aws

Benutzerhandbuch für Amazon EMR Serverless

Amazon EMR

Amazon EMR: Benutzerhandbuch für Amazon EMR Serverless

Table of Contents

Was ist Amazon EMR Serverless?

Amazon EMR Serverless ist eine Bereitstellungsoption für Amazon EMR, die eine serverlose Laufzeitumgebung bietet. Dies vereinfacht den Betrieb von Analyseanwendungen, die die neuesten Open-Source-Frameworks wie Apache Spark und Apache Hive verwenden. Mit EMR Serverless müssen Sie keine Cluster konfigurieren, optimieren, sichern oder betreiben, um Anwendungen mit diesen Frameworks auszuführen.

Mit EMR Serverless können Sie vermeiden, dass Ressourcen für Ihre Datenverarbeitungsaufgaben zu hoch oder zu niedrig bereitgestellt werden. EMR Serverless ermittelt automatisch die Ressourcen, die die Anwendung benötigt, ruft diese Ressourcen für die Verarbeitung Ihrer Jobs ab und gibt die Ressourcen frei, wenn die Jobs abgeschlossen sind. Für Anwendungsfälle, in denen Anwendungen innerhalb von Sekunden eine Antwort benötigen, wie z. B. interaktive Datenanalysen, können Sie die Ressourcen, die die Anwendung benötigt, bei der Erstellung der Anwendung vorab initialisieren.

Mit EMR Serverless profitieren Sie weiterhin von den Vorteilen von Amazon EMR, wie z. B. Open-Source-Kompatibilität, Parallelität und optimierte Laufzeitleistung für beliebte Frameworks.

EMR Serverless ist für Kunden geeignet, die den Betrieb von Anwendungen mithilfe von Open-Source-Frameworks vereinfachen möchten. Es bietet einen schnellen Start von Jobs, automatisches Kapazitätsmanagement und einfache Kostenkontrolle.

Konzepte

In diesem Abschnitt behandeln wir die Begriffe und Konzepte von EMR Serverless, die in unserem EMR Serverless User Guide enthalten sind.

Version veröffentlichen

Eine Amazon EMR-Version besteht aus einer Reihe von Open-Source-Anwendungen aus dem Big-Data-Ökosystem. Jede Version enthält verschiedene Big-Data-Anwendungen, Komponenten und Funktionen, die Sie für EMR Serverless auswählen, um sie bereitzustellen und zu konfigurieren, damit sie Ihre Anwendungen ausführen können. Wenn Sie eine Anwendung erstellen, geben Sie deren Release-Version an. Wählen Sie die Amazon EMR-Release-Version und die Open-Source-Framework-Version, die Sie in Ihrer Anwendung verwenden möchten. Weitere Informationen zu Vorabversionen finden Sie unter. Serverlose Release-Versionen von Amazon EMR

Anwendung

Mit EMR Serverless können Sie eine oder mehrere EMR Serverless-Anwendungen erstellen, die Open-Source-Analyse-Frameworks verwenden. Um eine Anwendung zu erstellen, geben Sie die folgenden Attribute an:

- Die Amazon EMR-Release-Version für die Open-Source-Framework-Version, die Sie verwenden möchten. Informationen zur Bestimmung Ihrer Release-Version finden Sie unter. Serverlose Release-Versionen von Amazon EMR
- Die spezifische Laufzeit, die Ihre Anwendung verwenden soll, z. B. Apache Spark oder Apache Hive.

Nachdem Sie eine Anwendung erstellt haben, reichen Sie Datenverarbeitungsaufträge oder interaktive Anfragen an Ihre Anwendung ein.

Jede serverlose EMR-Anwendung läuft auf einer sicheren Amazon Virtual Private Cloud (VPC), strikt getrennt von anderen Anwendungen. Verwenden Sie außerdem AWS Identity and Access Management (IAM) -Richtlinien, um zu definieren, welche Benutzer und Rollen auf die Anwendung zugreifen können. Sie können auch Grenzwerte festlegen, um die durch die Anwendung anfallenden Nutzungskosten zu kontrollieren und nachzuverfolgen.

Erwägen Sie, mehrere Anwendungen zu erstellen, wenn Sie Folgendes tun müssen:

- Verwenden Sie verschiedene Open-Source-Frameworks
- Verwenden Sie verschiedene Versionen von Open-Source-Frameworks für unterschiedliche Anwendungsfälle
- Führen Sie A/B Tests durch, wenn Sie von einer Version auf eine andere aktualisieren
- Pflegen Sie separate logische Umgebungen für Test- und Produktionsszenarien
- Stellen Sie separate logische Umgebungen für verschiedene Teams mit unabhängiger Kostenkontrolle und Nutzungsverfolgung bereit
- Trennen Sie verschiedene line-of-business Anwendungen

EMR Serverless ist ein regionaler Service, der die Ausführung von Workloads in mehreren Availability Zones in einer Region vereinfacht. Weitere Informationen zur Verwendung von Anwendungen mit EMR Serverless finden Sie unter. Interagieren Sie mit einer serverlosen EMR-Anwendung und konfigurieren Sie sie

Aufgabenausführung

Eine Auftragsausführung ist eine Anforderung, die an eine serverlose EMR-Anwendung gesendet wird und die Anwendung asynchron ausführt und bis zum Abschluss verfolgt. Beispiele für Jobs sind eine HiveQL-Abfrage, die Sie an eine Apache Hive-Anwendung senden, oder ein PySpark Datenverarbeitungsskript, das Sie an eine Apache Spark-Anwendung senden. Wenn Sie einen Job einreichen, müssen Sie eine in IAM verfasste Runtime-Rolle angeben, die der Job für den Zugriff auf AWS Ressourcen wie Amazon S3 S3-Objekte verwendet. Sie können mehrere Anfragen zur Auftragsausführung an eine Anwendung senden, und für jede Auftragsausführung kann eine andere Laufzeitrolle für den Zugriff auf Ressourcen verwendet werden. AWS Eine serverlose EMR-Anwendung beginnt mit der Ausführung von Jobs, sobald sie sie empfängt, und führt mehrere Jobanfragen gleichzeitig aus. Weitere Informationen darüber, wie EMR Serverless Jobs ausführt, finden Sie unter. Ausführen von Aufgaben

Worker

Eine serverlose EMR-Anwendung verwendet intern Worker, um Ihre Workloads auszuführen. Die Standardgrößen dieser Worker basieren auf Ihrem Anwendungstyp und der Amazon EMR-Release-Version. Wenn Sie eine Auftragsausführung planen, überschreiben Sie diese Größen.

Wenn Sie einen Job einreichen, berechnet EMR Serverless die Ressourcen, die die Anwendung für den Job benötigt, und plant Mitarbeiter ein. EMR Serverless unterteilt Ihre Workloads in Aufgaben, lädt Bilder herunter, stellt Mitarbeiter bereit und richtet sie ein und nimmt sie nach Abschluss des Auftrags wieder in Betrieb. EMR Serverless skaliert Mitarbeiter automatisch nach oben oder unten, je nach Arbeitslast und Parallelität, die in jeder Phase des Auftrags erforderlich sind. Durch diese automatische Skalierung müssen Sie nicht mehr abschätzen, wie viele Mitarbeiter die Anwendung zur Ausführung Ihrer Workloads benötigt.

Vorinitialisierte Kapazität

EMR Serverless bietet eine vorinitialisierte Kapazitätsfunktion, mit der Mitarbeiter innerhalb von Sekunden initialisiert und bereit sind, zu antworten. Diese Kapazität schafft effektiv einen warmen Pool von Mitarbeitern für eine Anwendung. Um diese Funktion für jede Anwendung zu konfigurieren, legen Sie den `initial-capacity` Parameter einer Anwendung fest. Wenn Sie vorinitialisierte Kapazität konfigurieren, können Jobs sofort gestartet werden, sodass Sie iterative Anwendungen und zeitkritische Jobs implementieren können. Weitere Informationen zu vorinitialisierten Workern finden Sie unter. Konfiguration einer Anwendung bei der Arbeit mit EMR Serverless

EMR Studio

EMR Studio ist die Benutzerkonsole für die Verwaltung Ihrer EMR Serverless-Anwendungen. Wenn in Ihrem Konto kein EMR Studio vorhanden ist, wenn Sie Ihre erste EMR Serverless-Anwendung erstellen, erstellen wir automatisch eines für Sie. Greifen Sie entweder über die Amazon EMR-Konsole auf EMR Studio zu oder aktivieren Sie den Verbundzugriff von Ihrem Identity Provider (IdP) über IAM oder IAM Identity Center. Auf diese Weise können Benutzer ohne direkten Zugriff auf die Amazon EMR-Konsole auf Studio zugreifen und EMR-Anwendungen ohne direkten Zugriff auf die Amazon EMR-Konsole verwalten. Weitere Informationen darüber, wie EMR Serverless-Anwendungen mit EMR Studio funktionieren, finden Sie unter und. Eine serverlose EMR-Anwendung von der EMR Studio-Konsole aus erstellen Jobs von der EMR Studio-Konsole aus ausführen

Voraussetzungen für den Einstieg in EMR Serverless

In diesem Abschnitt werden die administrativen Voraussetzungen für die Ausführung von EMR Serverless beschrieben. Dazu gehören die Kontokonfiguration und die Verwaltung von Berechtigungen.

Themen

Melde dich an für ein AWS-Konto

Wenn Sie noch keine haben AWS-Konto, führen Sie die folgenden Schritte aus, um eine zu erstellen.

Um sich für eine anzumelden AWS-Konto

1. Öffnen Sie https://portal.aws.amazon.com/billing/die Anmeldung.
2. Folgen Sie den Online-Anweisungen.

   Während der Anmeldung erhalten Sie einen Telefonanruf oder eine Textnachricht und müssen einen Verifizierungscode über die Telefontasten eingeben.

   Wenn Sie sich für eine anmelden AWS-Konto, Root-Benutzer des AWS-Kontoswird eine erstellt. Der Root-Benutzer hat Zugriff auf alle AWS-Services und Ressourcen des Kontos. Als bewährte Sicherheitsmethode weisen Sie einem Benutzer Administratorzugriff zu und verwenden Sie nur den Root-Benutzer, um Aufgaben auszuführen, die Root-Benutzerzugriff erfordern.

AWS sendet Ihnen nach Abschluss des Anmeldevorgangs eine Bestätigungs-E-Mail. Du kannst jederzeit deine aktuellen Kontoaktivitäten einsehen und dein Konto verwalten, indem du zu https://aws.amazon.com/gehst und Mein Konto auswählst.

Erstellen eines Benutzers mit Administratorzugriff

Nachdem Sie sich für einen angemeldet haben AWS-Konto, sichern Sie Ihren Root-Benutzer des AWS-Kontos AWS IAM Identity Center, aktivieren und erstellen Sie einen Administratorbenutzer, sodass Sie den Root-Benutzer nicht für alltägliche Aufgaben verwenden.

Sichern Sie Ihre Root-Benutzer des AWS-Kontos

1. Melden Sie sich AWS-Managementkonsoleals Kontoinhaber an, indem Sie Root-Benutzer auswählen und Ihre AWS-Konto E-Mail-Adresse eingeben. Geben Sie auf der nächsten Seite Ihr Passwort ein.

   Hilfe bei der Anmeldung mit dem Root-Benutzer finden Sie unter Anmelden als Root-Benutzer im AWS-Anmeldung -Benutzerhandbuch zu.

2. Aktivieren Sie die Multi-Faktor-Authentifizierung (MFA) für den Root-Benutzer.

   Anweisungen finden Sie unter Aktivieren eines virtuellen MFA-Geräts für Ihren AWS-Konto Root-Benutzer (Konsole) im IAM-Benutzerhandbuch.

Erstellen eines Benutzers mit Administratorzugriff

1. Aktivieren Sie das IAM Identity Center.

   Anweisungen finden Sie unter Aktivieren AWS IAM Identity Center im AWS IAM Identity Center -Benutzerhandbuch.

2. Gewähren Sie einem Administratorbenutzer im IAM Identity Center Benutzerzugriff.

   Ein Tutorial zur Verwendung von IAM-Identity-Center-Verzeichnis als Identitätsquelle finden Sie IAM-Identity-Center-Verzeichnis im Benutzerhandbuch unter Benutzerzugriff mit der Standardeinstellung konfigurieren.AWS IAM Identity Center

Anmelden als Administratorbenutzer

- Um sich mit Ihrem IAM-Identity-Center-Benutzer anzumelden, verwenden Sie die Anmelde-URL, die an Ihre E-Mail-Adresse gesendet wurde, als Sie den IAM-Identity-Center-Benutzer erstellt haben.

Hilfe bei der Anmeldung mit einem IAM Identity Center-Benutzer finden Sie im AWS-Anmeldung Benutzerhandbuch unter Anmeldung beim AWS Access-Portal.

Weiteren Benutzern Zugriff zuweisen

1. Erstellen Sie im IAM-Identity-Center einen Berechtigungssatz, der den bewährten Vorgehensweisen für die Anwendung von geringsten Berechtigungen folgt.

   Anweisungen hierzu finden Sie unter Berechtigungssatz erstellen im AWS IAM Identity Center - Benutzerhandbuch.
2. Weisen Sie Benutzer einer Gruppe zu und weisen Sie der Gruppe dann Single Sign-On-Zugriff zu.

   Eine genaue Anleitung finden Sie unter Gruppen hinzufügen im AWS IAM Identity Center - Benutzerhandbuch.

Erteilen Sie Berechtigungen

In Produktionsumgebungen empfehlen wir, detailliertere Richtlinien zu verwenden. Beispiele für solche Richtlinien finden Sie unter. Beispiele für Benutzerzugriffsrichtlinien für EMR Serverless Weitere Informationen zur Zugriffsverwaltung finden Sie unter AWS Ressourcen zur Zugriffsverwaltung im IAM-Benutzerhandbuch.

Verwenden Sie für Benutzer, die mit EMR Serverless in einer Sandbox-Umgebung beginnen müssen, eine Richtlinie, die der folgenden ähnelt:

JSON

```
{
  "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/*"
      ]
    }
  ]
}
```

Um Zugriff zu gewähren, fügen Sie Ihren Benutzern, Gruppen oder Rollen Berechtigungen hinzu:

- Benutzer und Gruppen in: AWS IAM Identity Center

  Erstellen Sie einen Berechtigungssatz. Befolgen Sie die Anweisungen unter Erstellen eines Berechtigungssatzes im AWS IAM Identity Center -Benutzerhandbuch.
- Benutzer, die in IAM über einen Identitätsanbieter verwaltet werden:

  Erstellen Sie eine Rolle für den Identitätsverbund. Befolgen Sie die Anleitung unter Eine Rolle für einen externen Identitätsanbieter (Verbund) erstellen im IAM-Benutzerhandbuch.
- IAM-Benutzer:
  - Erstellen Sie eine Rolle, die Ihr Benutzer annehmen kann. Befolgen Sie die Anleitung unter Eine Rolle für einen IAM-Benutzer erstellen im IAM-Benutzerhandbuch.
  - (Nicht empfohlen) Weisen Sie einem Benutzer eine Richtlinie direkt zu oder fügen Sie einen Benutzer zu einer Benutzergruppe hinzu. Befolgen Sie die Anweisungen unter Hinzufügen von Berechtigungen zu einem Benutzer (Konsole) im IAM-Benutzerhandbuch.

Erteilen programmgesteuerten Zugriffs

Benutzer benötigen programmatischen Zugriff, wenn sie mit AWS außerhalb des AWS-Managementkonsole interagieren möchten. Die Art und Weise, wie programmatischer Zugriff gewährt wird, hängt vom Benutzertyp ab, der zugreift. AWS

Um Benutzern programmgesteuerten Zugriff zu gewähren, wählen Sie eine der folgenden Optionen.

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
IAM	(Empfohlen) Verwenden Sie Konsolenanmeldeinformationen als temporäre Anmeldeinformationen, um programmatische Anfragen an AWS CLI AWS SDKs, oder zu signieren. AWS APIs	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten. • Informationen zu den AWS CLI finden Sie unter Anmeldung für AWS lokale Entwicklung im AWS

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
		Command Line Interface Benutzerhandbuch. • Weitere Informationen finden Sie unter Anmeldung für AWS lokale Entwicklung im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch. AWS SDKs
Mitarbeiteridentität (Benutzer, die in IAM Identity Center verwaltet werden)	Verwenden Sie temporäre Anmeldeinformationen, um programmatische Anfragen an das AWS CLI AWS SDKs, oder AWS APIs zu signieren.	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten. • Informationen zu den AWS CLI finden Sie unter Konfiguration der AWS CLI zur Verwendung AWS IAM Identity Center im AWS Command Line Interface Benutzerhandbuch. • Informationen zu AWS SDKs Tools und AWS APIs finden Sie unter IAM Identity Center-Authentifizierung im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch.
IAM	Verwenden Sie temporäre Anmeldeinformationen, um programmatische Anfragen an das AWS CLI AWS SDKs, oder zu signieren. AWS APIs	Folgen Sie den Anweisungen unter Verwenden temporärer Anmeldeinformationen mit AWS Ressourcen im IAM-Benutzerhandbuch.

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
IAM	(Nicht empfohlen) Verwenden Sie langfristige Anmeldeinformationen, um programmatische Anfragen an das AWS CLI AWS SDKs, oder zu signieren. AWS APIs	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten. • Informationen dazu AWS CLI finden Sie unter Authentifizierung mithilfe von IAM-Benutzeranmeldeinformationen im AWS Command Line Interface Benutzerhandbuch. • Informationen zu AWS SDKs und Tools finden Sie unter Authentifizieren mit langfristigen Anmeldeinformationen im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch. • Weitere Informationen finden Sie unter Verwaltung von Zugriffsschlüsseln für IAM-Benutzer im IAM-Benutzerhandbuch. AWS APIs

Installieren und konfigurieren Sie den AWS CLI

Wenn Sie EMR Serverless verwenden möchten APIs, installieren Sie die neueste Version von AWS Command Line Interface ()AWS CLI. Sie müssen EMR Serverless nicht von der AWS CLI EMR Studio-Konsole aus verwenden und können ohne die CLI loslegen, indem Sie die Schritte unter befolgen. Erste Schritte mit EMR Serverless über die Konsole

Um das einzurichten AWS CLI

1. Informationen zur Installation der neuesten Version von AWS CLI für macOS, Linux oder Windows finden Sie unter Installation oder Aktualisierung der neuesten Version von AWS CLI.
2. Informationen zur Konfiguration AWS CLI und sicheren Einrichtung Ihres Zugriffs auf AWS-Services, einschließlich EMR Serverless, finden Sie unter Schnellkonfiguration mit. `aws configure`
3. Um das Setup zu überprüfen, geben Sie in der DataBrew Befehlszeile den folgenden Befehl ein.

   ```
   aws emr-serverless help
   ```

   AWS CLI Befehle verwenden den Standard AWS-Region aus Ihrer Konfiguration, sofern Sie ihn nicht mit einem Parameter oder einem Profil festlegen. Um Ihren AWS-Region mit einem Parameter festzulegen, fügen Sie den `--region` Parameter jedem Befehl hinzu.

   Um Ihr AWS-Region Profil festzulegen, fügen Sie zunächst ein benanntes Profil in die `~/.aws/config` Datei oder die `%UserProfile%/.aws/config` Datei ein (für Microsoft Windows). Folgen Sie den Schritten unter Benannte Profile für AWS CLI. Legen Sie als Nächstes Ihre AWS-Region und andere Einstellungen mit einem Befehl fest, der dem im folgenden Beispiel ähnelt.

   ```
   [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
   ```

Öffnen Sie die -Konsole

Die meisten konsolenorientierten Themen in diesem Abschnitt beginnen in der Amazon EMR-Konsole. Wenn Sie noch nicht bei Ihrem angemeldet sind AWS-Konto, melden Sie sich an, öffnen Sie dann die Amazon EMR-Konsole und fahren Sie mit dem nächsten Abschnitt fort, um mit den ersten Schritten mit Amazon EMR fortzufahren.

Erste Schritte mit Amazon EMR Serverless

Dieses Tutorial hilft Ihnen bei den ersten Schritten mit EMR Serverless, wenn Sie einen Spark- oder Hive-Beispiel-Workload bereitstellen. Sie werden Ihre eigene Anwendung erstellen, ausführen und debuggen. In den meisten Teilen dieses Tutorials zeigen wir Standardoptionen.

Bevor Sie eine serverlose EMR-Anwendung starten, führen Sie die folgenden Aufgaben aus.

Themen

Erteilen Sie Berechtigungen zur Verwendung von EMR Serverless

Um EMR Serverless verwenden zu können, benötigen Sie eine Benutzer- oder IAM-Rolle mit einer angehängten Richtlinie, die Berechtigungen für EMR Serverless gewährt. Folgen Sie den Anweisungen unter, um einen Benutzer zu erstellen und diesem Benutzer die entsprechende Richtlinie zuzuweisen. Erteilen Sie Berechtigungen

Speicher für EMR Serverless vorbereiten

In diesem Tutorial verwenden Sie einen S3-Bucket, um Ausgabedateien und Protokolle aus dem Spark- oder Hive-Beispiel-Workload zu speichern, den Sie mit einer serverlosen EMR-Anwendung ausführen werden. Um einen Bucket zu erstellen, folgen Sie den Anweisungen unter Bucket erstellen im Amazon Simple Storage Service Console-Benutzerhandbuch. Ersetzen Sie alle weiteren Verweise auf *amzn-s3-demo-bucket* durch den Namen des neu erstellten Buckets.

Erstellen Sie ein EMR Studio zur Ausführung interaktiver Workloads

Wenn Sie EMR Serverless verwenden möchten, um interaktive Abfragen über Notebooks auszuführen, die in EMR Studio gehostet werden, müssen Sie einen S3-Bucket und die Mindestdienstrolle für EMR Serverless angeben, um einen Workspace zu erstellen. Die Schritte zur Einrichtung finden Sie unter EMR Studio einrichten im Amazon EMR Management Guide. Weitere Informationen zu interaktiven Workloads finden Sie unter. Führen Sie interaktive Workloads mit EMR Serverless über EMR Studio aus

Erstellen Sie eine Job-Runtime-Rolle

Auftragsausführungen in EMR Serverless verwenden eine Runtime-Rolle, die zur Laufzeit detaillierte Berechtigungen für bestimmte AWS-Services Ressourcen bereitstellt. In diesem Tutorial hostet ein öffentlicher S3-Bucket die Daten und Skripte. Der Bucket *amzn-s3-demo-bucket* speichert die Ausgabe.

Um eine Job-Runtime-Rolle einzurichten, erstellen Sie zunächst eine Runtime-Rolle mit einer Vertrauensrichtlinie, damit EMR Serverless die neue Rolle verwenden kann. Als Nächstes fügen Sie dieser Rolle die erforderliche S3-Zugriffsrichtlinie hinzu. Die folgenden Schritte führen Sie durch den Prozess.

Console

1. Navigieren Sie zur IAM-Konsole unterhttps://console.aws.amazon.com/iam/.
2. Wählen Sie im linken Navigationsbereich Richtlinien aus.
3. Wählen Sie Richtlinie erstellen aus.
4. Die Seite „Richtlinie erstellen“ wird auf einer neuen Registerkarte geöffnet. Wählen Sie den Richtlinien-Editor als Json aus und fügen Sie den Richtlinien-JSON unten ein.

> **Important**
>
> Ersetzen Sie die *amzn-s3-demo-bucket* nachstehende Richtlinie durch den tatsächlichen Bucket-Namen, der in erstellt wurdeSpeicher für EMR Serverless vorbereiten. Dies ist eine grundlegende Richtlinie für den S3-Zugriff. Weitere

Beispiele für Rollen zur Laufzeit von Jobs finden Sie unterJob-Runtime-Rollen für Amazon EMR Serverless.

JSON

```
{
  "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:::amzn-s3-demo-bucket",
        "arn:aws:s3:::amzn-s3-demo-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": [
        "*"
      ]
    }
  ]
}
```

5. Wählen Sie Weiter, um einen Namen für Ihre Richtlinie einzugeben, z. `EMRServerlessS3AndGlueAccessPolicy` B. Richtlinie erstellen
6. Wählen Sie im linken Navigationsbereich der IAM-Konsole die Option Rollen aus.
7. Wählen Sie Rolle erstellen aus.
8. Wählen Sie als Rollentyp die Option Benutzerdefinierte Vertrauensrichtlinie aus und fügen Sie die folgende Vertrauensrichtlinie ein. Auf diese Weise können Jobs, die an Ihre Amazon EMR Serverless-Anwendungen gesendet wurden, in Ihrem Namen AWS-Services auf andere Jobs zugreifen.

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole"
      ],
      "Resource": "arn:aws:iam::123456789012:role/
EMRServerlessExecutionRole",
      "Sid": "AllowSTSAssumerole"
    }
```

```
  ]
}
```

9. Wählen Sie Weiter, um zur Seite „Berechtigungen hinzufügen“ zu navigieren, und wählen Sie EMRServerless dann S3. AndGlueAccessPolicy
10. Geben Sie auf der Seite Name, überprüfen und erstellen unter Rollenname einen Namen für Ihre Rolle ein, `EMRServerlessS3RuntimeRole` z. B. Um diese IAM-Rolle zu erstellen, wählen Sie Rolle erstellen aus.

CLI

1. Erstellen Sie eine Datei namens `emr-serverless-trust-policy.json`, die die Vertrauensrichtlinie enthält, die für die IAM-Rolle verwendet werden soll. Die Datei sollte die folgende Richtlinie enthalten.

   JSON

   ```
   {
     "Version":"2012-10-17",
     "Statement": [
       {
         "Sid": "EMRServerlessTrustPolicy",
         "Action": [
           "sts:AssumeRole"
         ],
         "Effect": "Allow",
         "Resource": "arn:aws:iam::123456789012:role/
   EMRServerlessExecutionRole"
       }
     ]
   }
   ```

2. Erstellen einer IAM-Rolle namens `EMRServerlessS3RuntimeRole`. Verwenden Sie die Vertrauensrichtlinie, die Sie im vorherigen Schritt erstellt haben.

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

Notieren Sie sich den ARN in der Ausgabe. Sie verwenden den ARN der neuen Rolle bei der Einreichung des Jobs, im Folgenden als*job-role-arn*.

3. Erstellen Sie eine Datei mit dem Namen`emr-sample-access-policy.json`, die die IAM-Richtlinie für Ihren Workload definiert. Dies bietet Lesezugriff auf das Skript und die in öffentlichen S3-Buckets gespeicherten Daten sowie Lese- und Schreibzugriff auf. *amzn-s3-demo-bucket*

> **Important**
>
> Ersetzen Sie die folgende Richtlinie durch den tatsächlichen Bucket-Namen, der *amzn-s3-demo-bucket* in.. erstellt wurde. Speicher für EMR Serverless vorbereiten Dies ist eine grundlegende Richtlinie für den Zugriff auf AWS Glue und S3. Weitere Beispiele für Job-Runtime-Rollen finden Sie unterJob-Runtime-Rollen für Amazon EMR Serverless.

JSON

```
{
  "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:::amzn-s3-demo-bucket",
        "arn:aws:s3:::amzn-s3-demo-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": [
        "*"
      ]
    }
  ]
}
```

4. Erstellen Sie eine IAM-Richtlinie `EMRServerlessS3AndGlueAccessPolicy` mit dem Namen der Richtliniendatei, die Sie in Schritt 3 erstellt haben. Notieren Sie sich den ARN in der Ausgabe, da Sie im nächsten Schritt den ARN der neuen Richtlinie verwenden werden.

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

Notieren Sie sich den ARN der neuen Richtlinie in der Ausgabe. Sie werden ihn *policy-arn* im nächsten Schritt ersetzen.

5. Hängen Sie die IAM-Richtlinie `EMRServerlessS3AndGlueAccessPolicy` an die Job-Runtime-Rolle `EMRServerlessS3RuntimeRole` an.

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

Erste Schritte mit EMR Serverless über die Konsole

In diesem Abschnitt wird die Arbeit mit EMR Serverless beschrieben, einschließlich der Erstellung eines EMR Studios. Außerdem wird beschrieben, wie Sie Auftragsausführungen einreichen und Protokolle anzeigen.

Schritte zum Absolvieren

- Schritt 1: Eine serverlose EMR-Anwendung erstellen
- Schritt 2: Reichen Sie eine Auftragsausführung oder einen interaktiven Workload ein
- Schritt 3: Benutzeroberfläche und Protokolle der Anwendung anzeigen
- Schritt 4: Bereinigen

Schritt 1: Eine serverlose EMR-Anwendung erstellen

Erstellen Sie wie folgt eine neue Anwendung mit EMR Serverless.

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die Amazon EMR-Konsole unter https://console.aws.amazon.com/emr.
2. Wählen Sie im linken Navigationsbereich EMR Serverless aus, um zur EMR Serverless-Landingpage zu gelangen.
3. Um EMR Serverless-Anwendungen zu erstellen oder zu verwalten, benötigen Sie die EMR Studio-Benutzeroberfläche.

    - Wenn Sie in dem Bereich, in AWS-Region dem Sie eine Anwendung erstellen möchten, bereits über ein EMR Studio verfügen, wählen Sie Anwendungen verwalten aus, um zu Ihrem EMR Studio zu gelangen, oder wählen Sie das Studio aus, das Sie verwenden möchten.

- Wenn Sie in dem Bereich, in AWS-Region dem Sie eine Anwendung erstellen möchten, kein EMR Studio haben, wählen Sie Get started und dann Create and launch Studio. EMR Serverless erstellt ein EMR Studio für Sie, damit Sie Anwendungen erstellen und verwalten können.

4. Geben Sie in der Create Studio-Benutzeroberfläche, die auf einer neuen Registerkarte geöffnet wird, den Namen, den Typ und die Release-Version für Ihre Anwendung ein. Wenn Sie nur Batch-Jobs ausführen möchten, wählen Sie „Standardeinstellungen nur für Batch-Jobs verwenden“. Wählen Sie für interaktive Workloads die Option Standardeinstellungen für interaktive Workloads verwenden aus. Mit dieser Option können Sie auch Batch-Jobs für interaktive Anwendungen ausführen. Bei Bedarf können Sie diese Einstellungen später ändern.

   Weitere Informationen finden Sie unter Studio erstellen.
5. Wählen Sie Anwendung erstellen aus, um Ihre erste Anwendung zu erstellen.

Fahren Sie mit dem nächsten Abschnitt fortSchritt 2: Reichen Sie eine Auftragsausführung oder einen interaktiven Workload ein, um eine Auftragsausführung oder einen interaktiven Workload einzureichen.

Schritt 2: Reichen Sie eine Auftragsausführung oder einen interaktiven Workload ein

Spark job run

In diesem Tutorial verwenden wir ein PySpark Skript, um zu berechnen, wie oft eindeutige Wörter in mehreren Textdateien vorkommen. Ein öffentlicher, schreibgeschützter S3-Bucket speichert sowohl das Skript als auch den Datensatz.

Um einen Spark-Job auszuführen

1. Laden Sie das `wordcount.py` Beispielskript mit dem folgenden Befehl in Ihren neuen Bucket hoch.

   ```
   aws s3 cp s3://us-east-1.elasticmapreduce/emr-containers/samples/wordcount/
   scripts/wordcount.py s3://amzn-s3-demo-bucket/scripts/
   ```
2. Wenn Sie den Vorgang abschließen, Schritt 1: Eine serverlose EMR-Anwendung erstellen gelangen Sie zur Seite mit den Anwendungsdetails in EMR Studio. Wählen Sie dort die Option Job einreichen aus.

3. Gehen Sie auf der Seite „Job einreichen“ wie folgt vor.

   - Geben Sie im Feld Name den Namen ein, den Sie Ihren Job Run nennen möchten.
   - Geben Sie im Feld Runtime-Rolle den Namen der Rolle ein, in der Sie sie erstellt habenErstellen Sie eine Job-Runtime-Rolle.
   - Geben Sie im Feld Skriptspeicherort den Wert `s3://amzn-s3-demo-bucket/scripts/wordcount.py` als S3-URI ein.
   - Geben Sie im Feld Skriptargumente den Wert ein`["s3://amzn-s3-demo-bucket/emr-serverless-spark/output"]`.
   - Wählen Sie im Bereich Spark-Eigenschaften die Option Als Text bearbeiten aus und geben Sie die folgenden Konfigurationen ein.

     ```
     --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. Um den Job-Lauf zu starten, wählen Sie Job einreichen.
5. Auf der Registerkarte „Auftragsausführungen“ sollten Sie sehen, dass Ihr neuer Job mit dem Status „Wird ausgeführt“ ausgeführt wird.

Hive job run

In diesem Teil des Tutorials erstellen wir eine Tabelle, fügen einige Datensätze ein und führen eine Zählaggregationsabfrage aus. Um den Hive-Job auszuführen, erstellen Sie zunächst eine Datei, die alle Hive-Abfragen enthält, die als Teil eines einzelnen Jobs ausgeführt werden sollen, laden Sie die Datei auf S3 hoch und geben Sie diesen S3-Pfad an, wenn Sie den Hive-Job starten.

Um einen Hive-Job auszuführen

1. Erstellen Sie eine Datei mit dem Namen`hive-query.ql`, die alle Abfragen enthält, die Sie in Ihrem Hive-Job ausführen möchten.

   ```
   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. Laden Sie `hive-query.ql` es mit dem folgenden Befehl in Ihren S3-Bucket hoch.

```
aws s3 cp hive-query.ql s3://amzn-s3-demo-bucket/emr-serverless-hive/query/hive-query.ql
```

3. Wenn Sie den Vorgang abschließen, Schritt 1: Eine serverlose EMR-Anwendung erstellen gelangen Sie zur Seite mit den Anwendungsdetails in EMR Studio. Wählen Sie dort die Option Job einreichen aus.
4. Gehen Sie auf der Seite „Job einreichen“ wie folgt vor.
   - Geben Sie im Feld Name den Namen ein, den Sie Ihren Job Run nennen möchten.
   - Geben Sie im Feld Runtime-Rolle den Namen der Rolle ein, in der Sie sie erstellt habenErstellen Sie eine Job-Runtime-Rolle.
   - Geben Sie im Feld Skriptspeicherort den Wert `s3://amzn-s3-demo-bucket/emr-serverless-hive/query/hive-query.ql` als S3-URI ein.
   - Wählen Sie im Abschnitt Hive-Eigenschaften die Option Als Text bearbeiten aus und geben Sie die folgenden Konfigurationen ein.

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

   - Wählen Sie im Abschnitt Auftragskonfiguration die Option Als JSON bearbeiten aus und geben Sie den folgenden JSON-Code ein.

     ```
     {
        "applicationConfiguration":
        [{
            "classification": "hive-site",
               "properties": {
                   "hive.exec.scratchdir": "s3://amzn-s3-demo-bucket/emr-serverless-hive/hive/scratch",
                   "hive.metastore.warehouse.dir": "s3://amzn-s3-demo-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. Um die Auftragsausführung zu starten, wählen Sie Job einreichen aus.
6. Auf der Registerkarte „Auftragsausführungen“ sollten Sie sehen, dass Ihr neuer Job mit dem Status „Wird ausgeführt“ ausgeführt wird.

Interactive workload

Mit Amazon EMR 6.14.0 und höher können Sie Notebooks verwenden, die in EMR Studio gehostet werden, um interaktive Workloads für Spark in EMR Serverless auszuführen. Weitere Informationen, einschließlich Berechtigungen und Voraussetzungen, finden Sie unter. Führen Sie interaktive Workloads mit EMR Serverless über EMR Studio aus

Nachdem Sie Ihre Anwendung erstellt und die erforderlichen Berechtigungen eingerichtet haben, führen Sie die folgenden Schritte aus, um ein interaktives Notizbuch mit EMR Studio auszuführen:

1. Navigieren Sie in EMR Studio zur Registerkarte Workspaces. Wenn Sie noch einen Amazon S3 S3-Speicherort und eine EMR Studio-Servicerolle konfigurieren müssen, klicken Sie im Banner oben auf dem Bildschirm auf die Schaltfläche Studio konfigurieren.
2. Um auf ein Notizbuch zuzugreifen, wählen Sie einen Workspace aus oder erstellen Sie einen neuen Workspace. Verwenden Sie den Schnellstart, um Ihren Workspace in einem neuen Tab zu öffnen.
3. Gehe zum neu geöffneten Tab. Wählen Sie in der linken Navigationsleiste das Compute-Symbol aus. Wählen Sie EMR Serverless als Compute-Typ aus.
4. Wählen Sie die interaktive Anwendung aus, die Sie im vorherigen Abschnitt erstellt haben.
5. Geben Sie im Feld Runtime-Rolle den Namen der IAM-Rolle ein, die Ihre EMR Serverless-Anwendung für die Jobausführung annehmen kann. Weitere Informationen zu Runtime-Rollen finden Sie unter Job Runtime Roles im Amazon EMR Serverless User Guide.
6. Wählen Sie Anhängen aus. Dies kann bis zu einer Minute dauern. Die Seite wird aktualisiert, wenn sie angehängt ist.
7. Wählen Sie einen Kernel und starten Sie ein Notizbuch. Sie können auch Beispielnotizbücher auf EMR Serverless durchsuchen und sie in Ihren Workspace kopieren. Um auf die Beispielnotizbücher zuzugreifen, navigieren Sie zum **`{...}`**Menü in der linken Navigationsleiste und suchen Sie nach Notizbüchern, die `serverless` im Notizbuch den Dateinamen haben.

8. Im Notizbuch können Sie auf den Link zum Treiberprotokoll und einen Link zur Apache Spark-Benutzeroberfläche zugreifen, einer Echtzeitschnittstelle, die Messwerte zur Überwachung Ihres Jobs bereitstellt. Weitere Informationen finden Sie unter Monitoring EMR Serverless-Anwendungen und -Jobs im Amazon EMR Serverless User Guide.

Wenn Sie eine Anwendung an einen Studio-Workspace anhängen, wird der Anwendungsstart automatisch ausgelöst, sofern er nicht bereits ausgeführt wird. Sie können die Anwendung auch vorab starten und bereithalten, bevor Sie sie an den Workspace anhängen.

Schritt 3: Benutzeroberfläche und Protokolle der Anwendung anzeigen

Um die Benutzeroberfläche der Anwendung anzuzeigen, identifizieren Sie zunächst den ausgeführten Job. Eine Option für die Benutzeroberfläche von Spark oder Hive Tez ist je nach Jobtyp in der ersten Zeile mit Optionen für diese Jobausführung verfügbar. Wählen Sie die entsprechende Option aus.

Wenn Sie sich für die Spark-Benutzeroberfläche entschieden haben, wählen Sie die Registerkarte Executors, um die Treiber- und Executor-Protokolle anzuzeigen. Wenn Sie sich für die Hive Tez-Benutzeroberfläche entschieden haben, wählen Sie die Registerkarte Alle Aufgaben, um die Protokolle anzuzeigen.

Sobald der Status der Auftragsausführung als Erfolgreich angezeigt wird, können Sie sich die Ausgabe des Jobs in Ihrem S3-Bucket ansehen.

Schritt 4: Bereinigen

Obwohl die von Ihnen erstellte Anwendung nach 15 Minuten Inaktivität automatisch beendet werden sollte, empfehlen wir dennoch, Ressourcen freizugeben, die Sie nicht erneut verwenden möchten.

Um die Anwendung zu löschen, navigieren Sie zur Seite „Anwendungen auflisten“. Wählen Sie die Anwendung aus, die Sie erstellt haben, und wählen Sie Aktionen → Stopp, um die Anwendung zu beenden. Wenn sich die Anwendung im `STOPPED` Status befindet, wählen Sie dieselbe Anwendung aus und wählen Sie Aktionen → Löschen.

Weitere Beispiele für die Ausführung von Spark- und Hive-Jobs finden Sie unter Verwenden von Spark-Konfigurationen bei der Ausführung von EMR Serverless-Jobs undVerwenden von Hive-Konfigurationen bei der Ausführung von EMR Serverless-Jobs.

Erste Schritte vom AWS CLI

Beginnen Sie mit EMR Serverless AWS CLI mit Befehlen, um eine Anwendung zu erstellen, Jobs auszuführen, die Ausgabe der Jobausführung zu überprüfen und Ihre Ressourcen zu löschen.

Schritt 1: Eine serverlose EMR-Anwendung erstellen

Verwenden Sie den `emr-serverless create-application`Befehl, um Ihre erste EMR Serverless-Anwendung zu erstellen. Sie müssen den Anwendungstyp und das Amazon EMR-Release-Label angeben, das der Anwendungsversion zugeordnet ist, die Sie verwenden möchten. Der Name der Anwendung ist optional.

Spark

Führen Sie den folgenden Befehl aus, um eine Spark-Anwendung zu erstellen.

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

Hive

Führen Sie den folgenden Befehl aus, um eine Hive-Anwendung zu erstellen.

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

Notieren Sie sich die in der Ausgabe zurückgegebene Anwendungs-ID. Sie verwenden die ID, um die Bewerbung und bei der Einreichung des Jobs zu starten, im Folgenden als*`application-id`*.

Bevor Sie fortfahrenSchritt 2: Senden Sie einen Job Run an Ihre EMR Serverless-Anwendung, stellen Sie sicher, dass Ihre Bewerbung den `CREATED` Status mit der `get-application`API erreicht hat.

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

EMR Serverless stellt Mitarbeiter zusammen, um Ihre angeforderten Jobs zu bearbeiten. Standardmäßig werden diese bei Bedarf erstellt. Sie können jedoch auch eine vorinitialisierte Kapazität angeben, indem Sie den `initialCapacity` Parameter bei der Erstellung der Anwendung festlegen. Mit dem Parameter können Sie auch die maximale Gesamtkapazität einschränken, die eine Anwendung verwenden kann. `maximumCapacity` Weitere Informationen zu diesen Optionen finden Sie unter Konfiguration einer Anwendung bei der Arbeit mit EMR Serverless.

Schritt 2: Senden Sie einen Job Run an Ihre EMR Serverless-Anwendung

Jetzt ist Ihre EMR Serverless-Anwendung bereit, Jobs auszuführen.

Spark

In diesem Schritt verwenden wir ein PySpark Skript, um zu berechnen, wie oft eindeutige Wörter in mehreren Textdateien vorkommen. Ein öffentlicher, schreibgeschützter S3-Bucket speichert sowohl das Skript als auch den Datensatz. Die Anwendung sendet die Ausgabedatei und die Protokolldaten aus der Spark-Laufzeit an `/output` die `/logs` Verzeichnisse im S3-Bucket, die Sie erstellt haben.

Um einen Spark-Job auszuführen

1. Verwenden Sie den folgenden Befehl, um das Beispielskript, das wir ausführen werden, in Ihren neuen Bucket zu kopieren.

```
aws s3 cp s3://us-east-1.elasticmapreduce/emr-containers/samples/wordcount/
scripts/wordcount.py s3://amzn-s3-demo-bucket/scripts/
```

2. Geben Sie im folgenden Befehl Ihre Anwendungs-ID ein. *application-id job-role-arn*Ersetzen Sie es durch den ARN der Laufzeitrolle, in dem Sie ihn erstellt habenErstellen Sie eine Job-Runtime-Rolle. *job-run-name*Ersetzen Sie es durch den Namen, den Sie Ihren Job Run nennen möchten. Ersetzen Sie alle *amzn-s3-demo-bucket* Zeichenketten durch den Amazon S3 S3-Bucket, den Sie erstellt haben, und fügen Sie ihn dem Pfad `/output` hinzu. Dadurch wird ein neuer Ordner in Ihrem Bucket erstellt, in den EMR Serverless die Ausgabedateien Ihrer Anwendung kopieren kann.

```
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://amzn-s3-demo-bucket/scripts/wordcount.py",
          "entryPointArguments": ["s3://amzn-s3-demo-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. Notieren Sie sich die in der Ausgabe zurückgegebene ID der Auftragsausführung. *job-run-id*Ersetzen Sie sie in den folgenden Schritten durch diese ID.

Hive

In diesem Tutorial erstellen wir eine Tabelle, fügen einige Datensätze ein und führen eine Zählaggregationsabfrage aus. Um den Hive-Job auszuführen, erstellen Sie zunächst eine Datei, die alle Hive-Abfragen enthält, die als Teil eines einzelnen Jobs ausgeführt werden sollen, laden Sie die Datei auf S3 hoch und geben Sie diesen S3-Pfad an, wenn Sie den Hive-Job starten.

Um einen Hive-Job auszuführen

1. Erstellen Sie eine Datei mit dem Namen`hive-query.ql`, die alle Abfragen enthält, die Sie in Ihrem Hive-Job ausführen möchten.

```
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. Laden Sie `hive-query.ql` es mit dem folgenden Befehl in Ihren S3-Bucket hoch.

```
aws s3 cp hive-query.ql s3://amzn-s3-demo-bucket/emr-serverless-hive/query/hive-
query.ql
```

3. Ersetzen Sie den Befehl im folgenden Befehl *application-id* durch Ihre eigene Anwendungs-ID. *job-role-arn*Ersetzen Sie es durch den ARN der Laufzeitrolle, in dem Sie ihn erstellt habenErstellen Sie eine Job-Runtime-Rolle. Ersetzen Sie alle *amzn-s3-demo-bucket* Zeichenketten durch den Amazon S3 S3-Bucket, den Sie erstellt haben, /

`output` und fügen Sie `/logs` dem Pfad und hinzu. Dadurch werden neue Ordner in Ihrem Bucket erstellt, in die EMR Serverless die Ausgabe- und Protokolldateien Ihrer Anwendung kopieren kann.

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "hive": {
          "query": "s3://amzn-s3-demo-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://amzn-s3-demo-bucket/emr-serverless-
hive/hive/scratch",
            "hive.metastore.warehouse.dir": "s3://amzn-s3-demo-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://amzn-s3-demo-bucket/emr-serverless-hive/logs"
           }
        }
    }'
```

4. Notieren Sie sich die in der Ausgabe zurückgegebene ID der Auftragsausführung. *job-run-id*Ersetzen Sie sie in den folgenden Schritten durch diese ID.

Schritt 3: Überprüfen Sie die Ausgabe Ihres Joblaufs

Die Ausführung des Auftrags sollte in der Regel 3 bis 5 Minuten dauern.

Spark

Mit dem folgenden Befehl können Sie den Status Ihres Spark-Jobs überprüfen.

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

Wenn Ihr Protokollziel auf eingestellt ists3://*amzn-s3-demo-bucket*/emr-serverless-spark/logs, finden Sie die Protokolle für diesen speziellen Job, der ausgeführt wird, unters3://*amzn-s3-demo-bucket*/emr-serverless-spark/logs/applications/*application-id*/jobs/*job-run-id*.

Für Spark-Anwendungen überträgt EMR Serverless alle 30 Sekunden Ereignisprotokolle in den sparklogs Ordner in Ihrem S3-Protokollziel. Wenn Ihr Job abgeschlossen ist, werden die Spark-Laufzeitprotokolle für den Treiber und die Executoren in Ordner hochgeladen, die entsprechend dem Worker-Typ benannt sind, z. B. oder. driver executor Die Ausgabe des PySpark Jobs wird in hochgeladen. s3://*amzn-s3-demo-bucket*/output/

Hive

Mit dem folgenden Befehl können Sie den Status Ihres Hive-Jobs überprüfen.

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

Wenn Ihr Protokollziel auf eingestellt ists3://*amzn-s3-demo-bucket*/emr-serverless-hive/logs, finden Sie die Protokolle für diesen speziellen Job, der ausgeführt wird, unters3://*amzn-s3-demo-bucket*/emr-serverless-hive/logs/applications/*application-id*/jobs/*job-run-id*.

Für Hive-Anwendungen lädt EMR Serverless kontinuierlich den Hive-Treiber in den Ordner und die Tez-Aufgabenprotokolle in den HIVE_DRIVER Ordner Ihres S3-Protokollziels hoch. TEZ_TASK Nachdem der ausgeführte Job den SUCCEEDED Status erreicht hat, ist die Ausgabe Ihrer Hive-Abfrage an dem Amazon S3 S3-Speicherort verfügbar, den Sie im monitoringConfiguration Feld von configurationOverrides angegeben haben.

Schritt 4: Bereinigen

Wenn Sie mit der Arbeit an diesem Tutorial fertig sind, sollten Sie erwägen, die von Ihnen erstellten Ressourcen zu löschen. Wir empfehlen Ihnen, Ressourcen freizugeben, die Sie nicht erneut verwenden möchten.

Löschen Sie Ihre Bewerbung

Verwenden Sie den folgenden Befehl, um eine Anwendung zu löschen.

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

Löschen Sie Ihren S3-Log-Bucket

Verwenden Sie den folgenden Befehl, um Ihren S3-Logging- und Output-Bucket zu löschen. *amzn-s3-demo-bucket*Ersetzen Sie es durch den tatsächlichen Namen des S3-Buckets, der inSpeicher für EMR Serverless vorbereiten.. erstellt wurde

```
aws s3 rm s3://amzn-s3-demo-bucket --recursive
aws s3api delete-bucket --bucket amzn-s3-demo-bucket
```

Löschen Sie Ihre Job-Runtime-Rolle

Um die Runtime-Rolle zu löschen, trennen Sie die Richtlinie von der Rolle. Anschließend können Sie sowohl die Rolle als auch die Richtlinie löschen.

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

Verwenden Sie den folgenden Befehl, um die Rolle zu löschen.

```
aws iam delete-role \
    --role-name EMRServerlessS3RuntimeRole
```

Verwenden Sie den folgenden Befehl, um die Richtlinie zu löschen, die der Rolle zugeordnet war.

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

```
    --policy-arn policy-arn
```

Weitere Beispiele für die Ausführung von Spark- und Hive-Jobs finden Sie unter Verwenden von Spark-Konfigurationen bei der Ausführung von EMR Serverless-Jobs undVerwenden von Hive-Konfigurationen bei der Ausführung von EMR Serverless-Jobs.

Interagieren Sie mit einer serverlosen EMR-Anwendung und konfigurieren Sie sie

In diesem Abschnitt erfahren Sie, wie Sie mit Ihrer Amazon EMR Serverless-Anwendung mit dem interagieren. AWS CLI Außerdem werden die Konfiguration einer Anwendung, die Durchführung von Anpassungen und die Standardeinstellungen für Spark- und Hive-Engines beschrieben.

Themen

Status der Anwendung

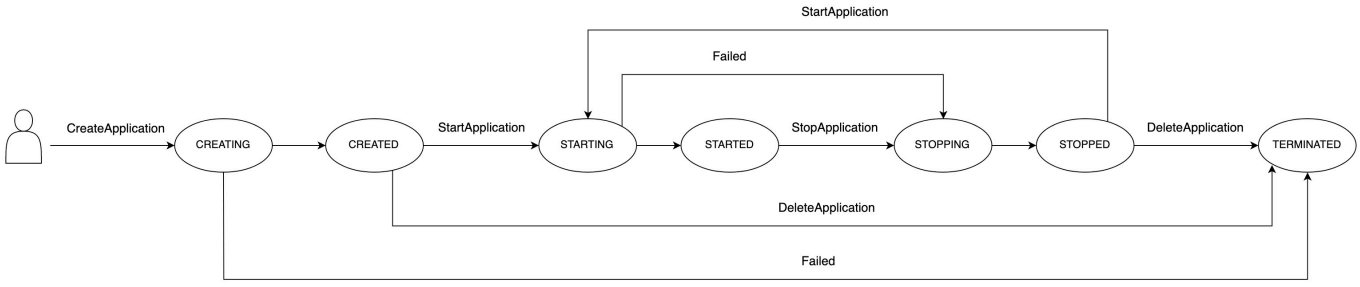
Wenn Sie eine Anwendung mit EMR Serverless erstellen, wechselt die ausgeführte Anwendung in den `CREATING` Status. Danach durchläuft er die folgenden Zustände, bis er erfolgreich ist (Abschluss mit Code `0`) oder fehlschlägt (Abschluss mit einem Code ungleich null).

Anwendungen können die folgenden Status haben:

Status	Description
Erstellen	Die Anwendung wird vorbereitet und ist noch nicht einsatzbereit.
Erstellt	Die Anwendung wurde erstellt, hat aber noch keine Kapazität bereitgestellt. Sie können die Anwendung ändern, um ihre anfängliche Kapazitätskonfiguration zu ändern.

Status	Description
Wird gestartet	Die Anwendung wird gestartet und stellt Kapazität bereit.
Gestartet	Die Bewerbung ist bereit, neue Jobs anzunehmen. Die Bewerbung akzeptiert nur Jobs, wenn sie sich in diesem Status befindet.
Wird angehalten	Alle Jobs wurden abgeschlossen und die Kapazität der Anwendung wird ausgeschöpft.
Angehalten	Die Anwendung wurde gestoppt und es werden keine Ressourcen für die Anwendung ausgeführt. Sie können die Anwendung ändern, um ihre anfängliche Kapazitätskonfiguration zu ändern.
Beendet	Die Anwendung wurde beendet und erscheint nicht auf Ihrer Anwendungsliste.

Das folgende Diagramm veranschaulicht den Verlauf der Serverless-Anwendungszustände von EMR.



Eine serverlose EMR-Anwendung von der EMR Studio-Konsole aus erstellen

Von der EMR Studio-Konsole aus können Sie EMR Serverless-Anwendungen erstellen, darauf zugreifen und sie verwalten. Folgen Sie den Anweisungen unter Erste Schritte mit der Konsole, um zur EMR Studio-Konsole zu navigieren.

Erstellen einer Anwendung

Erstellen Sie auf der Seite „Anwendung erstellen“ eine serverlose EMR-Anwendung, indem Sie die folgenden Schritte ausführen.

1. Geben Sie im Feld Name den Namen ein, den Sie für Ihre Anwendung verwenden möchten.
2. Wählen Sie im Feld Typ Spark oder Hive als Anwendungstyp aus.
3. Wählen Sie im Feld Release-Version die EMR-Release-Nummer aus.
4. Wählen Sie in den Architekturoptionen die zu verwendende Befehlssatzarchitektur aus. Weitere Informationen finden Sie unter Optionen für die serverlose Architektur von Amazon EMR.

   - arm64 — 64-Bit-ARM-Architektur; zur Verwendung von Graviton-Prozessoren
   - x86_64 — 64-Bit-x86-Architektur; zur Verwendung von x86-basierten Prozessoren
5. Für die übrigen Felder gibt es zwei Optionen zur Anwendungseinrichtung: Standardeinstellungen und benutzerdefinierte Einstellungen. Diese Felder sind optional.

   Standardeinstellungen — Mit den Standardeinstellungen können Sie schnell eine Anwendung mit vorinitialisierter Kapazität erstellen. Dazu gehören ein Treiber und ein Executor für Spark sowie ein Treiber und eine Tez Task für Hive. Die Standardeinstellungen ermöglichen keine Netzwerkkonnektivität zu Ihrem. VPCs Die Anwendung ist so konfiguriert, dass sie angehalten wird, wenn sie 15 Minuten lang inaktiv ist, und startet automatisch, wenn der Job eingereicht wird.

   Benutzerdefinierte Einstellungen — Mithilfe von benutzerdefinierten Einstellungen können Sie die folgenden Eigenschaften ändern.

   - Vorinitialisierte Kapazität — Die Anzahl der Fahrer und Ausführenden oder Hive Tez Task-Worker sowie die Größe der einzelnen Worker.
   - Anwendungsgrenzen — Die maximale Kapazität einer Anwendung.
   - Anwendungsverhalten — Das automatische Start- und Stoppverhalten der Anwendung.

- Netzwerkverbindungen — Netzwerkkonnektivität zu VPC-Ressourcen.
- Tags — Benutzerdefinierte Tags, die der Anwendung zugewiesen werden.

Weitere Informationen zu vorinitialisierter Kapazität, Anwendungslimits und Anwendungsverhalten finden Sie unter. Konfiguration einer Anwendung bei der Arbeit mit EMR Serverless Weitere Informationen zur Netzwerkkonnektivität finden Sie unter. Konfiguration des VPC-Zugriffs für serverlose EMR-Anwendungen zur Verbindung mit Daten

6. Um die Anwendung zu erstellen, wählen Sie Anwendung erstellen.

Anwendungen von der EMR Studio-Konsole aus auflisten

Sie können auf der Seite Anwendungen auflisten auf alle vorhandenen EMR Serverless-Anwendungen zugreifen. Sie können den Namen einer Anwendung wählen, um zur Detailseite für diese Anwendung zu navigieren.

Anwendungen von der EMR Studio-Konsole aus verwalten

Sie können die folgenden Aktionen für eine Anwendung entweder auf der Seite „Anwendungen auflisten“ oder auf der Detailseite einer bestimmten Anwendung ausführen.

Anwendung starten

Wählen Sie diese Option, um eine Anwendung manuell zu starten.

Anwendung beenden

Wählen Sie diese Option, um eine Anwendung manuell zu beenden. Eine Anwendung darf keine laufenden Jobs haben, um gestoppt zu werden. Weitere Informationen zu Statusübergängen bei Anwendungen finden Sie unterStatus der Anwendung.

Anwendung konfigurieren

Bearbeiten Sie die optionalen Einstellungen für eine Anwendung auf der Seite Anwendung konfigurieren. Sie können die meisten Anwendungseinstellungen ändern. Ändern Sie beispielsweise das Release-Label für eine Anwendung, um sie auf eine andere Version von Amazon EMR zu aktualisieren, oder wechseln Sie die Architektur von x86_64 auf arm64. Die anderen optionalen Einstellungen entsprechen denen, die sich im Abschnitt Benutzerdefinierte Einstellungen auf der Seite Anwendung erstellen befinden. Weitere Informationen zu den Anwendungseinstellungen finden Sie unterErstellen einer Anwendung.

Anwendung löschen

Wählen Sie diese Option, um eine Anwendung manuell zu löschen. Sie müssen eine Anwendung beenden, um sie zu löschen. Weitere Informationen zu Statusübergängen bei Anwendungen finden Sie unterStatus der Anwendung.

Interaktion mit Ihrer EMR Serverless-Anwendung auf dem AWS CLI

Erstellen AWS CLI, beschreiben und löschen Sie einzelne Anwendungen aus. Sie können auch alle Ihre Anwendungen auflisten, sodass Sie auf einen Blick darauf zugreifen können. In diesem Abschnitt wird beschrieben, wie Sie diese Aktionen ausführen. Weitere Anwendungsvorgänge wie Starten, Stoppen und Anwendungsupdates finden Sie in der EMR Serverless API Reference. Beispiele für die Verwendung der EMR Serverless API mithilfe von finden Sie in den AWS SDK für JavaJava-Beispielen in unserem GitHub Repository. Beispiele für die Verwendung der EMR Serverless API mithilfe von finden Sie in den AWS SDK für Python (Boto)Python-Beispielen in unserem GitHub Repository.

Um eine Anwendung zu erstellen, verwenden Sie. `create-application` Sie müssen `SPARK` oder `HIVE` als Anwendung angeben`type`. Dieser Befehl gibt den ARN, den Namen und die ID der Anwendung zurück.

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

Um eine Anwendung zu beschreiben, verwenden Sie sie `get-application` und geben Sie sie an`application-id`. Dieser Befehl gibt den Status und die kapazitätsbezogenen Konfigurationen für Ihre Anwendung zurück.

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

Rufen Sie an, um alle Ihre Anwendungen aufzulisten. `list-applications` Dieser Befehl gibt dieselben Eigenschaften zurück wie alle Ihre Anwendungen, schließt sie `get-application` jedoch ein.

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

Um Ihre Anwendung zu löschen, rufen Sie an `delete-application` und geben Sie Ihre an`application-id`.

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

Konfiguration einer Anwendung bei der Arbeit mit EMR Serverless

Konfigurieren Sie mit EMR Serverless die Anwendungen, die Sie verwenden. Legen Sie beispielsweise die maximale Kapazität fest, auf die eine Anwendung skaliert werden kann, konfigurieren Sie vorinitialisierte Kapazität, um Fahrer und Mitarbeiter einsatzbereit zu halten, und legen Sie gemeinsame Laufzeit- und Überwachungskonfigurationen auf Anwendungsebene fest. Auf den folgenden Seiten wird beschrieben, wie Sie Anwendungen konfigurieren, wenn Sie EMR Serverless verwenden.

Themen

- Grundlegendes zum Anwendungsverhalten in EMR Serverless
- Vorinitialisierte Kapazität für die Arbeit mit einer Anwendung in EMR Serverless
- Standardanwendungskonfiguration für EMR Serverless

Grundlegendes zum Anwendungsverhalten in EMR Serverless

In diesem Abschnitt werden das Verhalten bei der Auftragsübermittlung, die Kapazitätskonfiguration für die Skalierung und die Worker-Konfigurationseinstellungen für EMR Serverless beschrieben.

Standardverhalten von Anwendungen

Autostart — Eine Anwendung ist standardmäßig so konfiguriert, dass sie bei der Auftragserteilung automatisch gestartet wird. Sie können diese Funktion ausschalten.

Auto-Stop — Eine Anwendung ist standardmäßig so konfiguriert, dass sie automatisch stoppt, wenn sie 15 Minuten lang inaktiv ist. Wenn eine Anwendung in den `STOPPED` Status wechselt, gibt sie alle konfigurierten vorinitialisierten Kapazitäten frei. Sie können die Dauer der Leerlaufzeit ändern, bevor eine Anwendung automatisch beendet wird, oder Sie können diese Funktion deaktivieren.

Maximale Kapazität

Sie können die maximale Kapazität konfigurieren, auf die eine Anwendung skaliert werden kann. Sie können Ihre maximale Kapazität in Bezug auf CPU, Arbeitsspeicher (GB) und Festplatte (GB) angeben.

> **Note**
>
> Es empfiehlt sich, Ihre maximale Kapazität so zu konfigurieren, dass sie proportional zu Ihrer unterstützten Mitarbeitergröße ist, indem Sie die Anzahl der Mitarbeiter mit ihrer Größe multiplizieren. Wenn Sie Ihre Anwendung beispielsweise auf 50 Worker mit 2 VCPUs, 16 GB Arbeitsspeicher und 20 GB Festplatte beschränken möchten, legen Sie Ihre maximale Kapazität auf 100 VCPUs, 800 GB für Arbeitsspeicher und 1000 GB für Festplatte fest.

Unterstützte Worker-Konfigurationen

In der folgenden Tabelle sind die unterstützten Worker-Konfigurationen und -Größen aufgeführt, die für EMR Serverless angegeben werden können. Konfigurieren Sie je nach den Anforderungen Ihrer Arbeitslast unterschiedliche Größen für Treiber und Executoren.

Konfigurationen und Größen von Workern

CPU	Arbeitsspeicher	Temporärer Standardspeicher
1 vCPU	Mindestens 2 GB, maximal 8 GB, in Schritten von 1 GB	20 GB — 200 GB
2 vCPU	Mindestens 4 GB, maximal 16 GB, in Schritten von 1 GB	20 GB — 200 GB
4 vCPU	Mindestens 8 GB, maximal 30 GB, in Schritten von 1 GB	20 GB — 200 GB
8 vCPU	Mindestens 16 GB, maximal 60 GB, in Schritten von 4 GB	20 GB — 200 GB
16 vCPU	Mindestens 32 GB, maximal 120 GB, in Schritten von 8 GB	20 GB — 200 GB

CPU — Jeder Worker kann 1, 2, 4, 8 oder 16 V habenCPUs.

Arbeitsspeicher — Jeder Worker verfügt über Arbeitsspeicher, der in GB angegeben wird und innerhalb der in der vorherigen Tabelle aufgeführten Grenzwerte liegt. Spark-Jobs haben einen Speicheraufwand, was bedeutet, dass der Speicherplatz, den sie verwenden, die angegebenen Containergrößen übersteigt. Dieser Overhead wird mit den Eigenschaften `spark.driver.memory0verhead` und angegeben`spark.executor.memory0verhead`. Der Overhead hat einen Standardwert von 10% des Container-Speichers mit einem Minimum von 384 MB. Sie sollten diesen Mehraufwand berücksichtigen, wenn Sie die Größe der Mitarbeiter wählen.

Wenn Sie beispielsweise 4 V CPUs für Ihre Worker-Instance und eine vorinitialisierte Speicherkapazität von 30 GB wählen, legen Sie einen Wert von etwa 27 GB als Executor-Speicher für Ihren Spark-Job fest. Dadurch wird die Nutzung Ihrer vorinitialisierten Kapazität maximiert. Der nutzbare Arbeitsspeicher beträgt 27 GB plus 10% von 27 GB (2,7 GB), also insgesamt 29,7 GB.

Festplatte — Sie können jeden Worker mit temporären Speicherfestplatten mit einer Mindestgröße von 20 GB und einer Höchstgröße von 200 GB konfigurieren. Sie zahlen nur für zusätzlichen Speicherplatz über 20 GB, den Sie pro Mitarbeiter konfigurieren.

Vorinitialisierte Kapazität für die Arbeit mit einer Anwendung in EMR Serverless

EMR Serverless bietet eine optionale Funktion, mit der Fahrer und Mitarbeiter vorinitialisiert sind und innerhalb von Sekunden einsatzbereit sind. Dadurch wird effektiv ein warmer Pool von Mitarbeitern für eine Anwendung geschaffen. Diese Funktion wird als vorinitialisierte Kapazität bezeichnet. Um diese Funktion zu konfigurieren, setzen Sie den `initialCapacity` Parameter einer Anwendung auf die Anzahl der Worker, die Sie vorab initialisieren möchten. Bei vorinitialisierter Arbeitskapazität werden Jobs sofort gestartet. Dies ist ideal, wenn Sie iterative Anwendungen und zeitkritische Jobs implementieren möchten.

Durch die vorinitialisierte Kapazität steht ein warmer Pool von Mitarbeitern bereit, sodass Jobs und Sitzungen innerhalb von Sekunden gestartet werden können. Sie zahlen für bereitgestellte vorinitialisierte Worker auch dann, wenn sich die Anwendung im Leerlauf befindet. Wir empfehlen daher, sie für Anwendungsfälle zu aktivieren, die von der schnellen Startzeit profitieren, und sie so zu dimensionieren, dass die Ressourcen optimal genutzt werden. Serverlose EMR-Anwendungen werden automatisch heruntergefahren, wenn sie inaktiv sind. Wir empfehlen, diese Funktion aktiviert zu lassen, wenn Sie vorinitialisierte Worker verwenden, um unerwartete Gebühren zu vermeiden.

Wenn Sie einen Job weiterleiten und Mitarbeiter von verfügbar `initialCapacity` sind, verwendet der Job diese Ressourcen, um seine Ausführung zu starten. Wenn diese Arbeitskräfte bereits für andere Jobs verwendet werden oder wenn für die Stelle mehr Ressourcen benötigt werden, als verfügbar sind`initialCapacity`, werden in der Anwendung zusätzliche Arbeitskräfte angefordert und eingestellt, und zwar bis zu den für die Bewerbung festgelegten Höchstgrenzen für Ressourcen. Wenn die Ausführung eines Auftrags beendet ist, werden die von ihm verwendeten Worker wieder freigegeben, und die Anzahl der für die Anwendung verfügbaren Ressourcen wird wieder erreicht`initialCapacity`. Eine Anwendung behält die `initialCapacity` Ressourcen auch dann bei, wenn die Ausführung der Jobs abgeschlossen ist. Die Anwendung gibt überschüssige Ressourcen ab dem `initialCapacity` Zeitpunkt frei, zu dem sie für die Ausführung der Jobs nicht mehr benötigt werden.

Vorinitialisierte Kapazität ist verfügbar und einsatzbereit, sobald die Anwendung gestartet wurde. Die vorinitialisierte Kapazität wird inaktiv, wenn die Anwendung gestoppt wird. Eine Anwendung wechselt nur dann in den `STARTED` Status, wenn die angeforderte vorinitialisierte Kapazität erstellt wurde und einsatzbereit ist. Während der gesamten Zeit, in der sich die Anwendung im `STARTED` Status befindet, hält EMR Serverless die vorinitialisierte Kapazität für die Nutzung oder Nutzung durch Jobs oder interaktive Workloads verfügbar. Die Funktion stellt die Kapazität für freigegebene oder ausgefallene Container wieder her. Dadurch wird die Anzahl der Mitarbeiter beibehalten, die der `InitialCapacity` Parameter angibt. Der Status einer Anwendung ohne vorinitialisierte Kapazität kann sofort von `CREATED` zu geändert werden. `STARTED`

Sie können die Anwendung so konfigurieren, dass vorinitialisierte Kapazität freigegeben wird, wenn sie für einen bestimmten Zeitraum nicht verwendet wird. Die Standardeinstellung ist 15 Minuten. Eine gestoppte Bewerbung wird automatisch gestartet, wenn Sie einen neuen Job einreichen. Sie können diese automatischen Start- und Stoppkonfigurationen festlegen, wenn Sie die Anwendung erstellen, oder sie ändern, wenn sich die Anwendung im `STOPPED` Status `CREATED` oder befindet.

Sie können die `InitialCapacity` Anzahl ändern und Rechenkonfigurationen wie CPU, Arbeitsspeicher und Festplatte für jeden Worker angeben. Da Sie keine teilweisen Änderungen vornehmen können, geben Sie alle Rechenkonfigurationen an, wenn Sie Werte ändern. Sie können Konfigurationen nur ändern, wenn sich die Anwendung im `STOPPED` Status `CREATED` oder befindet.

> **Note**
>
> Um die Nutzung der Ressourcen durch Ihre Anwendung zu optimieren, empfehlen wir, Ihre Containergrößen an die Größe Ihrer vorinitialisierten Capacity-Worker anzupassen. Wenn Sie beispielsweise die Größe Ihres Spark-Executors auf 2 CPUs und Ihren Arbeitsspeicher

auf 8 GB konfigurieren, Ihre vorinitialisierte Worker-Größe jedoch 4 CPUs mit 16 GB Arbeitsspeicher beträgt, dann nutzen die Spark-Executoren nur die Hälfte der Ressourcen der Mitarbeiter, wenn sie diesem Job zugewiesen werden.

Anpassen der vorinitialisierten Kapazität für Spark und Hive

Sie können die vorinitialisierte Kapazität für Workloads, die auf bestimmten Big-Data-Frameworks ausgeführt werden, weiter anpassen. Wenn ein Workload beispielsweise auf Apache Spark ausgeführt wird, geben Sie an, wie viele Worker als Treiber und wie viele als Executoren starten. Wenn Sie Apache Hive verwenden, geben Sie auf ähnliche Weise an, wie viele Worker als Hive-Treiber starten und wie viele Tez-Aufgaben ausführen sollen.

Konfiguration einer Anwendung, auf der Apache Hive mit vorinitialisierter Kapazität ausgeführt wird

Die folgende API-Anfrage erstellt eine Anwendung, auf der Apache Hive ausgeführt wird, die auf der Amazon EMR-Version emr-6.6.0 basiert. Die Anwendung beginnt mit 5 vorinitialisierten Hive-Treibern mit jeweils 2 vCPUs und 4 GB Arbeitsspeicher und 50 vorinitialisierten Tez-Task-Workern mit jeweils 4 vCPUs und 8 GB Arbeitsspeicher. Wenn Hive-Abfragen in dieser Anwendung ausgeführt werden, verwenden sie zunächst die vorinitialisierten Worker und beginnen sofort mit der Ausführung. Wenn alle vorinitialisierten Worker ausgelastet sind und mehr Hive-Jobs eingereicht werden, kann die Anwendung auf insgesamt 400 vCPU und 1024 GB Arbeitsspeicher skaliert werden. Sie können optional die Kapazität für den oder den Worker weglassen. `DRIVER` `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"
}'
```

Konfiguration einer Anwendung, auf der Apache Spark mit vorinitialisierter Kapazität ausgeführt wird

Die folgende API-Anfrage erstellt eine Anwendung, die Apache Spark 3.2.0 auf Basis von Amazon EMR Version 6.6.0 ausführt. Die Anwendung beginnt mit 5 vorinitialisierten Spark-Treibern mit jeweils 2 vCPUs und 4 GB Arbeitsspeicher und 50 vorinitialisierten Executoren mit jeweils 4 vCPUs und 8 GB Arbeitsspeicher. Wenn Spark-Jobs in dieser Anwendung ausgeführt werden, verwenden sie zunächst die vorinitialisierten Worker und beginnen sofort mit der Ausführung. Wenn alle vorinitialisierten Worker ausgelastet sind und mehr Spark-Jobs eingereicht werden, kann die Anwendung auf insgesamt 400 vCPU und 1024 GB Arbeitsspeicher skaliert werden. Sie können optional die Kapazität für entweder den oder den weglassen. `DRIVER` `EXECUTOR`

> **Note**
>
> Spark fügt dem für Treiber und Executoren angeforderten Speicher einen konfigurierbaren Speicheraufwand mit einem Standardwert von 10% hinzu. Damit Jobs vorinitialisierte Worker verwenden, sollte die anfängliche Speicherkonfiguration für die Kapazität größer sein als der Arbeitsspeicher, den der Job und der Overhead anfordern.

```
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"
}'
```

Standardanwendungskonfiguration für EMR Serverless

Sie können auf Anwendungsebene gemeinsame Laufzeit- und Überwachungskonfigurationen für alle Jobs angeben, die Sie im Rahmen derselben Anwendung einreichen. Dadurch wird der zusätzliche Aufwand reduziert, der mit der Notwendigkeit verbunden ist, für jeden Job dieselben Konfigurationen einzureichen.

Sie können die Konfigurationen zu den folgenden Zeitpunkten ändern:

- Deklarieren Sie Konfigurationen auf Anwendungsebene bei der Auftragserteilung.
- Überschreiben Sie die Standardkonfigurationen während der Auftragsausführung.

Die folgenden Abschnitte enthalten weitere Informationen und ein Beispiel für weiteren Kontext.

Deklarieren von Konfigurationen auf Anwendungsebene

Sie können Protokollierungs- und Laufzeitkonfigurationseigenschaften auf Anwendungsebene für die Jobs angeben, die Sie im Rahmen der Anwendung einreichen.

`monitoringConfiguration`

Verwenden Sie das Feld, um die Protokollkonfigurationen für Jobs anzugeben, die Sie mit der Anwendung einreichen. `monitoringConfiguration` Weitere Informationen zur Protokollierung für EMR Serverless finden Sie unter. Speichern von Protokollen

`runtimeConfiguration`

Um Eigenschaften der Laufzeitkonfiguration anzugeben, geben Sie z. `spark-defaults` B. ein Konfigurationsobjekt in das `runtimeConfiguration` Feld ein. Dies wirkt sich auf die Standardkonfigurationen für alle Jobs aus, die Sie mit der Anwendung einreichen. Weitere

Informationen finden Sie unter Parameter zum Überschreiben der Hive-Konfiguration und Parameter zum Überschreiben der Spark-Konfiguration.

Die verfügbaren Konfigurationsklassifizierungen variieren je nach EMR Serverless-Version. Zum Beispiel Klassifizierungen für benutzerdefiniertes Log4j `spark-driver-log4j2` und `spark-executor-log4j2` sind nur mit Versionen 6.8.0 und höher verfügbar. Eine Liste der anwendungsspezifischen Eigenschaften finden Sie unter und. Eigenschaften von Spark-Jobs Eigenschaften von Hive-Jobs

Sie können Apache Log4j2-Eigenschaften AWS Secrets Manager für den Datenschutz und die Java 17-Laufzeit auch auf Anwendungsebene konfigurieren.

Um Secrets Manager Manager-Geheimnisse auf Anwendungsebene weiterzugeben, fügen Sie Benutzern und Rollen, die EMR Serverless-Anwendungen mit Geheimnissen erstellen oder aktualisieren müssen, die folgende Richtlinie hinzu.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "SecretsManagerPolicy",
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue",
        "secretsmanager:DescribeSecret"
      ],
      "Resource": [
        "arn:aws:secretsmanager:us-east-1:123456789012:secret:my-secret-
name-123abc"
      ]
    },
    {
      "Sid": "KMSDecryptPolicy",
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt"
      ],
      "Resource": [
        "arn:aws:kms:us-
east-1:123456789012:key/12345678-1234-1234-1234-123456789012"
      ]
```

```
    }
  ]
}
```

Weitere Informationen zum Erstellen benutzerdefinierter Richtlinien für geheime Daten finden Sie AWS Secrets Manager im AWS Secrets Manager Benutzerhandbuch unter Beispiele für Berechtigungsrichtlinien.

> **Note**
>
> Die `runtimeConfiguration`, die Sie auf Anwendungsebene angeben, `applicationConfiguration` entspricht der `StartJobRun`API.

Beispiel einer Deklaration

Das folgende Beispiel zeigt, wie Standardkonfigurationen mit deklariert `create-application` werden.

```
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://amzn-s3-demo-logging-bucket/logs/app-level"
        },
        "managedPersistenceMonitoringConfiguration": {
            "enabled": false
        }
    }'
```

Überschreiben von Konfigurationen während einer Jobausführung

Mit der `StartJobRun`API können Sie Konfigurationsüberschreibungen für die Anwendungskonfiguration und die Überwachungskonfiguration angeben. EMR Serverless führt dann die Konfigurationen zusammen, die Sie auf Anwendungs- und Jobebene angeben, um die Konfigurationen für die Jobausführung zu bestimmen.

Die Granularitätsstufe bei der Zusammenführung ist wie folgt:

- **`ApplicationConfiguration`**- Zum Beispiel `spark-defaults` der Klassifizierungstyp.
- **`MonitoringConfiguration`**- Konfigurationstyp, zum Beispiel`s3MonitoringConfiguration`.

> **Note**
>
> Die Priorität der Konfigurationen, die Sie auf Anwendungsebene bereitstellen, `StartJobRun`hat Vorrang vor den Konfigurationen, die Sie auf Anwendungsebene bereitstellen.

Weitere Informationen zur Rangfolge der Prioritäten finden Sie unter und. Parameter zum Überschreiben der Hive-Konfiguration Parameter zum Überschreiben der Spark-Konfiguration

Wenn Sie einen Job starten und keine bestimmte Konfiguration angeben, wird diese von der Anwendung übernommen. Wenn Sie die Konfigurationen auf Jobebene deklarieren, können Sie die folgenden Operationen ausführen:

- Eine bestehende Konfiguration überschreiben — Geben Sie denselben Konfigurationsparameter in der `StartJobRun` Anfrage mit Ihren Override-Werten an.
- Zusätzliche Konfiguration hinzufügen — Fügen Sie der `StartJobRun` Anfrage den neuen Konfigurationsparameter mit den Werten hinzu, die Sie angeben möchten.
- Eine bestehende Konfiguration entfernen — Um eine Laufzeitkonfiguration einer Anwendung zu entfernen, geben Sie den Schlüssel für die Konfiguration ein, die Sie entfernen möchten, und übergeben Sie eine leere Deklaration `{}` für die Konfiguration. Es wird nicht empfohlen, Klassifizierungen zu entfernen, die Parameter enthalten, die für die Ausführung eines Jobs erforderlich sind. Wenn Sie beispielsweise versuchen, die erforderlichen Eigenschaften für einen Hive-Job zu entfernen, schlägt der Job fehl.

  Um eine Konfiguration zur Anwendungsüberwachung zu entfernen, verwenden Sie die entsprechende Methode für den entsprechenden Konfigurationstyp:

  - **`cloudWatchLoggingConfiguration`**— Um zu entfernen`cloudWatchLogging`, übergeben Sie das Kennzeichen enabled als`false`.
  - **`managedPersistenceMonitoringConfiguration`**— Um die verwalteten Persistenzeinstellungen zu entfernen und zum Standardstatus aktiviert zurückzukehren, übergeben Sie eine leere Deklaration `{}` für die Konfiguration.
  - **`s3MonitoringConfiguration`**— Um zu entfernen`s3MonitoringConfiguration`, übergeben Sie eine leere Deklaration `{}` für die Konfiguration.

Beispiel für Override

Das folgende Beispiel zeigt verschiedene Operationen, die Sie bei der Einreichung eines Jobs unter ausführen können`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://amzn-s3-demo-destination-bucket1/
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
```

```
            }
        }
    }'
```

Zum Zeitpunkt der Auftragsausführung gelten die folgenden Klassifizierungen und Konfigurationen, die auf der unter Parameter zum Überschreiben der Hive-Konfiguration und Parameter zum Überschreiben der Spark-Konfiguration beschriebenen Rangfolge der Prioritätsüberschreibung basieren.

- Die Klassifizierung `spark-defaults` wird mit den auf Auftragsebene angegebenen Eigenschaften aktualisiert. Für diese Klassifizierung werden nur `StartJobRun` die in enthaltenen Eigenschaften berücksichtigt.
- Die Klassifizierung `spark-executor-log4j2` wird in die bestehende Liste der Klassifizierungen aufgenommen.
- Die Klassifizierung `spark-driver-log4j2` wird entfernt.
- Die Konfigurationen für `managedPersistenceMonitoringConfiguration` werden mit Konfigurationen auf Auftragsebene aktualisiert.
- Die Konfigurationen für `s3MonitoringConfiguration` werden entfernt.
- Die Konfigurationen für `cloudWatchLoggingConfiguration` werden zu bestehenden Überwachungskonfigurationen hinzugefügt.

Anpassen eines serverlosen EMR-Images

Verwenden Sie ab Amazon EMR 6.9.0 benutzerdefinierte Images, um Anwendungsabhängigkeiten und Laufzeitumgebungen mit Amazon EMR Serverless in einem einzigen Container zu verpacken. Dies vereinfacht die Verwaltung von Workload-Abhängigkeiten und macht Ihre Pakete portabler. Wenn Sie Ihr EMR Serverless-Image anpassen, bietet es die folgenden Vorteile:

- Installiert und konfiguriert Pakete, die für Ihre Workloads optimiert sind. Diese Pakete sind in der öffentlichen Distribution von Amazon EMR-Laufzeitumgebungen nicht allgemein verfügbar.
- Integriert EMR Serverless in die derzeit etablierten Build-, Test- und Bereitstellungsprozesse in Ihrem Unternehmen, einschließlich lokaler Entwicklung und Tests.
- Wendet etablierte Sicherheitsprozesse an, wie z. B. das Scannen von Bildern, die die Compliance- und Governance-Anforderungen in Ihrem Unternehmen erfüllen.
- Ermöglicht es Ihnen, Ihre eigenen Versionen von JDK und Python für Ihre Anwendungen zu verwenden.

EMR Serverless stellt Images bereit, die Sie als Basis verwenden, wenn Sie Ihre eigenen Images erstellen. Das Basis-Image enthält die wesentlichen JAR-Dateien, Konfigurationen und Bibliotheken für die Interaktion des Images mit EMR Serverless. Sie finden das Basisbild in der Amazon ECR Public Gallery. Verwenden Sie das Image, das Ihrem Anwendungstyp (Spark oder Hive) und Ihrer Release-Version entspricht. Wenn Sie beispielsweise eine Anwendung auf Amazon EMR Version 6.9.0 erstellen, verwenden Sie die folgenden Bilder.

Typ	Image
Spark	`public.ecr.aws/emr-serverless/spark/emr-6.9.0:latest`
Hive	`public.ecr.aws/emr-serverless/hive/emr-6.9.0:latest`

Voraussetzungen

Bevor Sie ein benutzerdefiniertes EMR Serverless-Image erstellen, müssen Sie diese Voraussetzungen erfüllen.

1. Erstellen Sie ein Amazon ECR-Repository in demselben Verzeichnis AWS-Region , das Sie zum Starten von EMR Serverless-Anwendungen verwenden. Informationen zum Erstellen eines privaten Amazon ECR-Repositorys finden Sie unter Privates Repository erstellen.
2. Um Benutzern Zugriff auf Ihr Amazon ECR-Repository zu gewähren, fügen Sie Benutzern und Rollen, die EMR Serverless-Anwendungen mit Bildern aus diesem Repository erstellen oder aktualisieren, die folgenden Richtlinien hinzu.

   JSON

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

```
      ],
      "Resource": [
        "arn:aws:ecr:*:123456789012:repository/my-repo"
      ]
    }
  ]
}
```

Weitere Beispiele für identitätsbasierte Amazon ECR-Richtlinien finden Sie unter Beispiele für identitätsbasierte Richtlinien von Amazon Elastic Container Registry.

Schritt 1: Erstellen Sie ein benutzerdefiniertes Image aus EMR Serverless-Basisimages

Erstellen Sie zunächst ein Dockerfile, das mit einer `FROM` Anweisung beginnt, die Ihr bevorzugtes Basis-Image verwendet. Fügen Sie nach der `FROM` Anweisung alle Änderungen hinzu, die Sie am Image vornehmen möchten. Das Basis-Image legt automatisch den Wert `USER` auf fest`hadoop`. Diese Einstellung verfügt nicht über Berechtigungen für alle Änderungen, die Sie vornehmen. Um das Problem zu umgehen, setzen Sie den `USER` Wert auf`root`, ändern Sie Ihr Bild und setzen Sie dann den Wert `USER` Zurück auf`hadoop:hadoop`. Beispiele für gängige Anwendungsfälle finden Sie unter. Verwenden von benutzerdefinierten Images mit EMR Serverless

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

USER root
# MODIFICATIONS GO HERE

# EMRS runs the image as hadoop
USER hadoop:hadoop
```

Nachdem Sie das Dockerfile haben, erstellen Sie das Image mit dem folgenden Befehl.

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

Schritt 2: Überprüfen Sie das Image lokal

EMR Serverless bietet ein Offline-Tool, mit dem Sie Ihr benutzerdefiniertes Image statisch überprüfen können, um grundlegende Dateien, Umgebungsvariablen und korrekte Image-Konfigurationen zu validieren. Informationen zur Installation und Ausführung des Tools finden Sie in der Amazon EMR Serverless Image CLI. GitHub

Führen Sie nach der Installation des Tools den folgenden Befehl aus, um ein Image zu validieren:

```
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
```

Die Ausgabe sieht wie folgt aus.

```
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.
----------------------------------------------------------------
```

Schritt 3: Laden Sie das Bild in Ihr Amazon ECR-Repository hoch

Übertragen Sie Ihr Amazon ECR-Image mit den folgenden Befehlen in Ihr Amazon ECR-Repository. Stellen Sie sicher, dass Sie über die richtigen IAM-Berechtigungen verfügen, um das Image in Ihr Repository zu übertragen. Weitere Informationen finden Sie unter Pushing an Image im Amazon ECR-Benutzerhandbuch.

```
# 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
```

Schritt 4: Erstellen oder aktualisieren Sie eine Anwendung mit benutzerdefinierten Bildern

Wählen Sie die AWS-Managementkonsole Registerkarte oder AWS CLI Registerkarte aus, je nachdem, wie Sie Ihre Anwendung starten möchten, und führen Sie dann die folgenden Schritte aus.

Console

1. Melden Sie sich bei der EMR Studio-Konsole unter https://console.aws.amazon.com/emr an. Navigieren Sie zu Ihrer Anwendung, oder erstellen Sie mithilfe der Anweisungen unter Anwendung erstellen eine neue Anwendung.
2. Um benutzerdefinierte Images anzugeben, wenn Sie eine EMR Serverless-Anwendung erstellen oder aktualisieren, wählen Sie in den Einrichtungsoptionen der Anwendung die Option Benutzerdefinierte Einstellungen aus.
3. Aktivieren Sie im Abschnitt Benutzerdefinierte Image-Einstellungen das Kontrollkästchen Benutzerdefiniertes Image mit dieser Anwendung verwenden.
4. Fügen Sie die Amazon ECR-Image-URI in das Feld Image-URI ein. EMR Serverless verwendet dieses Image für alle Worker-Typen der Anwendung. Alternativ können Sie Verschiedene benutzerdefinierte Bilder auswählen und URIs für jeden Arbeitertyp ein anderes Amazon ECR-Bild einfügen.

CLI

- Erstellen Sie eine Anwendung mit dem `image-configuration` Parameter. EMR Serverless wendet diese Einstellung auf alle Worker-Typen an.

```
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"
}'
```

Verwenden Sie den Parameter, um eine Anwendung mit unterschiedlichen Image-Einstellungen für jeden Worker-Typ zu erstellen. `worker-type-specifications`

```
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"
        }
    }
}'
```

Verwenden Sie den `image-configuration` Parameter, um eine Anwendung zu aktualisieren. EMR Serverless wendet diese Einstellung auf alle Worker-Typen an.

```
aws emr-serverless update-application \
--application-id application-id \
--image-configuration '{
```

```
    "imageUri": "aws-account-id.dkr.ecr.region.amazonaws.com/my-repository:tag/
@digest"
}'
```

Schritt 5: Erlauben Sie EMR Serverless, auf das benutzerdefinierte Image-Repository zuzugreifen

Fügen Sie dem Amazon ECR-Repository die folgende Ressourcenrichtlinie hinzu, damit der EMR Serverless Service Principal die `getdescribe`, und `download` -Anfragen aus diesem Repository verwenden kann.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "EmrServerlessCustomImageSupport",
      "Effect": "Allow",
      "Action": [
        "ecr:BatchGetImage",
        "ecr:DescribeImages",
        "ecr:GetDownloadUrlForLayer"
      ],
      "Resource": "arn:aws:ecr:*:123456789012:repository/my-repo",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:emr-serverless:*:123456789012:/applications/
*"
        }
      }
    }
  ]
}
```

Aus Sicherheitsgründen sollten Sie der Repository-Richtlinie einen `aws:SourceArn` Bedingungsschlüssel hinzufügen. Der globale IAM-Bedingungsschlüssel `aws:SourceArn` stellt sicher, dass EMR Serverless das Repository nur für einen Anwendungs-ARN verwendet. Weitere

Informationen zu den Amazon ECR-Repository-Richtlinien finden Sie unter Erstellen eines privaten Repositorys.

Überlegungen und Einschränkungen

Wenn Sie mit benutzerdefinierten Images arbeiten, sollten Sie Folgendes beachten:

- Verwenden Sie das richtige Basis-Image, das dem Typ (Spark oder Hive) und dem Release-Label (z. B.`emr-6.9.0`) für Ihre Anwendung entspricht.
- EMR Serverless ignoriert unsere `[CMD]` `[ENTRYPOINT]` Anweisungen in der Docker-Datei. Verwenden Sie allgemeine Anweisungen in der Docker-Datei, z. B., und. `[COPY]` `[RUN]` `[WORKDIR]`
- Ändern Sie die Umgebungsvariablen`JAVA_HOME`,, nicht `SPARK_HOME``HIVE_HOME`, `TEZ_HOME` wenn Sie ein benutzerdefiniertes Image erstellen.
- Benutzerdefinierte Bilder dürfen eine Größe von 10 GB nicht überschreiten.
- Wenn Sie Binärdateien oder JAR-Dateien in den Amazon EMR-Basis-Images ändern, kann dies zu Fehlern beim Starten von Anwendungen oder Jobs führen.
- Das Amazon ECR-Repository muss sich in demselben Verzeichnis befinden AWS-Region , das Sie zum Starten von EMR Serverless-Anwendungen verwenden.

Konfiguration des VPC-Zugriffs für serverlose EMR-Anwendungen zur Verbindung mit Daten

Sie können serverlose EMR-Anwendungen so konfigurieren, dass sie eine Verbindung zu Ihren Datenspeichern in Ihrer VPC herstellen, z. B. Amazon Redshift Redshift-Cluster, Amazon RDS-Datenbanken oder Amazon S3 S3-Buckets mit VPC-Endpunkten. Ihre EMR Serverless-Anwendung verfügt über ausgehende Konnektivität zu den Datenspeichern in Ihrer VPC. Standardmäßig blockiert EMR Serverless sowohl den eingehenden Zugriff auf Ihre Anwendungen als auch den ausgehenden Internetzugang, um die Sicherheit zu erhöhen.

> **Note**
>
> Sie müssen den VPC-Zugriff konfigurieren, wenn Sie eine externe Hive-Metastore-Datenbank für Ihre Anwendung verwenden möchten. Informationen zur Konfiguration eines externen Hive-Metastores finden Sie unter Metastore-Konfiguration.

Erstellen von Anwendungen

Wählen Sie auf der Seite Anwendung erstellen benutzerdefinierte Einstellungen aus und geben Sie die VPC, Subnetze und Sicherheitsgruppen an, die EMR Serverless-Anwendungen verwenden können.

VPCs

Wählen Sie den Namen der Virtual Private Cloud (VPC), die Ihre Datenspeicher enthält. Auf der Seite „Anwendung erstellen“ werden alle VPCs für Sie ausgewählten AWS-Region Anwendungen aufgeführt.

Subnets

Wählen Sie die Subnetze innerhalb der VPC aus, die Ihren Datenspeicher enthält. Auf der Seite Anwendung erstellen werden alle Subnetze für die Datenspeicher in Ihrer VPC aufgeführt. Sowohl öffentliche als auch private Subnetze werden unterstützt. Sie können entweder private oder öffentliche Subnetze an Ihre Anwendungen übergeben. Bei der Entscheidung, ob ein öffentliches oder ein privates Subnetz eingerichtet werden soll, sind einige Überlegungen zu beachten.

Für private Subnetze:

- Die zugehörigen Routing-Tabellen dürfen keine Internet-Gateways haben.
- Für ausgehende Verbindungen zum Internet konfigurieren Sie bei Bedarf ausgehende Routen mithilfe eines NAT-Gateways. Informationen zur Konfiguration eines NAT-Gateways finden Sie unter NAT-Gateways.
- Für Amazon S3 S3-Konnektivität konfigurieren Sie entweder ein NAT-Gateway oder einen VPC-Endpunkt. Informationen zur Konfiguration eines S3-VPC-Endpunkts finden Sie unter Gateway-Endpunkt erstellen.
- Wenn Sie einen S3-VPC-Endpunkt konfigurieren und eine Endpunktrichtlinie zur Zugriffskontrolle anhängen, folgen Sie den Anweisungen unter Logging for EMR Serverless with managed storage, um EMR Serverless Berechtigungen zum Speichern und Bereitstellen von Anwendungsprotokollen zu erteilen.
- Für Konnektivität zu anderen Geräten AWS-Services außerhalb der VPC, z. B. zu Amazon DynamoDB, konfigurieren Sie entweder VPC-Endpunkte oder ein NAT-Gateway. Informationen zur Konfiguration von VPC-Endpunkten finden Sie AWS-Services unter Arbeiten mit VPC-Endpunkten.

> **Note**
>
> Wenn Sie eine Amazon EMR Serverless-Anwendung in einem privaten Subnetz einrichten, empfehlen wir Ihnen, auch VPC-Endpunkte für Amazon S3 einzurichten. Wenn sich Ihre EMR Serverless-Anwendung in einem privaten Subnetz ohne VPC-Endpunkte für Amazon S3 befindet, fallen zusätzliche NAT-Gateway-Gebühren an, die mit dem S3-Verkehr verbunden sind. Dies liegt daran, dass der Datenverkehr zwischen Ihrer EMR-Anwendung und Amazon S3 nicht innerhalb Ihrer VPC verbleibt, wenn VPC-Endpunkte nicht konfiguriert sind.

Für öffentliche Subnetze:

- Diese haben eine Route zu einem Internet Gateway.
- Sie müssen sicherstellen, dass die Sicherheitsgruppen ordnungsgemäß konfiguriert sind, um den ausgehenden Verkehr zu kontrollieren.

Mitarbeiter können über ausgehenden Datenverkehr eine Verbindung zu den Datenspeichern in Ihrer VPC herstellen. Standardmäßig blockiert EMR Serverless den eingehenden Zugriff für Mitarbeiter. Dies dient der Verbesserung der Sicherheit.

Wenn Sie es verwenden AWS Config, erstellt EMR Serverless einen elastic network interface Interface-Elementdatensatz für jeden Worker. Um Kosten im Zusammenhang mit dieser Ressource zu vermeiden, sollten Sie die Option deaktivieren`AWS::EC2::NetworkInterface`. AWS Config

> **Note**
>
> Wir empfehlen, dass Sie mehrere Subnetze in mehreren Availability Zones auswählen. Dies liegt daran, dass die von Ihnen ausgewählten Subnetze die Availability Zones bestimmen, die für den Start einer EMR Serverless-Anwendung verfügbar sind. Jeder Worker verwendet eine IP-Adresse in dem Subnetz, in dem er gestartet wird. Bitte stellen Sie sicher, dass die angegebenen Subnetze über ausreichend IP-Adressen für die Anzahl der Worker verfügen, die Sie starten möchten. Weitere Informationen zur Subnetzplanung finden Sie unter. the section called “Bewährte Methoden für die Subnetzplanung”

Überlegungen und Einschränkungen für Subnetze

- EMR Serverless mit öffentlichen Subnetzen unterstützt AWS Lake Formation nicht.

- Eingehender Datenverkehr wird für öffentliche Subnetze nicht unterstützt.

Sicherheitsgruppen

Wählen Sie eine oder mehrere Sicherheitsgruppen aus, die mit Ihren Datenspeichern kommunizieren können. Auf der Seite Anwendung erstellen werden alle Sicherheitsgruppen in Ihrer VPC aufgeführt. EMR Serverless verknüpft diese Sicherheitsgruppen mit elastischen Netzwerkschnittstellen, die an Ihre VPC-Subnetze angeschlossen sind.

> **Note**
>
> Wir empfehlen, dass Sie eine separate Sicherheitsgruppe für EMR Serverless-Anwendungen erstellen. Mit EMR Serverless können Sie keine Anwendung erstellen/aktualisieren/starten, wenn Sicherheitsgruppen Ports für das öffentliche Internet im Bereich 0.0.0.0/0 oder: :/0 haben. Dies bietet mehr Sicherheit und Isolierung und macht die Verwaltung von Netzwerkregeln effizienter. Dadurch wird beispielsweise unerwarteter Datenverkehr an Mitarbeiter mit öffentlichen IP-Adressen blockiert. Um beispielsweise mit Amazon Redshift Redshift-Clustern zu kommunizieren, definieren Sie die Verkehrsregeln zwischen Redshift- und EMR-Serverless-Sicherheitsgruppen, wie im Beispiel im folgenden Abschnitt gezeigt.

Example Beispiel — Kommunikation mit Amazon Redshift Redshift-Clustern

1. Fügen Sie der Amazon Redshift Redshift-Sicherheitsgruppe eine Regel für eingehenden Datenverkehr aus einer der serverlosen EMR-Sicherheitsgruppen hinzu.

Typ	Protocol (Protokoll)	Port-Bereich	Quelle
Alle TCP	TCP	5439	`emr-serverless-security-group`

2. Fügen Sie eine Regel für ausgehenden Datenverkehr von einer der EMR Serverless Security Groups hinzu. Gehen Sie dazu auf eine von zwei Arten vor. Öffnen Sie zunächst den ausgehenden Verkehr zu allen Ports.

Typ	Protocol (Protokoll)	Port-Bereich	Ziel
Gesamter Datenverkehr	TCP	ALL	0.0.0.0/0

Alternativ können Sie den ausgehenden Datenverkehr auf Amazon Redshift Redshift-Cluster beschränken. Dies ist nur nützlich, wenn die Anwendung mit Amazon Redshift Redshift-Clustern kommunizieren muss und sonst nichts.

Typ	Protocol (Protokoll)	Port-Bereich	Quelle
Alle TCP	TCP	5439	`redshift-security-group`

Anwendung konfigurieren

Sie können die Netzwerkkonfiguration für eine bestehende EMR Serverless-Anwendung auf der Seite Anwendung konfigurieren ändern.

Greifen Sie auf Details zur Auftragsausführung zu

Greifen Sie auf der Detailseite der Auftragsausführung auf das Subnetz zu, das von Ihrem Job für einen bestimmten Lauf verwendet wurde. Beachten Sie, dass ein Job nur in einem Subnetz ausgeführt wird, das aus den angegebenen Subnetzen ausgewählt wurde.

Bewährte Methoden für die Subnetzplanung

AWS Ressourcen werden in einem Subnetz erstellt, das eine Teilmenge der verfügbaren IP-Adressen in einer Amazon VPC darstellt. Beispielsweise verfügt eine VPC mit einer /16-Netzmaske über bis zu 65.536 verfügbare IP-Adressen, die mithilfe von Subnetzmasken in mehrere kleinere Netzwerke aufgeteilt werden können. Sie können diesen Bereich beispielsweise in zwei Subnetze aufteilen, von denen jedes die /17-Maske und 32.768 verfügbare IP-Adressen verwendet. Ein Subnetz befindet sich in einer Availability Zone und kann sich nicht zonenübergreifend erstrecken.

Die Subnetze sollten unter Berücksichtigung Ihrer Skalierungsgrenzen für serverlose EMR-Anwendungen entworfen werden. Wenn Sie beispielsweise eine Anwendung haben, die 4 vCPU-Worker anfordert und auf bis zu 4.000 vCPUs skaliert werden kann, benötigt Ihre Anwendung maximal 1.000 Worker für insgesamt 1.000 Netzwerkschnittstellen. Wir empfehlen, Subnetze für mehrere Availability Zones zu erstellen. Auf diese Weise kann EMR Serverless in einem unwahrscheinlichen Fall, wenn eine Availability Zone ausfällt, Ihren Job erneut versuchen oder vorinitialisierte Kapazität in einer anderen Availability Zone bereitstellen. Daher sollte jedes Subnetz in mindestens zwei Availability Zones über mehr als 1.000 verfügbare IP-Adressen verfügen.

Sie benötigen Subnetze mit einer Maskengröße von weniger als oder gleich 22, um 1.000 Netzwerkschnittstellen bereitzustellen. Jede Maske, die größer als 22 ist, erfüllt die Anforderung nicht. Eine Subnetzmaske von /23 bietet beispielsweise 512 IP-Adressen, während eine Maske von /22 1024 und eine Maske von /21 2048 IP-Adressen bereitstellt. Im Folgenden finden Sie ein Beispiel für 4 Subnetze mit /22-Maske in einer VPC mit /16-Netzmaske, die verschiedenen Availability Zones zugewiesen werden können. Es gibt einen Unterschied von fünf zwischen verfügbaren und verwendbaren IP-Adressen, da die ersten vier IP-Adressen und die letzte IP-Adresse in jedem Subnetz von reserviert sind. AWS

Subnetz-ID	Subnetz-Adresse	Subnetzmaske	IP-Adressbereiche	Verfügbare IP-Adressen	Verwendbare IP-Adressen
1	10.0.0.0	255,255,252,0/22	10,0,0,0 - 10,0,3,255	1,024	1.019
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	10,0,8,0 - 10,0,11,255	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

Sie sollten abwägen, ob Ihr Workload am besten für größere Mitarbeiter geeignet ist. Für die Verwendung größerer Mitarbeiter sind weniger Netzwerkschnittstellen erforderlich. Beispielsweise sind für die Verwendung von 16 vCPU-Workern mit einem Anwendungsskalierungslimit von 4.000 vCPUs höchstens 250 Worker für insgesamt 250 verfügbare IP-Adressen für die Bereitstellung von

Netzwerkschnittstellen erforderlich. Sie benötigen Subnetze in mehreren Availability Zones mit einer Maskengröße von weniger als oder gleich 24, um 250 Netzwerkschnittstellen bereitzustellen. Jede Maskengröße von mehr als 24 bietet weniger als 250 IP-Adressen.

Wenn Sie Subnetze für mehrere Anwendungen gemeinsam nutzen, sollte jedes Subnetz so konzipiert werden, dass die kollektiven Skalierungsgrenzen all Ihrer Anwendungen berücksichtigt werden. Wenn Sie beispielsweise 3 Anwendungen haben, die 4 vCPU-Worker anfordern, und jede Anwendung kann auf bis zu 4000 vCPUs mit einem dienstbasierten Kontingent von 12.000 vCPUs skaliert werden, benötigt jedes Subnetz 3000 verfügbare IP-Adressen. Wenn die VPC, die Sie verwenden möchten, nicht über eine ausreichende Anzahl von IP-Adressen verfügt, versuchen Sie, die Anzahl der verfügbaren IP-Adressen zu erhöhen. Das ist möglich, indem Sie zusätzliche CIDR-Blöcke (Classless Inter-Domain Routing) mit Ihrer VPC verbinden. Weitere Informationen finden Sie unter Zusätzliche IPv4 CIDR-Blöcke mit Ihrer VPC verknüpfen im Amazon VPC-Benutzerhandbuch.

Sie können eines der vielen online verfügbaren Tools verwenden, um schnell Subnetzdefinitionen zu generieren und den verfügbaren IP-Adressbereich zu überprüfen.

Optionen für die serverlose Architektur von Amazon EMR

Die Befehlssatzarchitektur Ihrer Amazon EMR Serverless-Anwendung bestimmt die Art der Prozessoren, die die Anwendung zur Ausführung des Jobs verwendet. Amazon EMR bietet zwei Architekturoptionen für Ihre Anwendung: x86_64 und arm64. EMR Serverless aktualisiert automatisch auf die neueste Generation von Instances, sobald diese verfügbar sind, sodass Ihre Anwendungen die neueren Instances ohne zusätzlichen Aufwand verwenden können.

Themen

- Verwendung der x86_64-Architektur
- Verwendung der Arm64-Architektur (Graviton)
- Einführung neuer Anwendungen mit Graviton-Unterstützung
- Konfiguration vorhandener Anwendungen für die Verwendung von Graviton
- Überlegungen zur Verwendung von Graviton

Verwendung der x86_64-Architektur

Die x86_64-Architektur wird auch als x86 64-Bit oder x64 bezeichnet. x86_64 ist die Standardoption für serverlose EMR-Anwendungen. Diese Architektur verwendet x86-basierte Prozessoren und ist mit den meisten Tools und Bibliotheken von Drittanbietern kompatibel.

Die meisten Anwendungen sind mit der x86-Hardwareplattform kompatibel und können erfolgreich auf der x86_64-Standardarchitektur ausgeführt werden. Wenn Ihre Anwendung jedoch mit 64-Bit-ARM kompatibel ist, wechseln Sie zu arm64, um Graviton-Prozessoren zu verwenden, um Leistung, Rechenleistung und Speicher zu verbessern. Es kostet weniger, Instanzen auf einer arm64-Architektur auszuführen, als wenn Sie Instanzen gleicher Größe auf einer x86-Architektur ausführen.

Verwendung der Arm64-Architektur (Graviton)

AWS Graviton-Prozessoren wurden speziell AWS mit 64-Bit-ARM-Neoverse-Kernen entwickelt und nutzen die Arm64-Architektur (auch bekannt als Arch64 oder 64-Bit-ARM). Die AWS auf EMR Serverless verfügbaren Graviton-Prozessoren umfassen Graviton3- und Graviton2-Prozessoren. Diese Prozessoren bieten ein hervorragendes Preis-Leistungs-Verhältnis für Spark- und Hive-Workloads im Vergleich zu vergleichbaren Workloads, die auf der x86_64-Architektur ausgeführt werden. EMR Serverless verwendet automatisch die neueste Generation von Prozessoren, sofern verfügbar, ohne dass Sie ein Upgrade auf die neueste Prozessorgeneration vornehmen müssen.

Einführung neuer Anwendungen mit Graviton-Unterstützung

Verwenden Sie eine der folgenden Methoden, um eine Anwendung zu starten, die die Arm64-Architektur verwendet.

AWS CLI

Um eine Anwendung mit Graviton-Prozessoren von aus zu starten AWS CLI, geben Sie dies `ARM64` als `architecture` Parameter in der `create-application` API an. Geben Sie in den anderen Parametern die entsprechenden Werte für Ihre Anwendung an.

```
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

Um eine Anwendung mit Graviton-Prozessoren von EMR Studio aus zu starten, wählen Sie arm64 als Architekturoption, wenn Sie eine Anwendung erstellen oder aktualisieren.

Konfiguration vorhandener Anwendungen für die Verwendung von Graviton

Sie können Ihre vorhandenen Amazon EMR Serverless-Anwendungen so konfigurieren, dass sie die Graviton-Architektur (arm64) mit dem SDK oder EMR Studio verwenden. AWS CLI

Um eine bestehende Anwendung von x86 nach arm64 zu konvertieren

1. Vergewissern Sie sich, dass Sie die neueste Hauptversion des AWS CLI/SDK verwenden, die den `architecture` Parameter unterstützt.
2. Vergewissern Sie sich, dass keine Jobs ausgeführt werden, und beenden Sie dann die Anwendung.

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

3. Um die Anwendung `ARM64` für die Verwendung von Graviton zu aktualisieren, geben Sie den `architecture` Parameter in der `update-application` API an.

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

4. Verwenden Sie die `get-application` API, um zu überprüfen ARM64, ob die CPU-Architektur der Anwendung aktuell ist.

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

5. Wenn Sie bereit sind, starten Sie die Anwendung neu.

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

Überlegungen zur Verwendung von Graviton

Bevor Sie eine EMR Serverless-Anwendung mit arm64 für Graviton-Unterstützung starten, überprüfen Sie Folgendes.

Bibliothekskompatibilität

Wenn Sie Graviton (arm64) als Architekturoption auswählen, stellen Sie sicher, dass Pakete und Bibliotheken von Drittanbietern mit der 64-Bit-ARM-Architektur kompatibel sind. Informationen zum Packen von Python-Bibliotheken in eine virtuelle Python-Umgebung, die mit Ihrer ausgewählten Architektur kompatibel ist, finden Sie unterVerwenden von Python-Bibliotheken mit EMR Serverless.

Weitere Informationen finden Sie im AWS Graviton Getting Started Repository unter GitHub. Dieses Repository enthält wichtige Ressourcen, die Ihnen beim Einstieg in das ARM-basierte Graviton helfen können.

Parallelität von Aufträgen und Warteschlangen für eine serverlose EMR-Anwendung

Geben Sie ab Amazon EMR Version 7.0.0 und höher das Zeitlimit für die Warteschlange bei der Auftragsausführung und die Konfiguration der Parallelität für Ihre Anwendung an. Wenn Sie diese Konfiguration angeben, stellt Amazon EMR Serverless Ihren Job zunächst in die Warteschlange und beginnt mit der Ausführung auf der Grundlage der Parallelitätsnutzung in Ihrer Anwendung. Wenn die Parallelität Ihrer Auftragsausführung beispielsweise 10 beträgt, werden in Ihrer Anwendung jeweils nur zehn Jobs ausgeführt. Die verbleibenden Jobs werden in die Warteschlange gestellt, bis einer der laufenden Jobs beendet wird. Wenn das Zeitlimit für die Warteschlange früher erreicht wird, wird das Zeitlimit für Ihren Job überschritten. Weitere Informationen finden Sie unter Status der Auftragsausführung.

Hauptvorteile von Parallelität und Warteschlangen

Job-Parallelität und Warteschleifenbildung bieten die folgenden Vorteile, wenn viele Jobeinreichungen erforderlich sind:

- Es hilft dabei, gleichzeitig ausgeführte Jobs zu kontrollieren, um Ihre Kapazitätsgrenzen auf Anwendungsebene effizient zu nutzen.
- Die Warteschlange kann einen plötzlichen Anstieg von Auftragsübermittlungen mit einer konfigurierbaren Timeout-Einstellung enthalten.

Erste Schritte mit Parallelität und Warteschleifenbildung

Die folgenden Verfahren zeigen verschiedene Möglichkeiten, Parallelität und Warteschlangen zu implementieren.

Mit dem AWS CLI

1. Erstellen Sie eine serverlose Amazon EMR-Anwendung mit Warteschlangen-Timeout und gleichzeitigen Auftragsausführungen:

```
aws emr-serverless create-application \
--release-label emr-7.0.0 \
--type SPARK \
--scheduler-configuration '{"maxConcurrentRuns": 1, "queueTimeoutMinutes": 30}'
```

2. Aktualisieren Sie eine Anwendung, um das Zeitlimit für die Job-Warteschlange und die Parallelität zu ändern:

```
aws emr-serverless update-application \
--application-id application-id \
--scheduler-configuration '{"maxConcurrentRuns": 5, "queueTimeoutMinutes": 30}'
```

> **Note**
>
> Sie können Ihre bestehende Anwendung aktualisieren, um Parallelität und Warteschlangen für Jobs zu aktivieren. Dazu muss die Anwendung das Release-Label emr-7.0.0 oder höher haben.

Unter Verwendung des AWS-Managementkonsole

Die folgenden Schritte zeigen, wie Sie mit Job-Parallelität und Warteschlangen beginnen können. Verwenden Sie dazu: AWS-Managementkonsole

1. Gehen Sie zu EMR Studio und wählen Sie, ob Sie eine Anwendung mit dem Release-Label EMR-7.0.0 oder höher erstellen möchten.
2. Wählen Sie unter Optionen zur Anwendungskonfiguration die Option Benutzerdefinierte Einstellungen verwenden aus.

3. Unter Zusätzliche Konfigurationen gibt es einen Abschnitt für Job Run Settings. Wählen Sie die Option Jobparallelität aktivieren, um die Funktion zu aktivieren.
4. Wählen Sie nach der Auswahl Gleichzeitige Auftragsausführungen und Warteschlangen-Timeout aus, um die Anzahl der gleichzeitigen Auftragsausführungen bzw. das Warteschlangen-Timeout zu konfigurieren. Wenn Sie keine Werte für diese Einstellungen eingeben, werden die Standardwerte verwendet.
5. Wählen Sie Anwendung erstellen und die Anwendung wird mit aktivierter Funktion erstellt. Gehen Sie zur Überprüfung zum Dashboard, wählen Sie Ihre Anwendung aus und überprüfen Sie auf der Registerkarte Eigenschaften, ob die Funktion aktiviert ist.

Reichen Sie nach der Konfiguration Jobs mit aktivierter Funktion ein.

Überlegungen zu Parallelität und Warteschleifen

Beachten Sie bei der Implementierung von Parallelität und Warteschlangen die folgenden Punkte:

- Job-Parallelität und Warteschlangen werden in Amazon EMR Version 7.0.0 und höher unterstützt.
- Job-Parallelität und Warteschlangen sind in Amazon EMR Version 7.3.0 und höher standardmäßig aktiviert.
- Sie können die Parallelität für eine Anwendung im Status STARTED nicht aktualisieren.
- Der gültige Bereich für `maxConcurrentRuns` liegt zwischen 1 und 1000 und für `queueTimeoutMinutes` ihn zwischen 15 und 720.
- Für ein Konto können sich maximal 2000 Aufträge im Status QUEUED befinden.
- Parallelität und Warteschleife gelten für Batch- und Streaming-Jobs. Es kann nicht für interaktive Jobs verwendet werden. Weitere Informationen finden Sie unter Ausführen interaktiver Workloads mit EMR Serverless über EMR Studio.

Holen Sie sich Daten mit EMR Serverless in S3 Express One Zone

Mit Amazon EMR-Versionen 7.2.0 und höher können Sie EMR Serverless mit der Amazon S3 Express One Zone-Speicherklasse verwenden, um die Leistung bei der Ausführung von Jobs und Workloads zu verbessern. S3 Express One Zone ist eine leistungsstarke Amazon S3 S3-Speicherklasse mit einer Zone, die für die meisten latenzempfindlichen Anwendungen einen konsistenten Datenzugriff im einstelligen Millisekundenbereich bietet. Zum Zeitpunkt seiner Veröffentlichung bietet S3 Express One Zone den Cloud-Objektspeicher mit der niedrigsten Latenz und der höchsten Leistung in Amazon S3.

Voraussetzungen

- S3 Express One Zone-Berechtigungen — Wenn S3 Express One Zone anfänglich eine Aktion wie`GET`, oder für ein S3-Objekt ausführt`LIST`, ruft die Speicherklasse in Ihrem Namen `PUT` auf. `CreateSession` Ihre IAM-Richtlinie muss die `s3express:CreateSession`-Genehmigung zulassen, damit der S3A-Konnektor die `CreateSession`-API aufrufen kann. Ein Beispiel für eine Richtlinie mit dieser Berechtigung finden Sie unterErste Schritte mit S3 Express One Zone.
- S3Aconnector — Um Spark für den Zugriff auf Daten aus einem Amazon S3 S3-Bucket zu konfigurieren, der die Speicherklasse S3 Express One Zone verwendet, verwenden Sie den Apache Hadoop-ConnectorS3A. Um den Connector zu verwenden, stellen Sie sicher, dass alle S3 das `s3a` Schema URIs verwenden. Wenn dies nicht der Fall ist, ändern Sie die Dateisystemimplementierung, die Sie für `s3` und die `s3n` Schemas verwenden.

Um das `s3`-Schema zu ändern, geben Sie die folgenden Clusterkonfigurationen an:

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

Um das `s3n`-Schema zu ändern, geben Sie die folgenden Clusterkonfigurationen an:

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

Erste Schritte mit S3 Express One Zone

Folgen Sie diesen Schritten, um mit S3 Express One Zone zu beginnen.

1. Erstellen Sie einen VPC-Endpunkt. Fügen Sie den Endpunkt `com.amazonaws.us-west-2.s3express` zum VPC-Endpunkt hinzu.
2. Folgen Sie Getting started with Amazon EMR Serverless, um eine Anwendung mit der Amazon EMR-Versionsbezeichnung 7.2.0 oder höher zu erstellen.
3. Konfigurieren Sie Ihre Anwendung so, dass sie den neu erstellten VPC-Endpunkt, eine private Subnetzgruppe und eine Sicherheitsgruppe verwendet.
4. Fügen Sie die `CreateSession` Berechtigung zu Ihrer Jobausführungsrolle hinzu.

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Resource": [
        "*"
      ],
      "Action": [
        "s3express:CreateSession"
      ],
      "Sid": "AllowS3EXPRESSCreatesession"
    }
```

```
  ]
}
```

5. Führen Sie Ihren Job aus. Beachten Sie, dass Sie das `S3A` Schema verwenden, um auf S3 Express One Zone-Buckets zuzugreifen.

```
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
 }'
```

Ausführen von Aufgaben

Nachdem Sie Ihre Bewerbung eingereicht haben, reichen Sie Jobs in die Bewerbung ein. In diesem Abschnitt wird beschrieben, wie Sie die verwenden AWS CLI , um diese Jobs auszuführen. In diesem Abschnitt werden auch die Standardwerte für jeden Anwendungstyp identifiziert, der auf EMR Serverless verfügbar ist.

Themen

Status von Aufgabenausführungen

Wenn Sie eine Auftragsausführung an eine Amazon EMR Serverless-Auftragswarteschlange weiterleiten, wechselt die Auftragsausführung in den `SUBMITTED` Status. Der Status eines Auftrags wird von `SUBMITTED` durchgehend `RUNNING` bis zum Erreichen von `FAILED``SUCCESS`, oder übergeben. `CANCELLING`

Aufträge können die folgenden Status haben:

Status	Description
Eingereicht	Der anfängliche Jobstatus, wenn Sie eine Jobausführung an EMR Serverless senden. Der Job wartet darauf, für die Anwendung geplant zu werden. EMR Serverless beginnt mit der Priorisierung und Planung der Jobausführung.
In Warteschlange eingefügt	Die Auftragsausführung wartet in diesem Zustand, wenn die Parallelität der Jobausführung auf Anwendungsebene vollständig belegt ist. Weitere Informationen zu Warteschlangen und Parallelität finden Sie unter. Parallelität von Aufträgen und Warteschlangen für eine serverlose EMR-Anwendung
Ausstehend	Der Scheduler wertet die Auftragsausführung aus, um die Ausführung für die Anwendung zu priorisieren und zu planen.
Geplant	EMR Serverless hat die Ausführung des Jobs für die Anwendung geplant und weist Ressourcen für die Ausführung des Jobs zu.
In Ausführung	EMR Serverless hat die Ressourcen zugewiesen, die der Job ursprünglich benötigt, und der Job wird in der Anwendung ausgeführt. In Spark-Anwendungen bedeutet dies, dass sich der Spark-Treiberprozess im `running`-Status befindet.
Fehlgeschlagen	EMR Serverless konnte den ausgeführten Job nicht an die Anwendung senden, oder er wurde erfolglos abgeschlossen. Weitere Informationen `StateDetails` zu diesem Auftragsfehler finden Sie unter.

Status	Description
Herzlichen Glückwunsch	Die Auftragsausführung wurde erfolgreich abgeschlossen.
Abbrechen	Die `CancelJobRun` API hat die Stornierung der Auftragsausführung angefordert, oder es wurde eine Zeitüberschreitung für die Auftragsausführung festgestellt. EMR Serverless versucht, den Job in der Anwendung abzubrechen und die Ressourcen freizugeben.
Abgebrochen	Die Auftragsausführung wurde erfolgreich abgebrochen, und die verwendeten Ressourcen wurden freigegeben.

Kündigung der serverlosen EMR-Auftragsausführung mit Übergangsfrist

In Datenverarbeitungssystemen können abrupte Abbrüche zu Ressourcenverschwendung, unvollständigen Vorgängen und potenziellen Dateninkonsistenzen führen. Amazon EMR Serverless ermöglicht es Ihnen, einen Übergangszeitraum für das Abbrechen von Auftragsausführungen festzulegen. Diese Funktion bietet ausreichend Zeit für die ordnungsgemäße Reinigung und den Abschluss laufender Arbeiten, bevor der Auftrag beendet wird.

Wenn Sie eine Auftragsausführung abbrechen, geben Sie mithilfe `shutdownGracePeriodInSeconds` des Parameters einen Kulanzzeitraum (in Sekunden) an, innerhalb dessen der Auftrag Bereinigungsvorgänge durchführen kann, bevor er endgültig beendet wird. Das Verhalten und die Standardeinstellungen variieren zwischen Batch- und Streaming-Aufträgen.

Übergangsfrist für Batch-Jobs

Für Batch-Jobs können Sie mit EMR Serverless benutzerdefinierte Bereinigungsvorgänge implementieren, die während der Kulanzzeit ausgeführt werden. Sie können diese Bereinigungsvorgänge als Teil des JVM-Shutdown-Hooks in Ihrem Anwendungscode registrieren.

Standardverhalten

Das Standardverhalten beim Herunterfahren besteht darin, dass es keine Übergangsfrist gibt. Es besteht aus den folgenden zwei Aktionen:

- Sofortige Kündigung
- Ressourcen werden sofort freigegeben

Konfigurationsoptionen

Sie können Einstellungen angeben, die zu einem ordnungsgemäßen Herunterfahren führen:

- Gültiger Zeitraum für den Shutdown Grace-Zeitraum: 15-1800 Sekunden (optional)
- Sofortige Kündigung (ohne Nachfrist): 0 Sekunden

Ordnungsgemäßes Herunterfahren aktivieren

Gehen Sie wie folgt vor, um das ordnungsgemäße Herunterfahren für Batch-Jobs zu implementieren:

1. Fügen Sie Ihrem Anwendungscode einen Shutdown-Hook hinzu, der eine benutzerdefinierte Shutdown-Logik enthält.

   Example in Scala

   ```
   import org.apache.hadoop.util.ShutdownHookManager

   // Register shutdown hook with priority (second argument)
   // Higher priority hooks run first
   ShutdownHookManager.get().addShutdownHook(() => {
       logger.info("Performing cleanup operations...")
   }, 100)
   ```

   Verwenden von ShutdownHookManager

   Example in PySpark

   ```
   import atexit

   def cleanup():
       # Your cleanup logic here
       print("Performing cleanup operations...")
   ```

```
# Register the cleanup function
atexit.register(cleanup)
```

2. Geben Sie beim Abbrechen des Jobs einen Übergangszeitraum an, damit die zuvor hinzugefügten Hooks ausgeführt werden können

   Beispiel

```
# Default (immediate termination)
aws emr-serverless cancel-job-run \
  --application-id APPLICATION_ID \
  --job-run-id JOB_RUN_ID

# With 5-minute grace period
aws emr-serverless cancel-job-run \
  --application-id APPLICATION_ID \
  --job-run-id JOB_RUN_ID \
  --shutdown-grace-period-in-seconds 300
```

Nachfrist für Streaming-Jobs

In Spark Structured Streaming, wo Berechnungen das Lesen von oder Schreiben in externe Datenquellen beinhalten, können abrupte Abschaltungen zu unerwünschten Ergebnissen führen. Streaming-Jobs verarbeiten Daten in Mikrostapeln, und wenn diese Vorgänge auf halbem Weg unterbrochen werden, kann es bei nachfolgenden Versuchen zu einer doppelten Verarbeitung kommen. Dies passiert, wenn der letzte Checkpoint aus dem vorherigen Micro-Batch nicht geschrieben wurde, was dazu führt, dass dieselben Daten erneut verarbeitet werden, wenn der Streaming-Job neu gestartet wird. Eine solche doppelte Verarbeitung verschwendet nicht nur Rechenressourcen, sondern kann sich auch auf den Geschäftsbetrieb auswirken. Daher ist es wichtig, abrupte Abschaltungen zu vermeiden.

EMR Serverless bietet integrierte Unterstützung für ein ordnungsgemäßes Herunterfahren über einen Streaming-Abfrage-Listener. Dadurch wird gewährleistet, dass die laufenden Mikrobatches ordnungsgemäß abgeschlossen werden, bevor der Auftrag beendet wird. Der Dienst sorgt bei Streaming-Anwendungen automatisch für ein ordnungsgemäßes Herunterfahren zwischen Mikrobatches und stellt so sicher, dass der aktuelle Mikrobatch die Verarbeitung abschließt, die Checkpoints korrekt geschrieben werden und der Streaming-Kontext sauber beendet wird, ohne dass während des Shutdown-Vorgangs neue Daten aufgenommen werden.

Standardverhalten

- Die Kulanzzeit von 120 Sekunden ist standardmäßig aktiviert.
- Der integrierte Listener für Streaming-Abfragen sorgt für ein ordnungsgemäßes Herunterfahren.

Konfigurationsoptionen

- Gültiger Zeitraum für den Shutdown Grace-Zeitraum: 15-1800 Sekunden (optional)
- Sofortige Kündigung: 0 Sekunden

Aktivieren Sie Graceful Shutdown

So implementieren Sie ein ordnungsgemäßes Herunterfahren für Streaming-Jobs:

Geben Sie beim Abbrechen des Jobs einen Übergangszeitraum an, um genügend Zeit für die Fertigstellung des laufenden Mikrobatches zu haben.

Beispiel

```
# Default graceful shutdown (120 seconds)
aws emr-serverless cancel-job-run \
  --application-id APPLICATION_ID \
  --job-run-id JOB_RUN_ID

# Custom grace period (e.g. 300 seconds)
aws emr-serverless cancel-job-run \
  --application-id APPLICATION_ID \
  --job-run-id JOB_RUN_ID \
  --shutdown-grace-period-in-seconds 300

# Immediate Termination
aws emr-serverless cancel-job-run \
  --application-id APPLICATION_ID \
  --job-run-id JOB_RUN_ID \
  --shutdown-grace-period-in-seconds 0
```

Fügen Sie benutzerdefinierte Shutdown-Hooks hinzu (optional)

Während EMR Serverless das ordnungsgemäße Herunterfahren standardmäßig über seinen integrierten Streaming-Abfrage-Listener verwaltet, können Sie optional eine benutzerdefinierte

Shutdown-Logik für einzelne Streaming-Abfragen implementieren. EMR Serverless registriert seinen Graceful Shutdown-Listener mit Priorität 60 (using). ShutdownHookManager Da Hooks mit höherer Priorität zuerst ausgeführt werden, können Sie Ihre benutzerdefinierten Bereinigungsvorgänge mit einer Priorität von mehr als 60 registrieren, um sicherzustellen, dass sie ausgeführt werden, bevor der Shutdown-Vorgang von EMR Serverless beginnt.

Informationen zum Hinzufügen eines benutzerdefinierten Hooks finden Sie im ersten Beispiel in diesem Thema. Es zeigt, wie Sie einen Shutdown-Hook in Ihren Anwendungscode einfügen. Hier ist 100 die Priorität, die größer als 60 ist. Daher läuft ein solcher Shutdown-Hook zuerst.

> **Note**
>
> Benutzerdefinierte Shutdown-Hooks sind optional und für die Funktionalität zum ordnungsgemäßen Herunterfahren, die automatisch von EMR Serverless ausgeführt wird, nicht erforderlich.

Gebühren für die Nachfrist und Batchdauer

Wenn der Standardwert für die Nachfrist (120 Sekunden) verwendet wird:

- Wenn Ihre Batchdauer weniger als 120 Sekunden beträgt, wird Ihnen nur die tatsächliche Zeit in Rechnung gestellt, die für die Fertigstellung des Stapels benötigt wird.
- Wenn Ihre Batchdauer 120 Sekunden überschreitet, wird Ihnen die maximale Kulanzzeit (120 Sekunden) in Rechnung gestellt, aber die Abfrage wird möglicherweise nicht ordnungsgemäß beendet, da sie dann zwangsweise beendet wird.

So optimieren Sie die Kosten und sorgen für ein ordnungsgemäßes Herunterfahren:

- Für Batchdauern > 120 Sekunden: Erwägen Sie, die Kulanzzeit entsprechend Ihrer Batchdauer zu verlängern
- Für Batchdauern < 120 Sekunden: Sie müssen den Kulanzzeitraum nicht anpassen, da Ihnen nur die tatsächliche Verarbeitungszeit in Rechnung gestellt wird

Überlegungen

Verhalten im Rahmen der Nachfrist

- Die Übergangszeit gibt Ihnen Zeit, bis Ihre registrierten Shutdown-Hooks abgeschlossen sind.
- Der Job wird beendet, sobald der Shutdown-Hook beendet ist, auch wenn er weit vor der Kulanzfrist liegt.
- Wenn die Bereinigungsvorgänge den Kulanzzeitraum überschreiten, wird der Job gewaltsam beendet.

Verhalten der Dienste

- Das Herunterfahren im Grace Period ist nur für Jobs im Status RUNNING verfügbar.
- Nachfolgende Stornierungsanfragen im Status CANCELING werden ignoriert.
- Wenn EMR Serverless aufgrund interner Dienstfehler das Herunterfahren im Grace Period nicht initiieren kann:
  - Der Dienst versucht es für bis zu 2 Minuten erneut.
  - Wenn die Wiederholungsversuche nicht erfolgreich sind, wird der Job gewaltsam beendet.

Fakturierung

Aufträgen werden die verbrauchten Rechenressourcen in Rechnung gestellt, bis der Job vollständig beendet wird. Dies gilt auch für die Zeit, die während der Übergangszeit in Anspruch genommen wurde.

Jobs von der EMR Studio-Konsole aus ausführen

Sie können Jobausführungen an EMR Serverless-Anwendungen senden und über die EMR Studio-Konsole auf die Jobs zugreifen. Folgen Sie den Anweisungen unter Erste Schritte von der Konsole aus, um Ihre EMR Serverless-Anwendung auf der EMR Studio-Konsole zu erstellen oder zu ihr zu navigieren.

Reichen Sie einen Job ein

Senden Sie auf der Seite Job einreichen wie folgt einen Job an eine EMR Serverless-Anwendung.

Spark

1. Geben Sie im Feld Name einen Namen für Ihre Jobausführung ein.
2. Geben Sie im Feld Runtime-Rolle den Namen der IAM-Rolle ein, die Ihre EMR Serverless-Anwendung für die Jobausführung annehmen kann. Weitere Informationen zu Runtime-Rollen finden Sie unter. Job-Runtime-Rollen für Amazon EMR Serverless
3. Geben Sie im Feld Skriptspeicherort den Amazon S3 S3-Speicherort für das Skript oder die JAR ein, das Sie ausführen möchten. Für Spark-Jobs kann das Skript eine Python (`.py`) -Datei oder eine JAR (`.jar`) -Datei sein.
4. Wenn es sich bei Ihrem Skriptspeicherort um eine JAR-Datei handelt, geben Sie den Klassennamen, der den Einstiegspunkt für den Job darstellt, in das Feld Hauptklasse ein.
5. (Optional) Geben Sie Werte für die verbleibenden Felder ein.

   - Skriptargumente — Geben Sie alle Argumente ein, die Sie an Ihr JAR- oder Python-Skript übergeben möchten. Ihr Code liest diese Parameter. Trennen Sie jedes Argument im Array durch ein Komma.
   - Spark-Eigenschaften — Erweitern Sie den Abschnitt Spark-Eigenschaften und geben Sie alle Spark-Konfigurationsparameter in dieses Feld ein.

     > **Note**
     >
     > Wenn Sie die Größe des Spark-Treibers und des Executors angeben, berücksichtigen Sie den Speicheraufwand. Geben Sie die Werte für den Speicheraufwand in den Eigenschaften `spark.driver.memory0verhead` und an. `spark.executor.memory0verhead` Der Speicher-Overhead hat einen Standardwert von 10% des Container-Speichers, mit einem Minimum von 384 MB. Der Executor-Speicher und der Speicher-Overhead zusammen dürfen den Arbeitsspeicher nicht überschreiten. Beispielsweise muss der Höchstwert `spark.executor.memory` für einen 30-GB-Worker 27 GB betragen.

   - Auftragskonfiguration — Geben Sie in diesem Feld eine beliebige Jobkonfiguration an. Sie können diese Jobkonfigurationen verwenden, um die Standardkonfigurationen für Anwendungen zu überschreiben.
   - Zusätzliche Einstellungen — Aktivieren oder deaktivieren Sie den AWS Glue-Datenkatalog als Metastore und ändern Sie die Einstellungen für das Anwendungsprotokoll. Weitere Informationen zu Metastore-Konfigurationen finden Sie unter. Metastore-

Konfiguration für EMR Serverless Weitere Informationen zu den Optionen für die Anwendungsprotokollierung finden Sie Speichern von Protokollen unter.

- Tags — Weisen Sie der Anwendung benutzerdefinierte Tags zu.

6. Wählen Sie Auftrag absenden.

Hive

1. Geben Sie im Feld Name einen Namen für Ihre Jobausführung ein.
2. Geben Sie im Feld Runtime-Rolle den Namen der IAM-Rolle ein, die Ihre EMR Serverless-Anwendung für die Jobausführung annehmen kann.
3. Geben Sie im Feld Skriptspeicherort den Amazon S3 S3-Speicherort für das Skript oder die JAR ein, das Sie ausführen möchten. Für Hive-Jobs muss das Skript eine Hive (`.sql`) -Datei sein.
4. (Optional) Geben Sie Werte für die verbleibenden Felder ein.

   - Speicherort für das Initialisierungsskript — Geben Sie den Speicherort des Skripts ein, das Tabellen initialisiert, bevor das Hive-Skript ausgeführt wird.
   - Hive-Eigenschaften — Erweitern Sie den Bereich Hive-Eigenschaften und geben Sie alle Hive-Konfigurationsparameter in dieses Feld ein.
   - Auftragskonfiguration — Geben Sie eine beliebige Jobkonfiguration an. Sie können diese Jobkonfigurationen verwenden, um die Standardkonfigurationen für Anwendungen zu überschreiben. Für Hive-Jobs `hive.metastore.warehouse.dir` sind `hive.exec.scratchdir` diese Eigenschaften in der `hive-site` Konfiguration erforderlich.

```
{
    "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": {}
}
```

- Zusätzliche Einstellungen — Aktivieren oder deaktivieren Sie den AWS Glue-Datenkatalog als Metastore und ändern Sie die Einstellungen für das Anwendungsprotokoll. Weitere Informationen zu Metastore-Konfigurationen finden Sie unter. Metastore-Konfiguration für EMR Serverless Weitere Informationen zu den Optionen für die Anwendungsprotokollierung finden Sie Speichern von Protokollen unter.
- Tags — Weisen Sie der Anwendung beliebige benutzerdefinierte Tags zu.

5. Wählen Sie Auftrag absenden.

Der Access-Job wird ausgeführt

Greifen Sie auf der Detailseite einer Anwendung über die Registerkarte Auftragsausführungen auf Jobausführungen zu und führen Sie die folgenden Aktionen für Jobausführungen durch.

Job abbrechen — Um einen Joblauf abzubrechen, der sich im `RUNNING` Status befindet, wählen Sie diese Option. Weitere Informationen zu Übergängen bei der Auftragsausführung finden Sie unterStatus von Aufgabenausführungen.

Auftrag klonen — Um eine vorherige Auftragsausführung zu klonen und erneut einzureichen, wählen Sie diese Option.

Jobs werden von der ausgeführt AWS CLI

Sie können einzelne Jobs auf der erstellen, beschreiben und löschen AWS CLI. Sie können auch alle Ihre Jobs auflisten, um auf einen Blick darauf zuzugreifen.

Um einen neuen Job einzureichen, verwenden Sie`start-job-run`. Geben Sie die ID der Anwendung an, die Sie ausführen möchten, sowie die auftragsspezifischen Eigenschaften. Spark-Beispiele finden Sie unter. Verwenden von Spark-Konfigurationen bei der Ausführung von EMR Serverless-Jobs Hive-Beispiele finden Sie unter. Verwenden von Hive-Konfigurationen bei der Ausführung von EMR Serverless-Jobs Dieser Befehl gibt your`application-id`, ARN und new zurück`job-id`.

Jeder ausgeführte Job hat eine festgelegte Timeout-Dauer. Wenn die Auftragsausführung diese Dauer überschreitet, wird sie von EMR Serverless automatisch abgebrochen. Das Standard-Timeout beträgt 12 Stunden. Wenn Sie Ihre Auftragsausführung starten, konfigurieren Sie diese Timeout-

Einstellung auf einen Wert, der Ihren Jobanforderungen entspricht. Konfigurieren Sie den Wert mit der `executionTimeoutMinutes` Eigenschaft.

```
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://amzn-s3-demo-bucket/scripts/create_table.sql",
        "parameters": "--hiveconf hive.exec.scratchdir=s3://amzn-s3-demo-bucket/
hive/scratch --hiveconf hive.metastore.warehouse.dir=s3://amzn-s3-demo-bucket/hive/
warehouse"
    }
   }' \
  --configuration-overrides '{
    "applicationConfiguration": [{
        "classification": "hive-site",
        "properties": {
            "hive.client.cores": "2",
            "hive.client.memory": "4GIB"
        }
    }]
}'
```

Um einen Job zu beschreiben, verwenden Sie`get-job-run`. Dieser Befehl gibt auftragsspezifische Konfigurationen und die eingestellte Kapazität für Ihren neuen Job zurück.

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

Um Ihre Jobs aufzulisten, verwenden Sie. `list-job-runs` Dieser Befehl gibt einen abgekürzten Satz von Eigenschaften zurück, der Jobtyp, Status und andere allgemeine Attribute umfasst. Wenn Sie nicht auf alle Ihre Jobs zugreifen möchten, geben Sie die maximale Anzahl von Jobs an, auf die Sie zugreifen möchten (bis zu 50). Das folgende Beispiel gibt an, dass Sie auf Ihre letzten beiden Auftragsausführungen zugreifen möchten.

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

Um einen Job abzubrechen, verwenden Sie`cancel-job-run`. Geben Sie das `application-id` und das `job-id` des Jobs an, den Sie stornieren möchten.

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

Weitere Informationen zum Ausführen von Jobs aus dem finden Sie in der AWS CLIEMR Serverless API Reference.

IAM-Richtlinie für die Ausführung

Sie können zusätzlich zu einer Ausführungsrolle eine Ausführungs-IAM-Richtlinie angeben, wenn das Senden von Jobs auf EMR Serverless ausgeführt wird. Die sich daraus ergebenden Berechtigungen, die bei der Ausführung des Jobs angenommen werden, stellen die Schnittmenge der Berechtigungen in der Ausführungsrolle und der angegebenen Ausführungs-IAM-Richtlinie dar.

Erste Schritte

Schritte zur Verwendung der Ausführungs-IAM-Richtlinie:

Erstellen Sie eine `emr-serverless` Anwendung oder verwenden Sie eine vorhandene und führen Sie dann die folgende AWS-CLI aus, um eine Jobausführung mit einer Inline-IAM-Richtlinie zu starten:

```
aws emr-serverless start-job-run --region us-west-2 \
      --application-id application-id \
      --execution-role-arn execution-role-arn \
      --job-driver job-driver-options \
      --execution-iam-policy '{"policy": "inline-policy"}'
```

Beispiele für CLI-Befehle

Wenn wir die folgende Richtlinie in der `policy.json` Datei auf dem Computer gespeichert haben:

JSON

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

```
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::my-test-bucket",
        "arn:aws:s3:::my-test-bucket/*"
      ],
      "Sid": "AllowS3Getobject"
    }
  ]
}
```

Dann können wir mit dem folgenden AWS CLI Befehl einen Job mit dieser Richtlinie starten:

```
aws emr-serverless start-job-run --region us-west-2 \
      --application-id application-id \
      --execution-role-arn execution-role-arn \
      --job-driver job-driver-options
      --execution-iam-policy '{
          "policy": '$(jq -c '. | @json' policy.json)'
      }'
```

Sie können auch beide AWS und vom Kunden verwaltete Richtlinien verwenden und diese anhand der folgenden Optionen spezifizieren ARNs:

```
aws emr-serverless start-job-run --region us-west-2 \
      --application-id application-id \
      --execution-role-arn execution-role-arn \
      --job-driver job-driver-options
      --execution-iam-policy '{
          "policyArns": [
          "arn:aws:iam::aws:policy/AmazonS3FullAccess",
          "arn:aws:iam::aws:policy/CloudWatchLogsFullAccess"
          ]
    }'
```

Es ist auch möglich, ARNs in derselben Anfrage sowohl eine Inline-IAM-Richtlinie als auch eine verwaltete Richtlinie anzugeben:

```
aws emr-serverless start-job-run --region us-west-2 \
      --application-id application-id \
      --execution-role-arn execution-role-arn \
      --job-driver job-driver-options
      --execution-iam-policy '{
          "policy": '$(jq -c '. | @json' policy.json)',
          "policyArns": [
          "arn:aws:iam::aws:policy/AmazonS3FullAccess",
          "arn:aws:iam::aws:policy/CloudWatchLogsFullAccess"
          ]
      }'
```

Wichtige Hinweise

- `execution-role-policy`Das `policy` Feld 'kann eine maximale Länge von 2048 Zeichen haben.
- Die im `policy` Feld `execution-iam-policy`'s angegebene Inline-IAM-Richtlinienzeichenfolge muss dem JSON-Zeichenfolgenstandard entsprechen, ohne dass Zeilenumbrüche und Anführungszeichen wie im vorherigen Beispiel maskiert werden.
- Eine Liste mit bis zu 10 verwalteten Richtlinien ARNs kann als Wert für das `execution-iam-policy` Feld 'angegeben werden. `policyArns`
- Bei der verwalteten Richtlinie ARNs muss es sich um eine Liste gültiger AWS oder vom Kunden verwalteter Policy-ARN handeln. Wenn ein vom Kunden verwalteter Policy-ARN angegeben wird, muss die Richtlinie zu demselben AWS Konto des EMR-S JobRun gehören.
- Wenn sowohl die Inline-IAM-Richtlinie als auch die verwalteten Richtlinien verwendet werden, darf der Klartext, den Sie für die Inline- und verwalteten Richtlinien zusammen verwenden, 2.048 Zeichen nicht überschreiten.
- Die resultierenden Berechtigungen, von denen angenommen JobRun wird, sind die Schnittmenge der Berechtigungen in der Ausführungsrolle und der angegebenen Ausführungs-IAM-Richtlinie.

Überschneidung der Richtlinien

Die sich daraus ergebenden Berechtigungen, die von der Auftragsausführung angenommen werden, stellen die Schnittmenge der Berechtigungen in der Ausführungsrolle und der angegebenen Ausführungs-IAM-Richtlinie dar. Das bedeutet, dass alle erforderlichen Berechtigungen an beiden Stellen angegeben werden müssen, JobRun damit sie funktionieren. Es ist jedoch möglich, in

der Inline-Richtlinie eine zusätzliche pauschale Zulassungsanweisung für alle Berechtigungen anzugeben, die Sie nicht aktualisieren oder überschreiben möchten.

Beispiel

Angesichts der folgenden IAM-Rollenrichtlinie für die Ausführung:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:*"
      ],
      "Resource": [
        "*"
      ],
      "Sid": "AllowS3"
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "arn:aws:logs:us-west-2:123456789012:log-group::log-stream"
      ],
      "Sid": "AllowLOGSDescribeloggroups"
    },
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:DescribeTable"
      ],
      "Resource": [
        "arn:aws:dynamodb:*:*:table/MyCompany1table"
      ],
      "Sid": "AllowDYNAMODBDescribetable"
    }
  ]
```

```
}
```

Und die folgende Inline-IAM-Richtlinie:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::my-test-bucket/tenant1",
        "arn:aws:s3:::my-test-bucket/tenant1/*"
      ],
      "Sid": "AllowS3Getobject"
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:*",
        "dynamodb:*"
      ],
      "Resource": [
        "*"
      ],
      "Sid": "AllowLOGS"
    }
  ]
}
```

Die daraus resultierenden Berechtigungen, die von angenommen werden, lauten wie JobRun folgt:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::my-test-bucket/tenant1",
        "arn:aws:s3:::my-test-bucket/tenant1/*"
      ],
      "Sid": "AllowS3Getobject"
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "arn:aws:logs:us-west-2:123456789012:log-group::log-stream"
      ],
      "Sid": "AllowLOGSDescribeloggroups"
    },
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:DescribeTable"
      ],
      "Resource": [
        "arn:aws:dynamodb:*:*:table/MyCompany1table"
      ],
      "Sid": "AllowDYNAMODBDescribetable"
    }
  ]
}
```

Verwenden von für Shuffle optimierten Festplatten

Verwenden Sie bei Amazon EMR-Versionen 7.1.0 und höher für Shuffle-optimierte Festplatten, wenn Sie Apache Spark- oder Hive-Jobs ausführen, um die Leistung für I/O-intensive workloads. Compared to standard disks, shuffle-optimized disks provide higher IOPS (I/O Operationen (pro Sekunde) zu verbessern und so die Datenbewegung zu beschleunigen und die Latenz bei Shuffle-Vorgängen zu reduzieren. Mit Shuffle-optimierten Festplatten können Sie Festplattengrößen von bis zu 2 TB pro Worker zuordnen. Konfigurieren Sie also die entsprechende Kapazität für Ihre Workload-Anforderungen.

Wichtigste Vorteile

Für den Shuffle-Modus optimierte Festplatten bieten die folgenden Vorteile.

- Hohe IOPS-Leistung — Für Shuffle optimierte Festplatten bieten höhere IOPS als Standardfestplatten, was zu einer effizienteren und schnelleren Datenmischung bei Spark- und Hive-Jobs und anderen Shuffle-intensiven Workloads führt.
- Größere Festplattengröße — Für Shuffle optimierte Festplatten unterstützen Festplattengrößen von 20 GB bis 2 TB pro Mitarbeiter. Wählen Sie also die passende Kapazität für Ihre Workloads.

Erste Schritte

Sehen Sie sich die folgenden Schritte an, um für Shuffle optimierte Festplatten in Ihren Workflows zu verwenden.

Spark

1. Erstellen Sie mit dem folgenden Befehl eine EMR Serverless Release 7.1.0-Anwendung.

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

2. Konfigurieren Sie Ihren Spark-Job so, dass er die Parameter für die Ausführung mit `spark.emr-serverless.driver.disk.type` and/or `spark.emr-serverless.executor.disk.type` für Shuffle optimierten Festplatten enthält. Sie können je nach Anwendungsfall entweder einen oder beide Parameter verwenden.

```
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"
        }
    }'
```

Weitere Informationen finden Sie unter Spark-Jobeigenschaften.

Hive

1. Erstellen Sie mit dem folgenden Befehl eine EMR Serverless Release 7.1.0-Anwendung.

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

2. Konfigurieren Sie Ihren Hive-Job so, dass er die Parameter für die Ausführung mit für den Shuffle-Modus `hive.driver.disk.type` and/or `hive.tez.disk.type` optimierten Festplatten enthält. Sie können je nach Anwendungsfall entweder einen oder beide Parameter verwenden.

```
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"
            }
        }]
    }'
```

Weitere Informationen finden Sie unter Hive-Jobeigenschaften.

Konfiguration einer Anwendung mit vorinitialisierter Kapazität

Sehen Sie sich die folgenden Beispiele an, um Anwendungen zu erstellen, die auf Amazon EMR Version 7.1.0 basieren. Diese Anwendungen haben die folgenden Eigenschaften:

- 5 vorinitialisierte Spark-Treiber mit jeweils 2 vCPUs, 4 GB Arbeitsspeicher und 50 GB Shuffle-optimierter Festplatte.
- 50 vorinitialisierte Executoren mit jeweils 4 vCPUs, 8 GB Arbeitsspeicher und 500 GB Shuffle-optimierter Festplatte.

Wenn diese Anwendung Spark-Jobs ausführt, verbraucht sie zuerst die vorinitialisierten Worker und skaliert dann die On-Demand-Worker auf die maximale Kapazität von 400 vCPU und 1024 GB Arbeitsspeicher. Optional können Sie die Kapazität für entweder oder weglassen. `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"
}'
```

Serverlosen Speicher für Amazon EMR Serverless verwenden

Verwenden Sie bei Amazon EMR-Versionen 7.12 und höher serverlosen Speicher, wenn Sie Apache Spark-Jobs ausführen, um die lokale Festplattenbereitstellung zu vermeiden, die Datenverarbeitungskosten zu senken und Jobausfälle aufgrund von Festplattenkapazitätsbeschränkungen zu verhindern. Serverloser Speicher verarbeitet automatisch Shuffle-, Disk Spill- und Disk-Caching-Operationen für Ihre Jobs, ohne dass eine Kapazitätskonfiguration erforderlich ist, und speichert Zwischendaten kostenlos. Amazon EMR Serverless speichert Zwischendaten in einem vollständig verwalteten serverlosen Speicher, der je nach Workload-Anforderungen automatisch skaliert wird und es Spark ermöglicht, Rechenarbeiter sofort freizugeben, wenn sie inaktiv sind, wodurch die Rechenkosten gesenkt werden.

Wichtigste Vorteile

Serverloser Speicher für EMR Serverless bietet die folgenden Vorteile.

- Speicher ohne Konfiguration — Serverloser Speicher macht es überflüssig, den Typ und die Größe der lokalen Festplatte für jede Anwendung oder jeden Job zu konfigurieren. EMR Serverless verwaltet automatisch Zwischendatenoperationen ohne Kapazitätsplanung.

- Beugt Jobausfällen durch automatische Skalierung vor — Die Speicherkapazität wird automatisch auf der Grundlage des Workload-Bedarfs skaliert und verhindert so Jobausfälle aufgrund unzureichender Festplattenkapazität.
- Geringere Datenverarbeitungskosten — Serverloser Speicher senkt die Verarbeitungskosten durch zwei Mechanismen. Erstens wird der Datenzwischenspeicher kostenlos bereitgestellt — Sie zahlen nur für Rechen- und Speicherressourcen. Zweitens ermöglicht der entkoppelte Speicher mit der dynamischen Ressourcenzuweisung von Spark es Spark, Mitarbeiter sofort freizugeben, wenn sie inaktiv sind, anstatt sie zu behalten, um Zwischendaten auf lokalen Festplatten aufzubewahren. Dies ermöglicht eine schnellere Skalierung und Skalierung pro Spark-Phase und reduziert so die Rechenkosten für Jobs, bei denen in späteren Phasen weniger Mitarbeiter benötigt werden als in der Anfangsphase.
- Verschlüsselter Speicher mit Isolierung auf Jobebene — Alle zwischengeschalteten Daten werden bei der Übertragung und im Ruhezustand mit strikter Isolierung auf Jobebene verschlüsselt.
- Unterstützung für feinkörnige Zugriffskontrolle — Serverloser Speicher unterstützt eine differenzierte Zugriffskontrolle durch die Integration von AWS Lake Formation.

Erste Schritte

Sehen Sie sich die folgenden Schritte an, um serverlosen Speicher für EMR Serverless in Ihren Spark-Workflows zu verwenden.

1. Eine serverlose EMR-Anwendung erstellen

   Erstellen Sie eine EMR Serverless Version 7.12 (oder höher) mit aktiviertem serverlosem Speicher, indem Sie die Eigenschaft spark in der Spark-Defaults-Klassifizierung `spark.aws.serverlessStorage.enabled` auf true setzen.

   ```
   aws emr-serverless create-application \
     --type "SPARK" \
     --name my-application \
     --release-label emr-7.12.0 \
     --runtime-configuration '[{
         "classification": "spark-defaults",
           "properties": {
             "spark.aws.serverlessStorage.enabled": "true"
           }
       }]' \
     --region <AWS_REGION>
   ```

2. Starten Sie einen Spark-Job

   Starten Sie einen Job, der auf Ihrer Anwendung ausgeführt wird. Serverloser Speicher für EMR Serverless verarbeitet automatisch zwischengeschaltete Datenoperationen wie Shuffle für Ihren Job.

```
aws emr-serverless start-job-run \
  --application-id <application-id> \
  --execution-role-arn <job-role-arn> \
  --job-driver '{
    "sparkSubmit": {
      "entryPoint": "s3://<bucket>/script.py",
      "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=10"
    }
  }'
```

   Sie können serverlosen Speicher für EMR Serverless auch auf Auftragsebene aktivieren, auch wenn er auf Anwendungsebene nicht aktiviert ist. Dadurch werden Worker-Knoten gestartet, die mit serverlosem Speicher ausgestattet sind, um Ihre Jobs zu verarbeiten. Sie können den serverlosen Speicher auch für einen bestimmten Job deaktivieren, indem Sie dieselbe Spark-Eigenschaft **`spark.aws.serverlessStorage.enabled`** auf false setzen.

```
# Turn on serverless storage for EMR serverless for a specific job
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.aws.serverlessStorage.enabled": "true"
        }
    }'
```

> **Note**
>
> Um weiterhin die herkömmliche Bereitstellung lokaler Festplatten zu verwenden, lassen Sie die **`spark.aws.serverlessStorage.enabled`** Konfiguration weg oder setzen Sie sie auf False.

Überlegungen und Einschränkungen

- Release-Version — Serverloser Speicher wird in Amazon EMR Version 7.12 und höher unterstützt.
- Grenzwerte für das Datenvolumen — Jeder Auftrag kann bis zu 200 GB an Zwischendaten pro Auftragsausführung lesen und schreiben. Aufträge, die dieses Limit überschreiten, schlagen fehl und es wird eine Fehlermeldung angezeigt, die darauf hinweist, dass das Speicherlimit für serverloses Speichern erreicht wurde.
- Timeout bei der Auftragsausführung — Serverloser Speicher unterstützt Jobs mit Ausführungszeitüberschreitungen von bis zu 24 Stunden. Jobs, die für längere Ausführungszeitlimits konfiguriert sind, schlagen mit einer Fehlermeldung fehl.
- Vorinitialisierte Kapazität — Vorinitialisierte Capacity-Worker unterstützen keinen serverlosen Speicher. Wenn Sie vorinitialisierte Kapazität konfigurieren, wird sie nur von Aufträgen genutzt, bei denen serverloser Speicher auf Auftragsebene explizit deaktiviert wird. Jobs mit aktiviertem serverlosem Speicher stellen bei Bedarf immer neue Mitarbeiter bereit und nutzen keine vorinitialisierte Kapazität, unabhängig von der Konfiguration auf Anwendungsebene.
- Workload-Typen — Serverloser Speicher wird für Streaming- und interaktive Jobs nicht unterstützt.
- Worker-Konfiguration — Serverloser Speicher wird für Worker mit 1 oder 2 V nicht unterstützt. CPUs

Unterstützt AWS-Regionen

EMR Serverless unterstützt serverlosen Speicher in den folgenden Regionen:

- USA Ost (Nord-Virginia)
- USA West (Oregon)
- Europa (Irland)

Streaming-Jobs für die Verarbeitung kontinuierlich gestreamter Daten

Ein Streaming-Job in EMR Serverless ist ein Jobmodus, mit dem Sie Streaming-Daten nahezu in Echtzeit analysieren und verarbeiten können. Diese lang andauernden Jobs fragen Streaming-Daten ab und verarbeiten die Ergebnisse kontinuierlich, sobald Daten eintreffen. Streaming-Jobs eignen sich am besten für Aufgaben, die eine Datenverarbeitung in Echtzeit erfordern, wie z. B. Analysen, Betrugserkennung und Empfehlungsprogramme nahezu in Echtzeit. Serverlose EMR-Streaming-Jobs bieten Optimierungen, wie z. B. integrierte Job-Resilienz, Echtzeitüberwachung, verbessertes Protokollmanagement und Integration mit Streaming-Connectoren.

Im Folgenden sind einige Anwendungsfälle für Streaming-Jobs aufgeführt:

- Analysen nahezu in Echtzeit — Mit Streaming-Jobs in Amazon EMR Serverless können Sie Streaming-Daten nahezu in Echtzeit verarbeiten, sodass Sie Echtzeitanalysen für kontinuierliche Datenströme wie Protokolldaten, Sensordaten oder Clickstream-Daten durchführen können, um Erkenntnisse zu gewinnen und zeitnahe Entscheidungen auf der Grundlage der neuesten Informationen zu treffen.
- Betrugserkennung — Verwenden Sie Streaming-Jobs, um Betrug bei Finanztransaktionen, Kreditkartenoperationen oder Online-Aktivitäten nahezu in Echtzeit zu erkennen, wenn Sie Datenströme analysieren und verdächtige Muster oder Anomalien identifizieren, sobald sie auftreten.
- Empfehlungs-Engines — Streaming-Jobs können Benutzeraktivitätsdaten verarbeiten und Empfehlungsmodelle aktualisieren. Dies eröffnet Möglichkeiten für personalisierte Empfehlungen in Echtzeit, die auf Verhaltensweisen und Präferenzen basieren.
- Analyse sozialer Medien — Streaming-Jobs können Social-Media-Daten wie Tweets, Kommentare und Beiträge verarbeiten, sodass Unternehmen Trends verfolgen, Stimmungsanalysen durchführen und den Ruf der Marke nahezu in Echtzeit verwalten können.
- IoT-Analysen (Internet of Things) — Streaming-Jobs können Datenströme mit hoher Geschwindigkeit von IoT-Geräten, Sensoren und verbundenen Maschinen verarbeiten und analysieren, sodass Anomalieerkennung, vorausschauende Wartung und andere Anwendungsfälle für IoT-Analysen durchgeführt werden können.
- Clickstream-Analyse — Streaming-Jobs können Clickstream-Daten von Websites oder mobilen Anwendungen verarbeiten und analysieren. Unternehmen, die solche Daten verwenden, können Analysen durchführen, um mehr über das Nutzerverhalten zu erfahren, Benutzererlebnisse zu personalisieren und Marketingkampagnen zu optimieren.

- Überwachung und Analyse von Protokollen — Streaming-Jobs können auch Protokolldaten von Servern, Anwendungen und Netzwerkgeräten verarbeiten. Auf diese Weise können Sie Anomalien erkennen, Fehler beheben und den Zustand und die Leistung Ihres Systems verbessern.

Die wichtigsten Vorteile

Streaming-Jobs in EMR Serverless sorgen automatisch für Job-Resilienz, die sich aus einer Kombination der folgenden Faktoren ergibt:

- Automatische Wiederholung — EMR Serverless wiederholt automatisch alle fehlgeschlagenen Jobs ohne manuelles Eingreifen Ihrerseits.
- Resilienz in der Availability Zone (AZ) — EMR Serverless schaltet Streaming-Jobs automatisch auf eine fehlerfreie AZ um, wenn in der ursprünglichen AZ Probleme auftreten.
- Protokollverwaltung:
  - Protokollrotation — Für eine effizientere Festplattenspeicherverwaltung rotiert EMR Serverless die Protokolle für lange Streaming-Jobs regelmäßig. Dadurch wird eine Anhäufung von Protokollen verhindert, die möglicherweise den gesamten Festplattenspeicher beansprucht.
  - Protokollkomprimierung — unterstützt Sie bei der effizienten Verwaltung und Optimierung von Protokolldateien in verwalteter Persistenz. Die Komprimierung verbessert auch das Debug-Erlebnis, wenn Sie den Managed Spark History Server verwenden.

Unterstützte Datenquellen und Datensenken

EMR Serverless arbeitet mit einer Reihe von Eingabedatenquellen und Ausgabedatensenken:

- Unterstützte Eingabedatenquellen — Amazon Kinesis Data Streams, Amazon Managed Streaming for Apache Kafka und selbstverwaltete Apache Kafka-Cluster. Standardmäßig enthalten Amazon EMR-Versionen 7.1.0 und höher den Amazon Kinesis Data Streams Streams-Connector, sodass Sie keine zusätzlichen Pakete erstellen oder herunterladen müssen.
- Unterstützte Ausgabedatensenken — AWS Glue Data Catalog-Tabellen, Amazon S3, Amazon Redshift, MySQL, PostgreSQL Oracle, Oracle, Microsoft SQL, Apache Iceberg, Delta Lake und Apache Hudi.

Überlegungen und Einschränkungen

Beachten Sie bei der Verwendung von Streaming-Jobs die folgenden Überlegungen und Einschränkungen.

- Streaming-Jobs werden mit Amazon EMR-Versionen 7.1.0 und höher unterstützt.
- EMR Serverless geht davon aus, dass Streaming-Jobs über einen längeren Zeitraum ausgeführt werden. Sie können daher kein Ausführungstimeout festlegen, um die Laufzeit des Jobs zu begrenzen.
- Streaming-Jobs sind nur mit der Spark-Engine kompatibel, die auf dem strukturierten Streaming-Framework basiert.
- EMR Serverless versucht auf unbestimmte Zeit erneut, Streaming-Jobs zu streamen, und Sie können die Anzahl der maximalen Versuche nicht anpassen. Der Thrash-Schutz ist automatisch enthalten, um die Auftragswiederholung zu beenden, wenn die Anzahl der fehlgeschlagenen Versuche einen innerhalb eines Stundenfensters festgelegten Schwellenwert überschreitet. Der Standardschwellenwert liegt bei fünf fehlgeschlagenen Versuchen innerhalb einer Stunde. Sie können diesen Schwellenwert so konfigurieren, dass er zwischen 1 und 10 Versuchen liegt. Weitere Informationen finden Sie unter Job Resiliency.
- Streaming-Jobs verfügen über Checkpoints, um den Laufzeitstatus und den Fortschritt zu speichern, sodass EMR Serverless den Streaming-Job vom letzten Checkpoint aus fortsetzen kann. Weitere Informationen finden Sie unter Recovering from failure with Checkpointing in der Apache Spark-Dokumentation.

Erste Schritte beim Streamen von Jobs

In den folgenden Anweisungen erfahren Sie, wie Sie mit Streaming-Jobs beginnen können.

1. Folgen Sie Getting started with Amazon EMR Serverless, um eine Anwendung zu erstellen. Beachten Sie, dass Ihre Anwendung Amazon EMR Version 7.1.0 oder höher ausführen muss.
2. Sobald Ihre Anwendung bereit ist, setzen Sie den `mode` Parameter auf, `STREAMING` um einen Streaming-Job zu senden, ähnlich dem folgenden AWS CLI Beispiel.

```
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"
    }
}'
```

Unterstützte Streaming-Konnektoren

Streaming-Konnektoren erleichtern das Lesen von Daten aus einer Streaming-Quelle und können auch Daten in eine Streaming-Senke schreiben.

Die folgenden Streaming-Connectoren werden unterstützt:

Amazon Kinesis Data Streams Streams-Konnektor

Der Amazon Kinesis Data Streams-Connector für Apache Spark ermöglicht die Erstellung von Streaming-Anwendungen und -Pipelines, die Daten von Amazon Kinesis Data Streams nutzen und Daten in Amazon Kinesis Data Streams schreiben. Der Connector unterstützt einen erhöhten Fan-Out-Verbrauch mit einer speziellen Lesedurchsatzrate von bis zu 2 MB/Sekunde pro Shard. Standardmäßig enthalten Amazon EMR Serverless 7.1.0 und höher den Connector, sodass Sie keine zusätzlichen Pakete erstellen oder herunterladen müssen. Weitere Informationen zum Connector finden Sie auf der spark-sql-kinesis-connector Seite unter. GitHub

Im Folgenden finden Sie ein Beispiel dafür, wie Sie eine Jobausführung mit der Kinesis Data Streams Streams-Konnektorabhängigkeit starten.

```
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"
    }
}'
```

Um eine Verbindung zu Kinesis Data Streams herzustellen, konfigurieren Sie die EMR Serverless-Anwendung mit VPC-Zugriff und verwenden Sie einen VPC-Endpunkt, um privaten Zugriff zu ermöglichen. Oder verwenden Sie ein NAT-Gateway, um öffentlichen Zugriff zu erhalten. Weitere Informationen finden Sie unter VPC-Zugriff konfigurieren. Sie müssen außerdem sicherstellen, dass Ihre Job-Runtime-Rolle über die erforderlichen Lese- und Schreibberechtigungen für den Zugriff auf die erforderlichen Datenströme verfügt. Weitere Informationen zur Konfiguration einer Job-Runtime-Rolle finden Sie unter Job-Runtime-Rollen für Amazon EMR Serverless. Eine vollständige Liste aller erforderlichen Berechtigungen finden Sie auf der spark-sql-kinesis-connector Seite unter. GitHub

Apache Kafka-Konnektor

Der Apache Kafka-Konnektor für strukturiertes Spark-Streaming ist ein Open-Source-Konnektor der Spark-Community und in einem Maven-Repository verfügbar. Dieser Konnektor ermöglicht es strukturierten Spark-Streaming-Anwendungen, Daten aus selbstverwaltetem Apache Kafka und Amazon Managed Managed Streaming for Apache Kafka Apache Kafka zu lesen und Daten in diese zu schreiben. Weitere Informationen über den Konnektor finden Sie im Structured Streaming + Kafka Integration Guide in der Apache Spark-Dokumentation.

Das folgende Beispiel zeigt, wie Sie den Kafka-Konnektor in Ihre Job-Run-Anforderung aufnehmen können.

```
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>"
    }
}'
```

Die Version des Apache Kafka-Connectors hängt von Ihrer EMR Serverless Release-Version und der entsprechenden Spark-Version ab. Die richtige Kafka-Version finden Sie im Structured Streaming + Kafka Integration Guide.

Um Amazon Managed Streaming for Apache Kafka mit IAM-Authentifizierung zu verwenden, fügen Sie eine weitere Abhängigkeit hinzu, damit der Kafka-Connector über IAM eine Verbindung zu Amazon MSK herstellen kann. Weitere Informationen finden Sie im Repository unter. aws-msk-iam-auth GitHub Sie müssen außerdem sicherstellen, dass die Job-Runtime-Rolle über die erforderlichen IAM-Berechtigungen verfügt. Das folgende Beispiel zeigt, wie der Connector mit IAM-Authentifizierung verwendet wird.

```
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>"
    }
}'
```

Um den Kafka-Connector und die IAM-Authentifizierungsbibliothek von Amazon MSK zu verwenden, konfigurieren Sie die EMR Serverless-Anwendung mit VPC-Zugriff. Ihre Subnetze müssen über Internetzugang verfügen und ein NAT-Gateway verwenden, um auf die Maven-Abhängigkeiten zuzugreifen. Weitere Informationen finden Sie unter VPC-Zugriff konfigurieren. Die Subnetze müssen über Netzwerkkonnektivität verfügen, um auf den Kafka-Cluster zugreifen zu können. Dies

gilt unabhängig davon, ob Ihr Kafka-Cluster selbst verwaltet wird oder ob Sie Amazon Managed Managed Streaming for Apache Kafka verwenden.

Verwaltung von Streaming-Job-Protokollen

Streaming-Jobs unterstützen die Protokollrotation für Spark-Anwendungsprotokolle und -Ereignisprotokolle sowie die Protokollkomprimierung für Spark-Ereignisprotokolle. Dies hilft Ihnen, Ihre Ressourcen effektiv zu verwalten.

Rotation protokollieren

Streaming-Jobs unterstützen die Protokollrotation für Spark-Anwendungsprotokolle und Ereignisprotokolle. Die Protokollrotation verhindert, dass lange Streaming-Jobs große Protokolldateien generieren, die Ihren gesamten verfügbaren Speicherplatz beanspruchen könnten. Die Protokollrotation hilft Ihnen, Festplattenspeicher zu sparen, und verhindert Jobfehler, die auf zu wenig Speicherplatz zurückzuführen sind. Weitere Informationen finden Sie unter Rotation von Protokollen.

Verdichtung von Protokollen

Streaming-Jobs unterstützen auch die Protokollkomprimierung für Spark-Ereignisprotokolle, sofern verwaltete Protokollierung verfügbar ist. Weitere Informationen zur verwalteten Protokollierung finden Sie unter Protokollierung mit verwaltetem Speicher. Streaming-Jobs können lange laufen, und die Menge an Ereignisdaten kann sich im Laufe der Zeit ansammeln und die Größe der Protokolldateien erheblich erhöhen. Der Spark History Server liest diese Ereignisse und lädt sie in den Speicher für die Benutzeroberfläche der Spark-Anwendung. Dieser Prozess kann hohe Latenzen und Kosten verursachen, insbesondere wenn die in Amazon S3 gespeicherten Ereignisprotokolle sehr umfangreich sind.

Durch die Protokollkomprimierung wird die Größe des Ereignisprotokolls reduziert, sodass der Spark History Server zu keinem Zeitpunkt mehr als 1 GB an Ereignisprotokollen laden muss. Weitere Informationen finden Sie unter Monitoring and Instrumentation in der Apache Spark-Dokumentation.

Verwenden von Spark-Konfigurationen bei der Ausführung von EMR Serverless-Jobs

Sie können Spark-Jobs in einer Anwendung ausführen, bei der der `type` Parameter auf gesetzt ist`SPARK`. Jobs müssen mit der Spark-Version kompatibel sein, die mit der Amazon EMR-Release-

Version kompatibel ist. Wenn Sie beispielsweise Jobs mit Amazon EMR Version 6.6.0 ausführen, muss Ihr Job mit Apache Spark 3.2.0 kompatibel sein. Informationen zu den Anwendungsversionen der einzelnen Versionen finden Sie unter. Serverlose Release-Versionen von Amazon EMR

Spark-Job-Parameter

Wenn Sie die `StartJobRun`API verwenden, um einen Spark-Job auszuführen, geben Sie die folgenden Parameter an.

Erforderliche Parameter

- Runtime-Rolle für Spark-Jobs
- Spark-Job-Treiberparameter
- Parameter zum Überschreiben der Spark-Konfiguration
- Optimierung der dynamischen Ressourcenzuweisung in Spark

Runtime-Rolle für Spark-Jobs

Verwenden Sie diese Option **`executionRoleArn`**, um den ARN für die IAM-Rolle anzugeben, die Ihre Anwendung zur Ausführung von Spark-Jobs verwendet. Diese Rolle muss die folgenden Berechtigungen enthalten:

- Lesen Sie aus S3-Buckets oder anderen Datenquellen, in denen sich Ihre Daten befinden
- Lesen Sie aus S3-Buckets oder -Präfixen, in denen sich Ihr PySpark Skript oder Ihre JAR-Datei befindet
- Schreiben Sie in S3-Buckets, in die Sie Ihre endgültige Ausgabe schreiben möchten
- Schreiben Sie Protokolle in einen S3-Bucket oder ein Präfix, das Folgendes angibt `S3MonitoringConfiguration`
- Zugriff auf KMS-Schlüssel, wenn Sie KMS-Schlüssel zum Verschlüsseln von Daten in Ihrem S3-Bucket verwenden
- Zugriff auf den AWS Glue-Datenkatalog, wenn Sie SparkSQL verwenden

Wenn Ihr Spark-Job Daten in oder aus anderen Datenquellen liest oder schreibt, geben Sie die entsprechenden Berechtigungen in dieser IAM-Rolle an. Wenn Sie diese Berechtigungen nicht für die IAM-Rolle bereitstellen, schlägt der Job möglicherweise fehl. Weitere Informationen finden Sie unter Job-Runtime-Rollen für Amazon EMR Serverless und Speichern von Protokollen.

Spark-Job-Treiberparameter

Wird verwendet **`jobDriver`**, um Eingaben für den Job bereitzustellen. Der Job-Treiber-Parameter akzeptiert nur einen Wert für den Auftragstyp, den Sie ausführen möchten. Für einen Spark-Job lautet der Parameterwert`sparkSubmit`. Sie können diesen Jobtyp verwenden, um Scala PySpark, Java und alle anderen unterstützten Jobs über Spark Submit auszuführen. Spark-Jobs haben die folgenden Parameter:

- **`sparkSubmitParameters`**— Dies sind die zusätzlichen Spark-Parameter, die Sie an den Job senden möchten. Verwenden Sie diesen Parameter, um Spark-Standardeigenschaften wie Treiberspeicher oder Anzahl der Executoren zu überschreiben, wie sie in den Argumenten `--conf` oder `--class` definiert sind.
- **`entryPointArguments`**— Dies ist eine Reihe von Argumenten, die Sie an Ihre JAR- oder Python-Hauptdatei übergeben möchten. Sie sollten das Lesen dieser Parameter mit Ihrem Einstiegspunkt-Code regeln. Trennen Sie jedes Argument im Array durch ein Komma.
- **`entryPoint`**— Dies ist der Verweis in Amazon S3 auf die JAR- oder Python-Hauptdatei, die Sie ausführen möchten. Wenn Sie ein Scala- oder Java-JAR ausführen, geben Sie die Haupteintragsklasse `SparkSubmitParameters` unter Verwendung des `--class` Arguments an.

Weitere Informationen finden Sie unter Anwendungen mit spark-submit starten.

Parameter zum Überschreiben der Spark-Konfiguration

Wird verwendet **`configurationOverrides`**, um die Konfigurationseigenschaften auf Überwachungs- und Anwendungsebene zu überschreiben. Dieser Parameter akzeptiert ein JSON-Objekt mit den folgenden zwei Feldern:

- **`monitoringConfiguration`**- Verwenden Sie dieses Feld, um die Amazon S3 S3-URL (`s3MonitoringConfiguration`) anzugeben, unter der der EMR Serverless-Job die Protokolle Ihres Spark-Jobs speichern soll. Stellen Sie sicher, dass Sie diesen Bucket mit demselben Bucket erstellt haben AWS-Konto , der Ihre Anwendung hostet, und mit demselben Bucket, in AWS-Region dem Ihr Job ausgeführt wird.
- **`applicationConfiguration`**— Um die Standardkonfigurationen für Anwendungen zu überschreiben, können Sie in diesem Feld ein Konfigurationsobjekt angeben. Sie können eine Syntax-Kurznotation verwenden, um die Konfiguration anzugeben oder Sie können auf die Konfiguration in einer JSON-Datei zu verweisen. Konfigurationsobjekte bestehen aus einer Klassifizierung, Eigenschaften und optionalen verschachtelten Konfigurationen. Eigenschaften

bestehen aus den Einstellungen, die Sie in dieser Datei überschreiben möchten. Sie können mehrere Klassifizierungen für mehrere Anwendungen in einem einzigen JSON-Objekt angeben.

> **Note**
>
> Die verfügbaren Konfigurationsklassifizierungen variieren je nach EMR Serverless-Version. Zum Beispiel Klassifizierungen für benutzerdefiniertes Log4j `spark-driver-log4j2` und `spark-executor-log4j2` sind nur mit Versionen 6.8.0 und höher verfügbar.

Wenn Sie dieselbe Konfiguration in einer Anwendungsüberschreibung und in Spark-Übergabeparametern verwenden, haben die Spark-Submit-Parameter Priorität. Konfigurationen haben folgende Priorität, von der höchsten zur niedrigsten:

- Konfiguration, die EMR Serverless bei der Erstellung bereitstellt. `SparkSession`
- Konfiguration, die Sie als Teil des Arguments `sparkSubmitParameters` angeben. `--conf`
- Die Konfiguration, die Sie als Teil Ihrer Anwendung angeben, hat Vorrang vor dem Start eines Jobs.
- Konfiguration, die Sie bei der Erstellung einer Anwendung `runtimeConfiguration` als Teil Ihrer angeben.
- Optimierte Konfigurationen, die Amazon EMR für die Version verwendet.
- Open-Source-Standardkonfigurationen für die Anwendung.

Weitere Informationen zum Deklarieren von Konfigurationen auf Anwendungsebene und zum Überschreiben von Konfigurationen während der Auftragsausführung finden Sie unter. Standardanwendungskonfiguration für EMR Serverless

Optimierung der dynamischen Ressourcenzuweisung in Spark

Wird verwendet`dynamicAllocationOptimization`, um die Ressourcennutzung in EMR Serverless zu optimieren. Wenn Sie diese Eigenschaft `true` in Ihrer Spark-Konfigurationsklassifizierung auf festlegen, wird EMR Serverless angewiesen, die Ressourcenzuweisung für Executoren zu optimieren, um die Rate, mit der Spark Executoren anfordert und storniert, besser an die Rate anzupassen, mit der EMR Serverless Worker erstellt und freigibt. Auf diese Weise kann EMR Serverless Workers stufenübergreifend optimaler wiederverwenden, was zu niedrigeren Kosten führt, wenn Jobs mit mehreren Stufen ausgeführt werden, während die gleiche Leistung beibehalten wird.

Diese Eigenschaft ist in allen Amazon EMR-Release-Versionen verfügbar.

Im Folgenden finden Sie ein Beispiel für eine Konfigurationsklassifizierung mit`dynamicAllocationOptimization`.

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

Beachten Sie Folgendes, wenn Sie die dynamische Allokationsoptimierung verwenden:

- Diese Optimierung ist für die Spark-Jobs verfügbar, für die Sie die dynamische Ressourcenzuweisung aktiviert haben.
- Um eine optimale Kosteneffizienz zu erzielen, empfehlen wir die Konfiguration einer oberen Skalierungsgrenze für Mitarbeiter, wobei je nach Arbeitslast entweder die Einstellung für die maximale Kapazität auf Auftragsebene `spark.dynamicAllocation.maxExecutors` oder die Einstellung für die maximale Kapazität auf Anwendungsebene verwendet wird.
- Möglicherweise stellen Sie bei einfacheren Aufträgen keine Kostenverbesserung fest. Wenn Ihr Job beispielsweise auf einem kleinen Datensatz ausgeführt wird oder die Ausführung in einer Phase abgeschlossen wird, benötigt Spark möglicherweise keine größere Anzahl von Executoren oder mehrere Skalierungsereignisse.
- Bei Jobs mit einer Sequenz aus einer großen Phase, kleineren Phasen und dann wieder einer großen Phase kann es zu einer Regression der Job-Laufzeit kommen. Da EMR Serverless Ressourcen effizienter nutzt, kann dies dazu führen, dass für größere Phasen weniger Mitarbeiter zur Verfügung stehen, was zu einer längeren Laufzeit führt.

Eigenschaften von Spark-Jobs

In der folgenden Tabelle sind optionale Spark-Eigenschaften und ihre Standardwerte aufgeführt, die Sie überschreiben können, wenn Sie einen Spark-Job einreichen.

Optionale Spark-Eigenschaften und Standardwerte

Key (Schlüssel)	Description	Standardwert
`spark.archives`	Eine durch Kommas getrennte Liste von Archiven, die Spark in das Arbeitsverzeichnis jedes Executors extrahiert. Zu den unterstützten Dateitypen gehören `.jar`, und. `.tar.gz` `.tgz` `.zip` Um den zu extrahierenden Verzeichn isnamen anzugeben, fügen Sie # nach dem Namen der zu extrahierenden Datei ein. Beispiel, `file.zip#directory` .	`NULL`
`spark.authenticate`	Option, die die Authentif izierung der internen Verbindungen von Spark aktiviert.	`TRUE`
`spark.driver.cores`	Die Anzahl der Kerne, die der Treiber verwendet.	4
`spark.driver.extraJavaOptions`	Zusätzliche Java-Optionen für den Spark-Treiber.	`NULL`
`spark.driver.memory`	Die Menge an Speicher, die der Treiber verwendet.	14 G
`spark.dynamicAllocation.enabled`	Option, die die dynamisch e Ressourcenzuweisun g aktiviert. Diese Option skaliert die Anzahl der bei der Anwendung registrierten	`TRUE`

Key (Schlüssel)	Description	Standardwert
	Executoren je nach Arbeitslast nach oben oder unten.	
`spark.dynamicAllocation.executorIdleTimeout`	Die Zeitspanne, für die ein Executor inaktiv bleiben kann, bevor Spark ihn entfernt. Dies gilt nur, wenn Sie die dynamische Zuweisung aktivieren.	60er Jahre
`spark.dynamicAllocation.initialExecutors`	Die anfängliche Anzahl von Executoren, die ausgeführt werden sollen, wenn Sie die dynamische Zuweisung aktivieren.	`3`
`spark.dynamicAllocation.maxExecutors`	Die Obergrenze für die Anzahl der Executoren, wenn Sie die dynamische Zuweisung aktivieren.	Für 6.10.0 und höher `infinity` Für 6.9.0 und niedriger `100`
`spark.dynamicAllocation.minExecutors`	Die Untergrenze für die Anzahl der Executoren, wenn Sie die dynamische Zuweisung aktivieren.	`0`
`spark.emr-serverless.allocation.batch.size`	Die Anzahl der Container, die in jedem Zyklus der Executor-Zuweisung angefordert werden sollen. Zwischen den einzelnen Zuweisungszyklen besteht eine Lücke von einer Sekunde.	20
`spark.emr-serverless.driver.disk`	Die Spark-Treiberdiskette.	20G

Key (Schlüssel)	Description	Standardwert
`spark.emr-serverless.driverEnv.` *[KEY]*	Option, die dem Spark-Treiber Umgebungsvariablen hinzufügt.	NULL
`spark.emr-serverless.executor.disk`	Die Spark-Executor-Diskette.	20G
`spark.emr-serverless.memoryOverheadFactor`	Legt den Speicheraufwand fest, der dem Speicher des Treiber- und Executor-Containers hinzugefügt werden soll.	0.1
`spark.emr-serverless.driver.disk.type`	Der Festplattentyp, der an den Spark-Treiber angeschlossen ist.	Standard
`spark.emr-serverless.executor.disk.type`	Der Festplattentyp, der an Spark-Executoren angeschlossen ist.	Standard
`spark.executor.cores`	Die Anzahl der Kerne, die jeder Executor verwendet.	4
`spark.executor.extraJavaOptions`	Zusätzliche Java-Optionen für den Spark-Executor.	NULL
`spark.executor.instances`	Die Anzahl der zuzuweisenden Spark-Executor-Container.	3
`spark.executor.memory`	Die Menge an Speicher, die jeder Executor verwendet.	14 G
`spark.executorEnv.` *[KEY]*	Option, die Umgebungsvariablen zu den Spark-Executoren hinzufügt.	NULL

Key (Schlüssel)	Description	Standardwert
`spark.files`	Eine durch Kommas getrennte Liste von Dateien, die in das Arbeitsverzeichnis jedes Executors verschoben werden sollen. Sie können im Executor mit auf die Dateipfade dieser Dateien zugreifen. `SparkFiles.get( fileName)`	`NULL`
`spark.hadoop.hive.metastore.client.factory.class`	Die Hive-Metastore-Implementierungsklasse.	`NULL`
`spark.jars`	Zusätzliche JAR-Dateien, die dem Laufzeit-Klassenpfad des Treibers und der Executoren hinzugefügt werden sollen.	`NULL`
`spark.network.crypto.enabled`	Option, die die AES-basierte RPC-Verschlüsselung aktiviert. Dazu gehört das in Spark 2.2.0 hinzugefügte Authentifizierungsprotokoll.	`FALSE`
`spark.sql.warehouse.dir`	Der Standardspeicherort für verwaltete Datenbanken und Tabellen.	Der Wert von `$PWD/spark-warehouse`
`spark.submit.pyFiles`	Eine kommagetrennte Liste von `.zip`,, oder `.py` Dateien `.egg`, die in die `PYTHONPATH` für Python-Apps eingefügt werden sollen.	`NULL`

In der folgenden Tabelle sind die Standard-Submit-Parameter von Spark aufgeführt.

Standard-Submit-Parameter von Spark

Key (Schlüssel)	Description	Standardwert
`archives`	Eine durch Kommas getrennte Liste von Archiven, die Spark in das Arbeitsverzeichnis jedes Executors extrahiert.	`NULL`
`class`	Die Hauptklasse der Anwendung (für Java- und Scala-Apps).	`NULL`
`conf`	Eine beliebige Spark-Kon figurationseigenschaft.	`NULL`
`driver-cores`	Die Anzahl der Kerne, die der Treiber verwendet.	4
`driver-memory`	Die Menge an Speicher, die der Treiber verwendet.	14 G
`executor-cores`	Die Anzahl der Kerne, die jeder Executor verwendet.	4
`executor-memory`	Die Menge an Speicher, die der Executor verwendet.	14 G
`files`	Eine durch Kommas getrennte Liste von Dateien, die im Arbeitsverzeichnis jedes Executors abgelegt werden sollen. Sie können im Executor mit auf die Dateipfade dieser Dateien zugreifen. `SparkFile s.get(` *`fileName`*`)`	`NULL`

Key (Schlüssel)	Description	Standardwert
`jars`	Eine durch Kommas getrennte Liste von JAR-Dateien, die in die Klassenpfade des Treibers und des Executors aufgenommen werden sollen.	`NULL`
`num-executors`	Die Anzahl der Executors, die gestartet werden sollen.	3
`py-files`	Eine kommagetrennte Liste von `.zip`,, oder `.py` Dateien `.egg`, die in Python-Apps platziert werden `PYTHONPATH` sollen.	`NULL`
`verbose`	Option, die zusätzliche Debug-Ausgaben aktiviert.	`NULL`

Bewährte Methoden für die Konfiguration von Ressourcen

Konfiguration von Treiber- und Executor-Ressourcen über die API StartJobRun

> **Note**
>
> Spark-Treiber- und Executor-Cores sowie Speichereigenschaften müssen, sofern angegeben, direkt in der API-Anfrage angegeben werden. StartJobRun

Wenn Sie Ihre Ressourcen auf diese Weise konfigurieren, wird sichergestellt, dass EMR Serverless die richtigen Ressourcen zuweisen kann, bevor der Job ausgeführt wird. Dies steht im Gegensatz zu den im Benutzerskript bereitgestellten Einstellungen, z. B. in der .py- oder .jar-Datei, die zu spät ausgewertet werden, da Treiber- und Executor-Worker manchmal vorab bereitgestellt werden, bevor die Skriptausführung beginnt. Es gibt zwei unterstützte Methoden, um diese Ressourcen bei der Auftragsübermittlung zu konfigurieren:

Option 1: Verwenden sparkSubmitParameters

```
"jobDriver": {
 "sparkSubmit": {
    "entryPoint": "s3://your-script-path.py",
    "sparkSubmitParameters": "–conf spark.driver.memory=4g \
    –conf spark.driver.cores=2 \
    –conf spark.executor.memory=8g \
    –conf spark.executor.cores=4"
  }
 }
```

Option 2: Verwenden Sie ConfigurationOverrides für die Spark-Defaults-Klassifizierung

```
"configurationOverrides": {
 "applicationConfiguration": [
 {
 "classification": "spark-defaults",
 "properties": {
     "spark.driver.memory": "4g",
     "spark.driver.cores": "2",
     "spark.executor.memory": "8g",
     "spark.executor.cores": "4"
      }
    }
  ]
 }
```

Spark-Beispiele

Das folgende Beispiel zeigt, wie die `StartJobRun` API verwendet wird, um ein Python-Skript auszuführen. Ein end-to-end Tutorial, das dieses Beispiel verwendet, finden Sie unterErste Schritte mit Amazon EMR Serverless. Weitere Beispiele für die Ausführung von PySpark Jobs und das Hinzufügen von Python-Abhängigkeiten finden Sie im EMR Serverless GitHub Samples-Repository.

```
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://amzn-s3-demo-destination-bucket/
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"
        }
    }'
```

Das folgende Beispiel zeigt, wie Sie die `StartJobRun` API verwenden, um ein Spark-JAR auszuführen.

```
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"
        }
    }'
```

Verwenden von Hive-Konfigurationen bei der Ausführung von EMR Serverless-Jobs

Sie können Hive-Jobs in einer Anwendung ausführen, deren `type` Parameter auf gesetzt ist. `HIVE` Jobs müssen mit der Hive-Version kompatibel sein, die mit der Amazon EMR-Release-Version kompatibel ist. Wenn Sie beispielsweise Jobs in einer Anwendung mit Amazon EMR Version 6.6.0 ausführen, muss Ihr Job mit Apache Hive 3.1.2 kompatibel sein. Informationen zu den Anwendungsversionen der einzelnen Versionen finden Sie unter. Serverlose Release-Versionen von Amazon EMR

Hive-Auftragsparameter

Wenn Sie die `StartJobRun`API verwenden, um einen Hive-Job auszuführen, geben Sie die folgenden Parameter an.

Erforderliche Parameter

Laufzeitrolle für Hive-Jobs

Geben **`executionRoleArn`**Sie hier den ARN für die IAM-Rolle an, die Ihre Anwendung zur Ausführung von Hive-Jobs verwendet. Diese Rolle muss die folgenden Berechtigungen enthalten:

- Lesen Sie aus S3-Buckets oder anderen Datenquellen, in denen sich Ihre Daten befinden
- Lesen Sie aus S3-Buckets oder -Präfixen, in denen sich Ihre Hive-Abfragedatei und Ihre Init-Abfragedatei befinden
- Lesen und Schreiben in S3-Buckets, in denen sich Ihr Hive Scratch-Verzeichnis und Ihr Hive Metastore-Warehouse-Verzeichnis befinden
- Schreiben Sie in S3-Buckets, in die Sie Ihre endgültige Ausgabe schreiben möchten
- Schreiben Sie Protokolle in einen S3-Bucket oder ein Präfix, das Folgendes angibt `S3MonitoringConfiguration`
- Zugriff auf KMS-Schlüssel, wenn Sie KMS-Schlüssel zum Verschlüsseln von Daten in Ihrem S3-Bucket verwenden
- Zugriff auf den AWS Glue-Datenkatalog

Wenn Ihr Hive-Job Daten in oder aus anderen Datenquellen liest oder schreibt, geben Sie die entsprechenden Berechtigungen in dieser IAM-Rolle an. Wenn Sie diese Berechtigungen nicht für die IAM-Rolle bereitstellen, schlägt Ihr Job möglicherweise fehl. Weitere Informationen finden Sie unter Job-Runtime-Rollen für Amazon EMR Serverless.

Hive-Job-Treiberparameter

Wird verwendet **`jobDriver`**, um Eingaben für den Job bereitzustellen. Der Job-Treiber-Parameter akzeptiert nur einen Wert für den Auftragstyp, den Sie ausführen möchten. Wenn Sie `hive` als Jobtyp angeben, übergibt EMR Serverless eine Hive-Abfrage an den Parameter. `jobDriver` Hive-Jobs haben die folgenden Parameter:

- **`query`**— Dies ist der Verweis in Amazon S3 auf die Hive-Abfragedatei, die Sie ausführen möchten.
- **`parameters`**— Dies sind die zusätzlichen Hive-Konfigurationseigenschaften, die Sie überschreiben möchten. Um Eigenschaften zu überschreiben, übergeben Sie sie an diesen

Parameter als`--hiveconf property=value`. Um Variablen zu überschreiben, übergeben Sie sie an diesen Parameter als`--hivevar key=value`.

- **`initQueryFile`**— Dies ist die Init-Hive-Abfragedatei. Hive führt diese Datei vor Ihrer Abfrage aus und kann sie verwenden, um Tabellen zu initialisieren.

Parameter zum Überschreiben der Hive-Konfiguration

Wird verwendet **`configurationOverrides`**, um Konfigurationseigenschaften auf Überwachungs- und Anwendungsebene zu überschreiben. Dieser Parameter akzeptiert ein JSON-Objekt mit den folgenden zwei Feldern:

- **`monitoringConfiguration`**— Verwenden Sie dieses Feld, um die Amazon S3 S3-URL (`s3MonitoringConfiguration`) anzugeben, unter der der EMR Serverless-Job die Protokolle Ihres Hive-Jobs speichern soll. Stellen Sie sicher, dass Sie diesen Bucket mit demselben Bucket erstellen AWS-Konto , der Ihre Anwendung hostet, und mit demselben Bucket, in AWS-Region dem Ihr Job ausgeführt wird.
- **`applicationConfiguration`**— Sie können in diesem Feld ein Konfigurationsobjekt angeben, um die Standardkonfigurationen für Anwendungen zu überschreiben. Sie können eine Syntax-Kurznotation verwenden, um die Konfiguration anzugeben oder Sie können auf die Konfiguration in einer JSON-Datei zu verweisen. Konfigurationsobjekte bestehen aus einer Klassifizierung, Eigenschaften und optionalen verschachtelten Konfigurationen. Eigenschaften bestehen aus den Einstellungen, die Sie in dieser Datei überschreiben möchten. Sie können mehrere Klassifizierungen für mehrere Anwendungen in einem einzigen JSON-Objekt angeben.

> **Note**
>
> Die verfügbaren Konfigurationsklassifizierungen variieren je nach EMR Serverless-Version. Zum Beispiel Klassifizierungen für benutzerdefiniertes Log4j `spark-driver-log4j2` und `spark-executor-log4j2` sind nur mit Versionen 6.8.0 und höher verfügbar.

Wenn Sie dieselbe Konfiguration in einer Anwendungsüberschreibung und in Hive-Parametern übergeben, haben die Hive-Parameter Priorität. In der folgenden Liste werden die Konfigurationen von der höchsten Priorität zur niedrigsten Priorität geordnet.

- Konfiguration, die Sie als Teil der Hive-Parameter mit `--hiveconf property=value` angeben.

- Die Konfiguration, die Sie als Teil Ihrer Anwendung angeben, hat Vorrang, wenn Sie einen Job starten.
- Konfiguration, die Sie bei der Erstellung einer Anwendung `runtimeConfiguration` als Teil Ihrer angeben.
- Optimierte Konfigurationen, die Amazon EMR der Version zuweist.
- Standard-Open-Source-Konfigurationen für die Anwendung.

Weitere Informationen zum Deklarieren von Konfigurationen auf Anwendungsebene und zum Überschreiben von Konfigurationen während der Jobausführung finden Sie unter. Standardanwendungskonfiguration für EMR Serverless

Eigenschaften von Hive-Jobs

In der folgenden Tabelle sind die obligatorischen Eigenschaften aufgeführt, die konfiguriert werden, wenn Sie einen Hive-Job einreichen.

Obligatorische Eigenschaften von Hive-Jobs

Einstellung	Description
`hive.exec.scratchdir`	Der Amazon S3 S3-Speicherort, an dem EMR Serverless während der Hive-Auftragsausfü hrung temporäre Dateien erstellt.
`hive.metastore.warehouse.dir`	Der Amazon S3 S3-Speicherort von Datenbank en für verwaltete Tabellen in Hive.

In der folgenden Tabelle sind die optionalen Hive-Eigenschaften und ihre Standardwerte aufgeführt, die Sie überschreiben können, wenn Sie einen Hive-Job einreichen.

Optionale Hive-Eigenschaften und Standardwerte

Einstellung	Description	Standardwert
`fs.s3.customAWSCre dentialsProvider`	Der Anbieter AWS für Anmeldeinformationen, den Sie verwenden möchten.	com.amazonaws.auth .DEFAULT AWSCredentials ProviderChain

Einstellung	Description	Standardwert
`fs.s3a.aws.credentials.provider`	Der Anbieter AWS für Anmeldeinformationen, den Sie mit einem S3A-Dateisystem verwenden möchten.	com.amazonaws.auth.DEFAULT AWSCredentials ProviderChain
`hive.auto.convert.join`	Option, die die automatische Konvertierung von allgemeinen Joins in Mapjoin aktiviert, basierend auf der Größe der Eingabedatei.	`TRUE`
`hive.auto.convert.join.noconditionaltask`	Option, die die Optimierung aktiviert, wenn Hive einen gemeinsamen Join basierend auf der Größe der Eingabedatei in einen Mapjoin konvertiert.	`TRUE`
`hive.auto.convert.join.noconditionaltask.size`	Ein Join wird direkt in einen Mapjoin unter dieser Größe konvertiert.	Der optimale Wert wird auf der Grundlage des Tez-Tasks peichers berechnet
`hive.cbo.enable`	Option, die kostenbasierte Optimierungen mit dem Calcite-Framework aktiviert.	`TRUE`
`hive.cli.tez.session.async`	Option zum Starten einer Tez-Hintergrundsitzung, während Ihre Hive-Abfrage kompiliert wird. Wenn diese Option auf gesetzt ist`false`, wird Tez AM gestartet, nachdem Ihre Hive-Abfrage kompiliert wurde.	`TRUE`

Einstellung	Description	Standardwert
`hive.compute.query.using.stats`	Option, die Hive aktiviert, um bestimmte Abfragen mit im Metastore gespeicherten Statistiken zu beantworten. Stellen `hive.stats.autogather` Sie für grundlegende Statistiken auf ein. `TRUE` Für eine umfassendere Sammlung von Abfragen führen Sie den Befehl aus`analyze table queries`.	`TRUE`
`hive.default.fileformat`	Das Standarddateiformat für `CREATE TABLE` Anweisungen. Sie können dies explizit überschreiben, wenn Sie es `STORED AS [FORMAT]` in Ihrem `CREATE TABLE` Befehl angeben.	`TEXTFILE`
`hive.driver.cores`	Die Anzahl der Kerne, die für den Hive-Treiberprozess verwendet werden sollen.	2
`hive.driver.disk`	Die Festplattengröße für den Hive-Treiber.	20G
`hive.driver.disk.type`	Der Festplattentyp für den Hive-Treiber.	Standard
`hive.tez.disk.type`	Die Festplattengröße für die Tez-Arbeiter.	Standard

Einstellung	Description	Standardwert
`hive.driver.memory`	Die Menge an Arbeitsspeicher, die pro Hive-Treiberprozess verwendet werden soll. Die Hive CLI und der Tez Application Master teilen sich diesen Speicher zu gleichen Teilen mit 20% Headroom.	6 G
`hive.emr-serverless.launch.env.[KEY]`	Option zum Festlegen der *KEY* Umgebungsvariablen in allen Hive-spezifischen Prozessen, z. B. Ihrem Hive-Treiber, Tez AM und der Tez-Aufgabe.	
`hive.exec.dynamic.partition`	Optionen, die dynamische Partitionen in DML/DDL aktivieren.	`TRUE`
`hive.exec.dynamic.partition.mode`	Option, die angibt, ob Sie den strikten Modus oder den nicht strikten Modus verwenden möchten. Geben Sie im strikten Modus mindestens eine statische Partition an, falls Sie versehentlich alle Partitionen überschreiben. Im nicht-strikten Modus dürfen alle Partitionen dynamisch sein.	`strict`
`hive.exec.max.dynamic.partitions`	Die maximale Anzahl dynamischer Partitionen, die Hive insgesamt erstellt.	1000

Einstellung	Description	Standardwert
`hive.exec.max.dynamic.partitions.pernode`	Maximale Anzahl dynamischer Partitionen, die Hive in jedem Mapper- und Reducer-Knoten erstellt.	100
`hive.exec.orc.split.strategy`	Erwartet einen der folgenden Werte:`BI`,, `ETL` oder. `HYBRID` Dies ist keine Konfiguration auf Benutzerebene. `BI`gibt an, dass Sie weniger Zeit mit der Split-Generierung als mit der Ausführung von Abfragen verbringen möchten. `ETL`gibt an, dass Sie mehr Zeit mit der Split-Generierung verbringen möchten. `HYBRID`gibt eine Auswahl der vorherigen Strategien auf der Grundlage von Heuristiken an.	`HYBRID`
`hive.exec.reducers.bytes.per.reducer`	Die Größe pro Reduzierstück. Die Standardeinstellung ist 256 MB. Wenn die Eingabegröße 1 G ist, verwendet der Job 4 Reduzierungen.	256000000
`hive.exec.reducers.max`	Die maximale Anzahl von Reduzierstücken.	256

Einstellung	Description	Standardwert
`hive.exec.stagingdir`	Der Name des Verzeichnisses, in dem temporäre Dateien gespeichert werden, die Hive innerhalb von Tabellenpositionen und in dem in der Eigenschaft angegebenen Speicherort des Scratch-Verzeichnisses erstellt. `hive.exec.scratchdir`	`.hive-staging`
`hive.fetch.task.conversion`	Erwartet einen der folgenden Werte:`NONE`,`MINIMAL`, oder`MORE`. Hive kann ausgewählte Abfragen in eine einzelne `FETCH` Aufgabe konvertieren. Dadurch wird die Latenz minimiert.	`MORE`
`hive.groupby.position.alias`	Option, die Hive veranlasst, in `GROUP BY` Anweisungen einen Alias für die Spaltenposition zu verwenden.	`FALSE`
`hive.input.format`	Das Standardeingabeformat. Stellen Sie es auf ein, `HiveInputFormat` wenn Sie Probleme mit habenCombineHiveInputFormat .	`org.apache.hadoop.hive.ql.io.CombineHiveInputFormat`
`hive.log.explain.output`	Option, die Erläuterungen zur erweiterten Ausgabe für alle Abfragen in Ihrem Hive-Protokoll aktiviert.	`FALSE`

Einstellung	Description	Standardwert
`hive.log.level`	Die Hive-Protokollierungsebene.	`INFO`
`hive.mapred.reduce.tasks.speculative.execution`	Option, die die spekulative Markteinführung von Reducern aktiviert. Wird nur mit Amazon EMR 6.10.x und niedriger unterstützt.	`TRUE`
`hive.max-task-containers`	Die maximale Anzahl gleichzeitiger Container. Der konfigurierte Mapper-Speicher wird mit diesem Wert multipliziert, um den verfügbaren Speicher zu ermitteln, der von der Aufteilung der Berechnungen und der Abmeldung von Aufgaben genutzt wird.	1000
`hive.merge.mapfiles`	Option, die bewirkt, dass kleine Dateien am Ende eines Auftrags, der nur für die Zuordnung vorgesehen ist, zusammengeführt werden.	`TRUE`
`hive.merge.size.per.task`	Die Größe der zusammengeführten Dateien am Ende des Jobs.	256000000
`hive.merge.tezfiles`	Option, die das Zusammenführen kleiner Dateien am Ende einer Tez-DAG aktiviert.	`FALSE`

Einstellung	Description	Standardwert
`hive.metastore.client.factory.class`	Der Name der Factory-Klasse, die Objekte erzeugt, die die `IMetaStoreClient` Schnittstelle implementieren.	`com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory`
`hive.metastore.glue.catalogid`	Wenn der AWS Glue-Datenkatalog als Metastore fungiert, aber in einem anderen AWS-Konto als den Jobs ausgeführt wird, die ID des Ortes, AWS-Konto an dem die Jobs ausgeführt werden.	`NULL`
`hive.metastore.uris`	Der Thrift-URI, den der Metastore-Client verwendet, um eine Verbindung zum Remote-Metastore herzustellen.	`NULL`
`hive.optimize.ppd`	Option, die den Prädikats-Pushdown aktiviert.	`TRUE`
`hive.optimize.ppd.storage`	Option, die die Übertragung von Prädikaten an Speicherhandler aktiviert.	`TRUE`
`hive.orderby.position.alias`	Option, die Hive veranlasst, in Anweisungen einen Alias für die Spaltenposition zu verwenden. `ORDER BY`	`TRUE`
`hive.prewarm.enabled`	Option, die das Container-Prewarm für Tez aktiviert.	`FALSE`

Einstellung	Description	Standardwert
`hive.prewarm.numcontainers`	Die Anzahl der Behälter, die für Tez vorgewärmt werden sollen.	10
`hive.stats.autogather`	Option, die Hive veranlasst, während des Befehls automatisch grundlegende Statistiken zu sammeln. `INSERT OVERWRITE`	`TRUE`
`hive.stats.fetch.column.stats`	Option, die das Abrufen von Spaltenstatistiken aus dem Metastore deaktiviert. Das Abrufen von Spaltenstatistiken kann teuer sein, wenn die Anzahl der Spalten hoch ist.	`FALSE`
`hive.stats.gather.num.threads`	Die Anzahl der Threads, die die Befehle `partialscan` und `noscan` analyze für partitionierte Tabellen verwenden. Dies gilt nur für implementierte Dateiformate `StatsProvidingRecordReader` (wie ORC).	10
`hive.strict.checks.cartesian.product`	Optionen, die strenge kartesische Join-Prüfungen aktivieren. Bei diesen Prüfungen ist ein kartesisches Produkt (ein Cross-Join) nicht zulässig.	`FALSE`

Einstellung	Description	Standardwert
`hive.strict.checks.type.safety`	Option, mit der strenge Typsicherheitsprüfungen aktiviert und der Vergleich von sowohl `bigint` mit als auch deaktiviert wird. `string` `double`	`TRUE`
`hive.support.quoted.identifiers`	Erwartet einen Wert von `NONE` oder`COLUMN`. `NONE`impliziert, dass nur alphanumerische Zeichen und Unterstriche in Bezeichnern gültig sind. `COLUMN`impliziert, dass Spaltennamen jedes beliebige Zeichen enthalten können.	`COLUMN`
`hive.tez.auto.reducer.parallelism`	Option, die die automatische Reducer-Parallelitätsfunktion von Tez aktiviert. Hive schätzt weiterhin Datengrößen und legt Parallelitätsschätzungen fest. Tez nimmt Stichproben der Ausgabegrößen der Quellscheitelpunkte und passt die Schätzungen zur Laufzeit nach Bedarf an.	`TRUE`
`hive.tez.container.size`	Die Menge an Speicher, die pro Tez-Task-Prozess verwendet werden soll.	6144
`hive.tez.cpu.vcores`	Die Anzahl der Kerne, die für jede Tez-Aufgabe verwendet werden sollen.	2

Einstellung	Description	Standardwert
`hive.tez.disk.size`	Die Festplattengröße für jeden Task-Container.	20G
`hive.tez.input.format`	Das Eingabeformat für die Split-Generierung im Tez AM.	`org.apache.hadoop.hive.ql.io.HiveInputFormat`
`hive.tez.min.partition.factor`	Untere Grenze der Reduzierungen, die Tez festlegt, wenn Sie die automatische Reduzierung der Parallelität aktivieren.	0,25
`hive.vectorized.execution.enabled`	Option, die den vektorisierten Modus der Abfrageausführung aktiviert.	`TRUE`
`hive.vectorized.execution.reduce.enabled`	Option, die den vektorisierten Modus der Reduktionsseite einer Abfrageausführung aktiviert.	`TRUE`
`javax.jdo.option.ConnectionDriverName`	Der Treiberklassenname für einen JDBC-Metastore.	`org.apache.derby.jdbc.EmbeddedDriver`
`javax.jdo.option.ConnectionPassword`	Das mit einer Metastore-Datenbank verknüpfte Passwort.	`NULL`
`javax.jdo.option.ConnectionURL`	Die JDBC-Verbindungszeichenfolge für einen JDBC-Metastore.	`jdbc:derby:;databaseName=metastore_db;create=true`
`javax.jdo.option.ConnectionUserName`	Der mit einer Metastore-Datenbank verknüpfte Benutzername.	`NULL`

Einstellung	Description	Standardwert
`mapreduce.input.fileinputformat.split.maxsize`	Die maximale Größe eines Splits während der Split-Berechnung, wenn Ihr Eingabeformat ist. `org.apache.hadoop.hive.ql.io.CombineHiveInputFormat` Bei Angabe von 0 gibt es keinen Höchstwert.	0
`tez.am.dag.cleanup.on.completion`	Option, die die Bereinigung von Shuffle-Daten aktiviert, wenn DAG abgeschlossen ist.	`TRUE`
`tez.am.emr-serverless.launch.env.[ KEY]`	Option zum Festlegen der *KEY* Umgebungsvariablen im Tez AM-Prozess. Für Tez AM überschreibt dieser Wert den Wert. `hive.emr-serverless.launch.env.[ KEY]`	
`tez.am.log.level`	Die Root-Protokollierungsebene, die EMR Serverless an die primäre Tez-App weitergibt.	`INFO`
`tez.am.sleep.time.before.exit.millis`	EMR Serverless sollte ATS-Ereignisse nach diesem Zeitraum nach der Anfrage zum Herunterfahren von AM weiterleiten.	0

Einstellung	Description	Standardwert
`tez.am.speculation.enabled`	Option, die zum spekulativen Start langsamerer Aufgaben führt. Dies kann dazu beitragen, die Auftragslatenz zu verringern, wenn einige Aufgaben aufgrund fehlerhafter oder langsamer Computer langsamer ausgeführt werden. Wird nur mit Amazon EMR 6.10.x und niedriger unterstützt.	`FALSE`
`tez.am.task.max.failed.attempts`	Die maximale Anzahl von Versuchen, die bei einer bestimmten Aufgabe fehlschlagen können, bevor die Aufgabe fehlschlägt. Bei dieser Zahl werden manuell abgebrochene Versuche nicht mitgezählt.	3
`tez.am.vertex.cleanup.height`	Eine Entfernung, bei der Tez AM die Vertex-Shuffle-Daten löscht, wenn alle abhängigen Scheitelpunkte vollständig sind. Diese Funktion ist ausgeschaltet, wenn der Wert 0 ist. Amazon EMR-Versionen 6.8.0 und höher unterstützen diese Funktion.	0
`tez.client.asynchronous-stop`	Option, die EMR Serverless veranlasst, ATS-Ereignisse zu übertragen, bevor der Hive-Treiber beendet wird.	`FALSE`

Einstellung	Description	Standardwert
`tez.grouping.max-size`	Die obere Größenbeschränkung (in Byte) eines gruppierten Splits. Diese Grenze verhindert übermäßig große Teilungen.	1073741824
`tez.grouping.min-size`	Die untere Größenbeschränkung (in Byte) eines gruppierten Splits. Diese Grenze verhindert zu viele kleine Teilungen.	16777216
`tez.runtime.io.sort.mb`	Die Größe des Soft-Buffers, wenn Tez die Ausgabe sortiert, wird sortiert.	Der optimale Wert wird auf der Grundlage des Tez-Tasks peichers berechnet
`tez.runtime.unordered.output.buffer.size-mb`	Die Größe des zu verwenden den Puffers, wenn Tez nicht direkt auf die Festplatte schreibt.	Der optimale Wert wird auf der Grundlage des Tez-Tasks peichers berechnet

Einstellung	Description	Standardwert
`tez.shuffle-vertex-manager.max-src-fraction`	Der Anteil der Quellaufgaben, die abgeschlossen werden müssen, bevor EMR Serverless alle Aufgaben für den aktuellen Scheitelpunkt plant (im Fall einer `ScatterGather` Verbindung). Die Anzahl der Aufgaben, die für den aktuellen Scheitelpunkt zur Planung bereit sind, skaliert linear zwischen und. `min-fraction` `max-fraction` Dies ist der Standardwert oder`tez.shuffle-vertex-manager.min-src-fraction` , je nachdem, welcher Wert größer ist.	0.75
`tez.shuffle-vertex-manager.min-src-fraction`	Der Anteil der Quellaufgaben, die abgeschlossen werden müssen, bevor EMR Serverless Aufgaben für den aktuellen Scheitelpunkt plant (im Fall einer `ScatterGather` Verbindung).	0,25
`tez.task.emr-serverless.launch.env.[` *KEY*`]`	Option zum Festlegen der *KEY* Umgebungsvariablen im Tez-Taskprozess. Bei Tez-Aufgaben überschreibt dieser Wert den Wert. `hive.emr-serverless.launch.env.[` *KEY*`]`	

Einstellung	Description	Standardwert
`tez.task.log.level`	Die Root-Protokollierungsebene, die EMR Serverless an die Tez-Aufgaben weitergibt.	`INFO`
`tez.yarn.ats.event.flush.timeout.millis`	Die maximale Zeit, die AM warten soll, bis Ereignisse gelöscht werden, bevor das System heruntergefahren wird.	300000

Hive-Jobbeispiele

Das folgende Codebeispiel zeigt, wie eine Hive-Abfrage mit der `StartJobRun` API ausgeführt wird.

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "hive": {
            "query": "s3://amzn-s3-demo-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://amzn-s3-demo-bucket/emr-serverless-hive/
hive/scratch",
                "hive.metastore.warehouse.dir": "s3://amzn-s3-demo-bucket/emr-
serverless-hive/hive/warehouse",
                "hive.driver.cores": "2",
                "hive.driver.memory": "4g",
                "hive.tez.container.size": "4096",
                "hive.tez.cpu.vcores": "1"
            }
        }]
```

```
    }'
```

Weitere Beispiele für die Ausführung von Hive-Jobs finden Sie im EMR Serverless Samples Repository. GitHub

EMR Serverless Job Resilienz

EMR Serverless Releases 7.1.0 und höher bieten Unterstützung für Job-Resiliency, sodass fehlgeschlagene Jobs automatisch ohne manuelles Eingreifen Ihrerseits wiederholt werden. Ein weiterer Vorteil der Job-Resilienz besteht darin, dass EMR Serverless Job-Runs in eine andere Availability Zone (AZ) verschiebt, falls in einer AZ Probleme auftreten.

Um die Ausfallsicherheit für einen Job zu aktivieren, legen Sie die Wiederholungsrichtlinie für Ihren Job fest. Eine Wiederholungsrichtlinie stellt sicher, dass EMR Serverless einen Job automatisch neu startet, falls er zu irgendeinem Zeitpunkt fehlschlägt. Wiederholungsrichtlinien werden für Batch- und Streaming-Jobs unterstützt. Passen Sie die Job-Resilienz also an Ihren Anwendungsfall an. In der folgenden Tabelle werden das Verhalten und die Unterschiede der Job-Resilienz bei Batch- und Streaming-Jobs verglichen.

	Stapelverarbeitungsaufträge (Batch jobs)	Jobs streamen
Standardverhalten	Führt den Job nicht erneut aus.	Versucht immer erneut, den Job auszuführen, da die Anwendung während der Ausführung des Jobs Checkpoints erstellt.
Punkt wiederholen	Batch-Jobs haben keine Checkpoints, daher führt EMR Serverless den Job immer von Anfang an erneut aus.	Streaming-Jobs unterstützen Checkpoints. Konfigurieren Sie daher die Streaming-Abfrage so, dass der Laufzeitstatus und der Fortschritt an einem Checkpoint-Speicherort in Amazon S3 gespeichert werden. EMR Serverless setzt die Auftragsausführung vom

	Stapelverarbeitungsaufträge (Batch jobs)	Jobs streamen
		Checkpoint aus fort. Weitere Informationen finden Sie unter Recovering from failure with Checkpointing in der Apache Spark-Dokumentation.
Höchstzahl an Wiederholungsversuchen	Erlaubt maximal 10 Wiederholungen.	Streaming-Jobs verfügen über eine integrierte Steuerung zur Verhinderung von Datenmissbrauch, sodass die Anwendung die Wiederholung von Aufträgen beendet, wenn sie nach einer Stunde weiterhin fehlschlagen. Die Standardanzahl von Wiederholungen innerhalb einer Stunde beträgt fünf Versuche. Sie können diese Anzahl von Wiederholungen so konfigurieren, dass sie zwischen 1 und 10 liegt. Sie können die Anzahl der maximalen Versuche nicht anpassen. Ein Wert von 1 bedeutet, dass es keine Wiederholungen gibt.

Wenn EMR Serverless versucht, einen Job erneut auszuführen, indexiert es den Job auch mit einer Versuchsnummer, sodass Sie den Lebenszyklus eines Jobs über alle Versuche hinweg verfolgen können.

Verwenden Sie die serverlosen EMR-API-Operationen oder die, AWS CLI um die Job-Resilienz zu ändern oder auf Informationen zur Job-Resilienz zuzugreifen. Weitere Informationen finden Sie im EMR Serverless API-Handbuch.

Standardmäßig führt EMR Serverless Batch-Jobs nicht erneut aus. Um Wiederholungen für Batch-Jobs zu aktivieren, konfigurieren Sie den `maxAttempts` Parameter, wenn Sie eine Batch-Job-Ausführung starten. Der `maxAttempts` Parameter gilt nur für Batch-Jobs. Die Standardeinstellung ist 1, was bedeutet, dass der Job nicht erneut ausgeführt werden soll. Zulässige Werte sind 1 bis einschließlich 10.

Das folgende Beispiel zeigt, wie beim Starten einer Auftragsausführung eine maximale Anzahl von 10 Versuchen angegeben wird.

```
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"
     }
}'
```

EMR Serverless versucht auf unbestimmte Zeit, Streaming-Jobs erneut zu starten, wenn sie fehlschlagen. Um Thrashing aufgrund wiederholter nicht behebbarer Fehler zu verhindern, konfigurieren Sie die Steuerung `maxFailedAttemptsPerHour` zur Verhinderung von Datenübertragungen für die Wiederholung von Streaming-Aufträgen. Mit diesem Parameter können Sie die maximal zulässige Anzahl fehlgeschlagener Versuche innerhalb einer Stunde angeben, bevor EMR Serverless die erneuten Versuche beendet. Die Standardeinstellung ist fünf. Zulässige Werte sind 1 bis einschließlich 10.

```
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"
     }
}'
```

Sie können auch die anderen API-Operationen zum Ausführen von Jobs verwenden, um Informationen über Jobs abzurufen. Verwenden Sie beispielsweise den `attempt` Parameter mit dem `GetJobRun` Vorgang, um Details zu einem bestimmten Auftragsversuch abzurufen. Wenn Sie den `attempt` Parameter nicht angeben, gibt die Operation Informationen über den letzten Versuch zurück.

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

Der `ListJobRunAttempts` Vorgang gibt Informationen über alle Versuche zurück, die sich auf eine Auftragsausführung beziehen.

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

Der `GetDashboardForJobRun` Vorgang erstellt und gibt eine URL zurück, über die auf die Anwendung zugegriffen wird, UIs um einen Job auszuführen. Mit dem `attempt` Parameter können Sie eine URL für einen bestimmten Versuch abrufen. Wenn Sie den `attempt` Parameter nicht angeben, gibt der Vorgang Informationen über den letzten Versuch zurück.

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

Überwachen eines Auftrags mit einer Wiederholungsrichtlinie

Die Unterstützung von Job Resiliency fügt außerdem das neue Event EMR Serverless Job Run Retry hinzu. EMR Serverless veröffentlicht dieses Ereignis bei jeder Wiederholung des Jobs. Sie können

diese Benachrichtigung verwenden, um die Wiederholungen des Jobs nachzuverfolgen. Weitere Informationen zu Veranstaltungen finden Sie unter EventBridge Amazon-Veranstaltungen.

Protokollierung mit Wiederholungsrichtlinie

Jedes Mal, wenn EMR Serverless einen Job erneut versucht, generiert der Versuch seine eigenen Protokolle. Um sicherzustellen, dass EMR Serverless diese Protokolle erfolgreich an Amazon S3 und Amazon senden kann, CloudWatch ohne sie zu überschreiben, fügt EMR Serverless dem Format des S3-Protokollpfads und des CloudWatch Protokollstreamnamens ein Präfix hinzu, das die Versuchsnummer des Auftrags enthält.

Im Folgenden finden Sie ein Beispiel dafür, wie das Format aussieht.

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

Dieses Format stellt sicher, dass EMR Serverless alle Protokolle für jeden Versuch der Ausführung des Auftrags an seinem eigenen dafür vorgesehenen Speicherort in Amazon S3 und veröffentlicht. CloudWatch Weitere Informationen finden Sie unter Speichern von Protokollen.

> **Note**
>
> EMR Serverless verwendet dieses Präfixformat nur für alle Streaming-Jobs und alle Batch-Jobs, für die Wiederholungsversuche aktiviert sind.

Metastore-Konfiguration für EMR Serverless

Ein Hive-Metastore ist ein zentraler Ort, an dem Strukturinformationen zu Ihren Tabellen gespeichert werden, einschließlich Schemas, Partitionsnamen und Datentypen. Mit EMR Serverless können Sie diese Tabellenmetadaten in einem Metastore speichern, der Zugriff auf Ihre Jobs hat.

Sie haben zwei Optionen für einen Hive-Metastore:

- Der AWS Glue-Datenkatalog
- Ein externer Apache Hive-Metastore

Den AWS Glue-Datenkatalog als Metastore verwenden

Sie können Ihre Spark- und Hive-Jobs so konfigurieren, dass sie den AWS Glue-Datenkatalog als Metastore verwenden. Wir empfehlen diese Konfiguration, wenn Sie einen persistenten Metastore oder einen Metastore benötigen, der von verschiedenen Anwendungen, Diensten oder gemeinsam genutzt wird. AWS-Konten Weitere Informationen zum Datenkatalog finden Sie unter Füllen des AWS Glue-Datenkatalogs. Informationen zu den Preisen von AWS Glue finden Sie unter AWS Glue-Preise.

Sie können Ihren EMR Serverless-Job so konfigurieren, dass er den AWS Glue-Datenkatalog entweder in derselben AWS-Konto Anwendung oder in einer anderen verwendet. AWS-Konto

Den AWS Glue-Datenkatalog konfigurieren

Um den Datenkatalog zu konfigurieren, wählen Sie aus, welchen Typ von EMR Serverless-Anwendung Sie verwenden möchten.

Spark

Wenn Sie EMR Studio verwenden, um Ihre Jobs mit EMR Serverless Spark-Anwendungen auszuführen, ist der AWS Glue-Datenkatalog der Standard-Metastore.

Wenn Sie SDKs oder verwenden AWS CLI, stellen Sie die `spark.hadoop.hive.metastore.client.factory.class` Konfiguration `com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory` in den `sparkSubmit` Parametern Ihrer Jobausführung auf ein. Das folgende Beispiel zeigt, wie Sie den Datenkatalog mit dem konfigurieren AWS CLI.

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "sparkSubmit": {
            "entryPoint": "s3://amzn-s3-demo-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"
        }
    }'
```

Alternativ können Sie diese Konfiguration festlegen, wenn Sie `SparkSession` in Ihrem Spark-Code einen neuen erstellen.

```
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

Für EMR Serverless Hive-Anwendungen ist der Datenkatalog der Standard-Metastore. Das heißt, wenn Sie Jobs auf einer EMR Serverless Hive-Anwendung ausführen, zeichnet Hive Metastore-Informationen im Datenkatalog genauso auf wie Ihre Anwendung. AWS-Konto Sie benötigen keine Virtual Private Cloud (VPC), um den Datenkatalog als Metastore zu verwenden.

Um auf die Hive-Metastore-Tabellen zuzugreifen, fügen Sie die erforderlichen AWS Glue-Richtlinien hinzu, die unter Einrichten von IAM-Berechtigungen für Glue beschrieben sind. AWS

Kontoübergreifenden Zugriff für EMR Serverless und AWS Glue Data Catalog konfigurieren

Um den kontenübergreifenden Zugriff für EMR Serverless einzurichten, melden Sie sich zunächst bei folgenden Konten an: AWS-Konten

- `AccountA`— Und AWS-Konto wo Sie eine serverlose EMR-Anwendung erstellt haben.
- `AccountB`— Ein AWS-Konto , der einen AWS Glue-Datenkatalog enthält, auf den Ihr EMR Serverless-Job zugreifen soll.

1. Stellen Sie sicher, dass ein Administrator oder eine andere autorisierte Person eine Ressourcenrichtlinie an den Datenkatalog in `AccountB` anhängt. `AccountB` Diese Richtlinie gewährt `AccountA` spezifische kontoübergreifende Berechtigungen zur Ausführung von Vorgängen mit Ressourcen im `AccountB` Katalog.

   JSON

```
{
  "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:*:123456789012:catalog"
      ],
      "Sid": "AllowGLUEGetdatabase"
    }
  ]
}
```

2. Fügen Sie der Runtime-Rolle EMR Serverless Job eine IAM-Richtlinie hinzu, `AccountA` damit diese Rolle auf Datenkatalogressourcen in zugreifen kann. `AccountB`

   JSON

```
{
```

```
  "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:*:123456789012:catalog"
      ],
      "Sid": "AllowGLUEGetdatabase"
    }
  ]
}
```

3. Starten Sie Ihren Job Run. Dieser Schritt unterscheidet sich je `AccountA` nach EMR Serverless-Anwendungstyp geringfügig.

   Spark

   Übergeben Sie die `spark.hadoop.hive.metastore.glue.catalogid` Eigenschaft an, `sparkSubmitParameters` wie im folgenden Beispiel gezeigt. *`AccountB-catalog-id`*Ersetzen Sie es durch die ID des Datenkatalogs in`AccountB`.

   ```
   aws emr-serverless start-job-run \
   --application-id "application-id" \
   --execution-role-arn "job-role-arn" \
   --job-driver '{
       "sparkSubmit": {
           "entryPoint": "s3://amzn-s3-demo-bucket/scripts/test.py",
   ```

```
         "sparkSubmitParameters": "--conf
 spark.hadoop.hive.metastore.client.factory.class=com.amazonaws.glue.catalog.metastore.A
 --conf spark.hadoop.hive.metastore.glue.catalogid=AccountB-catalog-
id --conf spark.executor.cores=1 --conf spark.executor.memory=1g
 --conf spark.driver.cores=1 --conf spark.driver.memory=1g --conf
 spark.executor.instances=1"
    }
  }' \
--configuration-overrides '{
    "monitoringConfiguration": {
    "s3MonitoringConfiguration": {
    "logUri": "s3://amzn-s3-demo-bucket/logs/"
    }
  }
}'
```

Hive

Stellen Sie die `hive.metastore.glue.catalogid` Eigenschaft in der `hive-site` Klassifizierung ein, wie im folgenden Beispiel gezeigt. *AccountB-catalog-id*Ersetzen Sie es durch die ID des Datenkatalogs in`AccountB`.

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

Überlegungen zur Verwendung des AWS Glue-Datenkatalogs

Sie können Ihren JARs `ADD JAR` Hive-Skripten Hilfsmittel hinzufügen. Weitere Überlegungen finden Sie unter Überlegungen zur Verwendung des AWS Glue-Datenkatalogs.

Verwenden eines externen Hive-Metastores

Sie können Ihre EMR Serverless Spark- und Hive-Jobs so konfigurieren, dass sie eine Verbindung zu einem externen Hive-Metastore wie Amazon Aurora oder Amazon RDS for MySQL herstellen. In diesem Abschnitt wird beschrieben, wie Sie einen Amazon RDS Hive-Metastore einrichten, Ihre VPC konfigurieren und Ihre EMR Serverless-Jobs für die Verwendung eines externen Metastores konfigurieren.

Erstellen Sie einen externen Hive-Metastore

1. Erstellen Sie eine Amazon Virtual Private Cloud (Amazon VPC) mit privaten Subnetzen, indem Sie den Anweisungen unter Erstellen einer VPC folgen.
2. Erstellen Sie Ihre serverlose EMR-Anwendung mit Ihrer neuen Amazon VPC und privaten Subnetzen. Wenn Sie Ihre EMR Serverless-Anwendung mit einer VPC konfigurieren, stellt sie zunächst eine elastic network interface für jedes von Ihnen angegebene Subnetz bereit. Anschließend wird Ihre angegebene Sicherheitsgruppe an diese Netzwerkschnittstelle angehängt. Dadurch erhält Ihre Anwendung die Zugriffskontrolle. Weitere Informationen zur Einrichtung Ihrer VPC finden Sie unter. Konfiguration des VPC-Zugriffs für serverlose EMR-Anwendungen zur Verbindung mit Daten
3. Erstellen Sie eine MySQL- oder Aurora PostgreSQL-Datenbank in einem privaten Subnetz in Ihrer Amazon VPC. Informationen zum Erstellen einer Amazon RDS-Datenbank finden Sie unter Erstellen einer Amazon RDS-DB-Instance.
4. Ändern Sie die Sicherheitsgruppe Ihrer MySQL- oder Aurora-Datenbank, um JDBC-Verbindungen von Ihrer EMR Serverless-Sicherheitsgruppe aus zuzulassen, indem Sie die Schritte unter Ändern einer Amazon RDS-DB-Instance befolgen. Fügen Sie der RDS-Sicherheitsgruppe eine Regel für eingehenden Datenverkehr aus einer Ihrer EMR-Serverless-Sicherheitsgruppen hinzu.

Typ	Protocol (Protokoll)	Port-Bereich	Quelle
Alle TCP	TCP	3306	`emr-serverless-security-group`

Konfigurieren Sie Spark-Optionen

Verwenden von JDBC

Verwenden Sie eine JDBC-Verbindung, um Ihre EMR Serverless Spark-Anwendung so zu konfigurieren, dass sie eine Verbindung zu einem Hive-Metastore herstellt, der auf einer Amazon RDS for MySQL- oder Amazon Aurora MySQL-Instance basiert. Übergeben Sie das `mariadb-connector-java.jar` mit `--jars` in den Parametern Ihres Joblaufs`spark-submit`.

```
aws emr-serverless start-job-run \
  --application-id "application-id" \
  --execution-role-arn "job-role-arn" \
  --job-driver '{
        "sparkSubmit": {
            "entryPoint": "s3://amzn-s3-demo-bucket/scripts/spark-jdbc.py",
            "sparkSubmitParameters": "--jars s3://amzn-s3-demo-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://amzn-s3-demo-bucket/spark/logs/"
        }
    }
}'
```

Das folgende Codebeispiel ist ein Spark-Einstiegspunktskript, das mit einem Hive-Metastore auf Amazon RDS interagiert.

```
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()
```

Den Thrift-Service nutzen

Sie können Ihre EMR Serverless Hive-Anwendung so konfigurieren, dass sie eine Verbindung zu einem Hive-Metastore herstellt, der auf einer Amazon RDS for MySQL- oder Amazon Aurora MySQL-Instance basiert. Führen Sie dazu einen Thrift-Server auf dem primären Knoten eines vorhandenen Amazon EMR-Clusters aus. Diese Option ist ideal, wenn Sie bereits über einen Amazon EMR-Cluster mit einem Thrift-Server verfügen, den Sie verwenden möchten, um Ihre serverlosen EMR-Jobkonfigurationen zu vereinfachen.

```
aws emr-serverless start-job-run \
  --application-id "application-id" \
  --execution-role-arn "job-role-arn" \
  --job-driver '{
        "sparkSubmit": {
            "entryPoint": "s3://amzn-s3-demo-bucket/thriftscript.py",
```

```
            "sparkSubmitParameters": "--jars s3://amzn-s3-demo-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://amzn-s3-demo-bucket/spark/logs/"
        }
    }
}'
```

Das folgende Codebeispiel ist ein Entrypoint-Skript (`thriftscript.py`), das das Thrift-Protokoll verwendet, um eine Verbindung zu einem Hive-Metastore herzustellen. Beachten Sie, dass die `hive.metastore.uris` Eigenschaft so eingestellt sein muss, dass sie aus einem externen Hive-Metastore liest.

```
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()
```

Konfigurieren Sie die Hive-Optionen

Verwenden von JDBC

Wenn Sie einen externen Hive-Datenbankspeicherort auf einer Amazon RDS MySQL- oder Amazon Aurora Aurora-Instance angeben möchten, können Sie die Standard-Metastore-Konfiguration überschreiben.

> **Note**
>
> In Hive können Sie mehrere Schreibvorgänge in Metastore-Tabellen gleichzeitig ausführen. Wenn Sie Metastore-Informationen zwischen zwei Jobs gemeinsam nutzen, stellen Sie sicher, dass Sie nicht gleichzeitig in dieselbe Metastore-Tabelle schreiben, es sei denn, Sie schreiben in verschiedene Partitionen derselben Metastore-Tabelle.

Legen Sie die folgenden Konfigurationen in der `hive-site` Klassifizierung fest, um den externen Hive-Metastore zu aktivieren.

```
{
    "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"
    }
}
```

Verwenden Sie einen Thrift-Server

Sie können Ihre EMR Serverless Hive-Anwendung so konfigurieren, dass sie eine Verbindung zu einem Hive-Metastore herstellt, der auf Amazon RDS for MySQL oder Amazon Aurora My basiert. SQLinstance Führen Sie dazu einen Thrift-Server auf dem Hauptknoten eines vorhandenen Amazon EMR-Clusters aus. Diese Option ist ideal, wenn Sie bereits über einen Amazon EMR-Cluster verfügen, auf dem ein Thrift-Server ausgeführt wird, und Sie Ihre serverlosen EMR-Jobkonfigurationen verwenden möchten.

Legen Sie die folgenden Konfigurationen in der `hive-site` Klassifizierung fest, sodass EMR Serverless auf den Remote-Thrift-Metastore zugreifen kann. Beachten Sie, dass Sie die `hive.metastore.uris` Eigenschaft so einstellen müssen, dass sie aus einem externen Hive-Metastore liest.

```
{
    "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"
    }
}
```

Arbeiten mit der Multi-Katalog-Hierarchie von AWS Glue auf EMR Serverless

Sie können Ihre EMR Serverless-Anwendungen so konfigurieren, dass sie mit der AWS Glue-Hierarchie mit mehreren Katalogen funktionieren. Das folgende Beispiel zeigt, wie EMR-S Spark mit der Multi-Katalog-Hierarchie von AWS Glue verwendet wird.

Weitere Informationen zur Hierarchie mit mehreren Katalogen finden Sie unter Arbeiten mit einer Hierarchie mit mehreren Katalogen in AWS Glue Data Catalog with Spark auf Amazon EMR.

Verwendung von Redshift Managed Storage (RMS) mit Iceberg und AWS Glue Data Catalog

Im Folgenden wird gezeigt, wie Spark für die Integration mit einem AWS Glue-Datenkatalog mit Iceberg konfiguriert wird:

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "sparkSubmit": {
            "entryPoint": "s3://amzn-s3-demo-bucket/myscript.py",
            "sparkSubmitParameters": "--conf spark.sql.catalog.nfgac_rms =
 org.apache.iceberg.spark.SparkCatalog
             --conf spark.sql.catalog.rms.type=glue
             --conf spark.sql.catalog.rms.glue.id=Glue RMS catalog ID
```

```
            --conf spark.sql.defaultCatalog=rms
            --conf
 spark.sql.extensions=org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions"
       }
    }'
```

Eine Beispielabfrage aus einer Tabelle im Katalog nach der Integration:

```
SELECT * FROM my_rms_schema.my_table
```

Verwendung von Redshift Managed Storage (RMS) mit Iceberg REST API und AWS Glue Data Catalog

Im Folgenden wird gezeigt, wie Spark so konfiguriert wird, dass es mit dem Iceberg REST-Katalog funktioniert:

```
aws emr-serverless start-job-run \
--application-id application-id \
--execution-role-arn job-role-arn \
--job-driver '{
"sparkSubmit": {
"entryPoint": "s3://amzn-s3-demo-bucket/myscript.py",
    "sparkSubmitParameters": "
    --conf spark.sql.catalog.rms=org.apache.iceberg.spark.SparkCatalog
    --conf spark.sql.catalog.rms.type=rest
    --conf spark.sql.catalog.rms.warehouse=Glue RMS catalog ID
    --conf spark.sql.catalog.rms.uri=Glue endpoint URI/iceberg
    --conf spark.sql.catalog.rms.rest.sigv4-enabled=true
    --conf spark.sql.catalog.rms.rest.signing-name=glue
    --conf
 spark.sql.extensions=org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions"
    }
  }'
```

Eine Beispielabfrage aus einer Tabelle im Katalog:

```
SELECT * FROM my_rms_schema.my_table
```

Überlegungen bei der Verwendung eines externen Metastores

- Sie können Datenbanken, die mit MariaDB JDBC kompatibel sind, als Ihren Metastore konfigurieren. Beispiele für diese Datenbanken sind RDS for MariaDB, MySQL und Amazon Aurora.
- Metastoren werden nicht automatisch initialisiert. Wenn Ihr Metastore nicht mit einem Schema für Ihre Hive-Version initialisiert ist, verwenden Sie das Hive-Schematool.
- EMR Serverless unterstützt keine Kerberos-Authentifizierung. Sie können keinen Thrift-Metastore-Server mit Kerberos-Authentifizierung mit EMR Serverless Spark- oder Hive-Jobs verwenden.
- Sie müssen den VPC-Zugriff konfigurieren, um die Hierarchie mit mehreren Katalogen zu verwenden.

Zugreifen auf S3-Daten in einem anderen AWS Konto von EMR Serverless

Sie können Amazon EMR Serverless-Jobs von einem AWS Konto aus ausführen und sie so konfigurieren, dass sie auf Daten in Amazon S3 S3-Buckets zugreifen, die zu einem anderen Konto gehören. AWS Auf dieser Seite wird beschrieben, wie Sie den kontenübergreifenden Zugriff auf S3 von EMR Serverless aus konfigurieren.

Jobs, die auf EMR Serverless ausgeführt werden, können eine S3-Bucket-Richtlinie oder eine angenommene Rolle verwenden, um von einem anderen AWS Konto aus auf Daten in Amazon S3 zuzugreifen.

Voraussetzungen

Um den kontoübergreifenden Zugriff für Amazon EMR Serverless einzurichten, führen Sie Aufgaben aus, während Sie bei zwei Konten angemeldet sind: AWS

- **`AccountA`**— Dies ist das AWS Konto, in dem Sie eine serverlose Amazon EMR-Anwendung erstellt haben. Bevor Sie den kontoübergreifenden Zugriff einrichten, sollten Sie Folgendes für dieses Konto bereithalten:
  - Eine serverlose Amazon EMR-Anwendung, in der Sie Jobs ausführen möchten.
  - Eine Rolle zur Auftragsausführung, die über die erforderlichen Berechtigungen zum Ausführen von Jobs in der Anwendung verfügt. Weitere Informationen finden Sie unter Job-Runtime-Rollen für Amazon EMR Serverless.

- **`AccountB`**— Dies ist das AWS Konto, das den S3-Bucket enthält, auf den Ihre Amazon EMR Serverless-Jobs zugreifen sollen.

Verwenden Sie eine S3-Bucket-Richtlinie, um auf kontoübergreifende S3-Daten zuzugreifen

Um auf den S3-Bucket in account B zuzugreifenaccount A, fügen Sie dem S3-Bucket in die folgende Richtlinie beiaccount B.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "ExamplePermissions1",
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::my-bucket-name"
      ]
    },
    {
      "Sid": "ExamplePermissions2",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:DeleteObject"
      ],
      "Resource": [
        "arn:aws:s3:::my-bucket-name/*"
      ]
    }
  ]
}
```

Weitere Informationen zum kontoübergreifenden S3-Zugriff mit S3-Bucket-Richtlinien finden Sie unter Beispiel 2: Bucket-Besitzer gewährt kontoübergreifende Bucket-Berechtigungen im Amazon Simple Storage Service-Benutzerhandbuch.

Verwenden Sie eine angenommene Rolle, um auf kontoübergreifende S3-Daten zuzugreifen

Eine weitere Möglichkeit, den kontenübergreifenden Zugriff für Amazon EMR Serverless einzurichten, ist die `AssumeRole` Aktion von (). AWS -Security-Token-Service AWS STS AWS STS ist ein globaler Webservice, mit dem Sie temporäre Anmeldeinformationen mit eingeschränkten Rechten für Benutzer anfordern können. Mit den temporären Sicherheitsanmeldedaten, die Sie erstellen, können Sie API-Aufrufe an EMR Serverless und Amazon S3 tätigen. `AssumeRole`

Die folgenden Schritte veranschaulichen, wie Sie eine angenommene Rolle verwenden, um von EMR Serverless aus auf kontoübergreifende S3-Daten zuzugreifen:

1. Erstellen Sie einen Amazon-S3-Bucket,*`cross-account-bucket`*, in `AccountB`. Weitere Informationen finden Sie unter Erstellen eines Buckets im Amazon Simple Storage Service-Benutzerhandbuch. Wenn Sie kontenübergreifenden Zugriff auf DynamoDB haben möchten, erstellen Sie auch eine DynamoDB-Tabelle in. `AccountB` Weitere Informationen finden Sie unter Creating a DynamoDB table im Amazon DynamoDB Developer Guide.
2. Erstellen Sie eine `Cross-Account-Role-B` IAM-Rolle in `AccountB`, die auf das *`cross-account-bucket`* zugreifen kann.
   a. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die IAM-Konsole unter. https://console.aws.amazon.com/iam/
   b. Wählen Sie Rollen und anschließend Neue Rolle `Cross-Account-Role-`Berstellen aus. Weitere Informationen zum Erstellen von IAM-Rollen finden Sie unter Erstellen von IAM-Rollen im IAM-Benutzerhandbuch.
   c. Erstellen Sie eine IAM-Richtlinie, die die Berechtigungen für den `Cross-Account-Role-B` Zugriff auf den *`cross-account-bucket`* S3-Bucket festlegt, wie die folgende Richtlinienerklärung zeigt. Fügen Sie die IAM-Richtlinie an `Cross-Account-Role-B` an. Weitere Informationen finden Sie unter Erstellen von IAM-Richtlinien im IAM-Benutzerhandbuch.

   JSON

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

```
    {
      "Effect": "Allow",
      "Action": [
        "s3:*"
      ],
      "Resource": [
        "arn:aws:s3:::cross-account-bucket",
        "arn:aws:s3:::cross-account-bucket/*"
      ],
      "Sid": "AllowS3"
    }
  ]
}
```

Wenn Sie DynamoDB-Zugriff benötigen, erstellen Sie eine IAM-Richtlinie, die Berechtigungen für den Zugriff auf die kontoübergreifende DynamoDB-Tabelle festlegt. Fügen Sie die IAM-Richtlinie an `Cross-Account-Role-B` an. Weitere Informationen finden Sie unter Amazon DynamoDB: Ermöglicht den Zugriff auf eine bestimmte Tabelle im IAM-Benutzerhandbuch.

Die folgende Richtlinie ermöglicht den Zugriff auf die DynamoDB-Tabelle. `CrossAccountTable`

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:*"
      ],
      "Resource": [
        "arn:aws:dynamodb:*:123456789012:table/CrossAccountTable"
      ],
      "Sid": "AllowDYNAMODB"
    }
  ]
}
```

3. So bearbeiten Sie die Vertrauensbeziehung für die `Cross-Account-Role-B`-Rolle.

a. Um die Vertrauensstellung für die Rolle zu konfigurieren, wählen Sie in der IAM-Konsole die Registerkarte Trust Relationships für die Rolle aus`Cross-Account-Role-B`, die Sie in Schritt 2 erstellt haben.

b. Wählen Sie Vertrauensbeziehungen bearbeiten aus.

c. Fügen Sie das folgende Richtliniendokument hinzu. Dies ermöglicht es `Job-Execution-Role-A` uns`AccountA`, die `Cross-Account-Role-B` Rolle zu übernehmen.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole"
      ],
      "Resource": "arn:aws:iam::123456789012:role/Job-Execution-Role-A",
      "Sid": "AllowSTSAssumerole"
    }
  ]
}
```

4. `Job-Execution-Role-A`Erteilen `AccountA` Sie die AWS STS `AssumeRole` Erlaubnis zur Übernahme`Cross-Account-Role-B`.

a. Wählen Sie in der IAM-Konsole für das AWS Konto `AccountA` die Option aus`Job-Execution-Role-A`.

b. Fügen Sie die folgende Richtlinienanweisung zu `Job-Execution-Role-A` hinzu, um die `AssumeRole`-Aktion in der Rolle `Cross-Account-Role-B` zu verweigern.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole"
      ],
      "Resource": [
```

```
        "arn:aws:iam::123456789012:role/Cross-Account-Role-B"
      ],
      "Sid": "AllowSTSAssumerole"
    }
  ]
}
```

Beispiele für angenommene Rollen

Verwenden Sie eine einzelne angenommene Rolle, um auf alle S3-Ressourcen in einem Konto zuzugreifen, oder konfigurieren Sie mit Amazon EMR 6.11 und höher mehrere IAM-Rollen, die beim Zugriff auf verschiedene kontoübergreifende S3-Buckets übernommen werden sollen.

Themen

- Greifen Sie mit einer angenommenen Rolle auf S3-Ressourcen zu
- Greifen Sie auf S3-Ressourcen mit mehreren angenommenen Rollen zu

Greifen Sie mit einer angenommenen Rolle auf S3-Ressourcen zu

> **Note**
>
> Wenn Sie einen Job so konfigurieren, dass er eine einzelne angenommene Rolle verwendet, verwenden alle S3-Ressourcen des Jobs diese Rolle, einschließlich des `entryPoint` Skripts.

Wenn Sie eine einzige angenommene Rolle für den Zugriff auf alle S3-Ressourcen in Konto B verwenden möchten, geben Sie die folgenden Konfigurationen an:

1. Geben Sie die EMRFS-Konfiguration `fs.s3.customAWSCredentialsProvider` für an. `com.amazonaws.emr.AssumeRoleAWSCredentialsProvider`
2. Verwenden Sie für Spark `spark.emr-serverless.driverEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN` und, `spark.executorEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN` um die Umgebungsvariablen für Treiber und Executoren anzugeben.
3. Verwenden Sie für Hive, und `hive.emr-serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARNtez.am.emr-`

`serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARN`, `tez.task.emr-serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARN` um die Umgebungsvariablen für den Hive-Treiber, die primäre Tez-Anwendung und die Tez-Taskcontainer anzugeben.

Die folgenden Beispiele zeigen, wie eine angenommene Rolle verwendet wird, um eine serverlose EMR-Auftragsausführung mit kontenübergreifendem Zugriff zu starten.

Spark

Das folgende Beispiel zeigt, wie eine angenommene Rolle verwendet wird, um einen EMR Serverless Spark-Job mit kontenübergreifendem Zugriff auf S3 zu starten.

```
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.AssumeRoleAWSCredentialsProvider",
                "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

Das folgende Beispiel zeigt, wie eine angenommene Rolle verwendet wird, um einen EMR Serverless Hive-Auftrag mit kontenübergreifendem Zugriff auf S3 zu starten.

```
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"
            }
        }]
    }'
```

Greifen Sie auf S3-Ressourcen mit mehreren angenommenen Rollen zu

Mit den Versionen 6.11.0 und höher von EMR Serverless können Sie mehrere IAM-Rollen konfigurieren, die beim Zugriff auf verschiedene kontoübergreifende Buckets übernommen werden sollen. Wenn Sie auf verschiedene S3-Ressourcen mit unterschiedlichen angenommenen Rollen in Konto B zugreifen möchten, verwenden Sie die folgenden Konfigurationen, wenn Sie die Jobausführung starten:

1. Geben Sie die EMRFS-Konfiguration `fs.s3.customAWSCredentialsProvider` für an. `com.amazonaws.emr.serverless.credentialsprovider.BucketLevelAssumeRoleCredenti`

2. Geben Sie die EMRFS-Konfiguration `fs.s3.bucketLevelAssumeRoleMapping` an, um die Zuordnung vom S3-Bucket-Namen zur IAM-Rolle in Konto B zu definieren, die angenommen werden soll. Der Wert sollte das Format von haben. `bucket1->role1;bucket2->role2`

Verwenden Sie zum Beispiel für `arn:aws:iam::AccountB:role/Cross-Account-Role-B-1` den Zugriff auf den Bucket `bucket1` und verwenden Sie ihn für `arn:aws:iam::AccountB:role/Cross-Account-Role-B-2` den Zugriff auf den Bucket`bucket2`. Die folgenden Beispiele zeigen, wie Sie einen serverlosen EMR-Job starten, der mit kontenübergreifendem Zugriff über mehrere angenommene Rollen ausgeführt wird.

Spark

Das folgende Beispiel zeigt, wie mehrere angenommene Rollen verwendet werden, um eine EMR Serverless Spark-Jobausführung zu erstellen.

```
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

Die folgenden Beispiele zeigen, wie mehrere angenommene Rollen verwendet werden, um eine EMR Serverless Hive-Jobausführung zu erstellen.

```
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"
            }
        }]
    }'
```

Behebung von Fehlern in EMR Serverless

Verwenden Sie die folgenden Informationen, um häufig auftretende Probleme zu diagnostizieren und zu beheben, die bei der Arbeit mit Amazon EMR Serverless auftreten.

Themen

- Fehler: Der Job ist fehlgeschlagen, da das Konto das Dienstlimit für die maximale Anzahl von vCPU erreicht hat, die es gleichzeitig verwenden kann.
- Fehler: Der Job ist fehlgeschlagen, da die Anwendung die MaximumCapacity-Einstellungen überschritten hat.
- Fehler: Der Job ist fehlgeschlagen, da der Worker nicht zugewiesen werden konnte, da die Anwendung die maximale Kapazität überschritten hat.

- Fehler: Der S3-Zugriff wurde verweigert. Bitte überprüfen Sie die S3-Zugriffsberechtigungen der Job-Runtime-Rolle für die erforderlichen S3-Ressourcen.
- Fehler ModuleNotFoundError: Es wurde kein Modul benannt<module>. Informationen zur Verwendung von Python-Bibliotheken mit EMR Serverless finden Sie im Benutzerhandbuch.
- Fehler: Die Ausführungsrolle konnte nicht übernommen werden, <role name>da sie nicht existiert oder nicht mit der erforderlichen Vertrauensstellung eingerichtet ist.

Fehler: Der Job ist fehlgeschlagen, da das Konto das Dienstlimit für die maximale Anzahl von vCPU erreicht hat, die es gleichzeitig verwenden kann.

Dieser Fehler weist darauf hin, dass EMR Serverless den Job nicht senden konnte, da das Konto die maximale Kapazität überschritten hat. Erhöhen Sie die maximale Kapazität für das Konto. Überprüfen Sie Ihre Servicelimits bei EMR Serverless Service Quotas.

Fehler: Der Job ist fehlgeschlagen, da die Anwendung die MaximumCapacity-Einstellungen überschritten hat.

Dieser Fehler weist darauf hin, dass EMR Serverless den Job nicht senden konnte, da die Anwendung die konfigurierte maximale Kapazität überschritten hat. Erhöhen Sie die maximale Kapazität für die Anwendung.

Fehler: Der Job ist fehlgeschlagen, da der Worker nicht zugewiesen werden konnte, da die Anwendung die maximale Kapazität überschritten hat.

Dieser Fehler weist darauf hin, dass der Job nicht abgeschlossen werden konnte. Mitarbeiter konnten nicht zugewiesen werden, da die Anwendung die Einstellungen für die maximale Kapazität überschritten hat.

Fehler: Der S3-Zugriff wurde verweigert. Bitte überprüfen Sie die S3-Zugriffsberechtigungen der Job-Runtime-Rolle für die erforderlichen S3-Ressourcen.

Dieser Fehler weist darauf hin, dass Ihr Job keinen Zugriff auf Ihre S3-Ressourcen hat. Stellen Sie sicher, dass die Job-Runtime-Rolle berechtigt ist, auf die S3-Ressourcen zuzugreifen, die der Job

verwenden muss. Weitere Informationen zu Runtime-Rollen finden Sie unterJob-Runtime-Rollen für Amazon EMR Serverless.

Fehler ModuleNotFoundError: Es wurde kein Modul benannt<module>. Informationen zur Verwendung von Python-Bibliotheken mit EMR Serverless finden Sie im Benutzerhandbuch.

Dieser Fehler weist darauf hin, dass ein Python-Modul für den Spark-Job nicht verfügbar war. Überprüfen Sie, ob die abhängigen Python-Bibliotheken für den Job verfügbar sind. Weitere Informationen zum Verpacken von Python-Bibliotheken finden Sie unterVerwenden von Python-Bibliotheken mit EMR Serverless.

Fehler: Die Ausführungsrolle konnte nicht übernommen werden, <role name>da sie nicht existiert oder nicht mit der erforderlichen Vertrauensstellung eingerichtet ist.

Dieser Fehler weist darauf hin, dass die Job-Runtime-Rolle, die Sie für den Job angegeben haben, nicht existiert oder dass die Rolle keine Vertrauensstellung für EMR Serverless-Berechtigungen hat. Um zu überprüfen, ob die IAM-Rolle existiert und ob Sie die Vertrauensrichtlinie für die Rolle ordnungsgemäß eingerichtet haben, lesen Sie die Anweisungen unter. Job-Runtime-Rollen für Amazon EMR Serverless

Kostenzuweisung auf Tätigkeitsebene aktivieren

Die Kostenzuweisung auf Auftragsebene ermöglicht eine detaillierte Abrechnungszuweisung für EMR Serverless auf der Ebene der einzelnen Auftragsausführungen, anstatt alle Kosten auf Anwendungsebene zu aggregieren. Wenn diese Option aktiviert ist, können Sie die Kosten im AWS Cost Explorer und in Kosten- und Nutzungsberichten nach bestimmten Auftragsausführungen IDs und mit Auftragsläufen verknüpften Tags filtern und verfolgen, um einen besseren Überblick über die Gebühren für eingereichte Jobläufe zu erhalten.

Standardverhalten

Die Kostenzuweisung auf Auftragsebene ist standardmäßig nicht aktiviert.

Wie aktiviere oder deaktiviere ich die Funktion

Sie können die Kostenzuweisung auf Auftragsebene bei der Anwendungserstellung konfigurieren oder sie für bestehende Anwendungen aktualisieren.

Geben Sie den `jobLevelCostAllocation` Parameter an, wenn Sie eine neue Anwendung erstellen:

```
# Enable job-level cost allocation:
aws emr-serverless create-application \
    --name "my-application" \
    --release-label "emr-7.12.0" \
    --type "SPARK" \
    --job-level-cost-allocation-configuration '{
        "enabled": true
    }'

# Disable job-level cost allocation:
aws emr-serverless create-application \
    --name "my-application" \
    --release-label "emr-7.12.0" \
    --type "SPARK" \
    --job-level-cost-allocation-configuration '{
        "enabled": false
    }'
```

Aktualisieren Sie den `jobLevelCostAllocationConfiguration` Parameter für eine bestehende Anwendung:

```
# Enable job-level cost allocation:
aws emr-serverless update-application \
    --application-id <application-id> \
    --job-level-cost-allocation-configuration '{
        "enabled": true
    }'

# Disable job-level cost allocation:
aws emr-serverless update-application \
    --application-id <application-id> \
    --job-level-cost-allocation-configuration '{
        "enabled": false
    }'
```

Überlegungen und Einschränkungen

- Durch die Aktivierung der Kostenzuweisung auf Auftragsebene werden die Kosten nicht rückwirkend für Auftragsausführungen zugewiesen, die abgeschlossen wurden, bevor die Funktion aktiviert wurde. Auftragsausführungen, die nach der Aktivierung der Funktion gestartet werden, werden detailliert berechnet.
- Der Kostenzuweisungsparameter auf Auftragsebene kann nur aktualisiert werden, wenn sich eine Anwendung entweder im Status CREATED oder STOPPED befindet.
- Wenn die Kostenzuweisung auf Auftragsebene aktiviert ist, werden die Kosten einzelnen Auftragsläufen und nicht der Anwendung zugeordnet. Um aggregierte Kosten auf Anwendungsebene anzuzeigen, müssen Sie konsistente Tags (wie Anwendungsname oder Anwendungs-ID) auf alle Jobausführungen innerhalb dieser Anwendung anwenden und im Cost Explorer oder in Kosten- und Nutzungsberichten nach diesen Tags filtern.

Führen Sie interaktive Workloads mit EMR Serverless über EMR Studio aus

Mit interaktiven EMR Serverless-Anwendungen können Sie interaktive Workloads für Spark mit EMR Serverless mithilfe von Notebooks ausführen, die in EMR Studio gehostet werden.

-Übersicht

Eine interaktive Anwendung ist eine serverlose EMR-Anwendung, für die interaktive Funktionen aktiviert sind. Mit interaktiven Amazon EMR Serverless-Anwendungen können Sie interaktive Workloads mit Jupyter-Notebooks ausführen, die in Amazon EMR Studio verwaltet werden. Auf diese Weise können Dateningenieure, Datenwissenschaftler und Datenanalysten EMR Studio verwenden, um interaktive Analysen mit Datensätzen in Datenspeichern wie Amazon S3 und Amazon DynamoDB durchzuführen.

Zu den Anwendungsfällen für interaktive Anwendungen in EMR Serverless gehören:

- Dateningenieure verwenden die IDE-Erfahrung in EMR Studio, um ein ETL-Skript zu erstellen. Das Skript nimmt Daten vor Ort auf, transformiert die Daten für die Analyse und speichert die Daten in Amazon S3.
- Datenwissenschaftler verwenden Notebooks, um Datensätze zu untersuchen und Modelle für maschinelles Lernen (ML) zu trainieren, um Anomalien in den Datensätzen zu erkennen.
- Datenanalysten untersuchen Datensätze und erstellen Skripte, die tägliche Berichte generieren, um Anwendungen wie Geschäfts-Dashboards zu aktualisieren.

Voraussetzungen

Um interaktive Workloads mit EMR Serverless zu verwenden, müssen Sie die folgenden Anforderungen erfüllen:

- EMR Serverlose interaktive Anwendungen werden mit Amazon EMR 6.14.0 und höher unterstützt.
- Um auf Ihre interaktive Anwendung zuzugreifen, die von Ihnen eingereichten Workloads auszuführen und interaktive Notizbücher von EMR Studio aus auszuführen, benötigen Sie bestimmte Berechtigungen und Rollen. Weitere Informationen finden Sie unter Erforderliche Berechtigungen für interaktive Workloads.

Erforderliche Berechtigungen für interaktive Workloads

Konfigurieren Sie zusätzlich zu den grundlegenden Berechtigungen, die für den Zugriff auf EMR Serverless erforderlich sind, zusätzliche Berechtigungen für Ihre IAM-Identität oder Rolle:

So greifen Sie auf Ihre interaktive Anwendung zu

Richten Sie Benutzer- und Workspace-Berechtigungen für EMR Studio ein. Weitere Informationen finden Sie unter Benutzerberechtigungen für EMR Studio konfigurieren im Amazon EMR Management Guide.

Um die Workloads auszuführen, die Sie mit EMR Serverless einreichen

Richten Sie eine Job-Runtime-Rolle ein. Weitere Informationen finden Sie unter Erstellen Sie eine Job-Runtime-Rolle.

Um die interaktiven Notizbücher von EMR Studio aus auszuführen

Fügen Sie der IAM-Richtlinie für die Studio-Benutzer die folgenden zusätzlichen Berechtigungen hinzu:

- **`emr-serverless:AccessInteractiveEndpoints`**- Erteilt die Berechtigung, auf die interaktive Anwendung zuzugreifen und eine Verbindung zu ihr herzustellen. `Resource` Diese Berechtigung ist erforderlich, um von einem EMR Studio Workspace aus eine Verbindung zu einer EMR Serverless-Anwendung herzustellen.
- **`iam:PassRole`**- Erteilt die Berechtigung für den Zugriff auf die IAM-Ausführungsrolle, die Sie beim Anhängen an eine Anwendung verwenden möchten. Die entsprechende `PassRole` Berechtigung ist erforderlich, um von einem EMR Studio Workspace aus eine Verbindung zu einer EMR Serverless-Anwendung herzustellen.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "EMRServerlessInteractiveAccess",
      "Effect": "Allow",
      "Action": [
        "emr-serverless:AccessInteractiveEndpoints"
      ],
      "Resource": [
```

```
        "arn:aws:emr-serverless:*:123456789012:/applications/*"
      ]
    },
    {
      "Sid": "EMRServerlessRuntimeRoleAccess",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "arn:aws:iam::123456789012:role/EMRServerlessInteractiveRole"
      ],
      "Condition": {
        "StringLike": {
          "iam:PassedToService": "emr-serverless.amazonaws.com"
        }
      }
    }
  ]
}
```

Konfiguration interaktiver Anwendungen

Verwenden Sie die folgenden allgemeinen Schritte, um eine serverlose EMR-Anwendung mit interaktiven Funktionen aus Amazon EMR Studio in der zu erstellen. AWS-Managementkonsole

1. Folgen Sie den Schritten unter, um eine Anwendung Erste Schritte mit Amazon EMR Serverless zu erstellen.
2. Starten Sie dann einen Workspace in EMR Studio und fügen Sie ihn als Rechenoption an eine EMR Serverless-Anwendung an. Weitere Informationen finden Sie auf der Registerkarte Interaktive Arbeitslast in Schritt 2 der Dokumentation EMR Serverless Getting Started.

Wenn Sie eine Anwendung an einen Studio-Arbeitsbereich anhängen, wird der Anwendungsstart automatisch ausgelöst, sofern sie nicht bereits ausgeführt wird. Sie können die Anwendung auch vorab starten und bereithalten, bevor Sie sie an den Workspace anhängen.

Überlegungen zu interaktiven Anwendungen

- EMR Serverlose interaktive Anwendungen werden mit Amazon EMR 6.14.0 und höher unterstützt.

- EMR Studio ist der einzige Client, der in interaktive EMR Serverless-Anwendungen integriert ist. Die folgenden Funktionen von EMR Studio werden von interaktiven EMR-Anwendungen ohne Server nicht unterstützt: Workspace Collaboration, SQL Explorer und programmatische Ausführung von Notebooks.
- Interaktive Anwendungen werden nur für die Spark-Engine unterstützt.
- Interaktive Anwendungen unterstützen Python 3- PySpark und Spark-Scala-Kernel.
- Sie können bis zu 25 Notebooks gleichzeitig in einer einzigen interaktiven Anwendung ausführen.
- Es gibt keinen Endpunkt oder keine API-Schnittstelle, die selbst gehostete Jupyter-Notebooks mit interaktiven Anwendungen unterstützt.
- Für ein optimiertes Starterlebnis empfehlen wir Ihnen, die vorinitialisierte Kapazität für Treiber und Executoren zu konfigurieren und Ihre Anwendung vorab zu starten. Wenn Sie die Anwendung vorab starten, stellen Sie sicher, dass sie bereit ist, wenn Sie sie an Ihren Workspace anhängen möchten.

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

- `autoStopConfig`Ist standardmäßig für Anwendungen aktiviert. Dadurch wird die Anwendung nach 30 Minuten Leerlaufzeit heruntergefahren. Sie können diese Konfiguration im Rahmen Ihrer `create-application` `update-application` PR-Anfrage ändern.
- Wenn Sie eine interaktive Anwendung verwenden, empfehlen wir Ihnen, eine vorinitialisierte Kapazität von Kerneln, Treibern und Executoren für den Betrieb Ihrer Notebooks zu konfigurieren. Jede interaktive Spark-Sitzung erfordert einen Kernel und einen Treiber, sodass EMR Serverless für jeden vorinitialisierten Treiber einen vorinitialisierten Kernel-Worker verwaltet. Standardmäßig behält EMR Serverless die vorinitialisierte Kapazität eines Kernel-Workers für die gesamte Anwendung bei, auch wenn Sie keine vorinitialisierte Kapazität für Treiber angeben. Jeder Kernel-Worker verwendet 4 vCPUs und 16 GB Arbeitsspeicher. Aktuelle Preisinformationen finden Sie auf der Amazon EMR-Preisseite.
- Sie müssen über ein ausreichendes vCPU-Dienstkontingent verfügen AWS-Konto , um interaktive Workloads ausführen zu können. Wenn Sie keine Lake Formation-fähigen Workloads ausführen, empfehlen wir mindestens 24 vCPU. In diesem Fall empfehlen wir mindestens 28 vCPU.
- EMR Serverless beendet automatisch die Kernel von den Notebooks, wenn sie länger als 60 Minuten inaktiv waren. EMR Serverless berechnet die Kernel-Leerlaufzeit anhand der letzten Aktivität, die während der Notebook-Sitzung abgeschlossen wurde. Sie können die Einstellung für das Leerlauf-Timeout des Kernels derzeit nicht ändern.

- Um Lake Formation mit interaktiven Workloads zu aktivieren, stellen Sie die Konfiguration `spark.emr-serverless.lakeformation.enabled` auf `true` unter der `spark-defaults` Klassifizierung im `runtime-configuration` Objekt ein, wenn Sie eine serverlose EMR-Anwendung erstellen. Weitere Informationen finden Sie unter Enabling Lake Formation in Amazon EMR.

Führen Sie interaktive Workloads mit EMR Serverless über einen Apache Livy-Endpunkt aus

Mit Amazon EMR-Versionen 6.14.0 und höher können Sie beim Erstellen einer serverlosen EMR-Anwendung einen Apache Livy-Endpunkt erstellen und aktivieren und interaktive Workloads über Ihre selbst gehosteten Notebooks oder mit einem benutzerdefinierten Client ausführen. Ein Apache Livy-Endpunkt bietet die folgenden Vorteile:

- Sie können über Jupyter-Notebooks eine sichere Verbindung zu einem Apache Livy-Endpunkt herstellen und Apache Spark-Workloads mit der REST-Schnittstelle von Apache Livy verwalten.
- Verwenden Sie die Apache Livy REST-API-Operationen für interaktive Webanwendungen, die Daten aus Apache Spark-Workloads verwenden.

Voraussetzungen

Um einen Apache Livy-Endpunkt mit EMR Serverless zu verwenden, müssen Sie die folgenden Anforderungen erfüllen:

- Führen Sie die Schritte unter Erste Schritte mit Amazon EMR Serverless aus.
- Um interaktive Workloads über Apache Livy-Endpunkte auszuführen, benötigen Sie bestimmte Berechtigungen und Rollen. Weitere Informationen finden Sie unter Erforderliche Berechtigungen für interaktive Workloads.

Erforderliche Berechtigungen

Fügen Sie Ihrer IAM-Rolle zusätzlich zu den erforderlichen Berechtigungen für den Zugriff auf EMR Serverless die folgenden Berechtigungen hinzu, um auf einen Apache Livy-Endpunkt zuzugreifen und Anwendungen auszuführen:

- `emr-serverless:AccessLivyEndpoints`— erteilt die Erlaubnis, auf die Livy-fähige Anwendung, die Sie angeben, zuzugreifen und eine Verbindung zu ihr herzustellen. `Resource` Sie benötigen diese Berechtigung, um die REST-API-Operationen auszuführen, die vom Apache Livy-Endpunkt aus verfügbar sind.
- `iam:PassRole`— erteilt beim Erstellen der Apache Livy-Sitzung die Berechtigung zum Zugriff auf die IAM-Ausführungsrolle. EMR Serverless verwendet diese Rolle, um Ihre Workloads auszuführen.
- `emr-serverless:GetDashboardForJobRun`— erteilt die Genehmigung zur Generierung der Links zur Benutzeroberfläche von Spark Live und zum Treiberprotokoll und gewährt Zugriff auf die Protokolle als Teil der Apache Livy-Sitzungsergebnisse.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "EMRServerlessInteractiveAccess",
      "Effect": "Allow",
      "Action": [
        "emr-serverless:AccessLivyEndpoints"
      ],
      "Resource": [
        "arn:aws:emr-serverless:*:123456789012:/applications/*"
      ]
    },
    {
      "Sid": "EMRServerlessRuntimeRoleAccess",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "arn:aws:iam::123456789012:role/EMRServerlessExecutionRole"
      ],
      "Condition": {
        "StringLike": {
          "iam:PassedToService": "emr-serverless.amazonaws.com"
        }
      }
```

```
    },
    {
      "Sid": "EMRServerlessDashboardAccess",
      "Effect": "Allow",
      "Action": [
        "emr-serverless:GetDashboardForJobRun"
      ],
      "Resource": [
        "arn:aws:emr-serverless:*:123456789012:/applications/*"
      ]
    }
  ]
}
```

Erste Schritte

Gehen Sie wie folgt vor, um eine Apache Livy-fähige Anwendung zu erstellen und auszuführen.

1. Führen Sie den folgenden Befehl aus, um eine Apache Livy-fähige Anwendung zu erstellen.

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

2. Nachdem EMR Serverless Ihre Anwendung erstellt hat, starten Sie die Anwendung, um den Apache Livy-Endpunkt verfügbar zu machen.

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

   Verwenden Sie den folgenden Befehl, um zu überprüfen, ob der Status Ihrer Anwendung stimmt. Sobald der Status erreicht ist`STARTED`, greifen Sie auf den Apache Livy-Endpunkt zu.

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

3. Verwenden Sie die folgende URL, um auf den Endpunkt zuzugreifen:

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

Sobald der Endpunkt bereit ist, reichen Sie Workloads auf der Grundlage Ihres Anwendungsfalls ein. Sie müssen jede Anfrage an den Endpunkt mit dem SIGv4 Protokoll signieren und einen Autorisierungsheader übergeben. Sie können die folgenden Methoden verwenden, um Workloads auszuführen:

- HTTP-Client — reichen Sie Ihre Apache Livy-Endpunkt-API-Operationen mit einem benutzerdefinierten HTTP-Client ein.
- Sparkmagic-Kernel — Führen Sie den Sparkmagic-Kernel lokal aus und senden Sie interaktive Abfragen mit Jupyter-Notebooks.

HTTP-Clients

Um eine Apache Livy-Sitzung zu erstellen, geben `emr-serverless.session.executionRoleArn` Sie den Parameter Ihres Anfragetextes ein. `conf` Das folgende Beispiel ist eine `POST /sessions` Beispielanforderung.

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

In der folgenden Tabelle werden alle verfügbaren Apache Livy API-Operationen beschrieben.

API-Operation	Description
GET /sessions	Gibt eine Liste aller aktiven interaktiven Sitzungen zurück.
POST /Sessions	Erzeugt eine neue interaktive Sitzung über Spark oder Pyspark.

API-Operation	Description
GET /sessions/ < > *sessionId*	Gibt die Sitzungsinformationen zurück.
GET /sessions/ < >/state *sessionId*	Gibt den Status der Sitzung zurück.
LÖSCHE /sessions/ < > *sessionId*	Stoppt und löscht die Sitzung.
GET /sessions/ < >/statements *sessionId*	Gibt alle Anweisungen in einer Sitzung zurück.
POST /sessions/ < >/statements *sessionId*	Führt eine Anweisung in einer Sitzung aus.
GET /sessions/ < >/statements/< > *sessionId* *statementId*	Gibt die Details der angegebenen Anweisung in einer Sitzung zurück.
POST /sessions/ < >/statements/< >/cancel *sessionId* *statementId*	Bricht die angegebene Anweisung in dieser Sitzung ab.

Anfragen an den Apache Livy-Endpunkt senden

Sie können Anfragen auch direkt von einem HTTP-Client an den Apache Livy-Endpunkt senden. Auf diese Weise können Sie Code für Ihre Anwendungsfälle außerhalb eines Notebooks remote ausführen.

Bevor Sie Anfragen an den Endpunkt senden, stellen Sie sicher, dass Sie die folgenden Bibliotheken installiert haben:

```
pip3 install botocore awscrt requests
```

Im Folgenden finden Sie ein Python-Skript zum direkten Senden von HTTP-Anfragen an einen Endpunkt:

```
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())
```

Sparkmagic-Kernel

Bevor Sie Sparkmagic installieren, stellen Sie sicher, dass Sie AWS Anmeldeinformationen für die Instanz konfiguriert haben, in der Sie Sparkmagic installieren möchten

1. Installieren Sie sparkmagic, indem Sie die Installationsschritte befolgen. Beachten Sie, dass Sie nur die ersten vier Schritte ausführen.
2. Der Sparkmagic-Kernel unterstützt benutzerdefinierte Authentifikatoren, sodass Sie einen Authenticator in den Sparkmagic-Kernel integrieren können, sodass jede Anfrage signiert ist. SIGv4
3. Installieren Sie den benutzerdefinierten EMR Serverless Authenticator.

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

4. Geben Sie nun den Pfad zum benutzerdefinierten Authentifikator und die Apache Livy-Endpunkt-URL in der Sparkmagic-Konfigurations-JSON-Datei an. Verwenden Sie den folgenden Befehl, um die Konfigurationsdatei zu öffnen.

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

Im Folgenden finden Sie eine `config.json` Beispieldatei.

```
{
"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. Starten Sie Jupyter Lab. Es sollte die benutzerdefinierte Authentifizierung verwenden, die Sie im letzten Schritt eingerichtet haben.
6. Sie können dann die folgenden Notebook-Befehle und Ihren Code ausführen, um loszulegen.

```
%%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
```

Intern ruft jede Anweisung jede der Apache Livy-API-Operationen über die konfigurierte Apache Livy-Endpunkt-URL auf. Anschließend können Sie Ihre Anweisungen entsprechend Ihrem Anwendungsfall schreiben.

Überlegungen

Beachten Sie die folgenden Überlegungen, wenn Sie interaktive Workloads über Apache Livy-Endpunkte ausführen.

- EMR Serverless hält die Isolierung auf Sitzungsebene mithilfe des Anruferprinzipals aufrecht. Der Anruferprinzipal, der die Sitzung erstellt, ist der einzige, der auf diese Sitzung zugreifen kann. Für eine detailliertere Isolierung konfigurieren Sie eine Quellidentität, wenn Sie Anmeldeinformationen annehmen. In diesem Fall erzwingt EMR Serverless die Isolierung auf Sitzungsebene basierend auf dem Anrufer-Prinzipal und der Quellidentität. Weitere Informationen zur Quellidentität finden Sie unter Überwachen und Steuern von Aktionen, die mit übernommenen Rollen ergriffen wurden.
- Apache Livy-Endpunkte werden mit den EMR Serverless-Versionen 6.14.0 und höher unterstützt.
- Apache Livy-Endpunkte werden nur für die Apache Spark-Engine unterstützt.
- Apache Livy-Endpunkte unterstützen Scala Spark und. PySpark
- `autoStopConfig`Ist standardmäßig in Ihren Anwendungen aktiviert. Das bedeutet, dass Anwendungen nach 15 Minuten Inaktivität heruntergefahren werden. Sie können diese Konfiguration im Rahmen Ihrer `create-application` `update-application` PR-Anfrage ändern.
- Sie können bis zu 25 gleichzeitige Sitzungen auf einer einzigen Apache Livy-Endpoint-fähigen Anwendung ausführen.

- Für ein optimales Starterlebnis empfehlen wir, die vorinitialisierte Kapazität für Treiber und Executoren zu konfigurieren.
- Sie müssen Ihre Anwendung manuell starten, bevor Sie eine Verbindung zum Apache Livy-Endpunkt herstellen können.
- Sie müssen über ein ausreichendes vCPU-Dienstkontingent verfügen, AWS-Konto um interaktive Workloads mit dem Apache Livy-Endpunkt ausführen zu können. Wir empfehlen mindestens 24 vCPU.
- Das standardmäßige Apache Livy-Sitzungstimeout beträgt 1 Stunde. Wenn Sie eine Stunde lang keine Anweisungen ausführen, löscht Apache Livy die Sitzung und gibt den Treiber und die Executoren frei. Ab Version emr-7.8.0 kann dieser Wert gesetzt werden, indem der `ttl` Parameter als Teil einer `/sessions` `POST` Livy-Anfrage angegeben wird, zum Beispiel `2h` (Stunden), (Minuten), (Sekunden), `120m` (Millisekunden). `7200s` `7200000ms`

  > **Note**
  >
  > Diese Konfiguration kann vor emr-7.8.0 nicht geändert werden. Im Folgenden finden Sie ein Beispiel für einen Anfragetext. `POST /sessions`

  ```
  {
      "kind": "pyspark",
      "heartbeatTimeoutInSecond": 60,
      "conf": {
          "emr-serverless.session.executionRoleArn": "executionRoleArn"
      },
      "ttl": "2h"
  }
  ```

- Ab der Amazon EMR-Version emr-7.8.0 für Anwendungen mit detaillierter Zugriffskontrolle über LakeFormation aktiviert kann die Einstellung pro Sitzung deaktiviert werden. Weitere Informationen zur Aktivierung der feinkörnigen Zugriffskontrolle für eine serverlose EMR-Anwendung finden Sie unter Methoden für feinkörnige Zugriffskontrolle.

> **Note**
>
> Lake Formation kann nicht für eine Sitzung aktiviert werden, wenn sie nicht für eine Anwendung aktiviert wurde. Im Folgenden finden Sie ein Beispiel für einen `POST /sessions` Anfragetext.

```
{
    "kind": "pyspark",
    "heartbeatTimeoutInSecond": 60,
    "conf": {
        "emr-serverless.session.executionRoleArn": "executionRoleArn"
    },
    "spark.emr-serverless.lakeformation.enabled" : "false"
}
```

- Nur aktive Sitzungen können mit einem Apache Livy-Endpunkt interagieren. Sobald die Sitzung beendet, abgebrochen oder beendet wurde, können Sie nicht mehr über den Apache Livy-Endpunkt darauf zugreifen.

Protokollierung und Überwachung

Die Überwachung ist ein wichtiger Bestandteil der Aufrechterhaltung der Zuverlässigkeit, Verfügbarkeit und Leistung von EMR Serverless-Anwendungen und -Jobs. Sie sollten Überwachungsdaten aus allen Teilen Ihrer EMR Serverless-Lösungen sammeln, damit Multipoint-Ausfälle einfacher debuggt werden können, falls sie auftreten.

Themen

Speichern von Protokollen

Um Ihren Auftragsfortschritt auf EMR Serverless zu überwachen und Auftragsfehler zu beheben, wählen Sie aus, wie EMR Serverless Anwendungsprotokolle speichert und bereitstellt. Wenn Sie einen Job ausführen, geben Sie Managed Storage, Amazon S3 und Amazon CloudWatch als Protokollierungsoptionen an.

Geben Sie mit CloudWatch die Protokolltypen und Protokollspeicherorte an, die Sie verwenden möchten, oder akzeptieren Sie die Standardtypen und Speicherorte. Weitere Informationen zu CloudWatch Protokollen finden Sie unterthe section called “Amazon CloudWatch”. Bei verwaltetem Speicher und S3-Protokollierung sind in der folgenden Tabelle die Protokollspeicherorte und die Verfügbarkeit der Benutzeroberfläche aufgeführt, die Sie erwarten können, wenn Sie sich für verwalteten Speicher, Amazon S3 S3-Buckets oder beides entscheiden.

Option	Ereignisprotokolle	Containerprotokolle	Benutzeroberfläche der Anwendung
Verwalteter Speicher	In verwaltetem Speicher gespeichert	In verwaltetem Speicher gespeichert	Unterstützt

Option	Ereignisprotokolle	Containerprotokolle	Benutzeroberfläche der Anwendung
Sowohl verwalteter Speicher als auch S3-Bucket	An beiden Orten gespeichert	Im S3-Bucket gespeichert	Unterstützt
Amazon-S3-Bucket	Im S3-Bucket gespeichert	Im S3-Bucket gespeichert	Nicht unterstützt [1]

[1] Wir empfehlen, dass Sie die Option Verwalteter Speicher ausgewählt lassen. Andernfalls können Sie die integrierte Anwendung nicht verwenden UIs.

Protokollierung für EMR Serverless mit verwaltetem Speicher

Standardmäßig speichert EMR Serverless Anwendungsprotokolle sicher für maximal 30 Tage im von Amazon EMR verwalteten Speicher.

> **Note**
>
> Wenn Sie die Standardoption deaktivieren, kann Amazon EMR Ihre Jobs nicht in Ihrem Namen beheben. Beispiel: Sie können nicht von der EMR Serverless Console aus auf Spark-UI zugreifen.

Um diese Option in EMR Studio zu deaktivieren, deaktivieren Sie AWS auf der Seite Job einreichen im Abschnitt Zusätzliche Einstellungen das Kontrollkästchen Zulassen, Protokolle für 30 Tage aufzubewahren.

Um diese Option von zu deaktivieren AWS CLI, verwenden Sie die `managedPersistenceMonitoringConfiguration` Konfiguration, wenn Sie eine Jobausführung einreichen.

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

```
    }
}
```

Wenn sich Ihre EMR Serverless-Anwendung in einem privaten Subnetz mit VPC-Endpunkten für Amazon S3 befindet und Sie eine Endpunktrichtlinie zur Zugriffskontrolle anhängen, fügen Sie die folgenden Berechtigungen für EMR Serverless zum Speichern und Bereitstellen von Anwendungsprotokollen hinzu. Ersetzen Sie sie durch die `AppInfo` Buckets aus der Tabelle `Resource` mit den verfügbaren Regionen in Beispielrichtlinien für private Subnetze, die auf Amazon S3 zugreifen.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "EMRServerlessManagedLogging",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:PutObjectAcl"
      ],
      "Resource": [
        "arn:aws:s3:::prod.us-east-1.appinfo.src",
        "arn:aws:s3:::prod.us-east-1.appinfo.src/*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:PrincipalServiceName": "emr-serverless.amazonaws.com",
          "aws:SourceVpc": "vpc-12345678"
        }
      }
    }
  ]
}
```

Verwenden Sie außerdem den `aws:SourceVpc` Bedingungsschlüssel, um sicherzustellen, dass die Anfrage die VPC durchläuft, an die der VPC-Endpunkt angeschlossen ist.

Protokollierung für EMR Serverless mit Amazon S3 S3-Buckets

Bevor Ihre Jobs Protokolldaten an Amazon S3 senden können, müssen Sie die folgenden Berechtigungen in die Berechtigungsrichtlinie für die Job-Runtime-Rolle aufnehmen. *amzn-s3-demo-logging-bucket*Ersetzen Sie es durch den Namen Ihres Logging-Buckets.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
      ],
      "Sid": "AllowS3Putobject"
    }
  ]
}
```

Um einen Amazon S3 S3-Bucket zum Speichern von Protokollen einzurichten AWS CLI, verwenden Sie die `s3MonitoringConfiguration` Konfiguration, wenn Sie einen Joblauf starten. Geben Sie dazu `--configuration-overrides` in der Konfiguration Folgendes an.

```
{
    "monitoringConfiguration": {
        "s3MonitoringConfiguration": {
            "logUri": "s3://amzn-s3-demo-logging-bucket/logs/"
        }
    }
}
```

Bei Batch-Jobs, für die keine Wiederholungsversuche aktiviert sind, sendet EMR Serverless die Protokolle an den folgenden Pfad:

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

Spark-Treiberprotokolle werden von EMR Serverless im folgenden Pfad gespeichert

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

Spark-Executor-Logs werden von EMR Serverless im folgenden Pfad gespeichert

```
'/applications/<applicationId>/jobs/<jobId>/SPARK_EXECUTOR/<EXECUTOR-ID>'
```

<EXECUTOR-ID>Das ist eine Ganzzahl.

EMR Serverless Releases 7.1.0 und höher unterstützen Wiederholungsversuche für Streaming-Jobs und Batch-Jobs. Wenn Sie einen Job mit aktivierten Wiederholungsversuchen ausführen, fügt EMR Serverless dem Protokollpfadpräfix automatisch eine Versuchsnummer hinzu, sodass Sie Protokolle besser unterscheiden und verfolgen können.

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

Protokollierung für EMR Serverless mit Amazon CloudWatch

Wenn Sie einen Job an eine EMR Serverless-Anwendung senden, wählen Sie Amazon CloudWatch als Option zum Speichern Ihrer Bewerbungsprotokolle. Auf diese Weise können Sie Protokollanalysefunktionen wie CloudWatch CloudWatch Logs Insights und Live Tail verwenden. Sie können Protokolle auch von CloudWatch anderen Systemen streamen, z. B. OpenSearch zur weiteren Analyse.

EMR Serverless bietet Echtzeitprotokollierung für Treiberprotokolle. Sie können in Echtzeit mit der CloudWatch Live-Tail-Funktion oder über CloudWatch CLI-Tail-Befehle auf die Protokolle zugreifen.

Standardmäßig ist die CloudWatch Protokollierung für EMR Serverless deaktiviert. Um es zu aktivieren, verwenden Sie die Konfiguration in. AWS CLI

> **Note**
>
> Amazon CloudWatch veröffentlicht Protokolle in Echtzeit, sodass mehr Ressourcen von Mitarbeitern benötigt werden. Wenn Sie sich für eine geringe Arbeitskapazität entscheiden, kann sich die Auswirkung auf die Laufzeit Ihres Jobs erhöhen. Wenn Sie die CloudWatch

Protokollierung aktivieren, empfehlen wir Ihnen, eine höhere Arbeitskapazität zu wählen. Es ist auch möglich, dass die Protokollveröffentlichung drosselt, wenn die Transaktionsrate pro Sekunde (TPS) für `PutLogEvents` zu niedrig ist. Die CloudWatch Drosselungskonfiguration gilt global für alle Dienste, einschließlich EMR Serverless. Weitere Informationen finden Sie unter Wie stelle ich die Drosselung in meinen Protokollen fest? CloudWatch auf AWS re:post.

Erforderliche Berechtigungen für das Loggen mit CloudWatch

Bevor Ihre Jobs Protokolldaten an Amazon senden können CloudWatch, müssen Sie die folgenden Berechtigungen in die Berechtigungsrichtlinie für die Job-Runtime-Rolle aufnehmen.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "arn:aws:logs:*:123456789012:*"
      ],
      "Sid": "AllowLOGSDescribeloggroups"
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents",
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:DescribeLogStreams"
      ],
      "Resource": [
        "arn:aws:logs:*:123456789012:log-group:my-log-group-name:*"
      ],
      "Sid": "AllowLOGSPutlogevents"
    }
```

```
  ]
}
```

AWS CLI

CloudWatch Um Amazon so einzurichten, dass Protokolle für EMR Serverless von gespeichert werden AWS CLI, verwenden Sie die `cloudWatchLoggingConfiguration` Konfiguration, wenn Sie einen Joblauf starten. Geben Sie dazu die folgenden Konfigurationsüberschreibungen an. Geben Sie optional auch einen Protokollgruppennamen, einen Protokollstream-Präfixnamen, Protokolltypen und einen Verschlüsselungsschlüssel-ARN an.

Wenn Sie die optionalen Werte nicht angeben, werden die Protokolle in einer Standardprotokollgruppe `/aws/emr-serverless` mit dem Standardprotokollstream CloudWatch veröffentlicht`/applications/`*`applicationId`*`/jobs/`*`jobId`*`/`*`worker-type`*.

EMR Serverless Releases 7.1.0 und höher unterstützen Wiederholungsversuche für Streaming-Jobs und Batch-Jobs. Wenn Sie Wiederholungsversuche für einen Job aktiviert haben, fügt EMR Serverless dem Protokollpfadpräfix automatisch eine Versuchsnummer hinzu, sodass Sie Protokolle besser unterscheiden und verfolgen können.

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

Im Folgenden wird die Mindestkonfiguration veranschaulicht, die erforderlich ist, um die CloudWatch Amazon-Protokollierung mit den Standardeinstellungen für EMR Serverless zu aktivieren:

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

Das folgende Beispiel zeigt alle erforderlichen und optionalen Konfigurationen, die angeben, wann Sie die CloudWatch Amazon-Protokollierung für EMR Serverless aktivieren. Die unterstützten `logTypes` Werte sind auch im folgenden Beispiel aufgeführt.

```
{
```

```
    "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
             }
         }
     }
}
```

Standardmäßig veröffentlicht EMR Serverless nur die Treiber-Stdout- und Stderr-Protokolle. CloudWatch Wenn Sie andere Protokolle verwenden möchten, geben Sie im Feld eine Container-Rolle und die entsprechenden Protokolltypen an. `logTypes`

In der folgenden Liste sind die unterstützten Worker-Typen aufgeführt, die für die `logTypes` Konfiguration angegeben wurden:

Spark

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

Hive

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

Rotierende Protokolle

Amazon EMR Serverless kann Spark-Anwendungsprotokolle und Ereignisprotokolle rotieren. Die Protokollrotation hilft bei dem Problem, dass lange laufende Jobs große Protokolldateien erzeugen, die Ihren gesamten Festplattenspeicher beanspruchen können. Durch das Rotieren von Protokollen können Sie Festplattenspeicher sparen und die Anzahl der fehlgeschlagenen Jobs reduzieren, da Sie keinen Speicherplatz mehr auf Ihrer Festplatte haben.

Die Protokollrotation ist standardmäßig aktiviert und nur für Spark-Jobs verfügbar.

Spark-Ereignisprotokolle

> **Note**
>
> Die Rotation des Spark-Ereignisprotokolls ist für alle Amazon EMR-Release-Labels verfügbar.

Anstatt eine einzelne Ereignisprotokolldatei zu generieren, rotiert EMR Serverless das Ereignisprotokoll in regelmäßigen Zeitintervallen und entfernt die älteren Ereignisprotokolldateien. Das Rotieren von Protokollen wirkt sich nicht auf die in den S3-Bucket hochgeladenen Protokolle aus.

Spark-Anwendungsprotokolle

> **Note**
>
> Die Rotation der Spark-Anwendungsprotokolle ist für alle Amazon EMR-Release-Labels verfügbar.

EMR Serverless rotiert auch die Spark-Anwendungsprotokolle für Treiber und Executoren, wie z. B. AND-Dateien. `stdout` `stderr` Sie können auf die neuesten Protokolldateien zugreifen, indem Sie die Log-Links in Studio auswählen, indem Sie die Links zum Spark History Server und zur Live-Benutzeroberfläche verwenden. Protokolldateien sind die gekürzten Versionen der neuesten Protokolle. Um auf die älteren rotierten Protokolle zu verweisen, geben Sie beim Speichern von Protokollen einen Amazon S3 S3-Speicherort an. Weitere Informationen finden Sie unter Logging for EMR Serverless with Amazon S3 Buckets.

Die neuesten Protokolldateien finden Sie am folgenden Speicherort. EMR Serverless aktualisiert die Dateien alle 15 Sekunden. Diese Dateien können zwischen 0 MB und 128 MB liegen.

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

Der folgende Speicherort enthält die älteren rotierten Dateien. Jede Datei ist 128 MB groß.

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

Das gleiche Verhalten gilt auch für Spark-Executoren. Diese Änderung gilt nur für die S3-Protokollierung. Durch die Protokollrotation werden keine Änderungen an den auf Amazon hochgeladenen Protokollstreams vorgenommen CloudWatch.

EMR Serverless Releases 7.1.0 und höher unterstützen Wiederholungsversuche für Streaming- und Batch-Jobs. Wenn Sie für Ihren Job Wiederholungsversuche aktiviert haben, fügt EMR Serverless dem Protokollpfad für solche Jobs ein Präfix hinzu, sodass Sie die Protokolle besser verfolgen und voneinander unterscheiden können. Dieser Pfad enthält alle rotierten Protokolle.

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

Protokolle verschlüsseln

Verschlüsselung von serverlosen EMR-Protokollen mit verwaltetem Speicher

Um Protokolle im verwalteten Speicher mit Ihrem eigenen KMS-Schlüssel zu verschlüsseln, verwenden Sie die `managedPersistenceMonitoringConfiguration` Konfiguration, wenn Sie eine Jobausführung einreichen.

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

Verschlüsselung von serverlosen EMR-Protokollen mit Amazon S3 S3-Buckets

Um Logs in Ihrem Amazon S3 S3-Bucket mit Ihrem eigenen KMS-Schlüssel zu verschlüsseln, verwenden Sie die `s3MonitoringConfiguration` Konfiguration, wenn Sie eine Jobausführung einreichen.

```
{
    "monitoringConfiguration": {
        "s3MonitoringConfiguration": {
            "logUri": "s3://amzn-s3-demo-logging-bucket/logs/",
            "encryptionKeyArn": "key-arn"
        }
    }
```

```
}
```

Verschlüsselung von serverlosen EMR-Protokollen mit Amazon CloudWatch

Um Protokolle in Amazon CloudWatch mit Ihrem eigenen KMS-Schlüssel zu verschlüsseln, verwenden Sie die `cloudWatchLoggingConfiguration` Konfiguration, wenn Sie einen Job-Lauf einreichen.

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

Erforderliche Berechtigungen für die Protokollverschlüsselung

In diesem Abschnitt

- Erforderliche Benutzerberechtigungen
- Berechtigungen für Verschlüsselungsschlüssel für Amazon S3 und verwalteten Speicher
- Berechtigungen für Verschlüsselungsschlüssel für Amazon CloudWatch

Erforderliche Benutzerberechtigungen

Der Benutzer, der den Job einreicht oder die Protokolle oder die Anwendung ansieht, UIs muss über die erforderlichen Berechtigungen zur Verwendung des Schlüssels verfügen. Sie können die Berechtigungen entweder in der KMS-Schlüsselrichtlinie oder in der IAM-Richtlinie für den Benutzer, die Gruppe oder die Rolle angeben. Wenn der Benutzer, der den Job einreicht, nicht über die KMS-Schlüsselberechtigungen verfügt, lehnt EMR Serverless die Übermittlung der Auftragsausführung ab.

Beispiel für eine Schlüsselrichtlinie

Die folgende Schlüsselrichtlinie stellt die Berechtigungen für `kms:GenerateDataKey` und bereit`kms:Decrypt`:

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

Beispiel für eine IAM-Richtlinie

Die folgende IAM-Richtlinie bietet die Berechtigungen für `kms:GenerateDataKey` und`kms:Decrypt`:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:GenerateDataKey",
        "kms:Decrypt"
      ],
      "Resource": [
        "arn:aws:kms:*:123456789012:key/12345678-1234-1234-1234-123456789012"
      ],
      "Sid": "AllowKMSGeneratedatakey"
    }
  ]
}
```

Um die Spark- oder Tez-Benutzeroberfläche zu starten, geben Sie Ihren Benutzern, Gruppen oder Rollen die folgenden Zugriffsberechtigungen für die `emr-serverless:GetDashboardForJobRun` API:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-serverless:GetDashboardForJobRun"
      ],
      "Resource": [
        "*"
      ],
      "Sid": "AllowEMRSERVERLESSGetdashboardforjobrun"
    }
  ]
}
```

Berechtigungen für Verschlüsselungsschlüssel für Amazon S3 und verwalteten Speicher

Wenn Sie Protokolle mit Ihrem eigenen Verschlüsselungsschlüssel entweder im verwalteten Speicher oder in Ihren S3-Buckets verschlüsseln, konfigurieren Sie die KMS-Schlüsselberechtigungen wie folgt.

Der `emr-serverless.amazonaws.com` Prinzipal muss in der Richtlinie für den KMS-Schlüssel über die folgenden Berechtigungen verfügen:

```
{
    "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"
       }
     }
 }
```

Aus Sicherheitsgründen empfehlen wir, der KMS-Schlüsselrichtlinie einen `aws:SourceArn` Bedingungsschlüssel hinzuzufügen. Der globale IAM-Bedingungsschlüssel `aws:SourceArn` trägt dazu bei, dass EMR Serverless den KMS-Schlüssel nur für einen Anwendungs-ARN verwendet.

Die IAM-Richtlinie der Job-Runtime-Rolle muss über die folgenden Berechtigungen verfügen:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:GenerateDataKey",
        "kms:Decrypt"
      ],
      "Resource": [
        "arn:aws:kms:*:123456789012:key/12345678-1234-1234-1234-123456789012"
      ],
      "Sid": "AllowKMSGeneratedatakey"
    }
  ]
}
```

Berechtigungen für Verschlüsselungsschlüssel für Amazon CloudWatch

Verwenden Sie die folgende IAM-Richtlinie für die Job-Runtime-Rolle, um den KMS-Schlüssel-ARN Ihrer Protokollgruppe zuzuordnen.

JSON

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

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:AssociateKmsKey"
      ],
      "Resource": [
        "arn:aws:logs:*:123456789012:log-group:my-log-group-name:*"
      ],
      "Sid": "AllowLOGSAssociatekmskey"
    }
  ]
}
```

Konfigurieren Sie die KMS-Schlüsselrichtlinie, um Amazon KMS-Berechtigungen zu gewähren CloudWatch:

JSON

```
{
  "Version":"2012-10-17",
  "Id": "key-default-1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "ArnLike": {
          "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:*:123456789012:*"
        }
      },
      "Sid": "AllowKMSDecrypt"
    }
  ]
}
```

Apache Log4j2-Eigenschaften für Amazon EMR Serverless konfigurieren

Auf dieser Seite wird beschrieben, wie Sie benutzerdefinierte Apache Log4j 2.x-Eigenschaften für EMR Serverless-Jobs unter konfigurieren. `StartJobRun` Wenn Sie Log4j-Klassifizierungen auf Anwendungsebene konfigurieren möchten, finden Sie weitere Informationen unter. Standardanwendungskonfiguration für EMR Serverless

Konfigurieren Sie die Spark Log4J2-Eigenschaften für Amazon EMR Serverless

Mit den Amazon EMR-Versionen 6.8.0 und höher können Sie die Eigenschaften von Apache Log4j 2.x anpassen, um detaillierte Protokollkonfigurationen zu spezifizieren. Dies vereinfacht die Fehlerbehebung Ihrer Spark-Jobs auf EMR Serverless. Verwenden Sie die Klassifizierungen `spark-driver-log4j2` und`spark-executor-log4j2`, um diese Eigenschaften zu konfigurieren.

Themen

- Log4j2-Klassifizierungen für Spark
- Log4j2-Konfigurationsbeispiel für Spark
- Log4j2 in Spark-Beispielaufträgen
- Überlegungen zu Log4j2 für Spark

Log4j2-Klassifizierungen für Spark

Um die Spark-Protokollkonfigurationen anzupassen, verwenden Sie die folgenden Klassifizierungen mit. `applicationConfiguration` Verwenden Sie Folgendes, um die Log4j 2.x-Eigenschaften zu konfigurieren. `properties`

`spark-driver-log4j2`

Diese Klassifizierung legt die Werte in der `log4j2.properties` Datei für den Treiber fest.

`spark-executor-log4j2`

Diese Klassifizierung legt die Werte in der `log4j2.properties` Datei für den Executor fest.

Log4j2-Konfigurationsbeispiel für Spark

Das folgende Beispiel zeigt, wie Sie einen Spark-Job einreichen, `applicationConfiguration` um die Log4J2-Konfigurationen für den Spark-Treiber und -Executor anzupassen.

Informationen zur Konfiguration von Log4j-Klassifizierungen auf Anwendungsebene und nicht erst, wenn Sie den Job einreichen, finden Sie unter. Standardanwendungskonfiguration für EMR Serverless

```
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 in Spark-Beispielaufträgen

Die folgenden Codebeispiele zeigen, wie Sie eine Spark-Anwendung erstellen, während Sie eine benutzerdefinierte Log4j2-Konfiguration für die Anwendung initialisieren.

Python

Example- Log4j2 für einen Spark-Job mit Python verwenden

```
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()
```

Verwenden Sie die folgende Konfiguration, um Log4j2 für den Treiber anzupassen, wenn Sie einen Spark-Job ausführen:

```
{
   "classification": "spark-driver-log4j2",
      "properties": {
          "rootLogger.level":"error", // only display Spark error logs
```

```
          "logger.PySparkApp.level": "info",
          "logger.PySparkApp.name": "PySparkApp"
      }
}
```

Scala

Example- Verwenden von Log4j2 für einen Spark-Job mit 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()
  }
}
```

Verwenden Sie die folgende Konfiguration, um Log4j2 für den Treiber anzupassen, wenn Sie einen Spark-Job ausführen:

```
{
   "classification": "spark-driver-log4j2",
      "properties": {
          "rootLogger.level":"error", // only display Spark error logs
          "logger.ExampleClass.level": "info",
          "logger.ExampleClass.name": "ExampleClass"
      }
}
```

Überlegungen zu Log4j2 für Spark

Die folgenden Log4j2.x-Eigenschaften sind für Spark-Prozesse nicht konfigurierbar:

- `rootLogger.appenderRef.stdout.ref`
- `appender.console.type`
- `appender.console.name`
- `appender.console.target`
- `appender.console.layout.type`
- `appender.console.layout.pattern`

Ausführliche Informationen zu den Log4j2.x-Eigenschaften, die konfiguriert werden, finden Sie in der Datei unter. `log4j2.properties.template` GitHub

Serverlose EMR-Überwachung

In diesem Abschnitt wird beschrieben, wie Sie Ihre Amazon EMR Serverless-Anwendungen und -Jobs überwachen können.

Themen

- Überwachung von serverlosen EMR-Anwendungen und -Jobs
- Überwachen Sie Spark-Metriken mit Amazon Managed Service für Prometheus
- Metriken zur serverlosen Nutzung von EMR

Überwachung von serverlosen EMR-Anwendungen und -Jobs

Mit Amazon CloudWatch Metrics for EMR Serverless können Sie CloudWatch Metriken von einer Minute erhalten und auf CloudWatch Dashboards zugreifen, um auf den near-real-time Betrieb und die Leistung Ihrer EMR Serverless-Anwendungen zuzugreifen.

EMR Serverless sendet Metriken an CloudWatch jede Minute. EMR Serverless gibt diese Metriken auf Anwendungsebene sowie auf Job-, Worker-Typ- und Arbeitsebene aus. capacity-allocation-type

Verwenden Sie zunächst die EMR Serverless CloudWatch Dashboard-Vorlage, die im EMR GitHub Serverless-Repository bereitgestellt wird, und stellen Sie sie bereit.

> **Note**
>
> Bei serverlosen interaktiven EMR-Workloads ist nur die Überwachung auf Anwendungsebene aktiviert und sie verfügen über eine neue Worker-Typ-Dimension. `Spark_Kernel` Um Ihre interaktiven Workloads zu überwachen und zu debuggen, greifen Sie von Ihrem EMR Studio Workspace aus auf die Protokolle und die Apache Spark-Benutzeroberfläche zu.

Überwachung von Kennzahlen

> **Important**
>
> Wir strukturieren die Anzeige unserer Kennzahlen neu, um Dimensionen hinzuzufügen `ApplicationName` und `JobName` sie als Dimensionen hinzuzufügen. Für Version 7.10 und höher werden die älteren Metriken nicht mehr aktualisiert. Für EMR-Versionen unter 7.10 sind die älteren Metriken weiterhin verfügbar.

Aktuelle Abmessungen

In der folgenden Tabelle werden die im Namespace verfügbaren EMR-Serverless-Dimensionen beschrieben. `AWS/EMR Serverless`

Dimensionen für EMR Serverless-Metriken

Dimension	Description	
`ApplicationId`	Filtert nach allen Metriken einer serverlosen EMR-Anwendung unter Verwendung der Anwendungs-ID.	
`ApplicationName`	Filtert nach allen Metriken einer serverlosen EMR-Anwendung, die den Namen verwenden. Wenn der Name nicht angegeben wird oder Nicht-ASCII-Zeichen enthält,	

Dimension	Description	
	wird er als [Unspecified] veröffentlicht.	
`JobId`	Filtert nach allen Metriken eines EMR Serverless die Job-Run-ID.	
`JobName`	Filtert nach allen Metriken eines EMR Serverless-Jobs, der unter Verwendung des Namens ausgeführt wird. Wenn der Name nicht angegeben wird oder Nicht-ASCII-Zeichen enthält, wird er als [Unspecified] veröffentlicht.	
`WorkerType`	Filtert nach allen Metriken eines bestimmten Arbeitertyps. Sie können beispielsweise nach `SPARK_DRIVER` und `SPARK_EXECUTORS` nach Spark-Jobs filtern.	
`CapacityAllocationType`	Filtert nach allen Metriken eines bestimmten Kapazitätszuweisungstyps. Sie können beispielsweise nach `PreInitCapacity` vorinitialisierter Kapazität und `OnDemandCapacity` nach allem anderen filtern.	

Überwachung auf Anwendungsebene

Sie können die Kapazitätsnutzung auf Ebene der EMR-Serverless-Anwendung mit CloudWatch Amazon-Metriken überwachen. Sie können auch eine einzige Anzeige zur Überwachung der Anwendungskapazitätsnutzung in einem CloudWatch Dashboard einrichten.

Metriken für serverlose EMR-Anwendungen

Metrik	Description	Einheit	Dimension
`MaxCPUAllowed`	Die maximal zulässige CPU-Anzahl für die Anwendung.	vCPU	`ApplicationId`, `ApplicationName`
`MaxMemoryAllowed`	Der maximal zulässige Arbeitsspeicher in GB für die Anwendung.	Gigabyte (GB)	`ApplicationId`, `ApplicationName`
`MaxStorageAllowed`	Der maximal zulässige Speicherplatz in GB für die Anwendung.	Gigabyte (GB)	`ApplicationId`, `ApplicationName`
`CPUAllocated`	Die Gesamtzahl der zugewiesenen V. CPUs	vCPU	`ApplicationId`, `ApplicationName`, `WorkerType`, `CapacityAllocationType`
`IdleWorkerCount`	Die Gesamtzahl der untätigen Arbeitnehmer.	Anzahl	`ApplicationId`, `ApplicationName`, `WorkerType`, `CapacityAllocationType`
`MemoryAllocated`	Der gesamte zugewiesene Speicher in GB.	Gigabyte (GB)	`ApplicationId`, `ApplicationName`, `WorkerTyp`

Metrik	Description	Einheit	Dimension
			e , CapacityA llocationType
PendingCr eationWor kerCount	Die Gesamtzahl der Mitarbeiter, deren Erstellung noch aussteht.	Anzahl	ApplicationId , ApplicationName , WorkerTyp e , CapacityA llocationType
RunningWo rkerCount	Die Gesamtzahl der Mitarbeiter, die von der Anwendung verwendet werden.	Anzahl	ApplicationId , ApplicationName , WorkerTyp e , CapacityA llocationType
StorageAl located	Der gesamte zugewiesene Festplatt enspeicher in GB.	Gigabyte (GB)	ApplicationId , ApplicationName , WorkerTyp e , CapacityA llocationType
TotalWork erCount	Die Anzahl der insgesamt verfügbar en Arbeitskräfte.	Anzahl	ApplicationId , ApplicationName , WorkerTyp e , CapacityA llocationType

Überwachung auf Arbeitsebene

Amazon EMR Serverless sendet jede Minute die folgenden Metriken auf Amazon CloudWatch Auftragsebene. Sie können auf die Metrikwerte für aggregierte Auftragsausführungen nach Status der Auftragsausführung zugreifen. Die Einheit für jede der Metriken ist Anzahl.

EMR Serverless Metriken auf Jobebene

Metrik	Description	Dimension
`SubmittedJobs`	Die Anzahl der Jobs mit dem Status „Gesendet“.	`ApplicationId` , `ApplicationName`
`PendingJobs`	Die Anzahl der Jobs mit dem Status Ausstehend.	`ApplicationId` , `ApplicationName`
`ScheduledJobs`	Die Anzahl der Jobs mit dem Status „Geplant“.	`ApplicationId` , `ApplicationName`
`RunningJobs`	Die Anzahl der Jobs im Status Wird ausgeführt.	`ApplicationId` , `ApplicationName`
`SuccessJobs`	Die Anzahl der Jobs im Status Erfolgreich.	`ApplicationId` , `ApplicationName`
`FailedJobs`	Die Anzahl der Jobs mit dem Status Fehlgeschlagen.	`ApplicationId` , `ApplicationName`
`CancellingJobs`	Die Anzahl der Jobs mit dem Status „Storniert“.	`ApplicationId` , `ApplicationName`
`CancelledJobs`	Die Anzahl der Jobs mit dem Status Storniert.	`ApplicationId` , `ApplicationName`

Sie können Engine-spezifische Metriken für ausgeführte und abgeschlossene EMR Serverless-Jobs mit einer Engine-spezifischen Anwendung überwachen. UIs Wenn Sie auf die Benutzeroberfläche für einen laufenden Job zugreifen, wird die Benutzeroberfläche der Live-Anwendung mit Aktualisierungen in Echtzeit angezeigt. Wenn Sie auf die Benutzeroberfläche für einen abgeschlossenen Job zugreifen, wird die persistente App-Benutzeroberfläche angezeigt.

Ausführen von Aufgaben

Greifen Sie für Ihre laufenden EMR Serverless-Jobs auf eine Echtzeitschnittstelle zu, die Engine-spezifische Metriken bereitstellt. Sie können entweder die Apache Spark-Benutzeroberfläche oder die Hive Tez-Benutzeroberfläche verwenden, um Ihre Jobs zu überwachen und zu debuggen. Um auf

diese zuzugreifen UIs, verwenden Sie die EMR Studio-Konsole oder fordern Sie einen sicheren URL-Endpunkt mit dem an AWS Command Line Interface.

Abgeschlossene Jobs

Verwenden Sie für Ihre abgeschlossenen EMR Serverless-Jobs den Spark History Server oder die Persistent Hive Tez-Benutzeroberfläche, um auf Jobdetails, Phasen, Aufgaben und Metriken für Spark- oder Hive-Jobausführungen zuzugreifen. Um auf diese zuzugreifen UIs, verwenden Sie die EMR Studio-Konsole oder fordern Sie einen sicheren URL-Endpunkt mit dem AWS Command Line Interface an.

Überwachung auf Arbeitgeberebene

Amazon EMR Serverless sendet die folgenden Metriken auf Job-Worker-Ebene, die im `AWS/EMRServerless` Namespace und in der `Job Worker Metrics` Metrikgruppe verfügbar sind, an Amazon. CloudWatch EMR Serverless sammelt Datenpunkte von einzelnen Mitarbeitern während der Auftragsausführung auf Auftragsebene, Arbeitstyp und Ebene. capacity-allocation-type Sie können diese Dimension verwenden`ApplicationId`, um mehrere Jobs zu überwachen, die zu derselben Anwendung gehören.

> **Note**
>
> Um bei der Anzeige der Metriken in der CloudWatch Amazon-Konsole den gesamten CPU- und Speicherverbrauch eines EMR-Serverless-Jobs anzuzeigen, verwenden Sie die Statistik als Summe und Zeitraum als 1 Minute.

EMR Serverless Job Metriken auf Mitarbeiterebene

Metrik	Description	Einheit	Dimension	
`WorkerCpuAllocated`	Die Gesamtzahl der vCPU-Kerne, die Workern in einer Jobausführung zugewiesen wurden.	vCPU	`JobId`, `JobName`, `ApplicationId`, `ApplicationName`, `WorkerType`, und `CapacityA`	

Metrik	Description	Einheit	Dimension
			llocation Type
WorkerCpu Used	Die Gesamtzahl der vCPU-Kerne, die von Workern in einem Joblauf verwendet werden.	vCPU	JobId, JobName, Applicati onId , Applicati onName , WorkerType , und CapacityA llocation Type
WorkerMem oryAlloca ted	Der Gesamtspe icher in GB, der Workern in einer Auftragsa usführung zugewiesen wurde.	Gigabyte (GB)	JobId, JobName, Applicati onId , Applicati onName , WorkerType , und CapacityA llocation Type
WorkerMem oryUsed	Der gesamte Arbeitsspeicher in GB, der von Arbeitern bei einer Auftragsa usführung genutzt wird.	Gigabyte (GB)	JobId, JobName, Applicati onId , Applicati onName , WorkerType , und CapacityA llocation Type

Metrik	Description	Einheit	Dimension	
`WorkerEphemeralStorageAllocated`	Die Anzahl der Byte an flüchtigem Speicher, die Workern in einer Auftragsausführung zugewiesen wurden.	Gigabyte (GB)	`JobId`, `JobName`, `ApplicationId`, `ApplicationName`, `WorkerType`, und `CapacityAllocationType`	
`WorkerEphemeralStorageUsed`	Die Anzahl der Byte an flüchtigem Speicher, die von Arbeitern bei einer Auftragsausführung verwendet werden.	Gigabyte (GB)	`JobId`, `JobName`, `ApplicationId`, `ApplicationName`, `WorkerType`, und `CapacityAllocationType`	
`WorkerStorageReadBytes`	Die Anzahl der Byte, die von Arbeitern während einer Auftragsausführung aus dem Speicher gelesen wurden.	Bytes	`JobId`, `JobName`, `ApplicationId`, `ApplicationName`, `WorkerType`, und `CapacityAllocationType`	

Metrik	Description	Einheit	Dimension	
`WorkerStorageWriteBytes`	Die Anzahl der Byte, die während einer Auftragsausführung von Workern in den Speicher geschrieben wurden.	Bytes	`JobId`, `JobName`, `ApplicationId` , `ApplicationName` , `WorkerType` , und `CapacityAllocationType`	

In den folgenden Schritten wird beschrieben, wie Sie auf die verschiedenen Arten von Metriken zugreifen können.

Console

So greifen Sie mit der Konsole auf die Benutzeroberfläche Ihrer Anwendung zu

1. Navigieren Sie zu Ihrer EMR Serverless-Anwendung im EMR Studio. Folgen Sie den Anweisungen unter Erste Schritte von der Konsole aus.
2. So greifen Sie auf Engine-spezifische Anwendungen UIs und Protokolle für einen laufenden Job zu:
   a. Wählen Sie einen Job mit einem Status aus`RUNNING`.
   b. Wählen Sie die Job auf der Seite mit den Bewerbungsdetails aus, oder navigieren Sie zur Seite mit den Stellendetails für Ihre Stelle.
   c. Wählen Sie im Dropdownmenü „Benutzeroberfläche anzeigen“ entweder Spark UI oder Hive Tez UI aus, um zur Anwendungsoberfläche für Ihren Jobtyp zu gelangen.
   d. Um auf die Spark-Engine-Logs zuzugreifen, navigieren Sie in der Spark-Benutzeroberfläche zur Registerkarte Executors und wählen Sie den Link Logs für den Treiber. Um auf die Hive-Engine-Logs zuzugreifen, wählen Sie in der Hive Tez-Benutzeroberfläche den Link Logs für die entsprechende DAG.
3. So greifen Sie auf maschinenspezifische Anwendungen UIs und Protokolle für einen abgeschlossenen Job zu:

a. Wählen Sie einen Job mit einem Status aus`SUCCESS`.

b. Wählen Sie die Job auf der Seite mit den Bewerbungsdetails Ihrer Bewerbung aus oder navigieren Sie zur Seite mit den Stellendetails der Stelle.

c. Wählen Sie im Dropdownmenü „Benutzeroberfläche anzeigen“ entweder Spark History Server oder Persistent Hive Tez UI aus, um zur Anwendungsoberfläche für Ihren Jobtyp zu gelangen.

d. Um auf die Spark-Engine-Logs zuzugreifen, navigieren Sie in der Spark-Benutzeroberfläche zur Registerkarte Executors und wählen Sie den Link Logs für den Treiber. Um auf die Hive-Engine-Logs zuzugreifen, wählen Sie in der Hive Tez-Benutzeroberfläche den Link Logs für die entsprechende DAG.

AWS CLI

Um auf die Benutzeroberfläche Ihrer Anwendung zuzugreifen, verwenden Sie AWS CLI

- Rufen Sie die `GetDashboardForJobRun` API auf, um eine URL zu generieren, über die Sie auf die Benutzeroberfläche Ihrer Anwendung für laufende und abgeschlossene Jobs zugreifen können.

```
aws emr-serverless get-dashboard-for-job-run /
--application-id <application-id> /
--job-run-id <job-id>
```

Die URL, die Sie generieren, ist eine Stunde lang gültig.

Überwachen Sie Spark-Metriken mit Amazon Managed Service für Prometheus

Mit den Amazon EMR-Versionen 7.1.0 und höher können Sie EMR Serverless in Amazon Managed Service for Prometheus integrieren, um Apache Spark-Metriken für EMR Serverless-Jobs und -Anwendungen zu sammeln. Diese Integration ist verfügbar, wenn Sie einen Job einreichen oder eine Anwendung mit der AWS Konsole, der EMR Serverless API oder dem erstellen. AWS CLI

Voraussetzungen

Bevor Sie Ihre Spark-Metriken an Amazon Managed Service for Prometheus übermitteln können, müssen Sie die folgenden Voraussetzungen erfüllen.

- Erstellen Sie einen Amazon Managed Service for Prometheus Workspace. Dieser Workspace dient als Aufnahme-Endpunkt. Notieren Sie sich die URL, die für Endpoint — Remote Write-URL angezeigt wird. Sie müssen die URL angeben, wenn Sie Ihre EMR Serverless-Anwendung erstellen.
- Um Amazon Managed Service for Prometheus zu Überwachungszwecken Zugriff auf Ihre Jobs zu gewähren, fügen Sie Ihrer Job-Ausführungsrolle die folgende Richtlinie hinzu.

```
{
    "Sid": "AccessToPrometheus",
    "Effect": "Allow",
    "Action": ["aps:RemoteWrite"],
    "Resource": "arn:aws:aps:<AWS_REGION>:<AWS_ACCOUNT_ID>:workspace/<WORKSPACE_ID>"
}
```

Einrichtung

So verwenden Sie die AWS Konsole, um eine Anwendung zu erstellen, die in Amazon Managed Service for Prometheus integriert ist

1. Informationen zum Erstellen einer Anwendung finden Sie unter Erste Schritte mit Amazon EMR Serverless.
2. Wählen Sie beim Erstellen einer Anwendung die Option Benutzerdefinierte Einstellungen verwenden und konfigurieren Sie dann Ihre Anwendung, indem Sie die Informationen in die Felder eingeben, die Sie konfigurieren möchten.
3. Wählen Sie unter Anwendungsprotokolle und Metriken die Option Engine-Metriken an Amazon Managed Service for Prometheus liefern aus und geben Sie dann Ihre Remote-Write-URL an.
4. Geben Sie alle anderen gewünschten Konfigurationseinstellungen an und wählen Sie dann Anwendung erstellen und starten.

Verwenden Sie die AWS CLI oder EMR Serverless API

Sie können auch die AWS CLI oder EMR Serverless API verwenden, um Ihre EMR Serverless-Anwendung in Amazon Managed Service for Prometheus zu integrieren, wenn Sie die Befehle oder die Befehle ausführen. `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"
        }
    }
}'
```

Die Aufnahme `prometheusMonitoringConfiguration` in Ihrem Befehl bedeutet, dass EMR Serverless den Spark-Job mit einem Agenten ausführen muss, der die Spark-Metriken sammelt und sie auf Ihren `remoteWriteUrl` Endpunkt für Amazon Managed Service for Prometheus schreibt.

Anschließend können Sie die Spark-Metriken in Amazon Managed Service for Prometheus für Visualisierungen, Benachrichtigungen und Analysen verwenden.

Erweiterte Konfigurationseigenschaften

EMR Serverless verwendet eine Komponente innerhalb von Spark, die `PrometheusServlet` zur Erfassung von Spark-Metriken benannt ist, und übersetzt Leistungsdaten in Daten, die mit Amazon Managed Service for Prometheus kompatibel sind. Standardmäßig legt EMR Serverless Standardwerte in Spark fest und analysiert Treiber- und Executor-Metriken, wenn Sie einen Job mit senden. `PrometheusMonitoringConfiguration`

In der folgenden Tabelle werden alle Eigenschaften beschrieben, die konfiguriert werden, wenn ein Spark-Job übermittelt wird, der Metriken an Amazon Managed Service for Prometheus sendet.

Spark-Eigenschaft	Standardwert	Description
`spark.metrics.conf.*.sink.prometheusServlet.class`	org.apache.spark.metrics.sink.PrometheusServlet	Die Klasse, die Spark verwendet, um Metriken an Amazon Managed Service for Prometheus zu senden. Um das Standardverhalten zu überschreiben, geben Sie Ihre eigene benutzerdefinierte Klasse an.
`spark.metrics.conf.*.source.jvm.class`	org.apache.spark.metrics.source. JvmSource	Die Klasse, die Spark verwendet, um wichtige Metriken von der zugrunde liegenden virtuellen Java-Maschine zu sammeln und zu senden. Um die Erfassung von JVM-Metriken zu beenden, deaktivieren Sie diese Eigenschaft, indem Sie sie auf eine leere Zeichenfolge setzen, z. B. "" Um das Standardverhalten zu überschreiben, geben Sie

Spark-Eigenschaft	Standardwert	Description
		Ihre eigene benutzerdefinierte Klasse an.
`spark.metrics.conf.driver.sink.prometheusServlet.path`	/metrics/prometheus	Die eindeutige URL, die Amazon Managed Service for Prometheus verwendet, um Metriken vom Treiber zu sammeln. Um das Standardv erhalten zu überschreiben, geben Sie Ihren eigenen Pfad an. Um die Erfassung von Treibermetriken zu beenden, deaktivieren Sie diese Eigenschaft, indem Sie sie auf eine leere Zeichenfolge setzen, z. "" B.
`spark.metrics.conf.executor.sink.prometheusServlet.path`	/metrics/executor/prometheus	Die eindeutige URL, die Amazon Managed Service for Prometheus verwendet, um Metriken vom Testament svollstrecker zu sammeln. Um das Standardverhalten zu überschreiben, geben Sie Ihren eigenen Pfad an. Um die Erfassung von Executor-Metriken zu beenden, deaktivieren Sie diese Eigenschaft, indem Sie sie auf eine leere Zeichenfolge setzen, z. B. ""

Weitere Informationen zu den Spark-Metriken finden Sie unter Apache Spark-Metriken.

Überlegungen und Einschränkungen

Wenn Sie Amazon Managed Service for Prometheus verwenden, um Metriken von EMR Serverless zu sammeln, sollten Sie die folgenden Überlegungen und Einschränkungen berücksichtigen.

- Support für die Verwendung von Amazon Managed Service for Prometheus mit EMR Serverless ist nur dort verfügbar, AWS-Regionen wo Amazon Managed Service für Prometheus allgemein verfügbar ist.
- Die Ausführung des Agenten zur Erfassung von Spark-Metriken auf Amazon Managed Service for Prometheus erfordert mehr Ressourcen von den Mitarbeitern. Wenn Sie eine kleinere Worker-Größe wählen, z. B. einen vCPU-Worker, kann sich Ihre Job-Laufzeit verlängern.
- Support für die Verwendung von Amazon Managed Service for Prometheus mit EMR Serverless ist nur für Amazon EMR-Versionen 7.1.0 und höher verfügbar.
- Amazon Managed Service for Prometheus muss in demselben Konto bereitgestellt werden, in dem Sie EMR Serverless ausführen, um Metriken zu sammeln.

Metriken zur serverlosen Nutzung von EMR

Sie können die CloudWatch Amazon-Nutzungsmetriken verwenden, um einen Überblick über die Ressourcen zu erhalten, die Ihr Konto verwendet. Verwenden Sie diese Metriken, um Ihre Servicenutzung in CloudWatch Grafiken und Dashboards zu visualisieren.

Die Metriken zur Nutzung von EMR Serverless entsprechen Service Quotas. Sie können Alarme konfigurieren, mit denen Sie benachrichtigt werden, wenn sich Ihre Nutzung einem Servicekontingent nähert. Weitere Informationen finden Sie unter Service Quotas und CloudWatch Amazon-Alarme im Service Quotas Quotas-Benutzerhandbuch.

Weitere Informationen zu EMR Serverless Service Quotas finden Sie unter. Endpunkte und Kontingente für EMR Serverless

Metriken zur Nutzung von Servicekontingenten für EMR Serverless

EMR Serverless veröffentlicht die folgenden Messdaten zur Nutzung von Dienstkontingenten im `AWS/Usage` Namespace.

Metrik	Description
`ResourceCount`	Die Gesamtanzahl der angegebenen Ressource, die auf Ihrem Konto ausgeführt wird. Die Ressource wird durch die Dimensionen definiert, die der Metrik zugeordnet sind.

Dimensionen für Metriken zur Nutzung der Kontingente für EMR Serverless Service

Sie können die folgenden Dimensionen verwenden, um die von EMR Serverless veröffentlichten Nutzungsmetriken zu verfeinern.

Dimension	Wert	Description
`Service`	EMR Serverlos	Der Name des AWS-Service , der die Ressource enthält.
`Type`	Ressource	Der Entitätstyp, den EMR Serverless meldet.
`Resource`	vCPU	Der Ressourcentyp, den EMR Serverless verfolgt.
`Class`	Keine	Die Ressourcenklasse, die EMR Serverless verfolgt.

Automatisierung von EMR Serverless mit Amazon EventBridge

Sie können Amazon EventBridge damit Ihre Systemereignisse automatisieren AWS-Services und automatisch darauf reagieren, z. B. Probleme mit der Anwendungsverfügbarkeit oder Ressourcenänderungen. EventBridge liefert nahezu in Echtzeit einen Stream von Systemereignissen, die Änderungen an Ihren AWS Ressourcen beschreiben. Sie können einfache Regeln schreiben, um anzugeben, welche Ereignisse für Sie interessant sind und welche automatisierten Aktionen durchgeführt werden sollen, wenn sich für ein Ereignis eine Übereinstimmung mit einer Regel ergibt. Mit EventBridge können Sie automatisch:

- Eine AWS Lambda Funktion aufrufen
- Ein Ereignis an Amazon Kinesis Data Streams weiterleiten
- Aktivieren Sie eine AWS Step Functions Zustandsmaschine
- Ein Amazon SNS SNS-Thema oder eine Amazon SQS SQS-Warteschlange benachrichtigen

Wenn Sie beispielsweise EventBridge mit EMR Serverless arbeiten, können Sie eine AWS Lambda Funktion aktivieren, wenn ein ETL-Job erfolgreich ist, oder ein Amazon SNS SNS-Thema benachrichtigen, wenn ein ETL-Job fehlschlägt.

EMR Serverless sendet vier Arten von Ereignissen aus:

- Ereignisse zur Änderung des Anwendungsstatus — Ereignisse, die jede Statusänderung einer Anwendung auslösen. Weitere Informationen zu Anwendungsstatus finden Sie unterStatus der Anwendung.
- Ereignisse zur Statusänderung bei Auftragsausführung — Ereignisse, die jede Statusänderung eines Joblaufs auslösen. Weitere Informationen zu finden Sie unterStatus von Aufgabenausführungen.
- Ereignisse zur Wiederholung von Auftragsausführungen — Ereignisse, die bei jedem erneuten Versuch einer Auftragsausführung von Amazon EMR Serverless 7.1.0 und höher ausgelöst werden.
- Aktualisierungsereignisse für die Job-Ressourcenauslastung — Ereignisse, die in Intervallen von fast 30 Minuten Updates zur Ressourcennutzung für einen Job ausgeben.

Beispiel für serverlose EMR-Ereignisse EventBridge

Von EMR Serverless gemeldete Ereignisse haben den Wert `aws.emr-serverless` zugewiesen`source`, wie in den folgenden Beispielen.

Ereignis zur Änderung des Anwendungsstatus

Das folgende Beispielereignis zeigt eine Anwendung im `CREATING` Status.

```
{
    "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
        }
    }
}
```

Ereignis zur Änderung des Status des ausgeführten Job

Das folgende Beispielereignis zeigt eine Auftragsausführung, bei der vom `SCHEDULED` Status in den `RUNNING` Status gewechselt wird.

```
{
    "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"
    }
}
```

Ereignis zur Wiederholung eines Auftrags

Im Folgenden finden Sie ein Beispiel für ein Ereignis zur Wiederholung eines Auftrags.

```
{
    "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"
    }
}
```

Aktualisierung der Auslastung der Arbeitsressourcen

Das folgende Beispielereignis zeigt das letzte Update zur Ressourcennutzung für einen Job, der nach der Ausführung in einen Terminalstatus übergegangen ist.


```
{
    "version": "0",
    "id": "00df3ec6-5da1-36e6-ab71-20f0de68f8a0",
    "detail-type": "EMR Serverless Job Resource Utilization Update",
    "source": "aws.emr-serverless",
    "account": "123456789012",
    "time": "2022-05-31T21:07:42Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:emr-serverless:us-east-1:123456789012:/applications/00f1982r1uukb925/
jobruns/00f1cbn5g4bb0c01"
    ],
    "detail": {
        "applicationId": "00f1982r1uukb925",
        "jobRunId": "00f1cbn5g4bb0c01",
        "attempt": 1,
        "mode": "BATCH",
        "createdAt": "2022-05-31T21:07:25.325900Z",
        "startedAt": "2022-05-31T21:07:26.123Z",
        "calculatedFrom": "2022-05-31T21:07:42.299487Z",
        "calculatedTo": "2022-05-31T21:07:30.325900Z",
        "resourceUtilizationFinal": true,
        "resourceUtilizationForInterval": {
            "vCPUHour": 0.023,
            "memoryGBHour": 0.114,
            "storageGBHour": 0.228
        },
        "billedResourceUtilizationForInterval": {
            "vCPUHour": 0.067,
            "memoryGBHour": 0.333,
            "storageGBHour": 0
        },
        "totalResourceUtilization": {
            "vCPUHour": 0.023,
            "memoryGBHour": 0.114,
            "storageGBHour": 0.228
        },
        "totalBilledResourceUtilization": {
            "vCPUHour": 0.067,
            "memoryGBHour": 0.333,
```

```
            "storageGBHour": 0
        }
    }
}
```

Das Feld StartedAt ist nur dann vorhanden, wenn der Job in den Status Running übergegangen ist.

Taggen von Ressourcen

Weisen Sie jeder Ressource mithilfe von Tags Ihre eigenen Metadaten zu, um Ihre EMR Serverless-Ressourcen zu verwalten. Dieser Abschnitt bietet einen Überblick über die Tag-Funktionen und zeigt, wie Tags erstellt werden.

Themen

Was ist ein Tag?

Ein Tag ist eine Bezeichnung, die Sie einer AWS Ressource zuweisen. Jedes Tag besteht aus einem Schlüssel und einem Wert, die Sie beide selbst definieren. Mithilfe von Tags können Sie Ihre AWS Ressourcen nach Attributen wie Zweck, Eigentümer und Umgebung kategorisieren. Wenn Sie über viele Ressourcen desselben Typs verfügen, können Sie anhand der ihr zugewiesenen Tags schnell eine bestimmte Ressource identifizieren. Definieren Sie beispielsweise eine Reihe von Tags für Ihre serverlosen Amazon EMR-Anwendungen, um den Besitzer und die Stack-Ebene jeder Anwendung zu verfolgen. Wir empfehlen Ihnen, für jeden Ressourcentyp einen konsistenten Satz von Tag-Schlüsseln zu entwickeln.

Tags werden nicht automatisch Ihren Ressourcen zugewiesen. Nachdem Sie einer Ressource ein Tag hinzugefügt haben, können Sie jederzeit den Wert eines Tags ändern oder das Tag aus der Ressource entfernen. Tags haben für Amazon EMR Serverless keine semantische Bedeutung und werden ausschließlich als Zeichenketten interpretiert. Wenn Sie ein Tag (Markierung) mit demselben Schlüssel wie ein vorhandener Tag (Markierung) für die Ressource hinzufügen, wird der alte Wert mit dem neuen überschrieben.

Wenn Sie IAM verwenden, können Sie steuern, welche Benutzer in Ihrem AWS Konto berechtigt sind, Tags zu verwalten. Beispiele für tagbasierte Richtlinien zur Zugriffskontrolle finden Sie unter. Richtlinien für Tag-basierte Zugriffskontrolle

Markieren von Ressourcen

Sie können neue oder bestehende Anwendungen und Jobausführungen taggen. Wenn Sie die Amazon EMR Serverless API, das oder ein AWS SDK verwenden AWS CLI, können Sie Tags auf neue Ressourcen anwenden, indem Sie den `tags` Parameter für die entsprechende API-Aktion verwenden. Sie können auch über die `TagResource`-API Tags auf Ressourcen anwenden.

Sie können einige Aktionen zum Erstellen von Ressourcen verwenden, um Tags für eine Ressource anzugeben, wenn die Ressource erstellt wird. Wenn in diesem Fall die Tags nicht angewendet werden können, während die Ressource erstellt wird, kann die Ressource nicht erstellt werden. Auf diese Weise wird sichergestellt, dass Ressourcen, die Sie bei der Erstellung markieren möchten, entweder mit angegebenen Tags oder gar nicht erstellt werden. Wenn Sie Ressourcen bei der Erstellung taggen, müssen Sie nach dem Erstellen einer Ressource keine benutzerdefinierten Tagging-Skripts ausführen.

In der folgenden Tabelle werden die Amazon EMR Serverless-Ressourcen beschrieben, die markiert werden können.

Markierbare -Ressourcen

Ressource	Unterstützt Tags (Markierungen)	Unterstützt Tag-Propagierung	Unterstützt Tagging bei der Erstellung (Amazon EMR Serverless API und SDK AWS CLI) AWS	API für die Erstellung (Tags können während der Erstellung hinzugefügt werden)
Anwendung	Ja	Nein. Mit einer Anwendung verknüpfte Tags werden nicht auf Auftragsausführungen übertragen, die an diese Anwendung	Ja	`CreateApplication`

Ressource	Unterstützt Tags (Markierungen)	Unterstützt Tag-Propagierung	Unterstützt Tagging bei der Erstellung (Amazon EMR Serverless API und SDK AWS CLI) AWS	API für die Erstellung (Tags können während der Erstellung hinzugefügt werden)
		gesendet werden.		
Aufgabenausführung	Ja	Nein	Ja	`StartJobRun`

Einschränkungen beim Markieren

Die folgenden grundlegenden Einschränkungen gelten für Tags:

- Jede Ressource kann maximal 50 vom Benutzer erstellte Tags haben.
- Jeder Tag (Markierung) muss für jede Ressource eindeutig sein. Jeder Tag (Markierung) kann nur einen Wert haben.
- Die maximale Länge von Schlüsseln beträgt 128 Unicode-Zeichen in UTF-8.
- Die maximale Länge eines Werts beträgt 256 Unicode-Zeichen in UTF-8.
- Zulässige Zeichen sind Buchstaben, Zahlen, in UTF-8 darstellbare Leerzeichen und die folgenden Zeichen: _.:/= + - @.
- Ein Tag-Schlüssel kann keine leere Zeichenfolge sein. Ein Tag-Wert kann eine leere Zeichenfolge, aber nicht null sein.
- Bei Tag-Schlüsseln und -Werten muss die Groß- und Kleinschreibung beachtet werden.
- Verwenden Sie für Schlüssel `AWS:` oder Werte keine Kombination von Groß- oder Kleinbuchstaben, z. B. als Präfix. Sie sind für die AWS Verwendung reserviert.

Arbeiten mit Tags mithilfe der AWS CLI und der Amazon EMR Serverless API

Verwenden Sie die folgenden AWS CLI Befehle oder Amazon EMR Serverless API-Operationen, um die Tags für Ihre Ressourcen hinzuzufügen, zu aktualisieren, aufzulisten und zu löschen.

CLI-Befehle und API-Operationen für Tags

Ressource	Unterstützt Tags (Markierungen)	Unterstützt Tag-Propagierung
Hinzufügen oder Überschreiben eines oder mehrerer Tags (Markierung)	`tag-resource`	`TagResource`
Tags für eine Ressource auflisten	`list-tags-for-resource`	`ListTagsForResource`
Löschen eines oder mehrerer Tags (Markierung)	`untag-resource`	`UntagResource`

Die folgenden Beispiele zeigen, wie Sie Ressourcen mithilfe von kennzeichnen oder deren Markierung aufheben. AWS CLI

Kennzeichnen Sie eine bestehende Anwendung

Der folgende Befehl kennzeichnet eine bestehende Anwendung.

```
aws emr-serverless tag-resource --resource-arn resource_ARN --tags team=devs
```

Hebt die Markierung einer vorhandenen Anwendung auf

Mit dem folgenden Befehl wird ein Tag aus einer vorhandenen Anwendung gelöscht.

```
aws emr-serverless untag-resource --resource-arn resource_ARN --tag-keys tag_key
```

Listet die Tags für eine Ressource auf

Der folgende Befehl listet die Tags auf, die einer vorhandenen Ressource zugeordnet sind.

```
aws emr-serverless list-tags-for-resource --resource-arn resource_ARN
```

Tutorials für EMR Serverless

In diesem Abschnitt werden allgemeine Anwendungsfälle beschrieben, wenn Sie mit serverlosen EMR-Anwendungen arbeiten. Dazu gehören eine Vielzahl von Tools, darunter Hudi und Iceberg, für die Arbeit an großen Datensätzen und die Verwendung von Python- und Python-Bibliotheken zur Einreichung von Spark-Jobs.

Themen

Verwenden von Java 17 mit Amazon EMR Serverless

Mit Amazon EMR-Versionen 6.11.0 und höher konfigurieren Sie EMR Serverless Spark-Jobs so, dass sie die Java 17-Laufzeit für die Java Virtual Machine (JVM) verwenden. Verwenden Sie eine der folgenden Methoden, um Spark mit Java 17 zu konfigurieren.

`JAVA_HOME`

Um die JVM-Einstellung für EMR Serverless 6.11.0 und höher zu überschreiben, geben Sie die `JAVA_HOME` Einstellung an ihre und die Umgebungsklassifizierungen an. `spark.emr-serverless.driverEnv` `spark.executorEnv`

x86_64

Legen Sie die erforderlichen Eigenschaften fest, um Java 17 als `JAVA_HOME` Konfiguration für den Spark-Treiber und die Executoren anzugeben:

```
--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

Stellen Sie die erforderlichen Eigenschaften ein, um Java 17 als `JAVA_HOME` Konfiguration für den Spark-Treiber und die Executoren anzugeben:

```
--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`

Alternativ können Sie Java 17 in der `spark-defaults` Klassifizierung angeben, um die JVM-Einstellung für EMR Serverless 6.11.0 und höher zu überschreiben.

x86_64

Geben Sie Java 17 in der Klassifizierung an: `spark-defaults`

```
{
"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

Geben Sie Java 17 in der `spark-defaults` Klassifizierung an:

```
{
"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/"
            }
         }
      ]
}
```

Verwenden von Apache Hudi mit EMR Serverless

In diesem Abschnitt wird die Verwendung von Apache Hudi mit EMR Serverless-Anwendungen beschrieben. Hudi ist ein Datenmanagement-Framework, das die Datenverarbeitung vereinfacht.

Um Apache Hudi mit EMR Serverless-Anwendungen zu verwenden

1. Stellen Sie die erforderlichen Spark-Eigenschaften in der entsprechenden Spark-Jobausführung ein.

   ```
   spark.jars=/usr/lib/hudi/hudi-spark-bundle.jar,/usr/lib/hudi/hudi-utilities-
   bundle.jar,/usr/lib/hudi/hudi-aws-bundle.jar
   spark.serializer=org.apache.spark.serializer.KryoSerializer
   ```

2. Um eine Hudi-Tabelle mit dem konfigurierten Katalog zu synchronisieren, bestimmen Sie entweder den AWS Glue-Datenkatalog als Ihren Metastore oder konfigurieren Sie einen externen Metastore. EMR Serverless unterstützt `hms` als Synchronisierungsmodus für Hive-Tabellen für Hudi-Workloads. EMR Serverless aktiviert diese Eigenschaft standardmäßig. Weitere Informationen zum Einrichten Ihres Metastores finden Sie unter. Metastore-Konfiguration für EMR Serverless

> **Important**
>
> EMR Serverless unterstützt `HIVEQL` oder `JDBC` als Synchronisierungsmodus-Optionen für Hive-Tabellen zur Verarbeitung von Hudi-Workloads nicht. Weitere Informationen finden Sie unter Synchronisierungsmodi.

Wenn Sie den AWS Glue-Datenkatalog als Metastore verwenden, geben Sie die folgenden Konfigurationseigenschaften für Ihren Hudi-Job an.

```
--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
```

Weitere Informationen zu den Apache Hudi-Versionen von Amazon EMR finden Sie in der Hudi-Versionshistorie.

Verwenden von Apache Iceberg mit EMR Serverless

In diesem Abschnitt wird beschrieben, wie Apache Iceberg mit serverlosen EMR-Anwendungen verwendet wird. Apache Iceberg ist ein Tabellenformat, das die Arbeit mit großen Datensätzen in Data Lakes erleichtert.

So verwenden Sie Apache Iceberg mit serverlosen EMR-Anwendungen

1. Stellen Sie die erforderlichen Spark-Eigenschaften in der entsprechenden Spark-Jobausführung ein.

   ```
   spark.jars=/usr/share/aws/iceberg/lib/iceberg-spark3-runtime.jar
   ```

2. Benennen Sie entweder den AWS Glue-Datenkatalog als Ihren Metastore oder konfigurieren Sie einen externen Metastore. Weitere Informationen zur Einrichtung Ihres Metastores finden Sie unter. Metastore-Konfiguration für EMR Serverless

Konfigurieren Sie die Metastore-Eigenschaften, die Sie für Iceberg verwenden möchten. Wenn Sie beispielsweise den AWS Glue-Datenkatalog verwenden möchten, legen Sie die folgenden Eigenschaften in der Anwendungskonfiguration fest.

```
spark.sql.catalog.dev.warehouse=s3://amzn-s3-demo-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
```

Wenn Sie den AWS Glue-Datenkatalog als Metastore verwenden, geben Sie die folgenden Konfigurationseigenschaften für Ihren Iceberg-Job an.

```
--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://amzn-s3-demo-bucket/EXAMPLE-PREFIX/
--conf
 spark.hadoop.hive.metastore.client.factory.class=com.amazonaws.glue.catalog.metastore.AWSG
```

Weitere Informationen zu den Apache Iceberg-Versionen von Amazon EMR finden Sie in der Iceberg-Versionshistorie.

Verwenden von Python-Bibliotheken mit EMR Serverless

Wenn Sie PySpark Jobs auf Amazon EMR Serverless-Anwendungen ausführen, packen Sie verschiedene Python-Bibliotheken als Abhängigkeiten. Verwenden Sie dazu native Python-Funktionen, erstellen Sie eine virtuelle Umgebung oder konfigurieren Sie Ihre PySpark Jobs direkt für die Verwendung von Python-Bibliotheken. Diese Seite behandelt jeden Ansatz.

Verwendung nativer Python-Funktionen

Wenn Sie die folgende Konfiguration festlegen, verwenden Sie sie, PySpark um Python-Dateien (`.py`), gezippte Python-Pakete (`.zip`) und Egg-Dateien (`.egg`) auf Spark-Executoren hochzuladen.

```
--conf spark.submit.pyFiles=s3://amzn-s3-demo-bucket/EXAMPLE-PREFIX/<.py|.egg|.zip
 file>
```

Weitere Informationen zur Verwendung virtueller Python-Umgebungen für PySpark Jobs finden Sie unter Verwenden PySpark nativer Funktionen.

Wenn Sie EMR Notebook verwenden, können Sie die Python-Abhängigkeit in Ihrem Notebook verfügbar machen, indem Sie den folgenden Code ausführen:

```
   %%configure -f
{
   "conf": {
   "spark.submit.pyFiles":"s3:///amzn-s3-demo-bucket/EXAMPLE-PREFIX/<.py|.egg|.zip
file>
                  }
}
```

Aufbau einer virtuellen Python-Umgebung

Um mehrere Python-Bibliotheken für einen PySpark Job zu paketieren, erstellen Sie isolierte virtuelle Python-Umgebungen.

1. Verwenden Sie die folgenden Befehle, um die virtuelle Python-Umgebung zu erstellen. Das gezeigte Beispiel installiert die Pakete `scipy` und `matplotlib` in ein virtuelles Umgebungspaket und kopiert das Archiv an einen Amazon S3 S3-Speicherort.

   > **Important**
   >
   > Sie müssen die folgenden Befehle in einer ähnlichen Amazon Linux 2-Umgebung mit derselben Version von Python ausführen, die Sie in EMR Serverless verwenden, d. h. Python 3.7.10 für Amazon EMR Version 6.6.0. Ein Beispiel für ein Dockerfile finden Sie im EMR Serverless Samples Repository. 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://amzn-s3-demo-bucket/EXAMPLE-PREFIX/

# optionally, remove the virtual environment directory
rm -fr pyspark_venvsource
```

2. Reichen Sie den Spark-Job mit den für die Verwendung der virtuellen Python-Umgebung festgelegten Eigenschaften ein.

```
--conf spark.archives=s3://amzn-s3-demo-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
```

Beachten Sie, dass, wenn Sie die ursprüngliche Python-Binärdatei nicht überschreiben, die zweite Konfiguration in der vorherigen Reihenfolge der Einstellungen verwendet wird`--conf spark.executorEnv.PYSPARK_PYTHON=python`.

Weitere Informationen zur Verwendung virtueller Python-Umgebungen für PySpark Jobs finden Sie unter Virtualenv verwenden. Weitere Beispiele für das Einreichen von Spark-Jobs finden Sie unter. Verwenden von Spark-Konfigurationen bei der Ausführung von EMR Serverless-Jobs

PySpark Jobs für die Verwendung von Python-Bibliotheken konfigurieren

Mit den Amazon EMR-Versionen 6.12.0 und höher können Sie serverlose PySpark EMR-Jobs direkt so konfigurieren, dass sie beliebte Data-Science-Python-Bibliotheken wie Pandas verwenden NumPy, und das ohne zusätzliche Einrichtung. PyArrow

Die folgenden Beispiele zeigen, wie jede Python-Bibliothek für einen PySpark Job gepackt wird.

NumPy

NumPy ist eine Python-Bibliothek für wissenschaftliches Rechnen, die multidimensionale Arrays und Operationen für Mathematik, Sortierung, Zufallssimulation und grundlegende Statistik bietet. Führen Sie zur Verwendung NumPy den folgenden Befehl aus:

```
import numpy
```

pandas

pandas ist eine Python-Bibliothek, auf NumPy der aufgebaut ist. Die Pandas-Bibliothek bietet Datenwissenschaftlern Datenstrukturen und DataFrameDatenanalysetools. Führen Sie den folgenden Befehl aus, um Pandas zu verwenden:

```
import pandas
```

PyArrow

PyArrow ist eine Python-Bibliothek, die spaltenförmige Daten im Speicher verwaltet, um die Arbeitsleistung zu verbessern. PyArrow basiert auf der sprachübergreifenden Entwicklungsspezifikation von Apache Arrow, einer Standardmethode zur Darstellung und zum Austausch von Daten in einem Spaltenformat. Führen Sie zur Verwendung PyArrow den folgenden Befehl aus:

```
import pyarrow
```

Verwendung verschiedener Python-Versionen mit EMR Serverless

Neben dem Anwendungsfall in können Sie auch virtuelle Python-Umgebungen verwendenVerwenden von Python-Bibliotheken mit EMR Serverless, um mit anderen Python-Versionen als der Version zu arbeiten, die in der Amazon EMR-Version für Ihre Amazon EMR Serverless-Anwendung enthalten ist. Erstellen Sie dazu eine virtuelle Python-Umgebung mit der Python-Version, die Sie verwenden möchten.

So reichen Sie einen Job aus einer virtuellen Python-Umgebung ein

1. Erstellen Sie Ihre virtuelle Umgebung mit den Befehlen im folgenden Beispiel. In diesem Beispiel wird Python 3.9.9 in ein virtuelles Umgebungspaket installiert und das Archiv an einen Amazon S3 S3-Speicherort kopiert.

> **Important**
>
> Wenn Sie Amazon EMR-Versionen 7.0.0 und höher verwenden, führen Sie Ihre Befehle in einer Amazon Linux 2023-Umgebung aus, die der Umgebung ähnelt, die Sie für Ihre EMR Serverless-Anwendungen verwenden.
>
> Wenn Sie Version 6.15.0 oder niedriger verwenden, führen Sie die folgenden Befehle in einer ähnlichen Amazon Linux 2-Umgebung aus.

```
# 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. Stellen Sie Ihre Eigenschaften so ein, dass sie die virtuelle Python-Umgebung verwenden, und reichen Sie den Spark-Job ein.

```
# note that the archive suffix "environment" is the same as the directory where you
 copied the Python binary.
--conf spark.archives=s3://amzn-s3-demo-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
```

Weitere Informationen zur Verwendung virtueller Python-Umgebungen für PySpark Jobs finden Sie unter Virtualenv verwenden. Weitere Beispiele für das Einreichen von Spark-Jobs finden Sie unter. Verwenden von Spark-Konfigurationen bei der Ausführung von EMR Serverless-Jobs

Verwenden von Delta Lake OSS mit EMR Serverless

Amazon EMR-Versionen 6.9.0 und höher

> **Note**
>
> Amazon EMR 7.0.0 und höher verwendet Delta Lake 3.0.0, wodurch die Datei umbenannt wird in. `delta-core.jar` `delta-spark.jar` Wenn Sie Amazon EMR 7.0.0 oder höher verwenden, stellen Sie sicher, dass Sie dies `delta-spark.jar` in Ihren Konfigurationen angeben.

Amazon EMR 6.9.0 und höher beinhaltet Delta Lake, sodass Sie Delta Lake nicht mehr selbst verpacken oder die `--packages` Flagge mit Ihren EMR Serverless-Jobs bereitstellen müssen.

1. Wenn Sie EMR Serverless-Jobs einreichen, stellen Sie sicher, dass Sie über die folgenden Konfigurationseigenschaften verfügen, und geben Sie die folgenden Parameter in das `sparkSubmitParameters` Feld ein.

   ```
   --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. Erstellen Sie eine lokale Datei`delta_sample.py`, um das Erstellen und Lesen einer Delta-Tabelle zu testen.

```
# delta_sample.py
    from pyspark.sql import SparkSession

    import uuid

    url = "s3://amzn-s3-demo-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. Laden Sie die AWS CLI`delta_sample.py` Datei mithilfe von in Ihren Amazon S3 S3-Bucket hoch. Verwenden Sie dann den `start-job-run` Befehl, um einen Job an eine bestehende EMR Serverless-Anwendung zu senden.

```
aws s3 cp delta_sample.py s3://amzn-s3-demo-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://amzn-s3-demo-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"
            }
        }'
```

Um Python-Bibliotheken mit Delta Lake zu verwenden, fügen Sie die `delta-core` Bibliothek hinzu, indem Sie sie als Abhängigkeit packen oder als benutzerdefiniertes Image verwenden.

Alternativ können Sie die verwenden, `SparkContext.addPyFile` um die Python-Bibliotheken aus der `delta-core` JAR-Datei hinzuzufügen:

```
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-Versionen 6.8.0 und niedriger

Wenn Sie Amazon EMR 6.8.0 oder niedriger verwenden, gehen Sie wie folgt vor, um Delta Lake OSS mit Ihren EMR Serverless-Anwendungen zu verwenden.

1. Um eine Open-Source-Version von Delta Lake zu erstellen, die mit der Version von Spark auf Ihrer Amazon EMR Serverless-Anwendung kompatibel ist, navigieren Sie zu Delta GitHub und folgen Sie den Anweisungen.
2. Laden Sie die Delta Lake-Bibliotheken in einen Amazon S3 S3-Bucket in Ihrem hoch AWS-Konto.
3. Wenn Sie EMR Serverless-Jobs in der Anwendungskonfiguration einreichen, schließen Sie die Delta Lake JAR-Dateien ein, die sich jetzt in Ihrem Bucket befinden.

   ```
   --conf spark.jars=s3://amzn-s3-demo-bucket/jars/delta-core_2.12-1.1.0.jar
   ```

4. Führen Sie einen PySpark Beispieltest durch, um sicherzustellen, dass Sie in eine Delta-Tabelle lesen und aus ihr schreiben können.

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

       import uuid

       conf = SparkConf()
       sc = SparkContext(conf=conf)
       sqlContext = HiveContext(sc)

       url = "s3://amzn-s3-demo-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
```

Serverless-EMR-Jobs von Airflow einreichen

Der Amazon-Anbieter in Apache Airflow bietet EMR-Serverless-Operatoren. Weitere Informationen zu Operatoren finden Sie unter Amazon EMR Serverless Operators in der Apache Airflow Airflow-Dokumentation.

Sie können es verwenden`EmrServerlessCreateApplicationOperator`, um eine Spark- oder Hive-Anwendung zu erstellen. Sie können es auch verwenden`EmrServerlessStartJobOperator`, um einen oder mehrere Jobs mit Ihrer neuen Anwendung zu starten.

Um den Operator mit Amazon Managed Workflows for Apache Airflow (MWAA) mit Airflow 2.2.2 zu verwenden, fügen Sie Ihrer Datei die folgende Zeile hinzu und aktualisieren Sie Ihre MWAA-Umgebung, sodass sie die neue `requirements.txt` Datei verwendet.

```
apache-airflow-providers-amazon==6.0.0
boto3>=1.23.9
```

Beachten Sie, dass EMR Serverless Support zu Version 5.0.0 des Amazon-Anbieters hinzugefügt wurde. Version 6.0.0 ist die letzte Version, die mit Airflow 2.2.2 kompatibel ist. Sie können spätere Versionen mit Airflow 2.4.3 auf MWAA verwenden.

Das folgende abgekürzte Beispiel zeigt, wie Sie eine Anwendung erstellen, mehrere Spark-Jobs ausführen und die Anwendung dann beenden. Ein vollständiges Beispiel ist im EMR Serverless Samples GitHub Repository verfügbar. Weitere Einzelheiten zur `sparkSubmit` Konfiguration finden Sie unter. Verwenden von Spark-Konfigurationen bei der Ausführung von EMR Serverless-Jobs

```
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 = "amzn-s3-demo-bucket"

DEFAULT_MONITORING_CONFIG = {
    "monitoringConfiguration": {
        "s3MonitoringConfiguration": {"logUri": f"s3://amzn-s3-demo-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)
```

Verwenden benutzerdefinierter Hive-Funktionen mit EMR Serverless

Mit den benutzerdefinierten Hive-Funktionen (UDFs) können Sie benutzerdefinierte Funktionen zur Verarbeitung von Datensätzen oder Datensatzgruppen erstellen. In diesem Tutorial verwenden Sie eine Beispiel-UDF mit einer bereits vorhandenen Amazon EMR Serverless-Anwendung, um einen Job auszuführen, der ein Abfrageergebnis ausgibt. Informationen zum Einrichten einer Anwendung finden Sie unter. Erste Schritte mit Amazon EMR Serverless

So verwenden Sie eine UDF mit EMR Serverless

1. Navigieren Sie zum Verzeichnis für ein Beispiel GitHubfür eine UDF. Klonen Sie das Repo und wechseln Sie zu dem Git-Branch, den Sie verwenden möchten. Aktualisieren Sie das `maven-compiler-plugin` in der `pom.xml` Datei des Repositorys, um eine Quelle zu haben. Aktualisieren Sie auch die Konfiguration der Ziel-Java-Version auf`1.8`. Führen Sie den `mvn package -DskipTests` Befehl aus, um die JAR-Datei zu erstellen, die Ihr Beispiel enthält UDFs.
2. Nachdem Sie die JAR-Datei erstellt haben, laden Sie sie mit dem folgenden Befehl in Ihren S3-Bucket hoch.

```
aws s3 cp brickhouse-0.8.2-JS.jar s3://amzn-s3-demo-bucket/jars/
```

3. Erstellen Sie eine Beispieldatei, um eine der UDF-Beispielfunktionen zu verwenden. Speichern Sie diese Abfrage unter `udf_example.q` und laden Sie sie in Ihren S3-Bucket hoch.

```
add jar s3://amzn-s3-demo-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. Reichen Sie den folgenden Hive-Job ein.

```
aws emr-serverless start-job-run \
  --application-id application-id \
  --execution-role-arn job-role-arn \
  --job-driver '{
    "hive": {
        "query": "s3://amzn-s3-demo-bucket/queries/udf_example.q",
        "parameters": "--hiveconf hive.exec.scratchdir=s3://amzn-s3-demo-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://amzn-s3-demo-bucket/logs/"
        }
    }
}'
```

5. Verwenden Sie den `get-job-run` Befehl, um den Status Ihres Jobs zu überprüfen. Warten Sie, bis der Status geändert wird`SUCCESS`.

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

6. Laden Sie die Ausgabedateien mit dem folgenden Befehl herunter.

```
aws s3 cp --recursive s3://amzn-s3-demo-bucket/logs/applications/application-id/
jobs/job-id/HIVE_DRIVER/ .
```

Die `stdout.gz` Datei ähnelt der folgenden.

```
{"key1":[0,1,2],"key2":[3,4,5,6],"key3":[7,8,9]}
2
```

Verwenden von benutzerdefinierten Images mit EMR Serverless

Themen

Verwenden Sie eine benutzerdefinierte Python-Version

Sie können ein benutzerdefiniertes Image erstellen, um eine andere Version von Python zu verwenden. Um Python Version 3.10 beispielsweise für Spark-Jobs zu verwenden, führen Sie den folgenden Befehl aus:

```
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 runs the image as hadoop
USER hadoop:hadoop
```

Bevor Sie den Spark-Job einreichen, legen Sie Ihre Eigenschaften wie folgt für die Verwendung der virtuellen Python-Umgebung fest.

```
--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
```

Verwenden Sie eine benutzerdefinierte Java-Version

Das folgende Beispiel zeigt, wie Sie ein benutzerdefiniertes Image erstellen, um Java 11 für Ihre Spark-Jobs zu verwenden.

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

USER root

# install JDK 11
RUN amazon-linux-extras install java-openjdk11

# EMRS runs the image as hadoop
USER hadoop:hadoop
```

Bevor Sie den Spark-Job einreichen, stellen Sie die Spark-Eigenschaften wie folgt auf die Verwendung von Java 11 ein.

```
--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-
```

Erstellen Sie ein Data-Science-Image

Das folgende Beispiel zeigt, wie gängige Data-Science-Python-Pakete wie Pandas und NumPy eingebunden werden können.

```
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 runs the image as hadoop
USER hadoop:hadoop
```

Verarbeitung von Geodaten mit Apache Sedona

Das folgende Beispiel zeigt, wie ein Bild erstellt wird, das Apache Sedona für die Geodatenverarbeitung einbezieht.

```
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 runs the image as hadoop
USER hadoop:hadoop
```

Lizenzinformationen für die Verwendung benutzerdefinierter Bilder

Sie können mit EMR Serverless benutzerdefinierte Images erstellen, um bestimmte Aufgaben auszuführen oder bestimmte Versionen eines Softwarepakets zu verwenden. Die Änderung und Verteilung von benutzerdefinierten Images kann bestimmten Regeln und Lizenzbedingungen unterliegen. Der Lizenztext wird im folgenden Unterabschnitt angezeigt.

Lizenzierung, die für benutzerdefinierte Bilder gilt

Copyright Amazon.com und seine verbundenen Unternehmen; alle Rechte vorbehalten. Diese Software ist AWS Inhalt gemäß AWS Kundenvereinbarung und darf nicht ohne Genehmigung verbreitet werden. Zusätzlich zu den in der Lizenz für AWS geistiges Eigentum enthaltenen Genehmigungen gewährt Ihnen der AWS Lizenzgeber die folgenden zusätzlichen Genehmigungen:

Das Erstellen, Kopieren und Verwenden von Derivaten des AWS Inhalts ist zulässig, sofern die folgenden Bedingungen erfüllt sind:

- Sie ändern den AWS Inhalt selbst nicht, und alle Derivate sind ausschließlich das Ergebnis Ihrer Hinzufügung neuer Inhalte.
- Interne Reproduktionen müssen den oben genannten Copyright-Hinweis enthalten.
- Die externe Verbreitung, in Quell- oder Binärform, mit oder ohne Änderung, ist gemäß den Bedingungen dieser Lizenz nicht gestattet.

Weitere Informationen zur Verwendung benutzerdefinierter Images finden Sie unter Verwenden von benutzerdefinierten Images mit EMR Serverless.

Verwenden der Amazon Redshift Redshift-Integration für Apache Spark auf Amazon EMR Serverless

Mit Amazon-EMR-Version 6.9.0 und höher enthält jedes Versions-Image einen Konnektor zwischen Apache Spark und Amazon Redshift. Verwenden Sie mit diesem Connector Spark auf Amazon EMR Serverless, um in Amazon Redshift gespeicherte Daten zu verarbeiten. Die Integration basiert auf dem `spark-redshift`-Open-Source-Konnektor. Für Amazon EMR Serverless ist die Amazon Redshift Redshift-Integration für Apache Spark als native Integration enthalten.

Themen

- Starten einer Spark-Anwendung mit der Amazon Redshift Redshift-Integration für Apache Spark
- Authentifizierung mit der Amazon-Redshift-Integration für Apache Spark
- Lesen und Schreiben von und zu Amazon Redshift
- Überlegungen und Einschränkungen bei der Verwendung des Spark-Connectors

Starten einer Spark-Anwendung mit der Amazon Redshift Redshift-Integration für Apache Spark

Um die Integration mit EMR Serverless 6.9.0 zu verwenden, übergeben Sie die erforderlichen Spark-Redshift-Abhängigkeiten mit Ihrem Spark-Job. Wird verwendet`--jars`, um Redshift-Connector-bezogene Bibliotheken einzubeziehen. Informationen zum Zugriff auf andere von der `--jars` Option unterstützte Dateispeicherorte finden Sie im Abschnitt Advanced Dependency Management der Apache Spark-Dokumentation.

- `spark-redshift.jar`

- `spark-avro.jar`
- `RedshiftJDBC.jar`
- `minimal-json.jar`

Amazon-EMR-Versionen 6.10.0 und höher erfordern die `minimal-json.jar`-Abhängigkeit nicht und installieren die anderen Abhängigkeiten standardmäßig automatisch in jedem Cluster. Die folgenden Beispiele zeigen, wie Sie eine Spark-Anwendung mit der Amazon Redshift Redshift-Integration für Apache Spark starten.

Amazon EMR 6.10.0 +

Starten Sie einen Spark-Job auf Amazon EMR Serverless mit der Amazon Redshift Redshift-Integration für Apache Spark auf EMR Serverless Version 6.10.0 und höher.

```
spark-submit my_script.py
```

Amazon EMR 6.9.0

Um einen Spark-Job auf Amazon EMR Serverless mit der Amazon Redshift Redshift-Integration für Apache Spark auf EMR Serverless Version 6.9.0 zu starten, verwenden Sie die `--jars` Option, wie im folgenden Beispiel gezeigt. Beachten Sie, dass die mit der `--jars`-Option aufgeführten Pfade die Standardpfade für die JAR-Dateien sind.

```
--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
```

Authentifizierung mit der Amazon-Redshift-Integration für Apache Spark

Wird verwendet AWS Secrets Manager , um Anmeldeinformationen abzurufen und eine Verbindung zu Amazon Redshift herzustellen

Sie können sich sicher bei Amazon Redshift authentifizieren, indem Sie die Anmeldeinformationen in Secrets Manager speichern und den Spark-Job die `GetSecretValue` API aufrufen lassen, um sie abzurufen:

```
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
```

Authentifizierung bei Amazon Redshift mit einem JDBC-Treiber

Geben Sie den Benutzernamen und das Passwort in der JDBC-URL ein

Sie können einen Spark-Job bei einem Amazon Redshift-Cluster authentifizieren, indem Sie den Namen und das Passwort der Amazon Redshift Redshift-Datenbank in der JDBC-URL angeben.

> **Note**
>
> Wenn Sie die Datenbankanmeldedaten in der URL übergeben, kann jeder, der Zugriff auf die URL hat, auch auf die Anmeldeinformationen zugreifen. Diese Methode wird im Allgemeinen nicht empfohlen, da sie keine sichere Option ist.

Wenn die Sicherheit Ihrer Anwendung kein Problem darstellt, verwenden Sie das folgende Format, um den Benutzernamen und das Passwort in der JDBC-URL festzulegen:

```
jdbc:redshift://redshifthost:5439/database?user=username&password=password
```

Verwenden Sie die IAM-basierte Authentifizierung mit der Rolle Amazon EMR Serverless Job Execution

Ab Amazon EMR Serverless Version 6.9.0 ist der Amazon Redshift JDBC-Treiber 2.1 oder höher in der Umgebung enthalten. Mit dem JDBC-Treiber 2.1 und höher können Sie die JDBC-URL angeben, ohne den unformatierten Benutzernamen und das Passwort anzugeben.

`jdbc:redshift:iam://`Geben Sie stattdessen das Schema an. Dadurch wird der JDBC-Treiber angewiesen, Ihre EMR Serverless Job Execution Rolle zu verwenden, um die Anmeldeinformationen automatisch abzurufen. Weitere Informationen finden Sie unter Konfiguration einer JDBC- oder ODBC-Verbindung zur Verwendung von IAM-Anmeldeinformationen im Amazon Redshift Management Guide. Ein Beispiel für diese URL ist:

```
jdbc:redshift:iam://examplecluster.abc123xyz789.us-west-2.redshift.amazonaws.com:5439/
dev
```

Die folgenden Berechtigungen sind für Ihre Jobausführungsrolle erforderlich, wenn die angegebenen Bedingungen erfüllt sind:

Berechtigung	Bedingungen, sofern sie für die Rolle Auftragsausführung erforderlich sind
`redshift:GetClusterCredentials`	Erforderlich, damit der JDBC-Treiber die Anmeldeinformationen von Amazon Redshift abrufen kann

Berechtigung	Bedingungen, sofern sie für die Rolle Auftragsausführung erforderlich sind
`redshift:DescribeCluster`	Erforderlich, wenn Sie den Amazon-Redshift-Cluster und AWS-Region in der JDBC-URL anstelle des Endpunkts angeben
`redshift-serverless:GetCredentials`	Erforderlich, damit der JDBC-Treiber die Anmeldeinformationen von Amazon Redshift Serverless abrufen kann
`redshift-serverless:GetWorkgroup`	Erforderlich, wenn Sie Amazon Redshift Serverless verwenden und die URL in Form von Arbeitsgruppenname und Region angeben

Verbindung zu Amazon Redshift innerhalb einer anderen VPC herstellen

Wenn Sie einen bereitgestellten Amazon Redshift-Cluster oder eine Amazon Redshift Serverless-Arbeitsgruppe unter einer VPC einrichten, konfigurieren Sie die VPC-Konnektivität für Ihre Amazon EMR Serverless-Anwendung für den Zugriff auf die Ressourcen. Weitere Informationen zur Konfiguration der VPC-Konnektivität in einer serverlosen EMR-Anwendung finden Sie unter. Konfiguration des VPC-Zugriffs für serverlose EMR-Anwendungen zur Verbindung mit Daten

- Wenn Ihr bereitgestellter Amazon Redshift-Cluster oder Ihre Amazon Redshift Serverless-Arbeitsgruppe öffentlich zugänglich ist, geben Sie ein oder mehrere private Subnetze an, an die ein NAT-Gateway angeschlossen ist, wenn Sie serverlose EMR-Anwendungen erstellen.
- Wenn Ihr bereitgestellter Amazon Redshift-Cluster oder Ihre Amazon Redshift Serverless-Arbeitsgruppe nicht öffentlich zugänglich ist, müssen Sie einen von Amazon Redshift verwalteten VPC-Endpunkt für Ihren Amazon Redshift Redshift-Cluster erstellen, wie unter beschrieben. Konfiguration des VPC-Zugriffs für serverlose EMR-Anwendungen zur Verbindung mit Daten Alternativ können Sie Ihre Amazon Redshift Serverless-Arbeitsgruppe wie unter Verbinden mit Amazon Redshift Serverless im Amazon Redshift Management Guide beschrieben erstellen. Sie müssen Ihren Cluster oder Ihre Untergruppe den privaten Subnetzen zuordnen, die Sie bei der Erstellung Ihrer EMR Serverless-Anwendung angeben.

> **Note**
>
> Wenn Sie die IAM-basierte Authentifizierung verwenden und an Ihre privaten Subnetze für die EMR Serverless-Anwendung kein NAT-Gateway angeschlossen ist, müssen Sie in diesen Subnetzen auch einen VPC-Endpunkt für Amazon Redshift oder Amazon Redshift Serverless erstellen. Auf diese Weise kann der JDBC-Treiber die Anmeldeinformationen abrufen.

Lesen und Schreiben von und zu Amazon Redshift

Die folgenden Codebeispiele dienen PySpark zum Lesen und Schreiben von Beispieldaten aus und in eine Amazon Redshift Redshift-Datenbank mit einer Datenquellen-API und mit SparkSQL.

Data source API

Wird verwendet PySpark , um Beispieldaten von und in eine Amazon Redshift Redshift-Datenbank mit Datenquellen-API zu lesen und zu schreiben.

```
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

Wird verwendet PySpark , um Beispieldaten mit SparkSQL aus und in eine Amazon Redshift Redshift-Datenbank zu lesen und zu schreiben.

```
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()
```

Überlegungen und Einschränkungen bei der Verwendung des Spark-Connectors

- Wir empfehlen, dass Sie SSL für die JDBC-Verbindung von Spark auf Amazon EMR zu Amazon Redshift aktivieren.
- AWS Secrets Manager Als bewährte Methode empfehlen wir Ihnen, die Anmeldeinformationen für den Amazon Redshift Redshift-Cluster zu verwalten. Ein Beispiel finden Sie unter Verwenden AWS Secrets Manager zum Abrufen von Anmeldeinformationen für die Verbindung mit Amazon Redshift.
- Wir empfehlen, dass Sie eine IAM-Rolle mit dem Parameter `aws_iam_role` für den Amazon Redshift Redshift-Authentifizierungsparameter übergeben.
- Derzeit wird das Parquet-Format vom Parameter `tempformat` nicht unterstützt.
- Die `tempdir`-URI verweist auf einen Amazon-S3-Speicherort. Dieses temporäre Verzeichnis wird nicht automatisch bereinigt und kann zusätzliche Kosten verursachen.
- Beachten Sie die folgenden Empfehlungen für Amazon Redshift:
  - Wir empfehlen Ihnen, den öffentlichen Zugriff auf den Amazon Redshift Redshift-Cluster zu blockieren.
  - Wir empfehlen Ihnen, die Amazon Redshift Redshift-Auditprotokollierung zu aktivieren.
  - Wir empfehlen Ihnen, die Amazon Redshift Redshift-Verschlüsselung im Ruhezustand zu aktivieren.
- Beachten Sie die folgenden Empfehlungen für Amazon S3:
  - Wir empfehlen Ihnen, den öffentlichen Zugriff auf Amazon S3 S3-Buckets zu blockieren.
  - Wir empfehlen Ihnen, die serverseitige Amazon S3 S3-Verschlüsselung zu verwenden, um die verwendeten Amazon S3 S3-Buckets zu verschlüsseln.
  - Wir empfehlen Ihnen, Amazon S3 S3-Lebenszyklusrichtlinien zu verwenden, um die Aufbewahrungsregeln für den Amazon S3 S3-Bucket zu definieren.
  - Amazon EMR überprüft immer Code, der aus Open Source in das Image importiert wurde. Aus Sicherheitsgründen unterstützen wir die folgenden Authentifizierungsmethoden von Spark für Amazon S3 nicht:
    - Festlegung von AWS Zugriffsschlüsseln in der `hadoop-env` Konfigurationsklassifizierung
    - Kodierung der AWS Zugriffsschlüssel in der `tempdir` URI

Weitere Informationen zum Verwenden des Konnektors und seiner unterstützten Parameter finden Sie in den folgenden Ressourcen:

- Amazon-Redshift-Integration für Apache Spark im Amazon-Redshift-Verwaltungshandbuch
- Das `spark-redshift-Community-Repository` auf Github

Mit Amazon EMR Serverless eine Verbindung zu DynamoDB herstellen

In diesem Tutorial laden Sie eine Teilmenge der Daten vom Vereinigte Staaten Board on Geographic Names in einen Amazon S3 S3-Bucket hoch und kopieren die Daten dann mit Hive oder Spark auf Amazon EMR Serverless zur Abfrage in eine Amazon DynamoDB-Tabelle.

Schritt 1: Daten in einen Amazon S3 S3-Bucket hochladen

Um einen Amazon S3 S3-Bucket zu erstellen, folgen Sie den Anweisungen unter Bucket erstellen im Amazon Simple Storage Service Console-Benutzerhandbuch. Ersetzen Sie Verweise auf *amzn-s3-demo-bucket* durch den Namen Ihres neu erstellten Buckets. Jetzt ist Ihre EMR Serverless-Anwendung bereit, Jobs auszuführen.

1. Laden Sie das Beispieldatenarchiv `features.zip` mit dem folgenden Befehl herunter.

   ```
   wget https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/samples/
   features.zip
   ```

2. Extrahieren Sie die `features.txt` Datei aus dem Archiv und greifen Sie auf die ersten Zeilen der Datei zu:

   ```
   unzip features.zip
   head features.txt
   ```

   Das Ergebnis sollte dem Folgenden ähneln.

   ```
   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
```

Die Felder in jeder Zeile geben hier eine eindeutige Kennung, einen Namen, die Art des natürlichen Merkmals, den Bundesstaat, den Breitengrad in Grad, den Längengrad in Grad und die Höhe in Fuß an.

3. Laden Sie Ihre Daten auf Amazon S3 hoch

```
aws s3 cp features.txt s3://amzn-s3-demo-bucket/features/
```

Schritt 2: Erstellen Sie eine Hive-Tabelle

Verwenden Sie Apache Spark oder Hive, um eine neue Hive-Tabelle zu erstellen, die die hochgeladenen Daten in Amazon S3 enthält.

Spark

Um eine Hive-Tabelle mit Spark zu erstellen, führen Sie den folgenden Befehl aus.

```
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://amzn-s3-demo-bucket/features';")
```

Sie haben jetzt eine aufgefüllte Hive-Tabelle mit Daten aus der `features.txt` Datei. Um zu überprüfen, ob sich Ihre Daten in der Tabelle befinden, führen Sie eine Spark-SQL-Abfrage aus, wie im folgenden Beispiel gezeigt.

```
sparkSession.sql(
```

```
    "SELECT state_alpha, COUNT(*) FROM hive_features GROUP BY state_alpha;")
```

Hive

Führen Sie den folgenden Befehl aus, um eine Hive-Tabelle mit Hive zu erstellen.

```
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://amzn-s3-demo-bucket/features';
```

Sie haben jetzt eine Hive-Tabelle, die Daten aus der Datei enthält. `features.txt` Um zu überprüfen, ob sich Ihre Daten in der Tabelle befinden, führen Sie eine HiveQL-Abfrage aus, wie im folgenden Beispiel gezeigt.

```
SELECT state_alpha, COUNT(*) FROM hive_features GROUP BY state_alpha;
```

Schritt 3: Daten nach DynamoDB kopieren

Verwenden Sie Spark oder Hive, um Daten in eine neue DynamoDB-Tabelle zu kopieren.

Spark

Um Daten aus der Hive-Tabelle, die Sie im vorherigen Schritt erstellt haben, nach DynamoDB zu kopieren, folgen Sie den Schritten 1 bis 3 unter Daten nach DynamoDB kopieren. Dadurch wird eine neue DynamoDB-Tabelle mit dem Namen erstellt. `Features` Anschließend können Sie Daten direkt aus der Textdatei lesen und in Ihre DynamoDB-Tabelle kopieren, wie das folgende Beispiel zeigt.

```
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://amzn-s3-demo-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

Um Daten aus der Hive-Tabelle, die Sie im vorherigen Schritt erstellt haben, nach DynamoDB zu kopieren, folgen Sie den Anweisungen unter Daten nach DynamoDB kopieren.

Schritt 4: Daten von DynamoDB abfragen

Verwenden Sie Spark oder Hive, um Ihre DynamoDB-Tabelle abzufragen.

Spark

Verwenden Sie entweder Spark SQL oder die Spark-API, um Daten aus der DynamoDB-Tabelle abzufragen, die Sie im vorherigen Schritt erstellt haben. MapReduce

Example— Fragen Sie Ihre DynamoDB-Tabelle mit Spark SQL ab

Die folgende Spark-SQL-Abfrage gibt eine Liste aller Feature-Typen in alphabetischer Reihenfolge zurück.

```
val dataFrame = sparkSession.sql("SELECT DISTINCT feature_class \
    FROM ddb_features \
    ORDER BY feature_class;")
```

Die folgende Spark-SQL-Abfrage gibt eine Liste aller Seen zurück, die mit dem Buchstaben M beginnen.

```
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;")
```

Die folgende Spark-SQL-Abfrage gibt eine Liste aller Bundesstaaten mit mindestens drei Features zurück, die höher als eine Meile sind.

```
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— Fragen Sie Ihre DynamoDB-Tabelle mit der Spark-API ab MapReduce

Die folgende MapReduce Abfrage gibt eine Liste aller Feature-Typen in alphabetischer Reihenfolge zurück.

```
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")
```

Die folgende MapReduce Abfrage gibt eine Liste aller Seen zurück, die mit dem Buchstaben M beginnen.

```
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")
```

Die folgende MapReduce Abfrage gibt eine Liste aller Bundesstaaten mit mindestens drei Features zurück, die höher als eine Meile sind.

```
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

Um Daten aus der DynamoDB-Tabelle abzufragen, die Sie im vorherigen Schritt erstellt haben, folgen Sie den Anweisungen unter Daten in der DynamoDB-Tabelle abfragen.

Einrichten des kontoübergreifenden Zugriffs

Gehen Sie wie folgt vor, um den kontenübergreifenden Zugriff für EMR Serverless einzurichten. Im Beispiel `AccountA` ist das Konto, in dem Sie Ihre Amazon EMR Serverless-Anwendung erstellt haben, und das Konto, in dem sich Ihre Amazon DynamoDB `AccountB` befindet.

1. Erstellen Sie eine DynamoDB-Tabelle in. `AccountB` Weitere Informationen finden Sie unter Schritt 1: Tabelle erstellen.
2. Erstellen Sie eine `Cross-Account-Role-B` IAM-Rolle in`AccountB`, die auf die DynamoDB-Tabelle zugreifen kann.

    a. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die IAM-Konsole unter. https://console.aws.amazon.com/iam/

    b. Wählen Sie Rollen und erstellen Sie eine neue Rolle mit dem Namen`Cross-Account-Role-B`. Weitere Informationen zum Erstellen von IAM-Rollen finden Sie unter Erstellen von IAM-Rollen im Benutzerhandbuch.

    c. Erstellen Sie eine IAM-Richtlinie, die Berechtigungen für den Zugriff auf die kontoübergreifende DynamoDB-Tabelle gewährt. Fügen Sie die IAM-Richtlinie an `Cross-Account-Role-B` an.

    Die folgende Richtlinie gewährt Zugriff auf eine DynamoDB-Tabelle. `CrossAccountTable`

    d. So bearbeiten Sie die Vertrauensbeziehung für die `Cross-Account-Role-B`-Rolle.

    Um die Vertrauensstellung für die Rolle zu konfigurieren, wählen Sie in der IAM-Konsole die Registerkarte Trust Relationships für die Rolle aus, die Sie in Schritt 2 erstellt haben:. Cross-Account-Role-B

    Wählen Sie Vertrauensstellung bearbeiten aus und fügen Sie dann das folgende Richtliniendokument hinzu. Dieses Dokument ermöglicht es Ihnen`Job-Execution-Role-A`, diese `Cross-Account-Role-B` Rolle `AccountA` zu übernehmen.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole"
      ],
      "Resource": "arn:aws:iam::123456789012:role/Job-Execution-Role-A",
      "Sid": "AllowSTSAssumerole"
    }
  ]
}
```

e. Geben `Job-Execution-Role-A` Sie ein und `AccountA` erhalten `- STS Assume role` Sie die erforderlichen Rechte zur Übernahme`Cross-Account-Role-B`.

Wählen Sie in der IAM-Konsole für AWS-Konto `AccountA` die Option`Job-Execution-Role-A`. Fügen Sie die folgende Richtlinienanweisung zu `Job-Execution-Role-A` hinzu, um die `AssumeRole`-Aktion in der Rolle `Cross-Account-Role-B` zu verweigern.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole"
      ],
      "Resource": [
        "arn:aws:iam::123456789012:role/Cross-Account-Role-B"
      ],
      "Sid": "AllowSTSAssumerole"
    }
  ]
}
```

f. Stellen Sie die `dynamodb.customAWSCredentialsProvider` Eigenschaft mit einem Wert wie `com.amazonaws.emr.AssumeRoleAWSCredentialsProvider` bei der Core-Site-Klassifizierung ein. Setzen Sie die Umgebungsvariable auf `ASSUME_ROLE_CREDENTIALS_ROLE_ARN` den ARN-Wert von`Cross-Account-Role-B`.

3. Führen Sie den Spark- oder Hive-Job mit`Job-Execution-Role-A`.

Überlegungen

Beachten Sie diese Verhaltensweisen und Einschränkungen, wenn Sie den DynamoDB-Connector mit Apache Spark oder Apache Hive verwenden.

Überlegungen zur Verwendung des DynamoDB-Connectors mit Apache Spark

- Spark SQL unterstützt die Erstellung einer Hive-Tabelle mit der Storage-Handler-Option nicht. Weitere Informationen finden Sie unter Speicherformat für Hive-Tabellen angeben in der Apache Spark-Dokumentation.
- Spark SQL unterstützt den `STORED BY` Vorgang mit dem Storage-Handler nicht. Wenn Sie über eine externe Hive-Tabelle mit einer DynamoDB-Tabelle interagieren möchten, verwenden Sie Hive, um die Tabelle zuerst zu erstellen.
- Um eine Abfrage in eine DynamoDB-Abfrage zu übersetzen, verwendet der DynamoDB-Connector das Prädikat Pushdown. Das Prädikat Pushdown filtert Daten nach einer Spalte, die dem Partitionsschlüssel einer DynamoDB-Tabelle zugeordnet ist. Predicate Pushdown funktioniert nur, wenn Sie den Konnektor mit Spark SQL und nicht mit der API verwenden. MapReduce

Überlegungen zur Verwendung des DynamoDB-Connectors mit Apache Hive

Einstellung der maximalen Anzahl von Mappern

- Wenn Sie die `SELECT` Abfrage verwenden, um Daten aus einer externen Hive-Tabelle zu lesen, die DynamoDB zugeordnet ist, wird die Anzahl der Zuordnungsaufgaben auf EMR Serverless als der gesamte für die DynamoDB-Tabelle konfigurierte Lesedurchsatz geteilt durch den Durchsatz pro Zuordnungsaufgabe berechnet. Der Standarddurchsatz pro Kartenaufgabe ist 100.
- Der Hive-Job kann je nach dem für DynamoDB konfigurierten Lesedurchsatz die Anzahl der Zuordnungsaufgaben verwenden, die über die maximale Anzahl von Containern hinausgeht, die pro EMR Serverless-Anwendung konfiguriert sind. Außerdem kann eine Hive-Abfrage mit langer

Laufzeit die gesamte bereitgestellte Lesekapazität der DynamoDB-Tabelle verbrauchen. Dies wirkt sich negativ auf andere Benutzer aus.

- Sie können die `dynamodb.max.map.tasks` Eigenschaft verwenden, um eine Obergrenze für Kartenaufgaben festzulegen. Sie können diese Eigenschaft auch verwenden, um die von jeder Kartenaufgabe gelesene Datenmenge auf der Grundlage der Größe des Aufgabencontainers zu optimieren.
- Sie können die `dynamodb.max.map.tasks` Eigenschaft auf Hive-Abfrageebene oder in der `hive-site` Klassifizierung des start-job-run Befehls festlegen. Dieser Wert muss gleich oder größer 1 sein. Wenn Hive Ihre Abfrage verarbeitet, verwendet der resultierende Hive-Job nicht mehr als die Werte, die `dynamodb.max.map.tasks` beim Lesen aus der DynamoDB-Tabelle angegeben wurden.

Den Schreibdurchsatz pro Aufgabe optimieren

- Der Schreibdurchsatz pro Aufgabe auf EMR Serverless wird berechnet als der gesamte Schreibdurchsatz, der für eine DynamoDB-Tabelle konfiguriert ist, geteilt durch den Wert der Eigenschaft. `mapreduce.job.maps` Für Hive ist der Standardwert dieser Eigenschaft 2. Somit können die ersten beiden Aufgaben in der letzten Phase des Hive-Jobs den gesamten Schreibdurchsatz verbrauchen. Dies führt zu einer Drosselung der Schreibvorgänge anderer Aufgaben im selben Job oder in anderen Jobs.
- Um Schreibeinschränkungen zu vermeiden, legen Sie den Wert der `mapreduce.job.maps` Eigenschaft auf der Grundlage der Anzahl der Aufgaben in der letzten Phase oder des Schreibdurchsatzes fest, den Sie pro Aufgabe zuweisen möchten. Legen Sie diese Eigenschaft in der `mapred-site` Klassifizierung des start-job-run Befehls auf EMR Serverless fest.

Sicherheit

Cloud-Sicherheit hat AWS höchste Priorität. Als AWS Kunde profitieren Sie von einer Rechenzentrums- und Netzwerkarchitektur, die darauf ausgelegt sind, die Anforderungen der sicherheitssensibelsten Unternehmen zu erfüllen.

Sicherheit ist eine gemeinsame Verantwortung von Ihnen AWS und Ihnen. Das Modell der geteilten Verantwortung beschreibt dies als Sicherheit der Cloud und Sicherheit in der Cloud:

- Sicherheit der Cloud — AWS ist verantwortlich für den Schutz der Infrastruktur, die AWS Dienste in der AWS Cloud ausführt. AWS bietet Ihnen auch Dienste, die Sie sicher nutzen. Auditoren von Drittanbietern testen und überprüfen die Effektivität unserer Sicherheitsmaßnahmen im Rahmen der AWS -Compliance-Programme regelmäßig. Weitere Informationen zu den Compliance-Programmen, die für Amazon EMR Serverless gelten, finden Sie unter AWS Services im Umfang nach Compliance-Programmen.
- Sicherheit in der Cloud — Ihre Verantwortung richtet sich nach dem AWS Service, den Sie nutzen. Sie sind auch für andere Faktoren verantwortlich, einschließlich der Vertraulichkeit Ihrer Daten, für die Anforderungen Ihres Unternehmens und für die geltenden Gesetze und Vorschriften.

Diese Dokumentation hilft Ihnen zu verstehen, wie Sie das Modell der gemeinsamen Verantwortung bei der Verwendung von Amazon EMR Serverless anwenden können. In den Themen wird erklärt, wie Sie Amazon EMR Serverless konfigurieren und andere AWS Services nutzen können, um Ihre Sicherheits- und Compliance-Ziele zu erreichen.

Themen

- Protokollieren von Amazon EMR Serverless API-Aufrufen mit AWS CloudTrail
- Konformitätsprüfung für Amazon EMR Serverless
- Ausfallsicherheit in Amazon EMR Serverless
- Infrastruktursicherheit in Amazon EMR Serverless
- Konfiguration und Schwachstellenanalyse in Amazon EMR Serverless

Bewährte Sicherheitsmethoden für Amazon EMR Serverless

Amazon EMR Serverless bietet eine Reihe von Sicherheitsfunktionen, die Sie bei der Entwicklung und Implementierung Ihrer eigenen Sicherheitsrichtlinien berücksichtigen sollten. Die folgenden bewährten Methoden sind allgemeine Richtlinien und keine vollständige Sicherheitslösung. Da diese bewährten Methoden für Ihre Umgebung möglicherweise nicht angemessen oder ausreichend sind, sollten Sie sie als hilfreiche Überlegungen und nicht als bindend ansehen.

Anwendung des Prinzips der geringsten Privilegien

EMR Serverless bietet eine detaillierte Zugriffsrichtlinie für Anwendungen, die IAM-Rollen verwenden, z. B. Ausführungsrollen. Wir empfehlen, Ausführungsrollen nur die für den Job erforderlichen Mindestberechtigungen zu gewähren, z. B. die Abdeckung Ihrer Anwendung und den Zugriff auf das Protokollziel. Wir empfehlen außerdem, die Aufträge regelmäßig und bei jeder Änderung des Anwendungscodes auf ihre Berechtigungen hin zu überprüfen.

Den Code nicht vertrauenswürdiger Anwendungen isolieren

EMR Serverless sorgt für eine vollständige Netzwerkisolierung zwischen Jobs, die zu verschiedenen EMR Serverless-Anwendungen gehören. In Fällen, in denen eine Isolierung auf Jobebene gewünscht wird, sollten Sie erwägen, Jobs in verschiedene EMR Serverless-Anwendungen zu isolieren.

Rollenbasierte Zugriffskontrolle (RBAC) Berechtigungen

Administratoren sollten die Berechtigungen für die rollenbasierte Zugriffskontrolle (RBAC) für serverlose EMR-Anwendungen strikt kontrollieren.

Datenschutz

Das Modell der AWS gemeinsamen Verantwortung gilt für den Datenschutz in Amazon EMR Serverless. AWS Ist, wie in diesem Modell beschrieben, für den Schutz der globalen Infrastruktur

verantwortlich, auf der die AWS gesamte Cloud läuft. Sie sind dafür verantwortlich, die Kontrolle über Ihre in dieser Infrastruktur gehosteten Inhalte zu behalten. Dieser Inhalt umfasst die Sicherheitskonfiguration und die Verwaltungsaufgaben für die AWS Dienste, die Sie verwenden. Weitere Informationen zum Datenschutz finden Sie in den häufig gestellten Fragen zum Datenschutz. Informationen zum Datenschutz in Europa finden Sie im Blogbeitrag AWS Shared Responsibility Model und GDPR auf dem AWS Security Blog.

Aus Datenschutzgründen empfehlen wir Ihnen, die AWS Kontoanmeldeinformationen zu schützen und individuelle Konten mit AWS Identity and Access Management (IAM) einzurichten. So erhält jeder Benutzer nur die Berechtigungen, die zum Durchführen seiner Aufgaben erforderlich sind. Außerdem sollten Sie die Daten mit folgenden Methoden schützen:

- Verwenden Sie für jedes Konto die Multi-Faktor-Authentifizierung (MFA).
- Wird verwendet SSL/TLS , um mit AWS Ressourcen zu kommunizieren. Wir empfehlen TLS 1.2 oder höher.
- Richten Sie die API und die Protokollierung von Benutzeraktivitäten mit ein AWS CloudTrail.
- Verwenden Sie AWS Verschlüsselungslösungen zusammen mit allen Standardsicherheitskontrollen innerhalb der AWS Dienste.
- Verwenden Sie erweiterte verwaltete Sicherheitsservices wie Amazon Macie, die dabei helfen, in Amazon S3 gespeicherte persönliche Daten zu erkennen und zu sichern.
- Verwenden Sie die serverlosen Verschlüsselungsoptionen von Amazon EMR, um Daten im Ruhezustand und bei der Übertragung zu verschlüsseln.
- Wenn Sie für den Zugriff AWS über eine Befehlszeilenschnittstelle oder eine API FIPS 140-2-validierte kryptografische Module benötigen, verwenden Sie einen FIPS-Endpunkt. Weitere Informationen zu den verfügbaren FIPS-Endpunkten finden Sie im Federal Information Processing Standard (FIPS) 140-2.

Wir empfehlen dringend, dass Sie niemals vertrauliche Identifikationsinformationen, wie z. B. die Kontonummern Ihrer Kunden, in Freiformfelder wie ein Namensfeld eingeben. Dies gilt auch, wenn Sie mit Amazon EMR Serverless oder anderen AWS Services über die Konsole AWS CLI, API oder arbeiten. AWS SDKs Alle Daten, die Sie in Amazon EMR Serverless oder andere Services eingeben, werden möglicherweise für die Aufnahme in Diagnoseprotokolle aufgenommen. Wenn Sie eine URL für einen externen Server bereitstellen, schließen Sie keine Anmeldeinformationen zur Validierung Ihrer Anforderung an den betreffenden Server in die URL ein.

Verschlüsselung im Ruhezustand

Die Datenverschlüsselung verhindert, dass nicht autorisierte Benutzer Daten auf einem Cluster und in den dazugehörigen Datenspeichersystemen lesen können. Dies gilt für auf persistenten Medien gespeicherte Daten, auch als Daten im Ruhezustand bezeichnet, und für Daten, die während der Übertragung im Netzwerk möglicherweise abgefangen werden, auch als Daten während der Übertragung bezeichnet.

Die Datenverschlüsselung erfordert Aktivierungsschlüssel und Zertifikate. Sie können aus verschiedenen Optionen wählen, darunter Schlüssel AWS Key Management Service, die von Amazon S3 verwaltet werden, sowie Schlüssel und Zertifikate von benutzerdefinierten Anbietern, die Sie bereitstellen. Bei der Nutzung AWS KMS als Schlüsselanbieter fallen Gebühren für die Speicherung und Verwendung von Verschlüsselungsschlüsseln an. Weitere Informationen finden Sie unter AWS KMS Preise.

Bevor Sie Verschlüsselungsoptionen angeben, entscheiden Sie sich für die Schlüssel- und Zertifikatsverwaltungssysteme, die Sie verwenden möchten. Erstellen Sie anschließend die Schlüssel und Zertifikate für die benutzerdefinierten Anbieter, die Sie im Rahmen der Verschlüsselungseinstellungen angeben.

Verschlüsselung im Ruhezustand von EMRFS-Daten in Amazon S3

Jede EMR Serverless-Anwendung verwendet eine bestimmte Release-Version, die EMRFS (EMR File System) enthält. Die Amazon-S3-Verschlüsselung funktioniert mit EMR File System (EMRFS)-Objekten, die gelesen werden und zu Amazon S3 geschrieben werden. Sie können die serverseitige Verschlüsselung (SSE) oder die clientseitige Verschlüsselung (CSE) von Amazon S3 als Standardverschlüsselungsmodus angeben, wenn Sie die Verschlüsselung im Ruhezustand aktivieren. Geben Sie optional verschiedene Verschlüsselungsmethoden für einzelne Buckets an, indem Sie die Verschlüsselungsüberschreibungen pro Bucket verwenden. Unabhängig davon, ob Amazon-S3-Verschlüsselung aktiviert ist, verschlüsselt Transport Layer Security (TLS) EMRFS-Objekte bei der Übertragung zwischen EMR-Cluster-Knoten und Amazon S3. Wenn Sie Amazon S3 CSE mit vom Kunden verwalteten Schlüsseln verwenden, muss Ihre Ausführungsrolle, mit der Jobs in einer serverlosen EMR-Anwendung ausgeführt werden, Zugriff auf den Schlüssel haben. Ausführliche Informationen zur Amazon S3 S3-Verschlüsselung finden Sie unter Schützen von Daten mithilfe von Verschlüsselung im Amazon Simple Storage Service Developer Guide.

> **Note**
>
> Bei der Nutzung AWS KMS fallen Gebühren für die Speicherung und Verwendung von Verschlüsselungsschlüsseln an. Weitere Informationen finden Sie unter AWS KMS Preise.

Serverseitige Verschlüsselung im Amazon S3

Für alle Amazon S3-Buckets ist die Verschlüsselung standardmäßig konfiguriert, und alle neuen Objekte, die in einen S3-Bucket hochgeladen werden, werden im Ruhezustand automatisch verschlüsselt. Amazon S3 verschlüsselt Daten auf Objektebene, wenn die Daten auf die Festplatte geschrieben werden, und entschlüsselt die Daten, wenn darauf zugegriffen wird. Weitere Informationen zu SSE finden Sie unter Schützen von Daten mithilfe serverseitiger Verschlüsselung im Amazon Simple Storage Service Developer Guide.

Sie können zwischen zwei verschiedenen Schlüsselverwaltungssystemen wählen, wenn Sie SSE in Amazon EMR Serverless angeben:

- SSE-S3 – Hierbei verwaltet Amazon S3 die Aktivierungsschlüssel für Sie. Für EMR Serverless ist keine zusätzliche Einrichtung erforderlich.
- SSE-KMS - Sie verwenden eine AWS KMS key , um Richtlinien einzurichten, die für EMR Serverless geeignet sind. Für EMR Serverless ist keine zusätzliche Einrichtung erforderlich.

Um die AWS KMS Verschlüsselung für Daten zu verwenden, die Sie in Amazon S3 schreiben, haben Sie zwei Möglichkeiten, wenn Sie die `StartJobRun` API verwenden. Sie können entweder die Verschlüsselung für alles aktivieren, was Sie in Amazon S3 schreiben, oder die Verschlüsselung für Daten aktivieren, die Sie in einen bestimmten Bucket schreiben. Weitere Informationen zur `StartJobRun` API finden Sie in der EMR Serverless API Reference.

Verwenden Sie beim Aufrufen der `StartJobRun` API die folgenden Befehle, um die AWS KMS Verschlüsselung für alle Daten zu aktivieren, die Sie in Amazon S3 schreiben.

```
--conf spark.hadoop.fs.s3.enableServerSideEncryption=true
--conf spark.hadoop.fs.s3.serverSideEncryption.kms.keyId=<kms_id>
```

Um die AWS KMS Verschlüsselung für Daten zu aktivieren, die Sie in einen bestimmten Bucket schreiben, verwenden Sie beim Aufrufen der `StartJobRun` API die folgenden Befehle.

```
--conf spark.hadoop.fs.s3.bucket.<amzn-s3-demo-bucket1>.enableServerSideEncryption=true
--conf spark.hadoop.fs.s3.bucket.<amzn-s3-demo-
bucket1>.serverSideEncryption.kms.keyId=<kms-id>
```

SSE mit vom Kunden bereitgestellten Schlüsseln (SSE-C) ist nicht für die Verwendung mit EMR Serverless verfügbar.

Clientseitige Verschlüsselung für Amazon S3

Bei der clientseitigen Amazon S3 S3-Verschlüsselung erfolgt die Amazon S3 S3-Verschlüsselung und -Entschlüsselung im EMRFS-Client, der in jeder Amazon EMR-Version verfügbar ist. Objekte werden vor dem Hochladen nach Amazon S3 verschlüsselt und nach dem Herunterladen entschlüsselt. Der von Ihnen festgelegte Anbieter stellt den vom Client verwendeten Verschlüsselungsschlüssel bereit. Der Client kann vom AWS KMS bereitgestellte Schlüssel (CSE-KMS) oder eine benutzerdefinierte Java-Klasse verwenden, die den clientseitigen Root-Schlüssel (CSE-C) bereitstellt. Die Verschlüsselungseigenschaften unterscheiden sich geringfügig zwischen CSE-KMS und CSE-C, abhängig vom festgelegten Anbieter und von den Metadaten des Objekts, das entschlüsselt oder verschlüsselt werden soll. Wenn Sie Amazon S3 CSE mit vom Kunden verwalteten Schlüsseln verwenden, muss Ihre Ausführungsrolle, mit der Jobs in einer serverlosen EMR-Anwendung ausgeführt werden, Zugriff auf den Schlüssel haben. Zusätzliche KMS-Gebühren können anfallen. Weitere Informationen zu diesen Unterschieden finden Sie unter Schützen von Daten mit clientseitiger Verschlüsselung im Amazon Simple Storage Service Developer Guide.

Verschlüsselung lokaler Datenträger

Daten, die im flüchtigen Speicher gespeichert sind, werden mit diensteigenen Schlüsseln verschlüsselt, wobei der kryptografische Algorithmus AES-256 nach Industriestandard verwendet wird.

Schlüsselverwaltung

Sie können KMS so konfigurieren, dass Ihre KMS-Schlüssel automatisch rotiert werden. Dadurch werden Ihre Schlüssel einmal im Jahr rotiert, während alte Schlüssel auf unbestimmte Zeit gespeichert werden, sodass Ihre Daten weiterhin entschlüsselt werden können. Weitere Informationen finden Sie unter Rotierende, vom Kunden verwaltete Schlüssel.

Verschlüsselung während der Übertragung

Die folgenden anwendungsspezifischen Verschlüsselungsfunktionen sind mit Amazon EMR Serverless verfügbar:

- Spark
  - Standardmäßig ist die Kommunikation zwischen Spark-Treibern und Executoren authentifiziert und intern. Die RPC-Kommunikation zwischen Treibern und Executoren ist verschlüsselt.
- Hive
  - Die Kommunikation zwischen dem AWS Glue Metastore und den EMR Serverless-Anwendungen erfolgt über TLS.

Sie sollten nur verschlüsselte Verbindungen über HTTPS (TLS) zulassen, indem Sie die aws: SecureTransport -Bedingung in den Amazon S3 S3-Bucket-IAM-Richtlinien verwenden.

Identity and Access Management (IAM) in Amazon EMR Serverless

AWS Identity and Access Management (IAM) hilft einem Administrator AWS-Service , den Zugriff auf Ressourcen sicher zu kontrollieren. AWS IAM-Administratoren kontrollieren, wer authentifiziert (angemeldet) und autorisiert werden kann (über Berechtigungen verfügt), um Amazon EMR Serverless-Ressourcen zu verwenden. IAM ist ein Programm AWS-Service , das Sie ohne zusätzliche Kosten nutzen können.

Themen

- Zielgruppe
- Authentifizierung mit Identitäten
- Verwalten des Zugriffs mit Richtlinien
- So funktioniert EMR Serverless mit IAM
- Verwenden von serviceverknüpften Rollen für EMR Serverless
- Job-Runtime-Rollen für Amazon EMR Serverless
- Beispiele für Benutzerzugriffsrichtlinien für EMR Serverless
- Richtlinien für Tag-basierte Zugriffskontrolle
- Beispiele für identitätsbasierte Richtlinien für EMR Serverless

- Serverlose Amazon EMR-Updates für verwaltete Richtlinien AWS
- Fehlerbehebung für Amazon EMR Serverless Identity and Access

Zielgruppe

Wie Sie AWS Identity and Access Management (IAM) verwenden, hängt von Ihrer Rolle ab:

- Servicebenutzer – Fordern Sie von Ihrem Administrator Berechtigungen an, wenn Sie nicht auf Features zugreifen können (siehe Fehlerbehebung für Amazon EMR Serverless Identity and Access).
- Serviceadministrator – Bestimmen Sie den Benutzerzugriff und stellen Sie Berechtigungsanfragen (siehe Identity and Access Management (IAM) in Amazon EMR Serverless).
- IAM-Administrator – Schreiben Sie Richtlinien zur Zugriffsverwaltung (siehe Beispiele für identitätsbasierte Richtlinien für EMR Serverless).

Authentifizierung mit Identitäten

Authentifizierung ist die Art und Weise, wie Sie sich AWS mit Ihren Identitätsdaten anmelden. Sie müssen sich als IAM-Benutzer authentifizieren oder eine IAM-Rolle annehmen. Root-Benutzer des AWS-Kontos

Sie können sich als föderierte Identität anmelden, indem Sie Anmeldeinformationen aus einer Identitätsquelle wie AWS IAM Identity Center (IAM Identity Center), Single Sign-On-Authentifizierung oder Anmeldeinformationen verwenden. Google/Facebook Weitere Informationen zum Anmelden finden Sie unter So melden Sie sich bei Ihrem AWS-Konto an im Benutzerhandbuch für AWS-Anmeldung .

AWS Bietet für den programmatischen Zugriff ein SDK und eine CLI zum kryptografischen Signieren von Anfragen. Weitere Informationen finden Sie unter AWS Signature Version 4 for API requests im IAM-Benutzerhandbuch.

AWS-Konto Root-Benutzer

Wenn Sie einen erstellen AWS-Konto, beginnen Sie mit einer Anmeldeidentität, dem sogenannten AWS-Konto Root-Benutzer, der vollständigen Zugriff auf alle AWS-Services Ressourcen hat. Wir raten ausdrücklich davon ab, den Root-Benutzer für Alltagsaufgaben zu verwenden. Eine Liste der

Aufgaben, für die Sie sich als Root-Benutzer anmelden müssen, finden Sie unter Tasks that require root user credentials im IAM-Benutzerhandbuch.

Verbundidentität

Als bewährte Methode sollten menschliche Benutzer für den Zugriff AWS-Services mithilfe temporärer Anmeldeinformationen einen Verbund mit einem Identitätsanbieter verwenden.

Eine föderierte Identität ist ein Benutzer aus Ihrem Unternehmensverzeichnis, Ihrem Directory Service Web-Identitätsanbieter oder der AWS-Services mithilfe von Anmeldeinformationen aus einer Identitätsquelle zugreift. Verbundene Identitäten übernehmen Rollen, die temporäre Anmeldeinformationen bereitstellen.

Für die zentrale Zugriffsverwaltung empfehlen wir AWS IAM Identity Center. Weitere Informationen finden Sie unter Was ist IAM Identity Center? im AWS IAM Identity Center -Benutzerhandbuch.

IAM-Benutzer und -Gruppen

Ein IAM-Benutzer ist eine Identität mit bestimmten Berechtigungen für eine einzelne Person oder Anwendung. Wir empfehlen die Verwendung temporärer Anmeldeinformationen anstelle von IAM-Benutzern mit langfristigen Anmeldeinformationen. Weitere Informationen finden Sie im IAM-Benutzerhandbuch unter Erfordern, dass menschliche Benutzer den Verbund mit einem Identitätsanbieter verwenden müssen, um AWS mithilfe temporärer Anmeldeinformationen darauf zugreifen zu können.

Eine IAM-Gruppe spezifiziert eine Sammlung von IAM-Benutzern und erleichtert die Verwaltung von Berechtigungen für große Gruppen von Benutzern. Weitere Informationen finden Sie unter Anwendungsfälle für IAM-Benutzer im IAM-Benutzerhandbuch.

IAM-Rollen

Eine IAM-Rolle ist eine Identität mit spezifischen Berechtigungen, die temporäre Anmeldeinformationen bereitstellt. Sie können eine Rolle übernehmen, indem Sie von einer Benutzer- zu einer IAM-Rolle (Konsole) wechseln AWS CLI oder einen AWS API-Vorgang aufrufen. Weitere Informationen finden Sie unter Methoden, um eine Rolle zu übernehmen im IAM-Benutzerhandbuch.

IAM-Rollen sind nützlich für den Verbundbenutzer-Zugriff, temporäre IAM-Benutzerberechtigungen, kontoübergreifenden Zugriff, serviceübergreifenden Zugriff und Anwendungen, die auf Amazon EC2 laufen. Weitere Informationen finden Sie unter Kontoübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.

Verwalten des Zugriffs mit Richtlinien

Sie kontrollieren den Zugriff, AWS indem Sie Richtlinien erstellen und diese an AWS Identitäten oder Ressourcen anhängen. Eine Richtlinie definiert Berechtigungen, wenn sie mit einer Identität oder Ressource verknüpft sind. AWS bewertet diese Richtlinien, wenn ein Principal eine Anfrage stellt. Die meisten Richtlinien werden AWS als JSON-Dokumente gespeichert. Weitere Informationen zu JSON-Richtliniendokumenten finden Sie unter Übersicht über JSON-Richtlinien im IAM-Benutzerhandbuch.

Mit Hilfe von Richtlinien legen Administratoren fest, wer Zugriff auf was hat, indem sie definieren, welches Prinzipal welche Aktionen auf welchen Ressourcenund unter welchen Bedingungendurchführen darf.

Standardmäßig haben Benutzer, Gruppen und Rollen keine Berechtigungen. Ein IAM-Administrator erstellt IAM-Richtlinien und fügt sie zu Rollen hinzu, die die Benutzer dann übernehmen können. IAM-Richtlinien definieren Berechtigungen unabhängig von der Methode, die zur Ausführung der Operation verwendet wird.

Identitätsbasierte Richtlinien

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität (Benutzer, Gruppe oder Rolle) anfügen können. Diese Richtlinien steuern, welche Aktionen Identitäten für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Definieren benutzerdefinierter IAM-Berechtigungen mit vom Kunden verwalteten Richtlinien im IAM-Benutzerhandbuch.

Identitätsbasierte Richtlinien können Inline-Richtlinien (direkt in eine einzelne Identität eingebettet) oder verwaltete Richtlinien (eigenständige Richtlinien, die mit mehreren Identitäten verbunden sind) sein. Informationen dazu, wie Sie zwischen verwalteten und Inline-Richtlinien wählen, finden Sie unter Choose between managed policies and inline policies im IAM-Benutzerhandbuch.

Ressourcenbasierte Richtlinien

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele hierfür sind Vertrauensrichtlinien für IAM-Rollen und Amazon S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben.

Ressourcenbasierte Richtlinien sind Richtlinien innerhalb dieses Diensts. Sie können AWS verwaltete Richtlinien von IAM nicht in einer ressourcenbasierten Richtlinie verwenden.

Weitere Richtlinientypen

AWS unterstützt zusätzliche Richtlinientypen, mit denen die maximalen Berechtigungen festgelegt werden können, die durch gängigere Richtlinientypen gewährt werden:

- Berechtigungsgrenzen – Eine Berechtigungsgrenze legt die maximalen Berechtigungen fest, die eine identitätsbasierte Richtlinie einer IAM-Entität erteilen kann. Weitere Informationen finden Sie unter Berechtigungsgrenzen für IAM-Entitäten im -IAM-Benutzerhandbuch.
- Richtlinien zur Dienstkontrolle (SCPs) — Geben Sie die maximalen Berechtigungen für eine Organisation oder Organisationseinheit in an AWS Organizations. Weitere Informationen finden Sie unter Service-Kontrollrichtlinien im AWS Organizations -Benutzerhandbuch.
- Richtlinien zur Ressourcenkontrolle (RCPs) — Legen Sie die maximal verfügbaren Berechtigungen für Ressourcen in Ihren Konten fest. Weitere Informationen finden Sie im AWS Organizations Benutzerhandbuch unter Richtlinien zur Ressourcenkontrolle (RCPs).
- Sitzungsrichtlinien – Sitzungsrichtlinien sind erweiterte Richtlinien, die als Parameter übergeben werden, wenn Sie eine temporäre Sitzung für eine Rolle oder einen Verbundbenutzer erstellen. Weitere Informationen finden Sie unter Sitzungsrichtlinien im IAM-Benutzerhandbuch.

Mehrere Richtlinientypen

Wenn für eine Anfrage mehrere Arten von Richtlinien gelten, sind die daraus resultierenden Berechtigungen schwieriger zu verstehen. Informationen darüber, wie AWS bestimmt wird, ob eine Anfrage zulässig ist, wenn mehrere Richtlinientypen betroffen sind, finden Sie unter Bewertungslogik für Richtlinien im IAM-Benutzerhandbuch.

So funktioniert EMR Serverless mit IAM

Bevor Sie IAM verwenden, um den Zugriff auf Amazon EMR Serverless zu verwalten, sollten Sie sich darüber informieren, welche IAM-Funktionen für Amazon EMR Serverless verfügbar sind.

Verwendung von IAM-Funktionen mit EMR Serverless

IAM-Feature	Serverloser Support für Amazon EMR
Identitätsbasierte Richtlinien	Ja
Ressourcenbasierte Richtlinien	Nein

IAM-Feature	Serverloser Support für Amazon EMR
Richtlinienaktionen	Ja
Richtlinienressourcen	Ja
Bedingungsschlüssel für die Richtlinie	Nein
ACLs	Nein
ABAC (Tags in Richtlinien)	Ja
Temporäre Anmeldeinformationen	Ja
Prinzipalberechtigungen	Ja
Servicerollen	Nein
Serviceverknüpfte Rollen	Ja

Einen allgemeinen Überblick darüber, wie EMR Serverless und andere AWS Dienste mit den meisten IAM-Funktionen funktionieren, finden Sie im IAM-Benutzerhandbuch unter AWS Dienste, die mit IAM funktionieren.

Identitätsbasierte Richtlinien für EMR Serverless

Unterstützt Richtlinien auf Identitätsbasis: Ja

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität anfügen können, wie z. B. IAM-Benutzern, -Benutzergruppen oder -Rollen. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Definieren benutzerdefinierter IAM-Berechtigungen mit vom Kunden verwalteten Richtlinien im IAM-Benutzerhandbuch.

Mit identitätsbasierten IAM-Richtlinien können Sie angeben, welche Aktionen und Ressourcen zugelassen oder abgelehnt werden. Darüber hinaus können Sie die Bedingungen festlegen, unter denen Aktionen zugelassen oder abgelehnt werden. Informationen zu sämtlichen Elementen, die Sie in einer JSON-Richtlinie verwenden, finden Sie in der IAM-Referenz für JSON-Richtlinienelemente im IAM-Benutzerhandbuch.

Beispiele für identitätsbasierte Richtlinien für EMR Serverless

Beispiele für identitätsbasierte Richtlinien von Amazon EMR Serverless finden Sie unter. Beispiele für identitätsbasierte Richtlinien für EMR Serverless

Ressourcenbasierte Richtlinien innerhalb von EMR Serverless

Unterstützt ressourcenbasierte Richtlinien: Nein

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele für ressourcenbasierte Richtlinien sind IAM-Rollen-Vertrauensrichtlinien und Amazon-S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Für die Ressource, an welche die Richtlinie angehängt ist, legt die Richtlinie fest, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für diese Ressource ausführen kann. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben. Zu den Prinzipalen können Konten, Benutzer, Rollen, Verbundbenutzer oder gehören. AWS-Services

Um kontoübergreifenden Zugriff zu ermöglichen, können Sie ein gesamtes Konto oder IAM-Entitäten in einem anderen Konto als Prinzipal in einer ressourcenbasierten Richtlinie angeben. Weitere Informationen finden Sie unter Kontoübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.

Richtlinienaktionen für EMR Serverless

Unterstützt Richtlinienaktionen: Ja

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer auf was Zugriff hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Das Element `Action` einer JSON-Richtlinie beschreibt die Aktionen, mit denen Sie den Zugriff in einer Richtlinie zulassen oder verweigern können. Nehmen Sie Aktionen in eine Richtlinie auf, um Berechtigungen zur Ausführung des zugehörigen Vorgangs zu erteilen.

Eine Liste der EMR Serverless-Aktionen finden Sie unter Aktionen, Ressourcen und Bedingungsschlüssel für Amazon EMR Serverless in der Service Authorization Reference.

Richtlinienaktionen in EMR Serverless verwenden das folgende Präfix vor der Aktion.

```
emr-serverless
```

Um mehrere Aktionen in einer einzigen Anweisung anzugeben, trennen Sie sie mit Kommata:

```
"Action": [
      "emr-serverless:action1",
      "emr-serverless:action2"
         ]
```

Beispiele für identitätsbasierte Richtlinien von Amazon EMR Serverless finden Sie unter. Beispiele für identitätsbasierte Richtlinien für EMR Serverless

Richtlinienressourcen für EMR Serverless

Unterstützt Richtlinienressourcen: Ja

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer auf was Zugriff hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Das JSON-Richtlinienelement `Resource` gibt die Objekte an, auf welche die Aktion angewendet wird. Als Best Practice geben Sie eine Ressource mit dem zugehörigen Amazon-Ressourcennamen (ARN) an. Verwenden Sie für Aktionen, die keine Berechtigungen auf Ressourcenebene unterstützen, einen Platzhalter (*), um anzugeben, dass die Anweisung für alle Ressourcen gilt.

```
"Resource": "*"
```

Eine Liste der Amazon EMR Serverless-Ressourcentypen und ihrer ARNs Eigenschaften finden Sie unter Von Amazon EMR Serverless definierte Ressourcen in der Service Authorization Reference. Informationen darüber, welche Aktionen den ARN der einzelnen Ressourcen angeben, finden Sie unter Aktionen, Ressourcen und Bedingungsschlüssel für Amazon EMR Serverless.

Beispiele für identitätsbasierte Richtlinien von Amazon EMR Serverless finden Sie unter. Beispiele für identitätsbasierte Richtlinien für EMR Serverless

Schlüssel zur Richtlinienbedingung für EMR Serverless

Unterstützung von Richtlinien-Bedingungsschlüsseln

Unterstützt servicespezifische Richtlinienbedingungsschlüssel	Nein

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer auf was Zugriff hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Das Element `Condition` gibt an, wann Anweisungen auf der Grundlage definierter Kriterien ausgeführt werden. Sie können bedingte Ausdrücke erstellen, die Bedingungsoperatoren verwenden, z. B. ist gleich oder kleiner als, damit die Bedingung in der Richtlinie mit Werten in der Anforderung übereinstimmt. Eine Übersicht aller AWS globalen Bedingungsschlüssel finden Sie unter Kontextschlüssel für AWS globale Bedingungen im IAM-Benutzerhandbuch.

Eine Liste der Amazon EMR Serverless-Bedingungsschlüssel und Informationen darüber, welche Aktionen und Ressourcen Sie mit einem Bedingungsschlüssel verwenden können, finden Sie unter Aktionen, Ressourcen und Bedingungsschlüssel für Amazon EMR Serverless in der Service Authorization Reference.

Alle Amazon EC2-Aktionen unterstützen die Bedingungsschlüssel `aws:RequestedRegion` und `ec2:Region`. Weitere Informationen finden Sie unter Beispiel: Beschränken des Zugriffs auf eine bestimmte Region.

Zugriffskontrolllisten (ACLs) in EMR Serverless

Unterstützt ACLs: Nein

Zugriffskontrolllisten (ACLs) steuern, welche Principals (Kontomitglieder, Benutzer oder Rollen) über Zugriffsberechtigungen für eine Ressource verfügen. ACLs ähneln ressourcenbasierten Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

Attributbasierte Zugriffskontrolle (ABAC) mit EMR Serverless

Unterstützung für attributebasierte Zugriffskontrolle (ABAC)

Unterstützt ABAC (Tags in Richtlinien)	Ja

Die attributbasierte Zugriffskontrolle (ABAC) ist eine Autorisierungsstrategie, bei der Berechtigungen basierend auf Attributen, auch als Tags bezeichnet, definiert werden. Sie können Tags an IAM-Entitäten und AWS -Ressourcen anhängen und dann ABAC-Richtlinien entwerfen, die Operationen zulassen, wenn das Tag des Prinzipals mit dem Tag auf der Ressource übereinstimmt.

Um den Zugriff auf der Grundlage von Tags zu steuern, geben Sie im Bedingungselement einer Richtlinie Tag-Informationen an, indem Sie die Schlüssel `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, oder Bedingung `aws:TagKeys` verwenden.

Wenn ein Service alle drei Bedingungsschlüssel für jeden Ressourcentyp unterstützt, lautet der Wert für den Service Ja. Wenn ein Service alle drei Bedingungsschlüssel für nur einige Ressourcentypen unterstützt, lautet der Wert Teilweise.

Weitere Informationen zu ABAC finden Sie unter Definieren von Berechtigungen mit ABAC-Autorisierung im IAM-Benutzerhandbuch. Um ein Tutorial mit Schritten zur Einstellung von ABAC anzuzeigen, siehe Attributbasierte Zugriffskontrolle (ABAC) verwenden im IAM-Benutzerhandbuch.

Temporäre Anmeldeinformationen mit EMR Serverless verwenden

Unterstützt temporäre Anmeldeinformationen: Ja

Temporäre Anmeldeinformationen ermöglichen kurzfristigen Zugriff auf AWS Ressourcen und werden automatisch erstellt, wenn Sie einen Verbund verwenden oder die Rollen wechseln. AWS empfiehlt, temporäre Anmeldeinformationen dynamisch zu generieren, anstatt langfristige Zugriffsschlüssel zu verwenden. Weitere Informationen finden Sie unter Temporäre Anmeldeinformationen in IAM und AWS-Services , die mit IAM funktionieren im IAM-Benutzerhandbuch.

Serviceübergreifende Prinzipalberechtigungen für EMR Serverless

Unterstützt Forward Access Sessions (FAS): Ja

Forward Access Sessions (FAS) verwenden die Berechtigungen des Prinzipals, der einen aufruft, in Kombination mit der Anforderung AWS-Service, Anfragen an AWS-Service nachgelagerte Dienste zu stellen. Einzelheiten zu den Richtlinien für FAS-Anforderungen finden Sie unter Zugriffssitzungen weiterleiten.

Servicerollen für EMR Serverless

Unterstützt Servicerollen	Nein

Serviceverknüpfte Rollen für EMR Serverless

Unterstützt serviceverknüpfte Rollen	Ja

Einzelheiten zum Erstellen oder Verwalten von dienstbezogenen Rollen finden Sie unter AWS Dienste, die mit IAM funktionieren. Suchen Sie in der Tabelle nach einem Service mit einem `Yes` in der Spalte Service-linked role (Serviceverknüpfte Rolle). Wählen Sie den Link Ja, um auf die Dokumentation zu serviceverknüpften Rollen für diesen Dienst zuzugreifen.

Verwenden von serviceverknüpften Rollen für EMR Serverless

Amazon EMR Serverless verwendet AWS Identity and Access Management (IAM) serviceverknüpfte Rollen. Eine serviceverknüpfte Rolle ist eine einzigartige Art von IAM-Rolle, die direkt mit EMR Serverless verknüpft ist. Dienstbezogene Rollen sind von EMR Serverless vordefiniert und enthalten alle Berechtigungen, die der Dienst benötigt, um andere AWS Dienste in Ihrem Namen aufzurufen.

Eine serviceverknüpfte Rolle erleichtert die Einrichtung von EMR Serverless, da Sie die erforderlichen Berechtigungen nicht manuell hinzufügen müssen. EMR Serverless definiert die Berechtigungen seiner serviceverknüpften Rollen, und sofern nicht anders definiert, kann nur EMR Serverless seine Rollen übernehmen. Die definierten Berechtigungen umfassen die Vertrauens- und Berechtigungsrichtlinie. Diese Berechtigungsrichtlinie kann keinen anderen IAM-Entitäten zugewiesen werden.

Sie können eine serviceverknüpfte Rolle erst löschen, nachdem ihre verwandten Ressourcen gelöscht wurden. Dadurch werden Ihre serverlosen EMR-Ressourcen geschützt, da Sie die Zugriffsberechtigung für die Ressourcen nicht versehentlich entfernen können.

Informationen zu anderen Diensten, die dienstverknüpfte Rollen unterstützen, finden Sie unter AWS Services That Work with IAM. Suchen Sie in der Spalte Serviceverknüpfte Rollen nach den Diensten, für die Ja steht. Wählen Sie Ja mit einem Link aus, um auf die Dokumentation zu serviceverknüpften Rollen für diesen Dienst zuzugreifen.

Dienstbezogene Rollenberechtigungen für EMR Serverless

EMR Serverless verwendet die benannte dienstverknüpfte Rolle AWSServiceRoleForAmazonEMRServerless, damit es in AWS APIs Ihrem Namen anrufen kann.

Die AWSService RoleForAmazon EMRServerless dienstgebundene Rolle vertraut darauf, dass die folgenden Dienste die Rolle übernehmen:

- `ops.emr-serverless.amazonaws.com`

Die genannte Rollenberechtigungsrichtlinie `AmazonEMRServerlessServiceRolePolicy` ermöglicht es EMR Serverless, die folgenden Aktionen für die angegebenen Ressourcen durchzuführen.

> **Note**
>
> Der Inhalt der verwalteten Richtlinie ändert sich, sodass die hier gezeigte Richtlinie möglicherweise veraltet ist. Sehen Sie sich die meisten up-to-date Richtlinien von Amazon EMRServerless ServiceRolePolicy in der an AWS-Managementkonsole.

- Aktion: `ec2:CreateNetworkInterface`
- Aktion: `ec2:DeleteNetworkInterface`
- Aktion: `ec2:DescribeNetworkInterfaces`
- Aktion: `ec2:DescribeSecurityGroups`
- Aktion: `ec2:DescribeSubnets`
- Aktion: `ec2:DescribeVpcs`
- Aktion: `ec2:DescribeDhcpOptions`
- Aktion: `ec2:DescribeRouteTables`
- Aktion: `cloudwatch:PutMetricData`

Im Folgenden finden Sie die vollständige `AmazonEMRServerlessServiceRolePolicy` Richtlinie.

JSON

```
{
  "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"
          ]
        }
      }
    }
  ]
}
```

Die folgende Vertrauensrichtlinie ist dieser Rolle zugeordnet, damit der EMR Serverless-Prinzipal diese Rolle übernehmen kann.

JSON

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

```
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole"
      ],
      "Resource": "arn:aws:iam::123456789012:role/aws-service-role/emr-
serverless.amazonaws.com/AWSServiceRoleForEMRServerless",
      "Sid": "AllowSTSAssumerole"
    }
  ]
}
```

Sie müssen Berechtigungen konfigurieren, damit eine juristische Stelle von IAM (z. B. Benutzer, Gruppe oder Rolle) eine serviceverknüpfte Rolle erstellen, bearbeiten oder löschen kann. Weitere Informationen finden Sie unter Dienstbezogene Rollenberechtigungen im IAM-Benutzerhandbuch.

Eine serviceverknüpfte Rolle für EMR Serverless erstellen

Sie müssen eine serviceverknüpfte Rolle nicht manuell erstellen. Wenn Sie eine neue EMR Serverless-Anwendung in der AWS-Managementkonsole (mit EMR Studio), der oder der AWS API erstellen AWS CLI, erstellt EMR Serverless die serviceverknüpfte Rolle für Sie. Sie müssen Berechtigungen konfigurieren, damit eine juristische Stelle von IAM (z. B. Benutzer, Gruppe oder Rolle) eine serviceverknüpfte Rolle erstellen, bearbeiten oder löschen kann.

Um die serviceverknüpfte Rolle mithilfe von IAM zu erstellen AWSService RoleForAmazon EMRServerless

Fügen Sie der Berechtigungsrichtlinie für die IAM-Entität, die die dienstverknüpfte Rolle erstellen muss, die folgende Anweisung hinzu.

```
{
    "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"}}
}
```

Wenn Sie diese dienstverknüpfte Rolle löschen und sie dann erneut erstellen müssen, verwenden Sie dasselbe Verfahren, um die Rolle in Ihrem Konto neu zu erstellen. Wenn Sie eine neue EMR Serverless-Anwendung erstellen, erstellt EMR Serverless die serviceverknüpfte Rolle erneut für Sie.

Sie können die IAM-Konsole auch verwenden, um eine serviceverknüpfte Rolle mit dem Anwendungsfall EMR Serverless zu erstellen. Erstellen Sie in der AWS CLI oder der AWS API eine dienstverknüpfte Rolle mit dem Dienstnamen. `ops.emr-serverless.amazonaws.com` Weitere Informationen finden Sie unter Erstellen einer serviceverknüpften Rolle im IAM-Benutzerhandbuch. Wenn Sie diese serviceverknüpfte Rolle löschen, verwenden Sie denselben Prozess, um die Rolle erneut zu erstellen.

Bearbeiten einer serviceverknüpften Rolle für EMR Serverless

Mit EMR Serverless können Sie die AWSService RoleForAmazon EMRServerless serviceverknüpfte Rolle nicht bearbeiten, da verschiedene Entitäten möglicherweise auf die Rolle verweisen. Sie können die AWS eigene IAM-Richtlinie, die von der dienstverknüpften Rolle EMR Serverless verwendet wird, nicht bearbeiten, da sie alle erforderlichen Berechtigungen enthält, die EMR Serverless benötigt. Sie können jedoch die Beschreibung der Rolle mit IAM bearbeiten.

Um die Beschreibung der serviceverknüpften Rolle mithilfe von IAM zu bearbeiten AWSService RoleForAmazon EMRServerless

Fügen Sie die folgende Anweisung der Berechtigungsrichtlinie für die IAM-Entität hinzu, die die Beschreibung einer serviceverknüpften Rolle bearbeiten soll.

```
{
    "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"}}
}
```

Weitere Informationen finden Sie unter Bearbeiten einer serviceverknüpften Rolle im IAM-Benutzerhandbuch.

Löschen einer serviceverknüpften Rolle für EMR Serverless

Wenn Sie eine Funktion oder einen Dienst nicht mehr verwenden müssen, für den eine dienstverknüpfte Rolle erforderlich ist, empfehlen wir Ihnen, diese Rolle zu löschen. Auf diese Weise haben Sie keine ungenutzte Entität, die nicht aktiv überwacht oder verwaltet wird. Löschen Sie jedoch alle EMR Serverless-Anwendungen in allen Regionen, bevor Sie die dienstverknüpfte Rolle löschen.

> **Note**
>
> Wenn der EMR Serverless-Dienst die Rolle verwendet, wenn Sie versuchen, die mit der Rolle verknüpften Ressourcen zu löschen, schlägt das Löschen möglicherweise fehl. Wenn dies passiert, warten Sie einige Minuten und versuchen Sie es erneut.

Um die mit dem AWSService RoleForAmazon EMRServerless Dienst verknüpfte Rolle mithilfe von IAM zu löschen

Fügen Sie der Berechtigungsrichtlinie für die IAM-Entität, die eine dienstverknüpfte Rolle löschen muss, die folgende Anweisung hinzu.

```
{
    "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"}}
}
```

So löschen Sie die serviceverknüpfte Rolle mit IAM

Verwenden Sie die IAM-Konsole, die oder die AWS API AWS CLI, um die dienstverknüpfte Rolle zu löschen. AWSService RoleForAmazon EMRServerless Weitere Informationen finden Sie unter Löschen einer serviceverknüpften Rolle im IAM-Benutzerhandbuch.

Unterstützte Regionen für serverlose, serviceverknüpfte EMR-Rollen

EMR Serverless unterstützt die Verwendung von serviceverknüpften Rollen in allen Regionen, in denen der Service verfügbar ist. Weitere Informationen finden Sie unter AWS Regionen und Endpunkte.

Job-Runtime-Rollen für Amazon EMR Serverless

Sie können IAM-Rollenberechtigungen angeben, die ein serverloser EMR-Joblauf annehmen kann, wenn er andere Dienste in Ihrem Namen aufruft. Dies beinhaltet den Zugriff auf Amazon S3 für beliebige Datenquellen, Ziele und andere AWS Ressourcen wie Amazon Redshift Redshift-Cluster und DynamoDB-Tabellen. Weitere Informationen zum Erstellen einer Rolle finden Sie unter. Erstellen Sie eine Job-Runtime-Rolle

Beispiele für Laufzeitrichtlinien

Sie können einer Job-Runtime-Rolle eine Laufzeitrichtlinie wie die folgende hinzufügen. Die folgende Job-Runtime-Richtlinie ermöglicht:

- Lesezugriff auf Amazon S3 S3-Buckets mit EMR-Beispielen.
- Voller Zugriff auf S3-Buckets.
- Erstellen und lesen Sie den Zugriff auf den AWS Glue Data Catalog.

Um Zugriff auf andere AWS Ressourcen wie DynamoDB hinzuzufügen, müssen Sie beim Erstellen der Runtime-Rolle die entsprechenden Berechtigungen in die Richtlinie aufnehmen.

JSON

```
{
  "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:::amzn-s3-demo-bucket",
        "arn:aws:s3:::amzn-s3-demo-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": [
        "*"
      ]
    }
  ]
}
```

Übergeben Sie Rollenberechtigungen

Sie können IAM-Berechtigungsrichtlinien an die Rolle eines Benutzers anhängen, sodass der Benutzer nur genehmigte Rollen weitergeben kann. Auf diese Weise können Administratoren steuern, welche Benutzer bestimmte Job-Runtime-Rollen an EMR Serverless-Jobs übergeben können. Weitere Informationen zum Einrichten von Berechtigungen finden Sie unter Einem Benutzer Berechtigungen zur Übergabe einer Rolle an einen AWS Dienst erteilen.

Im Folgenden finden Sie eine Beispielrichtlinie, die es ermöglicht, eine Job-Runtime-Rolle an den EMR Serverless Service Principal zu übergeben.

```
{
     "Effect": "Allow",
     "Action": "iam:PassRole",
     "Resource": "arn:aws:iam::1234567890:role/JobRuntimeRoleForEMRServerless",
        "Condition": {
                "StringLike": {
                    "iam:PassedToService": "emr-serverless.amazonaws.com"
                }
            }
}
```

Verwaltete Berechtigungsrichtlinien, die Runtime-Rollen zugeordnet sind

Wenn Sie Jobausführungen über die EMR Studio-Konsole an EMR Serverless senden, müssen Sie in einem Schritt eine Runtime-Rolle auswählen, die Ihrer Anwendung zugeordnet werden soll. Jeder Auswahl in der Konsole sind zugrunde liegende verwaltete Richtlinien zugeordnet, die Sie unbedingt beachten sollten. Die drei Auswahlmöglichkeiten lauten wie folgt:

1. Alle Buckets — Wenn Sie diese Option wählen, wird die von AmazonS3 FullAccess AWS verwaltete Richtlinie angegeben, die vollen Zugriff auf alle Buckets gewährt.
2. Spezifische Buckets — Dies gibt den Amazon-Ressourcennamen (ARN) jedes Buckets an, den Sie auswählen. Es ist keine zugrunde liegende verwaltete Richtlinie enthalten.
3. Keine — Es sind keine Berechtigungen für verwaltete Richtlinien enthalten.

Wir empfehlen, bestimmte Buckets hinzuzufügen. Wenn Sie „Alle Buckets“ auswählen, denken Sie daran, dass dadurch der volle Zugriff für alle Buckets festgelegt wird.

Beispiele für Benutzerzugriffsrichtlinien für EMR Serverless

Sie können detaillierte Richtlinien für Ihre Benutzer einrichten, je nachdem, welche Aktionen jeder Benutzer bei der Interaktion mit EMR Serverless-Anwendungen ausführen soll. Die folgenden Richtlinien sind Beispiele, die Ihnen bei der Einrichtung der entsprechenden Berechtigungen für Ihre Benutzer helfen können. Dieser Abschnitt konzentriert sich nur auf serverlose EMR-Richtlinien. Beispiele für EMR Studio-Benutzerrichtlinien finden Sie unter EMR Studio-Benutzerberechtigungen konfigurieren. Informationen zum Anhängen von Richtlinien an IAM-Benutzer (Principals) finden Sie unter Verwaltung von IAM-Richtlinien im IAM-Benutzerhandbuch.

Richtlinie für Hauptbenutzer

Um alle erforderlichen Aktionen für EMR Serverless zu gewähren, erstellen Sie eine `AmazonEMRServerlessFullAccess` Richtlinie und fügen Sie sie dem erforderlichen IAM-Benutzer, der Rolle oder der Gruppe hinzu.

Im Folgenden finden Sie ein Beispiel für eine Richtlinie, mit der Hauptbenutzer serverlose EMR-Anwendungen erstellen und ändern sowie andere Aktionen wie das Senden und Debuggen von Aufträgen ausführen können. Es zeigt alle Aktionen, die EMR Serverless für andere Dienste benötigt.

JSON

```
{
  "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": [
        "*"
      ]
    }
  ]
}
```

Wenn Sie die Netzwerkkonnektivität zu Ihrer VPC aktivieren, erstellen EMR-Serverless-Anwendungen Amazon EC2 Elastic Network Interfaces (ENIs) für die Kommunikation mit VPC-Ressourcen. Die folgende Richtlinie stellt sicher, dass neue EC2 nur im Kontext von EMR Serverless-Anwendungen erstellt ENIs werden.

> **Note**
>
> Es wird dringend empfohlen, diese Richtlinie festzulegen, um sicherzustellen, dass Benutzer EC2 ENIs nur erstellen können, wenn sie EMR Serverless-Anwendungen starten.

JSON

```
{
  "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"
        }
      }
    }
  ]
}
```

Wenn Sie den serverlosen EMR-Zugriff auf bestimmte Subnetze einschränken möchten, können Sie jedes Subnetz mit einer Tag-Bedingung kennzeichnen. Diese IAM-Richtlinie stellt sicher, dass serverlose EMR-Anwendungen EC2 ENIs nur innerhalb zulässiger Subnetze erstellen können.

```
{
    "Sid": "AllowEC2ENICreationInSubnetAndSecurityGroupWithEMRTags",
    "Effect": "Allow",
    "Action": [
        "ec2:CreateNetworkInterface"
    ],
    "Resource": [
        "arn:aws:ec2:*:*:subnet/*",
        "arn:aws:ec2:*:*:security-group/*"
    ],
    "Condition": {
        "StringEquals": {
            "aws:ResourceTag/KEY": "VALUE"
        }
    }
}
```

Important

Wenn Sie ein Administrator oder Poweruser sind, der Ihre erste Anwendung erstellt, müssen Sie Ihre Berechtigungsrichtlinien so konfigurieren, dass Sie eine serviceverknüpfte Rolle mit EMR Serverless erstellen können. Weitere Informationen finden Sie unter. Verwenden von serviceverknüpften Rollen für EMR Serverless

Die folgende IAM-Richtlinie ermöglicht es Ihnen, eine dienstverknüpfte EMR-Serverless-Rolle für Ihr Konto zu erstellen.

```
{
   "Sid":"AllowEMRServerlessServiceLinkedRoleCreation",
   "Effect":"Allow",
   "Action":"iam:CreateServiceLinkedRole",
   "Resource":"arn:aws:iam::account-id:role/aws-service-role/ops.emr-
serverless.amazonaws.com/AWSServiceRoleForAmazonEMRServerless"
}
```

Richtlinie für Dateningenieure

Im Folgenden finden Sie eine Beispielrichtlinie, die Benutzern nur Leseberechtigungen für serverlose EMR-Anwendungen sowie die Möglichkeit bietet, Jobs zu senden und zu debuggen. Hinweis: Da diese Richtlinie Aktionen nicht ausdrücklich verweigert, kann dennoch eine andere Richtlinienanweisung verwendet werden, um den Zugriff auf bestimmte Aktionen zu gewähren.

JSON

```
{
  "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": [
        "*"
      ]
    }
  ]
}
```

Verwenden von Tags für die Zugriffskontrolle

Sie können Tag-Bedingungen für eine differenzierte Zugriffskontrolle verwenden. Sie können beispielsweise Benutzer aus einem Team einschränken, sodass sie nur Jobs an EMR Serverless-Anwendungen einreichen können, die mit ihrem Teamnamen gekennzeichnet sind.

JSON

```
{
  "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": [
        "*"
      ]
    }
  ]
}
```

Richtlinien für Tag-basierte Zugriffskontrolle

Sie können Bedingungen in Ihrer identitätsbasierten Richtlinie verwenden, um den Zugriff auf Anwendungen und Auftragsausführungen auf der Grundlage von Tags zu steuern.

Die folgenden Beispiele zeigen verschiedene Szenarien und Möglichkeiten zur Verwendung von Bedingungsoperatoren mit EMR Serverless Condition Keys. Diese IAM-Richtlinienanweisungen dienen nur zu Demonstrationszwecken und sollten nicht in Produktionsumgebungen verwendet werden. Es gibt mehrere Möglichkeiten für die Kombination von Richtlinienanweisungen zum Gewähren oder Verweigern von Berechtigungen entsprechend Ihren Anforderungen. Weitere Informationen zum Planen und Testen von IAM-Richtlinien finden Sie im IAM-Benutzerhandbuch.

> **Important**
>
> Das explizite Ablehnen von Berechtigungen für Markierungsaktionen stellt eine wichtige Überlegung dar. Dadurch wird verhindert, dass Benutzer eine Ressource markieren und

sich dadurch selbst Berechtigungen erteilen, die Sie nicht gewähren wollten. Wenn Tagging-Aktionen für eine Ressource nicht verweigert werden, kann ein Benutzer Tags ändern und so die Absicht der tagbasierten Richtlinien umgehen. Ein Beispiel für eine Richtlinie, die Tagging-Aktionen verweigert, finden Sie unter. Zugriff zum Hinzufügen und Entfernen von Tags verweigern

Die folgenden Beispiele zeigen identitätsbasierte Berechtigungsrichtlinien, mit denen die Aktionen gesteuert werden, die mit EMR Serverless-Anwendungen zulässig sind.

Erlauben Sie Aktionen nur für Ressourcen mit bestimmten Tag-Werten

Im folgenden Richtlinienbeispiel versucht der `StringEquals` Bedingungsoperator, eine Übereinstimmung `dev` mit dem Wert für das Tag department herzustellen. Wenn das Tag department der Anwendung nicht hinzugefügt wurde oder den Wert nicht enthält`dev`, gilt die Richtlinie nicht und die Aktionen sind nach dieser Richtlinie nicht zulässig. Wenn keine anderen Richtlinienerklärungen die Aktionen zulassen, kann der Benutzer nur mit Anwendungen arbeiten, die dieses Tag mit diesem Wert haben.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-serverless:GetApplication"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/department": "dev"
        }
      },
      "Sid": "AllowEMRSERVERLESSGetapplication"
    }
  ]
```

```
}
```

Sie können auch mehrere Tag-Werte mithilfe eines Bedingungsoperators angeben. Um beispielsweise Aktionen für Anwendungen zuzulassen, bei denen das `department` Tag den Wert enthält`test`, `dev` oder ersetzen Sie den Bedingungsblock aus dem vorherigen Beispiel durch den folgenden.

```
"Condition": {
        "StringEquals": {
          "emr-serverless:ResourceTag/department": ["dev", "test"]
        }
      }
```

Kennzeichnung bei der Erstellung einer Ressource vorschreiben

Im folgenden Beispiel muss das Tag bei der Erstellung der Anwendung angewendet werden.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-serverless:CreateApplication"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:RequestedRegion": "us-east-1"
        }
      },
      "Sid": "AllowEMRSERVERLESSCreateapplication"
    }
  ]
}
```

Die folgende Richtlinienanweisung ermöglicht es einem Benutzer, eine Anwendung nur zu erstellen, wenn die Anwendung über ein `department` Tag verfügt, das einen beliebigen Wert enthalten kann.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-serverless:CreateApplication"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:RequestedRegion": ["us-east-1", "us-west-2"]
        }
      },
      "Sid": "AllowEMRSERVERLESSCreateapplication"
    }
  ]
}
```

Zugriff zum Hinzufügen und Entfernen von Tags verweigern

Diese Richtlinie verhindert, dass ein Benutzer Tags zu EMR-Serverless-Anwendungen mit einem `department` Tag hinzufügt oder entfernt, dessen Wert nicht ist. `dev`

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "emr-serverless:TagResource",
```

```
        "emr-serverless:UntagResource"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringNotEquals": {
          "aws:PrincipalTag/department": "dev"
        }
      },
      "Sid": "AllowEMRSERVERLESSTagresource"
    }
  ]
}
```

Beispiele für identitätsbasierte Richtlinien für EMR Serverless

Standardmäßig sind Benutzer und Rollen nicht berechtigt, Amazon EMR Serverless-Ressourcen zu erstellen oder zu ändern. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern die Berechtigung erteilen, Aktionen für die Ressourcen auszuführen, die sie benötigen.

Informationen dazu, wie Sie unter Verwendung dieser beispielhaften JSON-Richtliniendokumente eine identitätsbasierte IAM-Richtlinie erstellen, finden Sie unter Erstellen von IAM-Richtlinien (Konsole) im IAM-Benutzerhandbuch.

Einzelheiten zu den von Amazon EMR Serverless definierten Aktionen und Ressourcentypen, einschließlich des Formats ARNs für die einzelnen Ressourcentypen, finden Sie unter Aktionen, Ressourcen und Bedingungsschlüssel für Amazon EMR Serverless in der Service Authorization Reference.

Themen

- Best Practices für Richtlinien
- Erlauben Sie Benutzern den Zugriff auf ihre eigenen Berechtigungen

Best Practices für Richtlinien

> **Note**
>
> EMR Serverless unterstützt keine verwalteten Richtlinien, sodass die erste im Folgenden aufgeführte Vorgehensweise nicht zutrifft.

Identitätsbasierte Richtlinien legen fest, ob jemand Amazon EMR Serverless-Ressourcen in Ihrem Konto erstellen, darauf zugreifen oder diese löschen kann. Dies kann zusätzliche Kosten für Ihr verursachen AWS-Konto. Beachten Sie beim Erstellen oder Bearbeiten identitätsbasierter Richtlinien die folgenden Richtlinien und Empfehlungen:

- Erste Schritte mit AWS verwalteten Richtlinien und Umstellung auf Berechtigungen mit den geringsten Rechten — Verwenden Sie die AWS verwalteten Richtlinien, die Berechtigungen für viele gängige Anwendungsfälle gewähren, um damit zu beginnen, Ihren Benutzern und Workloads Berechtigungen zu gewähren. Sie sind in Ihrem verfügbar. AWS-Konto Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie vom AWS Kunden verwaltete Richtlinien definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind. Weitere Informationen finden Sie unter Von AWS verwaltete Richtlinien oder Von AWS verwaltete Richtlinien für Auftragsfunktionen im IAM-Benutzerhandbuch.
- Anwendung von Berechtigungen mit den geringsten Rechten – Wenn Sie mit IAM-Richtlinien Berechtigungen festlegen, gewähren Sie nur die Berechtigungen, die für die Durchführung einer Aufgabe erforderlich sind. Sie tun dies, indem Sie die Aktionen definieren, die für bestimmte Ressourcen unter bestimmten Bedingungen durchgeführt werden können, auch bekannt als die geringsten Berechtigungen. Weitere Informationen zur Verwendung von IAM zum Anwenden von Berechtigungen finden Sie unter Richtlinien und Berechtigungen in IAM im IAM-Benutzerhandbuch.
- Verwenden von Bedingungen in IAM-Richtlinien zur weiteren Einschränkung des Zugriffs – Sie können Ihren Richtlinien eine Bedingung hinzufügen, um den Zugriff auf Aktionen und Ressourcen zu beschränken. Sie können beispielsweise eine Richtlinienbedingung schreiben, um festzulegen, dass alle Anforderungen mithilfe von SSL gesendet werden müssen. Sie können auch Bedingungen verwenden, um Zugriff auf Serviceaktionen zu gewähren, wenn diese für einen bestimmten Zweck verwendet werden AWS-Service, z. CloudFormation B. Weitere Informationen finden Sie unter IAM-JSON-Richtlinienelemente: Bedingung im IAM-Benutzerhandbuch.
- Verwenden von IAM Access Analyzer zur Validierung Ihrer IAM-Richtlinien, um sichere und funktionale Berechtigungen zu gewährleisten – IAM Access Analyzer validiert neue

und vorhandene Richtlinien, damit die Richtlinien der IAM-Richtliniensprache (JSON) und den bewährten IAM-Methoden entsprechen. IAM Access Analyzer stellt mehr als 100 Richtlinienprüfungen und umsetzbare Empfehlungen zur Verfügung, damit Sie sichere und funktionale Richtlinien erstellen können. Weitere Informationen finden Sie unter Richtlinienvalidierung mit IAM Access Analyzer im IAM-Benutzerhandbuch.

- Multi-Faktor-Authentifizierung (MFA) erforderlich — Wenn Sie ein Szenario haben, das IAM-Benutzer oder einen Root-Benutzer in Ihrem System erfordert AWS-Konto, aktivieren Sie MFA für zusätzliche Sicherheit. Um MFA beim Aufrufen von API-Vorgängen anzufordern, fügen Sie Ihren Richtlinien MFA-Bedingungen hinzu. Weitere Informationen finden Sie unter Sicherer API-Zugriff mit MFA im IAM-Benutzerhandbuch.

Weitere Informationen zu bewährten Methoden in IAM finden Sie unter Best Practices für die Sicherheit in IAM im IAM-Benutzerhandbuch.

Erlauben Sie Benutzern den Zugriff auf ihre eigenen Berechtigungen

In diesem Beispiel wird gezeigt, wie Sie eine Richtlinie erstellen, die IAM-Benutzern die Berechtigung zum Anzeigen der eingebundenen Richtlinien und verwalteten Richtlinien gewährt, die ihrer Benutzeridentität angefügt sind. Diese Richtlinie umfasst Berechtigungen zum Ausführen dieser Aktion auf der Konsole oder programmgesteuert mithilfe der AWS CLI AWS OR-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": "*"
        }
    ]
}
```

Serverlose Amazon EMR-Updates für verwaltete Richtlinien AWS

Greifen Sie auf Details zu Aktualisierungen der AWS verwalteten Richtlinien für Amazon EMR Serverless zu, seit dieser Service begonnen hat, diese Änderungen zu verfolgen. Um automatische Benachrichtigungen über Änderungen an dieser Seite zu erhalten, abonnieren Sie den RSS-Feed auf der Amazon EMR Serverless Document-Verlaufsseite.

Änderungen	Beschreibung	Date
Amazon EMRServerless ServiceRolePolicy — Aktualisierung einer bestehenden Richtlinie	Amazon EMR Serverless hat das neue `Sid` `CloudWatchPolicyStatement` und `EC2PolicyStatement` zur EMRServerlessServiceRolePolicy Amazon-Richtlinie hinzugefügt.	25. Januar 2024
Amazon EMRServerless ServiceRolePolicy — Aktualisierung einer bestehenden Richtlinie	Amazon EMR Serverless hat neue Berechtigungen hinzugefügt, damit Amazon EMR Serverless aggregierte	20. April 2023

Änderungen	Beschreibung	Date
	Kontometriken für die vCPU-Nutzung im Namespace veröffentlichen kann. `"AWS/Usage"`	
Amazon EMR Serverless hat mit der Nachverfolgung von Änderungen begonnen	Amazon EMR Serverless hat damit begonnen, Änderungen an seinen AWS verwalteten Richtlinien nachzuverfolgen.	20. April 2023

Fehlerbehebung für Amazon EMR Serverless Identity and Access

Verwenden Sie die folgenden Informationen, um allgemeine Probleme zu diagnostizieren und zu beheben, die bei der Arbeit mit Amazon EMR Serverless und IAM auftreten können.

Themen

- Ich bin nicht berechtigt, eine Aktion in Amazon EMR Serverless durchzuführen
- Ich bin nicht berechtigt, iam auszuführen: PassRole
- Ich möchte Personen außerhalb meines AWS Kontos den Zugriff auf meine Amazon EMR Serverless-Ressourcen ermöglichen
- Ich kann den UI/Spark Live-Verlaufsserver nicht von EMR Studio aus öffnen, um meinen Job zu debuggen, oder es tritt ein API-Fehler auf, wenn ich versuche, Protokolle abzurufen get-dashboard-for-job-run

Ich bin nicht berechtigt, eine Aktion in Amazon EMR Serverless durchzuführen

Wenn Ihnen AWS-Managementkonsole mitgeteilt wird, dass Sie nicht berechtigt sind, eine Aktion durchzuführen, wenden Sie sich an Ihren Administrator, um Unterstützung zu erhalten. Ihr Administrator ist die Person, die Ihnen Ihren Benutzernamen und Ihr Passwort zur Verfügung gestellt hat.

Der folgende Beispielfehler tritt auf, wenn der `mateojackson` Benutzer versucht, mit der Konsole auf Details zu einer fiktiven *`my-example-widget`* Ressource zuzugreifen, aber nicht über die fiktiven `emr-serverless:`*`GetWidget`* Berechtigungen verfügt.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform: emr-
serverless:GetWidget on resource: my-example-widget
```

In diesem Fall bittet Mateo seinen Administrator um die Aktualisierung seiner Richtlinien, um unter Verwendung der Aktion *my-example-widget* auf die Ressource `emr-serverless:`*GetWidget* zugreifen zu können.

Ich bin nicht berechtigt, iam auszuführen: PassRole

Wenn Sie eine Fehlermeldung erhalten, dass Sie nicht berechtigt sind, die `iam:PassRole` Aktion durchzuführen, müssen Ihre Richtlinien aktualisiert werden, damit Sie eine Rolle an Amazon EMR Serverless übergeben können.

Einige AWS-Services ermöglichen es Ihnen, eine bestehende Rolle an diesen Service zu übergeben, anstatt eine neue Servicerolle oder eine dienstbezogene Rolle zu erstellen. Hierzu benötigen Sie Berechtigungen für die Übergabe der Rolle an den Dienst.

Der folgende Beispielfehler tritt auf, wenn ein IAM-Benutzer mit dem Namen `marymajor` versucht, die Konsole zu verwenden, um eine Aktion in Amazon EMR Serverless auszuführen. Die Aktion erfordert jedoch, dass der Service über Berechtigungen verfügt, die durch eine Servicerolle gewährt werden. Mary besitzt keine Berechtigungen für die Übergabe der Rolle an den Dienst.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In diesem Fall müssen die Richtlinien von Mary aktualisiert werden, um die Aktion `iam:PassRole` ausführen zu können.

Wenn Sie Hilfe benötigen, wenden Sie sich an Ihren Administrator. AWS Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich möchte Personen außerhalb meines AWS Kontos den Zugriff auf meine Amazon EMR Serverless-Ressourcen ermöglichen

Sie können eine Rolle erstellen, mit der Benutzer in anderen Konten oder Personen außerhalb Ihrer Organisation auf Ihre Ressourcen zugreifen können. Sie können festlegen, wem die Übernahme der Rolle anvertraut wird. Für Dienste, die ressourcenbasierte Richtlinien oder Zugriffskontrolllisten (ACLs) unterstützen, können Sie diese Richtlinien verwenden, um Personen Zugriff auf Ihre Ressourcen zu gewähren.

Weitere Informationen dazu finden Sie hier:

- Informationen darüber, ob Amazon EMR Serverless diese Funktionen unterstützt, finden Sie unter. Identity and Access Management (IAM) in Amazon EMR Serverless
- Informationen dazu, wie Sie Zugriff auf Ihre Ressourcen gewähren können, AWS-Konten die Ihnen gehören, finden Sie im IAM-Benutzerhandbuch unter Gewähren des Zugriffs auf einen IAM-Benutzer in einem anderen AWS-Konto , den Sie besitzen.
- Informationen dazu, wie Sie Dritten Zugriff auf Ihre Ressourcen gewähren können AWS-Konten, finden Sie AWS-Konten im IAM-Benutzerhandbuch unter Gewähren des Zugriffs für Dritte.
- Informationen dazu, wie Sie über einen Identitätsverbund Zugriff gewähren, finden Sie unter Gewähren von Zugriff für extern authentifizierte Benutzer (Identitätsverbund) im IAM-Benutzerhandbuch.
- Informationen zum Unterschied zwischen der Verwendung von Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter Kontoübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.

Ich kann den UI/Spark Live-Verlaufsserver nicht von EMR Studio aus öffnen, um meinen Job zu debuggen, oder es tritt ein API-Fehler auf, wenn ich versuche, Protokolle abzurufen **`get-dashboard-for-job-run`**

Wenn Sie EMR Serverless Managed Storage für die Protokollierung verwenden und sich Ihre EMR Serverless-Anwendung in einem privaten Subnetz mit VPC-Endpunkten für Amazon S3 befindet und Sie eine Endpunktrichtlinie zur Zugriffskontrolle anhängen, fügen Sie die in Logging for EMR Serverless with managed storage genannten Berechtigungen in Ihrer VPC-Richtlinie zum S3-Gateway-Endpunkt für EMR Serverless hinzu, um Anwendungsprotokolle zu speichern und bereitzustellen.

Weitergabe von vertrauenswürdigen Identitäten

Mit den Amazon EMR-Versionen 7.8.0 und höher können Sie Benutzeridentitäten von AWS IAM Identity Center auf interaktive Workloads mit EMR Serverless über Apache Livy Endpoint übertragen. Interaktive Apache Livy-Workloads werden die bereitgestellte Identität weiter an nachgelagerte Dienste wie Amazon S3, Lake Formation und Amazon Redshift weitergeben und so einen sicheren Datenzugriff über Benutzeridentitäten in diesen nachgelagerten Umgebungen ermöglichen.
Die folgenden Abschnitte bieten einen konzeptionellen Überblick, die Voraussetzungen und die

erforderlichen Schritte, um Identität mit EMR Serverless über Apache Livy Endpoint zu starten und an interaktive Workloads weiterzugeben.

-Übersicht

IAM Identity Center ist der empfohlene Ansatz für die Authentifizierung und Autorisierung von Mitarbeitern AWS für Unternehmen aller Größen und Typen. Mit Identity Center können Sie Benutzeridentitäten in Ihrer vorhandenen Identitätsquelle AWS, einschließlich Microsoft Active Directory, Okta, Ping Identity, Google Workspace und Microsoft Entra ID (ehemals Azure AD) JumpCloud, erstellen und verwalten oder eine Verbindung zu Ihrer vorhandenen Identitätsquelle herstellen.

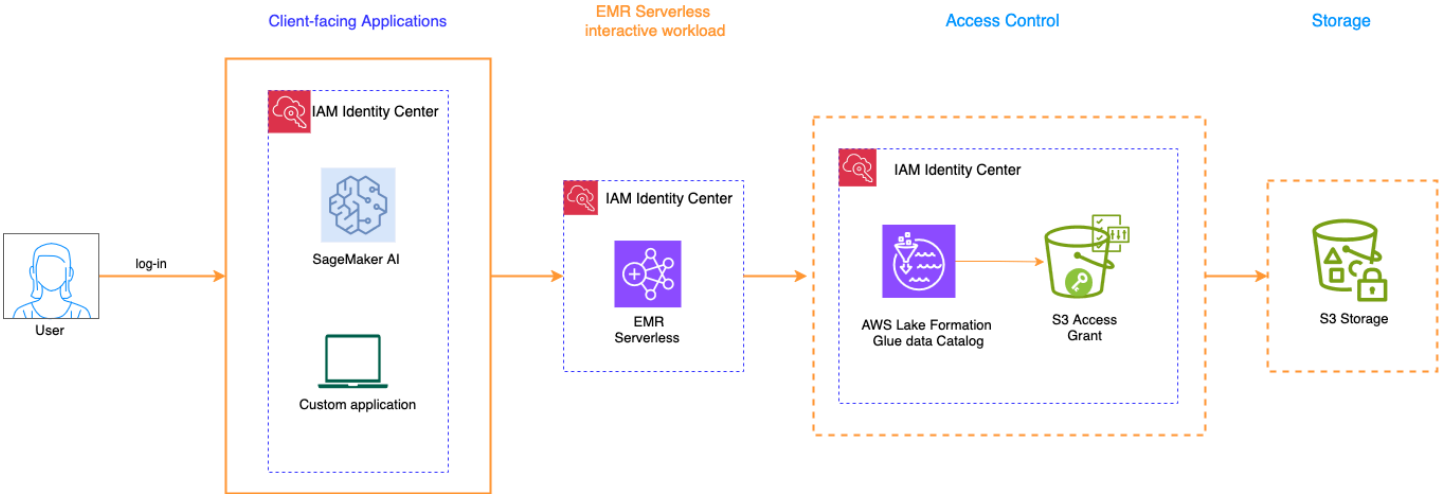
Die Verbreitung vertrauenswürdiger Identitäten ist eine AWS IAM Identity Center-Funktion, mit der Administratoren verbundener AWS Dienste Zugriff auf Servicedaten gewähren und prüfen können. Der Zugriff auf diese Daten basiert auf Benutzerattributen wie Gruppenzuordnungen. Die Einrichtung der Verbreitung vertrauenswürdiger Identitäten erfordert die Zusammenarbeit zwischen den Administratoren der verbundenen AWS Dienste und den IAM Identity Center-Administratoren. Weitere Informationen finden Sie unter Voraussetzungen und Überlegungen im IAM Identity Center-Benutzerhandbuch.

Features und Vorteile

Die Integration von EMR Serverless Apache Livy Endpoint mit IAM Identity Center Trusted Identity Propagation bietet die folgenden Vorteile:

- Die Möglichkeit, die Autorisierung auf Tabellenebene mit Identity Center-Identitäten in von AWS Lake Formation verwalteten AWS Glue-Datenkatalogtabellen durchzusetzen.
- Die Möglichkeit, die Autorisierung mit Identity-Center-Identitäten auf Amazon-Redshift-Clustern zu erzwingen.
- Die Möglichkeit der durchgängigen Nachverfolgung von Benutzeraktionen zu Prüfungszwecken.
- Die Möglichkeit, die Amazon-S3-Autorisierung auf Präfixebene mit Identity-Center-Identitäten auf von S3 Access Grants verwalteten S3-Präfixen durchzusetzen.

Funktionsweise



Beispiel für einen Anwendungsfall

Datenaufbereitung und Feature Engineering

Datenwissenschaftler aus mehreren Forschungsteams können dank einer einheitlichen Datenplattform an komplexen Projekten zusammenarbeiten. Sie melden sich mit ihren Unternehmensanmeldedaten bei SageMaker KI an und erhalten sofort Zugriff auf einen riesigen, gemeinsam genutzten Data Lake, der sich über mehrere Konten erstreckt. AWS Während sie mit der Feature-Entwicklung für neue Modelle des maschinellen Lernens beginnen, setzen die über EMR Serverless gestarteten Spark-Sitzungen die Sicherheitsrichtlinien von Lake Formation auf Spalten- und Zeilenebene auf der Grundlage ihrer propagierten Identitäten durch. Wissenschaftler können mit vertrauten Tools Daten effizient aufbereiten und Funktionen entwickeln, während die Compliance-Teams sicher sein können, dass jede Dateninteraktion automatisch verfolgt und geprüft wird. Diese sichere, kollaborative Umgebung beschleunigt die Forschungspipelines und hält gleichzeitig die strengen Datenschutzstandards aufrecht, die in regulierten Branchen erforderlich sind.

Erste Schritte mit Trusted-Identity Propagation

In diesem Abschnitt können Sie die serverlose EMR-Anwendung mit Apache Livy Endpoint konfigurieren, um sie in AWS IAM Identity Center zu integrieren und die Verbreitung vertrauenswürdiger Identitäten zu aktivieren.

Voraussetzungen

- Eine Identity Center-Instanz in der AWS Region, in der Sie EMR Serverless Apache Livy Endpoint mit Trusted Identity Propagation-fähig erstellen möchten. Eine Identity Center-Instanz kann

für ein AWS Konto nur in einer einzigen Region existieren. Weitere Informationen finden Sie unter Aktivieren von IAM Identity Center und Bereitstellen der Benutzer und Gruppen aus Ihrer Identitätsquelle für IAM Identity Center.

- Aktivieren Sie die Verbreitung vertrauenswürdiger Identitäten für nachgelagerte Dienste wie Lake Formation oder S3 Access Grants oder Amazon Redshift Redshift-Cluster, mit denen interaktive Workloads interagieren, um auf Daten zuzugreifen.

Berechtigungen zum Erstellen einer serverlosen EMR-Anwendung mit aktivierter Verbreitung vertrauenswürdiger Identitäten

Zusätzlich zu den grundlegenden Berechtigungen, die für den Zugriff auf EMR Serverless erforderlich sind, müssen Sie zusätzliche Berechtigungen für Ihre IAM-Identität oder Rolle konfigurieren, die verwendet wird, um EMR Serverless Application zu erstellen, die vertrauenswürdige Identitätsverbreitung aktiviert. Für die Verbreitung vertrauenswürdiger Identitäten ist EMR Serverless creates/bootstraps eine einzige dienstverwaltete Identity Center-Anwendung in Ihrem Konto, die der Dienst für die Identitätsvalidierung und die Identitätsweitergabe an nachgelagerte Bereiche nutzt.

```
"sso:DescribeInstance",
"sso:CreateApplication",
"sso:DeleteApplication",
"sso:PutApplicationAuthenticationMethod",
"sso:PutApplicationAssignmentConfiguration",
"sso:PutApplicationGrant",
"sso:PutApplicationAccessScope"
```

- `sso:DescribeInstance`— Erteilt die Berechtigung zur Beschreibung und Validierung der IAM Identity Center-Instanz, die Sie im Parameter angeben. identity-center-configuration
- `sso:CreateApplication`— Erteilt die Erlaubnis, eine von EMR Serverless verwaltete IAM Identity Center-Anwendung zu erstellen, die für Aktionen verwendet wird. trusted-identity-propatgion
- `sso:DeleteApplication`— erteilt die Erlaubnis, eine von EMR Serverless verwaltete IAM Identity Center-Anwendung zu bereinigen
- `sso:PutApplicationAuthenticationMethod`— Erteilt die Erlaubnis, AuthenticationMethod auf der serverlosen, verwalteten IAM Identity Center-Anwendung von EMR zu installieren, sodass der emr-serverlose Dienstprinzipal mit der IAM Identity Center-Anwendung interagieren kann.

- `sso:PutApplicationAssignmentConfiguration`— Erteilt die Berechtigung, die Einstellung „U“ in der IAM Identity Center-Anwendung festzulegen. ser-assignment-not-required
- `sso:PutApplicationGrant`— Erteilt die Erlaubnis, Token-Exchange-, IntrospectToken-, RefreshToken- und RevokeToken-Zuschüsse auf eine IAM Identity Center-Anwendung anzuwenden.
- `sso:PutApplicationAccessScope`— Erteilt die Erlaubnis, der IAM Identity Center-Anwendung den Downstream-Bereich mit aktivierter Weitergabe vertrauenswürdiger Identitäten zuzuweisen. Wir wenden die Bereiche „redshift:connect“, „lakeformation:query“ und „s3:read_write“ an, um diese Dienste zu aktivieren. trusted-identity-propagation

Erstellen Sie eine serverlose EMR-Anwendung mit aktivierter Verbreitung vertrauenswürdiger Identitäten

Sie müssen das `–identity-center-configuration` Feld mit angeben, um die Weitergabe vertrauenswürdiger Identitäten in der Anwendung `identityCenterInstanceArn` zu aktivieren. Verwenden Sie den folgenden Beispielbefehl, um eine serverlose EMR-Anwendung zu erstellen, für die die Weitergabe vertrauenswürdiger Identitäten aktiviert ist.

> **Note**
>
> Sie müssen außerdem angeben`--interactive-configuration '{"livyEndpointEnabled":true}'`, dass die Weitergabe vertrauenswürdiger Identitäten nur für Apache Livy Endpoint aktiviert ist.

```
aws emr-serverless create-application \
  --release-label emr-7.8.0 \
  --type "SPARK" \
  --identity-center-configuration '{"identityCenterInstanceArn" :
 "arn:aws:sso:::instance/ssoins-123456789"}' \
  --interactive-configuration '{"livyEndpointEnabled":true}'
```

- `identity-center-configuration`— (optional) Aktiviert die Weitergabe vertrauenswürdiger Identitäten durch Identity Center, falls angegeben.
- `identityCenterInstanceArn` – (erforderlich) Der ARN der Identity-Center-Instance.

Falls Sie nicht über die erforderlichen Identity Center-Berechtigungen (oben erwähnt) verfügen, erstellen Sie zunächst die EMR Serverless Application ohne Weitergabe vertrauenswürdiger Identitäten (geben Sie beispielsweise keinen `–identity-center-configuration` Parameter an) und bitten Sie später Ihren Identity Center-Administrator, die Weitergabe vertrauenswürdiger Identitäten zu aktivieren, indem Sie die Update-Application-API aufrufen, siehe Beispiel unten:

```
aws emr-serverless update-application \
  --application-id applicationId \
  --identity-center-configuration '{"identityCenterInstanceArn" :
 "arn:aws:sso:::instance/ssoins-123456789"}'
```

EMR Serverless erstellt in Ihrem Konto eine serviceverwaltete Identity Center-Anwendung, die der Service für Identitätsvalidierungen und die Weitergabe von Identitäten an nachgelagerte Dienste nutzt. Die von EMR Serverless erstellte verwaltete Identity Center-Anwendung wird von allen trusted-identity-propagation aktivierten EMR Serverless-Anwendungen in Ihrem Konto gemeinsam genutzt.

> **Note**
>
> Ändern Sie die Einstellungen der verwalteten Identity Center-Anwendung nicht manuell. Alle Änderungen können sich auf alle trusted-identity-propagation aktivierten EMR Serverless-Anwendungen in Ihrem Konto auswirken.

Berechtigungen der Jobausführungsrolle zur Weitergabe von Identität

Da EMR-Serverless identitätserweiterte job-execution-role Anmeldeinformationen nutzen, um Identität an nachgelagerte AWS Dienste weiterzugeben, muss die Vertrauensrichtlinie der Job Execution Role über zusätzliche Berechtigungen verfügen, um die Anmeldeinformationen für die Job Execution Role mit Identität `sts:SetContext` zu erweitern, um sie für nachgelagerte Dienste wie S3 Access-Grant, Lake Formation oder Amazon Redshift trusted-identity-propagation zu ermöglichen. Weitere Informationen zum Erstellen einer Rolle finden Sie unter Erstellen einer Job-Runtime-Rolle.

JSON

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

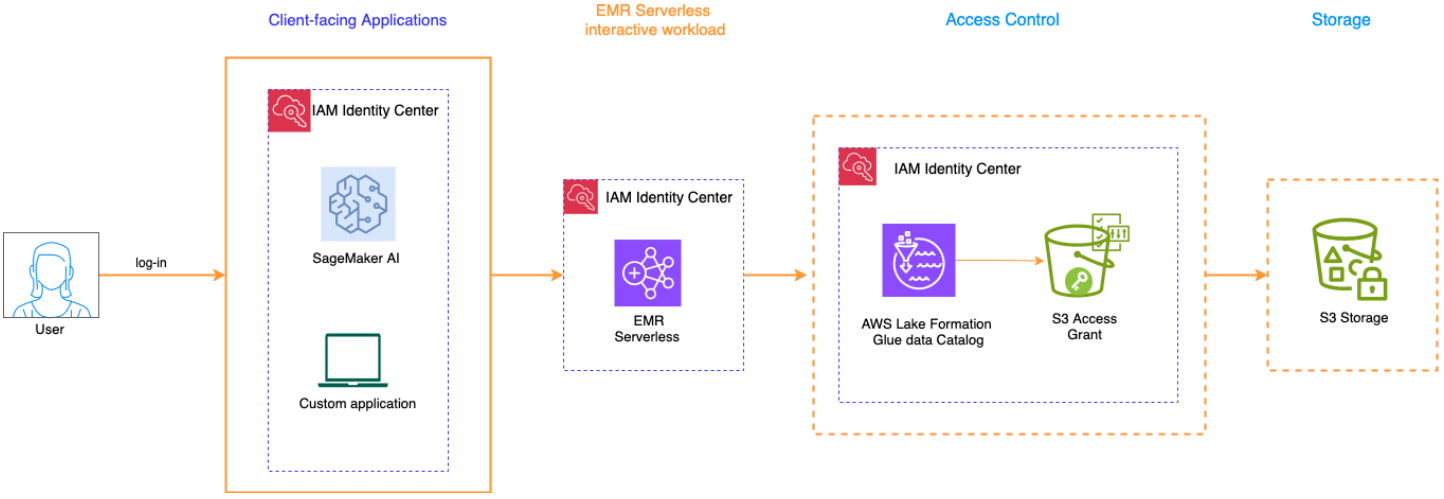
```
      "Effect": "Allow",
      "Principal": {
      "Service": "emr-serverless.amazonaws.com"
      },
      "Action": [ "sts:AssumeRole", "sts:SetContext"]
      }
  ]
}
```

Benötigt außerdem Berechtigungen für nachgelagerte AWS Dienste, JobExecutionRole die Job-Run aufrufen würde, um Daten mithilfe der Benutzeridentität abzurufen. Informationen zur Konfiguration von S3 Access Grant, Lake Formation finden Sie unter den folgenden Links.

- Verwenden von Lake Formation mit EMR Serverless
- Verwenden von Amazon S3 Access Grants mit EMR Serverless

Vertrauenswürdige Identitätsverbreitung für interaktive Workloads

Die Schritte zur Weitergabe von Identität an interaktive Workloads über einen Apache Livy-Endpunkt hängen davon ab, ob Ihre Benutzer mit einer AWS verwalteten Entwicklungsumgebung wie Amazon SageMaker AI oder mit Ihrer eigenen selbst gehosteten Notebook-Umgebung als clientseitige Anwendung interagieren.



AWS verwaltete Entwicklungsumgebung

Die folgende AWS verwaltete, clientseitige Anwendung unterstützt die Verbreitung vertrauenswürdiger Identitäten mit dem serverlosen EMR-Apache Livy-Endpunkt:

- Amazon SageMaker KI

Vom Kunden verwaltete, selbst gehostete Notebook-Umgebung

Informationen zur Aktivierung von Trusted Identity Propagation für Benutzer individuell entwickelter Anwendungen finden Sie unter Programmgesteuerter Zugriff auf AWS Services mithilfe von Trusted Identity Propagation im AWS Sicherheits-Blog.

Benutzersitzungen im Hintergrund

Mithilfe von Benutzerhintergrundsitzungen können Analysen und maschinelles Lernen auch dann fortgesetzt werden, wenn sich der Benutzer von seiner Notebook-Oberfläche abgemeldet hat. Diese Funktion wird durch die serverlose EMR-Integration mit der Trusted Identity Propagation-Funktion von IAM Identity Center implementiert. In diesem Abschnitt werden die Konfigurationsoptionen und das Verhalten für Benutzersitzungen im Hintergrund erläutert.

> **Note**
>
> Benutzerhintergrundsitzungen gelten für Spark-Workloads, die über Notebook-Schnittstellen wie Amazon SageMaker Unified Studio initiiert werden. Die Aktivierung oder Deaktivierung dieser Funktion wirkt sich nur auf neue Livy-Sitzungen aus; bestehende aktive Livy-Sitzungen sind davon nicht betroffen.

Konfigurieren Sie Hintergrundsitzungen für Benutzer

Benutzerhintergrundsitzungen müssen auf zwei Ebenen aktiviert werden, damit sie ordnungsgemäß funktionieren:

1. IAM Identity Center-Instanzebene — normalerweise von IdC-Administratoren konfiguriert
2. EMR Serverless-Anwendungsebene — konfiguriert von EMR Serverless-Anwendungsadministratoren

Benutzerhintergrundsitzungen für EMR Serverless-Anwendungen aktivieren

Um Benutzerhintergrundsitzungen für eine EMR Serverless-Anwendung zu `true` aktivieren, müssen Sie den `userBackgroundSessionsEnabled` Parameter `identityCenterConfiguration` beim Erstellen oder Aktualisieren einer Anwendung auf festlegen.

Voraussetzungen

- Ihre IAM-Rolle, die für die EMR Serverless-Anwendung verwendet wird, muss über create/update die entsprechende Berechtigung verfügen. `sso:PutApplicationSessionConfiguration` Diese Berechtigung ermöglicht EMR Serverless, Benutzerhintergrundsitzungen auf Ebene der verwalteten IDC-Anwendung von EMR Serverless zu aktivieren.
- Ihre EMR Serverless-Anwendung muss das Release-Label 7.8 oder höher verwenden und Trusted-Identity Propagation muss aktiviert sein.

Um Benutzerhintergrundsitzungen mit dem zu aktivieren AWS CLI

```
aws emr-serverless create-application \
    --name "my-analytics-app" \
    --type "SPARK" \
    --release-label "emr-7.8.0" \
    --identity-center-configuration '{"identityCenterInstanceArn":
 "arn:aws:sso:::instance/ssoins-1234567890abcdef", "userBackgroundSessionsEnabled":
 true}'
```

Um eine bestehende Anwendung zu aktualisieren:

```
aws emr-serverless update-application \
    --application-id applicationId \
    --identity-center-configuration '{"identityCenterInstanceArn":
 "arn:aws:sso:::instance/ssoins-1234567890abcdef", "userBackgroundSessionsEnabled":
 true}'
```

Konfigurationsmatrix

Die effektive Konfiguration der Benutzerhintergrundsitzung hängt sowohl von der Einstellung der EMR-Serverless-Anwendung als auch von den Einstellungen auf Instanzebene von IAM Identity Center ab:

Konfigurationsmatrix für Benutzerhintergrundsitzungen

IAM Identity Center aktiviert userBackgroundSession	EMR Serverless aktiviert userBackgroundSessions	Behavior
Ja	TRUE	Hintergrundsitzungen für Benutzer sind aktiviert

IAM Identity Center aktiviert userBackgroundSession	EMR Serverless aktiviert userBackgroundSessions	Behavior
Ja	FALSE	Die Sitzung läuft ab, wenn sich der Benutzer abmeldet
Nein	TRUE	Die Anwendung creation/ update schlägt mit einer Ausnahme fehl
Nein	FALSE	Die Sitzung läuft mit der Benutzerabmeldung ab

Standarddauer für Benutzersitzungen im Hintergrund

Standardmäßig haben alle Benutzer-Hintergrundsitzungen in IAM Identity Center ein Zeitlimit von 7 Tagen. Administratoren können diese Dauer in der Konsole von IAM Identity Center ändern. Diese Einstellung gilt für die IAM Identity Center-Instanzebene und wirkt sich auf alle unterstützten IAM Identity Center-Anwendungen innerhalb dieser Instanz aus.

- Die Dauer kann auf einen beliebigen Wert zwischen 15 Minuten und 90 Tagen festgelegt werden.
- Diese Einstellung wird in der IAM Identity Center-Konsole unter Einstellungen → Authentifizierung → Konfigurieren (Abschnitt „Nicht interaktive Jobs“) konfiguriert

> **Note**
>
> Serverlose EMR-Livy-Sitzungen haben eine separate Höchstdauer von 24 Stunden. Sitzungen werden beendet, wenn entweder das Livy-Sitzungslimit oder die Dauer der Benutzerhintergrundsitzung erreicht ist, je nachdem, was zuerst eintritt.

Auswirkungen der Deaktivierung von Benutzerhintergrundsitzungen

Wenn Benutzerhintergrundsitzungen in IAM Identity Center deaktiviert sind:

Bestehende Livy-Sitzungen

Läuft ohne Unterbrechung weiter, wenn sie mit aktivierten Benutzerhintergrundsitzungen gestartet wurden. Diese Sitzungen verwenden weiterhin ihre vorhandenen Sitzungs-Tokens im Hintergrund, bis sie automatisch beendet werden oder explizit beendet werden.

Neue Livy-Sitzungen

Verwendet den standardmäßigen Übertragungsablauf für vertrauenswürdige Identitäten und wird beendet, wenn sich der Benutzer abmeldet oder seine interaktive Sitzung abläuft (z. B. beim Schließen eines Amazon SageMaker Unified JupyterLab Studio-Notebooks).

Änderung der Dauer von Benutzersitzungen im Hintergrund

Wenn die Einstellung für die Dauer von Benutzerhintergrundsitzungen in IAM Identity Center geändert wird:

Bestehende Livy-Sitzungen

Läuft mit derselben Dauer der Hintergrundsitzung weiter, mit der sie gestartet wurden.

Neue Livy-Sitzungen

Wird die neue Sitzungsdauer für Hintergrundsitzungen verwenden.

Überlegungen

Bedingungen für die Beendigung der Sitzung

Wenn Sie Benutzersitzungen im Hintergrund verwenden, läuft eine Livy-Sitzung so lange weiter, bis einer der folgenden Fälle eintritt:

- Die Benutzerhintergrundsitzung läuft ab (je nach IdC-Konfiguration, bis zu 90 Tage)
- die Benutzersitzung im Hintergrund manuell von einem Administrator gesperrt wird
- Die Livy-Sitzung erreicht ihr Leerlauf-Timeout (Standard: 1 Stunde nach der letzten ausgeführten Anweisung)
- Die Livy-Sitzung erreicht ihre maximale Dauer (24 Stunden)
- Der Benutzer stoppt den Notebook-Kernel explizit oder startet ihn neu

Persistenz von Daten

Bei der Verwendung von Benutzerhintergrundsitzungen:

- Benutzer können sich nach dem Abmelden nicht mehr mit ihrer Notebook-Oberfläche verbinden, um die Ergebnisse anzusehen
- Konfigurieren Sie Ihre Spark-Anweisungen so, dass Ergebnisse vor Abschluss der Ausführung in einen persistenten Speicher (z. B. Amazon S3) geschrieben werden

Auswirkungen auf die Kosten

- Jobs werden auch dann bis zum Abschluss ausgeführt, wenn Benutzer ihre Amazon SageMaker Unified JupyterLab Studio-Sitzung beenden, und es fallen Gebühren für die gesamte Dauer des abgeschlossenen Laufs an.
- Überwachen Sie Ihre aktiven Hintergrundsitzungen, um unnötige Kosten durch vergessene oder abgebrochene Sitzungen zu vermeiden.

Verfügbarkeit von Funktionen

Benutzer-Hintergrundsitzungen für EMR Serverless sind verfügbar für:

- Nur Spark-Engine (Hive-Engine wird nicht unterstützt)
- Nur interaktive Livy-Sitzungen (Batch-Jobs und Streaming-Jobs werden nicht unterstützt)
- EMR Serverless Release-Labels 7.8 und höher

Überlegungen zur serverlosen EMR-Integration Trusted-Identity-Propagation

Beachten Sie Folgendes, wenn Sie IAM Identity Center Trusted-Identity-Propagation mit einer serverlosen EMR-Anwendung verwenden:

- Trusted Identity Propagation über Identity Center wird auf Amazon EMR 7.8.0 und höher und nur mit Apache Spark unterstützt.
- Trusted Identity Propagation kann nur für interaktive Workloads mit EMR Serverless über einen Apache Livy-Endpunkt verwendet werden. Interaktive Workloads über EMR Studio unterstützen Trusted Identity Propagation nicht

- Batch-Jobs und Streaming-Workloads unterstützen keine vertrauenswürdige Identitätsverbreitung
- Detaillierte Zugriffskontrollen mithilfe von AWS Lake Formation, die Trusted Identity Propagation verwenden, sind für interaktive Workloads mit EMR Serverless über einen Apache Livy-Endpunkt verfügbar.
- Trusted Identity Propagation mit Amazon EMR wird in den folgenden AWS Regionen unterstützt:
  - af-south-1 – Afrika (Kapstadt)
  - ap-east-1 – Asien-Pazifik (Hongkong)
  - ap-northeast-1 – Asien-Pazifik (Tokio)
  - ap-northeast-2 – Asien-Pazifik (Seoul)
  - ap-northeast-3 – Asien-Pazifik (Osaka)
  - ap-south-1 – Asien-Pazifik (Mumbai)
  - ap-southeast-1 – Asien-Pazifik (Singapur)
  - ap-southeast-2 – Asien-Pazifik (Sydney)
  - ap-southeast-3 – Asien-Pazifik (Jakarta)
  - ca-central-1 – Kanada (Zentral)
  - ca-west-1 — Kanada (Calgary)
  - eu-central-1 – EU (Frankfurt)
  - eu-north-1 – Europa (Stockholm)
  - eu-south-1 – Europa (Mailand)
  - eu-south-2 — Europa (Spanien)
  - eu-west-1 – EU (Irland)
  - eu-west-2 – Europa (London)
  - eu-west-3 – Europa (Paris)
  - me-central-1 — Naher Osten (VAE)
  - me-south-1 – Naher Osten (Bahrain)
  - sa-east-1 – Südamerika (São Paulo)
  - us-east-1 – USA Ost (Nord-Virginia)
  - us-east-2 – USA Ost (Ohio)
  - us-west-1 – US West (Nordkalifornien)
  - us-west-2 – USA West (Oregon)

Verwenden von Lake Formation mit EMR Serverless

Sie können serverlose EMR-Anwendungen so konfigurieren, dass sie Lake Formation entweder mit vollständigem Tabellenzugriff oder detaillierter Zugriffskontrolle verwenden. Einzelheiten zu den unterstützten Funktionen in den einzelnen Zugriffsmodi finden Sie in der folgenden Tabelle.

Verfügbarkeit von Features

Feature	Verfügbar bei
Leseoperationen (SELECT, DESCRIBE) für Hive- und Iceberg-Tabellen	EMR 7,2 +
Ansichten mit mehreren Dialekten	EMR 7,6+
Leseoperationen (SELECT, DESCRIBE) für Delta Lake- und Hudi-Tabellen	EMR 7,6+
Vollständiger Tabellenzugriff für Hive, Iceberg	EMR 7,9+
Vollständiger Tabellenzugriff für Delta Lake	EMR 7.11+
Schreiboperationen (DDL, DML) für Hive-, Iceberg- und Delta Lake-Tabellen	EMR 7.12+
Vollständiger Tabellenzugriff für Hudi	EMR 7.12+

Lake Formation Vollzugriff auf Tabellen für EMR Serverless

Mit den Amazon EMR-Versionen 7.8.0 und höher können Sie AWS Lake Formation mit Glue Data Catalog nutzen, wobei die Job-Runtime-Rolle über vollständige Tabellenberechtigungen verfügt, ohne die Einschränkungen einer detaillierten Zugriffskontrolle. Diese Funktion ermöglicht es Ihnen, von Ihren EMR Serverless Spark-Batch- und interaktiven Jobs aus zu lesen und in Tabellen zu schreiben, die durch Lake Formation geschützt sind. In den folgenden Abschnitten erfahren Sie mehr über Lake Formation und dessen Verwendung mit EMR Serverless.

Lake Formation mit vollständigem Tabellenzugriff verwenden

Sie können auf AWS Lake Formation Formation-geschützte Glue Data-Katalogtabellen von EMR Serverless Spark-Jobs oder interaktiven Sitzungen aus zugreifen, in denen die Runtime-Rolle des Jobs vollen Tabellenzugriff hat. Sie müssen AWS Lake Formation in der EMR Serverless-Anwendung nicht aktivieren. Wenn ein Spark-Job für Full Table Access (FTA) konfiguriert ist, werden AWS Lake Formation-Anmeldeinformationen für read/write S3-Daten für in AWS Lake Formation registrierte Tabellen verwendet, während die Anmeldeinformationen für die Laufzeitrolle des Jobs für read/write Tabellen verwendet werden, die nicht bei AWS Lake Formation registriert sind.

> **Important**
>
> Aktivieren Sie AWS Lake Formation nicht für eine detaillierte Zugriffskontrolle. Ein Job kann nicht gleichzeitig Full Table Access (FTA) und Fine-Grained Access Control (FGAC) auf demselben EMR-Cluster oder derselben EMR-Applikation ausführen.

Schritt 1: Vollständigen Tabellenzugriff in Lake Formation aktivieren

Um den Modus Full Table Access (FTA) zu verwenden, müssen Sie Drittanbieter-Abfrage-Engines den Zugriff auf Daten ohne die IAM-Sitzungs-Tag-Validierung in AWS Lake Formation gestatten. Folgen Sie zur Aktivierung den Schritten unter Application integration for full table access.

> **Note**
>
> Beim Zugriff auf kontenübergreifende Tabellen muss der Zugriff auf vollständige Tabellen sowohl in Producer- als auch in Consumer-Konten aktiviert sein. Ebenso muss diese Einstellung beim Zugriff auf regionsübergreifende Tabellen sowohl in Erzeuger- als auch in Verbraucherregionen aktiviert sein.

Schritt 2: Einrichten der IAM-Berechtigungen für die Auftrag-Laufzeitrolle

Für den Lese- oder Schreibzugriff auf zugrunde liegende Daten benötigt eine Job-Runtime-Rolle zusätzlich zu den Lake Formation Formation-Berechtigungen die `lakeformation:GetDataAccess` IAM-Berechtigung. Mit dieser Berechtigung gewährt Lake Formation die Anforderung von temporären Anmeldeinformationen für den Zugriff auf die Daten.

Im Folgenden finden Sie eine Beispielrichtlinie für die Bereitstellung von IAM-Berechtigungen für den Zugriff auf ein Skript in Amazon S3, für das Hochladen von Protokollen auf S3, für AWS Glue-API-Berechtigungen und für den Zugriff auf Lake Formation.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "ScriptAccess",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::*.amzn-s3-demo-bucket/scripts"
      ]
    },
    {
      "Sid": "LoggingAccess",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/logs/*"
      ]
    },
    {
      "Sid": "GlueCatalogAccess",
      "Effect": "Allow",
      "Action": [
        "glue:Get*",
        "glue:Create*",
        "glue:Update*"
      ],
      "Resource": [
        "*"
      ]
    },
    {
```

```
      "Sid": "LakeFormationAccess",
      "Effect": "Allow",
      "Action": [
        "lakeformation:GetDataAccess"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

Schritt 2.1 Lake Formation Formation-Berechtigungen konfigurieren

- Spark-Jobs, die Daten aus S3 lesen, benötigen die Lake Formation SELECT-Berechtigung.
- Spark-Jobs, für die write/delete Daten in S3 ist die Lake Formation ALL (SUPER) -Genehmigung erforderlich.
- Spark-Jobs, die mit dem Glue-Datenkatalog interagieren, benötigen die entsprechenden DESCRIBE-, ALTER- und DROP-Berechtigungen.

Weitere Informationen finden Sie unter Erteilen von Berechtigungen für Datenkatalogressourcen.

Schritt 3: Initialisieren Sie eine Spark-Sitzung für den vollständigen Tabellenzugriff mit Lake Formation

Voraussetzungen

AWS Der Glue Data Catalog muss als Metastore konfiguriert werden, um auf Lake Formation-Tabellen zugreifen zu können.

Legen Sie die folgenden Einstellungen fest, um den Glue-Katalog als Metastore zu konfigurieren:

```
--conf spark.sql.catalogImplementation=hive
--conf
 spark.hive.metastore.client.factory.class=com.amazonaws.glue.catalog.metastore.AWSGlueDataCata
```

Weitere Informationen zur Aktivierung von Data Catalog for EMR Serverless finden Sie unter Metastore-Konfiguration für EMR Serverless.

Um auf Tabellen zuzugreifen, die bei AWS Lake Formation registriert sind, müssen die folgenden Konfigurationen während der Spark-Initialisierung festgelegt werden, um Spark für die Verwendung von AWS Lake Formation Formation-Anmeldeinformationen zu konfigurieren.

Hive

```
--conf
 spark.hadoop.fs.s3.credentialsResolverClass=com.amazonaws.glue.accesscontrol.AWSLakeFormati
--conf spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject=true
--conf spark.hadoop.fs.s3.folderObject.autoAction.disabled=true
--conf spark.sql.catalog.skipLocationValidationOnCreateTable.enabled=true
--conf spark.sql.catalog.createDirectoryAfterTable.enabled=true
--conf spark.sql.catalog.dropDirectoryBeforeTable.enabled=true
```

Iceberg

```
--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.client.region=REGION
--conf spark.sql.catalog.spark_catalog.type=glue
--conf spark.sql.catalog.spark_catalog.glue.account-id=ACCOUNT_ID
--conf spark.sql.catalog.spark_catalog.glue.lakeformation-enabled=true
--conf spark.sql.catalog.dropDirectoryBeforeTable.enabled=true
```

Delta Lake

```
--conf
 spark.hadoop.fs.s3.credentialsResolverClass=com.amazonaws.glue.accesscontrol.AWSLakeFormati
--conf spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject=true
--conf spark.hadoop.fs.s3.folderObject.autoAction.disabled=true
--conf spark.sql.catalog.skipLocationValidationOnCreateTable.enabled=true
--conf spark.sql.catalog.createDirectoryAfterTable.enabled=true
--conf spark.sql.catalog.dropDirectoryBeforeTable.enabled=true
```

Hudi

```
--conf
 spark.hadoop.fs.s3.credentialsResolverClass=com.amazonaws.glue.accesscontrol.AWSLakeFormati
--conf spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject=true
--conf spark.hadoop.fs.s3.folderObject.autoAction.disabled=true
--conf spark.sql.catalog.skipLocationValidationOnCreateTable.enabled=true
```

```
--conf spark.sql.catalog.createDirectoryAfterTable.enabled=true
--conf spark.sql.catalog.dropDirectoryBeforeTable.enabled=true
--conf spark.jars=/usr/lib/hudi/hudi-spark-bundle.jar
--conf spark.sql.extensions=org.apache.spark.sql.hudi.HoodieSparkSessionExtension
--conf
 spark.sql.catalog.spark_catalog=org.apache.spark.sql.hudi.catalog.HoodieCatalog
--conf spark.serializer=org.apache.spark.serializer.KryoSerializer
```

- `spark.hadoop.fs.s3.credentialsResolverClass=com.amazonaws.glue.accesscontrol.AW` Konfigurieren Sie EMR Filesystem (EMRFS) oder EMR S3A so, dass Lake Formation S3-Anmeldeinformationen für registrierte AWS Lake Formation-Tabellen verwendet werden. Wenn die Tabelle nicht registriert ist, verwenden Sie die Anmeldeinformationen für die Auftrag-Laufzeitrolle.
- `spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject=true` und `spark.hadoop.fs.s3.folderObject.autoAction.disabled=true`: Konfigurieren Sie EMRFS so, dass beim Erstellen von S3-Ordnern der Content-Type-Header „application/x-directory“ anstelle des Suffixes „$folder$“ verwendet wird. Dies ist beim Lesen von Lake Formation-Tabellen erforderlich, da die Anmeldeinformationen von Lake Formation das Lesen von Tabellenordnern mit dem Suffix $folder$ nicht zulassen.
- `spark.sql.catalog.skipLocationValidationOnCreateTable.enabled=true`: Konfigurieren Sie Spark so, dass die Überprüfung, ob der Tabellenspeicherort leer ist, vor der Erstellung übersprungen wird. Dies ist für in Lake Formation registrierte Tabellen erforderlich, da die Anmeldeinformationen für Lake Formation zur Überprüfung des leeren Speicherorts erst nach der Erstellung der Glue Data Catalog-Tabelle verfügbar sind. Ohne diese Konfiguration validieren die Anmeldeinformationen für die Auftrag-Laufzeitrolle den Speicherort der leeren Tabelle.
- `spark.sql.catalog.createDirectoryAfterTable.enabled=true`: Konfigurieren Sie Spark so, dass der Amazon-S3-Ordner nach der Tabellenerstellung im Hive-Metastore erstellt wird. Dies ist für in Lake Formation registrierte Tabellen erforderlich, da die Anmeldeinformationen für Lake Formation zum Erstellen des S3-Ordners erst nach der Erstellung der Glue Data Catalog-Tabelle verfügbar sind.
- `spark.sql.catalog.dropDirectoryBeforeTable.enabled=true`: Konfigurieren Sie Spark so, dass der S3-Ordner vor dem Löschen der Tabelle im Hive-Metastore gelöscht wird. Dies ist für in Lake Formation registrierte Tabellen erforderlich, da die Lake Formation Formation-Anmeldeinformationen zum Löschen des S3-Ordners nach dem Löschen der Tabelle aus dem Glue-Datenkatalog nicht verfügbar sind.
- `spark.sql.catalog.<catalog>.glue.lakeformation-enabled=true`: Konfigurieren Sie den Iceberg-Katalog so, dass er AWS Lake Formation S3-Anmeldeinformationen für registrierte

Lake Formation-Tabellen verwendet. Wenn die Tabelle nicht registriert ist, verwenden Sie die Standardanmeldeinformationen für die Umgebung.

Konfigurieren Sie den vollständigen Tabellenzugriffsmodus in SageMaker Unified Studio

Verwenden Sie den Kompatibilitätsberechtigungsmodus, um über interaktive Spark-Sitzungen in JupyterLab Notebooks auf registrierte Tabellen von Lake Formation zuzugreifen. Verwenden Sie den magischen Befehl %%configure, um Ihre Spark-Konfiguration einzurichten. Wählen Sie die Konfiguration basierend auf Ihrem Tabellentyp aus:

For Hive tables

```
%%configure -f
{
    "conf": {
        "spark.hadoop.fs.s3.credentialsResolverClass":
 "com.amazonaws.glue.accesscontrol.AWSLakeFormationCredentialResolver",
        "spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject": true,
        "spark.hadoop.fs.s3.folderObject.autoAction.disabled": true,
        "spark.sql.catalog.skipLocationValidationOnCreateTable.enabled": true,
        "spark.sql.catalog.createDirectoryAfterTable.enabled": true,
        "spark.sql.catalog.dropDirectoryBeforeTable.enabled": true
    }
}
```

For Iceberg tables

```
%%configure -f
{
    "conf": {
        "spark.sql.catalog.spark_catalog":
 "org.apache.iceberg.spark.SparkSessionCatalog",
        "spark.sql.catalog.spark_catalog.warehouse": "S3_DATA_LOCATION",
        "spark.sql.catalog.spark_catalog.client.region": "REGION",
        "spark.sql.catalog.spark_catalog.type": "glue",
        "spark.sql.catalog.spark_catalog.glue.account-id": "ACCOUNT_ID",
        "spark.sql.catalog.spark_catalog.glue.lakeformation-enabled": "true",
        "spark.sql.catalog.dropDirectoryBeforeTable.enabled": "true",
    }
}
```

For Delta Lake tables

```
%%configure -f
{
    "conf": {
        "spark.hadoop.fs.s3.credentialsResolverClass":
 "com.amazonaws.glue.accesscontrol.AWSLakeFormationCredentialResolver",
        "spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject": true,
        "spark.hadoop.fs.s3.folderObject.autoAction.disabled": true,
        "spark.sql.catalog.skipLocationValidationOnCreateTable.enabled": true,
        "spark.sql.catalog.createDirectoryAfterTable.enabled": true,
        "spark.sql.catalog.dropDirectoryBeforeTable.enabled": true
    }
}
```

For Hudi tables

```
%%configure -f
{
    "conf": {
        "spark.hadoop.fs.s3.credentialsResolverClass":
 "com.amazonaws.glue.accesscontrol.AWSLakeFormationCredentialResolver",
        "spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject": true,
        "spark.hadoop.fs.s3.folderObject.autoAction.disabled": true,
        "spark.sql.catalog.skipLocationValidationOnCreateTable.enabled": true,
        "spark.sql.catalog.createDirectoryAfterTable.enabled": true,
        "spark.sql.catalog.dropDirectoryBeforeTable.enabled": true,
        "spark.jars": "/usr/lib/hudi/hudi-spark-bundle.jar",
        "spark.sql.extensions":
 "org.apache.spark.sql.hudi.HoodieSparkSessionExtension",
        "spark.sql.catalog.spark_catalog":
 "org.apache.spark.sql.hudi.catalog.HoodieCatalog",
        "spark.serializer": "org.apache.spark.serializer.KryoSerializer"
    }
}
```

Ersetzen Sie die Platzhalter:

- `S3_DATA_LOCATION`: Ihr S3-Bucket-Pfad
- `REGION`: AWS Region (z. B. us-east-1)
- `ACCOUNT_ID`: Ihre Konto-ID AWS

> **Note**
>
> Sie müssen diese Konfigurationen festlegen, bevor Sie Spark-Operationen in Ihrem Notebook ausführen.

Unterstützte Vorgänge

Bei diesen Vorgängen werden AWS Lake Formation Formation-Anmeldeinformationen für den Zugriff auf die Tabellendaten verwendet.

- CREATE TABLE
- ALTER TABLE
- INSERT INTO
- INSERT OVERWRITE
- UPDATE
- MERGE INTO
- DELETE FROM
- ANALYZE TABLE
- REPAIR TABLE
- DROP TABLE
- Spark-Datenquellenabfragen
- Spark-Datenquellenschreibvorgänge

> **Note**
>
> Operationen, die oben nicht aufgeführt sind, verwenden weiterhin IAM-Berechtigungen für den Zugriff auf Tabellendaten.

Überlegungen

- Wenn eine Hive-Tabelle mit einem Job erstellt wird, für den der vollständige Tabellenzugriff nicht aktiviert ist, und keine Datensätze eingefügt werden, schlagen nachfolgende Lese- oder Schreibvorgänge aus einem Job mit vollständigem Tabellenzugriff fehl. Dies liegt daran, dass EMR

Spark ohne vollständigen Tabellenzugriff das `$folder$` Suffix zum Namen des Tabellenordners hinzufügt. Um dieses Problem zu lösen, haben Sie folgende Möglichkeiten:

- Fügen Sie mindestens eine Zeile aus einem Auftrag in die Tabelle ein, für den FTA nicht aktiviert ist.
- Konfigurieren Sie den Job, für den FTA nicht aktiviert ist, so, dass er in S3 kein `$folder$` Suffix im Ordnernamen verwendet. Dies kann durch Einstellen der Spark-Konfiguration `spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject=true` erreicht werden.
- Erstellen Sie mit der S3-Konsole oder AWS der S3-CLI einen AWS S3-Ordner am Speicherort `s3://path/to/table/table_name` der Tabelle.

- Vollständiger Tabellenzugriff wird mit dem EMR-Dateisystem (EMRFS) ab Amazon EMR-Version 7.8.0 und mit dem S3A-Dateisystem ab Amazon EMR-Version 7.10.0 unterstützt.
- Full Table Access wird für Hive-, Iceberg-, Delta- und Hudi-Tabellen unterstützt.
- Überlegungen zum Hudi FTA Write Support:
  - Hudi FTA Writes müssen HoodieCredentialedHadoopStorage für den Verkauf von Anmeldeinformationen während der Auftragsausführung verwendet werden. Stellen Sie die folgende Konfiguration ein, wenn Sie Hudi-Jobs ausführen: `hoodie.storage.class=org.apache.spark.sql.hudi.storage.HoodieCredentialedHadc`
  - Full Table Access (FTA) -Schreibunterstützung für Hudi ist ab Amazon EMR Version 7.12 verfügbar.
  - Die Hudi FTA-Schreibunterstützung funktioniert derzeit nur mit den Hudi-Standardkonfigurationen. Benutzerdefinierte oder nicht standardmäßige Hudi-Einstellungen werden möglicherweise nicht vollständig unterstützt und können zu unerwartetem Verhalten führen.
  - Clustering für Hudi-Tabellen Merge-On-Read (MOR) wird derzeit im FTA-Schreibmodus nicht unterstützt.
- Jobs, die auf Tabellen mit Lake Formation Fine-Grained Access Control (FGAC) -Regeln oder Glue Data Catalog Views verweisen, schlagen fehl. Um eine Tabelle mit FGAC-Regeln oder einer Glue-Datenkatalogsicht abzufragen, müssen Sie den FGAC-Modus verwenden. Sie können den FGAC-Modus aktivieren, indem Sie die in der AWS Dokumentation beschriebenen Schritte ausführen: Verwenden von EMR Serverless with AWS Lake Formation für eine detaillierte Zugriffskontrolle.
- Der vollständige Tabellenzugriff unterstützt Spark Streaming nicht.
- Beim Schreiben von Spark DataFrame in eine Lake Formation-Tabelle wird nur der APPEND-Modus für Hive- und Iceberg-Tabellen unterstützt: `df.write.mode("append").saveAsTable(table_name)`

- Für das Erstellen externer Tabellen sind IAM-Berechtigungen erforderlich.
- Da Lake Formation Anmeldeinformationen vorübergehend innerhalb eines Spark-Jobs zwischenspeichert, spiegelt ein Spark-Batchjob oder eine interaktive Sitzung, die gerade ausgeführt wird, möglicherweise keine Berechtigungsänderungen wider.
- Sie müssen eine benutzerdefinierte Rolle und keine dienstverknüpfte Rolle verwenden: Lake Formation Formation-Anforderungen für Rollen.

Hudi FTA Write Support - Unterstützte Operationen

Die folgende Tabelle zeigt die unterstützten Schreibvorgänge für Hudi- Copy-On-Write (COW) - und Merge-On-Read (MOR) -Tabellen im Modus „Vollständiger Tabellenzugriff“:

Von Hudi FTA unterstützte Schreibvorgänge

Tabellentyp	Operation	SQL Write-Befehl	Status
KUH	INSERT	INSERT INTO TABLE	Unterstützt
KUH	INSERT	IN TABELLE EINFÜGEN — PARTITION (statisch, dynamisch)	Unterstützt
KUH	INSERT	INSERT OVERWRITE	Unterstützt
KUH	INSERT	INSERT OVERWRITE - PARTITION (statisch, dynamisch)	Unterstützt
UPDATE	UPDATE	UPDATE TABLE	Unterstützt
KUH	UPDATE	TABELLE AKTUALISI	Nicht unterstützt

Tabellentyp	Operation	SQL Write-Befehl	Status
		EREN — Partition ändern	
DELETE	DELETE	DELETE FROM TABLE	Unterstützt
ALTER	ALTER	TABELLE ÄNDERN - UMBENENNEN IN	Nicht unterstützt
KUH	ALTER	TABELLE ÄNDERN - TABELLENEIGENSCHAFTEN FESTLEGEN	Unterstützt
KUH	ALTER	TABELLE ÄNDERN - TBLPROPERTIES DEAKTIVIEREN	Unterstützt
KUH	ALTER	TABELLE ÄNDERN - SPALTE ÄNDERN	Unterstützt
KUH	ALTER	TABELLE ÄNDERN - SPALTEN HINZUFÜGEN	Unterstützt

Tabellentyp	Operation	SQL Write-Befehl	Status
KUH	ALTER	TABELLE ÄNDERN - PARTITION HINZUFÜGEN	Unterstützt
KUH	ALTER	TABELLE ÄNDERN - PARTITION LÖSCHEN	Unterstützt
KUH	ALTER	TABELLE ÄNDERN - PARTITIONEN WIEDERHERSTELLEN	Unterstützt
KUH	ALTER	REPARIERE TABELLENSYNCHRONISIERUNGSPARTITIONEN	Unterstützt
DROP	DROP	DROP TABLE	Unterstützt
KUH	DROP	TABELLE LÖSCHEN - LÖSCHEN	Unterstützt
CREATE	CREATE	TABELLE ERSTELLEN — Verwaltet	Unterstützt

Tabellentyp	Operation	SQL Write-Befehl	Status
KUH	CREATE	TABELLE ERSTELLEN - PARTITIONIEREN NACH	Unterstützt
KUH	CREATE	TABELLE ERSTELLEN, FALLS SIE NICHT EXISTIERT	Unterstützt
KUH	CREATE	CREATE TABLE LIKE	Unterstützt
KUH	CREATE	CREATE TABLE AS SELECT	Unterstützt
CREATE	CREATE	TABELLE MIT STANDORT ERSTELLEN — Externe Tabelle	Nicht unterstützt
DATENRAHMEN (EINFÜGEN)	DATENRAHMEN (EINFÜGEN)	saveAsTable. Überschreiben	Unterstützt
KUH	DATENRAHMEN (EINFÜGEN)	saveAsTable. Anhängen	Nicht unterstützt
KUH	DATENRAHMEN (EINFÜGEN)	saveAsTable. Ignorieren	Unterstützt
KUH	DATENRAHMEN (EINFÜGEN)	saveAsTable.ErrorIfExists	Unterstützt

Tabellentyp	Operation	SQL Write-Befehl	Status
KUH	DATENRAHMEN (EINFÜGEN)	saveAsTable - Externe Tabelle (Pfad)	Nicht unterstützt
KUH	DATENRAHMEN (EINFÜGEN)	speichern (Pfad) — DF v1	Nicht unterstützt
MEHR	INSERT	INSERT INTO TABLE	Unterstützt
MEHR	INSERT	IN TABELLE EINFÜGEN — PARTITION (statisch, dynamisch)	Unterstützt
MEHR	INSERT	INSERT OVERWRITE	Unterstützt
MEHR	INSERT	INSERT OVERWRITE - PARTITION (statisch, dynamisch)	Unterstützt
UPDATE	UPDATE	UPDATE TABLE	Unterstützt
MEHR	UPDATE	TABELLE AKTUALISIEREN — Partition ändern	Nicht unterstützt
DELETE	DELETE	DELETE FROM TABLE	Unterstützt

Tabellentyp	Operation	SQL Write-Befehl	Status
ALTER	ALTER	TABELLE ÄNDERN - UMBENENNEN IN	Nicht unterstützt
MEHR	ALTER	TABELLE ÄNDERN - TABELLENEIGENSCHAFTEN FESTLEGEN	Unterstützt
MEHR	ALTER	TABELLE ÄNDERN - TABELLENEIGENSCHAFTEN DEAKTIVIEREN	Unterstützt
MEHR	ALTER	TABELLE ÄNDERN - SPALTE ÄNDERN	Unterstützt
MEHR	ALTER	TABELLE ÄNDERN - SPALTEN HINZUFÜGEN	Unterstützt
MEHR	ALTER	TABELLE ÄNDERN - PARTITION HINZUFÜGEN	Unterstützt

Tabellentyp	Operation	SQL Write-Befehl	Status
MEHR	ALTER	TABELLE ÄNDERN - PARTITION LÖSCHEN	Unterstützt
MEHR	ALTER	TABELLE ÄNDERN - PARTITIONEN WIEDERHERSTELLEN	Unterstützt
MEHR	ALTER	REPARIERE TABELLENSYNCHRONISIERUNGSPARTITIONEN	Unterstützt
DROP	DROP	DROP TABLE	Unterstützt
MEHR	DROP	TABELLE LÖSCHEN - LÖSCHEN	Unterstützt
CREATE	CREATE	TABELLE ERSTELLEN — Verwaltet	Unterstützt
MEHR	CREATE	TABELLE ERSTELLEN - PARTITIONIEREN NACH	Unterstützt

Tabellentyp	Operation	SQL Write-Befehl	Status
MEHR	CREATE	TABELLE ERSTELLEN, FALLS NICHT VORHANDEN	Unterstützt
MEHR	CREATE	CREATE TABLE LIKE	Unterstützt
MEHR	CREATE	CREATE TABLE AS SELECT	Unterstützt
CREATE	CREATE	TABELLE MIT STANDORT ERSTELLEN - Externe Tabelle	Nicht unterstützt
DATENRAHMEN (UPSERT)	DATENRAHMEN (UPSERT)	saveAsTable. Überschreiben	Unterstützt
MEHR	DATENRAHMEN (VERÄRGERT)	saveAsTable. Anhängen	Nicht unterstützt
MEHR	DATENRAHMEN (VERÄRGERT)	saveAsTable. Ignorieren	Unterstützt
MEHR	DATENRAHMEN (VERÄRGERT)	saveAsTable.ErrorIfExists	Unterstützt
MEHR	DATENRAHMEN (VERÄRGERT)	saveAsTable - Externe Tabelle (Pfad)	Nicht unterstützt
MEHR	DATENRAHMEN (VERÄRGERT)	speichern (Pfad) — DF v1	Nicht unterstützt

Tabellentyp	Operation	SQL Write-Befehl	Status
DATENRAHMEN (LÖSCHEN)	DATENRAHMEN (LÖSCHEN)	saveAsTable. Anhängen	Nicht unterstützt
MEHR	DATENRAHMEN (LÖSCHEN)	saveAsTable - Externe Tabelle (Pfad)	Nicht unterstützt
MEHR	DATENRAHMEN (LÖSCHEN)	speichern (Pfad) — DF v1	Nicht unterstützt
DATENRAHMEN (BULK_INSERT)	DATENRAHMEN (BULK_INSERT)	saveAsTable. Überschreiben	Unterstützt
MEHR	DATENRAHMEN (BULK_INSERT)	saveAsTable. Anhängen	Nicht unterstützt
MEHR	DATENRAHMEN (BULK_INSERT)	saveAsTable. Ignorieren	Unterstützt
MEHR	DATENRAHMEN (BULK_INSERT)	saveAsTable.ErrorIfExists	Unterstützt
MEHR	DATENRAHMEN (BULK_INSERT)	saveAsTable - Externe Tabelle (Pfad)	Nicht unterstützt
MEHR	DATENRAHMEN (BULK_INSERT)	speichern (Pfad) — DF v1	Nicht unterstützt

Verwendung von EMR Serverless mit AWS Lake Formation für eine differenzierte Zugriffskontrolle

-Übersicht

Mit den Amazon EMR-Versionen 7.2.0 und höher können AWS Lake Formation Sie detaillierte Zugriffskontrollen auf Datenkatalogtabellen anwenden, die von S3 unterstützt werden. Mit dieser Funktion können Sie Zugriffskontrollen auf Tabellen-, Zeilen-, Spalten- und Zellenebene für read Abfragen innerhalb Ihrer Amazon EMR Serverless Spark-Jobs konfigurieren. Verwenden Sie EMR Studio, um eine detaillierte Zugriffskontrolle für Apache Spark-Batchjobs und interaktive Sitzungen zu konfigurieren. In den folgenden Abschnitten erfahren Sie mehr über Lake Formation und dessen Verwendung mit EMR Serverless.

Für die Nutzung von Amazon EMR Serverless mit AWS Lake Formation fallen zusätzliche Gebühren an. Weitere Informationen finden Sie unter Amazon EMR-Preise.

So funktioniert EMR Serverless mit AWS Lake Formation

Wenn Sie EMR Serverless mit Lake Formation verwenden, können Sie für jeden Spark-Job eine Berechtigungsebene erzwingen, um die Lake Formation Formation-Berechtigungssteuerung anzuwenden, wenn EMR Serverless Jobs ausführt. EMR Serverless verwendet Spark-Ressourcenprofile, um zwei Profile für die effektive Ausführung von Jobs zu erstellen. Das Benutzerprofil führt vom Benutzer bereitgestellten Code aus, während das Systemprofil die Lake-Formation-Richtlinien durchsetzt. Weitere Informationen finden Sie unter Was ist AWS Lake Formation und Überlegungen und Einschränkungen.

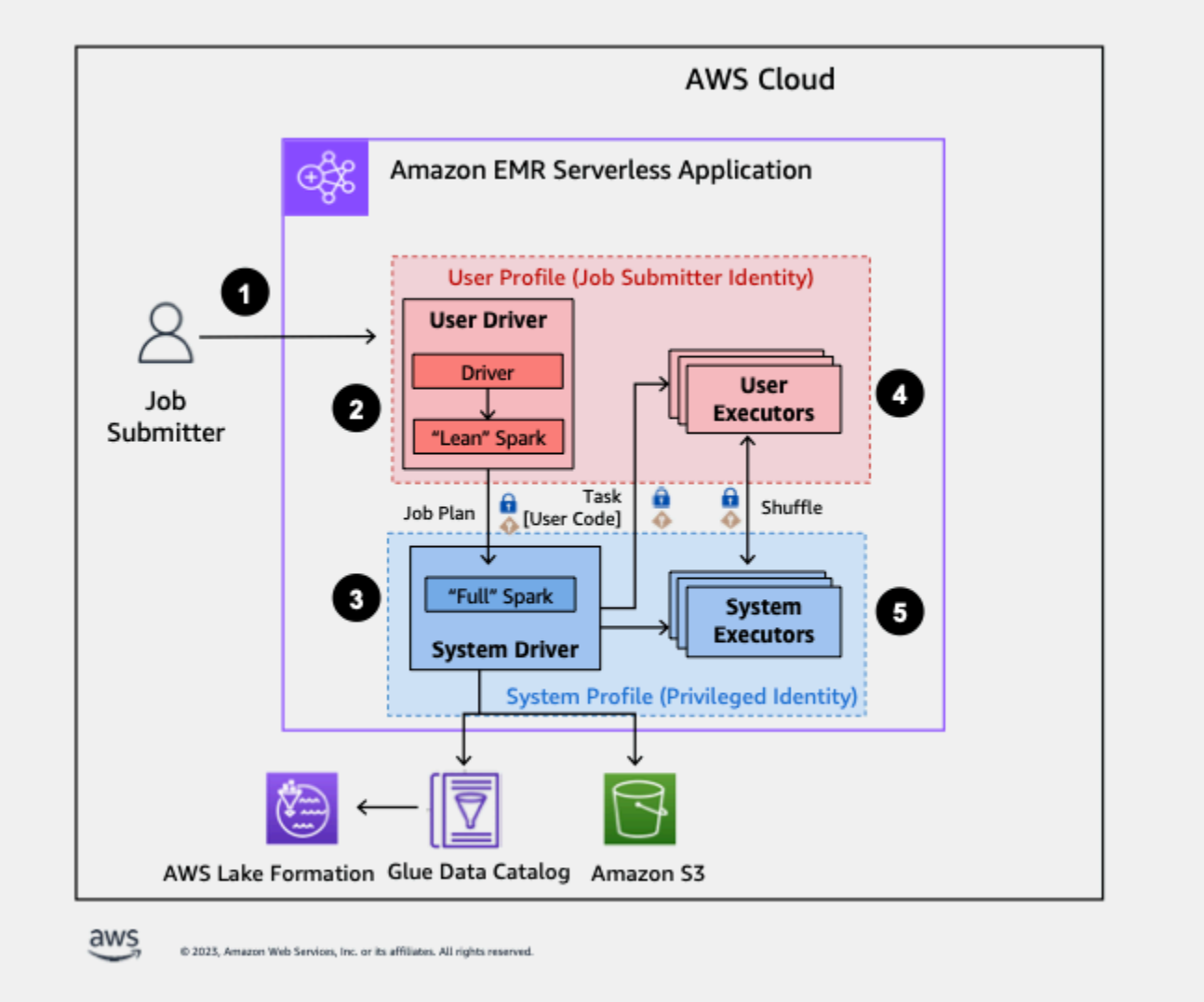
Wenn Sie vorinitialisierte Kapazität mit Lake Formation verwenden, empfehlen wir, mindestens zwei Spark-Treiber zu verwenden. Jeder Lake Formation-fähige Job verwendet zwei Spark-Treiber, einen für das Benutzerprofil und einen für das Systemprofil. Die beste Leistung erzielen Sie, wenn Sie für Lake Formation-fähige Jobs die doppelte Anzahl von Treibern verwenden, als wenn Sie Lake Formation nicht verwenden.

Wenn Sie Spark-Jobs auf EMR Serverless ausführen, sollten Sie auch die Auswirkungen der dynamischen Zuweisung auf das Ressourcenmanagement und die Clusterleistung berücksichtigen. Die Konfiguration `spark.dynamicAllocation.maxExecutors` der maximalen Anzahl von Executors pro Ressourcenprofil gilt für Benutzer- und System-Executoren. Wenn Sie diese Anzahl so konfigurieren, dass sie der maximal zulässigen Anzahl von Executoren entspricht, kann es sein, dass Ihre Jobausführung aufgrund eines Executortyps, der alle verfügbaren Ressourcen verwendet, blockiert wird, wodurch der andere Executor verhindert wird, wenn Sie Job-Jobs ausführen.

Damit Ihnen nicht die Ressourcen ausgehen, legt EMR Serverless die standardmäßige maximale Anzahl von Executoren pro Ressourcenprofil auf 90% des Werts fest. `spark.dynamicAllocation.maxExecutors` Sie können diese Konfiguration überschreiben, wenn Sie einen Wert zwischen `spark.dynamicAllocation.maxExecutorsRatio` 0 und 1 angeben. Konfigurieren Sie außerdem die folgenden Eigenschaften, um die Ressourcenzuweisung und die Gesamtleistung zu optimieren:

- `spark.dynamicAllocation.cachedExecutorIdleTimeout`
- `spark.dynamicAllocation.shuffleTracking.timeout`
- `spark.cleaner.periodicGC.interval`

Im Folgenden finden Sie einen allgemeinen Überblick darüber, wie EMR Serverless Zugriff auf Daten erhält, die durch Sicherheitsrichtlinien von Lake Formation geschützt sind.



1. Ein Benutzer sendet einen Spark-Job an eine AWS Lake Formation-fähige EMR Serverless-Anwendung.
2. EMR Serverless sendet den Job an einen Benutzertreiber und führt den Job im Benutzerprofil aus. Der Benutzertreiber führt eine schlanke Version von Spark aus, die nicht in der Lage ist, Aufgaben zu starten, Executors anzufordern, auf S3 oder den Glue-Katalog zuzugreifen. Er erstellt einen Auftragsplan.
3. EMR Serverless richtet einen zweiten Treiber ein, den Systemtreiber, und führt ihn im Systemprofil aus (mit einer privilegierten Identität). EMR Serverless richtet einen verschlüsselten TLS-Kanal zwischen den beiden Treibern für die Kommunikation ein. Der Benutzertreiber verwendet den Kanal, um die Auftragspläne an den Systemtreiber zu senden. Der Systemtreiber führt keinen vom Benutzer übermittelten Code aus. Er führt Spark vollständig aus und kommuniziert mit S3 und dem Datenkatalog für den Datenzugriff. Er fordert Executors an und stellt den Auftragsplan in eine Abfolge von Ausführungsphasen zusammen.
4. EMR Serverless führt dann die Stufen auf Executoren mit dem Benutzertreiber oder Systemtreiber aus. Benutzercode wird in jeder Phase ausschließlich auf Benutzerprofil-Executors ausgeführt.
5. Stufen, die Daten aus Datenkatalogtabellen lesen, die durch Sicherheitsfilter geschützt sind AWS Lake Formation oder solche, die Sicherheitsfilter anwenden, werden an System-Executoren delegiert.

Aktivierung der Lake Formation in Amazon EMR

Um Lake Formation zu aktivieren, legen Sie `spark.emr-serverless.lakeformation.enabled` beim Erstellen einer serverlosen EMR-Anwendung den Wert für den Laufzeitkonfigurationsparameter auf `true` `spark-defaults` Unterklassifizierung fest.

```
aws emr-serverless create-application \
    --release-label emr-7.12.0 \
    --runtime-configuration '{
     "classification": "spark-defaults",
     "properties": {
      "spark.emr-serverless.lakeformation.enabled": "true"
      }
    }' \
    --type "SPARK"
```

Sie können Lake Formation auch aktivieren, wenn Sie eine neue Anwendung in EMR Studio erstellen. Wählen Sie Lake Formation verwenden für eine detaillierte Zugriffskontrolle aus, die unter Zusätzliche Konfigurationen verfügbar ist.

Die Interworker-Verschlüsselung ist standardmäßig aktiviert, wenn Sie Lake Formation mit EMR Serverless verwenden, sodass Sie die Interworker-Verschlüsselung nicht erneut explizit aktivieren müssen.

Aktivierung von Lake Formation für Spark-Jobs

Um Lake Formation für einzelne Spark-Jobs `spark.emr-serverless.lakeformation.enabled` zu aktivieren, setzen Sie ihn bei der Verwendung auf true`spark-submit`.

```
--conf spark.emr-serverless.lakeformation.enabled=true
```

IAM-Berechtigungen für die Auftrag-Laufzeitrolle

Lake Formation Formation-Berechtigungen kontrollieren den Zugriff auf AWS Glue Data Catalog-Ressourcen, Amazon S3 S3-Standorte und die zugrunde liegenden Daten an diesen Standorten. IAM-Berechtigungen kontrollieren den Zugriff auf Lake Formation und AWS Glue APIs sowie auf Ressourcen. Obwohl Sie möglicherweise über die Lake-Formation-Berechtigung verfügen, auf eine Tabelle im Datenkatalog (SELECT) zuzugreifen, schlägt Ihr Vorgang fehl, wenn Sie nicht über die IAM-Berechtigung für den `glue:Get*`-API-Vorgang verfügen.

Die folgende Beispielrichtlinie beschreibt, wie Sie IAM-Berechtigungen für den Zugriff auf ein Skript in S3, das Hochladen von Protokollen in S3, API-Berechtigungen für AWS Glue und die Berechtigung für den Zugriff auf Lake Formation erteilen.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "ScriptAccess",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
```

```
      ],
      "Resource": [
        "arn:aws:s3:::*.amzn-s3-demo-bucket/scripts",
        "arn:aws:s3:::*.amzn-s3-demo-bucket/*"
      ]
    },
    {
      "Sid": "LoggingAccess",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/logs/*"
      ]
    },
    {
      "Sid": "GlueCatalogAccess",
      "Effect": "Allow",
      "Action": [
        "glue:Get*",
        "glue:Create*",
        "glue:Update*"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Sid": "LakeFormationAccess",
      "Effect": "Allow",
      "Action": [
        "lakeformation:GetDataAccess"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

Lake-Formation-Berechtigungen für die Auftrag-Laufzeitrolle einrichten

Registrieren Sie zunächst den Speicherort Ihrer Hive-Tabelle bei Lake Formation. Erstellen Sie anschließend Berechtigungen für Ihre Auftrag-Laufzeitrolle für die gewünschte Tabelle. Weitere Informationen zu Lake Formation finden Sie unter Was ist AWS Lake Formation? im AWS Lake Formation Entwicklerhandbuch.

Nachdem Sie die Lake Formation Formation-Berechtigungen eingerichtet haben, reichen Sie Spark-Jobs auf Amazon EMR Serverless ein. Weitere Informationen zu Spark-Jobs finden Sie in den Spark-Beispielen.

Senden einer Auftragsausführung

Nachdem Sie die Lake Formation Grants eingerichtet haben, können Sie Spark-Jobs auf EMR Serverless einreichen. Der folgende Abschnitt enthält Beispiele für die Konfiguration und Übermittlung der Eigenschaften für die Ausführung von Jobs.

Benötigte Berechtigungen

Tabellen, die nicht registriert sind in AWS Lake Formation

Bei Tabellen AWS Lake Formation, bei denen nicht registriert ist, greift die Job-Runtime-Rolle sowohl auf den AWS Glue-Datenkatalog als auch auf die zugrunde liegenden Tabellendaten in Amazon S3 zu. Dazu muss die Job-Runtime-Rolle über die entsprechenden IAM-Berechtigungen sowohl für AWS Glue- als auch für Amazon S3 S3-Operationen verfügen.

Tabellen sind registriert in AWS Lake Formation

Bei Tabellen AWS Lake Formation, bei denen registriert ist, greift die Job-Runtime-Rolle auf die Metadaten des AWS Glue-Datenkatalogs zu, während temporäre Anmeldeinformationen, die von Lake Formation bereitgestellt wurden, auf die zugrunde liegenden Tabellendaten in Amazon S3 zugreifen. Die Lake Formation Formation-Berechtigungen, die zur Ausführung eines Vorgangs erforderlich sind, hängen vom AWS Glue Data Catalog und den Amazon S3 S3-API-Aufrufen ab, die der Spark-Job initiiert, und lassen sich wie folgt zusammenfassen:

- Die DESCRIBE-Berechtigung ermöglicht es der Runtime-Rolle, Tabellen- oder Datenbankmetadaten im Datenkatalog zu lesen
- Die ALTER-Berechtigung ermöglicht der Runtime-Rolle, Tabellen- oder Datenbankmetadaten im Datenkatalog zu ändern

- Die DROP-Berechtigung ermöglicht der Runtime-Rolle, Tabellen- oder Datenbankmetadaten aus dem Datenkatalog zu löschen
- Die SELECT-Berechtigung ermöglicht es der Runtime-Rolle, Tabellendaten aus Amazon S3 zu lesen
- Die INSERT-Berechtigung ermöglicht es der Runtime-Rolle, Tabellendaten in Amazon S3 zu schreiben
- Die DELETE-Berechtigung ermöglicht es der Runtime-Rolle, Tabellendaten aus Amazon S3 zu löschen

> **Note**
>
> Lake Formation wertet Berechtigungen träge aus, wenn ein Spark-Job AWS Glue zum Abrufen von Tabellenmetadaten und Amazon S3 zum Abrufen von Tabellendaten aufruft. Jobs, die eine Runtime-Rolle mit unzureichenden Berechtigungen verwenden, schlagen erst fehl, wenn Spark einen AWS Glue- oder Amazon S3-Aufruf tätigt, für den die fehlende Berechtigung erforderlich ist.

> **Note**
>
> In der folgenden unterstützten Tabellenmatrix:
>
> - Als Unterstützt markierte Operationen verwenden ausschließlich Lake Formation-Anmeldeinformationen, um auf Tabellendaten für Tabellen zuzugreifen, die bei Lake Formation registriert sind. Wenn die Lake Formation Formation-Berechtigungen nicht ausreichen, greift der Vorgang nicht auf Anmeldeinformationen für Runtime-Rollen zurück. Bei Tabellen, die nicht bei Lake Formation registriert sind, greifen die Anmeldeinformationen der Job-Runtime-Rolle auf die Tabellendaten zu.
>
> - Operationen, die als Unterstützt mit IAM-Berechtigungen am Amazon S3-Standort markiert sind, verwenden keine Lake Formation Formation-Anmeldeinformationen für den Zugriff auf die zugrunde liegenden Tabellendaten in Amazon S3. Um diese Operationen auszuführen, muss die Job-Runtime-Rolle über die erforderlichen Amazon S3 S3-IAM-Berechtigungen für den Zugriff auf die Tabellendaten verfügen, unabhängig davon, ob die Tabelle bei Lake Formation registriert ist.

Hive

Operation	AWS Lake Formation Berechtigungen	Status der Support
SELECT	SELECT	Unterstützt
CREATE TABLE	CREATE_TABLE	Unterstützt
CREATE TABLE LIKE	TABELLE ERSTELLEN	Unterstützt mit IAM-Berechtigungen am Amazon S3 S3-Standort
CREATE TABLE AS SELECT	CREATE_TABLE	Unterstützt mit IAM-Berechtigungen am Amazon S3 S3-Standort
DESCRIBE TABLE	DESCRIBE	Unterstützt
SHOW TBLPROPERTIES	DESCRIBE	Unterstützt
SHOW_COLUMNS	DESCRIBE	Unterstützt
SHOW PARTITIONS	DESCRIBE	Unterstützt
SHOW CREATE TABLE	DESCRIBE	Unterstützt
TABELLE ÄNDERN `tablename`	SELECT und ALTER	Unterstützt
POSITION DES `tablename` TABELLENSETS ÄNDERN	-	Nicht unterstützt
TABELLE ÄNDERN PARTITION `tablename` HINZUFÜGEN	AUSWÄHLEN, EINFÜGEN und ÄNDERN	Unterstützt
REPAIR TABLE	SELECT und ALTER	Unterstützt

Operation	AWS Lake Formation Berechtigungen	Status der Support
DATEN LADEN		Nicht unterstützt
INSERT	INSERT und ALTER	Unterstützt
INSERT OVERWRITE	AUSWÄHLEN, EINFÜGEN, LÖSCHEN und ÄNDERN	Unterstützt
DROP TABLE	AUSWÄHLEN, ABLEGEN, LÖSCHEN und ÄNDERN	Unterstützt
TRUNCATE TABLE	AUSWÄHLEN, EINFÜGEN, LÖSCHEN und ÄNDERN	Unterstützt
Dataframe Writer V1	Entspricht der entsprech enden SQL-Operation	Wird unterstützt, wenn Daten an eine bestehende Tabelle angehängt werden. Weitere Informationen finden Sie unter Überlegungen und Einschränkungen
Dataframe Writer V2	Entspricht der entsprech enden SQL-Operation	Wird unterstützt, wenn Daten an eine bestehende Tabelle angehängt werden. Weitere Informationen finden Sie unter Überlegungen und Einschränkungen

Iceberg

Operation	AWS Lake Formation Berechtigungen	Status der Support
SELECT	SELECT	Unterstützt

Operation	AWS Lake Formation Berechtigungen	Status der Support
CREATE TABLE	CREATE_TABLE	Unterstützt
CREATE TABLE LIKE	TABELLE ERSTELLEN	Unterstützt mit IAM-Berechtigungen am Amazon S3 S3-Standort
CREATE TABLE AS SELECT	CREATE_TABLE	Unterstützt mit IAM-Berechtigungen am Amazon S3 S3-Standort
ERSETZT DIE TABELLE ALS AUSWAHL	SELECT, INSERT und ALTER	Unterstützt
DESCRIBE TABLE	DESCRIBE	Unterstützt mit IAM-Berechtigungen am Amazon S3 S3-Standort
SHOW TBLPROPERTIES	DESCRIBE	Unterstützt mit IAM-Berechtigungen am Amazon S3 S3-Standort
SHOW CREATE TABLE	DESCRIBE	Unterstützt mit IAM-Berechtigungen am Amazon S3 S3-Standort
ALTER TABLE	SELECT, INSERT und ALTER	Unterstützt
ALTER TABLE SET LOCATION	SELECT, INSERT und ALTER	Unterstützt mit IAM-Berechtigungen am Amazon S3 S3-Standort
ÄNDERN SIE DEN SCHREIBVORGANG IN DER TABELLE, GEORDNET NACH	SELECT, INSERT und ALTER	Unterstützt mit IAM-Berechtigungen am Amazon S3 S3-Standort
DER SCHREIBVORGANG IN DER TABELLE WIRD VERTEILT VON	SELECT, INSERT und ALTER	Unterstützt mit IAM-Berechtigungen am Amazon S3 S3-Standort

Operation	AWS Lake Formation Berechtigungen	Status der Support
TABELLE ÄNDERN TABELLE UMBENENNEN	CREATE_TABLE und DROP	Unterstützt
INSERT INTO	SELECT, INSERT und ALTER	Unterstützt
INSERT OVERWRITE	SELECT, INSERT und ALTER	Unterstützt
DELETE	SELECT, INSERT und ALTER	Unterstützt
UPDATE	SELECT, INSERT und ALTER	Unterstützt
MERGE INTO	SELECT, INSERT und ALTER	Unterstützt
DROP TABLE	AUSWÄHLEN , LÖSCHEN und ABLEGEN	Unterstützt
DataFrame Writer V1	-	Nicht unterstützt
DataFrame Schriftsteller V2	Entspricht der entsprechenden SQL-Operation	Wird unterstützt, wenn Daten an eine bestehende Tabelle angehängt werden. Weitere Informationen finden Sie unter Überlegungen und Einschränkungen.
Metadaten-Tabellen	SELECT	Unterstützt. Bestimmte Tabellen sind ausgeblendet. Weitere Informationen finden Sie unter Überlegungen und Einschränkungen.

Operation	AWS Lake Formation Berechtigungen	Status der Support
Gespeicherte Prozedure n	-	Wird für Tabellen unterstützt, die die folgenden Bedingungen erfüllen: • Tabellen, die nicht registriert sind in AWS Lake Formation • Tabellen, die `register_table` und nicht verwenden `migrate` Weitere Informationen finden Sie unter Überlegungen und Einschränkungen.

Spark-Konfiguration für Iceberg: Das folgende Beispiel zeigt, wie Spark mit Iceberg konfiguriert wird. Geben Sie die folgenden Eigenschaften an, um Iceberg-Jobs auszuführen. `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
```

Hudi

Operation	AWS Lake Formation Berechtigungen	Status der Support
SELECT	SELECT	Unterstützt
CREATE TABLE	CREATE_TABLE	Unterstützt mit IAM-Berechtigungen am Amazon S3 S3-Standort

Operation	AWS Lake Formation Berechtigungen	Status der Support
CREATE TABLE LIKE	CREATE_TABLE	Unterstützt mit IAM-Berechtigungen am Amazon S3 S3-Standort
CREATE TABLE AS SELECT	-	Nicht unterstützt
DESCRIBE TABLE	DESCRIBE	Unterstützt mit IAM-Berechtigungen am Amazon S3 S3-Standort
SHOW TBLPROPERTIES	DESCRIBE	Unterstützt mit IAM-Berechtigungen am Amazon S3 S3-Standort
SHOW_COLUMNS	DESCRIBE	Unterstützt mit IAM-Berechtigungen am Amazon S3 S3-Standort
SHOW CREATE TABLE	DESCRIBE	Unterstützt mit IAM-Berechtigungen am Amazon S3 S3-Standort
ALTER TABLE	SELECT	Unterstützt mit IAM-Berechtigungen am Amazon S3 S3-Standort
INSERT INTO	SELECT und ALTER	Unterstützt mit IAM-Berechtigungen am Amazon S3 S3-Standort
INSERT OVERWRITE	SELECT und ALTER	Unterstützt mit IAM-Berechtigungen am Amazon S3 S3-Standort
DELETE	-	Nicht unterstützt

Operation	AWS Lake Formation Berechtigungen	Status der Support
UPDATE	-	Nicht unterstützt
MERGE INTO	-	Nicht unterstützt
DROP TABLE	AUSWÄHLEN und ABLEGEN	Unterstützt mit IAM-Berechtigungen am Amazon S3 S3-Standort
DataFrame Writer V1	-	Nicht unterstützt
DataFrame Schriftsteller V2	Entspricht der entsprechenden SQL-Operation	Unterstützt mit IAM-Berechtigungen am Amazon S3 S3-Standort
Metadaten-Tabellen	-	Nicht unterstützt
Funktionen zur Tabellenverwaltung und Hilfsprogramme	-	Nicht unterstützt

In den folgenden Beispielen wird Spark mit Hudi konfiguriert, wobei Dateipositionen und andere Eigenschaften angegeben werden, die für die Verwendung erforderlich sind.

Spark-Konfiguration für Hudi: Wenn dieser Ausschnitt in einem Notizbuch verwendet wird, gibt er den Pfad zur Hudi Spark-Bundle-JAR-Datei an, die die Hudi-Funktionalität in Spark aktiviert. Außerdem wird Spark so konfiguriert, dass der AWS Glue-Datenkatalog als Metastore verwendet wird.

```
%%configure -f
{
    "conf": {
        "spark.jars": "/usr/lib/hudi/hudi-spark-bundle.jar",
        "spark.hadoop.hive.metastore.client.factory.class":
 "com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory",
        "spark.serializer": "org.apache.spark.serializer.JavaSerializer",
        "spark.sql.catalog.spark_catalog":
 "org.apache.spark.sql.hudi.catalog.HoodieCatalog",
```

```
        "spark.sql.extensions":
 "org.apache.spark.sql.hudi.HoodieSparkSessionExtension"
    }
}
```

Spark-Konfiguration für Hudi mit AWS Glue: Wenn dieses Snippet in einem Notizbuch verwendet wird, aktiviert es Hudi als unterstütztes Data-Lake-Format und stellt sicher, dass Hudi-Bibliotheken und Abhängigkeiten verfügbar sind.

```
%%configure
{
    "--conf": "spark.serializer=org.apache.spark.serializer.JavaSerializer --conf
spark.sql.catalog.spark_catalog=org.apache.spark.sql.hudi.catalog.HoodieCatalog --
conf
spark.sql.extensions=org.apache.spark.sql.hudi.HoodieSparkSessionExtension",
    "--datalake-formats": "hudi",
    "--enable-glue-datacatalog": True,
    "--enable-lakeformation-fine-grained-access": "true"
}
```

Delta Lake

Operation	AWS Lake Formation Berechtigungen	Status der Support
SELECT	SELECT	Unterstützt
CREATE TABLE	CREATE_TABLE	Unterstützt
CREATE TABLE LIKE	-	Nicht unterstützt
CREATE TABLE AS SELECT	TABELLE ERSTELLEN	Unterstützt
ERSETZT DIE TABELLE ALS AUSWAHL	SELECT, INSERT und ALTER	Unterstützt
DESCRIBE TABLE	DESCRIBE	Unterstützt mit IAM-Berechtigungen am Amazon S3 S3-Standort

Operation	AWS Lake Formation Berechtigungen	Status der Support
SHOW TBLPROPERTIES	DESCRIBE	Unterstützt mit IAM-Berechtigungen am Amazon S3 S3-Standort
SHOW_COLUMNS	DESCRIBE	Unterstützt mit IAM-Berechtigungen am Amazon S3 S3-Standort
SHOW CREATE TABLE	DESCRIBE	Unterstützt mit IAM-Berechtigungen am Amazon S3 S3-Standort
ALTER TABLE	AUSWÄHLEN und EINFÜGEN	Unterstützt
ALTER TABLE SET LOCATION	AUSWÄHLEN und EINFÜGEN	Unterstützt mit IAM-Berechtigungen am Amazon S3 S3-Standort
ÄNDERN SIE DEN `tablename` TABELLENCLUSTER VON	AUSWÄHLEN und EINFÜGEN	Unterstützt mit IAM-Berechtigungen am Amazon S3 S3-Standort
TABELLE ÄNDERN EINSCHRÄNKUNG `tablename` HINZUFÜGEN	AUSWÄHLEN und EINFÜGEN	Unterstützt mit IAM-Berechtigungen am Amazon S3 S3-Standort
ÄNDERN SIE DIE EINSCHRÄNKUNG BEIM `tablename` ABLEGEN VON	AUSWÄHLEN und EINFÜGEN	Unterstützt mit IAM-Berechtigungen am Amazon S3 S3-Standort
INSERT INTO	AUSWÄHLEN und EINFÜGEN	Unterstützt

Operation	AWS Lake Formation Berechtigungen	Status der Support
INSERT OVERWRITE	AUSWÄHLEN und EINFÜGEN	Unterstützt
DELETE	AUSWÄHLEN und EINFÜGEN	Unterstützt
UPDATE	AUSWÄHLEN und EINFÜGEN	Unterstützt
MERGE INTO	AUSWÄHLEN und EINFÜGEN	Unterstützt
DROP TABLE	AUSWÄHLEN, LÖSCHEN und ABLEGEN	Unterstützt
DataFrame Writer V1	-	Nicht unterstützt
DataFrame Schriftsteller V2	Entspricht der entsprech enden SQL-Operation	Unterstützt
Funktionen zur Tabellenv erwaltung und Hilfsprog ramme	-	Nicht unterstützt

EMR Serverless mit Delta Lake: Um Delta Lake with Lake Formation auf EMR Serverless zu verwenden, führen Sie den folgenden Befehl aus:

```
spark-sql \
  --conf
 spark.sql.extensions=io.delta.sql.DeltaSparkSessionExtension,com.amazonaws.emr.recordserver
 \
  --conf
 spark.sql.catalog.spark_catalog=org.apache.spark.sql.delta.catalog.DeltaCatalog \
```

Debuggen von Aufträgen

> **Note**
>
> Mit dieser Funktion können Sie auf stdout- und stderr-Protokolle für das Systemprofil von Workern zugreifen, die möglicherweise vertrauliche, ungefilterte Informationen enthalten. Die folgende Berechtigung sollte nur für den Zugriff auf Daten verwendet werden, die nicht zur Produktion verwendet werden. Für Anwendungen, die für die Verwendung mit Produktionsaufträgen erstellt wurden, empfehlen wir dringend, diese Berechtigungen nur Administratoren oder Benutzern mit erhöhtem Datenzugriff zuzuweisen.

Mit EMR-7.3.0 und höher ermöglicht EMR Serverless die Selbstdebugging-Funktion für Lake Formation-fähige Batch-Jobs. Verwenden Sie dazu den neuen Parameter Logs in der API. accessSystemProfile GetDashboardForJobRun Wenn accessSystemProfileLogs auf true gesetzt ist, können Sie auf die stdout- und stderr-Protokolle für die Systemprofil-Worker zugreifen, die zum Debuggen eines Lake Formation-fähigen EMR-Serverless-Batchjobs verwendet werden können.

```
aws emr-serverless get-dashboard-for-job-run \
  --application-id application-id
  --job-run-id job-run-id
  --access-system-profile-logs
```

Erforderliche Berechtigungen

Der Principal, der Lake Formation-fähige Batch-Jobs mithilfe von Batch-Jobs debuggen möchte, GetDashboardForJobRunmuss über die folgenden zusätzlichen Berechtigungen verfügen:

```
{
    "Sid": "AccessSystemProfileLogs",
    "Effect": "Allow",
    "Action": [
        "emr-serverless:GetDashboardForJobRun",
        "emr-serverless:AccessSystemProfileLogs",
        "glue:GetDatabases",
        "glue:SearchTables"
    ],
    "Resource": [
        "arn:aws:emr-serverless:region:account-id:/applications/applicationId/
jobruns/jobid",
```

```
        "arn:aws:glue:region:account-id:catalog",
        "arn:aws:glue:region:account-id:database/*",
        "arn:aws:glue:region:account-id:table/*/*"
    ]
}
```

Überlegungen

Systemprofilprotokolle für das Debuggen sind für Jobs sichtbar, die auf Datenbanken oder Tabellen in Lake Formation innerhalb desselben Kontos wie der Job zugreifen. Sie sind in den folgenden Szenarien nicht sichtbar:

- Wenn der mit Lake Formation Formation-Berechtigungen verwaltete Datenkatalog kontenübergreifende Datenbanken und Tabellen enthält
- Wenn der mit Lake Formation Formation-Berechtigungen verwaltete Datenkatalog Ressourcenlinks enthält

Mit Glue-Datenkatalogansichten arbeiten

Sie können Ansichten im AWS Glue Data Catalog für die Verwendung mit EMR Serverless erstellen und verwalten. Diese werden allgemein als AWS Glue Data Catalog-Ansichten bezeichnet. Diese Ansichten sind nützlich, da sie mehrere SQL-Abfrage-Engines unterstützen, sodass Sie über verschiedene AWS Dienste wie EMR Serverless und Amazon Redshift auf dieselbe Ansicht zugreifen können. Amazon Athena

Indem Sie eine Ansicht im Datenkatalog erstellen, verwenden Sie Ressourcenzuweisungen und tagbasierte Zugriffskontrollen, um Zugriff darauf AWS Lake Formation zu gewähren. Mit dieser Methode der Zugriffskontrolle müssen Sie keinen zusätzlichen Zugriff auf die Tabellen konfigurieren, auf die Sie beim Erstellen der Ansicht verwiesen haben. Diese Methode zum Gewähren von Berechtigungen wird als Definer-Semantik bezeichnet, und diese Ansichten werden als Definer-Ansichten bezeichnet. Weitere Informationen zur Zugriffskontrolle in Lake Formation finden Sie unter Gewähren und Widerrufen von Berechtigungen für Datenkatalogressourcen im AWS Lake Formation Developer Guide.

Data-Catalog-Ansichten sind in folgenden Anwendungsfällen nützlich:

- Präzisere Zugriffskontrolle – Sie können eine Ansicht erstellen, die den Datenzugriff auf der Grundlage der vom Benutzer benötigten Berechtigungen einschränkt. Mithilfe von Ansichten

im Datenkatalog können Sie beispielsweise verhindern, dass Mitarbeiter, die nicht in der Personalabteilung arbeiten, persönlich identifizierbare Informationen (PII) sehen.

- Vollständige Ansichtsdefinition — Indem Sie Filter auf Ihre Ansicht im Datenkatalog anwenden, stellen Sie sicher, dass die in einer Ansicht im Datenkatalog verfügbaren Datensätze immer vollständig sind.
- Verbesserte Sicherheit — Die zur Erstellung der Ansicht verwendete Abfragedefinition muss vollständig sein. Dieser Vorteil bedeutet, dass Ansichten im Datenkatalog weniger anfällig für SQL-Befehle von böswilligen Akteuren sind.
- Einfaches Teilen von Daten — Teilen Sie Daten mit anderen AWS Konten, ohne Daten verschieben zu müssen. Weitere Informationen finden Sie unter Kontoübergreifender Datenaustausch in Lake Formation.

Erstellen einer Data-Catalog-Ansicht

Es gibt verschiedene Möglichkeiten, eine Datenkatalogansicht zu erstellen. Dazu gehört die Verwendung von AWS CLI oder Spark-SQL. Es folgen einige Beispiele.

Using SQL

Im Folgenden wird die Syntax zum Erstellen einer Datenkatalogansicht veranschaulicht. Notieren Sie sich den `MULTI DIALECT` Ansichtstyp. Dies unterscheidet die Datenkatalogansicht von anderen Ansichten. Das `SECURITY` Prädikat ist angegeben als. `DEFINER` Dies weist auf eine Datenkatalogansicht mit `DEFINER` Semantik hin.

```
CREATE [ OR REPLACE ] PROTECTED MULTI DIALECT VIEW [IF NOT EXISTS] view_name
[(column_name [COMMENT column_comment], ...) ]
[ COMMENT view_comment ]
[TBLPROPERTIES (property_name = property_value, ... )]
SECURITY DEFINER
AS query;
```

Im Folgenden finden Sie eine `CREATE` Beispielanweisung, die der Syntax folgt:

```
CREATE PROTECTED MULTI DIALECT VIEW catalog_view
SECURITY DEFINER
AS
SELECT order_date, sum(totalprice) AS price
FROM source_table
```

```
GROUP BY order_date
```

Sie können eine Ansicht auch im Probelaufmodus mit SQL erstellen, um die Erstellung der Ansicht zu testen, ohne die Ressource tatsächlich zu erstellen. Die Verwendung dieser Option führt zu einem „Probelauf“, der die Eingabe validiert und, falls die Validierung erfolgreich ist, die JSON des AWS Glue-Tabellenobjekts zurückgibt, das die Ansicht darstellt. In diesem Fall wird die tatsächliche Ansicht nicht erstellt.

```
CREATE [ OR REPLACE ] PROTECTED MULTI DIALECT VIEW view_name
SECURITY DEFINER
[ SHOW VIEW JSON ]
AS view-sql
```

Using the AWS CLI

> **Note**
>
> Wenn Sie den CLI-Befehl verwenden, wird das SQL, das zum Erstellen der Ansicht verwendet wurde, nicht analysiert. Dies kann dazu führen, dass die Ansicht zwar erstellt wird, Abfragen jedoch nicht erfolgreich sind. Stellen Sie sicher, dass Sie Ihre SQL-Syntax testen, bevor Sie die Ansicht erstellen.

Sie verwenden den folgenden CLI-Befehl, um eine Ansicht zu erstellen:

```
aws glue create-table --cli-input-json '{
  "DatabaseName": "database",
  "TableInput": {
    "Name": "view",
    "StorageDescriptor": {
      "Columns": [
        {
          "Name": "col1",
          "Type": "data-type"
        },
        ...
        {
          "Name": "col_n",
          "Type": "data-type"
        }
```

```
      ],
      "SerdeInfo": {}
    },
    "ViewDefinition": {
      "SubObjects": [
        "arn:aws:glue:aws-region:aws-account-id:table/database/referenced-table1",
        ...
        "arn:aws:glue:aws-region:aws-account-id:table/database/referenced-tableN",
       ],
      "IsProtected": true,
      "Representations": [
        {
          "Dialect": "SPARK",
          "DialectVersion": "1.0",
          "ViewOriginalText": "Spark-SQL",
          "ViewExpandedText": "Spark-SQL"
        }
      ]
    }
  }
}'
```

Unterstützte Ansichtsvorgänge

Die folgenden Befehlsfragmente zeigen Ihnen verschiedene Möglichkeiten, mit Data-Catalog-Ansichten zu arbeiten:

- ANSICHT ERSTELLEN

  Erstellt eine Data-Catalog-Ansicht. Dies ist ein Beispiel für das Erstellen einer Ansicht aus einer vorhandenen Tabelle:

  ```
  CREATE PROTECTED MULTI DIALECT VIEW catalog_view
  SECURITY DEFINER AS SELECT * FROM my_catalog.my_database.source_table
  ```

- ALTER VIEW

  Verfügbare Syntax:

  - `ALTER VIEW view_name [FORCE] ADD DIALECT AS query`
  - `ALTER VIEW view_name [FORCE] UPDATE DIALECT AS query`
  - `ALTER VIEW view_name DROP DIALECT`

Sie können die Option `FORCE ADD DIALECT` verwenden, um das Aktualisieren des Schemas und der Unterobjekte gemäß dem neuen Engine-Dialekt zu erzwingen. Beachten Sie, dass dies zu Abfragefehlern führen kann, wenn Sie nicht auch `FORCE` verwenden, um andere Engine-Dialekte zu aktualisieren. Im Folgenden wird ein Beispiel veranschaulicht:

```
ALTER VIEW catalog_view FORCE ADD DIALECT
AS
SELECT order_date, sum(totalprice) AS price
FROM source_table
GROUP BY orderdate;
```

Im Folgenden wird gezeigt, wie eine Ansicht geändert werden kann, um den Dialekt zu aktualisieren:

```
ALTER VIEW catalog_view UPDATE DIALECT AS
SELECT count(*) FROM my_catalog.my_database.source_table;
```

- DESCRIBE VIEW

  Verfügbare Syntax für die Beschreibung einer Ansicht:

  - `SHOW COLUMNS {FROM|IN} view_name [{FROM|IN} database_name]`— Wenn der Benutzer über die erforderlichen AWS Glue- und Lake Formation Formation-Berechtigungen verfügt, um die Ansicht zu beschreiben, kann er die Spalten auflisten. Im Folgenden werden einige Beispielbefehle zum Anzeigen von Spalten veranschaulicht:

    ```
    SHOW COLUMNS FROM my_database.source_table;
    SHOW COLUMNS IN my_database.source_table;
    ```

  - `DESCRIBE view_name`— Wenn der Benutzer über die erforderlichen AWS Glue- und Lake Formation Formation-Berechtigungen verfügt, um die Ansicht zu beschreiben, kann er die Spalten in der Ansicht zusammen mit ihren Metadaten auflisten.

- DROP VIEW

  Verfügbare Syntax:

  - `DROP VIEW [ IF EXISTS ] view_name`

    Das folgende Beispiel zeigt eine `DROP`-Anweisung, die testet, ob eine Ansicht vorhanden ist, bevor sie gelöscht wird:

```
DROP VIEW IF EXISTS catalog_view;
```

- ANSICHT ANZEIGEN, ANSICHT ERSTELLEN
  - `SHOW CREATE VIEW view_name` – Zeigt die SQL-Anweisung an, die die angegebene Ansicht erstellt. Dies ist ein Beispiel für das Erstellen einer Data-Catalog-Ansicht:

```
SHOW CREATE TABLE my_database.catalog_view;
CREATE PROTECTED MULTI DIALECT VIEW my_catalog.my_database.catalog_view (
  net_profit,
  customer_id,
  item_id,
  sold_date)
TBLPROPERTIES (
  'transient_lastDdlTime' = '1736267222')
SECURITY DEFINER AS SELECT * FROM
my_database.store_sales_partitioned_lf WHERE customer_id IN (SELECT customer_id
 from source_table limit 10)
```

- SHOW VIEWS

  Listet alle Ansichten im Katalog auf, z. B. reguläre Ansichten, Ansichten mit mehreren Dialekten (MDV) und MDV ohne Spark-Dialekt. Die verfügbare Syntax lautet wie folgt:

  - `SHOW VIEWS [{ FROM | IN } database_name] [LIKE regex_pattern]`:

    Im Folgenden wird ein Beispielbefehl zum Anzeigen von Ansichten veranschaulicht:

```
SHOW VIEWS IN marketing_analytics LIKE 'catalog_view*';
```

Weitere Informationen zum Erstellen und Konfigurieren von Datenkatalog-Ansichten finden Sie unter Building AWS Glue Data Catalog Views im AWS Lake Formation Developer Guide.

Abfrage einer Data-Catalog-Ansicht

Nachdem Sie eine Datenkatalog-Ansicht erstellt haben, können Sie sie mit einem Amazon EMR Serverless Spark-Job abfragen, für den eine AWS Lake Formation detaillierte Zugriffskontrolle aktiviert ist. Die Job-Runtime-Rolle muss über die Lake Formation `SELECT` Formation-Berechtigung in der Datenkatalogansicht verfügen. Sie müssen keinen Zugriff auf die zugrunde liegenden Tabellen gewähren, auf die in der Ansicht verwiesen wird.

Sobald Sie alles eingerichtet haben, können Sie Ihre Ansicht abfragen. Führen Sie beispielsweise nach dem Erstellen einer serverlosen EMR-Anwendung in EMR Studio die folgende Abfrage aus, um auf eine Ansicht zuzugreifen.

```
SELECT * from my_database.catalog_view LIMIT 10;
```

Eine hilfreiche Funktion ist die. `invoker_principal` Sie gibt den eindeutigen Bezeichner der EMRS-Job-Runtime-Rolle zurück. Dies kann verwendet werden, um die View-Ausgabe auf der Grundlage des aufrufenden Prinzipals zu steuern. Sie können dies verwenden, um Ihrer Ansicht eine Bedingung hinzuzufügen, die die Abfrageergebnisse auf der Grundlage der aufrufenden Rolle verfeinert. Die Job-Runtime-Rolle muss über die Berechtigung für die `LakeFormation:GetDataLakePrincipal` IAM-Aktion verfügen, um diese Funktion verwenden zu können.

```
select invoker_principal();
```

Sie können diese Funktion beispielsweise einer `WHERE` Klausel hinzufügen, um die Abfrageergebnisse zu verfeinern.

Überlegungen und Einschränkungen

Wenn Sie Datenkatalogsichten erstellen, gilt Folgendes:

- Sie können Datenkatalogansichten nur mit Amazon EMR 7.6 und höher erstellen.
- Der Data-Catalog-Ansicht-Definer muss `SELECT`-Zugriff auf die zugrunde liegenden Basistabellen haben, auf die in der Ansicht zugegriffen wird. Das Erstellen der Data-Catalog-Ansicht schlägt fehl, wenn einer bestimmten Basistabelle Lake-Formation-Filter zugewiesen wurden, die der Definier-Rolle zugewiesen wurden.
- Basistabellen dürfen nicht über die `IAMAllowedPrincipals` Data Lake-Berechtigung in Lake Formation verfügen. Falls vorhanden, tritt der Fehler Multi Dialect views may only reference tables without IAMAllowed Principals permissions auf.
- Der Amazon-S3-Speicherort der Tabelle muss als Data-Lake-Speicherort registriert sein. Wenn die Tabelle nicht registriert ist, tritt der Fehler Multi Dialect views may only reference Lake Formation managed tables auf. Informationen zur Registrierung von Amazon S3 S3-Standorten in Lake Formation finden Sie unter Registrierung eines Amazon S3 S3-Standorts im AWS Lake Formation Entwicklerhandbuch.

- Sie können nur `PROTECTED`-Data-Catalog-Ansichten erstellen. `UNPROTECTED`-Ansichten werden nicht unterstützt.
- Sie können in einer Datenkatalog-Ansichtsdefinition nicht auf Tabellen in einem anderen AWS Konto verweisen. Sie können auch nicht auf eine Tabelle in demselben Konto verweisen, das sich in einer separaten Region befindet.
- Um Daten für ein Konto oder eine Region gemeinsam zu nutzen, muss die gesamte Ansicht mithilfe von Lake Formation Formation-Ressourcenlinks konto- und regionsübergreifend geteilt werden.
- Benutzerdefinierte Funktionen (UDFs) werden nicht unterstützt.
- Sie können Ansichten verwenden, die auf Iceberg-Tabellen basieren. Die Open-Table-Formate Apache Hudi und Delta Lake werden ebenfalls unterstützt.
- In Data-Catalog-Ansichten kann nicht auf andere Ansichten verwiesen werden.
- Ein AWS Ansichtschema für den Glue Data Catalog wird immer in Kleinbuchstaben gespeichert. Wenn Sie beispielsweise eine DDL-Anweisung verwenden, um eine Glue-Datenkatalog-Ansicht mit einer Spalte namens zu erstellen`Castle`, wird die im Glue-Datenkatalog erstellte Spalte in Kleinbuchstaben umgewandelt, also. `castle` Wenn Sie dann den Spaltennamen in einer DML-Abfrage als `Castle` oder angeben`CASTLE`, macht EMR Spark den Namen klein, damit Sie die Abfrage ausführen können. Die Spaltenüberschrift wird jedoch in der Groß-/Kleinschreibung angezeigt, die Sie in der Abfrage angegeben haben.

  Wenn Sie möchten, dass eine Abfrage fehlschlägt, wenn ein in der DML-Abfrage angegebener Spaltenname nicht mit dem Spaltennamen im Glue-Datenkatalog übereinstimmt, legen Sie fest`spark.sql.caseSensitive=true`.

Unterstützung für das offene Tabellenformat

EMR Serverless unterstützt SELECT-Abfragen auf Apache Hive, Apache Iceberg, Delta Lake (7.6.0+) und Apache Hudi (7.6.0+). Ab EMR 7.12 werden DML- und DDL-Operationen, die Tabellendaten ändern, für Apache Hive-, Apache Iceberg- und Delta Lake-Tabellen mit von Lake Formation vergebenen Anmeldeinformationen unterstützt.

Überlegungen und Einschränkungen

General

Beachten Sie die folgenden Einschränkungen bei der Verwendung von Lake Formation mit EMR Serverless.

> **Note**
>
> Wenn Sie Lake Formation für einen Spark-Job auf EMR Serverless aktivieren, startet der Job einen Systemtreiber und einen Benutzertreiber. Wenn Sie beim Start vorinitialisierte Kapazität angegeben haben, werden die Treiber anhand der vorinitialisierten Kapazität bereitgestellt, und die Anzahl der Systemtreiber entspricht der Anzahl der von Ihnen angegebenen Benutzertreiber. Wenn Sie On-Demand-Kapazität wählen, startet EMR Serverless zusätzlich zu einem Benutzertreiber auch einen Systemtreiber. Um die Kosten zu schätzen, die mit Ihrem EMR Serverless with Lake Formation Formation-Job verbunden sind, verwenden Sie den. AWS Pricing Calculator

- Amazon EMR Serverless with Lake Formation ist in allen unterstützten EMR-Serverless-Regionen verfügbar.

- Lake Formation-fähige Anwendungen unterstützen die Verwendung von benutzerdefinierten EMR Serverless-Images nicht.
- `DynamicResourceAllocation`Für Jobs in Lake Formation kann man nicht abschalten.
- Sie können Lake Formation nur mit Spark-Aufträgen verwenden.
- EMR Serverless mit Lake Formation unterstützt nur eine einzige Spark-Sitzung während eines Jobs.
- EMR Serverless with Lake Formation unterstützt nur kontenübergreifende Tabellenabfragen, die über Ressourcenlinks gemeinsam genutzt werden.
- Folgendes wird nicht unterstützt:
  - Resilient Distributed Datasets (RDD)
  - Spark-Streaming
  - Zugriffskontrolle für verschachtelte Spalten
- EMR Serverless blockiert Funktionen, die die vollständige Isolierung des Systemtreibers untergraben könnten, darunter die folgenden:
  - UDTs, Hive und alle benutzerdefinierten FunktionenUDFs, die benutzerdefinierte Klassen beinhalten
  - Benutzerdefinierte Datenquellen
  - Bereitstellung zusätzlicher JARs für Spark-Erweiterungen, Connectors oder Metastore
  - `ANALYZE TABLE` command

- Wenn sich Ihre EMR-Serverless-Anwendung in einem privaten Subnetz mit VPC-Endpunkten für Amazon S3 befindet und Sie eine Endpunktrichtlinie zur Zugriffskontrolle anhängen, müssen Sie, bevor Ihre Jobs Protokolldaten an AWS Managed Amazon S3 senden können, die unter Verwalteter Speicher aufgeführten Berechtigungen in Ihre VPC-Richtlinie für den S3-Gateway-Endpunkt aufnehmen. Wenden Sie sich bei Anfragen zur Fehlerbehebung an den Support. AWS
- Ab Amazon EMR 7.9.0 unterstützt Spark FGAC das AFile S3-System, wenn es mit dem s3a://-Schema verwendet wird.
- Amazon EMR 7.11 unterstützt die Erstellung verwalteter Tabellen mithilfe von CTAS.
- Amazon EMR 7.12 unterstützt die Erstellung verwalteter und externer Tabellen mithilfe von CTAS.

Berechtigungen

- Um Zugriffskontrollen durchzusetzen, geben EXPLAIN PLAN und DDL-Operationen wie DESCRIBE TABLE keine eingeschränkten Informationen preis.
- Wenn Sie einen Tabellenstandort bei Lake Formation registrieren, verwendet der Datenzugriff die in Lake Formation gespeicherten Anmeldeinformationen anstelle der IAM-Berechtigungen der EMR-Serverless-Joblaufzeitrolle. Jobs schlagen fehl, wenn die registrierte Rolle für den Tabellenspeicherort falsch konfiguriert ist, auch wenn die Runtime-Rolle über S3-IAM-Berechtigungen für diesen Speicherort verfügt.
- Ab Amazon EMR 7.12 können Sie mithilfe von DataFrameWriter (V2) mit Lake Formation Formation-Anmeldeinformationen im Anfügemodus in bestehende Hive- und Iceberg-Tabellen schreiben. Bei Überschreibvorgängen oder beim Erstellen neuer Tabellen verwendet EMR die Anmeldeinformationen der Runtime-Rolle, um Tabellendaten zu ändern.
- Die folgenden Einschränkungen gelten, wenn Ansichten oder zwischengespeicherte Tabellen als Quelldaten verwendet werden (diese Einschränkungen gelten nicht für AWS Glue Data Catalog-Ansichten):
  - Für MERGE-, DELETE- und UPDATE-Operationen
    - Unterstützt: Verwenden von Ansichten und zwischengespeicherten Tabellen als Quelltabellen.
    - Nicht unterstützt: Verwendung von Ansichten und zwischengespeicherten Tabellen in Zuweisungs- und Bedingungsklauseln.
  - Für die Operationen CREATE OR REPLACE und REPLACE TABLE AS SELECT:
    - Nicht unterstützt: Verwenden von Ansichten und zwischengespeicherten Tabellen als Quelltabellen.

- Delta Lake-Tabellen mit UDFs Quelldaten unterstützen MERGE-, DELETE- und UPDATE-Operationen nur, wenn der Löschvektor aktiviert ist.

Protokolle und Debugging

- EMR Serverless schränkt den Zugriff auf Systemtreiber-Spark-Protokolle für Lake Formation-fähige Anwendungen ein. Da der Systemtreiber mit erhöhten Rechten ausgeführt wird, können Ereignisse und Protokolle, die der Systemtreiber generiert, vertrauliche Informationen enthalten. Um zu verhindern, dass unbefugte Benutzer oder Code auf diese sensiblen Daten zugreifen, deaktiviert EMR Serverless den Zugriff auf Systemtreiberprotokolle.
- Systemprofilprotokolle werden immer im verwalteten Speicher gespeichert — dies ist eine obligatorische Einstellung, die nicht deaktiviert werden kann. Diese Protokolle werden sicher gespeichert und entweder mit einem vom Kunden verwalteten KMS-Schlüssel oder einem AWS verwalteten KMS-Schlüssel verschlüsselt.

Iceberg

Beachten Sie die folgenden Überlegungen bei der Verwendung von Apache Iceberg:

- Sie können Apache Iceberg nur mit Sitzungskatalogen und nicht mit beliebig benannten Katalogen verwenden.
- Iceberg-Tabellen, die in Lake Formation registriert sind, unterstützen nur die Metadatentabellen `history`,`metadata_log_entries`,`snapshots`,`files`,`manifests`, und`refs`. Amazon EMR blendet die Spalten aus, die möglicherweise vertrauliche Daten wie `partitions`, und enthalten. `path` `summaries` Diese Einschränkung gilt nicht für Iceberg-Tabellen, die nicht in Lake Formation registriert sind.
- Tabellen, die nicht in Lake Formation registriert sind, unterstützen alle gespeicherten Iceberg-Prozeduren. Die Prozeduren `register_table` und `migrate` werden für keine Tabellen unterstützt.
- Wir empfehlen, dass Sie Iceberg DataFrameWriter V2 anstelle von V1 verwenden.

Fehlerbehebung

In den folgenden Abschnitten finden Sie Lösungen zur Fehlerbehebung.

Protokollierung

EMR Serverless verwendet Spark-Ressourcenprofile, um die Auftragsausführung aufzuteilen. EMR Serverless verwendet das Benutzerprofil, um den von Ihnen bereitgestellten Code auszuführen, während das Systemprofil die Lake Formation-Richtlinien durchsetzt. Sie können auf die Protokolle für die Aufgaben zugreifen, die unter dem Benutzerprofil ausgeführt wurden.

Weitere Informationen zum Debuggen von Lake Formation-fähigen Jobs finden Sie unter Debuggen von Jobs.

Live-UI und Spark History Server

In der Live-UI und auf dem Spark History Server werden alle Spark-Ereignisse aus dem Benutzerprofil und aus dem Systemtreiber redigierte Ereignisse generiert.

Sie können alle Aufgaben der Benutzer- und Systemtreiber auf der Registerkarte Executors sehen. Protokolllinks sind jedoch nur für das Benutzerprofil verfügbar. Außerdem werden einige Informationen aus der Live-UI redigiert, z. B. die Anzahl der Ausgabedatensätze.

Auftrag schlägt fehl, weil die Lake-Formation-Berechtigungen unzureichend sind

Stellen Sie sicher, dass Ihre Auftrag-Laufzeitrolle über die Berechtigungen zum Ausführen von SELECT und DESCRIBE für die Tabelle verfügt, auf die Sie zugreifen.

Auftrag mit RDD-Ausführung ist fehlgeschlagen

EMR Serverless unterstützt derzeit keine Resilient Distributed Dataset (RDD) -Operationen für Lake Formation-fähige Jobs.

Auf Datendateien in Amazon S3 kann nicht zugegriffen werden

Stellen Sie sicher, dass Sie den Data-Lake-Standort in Lake Formation registriert haben.

Sicherheitsvalidierungsausnahme

EMR Serverless hat einen Fehler bei der Sicherheitsüberprüfung festgestellt. Wenden Sie sich für AWS weitere Unterstützung an den Support.

Gemeinsame Nutzung des AWS Glue-Datenkatalogs und der Tabellen für mehrere Konten

Sie können Datenbanken und Tabellen für mehrere Konten freigeben und trotzdem Lake Formation verwenden. Weitere Informationen finden Sie unter Kontoübergreifender Datenaustausch in Lake

Formation und Wie teile ich den AWS Glue-Datenkatalog und die Tabellen kontenübergreifend? AWS Lake Formation.

Die native, feinkörnige Zugriffskontrolle von Spark erlaubt die unten gelistete API PySpark

Um die Sicherheit und die Datenzugriffskontrollen aufrechtzuerhalten, schränkt Spark Fine-Grained Access Control (FGAC) bestimmte Funktionen ein. PySpark Diese Einschränkungen werden durchgesetzt durch:

- Explizites Blockieren, das die Funktionsausführung verhindert
- Architekturinkompatibilitäten, die Funktionen funktionsunfähig machen
- Funktionen, die Fehler auslösen, Meldungen zurückgeben, bei denen der Zugriff verweigert wurde, oder die beim Aufruf nichts bewirken

Die folgenden PySpark Funktionen werden in Spark FGAC nicht unterstützt:

- RDD-Operationen (mit Spark-Ausnahme blockiert) RDDUnsupported
- Spark Connect (nicht unterstützt)
- Spark Streaming (nicht unterstützt)

Wir haben die aufgelisteten Funktionen zwar in einer nativen Spark-FGAC-Umgebung getestet und bestätigt, dass sie erwartungsgemäß funktionieren, aber unsere Tests decken in der Regel nur die grundlegende Nutzung der einzelnen APIs ab. Für Funktionen mit mehreren Eingabetypen oder komplexen Logikpfaden gibt es möglicherweise noch nicht getestete Szenarien.

Für alle Funktionen, die hier nicht aufgeführt sind und nicht eindeutig zu den oben genannten, nicht unterstützten Kategorien gehören, empfehlen wir:

- Testen Sie sie zuerst in einer Gamma-Umgebung oder in kleinem Maßstab
- Überprüfung ihres Verhaltens, bevor sie in der Produktion eingesetzt werden

> **Note**
>
> Wenn eine Klassenmethode aufgeführt ist, aber nicht ihre Basisklasse, sollte die Methode trotzdem funktionieren. Das bedeutet nur, dass wir den Basisklassenkonstruktor nicht explizit verifiziert haben.

Die PySpark API ist in Module unterteilt. Die allgemeine Unterstützung für Methoden innerhalb der einzelnen Module ist in der folgenden Tabelle detailliert beschrieben.

Module name (Modulname)	Status	Hinweise
pyspark_core	Unterstützt	Dieses Modul enthält die wichtigsten RDD-Klassen, und diese Funktionen werden meistens nicht unterstützt.
pyspark_sql	Unterstützt	
pyspark_testing	Unterstützt	
pyspark_resource	Unterstützt	
pyspark_streaming	Blocked	Die Streaming-Nutzung ist in Spark FGAC blockiert.
pyspark_mllib	Experimentell	Dieses Modul enthält RDD-basierte ML-Operationen, und diese Funktionen werden größtenteils nicht unterstützt. Dieses Modul wurde nicht gründlich getestet.
pyspark_ml	Experimentell	Dieses Modul enthält DataFrame basierte

Module name (Modulname)	Status	Hinweise
		ML-Operationen, und diese Funktionen werden größtenteils unterstützt. Dieses Modul wurde nicht gründlich getestet.
pyspark_pandas	Unterstützt	
pyspark_pandas_langsam	Unterstützt	
pyspark_connect	Blocked	Die Nutzung von Spark Connect ist in Spark FGAC blockiert.
pyspark_pandas_connect	Blocked	Die Nutzung von Spark Connect ist in Spark FGAC blockiert.
pyspark_pandas_slow_connect	Blocked	Die Nutzung von Spark Connect ist in Spark FGAC blockiert.
pyspark_errors	Experimentell	Dieses Modul wurde nicht gründlich getestet. Benutzerdefinierte Fehlerklassen können nicht verwendet werden.

API-Zulassungsliste

Für eine herunterladbare und einfacher zu durchsuchende Liste ist eine Datei mit den Modulen und Klassen unter Python-Funktionen verfügbar, die in Native FGAC erlaubt sind.

Verschlüsselung zwischen Mitarbeitern

Mit Amazon EMR-Versionen 6.15.0 und höher können Sie die mit Mutual-TLS verschlüsselte Kommunikation zwischen Workern in Ihren Spark-Jobläufen aktivieren. Wenn diese Option

aktiviert ist, generiert und verteilt EMR Serverless automatisch ein eindeutiges Zertifikat für jeden Worker, der im Rahmen Ihrer Jobläufe bereitgestellt wird. Wenn diese Worker kommunizieren, um Kontrollnachrichten auszutauschen oder Shuffle-Daten zu übertragen, stellen sie eine gegenseitige TLS-Verbindung her und verwenden die konfigurierten Zertifikate, um die Identität der anderen Mitarbeiter zu überprüfen. Wenn ein Worker kein anderes Zertifikat verifizieren kann, schlägt der TLS-Handshake fehl und EMR Serverless bricht die Verbindung zwischen ihnen ab.

Wenn Sie Lake Formation mit EMR Serverless verwenden, ist die Mutual-TLS-Verschlüsselung standardmäßig aktiviert.

Aktivierung der Mutual-TLS-Verschlüsselung auf EMR Serverless

Um die gegenseitige TLS-Verschlüsselung in Ihrer Spark-Anwendung zu aktivieren, setzen Sie den Wert `spark.ssl.internode.enabled` auf true, wenn Sie die EMR Serverless-Anwendung erstellen. Wenn Sie die AWS Konsole verwenden, um eine serverlose EMR-Anwendung zu erstellen, wählen Sie Benutzerdefinierte Einstellungen verwenden, erweitern Sie dann Anwendungskonfiguration und geben Sie Ihre ein. `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"
```

Wenn Sie die gegenseitige TLS-Verschlüsselung für einzelne Spark-Jobausführungen aktivieren möchten, setzen Sie `spark.ssl.internode.enabled` diese Option bei der Verwendung auf true. `spark-submit`

```
--conf spark.ssl.internode.enabled=true
```

Festplattenverschlüsselung mit KMS CMK

EMR Serverless verschlüsselt standardmäßig alle Festplatten, die an Worker angeschlossen sind, mithilfe von diensteigenen Verschlüsselungsschlüsseln. Sie können diese Festplatten optional mit Ihren eigenen, vom AWS KMS Kunden verwalteten Schlüsseln () verschlüsseln. CMKs Auf diese Weise haben Sie mehr Kontrolle über Ihre Verschlüsselungsschlüssel, einschließlich der Möglichkeit, wichtige Richtlinien festzulegen und zu verwalten und die Verwendung von Schlüsseln zu überprüfen.

Sie können die Festplattenverschlüsselung entweder bei der Erstellung einer Anwendung oder beim Senden einzelner Jobs konfigurieren. Wenn sie auf Anwendungsebene aktiviert ist, erben alle Jobs in dieser Anwendung die Verschlüsselungseinstellungen. Sie können auch die Standardeinstellung der Anwendung außer Kraft setzen, indem Sie beim Absenden eines Jobs eine Festplattenverschlüsselungskonfiguration angeben.

> **Note**
>
> EMR Serverless Disk Encryption unterstützt nur symmetrische KMS-Schlüssel. Asymmetrische KMS-Schlüssel werden nicht unterstützt. Sie müssen einen KMS-Schlüssel für die symmetrische Verschlüsselung verwenden, der in erstellt wurde. AWS KMS Weitere Informationen zu finden Sie AWS KMS unter Was ist AWS KMS?

Verschlüsselungskontext verwenden

Optional verwendet EMR Serverless den Verschlüsselungskontext, um zusätzliche authentifizierte Daten für Verschlüsselungsvorgänge bereitzustellen. Der Verschlüsselungskontext besteht aus einer Reihe von Schlüssel-Wert-Paaren, die nicht geheime zusätzliche authentifizierte Daten enthalten können. Der Verschlüsselungskontext ist kryptografisch an die verschlüsselten Daten gebunden, sodass für die Entschlüsselung der Daten derselbe Verschlüsselungskontext erforderlich ist.

In EMR Serverless können Sie den benutzerdefinierten Verschlüsselungskontext bei der Konfiguration der Festplattenverschlüsselung angeben. Dieser Verschlüsselungskontext ist in AWS CloudTrail Protokollen enthalten, um Ihnen zu helfen, Ihre KMS-Operationen zu identifizieren und zu verstehen.

> **Note**
>
> Speichern Sie vertrauliche Informationen nicht im Verschlüsselungskontext, da sie in AWS CloudTrail Protokollen als Klartext erscheinen.

Konfiguration der Festplattenverschlüsselung mit vom Kunden verwalteten Schlüsseln

CreateApplication

Um Festplatten mit Ihrem eigenen KMS-Schlüssel zu verschlüsseln, fügen Sie den `diskEncryptionConfiguration` Parameter bei der Erstellung einer EMR Serverless-Anwendung hinzu.

```
aws emr-serverless create-application \
  --type TYPE \
  --name APPLICATION_ID \
  --release-label RELEASE_LABEL \
  --region AWS_REGION \
  --disk-encryption-configuration '{
        "encryptionKeyArn": "key-arn",
        "encryptionContext": {
            "key": "value"
        }
  }'
```

UpdateApplication

Um den and/or ARN-Verschlüsselungskontext für den KMS-Schlüssel zu aktualisieren, geben Sie beim Aktualisieren einer Anwendung den `diskEncryptionConfiguration` Parameter mit den neuen Werten an.

```
aws emr-serverless update-application \
  --name APPLICATION_ID \
  --region AWS_REGION \
  --disk-encryption-configuration '{
        "encryptionKeyArn": "key-arn",
        "encryptionContext": {
            "key": "value"
        }
  }'
```

> **Note**
>
> Um die konfigurierte Festplattenverschlüsselung für eine Anwendung aufzuheben, übergeben Sie `diskEncryptionConfiguration` beim Aktualisieren der Anwendung einen leeren Wert.

StartJobRun

Um Festplatten mit Ihrem eigenen KMS-Schlüssel zu verschlüsseln, verwenden Sie die `diskEncryptionConfiguration` Konfiguration, wenn Sie eine Jobausführung einreichen.

```
--configuration-overrides '{
        "diskEncryptionConfiguration": {
            "encryptionKeyArn": "key-arn",
            "encryptionContext": {
                "key": "value"
            }
        }
    }'
```

Öffentliche Livy-Endpunkte

Um Festplatten mit Ihrem eigenen KMS-Schlüssel zu verschlüsseln, wenn Sie Spark-Sitzungen über öffentliche Livy-Endpunkte erstellen, geben Sie die Verschlüsselungskonfiguration im Sitzungsobjekt an. `conf`

```
data = {
    "kind": "pyspark",
    "heartbeatTimeoutInSecond": 60,
    "conf": {
        "emr-serverless.session.executionRoleArn": "role_arn",
        "spark.emr-serverless.disk.encryptionKeyArn": "key-arn",
        "spark.emr-serverless.disk.encryptionContext": "key1:value1,key2:value2"  #
 Optional
    }
}

# Send request to create a session with the Livy API endpoint
request = AWSRequest(method='POST', url=endpoint + "/sessions", data=json.dumps(data),
 headers=headers)
```

Erforderliche Berechtigungen für die Festplattenverschlüsselung

Verschlüsselungsschlüsselberechtigungen für EMR Serverless

Wenn Sie Festplatten mit Ihrem eigenen Verschlüsselungsschlüssel verschlüsseln, müssen Sie die folgenden KMS-Schlüsselberechtigungen für den Prinzipal konfigurieren: `emr-serverless.amazonaws.com`

- `kms:GenerateDataKey`: Um Datenschlüssel für die Verschlüsselung von Festplattenvolumes zu generieren
- `kms:Decrypt`: Zum Entschlüsseln von Datenschlüsseln beim Zugriff auf verschlüsselte Festplatteninhalte

```
{
    "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"
        },
        "StringEquals": {
            "kms:EncryptionContext:applicationId": "application-id",
            "aws:SourceAccount": "aws-account-id"
        }
    }
}
```

Aus Sicherheitsgründen empfehlen wir, der KMS-Schlüsselrichtlinie einen `aws:SourceArn` Bedingungsschlüssel hinzuzufügen. Der globale IAM-Bedingungsschlüssel `aws:SourceArn` trägt dazu bei, dass EMR Serverless den KMS-Schlüssel nur für einen Anwendungs-ARN verwendet. Darüber hinaus bietet die Aufnahme des `aws:SourceAccount` Bedingungsschlüssels eine weitere

Sicherheitsebene, da die Verwendung Ihres KMS-Schlüssels auf Anfragen beschränkt wird, die von der in der Bedingung angegebenen AWS Konto-ID stammen.

Die IAM-Richtlinie der Job-Runtime-Rolle muss über die folgenden Berechtigungen verfügen:

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

Erforderliche Benutzerberechtigungen

Der Benutzer, der den Job einreicht, muss über die Berechtigungen zur Verwendung des Schlüssels verfügen. Sie können die Berechtigungen entweder in der KMS-Schlüsselrichtlinie oder in der IAM-Richtlinie für den Benutzer, die Gruppe oder die Rolle angeben. Wenn der Benutzer, der den Job einreicht, nicht über die KMS-Schlüsselberechtigungen verfügt, lehnt EMR Serverless die Übermittlung der Auftragsausführung ab.

Beispiel für eine Schlüsselrichtlinie

Die folgende wichtige Richtlinie bietet die Berechtigungen für und: `kms:DescribeKey` `kms:GenerateDataKey` `kms:Decrypt`

- `kms:DescribeKey`: Um zu überprüfen, ob der vom Kunden verwaltete KMS-Schlüssel aktiviert und SYMMETRISCH ist, bevor Sie ihn verwenden.

```
{
    "Sid": "Enable DescribeKey",
    "Effect": "Allow",
    "Principal":{
        "AWS": "arn:aws:iam::111122223333:user/user-name"
    },
    "Action": [
```

```
        "kms:DescribeKey"
    ],
    "Resource": "*"
},
{
    "Sid": "Enable GDK and Decrypt",
    "Effect": "Allow",
    "Principal":{
        "AWS": "arn:aws:iam::111122223333:user/user-name"
    },
    "Action": [
        "kms:GenerateDataKey",
        "kms:Decrypt"
    ],
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "kms:ViaService": "emr-serverless.region.amazonaws.com",
            "kms:EncryptionContext:key": "value"
        }
    }
}
```

Aus Sicherheitsgründen empfehlen wir, der KMS-Schlüsselrichtlinie einen `kms:viaService` Bedingungsschlüssel hinzuzufügen. Dadurch wird die Verwendung des KMS-Schlüssels auf Überprüfungsanforderungen beschränkt, die nur von EMR-serverlos gesendet werden.

Beispiel für eine IAM-Richtlinie

Die folgende IAM-Richtlinie bietet die Berechtigungen für, und. `kms:DescribeKey` `kms:GenerateDataKey` `kms:Decrypt`

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

```
}
```

Überwachung der Schlüsselnutzung

Sie können die Verwendung Ihrer vom Kunden verwalteten Schlüssel in EMR Serverless über überwachen. AWS CloudTrail AWS CloudTrail erfasst alle API-Aufrufe AWS KMS als Ereignisse, einschließlich Aufrufe von der EMR Serverless-Konsole, der EMR Serverless API, der AWS CLI oder dem SDK. AWS

Zu den erfassten Informationen gehört der von Ihnen angegebene Verschlüsselungskontext, der Ihnen helfen kann, die spezifischen EMR-Serverless-Ressourcen zu identifizieren und zu prüfen, die Ihren KMS-Schlüssel verwendet haben. Möglicherweise finden Sie unter Ereignisse, die den folgenden ähneln. AWS CloudTrail Weitere Informationen zur Verwendung AWS CloudTrail finden Sie im AWS CloudTrail Benutzerhandbuch.

GenerateDataKey

Beispielereignis für GenerateDataKey Operationen, bei denen EMR Serverless verschlüsselte Festplattenvolumes erstellt

```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "type": "AWSService",
        "principalId": "user",
        "invokedBy": "AWS Internal"
    },
    "eventTime": "2025-07-28T21:43:51Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "ipAddress",
    "userAgent": "userAgent",
    "requestParameters": {
        "encryptionContext": {
            "applicationId": "test"
        },
        "keyId": "arn:aws:kms:region:accountId:key/ffffffff-fffff-aaaaa-eeee-sample",
        "keySpec": "AES_256"
    },
    "responseElements": null,
```

```
    "additionalEventData": {
        "keyMaterialId":
 "145c963debe558dfb01848d2a4539da940f3478852f86cfe2f52d5df796a5a02"
    },
    "requestID": "cc9d1c5e-97c4-4a4f-ae7a-e576sample",
    "eventID": "0b0fef09-f28d-4da8-a5a1-17b74sample",
    "readOnly": true,
    "resources": [
        {
            "accountId": "account",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:region:accountId:key/ffffffff-fffff-aaaaa-eeee-sample"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "accountId",
    "eventCategory": "Management"
}
```

Decrypt

Beispielereignis für Decrypt-Operationen, wenn EMR Serverless auf verschlüsselte Daten zugreift.

```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "type": "AWSService",
        "principalId": "user",
        "invokedBy": "AWS Internal"
    },
    "eventTime": "2025-07-28T21:43:51Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "ipAddress",
    "userAgent": "userAgent",
    "requestParameters": {
        "encryptionContext": {
            "applicationId": "test"
        },
        "keyId": "arn:aws:kms:region:accountId:key/ffffffff-fffff-aaaaa-eeee-sample",
        "keySpec": "AES_256"
```

```
    },
    "responseElements": null,
    "additionalEventData": {
        "keyMaterialId":
 "145c963debe558dfb01848d2a4539da940f3478852f86cfe2f52d5df796a5a02"
    },
    "requestID": "cc9d1c5e-97c4-4a4f-ae7a-e576sample",
    "eventID": "0b0fef09-f28d-4da8-a5a1-17b74sample",
    "readOnly": true,
    "resources": [
        {
            "accountId": "account",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:region:accountId:key/ffffffff-fffff-aaaaa-eeee-sample"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "accountId",
    "eventCategory": "Management"
}
```

Weitere Informationen

Die folgenden Ressourcen enthalten weitere Informationen zur Datenverschlüsselung im Ruhezustand:

- Weitere Informationen zu AWS KMS grundlegenden Konzepten finden Sie im Entwicklerhandbuch.AWS KMS
- Weitere Informationen zu bewährten Sicherheitsmethoden für AWS KMS finden Sie im AWS KMS Entwicklerhandbuch.

Secrets Manager für Datenschutz mit EMR Serverless

AWS Secrets Manager ist ein geheimer Speicherdienst zum Schutz von Datenbankanmeldedaten, API-Schlüsseln und anderen geheimen Informationen. Ersetzen Sie dann in Ihrem Code hartcodierte Anmeldeinformationen durch einen API-Aufruf an Secrets Manager. Auf diese Weise wird sichergestellt, dass das Geheimnis nicht von jemandem kompromittiert werden kann, der Ihren Code untersucht, da das Geheimnis nicht vorhanden ist. Eine Übersicht finden Sie im AWS Secrets Manager Benutzerhandbuch.

Secrets Manager verschlüsselt Geheimnisse mithilfe von AWS Key Management Service Schlüsseln. Weitere Informationen finden Sie unter Secret Ver- und Entschlüsselung im AWS Secrets Manager Benutzerhandbuch.

Sie können Secrets Manager so konfigurieren, dass Secrets automatisch nach einem von Ihnen angegebenen Zeitplan für Sie rotiert werden. So können Sie Secrets mit langer Einsatzdauer durch Secrets mit kurzer Einsatzdauer ersetzen und damit das Risiko einer Kompromittierung erheblich verringern. Weitere Informationen finden Sie unter Rotation von AWS Secrets Manager Geheimnissen im AWS Secrets Manager Benutzerhandbuch.

Amazon EMR Serverless lässt sich integrieren, AWS Secrets Manager sodass Sie Ihre Daten in Secrets Manager speichern und die geheime ID in Ihren Konfigurationen verwenden können.

So verwendet EMR Serverless Secrets

Wenn Sie Ihre Daten in Secrets Manager speichern und die geheime ID in Ihren Konfigurationen für EMR Serverless verwenden, geben Sie vertrauliche Konfigurationsdaten nicht im Klartext an EMR Serverless weiter und stellen sie extern zur Verfügung. APIs Wenn Sie angeben, dass ein Schlüssel-Wert-Paar eine geheime ID für ein Geheimnis enthält, das Sie in Secrets Manager gespeichert haben, ruft EMR Serverless das Geheimnis ab, wenn es Konfigurationsdaten an Worker sendet, um Jobs auszuführen.

Um anzugeben, dass ein Schlüssel-Wert-Paar für eine Konfiguration einen Verweis auf ein in Secrets Manager gespeichertes Geheimnis enthält, fügen Sie die `EMR.secret@` Anmerkung zum Konfigurationswert hinzu. Für jede Konfigurationseigenschaft mit geheimer ID-Anmerkung ruft EMR Serverless Secrets Manager auf und löst das Geheimnis zum Zeitpunkt der Jobausführung auf.

Wie erstellt man ein Geheimnis

Um ein Geheimnis zu erstellen, folgen Sie den Schritten unter Ein AWS Secrets Manager Geheimnis erstellen im AWS Secrets Manager Benutzerhandbuch. Wählen Sie in Schritt 3 das Feld Klartext aus, um Ihren vertraulichen Wert einzugeben.

Geben Sie ein Geheimnis in einer Konfigurationsklassifizierung ein

Die folgenden Beispiele zeigen, wie Sie ein Geheimnis in einer Konfigurationsklassifizierung unter angeben. `StartJobRun` Wenn Sie Klassifizierungen für Secrets Manager auf Anwendungsebene konfigurieren möchten, finden Sie weitere Informationen unter. Standardanwendungskonfiguration für EMR Serverless

Ersetzen Sie es in den Beispielen *SecretName* durch den Namen des abzurufenden Geheimnisses. Weitere Informationen finden Sie unter Wie erstellt man ein Geheimnis.

In diesem Abschnitt

Geben Sie geheime Referenzen an — Spark

Example— Geben Sie geheime Referenzen in der externen Hive-Metastore-Konfiguration für Spark an

```
aws emr-serverless start-job-run \
  --application-id "application-id" \
  --execution-role-arn "job-role-arn" \
  --job-driver '{
        "sparkSubmit": {
            "entryPoint": "s3://amzn-s3-demo-bucket/scripts/spark-jdbc.py",
            "sparkSubmitParameters": "--jars s3://amzn-s3-demo-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://amzn-s3-demo-bucket/spark/logs/"
        }
    }
}'
```

Example— Geben Sie geheime Referenzen für die externe Hive-Metastore-Konfiguration in der Klassifizierung an 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",
        }
    }
```

Geben Sie geheime Referenzen an — Hive

Example— Geben Sie geheime Referenzen in der externen Hive-Metastore-Konfiguration für Hive an

```
aws emr-serverless start-job-run \
  --application-id "application-id" \
  --execution-role-arn "job-role-arn" \
    --job-driver '{
        "hive": {
        "query": "s3://amzn-s3-demo-bucket/emr-serverless-hive/query/hive-query.ql",
        "parameters": "--hiveconf hive.exec.scratchdir=s3://amzn-s3-demo-bucket/emr-
serverless-hive/hive/scratch
                    --hiveconf hive.metastore.warehouse.dir=s3://amzn-s3-demo-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://amzn-s3-demo-bucket"
        }
    }
}'
```

Example— Geben Sie geheime Referenzen für die externe Hive-Metastore-Konfiguration in der Klassifizierung an 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"
    }
}
```

Gewähren Sie EMR Serverless Zugriff, um das Geheimnis abzurufen

Damit EMR Serverless den geheimen Wert aus Secrets Manager abrufen kann, fügen Sie Ihrem Secret bei der Erstellung die folgende Richtlinienanweisung hinzu. Sie müssen Ihr Geheimnis mit dem vom Kunden verwalteten KMS-Schlüssel erstellen, damit EMR Serverless den geheimen Wert lesen kann. Weitere Informationen finden Sie im Benutzerhandbuch unter Berechtigungen für den KMS-Schlüssel.AWS Secrets Manager

Ersetzen Sie den Wert in der folgenden Richtlinie *`applicationId`* durch die ID für Ihre Anwendung.

Ressourcenrichtlinie für das Geheimnis

Sie müssen die folgenden Berechtigungen in die Ressourcenrichtlinie für das Secret in aufnehmen AWS Secrets Manager , damit EMR Serverless geheime Werte abrufen kann. Um sicherzustellen, dass nur eine bestimmte Anwendung dieses Geheimnis abrufen kann, können Sie optional die EMR Serverless Application ID als Bedingung in der Richtlinie angeben.

JSON

```
{
```

```
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue",
        "secretsmanager:DescribeSecret"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:SourceArn": "arn:aws:emr-serverless:*:123456789012:/applications/
*"
        }
      },
      "Sid": "AllowSECRETSMANAGERGetsecretvalue"
    }
  ]
}
```

Erstellen Sie Ihr Geheimnis mit der folgenden Richtlinie für den vom Kunden verwalteten Schlüssel AWS Key Management Service ()AWS KMS:

Richtlinie für den vom Kunden verwalteten Schlüssel AWS KMS

```
{
    "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.AWS-Region.amazonaws.com"
        }
    }
}
```

Das Geheimnis rotieren

Bei der Rotation aktualisieren Sie ein Geheimnis regelmäßig. Sie können so konfigurieren AWS Secrets Manager , dass das Geheimnis nach einem von Ihnen festgelegten Zeitplan automatisch für Sie rotiert wird. Auf diese Weise können Sie langfristige Geheimnisse durch kurzfristige ersetzen. Dies trägt dazu bei, das Risiko von Kompromissen zu verringern. EMR Serverless ruft den geheimen Wert aus einer mit Anmerkungen versehenen Konfiguration ab, wenn der Job in den Status Running übergeht. Wenn Sie oder ein Prozess den geheimen Wert in Secrets Manager aktualisieren, müssen Sie einen neuen Job einreichen, damit der Job den aktualisierten Wert abrufen kann.

> **Note**
>
> Jobs, die sich bereits im Status Running befinden, können keinen aktualisierten geheimen Wert abrufen. Dies kann zum Fehlschlagen des Auftrags führen.

Verwenden von Amazon S3 Access Grants mit EMR Serverless

Übersicht über S3 Access Grants für EMR Serverless

Mit den Amazon EMR-Versionen 6.15.0 und höher bieten Amazon S3 Access Grants eine skalierbare Zugriffskontrolllösung, mit der Sie den Zugriff auf Ihre Amazon S3 S3-Daten von EMR Serverless aus erweitern können. Wenn Sie über eine komplexe oder umfangreiche Berechtigungskonfiguration für Ihre S3-Daten verfügen, verwenden Sie Access Grants, um S3-Datenberechtigungen für Benutzer, Rollen und Anwendungen zu skalieren.

Verwenden Sie S3 Access Grants, um den Zugriff auf Amazon S3 S3-Daten über die Berechtigungen hinaus zu erweitern, die durch die Runtime-Rolle oder die IAM-Rollen gewährt wurden, die den Identitäten mit Zugriff auf Ihre EMR Serverless-Anwendung zugewiesen sind.

Weitere Informationen finden Sie unter Managing access with S3 Access Grants for Amazon EMR im Amazon EMR Management Guide und Managing access with S3 Access Grants im Amazon Simple Storage Service User Guide.

In diesem Abschnitt wird beschrieben, wie Sie eine serverlose EMR-Anwendung starten, die S3 Access Grants verwendet, um Zugriff auf Daten in Amazon S3 zu gewähren. Schritte zur Verwendung von S3 Access Grants mit anderen Amazon-EMR-Bereitstellungen finden Sie in der folgenden Dokumentation:

- Verwenden von S3 Access Grants mit Amazon EMR
- Verwenden von S3 Access Grants mit Amazon EMR in EKS

Starten Sie eine serverlose EMR-Anwendung mit S3 Access Grants für die Datenverwaltung

Sie können S3 Access Grants auf EMR Serverless aktivieren und eine Spark-Anwendung starten. Wenn Ihre Anwendung eine Anfrage für S3-Daten stellt, stellt Amazon S3 temporäre Anmeldeinformationen bereit, die auf den jeweiligen Bucket, das Präfix oder das Objekt beschränkt sind.

1. Richten Sie eine Jobausführungsrolle für Ihre EMR Serverless-Anwendung ein. Geben Sie die erforderlichen IAM-Berechtigungen an, sodass Sie Spark-Jobs ausführen und S3 Access Grants verwenden müssen, und: `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**
>
> Wenn Sie IAM-Rollen für die Auftragsausführung angeben, die über zusätzliche Berechtigungen für den direkten Zugriff auf S3 verfügen, können Benutzer auf die durch

die Rolle zugelassenen Daten zugreifen, auch wenn sie keine Genehmigung von S3 Access Grants haben.

2. Starten Sie Ihre EMR Serverless-Anwendung mit dem Amazon EMR-Release-Label 6.15.0 oder höher und der `spark-defaults` Klassifizierung, wie das folgende Beispiel zeigt. Ersetzen Sie die Werte in *`red text`* mit den passenden Werten für Ihr Nutzungsszenario.

```
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://amzn-s3-demo-destination-bucket1/
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"
         }
      }]
}'
```

Überlegungen zu S3 Access Grants mit EMR Serverless

Wichtige Informationen zu Support, Kompatibilität und Verhalten bei der Verwendung von Amazon S3 Access Grants mit EMR Serverless finden Sie unter Überlegungen zu S3 Access Grants with Amazon EMR im Amazon EMR Management Guide.

Protokollieren von Amazon EMR Serverless API-Aufrufen mit AWS CloudTrail

Amazon EMR Serverless ist in einen Service integriert AWS CloudTrail, der eine Aufzeichnung der Aktionen bereitstellt, die von einem Benutzer, einer Rolle oder einem AWS Service in EMR Serverless ausgeführt wurden. CloudTrail erfasst alle API-Aufrufe für EMR Serverless als Ereignisse. Zu den erfassten Aufrufen gehören Aufrufe von der EMR Serverless-Konsole und Codeaufrufen für die EMR Serverless API-Operationen. Wenn Sie einen Trail erstellen, aktivieren Sie die kontinuierliche Übermittlung von CloudTrail Ereignissen an einen Amazon S3 S3-Bucket, einschließlich Ereignissen für EMR Serverless. Wenn Sie keinen Trail konfigurieren, können Sie trotzdem auf die neuesten Ereignisse in der CloudTrail Konsole unter Ereignisverlauf zugreifen. Anhand der von gesammelten Informationen können Sie die Anfrage CloudTrail, die an EMR Serverless gestellt wurde, die IP-Adresse, von der aus die Anfrage gestellt wurde, wer die Anfrage gestellt hat, wann sie gestellt wurde, und weitere Details ermitteln.

Weitere Informationen finden Sie im AWS CloudTrail Benutzerhandbuch.

EMR Serverlose Informationen in CloudTrail

CloudTrail ist auf Ihrem aktiviert AWS-Konto , wenn Sie das Konto erstellen. Wenn in EMR Serverless eine Aktivität auftritt, wird diese Aktivität zusammen mit anderen AWS Dienstereignissen im CloudTrail Ereignisverlauf in einem Ereignis aufgezeichnet. In Ihrem können Sie auf aktuelle Ereignisse zugreifen, diese suchen und herunterladen. AWS-Konto Weitere Informationen finden Sie unter Ereignisse mit CloudTrail Ereignisverlauf anzeigen.

Für eine fortlaufende Aufzeichnung der Ereignisse in Ihrem AWS-Konto, einschließlich Ereignissen für EMR Serverless, erstellen Sie einen Trail. Ein Trail ermöglicht CloudTrail die Übermittlung von Protokolldateien an einen Amazon S3 S3-Bucket. Wenn Sie einen Trail in der Konsole anlegen, gilt dieser für alle AWS-Regionen-Regionen. Der Trail protokolliert Ereignisse aus allen Regionen der AWS Partition und übermittelt die Protokolldateien an den von Ihnen angegebenen Amazon S3 S3-Bucket. Konfigurieren Sie außerdem andere AWS Dienste, um die in den CloudTrail Protokollen gesammelten Ereignisdaten weiter zu analysieren und darauf zu reagieren. Weitere Informationen finden Sie im Folgenden:

- Übersicht zum Erstellen eines Trails
- CloudTrail unterstützte Dienste und Integrationen
- Konfiguration von Amazon SNS SNS-Benachrichtigungen für CloudTrail

- Empfangen von CloudTrail Protokolldateien aus mehreren Regionen und Empfangen von CloudTrail Protokolldateien von mehreren Konten

Alle EMR Serverless-Aktionen werden von der EMR Serverless API Reference protokolliert CloudTrail und sind in dieser dokumentiert. Beispielsweise generieren Aufrufe von `StartJobRun` und `CancelJobRun` Aktionen Einträge in den `CreateApplication` Protokolldateien. CloudTrail

Jeder Ereignis- oder Protokolleintrag enthält Informationen zu dem Benutzer, der die Anforderung generiert hat. Die Identitätsinformationen unterstützen Sie bei der Ermittlung der folgenden Punkte:

- Ob die Anfrage mit Root- oder AWS Identity and Access Management (IAM-) Benutzeranmeldedaten gestellt wurde.
- Gibt an, ob die Anforderung mit temporären Sicherheitsanmeldeinformationen für eine Rolle oder einen Verbundbenutzer gesendet wurde.
- Ob die Anfrage von einem anderen AWS Dienst gestellt wurde.

Weitere Informationen finden Sie im CloudTrail UserIdentity-Element.

Grundlegendes zu serverlosen EMR-Protokolldateieinträgen

Ein Trail ist eine Konfiguration, die die Übertragung von Ereignissen als Protokolldateien an einen von Ihnen angegebenen Amazon S3 S3-Bucket ermöglicht. CloudTrail Protokolldateien enthalten einen oder mehrere Protokolleinträge. Ein Ereignis stellt eine einzelne Anforderung aus einer beliebigen Quelle dar und enthält Informationen über die angeforderte Aktion, Datum und Uhrzeit der Aktion, Anforderungsparameter usw. CloudTrail Protokolldateien sind kein geordneter Stack-Trace der öffentlichen API-Aufrufe, sodass sie nicht in einer bestimmten Reihenfolge angezeigt werden.

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag, der die `CreateApplication` Aktion demonstriert.

```
{
    "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"
}
```

Konformitätsprüfung für Amazon EMR Serverless

Die Sicherheit und Konformität von EMR Serverless wird von externen Prüfern im Rahmen mehrerer AWS Compliance-Programme bewertet, darunter die folgenden:

- System and Organization Controls (SOC)
- Payment Card Industry Data Security Standard (PCI DSS)
- Föderales Risiko- und Autorisierungsmanagementprogramm (FedRAMP) Moderat
- Health Insurance Portability and Accountability Act (HIPAA)

AWS bietet unter AWS Services in Scope by Compliance Program eine häufig aktualisierte Liste der AWS Services im Rahmen bestimmter Compliance-Programme.

Prüfberichte von Drittanbietern können Sie hier herunterladen AWS Artifact. Weitere Informationen finden Sie unter Berichte in AWS Artifact herunterladen.

Weitere Informationen zu AWS Compliance-Programmen finden Sie unter AWS Compliance-Programme.

Ihre Compliance-Verantwortung bei der Verwendung von EMR Serverless hängt von der Sensibilität Ihrer Daten, den Compliance-Zielen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften ab. Falls Ihre Nutzung von EMR Serverless der Einhaltung von Standards wie HIPAA, PCI oder FedRAMP Moderate unterliegt, bietet Ihnen folgende Ressourcen: AWS

- Kurzanleitungen zu Sicherheit und Compliance, in denen architektonische Überlegungen und Schritte für die Implementierung von sicherheits- und Compliance-orientierten Basisumgebungen erörtert werden. AWS
- AWS Mithilfe der Kundenleitfäden zur Einhaltung gesetzlicher Vorschriften können Sie das Modell der gemeinsamen Verantwortung aus der Sicht der Einhaltung von Vorschriften besser verstehen. In den Leitfäden werden die bewährten Methoden zum Schutz von AWS-Services zusammengefasst und die Leitlinien den Sicherheitskontrollen in verschiedenen Frameworks (einschließlich des National Institute of Standards and Technology (NIST), des Payment Card Industry Security Standards Council (PCI) und der International Organization for Standardization (ISO)) zugeordnet.
- AWS Config Mit können Sie bewerten, zu welchem Grad die Konfiguration Ihrer Ressourcen den internen Vorgehensweisen, Branchenrichtlinien und Vorschriften entspricht.

- AWS Compliance Resources ist eine Sammlung von Arbeitsmappen und Leitfäden, die möglicherweise auf Ihre Branche und Ihren Standort zutreffen.
- AWS Security Hub bietet Ihnen einen umfassenden Überblick über Ihren Sicherheitsstatus AWS und hilft Ihnen dabei, Ihre Einhaltung der Sicherheitsstandards und Best Practices der Sicherheitsbranche zu überprüfen.
- AWS Audit Manager— auf diese AWS-Service Weise können Sie Ihre AWS Nutzung kontinuierlich überprüfen, um den Umgang mit Risiken und die Einhaltung von Vorschriften und Industriestandards zu vereinfachen.

Ausfallsicherheit in Amazon EMR Serverless

Die AWS globale Infrastruktur basiert auf AWS Regionen und Availability Zones. AWS Regionen bieten mehrere physisch getrennte und isolierte Availability Zones, die über Netzwerke mit niedriger Latenz, hohem Durchsatz und hoher Redundanz miteinander verbunden sind. Mit Availability Zones können Sie Anwendungen und Datenbanken entwerfen und betreiben, die automatisch und ohne Unterbrechung ein Failover zwischen Zonen durchführen. Availability Zones sind besser hoch verfügbar, fehlertoleranter und skalierbarer als herkömmliche Infrastrukturen mit einem oder mehreren Rechenzentren.

Weitere Informationen zu AWS Regionen und Availability Zones finden Sie unter AWS Globale Infrastruktur.

Zusätzlich zur AWS globalen Infrastruktur bietet Amazon EMR Serverless die Integration mit Amazon S3 über EMRFS, um Ihre Anforderungen an Datenstabilität und Datensicherung zu erfüllen.

Infrastruktursicherheit in Amazon EMR Serverless

Als verwalteter Service ist Amazon EMR durch AWS globale Netzwerksicherheit geschützt. Informationen zu AWS Sicherheitsdiensten und zum AWS Schutz der Infrastruktur finden Sie unter AWS Cloud-Sicherheit. Informationen zum Entwerfen Ihrer AWS Umgebung unter Verwendung der bewährten Methoden für die Infrastruktursicherheit finden Sie unter Infrastructure Protection in Security Pillar AWS Well-Architected Framework.

Sie verwenden AWS veröffentlichte API-Aufrufe, um über das Netzwerk auf Amazon EMR zuzugreifen. Kunden müssen Folgendes unterstützen:

- Transport Layer Security (TLS). Wir benötigen TLS 1.2 und empfehlen TLS 1.3.

- Verschlüsselungs-Suiten mit Perfect Forward Secrecy (PFS) wie DHE (Ephemeral Diffie-Hellman) oder ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Die meisten modernen Systeme wie Java 7 und höher unterstützen diese Modi.

Konfiguration und Schwachstellenanalyse in Amazon EMR Serverless

AWS erledigt grundlegende Sicherheitsaufgaben wie das Patchen von Gastbetriebssystemen (OS) und Datenbanken, die Firewall-Konfiguration und die Notfallwiederherstellung. Diese Verfahren wurden von qualifizierten Dritten überprüft und zertifiziert. Weitere Informationen finden Sie in den folgenden Ressourcen:

- Konformitätsprüfung für Amazon EMR Serverless
- Modell der übergreifenden Verantwortlichkeit
- Amazon Web Services – Übersicht über Sicherheitsverfahren

Endpunkte und Kontingente für EMR Serverless

Service-Endpunkte

Um programmgesteuert eine Verbindung zu einem herzustellen AWS-Service, verwenden Sie einen Endpunkt. Ein Endpunkt ist die URL des Einstiegspunkts für einen AWS Webdienst. Zusätzlich zu den AWS Standardendpunkten AWS-Services bieten einige FIPS-Endpunkte in ausgewählten Regionen an. In der folgenden Tabelle sind die Dienstendpunkte für EMR Serverless aufgeführt. Weitere Informationen finden Sie unter Endpunkte.AWS-Service

Serverlose EMR-Servicendpunkte

Name der Region	Region	Endpunkt	Protocol (Protokoll)
USA Ost (Ohio)	`us-east-2` (beschränkt auf die folgenden Availability Zones:`use2-az1`, und`use2-az2`) `use2-az3`	`emr-serverless.us-east-2.amazonaws.com`	HTTPS

USA Ost (Nord-Virginia)	`us-east-1` (beschränkt auf die folgenden Availability Zones: `use1-az1``use1-az2`,`use1-az4`,`use1-az5`, und`use1-az6`)	`emr-serverless.us-east-1.amazonaws.com`
		
		`emr-serverless-fips.us-east-1.amazonaws.com`
			HTTPS

USA West (Nordkalifornien)	`us-west-1`	`emr-serverless.us-west-1.amazonaws.com`	HTTPS
USA West (Oregon)	`us-west-2`	`emr-serverless.us-`	HTTPS

Name der Region	Region	Endpunkt	Protocol (Protokoll)

		`west-2.amazonaws.com`
		
		`emr-serverless-fips.us-west-2.amazonaws.com`	

Afrika (Kapstadt)	`af-south-1`	`emr-serverless.af-south-1.amazonaws.com`	HTTPS
Asien-Pazifik (Hongkong)	`ap-east-1`	`emr-serverless.ap-east-1.amazonaws.com`	HTTPS
Asien-Pazifik (Jakarta)	`ap-southeast-3`	`emr-serverless.ap-southeast-3.amazonaws.com`	HTTPS
Asien-Pazifik (Melbourne)	`ap-southeast-4`	`emr-serverless.ap-southeast-4.amazonaws.com`	HTTPS
Asien-Pazifik (Malaysia)	`ap-southeast-5`	`emr-serverless.ap-southeast-5.amazonaws.com`	HTTPS

Name der Region	Region	Endpunkt	Protocol (Protokoll)
Asien-Pazifik (Mumbai)	ap-south-1	emr-serverless.ap-south-1.amazonaws.com	HTTPS
Asien-Pazifik (Osaka)	ap-northeast-3	emr-serverless.ap-northeast-3.amazonaws.com	HTTPS
Asien-Pazifik (Seoul)	ap-northeast-2	emr-serverless.ap-northeast-2.amazonaws.com	HTTPS
Asien-Pazifik (Singapur)	ap-southeast-1	emr-serverless.ap-southeast-1.amazonaws.com	HTTPS
Asien-Pazifik (Sydney)	ap-southeast-2	emr-serverless.ap-southeast-2.amazonaws.com	HTTPS
Asien-Pazifik (Tokio)	ap-northeast-1	emr-serverless.ap-northeast-1.amazonaws.com	HTTPS

Name der Region	Region	Endpunkt	Protocol (Protokoll)
Kanada (Zentral)	`ca-centra l-1` (beschränkt auf die folgenden Availability Zones: `cac1-az1` und`cac1-az2`)	`emr-serve rless.ca- central-1 .amazonaws.com`	HTTPS
Kanada West (Calgary)	`ca-west-1`	`emr-serve rless.ca- west-1.am azonaws.com`	HTTPS
Europa (Frankfurt)	`eu-central-1`	`emr-serve rless.eu- central-1 .amazonaws.com`	HTTPS
Europa (Zürich)	`eu-central-2`	`emr-serve rless.eu- central-2 .amazonaws.com`	HTTPS
Europa (Irland)	`eu-west-1`	`emr-serve rless.eu- west-1.am azonaws.com`	HTTPS
Europa (London)	`eu-west-2`	`emr-serve rless.eu- west-2.am azonaws.com`	HTTPS
Europa (Milan)	`eu-south-1`	`emr-serve rless.eu- south-1.a mazonaws.com`	HTTPS

Name der Region	Region	Endpunkt	Protocol (Protokoll)
Europa (Paris)	`eu-west-3`	`emr-serverless.eu-west-3.amazonaws.com`	HTTPS
Europa (Spain)	`eu-south-2`	`emr-serverless.eu-south-2.amazonaws.com`	HTTPS
Europa (Stockholm)	`eu-north-1`	`emr-serverless.eu-north-1.amazonaws.com`	HTTPS
Israel (Tel Aviv)	`il-central-1`	`emr-serverless.il-central-1.amazonaws.com`	HTTPS
Middle East (Bahrain)	`me-south-1`	`emr-serverless.me-south-1.amazonaws.com`	HTTPS
Naher Osten (VAE)	`me-central-1`	`emr-serverless.me-central-1.amazonaws.com`	HTTPS
Südamerika (São Paulo)	`sa-east-1`	`emr-serverless.sa-east-1.amazonaws.com`	HTTPS

Name der Region	Region	Endpunkt	Protocol (Protokoll)
China (Peking)	`cn-north-1` (beschränkt auf die folgenden Availability Zones:`cnn1-az1`,`cnn1-az2`)	`emr-serverless.cn-north-1.amazonaws.com.cn`	HTTPS
AWS GovCloud (US-Ost)	`us-gov-east-1`	`emr-serverless.us-gov-east-1.amazonaws.com`	HTTPS
AWS GovCloud (US-West)	`us-gov-west-1`	`emr-serverless.us-gov-west-1.amazonaws.com`	HTTPS

Servicekontingente

Dienstkontingente, auch Limits genannt, sind die maximale Anzahl von Dienstressourcen oder Vorgängen, die Sie verwenden AWS-Konto können. EMR Serverless sammelt jede Minute Messdaten zur Nutzung von Servicekontingenten und veröffentlicht sie im `AWS/Usage` Namespace.

> **Note**
>
> Neue AWS Konten haben anfänglich niedrigere Kontingente, die sich im Laufe der Zeit erhöhen können. Amazon EMR Serverless überwacht die Kontonutzung innerhalb der einzelnen AWS-Region Konten und erhöht dann automatisch die Kontingente entsprechend Ihrer Nutzung.

In der folgenden Tabelle sind die Dienstkontingente für EMR Serverless aufgeführt. Weitere Informationen finden Sie unter Kontingente AWS-Service .

Name	Standardlimit	Anpassbar?	Description
Max. Anzahl gleichzeitiger V CPUs pro Konto	16	Ja	Die maximale Anzahl von vCPUs , die gleichzeitig für das Konto in der aktuellen Version ausgeführt werden können. AWS-Region
Max. Anzahl von Aufträgen in der Warteschlange pro Konto	2000	Ja	Die maximale Anzahl von Aufträgen in der Warteschlange für das aktuelle Konto. AWS-Region

API-Limits

Im Folgenden werden die API-Grenzwerte pro Region für Ihre AWS-Konto beschrieben.

Ressource	Standardkontingent
ListApplications	10 Transaktionen pro Sekunde. Burst von 50 Transaktionen pro Sekunde.
CreateApplication	1 Transaktion pro Sekunde. Burst von 25 Transaktionen pro Sekunde.
DeleteApplication	1 Transaktion pro Sekunde. Burst von 25 Transaktionen pro Sekunde.
GetApplication	10 Transaktionen pro Sekunde. Burst von 50 Transaktionen pro Sekunde.
UpdateApplication	1 Transaktion pro Sekunde. Burst von 25 Transaktionen pro Sekunde.

Ressource	Standardkontingent
ListJobRuns	1 Transaktion pro Sekunde. Burst von 25 Transaktionen pro Sekunde.
StartJobRun	1 Transaktion pro Sekunde. Burst von 25 Transaktionen pro Sekunde.
GetDashboardForJobRun	1 Transaktion pro Sekunde. Burst von 2 Transaktionen pro Sekunde.
CancelJobRun	1 Transaktion pro Sekunde. Burst von 25 Transaktionen pro Sekunde.
GetJobRun	10 Transaktionen pro Sekunde. Burst von 50 Transaktionen pro Sekunde.
StartApplication	1 Transaktion pro Sekunde. Burst von 25 Transaktionen pro Sekunde.
StopApplication	1 Transaktion pro Sekunde. Burst von 25 Transaktionen pro Sekunde.

Weitere Überlegungen

Die folgende Liste enthält weitere Überlegungen zu EMR Serverless.

- EMR Serverless ist in den folgenden Versionen verfügbar: AWS-Regionen
  - US East (Ohio)
  - USA Ost (Nord-Virginia)
  - USA West (Nordkalifornien)
  - USA West (Oregon)
  - Africa (Cape Town)
  - Asia Pacific (Hongkong)
  - Asien-Pazifik (Jakarta)
  - Asien-Pazifik (Mumbai)
  - Asien-Pazifik (Osaka)
  - Asien-Pazifik (Seoul)
  - Asien-Pazifik (Singapur)
  - Asien-Pazifik (Sydney)
  - Asien-Pazifik (Tokio)
  - Canada (Central)
  - Europe (Frankfurt)
  - Europa (Irland)
  - Europa (London)
  - Europa (Milan)
  - Europa (Paris)
  - Europa (Spain)
  - Europe (Stockholm)
  - Middle East (Bahrain)
  - Naher Osten (VAE)
  - Südamerika (São Paulo)
  - AWS GovCloud (US-Ost)
  - AWS GovCloud (US-West)

Eine Liste der Endgeräte, die diesen Regionen zugeordnet sind, finden Sie unter. Service-Endpunkte

- Das Standard-Timeout für die Ausführung eines Jobs beträgt 12 Stunden. Sie können diese Einstellung mit der `executionTimeoutMinutes` Eigenschaft in der `startJobRun` API oder im AWS SDK ändern. Sie können `executionTimeoutMinutes` den Wert auf 0 setzen, wenn Sie möchten, dass bei der Ausführung Ihres Jobs kein Timeout auftritt. Wenn Sie beispielsweise eine Streaming-Anwendung haben, legen Sie den Wert `executionTimeoutMinutes` auf 0 fest, damit der Streaming-Job kontinuierlich ausgeführt werden kann.
- Die `billedResourceUtilization` Eigenschaft in der `getJobRun` API zeigt die aggregierte vCPU, den Arbeitsspeicher und den Speicher an, die für die Jobausführung in Rechnung AWS gestellt wurden. Die abgerechneten Ressourcen beinhalten eine Mindestnutzung von 1 Minute für Mitarbeiter sowie zusätzlichen Speicherplatz von mehr als 20 GB pro Mitarbeiter. In diesen Ressourcen ist die Nutzung für ungenutzte, vorinitialisierte Mitarbeiter nicht enthalten.
- Ohne VPC-Konnektivität kann ein Job auf einige AWS-Service Endpunkte desselben zugreifen. AWS-Region Zu diesen Diensten gehören Amazon S3, AWS Glue, AWS Lake Formation, Amazon CloudWatch Logs AWS KMS, AWS -Security-Token-Service, Amazon DynamoDB, und AWS Secrets Manager. Sie können VPC-Konnektivität aktivieren, um AWS-Services auf andere zuzugreifen AWS PrivateLink, müssen dies jedoch nicht tun. Um auf externe Dienste zuzugreifen, erstellen Sie Ihre Anwendung mit einer VPC.
- EMR Serverless unterstützt HDFS nicht. Die lokalen Festplatten auf Workern sind temporäre Speicher, die EMR Serverless verwendet, um Daten während der Jobausführung zu mischen und zu verarbeiten.

Serverlose Release-Versionen von Amazon EMR

Eine Amazon EMR-Version ist eine Reihe von Open-Source-Anwendungen aus dem Big-Data-Ökosystem. Jede Version enthält Big-Data-Anwendungen, Komponenten und Funktionen, die Sie bei der Ausführung Ihres Jobs für die Bereitstellung und Konfiguration von Amazon EMR Serverless auswählen.

Stellen Sie mit Amazon EMR 6.6.0 und höher EMR Serverless bereit. Diese Bereitstellungsoption ist in früheren Amazon EMR-Release-Versionen nicht verfügbar. Wenn Sie Ihren Job einreichen, geben Sie eine der folgenden unterstützten Versionen an.

Themen

- EMR Serverless6.9.0
- EMR Serverless6.8.0
- EMR Serverless6.7.0
- EMR Serverless6.6.0

AWS runtime for Apache Spark(emr-Spark-8.0-Vorschau)

In der folgenden Tabelle sind die mit (emr-spark-8.0-preview) verfügbaren Anwendungsversionen aufgeführt. `AWS runtime for Apache Spark`

Informationen zur Anwendungsversion

Anwendung	Version
Spark	4.0.1-amzn-0

`AWS runtime for Apache Spark`Versionshinweise für (emr-spark-8.0-preview)

- Vorschauversion — Dies ist eine Vorabversion von Apache Spark 4.0.1. `AWS runtime for Apache Spark` Diese Vorschau ist nur auf EMR Serverless verfügbar.
- Regionale Verfügbarkeit — Diese Vorabversion ist in allen AWS Regionen verfügbar, in denen EMR Serverless verfügbar ist, außer in China und AWS GovCloud (USA).
- Informationen zur Anwendungsversion — Diese Version wird mit den folgenden Anwendungsversionen ausgeliefert:
  - AWS SDK for Java 2.35.5, 1.12.792
  - Python 3.9, 3.11, 3.12
  - Skala 2.13.16
  - AmazonCloudWatchAgent 1.300034.0-amzn-0
  - Delta 4.0.0-amzn-0-spark
  - Eisberg 1.10.0-amzn-spark-0
  - Diese Version wird standardmäßig mit Amazon Corretto 17 (basierend auf OpenJDK) für Anwendungen geliefert, die Corretto 17 (JDK 17) unterstützen.
- Einschränkungen der Vorschauversion — Die folgenden Funktionen sind in dieser Vorschauversion nicht verfügbar:

  - Interaktive Funktionen und Integrationsfunktionen: SageMaker Unified Studio, EMR Studio-Integration, Spark Connect, Livy und JupyterEnterpriseGateway werden nicht unterstützt.
  - Tabellenformate und Zugriffskontrolle: Hudi, Delta Universal Format und Fine-Grained Access Control (FGAC) mit Filtern und Operatoren auf Zeilen- oder Spaltenebene werden nicht unterstützt. DDL/DML
  - Datenkonnektoren:, emr-Dynamodb- und Spark-Redshift-Konnektoren spark-sql-kinesis sind nicht verfügbar.
  - History Server: Der Persistent Spark History Server ist in dieser Vorschauversion nicht verfügbar. Benutzer können weiterhin auf die Live-Benutzeroberfläche von Spark zugreifen, um aktive serverlose Jobs in Echtzeit zu überwachen und zu debuggen.
  - Spezialisierte Funktionen: Materialisierte Ansichten sind nicht verfügbar.
- Vorschaufunktionen — Sie können die folgenden Funktionen in dieser Vorschauversion testen. Diese Vorschauversion wird für Produktions-Workloads nicht empfohlen:
  - SQL-Funktionen: ANSI-SQL-Modus mit strengerer Typbehandlung, SQL PIPE-Syntax (|>) für Verkettungsoperationen, VARIANT-Datentyp für halbstrukturierte JSON-Daten, SQL-Scripting mit Kontrollflussanweisungen und Sitzungsvariablen sowie benutzerdefinierte SQL-Funktionen.
  - Streaming-Verbesserungen: Arbitrary Stateful Processing API v2 mit transformWithState Operator, State Data Source Reader für abfragbaren Streaming-Status (experimentell) und erweiterter Statusspeicher mit verbessertem RocksDB-Changelog-Checkpointing.
  - Support für Tabellenformate: Apache Iceberg v3 mit Unterstützung des VARIANT-Datentyps, Integration von AWS S3 Tables und Full Table Access (FTA) AWS Lake Formation für Iceberg-, Delta Lake- und Hive-Tabellen.
- Zusätzliche Dokumentation — Zusätzliche Apache Spark-Dokumentation finden Sie in der Apache Spark 4.0.1-Release-Dokumentation.

Erste Schritte

Um mit der Vorschauversion von Apache Spark 4.0.1 zu beginnen, erstellen Sie mit der CLI eine serverlose EMR-Anwendung: AWS

```
aws emr-serverless create-application --type spark \
  --release-label emr-spark-8.0-preview \
  --region us-east-1 --name spark4-preview
```

EMR Serverless7.12.0

In der folgenden Tabelle sind die mit EMR Serverless 7.12.0 verfügbaren Anwendungsversionen aufgeführt.

Anwendung	Version
Apache Spark	3.5.6
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 7.12.0 Versionshinweise

- Neue Features
  - Serverloser Speicher für EMR Serverless — Amazon EMR Serverless führt mit EMR Version 7.12 und höher serverlosen Speicher ein, der die lokale Festplattenbereitstellung für Apache Spark-Workloads überflüssig macht. EMR Serverless verarbeitet automatisch Zwischendatenoperationen wie Shuffle ohne Speichergebühren. Serverloser Speicher entkoppelt Speicher von Rechenleistung, sodass Spark Mitarbeiter sofort freilassen kann, wenn sie inaktiv sind, anstatt sie aktiv zu halten, um temporäre Daten aufzubewahren. Weitere Informationen hierzu finden Sie unter aws.amazon.com/serverless-storage-for-emr-serverless.
  - Iceberg Materialized Views — Ab Amazon EMR 7.12.0 unterstützt Amazon EMR Spark die Erstellung und Verwaltung von Iceberg Materialized Views (MV)
  - Hudi Full Table Access — Ab Amazon EMR 7.12.0 unterstützt Amazon EMR jetzt die Steuerung des vollständigen Tabellenzugriffs (FTA) für Apache Hudi in Apache Spark auf der Grundlage Ihrer in Lake Formation definierten Richtlinien. Diese Funktion ermöglicht Lese- und Schreibvorgänge von Ihren Amazon EMR Spark-Jobs aus in Lake Formation registrierten Tabellen, wenn die Jobrolle vollen Tabellenzugriff hat.
  - Iceberg-Versionsupgrade — Amazon EMR 7.12.0 unterstützt Apache Iceberg Version 1.10

EMR Serverless7.11.0

In der folgenden Tabelle sind die mit EMR Serverless 7.11.0 verfügbaren Anwendungsversionen aufgeführt.

Anwendung	Version
Apache Spark	3.5.6
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 7.11.0 Versionshinweise

- Maximale Auftragsausführungszeit — Der Höchstwert für `executionTimeoutMinutes` In `StartJobRun` Aktion für BATCH-Jobs beträgt ab dieser Version 7 Tage. `executionTimeoutMinutes`kann für Batch-Auftragsausführungen nicht mehr so eingestellt werden, dass es kein Timeout gibt. `0`

EMR Serverless7.10.0

In der folgenden Tabelle sind die mit EMR Serverless 7.10.0 verfügbaren Anwendungsversionen aufgeführt.

Anwendung	Version
Apache Spark	3.5.5
Apache Hive	3.1.3
Apache Tez	0.10.2

Versionshinweise zu EMR Serverless 7.10.0

- Metriken für EMR Serverless — Die Monitoring-Metriken wurden neu strukturiert, sodass sie sich auf die `ApplicationName` Dimensionen und konzentrieren. `JobName` Ältere Metriken werden künftig nicht mehr aktualisiert. Weitere Informationen finden Sie unter Monitoring EMR Serverless Applications and Jobs.

EMR Serverless7.9.0

In der folgenden Tabelle sind die mit EMR Serverless 7.9.0 verfügbaren Anwendungsversionen aufgeführt.

Anwendung	Version
Apache Spark	3.5.5
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless7.8.0

In der folgenden Tabelle sind die mit EMR Serverless 7.8.0 verfügbaren Anwendungsversionen aufgeführt.

Anwendung	Version
Apache Spark	3.5.4
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless7.7.0

In der folgenden Tabelle sind die mit EMR Serverless 7.7.0 verfügbaren Anwendungsversionen aufgeführt.

Anwendung	Version
Apache Spark	3.5.3
Apache Hive	3.1.3

Anwendung	Version
Apache Tez	0.10.2

EMR Serverless7.6.0

In der folgenden Tabelle sind die mit EMR Serverless 7.6.0 verfügbaren Anwendungsversionen aufgeführt.

Anwendung	Version
Apache Spark	3.5.3
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless7.5.0

In der folgenden Tabelle sind die mit EMR Serverless 7.5.0 verfügbaren Anwendungsversionen aufgeführt.

Anwendung	Version
Apache Spark	3.5.2
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless7.4.0

In der folgenden Tabelle sind die mit EMR Serverless 7.4.0 verfügbaren Anwendungsversionen aufgeführt.

Anwendung	Version
Apache Spark	3.5.2
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless7.3.0

In der folgenden Tabelle sind die mit EMR Serverless 7.3.0 verfügbaren Anwendungsversionen aufgeführt.

Anwendung	Version
Apache Spark	3.5.1
Apache Hive	3.1.3
Apache Tez	0.10.2

Versionshinweise zu EMR Serverless 7.3.0

- Job-Parallelität und Warteschlange mit EMR Serverless — Job-Parallelität und Warteschlange sind standardmäßig aktiviert, wenn Sie eine neue EMR Serverless-Anwendung auf Amazon EMR Version 7.3.0 oder höher erstellen. Weitere Informationen finden Sie unter. Dieses Dokument beschreibt die ersten Schritte mit Parallelität und Warteschlangen und enthält auch eine Liste mit wichtigen Funktionen. the section called “Parallelität von Aufträgen und Warteschlangen”

EMR Serverless7.2.0

In der folgenden Tabelle sind die mit EMR Serverless 7.2.0 verfügbaren Anwendungsversionen aufgeführt.

Anwendung	Version
Apache Spark	3.5.1
Apache Hive	3.1.3
Apache Tez	0.10.2

Versionshinweise zu EMR Serverless 7.2.0

- Lake Formation mit EMR Serverless — Sie können jetzt detaillierte Zugriffskontrollen auf Datenkatalogtabellen anwenden, die von S3 unterstützt werden. AWS Lake Formation Mit dieser Funktion können Sie Zugriffskontrollen auf Tabellen-, Zeilen-, Spalten- und Zellenebene für Leseabfragen innerhalb Ihrer EMR Serverless Spark-Jobs konfigurieren. Weitere Informationen finden Sie unter the section called “Lake Formation für FGAC” und the section called “Überlegungen und Einschränkungen”.

EMR Serverless7.1.0

In der folgenden Tabelle sind die mit EMR Serverless 7.1.0 verfügbaren Anwendungsversionen aufgeführt.

Anwendung	Version
Apache Spark	3.5.0
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless7.0.0

In der folgenden Tabelle sind die mit EMR Serverless 7.0.0 verfügbaren Anwendungsversionen aufgeführt.

Anwendung	Version
Apache Spark	3.5.0
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless6.15,0

In der folgenden Tabelle sind die mit EMR Serverless 6.15.0 verfügbaren Anwendungsversionen aufgeführt.

Anwendung	Version
Apache Spark	3.4.1
Apache Hive	3.1.3
Apache Tez	0.10.2

Versionshinweise zu EMR Serverless 6.15.0

- TLS-Unterstützung — Mit den Versionen 6.15.0 und höher von Amazon EMR Serverless können Sie die mit Mutual-TLS verschlüsselte Kommunikation zwischen Workern in Ihren Spark-Auftragsausführungen aktivieren. Wenn diese Option aktiviert ist, generiert EMR Serverless automatisch ein eindeutiges Zertifikat für jeden Worker, das es im Rahmen von Jobläufen bereitstellt, die Mitarbeiter während des TLS-Handshakes verwenden, um sich gegenseitig zu authentifizieren und einen verschlüsselten Kanal für die sichere Verarbeitung von Daten einzurichten. Weitere Informationen zur Mutual-TLS-Verschlüsselung finden Sie unter Verschlüsselung zwischen Mitarbeitern.

EMR Serverless6.14.0

In der folgenden Tabelle sind die mit EMR Serverless 6.14.0 verfügbaren Anwendungsversionen aufgeführt.

Anwendung	Version
Apache Spark	3.4.1
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless6.13,0

In der folgenden Tabelle sind die mit EMR Serverless 6.13.0 verfügbaren Anwendungsversionen aufgeführt.

Anwendung	Version
Apache Spark	3.4.1
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless6.12.0

In der folgenden Tabelle sind die mit EMR Serverless 6.12.0 verfügbaren Anwendungsversionen aufgeführt.

Anwendung	Version
Apache Spark	3.4.0
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless6.11.0

In der folgenden Tabelle sind die mit EMR Serverless 6.11.0 verfügbaren Anwendungsversionen aufgeführt.

Anwendung	Version
Apache Spark	3.3.2
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 6.11.0 Versionshinweise

- Zugriff auf S3-Ressourcen in anderen Konten — In den Versionen 6.11.0 und höher können Sie mehrere IAM-Rollen konfigurieren, die beim Zugriff auf Amazon S3 S3-Buckets in verschiedenen AWS Konten von EMR Serverless aus übernommen werden.

EMR Serverless6.10.0

In der folgenden Tabelle sind die mit EMR Serverless 6.10.0 verfügbaren Anwendungsversionen aufgeführt.

Anwendung	Version
Apache Spark	3.3.1
Apache Hive	3.1.3
Apache Tez	0.10.2

Versionshinweise zu EMR Serverless 6.10.0

- Für serverlose EMR-Anwendungen mit Version 6.10.0 oder höher ist der Standardwert für die `spark.dynamicAllocation.maxExecutors` Eigenschaft. `infinity` Frühere Versionen sind standardmäßig auf. `100` Weitere Informationen finden Sie unter Eigenschaften von Spark-Jobs.

EMR Serverless6.9.0

In der folgenden Tabelle sind die mit EMR Serverless 6.9.0 verfügbaren Anwendungsversionen aufgeführt.

Anwendung	Version
Apache Spark	3.3.0
Apache Hive	3.1.3
Apache Tez	0.10.2

Versionshinweise zu EMR Serverless 6.9.0

- Die Amazon-Redshift-Integration für Apache Spark ist in den Amazon-EMR-Versionen 6.9.0 und höher enthalten. Die native Integration war bisher ein Open-Source-Tool und ist ein Spark-Konnektor, mit dem Sie Apache-Spark-Anwendungen erstellen können, die Daten in Amazon Redshift und Amazon Redshift Serverless lesen und in diese schreiben. Weitere Informationen finden Sie unter Verwenden der Amazon Redshift Redshift-Integration für Apache Spark auf Amazon EMR Serverless.
- Die Version 6.9.0 von EMR Serverless bietet Unterstützung für die AWS Graviton2-Architektur (arm64). Sie können den `architecture` Parameter für `create-application` und `update-application` APIs verwenden, um die arm64-Architektur auszuwählen. Weitere Informationen finden Sie unter Optionen für die serverlose Architektur von Amazon EMR.
- Sie können jetzt Amazon DynamoDB-Tabellen direkt aus Ihren EMR Serverless Spark- und Hive-Anwendungen exportieren, importieren, abfragen und verbinden. Weitere Informationen finden Sie unter Mit Amazon EMR Serverless eine Verbindung zu DynamoDB herstellen.

Bekannte Probleme

- Wenn Sie die Amazon Redshift Redshift-Integration für Apache Spark verwenden und eine Zeit, Timetz, Zeitstempel oder Zeitstempeltz mit Mikrosekundengenauigkeit im Parquet-Format haben, rundet der Konnektor die Zeitwerte auf den nächsten Millisekundenwert. Um das Problem zu umgehen, verwenden Sie den `unload_s3_format`-Formatparameter-Text-Unload.

EMR Serverless6.8.0

In der folgenden Tabelle sind die mit EMR Serverless 6.8.0 verfügbaren Anwendungsversionen aufgeführt.

Anwendung	Version
Apache Spark	3.3.0
Apache Hive	3.1.3
Apache Tez	0.9.2

EMR Serverless6.7.0

In der folgenden Tabelle sind die mit EMR Serverless 6.7.0 verfügbaren Anwendungsversionen aufgeführt.

Anwendung	Version
Apache Spark	3.2.1
Apache Hive	3.1.3
Apache Tez	0.9.2

Engine-spezifische Änderungen, Verbesserungen und behobene Probleme

In der folgenden Tabelle ist eine neue motorspezifische Funktion aufgeführt.

Änderungen	Description
Feature	Der Tez-Scheduler unterstützt jetzt das Preemptive von Tez-Aufgaben anstelle des Preemptivs von Containern

EMR Serverless6.6.0

In der folgenden Tabelle sind die mit EMR Serverless 6.6.0 verfügbaren Anwendungsversionen aufgeführt.

Anwendung	Version
Apache Spark	3.2.0
Apache Hive	3.1.2
Apache Tez	0.9.2

Erste Versionshinweise zu EMR Serverless

- EMR Serverless unterstützt die Spark-Konfigurationsklassifizierung. `spark-defaults` Diese Klassifizierung ändert Werte in der `spark-defaults.conf` XML-Datei von Spark. Mithilfe von Konfigurationsklassifizierungen können Sie Anwendungen anpassen. Weitere Informationen finden Sie unter Anwendungen konfigurieren.
- EMR Serverless unterstützt die Hive-Konfigurationsklassifizierungen`hive-site`,, und`tez-site`. `emrfs-site` `core-site` Diese Klassifizierung kann die Werte in der Datei von Hive, der `hive-site.xml` Datei von Tez, den EMRFS-Einstellungen von Amazon EMR bzw. der `tez-site.xml` Hadoop-Datei ändern. `core-site.xml` Mithilfe von Konfigurationsklassifizierungen können Sie Anwendungen anpassen. Weitere Informationen finden Sie unter Anwendungen konfigurieren.

Engine-spezifische Änderungen, Verbesserungen und behobene Probleme

- In der folgenden Tabelle sind die Backports von Hive und Tez aufgeführt.

  Änderungen an Hive und Tez

Änderungen	Description
Backport	TEZ-4430: Ein Problem mit der Immobilie wurde behoben `tez.task.launch.cmd-opts`

Änderungen	Description
Backport	HIVE-25971: Verzögerungen beim Herunterfahren von Tez-Tasks aufgrund des offenen zwischengespeicherten Threadpools wurden behoben

Dokumentverlauf

In der folgenden Tabelle werden die wichtigen Änderungen an der Dokumentation seit der letzten Version von EMR Serverless beschrieben. Um mehr Informationen über Aktualisierungen dieser Dokumentation zu erhalten, können Sie einen RSS-Feed abonnieren.

Änderung	Beschreibung	Datum
Neue Vorschauversion	`AWS runtime for Apache Spark(emr-Spark-8.0-Vorschau)`	21. November 2025
Neue Veröffentlichung	EMR Serverless7.2.0	25. Juli 2024
Neue Veröffentlichung	EMR Serverless7.1.0	17. April 2024
Aktualisierung einer bestehenden Richtlinie.	Das neue `Sid CloudWatchPolicyStatement` und `EC2PolicyStatement` wurde der EMRServerlessServiceRolePolicy Amazon-Richtlinie hinzugefügt.	25. Januar 2024
Neue Veröffentlichung	EMR Serverless7.0.0	29. Dezember 2023
Neue Veröffentlichung	EMR Serverless6.15,0	17. November 2023
Neues Feature	Konfigurieren Sie mehrere IAM-Rollen, die beim Zugriff auf Amazon S3 S3-Buckets in verschiedenen Konten von EMR Serverless (6.11 und höher) übernommen werden sollen	18. Oktober 2023
Neue Version	EMR Serverless6.14.0	17. Oktober 2023

Neues Feature	Standardanwendungskonfiguration für EMR Serverless	25. September 2023
Aktualisieren Sie die Hive-Standardeigenschaften	Die Standardwerte für die `hive.driver.disk` Jobeigenschaften, `hive.tez.disk.size`, `hive.tez.auto.reducer.parallelism`, und `tez.grouping.min-size` Hive wurden aktualisiert.	12. September 2023
Neue Version	EMR Serverless6.13,0	11. September 2023
Neue Veröffentlichung	EMR Serverless6.12.0	21. Juli 2023
Neue Veröffentlichung	EMR Serverless6.11.0	08. Juni 2023
Aktualisierung der Richtlinie für dienstbezogene Rollen	Die `AmazonEMRServerlessServiceRolePolicy`_SLR-Rolle wurde aktualisiert, sodass die Nutzung auf Kontoebene im Namespace veröffentlicht wird. `"AWS/Usage"`	20. April 2023
EMR Serverlessallgemeine Verfügbarkeit (GA)	Dies ist die erste öffentliche Version von EMR Serverless.	1. Juni 2022

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.