data lainnya, lihat Kesalahan Pengiriman Data Titik Akhir HTTP.

Berikut ini adalah contoh catatan kesalahan.


```
{
```

```
 "attemptsMade":5,
 "arrivalTimestamp":1594265943615,
 "errorCode":"HttpEndpoint.DestinationException",
 "errorMessage":"Received the following response from the endpoint destination.
 {"requestId": "109777ac-8f9b-4082-8e8d-b4f12b5fc17b", "timestamp": 1594266081268,
 "errorMessage": "Unauthorized"}",
 "attemptEndingTimestamp":1594266081318,
 "rawData":"c2FtcGxlIHJhdyBkYXRh",
 "subsequenceNumber":0,
 "dataId":"49607357361271740811418664280693044274821622880012337186.0"
}
```

Kepingan salju

Untuk tujuan Snowflake, saat membuat aliran Firehose, Anda dapat menentukan durasi coba ulang opsional (0-7200 detik). Nilai default untuk durasi coba lagi adalah 60 detik.

Pengiriman data ke tabel Snowflake Anda mungkin gagal karena beberapa alasan seperti konfigurasi tujuan Snowflake yang salah, pemadaman kepingan salju, kegagalan jaringan, dll. Kebijakan coba lagi tidak berlaku untuk kesalahan yang tidak dapat diambil kembali. Misalnya, jika Snowflake menolak muatan JSON Anda karena memiliki kolom tambahan yang hilang di tabel, Firehose tidak mencoba mengirimkannya lagi. Sebagai gantinya, ini membuat cadangan untuk semua kegagalan penyisipan karena masalah payload JSON ke ember kesalahan S3 Anda.

Demikian pula, jika pengiriman gagal karena peran, tabel, atau database yang salah, Firehose tidak mencoba lagi dan menulis data ke bucket S3 Anda. Durasi coba lagi hanya berlaku untuk kegagalan karena masalah layanan Snowflake, gangguan jaringan sementara, dll. Dalam kondisi ini, Firehose mencoba ulang untuk durasi waktu yang ditentukan sebelum mengirimkannya ke S3. Catatan yang gagal dikirimkan dalam folder snowflake-failed/, yang dapat Anda gunakan untuk pengisian ulang manual.

Berikut ini adalah contoh JSON untuk setiap record yang Anda kirimkan ke S3.

```
{
    "attemptsMade": 3,
    "arrivalTimestamp": 1594265943615,
    "errorCode": "Snowflake.InvalidColumns",
    "errorMessage": "Snowpipe Streaming does not support columns of type AUTOINCREMENT,
 IDENTITY, GEO, or columns with a default value or collation",
    "attemptEndingTimestamp": 1712937865543,
    "rawData": "c2FtcGxlIHJhdyBkYXRh"
```

```
}
```

Konfigurasikan format nama objek Amazon S3

Ketika Firehose mengirimkan data ke Amazon S3, nama kunci objek S3 mengikuti format <evaluated prefix><suffix>, di mana akhiran memiliki format - - - - - - - - <Firehose stream name><Firehose stream version><year><month><day><hour><minute><second><uuid><file extension><Firehose stream version>dimulai dengan 1 dan meningkat 1 untuk setiap perubahan konfigurasi aliran Firehose. Anda dapat mengubah konfigurasi aliran Firehose (misalnya, nama bucket S3, petunjuk buffering, kompresi, dan enkripsi). Anda dapat melakukannya dengan menggunakan Firehose console atau operasi UpdateDestinationAPI.

Untuk<evaluated prefix>, Firehose menambahkan awalan waktu default dalam format. `YYYY/MM/dd/HH` Awalan ini menciptakan hierarki logis di bucket, di mana setiap garis miring maju (/) menciptakan level dalam hierarki. Anda dapat memodifikasi struktur ini dengan menentukan awalan kustom yang menyertakan ekspresi yang dievaluasi saat runtime. Untuk informasi tentang cara menentukan awalan kustom, lihat Awalan Kustom untuk Objek Layanan Penyimpanan Sederhana Amazon.

Secara default, zona waktu yang digunakan untuk awalan waktu dan akhiran ada di UTC, tetapi Anda dapat mengubahnya ke zona waktu yang Anda inginkan. Misalnya, untuk menggunakan Waktu Standar Jepang alih-alih UTC, Anda dapat mengonfigurasi zona waktu Asia/Tokyo ke dalam atau dalam Konsol Manajemen AWS pengaturan parameter API CustomTimeZone (). Daftar berikut berisi zona waktu yang didukung Firehose untuk konfigurasi awalan S3.

Zona waktu yang didukung

Berikut ini adalah daftar zona waktu yang Firehose mendukung untuk konfigurasi awalan S3.

Africa

```
Africa/Abidjan
Africa/Accra
Africa/Addis_Ababa
Africa/Algiers
Africa/Asmera
Africa/Bangui
Africa/Banjul
Africa/Bissau
Africa/Blantyre
```

```
Africa/Bujumbura
Africa/Cairo
Africa/Casablanca
Africa/Conakry
Africa/Dakar
Africa/Dar_es_Salaam
Africa/Djibouti
Africa/Douala
Africa/Freetown
Africa/Gaborone
Africa/Harare
Africa/Johannesburg
Africa/Kampala
Africa/Khartoum
Africa/Kigali
Africa/Kinshasa
Africa/Lagos
Africa/Libreville
Africa/Lome
Africa/Luanda
Africa/Lubumbashi
Africa/Lusaka
Africa/Malabo
Africa/Maputo
Africa/Maseru
Africa/Mbabane
Africa/Mogadishu
Africa/Monrovia
Africa/Nairobi
Africa/Ndjamena
Africa/Niamey
Africa/Nouakchott
Africa/Ouagadougou
Africa/Porto-Novo
Africa/Sao_Tome
Africa/Timbuktu
Africa/Tripoli
Africa/Tunis
Africa/Windhoek
```

America

```
America/Adak
```

```
America/Anchorage
America/Anguilla
America/Antigua
America/Aruba
America/Asuncion
America/Barbados
America/Belize
America/Bogota
America/Buenos_Aires
America/Caracas
America/Cayenne
America/Cayman
America/Chicago
America/Costa_Rica
America/Cuiaba
America/Curacao
America/Dawson_Creek
America/Denver
America/Dominica
America/Edmonton
America/El_Salvador
America/Fortaleza
America/Godthab
America/Grand_Turk
America/Grenada
America/Guadeloupe
America/Guatemala
America/Guayaquil
America/Guyana
America/Halifax
America/Havana
America/Indianapolis
America/Jamaica
America/La_Paz
America/Lima
America/Los_Angeles
America/Managua
America/Manaus
America/Martinique
America/Mazatlan
America/Mexico_City
America/Miquelon
America/Montevideo
America/Montreal
```

```
America/Montserrat
America/Nassau
America/New_York
America/Noronha
America/Panama
America/Paramaribo
America/Phoenix
America/Port_of_Spain
America/Port-au-Prince
America/Porto_Acre
America/Puerto_Rico
America/Regina
America/Rio_Branco
America/Santiago
America/Santo_Domingo
America/Sao_Paulo
America/Scoresbysund
America/St_Johns
America/St_Kitts
America/St_Lucia
America/St_Thomas
America/St_Vincent
America/Tegucigalpa
America/Thule
America/Tijuana
America/Tortola
America/Vancouver
America/Winnipeg
```

Antarctica

```
Antarctica/Casey
Antarctica/DumontDUrville
Antarctica/Mawson
Antarctica/McMurdo
Antarctica/Palmer
```

Asia

```
Asia/Aden
Asia/Almaty
Asia/Amman
Asia/Anadyr
```

```
Asia/Aqtau
Asia/Aqtobe
Asia/Ashgabat
Asia/Ashkhabad
Asia/Baghdad
Asia/Bahrain
Asia/Baku
Asia/Bangkok
Asia/Beirut
Asia/Bishkek
Asia/Brunei
Asia/Calcutta
Asia/Colombo
Asia/Dacca
Asia/Damascus
Asia/Dhaka
Asia/Dubai
Asia/Dushanbe
Asia/Hong_Kong
Asia/Irkutsk
Asia/Jakarta
Asia/Jayapura
Asia/Jerusalem
Asia/Kabul
Asia/Kamchatka
Asia/Karachi
Asia/Katmandu
Asia/Krasnoyarsk
Asia/Kuala_Lumpur
Asia/Kuwait
Asia/Macao
Asia/Magadan
Asia/Manila
Asia/Muscat
Asia/Nicosia
Asia/Novosibirsk
Asia/Phnom_Penh
Asia/Pyongyang
Asia/Qatar
Asia/Rangoon
Asia/Riyadh
Asia/Saigon
Asia/Seoul
Asia/Shanghai
```

```
Asia/Singapore
Asia/Taipei
Asia/Tashkent
Asia/Tbilisi
Asia/Tehran
Asia/Thimbu
Asia/Thimphu
Asia/Tokyo
Asia/Ujung_Pandang
Asia/Ulaanbaatar
Asia/Ulan_Bator
Asia/Vientiane
Asia/Vladivostok
Asia/Yakutsk
Asia/Yekaterinburg
Asia/Yerevan
```

Atlantic

```
Atlantic/Azores
Atlantic/Bermuda
Atlantic/Canary
Atlantic/Cape_Verde
Atlantic/Faeroe
Atlantic/Jan_Mayen
Atlantic/Reykjavik
Atlantic/South_Georgia
Atlantic/St_Helena
Atlantic/Stanley
```

Australia

```
Australia/Adelaide
Australia/Brisbane
Australia/Broken_Hill
Australia/Darwin
Australia/Hobart
Australia/Lord_Howe
Australia/Perth
Australia/Sydney
```

Europe

```
Europe/Amsterdam
Europe/Andorra
Europe/Athens
Europe/Belgrade
Europe/Berlin
Europe/Brussels
Europe/Bucharest
Europe/Budapest
Europe/Chisinau
Europe/Copenhagen
Europe/Dublin
Europe/Gibraltar
Europe/Helsinki
Europe/Istanbul
Europe/Kaliningrad
Europe/Kiev
Europe/Lisbon
Europe/London
Europe/Luxembourg
Europe/Madrid
Europe/Malta
Europe/Minsk
Europe/Monaco
Europe/Moscow
Europe/Oslo
Europe/Paris
Europe/Prague
Europe/Riga
Europe/Rome
Europe/Samara
Europe/Simferopol
Europe/Sofia
Europe/Stockholm
Europe/Tallinn
Europe/Tirane
Europe/Vaduz
Europe/Vienna
Europe/Vilnius
Europe/Warsaw
Europe/Zurich
```

Indian

```
Indian/Antananarivo
Indian/Chagos
Indian/Christmas
Indian/Cocos
Indian/Comoro
Indian/Kerguelen
Indian/Mahe
Indian/Maldives
Indian/Mauritius
Indian/Mayotte
Indian/Reunion
```

Pacific

```
Pacific/Apia
Pacific/Auckland
Pacific/Chatham
Pacific/Easter
Pacific/Efate
Pacific/Enderbury
Pacific/Fakaofo
Pacific/Fiji
Pacific/Funafuti
Pacific/Galapagos
Pacific/Gambier
Pacific/Guadalcanal
Pacific/Guam
Pacific/Honolulu
Pacific/Kiritimati
Pacific/Kosrae
Pacific/Majuro
Pacific/Marquesas
Pacific/Nauru
Pacific/Niue
Pacific/Norfolk
Pacific/Noumea
Pacific/Pago_Pago
Pacific/Palau
Pacific/Pitcairn
Pacific/Ponape
Pacific/Port_Moresby
```

```
Pacific/Rarotonga
Pacific/Saipan
Pacific/Tahiti
Pacific/Tarawa
Pacific/Tongatapu
Pacific/Truk
Pacific/Wake
Pacific/Wallis
```

<file extension>Anda tidak dapat mengubah bidang akhiran kecuali. Saat Anda mengaktifkan konversi atau kompresi format data, Firehose akan menambahkan ekstensi file berdasarkan konfigurasi. Tabel berikut menjelaskan ekstensi file default yang ditambahkan oleh Firehose:

Konfigurasi	Ekstensi file
Konversi Format Data: Parket	.parket
Konversi Format Data: ORC	.orc
Kompresi: Gzip	.gz
Kompresi: Zip	.zip
Kompresi: Snappy	.tajam
Kompresi: Hadoop-Snappy	.hsnappy

Anda juga dapat menentukan ekstensi file yang Anda inginkan di konsol Firehose atau API. Ekstensi file harus dimulai dengan titik (.) dan dapat berisi karakter yang diizinkan: 0-9a-z! -_.*' (). Ekstensi file tidak boleh melebihi 128 karakter.

> **Note**
>
> Saat Anda menentukan ekstensi file, itu akan mengganti ekstensi file default yang ditambahkan Firehose saat konversi atau kompresi format data diaktifkan.

Memahami awalan khusus untuk objek Amazon S3

<evaluated prefix><suffix>Objek yang dikirim ke Amazon S3 mengikuti format nama. Anda dapat menentukan awalan kustom yang menyertakan ekspresi yang dievaluasi saat runtime. Awalan kustom yang Anda tentukan akan mengganti awalan default. `yyyy/MM/dd/HH`

Anda dapat menggunakan ekspresi formulir berikut dalam awalan kustom Anda: `!{namespace:value}`, di mana `namespace` bisa menjadi salah satu dari yang berikut, seperti yang dijelaskan di bagian berikut.

- `firehose`
- `timestamp`
- `partitionKeyFromQuery`
- `partitionKeyFromLambda`

Jika prefiks berakhir dengan garis miring (/), itu akan muncul sebagai folder dalam bucket Amazon S3. Untuk informasi selengkapnya, lihat Format Nama Objek Amazon S3 di Panduan Data FirehoseDeveloper Amazon.

`timestamp`namespace

Nilai yang valid untuk namespace ini adalah string yang merupakan string Java yang valid. DateTimeFormatter Sebagai contoh, pada tahun 2018, ekspresi `!{timestamp:yyyy}` memberi nilai `2018`.

Saat mengevaluasi stempel waktu, Firehose menggunakan perkiraan stempel waktu kedatangan dari catatan tertua yang terdapat dalam objek Amazon S3 yang sedang ditulis.

Secara default, stempel waktu ada di UTC. Tapi, Anda dapat menentukan zona waktu yang Anda inginkan. Misalnya, Anda dapat mengonfigurasi zona waktu ke Asia/Tokyo dalam Konsol Manajemen AWS atau dalam pengaturan parameter API (CustomTimeZone) jika Anda ingin menggunakan Waktu Standar Jepang, bukan UTC. Untuk melihat daftar zona waktu yang didukung, lihat Format Nama Objek Amazon S3.

Jika Anda menggunakan namespace `timestamp` lebih dari sekali dalam ekspresi prefiks yang sama, setiap instans akan memberi nilai waktu yang sama.

firehosenamespace

Ada dua nilai yang dapat Anda gunakan dengan namespace ini: `error-output-type` dan `random-string`. Tabel berikut menjelaskan cara menggunakannya.

Nilai namespace **`firehose`**

Konversi	Deskripsi	Contoh input	Contoh Output	Catatan
`error-output-type`	Mengevaluasi ke salah satu string berikut, tergantung pada konfigurasi aliran Firehose Anda, dan alasan kegagalan: {processing-failed, AmazonOpenSearchService-failed, splunk-failed,,}. format-conversion-failed http-endpoint-failed Jika Anda menggunakannya lebih dari sekali dalam ekspresi yang sama, setiap instans akan memberi nilai string kesalahan yang sama.	`myPrefix/result=!{firehose:error-output-type}/!{timestamp:yyyy/MM/dd}`	`myPrefix/result=processing-failed/2018/08/03`	error-output-typeNilai hanya dapat digunakan di ErrorOutputPrefix lapangan.

Konversi	Deskripsi	Contoh input	Contoh Output	Catatan
`random-string`	Memberi nilai string acak berisi 11 karakter. Jika Anda menggunak annya lebih dari sekali dalam ekspresi yang sama, setiap instans akan memberi nilai string acak baru.	`myPrefix/!{firehose:random-string}/`	`myPrefix/046b6c7f-0b/`	Anda dapat menggunak annya dengan kedua jenis prefiks tersebut. Anda dapat menempatk annya di awal string format untuk mendapatk an prefiks acak, yang kadang-ka dang diperlukan untuk mencapai throughput yang sangat tinggi dengan Amazon S3.

`partitionKeyFromLambda`dan ruang **`partitionKeyFromQuery`** nama

Untuk partisi dinamis, Anda harus menggunakan format ekspresi berikut di awalan bucket S3:`!{namespace:value}`, di mana namespace dapat berupa atau, atau keduanya. `partitionKeyFromQuery` `partitionKeyFromLambda` Jika Anda menggunakan penguraian inline untuk membuat kunci partisi untuk data sumber Anda, Anda harus menentukan nilai awalan bucket S3 yang terdiri dari ekspresi yang ditentukan dalam format berikut:. `"partitionKeyFromQuery:keyID"` Jika Anda menggunakan fungsi AWS Lambda untuk membuat kunci partisi untuk data sumber Anda, Anda harus menentukan nilai awalan bucket S3 yang terdiri dari ekspresi yang ditentukan dalam format berikut:. `"partitionKeyFromLambda:keyID"` Untuk informasi selengkapnya, lihat “Pilih Amazon S3 untuk Tujuan Anda” di Membuat aliran Amazon Firehose.

Aturan semantik

Aturan berikut berlaku untuk ekspresi `Prefix` dan `ErrorOutputPrefix`.

- Untuk namespace `timestamp`, setiap karakter yang tidak diapit tanda kutip tunggal akan diberi nilai. Dengan kata lain, string yang tidak menggunakan tanda kutip tunggal di bidang nilai akan ditafsirkan secara harfiah.
- Jika Anda menentukan awalan yang tidak berisi ekspresi namespace stempel waktu, Firehose menambahkan ekspresi ke nilai di `!{timestamp:yyyy/MM/dd/HH/}` bidang. `Prefix`
- Urutan `!{` hanya dapat muncul dalam ekspresi `!{namespace:value}`.
- `ErrorOutputPrefix` boleh null hanya jika `Prefix` tidak mengandung ekspresi apa pun. Dalam kasus ini, `Prefix` memberi nilai `<specified-prefix>yyyy/MM/DDD/HH/` dan `ErrorOutputPrefix` memberi nilai `<specified-prefix><error-output-type>yyyy/MM/DDD/HH/`. `DDD` mewakili hari ke berapa dalam tahun itu.
- Jika Anda menentukan ekspresi untuk `ErrorOutputPrefix`, Anda harus menyertakan setidaknya satu instans dari `!{firehose:error-output-type}`.
- `Prefix` tidak boleh berisi `!{firehose:error-output-type}`.
- Baik `Prefix` maupun `ErrorOutputPrefix` boleh berisi lebih dari 512 karakter setelah diberi nilai.
- Jika tujuannya adalah Amazon Redshift, `Prefix` tidak boleh berisi ekspresi dan `ErrorOutputPrefix` harus null.
- Jika tujuannya adalah Amazon OpenSearch Service atau Splunk, dan tidak `ErrorOutputPrefix` ditentukan, Firehose menggunakan bidang untuk `Prefix` catatan gagal.
- Ketika tujuannya adalah Amazon S3, `Prefix` dan `ErrorOutputPrefix` dalam konfigurasi tujuan Amazon S3 secara berurutan digunakan untuk catatan yang berhasil dan catatan yang gagal. Jika Anda menggunakan AWS CLI atau API, Anda dapat menggunakan `ExtendedS3DestinationConfiguration` untuk menentukan konfigurasi pencadangan Amazon S3 dengan `Prefix` dan `ErrorOutputPrefix`-nya sendiri.
- Saat Anda menggunakan Konsol Manajemen AWS dan mengatur tujuan ke Amazon S3, Firehose masing-masing menggunakan `Prefix` dan `ErrorOutputPrefix` dalam konfigurasi tujuan untuk catatan yang berhasil dan catatan gagal. Jika Anda menentukan awalan menggunakan ekspresi, Anda harus menentukan awalan kesalahan termasuk. `!{firehose:error-output-type}`
- Saat Anda menggunakan `ExtendedS3DestinationConfiguration` AWS CLI, API, atau CloudFormation, jika Anda menentukan`S3BackupConfiguration`, Firehose tidak menyediakan default. `ErrorOutputPrefix`

- Anda tidak dapat menggunakan `partitionKeyFromLambda` dan `partitionKeyFromQuery` ruang nama saat membuat ErrorOutputPrefix ekspresi.

Contoh prefiks

Contoh **`Prefix`** dan **`ErrorOutputPrefix`**

Input	Prefiks yang diberi nilai (pukul 10:30 UTC pada 27 Agt 2018)

`Prefix`: Tidak ditentukan
 	
`ErrorOutputPrefix` : `myFirehoseFailures/!{firehose:error-output-type}/`
	`Prefix`: `2018/08/27/10`
	
	`ErrorOutputPrefix` : `myFirehoseFailures/processing-failed/`

`Prefix`: `!{timestamp:yyyy/MM/dd}` `ErrorOutputPrefix` : Tidak ditentukan	Input tidak valid: `ErrorOutputPrefix` tidak boleh null ketika Prefiks berisi ekspresi

`Prefix`: `myFirehose/DeliveredYear=!{timestamp:yyyy}/anyMonth/rand=!{firehose:random-string}`
 	
`ErrorOutputPrefix` : `myFirehoseFailures/!{firehose:error-output-type}/!{timestamp:yyyy}/anyMonth/!{timestamp:dd}`
	`Prefix`: `myFirehose/DeliveredYear=2018/anyMonth/rand=5abf82daaa5`
	
	`ErrorOutputPrefix` : `myFirehoseFailures/processing-failed/2018/anyMonth/10`

`Prefix`: `myPrefix/year=!{timestamp:yyyy}/month=!{timestamp:MM}/day=!{timestamp:dd}/hour=!{timestamp:HH}/`
 	
`ErrorOutputPrefix` : `myErrorPrefix/year=!{timestamp:yyyy}/month=!{timestamp:MM}/day=!{timestamp:dd}/hour=!{timestamp:HH}/!{firehose:error-output-type}`
	`Prefix`: `myPrefix/year=2018/month=07/day=06/hour=23/`
	
	`ErrorOutputPrefix` : `myErrorPrefix/year=2018/month=07/day=06/hour=23/processing-failed`

Input	Prefiks yang diberi nilai (pukul 10:30 UTC pada 27 Agt 2018)

`Prefix`: `myFirehosePrefix/`
 	
`ErrorOutputPrefix` : Tidak ditentukan
	`Prefix`: `myFirehosePrefix/2018/08/27/`
	
	`ErrorOutputPrefix` : `myFirehosePrefix/processing-failed/2018/08/27/`

Konfigurasikan rotasi indeks untuk OpenSearch Layanan

Untuk tujuan OpenSearch Layanan, Anda dapat menentukan opsi rotasi indeks berbasis waktu dari salah satu dari lima opsi berikut:NoRotation,,OneHour, OneDayOneWeek, atauOneMonth.

Bergantung pada opsi rotasi yang Anda pilih, Amazon Data Firehose menambahkan sebagian stempel waktu kedatangan UTC ke nama indeks yang Anda tentukan. Kinesis Data Firehose memutar stempel waktu yang ditambahkan dengan sesuai. Contoh berikut menunjukkan nama indeks yang dihasilkan di OpenSearch Layanan untuk setiap opsi rotasi indeks, di mana nama indeks yang ditentukan myindex dan stempel waktu kedatangan. `2016-02-25T13:00:00Z`

RotationPeriod	IndexName
`NoRotation`	`myindex`
`OneHour`	`myindex-2016-02-25-13`
`OneDay`	`myindex-2016-02-25`
`OneWeek`	`myindex-2016-w08`
`OneMonth`	`myindex-2016-02`

> **Note**
>
> Dengan opsi `OneWeek`, Data Firehose secara otomatis membuat indeks menggunakan format <YEAR>-w<WEEK NUMBER> (sebagai contoh, `2020-w33`), dengan jumlah minggu dihitung menggunakan waktu UTC dan sesuai dengan konvensi AS berikut:
>
> - Seminggu dimulai pada hari Minggu
> - Minggu pertama tahun ini adalah minggu pertama yang berisi hari Sabtu di tahun ini

Jeda dan lanjutkan pengiriman data

Setelah Anda menyiapkan aliran Firehose, data yang tersedia di sumber aliran akan terus dikirim ke tujuan. Jika Anda mengalami situasi di mana tujuan streaming sementara tidak tersedia (misalnya, selama operasi pemeliharaan yang direncanakan), Anda mungkin ingin menghentikan sementara pengiriman data, dan melanjutkan ketika tujuan tersedia lagi.

> **Important**
>
> Saat Anda menggunakan pendekatan yang dijelaskan di bawah ini untuk menjeda dan melanjutkan aliran, setelah melanjutkan streaming, Anda akan melihat bahwa beberapa catatan dikirim ke keranjang kesalahan di Amazon S3 sementara sisa aliran terus dikirim ke tujuan. Ini adalah batasan pendekatan yang diketahui, dan itu terjadi karena sejumlah kecil catatan yang sebelumnya tidak dapat dikirim ke tujuan setelah beberapa percobaan ulang dilacak sebagai gagal.

Jeda aliran Firehose

Untuk menjeda pengiriman streaming di Firehose, pertama-tama hapus izin Firehose untuk menulis ke lokasi cadangan S3 untuk pengiriman yang gagal. Misalnya, jika Anda ingin menjeda aliran Firehose dengan OpenSearch tujuan, Anda dapat melakukannya dengan memperbarui izin. Untuk informasi selengkapnya, lihat Memberikan Akses Firehose ke Tujuan OpenSearch Layanan Publik.

Hapus `"Effect": "Allow"` izin untuk tindakan`s3:PutObject`, dan secara eksplisit tambahkan pernyataan yang menerapkan `Effect": "Deny"` izin pada tindakan `s3:PutObject` untuk bucket S3 yang digunakan untuk mencadangkan pengiriman yang gagal. Selanjutnya, matikan tujuan streaming (misalnya, mematikan OpenSearch domain tujuan), atau hapus izin untuk Firehose

untuk menulis ke tujuan. Untuk memperbarui izin untuk tujuan lain, periksa bagian tujuan Anda di Mengontrol Akses dengan Amazon Data Firehose. Setelah Anda menyelesaikan dua tindakan ini, Firehose akan berhenti mengirimkan aliran, dan Anda dapat memantau ini menggunakan CloudWatch metrik untuk Firehose.

> **Important**
>
> Saat Anda menjeda pengiriman streaming di Firehose, Anda perlu memastikan bahwa sumber aliran (misalnya, di Kinesis Data Streams atau di Layanan Terkelola untuk Kafka) dikonfigurasi untuk menyimpan data hingga pengiriman aliran dilanjutkan dan data dikirim ke tujuan. Jika sumbernya DirectPut, Firehose akan menyimpan data selama 24 jam. Kehilangan data dapat terjadi jika Anda tidak melanjutkan aliran dan mengirimkan data sebelum berakhirnya periode penyimpanan data.

Lanjutkan aliran Firehose

Untuk melanjutkan pengiriman, pertama-tama kembalikan perubahan yang dilakukan sebelumnya ke tujuan streaming dengan menyalakan tujuan dan memastikan bahwa Firehose memiliki izin untuk mengirimkan aliran ke tujuan. Selanjutnya, kembalikan perubahan yang dibuat sebelumnya ke izin yang diterapkan ke bucket S3 untuk mencadangkan pengiriman yang gagal. Yaitu, terapkan `"Effect": "Allow"` izin untuk tindakan tersebut`s3:PutObject`, dan hapus `"Effect": "Deny"` izin pada tindakan `s3:PutObject` untuk bucket S3 yang digunakan untuk mencadangkan pengiriman yang gagal. Terakhir, pantau menggunakan CloudWatch metrik untuk Firehose untuk mengonfirmasi bahwa aliran sedang dikirim ke tujuan. Untuk melihat dan memecahkan masalah kesalahan, gunakan pemantauan Amazon CloudWatch Logs untuk Firehose.

Mengirimkan data ke Apache Iceberg Tables dengan Amazon Data Firehose

Apache Iceberg adalah format tabel open-source berkinerja tinggi untuk melakukan analisis data besar. Apache Iceberg menghadirkan keandalan dan kesederhanaan tabel SQL ke data lake Amazon S3, dan memungkinkan mesin analitik open-source seperti Spark, Flink, Trino, Hive, dan Impala bekerja dengan data yang sama secara bersamaan. Untuk informasi lebih lanjut, lihat Apache Iceberg dan. Pertimbangan dan batasan

Anda dapat menggunakan Firehose untuk mengirimkan data streaming ke Apache Iceberg Tables di Amazon S3. Tabel Apache Iceberg Anda dapat dikelola sendiri di Amazon S3 atau dihosting di Tabel Amazon S3. Dalam tabel Iceberg yang dikelola sendiri, Anda mengelola semua pengoptimalan tabel seperti pemadatan, dan kedaluwarsa snapshot. Tabel Amazon S3 menyediakan penyimpanan yang dioptimalkan untuk beban kerja analitik skala besar, dengan fitur yang terus meningkatkan kinerja kueri dan mengurangi biaya penyimpanan untuk data tabular. Untuk informasi selengkapnya tentang Tabel Amazon S3, lihat Tabel Amazon S3.

Fitur ini memungkinkan Anda untuk merutekan catatan dari satu aliran ke Tabel Gunung Es Apache yang berbeda. Anda dapat secara otomatis menerapkan menyisipkan, memperbarui, dan menghapus operasi ke catatan dalam tabel tersebut. Ini juga mendukung kontrol akses data berbutir halus pada tabel Apache Iceberg di Amazon S3 dengan. AWS Lake Formation Anda dapat menentukan kontrol akses secara terpusat AWS Lake Formation dan memberikan izin tingkat tabel dan tingkat kolom yang lebih terperinci untuk Firehose.

Pertimbangan dan batasan

> **Note**
>
> Firehose mendukung Apache Iceberg Tables sebagai tujuan di semua kecuali Wilayah AWSWilayah Tiongkok, Asia Pasifik (Taipei) AWS GovCloud (US) Regions, Asia Pasifik (Malaysia), Asia Pasifik (Selandia Baru), dan Meksiko (Tengah).

Dukungan Firehose untuk tabel Apache Iceberg memiliki pertimbangan dan batasan berikut.

- Throughput — Jika Anda menggunakan Direct PUT sebagai sumber untuk mengirimkan data ke tabel Apache Iceberg, maka throughput maksimum per aliran adalah 5 MiB/second di Wilayah

AS Timur (Virginia N.), AS Barat (Oregon), dan Eropa (Irlandia) dan 1 di semua lainnya. MiB/second Wilayah AWS Jika Anda ingin menyisipkan data ke tabel Iceberg tanpa pembaruan dan penghapusan dan Anda menginginkan throughput yang lebih tinggi untuk streaming Anda, maka Anda dapat menggunakan formulir Batas Firehose untuk meminta peningkatan batas throughput.

Anda juga dapat mengatur `AppendOnly` bendera `True` jika Anda hanya ingin menyisipkan data dan tidak melakukan pembaruan dan penghapusan. Dengan menyetel `AppendOnly` bendera ke`True`, Firehose secara otomatis menskalakan agar sesuai dengan throughput Anda. Saat ini, Anda dapat mengatur flag ini hanya dengan operasi CreateDeliveryStreamAPI.

Jika aliran PUT Langsung mengalami pelambatan karena volume konsumsi data yang lebih tinggi yang melebihi kapasitas throughput aliran Firehose, Firehose secara otomatis meningkatkan batas throughput aliran hingga pelambatan terisi. Bergantung pada peningkatan throughput dan throttling, Firehose mungkin membutuhkan waktu lebih lama untuk meningkatkan throughput aliran ke tingkat yang diinginkan. Karena itu, terus coba lagi catatan menelan data yang gagal. Jika Anda mengharapkan volume data meningkat dalam ledakan besar mendadak, atau jika aliran baru Anda membutuhkan throughput yang lebih tinggi daripada batas throughput default, mintalah untuk meningkatkan batas throughput.

- Throughput dan Penskalaan Partisi - Layanan ini dioptimalkan untuk mendukung sejumlah besar partisi Gunung Es atau throughput konsumsi yang sangat tinggi. Ketika throughput konsumsi meningkat, jumlah partisi yang dapat ditulis secara aktif berkurang.

  Berikut adalah batasan untuk throughput konsumsi dan partisi aktif maksimal yang didukung.

Throughput Tertelan	Partisi Aktif Maks didukung
≤ 20 MB/s	Hingga ~ 3.000
20—40 MB/s	1000
40—400 MB/s	100
400—750 MB/s	50
750 MB/s–1.5 GB/s	1

- Transaksi S3 Per Detik (TPS) - Untuk mengoptimalkan kinerja S3, jika Anda menggunakan Kinesis Data Streams atau Amazon MSK sebagai sumber, kami sarankan Anda mempartisi rekaman sumber menggunakan kunci partisi yang tepat. Dengan cara itu, catatan data yang dirutekan ke

tabel Iceberg yang sama dipetakan ke satu atau beberapa partisi sumber yang dikenal sebagai pecahan. Jika memungkinkan, sebarkan catatan data milik tabel Iceberg target yang berbeda menjadi berbeda. partitions/shards, so that you can use all the aggregate throughput available across all the partitions/shards of the source topic/stream

- Kolom - Untuk nama dan nilai kolom, Firehose hanya mengambil tingkat pertama node dalam JSON bersarang multi-level. Misalnya, Firehose memilih node yang tersedia di tingkat pertama termasuk bidang posisi. Nama kolom dan tipe data dari data sumber harus sama persis dengan tabel target agar Firehose berhasil dikirimkan. Dalam kasus ini, Firehose mengharapkan Anda memiliki kolom tipe data struct atau map di tabel Iceberg agar sesuai dengan bidang posisi. Firehose mendukung 16 tingkat bersarang. Berikut ini adalah contoh dari JSON bersarang.

```
{
   "version":"2016-04-01",
   "deviceId":"<solution_unique_device_id>",
   "sensorId":"<device_sensor_id>",
   "timestamp":"2024-01-11T20:42:45.000Z",
   "value":"<actual_value>",
   "position":{
      "x":143.595901,
      "y":476.399628,
      "z":0.24234876
   }
}
```

  Jika nama kolom atau tipe data tidak cocok, Firehose akan melempar kesalahan dan mengirimkan data ke bucket kesalahan S3. Jika semua nama kolom dan tipe data cocok dalam tabel Apache Iceberg, tetapi Anda memiliki bidang tambahan yang ada dalam catatan sumber, Firehose melewatkan bidang baru.

- Satu objek JSON per catatan - Anda hanya dapat mengirim satu objek JSON dalam satu catatan Firehose. Jika Anda menggabungkan dan mengirim beberapa objek JSON di dalam rekaman, Firehose akan melempar kesalahan dan mengirimkan data ke bucket kesalahan S3. Jika Anda menggabungkan rekaman dengan KPL dan menyerap data ke Firehose dengan Amazon Kinesis Data Streams sebagai sumber, Firehose secara otomatis melakukan de-agregasi dan menggunakan satu objek JSON per rekaman.
- Optimasi pemadatan dan penyimpanan — Setiap kali Anda menulis ke Iceberg Tables menggunakan Firehose, ia melakukan dan menghasilkan snapshot, file data, dan menghapus file. Memiliki banyak file data meningkatkan overhead metadata dan memengaruhi kinerja baca. Untuk mendapatkan kinerja kueri yang efisien, Anda mungkin ingin mempertimbangkan solusi

yang secara berkala mengambil file data kecil dan menulis ulang mereka menjadi lebih sedikit file data yang lebih besar. Proses ini disebut pemadatan. AWS Glue Data Catalog mendukung pemadatan otomatis dari Apache Iceberg Tables Anda. Untuk informasi selengkapnya, lihat Manajemen pemadatan di Panduan Pengguna AWS Glue. Untuk informasi tambahan, lihat Pemadatan otomatis Tabel Gunung Es Apache. Atau, Anda dapat menjalankan perintah Athena Optimize untuk melakukan pemadatan secara manual. Untuk informasi selengkapnya tentang perintah Optimalkan, lihat Athena Optimize.

Selain pemadatan file data, Anda juga dapat mengoptimalkan konsumsi penyimpanan dengan pernyataan VACUUM yang melakukan pemeliharaan tabel pada tabel Apache Iceberg, seperti kedaluwarsa snapshot dan penghapusan file yatim piatu. Atau, Anda dapat menggunakannya AWS Glue Data Catalog juga mendukung optimasi tabel terkelola tabel Apache Iceberg dengan secara otomatis menghapus file data, file yatim piatu, dan snapshot kedaluwarsa yang tidak lagi diperlukan. Untuk informasi lebih lanjut, lihat posting blog ini tentang Optimalisasi penyimpanan Apache Iceberg Tables.

- Kami tidak mendukung sumber Amazon MSK Tanpa Server untuk Apache Iceberg Tables sebagai tujuan.
- Untuk operasi pembaruan, Firehose menempatkan file hapus diikuti dengan operasi penyisipan. Menempatkan file hapus menimbulkan biaya Amazon S3.
- Firehose tidak merekomendasikan penggunaan beberapa aliran Firehose untuk menulis data ke tabel Apache Iceberg yang sama. Ini karena Apache Iceberg bergantung pada Optimistic Concurrency Control (OCC). Jika beberapa aliran Firehose mencoba menulis ke satu tabel Iceberg secara bersamaan, maka hanya satu aliran yang berhasil melakukan data pada waktu tertentu. Aliran lain yang gagal melakukan back-off, dan coba lagi operasi komit hingga durasi percobaan ulang yang dikonfigurasi berakhir. Setelah durasi coba lagi habis, data dan hapus kunci file (jalur Amazon S3) dikirim ke awalan kesalahan Amazon S3 yang dikonfigurasi.
- Versi Iceberg Library saat ini yang didukung Firehose adalah versi 1.5.2.
- Untuk mengirimkan data terenkripsi ke Tabel Amazon S3, Anda harus AWS Key Management Service mengonfigurasi parameter di Tabel Amazon S3, dan bukan dalam konfigurasi Firehose. Jika Anda mengonfigurasi AWS Key Management Service parameter di Firehose untuk mengirimkan data terenkripsi ke Tabel Amazon S3, Firehose tidak dapat menggunakan parameter tersebut untuk mengenkripsi. Untuk informasi selengkapnya, lihat Menggunakan enkripsi sisi server dengan kunci. AWS KMS
- Firehose stream hanya mendukung pengiriman ke database dan tabel yang dibuat melalui API Iceberg. GlueCatalog Pengiriman ke database dan tabel yang dibuat melalui Glue SDK tidak didukung. Perhatikan bahwa tanda hubung (-) bukan karakter yang didukung untuk database dan

nama tabel di pustaka Iceberg. Untuk detail selengkapnya, lihat Glue Database Regex dan Glue Table Regex yang didukung oleh perpustakaan Iceberg.

- Semua file yang ditulis oleh Firehose dihitung menggunakan partisi yang ada dalam catatan. Ini juga berlaku untuk file yang dihapus. Penghapusan global, seperti menulis file hapus yang tidak dipartisi untuk tabel yang dipartisi, tidak didukung.
- Firehose saat ini tidak mendukung properti filter mekar saat mengirimkan data ke tabel Apache Iceberg. Ketika properti filter bloom dikonfigurasi pada tabel Iceberg, Firehose akan mengabaikan properti ini selama operasi pengiriman data.

Prasyarat untuk menggunakan Apache Iceberg Tables sebagai tujuan

Pilih dari opsi berikut untuk menyelesaikan prasyarat yang diperlukan.

Topik

- Prasyarat untuk dikirim ke Tabel Gunung Es di Amazon S3
- Prasyarat untuk dikirim ke Tabel Amazon S3

Prasyarat untuk dikirim ke Tabel Gunung Es di Amazon S3

Sebelum Anda mulai, lengkapi prasyarat berikut.

- Membuat bucket Amazon S3 — Anda harus membuat bucket Amazon S3 untuk menambahkan jalur file metadata selama pembuatan tabel. Untuk informasi selengkapnya, lihat Membuat bucket S3.
- Buat peran IAM dengan izin yang diperlukan - Firehose memerlukan peran IAM dengan izin khusus untuk mengakses AWS Glue tabel dan menulis data ke Amazon S3. Peran yang sama digunakan untuk memberikan AWS Glue akses ke bucket Amazon S3. Anda memerlukan peran IAM ini saat membuat Tabel Gunung Es dan aliran Firehose. Untuk informasi selengkapnya, lihat Berikan akses Firehose ke Tabel Amazon S3.
- Buat Tabel Gunung Es Apache - Jika Anda mengonfigurasi kunci unik di aliran Firehose untuk pembaruan dan penghapusan, Firehose memvalidasi jika tabel dan kunci unik ada sebagai bagian dari pembuatan aliran. Untuk skenario ini, Anda harus membuat tabel sebelum membuat aliran Firehose. Anda dapat menggunakan AWS Glue untuk membuat Apache Iceberg Tables. Untuk informasi selengkapnya, lihat Membuat tabel Apache Iceberg. Jika Anda tidak mengonfigurasi

kunci unik di aliran Firehose, maka Anda tidak perlu membuat tabel Iceberg sebelum membuat aliran Firehose.

> **Note**
>
> Firehose mendukung versi tabel berikut dan format untuk tabel Apache Iceberg.
>
> - Versi format tabel - Firehose hanya mendukung format tabel V2. Jangan membuat tabel dalam format V1, jika tidak, Anda mendapatkan kesalahan dan data dikirim ke bucket kesalahan S3 sebagai gantinya.
> - Format penyimpanan data — Firehose menulis data ke Apache Iceberg Tables dalam format Parket.
> - Operasi tingkat baris - Firehose mendukung mode penulisan data Merge-on-Read (MOR) ke Apache Iceberg Tables.

Prasyarat untuk dikirim ke Tabel Amazon S3

Untuk mengirimkan data ke bucket tabel Amazon S3, lengkapi prasyarat berikut.

- Buat bucket S3 Table, namespace, tabel di bucket tabel, dan langkah integrasi lainnya yang diuraikan dalam Memulai Tabel Amazon S3. Nama kolom harus huruf kecil karena keterbatasan yang diberlakukan oleh integrasi katalog Tabel S3, seperti yang ditentukan dalam batasan integrasi katalog tabel S3.
- Buat peran IAM dengan izin yang diperlukan - Firehose memerlukan peran IAM dengan izin khusus untuk mengakses AWS AWS Glue tabel dan menulis data ke tabel di bucket tabel Amazon S3. Untuk menulis ke tabel di keranjang tabel Amazon S3, Anda juga harus memberikan peran IAM dengan izin yang diperlukan. Izin yang diperlukan untuk katalog Amazon S3 Tables bergantung pada mode kontrol akses yang Anda gunakan:
  - Kontrol akses IAM - Peran pengiriman Firehose memerlukan izin IAM langsung di sumber daya Amazon S3 Tables.
  - Kontrol akses Lake Formation — Peran pengiriman Firehose memerlukan AWS AWS Lake Formation izin untuk mengelola akses ke sumber daya tabel Anda. AWS Lake Formation menggunakan model izinnya sendiri yang memungkinkan kontrol akses berbutir halus untuk sumber daya Katalog Data.

Anda mengonfigurasi peran IAM ini saat membuat aliran Firehose. Untuk informasi selengkapnya, lihat Memberikan akses Firehose ke Tabel Amazon S3.

Untuk step-by-step integrasi, lihat blog Membangun data lake untuk streaming data dengan Amazon S3 Tables dan Amazon Data Firehose. Untuk informasi tambahan, lihat juga Menggunakan Tabel Amazon S3 dengan layanan AWS analitik.

Siapkan aliran Firehose

Untuk membuat aliran Firehose dengan Apache Iceberg Tables sebagai tujuan Anda, Anda harus mengonfigurasi yang berikut ini.

> **Note**
>
> Penyiapan aliran Firehose untuk pengiriman ke tabel di bucket tabel S3 sama dengan Apache Iceberg Tables di Amazon S3.

Mengonfigurasi sumber dan tujuan

Untuk mengirimkan data ke Apache Iceberg Tables, pilih sumber untuk streaming Anda.

Untuk mengonfigurasi sumber untuk aliran, lihat Mengonfigurasi setelan sumber.

Selanjutnya, pilih Apache Iceberg Tables sebagai tujuan dan berikan nama aliran Firehose.

Konfigurasikan transformasi data

Untuk melakukan transformasi kustom pada data Anda, seperti menambahkan atau memodifikasi catatan di aliran masuk, Anda dapat menambahkan fungsi Lambda ke aliran Firehose Anda. Untuk informasi selengkapnya tentang transformasi data menggunakan Lambda dalam aliran Firehose, lihat. Mengubah data sumber di Amazon Data Firehose

Untuk Apache Iceberg Tables, Anda harus menentukan bagaimana Anda ingin merutekan catatan masuk ke tabel tujuan yang berbeda dan operasi yang ingin Anda lakukan. Salah satu cara untuk memberikan informasi perutean yang diperlukan ke Firehose adalah menggunakan fungsi Lambda.

Untuk informasi selengkapnya, lihat Merutekan catatan ke tabel Gunung Es yang berbeda.

Connect katalog data

Apache Iceberg membutuhkan katalog data untuk menulis ke Apache Iceberg Tables. Firehose terintegrasi dengan AWS Glue Data Catalog untuk Apache Iceberg Tables.

Anda dapat menggunakan AWS Glue Data Catalog akun yang sama dengan aliran Firehose Anda atau di lintas akun dan di Wilayah yang sama dengan aliran Firehose Anda (default), atau di Wilayah lain.

Jika Anda mengirim ke Tabel Amazon S3 dan menggunakan konsol untuk mengatur aliran Firehose, pilih katalog yang sesuai dengan katalog Tabel Amazon S3 Anda. Jika Anda menggunakan CLI untuk mengatur aliran Firehose Anda, maka dalam `CatalogConfiguration` input, gunakan `CatalogARN` dengan format:. `arn:aws:glue:<region>:<account-id>:catalog/s3tablescatalog/<s3 table bucket name>` Untuk informasi selengkapnya, lihat Menyiapkan aliran Firehose ke tabel Amazon S3.

> **Note**
>
> Firehose mendukung tiga operasi untuk tabel Iceberg: insert, update, dan delete. Tanpa operasi tertentu, Firehosedefaults untuk menyisipkan, menambahkan setiap catatan masuk sebagai baris baru, dan mempertahankan duplikat. Untuk memodifikasi catatan yang ada, tentukan operasi “perbarui”, yang menggunakan kunci utama untuk menemukan, dan mengubah baris yang ada.
> Contoh:
>
> - Default (insert): Beberapa catatan pelanggan identik membuat baris duplikat.
> - Pembaruan yang ditentukan: Alamat pelanggan baru memperbarui catatan yang ada.

Konfigurasikan ekspresi JQ

Untuk Apache Iceberg Tables, Anda harus menentukan bagaimana Anda ingin merutekan catatan masuk ke tabel tujuan yang berbeda dan operasi seperti menyisipkan, memperbarui, dan menghapus yang ingin Anda lakukan. Anda dapat melakukan ini dengan mengonfigurasi ekspresi JQ untuk Firehose untuk mengurai dan mendapatkan informasi yang diperlukan. Untuk informasi selengkapnya, lihat ???.

Konfigurasikan tombol unik

Pembaruan dan Penghapusan dengan lebih dari satu tabel — Kunci unik adalah satu atau beberapa bidang dalam catatan sumber Anda yang secara unik mengidentifikasi baris di Apache Iceberg Tables. Jika Anda hanya memasukkan skenario dengan lebih dari satu tabel, maka Anda tidak perlu mengkonfigurasi kunci unik. Jika Anda ingin melakukan pembaruan dan penghapusan pada tabel tertentu, maka Anda harus mengonfigurasi kunci unik untuk tabel yang diperlukan. Perhatikan bahwa pembaruan akan secara otomatis menyisipkan baris jika baris dalam tabel hilang. Jika Anda hanya memiliki satu tabel, maka Anda dapat mengonfigurasi kunci unik. Untuk operasi pembaruan, Firehose menempatkan file hapus diikuti dengan sisipan.

Anda dapat mengonfigurasi kunci unik per tabel sebagai bagian dari pembuatan aliran Firehose atau Anda dapat mengatur identifier-field-idssecara native di Iceberg selama membuat tabel atau mengubah operasi tabel. Mengkonfigurasi kunci unik per tabel selama pembuatan stream adalah opsional. Jika Anda tidak mengonfigurasi kunci unik per tabel selama pembuatan streaming, Firehose memeriksa tabel yang diperlukan dan akan menggunakannya sebagai kunci unik. `identifier-field-ids` Jika keduanya tidak dikonfigurasi, maka pengiriman data dengan operasi pembaruan dan penghapusan gagal.

Untuk mengonfigurasi bagian ini, berikan nama database, nama tabel, dan kunci unik untuk tabel tempat Anda ingin memperbarui atau menghapus data. Anda hanya dapat memiliki entri untuk setiap tabel dalam konfigurasi. Anda tidak perlu mengonfigurasi bagian ini untuk skenario tambahan saja. Secara opsional, Anda juga dapat memilih untuk memberikan awalan bucket error jika data dari tabel gagal dikirimkan seperti yang ditunjukkan pada contoh berikut.

```
[
  {
    "DestinationDatabaseName": "MySampleDatabase",
    "DestinationTableName": "MySampleTable",
    "UniqueKeys": [
      "COLUMN_PLACEHOLDER"
    ],
    "S3ErrorOutputPrefix": "OPTIONAL_PREFIX_PLACEHOLDER"
  }
]
```

Firehose mendukung konfigurasi kunci unik jika nama kolom yang disediakan unik di seluruh tabel. Namun, itu tidak mendukung nama kolom yang sepenuhnya memenuhi syarat sebagai kunci unik. Misalnya, kunci bernama tidak `top._id` dianggap sebagai kunci unik jika nama kolom juga `_id` ada

di tingkat atas. Jika `_id` unik di seluruh tabel, maka digunakan terlepas dari lokasinya dalam struktur tabel. Ini adalah apakah itu kolom tingkat atas atau kolom bersarang. Dalam contoh berikut, `_id` adalah kunci unik yang valid untuk skema karena nama kolom unik di seluruh skema.

```
[
 "schema": {
  "type": "struct",
  "fields": [
    {
      "name": "top",
      "type": {
        "type": "struct",
        "fields": [
          { "name": "_id", "type": "string" },
          { "name": "name", "type": "string" }
        ]
      }
    },
    { "name": "user", "type": "string" }
  ]
}
]
```

Dalam contoh berikut, `_id` bukan kunci unik yang valid untuk skema karena digunakan di kolom tingkat atas, dan struct bersarang.

```
[
"schema": {
  "type": "struct",
  "fields": [
    {
      "name": "top",
      "type": {
        "type": "struct",
        "fields": [
          { "name": "_id", "type": "string" },
          { "name": "name", "type": "string" }
        ]
      }
    },
    { "name": "_id", "type": "string" }
  ]
```

```
}
]
```

Tentukan durasi coba lagi

Anda dapat menggunakan konfigurasi ini untuk menentukan durasi dalam detik yang Firehose harus mencoba lagi, jika mengalami kegagalan secara tertulis ke Apache Iceberg Tables di Amazon S3. Anda dapat mengatur nilai apa pun dari 0 hingga 7200 detik untuk melakukan percobaan ulang. Secara default, Firehose mencoba lagi selama 300 detik.

Menangani pengiriman atau pemrosesan yang gagal

Anda harus mengonfigurasi Firehose untuk mengirimkan catatan ke bucket cadangan S3 jika mengalami kegagalan dalam memproses atau mengirimkan aliran setelah kedaluwarsa durasi percobaan ulang. Untuk ini, konfigurasikan awalan keluaran kesalahan bucket cadangan S3 dan bucket cadangan S3 dari pengaturan Backup di konsol.

Menangani kesalahan

Firehose mengirimkan semua kesalahan pengiriman ke CloudWatch Log, dan bucket kesalahan Amazon S3.

Daftar kesalahan:

Pesan Kesalahan	Deskripsi
`Iceberg.NoSuchTable`	Firehose menulis ke tabel yang tidak ada, atau tabel tidak dalam format V2. Firehose tidak mendukung tabel dalam format V1.
`Iceberg.InvalidTableName`	Nama tabel nol atau kosong dilewatkan, atau tabel tidak dalam format V2. Firehosetidak mendukung tabel dalam format V1.
`S3.AccessDenied`	Pastikan bahwa peran IAM yang dibuat dalam langkah prasyarat memiliki izin yang diperluka n, dan kebijakan kepercayaan.

Pesan Kesalahan	Deskripsi
`Glue.AccessDenied`	Pastikan bahwa peran IAM yang dibuat dalam langkah prasyarat memiliki izin yang diperluka n, dan kebijakan kepercayaan.

Konfigurasikan petunjuk buffer

Firehose menyangga data streaming yang masuk dalam memori ke ukuran tertentu (ukuran Buffering) dan untuk jangka waktu tertentu (interval Buffering) sebelum mengirimkannya ke Apache Iceberg Tables. Anda dapat memilih ukuran buffer 1-128 MiBs dan interval buffer 0-900 detik. Petunjuk buffer yang lebih tinggi menghasilkan jumlah penulisan S3 yang lebih rendah, biaya pemadatan yang lebih sedikit karena file data yang lebih besar, dan runtime kueri yang lebih cepat, tetapi dengan latensi yang lebih tinggi. Nilai petunjuk buffer yang lebih rendah mengirimkan data dengan latensi yang lebih rendah.

Mengonfigurasi pengaturan lanjutan

Anda dapat mengonfigurasi enkripsi sisi server, pencatatan kesalahan, izin, dan tag untuk Apache Iceberg Tables Anda. Untuk informasi selengkapnya, lihat Mengonfigurasi pengaturan lanjutan. Anda harus menambahkan peran IAM yang Anda buat sebagai bagian dari. ??? Firehose akan berperan untuk mengakses AWS Glue tabel dan menulis ke bucket Amazon S3.

Pembuatan aliran Firehose dapat memakan waktu beberapa menit untuk diselesaikan. Setelah berhasil membuat aliran Firehose, Anda dapat mulai memasukkan data ke dalamnya dan dapat melihat data di tabel Apache Iceberg.

Rutekan catatan masuk ke satu tabel Gunung Es

Jika Anda ingin Firehose menyisipkan data ke tabel Iceberg tunggal, cukup konfigurasikan satu database dan tabel dalam konfigurasi aliran Anda seperti yang ditunjukkan pada contoh berikut JSON. Untuk satu tabel, Anda tidak memerlukan ekspresi JQ dan fungsi Lambda untuk memberikan informasi perutean ke Firehose. Jika Anda menyediakan bidang ini bersama dengan JQ atau Lambda, Firehose akan mengambil masukan dari JQ atau Lambda.

```
[
  {
    "DestinationDatabaseName": "UserEvents",
```

```
    "DestinationTableName": "customer_id",
    "UniqueKeys": [
      "COLUMN_PLACEHOLDER"
    ],
    "S3ErrorOutputPrefix": "OPTIONAL_PREFIX_PLACEHOLDER"
  }
]
```

Dalam contoh ini, Firehose merutekan semua catatan input ke `customer_id` tabel dalam `UserEvents` database. Jika Anda ingin melakukan pembaruan atau penghapusan operasi pada satu tabel, maka Anda harus menyediakan operasi untuk setiap catatan masuk ke Firehose menggunakan metode atau metode Lambda. JSONQuery

Rutekan catatan masuk ke tabel Gunung Es yang berbeda

Amazon Data Firehose dapat merutekan catatan yang masuk dalam aliran ke tabel Gunung Es yang berbeda berdasarkan konten catatan. Catatan tidak disimpan secara berurutan saat dikirim dari Amazon Data Firehose. Pertimbangkan catatan input sampel berikut.

```
{
  "deviceId": "Device1234",
  "timestamp": "2024-11-28T11:30:00Z",
  "data": {
    "temperature": 21.5,
    "location": {
      "latitude": 37.3324,
      "longitude": -122.0311
    }
  },
  "powerlevel": 84,
  "status": "online"
}
```

```
{
  "deviceId": "Device4567",
  "timestamp": "2023-11-28T10:40:00Z",
  "data": {
    "pressure": 1012.4,
    "location": {
      "zipcode": 24567
    }
```

```
  },
  "powerlevel": 82,
  "status": "online"
}
```

Dalam contoh ini, **`deviceId`** bidang memiliki dua nilai yang mungkin - `Device1234` dan`Device4567`. Ketika catatan masuk memiliki **`deviceId`** bidang sebagai`Device1234`, kita ingin menulis catatan ke tabel Iceberg bernama`Device1234`, dan ketika catatan masuk memiliki **`deviceId`** bidang sebagai`Device4567`, kita ingin menulis catatan ke tabel bernama. `Device4567`

Perhatikan bahwa catatan dengan `Device1234` dan `Device4567` mungkin memiliki kumpulan bidang berbeda yang dipetakan ke kolom yang berbeda dalam tabel Gunung Es yang sesuai. Catatan yang masuk mungkin memiliki struktur JSON bersarang di mana **`deviceId`**dapat bersarang dalam catatan JSON. Di bagian yang akan datang, kami membahas bagaimana Anda dapat merutekan catatan ke tabel yang berbeda dengan memberikan informasi perutean yang sesuai ke Firehose dalam skenario tersebut.

Berikan informasi perutean ke JSONQuery Firehose dengan ekspresi

Cara paling sederhana dan paling hemat biaya untuk memberikan informasi perutean rekaman ke Firehose adalah dengan memberikan JSONQuery ekspresi. Dengan pendekatan ini, Anda memberikan JSONQuery ekspresi untuk tiga parameter —`Database Name`,`Table Name`, dan (opsional)`Operation`. Firehose menggunakan ekspresi yang Anda berikan untuk mengekstrak informasi dari catatan aliran masuk untuk merutekan catatan.

`Database Name`Parameter menentukan nama database tujuan. `Table Name`Parameter menentukan nama tabel tujuan. `Operation`adalah parameter opsional yang menunjukkan apakah akan memasukkan catatan aliran masuk sebagai catatan baru ke dalam tabel tujuan, atau untuk memodifikasi atau menghapus catatan yang ada di tabel tujuan. Bidang Operasi harus memiliki salah satu nilai berikut —`insert`,`update`, atau`delete`.

Untuk masing-masing dari tiga parameter ini, Anda dapat memberikan nilai statis atau ekspresi dinamis di mana nilai diambil dari catatan masuk. Misalnya, jika Anda ingin mengirimkan semua catatan aliran masuk ke database tunggal bernama`IoTevents`, Nama Database akan memiliki nilai statis. `“IoTevents”` Jika nama tabel tujuan harus diperoleh dari bidang dalam catatan masuk, Nama Tabel adalah ekspresi dinamis yang menentukan bidang dalam catatan masuk dari mana nama tabel tujuan perlu diambil.

Dalam contoh berikut, kita menggunakan nilai statis untuk Nama Database, nilai dinamis untuk Nama Tabel, dan nilai statis untuk operasi. Perhatikan bahwa menentukan Operasi adalah opsional. Jika

tidak ada operasi yang ditentukan, Firehose menyisipkan catatan yang masuk ke dalam tabel tujuan sebagai catatan baru secara default.

```
Database Name : "IoTevents"
Table Name : .deviceId
Operation : "insert"
```

Jika `deviceId` bidang bersarang dalam catatan JSON, kami menentukan Nama Tabel dengan informasi bidang bersarang sebagai. `.event.deviceId`

> **Note**
>
> - Saat menentukan operasi sebagai `update` atau`delete`, Anda harus menentukan kunci unik untuk tabel tujuan saat menyiapkan aliran Firehose, atau mengatur identifier-field-idsdi Iceberg saat menjalankan create table atau mengubah operasi tabel di Iceberg. Jika Anda gagal menentukan ini, Firehose akan melempar kesalahan dan mengirimkan data ke bucket kesalahan S3.
> - `Table Name`Nilai `Database Name` dan harus sama persis dengan database tujuan dan nama tabel Anda. Jika tidak cocok, Firehose akan melempar kesalahan dan mengirimkan data ke bucket kesalahan S3.

Memberikan informasi routing menggunakan fungsi AWS Lambda

Mungkin ada skenario di mana Anda memiliki aturan kompleks yang menentukan cara merutekan catatan masuk ke tabel tujuan. Misalnya, Anda mungkin memiliki aturan yang menentukan jika bidang berisi nilai A, B, atau F, yang harus dirutekan ke tabel tujuan bernama `TableX` atau Anda mungkin ingin menambah catatan aliran masuk dengan menambahkan atribut tambahan. Misalnya, jika catatan berisi bidang `device_id` sebagai 1, Anda mungkin ingin menambahkan bidang lain yang `device_type` sebagai “modem “, dan menulis bidang tambahan ke kolom tabel tujuan. Dalam kasus seperti itu, Anda dapat mengubah aliran sumber dengan menggunakan AWS Lambda fungsi di Firehose dan memberikan informasi perutean sebagai bagian dari output fungsi transformasi Lambda. Untuk memahami cara mengubah aliran sumber menggunakan AWS Lambda fungsi di Firehose, lihat Mengubah data sumber di Amazon Data Firehose.

Saat Anda menggunakan Lambda untuk transformasi aliran sumber di Firehose, output harus berisi,, dan atau `recordId` parameter`result`. `data` `KafkaRecordValue` Parameter `recordId` berisi

catatan aliran input, `result` menunjukkan apakah transformasi berhasil, dan `data` berisi output transformasi yang dikodekan Base64 dari fungsi Lambda Anda. Untuk informasi selengkapnya, lihat ???.

```
{
  "recordId": "49655962066601463032522589543535113056108699331451682818000000",
  "result": "Ok",
  "data": "1IiwiI6ICJmYWxsIiwgImdgU21IiwiI6ICJmYWxsIiwg==tcHV0ZXIgU2NpZW5jZSIsICJzZW1"
}
```

Untuk menentukan informasi perutean ke Firehose tentang cara merutekan rekaman aliran ke tabel tujuan sebagai bagian dari fungsi Lambda Anda, output fungsi Lambda Anda harus berisi bagian tambahan untuk. `metadata` Contoh berikut menunjukkan bagaimana bagian metadata ditambahkan ke output Lambda untuk aliran Firehose yang menggunakan Kinesis Data Streams sebagai sumber data untuk menginstruksikan Firehose bahwa ia harus menyisipkan rekaman sebagai catatan baru ke dalam tabel bernama database. `Device1234` `IoTevents`

```
 {
"recordId": "49655962066601463032522589543535113056108699331451682818000000",
    "result": "Ok",
    "data":
 "1IiwiI6ICJmYWxsIiwgImdgU21IiwiI6ICJmYWxsIiwg==tcHV0ZXIgU2NpZW5jZSIsICJzZW1",

    "metadata":{
"otfMetadata":{
            "destinationTableName":"Device1234",
            "destinationDatabaseName":"IoTevents",
            "operation":"insert"
        }
    }
  }
```

Demikian pula, contoh berikut menunjukkan bagaimana Anda dapat menambahkan bagian metadata ke output Lambda untuk Firehose yang menggunakan Amazon Managed Streaming for Apache Kafka Kafka sebagai sumber data untuk menginstruksikan Firehose bahwa Firehose harus menyisipkan catatan sebagai catatan baru ke dalam tabel yang disebutkan dalam database. `Device1234` `IoTevents`

```
{
"recordId": "49655962066601463032522589543535113056108699331451682818000000",
```

```
    "result": "Ok",
    "kafkaRecordValue":
 "1IiwiI6ICJmYWxsIiwgImdgU21IiwiI6ICJmYWxsIiwg==tcHV0ZXIgU2NpZW5jZSIsICJzZW1",

    "metadata":{
"otfMetadata":{
            "destinationTableName":"Device1234",
            "destinationDatabaseName":"IoTevents",
            "operation":"insert"
        }
    }
  }
```

Untuk contoh ini,

- `destinationDatabaseName`mengacu pada nama database target dan merupakan bidang wajib.
- `destinationTableName`mengacu pada nama tabel target dan merupakan bidang wajib.
- `operation`adalah bidang opsional dengan nilai yang mungkin sebagai`insert`,`update`, dan`delete`. Jika Anda tidak menentukan nilai apa pun, operasi defaultnya adalah`insert`.

> **Note**
>
> - Saat menentukan operasi sebagai `update` ataudelete, Anda harus menentukan kunci unik untuk tabel tujuan saat menyiapkan aliran Firehose, atau mengatur identifier-field-idsdi Iceberg saat menjalankan create table atau mengubah operasi tabel di Iceberg. Jika Anda gagal menentukan ini, Firehose akan melempar kesalahan dan mengirimkan data ke bucket kesalahan S3.
> - `Table Name`Nilai `Database Name` dan harus sama persis dengan database tujuan dan nama tabel Anda. Jika tidak cocok, Firehose akan melempar kesalahan dan mengirimkan data ke bucket kesalahan S3.
> - Jika aliran Firehose Anda memiliki fungsi transformasi Lambda dan ekspresi JSONQuery , Firehose terlebih dahulu memeriksa bidang metadata dalam keluaran Lambda untuk menentukan cara merutekan rekaman ke tabel tujuan yang sesuai, lalu lihat output ekspresi Anda untuk bidang yang hilang. JSONQuery
>
>   Jika Lambda atau JSONQuery ekspresi tidak memberikan informasi perutean yang diperlukan, Firehose menganggap ini sebagai skenario tabel tunggal dan mencari informasi tabel tunggal dalam konfigurasi kunci unik.

Untuk informasi selengkapnya, lihat Rute catatan masuk ke tabel Gunung Es tunggal. Jika Firehose gagal menentukan informasi perutean dan mencocokkan catatan dengan tabel tujuan tertentu, Firehose akan mengirimkan data ke bucket kesalahan S3 yang Anda tentukan.

Contoh fungsi Lambda

Fungsi Lambda ini adalah contoh kode Python yang mengurai catatan aliran masuk dan menambahkan bidang yang diperlukan untuk menentukan bagaimana data harus ditulis ke tabel tertentu. Anda dapat menggunakan kode contoh ini untuk menambahkan bagian metadata untuk informasi perutean.

```
import json
import base64


def lambda_handler(firehose_records_input, context):
    print("Received records for processing from DeliveryStream: " +
 firehose_records_input['deliveryStreamArn'])

    firehose_records_output = {}
    firehose_records_output['records'] = []


    for firehose_record_input in firehose_records_input['records']:

        # Get payload from Lambda input, it could be different with different sources
        if 'kafkaRecordValue' in firehose_record_input:
            payload_bytes =
 base64.b64decode(firehose_record_input['kafkaRecordValue']).decode('utf-8')
        else
            payload_bytes =
 base64.b64decode(firehose_record_input['data']).decode('utf-8')

        # perform data processing on customer payload bytes here

        # Create output with proper record ID, output data (may be different with
 different sources), result, and metadata
        firehose_record_output = {}
```

```
        if 'kafkaRecordValue' in firehose_record_input:
            firehose_record_output['kafkaRecordValue'] =
base64.b64encode(payload_bytes.encode('utf-8'))
        else
            firehose_record_output['data'] =
base64.b64encode(payload_bytes.encode('utf-8'))

        firehose_record_output['recordId'] = firehose_record_input['recordId']
        firehose_record_output['result'] =  'Ok'
        firehose_record_output['metadata'] = {
            'otfMetadata': {
                'destinationDatabaseName': 'your_destination_database',
                'destinationTableName': 'your_destination_table',
                'operation': 'insert'
                }
            }
        firehose_records_output['records'].append(firehose_record_output)
    return firehose_records_output
```

Monitor metrik

Untuk pengiriman data ke Apache Iceberg Tables, Firehose memancarkan metrik berikut CloudWatch pada tingkat aliran.

Metrik	Deskripsi
`DeliveryToIceberg.Bytes`	Jumlah byte yang dikirim ke Apache Iceberg Tables selama periode waktu yang ditentukan. Unit: Byte
`DeliveryToIceberg.IncomingRowCount`	Jumlah catatan yang Firehose coba kirimkan ke Apache Iceberg Tables. Unit: Hitungan
`DeliveryToIceberg.SuccessfulRowCount`	Jumlah baris yang berhasil dikirim ke Apache Iceberg Tables. Unit: Hitungan

Metrik	Deskripsi
`DeliveryToIceberg.FailedRowCount`	Jumlah baris gagal dikirim ke bucket cadangan S3. Unit: Hitungan
`DeliveryToIceberg.DataFreshness`	Usia (dari masuk ke Firehose hingga sekarang) dari rekor paling awal di Firehose. Setiap catatan lebih awal dari usia ini telah dikirim ke Apache Iceberg Tables. Unit: detik
`DeliveryToIceberg.Success`	Jumlah komitmen yang berhasil ke Apache Iceberg Tables.
`JQProcessing.Duration`	Jumlah waktu yang dibutuhkan untuk menjalankan ekspresi JQ. Unit: Milidetik

Memahami tipe data yang didukung

Firehose mendukung semua tipe data primitif dan kompleks yang didukung Apache Iceberg. Untuk informasi selengkapnya, lihat Skema dan Jenis Data. Saat mengirim data biner sebagai string, Anda harus menggunakan jenis pengkodean yang didukung Firehose - Base64 Dasar, MIME Base64, URL dan nama file aman Base64, dan Hex. Untuk tipe data Timestamp, Anda harus selalu mengirim dalam mikrodetik.

Contoh tipe data

Bagian berikut menunjukkan contoh tipe data yang berbeda.

MapType

```
{
    "destination_column_0":
 {"WP5o0JOkuIQcDPcsvpJJygF1xzaOSq0wUlgTwuIeCEzgVneGxA":"PO3ReF3auyDqbfonx9Cd8NTmcQnqnw7JuZOCWwI
    "destination_column_1": "{\"{\\\"destination_nested_column_0\\\": \
\\"18:56:14.974\\\", \\\"destination_nested_column_1\\\": 241.86246}\":
\"M07kAvYdHvBh61F7RzfxtEd39YQI33LnM2NbGS67DOFFsRUyUUujKT5VnK7Wtfz1mHNeIix6FAY9cYpwTdedgr9XnFwG0
```

```
\",\"{\\\"destination_nested_column_0\\\": \\\"18:56:14.974\
\\", \\\"destination_nested_column_1\\\": 562.56384}\":
\"9GlxhDCt95LxBo51HybBZihqOqf6EU8jrDu7NMpxtGB2dY6q6kXpvxIrFuMdqHCJKIZIcDikwggLniUm8kgE4d
\",\"{\\\"destination_nested_column_0\\\": \\\"18:56:14.974\
\\", \\\"destination_nested_column_1\\\": 496.03268}\":
\"keTJZYLNvLRB50DMKzEI6M0AM4mueyNnA1m2YVnYdDwyxUpPqkb72Q6LiX0B9s8gCjZ6trW6C1PFk9KNBIpxYsj5Tc5Xs
\",\"{\\\"destination_nested_column_0\\\": \\\"18:56:14.974\\\", \\
\"destination_nested_column_1\\\": 559.0878}\":
\"mG0ZET84BUF28E312UCIWgmypyQFSUODH9NAMAnF3LJEutbooZWcBt97PP5AHaopNvC8pQZ4mGXB9hmVmjUNmuj5Qanyx
\",\"{\\\"destination_nested_column_0\\\": \\\"18:56:14.974\
\\", \\\"destination_nested_column_1\\\": 106.845245}\":
\"aidoVYrzu8gcLRkVVUyTKCN9gqTUFYi8uJQsrXEFEYl1f9ool7JhAtg9QKG5BBu67Ngb95ENsNKQyCHNImSu5x4hMnmHU
\"}"
}
```


DecimalType


```
{
    "destination_column_0": 9455262425851.1342772,
    "destination_column_1": "9455262425851.1342772",
    "destination_column_2": 9455262425852
}
```


BinaryType (base64-default, base64-mime, base64-url-safe, hex)


```
{
    "destination_column_0": "AsYhnHD\/Ra54hITl1daNV9glOjtWPEfopH
+PjgUKHYB6K7UcYi4K19b80wD4J\/93x5tyh+0y
+k5cMljVRlmfIkIuLxl9ERBiPPLhf4+yoJ2k70VavPnYWmNLs1hLDHlfeEMIfVhrqOGzJMoA
+CBAWXfIuiG420JSQP5iAx5xFG\/
mOfkM5zYothje8OGXltdthcCL6WYBiP0SlwXcE0uMeRfwclAc9fTOBz6RzdJlHhUDjoAXg
+4cvly27F82XpuGMNwpUj98AOrgbh2MoU9yvsM9ZrjD0eGVgOZP8Ky7Za4oE\/oK8j
+qABF6XV712iA6pVtTNJFvX6Ey3ssNYvno+LYF5ZsySs2rB5AbVM73RfOPqdS\/c\/
r3MEqoEqt+nPx6eGam4WSA+0swztt7aLdrlX6yK7xJeIJ0rTlIDBo0ZUaw011ykY
\/8Bvy+4byoPlmr4Z5yhN1z3ZTOkx7eDR6xMv+vDVsDBtItVazDwHgDy41r
\/hQNeNedPKrozc8TY9k7wZre\/6V2lCa3BmT8Uu9b9ydjR9z+fCSdG
+VRv35nz5kdqdKy8YIrynYs4eOcjh8jH3UwVYrYQcnWkBAfF7Xk9CoPVnL3ciHZtyiZOaTGIj9rO0xX\/
W5dGe9\/4YChs6LbD584kxLTxvHgSl4vadaTGNKci3SvNmZNsz8ducxtNXF\/Tv2DUub465hzgpaLPur3+MB
+kfdN2YXUfqb
+xJAgxThWfUe151nrH0EPow9lgSlp21rUBGznJAvPRl1ExGIAuc7JYAoUrJUkx5Hfl6PekPDhqt7+yJwCB8qxhTTryxo
+bjtai4ndRCGcuCaxT8KkOcXsS37urd3YGSDMinZdMNVc646s25415qK6nBRlqqAY8+EYmcUIVB9XcNdke4zoUfhVQoruwi
\/kFafoulo5DEoMOyaH1N2HCSxG5tZXNQocSZPaY8efZYMCpmDXsPAzkmgSkYRDSu\/r3wUqROa2tGK5\/
pQY24v+Jq0U\/jQ99GShlU283nZ85ot2ocbtMAgD\/WsrSEh6lNt9RaI3HfA7\/HcH\/
```

```
fgr9jsTtxDgZhabTBwwDwX0zjWGx1bCuTLKBN7byxg9ZvAVgqwPS4HERLer5T5UkKf74zn9Eq3HYH1Q5JpyDUx
+im7mte1sprf1+A24kksVU\/MD9aP9N8\/QDsQ13gkhOn5KwFMz3BC2Vw5gL
+gGNHFKDRL6wGIfhuYcx9LucolZ1yNy9Gbb3ioWSSufyFpyXqtndDLPI5QS1SJpJm2KDyqcH1SmRLIhd9MNRUC73EAEm
+NO5wxPzBRSjhCHZpf8SrYITWJl7K3XzGOfPFh2NgES3jMP9cvSXO6yyICcep2HBYGbFflni89+Rw==",
    "destination_column_1": "AsYhnHD\/Ra54hITl1daNV9glOjtWPEfopH
+PjgUKHYB6K7UcYi4K19b80wD4J\/93x5tyh+0y+k5c\r
\nMljVRlmfIkIuLxl9ERBiPPLhf4+yoJ2k70VavPnYWmNLs1hLDHlfeEMIfVhrqOGzJMoA+CBAWXfI\r
\nuiG420JSQP5iAx5xFG\/mOfkM5zYothje8OGXltdthcCL6WYBiP0SlwXcE0uMeRfwclAc9fTOBz6R\r
\nzdJlHhUDjoAXg+4cvly27F82XpuGMNwpUj98AOrgbh2MoU9yvsM9ZrjD0eGVgOZP8Ky7Za4oE\/oK\r
\n8j+qABF6XV712iA6pVtTNJFvX6Ey3ssNYvno+LYF5ZsySs2rB5AbVM73RfOPqdS\/c\/r3MEqoEqt+
\r\nnPx6eGam4WSA+0swztt7aLdrlX6yK7xJeIJ0rTlIDBo0ZUaw011ykY\/8Bvy+4byoPlmr4Z5yhN1z
\r\n3ZTOkx7eDR6xMv+vDVsDBtItVazDwHgDy41r\/hQNeNedPKrozc8TY9k7wZre\/6V2lCa3BmT8Uu9b
\r\n9ydjR9z+fCSdG+VRv35nz5kdqdKy8YIrynYs4eOcjh8jH3UwVYrYQcnWkBAfF7Xk9CoPVnL3ciHZ
\r\ntyiZOaTGIj9rO0xX\/W5dGe9\/4YChs6LbD584kxLTxvHgSl4vadaTGNKci3SvNmZNsz8ducxtNXF
\/\r\nTv2DUub465hzgpaLPur3+MB+kfdN2YXUfqb+xJAgxThWfUe151nrH0EPow9lgSlp21rUBGznJAvP
\r\nRl1ExGIAuc7JYAoUrJUkx5Hfl6PekPDhqt7+yJwCB8qxhTTryxo+bjtai4ndRCGcuCaxT8KkOcXs\r
\nS37urd3YGSDMinZdMNVc646s25415qK6nBRlqqAY8+EYmcUIVB9XcNdke4zoUfhVQoruwidzDU\/k\r
\nFafoulo5DEoMOyaH1N2HCSxG5tZXNQocSZPaY8efZYMCpmDXsPAzkmgSkYRDSu\/r3wUqROa2tGK5\r
\n\/pQY24v+Jq0U\/jQ99GShlU283nZ85ot2ocbtMAgD\/WsrSEh6lNt9RaI3HfA7\/HcH\/fgr9jsTtxDg
\r\nZhabTBwwDwX0zjWGx1bCuTLKBN7byxg9ZvAVgqwPS4HERLer5T5UkKf74zn9Eq3HYH1Q5JpyDUx+\r
\nim7mte1sprf1+A24kksVU\/MD9aP9N8\/QDsQ13gkhOn5KwFMz3BC2Vw5gL+gGNHFKDRL6wGIfhuYc
\r\nx9LucolZ1yNy9Gbb3ioWSSufyFpyXqtndDLPI5QS1SJpJm2KDyqcH1SmRLIhd9MNRUC73EAEm+NO\r
\n5wxPzBRSjhCHZpf8SrYITWJl7K3XzGOfPFh2NgES3jMP9cvSXO6yyICcep2HBYGbFflni89+Rw==",
    "destination_column_2": "AsYhnHD_Ra54hITl1daNV9glOjtWPEfopH-
PjgUKHYB6K7UcYi4K19b80wD4J_93x5tyh-0y-k5cMljVRlmfIkIuLxl9ERBiPPLhf4-
yoJ2k70VavPnYWmNLs1hLDHlfeEMIfVhrqOGzJMoA-
CBAWXfIuiG420JSQP5iAx5xFG_mOfkM5zYothje8OGXltdthcCL6WYBiP0SlwXcE0uMeRfwclAc9fTOBz6RzdJlHhUDjoAX
qABF6XV712iA6pVtTNJFvX6Ey3ssNYvno-LYF5ZsySs2rB5AbVM73RfOPqdS_c_r3MEqoEqt-
nPx6eGam4WSA-0swztt7aLdrlX6yK7xJeIJ0rTlIDBo0ZUaw011ykY_8Bvy-4byoPlmr4Z5yhN1z3ZTOkx7eDR6xMv-
vDVsDBtItVazDwHgDy41r_hQNeNedPKrozc8TY9k7wZre_6V2lCa3BmT8Uu9b9ydjR9z-fCSdG-
VRv35nz5kdqdKy8YIrynYs4eOcjh8jH3UwVYrYQcnWkBAfF7Xk9CoPVnL3ciHZtyiZOaTGIj9rO0xX_W5dGe9_4YChs6LbD
MB-kfdN2YXUfqb-
xJAgxThWfUe151nrH0EPow9lgSlp21rUBGznJAvPRl1ExGIAuc7JYAoUrJUkx5Hfl6PekPDhqt7-
yJwCB8qxhTTryxo-bjtai4ndRCGcuCaxT8KkOcXsS37urd3YGSDMinZdMNVc646s25415qK6nBRlqqAY8-
EYmcUIVB9XcNdke4zoUfhVQoruwidzDU_kFafoulo5DEoMOyaH1N2HCSxG5tZXNQocSZPaY8efZYMCpmDXsPAzkmgSkYRDS
Jq0U_jQ99GShlU283nZ85ot2ocbtMAgD_WsrSEh6lNt9RaI3HfA7_HcH_fgr9jsTtxDgZhabTBwwDwX0zjWGx1bCuTLKBN7
im7mte1sprf1-A24kksVU_MD9aP9N8_QDsQ13gkhOn5KwFMz3BC2Vw5gL-
gGNHFKDRL6wGIfhuYcx9LucolZ1yNy9Gbb3ioWSSufyFpyXqtndDLPI5QS1SJpJm2KDyqcH1SmRLIhd9MNRUC73EAEm-
NO5wxPzBRSjhCHZpf8SrYITWJl7K3XzGOfPFh2NgES3jMP9cvSXO6yyICcep2HBYGbFflni89-Rw==",
    "destination_column_3":
 "02c6219c70ff45ae788484e5d5d68d57d8253a3b563c47e8a47f8f8e050a1d807a2bb51c622e0ad7d6fcd300f827f
}
```

TimeType (Epoch dalam Mikrodetik, Objek LocalTime Java)

```
{
    "destination_column_0": 68175096000,
    "destination_column_1": "18:56:15.096"
}
```

TimestampType.withZone (Epoch dalam Mikrodetik, Objek OffsetDateTime Java, Objek Java) LocalDateTime

```
{
    "destination_column_0": 1725476175099000,
    "destination_column_1": "2024-09-04T18:56:15.099Z",
    "destination_column_2": "2024-09-04T18:56:15.099"
}
```

DoubleType

```
{
    "destination_column_0": 9.18477568715142,
    "destination_column_1": "9.18477568715142"
}
```

BooleanType

```
{
    "destination_column_0": true,
    "destination_column_1": "false",
    "destination_column_2": 1,
    "destination_column_3": 0
}
```

FloatType

```
{
    "destination_column_0": 0.6242226,
    "destination_column_1": "0.6242226"
}
```

IntegerType

```
{
    "destination_column_0": 7,
    "destination_column_1": "7"
}
```

TimestampType.withoutZone (Epoch dalam Microseconds, LocalDateTime Java Object, Java Object, OffsetDateTime Java Object) ZonedDateTime

```
{
    "destination_column_0": 1725476175114000,
    "destination_column_1": "2024-09-04T18:56:15.114",
    "destination_column_2": "2024-09-04T18:56:15.114Z",
    "destination_column_3": "2024-09-04T18:56:15.114-07:00"
}
```

DateType

```
{
    "destination_column_0": 19970,
    "destination_column_1": "2024-09-04"
}
```

LongType

```
{
    "destination_column_0": 8,
    "destination_column_1": "8"
}
```

UUIDType (Objek UUID Java)

```
{
    "destination_column_0": "21c5521c-a6d4-48d4-b2c8-7f6d842f72c3"
}
```

ListType

```
{
    "destination_column_0":
 ["s1FSrgb0lGDxfn2iYT0EtlP47aHSjwmLZgrdr1JqRs0dmbeCcQoaLr4Xhi2KIVvmus9ppFdpWIcOHnJ0omhAPhXH0yns
```

```
    "destination_column_1": "[{\"destination_nested_column_0\":\"bb00f8e6-
db82-4241-a5c5-0d9c0d2f71a4\",\"destination_nested_column_1\":907.35345},
{\"destination_nested_column_0\":\"2c77b702-d405-4fe1-beee-fb541d7ab833\",
\"destination_nested_column_1\":544.0026},{\"destination_nested_column_0\":
\"68389200-d6b1-413d-bcd9-fdb931708395\",\"destination_nested_column_1\":153.683},
{\"destination_nested_column_0\":\"bc31cbaa-39cd-4e2f-b357-9ea9ce75532b\",
\"destination_nested_column_1\":977.5165},{\"destination_nested_column_0\":
\"b7d627f9-0d5b-41b7-903a-525488259fba\",\"destination_nested_column_1\":434.17215},
{\"destination_nested_column_0\":\"06b6ec1e-1952-4582-b285-46aaf40064b8\",
\"destination_nested_column_1\":580.33124},{\"destination_nested_column_0\":
\"f04b3bbf-61ad-4c5c-8740-6f666f57c431\",\"destination_nested_column_1\":550.75793}]"
}
```

Sumber daya

Gunakan sumber daya berikut untuk mempelajari lebih lanjut:

- Streaming data real-time ke tabel Apache Iceberg di Amazon S3 menggunakan Amazon Data Firehose
- Sederhanakan analisis AWS WAF log dengan Apache Iceberg dan Amazon Data Firehose
- Membangun data lake untuk streaming data dengan Amazon S3 Tables dan Amazon Data Firehose

Menandai aliran Firehose

Anda dapat menetapkan metadata Anda sendiri ke aliran Firehose yang Anda buat di Amazon Data Firehose dalam bentuk tag. Tanda adalah pasangan nilai-kunci yang Anda tetapkan untuk sebuah aliran. Menggunakan tag adalah cara sederhana namun ampuh untuk mengelola AWS sumber daya dan mengatur data, termasuk data penagihan.

Anda dapat menentukan tag saat memanggil CreateDeliveryStreamuntuk membuat aliran Firehose baru. Untuk aliran Firehose yang ada, Anda dapat menambahkan, membuat daftar, dan menghapus tag menggunakan tiga operasi berikut:

- TagDeliveryStream
- ListTagsForDeliveryStream
- UntagDeliveryStream

Memahami dasar-dasar tag

Anda dapat menggunakan operasi Amazon Data Firehose API untuk menyelesaikan tugas-tugas berikut:

- Tambahkan tag ke aliran Firehose.
- Buat daftar tag untuk aliran Firehose Anda.
- Hapus tag dari aliran Firehose.

Anda dapat menggunakan tag untuk mengkategorikan aliran Firehose Anda. Misalnya, Anda dapat mengkategorikan aliran Firehose berdasarkan tujuan, pemilik, atau lingkungan. Karena Anda menentukan kunci dan nilai untuk setiap tanda, Anda dapat membuat serangkaian kategori khusus untuk memenuhi kebutuhan spesifik Anda. Misalnya, Anda dapat menentukan satu set tag yang membantu Anda melacak aliran Firehose menurut pemilik dan aplikasi terkait.

Berikut ini adalah beberapa contoh tanda:

- `Project: Project name`
- `Owner: Name`
- `Purpose: Load testing`

- `Application: Application name`
- `Environment: Production`

Jika Anda menentukan tag dalam `CreateDeliveryStream` tindakan, Amazon Data Firehose akan melakukan otorisasi tambahan pada `firehose:TagDeliveryStream` tindakan tersebut untuk memverifikasi apakah pengguna memiliki izin untuk membuat tag. Jika Anda tidak memberikan izin ini, permintaan untuk membuat aliran Firehose baru dengan tag sumber daya IAM akan gagal dengan hal seperti berikut. `AccessDeniedException`

```
AccessDeniedException
User: arn:aws:sts::x:assumed-role/x/x is not authorized to perform:
 firehose:TagDeliveryStream on resource: arn:aws:firehose:us-east-1:x:deliverystream/x
 with an explicit deny in an identity-based policy.
```

Contoh berikut menunjukkan kebijakan yang memungkinkan pengguna membuat aliran Firehose dan menerapkan tag.

Lacak biaya dengan penandaan

Anda dapat menggunakan tag untuk mengkategorikan dan melacak biaya Anda AWS . Saat Anda menerapkan tag ke AWS sumber daya Anda, termasuk aliran Firehose, laporan alokasi AWS biaya Anda mencakup penggunaan dan biaya yang dikumpulkan berdasarkan tag. Anda dapat mengatur biaya di berbagai layanan dengan menerapkan tanda yang mewakili kategori bisnis (seperti pusat biaya, nama aplikasi, atau pemilik). Untuk informasi selengkapnya, lihat Menggunakan Tanda Alokasi Biaya untuk Laporan Penagihan Khusus dalam Panduan Pengguna AWS Billing .

Ketahui batasan tag

Pembatasan berikut berlaku untuk tag di Amazon Data Firehose.

Batasan dasar

- Jumlah maksimum tanda per sumber daya (aliran) adalah 50.
- Kunci dan nilai tag peka terhadap huruf besar dan kecil.
- Anda tidak dapat mengubah atau mengedit tanda untuk aliran yang dihapus.

Batasan kunci tanda

- Setiap kunci tanda harus unik. Jika Anda menambahkan tanda dengan kunci yang sudah digunakan, tanda baru akan menimpa pasangan nilai-kunci yang sudah ada.
- Anda tidak dapat memulai kunci tanda dengan `aws:` karena prefiks ini disimpan untuk digunakan oleh AWS. AWS membuat tanda yang dimulai dengan prefiks ini atas nama Anda, tetapi Anda tidak dapat mengedit atau menghapusnya.
- Kunci tanda harus memiliki panjang antara 1 dan 128 karakter Unicode.
- Kunci tanda harus terdiri dari karakter berikut: huruf Unicode, digit, spasi, dan karakter khusus berikut: `_ . / = + - @`.

Batasan nilai tanda

- Panjang nilai tanda harus antara 0 dan 255 karakter Unicode.
- Nilai tanda dapat kosong. Jika tidak, nilai tanda harus terdiri dari karakter berikut: huruf Unicode, digit, spasi, dan salah satu karakter khusus berikut: `_ . / = + - @`.

Keamanan di Amazon Data Firehose

Keamanan cloud di AWS adalah prioritas tertinggi. Sebagai AWS pelanggan, Anda akan mendapat manfaat dari pusat data dan arsitektur jaringan yang dibangun untuk memenuhi persyaratan organisasi yang paling sensitif terhadap keamanan.

Keamanan adalah tanggung jawab bersama antara Anda AWS dan Anda. Model tanggung jawab bersama menggambarkan hal ini sebagai keamanan dari cloud dan keamanan di cloud:

- Keamanan cloud — AWS bertanggung jawab untuk melindungi infrastruktur yang menjalankan AWS layanan di AWS Cloud. AWS juga memberi Anda 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 Firehose Data, lihat AWS Layanan dalam Lingkup berdasarkan Program Kepatuhan.
- Keamanan di cloud — Tanggung jawab Anda ditentukan oleh AWS layanan yang Anda gunakan. Anda juga bertanggung jawab atas faktor-faktor lain termasuk sensitivitas data, kebutuhan organisasi, serta undang-undang dan peraturan yang berlaku.

Dokumentasi ini membantu Anda memahami cara menerapkan model tanggung jawab bersama saat menggunakan Data Firehose. Topik berikut menunjukkan cara mengonfigurasi Firehose Data untuk memenuhi tujuan keamanan dan kepatuhan Anda. Anda juga akan mempelajari cara menggunakan AWS layanan lain yang dapat membantu Anda memantau dan mengamankan sumber daya Firehose Data Anda.

Topik

Perlindungan data di Amazon Data Firehose

Amazon Data Firehose mengenkripsi semua data dalam perjalanan menggunakan protokol TLS. Selanjutnya, untuk data yang disimpan dalam penyimpanan sementara selama pemrosesan, Amazon Data Firehose mengenkripsi data AWS Key Management Servicemenggunakan dan memverifikasi integritas data menggunakan verifikasi checksum.

Jika memiliki data sensitif, Anda dapat mengaktifkan enkripsi data sisi server saat menggunakan Amazon Data Firehose. Cara Anda melakukannya tergantung pada sumber data Anda.

> **Note**
>
> Jika Anda memerlukan modul kriptografi tervalidasi FIPS 140-2 saat mengakses AWS melalui antarmuka baris perintah atau API, gunakan titik akhir FIPS. Lihat informasi selengkapnya tentang titik akhir FIPS yang tersedia di Standar Pemrosesan Informasi Federal (FIPS) 140-2.

Server-side enkripsi dengan Kinesis Data Streams

Saat Anda mengirim data dari produsen data ke aliran data, Kinesis Data Streams mengenkripsi data Anda AWS Key Management Service menggunakan kunci () sebelum AWS KMS menyimpan data saat istirahat. Saat Firehose stream Anda membaca data dari aliran data Anda, Kinesis Data Streams pertama-tama mendekripsi data dan kemudian mengirimkannya ke Amazon Data Firehose. Amazon Data Firehose menyangga data dalam memori berdasarkan petunjuk buffering yang Anda tentukan. Alat ini kemudian mengirimkannya ke tujuan Anda tanpa menyimpan data at rest yang tidak terenkripsi.

Untuk informasi tentang cara mengaktifkan enkripsi sisi server untuk Kinesis Data Streams, lihat Server-Side Menggunakan Enkripsi di Panduan Pengembang Amazon Kinesis Data Streams.

Server-side enkripsi dengan Direct PUT atau sumber data lainnya

Jika Anda mengirim data ke aliran Firehose menggunakan PutRecordatau PutRecordBatch, atau jika Anda mengirim data menggunakan, CloudWatch Log AWS IoT Amazon, atau CloudWatch Acara, Anda dapat mengaktifkan enkripsi sisi server dengan menggunakan operasi. StartDeliveryStreamEncryption

Untuk menghentikan enkripsi sisi server, gunakan operasi. StopDeliveryStreamEncryption

Anda juga dapat mengaktifkan SSE saat membuat aliran Firehose. Untuk melakukan itu, tentukan DeliveryStreamEncryptionConfigurationInputkapan Anda memanggil CreateDeliveryStream.

Agar berhasil digunakanCUSTOMER_MANAGED_CMK, kebijakan IAM pemanggil dan kebijakan kunci KMS harus mengizinkan kms:GenerateDataKey dan beroperasi. kms:Decrypt Firehose memvalidasi izin ini saat Anda menelepon PutRecord atau dengan enkripsi. PutRecordBatch CUSTOMER_MANAGED_CMK Selain itu, kms:CreateGrant izin diperlukan saat menelepon CreateDeliveryStream atau StartDeliveryStreamEncryption dengan CUSTOMER_MANAGED_CMK enkripsi.

Jika CMK bertipeCUSTOMER_MANAGED_CMK, jika layanan Amazon Data Firehose tidak dapat mendekripsi catatan karena, KMSNotFoundException a, KMSInvalidStateException KMSDisabledException a, atau KMSAccessDeniedException a, layanan menunggu hingga 24 jam (periode penyimpanan) untuk menyelesaikan masalah. Jika masalah berlanjut melampaui periode retensi, layanan akan melompati catatan yang telah melewati periode retensi dan tidak dapat didekripsi, kemudian membuang data tersebut. Amazon Data Firehose menyediakan empat CloudWatch metrik berikut yang dapat Anda gunakan untuk melacak empat pengecualian: AWS KMS

- KMSKeyAccessDenied
- KMSKeyDisabled
- KMSKeyInvalidState
- KMSKeyNotFound

Untuk informasi selengkapnya tentang keempat metrik ini, lihat the section called “Monitoring dengan CloudWatch Metrik”.

> **Important**
>
> Untuk mengenkripsi aliran Firehose Anda, gunakan CMK simetris. Amazon Data Firehose tidak mendukung CMK asimetris. Untuk informasi tentang CMK simetris dan asimetris, lihat Tentang CMK Simetris dan Asimetris di panduan pengembang. AWS Key Management Service

> **Note**
>
> Bila Anda menggunakan kunci terkelola pelanggan (CUSTOMER_MANAGED_CMK) untuk mengaktifkan enkripsi sisi server (SSE) untuk aliran Firehose Anda, layanan Firehose akan menetapkan konteks enkripsi setiap kali menggunakan kunci Anda. Karena konteks enkripsi ini mewakili kejadian di mana kunci yang dimiliki oleh AWS akun Anda digunakan, itu dicatat sebagai bagian dari log AWS CloudTrail peristiwa untuk AWS akun Anda. Konteks enkripsi ini adalah sistem yang dihasilkan oleh layanan Firehose. Aplikasi Anda tidak boleh membuat asumsi apa pun tentang format atau konten konteks enkripsi yang ditetapkan oleh layanan Firehose.

Mengontrol akses dengan Amazon Data Firehose

Bagian berikut mencakup cara mengontrol akses ke dan dari sumber daya Amazon Data Firehose Anda. Informasi yang mereka cakup mencakup cara memberikan akses aplikasi Anda sehingga dapat mengirim data ke aliran Firehose Anda. Mereka juga menjelaskan bagaimana Anda dapat memberikan Amazon Data Firehose akses ke bucket Amazon Simple Storage Service (Amazon S3), cluster Amazon Redshift, atau OpenSearch kluster Layanan Amazon, serta izin akses yang Anda perlukan jika Anda menggunakan Datadog, Dynatrace,, MongoDB, New Relic, Splunk, atau Sumo LogicMonitor Logic sebagai tujuan Anda. Terakhir, Anda akan menemukan panduan topik ini tentang cara mengonfigurasi Amazon Data Firehose sehingga dapat mengirimkan data ke tujuan yang termasuk dalam akun yang berbeda AWS . Teknologi untuk mengelola semua bentuk akses ini adalah AWS Identity and Access Management (IAM). Untuk informasi selengkapnya tentang IAM, lihat Apa itu IAM?.

Daftar Isi

- Berikan akses Firehose ke tujuan Layanan publik OpenSearch
- Berikan akses Firehose ke tujuan OpenSearch Layanan di VPC
- Berikan akses Firehose ke tujuan publik Tanpa Server OpenSearch
- Berikan akses Firehose ke tujuan OpenSearch Tanpa Server di VPC
- Berikan akses Firehose ke tujuan Splunk
- Mengakses Splunk di VPC
- Menelan log aliran VPC ke Splunk menggunakan Amazon Data Firehose
- Mengakses Snowflake atau titik akhir HTTP
- Berikan akses Firehose ke tujuan Snowflake
- Mengakses Snowflake di VPC
- Berikan akses Firehose ke tujuan titik akhir HTTP
- Cross-account pengiriman dari Amazon MSK
- Cross-account pengiriman ke tujuan Amazon S3
- Cross-account pengiriman ke tujuan OpenSearch Layanan
- Menggunakan tanda untuk mengontrol akses

Berikan akses ke sumber daya Firehose Anda

Untuk memberikan akses aplikasi Anda ke aliran Firehose Anda, gunakan kebijakan yang mirip dengan contoh ini. Anda dapat menyesuaikan tiap-tiap operasi API yang Anda beri akses dengan memodifikasi bagian `Action`, atau memberikan akses ke semua operasi dengan `"firehose:*"`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "firehose:DeleteDeliveryStream",
                "firehose:PutRecord",
                "firehose:PutRecordBatch",
                "firehose:UpdateDestination"
            ],
```

```
            "Resource": [
                "arn:aws:firehose:us-east-1:123456789012:deliverystream/delivery-
stream-name"
            ]
        }
    ]
}
```

Berikan akses Firehose ke kluster MSK Amazon pribadi Anda

Jika sumber aliran Firehose Anda adalah kluster MSK Amazon pribadi, gunakan kebijakan yang mirip dengan contoh ini.

Anda harus menambahkan kebijakan seperti ini ke kebijakan berbasis sumber daya kluster untuk memberikan izin kepada prinsipal layanan Firehose untuk menjalankan operasi Amazon MSK API. `CreateVpcConnection`

Izinkan Firehose untuk mengambil peran IAM

Bagian ini menjelaskan izin dan kebijakan yang memberikan Amazon Data Firehose akses untuk mencerna, memproses, dan mengirimkan data dari sumber ke tujuan.

> **Note**
>
> Jika Anda menggunakan konsol untuk membuat aliran Firehose dan memilih opsi untuk membuat peran baru, AWS lampirkan kebijakan kepercayaan yang diperlukan ke peran tersebut. Jika Anda ingin Amazon Data Firehose menggunakan peran IAM yang ada atau jika Anda membuat peran sendiri, lampirkan kebijakan kepercayaan berikut ke peran tersebut sehingga Amazon Data Firehose dapat menerimanya. Edit kebijakan untuk mengganti *account-id* dengan ID AWS akun Anda. Untuk informasi tentang cara mengubah hubungan kepercayaan suatu peran, lihat Mengubah Peran.

Amazon Data Firehose menggunakan peran IAM untuk semua izin yang dibutuhkan aliran Firehose untuk memproses dan mengirimkan data. Pastikan bahwa kebijakan kepercayaan berikut dilampirkan pada peran tersebut sehingga Amazon Data Firehose dapat menerapkannya.

Jika Anda memilih Amazon MSK sebagai sumber untuk aliran Firehose, Anda harus menentukan peran IAM lain yang memberikan izin Amazon Data Firehose untuk menyerap data sumber dari

klaster MSK Amazon yang ditentukan. Pastikan bahwa kebijakan kepercayaan berikut dilampirkan pada peran tersebut sehingga Amazon Data Firehose dapat menerapkannya.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Principal": {
        "Service": [
          "firehose.amazonaws.com"
        ]
      },
      "Effect": "Allow",
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Pastikan bahwa peran ini yang memberikan izin Amazon Data Firehose untuk mencerna data sumber dari klaster MSK Amazon yang ditentukan memberikan izin berikut:

Berikan akses Firehose ke AWS Glue untuk konversi format data

Jika aliran Firehose Anda melakukan konversi format data, Amazon Data Firehose mereferensikan definisi tabel yang disimpan. AWS Glue Untuk memberikan Amazon Data Firehose akses yang diperlukan AWS Glue, tambahkan pernyataan berikut ke kebijakan Anda. Untuk informasi tentang cara menemukan ARN tabel, lihat Menentukan ARN Sumber Daya AWS Glue.

```
{
    "Sid": "",
    "Effect": "Allow",
    "Action": [
        "glue:GetTable",
        "glue:GetTableVersion",
        "glue:GetTableVersions"
    ],
```

```
    "Resource": [
        "arn:aws:glue:us-east-1:123456789012:catalog",
        "arn:aws:glue:us-east-1:123456789012:database/b",
        "arn:aws:glue:us-east-1:123456789012:table/b/easd"
    ]
},
{
      actions: ['glue:GetSchemaVersion'],
      grantee: options.role,
      resourceArns: ['*'],
}
```

Kebijakan yang disarankan untuk mendapatkan skema dari registri skema tidak memiliki batasan sumber daya. Untuk informasi selengkapnya, lihat contoh IAM untuk deserializer di Panduan Pengembang. AWS Glue

Berikan akses Firehose ke tujuan Amazon S3

Saat Anda menggunakan tujuan Amazon S3, Amazon Data Firehose mengirimkan data ke bucket S3 Anda dan secara opsional dapat menggunakan AWS KMS kunci yang Anda miliki untuk enkripsi data. Jika pencatatan kesalahan diaktifkan, Amazon Data Firehose juga mengirimkan kesalahan pengiriman data ke grup CloudWatch log dan aliran Anda. Anda harus memiliki peran IAM saat membuat aliran Firehose. Amazon Data Firehose mengasumsikan bahwa IAM berperan dan mendapatkan akses ke grup dan aliran bucket, kunci, serta CloudWatch log yang ditentukan.

Gunakan kebijakan akses berikut untuk mengaktifkan Amazon Data Firehose mengakses bucket dan kunci S3 Anda. AWS KMS Jika Anda tidak memiliki bucket S3, tambahkan `s3:PutObjectAcl` ke daftar tindakan Amazon S3. Ini memberi pemilik bucket akses penuh ke objek yang dikirimkan oleh Amazon Data Firehose.

JSON

```
{
    "Version":"2012-10-17",
    "Statement":
    [
        {
            "Effect": "Allow",
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:GetBucketLocation",
```

```
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStream",
                "kinesis:GetShardIterator",
                "kinesis:GetRecords",
                "kinesis:ListShards"
            ],
            "Resource": "arn:aws:kinesis:us-east-1:123456789012:stream/stream-
name"
        },
        {
           "Effect": "Allow",
           "Action": [
               "kms:Decrypt",
               "kms:GenerateDataKey"
           ],
           "Resource": [
               "arn:aws:kms:us-east-1:123456789012:key/key-id"
           ],
           "Condition": {
               "StringEquals": {
                   "kms:ViaService": "s3.us-east-1.amazonaws.com"
               },
               "StringLike": {
                   "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::amzn-s3-
demo-bucket/prefix*"
               }
           }
        },
        {
           "Effect": "Allow",
           "Action": [
               "logs:PutLogEvents"
```

```
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:123456789012:log-group:log-group-name:log-
stream:log-stream-name"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction",
                "lambda:GetFunctionConfiguration"
            ],
            "Resource": [
                "arn:aws:lambda:us-east-1:123456789012:function:function-
name:function-version"
            ]
        }
    ]
}
```

Kebijakan di atas juga memiliki pernyataan yang memungkinkan akses ke Amazon Kinesis Data Streams. Jika Anda tidak menggunakan Kinesis Data Streams sebagai sumber data, Anda dapat menghapus pernyataan tersebut. Jika Anda menggunakan Amazon MSK sebagai sumber Anda, maka Anda dapat mengganti pernyataan itu dengan yang berikut:

```
{
   "Sid":"",
   "Effect":"Allow",
   "Action":[
      "kafka:GetBootstrapBrokers",
      "kafka:DescribeCluster",
      "kafka:DescribeClusterV2",
      "kafka-cluster:Connect"
   ],
   "Resource":"arn:aws:kafka:{{mskClusterRegion}}:{{mskClusterAccount}}:cluster/
{{mskClusterName}}/{{clusterUUID}}"
},
{
   "Sid":"",
   "Effect":"Allow",
   "Action":[
      "kafka-cluster:DescribeTopic",
```

```
      "kafka-cluster:DescribeTopicDynamicConfiguration",
      "kafka-cluster:ReadData"
   ],
   "Resource":"arn:aws:kafka:{{mskClusterRegion}}:{{mskClusterAccount}}:topic/
{{mskClusterName}}/{{clusterUUID}}/{{mskTopicName}}"
},
{
   "Sid":"",
   "Effect":"Allow",
   "Action":[
      "kafka-cluster:DescribeGroup"
   ],
   "Resource":"arn:aws:kafka:{{mskClusterRegion}}:{{mskClusterAccount}}:group/
{{mskClusterName}}/{{clusterUUID}}/*"
}
```

Untuk informasi selengkapnya tentang mengizinkan AWS layanan lain mengakses AWS sumber daya Anda, lihat Membuat Peran untuk Mendelegasikan Izin ke AWS Layanan di Panduan Pengguna IAM.

Untuk mempelajari cara memberikan Amazon Data Firehose akses ke tujuan Amazon S3 di akun lain, lihat. the section called “Cross-account pengiriman ke tujuan Amazon S3”

Berikan akses Firehose ke Tabel Amazon S3

Firehose memerlukan peran IAM dengan izin khusus untuk mengakses AWS AWS Glue tabel dan menulis data ke tabel di bucket tabel Amazon S3. Untuk menulis ke tabel di keranjang tabel Amazon S3, Anda juga harus memberikan peran IAM dengan izin yang diperlukan. Izin yang diperlukan untuk katalog Amazon S3 Tables bergantung pada mode kontrol akses yang Anda gunakan:

- Kontrol akses IAM - Peran pengiriman Firehose memerlukan izin IAM langsung di sumber daya Amazon S3 Tables.
- Kontrol akses Lake Formation — Peran pengiriman Firehose memerlukan AWS AWS Lake Formation izin untuk mengelola akses ke sumber daya tabel Anda. AWS Lake Formation menggunakan model izinnya sendiri yang memungkinkan kontrol akses berbutir halus untuk sumber daya Katalog Data.

Anda mengonfigurasi peran IAM ini saat membuat aliran Firehose. Pilih tab yang sesuai dengan mode kontrol akses Anda.

Kontrol akses IAM

Jika Anda menggunakan mode kontrol akses IAM (tanpa AWS Lake Formation), peran pengiriman Firehose memerlukan izin IAM secara langsung di sumber daya Tabel Amazon S3 dan objek Katalog Data. AWS Glue

Masuk ke Konsol Manajemen AWS dan buka konsol IAM di https://console.aws.amazon.com/iam/.

Buat kebijakan dan pilih JSON di editor kebijakan. Tambahkan kebijakan inline berikut yang memberikan izin yang diperlukan.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3TablesAccessPermission",
            "Effect": "Allow",
            "Action": [
                "s3tables:GetTable",
                "s3tables:GetTableData",
                "s3tables:GetTableMetadataLocation",
                "s3tables:UpdateTableMetadataLocation"
            ],
            "Resource": [
                "arn:aws:s3tables:region:account-id:bucket/*",
                "arn:aws:s3tables:region:account-id:bucket/*/table/*"
            ]
        },
        {
            "Sid": "S3TableBucketAccessPermission",
            "Effect": "Allow",
            "Action": [
                "s3tables:GetTableBucket"
            ],
            "Resource": "arn:aws:s3tables:region:account-id:bucket/*"
        },
        {
            "Sid": "GlueCatalogAccessPermission",
            "Effect": "Allow",
            "Action": [
                "glue:GetDatabase",
                "glue:GetDatabases",
                "glue:GetTable",
```

```
                "glue:GetTables",
                "glue:UpdateTable"
            ],
            "Resource": [
                "arn:aws:glue:region:account-id:catalog",
                "arn:aws:glue:region:account-id:catalog/s3tablescatalog",
                "arn:aws:glue:region:account-id:catalog/s3tablescatalog/*",
                "arn:aws:glue:region:account-id:database/*",
                "arn:aws:glue:region:account-id:table/*/*"
            ]
        },
        {
            "Sid": "S3DeliveryErrorBucketPermission",
            "Effect": "Allow",
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::error-delivery-bucket",
                "arn:aws:s3:::error-delivery-bucket/*"
            ]
        },
        {
            "Sid": "RequiredWhenUsingKinesisDataStreamsAsSource",
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStream",
                "kinesis:GetShardIterator",
                "kinesis:GetRecords",
                "kinesis:ListShards"
            ],
            "Resource": "arn:aws:kinesis:region:account-id:stream/stream-name"
        },
        {
            "Sid": "KMSPermissionForS3TablesEncryption",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey"
```

```
            ],
            "Resource": [
                "arn:aws:kms:region:account-id:key/key-id"
            ],
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "s3.amazonaws.com"
                },
                "StringLike": {
                    "kms:EncryptionContext:aws:s3:arn":
 "arn:aws:s3tables:region:account-id:bucket/*/table/*"
                }
            }
        },
        {
            "Sid": "RequiredWhenUsingLambdaForDataTransformation",
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction",
                "lambda:GetFunctionConfiguration"
            ],
            "Resource": "arn:aws:lambda:region:account-id:function:function-
name:function-version"
        },
        {
            "Sid": "CloudWatchLogsPermission",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:region:account-id:log-group:log-group-name:log-
stream:log-stream-name"
        }
    ]
}
```

Kebijakan ini memiliki pernyataan yang mengizinkan akses ke Amazon Kinesis Data Streams, menjalankan fungsi Lambda, dan akses ke kunci. AWS KMS Jika Anda tidak menggunakan salah satu sumber daya ini, Anda dapat menghapus pernyataan masing-masing. Jika pencatatan kesalahan diaktifkan, Amazon Data Firehose juga mengirimkan kesalahan pengiriman data ke grup CloudWatch log dan aliran Anda. Anda harus mengonfigurasi grup log dan nama aliran log untuk

menggunakan opsi ini. Untuk grup log dan nama aliran log, lihatPantau Amazon Data Firehose Menggunakan Log CloudWatch.

Dalam kebijakan sebaris, ganti nilai placeholder dengan nama sumber daya, Akun AWS nomor, dan Wilayah Anda yang sebenarnya.

Setelah Anda membuat kebijakan, buka konsol IAM di https://console.aws.amazon.com/iam/dan buat peran IAM dengan Layanan AWSsebagai jenis entitas Tepercaya.

Untuk Service atau use case, pilih Kinesis. Untuk kasus Penggunaan, pilih Kinesis Firehose.

Di halaman berikutnya, pilih kebijakan yang dibuat pada langkah sebelumnya untuk dilampirkan ke peran ini. Pada halaman ulasan, Anda akan menemukan kebijakan kepercayaan yang telah dilampirkan pada peran ini yang memberikan izin ke layanan Firehose untuk mengambil peran ini. Saat Anda membuat peran, Amazon Data Firehose dapat menganggapnya melakukan operasi yang diperlukan pada AWS Glue dan Tabel Amazon S3. Tambahkan prinsip layanan Firehose ke kebijakan kepercayaan peran yang dibuat. Untuk informasi selengkapnya, lihat Izinkan Firehose untuk mengambil peran IAM.

Kontrol akses Lake Formation

Jika Anda menggunakan mode kontrol AWS Lake Formation akses, peran pengiriman Firehose memerlukan AWS Lake Formation izin untuk penjual kredenal selain kebijakan IAM.

Masuk ke Konsol Manajemen AWS dan buka konsol IAM di https://console.aws.amazon.com/iam/.

Buat kebijakan dan pilih JSON di editor kebijakan. Tambahkan kebijakan sebaris berikut yang memberikan izin Amazon S3 read/write seperti izin, izin untuk memperbarui tabel di katalog data, dan lainnya.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "S3TableAccessViaGlueFederation",
      "Effect": "Allow",
      "Action": [
        "glue:GetTable",
        "glue:GetDatabase",
```

```
        "glue:UpdateTable"
      ],
      "Resource": [
        "arn:aws:glue:us-east-1:123456789012:catalog/s3tablescatalog/*",
        "arn:aws:glue:us-east-1:123456789012:catalog/s3tablescatalog",
        "arn:aws:glue:us-east-1:123456789012:catalog",
        "arn:aws:glue:us-east-1:123456789012:database/*",
        "arn:aws:glue:us-east-1:123456789012:table/*/*"
      ]
    },
    {
      "Sid": "S3DeliveryErrorBucketPermission",
      "Effect": "Allow",
      "Action": [
        "s3:AbortMultipartUpload",
        "s3:GetBucketLocation",
        "s3:GetObject",
        "s3:ListBucket",
        "s3:ListBucketMultipartUploads",
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::<error delivery bucket>",
        "arn:aws:s3:::<error delivery bucket>/*"
      ]
    },
    {
      "Sid": "RequiredWhenUsingKinesisDataStreamsAsSource",
      "Effect": "Allow",
      "Action": [
        "kinesis:DescribeStream",
        "kinesis:GetShardIterator",
        "kinesis:GetRecords",
        "kinesis:ListShards"
      ],
      "Resource": "arn:aws:kinesis:us-east-1:123456789012:stream/<stream-name>"
    },
    {
      "Sid":
"RequiredWhenDoingMetadataReadsANDDataAndMetadataWriteViaLakeformation",
      "Effect": "Allow",
      "Action": [
        "lakeformation:GetDataAccess"
      ],
```

```
      "Resource": "*"
    },
    {
      "Sid": "RequiredWhenUsingKMSEncryptionForS3ErrorBucketDelivery",
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
      ],
      "Resource": [
        "arn:aws:kms:us-east-1:123456789012:key/<KMS-key-id>"
      ],
      "Condition": {
        "StringEquals": {
          "kms:ViaService": "s3.us-east-1.amazonaws.com"
        },
        "StringLike": {
          "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::<error delivery
 bucket>/prefix*"
        }
      }
    },
    {
      "Sid": "LoggingInCloudWatch",
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents"
      ],
      "Resource": [
        "arn:aws:logs:us-east-1:123456789012:log-group:log-group-name>:log-
stream:<log-stream-name>"
      ]
    },
    {
      "Sid": "RequiredWhenAttachingLambdaToFirehose",
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeFunction",
        "lambda:GetFunctionConfiguration"
      ],
      "Resource": [
        "arn:aws:lambda:us-east-1:123456789012:function:<function-
name>:<function-version>"
      ]
```

```
    }
  ]
}
```

Kebijakan ini memiliki pernyataan yang memungkinkan akses ke Amazon Kinesis Data Streams, menjalankan fungsi Lambda, dan akses ke kunci. AWS KMS Jika Anda tidak menggunakan salah satu sumber daya ini, Anda dapat menghapus pernyataan masing-masing. Jika pencatatan kesalahan diaktifkan, Amazon Data Firehose juga mengirimkan kesalahan pengiriman data ke grup CloudWatch log dan aliran Anda. Anda harus mengonfigurasi grup log dan nama aliran log untuk menggunakan opsi ini. Untuk grup log dan nama aliran log, lihatPantau Amazon Data Firehose Menggunakan Log CloudWatch.

Dalam kebijakan sebaris, ganti `<error delivery bucket>` dengan nama bucket Amazon S3 Anda`aws-account-id`, dan Wilayah dengan nomor dan Wilayah sumber daya yang Akun AWS valid.

Selain kebijakan IAM, Anda juga harus memberikan peran pengiriman Firehose izin yang diperlukan. AWS Lake Formation Untuk informasi selengkapnya, lihat Memberikan izin pada tabel.

Setelah Anda membuat kebijakan, buka konsol IAM di https://console.aws.amazon.com/iam/dan buat peran IAM dengan Layanan AWSsebagai jenis entitas Tepercaya.

Untuk Service atau use case, pilih Kinesis. Untuk kasus Penggunaan, pilih Kinesis Firehose.

Di halaman berikutnya, pilih kebijakan yang dibuat pada langkah sebelumnya untuk dilampirkan ke peran ini. Pada halaman ulasan, Anda akan menemukan kebijakan kepercayaan yang telah dilampirkan pada peran ini yang memberikan izin ke layanan Firehose untuk mengambil peran ini. Saat Anda membuat peran, Amazon Data Firehose dapat menganggapnya melakukan operasi yang diperlukan pada bucket S3 AWS Glue dan S3. Tambahkan prinsip layanan Firehose ke kebijakan kepercayaan peran yang dibuat. Untuk informasi selengkapnya, lihat Izinkan Firehose untuk mengambil peran IAM.

Berikan akses Firehose ke tujuan Apache Iceberg Tables

Anda harus memiliki peran IAM sebelum membuat aliran Firehose dan Apache Iceberg Tables menggunakan. AWS Glue Gunakan langkah-langkah berikut untuk membuat kebijakan dan peran IAM. Firehose mengasumsikan peran IAM ini dan melakukan tindakan yang diperlukan.

1. Masuk ke Konsol Manajemen AWS dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Buat kebijakan dan pilih JSON di editor kebijakan.
3. Tambahkan kebijakan sebaris berikut yang memberikan izin Amazon S3 seperti izin, izin untuk memperbarui tabel read/write di katalog data, dll.

   Kebijakan ini memiliki pernyataan yang memungkinkan akses ke Amazon Kinesis Data Streams, menjalankan fungsi Lambda, dan akses ke kunci KMS. Jika Anda tidak menggunakan salah satu sumber daya ini, Anda dapat menghapus pernyataan masing-masing.

   Jika pencatatan kesalahan diaktifkan, Firehose juga mengirimkan kesalahan pengiriman data ke grup CloudWatch log dan aliran Anda. Untuk ini, Anda harus mengonfigurasi grup log dan nama aliran log. Untuk grup log dan nama aliran log, lihatPantau Amazon Data Firehose Menggunakan Log CloudWatch.
4. Dalam kebijakan sebaris, ganti *amzn-s3-demo-bucket* dengan nama bucket Amazon S3, aws-account-id, dan Region dengan nomor dan Wilayah sumber daya yang Akun AWS valid.

   > **Note**
   >
   > Peran ini memberikan izin ke semua database dan tabel dalam katalog data Anda. Jika mau, Anda dapat memberikan izin hanya untuk tabel dan database tertentu.
5. Setelah Anda membuat kebijakan, buka konsol IAM dan buat peran IAM dengan Layanan AWSsebagai jenis entitas Tepercaya.
6. Untuk Service atau use case, pilih Kinesis. Untuk kasus Penggunaan pilih Kinesis Firehose.
7. Di halaman berikutnya, pilih kebijakan yang dibuat pada langkah sebelumnya untuk dilampirkan ke peran ini. Pada halaman ulasan, Anda akan menemukan kebijakan kepercayaan yang telah dilampirkan pada peran ini yang memberikan izin ke layanan Firehose untuk mengambil peran ini. Saat Anda membuat peran, Amazon Data Firehose dapat menganggapnya melakukan operasi yang diperlukan pada bucket S3 AWS Glue dan S3.

Berikan akses Firehose ke tujuan Amazon Redshift

Lihat hal berikut saat Anda memberikan akses ke Amazon Data Firehose saat menggunakan tujuan Amazon Redshift.

Topik

- Peran IAM dan kebijakan akses
- Akses VPC ke klaster yang disediakan Amazon Redshift atau grup kerja Amazon Redshift Serverless

Peran IAM dan kebijakan akses

Saat Anda menggunakan tujuan Amazon Redshift, Amazon Data Firehose mengirimkan data ke bucket S3 Anda sebagai lokasi perantara. Secara opsional dapat menggunakan AWS KMS kunci yang Anda miliki untuk enkripsi data. Amazon Data Firehose kemudian memuat data dari bucket S3 ke cluster yang disediakan Amazon Redshift atau workgroup Amazon Redshift Serverless. Jika pencatatan kesalahan diaktifkan, Amazon Data Firehose juga mengirimkan kesalahan pengiriman data ke grup CloudWatch log dan aliran Anda. Amazon Data Firehose menggunakan nama pengguna dan kata sandi Amazon Redshift yang ditentukan untuk mengakses klaster yang disediakan atau grup kerja Tanpa Server Amazon Redshift, dan menggunakan peran IAM untuk mengakses bucket, kunci, grup log, dan aliran yang ditentukan. CloudWatch Anda harus memiliki peran IAM saat membuat aliran Firehose.

Gunakan kebijakan akses berikut untuk mengaktifkan Amazon Data Firehose mengakses bucket dan kunci S3 Anda. AWS KMS Jika Anda tidak memiliki bucket S3, tambahkan `s3:PutObjectAcl` ke daftar tindakan Amazon S3, yang memberi pemilik bucket akses penuh ke objek yang dikirimkan oleh Amazon Data Firehose. Kebijakan ini juga memiliki pernyataan yang mengizinkan akses ke Amazon Kinesis Data Streams. Jika Anda tidak menggunakan Kinesis Data Streams sebagai sumber data, Anda dapat menghapus pernyataan tersebut.

JSON

```
{
"Version":"2012-10-17",
    "Statement":
    [
        {
            "Effect": "Allow",
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:PutObject"
```

```
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        },
        {
           "Effect": "Allow",
           "Action": [
               "kms:Decrypt",
               "kms:GenerateDataKey"
           ],
           "Resource": [
               "arn:aws:kms:us-east-1:123456789012:key/key-id"
           ],
           "Condition": {
               "StringEquals": {
                   "kms:ViaService": "s3.us-east-1.amazonaws.com"
               },
               "StringLike": {
                   "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::amzn-s3-
demo-bucket/prefix*"
               }
           }
        },
        {
           "Effect": "Allow",
           "Action": [
               "kinesis:DescribeStream",
               "kinesis:GetShardIterator",
               "kinesis:GetRecords",
               "kinesis:ListShards"
           ],
           "Resource": "arn:aws:kinesis:us-east-1:123456789012:stream/stream-
name"
        },
        {
           "Effect": "Allow",
           "Action": [
               "logs:PutLogEvents"
           ],
           "Resource": [
               "arn:aws:logs:us-east-1:123456789012:log-group:log-group-name:log-
stream:log-stream-name"
```

```
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction",
                "lambda:GetFunctionConfiguration"
            ],
            "Resource": [
                "arn:aws:lambda:us-east-1:123456789012:function:function-
name:function-version"
            ]
        }
    ]
}
```

Untuk informasi selengkapnya tentang mengizinkan AWS layanan lain mengakses AWS sumber daya Anda, lihat Membuat Peran untuk Mendelegasikan Izin ke AWS Layanan di Panduan Pengguna IAM.

Akses VPC ke klaster yang disediakan Amazon Redshift atau grup kerja Amazon Redshift Serverless

Jika klaster yang disediakan Amazon Redshift atau grup kerja Tanpa Server Amazon Redshift berada di cloud pribadi virtual (VPC), klaster tersebut harus dapat diakses publik dengan alamat IP publik. Selain itu, berikan akses Amazon Data Firehose ke klaster yang disediakan Amazon Redshift atau grup kerja Tanpa Server Amazon Redshift dengan membuka blokir alamat IP Amazon Data Firehose. Amazon Data Firehose saat ini menggunakan satu blok CIDR untuk setiap Wilayah yang tersedia.

Region	Blok CIDR
AS Timur (Ohio)	`13.58.135.96/27`
AS Timur (Virginia Utara)	`52.70.63.192/27`
AS Barat (California Utara)	`13.57.135.192/27`
AS Barat (Oregon)	`52.89.255.224/27`

Region	Blok CIDR
AWS GovCloud (US-East)	18.253.138.96/27
AWS GovCloud (US-West)	52.61.204.160/27
Kanada (Pusat)	35.183.92.128/27
Kanada Barat (Calgary)	40.176.98.192/27
Asia Pasifik (Hong Kong)	18.162.221.32/27
Asia Pasifik (Mumbai)	13.232.67.32/27
Asia Pasifik (Hyderabad)	18.60.192.128/27
Asia Pasifik (Seoul)	13.209.1.64/27
Asia Pasifik (Singapura)	13.228.64.192/27
Asia Pasifik (Sydney)	13.210.67.224/27
Asia Pasifik (Jakarta)	108.136.221.64/27
Asia Pasifik (Tokyo)	13.113.196.224/27
Asia Pasifik (Osaka)	13.208.177.192/27
Asia Pasifik (Thailand)	43.208.112.96/27
Asia Pasifik (Taipei)	43.212.53.160/27
China (Beijing)	52.81.151.32/27
China (Ningxia)	161.189.23.64/27
Europe (Zurich)	16.62.183.32/27
Eropa (Frankfurt)	35.158.127.160/27

Region	Blok CIDR
Eropa (Irlandia)	52.19.239.192/27
Eropa (London)	18.130.1.96/27
Eropa (Paris)	35.180.1.96/27
Eropa (Stockholm)	13.53.63.224/27
Eropa (Spanyol)	18.100.71.96/27
Timur Tengah (Bahrain)	15.185.91.0/27
Meksiko (Tengah)	78.12.207.32/27
Amerika Selatan (Sao Paulo)	18.228.1.128/27
Europe (Milan)	15.161.135.128/27
Africa (Cape Town)	13.244.121.224/27
Timur Tengah (UAE)	3.28.159.32/27
Israel (Tel Aviv)	51.16.102.0/27
Asia Pacific (Melbourne)	16.50.161.128/27
Asia Pasifik (Malaysia)	43.216.58.0/27

Untuk informasi selengkapnya tentang cara membuka blokir alamat IP, lihat langkah Otorisasi Akses ke Cluster di panduan Panduan Memulai Pergeseran Merah Amazon.

Berikan akses Firehose ke tujuan Layanan publik OpenSearch

Saat Anda menggunakan tujuan OpenSearch Layanan, Amazon Data Firehose mengirimkan data ke kluster OpenSearch Layanan, dan secara bersamaan mencadangkan gagal atau semua dokumen ke bucket S3 Anda. Jika pencatatan kesalahan diaktifkan, Amazon Data Firehose juga mengirimkan kesalahan pengiriman data ke grup CloudWatch log dan aliran Anda. Amazon Data Firehose

menggunakan peran IAM untuk mengakses domain OpenSearch Layanan tertentu, bucket S3, AWS KMS kunci, dan grup CloudWatch log serta stream. Anda harus memiliki peran IAM saat membuat aliran Firehose.

Gunakan kebijakan akses berikut untuk mengaktifkan Amazon Data Firehose mengakses bucket, domain OpenSearch Layanan, dan kunci S3 Anda. AWS KMS Jika Anda tidak memiliki bucket S3, tambahkan `s3:PutObjectAcl` ke daftar tindakan Amazon S3, yang memberi pemilik bucket akses penuh ke objek yang dikirimkan oleh Amazon Data Firehose. Kebijakan ini juga memiliki pernyataan yang mengizinkan akses ke Amazon Kinesis Data Streams. Jika Anda tidak menggunakan Kinesis Data Streams sebagai sumber data, Anda dapat menghapus pernyataan tersebut.

Untuk informasi selengkapnya tentang mengizinkan AWS layanan lain mengakses AWS sumber daya Anda, lihat Membuat Peran untuk Mendelegasikan Izin ke AWS Layanan di Panduan Pengguna IAM.

Untuk mempelajari cara memberikan Amazon Data Firehose akses ke kluster OpenSearch Layanan di akun lain, lihat. the section called “Cross-account pengiriman ke tujuan OpenSearch Layanan”

Berikan akses Firehose ke tujuan OpenSearch Layanan di VPC

Jika domain OpenSearch Layanan Anda ada di VPC, pastikan Anda memberi Amazon Data Firehose izin yang dijelaskan di bagian sebelumnya. Selain itu, Anda perlu memberi Amazon Data Firehose izin berikut untuk mengaktifkannya mengakses VPC domain OpenSearch Layanan Anda.

- `ec2:DescribeVpcs`
- `ec2:DescribeVpcAttribute`
- `ec2:DescribeSubnets`
- `ec2:DescribeSecurityGroups`
- `ec2:DescribeNetworkInterfaces`
- `ec2:CreateNetworkInterface`
- `ec2:CreateNetworkInterfacePermission`
- `ec2:DeleteNetworkInterface`

> **Important**
>
> Jangan mencabut izin ini setelah Anda membuat aliran Firehose. Jika Anda mencabut izin ini, aliran Firehose Anda akan terdegradasi atau berhenti mengirimkan data ke domain

OpenSearch layanan Anda setiap kali layanan mencoba melakukan kueri atau memperbarui ENI.

Important

Saat Anda menentukan subnet untuk mengirimkan data ke tujuan dalam VPC pribadi, pastikan Anda memiliki cukup banyak alamat IP gratis di subnet yang dipilih. Jika tidak ada alamat IP gratis yang tersedia di subnet tertentu, Firehose tidak dapat membuat atau menambahkan ENI untuk pengiriman data di VPC pribadi, dan pengiriman akan terdegradasi atau gagal.

Saat membuat atau memperbarui aliran Firehose, Anda menentukan grup keamanan untuk Firehose yang akan digunakan saat mengirim data ke domain Layanan Anda. OpenSearch Anda dapat menggunakan grup keamanan yang sama dengan yang digunakan domain OpenSearch Layanan atau yang berbeda. Jika Anda menentukan grup keamanan yang berbeda, pastikan bahwa itu memungkinkan lalu lintas HTTPS keluar ke grup keamanan domain OpenSearch Layanan. Pastikan juga bahwa grup keamanan domain OpenSearch Layanan mengizinkan lalu lintas HTTPS dari grup keamanan yang Anda tentukan saat mengonfigurasi aliran Firehose. Jika Anda menggunakan grup keamanan yang sama untuk aliran Firehose dan domain OpenSearch Layanan, pastikan aturan masuk grup keamanan mengizinkan lalu lintas HTTPS. Untuk informasi lebih lanjut tentang aturan grup keamanan, lihat Aturan grup keamanan dalam Dokumentasi Amazon VPC.

Berikan akses Firehose ke tujuan publik Tanpa Server OpenSearch

Saat Anda menggunakan tujuan OpenSearch Tanpa Server, Amazon Data Firehose mengirimkan data ke koleksi OpenSearch Tanpa Server Anda, dan secara bersamaan mencadangkan gagal atau semua dokumen ke bucket S3 Anda. Jika pencatatan kesalahan diaktifkan, Amazon Data Firehose juga mengirimkan kesalahan pengiriman data ke grup CloudWatch log dan aliran Anda. Amazon Data Firehose menggunakan peran IAM untuk mengakses kumpulan OpenSearch Tanpa Server yang ditentukan, bucket S3, AWS KMS kunci, dan grup log serta stream. CloudWatch Anda harus memiliki peran IAM saat membuat aliran Firehose.

Gunakan kebijakan akses berikut untuk mengaktifkan Amazon Data Firehose mengakses bucket S3, domain OpenSearch Tanpa Server, dan kunci Anda. AWS KMS Jika Anda tidak memiliki bucket S3, tambahkan `s3:PutObjectAcl` ke daftar tindakan Amazon S3, yang memberi pemilik bucket akses penuh ke objek yang dikirimkan oleh Amazon Data Firehose. Kebijakan ini juga memiliki pernyataan

yang mengizinkan akses ke Amazon Kinesis Data Streams. Jika Anda tidak menggunakan Kinesis Data Streams sebagai sumber data, Anda dapat menghapus pernyataan tersebut.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        },
        {
           "Effect": "Allow",
           "Action": [
               "kms:Decrypt",
               "kms:GenerateDataKey"
           ],
           "Resource": [
               "arn:aws:kms:us-east-1:123456789012:key/key-id"
           ],
           "Condition": {
               "StringEquals": {
                   "kms:ViaService": "s3.us-east-1.amazonaws.com"
               },
               "StringLike": {
                   "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::amzn-s3-
demo-bucket/prefix*"
               }
           }
        },
       {
```

```
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStream",
                "kinesis:GetShardIterator",
                "kinesis:GetRecords",
                "kinesis:ListShards"
            ],
            "Resource": "arn:aws:kinesis:us-east-1:123456789012:stream/stream-name"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:123456789012:log-group:log-group-name:log-
stream:log-stream-name"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction",
                "lambda:GetFunctionConfiguration"
            ],
            "Resource": [
                "arn:aws:lambda:us-east-1:123456789012:function:function-
name:function-version"
            ]
        },
        {
         "Effect": "Allow",
         "Action": "aoss:APIAccessAll",
         "Resource": "arn:aws:aoss:us-east-1:123456789012:collection/collection-
id"
       }
     ]
}
```

Selain kebijakan di atas, Anda juga harus mengonfigurasi Amazon Data Firehose agar izin minimum berikut ditetapkan dalam kebijakan akses data:

```
[
   {
      "Rules":[
         {
            "ResourceType":"index",
            "Resource":[
               "index/target-collection/target-index"
            ],
            "Permission":[
               "aoss:WriteDocument",
               "aoss:UpdateIndex",
               "aoss:CreateIndex"
            ]
         }
      ],
      "Principal":[
         "arn:aws:sts::123456789012:assumed-role/firehose-delivery-role-name/*"
      ]
   }
]
```

Untuk informasi selengkapnya tentang mengizinkan AWS layanan lain mengakses AWS sumber daya Anda, lihat Membuat Peran untuk Mendelegasikan Izin ke AWS Layanan di Panduan Pengguna IAM.

Berikan akses Firehose ke tujuan OpenSearch Tanpa Server di VPC

Jika koleksi OpenSearch Tanpa Server Anda ada di VPC, pastikan Anda memberi Amazon Data Firehose izin yang dijelaskan di bagian sebelumnya. Selain itu, Anda perlu memberi Amazon Data Firehose izin berikut untuk mengaktifkannya mengakses VPC koleksi OpenSearch Tanpa Server Anda.

- `ec2:DescribeVpcs`
- `ec2:DescribeVpcAttribute`
- `ec2:DescribeSubnets`
- `ec2:DescribeSecurityGroups`
- `ec2:DescribeNetworkInterfaces`
- `ec2:CreateNetworkInterface`

- `ec2:CreateNetworkInterfacePermission`
- `ec2:DeleteNetworkInterface`

> **Important**
>
> Jangan mencabut izin ini setelah Anda membuat aliran Firehose. Jika Anda mencabut izin ini, aliran Firehose Anda akan terdegradasi atau berhenti mengirimkan data ke domain OpenSearch layanan Anda setiap kali layanan mencoba melakukan kueri atau memperbarui ENI.

> **Important**
>
> Saat Anda menentukan subnet untuk mengirimkan data ke tujuan dalam VPC pribadi, pastikan Anda memiliki cukup banyak alamat IP gratis di subnet yang dipilih. Jika tidak ada alamat IP gratis yang tersedia di subnet tertentu, Firehose tidak dapat membuat atau menambahkan ENI untuk pengiriman data di VPC pribadi, dan pengiriman akan terdegradasi atau gagal.

Saat membuat atau memperbarui aliran Firehose, Anda menentukan grup keamanan untuk Firehose yang akan digunakan saat mengirimkan data ke koleksi Tanpa Server Anda. OpenSearch Anda dapat menggunakan grup keamanan yang sama dengan yang digunakan oleh koleksi OpenSearch Tanpa Server atau yang berbeda. Jika Anda menentukan grup keamanan yang berbeda, pastikan bahwa itu memungkinkan lalu lintas HTTPS keluar ke grup keamanan koleksi OpenSearch Tanpa Server. Pastikan juga bahwa grup keamanan koleksi OpenSearch Tanpa Server mengizinkan lalu lintas HTTPS dari grup keamanan yang Anda tentukan saat mengonfigurasi aliran Firehose. Jika Anda menggunakan grup keamanan yang sama untuk aliran Firehose dan koleksi OpenSearch Tanpa Server, pastikan aturan masuk grup keamanan mengizinkan lalu lintas HTTPS. Untuk informasi lebih lanjut tentang aturan grup keamanan, lihat Aturan grup keamanan dalam Dokumentasi Amazon VPC.

Berikan akses Firehose ke tujuan Splunk

Saat Anda menggunakan tujuan Splunk, Amazon Data Firehose mengirimkan data ke titik akhir Splunk HTTP Event Collector (HEC) Anda. Ini juga mencadangkan data tersebut ke bucket Amazon S3 yang Anda tentukan, dan Anda dapat menggunakan AWS KMS kunci yang Anda

miliki untuk enkripsi sisi server Amazon S3 secara opsional. Jika pencatatan kesalahan diaktifkan, Firehose mengirimkan kesalahan pengiriman data ke aliran CloudWatch log Anda. Anda juga dapat menggunakan AWS Lambda untuk transformasi data.

Jika Anda menggunakan AWS load balancer, pastikan bahwa itu adalah Classic Load Balancer atau Application Load Balancer. Selain itu, aktifkan sesi lengket berbasis durasi dengan kedaluwarsa cookie dinonaktifkan untuk Classic Load Balancer dan kedaluwarsa diatur ke maksimum (7 hari) untuk Application Load Balancer. Untuk informasi tentang cara melakukannya, lihat Duration-Based Session Stickiness for Classic Load Balancer atau Application Load Balancer.

Anda harus memiliki peran IAM saat membuat aliran Firehose. Firehose mengasumsikan bahwa IAM berperan dan mendapatkan akses ke bucket, key, dan grup CloudWatch log serta stream yang ditentukan.

Gunakan kebijakan akses berikut untuk mengaktifkan Amazon Data Firehose mengakses bucket S3 Anda. Jika Anda tidak memiliki bucket S3, tambahkan `s3:PutObjectAcl` ke daftar tindakan Amazon S3, yang memberi pemilik bucket akses penuh ke objek yang dikirimkan oleh Amazon Data Firehose. Kebijakan ini juga memberikan Amazon Data Firehose akses CloudWatch untuk pencatatan kesalahan dan AWS Lambda untuk transformasi data. Kebijakan ini juga memiliki pernyataan yang mengizinkan akses ke Amazon Kinesis Data Streams. Jika Anda tidak menggunakan Kinesis Data Streams sebagai sumber data, Anda dapat menghapus pernyataan tersebut. Amazon Data Firehose tidak menggunakan IAM untuk mengakses Splunk. Untuk mengakses Splunk, ia menggunakan token HEC Anda.

JSON

```
{
    "Version":"2012-10-17",
    "Statement":
    [
        {
            "Effect": "Allow",
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:PutObject"
            ],
```

```
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey"
            ],
            "Resource": [
                "arn:aws:kms:us-east-1:123456789012:key/key-id"
            ],
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "s3.us-east-1.amazonaws.com"
                },
                "StringLike": {
                    "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::amzn-s3-
demo-bucket/prefix*"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStream",
                "kinesis:GetShardIterator",
                "kinesis:GetRecords",
                "kinesis:ListShards"
            ],
            "Resource": "arn:aws:kinesis:us-east-1:123456789012:stream/stream-
name"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:123456789012:log-group:log-group-name:log-
stream:*"
            ]
```

```
        },
        {
           "Effect": "Allow",
           "Action": [
               "lambda:InvokeFunction",
               "lambda:GetFunctionConfiguration"
           ],
           "Resource": [
               "arn:aws:lambda:us-east-1:123456789012:function:function-
name:function-version"
           ]
        }
    ]
}
```

Untuk informasi selengkapnya tentang mengizinkan AWS layanan lain mengakses AWS sumber daya Anda, lihat Membuat Peran untuk Mendelegasikan Izin ke AWS Layanan di Panduan Pengguna IAM.

Mengakses Splunk di VPC

Jika platform Splunk Anda ada di VPC, maka platform tersebut harus dapat diakses secara publik dengan alamat IP publik. Juga, berikan Amazon Data Firehose akses ke platform Splunk Anda dengan membuka blokir alamat IP Amazon Data Firehose. Amazon Data Firehose saat ini menggunakan blok CIDR berikut.

Region	Blok CIDR
AS Timur (Ohio)	`18.216.68.160/27, 18.216.170.64/27, 18.216.17 0.96/27 \`
AS Timur (Virginia Utara)	`34.238.188.128/26, 34.238.188.192/26, 34.238.19 5.0/26`
AS Barat (California Utara)	`13.57.180.0/26`
AS Barat (Oregon)	`34.216.24.32/27, 34.216.24.192/27, 34.216.24 .224/27`

Region	Blok CIDR
AWS GovCloud (US-East)	18.253.138.192/26
AWS GovCloud (US-West)	52.61.204.192/26
Asia Pasifik (Hong Kong)	18.162.221.64/26
Asia Pasifik (Taipei)	43.212.53.192/26
Asia Pasifik (Mumbai)	13.232.67.64/26
Asia Pasifik (Seoul)	13.209.71.0/26
Asia Pasifik (Singapura)	13.229.187.128/26
Asia Pasifik (Sydney)	13.211.12.0/26
Asia Pasifik (Thailand)	43.208.112.128/26
Asia Pasifik (Tokyo)	13.230.21.0/27, 13.230.21.32/27
(Canada (Central)	35.183.92.64/26
Kanada Barat (Calgary)	40.176.98.128/26
Eropa (Frankfurt)	18.194.95.192/27, 18.194.95.224/27, 18.195.48.0/27
Eropa (Irlandia)	34.241.197.32/27, 34.241.197.64/27, 34.241.197.96/27
Eropa (London)	18.130.91.0/26
Eropa (Paris)	35.180.112.0/26
Eropa (Spanyol)	18.100.194.0/26
Eropa (Stockholm)	13.53.191.0/26

Region	Blok CIDR
Timur Tengah (Bahrain)	15.185.91.64/26
Meksiko (Tengah)	78.12.207.64/26
Amerika Selatan (Sao Paulo)	18.228.1.192/26
Europe (Milan)	15.161.135.192/26
Africa (Cape Town)	13.244.165.128/26
Asia Pasifik (Osaka)	13.208.217.0/26
China (Beijing)	52.81.151.64/26
China (Ningxia)	161.189.23.128/26
Asia Pasifik (Jakarta)	108.136.221.128/26
Timur Tengah (UAE)	3.28.159.64/26
Israel (Tel Aviv)	51.16.102.64/26
Europe (Zurich)	16.62.183.64/26
Asia Pasifik (Hyderabad)	18.60.192.192/26
Asia Pacific (Melbourne)	16.50.161.192/26
Asia Pasifik (Malaysia)	43.216.44.192/26
Asia Pasifik (Selandia Baru)	3.102.119.128/26

Menelan log aliran VPC ke Splunk menggunakan Amazon Data Firehose

Untuk mempelajari lebih lanjut tentang cara membuat langganan log aliran VPC, terbitkan ke Firehose, dan kirim log aliran VPC ke tujuan yang didukung, lihat log aliran VPC Ingest ke Splunk menggunakan Amazon Data Firehose.

Mengakses Snowflake atau titik akhir HTTP

Tidak ada subset rentang alamat AWS IP khusus untuk Amazon Data Firehose ketika tujuannya adalah titik akhir HTTP atau cluster publik Snowflake.

Untuk menambahkan Firehose ke daftar izin untuk kluster Snowflake publik atau ke titik akhir HTTP atau HTTPS publik Anda, tambahkan semua rentang alamat AWS IP saat ini ke aturan ingress Anda.

> **Note**
>
> Notifikasi tidak selalu bersumber dari alamat IP di AWS Wilayah yang sama dengan topik terkait. Anda harus menyertakan rentang alamat AWS IP untuk semua Wilayah.

Berikan akses Firehose ke tujuan Snowflake

Saat Anda menggunakan Snowflake sebagai tujuan, Firehose mengirimkan data ke akun Snowflake menggunakan URL akun Snowflake Anda. Ini juga mencadangkan data kesalahan ke bucket Amazon Simple Storage Service yang Anda tentukan, dan Anda dapat menggunakan AWS Key Management Service kunci yang Anda miliki untuk enkripsi sisi server Amazon S3 secara opsional. Jika pencatatan kesalahan diaktifkan, Firehose mengirimkan kesalahan pengiriman data ke aliran CloudWatch Log Anda.

Anda harus memiliki peran IAM sebelum membuat aliran Firehose. Firehose mengasumsikan bahwa IAM berperan dan mendapatkan akses ke grup dan aliran bucket, kunci, serta CloudWatch Log yang ditentukan. Gunakan kebijakan akses berikut untuk mengaktifkan Firehose mengakses bucket S3 Anda. Jika Anda tidak memiliki bucket S3, tambahkan `s3:PutObjectAcl` ke daftar tindakan Amazon Simple Storage Service, yang memberi pemilik bucket akses penuh ke objek yang dikirimkan oleh Firehose. Kebijakan ini juga memberikan akses Firehose CloudWatch untuk pencatatan kesalahan. Kebijakan ini juga memiliki pernyataan yang mengizinkan akses ke Amazon Kinesis Data Streams. Jika Anda tidak menggunakan Kinesis Data Streams sebagai sumber data, Anda dapat menghapus pernyataan tersebut. Firehose tidak menggunakan IAM untuk mengakses

Snowflake. Untuk mengakses Snowflake, ia menggunakan Url akun Snowflake dan PrivateLink Vpce Id Anda dalam kasus cluster pribadi.

JSON

```
{
"Version":"2012-10-17",
    "Statement":
    [
        {
"Effect": "Allow",
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        },
        {
"Effect": "Allow",
           "Action": [
               "kms:Decrypt",
               "kms:GenerateDataKey"
           ],
           "Resource": [
               "arn:aws:kms:us-east-1:123456789012:key/key-id"
           ],
           "Condition": {
"StringEquals": {
"kms:ViaService": "s3.us-east-1.amazonaws.com"
               },
               "StringLike": {
"kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::amzn-s3-demo-bucket/prefix*"
               }
           }
        },
        {
```

```
"Effect": "Allow",
            "Action": [
                "kinesis:DescribeStream",
                "kinesis:GetShardIterator",
                "kinesis:GetRecords",
                "kinesis:ListShards"
            ],
            "Resource": "arn:aws:kinesis:us-east-1:123456789012:stream/stream-
name"
        },
        {
"Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:123456789012:log-group:log-group-name:log-
stream:*"
            ]
        }
    ]
}
```

Untuk informasi selengkapnya tentang mengizinkan AWS layanan lain mengakses AWS sumber daya Anda, lihat Membuat Peran untuk Mendelegasikan Izin ke AWS Layanan di Panduan Pengguna IAM.

Mengakses Snowflake di VPC

Jika cluster Snowflake Anda mengaktifkan tautan pribadi, Firehose akan menggunakan salah satu titik akhir VPC berikut pada saat pembuatan tautan pribadi untuk mengirimkan data ke cluster pribadi Anda tanpa melalui internet publik. Untuk ini, buat aturan jaringan Snowflake untuk mengizinkan masuknya dari berikut `AwsVpceIds` untuk klaster Wilayah AWS Anda. Untuk informasi selengkapnya, lihat Membuat aturan jaringan di Panduan Pengguna Kepingan Salju.

Id Titik Akhir VPC yang akan digunakan berdasarkan Wilayah tempat klaster Anda berada

Wilayah AWS	**VPCE IDs**
AS Timur (Ohio)	vpce-0d96cafcd96a50aeb

Wilayah AWS	**VPCE IDs**
	vpce-0cec34343d48f537b

Wilayah AWS	**VPCE IDs**
AS Timur (Virginia Utara)	vpce-0b4d7e8478e141ba8 vpce-0b75cd681fb507352 vpce-01c03e63820ec00d8 vpce-0c2cfc51dc2882422 vpce-06ca862f019e4e056 vpce-020cda0cfa63f8d1c vpce-0b80504a1a783cd70 vpce-0289b9ff0b5259a96 vpce-0d7add8628bd69a12 vpce-02bfb5966cc59b2af vpce-09e707674af878bf2 vpce-049b52e96cc1a2165 vpce-0bb6c7b7a8a86cdbb vpce-03b22d599f51e80f3 vpce-01d60dc60fc106fe1 vpce-0186d20a4b24ecbef vpce-0533906401a36e416 vpce-05111fb13d396710e vpce-0694613f4fbd6f514 vpce-09b21cb25fe4cc4f4 vpce-06029c3550e4d2399

Wilayah AWS	**VPCE IDs**
	vpce-00961862a21b033da vpce-01620b9ae33273587 vpce-078cf4ec226880ac9 vpce-0d711bf076ce56381 vpce-066b7e13cbfca6f6e vpce-0674541252d9ccc26 vpce-03540b88dedb4b000 vpce-0b1828e79ad394b95 vpce-0dc0e6f001fb1a60d vpce-0d8f82e71a244098a vpce-00e374d9e3f1af5ce vpce-0c1e3d6631ddb442f
AS Barat (Oregon)	vpce-0f60f72da4cd1e4e7 vpce-0c60d21eb8b1669fd vpce-01c4e3e29afdafbef vpce-0cc6bf2a88da139de vpce-0797e08e169e50662 vpce-033cbe480381b5c0e vpce-00debbdd8f9eb10a5 vpce-08ec2f386c809e889 vpce-0856d14310857b545

Wilayah AWS	**VPCE IDs**
Eropa (Frankfurt)	vpce-068dbb7d71c9460fb vpce-0a7a7f095942d4ec9
Eropa (Irlandia)	vpce-06857e59c005a6276 vpce-04390f4f8778b75f2 vpce-011fd2b1f0aa172fd
Asia Pasifik (Tokyo)	vpce-06369e5258144e68a vpce-0f2363cdb8926fbe8
Asia Pasifik (Singapura)	vpce-049cd46cce7a12d52 vpce-0e8965a1a4bdb8941
Asia Pasifik (Seoul)	vpce-0aa444d9001e1faa1 vpce-04a49dcfd02b884
Asia Pasifik (Sydney)	vpce-048a60a182c52be63 vpce-03c19949787fd1859
Asia Pasifik (Mumbai)	vpce-0d68cb822f6f0db68 vpce-0517d32692ffcbde2
Eropa (London)	vpce-0fd1874a0ba3b9374 vpce-08091b1a85e206029
Amerika Selatan (Sao Paulo)	vpce-065169b8144e4f12e vpce-0493699f0e5762d63

Wilayah AWS	**VPCE IDs**
Kanada (Pusat)	vpce-07e6ed81689d5271f vpce-0f53239730541394c
Eropa (Paris)	vpce-09419680077e6488a vpce-0ea81ba2c08140c14
Asia Pasifik (Osaka)	vpce-0a9f003e6a7e38c05 vpce-02886510b897b1c5a
Eropa (Stockholm)	vpce-0d96410833219025a vpce-060a32f9a75ba969f
Asia Pasifik (Jakarta)	vpce-00add4b9a25e5c649 vpce-004ae2de34338a856

Berikan akses Firehose ke tujuan titik akhir HTTP

Anda dapat menggunakan Amazon Data Firehose untuk mengirimkan data ke tujuan titik akhir HTTP mana pun. Amazon Data Firehose juga mencadangkan data tersebut ke bucket Amazon S3 yang Anda tentukan, dan Anda dapat menggunakan AWS KMS kunci yang Anda miliki untuk enkripsi sisi server Amazon S3 secara opsional. Jika pencatatan kesalahan diaktifkan, Amazon Data Firehose mengirimkan kesalahan pengiriman data ke aliran CloudWatch log Anda. Anda juga dapat menggunakan AWS Lambda untuk transformasi data.

Anda harus memiliki peran IAM saat membuat aliran Firehose. Amazon Data Firehose mengasumsikan bahwa IAM berperan dan mendapatkan akses ke grup dan aliran bucket, kunci, serta CloudWatch log yang ditentukan.

Gunakan kebijakan akses berikut untuk mengaktifkan Amazon Data Firehose mengakses bucket S3 yang Anda tentukan untuk pencadangan data. Jika Anda tidak memiliki bucket S3, tambahkan `s3:PutObjectAcl` ke daftar tindakan Amazon S3, yang memberi pemilik bucket akses penuh ke objek yang dikirimkan oleh Amazon Data Firehose. Kebijakan ini juga memberikan Amazon Data Firehose akses CloudWatch untuk pencatatan kesalahan dan AWS Lambda untuk transformasi data.

Kebijakan ini juga memiliki pernyataan yang mengizinkan akses ke Amazon Kinesis Data Streams. Jika Anda tidak menggunakan Kinesis Data Streams sebagai sumber data, Anda dapat menghapus pernyataan tersebut.

> **Important**
>
> Amazon Data Firehose tidak menggunakan IAM untuk mengakses tujuan endpoint HTTP yang dimiliki oleh penyedia layanan pihak ketiga yang didukung, termasuk Datadog, Dynatrace, LogicMonitor MongoDB, New Relic, Splunk, atau Sumo Logic. Untuk mengakses tujuan titik akhir HTTP tertentu yang dimiliki oleh penyedia layanan pihak ketiga yang didukung, hubungi penyedia layanan tersebut untuk mendapatkan kunci API atau kunci akses yang diperlukan untuk mengaktifkan pengiriman data ke layanan tersebut dari Amazon Data Firehose.

Untuk informasi selengkapnya tentang mengizinkan AWS layanan lain mengakses AWS sumber daya Anda, lihat Membuat Peran untuk Mendelegasikan Izin ke AWS Layanan di Panduan Pengguna IAM.

> **Important**
>
> Saat ini Amazon Data Firehose TIDAK mendukung pengiriman data ke titik akhir HTTP di VPC.

Cross-account pengiriman dari Amazon MSK

Saat membuat aliran Firehose dari akun Firehose (misalnya, Akun B) dan sumber Anda adalah klaster MSK di AWS akun lain (Akun A), Anda harus memiliki konfigurasi berikut.

Akun A:

1. Di konsol MSK Amazon, pilih cluster yang disediakan lalu pilih Properties.
2. Di bawah Pengaturan jaringan, pilih Edit dan nyalakan Multi-VPC konektivitas.
3. Di bawah Setelan keamanan pilih Edit kebijakan klaster.
    a. Jika klaster belum memiliki kebijakan yang dikonfigurasi, periksa Sertakan prinsipal layanan Firehose dan Aktifkan pengiriman S3 lintas akun Firehose. Secara otomatis Konsol Manajemen AWS akan menghasilkan kebijakan dengan izin yang sesuai.

b. Jika klaster sudah memiliki kebijakan yang dikonfigurasi, tambahkan izin berikut ke kebijakan yang ada:

```
{
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::us-east-1:role/mskaasTestDeliveryRole"
      },
      "Action": [
        "kafka:GetBootstrapBrokers",
        "kafka:DescribeCluster",
        "kafka:DescribeClusterV2",
        "kafka-cluster:Connect"
      ],
      "Resource": "arn:aws:kafka:us-east-1:123456789012:cluster/DO-NOT-TOUCH-
mskaas-provisioned-privateLink/xxxxxxxxx-2f3a-462a-ba09-xxxxxxxxxx-20"  // ARN
 of the cluster
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws::iam::us-east-1:role/mskaasTestDeliveryRole"
      },
      "Action": [
        "kafka-cluster:DescribeTopic",
        "kafka-cluster:DescribeTopicDynamicConfiguration",
        "kafka-cluster:ReadData"
      ],
      "Resource": "arn:aws:kafka:us-east-1:arn:topic/DO-NOT-TOUCH-mskaas-
provisioned-privateLink/xxxxxxxxx-2f3a-462a-ba09-xxxxxxxxxx-20/*"//topic of the
 cluster
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::us-east-1:role/mskaasTestDeliveryRole"
      },
      "Action": "kafka-cluster:DescribeGroup",
      "Resource": "arn:aws:kafka:us-east-1:arn:group/DO-NOT-TOUCH-mskaas-
provisioned-privateLink/xxxxxxxxx-2f3a-462a-ba09-xxxxxxxxxx-20/*" //topic of
 the cluster
    },
```

```
}
```

4. Di bawah AWS prinsipal, masukkan ID utama dari Akun B.
5. Di bawah Topik, tentukan topik Apache Kafka dari mana Anda ingin aliran Firehose Anda menyerap data. Setelah aliran Firehose dibuat, Anda tidak dapat memperbarui topik ini.
6. Pilih Save changes (Simpan perubahan)

Akun B:

1. Di Firehose console, pilih Create Firehose stream menggunakan Akun B.
2. Di bawah Sumber, pilih Amazon Managed Streaming for Apache Kafka.
3. Di bawah pengaturan Sumber, untuk cluster Amazon Managed Streaming for Apache Kafka, masukkan ARN cluster MSK Amazon di Akun A.
4. Di bawah Topik, tentukan topik Apache Kafka dari mana Anda ingin aliran Firehose Anda menyerap data. Setelah aliran Firehose dibuat, Anda tidak dapat memperbarui topik ini.
5. Di Nama aliran pengiriman, tentukan nama untuk aliran Firehose Anda.

Di Akun B saat membuat aliran Firehose, Anda harus memiliki peran IAM (dibuat secara default saat menggunakan Konsol Manajemen AWS) yang memberikan akses 'baca' aliran Firehose ke klaster MSK Amazon lintas akun untuk topik yang dikonfigurasi.

Berikut ini adalah apa yang dikonfigurasi oleh Konsol Manajemen AWS:

```
{
    "Sid": "",
    "Effect": "Allow",
    "Action": [
        "kafka:GetBootstrapBrokers",
        "kafka:DescribeCluster",
        "kafka:DescribeClusterV2",
        "kafka-cluster:Connect"
        ],
    "Resource": "arn:aws:kafka:us-east-1:arn:aws::cluster/DO-NOT-TOUCH-mskaas-
provisioned-privateLink/xxxxxxxxx-2f3a-462a-ba09-xxxxxxxxxx-20/*" //topic of the
 cluster
    },
    {
    "Sid": "",
    "Effect": "Allow",
```

```
    "Action": [
        "kafka-cluster:DescribeTopic",
        "kafka-cluster:DescribeTopicDynamicConfiguration",
        "kafka-cluster:ReadData"
    ],
    "Resource": "arn:aws:kafka:us-east-1:arn:aws:topic/DO-NOT-TOUCH-mskaas-
provisioned-privateLink/xxxxxxxxx-2f3a-462a-ba09-xxxxxxxxxx-20/mskaas_test_topic" //
topic of the cluster
    },
    {
    "Sid": "",
    "Effect": "Allow",
    "Action": [
        "kafka-cluster:DescribeGroup"
    ],
    "Resource": "arn:aws:kafka:us-east-1:arn:aws:group/DO-NOT-TOUCH-mskaas-
provisioned-privateLink/xxxxxxxxx-2f3a-462a-ba09-xxxxxxxxxx-20/*" //topic of the
 cluster
    },
 }
```

Selanjutnya, Anda dapat menyelesaikan langkah opsional untuk mengonfigurasi transformasi catatan dan konversi format rekaman. Untuk informasi selengkapnya, lihat (Opsional) Konfigurasikan transformasi rekaman dan konversi format.

Cross-account pengiriman ke tujuan Amazon S3

Anda dapat menggunakan AWS CLI atau Amazon Data Firehose API untuk membuat aliran Firehose di satu AWS akun dengan tujuan Amazon S3 di akun lain. Prosedur berikut menunjukkan contoh mengonfigurasi aliran Firehose yang dimiliki oleh akun A untuk mengirimkan data ke bucket Amazon S3 yang dimiliki oleh akun B.

1. Buat peran IAM di bawah akun A menggunakan langkah-langkah yang dijelaskan dalam Berikan Akses Firehose ke Tujuan Amazon S3.

   > **Note**
   >
   > Dalam kasus ini, bucket Amazon S3 yang ditentukan dalam kebijakan akses dimiliki oleh akun B. Pastikan Anda menambahkan `s3:PutObjectAcl` ke daftar tindakan Amazon S3 dalam kebijakan akses, yang memberikan akun B akses penuh ke objek yang dikirimkan oleh Amazon Data Firehose. Izin ini diperlukan untuk pengiriman lintas

akun. Amazon Data Firehose menetapkan header “x-amz-acl” pada permintaan ke “bucket-owner-full-control”.

2. Untuk mengizinkan akses dari IAM role yang dibuat sebelumnya, buat kebijakan bucket S3 pada akun B. Kode berikut adalah contoh kebijakan bucket. Untuk informasi selengkapnya, lihat Menggunakan Kebijakan Bucket dan Kebijakan Pengguna.

   JSON

```
{

    "Version":"2012-10-17",
    "Id": "PolicyID",
    "Statement": [
        {
            "Sid": "StmtID",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:role/iam-role-name"
            },
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        }
    ]
}
```

3. Buat aliran Firehose di bawah akun A menggunakan peran IAM yang Anda buat di langkah 1.

Cross-account pengiriman ke tujuan OpenSearch Layanan

Anda dapat menggunakan AWS CLI atau Amazon Data Firehose API untuk membuat aliran Firehose di satu AWS akun dengan tujuan OpenSearch Layanan di akun lain. Prosedur berikut menunjukkan contoh bagaimana Anda dapat membuat aliran Firehose di bawah akun A dan mengonfigurasinya untuk mengirimkan data ke tujuan OpenSearch Layanan yang dimiliki oleh akun B.

1. Buat IAM role di akun A menggunakan langkah-langkah yang dijelaskan pada the section called "Berikan akses Firehose ke tujuan Layanan publik OpenSearch".
2. Untuk mengizinkan akses dari peran IAM yang Anda buat pada langkah sebelumnya, buat kebijakan OpenSearch Layanan di bawah akun B. JSON berikut adalah contohnya.

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:role/firehose_delivery_role "
      },
      "Action": "es:ESHttpGet",
      "Resource": [
        "arn:aws:es:us-east-1:123456789012:domain/cross-account-cluster/_all/
_settings",
        "arn:aws:es:us-east-1:123456789012:domain/cross-account-cluster/
_cluster/stats",
        "arn:aws:es:us-east-1:123456789012:domain/cross-account-cluster/
roletest*/_mapping/roletest",
        "arn:aws:es:us-east-1:123456789012:domain/cross-account-cluster/
_nodes",
        "arn:aws:es:us-east-1:123456789012:domain/cross-account-cluster/
_nodes/stats",
        "arn:aws:es:us-east-1:123456789012:domain/cross-account-cluster/
_nodes/*/stats",
        "arn:aws:es:us-east-1:123456789012:domain/cross-account-cluster/
_stats",
        "arn:aws:es:us-east-1:123456789012:domain/cross-account-cluster/
roletest*/_stats",
        "arn:aws:es:us-east-1:123456789012:domain/cross-account-cluster/"
```

```
      ]
    }
  ]
}
```

3. Buat aliran Firehose di bawah akun A menggunakan peran IAM yang Anda buat di langkah 1. Saat Anda membuat aliran Firehose, gunakan API Firehose AWS CLI Data Amazon atau Amazon dan tentukan `ClusterEndpoint` bidangnya, bukan untuk Layanan. `DomainARN` OpenSearch

> **Note**
>
> Untuk membuat aliran Firehose di satu AWS akun dengan tujuan OpenSearch Layanan di akun lain, Anda harus menggunakan AWS CLI atau Amazon Data Firehose API. Anda tidak dapat menggunakan Konsol Manajemen AWS untuk membuat konfigurasi lintas akun semacam ini.

Menggunakan tanda untuk mengontrol akses

Anda dapat menggunakan `Condition` elemen opsional (atau `Condition` memblokir) dalam kebijakan IAM untuk menyempurnakan akses ke operasi Amazon Data Firehose berdasarkan kunci dan nilai tag. Subbagian berikut menjelaskan cara melakukan ini untuk operasi Amazon Data Firehose yang berbeda. Untuk informasi selengkapnya tentang penggunaan elemen `Condition` dan operator yang dapat Anda gunakan di dalamnya, lihat Elemen Kebijakan JSON IAM: Syarat.

CreateDeliveryStream

Untuk operasi `CreateDeliveryStream`, gunakan kunci syarat `aws:RequestTag`. Pada contoh berikut, `MyKey` dan `MyValue` mewakili kunci dan nilai yang sesuai untuk sebuah tanda. Untuk informasi selengkapnya, lihat Memahami dasar-dasar tag

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
```

```
            "firehose:CreateDeliveryStream",
            "firehose:TagDeliveryStream"
        ],
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "aws:RequestTag/MyKey": "MyValue"
            }
        }
    }]
}
```

TagDeliveryStream

Untuk operasi `TagDeliveryStream`, gunakan kunci syarat `aws:TagKeys`. Pada contoh berikut, `MyKey` adalah contoh kunci tanda.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "firehose:TagDeliveryStream",
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "aws:TagKeys": "MyKey"
                 }
            }
        }
    ]
}
```

UntagDeliveryStream

Untuk operasi `UntagDeliveryStream`, gunakan kunci syarat `aws:TagKeys`. Pada contoh berikut, `MyKey` adalah contoh kunci tanda.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "firehose:UntagDeliveryStream",
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "aws:TagKeys": "MyKey"
                 }
            }
        }
    ]
}
```

ListDeliveryStreams

Anda tidak dapat menggunakan kendali akses berbasis tanda dengan `ListDeliveryStreams`.

Operasi lainnya

Untuk semua operasi Firehose selain`CreateDeliveryStream`,,`TagDeliveryStream`, dan `UntagDeliveryStream``ListDeliveryStreams`, gunakan tombol `aws:RequestTag` kondisi. Pada contoh berikut, `MyKey` dan `MyValue` mewakili kunci dan nilai yang sesuai untuk sebuah tanda.

`ListDeliveryStreams`, gunakan tombol `firehose:ResourceTag` kondisi untuk mengontrol akses berdasarkan tag pada aliran Firehose itu.

Pada contoh berikut, `MyKey` dan `MyValue` mewakili kunci dan nilai yang sesuai untuk sebuah tanda. Kebijakan ini hanya akan berlaku untuk aliran Firehose Data yang memiliki tag bernama `MyKey` dengan nilai. `MyValue` Untuk informasi selengkapnya tentang mengontrol akses berdasarkan tag sumber daya, lihat Mengontrol akses ke AWS sumber daya menggunakan tag di Panduan Pengguna IAM.

Otentikasi dengan AWS Secrets Manager di Amazon Data Firehose

Amazon Data Firehose terintegrasi AWS Secrets Manager untuk menyediakan akses aman ke rahasia Anda dan mengotomatiskan rotasi kredensi. Integrasi ini memungkinkan Firehose untuk mengambil rahasia dari Secrets Manager saat runtime untuk terhubung ke tujuan streaming yang disebutkan sebelumnya dan mengirimkan aliran data Anda. Dengan ini, rahasia Anda tidak terlihat dalam teks biasa selama alur kerja pembuatan aliran baik dalam Konsol Manajemen AWS atau parameter API. Ini memberikan praktik yang aman untuk mengelola rahasia Anda dan membebaskan Anda dari aktivitas manajemen kredensitas yang kompleks seperti menyiapkan fungsi Lambda khusus untuk mengelola rotasi kata sandi.

Untuk informasi selengkapnya, silakan lihat Panduan Pengguna AWS Secrets Manager.

Topik

Memahami rahasia

Rahasia dapat berupa kata sandi, seperangkat kredensil seperti nama pengguna dan kata sandi, token OAuth, atau informasi rahasia lainnya yang Anda simpan dalam bentuk terenkripsi di Secrets Manager.

Untuk setiap tujuan, Anda harus menentukan pasangan nilai kunci rahasia dalam format JSON yang benar seperti yang ditunjukkan pada bagian berikut. Amazon Data Firehose akan gagal terhubung ke tujuan Anda jika rahasia Anda tidak memiliki format JSON yang benar sesuai tujuan.

Format rahasia untuk database seperti MySQL dan PostgreSQL

```
{
    "username":  "<username>",
    "password":  "<password>"
}
```

Format rahasia untuk cluster Amazon Redshift Provisioned dan grup kerja Amazon Redshift Tanpa Server

```
{
    "username":  "<username>",
    "password":  "<password>"
}
```

Format rahasia untuk Splunk

```
{
    "hec_token":  "<hec token>"
}
```

Format rahasia untuk Snowflake

```
{
    "user":  "<snowflake-username>",
    "private_key":  "<snowflake-private-key>", // without the beginning and ending
 private key, remove all spaces and newlines
    "key_passphrase":  "<snowflake-private-key-passphrase>" // optional
}
```

Format rahasia untuk titik akhir HTTP, Coralogix, Datadog, Dynatrace, Elastic, Honeycomb,,, MongoDB Cloud, dan New Relic LogicMonitor Logz.io

```
{
    "api_key":  "<apikey>"
}
```

Buat rahasia

Untuk membuat rahasia, ikuti langkah-langkah di Buat AWS Secrets Manager rahasia di Panduan AWS Secrets Manager Pengguna.

Gunakan rahasianya

Kami menyarankan Anda menggunakan AWS Secrets Manager untuk menyimpan kredensil atau kunci Anda untuk terhubung ke tujuan streaming seperti Amazon Redshift, titik akhir HTTP,

Snowflake, Splunk, Coralogix, Datadog, Dynatrace, Elastic, Honeycomb,,, MongoDB Cloud, dan New Relic. LogicMonitor Logz.io

Anda dapat mengonfigurasi autentikasi dengan Secrets Manager untuk tujuan ini melalui AWS Management Console pada saat pembuatan aliran Firehose. Untuk informasi selengkapnya, lihat Konfigurasikan pengaturan tujuan. Atau, Anda juga dapat menggunakan operasi UpdateDestinationAPI CreateDeliveryStreamdan untuk mengonfigurasi otentikasi dengan Secrets Manager.

Firehose menyimpan rahasia dengan enkripsi dan menggunakannya untuk setiap koneksi ke tujuan. Ini menyegarkan cache setiap 10 menit untuk memastikan bahwa kredenal terbaru digunakan.

Anda dapat memilih untuk mematikan kemampuan mengambil rahasia dari Secrets Manager kapan saja selama siklus hidup streaming. Jika Anda tidak ingin menggunakan Secrets Manager untuk mengambil rahasia, Anda dapat menggunakan username/password atau kunci API sebagai gantinya.

> **Note**
>
> Meskipun, tidak ada biaya tambahan untuk fitur ini di Firehose, Anda ditagih untuk akses dan pemeliharaan Secrets Manager. Untuk informasi selengkapnya, lihat halaman harga AWS Secrets Manager.

Berikan akses ke Firehose untuk mengambil rahasia

Agar Firehose dapat mengambil rahasia AWS Secrets Manager, Anda harus memberikan Firehose izin yang diperlukan untuk mengakses rahasia dan kunci yang mengenkripsi rahasia Anda.

Saat menggunakan AWS Secrets Manager untuk menyimpan dan mengambil rahasia, ada beberapa opsi konfigurasi yang berbeda tergantung di mana rahasia disimpan dan bagaimana itu dienkripsi.

- Jika rahasia disimpan di AWS akun yang sama dengan peran IAM Anda dan dienkripsi dengan kunci AWS terkelola default (`aws/secretsmanager`), peran IAM yang diasumsikan Firehose hanya memerlukan izin pada rahasia tersebut. `secretsmanager:GetSecretValue`

```
// secret role policy
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "Secret ARN"
        }
    ]
}
```

Untuk informasi selengkapnya tentang kebijakan IAM, lihat contoh kebijakan izin untuk. AWS Secrets Manager

- Jika rahasia disimpan di akun yang sama dengan peran tetapi dienkripsi dengan kunci yang dikelola pelanggan (CMK), peran tersebut membutuhkan keduanya `secretsmanager:GetSecretValue` dan izin. `kms:Decrypt` Kebijakan CMK juga perlu memungkinkan peran IAM untuk melakukan. `kms:Decrypt`
- Jika rahasia disimpan di AWS akun yang berbeda dari peran Anda, dan dienkripsi dengan kunci AWS terkelola default, konfigurasi ini tidak dimungkinkan karena Secrets Manager tidak mengizinkan akses lintas akun ketika rahasia dienkripsi dengan kunci terkelola. AWS
- Jika rahasia disimpan di akun yang berbeda dan dienkripsi dengan CMK, peran IAM memerlukan `secretsmanager:GetSecretValue` izin pada rahasia dan `kms:Decrypt` izin pada CMK. Kebijakan sumber daya rahasia dan kebijakan CMK di akun lain juga perlu mengizinkan peran IAM izin yang diperlukan. Untuk informasi selengkapnya, lihat Cross-account akses.

Putar rahasianya

Rotasi adalah saat Anda memperbarui rahasia secara berkala. Anda dapat mengonfigurasi AWS Secrets Manager untuk secara otomatis memutar rahasia untuk Anda pada jadwal yang Anda tentukan. Dengan cara ini, Anda dapat mengganti rahasia jangka panjang dengan rahasia jangka pendek. Ini membantu mengurangi risiko kompromi. Untuk informasi selengkapnya, lihat Memutar AWS Secrets Manager rahasia di Panduan AWS Secrets Manager Pengguna.

Kelola peran IAM melalui konsol Amazon Data Firehose

Amazon Data Firehose adalah layanan yang dikelola sepenuhnya yang memberikan data streaming waktu nyata ke tujuan. Anda juga dapat mengonfigurasi Firehose untuk mengubah dan mengonversi format data Anda sebelum pengiriman. Untuk menggunakan fitur ini, Anda harus terlebih dahulu memberikan peran IAM untuk memberikan izin ke Firehose saat membuat atau mengedit aliran Firehose. Firehose menggunakan peran IAM ini untuk semua izin yang dibutuhkan aliran Firehose.

Misalnya, pertimbangkan skenario di mana Anda membuat aliran Firehose yang mengirimkan data ke Amazon S3, dan aliran Firehose ini memiliki Transform source record dengan fitur yang diaktifkan. AWS Lambda Dalam hal ini, Anda harus memberikan peran IAM untuk memberikan izin Firehose untuk mengakses bucket S3 dan menjalankan fungsi Lambda, seperti yang ditunjukkan di bawah ini.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Sid": "lambdaProcessing",
        "Effect": "Allow",
        "Action": ["lambda:InvokeFunction", "lambda:GetFunctionConfiguration"],
        "Resource": "arn:aws:lambda:us-east-1:123456789012:function:<lambda
 function name>:<lambda function version>"
    }, {
        "Sid": "s3Permissions",
        "Effect": "Allow",
        "Action": ["s3:AbortMultipartUpload", "s3:GetBucketLocation",
 "s3:GetObject", "s3:ListBucket", "s3:ListBucketMultipartUploads",
 "s3:PutObject"],
        "Resource": ["arn:aws:s3:::<bucket name>", "arn:aws:s3:::<bucket name>/
*"]
    }]
}
```

Firehose console memungkinkan Anda untuk memilih bagaimana Anda ingin memberikan peran ini. Anda dapat memilih dari salah satu opsi berikut.

- Pilih peran IAM yang ada
- Buat peran IAM baru dari konsol

Pilih peran IAM yang ada

Anda dapat memilih dari peran IAM yang ada. Dengan opsi ini, pastikan bahwa peran IAM yang Anda pilih memiliki kebijakan kepercayaan yang tepat dan izin yang diperlukan untuk sumber dan tujuan Anda. Untuk informasi selengkapnya, lihat Mengontrol akses dengan Amazon Data Firehose.

Buat peran IAM baru dari konsol

Atau, Anda juga dapat menggunakan konsol Firehose untuk membuat peran baru atas nama Anda.

Saat Firehose membuat peran IAM atas nama Anda, peran tersebut secara otomatis menyertakan semua kebijakan izin dan kepercayaan yang memberikan izin yang diperlukan berdasarkan konfigurasi aliran Firehose.

Misalnya, jika Anda tidak mengaktifkan data sumber Transform dengan AWS Lambda fitur, konsol akan menghasilkan pernyataan berikut dalam kebijakan izin.

```
{
  "Sid": "lambdaProcessing",
  "Effect": "Allow",
   "Action": [
     "lambda:InvokeFunction",
     "lambda:GetFunctionConfiguration"
   ],
   "Resource": "arn:aws:lambda:us-east-1123456789012:function:
%FIREHOSE_POLICY_TEMPLATE_PLACEHOLDER%"
}
```

> **Note**
>
> Aman untuk mengabaikan pernyataan kebijakan yang berisi `%FIREHOSE_POLICY_TEMPLATE_PLACEHOLDER%` karena tidak memberikan izin pada sumber daya apa pun.

Konsol yang membuat dan mengedit alur kerja aliran Firehose juga membuat kebijakan kepercayaan dan melampirkannya ke peran IAM. Kebijakan kepercayaan memungkinkan Firehose untuk mengambil peran IAM. Berikut ini adalah contoh kebijakan kepercayaan.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Sid": "firehoseAssume",
```

```
        "Effect": "Allow",
        "Principal": {
            "Service": "firehose.amazonaws.com"
        },
        "Action": "sts:AssumeRole"
    }]
}
```

Important

- Anda harus menghindari penggunaan peran IAM yang dikelola konsol yang sama untuk beberapa aliran Firehose. Jika tidak, peran IAM bisa menjadi terlalu permisif atau mengakibatkan kesalahan.
- Untuk menggunakan pernyataan kebijakan yang berbeda dalam kebijakan izin dari peran IAM yang dikelola konsol, Anda dapat membuat peran IAM Anda sendiri, dan menyalin pernyataan kebijakan ke kebijakan izin yang dilampirkan ke peran baru. Untuk melampirkan peran ke aliran Firehose, pilih opsi Pilih peran IAM yang ada di akses Layanan.
- Console mengelola peran IAM apa pun yang berisi peran layanan string di ARN-nya. Saat Anda memilih opsi peran IAM yang ada, pastikan untuk memilih peran IAM tanpa string peran layanan di ARN-nya sehingga konsol tidak membuat perubahan apa pun padanya.

Langkah-langkah untuk membuat peran IAM dari konsol

1. Buka konsol Firehose di. https://console.aws.amazon.com/firehose/
2. Pilih Buat aliran Firehose.
3. Pilih sumber dan tujuan. Untuk informasi selengkapnya, lihat Tutorial: Membuat aliran Firehose dari konsol.
4. Pilih pengaturan tujuan. Untuk informasi selengkapnya, lihat Konfigurasikan pengaturan tujuan.
5. Di bawah Pengaturan lanjutan, untuk akses Layanan, pilih Buat atau perbarui peran IAM.

> **Note**
>
> Ini adalah opsi default. Untuk menggunakan peran yang ada, pilih opsi Pilih peran IAM yang ada. Firehose console tidak akan membuat perubahan apa pun pada peran Anda sendiri.

6. Pilih Buat aliran Firehose.

Edit peran IAM dari konsol

Saat Anda mengedit aliran Firehose, Firehose memperbarui kebijakan izin yang sesuai untuk mencerminkan perubahan konfigurasi dan izin.

Misalnya, saat Anda mengedit aliran Firehose dan mengaktifkan Transform data sumber dengan AWS Lambda fitur menggunakan fungsi Lambda versi terbaru`exampleLambdaFunction`, Anda mendapatkan pernyataan kebijakan berikut dalam kebijakan izin.

```
{
  "Sid": "lambdaProcessing",
  "Effect": "Allow",
  "Action": [
    "lambda:InvokeFunction",
    "lambda:GetFunctionConfiguration"
  ],
  "Resource": "arn:aws:lambda:us-east-1:123456789012:function:exampleLambdaFunction:
$LATEST"
}
```

> **Important**
>
> Peran IAM yang dikelola konsol dirancang untuk menjadi otonom. Kami tidak menyarankan Anda mengubah kebijakan izin atau kebijakan kepercayaan di luar konsol.

Langkah-langkah untuk mengedit peran IAM dari konsol

1. Buka konsol Firehose di. https://console.aws.amazon.com/firehose/
2. Pilih aliran Firehose dan pilih nama aliran Firehose yang ingin Anda perbarui.

3. Pada tab Konfigurasi, di bagian Akses server, pilih Edit.
4. Perbarui opsi peran IAM.

> **Note**
>
> Secara default, konsol selalu memperbarui peran IAM dengan peran layanan pola di ARN-nya. Saat Anda memilih opsi peran IAM yang ada, pastikan untuk memilih peran IAM tanpa string peran layanan di ARN-nya sehingga konsol tidak membuat perubahan apa pun padanya.

5. Pilih Simpan perubahan.

Memahami kepatuhan untuk Amazon Data Firehose

Third-party auditor menilai keamanan dan kepatuhan Amazon Data Firehose sebagai bagian dari AWS beberapa program kepatuhan. Program ini mencakup SOC, PCI, FedRAMP, HIPAA, dan lainnya.

Untuk daftar AWS layanan dalam lingkup program kepatuhan tertentu, lihat AWS Layanan dalam Lingkup berdasarkan Program Kepatuhan. 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 Artifak.

Tanggung jawab kepatuhan Anda saat menggunakan Data Firehose ditentukan oleh sensitivitas data Anda, tujuan kepatuhan perusahaan Anda, serta hukum dan peraturan yang berlaku. Jika penggunaan Firehose Data Anda tunduk pada kepatuhan terhadap standar seperti HIPAA, PCI, atau FedRAMP, menyediakan sumber daya untuk membantu: AWS

- Panduan Memulai Cepat Keamanan dan Kepatuhan — Panduan penerapan ini membahas pertimbangan arsitektur dan memberikan langkah-langkah untuk menerapkan lingkungan dasar yang berfokus pada keamanan dan kepatuhan. AWS
- Arsitektur untuk HIPAA Security and Compliance Whitepaper — Whitepaper ini menjelaskan bagaimana perusahaan dapat menggunakan untuk membuat aplikasi. AWS HIPAA-compliant
- AWS Sumber Daya Kepatuhan — Kumpulan buku kerja dan panduan ini mungkin berlaku untuk industri dan lokasi Anda.
- AWS Config AWS Layanan ini menilai seberapa baik konfigurasi sumber daya Anda mematuhi praktik internal, pedoman industri, dan peraturan.

- AWS Security Hub CSPM— AWS Layanan ini memberikan pandangan komprehensif tentang keadaan keamanan Anda di dalamnya AWS yang membantu Anda memeriksa kepatuhan Anda terhadap standar industri keamanan dan praktik terbaik.

Ketahanan di Amazon Data Firehose

Infrastruktur AWS global dibangun di sekitar AWS Wilayah dan Zona Ketersediaan. AWS Wilayah menyediakan beberapa Availability Zone yang terpisah secara fisik dan terisolasi, yang terhubung dengan latensi rendah, throughput tinggi, dan jaringan yang sangat redundan. Dengan Zona Ketersediaan, Anda dapat merancang dan mengoperasikan aplikasi dan basis data yang secara otomatis melakukan failover di antara Zona Ketersediaan tanpa gangguan. Zona Ketersediaan memiliki ketersediaan dan toleransi kesalahan yang lebih baik, dan dapat diskalakan dibandingkan infrastruktur biasa yang terdiri dari satu atau beberapa pusat data.

Untuk informasi selengkapnya tentang AWS Wilayah dan Availability Zone, lihat Infrastruktur AWS Global.

Selain infrastruktur AWS global, Data Firehose menawarkan beberapa fitur untuk membantu mendukung ketahanan data dan kebutuhan pencadangan Anda.

Pemulihan bencana

Amazon Data Firehose berjalan dalam mode tanpa server, dan menangani degradasi host, ketersediaan Zona Ketersediaan, dan masalah terkait infrastruktur lainnya dengan melakukan migrasi otomatis. Ketika ini terjadi, Amazon Data Firehose memastikan bahwa aliran Firehose dimigrasikan tanpa kehilangan data.

Memahami keamanan infrastruktur di Amazon Data Firehose

Sebagai layanan terkelola, Amazon Data Firehose dilindungi oleh keamanan jaringan AWS global. Untuk informasi tentang layanan AWS keamanan dan cara AWS melindungi infrastruktur, lihat Keamanan AWS Cloud. Untuk mendesain AWS lingkungan Anda menggunakan praktik terbaik untuk keamanan infrastruktur, lihat Perlindungan Infrastruktur dalam Kerangka Kerja yang AWS Diarsiteksikan dengan Baik Pilar Keamanan.

Anda menggunakan panggilan API yang AWS dipublikasikan untuk mengakses Firehose melalui jaringan. Klien harus mendukung hal-hal berikut:

- Keamanan Lapisan Pengangkutan (TLS). Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.

- Cipher suite dengan perfect forward secrecy (PFS) seperti DHE (Ephemeral) atau ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Diffie-Hellman Sebagian besar sistem modern seperti Java 7 dan versi lebih baru mendukung mode-mode ini.

> **Note**
>
> Untuk permintaan HTTPS keluar, Amazon Data Firehose menggunakan pustaka HTTP yang secara otomatis memilih versi protokol TLS tertinggi yang didukung di sisi tujuan.

Menggunakan Amazon Data Firehose dengan AWS PrivateLink

Anda dapat menggunakan antarmuka VPC endpoint (AWS PrivateLink) untuk mengakses Amazon Data Firehose dari VPC Anda tanpa memerlukan Internet Gateway atau NAT Gateway. Endpoint VPC antarmuka tidak memerlukan gateway internet, perangkat NAT, koneksi VPN, atau koneksi. Direct Connect Endpoint VPC antarmuka didukung oleh AWS PrivateLink, sebuah AWS teknologi yang memungkinkan komunikasi pribadi antar AWS layanan menggunakan antarmuka network elastis dengan IP pribadi di Amazon VPC Anda. Untuk informasi selengkapnya, lihat Amazon Virtual Private Cloud.

Menggunakan titik akhir VPC antarmuka (AWS PrivateLink) untuk Firehose

Untuk memulai, buat titik akhir VPC antarmuka agar lalu lintas Amazon Data Firehose Anda dari sumber daya Amazon VPC Anda mulai mengalir melalui titik akhir VPC antarmuka. Saat membuat endpoint, Anda dapat melampirkan kebijakan endpoint yang mengontrol akses ke Amazon Data Firehose. Untuk selengkapnya tentang menggunakan kebijakan untuk mengontrol akses dari titik akhir VPC ke Amazon Data Firehose, lihat Mengontrol Akses ke Layanan dengan Titik Akhir VPC.

Contoh berikut menunjukkan bagaimana Anda dapat mengatur AWS Lambda fungsi di VPC dan membuat titik akhir VPC untuk memungkinkan fungsi berkomunikasi secara aman dengan layanan Amazon Data Firehose. Dalam contoh ini, Anda menggunakan kebijakan yang memungkinkan fungsi Lambda mencantumkan aliran Firehose di Wilayah saat ini, tetapi tidak menjelaskan aliran Firehose apa pun.

Buat VPC endpoint

1. Masuk ke Konsol Manajemen AWS dan buka konsol VPC Amazon di. https://console.aws.amazon.com/vpc/

2. Pada Dasbor VPC pilih Titik Akhir.
3. Pilih Buat Titik Akhir.
4. Pada daftar nama layanan, pilih `com.amazonaws.your_region.kinesis-firehose`.
5. Pilih VPC dan satu atau beberapa subnet yang akan jadi tempat menciptakan titik akhir.
6. Pilih satu atau beberapa grup keamanan untuk dikaitkan dengan titik akhir.
7. Untuk Kebijakan, pilih Kustom dan tempelkan kebijakan berikut ini:

```
{
    "Statement": [
        {
            "Sid": "Allow-only-specific-PrivateAPIs",
            "Principal": "*",
            "Action": [
                "firehose:ListDeliveryStreams"
            ],
            "Effect": "Allow",
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "Allow-only-specific-PrivateAPIs",
            "Principal": "*",
            "Action": [
                "firehose:DescribeDeliveryStream"
            ],
            "Effect": "Deny",
            "Resource": [
                "*"
            ]
        }
    ]
}
```

8. Pilih Buat titik akhir.

Buat IAM role untuk digunakan dengan fungsi Lambda

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel kiri, pilih Peran lalu pilih Buat peran.

3. Di bawah Pilih jenis entitas tepercaya, pertahankan pilihan default Layanan AWS .
4. Di bawah Pilih layanan yang akan menggunakan di peran ini, pilih Lambda.
5. Pilih Berikutnya: Izin.
6. Dari daftar kebijakan, cari dan tambahkan dua kebijakan bernama `AWS LambdaVPCAccessExecutionRole` dan `AmazonDataFirehoseReadOnlyAccess`.

> **Important**
>
> Ini adalah contoh. Anda mungkin memerlukan kebijakan yang lebih ketat untuk lingkungan produksi Anda.

7. Pilih Berikutnya: Tanda. Anda tidak perlu menambahkan tanda untuk tujuan latihan ini. Pilih Berikutnya: Tinjauan.
8. Masukkan nama peran, lalu pilih Buat peran.

Buat fungsi Lambda di dalam VPC

1. Buka AWS Lambda konsol di https://console.aws.amazon.com/lambda/.
2. Pilih Buat fungsi.
3. Pilih Penulis dari scratch.
4. Masukkan nama untuk fungsi tersebut, lalu atur Runtime ke `Python 3.9` atau lebih tinggi.
5. Di bagian Izin, perluas Pilih atau buat peran eksekusi.
6. Pada daftar Peran eksekusi, pilih Gunakan peran yang ada.
7. Pada daftar Peran yang ada, pilih peran yang Anda buat sebelumnya.
8. Pilih Buat fungsi.
9. Di bawah Kode fungsi, tempelkan kode berikut.

```
import json
import boto3
import os
from botocore.exceptions import ClientError

def lambda_handler(event, context):
    REGION = os.environ['AWS_REGION']
    client = boto3.client(
        'firehose',
```

```
            REGION

        )
        print("Calling list_delivery_streams with ListDeliveryStreams allowed
policy.")
        delivery_stream_request = client.list_delivery_streams()
        print("Successfully returned list_delivery_streams request %s." % (
            delivery_stream_request
        ))
        describe_access_denied = False
        try:
            print("Calling describe_delivery_stream with DescribeDeliveryStream
denied policy.")
            delivery_stream_info =
client.describe_delivery_stream(DeliveryStreamName='test-describe-denied')
        except ClientError as e:
            error_code = e.response['Error']['Code']
            print ("Caught %s." % (error_code))
            if error_code == 'AccessDeniedException':
                describe_access_denied = True

        if not describe_access_denied:
            raise
        else:
            print("Access denied test succeeded.")
```

10. Di bawah Pengaturan dasar, atur batas waktu menjadi 1 menit.
11. Di bawah Jaringan, pilih VPC tempat Anda membuat titik akhir di atas, lalu pilih subnet dan grup keamanan yang Anda kaitkan dengan titik akhir saat Anda membuatnya.
12. Di dekat bagian atas halaman, pilih Simpan.
13. Pilih Uji.
14. Masukkan nama peristiwa, lalu pilih Buat.
15. Pilih Uji sekali lagi. Hal ini akan menyebabkan fungsi berjalan. Setelah hasil eksekusi muncul, perluas Detail dan bandingkan output log dengan kode fungsi. Hasil yang berhasil menunjukkan daftar aliran Firehose di Wilayah, serta output berikut:

    ```
    Calling describe_delivery_stream.

    AccessDeniedException

    Access denied test succeeded.
    ```

Didukung Wilayah AWS

Titik akhir VPC antarmuka saat ini didukung dalam wilayah berikut.

- AS Timur (Ohio)
- AS Timur (Virginia Utara)
- AS Barat (California Utara)
- AS Barat (Oregon)
- Asia Pasifik (Mumbai)
- Asia Pasifik (Seoul)
- Asia Pasifik (Singapura)
- Asia Pasifik (Sydney)
- Asia Pasifik (Thailand)
- Asia Pasifik (Tokyo)
- Asia Pacific (Hong Kong)
- Kanada (Pusat)
- Kanada Barat (Calgary)
- China (Beijing)
- Tiongkok (Ningxia)
- Eropa (Frankfurt)
- Eropa (Irlandia)
- Eropa (London)
- Eropa (Paris)
- Meksiko (Tengah)
- Amerika Selatan (Sao Paulo)
- AWS GovCloud (US-East)
- AWS GovCloud (US-West)
- Eropa (Spanyol)
- Timur Tengah (UAE)
- Asia Pasifik (Jakarta)
- Asia Pasifik (Osaka)

- Israel (Tel Aviv)
- Asia Pasifik (Malaysia)

Menerapkan praktik terbaik keamanan untuk Amazon Data Firehose

Amazon Data Firehose menyediakan sejumlah fitur keamanan yang perlu 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 sumber daya Amazon Data Firehose mana. 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

Aplikasi produsen dan klien harus memiliki kredensyal yang valid untuk mengakses aliran Firehose, dan aliran Firehose Anda harus memiliki kredensyal yang valid untuk mengakses tujuan. Anda tidak boleh menyimpan AWS kredensil secara langsung di aplikasi klien atau di bucket Amazon S3. Ini adalah kredensial jangka panjang yang tidak dirotasi secara otomatis dan dapat menimbulkan dampak bisnis yang signifikan jika dibobol.

Sebagai gantinya, Anda harus menggunakan peran IAM untuk mengelola kredensi sementara untuk aplikasi produsen dan klien Anda untuk mengakses aliran Firehose. Saat Anda menggunakan peran, Anda tidak perlu menggunakan kredensial jangka panjang (seperti nama pengguna dan kata sandi atau access key) untuk mengakses sumber daya lainnya.

Untuk informasi selengkapnya, lihat topik berikut di Panduan Pengguna IAM:

- Peran IAM
- Skenario Umum untuk Peran: Pengguna, Aplikasi, dan Layanan

Menerapkan enkripsi sisi server dalam sumber daya dependen

Data saat istirahat dan data dalam perjalanan dapat dienkripsi di Amazon Data Firehose. Untuk informasi selengkapnya, lihat Perlindungan data di Amazon Data Firehose.

Gunakan CloudTrail untuk memantau panggilan API

Amazon Data Firehose terintegrasi dengan AWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau AWS layanan di Amazon Data Firehose.

Dengan menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan permintaan yang dibuat untuk Amazon Data Firehose, alamat IP dari mana permintaan dibuat, siapa yang membuat permintaan, kapan dibuat, dan detail tambahan.

Untuk informasi selengkapnya, lihat the section called “Log panggilan Firehose API”.

Pantau Firehose Data Amazon

Anda dapat memantau Amazon Data Firehose menggunakan fitur-fitur berikut:

Topik

Menerapkan praktik terbaik dengan CloudWatch Alarm

Tambahkan CloudWatch alarm ketika metrik berikut melebihi batas buffering (maksimal 15 menit).

- `DeliveryToS3.DataFreshness`
- `DeliveryToIceberg.DataFreshness`
- `DeliveryToSplunk.DataFreshness`
- `DeliveryToAmazonOpenSearchService.DataFreshness`
- `DeliveryToAmazonOpenSearchServerless.DataFreshness`
- `DeliveryToHttpEndpoint.DataFreshness`

Selain itu, buat alarm berdasarkan ekspresi matematika metrik berikut.

- `IncomingBytes (Sum per 5 Minutes) / 300` mendekati persentase `BytesPerSecondLimit`.
- `IncomingRecords (Sum per 5 Minutes) / 300` mendekati persentase `RecordsPerSecondLimit`.
- `IncomingPutRequests (Sum per 5 Minutes) / 300` mendekati persentase `PutRequestsPerSecondLimit`.

Metrik lain yang kami sarankan dipasangi alarm adalah `ThrottledRecords`.

Untuk informasi tentang pemecahan masalah saat alarm berubah status menjadi `ALARM`, lihat Memecahkan masalah kesalahan.

Pantau Amazon Data Firehose dengan metrik CloudWatch

> **Important**
>
> Pastikan untuk mengaktifkan alarm pada semua CloudWatch metrik yang menjadi milik tujuan Anda untuk mengidentifikasi kesalahan tepat waktu.

Amazon Data Firehose terintegrasi dengan CloudWatch metrik Amazon sehingga Anda dapat mengumpulkan, melihat, dan menganalisis CloudWatch metrik untuk aliran Firehose Anda. Misalnya, Anda dapat memantau `IncomingBytes` dan `IncomingRecords` metrik untuk melacak data yang tertelan ke Amazon Data Firehose dari produsen data.

Amazon Data Firehose mengumpulkan dan menerbitkan CloudWatch metrik setiap menit. Namun, jika semburan data yang masuk hanya terjadi selama beberapa detik, data tersebut mungkin tidak sepenuhnya ditangkap atau terlihat dalam metrik satu menit. Ini karena CloudWatch metrik dikumpulkan dari Amazon Data Firehose selama interval satu menit.

Metrik yang dikumpulkan untuk aliran Firehose tidak dikenai biaya. Untuk informasi tentang metrik agen Kinesis, lihat Pantau kesehatan Agen Kinesis.

Topik

- Metrik Penggunaan Firehose Data Amazon

CloudWatch metrik untuk partisi dinamis

Jika partisi dinamis diaktifkan, namespace AWS/Firehose menyertakan metrik berikut.

Metrik	Deskripsi
`ActivePartitionsLimit`	Jumlah maksimum partisi aktif yang diproses oleh aliran Firehose sebelum mengirim data ke bucket kesalahan. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan
`PartitionCount`	Jumlah partisi yang sedang diproses, dengan kata lain, jumlah partisi aktif. Jumlah ini bervariasi antara 1 dan batas jumlah partisi 500 (default). Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan
`PartitionCountExceeded`	Metrik ini menunjukkan jika Anda melebihi batas jumlah partisi. Ini memancarkan 1 atau 0 berdasarkan apakah batas dilanggar atau tidak.
`JQProcessing.Duration`	Mengembalikan jumlah waktu yang dibutuhkan untuk mengeksekusi ekspresi JQ dalam fungsi JQ Lambda. Unit: Milidetik
`PerPartitionThroughput`	Menunjukkan throughput yang sedang diproses per partisi. Metrik ini memungkinkan Anda untuk memantau throughput per partisi. Unit: StandardUnit. BytesSecond

Metrik	Deskripsi
`DeliveryToS3.ObjectCount`	Menunjukkan jumlah objek yang dikirim ke bucket S3 Anda. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan

CloudWatch metrik untuk pengiriman data

Namespace `AWS/Firehose` mencakup metrik tingkat layanan berikut. Jika Anda melihat penurunan kecil pada rata-rata untuk `BackupToS3.Success`, `DeliveryToS3.Success`, `DeliveryToSplunk.Success`, `DeliveryToAmazonOpenSearchService.Success`, atau `DeliveryToRedshift.Success`, hal itu tidak menunjukkan adanya kehilangan data. Amazon Data Firehose mencoba ulang kesalahan pengiriman dan tidak melanjutkan hingga catatan berhasil dikirim ke tujuan yang dikonfigurasi atau ke bucket S3 cadangan.

Topik

- Pengiriman ke OpenSearch Layanan
- Pengiriman ke Tanpa OpenSearch Server
- Pengiriman ke Amazon Redshift
- Pengiriman ke Amazon S3
- Pengiriman ke Snowflake
- Pengiriman ke Splunk
- Pengiriman ke Titik Akhir HTTP

Pengiriman ke OpenSearch Layanan

Metrik	Deskripsi
`DeliveryToAmazonOpenSearchService.Bytes`	Jumlah byte yang diindeks ke OpenSearch Layanan selama periode waktu yang ditentukan.

Metrik	Deskripsi
	Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Byte
`DeliveryToAmazonOpenSearchService.DataFreshness`	Usia (dari masuk ke Amazon Data Firehose hingga sekarang) dari rekor tertua di Amazon Data Firehose. Setiap catatan yang lebih tua dari usia ini telah dikirimkan ke OpenSearch Layanan. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: detik
`DeliveryToAmazonOpenSearchService.Records`	Jumlah catatan yang diindeks ke OpenSearch Layanan selama periode waktu yang ditentukan. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan
`DeliveryToAmazonOpenSearchService.Success`	Jumlah catatan yang berhasil diindeks.
`DeliveryToS3.Bytes`	Jumlah byte yang dikirim ke Amazon S3 selama periode waktu yang ditentukan. Amazon Data Firehose memancarkan metrik ini hanya jika Anda mengaktifkan pencadangan untuk semua dokumen. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan

Metrik	Deskripsi
DeliveryToS3.DataFreshness	Usia (dari masuk ke Amazon Data Firehose hingga sekarang) dari rekor tertua di Amazon Data Firehose. Semua catatan yang lebih lama dari usia ini telah dikirim ke bucket S3. Amazon Data Firehose memancarkan metrik ini hanya jika Anda mengaktifkan pencadangan untuk semua dokumen. Unit: detik
DeliveryToS3.Records	Jumlah catatan yang dikirim ke Amazon S3 selama periode waktu yang ditentukan. Amazon Data Firehose memancarkan metrik ini hanya jika Anda mengaktifkan pencadangan untuk semua dokumen. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan
DeliveryToS3.Success	Jumlah perintah Amazon S3 yang berhasil menempatkan. Amazon Data Firehose selalu memancarkan metrik ini terlepas dari apakah cadangan diaktifkan hanya untuk dokumen yang gagal atau untuk semua dokumen.
DeliveryToAmazonOpenSearchService.AuthFailure	Authentication/authorization error. Verify the OS/ESkebijakan klaster dan izin peran. 0 menunjukkan bahwa tidak ada masalah. 1 menunjukkan kegagalan otentikasi.
DeliveryToAmazonOpenSearchService.DeliveryRejected	Kesalahan pengiriman ditolak. Verifikasi kebijakan OS/ES klaster dan izin peran. 0 menunjukkan bahwa tidak ada masalah. 1 menunjukkan bahwa ada kegagalan pengiriman.

Pengiriman ke Tanpa OpenSearch Server

Metrik	Deskripsi
`DeliveryToAmazonOpenSearchServerless.Bytes`	Jumlah byte yang diindeks ke OpenSearch Tanpa Server selama periode waktu yang ditentukan. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Byte
`DeliveryToAmazonOpenSearchServerless.DataFreshness`	Usia (dari masuk ke Amazon Data Firehose hingga sekarang) dari rekor tertua di Amazon Data Firehose. Setiap catatan yang lebih tua dari usia ini telah dikirim ke Tanpa OpenSearch Server. Unit: detik
`DeliveryToAmazonOpenSearchServerless.Records`	Jumlah catatan yang diindeks ke OpenSearch Tanpa Server selama periode waktu yang ditentukan. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan
`DeliveryToAmazonOpenSearchServerless.Success`	Jumlah catatan yang berhasil diindeks.
`DeliveryToS3.Bytes`	Jumlah byte yang dikirim ke Amazon S3 selama periode waktu yang ditentukan. Amazon Data Firehose memancarkan metrik ini hanya jika Anda mengaktifkan pencadangan untuk semua dokumen. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan

Metrik	Deskripsi
DeliveryToS3.DataFreshness	Usia (dari masuk ke Amazon Data Firehose hingga sekarang) dari rekor tertua di Amazon Data Firehose. Semua catatan yang lebih lama dari usia ini telah dikirim ke bucket S3. Amazon Data Firehose memancarkan metrik ini hanya jika Anda mengaktifkan pencadangan untuk semua dokumen. Unit: detik
DeliveryToS3.Records	Jumlah catatan yang dikirim ke Amazon S3 selama periode waktu yang ditentukan. Amazon Data Firehose memancarkan metrik ini hanya jika Anda mengaktifkan pencadangan untuk semua dokumen. Unit: Hitungan
DeliveryToS3.Success	Jumlah perintah Amazon S3 yang berhasil menempatkan. Amazon Data Firehose selalu memancarkan metrik ini terlepas dari apakah cadangan diaktifkan hanya untuk dokumen yang gagal atau untuk semua dokumen.
DeliveryToAmazonOpenSearchServerless.AuthFailure	Authentication/authorization error. Verify the OS/ESkebijakan klaster dan izin peran. 0 menunjukkan bahwa tidak ada masalah. 1 menunjukkan bahwa ada kegagalan otentikasi.
DeliveryToAmazonOpenSearchServerless.DeliveryRejected	Kesalahan pengiriman ditolak. Verifikasi kebijakan OS/ES klaster dan izin peran. 0 menunjukkan bahwa tidak ada masalah. 1 menunjukkan bahwa ada kegagalan pengiriman.

Pengiriman ke Amazon Redshift

Metrik	Deskripsi
DeliveryToRedshift.Bytes	Jumlah byte yang disalin ke Amazon Redshift selama periode waktu yang ditentukan. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan
DeliveryToRedshift.Records	Jumlah catatan yang disalin ke Amazon Redshift selama periode waktu yang ditentukan. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan
DeliveryToRedshift.Success	Jumlah perintah Amazon Redshift COPY yang berhasil.
DeliveryToS3.Bytes	Jumlah byte yang dikirim ke Amazon S3 selama periode waktu yang ditentukan. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Byte
DeliveryToS3.DataFreshness	Usia (dari masuk ke Amazon Data Firehose hingga sekarang) dari rekor tertua di Amazon Data Firehose. Catatan apa pun yang lebih tua dari usia ini dikirim ke ember S3. Unit: detik
DeliveryToS3.Records	Jumlah catatan yang dikirim ke Amazon S3 selama periode waktu yang ditentukan.

Metrik	Deskripsi
	Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan
`DeliveryToS3.Success`	Jumlah perintah Amazon S3 yang berhasil menempatkan.
`DeliveryToRedshift.DataFreshness`	Usia (dari masuk ke Amazon Data Firehose hingga sekarang) dari rekor tertua di Amazon Data Firehose. Catatan apa pun yang lebih tua dari usia ini dikirim ke cluster Amazon Redshift.
`BackupToS3.Bytes`	Jumlah byte yang dikirimkan ke Amazon S3 untuk dicadangkan selama periode waktu yang ditentukan. Amazon Data Firehose memancarkan metrik ini saat cadangan ke Amazon S3 diaktifkan. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan
`BackupToS3.DataFreshness`	Usia (dari masuk ke Amazon Data Firehose hingga sekarang) dari rekor tertua di Amazon Data Firehose. Semua catatan yang lebih lama dari usia ini telah dikirim ke bucket Amazon S3 untuk dicadangkan. Amazon Data Firehose memancarkan metrik ini saat cadangan ke Amazon S3 diaktifkan. Statistik: Minimum, Maksimum, Rata-rata, Sampel Unit: detik

Metrik	Deskripsi
BackupToS3.Records	Jumlah catatan yang dikirim ke Amazon S3 untuk dicadangkan selama periode waktu yang ditentukan. Amazon Data Firehose memancarkan metrik ini saat cadangan ke Amazon S3 diaktifkan. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan
BackupToS3.Success	Jumlah Amazon S3 yang berhasil menempatkan perintah untuk cadangan. Amazon Data Firehose memancarkan metrik ini saat cadangan ke Amazon S3 diaktifkan.

Pengiriman ke Amazon S3

Metrik dalam tabel berikut terkait dengan pengiriman ke Amazon S3 ketika itu adalah tujuan utama aliran Firehose.

Metrik	Deskripsi
DeliveryToS3.Bytes	Jumlah byte yang dikirim ke Amazon S3 selama periode waktu yang ditentukan. Saat transformasi data diaktifkan, metrik ini mencerminkan ukuran byte yang telah diproses sebelumnya sebelum transformasi. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Byte
DeliveryToS3.DataFreshness	Usia (dari masuk ke Amazon Data Firehose hingga sekarang) dari rekor tertua di Amazon Data Firehose. Semua catatan yang lebih lama dari usia ini telah dikirim ke bucket S3.

Metrik	Deskripsi
	Statistik: Minimum, Maksimum, Rata-rata, Sampel Unit: detik
`DeliveryToS3.Records`	Jumlah catatan yang dikirim ke Amazon S3 selama periode waktu yang ditentukan. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan
`DeliveryToS3.Success`	Jumlah perintah Amazon S3 yang berhasil menempatkan.
`BackupToS3.Bytes`	Jumlah byte yang dikirimkan ke Amazon S3 untuk dicadangkan selama periode waktu yang ditentukan. Amazon Data Firehose memancarkan metrik ini saat pencadangan diaktifkan (yang hanya dimungkinkan ketika transformasi data juga diaktifkan). Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan
`BackupToS3.DataFreshness`	Usia (dari masuk ke Amazon Data Firehose hingga sekarang) dari rekor tertua di Amazon Data Firehose. Semua catatan yang lebih lama dari usia ini telah dikirim ke bucket Amazon S3 untuk dicadangkan. Amazon Data Firehose memancarkan metrik ini saat pencadangan diaktifkan (yang hanya dimungkinkan ketika transformasi data juga diaktifkan). Statistik: Minimum, Maksimum, Rata-rata, Sampel Unit: detik

Metrik	Deskripsi
BackupToS3.Records	Jumlah catatan yang dikirim ke Amazon S3 untuk dicadangkan selama periode waktu yang ditentukan. Amazon Data Firehose memancarkan metrik ini saat pencadangan diaktifkan (yang hanya dimungkinkan ketika transformasi data juga diaktifkan). Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan
BackupToS3.Success	Jumlah Amazon S3 yang berhasil menempatkan perintah untuk cadangan. Amazon Data Firehose memancark an metrik ini saat pencadangan diaktifkan (yang hanya dimungkinkan ketika transformasi data juga diaktifkan).

Pengiriman ke Snowflake

Metrik	Deskripsi
DeliveryToSnowflake.Bytes	Jumlah byte yang dikirim ke Snowflake selama periode waktu yang ditentukan. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Byte
DeliveryToSnowflak e.DataFreshness	Usia (dari masuk ke Firehose hingga sekarang) dari rekor tertua di Firehose. Setiap catatan yang lebih tua dari usia ini telah dikirim ke Snowflake. Perhatika n bahwa diperlukan beberapa detik untuk mengkomit data ke Snowflake setelah panggilan penyisipan Firehose berhasil. Untuk waktu yang diperlukan untuk memasukkan data ke Snowflake, lihat metrik. DeliveryToSnowflake.DataCommitLatency

Metrik	Deskripsi
	Statistik: Minimum, Maksimum, Rata-rata, Sampel Unit: detik
`DeliveryToSnowflake.DataCommitLatency`	Waktu yang dibutuhkan data untuk berkomitmen ke Snowflake setelah Firehose berhasil memasukkan catatan. Statistik: Minimum, Maksimum, Rata-rata, Sampel Unit: detik
`DeliveryToSnowflake.Records`	Jumlah catatan yang dikirim ke Snowflake selama periode waktu yang ditentukan. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan
`DeliveryToSnowflake.Success`	Jumlah panggilan insert yang berhasil dilakukan ke Snowflake.
`DeliveryToS3.Bytes`	Jumlah byte yang dikirim ke Amazon S3 selama periode waktu yang ditentukan. Metrik ini hanya tersedia saat pengiriman ke Snowflake gagal dan Firehose mencoba mencadangkan data yang gagal ke S3. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Byte

Metrik	Deskripsi
DeliveryToS3.Records	Jumlah catatan yang dikirim ke Amazon S3 selama periode waktu yang ditentukan. Metrik ini hanya tersedia saat pengiriman ke Snowflake gagal dan Firehose mencoba mencadangkan data yang gagal ke S3. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan
DeliveryToS3.Success	Jumlah perintah Amazon S3 yang berhasil menempatkan. Metrik ini hanya tersedia saat pengiriman ke Snowflake gagal dan Firehose mencoba mencadangkan data yang gagal ke S3.
BackupToS3.DataFreshness	Usia (dari ke Firehose hingga sekarang) dari rekor tertua di Firehose. Catatan apa pun yang lebih tua dari usia ini didukung ke ember Amazon S3. Metrik ini tersedia saat aliran Firehose dikonfigurasi untuk mencadangkan semua data. Statistik: Minimum, Maksimum, Rata-rata, Sampel Unit: detik
BackupToS3.Records	Jumlah catatan yang dikirim ke Amazon S3 untuk dicadangkan selama periode waktu yang ditentukan. Metrik ini tersedia saat aliran Firehose dikonfigurasi untuk mencadangkan semua data. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit:Hitung

Metrik	Deskripsi
BackupToS3.Bytes	Jumlah byte yang dikirimkan ke Amazon S3 untuk dicadangkan selama periode waktu yang ditentukan. Metrik ini tersedia saat aliran Firehose dikonfigurasi untuk mencadangkan semua data. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit:Hitung
BackupToS3.Success	Jumlah Amazon S3 yang berhasil menempatkan perintah untuk cadangan. Firehose memancarkan metrik ini saat aliran Firehose dikonfigurasi untuk mencadangkan semua data.

Pengiriman ke Splunk

Metrik	Deskripsi
DeliveryToSplunk.Bytes	Jumlah byte yang dikirim ke Splunk selama periode waktu yang ditentukan. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Byte
DeliveryToSplunk.DataAckLatency	Perkiraan durasi yang diperlukan untuk menerima pengakuan dari Splunk setelah Amazon Data Firehose mengirimkan data. Tren yang meningkat atau menurun untuk metrik ini lebih berguna daripada nilai perkiraan absolut. Tren yang meningkat dapat mengindikasikan laju pengindeksan dan pengakuan yang lebih lambat dari pengindeks Splunk. Statistik: Minimum, Maksimum, Rata-rata, Sampel

Metrik	Deskripsi
	Unit: detik
DeliveryToSplunk.DataFreshness	Usia (dari masuk ke Amazon Data Firehose hingga sekarang) dari rekor tertua di Amazon Data Firehose. Semua catatan yang lebih lama dari usia ini telah dikirim ke Splunk. Statistik: Minimum, Maksimum, Rata-rata, Sampel Unit: detik
DeliveryToSplunk.Records	Jumlah catatan yang dikirim ke Splunk selama periode waktu yang ditentukan. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan
DeliveryToSplunk.Success	Jumlah catatan yang berhasil diindeks.
DeliveryToS3.Success	Jumlah perintah Amazon S3 yang berhasil menempatkan. Metrik ini dikeluarkan saat pencadangan ke Amazon S3 diaktifkan.
BackupToS3.Bytes	Jumlah byte yang dikirimkan ke Amazon S3 untuk dicadangkan selama periode waktu yang ditentukan. Amazon Data Firehose memancarkan metrik ini saat aliran Firehose dikonfigurasi untuk mencadangkan semua dokumen. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan

Metrik	Deskripsi
BackupToS3.DataFreshness	Usia (dari masuk ke Amazon Data Firehose hingga sekarang) dari rekor tertua di Amazon Data Firehose. Semua catatan yang lebih lama dari usia ini telah dikirim ke bucket Amazon S3 untuk dicadangkan. Amazon Data Firehose memancarkan metrik ini saat aliran Firehose dikonfigurasi untuk mencadangkan semua dokumen. Statistik: Minimum, Maksimum, Rata-rata, Sampel Unit: detik
BackupToS3.Records	Jumlah catatan yang dikirim ke Amazon S3 untuk dicadangkan selama periode waktu yang ditentukan. Amazon Data Firehose memancarkan metrik ini saat aliran Firehose dikonfigurasi untuk mencadangkan semua dokumen. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan
BackupToS3.Success	Jumlah perintah Amazon S3 yang berhasil dimasukkan untuk cadangan. Amazon Data Firehose memancarkan metrik ini saat aliran Firehose dikonfigurasi untuk mencadangkan semua dokumen.

Pengiriman ke Titik Akhir HTTP

Metrik	Deskripsi
DeliveryToHttpEndpoint.Bytes	Jumlah byte yang berhasil dikirim ke titik akhir HTTP. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel

Metrik	Deskripsi
	Unit: Byte
DeliveryToHttpEndpoint.Records	Jumlah catatan yang berhasil dikirim ke titik akhir HTTP. Metrik ini hanya dipancarkan untuk upaya pengiriman yang berhasil dan tidak dipancarkan ketika upaya pengiriman gagal. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Jumlah
DeliveryToHttpEndpoint.DataFreshness	Usia rekor tertua di Amazon Data Firehose. Statistik: Minimum, Maksimum, Rata-rata, Sampel Unit: detik
DeliveryToHttpEndpoint.Success	Jumlah catatan yang berhasil dikirim ke titik akhir HTTP per upaya pengiriman. Tidak sepertiDeliveryToHttpEndpoint.Records , metrik ini dipancarkan untuk setiap upaya pengiriman. Jika berhasil, nilainya sama dengan jumlah catatan dalam upaya pengiriman. Pada kegagalan semua catatan dalam upaya pengiriman, nilainya adalah 0. Gunakan statistik Minimum untuk memantau kegagalan pengiriman. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan
DeliveryToHttpEndpoint.ProcessedBytes	Jumlah percobaan byte diproses, termasuk percobaan ulang.
DeliveryToHttpEndpoint.ProcessedRecords	Jumlah percobaan catatan termasuk percobaan ulang.

Metrik konsumsi data

Topik

Konsumsi data melalui Kinesis Data Streams

Metrik	Deskripsi
`DataReadFromKinesisStream.Bytes`	Bila sumber data adalah aliran data Kinesis, metrik ini menunjukkan jumlah byte yang dibaca dari aliran data tersebut. Jumlah ini termasuk pembacaan ulang karena failover. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Byte
`DataReadFromKinesisStream.Records`	Bila sumber data adalah aliran data Kinesis, metrik ini menunjukkan jumlah catatan yang dibaca dari aliran data tersebut. Jumlah ini termasuk pembacaan ulang karena failover. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan
`ThrottledDescribeStream`	Frekuensi total operasi `DescribeStream` di-throttling saat sumber data merupakan aliran data Kinesis. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan

Metrik	Deskripsi
ThrottledGetRecords	Frekuensi total operasi GetRecords di-throttling saat sumber data merupakan aliran data Kinesis. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan
ThrottledGetShardIterator	Frekuensi total operasi GetShardIterator di-throttling saat sumber data merupakan aliran data Kinesis. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan
KinesisMillisBehindLatest	Bila sumber data adalah aliran data Kinesis, metrik ini menunjukkan bahwa jumlah milidetik yang terakhir dicatat dalam pembacaan tertinggal dari catatan terbaru dalam aliran data Kinesis. Statistik: Minimum, Maksimum, Rata-rata, Sampel Unit: Milidetik

Konsumsi data melalui PUT Langsung

Metrik	Deskripsi
BackupToS3.Bytes	Jumlah byte yang dikirimkan ke Amazon S3 untuk dicadangkan selama periode waktu yang ditentukan. Amazon Data Firehose memancarkan metrik ini saat transformasi data diaktifkan untuk tujuan Amazon S3 atau Amazon Redshift. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel

Metrik	Deskripsi
	Unit: Byte
`BackupToS3.DataFreshness`	Usia (dari masuk ke Amazon Data Firehose hingga sekarang) dari rekor tertua di Amazon Data Firehose. Semua catatan yang lebih lama dari usia ini telah dikirim ke bucket Amazon S3 untuk dicadangkan. Amazon Data Firehose memancarkan metrik ini saat transformasi data diaktifkan untuk tujuan Amazon S3 atau Amazon Redshift. Statistik: Minimum, Maksimum, Rata-rata, Sampel Unit: detik
`BackupToS3.Records`	Jumlah catatan yang dikirim ke Amazon S3 untuk dicadangkan selama periode waktu yang ditentukan. Amazon Data Firehose memancarkan metrik ini saat transformasi data diaktifkan untuk tujuan Amazon S3 atau Amazon Redshift. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan
`BackupToS3.Success`	Jumlah perintah Amazon S3 yang berhasil dimasukkan untuk cadangan. Amazon Data Firehose memancarkan metrik ini saat transformasi data diaktifkan untuk tujuan Amazon S3 atau Amazon Redshift.
`BytesPerSecondLimit`	Jumlah maksimum byte per detik saat ini yang dapat diserap oleh aliran Firehose sebelum throttling. Untuk meminta peningkatan batas ini, kunjungi Pusat Dukungan AWS dan pilih Buat kasus, lalu pilih Peningkatan batas layanan.

Metrik	Deskripsi
DeliveryToAmazonOpenSearchService.Bytes	Jumlah byte yang diindeks ke OpenSearch Layanan selama periode waktu yang ditentukan. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Byte
DeliveryToAmazonOpenSearchService.DataFreshness	Usia (dari masuk ke Amazon Data Firehose hingga sekarang) dari rekor tertua di Amazon Data Firehose. Setiap catatan yang lebih tua dari usia ini telah dikirimkan ke OpenSearch Layanan. Statistik: Minimum, Maksimum, Rata-rata, Sampel Unit: detik
DeliveryToAmazonOpenSearchService.Records	Jumlah catatan yang diindeks ke OpenSearch Layanan selama periode waktu yang ditentukan. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan
DeliveryToAmazonOpenSearchService.Success	Jumlah catatan yang berhasil diindeks.
DeliveryToRedshift.Bytes	Jumlah byte yang disalin ke Amazon Redshift selama periode waktu yang ditentukan. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Byte

Metrik	Deskripsi
DeliveryToRedshift.Records	Jumlah catatan yang disalin ke Amazon Redshift selama periode waktu yang ditentukan. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan
DeliveryToRedshift.Success	Jumlah perintah Amazon Redshift COPY yang berhasil.
DeliveryToS3.Bytes	Jumlah byte yang dikirim ke Amazon S3 selama periode waktu yang ditentukan. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Byte
DeliveryToS3.DataFreshness	Usia (dari masuk ke Amazon Data Firehose hingga sekarang) dari rekor tertua di Amazon Data Firehose. Semua catatan yang lebih lama dari usia ini telah dikirim ke bucket S3. Statistik: Minimum, Maksimum, Rata-rata, Sampel Unit: detik
DeliveryToS3.Records	Jumlah catatan yang dikirim ke Amazon S3 selama periode waktu yang ditentukan. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan
DeliveryToS3.Success	Jumlah perintah Amazon S3 yang berhasil menempatkan.

Metrik	Deskripsi
DeliveryToSplunk.Bytes	Jumlah byte yang dikirim ke Splunk selama periode waktu yang ditentukan. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Byte
DeliveryToSplunk.DataAckLatency	Perkiraan durasi yang diperlukan untuk menerima pengakuan dari Splunk setelah Amazon Data Firehose mengirimkan data. Tren yang meningkat atau menurun untuk metrik ini lebih berguna daripada nilai perkiraan absolut. Tren yang meningkat dapat mengindikasikan laju pengindeksan dan pengakuan yang lebih lambat dari pengindeks Splunk. Statistik: Minimum, Maksimum, Rata-rata, Sampel Unit: detik
DeliveryToSplunk.DataFreshness	Usia (dari masuk ke Amazon Data Firehose hingga sekarang) dari rekor tertua di Amazon Data Firehose. Semua catatan yang lebih lama dari usia ini telah dikirim ke Splunk. Statistik: Minimum, Maksimum, Rata-rata, Sampel Unit: detik
DeliveryToSplunk.Records	Jumlah catatan yang dikirim ke Splunk selama periode waktu yang ditentukan. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan
DeliveryToSplunk.Success	Jumlah catatan yang berhasil diindeks.

Metrik	Deskripsi
IncomingBytes	Jumlah byte yang berhasil tertelan ke aliran Firehose selama periode waktu yang ditentukan. Konsumsi data dapat dibatasi ketika melebihi salah satu batas aliran Firehose. Data yang dibatasi tidak akan dihitung. IncomingBytes Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Byte
IncomingPutRequests	Jumlah yang berhasil PutRecord dan PutRecordBatch permintaan selama periode waktu tertentu. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan
IncomingRecords	Jumlah catatan yang berhasil tertelan ke aliran Firehose selama periode waktu yang ditentukan. Konsumsi data dapat dibatasi ketika melebihi salah satu batas aliran Firehose. Data yang dibatasi tidak akan dihitung. IncomingRecords Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan
RecordsPerSecondLimit	Jumlah maksimum rekaman per detik saat ini yang dapat diserap oleh aliran Firehose sebelum throttling. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan

Metrik	Deskripsi
ThrottledRecords	Jumlah catatan yang dibatasi karena konsumsi data melebihi salah satu batas aliran Firehose. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan

Konsumsi data dari MSK

Metrik	Deskripsi
DataReadFromSource .Records	Jumlah catatan dibaca dari sumber Kafka Topic. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan
DataReadFromSource.Bytes	Jumlah byte dibaca dari sumber Kafka Topic. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Byte
SourceThrottled.Delay	Jumlah waktu sumber Kafka cluster tertunda dalam mengembalikan catatan dari sumber Kafka Topic. Statistik: Minimum, Maksimum, Rata-rata, Sampel Unit: Milidetik
BytesPerSecondLimit	Batas throughput saat ini di mana Firehose akan membaca dari setiap partisi sumber Kafka Topic. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel

Metrik	Deskripsi
	Unit: Byte/detik
`KafkaOffsetLag`	Perbedaan antara offset terbesar dari catatan yang telah dibaca Firehose dari sumber Kafka Topic dan offset terbesar dari catatan yang tersedia dari sumber Kafka Topic. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan
`FailedValidation.Records`	Jumlah catatan yang gagal validasi rekaman. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan
`FailedValidation.Bytes`	Jumlah byte yang gagal validasi rekaman. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Byte
`DataReadFromSource .Backpressured`	Menunjukkan bahwa aliran Firehose tertunda dalam membaca catatan dari partisi sumber baik karena BytesPerSecondLimit per partisi telah terlampaui atau bahwa aliran normal pengiriman lambat atau telah berhenti Unit: Boolean

Metrik tingkat API CloudWatch

Namespace `AWS/Firehose` mencakup metrik tingkat API berikut.

Metrik	Deskripsi
DescribeDeliveryStream.Latency	Waktu yang dibutuhkan per operasi DescribeDeliveryStream , diukur selama periode waktu yang ditentukan. Statistik: Minimum, Maksimum, Rata-rata, Sampel Unit: Milidetik
DescribeDeliveryStream.Requests	Jumlah total permintaan DescribeDeliveryStream . Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan
ListDeliveryStreams.Latency	Waktu yang dibutuhkan per operasi ListDeliveryStream , diukur selama periode waktu yang ditentukan. Statistik: Minimum, Maksimum, Rata-rata, Sampel Unit: Milidetik
ListDeliveryStreams.Requests	Jumlah total permintaan ListFirehose . Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan
PutRecord.Bytes	Jumlah byte yang dimasukkan ke aliran Firehose PutRecord menggunakan selama periode waktu yang ditentukan. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Byte

Metrik	Deskripsi
PutRecord.Latency	Waktu yang dibutuhkan per operasi PutRecord , diukur selama periode waktu yang ditentukan. Statistik: Minimum, Maksimum, Rata-rata, Sampel Unit: Milidetik
PutRecord.Requests	Jumlah total permintaan PutRecord , yang sama dengan jumlah total catatan dari operasi PutRecord . Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan
PutRecordBatch.Bytes	Jumlah byte yang dimasukkan ke aliran Firehose PutRecordBatch menggunakan selama periode waktu yang ditentukan. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Byte
PutRecordBatch.Latency	Waktu yang dibutuhkan per operasi PutRecordBatch , diukur selama periode waktu yang ditentukan. Statistik: Minimum, Maksimum, Rata-rata, Sampel Unit: Milidetik
PutRecordBatch.Records	Jumlah total catatan dari operasi PutRecordBatch . Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan

Metrik	Deskripsi
PutRecordBatch.Requests	Jumlah total permintaan PutRecordBatch . Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan
PutRequestsPerSecondLimit	Jumlah maksimum permintaan put per detik yang dapat ditangani oleh aliran Firehose sebelum throttling. Nomor ini termasuk PutRecord dan PutRecordBatch permintaan. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan
ThrottledDescribeStream	Frekuensi total operasi DescribeStream di-throttling saat sumber data merupakan aliran data Kinesis. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan
ThrottledGetRecords	Frekuensi total operasi GetRecords di-throttling saat sumber data merupakan aliran data Kinesis. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan
ThrottledGetShardIterator	Frekuensi total operasi GetShardIterator di-throtting saat sumber data merupakan aliran data Kinesis. Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan

Metrik	Deskripsi
`UpdateDeliveryStream.Latency`	Waktu yang dibutuhkan per operasi `UpdateDeliveryStream` , diukur selama periode waktu yang ditentukan. Statistik: Minimum, Maksimum, Rata-rata, Sampel Unit: Milidetik
`UpdateDeliveryStream.Requests`	Jumlah total permintaan `UpdateDeliveryStream` . Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan

CloudWatch Metrik Transformasi Data

Jika transformasi data dengan Lambda diaktifkan, namespace `AWS/Firehose` mencakup metrik berikut.

Metrik	Deskripsi
`ExecuteProcessing.Duration`	Waktu yang dibutuhkan untuk setiap pemanggilan fungsi Lambda yang dilakukan oleh Firehose. Unit: Milidetik
`ExecuteProcessing.Success`	Jumlah pemanggilan fungsi Lambda yang berhasil dari jumlah total pemanggilan fungsi Lambda.
`SucceedProcessing.Records`	Jumlah catatan yang berhasil diproses selama periode waktu yang ditentukan. Unit: Hitungan

Metrik	Deskripsi
`SucceedProcessing.Bytes`	Jumlah byte yang berhasil diproses selama periode waktu yang ditentukan. Unit: Byte

CloudWatch Metrik Dekompresi Log

Jika dekompresi diaktifkan untuk pengiriman CloudWatch Log, `AWS/Firehose` namespace menyertakan metrik berikut.

Metrik	Deskripsi
`OutputDecompressedBytes.Success`	Data dekompresi yang berhasil dalam byte Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Byte
`OutputDecompressedBytes.Failed`	Gagal mendekompresi data dalam byte Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Byte
`OutputDecompressedRecords.Success`	Jumlah catatan dekompresi yang berhasil Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan
`OutputDecompressedRecords.Failed`	Jumlah catatan dekompresi yang gagal Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel

Metrik	Deskripsi
	Unit: Hitungan

Format CloudWatch Metrik Konversi

Jika konversi format diaktifkan, namespace `AWS/Firehose` mencakup metrik berikut.

Metrik	Deskripsi
`SucceedConversion.Records`	Jumlah catatan yang berhasil dikonversi. Unit: Hitungan
`SucceedConversion.Bytes`	Jumlah catatan yang berhasil dikonversi. Unit: Byte
`FailedConversion.Records`	Jumlah catatan yang tidak dapat dikonversi. Unit: Hitungan
`FailedConversion.Bytes`	Ukuran catatan yang tidak dapat dikonversi. Unit: Byte

Metrik Enkripsi Sisi Server (SSE) CloudWatch

Namespace `AWS/Firehose` mencakup metrik berikut yang terkait dengan SSE.

Metrik	Deskripsi
`KMSKeyAccessDenied`	Berapa kali layanan menemukan aliran Firehose. `KMSAccessDeniedException` Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel

Metrik	Deskripsi
	Unit: Hitungan
`KMSKeyDisabled`	Berapa kali layanan menemukan aliran Firehose. `KMSDisabledException` Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan
`KMSKeyInvalidState`	Berapa kali layanan menemukan aliran Firehose. `KMSInvalidStateException` Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan
`KMSKeyNotFound`	Berapa kali layanan menemukan aliran Firehose. `KMSNotFoundException` Statistik: Minimum, Maksimum, Rata-rata, Jumlah, Sampel Unit: Hitungan

Dimensi untuk Amazon Data Firehose

Untuk memfilter metrik menurut aliran Firehose, gunakan `DeliveryStreamName` dimensi.

Metrik Penggunaan Firehose Data Amazon

Anda dapat menggunakan metrik CloudWatch penggunaan untuk memberikan visibilitas ke dalam penggunaan sumber daya akun Anda. Gunakan metrik ini untuk memvisualisasikan penggunaan layanan Anda saat ini pada CloudWatch grafik dan dasbor.

Metrik penggunaan kuota layanan ada di namespace AWS/Usage dan dikumpulkan setiap tiga menit.

Saat ini, satu-satunya nama metrik di namespace ini yang CloudWatch diterbitkan adalah. `ResourceCount` Metrik ini diterbitkan dengan dimensi `Service`, `Class`, `Type`, dan `Resource`.

Metrik	Deskripsi
`ResourceCount`	Jumlah sumber daya yang ditentukan yang berjalan di akun Anda. Sumber daya tersebut ditentukan oleh dimensi yang dikaitkan dengan metrik. Statistik yang paling berguna untuk metrik ini adalah MAXIMUM, yang mewakili jumlah maksimum sumber daya yang digunakan selama periode 3 menit.

Dimensi berikut digunakan untuk menyempurnakan metrik penggunaan yang diterbitkan oleh Amazon Data Firehose.

Dimensi	Deskripsi
`Service`	Nama AWS layanan yang berisi sumber daya. Untuk metrik penggunaan Amazon Data Firehose, nilai untuk dimensi ini adalah. `Firehose`
`Class`	Kelas sumber daya yang akan dilacak. Metrik penggunaan Amazon Data Firehose API menggunakan dimensi ini dengan nilai. `None`
`Type`	Jenis sumber daya yang sedang ditelusuri. Saat ini, ketika dimensi Layanan adalah `Firehose`, satu-satunya nilai yang valid untuk Tipe adalah `Resource`.
`Resource`	Nama sumber AWS daya. Saat ini, ketika dimensi Layanan adalah `Firehose`, satu-satunya nilai yang valid untuk Sumber Daya adalah `DeliveryStreams` .

Akses CloudWatch Metrik untuk Amazon Data Firehose

Anda dapat memantau metrik Amazon Data Firehose menggunakan CloudWatch konsol, baris perintah, atau API. CloudWatch Prosedur berikut menunjukkan cara mengakses metrik menggunakan berbagai metode ini.

Untuk mengakses metrik menggunakan konsol CloudWatch

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada bilah navigasi, pilih Wilayah.
3. Pada panel navigasi, silakan pilih Metrik.
4. Pilih namespace Firehose.
5. Pilih Metrik aliran Firehose atau Metrik Firehose.
6. Pilih metrik yang akan ditambahkan ke grafik.

Untuk mengakses metrik menggunakan AWS CLI

Gunakan daftar-metrik dan perintah. get-metric-statistics

```
aws cloudwatch list-metrics --namespace "AWS/Firehose"
```

```
aws cloudwatch get-metric-statistics --namespace "AWS/Firehose" \
--metric-name DescribeDeliveryStream.Latency --statistics Average --period 3600 \
--start-time 2017-06-01T00:00:00Z --end-time 2017-06-30T00:00:00Z
```

Pantau Amazon Data Firehose Menggunakan Log CloudWatch

Amazon Data Firehose terintegrasi dengan Amazon CloudWatch Logs sehingga Anda dapat melihat log kesalahan tertentu saat pemanggilan Lambda untuk transformasi data atau pengiriman data gagal. Anda dapat mengaktifkan pencatatan kesalahan Amazon Data Firehose saat membuat aliran Firehose.

Jika Anda mengaktifkan pencatatan kesalahan Amazon Data Firehose di konsol Amazon Data Firehose, grup log dan aliran log terkait akan dibuat untuk aliran Firehose atas nama Anda. Format nama grup log adalah`/aws/kinesisfirehose/`*`delivery-stream-name`*, di mana *`delivery-stream-name`* nama aliran Firehose yang sesuai. `DestinationDelivery`adalah aliran log yang

dibuat dan digunakan untuk mencatat kesalahan apa pun yang terkait dengan pengiriman ke tujuan utama. Aliran log lain `BackupDelivery` yang disebut dibuat hanya jika cadangan S3 diaktifkan untuk tujuan. Aliran `BackupDelivery` log digunakan untuk mencatat kesalahan apa pun yang terkait dengan pengiriman ke cadangan S3.

Misalnya, jika Anda membuat aliran Firehose "MyStream" dengan Amazon Redshift sebagai tujuan dan mengaktifkan pencatatan kesalahan Amazon Data Firehose, berikut ini dibuat atas nama Anda: grup log bernama dan dua aliran log `aws/kinesisfirehose/MyStream` bernama dan. `DestinationDelivery` BackupDelivery Dalam contoh ini, `DestinationDelivery` akan digunakan untuk mencatat kesalahan apa pun yang terkait dengan pengiriman ke tujuan Amazon Redshift dan juga ke tujuan S3 menengah. `BackupDelivery`, jika cadangan S3 diaktifkan, akan digunakan untuk mencatat kesalahan apa pun yang terkait dengan pengiriman ke bucket cadangan S3.

Anda dapat mengaktifkan pencatatan kesalahan Amazon Data Firehose melalui API, atau CloudFormation menggunakan konfigurasi. AWS CLI`CloudWatchLoggingOptions` Untuk melakukannya, buat grup log dan aliran log terlebih dahulu. Kami merekomendasikan untuk memesan grup log dan aliran log tersebut untuk pencatatan kesalahan Amazon Data Firehose secara eksklusif. Juga pastikan bahwa kebijakan IAM terkait memiliki izin `"logs:putLogEvents"`. Untuk informasi selengkapnya, lihat Mengontrol akses dengan Amazon Data Firehose.

Perhatikan bahwa Amazon Data Firehose tidak menjamin bahwa semua log kesalahan pengiriman dikirim ke CloudWatch Log. Dalam keadaan di mana tingkat kegagalan pengiriman tinggi, Amazon Data Firehose melakukan log kesalahan pengiriman sebelum mengirimnya ke CloudWatch Log.

Ada biaya nominal untuk log kesalahan yang dikirim ke CloudWatch Log. Untuk informasi selengkapnya, lihat CloudWatch Harga Amazon.

Daftar Isi

Kesalahan pengiriman data

Berikut ini adalah daftar kode kesalahan pengiriman data dan pesan untuk setiap tujuan Amazon Data Firehose. Setiap pesan kesalahan juga menjelaskan tindakan yang tepat untuk memperbaiki masalah.

Kesalahan

- Kesalahan pengiriman data Amazon S3
- Apache Iceberg Tabel Kesalahan Pengiriman Data
- Kesalahan pengiriman Data Amazon Redshift
- Kesalahan pengiriman data kepingan salju
- Kesalahan pengiriman data Splunk
- ElasticSearch Kesalahan pengiriman data
- Kesalahan pengiriman Data Titik Akhir HTTPS
- Kesalahan pengiriman Data OpenSearch Layanan Amazon
- Kesalahan pemanggilan Lambda
- Kesalahan pemanggilan Kinesis
- Kesalahan pemanggilan Kinesis DirectPut
- AWS Glue kesalahan pemanggilan
- DataFormatConversion kesalahan pemanggilan

Kesalahan pengiriman data Amazon S3

Amazon Data Firehose dapat mengirim kesalahan terkait Amazon S3 berikut ke Log. CloudWatch

Kode Kesalahan	Pesan Kesalahan dan Informasi
`S3.KMS.NotFoundException`	“ AWS KMS Kunci yang disediakan tidak ditemukan. Jika Anda menggunakan apa yang Anda yakini sebagai AWS KMS kunci yang valid dengan peran yang benar, periksa apakah ada masalah dengan akun tempat AWS KMS kunci tersebut dilampirkan.”
`S3.KMS.RequestLimitExceeded`	“Batas permintaan KMS per detik terlampaui saat mencoba mengenkripsi objek S3. Tingkatkan batas permintaan per detik.” Untuk informasi selengkapnya, lihat Batas di AWS Key Management Service Panduan Developer.
`S3.AccessDenied`	“Akses ditolak. Pastikan bahwa kebijakan kepercayaan untuk peran IAM yang disediakan memungkinkan Amazon Data Firehose untuk mengambil peran tersebut, dan kebijakan akses memungkinkan akses ke bucket S3.”

Kode Kesalahan	Pesan Kesalahan dan Informasi
`S3.AccountProblem`	“Ada masalah dengan AWS akun Anda yang mencegah operasi selesai dengan sukses. Hubungi AWS Support.”
`S3.AllAccessDisabled`	“Akses ke akun yang diberikan telah dinonaktifkan. Hubungi AWS Support.”
`S3.InvalidPayer`	“Akses ke akun yang diberikan telah dinonaktifkan. Hubungi AWS Support.”
`S3.NotSignedUp`	“Akun tersebut tidak terdaftar untuk Amazon S3. Daftarkan akun tersebut atau gunakan akun berbeda.”
`S3.NoSuchBucket`	“Bucket yang ditentukan tidak ada. Buat bucket atau gunakan bucket berbeda yang memang ada.”
`S3.MethodNotAllowed`	“Metode yang ditentukan tidak diperbolehkan untuk sumber daya ini. Modifikasi kebijakan bucket untuk memungkinkan izin operasi Amazon S3 yang benar.”
`InternalError`	“Terjadi kesalahan internal saat mencoba mengirimkan data. Pengiriman akan dicoba lagi; jika kesalahan berlanjut, maka akan dilaporkan AWS untuk resolusi.”
`S3.KMS.KeyDisabled`	“Kunci KMS yang disediakan dinonaktifkan. Aktifkan kunci atau gunakan tombol yang berbeda.”
`S3.KMS.InvalidStateException`	“Kunci KMS yang disediakan dalam keadaan tidak valid. Silakan gunakan kunci yang berbeda.”
`KMS.InvalidStateException`	“Kunci KMS yang disediakan dalam keadaan tidak valid. Silakan gunakan kunci yang berbeda.”
`KMS.DisabledException`	“Kunci KMS yang disediakan dinonaktifkan. Harap perbaiki kunci atau gunakan kunci yang berbeda.”

Kode Kesalahan	Pesan Kesalahan dan Informasi
`S3.SlowDown`	“Tingkat permintaan put ke ember yang ditentukan terlalu tinggi. Tingkatkan ukuran buffer aliran Firehose atau kurangi permintaan put dari aplikasi lain.
`S3.SubscriptionRequired`	“Akses ditolak saat menelepon S3. Pastikan bahwa peran IAM dan Kunci KMS (jika tersedia) yang diteruskan memiliki langganan Amazon S3.”
`S3.InvalidToken`	“Token yang disediakan cacat atau tidak valid. Silakan periksa kredensi yang diberikan.”
`S3.KMS.KeyNotConfigured`	“Kunci KMS tidak dikonfigurasi. Konfigurasikan KMSMaster KeyID Anda, atau nonaktifkan enkripsi untuk bucket S3 Anda.
`S3.KMS.AsymmetricCMKNotSupported`	“Amazon S3 hanya mendukung simetris. CMKs Anda tidak dapat menggunakan CMK asimetris untuk mengenkripsi data di Amazon S3. Untuk mendapatkan jenis CMK Anda, gunakan DescribeKey operasi KMS.”
`S3.IllegalLocationConstraintException`	“Firehose saat ini menggunakan titik akhir global s3 untuk pengiriman data ke bucket s3 yang dikonfigurasi. Wilayah bucket s3 yang dikonfigurasi tidak mendukung titik akhir global s3. Harap buat aliran Firehose di wilayah yang sama dengan bucket s3 atau gunakan bucket s3 di wilayah yang mendukung titik akhir global.”
`S3.InvalidPrefixConfigurationException`	“Awalan s3 khusus yang digunakan untuk evaluasi stempel waktu tidak valid. Periksa awalan s3 Anda berisi ekspresi yang valid untuk tanggal dan waktu saat ini dalam setahun.”
`DataFormatConversion.MalformedData`	“Karakter ilegal ditemukan di antara token.”

Apache Iceberg Tabel Kesalahan Pengiriman Data

Untuk kesalahan pengiriman data Apache Iceberg Tables, lihat. Kirimkan data ke Apache Iceberg Tables

Kesalahan pengiriman Data Amazon Redshift

Amazon Data Firehose dapat mengirimkan kesalahan terkait Amazon RedShift berikut ke Log. CloudWatch

Kode Kesalahan	Pesan Kesalahan dan Informasi
`Redshift.TableNotFound`	“Tabel tujuan pemuatan data tidak ditemukan. Pastikan bahwa tabel yang ditentukan memang ada.” Tabel tujuan di Amazon Redshift untuk menyalin data dari S3 tidak ditemukan. Perhatikan bahwa Amazon Data Firehose tidak membuat tabel Amazon Redshift jika tidak ada.
`Redshift.SyntaxError`	“Perintah COPY berisi kesalahan sintaks. Coba lagi perintah tersebut.”
`Redshift.AuthenticationFailed`	“Nama pengguna dan kata sandi yang diberikan gagal diautentikasi. Berikan nama pengguna dan kata sandi yang valid.
`Redshift.AccessDenied`	“Akses ditolak. Pastikan bahwa kebijakan kepercayaan untuk peran IAM yang disediakan memungkinkan Amazon Data Firehose untuk mengambil peran tersebut.
`Redshift.S3BucketAccessDenied`	“Perintah COPY tidak dapat mengakses bucket S3. Pastikan bahwa kebijakan akses untuk IAM role yang diberikan mengizinkan akses ke bucket S3.”
`Redshift.DataLoadFailed`	“Pemuatan data ke dalam tabel gagal. Periksa tabel sistem STL_LOAD_ERRORS untuk detailnya.”
`Redshift.ColumnNotFound`	“Sebuah kolom dalam perintah COPY tidak ada dalam tabel. Tentukan nama kolom yang valid.”

Kode Kesalahan	Pesan Kesalahan dan Informasi
`Redshift.DatabaseNotFound`	“Basis data yang ditentukan dalam konfigurasi tujuan Amazon Redshift atau JDBC URL tidak ditemukan. Tentukan nama basis data yang valid.”
`Redshift.IncorrectCopyOptions`	“Opsi COPY yang bertentangan atau redundan diberikan. Beberapa opsi tidak kompatibel dalam kombinasi tertentu. Periksa referensi perintah COPY untuk informasi selengkapnya.” Untuk informasi selengkapnya, lihat Perintah COPY Amazon Redshift dalam Panduan Developer Basis Data Amazon Redshift.
`Redshift.MissingColumn`	“Terdapat kolom yang didefinisikan dalam skema tabel sebagai NOT NULL tanpa nilai DEFAULT dan tidak termasuk dalam daftar kolom. Kecualikan kolom ini, pastikan bahwa data yang dimuat selalu memberikan nilai untuk kolom ini, atau tambahkan nilai default ke skema Amazon Redshift untuk tabel ini.”
`Redshift.ConnectionFailed`	“Sambungan ke klaster Amazon Redshift yang ditentukan gagal. Pastikan bahwa pengaturan keamanan memungkinkan koneksi Amazon Data Firehose, bahwa klaster atau database yang ditentukan dalam konfigurasi tujuan Amazon Redshift atau URL JDBC sudah benar, dan klaster tersedia.
`Redshift.ColumnMismatch`	“Jumlah jsonpaths dalam perintah COPY dan jumlah kolom dalam tabel tujuan harus sesuai. Coba lagi perintah tersebut.”
`Redshift.IncorrectOrMissingRegion`	“Amazon Redshift mencoba menggunakan titik akhir wilayah yang salah untuk mengakses bucket S3. Tentukan nilai wilayah yang benar dalam opsi perintah COPY atau pastikan bahwa bucket S3 berada di wilayah yang sama dengan basis data Amazon Redshift.”
`Redshift.IncorrectJsonPathsFile`	“File jsonpaths yang diberikan tidak dalam format JSON yang didukung. Coba lagi perintah tersebut.”

Kode Kesalahan	Pesan Kesalahan dan Informasi
`Redshift.MissingS3File`	“Satu atau beberapa file S3 yang dibutuhkan oleh Amazon Redshift telah dihapus dari bucket S3. Periksa kebijakan bucket S3 untuk menghapus penghapusan otomatis file S3.”
`Redshift.InsufficientPrivilege`	“Pengguna tidak memiliki izin untuk memuat data ke dalam tabel. Periksa izin pengguna Amazon Redshift untuk hak istimewa INSERT.”
`Redshift.ReadOnlyCluster`	“Kueri tidak dapat dijalankan karena sistem dalam mode perubahan ukuran. Coba lagi kueri ini nanti.”
`Redshift.DiskFull`	“Data tidak dapat dimuat karena disk penuh. Tingkatkan kapasitas klaster Amazon Redshift atau hapus data yang tidak digunakan untuk mengosongkan ruang disk.”
`InternalError`	“Terjadi kesalahan internal saat mencoba mengirimkan data. Pengiriman akan dicoba lagi; jika kesalahan berlanjut, maka akan dilaporkan AWS untuk resolusi.”
`Redshift.ArgumentNotSupported`	“Perintah COPY berisi opsi yang tidak didukung.”
`Redshift.AnalyzeTableAccessDenied`	“Akses ditolak. Salinan dari S3 ke Redshift gagal karena tabel analisis hanya dapat dilakukan oleh pemilik tabel atau database.
`Redshift.SchemaNotFound`	“Skema yang ditentukan dalam DataTableName konfigurasi tujuan Amazon Redshift tidak ditemukan. Tentukan nama skema yang valid.”
`Redshift.ColumnSpecifiedMoreThanOnce`	“Ada kolom yang ditentukan lebih dari sekali dalam daftar kolom. Pastikan kolom duplikat dihapus.”

Kode Kesalahan	Pesan Kesalahan dan Informasi
`Redshift.ColumnNotNullWithoutDefault`	“Ada kolom non-null tanpa DEFAULT yang tidak termasuk dalam daftar kolom. Pastikan kolom tersebut disertakan dalam daftar kolom.”
`Redshift.IncorrectBucketRegion`	“Redshift mencoba menggunakan ember di wilayah yang berbeda dari cluster. Harap tentukan bucket dalam wilayah yang sama dengan cluster.”
`Redshift.S3SlowDown`	“Tingkat permintaan tinggi ke S3. Kurangi laju untuk menghindari terhambat.”
`Redshift.InvalidCopyOptionForJson`	“Silakan gunakan jalur auto atau S3 yang valid untuk json CopyOption.”
`Redshift.InvalidCopyOptionJSONPathFormat`	“COPY gagal dengan kesalahan\" Format tidak valid JSONPath . Indeks array berada di luar jangkauan\”. Tolong perbaiki JSONPath ekspresinya.”
`Redshift.InvalidCopyOptionRBACAclNotAllowed`	“SALINAN gagal dengan kesalahan\" Tidak dapat menggunakan kerangka kerja acl RBAC sementara propagasi izin tidak diaktifkan. \”
`Redshift.DiskSpaceQuotaExceeded`	“Transaksi dibatalkan karena kuota ruang disk melebihi. Kosongkan ruang disk atau minta peningkatan kuota untuk skema.”
`Redshift.ConnectionsLimitExceeded`	“Batas koneksi terlampaui untuk pengguna.”

Kode Kesalahan	Pesan Kesalahan dan Informasi
`Redshift.SslNotSupported`	“Koneksi ke cluster Amazon Redshift yang ditentukan gagal karena server tidak mendukung SSL. Silakan periksa pengaturan cluster Anda.”
`Redshift.HoseNotFound`	“Selang sudah dihapus. Silakan periksa status selang Anda.”
`Redshift.Delimiter`	“Pembatas CopyOptions di CopyCommand tidak valid. Pastikan bahwa itu adalah karakter tunggal.”
`Redshift.QueryCancelled`	“Pengguna telah membatalkan operasi COPY.”
`Redshift.CompressionMismatch`	“Selang dikonfigurasi dengan UNCOMPRESSED, tetapi CopyOption menyertakan format kompresi.”
`Redshift.EncryptionCredentials`	“Opsi ENCRYPTED memerlukan kredensyal dalam format: 'aws_iam_role=... ; master_symmetric_key=... 'atau 'aws_access_key_id=... ; aws_secret_access_key=... [; token =...] ; master_symmetric_key=... '”
`Redshift.InvalidCopyOptions`	“Opsi konfigurasi COPY tidak valid.”
`Redshift.InvalidMessageFormat`	“Salin perintah berisi karakter yang tidak valid.”
`Redshift.TransactionIdLimitReached`	“Batas ID transaksi tercapai.”
`Redshift.DestinationRemoved`	“Harap verifikasi bahwa tujuan pergeseran merah ada dan dikonfigurasi dengan benar dalam konfigurasi Firehose.”

Kode Kesalahan	Pesan Kesalahan dan Informasi
`Redshift.OutOfMemory`	“Cluster Redshift kehabisan memori. Harap pastikan cluster memiliki kapasitas yang cukup.”
`Redshift.CannotForkProcess`	“Cluster Redshift kehabisan memori. Harap pastikan cluster memiliki kapasitas yang cukup.”
`Redshift.SslFailure`	“Koneksi SSL ditutup selama jabat tangan.”
`Redshift.Resize`	“Cluster Redshift sedang mengubah ukuran. Firehose tidak akan dapat mengirimkan data saat cluster sedang mengubah ukuran.”
`Redshift.ImproperQualifiedName`	“Nama yang memenuhi syarat tidak pantas (terlalu banyak nama putus-putus).”
`Redshift.InvalidJsonPathFormat`	“ JSONPath Format Tidak Valid.”
`Redshift.TooManyConnectionsException`	“Terlalu banyak koneksi ke Redshift.”
`Redshift.PSQLException`	“PSQlPengecualian diamati dari Redshift.”
`Redshift.DuplicateSecondsSpecification`	“Spesifikasi detik duplikat dalam date/time format.”

Kode Kesalahan	Pesan Kesalahan dan Informasi
`Redshift.RelationCouldNotBeOpened`	“Mengalami kesalahan Redshift, relasi tidak bisa dibuka. Periksa log Redshift untuk DB yang ditentukan.”
`Redshift.TooManyClients`	“Menemui terlalu banyak pengecualian klien dari Redshift. Kunjungi kembali koneksi maksimal ke database jika ada beberapa produsen yang menulisnya secara bersamaan.

Kesalahan pengiriman data kepingan salju

Firehose dapat mengirim kesalahan terkait Snowflake berikut ke Log. CloudWatch

Kode Kesalahan	Pesan Kesalahan dan Informasi
`Snowflake.InvalidUrl`	“Firehose tidak dapat terhubung ke Snowflake. Harap pastikan bahwa url Akun ditentukan dengan benar dalam konfigurasi tujuan Snowflake.”
`Snowflake.InvalidUser`	“Firehose tidak dapat terhubung ke Snowflake. Harap pastikan bahwa Pengguna ditentukan dengan benar dalam konfigurasi tujuan Snowflake .”
`Snowflake.InvalidRole`	“Peran kepingan salju yang ditentukan tidak ada atau tidak diizinkan. Harap pastikan bahwa peran diberikan kepada pengguna yang ditentuka n”
`Snowflake.InvalidTable`	“Tabel yang disediakan tidak ada atau tidak diizinkan”
`Snowflake.InvalidSchema`	“Skema yang disediakan tidak ada atau tidak diizinkan”
`Snowflake.InvalidDatabase`	“Database yang disediakan tidak ada atau tidak diotorisasi”

Kode Kesalahan	Pesan Kesalahan dan Informasi
`Snowflake.InvalidPrivateKeyOrPassphrase`	“Kunci pribadi atau frasa sandi yang ditentukan tidak valid. Perhatikan bahwa kunci pribadi yang disediakan harus menjadi kunci pribadi PEM RSA yang valid”
`Snowflake.MissingColumns`	“Permintaan penyisipan ditolak karena kolom yang hilang di muatan input. Pastikan bahwa nilai ditentukan untuk semua kolom non-nullable”
`Snowflake.ExtraColumns`	“Permintaan sisipan ditolak karena kolom tambahan. Kolom yang tidak ada dalam tabel seharusnya tidak ditentukan”
`Snowflake.InvalidInput`	“Pengiriman gagal karena format input tidak valid. Pastikan bahwa muatan input yang disediakan dalam format JSON dapat diterima”
`Snowflake.IncorrectValue`	“Pengiriman gagal karena tipe data yang salah dalam muatan input. Pastikan bahwa nilai JSON yang ditentukan dalam muatan input mematuhi tipe data yang dideklarasikan dalam definisi tabel Snowflake”

Kesalahan pengiriman data Splunk

Amazon Data Firehose dapat mengirimkan kesalahan terkait Splunk berikut ke Log. CloudWatch

Kode Kesalahan	Pesan Kesalahan dan Informasi
`Splunk.ProxyWithoutStickySessions`	“Jika Anda memiliki proxy (ELB atau lainnya) antara Amazon Data Firehose dan node HEC, Anda harus mengaktifkan sesi lengket untuk mendukung HEC.” ACKs
`Splunk.DisabledToken`	“Token HEC dinonaktifkan. Aktifkan token untuk mengizinkan pengiriman data ke Splunk.”
`Splunk.InvalidToken`	“Token HEC tidak valid. Perbarui Amazon Data Firehose dengan token HEC yang valid.

Kode Kesalahan	Pesan Kesalahan dan Informasi
`Splunk.InvalidDataFormat`	“Data tidak diformat dengan benar. Untuk melihat cara memformat data dengan benar untuk titik akhir HEC Raw atau Event, lihat Data Peristiwa Splunk.”
`Splunk.InvalidIndex`	“Token atau input HEC dikonfigurasi dengan indeks yang tidak valid. Periksa konfigurasi indeks Anda dan coba lagi.
`Splunk.ServerError`	“Pengiriman data ke Splunk gagal karena kesalahan server dari simpul HEC. Amazon Data Firehose akan mencoba mengirim data lagi jika durasi coba lagi di Amazon Data Firehose Anda lebih besar dari 0. Jika semua percobaan ulang gagal, Amazon Data Firehose mencadangkan data ke Amazon S3.
`Splunk.DisabledAck`	“Pengakuan pengindeks dinonaktifkan untuk token HEC. Aktifkan pengakuan pengindeks dan coba lagi. Untuk info selengkapnya, lihat Aktifkan pengakuan pengindeks.”
`Splunk.AckTimeout`	“Tidak menerima pengakuan dari HEC sebelum batas waktu pengakuan HEC berakhir. Terlepas dari batas waktu pengakuan, mungkin data berhasil diindeks di Splunk. Amazon Data Firehose mencadangkan data Amazon S3 yang batas waktu pengakuannya berakhir.”
`Splunk.MaxRetriesFailed`	“Gagal mengirimkan data ke Splunk atau menerima pengakuan. Periksa kondisi HEC Anda dan coba lagi.
`Splunk.ConnectionTimeout`	“Waktu koneksi ke Splunk habis. Ini mungkin kesalahan sementara dan permintaan akan dicoba lagi. Amazon Data Firehose mencadangkan data ke Amazon S3 jika semua percobaan ulang gagal.
`Splunk.InvalidEndpoint`	“Tidak dapat menyambung ke titik akhir HEC. Pastikan URL endpoint HEC valid dan dapat dijangkau dari Amazon Data Firehose.”
`Splunk.ConnectionClosed`	“Tidak dapat mengirim data ke Splunk karena kegagalan koneksi. Ini mungkin kesalahan sementara. Meningkatkan durasi coba lagi dalam konfigurasi Amazon Data Firehose Anda mungkin mencegah kegagalan sementara tersebut.”

Kode Kesalahan	Pesan Kesalahan dan Informasi
`Splunk.SSLUnverified`	“Tidak dapat menyambung ke titik akhir HEC. Host tidak cocok dengan sertifikat yang diberikan oleh peer. Pastikan bahwa sertifikat dan host valid.”
`Splunk.SSLHandshake`	“Tidak dapat menyambung ke titik akhir HEC. Pastikan bahwa sertifikat dan host valid.”
`Splunk.URLNotFound`	URL yang diminta tidak ditemukan di server Splunk. Silakan periksa cluster Splunk dan pastikan sudah dikonfigurasi dengan benar.”
`Splunk.ServerError.ContentTooLarge`	“Pengiriman data ke Splunk gagal karena kesalahan server dengan StatusCode: 413, pesan: permintaan yang dikirim klien Anda terlalu besar. Lihat dokumen splunk untuk mengonfigurasi max_content_length.”
`Splunk.IndexerBusy`	“Pengiriman data ke Splunk gagal karena kesalahan server dari simpul HEC. Pastikan titik akhir HEC atau Elastic Load Balancer dapat dijangkau dan sehat.”
`Splunk.ConnectionRecycled`	“Koneksi dari Firehose ke Splunk telah didaur ulang. Pengiriman akan dicoba lagi.”
`Splunk.AcknowledgementsDisabled`	“Tidak bisa mendapatkan ucapan terima kasih di POST. Pastikan bahwa pengakuan diaktifkan pada titik akhir HEC.”
`Splunk.InvalidHecResponseCharacter`	“Karakter tidak valid ditemukan dalam respons HEC, pastikan untuk memeriksa ke layanan dan konfigurasi HEC.”

ElasticSearch Kesalahan pengiriman data

Amazon Data Firehose dapat mengirim ElasticSearch kesalahan berikut ke CloudWatch Log.

Kode Kesalahan	Pesan Kesalahan dan Informasi
`ES.AccessDenied`	“Akses ditolak. Pastikan bahwa peran IAM yang disediakan terkait dengan firehose tidak dihapus.”
`ES.ResourceNotFound`	“Domain AWS Elasticsearch yang ditentukan tidak ada.”

Kesalahan pengiriman Data Titik Akhir HTTPS

Amazon Data Firehose dapat mengirimkan kesalahan terkait Titik Akhir HTTP berikut ke Log. CloudWatch Jika tidak ada satu pun dari kesalahan ini yang cocok dengan masalah yang Anda alami, kesalahan defaultnya adalah sebagai berikut: “Terjadi kesalahan internal saat mencoba mengirimkan data. Pengiriman akan dicoba lagi; jika kesalahan berlanjut, maka akan dilaporkan AWS untuk resolusi.”

Kode Kesalahan	Pesan Kesalahan dan Informasi
`HttpEndpoint.RequestTimeout`	Waktu pengiriman habis sebelum respons diterima dan akan dicoba lagi. Jika kesalahan ini berlanjut, hubungi tim layanan AWS Firehose.
`HttpEndpoint.ResponseTooLarge`	“Respons yang diterima dari titik akhir terlalu besar. Hubungi pemilik titik akhir untuk mengatasi masalah ini.”
`HttpEndpoint.InvalidResponseFromDestination`	“Respons yang diterima dari titik akhir yang ditentukan tidak valid. Hubungi pemilik titik akhir untuk mengatasi masalah ini.”
`HttpEndpoint.DestinationException`	“Respons berikut diterima dari tujuan titik akhir.”

Kode Kesalahan	Pesan Kesalahan dan Informasi
`HttpEndpoint.ConnectionFailed`	“Tidak dapat menyambung ke titik akhir tujuan. Hubungi pemilik titik akhir untuk mengatasi masalah ini.”
`HttpEndpoint.ConnectionReset`	“Tidak dapat mempertahankan koneksi dengan titik akhir. Hubungi pemilik titik akhir untuk mengatasi masalah ini.”
`HttpEndpoint.ConnectionReset`	“Ada masalah saat mempertahankan koneksi dengan titik akhir. Silakan hubungi pemilik titik akhir.”
`HttpEndpoint.ResponseReasonPhraseExceededLimit`	“Frasa alasan respons yang diterima dari titik akhir melebihi batas yang dikonfigurasi 64 karakter.”
`HttpEndpoint.InvalidResponseFromDestination`	“Tanggapan yang diterima dari endpoint tidak valid. Lihat Memecahkan Masalah Titik Akhir HTTP di dokumentasi Firehose untuk informasi selengkapnya. Alasan:”
`HttpEndpoint.DestinationException`	“Pengiriman ke titik akhir tidak berhasil. Lihat Memecahkan Masalah Titik Akhir HTTP di dokumentasi Firehose untuk informasi selengkapnya. Tanggapan diterima dengan kode status”
`HttpEndpoint.InvalidStatusCode`	“Menerima kode status respons yang tidak valid.”
`HttpEndpoint.SSLHandshakeFailure`	“Tidak dapat menyelesaikan Jabat Tangan SSL dengan titik akhir. Hubungi pemilik titik akhir untuk mengatasi masalah ini.”

Kode Kesalahan	Pesan Kesalahan dan Informasi
`HttpEndpoint.SSLHandshakeFailure`	“Tidak dapat menyelesaikan Jabat Tangan SSL dengan titik akhir. Hubungi pemilik titik akhir untuk mengatasi masalah ini.”
`HttpEndpoint.SSLFailure`	“Tidak dapat menyelesaikan jabat tangan TLS dengan titik akhir. Hubungi pemilik titik akhir untuk mengatasi masalah ini.”
`HttpEndpoint.SSLHandshakeCertificatePathFailure`	“Tidak dapat menyelesaikan Jabat Tangan SSL dengan titik akhir karena jalur sertifikasi tidak valid. Hubungi pemilik titik akhir untuk mengatasi masalah ini.”
`HttpEndpoint.SSLHandshakeCertificatePathValidationFailure`	“Tidak dapat menyelesaikan Jabat Tangan SSL dengan titik akhir karena kegagalan validasi jalur sertifikasi. Hubungi pemilik titik akhir untuk mengatasi masalah ini.”
`HttpEndpoint.MakeRequestFailure.IllegalUriException`	“HttpEndpoint permintaan gagal karena input tidak valid di URI. Pastikan semua karakter dalam URI input valid.”
`HttpEndpoint.MakeRequestFailure.IllegalCharacterInHeaderValue`	“HttpEndpoint Permintaan gagal karena kesalahan respons ilegal. Karakter ilegal '\n' dalam nilai header.”

Kode Kesalahan	Pesan Kesalahan dan Informasi
`HttpEndpoint.IllegalResponseFailure`	“HttpEndpoint Permintaan gagal karena kesalahan respons ilegal. Pesan HTTP tidak boleh berisi lebih dari satu header Content-Type.
`HttpEndpoint.IllegalMessageStart`	“HttpEndpoint Permintaan gagal karena kesalahan respons ilegal. Pesan HTTP ilegal dimulai. Lihat Pemecahan Masalah Titik Akhir HTTP di dokumentasi Firehose untuk informasi selengkapnya.”

Kesalahan pengiriman Data OpenSearch Layanan Amazon

Untuk tujuan OpenSearch Layanan, Amazon Data Firehose mengirimkan kesalahan ke CloudWatch Log saat dikembalikan oleh OpenSearch Layanan.

Selain kesalahan yang mungkin kembali dari OpenSearch cluster, Anda mungkin mengalami dua kesalahan berikut:

- Authentication/authorization error occurs during attempt to deliver data to destination OpenSearch Service cluster. This can happen due to any permission issues and/orsebentar-sebentar ketika konfigurasi domain OpenSearch Layanan target Amazon Data Firehose Anda diubah. Silakan periksa kebijakan klaster dan izin peran.
- Data tidak dapat dikirim ke kluster OpenSearch Layanan tujuan karena authentication/authorization kegagalan. Hal ini dapat terjadi karena masalah izin and/or sebentar-sebentar ketika konfigurasi domain OpenSearch Layanan target Amazon Data Firehose Anda diubah. Silakan periksa kebijakan klaster dan izin peran.

Kode Kesalahan	Pesan Kesalahan dan Informasi
`OS.AccessDenied`	“Akses ditolak. Pastikan bahwa kebijakan kepercayaan untuk peran IAM yang disediakan memungkinkan Firehose untuk mengambil peran tersebut, dan kebijakan akses memungkinkan akses ke OpenSearch Amazon Service API.”
`OS.AccessDenied`	“Akses ditolak. Pastikan bahwa kebijakan kepercayaan untuk peran IAM yang disediakan memungkinkan Firehose untuk mengambil peran

Kode Kesalahan	Pesan Kesalahan dan Informasi
	tersebut, dan kebijakan akses memungkinkan akses ke OpenSearch Amazon Service API.”
OS.AccessDenied	“Akses ditolak. Pastikan bahwa peran IAM yang disediakan terkait dengan firehose tidak dihapus.”
OS.AccessDenied	“Akses ditolak. Pastikan bahwa peran IAM yang disediakan terkait dengan firehose tidak dihapus.”
OS.ResourceNotFound	“Domain OpenSearch Layanan Amazon yang ditentukan tidak ada.”
OS.ResourceNotFound	“Domain OpenSearch Layanan Amazon yang ditentukan tidak ada.”
OS.AccessDenied	“Akses ditolak. Pastikan bahwa kebijakan kepercayaan untuk peran IAM yang disediakan memungkinkan Firehose untuk mengambil peran tersebut, dan kebijakan akses memungkinkan akses ke OpenSearch Amazon Service API.”
OS.RequestTimeout	“Permintaan ke kluster OpenSearch Layanan Amazon atau koleksi OpenSearch Tanpa Server habis. Pastikan bahwa cluster atau koleksi memiliki kapasitas yang cukup untuk beban kerja saat ini.”
OS.ClusterError	“Kluster OpenSearch Layanan Amazon mengembalikan kesalahan yang tidak ditentukan.”
OS.RequestTimeout	“Permintaan ke kluster OpenSearch Layanan Amazon habis waktunya. Pastikan bahwa cluster memiliki kapasitas yang cukup untuk beban kerja saat ini.”
OS.ConnectionFailed	“Kesulitan menghubungkan ke kluster OpenSearch Layanan Amazon atau OpenSearch koleksi Tanpa Server. Pastikan bahwa cluster atau koleksinya sehat dan dapat dijangkau.”

Kode Kesalahan	Pesan Kesalahan dan Informasi
`OS.ConnectionReset`	“Tidak dapat mempertahankan koneksi dengan kluster OpenSearch Layanan Amazon atau koleksi OpenSearch Tanpa Server. Hubungi pemilik cluster atau koleksi untuk menyelesaikan masalah ini.”
`OS.ConnectionReset`	“Kesulitan menjaga koneksi dengan kluster OpenSearch Layanan Amazon atau OpenSearch koleksi Tanpa Server. Pastikan bahwa cluster atau koleksi sehat dan memiliki kapasitas yang cukup untuk beban kerja saat ini.
`OS.ConnectionReset`	“Kesulitan menjaga koneksi dengan kluster OpenSearch Layanan Amazon atau OpenSearch koleksi Tanpa Server. Pastikan bahwa cluster atau koleksi sehat dan memiliki kapasitas yang cukup untuk beban kerja saat ini.
`OS.AccessDenied`	“Akses ditolak. Pastikan kebijakan akses pada kluster OpenSearch Layanan Amazon memberikan akses ke peran IAM yang dikonfigurasi.”
`OS.ValidationException`	“ OpenSearch Cluster mengembalikan ESService Pengecualian. Salah satu alasannya adalah bahwa cluster telah ditingkatkan ke OS 2.x atau lebih tinggi, tetapi selang masih memiliki TypeName parameter yang dikonfigurasi. Perbarui konfigurasi selang dengan menyetel TypeName ke string kosong, atau ubah titik akhir ke cluster, yang mendukung parameter Type.”
`OS.ValidationException`	“Anggota harus memenuhi pola ekspresi reguler: [a-z] [a-z0-9\\ -] +
`OS.JsonParseException`	“Cluster OpenSearch Layanan Amazon mengembalikan a JsonParse Exception. Pastikan bahwa data yang dimasukkan valid.”
`OS.AmazonOpenSearchServiceParseException`	“Cluster OpenSearch Layanan Amazon mengembalikan file AmazonOpenSearchServiceParseException. Pastikan bahwa data yang dimasukkan valid.”

Kode Kesalahan	Pesan Kesalahan dan Informasi
`OS.ExplicitIndexInBulkNotAllowed`	“Pastikan rest.action.multi.allow_explicit_index disetel ke true di kluster Layanan Amazon.” OpenSearch
`OS.ClusterError`	“Cluster OpenSearch Layanan Amazon atau koleksi OpenSearch Tanpa Server mengembalikan kesalahan yang tidak ditentukan.”
`OS.ClusterBlockException`	“Cluster mengembalikan a ClusterBlockException. Mungkin kelebihan beban.”
`OS.InvalidARN`	“ OpenSearch Layanan Amazon ARN yang disediakan tidak valid. Silakan periksa DeliveryStream konfigurasi Anda.”
`OS.MalformedData`	“Satu atau lebih catatan cacat. Harap pastikan bahwa setiap catatan adalah objek JSON tunggal yang valid dan tidak berisi baris baru.
`OS.InternalError`	“Terjadi kesalahan internal saat mencoba mengirimkan data. Pengiriman akan dicoba lagi; jika kesalahan berlanjut, itu akan dilaporkan AWS untuk resolusi.”
`OS.AliasWithMultipleIndicesNotAllowed`	“Alias memiliki lebih dari satu indeks yang terkait dengannya. Pastikan bahwa alias hanya memiliki satu indeks yang terkait dengannya.”
`OS.UnsupportedVersion`	Amazon OpenSearch Service 6.0 saat ini tidak didukung oleh Amazon Data Firehose. Hubungi AWS Support untuk informasi lebih lanjut.”
`OS.CharConversionException`	“Satu atau lebih catatan berisi karakter yang tidak valid.”
`OS.InvalidDomainNameLength`	“Panjang nama domain tidak dalam batas OS yang valid.”

Kode Kesalahan	Pesan Kesalahan dan Informasi
`OS.VPCDomainNotSupported`	“Domain OpenSearch Layanan Amazon di dalamnya saat VPCs ini tidak didukung.”
`OS.ConnectionError`	“Server http menutup koneksi secara tidak terduga, harap verifikasi kesehatan kluster OpenSearch Layanan Amazon atau koleksi Tanpa OpenSearch Server.”
`OS.LargeFieldData`	“Cluster OpenSearch Layanan Amazon membatalkan permintaan karena berisi data bidang yang lebih besar dari yang diizinkan.”
`OS.BadGateway`	“Cluster OpenSearch Layanan Amazon atau koleksi OpenSearch Tanpa Server membatalkan permintaan dengan tanggapan: 502 Bad Gateway.”
`OS.ServiceException`	“Kesalahan diterima dari kluster OpenSearch Layanan Amazon atau koleksi OpenSearch Tanpa Server. Jika cluster atau koleksi berada di belakang VPC, pastikan konfigurasi jaringan memungkinkan konektivitas.”
`OS.GatewayTimeout`	“Firehose mengalami kesalahan batas waktu saat menyambungkan ke kluster OpenSearch Layanan Amazon atau koleksi Tanpa OpenSearch Server.”
`OS.MalformedData`	Amazon Data Firehose tidak mendukung perintah Amazon OpenSearch Service Bulk API di dalam catatan Firehose.
`OS.ResponseEntryCountMismatch`	“Respons dari API Massal berisi lebih banyak entri daripada jumlah catatan yang dikirim. Pastikan bahwa setiap catatan hanya berisi satu objek JSON dan tidak ada baris baru.”

Kesalahan pemanggilan Lambda

Amazon Data Firehose dapat mengirim kesalahan pemanggilan Lambda berikut ke Log. CloudWatch

Kode Kesalahan	Pesan Kesalahan dan Informasi
`Lambda.AssumeRoleAccessDenied`	“Akses ditolak. Pastikan bahwa kebijakan kepercayaan untuk peran IAM yang disediakan memungkinkan Amazon Data Firehose untuk mengambil peran tersebut.
`Lambda.InvokeAccessDenied`	“Akses ditolak. Pastikan bahwa kebijakan akses mengizinkan akses ke fungsi Lambda.”
`Lambda.JsonProcessingException`	“Terdapat kesalahan yang menguraikan catatan yang dikembalikan dari fungsi Lambda. Pastikan bahwa catatan yang dikembalikan mengikuti model status yang diperlukan oleh Amazon Data Firehose.” Untuk informasi selengkapnya, lihat Parameter yang diperlukan untuk transformasi data.
`Lambda.InvokeLimitExceeded`	“Batas eksekusi bersamaan Lambda terlampaui. Tingkatkan batas eksekusi bersamaan.” Untuk informasi selengkapnya, lihat Batas AWS Lambda dalam Panduan Developer AWS Lambda .
`Lambda.DuplicatedRecordId`	“Beberapa catatan dikembalikan dengan ID catatan yang sama. Pastikan bahwa fungsi Lambda mengembalikan catatan unik IDs untuk setiap catatan. Untuk informasi selengkapnya, lihat Parameter yang diperlukan untuk transformasi data.
`Lambda.MissingRecordId`	“Satu atau lebih IDs rekor tidak dikembalikan. Pastikan bahwa fungsi Lambda mengembalikan semua catatan IDs yang diterima. Untuk informasi selengkapnya, lihat Parameter yang diperlukan untuk transformasi data.
`Lambda.ResourceNotFound`	“Fungsi Lambda yang ditentukan tidak ada. Gunakan fungsi berbeda yang memang ada.”

Kode Kesalahan	Pesan Kesalahan dan Informasi
`Lambda.InvalidSubnetIDException`	“ID Subnet yang ditentukan dalam konfigurasi VPC fungsi Lambda tidak valid. Pastikan ID subnet valid.”
`Lambda.InvalidSecurityGroupIDException`	“ID grup keamanan yang ditentukan dalam konfigurasi VPC fungsi Lambda tidak valid. Pastikan ID grup keamanan valid.”
`Lambda.SubnetIPAddressLimitReachedException`	“AWS Lambda Tidak dapat mengatur akses VPC untuk fungsi Lambda karena satu atau lebih subnet yang dikonfigurasi tidak memiliki alamat IP yang tersedia. Tingkatkan batas alamat IP.” Untuk informasi selengkapnya, lihat Batas Amazon VPC - VPC dan Subnet dalam Panduan Pengguna Amazon VPC.
`Lambda.ENILimitReachedException`	“AWS Lambda Tidak dapat membuat Elastic Network Interface (ENI) di VPC, yang ditentukan sebagai bagian dari konfigurasi fungsi Lambda, karena batas untuk antarmuka jaringan telah tercapai. Tingkatkan batas antarmuka jaringan.” Untuk informasi selengkapnya, lihat Batas Amazon VPC - Antarmuka Jaringan dalam Panduan Pengguna Amazon VPC.
`Lambda.FunctionTimedOut`	Waktu pemanggilan fungsi Lambda habis. Tingkatkan pengaturan Timeout dalam fungsi Lambda. Untuk informasi selengkapnya, lihat Mengonfigurasi batas waktu fungsi.

Kode Kesalahan	Pesan Kesalahan dan Informasi
`Lambda.FunctionError`	Ini dapat disebabkan oleh salah satu kesalahan berikut: • Struktur keluaran tidak valid. Periksa fungsi Anda dan pastikan output dalam format yang diperlukan. Juga, pastikan catatan yang diproses berisi status hasil yang valid dari`Dropped`,`Ok`, atau`ProcessingFailed` . • Fungsi Lambda berhasil dipanggil tetapi mengembalikan hasil kesalahan. • Lambda tidak dapat mendekripsi variabel lingkungan karena akses KMS ditolak. Periksa pengaturan tombol KMS fungsi serta kebijakan kunci. Untuk informasi selengkapnya, lihat Memecahkan Masalah Akses Kunci.
`Lambda.FunctionRequestTimedOut`	Amazon Data Firehose ditemui Permintaan tidak selesai sebelum kesalahan konfigurasi batas waktu permintaan saat menjalankan Lambda. Kunjungi kembali kode Lambda untuk memeriksa apakah kode Lambda dimaksudkan untuk berjalan melampaui batas waktu yang dikonfigurasi. Jika demikian, pertimbangkan untuk menyetel pengaturan konfigurasi Lambda, termasuk memori, batas waktu. Untuk informasi selengkapnya, lihat Mengonfigurasi opsi fungsi Lambda.
`Lambda.TargetServerFailedToRespond`	Amazon Data Firehose mengalami kesalahan. Server target gagal merespons kesalahan saat memanggil layanan AWS Lambda.
`Lambda.InvalidZipFileException`	Amazon Data Firehose ditemui InvalidZipFileException saat menjalankan fungsi Lambda. Periksa pengaturan konfigurasi fungsi Lambda Anda dan file zip kode Lambda.

Kode Kesalahan	Pesan Kesalahan dan Informasi
`Lambda.InternalServerError`	Amazon Data Firehose ditemui InternalServerError saat menelepon layanan Lambda AWS . Amazon Data Firehose akan mencoba mengirim data beberapa kali. Anda dapat menentukan atau mengganti opsi coba lagi menggunakan atau. `CreateDeliveryStream` `UpdateDestination` APIs Jika kesalahan berlanjut, hubungi tim dukungan AWS Lambda.
`Lambda.ServiceUnavailable`	Amazon Data Firehose ditemui ServiceUnavailableException saat menelepon layanan Lambda AWS . Amazon Data Firehose akan mencoba mengirim data beberapa kali. Anda dapat menentukan atau mengganti opsi coba lagi menggunakan atau. `CreateDeliveryStream` `UpdateDestination` APIs Jika kesalahan berlanjut, hubungi dukungan AWS Lambda.
`Lambda.InvalidSecurityToken`	Tidak dapat menjalankan fungsi Lambda karena token keamanan tidak valid. Pemanggilan Lambda partisi silang tidak didukung.
`Lambda.InvocationFailure`	Ini dapat disebabkan oleh salah satu kesalahan berikut: • Amazon Data Firehose mengalami kesalahan saat memanggil Lambda AWS . Operasi akan dicoba lagi; jika kesalahan berlanjut, itu akan dilaporkan AWS untuk resolusi. • Amazon Data Firehose menemukan a dari KMSInvalid StateException Lambda. Lambda tidak dapat mendekripsi variabel lingkungan karena kunci KMS yang digunakan dalam status dinonaktifkan untuk Dekripsi. Periksa tombol KMS fungsi lambda. • Amazon Data Firehose menemukan dari AWS LambdaException Lambda. Lambda tidak dapat menginisialisasi gambar kontainer yang disediakan. Verifikasi gambar. • Amazon Data Firehose mengalami kesalahan batas waktu saat memanggil Lambda. AWS Waktu habis fungsi maksimum yang didukung adalah 5 menit. Untuk informasi selengkapnya, lihat Durasi Eksekusi Transformasi Data.

Kode Kesalahan	Pesan Kesalahan dan Informasi
`Lambda.JsonMappingException`	Ada kesalahan penguraian catatan yang dikembalikan dari fungsi Lambda. Pastikan bahwa bidang data dikodekan basis-64.

Kesalahan pemanggilan Kinesis

Amazon Data Firehose dapat mengirimkan error pemanggilan Kinesis berikut ke Log. CloudWatch

Kode Kesalahan	Pesan Kesalahan dan Informasi
`Kinesis.AccessDenied`	“Akses ditolak saat memanggil Kinesis. Pastikan kebijakan akses pada peran IAM yang digunakan memungkinkan akses ke APIs Kinesis yang sesuai.”
`Kinesis.ResourceNotFound`	“Firehose gagal membaca dari sungai. Jika Firehose terpasang dengan Kinesis Stream, aliran mungkin tidak ada, atau pecahan mungkin telah digabungkan atau dipecah. Jika Firehose adalah DirectPut tipe, Firehose mungkin tidak ada lagi.”
`Kinesis.SubscriptionRequired`	“Akses ditolak saat memanggil Kinesis. Pastikan peran IAM yang diteruskan untuk akses aliran Kinesis memiliki langganan AWS Kinesis.”
`Kinesis.Throttling`	“Kesalahan pelambatan ditemui saat memanggil Kinesis. Ini bisa disebabkan oleh aplikasi lain yang memanggil sama APIs dengan aliran Firehose, atau karena Anda telah membuat terlalu banyak aliran Firehose dengan aliran Kinesis yang sama dengan sumbernya.”
`Kinesis.Throttling`	“Kesalahan pelambatan ditemui saat memanggil Kinesis. Ini bisa disebabkan oleh aplikasi lain yang memanggil sama APIs dengan aliran Firehose, atau karena Anda telah membuat terlalu banyak aliran Firehose dengan aliran Kinesis yang sama dengan sumbernya.”

Kode Kesalahan	Pesan Kesalahan dan Informasi
`Kinesis.AccessDenied`	“Akses ditolak saat memanggil Kinesis. Pastikan kebijakan akses pada peran IAM yang digunakan memungkinkan akses ke APIs Kinesis yang sesuai.”
`Kinesis.AccessDenied`	“Akses ditolak saat mencoba memanggil operasi API pada Kinesis Stream yang mendasarinya. Pastikan bahwa peran IAM disebarkan dan valid.”
`Kinesis.KMS.AccessDeniedException`	“Firehose tidak memiliki akses ke KMS Key yang digunakan untuk encrypt/decrypt Kinesis Stream. Tolong berikan akses peran pengiriman Firehose ke kunci.”
`Kinesis.KMS.KeyDisabled`	“Firehose tidak dapat membaca dari sumber Kinesis Stream karena kunci KMS yang digunakan untuk itu dinonaktifkan. encrypt/decrypt Aktifkan kunci sehingga pembacaan dapat dilanjutkan.”
`Kinesis.KMS.InvalidStateException`	“Firehose tidak dapat membaca dari sumber Kinesis Stream karena kunci KMS yang digunakan untuk mengenkripsi itu dalam keadaan tidak valid.”
`Kinesis.KMS.NotFoundException`	“Firehose tidak dapat membaca dari sumber Kinesis Stream karena kunci KMS yang digunakan untuk mengenkripsi itu tidak ditemukan.”

Kesalahan pemanggilan Kinesis DirectPut

Amazon Data Firehose dapat mengirim kesalahan DirectPut pemanggilan Kinesis berikut ke Log. CloudWatch

Kode Kesalahan	Pesan Kesalahan dan Informasi
`Firehose.KMS.Acces`	“Firehose tidak memiliki akses ke KMS Key. Silakan periksa kebijakan utamanya.”

Kode Kesalahan	Pesan Kesalahan dan Informasi
`sDeniedEx ception`	
`Firehose. KMS.Inval idStateEx ception`	Firehose tidak dapat mendekripsi data karena kunci KMS yang digunakan untuk mengenkripsi data dalam keadaan tidak valid.
`Firehose. KMS.NotFo undException`	“Firehose tidak dapat mendekripsi data karena kunci KMS yang digunakan untuk mengenkripsi itu tidak ditemukan.”
`Firehose. KMS.KeyDi sabled`	Firehose tidak dapat mendekripsi data karena kunci KMS yang digunakan untuk mengenkripsi data dinonaktifkan. Aktifkan kunci sehingga pengiriman data dapat dilanjutkan.”

AWS Glue kesalahan pemanggilan

Amazon Data Firehose dapat mengirim kesalahan AWS Glue pemanggilan berikut ke Log. CloudWatch

Kode Kesalahan	Pesan Kesalahan dan Informasi
`DataForma tConversi on.Invali dSchema`	“Skema ini tidak valid.”
`DataForma tConversi on.Entity NotFound`	“Yang ditentukan tidak table/database dapat ditemukan. Harap pastikan bahwa table/database ada dan bahwa nilai yang disediakan dalam konfigurasi skema sudah benar, terutama yang berkaitan dengan casing.”
`DataForma tConversi`	“Tidak dapat menemukan skema yang cocok dari lem. Pastikan database yang ditentukan dengan ID katalog yang disediakan ada.”

Kode Kesalahan	Pesan Kesalahan dan Informasi
`on.Invali dInput`	
`DataForma tConversi on.Invali dInput`	“Tidak dapat menemukan skema yang cocok dari lem. Pastikan ARN yang lulus dalam format yang benar.”
`DataForma tConversi on.Invali dInput`	“Tidak dapat menemukan skema yang cocok dari lem. Pastikan CataloGid yang disediakan valid.”
`DataForma tConversi on.Invali dVersionId`	“Tidak dapat menemukan skema yang cocok dari lem. Pastikan versi tabel yang ditentukan ada.”
`DataForma tConversi on.NonExi stentColumns`	“Tidak dapat menemukan skema yang cocok dari lem. Pastikan tabel dikonfigurasi dengan deskriptor penyimpanan non-null yang berisi kolom target.”
`DataForma tConversi on.Access Denied`	“Akses ditolak saat mengambil peran. Harap pastikan bahwa peran yang ditentukan dalam konfigurasi konversi format data telah memberikan izin layanan Firehose untuk mengambilnya.”
`DataForma tConversi on.Thrott ledByGlue`	“Kesalahan pelambatan ditemui saat memanggil Glue. Baik meningkat kan batas tingkat permintaan atau mengurangi tingkat lem panggilan saat ini melalui aplikasi lain.”

Kode Kesalahan	Pesan Kesalahan dan Informasi
`DataFormatConversion.AccessDenied`	“Akses ditolak saat menelepon Glue. Harap pastikan bahwa peran yang ditentukan dalam konfigurasi konversi format data memiliki izin yang diperlukan.”
`DataFormatConversion.InvalidGlueRole`	“Peran tidak valid. Harap pastikan bahwa peran yang ditentukan dalam konfigurasi konversi format data ada.”
`DataFormatConversion.InvalidGlueRole`	“Token keamanan yang termasuk dalam permintaan tidak valid. Pastikan bahwa peran IAM yang disediakan terkait dengan firehose tidak dihapus.”
`DataFormatConversion.GlueNotAvailableInRegion`	“AWS Glue belum tersedia di wilayah yang telah Anda tentukan; harap tentukan wilayah yang berbeda.”
`DataFormatConversion.GlueEncryptionException`	“Ada kesalahan saat mengambil kunci master. Pastikan kunci itu ada dan memiliki izin akses yang benar.
`DataFormatConversion.SchemaValidationTimeout`	“Habis waktu saat mengambil tabel dari Glue. Jika Anda memiliki banyak versi tabel Glue, harap tambahkan izin 'lem: GetTableVersion '(disaran kan) atau hapus versi tabel yang tidak digunakan. Jika Anda tidak memiliki banyak tabel di Glue, silakan hubungi AWS Support.”

Kode Kesalahan	Pesan Kesalahan dan Informasi
`DataFirehose.InternalError`	“Habis waktu saat mengambil tabel dari Glue. Jika Anda memiliki banyak versi tabel Glue, harap tambahkan izin 'lem: GetTableVersion '(disarankan) atau hapus versi tabel yang tidak digunakan. Jika Anda tidak memiliki banyak tabel di Glue, silakan hubungi AWS Support.”
`DataFormatConversion.GlueEncryptionException`	“Ada kesalahan saat mengambil kunci master. Pastikan bahwa kuncinya ada dan statusnya benar.”

DataFormatConversion kesalahan pemanggilan

Amazon Data Firehose dapat mengirim kesalahan DataFormatConversion pemanggilan berikut ke Log. CloudWatch

Kode Kesalahan	Pesan Kesalahan dan Informasi
`DataFormatConversion.InvalidSchema`	“Skema ini tidak valid.”
`DataFormatConversion.ValidationException`	“Nama dan tipe kolom harus berupa string yang tidak kosong.”
`DataFormatConversion.ParseError`	“Menemui JSON yang cacat.”
`DataFormatConversi`	“Data tidak cocok dengan skema.”

Kode Kesalahan	Pesan Kesalahan dan Informasi
on.Malfor medData	
DataForma tConversi on.Malfor medData	“Panjang kunci json tidak boleh lebih besar dari 262144"
DataForma tConversi on.Malfor medData	“Data tidak dapat diterjemahkan sebagai UTF-8.”
DataForma tConversi on.Malfor medData	“Karakter ilegal ditemukan di antara token.”
DataForma tConversi on.Invali dTypeFormat	“Format tipe tidak valid. Periksa sintaks tipe.”
DataForma tConversi on.Invali dSchema	“Skema Tidak Valid. Harap pastikan bahwa tidak ada karakter khusus atau spasi putih di nama kolom.”
DataForma tConversi on.Invali dRecord	“Rekaman tidak sesuai skema. <string, string>Satu atau lebih kunci peta tidak valid untuk peta.”

Kode Kesalahan	Pesan Kesalahan dan Informasi
`DataFormatConversion.MalformedData`	“Masukan JSON berisi primitif di tingkat atas. Tingkat atas harus berupa objek atau array.”
`DataFormatConversion.MalformedData`	“Masukan JSON berisi primitif di tingkat atas. Tingkat atas harus berupa objek atau array.”
`DataFormatConversion.MalformedData`	“Rekaman itu kosong atau hanya berisi spasi putih.”
`DataFormatConversion.MalformedData`	“Menemukan karakter yang tidak valid.”
`DataFormatConversion.MalformedData`	“Menemukan format stempel waktu yang tidak valid atau tidak didukung. Silakan lihat panduan pengembang Firehose untuk format stempel waktu yang didukung.”
`DataFormatConversion.MalformedData`	“Jenis skalar ditemukan dalam data tetapi tipe kompleks ditentukan pada skema.”
`DataFormatConversion.MalformedData`	“Data tidak cocok dengan skema.”

Kode Kesalahan	Pesan Kesalahan dan Informasi
`DataFormatConversion.MalformedData`	“Jenis skalar ditemukan dalam data tetapi tipe kompleks ditentukan pada skema.”
`DataFormatConversion.ConversionFailureException`	"ConversionFailureException"
`DataFormatConversion.DataFormatConversionCustomerErrorException`	"DataFormatConversionCustomerErrorException"
`DataFormatConversion.DataFormatConversionCustomerErrorException`	"DataFormatConversionCustomerErrorException"
`DataFormatConversion.MalformedData`	“Data tidak cocok dengan skema.”

Kode Kesalahan	Pesan Kesalahan dan Informasi
`DataFormatConversion.InvalidSchema`	“Skema ini tidak valid.”
`DataFormatConversion.MalformedData`	“Data tidak cocok dengan skema. Format tidak valid untuk satu atau beberapa tanggal.
`DataFormatConversion.MalformedData`	“Data berisi struktur JSON yang sangat bersarang yang tidak didukung.”
`DataFormatConversion.EntityNotFound`	“Yang ditentukan tidak table/database dapat ditemukan. Harap pastikan bahwa table/database ada dan bahwa nilai yang disediakan dalam konfigurasi skema sudah benar, terutama yang berkaitan dengan casing.”
`DataFormatConversion.InvalidInput`	“Tidak dapat menemukan skema yang cocok dari lem. Pastikan database yang ditentukan dengan ID katalog yang disediakan ada.”
`DataFormatConversion.InvalidInput`	“Tidak dapat menemukan skema yang cocok dari lem. Pastikan ARN yang lulus dalam format yang benar.”
`DataFormatConversion.InvalidInput`	“Tidak dapat menemukan skema yang cocok dari lem. Pastikan CataloGid yang disediakan valid.”

Kode Kesalahan	Pesan Kesalahan dan Informasi
`DataFormatConversion.InvalidVersionId`	“Tidak dapat menemukan skema yang cocok dari lem. Pastikan versi tabel yang ditentukan ada.”
`DataFormatConversion.NonExistentColumns`	“Tidak dapat menemukan skema yang cocok dari lem. Pastikan tabel dikonfigurasi dengan deskriptor penyimpanan non-null yang berisi kolom target.”
`DataFormatConversion.AccessDenied`	“Akses ditolak saat mengambil peran. Harap pastikan bahwa peran yang ditentukan dalam konfigurasi konversi format data telah memberikan izin layanan Firehose untuk mengambilnya.”
`DataFormatConversion.ThrottledByGlue`	“Kesalahan pelambatan ditemui saat memanggil Glue. Baik meningkatkan batas tingkat permintaan atau mengurangi tingkat lem panggilan saat ini melalui aplikasi lain.”
`DataFormatConversion.AccessDenied`	“Akses ditolak saat menelepon Glue. Harap pastikan bahwa peran yang ditentukan dalam konfigurasi konversi format data memiliki izin yang diperlukan.”
`DataFormatConversion.InvalidGlueRole`	“Peran tidak valid. Harap pastikan bahwa peran yang ditentukan dalam konfigurasi konversi format data ada.”
`DataFormatConversion.GlueNotAvailableInRegion`	“AWS Glue belum tersedia di wilayah yang telah Anda tentukan; harap tentukan wilayah yang berbeda.”

Kode Kesalahan	Pesan Kesalahan dan Informasi
`DataFormatConversion.GlueEncryptionException`	“Ada kesalahan saat mengambil kunci master. Pastikan kunci itu ada dan memiliki izin akses yang benar.
`DataFormatConversion.SchemaValidationTimeout`	“Habis waktu saat mengambil tabel dari Glue. Jika Anda memiliki banyak versi tabel Glue, harap tambahkan izin 'lem: GetTableVersion '(disarankan) atau hapus versi tabel yang tidak digunakan. Jika Anda tidak memiliki banyak tabel di Glue, silakan hubungi AWS Support.”
`DataFirehose.InternalError`	“Habis waktu saat mengambil tabel dari Glue. Jika Anda memiliki banyak versi tabel Glue, harap tambahkan izin 'lem: GetTableVersion '(disarankan) atau hapus versi tabel yang tidak digunakan. Jika Anda tidak memiliki banyak tabel di Glue, silakan hubungi AWS Support.”
`DataFormatConversion.MalformedData`	“Satu atau beberapa bidang memiliki format yang salah.”

Akses CloudWatch log untuk Amazon Data Firehose

Anda dapat melihat log kesalahan yang terkait dengan kegagalan pengiriman data Amazon Data Firehose menggunakan konsol Amazon Data Firehose atau konsol. CloudWatch Prosedur berikut menunjukkan cara mengakses log kesalahan menggunakan kedua metode ini.

Untuk mengakses log kesalahan menggunakan konsol Amazon Data Firehose

1. Masuk ke Konsol Manajemen AWS dan buka Firehose console di /firehose https://console.aws.amazon.com
2. Pada bilah navigasi, pilih AWS Wilayah.
3. Pilih nama aliran Firehose untuk membuka halaman detail aliran Firehose.

4. Pilih Log Kesalahan untuk melihat daftar log kesalahan yang terkait dengan kegagalan pengiriman data.

Untuk mengakses log kesalahan menggunakan CloudWatch konsol

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pada bilah navigasi, pilih Wilayah.
3. Pilih Log di panel navigasi.
4. Pilih grup log dan aliran log untuk melihat daftar log kesalahan yang terkait dengan kegagalan pengiriman data.

Pantau kesehatan Agen Kinesis

Kinesis Agent menerbitkan CloudWatch metrik kustom dengan namespace. AWS KinesisAgent Ini membantu menilai apakah agen sehat, mengirimkan data ke Amazon Data Firehose seperti yang ditentukan, dan mengkonsumsi jumlah CPU dan sumber daya memori yang sesuai pada produsen data.

Metrik seperti jumlah catatan dan byte yang dikirim berguna untuk memahami tingkat di mana agen mengirimkan data ke aliran Firehose. Bila metrik ini turun di bawah ambang batas yang diharapkan sejumlah sekian persen atau turun ke nol, hal ini dapat menunjukkan masalah konfigurasi, kesalahan jaringan, atau masalah kondisi agen. Metrik seperti konsumsi CPU dan memori on-host serta penghitung kesalahan agen menunjukkan penggunaan sumber daya produsen data dan memberikan gambaran mengenai kemungkinan kesalahan konfigurasi atau host. Pada akhirnya, agen juga mencatat pengecualian layanan untuk membantu menyelidiki masalah agen.

Metrik agen dilaporkan di wilayah yang ditentukan dalam pengaturan konfigurasi agen `cloudwatch.endpoint`. Untuk informasi selengkapnya, lihat Tentukan pengaturan konfigurasi agen.

Metrik Cloudwatch yang diterbitkan dari beberapa Agen Kinesis dikumpulkan atau digabungkan.

Terdapat biaya nominal untuk metrik yang dikeluarkan dari Agen Kinesis, yang diaktifkan secara default. Untuk informasi selengkapnya, lihat CloudWatch Harga Amazon.

Monitor dengan CloudWatch

Agen Kinesis mengirimkan metrik berikut ke. CloudWatch

Metrik	Deskripsi
BytesSent	Jumlah byte yang dikirim ke aliran Firehose selama periode waktu yang ditentukan. Unit: Byte
RecordSendAttempts	Jumlah catatan yang dicoba (baik pertama kali, atau percobaan ulang) dalam panggilan ke PutRecordBatch selama periode waktu yang ditentukan. Unit: Hitungan
RecordSendErrors	Jumlah catatan yang menghasilkan status gagal dalam panggilan ke PutRecordBatch , termasuk percobaan ulang, selama periode waktu yang ditentukan. Unit: Hitungan
ServiceErrors	Jumlah panggilan ke PutRecordBatch yang mengakibatkan kesalahan layanan (selain kesalahan throttling) selama periode waktu yang ditentukan. Unit: Hitungan

Log panggilan Amazon Data Firehose API dengan AWS CloudTrail

Amazon Data Firehose terintegrasi dengan AWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau AWS layanan di Amazon Data Firehose. CloudTrail menangkap semua panggilan API untuk Amazon Data Firehose sebagai peristiwa. Panggilan yang diambil termasuk panggilan dari konsol Amazon Data Firehose dan panggilan kode ke operasi Amazon Data Firehose API. Jika Anda membuat jejak, Anda dapat mengaktifkan pengiriman CloudTrail acara secara terus menerus ke bucket Amazon S3, termasuk acara untuk Amazon Data Firehose. 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 Amazon Data Firehose, alamat IP dari mana permintaan dibuat, siapa yang membuat permintaan, kapan dibuat, dan detail tambahan.

Untuk mempelajari selengkapnya CloudTrail, termasuk cara mengonfigurasi dan mengaktifkannya, lihat Panduan AWS CloudTrail Pengguna.

Informasi Firehose di CloudTrail

CloudTrail diaktifkan di AWS akun Anda saat Anda membuat akun. Ketika aktivitas peristiwa yang didukung terjadi di Amazon Data Firehose, 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 Anda. Untuk informasi lain, lihat Melihat Peristiwa dengan Riwayat Peristiwa CloudTrail.

Untuk catatan peristiwa yang sedang berlangsung di AWS akun Anda, termasuk peristiwa untuk Amazon Data Firehose, buat jejak. Jejak memungkinkan CloudTrail untuk mengirimkan file log ke bucket Amazon S3. Secara default, saat Anda membuat jejak di konsol, jejak tersebut berlaku untuk semua AWS Wilayah. Jejak mencatat peristiwa dari semua Wilayah di AWS partisi 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
- Mengonfigurasi Notifikasi Amazon SNS untuk CloudTrail
- Menerima File CloudTrail Log dari Beberapa Wilayah dan Menerima File CloudTrail Log dari Beberapa Akun

Amazon Data Firehose mendukung pencatatan tindakan berikut sebagai peristiwa dalam file CloudTrail log:

- CreateDeliveryStream
- DeleteDeliveryStream
- DescribeDeliveryStream
- ListDeliveryStreams
- ListTagsForDeliveryStream
- TagDeliveryStream
- StartDeliveryStreamEncryption
- StopDeliveryStreamEncryption

- UntagDeliveryStream
- UpdateDestination

Setiap entri peristiwa atau log berisi informasi tentang entitas yang membuat permintaan tersebut. Informasi identitas membantu Anda menentukan hal berikut ini:

- Apakah permintaan itu dibuat dengan kredensyal pengguna root atau AWS Identity and Access Management (IAM).
- Apakah permintaan tersebut dibuat dengan kredensial keamanan sementara untuk satu peran atau pengguna gabungan.
- Apakah permintaan itu dibuat oleh AWS layanan lain.

Untuk informasi lain, lihat Elemen userIdentity CloudTrail.

Contoh: entri file log Firehose

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. Sebuah kejadian mewakili permintaan tunggal dari sumber apa pun dan mencakup informasi tentang tindakan yang diminta, tanggal dan waktu tindakan, parameter permintaan, dan sebagainya. Berkas log CloudTrail bukan merupakan jejak tumpukan terurut dari panggilan API publik, sehingga tidak muncul dalam urutan tertentu.

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan`CreateDeliveryStream`,,`DescribeDeliveryStream`, `ListDeliveryStreams`,`UpdateDestination`, dan `DeleteDeliveryStream` tindakan.

```
{
  "Records":[
        {
            "eventVersion":"1.02",
            "userIdentity":{
                "type":"IAMUser",
                "principalId":"AKIAIOSFODNN7EXAMPLE",
                "arn":"arn:aws:iam::111122223333:user/CloudTrail_Test_User",
                "accountId":"111122223333",
                "accessKeyId":"AKIAI44QH8DHBEXAMPLE",
                "userName":"CloudTrail_Test_User"
            },
```

```
            "eventTime":"2016-02-24T18:08:22Z",
            "eventSource":"firehose.amazonaws.com",
            "eventName":"CreateDeliveryStream",
            "awsRegion":"us-east-1",
            "sourceIPAddress":"127.0.0.1",
            "userAgent":"aws-internal/3",
            "requestParameters":{
                "deliveryStreamName":"TestRedshiftStream",
                "redshiftDestinationConfiguration":{
                "s3Configuration":{
                    "compressionFormat":"GZIP",
                    "prefix":"prefix",
                    "bucketARN":"arn:aws:s3:::amzn-s3-demo-bucket",
                    "roleARN":"arn:aws:iam::111122223333:role/Firehose",
                    "bufferingHints":{
                        "sizeInMBs":3,
                        "intervalInSeconds":900
                    },
                    "encryptionConfiguration":{
                        "kMSEncryptionConfig":{
                            "aWSKMSKeyARN":"arn:aws:kms:us-east-1:key"
                        }
                    }
                },
                "clusterJDBCURL":"jdbc:redshift://example.abc123.us-
west-2.redshift.amazonaws.com:5439/dev",
                "copyCommand":{
                    "copyOptions":"copyOptions",
                    "dataTableName":"dataTable"
                },
                "password":"",
                "username":"",
                "roleARN":"arn:aws:iam::111122223333:role/Firehose"
            }
        },
        "responseElements":{
            "deliveryStreamARN":"arn:aws:firehose:us-
east-1:111122223333:deliverystream/TestRedshiftStream"
        },
        "requestID":"958abf6a-db21-11e5-bb88-91ae9617edf5",
        "eventID":"875d2d68-476c-4ad5-bbc6-d02872cfc884",
        "eventType":"AwsApiCall",
        "recipientAccountId":"111122223333"
    },
```

```
    {
        "eventVersion":"1.02",
        "userIdentity":{
            "type":"IAMUser",
            "principalId":"AKIAIOSFODNN7EXAMPLE",
            "arn":"arn:aws:iam::111122223333:user/CloudTrail_Test_User",
            "accountId":"111122223333",
            "accessKeyId":"AKIAI44QH8DHBEXAMPLE",
            "userName":"CloudTrail_Test_User"
        },
        "eventTime":"2016-02-24T18:08:54Z",
        "eventSource":"firehose.amazonaws.com",
        "eventName":"DescribeDeliveryStream",
        "awsRegion":"us-east-1",
        "sourceIPAddress":"127.0.0.1",
        "userAgent":"aws-internal/3",
        "requestParameters":{
            "deliveryStreamName":"TestRedshiftStream"
        },
        "responseElements":null,
        "requestID":"aa6ea5ed-db21-11e5-bb88-91ae9617edf5",
        "eventID":"d9b285d8-d690-4d5c-b9fe-d1ad5ab03f14",
        "eventType":"AwsApiCall",
        "recipientAccountId":"111122223333"
    },
    {
        "eventVersion":"1.02",
        "userIdentity":{
            "type":"IAMUser",
            "principalId":"AKIAIOSFODNN7EXAMPLE",
            "arn":"arn:aws:iam::111122223333:user/CloudTrail_Test_User",
            "accountId":"111122223333",
            "accessKeyId":"AKIAI44QH8DHBEXAMPLE",
            "userName":"CloudTrail_Test_User"
        },
        "eventTime":"2016-02-24T18:10:00Z",
        "eventSource":"firehose.amazonaws.com",
        "eventName":"ListDeliveryStreams",
        "awsRegion":"us-east-1",
        "sourceIPAddress":"127.0.0.1",
        "userAgent":"aws-internal/3",
        "requestParameters":{
            "limit":10
        },
```

```
        "responseElements":null,
        "requestID":"d1bf7f86-db21-11e5-bb88-91ae9617edf5",
        "eventID":"67f63c74-4335-48c0-9004-4ba35ce00128",
        "eventType":"AwsApiCall",
        "recipientAccountId":"111122223333"
    },
    {
        "eventVersion":"1.02",
        "userIdentity":{
            "type":"IAMUser",
            "principalId":"AKIAIOSFODNN7EXAMPLE",
            "arn":"arn:aws:iam::111122223333:user/CloudTrail_Test_User",
            "accountId":"111122223333",
            "accessKeyId":"AKIAI44QH8DHBEXAMPLE",
            "userName":"CloudTrail_Test_User"
        },
        "eventTime":"2016-02-24T18:10:09Z",
        "eventSource":"firehose.amazonaws.com",
        "eventName":"UpdateDestination",
        "awsRegion":"us-east-1",
        "sourceIPAddress":"127.0.0.1",
        "userAgent":"aws-internal/3",
        "requestParameters":{
            "destinationId":"destinationId-000000000001",
            "deliveryStreamName":"TestRedshiftStream",
            "currentDeliveryStreamVersionId":"1",
            "redshiftDestinationUpdate":{
                "roleARN":"arn:aws:iam::111122223333:role/Firehose",
                "clusterJDBCURL":"jdbc:redshift://example.abc123.us-
west-2.redshift.amazonaws.com:5439/dev",
                "password":"",
                "username":"",
                "copyCommand":{
                    "copyOptions":"copyOptions",
                    "dataTableName":"dataTable"
                },
                "s3Update":{
                    "bucketARN":"arn:aws:s3:::amzn-s3-demo-bucket-update",
                    "roleARN":"arn:aws:iam::111122223333:role/Firehose",
                    "compressionFormat":"GZIP",
                    "bufferingHints":{
                        "sizeInMBs":3,
                        "intervalInSeconds":900
                    },
```

```
                    "encryptionConfiguration":{
                        "kMSEncryptionConfig":{
                            "aWSKMSKeyARN":"arn:aws:kms:us-east-1:key"
                        }
                    },
                    "prefix":"arn:aws:s3:::amzn-s3-demo-bucket"
                }
            }
        },
        "responseElements":null,
        "requestID":"d549428d-db21-11e5-bb88-91ae9617edf5",
        "eventID":"1cb21e0b-416a-415d-bbf9-769b152a6585",
        "eventType":"AwsApiCall",
        "recipientAccountId":"111122223333"
    },
    {
        "eventVersion":"1.02",
        "userIdentity":{
            "type":"IAMUser",
            "principalId":"AKIAIOSFODNN7EXAMPLE",
            "arn":"arn:aws:iam::111122223333:user/CloudTrail_Test_User",
            "accountId":"111122223333",
            "accessKeyId":"AKIAI44QH8DHBEXAMPLE",
            "userName":"CloudTrail_Test_User"
        },
        "eventTime":"2016-02-24T18:10:12Z",
        "eventSource":"firehose.amazonaws.com",
        "eventName":"DeleteDeliveryStream",
        "awsRegion":"us-east-1",
        "sourceIPAddress":"127.0.0.1",
        "userAgent":"aws-internal/3",
        "requestParameters":{
            "deliveryStreamName":"TestRedshiftStream"
        },
        "responseElements":null,
        "requestID":"d85968c1-db21-11e5-bb88-91ae9617edf5",
        "eventID":"dd46bb98-b4e9-42ff-a6af-32d57e636ad1",
        "eventType":"AwsApiCall",
        "recipientAccountId":"111122223333"
    }
  ]
}
```

Contoh kode untuk Firehose menggunakan AWS SDKs

Contoh kode berikut menunjukkan cara menggunakan Firehose dengan kit pengembangan AWS perangkat lunak (SDK).

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Skenario adalah contoh kode yang menunjukkan kepada Anda bagaimana menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan atau dikombinasikan dengan yang lain Layanan AWS.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan Firehose dengan SDK AWS. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Contoh kode

- Contoh dasar untuk Firehose menggunakan AWS SDKs
  - Tindakan untuk Firehose menggunakan AWS SDKs
    - Gunakan PutRecord dengan AWS SDK atau CLI
    - Gunakan PutRecordBatch dengan AWS SDK atau CLI
- Skenario untuk Firehose menggunakan AWS SDKs
  - Menggunakan Amazon Data Firehose untuk memproses catatan individu dan batch

Contoh dasar untuk Firehose menggunakan AWS SDKs

Contoh kode berikut menunjukkan cara menggunakan dasar-dasar Amazon Data Firehose dengan. AWS SDKs

Contoh

- Tindakan untuk Firehose menggunakan AWS SDKs
  - Gunakan PutRecord dengan AWS SDK atau CLI
  - Gunakan PutRecordBatch dengan AWS SDK atau CLI

Tindakan untuk Firehose menggunakan AWS SDKs

Contoh kode berikut menunjukkan cara melakukan tindakan Firehose individual dengan. AWS SDKs Setiap contoh menyertakan tautan ke GitHub, di mana Anda dapat menemukan instruksi untuk mengatur dan menjalankan kode.

Kutipan ini memanggil Firehose API dan merupakan kutipan kode dari program yang lebih besar yang harus dijalankan dalam konteks. Anda dapat melihat tindakan dalam konteks diSkenario untuk Firehose menggunakan AWS SDKs.

Contoh berikut hanya mencakup tindakan yang paling umum digunakan. Untuk daftar lengkapnya, lihat Referensi API Amazon Data Firehose.

Contoh

- Gunakan PutRecord dengan AWS SDK atau CLI
- Gunakan PutRecordBatch dengan AWS SDK atau CLI

Gunakan **`PutRecord`** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan`PutRecord`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- Masukkan catatan ke Firehose

CLI

AWS CLI

Untuk menulis catatan ke aliran

`put-record`Contoh berikut menulis data ke aliran. Data dikodekan dalam format Base64.

```
aws firehose put-record \
    --delivery-stream-name my-stream \
    --record '{"Data":"SGVsbG8gd29ybGQ="}'
```

Output:

```
{
    "RecordId": "RjB5K/nnoGFHqwTsZlNd/
TTqvjE8V5dsyXZTQn2JXrdpMTOwssyEb6nfC8fwf1whhwnItt4mvrn+gsqeK5jB7QjuLg283+Ps4Sz/
j1Xujv31iDhnPdaLw4BOyM9Amv7PcCuB2079RuM0NhoakbyUymlwY8yt20G8X2420wu1jlFafhci4erAt7QhDEvpw
    "Encrypted": false
}
```

Untuk informasi selengkapnya, lihat Mengirim Data ke Aliran Pengiriman Amazon Kinesis Data Firehose di Panduan Pengembang Amazon Kinesis Data Firehose.

- Untuk detail API, lihat PutRecorddi Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
   /**
    * Puts a record to the specified Amazon Kinesis Data Firehose delivery
stream.
    *
    * @param record The record to be put to the delivery stream. The record must
be a {@link Map} of String keys and Object values.
    * @param deliveryStreamName The name of the Amazon Kinesis Data Firehose
delivery stream to which the record should be put.
    * @throws IllegalArgumentException if the input record or delivery stream
name is null or empty.
    * @throws RuntimeException if there is an error putting the record to the
delivery stream.
    */
   public static void putRecord(Map<String, Object> record, String
deliveryStreamName) {
       if (record == null || deliveryStreamName == null ||
deliveryStreamName.isEmpty()) {
           throw new IllegalArgumentException("Invalid input: record or delivery
stream name cannot be null/empty");
```

```
        }
        try {
            String jsonRecord = new ObjectMapper().writeValueAsString(record);
            Record firehoseRecord = Record.builder()

.data(SdkBytes.fromByteArray(jsonRecord.getBytes(StandardCharsets.UTF_8)))
                .build();

            PutRecordRequest putRecordRequest = PutRecordRequest.builder()
                .deliveryStreamName(deliveryStreamName)
                .record(firehoseRecord)
                .build();

            getFirehoseClient().putRecord(putRecordRequest);
            System.out.println("Record sent: " + jsonRecord);
        } catch (Exception e) {
            throw new RuntimeException("Failed to put record: " + e.getMessage(),
e);
        }
    }
```

- Untuk detail API, lihat PutRecorddi Referensi AWS SDK for Java 2.x API.

Python

SDK untuk Python (Boto3)

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
class FirehoseClient:
    """
    AWS Firehose client to send records and monitor metrics.

    Attributes:
        config (object): Configuration object with delivery stream name and
 region.
        delivery_stream_name (str): Name of the Firehose delivery stream.
```

```
        region (str): AWS region for Firehose and CloudWatch clients.
        firehose (boto3.client): Boto3 Firehose client.
        cloudwatch (boto3.client): Boto3 CloudWatch client.
    """

    def __init__(self, config):
        """
        Initialize the FirehoseClient.

        Args:
            config (object): Configuration object with delivery stream name and
region.
        """
        self.config = config
        self.delivery_stream_name = config.delivery_stream_name
        self.region = config.region
        self.firehose = boto3.client("firehose", region_name=self.region)
        self.cloudwatch = boto3.client("cloudwatch", region_name=self.region)


    @backoff.on_exception(
        backoff.expo, Exception, max_tries=5, jitter=backoff.full_jitter
    )
    def put_record(self, record: dict):
        """
        Put individual records to Firehose with backoff and retry.

        Args:
            record (dict): The data record to be sent to Firehose.

        This method attempts to send an individual record to the Firehose
delivery stream.
        It retries with exponential backoff in case of exceptions.
        """
        try:
            entry = self._create_record_entry(record)
            response = self.firehose.put_record(
                DeliveryStreamName=self.delivery_stream_name, Record=entry
            )
            self._log_response(response, entry)
        except Exception:
            logger.info(f"Fail record: {record}.")
            raise
```

- Untuk detail API, lihat PutRecorddi AWS SDK for Python (Boto3) Referensi API.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
TRY.
    DATA(lo_record) = NEW /aws1/cl_frhrecord( iv_data = iv_data ).

    DATA(lo_result) = lo_frh->putrecord(
      iv_deliverystreamname = iv_deliv_stream_name
      io_record             = lo_record ).

    MESSAGE 'Record sent to Firehose delivery stream.' TYPE 'I'.
  CATCH /aws1/cx_frhresourcenotfoundex.
    MESSAGE 'Delivery stream not found.' TYPE 'E'.
  CATCH /aws1/cx_frhinvalidargumentex.
    MESSAGE 'Invalid argument provided.' TYPE 'E'.
  CATCH /aws1/cx_frhserviceunavailex.
    MESSAGE 'Service temporarily unavailable.' TYPE 'E'.
ENDTRY.
```

- Untuk detail API, lihat PutRecorddi AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan Firehose dengan SDK AWS. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **`PutRecordBatch`** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan`PutRecordBatch`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- Masukkan catatan ke Firehose

CLI

AWS CLI

Untuk menulis beberapa catatan ke aliran

`put-record-batch`Contoh berikut menulis tiga catatan ke aliran. Data dikodekan dalam format Base64.

```
aws firehose put-record-batch \
    --delivery-stream-name my-stream \
    --records file://records.json
```

Isi dari `myfile.json`:

```
[
    {"Data": "Rmlyc3QgdGhpbmc="},
    {"Data": "U2Vjb25kIHRoaW5n"},
    {"Data": "VGhpcmQgdGhpbmc="}
]
```

Output:

```
{
    "FailedPutCount": 0,
    "Encrypted": false,
    "RequestResponses": [
        {
            "RecordId": "9D2OJ6t2EqCTZTXwGzeSv/EVHxRoRCw89xd+o3+sXg8DhYOaWKPSmZy/
CGlRVEys1u1xbeKh6VofEYKkoeiDrcjrxhQp9iF7sUW7pujiMEQ5LzlrzCkGosxQn
+3boDnURDEaD42V7GiixpOyLJkYZcae1i7HzlCEoy9LJhMr8EjDSi4Om/9Vc2uhwwuAtGE0XKpxJ2WD7ZRWtAnYlK
        },
        {
            "RecordId": "jFirejqxCLlK5xjH/UNmlMVcjktEN76I7916X9PaZ
+PVaOSXDfU1WGOqEZhxq2js7xcZ552eoeDxsuTU1MSq9nZTbVfb6cQTIXnm/
GsuF37Uhg67GKmR5z90l6XKJ+/
```

```
+pDloFv7Hh9a3oUS6wYm3DcNRLTHHAimANp1PhkQvWpvLRfzbuCUkBphR2QVzhP9OiHLbzGwy8/
DfH8sqWEUYASNJKS8GXP5s"
        },
        {
            "RecordId":
 "oy0amQ40o5Y2YV4vxzufdcMOOw6n3EPr3tpPJGoYVNKH4APPVqNcbUgefo1stEFRg4hTLrf2k6eliHu/9+YJ5R3
DTBt3qBlmTj7Xq8SKVbO1S7YvMTpWkMKA86f8JfmT8BMKoMb4XZS/sOkQLe+qh0sYKXWl"
        }
    ]
}
```

Untuk informasi selengkapnya, lihat Mengirim Data ke Aliran Pengiriman Amazon Kinesis Data Firehose di Panduan Pengembang Amazon Kinesis Data Firehose.

- Untuk detail API, lihat PutRecordBatchdi Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    /**
     * Puts a batch of records to an Amazon Kinesis Data Firehose delivery
stream.
     *
     * @param records           a list of maps representing the records to be
sent
     * @param batchSize         the maximum number of records to include in each
batch
     * @param deliveryStreamName the name of the Kinesis Data Firehose delivery
stream
     * @throws IllegalArgumentException if the input parameters are invalid (null
or empty)
     * @throws RuntimeException         if there is an error putting the record
batch
     */
```

```
   public static void putRecordBatch(List<Map<String, Object>> records, int
batchSize, String deliveryStreamName) {
       if (records == null || records.isEmpty() || deliveryStreamName == null ||
deliveryStreamName.isEmpty()) {
           throw new IllegalArgumentException("Invalid input: records or
delivery stream name cannot be null/empty");
       }
       ObjectMapper objectMapper = new ObjectMapper();

       try {
           for (int i = 0; i < records.size(); i += batchSize) {
               List<Map<String, Object>> batch = records.subList(i, Math.min(i +
batchSize, records.size()));

               List<Record> batchRecords = batch.stream().map(record -> {
                   try {
                       String jsonRecord =
objectMapper.writeValueAsString(record);
                       return Record.builder()

.data(SdkBytes.fromByteArray(jsonRecord.getBytes(StandardCharsets.UTF_8)))
                           .build();
                   } catch (Exception e) {
                       throw new RuntimeException("Error creating Firehose
record", e);
                   }
               }).collect(Collectors.toList());

               PutRecordBatchRequest request = PutRecordBatchRequest.builder()
                   .deliveryStreamName(deliveryStreamName)
                   .records(batchRecords)
                   .build();

               PutRecordBatchResponse response =
getFirehoseClient().putRecordBatch(request);

               if (response.failedPutCount() > 0) {
                   response.requestResponses().stream()
                       .filter(r -> r.errorCode() != null)
                       .forEach(r -> System.err.println("Failed record: " +
r.errorMessage()));
               }
               System.out.println("Batch sent with size: " +
batchRecords.size());
```

```
            }
        } catch (Exception e) {
            throw new RuntimeException("Failed to put record batch: " +
 e.getMessage(), e);
        }
    }
```

- Untuk detail API, lihat PutRecordBatchdi Referensi AWS SDK for Java 2.x API.

Python

SDK untuk Python (Boto3)

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
class FirehoseClient:
    """
    AWS Firehose client to send records and monitor metrics.

    Attributes:
        config (object): Configuration object with delivery stream name and
 region.
        delivery_stream_name (str): Name of the Firehose delivery stream.
        region (str): AWS region for Firehose and CloudWatch clients.
        firehose (boto3.client): Boto3 Firehose client.
        cloudwatch (boto3.client): Boto3 CloudWatch client.
    """

    def __init__(self, config):
        """
        Initialize the FirehoseClient.

        Args:
            config (object): Configuration object with delivery stream name and
 region.
        """
        self.config = config
```

```
        self.delivery_stream_name = config.delivery_stream_name
        self.region = config.region
        self.firehose = boto3.client("firehose", region_name=self.region)
        self.cloudwatch = boto3.client("cloudwatch", region_name=self.region)


    @backoff.on_exception(
        backoff.expo, Exception, max_tries=5, jitter=backoff.full_jitter
    )
    def put_record_batch(self, data: list, batch_size: int = 500):
        """
        Put records in batches to Firehose with backoff and retry.

        Args:
            data (list): List of data records to be sent to Firehose.
            batch_size (int): Number of records to send in each batch. Default is
500.

        This method attempts to send records in batches to the Firehose delivery
stream.
        It retries with exponential backoff in case of exceptions.
        """
        for i in range(0, len(data), batch_size):
            batch = data[i : i + batch_size]
            record_dicts = [{"Data": json.dumps(record)} for record in batch]
            try:
                response = self.firehose.put_record_batch(
                    DeliveryStreamName=self.delivery_stream_name,
Records=record_dicts
                )
                self._log_batch_response(response, len(batch))
            except Exception as e:
                logger.info(f"Failed to send batch of {len(batch)} records.
Error: {e}")
```

- Untuk detail API, lihat PutRecordBatchdi AWS SDK for Python (Boto3) Referensi API.

Rust

SDK for Rust

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
async fn put_record_batch(
    client: &Client,
    stream: &str,
    data: Vec<Record>,
) -> Result<PutRecordBatchOutput, SdkError<PutRecordBatchError>> {
    client
        .put_record_batch()
        .delivery_stream_name(stream)
        .set_records(Some(data))
        .send()
        .await
}
```

- Untuk detail API, lihat PutRecordBatchreferensi AWS SDK for Rust API.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

```
    TRY.
        DATA(lo_result) = lo_frh->putrecordbatch(
          iv_deliverystreamname = iv_deliv_stream_name
          it_records            = it_records ).
```

```
        DATA(lv_failed_count) = lo_result->get_failedputcount( ).

        IF lv_failed_count > 0.
          MESSAGE |{ lv_failed_count } records failed to send.| TYPE 'I'.
        ELSE.
          MESSAGE 'All records sent successfully to Firehose delivery stream.'
TYPE 'I'.
        ENDIF.
      CATCH /aws1/cx_frhresourcenotfoundex.
        MESSAGE 'Delivery stream not found.' TYPE 'E'.
      CATCH /aws1/cx_frhinvalidargumentex.
        MESSAGE 'Invalid argument provided.' TYPE 'E'.
      CATCH /aws1/cx_frhserviceunavailex.
        MESSAGE 'Service temporarily unavailable.' TYPE 'E'.
    ENDTRY.
```

- Untuk detail API, lihat PutRecordBatchdi AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan Firehose dengan SDK AWS. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Skenario untuk Firehose menggunakan AWS SDKs

Contoh kode berikut menunjukkan kepada Anda cara menerapkan skenario umum di Firehose dengan. AWS SDKs Skenario ini menunjukkan kepada Anda cara menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam Firehose atau digabungkan dengan yang lain. Layanan AWS Setiap skenario menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode.

Skenario menargetkan tingkat pengalaman menengah untuk membantu Anda memahami tindakan layanan dalam konteks.

Contoh

- Menggunakan Amazon Data Firehose untuk memproses catatan individu dan batch

Menggunakan Amazon Data Firehose untuk memproses catatan individu dan batch

Contoh kode berikut menunjukkan cara menggunakan Firehose untuk memproses catatan individu dan batch.

Java

SDK untuk Java 2.x

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

Contoh ini menempatkan catatan individu dan batch ke Firehose.

```
/**
 * Amazon Firehose Scenario example using Java V2 SDK.
 *
 * Demonstrates individual and batch record processing,
 * and monitoring Firehose delivery stream metrics.
 */
public class FirehoseScenario {

    private static FirehoseClient firehoseClient;
    private static CloudWatchClient cloudWatchClient;

    public static void main(String[] args) {
        final String usage = """
                Usage:
                    <deliveryStreamName>
                Where:
                    deliveryStreamName - The Firehose delivery stream name.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            return;
        }
```

```
        String deliveryStreamName = args[0];

        try {
            // Read and parse sample data.
            String jsonContent = readJsonFile("sample_records.json");
            ObjectMapper objectMapper = new ObjectMapper();
            List<Map<String, Object>> sampleData =
objectMapper.readValue(jsonContent, new TypeReference<>() {});

            // Process individual records.
            System.out.println("Processing individual records...");
            sampleData.subList(0, 100).forEach(record -> {
                try {
                    putRecord(record, deliveryStreamName);
                } catch (Exception e) {
                    System.err.println("Error processing record: " +
e.getMessage());
                }
            });

            // Monitor metrics.
            monitorMetrics(deliveryStreamName);

            // Process batch records.
            System.out.println("Processing batch records...");
            putRecordBatch(sampleData.subList(100, sampleData.size()), 500,
deliveryStreamName);
            monitorMetrics(deliveryStreamName);

        } catch (Exception e) {
            System.err.println("Scenario failed: " + e.getMessage());
        } finally {
            closeClients();
        }
    }

    private static FirehoseClient getFirehoseClient() {
        if (firehoseClient == null) {
            firehoseClient = FirehoseClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return firehoseClient;
    }
```

```
   private static CloudWatchClient getCloudWatchClient() {
       if (cloudWatchClient == null) {
           cloudWatchClient = CloudWatchClient.builder()
                   .region(Region.US_EAST_1)
                   .build();
       }
       return cloudWatchClient;
   }

   /**
    * Puts a record to the specified Amazon Kinesis Data Firehose delivery
stream.
    *
    * @param record The record to be put to the delivery stream. The record must
be a {@link Map} of String keys and Object values.
    * @param deliveryStreamName The name of the Amazon Kinesis Data Firehose
delivery stream to which the record should be put.
    * @throws IllegalArgumentException if the input record or delivery stream
name is null or empty.
    * @throws RuntimeException if there is an error putting the record to the
delivery stream.
    */
   public static void putRecord(Map<String, Object> record, String
deliveryStreamName) {
       if (record == null || deliveryStreamName == null ||
deliveryStreamName.isEmpty()) {
           throw new IllegalArgumentException("Invalid input: record or delivery
stream name cannot be null/empty");
       }
       try {
           String jsonRecord = new ObjectMapper().writeValueAsString(record);
           Record firehoseRecord = Record.builder()

.data(SdkBytes.fromByteArray(jsonRecord.getBytes(StandardCharsets.UTF_8)))
               .build();

           PutRecordRequest putRecordRequest = PutRecordRequest.builder()
               .deliveryStreamName(deliveryStreamName)
               .record(firehoseRecord)
               .build();

           getFirehoseClient().putRecord(putRecordRequest);
           System.out.println("Record sent: " + jsonRecord);
```

```
        } catch (Exception e) {
            throw new RuntimeException("Failed to put record: " + e.getMessage(),
e);
        }
    }


    /**
     * Puts a batch of records to an Amazon Kinesis Data Firehose delivery
stream.
     *
     * @param records           a list of maps representing the records to be
sent
     * @param batchSize         the maximum number of records to include in each
batch
     * @param deliveryStreamName the name of the Kinesis Data Firehose delivery
stream
     * @throws IllegalArgumentException if the input parameters are invalid (null
or empty)
     * @throws RuntimeException         if there is an error putting the record
batch
     */
    public static void putRecordBatch(List<Map<String, Object>> records, int
batchSize, String deliveryStreamName) {
        if (records == null || records.isEmpty() || deliveryStreamName == null ||
deliveryStreamName.isEmpty()) {
            throw new IllegalArgumentException("Invalid input: records or
delivery stream name cannot be null/empty");
        }
        ObjectMapper objectMapper = new ObjectMapper();

        try {
            for (int i = 0; i < records.size(); i += batchSize) {
                List<Map<String, Object>> batch = records.subList(i, Math.min(i +
batchSize, records.size()));

                List<Record> batchRecords = batch.stream().map(record -> {
                    try {
                        String jsonRecord =
objectMapper.writeValueAsString(record);
                        return Record.builder()

.data(SdkBytes.fromByteArray(jsonRecord.getBytes(StandardCharsets.UTF_8)))
                            .build();
```

```
                    } catch (Exception e) {
                        throw new RuntimeException("Error creating Firehose
record", e);
                    }
                }).collect(Collectors.toList());

                PutRecordBatchRequest request = PutRecordBatchRequest.builder()
                    .deliveryStreamName(deliveryStreamName)
                    .records(batchRecords)
                    .build();

                PutRecordBatchResponse response =
getFirehoseClient().putRecordBatch(request);

                if (response.failedPutCount() > 0) {
                    response.requestResponses().stream()
                        .filter(r -> r.errorCode() != null)
                        .forEach(r -> System.err.println("Failed record: " +
r.errorMessage()));
                }
                System.out.println("Batch sent with size: " +
batchRecords.size());
            }
        } catch (Exception e) {
            throw new RuntimeException("Failed to put record batch: " +
e.getMessage(), e);
        }
    }

    public static void monitorMetrics(String deliveryStreamName) {
        Instant endTime = Instant.now();
        Instant startTime = endTime.minusSeconds(600);

        List<String> metrics = List.of("IncomingBytes", "IncomingRecords",
"FailedPutCount");
        metrics.forEach(metric -> monitorMetric(metric, startTime, endTime,
deliveryStreamName));
    }

    private static void monitorMetric(String metricName, Instant startTime,
Instant endTime, String deliveryStreamName) {
        try {
            GetMetricStatisticsRequest request =
GetMetricStatisticsRequest.builder()
```

```
                .namespace("AWS/Firehose")
                .metricName(metricName)

 .dimensions(Dimension.builder().name("DeliveryStreamName").value(deliveryStreamName).bui
                .startTime(startTime)
                .endTime(endTime)
                .period(60)
                .statistics(Statistic.SUM)
                .build();

            GetMetricStatisticsResponse response =
 getCloudWatchClient().getMetricStatistics(request);
            double totalSum =
 response.datapoints().stream().mapToDouble(Datapoint::sum).sum();
            System.out.println(metricName + ": " + totalSum);
        } catch (Exception e) {
            System.err.println("Failed to monitor metric " + metricName + ": " +
 e.getMessage());
        }
    }

    public static String readJsonFile(String fileName) throws IOException {
        try (InputStream inputStream =
 FirehoseScenario.class.getResourceAsStream("/" + fileName);
             Scanner scanner = new Scanner(inputStream, StandardCharsets.UTF_8))
 {
            return scanner.useDelimiter("\\\\A").next();
        } catch (Exception e) {
            throw new RuntimeException("Error reading file: " + fileName, e);
        }
    }

    private static void closeClients() {
        try {
            if (firehoseClient != null) firehoseClient.close();
            if (cloudWatchClient != null) cloudWatchClient.close();
        } catch (Exception e) {
            System.err.println("Error closing clients: " + e.getMessage());
        }
    }
}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for Java 2.x .

- PutRecord
- PutRecordBatch

Python

SDK untuk Python (Boto3)

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

Skrip ini menempatkan catatan individu dan batch ke Firehose.

```
import json
import logging
import random
from datetime import datetime, timedelta

import backoff
import boto3

from config import get_config


def load_sample_data(path: str) -> dict:
    """
    Load sample data from a JSON file.

    Args:
        path (str): The file path to the JSON file containing sample data.

    Returns:
        dict: The loaded sample data as a dictionary.
    """
    with open(path, "r") as f:
        return json.load(f)


# Configure logging
```

```
logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)


class FirehoseClient:
    """
    AWS Firehose client to send records and monitor metrics.

    Attributes:
        config (object): Configuration object with delivery stream name and
 region.
        delivery_stream_name (str): Name of the Firehose delivery stream.
        region (str): AWS region for Firehose and CloudWatch clients.
        firehose (boto3.client): Boto3 Firehose client.
        cloudwatch (boto3.client): Boto3 CloudWatch client.
    """

    def __init__(self, config):
        """
        Initialize the FirehoseClient.

        Args:
            config (object): Configuration object with delivery stream name and
 region.
        """
        self.config = config
        self.delivery_stream_name = config.delivery_stream_name
        self.region = config.region
        self.firehose = boto3.client("firehose", region_name=self.region)
        self.cloudwatch = boto3.client("cloudwatch", region_name=self.region)


    @backoff.on_exception(
        backoff.expo, Exception, max_tries=5, jitter=backoff.full_jitter
    )
    def put_record(self, record: dict):
        """
        Put individual records to Firehose with backoff and retry.

        Args:
            record (dict): The data record to be sent to Firehose.

        This method attempts to send an individual record to the Firehose
 delivery stream.
```

```
        It retries with exponential backoff in case of exceptions.
        """
        try:
            entry = self._create_record_entry(record)
            response = self.firehose.put_record(
                DeliveryStreamName=self.delivery_stream_name, Record=entry
            )
            self._log_response(response, entry)
        except Exception:
            logger.info(f"Fail record: {record}.")
            raise


    @backoff.on_exception(
        backoff.expo, Exception, max_tries=5, jitter=backoff.full_jitter
    )
    def put_record_batch(self, data: list, batch_size: int = 500):
        """
        Put records in batches to Firehose with backoff and retry.

        Args:
            data (list): List of data records to be sent to Firehose.
            batch_size (int): Number of records to send in each batch. Default is
500.

        This method attempts to send records in batches to the Firehose delivery
stream.
        It retries with exponential backoff in case of exceptions.
        """
        for i in range(0, len(data), batch_size):
            batch = data[i : i + batch_size]
            record_dicts = [{"Data": json.dumps(record)} for record in batch]
            try:
                response = self.firehose.put_record_batch(
                    DeliveryStreamName=self.delivery_stream_name,
Records=record_dicts
                )
                self._log_batch_response(response, len(batch))
            except Exception as e:
                logger.info(f"Failed to send batch of {len(batch)} records.
Error: {e}")


    def get_metric_statistics(
```

```
        self,
        metric_name: str,
        start_time: datetime,
        end_time: datetime,
        period: int,
        statistics: list = ["Sum"],
    ) -> list:
        """
        Retrieve metric statistics from CloudWatch.

        Args:
            metric_name (str): The name of the metric.
            start_time (datetime): The start time for the metric statistics.
            end_time (datetime): The end time for the metric statistics.
            period (int): The granularity, in seconds, of the returned data
points.
            statistics (list): A list of statistics to retrieve. Default is
['Sum'].

        Returns:
            list: List of datapoints containing the metric statistics.
        """
        response = self.cloudwatch.get_metric_statistics(
            Namespace="AWS/Firehose",
            MetricName=metric_name,
            Dimensions=[
                {"Name": "DeliveryStreamName", "Value":
self.delivery_stream_name},
            ],
            StartTime=start_time,
            EndTime=end_time,
            Period=period,
            Statistics=statistics,
        )
        return response["Datapoints"]

    def monitor_metrics(self):
        """
        Monitor Firehose metrics for the last 5 minutes.

        This method retrieves and logs the 'IncomingBytes', 'IncomingRecords',
and 'FailedPutCount' metrics
        from CloudWatch for the last 5 minutes.
        """
```

```
        end_time = datetime.utcnow()
        start_time = end_time - timedelta(minutes=10)
        period = int((end_time - start_time).total_seconds())

        metrics = {
            "IncomingBytes": self.get_metric_statistics(
                "IncomingBytes", start_time, end_time, period
            ),
            "IncomingRecords": self.get_metric_statistics(
                "IncomingRecords", start_time, end_time, period
            ),
            "FailedPutCount": self.get_metric_statistics(
                "FailedPutCount", start_time, end_time, period
            ),
        }

        for metric, datapoints in metrics.items():
            if datapoints:
                total_sum = sum(datapoint["Sum"] for datapoint in datapoints)
                if metric == "IncomingBytes":
                    logger.info(
                        f"{metric}: {round(total_sum)} ({total_sum / (1024 *
1024):.2f} MB)"
                    )
                else:
                    logger.info(f"{metric}: {round(total_sum)}")
            else:
                logger.info(f"No data found for {metric} over the last 5
minutes")


    def _create_record_entry(self, record: dict) -> dict:
        """
        Create a record entry for Firehose.

        Args:
            record (dict): The data record to be sent.

        Returns:
            dict: The record entry formatted for Firehose.

        Raises:
            Exception: If a simulated network error occurs.
        """
```

```
        if random.random() < 0.2:
            raise Exception("Simulated network error")
        elif random.random() < 0.1:
            return {"Data": '{"malformed": "data"'}
        else:
            return {"Data": json.dumps(record)}

    def _log_response(self, response: dict, entry: dict):
        """
        Log the response from Firehose.

        Args:
            response (dict): The response from the Firehose put_record API call.
            entry (dict): The record entry that was sent.
        """
        if response["ResponseMetadata"]["HTTPStatusCode"] == 200:
            logger.info(f"Sent record: {entry}")
        else:
            logger.info(f"Fail record: {entry}")

    def _log_batch_response(self, response: dict, batch_size: int):
        """
        Log the batch response from Firehose.

        Args:
            response (dict): The response from the Firehose put_record_batch API
 call.
            batch_size (int): The number of records in the batch.
        """
        if response.get("FailedPutCount", 0) > 0:
            logger.info(
                f'Failed to send {response["FailedPutCount"]} records in batch of
 {batch_size}'
            )
        else:
            logger.info(f"Successfully sent batch of {batch_size} records")


if __name__ == "__main__":
    config = get_config()
    data = load_sample_data(config.sample_data_file)
    client = FirehoseClient(config)

    # Process the first 100 sample network records
```

```
    for record in data[:100]:
        try:
            client.put_record(record)
        except Exception as e:
            logger.info(f"Put record failed after retries and backoff: {e}")
    client.monitor_metrics()

    # Process remaining records using the batch method
    try:
        client.put_record_batch(data[100:])
    except Exception as e:
        logger.info(f"Put record batch failed after retries and backoff: {e}")
    client.monitor_metrics()
```

File ini berisi konfigurasi untuk skrip di atas.

```
class Config:
    def __init__(self):
        self.delivery_stream_name = "ENTER YOUR DELIVERY STREAM NAME HERE"
        self.region = "us-east-1"
        self.sample_data_file = (
            "../../../../../scenarios/features/firehose/resources/
sample_records.json"
        )


def get_config():
    return Config()
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK untuk Python (Boto3).
  - PutRecord
  - PutRecordBatch

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan Firehose dengan SDK AWS. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Memecahkan masalah kesalahan di Amazon Data Firehose

Jika Firehose mengalami error saat mengirimkan atau memproses data, Firehose mencoba ulang hingga durasi percobaan ulang yang dikonfigurasi berakhir. Jika durasi percobaan ulang berakhir sebelum data berhasil dikirim, Firehose mencadangkan data ke bucket cadangan S3 yang dikonfigurasi. Jika tujuannya adalah Amazon S3 dan pengiriman gagal atau jika pengiriman ke bucket S3 cadangan gagal, Firehose terus mencoba lagi hingga periode retensi berakhir.

Untuk informasi tentang melacak kesalahan pengiriman menggunakan CloudWatch, lihatthe section called “Monitor dengan CloudWatch Log”.

Direct PUT

Untuk aliran `DirectPut` Firehose, Firehose menyimpan catatan selama 24 jam. Untuk aliran Firehose yang sumber datanya adalah aliran data Kinesis, Anda dapat mengubah periode retensi seperti yang dijelaskan dalam Mengubah Periode Retensi Data. Dalam hal ini, Firehose mencoba kembali operasi berikut tanpa batas waktu:`DescribeStream`,, dan. `GetRecords` `GetShardIterator`

Jika aliran Firehose digunakan`DirectPut`, periksa `IncomingRecords` metrik `IncomingBytes` dan untuk melihat apakah ada lalu lintas masuk. Jika Anda menggunakan `PutRecord` atau `PutRecordBatch`, pastikan Anda menangkap pengecualian dan mencoba lagi. Kami merekomendasikan kebijakan percobaan ulang dengan back-off eksponensial dengan jitter dan beberapa percobaan ulang. Juga, jika Anda menggunakan `PutRecordBatch` API, pastikan kode Anda memeriksa nilai FailedPutCountdalam respons bahkan ketika panggilan API berhasil.

Kinesis Data Stream

Jika aliran Firehose menggunakan aliran data Kinesis sebagai sumbernya, periksa `IncomingBytes` dan `IncomingRecords` metrik untuk aliran data sumber. Selain itu, pastikan bahwa `DataReadFromKinesisStream.Records` metrik `DataReadFromKinesisStream.Bytes` dan dipancarkan untuk aliran Firehose.

Masalah umum

Berikut ini adalah tips pemecahan masalah untuk membantu Anda memecahkan masalah umum saat Anda bekerja dengan aliran Firehose.

Aliran Firehose tidak tersedia

Aliran Firehose tidak tersedia sebagai target untuk tindakan CloudWatch Log, CloudWatch Acara, atau AWS IoT karena beberapa AWS layanan hanya dapat mengirim pesan dan peristiwa ke aliran Firehose yang sama. Wilayah AWS Pastikan aliran Firehose Anda berada di Wilayah yang sama dengan layanan Anda yang lain.

Tidak ada data di tujuan

Jika tidak ada masalah konsumsi data dan metrik yang dipancarkan untuk aliran Firehose terlihat bagus, tetapi Anda tidak melihat data di tujuan, periksa logika pembaca. Pastikan pembaca Anda mengurai semua data dengan benar.

Metrik kesegaran data meningkat atau tidak dipancarkan

Kesegaran data adalah ukuran seberapa terkini data Anda dalam aliran Firehose Anda. Ini adalah usia catatan data tertua dalam aliran Firehose, diukur dari waktu Firehose menelan data hingga saat ini. Firehose menyediakan metrik yang dapat Anda gunakan untuk memantau kesegaran data. Untuk mengidentifikasi metrik kesegaran data untuk tujuan tertentu, lihat the section called “Monitoring dengan CloudWatch Metrik”.

Jika Anda mengaktifkan pencadangan untuk semua peristiwa atau semua dokumen, pantau dua metrik kesegaran data terpisah: satu untuk tujuan utama dan satu untuk cadangan.

Jika metrik kesegaran data tidak dipancarkan, ini berarti tidak ada pengiriman aktif untuk aliran Firehose. Hal ini terjadi ketika pengiriman data diblokir sepenuhnya atau ketika tidak ada data yang masuk.

Jika metrik kesegaran data terus meningkat, ini berarti pengiriman data tertinggal. Hal ini dapat terjadi karena salah satu alasan berikut:

- Tujuan tidak dapat menangani laju pengiriman. Jika Firehose mengalami kesalahan sementara karena lalu lintas tinggi, maka pengiriman mungkin tertinggal. Ini dapat terjadi untuk tujuan selain Amazon S3 (dapat terjadi untuk OpenSearch Layanan, Amazon Redshift, atau Splunk). Pastikan tujuan Anda memiliki kapasitas yang cukup untuk menangani lalu lintas yang masuk.
- Tujuannya lambat. Pengiriman data mungkin tertinggal jika Firehose menghadapi latensi tinggi. Pantau metrik latensi tujuan.
- Fungsi Lambda lambat. Hal ini dapat menyebabkan tingkat pengiriman data yang kurang dari tingkat konsumsi data untuk aliran Firehose. Jika memungkinkan, tingkatkan efisiensi fungsi

Lambda. Misalnya, jika fungsi melakukan IO jaringan, gunakan beberapa utas atau IO asinkron untuk meningkatkan paralelisme. Selain itu, pertimbangkan untuk meningkatkan ukuran memori fungsi Lambda sehingga alokasi CPU dapat meningkat pula. Hal ini mungkin menyebabkan pemanggilan Lambda lebih cepat. Untuk informasi tentang mengonfigurasi fungsi Lambda, lihat Mengonfigurasi AWS Fungsi Lambda.

- Terdapat kegagalan selama pengiriman data. Untuk informasi tentang cara memantau kesalahan menggunakan Amazon CloudWatch Logs, lihatthe section called “Monitor dengan CloudWatch Log”.
- Jika sumber data aliran Firehose adalah aliran data Kinesis, pelambatan mungkin terjadi. Periksa metrik `ThrottledGetRecords`, `ThrottledGetShardIterator`, dan `ThrottledDescribeStream`. Jika terdapat beberapa konsumen yang melekat pada aliran data Kinesis, pertimbangkan hal berikut:
  - Jika metrik `ThrottledGetRecords` dan `ThrottledGetShardIterator` tinggi, kami sarankan Anda meningkatkan jumlah serpihan yang disediakan untuk aliran data.
  - Jika tinggi, kami sarankan Anda menambahkan `kinesis:listshards` izin ke peran yang dikonfigurasi KinesisStreamSourceConfiguration. `ThrottledDescribeStream`
- Petunjuk buffering rendah untuk tujuan. Ini dapat meningkatkan jumlah perjalanan pulang pergi yang harus dilakukan Firehose ke tujuan, yang dapat menyebabkan pengiriman tertinggal. Pertimbangkan untuk meningkatkan nilai petunjuk buffering. Untuk informasi selengkapnya, lihat BufferingHints.
- Durasi percobaan ulang yang tinggi dapat menyebabkan pengiriman tertinggal ketika kesalahan sering terjadi. Pertimbangkan untuk mengurangi durasi percobaan ulang. Selain itu, pantau kesalahannya dan coba kurangi. Untuk informasi tentang cara memantau kesalahan menggunakan Amazon CloudWatch Logs, lihatthe section called “Monitor dengan CloudWatch Log”.
- Jika tujuannya adalah Splunk dan `DeliveryToSplunk.DataFreshness` tinggi tetapi `DeliveryToSplunk.Success` terlihat baik, klaster Splunk mungkin sedang sibuk. Kosongkan klaster Splunk jika memungkinkan. Atau, hubungi AWS Support dan minta peningkatan jumlah saluran yang Firehose gunakan untuk berkomunikasi dengan cluster Splunk.

Konversi format rekaman ke Apache Parquet gagal

Ini terjadi jika Anda mengambil data DynamoDB yang menyertakan `Set` jenisnya, mengalirkannya melalui Lambda ke aliran Firehose, dan menggunakan file untuk mengonversi format rekaman ke Apache AWS Glue Data Catalog Parquet.

Ketika AWS Glue crawler mengindeks tipe data set DynamoDB (`StringSet`,, dan`BinarySet`)`NumberSet`, crawler menyimpannya dalam katalog data sebagai`SET<STRING>`,, dan, masing-masing. `SET<BIGINT>` `SET<BINARY>` Namun, untuk Firehose untuk mengonversi catatan data ke format Apache Parquet, itu memerlukan tipe data Apache Hive. Karena tipe set bukan tipe data Apache Hive yang valid, konversi gagal. Agar konversi bekerja, perbarui katalog data dengan tipe data Apache Hive. Anda dapat melakukannya dengan mengubah `set` ke `array` pada katalog data.

Untuk mengubah satu atau beberapa tipe data dari **`set`** ke **`array`** dalam AWS Glue katalog data

1. Masuk ke Konsol Manajemen AWS dan buka AWS Glue konsol di https://console.aws.amazon.com/glue/.
2. Di panel sebelah kiri, di bawah judul Katalog data, pilih Tabel.
3. Dalam daftar tabel, pilih nama tabel di mana Anda perlu mengubah satu atau lebih tipe data. Langkah ini akan membawa Anda ke halaman detail untuk tabel tersebut.
4. Pilih tombol Edit skema di pojok kanan atas halaman detail.
5. Pada kolom Tipe data, pilih tipe data `set` yang pertama.
6. Pada daftar tarik turun Tipe kolom, ubah tipe dari `set` ke `array`.
7. Di ArraySchemabidang, masukkan`array<string>`,`array<int>`, atau`array<binary>`, tergantung pada jenis data yang sesuai untuk skenario Anda.
8. Pilih Perbarui.
9. Ulangi langkah sebelumnya untuk mengonversi tipe `set` lainnya menjadi tipe `array`.
10. Pilih Simpan.

Bidang yang hilang untuk objek yang diubah untuk Lambda

Bila Anda menggunakan transformasi data Lambda untuk mengubah data JSON ke objek Parquet, beberapa bidang mungkin hilang setelah transformasi. Itu terjadi jika objek JSON Anda memiliki huruf kapital dan sensitivitas huruf besar diatur`false`, yang dapat menyebabkan ketidakcocokan pada kunci JSON setelah transformasi data yang menyebabkan data hilang pada objek Parket yang dihasilkan di ember s3.

Untuk memperbaikinya, pastikan konfigurasi selang memiliki `deserializationOption: case.insensitive` set ke `true` sehingga kunci JSON cocok setelah transformasi.

Pemecahan Masalah Amazon S3

Periksa yang berikut ini jika data tidak terkirim ke bucket Amazon Simple Storage Service (Amazon S3) Anda.

- Periksa Firehose `IncomingBytes` dan `IncomingRecords` metrik untuk memastikan data berhasil dikirim ke aliran Firehose Anda. Untuk informasi selengkapnya, lihat Pantau Amazon Data Firehose dengan metrik CloudWatch.
- Jika transformasi data dengan Lambda diaktifkan, periksa `ExecuteProcessingSuccess` metrik Firehose untuk memastikan Firehose telah mencoba menjalankan fungsi Lambda Anda. Untuk informasi selengkapnya, lihat Pantau Amazon Data Firehose dengan metrik CloudWatch.
- Periksa `DeliveryToS3.Success` metrik Firehose untuk memastikan Firehose telah mencoba memasukkan data ke bucket Amazon S3 Anda. Untuk informasi selengkapnya, lihat Pantau Amazon Data Firehose dengan metrik CloudWatch.
- Aktifkan pencatatan galat jika belum diaktifkan, dan periksa apakah ada kegagalan pengiriman pada log kesalahan. Untuk informasi selengkapnya, lihat Pantau Amazon Data Firehose Menggunakan Log CloudWatch.
- Jika Anda melihat pesan kesalahan di log yang mengatakan “Firehose ditemui InternalServerError saat memanggil layanan Amazon S3. Operasi akan dicoba lagi; jika kesalahan berlanjut, silakan hubungi S3 untuk resolusi.” , bisa jadi karena peningkatan yang signifikan dalam tingkat permintaan pada satu partisi di S3. Anda dapat mengoptimalkan pola desain awalan S3 untuk mengurangi masalah. Untuk informasi selengkapnya, lihat Pola desain praktik terbaik: mengoptimalkan kinerja Amazon S3. Jika ini tidak menyelesaikan masalah, hubungi AWS Support untuk bantuan lebih lanjut.
- Pastikan bucket Amazon S3 yang ditentukan dalam aliran Firehose Anda masih ada.
- Jika transformasi data dengan Lambda diaktifkan, pastikan fungsi Lambda yang ditentukan dalam aliran Firehose Anda masih ada.
- Pastikan bahwa peran IAM yang ditentukan dalam aliran Firehose Anda memiliki akses ke bucket S3 dan fungsi Lambda Anda (jika transformasi data diaktifkan). Juga, pastikan bahwa peran IAM memiliki akses ke grup CloudWatch log dan aliran log untuk memeriksa log kesalahan. Untuk informasi selengkapnya, lihat Berikan akses Firehose ke tujuan Amazon S3.
- Jika Anda menggunakan transformasi data, pastikan fungsi Lambda Anda tidak pernah mengembalikan respons yang ukuran muatannya melebihi 6 MB. Untuk informasi selengkapnya, lihat FirehoseData Transformasi Data Amazon.

Memecahkan Masalah Amazon Redshift

Periksa hal berikut jika data tidak dikirimkan ke klaster yang disediakan Amazon Redshift atau grup kerja Amazon Redshift Tanpa Server.

Data dikirim ke bucket S3 Anda sebelum memuat ke Amazon Redshift. Jika data tidak terkirim ke bucket S3 Anda, lihat Pemecahan Masalah Amazon S3.

- Periksa `DeliveryToRedshift.Success` metrik Firehose untuk memastikan Firehose telah mencoba menyalin data dari bucket S3 Anda ke cluster yang disediakan Amazon Redshift atau workgroup Amazon Redshift Serverless. Untuk informasi selengkapnya, lihat Pantau Amazon Data Firehose dengan metrik CloudWatch.
- Aktifkan pencatatan galat jika belum diaktifkan, dan periksa apakah ada kegagalan pengiriman pada log kesalahan. Untuk informasi selengkapnya, lihat Pantau Amazon Data Firehose Menggunakan Log CloudWatch.
- Periksa `STL_CONNECTION_LOG` tabel Amazon Redshift untuk melihat apakah Firehose dapat membuat koneksi berhasil. Dalam tabel ini, Anda harus dapat melihat koneksi dan statusnya berdasarkan nama pengguna. Untuk informasi selengkapnya, lihat `STL_CONNECTION_LOG` dalam Panduan Developer Basis Data Amazon Redshift.
- Jika pemeriksaan sebelumnya menunjukkan bahwa koneksi sedang dibuat, periksa tabel `STL_LOAD_ERRORS` Amazon Redshift untuk memverifikasi alasan kegagalan COPY. Untuk informasi selengkapnya, lihat `STL_LOAD_ERRORS` dalam Panduan Developer Basis Data Amazon Redshift.
- Pastikan konfigurasi Amazon Redshift di aliran Firehose Anda akurat dan valid.
- Pastikan peran IAM yang ditentukan dalam aliran Firehose Anda dapat mengakses bucket S3 tempat Amazon Redshift menyalin data, dan juga fungsi Lambda untuk transformasi data (jika transformasi data diaktifkan). Juga, pastikan bahwa peran IAM memiliki akses ke grup CloudWatch log dan aliran log untuk memeriksa log kesalahan. Untuk informasi selengkapnya, lihat Berikan akses Firehose ke tujuan Amazon Redshift.
- Jika klaster yang disediakan Amazon Redshift atau workgroup Amazon Redshift Serverless Anda berada di cloud pribadi virtual (VPC), pastikan klaster mengizinkan akses dari alamat IP Firehose. Untuk informasi selengkapnya, lihat Berikan akses Firehose ke tujuan Amazon Redshift.
- Pastikan klaster yang disediakan Amazon Redshift atau workgroup Amazon Redshift Tanpa Server tersedia untuk umum.

- Jika Anda menggunakan transformasi data, pastikan fungsi Lambda Anda tidak pernah mengembalikan respons yang ukuran muatannya melebihi 6 MB. Untuk informasi selengkapnya, lihat FirehoseData Transformasi Data Amazon.

Memecahkan Masalah Layanan Amazon OpenSearch

Periksa hal berikut jika data tidak dikirimkan ke domain OpenSearch Layanan Anda.

Data dapat dicadangkan ke bucket Amazon S3 Anda secara bersamaan. Jika data tidak terkirim ke bucket S3 Anda, lihat Pemecahan Masalah Amazon S3.

- Periksa Firehose `IncomingBytes` dan `IncomingRecords` metrik untuk memastikan data berhasil dikirim ke aliran Firehose Anda. Untuk informasi selengkapnya, lihat Pantau Amazon Data Firehose dengan metrik CloudWatch.
- Jika transformasi data dengan Lambda diaktifkan, periksa `ExecuteProcessingSuccess` metrik Firehose untuk memastikan Firehose telah mencoba menjalankan fungsi Lambda Anda. Untuk informasi selengkapnya, lihat Pantau Amazon Data Firehose dengan metrik CloudWatch.
- Periksa `DeliveryToAmazonOpenSearchService.Success` metrik Firehose untuk memastikan Firehose telah mencoba mengindeks data ke kluster Service. OpenSearch Untuk informasi selengkapnya, lihat Pantau Amazon Data Firehose dengan metrik CloudWatch.
- Aktifkan pencatatan galat jika belum diaktifkan, dan periksa apakah ada kegagalan pengiriman pada log kesalahan. Untuk informasi selengkapnya, lihat Pantau Amazon Data Firehose Menggunakan Log CloudWatch.
- Pastikan bahwa konfigurasi OpenSearch Layanan di aliran Firehose Anda akurat dan valid.
- Jika transformasi data dengan Lambda diaktifkan, pastikan fungsi Lambda yang ditentukan dalam aliran Firehose Anda masih ada. Juga, pastikan bahwa peran IAM memiliki akses ke grup CloudWatch log dan aliran log untuk memeriksa log kesalahan. Untuk informasi selengkapnya, lihat Pemberian FirehoseAccess ke Tujuan OpenSearch Layanan Publik.
- Pastikan peran IAM yang ditentukan dalam aliran Firehose dapat mengakses kluster Layanan, OpenSearch bucket cadangan S3, dan fungsi Lambda (jika transformasi data diaktifkan). Juga, pastikan bahwa peran IAM memiliki akses ke grup CloudWatch log dan aliran log untuk memeriksa log kesalahan. Untuk informasi selengkapnya, lihat Berikan akses Firehose ke tujuan Layanan publik OpenSearch.

- Jika Anda menggunakan transformasi data, pastikan fungsi Lambda Anda tidak pernah mengembalikan respons yang ukuran muatannya melebihi 6 MB. Untuk informasi selengkapnya, lihat FirehoseData Transformasi Data Amazon.
- Amazon Data Firehosesaat ini tidak mendukung pengiriman Log CloudWatch ke tujuan OpenSearch Layanan Amazon karena Amazon CloudWatch menggabungkan beberapa peristiwa log menjadi satu catatan Firehose dan Layanan OpenSearch Amazon tidak dapat menerima beberapa peristiwa log dalam satu catatan. Sebagai alternatif, Anda dapat mempertimbangkan Menggunakan filter langganan untuk OpenSearch Layanan Amazon di CloudWatch Log.

Pemecahan Masalah Splunk

Periksa yang berikut ini jika data tidak terkirim ke titik akhir Splunk Anda.

- Jika platform Splunk Anda ada di VPC, pastikan Firehose dapat mengaksesnya. Untuk informasi selengkapnya, lihat Mengakses Splunk di VPC.
- Jika Anda menggunakan AWS load balancer, pastikan bahwa itu adalah Classic Load Balancer atau Application Load Balancer. Selain itu, aktifkan sesi lengket berbasis durasi dengan kedaluwarsa cookie dinonaktifkan untuk Classic Load Balancer dan kedaluwarsa diatur ke maksimum (7 hari) untuk Application Load Balancer. Untuk informasi tentang cara melakukannya, lihat Duration-Based Session Stickiness for Classic Load Balancer atau Application Load Balancer.
- Tinjau persyaratan platform Splunk. Add-on Splunk untuk Firehose memerlukan platform Splunk versi 6.6.X atau yang lebih baru. Untuk informasi lebih lanjut, lihat Splunk Add-on untuk Amazon Kinesis Firehose.
- Jika Anda memiliki proxy (Elastic Load Balancing atau lainnya) antara Firehose dan node HTTP Event Collector (HEC), aktifkan sesi lengket untuk mendukung pengakuan HEC (ACK).
- Pastikan Anda menggunakan token HEC yang valid.
- Pastikan token HEC diaktifkan.
- Periksa apakah data yang Anda kirim ke Splunk diformat dengan benar. Untuk informasi selengkapnya, lihat Format peristiwa untuk HTTP Event Collector.
- Pastikan token HEC dan peristiwa input dikonfigurasi dengan indeks yang valid.
- Ketika pengunggahan ke Splunk gagal karena kesalahan server dari simpul HEC, permintaan secara otomatis dicoba lagi. Jika semua percobaan ulang gagal, data akan dicadangkan ke Amazon S3. Periksa apakah data Anda muncul di Amazon S3, yang merupakan indikasi kegagalan tersebut.

- Pastikan Anda mengaktifkan pengakuan pengindeks pada token HEC Anda.
- Tingkatkan nilai `HECAcknowledgmentTimeoutInSeconds` dalam konfigurasi tujuan Splunk aliran Firehose Anda.
- Tingkatkan nilai `DurationInSeconds` under `RetryOptions` dalam konfigurasi tujuan Splunk aliran Firehose Anda.
- Periksa kondisi HEC Anda. Mengaktifkan pemeriksaan kesehatan adalah pra-requsite untuk transfer data ke Splunk.
- Jika Anda menggunakan transformasi data, pastikan fungsi Lambda Anda tidak pernah mengembalikan respons yang ukuran muatannya melebihi 6 MB. Untuk informasi selengkapnya, lihat Transformasi Data Firehose Data Amazon.
- Pastikan parameter Splunk yang bernama `ackIdleCleanup` diatur ke `true`. Nilainya secara default salah. Untuk mengatur parameter ini ke `true`, lakukan hal berikut:
  - Untuk deployment Splunk Cloud terkelola, kirimkan kasus menggunakan portal dukungan Splunk. Dalam hal ini, mintalah dukungan Splunk untuk mengaktifkan HTTP event collector, atur `ackIdleCleanup` ke `true` dalam `inputs.conf`, dan buat atau ubah penyeimbang beban untuk digunakan dengan pengaya ini.
  - Untuk deployment Splunk Enterprise terdistribusi, atur parameter `ackIdleCleanup` ke nilai benar dalam file `inputs.conf`. Untuk pengguna *nix, file ini terletak di `$SPLUNK_HOME/etc/apps/splunk_httpinput/local/`. Untuk pengguna Windows, file ini berada di `%SPLUNK_HOME%\etc\apps\splunk_httpinput\local\`.
  - Untuk deployment Splunk Enterprise instans tunggal, atur parameter `ackIdleCleanup` ke nilai `true` dalam file `inputs.conf`. Untuk pengguna *nix, file ini terletak di `$SPLUNK_HOME/etc/apps/splunk_httpinput/local/`. Untuk pengguna Windows, file ini berada di `%SPLUNK_HOME%\etc\apps\splunk_httpinput\local\`.
- Pastikan bahwa peran IAM yang ditentukan dalam aliran Firehose Anda dapat mengakses bucket cadangan S3 dan fungsi Lambda untuk transformasi data (jika transformasi data diaktifkan). Juga, pastikan bahwa peran IAM memiliki akses ke grup CloudWatch Log dan aliran log untuk memeriksa log kesalahan. Untuk informasi selengkapnya, lihat Grant FirehoseAccess to a Splunk Destination.
- Untuk mengarahkan ulang data yang dikirim ke bucket kesalahan S3 (cadangan S3) kembali ke Splunk, ikuti langkah-langkah yang disebutkan dalam dokumentasi Splunk.
- Lihat Memecahkan Masalah Splunk Add-on untuk Amazon Kinesis Firehose.

Pemecahan Masalah Kepingan Salju

Bagian ini menjelaskan langkah-langkah pemecahan masalah umum saat menggunakan Snowflake sebagai tujuan

Pembuatan aliran Firehose gagal

Jika pembuatan aliran Firehose gagal untuk aliran yang mengirimkan data ke Cluster PrivateLink-enabled Kepingan Salju, ini menunjukkan bahwa aliran tersebut tidak dapat VPCE-ID dijangkau oleh Firehose. Ini bisa disebabkan oleh salah satu alasan berikut:

- Salah VPCE-ID. Konfirmasikan bahwa tidak ada kesalahan tipografi.
- Firehose tidak mendukung URL Snowflake tanpa wilayah dalam pratinjau. Berikan URL menggunakan Pencari Akun Snowflake. Lihat dokumentasi Snowflake untuk lebih jelasnya.
- Konfirmasikan bahwa aliran Firehose dibuat di AWS Wilayah yang sama dengan Wilayah Kepingan Salju.
- Jika masalah berlanjut, hubungi AWS dukungan.

Kegagalan pengiriman

Periksa hal berikut jika data tidak dikirim ke tabel Snowflake Anda. Data gagal pengiriman kepingan salju akan dikirim ke bucket kesalahan S3 bersama dengan kode kesalahan dan pesan kesalahan yang sesuai dengan muatan. Berikut ini adalah beberapa skenario kesalahan umum. Untuk seluruh daftar kode kesalahan, lihatKesalahan pengiriman data kepingan salju.

- Kode kesalahan: Snowflake.DefaultRoleMissing: Menunjukkan bahwa peran kepingan salju tidak dikonfigurasi saat membuat aliran Firehose. Jika peran Snowflake tidak dikonfigurasi, pastikan Anda menetapkan peran default ke pengguna Snowflake yang ditentukan.
- Kode kesalahan: Snowflake.ExtraColumns: Menunjukkan bahwa insert ke Snowflake ditolak karena kolom tambahan dalam muatan input. Kolom yang tidak ada dalam tabel tidak boleh ditentukan. Perhatikan bahwa nama kolom Snowflake peka huruf besar/kecil. Jika pengiriman gagal dengan kesalahan ini meskipun kolom ada dalam tabel, pastikan bahwa kasus nama kolom di muatan input cocok dengan nama kolom yang dinyatakan dalam definisi tabel.
- Kode kesalahan: Snowflake.MissingColumns: Menunjukkan bahwa insert ke Snowflake ditolak karena kolom yang hilang dalam muatan input. Pastikan bahwa nilai ditentukan untuk semua kolom non-nullable.

- Kode kesalahan: Snowflake.InvalidInput: Ini bisa terjadi ketika Firehose gagal mengurai muatan input yang disediakan ke dalam format JSON yang valid. Pastikan payload json terbentuk dengan baik, tidak memiliki tanda kutip ganda tambahan, tanda kutip, karakter pelarian, dll. Saat ini Firehose hanya mendukung item JSON tunggal sebagai muatan catatan, array JSON tidak didukung.
- Kode kesalahan: Snowflake.InvalidValue: Menunjukkan bahwa pengiriman gagal karena tipe data yang salah dalam muatan input. Pastikan bahwa nilai JSON yang ditentukan dalam muatan input mematuhi tipe data yang dideklarasikan dalam definisi tabel Snowflake.
- Kode galat: Snowflake.InvalidTableType: Menunjukkan bahwa jenis tabel yang dikonfigurasi dalam aliran Firehose tidak didukung. Lihat batasan di Batasan) streaming snowpipe untuk tabel, kolom, dan tipe data yang didukung.

> **Note**
>
> Untuk alasan apa pun, jika definisi tabel atau izin peran diubah di tujuan Snowflake Anda setelah membuat aliran Firehose, Firehose dapat mendeteksi perubahan tersebut selama beberapa menit. Jika Anda melihat kesalahan pengiriman karena ini, coba hapus dan buat ulang aliran Firehose.

Memecahkan masalah jangkauan titik akhir Firehose

Jika Firehose API menemukan batas waktu, lakukan langkah-langkah berikut untuk menguji jangkauan titik akhir:

- Periksa apakah permintaan API dibuat dari host di VPC. Semua lalu lintas dari VPC memerlukan pengaturan endpoint Firehose VPC. Untuk informasi selengkapnya, lihat Menggunakan Firehose dengan. AWS PrivateLink
- Jika lalu lintas berasal dari jaringan publik atau VPC dengan endpoint Firehose VPC yang diatur di subnet tertentu, jalankan perintah berikut dari host untuk memeriksa konektivitas jaringan. Titik akhir Firehose dapat ditemukan di titik akhir dan kuota Firehose.
  - Gunakan alat seperti traceroute atau tcping untuk memeriksa apakah pengaturan jaringan sudah benar. Jika gagal, periksa pengaturan jaringan Anda:

    Contoh:

```
traceroute firehose.us-east-2.amazonaws.com
```

atau

```
tcping firehose.us-east-2.amazonaws.com 443
```

- Jika tampaknya pengaturan jaringan sudah benar dan perintah berikut gagal, periksa apakah Amazon CA (Certficate Authority) ada dalam rantai kepercayaan.

  Contoh:

```
curl firehose.us-east-2.amazonaws.com
```

Jika perintah di atas berhasil, coba API lagi untuk melihat apakah ada respons yang dikembalikan dari API.

Penyelesaian Masalah Titik Akhir HTTP

Bagian ini menjelaskan langkah-langkah pemecahan masalah umum saat menangani Amazon Data Firehose yang mengirimkan data ke tujuan titik akhir HTTP generik dan ke tujuan mitra, termasuk Datadog, Dynatrace,, MongoDB, New Relic, Splunk LogicMonitor, atau Sumo Logic. Untuk tujuan bagian ini, semua tujuan yang berlaku disebut sebagai titik akhir HTTP. Pastikan bahwa peran IAM yang ditentukan dalam aliran Firehose Anda dapat mengakses bucket cadangan S3 dan fungsi Lambda untuk transformasi data (jika transformasi data diaktifkan). Juga, pastikan bahwa peran IAM memiliki akses ke grup CloudWatch log dan aliran log untuk memeriksa log kesalahan. Untuk informasi selengkapnya, lihat Memberikan Akses Firehose ke Tujuan Titik Akhir HTTP.

> **Note**
>
> Informasi di bagian ini tidak berlaku untuk tujuan berikut: Splunk, OpenSearch Service, S3, dan Redshift.

CloudWatch Log

Sangat disarankan agar Anda mengaktifkan CloudWatch Logging for. Log hanya diterbitkan ketika ada kesalahan pengiriman ke tujuan Anda.

Pengecualian Tujuan

ErrorCode: HttpEndpoint.DestinationException

```
{
    "deliveryStreamARN": "arn:aws:firehose:us-east-1:123456789012:deliverystream/
ronald-test",
    "destination": "custom.firehose.endpoint.com...",
    "deliveryStreamVersionId": 1,
    "message": "The following response was received from the endpoint destination.
 413: {\"requestId\": \"43b8e724-dbac-4510-adb7-ef211c6044b9\", \"timestamp\":
 1598556019164, \"errorMessage\": \"Payload too large\"}",
    "errorCode": "HttpEndpoint.DestinationException",
    "processor": "arn:aws:lambda:us-east-1:379522611494:function:httpLambdaProcessing"
}
```

Pengecualian tujuan menunjukkan bahwa Firehose mampu membuat koneksi ke titik akhir Anda dan membuat permintaan HTTP, tetapi tidak menerima kode respon 200. 2xx tanggapan yang bukan 200 juga akan menghasilkan pengecualian tujuan. Amazon Data Firehose mencatat kode respons dan payload respons terpotong yang diterima dari titik akhir yang dikonfigurasi ke Log. CloudWatch Karena Amazon Data Firehose mencatat kode respons dan payload tanpa modifikasi atau interpretasi, terserah titik akhir untuk memberikan alasan yang tepat mengapa ia menolak permintaan pengiriman HTTP Amazon Data Firehose. Berikut ini adalah rekomendasi pemecahan masalah paling umum untuk pengecualian ini:

- 400: Menunjukkan bahwa Anda mengirim permintaan yang buruk karena kesalahan konfigurasi Amazon Data Firehose Anda. Pastikan Anda memiliki url, atribut umum, pengkodean konten,access key, dan petunjuk buffering yang benar untuk tujuan Anda. Lihat dokumentasi spesifik tujuan pada konfigurasi yang diperlukan.
- 401: Menunjukkan bahwa kunci akses yang Anda konfigurasikan untuk aliran Firehose Anda salah atau hilang.
- 403: Menunjukkan bahwa kunci akses yang Anda konfigurasikan untuk aliran Firehose tidak memiliki izin untuk mengirimkan data ke titik akhir yang dikonfigurasi.
- 413: Menunjukkan bahwa payload permintaan yang dikirimkan Amazon Data Firehose ke endpoint terlalu besar untuk ditangani oleh endpoint. Coba turunkan petunjuk buffering ke ukuran yang disarankan untuk tujuan Anda.

- 429: Menunjukkan bahwa Amazon Data Firehose mengirimkan permintaan pada tingkat yang lebih besar daripada yang dapat ditangani oleh tujuan. Sempurnakan petunjuk buffering Anda dengan meningkatkan waktu buffering Anda and/or meningkatkan ukuran buffering Anda (tetapi masih dalam batas tujuan Anda).
- 5xx: Menunjukkan bahwa ada masalah dengan tujuan. Layanan Amazon Data Firehose masih berfungsi dengan baik.

> **Important**
>
> Penting: Meski ini adalah rekomendasi pemecahan masalah umum, titik akhir tertentu mungkin memiliki alasan berbeda saat memberikan kode respons dan Anda perlu mengikuti rekomendasi khusus titik akhir tersebut terlebih dahulu..

Respons Tidak Valid

ErrorCode: HttpEndpoint.InvalidResponseFromDestination

```
{
    "deliveryStreamARN": "arn:aws:firehose:us-east-1:123456789012:deliverystream/
ronald-test",
    "destination": "custom.firehose.endpoint.com...",
    "deliveryStreamVersionId": 1,
    "message": "The response received from the specified endpoint is invalid.
 Contact the owner of the endpoint to resolve the issue. Response for request
 2de9e8e9-7296-47b0-bea6-9f17b133d847 is not recognized as valid JSON or has unexpected
 fields. Raw response received: 200 {\"requestId\": null}",
    "errorCode": "HttpEndpoint.InvalidResponseFromDestination",
    "processor": "arn:aws:lambda:us-east-1:379522611494:function:httpLambdaProcessing"
}
```

Pengecualian respons yang tidak valid menunjukkan bahwa Amazon Data Firehose menerima respons yang tidak valid dari tujuan titik akhir. Respons harus sesuai dengan spesifikasi respons atau Amazon Data Firehose akan menganggap upaya pengiriman gagal dan akan mengirimkan ulang data yang sama hingga durasi percobaan ulang yang dikonfigurasi terlampaui. Amazon Data Firehose memperlakukan respons yang tidak mengikuti spesifikasi respons sebagai kegagalan meskipun respons memiliki status 200. Jika Anda mengembangkan titik akhir yang kompatibel dengan Amazon Data Firehose, ikuti spesifikasi respons untuk memastikan data berhasil dikirimkan.

Berikut adalah beberapa jenis respons tidak valid yang umum dan cara memperbaikinya:

- JSON Tidak Valid atau Bidang Tak Terduga: Menunjukkan bahwa respons tidak dapat dideserialisasi dengan benar sebagai JSON atau memiliki bidang tak terduga. Pastikan respons tidak dikodekan dengan konten.
- Hilang RequestId: Menunjukkan bahwa respon tidak mengandung RequesTid.
- RequestId tidak cocok: Menunjukkan bahwa requeStyD dalam respons tidak cocok dengan RequeStyD keluar.
- Stempel Waktu Hilang: Menunjukkan bahwa respons tidak mengandung bidang stempel waktu. Bidang stempel waktu harus berisi angka dan bukan string.
- Content-Type Header Hilang: Menunjukkan bahwa respons tidak berisi header “content-type: application/json”. Tipe konten lainnya tidak diterima.

> **Important**
>
> Penting: Amazon Data Firehose hanya dapat mengirimkan data ke titik akhir yang mengikuti permintaan Firehose dan spesifikasi respons. Jika Anda mengonfigurasi tujuan Anda ke layanan pihak ketiga, pastikan Anda menggunakan titik akhir yang kompatibel dengan Amazon Data Firehose yang benar yang kemungkinan akan berbeda dari titik akhir konsumsi publik. Misalnya titik akhir Amazon Data Firehose Datadog adalah `https://aws-kinesis-http-intake.logs.datadoghq.com/` saat titik akhir publiknya. `https://api.datadoghq.com/`

Kesalahan Umum Lainnya

Kode kesalahan tambahan dan definisinya tercantum di bawah ini.

- Kode Kesalahan: HttpEndpoint.RequestTimeout - Menunjukkan bahwa titik akhir membutuhkan waktu lebih dari 3 menit untuk merespons. Jika Anda adalah pemilik tujuan, kurangi waktu respons titik akhir tujuan. Jika Anda bukan pemilik tujuan, hubungi pemilik dan tanyakan apakah ada sesuatu yang dapat dilakukan untuk menurunkan waktu respons (yaitu mengurangi petunjuk buffering sehingga data yang diproses per permintaan lebih sedikit).
- Kode Kesalahan: HttpEndpoint.ResponseTooLarge - Menunjukkan bahwa responsnya terlalu besar. Respons harus kurang dari 1 MiB termasuk headernya.

- Kode Kesalahan: HttpEndpoint.ConnectionFailed - Menunjukkan koneksi tidak dapat dibuat dengan titik akhir yang dikonfigurasi. Ini mungkin karena kesalahan ketik di url yang dikonfigurasi, titik akhir tidak dapat diakses oleh Amazon Data Firehose, atau titik akhir yang terlalu lama untuk menanggapi permintaan koneksi.
- Kode Kesalahan: HttpEndpoint.ConnectionReset - Menunjukkan koneksi dibuat tetapi diatur ulang atau ditutup sebelum waktunya oleh titik akhir.
- Kode Kesalahan: HttpEndpoint.SSLHandshakeFailure - Menunjukkan jabat tangan SSL tidak berhasil diselesaikan dengan titik akhir yang dikonfigurasi.

Pemecahan Masalah MSK Sebagai Sumber

Bagian ini menjelaskan langkah-langkah pemecahan masalah umum saat menggunakan MSK Sebagai Sumber

> **Note**
>
> Untuk mengatasi masalah pemrosesan, transformasi, atau pengiriman S3, silakan lihat bagian sebelumnya

Pembuatan selang gagal

Periksa hal berikut jika selang Anda dengan MSK As Source gagal dibuat:

- Periksa apakah cluster MSK sumber dalam keadaan aktif.
- Jika Anda menggunakan konektivitas Private, pastikan Private Link pada cluster diaktifkan.

  Jika Anda menggunakan konektivitas Publik, pastikan akses Publik di klaster diaktifkan.
- Jika Anda menggunakan Konektivitas pribadi, pastikan Anda menambahkan kebijakan berbasis sumber daya yang memungkinkan Firehose membuat Tautan Pribadi. Lihat juga: Izin lintas akun MSK.
- Pastikan bahwa peran dalam konfigurasi sumber memiliki izin untuk menyerap data dari Topik klaster.
- Pastikan grup keamanan VPC Anda mengizinkan lalu lintas masuk pada port yang digunakan oleh server bootstrap cluster.

Selang Ditangguhkan

Periksa hal berikut jika selang Anda dalam keadaan SUSPEND

- Periksa apakah cluster MSK sumber dalam keadaan aktif.
- Periksa apakah topik sumber ada. Jika topik telah dihapus dan dibuat ulang, Anda harus menghapus dan membuat ulang aliran Firehose juga.

Selang Backpresurred

Nilai DataReadFromSource.Backpressured akan menjadi 1 ketika BytesPerSecondLimit per partisi terlampaui atau bahwa aliran pengiriman normal lambat atau berhenti.

- Jika Anda menekan, BytesPerSecondLimit silakan periksa DataReadFromSource.Bytes metrik dan minta kenaikan batas.
- Periksa CloudWatch log, metrik tujuan, metrik Transformasi Data, dan metrik Konversi Format untuk mengidentifikasi kemacetan.

Kesegaran Data Salah

Kesegaran data tampaknya salah

- Firehose menghitung kesegaran data berdasarkan stempel waktu dari catatan yang dikonsumsi. Untuk memastikan bahwa stempel waktu ini direkam dengan benar ketika catatan produsen disimpan di log broker Kafka, atur konfigurasi tipe stempel waktu topik Kafka menjadi. `message.timestamp.type=LogAppendTime`

Masalah koneksi kluster MSK

Prosedur berikut menjelaskan bagaimana Anda dapat memvalidasi konektivitas ke kluster MSK. Untuk detail tentang pengaturan klien MSK Amazon, lihat Memulai menggunakan MSK Amazon di Panduan Pengembang Amazon Managed Streaming for Apache Kafka.

Untuk memvalidasi konektivitas ke kluster MSK

1. Buat instans Amazon EC2 Unix-based (sebaiknya AL2). Jika Anda hanya mengaktifkan konektivitas VPC di cluster Anda, pastikan instans EC2 Anda berjalan di VPC yang sama. SSH

ke dalam instance setelah tersedia. Untuk informasi lebih lanjut, lihat tutorial ini di Panduan Pengguna Amazon EC2.

2. Instal Java menggunakan manajer paket Yum dengan menjalankan perintah berikut. Untuk informasi selengkapnya, lihat petunjuk penginstalan di Panduan Pengguna Amazon Corretto 8.

```
sudo yum install java-1.8.0
```

3. Instal AWS klien dengan menjalankan perintah berikut.

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
```

4. Unduh versi Apache Kafka client 2.6* dengan menjalankan perintah berikut.

```
wget https://archive.apache.org/dist/kafka/2.6.2/kafka_2.12-2.6.2.tgz
tar -xzf kafka_2.12-2.6.2.tgz
```

5. Buka `kafka_2.12-2.6.2/libs` direktori, lalu jalankan perintah berikut untuk mengunduh file Amazon MSK IAM JAR.

```
wget https://github.com/aws/aws-msk-iam-auth/releases/download/v1.1.3/aws-msk-iam-
auth-1.1.3-all.jar
```

6. Buat `client.properties` file di folder bin Kafka.

7. Ganti `awsRoleArn` dengan peran ARN yang telah Anda gunakan di Firehose Anda `SourceConfiguration` dan verifikasi lokasi sertifikat. Izinkan pengguna AWS klien Anda untuk mengambil peran`awsRoleArn`. AWS pengguna klien akan mencoba untuk mengambil peran yang Anda tentukan di sini.

```
[ec2-user@ip-xx-xx-xx-xx bin]$ cat client.properties
security.protocol=SASL_SSL
sasl.mechanism=AWS_MSK_IAM
sasl.jaas.config=software.amazon.msk.auth.iam.IAMLoginModule required
 awsRoleArn="<role arn>" awsStsRegion="<region name>";
sasl.client.callback.handler.class=software.amazon.msk.auth.iam.IAMClientCallbackHandler
awsDebugCreds=true
ssl.truststore.location=/usr/lib/jvm/java-1.8.0-
openjdk-1.8.0.342.b07-1.amzn2.0.1.x86_64/jre/lib/security/cacerts
```

```
ssl.truststore.password=changeit
```

8. Jalankan perintah Kafka berikut untuk membuat daftar topik. Jika koneksi Anda bersifat publik, gunakan server Bootstrap titik akhir publik. Jika koneksi Anda bersifat pribadi, gunakan server Bootstrap endpoint pribadi.

```
bin/kafka-topics.sh --list --bootstrap-server <bootstrap servers> --command-config
 bin/client.properties
```

Jika permintaan berhasil, Anda akan melihat output yang mirip dengan contoh berikut.

```
[ec2-user@ip-xx-xx-xx-xx kafka_2.12-2.6.2]$ bin/kafka-topics.sh --list --bootstrap-
server <bootstrap servers> --command-config bin/client.properties

[xxxx-xx-xx 05:49:50,877] WARN The configuration 'awsDebugCreds' was supplied but
 isn't a known config. (org.apache.kafka.clients.admin.AdminClientConfig)
[xxxx-xx-xx 05:49:50,878] WARN The configuration 'ssl.truststore.location' was
 supplied but isn't a known config.
 (org.apache.kafka.clients.admin.AdminClientConfig)
[xxxx-xx-xx 05:49:50,878] WARN The configuration 'sasl.jaas.config' was supplied
 but isn't a known config. (org.apache.kafka.clients.admin.AdminClientConfig)
[xxxx-xx-xx 05:49:50,878] WARN The configuration
 'sasl.client.callback.handler.class' was supplied but isn't a known config.
 (org.apache.kafka.clients.admin.AdminClientConfig)
[xxxx-xx-xx 05:49:50,878] WARN The configuration 'ssl.truststore.password' was
 supplied but isn't a known config.
 (org.apache.kafka.clients.admin.AdminClientConfig)
[xxxx-xx-xx 05:50:21,629] WARN [AdminClient clientId=adminclient-1] Connection to
 node...
__amazon_msk_canary
__consumer_offsets
```

9. Jika Anda memiliki masalah dalam menjalankan skrip sebelumnya, verifikasi bahwa server bootstrap yang Anda berikan dapat dijangkau pada port yang ditentukan. Untuk melakukan ini, Anda dapat mengunduh dan menggunakan telnet atau utilitas serupa seperti yang ditunjukkan pada perintah berikut.

```
sudo yum install telnet
telnet <bootstrap servers><port>
```

Jika permintaan berhasil, Anda akan mendapatkan output berikut. Ini berarti bahwa Anda dapat terhubung ke cluster MSK Anda dalam VPC lokal Anda dan server bootstrap sehat pada port yang ditentukan.

```
Connected to ..
```

10. Jika permintaan tidak berhasil, periksa aturan masuk pada grup keamanan VPC Anda. Sebagai contoh, Anda dapat menggunakan properti berikut pada aturan masuk.

```
Type: All traffic
Port: Port used by the bootstrap server (e.g. 14001)
Source: 0.0.0.0/0
```

Coba lagi koneksi telnet seperti yang ditunjukkan pada langkah sebelumnya. Jika Anda masih tidak dapat terhubung atau koneksi Firehose Anda masih gagal, hubungi dukungan.AWS

Kuota Amazon Data Firehose

Bagian ini menjelaskan kuota saat ini, sebelumnya disebut sebagai batas, dalam Amazon Data Firehose. Kecuali ditentukan lain, masing-masing kuota berlaku untuk setiap Wilayah.

Konsol Service Quotas adalah lokasi pusat tempat Anda dapat melihat dan mengelola kuota untuk AWS layanan, dan meminta peningkatan kuota untuk banyak sumber daya yang Anda gunakan. Gunakan informasi kuota yang kami sediakan untuk mengelola AWS infrastruktur Anda. Rencanakan permintaan peningkatan kuota sebelum Anda membutuhkannya.

Untuk informasi selengkapnya, lihat titik akhir Amazon Data Firehose dan kuota di. Referensi Umum Amazon Web

Bagian berikut menunjukkan Amazon Data Firehose memiliki kuota berikut.

- Dengan Amazon MSK sebagai sumber aliran Firehose, setiap aliran Firehose memiliki kuota default MB/sec 10 throughput baca per partisi dan ukuran rekaman maksimal 10MB.
- Dengan Amazon MSK sebagai sumber aliran Firehose, ada ukuran rekaman maksimum 6 MB jika AWS Lambda diaktifkan, dan ukuran rekaman maksimum 10 MB jika Lambda dinonaktifkan. AWS Lambda membatasi catatan masuknya menjadi 6 MB, dan Amazon Data Firehose meneruskan catatan di atas 6Mb ke bucket S3 kesalahan. Jika Lambda dinonaktifkan, Firehose membatasi catatan masuknya menjadi 10 MB. Jika Amazon Data Firehose menerima ukuran rekaman dari Amazon MSK yang lebih besar dari 10 MB, Amazon Data Firehose mengirimkan catatan ini ke bucket kesalahan S3 dan memancarkan metrik Cloudwatch ke akun Anda. Untuk informasi selengkapnya tentang batas AWS Lambda, lihat Kuota Lambda.
- Saat partisi dinamis pada aliran Firehose diaktifkan, ada kuota default 500 partisi aktif yang dapat dibuat untuk aliran Firehose tersebut. Jumlah partisi aktif adalah jumlah total partisi aktif dalam buffer pengiriman. Misalnya, jika kueri partisi dinamis membangun 3 partisi per detik dan Anda memiliki konfigurasi petunjuk buffer yang memicu pengiriman setiap 60 detik, maka, rata-rata, Anda akan memiliki 180 partisi aktif. Setelah data dikirim dalam partisi, maka partisi ini tidak lagi aktif. Jika Anda membutuhkan lebih banyak partisi, Anda dapat membuat lebih banyak aliran Firehose dan mendistribusikan partisi aktif di dalamnya.
- Saat partisi dinamis pada aliran Firehose diaktifkan, throughput maksimal 1 GB per detik didukung untuk setiap partisi aktif.
- Setiap akun akan memiliki kuota berikut untuk jumlah aliran Firehose per Wilayah:
  - AS Timur (Virginia N.), AS Timur (Ohio), AS Barat (Oregon), Eropa (Irlandia), Asia Pasifik (Tokyo): 5.000 aliran Firehose

- Eropa (Frankfurt), Eropa (London), Asia Pasifik (Singapura), Asia Pasifik (Sydney), Asia Pasifik (Seoul), Asia Pasifik (Mumbai), (AS-Barat), Kanada AWS GovCloud (Barat), Kanada (Tengah): 2.000 aliran Firehose
- Eropa (Paris), Eropa (Milan), Eropa (Stockholm), Asia Pasifik (Hong Kong), Asia Pasifik (Osaka), Amerika Selatan (Sao Paulo), Tiongkok (Ningxia), Tiongkok (Beijing), Timur Tengah (Bahrain), (AS-Timur) AWS GovCloud , Afrika (Cape Town): 500 aliran Firehose
- Eropa (Zurich), Eropa (Spanyol), Asia Pasifik (Hyderabad), Asia Pasifik (Jakarta), Asia Pasifik (Melbourne), Timur Tengah (UEA), Israel (Tel Aviv), Kanada Barat (Calgary), Kanada (Tengah), Asia Pasifik (Malaysia), Asia Pasifik (Thailand), Meksiko (Tengah): 100 aliran Firehose
- Jika Anda melebihi jumlah ini, panggilan ke CreateDeliveryStream akan menghasilkan pengecualian `LimitExceededException`. Untuk menambah kuota ini, Anda dapat menggunakan Service Quotas jika tersedia di Wilayah Anda. Untuk informasi tentang penggunaan Service Quotas, lihat Meminta Peningkatan Kuota.

- Ketika Direct PUT dikonfigurasi sebagai sumber data, setiap aliran Firehose menyediakan kuota gabungan untuk PutRecorddan permintaan berikut: PutRecordBatch
  - Untuk AS Timur (Virginia N.), AS Barat (Oregon), dan Eropa (Irlandia): 500.000. records/second, 2,000 requests/second, and 5 MiB/second
  - Untuk lainnya Wilayah AWS: 100.000records/second, 1,000 requests/second, and 1 MiB/second.

  Jika aliran PUT Langsung mengalami pelambatan karena volume konsumsi data yang lebih tinggi yang melebihi kapasitas throughput aliran Firehose, Amazon Data Firehose secara otomatis meningkatkan batas throughput aliran hingga throttling terisi. Bergantung pada peningkatan throughput dan throttling, Firehose mungkin membutuhkan waktu lebih lama untuk meningkatkan throughput aliran ke tingkat yang diinginkan. Karena itu, terus coba lagi catatan menelan data yang gagal. Jika Anda mengharapkan volume data meningkat dalam ledakan besar yang tiba-tiba, atau jika aliran baru Anda membutuhkan throughput yang lebih tinggi daripada batas throughput default, mintalah untuk meningkatkan batas throughput.

  Ada tiga skala kuota secara proporsional untuk kuota. Misalnya, jika Anda meningkatkan kuota throughput di AS Timur (Virginia N.), AS Barat (Oregon), atau Eropa (Irlandia) menjadi 10. MiB/second, the other two quota increase to 4,000 requests/second and 1,000,000 records/second

  > **Note**
  >
  > - Jangan gunakan batas dan kuota tingkat sumber daya sebagai cara untuk mengontrol penggunaan layanan Anda.

- Jika Kinesis Data Streams dikonfigurasi sebagai sumber data, kuota ini tidak berlaku, dan Amazon Data Firehose menskalakan naik dan turun tanpa batas.
- Jika kuota meningkat jauh lebih tinggi daripada lalu lintas berjalan, hal itu menyebabkan batch pengiriman yang kecil ke tujuan. Hal ini tidak efisien dan dapat mengakibatkan biaya yang lebih tinggi di layanan tujuan. Pastikan untuk meningkatkan kuota hanya untuk mencocokkan lalu lintas berjalan saat ini, dan tingkatkan kuota lebih lanjut jika lalu lintas meningkat.
- Catatan data yang lebih kecil dapat menyebabkan biaya yang lebih tinggi. Harga konsumsi Firehose didasarkan pada jumlah catatan data yang Anda kirim ke layanan, kali ukuran setiap catatan yang dibulatkan ke 5KB terdekat (5120 byte). Jadi, untuk volume data masuk yang sama (byte), jika ada lebih banyak catatan masuk, biaya yang dikenakan akan lebih tinggi. Misalnya, jika total volume data yang masuk adalah 5MiB, mengirim 5MiB data menggunakan lebih dari 5.000 catatan akan memakan biaya lebih tinggi dibandingkan dengan mengirim jumlah data yang sama menggunakan 1.000 catatan. Untuk informasi selengkapnya, lihat Amazon Data Firehose di Kalkulator.AWS

- Setiap aliran Firehose menyimpan catatan data hingga 24 jam jika tujuan pengiriman tidak tersedia dan jika sumbernya. DirectPut Jika sumbernya adalah Kinesis Data Streams (KDS) dan tujuan tidak tersedia, maka data akan disimpan berdasarkan konfigurasi KDS Anda.
- Ukuran maksimum catatan yang dikirim ke Amazon Data Firehose, sebelum pengkodean base64, adalah 1.000 KiB.
- PutRecordBatchOperasi dapat memakan waktu hingga 500 catatan per panggilan atau 4 MiB per panggilan, mana yang lebih kecil. Kuota ini tidak bisa diubah.
- Masing-masing operasi berikut dapat memberikan hingga lima pemanggilan per detik, yang merupakan batas sulit.
  - `CreateDeliveryStream`
  - `DeleteDeliveryStream`
  - `DescribeDeliveryStream`
  - `ListDeliveryStreams`
  - `UpdateDestination`
  - `TagDeliveryStream`
  - `UntagDeliveryStream`
  - `ListTagsForDeliveryStream`

  - StartDeliveryStreamEncryption
  - StopDeliveryStreamEncryption
- Petunjuk interval buffer berkisar antara 60 detik hingga 900 detik.
- Untuk pengiriman dari Amazon Data Firehose ke Amazon Redshift, hanya klaster Amazon Redshift yang dapat diakses publik yang didukung.
- Rentang durasi coba lagi adalah dari 0 detik hingga 7.200 detik untuk Amazon Redshift OpenSearch dan pengiriman Layanan.
- Jika tujuannya adalah Amazon S3, Amazon Redshift, atau OpenSearch Service, Amazon Data Firehose memungkinkan hingga 5 pemanggilan Lambda yang luar biasa per pecahan. Untuk Splunk, kuotanya adalah 10 panggilan Lambda yang tertunda per serpihan.
- Anda dapat menggunakan tipe CMK CUSTOMER_MANAGED_CMK untuk mengenkripsi hingga 500 aliran Firehose.

Riwayat dokumen

Tabel berikut menjelaskan perubahan penting pada dokumentasi Amazon Data Firehose.

Ubah	Deskripsi	Tanggal Diubah
Penghapusan Database sebagai sumber (pratinjau publik)	Database sebagai sumber (pratinjau publik) sekarang dihapus.	September 24, 2025
Ditambahkan dukungan untuk Glue multi-katalog hirarki	Ini menyederhanakan integrasi Firehose dengan Tabel Amazon S3 tanpa memerlukan tautan sumber daya antara katalog data default dan. `S3TablesCatalog` Lihat Menyiapkan aliran Firehose ke tabel Amazon S3.	14 Mei 2025
Ditambahkan Database sebagai sumber (pratinjau publik)	Anda sekarang dapat mereplikasi perubahan database ke Apache Iceberg Tables di Amazon S3.	November 15, 2024
Rilis General Availability (GA) untuk Menambahkan Apache Iceberg Tables sebagai tujuan	Anda dapat membuat aliran Firehose dengan Apache Iceberg Tables sebagai tujuan. Lihat Kirimkan data ke Apache Iceberg Tables.	September 30, 2024
Contoh tipe data yang ditambahkan	Menambahkan contoh tipe data yang didukung untuk Apache Iceberg Tables. Lihat Memahami tipe data yang didukung.	Agustus 22, 2024
Peluncuran Wilayah Baru	Amazon Data Firehose sekarang tersedia di Asia Pasifik (Malaysia). Lihat Kuota Amazon Data Firehose.	Agustus 22, 2024

Ubah	Deskripsi	Tanggal Diubah
Ditambahkan Apache Iceberg Tables sebagai tujuan (pratinjau publik)	Anda dapat membuat aliran Firehose dengan Apache Iceberg Tables sebagai tujuan. Lihat Kirimkan data ke Apache Iceberg Tables.	Juli 25, 2024
Petunjuk buffering untuk Snowflake	Snowflake sekarang mendukung petunjuk buffering . Lihat the section called “Konfigurasikan pengaturan tujuan untuk Snowflake”.	Juli 25, 2024
Kepingan salju sebagai tujuan di wilayah baru	Snowflake sekarang tersedia sebagai tujuan di Asia Pasifik (Singapura), Asia Pasifik (Seoul), dan Asia Pasifik (Sydney). Lihat the section called “Konfigurasikan pengaturan tujuan untuk Snowflake”.	Juli 25, 2024
Bagian panduan pengguna yang direstrukturisasi	Navigasi yang disederhanakan untuk bagian dalam panduan pengguna. Lihat Mengirim data ke aliran Firehose dan Memecahkan masalah kesalahan.	Juli 5, 2024
Amazon Data Firehose terintegrasi dengan AWS Secrets Manager	Anda sekarang dapat mengakses rahasia Anda dan mengotomatiskan rotasi kredenal dengan aman dengan Secrets Manager. Lihat the section called “Otentikasi dengan AWS Secrets Manager”.	Juni 06, 2024
Menambahkan dukungan untuk menelan log untuk Dynatrace	Anda sekarang dapat mengirim log dan peristiwa ke Dynatrace untuk analisis lebih lanjut. Lihat the section called “Konfigurasikan setelan tujuan untuk Dynatrace ”.	April 18, 2024
Rilis General Availability (GA) untuk Snowflake sebagai tujuan	Snowflake sekarang umumnya tersedia sebagai tujuan. Lihat the section called “Konfigurasikan pengaturan tujuan untuk Snowflake”.	April 17, 2024

Ubah	Deskripsi	Tanggal Diubah
Amazon Kinesis Data Firehose sekarang dikenal sebagai Amazon Data Firehose	Amazon Kinesis Data Firehose telah berganti nama menjadi Amazon Data Firehose. Lihat Apa itu Amazon Data Firehose	Februari 9, 2024
Ditambahkan Snowflake sebagai tujuan (pratinjau publik)	Anda dapat membuat aliran Firehose dengan Snowflake sebagai tujuannya. Lihat the section called “Konfigurasikan pengaturan tujuan untuk Snowflake”.	Januari 19, 2024
Ditambahkan dekompresi otomatis Log CloudWatch	Anda dapat mengaktifkan dekompresi pada aliran baru atau yang sudah ada untuk mengirim data CloudWatch Log yang didekompresi ke tujuan Firehose. Lihat the section called “Kirim CloudWatch Log ke Firehose”.	15 Desember 2023
Menambahkan Splunk Observability Cloud sebagai tujuan	Anda dapat membuat aliran Firehose dengan Splunk Observability Cloud sebagai tujuan. Lihat the section called “Konfigurasikan pengaturan tujuan untuk Splunk Observability Cloud”.	3 Oktober 2023
Menambahkan Amazon Managed Streaming untuk Apache Kafka sebagai sumber data	Anda sekarang dapat mengonfigurasi Amazon MSK untuk mengirim informasi ke aliran Firehose. Lihat the section called “Konfigurasikan pengaturan sumber untuk Amazon MSK”.	26 September 2023
Ditambahkan dukungan untuk jenis DocumentID untuk tujuan Layanan OpenSearch	Jika OpenSearch Layanan adalah tujuan aliran Firehose Anda, jenis DocumentID menunjukkan metode untuk menyiapkan ID dokumen. Metode yang didukung adalah ID dokumen yang dihasilkan Firehose dan ID dokumen yang dihasilkan OpenSearch Layanan. Lihat the section called “Konfigurasikan pengaturan tujuan”.	10 Mei 2023

Ubah	Deskripsi	Tanggal Diubah
Ditambahkan dukungan partisi dinamis	Menambahkan dukungan untuk partisi dinamis berkelanjutan dari data streaming di Amazon Data Firehose. Lihat Data streaming partisi.	31 Agustus 2021
Menambahkan topik pada prefiks kustom.	Menambahkan topik tentang ekspresi yang dapat Anda gunakan saat membangun prefiks kustom untuk data yang dikirim ke Amazon S3. Lihat the section called “Memahami awalan khusus untuk objek Amazon S3”.	20 Desember 2018
Menambahkan Tutorial Firehose Data Amazon Baru	Menambahkan tutorial yang menunjukkan cara mengirim log aliran VPC Amazon ke Splunk melalui Amazon Data Firehose. Lihat Menelan log aliran VPC ke Splunk menggunakan Amazon Data Firehose.	30 Oktober 2018
Menambahkan Empat Wilayah Firehose Data Amazon Baru	Menambahkan Paris, Mumbai, Sao Paulo, dan London. Untuk informasi selengkapnya, lihat Kuota Amazon Data Firehose.	27 Juni 2018
Menambahkan Dua Wilayah Firehose Data Amazon Baru	Menambahkan Seoul dan Montreal. Untuk informasi selengkapnya, lihat Kuota Amazon Data Firehose.	13 Juni 2018
Aliran Kinesis Baru sebagai fitur Sumber	Menambahkan Kinesis Streams sebagai sumber potensial untuk catatan untuk aliran Firehose. Untuk informasi selengkapnya, lihat Pilih sumber dan tujuan untuk aliran Firehose Anda.	18 Agustus 2017
Pembaruan pada dokumentasi konsol	Wisaya pembuatan aliran Firehose telah diperbaru i. Untuk informasi selengkapnya, lihat Tutorial: Membuat aliran Firehose dari konsol.	19 Juli 2017

Ubah	Deskripsi	Tanggal Diubah
Transformasi data baru	Anda dapat mengonfigurasi Amazon Data Firehose untuk mengubah data Anda sebelum pengiriman data. Untuk informasi selengkapnya, lihat Mengubah data sumber di Amazon Data Firehose.	19 Desember 2016
Fitur baru percobaan ulang COPY Amazon Redshift	Anda dapat mengonfigurasi Amazon Data Firehose untuk mencoba kembali perintah COPY ke klaster Amazon Redshift jika gagal. Lihat informasi selengkapnya di Tutorial: Membuat aliran Firehose dari konsol, Memahami pengiriman data di Amazon Data Firehose, dan Kuota Amazon Data Firehose.	18 Mei 2016
Tujuan Firehose Data Amazon Baru, Layanan Amazon OpenSearch	Anda dapat membuat aliran Firehose dengan Amazon OpenSearch Service sebagai tujuan. Lihat informasi selengkapnya di Tutorial: Membuat aliran Firehose dari konsol, Memahami pengiriman data di Amazon Data Firehose, dan Berikan akses Firehose ke tujuan Layanan publik OpenSearch.	19 April 2016
CloudWatch Metrik baru yang disempurnakan dan fitur pemecahan masalah	Memperbarui Pantau Firehose Data Amazon dan Memecahkan masalah kesalahan di Amazon Data Firehose.	19 April 2016
Agen Kinesis baru yang ditingkatkan	DiperbaruiKonfigurasikan agen Kinesis untuk mengirim data.	11 April 2016
Agen Kinesis baru	Menambahkan Konfigurasikan agen Kinesis untuk mengirim data.	2 Oktober 2015
Rilis awal	Rilis awal Panduan Pengembang Amazon Data Firehose.	4 Oktober 2015