aws

Panduan EMR Pengguna Amazon Tanpa Server

# Amazon EMR

# Amazon EMR: Panduan EMR Pengguna Amazon Tanpa Server

# Table of Contents

# Apa itu Amazon Tanpa EMR Server?

Amazon EMR Serverless adalah opsi penerapan untuk Amazon EMR yang menyediakan lingkungan runtime tanpa server. Ini menyederhanakan pengoperasian aplikasi analitik yang menggunakan kerangka kerja open-source terbaru, seperti Apache Spark dan Apache Hive. Dengan EMR Tanpa Server, Anda tidak perlu mengonfigurasi, mengoptimalkan, mengamankan, atau mengoperasikan cluster untuk menjalankan aplikasi dengan kerangka kerja ini.

EMRTanpa server membantu Anda menghindari sumber daya yang berlebihan atau kurang penyediaan untuk pekerjaan pemrosesan data Anda. EMRTanpa server secara otomatis menentukan sumber daya yang dibutuhkan aplikasi, mendapatkan sumber daya ini untuk memproses pekerjaan Anda, dan melepaskan sumber daya saat pekerjaan selesai. Untuk kasus penggunaan di mana aplikasi memerlukan respons dalam hitungan detik, seperti analisis data interaktif, Anda dapat melakukan pra-inisialisasi sumber daya yang dibutuhkan aplikasi saat membuat aplikasi.

Dengan EMR Tanpa Server, Anda akan terus mendapatkan manfaat Amazon, seperti kompatibilitas open sourceEMR, konkurensi, dan kinerja runtime yang dioptimalkan untuk kerangka kerja populer.

EMRServerless cocok untuk pelanggan yang menginginkan kemudahan dalam mengoperasikan aplikasi menggunakan kerangka kerja open source. Ini menawarkan startup pekerjaan cepat, manajemen kapasitas otomatis, dan kontrol biaya langsung.

## Konsep

Di bagian ini, kami membahas istilah dan konsep EMR Tanpa Server yang muncul di seluruh Panduan Pengguna Tanpa EMR Server kami.

### Versi rilis

EMRRilis Amazon adalah seperangkat aplikasi open-source dari ekosistem big data. Setiap rilis mencakup berbagai aplikasi data besar, komponen, dan fitur yang Anda pilih untuk EMR Serverless untuk disebarkan dan dikonfigurasi sehingga mereka dapat menjalankan aplikasi Anda. Saat Anda membuat aplikasi, Anda harus menentukan versi rilisnya. Pilih versi EMR rilis Amazon dan versi kerangka kerja sumber terbuka yang ingin Anda gunakan dalam aplikasi Anda. Untuk mempelajari lebih lanjut tentang versi pra-rilis, lihatEMRVersi rilis Amazon Tanpa Server.

## Aplikasi

Dengan EMR Tanpa Server, Anda dapat membuat satu atau lebih aplikasi EMR Tanpa Server yang menggunakan kerangka kerja analitik open source. Untuk membuat aplikasi, Anda harus menentukan atribut berikut:

- Versi EMR rilis Amazon untuk versi kerangka kerja sumber terbuka yang ingin Anda gunakan. Untuk menentukan versi rilis Anda, lihatEMRVersi rilis Amazon Tanpa Server.
- Runtime spesifik yang Anda ingin aplikasi Anda gunakan, seperti Apache Spark atau Apache Hive.

Setelah Anda membuat aplikasi, Anda dapat mengirimkan pekerjaan pemrosesan data atau permintaan interaktif ke aplikasi Anda.

Setiap aplikasi EMR Tanpa Server berjalan pada Amazon Virtual Private Cloud (VPC) yang aman terpisah dari aplikasi lain. Selain itu, Anda dapat menggunakan AWS Identity and Access Management (IAM) kebijakan untuk menentukan pengguna dan peran mana yang dapat mengakses aplikasi. Anda juga dapat menentukan batasan untuk mengontrol dan melacak biaya penggunaan yang dikeluarkan oleh aplikasi.

Pertimbangkan untuk membuat beberapa aplikasi saat Anda perlu melakukan hal berikut:

- Gunakan kerangka kerja open source yang berbeda
- Gunakan versi kerangka kerja open source yang berbeda untuk kasus penggunaan yang berbeda
- Lakukan pengujian A/B saat memutakhirkan dari satu versi ke versi lainnya
- Pertahankan lingkungan logis yang terpisah untuk skenario pengujian dan produksi
- Menyediakan lingkungan logis terpisah untuk tim yang berbeda dengan kontrol biaya independen dan pelacakan penggunaan
- Pisahkan line-of-business aplikasi yang berbeda

EMRServerless adalah layanan Regional yang menyederhanakan bagaimana beban kerja berjalan di beberapa Availability Zone di Region. Untuk mempelajari lebih lanjut tentang cara menggunakan aplikasi dengan EMR Tanpa Server, lihat. Berinteraksi dengan aplikasi

## Tugas berjalan

Job run adalah permintaan yang dikirimkan ke aplikasi EMR Tanpa Server yang dijalankan dan dilacak oleh aplikasi secara askron hingga selesai. Contoh pekerjaan termasuk kueri HiveQL yang

Anda kirimkan ke aplikasi Apache Hive, atau PySpark skrip pemrosesan data yang Anda kirimkan ke aplikasi Apache Spark. Saat mengirimkan pekerjaan, Anda harus menentukan peran runtime, yang ditulisIAM, yang digunakan pekerjaan untuk mengakses AWS sumber daya, seperti objek Amazon S3. Anda dapat mengirimkan beberapa permintaan job run ke aplikasi, dan setiap job run dapat menggunakan peran runtime yang berbeda untuk mengakses AWS sumber daya. Aplikasi EMR tanpa server mulai mengeksekusi pekerjaan segera setelah menerimanya dan menjalankan beberapa permintaan pekerjaan secara bersamaan. Untuk mempelajari lebih lanjut tentang cara EMR Serverless menjalankan pekerjaan, lihat. Menjalankan pekerjaan

## Pekerja

Aplikasi EMR tanpa server secara internal menggunakan pekerja untuk mengeksekusi beban kerja Anda. Ukuran default pekerja ini didasarkan pada jenis aplikasi Anda dan versi EMR rilis Amazon. Saat Anda menjadwalkan pekerjaan, Anda dapat mengganti ukuran ini.

Saat Anda mengirimkan pekerjaan, EMR Tanpa Server menghitung sumber daya yang dibutuhkan aplikasi untuk pekerjaan dan menjadwalkan pekerja. EMRTanpa server memecah beban kerja Anda menjadi tugas, mengunduh gambar, ketentuan, dan menyiapkan pekerja, dan menonaktifkannya saat pekerjaan selesai. EMRTanpa server secara otomatis meningkatkan atau menurunkan pekerja berdasarkan beban kerja dan paralelisme yang diperlukan pada setiap tahap pekerjaan. Penskalaan otomatis ini menghilangkan kebutuhan bagi Anda untuk memperkirakan jumlah pekerja yang dibutuhkan aplikasi untuk menjalankan beban kerja Anda.

## Kapasitas pra-inisialisasi

EMRTanpa server menyediakan fitur kapasitas pra-inisialisasi yang membuat pekerja diinisialisasi dan siap merespons dalam hitungan detik. Kapasitas ini secara efektif menciptakan kumpulan pekerja yang hangat untuk suatu aplikasi. Untuk mengkonfigurasi fitur ini untuk setiap aplikasi, atur `initial-capacity` parameter aplikasi. Saat Anda mengonfigurasi kapasitas pra-inisialisasi, pekerjaan dapat segera dimulai sehingga Anda dapat menerapkan aplikasi berulang dan pekerjaan yang sensitif terhadap waktu. Untuk mempelajari lebih lanjut tentang pekerja pra-inisialisasi, lihat. Mengkonfigurasi aplikasi

## EMRStudio

EMRStudio adalah konsol pengguna yang dapat Anda gunakan untuk mengelola aplikasi EMR Tanpa Server Anda. Jika EMR Studio tidak ada di akun Anda saat Anda membuat aplikasi EMR Tanpa Server pertama, kami secara otomatis membuatnya untuk Anda. Anda dapat mengakses EMR

Studio baik dari EMR konsol Amazon, atau Anda dapat mengaktifkan akses federasi dari penyedia identitas (iDP) Anda IAM melalui IAM atau Pusat Identitas. Saat Anda melakukan ini, pengguna dapat mengakses Studio dan mengelola aplikasi EMR Tanpa Server tanpa akses langsung ke konsol AmazonEMR. Untuk mempelajari lebih lanjut tentang cara EMR kerja aplikasi Tanpa Server dengan EMR Studio, lihat Berinteraksi dengan aplikasi Anda dari konsol EMR Studio dan. Menjalankan pekerjaan dari konsol EMR Studio

# Prasyarat untuk memulai dengan Tanpa Server EMR

Topik



## Mendaftar untuk Akun AWS

Jika Anda tidak memiliki Akun AWS, selesaikan langkah-langkah berikut untuk membuatnya.

Untuk mendaftar untuk Akun AWS

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

   Bagian dari prosedur pendaftaran melibatkan tindakan menerima panggilan telepon dan memasukkan kode verifikasi di keypad telepon.

   Ketika Anda mendaftar untuk Akun AWS, sebuah Pengguna root akun AWSdiciptakan. 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.

AWS mengirimkan email konfirmasi setelah proses pendaftaran selesai. Kapan saja, Anda dapat melihat aktivitas akun Anda saat ini dan mengelola akun Anda dengan masuk https://aws.amazon.comke/ dan memilih Akun Saya.

## Buat pengguna dengan akses administratif

Setelah Anda mendaftar untuk Akun AWS, amankan Pengguna root akun AWS, aktifkan AWS IAM Identity Center, dan buat pengguna administratif sehingga Anda tidak menggunakan pengguna root untuk tugas sehari-hari.

Amankan Anda Pengguna root akun AWS

1. Masuk ke AWS Management Consolesebagai pemilik akun dengan memilih pengguna Root dan memasukkan Anda Akun AWS alamat email. Di laman berikutnya, masukkan kata sandi.

   Untuk bantuan masuk menggunakan pengguna root, lihat Masuk sebagai pengguna root di AWS Sign-In Panduan Pengguna.
2. Aktifkan otentikasi multi-faktor (MFA) untuk pengguna root Anda.

   Untuk petunjuk, lihat Mengaktifkan MFA perangkat virtual untuk Akun AWS root user (konsol) di Panduan IAM Pengguna.

Buat pengguna dengan akses administratif

1. Aktifkan Pusat IAM Identitas.

   Untuk petunjuk, lihat Mengaktifkan AWS IAM Identity Centerdi AWS IAM Identity Center Panduan Pengguna.
2. Di Pusat IAM Identitas, berikan akses administratif ke pengguna.

   Untuk tutorial tentang menggunakan Direktori Pusat Identitas IAM sebagai sumber identitas Anda, lihat Mengkonfigurasi akses pengguna dengan default Direktori Pusat Identitas IAMdi AWS IAM Identity Center Panduan Pengguna.

Masuk sebagai pengguna dengan akses administratif

- Untuk masuk dengan pengguna Pusat IAM Identitas, gunakan login URL yang dikirim ke alamat email saat Anda membuat pengguna Pusat IAM Identitas.

  Untuk bantuan masuk menggunakan pengguna Pusat IAM Identitas, lihat Masuk ke AWS akses portal di AWS Sign-In Panduan Pengguna.

Tetapkan akses ke pengguna tambahan

1. Di Pusat IAM Identitas, buat set izin yang mengikuti praktik terbaik menerapkan izin hak istimewa paling sedikit.

   Untuk petunjuk, lihat Membuat set izin di AWS IAM Identity Center Panduan Pengguna.

2. Tetapkan pengguna ke grup, lalu tetapkan akses masuk tunggal ke grup.

   Untuk petunjuk, lihat Menambahkan grup di AWS IAM Identity Center Panduan Pengguna.

## Berikan izin

Di lingkungan produksi, kami menyarankan Anda menggunakan kebijakan yang lebih halus. Untuk contoh kebijakan tersebut, lihatContoh kebijakan akses pengguna untuk Tanpa EMR Server. Untuk mempelajari lebih lanjut tentang manajemen akses, lihat Manajemen akses untuk AWS sumber daya dalam Panduan IAM Pengguna.

Untuk pengguna yang perlu memulai dengan EMR Tanpa Server di lingkungan kotak pasir, gunakan kebijakan yang serupa dengan berikut ini:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EMRStudioCreate",
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:CreateStudioPresignedUrl",
                "elasticmapreduce:DescribeStudio",
                "elasticmapreduce:CreateStudio",
                "elasticmapreduce:ListStudios"
            ],
            "Resource": "*"
        },
        {
            "Sid": "EMRServerlessFullAccess",
            "Effect": "Allow",
            "Action": [
                "emr-serverless:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowEC2ENICreationWithEMRTags",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface"
            ],
```

```
            "Resource": [
                "arn:aws:ec2:*:*:network-interface/*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:CalledViaLast": "ops.emr-serverless.amazonaws.com"
                }
            }
        },
        {
            "Sid": "AllowEMRServerlessServiceLinkedRoleCreation",
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/*"
        }
    ]
}
```

Untuk memberikan akses, menambahkan izin ke pengguna, grup, atau peran Anda:

- Pengguna dan grup di AWS IAM Identity Center:

  Buat rangkaian izin. Ikuti petunjuk di Buat set izin di AWS IAM Identity Center Panduan Pengguna.

- Pengguna dikelola IAM melalui penyedia identitas:

  Buat peran untuk federasi identitas. Ikuti petunjuk dalam Membuat peran untuk penyedia identitas pihak ketiga (federasi) di Panduan IAM Pengguna.

- IAMpengguna:
  - Buat peran yang dapat diambil pengguna Anda. Ikuti petunjuk dalam Membuat peran bagi IAM pengguna di Panduan IAM Pengguna.
  - (Tidak disarankan) Pasang kebijakan langsung ke pengguna atau tambahkan pengguna ke grup pengguna. Ikuti petunjuk di Menambahkan izin ke pengguna (konsol) di Panduan IAM Pengguna.

## Memberikan akses programatis

Pengguna membutuhkan akses terprogram jika mereka ingin berinteraksi dengan AWS di luar AWS Management Console. Cara untuk memberikan akses terprogram tergantung pada jenis pengguna yang mengakses AWS.

Untuk memberi pengguna akses programatis, pilih salah satu opsi berikut.

| Pengguna mana yang membutuhkan akses programatis? | Untuk | Oleh |
| --- | --- | --- |
| Identitas tenaga kerja<br><br>(Pengguna dikelola di Pusat IAM Identitas) | Gunakan kredensi sementara untuk menandatangani permintaan terprogram ke AWS CLI, AWS SDKs, atau AWS APIs. | Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan.<br><br>• Untuk AWS CLI, lihat Mengonfigurasi AWS CLI untuk menggunakan AWS IAM Identity Centerdi AWS Command Line Interface Panduan Pengguna.<br>• Untuk AWS SDKs, alat, dan AWS APIs, lihat otentikasi Pusat IAM Identitas di AWS SDKsdan Panduan Referensi Alat. |
| IAM | Gunakan kredensi sementara untuk menandatangani permintaan terprogram ke AWS CLI, AWS SDKs, atau AWS APIs. | Mengikuti petunjuk dalam Menggunakan kredensil sementara dengan AWS sumber daya dalam Panduan IAM Pengguna. |
| IAM | (Tidak direkomendasikan) Gunakan kredensi jangka panjang untuk menandata ngani permintaan terprogram ke AWS CLI, AWS SDKs, atau AWS APIs. | Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan.<br><br>• Untuk AWS CLI, lihat Mengautentikasi menggunakan kredensi IAM pengguna di AWS Command Line Interface Panduan Pengguna. |

| Pengguna mana yang membutuhkan akses programatis? | Untuk | Oleh |
| --- | --- | --- |
| | | • Untuk AWS SDKsdan alat, lihat Mengauten tikasi menggunakan kredensi jangka panjang di AWS SDKsdan Panduan Referensi Alat.<br>• Untuk AWS APIs, lihat Mengelola kunci akses untuk IAM pengguna di Panduan IAM Pengguna. |

## Instal dan konfigurasikan AWS CLI

Jika Anda ingin menggunakan EMR Tanpa ServerAPIs, Anda harus menginstal versi terbaru AWS Command Line Interface (AWS CLI). Anda tidak membutuhkan AWS CLI untuk menggunakan EMR Tanpa Server dari konsol EMR Studio, dan Anda dapat memulai tanpa CLI mengikuti langkah-langkah di. Memulai dengan EMR Tanpa Server dari konsol

Untuk mengatur AWS CLI

1. Untuk menginstal versi terbaru dari AWS CLI untuk macOS, Linux, atau Windows, lihat Menginstal atau memperbarui versi terbaru AWS CLI.
2. Untuk mengkonfigurasi AWS CLI dan pengaturan aman akses Anda ke Layanan AWS, termasuk EMR Tanpa Server, lihat Konfigurasi cepat dengan. `aws configure`
3. Untuk memverifikasi pengaturan, masukkan DataBrew perintah berikut pada prompt perintah.

   ```
   aws emr-serverless help
   ```

   AWS CLI perintah menggunakan default Wilayah AWS dari konfigurasi Anda, kecuali Anda mengaturnya dengan parameter atau profil. Untuk mengatur Wilayah AWS dengan parameter, Anda dapat menambahkan `--region` parameter ke setiap perintah.

Untuk mengatur Wilayah AWS dengan profil, pertama-tama tambahkan profil bernama dalam `~/.aws/config` file atau `%UserProfile%/.aws/config` file (untuk Microsoft Windows). Ikuti langkah-langkah di Profil bernama untuk AWS CLI. Selanjutnya, atur Wilayah AWS dan pengaturan lain dengan perintah yang mirip dengan yang ada di contoh berikut.

```
[profile emr-serverless]
aws_access_key_id = ACCESS-KEY-ID-OF-IAM-USER
aws_secret_access_key = SECRET-ACCESS-KEY-ID-OF-IAM-USER
region = us-east-1
output = text
```

# Buka konsol

Sebagian besar topik berorientasi konsol di bagian ini dimulai dari konsol Amazon. EMR Jika Anda belum masuk ke Akun AWS, masuk, lalu buka EMRkonsol Amazon dan lanjutkan ke bagian berikutnya untuk melanjutkan memulai dengan AmazonEMR.

# Memulai dengan Amazon Tanpa EMR Server

Tutorial ini membantu Anda memulai dengan EMR Tanpa Server saat Anda menerapkan contoh beban kerja Spark atau Hive. Anda akan membuat, menjalankan, dan men-debug aplikasi Anda sendiri. Kami menampilkan opsi default di sebagian besar bagian tutorial ini.

Sebelum Anda meluncurkan aplikasi EMR Tanpa Server, selesaikan tugas-tugas berikut.

Topik



## Berikan izin untuk menggunakan Tanpa Server EMR

Untuk menggunakan EMR Tanpa Server, Anda memerlukan pengguna atau IAM peran dengan kebijakan terlampir yang memberikan izin untuk Tanpa Server. EMR Untuk membuat pengguna dan melampirkan kebijakan yang sesuai kepada pengguna tersebut, ikuti instruksi diBerikan izin.

## Siapkan penyimpanan untuk Tanpa EMR Server

Dalam tutorial ini, Anda akan menggunakan bucket S3 untuk menyimpan file output dan log dari sampel beban kerja Spark atau Hive yang akan Anda jalankan menggunakan aplikasi Tanpa Server. EMR Untuk membuat bucket, ikuti petunjuk dalam Membuat bucket di Panduan Pengguna Amazon Simple Storage Service Console. Ganti referensi lebih lanjut *DOC-EXAMPLE-BUCKET* dengan nama bucket yang baru dibuat.

## Membuat EMR Studio untuk menjalankan beban kerja interaktif

Jika Anda ingin menggunakan EMR Tanpa Server untuk menjalankan kueri interaktif melalui buku catatan yang dihosting di EMR Studio, Anda perlu menentukan bucket S3 dan peran layanan minimum untuk Tanpa EMR Server untuk membuat Ruang Kerja. Untuk langkah-langkah penyiapan,

lihat Menyiapkan EMR Studio di Panduan EMR Manajemen Amazon. Untuk informasi selengkapnya tentang beban kerja interaktif, lihatJalankan beban kerja interaktif dengan Tanpa EMR Server melalui Studio EMR.

## Buat peran runtime pekerjaan

Job berjalan di EMR Serverless menggunakan peran runtime yang memberikan izin terperinci untuk spesifik Layanan AWS dan sumber daya saat runtime. Dalam tutorial ini, bucket S3 publik menghosting data dan skrip. Ember *DOC-EXAMPLE-BUCKET* menyimpan output.

Untuk menyiapkan peran runtime pekerjaan, pertama-tama buat peran runtime dengan kebijakan kepercayaan sehingga EMR Tanpa Server dapat menggunakan peran baru. Selanjutnya, lampirkan kebijakan akses S3 yang diperlukan ke peran itu. Langkah-langkah berikut memandu Anda melalui proses tersebut.

Console

1. Arahkan ke IAM konsol dihttps://console.aws.amazon.com/iam/.
2. Di panel navigasi sebelah kiri, pilih Peran.
3. Pilih Buat peran.
4. Untuk jenis peran, pilih Kebijakan kepercayaan khusus dan tempel kebijakan kepercayaan berikut. Ini memungkinkan pekerjaan yang dikirimkan ke aplikasi Amazon EMR Tanpa Server Anda untuk mengakses lainnya Layanan AWS atas nama Anda.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "emr-serverless.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

5. Pilih Berikutnya untuk menavigasi ke halaman Tambahkan izin, lalu pilih Buat kebijakan.
6. Halaman Buat kebijakan terbuka di tab baru. Tempel kebijakan di JSON bawah ini.

> **Important**
>
> Ganti *DOC-EXAMPLE-BUCKET* dalam kebijakan di bawah ini dengan nama bucket aktual yang dibuat diSiapkan penyimpanan untuk Tanpa EMR Server. Ini adalah kebijakan dasar untuk akses S3. Untuk contoh peran runtime pekerjaan lainnya, lihatPeran runtime Job untuk Amazon Serverless EMR.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadAccessForEMRSamples",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::*.elasticmapreduce",
                "arn:aws:s3:::*.elasticmapreduce/*"
            ]
        },
        {
            "Sid": "FullAccessToOutputBucket",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
            ]
        },
        {
            "Sid": "GlueCreateAndReadDataCatalog",
            "Effect": "Allow",
            "Action": [
```

```
                "glue:GetDatabase",
                "glue:CreateDatabase",
                "glue:GetDataBases",
                "glue:CreateTable",
                "glue:GetTable",
                "glue:UpdateTable",
                "glue:DeleteTable",
                "glue:GetTables",
                "glue:GetPartition",
                "glue:GetPartitions",
                "glue:CreatePartition",
                "glue:BatchCreatePartition",
                "glue:GetUserDefinedFunctions"
            ],
            "Resource": ["*"]
        }
    ]
}
```

7. Pada halaman Kebijakan ulasan, masukkan nama untuk kebijakan Anda, sepertiEMRServerlessS3AndGlueAccessPolicy.
8. Segarkan halaman Kebijakan izin Lampirkan, lalu pilihEMRServerlessS3AndGlueAccessPolicy.
9. Di halaman Nama, tinjau, dan buat, untuk nama Peran, masukkan nama untuk peran Anda, misalnya,EMRServerlessS3RuntimeRole. Untuk membuat IAM peran ini, pilih Buat peran.

CLI

1. Buat file bernama emr-serverless-trust-policy.json yang berisi kebijakan kepercayaan yang akan digunakan untuk IAM peran tersebut. File harus berisi kebijakan berikut.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Sid": "EMRServerlessTrustPolicy",
        "Action": "sts:AssumeRole",
        "Effect": "Allow",
        "Principal": {
            "Service": "emr-serverless.amazonaws.com"
        }
```

```
    }]
}
```

2. Buat IAM peran bernama`EMRServerlessS3RuntimeRole`. Gunakan kebijakan kepercayaan yang Anda buat di langkah sebelumnya.

```
aws iam create-role \
    --role-name EMRServerlessS3RuntimeRole \
    --assume-role-policy-document file://emr-serverless-trust-policy.json
```

Perhatikan ARN pada outputnya. Anda menggunakan peran baru selama pengajuan pekerjaan, yang disebut setelah ini sebagai. ARN *`job-role-arn`*

3. Buat file bernama `emr-sample-access-policy.json` yang menentukan IAM kebijakan untuk beban kerja Anda. Ini menyediakan akses baca ke skrip dan data yang disimpan di bucket S3 publik dan akses baca-tulis. *`DOC-EXAMPLE-BUCKET`*

> **Important**
>
> Ganti *`DOC-EXAMPLE-BUCKET`* dalam kebijakan di bawah ini dengan nama bucket sebenarnya yang dibuat diSiapkan penyimpanan untuk Tanpa EMR Server.. Ini adalah kebijakan dasar untuk AWS Akses Glue dan S3. Untuk contoh peran runtime pekerjaan lainnya, lihatPeran runtime Job untuk Amazon Serverless EMR.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadAccessForEMRSamples",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::*.elasticmapreduce",
                "arn:aws:s3:::*.elasticmapreduce/*"
            ]
        },
        {
```

```
            "Sid": "FullAccessToOutputBucket",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
            ]
        },
        {
            "Sid": "GlueCreateAndReadDataCatalog",
            "Effect": "Allow",
            "Action": [
                "glue:GetDatabase",
                "glue:CreateDatabase",
                "glue:GetDataBases",
                "glue:CreateTable",
                "glue:GetTable",Understanding default application behavior,
 including auto-start and auto-stop, as well as maximum capacity and worker
 configurations for configuring an application with &EMRServerless;.
                "glue:UpdateTable",
                "glue:DeleteTable",
                "glue:GetTables",
                "glue:GetPartition",
                "glue:GetPartitions",
                "glue:CreatePartition",
                "glue:BatchCreatePartition",
                "glue:GetUserDefinedFunctions"
            ],
            "Resource": ["*"]
        }
    ]
}
```

4. Buat IAM kebijakan bernama `EMRServerlessS3AndGlueAccessPolicy` dengan file kebijakan yang Anda buat di Langkah 3. Perhatikan ARN di output, karena Anda akan menggunakan kebijakan baru di langkah berikutnya. ARN

```
aws iam create-policy \
```

```
    --policy-name EMRServerlessS3AndGlueAccessPolicy \
    --policy-document file://emr-sample-access-policy.json
```

Perhatikan kebijakan baru ARN di output. Anda akan menggantinya *policy-arn* di langkah berikutnya.

5. Lampirkan IAM kebijakan `EMRServerlessS3AndGlueAccessPolicy` ke peran `EMRServerlessS3RuntimeRole` runtime pekerjaan.

```
aws iam attach-role-policy \
    --role-name EMRServerlessS3RuntimeRole \
    --policy-arn policy-arn
```

# Memulai dengan EMR Tanpa Server dari konsol

Langkah-langkah untuk menyelesaikan

- Langkah 1: Buat aplikasi EMR Tanpa Server
- Langkah 2: Kirim pekerjaan atau beban kerja interaktif
- Langkah 3: Lihat UI aplikasi dan log
- Langkah 4: Membersihkan

## Langkah 1: Buat aplikasi EMR Tanpa Server

Buat aplikasi baru dengan EMR Serverless sebagai berikut.

1. Masuk ke AWS Management Console dan buka EMR konsol Amazon di https://console.aws.amazon.com/emr.
2. Di panel navigasi kiri, pilih EMRTanpa Server untuk menavigasi ke halaman landing Tanpa EMR Server.
3. Untuk membuat atau mengelola aplikasi EMR Tanpa Server, Anda memerlukan UI EMR Studio.
    - Jika Anda sudah memiliki EMR studio di Wilayah AWS di mana Anda ingin membuat aplikasi, lalu pilih Kelola aplikasi untuk menavigasi ke EMR Studio Anda, atau pilih studio yang ingin Anda gunakan.

- Jika Anda tidak memiliki EMR studio di Wilayah AWS di mana Anda ingin membuat aplikasi, pilih Mulai dan kemudian Pilih Buat dan luncurkan Studio. EMRTanpa server membuat EMR Studio untuk Anda sehingga Anda dapat membuat dan mengelola aplikasi.

4. Di UI Buat studio yang terbuka di tab baru, masukkan nama, jenis, dan versi rilis untuk aplikasi Anda. Jika Anda hanya ingin menjalankan pekerjaan batch, pilih Gunakan pengaturan default untuk pekerjaan batch saja. Untuk beban kerja interaktif, pilih Gunakan pengaturan default untuk beban kerja interaktif. Anda juga dapat menjalankan pekerjaan batch pada aplikasi yang diaktifkan interaktif dengan opsi ini. Jika perlu, Anda dapat mengubah pengaturan ini nanti.

   Untuk informasi selengkapnya, lihat Membuat studio.

5. Pilih Buat aplikasi untuk membuat aplikasi pertama Anda.

Lanjutkan ke bagian berikutnya Langkah 2: Kirim pekerjaan atau beban kerja interaktif untuk mengirimkan pekerjaan atau beban kerja interaktif.

## Langkah 2: Kirim pekerjaan atau beban kerja interaktif

Spark job run

Dalam tutorial ini, kita menggunakan PySpark script untuk menghitung jumlah kemunculan kata-kata unik di beberapa file teks. Bucket S3 publik dan hanya-baca menyimpan skrip dan kumpulan data.

Untuk menjalankan pekerjaan Spark

1. Unggah skrip sampel `wordcount.py` ke bucket baru Anda dengan perintah berikut.

   ```
   aws s3 cp s3://us-east-1.elasticmapreduce/emr-containers/samples/wordcount/
   scripts/wordcount.py s3://DOC-EXAMPLE-BUCKET/scripts/
   ```

2. Menyelesaikan Langkah 1: Buat aplikasi EMR Tanpa Server membawa Anda ke halaman detail Aplikasi di EMR Studio. Di sana, pilih opsi Kirim pekerjaan.
3. Pada halaman Kirim pekerjaan, lengkapi yang berikut ini.
   - Di bidang Nama, masukkan nama yang ingin Anda panggil job run.
   - Di bidang peran Runtime, masukkan nama peran yang Anda buat. Buat peran runtime pekerjaan

- Di bidang Lokasi skrip, masukkan `s3://DOC-EXAMPLE-BUCKET/scripts/wordcount.py` sebagai S3URI.
- Di bidang argumen Script, masukkan `["s3://DOC-EXAMPLE-BUCKET/emr-serverless-spark/output"]`.
- Di bagian properti Spark, pilih Edit sebagai teks dan masukkan konfigurasi berikut.

```
--conf spark.executor.cores=1 --conf spark.executor.memory=4g --
conf spark.driver.cores=1 --conf spark.driver.memory=4g --conf
 spark.executor.instances=1
```

4. Untuk memulai pekerjaan, pilih Kirim pekerjaan.
5. Di tab Job runs, Anda akan melihat pekerjaan baru Anda berjalan dengan status Running.

Hive job run

Di bagian tutorial ini, kita membuat tabel, menyisipkan beberapa catatan, dan menjalankan kueri agregasi hitungan. Untuk menjalankan pekerjaan Hive, pertama-tama buat file yang berisi semua kueri Hive untuk dijalankan sebagai bagian dari pekerjaan tunggal, unggah file ke S3, dan tentukan jalur S3 ini saat memulai pekerjaan Hive.

Untuk menjalankan pekerjaan Hive

1. Buat file bernama `hive-query.ql` yang berisi semua kueri yang ingin Anda jalankan dalam pekerjaan Hive Anda.

```
create database if not exists emrserverless;
use emrserverless;
create table if not exists test_table(id int);
drop table if exists Values__Tmp__Table__1;
insert into test_table values (1),(2),(2),(3),(3),(3);
select id, count(id) from test_table group by id order by id desc;
```

2. Unggah `hive-query.ql` ke bucket S3 Anda dengan perintah berikut.

```
aws s3 cp hive-query.ql s3://DOC-EXAMPLE-BUCKET/emr-serverless-hive/query/hive-
query.ql
```

3. Menyelesaikan Langkah 1: Buat aplikasi EMR Tanpa Server membawa Anda ke halaman detail Aplikasi di EMR Studio. Di sana, pilih opsi Kirim pekerjaan.

4. Pada halaman Kirim pekerjaan, lengkapi yang berikut ini.

   - Di bidang Nama, masukkan nama yang ingin Anda panggil job run.
   - Di bidang peran Runtime, masukkan nama peran yang Anda buat. Buat peran runtime pekerjaan
   - Di bidang Lokasi skrip, masukkan `s3://DOC-EXAMPLE-BUCKET/emr-serverless-hive/query/hive-query.ql` sebagai S3URI.
   - Di bagian Properti sarang, pilih Edit sebagai teks, dan masukkan konfigurasi berikut.

     ```
     --hiveconf hive.log.explain.output=false
     ```

   - Di bagian konfigurasi Job, pilih Edit sebagai JSON, dan masukkan yang berikut iniJSON.

     ```
     {
        "applicationConfiguration":
        [{
            "classification": "hive-site",
               "properties": {
                   "hive.exec.scratchdir": "s3://DOC-EXAMPLE-BUCKET/emr-serverless-
     hive/hive/scratch",
                   "hive.metastore.warehouse.dir": "s3://DOC-EXAMPLE-BUCKET/emr-
     serverless-hive/hive/warehouse",
                   "hive.driver.cores": "2",
                   "hive.driver.memory": "4g",
                   "hive.tez.container.size": "4096",
                   "hive.tez.cpu.vcores": "1"
               }
         }]
     }
     ```

5. Untuk memulai pekerjaan, pilih Kirim pekerjaan.
6. Di tab Job runs, Anda akan melihat pekerjaan baru Anda berjalan dengan status Running.

Interactive workload

Dengan Amazon EMR 6.14.0 dan yang lebih tinggi, Anda dapat menggunakan notebook yang di-host di EMR Studio untuk menjalankan beban kerja interaktif untuk Spark di Tanpa Server. EMR Untuk informasi selengkapnya termasuk izin dan prasyarat, lihat. Jalankan beban kerja interaktif dengan Tanpa EMR Server melalui Studio EMR

Setelah membuat aplikasi dan menyiapkan izin yang diperlukan, gunakan langkah-langkah berikut untuk menjalankan buku catatan interaktif dengan EMR Studio:

1. Arahkan ke tab Workspaces di EMR Studio. Jika Anda masih perlu mengonfigurasi lokasi penyimpanan Amazon S3 dan peran layanan EMR Studio, pilih tombol Configure studio di spanduk di bagian atas layar.
2. Untuk mengakses buku catatan, pilih Workspace atau buat Workspace baru. Gunakan Quick launch untuk membuka Workspace Anda di tab baru.
3. Buka tab yang baru dibuka. Pilih ikon Compute dari navigasi kiri. Pilih EMR Tanpa Server sebagai tipe Compute.
4. Pilih aplikasi berkemampuan interaktif yang Anda buat di bagian sebelumnya.
5. Di bidang peran Runtime, masukkan nama IAM peran yang dapat diasumsikan oleh aplikasi EMR Tanpa Server Anda untuk menjalankan pekerjaan. Untuk mempelajari lebih lanjut tentang peran runtime, lihat Peran runtime Job di Panduan Pengguna Tanpa EMR Server Amazon.
6. Pilih Lampirkan. Ini mungkin memakan waktu hingga satu menit. Halaman akan disegarkan saat dilampirkan.
7. Pilih kernel dan mulai notebook. Anda juga dapat menelusuri contoh buku catatan di EMR Tanpa Server dan menyalinnya ke Workspace Anda. Untuk mengakses contoh buku catatan, navigasikan ke **`{...}`**menu di navigasi kiri dan telusuri buku catatan yang ada `serverless` di nama file notebook.
8. Di buku catatan, Anda dapat mengakses tautan log driver dan tautan ke Apache Spark UI, antarmuka real-time yang menyediakan metrik untuk memantau pekerjaan Anda. Untuk informasi selengkapnya, lihat Memantau aplikasi dan pekerjaan EMR Tanpa Server di Panduan Pengguna Tanpa EMRServer Amazon.

Saat Anda melampirkan aplikasi ke ruang kerja Studio, aplikasi mulai terpicu secara otomatis jika aplikasi tersebut belum berjalan. Anda juga dapat memulai aplikasi terlebih dahulu dan menyiapkannya sebelum Anda melampirkannya ke ruang kerja.

## Langkah 3: Lihat UI aplikasi dan log

Untuk melihat UI aplikasi, pertama-tama identifikasi pekerjaan yang dijalankan. Opsi untuk Spark UI atau Hive Tez UI tersedia di baris pertama opsi untuk pekerjaan itu, berdasarkan jenis pekerjaan. Pilih opsi yang sesuai.

Jika Anda memilih UI Spark, pilih tab Executors untuk melihat log driver dan pelaksana. Jika Anda memilih Hive Tez UI, pilih tab Semua Tugas untuk melihat log.

Setelah status job run ditampilkan sebagai Sukses, Anda dapat melihat output pekerjaan di bucket S3 Anda.

### Langkah 4: Membersihkan

Meskipun aplikasi yang Anda buat harus berhenti otomatis setelah 15 menit tidak aktif, kami tetap menyarankan Anda merilis sumber daya yang tidak ingin Anda gunakan lagi.

Untuk menghapus aplikasi, navigasikan ke halaman Daftar aplikasi. Pilih aplikasi yang Anda buat dan pilih Tindakan → Berhenti untuk menghentikan aplikasi. Setelah aplikasi dalam `STOPPED` keadaan, pilih aplikasi yang sama dan pilih Tindakan → Hapus.

Untuk lebih banyak contoh menjalankan pekerjaan Spark dan Hive, lihat Lowongan kerja Spark dan. Pekerjaan sarang

## Memulai dari AWS CLI

### Langkah 1: Buat aplikasi EMR Tanpa Server

Gunakan `emr-serverless create-application`perintah untuk membuat aplikasi EMR Tanpa Server pertama Anda. Anda perlu menentukan jenis aplikasi dan label EMR rilis Amazon yang terkait dengan versi aplikasi yang ingin Anda gunakan. Nama aplikasi adalah opsional.

Spark

Untuk membuat aplikasi Spark, jalankan perintah berikut.

```
aws emr-serverless create-application \
    --release-label emr-6.6.0 \
    --type "SPARK" \
    --name my-application
```

Hive

Untuk membuat aplikasi Hive, jalankan perintah berikut.

```
aws emr-serverless create-application \
    --release-label emr-6.6.0 \
```

```
    --type "HIVE" \
    --name my-application
```

Perhatikan ID aplikasi yang dikembalikan dalam output. Anda akan menggunakan ID untuk memulai aplikasi dan selama pengiriman pekerjaan, yang disebut setelah ini sebagai. *application-id*

Sebelum Anda melanjutkan keLangkah 2: Kirim pekerjaan ke aplikasi Tanpa EMR Server Anda, pastikan bahwa aplikasi Anda telah mencapai CREATED status dengan get-applicationAPI.

```
aws emr-serverless get-application \
    --application-id application-id
```

EMRTanpa server menciptakan pekerja untuk mengakomodasi pekerjaan yang Anda minta. Secara default, ini dibuat sesuai permintaan, tetapi Anda juga dapat menentukan kapasitas pra-inisialisasi dengan mengatur `initialCapacity` parameter saat Anda membuat aplikasi. Anda juga dapat membatasi total kapasitas maksimum yang dapat digunakan aplikasi dengan `maximumCapacity` parameter. Untuk mempelajari selengkapnya tentang opsi ini, lihat Mengkonfigurasi aplikasi.

## Langkah 2: Kirim pekerjaan ke aplikasi Tanpa EMR Server Anda

Sekarang aplikasi EMR Tanpa Server Anda siap menjalankan pekerjaan.

Spark

Pada langkah ini, kami menggunakan PySpark skrip untuk menghitung jumlah kemunculan kata-kata unik di beberapa file teks. Bucket S3 publik dan hanya-baca menyimpan skrip dan kumpulan data. Aplikasi mengirimkan file output dan data log dari runtime Spark ke `/output` dan `/logs` direktori di bucket S3 yang Anda buat.

Untuk menjalankan pekerjaan Spark

1. Gunakan perintah berikut untuk menyalin skrip sampel yang akan kami jalankan ke bucket baru Anda.

   ```
   aws s3 cp s3://us-east-1.elasticmapreduce/emr-containers/samples/wordcount/
   scripts/wordcount.py s3://DOC-EXAMPLE-BUCKET/scripts/
   ```

2. Dalam perintah berikut, ganti *application-id* dengan ID aplikasi Anda. Ganti *job-role-arn* dengan peran runtime yang ARN Anda buat. Buat peran runtime pekerjaan Pengganti *job-run-name* dengan nama yang ingin Anda sebut pekerjaan Anda berjalan. Ganti

semua *DOC-EXAMPLE-BUCKET* string dengan bucket Amazon S3 yang Anda buat, dan `/output` tambahkan ke path. Ini membuat folder baru di bucket tempat EMR Serverless dapat menyalin file keluaran aplikasi Anda.

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --name job-run-name \
    --job-driver '{
        "sparkSubmit": {
          "entryPoint": "s3://DOC-EXAMPLE-BUCKET/scripts/wordcount.py",
          "entryPointArguments": ["s3://DOC-EXAMPLE-BUCKET/emr-serverless-spark/
output"],
          "sparkSubmitParameters": "--conf spark.executor.cores=1
 --conf spark.executor.memory=4g --conf spark.driver.cores=1 --conf
 spark.driver.memory=4g --conf spark.executor.instances=1"
        }
    }'
```

3. Perhatikan ID job run yang dikembalikan dalam output. Ganti *job-run-id* dengan ID ini dalam langkah-langkah berikut.

Hive

Dalam tutorial ini, kita membuat tabel, menyisipkan beberapa catatan, dan menjalankan kueri agregasi hitungan. Untuk menjalankan pekerjaan Hive, pertama-tama buat file yang berisi semua kueri Hive untuk dijalankan sebagai bagian dari pekerjaan tunggal, unggah file ke S3, dan tentukan jalur S3 ini saat Anda memulai pekerjaan Hive.

Untuk menjalankan pekerjaan Hive

1. Buat file bernama `hive-query.ql` yang berisi semua kueri yang ingin Anda jalankan dalam pekerjaan Hive Anda.

```
create database if not exists emrserverless;
use emrserverless;
create table if not exists test_table(id int);
drop table if exists Values__Tmp__Table__1;
insert into test_table values (1),(2),(2),(3),(3),(3);
select id, count(id) from test_table group by id order by id desc;
```

2. Unggah `hive-query.ql` ke bucket S3 Anda dengan perintah berikut.

```
aws s3 cp hive-query.ql s3://DOC-EXAMPLE-BUCKET/emr-serverless-hive/query/hive-query.ql
```

3. Dalam perintah berikut, ganti *`application-id`* dengan ID aplikasi Anda sendiri. Ganti *`job-role-arn`* dengan peran runtime yang ARN Anda buat. Buat peran runtime pekerjaan Ganti semua *`DOC-EXAMPLE-BUCKET`* string dengan bucket Amazon S3 yang Anda buat, lalu `/output` tambahkan `/logs` dan ke path. Ini membuat folder baru di bucket Anda, tempat EMR Tanpa Server dapat menyalin file keluaran dan log aplikasi Anda.

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "hive": {
          "query": "s3://DOC-EXAMPLE-BUCKET/emr-serverless-hive/query/hive-query.ql",
          "parameters": "--hiveconf hive.log.explain.output=false"
        }
    }' \
    --configuration-overrides '{
      "applicationConfiguration": [{
        "classification": "hive-site",
          "properties": {
            "hive.exec.scratchdir": "s3://DOC-EXAMPLE-BUCKET/emr-serverless-hive/hive/scratch",
            "hive.metastore.warehouse.dir": "s3://DOC-EXAMPLE-BUCKET/emr-serverless-hive/hive/warehouse",
            "hive.driver.cores": "2",
            "hive.driver.memory": "4g",
            "hive.tez.container.size": "4096",
            "hive.tez.cpu.vcores": "1"
            }
        }],
        "monitoringConfiguration": {
          "s3MonitoringConfiguration": {
            "logUri": "s3://DOC-EXAMPLE-BUCKET/emr-serverless-hive/logs"
           }
        }
    }'
```

4. Perhatikan ID job run yang dikembalikan dalam output. Ganti *job-run-id* dengan ID ini dalam langkah-langkah berikut.

## Langkah 3: Tinjau output pekerjaan Anda

Jalankan pekerjaan biasanya membutuhkan waktu 3-5 menit untuk menyelesaikannya.

Spark

Anda dapat memeriksa status pekerjaan Spark Anda dengan perintah berikut.

```
aws emr-serverless get-job-run \
    --application-id application-id \
    --job-run-id job-run-id
```

Dengan tujuan log Anda disetel kes3://*DOC-EXAMPLE-BUCKET*/emr-serverless-spark/logs, Anda dapat menemukan log untuk pekerjaan khusus ini berjalan di bawahs3://*DOC-EXAMPLE-BUCKET*/emr-serverless-spark/logs/applications/*application-id*/jobs/*job-run-id*.

Untuk aplikasi Spark, EMR Serverless mendorong log peristiwa setiap 30 detik ke sparklogs folder di tujuan log S3 Anda. Ketika pekerjaan Anda selesai, log runtime Spark untuk driver dan pelaksana mengunggah ke folder yang diberi nama sesuai dengan jenis pekerja, seperti atau. driver executor Output dari PySpark pekerjaan diunggah kes3://*DOC-EXAMPLE-BUCKET*/output/.

Hive

Anda dapat memeriksa status pekerjaan Hive Anda dengan perintah berikut.

```
aws emr-serverless get-job-run \
    --application-id application-id \
    --job-run-id job-run-id
```

Dengan tujuan log Anda disetel kes3://*DOC-EXAMPLE-BUCKET*/emr-serverless-hive/logs, Anda dapat menemukan log untuk pekerjaan khusus ini berjalan di bawahs3://*DOC-EXAMPLE-BUCKET*/emr-serverless-hive/logs/applications/*application-id*/jobs/*job-run-id*.

Untuk aplikasi Hive, EMR Tanpa Server terus mengunggah driver Hive ke `HIVE_DRIVER` folder, dan tugas Tez log ke folder, dari `TEZ_TASK` tujuan log S3 Anda. Setelah job run mencapai `SUCCEEDED` status, output kueri Hive Anda akan tersedia di lokasi Amazon S3 yang Anda tentukan `monitoringConfiguration` di bidang. `configurationOverrides`

## Langkah 4: Membersihkan

Setelah selesai mengerjakan tutorial ini, pertimbangkan untuk menghapus sumber daya yang Anda buat. Kami menyarankan Anda merilis sumber daya yang tidak ingin Anda gunakan lagi.

### Hapus aplikasi Anda

Untuk menghapus aplikasi, gunakan perintah berikut.

```
aws emr-serverless delete-application \
    --application-id application-id
```

### Hapus bucket log S3 Anda

Untuk menghapus bucket logging dan output S3 Anda, gunakan perintah berikut. Ganti *DOC-EXAMPLE-BUCKET* dengan nama sebenarnya dari bucket S3 yang dibuat di.. Siapkan penyimpanan untuk Tanpa EMR Server

```
aws s3 rm s3://DOC-EXAMPLE-BUCKET --recursive
aws s3api delete-bucket --bucket DOC-EXAMPLE-BUCKET
```

### Hapus peran runtime pekerjaan Anda

Untuk menghapus peran runtime, lepaskan kebijakan dari peran. Anda kemudian dapat menghapus peran dan kebijakan.

```
aws iam detach-role-policy \
    --role-name EMRServerlessS3RuntimeRole \
    --policy-arn policy-arn
```

Untuk menghapus peran, gunakan perintah berikut.

```
aws iam delete-role \
```

```
    --role-name EMRServerlessS3RuntimeRole
```

Untuk menghapus kebijakan yang dilampirkan pada peran, gunakan perintah berikut.

```
aws iam delete-policy \
    --policy-arn policy-arn
```

Untuk lebih banyak contoh menjalankan pekerjaan Spark dan Hive, lihat Lowongan kerja Spark dan. Pekerjaan sarang

# Berinteraksi dengan aplikasi

Bagian ini mencakup bagaimana Anda dapat berinteraksi dengan aplikasi Amazon EMR Tanpa Server Anda dengan AWS CLI dan default untuk mesin Spark dan Hive.

Topik



## Status aplikasi

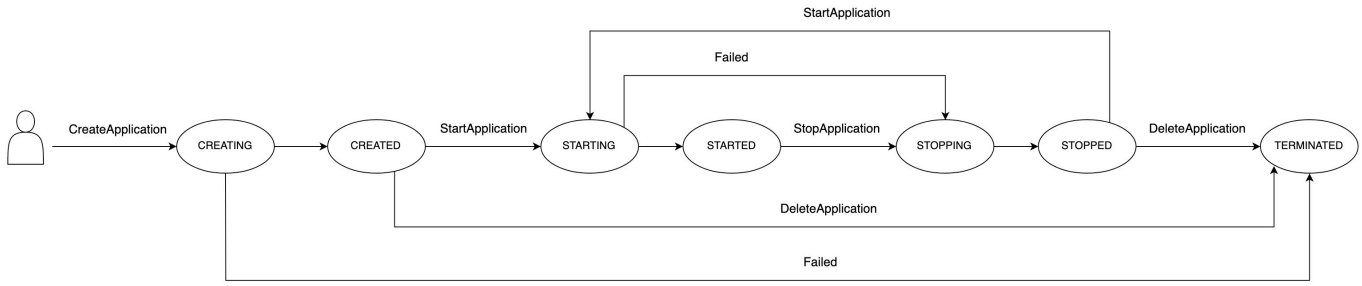
Saat Anda membuat aplikasi dengan EMR Tanpa Server, aplikasi yang dijalankan memasuki status. `CREATING` Kemudian melewati status-status berikut sampai berhasil (keluar dengan kode `0`) atau gagal (keluar dengan kode bukan nol).

Aplikasi dapat memiliki status berikut:

| Status | Deskripsi |
| --- | --- |
| Creating | Aplikasi sedang disiapkan dan belum siap digunakan. |
| Dibuat | Aplikasi telah dibuat tetapi belum menyediakan kapasitas. Anda dapat memodifikasi aplikasi untuk mengubah konfigurasi kapasitas awalnya. |
| Starting | Aplikasi dimulai dan menyediakan kapasitas. |

| Status | Deskripsi |
| --- | --- |
| Dimulai | Aplikasi siap menerima pekerjaan baru. Aplikasi hanya menerima pekerjaan ketika berada di negara bagian ini. |
| Stopping | Semua pekerjaan telah selesai dan aplikasi merilis kapasitasnya. |
| Stopped | Aplikasi dihentikan dan tidak ada sumber daya yang berjalan pada aplikasi. Anda dapat memodifikasi aplikasi untuk mengubah konfigurasi kapasitas awalnya. |
| Diakhiri | Aplikasi telah dihentikan dan tidak muncul di daftar aplikasi Anda. |

Diagram berikut menunjukkan lintasan status aplikasi Tanpa EMR Server.




# Berinteraksi dengan aplikasi Anda dari konsol EMR Studio

Dari konsol EMR Studio, Anda dapat membuat, melihat, dan mengelola aplikasi EMR Tanpa Server. Untuk menavigasi ke konsol EMR Studio, ikuti petunjuk di Memulai dari konsol.

## Membuat aplikasi

Dengan halaman Buat aplikasi, Anda dapat membuat aplikasi EMR Tanpa Server dengan mengikuti langkah-langkah ini.

1. Di bidang Nama, masukkan nama yang ingin Anda panggil aplikasi Anda.
2. Di bidang Type, pilih Spark atau Hive sebagai jenis aplikasi.
3. Di bidang Versi rilis, pilih nomor EMR rilis.
4. Dalam opsi Arsitektur, pilih arsitektur set instruksi yang akan digunakan. Untuk informasi selengkapnya, lihat Opsi EMR arsitektur Amazon Tanpa Server.

   - arm64 - ARM arsitektur 64-bit; untuk menggunakan prosesor Graviton
   - x86_64 - arsitektur x86 64-bit; untuk menggunakan prosesor berbasis x86

5. Ada dua opsi pengaturan aplikasi untuk bidang yang tersisa: pengaturan default dan pengaturan khusus. Bidang ini opsional.

   Pengaturan default - Pengaturan default memungkinkan Anda membuat aplikasi dengan cepat dengan kapasitas pra-inisialisasi. Ini termasuk satu driver dan satu eksekutor untuk Spark, dan satu driver dan satu Tez Task for Hive. Pengaturan default tidak mengaktifkan konektivitas jaringan ke AndaVPCs. Aplikasi dikonfigurasi untuk berhenti jika idle selama 15 menit, dan otomatis dimulai pada pengiriman pekerjaan.

   Pengaturan kustom - Pengaturan kustom memungkinkan Anda untuk memodifikasi properti berikut.

   - Kapasitas pra-inisialisasi — Jumlah driver dan pelaksana atau pekerja Hive Tez Task, dan ukuran setiap pekerja.
   - Batas aplikasi — Kapasitas maksimum aplikasi.
   - Perilaku aplikasi — Perilaku auto-start dan auto-stop aplikasi.
   - Koneksi jaringan — Konektivitas jaringan ke VPC sumber daya.
   - Tag - Tag khusus yang dapat Anda tetapkan ke aplikasi.

   Untuk informasi selengkapnya tentang kapasitas pra-inisialisasi, batasan aplikasi, dan perilaku aplikasi, lihat. Mengkonfigurasi aplikasi Untuk informasi selengkapnya tentang konektivitas jaringan, lihatMengkonfigurasi akses VPC.

6. Untuk membuat aplikasi, pilih Buat aplikasi.

## Daftar aplikasi

Anda dapat melihat semua aplikasi EMR Tanpa Server yang ada dari halaman Daftar aplikasi. Anda dapat memilih nama aplikasi untuk menavigasi ke halaman Detail untuk aplikasi itu.

## Mengelola aplikasi

Anda dapat melakukan tindakan berikut pada aplikasi baik dari halaman Daftar aplikasi atau dari halaman Detail aplikasi tertentu.

Mulai aplikasi

Pilih opsi ini untuk memulai aplikasi secara manual.

Hentikan aplikasi

Pilih opsi ini untuk menghentikan aplikasi secara manual. Aplikasi seharusnya tidak memiliki pekerjaan yang berjalan agar dapat dihentikan. Untuk mempelajari lebih lanjut tentang transisi status aplikasi, lihatStatus aplikasi.

Konfigurasikan aplikasi

Edit pengaturan opsional untuk aplikasi dari halaman Konfigurasi aplikasi. Anda dapat mengubah sebagian besar pengaturan aplikasi. Misalnya, Anda dapat mengubah label rilis untuk aplikasi untuk memutakhirkannya ke versi Amazon yang berbedaEMR, atau Anda dapat mengganti arsitektur dari x86_64 ke arm64. Pengaturan opsional lainnya sama dengan yang ada di bagian Pengaturan kustom di halaman Buat aplikasi. Untuk informasi selengkapnya tentang pengaturan aplikasi, lihatMembuat aplikasi.

Hapus aplikasi

Pilih opsi ini untuk menghapus aplikasi secara manual. Anda harus menghentikan aplikasi untuk menghapusnya. Untuk mempelajari lebih lanjut tentang transisi status aplikasi, lihatStatus aplikasi.

# Berinteraksi dengan aplikasi Anda di AWS CLI

Dari AWS CLI, Anda dapat membuat, mendeskripsikan, dan menghapus aplikasi individual. Anda juga dapat membuat daftar semua aplikasi Anda sehingga Anda dapat melihatnya sekilas. Bagian ini menjelaskan cara melakukan tindakan ini. Untuk operasi aplikasi lainnya, seperti memulai, menghentikan, dan memperbarui aplikasi Anda, lihat EMRReferensi Tanpa Server API. Untuk contoh

cara menggunakan EMR Serverless API menggunakan AWS SDK for Java, lihat contoh Java di GitHub repositori kami. Untuk contoh cara menggunakan EMR Serverless API menggunakan AWS SDK for Python (Boto), lihat contoh Python di repositori kami GitHub .

Untuk membuat aplikasi, gunakan`create-application`. Anda harus menentukan `SPARK` atau `HIVE` sebagai aplikasi`type`. Perintah ini mengembalikan aplikasiARN, nama, dan ID.

```
aws emr-serverless create-application \
--name my-application-name \
--type 'application-type' \
--release-label release-version
```

Untuk mendeskripsikan aplikasi, gunakan `get-application` dan sediakan`application-id`. Perintah ini mengembalikan status dan konfigurasi terkait kapasitas untuk aplikasi Anda.

```
aws emr-serverless get-application \
--application-id application-id
```

Untuk membuat daftar semua aplikasi Anda, hubungi`list-applications`. Perintah ini mengembalikan properti yang sama seperti `get-application` tetapi mencakup semua aplikasi Anda.

```
aws emr-serverless list-applications
```

Untuk menghapus aplikasi Anda, hubungi `delete-application` dan sediakan aplikasi Anda`application-id`.

```
aws emr-serverless delete-application \
--application-id application-id
```

## Mengkonfigurasi aplikasi

Dengan EMR Tanpa Server, Anda dapat mengonfigurasi aplikasi yang Anda gunakan. Misalnya, Anda dapat mengatur kapasitas maksimum yang dapat ditingkatkan oleh aplikasi, mengonfigurasi kapasitas pra-inisialisasi agar pengemudi dan pekerja siap merespons, dan menentukan serangkaian konfigurasi runtime dan pemantauan umum di tingkat aplikasi. Halaman berikut menjelaskan cara mengkonfigurasi aplikasi saat Anda menggunakan Tanpa EMR Server.

Topik



# Memahami perilaku aplikasi

## Perilaku aplikasi default

Mulai otomatis - Aplikasi secara default dikonfigurasi untuk memulai otomatis pada pengiriman pekerjaan. Anda dapat menonaktifkan fitur ini.

Auto-stop — Aplikasi secara default dikonfigurasi untuk berhenti otomatis saat idle selama 15 menit. Ketika aplikasi berubah ke `STOPPED` status, ia melepaskan kapasitas pra-diinisialisasi yang dikonfigurasi. Anda dapat mengubah jumlah waktu idle sebelum aplikasi berhenti otomatis, atau Anda dapat menonaktifkan fitur ini.

## Kapasitas maksimal

Anda dapat mengonfigurasi kapasitas maksimum yang dapat ditingkatkan oleh aplikasi. Anda dapat menentukan kapasitas maksimum Anda dalam halCPU, memori (GB), dan disk (GB).

> **Note**
>
> Sebaiknya konfigurasi kapasitas maksimum Anda agar proporsional dengan ukuran pekerja yang didukung dengan mengalikan jumlah pekerja dengan ukurannya. Misalnya, jika Anda ingin membatasi aplikasi Anda ke 50 pekerja dengan 2vCPUs, 16 GB untuk memori, dan 20 GB untuk disk, atur kapasitas maksimum Anda menjadi 100vCPUs, 800 GB untuk memori, dan 1000 GB untuk disk.

## Konfigurasi pekerja yang didukung

Tabel berikut menunjukkan konfigurasi dan ukuran pekerja yang didukung yang dapat Anda tentukan untuk Tanpa EMR Server. Anda dapat mengonfigurasi berbagai ukuran untuk driver dan pelaksana berdasarkan kebutuhan beban kerja Anda.

| CPU | Memori | Penyimpanan fana default |
| --- | --- | --- |
| 1 v CPU | Minimum 2 GB, maksimum 8 GB, dengan peningkatan 1 GB | 20 GB - 200 GB |
| 2 v CPU | Minimum 4 GB, maksimum 16 GB, dengan peningkatan 1 GB | 20 GB - 200 GB |
| 4 v CPU | Minimum 8 GB, maksimum 30 GB, dengan peningkatan 1 GB | 20 GB - 200 GB |
| 8 v CPU | Minimum 16 GB, maksimum 60 GB, dengan peningkatan 4 GB | 20 GB - 200 GB |
| 16 v CPU | Minimum 32 GB, maksimum 120 GB, dengan peningkatan 8 GB | 20 GB - 200 GB |

CPUSetiap pekerja dapat memiliki 1, 2, 4, 8, atau 16vCPUs.

Memori — Setiap pekerja memiliki memori, ditentukan dalam GB, dalam batas yang tercantum dalam tabel sebelumnya. Pekerjaan percikan memiliki overhead memori, yang berarti bahwa memori yang mereka gunakan lebih dari ukuran wadah yang ditentukan. Overhead ini ditentukan dengan properti `spark.driver.memory0verhead` dan`spark.executor.memory0verhead`. Overhead memiliki nilai default 10% dari memori kontainer, dengan minimal 384 MB. Anda harus mempertimbangkan overhead ini ketika Anda memilih ukuran pekerja.

Misalnya, Jika Anda memilih 4 vCPUs untuk instance pekerja Anda, dan kapasitas penyimpanan pra-inisialisasi 30 GB, maka Anda harus menetapkan nilai sekitar 27 GB sebagai memori pelaksana untuk pekerjaan Spark Anda. Ini memaksimalkan pemanfaatan kapasitas pra-inisialisasi Anda. Memori yang dapat digunakan adalah 27 GB, ditambah 10% dari 27 GB (2,7 GB), dengan total 29, 7 GB.

Disk — Anda dapat mengonfigurasi setiap pekerja dengan disk penyimpanan sementara dengan ukuran minimum 20 GB dan maksimum 200 GB. Anda hanya membayar penyimpanan tambahan melebihi 20 GB yang Anda konfigurasikan per pekerja.

# Kapasitas pra-inisialisasi

EMRTanpa server menyediakan fitur opsional yang membuat pengemudi dan pekerja diinisialisasi sebelumnya dan siap merespons dalam hitungan detik. Ini secara efektif menciptakan kumpulan pekerja yang hangat untuk aplikasi. Fitur ini disebut kapasitas pra-inisialisasi. Untuk mengonfigurasi fitur ini, Anda dapat mengatur `initialCapacity` parameter aplikasi ke jumlah pekerja yang ingin Anda inisialisasi sebelumnya. Dengan kapasitas pekerja pra-inisialisasi, pekerjaan segera dimulai. Ini sangat ideal ketika Anda ingin menerapkan aplikasi berulang dan pekerjaan yang sensitif terhadap waktu.

Ketika Anda mengirimkan pekerjaan, jika pekerja dari `initialCapacity` tersedia, pekerjaan menggunakan sumber daya tersebut untuk memulai operasinya. Jika pekerja tersebut sudah digunakan oleh pekerjaan lain, atau jika pekerjaan membutuhkan lebih banyak sumber daya daripada yang tersedia`initialCapacity`, maka aplikasi meminta dan mendapatkan pekerja tambahan, hingga batas maksimum sumber daya yang ditetapkan untuk aplikasi. Ketika pekerjaan selesai dijalankan, ia melepaskan pekerja yang digunakannya, dan jumlah sumber daya yang tersedia untuk aplikasi kembali. `initialCapacity` Aplikasi mempertahankan sumber `initialCapacity` daya bahkan setelah pekerjaan selesai berjalan. Aplikasi melepaskan kelebihan sumber daya di luar `initialCapacity` ketika pekerjaan tidak lagi membutuhkannya untuk dijalankan.

Kapasitas pra-inisialisasi tersedia dan siap digunakan ketika aplikasi telah dimulai. Kapasitas pra-inisialisasi menjadi tidak aktif ketika aplikasi dihentikan. Aplikasi pindah ke `STARTED` status hanya jika kapasitas pra-inisialisasi yang diminta telah dibuat dan siap digunakan. Sepanjang waktu aplikasi dalam `STARTED` keadaan, EMR Tanpa Server menjaga kapasitas pra-inisialisasi tersedia untuk digunakan atau digunakan oleh pekerjaan atau beban kerja interaktif. Fitur ini mengembalikan kapasitas untuk kontainer yang dirilis atau gagal. Ini mempertahankan jumlah pekerja yang ditentukan `InitialCapacity` parameter. Keadaan aplikasi tanpa kapasitas pra-inisialisasi dapat segera berubah dari `CREATED` ke. `STARTED`

Anda dapat mengonfigurasi aplikasi untuk melepaskan kapasitas pra-inisialisasi jika tidak digunakan untuk jangka waktu tertentu, dengan default 15 menit. Aplikasi yang dihentikan dimulai secara otomatis saat Anda mengirimkan pekerjaan baru. Anda dapat mengatur konfigurasi mulai dan berhenti otomatis ini saat Anda membuat aplikasi, atau Anda dapat mengubahnya saat aplikasi dalam `STOPPED` keadaan `CREATED` atau.

Anda dapat mengubah `InitialCapacity` hitungan, dan menentukan konfigurasi komputasi sepertiCPU, memori, dan disk, untuk setiap pekerja. Karena Anda tidak dapat membuat modifikasi

sebagian, Anda harus menentukan semua konfigurasi komputasi saat Anda mengubah nilai. Anda hanya dapat mengubah konfigurasi saat aplikasi dalam `STOPPED` status `CREATED` atau.

> **Note**
>
> Untuk mengoptimalkan penggunaan sumber daya aplikasi Anda, kami sarankan untuk menyelaraskan ukuran kontainer Anda dengan ukuran pekerja kapasitas yang telah diinisialisasi sebelumnya. Misalnya, jika Anda mengonfigurasi ukuran pelaksana Spark Anda menjadi 2 CPUs dan memori Anda menjadi 8 GB, tetapi ukuran pekerja kapasitas pra-inisialisasi Anda adalah 4 CPUs dengan memori 16 GB, maka pelaksana Spark hanya menggunakan setengah dari sumber daya pekerja saat mereka ditugaskan untuk pekerjaan ini.

## Menyesuaikan kapasitas pra-inisialisasi untuk Spark dan Hive

Anda dapat lebih lanjut menyesuaikan kapasitas pra-inisialisasi untuk beban kerja yang berjalan pada kerangka kerja data besar tertentu. Misalnya, ketika beban kerja berjalan di Apache Spark, Anda dapat menentukan berapa banyak pekerja yang memulai sebagai driver dan berapa banyak yang memulai sebagai pelaksana. Demikian pula, ketika Anda menggunakan Apache Hive, Anda dapat menentukan berapa banyak pekerja yang memulai sebagai driver Hive, dan berapa banyak yang harus menjalankan tugas Tez.

Mengkonfigurasi aplikasi yang menjalankan Apache Hive dengan kapasitas pra-inisialisasi

APIPermintaan berikut membuat aplikasi yang menjalankan Apache Hive berdasarkan EMR rilis Amazon emr-6.6.0. Aplikasi dimulai dengan 5 driver Hive pra-inisialisasi, masing-masing dengan memori 2 v CPU dan 4 GB, dan 50 pekerja tugas Tez pra-inisialisasi, masing-masing dengan memori 4 v CPU dan 8 GB. Ketika Hive query berjalan pada aplikasi ini, mereka pertama kali menggunakan pekerja pra-inisialisasi dan mulai mengeksekusi segera. Jika semua pekerja pra-inisialisasi sibuk dan lebih banyak pekerjaan Hive diajukan, aplikasi dapat menskalakan ke total memori 400 v CPU dan 1024 GB. Anda dapat secara opsional menghilangkan kapasitas untuk pekerja `DRIVER` atau pekerja. `TEZ_TASK`

```
aws emr-serverless create-application \
  --type "HIVE" \
  --name my-application-name \
  --release-label emr-6.6.0 \
  --initial-capacity '{
```

```
    "DRIVER": {
        "workerCount": 5,
        "workerConfiguration": {
            "cpu": "2vCPU",
            "memory": "4GB"
        }
    },
    "TEZ_TASK": {
        "workerCount": 50,
        "workerConfiguration": {
            "cpu": "4vCPU",
            "memory": "8GB"
        }
    }
  }' \
  --maximum-capacity '{
    "cpu": "400vCPU",
    "memory": "1024GB"
  }'
```

Mengkonfigurasi aplikasi yang menjalankan Apache Spark dengan kapasitas pra-inisialisasi

APIPermintaan berikut membuat aplikasi yang menjalankan Apache Spark 3.2.0 berdasarkan Amazon EMR rilis 6.6.0. Aplikasi dimulai dengan 5 driver Spark pra-inisialisasi, masing-masing dengan memori 2 v CPU dan 4 GB, dan 50 pelaksana pra-inisialisasi, masing-masing dengan memori 4 v dan 8 GB. CPU Ketika pekerjaan Spark dijalankan pada aplikasi ini, mereka pertama kali menggunakan pekerja pra-inisialisasi dan mulai mengeksekusi segera. Jika semua pekerja pra-inisialisasi sibuk dan lebih banyak pekerjaan Spark diajukan, aplikasi dapat menskalakan ke total memori 400 v CPU dan 1024 GB. Anda dapat secara opsional menghilangkan kapasitas untuk `DRIVER` atau. `EXECUTOR`

> **Note**
>
> Spark menambahkan overhead memori yang dapat dikonfigurasi, dengan nilai default 10%, ke memori yang diminta untuk driver dan pelaksana. Agar pekerjaan dapat menggunakan pekerja pra-inisialisasi, konfigurasi memori kapasitas awal harus lebih besar daripada memori yang diminta pekerjaan dan overhead.

```
aws emr-serverless create-application \
  --type "SPARK" \
```

```
  --name my-application-name \
  --release-label emr-6.6.0 \
  --initial-capacity '{
    "DRIVER": {
        "workerCount": 5,
        "workerConfiguration": {
            "cpu": "2vCPU",
            "memory": "4GB"
        }
    },
    "EXECUTOR": {
        "workerCount": 50,
        "workerConfiguration": {
            "cpu": "4vCPU",
            "memory": "8GB"
        }
    }
  }' \
  --maximum-capacity '{
    "cpu": "400vCPU",
    "memory": "1024GB"
  }'
```

## Konfigurasi aplikasi default untuk Tanpa EMR Server

Anda dapat menentukan satu set umum runtime dan konfigurasi pemantauan di tingkat aplikasi untuk semua pekerjaan yang Anda kirimkan di bawah aplikasi yang sama. Ini mengurangi overhead tambahan yang terkait dengan kebutuhan untuk mengirimkan konfigurasi yang sama untuk setiap pekerjaan.

Anda dapat memodifikasi konfigurasi pada titik-titik waktu berikut:

- Deklarasikan konfigurasi tingkat aplikasi saat pengajuan pekerjaan.
- Ganti konfigurasi default selama menjalankan pekerjaan.

Bagian berikut memberikan rincian lebih lanjut dan contoh untuk konteks lebih lanjut.

### Mendeklarasikan konfigurasi di tingkat aplikasi

Anda dapat menentukan pencatatan tingkat aplikasi dan properti konfigurasi runtime untuk pekerjaan yang Anda kirimkan di bawah aplikasi.

**`monitoringConfiguration`**

Untuk menentukan konfigurasi log untuk pekerjaan yang Anda kirimkan dengan aplikasi, gunakan `monitoringConfiguration`bidang. Untuk informasi selengkapnya tentang pencatatan untuk EMR Tanpa Server, lihat. Menyimpan log

**`runtimeConfiguration`**

Untuk menentukan properti konfigurasi runtime seperti`spark-defaults`, berikan objek konfigurasi di `runtimeConfiguration` bidang. Ini memengaruhi konfigurasi default untuk semua pekerjaan yang Anda kirimkan dengan aplikasi. Untuk informasi selengkapnya, silakan lihat Parameter penggantian konfigurasi sarang dan Parameter penggantian konfigurasi percikan.

Klasifikasi konfigurasi yang tersedia bervariasi menurut rilis Tanpa EMR Server tertentu. Misalnya, klasifikasi untuk Log4j kustom `spark-driver-log4j2` dan hanya `spark-executor-log4j2` tersedia dengan rilis 6.8.0 dan yang lebih tinggi. Untuk daftar properti khusus aplikasi, lihat Properti pekerjaan percikan dan. Properti pekerjaan sarang

Anda juga dapat mengkonfigurasi properti Apache Log4j2, AWS Secrets Manager untuk perlindungan data, dan runtime Java 17 di tingkat aplikasi.

Untuk meneruskan rahasia Secrets Manager di tingkat aplikasi, lampirkan kebijakan berikut kepada pengguna dan peran yang perlu membuat atau memperbarui aplikasi EMR Tanpa Server dengan rahasia.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "SecretsManagerPolicy",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue",
                "secretsmanager:DescribeSecret",
                "kms:Decrypt"
            ],
            "Resource": "arn:aws:secretsmanager:your-secret-arn"
        }
    ]
}
```

Untuk informasi selengkapnya tentang membuat kebijakan khusus untuk rahasia, lihat contoh kebijakan izin untuk AWS Secrets Managerdi AWS Secrets Manager Panduan Pengguna.

> **Note**
>
> `runtimeConfiguration`Yang Anda tentukan di peta tingkat aplikasi ke `applicationConfiguration` dalam `StartJobRun`API.

Contoh deklarasi

Contoh berikut menunjukkan bagaimana untuk mendeklarasikan konfigurasi default dengan. `create-application`

```
aws emr-serverless create-application \
    --release-label release-version  \
    --type SPARK \
    --name my-application-name \
    --runtime-configuration '[
        {
            "classification": "spark-defaults",
            "properties": {
                "spark.driver.cores": "4",
                "spark.executor.cores": "2",
                "spark.driver.memory": "8G",
                "spark.executor.memory": "8G",
                "spark.executor.instances": "2",

 "spark.hadoop.javax.jdo.option.ConnectionDriverName":"org.mariadb.jdbc.Driver",
                "spark.hadoop.javax.jdo.option.ConnectionURL":"jdbc:mysql://db-host:db-
port/db-name",
                "spark.hadoop.javax.jdo.option.ConnectionUserName":"connection-user-
name",
                "spark.hadoop.javax.jdo.option.ConnectionPassword":
 "EMR.secret@SecretID"
            }
        },
        {
            "classification": "spark-driver-log4j2",
            "properties": {
                "rootLogger.level":"error",
```

```
                "logger.IdentifierForClass.name": "classpathForSettingLogger",
                "logger.IdentifierForClass.level": "info"
            }
        }
    ]' \
    --monitoring-configuration '{
        "s3MonitoringConfiguration": {
            "logUri": "s3://DOC-EXAMPLE-BUCKET-LOGGING/logs/app-level"
        },
        "managedPersistenceMonitoringConfiguration": {
            "enabled": false
        }
    }'
```

## Mengganti konfigurasi selama menjalankan pekerjaan

Anda dapat menentukan penggantian konfigurasi untuk konfigurasi aplikasi dan konfigurasi pemantauan dengan. `StartJobRun`API EMRServerless kemudian menggabungkan konfigurasi yang Anda tentukan di tingkat aplikasi dan tingkat pekerjaan untuk menentukan konfigurasi untuk pelaksanaan pekerjaan.

Tingkat granularitas saat penggabungan terjadi adalah sebagai berikut:

- **`ApplicationConfiguration`**- Jenis klasifikasi, misalnya`spark-defaults`.
- **`MonitoringConfiguration`**- Jenis konfigurasi, misalnya`s3MonitoringConfiguration`.

> **Note**
>
> Prioritas konfigurasi yang Anda berikan `StartJobRun`menggantikan konfigurasi yang Anda berikan di tingkat aplikasi.

Untuk informasi lebih lanjut peringkat prioritas, lihat Parameter penggantian konfigurasi sarang danParameter penggantian konfigurasi percikan.

Ketika Anda memulai pekerjaan, jika Anda tidak menentukan konfigurasi tertentu, itu akan diwarisi dari aplikasi. Jika Anda mendeklarasikan konfigurasi di tingkat pekerjaan, Anda dapat melakukan operasi berikut:

- Ganti konfigurasi yang ada - Berikan parameter konfigurasi yang sama dalam `StartJobRun` permintaan dengan nilai override Anda.
- Tambahkan konfigurasi tambahan - Tambahkan parameter konfigurasi baru dalam `StartJobRun` permintaan dengan nilai yang ingin Anda tentukan.
- Hapus konfigurasi yang ada - Untuk menghapus konfigurasi runtime aplikasi, berikan kunci untuk konfigurasi yang ingin Anda hapus, dan berikan deklarasi kosong `{}` untuk konfigurasi. Kami tidak menyarankan untuk menghapus klasifikasi apa pun yang berisi parameter yang diperlukan untuk menjalankan pekerjaan. Misalnya, jika Anda mencoba menghapus properti yang diperlukan untuk pekerjaan Hive, pekerjaan itu akan gagal.

  Untuk menghapus konfigurasi pemantauan aplikasi, gunakan metode yang sesuai untuk jenis konfigurasi yang relevan:

  - **`cloudWatchLoggingConfiguration`**- Untuk menghapus`cloudWatchLogging`, berikan bendera yang diaktifkan sebagai`false`.
  - **`managedPersistenceMonitoringConfiguration`**- Untuk menghapus pengaturan persistensi terkelola dan kembali ke status default yang diaktifkan, berikan deklarasi kosong `{}` untuk konfigurasi.
  - **`s3MonitoringConfiguration`**- Untuk menghapus`s3MonitoringConfiguration`, berikan deklarasi kosong `{}` untuk konfigurasi.

Contoh penggantian

Contoh berikut menunjukkan operasi yang berbeda yang dapat Anda lakukan selama pengajuan pekerjaan di`start-job-run`.

```
aws emr-serverless start-job-run \
    --application-id your-application-id \
    --execution-role-arn your-job-role-arn \
    --job-driver '{
        "sparkSubmit": {
            "entryPoint": "s3://us-east-1.elasticmapreduce/emr-containers/samples/
wordcount/scripts/wordcount.py",
            "entryPointArguments": ["s3://DOC-EXAMPLE-BUCKET-OUTPUT/wordcount_output"]
        }
    }' \
    --configuration-overrides '{
        "applicationConfiguration": [
            {
```

```
                // Override existing configuration for spark-defaults in the
application
                "classification": "spark-defaults",
                "properties": {
                    "spark.driver.cores": "2",
                    "spark.executor.cores": "1",
                    "spark.driver.memory": "4G",
                    "spark.executor.memory": "4G"
                }
            },
            {
                // Add configuration for spark-executor-log4j2
                "classification": "spark-executor-log4j2",
                "properties": {
                    "rootLogger.level": "error",
                    "logger.IdentifierForClass.name": "classpathForSettingLogger",
                    "logger.IdentifierForClass.level": "info"
                }
            },
            {
                // Remove existing configuration for spark-driver-log4j2 from the
application
                "classification": "spark-driver-log4j2",
                "properties": {}
            }
        ],
        "monitoringConfiguration": {
            "managedPersistenceMonitoringConfiguration": {
                // Override existing configuration for managed persistence
                "enabled": true
            },
            "s3MonitoringConfiguration": {
                // Remove configuration of S3 monitoring
            },
            "cloudWatchLoggingConfiguration": {
                // Add configuration for CloudWatch logging
                "enabled": true
            }
        }
    }'
```

Pada saat pelaksanaan pekerjaan, klasifikasi dan konfigurasi berikut akan berlaku berdasarkan peringkat penggantian prioritas yang dijelaskan dalam dan. Parameter penggantian konfigurasi sarang Parameter penggantian konfigurasi percikan

- Klasifikasi `spark-defaults` akan diperbarui dengan properti yang ditentukan di tingkat pekerjaan. Hanya properti yang termasuk dalam `StartJobRun` yang akan dipertimbangkan untuk klasifikasi ini.
- Klasifikasi `spark-executor-log4j2` akan ditambahkan dalam daftar klasifikasi yang ada.
- Klasifikasi `spark-driver-log4j2` akan dihapus.
- Konfigurasi untuk `managedPersistenceMonitoringConfiguration` akan diperbarui dengan konfigurasi di tingkat pekerjaan.
- Konfigurasi untuk `s3MonitoringConfiguration` akan dihapus.
- Konfigurasi untuk `cloudWatchLoggingConfiguration` akan ditambahkan ke konfigurasi pemantauan yang ada.

## Menyesuaikan gambar Tanpa EMR Server

Dimulai dengan Amazon EMR 6.9.0, Anda dapat menggunakan gambar khusus untuk mengemas dependensi aplikasi dan lingkungan runtime ke dalam satu wadah dengan Amazon Serverless. EMR Ini menyederhanakan cara Anda mengelola dependensi beban kerja dan membuat paket Anda lebih portabel. Saat Anda menyesuaikan gambar EMR Tanpa Server Anda, ini memberikan manfaat berikut:

- Menginstal dan mengonfigurasi paket yang dioptimalkan untuk beban kerja Anda. Paket-paket ini mungkin tidak tersedia secara luas di distribusi publik lingkungan EMR runtime Amazon.
- Mengintegrasikan EMR Tanpa Server dengan proses pembuatan, pengujian, dan penerapan yang sudah ada saat ini dalam organisasi Anda, termasuk pengembangan dan pengujian lokal.
- Menerapkan proses keamanan yang telah ditetapkan, seperti pemindaian gambar, yang memenuhi persyaratan kepatuhan dan tata kelola dalam organisasi Anda.
- Memungkinkan Anda menggunakan versi JDK dan Python Anda sendiri untuk aplikasi Anda.

EMRServerless menyediakan gambar yang dapat Anda gunakan sebagai basis saat membuat gambar sendiri. Gambar dasar menyediakan toples, konfigurasi, dan pustaka penting bagi gambar untuk berinteraksi dengan Tanpa EMR Server. Anda dapat menemukan gambar dasar di Galeri ECR

Publik Amazon. Gunakan gambar yang cocok dengan jenis aplikasi Anda (Spark atau Hive) dan versi rilis. Misalnya, jika Anda membuat aplikasi di Amazon EMR rilis 6.9.0, gunakan gambar berikut.

| Tipe | Citra |
| --- | --- |
| Spark | `public.ecr.aws/emr-serverless/spark/emr-6.9.0:latest` |
| Hive | `public.ecr.aws/emr-serverless/hive/emr-6.9.0:latest` |

## Prasyarat

Sebelum Anda membuat gambar kustom EMR Tanpa Server, lengkapi prasyarat ini.

1. Buat ECR repositori Amazon dalam hal yang sama Wilayah AWS yang Anda gunakan untuk meluncurkan aplikasi EMR Tanpa Server. Untuk membuat repositori ECR pribadi Amazon, lihat Membuat repositori pribadi.
2. Untuk memberi pengguna akses ke ECR repositori Amazon Anda, tambahkan kebijakan berikut ke pengguna dan peran yang membuat atau memperbarui aplikasi EMR Tanpa Server dengan gambar dari repositori ini.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ECRRepositoryListGetPolicy",
            "Effect": "Allow",
            "Action": [
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage",
                "ecr:DescribeImages"
            ],
            "Resource": "ecr-repository-arn"
        }
    ]
}
```

Untuk lebih banyak contoh kebijakan berbasis ECR identitas Amazon, lihat contoh kebijakan berbasis identitas Amazon Elastic Container Registry.

## Langkah 1: Buat gambar khusus dari gambar dasar EMR Tanpa Server

Pertama, buat Dockerfile yang dimulai dengan `FROM` instruksi yang menggunakan gambar dasar pilihan Anda. Setelah `FROM` instruksi, Anda dapat memasukkan modifikasi apa pun yang ingin Anda buat pada gambar. Gambar dasar secara otomatis menyetel `USER` kehadoop. Pengaturan ini mungkin tidak memiliki izin untuk semua modifikasi yang Anda sertakan. Sebagai solusinya, atur `USER` ke`root`, ubah gambar Anda, lalu atur kembali ke`USER`. `hadoop:hadoop` Untuk melihat sampel untuk kasus penggunaan umum, lihatMenggunakan gambar kustom dengan EMR Serverless.

```
# Dockerfile
FROM public.ecr.aws/emr-serverless/spark/emr-6.9.0:latest

USER root
# MODIFICATIONS GO HERE

# EMRS will run the image as hadoop
USER hadoop:hadoop
```

Setelah Anda memiliki Dockerfile, buat gambar dengan perintah berikut.

```
# build the docker image
docker build . -t aws-account-id.dkr.ecr.region.amazonaws.com/my-
repository[:tag]or[@digest]
```

## Langkah 2: Validasi gambar secara lokal

EMRServerless menyediakan alat offline yang dapat memeriksa gambar kustom Anda secara statis untuk memvalidasi file dasar, variabel lingkungan, dan konfigurasi gambar yang benar. Untuk informasi tentang cara menginstal dan menjalankan alat, lihat Amazon EMR Serverless Image. CLI GitHub

Setelah Anda menginstal alat, jalankan perintah berikut untuk memvalidasi gambar:

```
amazon-emr-serverless-image \
validate-image -r emr-6.9.0 -t spark \
```

```
-i aws-account-id.dkr.ecr.region.amazonaws.com/my-repository:tag/@digest
```

Anda akan melihat output yang mirip dengan berikut ini.

```
Amazon EMR Serverless - Image CLI
Version: 0.0.1
... Checking if docker cli is installed
... Checking Image Manifest
[INFO] Image ID: 9e2f4359cf5beb466a8a2ed047ab61c9d37786c555655fc122272758f761b41a
[INFO] Created On: 2022-12-02T07:46:42.586249984Z
[INFO] Default User Set to hadoop:hadoop : PASS
[INFO] Working Directory Set to  : PASS
[INFO] Entrypoint Set to /usr/bin/entrypoint.sh : PASS
[INFO] HADOOP_HOME is set with value: /usr/lib/hadoop : PASS
[INFO] HADOOP_LIBEXEC_DIR is set with value: /usr/lib/hadoop/libexec : PASS
[INFO] HADOOP_USER_HOME is set with value: /home/hadoop : PASS
[INFO] HADOOP_YARN_HOME is set with value: /usr/lib/hadoop-yarn : PASS
[INFO] HIVE_HOME is set with value: /usr/lib/hive : PASS
[INFO] JAVA_HOME is set with value: /etc/alternatives/jre : PASS
[INFO] TEZ_HOME is set with value: /usr/lib/tez : PASS
[INFO] YARN_HOME is set with value: /usr/lib/hadoop-yarn : PASS
[INFO] File Structure Test for hadoop-files in /usr/lib/hadoop: PASS
[INFO] File Structure Test for hadoop-jars in /usr/lib/hadoop/lib: PASS
[INFO] File Structure Test for hadoop-yarn-jars in /usr/lib/hadoop-yarn: PASS
[INFO] File Structure Test for hive-bin-files in /usr/bin: PASS
[INFO] File Structure Test for hive-jars in /usr/lib/hive/lib: PASS
[INFO] File Structure Test for java-bin in /etc/alternatives/jre/bin: PASS
[INFO] File Structure Test for tez-jars in /usr/lib/tez: PASS
----------------------------------------------------------------
Overall Custom Image Validation Succeeded.
----------------------------------------------------------------
```

## Langkah 3: Unggah gambar ke ECR repositori Amazon Anda

Dorong ECR gambar Amazon Anda ke ECR repositori Amazon Anda dengan perintah berikut. Pastikan Anda memiliki IAM izin yang benar untuk mendorong gambar ke repositori Anda. Untuk informasi selengkapnya, lihat Mendorong gambar di Panduan ECR Pengguna Amazon.

```
# login to ECR repo
aws ecr get-login-password --region region | docker login --username AWS --password-
stdin aws-account-id.dkr.ecr.region.amazonaws.com
```

```
# push the docker image
docker push aws-account-id.dkr.ecr.region.amazonaws.com/my-repository:tag/@digest
```

## Langkah 4: Buat atau perbarui aplikasi dengan gambar khusus

Pilih AWS Management Console tab atau AWS CLI tab sesuai dengan bagaimana Anda ingin meluncurkan aplikasi Anda, lalu selesaikan langkah-langkah berikut.

Console

1. Masuk ke konsol EMR Studio di https://console.aws.amazon.com/emr. Arahkan ke aplikasi Anda, atau buat aplikasi baru dengan instruksi di Buat aplikasi.
2. Untuk menentukan gambar kustom saat Anda membuat atau memperbarui aplikasi EMR Tanpa Server, pilih Pengaturan khusus di opsi pengaturan aplikasi.
3. Di bagian Pengaturan gambar kustom, pilih kotak centang Gunakan gambar khusus dengan aplikasi ini.
4. Tempelkan ECR gambar Amazon URI ke URI bidang Gambar. EMRTanpa server menggunakan gambar ini untuk semua jenis pekerja untuk aplikasi. Atau, Anda dapat memilih Gambar kustom yang berbeda dan menempelkan ECR gambar Amazon yang berbeda URIs untuk setiap jenis pekerja.

CLI

- Buat aplikasi dengan `image-configuration` parameter. EMRTanpa server menerapkan pengaturan ini ke semua jenis pekerja.

  ```
  aws emr-serverless create-application \
  --release-label emr-6.9.0 \
  --type SPARK \
  --image-configuration '{
      "imageUri": "aws-account-id.dkr.ecr.region.amazonaws.com/my-repository:tag/
  @digest"
  }'
  ```

  Untuk membuat aplikasi dengan pengaturan gambar yang berbeda untuk setiap jenis pekerja, gunakan `worker-type-specifications` parameter.

  ```
  aws emr-serverless create-application \
  ```

```
--release-label emr-6.9.0 \
--type SPARK \
--worker-type-specifications '{
    "Driver": {
        "imageConfiguration": {
            "imageUri": "aws-account-id.dkr.ecr.region.amazonaws.com/my-
repository:tag/@digest"
        }
    },
    "Executor" : {
        "imageConfiguration": {
            "imageUri": "aws-account-id.dkr.ecr.region.amazonaws.com/my-
repository:tag/@digest"
        }
    }
}'
```

Untuk memperbarui aplikasi, gunakan `image-configuration` parameter. EMRTanpa server menerapkan pengaturan ini ke semua jenis pekerja.

```
aws emr-serverless update-application \
--application-id application-id \
--image-configuration '{
    "imageUri": "aws-account-id.dkr.ecr.region.amazonaws.com/my-repository:tag/
@digest"
}'
```

## Langkah 5: Izinkan EMR Tanpa Server untuk mengakses repositori gambar kustom

Tambahkan kebijakan sumber daya berikut ke ECR repositori Amazon untuk mengizinkan prinsipal layanan EMR Tanpa Server menggunakan`get`,`describe`, dan `download` permintaan dari repositori ini.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Emr Serverless Custom Image Support",
      "Effect": "Allow",
```

```
      "Principal": {
        "Service": "emr-serverless.amazonaws.com"
      },
      "Action": [
        "ecr:BatchGetImage",
        "ecr:DescribeImages",
        "ecr:GetDownloadUrlForLayer"
      ],
      "Condition":{
        "StringEquals":{
          "aws:SourceArn": "arn:aws:emr-serverless:region:aws-account-id:/
applications/application-id"
        }
      }
    }
  ]
}
```

Sebagai praktik terbaik keamanan, tambahkan kunci `aws:SourceArn` kondisi ke kebijakan repositori. Kunci kondisi IAM global `aws:SourceArn` memastikan bahwa EMR Tanpa Server menggunakan repositori hanya untuk aplikasi. ARN Untuk informasi selengkapnya tentang kebijakan ECR repositori Amazon, lihat Membuat repositori pribadi.

## Pertimbangan dan batasan

Saat Anda bekerja dengan gambar khusus, pertimbangkan hal berikut:

- Gunakan gambar dasar yang benar yang cocok dengan jenis (Spark atau Hive) dan label rilis (misalnya,`emr-6.9.0`) untuk aplikasi Anda.
- EMRTanpa server mengabaikan `[CMD]` atau `[ENTRYPOINT]` instruksi dalam file Docker. Gunakan instruksi umum dalam file Docker, seperti`[COPY]`,`[RUN]`, dan`[WORKDIR]`.
- Anda tidak harus memodifikasi variabel lingkungan `JAVA_HOMESPARK_HOME`,`HIVE_HOME`, `TEZ_HOME` ketika Anda membuat gambar kustom.
- Gambar khusus tidak boleh melebihi ukuran 5 GB.
- Jika Anda memodifikasi binari atau stoples di gambar EMR dasar Amazon, hal itu dapat menyebabkan kegagalan aplikasi atau peluncuran pekerjaan.
- ECRRepositori Amazon harus sama Wilayah AWS yang Anda gunakan untuk meluncurkan aplikasi EMR Tanpa Server.

# Mengkonfigurasi akses VPC

Anda dapat mengonfigurasi aplikasi EMR Tanpa Server untuk terhubung ke penyimpanan data di dalam AndaVPC, seperti klaster Amazon Redshift, RDS database Amazon, atau bucket Amazon S3 dengan titik akhir. VPC Aplikasi EMR Tanpa Server Anda memiliki konektivitas keluar ke penyimpanan data di dalam Anda. VPC Secara default, EMR Tanpa Server memblokir akses masuk ke aplikasi Anda untuk meningkatkan keamanan.

> **Note**
>
> Anda harus mengkonfigurasi VPC akses jika Anda ingin menggunakan database metastore Hive eksternal untuk aplikasi Anda. Untuk informasi tentang cara mengonfigurasi metastore Hive eksternal, lihat konfigurasi Metastore.

## Buat aplikasi

Pada halaman Buat aplikasi, Anda dapat memilih pengaturan khusus dan menentukanVPC, subnet, dan grup keamanan yang dapat digunakan aplikasi EMR Tanpa Server.

### VPCs

Pilih nama virtual private cloud (VPC) yang berisi penyimpanan data Anda. Halaman Buat aplikasi mencantumkan semua VPCs untuk pilihan Anda Wilayah AWS.

### Subnet

Pilih subnet di dalam VPC yang berisi penyimpanan data Anda. Halaman Buat aplikasi mencantumkan semua subnet untuk penyimpanan data di AndaVPC.

Subnet yang dipilih harus subnet pribadi. Ini berarti bahwa tabel rute terkait untuk subnet tidak boleh memiliki gateway internet.

Untuk konektivitas outbound ke internet, subnet harus memiliki rute keluar menggunakan Gateway. NAT Untuk mengonfigurasi NAT Gateway, lihat Bekerja dengan NAT gateway.

Untuk konektivitas Amazon S3, subnet harus memiliki NAT Gateway atau titik akhir yang dikonfigurasi. VPC Untuk mengonfigurasi VPC titik akhir S3, lihat Membuat titik akhir gateway.

Untuk konektivitas ke yang lain Layanan AWS di luarVPC, seperti Amazon DynamoDB, Anda harus mengonfigurasi titik akhir VPC atau gateway. NAT Untuk mengonfigurasi VPC titik akhir untuk Layanan AWS, lihat Bekerja dengan VPC titik akhir.

Pekerja dapat terhubung ke penyimpanan data di dalam Anda VPC melalui lalu lintas keluar. Secara default, EMR Tanpa Server memblokir akses masuk ke pekerja untuk meningkatkan keamanan.

Saat Anda menggunakan AWS Config, EMR Serverless membuat catatan item elastic network interface untuk setiap pekerja. Untuk menghindari biaya yang terkait dengan sumber daya ini, pertimbangkan untuk mematikan `AWS::EC2::NetworkInterface` AWS Config.

> **Note**
>
> Kami menyarankan Anda memilih beberapa subnet di beberapa Availability Zone. Ini karena subnet yang Anda pilih menentukan Availability Zones yang tersedia untuk aplikasi EMR Tanpa Server untuk diluncurkan. Setiap pekerja akan menggunakan alamat IP pada subnet tempat ia diluncurkan. Harap pastikan bahwa subnet yang ditentukan memiliki alamat IP yang cukup untuk jumlah pekerja yang Anda rencanakan untuk diluncurkan. Untuk informasi lebih lanjut tentang perencanaan subnet, lihatthe section called “Praktik terbaik untuk perencanaan subnet”.

## Grup keamanan

Pilih satu atau beberapa grup keamanan yang dapat berkomunikasi dengan penyimpanan data Anda. Halaman Buat aplikasi mencantumkan semua grup keamanan di AndaVPC. EMRTanpa server mengaitkan grup keamanan ini dengan antarmuka jaringan elastis yang melekat pada subnet Anda. VPC

> **Note**
>
> Kami menyarankan Anda membuat grup keamanan terpisah untuk aplikasi EMR Tanpa Server. Hal ini membuat mengisolasi dan mengelola aturan jaringan lebih efisien. Misalnya, untuk berkomunikasi dengan klaster Amazon Redshift, Anda dapat menentukan aturan lalu lintas antara grup keamanan Redshift dan EMR Tanpa Server, seperti yang ditunjukkan pada contoh di bawah ini.

Example Contoh - Komunikasi dengan cluster Amazon Redshift

1. Tambahkan aturan untuk lalu lintas masuk ke grup keamanan Amazon Redshift dari salah EMR satu grup keamanan Tanpa Server.

| Tipe | Protokol | Rentang port | Sumber |
| --- | --- | --- | --- |
| Semua TCP | TCP | 5439 | `emr-serverless-security-group` |

2. Tambahkan aturan untuk lalu lintas keluar dari salah satu grup keamanan EMR Tanpa Server. Anda dapat melakukan ini dengan salah satu dari dua cara. Pertama, Anda dapat membuka lalu lintas keluar ke semua port.

| Tipe | Protokol | Rentang Port | Tujuan |
| --- | --- | --- | --- |
| Semua Lalu lintas | TCP | ALL | 0.0.0.0/0 |

Atau, Anda dapat membatasi lalu lintas keluar ke cluster Amazon Redshift. Ini berguna hanya ketika aplikasi harus berkomunikasi dengan cluster Amazon Redshift dan tidak ada yang lain.

| Tipe | Protokol | Rentang port | Sumber |
| --- | --- | --- | --- |
| Semua TCP | TCP | 5439 | `redshift-security-group` |

## Konfigurasikan aplikasi

Anda dapat mengubah konfigurasi jaringan untuk aplikasi EMR Tanpa Server yang ada dari halaman Konfigurasi aplikasi.

### Lihat detail pekerjaan

Pada halaman detail Job run, Anda dapat melihat subnet yang digunakan oleh pekerjaan Anda untuk menjalankan tertentu. Perhatikan bahwa pekerjaan hanya berjalan di satu subnet yang dipilih dari subnet yang ditentukan.

## Praktik terbaik untuk perencanaan subnet

AWS sumber daya dibuat dalam subnet yang merupakan bagian dari alamat IP yang tersedia di Amazon. VPC Misalnya, netmask VPC dengan /16 memiliki hingga 65.536 alamat IP yang tersedia yang dapat dipecah menjadi beberapa jaringan yang lebih kecil menggunakan subnet mask. Sebagai contoh, Anda dapat membagi rentang ini menjadi dua subnet dengan masing-masing menggunakan /17 mask dan 32.768 alamat IP yang tersedia. Subnet berada dalam Availability Zone dan tidak dapat menjangkau seluruh zona.

Subnet harus dirancang dengan mengingat batas penskalaan aplikasi EMR Tanpa Server Anda. Misalnya, jika Anda memiliki aplikasi yang meminta 4 vCpu pekerja dan dapat menskalakan hingga 4.000vCpu, maka aplikasi Anda akan membutuhkan paling banyak 1.000 pekerja untuk total 1.000 antarmuka jaringan. Kami menyarankan Anda membuat subnet di beberapa Availability Zone. Hal ini memungkinkan EMR Tanpa Server untuk mencoba kembali pekerjaan Anda atau menyediakan kapasitas pra-inisialisasi di Availability Zone yang berbeda dalam kejadian yang tidak mungkin terjadi ketika Availability Zone gagal. Oleh karena itu, setiap subnet di setidaknya dua Availability Zone harus memiliki lebih dari 1.000 alamat IP yang tersedia.

Anda memerlukan subnet dengan ukuran topeng lebih rendah dari atau sama dengan 22 untuk menyediakan 1.000 antarmuka jaringan. Masker apa pun yang lebih besar dari 22 tidak akan memenuhi persyaratan. Misalnya, subnet mask dari /23 menyediakan 512 alamat IP, sedangkan mask /22 menyediakan 1024 dan mask /21 menyediakan 2048 alamat IP. Di bawah ini adalah contoh 4 subnet dengan/22 mask di netmask /16 yang VPC dapat dialokasikan ke Availability Zones yang berbeda. Ada perbedaan lima antara alamat IP yang tersedia dan yang dapat digunakan karena empat alamat IP pertama dan alamat IP terakhir di setiap subnet dicadangkan oleh AWS.

| ID Subnet | Alamat Subnet | Topeng Subnet | Rentang Alamat IP | Alamat IP yang tersedia | Alamat IP yang Dapat Digunakan |
|---|---|---|---|---|---|
| 1 | 10.0.0.0 | 255.255.252.0/22 | 10.0.0.0 - 10.0.3.255 | 1,024 | 1,019 |

| ID Subnet | Alamat Subnet | Topeng Subnet | Rentang Alamat IP | Alamat IP yang tersedia | Alamat IP yang Dapat Digunakan |
|---|---|---|---|---|---|
| 2 | 10.0.4.0 | 255.255.252.0/22 | 10.0.4.0 - 10.0.7.255 | 1,024 | 1,019 |
| 3 | 10.0.8.0 | 255.255.252.0/22 | 10.0.4.0 - 10.0.7.255 | 1,024 | 1,019 |
| 4 | 10.0.12.0 | 255.255.252.0/22 | 10.0.12.0 - 10.0.15.255 | 1,024 | 1,019 |

Anda harus mengevaluasi apakah beban kerja Anda paling cocok untuk ukuran pekerja yang lebih besar. Menggunakan ukuran pekerja yang lebih besar membutuhkan antarmuka jaringan yang lebih sedikit. Misalnya, menggunakan 16 vCpu pekerja dengan batas penskalaan aplikasi 4.000 vCpu akan membutuhkan paling banyak 250 pekerja untuk total 250 alamat IP yang tersedia untuk menyediakan antarmuka jaringan. Anda memerlukan subnet di beberapa Availability Zone dengan ukuran mask lebih rendah dari atau sama dengan 24 untuk menyediakan 250 antarmuka jaringan. Setiap ukuran topeng yang lebih besar dari 24 menawarkan kurang dari 250 alamat IP.

Jika Anda berbagi subnet di beberapa aplikasi, setiap subnet harus dirancang dengan mengingat batas penskalaan kolektif dari semua aplikasi Anda. Misalnya, jika Anda memiliki 3 aplikasi yang meminta 4 vCpu pekerja dan masing-masing dapat meningkatkan hingga 4000 vCpu dengan kuota berbasis layanan vCpu tingkat akun 12.000, setiap subnet akan membutuhkan 3000 alamat IP yang tersedia. Jika VPC yang ingin Anda gunakan tidak memiliki jumlah alamat IP yang cukup, cobalah untuk menambah jumlah alamat IP yang tersedia. Anda dapat melakukan ini dengan mengaitkan blok Classless Inter-Domain Routing () CIDR tambahan dengan blok Anda. VPC Untuk informasi selengkapnya, lihat Mengaitkan IPv4 CIDR blok tambahan dengan Anda VPC di Panduan VPC Pengguna Amazon.

Anda dapat menggunakan salah satu dari banyak alat yang tersedia secara online untuk menghasilkan definisi subnet dengan cepat dan meninjau berbagai alamat IP yang tersedia.

# Opsi EMR arsitektur Amazon Tanpa Server

Arsitektur set instruksi aplikasi Amazon EMR Serverless menentukan jenis prosesor yang digunakan aplikasi untuk menjalankan pekerjaan. Amazon EMR menyediakan dua opsi arsitektur untuk aplikasi Anda: x86_64 dan arm64. EMRTanpa server secara otomatis memperbarui instans generasi terbaru saat tersedia, sehingga aplikasi Anda dapat menggunakan instans yang lebih baru tanpa memerlukan upaya tambahan dari Anda.

Topik



## Menggunakan arsitektur x86_64

Arsitektur x86_64 juga dikenal sebagai x86 64-bit atau x64. x86_64 adalah opsi default untuk EMR aplikasi Tanpa Server. Arsitektur ini menggunakan prosesor berbasis x86 dan kompatibel dengan sebagian besar alat dan perpustakaan pihak ketiga.

Sebagian besar aplikasi kompatibel dengan platform perangkat keras x86 dan dapat berjalan dengan sukses pada arsitektur x86_64 default. Namun, jika aplikasi Anda kompatibel dengan 64-bitARM, maka Anda dapat beralih ke arm64 untuk menggunakan prosesor Graviton untuk meningkatkan kinerja, daya komputasi, dan memori. Biayanya lebih murah untuk menjalankan instance pada arsitektur arm64 daripada saat Anda menjalankan instance dengan ukuran yang sama pada arsitektur x86.

## Menggunakan arsitektur arm64 (Graviton)

AWS Prosesor Graviton dirancang khusus oleh AWS dengan inti ARM Neoverse 64-bit dan memanfaatkan arsitektur arm64 (juga dikenal sebagai Arch64 atau 64-bit). ARM Bagian AWS Jajaran prosesor Graviton yang tersedia di EMR Serverless termasuk prosesor Graviton3 dan Graviton2. Prosesor ini memberikan kinerja harga yang unggul untuk beban kerja Spark dan Hive dibandingkan dengan beban kerja setara yang berjalan pada arsitektur x86_64. EMRTanpa server secara otomatis menggunakan prosesor generasi terbaru bila tersedia tanpa upaya dari pihak Anda untuk meningkatkan ke prosesor generasi terbaru.

## Meluncurkan aplikasi baru dengan dukungan Graviton

Gunakan salah satu metode berikut untuk meluncurkan aplikasi yang menggunakan arsitektur arm64.

AWS CLI

Untuk meluncurkan aplikasi menggunakan prosesor Graviton dari AWS CLI, tentukan `ARM64` sebagai `architecture` parameter di `create-application`API. Berikan nilai yang sesuai untuk aplikasi Anda di parameter lain.

```
aws emr-serverless create-application \
 --name my-graviton-app \
 --release-label emr-6.8.0 \
 --type "SPARK" \
 --architecture "ARM64" \
 --region us-west-2
```

EMR Studio

Untuk meluncurkan aplikasi menggunakan prosesor Graviton dari EMR Studio, pilih arm64 sebagai opsi Arsitektur saat Anda membuat atau memperbarui aplikasi.

## Mengkonfigurasi aplikasi yang ada untuk menggunakan Graviton

Anda dapat mengonfigurasi aplikasi Amazon EMR Tanpa Server yang ada untuk menggunakan arsitektur Graviton (arm64) dengan, SDK AWS CLI, atau EMR Studio.

Untuk mengonversi aplikasi yang ada dari x86 ke arm64

1. Konfirmasikan bahwa Anda menggunakan versi utama terbaru dari AWS CLI/SDKyang mendukung `architecture` parameter.
2. Konfirmasikan bahwa tidak ada pekerjaan yang berjalan dan kemudian hentikan aplikasi.

   ```
   aws emr-serverless stop-application \
    --application-id application-id \
    --region us-west-2
   ```
3. Untuk memperbarui aplikasi untuk menggunakan Graviton, `ARM64` tentukan `architecture` parameter di. `update-application` API

```
aws emr-serverless update-application \
 --application-id application-id \
 --architecture 'ARM64' \
 --region us-west-2
```

4. Untuk memverifikasi bahwa CPU arsitektur aplikasi sekarangARM64, gunakan file `get-application`API.

```
aws emr-serverless get-application \
 --application-id application-id \
 --region us-west-2
```

5. Saat Anda siap, mulai ulang aplikasi.

```
aws emr-serverless start-application \
 --application-id application-id \
 --region us-west-2
```

## Pertimbangan saat menggunakan Graviton

Sebelum Anda meluncurkan aplikasi EMR Tanpa Server menggunakan arm64 untuk dukungan Graviton, konfirmasikan hal berikut.

### Kompatibilitas perpustakaan

Saat Anda memilih Graviton (arm64) sebagai opsi arsitektur, pastikan paket dan pustaka pihak ketiga kompatibel dengan arsitektur 64-bit. ARM Untuk informasi tentang cara mengemas pustaka Python ke dalam lingkungan virtual Python yang kompatibel dengan arsitektur yang Anda pilih, lihat. Menggunakan pustaka Python dengan Tanpa Server EMR

Untuk mempelajari selengkapnya tentang cara mengonfigurasi beban kerja Spark atau Hive agar menggunakan 64-bitARM, lihat AWS Graviton Memulai repositori aktif. GitHub Repositori ini berisi sumber daya penting yang dapat membantu Anda memulai Graviton ARM berbasis.

# Dapatkan data ke S3 Express One Zone dengan Tanpa Server EMR

Dengan Amazon EMR merilis 7.2.0 dan yang lebih tinggi, Anda dapat menggunakan EMR Tanpa Server dengan kelas penyimpanan Amazon S3 Express One Zone untuk meningkatkan kinerja saat menjalankan pekerjaan dan beban kerja. S3 Express One Zone adalah kelas penyimpanan Amazon S3 zona tunggal berkinerja tinggi yang memberikan akses data milidetik satu digit yang konsisten untuk sebagian besar aplikasi yang sensitif terhadap latensi. Pada saat rilis, S3 Express One Zone memberikan latensi terendah dan penyimpanan objek cloud kinerja tertinggi di Amazon S3.

## Prasyarat

- Izin S3 Express One Zone — Ketika S3 Express One Zone awalnya melakukan tindakan seperti`GET`,`LIST`, atau PUT pada objek S3, kelas penyimpanan memanggil `CreateSession` atas nama Anda. IAMKebijakan Anda harus mengizinkan `s3express:CreateSession` izin sehingga S3A konektor dapat memanggil. `CreateSession` API Untuk contoh kebijakan dengan izin ini, lihatMemulai dengan S3 Express One Zone.
- S3A konektor - Untuk mengonfigurasi Spark untuk mengakses data dari bucket Amazon S3 yang menggunakan kelas penyimpanan S3 Express One Zone, Anda harus menggunakan konektor Apache Hadoop S3A. Untuk menggunakan konektor, pastikan semua S3 URIs menggunakan `s3a` skema. Jika tidak, Anda dapat mengubah implementasi sistem file yang Anda gunakan untuk `s3` dan skema. `s3n`

Untuk mengubah `s3` skema, tentukan konfigurasi cluster berikut:

```
[
  {
    "Classification": "core-site",
    "Properties": {
      "fs.s3.impl": "org.apache.hadoop.fs.s3a.S3AFileSystem",
      "fs.AbstractFileSystem.s3.impl": "org.apache.hadoop.fs.s3a.S3A"
    }
  }
]
```

Untuk mengubah `s3n` skema, tentukan konfigurasi cluster berikut:

```
[
  {
    "Classification": "core-site",
    "Properties": {
      "fs.s3n.impl": "org.apache.hadoop.fs.s3a.S3AFileSystem",
      "fs.AbstractFileSystem.s3n.impl": "org.apache.hadoop.fs.s3a.S3A"
    }
  }
]
```

# Memulai dengan S3 Express One Zone

Ikuti langkah-langkah ini untuk memulai dengan S3 Express One Zone.

1. Buat VPC titik akhir. Tambahkan titik akhir `com.amazonaws.us-west-2.s3express` ke titik VPC akhir.
2. Ikuti Memulai Amazon EMR Tanpa Server untuk membuat aplikasi dengan label EMR rilis Amazon 7.2.0 atau lebih tinggi.
3. Konfigurasikan aplikasi Anda untuk menggunakan VPC endpoint yang baru dibuat, grup subnet pribadi, dan grup keamanan.
4. Tambahkan `CreateSession` izin ke peran eksekusi pekerjaan Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Resource": "*",
            "Action": [
                "s3express:CreateSession"
            ]
        }
    ]
}
```

5. Jalankan pekerjaan Anda. Perhatikan bahwa Anda harus menggunakan `S3A` skema untuk mengakses bucket S3 Express One Zone.

```
aws emr-serverless start-job-run \
```

```
--application-id <application-id> \
--execution-role-arn <job-role-arn> \
--name <job-run-name> \
--job-driver '{
 "sparkSubmit": {


 "entryPoint": "s3a://<DOC-EXAMPLE-BUCKET>/scripts/wordcount.py",
 "entryPointArguments":["s3a://<DOC-EXAMPLE-BUCKET>/emr-serverless-spark/output"],
 "sparkSubmitParameters": "--conf spark.executor.cores=4
 --conf spark.executor.memory=8g --conf spark.driver.cores=4
 --conf spark.driver.memory=8g --conf spark.executor.instances=2
 --conf spark.hadoop.fs.s3a.change.detection.mode=none
 --conf spark.hadoop.fs.s3a.endpoint.region={<AWS_REGION>}
 --conf spark.hadoop.fs.s3a.select.enabled=false
 --conf spark.sql.sources.fastS3PartitionDiscovery.enabled=false
 }'
```

# Menjalankan pekerjaan

Setelah Anda menyediakan aplikasi Anda, Anda dapat mengirimkan pekerjaan ke aplikasi. Bagian ini mencakup cara menggunakan AWS CLI untuk menjalankan pekerjaan ini. Bagian ini juga mengidentifikasi nilai default untuk setiap jenis aplikasi yang tersedia di Tanpa EMR Server.

Topik



## Status tugas berjalan

Saat Anda mengirimkan pekerjaan ke antrean pekerjaan Amazon EMR Tanpa Server, pekerjaan berjalan memasuki status. `SUBMITTED` Suatu keadaan pekerjaan berlalu dari `SUBMITTED` melalui `RUNNING` sampai mencapai`FAILED`,`SUCCESS`, atau`CANCELLING`.

Tugas berjalan dapat memiliki status berikut:

| Status | Deskripsi |
| --- | --- |
| Dikirim | Status pekerjaan awal saat Anda mengirimkan pekerjaan dijalankan ke Tanpa EMR Server. Pekerjaan menunggu untuk dijadwalkan untuk aplikasi. EMRTanpa server mulai mempriori taskan dan menjadwalkan pekerjaan berjalan. |

| Status | Deskripsi |
| --- | --- |
| Tertunda | Penjadwal mengevaluasi pekerjaan yang dijalankan untuk memprioritaskan dan menjadwalkan proses untuk aplikasi. |
| Dijadwalkan | EMRTanpa server telah menjadwalkan pekerjaan yang dijalankan untuk aplikasi, dan mengalokasikan sumber daya untuk menjalank an pekerjaan. |
| Berlari | EMRTanpa server telah mengalokasikan sumber daya yang awalnya dibutuhkan pekerjaan, dan pekerjaan berjalan dalam aplikasi. Dalam aplikasi Spark, ini berarti bahwa proses driver Spark ada di status `running`. |
| Failed | EMRTanpa server gagal mengirimkan pekerjaan yang dijalankan ke aplikasi, atau tidak berhasil diselesaikan. Lihat `StateDeta ils` untuk informasi tambahan tentang kegagalan pekerjaan ini. |
| Sukses | Jalankan pekerjaan telah selesai dengan sukses. |
| Membatalkan | Pembatalan pekerjaan yang `CancelJob Run` API telah diminta, atau pekerjaan telah habis waktunya. EMRTanpa server mencoba membatalkan pekerjaan dalam aplikasi dan melepaskan sumber daya. |
| Dibatalkan | Pekerjaan dibatalkan dengan sukses, dan sumber daya yang digunakannya telah dirilis. |

# Menjalankan pekerjaan dari konsol EMR Studio

Anda dapat mengirimkan pekerjaan berjalan ke aplikasi EMR Tanpa Server dan melihat pekerjaan dari konsol EMR Studio. Untuk membuat atau menavigasi ke aplikasi EMR Tanpa Server di konsol EMR Studio, ikuti petunjuk di Memulai dari konsol.

## Mengirim tugas

Pada halaman Kirim pekerjaan, Anda dapat mengirimkan pekerjaan ke aplikasi EMR Tanpa Server sebagai berikut.

Spark

1. Di bidang Nama, masukkan nama untuk menjalankan pekerjaan Anda.
2. Di bidang peran Runtime, masukkan nama IAM peran yang dapat diasumsikan oleh aplikasi EMR Tanpa Server Anda untuk menjalankan pekerjaan. Untuk mempelajari lebih lanjut tentang peran runtime, lihatPeran runtime Job untuk Amazon Serverless EMR.
3. Di bidang Lokasi skrip, masukkan lokasi Amazon S3 untuk skrip atau JAR yang ingin Anda jalankan. Untuk pekerjaan Spark, skrip dapat berupa file Python `.py` () atau JAR file `.jar` ().
4. Jika lokasi skrip Anda adalah JAR file, masukkan nama kelas yang merupakan titik masuk untuk pekerjaan di bidang kelas Utama.
5. (Opsional) Masukkan nilai untuk bidang yang tersisa.

   - Argumen skrip — Masukkan argumen apa pun yang ingin Anda teruskan ke skrip utama JAR atau Python Anda. Kode Anda membaca parameter ini. Pisahkan setiap argumen dalam array dengan koma.
   - Properti percikan - Perluas bagian properti Spark dan masukkan parameter konfigurasi Spark apa pun di bidang ini.

     > **Note**
     >
     > Jika Anda menentukan driver Spark dan ukuran pelaksana, Anda harus memperhitungkan overhead memori. Tentukan nilai overhead memori di properti `spark.driver.memoryOverhead` dan`spark.executor.memoryOverhead`. Memory overhead memiliki nilai default 10% dari memori kontainer, dengan minimal 384 MB. Memori eksekutor dan overhead memori bersama-sama tidak

dapat melebihi memori pekerja. Misalnya, maksimum `spark.executor.memory` pada pekerja 30 GB harus 27 GB.

- Konfigurasi Job - Tentukan konfigurasi pekerjaan apa pun di bidang ini. Anda dapat menggunakan konfigurasi pekerjaan ini untuk mengganti konfigurasi default untuk aplikasi.
- Pengaturan tambahan — Aktif atau nonaktifkan AWS Glue Data Catalog sebagai metastore dan memodifikasi pengaturan log aplikasi. Untuk mempelajari lebih lanjut tentang konfigurasi metastore, lihat. Konfigurasi metastore Untuk mempelajari lebih lanjut tentang opsi pencatatan aplikasi, lihatMenyimpan log.
- Tag - Tetapkan tag kustom ke aplikasi.

6. Pilih Submit job (Kirim tugas).

Hive

1. Di bidang Nama, masukkan nama untuk menjalankan pekerjaan Anda.
2. Di bidang peran Runtime, masukkan nama IAM peran yang dapat diasumsikan oleh aplikasi EMR Tanpa Server Anda untuk menjalankan pekerjaan.
3. Di bidang Lokasi skrip, masukkan lokasi Amazon S3 untuk skrip atau JAR yang ingin Anda jalankan. Untuk pekerjaan Hive, skrip harus berupa file Hive (`.sql`).
4. (Opsional) Masukkan nilai untuk bidang yang tersisa.

   - Lokasi skrip inisialisasi - Masukkan lokasi skrip yang menginisialisasi tabel sebelum skrip Hive berjalan.
   - Properti sarang - Perluas bagian properti Hive dan masukkan parameter konfigurasi Hive apa pun di bidang ini.
   - Konfigurasi Job - Tentukan konfigurasi pekerjaan apa pun. Anda dapat menggunakan konfigurasi pekerjaan ini untuk mengganti konfigurasi default untuk aplikasi. Untuk pekerjaan Hive, `hive.exec.scratchdir` dan properti `hive.metastore.warehouse.dir` yang diperlukan dalam `hive-site` konfigurasi.

```
{
    "applicationConfiguration": [
        {
            "classification": "hive-site",
            "configurations": [],
            "properties": {
```

```
                "hive.exec.scratchdir": "s3://DOC-EXAMPLE_BUCKET/hive/
scratch",
                "hive.metastore.warehouse.dir": "s3://DOC-EXAMPLE_BUCKET/hive/
warehouse"
            }
        }
    ],
    "monitoringConfiguration": {}
}
```

- Pengaturan tambahan - Aktifkan atau nonaktifkan AWS Glue Data Catalog sebagai metastore dan memodifikasi pengaturan log aplikasi. Untuk mempelajari lebih lanjut tentang konfigurasi metastore, lihat. Konfigurasi metastore Untuk mempelajari lebih lanjut tentang opsi pencatatan aplikasi, lihatMenyimpan log.
- Tag - Tetapkan tag kustom apa pun ke aplikasi.

5. Pilih Submit job (Kirim tugas).

## Lihat pekerjaan berjalan

Dari tab Job runs pada halaman Detail aplikasi, Anda dapat melihat pekerjaan berjalan dan melakukan tindakan berikut untuk menjalankan pekerjaan.

Batalkan pekerjaan — Untuk membatalkan pekerjaan yang ada di `RUNNING` negara bagian, pilih opsi ini. Untuk mempelajari lebih lanjut tentang transisi job run, lihatStatus tugas berjalan.

Clone job — Untuk mengkloning pekerjaan sebelumnya dan mengirimkannya kembali, pilih opsi ini.

# Menjalankan pekerjaan dari AWS CLI

Anda dapat membuat, mendeskripsikan, dan menghapus pekerjaan individual di AWS CLI. Anda juga dapat membuat daftar semua pekerjaan Anda untuk melihatnya sekilas.

Untuk mengirimkan pekerjaan baru, gunakan`start-job-run`. Berikan ID aplikasi yang ingin Anda jalankan, bersama dengan properti khusus pekerjaan. Untuk contoh Spark, lihatLowongan kerja Spark. Untuk contoh Hive, lihatPekerjaan sarang. Perintah ini mengembalikan Anda`application-id`,ARN, dan baru`job-id`.

Setiap job run memiliki durasi timeout yang ditetapkan. Jika pekerjaan yang dijalankan melebihi durasi ini, EMR Tanpa Server akan membatalkannya secara otomatis. Batas waktu default adalah 12

jam. Saat memulai pekerjaan, Anda dapat mengonfigurasi pengaturan batas waktu ini ke nilai yang memenuhi persyaratan pekerjaan Anda. Konfigurasikan nilai dengan `executionTimeoutMinutes` properti.

```
aws emr-serverless start-job-run \
  --application-id application-id \
  --execution-role-arn job-role-arn \
  --execution-timeout-minutes 15 \
  --job-driver '{
    "hive": {
        "query": "s3://DOC-EXAMPLE-BUCKET/scripts/create_table.sql",
        "parameters": "--hiveconf hive.exec.scratchdir=s3://DOC-EXAMPLE-BUCKET/hive/
scratch --hiveconf hive.metastore.warehouse.dir=s3://DOC-EXAMPLE-BUCKET/hive/warehouse"
    }
   }' \
  --configuration-overrides '{
    "applicationConfiguration": [{
        "classification": "hive-site",
        "properties": {
            "hive.client.cores": "2",
            "hive.client.memory": "4GIB"
        }
    }]
}'
```

Untuk menggambarkan pekerjaan, gunakan`get-job-run`. Perintah ini mengembalikan konfigurasi khusus pekerjaan dan kapasitas yang ditetapkan untuk pekerjaan baru Anda.

```
aws emr-serverless get-job-run \
--job-run-id job-id \
--application-id application-id
```

Untuk daftar pekerjaan Anda, gunakan`list-job-runs`. Perintah ini mengembalikan serangkaian properti yang disingkat yang mencakup tipe pekerjaan, status, dan atribut tingkat tinggi lainnya. Jika Anda tidak ingin melihat semua pekerjaan Anda, Anda dapat menentukan jumlah maksimum pekerjaan yang ingin Anda lihat, hingga 50. Contoh berikut menentukan bahwa Anda ingin melihat dua pekerjaan terakhir Anda berjalan.

```
aws emr-serverless list-job-runs \
--max-results 2 \
--application-id application-id
```

Untuk membatalkan pekerjaan, gunakan`cancel-job-run`. Berikan `application-id` dan pekerjaan `job-id` yang ingin Anda batalkan.

```
aws emr-serverless cancel-job-run \
--job-run-id job-id \
--application-id application-id
```

Untuk informasi lebih lanjut tentang cara menjalankan pekerjaan dari AWS CLI, lihat EMRReferensi Tanpa Server API.

# Menggunakan disk yang dioptimalkan dengan shuffle

Dengan Amazon EMR merilis 7.1.0 dan yang lebih tinggi, Anda dapat menggunakan disk yang dioptimalkan secara acak saat menjalankan pekerjaan Apache Spark atau Hive untuk meningkatkan kinerja beban kerja intensif I/O. Dibandingkan dengan disk standar, disk yang dioptimalkan dengan shuffle memberikan lebih tinggi IOPS (operasi I/O per detik) untuk pergerakan data yang lebih cepat dan mengurangi latensi selama operasi shuffle. Disk yang dioptimalkan dengan shuffle memungkinkan Anda melampirkan ukuran disk hingga 2 TB per pekerja, sehingga Anda dapat mengonfigurasi kapasitas yang sesuai untuk kebutuhan beban kerja Anda.

## Manfaat utama

Disk yang dioptimalkan dengan shuffle memberikan manfaat berikut.

- IOPSKinerja tinggi - disk yang dioptimalkan dengan shuffle memberikan disk yang IOPS lebih tinggi dari disk standar, yang mengarah ke pengocokan data yang lebih efisien dan cepat selama pekerjaan Spark and Hive dan beban kerja intensif acak lainnya.
- Ukuran disk yang lebih besar - Disk yang dioptimalkan dengan shuffle mendukung ukuran disk dari 20GB hingga 2TB per pekerja, sehingga Anda dapat memilih kapasitas yang sesuai berdasarkan beban kerja Anda.

## Memulai

Lihat langkah-langkah berikut untuk menggunakan disk yang dioptimalkan secara acak di alur kerja Anda.

Spark

1. Buat aplikasi rilis 7.1.0 EMR Tanpa Server dengan perintah berikut.

```
aws emr-serverless create-application \
  --type "SPARK" \
  --name my-application-name \
  --release-label emr-7.1.0 \
  --region <AWS_REGION>
```

2. Konfigurasikan pekerjaan Spark Anda untuk menyertakan parameter `spark.emr-serverless.driver.disk.type` dan/atau `spark.emr-serverless.executor.disk.type` untuk dijalankan dengan disk yang dioptimalkan secara acak. Anda dapat menggunakan salah satu atau kedua parameter, tergantung pada kasus penggunaan Anda.

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "sparkSubmit": {
            "entryPoint": "/usr/lib/spark/examples/jars/spark-examples.jar",
            "entryPointArguments": ["1"],
            "sparkSubmitParameters": "--class org.apache.spark.examples.SparkPi
            --conf spark.executor.cores=4
            --conf spark.executor.memory=20g
            --conf spark.driver.cores=4
            --conf spark.driver.memory=8g
            --conf spark.executor.instances=1
            --conf spark.emr-serverless.executor.disk.type=shuffle_optimized"
        }
    }'
```

Untuk informasi selengkapnya, lihat Properti pekerjaan Spark.

Hive

1. Buat aplikasi rilis 7.1.0 EMR Tanpa Server dengan perintah berikut.

```
aws emr-serverless create-application \
  --type "HIVE" \
```

```
  --name my-application-name \
  --release-label emr-7.1.0 \
  --region <AWS_REGION>
```

2. Konfigurasikan pekerjaan Hive Anda untuk menyertakan parameter `hive.driver.disk.type` dan/atau `hive.tez.disk.type` untuk dijalankan dengan disk yang dioptimalkan acak. Anda dapat menggunakan salah satu atau kedua parameter, tergantung pada kasus penggunaan Anda.

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "hive": {
            "query": "s3://<DOC-EXAMPLE-BUCKET>/emr-serverless-hive/query/hive-
query.ql",
            "parameters": "--hiveconf hive.log.explain.output=false"
        }
    }' \
    --configuration-overrides '{
        "applicationConfiguration": [{
            "classification": "hive-site",
            "properties": {
                "hive.exec.scratchdir": "s3://<DOC-EXAMPLE-BUCKET>/emr-
serverless-hive/hive/scratch",
                "hive.metastore.warehouse.dir": "s3://<DOC-EXAMPLE-BUCKET>/emr-
serverless-hive/hive/warehouse",
                "hive.driver.cores": "2",
                "hive.driver.memory": "4g",
                "hive.tez.container.size": "4096",
                "hive.tez.cpu.vcores": "1",
                "hive.driver.disk.type": "shuffle_optimized",
                "hive.tez.disk.type": "shuffle_optimized"
            }
        }]
    }'
```

Untuk informasi lebih lanjut, properti pekerjaan Hive.

Mengkonfigurasi aplikasi dengan kapasitas pra-inisialisasi

Lihat contoh berikut untuk membuat aplikasi berdasarkan Amazon EMR rilis 7.1.0. Aplikasi ini memiliki properti berikut:

- 5 driver Spark pra-inisialisasi, masing-masing dengan 2 v, 4 GB memoriCPU, dan 50 GB disk yang dioptimalkan shuffle.
- 50 eksekutor pra-inisialisasi, masing-masing dengan 4 v, 8 GB memoriCPU, dan 500 GB disk yang dioptimalkan shuffle.

Ketika aplikasi ini menjalankan pekerjaan Spark, pertama-tama ia mengkonsumsi pekerja pra-inisialisasi dan kemudian menskalakan pekerja sesuai permintaan hingga kapasitas maksimum 400 v CPU dan 1024 GB memori. Secara opsional, Anda dapat menghilangkan kapasitas untuk salah satu atau`DRIVER`. `EXECUTOR`

Spark

```
aws emr-serverless create-application \
  --type "SPARK" \
  --name <my-application-name> \
  --release-label emr-7.1.0 \
  --initial-capacity '{
    "DRIVER": {
        "workerCount": 5,
        "workerConfiguration": {
            "cpu": "2vCPU",
            "memory": "4GB",
            "disk": "50GB",
            "diskType": "SHUFFLE_OPTIMIZED"
        }
    },
    "EXECUTOR": {
        "workerCount": 50,
        "workerConfiguration": {
            "cpu": "4vCPU",
            "memory": "8GB",
            "disk": "500GB",
            "diskType": "SHUFFLE_OPTIMIZED"
        }
    }
  }' \
  --maximum-capacity '{
    "cpu": "400vCPU",
```

```
    "memory": "1024GB"
  }'
```

Hive

```
aws emr-serverless create-application \
  --type "HIVE" \
  --name <my-application-name> \
  --release-label emr-7.1.0 \
  --initial-capacity '{
    "DRIVER": {
        "workerCount": 5,
        "workerConfiguration": {
            "cpu": "2vCPU",
            "memory": "4GB",
            "disk": "50GB",
            "diskType": "SHUFFLE_OPTIMIZED"
        }
    },
    "EXECUTOR": {
        "workerCount": 50,
        "workerConfiguration": {
            "cpu": "4vCPU",
            "memory": "8GB",
            "disk": "500GB",
            "diskType": "SHUFFLE_OPTIMIZED"
        }
    }
  }' \
  --maximum-capacity '{
    "cpu": "400vCPU",
    "memory": "1024GB"
  }'
```

# Lowongan kerja Streaming

Pekerjaan streaming di EMR Tanpa Server adalah mode pekerjaan yang memungkinkan Anda menganalisis dan memproses data streaming dalam waktu dekat. Pekerjaan yang sudah berjalan lama ini melakukan polling data streaming dan terus memproses hasil saat data tiba. Pekerjaan streaming paling cocok untuk tugas yang memerlukan pemrosesan data waktu nyata, seperti analitik dekat waktu nyata, deteksi penipuan, dan mesin rekomendasi. EMRPekerjaan streaming tanpa

server memberikan pengoptimalan, seperti ketahanan kerja bawaan, pemantauan waktu nyata, manajemen log yang ditingkatkan, dan integrasi dengan konektor streaming.

Berikut ini adalah beberapa kasus penggunaan dengan pekerjaan streaming:

- Near real-time analytics — pekerjaan streaming di Amazon EMR Serverless memungkinkan Anda memproses data streaming dalam waktu dekat, sehingga Anda dapat melakukan analisis real-time pada aliran data berkelanjutan, seperti data log, data sensor, atau data clickstream untuk memperoleh wawasan dan membuat keputusan tepat waktu berdasarkan informasi terbaru.
- Deteksi penipuan — Anda dapat menggunakan pekerjaan streaming untuk menjalankan deteksi penipuan real-time dalam transaksi keuangan, operasi kartu kredit, atau aktivitas online saat Anda menganalisis aliran data dan mengidentifikasi pola atau anomali yang mencurigakan saat terjadi.
- Mesin rekomendasi — pekerjaan streaming dapat memproses data aktivitas pengguna dan memperbarui model rekomendasi. Melakukan hal itu membuka kemungkinan untuk rekomendasi yang dipersonalisasi dan real-time berdasarkan perilaku dan preferensi.
- Analisis media sosial — pekerjaan streaming dapat memproses data media sosial, seperti tweet, komentar, dan posting, sehingga organisasi dapat memantau tren, analisis sentimen, dan mengelola reputasi merek dalam waktu dekat.
- Analisis Internet of Things (IoT) — pekerjaan streaming dapat menangani dan menganalisis aliran data berkecepatan tinggi dari perangkat IoT, sensor, dan mesin yang terhubung, sehingga Anda dapat menjalankan deteksi anomali, pemeliharaan prediktif, dan kasus penggunaan analitik IoT lainnya.
- Analisis Clickstream — pekerjaan streaming dapat memproses dan menganalisis data clickstream dari situs web atau aplikasi seluler. Bisnis yang menggunakan data tersebut dapat menjalankan analitik untuk mempelajari lebih lanjut tentang perilaku pengguna, mempersonalisasi pengalaman pengguna, dan mengoptimalkan kampanye pemasaran.
- Pemantauan dan analisis log — pekerjaan streaming juga dapat memproses data log dari server, aplikasi, dan perangkat jaringan. Ini memberi Anda deteksi anomali, pemecahan masalah, serta kesehatan dan kinerja sistem.

### Manfaat utama

Pekerjaan streaming di EMR Tanpa Server secara otomatis memberikan ketahanan kerja, yang merupakan kombinasi dari faktor-faktor berikut:

- Coba ulang otomatis - EMR Tanpa server secara otomatis mencoba ulang pekerjaan apa pun yang gagal tanpa masukan manual dari Anda.

- Ketahanan Availability Zone (AZ) — EMR Tanpa server secara otomatis mengalihkan pekerjaan streaming ke AZ yang sehat jika AZ asli mengalami masalah.
- Manajemen log:
  - Rotasi log — untuk manajemen penyimpanan disk yang lebih efisien, EMR Serverless secara berkala memutar log untuk pekerjaan streaming yang lama. Melakukannya mencegah akumulasi log yang mungkin menghabiskan semua ruang disk.
  - Pemadatan log - membantu Anda mengelola dan mengoptimalkan file log secara efisien dalam kegigihan terkelola. Pemadatan juga meningkatkan pengalaman debug saat Anda menggunakan server riwayat percikan terkelola.

Sumber data dan sink data yang didukung

EMRServerless bekerja dengan sejumlah sumber data input dan sink data output:

- Sumber data input yang didukung - Amazon Kinesis Data Streams, Amazon Managed Streaming untuk Apache Kafka Kafka, dan cluster Apache Kafka yang dikelola sendiri. Secara default, Amazon EMR merilis 7.1.0 dan yang lebih tinggi menyertakan konektor Amazon Kinesis Data Streams, jadi Anda tidak perlu membuat atau mengunduh paket tambahan apa pun.
- Sinks data keluaran yang didukung - AWS Tabel Katalog Data Glue, Amazon S3, Amazon Redshift, SQL My, SQL Postgre Oracle, Oracle, SQL Microsoft, Apache Iceberg, Delta Lake, dan Apache Hudi.

## Pertimbangan dan batasan

Saat Anda menggunakan pekerjaan streaming, ingatlah pertimbangan dan batasan berikut.

- Pekerjaan streaming didukung dengan EMRrilis Amazon 7.1.0 dan yang lebih tinggi.
- EMRTanpa server mengharapkan pekerjaan streaming berjalan untuk waktu yang lama, sehingga Anda tidak dapat mengatur batas waktu eksekusi untuk membatasi runtime pekerjaan.
- Pekerjaan streaming hanya kompatibel dengan mesin Spark, yang dibangun di atas kerangka kerja streaming terstruktur.
- EMRTanpa server mencoba ulang pekerjaan streaming tanpa batas waktu, dan Anda tidak dapat menyesuaikan jumlah upaya maksimum. Pencegahan thrash secara otomatis disertakan untuk menghentikan percobaan ulang pekerjaan jika jumlah upaya yang gagal telah melampaui ambang batas yang ditetapkan selama jendela per jam. Ambang batas default adalah lima upaya gagal

selama satu jam. Anda dapat mengonfigurasi ambang batas ini menjadi antara 1 dan 10 upaya. Untuk informasi lebih lanjut, lihat Ketahanan Job.

- Pekerjaan streaming memiliki pos pemeriksaan untuk menghemat status dan kemajuan runtime, sehingga EMR Tanpa Server dapat melanjutkan pekerjaan streaming dari pos pemeriksaan terbaru. Untuk informasi selengkapnya, lihat Memulihkan dari kegagalan dengan Checkpointing di dokumentasi Apache Spark.

## Memulai pekerjaan streaming

Lihat petunjuk berikut untuk mempelajari cara memulai pekerjaan streaming.

1. Ikuti Memulai Amazon EMR Tanpa Server untuk membuat aplikasi. Perhatikan bahwa aplikasi Anda harus menjalankan Amazon EMR release 7.1.0 atau yang lebih tinggi.
2. Setelah aplikasi Anda siap, atur `mode` parameter `STREAMING` untuk mengirimkan pekerjaan streaming, mirip dengan yang berikut ini AWS CLI contoh.

```
aws emr-serverless start-job-run \
--application-id <APPPLICATION_ID> \
--execution-role-arn <JOB_EXECUTION_ROLE> \
--mode 'STREAMING' \
--job-driver '{
    "sparkSubmit": {
        "entryPoint": "s3://<streaming script>",
        "entryPointArguments": ["s3://<DOC-EXAMPLE-BUCKET-OUTPUT>/output"],
        "sparkSubmitParameters": "--conf spark.executor.cores=4
            --conf spark.executor.memory=16g
            --conf spark.driver.cores=4
            --conf spark.driver.memory=16g
            --conf spark.executor.instances=3"
    }
}'
```

## Konektor streaming yang didukung

Konektor streaming memfasilitasi membaca data dari sumber streaming dan juga dapat menulis data ke wastafel streaming.

Berikut ini adalah konektor streaming yang didukung:

### Konektor Amazon Kinesis Data Streams

Konektor Amazon Kinesis Data Streams untuk Apache Spark memungkinkan pembuatan aplikasi streaming dan pipeline yang menggunakan data dari dan menulis data ke Amazon Kinesis Data Streams. Konektor mendukung peningkatan konsumsi kipas dengan tingkat throughput baca khusus hingga 2MB/detik per pecahan. Secara default, Amazon EMR Serverless 7.1.0 dan yang lebih tinggi menyertakan konektor, jadi Anda tidak perlu membuat atau mengunduh paket tambahan apa pun. Untuk informasi lebih lanjut tentang konektor, lihat spark-sql-kinesis-connector halaman di GitHub.

Berikut ini adalah contoh bagaimana memulai pekerjaan yang dijalankan dengan ketergantungan konektor Kinesis Data Streams.

```
aws emr-serverless start-job-run \
--application-id <APPLICATION_ID> \
--execution-role-arn <JOB_EXECUTION_ROLE> \
--mode 'STREAMING' \
--job-driver '{
    "sparkSubmit": {
        "entryPoint": "s3://<Kinesis-streaming-script>",
        "entryPointArguments": ["s3://<DOC-EXAMPLE-BUCKET-OUTPUT>/output"],
        "sparkSubmitParameters": "--conf spark.executor.cores=4
                --conf spark.executor.memory=16g
                --conf spark.driver.cores=4
                --conf spark.driver.memory=16g
                --conf spark.executor.instances=3
                --jars /usr/share/aws/kinesis/spark-sql-kinesis/lib/spark-streaming-
sql-kinesis-connector.jar"
    }
}'
```

Untuk terhubung ke Kinesis Data Streams, Anda harus EMR mengonfigurasi VPC aplikasi Tanpa Server dengan akses dan VPC menggunakan titik akhir untuk memungkinkan akses pribadi. atau NAT gunakan Gateway untuk mendapatkan akses publik. Untuk informasi selengkapnya, lihat Mengonfigurasi VPC akses. Anda juga harus memastikan bahwa peran runtime pekerjaan Anda memiliki izin baca dan tulis yang diperlukan untuk mengakses aliran data yang diperlukan. Untuk mempelajari lebih lanjut tentang cara mengonfigurasi peran runtime pekerjaan, lihat Peran runtime Job untuk Amazon EMR Tanpa Server. Untuk daftar lengkap semua izin yang diperlukan, lihat spark-sql-kinesis-connector halaman di GitHub.

### Konektor Apache Kafka

Konektor Apache Kafka untuk streaming terstruktur Spark adalah konektor open-source dari komunitas Spark dan tersedia di repositori Maven. Konektor ini memfasilitasi aplikasi streaming terstruktur Spark untuk membaca data dari dan menulis data ke Apache Kafka yang dikelola sendiri dan Amazon Managed Streaming for Apache Kafka. Untuk informasi selengkapnya tentang konektor, lihat Panduan Integrasi Streaming Terstruktur+Kafka di dokumentasi Apache Spark.

Contoh berikut menunjukkan cara memasukkan konektor Kafka dalam permintaan menjalankan pekerjaan Anda.

```
aws emr-serverless start-job-run \
--application-id <APPLICATION_ID> \
--execution-role-arn <JOB_EXECUTION_ROLE> \
--mode 'STREAMING' \
--job-driver '{
    "sparkSubmit": {
        "entryPoint": "s3://<Kafka-streaming-script>",
        "entryPointArguments": ["s3://<DOC-EXAMPLE-BUCKET-OUTPUT>/output"],
        "sparkSubmitParameters": "--conf spark.executor.cores=4
                --conf spark.executor.memory=16g
                --conf spark.driver.cores=4
                --conf spark.driver.memory=16g
                --conf spark.executor.instances=3
                --packages org.apache.spark:spark-sql-
kafka-0-10_2.12:<KAFKA_CONNECTOR_VERSION>"
    }
}'
```

Versi konektor Apache Kafka bergantung pada versi rilis EMR Tanpa Server Anda dan versi Spark yang sesuai. Untuk menemukan versi Kafka yang benar, lihat lihat Panduan Streaming Terstruktur +Integrasi Kafka.

Untuk menggunakan Amazon Managed Streaming for Apache Kafka Kafka IAM dengan autentikasi, Anda harus menyertakan dependensi lain untuk mengaktifkan konektor Kafka untuk terhubung ke Amazon. MSK IAM Untuk informasi selengkapnya, lihat aws-msk-iam-auth repositori di. GitHub Anda juga harus memastikan bahwa peran runtime pekerjaan memiliki IAM izin yang diperlukan. Contoh berikut menunjukkan bagaimana menggunakan konektor dengan IAM otentikasi.

```
aws emr-serverless start-job-run \
--application-id <APPLICATION_ID> \
--execution-role-arn <JOB_EXECUTION_ROLE> \
--mode 'STREAMING' \
```

```
--job-driver '{
    "sparkSubmit": {
        "entryPoint": "s3://<Kafka-streaming-script>",
        "entryPointArguments": ["s3://<DOC-EXAMPLE-BUCKET-OUTPUT>/output"],
        "sparkSubmitParameters": "--conf spark.executor.cores=4
                --conf spark.executor.memory=16g
                --conf spark.driver.cores=4
                --conf spark.driver.memory=16g
                --conf spark.executor.instances=3
                --packages org.apache.spark:spark-sql-
kafka-0-10_2.12:<KAFKA_CONNECTOR_VERSION>,software.amazon.msk:aws-msk-iam-
auth:<MSK_IAM_LIB_VERSION>"
    }
}'
```

Untuk menggunakan konektor Kafka dan pustaka IAM otentikasi dari Amazon, MSK Anda harus mengonfigurasi aplikasi EMR Tanpa Server dengan akses. VPC Subnet Anda harus memiliki akses Internet dan menggunakan NAT Gateway untuk mengakses dependensi Maven. Untuk informasi selengkapnya, lihat Mengonfigurasi VPC akses. Subnet harus memiliki konektivitas jaringan untuk mengakses cluster Kafka. Ini benar terlepas dari apakah cluster Kafka Anda dikelola sendiri atau jika Anda menggunakan Amazon Managed Streaming for Apache Kafka.

## Manajemen log pekerjaan streaming

### Rotasi log

Pekerjaan streaming mendukung rotasi log untuk log aplikasi Spark dan log peristiwa. Rotasi log mencegah pekerjaan streaming panjang menghasilkan file log besar yang mungkin menghabiskan semua ruang disk Anda yang tersedia. Rotasi log membantu Anda menghemat penyimpanan disk dan mencegah kegagalan pekerjaan karena ruang disk yang rendah. Untuk informasi selengkapnya, lihat Memutar log.

### Pemadatan log

Pekerjaan streaming juga mendukung pemadatan log untuk log peristiwa Spark setiap kali logging terkelola tersedia. Untuk detail selengkapnya tentang logging terkelola, lihat Logging dengan penyimpanan terkelola. Pekerjaan streaming dapat berjalan untuk waktu yang lama, dan jumlah data acara dapat bertambah dari waktu ke waktu dan secara signifikan meningkatkan ukuran file log. Spark History Server membaca dan memuat peristiwa ini ke dalam memori untuk UI aplikasi Spark. Proses ini dapat menyebabkan latensi dan biaya tinggi, terutama jika log peristiwa yang disimpan di Amazon S3 sangat besar.

Pemadatan log mengurangi ukuran log peristiwa, sehingga Server Riwayat Spark tidak perlu memuat lebih dari 1 GB log peristiwa kapan saja. Untuk informasi selengkapnya, lihat Pemantauan dan Instrumentasi dalam dokumentasi Apache Spark.

# Lowongan kerja Spark

Anda dapat menjalankan pekerjaan Spark pada aplikasi dengan `type` parameter yang disetel ke`SPARK`. Pekerjaan harus kompatibel dengan versi Spark yang kompatibel dengan versi EMR rilis Amazon. Misalnya, ketika Anda menjalankan pekerjaan dengan Amazon EMR rilis 6.6.0, pekerjaan Anda harus kompatibel dengan Apache Spark 3.2.0. Untuk informasi tentang versi aplikasi untuk setiap rilis, lihatEMRVersi rilis Amazon Tanpa Server.

## Parameter pekerjaan percikan

Bila Anda menggunakan `StartJobRun`APIuntuk menjalankan pekerjaan Spark, Anda dapat menentukan parameter berikut.

Parameter yang diperlukan

- Peran runtime pekerjaan percikan
- Parameter pengemudi pekerjaan percikan
- Parameter penggantian konfigurasi percikan
- Spark optimasi alokasi sumber daya dinamis

### Peran runtime pekerjaan percikan

Gunakan **`executionRoleArn`**ARNuntuk menentukan IAM peran yang digunakan aplikasi Anda untuk menjalankan pekerjaan Spark. Peran ini harus berisi izin berikut:

- Baca dari bucket S3 atau sumber data lain di mana data Anda berada
- Baca dari bucket atau awalan S3 tempat skrip atau file Anda PySpark berada JAR
- Menulis ke ember S3 di mana Anda ingin menulis hasil akhir Anda
- Menulis log ke bucket S3 atau awalan yang menentukan `S3MonitoringConfiguration`
- Akses ke KMS kunci jika Anda menggunakan KMS kunci untuk mengenkripsi data di bucket S3
- Akses ke AWS Glue Data Catalog jika Anda menggunakan Spark SQL

Jika pekerjaan Spark Anda membaca atau menulis data ke atau dari sumber data lain, tentukan izin yang sesuai dalam peran iniIAM. Jika Anda tidak memberikan izin ini untuk IAM peran tersebut, pekerjaan mungkin gagal. Untuk informasi selengkapnya, silakan lihat Peran runtime Job untuk Amazon Serverless EMR dan Menyimpan log.

### Parameter pengemudi pekerjaan percikan

Gunakan **`jobDriver`**untuk memberikan masukan pada pekerjaan. Parameter driver pekerjaan hanya menerima satu nilai untuk jenis pekerjaan yang ingin Anda jalankan. Untuk pekerjaan Spark, nilai parameternya adalah`sparkSubmit`. Anda dapat menggunakan jenis pekerjaan ini untuk menjalankan Scala, Java, PySpark, SparkR, dan pekerjaan lain yang didukung melalui pengiriman Spark. Pekerjaan percikan memiliki parameter berikut:

- **`sparkSubmitParameters`**— Ini adalah parameter Spark tambahan yang ingin Anda kirim ke pekerjaan. Gunakan parameter ini untuk mengganti properti Spark default seperti memori driver atau jumlah pelaksana, seperti yang didefinisikan dalam argumen atau. `--conf` `--class`
- **`entryPointArguments`**— Ini adalah array argumen yang ingin Anda sampaikan ke file utama JAR atau Python Anda. Anda harus menangani membaca parameter ini menggunakan kode entrypoint Anda. Pisahkan setiap argumen dalam array dengan koma.
- **`entryPoint`**— Ini adalah referensi di Amazon S3 ke file utama atau JAR Python yang ingin Anda jalankan. Jika Anda menjalankan Scala atau JavaJAR, tentukan kelas entri utama dalam `SparkSubmitParameters` menggunakan `--class` argumen.

Untuk informasi tambahan, lihat Peluncuran Aplikasi dengan spark-submit.

### Parameter penggantian konfigurasi percikan

Gunakan **`configurationOverrides`**untuk mengganti properti konfigurasi tingkat pemantauan dan tingkat aplikasi. Parameter ini menerima JSON objek dengan dua bidang berikut:

- **`monitoringConfiguration`**- Gunakan bidang ini untuk menentukan Amazon S3 URL (`s3MonitoringConfiguration`) tempat Anda ingin pekerjaan EMR Tanpa Server menyimpan log pekerjaan Spark Anda. Pastikan Anda telah membuat bucket ini dengan hal yang sama Akun AWS yang meng-host aplikasi Anda, dan dalam hal yang sama Wilayah AWS Di mana pekerjaan Anda berjalan.
- **`applicationConfiguration`**— Untuk mengganti konfigurasi default untuk aplikasi, Anda dapat memberikan objek konfigurasi di bidang ini. Anda dapat menggunakan sintaks singkatan untuk

menyediakan konfigurasi, atau Anda dapat mereferensikan objek konfigurasi dalam file. JSON Objek konfigurasi terdiri dari klasifikasi, properti, dan konfigurasi bersarang opsional. Properti terdiri dari pengaturan yang ingin Anda timpa dalam file itu. Anda dapat menentukan beberapa klasifikasi untuk beberapa aplikasi dalam satu JSON objek.

> **Note**
>
> Klasifikasi konfigurasi yang tersedia bervariasi menurut rilis Tanpa EMR Server tertentu. Misalnya, klasifikasi untuk Log4j kustom `spark-driver-log4j2` dan hanya `spark-executor-log4j2` tersedia dengan rilis 6.8.0 dan yang lebih tinggi.

Jika Anda menggunakan konfigurasi yang sama dalam penggantian aplikasi dan dalam parameter pengiriman Spark, parameter pengiriman Spark akan diprioritaskan. Konfigurasi peringkat dalam prioritas sebagai berikut, dari tertinggi ke terendah:

- Konfigurasi yang disediakan EMR Tanpa Server saat dibuat. `SparkSession`
- Konfigurasi yang Anda berikan sebagai bagian dari `sparkSubmitParameters` `--conf` argumen.
- Konfigurasi yang Anda berikan sebagai bagian dari penggantian aplikasi Anda ketika Anda memulai pekerjaan.
- Konfigurasi yang Anda berikan sebagai bagian dari `runtimeConfiguration` saat Anda membuat aplikasi.
- Konfigurasi yang dioptimalkan yang EMR digunakan Amazon untuk rilis.
- Konfigurasi sumber terbuka default untuk aplikasi.

Untuk informasi selengkapnya tentang mendeklarasikan konfigurasi di tingkat aplikasi, dan mengganti konfigurasi selama menjalankan pekerjaan, lihat. Konfigurasi aplikasi default untuk Tanpa EMR Server

## Spark optimasi alokasi sumber daya dinamis

Gunakan `dynamicAllocationOptimization` untuk mengoptimalkan penggunaan sumber daya di EMR Tanpa Server. Menyetel properti ini ke `true` dalam klasifikasi konfigurasi Spark Anda menunjukkan ke EMR Tanpa Server untuk mengoptimalkan alokasi sumber daya pelaksana untuk menyelaraskan tingkat permintaan dan pembatalan Spark dengan tingkat di mana Serverless membuat dan melepaskan pekerja. EMR Dengan demikian, EMR Serverless menggunakan kembali

pekerja secara lebih optimal di seluruh tahapan, menghasilkan biaya yang lebih rendah saat menjalankan pekerjaan dengan beberapa tahap sambil mempertahankan kinerja yang sama.

Properti ini tersedia di semua versi EMR rilis Amazon.

Berikut ini adalah klasifikasi konfigurasi sampel dengan`dynamicAllocationOptimization`.

```
[
  {
    "Classification": "spark",
    "Properties": {
      "dynamicAllocationOptimization": "true"
    }
  }
]
```

Pertimbangkan hal berikut jika Anda menggunakan optimasi alokasi dinamis:

- Pengoptimalan ini tersedia untuk pekerjaan Spark yang Anda aktifkan alokasi sumber daya dinamis.
- Untuk mencapai efisiensi biaya terbaik, kami sarankan untuk mengonfigurasi batas skala atas pada pekerja menggunakan pengaturan tingkat pekerjaan `spark.dynamicAllocation.maxExecutors` atau pengaturan kapasitas maksimum tingkat aplikasi berdasarkan beban kerja Anda.
- Anda mungkin tidak melihat peningkatan biaya dalam pekerjaan yang lebih sederhana. Misalnya, jika pekerjaan Anda berjalan pada kumpulan data kecil atau selesai berjalan dalam satu tahap, Spark mungkin tidak memerlukan jumlah pelaksana yang lebih besar atau beberapa peristiwa penskalaan.
- Pekerjaan dengan urutan tahap besar, tahapan yang lebih kecil, dan kemudian tahap besar lagi mungkin mengalami regresi dalam runtime pekerjaan. Karena EMR Tanpa Server menggunakan sumber daya secara lebih efisien, ini mungkin menyebabkan lebih sedikit pekerja yang tersedia untuk tahap yang lebih besar, yang mengarah ke runtime yang lebih lama.

## Properti pekerjaan percikan

Tabel berikut mencantumkan properti Spark opsional dan nilai defaultnya yang dapat Anda ganti saat mengirimkan pekerjaan Spark.

| Kunci | Deskripsi | Nilai default |
| --- | --- | --- |
| `spark.archives` | Daftar arsip yang dipisahkan koma yang diekstrak Spark ke dalam direktori kerja masing-masing pelaksana. Jenis file yang didukung termasuk `.jar`, `.tar.gz`, `.tgz` dan `.zip`. Untuk menentukan nama direktori yang akan diekstrak , tambahkan # setelah nama file yang ingin Anda ekstrak. Misalnya, `file.zip#directory` . | `NULL` |
| `spark.authenticate` | Opsi yang mengaktifkan otentikasi koneksi internal Spark. | `TRUE` |
| `spark.driver.cores` | Jumlah core yang digunakan driver. | 4 |
| `spark.driver.extraJavaOptions` | Opsi Java ekstra untuk driver Spark. | `NULL` |
| `spark.driver.memory` | Jumlah memori yang digunakan pengemudi. | 14G |
| `spark.dynamicAllocation.enabled` | Opsi yang mengaktifkan alokasi sumber daya dinamis. Opsi ini menaikkan atau menurunkan jumlah pelaksana yang terdaftar dengan aplikasi, berdasarkan beban kerja. | `TRUE` |
| `spark.dynamicAllocation.executorIdleTimeout` | Lamanya waktu seorang eksekutor dapat tetap menganggur sebelum Spark | 60-an |

| Kunci | Deskripsi | Nilai default |
| --- | --- | --- |
| | menghapusnya. Ini hanya berlaku jika Anda mengaktif kan alokasi dinamis. | |
| `spark.dynamicAlloc ation.initialExecu tors` | Jumlah awal pelaksana untuk dijalankan jika Anda mengaktif kan alokasi dinamis. | `3` |
| `spark.dynamicAlloc ation.maxExecutors` | Batas atas untuk jumlah pelaksana jika Anda mengaktif kan alokasi dinamis. | Untuk 6.10.0 dan lebih tinggi, `infinity`<br><br>Untuk 6.9.0 dan lebih rendah, `100` |
| `spark.dynamicAlloc ation.minExecutors` | Batas bawah untuk jumlah pelaksana jika Anda mengaktif kan alokasi dinamis. | `0` |
| `spark.emr-serverle ss.allocation.batc h.size` | Jumlah kontainer untuk meminta dalam setiap siklus alokasi pelaksana. Ada kesenjangan satu detik antara setiap siklus alokasi. | 20 |
| `spark.emr-serverle ss.driver.disk` | Disk driver Spark. | 20G |
| `spark.emr-serverle ss.driverEnv.` *`[KEY]`* | Opsi yang menambahkan variabel lingkungan ke driver Spark. | `NULL` |
| `spark.emr-serverle ss.executor.disk` | Disk eksekutor Spark. | 20G |

| Kunci | Deskripsi | Nilai default |
| --- | --- | --- |
| `spark.emr-serverless.memoryOverheadFactor` | Mengatur overhead memori untuk ditambahkan ke driver dan memori kontainer pelaksana. | 0,1 |
| `spark.emr-serverless.driver.disk.type` | Jenis disk yang terpasang pada driver Spark. | Standar |
| `spark.emr-serverless.executor.disk.type` | Jenis disk yang melekat pada pelaksana Spark. | Standar |
| `spark.executor.cores` | Jumlah core yang digunakan setiap eksekutor. | 4 |
| `spark.executor.extraJavaOptions` | Opsi Java ekstra untuk eksekutor Spark. | `NULL` |
| `spark.executor.instances` | Jumlah kontainer pelaksana Spark untuk dialokasikan. | 3 |
| `spark.executor.memory` | Jumlah memori yang digunakan setiap eksekutor. | 14G |
| `spark.executorEnv.`*`[KEY]`* | Opsi yang menambahkan variabel lingkungan ke pelaksana Spark. | `NULL` |
| `spark.files` | Daftar file yang dipisahkan koma untuk masuk ke direktori kerja masing-masing pelaksana. Anda dapat mengakses jalur file dari file-file ini di pelaksana dengan`SparkFiles.get(`*`fileName`*`)`. | `NULL` |

| Kunci | Deskripsi | Nilai default |
| --- | --- | --- |
| `spark.hadoop.hive.metastore.client.factory.class` | Kelas implementasi metastore Hive. | `NULL` |
| `spark.jars` | Guci tambahan untuk ditambahkan ke classpath runtime driver dan executor. | `NULL` |
| `spark.network.crypto.enabled` | Opsi yang mengaktifkan RPC enkripsi AES berbasis. Ini termasuk protokol otentikasi yang ditambahkan di Spark 2.2.0. | `FALSE` |
| `spark.sql.warehouse.dir` | Lokasi default untuk database dan tabel terkelola. | Nilai `$PWD/spark-warehouse` |
| `spark.submit.pyFiles` | Daftar `.zip`, `.egg`, atau `.py` file yang dipisahkan koma untuk ditempatkan di aplikasi untuk `PYTHONPATH` Python. | `NULL` |

Tabel berikut mencantumkan parameter pengiriman Spark default.

| Kunci | Deskripsi | Nilai default |
| --- | --- | --- |
| `archives` | Daftar arsip yang dipisahkan koma yang diekstrak Spark ke dalam direktori kerja masing-masing pelaksana. | `NULL` |
| `class` | Kelas utama aplikasi (untuk aplikasi Java dan Scala). | `NULL` |

| Kunci | Deskripsi | Nilai default |
| --- | --- | --- |
| `conf` | Properti konfigurasi Spark arbitrer. | `NULL` |
| `driver-cores` | Jumlah core yang digunakan driver. | 4 |
| `driver-memory` | Jumlah memori yang digunakan pengemudi. | 14G |
| `executor-cores` | Jumlah core yang digunakan setiap eksekutor. | 4 |
| `executor-memory` | Jumlah memori yang digunakan eksekutor. | 14G |
| `files` | Daftar file yang dipisahkan koma untuk ditempatkan di direktori kerja masing-masing pelaksana. Anda dapat mengakses jalur file dari file-file ini di pelaksana dengan`SparkFiles.get(` *`fileName`*`)`. | `NULL` |
| `jars` | Daftar stoples yang dipisahkan koma untuk disertakan pada classpath driver dan eksekutor. | `NULL` |
| `num-executors` | Jumlah eksekutor yang akan diluncurkan. | 3 |
| `py-files` | Daftar`.zip`,`.egg`, atau `.py` file yang dipisahkan koma untuk ditempatkan di aplikasi untuk `PYTHONPATH` Python. | `NULL` |

| Kunci | Deskripsi | Nilai default |
| --- | --- | --- |
| `verbose` | Opsi yang mengaktifkan output debug tambahan. | `NULL` |

## Contoh percikan

Contoh berikut menunjukkan bagaimana menggunakan `StartJobRun` API untuk menjalankan script Python. Untuk end-to-end tutorial yang menggunakan contoh ini, lihatMemulai dengan Amazon Tanpa EMR Server. Anda dapat menemukan contoh tambahan tentang cara menjalankan PySpark pekerjaan dan menambahkan dependensi Python di repositori Sampel Tanpa Server. EMR GitHub

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "sparkSubmit": {
            "entryPoint": "s3://us-east-1.elasticmapreduce/emr-containers/samples/
wordcount/scripts/wordcount.py",
            "entryPointArguments": ["s3://DOC-EXAMPLE-BUCKET-OUTPUT/wordcount_output"],
            "sparkSubmitParameters": "--conf spark.executor.cores=1 --conf
 spark.executor.memory=4g --conf spark.driver.cores=1 --conf spark.driver.memory=4g --
conf spark.executor.instances=1"
        }
    }'
```

Contoh berikut menunjukkan bagaimana menggunakan `StartJobRun` API untuk menjalankan SparkJAR.

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "sparkSubmit": {
            "entryPoint": "/usr/lib/spark/examples/jars/spark-examples.jar",
            "entryPointArguments": ["1"],
            "sparkSubmitParameters": "--class org.apache.spark.examples.SparkPi --conf
 spark.executor.cores=4 --conf spark.executor.memory=20g --conf spark.driver.cores=4 --
conf spark.driver.memory=8g --conf spark.executor.instances=1"
        }
```

```
    }'
```

# Pekerjaan sarang

Anda dapat menjalankan pekerjaan Hive pada aplikasi dengan `type` parameter yang disetel ke`HIVE`. Jobs harus kompatibel dengan versi Hive yang kompatibel dengan versi EMR rilis Amazon. Misalnya, ketika Anda menjalankan pekerjaan pada aplikasi dengan Amazon EMR rilis 6.6.0, pekerjaan Anda harus kompatibel dengan Apache Hive 3.1.2. Untuk informasi tentang versi aplikasi untuk setiap rilis, lihatEMRVersi rilis Amazon Tanpa Server.

## Parameter pekerjaan sarang

Bila Anda menggunakan `StartJobRun`APIuntuk menjalankan pekerjaan Hive, Anda harus menentukan parameter berikut.

Parameter yang diperlukan

- Peran runtime pekerjaan sarang
- Parameter pengemudi pekerjaan sarang
- Parameter penggantian konfigurasi sarang

### Peran runtime pekerjaan sarang

Gunakan **`executionRoleArn`**ARNuntuk menentukan IAM peran yang digunakan aplikasi Anda untuk menjalankan pekerjaan Hive. Peran ini harus berisi izin berikut:

- Baca dari bucket S3 atau sumber data lain di mana data Anda berada
- Baca dari bucket atau awalan S3 tempat file kueri Hive dan file kueri init Anda berada
- Baca dan tulis ke bucket S3 tempat direktori Hive Scratch dan direktori gudang Hive Metastore Anda berada
- Menulis ke ember S3 di mana Anda ingin menulis hasil akhir Anda
- Menulis log ke bucket S3 atau awalan yang menentukan `S3MonitoringConfiguration`
- Akses ke KMS kunci jika Anda menggunakan KMS kunci untuk mengenkripsi data di bucket S3 Anda
- Akses ke AWS Katalog Data Glue

Jika pekerjaan Hive Anda membaca atau menulis data ke atau dari sumber data lain, tentukan izin yang sesuai dalam peran iniIAM. Jika Anda tidak memberikan izin ini untuk IAM peran tersebut, pekerjaan Anda mungkin gagal. Untuk informasi selengkapnya, lihat Peran runtime Job untuk Amazon Serverless EMR.

## Parameter pengemudi pekerjaan sarang

Gunakan **`jobDriver`**untuk memberikan masukan pada pekerjaan. Parameter driver pekerjaan hanya menerima satu nilai untuk jenis pekerjaan yang ingin Anda jalankan. Saat Anda menentukan `hive` sebagai jenis pekerjaan, EMR Tanpa Server meneruskan kueri Hive ke parameter. `jobDriver` Pekerjaan sarang memiliki parameter berikut:

- **`query`**— Ini adalah referensi di Amazon S3 ke file kueri Hive yang ingin Anda jalankan.
- **`parameters`**— Ini adalah properti konfigurasi Hive tambahan yang ingin Anda timpa. Untuk mengganti properti, teruskan ke parameter ini sebagai`--hiveconf` *`property=value`*. Untuk mengganti variabel, berikan mereka ke parameter ini sebagai`--hivevar` *`key=value`*.
- **`initQueryFile`**- Ini adalah file query init Hive. Hive menjalankan file ini sebelum kueri Anda dan dapat menggunakannya untuk menginisialisasi tabel.

## Parameter penggantian konfigurasi sarang

Gunakan **`configurationOverrides`**untuk mengganti properti konfigurasi tingkat pemantauan dan tingkat aplikasi. Parameter ini menerima JSON objek dengan dua bidang berikut:

- **`monitoringConfiguration`**— Gunakan bidang ini untuk menentukan Amazon S3 URL (`s3MonitoringConfiguration`) tempat Anda ingin pekerjaan EMR Tanpa Server menyimpan log pekerjaan Hive Anda. Pastikan Anda membuat bucket ini dengan cara yang sama Akun AWS yang meng-host aplikasi Anda, dan dalam hal yang sama Wilayah AWS Di mana pekerjaan Anda berjalan.
- **`applicationConfiguration`**— Anda dapat memberikan objek konfigurasi di bidang ini untuk mengganti konfigurasi default untuk aplikasi. Anda dapat menggunakan sintaks singkatan untuk menyediakan konfigurasi, atau Anda dapat mereferensikan objek konfigurasi dalam file. JSON Objek konfigurasi terdiri dari klasifikasi, properti, dan konfigurasi bersarang opsional. Properti terdiri dari pengaturan yang ingin Anda timpa dalam file itu. Anda dapat menentukan beberapa klasifikasi untuk beberapa aplikasi dalam satu JSON objek.

> **Note**
>
> Klasifikasi konfigurasi yang tersedia bervariasi menurut rilis Tanpa EMR Server tertentu. Misalnya, klasifikasi untuk Log4j kustom `spark-driver-log4j2` dan hanya `spark-executor-log4j2` tersedia dengan rilis 6.8.0 dan yang lebih tinggi.

Jika Anda melewati konfigurasi yang sama dalam penggantian aplikasi dan dalam parameter Hive, parameter Hive akan diprioritaskan. Daftar berikut memberi peringkat konfigurasi dari prioritas tertinggi hingga prioritas terendah.

- Konfigurasi yang Anda berikan sebagai bagian dari parameter Hive. `--hiveconf property=value`
- Konfigurasi yang Anda berikan sebagai bagian dari penggantian aplikasi Anda ketika Anda memulai pekerjaan.
- Konfigurasi yang Anda berikan sebagai bagian dari `runtimeConfiguration` saat Anda membuat aplikasi.
- Konfigurasi yang dioptimalkan yang EMR ditetapkan Amazon untuk rilis.
- Konfigurasi sumber terbuka default untuk aplikasi.

Untuk informasi selengkapnya tentang mendeklarasikan konfigurasi di tingkat aplikasi, dan mengganti konfigurasi selama menjalankan pekerjaan, lihat. Konfigurasi aplikasi default untuk Tanpa EMR Server

## Properti pekerjaan sarang

Tabel berikut mencantumkan properti wajib yang harus Anda konfigurasikan saat mengirimkan pekerjaan Hive.

| Pengaturan | Deskripsi |
| --- | --- |
| `hive.exec.scratchdir` | Lokasi Amazon S3 tempat EMR Tanpa Server membuat file sementara selama eksekusi pekerjaan Hive. |

| Pengaturan | Deskripsi |
| --- | --- |
| `hive.metastore.warehouse.dir` | Lokasi Amazon S3 database untuk tabel terkelola di Hive. |

Tabel berikut mencantumkan properti Hive opsional dan nilai defaultnya yang dapat Anda ganti saat mengirimkan pekerjaan Hive.

| Pengaturan | Deskripsi | Nilai default |
| --- | --- | --- |
| `fs.s3.customAWSCredentialsProvider` | Bagian AWS Penyedia kredensyal yang ingin Anda gunakan. | com.amazonaws.auth.defaultAWSCredentialsProviderChain |
| `fs.s3a.aws.credentials.provider` | Bagian AWS Penyedia kredensil yang ingin Anda gunakan dengan sistem file S3A. | com.amazonaws.auth.defaultAWSCredentialsProviderChain |
| `hive.auto.convert.join` | Opsi yang mengaktifkan konversi otomatis gabungan umum menjadi mapjoins, berdasarkan ukuran file input. | `TRUE` |
| `hive.auto.convert.join.noconditionaltask` | Opsi yang mengaktifkan pengoptimalan saat Hive mengonversi gabungan umum menjadi mapjoin berdasarkan ukuran file input. | `TRUE` |
| `hive.auto.convert.join.noconditionaltask.size` | Gabungan mengonversi langsung ke mapjoin di bawah ukuran ini. | Nilai optimal dihitung berdasarkan memori tugas Tez |
| `hive.cbo.enable` | Opsi yang mengaktifkan pengoptimalan berbasis biaya dengan kerangka Calcite. | `TRUE` |

| Pengaturan | Deskripsi | Nilai default |
| --- | --- | --- |
| `hive.cli.tez.session.async` | Opsi untuk memulai sesi Tez latar belakang saat kueri Hive Anda dikompilasi. Saat disetel ke`false`, Tez AM diluncurkan setelah kueri Hive Anda dikompilasi. | `TRUE` |
| `hive.compute.query.using.stats` | Opsi yang mengaktifkan Hive untuk menjawab pertanyaan tertentu dengan statistik yang disimpan di metastore. Untuk statistik dasar, atur `hive.stats.autogather` ke`TRUE`. Untuk koleksi kueri yang lebih canggih, jalankan`analyze table queries`. | `TRUE` |
| `hive.default.fileformat` | Format file default untuk `CREATE TABLE` pernyataan. Anda dapat secara eksplisit mengganti ini jika Anda menentukan `STORED AS [FORMAT]` dalam perintah Anda. `CREATE TABLE` | `TEXTFILE` |
| `hive.driver.cores` | Jumlah core yang digunakan untuk proses driver Hive. | 2 |
| `hive.driver.disk` | Ukuran disk untuk driver Hive. | 20G |
| `hive.driver.disk.type` | Jenis disk untuk driver Hive. | Standar |
| `hive.tez.disk.type` | Ukuran disk untuk pekerja tez. | Standar |

| Pengaturan | Deskripsi | Nilai default |
| --- | --- | --- |
| `hive.driver.memory` | Jumlah memori yang digunakan per proses driver Hive. Master Aplikasi Hive CLI dan Tez berbagi memori ini secara setara dengan 20% ruang kepala. | 6G |
| `hive.emr-serverless.launch.env.[ KEY]` | Opsi untuk mengatur variabel *KEY* lingkungan di semua proses khusus HIVE, seperti driver Hive Anda, Tez AM, dan tugas Tez. | |
| `hive.exec.dynamic.partition` | Opsi yang mengaktifkan partisi dinamis diDML/DDL. | `TRUE` |
| `hive.exec.dynamic.partition.mode` | Opsi yang menentukan apakah Anda ingin menggunakan modus ketat atau modus non-ketat. Dalam mode ketat, Anda harus menentukan setidaknya satu partisi statis jika Anda secara tidak sengaja menimpa semua partisi. Dalam mode non-ketat, semua partisi dibiarkan dinamis. | `strict` |
| `hive.exec.max.dynamic.partitions` | Jumlah maksimum partisi dinamis yang dibuat Hive secara total. | 1000 |
| `hive.exec.max.dynamic.partitions.pernode` | Jumlah maksimum partisi dinamis yang dibuat Hive di setiap node mapper dan reducer. | 100 |

| Pengaturan | Deskripsi | Nilai default |
| --- | --- | --- |
| `hive.exec.orc.split.strategy` | Mengharapkan salah satu nilai berikut:`BI`,`ETL`, atau`HYBRID`. Ini bukan konfigurasi tingkat pengguna. `BI`menentukan bahwa Anda ingin menghabiskan lebih sedikit waktu dalam pembuatan terpisah dibandingkan dengan eksekusi kueri. `ETL`menentukan bahwa Anda ingin menghabiskan lebih banyak waktu dalam generasi terpisah. `HYBRID`menentukan pilihan strategi di atas berdasarkan heuristik. | `HYBRID` |
| `hive.exec.reducers.bytes.per.reducer` | Ukuran per peredam. Defaultnya adalah 256 MB. Jika ukuran input 1G, pekerjaan menggunakan 4 reduksi. | 256000000 |
| `hive.exec.reducers.max` | Jumlah maksimum reduksi. | 256 |
| `hive.exec.stagingdir` | Nama direktori yang menyimpan file sementara yang Hive buat di dalam lokasi tabel dan di lokasi direktori awal yang ditentukan dalam `hive.exec.scratchdir` properti. | `.hive-staging` |

| Pengaturan | Deskripsi | Nilai default |
| --- | --- | --- |
| hive.fetch.task.conversion | Mengharapkan salah satu nilai berikut:NONE,MINIMAL, atauMORE. Hive dapat mengonversi kueri pilih menjadi satu FETCH tugas. Ini meminimalkan latensi. | MORE |
| hive.groupby.position.alias | Opsi yang menyebabkan Hive menggunakan alias posisi kolom dalam GROUP BY pernyataan. | FALSE |
| hive.input.format | Format input default. Atur ke HiveInputFormat jika Anda mengalami masalah denganCombineHiveInputFormat . | org.apache.hadoop.hive.ql.io.CombineHiveInputFormat |
| hive.log.explain.output | Opsi yang mengaktifkan penjelasan output diperpanjang untuk kueri apa pun di log Hive Anda. | FALSE |
| hive.log.level | Tingkat penebangan sarang. | INFO |
| hive.mapred.reduce.tasks.speculative.execution | Opsi yang mengaktifkan peluncuran spekulatif untuk reduksi. Hanya didukung dengan Amazon EMR 6.10.x dan yang lebih rendah. | TRUE |

| Pengaturan | Deskripsi | Nilai default |
| --- | --- | --- |
| `hive.max-task-containers` | Jumlah maksimum kontainer bersamaan. Memori mapper yang dikonfigurasi dikalikan dengan nilai ini untuk menentukan memori yang tersedia yang membagi komputasi dan penggunaan preemption tugas. | 1000 |
| `hive.merge.mapfiles` | Opsi yang menyebabkan file kecil bergabung di akhir pekerjaan khusus peta. | `TRUE` |
| `hive.merge.size.per.task` | Ukuran file gabungan di akhir pekerjaan. | 256000000 |
| `hive.merge.tezfiles` | Opsi yang mengaktifkan penggabungan file kecil di akhir DAG Tez. | `FALSE` |
| `hive.metastore.client.factory.class` | Nama kelas pabrik yang menghasilkan objek yang mengimplementasikan `IMetaStoreClient` antarmuka. | `com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory` |
| `hive.metastore.glue.catalogid` | Jika AWS Glue Data Catalog bertindak sebagai metastore tetapi berjalan di tempat yang berbeda Akun AWS dari pekerjaan, ID dari Akun AWS di mana pekerjaan berjalan. | `NULL` |

| Pengaturan | Deskripsi | Nilai default |
| --- | --- | --- |
| `hive.metastore.uris` | Penghematan URI yang digunakan klien metastore untuk terhubung ke metastore jarak jauh. | `NULL` |
| `hive.optimize.ppd` | Opsi yang menyalakan predikat pushdown. | `TRUE` |
| `hive.optimize.ppd.storage` | Opsi yang mengaktifkan predikat pushdown ke penangan penyimpanan. | `TRUE` |
| `hive.orderby.position.alias` | Opsi yang menyebabkan Hive menggunakan alias posisi kolom dalam `ORDER BY` pernyataan. | `TRUE` |
| `hive.prewarm.enabled` | Opsi yang menyalakan wadah prewarm untuk Tez. | `FALSE` |
| `hive.prewarm.numcontainers` | Jumlah wadah untuk pra-hangat untuk Tez. | 10 |
| `hive.stats.autogather` | Opsi yang menyebabkan Hive mengumpulkan statistik dasar secara otomatis selama `INSERT OVERWRITE` perintah. | `TRUE` |
| `hive.stats.fetch.column.stats` | Opsi yang mematikan pengambilan statistik kolom dari metastore. Pengambilan statistik kolom bisa mahal ketika jumlah kolom tinggi. | `FALSE` |

| Pengaturan | Deskripsi | Nilai default |
| --- | --- | --- |
| `hive.stats.gather.num.threads` | Jumlah utas yang digunakan `partialscan` dan `noscan` menganalisis perintah untuk tabel yang dipartisi. Ini hanya berlaku untuk format file yang mengimplementasikan `StatsProvidingRecordReader` (like`ORC`). | 10 |
| `hive.strict.checks.cartesian.product` | Opsi yang mengaktifkan pemeriksaan gabungan Cartesian yang ketat. Pemeriksaan ini melarang produk Cartesian (gabungan silang). | `FALSE` |
| `hive.strict.checks.type.safety` | Opsi yang mengaktifkan pemeriksaan keamanan tipe ketat dan mematikan perbandingan `bigint` dengan keduanya `string` dan`double`. | `TRUE` |
| `hive.support.quoted.identifiers` | Mengharapkan nilai `NONE` atau`COLUMN`. `NONE`menyiratkan hanya karakter alfanumerik dan garis bawah yang valid dalam pengidentifikasi. `COLUMN`menyiratkan nama kolom dapat berisi karakter apa pun. | `COLUMN` |

| Pengaturan | Deskripsi | Nilai default |
| --- | --- | --- |
| hive.tez.auto.reducer.parallelism | Opsi yang mengaktifkan fitur paralelisme peredam otomatis Tez. Hive masih memperkirakan ukuran data dan menetapkan perkiraan paralelisme. Tez mengambil sampel ukuran keluaran simpul sumber dan menyesuaikan perkiraan saat runtime seperlunya. | TRUE |
| hive.tez.container.size | Jumlah memori yang digunakan per proses tugas Tez. | 6144 |
| hive.tez.cpu.vcores | Jumlah core yang digunakan untuk setiap tugas Tez. | 2 |
| hive.tez.disk.size | Ukuran disk untuk setiap wadah tugas. | 20G |
| hive.tez.input.format | Format input untuk generasi split di Tez AM. | org.apache.hadoop.hive.ql.io.HiveInputFormat |
| hive.tez.min.partition.factor | Batas bawah reduksi yang ditentukan Tez saat Anda mengaktifkan paralelisme peredam otomatis. | 0,25 |
| hive.vectorized.execution.enabled | Opsi yang mengaktifkan mode vektor eksekusi kueri. | TRUE |
| hive.vectorized.execution.reduce.enabled | Opsi yang mengaktifkan mode vektor dari sisi pengurangan eksekusi kueri. | TRUE |

| Pengaturan | Deskripsi | Nilai default |
| --- | --- | --- |
| `javax.jdo.option.ConnectionDriverName` | Nama kelas pengemudi untuk JDBC metastore. | `org.apache.derby.jdbc.EmbeddedDriver` |
| `javax.jdo.option.ConnectionPassword` | Kata sandi yang terkait dengan database metastore. | `NULL` |
| `javax.jdo.option.ConnectionURL` | String JDBC penghubung untuk JDBC metastore. | `jdbc:derby:;databaseName=metastore_db;create=true` |
| `javax.jdo.option.ConnectionUserName` | Nama pengguna yang terkait dengan database metastore. | `NULL` |
| `mapreduce.input.fileinputformat.split.maxsize` | Ukuran maksimum split selama komputasi split saat format input Anda. `org.apache.hadoop.hive.ql.io.CombineHiveInputFormat` Nilai 0 menunjukkan tidak ada batas. | 0 |
| `tez.am.dag.cleanup.on.completion` | Opsi yang mengaktifkan pembersihan data acak saat DAG selesai. | `TRUE` |
| `tez.am.emr-serverless.launch.env.[KEY]` | Pilihan untuk mengatur variabel *KEY* lingkungan dalam proses Tez AM. Untuk Tez AM, nilai ini mengesampingkan nilainya. `hive.emr-serverless.launch.env.[KEY]` | |
| `tez.am.log.level` | Level pencatatan root yang diteruskan EMR Tanpa Server ke master aplikasi Tez. | `INFO` |

| Pengaturan | Deskripsi | Nilai default |
| --- | --- | --- |
| `tez.am.sleep.time.before.exit.millis` | EMRTanpa server harus mendorong ATS peristiwa setelah periode waktu ini setelah permintaan shutdown AM. | 0 |
| `tez.am.speculation.enabled` | Opsi yang menyebabkan peluncuran spekulatif tugas yang lebih lambat. Ini dapat membantu mengurangi latensi pekerjaan ketika beberapa tugas berjalan lebih lambat karena mesin yang buruk atau lambat. Hanya didukung dengan Amazon EMR 6.10.x dan yang lebih rendah. | `FALSE` |
| `tez.am.task.max.failed.attempts` | Jumlah maksimum upaya yang dapat gagal untuk tugas tertentu sebelum tugas gagal. Nomor ini tidak menghitung upaya yang dihentikan secara manual. | 3 |
| `tez.am.vertex.cleanup.height` | Jarak di mana, jika semua simpul dependen selesai, Tez AM akan menghapus data vertex shuffle. Fitur ini dimatikan ketika nilainya 0. Amazon EMR versi 6.8.0 dan yang lebih baru mendukung fitur ini. | 0 |

| Pengaturan | Deskripsi | Nilai default |
| --- | --- | --- |
| `tez.client.asynchronous-stop` | Opsi yang menyebabkan EMR Serverless mendorong ATS peristiwa sebelum mengakhiri driver Hive. | `FALSE` |
| `tez.grouping.max-size` | Batas ukuran atas (dalam byte) dari pemisahan yang dikelompokkan. Batas ini mencegah perpecahan yang terlalu besar. | 1073741824 |
| `tez.grouping.min-size` | Batas ukuran yang lebih rendah (dalam byte) dari pemisahan yang dikelompokkan. Batas ini mencegah terlalu banyak perpecahan kecil. | 16777216 |
| `tez.runtime.io.sort.mb` | Ukuran buffer lunak saat Tez mengurutkan output diurutkan. | Nilai optimal dihitung berdasarkan memori tugas Tez |
| `tez.runtime.unordered.output.buffer.size-mb` | Ukuran buffer yang akan digunakan jika Tez tidak menulis langsung ke disk. | Nilai optimal dihitung berdasarkan memori tugas Tez |

| Pengaturan | Deskripsi | Nilai default |
| --- | --- | --- |
| `tez.shuffle-vertex-manager.max-src-fraction` | Fraksi tugas sumber yang harus diselesaikan sebelum EMR Tanpa Server menjadwalkan semua tugas untuk simpul saat ini (dalam kasus koneksi). `ScatterGather` Jumlah tugas yang siap untuk penjadwalan pada skala simpul saat ini secara linier antara dan. `min-fraction` `max-fraction` Ini default nilai default atau`tez.shuffle-vertex-manager.min-src-fraction` , mana yang lebih besar. | 0,75 |
| `tez.shuffle-vertex-manager.min-src-fraction` | Fraksi tugas sumber yang harus diselesaikan sebelum EMR Serverless menjadwalkan tugas untuk simpul saat ini (dalam kasus koneksi). `ScatterGather` | 0,25 |
| `tez.task.emr-serverless.launch.env.[` *KEY*`]` | Opsi untuk mengatur variabel *KEY* lingkungan dalam proses tugas Tez. Untuk tugas Tez, nilai ini mengesampingkan nilainya. `hive.emr-serverless.launch.env.[` *KEY*`]` | |
| `tez.task.log.level` | Level logging root yang EMR diserahkan Tanpa Server ke tugas Tez. | `INFO` |

| Pengaturan | Deskripsi | Nilai default |
| --- | --- | --- |
| `tez.yarn.ats.event.flush.timeout.millis` | Jumlah maksimum waktu AM harus menunggu acara dibilas sebelum dimatikan. | 300000 |

## Contoh pekerjaan sarang

Contoh kode berikut menunjukkan bagaimana menjalankan query Hive dengan. `StartJobRun` API

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "hive": {
            "query": "s3://DOC-EXAMPLE-BUCKET/emr-serverless-hive/query/hive-query.ql",
            "parameters": "--hiveconf hive.log.explain.output=false"
        }
    }' \
    --configuration-overrides '{
        "applicationConfiguration": [{
            "classification": "hive-site",
            "properties": {
                "hive.exec.scratchdir": "s3://DOC-EXAMPLE-BUCKET/emr-serverless-hive/
hive/scratch",
                "hive.metastore.warehouse.dir": "s3://DOC-EXAMPLE-BUCKET/emr-
serverless-hive/hive/warehouse",
                "hive.driver.cores": "2",
                "hive.driver.memory": "4g",
                "hive.tez.container.size": "4096",
                "hive.tez.cpu.vcores": "1"
            }
        }]
    }'
```

Anda dapat menemukan contoh tambahan tentang cara menjalankan pekerjaan Hive di repositori Sampel EMRTanpa Server. GitHub

# EMRKetahanan Job Tanpa Server

EMRRilis tanpa server 7.1.0 dan yang lebih tinggi menyertakan dukungan untuk ketahanan pekerjaan, sehingga secara otomatis mencoba ulang pekerjaan yang gagal tanpa masukan manual dari Anda. Manfaat lain dari ketahanan kerja adalah bahwa EMR Serverless memindahkan pekerjaan ke Availability Zone (AZ) yang berbeda jika AZ mengalami masalah apa pun.

Untuk mengaktifkan ketahanan pekerjaan untuk suatu pekerjaan, tetapkan kebijakan coba lagi untuk pekerjaan Anda. Kebijakan coba lagi memastikan bahwa EMR Tanpa Server secara otomatis memulai ulang pekerjaan jika gagal pada titik mana pun. Kebijakan coba lagi didukung untuk pekerjaan batch dan streaming, sehingga Anda dapat menyesuaikan ketahanan pekerjaan sesuai dengan kasus penggunaan Anda. Tabel berikut membandingkan perilaku dan perbedaan ketahanan kerja di seluruh pekerjaan batch dan streaming.

| | Tugas Batch | Lowongan kerja Streaming |
|---|---|---|
| Perilaku default | Tidak menjalankan kembali pekerjaan. | Selalu mencoba menjalank an pekerjaan karena aplikasi membuat pos pemeriksaan saat menjalankan pekerjaan. |
| Poin coba lagi | Pekerjaan batch tidak memiliki pos pemeriksa an, jadi EMR Tanpa Server selalu menjalankan kembali pekerjaan dari awal. | Pekerjaan streaming mendukung pos pemeriksa an, sehingga Anda dapat mengonfigurasi kueri streaming untuk menyimpan status waktu proses dan melanjutkan ke lokasi pos pemeriksaan di Amazon S3. EMRTanpa server melanjutk an pekerjaan yang dijalankan dari pos pemeriksaan. Untuk informasi selengkapnya, lihat Memulihkan dari kegagalan dengan Checkpointing di dokumentasi Apache Spark. |

| | Tugas Batch | Lowongan kerja Streaming |
|---|---|---|
| Maksimal upaya coba lagi | Memungkinkan maksimal 10 percobaan ulang. | Pekerjaan streaming memiliki kontrol pencegahan thrash bawaan, sehingga aplikasi berhenti mencoba lagi pekerjaan jika terus gagal setelah satu jam. Jumlah default percobaan ulang dalam satu jam adalah lima upaya. Anda dapat mengonfig urasi jumlah percobaan ulang ini menjadi antara 1 atau 10. Anda tidak dapat menyesuai kan jumlah upaya maksimum. Nilai 1 menunjukkan tidak ada percobaan ulang. |

Saat EMR Serverless mencoba menjalankan kembali pekerjaan, pekerjaan tersebut juga mengindeks pekerjaan dengan nomor percobaan, sehingga Anda dapat melacak siklus hidup pekerjaan di seluruh upayanya.

Anda dapat menggunakan API operasi EMR Tanpa Server atau AWS CLI untuk mengubah ketahanan pekerjaan atau melihat informasi yang berkaitan dengan ketahanan kerja. Untuk informasi selengkapnya, lihat panduan EMRTanpa Server API.

Secara default, EMR Tanpa Server tidak menjalankan kembali pekerjaan batch. Untuk mengaktifkan percobaan ulang untuk pekerjaan batch, konfigurasikan `maxAttempts` parameter saat memulai pekerjaan batch. `maxAttempts`Parameter ini hanya berlaku untuk pekerjaan batch. Defaultnya adalah 1, yang berarti tidak menjalankan kembali pekerjaan. Nilai yang diterima adalah 1 hingga 10, inklusif.

Contoh berikut menunjukkan bagaimana menentukan jumlah maksimal 10 upaya ketika memulai pekerjaan berjalan.

```
aws emr-serverless start-job-run
 --application-id <APPLICATION_ID> \
 --execution-role-arn <JOB_EXECUTION_ROLE> \
```

```
 --mode 'BATCH' \
 --retry-policy '{
    "maxAttempts": 10
 }' \
 --job-driver '{
    "sparkSubmit": {
         "entryPoint": "/usr/lib/spark/examples/jars/spark-examples-does-not-
exist.jar",
         "entryPointArguments": ["1"],
         "sparkSubmitParameters": "--class org.apache.spark.examples.SparkPi"
     }
}'
```

EMRTanpa server mencoba ulang pekerjaan streaming tanpa batas waktu jika gagal. Untuk mencegah meronta-ronta karena kegagalan berulang yang tidak dapat dipulihkan, gunakan `maxFailedAttemptsPerHour` untuk mengonfigurasi kontrol pencegahan thrash untuk streaming percobaan ulang pekerjaan. Parameter ini memungkinkan Anda menentukan jumlah maksimum upaya gagal yang diizinkan dengan satu jam sebelum EMR Serverless berhenti mencoba lagi. Defaultnya adalah lima. Nilai yang diterima adalah 1 hingga 10, inklusif.

```
aws emr-serverless start-job-run
 --application-id <APPPLICATION_ID> \
 --execution-role-arn <JOB_EXECUTION_ROLE> \
 --mode 'STREAMING' \
 --retry-policy '{
    "maxFailedAttemptsPerHour": 7
 }' \
 --job-driver '{
    "sparkSubmit": {
         "entryPoint": "/usr/lib/spark/examples/jars/spark-examples-does-not-
exist.jar",
         "entryPointArguments": ["1"],
         "sparkSubmitParameters": "--class org.apache.spark.examples.SparkPi"
     }
}'
```

Anda juga dapat menggunakan API operasi lari pekerjaan lainnya untuk mendapatkan informasi tentang pekerjaan. Misalnya, Anda dapat menggunakan `attempt` parameter dengan `GetJobRun` operasi untuk mendapatkan detail tentang upaya pekerjaan tertentu. Jika Anda tidak menyertakan `attempt` parameter, operasi mengembalikan informasi tentang upaya terbaru.

```
aws emr-serverless get-job-run \
    --job-run-id job-run-id \
    --application-id application-id \
    --attempt 1
```

`ListJobRunAttempts`Operasi mengembalikan informasi tentang semua upaya yang terkait dengan menjalankan pekerjaan.

```
aws emr-serverless list-job-run-attempts \
  --application-id application-id \
  --job-run-id job-run-id
```

`GetDashboardForJobRun`Operasi membuat dan mengembalikan URL yang dapat Anda gunakan untuk mengakses aplikasi UIs untuk menjalankan pekerjaan. `attempt`Parameter memungkinkan Anda mendapatkan URL untuk upaya tertentu. Jika Anda tidak menyertakan `attempt` parameter, operasi mengembalikan informasi tentang upaya terbaru.

```
aws emr-serverless get-dashboard-for-job-run \
    --application-id application-id \
    --job-run-id job-run-id \
    --attempt 1
```

## Memantau pekerjaan dengan kebijakan coba lagi

Dukungan ketahanan Job juga menambahkan acara baru EMRServerless job run retry. EMRTanpa server menerbitkan acara ini pada setiap percobaan ulang pekerjaan. Anda dapat menggunakan notifikasi ini untuk melacak percobaan ulang pekerjaan. Untuk informasi selengkapnya tentang acara, lihat EventBridge acara Amazon.

## Logging dengan kebijakan coba lagi

Setiap kali EMR Serverless mencoba kembali pekerjaan, upaya tersebut menghasilkan kumpulan lognya sendiri. Untuk memastikan bahwa EMR Tanpa Server berhasil mengirimkan log ini ke Amazon S3 dan CloudWatch Amazon tanpa menimpa EMR apa pun, Tanpa Server menambahkan awalan ke format CloudWatch jalur log S3 dan nama aliran log untuk menyertakan nomor percobaan pekerjaan.

Berikut ini adalah contoh seperti apa formatnya.

```
'/applications/<applicationId>/jobs/<jobId>/attempts/<attemptNumber>/'.
```

Format ini memastikan EMR Tanpa Server menerbitkan semua log untuk setiap upaya pekerjaan ke lokasi yang ditunjuk sendiri di Amazon S3 dan. CloudWatch Untuk detail selengkapnya, lihat Menyimpan log.

> **Note**
>
> EMRTanpa server hanya menggunakan format awalan ini dengan semua pekerjaan streaming dan pekerjaan batch apa pun yang telah diaktifkan coba lagi.

# Konfigurasi metastore

Metastore Hive adalah lokasi terpusat yang menyimpan informasi struktural tentang tabel Anda, termasuk skema, nama partisi, dan tipe data. Dengan EMR Tanpa Server, Anda dapat mempertahankan metadata tabel ini dalam metastore yang memiliki akses ke pekerjaan Anda.

Anda memiliki dua opsi untuk metastore Hive:

- Bagian AWS Katalog Data Glue
- Metastore Apache Hive eksternal

## Menggunakan AWS Glue Data Catalog sebagai metastore

Anda dapat mengonfigurasi pekerjaan Spark and Hive Anda untuk menggunakan AWS Glue Data Catalog sebagai metastore nya. Kami merekomendasikan konfigurasi ini ketika Anda memerlukan metastore persisten atau metastore yang dibagikan oleh berbagai aplikasi, layanan, atau Akun AWS. Untuk informasi selengkapnya tentang Katalog Data, lihat Mengisi AWS Katalog Data Glue. Untuk informasi tentang AWS Harga Glue, lihat AWS Harga Glue.

Anda dapat mengonfigurasi pekerjaan EMR Tanpa Server Anda untuk menggunakan AWS Glue Data Catalog baik dalam hal yang sama Akun AWS sebagai aplikasi Anda, atau dalam yang berbeda Akun AWS.

### Konfigurasikan AWS Katalog Data Glue

Untuk mengkonfigurasi Katalog Data, pilih jenis aplikasi EMR Tanpa Server yang ingin Anda gunakan.

Spark

Saat Anda menggunakan EMR Studio untuk menjalankan pekerjaan Anda dengan aplikasi EMR Serverless Spark, AWS Glue Data Catalog adalah metastore default.

Ketika Anda menggunakan SDKs atau AWS CLI, Anda dapat mengatur `spark.hadoop.hive.metastore.client.factory.class` konfigurasi ke `com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory` dalam `sparkSubmit` parameter menjalankan pekerjaan Anda. Contoh berikut menunjukkan cara mengkonfigurasi Katalog Data dengan AWS CLI.

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "sparkSubmit": {
            "entryPoint": "s3://DOC-EXAMPLE-BUCKET/code/pyspark/extreme_weather.py",
            "sparkSubmitParameters": "--conf
 spark.hadoop.hive.metastore.client.factory.class=com.amazonaws.glue.catalog.metastore.AWSGl
 --conf spark.driver.cores=1 --conf spark.driver.memory=3g --conf
 spark.executor.cores=4 --conf spark.executor.memory=3g"
        }
    }'
```

Atau, Anda dapat mengatur konfigurasi ini ketika Anda membuat yang baru `SparkSession` dalam kode Spark Anda.

```
from pyspark.sql import SparkSession

spark = (
    SparkSession.builder.appName("SparkSQL")
    .config(
        "spark.hadoop.hive.metastore.client.factory.class",
        "com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory",
    )
    .enableHiveSupport()
    .getOrCreate()
)

# we can query tables with SparkSQL
spark.sql("SHOW TABLES").show()
```

```
# we can also them with native Spark
print(spark.catalog.listTables())
```

Hive

Untuk aplikasi EMR Serverless Hive, Katalog Data adalah metastore default. Artinya, ketika Anda menjalankan pekerjaan pada aplikasi EMR Serverless Hive, Hive mencatat informasi metastore di Katalog Data dalam hal yang sama Akun AWS sebagai aplikasi Anda. Anda tidak memerlukan virtual private cloud (VPC) untuk menggunakan Katalog Data sebagai metastore Anda.

Untuk mengakses tabel metastore Hive, tambahkan yang diperlukan AWS Kebijakan Glue yang diuraikan dalam Menyiapkan IAM Izin untuk AWS Glue.

## Konfigurasikan akses lintas akun untuk Tanpa EMR Server dan AWS Katalog Data Glue

Untuk mengatur akses lintas akun untuk EMR Tanpa Server, Anda harus terlebih dahulu masuk ke yang berikut Akun AWS:

- `AccountA`— Sebuah Akun AWS di mana Anda telah membuat aplikasi EMR Tanpa Server.
- `AccountB`— Sebuah Akun AWS yang berisi AWS Glue Data Catalog yang Anda inginkan agar pekerjaan EMR Tanpa Server Anda dapat diakses.

1. Pastikan administrator atau identitas resmi lainnya `AccountB` melampirkan kebijakan sumber daya ke Katalog Data di`AccountB`. Kebijakan ini memberikan izin lintas akun `AccountA` tertentu untuk melakukan operasi pada sumber daya dalam katalog. `AccountB`

```
{
  "Version" : "2012-10-17",
  "Statement" : [ {
    "Effect" : "Allow",
    "Principal": {
        "AWS": [
            "arn:aws:iam::accountA:role/job-runtime-role-A"
        ]},
    "Action" : [
        "glue:GetDatabase",
        "glue:CreateDatabase",
        "glue:GetDataBases",
```

```
        "glue:CreateTable",
        "glue:GetTable",
        "glue:UpdateTable",
        "glue:DeleteTable",
        "glue:GetTables",
        "glue:GetPartition",
        "glue:GetPartitions",
        "glue:CreatePartition",
        "glue:BatchCreatePartition",
        "glue:GetUserDefinedFunctions"
    ],
    "Resource": ["arn:aws:glue:region:AccountB:catalog"]
  } ]
}
```

2. Tambahkan IAM kebijakan ke peran runtime pekerjaan EMR Tanpa Server `AccountA` agar peran tersebut dapat mengakses sumber daya Katalog Data di. `AccountB`

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "glue:GetDatabase",
        "glue:CreateDatabase",
        "glue:GetDataBases",
        "glue:CreateTable",
        "glue:GetTable",
        "glue:UpdateTable",
        "glue:DeleteTable",
        "glue:GetTables",
        "glue:GetPartition",
        "glue:GetPartitions",
        "glue:CreatePartition",
        "glue:BatchCreatePartition",
        "glue:GetUserDefinedFunctions"
      ],
      "Resource": ["arn:aws:glue:region:AccountB:catalog"]
    }
  ]
}
```

3. Mulai menjalankan pekerjaan Anda. Langkah ini sedikit berbeda tergantung pada `AccountA` jenis aplikasi EMR Tanpa Server.

   Spark

   Tetapkan `spark.hadoop.hive.metastore.glue.catalogid` properti dalam `hive-site` klasifikasi seperti yang ditunjukkan pada contoh berikut. Ganti *`AkuntB-Katalog-ID`* dengan ID Katalog Data di`AccountB`.

```
aws emr-serverless start-job-run \
--application-id "application-id" \
--execution-role-arn "job-role-arn" \
--job-driver '{
    "sparkSubmit": {
        "query": "s3://DOC-EXAMPLE-BUCKET/hive/scripts/create_table.sql",
        "parameters": "--hiveconf hive.exec.scratchdir=s3://DOC-EXAMPLE-BUCKET/
hive/scratch --hiveconf hive.metastore.warehouse.dir=s3://DOC-EXAMPLE-BUCKET/
hive/warehouse"
    }
  }' \
--configuration-overrides '{
    "applicationConfiguration": [{
        "classification": "hive-site",
        "properties": {
            "spark.hadoop.hive.metastore.glue.catalogid": "AccountB-catalog-id"
        }
    }]
}'
```

   Hive

   Tetapkan `hive.metastore.glue.catalogid` properti dalam `hive-site` klasifikasi seperti yang ditunjukkan pada contoh berikut. Ganti *`AkuntB-Katalog-ID`* dengan ID Katalog Data di`AccountB`.

```
aws emr-serverless start-job-run \
--application-id "application-id" \
--execution-role-arn "job-role-arn" \
--job-driver '{
    "hive": {
    "query": "s3://DOC-EXAMPLE-BUCKET/hive/scripts/create_table.sql",
```

```
    "parameters": "--hiveconf hive.exec.scratchdir=s3://DOC-EXAMPLE-BUCKET/hive/
scratch --hiveconf hive.metastore.warehouse.dir=s3://DOC-EXAMPLE-BUCKET/hive/
warehouse"
    }
}' \
--configuration-overrides '{
    "applicationConfiguration": [{
        "classification": "hive-site",
        "properties": {
            "hive.metastore.glue.catalogid": "AccountB-catalog-id"
        }
    }]
}'
```

### Pertimbangan saat menggunakan AWS Katalog Data Glue

Anda dapat menambahkan tambahan JARs dengan `ADD JAR` skrip Hive Anda. Untuk pertimbangan tambahan, lihat Pertimbangan saat menggunakan AWS Katalog Data Glue.

## Menggunakan metastore Hive eksternal

Anda dapat mengonfigurasi pekerjaan Spark dan Hive EMR Tanpa Server untuk terhubung ke metastore Hive eksternal, seperti Amazon Aurora atau Amazon for My. RDS SQL Bagian ini menjelaskan cara menyiapkan metastore Amazon RDS Hive, mengonfigurasiVPC, dan mengonfigurasi pekerjaan EMR Tanpa Server untuk menggunakan metastore eksternal.

### Buat metastore Hive eksternal

1. Buat Amazon Virtual Private Cloud (AmazonVPC) dengan subnet pribadi dengan mengikuti petunjuk di Buat VPC.
2. Buat aplikasi EMR Tanpa Server Anda dengan Amazon baru VPC dan subnet pribadi Anda. Ketika Anda mengkonfigurasi aplikasi EMR Serverless Anda dengan aVPC, pertama-tama menyediakan sebuah elastic network interface untuk setiap subnet yang Anda tentukan. Kemudian melampirkan grup keamanan yang Anda tentukan ke antarmuka jaringan itu. Ini memberikan kontrol akses aplikasi Anda. Untuk detail selengkapnya tentang cara mengaturVPC, lihatMengkonfigurasi akses VPC.
3. Buat SQL database My SQL atau Aurora Postgre di subnet pribadi di Amazon Anda. VPC Untuk informasi tentang cara membuat RDS database Amazon, lihat Membuat instans Amazon RDS DB.

4. Ubah grup keamanan database Saya SQL atau Aurora Anda untuk mengizinkan JDBC koneksi dari grup keamanan EMR Tanpa Server Anda dengan mengikuti langkah-langkah dalam Memodifikasi instans Amazon DB. RDS Tambahkan aturan untuk lalu lintas masuk ke grup RDS keamanan dari salah satu grup keamanan EMR Tanpa Server Anda.

| Tipe | Protokol | Rentang port | Sumber |
| --- | --- | --- | --- |
| Semua TCP | TCP | 3306 | `emr-serverless-security-group` |

## Konfigurasikan opsi Spark

### Menggunakan JDBC

Untuk mengonfigurasi aplikasi EMR Serverless Spark agar tersambung ke metastore Hive berdasarkan instans Amazon for RDS My atau SQL Amazon Aurora My, gunakan koneksi. SQL JDBC Lewati `mariadb-connector-java.jar` dengan `--jars` dalam `spark-submit` parameter lari pekerjaan Anda.

```
aws emr-serverless start-job-run \
  --application-id "application-id" \
  --execution-role-arn "job-role-arn" \
  --job-driver '{
        "sparkSubmit": {
            "entryPoint": "s3://DOC-EXAMPLE-BUCKET/scripts/spark-jdbc.py",
            "sparkSubmitParameters": "--jars s3://DOC-EXAMPLE-BUCKET/mariadb-connector-
java.jar
            --conf
 spark.hadoop.javax.jdo.option.ConnectionDriverName=org.mariadb.jdbc.Driver
            --conf spark.hadoop.javax.jdo.option.ConnectionUserName=<connection-user-
name>
            --conf spark.hadoop.javax.jdo.option.ConnectionPassword=<connection-
password>
            --conf spark.hadoop.javax.jdo.option.ConnectionURL=<JDBC-Connection-
string>
            --conf spark.driver.cores=2
            --conf spark.executor.memory=10G
            --conf spark.driver.memory=6G
            --conf spark.executor.cores=4"
```

```
        }
    }' \
    --configuration-overrides '{
        "monitoringConfiguration": {
        "s3MonitoringConfiguration": {
            "logUri": "s3://DOC-EXAMPLE-BUCKET/spark/logs/"
        }
    }
}'
```

Contoh kode berikut adalah skrip entrypoint Spark yang berinteraksi dengan metastore Hive di Amazon. RDS

```
from os.path import expanduser, join, abspath
from pyspark.sql import SparkSession
from pyspark.sql import Row
# warehouse_location points to the default location for managed databases and tables
warehouse_location = abspath('spark-warehouse')
spark = SparkSession \
    .builder \
    .config("spark.sql.warehouse.dir", warehouse_location) \
    .enableHiveSupport() \
    .getOrCreate()
spark.sql("SHOW DATABASES").show()
spark.sql("CREATE EXTERNAL TABLE `sampledb`.`sparknyctaxi`(`dispatching_base_num`
 string, `pickup_datetime` string, `dropoff_datetime` string, `pulocationid` bigint,
 `dolocationid` bigint, `sr_flag` bigint) STORED AS PARQUET LOCATION 's3://<s3 prefix>/
nyctaxi_parquet/'")
spark.sql("SELECT count(*) FROM sampledb.sparknyctaxi").show()
spark.stop()
```

Menggunakan layanan hemat

Anda dapat mengonfigurasi aplikasi EMR Serverless Hive untuk terhubung ke metastore Hive berdasarkan instans Amazon for RDS My atau SQL Amazon Aurora My. SQL Untuk melakukan ini, jalankan server penghematan pada node master dari cluster Amazon EMR yang ada. Opsi ini sangat ideal jika Anda sudah memiliki EMR cluster Amazon dengan server hemat yang ingin Anda gunakan untuk menyederhanakan konfigurasi pekerjaan Tanpa EMR Server Anda.

```
aws emr-serverless start-job-run \
  --application-id "application-id" \
  --execution-role-arn "job-role-arn" \
```

```
  --job-driver '{
        "sparkSubmit": {
            "entryPoint": "s3://DOC-EXAMPLE-BUCKET/thriftscript.py",
            "sparkSubmitParameters": "--jars s3://DOC-EXAMPLE-BUCKET/mariadb-connector-
java.jar
            --conf spark.driver.cores=2
            --conf spark.executor.memory=10G
            --conf spark.driver.memory=6G
            --conf spark.executor.cores=4"
        }
    }' \
    --configuration-overrides '{
        "monitoringConfiguration": {
        "s3MonitoringConfiguration": {
            "logUri": "s3://DOC-EXAMPLE-BUCKET/spark/logs/"
        }
    }
}'
```

Contoh kode berikut adalah entrypoint script (`thriftscript.py`) yang menggunakan protokol hemat untuk terhubung ke metastore Hive. Perhatikan bahwa `hive.metastore.uris` properti perlu disetel untuk membaca dari metastore Hive eksternal.

```
from os.path import expanduser, join, abspath
from pyspark.sql import SparkSession
from pyspark.sql import Row
# warehouse_location points to the default location for managed databases and tables
warehouse_location = abspath('spark-warehouse')
spark = SparkSession \
    .builder \
    .config("spark.sql.warehouse.dir", warehouse_location) \
    .config("hive.metastore.uris","thrift://thrift-server-host:thift-server-port") \
    .enableHiveSupport() \
    .getOrCreate()
spark.sql("SHOW DATABASES").show()
spark.sql("CREATE EXTERNAL TABLE sampledb.`sparknyctaxi`( `dispatching_base_num`
 string, `pickup_datetime` string, `dropoff_datetime` string, `pulocationid` bigint,
 `dolocationid` bigint, `sr_flag` bigint) STORED AS PARQUET LOCATION 's3://<s3 prefix>/
nyctaxi_parquet/'")
spark.sql("SELECT * FROM sampledb.sparknyctaxi").show()
spark.stop()
```

## Konfigurasikan opsi Hive

### Menggunakan JDBC

Jika Anda ingin menentukan lokasi database Hive eksternal pada instans Amazon RDS My SQL atau Amazon Aurora, Anda dapat mengganti konfigurasi metastore default.

> **Note**
>
> Di Hive, Anda dapat melakukan beberapa penulisan ke tabel metastore secara bersamaan. Jika Anda berbagi informasi metastore antara dua pekerjaan, pastikan Anda tidak menulis ke tabel metastore yang sama secara bersamaan kecuali Anda menulis ke partisi yang berbeda dari tabel metastore yang sama.

Atur konfigurasi berikut dalam `hive-site` klasifikasi untuk mengaktifkan metastore Hive eksternal.

```
{
    "classification": "hive-site",
    "properties": {
        "hive.metastore.client.factory.class":
 "org.apache.hadoop.hive.ql.metadata.SessionHiveMetaStoreClientFactory",
        "javax.jdo.option.ConnectionDriverName": "org.mariadb.jdbc.Driver",
        "javax.jdo.option.ConnectionURL": "jdbc:mysql://db-host:db-port/db-name",
        "javax.jdo.option.ConnectionUserName": "username",
        "javax.jdo.option.ConnectionPassword": "password"
    }
}
```

### Menggunakan server penghematan

Anda dapat mengonfigurasi aplikasi EMR Serverless Hive Anda untuk terhubung ke metastore Hive berdasarkan Amazon untuk RDS My atau SQL Amazon Aurora M. ySQLinstance Untuk melakukan ini, jalankan server barang bekas di node utama cluster Amazon EMR yang ada. Opsi ini sangat ideal jika Anda sudah memiliki EMR cluster Amazon yang menjalankan server barang bekas dan Anda ingin menggunakan konfigurasi pekerjaan Tanpa EMR Server Anda.

Atur konfigurasi berikut dalam `hive-site` klasifikasi sehingga EMR Tanpa Server dapat mengakses metastore penghematan jarak jauh. Perhatikan bahwa Anda harus mengatur `hive.metastore.uris` properti untuk dibaca dari metastore Hive eksternal.

```
{
    "classification": "hive-site",
    "properties": {
        "hive.metastore.client.factory.class":
 "org.apache.hadoop.hive.ql.metadata.SessionHiveMetaStoreClientFactory",
        "hive.metastore.uris": "thrift://thrift-server-host:thirft-server-port"
    }
}
```

### Pertimbangan saat menggunakan metastore eksternal

- Anda dapat mengonfigurasi database yang kompatibel dengan JDBC MariaDB sebagai metastore Anda. Contoh database ini adalah RDS untuk MariaDB, My, SQL dan Amazon Aurora.
- Metastores tidak diinisialisasi secara otomatis. Jika metastore Anda tidak diinisialisasi dengan skema untuk versi Hive Anda, gunakan Hive Schema Tool.
- EMRTanpa server tidak mendukung otentikasi Kerberos. Anda tidak dapat menggunakan server metastore barang bekas dengan otentikasi Kerberos dengan pekerjaan Serverless Spark atau Hive. EMR

## Mengakses data S3 di tempat lain AWS akun dari EMR Serverless

Anda dapat menjalankan pekerjaan Amazon EMR Tanpa Server dari satu AWS akun dan konfigurasikan untuk mengakses data di bucket Amazon S3 milik orang lain AWS akun. Halaman ini menjelaskan cara mengonfigurasi akses lintas akun ke S3 dari EMR Tanpa Server.

Pekerjaan yang berjalan di EMR Tanpa Server dapat menggunakan kebijakan bucket S3 atau peran yang diasumsikan untuk mengakses data di Amazon S3 dari yang lain AWS akun.

### Prasyarat

Untuk mengatur akses lintas akun untuk Amazon EMR Tanpa Server, Anda harus menyelesaikan tugas saat masuk ke dua AWS akun:

- **`AccountA`**- Ini adalah AWS akun tempat Anda membuat aplikasi Amazon EMR Tanpa Server. Sebelum Anda mengatur akses lintas akun, Anda harus memiliki yang berikut ini siap di akun ini:
  - Aplikasi Amazon EMR Tanpa Server tempat Anda ingin menjalankan pekerjaan.

- Peran eksekusi pekerjaan yang memiliki izin yang diperlukan untuk menjalankan pekerjaan dalam aplikasi. Untuk informasi selengkapnya, lihat Peran runtime Job untuk Amazon Serverless EMR.

- **`AccountB`**- Ini adalah AWS akun yang berisi bucket S3 yang Anda inginkan untuk diakses oleh pekerjaan Amazon EMR Tanpa Server Anda.

## Menggunakan kebijakan bucket S3 untuk mengakses data S3 lintas akun

Untuk mengakses bucket S3 di account B From account A, lampirkan kebijakan berikut ke bucket S3 di account B.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "Example permissions 1",
         "Effect": "Allow",
         "Principal": {
            "AWS": "arn:aws:iam::AccountA:root"
         },
         "Action": [
            "s3:ListBucket"
         ],
         "Resource": [
            "arn:aws:s3:::bucket_name_in_AccountB"
         ]
      },
      {
         "Sid": "Example permissions 2",
         "Effect": "Allow",
         "Principal": {
            "AWS": "arn:aws:iam::AccountA:root"
         },
         "Action": [
            "s3:PutObject",
            "s3:GetObject",
            "s3:DeleteObject"
         ],
         "Resource": [
            "arn:aws:s3:::bucket_name_in_AccountB/*"
         ]
```

```
      }
   ]
}
```

Untuk informasi selengkapnya tentang akses lintas akun S3 dengan kebijakan bucket S3, lihat Contoh 2: Pemilik bucket yang memberikan izin bucket lintas akun di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

## Gunakan peran yang diasumsikan untuk mengakses data S3 lintas akun

Cara lain untuk mengatur akses lintas akun untuk Amazon EMR Tanpa Server adalah dengan tindakan dari `AssumeRole` AWS Security Token Service (AWS STS). AWS STS adalah layanan web global yang memungkinkan Anda meminta kredenal hak istimewa terbatas sementara untuk pengguna. Anda dapat melakukan API panggilan ke EMR Tanpa Server dan Amazon S3 dengan kredensil keamanan sementara yang Anda buat. `AssumeRole`

Langkah-langkah berikut menggambarkan cara menggunakan peran yang diasumsikan untuk mengakses data S3 lintas akun dari Tanpa Server: EMR

1. Buat ember Amazon S3, *`cross-account-bucket`*, di`AccountB`. Untuk informasi selengkapnya, lihat Membuat bucket di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon. Jika Anda ingin memiliki akses lintas-akun ke DynamoDB, Anda juga dapat membuat tabel DynamoDB di `AccountB`. Untuk informasi selengkapnya, lihat Membuat tabel DynamoDB di Panduan Pengembang Amazon DynamoDB.
2. Buat `Cross-Account-Role-B` IAM peran `AccountB` yang dapat mengakses *`cross-account-bucket`*.
   a. Masuk ke AWS Management Console dan buka IAM konsol di https://console.aws.amazon.com/iam/.
   b. Pilih Peran dan buat peran baru: `Cross-Account-Role-B`. Untuk informasi selengkapnya tentang cara membuat IAM peran, lihat Membuat IAM peran di Panduan IAM Pengguna.
   c. Buat IAM kebijakan yang menentukan izin `Cross-Account-Role-B` untuk mengakses *`cross-account-bucket`* Bucket S3, seperti yang ditunjukkan oleh pernyataan kebijakan berikut. Kemudian lampirkan IAM kebijakan ke`Cross-Account-Role-B`. Untuk informasi selengkapnya, lihat Membuat IAM kebijakan di Panduan IAM Pengguna.

```
{
"Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Allow",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::cross-account-bucket",
                "arn:aws:s3:::cross-account-bucket/*"
            ]
        }
    ]
}
```

Jika Anda memerlukan akses DynamoDB, buat kebijakan IAM yang menentukan izin untuk mengakses tabel DynamoDB lintas akun. Kemudian lampirkan IAM kebijakan ke`Cross-Account-Role-B`. Untuk informasi selengkapnya, lihat Amazon DynamoDB: Mengizinkan akses ke tabel tertentu di IAM Panduan Pengguna.

Berikut ini adalah kebijakan untuk mengizinkan akses ke tabel DynamoDB. `CrossAccountTable`

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "dynamodb:*",
            "Resource": "arn:aws:dynamodb:MyRegion:AccountB:table/CrossAccountTable"
        }
    ]
}
```

3. Cara mengedit hubungan kepercayaan untuk peran `Cross-Account-Role-B`.
   a. Untuk mengonfigurasi hubungan kepercayaan untuk peran tersebut, pilih tab Trust Relationships di IAM konsol untuk peran `Cross-Account-Role-B` yang Anda buat di Langkah 2.
   b. Pilih Edit Hubungan Kepercayaan.
   c. Tambahkan dokumen kebijakan berikut. Hal ini memungkinkan `Job-Execution-Role-A AccountA` untuk mengambil `Cross-Account-Role-B` peran.

```
{
"Version": "2012-10-17",
  "Statement": [
```

```
    {
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::AccountA:role/Job-Execution-Role-A"
        },
        "Action": "sts:AssumeRole"
    }
  ]
}
```

4. Hibah `Job-Execution-Role-A` `AccountA` di AWS STS `AssumeRole`izin untuk berasumsi`Cross-Account-Role-B`.

   a. Di IAM konsol untuk AWS akun`AccountA`, pilih`Job-Execution-Role-A`.

   b. Tambahkan pernyataan kebijakan berikut pada `Job-Execution-Role-A` untuk mengizinkan tindakan `AssumeRole` di peran `Cross-Account-Role-B`.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::AccountB:role/Cross-Account-Role-B"
        }
    ]
}
```

## Contoh peran yang diasumsikan

Anda dapat menggunakan satu peran yang diasumsikan untuk mengakses semua sumber daya S3 di akun, atau dengan Amazon EMR 6.11 dan yang lebih tinggi, Anda dapat mengonfigurasi beberapa IAM peran untuk diasumsikan saat mengakses bucket S3 lintas akun yang berbeda.

Topik

- Akses sumber daya S3 dengan satu peran yang diasumsikan
- Akses sumber daya S3 dengan beberapa peran yang diasumsikan

## Akses sumber daya S3 dengan satu peran yang diasumsikan

> **Note**
>
> Saat Anda mengonfigurasi pekerjaan untuk menggunakan satu peran yang diasumsikan, semua sumber daya S3 di seluruh pekerjaan menggunakan peran tersebut, termasuk `entryPoint` skrip.

Jika Anda ingin menggunakan satu peran yang diasumsikan untuk mengakses semua sumber daya S3 di akun B, tentukan konfigurasi berikut:

1. Tentukan EMRFS konfigurasi `fs.s3.customAWSCredentialsProvider` kespark.hadoop.fs.s3.customAWSCredentialsProvider=com.amazonaws.emr.AssumeRole/
2. Untuk Spark, gunakan `spark.emr-serverless.driverEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN` dan `spark.executorEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN` tentukan variabel lingkungan pada driver dan pelaksana.
3. Untuk Hive, gunakan`hive.emr-serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARN`,`tez.am.emr-serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARN`, dan `tez.task.emr-serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARN` untuk menentukan variabel lingkungan pada driver Hive, master aplikasi Tez, dan wadah tugas Tez.

Contoh berikut menunjukkan cara menggunakan peran yang diasumsikan untuk memulai pekerjaan EMR Tanpa Server dengan akses lintas akun.

Spark

Contoh berikut menunjukkan cara menggunakan peran yang diasumsikan untuk memulai pekerjaan Spark EMR Tanpa Server dengan akses lintas akun ke S3.

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "sparkSubmit": {
            "entryPoint": "entrypoint_location",
```

```
            "entryPointArguments": [":argument_1:", ":argument_2:"],
            "sparkSubmitParameters": "--conf spark.executor.cores=4 --conf
 spark.executor.memory=20g --conf spark.driver.cores=4 --conf spark.driver.memory=8g
 --conf spark.executor.instances=1"
        }
    }' \
     --configuration-overrides '{
        "applicationConfiguration": [{
            "classification": "spark-defaults",
            "properties": {
                "spark.hadoop.fs.s3.customAWSCredentialsProvider":
 "spark.hadoop.fs.s3.customAWSCredentialsProvider=com.amazonaws.emr.AssumeRoleAWSCredentials
                "spark.emr-serverless.driverEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN":
 "arn:aws:iam::AccountB:role/Cross-Account-Role-B",
                "spark.executorEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN":
 "arn:aws:iam::AccountB:role/Cross-Account-Role-B"
            }
        }]
    }'
```

Hive

Contoh berikut menunjukkan cara menggunakan peran yang diasumsikan untuk memulai pekerjaan Sarang EMR Tanpa Server dengan akses lintas akun ke S3.

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "hive": {
            "query": "query_location",
            "parameters": "hive_parameters"
        }
    }' \
    --configuration-overrides '{
        "applicationConfiguration": [{
            "classification": "hive-site",
            "properties": {
                "fs.s3.customAWSCredentialsProvider":
 "com.amazonaws.emr.serverless.credentialsprovider.AssumeRoleAWSCredentialsProvider",
                "hive.emr-serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARN":
 "arn:aws:iam::AccountB:role/Cross-Account-Role-B",
```

```
                "tez.am.emr-serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARN":
 "arn:aws:iam::AccountB:role/Cross-Account-Role-B",
                "tez.task.emr-
serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARN":
 "arn:aws:iam::AccountB:role/Cross-Account-Role-B"
            }
        }]
    }'
```

## Akses sumber daya S3 dengan beberapa peran yang diasumsikan

Dengan rilis EMR Tanpa Server 6.11.0 dan yang lebih tinggi, Anda dapat mengonfigurasi beberapa IAM peran untuk diasumsikan saat mengakses bucket lintas akun yang berbeda. Jika Anda ingin mengakses sumber daya S3 yang berbeda dengan peran yang diasumsikan berbeda di akun B, gunakan konfigurasi berikut saat Anda memulai pekerjaan:

1. Tentukan EMRFS konfigurasi `fs.s3.customAWSCredentialsProvider` `kecom.amazonaws.emr.serverless.credentialsprovider.BucketLevelAssumeRoleCreden`
2. Tentukan EMRFS konfigurasi `fs.s3.bucketLevelAssumeRoleMapping` untuk menentukan pemetaan dari nama bucket S3 ke IAM peran di akun B untuk diasumsikan. Nilai harus dalam format`bucket1->role1;bucket2->role2`.

Misalnya, Anda dapat menggunakan `arn:aws:iam::AccountB:role/Cross-Account-Role-B-1` untuk mengakses bucket`bucket1`, dan menggunakannya `arn:aws:iam::AccountB:role/Cross-Account-Role-B-2` untuk mengakses bucket`bucket2`. Contoh berikut menunjukkan cara memulai pekerjaan EMR Tanpa Server dengan akses lintas akun melalui beberapa peran yang diasumsikan.

Spark

Contoh berikut menunjukkan cara menggunakan beberapa peran yang diasumsikan untuk membuat pekerjaan Spark EMR Tanpa Server.

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "sparkSubmit": {
            "entryPoint": "entrypoint_location",
```

```
            "entryPointArguments": [":argument_1:", ":argument_2:"],
            "sparkSubmitParameters": "--conf spark.executor.cores=4 --conf
 spark.executor.memory=20g --conf spark.driver.cores=4 --conf spark.driver.memory=8g
 --conf spark.executor.instances=1"
        }
    }' \
     --configuration-overrides '{
        "applicationConfiguration": [{
            "classification": "spark-defaults",
            "properties": {
                "spark.hadoop.fs.s3.customAWSCredentialsProvider":
 "com.amazonaws.emr.serverless.credentialsprovider.BucketLevelAssumeRoleCredentialsProvider"
                "spark.hadoop.fs.s3.bucketLevelAssumeRoleMapping":
 "bucket1->arn:aws:iam::AccountB:role/Cross-Account-Role-B-1;bucket2-
>arn:aws:iam::AccountB:role/Cross-Account-Role-B-2"
            }
        }]
    }'
```

Hive

Contoh berikut menunjukkan cara menggunakan beberapa peran yang diasumsikan untuk membuat pekerjaan Sarang EMR Tanpa Server.

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "hive": {
            "query": "query_location",
            "parameters": "hive_parameters"
        }
    }' \
    --configuration-overrides '{
        "applicationConfiguration": [{
            "classification": "hive-site",
            "properties": {
                "fs.s3.customAWSCredentialsProvider":
 "com.amazonaws.emr.serverless.credentialsprovider.AssumeRoleAWSCredentialsProvider",
                "fs.s3.bucketLevelAssumeRoleMapping": "bucket1-
>arn:aws:iam::AccountB:role/Cross-Account-Role-B-1;bucket2-
>arn:aws:iam::AccountB:role/Cross-Account-Role-B-2"
            }
```

```
        }]
    }'
```

# Memecahkan masalah kesalahan di Tanpa Server EMR

Gunakan informasi berikut untuk membantu mendiagnosis dan memperbaiki masalah umum yang mungkin Anda temui saat bekerja dengan Amazon Tanpa EMR Server.

Topik



## Kesalahan: Batas terlampaui untuk kapasitas maksimum yang diizinkan.

Kesalahan ini menunjukkan bahwa EMR Tanpa Server tidak dapat mengirimkan pekerjaan karena aplikasi telah melampaui batas kapasitas maksimum yang dikonfigurasi. Tingkatkan batas kapasitas maksimum untuk aplikasi.

## Kesalahan: Kapasitas maksimum yang dikonfigurasi telah terlampaui. Silakan di coba lagi nanti.

Kesalahan ini menunjukkan bahwa EMR Tanpa Server tidak dapat memulai pekerjaan baru karena aplikasi telah melampaui batas kapasitas maksimum yang dikonfigurasi. Tingkatkan batas kapasitas maksimum untuk aplikasi.

## Kesalahan: Akses S3 ditolak. Silakan periksa izin akses S3 dari peran runtime pekerjaan pada sumber daya S3 yang diperlukan.

Kesalahan ini menunjukkan bahwa pekerjaan Anda tidak memiliki akses ke sumber daya S3 Anda. Verifikasi bahwa peran runtime pekerjaan memiliki izin untuk mengakses sumber daya S3 yang perlu

digunakan pekerjaan. Untuk mempelajari lebih lanjut tentang peran runtime, lihatPeran runtime Job untuk Amazon Serverless EMR.

## Kesalahan: ModuleNotFoundError: Tidak ada modul bernama<module>. Silakan merujuk ke panduan pengguna tentang cara menggunakan pustaka python dengan EMR Tanpa Server.

Kesalahan ini menunjukkan bahwa modul Python tidak tersedia untuk pekerjaan Spark. Periksa apakah pustaka Python dependen tersedia untuk pekerjaan itu. Untuk informasi selengkapnya tentang cara mengemas pustaka Python, lihat. Menggunakan pustaka Python dengan Tanpa Server EMR

## Kesalahan: Tidak dapat mengambil peran eksekusi <role name>karena tidak ada atau tidak diatur dengan hubungan kepercayaan yang diperlukan.

Kesalahan ini menunjukkan bahwa peran runtime pekerjaan yang Anda tentukan untuk pekerjaan tidak ada, atau bahwa peran tersebut tidak memiliki hubungan kepercayaan untuk izin Tanpa EMR Server. Untuk memverifikasi bahwa IAM peran tersebut ada dan memvalidasi bahwa Anda telah menyiapkan kebijakan kepercayaan peran dengan benar, lihat petunjuknya. Peran runtime Job untuk Amazon Serverless EMR

# Jalankan beban kerja interaktif dengan Tanpa EMR Server melalui Studio EMR

## Gambaran Umum

Aplikasi interaktif adalah aplikasi EMR Tanpa Server yang memiliki kemampuan interaktif diaktifkan. Dengan aplikasi interaktif Amazon EMR Serverless, Anda dapat menjalankan beban kerja interaktif dengan notebook Jupyter yang dikelola di Amazon Studio. EMR Ini membantu insinyur data, ilmuwan data, dan analis data menggunakan EMR Studio untuk menjalankan analitik interaktif dengan kumpulan data di penyimpanan data seperti Amazon S3 dan Amazon DynamoDB.

Kasus penggunaan untuk aplikasi interaktif di EMR Tanpa Server meliputi yang berikut:

- Insinyur data menggunakan IDE pengalaman di EMR Studio untuk membuat ETL skrip. Skrip menyerap data dari lokal, mengubah data untuk analisis, dan menyimpan data di Amazon S3.
- Ilmuwan data menggunakan notebook untuk mengeksplorasi kumpulan data dan melatih model pembelajaran mesin (ML) untuk mendeteksi anomali dalam kumpulan data.
- Analis data mengeksplorasi kumpulan data dan membuat skrip yang menghasilkan laporan harian untuk memperbarui aplikasi seperti dasbor bisnis.

## Prasyarat

Untuk menggunakan beban kerja interaktif dengan EMR Tanpa Server, Anda harus memenuhi persyaratan berikut:

- EMRAplikasi interaktif tanpa server didukung dengan Amazon EMR 6.14.0 dan yang lebih tinggi.
- Untuk mengakses aplikasi interaktif Anda, jalankan beban kerja yang Anda kirimkan, dan jalankan buku catatan interaktif dari EMR Studio, Anda memerlukan izin dan peran tertentu. Untuk informasi selengkapnya, lihat Izin yang diperlukan untuk beban kerja interaktif.

## Izin yang diperlukan untuk beban kerja interaktif

Selain izin dasar yang diperlukan untuk mengakses EMR Tanpa Server, Anda harus mengonfigurasi izin tambahan untuk identitas atau peran AndaIAM:

Untuk mengakses aplikasi interaktif Anda

Siapkan izin pengguna dan Ruang Kerja untuk EMR Studio. Untuk informasi selengkapnya, lihat Mengonfigurasi izin pengguna EMR Studio di Panduan EMR Manajemen Amazon.

Untuk menjalankan beban kerja yang Anda kirimkan dengan Tanpa Server EMR

Siapkan peran runtime pekerjaan. Untuk informasi selengkapnya, lihat Buat peran runtime pekerjaan.

Untuk menjalankan notebook interaktif dari Studio EMR

Tambahkan izin tambahan berikut ke IAM kebijakan untuk pengguna Studio:

- **`emr-serverless:AccessInteractiveEndpoints`**- Memberikan izin untuk mengakses dan terhubung ke aplikasi interaktif yang Anda tentukan sebagai`Resource`. Izin ini diperlukan untuk melampirkan ke aplikasi EMR Tanpa Server dari Ruang Kerja EMR Studio.
- **`iam:PassRole`**- Memberikan izin untuk mengakses peran IAM eksekusi yang Anda rencanakan untuk digunakan saat Anda melampirkan ke aplikasi. `PassRole`Izin yang sesuai diperlukan untuk melampirkan ke aplikasi EMR Tanpa Server dari Ruang Kerja EMR Studio.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EMRServerlessInteractiveAccess",
            "Effect": "Allow",
            "Action": "emr-serverless:AccessInteractiveEndpoints",
            "Resource": "arn:aws:emr-serverless:Region:account:/applications/*"
        },
        {
            "Sid": "EMRServerlessRuntimeRoleAccess",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "interactive-execution-role-ARN",
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": "emr-serverless.amazonaws.com"
                }
            }
        }
    ]
}
```

# Mengkonfigurasi aplikasi interaktif

Gunakan langkah-langkah tingkat tinggi berikut untuk membuat aplikasi EMR Tanpa Server dengan kemampuan interaktif dari Amazon Studio di EMR AWS Management Console.

1. Ikuti langkah-langkah Memulai dengan Amazon Tanpa EMR Server untuk membuat aplikasi.
2. Kemudian, luncurkan ruang kerja dari EMR Studio dan lampirkan ke aplikasi EMR Tanpa Server sebagai opsi komputasi. Untuk informasi selengkapnya, lihat tab Beban kerja interaktif di Langkah 2 dari dokumentasi Memulai EMR Tanpa Server.

Saat Anda melampirkan aplikasi ke Studio Workspace, aplikasi mulai terpicu secara otomatis jika aplikasi tersebut belum berjalan. Anda juga dapat memulai aplikasi terlebih dahulu dan menyiapkannya sebelum Anda melampirkannya ke Workspace.

# Pertimbangan dengan aplikasi interaktif

- EMRAplikasi interaktif tanpa server didukung dengan Amazon EMR 6.14.0 dan yang lebih tinggi.
- EMRStudio adalah satu-satunya klien yang terintegrasi dengan aplikasi interaktif EMR Tanpa Server. Kemampuan EMR Studio berikut tidak didukung dengan aplikasi interaktif EMR Tanpa Server: Kolaborasi ruang kerja, SQL Explorer, dan eksekusi terprogram notebook.
- Aplikasi interaktif hanya didukung untuk mesin Spark.
- Aplikasi interaktif mendukung kernel Python 3, PySpark dan Spark Scala.
- Anda dapat menjalankan hingga 25 notebook bersamaan pada satu aplikasi interaktif.
- Tidak ada titik akhir atau API antarmuka yang mendukung notebook Jupyter yang dihosting sendiri dengan aplikasi interaktif.
- Untuk pengalaman startup yang dioptimalkan, kami menyarankan Anda mengonfigurasi kapasitas pra-inisialisasi untuk driver dan pelaksana, dan Anda memulai aplikasi terlebih dahulu. Ketika Anda memulai aplikasi terlebih dahulu, Anda memastikan bahwa itu siap ketika Anda ingin melampirkannya ke Workspace Anda.

  ```
  aws emr-serverless start-application \
  --application-id your-application-id
  ```

- Secara default, `autoStopConfig` diaktifkan untuk aplikasi. Ini mematikan aplikasi setelah 30 menit waktu idle. Anda dapat mengubah konfigurasi ini sebagai bagian dari `update-application` permintaan `create-application` atau permintaan Anda.

- Saat menggunakan aplikasi interaktif, kami menyarankan Anda mengonfigurasi kapasitas kernel, driver, dan pelaksana pra-intialisasi untuk menjalankan notebook Anda. Setiap sesi interaktif Spark memerlukan satu kernel dan satu driver, sehingga EMR Tanpa Server mempertahankan pekerja kernel yang telah diinisialisasi sebelumnya untuk setiap driver yang telah diinisialisasi sebelumnya. Secara default, EMR Tanpa Server mempertahankan kapasitas pra-inisialisasi dari satu pekerja kernel di seluruh aplikasi bahkan jika Anda tidak menentukan kapasitas pra-inisialisasi untuk driver. Setiap pekerja kernel menggunakan memori 4 v CPU dan 16 GB. Untuk informasi harga saat ini, lihat halaman EMRHarga Amazon.
- Anda harus memiliki kuota CPU layanan v yang memadai di Akun AWS untuk menjalankan beban kerja interaktif. Jika Anda tidak menjalankan beban kerja yang mendukung Lake Formation, kami sarankan setidaknya 24 v. CPU Jika Anda melakukannya, kami sarankan setidaknya 28 vCPU.
- EMRTanpa server secara otomatis menghentikan kernel dari notebook jika mereka telah menganggur selama lebih dari 60 menit. EMRTanpa server menghitung waktu idle kernel dari aktivitas terakhir yang diselesaikan selama sesi notebook. Saat ini Anda tidak dapat mengubah pengaturan batas waktu idle kernel.
- Untuk mengaktifkan Lake Formation dengan beban kerja interaktif, atur konfigurasi `spark.emr-serverless.lakeformation.enabled` ke `true` bawah `spark-defaults` klasifikasi dalam `runtime-configuration` objek saat Anda membuat aplikasi Tanpa EMR Server. Untuk mempelajari lebih lanjut tentang mengaktifkan Lake Formation di EMR Tanpa Server, lihat Mengaktifkan Lake Formation di Amazon. EMR

# Jalankan beban kerja interaktif dengan EMR Tanpa Server melalui titik akhir Apache Livy

Dengan Amazon EMR merilis 6.14.0 dan yang lebih tinggi, Anda dapat membuat dan mengaktifkan titik akhir Apache Livy sambil membuat aplikasi EMR Tanpa Server dan menjalankan beban kerja interaktif melalui notebook yang dihosting sendiri atau dengan klien khusus. Endpoint Apache Livy menawarkan manfaat berikut:

- Anda dapat terhubung dengan aman ke titik akhir Apache Livy melalui notebook Jupyter dan mengelola beban kerja Apache Spark dengan antarmuka Apache Livy. REST
- Gunakan REST API operasi Apache Livy untuk aplikasi web interaktif yang menggunakan data dari beban kerja Apache Spark.

## Prasyarat

Untuk menggunakan endpoint Apache Livy dengan EMR Tanpa Server, Anda harus memenuhi persyaratan berikut:

- Selesaikan langkah-langkah dalam Memulai dengan Amazon Tanpa EMR Server.
- Untuk menjalankan beban kerja interaktif melalui titik akhir Apache Livy, Anda memerlukan izin dan peran tertentu. Untuk informasi selengkapnya, lihat Izin yang diperlukan untuk beban kerja interaktif.

## Izin yang diperlukan

Selain izin yang diperlukan untuk mengakses EMR Tanpa Server, Anda juga harus menambahkan izin berikut ke IAM peran Anda untuk mengakses titik akhir Apache Livy dan menjalankan aplikasi:

- `emr-serverless:AccessLivyEndpoints`— memberikan izin untuk mengakses dan terhubung ke aplikasi berkemampuan Livy yang Anda tentukan sebagai. `Resource` Anda memerlukan izin ini untuk menjalankan REST API operasi yang tersedia dari titik akhir Apache Livy.
- `iam:PassRole`— memberikan izin untuk mengakses peran IAM eksekusi saat membuat sesi Apache Livy. EMRTanpa server akan menggunakan peran ini untuk menjalankan beban kerja Anda.
- `emr-serverless:GetDashboardForJobRun`— memberikan izin untuk menghasilkan UI Spark Live dan tautan log driver dan menyediakan akses ke log sebagai bagian dari hasil sesi Apache Livy.

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Sid": "EMRServerlessInteractiveAccess",
            "Effect": "Allow",
            "Action": "emr-serverless:AccessLivyEndpoints",
            "Resource": "arn:aws:emr-serverless:<AWS_REGION>:account:/applications/*"
        },
        {
            "Sid": "EMRServerlessRuntimeRoleAccess",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "execution-role-ARN",
```

```
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": "emr-serverless.amazonaws.com"
                }
            }
        },
        {
            "Sid": "EMRServerlessDashboardAccess",
            "Effect": "Allow",
            "Action": "emr-serverless:GetDashboardForJobRun",
            "Resource": "arn:aws:emr-serverless:<AWS_REGION>:account:/applications/*"
        }
    ]
}
```

## Memulai

1. Untuk membuat aplikasi Apache Livy-enabled, jalankan perintah berikut.

```
 aws emr-serverless create-application \
--name my-application-name \
--type 'application-type' \
 --release-label <Amazon EMR-release-version>
--interactive-configuration '{"livyEndpointEnabled": true}'
```

2. Setelah EMR Serverless membuat aplikasi Anda, mulai aplikasi untuk membuat endpoint Apache Livy tersedia.

```
 aws emr-serverless start-application \
 --application-id application-id
```

   Gunakan perintah berikut untuk memeriksa apakah status aplikasi Anda. Setelah status menjadi`STARTED`, Anda dapat mengakses titik akhir Apache Livy.

```
aws emr-serverless get-application \
--region <AWS_REGION> --application-id >application_id>
```

3. Gunakan yang berikut ini URL untuk mengakses titik akhir:

```
https://_<application-id>_.livy.emr-serverless-
services._<AWS_REGION>_.amazonaws.com
```

Setelah titik akhir siap, Anda dapat mengirimkan beban kerja berdasarkan kasus penggunaan Anda. Anda harus menandatangani setiap permintaan ke titik akhir dengan SIGv4protokol dan meneruskan header otorisasi. Anda dapat menggunakan metode berikut untuk menjalankan beban kerja:

- HTTPklien — Anda harus mengirimkan API operasi endpoint Apache Livy Anda dengan klien khusus. HTTP
- Kernel Sparkmagic — Anda harus menjalankan kernel sparkmagic secara lokal dan mengirimkan kueri interaktif dengan notebook Jupyter.

## HTTPklien

Untuk membuat sesi Apache Livy, Anda harus mengirimkan `emr-serverless.session.executionRoleArn` `conf` parameter badan permintaan Anda. Contoh berikut adalah `POST /sessions` permintaan sampel.

```
{
    "kind": "pyspark",
    "heartbeatTimeoutInSecond": 60,
    "conf": {
        "emr-serverless.session.executionRoleArn": "<executionRoleArn>"
    }
}
```

Tabel berikut menjelaskan semua operasi Apache Livy API yang tersedia.

| APIoperasi | Deskripsi |
| --- | --- |
| GET/sesi | Mengembalikan daftar semua sesi interaktif aktif. |
| POST/sesi | Membuat sesi interaktif baru melalui spark atau pyspark. |
| GET/sesi/ <*sessionId* > | Mengembalikan informasi sesi. |
| GET/sesi/ <*sessionId* >/negara | Mengembalikan keadaan sesi. |
| DELETE/sesi/ <*sessionId* > | Menghentikan dan menghapus sesi. |

| APIoperasi | Deskripsi |
| --- | --- |
| GET/sesi/ <*sessionId* >/pernyataan | Mengembalikan semua pernyataan dalam sesi. |
| POST/sesi/ <*sessionId* >/pernyataan | Menjalankan pernyataan dalam sesi. |
| GET/sesi/ <*sessionId* >/pernyataan/ <*statementId* > | Mengembalikan rincian pernyataan yang ditentukan dalam sesi. |
| POST/sesi/ <*sessionId* >/pernyataan/ <*statementId* >/batalkan | Membatalkan pernyataan yang ditentukan dalam sesi ini. |

Mengirim permintaan ke titik akhir Apache Livy

Anda juga dapat mengirim permintaan langsung ke titik akhir Apache Livy dari klien. HTTP Melakukannya memungkinkan Anda menjalankan kode dari jarak jauh untuk kasus penggunaan di luar buku catatan.

Sebelum Anda dapat mulai mengirim permintaan ke titik akhir, pastikan Anda telah menginstal pustaka berikut:

```
pip3 install botocore awscrt requests
```

Berikut ini adalah contoh skrip Python untuk mengirim HTTP permintaan langsung ke titik akhir:

```
from botocore import crt
import requests
from botocore.awsrequest import AWSRequest
from botocore.credentials import Credentials
import botocore.session
import json, pprint, textwrap

endpoint = 'https://<application_id>.livy.emr-serverless-
services-<AWS_REGION>.amazonaws.com'
headers = {'Content-Type': 'application/json'}

session = botocore.session.Session()
signer = crt.auth.CrtS3SigV4Auth(session.get_credentials(), 'emr-serverless',
 '<AWS_REGION>')
```

```
### Create session request

data = {'kind': 'pyspark', 'heartbeatTimeoutInSecond': 60, 'conf': { 'emr-
serverless.session.executionRoleArn': 'arn:aws:iam::123456789012:role/role1'}}

request = AWSRequest(method='POST', url=endpoint + "/sessions", data=json.dumps(data),
 headers=headers)

request.context["payload_signing_enabled"] = False

signer.add_auth(request)

prepped = request.prepare()

r = requests.post(prepped.url, headers=prepped.headers, data=json.dumps(data))

pprint.pprint(r.json())


### List Sessions Request

request = AWSRequest(method='GET', url=endpoint + "/sessions", headers=headers)

request.context["payload_signing_enabled"] = False

signer.add_auth(request)

prepped = request.prepare()

r2 = requests.get(prepped.url, headers=prepped.headers)
pprint.pprint(r2.json())


### Get session state

session_url = endpoint + r.headers['location']

request = AWSRequest(method='GET', url=session_url, headers=headers)

request.context["payload_signing_enabled"] = False

signer.add_auth(request)

prepped = request.prepare()
```

```
r3 = requests.get(prepped.url, headers=prepped.headers)

pprint.pprint(r3.json())


### Submit Statement

data = {
      'code': "1 + 1"
}

statements_url = endpoint + r.headers['location'] + "/statements"

request = AWSRequest(method='POST', url=statements_url, data=json.dumps(data),
 headers=headers)

request.context["payload_signing_enabled"] = False

signer.add_auth(request)

prepped = request.prepare()

r4 = requests.post(prepped.url, headers=prepped.headers, data=json.dumps(data))

pprint.pprint(r4.json())

### Check statements results

specific_statement_url = endpoint + r4.headers['location']

request = AWSRequest(method='GET', url=specific_statement_url, headers=headers)

request.context["payload_signing_enabled"] = False

signer.add_auth(request)

prepped = request.prepare()

r5 = requests.get(prepped.url, headers=prepped.headers)

pprint.pprint(r5.json())
```

```
### Delete session


session_url = endpoint + r.headers['location']

request = AWSRequest(method='DELETE', url=session_url, headers=headers)

request.context["payload_signing_enabled"] = False

signer.add_auth(request)

prepped = request.prepare()

r6 = requests.delete(prepped.url, headers=prepped.headers)

pprint.pprint(r6.json())
```

## Kernel Sparkmagic

Sebelum Anda menginstal sparkmagic, pastikan Anda telah mengonfigurasi AWS kredensyal dalam contoh di mana Anda ingin menginstal sparkmagic

1. Instal sparkmagic dengan mengikuti langkah-langkah instalasi. Perhatikan bahwa Anda hanya perlu melakukan empat langkah pertama.
2. Kernel sparkmagic mendukung autentikator khusus, sehingga Anda dapat mengintegrasikan autentikator dengan kernel sparkmagic sehingga setiap permintaan ditandatangani. SIGv4
3. Instal EMR autentikator kustom Tanpa Server.

   ```
   pip install emr-serverless-customauth
   ```

4. Sekarang berikan jalur ke autentikator khusus dan titik akhir Apache Livy URL di file json konfigurasi sparkmagic. Gunakan perintah berikut untuk membuka file konfigurasi.

   ```
   vim ~/.sparkmagic/config.json
   ```

   Berikut ini adalah `config.json` file sampel.

   ```
   {
   "kernel_python_credentials" : {
       "username": "",
   ```

```
    "password": "",
    "url": "https://<application-id>.livy.emr-serverless-
services.<AWS_REGION>.amazonaws.com",
    "auth": "Custom_Auth"
  },

  "kernel_scala_credentials" : {
    "username": "",
    "password": "",
    "url": "https://<application-id>.livy.emr-serverless-
services.<AWS_REGION>.amazonaws.com",
    "auth": "Custom_Auth"
  },
  "authenticators": {
    "None": "sparkmagic.auth.customauth.Authenticator",
    "Basic_Access": "sparkmagic.auth.basic.Basic",
    "Custom_Auth":
 "emr_serverless_customauth.customauthenticator.EMRServerlessCustomSigV4Signer"
  },
  "livy_session_startup_timeout_seconds": 600,
  "ignore_ssl_errors": false
}
```

5. Mulai lab Jupyter. Ini harus menggunakan otentikasi khusus yang Anda atur pada langkah terakhir.
6. Anda kemudian dapat menjalankan perintah notebook berikut dan kode Anda untuk memulai.

```
%%info //Returns the information about the current sessions.
```

```
%%configure -f //Configure information specific to a session. We supply
 executionRoleArn in this example. Change it for your use case.
{
    "driverMemory": "4g",
    "conf": {
          "emr-serverless.session.executionRoleArn":
 "arn:aws:iam::123456789012:role/JobExecutionRole"
    }
}
```

```
<your code>//Run your code to start the session
```

Secara internal, setiap instruksi memanggil setiap API operasi Apache Livy melalui titik akhir Apache Livy yang dikonfigurasi. URL Anda kemudian dapat menulis instruksi Anda sesuai dengan kasus penggunaan Anda.

## Pertimbangan

Pertimbangkan pertimbangan berikut saat menjalankan beban kerja interaktif melalui titik akhir Apache Livy.

- EMRTanpa server mempertahankan isolasi tingkat sesi menggunakan prinsipal pemanggil. Prinsipal penelepon yang membuat sesi adalah satu-satunya yang dapat mengakses sesi itu. Untuk isolasi yang lebih terperinci, Anda dapat mengonfigurasi identitas sumber saat Anda mengasumsikan kredensyal. Dalam hal ini, EMR Tanpa Server memberlakukan isolasi tingkat sesi berdasarkan prinsip penelepon dan identitas sumber. Untuk informasi selengkapnya tentang identitas sumber, lihat Memantau dan mengontrol tindakan yang diambil dengan peran yang diasumsikan.
- Endpoint Apache Livy didukung dengan rilis EMR Tanpa Server 6.14.0 dan yang lebih tinggi.
- Endpoint Apache Livy hanya didukung untuk mesin Apache Spark.
- Titik akhir Apache Livy mendukung Scala Spark dan. PySpark
- Secara default, `autoStopConfig` diaktifkan di aplikasi Anda. Ini berarti bahwa aplikasi dimatikan setelah 15 menit menganggur. Anda dapat mengubah konfigurasi ini sebagai bagian dari `update-application` permintaan `create-application` atau permintaan Anda.
- Anda dapat menjalankan hingga 25 sesi bersamaan pada satu aplikasi berkemampuan endpoint Apache Livy.
- Untuk pengalaman startup terbaik, kami menyarankan Anda mengonfigurasi kapasitas pra-inisialisasi untuk driver dan pelaksana.
- Anda harus memulai aplikasi secara manual sebelum menghubungkan ke titik akhir Apache Livy.
- Anda harus memiliki kuota CPU layanan v yang memadai di Akun AWS untuk menjalankan beban kerja interaktif dengan endpoint Apache Livy. Kami merekomendasikan setidaknya 24 vCPU.
- Batas waktu sesi Apache Livy default adalah 1 jam. Jika Anda tidak menjalankan pernyataan satu jam, maka Apache Livy menghapus sesi dan melepaskan driver dan pelaksana. Anda tidak dapat mengubah konfigurasi ini.
- Hanya sesi aktif yang dapat berinteraksi dengan titik akhir Apache Livy. Setelah sesi selesai, membatalkan, atau berakhir, Anda tidak dapat mengaksesnya melalui titik akhir Apache Livy.

# Pencatatan dan pemantauan

Pemantauan adalah bagian penting dari menjaga keandalan, ketersediaan, dan kinerja aplikasi dan EMR pekerjaan Tanpa Server. Anda harus mengumpulkan data pemantauan dari semua bagian solusi EMR Tanpa Server Anda sehingga Anda dapat lebih mudah men-debug kegagalan multipoint jika terjadi.

Topik



## Menyimpan log

Untuk memantau kemajuan pekerjaan Anda di EMR Tanpa Server dan memecahkan masalah kegagalan pekerjaan, Anda dapat memilih cara EMR Tanpa Server menyimpan dan menyajikan log aplikasi. Saat mengirimkan pekerjaan, Anda dapat menentukan penyimpanan terkelola, Amazon S3, dan Amazon CloudWatch sebagai opsi pencatatan.

Dengan CloudWatch, Anda dapat menentukan jenis log dan lokasi log yang ingin Anda gunakan, atau menerima jenis dan lokasi default. Untuk informasi lebih lanjut tentang CloudWatch log, lihatthe section called “Amazon CloudWatch”. Dengan penyimpanan terkelola dan pencatatan S3, tabel berikut menunjukkan lokasi log dan ketersediaan UI yang dapat Anda harapkan jika Anda memilih penyimpanan terkelola, bucket Amazon S3, atau keduanya.

| Opsi | Log peristiwa | Log Kontainer | UI Aplikasi |
| --- | --- | --- | --- |
| Penyimpanan terkelola | Disimpan dalam penyimpanan terkelola | Disimpan dalam penyimpanan terkelola | Didukung |

| Opsi | Log peristiwa | Log Kontainer | UI Aplikasi |
| --- | --- | --- | --- |
| Penyimpanan terkelola dan bucket S3 | Disimpan di kedua tempat | Disimpan dalam ember S3 | Didukung |
| Bucket Amazon S3 | Disimpan dalam ember S3 | Disimpan dalam ember S3 | Tidak didukung [1] |

[1] Kami menyarankan agar Anda tetap memilih opsi Penyimpanan Terkelola. Jika tidak, Anda tidak dapat menggunakan aplikasi bawaanUIs.

## Logging untuk EMR Tanpa Server dengan penyimpanan terkelola

Secara default, EMR Tanpa Server menyimpan log aplikasi dengan aman di penyimpanan EMR terkelola Amazon selama maksimal 30 hari.

> **Note**
>
> Jika Anda mematikan opsi default, Amazon tidak EMR dapat memecahkan masalah pekerjaan Anda atas nama Anda.

Untuk mematikan opsi ini dari EMR Studio, batalkan pilihan Izinkan  AWS untuk menyimpan log selama 30 hari kotak  centang di bagian Pengaturan tambahan pada halaman Kirim pekerjaan.

Untuk mematikan opsi ini dari AWS CLI, gunakan `managedPersistenceMonitoringConfiguration` konfigurasi saat Anda mengirimkan pekerjaan.

```
{
    "monitoringConfiguration": {
        "managedPersistenceMonitoringConfiguration": {
            "enabled": false
        }
    }
}
```

# Logging untuk EMR Tanpa Server dengan bucket Amazon S3

Sebelum pekerjaan Anda dapat mengirim data log ke Amazon S3, Anda harus menyertakan izin berikut dalam kebijakan izin untuk peran runtime pekerjaan. Ganti *DOC-EXAMPLE-BUCKET-LOGGING* dengan nama bucket logging Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET-LOGGING/*"
            ]
        }
    ]
}
```

Untuk menyiapkan bucket Amazon S3 untuk menyimpan log dari AWS CLI, gunakan `s3MonitoringConfiguration` konfigurasi saat Anda memulai pekerjaan. Untuk melakukan ini, berikan yang berikut `--configuration-overrides` dalam konfigurasi.

```
{
    "monitoringConfiguration": {
        "s3MonitoringConfiguration": {
            "logUri": "s3://DOC-EXAMPLE-BUCKET-LOGGING/logs/"
        }
    }
}
```

Untuk pekerjaan batch yang tidak mengaktifkan percobaan ulang, EMR Serverless mengirimkan log ke jalur berikut:

```
'/applications/<applicationId>/jobs/<jobId>'
```

EMRRilis tanpa server 7.1.0 dan mendukung upaya coba lagi yang lebih tinggi untuk pekerjaan streaming dan pekerjaan batch. Jika Anda menjalankan pekerjaan dengan percobaan ulang

diaktifkan, EMR Tanpa Server secara otomatis menambahkan nomor percobaan ke awalan jalur log, sehingga Anda dapat membedakan dan melacak log dengan lebih baik.

```
'/applications/<applicationId>/jobs/<jobId>/attempts/<attemptNumber>/'
```

## Logging untuk EMR Tanpa Server dengan Amazon CloudWatch

Saat Anda mengirimkan pekerjaan ke aplikasi EMR Tanpa Server, Anda dapat memilih Amazon CloudWatch sebagai opsi untuk menyimpan log aplikasi Anda. Ini memungkinkan Anda untuk menggunakan fitur analisis CloudWatch log seperti Wawasan CloudWatch Log dan Live Tail. Anda juga dapat melakukan streaming log dari CloudWatch sistem lain seperti OpenSearch untuk analisis lebih lanjut.

EMRTanpa server menyediakan pencatatan waktu nyata untuk log driver. Anda dapat melihat log secara real time dengan kemampuan CloudWatch live tail, atau melalui perintah CloudWatch CLI ekor.

Secara default, CloudWatch logging dinonaktifkan untuk Tanpa EMR Server. Untuk mengaktifkannya, lihat konfigurasi diAWS CLI.

> **Note**
>
> Amazon CloudWatch menerbitkan log secara real time, sehingga menghasilkan lebih banyak sumber daya dari pekerja. Jika Anda memilih kapasitas pekerja yang rendah, dampaknya terhadap waktu kerja Anda mungkin meningkat. Jika Anda mengaktifkan CloudWatch pencatatan, kami sarankan Anda memilih kapasitas pekerja yang lebih besar. Mungkin juga publikasi log dapat menghambat jika tingkat transaksi per detik (TPS) terlalu rendah. `PutLogEvents` Konfigurasi CloudWatch throttling bersifat global untuk semua layanan, termasuk EMR Tanpa Server. Untuk informasi selengkapnya, lihat Bagaimana cara menentukan pembatasan di log saya? CloudWatch pada AWS re:posting.

### Izin yang diperlukan untuk login dengan CloudWatch

Sebelum pekerjaan Anda dapat mengirim data log ke Amazon CloudWatch, Anda harus menyertakan izin berikut dalam kebijakan izin untuk peran runtime pekerjaan.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:Wilayah AWS:111122223333:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
               "logs:PutLogEvents",
               "logs:CreateLogGroup",
               "logs:CreateLogStream",
               "logs:DescribeLogStreams"
            ],
            "Resource": [
               "arn:aws:logs:Wilayah AWS:111122223333:log-group:my-log-group-name:*"
            ]
        }
    ]
}
```

## AWS CLI

Untuk mengatur Amazon CloudWatch untuk menyimpan log untuk EMR Tanpa Server dari AWS CLI, gunakan `cloudWatchLoggingConfiguration` konfigurasi saat Anda memulai pekerjaan. Untuk melakukan ini, berikan penggantian konfigurasi berikut. Secara opsional, Anda juga dapat memberikan nama grup log, nama awalan aliran log, jenis log, dan kunci enkripsi. ARN

Jika Anda tidak menentukan nilai opsional, maka CloudWatch menerbitkan log ke grup log default`/aws/emr-serverless`, dengan aliran `/applications/`*`applicationId`*`/jobs/`*`jobId`*`/`*`worker-type`* log default.

EMRRilis tanpa server 7.1.0 dan mendukung upaya coba lagi yang lebih tinggi untuk pekerjaan streaming dan pekerjaan batch. Jika Anda mengaktifkan percobaan ulang untuk suatu pekerjaan, EMR Tanpa Server secara otomatis menambahkan nomor percobaan ke awalan jalur log, sehingga Anda dapat membedakan dan melacak log dengan lebih baik.

```
'/applications/<applicationId>/jobs/<jobId>/attempts/<attemptNumber>/worker-type'
```

Berikut ini menunjukkan konfigurasi minimum yang diperlukan untuk mengaktifkan CloudWatch pencatatan Amazon dengan pengaturan default untuk Tanpa EMR Server:

```
{
    "monitoringConfiguration": {
        "cloudWatchLoggingConfiguration": {
            "enabled": true
         }
     }
}
```

Contoh berikut menunjukkan semua konfigurasi wajib dan opsional yang dapat Anda tentukan saat mengaktifkan CloudWatch pencatatan Amazon untuk Tanpa EMR Server. `logTypes`Nilai yang didukung juga tercantum di bawah contoh ini.

```
{
    "monitoringConfiguration": {
        "cloudWatchLoggingConfiguration": {
            "enabled": true, // Required
            "logGroupName": "Example_logGroup", // Optional
            "logStreamNamePrefix": "Example_logStream", // Optional
            "encryptionKeyArn": "key-arn", // Optional
            "logTypes": {
                "SPARK_DRIVER": ["stdout", "stderr"] //List of values
             }
         }
     }
}
```

Secara default, EMR Tanpa Server hanya menerbitkan stdout driver dan stderr log ke. CloudWatch Jika Anda menginginkan log lain, maka Anda dapat menentukan peran kontainer dan jenis log yang sesuai dengan `logTypes` bidang tersebut.

Daftar berikut menunjukkan jenis pekerja yang didukung yang dapat Anda tentukan untuk `logTypes` konfigurasi:

Spark

- `SPARK_DRIVER : ["STDERR", "STDOUT"]`

- `SPARK_EXECUTOR : ["STDERR", "STDOUT"]`

Hive

- `HIVE_DRIVER : ["STDERR", "STDOUT", "HIVE_LOG", "TEZ_AM"]`
- `TEZ_TASK : ["STDERR", "STDOUT", "SYSTEM_LOGS"]`

## Memutar log

Amazon EMR Serverless dapat memutar log aplikasi Spark dan log peristiwa. Rotasi log membantu masalah pekerjaan yang berjalan lama menghasilkan file log besar yang dapat menghabiskan semua ruang disk Anda. Memutar log membantu Anda menghemat penyimpanan disk dan mengurangi jumlah kegagalan pekerjaan karena Anda tidak memiliki lebih banyak ruang tersisa di disk Anda.

Rotasi log diaktifkan secara default dan hanya tersedia untuk pekerjaan Spark.

Log peristiwa percikan

> **Note**
>
> Rotasi log peristiwa percikan tersedia di semua label EMR rilis Amazon.

Alih-alih menghasilkan satu file log peristiwa, EMR Serverless memutar log peristiwa pada interval waktu reguler dan menghapus file log peristiwa yang lebih lama. Memutar log tidak memengaruhi log yang diunggah ke bucket S3.

Log aplikasi percikan

> **Note**
>
> Rotasi log aplikasi Spark tersedia di semua label EMR rilis Amazon.

EMRServerless juga memutar log aplikasi spark untuk driver dan pelaksana, seperti dan file. `stdout` `stderr` Anda dapat mengakses file log terbaru dengan memilih tautan log di Studio dengan menggunakan tautan Spark History Server dan Live UI. File log adalah versi terpotong dari log terbaru. Untuk melihat log yang diputar lebih lama, Anda harus menentukan lokasi Amazon S3 saat menyimpan log. Lihat Logging untuk EMR Tanpa Server dengan bucket Amazon S3 untuk informasi selengkapnya.

Anda dapat menemukan file log terbaru di lokasi berikut. EMRTanpa server menyegarkan file setiap 15 detik. File-file ini dapat berkisar dari 0 MB hingga 128 MB.

```
<example-S3-logUri>/applications/<application-id>/jobs/<job-id>/SPARK_DRIVER/stderr.gz
```

Lokasi berikut berisi file yang diputar yang lebih lama. Setiap file berukuran 128 MB.

```
<example-S3-logUri>/applications/<application-id>/jobs/<job-id>/SPARK_DRIVER/archived/
stderr_<index>.gz
```

Perilaku yang sama berlaku untuk pelaksana Spark juga. Perubahan ini hanya berlaku untuk logging S3. Rotasi log tidak memperkenalkan perubahan apa pun pada aliran log yang diunggah ke Amazon. CloudWatch

EMRRilis tanpa server 7.1.0 dan mendukung upaya coba lagi yang lebih tinggi untuk streaming dan pekerjaan batch. Jika Anda mengaktifkan percobaan ulang dengan pekerjaan Anda, EMR Tanpa Server menambahkan awalan ke jalur log untuk pekerjaan tersebut sehingga Anda dapat melacak dan membedakan log satu sama lain dengan lebih baik. Jalur ini berisi semua log yang diputar.

```
'/applications/<applicationId>/jobs/<jobId>/attempts/<attemptNumber>/'.
```

# Mengenkripsi log

## Mengenkripsi log EMR Tanpa Server dengan penyimpanan terkelola

Untuk mengenkripsi log di penyimpanan terkelola dengan KMS kunci Anda sendiri, gunakan `managedPersistenceMonitoringConfiguration` konfigurasi saat Anda mengirimkan tugas.

```
{
    "monitoringConfiguration": {
        "managedPersistenceMonitoringConfiguration" : {
            "encryptionKeyArn": "key-arn"
        }
    }
}
```

## Mengenkripsi log EMR Tanpa Server dengan bucket Amazon S3

Untuk mengenkripsi log di bucket Amazon S3 Anda dengan kunci Anda KMS sendiri, gunakan `s3MonitoringConfiguration` konfigurasi saat Anda mengirimkan pekerjaan yang dijalankan.

```
{
    "monitoringConfiguration": {
        "s3MonitoringConfiguration": {
            "logUri": "s3://DOC-EXAMPLE-BUCKET-LOGGING/logs/",
            "encryptionKeyArn": "key-arn"
        }
    }
}
```

## Mengenkripsi log Tanpa EMR Server dengan Amazon CloudWatch

Untuk mengenkripsi log di Amazon CloudWatch dengan KMS kunci Anda sendiri, gunakan `cloudWatchLoggingConfiguration` konfigurasi saat Anda mengirimkan pekerjaan.

```
{
    "monitoringConfiguration": {
        "cloudWatchLoggingConfiguration": {
            "enabled": true,
            "encryptionKeyArn": "key-arn"
         }
     }
}
```

## Izin yang diperlukan untuk enkripsi log

Dalam bagian ini

- Izin pengguna yang diperlukan
- Izin kunci enkripsi untuk Amazon S3 dan penyimpanan terkelola
- Izin kunci enkripsi untuk Amazon CloudWatch

### Izin pengguna yang diperlukan

Pengguna yang mengirimkan pekerjaan atau melihat log atau aplikasi UIs harus memiliki izin untuk menggunakan kunci. Anda dapat menentukan izin baik dalam kebijakan KMS kunci atau IAM

kebijakan untuk pengguna, grup, atau peran. Jika pengguna yang mengirimkan pekerjaan tidak memiliki izin KMS kunci, EMR Serverless menolak pengiriman job run.

Contoh kebijakan kunci

Kebijakan utama berikut memberikan izin untuk `kms:GenerateDataKey` dan`kms:Decrypt`:

```
{
    "Effect": "Allow",
    "Principal":{
       "AWS": "arn:aws:iam::111122223333:user/user-name"
     },
     "Action": [
       "kms:GenerateDataKey",
       "kms:Decrypt"
      ],
     "Resource": "*"
 }
```

Contoh IAM kebijakan

IAMKebijakan berikut memberikan izin untuk `kms:GenerateDataKey` dan`kms:Decrypt`:

```
{
 "Version": "2012-10-17",
  "Statement": {
     "Effect": "Allow",
     "Action": [
           "kms:GenerateDataKey",
           "kms:Decrypt"
      ],
     "Resource": "key-arn"
   }
}
```

Untuk meluncurkan UI Spark atau Tez, Anda harus memberikan izin kepada pengguna, grup, atau peran Anda untuk mengakses sebagai berikut `emr-serverless:GetDashboardForJobRun`API:

```
{
 "Version": "2012-10-17",
  "Statement": {
     "Effect": "Allow",
```

```
     "Action": [
           "emr-serverless:GetDashboardForJobRun"
      ]
   }
}
```

## Izin kunci enkripsi untuk Amazon S3 dan penyimpanan terkelola

Saat Anda mengenkripsi log dengan kunci enkripsi Anda sendiri baik di penyimpanan terkelola atau di bucket S3 Anda, Anda harus mengonfigurasi izin KMS kunci sebagai berikut.

`emr-serverless.amazonaws.com`Kepala sekolah harus memiliki izin berikut dalam kebijakan untuk KMS kunci:

```
{
    "Effect": "Allow",
    "Principal":{
       "Service": "emr-serverless.amazonaws.com"
     },
     "Action": [
       "kms:Decrypt",
       "kms:GenerateDataKey"
      ],
     "Resource": "*"
     "Condition": {
       "StringLike": {
         "aws:SourceArn": "arn:aws:emr-serverless:region:aws-account-id:/
applications/application-id"
       }
     }
 }
```

Sebagai praktik keamanan terbaik, kami menyarankan Anda menambahkan kunci `aws:SourceArn` kondisi ke kebijakan KMS kunci. Kunci kondisi IAM global `aws:SourceArn` membantu memastikan bahwa EMR Tanpa Server hanya menggunakan KMS kunci untuk aplikasi. ARN

Peran runtime pekerjaan harus memiliki izin berikut dalam kebijakannyaIAM:

```
{
 "Version": "2012-10-17",
  "Statement": {
     "Effect": "Allow",
```

```
     "Action": [
           "kms:GenerateDataKey",
           "kms:Decrypt"
      ],
     "Resource": "key-arn"
   }
}
```

## Izin kunci enkripsi untuk Amazon CloudWatch

Untuk mengaitkan KMS kunci ARN ke grup log Anda, gunakan IAM kebijakan berikut untuk peran runtime pekerjaan.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "logs:AssociateKmsKey"
        ],
        "Resource": [
            "arn:aws:logs:Wilayah AWS:111122223333:log-group:my-log-group-name:*"
        ]
    }
}
```

Konfigurasikan kebijakan KMS kunci untuk memberikan KMS izin ke Amazon CloudWatch:

```
{
    "Version": "2012-10-17",
    "Id": "key-default-1",
    "Statement":
     {
        "Effect": "Allow",
        "Principal": {
            "Service": "logs.Wilayah AWS.amazonaws.com"
        },
        "Action": [
            "kms:Decrypt",
            "kms:GenerateDataKey",
        ],
        "Resource": "*",
```

```
        "Condition": {
            "ArnLike": {
                "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:Wilayah
 AWS:111122223333:*"
            }
        }
    }
}
```

# Konfigurasikan properti Apache Log4j2 untuk Amazon Tanpa Server EMR

Halaman ini menjelaskan cara mengonfigurasi properti Apache Log4j 2.x kustom untuk EMR pekerjaan Tanpa Server di. `StartJobRun` Jika Anda ingin mengonfigurasi klasifikasi Log4j di tingkat aplikasi, lihat. Konfigurasi aplikasi default untuk Tanpa EMR Server

## Konfigurasikan properti Spark Log4j2 untuk Amazon Tanpa Server EMR

Dengan Amazon EMR merilis 6.8.0 dan yang lebih tinggi, Anda dapat menyesuaikan properti Apache Log4j 2.x untuk menentukan konfigurasi log berbutir halus. Ini menyederhanakan pemecahan masalah pekerjaan Spark Anda di Tanpa Server. EMR Untuk mengkonfigurasi properti ini, gunakan `spark-driver-log4j2` dan `spark-executor-log4j2` klasifikasi.

Topik

- Klasifikasi Log4j2 untuk Spark
- Contoh konfigurasi Log4j2 untuk Spark
- Log4j2 dalam contoh pekerjaan Spark
- Pertimbangan Log4j2 untuk Spark

### Klasifikasi Log4j2 untuk Spark

Untuk menyesuaikan konfigurasi log Spark, gunakan klasifikasi berikut dengan. `applicationConfiguration` Untuk mengkonfigurasi properti Log4j 2.x, gunakan yang berikut ini. `properties`

**`spark-driver-log4j2`**

Klasifikasi ini menetapkan nilai dalam `log4j2.properties` file untuk driver.

**`spark-executor-log4j2`**

Klasifikasi ini menetapkan nilai dalam `log4j2.properties` file untuk pelaksana.

## Contoh konfigurasi Log4j2 untuk Spark

Contoh berikut menunjukkan cara mengirimkan pekerjaan Spark dengan untuk menyesuaikan konfigurasi Log4j2 `applicationConfiguration` untuk driver dan eksekutor Spark.

Untuk mengonfigurasi klasifikasi Log4j di tingkat aplikasi, bukan saat Anda mengirimkan pekerjaan, lihat. Konfigurasi aplikasi default untuk Tanpa EMR Server

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "sparkSubmit": {
            "entryPoint": "/usr/lib/spark/examples/jars/spark-examples.jar",
            "entryPointArguments": ["1"],
            "sparkSubmitParameters": "--class org.apache.spark.examples.SparkPi --conf
 spark.executor.cores=4 --conf spark.executor.memory=20g --conf spark.driver.cores=4 --
conf spark.driver.memory=8g --conf spark.executor.instances=1"
        }
    }'
    --configuration-overrides '{
        "applicationConfiguration": [
             {
                "classification": "spark-driver-log4j2",
                "properties": {
                    "rootLogger.level":"error", // will only display Spark error logs
                    "logger.IdentifierForClass.name": "classpath for setting logger",
                    "logger.IdentifierForClass.level": "info"

                }
            },
            {
                "classification": "spark-executor-log4j2",
                "properties": {
                    "rootLogger.level":"error", // will only display Spark error logs
                    "logger.IdentifierForClass.name": "classpath for setting logger",
                    "logger.IdentifierForClass.level": "info"
                }
            }
```

```
    ]
  }'
```

## Log4j2 dalam contoh pekerjaan Spark

Contoh kode berikut menunjukkan cara membuat aplikasi Spark saat Anda menginisialisasi konfigurasi Log4j2 kustom untuk aplikasi.

Python

Example - Menggunakan Log4j2 untuk pekerjaan Spark dengan Python

```
import os
import sys

from pyspark import SparkConf, SparkContext
from pyspark.sql import SparkSession

app_name = "PySparkApp"
if __name__ == "__main__":
    spark = SparkSession\
        .builder\
        .appName(app_name)\
        .getOrCreate()

    spark.sparkContext._conf.getAll()
    sc = spark.sparkContext
    log4jLogger = sc._jvm.org.apache.log4j
    LOGGER = log4jLogger.LogManager.getLogger(app_name)

    LOGGER.info("pyspark script logger info")
    LOGGER.warn("pyspark script logger warn")
    LOGGER.error("pyspark script logger error")

    // your code here

    spark.stop()
```

Untuk menyesuaikan Log4j2 untuk driver saat Anda menjalankan pekerjaan Spark, Anda dapat menggunakan konfigurasi berikut:

```
{
```

```
   "classification": "spark-driver-log4j2",
      "properties": {
          "rootLogger.level":"error", // only display Spark error logs
          "logger.PySparkApp.level": "info",
          "logger.PySparkApp.name": "PySparkApp"
      }
}
```

Scala

Example - Menggunakan Log4j2 untuk pekerjaan Spark dengan Scala

```
import org.apache.log4j.Logger
import org.apache.spark.sql.SparkSession

object ExampleClass {
  def main(args: Array[String]): Unit = {
    val spark = SparkSession
    .builder
    .appName(this.getClass.getName)
    .getOrCreate()

    val logger = Logger.getLogger(this.getClass);
    logger.info("script logging info logs")
    logger.warn("script logging warn logs")
    logger.error("script logging error logs")

// your code here
    spark.stop()
  }
}
```

Untuk menyesuaikan Log4j2 untuk driver saat Anda menjalankan pekerjaan Spark, Anda dapat menggunakan konfigurasi berikut:

```
{
   "classification": "spark-driver-log4j2",
      "properties": {
          "rootLogger.level":"error", // only display Spark error logs
          "logger.ExampleClass.level": "info",
          "logger.ExampleClass.name": "ExampleClass"
      }
```

```
}
```

### Pertimbangan Log4j2 untuk Spark

Properti Log4j2.x berikut tidak dapat dikonfigurasi untuk proses Spark:

- `rootLogger.appenderRef.stdout.ref`
- `appender.console.type`
- `appender.console.name`
- `appender.console.target`
- `appender.console.layout.type`
- `appender.console.layout.pattern`

Untuk informasi rinci tentang properti Log4j2.x yang dapat Anda konfigurasi, lihat file di. `log4j2.properties.template` GitHub

## Pemantauan Tanpa EMR Server

Bagian ini mencakup cara-cara Anda dapat memantau aplikasi dan pekerjaan EMR Tanpa Server Amazon Anda.

Topik

- Memantau aplikasi dan EMR pekerjaan tanpa server
- Pantau metrik Spark dengan Amazon Managed Service untuk Prometheus
- EMRMetrik penggunaan tanpa server

### Memantau aplikasi dan EMR pekerjaan tanpa server

Dengan CloudWatch metrik Amazon untuk EMR Tanpa Server, Anda dapat menerima CloudWatch metrik 1 menit dan mengakses CloudWatch dasbor untuk melihat near-real-time operasi dan kinerja aplikasi Tanpa Server Anda. EMR

EMRTanpa server mengirimkan metrik ke CloudWatch setiap menit. EMRTanpa server memancarkan metrik ini di tingkat aplikasi serta pekerjaan, tipe pekerja, dan level. capacity-allocation-type

Untuk memulai, gunakan template CloudWatch dasbor EMR Tanpa Server yang disediakan di GitHub repositori EMR Tanpa Server dan terapkan.

> **Note**
>
> EMRBeban kerja interaktif tanpa server hanya mengaktifkan pemantauan tingkat aplikasi, dan memiliki dimensi tipe pekerja baru. `Spark_Kernel` Untuk memantau dan men-debug beban kerja interaktif Anda, Anda dapat melihat log dan Apache Spark UI dari dalam Ruang Kerja Studio Anda. EMR

Tabel di bawah ini menjelaskan dimensi EMR Tanpa Server yang tersedia dalam namespace. `AWS/EMRServerless`

Dimensi untuk EMR metrik Tanpa Server

| Dimensi | Deskripsi |
| --- | --- |
| `ApplicationId` | Filter untuk semua metrik aplikasi Tanpa EMR Server. |
| `JobId` | Filter untuk semua metrik menjalankan pekerjaan EMR Tanpa Server. |
| `WorkerType` | Filter untuk semua metrik dari jenis pekerja tertentu. Misalnya, Anda dapat memfilter untuk `SPARK_DRIVER` dan `SPARK_EXECUTORS` untuk pekerjaan Spark. |
| `CapacityAllocationType` | Filter untuk semua metrik dari jenis alokasi kapasitas tertentu. Misalnya, Anda dapat memfilter `PreInitCapacity` untuk kapasitas pra- |

| Dimensi | Deskripsi | |
|---|---|---|
| | inisialisasi dan `OnDemandCapacity` untuk yang lainnya. | |

## Pemantauan tingkat aplikasi

Anda dapat memantau penggunaan kapasitas di tingkat aplikasi EMR Tanpa Server dengan metrik Amazon CloudWatch . Anda juga dapat mengatur satu tampilan untuk memantau penggunaan kapasitas aplikasi di CloudWatch dasbor.

EMRMetrik aplikasi tanpa server

| Metrik | Deskripsi | Dimensi primer | Dimensi sekunder |
|---|---|---|---|
| `CPUAllocated` | Jumlah total yang vCPUs dialokasikan. | `ApplicationId` | `ApplicationId` , `WorkerType` , `CapacityAllocationType` |
| `IdleWorkerCount` | Jumlah total pekerja yang menganggur. | `ApplicationId` | `ApplicationId` , `WorkerType` , `CapacityAllocationType` |
| `MaxCPUAllowed` | Maksimum yang CPU diizinkan untuk aplikasi. | `ApplicationId` | N/A |
| `MaxMemoryAllowed` | Memori maksimum dalam GB diperbolehkan untuk aplikasi. | `ApplicationId` | N/A |
| `MaxStorageAllowed` | Penyimpanan maksimum dalam GB diperbolehkan untuk aplikasi. | `ApplicationId` | N/A |

| Metrik | Deskripsi | Dimensi primer | Dimensi sekunder |
| --- | --- | --- | --- |
| MemoryAllocated | Total memori dalam GB dialokasikan. | ApplicationId | ApplicationId , WorkerType , CapacityAllocationType |
| PendingCreationWorkerCount | Jumlah total pekerja yang menunggu penciptaan. | ApplicationId | ApplicationId , WorkerType , CapacityAllocationType |
| RunningWorkerCount | Jumlah total pekerja yang digunakan oleh aplikasi. | ApplicationId | ApplicationId , WorkerType , CapacityAllocationType |
| StorageAllocated | Total penyimpanan disk dalam GB dialokasikan. | ApplicationId | ApplicationId , WorkerType , CapacityAllocationType |
| TotalWorkerCount | Jumlah total pekerja yang tersedia. | ApplicationId | ApplicationId , WorkerType , CapacityAllocationType |

## Pemantauan tingkat pekerjaan

Amazon EMR Serverless mengirimkan metrik tingkat pekerjaan berikut ke Amazon CloudWatch setiap satu menit. Anda dapat melihat nilai metrik untuk pekerjaan agregat yang dijalankan berdasarkan status menjalankan pekerjaan. Satuan untuk setiap metrik adalah hitungan.

EMRMetrik tingkat pekerjaan tanpa server

| Metrik | Deskripsi | Dimensi primer |
| --- | --- | --- |
| `SubmittedJobs` | Jumlah pekerjaan di negara bagian yang Dikirim. | `ApplicationId` |
| `PendingJobs` | Jumlah pekerjaan dalam keadaan Tertunda. | `ApplicationId` |
| `ScheduledJobs` | Jumlah pekerjaan dalam keadaan Terjadwal. | `ApplicationId` |
| `RunningJobs` | Jumlah pekerjaan dalam keadaan Running. | `ApplicationId` |
| `SuccessJobs` | Jumlah pekerjaan dalam keadaan Sukses. | `ApplicationId` |
| `FailedJobs` | Jumlah pekerjaan dalam keadaan Gagal. | `ApplicationId` |
| `CancellingJobs` | Jumlah pekerjaan di negara Membatalkan. | `ApplicationId` |
| `CancelledJobs` | Jumlah pekerjaan di negara yang Dibatalkan. | `ApplicationId` |

Anda dapat memantau metrik khusus mesin untuk menjalankan dan menyelesaikan pekerjaan Tanpa EMR Server dengan aplikasi khusus mesin. UIs Saat Anda melihat UI untuk pekerjaan yang sedang berjalan, Anda melihat UI aplikasi langsung dengan pembaruan waktu nyata. Saat Anda melihat UI untuk pekerjaan yang diselesaikan, Anda akan melihat UI aplikasi persisten.

Menjalankan pekerjaan

Untuk menjalankan pekerjaan EMR Tanpa Server, Anda dapat melihat antarmuka real-time yang menyediakan metrik khusus mesin. Anda dapat menggunakan Apache Spark UI atau Hive Tez UI untuk memantau dan men-debug pekerjaan Anda. Untuk mengaksesnyaUIs, gunakan konsol EMR Studio atau minta URL titik akhir yang aman dengan AWS Command Line Interface.

Lowongan kerja yang telah selesai

Untuk pekerjaan EMR Tanpa Server yang telah selesai, Anda dapat menggunakan Spark History Server atau Persistent Hive Tez UI untuk melihat detail pekerjaan, tahapan, tugas, dan metrik untuk menjalankan pekerjaan Spark atau Hive. Untuk mengaksesnyaUIs, gunakan konsol EMR Studio, atau minta URL titik akhir yang aman dengan AWS Command Line Interface.

## Pemantauan tingkat pekerja Job

Amazon EMR Serverless mengirimkan metrik tingkat pekerja kerja berikut yang tersedia di `AWS/EMRServerless` namespace dan grup metrik `Job Worker Metrics` ke Amazon. CloudWatch EMRTanpa server mengumpulkan poin data dari pekerja individu selama pekerjaan berjalan di tingkat pekerjaan, tipe pekerja, dan tingkat. capacity-allocation-type Anda dapat menggunakan `ApplicationId` sebagai dimensi untuk memantau beberapa pekerjaan yang termasuk dalam aplikasi yang sama.

EMRMetrik tingkat pekerja pekerjaan tanpa server

| Metrik | Deskripsi | Unit | Dimensi primer | Dimensi sekunder |
|---|---|---|---|---|
| `WorkerCpuAllocated` | Jumlah total v CPU core yang dialokasikan untuk pekerja dalam menjalankan pekerjaan. | Tidak ada | `JobId` | `ApplicationId` , `WorkerType` , dan `CapacityAllocationType` |
| `WorkerCpuUsed` | Jumlah total v CPU core yang digunakan oleh pekerja dalam menjalankan pekerjaan. | Tidak ada | `JobId` | `ApplicationId` , `WorkerType` , dan `CapacityAllocationType` |
| `WorkerMemoryAllocated` | Total memori dalam GB dialokasikan untuk pekerja | Gigabyte (GB) | `JobId` | `ApplicationId` , `WorkerType` , dan `CapacityA` |

| Metrik | Deskripsi | Unit | Dimensi primer | Dimensi sekunder |
|---|---|---|---|---|
| | dalam menjalankan pekerjaan. | | | `llocation Type` |
| `WorkerMemoryUsed` | Total memori dalam GB yang digunakan oleh pekerja dalam menjalankan pekerjaan. | Gigabyte (GB) | `JobId` | `ApplicationId`, `WorkerType`, dan `CapacityAllocation Type` |
| `WorkerEphemeralStorageAllocated` | Jumlah byte penyimpanan sementara yang dialokasikan untuk pekerja dalam menjalankan pekerjaan. | Gigabyte (GB) | `JobId` | `ApplicationId`, `WorkerType`, dan `CapacityAllocation Type` |
| `WorkerEphemeralStorageUsed` | Jumlah byte penyimpanan sementara yang digunakan oleh pekerja dalam menjalankan pekerjaan. | Gigabyte (GB) | `JobId` | `ApplicationId`, `WorkerType`, dan `CapacityAllocation Type` |
| `WorkerStorageReadBytes` | Jumlah byte yang dibaca dari penyimpanan oleh pekerja dalam menjalankan pekerjaan. | Byte | `JobId` | `ApplicationId`, `WorkerType`, dan `CapacityAllocation Type` |

| Metrik | Deskripsi | Unit | Dimensi primer | Dimensi sekunder |
|---|---|---|---|---|
| `WorkerStorageWriteBytes` | Jumlah byte yang ditulis ke penyimpanan dari pekerja dalam menjalankan pekerjaan. | Byte | `JobId` | `ApplicationId` , `WorkerType` , dan `CapacityAllocationType` |

Langkah-langkah di bawah ini menjelaskan cara melihat berbagai jenis metrik.

Console

Untuk mengakses UI aplikasi Anda dengan konsol

1. Arahkan ke aplikasi EMR Tanpa Server Anda di EMR Studio dengan petunjuk di Memulai dari konsol.
2. Untuk melihat aplikasi UIs dan log khusus mesin untuk pekerjaan yang sedang berjalan:
   a. Pilih pekerjaan dengan `RUNNING` status.
   b. Pilih pekerjaan di halaman Detail aplikasi, atau navigasikan ke halaman Detail Pekerjaan untuk pekerjaan Anda.
   c. Di bawah menu tarik-turun Display UI, pilih Spark UI atau Hive Tez UI untuk menavigasi ke UI aplikasi untuk jenis pekerjaan Anda.
   d. Untuk melihat log mesin Spark, navigasikan ke tab Executors di UI Spark, dan pilih tautan Log untuk driver. Untuk melihat log mesin Hive, pilih tautan Log yang sesuai DAG di UI Hive Tez.
3. Untuk melihat aplikasi UIs dan log khusus mesin untuk pekerjaan yang diselesaikan:
   a. Pilih pekerjaan dengan `SUCCESS` status.
   b. Pilih pekerjaan di halaman detail Aplikasi lamaran Anda atau navigasikan ke halaman detail Pekerjaan.
   c. Di bawah menu tarik-turun Display UI, pilih Spark History Server atau Persistent Hive Tez UI untuk menavigasi ke UI aplikasi untuk jenis pekerjaan Anda.

d. Untuk melihat log mesin Spark, navigasikan ke tab Executors di UI Spark, dan pilih tautan Log untuk driver. Untuk melihat log mesin Hive, pilih tautan Log yang sesuai DAG di UI Hive Tez.

AWS CLI

Untuk mengakses UI aplikasi Anda dengan AWS CLI

- Untuk menghasilkan URL yang dapat Anda gunakan untuk mengakses UI aplikasi Anda untuk pekerjaan yang berjalan dan diselesaikan, hubungi file `GetDashboardForJobRun`API.

```
aws emr-serverless get-dashboard-for-job-run /
--application-id <application-id> /
--job-run-id <job-id>
```

URLYang Anda hasilkan berlaku selama satu jam.

## Pantau metrik Spark dengan Amazon Managed Service untuk Prometheus

Dengan Amazon EMR merilis 7.1.0 dan yang lebih tinggi, Anda dapat mengintegrasikan Tanpa EMR Server dengan Amazon Managed Service untuk Prometheus guna mengumpulkan metrik Apache Spark untuk pekerjaan dan aplikasi Tanpa Server. EMR Integrasi ini tersedia saat Anda mengirimkan pekerjaan atau membuat aplikasi menggunakan salah satu AWS konsol, EMR Tanpa ServerAPI, atau AWS CLI.

### Prasyarat

Sebelum Anda dapat mengirimkan metrik Spark Anda ke Amazon Managed Service untuk Prometheus, Anda harus menyelesaikan prasyarat berikut.

- Buat Layanan Terkelola Amazon untuk ruang kerja Prometheus. Ruang kerja ini berfungsi sebagai titik akhir konsumsi. Buat catatan yang URL ditampilkan untuk Endpoint - tulis URL jarak jauh. Anda harus menentukan URL kapan Anda membuat aplikasi EMR Tanpa Server Anda.
- Untuk memberikan akses pekerjaan Anda ke Amazon Managed Service untuk Prometheus untuk tujuan pemantauan, tambahkan kebijakan berikut ke peran pelaksanaan pekerjaan Anda.

```
{
```

```
    "Sid": "AccessToPrometheus",
    "Effect": "Allow",
    "Action": ["aps:RemoteWrite"],
    "Resource": "arn:aws:aps:<AWS_REGION>:<AWS_ACCOUNT_ID>:workspace/<WORKSPACE_ID>"
}
```

## Pengaturan

Untuk menggunakan AWS konsol untuk membuat aplikasi yang terintegrasi dengan Amazon Managed Service untuk Prometheus

1. Lihat Memulai Amazon EMR Tanpa Server untuk membuat aplikasi.
2. Saat Anda membuat aplikasi, pilih Gunakan pengaturan khusus, lalu konfigurasikan aplikasi Anda dengan menentukan informasi ke dalam bidang yang ingin Anda konfigurasikan.
3. Di bawah Log dan metrik aplikasi, pilih Mengirimkan metrik engine ke Amazon Managed Service for Prometheus, lalu tentukan penulisan jarak jauh Anda. URL
4. Tentukan pengaturan konfigurasi lain yang Anda inginkan, lalu pilih Buat dan mulai aplikasi.

### Gunakan AWS CLI atau Tanpa EMR Server API

Anda juga dapat menggunakan AWS CLI atau EMR Tanpa Server API untuk mengintegrasikan EMR aplikasi Tanpa Server Anda dengan Amazon Managed Service untuk Prometheus saat Anda menjalankan atau perintah. `create-application` `start-job-run`

create-application

```
aws emr-serverless create-application \
--release-label emr-7.1.0 \
--type "SPARK" \
--monitoring-configuration '{
    "prometheusMonitoringConfiguration": {
        "remoteWriteUrl": "https://aps-workspaces.<AWS_REGION>.amazonaws.com/
workspaces/<WORKSPACE_ID>/api/v1/remote_write"
    }
}'
```

start-job-run

```
aws emr-serverless start-job-run \
```

```
--application-id <APPPLICATION_ID> \
--execution-role-arn <JOB_EXECUTION_ROLE> \
--job-driver '{
    "sparkSubmit": {
        "entryPoint": "local:///usr/lib/spark/examples/src/main/python/pi.py",
        "entryPointArguments": ["10000"],
        "sparkSubmitParameters": "--conf spark.dynamicAllocation.maxExecutors=10"
    }
}' \
--configuration-overrides '{
     "monitoringConfiguration": {
        "prometheusMonitoringConfiguration": {
            "remoteWriteUrl": "https://aps-workspaces.<AWS_REGION>.amazonaws.com/
workspaces/<WORKSPACE_ID>/api/v1/remote_write"
        }
    }
}'
```

Termasuk `prometheusMonitoringConfiguration` dalam perintah Anda menunjukkan bahwa EMR Tanpa Server harus menjalankan pekerjaan Spark dengan agen yang mengumpulkan metrik Spark dan menuliskannya ke titik akhir Anda untuk `remoteWriteUrl` Amazon Managed Service for Prometheus. Anda kemudian dapat menggunakan metrik Spark di Amazon Managed Service for Prometheus untuk visualisasi, peringatan, dan analisis.

## Properti konfigurasi lanjutan

EMRTanpa server menggunakan komponen dalam nama Spark `PrometheusServlet` untuk mengumpulkan metrik Spark dan menerjemahkan data kinerja ke dalam data yang kompatibel dengan Amazon Managed Service for Prometheus. Secara default, EMR Tanpa Server menetapkan nilai default di Spark dan mem-parsing metrik driver dan eksekutor saat Anda mengirimkan pekerjaan menggunakan. `PrometheusMonitoringConfiguration`

Tabel berikut menjelaskan semua properti yang dapat Anda konfigurasikan saat mengirimkan pekerjaan Spark yang mengirimkan metrik ke Amazon Managed Service for Prometheus.

| Properti percikan | Nilai default | Deskripsi |
| --- | --- | --- |
| `spark.metrics.conf .*.sink.prometheus Servlet.class` | org.apache.spark.metrics.sink. PrometheusServlet | Kelas yang digunakan Spark untuk mengirim metrik ke Amazon Managed Service |

| Properti percikan | Nilai default | Deskripsi |
| --- | --- | --- |
| | | untuk Prometheus. Untuk mengganti perilaku default, tentukan kelas kustom Anda sendiri. |
| `spark.metrics.conf.*.source.jvm.class` | org.apache.spark.metrics.source. JvmSource | Kelas Spark digunakan untuk mengumpulkan dan mengirim metrik penting dari mesin virtual Java yang mendasarinya. Untuk berhenti mengumpulkan JVM metrik, nonaktifkan properti ini dengan menyetelnya ke string kosong, seperti"". Untuk mengganti perilaku default, tentukan kelas kustom Anda sendiri. |
| `spark.metrics.conf.driver.sink.prometheusServlet.path` | /metrik/prometheus | Perbedaan URL yang digunakan Amazon Managed Service untuk Prometheus untuk mengumpulkan metrik dari driver. Untuk mengganti perilaku default, tentukan jalur Anda sendiri. Untuk berhenti mengumpulkan metrik driver, nonaktifkan properti ini dengan menyetelnya ke string kosong, seperti"". |

| Properti percikan | Nilai default | Deskripsi |
| --- | --- | --- |
| `spark.metrics.conf.executor.sink.prometheusServlet.path` | /metrik/pelaksana/prometheus | Perbedaan URL yang digunakan Amazon Managed Service untuk Prometheus untuk mengumpulkan metrik dari pelaksana. Untuk mengganti perilaku default, tentukan jalur Anda sendiri. Untuk berhenti mengumpulkan metrik pelaksana, nonaktifkan properti ini dengan menyetelnya ke string kosong, seperti. "" |

Untuk informasi selengkapnya tentang metrik Spark, lihat metrik Apache Spark.

### Pertimbangan dan batasan

Saat menggunakan Layanan Terkelola Amazon untuk Prometheus untuk mengumpulkan metrik EMR dari Tanpa Server, pertimbangkan pertimbangan dan batasan berikut.

- Support untuk menggunakan Amazon Managed Service untuk Prometheus EMR dengan Serverless hanya tersedia di Wilayah AWS di mana Amazon Managed Service untuk Prometheus umumnya tersedia.
- Menjalankan agen untuk mengumpulkan metrik Spark di Amazon Managed Service untuk Prometheus membutuhkan lebih banyak sumber daya dari pekerja. Jika Anda memilih ukuran pekerja yang lebih kecil, seperti satu CPU pekerja v, waktu kerja Anda mungkin meningkat.
- Support untuk menggunakan Amazon Managed Service untuk Prometheus EMR dengan Serverless hanya tersedia untuk Amazon rilis 7.1.0 dan yang lebih tinggi. EMR

## EMRMetrik penggunaan tanpa server

Anda dapat menggunakan metrik CloudWatch penggunaan Amazon untuk memberikan visibilitas ke sumber daya yang digunakan akun Anda. Gunakan metrik ini untuk memvisualisasikan penggunaan layanan Anda pada CloudWatch grafik dan dasbor.

EMRMetrik penggunaan tanpa server sesuai dengan Service Quotas. Anda dapat mengonfigurasi alarm yang memberi tahu Anda saat penggunaan mendekati kuota layanan. Untuk informasi selengkapnya, lihat Service Quotas dan CloudWatch alarm Amazon di Panduan Pengguna Service Quotas.

Untuk informasi selengkapnya tentang kuota layanan EMR Tanpa Server, lihat. Titik akhir dan kuota untuk EMR Serverless

## Metrik penggunaan kuota layanan untuk Tanpa Server EMR

EMRTanpa server menerbitkan metrik penggunaan kuota layanan berikut di namespace. `AWS/Usage`

| Metrik | Deskripsi |
| --- | --- |
| `ResourceCount` | Jumlah total sumber daya yang ditentukan yang berjalan di akun Anda. Sumber daya ditentukan oleh dimensi yang terkait dengan metrik. |

## Dimensi untuk EMR metrik penggunaan kuota layanan Tanpa Server

Anda dapat menggunakan dimensi berikut untuk menyempurnakan metrik penggunaan yang diterbitkan Tanpa EMR Server.

| Dimensi | Nilai | Deskripsi |
| --- | --- | --- |
| `Service` | EMRTanpa server | Nama dari Layanan AWS yang berisi sumber daya. |
| `Type` | Sumber Daya | Jenis entitas yang dilaporkan EMR Tanpa Server. |
| `Resource` | v CPU | Jenis sumber daya yang EMR dilacak Tanpa Server. |
| `Class` | Tidak ada | Kelas sumber daya yang EMR dilacak Tanpa Server. |

# Mengotomatisasi Tanpa EMR Server dengan Amazon EventBridge

Anda dapat menggunakan Amazon EventBridge untuk mengotomatiskan Layanan AWS dan merespons secara otomatis peristiwa sistem, seperti masalah ketersediaan aplikasi atau perubahan sumber daya. EventBridge memberikan aliran peristiwa sistem yang mendekati waktu nyata yang menggambarkan perubahan dalam AWS sumber daya. Anda dapat menulis aturan sederhana untuk menunjukkan kejadian mana yang sesuai kepentingan Anda, dan tindakan otomatis apa yang diambil ketika suatu kejadian sesuai dengan suatu aturan. Dengan EventBridge, Anda dapat secara otomatis:

- Memohon AWS Lambda Fungsi
- Relay peristiwa ke Amazon Kinesis Data Streams
- Aktifkan sebuah AWS Step Functions mesin status
- Beri tahu SNS topik Amazon atau antrian Amazon SQS

Misalnya, ketika Anda menggunakan EventBridge dengan EMR Tanpa Server, Anda dapat mengaktifkan AWS Lambda berfungsi saat ETL pekerjaan berhasil atau beri tahu SNS topik Amazon saat ETL pekerjaan gagal.

EMRTanpa server memancarkan tiga jenis peristiwa:

- Peristiwa perubahan status aplikasi — Peristiwa yang memancarkan setiap perubahan status aplikasi. Untuk informasi selengkapnya tentang status aplikasi, lihatStatus aplikasi.
- Job run state change events — Peristiwa yang memancarkan setiap perubahan status dari pekerjaan yang dijalankan. Untuk informasi lebih lanjut tentang, lihatStatus tugas berjalan.
- Job run retry events — Peristiwa yang memancarkan setiap percobaan ulang pekerjaan yang dijalankan dari Amazon EMR Serverless merilis 7.1.0 dan yang lebih tinggi.

## Contoh EMR acara Tanpa Server EventBridge

Peristiwa yang dilaporkan oleh EMR Tanpa Server memiliki nilai yang `aws.emr-serverless` ditetapkan`source`, seperti pada contoh berikut.

Acara perubahan status aplikasi

Contoh peristiwa berikut menunjukkan aplikasi di `CREATING` negara bagian.

```
{
```

```
    "version": "0",
    "id": "9fd3cf79-1ff1-b633-4dd9-34508dc1e660",
    "detail-type": "EMR Serverless Application State Change",
    "source": "aws.emr-serverless",
    "account": "123456789012",
    "time": "2022-05-31T21:16:31Z",
    "region": "us-east-1",
    "resources": [],
    "detail": {
        "applicationId": "00f1cbsc6anuij25",
        "applicationName": "3965ad00-8fba-4932-a6c8-ded32786fd42",
        "arn": "arn:aws:emr-serverless:us-east-1:111122223333:/
applications/00f1cbsc6anuij25",
        "releaseLabel": "emr-6.6.0",
        "state": "CREATING",
        "type": "HIVE",
        "createdAt": "2022-05-31T21:16:31.547953Z",
        "updatedAt": "2022-05-31T21:16:31.547970Z",
        "autoStopConfig": {
            "enabled": true,
            "idleTimeout": 15
        },
        "autoStartConfig": {
            "enabled": true
        }
    }
}
```

Acara perubahan status Job run

Contoh peristiwa berikut menunjukkan job run yang berpindah dari `SCHEDULED` state ke `RUNNING` state.

```
{
    "version": "0",
    "id": "00df3ec6-5da1-36e6-ab71-20f0de68f8a0",
    "detail-type": "EMR Serverless Job Run State Change",
    "source": "aws.emr-serverless",
    "account": "123456789012",
    "time": "2022-05-31T21:07:42Z",
    "region": "us-east-1",
    "resources": [],
    "detail": {
```

```
        "jobRunId": "00f1cbn5g4bb0c01",
        "applicationId": "00f1982r1uukb925",
        "arn": "arn:aws:emr-serverless:us-east-1:123456789012:/
applications/00f1982r1uukb925/jobruns/00f1cbn5g4bb0c01",
        "releaseLabel": "emr-6.6.0",
        "state": "RUNNING",
        "previousState": "SCHEDULED",
        "createdBy": "arn:aws:sts::123456789012:assumed-role/
TestRole-402dcef3ad14993c15d28263f64381e4cda34775/6622b6233b6d42f59c25dd2637346242",
        "updatedAt": "2022-05-31T21:07:42.299487Z",
        "createdAt": "2022-05-31T21:07:25.325900Z"
    }
}
```

Job run acara coba lagi

Berikut ini adalah contoh acara coba ulang job run.

```
{
    "version": "0",
    "id": "00df3ec6-5da1-36e6-ab71-20f0de68f8a0",
    "detail-type": "EMR Serverless Job Run Retry",
    "source": "aws.emr-serverless",
    "account": "123456789012",
    "time": "2022-05-31T21:07:42Z",
    "region": "us-east-1",
    "resources": [],
    "detail": {
        "jobRunId": "00f1cbn5g4bb0c01",
        "applicationId": "00f1982r1uukb925",
        "arn": "arn:aws:emr-serverless:us-east-1:123456789012:/
applications/00f1982r1uukb925/jobruns/00f1cbn5g4bb0c01",
        "releaseLabel": "emr-6.6.0",
        "createdBy": "arn:aws:sts::123456789012:assumed-role/
TestRole-402dcef3ad14993c15d28263f64381e4cda34775/6622b6233b6d42f59c25dd2637346242",
        "updatedAt": "2022-05-31T21:07:42.299487Z",
        "createdAt": "2022-05-31T21:07:25.325900Z",
        //Attempt Details
        "previousAttempt": 1,
        "previousAttemptState": "FAILED",
        "previousAttemptCreatedAt": "2022-05-31T21:07:25.325900Z",
        "previousAttemptEndedAt": "2022-05-31T21:07:30.325900Z",
        "newAttempt": 2,
        "newAttemptCreatedAt": "2022-05-31T21:07:30.325900Z"
```

```
    }
}
```

# Penandaan pada sumber daya

Anda dapat menetapkan metadata Anda sendiri ke setiap sumber daya menggunakan tag untuk membantu Anda EMR mengelola sumber daya Tanpa Server. Bagian ini memberikan ikhtisar fungsi tag dan menunjukkan cara membuat tag.

Topik



## Apa itu tag?

Tag adalah label yang Anda tetapkan ke AWS sumber daya. Setiap tanda terdiri dari kunci dan nilai, yang keduanya Anda tentukan. Tag memungkinkan Anda untuk mengkategorikan AWS sumber daya berdasarkan atribut seperti tujuan, pemilik, dan lingkungan. Saat Anda memiliki banyak sumber daya dengan jenis yang sama, Anda dapat dengan segera mengidentifikasi sumber daya yang spesifik berdasarkan tanda yang telah Anda tetapkan pada sumber daya. Misalnya, Anda dapat menentukan satu set tag untuk aplikasi Amazon EMR Tanpa Server untuk membantu Anda melacak setiap pemilik dan tingkat tumpukan aplikasi. Kami menyarankan agar Anda merancang serangkaian kunci tanda yang konsisten untuk setiap jenis sumber daya.

Selain itu, tanda tidak dapat menetapkan secara otomatis ke sumber daya Anda. Setelah menambahkan tag ke sumber daya, Anda dapat mengubah nilai tag atau menghapus tag dari sumber daya kapan saja. Tag tidak memiliki arti semantik untuk Amazon EMR Tanpa Server dan ditafsirkan secara ketat sebagai string karakter. Jika Anda menambahkan tanda yang memiliki kunci yang sama dengan tanda yang ada pada sumber daya tersebut, nilai baru akan menimpa nilai sebelumnya.

Jika Anda menggunakanIAM, Anda dapat mengontrol pengguna mana yang ada di AWS akun memiliki izin untuk mengelola tag. Untuk contoh kebijakan kontrol akses berbasis tanda, lihat Kebijakan untuk kendali akses berbasis tanda.

# Pemberian tag pada sumber daya Anda

Anda dapat menandai aplikasi baru atau yang sudah ada dan menjalankan pekerjaan. Jika Anda menggunakan Amazon EMR Tanpa ServerAPI, AWS CLI, atau AWS SDK, Anda dapat menerapkan tag ke sumber daya baru menggunakan `tags` parameter pada API tindakan yang relevan. Anda dapat menerapkan tag ke sumber daya yang ada menggunakan `TagResource` API tindakan.

Anda dapat menggunakan beberapa tindakan penciptaan sumber daya untuk menentukan tanda untuk sumber daya saat sumber daya diciptakan. Dalam kasus ini, jika tanda tidak dapat diterapkan saat sumber daya sedang dibuat, sumber daya gagal untuk dibuat. Mekanisme ini memastikan bahwa sumber daya yang Anda inginkan untuk ditandai pada penciptaan dibuat dengan tanda tertentu atau tidak dibuat sama sekali. Jika Anda menandai sumber daya pada saat pembuatan, Anda tidak perlu menjalankan skrip penandaan khusus setelah membuat sumber daya.

Tabel berikut menjelaskan sumber daya Amazon EMR Tanpa Server yang dapat ditandai.

| Sumber daya | Mendukung tanda | Penyebaran tanda Support | Mendukung penandaan pada pembuatan (Amazon Tanpa EMR ServerAPI , AWS CLI, dan AWS SDK) | APIuntuk pembuatan (tag dapat ditambahkan selama pembuatan) |
|---|---|---|---|---|
| Aplikasi | Ya | Tidak. Tag yang terkait dengan aplikasi tidak menyebar ke pekerjaan yang dikirimkan ke aplikasi itu. | Ya | `CreateApplication` |
| Tugas berjalan | Ya | Tidak | Ya | `StartJobRun` |

# Batasan penandaan

Batasan dasar berikut berlaku untuk tag:

- Setiap sumber daya dapat memiliki maksimal 50 tag yang dibuat pengguna.
- Untuk setiap sumber daya, setiap kunci tag harus unik, dan setiap kunci tag hanya dapat memiliki satu nilai.
- Panjang kunci maksimum adalah 128 karakter Unicode di UTF -8.
- Panjang nilai maksimum adalah 256 karakter Unicode di UTF -8.
- Karakter yang diizinkan adalah huruf, angka, spasi yang dapat direpresentasikan dalam UTF -8, dan karakter berikut: _.:/= + - @.
- Kunci tanda tidak bisa berupa string kosong. Nilai tag dapat berupa string kosong, tetapi bukan nol.
- Kunci dan nilai tanda peka huruf besar-kecil.
- Jangan gunakan `AWS:` atau kombinasi huruf besar atau kecil seperti awalan untuk kunci atau nilai. Ini hanya disediakan untuk AWS gunakan.

## Bekerja dengan tag menggunakan AWS CLI dan Amazon Tanpa EMR Server API

Gunakan yang berikut AWS CLI perintah atau API operasi Amazon EMR Tanpa Server untuk menambahkan, memperbarui, mencantumkan, dan menghapus tag untuk sumber daya Anda.

| Sumber daya | Mendukung tanda | Penyebaran tanda Support |
| --- | --- | --- |
| Menambahkan atau menimpa satu tanda atau lebih | `tag-resource` | `TagResource` |
| Membuat daftar tanda untuk sumber daya | `list-tags-for-resource` | `ListTagsForResource` |
| Menghapus satu tanda atau lebih | `untag-resource` | `UntagResource` |

Contoh berikut menunjukkan cara menandai atau menghapus tag sumber daya menggunakan AWS CLI.

Menandai aplikasi yang sudah ada

Perintah berikut menandai aplikasi yang ada.

```
aws emr-serverless tag-resource --resource-arn resource_ARN --tags team=devs
```

Membatalkan tag aplikasi yang sudah ada

Perintah berikut menghapus tag dari aplikasi yang ada.

```
aws emr-serverless untag-resource --resource-arn resource_ARN --tag-keys tag_key
```

Daftar tag untuk sumber daya

Perintah berikut membuat daftar tanda yang terkait dengan sumber daya yang ada.

```
aws emr-serverless list-tags-for-resource --resource-arn resource_ARN
```

# Tutorial untuk EMR Tanpa Server

Bagian ini menjelaskan kasus penggunaan umum saat Anda bekerja dengan aplikasi EMR Tanpa Server.

Topik



## Menggunakan Java 17 dengan Amazon Tanpa EMR Server

Dengan Amazon EMR merilis 6.11.0 dan yang lebih tinggi, Anda dapat mengonfigurasi pekerjaan Spark EMR Tanpa Server untuk menggunakan runtime Java 17 untuk Java Virtual Machine (). JVM Gunakan salah satu metode berikut untuk mengkonfigurasi Spark dengan Java 17.

### `JAVA_HOME`

Untuk mengganti JVM pengaturan untuk EMR Serverless 6.11.0 dan yang lebih tinggi, Anda dapat menyediakan `JAVA_HOME` pengaturan ke klasifikasi dan lingkungannya. `spark.emr-serverless.driverEnv` `spark.executorEnv`

x86_64

Tetapkan properti yang diperlukan untuk menentukan Java 17 sebagai `JAVA_HOME` konfigurasi untuk driver dan pelaksana Spark:

```
--conf spark.emr-serverless.driverEnv.JAVA_HOME=/usr/lib/jvm/java-17-amazon-
corretto.x86_64/
--conf spark.executorEnv.JAVA_HOME=/usr/lib/jvm/java-17-amazon-corretto.x86_64/
```

arm_64

Tetapkan properti yang diperlukan untuk menentukan Java 17 sebagai `JAVA_HOME` konfigurasi untuk driver dan pelaksana Spark:

```
--conf spark.emr-serverless.driverEnv.JAVA_HOME=/usr/lib/jvm/java-17-amazon-
corretto.aarch64/
--conf spark.executorEnv.JAVA_HOME=/usr/lib/jvm/java-17-amazon-corretto.aarch64/
```

## `spark-defaults`

Atau, Anda dapat menentukan Java 17 dalam `spark-defaults` klasifikasi untuk mengganti JVM pengaturan untuk EMR Tanpa Server 6.11.0 dan yang lebih tinggi.

x86_64

Tentukan Java 17 dalam `spark-defaults` klasifikasi:

```
{
"applicationConfiguration": [
         {
            "classification": "spark-defaults",
            "properties": {
               "spark.emr-serverless.driverEnv.JAVA_HOME" : "/usr/lib/jvm/java-17-
amazon-corretto.x86_64/",
               "spark.executorEnv.JAVA_HOME": "/usr/lib/jvm/java-17-amazon-
corretto.x86_64/"
            }
         }
      ]
}
```

arm_64

Tentukan Java 17 dalam `spark-defaults` klasifikasi:

```
{
```

```
"applicationConfiguration": [
         {
            "classification": "spark-defaults",
            "properties": {
               "spark.emr-serverless.driverEnv.JAVA_HOME" : "/usr/lib/jvm/java-17-
amazon-corretto.aarch64/",
               "spark.executorEnv.JAVA_HOME": "/usr/lib/jvm/java-17-amazon-
corretto.aarch64/"
            }
         }
      ]
}
```

# Menggunakan Apache Hudi dengan Tanpa Server EMR

Untuk menggunakan Apache Hudi dengan aplikasi Tanpa Server EMR

1. Setel properti Spark yang diperlukan dalam menjalankan pekerjaan Spark yang sesuai.

   ```
   spark.jars=/usr/lib/hudi/hudi-spark-bundle.jar
   spark.serializer=org.apache.spark.serializer.KryoSerializer
   ```

2. Untuk menyinkronkan tabel Hudi ke katalog yang dikonfigurasi, tentukan salah satu AWS Glue Data Catalog sebagai metastore Anda, atau konfigurasikan metastore eksternal. EMRDukungan tanpa server `hms` sebagai mode sinkronisasi untuk tabel Hive untuk beban kerja Hudi. EMRTanpa server mengaktifkan properti ini sebagai default. Untuk mempelajari lebih lanjut tentang cara mengatur metastore Anda, lihat. Konfigurasi metastore

   > **Important**
   >
   > EMRTanpa server tidak mendukung `HIVEQL` atau `JDBC` sebagai opsi mode sinkronisasi untuk tabel Hive untuk menangani beban kerja Hudi. Untuk mempelajari lebih lanjut, lihat Mode sinkronisasi.

   Saat Anda menggunakan AWS Glue Data Catalog sebagai metastore Anda, Anda dapat menentukan properti konfigurasi berikut untuk pekerjaan Hudi Anda.

   ```
   --conf spark.jars=/usr/lib/hudi/hudi-spark-bundle.jar,
   ```

```
--conf spark.serializer=org.apache.spark.serializer.KryoSerializer,
--conf
 spark.hadoop.hive.metastore.client.factory.class=com.amazonaws.glue.catalog.metastore.AWSG
```

Untuk mempelajari lebih lanjut tentang rilis Amazon Apache HudiEMR, lihat Riwayat rilis Hudi.

## Menggunakan Apache Iceberg dengan Tanpa Server EMR

Untuk menggunakan Apache Iceberg dengan aplikasi Tanpa Server EMR

1. Setel properti Spark yang diperlukan dalam menjalankan pekerjaan Spark yang sesuai.

   ```
   spark.jars=/usr/share/aws/iceberg/lib/iceberg-spark3-runtime.jar
   ```

2. Tentukan salah satu AWS Glue Data Catalog sebagai metastore Anda atau konfigurasikan metastore eksternal. Untuk mempelajari lebih lanjut tentang pengaturan metastore Anda, lihat. Konfigurasi metastore

   Konfigurasikan properti metastore yang ingin Anda gunakan untuk Iceberg. Misalnya, jika Anda ingin menggunakan AWS Glue Data Catalog, atur properti berikut dalam konfigurasi aplikasi.

   ```
   spark.sql.catalog.dev.warehouse=s3://DOC-EXAMPLE-BUCKET/EXAMPLE-PREFIX/
   spark.sql.extensions=org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions
   spark.sql.catalog.dev=org.apache.iceberg.spark.SparkCatalog
   spark.sql.catalog.dev.catalog-impl=org.apache.iceberg.aws.glue.GlueCatalog
   spark.hadoop.hive.metastore.client.factory.class=com.amazonaws.glue.catalog.metastore.AWSGl
   ```

   Saat Anda menggunakan AWS Glue Data Catalog sebagai metastore Anda, Anda dapat menentukan properti konfigurasi berikut untuk pekerjaan Iceberg Anda.

   ```
   --conf spark.jars=/usr/share/aws/iceberg/lib/iceberg-spark3-runtime.jar,
   --conf
    spark.sql.extensions=org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions,
   --conf spark.sql.catalog.dev=org.apache.iceberg.spark.SparkCatalog,
   --conf spark.sql.catalog.dev.catalog-impl=org.apache.iceberg.aws.glue.GlueCatalog,
   --conf spark.sql.catalog.dev.warehouse=s3://DOC-EXAMPLE-BUCKET/EXAMPLE-PREFIX/
   --conf
    spark.hadoop.hive.metastore.client.factory.class=com.amazonaws.glue.catalog.metastore.AWSG
   ```

Untuk mempelajari lebih lanjut tentang rilis Apache Iceberg dari AmazonEMR, lihat Riwayat rilis Iceberg.

# Menggunakan pustaka Python dengan Tanpa Server EMR

Saat Anda menjalankan PySpark pekerjaan di aplikasi Amazon EMR Tanpa Server, Anda dapat mengemas berbagai pustaka Python sebagai dependensi. Untuk melakukan ini, Anda dapat menggunakan fitur Python asli, membangun lingkungan virtual, atau langsung mengkonfigurasi PySpark pekerjaan Anda untuk menggunakan pustaka Python. Halaman ini mencakup setiap pendekatan.

## Menggunakan fitur Python asli

Saat Anda mengatur konfigurasi berikut, Anda dapat menggunakan PySpark untuk mengunggah file Python (), paket Python zip (`.py`), dan file `.egg` Egg `.zip` () ke pelaksana Spark.

```
--conf spark.submit.pyFiles=s3://DOC-EXAMPLE-BUCKET/EXAMPLE-PREFIX/<.py|.egg|.zip file>
```

Untuk detail selengkapnya tentang cara menggunakan lingkungan virtual Python untuk PySpark pekerjaan, lihat Menggunakan Fitur PySpark Asli.

## Membangun lingkungan virtual Python

Untuk mengemas beberapa pustaka Python untuk suatu PySpark pekerjaan, Anda dapat membuat lingkungan virtual Python yang terisolasi.

1. Untuk membangun lingkungan virtual Python, gunakan perintah berikut. Contoh yang ditampilkan menginstal paket `scipy` dan `matplotlib` ke dalam paket lingkungan virtual dan menyalin arsip ke lokasi Amazon S3.

   > **Important**
   >
   > Anda harus menjalankan perintah berikut di lingkungan Amazon Linux 2 yang serupa dengan versi Python yang sama seperti yang Anda gunakan di EMR Tanpa Server, yaitu, Python 3.7.10 untuk Amazon rilis 6.6.0. EMR Anda dapat menemukan contoh Dockerfile di repositori Sampel EMRTanpa Server. GitHub

```
# initialize a python virtual environment
python3 -m venv pyspark_venvsource
source pyspark_venvsource/bin/activate

# optionally, ensure pip is up-to-date
pip3 install --upgrade pip

# install the python packages
pip3 install scipy
pip3 install matplotlib

# package the virtual environment into an archive
pip3 install venv-pack
venv-pack -f -o pyspark_venv.tar.gz

# copy the archive to an S3 location
aws s3 cp pyspark_venv.tar.gz s3://DOC-EXAMPLE-BUCKET/EXAMPLE-PREFIX/

# optionally, remove the virtual environment directory
rm -fr pyspark_venvsource
```

2. Kirimkan pekerjaan Spark dengan properti Anda yang disetel untuk menggunakan lingkungan virtual Python.

```
--conf spark.archives=s3://DOC-EXAMPLE-BUCKET/EXAMPLE-PREFIX/
pyspark_venv.tar.gz#environment
--conf spark.emr-serverless.driverEnv.PYSPARK_DRIVER_PYTHON=./environment/bin/
python
--conf spark.emr-serverless.driverEnv.PYSPARK_PYTHON=./environment/bin/python
--conf spark.executorEnv.PYSPARK_PYTHON=./environment/bin/python
```

Perhatikan bahwa jika Anda tidak mengganti biner Python asli, konfigurasi kedua dalam urutan pengaturan sebelumnya adalah. `--conf spark.executorEnv.PYSPARK_PYTHON=python`

Untuk informasi lebih lanjut tentang cara menggunakan lingkungan virtual Python untuk PySpark pekerjaan, lihat Menggunakan Virtualenv. Untuk contoh lebih lanjut tentang cara mengirimkan pekerjaan Spark, lihatLowongan kerja Spark.

## Mengkonfigurasi PySpark pekerjaan untuk menggunakan pustaka Python

Dengan Amazon EMR merilis 6.12.0 dan yang lebih tinggi, Anda dapat langsung mengonfigurasi PySpark pekerjaan EMR Tanpa Server untuk menggunakan pustaka Python ilmu data populer seperti panda,, dan tanpa pengaturan tambahan apa pun. NumPyPyArrow

Contoh berikut menunjukkan cara mengemas setiap pustaka Python untuk suatu PySpark pekerjaan.

NumPy

NumPy adalah perpustakaan Python untuk komputasi ilmiah yang menawarkan array multidimensi dan operasi untuk matematika, penyortiran, simulasi acak, dan statistik dasar. Untuk menggunakan NumPy, jalankan perintah berikut:

```
import numpy
```

pandas

panda adalah pustaka Python yang dibangun di atasnya. NumPy Perpustakaan panda menyediakan para ilmuwan data dengan struktur DataFramedata dan alat analisis data. Untuk menggunakan panda, jalankan perintah berikut:

```
import pandas
```

PyArrow

PyArrow adalah pustaka Python yang mengelola data kolumnar dalam memori untuk meningkatkan kinerja pekerjaan. PyArrow didasarkan pada spesifikasi pengembangan lintas bahasa Apache Arrow, yang merupakan cara standar untuk mewakili dan bertukar data dalam format kolumnar. Untuk menggunakan PyArrow, jalankan perintah berikut:

```
import pyarrow
```

# Menggunakan versi Python yang berbeda dengan Tanpa Server EMR

Selain kasus penggunaan diMenggunakan pustaka Python dengan Tanpa Server EMR, Anda juga dapat menggunakan lingkungan virtual Python untuk bekerja dengan versi Python yang berbeda dari

versi yang dikemas dalam EMR rilis Amazon untuk aplikasi Amazon Tanpa Server Anda. EMR Untuk melakukan ini, Anda harus membangun lingkungan virtual Python dengan versi Python yang ingin Anda gunakan.

Untuk mengirimkan pekerjaan dari lingkungan virtual Python

1. Bangun lingkungan virtual Anda dengan perintah dalam contoh berikut. Contoh ini menginstal Python 3.9.9 ke dalam paket lingkungan virtual dan menyalin arsip ke lokasi Amazon S3.

> **Important**
>
> Jika Anda menggunakan Amazon EMR merilis 7.0.0 dan yang lebih tinggi, Anda harus menjalankan perintah Anda di lingkungan Amazon Linux 2023 yang mirip dengan yang Anda gunakan untuk aplikasi Tanpa Server AndaEMR.
> Jika Anda menggunakan rilis 6.15.0 atau lebih rendah, Anda harus menjalankan perintah berikut di lingkungan Amazon Linux 2 yang serupa.

```
# install Python 3.9.9 and activate the venv
yum install -y gcc openssl-devel bzip2-devel libffi-devel tar gzip wget make
wget https://www.python.org/ftp/python/3.9.9/Python-3.9.9.tgz && \
tar xzf Python-3.9.9.tgz && cd Python-3.9.9 && \
./configure --enable-optimizations && \
make altinstall

# create python venv with Python 3.9.9
python3.9 -m venv pyspark_venv_python_3.9.9 --copies
source pyspark_venv_python_3.9.9/bin/activate

# copy system python3 libraries to venv
cp -r /usr/local/lib/python3.9/* ./pyspark_venv_python_3.9.9/lib/python3.9/

# package venv to archive.
# **Note** that you have to supply --python-prefix option
# to make sure python starts with the path where your
# copied libraries are present.
# Copying the python binary to the "environment" directory.
pip3 install venv-pack
venv-pack -f -o pyspark_venv_python_3.9.9.tar.gz --python-prefix /home/hadoop/
environment
```

```
# stage the archive in S3
aws s3 cp pyspark_venv_python_3.9.9.tar.gz s3://<path>

# optionally, remove the virtual environment directory
rm -fr pyspark_venv_python_3.9.9
```

2. Atur properti Anda untuk menggunakan lingkungan virtual Python dan kirimkan pekerjaan Spark.

```
# note that the archive suffix "environment" is the same as the directory where you
 copied the Python binary.
--conf spark.archives=s3://DOC-EXAMPLE-BUCKET/EXAMPLE-PREFIX/
pyspark_venv_python_3.9.9.tar.gz#environment
--conf spark.emr-serverless.driverEnv.PYSPARK_DRIVER_PYTHON=./environment/bin/
python
--conf spark.emr-serverless.driverEnv.PYSPARK_PYTHON=./environment/bin/python
--conf spark.executorEnv.PYSPARK_PYTHON=./environment/bin/python
```

Untuk informasi lebih lanjut tentang cara menggunakan lingkungan virtual Python untuk PySpark pekerjaan, lihat Menggunakan Virtualenv. Untuk contoh lebih lanjut tentang cara mengirimkan pekerjaan Spark, lihatLowongan kerja Spark.

# Menggunakan Delta Lake OSS dengan Serverless EMR

## Amazon EMR versi 6.9.0 dan lebih tinggi

> **Note**
>
> Amazon EMR 7.0.0 dan yang lebih tinggi menggunakan Delta Lake 3.0.0, yang mengganti nama file menjadi. `delta-core.jar` `delta-spark.jar` Jika Anda menggunakan Amazon EMR 7.0.0 atau yang lebih tinggi, pastikan untuk menentukan `delta-spark.jar` dalam konfigurasi Anda.

Amazon EMR 6.9.0 dan yang lebih tinggi termasuk Delta Lake, jadi Anda tidak perlu lagi mengemas Delta Lake sendiri atau memberikan `--packages` bendera dengan pekerjaan Tanpa Server Anda. EMR

1. Saat Anda mengirimkan pekerjaan EMR Tanpa Server, pastikan Anda memiliki properti konfigurasi berikut dan sertakan parameter berikut di `sparkSubmitParameters` bidang.

```
--conf spark.jars=/usr/share/aws/delta/lib/delta-core.jar,/usr/share/aws/delta/lib/
delta-storage.jar
    --conf spark.sql.extensions=io.delta.sql.DeltaSparkSessionExtension
    --conf
 spark.sql.catalog.spark_catalog=org.apache.spark.sql.delta.catalog.DeltaCatalog
```

2. Buat lokal `delta_sample.py` untuk menguji pembuatan dan membaca tabel Delta.

```
# delta_sample.py
    from pyspark.sql import SparkSession

    import uuid

    url = "s3://DOC-EXAMPLE-BUCKET/delta-lake/output/%s/" % str(uuid.uuid4())
    spark = SparkSession.builder.appName("DeltaSample").getOrCreate()

    ## creates a Delta table and outputs to target S3 bucket
    spark.range(5).write.format("delta").save(url)

    ## reads a Delta table and outputs to target S3 bucket
    spark.read.format("delta").load(url).show
```

3. Menggunakan AWS CLI, unggah `delta_sample.py` file ke bucket Amazon S3 Anda. Kemudian gunakan `start-job-run` perintah untuk mengirimkan pekerjaan ke aplikasi EMR Tanpa Server yang ada.

```
aws s3 cp delta_sample.py s3://DOC-EXAMPLE-BUCKET/code/

    aws emr-serverless start-job-run \
        --application-id application-id \
        --execution-role-arn job-role-arn \
        --name emr-delta \
        --job-driver '{
            "sparkSubmit": {
                "entryPoint": "s3://DOC-EXAMPLE-BUCKET/code/delta_sample.py",
                "sparkSubmitParameters": "--conf spark.jars=/usr/share/
aws/delta/lib/delta-core.jar,/usr/share/aws/delta/lib/delta-storage.jar --
conf spark.sql.extensions=io.delta.sql.DeltaSparkSessionExtension --conf
 spark.sql.catalog.spark_catalog=org.apache.spark.sql.delta.catalog.DeltaCatalog"
            }
        }'
```

Untuk menggunakan pustaka Python dengan Delta Lake, Anda dapat menambahkan `delta-core` pustaka dengan mengemasnya sebagai dependensi atau dengan menggunakannya sebagai gambar khusus.

Atau, Anda dapat menggunakan `SparkContext.addPyFile` untuk menambahkan pustaka Python dari file: `delta-core` JAR

```
import glob
from pyspark.sql import SparkSession

spark = SparkSession.builder.getOrCreate()
spark.sparkContext.addPyFile(glob.glob("/usr/share/aws/delta/lib/delta-core_*.jar")[0])
```

## Amazon EMR versi 6.8.0 dan lebih rendah

Jika Anda menggunakan Amazon EMR 6.8.0 atau yang lebih rendah, ikuti langkah-langkah berikut untuk menggunakan Delta Lake OSS dengan aplikasi Tanpa Server AndaEMR.

1. Untuk membuat Delta Lake versi open source yang kompatibel dengan versi Spark di aplikasi Amazon EMR Serverless Anda, navigasikan ke Delta GitHub dan ikuti petunjuknya.
2. Unggah perpustakaan Delta Lake ke bucket Amazon S3 di Akun AWS.
3. Saat Anda mengirimkan pekerjaan EMR Tanpa Server dalam konfigurasi aplikasi, sertakan JAR file Delta Lake yang sekarang ada di bucket Anda.

   ```
   --conf spark.jars=s3://DOC-EXAMPLE-BUCKET/jars/delta-core_2.12-1.1.0.jar
   ```

4. Untuk memastikan bahwa Anda dapat membaca dan menulis dari tabel Delta, jalankan PySpark tes sampel.

   ```
   from pyspark import SparkConf, SparkContext
       from pyspark.sql import HiveContext, SparkSession

       import uuid

       conf = SparkConf()
       sc = SparkContext(conf=conf)
       sqlContext = HiveContext(sc)

       url = "s3://DOC-EXAMPLE-BUCKET/delta-lake/output/1.0.1/%s/" % str(uuid.uuid4())
   ```

```
    ## creates a Delta table and outputs to target S3 bucket
    session.range(5).write.format("delta").save(url)

    ## reads a Delta table and outputs to target S3 bucket
    session.read.format("delta").load(url).show
```

# Mengirimkan pekerjaan Tanpa EMR Server dari Airflow

Penyedia Amazon di Apache Airflow menyediakan operator Tanpa EMR Server. Untuk informasi selengkapnya tentang operator, lihat Operator EMRTanpa Server Amazon di dokumentasi Apache Airflow.

Anda dapat menggunakan `EmrServerlessCreateApplicationOperator` untuk membuat aplikasi Spark atau Hive. Anda juga dapat menggunakan `EmrServerlessStartJobOperator` untuk memulai satu atau lebih pekerjaan dengan aplikasi baru Anda.

Untuk menggunakan operator dengan Amazon Managed Workflows for Apache Airflow (MWAA) dengan Airflow 2.2.2, tambahkan baris berikut ke file Anda dan perbarui MWAA lingkungan Anda untuk menggunakan `requirements.txt` file baru.

```
apache-airflow-providers-amazon==6.0.0
boto3>=1.23.9
```

Perhatikan bahwa dukungan EMR Tanpa Server ditambahkan untuk merilis 5.0.0 dari penyedia Amazon. Rilis 6.0.0 adalah versi terakhir yang kompatibel dengan Airflow 2.2.2. Anda dapat menggunakan versi yang lebih baru dengan Airflow 2.4.3 aktif. MWAA

Contoh singkat berikut menunjukkan cara membuat aplikasi, menjalankan beberapa pekerjaan Spark, dan kemudian menghentikan aplikasi. Contoh lengkap tersedia di repositori Sampel EMR GitHub Tanpa Server. Untuk detail `sparkSubmit` konfigurasi tambahan, lihatLowongan kerja Spark.

```
from datetime import datetime

from airflow import DAG
from airflow.providers.amazon.aws.operators.emr import (
    EmrServerlessCreateApplicationOperator,
    EmrServerlessStartJobOperator,
    EmrServerlessDeleteApplicationOperator,
)
```

```
# Replace these with your correct values
JOB_ROLE_ARN = "arn:aws:iam::account-id:role/emr_serverless_default_role"
S3_LOGS_BUCKET = "DOC-EXAMPLE-BUCKET"

DEFAULT_MONITORING_CONFIG = {
    "monitoringConfiguration": {
        "s3MonitoringConfiguration": {"logUri": f"s3://DOC-EXAMPLE-BUCKET/logs/"}
    },
}

with DAG(
    dag_id="example_endtoend_emr_serverless_job",
    schedule_interval=None,
    start_date=datetime(2021, 1, 1),
    tags=["example"],
    catchup=False,
) as dag:
    create_app = EmrServerlessCreateApplicationOperator(
        task_id="create_spark_app",
        job_type="SPARK",
        release_label="emr-6.7.0",
        config={"name": "airflow-test"},
    )

    application_id = create_app.output

    job1 = EmrServerlessStartJobOperator(
        task_id="start_job_1",
        application_id=application_id,
        execution_role_arn=JOB_ROLE_ARN,
        job_driver={
            "sparkSubmit": {
                "entryPoint": "local:///usr/lib/spark/examples/src/main/python/
pi_fail.py",
            }
        },
        configuration_overrides=DEFAULT_MONITORING_CONFIG,
    )

    job2 = EmrServerlessStartJobOperator(
        task_id="start_job_2",
        application_id=application_id,
        execution_role_arn=JOB_ROLE_ARN,
        job_driver={
```

```
            "sparkSubmit": {
                "entryPoint": "local:///usr/lib/spark/examples/src/main/python/pi.py",
                "entryPointArguments": ["1000"]
            }
        },
        configuration_overrides=DEFAULT_MONITORING_CONFIG,
    )

    delete_app = EmrServerlessDeleteApplicationOperator(
        task_id="delete_app",
        application_id=application_id,
        trigger_rule="all_done",
    )

    (create_app >> [job1, job2] >> delete_app)
```

# Menggunakan fungsi yang ditentukan pengguna Hive dengan Tanpa Server EMR

Hive user-defined functions (UDFs) memungkinkan Anda membuat fungsi kustom untuk memproses catatan atau grup rekaman. Dalam tutorial ini, Anda akan menggunakan contoh UDF dengan aplikasi Amazon EMR Serverless yang sudah ada sebelumnya untuk menjalankan pekerjaan yang menghasilkan hasil kueri. Untuk mempelajari cara menyiapkan aplikasi, lihatMemulai dengan Amazon Tanpa EMR Server.

Untuk menggunakan UDF dengan Tanpa EMR Server

1. Arahkan ke GitHubuntuk sampelUDF. Kloning repo dan beralih ke cabang git yang ingin Anda gunakan. Perbarui `pom.xml` file `maven-compiler-plugin` dalam repositori untuk memiliki sumber. Juga perbarui konfigurasi versi java target ke`1.8`. Jalankan `mvn package -DskipTests` untuk membuat JAR file yang berisi sampel AndaUDFs.
2. Setelah Anda membuat JAR file, unggah ke bucket S3 Anda dengan perintah berikut.

   ```
   aws s3 cp brickhouse-0.8.2-JS.jar s3://DOC-EXAMPLE-BUCKET/jars/
   ```

3. Buat file contoh untuk menggunakan salah satu UDF fungsi sampel. Simpan kueri ini sebagai `udf_example.q` dan unggah ke bucket S3 Anda.

   ```
   add jar s3://DOC-EXAMPLE-BUCKET/jars/brickhouse-0.8.2-JS.jar;
   ```

```
CREATE TEMPORARY FUNCTION from_json AS 'brickhouse.udf.json.FromJsonUDF';
select from_json('{"key1":[0,1,2], "key2":[3,4,5,6], "key3":[7,8,9]}', map("",
 array(cast(0 as int))));
select from_json('{"key1":[0,1,2], "key2":[3,4,5,6], "key3":[7,8,9]}', map("",
 array(cast(0 as int))))["key1"][2];
```

4. Kirimkan pekerjaan Hive berikut.

```
aws emr-serverless start-job-run \
  --application-id application-id \
  --execution-role-arn job-role-arn \
  --job-driver '{
    "hive": {
        "query": "s3://DOC-EXAMPLE-BUCKET/queries/udf_example.q",
        "parameters": "--hiveconf hive.exec.scratchdir=s3://DOC-EXAMPLE-BUCKET/emr-
serverless-hive/scratch --hiveconf hive.metastore.warehouse.dir=s3://'$BUCKET'/emr-
serverless-hive/warehouse"
    }
}' --configuration-overrides '{
    "applicationConfiguration": [{
        "classification": "hive-site",
        "properties": {
            "hive.driver.cores": "2",
            "hive.driver.memory": "6G"
        }
    }],
    "monitoringConfiguration": {
        "s3MonitoringConfiguration": {
            "logUri": "s3://DOC-EXAMPLE-BUCKET/logs/"
        }
    }
}'
```

5. Gunakan `get-job-run` perintah untuk memeriksa status pekerjaan Anda. Tunggu sampai negara berubah`SUCCESS`.

```
aws emr-serverless get-job-run --application-id application-id --job-run-id job-id
```

6. Unduh file output dengan perintah berikut.

```
aws s3 cp --recursive s3://DOC-EXAMPLE-BUCKET/logs/applications/application-id/
jobs/job-id/HIVE_DRIVER/ .
```

`stdout.gz`File tersebut menyerupai yang berikut ini.

```
{"key1":[0,1,2],"key2":[3,4,5,6],"key3":[7,8,9]}
2
```

# Menggunakan gambar kustom dengan EMR Serverless

Topik

- Gunakan versi Python khusus
- Gunakan versi Java kustom
- Membangun citra ilmu data
- Memproses data geospasial dengan Apache Sedona

## Gunakan versi Python khusus

Anda dapat membuat gambar khusus untuk menggunakan versi Python yang berbeda. Untuk menggunakan Python versi 3.10 untuk pekerjaan Spark, misalnya, jalankan perintah berikut:

```
FROM public.ecr.aws/emr-serverless/spark/emr-6.9.0:latest

USER root

# install python 3
RUN yum install -y gcc openssl-devel bzip2-devel libffi-devel tar gzip wget make
RUN wget https://www.python.org/ftp/python/3.10.0/Python-3.10.0.tgz && \
tar xzf Python-3.10.0.tgz && cd Python-3.10.0 && \
./configure --enable-optimizations && \
make altinstall

# EMRS will run the image as hadoop
USER hadoop:hadoop
```

Sebelum Anda mengirimkan pekerjaan Spark, atur properti Anda untuk menggunakan lingkungan virtual Python, sebagai berikut.

```
--conf spark.emr-serverless.driverEnv.PYSPARK_DRIVER_PYTHON=/usr/local/bin/python3.10
```

```
--conf spark.emr-serverless.driverEnv.PYSPARK_PYTHON=/usr/local/bin/python3.10
--conf spark.executorEnv.PYSPARK_PYTHON=/usr/local/bin/python3.10
```

## Gunakan versi Java kustom

Contoh berikut menunjukkan cara membuat gambar kustom untuk menggunakan Java 11 untuk pekerjaan Spark Anda.

```
FROM public.ecr.aws/emr-serverless/spark/emr-6.9.0:latest

USER root

# install JDK 11
RUN sudo amazon-linux-extras install java-openjdk11

# EMRS will run the image as hadoop
USER hadoop:hadoop
```

Sebelum Anda mengirimkan pekerjaan Spark, atur properti Spark untuk menggunakan Java 11, sebagai berikut.

```
--conf spark.executorEnv.JAVA_HOME=/usr/lib/jvm/java-11-
openjdk-11.0.16.0.8-1.amzn2.0.1.x86_64
--conf spark.emr-serverless.driverEnv.JAVA_HOME=/usr/lib/jvm/java-11-
openjdk-11.0.16.0.8-
```

## Membangun citra ilmu data

Contoh berikut menunjukkan cara memasukkan paket Python ilmu data umum, seperti Pandas dan. NumPy

```
FROM public.ecr.aws/emr-serverless/spark/emr-6.9.0:latest

USER root

# python packages
RUN pip3 install boto3 pandas numpy
RUN pip3 install -U scikit-learn==0.23.2 scipy
RUN pip3 install sk-dist
RUN pip3 install xgboost
```

```
# EMR Serverless will run the image as hadoop
USER hadoop:hadoop
```

### Memproses data geospasial dengan Apache Sedona

Contoh berikut menunjukkan bagaimana membangun gambar untuk menyertakan Apache Sedona untuk pemrosesan geospasial.

```
FROM public.ecr.aws/emr-serverless/spark/emr-6.9.0:latest

USER root

RUN yum install -y wget
RUN wget https://repo1.maven.org/maven2/org/apache/sedona/sedona-core-3.0_2.12/1.3.0-
incubating/sedona-core-3.0_2.12-1.3.0-incubating.jar -P /usr/lib/spark/jars/
RUN pip3 install apache-sedona

# EMRS will run the image as hadoop
USER hadoop:hadoop
```

## Menggunakan integrasi Amazon Redshift untuk Apache Spark di Amazon Tanpa Server EMR

Dengan EMR rilis Amazon 6.9.0 dan yang lebih baru, setiap gambar rilis menyertakan konektor antara Apache Spark dan Amazon Redshift. Dengan konektor ini, Anda dapat menggunakan Spark di Amazon EMR Serverless untuk memproses data yang disimpan di Amazon Redshift. Integrasi ini didasarkan pada konektor `spark-redshift` open-source. Untuk Amazon EMR Tanpa Server, integrasi Amazon Redshift untuk Apache Spark disertakan sebagai integrasi asli.

Topik

- Meluncurkan aplikasi Spark dengan integrasi Amazon Redshift untuk Apache Spark
- Mengautentikasi dengan integrasi Amazon Redshift untuk Apache Spark
- Membaca dan menulis dari dan ke Amazon Redshift
- Pertimbangan dan batasan saat menggunakan konektor Spark

# Meluncurkan aplikasi Spark dengan integrasi Amazon Redshift untuk Apache Spark

Untuk menggunakan integrasi dengan EMR Serverless 6.9.0, Anda harus meneruskan dependensi Spark-Redshift yang diperlukan dengan pekerjaan Spark Anda. Gunakan `--jars` untuk menyertakan pustaka terkait konektor Redshift. Untuk melihat lokasi file lain yang didukung oleh `--jars` opsi, lihat bagian Advanced Dependency Management dari dokumentasi Apache Spark.

- `spark-redshift.jar`
- `spark-avro.jar`
- `RedshiftJDBC.jar`
- `minimal-json.jar`

Amazon EMR merilis 6.10.0 dan yang lebih tinggi tidak memerlukan `minimal-json.jar` dependensi, dan secara otomatis menginstal dependensi lain ke setiap cluster secara default. Contoh berikut menunjukkan cara meluncurkan aplikasi Spark dengan integrasi Amazon Redshift untuk Apache Spark.

Amazon EMR 6.10.0 +

Luncurkan pekerjaan Spark di Amazon EMR Tanpa Server dengan integrasi Amazon Redshift untuk Apache Spark pada rilis Tanpa Server 6.10.0 dan yang lebih tinggi. EMR

```
spark-submit my_script.py
```

Amazon EMR 6.9.0

Untuk meluncurkan pekerjaan Spark di Amazon EMR Tanpa Server dengan integrasi Amazon Redshift untuk Apache Spark pada rilis EMR Tanpa Server 6.9.0, gunakan opsi seperti yang ditunjukkan pada contoh berikut. `--jars` Perhatikan bahwa jalur yang tercantum dengan `--jars` opsi adalah jalur default untuk JAR file.

```
--jars
    /usr/share/aws/redshift/jdbc/RedshiftJDBC.jar,
    /usr/share/aws/redshift/spark-redshift/lib/spark-redshift.jar,
    /usr/share/aws/redshift/spark-redshift/lib/spark-avro.jar,
    /usr/share/aws/redshift/spark-redshift/lib/minimal-json.jar
```

```
spark-submit \
  --jars /usr/share/aws/redshift/jdbc/RedshiftJDBC.jar,/usr/share/aws/redshift/
spark-redshift/lib/spark-redshift.jar,/usr/share/aws/redshift/spark-redshift/lib/
spark-avro.jar,/usr/share/aws/redshift/spark-redshift/lib/minimal-json.jar \
  my_script.py
```

## Mengautentikasi dengan integrasi Amazon Redshift untuk Apache Spark

### Gunakan AWS Secrets Manager untuk mengambil kredensil dan terhubung ke Amazon Redshift

Anda dapat melakukan autentikasi dengan aman ke Amazon Redshift dengan menyimpan kredensialnya di Secrets Manager dan meminta tugas Spark memanggil untuk mengambilnya `GetSecretValue`API:

```
from pyspark.sql import SQLContextimport boto3

sc = # existing SparkContext
sql_context = SQLContext(sc)

secretsmanager_client = boto3.client('secretsmanager',
 region_name=os.getenv('AWS_REGION'))
secret_manager_response = secretsmanager_client.get_secret_value(
    SecretId='string',
    VersionId='string',
    VersionStage='string'
)
username = # get username from secret_manager_response
password = # get password from secret_manager_response
url = "jdbc:redshift://redshifthost:5439/database?user=" + username + "&password="
 + password

# Access to Redshift cluster using Spark
```

### Otentikasi ke Amazon Redshift dengan driver JDBC

Tetapkan nama pengguna dan kata sandi di dalam JDBC URL

Anda dapat mengautentikasi pekerjaan Spark ke cluster Amazon Redshift dengan menentukan nama database Amazon Redshift dan kata sandi di file. JDBC URL

> **Note**
>
> Jika Anda meneruskan kredensi database diURL, siapa pun yang memiliki akses ke juga URL dapat mengakses kredensialnya. Metode ini umumnya tidak disarankan karena ini bukan opsi yang aman.

Jika keamanan tidak menjadi perhatian untuk aplikasi Anda, Anda dapat menggunakan format berikut untuk mengatur nama pengguna dan kata sandi di JDBCURL:

```
jdbc:redshift://redshifthost:5439/database?user=username&password=password
```

## Menggunakan otentikasi IAM berbasis dengan peran eksekusi EMR pekerjaan Amazon Tanpa Server

Dimulai dengan rilis Amazon EMR Tanpa Server 6.9.0, JDBC driver Amazon Redshift 2.1 atau yang lebih tinggi dikemas ke lingkungan. Dengan JDBC driver 2.1 dan yang lebih tinggi, Anda dapat menentukan JDBC URL dan tidak menyertakan nama pengguna dan kata sandi mentah.

Sebagai gantinya, Anda dapat menentukan `jdbc:redshift:iam://` skema. Ini memerintahkan JDBC driver untuk menggunakan peran eksekusi pekerjaan EMR Tanpa Server Anda untuk mengambil kredensi secara otomatis. Lihat Mengonfigurasi JDBC atau ODBC koneksi untuk menggunakan IAM kredensil di Panduan Manajemen Amazon Redshift untuk informasi selengkapnya. Contoh dari ini URL adalah:

```
jdbc:redshift:iam://examplecluster.abc123xyz789.us-west-2.redshift.amazonaws.com:5439/
dev
```

Izin berikut diperlukan untuk peran pelaksanaan pekerjaan Anda ketika kondisi yang disediakan terpenuhi:

| Izin | Kondisi bila diperlukan untuk peran pelaksanaan pekerjaan |
| --- | --- |
| `redshift:GetClusterCredentials` | Diperlukan bagi JDBC pengemudi untuk mengambil kredensil dari Amazon Redshift |

| Izin | Kondisi bila diperlukan untuk peran pelaksanaan pekerjaan |
| --- | --- |
| `redshift:DescribeCluster` | Diperlukan jika Anda menentukan cluster Amazon Redshift dan Wilayah AWS di JDBC URL alih-alih titik akhir |
| `redshift-serverless:GetCredentials` | Diperlukan bagi JDBC driver untuk mengambil kredensil dari Amazon Redshift Tanpa Server |
| `redshift-serverless:GetWorkgroup` | Diperlukan jika Anda menggunakan Amazon Redshift Serverless dan Anda menentukan URL dalam hal nama workgroup dan Wilayah |

## Menghubungkan ke Amazon Redshift dalam yang berbeda VPC

Saat menyiapkan klaster Amazon Redshift atau grup kerja Amazon Redshift Serverless yang disediakan di bawah aVPC, Anda harus mengonfigurasi VPC konektivitas untuk aplikasi Amazon EMR Tanpa Server untuk mengakses sumber daya. Untuk informasi selengkapnya tentang cara mengonfigurasi VPC konektivitas pada aplikasi EMR Tanpa Server, lihat. Mengkonfigurasi akses VPC

- Jika klaster Amazon Redshift atau grup kerja Amazon Redshift Tanpa Server yang disediakan dapat diakses publik, Anda dapat menentukan satu atau beberapa subnet pribadi yang memiliki gateway terpasang saat membuat aplikasi Tanpa Server. NAT EMR
- Jika klaster Amazon Redshift atau grup kerja Tanpa Server Amazon Redshift yang disediakan tidak dapat diakses publik, Anda harus membuat titik akhir VPC terkelola Amazon Redshift untuk klaster Amazon Redshift seperti yang dijelaskan dalam. Mengkonfigurasi akses VPC Atau, Anda dapat membuat grup kerja Amazon Redshift Tanpa Server seperti yang dijelaskan dalam Menghubungkan ke Amazon Redshift Tanpa Server di Panduan Manajemen Pergeseran Merah Amazon. Anda harus mengaitkan klaster atau subgrup Anda ke subnet pribadi yang Anda tentukan saat membuat aplikasi Tanpa EMR Server.

> **Note**
>
> Jika Anda menggunakan otentikasi IAM berbasis, dan subnet pribadi Anda untuk aplikasi EMR Tanpa Server tidak memiliki NAT gateway yang terpasang, maka Anda juga harus

membuat VPC titik akhir pada subnet tersebut untuk Amazon Redshift atau Amazon Redshift Tanpa Server. Dengan cara ini, JDBC pengemudi dapat mengambil kredensialnya.

## Membaca dan menulis dari dan ke Amazon Redshift

Contoh kode berikut digunakan PySpark untuk membaca dan menulis data sampel dari dan ke database Amazon Redshift dengan sumber data API dan dengan Spark. SQL

Data source API

Gunakan PySpark untuk membaca dan menulis data sampel dari dan ke database Amazon Redshift dengan sumber data. API

```
import boto3
from pyspark.sql import SQLContext

sc = # existing SparkContext
sql_context = SQLContext(sc)

url = "jdbc:redshift:iam://redshifthost:5439/database"
aws_iam_role_arn = "arn:aws:iam::account-id:role/role-name"

df = sql_context.read \
    .format("io.github.spark_redshift_community.spark.redshift") \
    .option("url", url) \
    .option("dbtable", "table-name") \
    .option("tempdir", "s3://path/for/temp/data") \
    .option("aws_iam_role", "aws-iam-role-arn") \
    .load()

df.write \
    .format("io.github.spark_redshift_community.spark.redshift") \
    .option("url", url) \
    .option("dbtable", "table-name-copy") \
    .option("tempdir", "s3://path/for/temp/data") \
    .option("aws_iam_role", "aws-iam-role-arn") \
    .mode("error") \
    .save()
```

SparkSQL

Gunakan PySpark untuk membaca dan menulis data sampel dari dan ke database Amazon Redshift dengan Spark. SQL

```
import boto3
import json
import sys
import os
from pyspark.sql import SparkSession

spark = SparkSession \
    .builder \
    .enableHiveSupport() \
    .getOrCreate()

url = "jdbc:redshift:iam://redshifthost:5439/database"
aws_iam_role_arn = "arn:aws:iam::account-id:role/role-name"

bucket = "s3://path/for/temp/data"
tableName = "table-name" # Redshift table name

s = f"""CREATE TABLE IF NOT EXISTS {table-name} (country string, data string)
    USING io.github.spark_redshift_community.spark.redshift
    OPTIONS (dbtable '{table-name}', tempdir '{bucket}', url '{url}', aws_iam_role
 '{aws-iam-role-arn}' ); """

spark.sql(s)

columns = ["country" ,"data"]
data = [("test-country","test-data")]
df = spark.sparkContext.parallelize(data).toDF(columns)

# Insert data into table
df.write.insertInto(table-name, overwrite=False)
df = spark.sql(f"SELECT * FROM {table-name}")
df.show()
```

## Pertimbangan dan batasan saat menggunakan konektor Spark

- Kami menyarankan Anda mengaktifkan JDBC koneksi dari Spark di Amazon EMR ke Amazon Redshift. SSL

- Kami menyarankan Anda mengelola kredensil untuk cluster Amazon Redshift di AWS Secrets Manager sebagai praktik terbaik. Lihat Menggunakan AWS Secrets Manager untuk mengambil kredensil untuk menghubungkan ke Amazon Redshift sebagai contoh.
- Kami menyarankan Anda meneruskan IAM peran dengan parameter `aws_iam_role` untuk parameter autentikasi Amazon Redshift.
- Parameter `tempformat` saat ini tidak mendukung format Parket.
- `tempdir`URIMenunjuk ke lokasi Amazon S3. Direktori temp ini tidak dibersihkan secara otomatis dan karenanya dapat menambah biaya tambahan.
- Pertimbangkan rekomendasi berikut untuk Amazon Redshift:
  - Kami menyarankan Anda memblokir akses publik ke cluster Amazon Redshift.
  - Kami menyarankan Anda mengaktifkan pencatatan audit Amazon Redshift.
  - Kami menyarankan Anda mengaktifkan enkripsi saat istirahat Amazon Redshift.
- Pertimbangkan rekomendasi berikut untuk Amazon S3:
  - Kami menyarankan Anda memblokir akses publik ke bucket Amazon S3.
  - Kami menyarankan Anda menggunakan enkripsi sisi server Amazon S3 untuk mengenkripsi bucket Amazon S3 yang digunakan.
  - Sebaiknya gunakan kebijakan siklus hidup Amazon S3 untuk menentukan aturan retensi bucket Amazon S3.
  - Amazon EMR selalu memverifikasi kode yang diimpor dari sumber terbuka ke dalam gambar. Demi keamanan, kami tidak mendukung metode otentikasi berikut dari Spark ke Amazon S3:
    - Pengaturan AWS kunci akses dalam klasifikasi `hadoop-env` konfigurasi
    - Encoding AWS kunci akses di `tempdir` URI

Untuk informasi selengkapnya tentang penggunaan konektor dan parameter yang didukung, lihat sumber daya berikut:

- Integrasi Amazon Redshift untuk Apache Spark di Panduan Manajemen Amazon Redshift
- Repositori `spark-redshift` komunitas di Github

# Menghubungkan ke DynamoDB dengan Amazon Serverless EMR

Dalam tutorial ini, Anda mengunggah subset data dari United States Board on Geographic Names ke bucket Amazon S3 dan kemudian menggunakan Hive atau Spark di Amazon Tanpa Server untuk menyalin data ke tabel EMR Amazon DynamoDB yang dapat Anda kueri.

## Langkah 1: Unggah data ke bucket Amazon S3

Untuk membuat bucket Amazon S3, ikuti petunjuk dalam Membuat bucket di Panduan Pengguna Amazon Simple Storage Service Console. Ganti referensi *DOC-EXAMPLE-BUCKET* dengan nama bucket yang baru Anda buat. Sekarang aplikasi EMR Tanpa Server Anda siap menjalankan pekerjaan.

1. Unduh arsip data sampel `features.zip` dengan perintah berikut.

```
wget https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/samples/
features.zip
```

2. Ekstrak `features.txt` file dari arsip dan lihat yang pertama beberapa baris dalam file:

```
unzip features.zip
head features.txt
```

Hasilnya akan terlihat mirip dengan yang berikut ini.

```
1535908|Big Run|Stream|WV|38.6370428|-80.8595469|794
875609|Constable Hook|Cape|NJ|40.657881|-74.0990309|7
1217998|Gooseberry Island|Island|RI|41.4534361|-71.3253284|10
26603|Boone Moore Spring|Spring|AZ|34.0895692|-111.410065|3681
1506738|Missouri Flat|Flat|WA|46.7634987|-117.0346113|2605
1181348|Minnow Run|Stream|PA|40.0820178|-79.3800349|1558
1288759|Hunting Creek|Stream|TN|36.343969|-83.8029682|1024
533060|Big Charles Bayou|Bay|LA|29.6046517|-91.9828654|0
829689|Greenwood Creek|Stream|NE|41.596086|-103.0499296|3671
541692|Button Willow Island|Island|LA|31.9579389|-93.0648847|98
```

Bidang di setiap baris di sini menunjukkan pengidentifikasi unik, nama, jenis fitur alami, keadaan, garis lintang dalam derajat, bujur dalam derajat, dan tinggi dalam kaki.

3. Unggah data Anda ke Amazon S3

```
aws s3 cp features.txt s3://DOC-EXAMPLE-BUCKET/features/
```

## Langkah 2: Buat tabel Hive

Gunakan Apache Spark atau Hive untuk membuat tabel Hive baru yang berisi data yang diunggah di Amazon S3.

Spark

Untuk membuat tabel Hive dengan Spark, jalankan perintah berikut.

```
import org.apache.spark.sql.SparkSession

val sparkSession = SparkSession.builder().enableHiveSupport().getOrCreate()

sparkSession.sql("CREATE TABLE hive_features \
    (feature_id BIGINT, \
    feature_name STRING, \
    feature_class STRING, \
    state_alpha STRING, \
    prim_lat_dec DOUBLE, \
    prim_long_dec DOUBLE, \
    elev_in_ft BIGINT) \
    ROW FORMAT DELIMITED \
    FIELDS TERMINATED BY '|' \
    LINES TERMINATED BY '\n' \
    LOCATION 's3://DOC-EXAMPLE_BUCKET/features';")
```

Anda sekarang memiliki tabel Hive terisi dengan data dari file. `features.txt` Untuk memverifikasi bahwa data Anda ada dalam tabel, jalankan SQL kueri Spark seperti yang ditunjukkan pada contoh berikut.

```
sparkSession.sql(
    "SELECT state_alpha, COUNT(*) FROM hive_features GROUP BY state_alpha;")
```

Hive

Untuk membuat tabel Hive dengan Hive, jalankan perintah berikut.

```
CREATE TABLE hive_features
```

```
    (feature_id             BIGINT,
    feature_name            STRING ,
    feature_class           STRING ,
    state_alpha             STRING,
    prim_lat_dec            DOUBLE ,
    prim_long_dec           DOUBLE ,
    elev_in_ft              BIGINT)
    ROW FORMAT DELIMITED
    FIELDS TERMINATED BY '|'
    LINES TERMINATED BY '\n'
    LOCATION 's3://DOC-EXAMPLE-BUCKET/features';
```

Anda sekarang memiliki tabel Hive yang berisi data dari `features.txt` file. Untuk memverifikasi bahwa data Anda ada dalam tabel, jalankan kueri HiveQL, seperti yang ditunjukkan pada contoh berikut.

```
SELECT state_alpha, COUNT(*) FROM hive_features GROUP BY state_alpha;
```

## Langkah 3: Salin data ke DynamoDB

Gunakan Spark atau Hive untuk menyalin data ke tabel DynamoDB baru.

Spark

Untuk menyalin data dari tabel Hive yang Anda buat pada langkah sebelumnya ke DynamoDB, ikuti Langkah 1-3 di Salin data ke DynamoDB. Ini menciptakan tabel DynamoDB baru yang disebut. `Features` Anda kemudian dapat membaca data langsung dari file teks dan menyalinnya ke tabel DynamoDB Anda, seperti contoh berikut menunjukkan.

```
import com.amazonaws.services.dynamodbv2.model.AttributeValue
import org.apache.hadoop.dynamodb.DynamoDBItemWritable
import org.apache.hadoop.dynamodb.read.DynamoDBInputFormat
import org.apache.hadoop.io.Text
import org.apache.hadoop.mapred.JobConf
import org.apache.spark.SparkContext

import scala.collection.JavaConverters._

object EmrServerlessDynamoDbTest {
```

```
    def main(args: Array[String]): Unit = {

        jobConf.set("dynamodb.input.tableName", "Features")
        jobConf.set("dynamodb.output.tableName", "Features")
        jobConf.set("dynamodb.region", "region")

        jobConf.set("mapred.output.format.class",
 "org.apache.hadoop.dynamodb.write.DynamoDBOutputFormat")
        jobConf.set("mapred.input.format.class",
 "org.apache.hadoop.dynamodb.read.DynamoDBInputFormat")

        val rdd = sc.textFile("s3://DOC-EXAMPLE-BUCKET/ddb-connector/")
            .map(row => {
                val line = row.split("\\|")
                val item = new DynamoDBItemWritable()

                val elevInFt = if (line.length > 6) {
                    new AttributeValue().withN(line(6))
                } else {
                    new AttributeValue().withNULL(true)
                }

                item.setItem(Map(
                    "feature_id" -> new AttributeValue().withN(line(0)),
                    "feature_name" -> new AttributeValue(line(1)),
                    "feature_class" -> new AttributeValue(line(2)),
                    "state_alpha" -> new AttributeValue(line(3)),
                    "prim_lat_dec" -> new AttributeValue().withN(line(4)),
                    "prim_long_dec" -> new AttributeValue().withN(line(5)),
                    "elev_in_ft" -> elevInFt)
                    .asJava)
                (new Text(""), item)
        })
        rdd.saveAsHadoopDataset(jobConf)
    }
}
```

### Hive

Untuk menyalin data dari tabel Hive yang Anda buat pada langkah sebelumnya ke DynamoDB, ikuti petunjuk di Salin data ke DynamoDB.

# Langkah 4: Kueri data dari DynamoDB

Gunakan Spark atau Hive untuk menanyakan tabel DynamoDB Anda.

Spark

Untuk kueri data dari tabel DynamoDB yang Anda buat pada langkah sebelumnya, Anda dapat menggunakan SQL Spark atau Spark. MapReduce API

Example — Kueri tabel DynamoDB Anda dengan Spark SQL

SQLKueri Spark berikut mengembalikan daftar semua jenis fitur dalam urutan abjad.

```
val dataFrame = sparkSession.sql("SELECT DISTINCT feature_class \
    FROM ddb_features \
    ORDER BY feature_class;")
```

SQLKueri Spark berikut mengembalikan daftar semua danau yang dimulai dengan huruf M.

```
val dataFrame = sparkSession.sql("SELECT feature_name, state_alpha \
    FROM ddb_features \
    WHERE feature_class = 'Lake' \
    AND feature_name LIKE 'M%' \
    ORDER BY feature_name;")
```

SQLKueri Spark berikut mengembalikan daftar semua status dengan setidaknya tiga fitur yang lebih tinggi dari satu mil.

```
val dataFrame = sparkSession.dql("SELECT state_alpha, feature_class, COUNT(*) \
    FROM ddb_features \
    WHERE elev_in_ft > 5280 \
    GROUP by state_alpha, feature_class \
    HAVING COUNT(*) >= 3 \
    ORDER BY state_alpha, feature_class;")
```

Example — Kueri tabel DynamoDB Anda dengan Spark MapReduce API

MapReduce Query berikut mengembalikan daftar semua jenis fitur dalam urutan abjad.

```
val df = sc.hadoopRDD(jobConf, classOf[DynamoDBInputFormat], classOf[Text],
 classOf[DynamoDBItemWritable])
    .map(pair => (pair._1, pair._2.getItem))
```

```
    .map(pair => pair._2.get("feature_class").getS)
    .distinct()
    .sortBy(value => value)
    .toDF("feature_class")
```

MapReduce Query berikut mengembalikan daftar semua danau yang dimulai dengan huruf M.

```
val df = sc.hadoopRDD(jobConf, classOf[DynamoDBInputFormat], classOf[Text],
 classOf[DynamoDBItemWritable])
    .map(pair => (pair._1, pair._2.getItem))
    .filter(pair => "Lake".equals(pair._2.get("feature_class").getS))
    .filter(pair => pair._2.get("feature_name").getS.startsWith("M"))
    .map(pair => (pair._2.get("feature_name").getS,
 pair._2.get("state_alpha").getS))
    .sortBy(_._1)
    .toDF("feature_name", "state_alpha")
```

MapReduce Kueri berikut mengembalikan daftar semua negara dengan setidaknya tiga fitur yang lebih tinggi dari satu mil.

```
val df = sc.hadoopRDD(jobConf, classOf[DynamoDBInputFormat], classOf[Text],
 classOf[DynamoDBItemWritable])
    .map(pair => pair._2.getItem)
    .filter(pair => pair.get("elev_in_ft").getN != null)
    .filter(pair => Integer.parseInt(pair.get("elev_in_ft").getN) > 5280)
    .groupBy(pair => (pair.get("state_alpha").getS, pair.get("feature_class").getS))
    .filter(pair => pair._2.size >= 3)
    .map(pair => (pair._1._1, pair._1._2, pair._2.size))
    .sortBy(pair => (pair._1, pair._2))
    .toDF("state_alpha", "feature_class", "count")
```

Hive

Untuk kueri data dari tabel DynamoDB yang Anda buat pada langkah sebelumnya, ikuti petunjuk di Kueri data dalam tabel DynamoDB.

## Menyiapkan akses lintas akun

Untuk mengatur akses lintas akun untuk EMR Tanpa Server, selesaikan langkah-langkah berikut. Dalam contoh, `AccountA` adalah akun tempat Anda membuat aplikasi Amazon EMR Tanpa Server, dan `AccountB` merupakan akun tempat Amazon DynamoDB Anda berada.

1. Buat tabel DynamoDB di. `AccountB` Untuk informasi selengkapnya, lihat Langkah 1: Membuat tabel.
2. Buat `Cross-Account-Role-B` IAM peran `AccountB` yang dapat mengakses tabel DynamoDB.

   a. Masuk ke AWS Management Console dan buka IAM konsol di https://console.aws.amazon.com/iam/.

   b. Pilih Peran, dan buat peran baru yang disebut`Cross-Account-Role-B`. Untuk informasi selengkapnya tentang cara membuat IAM peran, lihat Membuat IAM peran di Panduan pengguna.

   c. Buat IAM kebijakan yang memberikan izin untuk mengakses tabel DynamoDB lintas akun. Kemudian lampirkan IAM kebijakan ke`Cross-Account-Role-B`.

      Berikut ini adalah kebijakan yang memberikan akses ke tabel DynamoDB. `CrossAccountTable`

      ```
      {"Version": "2012-10-17",
          "Statement": [
              {"Effect": "Allow",
                  "Action": "dynamodb:*",
                  "Resource": "arn:aws:dynamodb:region:AccountB:table/
      CrossAccountTable"
              }
          ]
      }
      ```

   d. Cara mengedit hubungan kepercayaan untuk peran `Cross-Account-Role-B`.

      Untuk mengonfigurasi hubungan kepercayaan untuk peran tersebut, pilih tab Trust Relationships di IAM konsol untuk peran yang Anda buat di Langkah 2: Cross-Account-Role-B.

      Pilih Edit Trust Relationship lalu tambahkan dokumen kebijakan berikut. Dokumen ini memungkinkan `Job-Execution-Role-A` `AccountA` untuk mengambil `Cross-Account-Role-B` peran ini.

      ```
      {"Version": "2012-10-17",
        "Statement": [
          {"Effect": "Allow",
            "Principal": {"AWS": "arn:aws:iam::AccountA:role/Job-Execution-Role-A"
      ```

```
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

e. Berikan `Job-Execution-Role-A - STS Assume role` izin untuk berasumsi`Cross-Account-Role-B`. `AccountA`

Di IAM konsol untuk Akun AWS `AccountA`, pilih`Job-Execution-Role-A`. Tambahkan pernyataan kebijakan berikut pada `Job-Execution-Role-A` untuk mengizinkan tindakan `AssumeRole` di peran `Cross-Account-Role-B`.

```
{"Version": "2012-10-17",
    "Statement": [
        {"Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::AccountB:role/Cross-Account-Role-B"
        }
    ]
}
```

f. Tetapkan `dynamodb.customAWSCredentialsProvider` properti dengan nilai seperti `com.amazonaws.emr.AssumeRoleAWSCredentialsProvider` dalam klasifikasi inti-situs. Atur variabel lingkungan `ASSUME_ROLE_CREDENTIALS_ROLE_ARN` dengan ARN nilai`Cross-Account-Role-B`.

3. Jalankan Spark atau Hive job menggunakan. `Job-Execution-Role-A`

## Pertimbangan

### Pertimbangan saat menggunakan konektor DynamoDB dengan Apache Spark

- Spark SQL tidak mendukung pembuatan tabel Hive dengan opsi penangan penyimpanan. Untuk informasi selengkapnya, lihat Menentukan format penyimpanan untuk tabel Hive dalam dokumentasi Apache Spark.
- Spark SQL tidak mendukung `STORED BY` operasi dengan handler penyimpanan. Jika Anda ingin berinteraksi dengan tabel DynamoDB melalui tabel Hive eksternal, gunakan Hive untuk membuat tabel terlebih dahulu.

- Untuk menerjemahkan kueri ke query DynamoDB, konektor DynamoDB menggunakan pushdown predikat. Predikat pushdown memfilter data dengan kolom yang dipetakan ke kunci partisi dari tabel DynamoDB. Predikat pushdown hanya beroperasi ketika Anda menggunakan konektor dengan SparkSQL, dan bukan dengan. MapReduce API

## Pertimbangan saat menggunakan konektor DynamoDB dengan Apache Hive

### Menyetel jumlah maksimum mapper

- Jika Anda menggunakan `SELECT` kueri untuk membaca data dari tabel Hive eksternal yang memetakan ke DynamoDB, jumlah tugas peta EMR di Tanpa Server dihitung sebagai total throughput baca yang dikonfigurasi untuk tabel DynamoDB, dibagi dengan throughput per tugas peta. Throughput default per tugas peta adalah 100.
- Pekerjaan Hive dapat menggunakan jumlah tugas peta di luar jumlah maksimum kontainer yang dikonfigurasi per aplikasi EMR Tanpa Server, tergantung pada throughput baca yang dikonfigurasi untuk DynamoDB. Selain itu, kueri Hive yang berjalan lama dapat menggunakan semua kapasitas baca yang disediakan dari tabel DynamoDB. Ini berdampak negatif pada pengguna lain.
- Anda dapat menggunakan `dynamodb.max.map.tasks` properti untuk menetapkan batas atas untuk tugas peta. Anda juga dapat menggunakan properti ini untuk menyetel jumlah data yang dibaca oleh setiap tugas peta berdasarkan ukuran wadah tugas.
- Anda dapat mengatur `dynamodb.max.map.tasks` properti di tingkat kueri Hive, atau dalam `hive-site` klasifikasi start-job-run perintah. Nilai ini harus lebih besar atau sama dengan 1. Saat Hive memproses kueri Anda, pekerjaan Hive yang dihasilkan menggunakan tidak lebih dari nilai `dynamodb.max.map.tasks` saat membaca dari tabel DynamoDB.

### Menyetel throughput tulis per tugas

- Tulis throughput per tugas di EMR Tanpa Server dihitung sebagai total throughput tulis yang dikonfigurasi untuk tabel DynamoDB, dibagi dengan nilai properti. `mapreduce.job.maps` Untuk Hive, nilai default properti ini adalah 2. Oleh karena itu, dua tugas pertama di tahap akhir pekerjaan Hive dapat menghabiskan semua throughput penulisan. Hal ini menyebabkan pembatasan penulisan tugas lain dalam pekerjaan yang sama atau pekerjaan lain.
- Untuk menghindari pelambatan tulis, Anda dapat mengatur nilai `mapreduce.job.maps` properti berdasarkan jumlah tugas di tahap akhir atau throughput tulis yang ingin Anda alokasikan per tugas. Tetapkan properti ini dalam `mapred-site` klasifikasi start-job-run perintah di Tanpa EMR Server.

# Keamanan

Keamanan cloud di AWS adalah prioritas tertinggi. Sebagai AWS pelanggan, Anda 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 AWS dan kamu. Model tanggung jawab bersama menjelaskan hal ini sebagai keamanan  cloud dan keamanan dalam cloud:

- Keamanan Cloud — AWS Bertanggung jawab untuk melindungi infrastruktur yang berjalan AWS layanan di AWS Awan. AWS juga memberi Anda layanan yang dapat Anda gunakan dengan aman. Auditor pihak ketiga secara teratur menguji dan memverifikasi efektivitas keamanan kami sebagai bagian dari AWS program kepatuhan. Untuk mempelajari tentang program kepatuhan yang berlaku untuk Amazon EMR Tanpa Server, lihat AWS layanan dalam lingkup oleh program kepatuhan.
- Keamanan di cloud — Tanggung jawab Anda ditentukan oleh AWS layanan yang Anda gunakan. Anda juga bertanggung jawab atas faktor lain, termasuk sensitivitas data Anda, persyaratan perusahaan Anda, serta undang-undang dan peraturan yang berlaku.

Dokumentasi ini membantu Anda memahami cara menerapkan model tanggung jawab bersama saat menggunakan Amazon Tanpa EMR Server. Topik dalam Bab ini menunjukkan kepada Anda cara mengonfigurasi Amazon EMR Tanpa Server dan menggunakan yang lain AWS layanan untuk memenuhi tujuan keamanan dan kepatuhan Anda.

Topik

- Praktik terbaik keamanan untuk Amazon Tanpa EMR Server
- Perlindungan data
- Identity and Access Management (IAM) di Amazon Tanpa EMR Server
- Menggunakan EMR Serverless dengan AWS Lake Formation untuk kontrol akses berbutir halus
- Enkripsi antar pekerja
- Secrets Manager untuk perlindungan data dengan EMR Serverless
- Menggunakan Hibah Akses Amazon S3 dengan Tanpa Server EMR
- Mencatat panggilan Amazon EMR Tanpa Server menggunakan API AWS CloudTrail
- Validasi kepatuhan untuk Amazon Tanpa Server EMR

- Ketahanan di Amazon Tanpa Server EMR
- Keamanan infrastruktur di Amazon Tanpa EMR Server
- Analisis konfigurasi dan kerentanan di Amazon Tanpa Server EMR

# Praktik terbaik keamanan untuk Amazon Tanpa EMR Server

Amazon EMR Serverless menyediakan sejumlah fitur keamanan untuk dipertimbangkan saat Anda mengembangkan dan menerapkan kebijakan keamanan Anda sendiri. Praktik terbaik berikut adalah pedoman umum dan tidak mewakili solusi keamanan yang lengkap. Karena praktik terbaik ini mungkin tidak sesuai atau cukup untuk lingkungan Anda, anggap sebagai pertimbangan yang membantu dan bukan sebagai resep.

## Terapkan prinsip hak istimewa paling rendah

EMRTanpa server menyediakan kebijakan akses terperinci untuk aplikasi yang menggunakan IAM peran, seperti peran eksekusi. Kami merekomendasikan bahwa peran eksekusi diberikan hanya seperangkat minimum hak istimewa yang diperlukan oleh tugas, seperti mencakup aplikasi Anda dan akses ke tujuan log. Kami juga merekomendasikan mengaudit tugas untuk izin secara teratur dan pada setiap perubahan pada kode aplikasi.

## Mengisolasi kode aplikasi yang tidak tepercaya

EMRTanpa server menciptakan isolasi jaringan penuh antara pekerjaan milik aplikasi Tanpa EMR Server yang berbeda. Dalam kasus di mana isolasi tingkat pekerjaan diinginkan, pertimbangkan untuk mengisolasi pekerjaan ke dalam aplikasi Tanpa Server yang berbedaEMR.

## Izin kontrol akses berbasis peran () RBAC

Administrator harus secara ketat mengontrol izin kontrol akses (RBAC) berbasis peran untuk aplikasi Tanpa Server. EMR

# Perlindungan data

Bagian AWS Model tanggung jawab bersama berlaku untuk perlindungan data di Amazon Tanpa EMR Server. Seperti yang dijelaskan dalam model ini, AWS bertanggung jawab untuk melindungi infrastruktur global yang menjalankan semua AWS Awan. Anda bertanggung jawab untuk mempertahankan kendali atas konten yang di-host pada infrastruktur ini. Konten ini mencakup

konfigurasi keamanan dan tugas manajemen untuk AWS layanan yang Anda gunakan. Untuk informasi selengkapnya tentang privasi data, lihat Privasi Data FAQ. Untuk informasi tentang perlindungan data di Eropa, lihat AWS Model Tanggung Jawab Bersama dan posting GDPR blog di AWS Blog Keamanan.

Untuk tujuan perlindungan data, kami menyarankan Anda untuk melindungi AWS kredensi akun dan mengatur akun individu dengan AWS Identity and Access Management (IAM) Dengan cara ini, setiap pengguna hanya diberi izin yang diperlukan untuk memenuhi tanggung jawab tugas mereka. Kami juga merekomendasikan agar Anda mengamankan data Anda dengan cara-cara berikut ini:

- Gunakan otentikasi multi-faktor (MFA) dengan setiap akun.
- GunakanSSL/TLSuntuk berkomunikasi dengan AWS sumber daya. Kami merekomendasikan TLS 1.2 atau yang lebih baru.
- Siapkan API dan pencatatan aktivitas pengguna dengan AWS CloudTrail.
- Gunakan AWS solusi enkripsi, bersama dengan semua kontrol keamanan default di dalamnya AWS layanan.
- Gunakan layanan keamanan terkelola lanjutan seperti Amazon Macie, yang membantu menemukan dan mengamankan data pribadi yang disimpan di Amazon S3.
- Gunakan opsi enkripsi Amazon EMR Tanpa Server untuk mengenkripsi data saat istirahat dan dalam perjalanan.
- Jika Anda memerlukan FIPS 140-2 modul kriptografi yang divalidasi saat mengakses AWS melalui antarmuka baris perintah atauAPI, gunakan FIPS titik akhir. Untuk informasi selengkapnya tentang FIPS titik akhir yang tersedia, lihat Federal Information Processing Standard (FIPS) 140-2.

Sebaiknya jangan pernah memasukkan informasi identitas yang sensitif, seperti nomor rekening pelanggan Anda, ke dalam bidang isian bebas seperti bidang Nama. Ini termasuk saat Anda bekerja dengan Amazon EMR Tanpa Server atau lainnya AWS layanan menggunakan konsol,API, AWS CLI, atau AWS SDKs. Data apa pun yang Anda masukkan ke Amazon EMR Tanpa Server atau layanan lain mungkin diambil untuk dimasukkan dalam log diagnostik. Ketika Anda memberikan URL ke server eksternal, jangan sertakan informasi kredensial dalam URL untuk memvalidasi permintaan Anda ke server tersebut.

## Enkripsi diam

Enkripsi data membantu mencegah pengguna yang tidak sah membaca data pada klaster dan sistem penyimpanan data terkait. Ini termasuk data yang disimpan ke media persisten, yang dikenal sebagai

data at rest, dan data yang mungkin dicegat saat perjalanan jaringan, yang dikenal sebagai data dalam transit.

Enkripsi data memerlukan kunci dan sertifikat. Anda dapat memilih dari beberapa opsi, termasuk kunci yang dikelola oleh AWS Key Management Service, kunci yang dikelola oleh Amazon S3, serta kunci serta sertifikat dari penyedia khusus yang Anda berikan. Saat menggunakan AWS KMS sebagai penyedia kunci Anda, biaya berlaku untuk penyimpanan dan penggunaan kunci enkripsi. Untuk informasi selengkapnya, silakan lihat AWS KMS harga.

Sebelum Anda menentukan opsi enkripsi, tentukan sistem manajemen kunci dan sertifikat yang ingin Anda gunakan. Kemudian buat kunci dan sertifikat untuk penyedia kustom yang Anda tentukan sebagai bagian dari pengaturan enkripsi.

### Enkripsi saat istirahat untuk EMRFS data di Amazon S3

Setiap aplikasi EMR Tanpa Server menggunakan versi rilis tertentu, yang mencakup EMRFS (Sistem EMR File). Enkripsi Amazon S3 berfungsi dengan objek Sistem EMR File (EMRFS) yang dibaca dan ditulis ke Amazon S3. Anda dapat menentukan enkripsi sisi server Amazon S3 (SSE) atau enkripsi sisi klien (CSE) sebagai mode enkripsi Default saat Anda mengaktifkan enkripsi saat istirahat. Secara opsional, Anda dapat menentukan metode enkripsi yang berbeda untuk setiap bucket menggunakan Per penimpaan enkripsi bucket. Terlepas dari apakah enkripsi Amazon S3 diaktifkan, Transport Layer Security (TLS) mengenkripsi EMRFS objek yang sedang transit antara node EMR cluster dan Amazon S3. Jika Anda menggunakan Amazon S3 CSE dengan kunci yang dikelola pelanggan, peran eksekusi yang digunakan untuk menjalankan pekerjaan di aplikasi EMR Tanpa Server harus memiliki akses ke kunci tersebut. Untuk informasi mendalam tentang enkripsi Amazon S3, lihat Melindungi data menggunakan enkripsi di Panduan Pengembang Layanan Penyimpanan Sederhana Amazon.

> **Note**
>
> Saat Anda menggunakan AWS KMS, biaya berlaku untuk penyimpanan dan penggunaan kunci enkripsi. Untuk informasi selengkapnya, silakan lihat AWS KMS harga.

### Enkripsi sisi server Amazon S3

Saat Anda mengatur enkripsi sisi server Amazon S3, Amazon S3 akan mengenkripsi data pada tingkat objek saat menulis data ke disk dan mendekripsi data saat diakses. Untuk informasi selengkapnyaSSE, lihat Melindungi data menggunakan enkripsi sisi server di Panduan Pengembang Layanan Penyimpanan Sederhana Amazon.

Anda dapat memilih di antara dua sistem manajemen kunci yang berbeda saat Anda menentukan SSE di Amazon Tanpa EMR Server:

- SSE-S3 - Amazon S3 mengelola kunci untuk Anda. Tidak ada pengaturan tambahan yang diperlukan di Tanpa EMR Server.
- SSE- KMS - Anda menggunakan AWS KMS key untuk mengatur dengan kebijakan yang cocok untuk Tanpa EMR Server. Tidak ada pengaturan tambahan yang diperlukan di Tanpa EMR Server.

Untuk menggunakan AWS KMS enkripsi untuk data yang Anda tulis ke Amazon S3, Anda memiliki dua opsi saat Anda menggunakan. `StartJobRun` API Anda dapat mengaktifkan enkripsi untuk semua yang Anda tulis ke Amazon S3, atau Anda dapat mengaktifkan enkripsi untuk data yang Anda tulis ke bucket tertentu. Untuk informasi selengkapnya tentang `StartJobRun`API, lihat Referensi EMRTanpa Server API.

Untuk menghidupkan AWS KMS enkripsi untuk semua data yang Anda tulis ke Amazon S3, gunakan perintah berikut saat Anda memanggil file. `StartJobRun` API

```
--conf spark.hadoop.fs.s3.enableServerSideEncryption=true
--conf spark.hadoop.fs.s3.serverSideEncryption.kms.keyId=<kms_id>
```

Untuk menghidupkan AWS KMS enkripsi untuk data yang Anda tulis ke bucket tertentu, gunakan perintah berikut saat Anda memanggil `StartJobRun`API.

```
--conf spark.hadoop.fs.s3.bucket.<DOC-EXAMPLE-BUCKET>.enableServerSideEncryption=true
--conf spark.hadoop.fs.s3.bucket.<DOC-EXAMPLE-
BUCKET>.serverSideEncryption.kms.keyId=<kms-id>
```

SSEdengan kunci yang disediakan pelanggan (SSE-C) tidak tersedia untuk digunakan dengan Tanpa Server. EMR

### Enkripsi di sisi klien Amazon S3

Dengan enkripsi sisi klien Amazon S3, enkripsi dan dekripsi Amazon S3 berlangsung di klien yang tersedia di setiap rilis Amazon. EMRFS EMR Objek dienkripsi sebelum diunggah ke Amazon S3 dan didekripsi setelah diunduh. Penyedia yang Anda tentukan menyediakan kunci enkripsi yang digunakan klien. Klien dapat menggunakan kunci yang disediakan oleh AWS KMS (CSE-KMS) atau kelas Java kustom yang menyediakan kunci root sisi klien (CSE-C). Spesifikasi enkripsi sedikit berbeda antara CSE - KMS dan CSE -C, tergantung pada penyedia yang ditentukan dan metadata objek yang didekripsi atau dienkripsi. Jika Anda menggunakan Amazon S3 CSE dengan kunci yang

dikelola pelanggan, peran eksekusi yang digunakan untuk menjalankan pekerjaan di aplikasi EMR Tanpa Server harus memiliki akses ke kunci tersebut. KMSBiaya tambahan mungkin berlaku. Untuk informasi selengkapnya tentang perbedaan ini, lihat Melindungi data menggunakan enkripsi sisi klien di Panduan Pengembang Layanan Penyimpanan Sederhana Amazon.

### Enkripsi disk lokal

Data yang disimpan dalam penyimpanan sementara dienkripsi dengan kunci yang dimiliki layanan menggunakan standar AES industri -256 algoritma kriptografi.

### Manajemen kunci

Anda dapat mengonfigurasi KMS untuk memutar KMS kunci Anda secara otomatis. Ini merotasi kunci Anda setahun sekali sambil menyimpan kunci lama tanpa batas waktu sehingga data Anda masih dapat didekripsi. Untuk informasi tambahan, lihat Merotasi kunci master pelanggan.

## Enkripsi bergerak

Fitur enkripsi khusus aplikasi berikut tersedia dengan Amazon EMR Serverless:

- Spark
  - Secara default, komunikasi antara driver Spark dan pelaksana diautentikasi dan internal. RPCkomunikasi antara driver dan pelaksana dienkripsi.
- Hive
  - Komunikasi antara AWS Glue metastore dan aplikasi EMR Tanpa Server terjadi melalui. TLS

Anda harus mengizinkan hanya koneksi terenkripsi over HTTPS (TLS) menggunakan aws: SecureTransport condition pada kebijakan bucket Amazon IAM S3.

# Identity and Access Management (IAM) di Amazon Tanpa EMR Server

AWS Identity and Access Management (IAM) adalah Layanan AWS yang membantu administrator mengontrol akses ke AWS sumber daya. IAMadministrator mengontrol siapa yang dapat diautentikasi (masuk) dan diberi wewenang (memiliki izin) untuk menggunakan sumber daya Amazon EMR Tanpa Server. IAMadalah sebuah Layanan AWS yang dapat Anda gunakan tanpa biaya tambahan.

Topik

## Audiens

Bagaimana Anda menggunakan AWS Identity and Access Management (IAM) berbeda, tergantung pada pekerjaan yang Anda lakukan di Amazon Tanpa EMR Server.

Pengguna layanan - Jika Anda menggunakan layanan Amazon EMR Tanpa Server untuk melakukan pekerjaan Anda, administrator Anda memberi Anda kredensi dan izin yang Anda butuhkan. Saat Anda menggunakan lebih banyak fitur Amazon EMR Tanpa Server untuk melakukan pekerjaan Anda, Anda mungkin memerlukan izin tambahan. Memahami cara akses dikelola dapat membantu Anda meminta izin yang tepat dari administrator Anda. Jika Anda tidak dapat mengakses fitur di Amazon EMR Tanpa Server, lihat. Memecahkan masalah identitas dan akses Amazon EMR Tanpa Server

Administrator layanan - Jika Anda bertanggung jawab atas sumber daya Amazon EMR Tanpa Server di perusahaan Anda, Anda mungkin memiliki akses penuh ke Amazon EMR Tanpa Server. Tugas Anda adalah menentukan fitur dan sumber daya Amazon EMR Tanpa Server mana yang harus diakses pengguna layanan Anda. Anda kemudian harus mengirimkan permintaan ke IAM administrator Anda untuk mengubah izin pengguna layanan Anda. Tinjau informasi di halaman ini untuk memahami konsep dasarIAM. Untuk mempelajari selengkapnya tentang bagaimana perusahaan Anda dapat menggunakan Amazon IAM EMR Tanpa Server, lihat. Identity and Access Management (IAM) di Amazon Tanpa EMR Server

IAMadministrator - Jika Anda seorang IAM administrator, Anda mungkin ingin mempelajari detail tentang cara menulis kebijakan untuk mengelola akses ke Amazon Tanpa EMR Server. Untuk

melihat contoh kebijakan berbasis identitas Amazon EMR Tanpa Server yang dapat Anda gunakan, lihat. IAM Contoh kebijakan berbasis identitas untuk Tanpa Server EMR

## Mengautentikasi dengan identitas

Otentikasi adalah cara Anda masuk AWS menggunakan kredensil identitas Anda. Anda harus diautentikasi (masuk ke AWS) sebagai Pengguna root akun AWS, sebagai IAM pengguna, atau dengan mengambil IAM peran.

Anda dapat masuk ke AWS sebagai identitas federasi dengan menggunakan kredensi yang disediakan melalui sumber identitas. AWS IAM Identity Center Pengguna (Pusat IAM Identitas), autentikasi masuk tunggal perusahaan Anda, dan kredensi Google atau Facebook Anda adalah contoh identitas federasi. Saat Anda masuk sebagai identitas federasi, administrator Anda sebelumnya menyiapkan federasi identitas menggunakan IAM peran. Saat Anda mengakses AWS dengan menggunakan federasi, Anda secara tidak langsung mengambil peran.

Tergantung pada jenis pengguna Anda, Anda dapat masuk ke AWS Management Console atau AWS portal akses. Untuk informasi lebih lanjut tentang masuk AWS, lihat Cara masuk ke Akun AWSdi AWS Sign-In Panduan Pengguna.

Jika Anda mengakses AWS secara terprogram, AWS menyediakan kit pengembangan perangkat lunak (SDK) dan antarmuka baris perintah (CLI) untuk menandatangani permintaan Anda secara kriptografis dengan menggunakan kredensil Anda. Jika Anda tidak menggunakan AWS alat, Anda harus menandatangani permintaan sendiri. Untuk informasi selengkapnya tentang menggunakan metode yang disarankan untuk menandatangani permintaan sendiri, lihat Menandatangani AWS APIpermintaan di Panduan IAM Pengguna.

Apa pun metode autentikasi yang digunakan, Anda mungkin diminta untuk menyediakan informasi keamanan tambahan. Misalnya, AWS merekomendasikan agar Anda menggunakan otentikasi multi-faktor (MFA) untuk meningkatkan keamanan akun Anda. Untuk mempelajari selengkapnya, lihat Autentikasi multi-faktor di AWS IAM Identity Center Panduan Pengguna dan Menggunakan otentikasi multi-faktor () MFA di AWS di Panduan Pengguna IAM.

### Akun AWS pengguna root

Saat Anda membuat Akun AWS, Anda mulai dengan satu identitas masuk yang memiliki akses lengkap ke semua Layanan AWS dan sumber daya di akun. Identitas ini disebut Akun AWS pengguna root dan diakses dengan masuk dengan alamat email dan kata sandi yang Anda gunakan untuk membuat akun. Kami sangat menyarankan agar Anda tidak menggunakan pengguna root untuk tugas sehari-hari. Lindungi kredensial pengguna root Anda dan gunakan kredensial tersebut

untuk melakukan tugas yang hanya dapat dilakukan pengguna root. Untuk daftar lengkap tugas yang mengharuskan Anda masuk sebagai pengguna root, lihat Tugas yang memerlukan kredensi pengguna root di IAMPanduan Pengguna.

### Identitas gabungan

Sebagai praktik terbaik, mewajibkan pengguna manusia, termasuk pengguna yang memerlukan akses administrator, untuk menggunakan federasi dengan penyedia identitas untuk mengakses Layanan AWS dengan menggunakan kredensil sementara.

Identitas federasi adalah pengguna dari direktori pengguna perusahaan Anda, penyedia identitas web, AWS Directory Service, direktori Pusat Identitas, atau pengguna mana pun yang mengakses Layanan AWS dengan menggunakan kredensil yang disediakan melalui sumber identitas. Ketika akses identitas federasi Akun AWS, mereka mengambil peran, dan peran memberikan kredensi sementara.

Untuk manajemen akses terpusat, kami sarankan Anda menggunakan AWS IAM Identity Center. Anda dapat membuat pengguna dan grup di Pusat IAM Identitas, atau Anda dapat menghubungkan dan menyinkronkan ke sekumpulan pengguna dan grup di sumber identitas Anda sendiri untuk digunakan di semua Akun AWS dan aplikasi. Untuk informasi tentang Pusat IAM Identitas, lihat Apa itu Pusat IAM Identitas? di AWS IAM Identity Center Panduan Pengguna.

### Pengguna dan grup IAM

IAMPengguna adalah identitas di dalam Akun AWS yang memiliki izin khusus untuk satu orang atau aplikasi. Jika memungkinkan, sebaiknya mengandalkan kredensi sementara daripada membuat IAM pengguna yang memiliki kredensi jangka panjang seperti kata sandi dan kunci akses. Namun, jika Anda memiliki kasus penggunaan khusus yang memerlukan kredensil jangka panjang dengan IAM pengguna, kami sarankan Anda memutar kunci akses. Untuk informasi selengkapnya, lihat Memutar kunci akses secara teratur untuk kasus penggunaan yang memerlukan kredensi jangka panjang di IAMPanduan Pengguna.

IAMGrup adalah identitas yang menentukan kumpulan IAM pengguna. Anda tidak dapat masuk sebagai grup. Anda dapat menggunakan grup untuk menentukan izin bagi beberapa pengguna sekaligus. Grup mempermudah manajemen izin untuk sejumlah besar pengguna sekaligus. Misalnya, Anda dapat memiliki grup bernama IAMAdminsdan memberikan izin grup tersebut untuk mengelola sumber dayaIAM.

Pengguna berbeda dari peran. Pengguna secara unik terkait dengan satu orang atau aplikasi, tetapi peran dimaksudkan untuk dapat digunakan oleh siapa pun yang membutuhkannya. Pengguna

memiliki kredensial jangka panjang permanen, tetapi peran memberikan kredensial sementara. Untuk mempelajari lebih lanjut, lihat Kapan membuat IAM pengguna (bukan peran) di Panduan IAM Pengguna.

## IAMperan

IAMPeran adalah identitas dalam diri Anda Akun AWS yang memiliki izin khusus. Ini mirip dengan IAM pengguna, tetapi tidak terkait dengan orang tertentu. Anda dapat mengambil IAM peran sementara dalam AWS Management Console dengan beralih peran. Anda dapat mengambil peran dengan memanggil AWS CLI atau AWS APIoperasi atau dengan menggunakan kustomURL. Untuk informasi selengkapnya tentang metode penggunaan peran, lihat Menggunakan IAM peran di Panduan IAM Pengguna.

IAMperan dengan kredensi sementara berguna dalam situasi berikut:

- Akses pengguna terfederasi – Untuk menetapkan izin ke identitas terfederasi, Anda membuat peran dan menentukan izin untuk peran tersebut. Ketika identitas terfederasi mengautentikasi, identitas tersebut terhubung dengan peran dan diberi izin yang ditentukan oleh peran. Untuk informasi tentang peran untuk federasi, lihat Membuat peran untuk Penyedia Identitas pihak ketiga di Panduan IAM Pengguna. Jika Anda menggunakan Pusat IAM Identitas, Anda mengonfigurasi set izin. Untuk mengontrol apa yang dapat diakses identitas Anda setelah diautentikasi, Pusat IAM Identitas menghubungkan izin yang disetel ke peran. IAM Untuk informasi tentang set izin, lihat Set izin di AWS IAM Identity Center Panduan Pengguna.
- Izin IAM pengguna sementara — IAM Pengguna atau peran dapat mengambil IAM peran untuk sementara mengambil izin yang berbeda untuk tugas tertentu.
- Akses lintas akun — Anda dapat menggunakan IAM peran untuk memungkinkan seseorang (prinsipal tepercaya) di akun lain mengakses sumber daya di akun Anda. Peran adalah cara utama untuk memberikan akses lintas akun. Namun, dengan beberapa Layanan AWS, Anda dapat melampirkan kebijakan langsung ke sumber daya (alih-alih menggunakan peran sebagai proxy). Untuk mempelajari perbedaan antara peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Akses sumber daya lintas akun di IAM Panduan Pengguna. IAM
- Akses lintas layanan - Beberapa Layanan AWS menggunakan fitur di lain Layanan AWS. Misalnya, saat Anda melakukan panggilan dalam suatu layanan, biasanya layanan tersebut menjalankan aplikasi di Amazon EC2 atau menyimpan objek di Amazon S3. Sebuah layanan mungkin melakukannya menggunakan izin prinsipal yang memanggil, menggunakan peran layanan, atau peran terkait layanan.

  - Teruskan sesi akses (FAS) — Saat Anda menggunakan IAM pengguna atau peran untuk melakukan tindakan AWS Anda dianggap sebagai kepala sekolah. Ketika Anda menggunakan beberapa layanan, Anda mungkin melakukan sebuah tindakan yang kemudian menginisiasi tindakan lain di layanan yang berbeda. FASmenggunakan izin dari prinsipal yang memanggil Layanan AWS, dikombinasikan dengan permintaan Layanan AWS untuk membuat permintaan ke layanan hilir. FASPermintaan hanya dibuat ketika layanan menerima permintaan yang memerlukan interaksi dengan orang lain Layanan AWS atau sumber daya untuk diselesaikan. Dalam hal ini, Anda harus memiliki izin untuk melakukan kedua tindakan tersebut. Untuk detail kebijakan saat membuat FAS permintaan, lihat Meneruskan sesi akses.
  - Peran layanan — Peran layanan adalah IAMperan yang diasumsikan layanan untuk melakukan tindakan atas nama Anda. IAMAdministrator dapat membuat, memodifikasi, dan menghapus peran layanan dari dalamIAM. Untuk informasi selengkapnya, lihat Membuat peran untuk mendelegasikan izin ke Layanan AWS di Panduan Pengguna IAM.
  - Peran terkait layanan — Peran terkait layanan adalah jenis peran layanan yang ditautkan ke Layanan AWS. Layanan dapat mengambil peran untuk melakukan tindakan atas nama Anda. Peran terkait layanan muncul di Akun AWS dan dimiliki oleh layanan. IAMAdministrator dapat melihat, tetapi tidak mengedit izin untuk peran terkait layanan.
- Aplikasi yang berjalan di Amazon EC2 — Anda dapat menggunakan IAM peran untuk mengelola kredensional sementara untuk aplikasi yang berjalan pada EC2 instance dan membuat AWS CLI atau AWS APIpermintaan. Ini lebih baik untuk menyimpan kunci akses dalam EC2 instance. Untuk menetapkan AWS peran ke sebuah EC2 instance dan membuatnya tersedia untuk semua aplikasinya, Anda membuat profil instance yang dilampirkan ke instance. Profil instance berisi peran dan memungkinkan program yang berjalan pada EC2 instance untuk mendapatkan kredensi sementara. Untuk informasi selengkapnya, lihat Menggunakan IAM peran untuk memberikan izin ke aplikasi yang berjalan di EC2 instans Amazon di IAMPanduan Pengguna.

Untuk mempelajari apakah akan menggunakan IAM peran atau IAM pengguna, lihat Kapan membuat IAM peran (bukan pengguna) di Panduan IAM Pengguna.

## Mengelola akses menggunakan kebijakan

Anda mengontrol akses di AWS dengan membuat kebijakan dan melampirkannya AWS identitas atau sumber daya. Kebijakan adalah objek di AWS bahwa, ketika dikaitkan dengan identitas atau sumber daya, mendefinisikan izin mereka. AWS mengevaluasi kebijakan ini ketika prinsipal (pengguna, pengguna root, atau sesi peran) membuat permintaan. Izin dalam kebijakan menentukan apakah permintaan diizinkan atau ditolak. Sebagian besar kebijakan disimpan di AWS sebagai JSON

dokumen. Untuk informasi selengkapnya tentang struktur dan isi dokumen JSON kebijakan, lihat Ringkasan JSON kebijakan di Panduan IAM Pengguna.

Administrator dapat menggunakan AWS JSONkebijakan untuk menentukan siapa yang memiliki akses ke apa. Yaitu, principal dapat melakukan tindakan pada suatu sumber daya, dan dalam suatu syarat.

Secara default, pengguna dan peran tidak memiliki izin. Untuk memberikan izin kepada pengguna untuk melakukan tindakan pada sumber daya yang mereka butuhkan, IAM administrator dapat membuat IAM kebijakan. Administrator kemudian dapat menambahkan IAM kebijakan ke peran, dan pengguna dapat mengambil peran.

IAMkebijakan menentukan izin untuk tindakan terlepas dari metode yang Anda gunakan untuk melakukan operasi. Misalnya, anggaplah Anda memiliki kebijakan yang mengizinkan tindakan `iam:GetRole`. Pengguna dengan kebijakan tersebut dapat memperoleh informasi peran dari AWS Management Console, AWS CLI, atau AWS API.

### Kebijakan berbasis identitas

Kebijakan berbasis identitas adalah dokumen kebijakan JSON izin yang dapat Anda lampirkan ke identitas, seperti pengguna, grup IAM pengguna, atau peran. Kebijakan ini mengontrol jenis tindakan yang dapat dilakukan oleh pengguna dan peran, di sumber daya mana, dan berdasarkan kondisi seperti apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Membuat IAM kebijakan di Panduan Pengguna. IAM

Kebijakan berbasis identitas dapat dikategorikan lebih lanjut sebagai kebijakan inline atau kebijakan yang dikelola. Kebijakan inline disematkan langsung ke satu pengguna, grup, atau peran. Kebijakan terkelola adalah kebijakan mandiri yang dapat Anda lampirkan ke beberapa pengguna, grup, dan peran di Akun AWS. Kebijakan terkelola meliputi AWS kebijakan terkelola dan kebijakan yang dikelola pelanggan. Untuk mempelajari cara memilih antara kebijakan terkelola atau kebijakan sebaris, lihat Memilih antara kebijakan terkelola dan kebijakan sebaris di IAMPanduan Pengguna.

### Kebijakan berbasis sumber daya

Kebijakan berbasis sumber daya adalah dokumen JSON kebijakan yang Anda lampirkan ke sumber daya. Contoh kebijakan berbasis sumber daya adalah kebijakan kepercayaan IAM peran dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya, administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Untuk sumber daya tempat kebijakan dilampirkan, kebijakan menentukan tindakan apa yang dapat

dilakukan oleh prinsipal tertentu pada sumber daya tersebut dan dalam kondisi apa. Anda harus menentukan prinsipal dalam kebijakan berbasis sumber daya. Prinsipal dapat mencakup akun, pengguna, peran, pengguna federasi, atau Layanan AWS.

Kebijakan berbasis sumber daya merupakan kebijakan inline yang terletak di layanan tersebut. Anda tidak dapat menggunakan AWS kebijakan terkelola dari IAM dalam kebijakan berbasis sumber daya.

### Daftar kontrol akses (ACLs)

Access control lists (ACLs) mengontrol prinsipal mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACLsmirip dengan kebijakan berbasis sumber daya, meskipun mereka tidak menggunakan format dokumen kebijakan. JSON

Amazon S3, AWS WAF, dan Amazon VPC adalah contoh layanan yang mendukungACLs. Untuk mempelajari selengkapnyaACLs, lihat Ikhtisar daftar kontrol akses (ACL) di Panduan Pengembang Layanan Penyimpanan Sederhana Amazon.

### Jenis-jenis kebijakan lain

AWS mendukung jenis kebijakan tambahan yang kurang umum. Jenis-jenis kebijakan ini dapat mengatur izin maksimum yang diberikan kepada Anda oleh jenis kebijakan yang lebih umum.

- Batas izin — Batas izin adalah fitur lanjutan tempat Anda menetapkan izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas kepada entitas (pengguna atau peran). IAM IAM Anda dapat menetapkan batasan izin untuk suatu entitas. Izin yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas milik entitas dan batasan izinnya. Kebijakan berbasis sumber daya yang menentukan pengguna atau peran dalam bidang `Principal` tidak dibatasi oleh batasan izin. Penolakan eksplisit dalam salah satu kebijakan ini akan menggantikan pemberian izin. Untuk informasi selengkapnya tentang batas izin, lihat Batas izin untuk IAM entitas di IAMPanduan Pengguna.
- Kebijakan kontrol layanan (SCPs) — SCPs adalah JSON kebijakan yang menentukan izin maksimum untuk organisasi atau unit organisasi (OU) di AWS Organizations. AWS Organizations adalah layanan untuk mengelompokkan dan mengelola beberapa Akun AWS yang dimiliki bisnis Anda. Jika Anda mengaktifkan semua fitur dalam organisasi, Anda dapat menerapkan kebijakan kontrol layanan (SCPs) ke salah satu atau semua akun Anda. SCPMembatasi izin untuk entitas di akun anggota, termasuk masing-masing Pengguna root akun AWS. Untuk informasi selengkapnya tentang Organizations danSCPs, lihat Kebijakan kontrol layanan di AWS Organizations Panduan Pengguna.

- Kebijakan sesi – Kebijakan sesi adalah kebijakan lanjutan yang Anda berikan sebagai parameter ketika Anda membuat sesi sementara secara programatis untuk peran atau pengguna terfederasi. Izin sesi yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas pengguna atau peran dan kebijakan sesi. Izin juga bisa datang dari kebijakan berbasis sumber daya. Penolakan secara tegas dalam salah satu kebijakan ini membatalkan izin. Untuk informasi selengkapnya, lihat Kebijakan sesi di Panduan IAM Pengguna.

### Berbagai jenis kebijakan

Ketika beberapa jenis kebijakan berlaku pada suatu permintaan, izin yang dihasilkan lebih rumit untuk dipahami. Untuk mempelajari caranya AWS menentukan apakah akan mengizinkan permintaan saat beberapa jenis kebijakan terlibat, lihat Logika evaluasi kebijakan di Panduan IAM Pengguna.

## Bagaimana EMR Serverless bekerja dengan IAM

Sebelum Anda menggunakan IAM untuk mengelola akses ke Amazon EMR Tanpa Server, pelajari IAM fitur apa saja yang tersedia untuk digunakan dengan Amazon EMR Tanpa Server.

IAMfitur yang dapat Anda gunakan dengan Tanpa EMR Server

| IAMfitur | Dukungan Amazon EMR Tanpa Server |
|---|---|
| Kebijakan berbasis identitas | Ya |
| Kebijakan berbasis sumber daya | Tidak |
| Tindakan kebijakan | Ya |
| Sumber daya kebijakan | Ya |
| Kunci kondisi kebijakan | Tidak |
| ACLs | Tidak |
| ABAC(tag dalam kebijakan) | Ya |
| Kredensial sementara | Ya |
| Izin prinsipal | Ya |

| IAMfitur | Dukungan Amazon EMR Tanpa Server |
| --- | --- |
| Peran layanan | Tidak |
| Peran terkait layanan | Ya |

Untuk mendapatkan tampilan tingkat tinggi tentang bagaimana Tanpa EMR Server dan lainnya AWS layanan bekerja dengan sebagian besar IAM fitur, lihat AWS layanan yang bekerja dengan IAM dalam Panduan IAM Pengguna.

## Kebijakan berbasis identitas untuk Tanpa Server EMR

Mendukung kebijakan berbasis identitas: Ya

Kebijakan berbasis identitas adalah dokumen kebijakan JSON izin yang dapat Anda lampirkan ke identitas, seperti pengguna, grup IAM pengguna, atau peran. Kebijakan ini mengontrol jenis tindakan yang dapat dilakukan oleh pengguna dan peran, di sumber daya mana, dan berdasarkan kondisi seperti apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Membuat IAM kebijakan di Panduan Pengguna. IAM

Dengan kebijakan IAM berbasis identitas, Anda dapat menentukan tindakan dan sumber daya yang diizinkan atau ditolak serta kondisi di mana tindakan diizinkan atau ditolak. Anda tidak dapat menentukan secara spesifik prinsipal dalam sebuah kebijakan berbasis identitas karena prinsipal berlaku bagi pengguna atau peran yang melekat kepadanya. Untuk mempelajari semua elemen yang dapat Anda gunakan dalam JSON kebijakan, lihat referensi elemen IAM JSON kebijakan di Panduan IAM Pengguna.

Contoh kebijakan berbasis identitas untuk Tanpa Server EMR

Untuk melihat contoh kebijakan berbasis identitas Amazon EMR Tanpa Server, lihat. Contoh kebijakan berbasis identitas untuk Tanpa Server EMR

## Kebijakan berbasis sumber daya dalam Tanpa Server EMR

Mendukung kebijakan berbasis sumber daya: Tidak

Kebijakan berbasis sumber daya adalah dokumen JSON kebijakan yang Anda lampirkan ke sumber daya. Contoh kebijakan berbasis sumber daya adalah kebijakan kepercayaan IAM peran dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya,

administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Untuk sumber daya tempat kebijakan dilampirkan, kebijakan menentukan tindakan apa yang dapat dilakukan oleh prinsipal tertentu pada sumber daya tersebut dan dalam kondisi apa. Anda harus menentukan prinsipal dalam kebijakan berbasis sumber daya. Prinsipal dapat mencakup akun, pengguna, peran, pengguna federasi, atau Layanan AWS.

Untuk mengaktifkan akses lintas akun, Anda dapat menentukan seluruh akun atau IAM entitas di akun lain sebagai prinsipal dalam kebijakan berbasis sumber daya. Menambahkan prinsipal akun silang ke kebijakan berbasis sumber daya hanya setengah dari membangun hubungan kepercayaan. Ketika kepala sekolah dan sumber daya berbeda Akun AWS, IAM administrator di akun tepercaya juga harus memberikan izin entitas utama (pengguna atau peran) untuk mengakses sumber daya. Mereka memberikan izin dengan melampirkan kebijakan berbasis identitas kepada entitas. Namun, jika kebijakan berbasis sumber daya memberikan akses ke prinsipal dalam akun yang sama, tidak diperlukan kebijakan berbasis identitas tambahan. Untuk informasi selengkapnya, lihat Akses sumber daya lintas akun IAM di Panduan IAM Pengguna.

## Tindakan kebijakan untuk Tanpa EMR Server

Mendukung tindakan kebijakan: Ya

Administrator dapat menggunakan AWS JSONkebijakan untuk menentukan siapa yang memiliki akses ke apa. Yaitu, principal dapat melakukan tindakan pada suatu sumber daya, dan dalam suatu syarat.

`Action`Elemen JSON kebijakan menjelaskan tindakan yang dapat Anda gunakan untuk mengizinkan atau menolak akses dalam kebijakan. Tindakan kebijakan biasanya memiliki nama yang sama dengan yang terkait AWS APIoperasi. Ada beberapa pengecualian, seperti tindakan khusus izin yang tidak memiliki operasi yang cocok. API Ada juga beberapa operasi yang memerlukan beberapa tindakan dalam suatu kebijakan. Tindakan tambahan ini disebut tindakan dependen.

Menyertakan tindakan dalam kebijakan untuk memberikan izin untuk melakukan operasi terkait.

Untuk melihat daftar tindakan EMR Tanpa Server, lihat Tindakan, sumber daya, dan kunci kondisi untuk Amazon EMR Tanpa Server di Referensi Otorisasi Layanan.

Tindakan kebijakan di EMR Tanpa Server menggunakan awalan berikut sebelum tindakan.

```
emr-serverless
```

Untuk menetapkan secara spesifik beberapa tindakan dalam satu pernyataan, pisahkan tindakan tersebut dengan koma.

```
"Action": [
      "emr-serverless:action1",
      "emr-serverless:action2"
         ]
```

Untuk melihat contoh kebijakan berbasis identitas Amazon EMR Tanpa Server, lihat. Contoh kebijakan berbasis identitas untuk Tanpa Server EMR

### Sumber daya kebijakan untuk Tanpa EMR Server

Mendukung sumber daya kebijakan: Ya

Administrator dapat menggunakan AWS JSONkebijakan untuk menentukan siapa yang memiliki akses ke apa. Yaitu, principal dapat melakukan tindakan pada suatu sumber daya, dan dalam suatu syarat.

Elemen `Resource` JSON kebijakan menentukan objek atau objek yang tindakan tersebut berlaku. Pernyataan harus menyertakan elemen `Resource` atau `NotResource`. Sebagai praktik terbaik, tentukan sumber daya menggunakan Amazon Resource Name (ARN). Anda dapat melakukan ini untuk tindakan yang mendukung jenis sumber daya tertentu, yang dikenal sebagai izin tingkat sumber daya.

Untuk tindakan yang tidak mendukung izin di tingkat sumber daya, misalnya operasi pencantuman, gunakan wildcard (*) untuk menunjukkan bahwa pernyataan tersebut berlaku untuk semua sumber daya.

```
"Resource": "*"
```

Untuk melihat daftar jenis sumber daya Amazon EMR Tanpa Server dan jenisnyaARNs, lihat Sumber daya yang ditentukan oleh Amazon EMR Tanpa Server di Referensi Otorisasi Layanan. Untuk mempelajari tindakan mana yang dapat Anda tentukan ARN dari setiap sumber daya, lihat Tindakan, sumber daya, dan kunci kondisi untuk Amazon Tanpa EMR Server.

Untuk melihat contoh kebijakan berbasis identitas Amazon EMR Tanpa Server, lihat. Contoh kebijakan berbasis identitas untuk Tanpa Server EMR

## Kunci kondisi kebijakan untuk Tanpa EMR Server

| | |
|---|---|
| Mendukung kunci kondisi kebijakan khusus layanan | Tidak |

Administrator dapat menggunakan AWS JSONkebijakan untuk menentukan siapa yang memiliki akses ke apa. Yaitu, di mana utama dapat melakukan tindakan pada sumber daya, dan dalam kondisi apa.

Elemen `Condition` (atau blok `Condition`) akan memungkinkan Anda menentukan kondisi yang menjadi dasar suatu pernyataan berlaku. Elemen `Condition` bersifat opsional. Anda dapat membuat ekspresi bersyarat yang menggunakan operator kondisi, misalnya sama dengan atau kurang dari, untuk mencocokkan kondisi dalam kebijakan dengan nilai-nilai yang diminta.

Jika Anda menentukan beberapa `Condition` elemen dalam pernyataan, atau beberapa kunci dalam satu `Condition` elemen, AWS mengevaluasi mereka menggunakan `AND` operasi logis. Jika Anda menentukan beberapa nilai untuk satu kunci kondisi, AWS mengevaluasi kondisi menggunakan `OR` operasi logis. Semua kondisi harus dipenuhi sebelum izin pernyataan diberikan.

Anda juga dapat menggunakan variabel placeholder saat menentukan kondisi. Misalnya, Anda dapat memberikan izin IAM pengguna untuk mengakses sumber daya hanya jika ditandai dengan nama IAM pengguna mereka. Untuk informasi selengkapnya, lihat elemen IAM kebijakan: variabel dan tag di Panduan IAM Pengguna.

AWS mendukung kunci kondisi global dan kunci kondisi khusus layanan. Untuk melihat semua AWS kunci kondisi global, lihat AWS kunci konteks kondisi global di Panduan IAM Pengguna.

Untuk melihat daftar kunci kondisi Amazon EMR Tanpa Server dan untuk mempelajari tindakan dan sumber daya yang dapat Anda gunakan kunci kondisi, lihat Tindakan, sumber daya, dan kunci kondisi untuk Amazon EMR Tanpa Server di Referensi Otorisasi Layanan.

Semua EC2 tindakan Amazon mendukung kunci `aws:RequestedRegion` dan `ec2:Region` kondisi. Untuk informasi selengkapnya, lihat Contoh: Membatasi akses ke wilayah tertentu.

## Daftar kontrol akses (ACLs) di Tanpa EMR Server

MendukungACLs: Tidak

Access control lists (ACLs) mengontrol prinsipal mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACLsmirip dengan kebijakan berbasis sumber daya, meskipun mereka tidak menggunakan format dokumen kebijakan. JSON

## Kontrol akses berbasis atribut (ABAC) dengan Tanpa Server EMR

| | |
|---|---|
| Mendukung ABAC (tag dalam kebijakan) | Ya |

Attribute-based access control (ABAC) adalah strategi otorisasi yang mendefinisikan izin berdasarkan atribut. Masuk AWS, atribut ini disebut tag. Anda dapat melampirkan tag ke IAM entitas (pengguna atau peran) dan ke banyak AWS sumber daya. Menandai entitas dan sumber daya adalah langkah pertama dari. ABAC Kemudian Anda merancang ABAC kebijakan untuk mengizinkan operasi ketika tag prinsipal cocok dengan tag pada sumber daya yang mereka coba akses.

ABACmembantu dalam lingkungan yang berkembang pesat dan membantu dengan situasi di mana manajemen kebijakan menjadi rumit.

Untuk mengendalikan akses berdasarkan tag, berikan informasi tentang tag di elemen kondisi dari kebijakan menggunakan kunci kondisi `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`*, atau `aws:TagKeys`.

Jika sebuah layanan mendukung ketiga kunci kondisi untuk setiap jenis sumber daya, nilainya adalah Ya untuk layanan tersebut. Jika suatu layanan mendukung ketiga kunci kondisi untuk hanya beberapa jenis sumber daya, nilainya adalah Parsial.

Untuk informasi lebih lanjut tentangABAC, lihat Apa ituABAC? dalam IAMUser Guide. Untuk melihat tutorial dengan langkah-langkah penyiapanABAC, lihat Menggunakan kontrol akses berbasis atribut (ABAC) di IAMPanduan Pengguna.

## Menggunakan kredensil Sementara dengan Tanpa Server EMR

Mendukung kredensi sementara: Ya

Beberapa Layanan AWS tidak berfungsi saat Anda masuk menggunakan kredensil sementara. Untuk informasi tambahan, termasuk yang Layanan AWS bekerja dengan kredensi sementara, lihat Layanan AWS yang bekerja dengan IAM dalam Panduan IAM Pengguna.

Anda menggunakan kredensi sementara jika Anda masuk ke AWS Management Console menggunakan metode apa pun kecuali nama pengguna dan kata sandi. Misalnya, ketika Anda

mengakses AWS menggunakan tautan single sign-on (SSO) perusahaan Anda, proses itu secara otomatis membuat kredensi sementara. Anda juga akan secara otomatis membuat kredensial sementara ketika Anda masuk ke konsol sebagai seorang pengguna lalu beralih peran. Untuk informasi selengkapnya tentang beralih peran, lihat Beralih ke peran (konsol) di Panduan IAM Pengguna.

Anda dapat secara manual membuat kredensi sementara menggunakan AWS CLI atau AWS API. Anda kemudian dapat menggunakan kredensi sementara tersebut untuk mengakses AWS. AWS merekomendasikan agar Anda secara dinamis menghasilkan kredensi sementara alih-alih menggunakan kunci akses jangka panjang. Untuk informasi selengkapnya, lihat Kredensi keamanan sementara di. IAM

## Izin utama lintas layanan untuk Tanpa Server EMR

Mendukung sesi akses maju (FAS): Ya

Saat Anda menggunakan IAM pengguna atau peran untuk melakukan tindakan AWS Anda dianggap sebagai kepala sekolah. Ketika Anda menggunakan beberapa layanan, Anda mungkin melakukan sebuah tindakan yang kemudian menginisiasi tindakan lain di layanan yang berbeda. FASmenggunakan izin dari prinsipal yang memanggil Layanan AWS, dikombinasikan dengan permintaan Layanan AWS untuk membuat permintaan ke layanan hilir. FASPermintaan hanya dibuat ketika layanan menerima permintaan yang memerlukan interaksi dengan orang lain Layanan AWS atau sumber daya untuk diselesaikan. Dalam hal ini, Anda harus memiliki izin untuk melakukan kedua tindakan tersebut. Untuk detail kebijakan saat membuat FAS permintaan, lihat Meneruskan sesi akses.

## Peran layanan untuk Tanpa EMR Server

| | |
|---|---|
| Mendukung peran layanan | Tidak |

## Peran terkait layanan untuk Tanpa Server EMR

| | |
|---|---|
| Mendukung peran terkait layanan | Ya |

Untuk detail tentang membuat atau mengelola peran terkait layanan, lihat AWS layanan yang bekerja dengan IAM. Cari layanan dalam tabel yang memiliki `Yes` di kolom Peran terkait layanan. Pilih tautan Ya untuk melihat dokumentasi peran terkait layanan untuk layanan tersebut.

# Menggunakan peran terkait layanan untuk Tanpa Server EMR

Penggunaan Amazon EMR Tanpa Server AWS Identity and Access Management (IAM) peran terkait layanan. Peran terkait layanan adalah jenis peran unik yang ditautkan langsung ke EMR Tanpa Server. IAM Peran terkait layanan telah ditentukan sebelumnya oleh EMR Tanpa Server dan menyertakan semua izin yang diperlukan layanan untuk memanggil lainnya AWS layanan atas nama Anda.

Peran terkait layanan membuat pengaturan EMR Tanpa Server lebih mudah karena Anda tidak perlu menambahkan izin yang diperlukan secara manual. EMRTanpa server mendefinisikan izin peran terkait layanan, dan kecuali ditentukan lain, hanya EMR Tanpa Server yang dapat mengambil perannya. Izin yang ditetapkan mencakup kebijakan kepercayaan dan kebijakan izin, dan kebijakan izin tersebut tidak dapat dilampirkan ke entitas lain mana pun. IAM

Anda dapat menghapus peran tertaut layanan hanya setelah menghapus sumber daya terkait terlebih dahulu. Ini melindungi sumber daya EMR Tanpa Server Anda karena Anda tidak dapat secara tidak sengaja menghapus izin untuk mengakses sumber daya.

Untuk informasi tentang layanan lain yang mendukung peran terkait layanan, lihat AWS Layanan yang Bekerja dengan IAM dan mencari layanan yang memiliki Ya di kolom Peran terkait layanan. Pilih Ya bersama tautan untuk melihat dokumentasi peran tertaut layanan untuk layanan tersebut.

## Izin peran terkait layanan untuk Tanpa Server EMR

EMRTanpa server menggunakan peran terkait layanan bernama AWSServiceRoleForAmazonEMRServerlessuntuk memungkinkannya memanggil AWS APIsatas nama Anda.

Peran AWSServiceRoleForAmazonEMRServerless terkait layanan mempercayai layanan berikut untuk mengambil peran:

- `ops.emr-serverless.amazonaws.com`

Kebijakan izin peran bernama `AmazonEMRServerlessServiceRolePolicy` memungkinkan EMR Tanpa Server untuk menyelesaikan tindakan berikut pada sumber daya yang ditentukan.

> **Note**
>
> Konten kebijakan terkelola berubah, sehingga kebijakan yang ditampilkan di sini mungkin kedaluwarsa. Lihat up-to-date kebijakan A terbanyak mazonEMRServerless ServiceRolePolicy di AWS Management Console.

- Tindakan: `ec2:CreateNetworkInterface`
- Tindakan: `ec2:DeleteNetworkInterface`
- Tindakan: `ec2:DescribeNetworkInterfaces`
- Tindakan: `ec2:DescribeSecurityGroups`
- Tindakan: `ec2:DescribeSubnets`
- Tindakan: `ec2:DescribeVpcs`
- Tindakan: `ec2:DescribeDhcpOptions`
- Tindakan: `ec2:DescribeRouteTables`
- Tindakan: `cloudwatch:PutMetricData`

Berikut ini adalah `AmazonEMRServerlessServiceRolePolicy` kebijakan lengkapnya.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EC2PolicyStatement",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface",
                "ec2:DeleteNetworkInterface",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "ec2:DescribeDhcpOptions",
                "ec2:DescribeRouteTables"
            ],
            "Resource": "*"
        },
```

```
        {
            "Sid": "CloudWatchPolicyStatement",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringEquals": {
                    "cloudwatch:namespace": [
                        "AWS/EMRServerless",
                        "AWS/Usage"
                    ]
                }
            }
        }
    ]
}
```

Kebijakan kepercayaan berikut dilampirkan pada peran ini untuk memungkinkan prinsipal EMR Tanpa Server untuk mengambil peran ini.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "ops.emr-serverless.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Anda harus mengonfigurasi izin untuk mengizinkan IAM entitas (seperti pengguna, grup, atau peran) membuat, mengedit, atau menghapus peran terkait layanan. Untuk informasi selengkapnya, lihat Izin peran terkait layanan di Panduan Pengguna. IAM

## Membuat peran terkait layanan untuk Tanpa Server EMR

Anda tidak perlu membuat peran terkait layanan secara manual. Saat Anda membuat aplikasi EMR Tanpa Server baru di AWS Management Console (menggunakan EMR Studio), AWS CLI, atau AWS API, EMR Tanpa server menciptakan peran terkait layanan untuk Anda. Anda harus mengonfigurasi izin untuk mengizinkan IAM entitas (seperti pengguna, grup, atau peran) membuat, mengedit, atau menghapus peran terkait layanan.

Untuk membuat peran AWSServiceRoleForAmazonEMRServerless terkait layanan menggunakan IAM

Tambahkan pernyataan berikut ke kebijakan izin untuk IAM entitas yang perlu membuat peran terkait layanan.

```
{
    "Effect": "Allow",
    "Action": [
        "iam:CreateServiceLinkedRole"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/ops.emr-serverless.amazonaws.com/
AWSServiceRoleForAmazonEMRServerless*",
    "Condition": {"StringLike": {"iam:AWSServiceName": "ops.emr-
serverless.amazonaws.com"}}
}
```

Jika Anda menghapus peran tertaut layanan ini, dan ingin membuatnya lagi, Anda dapat mengulangi proses yang sama untuk membuat kembali peran tersebut di akun Anda. Saat Anda membuat aplikasi EMR Tanpa Server baru, EMR Tanpa Server membuat peran terkait layanan untuk Anda lagi.

Anda juga dapat menggunakan IAM konsol untuk membuat peran terkait layanan dengan kasus penggunaan Tanpa EMRServer. Dalam AWS CLI atau AWS API, buat peran terkait layanan dengan nama `ops.emr-serverless.amazonaws.com` layanan. Untuk informasi selengkapnya, lihat Membuat peran terkait layanan di IAMPanduan Pengguna. Jika Anda menghapus peran tertaut layanan ini, Anda dapat mengulang proses yang sama untuk membuat peran tersebut lagi.

## Mengedit peran terkait layanan untuk Tanpa Server EMR

EMRTanpa server tidak memungkinkan Anda mengedit peran AWSServiceRoleForAmazonEMRServerless terkait layanan karena berbagai entitas mungkin

mereferensikan peran tersebut. Anda tidak dapat mengedit AWS IAMKebijakan milik yang digunakan peran terkait layanan EMR Tanpa Server, karena berisi semua izin yang diperlukan yang dibutuhkan Tanpa Server. EMR Namun, Anda dapat mengedit deskripsi peran menggunakanIAM.

Untuk mengedit deskripsi peran AWSServiceRoleForAmazonEMRServerless terkait layanan menggunakan IAM

Tambahkan pernyataan berikut ke kebijakan izin untuk IAM entitas yang perlu mengedit deskripsi peran terkait layanan.

```
{
    "Effect": "Allow",
    "Action": [
        "iam: UpdateRoleDescription"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/ops.emr-serverless.amazonaws.com/
AWSServiceRoleForAmazonEMRServerless*",
    "Condition": {"StringLike": {"iam:AWSServiceName": "ops.emr-
serverless.amazonaws.com"}}
}
```

Untuk informasi selengkapnya, lihat Mengedit peran terkait layanan di IAMPanduan Pengguna.

## Menghapus peran terkait layanan untuk Tanpa Server EMR

Jika Anda tidak perlu lagi menggunakan fitur atau layanan yang memerlukan peran terkait layanan, kami merekomendasikan Anda menghapus peran tersebut. Ini agar Anda tidak memiliki entitas yang tidak terpakai yang tidak dipantau atau dipelihara secara aktif. Namun, Anda harus menghapus semua aplikasi EMR Tanpa Server di semua Wilayah sebelum dapat menghapus peran terkait layanan.

> **Note**
>
> Jika layanan EMR Tanpa Server menggunakan peran saat Anda mencoba menghapus sumber daya yang terkait dengan peran, maka penghapusan mungkin gagal. Jika hal itu terjadi, tunggu beberapa menit dan coba mengoperasikannya lagi.

Untuk menghapus peran AWSServiceRoleForAmazonEMRServerless terkait layanan menggunakan IAM

Tambahkan pernyataan berikut ke kebijakan izin untuk IAM entitas yang perlu menghapus peran terkait layanan.

```
{
    "Effect": "Allow",
    "Action": [
        "iam:DeleteServiceLinkedRole",
        "iam:GetServiceLinkedRoleDeletionStatus"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/ops.emr-serverless.amazonaws.com/
AWSServiceRoleForAmazonEMRServerless*",
    "Condition": {"StringLike": {"iam:AWSServiceName": "ops.emr-
serverless.amazonaws.com"}}
}
```

Untuk menghapus peran terkait layanan secara manual menggunakan IAM

Gunakan IAM konsol, AWS CLI, atau AWS APIuntuk menghapus peran AWSServiceRoleForAmazonEMRServerless terkait layanan. Untuk informasi selengkapnya, lihat Menghapus peran terkait layanan di Panduan Pengguna. IAM

### Wilayah yang Didukung untuk EMR peran terkait layanan Tanpa Server

EMRDukungan tanpa server menggunakan peran terkait layanan di semua Wilayah tempat layanan tersedia. Untuk informasi selengkapnya, silakan lihat AWS Wilayah dan titik akhir.

## Peran runtime Job untuk Amazon Serverless EMR

Anda dapat menentukan izin IAM peran yang dapat diasumsikan oleh menjalankan pekerjaan EMR Tanpa Server saat memanggil layanan lain atas nama Anda. Ini termasuk akses ke Amazon S3 untuk sumber data, target, serta lainnya AWS sumber daya seperti cluster Amazon Redshift dan tabel DynamoDB. Untuk mempelajari lebih lanjut tentang cara membuat peran, lihatBuat peran runtime pekerjaan.

Contoh kebijakan runtime

Anda dapat melampirkan kebijakan runtime, seperti berikut ini, ke peran runtime pekerjaan. Kebijakan runtime pekerjaan berikut memungkinkan:

- Baca akses ke ember Amazon S3 dengan sampel. EMR
- Akses penuh ke ember S3.

- Buat dan baca akses ke AWS Katalog Data Glue.

Untuk menambahkan akses ke yang lain AWS resource seperti DynamoDB, Anda harus menyertakan izin untuk mereka dalam kebijakan saat membuat peran runtime.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadAccessForEMRSamples",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::*.elasticmapreduce",
                "arn:aws:s3:::*.elasticmapreduce/*"
            ]
        },
        {
            "Sid": "FullAccessToS3Bucket",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
            ]
        },
        {
            "Sid": "GlueCreateAndReadDataCatalog",
            "Effect": "Allow",
            "Action": [
                "glue:GetDatabase",
                "glue:CreateDatabase",
                "glue:GetDataBases",
                "glue:CreateTable",
                "glue:GetTable",
```

```
                "glue:UpdateTable",
                "glue:DeleteTable",
                "glue:GetTables",
                "glue:GetPartition",
                "glue:GetPartitions",
                "glue:CreatePartition",
                "glue:BatchCreatePartition",
                "glue:GetUserDefinedFunctions"
            ],
            "Resource": ["*"]
        }
    ]
}
```

Lulus hak istimewa peran

Anda dapat melampirkan kebijakan IAM izin ke peran pengguna untuk memungkinkan pengguna hanya meneruskan peran yang disetujui. Hal ini memungkinkan administrator untuk mengontrol pengguna mana yang dapat meneruskan peran runtime pekerjaan tertentu ke pekerjaan Tanpa EMR Server. Untuk mempelajari lebih lanjut tentang menyetel izin, lihat Memberikan izin pengguna untuk meneruskan peran ke AWS layanan.

Berikut ini adalah contoh kebijakan yang memungkinkan meneruskan peran runtime pekerjaan ke prinsipal layanan EMR Tanpa Server.

```
{
     "Effect": "Allow",
     "Action": "iam:PassRole",
     "Resource": "arn:aws:iam::1234567890:role/JobRuntimeRoleForEMRServerless",
        "Condition": {
                "StringLike": {
                    "iam:PassedToService": "emr-serverless.amazonaws.com"
                }
            }
}
```

## Contoh kebijakan akses pengguna untuk Tanpa EMR Server

Anda dapat menyiapkan kebijakan berbutir halus untuk pengguna Anda tergantung pada tindakan yang ingin dilakukan setiap pengguna saat berinteraksi dengan aplikasi Tanpa Server. EMR Kebijakan berikut adalah contoh yang mungkin membantu dalam menyiapkan izin yang tepat untuk

pengguna Anda. Bagian ini hanya berfokus pada kebijakan EMR Tanpa Server. Untuk contoh kebijakan pengguna EMR Studio, lihat Mengonfigurasi izin pengguna EMR Studio. Untuk informasi tentang cara melampirkan kebijakan ke IAM pengguna (prinsipal), lihat Mengelola IAM kebijakan di Panduan Pengguna. IAM

## Kebijakan pengguna daya

Untuk memberikan semua tindakan yang diperlukan untuk EMR Tanpa Server, buat dan lampirkan `AmazonEMRServerlessFullAccess` kebijakan ke IAM pengguna, peran, atau grup yang diperlukan.

Berikut ini adalah contoh kebijakan yang memungkinkan pengguna daya untuk membuat dan memodifikasi aplikasi EMR Tanpa Server, serta melakukan tindakan lain seperti mengirimkan dan men-debug pekerjaan. Ini mengungkapkan semua tindakan yang diperlukan EMR Tanpa Server untuk layanan lain.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EMRServerlessActions",
            "Effect": "Allow",
            "Action": [
                "emr-serverless:CreateApplication",
                "emr-serverless:UpdateApplication",
                "emr-serverless:DeleteApplication",
                "emr-serverless:ListApplications",
                "emr-serverless:GetApplication",
                "emr-serverless:StartApplication",
                "emr-serverless:StopApplication",
                "emr-serverless:StartJobRun",
                "emr-serverless:CancelJobRun",
                "emr-serverless:ListJobRuns",
                "emr-serverless:GetJobRun"
            ],
            "Resource": "*"
        }
    ]
}
```

Saat Anda mengaktifkan konektivitas jaringan ke AndaVPC, aplikasi EMR Tanpa Server membuat antarmuka jaringan EC2 elastis Amazon (ENIs) untuk berkomunikasi dengan sumber daya. VPC

Kebijakan berikut memastikan bahwa setiap EC2 ENIs yang baru hanya dibuat dalam konteks aplikasi EMR Tanpa Server.

> **Note**
>
> Kami sangat menyarankan untuk menetapkan kebijakan ini untuk memastikan bahwa pengguna tidak dapat membuat EC2 ENIs kecuali dalam konteks peluncuran aplikasi EMR Tanpa Server.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowEC2ENICreationWithEMRTags",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:network-interface/*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:CalledViaLast": "ops.emr-serverless.amazonaws.com"
                }
            }
        }
}
```

Jika Anda ingin membatasi akses EMR Tanpa Server ke subnet tertentu, Anda dapat menandai setiap subnet dengan kondisi tag. IAMKebijakan ini memastikan bahwa aplikasi EMR Tanpa Server hanya dapat membuat EC2 ENIs dalam subnet yang diizinkan.

```
{
    "Sid": "AllowEC2ENICreationInSubnetAndSecurityGroupWithEMRTags",
    "Effect": "Allow",
    "Action": [
        "ec2:CreateNetworkInterface"
    ],
    "Resource": [
```

```
            "arn:aws:ec2:*:*:subnet/*",
            "arn:aws:ec2:*:*:security-group/*"
        ],
        "Condition": {
            "StringEquals": {
                "aws:ResourceTag/KEY": "VALUE"
            }
        }
}
```

> **Important**
>
> Jika Anda seorang Administrator atau pengguna daya yang membuat aplikasi pertama, Anda harus mengonfigurasi kebijakan izin untuk memungkinkan Anda membuat peran terkait layanan EMR Tanpa Server. Untuk mempelajari selengkapnya, lihat Menggunakan peran terkait layanan untuk Tanpa Server EMR.

IAMKebijakan berikut memungkinkan Anda untuk membuat peran terkait layanan EMR Tanpa Server untuk akun Anda.

```
{
   "Sid":"AllowEMRServerlessServiceLinkedRoleCreation",
   "Effect":"Allow",
   "Action":"iam:CreateServiceLinkedRole",
   "Resource":"arn:aws:iam::account-id:role/aws-service-role/ops.emr-
serverless.amazonaws.com/AWSServiceRoleForAmazonEMRServerless"
}
```

## Kebijakan insinyur data

Berikut ini adalah contoh kebijakan yang memungkinkan pengguna izin hanya-baca pada aplikasi EMR Tanpa Server, serta kemampuan untuk mengirimkan dan men-debug pekerjaan. Perlu diingat bahwa karena kebijakan ini tidak secara eksplisit menolak tindakan, pernyataan kebijakan yang berbeda masih dapat digunakan untuk memberikan akses ke tindakan tertentu.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EMRServerlessActions",
```

```
            "Effect": "Allow",
            "Action": [
                "emr-serverless:ListApplications",
                "emr-serverless:GetApplication",
                "emr-serverless:StartApplication",
                "emr-serverless:StartJobRun",
                "emr-serverless:CancelJobRun",
                "emr-serverless:ListJobRuns",
                "emr-serverless:GetJobRun"
            ],
            "Resource": "*"
        }
    ]
}
```

## Menggunakan tag untuk kontrol akses

Anda dapat menggunakan kondisi tag untuk kontrol akses berbutir halus. Misalnya, Anda dapat membatasi pengguna dari satu tim sehingga mereka hanya dapat mengirimkan pekerjaan ke aplikasi EMR Tanpa Server yang ditandai dengan nama tim mereka.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EMRServerlessActions",
            "Effect": "Allow",
            "Action": [
                "emr-serverless:ListApplications",
                "emr-serverless:GetApplication",
                "emr-serverless:StartApplication",
                "emr-serverless:StartJobRun",
                "emr-serverless:CancelJobRun",
                "emr-serverless:ListJobRuns",
                "emr-serverless:GetJobRun"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Team": "team-name"
                }
            }
        }
```

```
    ]
}
```

## Kebijakan untuk kendali akses berbasis tanda

Anda dapat menggunakan kondisi dalam kebijakan berbasis identitas untuk mengontrol akses ke aplikasi dan pekerjaan berjalan berdasarkan tag.

Contoh berikut menunjukkan skenario dan cara yang berbeda untuk menggunakan operator kondisi dengan kunci kondisi EMR Tanpa Server. Pernyataan IAM kebijakan ini dimaksudkan untuk tujuan demonstrasi saja dan tidak boleh digunakan dalam lingkungan produksi. Ada beberapa cara untuk menggabungkan pernyataan kebijakan untuk memberikan dan menolak izin sesuai dengan kebutuhan Anda. Untuk informasi selengkapnya tentang IAM kebijakan perencanaan dan pengujian, lihat Panduan IAM pengguna.

> **Important**
>
> Secara eksplisit menolak izin untuk tindakan penandaan adalah pertimbangan penting. Hal ini mencegah pengguna dari penandaan sumber daya dan dengan demikian memberikan sendiri izin yang tidak ingin Anda berikan. Jika tindakan penandaan untuk sumber daya tidak ditolak, pengguna dapat memodifikasi tanda dan menghindari maksud kebijakan berbasis tanda. Untuk contoh kebijakan yang menolak tindakan penandaan, lihat Tolak akses untuk menambah dan menghapus tanda.

Contoh di bawah ini menunjukkan kebijakan izin berbasis identitas yang digunakan untuk mengontrol tindakan yang diizinkan dengan aplikasi Tanpa Server. EMR

### Izinkan tindakan hanya pada sumber daya dengan nilai tanda tertentu

Dalam contoh kebijakan berikut, operator `StringEquals` kondisi mencoba mencocokkan `dev` dengan nilai untuk departemen tag. Jika departemen tag belum ditambahkan ke aplikasi, atau tidak berisi nilai`dev`, kebijakan tidak berlaku, dan tindakan tidak diizinkan oleh kebijakan ini. Jika tidak ada pernyataan kebijakan lain yang mengizinkan tindakan, pengguna hanya dapat bekerja dengan aplikasi yang memiliki tag ini dengan nilai ini.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```
      "Effect": "Allow",
      "Action": [
        "emr-serverless:GetApplication"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "emr-serverless:ResourceTag/department": "dev"
        }
      }
    }
  ]
}
```

Anda juga dapat menentukan beberapa nilai tanda menggunakan operator syarat. Misalnya, untuk mengizinkan tindakan pada aplikasi di mana `department` tag berisi nilai `dev` atau`test`, Anda dapat mengganti blok kondisi pada contoh sebelumnya dengan yang berikut ini.

```
"Condition": {
        "StringEquals": {
          "emr-serverless:ResourceTag/department": ["dev", "test"]
        }
      }
```

## Memerlukan penandaan ketika sumber daya dibuat

Pada contoh di bawah ini, tag perlu diterapkan saat membuat aplikasi.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-serverless:CreateApplication"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "emr-serverless:RequestTag/department": "dev"
        }
      }
```

```
    }
  ]
}
```

Pernyataan kebijakan berikut memungkinkan pengguna untuk membuat aplikasi hanya jika aplikasi memiliki `department` tag, yang dapat berisi nilai apa pun.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-serverless:CreateApplication"
      ],
      "Resource": "*",
      "Condition": {
        "Null": {
          "emr-serverless:RequestTag/department": "false"
        }
      }
    }
  ]
}
```

## Tolak akses untuk menambah dan menghapus tanda

Kebijakan ini mencegah pengguna menambahkan atau menghapus tag pada aplikasi EMR Tanpa Server dengan `department` tag yang nilainya tidak. `dev`

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "emr-serverless:TagResource",
        "emr-serverless:UntagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
```

```
          "emr-serverless:ResourceTag/department": "dev"
        }
      }
    }
  ]
}
```

## Contoh kebijakan berbasis identitas untuk Tanpa Server EMR

Secara default, pengguna dan peran tidak memiliki izin untuk membuat atau memodifikasi sumber daya Amazon EMR Tanpa Server. Mereka juga tidak dapat melakukan tugas dengan menggunakan AWS Management Console, AWS Command Line Interface (AWS CLI), atau AWS API. Untuk memberikan izin kepada pengguna untuk melakukan tindakan pada sumber daya yang mereka butuhkan, IAM administrator dapat membuat IAM kebijakan. Administrator kemudian dapat menambahkan IAM kebijakan ke peran, dan pengguna dapat mengambil peran.

Untuk mempelajari cara membuat kebijakan IAM berbasis identitas menggunakan contoh dokumen kebijakan ini, lihat Membuat JSON IAM kebijakan di Panduan Pengguna. IAM

Untuk detail tentang tindakan dan jenis sumber daya yang ditentukan oleh Amazon EMR Tanpa Server, termasuk format ARNs untuk setiap jenis sumber daya, lihat Kunci tindakan, sumber daya, dan kondisi untuk Amazon EMR Tanpa Server di Referensi Otorisasi Layanan.

Topik

- Praktik terbaik kebijakan
- Mengizinkan pengguna melihat izin mereka sendiri

### Praktik terbaik kebijakan

> **Note**
>
> EMRTanpa server tidak mendukung kebijakan terkelola, jadi praktik pertama yang tercantum di bawah ini tidak berlaku.

Kebijakan berbasis identitas menentukan apakah seseorang dapat membuat, mengakses, atau menghapus sumber daya Amazon EMR Tanpa Server di akun Anda. Tindakan ini dapat menimbulkan biaya untuk Akun AWS. Saat Anda membuat atau mengedit kebijakan berbasis identitas, ikuti panduan dan rekomendasi berikut:

- Memulai dengan AWS kebijakan terkelola dan beralih ke izin hak istimewa paling sedikit — Untuk memulai pemberian izin kepada pengguna dan beban kerja Anda, gunakan AWS kebijakan terkelola yang memberikan izin untuk banyak kasus penggunaan umum. Mereka tersedia di Akun AWS. Kami menyarankan Anda mengurangi izin lebih lanjut dengan mendefinisikan AWS kebijakan yang dikelola pelanggan yang khusus untuk kasus penggunaan Anda. Untuk informasi selengkapnya, silakan lihat AWS kebijakan terkelola atau AWS kebijakan terkelola untuk fungsi pekerjaan di Panduan IAM Pengguna.
- Menerapkan izin hak istimewa paling sedikit — Saat Anda menetapkan izin dengan IAM kebijakan, berikan hanya izin yang diperlukan untuk melakukan tugas. Anda melakukannya dengan mendefinisikan tindakan yang dapat diambil pada sumber daya tertentu dalam kondisi tertentu, yang juga dikenal sebagai izin dengan hak akses paling rendah. Untuk informasi selengkapnya tentang penggunaan IAM untuk menerapkan izin, lihat Kebijakan dan izin IAM di IAM Panduan Pengguna.
- Gunakan ketentuan dalam IAM kebijakan untuk membatasi akses lebih lanjut — Anda dapat menambahkan kondisi ke kebijakan Anda untuk membatasi akses ke tindakan dan sumber daya. Misalnya, Anda dapat menulis kondisi kebijakan untuk menentukan bahwa semua permintaan harus dikirim menggunakanSSL. Anda juga dapat menggunakan ketentuan untuk memberikan akses ke tindakan layanan jika digunakan melalui tindakan tertentu Layanan AWS, seperti AWS CloudFormation. Untuk informasi selengkapnya, lihat elemen IAM JSON kebijakan: Kondisi dalam Panduan IAM Pengguna.
- Gunakan IAM Access Analyzer untuk memvalidasi IAM kebijakan Anda guna memastikan izin yang aman dan fungsional — IAM Access Analyzer memvalidasi kebijakan baru dan yang sudah ada sehingga kebijakan mematuhi bahasa IAM kebijakan () JSON dan praktik terbaik. IAM IAMAccess Analyzer menyediakan lebih dari 100 pemeriksaan kebijakan dan rekomendasi yang dapat ditindaklanjuti untuk membantu Anda membuat kebijakan yang aman dan fungsional. Untuk informasi selengkapnya, lihat Validasi kebijakan IAM Access Analyzer di IAMPanduan Pengguna.
- Memerlukan otentikasi multi-faktor (MFA) - Jika Anda memiliki skenario yang mengharuskan IAM pengguna atau pengguna root di Akun AWS, nyalakan MFA untuk keamanan tambahan. Untuk meminta MFA kapan API operasi dipanggil, tambahkan MFA kondisi ke kebijakan Anda. Untuk informasi selengkapnya, lihat Mengonfigurasi API akses MFA yang dilindungi di IAMPanduan Pengguna.

Untuk informasi selengkapnya tentang praktik terbaik diIAM, lihat Praktik terbaik keamanan IAM di Panduan IAM Pengguna.

## Mengizinkan pengguna melihat izin mereka sendiri

Contoh ini menunjukkan cara Anda membuat kebijakan yang memungkinkan IAM pengguna melihat kebijakan sebaris dan terkelola yang dilampirkan pada identitas pengguna mereka. Kebijakan ini mencakup izin untuk menyelesaikan tindakan ini di konsol atau secara terprogram menggunakan AWS CLI atau AWS API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

# Pembaruan Amazon EMR Tanpa Server ke AWS kebijakan terkelola

Lihat detail tentang pembaruan AWS kebijakan terkelola untuk Amazon EMR Tanpa Server sejak layanan ini mulai melacak perubahan ini. Untuk peringatan otomatis tentang perubahan pada halaman ini, berlangganan RSS feed di halaman riwayat Dokumen EMR Tanpa Server Amazon.

| Perubahan | Deskripsi | Tanggal |
| --- | --- | --- |
| A mazonEMRServerless ServiceRolePolicy — Perbarui ke kebijakan yang ada | Amazon EMR Serverless menambahkan yang baru `Sid` `CloudWatchPolicySt atement` dan `EC2Policy Statement` ke kebijakan A mazonEMRServerless ServiceRolePolicy . | Januari 25, 2024 |
| A mazonEMRServerless ServiceRolePolicy — Perbarui ke kebijakan yang ada | Amazon EMR Serverles s menambahkan izin baru untuk memungkinkan Amazon Tanpa EMR Server mempublik asikan metrik akun agregat untuk penggunaan v di namespace. CPU `"AWS/Usag e"` | 20 April 2023 |
| Amazon EMR Serverless mulai melacak perubahan | Amazon EMR Serverless mulai melacak perubahannya AWS kebijakan terkelola. | 20 April 2023 |

# Memecahkan masalah identitas dan akses Amazon EMR Tanpa Server

Gunakan informasi berikut untuk membantu Anda mendiagnosis dan memperbaiki masalah umum yang mungkin Anda temui saat bekerja dengan Amazon EMR Tanpa Server dan. IAM

Topik

- Saya tidak berwenang untuk melakukan tindakan di Amazon Tanpa EMR Server

- Saya tidak berwenang untuk melakukan iam: PassRole
- Saya ingin mengizinkan orang di luar saya AWS akun untuk mengakses sumber daya Tanpa EMR Server Amazon saya

## Saya tidak berwenang untuk melakukan tindakan di Amazon Tanpa EMR Server

Jika AWS Management Console memberitahu Anda bahwa Anda tidak berwenang untuk melakukan suatu tindakan, maka Anda harus menghubungi administrator Anda untuk bantuan. Administrator Anda adalah orang yang memberikan nama pengguna dan kata sandi Anda.

Contoh kesalahan berikut terjadi ketika pengguna `mateojackson` mencoba menggunakan konsol untuk melihat detail tentang suatu sumber daya *`my-example-widget`* fiktif, tetapi tidak memiliki izin `emr-serverless:`*`GetWidget`* fiktif.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform: emr-
serverless:GetWidget on resource: my-example-widget
```

Dalam hal ini, Mateo meminta administratornya untuk memperbarui kebijakannya untuk mengizinkan dia mengakses sumber daya *`my-example-widget`* menggunakan tindakan `emr-serverless:`*`GetWidget`*.

## Saya tidak berwenang untuk melakukan iam: PassRole

Jika Anda menerima kesalahan yang tidak diizinkan untuk melakukan `iam:PassRole` tindakan, kebijakan Anda harus diperbarui agar Anda dapat meneruskan peran ke Amazon Tanpa EMR Server.

Beberapa Layanan AWS memungkinkan Anda untuk meneruskan peran yang ada ke layanan tersebut alih-alih membuat peran layanan baru atau peran terkait layanan. Untuk melakukannya, Anda harus memiliki izin untuk meneruskan peran ke layanan.

Contoh kesalahan berikut terjadi ketika IAM pengguna bernama `marymajor` mencoba menggunakan konsol untuk melakukan tindakan di Amazon Tanpa EMR Server. Namun, tindakan tersebut memerlukan layanan untuk mendapatkan izin yang diberikan oleh peran layanan. Mary tidak memiliki izin untuk meneruskan peran tersebut pada layanan.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Dalam kasus ini, kebijakan Mary harus diperbarui agar dia mendapatkan izin untuk melakukan tindakan `iam:PassRole` tersebut.

Jika Anda membutuhkan bantuan, hubungi AWS administrator. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

### Saya ingin mengizinkan orang di luar saya AWS akun untuk mengakses sumber daya Tanpa EMR Server Amazon saya

Anda dapat membuat peran yang dapat digunakan pengguna di akun lain atau orang-orang di luar organisasi Anda untuk mengakses sumber daya Anda. Anda dapat menentukan siapa saja yang dipercaya untuk mengambil peran tersebut. Untuk layanan yang mendukung kebijakan berbasis sumber daya atau daftar kontrol akses (ACLs), Anda dapat menggunakan kebijakan tersebut untuk memberi orang akses ke sumber daya Anda.

Untuk mempelajari selengkapnya, periksa referensi berikut:

- Untuk mengetahui apakah Amazon EMR Serverless mendukung fitur-fitur ini, lihat. Identity and Access Management (IAM) di Amazon Tanpa EMR Server
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda Akun AWS yang Anda miliki, lihat Menyediakan akses ke IAM pengguna di pengguna lain Akun AWS yang Anda miliki di Panduan IAM Pengguna.
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda ke pihak ketiga Akun AWS, lihat Menyediakan akses ke Akun AWS dimiliki oleh pihak ketiga dalam Panduan IAM Pengguna.
- Untuk mempelajari cara menyediakan akses melalui federasi identitas, lihat Menyediakan akses ke pengguna yang diautentikasi secara eksternal (federasi identitas) di Panduan Pengguna. IAM
- Untuk mempelajari perbedaan antara menggunakan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Akses sumber daya lintas akun di IAM Panduan Pengguna. IAM

# Menggunakan EMR Serverless dengan AWS Lake Formation untuk kontrol akses berbutir halus

## Gambaran Umum

Dengan Amazon EMR merilis 7.2.0 dan lebih tinggi, Anda dapat memanfaatkan AWS Lake Formation untuk menerapkan kontrol akses berbutir halus pada tabel Katalog Data yang didukung oleh S3.

Kemampuan ini memungkinkan Anda mengonfigurasi kontrol akses tingkat tabel, baris, kolom, dan sel untuk read kueri dalam pekerjaan Amazon EMR Serverless Spark Anda. Untuk mengonfigurasi kontrol akses berbutir halus untuk pekerjaan batch Apache Spark dan sesi interaktif, gunakan Studio. EMR Lihat bagian berikut untuk mempelajari lebih lanjut tentang Lake Formation dan cara menggunakannya dengan Tanpa EMR Server.

Menggunakan Amazon Tanpa EMR Server dengan AWS Lake Formation dikenakan biaya tambahan. Untuk informasi selengkapnya, lihat EMRharga Amazon.

## Bagaimana EMR Serverless bekerja dengan AWS Lake Formation

Menggunakan EMR Tanpa Server dengan Lake Formation memungkinkan Anda menerapkan lapisan izin pada setiap pekerjaan Spark untuk menerapkan kontrol izin Lake Formation saat Tanpa Server mengeksekusi pekerjaan. EMR EMRTanpa server menggunakan profil sumber daya Spark untuk membuat dua profil untuk menjalankan pekerjaan secara efektif. Profil pengguna mengeksekusi kode yang disediakan pengguna, sementara profil sistem memberlakukan kebijakan Lake Formation. Untuk informasi lebih lanjut, lihat Apa itu AWS Lake Formationdan Pertimbangan dan keterbatasan.

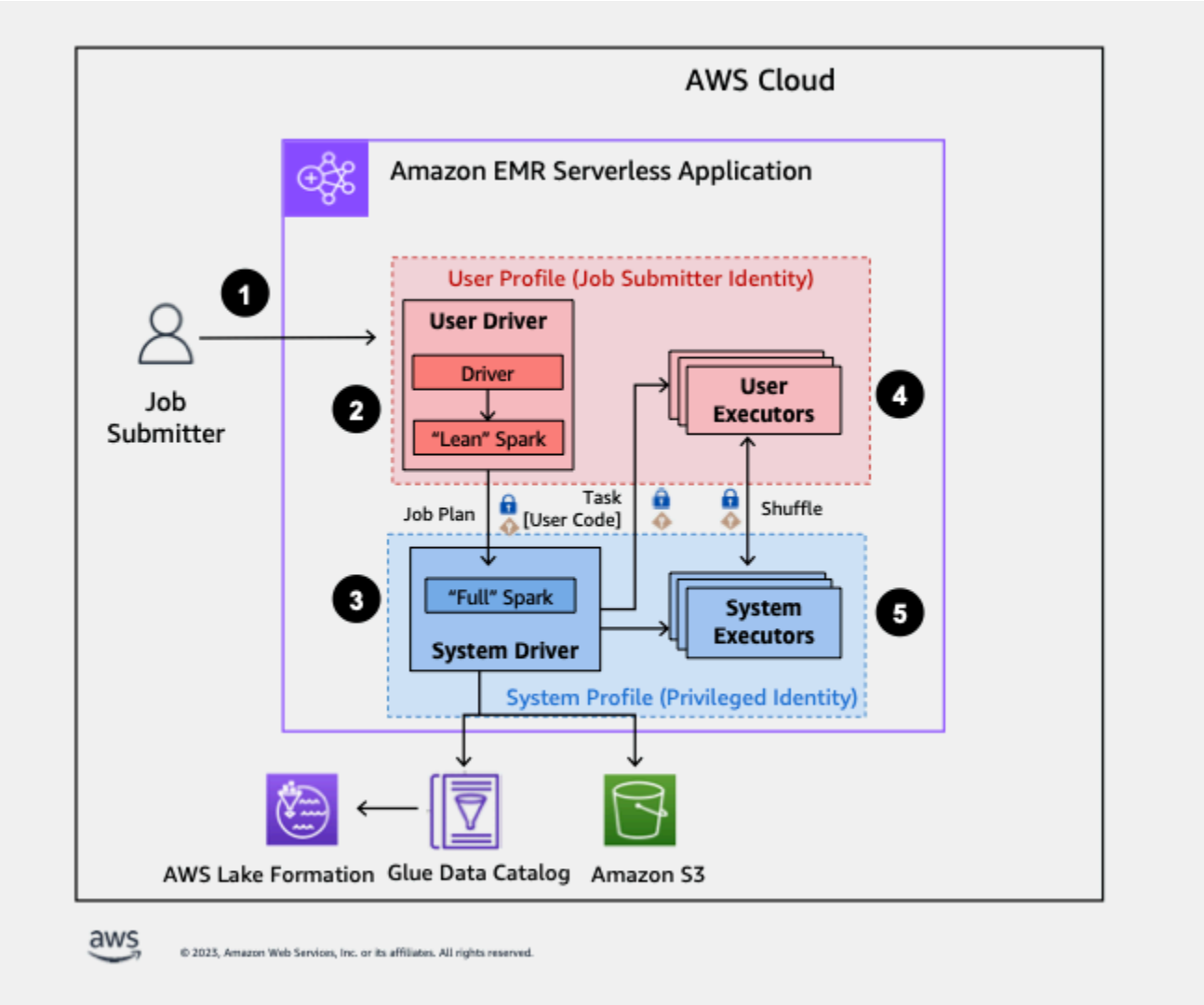
Saat Anda menggunakan kapasitas pra-inisialisasi dengan Lake Formation, kami sarankan Anda memiliki minimal dua driver Spark. Setiap pekerjaan yang mendukung Lake Formation menggunakan dua driver Spark, satu untuk profil pengguna dan satu untuk profil sistem. Untuk kinerja terbaik, Anda harus menggunakan dua kali lipat jumlah driver untuk pekerjaan yang mendukung Lake Formation dibandingkan jika Anda tidak menggunakan Lake Formation.

Saat Anda menjalankan pekerjaan Spark di EMR Tanpa Server, Anda juga harus mempertimbangkan dampak alokasi dinamis pada manajemen sumber daya dan kinerja klaster. Konfigurasi `spark.dynamicAllocation.maxExecutors` jumlah maksimum pelaksana per profil sumber daya berlaku untuk pengguna dan pelaksana sistem. Jika Anda mengonfigurasi angka tersebut agar sama dengan jumlah maksimum pelaksana yang diizinkan, pekerjaan Anda mungkin macet karena satu jenis pelaksana yang menggunakan semua sumber daya yang tersedia, yang mencegah pelaksana lain saat Anda menjalankan pekerjaan.

Agar Anda tidak kehabisan sumber daya, EMR Tanpa Server menetapkan jumlah maksimum pelaksana default per profil sumber daya menjadi 90% dari nilai. `spark.dynamicAllocation.maxExecutors` Anda dapat mengganti konfigurasi ini ketika Anda menentukan `spark.dynamicAllocation.maxExecutorsRatio` dengan nilai antara 0 dan 1. Selain itu, Anda juga dapat mengonfigurasi properti berikut untuk mengoptimalkan alokasi sumber daya dan kinerja keseluruhan:

- `spark.dynamicAllocation.cachedExecutorIdleTimeout`
- `spark.dynamicAllocation.shuffleTracking.timeout`
- `spark.cleaner.periodicGC.interval`

Berikut ini adalah ikhtisar tingkat tinggi tentang bagaimana EMR Tanpa Server mendapatkan akses ke data yang dilindungi oleh kebijakan keamanan Lake Formation.




1. Seorang pengguna mengirimkan pekerjaan Spark ke AWS Lake Formation-mengaktifkan aplikasi EMR Serverless.
2. EMRTanpa server mengirimkan pekerjaan ke driver pengguna dan menjalankan pekerjaan di profil pengguna. Driver pengguna menjalankan versi lean Spark yang tidak memiliki kemampuan untuk meluncurkan tugas, meminta pelaksana, mengakses S3 atau Glue Catalog. Ini membangun rencana kerja.

3. EMRTanpa server mengatur driver kedua yang disebut driver sistem dan menjalankannya di profil sistem (dengan identitas istimewa). EMRTanpa server menyiapkan TLS saluran terenkripsi antara dua driver untuk komunikasi. Driver pengguna menggunakan saluran untuk mengirim rencana pekerjaan ke driver sistem. Driver sistem tidak menjalankan kode yang dikirimkan pengguna. Ini menjalankan Spark penuh dan berkomunikasi dengan S3, dan Katalog Data untuk akses data. Ini meminta pelaksana dan mengkompilasi Job Plan ke dalam urutan tahapan eksekusi.
4. EMRServerless kemudian menjalankan tahapan pada pelaksana dengan driver pengguna atau driver sistem. Kode pengguna dalam tahap apa pun dijalankan secara eksklusif pada pelaksana profil pengguna.
5. Tahapan yang membaca data dari tabel Katalog Data dilindungi oleh AWS Lake Formation atau yang menerapkan filter keamanan didelegasikan ke pelaksana sistem.

## Mengaktifkan Lake Formation di Amazon EMR

Untuk mengaktifkan Lake Formation, Anda harus mengatur `spark.emr-serverless.lakeformation.enabled` ke `true` bawah `spark-defaults` klasifikasi untuk parameter konfigurasi runtime saat membuat aplikasi Tanpa EMR Server.

```
aws emr-serverless create-application \
    --release-label emr-7.2.0 \
    --runtime-configuration '{
     "classification": "spark-defaults",
     "properties": {
      "spark.emr-serverless.lakeformation.enabled": "true"
      }
    }' \
    --type "SPARK"
```

Anda juga dapat mengaktifkan Lake Formation saat membuat aplikasi baru di EMR Studio. Pilih Gunakan Lake Formation untuk kontrol akses berbutir halus, tersedia di bawah Konfigurasi tambahan.

Enkripsi antar pekerja diaktifkan secara default saat Anda menggunakan Lake Formation dengan EMR Tanpa Server, jadi Anda tidak perlu mengaktifkan enkripsi antar-pekerja secara eksplisit lagi.

Mengaktifkan Lake Formation untuk pekerjaan Spark

Untuk mengaktifkan Lake Formation untuk pekerjaan Spark individual, setel `spark.emr-serverless.lakeformation.enabled` ke true saat menggunakan`spark-submit`.

```
--conf spark.emr-serverless.lakeformation.enabled=true
```

## Izin peran IAM runtime Job

Izin Lake Formation mengontrol akses ke AWS Sumber daya Glue Data Catalog, lokasi Amazon S3, dan data dasar di lokasi tersebut. IAMizin mengontrol akses ke Lake Formation dan AWS Glue APIs dan sumber daya. Meskipun Anda mungkin memiliki izin Lake Formation untuk mengakses tabel di Data Catalog (SELECT), operasi Anda gagal jika Anda tidak memiliki IAM izin pada `glue:Get*` API operasi.

Berikut ini adalah contoh kebijakan tentang cara memberikan IAM izin untuk mengakses skrip di S3, mengunggah log ke S3, AWS APIIzin Glue, dan izin untuk mengakses Lake Formation.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ScriptAccess",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
            "arn:aws:s3:::*.DOC-EXAMPLE-BUCKET/scripts",
            "arn:aws:s3:::*.DOC-EXAMPLE-BUCKET/*" ]
        },
        {
         "Sid": "LoggingAccess",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
               "arn:aws:s3:::DOC-EXAMPLE-BUCKET/logs/*"
            ]
        },
        {
```

```
            "Sid": "GlueCatalogAccess",
            "Effect": "Allow",
            "Action": [
                 "glue:Get*",
                 "glue:Create*",
                 "glue:Update*"
            ],
            "Resource": ["*"]
        },
        {
            "Sid": "LakeFormationAccess",
            "Effect": "Allow",
            "Action": [
             "lakeformation:GetDataAccess"
             ],
            "Resource": ["*"]
        }
    ]
}
```

## Menyiapkan izin Lake Formation untuk peran runtime pekerjaan

Pertama, daftarkan lokasi meja Hive Anda dengan Lake Formation. Kemudian buat izin untuk peran runtime pekerjaan Anda di tabel yang Anda inginkan. Untuk detail lebih lanjut tentang Lake Formation, lihat Apa itu AWS Lake Formation? di AWS Lake Formation Panduan Pengembang.

Setelah mengatur izin Lake Formation, Anda dapat mengirimkan pekerjaan Spark di Amazon EMR Tanpa Server. Untuk informasi selengkapnya tentang pekerjaan Spark, lihat contoh Spark.

## Mengirimkan pekerjaan

Setelah Anda selesai menyiapkan hibah Lake Formation, Anda dapat mengirimkan pekerjaan Spark di EMR Tanpa Server. Untuk menjalankan pekerjaan Iceberg, Anda harus menyediakan properti berikut`spark-submit`.

```
--conf spark.sql.catalog.spark_catalog=org.apache.iceberg.spark.SparkSessionCatalog
--conf spark.sql.catalog.spark_catalog.warehouse=<S3_DATA_LOCATION>
--conf spark.sql.catalog.spark_catalog.glue.account-id=<ACCOUNT_ID>
--conf spark.sql.catalog.spark_catalog.client.region=<REGION>
--conf spark.sql.catalog.spark_catalog.glue.endpoint=https://
glue.<REGION>.amazonaws.com
```

# Dukungan format meja terbuka

Amazon EMR rilis 7.2.0 mencakup dukungan untuk kontrol akses berbutir halus berdasarkan Lake Formation. EMRServerless mendukung jenis tabel Hive dan Iceberg. Tabel berikut menjelaskan semua operasi yang didukung.

| Operasi | Hive | Gunung es |
| --- | --- | --- |
| DDLperintah | Dengan izin IAM peran saja | Dengan izin IAM peran saja |
| Kueri tambahan | Tidak berlaku | Sepenuhnya didukung |
| Pertanyaan perjalanan waktu | Tidak berlaku untuk format tabel ini | Sepenuhnya didukung |
| Tabel metadata | Tidak berlaku untuk format tabel ini | Didukung, tetapi tabel tertentu disembunyikan. Lihat pertimbangan dan batasan untuk informasi lebih lanjut. |
| `DML INSERT` | Hanya dengan IAM izin | Hanya dengan IAM izin |
| DML UPDATE | Tidak berlaku untuk format tabel ini | Hanya dengan IAM izin |
| `DML DELETE` | Tidak berlaku untuk format tabel ini | Hanya dengan IAM izin |
| Operasi baca | Sepenuhnya didukung | Sepenuhnya didukung |
| Prosedur tersimpan | Tidak berlaku | Didukung dengan pengecualian `register_table` dan`migrate`. Lihat pertimbangan dan batasan untuk informasi lebih lanjut. |

# Pertimbangan dan batasan

Pertimbangkan pertimbangan dan batasan berikut saat Anda menggunakan Lake Formation with EMR Serverless.

> **Note**
>
> Saat Anda mengaktifkan Lake Formation untuk pekerjaan Spark di EMR Tanpa Server, pekerjaan tersebut meluncurkan driver sistem dan driver pengguna. Jika Anda menentukan kapasitas pra-inisialisasi saat peluncuran, ketentuan driver dari kapasitas pra-inisialisasi, dan jumlah driver sistem sama dengan jumlah driver pengguna yang Anda tentukan. Jika Anda memilih kapasitas On Demand, EMR Serverless meluncurkan driver sistem selain driver pengguna. Untuk memperkirakan biaya yang terkait dengan pekerjaan EMR Tanpa Server dengan Lake Formation Anda, gunakan AWS Pricing Calculator.

Amazon EMR Tanpa Server dengan Lake Formation tersedia di semua Wilayah Tanpa EMRServer yang didukung kecuali AWS GovCloud (AS-Timur) dan AWS GovCloud (AS-Barat).

- Amazon EMR Serverless mendukung kontrol akses berbutir halus melalui Lake Formation hanya untuk tabel Apache Hive dan Apache Iceberg. Format Apache Hive termasuk Parket,ORC, dan xSV.
- Aplikasi yang mendukung Lake Formation tidak mendukung penggunaan gambar Tanpa EMRServer yang disesuaikan.
- Anda tidak dapat mematikan `DynamicResourceAllocation` untuk pekerjaan Lake Formation.
- Anda hanya dapat menggunakan Lake Formation dengan pekerjaan Spark.
- EMRTanpa server dengan Lake Formation hanya mendukung satu sesi Spark selama pekerjaan.
- EMRTanpa server dengan Lake Formation hanya mendukung kueri tabel lintas akun yang dibagikan melalui tautan sumber daya.
- Berikut ini tidak didukung:
  - Kumpulan data terdistribusi yang tangguh () RDD
  - Streaming percikan
  - Menulis dengan izin yang diberikan Lake Formation
  - Kontrol akses untuk kolom bersarang

- EMRTanpa server memblokir fungsionalitas yang mungkin merusak isolasi lengkap driver sistem, termasuk yang berikut ini:
  - UDTs, HiveUDFs, dan fungsi apa pun yang ditentukan pengguna yang melibatkan kelas khusus
  - Sumber data kustom
  - Pasokan stoples tambahan untuk ekstensi Spark, konektor, atau metastore
  - Perintah `ANALYZE TABLE`
- Untuk menegakkan kontrol akses, `EXPLAIN PLAN` dan DDL operasi seperti `DESCRIBE TABLE` tidak mengekspos informasi terbatas.
- EMRTanpa server membatasi akses ke driver sistem Spark log pada aplikasi yang mendukung Lake Formation. Karena driver sistem berjalan dengan lebih banyak akses, peristiwa dan log yang dihasilkan driver sistem dapat mencakup informasi sensitif. Untuk mencegah pengguna atau kode yang tidak sah mengakses data sensitif ini, EMR Tanpa Server menonaktifkan akses ke log driver sistem. Untuk pemecahan masalah, hubungi AWS dukungan.
- Jika Anda mendaftarkan lokasi tabel dengan Lake Formation, jalur akses data akan melewati kredensil yang disimpan Lake Formation terlepas dari IAM izin untuk peran runtime pekerjaan EMR Tanpa Server. Jika Anda salah mengonfigurasi peran yang terdaftar dengan lokasi tabel, pekerjaan yang dikirimkan yang menggunakan peran dengan IAM izin S3 ke lokasi tabel akan gagal.
- Menulis ke tabel Lake Formation menggunakan IAM izin daripada izin yang diberikan Lake Formation. Jika peran runtime pekerjaan Anda memiliki izin S3 yang diperlukan, Anda dapat menggunakannya untuk menjalankan operasi penulisan.

Berikut ini adalah pertimbangan dan batasan saat menggunakan Apache Iceberg:

- Anda hanya dapat menggunakan Apache Iceberg dengan katalog sesi dan tidak sewenang-wenang bernama katalog.
- Tabel gunung es yang terdaftar di Lake Formation hanya mendukung tabel metadata`history`,,,, `metadata_log_entries``snapshots`, `files` dan. `manifests` `refs` Amazon EMR menyembunyikan kolom yang mungkin memiliki data sensitif, sepert`partitions`,`path`, dan`summaries`. Batasan ini tidak berlaku untuk tabel Gunung Es yang tidak terdaftar di Lake Formation.
- Tabel yang tidak Anda daftarkan di Lake Formation mendukung semua prosedur yang disimpan Gunung Es. Prosedur `register_table` dan `migrate` prosedur tidak didukung untuk tabel apa pun.
- Kami menyarankan Anda menggunakan Iceberg DataFrameWriter V2 alih-alih V1.

# Pemecahan Masalah

Lihat bagian berikut untuk solusi pemecahan masalah.

## Pencatatan log

EMRTanpa server menggunakan profil sumber daya Spark untuk membagi eksekusi pekerjaan. EMRTanpa server menggunakan profil pengguna untuk menjalankan kode yang Anda berikan, sementara profil sistem memberlakukan kebijakan Lake Formation. Anda dapat mengakses log untuk tugas yang dijalankan sebagai profil pengguna.

## UI Langsung dan Server Sejarah Spark

Live UI dan Spark History Server memiliki semua peristiwa Spark yang dihasilkan dari profil pengguna dan peristiwa yang disunting yang dihasilkan dari driver sistem.

Anda dapat melihat semua tugas dari driver pengguna dan sistem di tab Executors. Namun, tautan log hanya tersedia untuk profil pengguna. Selain itu, beberapa informasi disunting dari Live UI, seperti jumlah catatan keluaran.

## Job gagal dengan izin Lake Formation yang tidak mencukupi

Pastikan peran runtime pekerjaan Anda memiliki izin untuk dijalankan SELECT dan DESCRIBE di atas meja yang Anda akses.

## Job dengan RDD eksekusi gagal

EMRTanpa server saat ini tidak mendukung operasi dataset (RDD) terdistribusi yang tangguh pada pekerjaan yang mendukung Lake Formation.

## Tidak dapat mengakses file data di Amazon S3

Pastikan Anda telah mendaftarkan lokasi data lake di Lake Formation.

## Pengecualian validasi keamanan

EMRTanpa server mendeteksi kesalahan validasi keamanan. Kontak AWS dukungan untuk bantuan.

### Berbagi AWS Glue Data Catalog dan tabel di seluruh akun

Anda dapat berbagi database dan tabel di seluruh akun dan masih menggunakan Lake Formation. Untuk informasi selengkapnya, lihat Berbagi data lintas akun di Lake Formation dan Bagaimana cara berbagiAWS Glue Data Catalog dan tabel cross-account menggunakan AWS Lake Formation?.

## Enkripsi antar pekerja

Dengan Amazon EMR versi 6.15.0 dan yang lebih tinggi, Anda dapat mengaktifkan komunikasi TLS terenkripsi timbal balik antara pekerja dalam pekerjaan Spark Anda. Saat diaktifkan, EMR Tanpa Server secara otomatis menghasilkan dan mendistribusikan sertifikat unik untuk setiap pekerja yang disediakan di bawah pekerjaan Anda. Ketika para pekerja ini berkomunikasi untuk bertukar pesan kontrol atau mentransfer data shuffle, mereka membuat TLS koneksi timbal balik dan menggunakan sertifikat yang dikonfigurasi untuk memverifikasi identitas satu sama lain. Jika pekerja tidak dapat memverifikasi sertifikat lain, TLS jabat tangan gagal, dan EMR Tanpa Server membatalkan koneksi di antara mereka.

Jika Anda menggunakan Lake Formation dengan EMR Tanpa Server, TLS enkripsi timbal balik diaktifkan secara default.

### Mengaktifkan TLS enkripsi timbal balik di Tanpa Server EMR

Untuk mengaktifkan TLS enkripsi timbal balik pada aplikasi spark Anda, atur `spark.ssl.internode.enabled` ke true saat membuat aplikasi Tanpa EMR Server. Jika Anda menggunakan AWS konsol untuk membuat aplikasi EMR Tanpa Server, pilih Gunakan pengaturan kustom, lalu perluas konfigurasi Aplikasi, dan masukkan. `runtimeConfiguration`

```
aws emr-serverless create-application \
--release-label emr-6.15.0 \
--runtime-configuration '{
  "classification": "spark-defaults",
  "properties": {"spark.ssl.internode.enabled": "true"}
}' \
--type "SPARK"
```

Jika Anda ingin mengaktifkan TLS enkripsi timbal balik untuk menjalankan tugas percikan individual, setel `spark.ssl.internode.enabled` ke true saat menggunakan`spark-submit`.

```
--conf spark.ssl.internode.enabled=true
```

# Secrets Manager untuk perlindungan data dengan EMR Serverless

AWS Secrets Manager adalah layanan penyimpanan rahasia yang dapat Anda gunakan untuk melindungi kredensi database, API kunci, dan informasi rahasia lainnya. Kemudian dalam kode Anda, Anda dapat mengganti kredensi hardcode dengan panggilan ke API Secrets Manager. Ini membantu memastikan bahwa rahasia tidak dapat dikompromikan oleh seseorang yang memeriksa kode Anda, karena rahasianya tidak ada. Untuk ikhtisar, lihat AWS Secrets Manager Panduan Pengguna.

Secrets Manager mengenkripsi rahasia menggunakan AWS Key Management Service kunci. Untuk informasi selengkapnya, lihat Enkripsi rahasia dan dekripsi di AWS Secrets Manager Panduan Pengguna.

Anda dapat mengonfigurasi Secrets Manager untuk secara otomatis memutar rahasia untuk Anda sesuai dengan jadwal yang Anda tentukan. Ini memungkinkan Anda mengganti rahasia jangka panjang dengan rahasia jangka pendek, yang membantu mengurangi risiko kompromi secara signifikan. Untuk informasi selengkapnya, lihat Memutar AWS Secrets Manager rahasia di AWS Secrets Manager Panduan Pengguna.

Amazon EMR Serverless terintegrasi dengan AWS Secrets Manager sehingga Anda dapat menyimpan data Anda di Secrets Manager dan menggunakan ID rahasia dalam konfigurasi Anda.

## Bagaimana EMR Serverless menggunakan rahasia

Saat menyimpan data di Secrets Manager dan menggunakan ID rahasia dalam konfigurasi untuk EMR Tanpa Server, Anda tidak meneruskan data konfigurasi sensitif ke EMR Tanpa Server dalam teks biasa dan memaparkannya ke eksternal. APIs Jika Anda menunjukkan bahwa pasangan kunci-nilai berisi ID rahasia untuk rahasia yang Anda simpan di Secrets Manager, EMR Serverless mengambil rahasia ketika mengirimkan data konfigurasi ke pekerja untuk menjalankan pekerjaan.

Untuk menunjukkan bahwa pasangan kunci-nilai untuk konfigurasi berisi referensi ke rahasia yang disimpan di Secrets Manager, tambahkan `EMR.secret@` anotasi ke nilai konfigurasi. Untuk properti konfigurasi apa pun dengan anotasi Id rahasia, EMR Tanpa Server memanggil Secrets Manager dan menyelesaikan rahasia pada saat eksekusi pekerjaan.

## Cara membuat rahasia

Untuk membuat rahasia, ikuti langkah-langkah di Buat AWS Secrets Manager rahasia di AWS Secrets Manager Panduan Pengguna. Pada Langkah 3, pilih bidang Plaintext untuk memasukkan nilai sensitif Anda.

# Berikan rahasia dalam klasifikasi konfigurasi

Contoh berikut menunjukkan bagaimana memberikan rahasia dalam klasifikasi konfigurasi di`StartJobRun`. Jika Anda ingin mengonfigurasi klasifikasi untuk Secrets Manager di tingkat aplikasi, lihatKonfigurasi aplikasi default untuk Tanpa EMR Server.

Dalam contoh, ganti *`SecretName`* dengan nama rahasia untuk diambil. Sertakan tanda hubung, diikuti oleh enam karakter yang ditambahkan Secrets Manager ke akhir rahasia. ARN Untuk informasi selengkapnya, lihat Cara membuat rahasia.

Dalam bagian ini

- Tentukan referensi rahasia - Spark
- Tentukan referensi rahasia - Sarang

## Tentukan referensi rahasia - Spark

Example — Tentukan referensi rahasia dalam konfigurasi metastore Hive eksternal untuk Spark

```
aws emr-serverless start-job-run \
  --application-id "application-id" \
  --execution-role-arn "job-role-arn" \
  --job-driver '{
        "sparkSubmit": {
            "entryPoint": "s3://DOC-EXAMPLE-BUCKET/scripts/spark-jdbc.py",
            "sparkSubmitParameters": "--jars s3://DOC-EXAMPLE-BUCKET/mariadb-connector-
java.jar
            --conf
 spark.hadoop.javax.jdo.option.ConnectionDriverName=org.mariadb.jdbc.Driver
            --conf spark.hadoop.javax.jdo.option.ConnectionUserName=connection-user-
name
            --conf
 spark.hadoop.javax.jdo.option.ConnectionPassword=EMR.secret@SecretName
            --conf spark.hadoop.javax.jdo.option.ConnectionURL=jdbc:mysql://db-host:db-
port/db-name
            --conf spark.driver.cores=2
            --conf spark.executor.memory=10G
            --conf spark.driver.memory=6G
            --conf spark.executor.cores=4"
        }
    }' \
    --configuration-overrides '{
```

```
        "monitoringConfiguration": {
        "s3MonitoringConfiguration": {
            "logUri": "s3://DOC-EXAMPLE-BUCKET/spark/logs/"
        }
    }
}'
```

Example — Tentukan referensi rahasia untuk konfigurasi metastore Hive eksternal dalam klasifikasi **spark-defaults**

```
{
        "classification": "spark-defaults",
        "properties": {

 "spark.hadoop.javax.jdo.option.ConnectionDriverName":"org.mariadb.jdbc.Driver"
            "spark.hadoop.javax.jdo.option.ConnectionURL":"jdbc:mysql://db-host:db-
port/db-name"
            "spark.hadoop.javax.jdo.option.ConnectionUserName":"connection-user-name"
            "spark.hadoop.javax.jdo.option.ConnectionPassword":
 "EMR.secret@SecretName",
        }
    }
```

## Tentukan referensi rahasia - Sarang

Example — Tentukan referensi rahasia dalam konfigurasi metastore Hive eksternal untuk Hive

```
aws emr-serverless start-job-run \
  --application-id "application-id" \
  --execution-role-arn "job-role-arn" \
    --job-driver '{
        "hive": {
        "query": "s3://DOC-EXAMPLE-BUCKET/emr-serverless-hive/query/hive-query.ql",
        "parameters": "--hiveconf hive.exec.scratchdir=s3://DOC-EXAMPLE-BUCKET/emr-
serverless-hive/hive/scratch
                    --hiveconf hive.metastore.warehouse.dir=s3://DOC-EXAMPLE-BUCKET/
emr-serverless-hive/hive/warehouse
                    --hiveconf javax.jdo.option.ConnectionUserName=username
                    --hiveconf
 javax.jdo.option.ConnectionPassword=EMR.secret@SecretName
                    --hiveconf
 hive.metastore.client.factory.class=org.apache.hadoop.hive.ql.metadata.SessionHiveMetaStoreCli
```

```
                    --hiveconf
 javax.jdo.option.ConnectionDriverName=org.mariadb.jdbc.Driver
                    --hiveconf javax.jdo.option.ConnectionURL=jdbc:mysql://db-host:db-
port/db-name"
        }
    }' \
    --configuration-overrides '{
        "monitoringConfiguration": {
        "s3MonitoringConfiguration": {
            "logUri": "s3://EXAMPLE-LOG-BUCKET"
        }
    }
}'
```

Example — Tentukan referensi rahasia untuk konfigurasi metastore Hive eksternal dalam klasifikasi **`hive-site`**

```
{
    "classification": "hive-site",
    "properties": {
        "hive.metastore.client.factory.class":
 "org.apache.hadoop.hive.ql.metadata.SessionHiveMetaStoreClientFactory",
        "javax.jdo.option.ConnectionDriverName": "org.mariadb.jdbc.Driver",
        "javax.jdo.option.ConnectionURL": "jdbc:mysql://db-host:db-port/db-name",
        "javax.jdo.option.ConnectionUserName": "username",
        "javax.jdo.option.ConnectionPassword": "EMR.secret@SecretName"
    }
}
```

## Berikan akses untuk EMR Tanpa Server untuk mengambil rahasia

Untuk mengizinkan EMR Tanpa Server mengambil nilai rahasia dari Secrets Manager, tambahkan pernyataan kebijakan berikut ke rahasia Anda saat Anda membuatnya. Anda harus membuat rahasia Anda dengan KMS kunci yang dikelola pelanggan agar EMR Tanpa Server membaca nilai rahasia. Untuk informasi selengkapnya, lihat Izin untuk KMS kunci di AWS Secrets Manager Panduan Pengguna.

Dalam kebijakan berikut, ganti *`applicationId`* dengan ID untuk aplikasi Anda.

Kebijakan sumber daya untuk rahasia

Anda harus menyertakan izin berikut dalam kebijakan sumber daya untuk rahasia di AWS Secrets Manager untuk memungkinkan EMR Tanpa Server mengambil nilai rahasia. Untuk memastikan bahwa hanya aplikasi tertentu yang dapat mengambil rahasia ini, Anda dapat secara opsional menentukan ID aplikasi EMR Tanpa Server sebagai kondisi dalam kebijakan.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue",
        "secretsmanager:DescribeSecret"
      ],
      "Principal": {
        "Service": [
          "emr-serverless.amazonaws.com"
        ]
      },
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:SourceArn": "arn:aws:emr-serverless:Wilayah AWS:aws_account_id:/
applications/applicationId"
        }
      }
    }
  ]
}
```

Buat rahasia Anda dengan kebijakan berikut untuk yang dikelola pelanggan AWS Key Management Service (AWS KMS) kunci:

Kebijakan untuk dikelola pelanggan AWS KMS kunci

```
{
    "Sid": "Allow EMR Serverless to use the key for decrypting secrets",
    "Effect": "Allow",
    "Principal": {
        "Service": [
```

```
            "emr-serverless.amazonaws.com"
        ]
    },
    "Action": [
        "kms:Decrypt",
        "kms:DescribeKey"
    ],
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "kms:ViaService": "secretsmanager.Wilayah AWS.amazonaws.com"
        }
    }
}
```

## Memutar rahasianya

Rotasi adalah saat Anda memperbarui rahasia secara berkala. Anda dapat mengkonfigurasi 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. EMRTanpa server mengambil nilai rahasia dari konfigurasi beranotasi saat pekerjaan bertransisi ke status berjalan. Jika Anda atau proses memperbarui nilai rahasia di Secrets Manager, Anda harus mengirimkan pekerjaan baru sehingga pekerjaan dapat mengambil nilai yang diperbarui.

> **Note**
>
> Pekerjaan yang sudah dalam status berjalan tidak dapat mengambil nilai rahasia yang diperbarui. Hal ini dapat mengakibatkan kegagalan pekerjaan.

# Menggunakan Hibah Akses Amazon S3 dengan Tanpa Server EMR

## Ikhtisar Hibah Akses S3 untuk Tanpa Server EMR

Dengan Amazon EMR merilis 6.15.0 dan yang lebih tinggi, Amazon S3 Access Grants menyediakan solusi kontrol akses yang dapat diskalakan yang dapat Anda gunakan untuk menambah akses ke data Amazon S3 Anda dari Tanpa Server. EMR Jika Anda memiliki konfigurasi izin yang kompleks

atau besar untuk data S3, Anda dapat menggunakan Access Grants untuk menskalakan izin data S3 untuk pengguna, peran, dan aplikasi.

Gunakan S3 Access Grants untuk menambah akses ke data Amazon S3 di luar izin yang diberikan oleh peran runtime atau IAM peran yang dilampirkan ke identitas dengan akses ke aplikasi Tanpa Server Anda. EMR

Untuk informasi selengkapnya, lihat Mengelola akses dengan Hibah Akses S3 untuk Amazon EMR di Panduan EMR Manajemen Amazon dan Mengelola akses dengan Hibah Akses S3 di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Bagian ini menjelaskan cara meluncurkan aplikasi EMR Tanpa Server yang menggunakan S3 Access Grants untuk menyediakan akses ke data di Amazon S3. Untuk langkah-langkah menggunakan S3 Access Grants dengan EMR penerapan Amazon lainnya, lihat dokumentasi berikut:

- Menggunakan Hibah Akses S3 dengan Amazon EMR
- Menggunakan Hibah Akses S3 dengan Amazon di EMR EKS

## Luncurkan aplikasi EMR Tanpa Server dengan S3 Access Grants untuk manajemen data

Anda dapat mengaktifkan S3 Access Grants di EMR Serverless dan meluncurkan aplikasi Spark. Saat aplikasi Anda membuat permintaan untuk data S3, Amazon S3 menyediakan kredenal sementara yang dicakup ke bucket, awalan, atau objek tertentu.

1. Siapkan peran eksekusi pekerjaan untuk aplikasi EMR Tanpa Server Anda. Sertakan IAM izin yang diperlukan yang Anda perlukan untuk menjalankan pekerjaan Spark dan menggunakan S3 Access Grants, dan: `s3:GetDataAccess` `s3:GetAccessGrantsInstanceForPrefix`

```
{
    "Effect": "Allow",
    "Action": [
    "s3:GetDataAccess",
    "s3:GetAccessGrantsInstanceForPrefix"
    ],
    "Resource": [     //LIST ALL INSTANCE ARNS THAT THE ROLE IS ALLOWED TO QUERY
         "arn:aws_partition:s3:Region:account-id1:access-grants/default",
         "arn:aws_partition:s3:Region:account-id2:access-grants/default"
    ]
```

```
}
```

> **Note**
>
> Jika Anda menentukan IAM peran untuk eksekusi pekerjaan yang memiliki izin tambahan untuk mengakses S3 secara langsung, maka pengguna akan dapat mengakses data yang diizinkan oleh peran tersebut meskipun mereka tidak memiliki izin dari S3 Access Grants.

2. Luncurkan aplikasi EMR Tanpa Server Anda dengan label EMR rilis Amazon 6.15.0 atau lebih tinggi dan `spark-defaults` klasifikasi, seperti yang ditunjukkan contoh berikut. Ganti nilai *red text* dengan nilai yang sesuai untuk skenario penggunaan Anda.

```
aws emr-serverless start-job-run \
  --application-id application-id \
  --execution-role-arn job-role-arn \
  --job-driver '{
        "sparkSubmit": {
            "entryPoint": "s3://us-east-1.elasticmapreduce/emr-containers/samples/
wordcount/scripts/wordcount.py",
            "entryPointArguments": ["s3://DOC-EXAMPLE-BUCKET-OUTPUT/
wordcount_output"],
            "sparkSubmitParameters": "--conf spark.executor.cores=1 --conf
 spark.executor.memory=4g --conf spark.driver.cores=1 --conf spark.driver.memory=4g
 --conf spark.executor.instances=1"
        }
    }' \
  --configuration-overrides '{
    "applicationConfiguration": [{
        "classification": "spark-defaults",
        "properties": {
          "spark.hadoop.fs.s3.s3AccessGrants.enabled": "true",
          "spark.hadoop.fs.s3.s3AccessGrants.fallbackToIAM": "false"
         }
      }]
}'
```

## Akses S3 Memberikan pertimbangan dengan Tanpa Server EMR

Untuk informasi penting dukungan, kompatibilitas, dan perilaku saat Anda menggunakan Hibah Akses Amazon S3 dengan EMR Tanpa Server, lihat pertimbangan Hibah Akses S3 dengan Amazon di Panduan Manajemen Amazon. EMR EMR

# Mencatat panggilan Amazon EMR Tanpa Server menggunakan API AWS CloudTrail

Amazon EMR Serverless terintegrasi dengan AWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau AWS layanan di EMR Serverless. CloudTrail menangkap semua API panggilan untuk EMR Tanpa Server sebagai acara. Panggilan yang diambil termasuk panggilan dari konsol EMR Tanpa Server dan panggilan kode ke operasi Tanpa EMR ServerAPI. Jika Anda membuat jejak, Anda dapat mengaktifkan pengiriman CloudTrail acara secara terus menerus ke bucket Amazon S3, termasuk peristiwa untuk EMR Tanpa Server. 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 ke EMR Tanpa Server, alamat IP dari mana permintaan dibuat, siapa yang membuat permintaan, kapan dibuat, dan detail tambahan.

Untuk mempelajari lebih lanjut tentang CloudTrail, lihat AWS CloudTrail Panduan Pengguna.

## EMRInformasi tanpa server di CloudTrail

CloudTrail diaktifkan pada Akun AWS saat Anda membuat akun. Ketika aktivitas terjadi di EMR Tanpa Server, aktivitas tersebut direkam dalam suatu CloudTrail peristiwa bersama dengan yang lain AWS acara layanan dalam sejarah Acara. Anda dapat melihat, mencari, dan mengunduh acara terbaru di Akun AWS. Untuk informasi selengkapnya, lihat Melihat peristiwa dengan Riwayat CloudTrail acara.

Untuk catatan peristiwa yang sedang berlangsung di Akun AWS, termasuk acara untuk EMR Tanpa Server, buat jejak. Jejak memungkinkan CloudTrail untuk mengirimkan file log ke bucket Amazon S3. Secara default, saat Anda membuat jejak di konsol, jejak berlaku untuk semua Wilayah AWS. Jejak mencatat peristiwa dari semua Wilayah di AWS partisi dan mengirimkan file log ke bucket Amazon S3 yang Anda tentukan. Selain itu, Anda dapat mengkonfigurasi AWS layanan untuk menganalisis lebih lanjut dan bertindak atas data peristiwa yang dikumpulkan dalam CloudTrail log. Untuk informasi selengkapnya, lihat berikut:

- Gambaran umum untuk membuat jejak
- CloudTrail layanan dan integrasi yang didukung
- Mengonfigurasi SNS notifikasi Amazon untuk CloudTrail
- Menerima file CloudTrail log dari beberapa wilayah dan Menerima file CloudTrail log dari beberapa akun

Semua tindakan EMR Tanpa Server dicatat oleh CloudTrail dan didokumentasikan dalam Referensi Tanpa EMRServer API. Misalnya, panggilan ke`CreateApplication`, `StartJobRun` dan `CancelJobRun` tindakan menghasilkan entri dalam file CloudTrail log.

Setiap entri peristiwa atau log berisi informasi tentang siapa yang membuat permintaan tersebut. Informasi identitas membantu Anda menentukan berikut ini:

- Apakah permintaan dibuat dengan root atau AWS Identity and Access Management (IAM) kredensi pengguna.
- Apakah permintaan tersebut dibuat dengan kredensial keamanan sementara untuk satu peran atau pengguna terfederasi.
- Apakah permintaan itu dibuat oleh orang lain AWS layanan.

Untuk informasi lebih lanjut, lihat CloudTrail userIdentity elemen.

## Memahami EMR entri file log tanpa server

Trail adalah konfigurasi yang memungkinkan pengiriman peristiwa sebagai file log ke bucket Amazon S3 yang Anda tentukan. CloudTrail file log berisi satu atau lebih entri log. Peristiwa mewakili permintaan tunggal dari sumber manapun dan mencakup informasi tentang tindakan yang diminta, tanggal dan waktu tindakan, parameter permintaan, dan sebagainya. CloudTrail file log bukanlah jejak tumpukan yang diurutkan dari API panggilan publik, sehingga tidak muncul dalam urutan tertentu.

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan `CreateApplication` tindakan.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE:admin",
        "arn": "arn:aws:sts::012345678910:assumed-role/Admin/admin",
```

```
        "accountId": "012345678910",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::012345678910:role/Admin",
                "accountId": "012345678910",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-06-01T23:46:52Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2022-06-01T23:49:28Z",
    "eventSource": "emr-serverless.amazonaws.com",
    "eventName": "CreateApplication",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "203.0.113.0",
    "userAgent": "PostmanRuntime/7.26.10",
    "requestParameters": {
        "name": "my-serverless-application",
        "releaseLabel": "emr-6.6",
        "type": "SPARK",
        "clientToken": "0a1b234c-de56-7890-1234-567890123456"
    },
    "responseElements": {
        "name": "my-serverless-application",
        "applicationId": "1234567890abcdef0",
        "arn": "arn:aws:emr-serverless:us-west-2:555555555555:/
applications/1234567890abcdef0"
    },
    "requestID": "890b8639-e51f-11e7-b038-EXAMPLE",
    "eventID": "874f89fa-70fc-4798-bc00-EXAMPLE",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "012345678910",
    "eventCategory": "Management"
}
```

# Validasi kepatuhan untuk Amazon Tanpa Server EMR

Keamanan dan kepatuhan EMR Tanpa Server dinilai oleh auditor pihak ketiga sebagai bagian dari beberapa AWS program kepatuhan, termasuk yang berikut:

- Sistem dan Kontrol Organisasi (SOC)
- Standar Keamanan Data Industri Kartu Pembayaran (PCIDSS)
- Program Manajemen Risiko dan Otorisasi Federal (FedRAMP) Moderat
- Undang-Undang Portabilitas dan Akuntabilitas Asuransi Kesehatan () HIPAA

AWS menyediakan daftar yang sering diperbarui AWS layanan dalam lingkup program kepatuhan khusus di AWS Layanan dalam Lingkup oleh Program Kepatuhan.

Laporan audit pihak ketiga tersedia untuk Anda unduh menggunakan AWS Artifact. Untuk informasi selengkapnya, lihat Mengunduh Laporan di AWS Artifak.

Untuk informasi lebih lanjut tentang AWS program kepatuhan, lihat AWS Program Kepatuhan.

Tanggung jawab kepatuhan Anda saat menggunakan EMR Tanpa Server ditentukan oleh sensitivitas data Anda, tujuan kepatuhan organisasi Anda, serta undang-undang dan peraturan yang berlaku. Jika penggunaan EMR Tanpa Server Anda tunduk pada kepatuhan terhadap standar sepertiHIPAA,, atau Fed PCI RAMP Moderate, AWS menyediakan sumber daya untuk membantu:

- Panduan Memulai Cepat Keamanan dan Kepatuhan yang membahas pertimbangan arsitektur dan langkah-langkah untuk menerapkan lingkungan dasar yang berfokus pada keamanan dan kepatuhan pada AWS.
- AWS Panduan Kepatuhan Pelanggan dapat membantu Anda memahami model tanggung jawab bersama melalui lensa kepatuhan. Panduan merangkum praktik terbaik untuk mengamankan Layanan AWS dan memetakan panduan untuk kontrol keamanan di berbagai kerangka kerja (termasuk Institut Standar dan Teknologi Nasional (NIST), Dewan Standar Keamanan Industri Kartu Pembayaran (PCI), dan Organisasi Internasional untuk Standardisasi (ISO)).
- AWS Config dapat digunakan untuk menilai seberapa baik konfigurasi sumber daya Anda telah mematuhi praktik internal, pedoman industri, dan peraturan yang berlaku.
- AWS Sumber Daya Kepatuhan adalah kumpulan buku kerja dan panduan yang mungkin berlaku untuk industri dan lokasi Anda.

- AWS Security Hub memberi Anda pandangan komprehensif tentang status keamanan Anda di dalamnya AWS dan membantu Anda memeriksa kepatuhan Anda terhadap standar industri keamanan dan praktik terbaik.
- AWS Audit Manager— ini Layanan AWS membantu Anda terus mengaudit AWS penggunaan untuk menyederhanakan cara Anda mengelola risiko dan kepatuhan terhadap peraturan dan standar industri.

## Ketahanan di Amazon Tanpa Server EMR

Bagian AWS Infrastruktur global dibangun di sekitar AWS Wilayah dan Zona Ketersediaan. AWS Wilayah menyediakan beberapa Availability Zone yang terpisah secara fisik dan terisolasi, yang terhubung dengan jaringan berlatensi rendah, throughput yang tinggi, dan sangat redundan. Dengan Availability Zone, Anda dapat mendesain dan mengoperasikan aplikasi dan basis data yang secara otomatis mengalami kegagalan di antara zona tanpa gangguan. Zona Ketersediaan memiliki ketersediaan dan toleransi kesalahan yang lebih baik, dan dapat diskalakan dibandingkan infrastruktur pusat data tunggal atau multi tradisional.

Untuk informasi lebih lanjut tentang AWS Wilayah dan Availability Zone, lihat AWS Infrastruktur Global.

Selain AWS Infrastruktur global, Amazon EMR Serverless menawarkan integrasi dengan Amazon S3 EMRFS untuk membantu mendukung ketahanan data dan kebutuhan pencadangan Anda.

## Keamanan infrastruktur di Amazon Tanpa EMR Server

Sebagai layanan terkelola, Amazon EMR dilindungi oleh AWS keamanan jaringan global. Untuk informasi tentang AWS Layanan keamanan dan bagaimana AWS melindungi infrastruktur, lihat AWS Keamanan Cloud. Untuk mendesain Anda AWS lingkungan menggunakan praktik terbaik untuk keamanan infrastruktur, lihat Perlindungan Infrastruktur di Pilar  Keamanan AWS Kerangka Kerja yang Diarsiteksikan dengan Baik.

Anda menggunakan AWS APIpanggilan yang diterbitkan untuk mengakses Amazon EMR melalui jaringan. Klien harus mendukung hal-hal berikut:

- Keamanan Lapisan Transportasi (TLS). Kami membutuhkan TLS 1.2 dan merekomendasikan TLS 1.3.

- Suite cipher dengan kerahasiaan maju yang sempurna (PFS) seperti (Ephemeral Diffie-Hellman) atau DHE (Elliptic Curve Ephemeral Diffie-Hellman). ECDHE Sebagian besar sistem modern seperti Java 7 dan versi lebih baru mendukung mode-mode ini.

Selain itu, permintaan harus ditandatangani dengan menggunakan ID kunci akses dan kunci akses rahasia yang terkait dengan IAM prinsipal. Atau Anda dapat menggunakan AWS Security Token Service (AWS STS) untuk menghasilkan kredensi keamanan sementara untuk menandatangani permintaan.

## Analisis konfigurasi dan kerentanan di Amazon Tanpa Server EMR

AWS menangani tugas-tugas keamanan dasar seperti sistem operasi tamu (OS) dan patch database, konfigurasi firewall, dan pemulihan bencana. Prosedur ini telah ditinjau dan disertifikasi oleh pihak ketiga yang sesuai. Untuk detail selengkapnya, lihat sumber daya berikut:

- Validasi kepatuhan untuk Amazon Tanpa Server EMR
- Model Tanggung Jawab Bersama
- Amazon Web Services: Gambaran Umum Proses Keamanan (whitepaper)

# Titik akhir dan kuota untuk EMR Serverless

## Titik akhir layanan

Untuk terhubung secara terprogram ke Layanan AWS, Anda menggunakan titik akhir. Endpoint URL adalah titik masuk untuk AWS layanan web. Selain standar AWS titik akhir, beberapa Layanan AWS menawarkan FIPS titik akhir di Wilayah tertentu. Tabel berikut mencantumkan endpoint layanan untuk EMR Serverless. Untuk informasi selengkapnya, silakan lihat Layanan AWS titik akhir.

| Nama Wilayah | Wilayah | Titik Akhir | Protokol |
|---|---|---|---|
| AS Timur (Ohio) | `us-east-2` (terbatas pada Availability Zone berikut:`use2-az1`,`use2-az2`, dan`use2-az3`) | `emr-serverless.us-east-2.amazonaws.com` | HTTPS |
| AS Timur (Virginia Utara) | `us-east-1` (terbatas pada Availability Zone berikut: `use1-az1`,`use1-az2`,,`use1-az4`,`use1-az5`, dan`use1-az6`) | `emr-serverless.us-east-1.amazonaws.com`<br><br>`emr-serverless-fips.us-east-1.amazonaws.com` | HTTPS |
| AS Barat (California Utara) | `us-west-1` | `emr-serverless.us-west-1.amazonaws.com` | HTTPS |
| AS Barat (Oregon) | `us-west-2` | `emr-serverless.us-west-2.amazonaws.com` | HTTPS |

| Nama Wilayah | Wilayah | Titik Akhir | Protokol |
|---|---|---|---|
| | | emr-serverless-fips.us-west-2.amazonaws.com | |
| Afrika (Cape Town) | af-south-1 | emr-serverless.af-south-1.amazonaws.com | HTTPS |
| Asia Pasifik (Hong Kong) | ap-east-1 | emr-serverless.ap-east-1.amazonaws.com | HTTPS |
| Asia Pasifik (Jakarta) | ap-southeast-3 | emr-serverless.ap-southeast-3.amazonaws.com | HTTPS |
| Asia Pasifik (Mumbai) | ap-south-1 | emr-serverless.ap-south-1.amazonaws.com | HTTPS |
| Asia Pasifik (Osaka) | ap-northeast-3 | emr-serverless.ap-northeast-3.amazonaws.com | HTTPS |

| Nama Wilayah | Wilayah | Titik Akhir | Protokol |
| --- | --- | --- | --- |
| Asia Pasifik (Seoul) | ap-northeast-2 | emr-serverless.ap-northeast-2.amazonaws.com | HTTPS |
| Asia Pasifik (Singapura) | ap-southeast-1 | emr-serverless.ap-southeast-1.amazonaws.com | HTTPS |
| Asia Pasifik (Sydney) | ap-southeast-2 | emr-serverless.ap-southeast-2.amazonaws.com | HTTPS |
| Asia Pasifik (Tokyo) | ap-northeast-1 | emr-serverless.ap-northeast-1.amazonaws.com | HTTPS |
| (Canada (Central) | ca-central-1 (terbatas pada Availability Zone berikut: cac1-az1 dancac1-az2) | emr-serverless.ca-central-1.amazonaws.com | HTTPS |
| Eropa (Frankfurt) | eu-central-1 | emr-serverless.eu-central-1.amazonaws.com | HTTPS |

| Nama Wilayah | Wilayah | Titik Akhir | Protokol |
|---|---|---|---|
| Eropa (Irlandia) | eu-west-1 | emr-serverless.eu-west-1.amazonaws.com | HTTPS |
| Eropa (London) | eu-west-2 | emr-serverless.eu-west-2.amazonaws.com | HTTPS |
| Eropa (Milan) | eu-south-1 | emr-serverless.eu-south-1.amazonaws.com | HTTPS |
| Eropa (Paris) | eu-west-3 | emr-serverless.eu-west-3.amazonaws.com | HTTPS |
| Eropa (Spanyol) | eu-south-2 | emr-serverless.eu-south-2.amazonaws.com | HTTPS |
| Eropa (Stockholm) | eu-north-1 | emr-serverless.eu-north-1.amazonaws.com | HTTPS |
| Timur Tengah (Bahrain) | me-south-1 | emr-serverless.me-south-1.amazonaws.com | HTTPS |

| Nama Wilayah | Wilayah | Titik Akhir | Protokol |
| --- | --- | --- | --- |
| Timur Tengah (UAE) | `me-central-1` | `emr-serverless.me-central-1.amazonaws.com` | HTTPS |
| Amerika Selatan (Sao Paulo) | `sa-east-1` | `emr-serverless.sa-east-1.amazonaws.com` | HTTPS |
| AWS GovCloud (AS-Timur) | `us-gov-east-1` | `emr-serverless.us-gov-east-1.amazonaws.com` | HTTPS |
| AWS GovCloud (AS-Barat) | `us-gov-west-1` | `emr-serverless.us-gov-west-1.amazonaws.com` | HTTPS |

# Kuota layanan

Kuota layanan, juga dikenal sebagai batas, adalah jumlah maksimum sumber daya layanan atau operasi yang Anda Akun AWS dapat digunakan. EMRTanpa server mengumpulkan metrik penggunaan kuota layanan setiap menit dan mempublikasikannya di namespace. `AWS/Usage`

> **Note**
>
> Baru AWS akun mungkin memiliki kuota awal yang lebih rendah yang dapat meningkat seiring waktu. Amazon EMR Serverless memantau penggunaan akun dalam masing-masing Wilayah AWS, dan kemudian secara otomatis meningkatkan kuota berdasarkan penggunaan Anda.

Tabel berikut mencantumkan kuota layanan untuk Tanpa EMR Server. Untuk informasi selengkapnya, silakan lihat Layanan AWS kuota.

| Nama | Batas default | Dapat disesuaikan? | Deskripsi |
| --- | --- | --- | --- |
| Maks bersamaan vCPUs per akun | 16 | Ya | Jumlah maksimum vCPUs yang dapat berjalan secara bersamaan untuk akun saat ini Wilayah AWS. |

## APIbatas

Berikut ini menjelaskan API batas per Wilayah untuk Anda Akun AWS.

| Sumber Daya | Kuota bawaan |
| --- | --- |
| ListApplications | 10 transaksi per detik. Burst 50 transaksi per detik. |
| CreateApplication | 1 transaksi per detik. Burst 25 transaksi per detik. |
| DeleteApplication | 1 transaksi per detik. Burst 25 transaksi per detik. |
| GetApplication | 10 transaksi per detik. Burst 50 transaksi per detik. |
| UpdateApplication | 1 transaksi per detik. Burst 25 transaksi per detik. |
| ListJobRuns | 1 transaksi per detik. Burst 25 transaksi per detik. |
| StartJobRun | 1 transaksi per detik. Burst 25 transaksi per detik. |

| Sumber Daya | Kuota bawaan |
| --- | --- |
| GetDashboardForJobRun | 1 transaksi per detik. Burst 2 transaksi per detik. |
| CancelJobRun | 1 transaksi per detik. Burst 25 transaksi per detik. |
| GetJobRun | 10 transaksi per detik. Burst 50 transaksi per detik. |
| StartApplication | 1 transaksi per detik. Burst 25 transaksi per detik. |
| StopApplication | 1 transaksi per detik. Burst 25 transaksi per detik. |

# Pertimbangan lainnya

Daftar berikut berisi pertimbangan lain dengan Tanpa EMR Server.

- EMRServerless tersedia di berikut ini Wilayah AWS:
  - AS Timur (Ohio)
  - AS Timur (Virginia Utara)
  - AS Barat (California Utara)
  - AS Barat (Oregon)
  - Afrika (Cape Town)
  - Asia Pasifik (Hong Kong)
  - Asia Pasifik (Jakarta)
  - Asia Pasifik (Mumbai)
  - Asia Pasifik (Osaka)
  - Asia Pasifik (Seoul)
  - Asia Pasifik (Singapura)
  - Asia Pasifik (Sydney)
  - Asia Pasifik (Tokyo)
  - Kanada (Pusat)
  - Eropa (Frankfurt)
  - Eropa (Irlandia)
  - Eropa (London)
  - Eropa (Milan)
  - Eropa (Paris)
  - Eropa (Spanyol)
  - Eropa (Stockholm)
  - Timur Tengah (Bahrain)
  - Timur Tengah (UAE)
  - Amerika Selatan (Sao Paulo)
  - AWS GovCloud (AS-Timur)
  - AWS GovCloud (AS-Barat)

Untuk daftar titik akhir yang terkait dengan Wilayah ini, lihatTitik akhir layanan.

- Batas waktu default untuk menjalankan pekerjaan adalah 12 jam. Anda dapat mengubah pengaturan ini dengan `executionTimeoutMinutes` properti di `startJobRun` API atau AWS SDK. Anda dapat mengatur `executionTimeoutMinutes` ke 0 jika Anda ingin pekerjaan Anda tidak pernah habis. Misalnya, jika Anda memiliki aplikasi streaming, Anda dapat mengatur `executionTimeoutMinutes` ke 0 untuk memungkinkan pekerjaan streaming berjalan terus menerus.
- `billedResourceUtilization`Properti dalam `getJobRun` API menunjukkan agregat vCPU, memori, dan penyimpanan yang AWS telah ditagih untuk menjalankan pekerjaan. Sumber daya yang ditagih mencakup penggunaan minimum 1 menit untuk pekerja, ditambah penyimpanan tambahan lebih dari 20 GB per pekerja. Sumber daya ini tidak termasuk penggunaan untuk pekerja pra-inisialisasi yang menganggur.
- Tanpa VPC konektivitas, pekerjaan dapat mengakses beberapa Layanan AWS titik akhir dalam hal yang sama Wilayah AWS. Layanan ini termasuk Amazon S3, AWS Glue, CloudWatch Log Amazon, AWS KMS, AWS Security Token Service, Amazon DynamoDB, dan AWS Secrets Manager. Anda dapat mengaktifkan VPC konektivitas untuk mengakses lainnya Layanan AWS melalui AWS PrivateLink, tetapi Anda tidak diharuskan untuk melakukan ini. Untuk mengakses layanan eksternal, Anda dapat membuat aplikasi Anda dengan fileVPC.
- EMRTanpa server tidak mendukung. HDFS Disk lokal pada pekerja adalah penyimpanan temporal yang digunakan EMR Tanpa Server untuk mengacak dan memproses data selama pekerjaan berjalan.
- EMRTanpa server tidak mendukung yang ada. emr-dynamodb-connector

# EMRVersi rilis Amazon Tanpa Server

EMRRilis Amazon adalah seperangkat aplikasi open source dari ekosistem big data. Setiap rilis menyertakan aplikasi data besar, komponen, dan fitur yang Anda pilih agar Amazon EMR Serverless disebarkan dan dikonfigurasi saat menjalankan pekerjaan.

Dengan Amazon EMR 6.6.0 dan yang lebih tinggi, Anda dapat menerapkan EMR Tanpa Server. Opsi penerapan ini tidak tersedia dengan versi EMR rilis Amazon sebelumnya. Ketika Anda mengirimkan pekerjaan Anda, Anda harus menentukan salah satu rilis yang didukung berikut.

Topik



## EMR Serverless 7.2.0

Tabel berikut mencantumkan versi aplikasi yang tersedia EMR Serverless 7.2.0.

| Aplikasi | Versi |
| --- | --- |
| Apache Spark | 3.5.1 |
| Apache Hive | 3.1.3 |

| Aplikasi | Versi |
| --- | --- |
| Apache Tez | 0.10.2 |

EMRCatatan rilis Serverless 7.2.0

- Lake Formation with EMR Serverless — Anda sekarang dapat menggunakan AWS Lake Formation untuk menerapkan kontrol akses berbutir halus pada tabel Katalog Data yang didukung oleh S3. Kemampuan ini memungkinkan Anda mengonfigurasi kontrol akses tingkat tabel, baris, kolom, dan sel untuk kueri baca dalam pekerjaan Spark EMR Tanpa Server Anda. Untuk informasi selengkapnya, silakan lihat the section called “Lake Formation untuk FGAC” dan the section called “Pertimbangan”.

## EMR Serverless 7.1.0

Tabel berikut mencantumkan versi aplikasi yang tersedia EMR Serverless 7.1.0.

| Aplikasi | Versi |
| --- | --- |
| Apache Spark | 3.5.0 |
| Apache Hive | 3.1.3 |
| Apache Tez | 0.10.2 |

## EMR Serverless 7.0.0

Tabel berikut mencantumkan versi aplikasi yang tersedia EMR Serverless 7.0.0.

| Aplikasi | Versi |
| --- | --- |
| Apache Spark | 3.5.0 |
| Apache Hive | 3.1.3 |
| Apache Tez | 0.10.2 |

# EMR Serverless 6.15.0

Tabel berikut mencantumkan versi aplikasi yang tersedia EMR Serverless 6.15.0.

| Aplikasi | Versi |
| --- | --- |
| Apache Spark | 3.4.1 |
| Apache Hive | 3.1.3 |
| Apache Tez | 0.10.2 |

EMRCatatan rilis Serverless 6.15.0

- TLSdukungan - Dengan Amazon EMR Serverless rilis 6.15.0 dan yang lebih tinggi, Anda dapat mengaktifkan komunikasi TLS terenkripsi timbal balik antara pekerja dalam pekerjaan Spark Anda. Saat diaktifkan, EMR Serverless secara otomatis menghasilkan sertifikat unik untuk setiap pekerja yang disediakan di bawah pekerjaan yang digunakan pekerja selama TLS jabat tangan untuk mengautentikasi satu sama lain dan membuat saluran terenkripsi untuk memproses data dengan aman. Untuk informasi selengkapnya tentang TLS enkripsi timbal balik, lihat Enkripsi antar pekerja.

# EMR Serverless 6.14.0

Tabel berikut mencantumkan versi aplikasi yang tersedia EMR Serverless 6.14.0.

| Aplikasi | Versi |
| --- | --- |
| Apache Spark | 3.4.1 |
| Apache Hive | 3.1.3 |
| Apache Tez | 0.10.2 |

# EMR Serverless 6.13.0

Tabel berikut mencantumkan versi aplikasi yang tersedia EMR Serverless 6.13.0.

| Aplikasi | Versi |
| --- | --- |
| Apache Spark | 3.4.1 |
| Apache Hive | 3.1.3 |
| Apache Tez | 0.10.2 |

## EMR Serverless 6.12.0

Tabel berikut mencantumkan versi aplikasi yang tersedia EMR Serverless 6.12.0.

| Aplikasi | Versi |
| --- | --- |
| Apache Spark | 3.4.0 |
| Apache Hive | 3.1.3 |
| Apache Tez | 0.10.2 |

## EMR Serverless 6.11.0

Tabel berikut mencantumkan versi aplikasi yang tersedia EMR Serverless 6.11.0.

| Aplikasi | Versi |
| --- | --- |
| Apache Spark | 3.3.2 |
| Apache Hive | 3.1.3 |
| Apache Tez | 0.10.2 |

EMRCatatan rilis Serverless 6.11.0

- Mengakses sumber daya S3 di akun lain - Dengan rilis 6.11.0 dan yang lebih tinggi, Anda dapat mengonfigurasi beberapa IAM peran untuk diasumsikan saat mengakses bucket Amazon S3 di berbagai AWS akun dari EMR Serverless.

# EMR Serverless 6.10.0

Tabel berikut mencantumkan versi aplikasi yang tersedia EMR Serverless 6.10.0.

| Aplikasi | Versi |
| --- | --- |
| Apache Spark | 3.3.1 |
| Apache Hive | 3.1.3 |
| Apache Tez | 0.10.2 |

EMRCatatan rilis Serverless 6.10.0

- Untuk aplikasi EMR Tanpa Server dengan rilis 6.10.0 atau lebih tinggi, nilai default untuk properti adalah. `spark.dynamicAllocation.maxExecutors` `infinity` Sebelumnya rilis default ke`100`. Untuk informasi selengkapnya, lihat Properti pekerjaan percikan.

# EMR Serverless 6.9.0

Tabel berikut mencantumkan versi aplikasi yang tersedia EMR Serverless 6.9.0.

| Aplikasi | Versi |
| --- | --- |
| Apache Spark | 3.3.0 |
| Apache Hive | 3.1.3 |
| Apache Tez | 0.10.2 |

EMRCatatan rilis Serverless 6.9.0

- Integrasi Amazon Redshift untuk Apache Spark disertakan dalam EMR rilis Amazon 6.9.0 dan yang lebih baru. Sebelumnya alat open-source, integrasi asli adalah konektor Spark yang dapat Anda gunakan untuk membangun aplikasi Apache Spark yang membaca dan menulis ke data di Amazon Redshift dan Amazon Redshift Serverless. Untuk informasi selengkapnya, lihat Menggunakan integrasi Amazon Redshift untuk Apache Spark di Amazon Tanpa Server EMR.

- EMRRilis tanpa server 6.9.0 menambahkan dukungan untuk AWS Arsitektur Graviton2 (arm64). Anda dapat menggunakan `architecture` parameter untuk `create-application` dan `update-application` APIs untuk memilih arsitektur arm64. Untuk informasi selengkapnya, lihat Opsi EMR arsitektur Amazon Tanpa Server.
- Anda sekarang dapat mengekspor, mengimpor, menanyakan, dan bergabung dengan tabel Amazon DynamoDB langsung dari aplikasi Serverless Spark dan EMR Hive Anda. Untuk informasi selengkapnya, lihat Menghubungkan ke DynamoDB dengan Amazon Serverless EMR.

Masalah yang diketahui

- Jika Anda menggunakan integrasi Amazon Redshift untuk Apache Spark dan memiliki waktu, jadwal, stempel waktu, atau timestamptz dengan presisi mikrodetik dalam format Parket, konektor membulatkan nilai waktu ke nilai milidetik terdekat. Sebagai solusinya, gunakan parameter format pembongkaran teks. `unload_s3_format`

## EMR Serverless 6.8.0

Tabel berikut mencantumkan versi aplikasi yang tersedia EMR Serverless 6.8.0.

| Aplikasi | Versi |
| --- | --- |
| Apache Spark | 3.3.0 |
| Apache Hive | 3.1.3 |
| Apache Tez | 0.9.2 |

## EMR Serverless 6.7.0

Tabel berikut mencantumkan versi aplikasi yang tersedia EMR Serverless 6.7.0.

| Aplikasi | Versi |
| --- | --- |
| Apache Spark | 3.2.1 |
| Apache Hive | 3.1.3 |

| Aplikasi | Versi |
| --- | --- |
| Apache Tez | 0.9.2 |

## Perubahan khusus mesin, penyempurnaan, dan masalah yang diselesaikan

Tabel berikut mencantumkan fitur khusus mesin baru.

| Perubahan | Deskripsi |
| --- | --- |
| Fitur | Penjadwal Tez sekarang mendukung preemptio n tugas Tez alih-alih preemption wadah |

# EMR Serverless 6.6.0

Tabel berikut mencantumkan versi aplikasi yang tersedia EMR Serverless 6.6.0.

| Aplikasi | Versi |
| --- | --- |
| Apache Spark | 3.2.0 |
| Apache Hive | 3.1.2 |
| Apache Tez | 0.9.2 |

EMRCatatan rilis awal tanpa server

- EMRServerless mendukung klasifikasi konfigurasi Spark. `spark-defaults` Klasifikasi ini mengubah nilai dalam `spark-defaults.conf` XML file Spark. Klasifikasi konfigurasi memungkinkan Anda menyesuaikan aplikasi. Untuk informasi selengkapnya, lihat Mengkonfigurasi aplikasi.
- EMRTanpa server mendukung klasifikasi konfigurasi Hive`hive-site`,,, `tez-site` dan. `emrfs-site` `core-site` Klasifikasi ini dapat mengubah nilai dalam file Hive, `hive-site.xml` file Tez, EMRFS pengaturan AmazonEMR, atau `tez-site.xml` file Hadoop, masing-masing. `core-site.xml` Klasifikasi konfigurasi memungkinkan Anda menyesuaikan aplikasi. Untuk informasi selengkapnya, lihat Mengkonfigurasi aplikasi.

Perubahan khusus mesin, penyempurnaan, dan masalah yang diselesaikan

- Tabel berikut mencantumkan Hive dan Tez backports.

  Hive dan Tez berubah

| Perubahan | Deskripsi |
| --- | --- |
| Backport | TEZ-4430: Memperbaiki masalah dengan properti `tez.task.launch.cmd-opts` |
| Backport | HIVE-25971: Memperbaiki penundaan shutdown tugas Tez karena kumpulan utas yang di-cache terbuka |

# Riwayat dokumen

Tabel berikut menjelaskan perubahan penting pada dokumentasi sejak rilis terakhir EMR Serverless. Untuk informasi lebih lanjut tentang pembaruan dokumentasi ini, Anda dapat berlangganan RSS umpan.

| Perubahan | Deskripsi | Tanggal |
| --- | --- | --- |
| Rilis baru | EMR Serverless 7.2.0 | Juli 25, 2024 |
| Rilis baru | EMR Serverless 7.1.0 | April 17, 2024 |
| Perbarui ke kebijakan yang ada. | Menambahkan yang baru `Sid` `CloudWatchPolicyStatement` dan `EC2PolicyStatement` ke mazonEMRServerless ServiceRolePolicy kebijakan A. | Januari 25, 2024 |
| Rilis baru | EMR Serverless 7.0.0 | Desember 29, 2023 |
| Rilis baru | EMR Serverless 6.15.0 | 17 November 2023 |
| Fitur baru | Konfigurasikan beberapa IAM peran untuk diasumsikan saat Anda mengakses bucket Amazon S3 di akun berbeda dari EMR Tanpa Server (6.11 dan lebih tinggi) | 18 Oktober 2023 |
| Rilis baru | EMR Serverless 6.14.0 | 17 Oktober 2023 |
| Fitur baru | Konfigurasi aplikasi default untuk Tanpa EMR Server | 25 September 2023 |
| Perbarui ke properti Hive default | Memperbarui nilai default untuk properti pekerjaan `hive.driv` | 12 September 2023 |

| | | |
|---|---|---|
| | er.disk hive.tez. disk.size ,hive.tez. auto.reducer.paral lelism ,, dan tez.group ing.min-size Hive. | |
| Rilis baru | EMR Serverless 6.13.0 | 11 September 2023 |
| Rilis baru | EMR Serverless 6.12.0 | 21 Juli 2023 |
| Rilis baru | EMR Serverless 6.11.0 | 8 Juni 2023 |
| Memperbarui ke kebijakan peran terkait layanan | Memperbarui AmazonEMR ServerlessServiceR olePolicy SLRperan untuk mempublikasikan penggunaan tingkat akun di "AWS/Usage" namespace. | 20 April 2023 |
| EMR Serverless ketersediaan umum (GA) | Ini adalah rilis publik pertama dari EMR Serverless. | 1 Juni 2022 |

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