Amazon EMR Serverless Benutzerhandbuch

Amazon EMR

Amazon EMR: Amazon EMR Serverless Benutzerhandbuch

Table of Contents

Was ist Amazon EMR Serverless?

Amazon EMR Serverless ist eine Bereitstellungsoption für AmazonEMR, 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.

EMRServerless hilft Ihnen, eine Über- oder Unterversorgung von Ressourcen für Ihre Datenverarbeitungsaufgaben zu vermeiden. EMRServerless ermittelt automatisch die Ressourcen, die die Anwendung benötigt, ruft diese Ressourcen für die Verarbeitung Ihrer Jobs ab und gibt die Ressourcen wieder 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 AmazonEMR, wie Open-Source-Kompatibilität, Parallelität und optimierter Laufzeitleistung für beliebte Frameworks.

EMRServerless 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-Benutzerhandbuch immer wieder vorkommen.

Version veröffentlichen

Eine EMR Amazon-Version ist eine 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, müssen Sie deren Release-Version angeben. Wählen Sie die EMR Amazon-Release-Version und die Open-Source-Framework-Version, die Sie in Ihrer Anwendung verwenden möchten. Weitere Informationen zu Vorabversionen finden Sie unterRelease-Versionen von Amazon EMR Serverless.

Anwendung

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

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

Nachdem Sie eine Anwendung erstellt haben, können Sie Datenverarbeitungsaufträge oder interaktive Anfragen an Ihre Anwendung senden.

Jede EMR serverlose Anwendung läuft auf einer sicheren Amazon Virtual Private Cloud (VPC), strikt getrennt von anderen Anwendungen. Darüber hinaus können Sie Folgendes verwenden 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

EMRServerless ist ein regionaler Dienst, 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. Interaktion mit einer Anwendung

Aufgabenausführung

Bei einer Auftragsausführung handelt es sich um eine an eine EMR serverlose Anwendung gesendete Anforderung, die von der Anwendung asynchron ausgeführt und bis zum Abschluss verfolgt wird. 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 Runtime-Rolle angeben, in der der Job erstellt wurde und auf die der Job IAM zugreift AWS Ressourcen, wie Amazon S3 S3-Objekte. Sie können mehrere Anfragen zur Auftragsausführung an eine Anwendung senden, und für jede Auftragsausführung kann eine andere Runtime-Rolle für den Zugriff verwendet werden AWS Ressourcen schätzen. Eine EMR serverlose Anwendung beginnt mit der Ausführung von Aufträgen, 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 EMR serverlose Anwendung verwendet intern Worker, um Ihre Workloads auszuführen. Die Standardgrößen dieser Worker basieren auf Ihrem Anwendungstyp und der EMR Amazon-Release-Version. Wenn Sie eine Auftragsausführung planen, können Sie diese Größen überschreiben.

Wenn Sie einen Job einreichen, berechnet EMR Serverless die Ressourcen, die die Anwendung für den Job benötigt, und plant die Mitarbeiter ein. EMRServerless unterteilt Ihre Workloads in Aufgaben, lädt Bilder herunter, stellt Mitarbeiter bereit und richtet sie ein und nimmt sie wieder in Betrieb, wenn der Job abgeschlossen ist. EMRServerless 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

EMRServerless 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

EMRStudio

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

Voraussetzungen für den Einstieg in EMR Serverless

Themen

Melde dich an für ein AWS-Konto

Wenn Sie kein haben AWS-Konto, führen Sie die folgenden Schritte aus, um einen 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.

   Bei der Anmeldung müssen Sie auch einen Telefonanruf entgegennehmen und einen Verifizierungscode über die Telefontasten eingeben.

   Wenn Sie sich für eine anmelden AWS-Konto, ein Root-Benutzer des AWS-Kontoswird erstellt. Der Root-Benutzer hat Zugriff auf alle AWS-Services und Ressourcen im Konto. Als bewährte Sicherheitsmethode weisen Sie einem Administratorbenutzer 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 sich Ihre Root-Benutzer des AWS-Kontos, aktivieren AWS IAM Identity Center, und erstellen Sie einen Administratorbenutzer, sodass Sie den Root-Benutzer nicht für alltägliche Aufgaben verwenden.

Sichern Sie Ihre Root-Benutzer des AWS-Kontos

1. Melde dich an bei AWS Management Consoleals Kontoinhaber wählen Sie Root-Benutzer und geben Sie Ihren AWS-Konto E-Mail-Adresse. Geben Sie auf der nächsten Seite Ihr Passwort ein.

   Hilfe bei der Anmeldung mit dem Root-Benutzer finden Sie unter Als Root-Benutzer anmelden im AWS-Anmeldung Benutzerleitfaden.
2. Aktivieren Sie die Multi-Faktor-Authentifizierung (MFA) für Ihren Root-Benutzer.

   Anweisungen finden Sie unter Aktivieren Sie ein virtuelles MFA Gerät für AWS-Konto Root-Benutzer (Konsole) im IAMBenutzerhandbuch.

Erstellen eines Benutzers mit Administratorzugriff

1. Aktivieren Sie IAM Identity Center.

   Anweisungen finden Sie unter Aktivieren AWS IAM Identity Center in der AWS IAM Identity Center Benutzerleitfaden.
2. Gewähren Sie einem Benutzer in IAM Identity Center Administratorzugriff.

   Für ein Tutorial zur Verwendung des IAM-Identity-Center-Verzeichnis als Identitätsquelle finden Sie unter Benutzerzugriff mit der Standardeinstellung konfigurieren IAM-Identity-Center-Verzeichnis in der AWS IAM Identity Center Benutzerleitfaden.

Anmelden als Administratorbenutzer

- Um sich mit Ihrem IAM Identity Center-Benutzer anzumelden, verwenden Sie die Anmeldung, 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 unter Anmelden bei AWS Zugriffsportal im AWS-Anmeldung Benutzerleitfaden.

Weiteren Benutzern Zugriff zuweisen

1. Erstellen Sie in IAM Identity Center einen Berechtigungssatz, der der bewährten Methode zur Anwendung von Berechtigungen mit den geringsten Rechten folgt.

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

   Anweisungen finden Sie unter Gruppen hinzufügen im AWS IAM Identity Center Benutzerleitfaden.

Erteilen Sie Berechtigungen

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

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

```
{
    "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. Folgen Sie den Anweisungen unter Einen Berechtigungssatz erstellen im AWS IAM Identity Center Benutzerleitfaden.
- Benutzer, IAM die über einen Identitätsanbieter verwaltet werden:

  Erstellen Sie eine Rolle für den Identitätsverbund. Folgen Sie den Anweisungen unter Erstellen einer Rolle für einen externen Identitätsanbieter (Federation) im IAMBenutzerhandbuch.
- IAMBenutzer:
  - Erstellen Sie eine Rolle, die Ihr Benutzer annehmen kann. Folgen Sie den Anweisungen unter Eine Rolle für einen IAM Benutzer erstellen im IAMBenutzerhandbuch.

- (Nicht empfohlen) Weisen Sie einem Benutzer eine Richtlinie direkt zu oder fügen Sie einen Benutzer zu einer Benutzergruppe hinzu. Folgen Sie den Anweisungen unter Hinzufügen von Berechtigungen für einen Benutzer (Konsole) im IAMBenutzerhandbuch.

Erteilen programmgesteuerten Zugriffs

Benutzer benötigen programmgesteuerten Zugriff, wenn sie mit AWS außerhalb der AWS Management Console. Die Art und Weise, wie programmatischer Zugriff gewährt wird, hängt von der Art des Benutzers ab, der zugreift AWS.

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

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
Mitarbeiteridentität (In IAM Identity Center verwaltete Benutzer)	Verwenden Sie temporäre Anmeldeinformationen, um programmatische Anfragen an das zu signieren AWS CLI, AWS SDKs, oder AWS APIs.	Befolgen Sie die Anweisung en für die Schnittstelle, die Sie verwenden möchten. • Für den AWS CLI, siehe Konfiguration der AWS CLI zu verwenden AWS IAM Identity Center in der AWS Command Line Interface Benutzerleitfaden. • Wählen Sie in der &Snowconsole; Ihren Auftrag aus der Tabelle. AWS SDKs, Tools und AWS APIs, siehe IAMIdentity Center-Authentifizierung im AWS SDKsund Referenzh andbuch für Tools.
IAM	Verwenden Sie temporäre Anmeldeinformationen, um programmatische Anfragen	Folgen Sie den Anweisungen unter Temporäre Anmeldein formationen verwenden

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
	an das zu signieren AWS CLI, AWS SDKs, oder AWS APIs.	mit AWS Ressourcen im IAMBenutzerhandbuch.
IAM	(Nicht empfohlen) Verwenden Sie langfristige Anmeldeinformationen, um programmatische Anfragen an das zu signieren AWS CLI, AWS SDKs, oder AWS APIs.	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten. • Für den AWS CLI, siehe Authentifizierung mit IAM Benutzeranmeldedaten in der AWS Command Line Interface Benutzerleitfaden. • Wählen Sie in der &Snowconsole; Ihren Auftrag aus der Tabelle. AWS SDKsund Tools finden Sie unter Authentifizieren mit langfristigen Anmeldeinformationen in der AWS SDKsund Referenzhandbuch für Tools. • Wählen Sie in der &Snowconsole; Ihren Auftrag aus der Tabelle. AWS APIs, siehe Verwaltung von Zugriffsschlüsseln für IAM IAM Benutzer im Benutzerhandbuch.

Installieren und konfigurieren Sie AWS CLI

Wenn Sie EMR Serverless verwenden möchtenAPIs, müssen Sie die neueste Version von installieren AWS Command Line Interface (AWS CLI). Du brauchst das nicht AWS CLI um EMR

Serverless von der EMR Studio-Konsole aus zu verwenden, und Sie können ohne das loslegen, CLI indem Sie die Schritte unter befolgen. Erste Schritte mit EMR Serverless von der Konsole aus

Um das einzurichten AWS CLI

1. Um die neueste Version von zu installieren AWS CLI für macOS, Linux oder Windows finden Sie unter Installation oder Aktualisierung der neuesten Version von AWS CLI.
2. Um das zu konfigurieren AWS CLI und sichere Einrichtung Ihres Zugangs zu AWS-Services, einschließlich EMR Serverless, siehe Schnellkonfiguration mit `aws configure`.
3. Um das Setup zu überprüfen, geben Sie an der DataBrew Befehlszeile den folgenden Befehl ein.

   ```
   aws emr-serverless help
   ```

   AWS CLI Befehle verwenden die Standardeinstellung AWS-Region aus Ihrer Konfiguration, sofern Sie sie nicht mit einem Parameter oder einem Profil festgelegt haben. Um deine einzustellen AWS-Region Mit einem Parameter können Sie den `--region` Parameter zu jedem Befehl hinzufügen.

   Um deine einzustellen AWS-Region Fügen Sie mit einem Profil 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. Stellen Sie als Nächstes Ihre ein AWS-Region und andere Einstellungen mit einem Befehl, 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 mit der EMRAmazon-Konsole. Wenn Sie noch nicht bei Ihrem angemeldet sind AWS-Konto, melden Sie sich an, öffnen Sie dann die EMRAmazon-Konsole und fahren Sie mit dem nächsten Abschnitt fort, um die ersten Schritte mit Amazon fortzusetzenEMR.

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

Themen

Erteilen Sie Berechtigungen zur Verwendung von Serverless EMR

Um EMR Serverless verwenden zu können, benötigen Sie einen Benutzer oder eine 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

Bereiten Sie den Speicher für EMR Serverless vor

In diesem Tutorial verwenden Sie einen S3-Bucket, um Ausgabedateien und Protokolle des Spark- oder Hive-Beispiel-Workloads zu speichern, den Sie mit einer EMR serverlosen 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 *DOC-EXAMPLE-BUCKET* durch den Namen des neu erstellten Buckets.

Erstellen Sie ein EMR Studio, um interaktive Workloads auszuführen

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 EMRStudio einrichten im Amazon EMR Management Guide. Weitere Informationen zu interaktiven Workloads finden Sie unterFühren Sie interaktive Workloads mit EMR Serverless über Studio aus EMR.

Erstellen Sie eine Job-Runtime-Rolle

Auftragsausführungen in EMR Serverless verwenden eine Runtime-Rolle, die detaillierte Berechtigungen für bestimmte AWS-Services und Ressourcen zur Laufzeit. In diesem Tutorial hostet ein öffentlicher S3-Bucket die Daten und Skripte. Der Bucket *DOC-EXAMPLE-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 unter https://console.aws.amazon.com/iam/.
2. Wählen Sie im linken Navigationsbereich Roles aus.
3. Wählen Sie Rolle erstellen.
4. 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, auf andere zugreifen AWS-Services in Ihrem Namen.

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

```
      "Principal": {
        "Service": "emr-serverless.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

5. Wählen Sie Weiter, um zur Seite „Berechtigungen hinzufügen“ zu navigieren, und wählen Sie dann Richtlinie erstellen aus.
6. Die Seite „Richtlinie erstellen“ wird auf einer neuen Registerkarte geöffnet. Fügen Sie die Richtlinie JSON unten ein.

> **Important**
>
> Ersetzen Sie die *DOC-EXAMPLE-BUCKET* nachstehende Richtlinie durch den tatsächlichen Bucket-Namen, der in erstellt wurdeBereiten Sie den Speicher für EMR Serverless vor. 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.

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

```
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
            ]
        },
        {
            "Sid": "GlueCreateAndReadDataCatalog",
            "Effect": "Allow",
            "Action": [
                "glue:GetDatabase",
                "glue:CreateDatabase",
                "glue:GetDataBases",
                "glue:CreateTable",
                "glue:GetTable",
                "glue:UpdateTable",
                "glue:DeleteTable",
                "glue:GetTables",
                "glue:GetPartition",
                "glue:GetPartitions",
                "glue:CreatePartition",
                "glue:BatchCreatePartition",
                "glue:GetUserDefinedFunctions"
            ],
            "Resource": ["*"]
        }
    ]
}
```

7. Geben Sie auf der Seite „Richtlinie überprüfen“ einen Namen für Ihre Richtlinie ein, z. `EMRServerlessS3AndGlueAccessPolicy` B.
8. Aktualisieren Sie die Seite mit den Richtlinien zum Anhängen von Berechtigungen und wählen Sie`EMRServerlessS3AndGlueAccessPolicy`.
9. Geben Sie auf der Seite Name, Überprüfung und Erstellung für 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 mit dem Namen`emr-serverless-trust-policy.json`, die die Vertrauensrichtlinie enthält, die für die IAM Rolle verwendet werden soll. Die Datei sollte die folgende Richtlinie enthalten.

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

2. Erstellen Sie eine IAM Rolle mit dem Namen`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 die neue Rolle bei ARN der Einreichung von Jobs, die im Folgenden als die bezeichnet wird*`job-role-arn`*.

3. Erstellen Sie eine Datei mit dem Namen`emr-sample-access-policy.json`, der 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. *`DOC-EXAMPLE-BUCKET`*

> **Important**
>
> Ersetzen Sie die folgende Richtlinie durch den tatsächlichen Bucket-Namen, der *`DOC-EXAMPLE-BUCKET`* in.. erstellt wurde. Bereiten Sie den Speicher für EMR Serverless vor Dies ist eine grundlegende Richtlinie für AWS Glue- und S3-Zugriff. Weitere Beispiele für Rollen zur Laufzeit von Jobs finden Sie unterJob-Runtime-Rollen für Amazon EMR Serverless.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadAccessForEMRSamples",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::*.elasticmapreduce",
                "arn:aws:s3:::*.elasticmapreduce/*"
            ]
        },
        {
            "Sid": "FullAccessToOutputBucket",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
            ]
        },
        {
            "Sid": "GlueCreateAndReadDataCatalog",
            "Effect": "Allow",
            "Action": [
                "glue:GetDatabase",
                "glue:CreateDatabase",
                "glue:GetDataBases",
                "glue:CreateTable",
                "glue:GetTable",Understanding default application behavior,
 including auto-start and auto-stop, as well as maximum capacity and worker
 configurations for configuring an application with &EMRServerless;.
                "glue:UpdateTable",
                "glue:DeleteTable",
```

```
                "glue:GetTables",
                "glue:GetPartition",
                "glue:GetPartitions",
                "glue:CreatePartition",
                "glue:BatchCreatePartition",
                "glue:GetUserDefinedFunctions"
            ],
            "Resource": ["*"]
        }
    ]
}
```

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

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

   Beachten Sie die neuen Richtlinien ARN in der Ausgabe. Sie werden sie *`policy-arn`* im nächsten Schritt ersetzen.

5. Ordnen Sie die IAM Richtlinie `EMRServerlessS3AndGlueAccessPolicy` der Job-Runtime-Rolle zu`EMRServerlessS3RuntimeRole`.

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

Erste Schritte mit EMR Serverless von der Konsole aus

Schritte zum Absolvieren

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

Schritt 1: Erstellen Sie eine EMR serverlose Anwendung

Erstellen Sie wie folgt eine neue Anwendung mit EMR Serverless.

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

   - Wenn Sie bereits ein EMR Studio in der haben AWS-Region wo Sie eine Anwendung erstellen möchten, wählen Sie dann Anwendungen verwalten aus, um zu Ihrem EMR Studio zu navigieren, oder wählen Sie das Studio aus, das Sie verwenden möchten.
   - Wenn Sie kein EMR Studio in der haben AWS-Region wo Sie eine Anwendung erstellen möchten, wählen Sie Erste Schritte und dann Studio erstellen und starten. EMRServerless 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://DOC-EXAMPLE-BUCKET/scripts/
   ```

2. Wenn Sie den Vorgang abschließen, Schritt 1: Erstellen Sie eine EMR serverlose Anwendung gelangen Sie zur Seite mit den Anwendungsdetails in EMR Studio. Wählen Sie dort die Option Job einreichen.
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 erstellt habenErstellen Sie eine Job-Runtime-Rolle.
   - Geben Sie im Feld Skriptspeicherort den `s3://DOC-EXAMPLE-BUCKET/scripts/wordcount.py` Wert S3 einURI.
   - Geben Sie im Feld Skriptargumente den Wert ein`["s3://DOC-EXAMPLE-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://DOC-EXAMPLE-BUCKET/emr-serverless-hive/query/hive-query.ql
```

3. Wenn Sie den Vorgang abschließen, Schritt 1: Erstellen Sie eine EMR serverlose Anwendung gelangen Sie zur Seite mit den Anwendungsdetails in EMR Studio. Wählen Sie dort die Option Job einreichen.
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 erstellt habenErstellen Sie eine Job-Runtime-Rolle.
   - Geben Sie im Feld Skriptspeicherort den `s3://DOC-EXAMPLE-BUCKET/emr-serverless-hive/query/hive-query.ql` Wert S3 einURI.

- Wählen Sie im Bereich 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 Bearbeiten als JSON und geben Sie Folgendes einJSON.

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

5. Um die Auftragsausführung zu starten, wählen Sie Job einreichen.
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 Serverless auszuführen. EMR Weitere Informationen, einschließlich Berechtigungen und Voraussetzungen, finden Sie unter. Führen Sie interaktive Workloads mit EMR Serverless über Studio aus EMR

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 Arbeitsbereiche. Wenn Sie noch einen Amazon S3 S3-Speicherort und eine EMRStudio-Servicerolle konfigurieren müssen, wählen Sie im Banner oben auf dem Bildschirm 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 übernehmen 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 Beispiel-Notebooks 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 Überwachung EMR serverloser 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 Stellen bei Spark und Jobs bei Hive.

Erste Schritte vom AWS CLI

Schritt 1: Erstellen Sie eine EMR serverlose Anwendung

Verwenden Sie den `emr-serverless create-application` Befehl, um Ihre erste EMR serverlose Anwendung zu erstellen. Sie müssen den Anwendungstyp und das EMR Amazon-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 ausgeführten Job an Ihre EMR Serverless-Anwendung, stellen Sie sicher, dass Ihre Bewerbung den `CREATED` Status mit dem erreicht hat `get-application`API.

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

EMRServerless erstellt Mitarbeiter, 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.

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

Jetzt ist Ihre EMR serverlose 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://DOC-EXAMPLE-BUCKET/scripts/
```

2. Geben Sie im folgenden Befehl Ihre Anwendungs-ID ein. *application-id job-role-arn*Ersetzen Sie es durch die Runtime-Rolle, in der ARN Sie sie erstellt habenErstellen Sie eine Job-Runtime-Rolle. Ersetze *job-run-name* mit dem Namen, den Sie Ihren Job Run nennen möchten. Ersetzen Sie alle *DOC-EXAMPLE-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://DOC-EXAMPLE-BUCKET/scripts/wordcount.py",
          "entryPointArguments": ["s3://DOC-EXAMPLE-BUCKET/emr-serverless-spark/
output"],
          "sparkSubmitParameters": "--conf spark.executor.cores=1
 --conf spark.executor.memory=4g --conf spark.driver.cores=1 --conf
 spark.driver.memory=4g --conf spark.executor.instances=1"
        }
    }'
```

3. Notieren Sie sich die in der Ausgabe zurückgegebene Job-Run-ID. *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://DOC-EXAMPLE-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 durch die Runtime-Rolle, in der ARN Sie sie erstellt habenErstellen Sie eine Job-Runtime-Rolle. Ersetzen Sie alle *`DOC-EXAMPLE-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://DOC-EXAMPLE-BUCKET/emr-serverless-hive/query/hive-
query.ql",
          "parameters": "--hiveconf hive.log.explain.output=false"
        }
    }' \
    --configuration-overrides '{
      "applicationConfiguration": [{
        "classification": "hive-site",
          "properties": {
            "hive.exec.scratchdir": "s3://DOC-EXAMPLE-BUCKET/emr-serverless-
hive/hive/scratch",
            "hive.metastore.warehouse.dir": "s3://DOC-EXAMPLE-BUCKET/emr-
serverless-hive/hive/warehouse",
```

```
            "hive.driver.cores": "2",
            "hive.driver.memory": "4g",
            "hive.tez.container.size": "4096",
            "hive.tez.cpu.vcores": "1"
            }
        }],
        "monitoringConfiguration": {
          "s3MonitoringConfiguration": {
            "logUri": "s3://DOC-EXAMPLE-BUCKET/emr-serverless-hive/logs"
           }
        }
    }'
```

4. Notieren Sie sich die in der Ausgabe zurückgegebene Job-Run-ID. *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://*DOC-EXAMPLE-BUCKET*/emr-serverless-spark/logs, finden Sie die Protokolle für diesen speziellen Job, der ausgeführt wird, unters3://*DOC-EXAMPLE-BUCKET*/emr-serverless-spark/logs/applications/*application-id*/jobs/*job-run-id*.

Bei 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://*DOC-EXAMPLE-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://*DOC-EXAMPLE-BUCKET*/emr-serverless-hive/logs, finden Sie die Protokolle für diesen speziellen Job, der ausgeführt wird, unters3://*DOC-EXAMPLE-BUCKET*/emr-serverless-hive/logs/applications/*application-id*/jobs/*job-run-id*.

Bei Hive-Anwendungen lädt EMR Serverless kontinuierlich den Hive-Treiber in den HIVE_DRIVER Ordner und die Tez-Aufgabenprotokolle in den TEZ_TASK Ordner Ihres S3-Protokollziels hoch. 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. *DOC-EXAMPLE-BUCKET*Ersetzen Sie es durch den tatsächlichen Namen des S3-Buckets, der inBereiten Sie den Speicher für EMR Serverless vor.. erstellt wurde.

```
aws s3 rm s3://DOC-EXAMPLE-BUCKET --recursive
aws s3api delete-bucket --bucket DOC-EXAMPLE-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 Stellen bei Spark undJobs bei Hive.

Interaktion mit einer Anwendung

In diesem Abschnitt erfahren Sie, wie Sie mit Ihrer Amazon EMR Serverless-Anwendung interagieren können, indem Sie AWS CLI und die Standardeinstellungen für Spark- und Hive-Engines.

Themen

Status der Anwendung

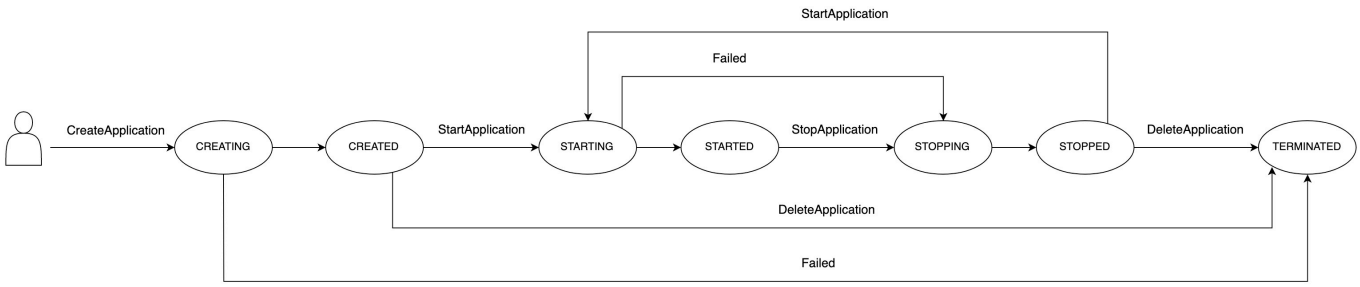
Wenn Sie eine Anwendung mit EMR Serverless erstellen, wechselt die Ausführung der Anwendung in den Status. `CREATING` Danach durchläuft er die folgenden Zustände, bis er erfolgreich ist (Abschluss mit Code `0`) oder fehlschlägt (Abschluss mit einem Code ungleich null).

Anwendungen können die folgenden Status haben:

Status	Beschreibung
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.
Wird gestartet	Die Anwendung wird gestartet und stellt Kapazität bereit.

Status	Beschreibung
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 zeigt den Verlauf der Zustände EMR serverloser Anwendungen.



Interaktion mit Ihrer Anwendung über die EMR Studio-Konsole

Von der EMR Studio-Konsole aus können Sie EMR serverlose Anwendungen erstellen, anzeigen und verwalten. Folgen Sie den Anweisungen unter Erste Schritte von der Konsole aus, um zur EMR Studio-Konsole zu navigieren.

Erstellen einer Anwendung

Auf der Seite „Anwendung erstellen“ können Sie eine EMR serverlose Anwendung erstellen, 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 Versionsnummer aus.
4. Wählen Sie in den Architekturoptionen die zu verwendende Befehlssatzarchitektur aus. Weitere Informationen finden Sie unter Optionen für die EMR serverlose Architektur von Amazon.

   - arm64 — ARM 64-Bit-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 Sie der Anwendung zuweisen können.

   Weitere Informationen zu vorinitialisierter Kapazität, Anwendungslimits und Anwendungsverhalten finden Sie unter. Konfiguration einer Anwendung Weitere Informationen zur Netzwerkkonnektivität finden Sie unter. Zugriff konfigurieren VPC

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

Anwendungen auflisten

Sie können alle vorhandenen EMR serverlosen Anwendungen auf der Seite „Anwendungen auflisten“ anzeigen. Sie können den Namen einer Anwendung wählen, um zur Detailseite für diese Anwendung zu gelangen.

Verwalten von Anwendungen

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

Starten Sie die Anwendung

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

Anwendung beenden

Wählen Sie diese Option, um eine Anwendung manuell zu beenden. Eine Anwendung sollte 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. Sie können beispielsweise das Release-Label für eine Anwendung ändern, um sie auf eine andere Version von Amazon zu aktualisierenEMR, oder Sie können die Architektur von x86_64 auf arm64 umstellen. Die anderen optionalen Einstellungen sind dieselben wie im Abschnitt Benutzerdefinierte Einstellungen auf der Seite Anwendung erstellen. 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 Anwendung auf der AWS CLI

Aus dem AWS CLI, können Sie einzelne Anwendungen erstellen, beschreiben und löschen. Sie können auch alle Ihre Anwendungen auflisten, sodass Sie sie auf einen Blick sehen können. In diesem Abschnitt wird beschrieben, wie Sie diese Aktionen ausführen. Weitere Anwendungsvorgänge, wie das Starten, Stoppen und Aktualisieren Ihrer Anwendung, finden Sie in der EMRServerless API Reference. Beispiele für die Verwendung von EMR Serverless API finden Sie unter AWS SDK for Java, siehe Java-Beispiele in unserem GitHub Repository. Beispiele für die Verwendung von EMR Serverless finden API Sie unter AWS SDK for Python (Boto), siehe Python-Beispiele 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 Namen und ARN 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 ihre `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

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

Themen

Grundlegendes zum Verhalten von Anwendungen

Standardverhalten der Anwendung

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 Arbeitsspeicher (GB) und Festplatte (GB) angeben. CPU

> **Note**
>
> Wir empfehlen, 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 Mitarbeiter mit 2vCPUs, 16

GB Arbeitsspeicher und 20 GB Festplatte beschränken möchten, legen Sie Ihre maximale Kapazität auf 100vCPUs, 800 GB für Arbeitsspeicher und 1000 GB für Festplatte fest.

Unterstützte Worker-Konfigurationen

Die folgende Tabelle zeigt die unterstützten Worker-Konfigurationen und -Größen, die Sie für EMR Serverless angeben können. Sie können je nach den Anforderungen Ihrer Arbeitslast unterschiedliche Größen für Treiber und Executoren konfigurieren.

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

CPU— Jeder Arbeiter kann 1, 2, 4, 8 oder 16 habenvCPUs.

Arbeitsspeicher — Jeder Worker verfügt über Arbeitsspeicher, angegeben in GB, innerhalb der in der vorherigen Tabelle aufgeführten Grenzwerte. 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 bei der Auswahl der Mitarbeitergröße berücksichtigen.

Wenn Sie beispielsweise 4 vCPUs für Ihre Worker-Instance und eine vorinitialisierte Speicherkapazität von 30 GB wählen, sollten Sie einen Wert von etwa 27 GB als Executor-Speicher für Ihren Spark-Job festlegen. Dadurch wird die Nutzung Ihrer vorinitialisierten Kapazität maximiert. Der nutzbare Arbeitsspeicher wäre 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

EMRServerless 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, können Sie den `initialCapacity` Parameter einer Anwendung auf die Anzahl der Worker festlegen, die Sie vorab initialisieren möchten. Bei vorinitialisierter Arbeitskapazität werden Jobs sofort gestartet. Dies ist ideal, wenn Sie iterative Anwendungen und zeitkritische Jobs implementieren möchten.

Wenn Sie einen Job einreichen 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 Jobs 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. Solange 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 können 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, sollten Sie alle Rechenkonfigurationen angeben, 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, können Sie angeben, wie viele Worker als Treiber und wie viele als Executoren starten. In ähnlicher Weise können Sie bei der Verwendung von Apache Hive angeben, 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 EMR Amazon-Version emr-6.6.0 basiert. Die Anwendung beginnt mit 5 vorinitialisierten Hive-Treibern mit jeweils 2 V CPU und 4 GB Arbeitsspeicher und 50 vorinitialisierten Tez-Task-Workern

mit jeweils 4 V und 8 GB Arbeitsspeicher. CPU Wenn Hive-Abfragen in dieser Anwendung ausgeführt werden, verwenden sie zunächst die vorinitialisierten Worker und beginnen sofort mit der Ausführung. Wenn alle vorinitialisierten Worker beschäftigt sind und mehr Hive-Jobs eingereicht werden, kann die Anwendung auf insgesamt 400 V CPU und 1024 GB Arbeitsspeicher skaliert werden. Sie können optional die Kapazität für den oder den `DRIVER` Worker weglassen. `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 basierend auf EMR Amazon-Version 6.6.0 ausführt. Die Anwendung beginnt mit 5 vorinitialisierten Spark-Treibern mit jeweils 2 V CPU und 4 GB Arbeitsspeicher und 50 vorinitialisierten Executoren mit jeweils 4 V und 8 GB Arbeitsspeicher. CPU Wenn Spark-Jobs in dieser Anwendung ausgeführt werden, verwenden sie zunächst die vorinitialisierten Worker und beginnen sofort mit der Ausführung. Wenn alle vorinitialisierten Worker beschäftigt sind und mehr Spark-Jobs eingereicht werden, kann die Anwendung auf insgesamt 400 V CPU und 1024 GB Arbeitsspeicher skaliert werden. Sie können optional die Kapazität für entweder den oder den `DRIVER` weglassen. `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 mit 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 Serverless EMR

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 unterSpeichern 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 erhalten Sie unter Parameter zum Überschreiben der Hive-Konfiguration und Parameter zur Überschreibung 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 auch Apache Log4j2-Eigenschaften konfigurieren, AWS Secrets Manager für Datenschutz und Java 17-Laufzeit auf Anwendungsebene.

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

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

```
            "Sid": "SecretsManagerPolicy",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue",
                "secretsmanager:DescribeSecret",
                "kms:Decrypt"
            ],
            "Resource": "arn:aws:secretsmanager:your-secret-arn"
        }
    ]
}
```

Weitere Informationen zum Erstellen benutzerdefinierter Richtlinien für geheime Daten finden Sie unter Beispiele für Berechtigungsrichtlinien für AWS Secrets Manager in der AWS Secrets Manager Benutzerleitfaden.

> **Note**
>
> Das `runtimeConfiguration`, was Sie auf Anwendungsebene angeben, `applicationConfiguration` entspricht dem `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://DOC-EXAMPLE-BUCKET-LOGGING/logs/app-level"
        },
        "managedPersistenceMonitoringConfiguration": {
            "enabled": false
        }
    }'
```

Überschreiben von Konfigurationen während eines Joblaufs

Sie können Konfigurationsüberschreibungen für die Anwendungskonfiguration und die Überwachungskonfiguration mit dem StartJobRunAPIangeben. EMRServerless führt dann die Konfigurationen zusammen, die Sie auf Anwendungs- und Auftragsebene angeben, um die Konfigurationen für die Auftragsausführung festzulegen.

Die Granularitätsstufe bei der Zusammenführung ist wie folgt:

- **ApplicationConfiguration**- Zum Beispiel `spark-defaults` Klassifizierungstyp.
- **MonitoringConfiguration**- Konfigurationstyp, zum Beispiels`3MonitoringConfiguration`.

> **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 Parameter zum Überschreiben der Hive-Konfiguration und. Parameter zur Überschreibung 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://DOC-EXAMPLE-BUCKET-OUTPUT/wordcount_output"]
        }
    }' \
    --configuration-overrides '{
        "applicationConfiguration": [
            {
                // Override existing configuration for spark-defaults in the
 application
                "classification": "spark-defaults",
                "properties": {
                    "spark.driver.cores": "2",
                    "spark.executor.cores": "1",
                    "spark.driver.memory": "4G",
                    "spark.executor.memory": "4G"
                }
            },
            {
                // Add configuration for spark-executor-log4j2
                "classification": "spark-executor-log4j2",
                "properties": {
                    "rootLogger.level": "error",
                    "logger.IdentifierForClass.name": "classpathForSettingLogger",
                    "logger.IdentifierForClass.level": "info"
                }
            },
            {
                // Remove existing configuration for spark-driver-log4j2 from the
 application
                "classification": "spark-driver-log4j2",
                "properties": {}
            }
        ],
```

```
        "monitoringConfiguration": {
            "managedPersistenceMonitoringConfiguration": {
                // Override existing configuration for managed persistence
                "enabled": true
            },
            "s3MonitoringConfiguration": {
                // Remove configuration of S3 monitoring
            },
            "cloudWatchLoggingConfiguration": {
                // Add configuration for CloudWatch logging
                "enabled": true
            }
        }
    }'
```

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

- Die Klassifizierung `spark-defaults` wird mit den auf Auftragsebene angegebenen Eigenschaften aktualisiert. Nur die in enthaltenen Eigenschaften `StartJobRun` würden für diese Klassifizierung 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 EMR serverlosen Images

Ab Amazon EMR 6.9.0 können Sie benutzerdefinierte Images verwenden, um Anwendungsabhängigkeiten und Laufzeitumgebungen mit Amazon EMR Serverless in einen einzigen Container zu packen. 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 EMR Amazon-Laufzeitumgebungen möglicherweise nicht allgemein verfügbar.
- Integriert EMR Serverless in die derzeit etablierten Erstellungs-, Test- und Bereitstellungsprozesse in Ihrem Unternehmen, einschließlich lokaler Entwicklungs- und Testprozesse.
- 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.

EMRServerless stellt Bilder bereit, die Sie als Grundlage verwenden können, wenn Sie Ihre eigenen Images erstellen. Das Basis-Image stellt die wesentlichen JAR-Dateien, Konfigurationen und Bibliotheken für die Interaktion des Images mit EMR Serverless bereit. 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 ECR Amazon-Repository in demselben AWS-Region das Sie verwenden, um EMR serverlose Anwendungen zu starten. Informationen zum Erstellen eines ECR privaten Amazon-Repositorys finden Sie unter Privates Repository erstellen.
2. Um Benutzern Zugriff auf Ihr ECR Amazon-Repository zu gewähren, fügen Sie Benutzern und Rollen, die EMR serverlose Anwendungen mit Bildern aus diesem Repository erstellen oder aktualisieren, die folgenden Richtlinien hinzu.

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

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

Schritt 1: Erstellen Sie ein benutzerdefiniertes Image aus serverlosen Basis-Images EMR

Erstellen Sie zunächst ein Dockerfile, das mit einer `FROM` Anweisung beginnt, die Ihr bevorzugtes Basis-Image verwendet. Nach der `FROM` Anweisung können Sie alle Änderungen hinzufügen, die Sie am Image vornehmen möchten. Das Basis-Image legt automatisch den Wert `USER` auf fest`hadoop`. Diese Einstellung verfügt möglicherweise 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 unterVerwenden 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 will run 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

EMRServerless 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 im 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
```

Sie sollten eine Ausgabe sehen, die der folgenden ähnelt.

```
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 ECR Amazon-Repository hoch

Übertragen Sie Ihr ECR Amazon-Image mit den folgenden Befehlen in Ihr ECR Amazon-Repository. Stellen Sie sicher, dass Sie über die richtigen IAM Berechtigungen verfügen, um das Bild in Ihr Repository zu übertragen. Weitere Informationen finden Sie unter Bild übertragen im ECRAmazon-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 das Symbol AWS Management Console Tab oder AWS CLI Klicken Sie auf die Tabulatortaste, 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 eine neue Anwendung. Folgen Sie den Anweisungen unter Anwendung erstellen.
2. Um benutzerdefinierte Images anzugeben, wenn Sie eine EMR serverlose 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 das ECR Amazon-Bild URI in das URI Feld Bild ein. EMRServerless verwendet dieses Image für alle Worker-Typen der Anwendung. Alternativ können Sie Verschiedene benutzerdefinierte Bilder auswählen und URIs für jeden Mitarbeitertyp ein anderes ECR Amazon-Bild einfügen.

CLI

- Erstellen Sie eine Anwendung mit dem `image-configuration` Parameter. EMRServerless 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 `worker-type-specifications` Parameter, um eine Anwendung mit unterschiedlichen Image-Einstellungen für jeden Worker-Typ zu erstellen.

```
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. EMRServerless 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 den Zugriff auf das benutzerdefinierte Image-Repository

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

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Emr Serverless Custom Image Support",
      "Effect": "Allow",
      "Principal": {
        "Service": "emr-serverless.amazonaws.com"
      },
      "Action": [
        "ecr:BatchGetImage",
        "ecr:DescribeImages",
        "ecr:GetDownloadUrlForLayer"
      ],
      "Condition":{
        "StringEquals":{
          "aws:SourceArn": "arn:aws:emr-serverless:region:aws-account-id:/
applications/application-id"
        }
      }
    }
  ]
```

```
}
```

Aus Sicherheitsgründen sollten Sie der Repository-Richtlinie einen `aws:SourceArn` Bedingungsschlüssel hinzufügen. Der IAM globale Bedingungsschlüssel `aws:SourceArn` stellt sicher, dass EMR Serverless das Repository nur für eine Anwendung ARN verwendet. Weitere Informationen zu ECR Amazon-Repository-Richtlinien finden Sie unter Privates Repository erstellen.

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.
- EMRServerless ignoriert unsere `[CMD]` `[ENTRYPOINT]` Anweisungen in der Docker-Datei. Verwenden Sie allgemeine Anweisungen in der Docker-Datei, z. B.`[COPY]`, und. `[RUN]` `[WORKDIR]`
- Sie sollten die Umgebungsvariablen`JAVA_HOME`,, nicht ändern `SPARK_HOME``HIVE_HOME`, `TEZ_HOME` wenn Sie ein benutzerdefiniertes Image erstellen.
- Benutzerdefinierte Bilder dürfen eine Größe von 5 GB nicht überschreiten.
- Wenn Sie Binärdateien oder JAR-Dateien in den EMR Amazon-Basis-Images ändern, kann dies zu Fehlern beim Starten von Anwendungen oder Jobs führen.
- Das ECR Amazon-Repository sollte sich im selben befinden AWS-Region das Sie verwenden, um EMR serverlose Anwendungen zu starten.

Zugriff konfigurieren VPC

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

> **Note**
>
> Sie müssen VPC den 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.

Anwendung erstellen

Auf der Seite Anwendung erstellen können Sie benutzerdefinierte Einstellungen auswählen und die Subnetze und Sicherheitsgruppen angebenVPC, die EMR serverlose Anwendungen verwenden können.

VPCs

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

Subnetze

Wählen Sie die Subnetze in dem ausVPC, das Ihren Datenspeicher enthält. Auf der Seite Anwendung erstellen werden alle Subnetze für die Datenspeicher in Ihrem aufgeführt. VPC

Bei den ausgewählten Subnetzen muss es sich um private Subnetze handeln. Das bedeutet, dass die zugehörigen Routing-Tabellen für die Subnetze keine Internet-Gateways haben sollten.

Für ausgehende Verbindungen zum Internet müssen die Subnetze über ausgehende Routen verfügen, die ein Gateway verwenden. NAT Informationen zur Konfiguration eines NAT Gateways finden Sie unter Arbeiten mit Gateways. NAT

Für die Amazon S3 S3-Konnektivität muss für die Subnetze ein NAT Gateway oder ein VPC Endpunkt konfiguriert sein. Informationen zur Konfiguration eines VPC S3-Endpunkts finden Sie unter Einen Gateway-Endpunkt erstellen.

Informationen zur Konnektivität zu anderen AWS-Services Außerhalb vonVPC, z. B. Amazon DynamoDB, müssen Sie entweder VPC Endpunkte oder ein Gateway konfigurieren. NAT Um Endpunkte zu konfigurieren für VPC AWS-Services, siehe Arbeiten mit VPC Endpunkten.

Mitarbeiter können VPC über ausgehenden Datenverkehr eine Verbindung zu den Datenspeichern innerhalb Ihres Netzwerks herstellen. Standardmäßig blockiert EMR Serverless den eingehenden Zugriff von Mitarbeitern, um die Sicherheit zu erhöhen.

Wenn Sie verwenden AWS Config, EMR Serverless erstellt für jeden Worker einen elastic network interface Interface-Elementdatensatz. 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 serverlosen 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”

Sicherheitsgruppen

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

> **Note**
>
> Wir empfehlen, dass Sie eine separate Sicherheitsgruppe für EMR serverlose Anwendungen erstellen. Dies macht das Isolieren und Verwalten von Netzwerkregeln effizienter. Um beispielsweise mit Amazon Redshift Redshift-Clustern zu kommunizieren, können Sie die Verkehrsregeln zwischen den Sicherheitsgruppen Redshift und EMR Serverless definieren, wie im folgenden Beispiel 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 EMR serverlosen Sicherheitsgruppen hinzu.

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

2. Fügen Sie eine Regel für ausgehenden Datenverkehr aus einer der EMR serverlosen Sicherheitsgruppen hinzu. Dafür stehen Ihnen zwei Optionen zur Verfügung: Zunächst können Sie ausgehenden Datenverkehr zu allen Ports öffnen.

Typ	Protocol (Protokoll)	Port-Bereich	Bestimmungsort
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
Alles TCP	TCP	5439	`redshift-security-group`

Anwendung konfigurieren

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

Details zur Auftragsausführung anzeigen

Auf der Detailseite der Auftragsausführung können Sie das Subnetz anzeigen, 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 einem Amazon darstellt. VPC Beispielsweise verfügt eine Netzwerkmaske VPC mit einer /16-Netzwerkmaske 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 EMR serverlose Anwendungen entworfen werden. Wenn Sie beispielsweise eine Anwendung haben, die 4 vCpu Worker anfordert und auf bis zu 4.000 skaliert werden kannvCpu, 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 der Maske /22 in einer Netzmaske VPC von /16, 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 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

Subnetz-ID	Subnetz-Adresse	Subnetzmaske	IP-Adressbereiche	Verfügbare IP-Adressen	Verwendbare IP-Adressen
3	10.0.8.0	255,255,252,0/22	10,0,4,0 - 10,0,7,255	1,024	1.019
4	10.0.12.0	255,255,252,0/22	10,0,12,0 - 10,0,15,255	1,024	1.019

Sie sollten abwägen, ob Ihr Workload am besten für größere Mitarbeiter geeignet ist. Für die Verwendung größerer Mitarbeiter sind weniger Netzwerkschnittstellen erforderlich. Wenn Sie beispielsweise 16 vCpu Worker mit einem Anwendungsskalierungslimit von 4.000 vCpu verwenden, sind maximal 250 Mitarbeiter für insgesamt 250 verfügbare IP-Adressen für die Bereitstellung von Netzwerkschnittstellen erforderlich. Sie benötigen Subnetze in mehreren Availability Zones mit einer Maskengröße von mindestens 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 über 3 Anwendungen verfügen, für die 4 vCpu Mitarbeiter erforderlich sind, und jede Anwendung auf bis zu 4000 skaliert werden kann, vCpu wobei ein auf Service basierendes Kontingent von 12.000 auf vCpu Kontoebene erforderlich ist, benötigt jedes Subnetz 3000 verfügbare IP-Adressen. Wenn dieVPC, die Sie verwenden möchten, nicht über eine ausreichende Anzahl von IP-Adressen verfügt, versuchen Sie, die Anzahl der verfügbaren IP-Adressen zu erhöhen. Sie können dies tun, indem Sie Ihrem zusätzliche Classless Inter-Domain Routing (CIDR) -Blöcke zuordnen. VPC Weitere Informationen finden Sie VPC im VPCAmazon-Benutzerhandbuch unter Zusätzliche IPv4 CIDR Blöcke mit Ihrem verknüpfen.

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

Optionen für die EMR serverlose Architektur von Amazon

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. EMRServerless aktualisiert automatisch

auf die neueste Generation von Instances, sobald diese verfügbar sind, sodass Ihre Anwendungen die neueren Instances ohne zusätzlichen Aufwand von Ihnen verwenden können.

Themen

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 Anwendungen. EMR 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 kompatibel istARM, können Sie zu arm64 wechseln, 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 kundenspezifisch entwickelt von AWS mit ARM 64-Bit-Neoverse-Kernen und Nutzung der Arm64-Architektur (auch bekannt als Arch64 oder 64-Bit). ARM Das Tool AWS Zu den auf EMR Serverless verfügbaren Prozessoren der Graviton-Reihe gehören 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. EMRServerless verwendet automatisch die neueste Generation von Prozessoren, sofern verfügbar, ohne dass Sie ein Upgrade auf die neueste Generation von Prozessoren 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 zu starten AWS CLI, geben Sie `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) verwenden, SDK AWS CLI, oder Studio. EMR

Um eine bestehende Anwendung von x86 nach arm64 zu konvertieren

1. Vergewissern Sie sich, dass Sie die neueste Hauptversion von verwenden AWS CLI/SDK, das 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 für die Verwendung von Graviton zu aktualisieren, geben Sie `ARM64` für den `architecture` Parameter im `update-application` API an.

   ```
   aws emr-serverless update-application \
    --application-id application-id \
    --architecture 'ARM64' \
   ```

```
 --region us-west-2
```

4. Um zu überprüfen, ob die CPU Architektur der Anwendung aktuell istARM64, verwenden Sie den `get-application`API.

```
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 serverlose Anwendung mit arm64 für Graviton-Unterstützung starten, sollten Sie Folgendes überprüfen.

Bibliothekskompatibilität

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

Weitere Informationen zur Konfiguration eines Spark- oder Hive-Workloads für die Verwendung von 64-Bit ARM finden Sie im AWS Graviton Getting Started Repository auf. GitHub Dieses Repository enthält wichtige Ressourcen, die Ihnen den Einstieg in das ARM auf Graviton basierende Spiel erleichtern können.

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

Mit den EMR Amazon-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 S3A Der Konnektor kann den `CreateSession` API aufrufen. Ein Beispielrichtlinie mit dieser Berechtigung finden Sie unter Erste Schritte mit S3 Express One Zone.
- S3A connector — Um Spark für den Zugriff auf Daten aus einem Amazon S3 S3-Bucket zu konfigurieren, der die Speicherklasse S3 Express One Zone verwendet, müssen Sie den Apache Hadoop-Connector verwenden S3A. Um den Connector zu verwenden, stellen Sie sicher, dass alle S3 das `s3a` Schema URIs verwenden. Wenn dies nicht der Fall ist, können Sie die Dateisystemimplementierung, die Sie für `s3`- und `s3n`-Schemata verwenden, ändern.

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 dem EMR Amazon-Release-Label 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.

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

5. Führen Sie Ihren Job aus. Beachten Sie, dass Sie das `S3A` Schema verwenden müssen, 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, können Sie Jobs für die Bewerbung einreichen. In diesem Abschnitt erfahren Sie, wie Sie den 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 reicht von `SUBMITTED` durchgehend `RUNNING` bis zum Erreichen von `FAILEDSUCCESS`, oder`CANCELLING`.

Aufträge können die folgenden Status haben:

Status	Beschreibung
Eingereicht	Der anfängliche Auftragsstatus, wenn Sie einen Job an EMR Serverless senden. Der Job wartet darauf, für die Anwendung geplant zu

Status	Beschreibung
	werden. EMRServerless beginnt, die Auftragsausführung zu priorisieren und zu planen.
Ausstehend	Der Scheduler bewertet die Auftragsausführung, um die Ausführung für die Anwendung zu priorisieren und zu planen.
Geplant	EMRServerless hat die Ausführung des Jobs für die Anwendung geplant und weist Ressourcen für die Ausführung des Jobs zu.
In Ausführung	EMRServerless 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	EMRServerless konnte den ausgeführten Job nicht an die Anwendung weiterleiten, oder er wurde erfolglos abgeschlossen. Weitere Informationen `StateDetails` zu diesem Auftragsfehler finden Sie unter.
Herzlichen Glückwunsch	Die Auftragsausführung wurde erfolgreich abgeschlossen.
Abbrechen	Der `CancelJobRun` API hat die Stornierung der Auftragsausführung angefordert, oder es wurde eine Zeitüberschreitung für die Ausführung des Auftrags festgestellt. EMRServerless versucht, den Job in der Anwendung abzubrechen und die Ressourcen freizugeben.

Status	Beschreibung
Abgebrochen	Die Auftragsausführung wurde erfolgreich abgebrochen und die verwendeten Ressourcen wurden freigegeben.

Jobs von der EMR Studio-Konsole aus ausführen

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

Übermitteln eines Auftrags

Auf der Seite Job einreichen können Sie einen Job wie folgt an eine EMR serverlose Anwendung senden.

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 übernehmen kann. Weitere Informationen zu Runtime-Rollen finden Sie unterJob-Runtime-Rollen für Amazon EMR Serverless.
3. Geben Sie im Feld Skriptspeicherort den Amazon S3 S3-Speicherort für das Skript einJAR, 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 Haupt 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, müssen Sie den Speicheraufwand berücksichtigen. Geben Sie die Werte für den Speicheraufwand in den Eigenschaften `spark.driver.memoryOverhead` und an. `spark.executor.memoryOverhead` 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 — Aktiviert oder deaktiviert die AWS Glue Sie Data Catalog als Metastore und ändern Sie die Einstellungen für das Anwendungsprotokoll. Weitere Informationen zu Metastore-Konfigurationen finden Sie unter. Metastore-Konfiguration Weitere Informationen zu den Optionen für die Anwendungsprotokollierung finden Sie unterSpeichern von Protokollen.
- 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 übernehmen kann.
3. Geben Sie im Feld Skriptspeicherort den Amazon S3 S3-Speicherort für das Skript einJAR, 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 AWS Glue Sie Data Catalog als Metastore und ändern Sie die Einstellungen für das Anwendungsprotokoll. Weitere Informationen zu Metastore-Konfigurationen finden Sie unter. Metastore-Konfiguration Weitere Informationen zu den Optionen für die Anwendungsprotokollierung finden Sie unterSpeichern von Protokollen.
- Tags — Weisen Sie der Anwendung beliebige benutzerdefinierte Tags zu.

5. Wählen Sie Auftrag absenden.

Anzeigen von Auftragsausführungen

Auf der Detailseite einer Anwendung auf der Registerkarte Auftragsausführungen können Sie Jobausführungen anzeigen und die folgenden Aktionen für Jobausführungen ausführen.

Job abbrechen — Um eine Jobausführung abzubrechen, die 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 erstellen, beschreiben und löschen auf AWS CLI. Sie können auch alle Ihre Jobs auflisten, um sie auf einen Blick zu sehen.

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. Stellen bei Spark Hive-Beispiele finden Sie unterJobs bei Hive. Dieser Befehl gibt Ihr`application-id`,ARN, und new `job-id` zurück.

Jeder ausgeführte Job hat eine festgelegte Timeout-Dauer. Wenn die Auftragsausführung diese Dauer überschreitet, wird sie von EMR Serverless automatisch storniert. Das Standard-Timeout beträgt 12 Stunden. Wenn Sie Ihre Auftragsausführung starten, können Sie diese Timeout-Einstellung auf einen Wert konfigurieren, 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://DOC-EXAMPLE-BUCKET/scripts/create_table.sql",
        "parameters": "--hiveconf hive.exec.scratchdir=s3://DOC-EXAMPLE-BUCKET/hive/
scratch --hiveconf hive.metastore.warehouse.dir=s3://DOC-EXAMPLE-BUCKET/hive/warehouse"
    }
   }' \
  --configuration-overrides '{
    "applicationConfiguration": [{
        "classification": "hive-site",
        "properties": {
            "hive.client.cores": "2",
            "hive.client.memory": "4GIB"
        }
    }]
}'
```

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 alle Ihre Jobs sehen möchten, können Sie die maximale Anzahl von Jobs angeben, die Sie sehen möchten, bis zu 50. Das folgende Beispiel gibt an, dass Sie Ihre letzten beiden Auftragsausführungen sehen 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 finden Sie im AWS CLI, siehe EMRServerless API Reference.

Verwenden von für Shuffle optimierten Festplatten

Mit EMR Amazon-Versionen 7.1.0 und höher können Sie für Shuffle optimierte Festplatten verwenden, wenn Sie Apache Spark- oder Hive-Jobs ausführen, um die Leistung für I/O-intensive Workloads zu verbessern. Im Vergleich zu Standardfestplatten bieten Shuffle-optimierte Festplatten einen höheren Wert IOPS (I/O-Operationen pro Sekunde) für schnellere Datenbewegungen und eine geringere Latenz bei Shuffle-Vorgängen. Durch die Shuffle-Optimierung können Sie Festplatten mit einer Größe von bis zu 2 TB pro Mitarbeiter hinzufügen, sodass Sie die Kapazität entsprechend Ihren Workload-Anforderungen konfigurieren können.

Wichtigste Vorteile

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

- Hohe IOPS Leistung — Für den Shuffle-Modus optimierte Festplatten bieten mehr Leistung IOPS als Standardfestplatten, was zu einer effizienteren und schnelleren Datenmischung bei Spark- und Hive-Aufträgen und anderen Workloads mit hohem Shuffle-Aufwand führt.
- Größere Festplattengröße — Für den Shuffle-Modus optimierte Festplatten unterstützen Festplattengrößen von 20 GB bis 2 TB pro Mitarbeiter, sodass Sie die für Ihre Workloads geeignete Kapazität auswählen können.

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 EMR folgenden Befehl eine serverlose Anwendung der Version 7.1.0.

```
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 enthält `spark.emr-serverless.driver.disk.type` und/oder dass er mit für `spark.emr-serverless.executor.disk.type` Shuffle optimierten Festplatten ausgeführt wird. 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 serverlose Anwendung der Version 7.1.0.

```
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 enthält `hive.driver.disk.type` und/oder dass er mit für Shuffle `hive.tez.disk.type` optimierten Festplatten ausgeführt wird. Sie können je nach Anwendungsfall 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 der EMR Amazon-Version 7.1.0 basieren. Diese Anwendungen haben die folgenden Eigenschaften:

- 5 vorinitialisierte Spark-Treiber mit jeweils 2 VCPU, 4 GB Arbeitsspeicher und 50 GB Shuffle-optimierter Festplatte.
- 50 vorinitialisierte Executoren mit jeweils 4 V, 8 GB Arbeitsspeicher und 500 GB an für CPU 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 V und 1024 GB Arbeitsspeicher. CPU 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"
}'
```

Jobs streamen

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, z. B. Analysen nahezu in Echtzeit, Betrugserkennung und Empfehlungsprogramme. EMRServerlose Streaming-Jobs bieten Optimierungen, wie z. B. integrierte Job-Resilienz, Echtzeitüberwachung, erweiterte Protokollverwaltung und Integration mit Streaming-Konnektoren.

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 — Sie können Streaming-Jobs verwenden, 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 Daten zu Benutzeraktivitäten 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 Sie Anomalieerkennung, vorausschauende Wartung und andere Anwendungsfälle für IoT-Analysen durchführen 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

Das Streamen von Jobs in EMR Serverless sorgt automatisch für Job-Resilienz, was eine Kombination der folgenden Faktoren ist:

- Automatische Wiederholung — EMR Serverless wiederholt automatisch alle fehlgeschlagenen Jobs, ohne dass Sie dazu manuell etwas eingeben müssen.
- 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

EMRServerless funktioniert 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 EMR Amazon-Versionen 7.1.0 und höher den Amazon Kinesis Data Streams-Connector, sodass Sie keine zusätzlichen Pakete erstellen oder herunterladen müssen.

- Unterstützte Ausgangsdatensenken — AWS Glue Data Catalog-Tabellen, Amazon S3, Amazon Redshift, MySQL, Postgre SQL Oracle, Oracle, Microsoft, Apache IcebergSQL, 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 EMRAmazon-Versionen 7.1.0 und höher unterstützt.
- EMRServerless geht davon aus, dass Streaming-Jobs über einen langen Zeitraum ausgeführt werden. Daher können Sie kein Ausführungs-Timeout festlegen, um die Laufzeit des Jobs zu begrenzen.
- Streaming-Jobs sind nur mit der Spark-Engine kompatibel, die auf dem strukturierten Streaming-Framework aufbaut.
- EMRServerless versucht auf unbestimmte Zeit, Streaming-Jobs erneut zu starten, 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 Jobresistenz.
- 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 Wiederherstellung nach Fehlern mit 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 auf Ihrer Anwendung Amazon EMR Version 7.1.0 oder höher ausgeführt werden muss.
2. Sobald Ihre Anwendung bereit ist, setzen Sie den `mode` Parameter auf `STREAMING` zum Senden eines Streaming-Jobs, ähnlich wie im 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-Konnektoren 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, müssen Sie die EMR Serverless-Anwendung mit VPC Zugriff konfigurieren und einen VPC Endpunkt verwenden, um privaten Zugriff zu ermöglichen. Oder Sie müssen ein NAT Gateway verwenden, um öffentlichen Zugriff zu erhalten. Weitere Informationen finden Sie unter Zugriff konfigurieren. VPC 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 GitHub der spark-sql-kinesis-connector Seite unter.

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-Connector 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, müssen Sie eine weitere Abhängigkeit angeben, mit der der Kafka-Connector eine Verbindung zu Amazon MSK herstellen kann. IAM Weitere Informationen finden Sie im aws-msk-iam-auth Repository unter. 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 verwenden zu können, müssen MSK Sie die EMR Serverless-Anwendung mit VPC Zugriff konfigurieren. Ihre Subnetze müssen über einen Internetzugang verfügen und ein NAT Gateway verwenden, um auf die Maven-Abhängigkeiten zuzugreifen. Weitere Informationen finden Sie unter Zugriff konfigurieren VPC. 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

Rotation der Protokolle

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 Überwachung und Instrumentierung in der Apache Spark-Dokumentation.

Stellen bei Spark

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 EMR Amazon-Release-

Version kompatibel ist. Wenn Sie beispielsweise Jobs mit Amazon EMR Release 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. Release-Versionen von Amazon EMR Serverless

Spark-Job-Parameter

Wenn Sie den verwenden `StartJobRun`API, um einen Spark-Job auszuführen, können Sie die folgenden Parameter angeben.

Erforderliche Parameter

- Runtime-Rolle für Spark-Jobs
- Treiberparameter für Spark-Jobs
- Parameter zur Überschreibung der Spark-Konfiguration
- Optimierung der dynamischen Ressourcenzuweisung in Spark

Runtime-Rolle für Spark-Jobs

Geben **`executionRoleArn`**Sie hier die ARN für die IAM Rolle an, 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 Skript oder Ihre Datei PySpark befindet JAR
- 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 Data Catalog, wenn du Spark verwendest SQL

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

Treiberparameter für Spark-Jobs

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, Java PySpark, SparkR 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 Haupt JAR - oder Python-Datei 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 Haupt JAR - oder Python-Datei, die Sie ausführen möchten. Wenn Sie Scala oder Java ausführenJAR, geben Sie `SparkSubmitParameters` mithilfe des `--class` Arguments die Haupteintragsklasse an.

Weitere Informationen finden Sie unter Starten von Anwendungen mit Spark-Submit.

Parameter zur Überschreibung 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 Amazon S3 URL (`s3MonitoringConfiguration`) anzugeben, in dem 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 das hostet Ihre Anwendung, und zwar in derselben AWS-Region wo dein Job läuft.
- **`applicationConfiguration`**— Um die Standardkonfigurationen für Anwendungen zu überschreiben, können Sie in diesem Feld ein Konfigurationsobjekt angeben. Sie können eine Kurzsyntax verwenden, um die Konfiguration bereitzustellen, oder Sie können das Konfigurationsobjekt in einer JSON Datei referenzieren. 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 Objekt angeben. JSON

> **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 `sparkSubmitParameters` `--conf` Arguments angeben.
- 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 für die Veröffentlichung EMR verwendet.
- Open-Source-Standardkonfigurationen für die Anwendung.

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

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 setzen, bedeutet das, dass EMR Serverless die Ressourcenzuweisung für Executoren optimiert, um die Geschwindigkeit, mit der Spark Executoren anfordert und storniert, besser an die Rate anzupassen, mit der Serverless Worker erstellt und freigibt. EMR Dadurch werden Workers bei EMR Serverless stufenübergreifend optimaler wiederverwendet, was zu niedrigeren Kosten führt, wenn Jobs mit mehreren Stufen ausgeführt werden, während die gleiche Leistung beibehalten wird.

Diese Eigenschaft ist in allen EMR Amazon-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, je nach Arbeitslast eine obere Skalierungsgrenze für Mitarbeiter zu konfigurieren, indem Sie entweder die Einstellung auf Auftragsebene `spark.dynamicAllocation.maxExecutors` oder die Einstellung für die maximale Kapazität auf Anwendungsebene verwenden.
- Bei einfacheren Aufträgen werden Sie möglicherweise keine Kostenverbesserung feststellen. 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 Arbeitskräfte 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.

Schlüssel	Beschreibung	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 Executoren je nach Arbeitslast nach oben oder unten.	`TRUE`

Schlüssel	Beschreibung	Standardwert
`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

Schlüssel	Beschreibung	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

Schlüssel	Beschreibung	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 basierte Verschlüsselung aktiviert. AES RPC 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.

Schlüssel	Beschreibung	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`

Schlüssel	Beschreibung	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`

Spark-Beispiele

Das folgende Beispiel zeigt, wie Sie mit dem `StartJobRun` API ein Python-Skript ausführen können. 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 EMRServerless 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://DOC-EXAMPLE-BUCKET-OUTPUT/wordcount_output"],
            "sparkSubmitParameters": "--conf spark.executor.cores=1 --conf
 spark.executor.memory=4g --conf spark.driver.cores=1 --conf spark.driver.memory=4g --
conf spark.executor.instances=1"
        }
```

```
    }'
```

Das folgende Beispiel zeigt, wie Sie den verwenden, `StartJobRun` API um einen 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"
        }
    }'
```

Jobs bei Hive

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 EMR Amazon-Release-Version kompatibel ist. Wenn Sie beispielsweise Jobs in einer Anwendung mit EMR Amazon-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. Release-Versionen von Amazon EMR Serverless

Hive-Auftragsparameter

Wenn Sie den verwenden `StartJobRun`API, um einen Hive-Job auszuführen, müssen Sie die folgenden Parameter angeben.

Erforderliche Parameter

- Runtime-Rolle für Hive-Jobs
- Hive-Job-Treiberparameter
- Parameter zum Überschreiben der Hive-Konfiguration

Runtime-Rolle für Hive-Jobs

Geben **`executionRoleArn`**Sie hier die 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 Data Catalog

Wenn Ihr Hive-Job Daten in oder aus anderen Datenquellen liest oder schreibt, geben Sie die entsprechenden Berechtigungen für diese 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 den Auftragstyp angeben`hive`, übergibt EMR Serverless eine Hive-Abfrage an den `jobDriver` Parameter. 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 Amazon S3 URL (`s3MonitoringConfiguration`) anzugeben, in dem der EMR Serverless-Job die Protokolle Ihres Hive-Jobs speichern soll. Stellen Sie sicher, dass Sie diesen Bucket mit demselben Code erstellen AWS-Konto das hostet Ihre Anwendung, und zwar in derselben AWS-Region wo dein Job läuft.
- **`applicationConfiguration`**— Sie können in diesem Feld ein Konfigurationsobjekt angeben, um die Standardkonfigurationen für Anwendungen zu überschreiben. Sie können eine Kurzsyntax verwenden, um die Konfiguration bereitzustellen, oder Sie können das Konfigurationsobjekt in einer JSON Datei referenzieren. 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 Objekt angeben. JSON

> **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 für die Version EMR zuweist.
- Standard-Open-Source-Konfigurationen für die Anwendung.

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

Eigenschaften von Hive-Jobs

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

Einstellung	Beschreibung
`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.

Einstellung	Beschreibung	Standardwert
`fs.s3.customAWSCre dentialsProvider`	Das Tool AWS Anbieter für Anmeldeinformationen, den Sie verwenden möchten.	com.amazonaws.auth .D efaultAWSCredentials ProviderChain
`fs.s3a.aws.credent ials.provider`	Das Tool AWS Anbieter für Anmeldeinformationen, den Sie mit einem S3A-Datei system verwenden möchten.	com.amazonaws.auth .D efaultAWSCredentials ProviderChain
`hive.auto.convert. join`	Option, die die automatische Konvertierung von allgemein en Joins in Mapjoin aktiviert, basierend auf der Größe der Eingabedatei.	`TRUE`

Einstellung	Beschreibung	Standardwert
`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`
`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`

Einstellung	Beschreibung	Standardwert
`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
`hive.driver.memory`	Die Menge an Arbeitsspeicher, die pro Hive-Treiberprozess verwendet werden soll. Der Hive CLI und der Tez Application Master teilen sich diesen Speicher zu gleichen Teilen mit 20% der verfügbaren Kapazität.	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.	

Einstellung	Beschreibung	Standardwert
`hive.exec.dynamic.partition`	Optionen, die dynamische Partitionen in/aktivieren. DML DDL	`TRUE`
`hive.exec.dynamic.partition.mode`	Option, die angibt, ob Sie den strikten Modus oder den nicht strikten Modus verwenden möchten. Im strikten Modus müssen Sie mindestens eine statische Partition angeben, 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
`hive.exec.max.dynamic.partitions.pernode`	Maximale Anzahl dynamischer Partitionen, die Hive in jedem Mapper- und Reducer-Knoten erstellt.	100

Einstellung	Beschreibung	Standardwert
`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`spezifiziert eine Auswahl der oben genannten Strategien, die auf Heuristiken basieren.	`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
`hive.exec.stagingdir`	Der Name des Verzeichnisses, in dem temporäre Dateien gespeichert werden, die Hive innerhalb von Tabellenpositionen und in dem in der Eigenschaft angegeben en Speicherort des Scratch-Verzeichnisses erstellt. `hive.exec.scratchdir`	`.hive-staging`

Einstellung	Beschreibung	Standardwert
`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. Auf einstellen, `HiveInputFormat` wenn Sie Probleme mit haben`CombineHiveInputFormat` .	`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`
`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`

Einstellung	Beschreibung	Standardwert
`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 eines Tez aktiviert. DAG	`FALSE`
`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 das Symbol AWS Glue Data Catalog fungiert als Metastore, läuft aber in einem anderen AWS-Konto als die Jobs, die ID des AWS-Konto wo die Jobs laufen.	`NULL`

Einstellung	Beschreibung	Standardwert
`hive.metastore.uris`	Die SparsamkeitURI, mit der der Metastore-Client eine Verbindung zum Remote-Metastore herstellt.	`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`
`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`

Einstellung	Beschreibung	Standardwert
`hive.stats.gather.num.threads`	Die Anzahl der Threads, die die Befehle `partialscan` und `noscan` analyze für partitionierte Tabellen verwenden. Dies gilt nur für Dateiformate, die `StatsProvidingRecordReader` (likeORC) implementieren.	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`
`hive.strict.checks.type.safety`	Option, die strenge Typsicherheitsprüfungen aktiviert und den `bigint` Vergleich von sowohl als auch deaktiviert. `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`

Einstellung	Beschreibung	Standardwert
`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
`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`

Einstellung	Beschreibung	Standardwert
`hive.vectorized.execution.reduce.enabled`	Option, die den vektorisierten Modus der Reduktionsseite einer Abfrageausführung aktiviert.	`TRUE`
`javax.jdo.option.ConnectionDriverName`	Der Name der Treiberklasse für einen Metastore. JDBC	`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`
`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, mit der die Bereinigung von Shuffle-Daten aktiviert wird, wenn der Vorgang abgeschlossen ist. DAG	`TRUE`

Einstellung	Beschreibung	Standardwert
`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 den Tez-App-Master weitergibt.	`INFO`
`tez.am.sleep.time.before.exit.millis`	EMRServerless sollte ATS Ereignisse nach diesem Zeitraum nach der Anfrage zum Herunterfahren am übertragen.	0
`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`

Einstellung	Beschreibung	Standardwert
`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. EMRAmazon-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`
`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

Einstellung	Beschreibung	Standardwert
`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-Taskspeichers berechnet
`tez.runtime.unordered.output.buffer.size-mb`	Die Größe des zu verwendenden Puffers, wenn Tez nicht direkt auf die Festplatte schreibt.	Der optimale Wert wird auf der Grundlage des Tez-Taskspeichers berechnet
`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 Verbindung). `ScatterGather`	0,25

Einstellung	Beschreibung	Standardwert
`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]`	
`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, in der AM warten sollte, bis Ereignisse gelöscht werden, bevor das System heruntergefahren wird.	300000

Hive-Jobbeispiele

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

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "hive": {
            "query": "s3://DOC-EXAMPLE-BUCKET/emr-serverless-hive/query/hive-query.ql",
            "parameters": "--hiveconf hive.log.explain.output=false"
        }
    }' \
    --configuration-overrides '{
        "applicationConfiguration": [{
            "classification": "hive-site",
            "properties": {
                "hive.exec.scratchdir": "s3://DOC-EXAMPLE-BUCKET/emr-serverless-hive/
hive/scratch",
```

```
                "hive.metastore.warehouse.dir": "s3://DOC-EXAMPLE-BUCKET/emr-
serverless-hive/hive/warehouse",
                "hive.driver.cores": "2",
                "hive.driver.memory": "4g",
                "hive.tez.container.size": "4096",
                "hive.tez.cpu.vcores": "1"
            }
        }]
    }'
```

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

EMRAusfallsicherheit für serverlose Job

EMRServerlose Versionen 7.1.0 und höher bieten Unterstützung für Job-Resilienz, sodass fehlgeschlagene Jobs automatisch ohne manuelles Eingreifen Ihrerseits wiederholt werden. Ein weiterer Vorteil der Job-Resilienz besteht darin, dass EMR Serverless Jobläufe 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 sowohl für Batch- als auch für Streaming-Jobs unterstützt, sodass Sie die Job-Resilienz an Ihren Anwendungsfall anpassen können. 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, sodass EMR	Streaming-Jobs unterstüt zen Checkpoints, sodass Sie die Streaming-Abfrage

	Stapelverarbeitungsaufträge (Batch jobs)	Jobs streamen
	Serverless den Job immer von Anfang an erneut ausführt.	so konfigurieren können, dass der Laufzeitstatus und der Fortschritt an einem Checkpoint-Speicherort in Amazon S3 gespeiche rt werden. EMRServerless setzt den vom Checkpoint ausgeführten Job fort. Weitere Informationen finden Sie in der Apache Spark-Dokumentation unter Wiederherstellung nach Fehlern mit Checkpointing.
Höchstzahl an Wiederhol ungsversuchen	Erlaubt maximal 10 Wiederhol ungen.	Streaming-Jobs verfügen über eine integrierte Steuerung zur Verhinderung von Datenmissbrauch, sodass die Anwendung die Wiederhol ung von Aufträgen beendet, wenn sie nach einer Stunde weiterhin fehlschlagen. Die Standardanzahl von Wiederhol ungen innerhalb einer Stunde beträgt fünf Versuche. Sie können diese Anzahl von Wiederholungen so konfiguri eren, 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.

Sie können die EMR API serverlosen Operationen oder die AWS CLI um die Resilienz von Jobs zu ändern oder Informationen zur Job-Resilienz einzusehen. Weitere Informationen finden Sie im EMRAPIServerless-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"
     }
}'
```

EMRServerless versucht auf unbestimmte Zeit erneut, Streaming-Jobs zu streamen, wenn sie fehlschlagen. Um zu verhindern, dass es aufgrund wiederholter nicht behebbarer Fehler zu einer Überlastung kommt, verwenden Sie die, um die Steuerung zur Verhinderung von Datenübertragungen für die Wiederholung von Streaming-Aufträgen `maxFailedAttemptsPerHour` zu konfigurieren. Mit diesem Parameter können Sie die maximal zulässige Anzahl fehlgeschlagener Versuche innerhalb einer Stunde angeben, bevor Serverless die Wiederholungsversuche beendet. EMR 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 Job-Ausführungsvorgänge verwenden, um Informationen über Jobs zu erhalten. Beispielsweise können Sie den `attempt` Parameter mit dem `GetJobRun` Vorgang verwenden, um Details zu einem bestimmten Auftragsversuch abzurufen. Wenn Sie den `attempt` Parameter nicht angeben, gibt der Vorgang 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 eine Datei und gibt sie zurückURL, mit der Sie auf die Anwendung zugreifen können, 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 Ereignis EMRServerless Job Run Retry hinzu. EMRServerless veröffentlicht dieses Ereignis bei jedem erneuten Versuch des Auftrags. 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 dieser 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 Jobs 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 Jobs an seinem eigenen dafür vorgesehenen Speicherort in Amazon S3 und CloudWatch veröffentlicht. Weitere Informationen finden Sie unter Speichern von Protokollen.

> **Note**
>
> EMRServerless verwendet dieses Präfixformat nur für alle Streaming-Jobs und alle Batch-Jobs, für die Wiederholungen aktiviert sind.

Metastore-Konfiguration

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:

- Das Tool AWS Glue Data Catalog
- Ein externer Apache Hive-Metastore

Verwendung der AWS Glue Sie den Datenkatalog als Metastore

Sie können Ihre Spark- und Hive-Jobs so konfigurieren, dass sie AWS Glue Sie Data Catalog als Metastore ein. 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 Auffüllen der AWS Datenkatalog Glue. Für Informationen über AWS Preise für Glue finden Sie unter AWS Preisgestaltung.

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

Konfigurieren Sie das AWS Glue Data Catalog

Um den Datenkatalog zu konfigurieren, wählen Sie aus, welche Art von EMR serverloser Anwendung Sie verwenden möchten.

Spark

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

Wenn Sie oder verwenden SDKs AWS CLI, Sie können die `spark.hadoop.hive.metastore.client.factory.class` Konfiguration `com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory` in den `sparkSubmit` Parametern Ihres Joblaufs auf einstellen. 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://DOC-EXAMPLE-BUCKET/code/pyspark/extreme_weather.py",
            "sparkSubmitParameters": "--conf
 spark.hadoop.hive.metastore.client.factory.class=com.amazonaws.glue.catalog.metastore.AWSGl
```

```
 --conf spark.driver.cores=1 --conf spark.driver.memory=3g --conf
 spark.executor.cores=4 --conf spark.executor.memory=3g"
        }
    }'
```

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 serverlose 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 in derselben Datei auf AWS-Konto wie Ihre Anwendung. Sie benötigen keine virtuelle private Cloud (VPC), um den Datenkatalog als Metastore zu verwenden.

Um auf die Hive-Metastore-Tabellen zuzugreifen, fügen Sie die erforderlichen AWS Die Richtlinien von Glue werden unter IAM Berechtigungen einrichten für beschrieben AWS Glue.

Konfigurieren Sie den kontoübergreifenden Zugriff für EMR Serverless und AWS Glue Data Catalog

Um den kontenübergreifenden Zugriff für EMR Serverless einzurichten, müssen Sie sich zunächst bei folgenden Geräten anmelden AWS-Konten:

- `AccountA`— Ein AWS-Konto wo Sie eine EMR serverlose Anwendung erstellt haben.
- `AccountB`— Ein AWS-Konto das enthält ein AWS Glue Data Catalog, auf den Ihr EMR Serverless-Job zugreifen soll.

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

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

2. Fügen Sie der Runtime-Rolle für EMR serverlose Jobs eine IAM Richtlinie hinzu, `AccountA` sodass diese Rolle auf Datenkatalogressourcen in zugreifen kann. `AccountB`

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

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

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

   Spark

   Legen Sie die `spark.hadoop.hive.metastore.glue.catalogid` Eigenschaft in der `hive-site` Klassifizierung fest, wie im folgenden Beispiel gezeigt. Ersetzen *Konto B-Katalog-ID* mit der ID des Datenkatalogs in. `AccountB`

```
aws emr-serverless start-job-run \
--application-id "application-id" \
--execution-role-arn "job-role-arn" \
--job-driver '{
    "sparkSubmit": {
        "query": "s3://DOC-EXAMPLE-BUCKET/hive/scripts/create_table.sql",
        "parameters": "--hiveconf hive.exec.scratchdir=s3://DOC-EXAMPLE-BUCKET/
hive/scratch --hiveconf hive.metastore.warehouse.dir=s3://DOC-EXAMPLE-BUCKET/
hive/warehouse"
    }
  }' \
```

```
--configuration-overrides '{
    "applicationConfiguration": [{
        "classification": "hive-site",
        "properties": {
            "spark.hadoop.hive.metastore.glue.catalogid": "AccountB-catalog-id"
        }
    }]
}'
```

Hive

Stellen Sie die `hive.metastore.glue.catalogid` Eigenschaft in der `hive-site` Klassifizierung ein, wie im folgenden Beispiel gezeigt. Ersetzen *Konto B-Katalog-ID* mit der 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://DOC-EXAMPLE-BUCKET/hive/scripts/create_table.sql",
    "parameters": "--hiveconf hive.exec.scratchdir=s3://DOC-EXAMPLE-BUCKET/hive/
scratch --hiveconf hive.metastore.warehouse.dir=s3://DOC-EXAMPLE-BUCKET/hive/
warehouse"
    }
}' \
--configuration-overrides '{
    "applicationConfiguration": [{
        "classification": "hive-site",
        "properties": {
            "hive.metastore.glue.catalogid": "AccountB-catalog-id"
        }
    }]
}'
```

Überlegungen bei der Verwendung von AWS Glue Data Catalog

Sie können Ihren JARs `ADD JAR` Hive-Skripten Hilfsprogramme hinzufügen. Weitere Überlegungen finden Sie unter Überlegungen bei der Verwendung AWS Datenkatalog Glue.

Verwenden eines externen Hive-Metastores

Sie können Ihre EMR serverlosen Spark- und Hive-Jobs so konfigurieren, dass sie eine Verbindung zu einem externen Hive-Metastore wie Amazon Aurora oder Amazon for My herstellen. RDS SQL In diesem Abschnitt wird beschrieben, wie Sie einen Amazon RDS Hive-Metastore einrichten, Ihre EMR serverlosen Jobs konfigurieren und so konfigurierenVPC, dass sie einen externen Metastore verwenden.

Erstellen Sie einen externen Hive-Metastore

1. Erstellen Sie eine Amazon Virtual Private Cloud (AmazonVPC) mit privaten Subnetzen, indem Sie den Anweisungen unter Erstellen einer VPC folgen.
2. Erstellen Sie Ihre EMR serverlose Anwendung mit Ihren neuen Amazon VPC - und privaten Subnetzen. Wenn Sie Ihre EMR Serverless-Anwendung mit a konfigurierenVPC, 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 Ihres finden Sie VPC unterZugriff konfigurieren VPC.
3. Erstellen Sie eine My SQL - oder Aurora SQL Postgre-Datenbank in einem privaten Subnetz in Ihrem Amazon. VPC Informationen zum Erstellen einer RDS Amazon-Datenbank finden Sie unter RDSAmazon-DB-Instance erstellen.
4. Ändern Sie die Sicherheitsgruppe Ihrer My SQL - oder Aurora-Datenbank, um JDBC Verbindungen von Ihrer EMR Serverless-Sicherheitsgruppe aus zuzulassen, indem Sie die Schritte unter Ändern einer RDS Amazon-DB-Instance befolgen. Fügen Sie der RDS Sicherheitsgruppe eine Regel für eingehenden Datenverkehr aus einer Ihrer EMR serverlosen Sicherheitsgruppen hinzu.

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

Spark-Optionen konfigurieren

Verwenden JDBC

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

```
aws emr-serverless start-job-run \
  --application-id "application-id" \
  --execution-role-arn "job-role-arn" \
  --job-driver '{
        "sparkSubmit": {
            "entryPoint": "s3://DOC-EXAMPLE-BUCKET/scripts/spark-jdbc.py",
            "sparkSubmitParameters": "--jars s3://DOC-EXAMPLE-BUCKET/mariadb-connector-
java.jar
            --conf
 spark.hadoop.javax.jdo.option.ConnectionDriverName=org.mariadb.jdbc.Driver
            --conf spark.hadoop.javax.jdo.option.ConnectionUserName=<connection-user-
name>
            --conf spark.hadoop.javax.jdo.option.ConnectionPassword=<connection-
password>
            --conf spark.hadoop.javax.jdo.option.ConnectionURL=<JDBC-Connection-
string>
            --conf spark.driver.cores=2
            --conf spark.executor.memory=10G
            --conf spark.driver.memory=6G
            --conf spark.executor.cores=4"
        }
    }' \
    --configuration-overrides '{
        "monitoringConfiguration": {
        "s3MonitoringConfiguration": {
            "logUri": "s3://DOC-EXAMPLE-BUCKET/spark/logs/"
        }
    }
}'
```

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

```
from os.path import expanduser, join, abspath
from pyspark.sql import SparkSession
from pyspark.sql import Row
# warehouse_location points to the default location for managed databases and tables
```

```
warehouse_location = abspath('spark-warehouse')
spark = SparkSession \
    .builder \
    .config("spark.sql.warehouse.dir", warehouse_location) \
    .enableHiveSupport() \
    .getOrCreate()
spark.sql("SHOW DATABASES").show()
spark.sql("CREATE EXTERNAL TABLE `sampledb`.`sparknyctaxi`(`dispatching_base_num`
 string, `pickup_datetime` string, `dropoff_datetime` string, `pulocationid` bigint,
 `dolocationid` bigint, `sr_flag` bigint) STORED AS PARQUET LOCATION 's3://<s3 prefix>/
nyctaxi_parquet/'")
spark.sql("SELECT count(*) FROM sampledb.sparknyctaxi").show()
spark.stop()
```

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 My SQL - oder Amazon Aurora My-Instance basiert. SQL Führen Sie dazu einen Thrift-Server auf dem Master-Knoten eines vorhandenen EMR Amazon-Clusters aus. Diese Option ist ideal, wenn Sie bereits über einen EMR Amazon-Cluster mit einem Thrift-Server verfügen, den Sie verwenden möchten, um Ihre EMR serverlosen Jobkonfigurationen zu vereinfachen.

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

```
}'
```

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 JDBC

Wenn Sie einen externen Hive-Datenbankspeicherort auf einer Amazon RDS My SQL - 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 My SQL oder Amazon Aurora M basiert. ySQLinstance Führen Sie dazu einen Thrift-Server auf dem Hauptknoten eines vorhandenen EMR Amazon-Clusters aus. Diese Option ist ideal, wenn Sie bereits über einen EMR Amazon-Cluster verfügen, auf dem ein Thrift-Server ausgeführt wird, und Sie Ihre EMR serverlosen Jobkonfigurationen verwenden möchten.

Legen Sie die folgenden Konfigurationen in der `hive-site` Klassifizierung fest, sodass EMR Serverless auf den Remote-Thrrift-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"
    }
}
```

Überlegungen bei der Verwendung eines externen Metastores

- Sie können Datenbanken, die mit MariaDB kompatibel sind, JDBC als Ihren Metastore konfigurieren. Beispiele für diese Datenbanken sind RDS für MariaDBSQL, My 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.
- EMRServerless unterstützt keine Kerberos-Authentifizierung. Sie können keinen Thrift-Metastore-Server mit Kerberos-Authentifizierung für serverlose Spark- oder Hive-Jobs verwenden. EMR

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

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

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

Voraussetzungen

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

- **`AccountA`**— Das ist der AWS Konto, in dem Sie eine Amazon EMR Serverless-Anwendung erstellt haben. Bevor Sie den kontoübergreifenden Zugriff einrichten, müssen Sie für dieses Konto Folgendes bereithalten:
  - Eine Amazon EMR Serverless-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`**— Das ist 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 zuzugreifen in account B from account A, hängen Sie die folgende Richtlinie an den S3-Bucket in an account B.

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

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

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

Eine weitere Möglichkeit, den kontoübergreifenden 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. Sie können EMR Serverless und Amazon S3 mit den temporären Sicherheitsanmeldedaten API aufrufen, mit `AssumeRole` denen Sie erstellt haben.

Die folgenden Schritte veranschaulichen, wie Sie mithilfe einer angenommenen Rolle von Serverless aus EMR auf kontoübergreifende S3-Daten zugreifen können:

1. Erstellen Sie einen Amazon S3 S3-Bucket, *`cross-account-bucket`*, im`AccountB`. Weitere Informationen finden Sie unter Bucket erstellen im Amazon Simple Storage Service-Benutzerhandbuch. Wenn Sie kontenübergreifenden Zugriff auf DynamoDB haben möchten, können Sie auch eine DynamoDB-Tabelle in `AccountB` erstellen. Weitere Informationen finden Sie unter Erstellen einer DynamoDB-Tabelle im Amazon DynamoDB DynamoDB-Entwicklerhandbuch.
2. Erstellen Sie eine `Cross-Account-Role-B` IAM Rolle in, die Zugriff auf die `AccountB` *`cross-account-bucket`*.
   a. Melden Sie sich an bei AWS Management Console 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 IAMRollen erstellen im IAM Benutzerhandbuch.
   c. Erstellen Sie eine IAM Richtlinie, die die `Cross-Account-Role-B` Zugriffsberechtigungen für den *`cross-account-bucket`* S3-Bucket, wie die folgende Richtlinienerklärung zeigt. Hängen Sie dann die IAM Richtlinie an an`Cross-Account-Role-B`. Weitere Informationen finden Sie im IAMBenutzerhandbuch unter IAMRichtlinien erstellen.

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

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

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

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

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

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.

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

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

4. Grant `Job-Execution-Role-A` in `AccountA` der AWS STS `AssumeRole`Erlaubnis zur Annahme`Cross-Account-Role-B`.

   a. In der IAM Konsole für AWS Konto`AccountA`, wählen Sie`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.

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

Beispiele für angenommene Rollen

Sie können eine einzelne angenommene Rolle verwenden, um auf alle S3-Ressourcen in einem Konto zuzugreifen, oder mit Amazon EMR 6.11 und höher können Sie mehrere IAM Rollen konfigurieren, 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 `anspark.hadoop.fs.s3.customAWSCredentialsProvider=com.amazonaws.emr.AssumeRole`
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 in den Hive-Treibern, dem Tez-Anwendungsmaster und den Tez-Task-Containern anzugeben.

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

Spark

Das folgende Beispiel zeigt, wie eine angenommene Rolle verwendet wird, um eine EMR serverlose Spark-Auftragsausführung 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":
 "spark.hadoop.fs.s3.customAWSCredentialsProvider=com.amazonaws.emr.AssumeRoleAWSCredentials
                "spark.emr-serverless.driverEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN":
 "arn:aws:iam::AccountB:role/Cross-Account-Role-B",
                "spark.executorEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN":
 "arn:aws:iam::AccountB:role/Cross-Account-Role-B"
            }
        }]
    }'
```

Hive

Das folgende Beispiel zeigt, wie eine angenommene Rolle verwendet wird, um einen EMR Serverless Hive-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 '{
        "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 EMR Serverless-Versionen 6.11.0 und höher 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 beim Starten der Jobausführung die folgenden Konfigurationen:

1. Geben Sie die EMRFS Konfiguration `fs.s3.customAWSCredentialsProvider` für an`com.amazonaws.emr.serverless.credentialsprovider.BucketLevelAssumeRoleCreden`
2. Geben Sie die EMRFS Konfiguration an`fs.s3.bucketLevelAssumeRoleMapping`, um die Zuordnung vom S3-Bucket-Namen zur IAM Rolle in Konto B zu definieren, die übernommen werden soll. Der Wert sollte das Format von haben`bucket1->role1;bucket2->role2`.

Sie können beispielsweise für den Zugriff auf `arn:aws:iam::AccountB:role/Cross-Account-Role-B-1` den Bucket `bucket1` und für den `arn:aws:iam::AccountB:role/Cross-Account-Role-B-2` Zugriff auf den Bucket verwenden`bucket2`. Die folgenden Beispiele zeigen, wie Sie eine EMR serverlose Auftragsausführung mit kontenübergreifendem Zugriff über mehrere angenommene Rollen starten.

Spark

Das folgende Beispiel zeigt, wie mehrere angenommene Rollen verwendet werden, um eine EMR serverlose 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 können.

Themen

Fehler: Das Limit für die maximal zulässige Kapazität wurde überschritten.

Dieser Fehler weist darauf hin, dass EMR Serverless den Job nicht senden konnte, da die Anwendung Ihre konfigurierten maximalen Kapazitätsgrenzen überschritten hat. Erhöhen Sie die maximalen Kapazitätsgrenzen für die Anwendung.

Fehler: Die konfigurierte maximale Kapazität wurde überschritten. Bitte versuchen Sie es später noch einmal.

Dieser Fehler weist darauf hin, dass EMR Serverless keinen neuen Job starten konnte, da die Anwendung Ihre konfigurierten maximalen Kapazitätsgrenzen überschritten hat. Erhöhen Sie die maximalen Kapazitätsgrenzen für die Anwendung.

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 für die Rolle keine Vertrauensstellung für EMR serverlose Berechtigungen besteht. Um zu überprüfen, ob die IAM Rolle existiert und ob Sie die Vertrauensrichtlinie für die Rolle ordnungsgemäß eingerichtet haben, finden Sie die Anweisungen unterJob-Runtime-Rollen für Amazon EMR Serverless.

Führen Sie interaktive Workloads mit EMR Serverless über Studio aus EMR

Übersicht

Eine interaktive Anwendung ist eine EMR serverlose Anwendung, für die interaktive Funktionen aktiviert sind. Mit interaktiven Amazon EMR Serverless-Anwendungen können Sie interaktive Workloads mit Jupyter-Notebooks ausführen, die in Amazon Studio verwaltet werden. EMR 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 nutzen 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 verwenden zu können, müssen Sie die folgenden Anforderungen erfüllen:

- EMRServerlose 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

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:

Um auf Ihre interaktive Anwendung zuzugreifen

Richten Sie Benutzer- und Workspace-Berechtigungen für EMR Studio ein. Weitere Informationen finden Sie unter Configure EMR Studio-Benutzerberechtigungen 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 die folgenden zusätzlichen Berechtigungen für die Studio-Benutzer 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 serverlosen 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 serverlosen Anwendung herzustellen.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EMRServerlessInteractiveAccess",
            "Effect": "Allow",
            "Action": "emr-serverless:AccessInteractiveEndpoints",
            "Resource": "arn:aws:emr-serverless:Region:account:/applications/*"
        },
        {
            "Sid": "EMRServerlessRuntimeRoleAccess",
            "Effect": "Allow",
```

```
            "Action": "iam:PassRole",
            "Resource": "interactive-execution-role-ARN",
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": "emr-serverless.amazonaws.com"
                }
            }
        }
    ]
}
```

Konfiguration interaktiver Anwendungen

Verwenden Sie die folgenden allgemeinen Schritte, um eine EMR serverlose Anwendung mit interaktiven Funktionen von Amazon EMR Studio in der AWS Management Console.

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

Wenn Sie eine Anwendung an einen Studio-Arbeitsbereich 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.

Überlegungen zu interaktiven Anwendungen

- EMRServerlose interaktive Anwendungen werden mit Amazon EMR 6.14.0 und höher unterstützt.
- EMRStudio ist der einzige Client, der in EMR serverlose interaktive Anwendungen integriert ist. Die folgenden EMR Studio-Funktionen werden bei EMR serverlosen interaktiven Anwendungen 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, 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 während der gesamten Anwendung eine vorinitialisierte Kapazität von einem Kernel-Worker bei, auch wenn Sie keine vorinitialisierte Kapazität für Treiber angeben. Jeder Kernel-Worker verwendet 4 V CPU und 16 GB Arbeitsspeicher. Aktuelle Preisinformationen finden Sie auf der Seite mit den EMRAmazon-Preisen.
- Sie müssen über ein ausreichendes CPU v-Service-Kontingent in Ihrem AWS-Konto um interaktive Workloads auszuführen. Wenn Sie keine Lake Formation-fähigen Workloads ausführen, empfehlen wir mindestens 24 v. CPU In diesem Fall empfehlen wir mindestens 28 V. CPU
- EMRServerless beendet automatisch die Kernel von den Notebooks, wenn sie länger als 60 Minuten inaktiv waren. EMRServerless berechnet die Leerlaufzeit des Kernels 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 EMR serverlose Anwendung erstellen. Weitere Informationen zur Aktivierung von Lake Formation in EMR Serverless finden Sie unter Lake Formation in Amazon EMR aktivieren.

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

Mit EMR Amazon-Versionen 6.14.0 und höher können Sie einen Apache Livy-Endpunkt erstellen und aktivieren und gleichzeitig eine EMR serverlose Anwendung erstellen 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 Benutzeroberfläche von Apache Livy verwalten. REST
- Verwenden Sie die Apache REST API Livy-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

Zusätzlich zu den erforderlichen Berechtigungen für den Zugriff auf EMR Serverless müssen Sie Ihrer IAM Rolle auch die folgenden Berechtigungen hinzufügen, 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 vom Apache Livy-Endpunkt aus verfügbaren REST API Operationen auszuführen.

- `iam:PassRole`— erteilt die Erlaubnis, während der Erstellung der Apache Livy-Sitzung auf die IAM Ausführungsrolle zuzugreifen. EMRServerless verwendet diese Rolle, um Ihre Workloads auszuführen.
- `emr-serverless:GetDashboardForJobRun`— erteilt die Erlaubnis, die Links zur Spark Live-Benutzeroberfläche und zu den Treiberprotokollen zu generieren, und gewährt Zugriff auf die Protokolle als Teil der Apache Livy-Sitzungsergebnisse.

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

Erste Schritte

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`, können Sie auf den Apache Livy-Endpunkt zugreifen.

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

3. Verwenden Sie FolgendesURL, um auf den Endpunkt zuzugreifen:

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

Sobald der Endpunkt bereit ist, können Sie Workloads auf der Grundlage Ihres Anwendungsfalls einreichen. 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:

- HTTPClient — Sie müssen Ihre Apache API Livy-Endpunktoperationen mit einem benutzerdefinierten HTTP Client einreichen.
- Sparkmagic-Kernel — Sie müssen den Sparkmagic-Kernel lokal ausführen und interaktive Abfragen mit Jupyter-Notebooks einreichen.

HTTPKunden

Um eine Apache Livy-Sitzung zu erstellen, müssen Sie `emr-serverless.session.executionRoleArn` den `conf` Parameter Ihres Anfragetextes angeben. 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 API Livy-Operationen beschrieben.

APIOperation	Beschreibung
GET/sitzungen	Gibt eine Liste aller aktiven interaktiven Sitzungen zurück.
POST/Sessions	Erzeugt eine neue interaktive Sitzung über Spark oder Pyspark.
GET/sessions/ <*sessionId* >	Gibt die Sitzungsinformationen zurück.
GET/sessions/ <*sessionId* /sessions//state	Gibt den Status der Sitzung zurück.
DELETE/Sitzungen/ <*sessionId* >	Stoppt und löscht die Sitzung.
GET/sessions/ <*sessionId* /sessions// statements	Gibt alle Anweisungen in einer Sitzung zurück.
POST/sessions/ <*sessionId* /sessions// statements	Führt eine Anweisung in einer Sitzung aus.
GET/sessions/ <*sessionId* /sessionen/ aussagen/<*statementId* >	Gibt die Details der angegebenen Anweisung in einer Sitzung zurück.
POST/sessions/ <*sessionId* /sessionen/ aussagen/<*statementId* >/statements/storn ieren	Bricht die angegebene Anweisung in dieser Sitzung ab.

Anfragen an den Apache Livy-Endpunkt senden

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

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

```
pip3 install botocore awscrt requests
```

Im Folgenden finden Sie ein Python-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 die Konfiguration vorgenommen haben AWS Anmeldeinformationen in der Instanz, in der Sie Sparkmagic installieren möchten

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

   ```
   pip install emr-serverless-customauth
   ```
4. Geben Sie nun den Pfad zum benutzerdefinierten Authentifikator und zum Apache Livy-Endpunkt URL in der JSON-Datei der Sparkmagic-Konfiguration 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 API Livy-Operationen über den konfigurierten Apache Livy-Endpunkt auf. URL Anschließend können Sie Ihre Anweisungen entsprechend Ihrem Anwendungsfall schreiben.

Überlegungen

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

- EMRServerless behält die Isolierung auf Sitzungsebene mithilfe des Anrufer-Prinzipals bei. Der Anruferprinzipal, der die Sitzung erstellt, ist der einzige, der auf diese Sitzung zugreifen kann. Für eine detailliertere Isolierung können Sie eine Quellidentität konfigurieren, wenn Sie Anmeldeinformationen annehmen. In diesem Fall erzwingt EMR Serverless eine Isolierung auf Sitzungsebene, die sowohl auf dem Anruferprinzipal als auch auf der Quellidentität basiert. 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 CPU v-Service-Kontingent in Ihrem AWS-Konto um interaktive Workloads mit dem Apache Livy-Endpunkt auszuführen. Wir empfehlen mindestens 24 v. CPU
- Das standardmäßige Apache Livy-Sitzungstimeout beträgt 1 Stunde. Wenn Sie Anweisungen eine Stunde lang nicht ausführen, löscht Apache Livy die Sitzung und gibt den Treiber und die Executoren frei. Sie können diese Konfiguration nicht ändern.
- 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 EMR serverloser Anwendungen und Jobs. Sie sollten Überwachungsdaten aus allen Teilen Ihrer EMR serverlosen Lösungen sammeln, damit Sie einen etwaigen Ausfall an mehreren Punkten leichter debuggen können.

Themen

Speichern von Protokollen

Um Ihren Auftragsfortschritt auf EMR Serverless zu überwachen und Auftragsfehler zu beheben, können Sie wählen, wie EMR Serverless Anwendungsprotokolle speichert und bereitstellt. Wenn Sie eine Auftragsausführung einreichen, können Sie Managed Storage, Amazon S3 und Amazon CloudWatch als Protokollierungsoptionen angeben.

Mit CloudWatch können Sie die Protokolltypen und Protokollspeicherorte angeben, die Sie verwenden möchten, oder die Standardtypen und Speicherorte akzeptieren. Weitere Informationen zu CloudWatch Protokollen finden Sie unterthe section called “Amazon CloudWatch”. Bei verwaltetem Speicher und S3-Protokollierung zeigt die folgende Tabelle die Protokollspeicherorte und die Verfügbarkeit der Benutzeroberfläche, 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	Im verwalteten 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 verwendenUIs.

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, EMR kann Amazon Ihre Jobs nicht in Ihrem Namen beheben.

Um diese Option in EMR Studio zu deaktivieren, deaktivieren Sie die Option Zulassen AWS um Protokolle 30 Tage lang aufzubewahren, aktivieren Sie das Kontrollkästchen im Bereich Zusätzliche Einstellungen auf der Seite „Job einreichen“.

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

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

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. *DOC-EXAMPLE-BUCKET-LOGGING*Ersetzen Sie es durch den Namen Ihres Logging-Buckets.

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

So richten Sie einen Amazon S3 S3-Bucket zum Speichern von Protokollen aus dem 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://DOC-EXAMPLE-BUCKET-LOGGING/logs/"
        }
    }
}
```

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

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

EMRServerless 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>/'
```

Logging für EMR Serverless mit Amazon CloudWatch

Wenn Sie einen Job für eine EMR serverlose Anwendung einreichen, können Sie Amazon CloudWatch als Option zum Speichern Ihrer Bewerbungsprotokolle wählen. Auf diese Weise können Sie CloudWatch Protokollanalysefunktionen wie CloudWatch Logs Insights und Live Tail verwenden. Sie können Protokolle auch von CloudWatch anderen Systemen streamen, z. B. OpenSearch zur weiteren Analyse.

EMRServerless bietet Echtzeitprotokollierung für Treiberprotokolle. Sie können die Protokolle in Echtzeit mit der CloudWatch Live-Tail-Funktion oder über CloudWatch CLI Tail-Befehle anzeigen.

Standardmäßig ist die CloudWatch Protokollierung für EMR Serverless deaktiviert. Informationen zur Aktivierung finden Sie in AWS CLI der Konfiguration unter.

> **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 Rate der Transaktionen pro Sekunde (TPS) für zu niedrig ist`PutLogEvents`. 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.

```
{
```

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

AWS CLI

So richten Sie Amazon CloudWatch zum Speichern von Protokollen für EMR Serverless vom AWS CLI, verwenden Sie die `cloudWatchLoggingConfiguration` Konfiguration, wenn Sie einen Joblauf starten. Stellen Sie dazu die folgenden Konfigurationsüberschreibungen bereit. Optional können Sie auch einen Protokollgruppennamen, einen Protokollstream-Präfixnamen, Protokolltypen und einen Verschlüsselungsschlüssel ARN angeben.

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`*.

EMRServerlose Versionen 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 gezeigt, 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 Sie angeben können, wenn Sie die CloudWatch Amazon-Protokollierung für EMR Serverless aktivieren. Die unterstützten `logTypes` Werte sind ebenfalls unter diesem 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 Treiberprotokolle von stdout und stderr. CloudWatch Wenn Sie andere Protokolle wünschen, können Sie im Feld eine Container-Rolle und die entsprechenden Protokolltypen angeben. `logTypes`

Die folgende Liste zeigt die unterstützten Worker-Typen, die Sie für die `logTypes` Konfiguration angeben können:

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 EMR Amazon-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 hat keine Auswirkungen auf die in den S3-Bucket hochgeladenen Protokolle.

Spark-Anwendungsprotokolle

> **Note**
>
> Die Rotation des Spark-Anwendungsprotokolls ist für alle EMR Amazon-Release-Labels verfügbar.

EMRServerless rotiert auch die Spark-Anwendungsprotokolle für Treiber und Executoren, wie z. B. Dateien und. `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 die älteren rotierten Protokolle zu sehen, müssen Sie beim Speichern von Protokollen einen Amazon S3 S3-Standort angeben. Weitere Informationen finden Sie unter Logging for EMR Serverless with Amazon S3 Buckets.

Die neuesten Protokolldateien finden Sie am folgenden Speicherort. EMRServerless 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.

EMRServerlose Versionen 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, damit Sie die Protokolle besser verfolgen und voneinander unterscheiden können. Dieser Pfad enthält alle rotierten Protokolle.

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

Protokolle verschlüsseln

Verschlüsselung EMR serverloser Protokolle 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"
```

```
        }
    }
}
```

Verschlüsseln von EMR serverlosen 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 einen Job-Lauf einreichen.

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

EMRServerlose Logs mit Amazon verschlüsseln CloudWatch

Um Logs 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 IAM eine Richtlinie

Die folgende IAM Richtlinie bietet die Berechtigungen für `kms:GenerateDataKey` und`kms:Decrypt`:

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

```
}
```

Um die Spark- oder Tez-Benutzeroberfläche zu starten, müssen Sie Ihren Benutzern, Gruppen oder Rollen die `emr-serverless:GetDashboardForJobRun` API folgenden Zugriffsberechtigungen erteilen:

```
{
 "Version": "2012-10-17",
  "Statement": {
     "Effect": "Allow",
     "Action": [
           "emr-serverless:GetDashboardForJobRun"
      ]
   }
}
```

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, müssen Sie die KMS Schlüsselberechtigungen wie folgt konfigurieren.

Der `emr-serverless.amazonaws.com` Principal muss in der Richtlinie für den Schlüssel über die folgenden Berechtigungen verfügen: KMS

```
{
    "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 Schlüsselrichtlinie einen `aws:SourceArn` KMS Bedingungsschlüssel hinzuzufügen. Der IAM globale Bedingungsschlüssel `aws:SourceArn` trägt dazu bei, dass EMR Serverless den KMS Schlüssel nur für eine Anwendung ARN verwendet.

Die Job-Runtime-Rolle muss in ihrer IAM Richtlinie über die folgenden Berechtigungen verfügen:

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

Berechtigungen für Verschlüsselungsschlüssel für Amazon CloudWatch

Verwenden Sie die folgende IAM Richtlinie für die Job-Runtime-Rolle, ARN um den KMS Schlüssel Ihrer Protokollgruppe zuzuordnen.

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

Konfigurieren Sie die KMS Schlüsselrichtlinie, um Amazon KMS Berechtigungen zu gewähren CloudWatch:

```
{
```

```
    "Version": "2012-10-17",
    "Id": "key-default-1",
    "Statement":
     {
        "Effect": "Allow",
        "Principal": {
            "Service": "logs.AWS-Region.amazonaws.com"
        },
        "Action": [
            "kms:Decrypt",
            "kms:GenerateDataKey",
        ],
        "Resource": "*",
        "Condition": {
            "ArnLike": {
                "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:AWS-
Region:111122223333:*"
            }
        }
     }
}
```

Apache Log4j2-Eigenschaften für Amazon Serverless konfigurieren EMR

Auf dieser Seite wird beschrieben, wie Sie benutzerdefinierte Apache Log4j 2.x-Eigenschaften für serverlose Jobs unter konfigurieren. EMR `StartJobRun` Informationen zur Konfiguration von Log4j-Klassifizierungen auf Anwendungsebene finden Sie unter. Standardanwendungskonfiguration für Serverless EMR

Konfigurieren Sie die Spark Log4j2-Eigenschaften für Amazon Serverless EMR

Mit EMR Amazon-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 bei Ihren Spark-Jobs auf Serverless. EMR Verwenden Sie die `spark-executor-log4j2` Klassifizierungen `spark-driver-log4j2` und, 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 beim Absenden des Jobs finden Sie unter. Standardanwendungskonfiguration für Serverless EMR

```
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()
```

Um Log4j2 für den Treiber anzupassen, wenn Sie einen Spark-Job ausführen, können Sie die folgende Konfiguration verwenden:

```
{
   "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()
  }
}
```

Um Log4j2 für den Treiber anzupassen, wenn Sie einen Spark-Job ausführen, können Sie die folgende Konfiguration verwenden:

```
{
   "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 Sie konfigurieren können, finden Sie in der Datei unter. `log4j2.properties.template` GitHub

Serverlose Überwachung EMR

In diesem Abschnitt wird beschrieben, wie Sie Ihre Amazon EMR Serverless-Anwendungen und -Jobs überwachen können.

Themen

- Überwachung EMR serverloser Anwendungen und Jobs

- Überwachen Sie Spark-Metriken mit Amazon Managed Service für Prometheus
- EMRMetriken zur Nutzung serverloser Server

Überwachung EMR serverloser Anwendungen und Jobs

Mit Amazon CloudWatch Metrics for EMR Serverless können Sie CloudWatch Metriken von einer Minute abrufen und auf CloudWatch Dashboards zugreifen, um den near-real-time Betrieb und die Leistung Ihrer EMR serverlosen Anwendungen zu überprüfen.

EMRServerless sendet Metriken in jeder Minute. CloudWatch EMRServerless gibt diese Metriken sowohl auf Anwendungsebene als auch auf Job-, Worker-Typ- und Arbeitsebene aus. capacity-allocation-type

Verwenden Sie zunächst die EMR CloudWatch Serverless-Dashboard-Vorlage, die im Serverless-Repository bereitgestellt wird, und stellen Sie EMR sie bereit. GitHub

> **Note**
>
> EMRBei serverlosen interaktiven 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, können Sie die Protokolle und die Apache Spark-Benutzeroberfläche von Ihrem Studio-Arbeitsbereich aus einsehen. EMR

In der folgenden Tabelle werden die EMR Serverless-Dimensionen beschrieben, die im Namespace verfügbar sind. `AWS/EMRServerless`

Dimensionen für serverlose Metriken EMR

Dimension	Beschreibung	
`ApplicationId`	Filtert nach allen Metriken einer EMR serverlosen Anwendung.	
`JobId`	Filtert nach allen Metriken einer EMR serverlosen Jobausführung.	

Dimension	Beschreibung	
`WorkerType`	Filtert nach allen Metriken eines bestimmten Worker-Typs. 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 EMR serverloser Anwendungsebene mit CloudWatch Amazon-Metriken überwachen. Sie können auch eine einzige Ansicht einrichten, um die Kapazitätsnutzung von Anwendungen in einem CloudWatch Dashboard zu überwachen.

EMRMetriken für serverlose Anwendungen

Metrik	Beschreibung	Primäre Dimension	Sekundäre Dimension
`CPUAllocated`	Die Gesamtzahl der vCPUs zugewiesenen.	`ApplicationId`	`ApplicationId` , `WorkerType` , `CapacityAllocationType`
`IdleWorkerCount`	Die Gesamtzahl der untätigen Arbeitnehmer.	`ApplicationId`	`ApplicationId` , `WorkerType` , `CapacityAllocationType`

Metrik	Beschreibung	Primäre Dimension	Sekundäre Dimension
MaxCPUAllowed	Das für die Anwendung CPU zulässige Maximum.	ApplicationId	N/A
MaxMemoryAllowed	Der maximal zulässige Arbeitsspeicher in GB für die Anwendung.	ApplicationId	N/A
MaxStorageAllowed	Der maximal zulässige Speicherplatz in GB für die Anwendung.	ApplicationId	N/A
MemoryAllocated	Der gesamte zugewiesene Speicher in GB.	ApplicationId	ApplicationId , WorkerType , CapacityAllocationType
PendingCreationWorkerCount	Die Gesamtzahl der Mitarbeiter, deren Erstellung noch aussteht.	ApplicationId	ApplicationId , WorkerType , CapacityAllocationType
RunningWorkerCount	Die Gesamtzahl der Mitarbeiter, die von der Anwendung verwendet werden.	ApplicationId	ApplicationId , WorkerType , CapacityAllocationType
StorageAllocated	Der gesamte zugewiesene Festplattenspeicher in GB.	ApplicationId	ApplicationId , WorkerType , CapacityAllocationType

Metrik	Beschreibung	Primäre Dimension	Sekundäre Dimension
`TotalWorkerCount`	Die Anzahl der insgesamt verfügbaren Mitarbeiter.	`ApplicationId`	`ApplicationId` , `WorkerType` , `CapacityAllocationType`

Überwachung auf Arbeitsebene

Amazon EMR Serverless sendet die folgenden Messwerte auf Auftragsebene an Amazon CloudWatch jede Minute. Sie können die Metrikwerte für aggregierte Auftragsausführungen nach Status der Auftragsausführung anzeigen. Die Einheit für jede der Metriken ist Anzahl.

EMRMetriken auf Jobebene ohne Server

Metrik	Beschreibung	Primäre Dimension
`SubmittedJobs`	Die Anzahl der Jobs mit dem Status „Eingereicht“.	`ApplicationId`
`PendingJobs`	Die Anzahl der Jobs mit dem Status Ausstehend.	`ApplicationId`
`ScheduledJobs`	Die Anzahl der Jobs mit dem Status „Geplant“.	`ApplicationId`
`RunningJobs`	Die Anzahl der Jobs im Status Wird ausgeführt.	`ApplicationId`
`SuccessJobs`	Die Anzahl der Jobs im Status Erfolgreich.	`ApplicationId`
`FailedJobs`	Die Anzahl der Jobs mit dem Status Fehlgeschlagen.	`ApplicationId`
`CancellingJobs`	Die Anzahl der Jobs mit dem Status „Storniert“.	`ApplicationId`

Metrik	Beschreibung	Primäre Dimension
`CancelledJobs`	Die Anzahl der Jobs mit dem Status Storniert.	`ApplicationId`

Sie können Engine-spezifische Messwerte sowohl für laufende als auch für abgeschlossene EMR serverlose Jobs mit einer Engine-spezifischen Anwendung überwachen. UIs Wenn Sie die Benutzeroberfläche für einen laufenden Job aufrufen, sehen Sie die Benutzeroberfläche der Live-Anwendung mit Aktualisierungen in Echtzeit. Wenn Sie die Benutzeroberfläche für einen abgeschlossenen Job aufrufen, sehen Sie die persistente App-Benutzeroberfläche.

Ausführen von Aufgaben

Für Ihre laufenden EMR serverlosen Jobs können Sie sich eine Echtzeit-Oberfläche ansehen, 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 zuzugreifenUIs, verwenden Sie die EMR Studio-Konsole oder fordern Sie einen sicheren URL Endpunkt mit AWS Command Line Interface.

Abgeschlossene Jobs

Für Ihre abgeschlossenen EMR serverlosen Jobs können Sie den Spark History Server oder die Persistent Hive Tez-Benutzeroberfläche verwenden, um Jobdetails, Phasen, Aufgaben und Metriken für Spark- oder Hive-Jobausführungen anzuzeigen. Um auf diese zuzugreifenUIs, verwenden Sie die EMR Studio-Konsole oder fordern Sie einen sicheren Endpunkt mit dem URL AWS Command Line Interface.

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 EMRServerless sammelt während der Auftragsausführung Datenpunkte von einzelnen Mitarbeitern auf Auftragsebene, Mitarbeitertyp und Ebene. capacity-allocation-type Sie können diese Dimension verwenden`ApplicationId`, um mehrere Jobs zu überwachen, die zu derselben Anwendung gehören.

EMRServerloser Job — Metriken auf Mitarbeiterebene

Metrik	Beschreibung	Einheit	Primäre Dimension	Sekundäre Dimension
`WorkerCpuAllocated`	Die Gesamtzahl der CPU V-Cores, die den Arbeitern in einem Joblauf zugewiesen wurden.	None	`JobId`	`ApplicationId`, `WorkerType` und `CapacityAllocationType`
`WorkerCpuUsed`	Die Gesamtzahl der CPU V-Cores, die von Arbeitern in einem Joblauf verwendet wurden.	None	`JobId`	`ApplicationId`, `WorkerType` und `CapacityAllocationType`
`WorkerMemoryAllocated`	Die Gesamtspeicherkapazität in GB, die Workern in einer Auftragsausführung zugewiesen wurde.	Gigabyte (GB)	`JobId`	`ApplicationId`, `WorkerType` und `CapacityAllocationType`
`WorkerMemoryUsed`	Der gesamte Arbeitsspeicher in GB, der von Arbeitern bei einer Auftragsausführung genutzt wird.	Gigabyte (GB)	`JobId`	`ApplicationId`, `WorkerType` und `CapacityAllocationType`

Metrik	Beschreibung	Einheit	Primäre Dimension	Sekundäre Dimension
`WorkerEphemeralStorageAllocated`	Die Anzahl der Byte an flüchtigem Speicher, die Workern in einer Auftragsausführung zugewiesen wurden.	Gigabyte (GB)	`JobId`	`ApplicationId`, `WorkerType` und `CapacityAllocationType`
`WorkerEphemeralStorageUsed`	Die Anzahl der Byte an flüchtigem Speicher, die von Arbeitern bei einer Auftragsausführung verwendet werden.	Gigabyte (GB)	`JobId`	`ApplicationId`, `WorkerType` und `CapacityAllocationType`
`WorkerStorageReadBytes`	Die Anzahl der Byte, die von Arbeitern während einer Auftragsausführung aus dem Speicher gelesen wurden.	Bytes	`JobId`	`ApplicationId`, `WorkerType` und `CapacityAllocationType`

Metrik	Beschreibung	Einheit	Primäre Dimension	Sekundäre Dimension
`WorkerStorageWriteBytes`	Die Anzahl der Byte, die während einer Auftragsausführung von Workern in den Speicher geschrieben wurden.	Bytes	`JobId`	`ApplicationId`, `WorkerType` und `CapacityAllocationType`

In den folgenden Schritten wird beschrieben, wie Sie die verschiedenen Arten von Metriken anzeigen können.

Console

So greifen Sie mit der Konsole auf die Benutzeroberfläche Ihrer Anwendung zu

1. Navigieren Sie zu Ihrer EMR serverlosen Anwendung im EMR Studio. Folgen Sie den Anweisungen unter Erste Schritte von der Konsole aus.
2. So zeigen Sie Engine-spezifische Anwendungen UIs und Protokolle für einen laufenden Job an:
   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 die Spark-Engine-Logs anzuzeigen, navigieren Sie in der Spark-Benutzeroberfläche zur Registerkarte Executors und wählen Sie den Link Logs für den Treiber. Um Hive-Engine-Logs anzuzeigen, wählen Sie DAG in der Hive Tez-Benutzeroberfläche den entsprechenden Link Logs aus.
3. So zeigen Sie maschinenspezifische Anwendungen UIs und Protokolle für einen abgeschlossenen Job an:

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 die Spark-Engine-Logs anzuzeigen, navigieren Sie in der Spark-Benutzeroberfläche zur Registerkarte Executors und wählen Sie den Link Logs für den Treiber. Um Hive-Engine-Logs anzuzeigen, wählen Sie DAG in der Hive Tez-Benutzeroberfläche den entsprechenden Link Logs aus.

AWS CLI

Um auf die Benutzeroberfläche Ihrer Anwendung zuzugreifen, verwenden Sie AWS CLI

- Um eine zu generierenURL, mit der Sie auf die Benutzeroberfläche Ihrer Anwendung sowohl für laufende als auch für abgeschlossene Jobs zugreifen können, rufen Sie den auf `GetDashboardForJobRun`API.

```
aws emr-serverless get-dashboard-for-job-run /
--application-id <application-id> /
--job-run-id <job-id>
```

DasURL, was Sie generieren, ist eine Stunde lang gültig.

Überwachen Sie Spark-Metriken mit Amazon Managed Service für Prometheus

Mit den EMR Amazon-Versionen 7.1.0 und höher können Sie EMR Serverless in Amazon Managed Service for Prometheus integrieren, um Apache Spark-Metriken für EMR serverlose Jobs und Anwendungen zu sammeln. Diese Integration ist verfügbar, wenn Sie einen Job einreichen oder eine Bewerbung erstellen, indem Sie entweder AWS Konsole, EMR Serverless API oder 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 den für Endpoint URL angezeigten Wert — Remote Write. URL Sie müssen das angeben, URL wenn Sie Ihre EMR serverlose 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>"
}
```

Aufstellen

Um das AWS Konsole zum Erstellen einer Anwendung, 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 Ihren Remote-Schreibvorgang an. URL
4. Geben Sie alle anderen gewünschten Konfigurationseinstellungen an und wählen Sie dann Anwendung erstellen und starten aus.

Verwenden Sie AWS CLI oder EMR Serverlos API

Sie können auch das verwenden AWS CLI oder EMR ServerlessAPI, um Ihre EMR serverlose Anwendung in Amazon Managed Service for Prometheus zu integrieren, wenn Sie die Befehle oder die `create-application` Befehle ausführen. `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

EMRServerless 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 einreichen mit. `PrometheusMonitoringConfiguration`

In der folgenden Tabelle werden alle Eigenschaften beschrieben, die Sie konfigurieren können, wenn Sie einen Spark-Job einreichen, der Metriken an Amazon Managed Service for Prometheus sendet.

Spark-Eigenschaft	Standardwert	Beschreibung
`spark.metrics.conf .*.sink.prometheus Servlet.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.so urce. JvmSource	Die Klasse, die Spark verwendet, um wichtige Metriken von der zugrunde liegenden virtuellen Java-Maschine zu sammeln und zu senden. Um das Sammeln von JVM Metriken zu beenden, deaktivieren Sie diese Eigenschaft, indem Sie sie auf eine leere Zeichenfo lge setzen, z. "" B. Um das Standardverhalten zu überschreiben, geben Sie

Spark-Eigenschaft	Standardwert	Beschreibung
		Ihre eigene benutzerdefinierte Klasse an.
`spark.metrics.conf.driver.sink.prometheusServlet.path`	/metrics/prometheus	Die eindeutige KennungURL, die Amazon Managed Service for Prometheus verwendet, um Metriken vom Fahrer 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 KennungUR L, die Amazon Managed Service for Prometheus verwendet, um Metriken vom Testamentsvollstrecker zu sammeln. Um das Standardv erhalten 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 für Prometheus mit EMR Serverless ist nur verfügbar in 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 sich für eine kleinere Mitarbeitergröße entscheiden, z. B. einen CPU V-Worker, kann sich Ihre Job-Laufzeit verlängern.
- Support für die Verwendung von Amazon Managed Service für Prometheus mit EMR Serverless ist nur für EMR Amazon-Versionen 7.1.0 und höher verfügbar.

EMRMetriken zur Nutzung serverloser Server

Sie können die CloudWatch Amazon-Nutzungsmetriken verwenden, um einen Überblick über die Ressourcen zu erhalten, die Ihr Konto verwendet. Verwenden Sie diese Kennzahlen, um Ihre Servicenutzung in CloudWatch Grafiken und Dashboards zu visualisieren.

EMRMetriken zur serverlosen Nutzung entsprechen den 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-Servicekontingenten finden Sie unterEndpunkte und Kontingente für EMR Serverless.

Messdaten zur Nutzung von Servicekontingenten für Serverless EMR

EMRServerless veröffentlicht die folgenden Messdaten zur Nutzung von Dienstkontingenten im `AWS/Usage` Namespace.

Metrik	Beschreibung
`ResourceCount`	Die Gesamtanzahl der angegebenen Ressource, die auf Ihrem Konto ausgeführt

Metrik	Beschreibung
	wird. Die Ressource wird durch die Dimension en definiert, die der Metrik zugeordnet sind.

Dimensionen für Kennzahlen zur Nutzung von Kontingenten für EMR serverlose Dienste

Sie können die folgenden Dimensionen verwenden, um die von EMR Serverless veröffentlichten Nutzungsmetriken zu verfeinern.

Dimension	Wert	Beschreibung
`Service`	EMRServerlos	Der Name des AWS-Service das enthält die Ressource.
`Type`	Ressource	Der Entitätstyp, den EMR Serverless meldet.
`Resource`	v CPU	Der Ressourcentyp, den EMR Serverless verfolgt.
`Class`	None	Die Ressourcenklasse, die EMR Serverless verfolgt.

Automatisieren von Serverless mit EMR Amazon EventBridge

Sie können Folgendes verwenden … Amazon EventBridge um deine zu automatisieren AWS-Services und reagieren automatisch auf Systemereignisse wie Probleme mit der Anwendungsverfügbarkeit oder Ressourcenänderungen. EventBridge liefert nahezu in Echtzeit einen Stream von Systemereignissen, die Änderungen in Ihrem AWS Ressourcen schätzen. 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:

- Rufen Sie eine auf AWS Lambda function
- Ein Ereignis an Amazon Kinesis Data Streams weiterleiten

- Aktiviere ein AWS Step Functions Zustandsautomat
- Ein SNS Amazon-Thema oder eine SQS Amazon-Warteschlange benachrichtigen

Wenn Sie beispielsweise EMR Serverless verwenden EventBridge , können Sie eine aktivieren AWS Lambda funktioniert, wenn ein ETL Job erfolgreich ist, oder benachrichtigt ein SNS Amazon-Thema, wenn ein ETL Job fehlschlägt.

EMRServerless sendet drei 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-Versionen 7.1.0 und höher ausgelöst werden.

Beispiel für serverlose Ereignisse EMR 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 „Job ausführen und wiederholen“

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"
    }
}
```

Taggen von -Ressourcen

Sie können jeder Ressource mithilfe von Tags Ihre eigenen Metadaten zuweisen, um Ihre EMR serverlosen Ressourcen zu verwalten. Dieser Abschnitt bietet einen Überblick über die Tag-Funktionen und zeigt Ihnen, wie Sie Tags erstellen.

Themen

Was ist ein Tag?

Ein Tag ist eine Bezeichnung, die Sie einem zuweisen AWS Ressource. Jedes Tag besteht aus einem Schlüssel und einem Wert, die Sie beide selbst definieren können. Mithilfe von Tags können Sie Ihre kategorisieren AWS Ressourcen nach Attributen wie Zweck, Eigentümer und Umgebung. Wenn Sie viele Ressourcen desselben Typs haben, können Sie bestimmte Ressourcen basierend auf den zugewiesenen Tags schnell bestimmen. Sie können beispielsweise eine Reihe von Tags für Ihre Amazon EMR Serverless-Anwendungen definieren, um den Besitzer und die Stack-Ebene jeder Anwendung zu verfolgen. Sie sollten für jeden Ressourcentyp einen konsistenten Satz von Tag-Schlüsseln entwickeln.

Tags werden nicht automatisch Ihren Ressourcen zugewiesen. Nachdem Sie einer Ressource ein Tag hinzugefügt haben, können Sie den Wert eines Tags ändern oder das Tag jederzeit 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 verwendenIAM, können Sie steuern, welche Benutzer in Ihrem AWS Konten haben die Erlaubnis, Tags zu verwalten. Beispiele für Tag-basierte Zugriffssteuerungsrichtlinien finden Sie unter Richtlinien für Tag-basierte Zugriffskontrolle.

Markieren Ihrer -Ressourcen

Sie können neue oder bestehende Anwendungen und Jobausführungen taggen. Wenn Sie Amazon EMR Serverless verwendenAPI, AWS CLI, oder ein AWS SDK, Sie können Tags auf neue Ressourcen anwenden, indem Sie den `tags` Parameter für die entsprechende API Aktion verwenden. Mithilfe der `TagResource` API Aktion können Sie Tags auf vorhandene 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.

Ressource	Unterstützt Tags (Markierungen)	Unterstützt Tag-Propagierung	Unterstützt das Tagging bei der Erstellung (Amazon EMR ServerlessAPI, AWS CLI, und AWS SDK)	APIzur 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 das Tagging bei der Erstellung (Amazon EMR ServerlessAPI, AWS CLI, und AWS SDK)	APIzur 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 Schlüssellänge beträgt 128 Unicode-Zeichen in UTF -8.
- Die maximale Wertelänge beträgt 256 Unicode-Zeichen in UTF -8.
- Zulässige Zeichen sind Buchstaben, Zahlen, Leerzeichen, die in UTF -8 dargestellt werden können, 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 `AWS:` Sie weder für Schlüssel noch für Werte eine Kombination von Groß- oder Kleinbuchstaben, z. B. als Präfix. Diese sind nur reserviert für AWS benutzen.

Arbeiten mit Tags unter Verwendung der AWS CLI und Amazon EMR Serverless API

Verwenden Sie Folgendes AWS CLI Befehle oder Amazon EMR API Serverless-Operationen zum Hinzufügen, Aktualisieren, Auflisten und Löschen der Tags für Ihre Ressourcen.

Ressource	Unterstützt Tags (Markierungen)	Unterstützt Tag-Propagierung
Hinzufügen oder Überschreiben eines oder mehrerer Tags (Markierung)	`tag-resource`	`TagResource`
Listet Tags für eine Ressource auf	`list-tags-for-resource`	`ListTagsForResource`
Löschen eines oder mehrerer Tags (Markierung)	`untag-resource`	`UntagResource`

Die folgenden Beispiele zeigen, wie Sie Ressourcen mit dem Tag 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 häufige Anwendungsfälle beschrieben, wenn Sie mit EMR serverlosen Anwendungen arbeiten.

Themen

Verwenden von Java 17 mit Amazon EMR Serverless

Mit EMR Amazon-Versionen 6.11.0 und höher können Sie EMR serverlose Spark-Jobs so konfigurieren, dass sie die Java 17-Laufzeit für die Java Virtual Machine () verwenden. JVM 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, können Sie die `JAVA_HOME` Einstellung für die zugehörigen Klassifizierungen `spark.emr-serverless.driverEnv` und `spark.executorEnv` die Umgebungsklassifizierungen angeben.

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/"
            }
         }
      ]
}
```

Apache Hudi mit EMR Serverless verwenden

Um Apache Hudi mit EMR serverlosen 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
   spark.serializer=org.apache.spark.serializer.KryoSerializer
   ```

2. Um eine Hudi-Tabelle mit dem konfigurierten Katalog zu synchronisieren, geben Sie entweder AWS Glue Sie Data Catalog als Ihren Metastore oder konfigurieren Sie einen externen Metastore. EMRServerless unterstützt `hms` als Synchronisierungsmodus für Hive-Tabellen für Hudi-Workloads. EMRServerless aktiviert diese Eigenschaft standardmäßig. Weitere Informationen zum Einrichten Ihres Metastores finden Sie unter. Metastore-Konfiguration

   > **Important**
   >
   > EMRServerless unterstützt `HIVEQL` keine Optionen für den Synchronisierungsmodus für Hive-Tabellen zur Verarbeitung von Hudi-Workloads. `JDBC` Weitere Informationen finden Sie unter Synchronisierungsmodi.

   Wenn Sie das verwenden AWS Glue Sie Data Catalog als Ihren Metastore verwenden, können Sie die folgenden Konfigurationseigenschaften für Ihren Hudi-Job angeben.

```
--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.

Apache Iceberg mit EMR Serverless verwenden

Um Apache Iceberg mit serverlosen Anwendungen zu verwenden EMR

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 AWS Glue Sie Data Catalog als Ihren Metastore oder konfigurieren Sie einen externen Metastore. Weitere Informationen zum Einrichten Ihres Metastores finden Sie unter. Metastore-Konfiguration

   Konfigurieren Sie die Metastore-Eigenschaften, die Sie für Iceberg verwenden möchten. Wenn Sie beispielsweise die verwenden möchten AWS Glue Data Catalog, legen Sie die folgenden Eigenschaften in der Anwendungskonfiguration fest.

   ```
   spark.sql.catalog.dev.warehouse=s3://DOC-EXAMPLE-BUCKET/EXAMPLE-PREFIX/
   spark.sql.extensions=org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions
   spark.sql.catalog.dev=org.apache.iceberg.spark.SparkCatalog
   spark.sql.catalog.dev.catalog-impl=org.apache.iceberg.aws.glue.GlueCatalog
   spark.hadoop.hive.metastore.client.factory.class=com.amazonaws.glue.catalog.metastore.AWSGl
   ```

   Wenn Sie den verwenden AWS Glue Sie Data Catalog als Ihren Metastore verwenden, können Sie die folgenden Konfigurationseigenschaften für Ihren Iceberg-Job angeben.

   ```
   --conf spark.jars=/usr/share/aws/iceberg/lib/iceberg-spark3-runtime.jar,
   --conf
    spark.sql.extensions=org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions,
   --conf spark.sql.catalog.dev=org.apache.iceberg.spark.SparkCatalog,
   ```

```
--conf spark.sql.catalog.dev.catalog-impl=org.apache.iceberg.aws.glue.GlueCatalog,
--conf spark.sql.catalog.dev.warehouse=s3://DOC-EXAMPLE-BUCKET/EXAMPLE-PREFIX/
--conf
 spark.hadoop.hive.metastore.client.factory.class=com.amazonaws.glue.catalog.metastore.AWSG
```

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, können Sie verschiedene Python-Bibliotheken als Abhängigkeiten paketieren. Dazu können Sie native Python-Funktionen verwenden, eine virtuelle Umgebung erstellen oder Ihre PySpark Jobs direkt für die Verwendung von Python-Bibliotheken konfigurieren. Diese Seite behandelt jeden Ansatz.

Verwendung nativer Python-Funktionen

Wenn Sie die folgende Konfiguration festlegen, können Sie sie verwenden, PySpark um Python-Dateien (`.py`), gezippte Python-Pakete (`.zip`) und Egg-Dateien (`.egg`) auf Spark-Executoren hochzuladen.

```
--conf spark.submit.pyFiles=s3://DOC-EXAMPLE-BUCKET/EXAMPLE-PREFIX/<.py|.egg|.zip file>
```

Weitere Informationen zur Verwendung virtueller Python-Umgebungen für PySpark Jobs finden Sie unter Verwenden PySpark systemeigener Funktionen.

Aufbau einer virtuellen Python-Umgebung

Um mehrere Python-Bibliotheken für einen PySpark Job zu paketieren, können Sie isolierte virtuelle Python-Umgebungen erstellen.

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 Serverless Samples-Repository. EMR GitHub

```
# initialize a python virtual environment
python3 -m venv pyspark_venvsource
source pyspark_venvsource/bin/activate

# optionally, ensure pip is up-to-date
pip3 install --upgrade pip

# install the python packages
pip3 install scipy
pip3 install matplotlib

# package the virtual environment into an archive
pip3 install venv-pack
venv-pack -f -o pyspark_venv.tar.gz

# copy the archive to an S3 location
aws s3 cp pyspark_venv.tar.gz s3://DOC-EXAMPLE-BUCKET/EXAMPLE-PREFIX/

# optionally, remove the virtual environment directory
rm -fr pyspark_venvsource
```

2. Reichen Sie den Spark-Job mit den für die Verwendung der virtuellen Python-Umgebung festgelegten Eigenschaften ein.

```
--conf spark.archives=s3://DOC-EXAMPLE-BUCKET/EXAMPLE-PREFIX/
pyspark_venv.tar.gz#environment
--conf spark.emr-serverless.driverEnv.PYSPARK_DRIVER_PYTHON=./environment/bin/
python
--conf spark.emr-serverless.driverEnv.PYSPARK_PYTHON=./environment/bin/python
--conf spark.executorEnv.PYSPARK_PYTHON=./environment/bin/python
```

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. Stellen bei Spark

PySpark Jobs für die Verwendung von Python-Bibliotheken konfigurieren

Mit EMR Amazon-Versionen 6.12.0 und höher können Sie EMR serverlose PySpark Jobs direkt so konfigurieren, dass sie beliebte Data-Science-Python-Bibliotheken wie Pandas verwenden NumPy, und das PyArrowohne zusätzliche Einrichtung.

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, die darauf aufbaut. NumPy 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 EMR Amazon-Version für Ihre Amazon EMR Serverless-Anwendung enthalten ist. Dazu müssen Sie eine virtuelle Python-Umgebung mit der Python-Version erstellen, die Sie verwenden möchten.

Um einen Job aus einer virtuellen Python-Umgebung einzureichen

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 EMR Amazon-Versionen 7.0.0 und höher verwenden, müssen Sie Ihre Befehle in einer Amazon Linux 2023-Umgebung ausführen, die der Umgebung ähnelt, die Sie für Ihre EMR serverlosen Anwendungen verwenden.
   > Wenn Sie Version 6.15.0 oder niedriger verwenden, müssen Sie die folgenden Befehle in einer ähnlichen Amazon Linux 2-Umgebung ausführen.

```
# 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://DOC-EXAMPLE-BUCKET/EXAMPLE-PREFIX/
pyspark_venv_python_3.9.9.tar.gz#environment
--conf spark.emr-serverless.driverEnv.PYSPARK_DRIVER_PYTHON=./environment/bin/
python
--conf spark.emr-serverless.driverEnv.PYSPARK_PYTHON=./environment/bin/python
--conf spark.executorEnv.PYSPARK_PYTHON=./environment/bin/python
```

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. Stellen bei Spark

Delta Lake OSS mit EMR Serverless verwenden

EMRAmazon-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 `delta-core.jar` in. `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 serverlosen Jobs angeben müssen.

1. Wenn Sie EMR serverlose Jobs einreichen, stellen Sie sicher, dass Sie über die folgenden Konfigurationseigenschaften verfügen und geben Sie die folgenden Parameter in das Feld ein. `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
```

2. Erstellen Sie eine lokale Tabelle`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://DOC-EXAMPLE-BUCKET/delta-lake/output/%s/" % str(uuid.uuid4())
    spark = SparkSession.builder.appName("DeltaSample").getOrCreate()

    ## creates a Delta table and outputs to target S3 bucket
    spark.range(5).write.format("delta").save(url)

    ## reads a Delta table and outputs to target S3 bucket
    spark.read.format("delta").load(url).show
```

3. Verwendung der AWS CLI, laden Sie die `delta_sample.py` Datei in Ihren Amazon S3 S3-Bucket hoch. Verwenden Sie dann den `start-job-run` Befehl, um einen Job an eine bestehende EMR serverlose Anwendung zu senden.

```
aws s3 cp delta_sample.py s3://DOC-EXAMPLE-BUCKET/code/

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

```
                "entryPoint": "s3://DOC-EXAMPLE-BUCKET/code/delta_sample.py",
                "sparkSubmitParameters": "--conf spark.jars=/usr/share/
aws/delta/lib/delta-core.jar,/usr/share/aws/delta/lib/delta-storage.jar --
conf spark.sql.extensions=io.delta.sql.DeltaSparkSessionExtension --conf
 spark.sql.catalog.spark_catalog=org.apache.spark.sql.delta.catalog.DeltaCatalog"
            }
        }'
```

Um Python-Bibliotheken mit Delta Lake zu verwenden, können Sie die `delta-core` Bibliothek hinzufügen, 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])
```

EMRAmazon-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 serverlosen 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 AWS-Konto.
3. Wenn Sie EMR serverlose Jobs in der Anwendungskonfiguration einreichen, schließen Sie die Delta JAR Lake-Dateien ein, die sich jetzt in Ihrem Bucket befinden.

   ```
   --conf spark.jars=s3://DOC-EXAMPLE-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://DOC-EXAMPLE-BUCKET/delta-lake/output/1.0.1/%s/" % str(uuid.uuid4())

## creates a Delta table and outputs to target S3 bucket
session.range(5).write.format("delta").save(url)

## reads a Delta table and outputs to target S3 bucket
session.read.format("delta").load(url).show
```

EMRServerlose Jobs von Airflow aus einreichen

Der Amazon-Anbieter in Apache Airflow bietet EMR serverlose Betreiber. Weitere Informationen zu Operatoren finden Sie unter Amazon EMR Serverless Operators in der Apache 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 so, dass sie die neue `requirements.txt` Datei verwendet.

```
apache-airflow-providers-amazon==6.0.0
boto3>=1.23.9
```

Beachten Sie, EMR dass Serverless-Unterstützung 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 aktiviertem Airflow 2.4.3 verwenden. MWAA

Das folgende abgekürzte Beispiel zeigt, wie Sie eine Anwendung erstellen, mehrere Spark-Jobs ausführen und die Anwendung dann beenden. Ein vollständiges Beispiel ist im EMRServerless

GitHub Samples-Repository verfügbar. Weitere Einzelheiten zur `sparkSubmit` Konfiguration finden Sie unterStellen bei Spark.

```
from datetime import datetime

from airflow import DAG
from airflow.providers.amazon.aws.operators.emr import (
    EmrServerlessCreateApplicationOperator,
    EmrServerlessStartJobOperator,
    EmrServerlessDeleteApplicationOperator,
)

# Replace these with your correct values
JOB_ROLE_ARN = "arn:aws:iam::account-id:role/emr_serverless_default_role"
S3_LOGS_BUCKET = "DOC-EXAMPLE-BUCKET"

DEFAULT_MONITORING_CONFIG = {
    "monitoringConfiguration": {
        "s3MonitoringConfiguration": {"logUri": f"s3://DOC-EXAMPLE-BUCKET/logs/"}
    },
}

with DAG(
    dag_id="example_endtoend_emr_serverless_job",
    schedule_interval=None,
    start_date=datetime(2021, 1, 1),
    tags=["example"],
    catchup=False,
) as dag:
    create_app = EmrServerlessCreateApplicationOperator(
        task_id="create_spark_app",
        job_type="SPARK",
        release_label="emr-6.7.0",
        config={"name": "airflow-test"},
    )

    application_id = create_app.output

    job1 = EmrServerlessStartJobOperator(
        task_id="start_job_1",
        application_id=application_id,
        execution_role_arn=JOB_ROLE_ARN,
        job_driver={
```

```
            "sparkSubmit": {
                "entryPoint": "local:///usr/lib/spark/examples/src/main/python/
pi_fail.py",
            }
        },
        configuration_overrides=DEFAULT_MONITORING_CONFIG,
    )

    job2 = EmrServerlessStartJobOperator(
        task_id="start_job_2",
        application_id=application_id,
        execution_role_arn=JOB_ROLE_ARN,
        job_driver={
            "sparkSubmit": {
                "entryPoint": "local:///usr/lib/spark/examples/src/main/python/pi.py",
                "entryPointArguments": ["1000"]
            }
        },
        configuration_overrides=DEFAULT_MONITORING_CONFIG,
    )

    delete_app = EmrServerlessDeleteApplicationOperator(
        task_id="delete_app",
        application_id=application_id,
        trigger_rule="all_done",
    )

    (create_app >> [job1, job2] >> delete_app)
```

Verwenden benutzerdefinierter Hive-Funktionen mit Serverless EMR

Mit den benutzerdefinierten Hive-Funktionen (UDFs) können Sie benutzerdefinierte Funktionen zur Verarbeitung von Datensätzen oder Datensatzgruppen erstellen. In diesem Tutorial verwenden Sie ein 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

Um eine UDF mit EMR Serverless zu verwenden

1. Navigieren Sie zu, um ein GitHubBeispiel UDF zu finden. 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 `mvn package -DskipTests` den Befehl aus, um die JAR Datei zu erstellen, die Ihr Beispiel enthältUDFs.
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://DOC-EXAMPLE-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://DOC-EXAMPLE-BUCKET/jars/brickhouse-0.8.2-JS.jar;
CREATE TEMPORARY FUNCTION from_json AS 'brickhouse.udf.json.FromJsonUDF';
select from_json('{"key1":[0,1,2], "key2":[3,4,5,6], "key3":[7,8,9]}', map("",
 array(cast(0 as int))));
select from_json('{"key1":[0,1,2], "key2":[3,4,5,6], "key3":[7,8,9]}', map("",
 array(cast(0 as int))))["key1"][2];
```

4. 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://DOC-EXAMPLE-BUCKET/queries/udf_example.q",
        "parameters": "--hiveconf hive.exec.scratchdir=s3://DOC-EXAMPLE-BUCKET/emr-
serverless-hive/scratch --hiveconf hive.metastore.warehouse.dir=s3://'$BUCKET'/emr-
serverless-hive/warehouse"
    }
}' --configuration-overrides '{
    "applicationConfiguration": [{
        "classification": "hive-site",
        "properties": {
            "hive.driver.cores": "2",
            "hive.driver.memory": "6G"
        }
```

```
    }],
    "monitoringConfiguration": {
        "s3MonitoringConfiguration": {
            "logUri": "s3://DOC-EXAMPLE-BUCKET/logs/"
        }
    }
}'
```

5. 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://DOC-EXAMPLE-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
- Verwenden Sie eine benutzerdefinierte Java-Version
- Erstellen Sie ein Data-Science-Image
- Verarbeitung von Geodaten mit Apache Sedona

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 will run 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 sudo amazon-linux-extras install java-openjdk11

# EMRS will run 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, datenwissenschaftliche 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 will run 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 enthält.

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

USER root

RUN yum install -y wget
RUN wget https://repo1.maven.org/maven2/org/apache/sedona/sedona-core-3.0_2.12/1.3.0-
incubating/sedona-core-3.0_2.12-1.3.0-incubating.jar -P /usr/lib/spark/jars/
RUN pip3 install apache-sedona

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

Verwenden der Amazon Redshift Redshift-Integration für Apache Spark auf Amazon Serverless EMR

Mit EMR Amazon-Version 6.9.0 und höher enthält jedes Release-Image einen Connector zwischen Apache Spark und Amazon Redshift. Mit diesem Connector können Sie Spark auf Amazon EMR Serverless verwenden, 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

Um die Integration mit EMR Serverless 6.9.0 zu verwenden, müssen Sie die erforderlichen Spark-Redshift-Abhängigkeiten mit Ihrem Spark-Job übergeben. Wird verwendet`--jars`, um Redshift-Connector-bezogene Bibliotheken einzubeziehen. Weitere von der `--jars`-Option unterstützte Dateispeicherorte finden Sie im Abschnitt Erweitertes Abhängigkeitsmanagement der Apache-Spark-Dokumentation.

- `spark-redshift.jar`
- `spark-avro.jar`
- `RedshiftJDBC.jar`
- `minimal-json.jar`

EMRAmazon-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-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 Dateien sind. JAR

```
--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

Verwenden Sie 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 aufrufen lassen, `GetSecretValue` API 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
```

Authentifizieren Sie sich mit einem Treiber bei Amazon Redshift JDBC

Geben Sie den Benutzernamen und das Passwort in der JDBC URL

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 angeben. JDBC URL

> **Note**
>
> Wenn Sie die Datenbankanmeldedaten in übergeben, URL kann jederURL, der Zugriff auf die 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, können Sie das folgende Format verwenden, um den Benutzernamen und das Passwort in der Datei festzulegen JDBCURL:

```
jdbc:redshift://redshifthost:5439/database?user=username&password=password
```

Verwenden Sie die IAM basierte Authentifizierung mit der Amazon EMR Serverless-Jobausführungsrolle

Ab Amazon EMR Serverless Version 6.9.0 ist der Amazon Redshift JDBC Redshift-Treiber 2.1 oder höher in der Umgebung enthalten. Bei JDBC Treiber 2.1 und höher können Sie den unformatierten Benutzernamen JDBC URL und das Passwort angeben und nicht angeben.

Stattdessen können Sie ein `jdbc:redshift:iam://`-Schema angeben. Dadurch wird der JDBC Treiber angewiesen, Ihre EMR Serverless-Job-Ausführungsrolle zu verwenden, um die Anmeldeinformationen automatisch abzurufen. Weitere Informationen finden Sie unter Konfiguration einer JDBC ODBC OR-Verbindung für die Verwendung von IAM Anmeldeinformationen im Amazon Redshift Management Guide. Ein Beispiel dafür 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
`redshift:DescribeCluster`	Erforderlich, wenn Sie den Amazon Redshift Redshift-Cluster angeben und AWS-Region im JDBC URL statt im Endpunkt
`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

Herstellen einer Verbindung zu Amazon Redshift innerhalb eines anderen VPC

Wenn Sie einen bereitgestellten Amazon Redshift-Cluster oder eine Amazon Redshift Serverless-Arbeitsgruppe unter einer einrichten, müssen Sie die VPC Konnektivität für Ihre Amazon EMR Serverless-Anwendung konfigurierenVPC, um auf die Ressourcen zugreifen zu können. Weitere Informationen zur Konfiguration der VPC Konnektivität in einer serverlosen Anwendung finden Sie unter. EMR Zugriff konfigurieren VPC

- Wenn Ihr bereitgestellter Amazon Redshift-Cluster oder Ihre Amazon Redshift Serverless-Arbeitsgruppe öffentlich zugänglich ist, können Sie bei der Erstellung serverloser Anwendungen ein oder mehrere private Subnetze angeben, an die ein NAT Gateway angeschlossen ist. EMR
- Wenn Ihr bereitgestellter Amazon Redshift-Cluster oder Ihre Amazon Redshift Serverless-Arbeitsgruppe nicht öffentlich zugänglich ist, müssen Sie einen von Amazon Redshift verwalteten VPC Endpoint für Ihren Amazon Redshift Redshift-Cluster erstellen, wie unter beschrieben. Zugriff konfigurieren VPC 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 serverlosen Anwendung angeben. EMR

> **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 auch einen VPC Endpunkt in diesen Subnetzen für Amazon Redshift oder Amazon Redshift Serverless erstellen. Auf diese Weise kann der Treiber die Anmeldeinformationen abrufenJDBC.

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 Datenquelle API und mit SparkSQL.

Data source API

Wird verwendet PySpark , um Beispieldaten aus und in eine Amazon Redshift Redshift-Datenbank mit Datenquelle 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 Spark SQL 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 die JDBC Verbindung von Spark auf Amazon EMR zu Amazon Redshift aktivieren. SSL
- Wir empfehlen, dass Sie die Anmeldeinformationen für den Amazon Redshift Redshift-Cluster in verwalten AWS Secrets Manager als bewährte Methode. Siehe Verwenden AWS Secrets Manager um beispielsweise Anmeldeinformationen für die Verbindung mit Amazon Redshift abzurufen.
- 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.
- Das `tempdir` URI zeigt auf einen Amazon S3 S3-Standort. Dieses temporäre Verzeichnis wird nicht automatisch bereinigt und kann zusätzliche Kosten verursachen.
- Beachten Sie die folgenden Empfehlungen für Amazon Redshift:
  - Wir empfehlen, den öffentlichen Zugriff auf den Amazon-Redshift-Cluster zu blockieren.
  - Wir empfehlen, die Amazon-Redshift-Auditprotokollierung zu aktivieren.
  - Wir empfehlen Ihnen die Amazon-Redshift-Verschlüsselung im Ruhezustand zu aktivieren.
- Beachten Sie die folgenden Empfehlungen für Amazon S3:
  - Wir empfehlen Ihnen den öffentlichen Zugriff auf Amazon-S3-Buckets zu blockieren.
  - Wir empfehlen die Verwendung der serverseitigen Amazon-S3-Verschlüsselung, um die verwendeten Amazon-S3-Buckets zu verschlüsseln.
  - Wir empfehlen, die Lebenszyklusrichtlinien für Amazon S3 zu verwenden, um die Aufbewahrungsregeln für den Amazon-S3-Bucket zu definieren.

- Amazon überprüft EMR immer Code, der aus Open Source in das Bild importiert wurde. Aus Sicherheitsgründen unterstützen wir die folgenden Authentifizierungsmethoden von Spark für Amazon S3 nicht:
  - Einstellung AWS Zugriffsschlüssel in der Konfigurationsklassifizierung `hadoop-env`
  - Codierung 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 Serverless eine Verbindung zu DynamoDB herstellen EMR

In diesem Tutorial laden Sie eine Teilmenge der Daten vom United States Board on Geographic Names in einen Amazon S3 S3-Bucket hoch und kopieren die Daten dann mit Hive oder Spark auf Amazon EMR Serverless in eine Amazon DynamoDB-Tabelle, die Sie abfragen können.

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 *DOC-EXAMPLE-BUCKET* durch den Namen Ihres neu erstellten Buckets. Jetzt ist Ihre EMR serverlose 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 sehen Sie sich die ersten Zeilen der Datei an:

   ```
   unzip features.zip
   head features.txt
   ```

Das Ergebnis sollte in etwa wie folgt aussehen.

```
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://DOC-EXAMPLE-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://DOC-EXAMPLE_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 SQL Spark-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://DOC-EXAMPLE-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://DOC-EXAMPLE-BUCKET/ddb-connector/")
            .map(row => {
                val line = row.split("\\|")
                val item = new DynamoDBItemWritable()

                val elevInFt = if (line.length > 6) {
                    new AttributeValue().withN(line(6))
                } else {
                    new AttributeValue().withNULL(true)
                }

                item.setItem(Map(
                    "feature_id" -> new AttributeValue().withN(line(0)),
```

```
                    "feature_name" -> new AttributeValue(line(1)),
                    "feature_class" -> new AttributeValue(line(2)),
                    "state_alpha" -> new AttributeValue(line(3)),
                    "prim_lat_dec" -> new AttributeValue().withN(line(4)),
                    "prim_long_dec" -> new AttributeValue().withN(line(5)),
                    "elev_in_ft" -> elevInFt)
                    .asJava)
                (new Text(""), item)
        })
        rdd.saveAsHadoopDataset(jobConf)
    }
}
```

Hive

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

Um Daten aus der DynamoDB-Tabelle abzufragen, die Sie im vorherigen Schritt erstellt haben, können Sie entweder Spark SQL oder Spark verwenden. MapReduce API

Example — Fragen Sie Ihre DynamoDB-Tabelle mit Spark ab SQL

Die folgende SQL Spark-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 SQL Spark-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 SQL Spark-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 dem Spark ab MapReduce API

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: Eine 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 AWS Management Console 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 IAMRollen erstellen im Benutzerhandbuch.

   c. Erstellen Sie eine IAM Richtlinie, die Berechtigungen für den Zugriff auf die kontoübergreifende DynamoDB-Tabelle gewährt. Hängen Sie dann die IAM Richtlinie an an. `Cross-Account-Role-B`

      Die folgende Richtlinie gewährt Zugriff auf eine DynamoDB-Tabelle. `CrossAccountTable`

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

```
    "Statement": [
        {"Effect": "Allow",
            "Action": "dynamodb:*",
            "Resource": "arn:aws:dynamodb:region:AccountB:table/
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: Account-Cross-Role-B erstellt haben.

   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.

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

e. Geben `Job-Execution-Role-A` Sie ein und `AccountA` erhalten `- STS Assume role` Sie die erforderlichen Rechte zur Übernahme`Cross-Account-Role-B`.

   In der IAM Konsole für AWS-Konto `AccountA`, wählen Sie`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.

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

```
    ]
}
```

f. Legen Sie für die `dynamodb.customAWSCredentialsProvider` Eigenschaft einen Wert wie `com.amazonaws.emr.AssumeRoleAWSCredentialsProvider` bei der Klassifizierung der Kernstandorte fest. Stellen Sie die Umgebungsvariable auf `ASSUME_ROLE_CREDENTIALS_ROLE_ARN` den ARN Wert von ein. `Cross-Account-Role-B`

3. Führen Sie den Spark- oder Hive-Job aus mit`Job-Execution-Role-A`.

Überlegungen

Überlegungen zur Verwendung des DynamoDB-Connectors mit Apache Spark

- Spark unterstützt die Erstellung einer Hive-Tabelle mit der Storage-Handler-Option SQL nicht. Weitere Informationen finden Sie unter Speicherformat für Hive-Tabellen angeben in der Apache Spark-Dokumentation.
- Spark unterstützt den `STORED BY` Vorgang mit dem Storage-Handler SQL 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 verwenden, und nicht mit dem. SQL MapReduce API

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 für die DynamoDB-Tabelle konfigurierte Gesamtlesedurchsatz 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 pro EMR serverloser Anwendung hinausgeht. 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 als der gesamte Schreibdurchsatz, der für eine DynamoDB-Tabelle konfiguriert ist, geteilt durch den Wert der Eigenschaft berechnet. `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, können Sie den Wert der `mapreduce.job.maps` Eigenschaft auf der Grundlage der Anzahl der Aufgaben in der letzten Phase oder des Schreibdurchsatzes festlegen, den Sie pro Aufgabe zuweisen möchten. Legen Sie diese Eigenschaft in der `mapred-site` Klassifizierung des start-job-run Befehls auf Serverless fest. EMR

Sicherheit

Cloud-Sicherheit bei AWS hat höchste Priorität. Als AWS Als Kunde profitieren Sie von einer Rechenzentrums- und Netzwerkarchitektur, die darauf ausgelegt ist, die Anforderungen der sicherheitssensibelsten Unternehmen zu erfüllen.

Sicherheit ist eine gemeinsame Verantwortung zwischen AWS und du. 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 läuft AWS Dienstleistungen in der AWS Wolke. AWS bietet Ihnen auch Dienste, die Sie sicher nutzen können. Externe Prüfer testen und verifizieren regelmäßig die Wirksamkeit unserer Sicherheit im Rahmen der AWS Compliance-Programme. Weitere Informationen zu den Compliance-Programmen, die für Amazon EMR Serverless gelten, finden Sie unter AWS Dienstleistungen im Umfang des Compliance-Programms.
- Sicherheit in der Cloud — Ihre Verantwortung richtet sich nach AWS Dienst, 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. Die Themen in diesem Kapitel zeigen Ihnen, wie Sie Amazon EMR Serverless konfigurieren und andere verwenden AWS Services zur Erreichung Ihrer Sicherheits- und Compliance-Ziele.

Themen

- Resilienz bei 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

EMRServerless bietet eine detaillierte Zugriffsrichtlinie für Anwendungen, die IAM Rollen verwenden, z. B. Ausführungsrollen. Wir empfehlen, Ausführungsrollen nur die für den Auftrag 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

EMRServerless sorgt für eine vollständige Netzwerkisolierung zwischen Aufträgen, die zu verschiedenen EMR serverlosen Anwendungen gehören. In Fällen, in denen eine Isolierung auf Jobebene gewünscht wird, sollten Sie erwägen, Jobs in verschiedenen serverlosen Anwendungen zu isolieren. EMR

Rollenbasierte Zugriffssteuerungsberechtigungen () RBAC

Administratoren sollten die Berechtigungen für die rollenbasierte Zugriffskontrolle (RBAC) für serverlose Anwendungen strikt kontrollieren. EMR

Datenschutz

Das Tool AWS Das Modell der gemeinsamen Verantwortung gilt für den Datenschutz in Amazon EMR Serverless. Wie in diesem Modell beschrieben, AWS ist verantwortlich für den Schutz

der globalen Infrastruktur, auf der alle AWS Wolke. Sie sind dafür verantwortlich, die Kontrolle über Ihre in dieser Infrastruktur gehosteten Inhalte zu behalten. Dieser Inhalt umfasst die Sicherheitskonfigurations- und Verwaltungsaufgaben für AWS Dienste, die Sie verwenden. Weitere Informationen zum Datenschutz finden Sie in der Datenschutzerklärung FAQ. Informationen zum Datenschutz in Europa finden Sie auf der AWS Modell der geteilten Verantwortung und GDPR Blogbeitrag auf der AWS Blog zum Thema Sicherheit.

Aus Datenschutzgründen empfehlen wir Ihnen, Folgendes zu schützen AWS Kontoinformationen und richten Sie individuelle Konten ein mit AWS Identity and Access Management (IAM). 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).
- Verwenden SieSSL/TLS, um mit zu kommunizieren AWS Ressourcen schätzen. Wir empfehlen TLS 1.2 oder höher.
- Einrichtung API und Protokollierung von Benutzeraktivitäten mit AWS CloudTrail.
- Verwenden Sie AWS Verschlüsselungslösungen, zusammen mit allen darin enthaltenen Standardsicherheitskontrollen AWS Dienstleistungen.
- 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 Amazon EMR Serverless-Verschlüsselungsoptionen, um Daten im Ruhezustand und bei der Übertragung zu verschlüsseln.
- Wenn Sie für den Zugriff FIPS 140-2 validierte kryptografische Module benötigen AWS über eine Befehlszeilenschnittstelle oder einenAPI, verwenden Sie einen Endpunkt. FIPS Weitere Informationen zu den verfügbaren FIPS Endpunkten finden Sie unter Federal Information Processing Standard (FIPS) 140-2.

Wir empfehlen dringend, in Freitextfeldern wie z. B. im Feld Name keine sensiblen, identifizierenden Informationen wie Kontonummern von Kunden einzugeben. Dies gilt auch, wenn Sie mit Amazon EMR Serverless oder anderen Geräten arbeiten AWS Dienste, die die Konsole verwenden, API AWS CLI, oder AWS SDKs. Alle Daten, die Sie in Amazon EMR Serverless oder andere Dienste eingeben, werden möglicherweise zur Aufnahme in Diagnoseprotokolle aufgenommen. Wenn Sie einem externen Server eine URL zur Verfügung stellen, geben Sie keine Anmeldeinformationen an, URL um Ihre Anfrage an diesen Server zu validieren.

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 mehreren Optionen wählen, einschließlich Schlüsseln, die von verwaltet werden AWS Key Management Service, Schlüssel, die von Amazon S3 verwaltet werden, sowie Schlüssel und Zertifikate von benutzerdefinierten Anbietern, die Sie bereitstellen. Bei der Verwendung AWS KMS als Ihr Schlüsselanbieter fallen Gebühren für die Speicherung und Verwendung von Verschlüsselungsschlüsseln an. Weitere Informationen finden Sie unter AWS KMS Preisgestaltung.

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 für EMRFS Daten in Amazon S3

Jede EMR serverlose Anwendung verwendet eine bestimmte Release-Version, zu der auch EMRFS (EMRDateisystem) gehört. Die Amazon S3-Verschlüsselung funktioniert mit EMR Dateisystem (EMRFS) -Objekten, die aus Amazon S3 gelesen und in Amazon S3 geschrieben wurden. Sie können die serverseitige Amazon S3 S3-Verschlüsselung (SSE) oder die clientseitige Verschlüsselung (CSE) als Standardverschlüsselungsmodus angeben, wenn Sie die Verschlüsselung im Ruhezustand aktivieren. Optional können Sie verschiedene Verschlüsselungsmethoden für einzelne Buckets mithilfe von Per bucket encryption overrides (Bucket-weises Überschreiben der Verschlüsselung) angeben. Unabhängig davon, ob die Amazon S3-Verschlüsselung aktiviert ist, verschlüsselt Transport Layer Security (TLS) die EMRFS Objekte, die zwischen EMR Clusterknoten und Amazon S3 übertragen werden. Wenn Sie Amazon S3 CSE mit vom Kunden verwalteten Schlüsseln verwenden, muss Ihre Ausführungsrolle, mit der Jobs in einer EMR serverlosen 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**
>
> Wenn Sie verwenden AWS KMS, für die Speicherung und Verwendung von Verschlüsselungsschlüsseln fallen Gebühren an. Weitere Informationen finden Sie unter AWS KMS Preisgestaltung.

Serverseitige Verschlüsselung im Amazon S3

Wenn Sie die Amazon-S3-Verschlüsselung einrichten, verschlüsselt Amazon S3 die Daten auf der Objektebene, während die Daten auf den Datenträger geschrieben werden, und entschlüsselt sie, wenn auf sie zugegriffen wird. Weitere Informationen SSE dazu 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 - Amazon S3 verwaltet Schlüssel für Sie. Bei EMR Serverless ist keine zusätzliche Einrichtung erforderlich.
- SSE- KMS - Sie verwenden eine AWS KMS key zur Einrichtung mit Richtlinien, die für EMR Serverless geeignet sind. Bei EMR Serverless ist keine zusätzliche Einrichtung erforderlich.

Zur Verwendung AWS KMS Verschlüsselung für Daten, die Sie in Amazon S3 schreiben, haben Sie zwei Möglichkeiten, wenn Sie die verwenden `StartJobRun`API. Sie können entweder die Verschlüsselung für alles aktivieren, was Sie in Amazon S3 schreiben, oder Sie können die Verschlüsselung für Daten aktivieren, die Sie in einen bestimmten Bucket schreiben. Weitere Informationen zu finden Sie in der `StartJobRun` API Serverless Reference. EMR API

Zum Einschalten AWS KMS Verschlüsselung für alle Daten, die Sie in Amazon S3 schreiben, verwenden Sie die folgenden Befehle, wenn Sie die aufrufen `StartJobRun`API.

```
--conf spark.hadoop.fs.s3.enableServerSideEncryption=true
--conf spark.hadoop.fs.s3.serverSideEncryption.kms.keyId=<kms_id>
```

Zum Einschalten AWS KMS Verwenden Sie zur Verschlüsselung von Daten, die Sie in einen bestimmten Bucket schreiben, die folgenden Befehle, wenn Sie den aufrufen `StartJobRun`API.

```
--conf spark.hadoop.fs.s3.bucket.<DOC-EXAMPLE-BUCKET>.enableServerSideEncryption=true
```

```
--conf spark.hadoop.fs.s3.bucket.<DOC-EXAMPLE-
BUCKET>.serverSideEncryption.kms.keyId=<kms-id>
```

SSEmit 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-Version verfügbar ist. EMR 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 Schlüssel verwenden, die bereitgestellt werden von AWS KMS (CSE-KMS) oder eine benutzerdefinierte Java-Klasse, die den clientseitigen Stammschlüssel (CSE-C) bereitstellt. Die Verschlüsselungsspezifikationen zwischen CSE - KMS und CSE -C unterscheiden sich geringfügig, abhängig vom angegebenen Anbieter und den Metadaten des Objekts, das entschlüsselt oder verschlüsselt wird. Wenn Sie Amazon S3 CSE mit vom Kunden verwalteten Schlüsseln verwenden, muss Ihre Ausführungsrolle, mit der Jobs in einer EMR serverlosen 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 nach Industriestandard AES -256 verwendet wird.

Schlüsselverwaltung

Sie können so konfigurieren, dass Ihre Schlüssel automatisch KMS rotiert werden. KMS 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 Rotieren von Kundenhauptschlüsseln.

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. RPCDie Kommunikation zwischen Treibern und Executoren ist verschlüsselt.
- Hive
  - Kommunikation zwischen AWS Glue Metastore und EMR serverlose Anwendungen erfolgen über. TLS

Sie sollten nur verschlüsselte Verbindungen über HTTPS (TLS) zulassen, indem Sie die SecureTransport Bedingung aws: in den Amazon S3 IAM S3-Bucket-Richtlinien verwenden.

Identity and Access Management (IAM) in Amazon EMR Serverless

AWS Identity and Access Management (IAM) ist ein AWS-Service das hilft einem Administrator, den Zugriff auf sicher zu kontrollieren AWS Ressourcen schätzen. IAMAdministratoren kontrollieren, wer authentifiziert (angemeldet) und autorisiert werden kann (über Berechtigungen verfügt), um Amazon EMR Serverless-Ressourcen zu verwenden. IAMist ein AWS-Service das Sie ohne zusätzliche Kosten nutzen können.

Themen

Zielgruppe

Wie benutzt du AWS Identity and Access Management (IAM) unterscheidet sich je nach der Arbeit, die Sie in Amazon EMR Serverless ausführen.

Servicebenutzer — Wenn Sie den Amazon EMR Serverless-Service für Ihre Arbeit verwenden, stellt Ihnen Ihr Administrator die Anmeldeinformationen und Berechtigungen zur Verfügung, die Sie benötigen. Da Sie für Ihre Arbeit mehr EMR Amazon Serverless-Funktionen verwenden, benötigen Sie möglicherweise zusätzliche Berechtigungen. Wenn Sie die Fuktionsweise der Zugriffskontrolle nachvollziehen, wissen Sie bereits, welche Berechtigungen Sie von Ihrem Administrator anzufordern müssen. Wenn Sie auf eine Funktion in Amazon EMR Serverless nicht zugreifen können, finden Sie weitere Informationen unterFehlerbehebung bei Amazon EMR Serverless Identity and Access.

Service Administrator — Wenn Sie in Ihrem Unternehmen für Amazon EMR Serverless-Ressourcen verantwortlich sind, haben Sie wahrscheinlich vollen Zugriff auf Amazon EMR Serverless. Es ist Ihre Aufgabe, zu bestimmen, auf welche Funktionen und Ressourcen von Amazon EMR Serverless Ihre Servicebenutzer zugreifen sollen. Anschließend müssen Sie Anfragen an Ihren IAM Administrator senden, um die Berechtigungen Ihrer Servicebenutzer zu ändern. Lesen Sie die Informationen auf dieser Seite, um die grundlegenden Konzepte von zu verstehenIAM. Weitere Informationen darüber, wie Ihr Unternehmen Amazon EMR Serverless nutzen IAM kann, finden Sie unterIdentity and Access Management (IAM) in Amazon EMR Serverless.

IAMAdministrator — Wenn Sie ein IAM Administrator sind, möchten Sie vielleicht mehr darüber erfahren, wie Sie Richtlinien schreiben können, um den Zugriff auf Amazon EMR Serverless zu verwalten. Beispiele für identitätsbasierte Amazon EMR Serverless-Richtlinien, die Sie in IAM verwenden können, finden Sie unter. Beispiele für identitätsbasierte Richtlinien für Serverless EMR

Authentifizierung mit Identitäten

Authentifizierung ist die Art und Weise, wie Sie sich anmelden AWS mit Ihren Identitätsdaten. Sie müssen authentifiziert (angemeldet) sein AWS) als Root-Benutzer des AWS-Kontos, als IAM Benutzer oder indem Sie eine IAM Rolle übernehmen.

Sie können sich anmelden bei AWS als föderierte Identität mithilfe von Anmeldeinformationen, die über eine Identitätsquelle bereitgestellt wurden. AWS IAM Identity Center (IAMIdentity Center) - Nutzer, die Single-Sign-On-Authentifizierung Ihres Unternehmens und Ihre Google- oder Facebook-Anmeldeinformationen sind Beispiele für föderierte Identitäten. Wenn Sie sich als föderierte Identität anmelden, hat Ihr Administrator zuvor einen Identitätsverbund mithilfe von Rollen eingerichtet. IAM

Wenn Sie darauf zugreifen AWS Wenn Sie den Verbund verwenden, übernehmen Sie indirekt eine Rolle.

Je nachdem, welcher Benutzertyp Sie sind, können Sie sich bei der anmelden AWS Management Console oder das AWS Zugangsportal. Weitere Informationen zur Anmeldung bei AWS, siehe So melden Sie sich bei AWS-Konto in der AWS-Anmeldung Benutzerleitfaden.

Wenn Sie darauf zugreifen AWS programmatisch AWS stellt ein Software Development Kit (SDK) und eine Befehlszeilenschnittstelle (CLI) bereit, um Ihre Anfragen mithilfe Ihrer Anmeldeinformationen kryptografisch zu signieren. Wenn Sie es nicht verwenden AWS Tools, Sie müssen Anfragen selbst unterschreiben. Weitere Informationen zur Verwendung der empfohlenen Methode, um Anfragen selbst zu signieren, finden Sie unter Signieren AWS APIAnfragen im IAMBenutzerhandbuch.

Unabhängig von der verwendeten Authentifizierungsmethode müssen Sie möglicherweise zusätzliche Sicherheitsinformationen angeben. Zum Beispiel AWS empfiehlt, die Multi-Faktor-Authentifizierung (MFA) zu verwenden, um die Sicherheit Ihres Kontos zu erhöhen. Weitere Informationen finden Sie unter Multi-Faktor-Authentifizierung im AWS IAM Identity Center Benutzerhandbuch und Verwendung der Multi-Faktor-Authentifizierung () MFA in AWS im IAM-Benutzerhandbuch.

AWS-Konto Root-Benutzer

Wenn Sie eine erstellen AWS-Konto, beginnen Sie mit einer Anmeldeidentität, die vollständigen Zugriff auf alle hat AWS-Services und Ressourcen im Konto. Diese Identität wird als AWS-Konto Root-Benutzer. Der Zugriff erfolgt, indem Sie sich mit der E-Mail-Adresse und dem Passwort anmelden, mit denen Sie das Konto erstellt haben. Wir raten ausdrücklich davon ab, den Root-Benutzer für Alltagsaufgaben zu verwenden. Schützen Sie Ihre Root-Benutzer-Anmeldeinformationen und verwenden Sie diese, um die Aufgaben auszuführen, die nur der Root-Benutzer ausführen kann. Eine vollständige Liste der Aufgaben, für die Sie sich als Root-Benutzer anmelden müssen, finden Sie im Benutzerhandbuch unter Aufgaben, für die Root-Benutzeranmeldedaten erforderlich sind. IAM

Verbundidentität

Es hat sich bewährt, menschlichen Benutzern, einschließlich Benutzern, die Administratorzugriff benötigen, vorzuschreiben, für den Zugriff den Verbund mit einem Identitätsanbieter zu verwenden AWS-Services mithilfe temporärer Anmeldeinformationen.

Eine föderierte Identität ist ein Benutzer aus Ihrem Unternehmensbenutzerverzeichnis, ein Web-Identitätsanbieter, der AWS Directory Service, das Identity Center-Verzeichnis oder ein

beliebiger Benutzer, der zugreift AWS-Services mithilfe von Anmeldeinformationen, die über eine Identitätsquelle bereitgestellt wurden. Wenn föderierte Identitäten darauf zugreifen AWS-Konten, übernehmen sie Rollen, und die Rollen stellen temporäre Anmeldeinformationen bereit.

Für eine zentralisierte Zugriffsverwaltung empfehlen wir die Verwendung von AWS IAM Identity Center. Sie können Benutzer und Gruppen in IAM Identity Center erstellen, oder Sie können eine Verbindung zu einer Gruppe von Benutzern und Gruppen in Ihrer eigenen Identitätsquelle herstellen und diese synchronisieren, um sie in all Ihren AWS-Konten und Anwendungen. Informationen zu IAM Identity Center finden Sie unter Was ist IAM Identity Center? in der AWS IAM Identity Center Benutzerleitfaden.

IAM-Benutzer und -Gruppen

Ein IAMBenutzer ist eine Identität innerhalb Ihres AWS-Konto das über spezifische Berechtigungen für eine einzelne Person oder Anwendung verfügt. Wir empfehlen, sich nach Möglichkeit auf temporäre Anmeldeinformationen zu verlassen, anstatt IAM Benutzer mit langfristigen Anmeldeinformationen wie Passwörtern und Zugriffsschlüsseln zu erstellen. Wenn Sie jedoch spezielle Anwendungsfälle haben, für die langfristige Anmeldeinformationen von IAM Benutzern erforderlich sind, empfehlen wir, die Zugriffsschlüssel abwechselnd zu verwenden. Weitere Informationen finden Sie im Benutzerhandbuch unter Regelmäßiges Rotieren von Zugriffsschlüsseln für Anwendungsfälle, für die IAM langfristige Anmeldeinformationen erforderlich sind.

Eine IAMGruppe ist eine Identität, die eine Sammlung von IAM Benutzern angibt. Sie können sich nicht als Gruppe anmelden. Mithilfe von Gruppen können Sie Berechtigungen für mehrere Benutzer gleichzeitig angeben. Gruppen vereinfachen die Verwaltung von Berechtigungen, wenn es zahlreiche Benutzer gibt. Sie könnten beispielsweise eine Gruppe benennen IAMAdminsund dieser Gruppe Berechtigungen zur Verwaltung von IAM Ressourcen erteilen.

Benutzer unterscheiden sich von Rollen. Ein Benutzer ist einer einzigen Person oder Anwendung eindeutig zugeordnet. Eine Rolle kann von allen Personen angenommen werden, die sie benötigen. Benutzer besitzen dauerhafte Anmeldeinformationen. Rollen stellen temporäre Anmeldeinformationen bereit. Weitere Informationen finden Sie unter Wann sollte ein IAM Benutzer (statt einer Rolle) erstellt werden? im IAMBenutzerhandbuch.

IAMRollen

Eine IAMRolle ist eine Identität in deinem AWS-Konto das hat spezifische Berechtigungen. Es ähnelt einem IAM Benutzer, ist jedoch keiner bestimmten Person zugeordnet. Sie können vorübergehend eine IAM Rolle in der übernehmen AWS Management Console indem Sie die Rollen wechseln.

Sie können eine Rolle übernehmen, indem Sie einen anrufen AWS CLI or AWS APIOperation oder mithilfe eines benutzerdefiniertenURL. Weitere Informationen zu Methoden zur Verwendung von Rollen finden Sie unter Verwenden von IAM Rollen im IAMBenutzerhandbuch.

IAMRollen mit temporären Anmeldeinformationen sind in den folgenden Situationen nützlich:

- Verbundbenutzerzugriff – Um einer Verbundidentität Berechtigungen zuzuweisen, erstellen Sie eine Rolle und definieren Berechtigungen für die Rolle. Wird eine Verbundidentität authentifiziert, so wird die Identität der Rolle zugeordnet und erhält die von der Rolle definierten Berechtigungen. Informationen zu Rollen für den Verbund finden Sie im IAMBenutzerhandbuch unter Erstellen einer Rolle für einen externen Identitätsanbieter. Wenn Sie IAM Identity Center verwenden, konfigurieren Sie einen Berechtigungssatz. Um zu kontrollieren, worauf Ihre Identitäten nach der Authentifizierung zugreifen können, korreliert IAM Identity Center den Berechtigungssatz mit einer Rolle in. IAM Informationen zu Berechtigungssätzen finden Sie unter Berechtigungssätze im AWS IAM Identity Center Benutzerleitfaden.
- Temporäre IAM Benutzerberechtigungen — Ein IAM Benutzer oder eine Rolle kann eine IAM Rolle übernehmen, um vorübergehend verschiedene Berechtigungen für eine bestimmte Aufgabe zu übernehmen.
- Kontoübergreifender Zugriff — Sie können eine IAM Rolle verwenden, um einer Person (einem vertrauenswürdigen Principal) in einem anderen Konto den Zugriff auf Ressourcen in Ihrem Konto zu ermöglichen. Rollen stellen die primäre Möglichkeit dar, um kontoübergreifendem Zugriff zu gewähren. Allerdings mit einigen AWS-Services, Sie können eine Richtlinie direkt an eine Ressource anhängen (anstatt eine Rolle als Proxy zu verwenden). Informationen zum Unterschied zwischen Rollen und ressourcenbasierten Richtlinien für den kontenübergreifenden Zugriff finden Sie IAMim Benutzerhandbuch unter Kontoübergreifender Ressourcenzugriff. IAM
- Serviceübergreifender Zugriff — Einige AWS-Services Funktionen in anderen verwenden AWS-Services. Wenn Sie beispielsweise einen Service aufrufen, ist es üblich, dass dieser Service Anwendungen in Amazon ausführt EC2 oder Objekte in Amazon S3 speichert. Ein Dienst kann dies mit den Berechtigungen des aufrufenden Prinzipals mit einer Servicerolle oder mit einer serviceverknüpften Rolle tun.
  - Zugriffssitzungen weiterleiten (FAS) — Wenn Sie einen IAM Benutzer oder eine Rolle verwenden, um Aktionen auszuführen in AWS, Sie gelten als Principal. Bei einigen Services könnte es Aktionen geben, die dann eine andere Aktion in einem anderen Service initiieren. FASverwendet die Rechte des Prinzipals, der einen aufruft AWS-Service, kombiniert mit der Anfrage AWS-Service um Anfragen an nachgelagerte Dienste zu stellen. FASAnfragen werden nur gestellt, wenn ein Dienst eine Anfrage erhält, die Interaktionen mit anderen erfordert AWS-

Services oder Ressourcen zum Ausfüllen. In diesem Fall müssen Sie über Berechtigungen zum Ausführen beider Aktionen verfügen. Einzelheiten zu den Richtlinien für FAS Anfragen finden Sie unter Zugriffssitzungen weiterleiten.

- Servicerolle — Eine Servicerolle ist eine IAMRolle, die ein Dienst übernimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM Administrator kann eine Servicerolle von innen heraus erstellen, ändern und löschenIAM. Weitere Informationen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an ein AWS-Service im IAM-Benutzerhandbuch.
- Dienstverknüpfte Rolle — Eine dienstverknüpfte Rolle ist eine Art von Servicerolle, die mit einem verknüpft ist AWS-Service. Der Dienst kann die Rolle übernehmen, eine Aktion in Ihrem Namen durchzuführen. Mit Diensten verknüpfte Rollen erscheinen in Ihrem AWS-Konto und gehören dem Dienst. Ein IAM Administrator kann die Berechtigungen für dienstbezogene Rollen anzeigen, aber nicht bearbeiten.

- Anwendungen, die auf Amazon laufen EC2 — Sie können eine IAM Rolle verwenden, um temporäre Anmeldeinformationen für Anwendungen zu verwalten, die auf einer EC2 Instance ausgeführt werden und AWS CLI or AWS APIAnfragen. Dies ist dem Speichern von Zugriffsschlüsseln innerhalb der EC2 Instanz vorzuziehen. Um eine zuzuweisen AWS Sie erstellen ein EC2 Instanzprofil, das an die Instanz angehängt ist. Sie müssen einer Instanz eine Rolle zuweisen und sie allen ihren Anwendungen zur Verfügung stellen. Ein Instanzprofil enthält die Rolle und ermöglicht Programmen, die auf der EC2 Instanz ausgeführt werden, temporäre Anmeldeinformationen abzurufen. Weitere Informationen finden Sie im IAMBenutzerhandbuch unter Verwenden einer IAM Rolle zur Erteilung von Berechtigungen für Anwendungen, die auf EC2 Amazon-Instances ausgeführt werden.

Informationen darüber, ob Sie IAM Rollen oder IAM Benutzer verwenden sollten, finden Sie im Benutzerhandbuch unter Wann sollte eine IAM Rolle (anstelle eines IAM Benutzers) erstellt werden.

Verwalten des Zugriffs mit Richtlinien

Sie kontrollieren den Zugriff in AWS indem Sie Richtlinien erstellen und diese anhängen AWS Identitäten oder Ressourcen. Eine Richtlinie ist ein Objekt in AWS das, wenn es mit einer Identität oder Ressource verknüpft ist, ihre Berechtigungen definiert. AWS wertet diese Richtlinien aus, wenn ein Principal (Benutzer, Root-Benutzer oder Rollensitzung) eine Anfrage stellt. Berechtigungen in den Richtlinien bestimmen, ob die Anforderung zugelassen oder abgelehnt wird. Die meisten Richtlinien sind gespeichert in AWS als JSON Dokumente. Weitere Informationen zur Struktur und zum Inhalt von JSON Richtliniendokumenten finden Sie im IAMBenutzerhandbuch unter Überblick über JSON Richtlinien.

Administratoren können Folgendes verwenden AWS JSONRichtlinien, um festzulegen, wer Zugriff auf was hat. Das bedeutet, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Standardmäßig haben Benutzer, Gruppen und Rollen keine Berechtigungen. Um Benutzern die Erlaubnis zu erteilen, Aktionen mit den Ressourcen durchzuführen, die sie benötigen, kann ein IAM Administrator IAM Richtlinien erstellen. Der Administrator kann dann die IAM Richtlinien zu Rollen hinzufügen, und Benutzer können die Rollen übernehmen.

IAMRichtlinien definieren Berechtigungen für eine Aktion, unabhängig von der Methode, mit der Sie den Vorgang ausführen. Angenommen, es gibt eine Richtlinie, die Berechtigungen für die `iam:GetRole`-Aktion erteilt. Ein Benutzer mit dieser Richtlinie kann Rolleninformationen von der abrufen AWS Management Console, der AWS CLI, oder der AWS API.

Identitätsbasierte Richtlinien

Identitätsbasierte Richtlinien sind Dokumente mit JSON Berechtigungsrichtlinien, die Sie an eine Identität anhängen können, z. B. an einen IAM Benutzer, eine Benutzergruppe oder eine Rolle. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen einer identitätsbasierten Richtlinie finden Sie unter IAMRichtlinien erstellen im Benutzerhandbuch. IAM

Identitätsbasierte Richtlinien können weiter als Inline-Richtlinien oder verwaltete Richtlinien kategorisiert werden. Inline-Richtlinien sind direkt in einen einzelnen Benutzer, eine einzelne Gruppe oder eine einzelne Rolle eingebettet. Verwaltete Richtlinien sind eigenständige Richtlinien, die Sie mehreren Benutzern, Gruppen und Rollen in Ihrem Unternehmen zuordnen können AWS-Konto. Zu den verwalteten Richtlinien gehören AWS verwaltete Richtlinien und vom Kunden verwaltete Richtlinien. Informationen dazu, wie Sie zwischen einer verwalteten Richtlinie oder einer Inline-Richtlinie wählen können, finden Sie im IAMBenutzerhandbuch unter Auswahl zwischen verwalteten Richtlinien und Inline-Richtlinien.

Ressourcenbasierte Richtlinien

Ressourcenbasierte Richtlinien sind JSON Richtliniendokumente, die Sie an eine Ressource anhängen. Beispiele für ressourcenbasierte Richtlinien sind IAM Rollenvertrauensrichtlinien und Amazon S3 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. Prinzipale können Konten, Benutzer, Rollen, Verbundbenutzer oder AWS-Services.

Ressourcenbasierte Richtlinien sind Richtlinien innerhalb dieses Diensts. Sie können nicht verwenden AWS verwaltete Richtlinien aus IAM einer ressourcenbasierten Richtlinie.

Zugriffskontrolllisten () ACLs

Zugriffskontrolllisten (ACLs) steuern, welche Principals (Kontomitglieder, Benutzer oder Rollen) über Zugriffsberechtigungen für eine Ressource verfügen. ACLsähneln ressourcenbasierten Richtlinien, verwenden jedoch nicht das JSON Richtliniendokumentformat.

Amazon S3 S3, AWS WAF, und Amazon VPC sind Beispiele für Dienste, die unterstützenACLs. Weitere Informationen finden Sie unter Übersicht über ACLs die Zugriffskontrollliste (ACL) im Amazon Simple Storage Service Developer Guide.

Weitere Richtlinientypen

AWS unterstützt zusätzliche, weniger verbreitete Richtlinientypen. Diese Richtlinientypen können die maximalen Berechtigungen festlegen, die Ihnen von den häufiger verwendeten Richtlinientypen erteilt werden können.

- Berechtigungsgrenzen — Eine Berechtigungsgrenze ist eine erweiterte Funktion, mit der Sie die maximalen Berechtigungen festlegen, die eine identitätsbasierte Richtlinie einer IAM Entität (IAMBenutzer oder Rolle) gewähren kann. Sie können eine Berechtigungsgrenze für eine Entität festlegen. Die daraus resultierenden Berechtigungen sind der Schnittpunkt der identitätsbasierten Richtlinien einer Entität und ihrer Berechtigungsgrenzen. Ressourcenbasierte Richtlinien, die den Benutzer oder die Rolle im Feld `Principal` angeben, werden nicht durch Berechtigungsgrenzen eingeschränkt. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen zu Berechtigungsgrenzen finden Sie im IAMBenutzerhandbuch unter Berechtigungsgrenzen für IAM Entitäten.
- Dienststeuerungsrichtlinien (SCPs) — SCPs sind JSON Richtlinien, die die maximalen Berechtigungen für eine Organisation oder Organisationseinheit (OU) in festlegen AWS Organizations. AWS Organizations ist ein Dienst zur Gruppierung und zentralen Verwaltung mehrerer AWS-Konten den Ihr Unternehmen besitzt. Wenn Sie alle Funktionen in einer Organisation aktivieren, können Sie Richtlinien zur Servicesteuerung (SCPs) auf einige oder alle Ihre Konten anwenden. Das SCP schränkt die Berechtigungen für Entitäten in Mitgliedskonten ein, einschließlich der einzelnen Root-Benutzer des AWS-Kontos. Weitere Informationen

zu Organizations und finden Sie SCPs unter Richtlinien zur Servicesteuerung in der AWS Organizations Benutzerleitfaden.

- Sitzungsrichtlinien – Sitzungsrichtlinien sind erweiterte Richtlinien, die Sie als Parameter übergeben, wenn Sie eine temporäre Sitzung für eine Rolle oder einen verbundenen Benutzer programmgesteuert erstellen. Die resultierenden Sitzungsberechtigungen sind eine Schnittmenge der auf der Identität des Benutzers oder der Rolle basierenden Richtlinien und der Sitzungsrichtlinien. Berechtigungen können auch aus einer ressourcenbasierten Richtlinie stammen. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen finden Sie im IAMBenutzerhandbuch unter Sitzungsrichtlinien.

Mehrere Richtlinientypen

Wenn mehrere auf eine Anforderung mehrere Richtlinientypen angewendet werden können, sind die entsprechenden Berechtigungen komplizierter. Um zu erfahren, wie AWS bestimmt, ob eine Anfrage zulässig ist, wenn mehrere Richtlinientypen betroffen sind, siehe Bewertungslogik für Richtlinien im IAMBenutzerhandbuch.

Wie funktioniert EMR Serverless mit IAM

Informieren Sie sich vor der Verwendung IAM zur Verwaltung des Zugriffs auf Amazon EMR Serverless, welche IAM Funktionen für Amazon EMR Serverless verfügbar sind.

IAMFunktionen, die Sie mit Serverless verwenden können EMR

IAMFunktion	EMRServerloser Support von Amazon
Identitätsbasierte Richtlinien	Ja
Ressourcenbasierte Richtlinien	Nein
Richtlinienaktionen	Ja
Richtlinienressourcen	Ja
Bedingungsschlüssel für die Richtlinie	Nein
ACLs	Nein
ABAC(Tags in Richtlinien)	Ja

IAMFunktion	EMRServerloser Support von Amazon
Temporäre Anmeldeinformationen	Ja
Hauptberechtigungen	Ja
Servicerollen	Nein
Serviceverknüpfte Rollen	Ja

Um einen allgemeinen Überblick darüber zu erhalten, wie EMR Serverless und andere AWS Dienste funktionieren mit den meisten IAM Funktionen, siehe AWS Dienste, mit denen IAM im IAMBenutzerhandbuch gearbeitet werden kann.

Identitätsbasierte Richtlinien für Serverless EMR

Unterstützt Richtlinien auf Identitätsbasis: Ja

Identitätsbasierte Richtlinien sind Dokumente mit JSON Berechtigungsrichtlinien, die Sie an eine Identität anhängen können, z. B. an einen Benutzer, eine IAM Benutzergruppe oder eine Rolle. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen einer identitätsbasierten Richtlinie finden Sie unter IAMRichtlinien erstellen im Benutzerhandbuch. IAM

Mit IAM identitätsbasierten Richtlinien können Sie zulässige oder verweigerte Aktionen und Ressourcen sowie die Bedingungen angeben, unter denen Aktionen zulässig oder verweigert werden. Sie können den Prinzipal nicht in einer identitätsbasierten Richtlinie angeben, da er für den Benutzer oder die Rolle gilt, dem er zugeordnet ist. Weitere Informationen zu allen Elementen, die Sie in einer JSON Richtlinie verwenden können, finden Sie in der Referenz zu den IAM JSON Richtlinienelementen im IAMBenutzerhandbuch.

Beispiele für identitätsbasierte Richtlinien für Serverless EMR

Beispiele für identitätsbasierte Richtlinien von Amazon EMR Serverless finden Sie unter. Beispiele für identitätsbasierte Richtlinien für Serverless EMR

Ressourcenbasierte Richtlinien innerhalb von Serverless EMR

Unterstützt ressourcenbasierte Richtlinien: Nein

Ressourcenbasierte Richtlinien sind JSON Richtliniendokumente, die Sie an eine Ressource anhängen. Beispiele für ressourcenbasierte Richtlinien sind IAM Rollenvertrauensrichtlinien und Amazon S3 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. Prinzipale können Konten, Benutzer, Rollen, Verbundbenutzer oder AWS-Services.

Um den kontoübergreifenden Zugriff zu ermöglichen, können Sie in einer ressourcenbasierten Richtlinie ein ganzes Konto oder IAM Entitäten in einem anderen Konto als Prinzipal angeben. Durch das Hinzufügen eines kontoübergreifenden Auftraggebers zu einer ressourcenbasierten Richtlinie ist nur die halbe Vertrauensbeziehung eingerichtet. Wenn sich der Prinzipal und die Ressource unterscheiden AWS-Konten, muss ein IAM Administrator des vertrauenswürdigen Kontos auch der Prinzipalentität (Benutzer oder Rolle) die Erlaubnis erteilen, auf die Ressource zuzugreifen. Sie erteilen Berechtigungen, indem Sie der juristischen Stelle eine identitätsbasierte Richtlinie anfügen. Wenn jedoch eine ressourcenbasierte Richtlinie Zugriff auf einen Prinzipal in demselben Konto gewährt, ist keine zusätzliche identitätsbasierte Richtlinie erforderlich. Weitere Informationen finden Sie IAMim IAMBenutzerhandbuch unter Kontoübergreifender Ressourcenzugriff.

Richtlinienaktionen für EMR Serverless

Unterstützt Richtlinienaktionen: Ja

Administratoren können Folgendes verwenden AWS JSONRichtlinien, um festzulegen, wer Zugriff auf was hat. Das bedeutet, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Das `Action` Element einer JSON Richtlinie beschreibt die Aktionen, mit denen Sie den Zugriff in einer Richtlinie zulassen oder verweigern können. Richtlinienaktionen haben normalerweise denselben Namen wie die zugehörigen AWS APIBetrieb. Es gibt einige Ausnahmen, z. B. Aktionen, für die nur Berechtigungen erforderlich sind und für die es keine entsprechende Operation gibt. API Es gibt auch einige Operationen, die mehrere Aktionen in einer Richtlinie erfordern. Diese zusätzlichen Aktionen werden als abhängige Aktionen bezeichnet.

Schließen Sie Aktionen in eine Richtlinie ein, um Berechtigungen zur Durchführung der zugeordneten Operation zu erteilen.

Eine Liste der EMR serverlosen Aktionen finden Sie unter Aktionen, Ressourcen und Bedingungsschlüssel für Amazon EMR Serverless in der Service Authorization Reference.

Bei Richtlinienaktionen in EMR Serverless wird vor der Aktion das folgende Präfix verwendet.

```
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 Serverless EMR

Richtlinienressourcen für Serverless EMR

Unterstützt Richtlinienressourcen: Ja

Administratoren können Folgendes verwenden AWS JSONRichtlinien, um festzulegen, wer Zugriff auf was hat. Das bedeutet, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Das `Resource` JSON Richtlinienelement gibt das Objekt oder die Objekte an, für die die Aktion gilt. Anweisungen müssen entweder ein – `Resource`oder ein `NotResource`-Element enthalten. Es hat sich bewährt, eine Ressource mit ihrem Amazon-Ressourcennamen (ARN) anzugeben. Sie können dies für Aktionen tun, die einen bestimmten Ressourcentyp unterstützen, der als Berechtigungen auf Ressourcenebene bezeichnet wird.

Verwenden Sie für Aktionen, die keine Berechtigungen auf Ressourcenebene unterstützen, z. B. Auflistungsoperationen, 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 zu den ARN Aktionen, die Sie für jede Ressource angeben können, 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 Serverless EMR

Bedingungsschlüssel für Richtlinien für Serverless EMR

Unterstützt servicespezifische Richtlinienbedingungsschlüssel	Nein

Administratoren können Folgendes verwenden AWS JSONRichtlinien, um festzulegen, wer Zugriff auf was hat. Das heißt, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Das Element `Condition` (oder `Condition` block) ermöglicht Ihnen die Angabe der Bedingungen, unter denen eine Anweisung wirksam ist. Das Element `Condition` ist optional. 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.

Wenn Sie mehrere `Condition` Elemente in einer Anweisung oder mehrere Schlüssel in einem einzelnen `Condition` Element angeben, AWS wertet sie mithilfe einer logischen `AND` Operation aus. Wenn Sie mehrere Werte für einen einzelnen Bedingungsschlüssel angeben, AWS wertet die Bedingung mithilfe einer logischen `OR` Operation aus. Alle Bedingungen müssen erfüllt werden, bevor die Berechtigungen der Anweisung gewährt werden.

Sie können auch Platzhaltervariablen verwenden, wenn Sie Bedingungen angeben. Sie können einem IAM Benutzer beispielsweise nur dann Zugriff auf eine Ressource gewähren, wenn sie mit seinem IAM Benutzernamen gekennzeichnet ist. Weitere Informationen finden Sie im IAMBenutzerhandbuch unter IAMRichtlinienelemente: Variablen und Tags.

AWS unterstützt globale Bedingungsschlüssel und dienstspezifische Bedingungsschlüssel. Um alle zu sehen AWS globale Bedingungsschlüssel finden Sie unter AWS Kontexttasten für globale Bedingungen im IAMBenutzerhandbuch.

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 EC2 Amazon-Aktionen unterstützen die Tasten `aws:RequestedRegion` und `ec2:Region` Condition. Weitere Informationen finden Sie unter Beispiel: Beschränkung des Zugriffs auf eine bestimmte Region.

Zugriffskontrolllisten (ACLs) in Serverless EMR

UnterstütztACLs: Nein

Zugriffskontrolllisten (ACLs) steuern, welche Principals (Kontomitglieder, Benutzer oder Rollen) über Zugriffsberechtigungen für eine Ressource verfügen. ACLsähneln ressourcenbasierten Richtlinien, verwenden jedoch nicht das JSON Richtliniendokumentformat.

Attributbasierte Zugriffskontrolle () mit Serverless ABAC EMR

Unterstützt ABAC (Tags in Richtlinien)	Ja

Die attributbasierte Zugriffskontrolle (ABAC) ist eine Autorisierungsstrategie, bei der Berechtigungen auf der Grundlage von Attributen definiert werden. In AWS, diese Attribute werden Tags genannt. Sie können Tags an IAM Entitäten (Benutzer oder Rollen) und an viele andere anhängen AWS Ressourcen schätzen. Das Markieren von Entitäten und Ressourcen ist der erste Schritt vonABAC. Anschließend entwerfen Sie ABAC Richtlinien, die Operationen zulassen, wenn das Tag des Prinzipals mit dem Tag auf der Ressource übereinstimmt, auf die er zugreifen möchte.

ABACist hilfreich in Umgebungen, die schnell wachsen, und hilft in Situationen, in denen die Richtlinienverwaltung umständlich wird.

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 finden Sie ABAC unter Was ist? ABAC im IAMBenutzerhandbuch. Ein Tutorial mit Schritten zur Einrichtung finden Sie im ABAC Benutzerhandbuch unter Verwenden der attributebasierten Zugriffskontrolle (ABAC). IAM

Temporäre Anmeldeinformationen mit Serverless verwenden EMR

Unterstützt temporäre Anmeldeinformationen: Ja

Etwas AWS-Services funktioniert nicht, wenn Sie sich mit temporären Anmeldeinformationen anmelden. Für zusätzliche Informationen, einschließlich AWS-Services mit temporären Anmeldeinformationen arbeiten, finden Sie unter AWS-Services mit denen IAM im IAMBenutzerhandbuch gearbeitet werden kann.

Sie verwenden temporäre Anmeldeinformationen, wenn Sie sich bei der AWS Management Console mit einer beliebigen Methode außer einem Benutzernamen und einem Passwort. Zum Beispiel, wenn Sie darauf zugreifen AWS Wenn Sie den Single Sign-On-Link (SSO) Ihres Unternehmens verwenden, erstellt dieser Prozess automatisch temporäre Anmeldeinformationen. Sie erstellen auch automatisch temporäre Anmeldeinformationen, wenn Sie sich als Benutzer bei der Konsole anmelden und dann die Rollen wechseln. Weitere Informationen zum Rollenwechsel finden Sie unter Wechseln zu einer Rolle (Konsole) im IAMBenutzerhandbuch.

Sie können temporäre Anmeldeinformationen manuell erstellen, indem Sie den AWS CLI or AWS API. Sie können dann diese temporären Anmeldeinformationen für den Zugriff verwenden AWS. AWS empfiehlt, temporäre Anmeldeinformationen dynamisch zu generieren, anstatt langfristige Zugriffsschlüssel zu verwenden. Weitere Informationen finden Sie unter Temporäre Sicherheitsanmeldeinformationen unter IAM.

Serviceübergreifende Prinzipalberechtigungen für Serverless EMR

Unterstützt Forward-Access-Sitzungen (FAS): Ja

Wenn Sie einen IAM Benutzer oder eine Rolle verwenden, um Aktionen auszuführen in AWS, gelten Sie als Principal. Bei einigen Services könnte es Aktionen geben, die dann eine andere Aktion in einem anderen Service initiieren. FASverwendet die Rechte des Prinzipals, der einen aufruft AWS-Service, kombiniert mit der Anfrage AWS-Service um Anfragen an nachgelagerte Dienste zu stellen. FASAnfragen werden nur gestellt, wenn ein Dienst eine Anfrage erhält, die Interaktionen mit anderen erfordert AWS-Services oder Ressourcen zum Ausfüllen. In diesem Fall müssen Sie über Berechtigungen zum Ausführen beider Aktionen verfügen. Einzelheiten zu den Richtlinien für FAS Anfragen finden Sie unter Zugriffssitzungen weiterleiten.

Servicerollen für EMR Serverless

Unterstützt Servicerollen	Nein

Dienstbezogene Rollen für Serverless EMR

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 Yes (Ja) aus, um die Dokumentation für die serviceverknüpfte Rolle für diesen Service anzuzeigen.

Verwenden von serviceverknüpften Rollen für Serverless EMR

Amazon EMR Serverless verwendet AWS Identity and Access Management (IAM) Rollen, die mit Diensten verknüpft sind. Eine dienstgebundene Rolle ist ein einzigartiger Rollentyp, der IAM direkt mit Serverless verknüpft ist. EMR Dienstbezogene Rollen sind von EMR Serverless vordefiniert und beinhalten alle Berechtigungen, die der Dienst benötigt, um andere Rollen aufzurufen AWS Dienste in Ihrem Namen.

Eine dienstbezogene Rolle erleichtert die Einrichtung von EMR Serverless, da Sie die erforderlichen Berechtigungen nicht manuell hinzufügen müssen. EMRServerless definiert die Berechtigungen seiner dienstbezogenen Rollen, und sofern nicht anders definiert, kann nur EMR Serverless seine Rollen übernehmen. Zu den definierten Berechtigungen gehören die Vertrauensrichtlinie und die Berechtigungsrichtlinie, und diese Berechtigungsrichtlinie kann keiner anderen Entität zugeordnet werden. IAM

Sie können eine serviceverknüpfte Rolle erst löschen, nachdem ihre verwandten Ressourcen gelöscht wurden. Dadurch werden Ihre EMR serverlosen Ressourcen geschützt, da Sie die Zugriffsberechtigung für die Ressourcen nicht versehentlich entfernen können.

Informationen zu anderen Diensten, die dienstbezogene Rollen unterstützen, finden Sie unter AWS Dienste, die mit Diensten funktionieren, IAM und suchen nach den Diensten, für die in der Spalte Dienstbezogene Rollen die Option Ja steht. Wählen Sie über einen Link Ja aus, um die Dokumentation zu einer serviceverknüpften Rolle für diesen Service anzuzeigen.

Dienstbezogene Rollenberechtigungen für Serverless EMR

EMRServerless verwendet die benannte dienstverknüpfte Rolle, um das Aufrufen AWSServiceRoleForAmazonEMRServerlesszu ermöglichen AWS APIsin Ihrem Namen.

Die AWSServiceRoleForAmazonEMRServerless dienstbezogene 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 an den 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 A mazonEMRServerless ServiceRolePolicy in der AWS Management Console.

- 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.

```
{
    "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 Principal diese Rolle übernehmen kann.

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

```
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "ops.emr-serverless.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Sie müssen Berechtigungen konfigurieren, damit eine IAM Entität (z. B. ein Benutzer, eine Gruppe oder eine Rolle) eine dienstverknüpfte Rolle erstellen, bearbeiten oder löschen kann. Weitere Informationen finden Sie unter Berechtigungen für dienstbezogene Rollen im IAMBenutzerhandbuch.

Eine dienstverknüpfte Rolle für Serverless erstellen EMR

Sie müssen eine serviceverknüpfte Rolle nicht manuell erstellen. Wenn Sie eine neue EMR serverlose Anwendung in der AWS Management Console (mit EMR Studio) AWS CLI, oder das AWS API, EMR Serverless erstellt die serviceverknüpfte Rolle für Sie. Sie müssen Berechtigungen konfigurieren, damit eine IAM Entität (z. B. ein Benutzer, eine Gruppe oder eine Rolle) eine dienstbezogene Rolle erstellen, bearbeiten oder löschen kann.

Um die AWSServiceRoleForAmazonEMRServerless dienstverknüpfte Rolle zu erstellen, verwenden Sie IAM

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 serviceverknüpfte Rolle löschen und sie dann erneut erstellen müssen, können Sie dasselbe Verfahren anwenden, um die Rolle in Ihrem Konto neu anzulegen. Wenn Sie eine neue

EMR serverlose Anwendung erstellen, erstellt EMR Serverless die dienstverknüpfte Rolle erneut für Sie.

Sie können die IAM Konsole auch verwenden, um eine dienstverknüpfte Rolle für den Serverless-Anwendungsfall zu erstellen. EMR Im AWS CLI oder das AWS API, erstellen Sie eine dienstbezogene Rolle mit dem `ops.emr-serverless.amazonaws.com` Dienstnamen. Weitere Informationen finden Sie im IAMBenutzerhandbuch unter Erstellen einer dienstbezogenen Rolle. Wenn Sie diese serviceverknüpfte Rolle löschen, können Sie mit demselben Verfahren die Rolle erneut erstellen.

Eine dienstverknüpfte Rolle für Serverless bearbeiten EMR

EMRServerless ermöglicht es Ihnen nicht, die AWSServiceRoleForAmazonEMRServerless dienstverknüpfte Rolle zu bearbeiten, da verschiedene Entitäten möglicherweise auf die Rolle verweisen. Sie können die nicht bearbeiten AWS-eigene IAM Richtlinie, die von der Rolle EMR „Serverless Service Linked“ verwendet wird, da sie alle erforderlichen Berechtigungen EMR enthält, die Serverless benötigt. Sie können die Beschreibung der Rolle jedoch mithilfe von bearbeiten. IAM

Um die Beschreibung der AWSServiceRoleForAmazonEMRServerless serviceverknüpften Rolle zu bearbeiten, verwenden Sie IAM

Fügen Sie der Berechtigungsrichtlinie für die IAM Entität, die die Beschreibung einer dienstbezogenen Rolle bearbeiten muss, die folgende Anweisung hinzu.

```
{
    "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 im IAMBenutzerhandbuch unter Bearbeiten einer dienstbezogenen Rolle.

Löschen einer dienstverknüpften Rolle für Serverless EMR

Wenn Sie ein Feature oder einen Service, die bzw. der eine serviceverknüpfte Rolle erfordert, nicht mehr benötigen, sollten Sie diese Rolle löschen. Auf diese Weise haben Sie keine ungenutzte Entität, die nicht aktiv überwacht oder verwaltet wird. Sie müssen jedoch alle EMR serverlosen Anwendungen in allen Regionen löschen, bevor Sie die dienstverknüpfte Rolle löschen können.

> **Note**
>
> Wenn der EMR serverlose 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 AWSServiceRoleForAmazonEMRServerless Dienst verknüpfte Rolle zu löschen, verwenden Sie IAM

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"}}
}
```

Um die dienstverknüpfte Rolle manuell zu löschen, verwenden Sie IAM

Verwenden Sie die IAM Konsole, die AWS CLI, oder das AWS APIum die mit dem AWSServiceRoleForAmazonEMRServerless Dienst verknüpfte Rolle zu löschen. Weitere Informationen finden Sie im IAMBenutzerhandbuch unter Löschen einer dienstbezogenen Rolle.

Unterstützte Regionen für EMR serverlose dienstverknüpfte Rollen

EMRServerless unterstützt die Verwendung von dienstbezogenen Rollen in allen Regionen, in denen der Dienst 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 EMR serverloser Job annehmen kann, wenn er in Ihrem Namen andere Dienste 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 Zugriff auf AWS Datenkatalog Glue.

Um Zugriff auf andere hinzuzufügen AWS Bei Ressourcen wie DynamoDB müssen Sie beim Erstellen der Runtime-Rolle die entsprechenden Berechtigungen in die Richtlinie aufnehmen.

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

```
            "Sid": "FullAccessToS3Bucket",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
            ]
        },
        {
            "Sid": "GlueCreateAndReadDataCatalog",
            "Effect": "Allow",
            "Action": [
                "glue:GetDatabase",
                "glue:CreateDatabase",
                "glue:GetDataBases",
                "glue:CreateTable",
                "glue:GetTable",
                "glue:UpdateTable",
                "glue:DeleteTable",
                "glue:GetTables",
                "glue:GetPartition",
                "glue:GetPartitions",
                "glue:CreatePartition",
                "glue:BatchCreatePartition",
                "glue:GetUserDefinedFunctions"
            ],
            "Resource": ["*"]
        }
    ]
}
```

Übergeben Sie Rollenberechtigungen

Sie können der Rolle eines Benutzers IAM Berechtigungsrichtlinien zuordnen, sodass der Benutzer nur genehmigte Rollen weitergeben kann. Auf diese Weise können Administratoren steuern, welche Benutzer bestimmte Job-Runtime-Rollen an EMR serverlose Jobs übergeben können. Weitere Informationen zum Festlegen von Berechtigungen finden Sie unter Einem Benutzer Berechtigungen zur Übergabe einer Rolle an einen AWS Dienst.

Im Folgenden finden Sie ein Beispiel für eine Richtlinie, 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"
                }
            }
}
```

Beispiele für Benutzerzugriffsrichtlinien für Serverless EMR

Sie können detaillierte Richtlinien für Ihre Benutzer einrichten, je nachdem, welche Aktionen jeder Benutzer bei der Interaktion mit serverlosen Anwendungen ausführen soll. EMR Die folgenden Richtlinien sind Beispiele, die Ihnen dabei helfen können, die richtigen Berechtigungen für Ihre Benutzer einzurichten. Dieser Abschnitt konzentriert sich nur auf EMR serverlose Richtlinien. Beispiele für EMR Studio-Benutzerrichtlinien finden Sie unter EMR Studio-Benutzerberechtigungen konfigurieren. Informationen zum Anhängen von Richtlinien an IAM Benutzer (Prinzipale) finden Sie im IAM Benutzerhandbuch unter IAMRichtlinien verwalten.

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 eine Beispielrichtlinie, die es Hauptbenutzern ermöglicht, EMR serverlose Anwendungen zu erstellen und zu ändern sowie andere Aktionen wie das Senden und Debuggen von Jobs auszuführen. Sie zeigt alle Aktionen, die EMR Serverless für andere Dienste benötigt.

```
{
    "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 für Ihre EMR serverlosen Anwendungen aktivierenVPC, erstellen Sie Amazon EC2 Elastic Network Interfaces (ENIs) für die Kommunikation mit VPC Ressourcen. Die folgende Richtlinie stellt sicher, dass neue EC2 ENIs Anwendungen nur im Kontext EMR serverloser Anwendungen erstellt werden.

> **Note**
>
> Es wird dringend empfohlen, diese Richtlinie festzulegen, um sicherzustellen, dass Benutzer EC2 ENIs nur dann etwas erstellen können, wenn EMR serverlose Anwendungen gestartet werden.

```
{
    "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 EMR serverlosen Zugriff auf bestimmte Subnetze einschränken möchten, können Sie jedes Subnetz mit einer Tag-Bedingung kennzeichnen. Diese IAM Richtlinie stellt sicher, dass EMR serverlose Anwendungen nur innerhalb der zulässigen Subnetze erstellt EC2 ENIs werden 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 EMR serverlose, dienstverknüpfte Rolle erstellen können. Weitere Informationen hierzu finden Sie unter Verwenden von serviceverknüpften Rollen für Serverless EMR.

Die folgende IAM Richtlinie ermöglicht es Ihnen, eine EMR serverlose, dienstverknüpfte 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 ein Beispiel für eine Richtlinie, die Benutzern nur Leseberechtigungen für EMR serverlose 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.

```
{
    "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 so einschränken, dass sie nur Jobs an EMR serverlose Anwendungen senden können, die mit ihrem Teamnamen gekennzeichnet sind.

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

```
    "Statement": [
        {
            "Sid": "EMRServerlessActions",
            "Effect": "Allow",
            "Action": [
                "emr-serverless:ListApplications",
                "emr-serverless:GetApplication",
                "emr-serverless:StartApplication",
                "emr-serverless:StartJobRun",
                "emr-serverless:CancelJobRun",
                "emr-serverless:ListJobRuns",
                "emr-serverless:GetJobRun"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Team": "team-name"
                }
            }
        }
    ]
}
```

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 serverlosen Bedingungsschlüsseln. Diese IAM Richtlinienerklärungen 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 zur Planung und zum Testen von IAM Richtlinien finden Sie im IAMBenutzerhandbuch.

> **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, die zur Steuerung der Aktionen verwendet werden, die bei EMR serverlosen 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.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-serverless:GetApplication"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "emr-serverless:ResourceTag/department": "dev"
        }
      }
    }
  ]
}
```

Sie können auch mehrere Tag-Werte mithilfe eines Bedingungsoperators angeben. Um beispielsweise Aktionen für Anwendungen zuzulassen, bei denen das `department` Tag den Wert `dev` oder enthält`test`, könnten Sie den Bedingungsblock aus dem vorherigen Beispiel durch den folgenden ersetzen.

```
"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.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-serverless:CreateApplication"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "emr-serverless:RequestTag/department": "dev"
        }
      }
    }
  ]
}
```

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.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-serverless:CreateApplication"
      ],
      "Resource": "*",
      "Condition": {
        "Null": {
```

```
          "emr-serverless:RequestTag/department": "false"
        }
      }
    }
  ]
}
```

Zugriff zum Hinzufügen und Entfernen von Tags verweigern

Diese Richtlinie verhindert, dass Benutzer Tags zu EMR serverlosen Anwendungen hinzufügen oder entfernen, deren Wert nicht `dev` angegeben ist. `department`

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "emr-serverless:TagResource",
        "emr-serverless:UntagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "emr-serverless:ResourceTag/department": "dev"
        }
      }
    }
  ]
}
```

Beispiele für identitätsbasierte Richtlinien für Serverless EMR

Standardmäßig sind Benutzer und Rollen nicht berechtigt, Amazon EMR Serverless-Ressourcen zu erstellen oder zu ändern. Sie können auch keine Aufgaben mit dem ausführen AWS Management Console, AWS Command Line Interface (AWS CLI), oder AWS API. Um Benutzern die Erlaubnis zu erteilen, Aktionen mit den Ressourcen durchzuführen, die sie benötigen, kann ein IAM Administrator IAM Richtlinien erstellen. Der Administrator kann dann die IAM Richtlinien zu Rollen hinzufügen, und Benutzer können die Rollen übernehmen.

Informationen zum Erstellen einer IAM identitätsbasierten Richtlinie anhand dieser JSON Beispieldokumente finden Sie unter IAMRichtlinien erstellen im IAMBenutzerhandbuch.

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

- Bewährte Methoden für Richtlinien
- Gewähren der Berechtigung zur Anzeige der eigenen Berechtigungen für Benutzer

Bewährte Methoden für Richtlinien

> **Note**
>
> EMRServerless unterstützt keine verwalteten Richtlinien, sodass die erste unten aufgeführte Vorgehensweise nicht zutrifft.

Identitätsbasierte Richtlinien legen fest, ob jemand Amazon EMR Serverless-Ressourcen in Ihrem Konto erstellen, darauf zugreifen oder diese löschen kann. Durch diese Aktionen können Kosten für Sie entstehen AWS-Konto. Beachten Sie beim Erstellen oder Bearbeiten identitätsbasierter Richtlinien die folgenden Richtlinien und Empfehlungen:

- Beginnen Sie mit AWS verwaltete Richtlinien und Umstellung auf Berechtigungen mit den geringsten Rechten — Um zu beginnen, Ihren Benutzern und Workloads Berechtigungen zu gewähren, verwenden Sie AWS verwaltete Richtlinien, die Berechtigungen für viele gängige Anwendungsfälle gewähren. Sie sind in Ihrem verfügbar AWS-Konto. Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie Folgendes definieren AWS vom Kunden verwaltete Richtlinien, die speziell auf Ihre Anwendungsfälle zugeschnitten sind. Weitere Informationen finden Sie unter AWS verwaltete Richtlinien oder AWS verwaltete Richtlinien für Jobfunktionen im IAMBenutzerhandbuch.
- Berechtigungen mit den geringsten Rechten anwenden — Wenn Sie Berechtigungen mit IAM Richtlinien festlegen, gewähren Sie nur die Berechtigungen, die für die Ausführung 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 IAM zum Anwenden von Berechtigungen finden Sie IAMim Benutzerhandbuch unter Richtlinien und Berechtigungen. IAM

- Verwenden Sie Bedingungen in IAM Richtlinien, um den Zugriff weiter einzuschränken — Sie können Ihren Richtlinien eine Bedingung hinzufügen, um den Zugriff auf Aktionen und Ressourcen einzuschränken. Sie können beispielsweise eine Richtlinienbedingung schreiben, um anzugeben, dass alle Anfragen über gesendet werden müssenSSL. Sie können auch Bedingungen verwenden, um Zugriff auf Serviceaktionen zu gewähren, wenn diese über eine bestimmte AWS-Service, wie beispielsweise AWS CloudFormation. Weitere Informationen finden Sie unter IAMJSONRichtlinienelemente: Zustand im IAMBenutzerhandbuch.
- Verwenden Sie IAM Access Analyzer, um Ihre IAM Richtlinien zu validieren, um sichere und funktionale Berechtigungen zu gewährleisten. IAM Access Analyzer validiert neue und bestehende Richtlinien, sodass die Richtlinien der IAM Richtliniensprache (JSON) und den IAM bewährten Methoden entsprechen. IAMAccess Analyzer bietet mehr als 100 Richtlinienprüfungen und umsetzbare Empfehlungen, um Sie bei der Erstellung sicherer und funktionaler Richtlinien zu unterstützen. Weitere Informationen finden Sie unter IAMAccess Analyzer-Richtlinienvalidierung im IAMBenutzerhandbuch.
- Multi-Faktor-Authentifizierung erforderlich (MFA) — Wenn Sie ein Szenario haben, das IAM Benutzer oder einen Root-Benutzer in Ihrem System erfordert AWS-Konto, schalten Sie MFA für zusätzliche Sicherheit ein. Wenn Sie festlegen möchten, MFA wann API Operationen aufgerufen werden, fügen Sie MFA Bedingungen zu Ihren Richtlinien hinzu. Weitere Informationen finden Sie unter Konfiguration des MFA -geschützten API Zugriffs im IAMBenutzerhandbuch.

Weitere Informationen zu bewährten Methoden finden Sie unter Bewährte Sicherheitsmethoden IAM im IAM Benutzerhandbuch. IAM

Gewähren der Berechtigung zur Anzeige der eigenen Berechtigungen für Benutzer

Dieses Beispiel zeigt, wie Sie eine Richtlinie erstellen könnten, die es IAM Benutzern ermöglicht, die internen und verwalteten Richtlinien einzusehen, die mit ihrer Benutzeridentität verknüpft sind. Diese Richtlinie umfasst Berechtigungen zum Ausführen dieser Aktion auf der Konsole oder programmgesteuert mithilfe von AWS CLI or AWS API.

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

```
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon EMR Serverless-Updates für AWS Verwaltete Richtlinien

Einzelheiten zu Updates anzeigen für AWS verwaltete Richtlinien für Amazon EMR Serverless, 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.

Änderung	Beschreibung	Datum
A mazonEMRServerless ServiceRolePolicy — Aktualisi	Amazon EMR Serverless hat das neue `Sid` `CloudWatc`	25. Januar 2024

Änderung	Beschreibung	Datum
erung einer bestehenden Richtlinie	`hPolicyStatement` und `EC2PolicyStatement` zur mazonEMRServerless ServiceRolePolicy A-Richtlinie hinzugefügt.	
A mazonEMRServerless ServiceRolePolicy — Aktualisierung einer bestehenden Richtlinie	Amazon EMR Serverless hat neue Berechtigungen hinzugefügt, die es Amazon EMR Serverless ermöglichen, aggregierte Kontokennzahlen für die CPU Verwendung von v im Namespace zu veröffentlichen. `"AWS/Usage"`	20. April 2023
Amazon EMR Serverless hat mit der Nachverfolgung von Änderungen begonnen	Amazon EMR Serverless hat begonnen, Änderungen für seine AWS verwaltete Richtlinien.	20. April 2023

Fehlerbehebung bei Amazon EMR Serverless Identity and Access

Verwenden Sie die folgenden Informationen, um häufig auftretende 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 Leute außerhalb meines AWS Konto für den Zugriff auf meine Amazon EMR Serverless-Ressourcen

Ich bin nicht berechtigt, eine Aktion in Amazon EMR Serverless durchzuführen

Wenn das Symbol AWS Management Console teilt Ihnen mit, dass Sie nicht berechtigt sind, eine Aktion durchzuführen. Dann müssen Sie sich an Ihren Administrator wenden, um Unterstützung zu erhalten. Ihr Administrator ist die Person, die Ihnen Ihren Benutzernamen und Ihr Passwort bereitgestellt hat.

Der folgende Beispielfehler tritt auf, wenn der `mateojackson`-Benutzer versucht, die Konsole zum Anzeigen von Details zu einer fiktiven *`my-example-widget`*-Ressource zu verwenden, jedoch nicht über `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.

Etwas 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 AWS Administrator. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich möchte Leute außerhalb meines AWS Konto für den Zugriff auf meine Amazon EMR Serverless-Ressourcen

Sie können eine Rolle erstellen, die Benutzer in anderen Konten oder Personen außerhalb Ihrer Organisation für den Zugriff auf Ihre Ressourcen verwenden 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 unterIdentity and Access Management (IAM) in Amazon EMR Serverless.
- Um zu erfahren, wie Sie Zugriff auf Ihre Ressourcen gewähren können AWS-Konten die Ihnen gehören, finden Sie unter Gewähren des Zugriffs für einen IAM Benutzer in einem anderen AWS-Konto die Sie besitzen, finden Sie im IAMBenutzerhandbuch.
- Um zu erfahren, wie Sie Dritten Zugriff auf Ihre Ressourcen gewähren können AWS-Konten, siehe Zugriff gewähren auf AWS-Konten Eigentum Dritter im IAMBenutzerhandbuch.
- Informationen zur Bereitstellung des Zugriffs über einen Identitätsverbund finden Sie im Benutzerhandbuch unter Zugriff für extern authentifizierte Benutzer (Identitätsverbund). IAM
- Informationen zum Unterschied zwischen der Verwendung von Rollen und ressourcenbasierten Richtlinien für den kontenübergreifenden Zugriff finden Sie IAMim Benutzerhandbuch unter Kontoübergreifender Ressourcenzugriff. IAM

EMRServerless verwenden mit AWS Lake Formation für eine feinkörnige Zugriffskontrolle

Übersicht

Mit EMR Amazon-Versionen 7.2.0 und höher können Sie AWS Lake Formation um detaillierte Zugriffskontrollen auf Datenkatalogtabellen anzuwenden, die von S3 unterstützt werden. Mit dieser Funktion können Sie Zugriffskontrollen auf Tabellen-, Zeilen-, Spalten- und Zellenebene konfigurieren für read Abfragen innerhalb Ihrer Amazon EMR Serverless Spark-Jobs. Verwenden Sie Studio,

um eine detaillierte Zugriffskontrolle für Apache Spark-Batch-Jobs und interaktive Sitzungen zu konfigurieren. EMR In den folgenden Abschnitten erfahren Sie mehr über Lake Formation und dessen Verwendung mit EMR Serverless.

Verwenden von Amazon EMR Serverless mit AWS Lake Formation fallen zusätzliche Gebühren an. Weitere Informationen finden Sie unter EMRAmazon-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. EMRServerless verwendet Spark-Ressourcenprofile, um zwei Profile zur effektiven 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 Formationund Überlegungen und Einschränkungen.

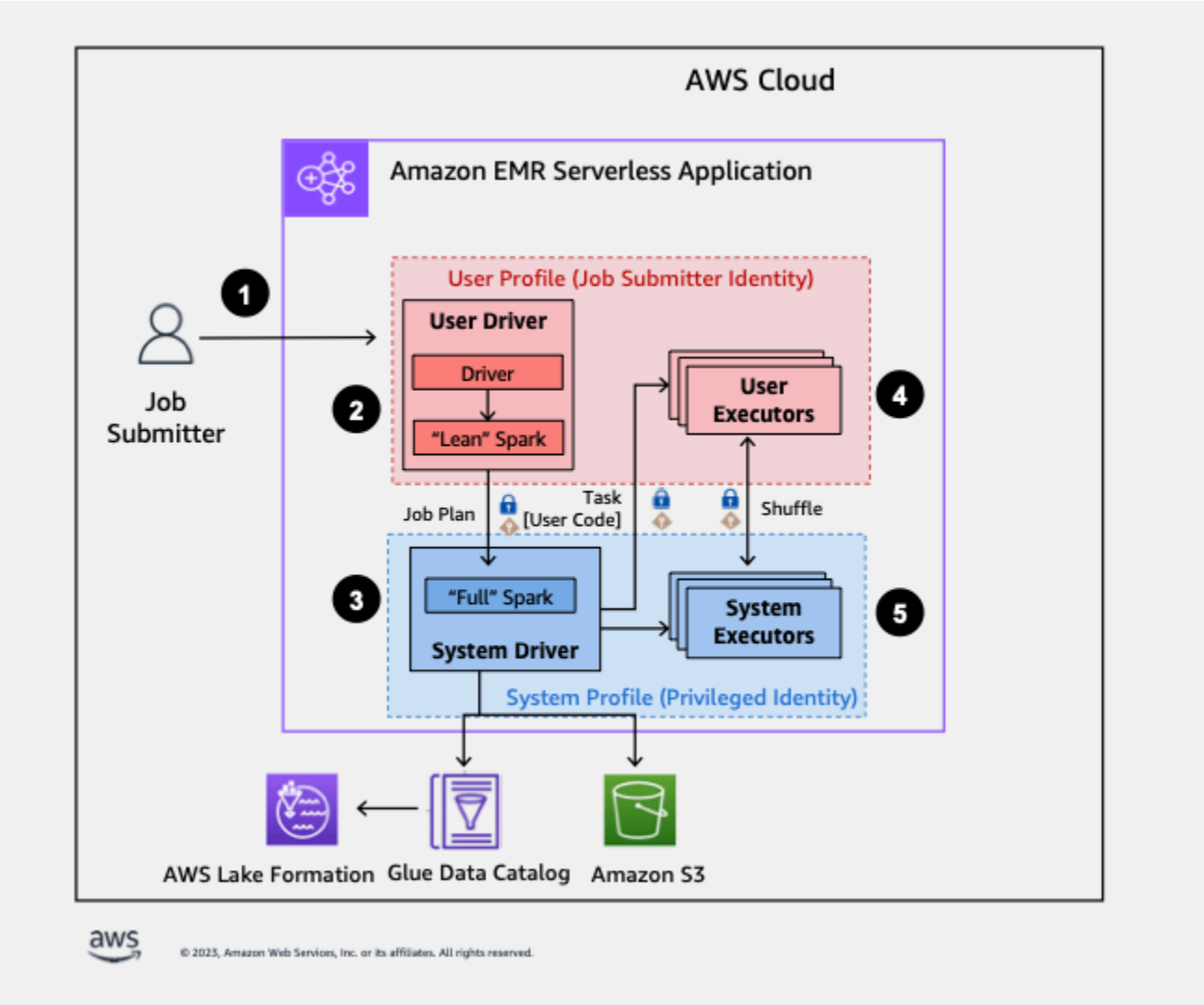
Wenn Sie vorinitialisierte Kapazität mit Lake Formation verwenden, empfehlen wir, mindestens zwei Spark-Treiber zu verwenden. Jeder Lake Formation-fähige Job verwendet zwei Spark-Treiber, einen für das Benutzerprofil und einen für das Systemprofil. Um die beste Leistung zu erzielen, sollten Sie für Lake Formation-fähige Jobs doppelt so viele Treiber verwenden wie für Jobs, die Lake Formation nicht verwenden.

Wenn Sie Spark-Jobs auf EMR Serverless ausführen, müssen 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 sowohl für Benutzer- als auch für 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 0 und 1 angeben`spark.dynamicAllocation.maxExecutorsRatio`. Darüber hinaus können Sie auch die folgenden Eigenschaften konfigurieren, 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 einen AWS Lake Formation-aktivierte EMR serverlose Anwendung.
2. EMRServerless 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. Es erstellt einen Jobplan.

3. EMRServerless richtet einen zweiten Treiber ein, den Systemtreiber, und führt ihn im Systemprofil aus (mit einer privilegierten Identität). EMRServerless richtet für die Kommunikation einen verschlüsselten TLS Kanal zwischen den beiden Treibern ein. Der Benutzertreiber verwendet den Kanal, um die Jobpläne an den Systemtreiber zu senden. Der Systemtreiber führt keinen vom Benutzer übermittelten Code aus. Er läuft in vollem Umfang mit Spark und kommuniziert mit S3 und dem Datenkatalog für den Datenzugriff. Es fordert Ausführende an und stellt den Jobplan in eine Abfolge von Ausführungsphasen zusammen.
4. EMRServerless führt die Phasen dann auf Executoren mit dem Benutzertreiber oder Systemtreiber aus. Benutzercode wird in jeder Phase ausschließlich auf Benutzerprofil-Executoren ausgeführt.
5. Stufen, die Daten aus Datenkatalogtabellen lesen, die geschützt sind durch AWS Lake Formation oder diejenigen, die Sicherheitsfilter anwenden, werden an Systemausführende delegiert.

Förderung der Lake Formation im Amazon EMR

Um Lake Formation zu aktivieren, müssen Sie beim Erstellen einer EMR serverlosen Anwendung den Wert `true` `spark-defaults` Unterklassifizierung für den Laufzeitkonfigurationsparameter festlegen`spark.emr-serverless.lakeformation.enabled`.

```
aws emr-serverless create-application \
    --release-label emr-7.2.0 \
    --runtime-configuration '{
     "classification": "spark-defaults",
     "properties": {
      "spark.emr-serverless.lakeformation.enabled": "true"
      }
    }' \
    --type "SPARK"
```

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
```

IAMRollenberechtigungen zur Laufzeit von Job

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. IAMBerechtigungen kontrollieren den Zugang zur Lake Formation und AWS Glue APIs und Ressourcen. Obwohl Sie möglicherweise über die Lake Formation Formation-Berechtigung verfügen, auf eine Tabelle im Datenkatalog (SELECT) zuzugreifen, schlägt Ihr Vorgang fehl, wenn Sie nicht über die IAM entsprechende Berechtigung für den `glue:Get*` API Vorgang verfügen.

Im Folgenden finden Sie ein Beispiel für eine Richtlinie zur Bereitstellung von IAM Berechtigungen für den Zugriff auf ein Skript in S3, d. h. für das Hochladen von Protokollen nach S3. AWS APIGlue-Berechtigungen und die Erlaubnis, auf Lake Formation zuzugreifen.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ScriptAccess",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
            "arn:aws:s3:::*.DOC-EXAMPLE-BUCKET/scripts",
            "arn:aws:s3:::*.DOC-EXAMPLE-BUCKET/*" ]
        },
        {
         "Sid": "LoggingAccess",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
```

```
            "Resource": [
               "arn:aws:s3:::DOC-EXAMPLE-BUCKET/logs/*"
            ]
        },
        {
            "Sid": "GlueCatalogAccess",
            "Effect": "Allow",
            "Action": [
                 "glue:Get*",
                 "glue:Create*",
                 "glue:Update*"
            ],
            "Resource": ["*"]
        },
        {
            "Sid": "LakeFormationAccess",
            "Effect": "Allow",
            "Action": [
             "lakeformation:GetDataAccess"
             ],
            "Resource": ["*"]
        }
    ]
}
```

Lake Formation Formation-Berechtigungen für die Job-Runtime-Rolle einrichten

Registrieren Sie zunächst den Standort Ihres Hive-Tisches bei Lake Formation. Erstellen Sie dann Berechtigungen für Ihre Job-Runtime-Rolle in der gewünschten Tabelle. Weitere Informationen zu Lake Formation finden Sie unter Was ist AWS Lake Formation? in der AWS Lake Formation Leitfaden für Entwickler.

Nachdem Sie die Lake Formation Formation-Berechtigungen eingerichtet haben, können Sie Spark-Jobs auf Amazon EMR Serverless einreichen. Weitere Informationen zu Spark-Jobs finden Sie unter Spark-Beispiele.

Einen Joblauf einreichen

Nachdem Sie die Lake Formation Grants eingerichtet haben, können Sie Spark-Jobs auf EMR Serverless einreichen. Um Iceberg-Jobs auszuführen, müssen Sie die folgenden `spark-submit` Eigenschaften angeben.

```
--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
```

Unterstützung für das Open-Table-Format

Die EMR Amazon-Version 7.2.0 bietet Unterstützung für eine differenzierte Zugriffskontrolle auf Basis von Lake Formation. EMRServerless unterstützt die Tabellentypen Hive und Iceberg. In der folgenden Tabelle werden alle unterstützten Operationen beschrieben.

Operationen	Hive	Iceberg
DDLBefehle	Nur mit IAM Rollenber echtigungen	Nur mit IAM Rollenber echtigungen
Inkrementelle Abfragen	Nicht zutreffend	Vollständig unterstützt
Zeitreiseabfragen	Gilt nicht für dieses Tabellenf ormat	Vollständig unterstützt
Metadaten-Tabellen	Gilt nicht für dieses Tabellenf ormat	Wird unterstützt, aber bestimmte Tabellen sind ausgeblendet. Weitere Informationen finden Sie unter Überlegungen und Einschrän kungen.
`DML INSERT`	Nur mit IAM Berechtigungen	Nur mit IAM Berechtigungen

Operationen	Hive	Iceberg
DML UPDATE	Gilt nicht für dieses Tabellenformat	Nur mit IAM Berechtigungen
`DML DELETE`	Gilt nicht für dieses Tabellenformat	Nur mit IAM Berechtigungen
Lesevorgänge	Vollständig unterstützt	Vollständig unterstützt
Gespeicherte Prozeduren	Nicht zutreffend	Unterstützt mit den Ausnahmen von `register_table` und `migrate`. Weitere Informationen finden Sie unter Überlegungen und Einschränkungen.

Überlegungen und Einschränkungen

Beachten Sie die folgenden Überlegungen und Einschränkungen, wenn Sie Lake Formation mit EMR Serverless verwenden.

> **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 aus der vorinitialisierten Kapazität bereitgestellt, und die Anzahl der Systemtreiber entspricht der Anzahl der Benutzertreiber, die Sie angeben. 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 EMRserverlosen Regionen verfügbar, außer AWS GovCloud (US-Ost) und AWS GovCloud (US-West).

- Amazon EMR Serverless unterstützt eine differenzierte Zugriffskontrolle über Lake Formation nur für Apache Hive- und Apache Iceberg-Tabellen. Zu den Apache Hive-Formaten gehören Parquet, und XSv. ORC
- Lake Formation-fähige Anwendungen unterstützen die Verwendung von benutzerdefinierten EMR serverlosen Images nicht.
- `DynamicResourceAllocation`Für Jobs in Lake Formation kann man nicht abschalten.
- Sie können Lake Formation nur mit Spark-Jobs verwenden.
- EMRServerless mit Lake Formation unterstützt nur eine einzige Spark-Sitzung während eines Jobs.
- EMRServerless mit Lake Formation unterstützt nur kontenübergreifende Tabellenabfragen, die über Ressourcenlinks gemeinsam genutzt werden.
- Folgendes wird nicht unterstützt:
  - Belastbare verteilte Datensätze () RDD
  - Spark-Streaming
  - Schreiben Sie mit Lake Formation erteilten Berechtigungen
  - Zugriffskontrolle für verschachtelte Spalten
- EMRServerless blockiert Funktionen, die die vollständige Isolierung des Systemtreibers untergraben könnten, darunter die folgenden:
  - UDTsiveUDFs, H und jede benutzerdefinierte Funktion, die benutzerdefinierte Klassen beinhaltet
  - Benutzerdefinierte Datenquellen
  - Bereitstellung zusätzlicher Jars für Spark-Erweiterungen, Konnektoren oder Metastore
  - `ANALYZE TABLE` command
- Zur Durchsetzung von Zugriffskontrollen `EXPLAIN PLAN` und DDL Vorgängen wie der `DESCRIBE TABLE` Vermeidung der Offenlegung geschützter Informationen.
- EMRServerless schränkt den Zugriff auf Systemtreiber-Spark-Protokolle für Lake Formation-fähige Anwendungen ein. Da der Systemtreiber mit mehr Zugriffen 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, hat EMR Serverless den Zugriff auf Systemtreiberprotokolle deaktiviert. Wenden Sie sich zur Problembehebung an AWS Unterstützung.
- Wenn Sie einen Tabellenstandort bei Lake Formation registriert haben, durchläuft der Datenzugriffspfad unabhängig von der IAM Berechtigung für die EMR Serverless-Job-Runtime-Rolle die gespeicherten Anmeldeinformationen von Lake Formation. Wenn Sie die mit der

Tabellenposition registrierte Rolle falsch konfigurieren, schlagen übermittelte Jobs fehl, die die Rolle mit IAM S3-Berechtigungen für den Tabellenspeicherort verwenden.

- Beim Schreiben in eine Lake Formation-Tabelle werden IAM Berechtigungen und nicht die von Lake Formation erteilten Berechtigungen verwendet. Wenn Ihre Job-Runtime-Rolle über die erforderlichen S3-Berechtigungen verfügt, können Sie sie zum Ausführen von Schreibvorgängen verwenden.

Im Folgenden finden Sie Überlegungen und Einschränkungen 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`path, und `summaries` enthalten. Diese Einschränkung gilt nicht für Iceberg-Tabellen, die nicht in Lake Formation registriert sind.
- Tabellen, die Sie nicht in Lake Formation registrieren, unterstützen alle gespeicherten Iceberg-Prozeduren. Die `migrate` Prozeduren `register_table` und werden für keine Tabellen unterstützt.
- Wir empfehlen, Iceberg DataFrameWriter V2 statt V1 zu verwenden.

Fehlerbehebung

In den folgenden Abschnitten finden Sie Lösungen zur Fehlerbehebung.

Protokollierung

EMRServerless verwendet Spark-Ressourcenprofile, um die Auftragsausführung aufzuteilen. EMRServerless 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 als Benutzerprofil ausgeführt wurden.

Live-Benutzeroberfläche und Spark History Server

In der Live-Benutzeroberfläche und auf dem Spark History Server werden alle Spark-Ereignisse aus dem Benutzerprofil und aus dem Systemtreiber redigierte Ereignisse generiert.

Sie können alle Aufgaben sowohl der Benutzer- als auch der Systemtreiber auf der Registerkarte Executors sehen. Protokolllinks sind jedoch nur für das Benutzerprofil verfügbar. Außerdem werden einige Informationen aus der Live-Benutzeroberfläche entfernt, z. B. die Anzahl der Ausgabedatensätze.

Job mit unzureichenden Lake Formation Formation-Berechtigungen fehlgeschlagen

Stellen Sie sicher, dass Ihre Job-Runtime-Rolle über die erforderlichen Berechtigungen für die Ausführung SELECT und DESCRIBE für die Tabelle verfügt, auf die Sie zugreifen.

Job mit RDD Ausführung ist fehlgeschlagen

EMRServerless unterstützt derzeit keine belastbaren verteilten Dataset (RDD) -Operationen für Lake Formation-fähige Jobs.

Auf Datendateien in Amazon S3 kann nicht zugegriffen werden

Stellen Sie sicher, dass Sie den Standort des Data Lake in Lake Formation registriert haben.

Ausnahme bei der Sicherheitsüberprüfung

EMRServerless hat einen Fehler bei der Sicherheitsüberprüfung festgestellt. Kontakt AWS Unterstützung für Unterstützung.

Freigabe AWS Datenkatalog und Tabellen kontenübergreifend zusammenfügen

Sie können Datenbanken und Tabellen für mehrere Konten gemeinsam nutzen und trotzdem Lake Formation verwenden. Weitere Informationen finden Sie unter Kontoübergreifender Datenaustausch in Lake Formation und Wie teile ich AWS Glue Data Catalog und Tabellen kontenübergreifend mit AWS Lake Formation?.

Verschlüsselung zwischen Mitarbeitern

Mit den EMR Amazon-Versionen 6.15.0 und höher können Sie die wechselseitige TLS verschlüsselte Kommunikation zwischen Mitarbeitern 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 überprüfen anhand der konfigurierten Zertifikate die Identität der anderen.

Wenn ein Mitarbeiter 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 gegenseitige TLS Verschlüsselung standardmäßig aktiviert.

Aktivierung der gegenseitigen TLS Verschlüsselung auf Serverless EMR

Um die gegenseitige TLS Verschlüsselung für Ihre Spark-Anwendung `spark.ssl.internode.enabled` zu aktivieren, setzen Sie diesen Wert bei der Erstellung einer EMR serverlosen Anwendung auf true. Wenn Sie das verwenden AWS Konsole, um eine EMR serverlose Anwendung zu erstellen, wählen Sie Benutzerdefinierte Einstellungen verwenden, erweitern Sie dann Anwendungskonfiguration und geben Sie Ihre `runtimeConfiguration` ein.

```
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 `spark-submit` auf true.

```
--conf spark.ssl.internode.enabled=true
```

Secrets Manager für Datenschutz mit EMR Serverless

AWS Secrets Manager ist ein geheimer Speicherdienst, mit dem Sie Datenbankanmeldeinformationen, API Schlüssel und andere geheime Informationen schützen können. Dann können Sie in Ihrem Code hartcodierte Anmeldeinformationen durch einen API Aufruf von Secrets Manager ersetzen. Auf diese Weise wird sichergestellt, dass das Geheimnis nicht von jemandem missbraucht werden kann, der Ihren Code untersucht, da das Geheimnis nicht vorhanden ist. Einen Überblick finden Sie im AWS Secrets Manager Benutzerleitfaden.

Secrets Manager verschlüsselt Geheimnisse mit AWS Key Management Service Schlüssel. Weitere Informationen finden Sie unter Geheime Verschlüsselung und Entschlüsselung im AWS Secrets Manager Benutzerleitfaden.

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 Rotieren AWS Secrets Manager Geheimnisse in der AWS Secrets Manager Benutzerleitfaden.

Amazon EMR Serverless lässt sich integrieren mit AWS Secrets Manager damit Sie Ihre Daten in Secrets Manager speichern und die geheime ID in Ihren Konfigurationen verwenden können.

Wie EMR Serverless Secrets verwendet

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 externen Benutzern 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 Erstellen eines AWS Secrets Manager Geheimnis in der AWS Secrets Manager Benutzerleitfaden. Wählen Sie in Schritt 3 das Klartext-Feld aus, um Ihren sensiblen Wert einzugeben.

Geben Sie ein Geheimnis in einer Konfigurationsklassifizierung ein

Die folgenden Beispiele zeigen, wie ein Geheimnis in einer Konfigurationsklassifizierung unter `StartJobRun` angegeben wird. Wenn Sie Klassifizierungen für Secrets Manager auf Anwendungsebene konfigurieren möchten, finden Sie weitere Informationen unterStandardanwendungskonfiguration für Serverless EMR.

Ersetzen Sie es in den Beispielen *SecretName* durch den Namen des abzurufenden Geheimnisses. Fügen Sie den Bindestrich ein, gefolgt von den sechs Zeichen, die Secrets Manager am Ende des Geheimnisses ARN hinzufügt. Weitere Informationen finden Sie unter Wie erstellt man ein Geheimnis.

In diesem Abschnitt

- Geben Sie geheime Verweise an — Spark
- Geben Sie geheime Referenzen an — Hive

Geben Sie geheime Verweise 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://DOC-EXAMPLE-BUCKET/scripts/spark-jdbc.py",
            "sparkSubmitParameters": "--jars s3://DOC-EXAMPLE-BUCKET/mariadb-connector-
java.jar
            --conf
 spark.hadoop.javax.jdo.option.ConnectionDriverName=org.mariadb.jdbc.Driver
            --conf spark.hadoop.javax.jdo.option.ConnectionUserName=connection-user-
name
            --conf
 spark.hadoop.javax.jdo.option.ConnectionPassword=EMR.secret@SecretName
            --conf spark.hadoop.javax.jdo.option.ConnectionURL=jdbc:mysql://db-host:db-
port/db-name
            --conf spark.driver.cores=2
            --conf spark.executor.memory=10G
            --conf spark.driver.memory=6G
            --conf spark.executor.cores=4"
        }
    }' \
    --configuration-overrides '{
        "monitoringConfiguration": {
        "s3MonitoringConfiguration": {
            "logUri": "s3://DOC-EXAMPLE-BUCKET/spark/logs/"
        }
```

```
    }
}'
```

Example — 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://DOC-EXAMPLE-BUCKET/emr-serverless-hive/query/hive-query.ql",
        "parameters": "--hiveconf hive.exec.scratchdir=s3://DOC-EXAMPLE-BUCKET/emr-
serverless-hive/hive/scratch
                    --hiveconf hive.metastore.warehouse.dir=s3://DOC-EXAMPLE-BUCKET/
emr-serverless-hive/hive/warehouse
                    --hiveconf javax.jdo.option.ConnectionUserName=username
                    --hiveconf
 javax.jdo.option.ConnectionPassword=EMR.secret@SecretName
                    --hiveconf
 hive.metastore.client.factory.class=org.apache.hadoop.hive.ql.metadata.SessionHiveMetaStoreCli
                    --hiveconf
 javax.jdo.option.ConnectionDriverName=org.mariadb.jdbc.Driver
```

```
                    --hiveconf javax.jdo.option.ConnectionURL=jdbc:mysql://db-host:db-
port/db-name"
        }
    }' \
    --configuration-overrides '{
        "monitoringConfiguration": {
        "s3MonitoringConfiguration": {
            "logUri": "s3://EXAMPLE-LOG-BUCKET"
        }
    }
}'
```

Example — 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 unter Berechtigungen für den KMS Schlüssel in der AWS Secrets Manager Benutzerleitfaden.

Ersetzen Sie es 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 Geheimnis 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-Anwendungs-ID als Bedingung in der Richtlinie angeben.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue",
        "secretsmanager:DescribeSecret"
      ],
      "Principal": {
        "Service": [
          "emr-serverless.amazonaws.com"
        ]
      },
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:SourceArn": "arn:aws:emr-serverless:AWS-Region:aws_account_id:/
applications/applicationId"
        }
      }
    }
  ]
}
```

Erstellen Sie Ihr Geheimnis mit der folgenden Richtlinie für den vom Kunden verwalteten AWS Key Management Service (AWS KMS) Schlüssel:

Richtlinie für vom Kunden verwaltete AWS KMS Schlüssel

```
{
    "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 drehen

Rotation bedeutet, dass Sie ein Geheimnis regelmäßig aktualisieren. Sie können konfigurieren AWS Secrets Manager um das Secret nach einem von Ihnen festgelegten Zeitplan automatisch für Sie zu rotieren. Auf diese Weise können Sie langfristige Geheimnisse durch kurzfristige ersetzen. Dies trägt dazu bei, das Risiko von Kompromissen zu verringern. EMRServerless 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 EMR Amazon-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 Serverless aus erweitern können. EMR Wenn Sie für Ihre S3-Daten eine komplexe oder umfangreiche Berechtigungskonfiguration haben, können Sie mit S3 Access Grants die S3-Datenberechtigungen für Benutzer, Gruppen, Rollen und Anwendungen skalieren.

Verwenden Sie S3 Access Grants, um den Zugriff auf Amazon S3 S3-Daten über die von der Runtime-Rolle gewährten Berechtigungen oder die IAM Rollen hinaus zu erweitern, die den Identitäten mit Zugriff auf Ihre EMR serverlose Anwendung zugewiesen sind.

Weitere Informationen finden Sie unter Verwaltung des Zugriffs mit S3 Access Grants for Amazon EMR im Amazon EMR Management Guide und Verwaltung des Zugriffs mit S3 Access Grants im Amazon Simple Storage Service User Guide.

In diesem Abschnitt wird beschrieben, wie Sie eine EMR serverlose 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 EMR Amazon-Bereitstellungen finden Sie in der folgenden Dokumentation:

- Verwenden von S3 Access Grants mit Amazon EMR
- Verwenden von S3 Access Grants mit Amazon EMR auf EKS

Starten Sie eine EMR serverlose 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 Rolle zur Auftragsausführung für Ihre EMR serverlose Anwendung ein. Geben Sie die erforderlichen IAM Berechtigungen an, die Sie für die Ausführung von Spark-Jobs und die Verwendung von S3 Access Grants benötigen, `s3:GetDataAccess` und`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 auch dann auf die durch die Rolle zugelassenen Daten zugreifen, wenn sie keine Genehmigung von S3 Access Grants haben.

2. Starten Sie Ihre EMR serverlose Anwendung mit einem EMR Amazon-Release-Label von 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://DOC-EXAMPLE-BUCKET-OUTPUT/
wordcount_output"],
            "sparkSubmitParameters": "--conf spark.executor.cores=1 --conf
 spark.executor.memory=4g --conf spark.driver.cores=1 --conf spark.driver.memory=4g
 --conf spark.executor.instances=1"
        }
    }' \
  --configuration-overrides '{
    "applicationConfiguration": [{
        "classification": "spark-defaults",
        "properties": {
          "spark.hadoop.fs.s3.s3AccessGrants.enabled": "true",
          "spark.hadoop.fs.s3.s3AccessGrants.fallbackToIAM": "false"
         }
      }]
}'
```

Überlegungen zu S3 Access Grants mit Serverless EMR

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 API Serverless-Anrufen mit AWS CloudTrail

Amazon EMR Serverless ist integriert in AWS CloudTrail, ein Service, der eine Aufzeichnung der Aktionen bereitstellt, die von einem Benutzer, einer Rolle oder einem AWS Dienst in EMR Serverless. CloudTrail erfasst alle API Aufrufe für EMR Serverless als Ereignisse. Zu den erfassten Aufrufen gehören Aufrufe von der EMR Serverless-Konsole und Code-Aufrufe für die EMR API serverlosen Operationen. Wenn Sie einen Trail erstellen, können Sie die kontinuierliche Übermittlung von CloudTrail Ereignissen an einen Amazon S3 S3-Bucket aktivieren, einschließlich Ereignissen für EMR Serverless. Wenn Sie keinen Trail konfigurieren, können Sie die neuesten Ereignisse trotzdem in der CloudTrail Konsole im Ereignisverlauf anzeigen. 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 dazu finden CloudTrail Sie im AWS CloudTrail Benutzerleitfaden.

EMRServerlose Informationen in CloudTrail

CloudTrail ist auf Ihrem aktiviert AWS-Konto wenn Sie das Konto erstellen. Wenn eine Aktivität in EMR Serverless auftritt, wird diese Aktivität zusammen mit anderen Aktivitäten in einem CloudTrail Ereignis aufgezeichnet AWS Serviceereignisse im Ereignisverlauf. Sie können aktuelle Ereignisse in Ihrem ansehen, suchen und herunterladen AWS-Konto. Weitere Informationen finden Sie unter Ereignisse mit dem CloudTrail Ereignisverlauf anzeigen.

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 erstellen, gilt der Trail standardmäßig für alle AWS-Regionen. Der Trail protokolliert Ereignisse aus allen Regionen der AWS partitioniert und übermittelt die Protokolldateien an den Amazon S3 S3-Bucket, den Sie angeben. Darüber hinaus können Sie weitere konfigurieren AWS Dienste zur weiteren Analyse der

in CloudTrail Protokollen gesammelten Ereignisdaten und zur weiteren Bearbeitung dieser Daten. Weitere Informationen finden Sie hier:

- Übersicht zum Erstellen eines Trails
- CloudTrail unterstützte Dienste und Integrationen
- Konfiguration von SNS Amazon-Benachrichtigungen für CloudTrail
- Empfangen von CloudTrail Protokolldateien aus mehreren Regionen und Empfangen von CloudTrail Protokolldateien von mehreren Konten

Alle EMR serverlosen Aktionen werden von der EMRServerless Reference protokolliert CloudTrail und sind in dieser API dokumentiert. Beispielsweise generieren Aufrufe von `StartJobRun` und `CancelJobRun` Aktionen Einträge in den CloudTrail Protokolldateien. `CreateApplication`

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 gestellt wurde oder AWS Identity and Access Management (IAM) Benutzeranmeldedaten.
- Gibt an, ob die Anforderung mit temporären Sicherheitsanmeldeinformationen für eine Rolle oder einen Verbundbenutzer gesendet wurde.
- Ob die Anfrage von einem anderen gestellt wurde AWS Dienst.

Weitere Informationen finden Sie im CloudTrail userIdentity Element.

Grundlegendes zu EMR serverlosen 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"
}
```

Konformitätsvalidierung für Amazon EMR Serverless

Die Sicherheit und Konformität von EMR Serverless wird von unabhängigen Prüfern im Rahmen mehrerer Prüfungen bewertet AWS Compliance-Programme, einschließlich der folgenden:

- System- und Organisationskontrollen (SOC)
- Datensicherheitsstandard der Zahlungskartenbranche (PCIDSS)
- Das Risiko- und Autorisierungsmanagementprogramm des Bundes (FedRAMP) ist moderat
- Gesetz über die Portabilität und Rechenschaftspflicht von Krankenversicherungen () HIPAA

AWS bietet eine häufig aktualisierte Liste von AWS Dienstleistungen im Rahmen spezifischer Compliance-Programme unter AWS Dienstleistungen im Geltungsbereich des Compliance-Programms.

Prüfberichte von Drittanbietern stehen Ihnen zum Herunterladen zur Verfügung unter AWS Artifact. Weitere Informationen finden Sie unter Berichte herunterladen in AWS Artifact.

Weitere Informationen zur AWS Compliance-Programme, siehe AWS Compliance-Programme.

Ihre Verantwortung für die Einhaltung von Vorschriften bei der Verwendung von EMR Serverless hängt von der Vertraulichkeit Ihrer Daten, den Compliance-Zielen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften ab. Wenn Ihre Nutzung von EMR Serverless der Einhaltung von Standards wieHIPAA, oder Fed RAMP Moderate unterliegtPCI, AWS stellt Ressourcen zur Verfügung, um Ihnen zu helfen:

- Schnellstartanleitungen zu Sicherheit und Compliance, in denen architektonische Überlegungen und Schritte zur Implementierung von sicherheits- und Compliance-orientierten Basisumgebungen erläutert werden AWS.
- AWS Mithilfe der Kundenleitfäden zur Einhaltung gesetzlicher Vorschriften können Sie das Modell der geteilten Verantwortung aus dem Blickwinkel der Einhaltung von Vorschriften besser verstehen. In den Leitfäden werden die bewährten Methoden zur Sicherung zusammengefasst AWS-Services und geben einen Überblick über die Leitlinien für Sicherheitskontrollen in verschiedenen Regelwerken (darunter National Institute of Standards and Technology (NIST), Payment Card

Industry Security Standards Council (PCI) und International Organization for Standardization (ISO)).

- 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 innerhalb AWS und hilft Ihnen dabei, Ihre Einhaltung der Sicherheitsstandards und Best Practices der Branche zu überprüfen.
- AWS Audit Manager— das AWS-Service hilft Ihnen dabei, Ihre kontinuierlich zu überprüfen AWS Nutzung, um das Risikomanagement und die Einhaltung von Vorschriften und Industriestandards zu vereinfachen.

Resilienz bei Amazon EMR Serverless

Das Tool AWS Die globale Infrastruktur basiert auf AWS Regionen und Verfügbarkeitszonen. AWS -Regionen stellen mehrere physisch getrennte und isolierte Availability Zones bereit, die über hoch redundante Netzwerke mit niedriger Latenz und hohen Durchsätzen verbunden sind. Mithilfe von Availability Zones können Sie Anwendungen und Datenbanken erstellen und ausführen, die automatisch Failover zwischen Zonen ausführen, ohne dass es zu Unterbrechungen kommt. Availability Zones sind besser verfügbar, fehlertoleranter und skalierbarer als herkömmliche Infrastrukturen mit einem oder mehreren Rechenzentren.

Weitere Informationen zur AWS Regionen und Verfügbarkeitszonen finden Sie unter AWS Globale Infrastruktur.

Zusätzlich zu AWS Amazon EMR Serverless bietet eine globale Infrastruktur und die Integration mit Amazon S3EMRFS, um Ihre Anforderungen an Datenstabilität und Datensicherung zu erfüllen.

Infrastruktursicherheit in Amazon EMR Serverless

Als verwalteter Service EMR ist Amazon geschützt durch AWS globale Netzwerksicherheit. Für Informationen über AWS Sicherheitsdienste und wie AWS schützt die Infrastruktur, siehe AWS Cloud-Sicherheit. Um Ihre zu entwerfen AWS Umgebung, in der die besten Methoden für die Infrastruktursicherheit verwendet werden, finden Sie unter Infrastructure Protection in Security Pillar AWS Gut durchdachtes Framework.

Du verwendest AWS veröffentlichte API Aufrufe zum Zugriff auf Amazon EMR über das Netzwerk. Kunden müssen Folgendes unterstützen:

- Sicherheit auf Transportschicht (TLS). Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Cipher-Suites mit perfekter Vorwärtsgeheimhaltung (PFS) wie (Ephemeral Diffie-Hellman) oder DHE (Elliptic Curve Ephemeral Diffie-Hellman). ECDHE Die meisten modernen Systeme wie Java 7 und höher unterstützen diese Modi.

Darüber hinaus müssen Anfragen mithilfe einer Zugriffsschlüssel-ID und eines geheimen Zugriffsschlüssels, der einem Prinzipal zugeordnet ist, signiert werden. IAM Oder Sie können das verwenden AWS Security Token Service (AWS STS), um temporäre Sicherheitsanmeldedaten zum Signieren von Anfragen zu generieren.

Konfiguration und Schwachstellenanalyse in Amazon EMR Serverless

AWS kümmert sich um 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:

- Konformitätsvalidierung für Amazon EMR Serverless
- Modell der übergreifenden Verantwortlichkeit
- Amazon Web Services: Übersicht über Sicherheitsverfahren (Whitepaper)

Endpunkte und Kontingente für EMR Serverless

Service-Endpunkte

Um eine programmgesteuerte Verbindung zu einem herzustellen AWS-Service, Sie verwenden einen Endpunkt. Ein Endpunkt ist URL der Einstiegspunkt für ein AWS Webdienst. Zusätzlich zum Standard AWS Endpunkte, einige AWS-Services bieten FIPS Endpunkte in ausgewählten Regionen an. In der folgenden Tabelle sind die Dienstendpunkte für EMR Serverless aufgeführt. Weitere Informationen finden Sie unter AWS-Service Endpunkte.

Name der Region	Region	Endpunkt	Protokoll
USA Ost (Ohio)	`us-east-2` (beschränkt auf die folgenden Availability Zones: `use2-az1use2-az2`, und`use2-az3`)	`emr-serverless.us-east-2.amazonaws.com`	HTTPS

USA Ost (Nord-Virginia)	`us-east-1` (beschränkt auf die folgenden Availability Zones: `use1-az1use1-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	Protokoll
		west-2.amazonaws.com emr-serverless-fips.us-west-2.amazonaws.com	
Africa (Cape Town)	af-south-1	emr-serverless.af-south-1.amazonaws.com	HTTPS
Asia Pacific (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 (Mumbai)	ap-south-1	emr-serverless.ap-south-1.amazonaws.com	HTTPS
Asia Pacific (Osaka)	ap-northeast-3	emr-serverless.ap-northeast-3.amazonaws.com	HTTPS

Name der Region	Region	Endpunkt	Protokoll
Asia Pacific (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
Canada (Central)	ca-central-1 (beschränkt auf die folgenden Availability Zones: cac1-az1 undcac1-az2)	emr-serverless.ca-central-1.amazonaws.com	HTTPS
Europe (Frankfurt)	eu-central-1	emr-serverless.eu-central-1.amazonaws.com	HTTPS

Name der Region	Region	Endpunkt	Protokoll
Europa (Irland)	eu-west-1	emr-serverless.eu-west-1.amazonaws.com	HTTPS
Europa (London)	eu-west-2	emr-serverless.eu-west-2.amazonaws.com	HTTPS
Europa (Milan)	eu-south-1	emr-serverless.eu-south-1.amazonaws.com	HTTPS
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
Europe (Stockholm)	eu-north-1	emr-serverless.eu-north-1.amazonaws.com	HTTPS
Middle East (Bahrain)	me-south-1	emr-serverless.me-south-1.amazonaws.com	HTTPS

Name der Region	Region	Endpunkt	Protokoll
Naher Osten (UAE)	`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
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

Servicekontingenten, auch Limits genannt, sind die maximale Anzahl von Serviceressourcen oder Vorgängen, die Ihr AWS-Konto verwenden kann. EMRServerless sammelt jede Minute Messdaten zur Nutzung von Dienstkontingenten und veröffentlicht sie im `AWS/Usage` Namespace.

> **Note**
>
> Neu AWS Konten haben möglicherweise anfänglich niedrigere Kontingente, die sich im Laufe der Zeit erhöhen können. Amazon EMR Serverless überwacht die Kontonutzung in jedem AWS-Region und erhöht dann automatisch die Kontingente auf der Grundlage Ihrer Nutzung.

In der folgenden Tabelle sind die Dienstkontingente für EMR Serverless aufgeführt. Weitere Informationen finden Sie unter AWS-Service Kontingente.

Name	Standardlimit	Anpassbar?	Beschreibung
Max. Anzahl gleichzeitiger Zugriffe vCPUs pro Konto	16	Ja	Die maximale Anzahl davon vCPUs , die gleichzeitig für das Konto ausgeführt werden können AWS-Region.

APIGrenzwerte

Im Folgenden werden die API Grenzwerte pro Region für Ihre AWS-Konto.

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.
ListJobRuns	1 Transaktion pro Sekunde. Burst von 25 Transaktionen pro Sekunde.
StartJobRun	1 Transaktion pro Sekunde. Burst von 25 Transaktionen pro Sekunde.

Ressource	Standardkontingent
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 Serverless. EMR

- EMRServerless ist in den folgenden Bereichen 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)
  - Asia Pacific (Osaka)
  - Asia Pacific (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 () UAE
  - 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 im `startJobRun` API oder im ändern AWS SDK. 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, können Sie den Wert `executionTimeoutMinutes` auf 0 setzen, damit der Streaming-Job kontinuierlich ausgeführt werden kann.
- Die `billedResourceUtilization` Eigenschaft in der `getJobRun` API zeigt das Aggregat vCPU, den Arbeitsspeicher und den Speicher an AWS hat die Ausführung des Jobs in Rechnung gestellt. 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 zugreifen AWS-Service Endpunkte im selben AWS-Region. Zu diesen Diensten gehören Amazon S3 S3, AWS Glue, Amazon CloudWatch Logs, AWS KMS, AWS Security Token Service, Amazon DynamoDB, und AWS Secrets Manager. Sie können VPC Konnektivität aktivieren, um auf andere zuzugreifen AWS-Services durch AWS PrivateLink, aber Sie müssen das nicht tun. Um auf externe Dienste zuzugreifen, können Sie Ihre Anwendung mit einem erstellenVPC.
- EMRServerless unterstützt HDFS nicht. Die lokalen Festplatten auf Workern sind temporäre Speicher, die EMR Serverless verwendet, um Daten während der Auftragsausführung zu mischen und zu verarbeiten.
- EMRServerless unterstützt das Bestehende nicht. emr-dynamodb-connector

Release-Versionen von Amazon EMR Serverless

Eine EMR Amazon-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.

Mit Amazon EMR 6.6.0 und höher können Sie EMR Serverless bereitstellen. Diese Bereitstellungsoption ist in früheren EMR Amazon-Release-Versionen nicht verfügbar. Wenn Sie Ihren Job einreichen, müssen Sie eine der folgenden unterstützten Versionen angeben.

Themen

EMR Serverless 7.2.0

In der folgenden Tabelle sind die Anwendungsversionen aufgeführt, die mit verfügbar sind EMR Serverless 7.2.0.

Anwendung	Version
Apache Spark	3.5.1
Apache Hive	3.1.3
Apache Tez	0.10.2

EMRVersionshinweise zu Serverless 7.2.0

- Lake Formation mit EMR Serverless — das können Sie jetzt verwenden AWS Lake Formation um detaillierte Zugriffskontrollen auf Datenkatalogtabellen anzuwenden, die von S3 unterstützt werden. Mit dieser Funktion können Sie Zugriffskontrollen auf Tabellen-, Zeilen-, Spalten- und Zellenebene für Leseabfragen innerhalb Ihrer EMR serverlosen Spark-Jobs konfigurieren. Weitere Informationen erhalten Sie unter the section called “Lake Formation für FGAC” und the section called “Überlegungen”.

EMR Serverless 7.1.0

In der folgenden Tabelle sind die Anwendungsversionen aufgeführt, die mit verfügbar sind EMR Serverless 7.1.0.

Anwendung	Version
Apache Spark	3.5.0
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 7.0.0

In der folgenden Tabelle sind die Anwendungsversionen aufgeführt, die mit verfügbar sind EMR Serverless 7.0.0.

Anwendung	Version
Apache Spark	3.5.0
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 6.15,0

In der folgenden Tabelle sind die Anwendungsversionen aufgeführt, die mit verfügbar sind EMR Serverless 6.15.0.

Anwendung	Version
Apache Spark	3.4.1
Apache Hive	3.1.3
Apache Tez	0.10.2

EMRVersionshinweise zu Serverless 6.15.0

- TLSSupport — Mit den Amazon EMR Serverless-Versionen 6.15.0 und höher können Sie die wechselseitige TLS verschlüsselte Kommunikation zwischen Mitarbeitern in Ihren Spark-Jobläufen ermöglichen. Wenn diese Option aktiviert ist, generiert EMR Serverless automatisch ein eindeutiges Zertifikat für jeden Worker, das es im Rahmen von Jobläufen bereitstellt. Dieses Zertifikat verwenden die Mitarbeiter während des TLS Handshakes, um sich gegenseitig zu authentifizieren und einen verschlüsselten Kanal für die sichere Verarbeitung von Daten einzurichten. Weitere Informationen zur gegenseitigen Verschlüsselung finden Sie unter TLS Verschlüsselung zwischen Mitarbeitern.

EMR Serverless 6.14.0

In der folgenden Tabelle sind die Anwendungsversionen aufgeführt, die mit verfügbar sind EMR Serverless 6.14.0.

Anwendung	Version
Apache Spark	3.4.1
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 6.13,0

In der folgenden Tabelle sind die Anwendungsversionen aufgeführt, die mit verfügbar sind EMR Serverless 6.13.0.

Anwendung	Version
Apache Spark	3.4.1
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 6.12.0

In der folgenden Tabelle sind die Anwendungsversionen aufgeführt, die mit verfügbar sind EMR Serverless 6.12.0.

Anwendung	Version
Apache Spark	3.4.0
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 6.11.0

In der folgenden Tabelle sind die Anwendungsversionen aufgeführt, die mit verfügbar sind EMR Serverless 6.11.0.

Anwendung	Version
Apache Spark	3.3.2
Apache Hive	3.1.3
Apache Tez	0.10.2

EMRVersionshinweise zu Serverless 6.11.0

- 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 Konten übernommen werden AWS Konten von Serverless aus. EMR

EMR Serverless 6.10.0

In der folgenden Tabelle sind die Anwendungsversionen aufgeführt, die mit verfügbar sind EMR Serverless 6.10.0.

Anwendung	Version
Apache Spark	3.3.1
Apache Hive	3.1.3
Apache Tez	0.10.2

EMRVersionshinweise zu Serverless 6.10.0

- Für EMR serverlose Anwendungen mit Version 6.10.0 oder höher ist der Standardwert für die Eigenschaft. `spark.dynamicAllocation.maxExecutors` `infinity` Frühere Versionen sind standardmäßig auf. `100` Weitere Informationen finden Sie unter Eigenschaften von Spark-Jobs.

EMR Serverless 6.9.0

In der folgenden Tabelle sind die Anwendungsversionen aufgeführt, die mit verfügbar sind EMR Serverless 6.9.0.

Anwendung	Version
Apache Spark	3.3.0
Apache Hive	3.1.3
Apache Tez	0.10.2

EMRVersionshinweise zu Serverless 6.9.0

- Die Amazon Redshift Redshift-Integration für Apache Spark ist in EMR Amazon-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 Serverless EMR.
- EMRDie serverlose Version 6.9.0 bietet Unterstützung für AWS Graviton2-Architektur (arm64). Sie können den `architecture` Parameter für `create-application` und verwenden, `update-application` APIs um die arm64-Architektur auszuwählen. Weitere Informationen finden Sie unter Optionen für die EMR serverlose Architektur von Amazon.
- Sie können jetzt Amazon DynamoDB-Tabellen direkt aus Ihren EMR serverlosen Spark- und Hive-Anwendungen exportieren, importieren, abfragen und verbinden. Weitere Informationen finden Sie unter Mit Amazon Serverless eine Verbindung zu DynamoDB herstellen EMR.

Bekannte Probleme

- Wenn Sie die Amazon-Redshift-Integration für Apache Spark verwenden und eine Zeit, timetz, timestamp oder timestamptz mit Mikrosekundengenauigkeit im Parquet-Format haben, rundet der Konnektor die Zeitwerte auf den nächstliegenden Millisekundenwert. Um das Problem zu umgehen, verwenden Sie den `unload_s3_format`-Formatparameter-Text-Unload.

EMR Serverless 6.8.0

In der folgenden Tabelle sind die Anwendungsversionen aufgeführt, die mit verfügbar sind EMR Serverless 6.8.0.

Anwendung	Version
Apache Spark	3.3.0
Apache Hive	3.1.3
Apache Tez	0.9.2

EMR Serverless 6.7.0

In der folgenden Tabelle sind die Anwendungsversionen aufgeführt, die mit verfügbar sind EMR Serverless 6.7.0.

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.

Änderung	Beschreibung
Funktion	Der Tez-Scheduler unterstützt jetzt das Preemptive von Tez-Aufgaben anstelle des Preemptivs von Containern

EMR Serverless 6.6.0

In der folgenden Tabelle sind die Anwendungsversionen aufgeführt, die mit verfügbar sind EMR Serverless 6.6.0.

Anwendung	Version
Apache Spark	3.2.0
Apache Hive	3.1.2
Apache Tez	0.9.2

EMRErste Versionshinweise für Serverless

- EMRServerless unterstützt die Spark-Konfigurationsklassifizierung. `spark-defaults` Diese Klassifizierung ändert Werte in der `spark-defaults.conf` XML Spark-Datei. Mithilfe von Konfigurationsklassifizierungen können Sie Anwendungen anpassen. Weitere Informationen finden Sie unter Konfigurieren von Anwendungen.
- EMRServerless unterstützt die Hive-Konfigurationsklassifizierungen`hive-site`,`tez-site`, `emrfs-site` und. `core-site` Diese Klassifizierung kann die Werte in der `hive-site.xml` Datei von Hive, Tez, den EMRFS Einstellungen EMR von Amazon bzw. der `core-site.xml` Hadoop-Datei ändern. `tez-site.xml` Mithilfe von Konfigurationsklassifizierungen können Sie Anwendungen anpassen. Weitere Informationen finden Sie unter Konfigurieren von Anwendungen.

Engine-spezifische Änderungen, Verbesserungen und behobene Probleme

- In der folgenden Tabelle sind die Backports von Hive und Tez aufgeführt.

  Änderungen an Hive und Tez

Änderung	Beschreibung
Backport	TEZ-4430: Ein Problem mit der Immobilie wurde behoben `tez.task.launch.cmd-opts`

Änderung	Beschreibung
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. Für weitere Informationen zu Aktualisierungen dieser Dokumentation können Sie einen RSS Feed abonnieren.

Änderung	Beschreibung	Datum
Neue Version	EMR Serverless 7.2.0	25. Juli 2024
Neue Veröffentlichung	EMR Serverless 7.1.0	17. April 2024
Aktualisierung einer bestehenden Richtlinie.	Die neue `Sid` `CloudWatchPolicyStatement` und wurde `EC2PolicyStatement` der mazonEMRServerless ServiceRolePolicy A-Richtlinie hinzugefügt.	25. Januar 2024
Neue Veröffentlichung	EMR Serverless 7.0.0	29. Dezember 2023
Neue Veröffentlichung	EMR Serverless 6.15,0	17. November 2023
Neues Feature	Konfigurieren Sie mehrere IAM Rollen, die übernommen werden sollen, wenn Sie von EMR Serverless aus auf Amazon S3 S3-Buckets in verschiedenen Konten zugreifen (6.11 und höher)	18. Oktober 2023
Neue Version	EMR Serverless 6.14.0	17. Oktober 2023
Neues Feature	Standardanwendungskonfiguration für Serverless EMR	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 Serverless 6.13,0	11. September 2023
Neue Veröffentlichung	EMR Serverless 6.12.0	21. Juli 2023
Neue Veröffentlichung	EMR Serverless 6.11.0	08. Juni 2023
Aktualisierung der Richtlinie für dienstbezogene Rollen	Die `AmazonEMRServerlessServiceRolePolicy`_SLRRolle wurde aktualisiert, sodass die Nutzung auf Kontoebene im Namespace veröffentlicht wird. `"AWS/Usage"`	20. April 2023
EMR Serverless allgemeine 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.