Entwicklungsleitfaden für Amazon EMR in EKS

Amazon EMR

Amazon EMR: Entwicklungsleitfaden für Amazon EMR in EKS

Table of Contents

Was ist Amazon EMR in EKS?

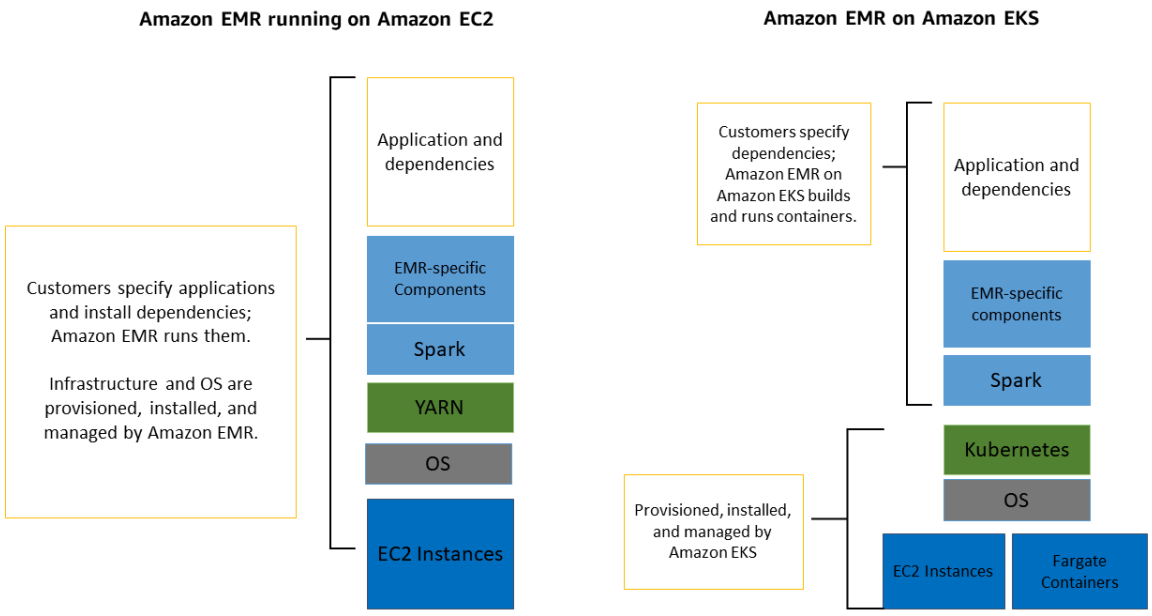
Amazon EMR in EKS bietet eine Bereitstellungsoption für Amazon EMR, mit der Sie Open-Source-Big-Data-Frameworks auf Amazon Elastic Kubernetes Service (Amazon EKS) ausführen können. Mit dieser Bereitstellungsoption können Sie sich auf die Ausführung von Analyse-Workloads konzentrieren, während Amazon EMR in EKS Container für Open-Source-Anwendungen erstellt, konfiguriert und verwaltet.

Wenn Sie Amazon EMR bereits verwenden, können Sie jetzt Amazon-EMR-basierte Anwendungen mit anderen Anwendungstypen auf demselben Amazon-EKS-Cluster ausführen. Diese Bereitstellungsoption verbessert auch die Ressourcennutzung und vereinfacht die Infrastrukturverwaltung in mehreren Availability Zones. Wenn Sie bereits Big-Data-Frameworks auf Amazon EKS ausführen, können Sie jetzt Amazon EMR verwenden, um die Bereitstellung und Verwaltung zu automatisieren und Apache Spark schneller auszuführen.

Amazon EMR in EKS ermöglicht es Ihrem Team, effizienter zusammenzuarbeiten und riesige Datenmengen einfacher und kostengünstiger zu verarbeiten:

- Sie können Anwendungen auf einem gemeinsamen Ressourcenpool ausführen, ohne Infrastruktur bereitstellen zu müssen. Sie können Amazon EMR Studio und das AWS-SDK oder AWS CLI verwenden, um Analyseanwendungen zu entwickeln, einzureichen und zu diagnostizieren, die auf EKS-Clustern ausgeführt werden. Sie können geplante Aufträge in Amazon EMR in EKS mit dem selbstverwalteten Apache Airflow oder Amazon Managed Workflows für Apache Airflow (MWAA) ausführen.
- Infrastrukturteams können eine gemeinsame Computerplattform zentral verwalten, um Amazon-EMR-Workloads mit anderen containerbasierten Anwendungen zu konsolidieren. Sie können das Infrastrukturmanagement mit gängigen Amazon-EKS-Tools vereinfachen und einen gemeinsamen Cluster für Workloads nutzen, die unterschiedliche Versionen von Open-Source-Frameworks benötigen. Mit automatisiertem Kubernetes-Clustermanagement und Betriebssystem-Patching können Sie auch den Betriebsaufwand reduzieren. Mit Amazon EC2 und AWS Fargate können Sie mehrere Rechenressourcen aktivieren, um Leistungs-, Betriebs- oder Finanzanforderungen zu erfüllen.

Das folgende Diagramm stellt die beiden verschiedenen Bereitstellungsmodelle für Amazon EMR dar.



Themen

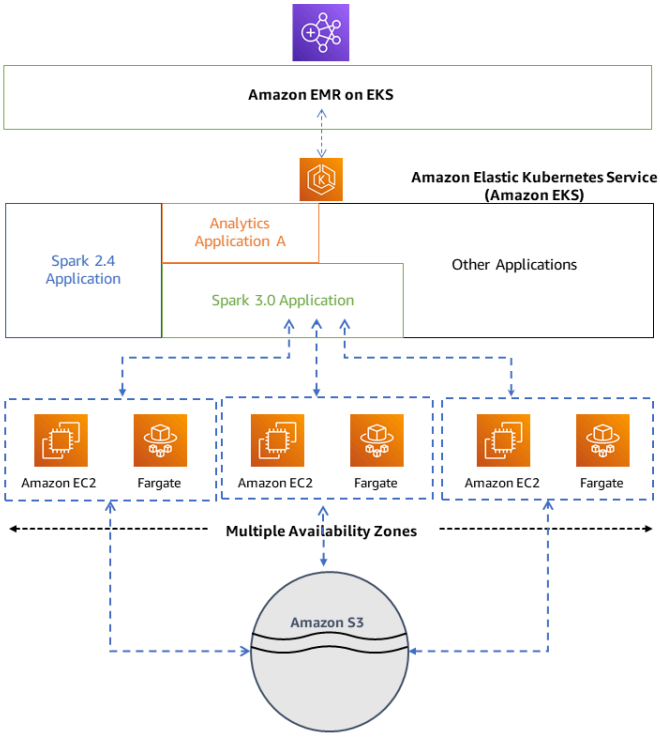
- Architektur
- Konzepte
- Wie die Komponenten zusammenarbeiten

Architektur

Amazon EMR in EKS koppelt Anwendungen lose an die Infrastruktur, auf der sie ausgeführt werden. Jede Infrastrukturebene bietet die Orchestrierung für die nachfolgende Ebene. Wenn Sie einen Auftrag an Amazon EMR senden, enthält Ihre Aufgabendefinition alle anwendungsspezifischen Parameter. Amazon EMR verwendet diese Parameter, um Amazon EKS mitzuteilen, welche Pods und Container bereitgestellt werden sollen. Anschließend stellt Amazon EKS die für die Ausführung der Aufgabe erforderlichen Rechenressourcen von Amazon EC2 und AWS Fargate online.

Mit dieser losen Verkoppelung von Services können Sie mehrere, sicher isolierte Aufträge gleichzeitig ausführen. Sie können denselben Auftrag auch mit verschiedenen Datenverarbeitungs-Backends vergleichen oder Ihren Auftrag auf mehrere Availability Zones verteilen, um die Verfügbarkeit zu verbessern.

Das folgende Diagramm veranschaulicht, wie Amazon EMR in EKS mit anderen AWS-Services funktioniert.



Konzepte

Kubernetes-Namespace

Amazon EKS verwendet Kubernetes-Namespaces, um Cluster-Ressourcen auf mehrere Benutzer und Anwendungen aufzuteilen. Diese Namespaces bilden die Grundlage für Multi-Tenant-Umgebungen. Ein Kubernetes-Namespace kann entweder Amazon EC2 oder AWS Fargate als Datenverarbeitungsanbieter haben. Diese Flexibilität bietet Ihnen verschiedene Leistungs- und Kostenoptionen für die Ausführung Ihrer Aufträge.

Virtueller Cluster

Ein virtueller Cluster ist ein Kubernetes-Namespace, bei dem Amazon EMR registriert ist. Amazon EMR verwendet virtuelle Cluster, um Aufträge auszuführen und Endpunkte zu hosten. Mehrere virtuelle Cluster können durch denselben physischen Cluster unterstützt werden. Jeder virtuelle Cluster ist jedoch einem Namespace auf einem EKS-Cluster zugeordnet. Virtuelle Cluster erzeugen

keine aktiven Ressourcen, die zu Ihrer Rechnung beitragen oder für die ein Lebenszyklus-Management außerhalb des Services erforderlich ist.

Aufgabenausführung

Eine Aufgabenausführung ist eine Arbeitseinheit, z. B. eine Spark-Jar-, PySpark-Skript- oder SparkSQL-Abfrage, die Sie an Amazon EMR in EKS senden. Ein Auftrag kann mehrere Auftragsausführungen haben. Wenn Sie eine Aufgabenausführung einreichen, geben Sie die folgenden Informationen an:

- Ein virtueller Cluster, in dem der Auftrag ausgeführt werden soll.
- Ein Auftragsname zur Identifizierung des Aufträge.
- Die Ausführungsrolle – eine bereichsbezogene IAM-Rolle, die den Auftrag ausführt und es Ihnen ermöglicht, anzugeben, auf welche Ressourcen der Auftrag zugreifen kann.
- Das Amazon-EMR-Versions-Label, das die Version der zu verwendenden Open-Source-Anwendungen bestimmt.
- Die Artefakte, die Sie beim Absenden Ihres Aufträge verwenden sollen, z. B. Spark-Submit-Parameter.

Standardmäßig werden Protokolle auf den Spark-History-Server hochgeladen und sind über AWS Management Console zugänglich. Sie können auch Ereignisprotokolle, Ausführungsprotokolle und Metriken an Amazon S3 und Amazon CloudWatch übertragen.

Amazon-EMR-Container

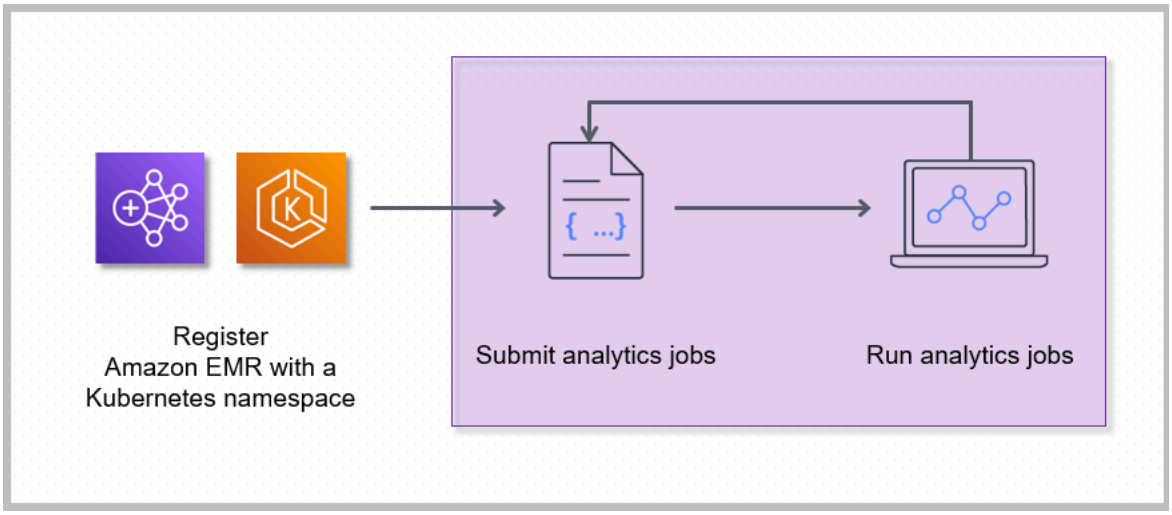
Amazon-EMR-Containers ist der API-Name für Amazon EMR in EKS. Das `emr-containers`-Präfix wird in den folgenden Szenarien verwendet:

- Es ist das Präfix in den CLI-Befehlen für Amazon EMR in EKS. Zum Beispiel `aws emr-containers start-job-run`.
- Es ist das Präfix vor IAM-Richtlinienaktionen für Amazon EMR in EKS. Zum Beispiel `"Action": [ "emr-containers:StartJobRun"]`. Weitere Informationen finden Sie unter Richtlinienaktionen für Amazon EMR in EKS.
- Es ist das Präfix, das in Amazon EMR in EKS-Serviceendpunkten verwendet wird. Zum Beispiel `emr-containers.us-east-1.amazonaws.com`. Weitere Informationen finden Sie unter Service-Endpunkte für Amazon EMR in EKS.

Wie die Komponenten zusammenarbeiten

Die folgenden Schritte und das Diagramm veranschaulichen den Arbeitsablauf von Amazon EMR in EKS:

- Verwenden Sie einen vorhandenen Amazon-EKS-Cluster oder erstellen Sie einen mit dem Befehlszeilenprogramm eksctl oder der Amazon-EKS-Konsole.
- Erstellen Sie einen virtuellen Cluster, indem Sie Amazon EMR mit einem Namespace auf einem EKS-Cluster registrieren.
- Senden Sie Ihren Auftrag mithilfe von AWS CLI oder SDK an den virtuellen Cluster.



Durch die Registrierung von Amazon EMR mit einem Kubernetes-Namespace auf Amazon EKS wird ein virtueller Cluster erstellt. Amazon EMR kann dann Analytics-Workloads in diesem Namespace ausführen. Wenn Sie Amazon EMR in EKS verwenden, um Spark-Aufträge an den virtuellen Cluster zu senden, fordert Amazon EMR in EKS den Kubernetes-Planer auf Amazon EKS auf, Pods zu planen.

Für jeden Auftrag, den Sie ausführen, erstellt Amazon EMR in EKS einen Container mit einem Amazon-Linux-2-Basis-Image, Apache Spark und zugehörigen Abhängigkeiten. Jeder Auftrag wird in einem Pod ausgeführt, der den Container herunterlädt und mit der Ausführung beginnt. Der Pod wird beendet, nachdem der Auftrag beendet wurde. Wenn das Image des Containers zuvor auf dem Knoten bereitgestellt wurde, wird ein zwischengespeichertes Image verwendet und der Download wird umgangen. Sidecar-Container, wie z. B. Protokoll- oder Metrik-Forwarder, können im Pod bereitgestellt werden. Nachdem der Auftrag beendet wurde, können Sie ihn immer noch mit der Benutzeroberfläche der Spark-Anwendung in der Amazon-EMR-Konsole debuggen.

Erste Schritte

Dieses Thema hilft Ihnen bei den ersten Schritten mit Amazon EMR in EKS, indem Sie eine Spark-Anwendung auf einem virtuellen Cluster bereitstellen. Bevor Sie beginnen, sollten Sie sicherstellen, dass Sie die in Einrichten von Amazon EMR in EKS beschriebenen Schritte ausgeführt haben. Weitere Vorlagen, die Ihnen beim Einstieg helfen können, siehe unseren Handbuch für bewährte Methoden für EMR Container auf GitHub.

Sie benötigen die folgenden Informationen aus den Einrichtungsschritten:

- Virtuelle Cluster-ID für den Amazon-EKS-Cluster und den Kubernetes-Namespace, die bei Amazon EMR registriert sind

  > **Important**
  >
  > Achten Sie beim Erstellen eines EKS-Clusters darauf, m5.xlarge als Instance-Typ oder einen anderen Instance-Typ mit einer höheren CPU und einem höheren Arbeitsspeicher zu verwenden. Die Verwendung eines Instance-Typs mit weniger CPU oder Arbeitsspeicher als m5.xlarge kann aufgrund unzureichender verfügbarer Ressourcen im Cluster zu einem Auftragsfehler führen.

- Name der IAM-Rolle, die für die Auftragsausführung verwendet wird
- Release-Label für die Amazon-EMR-Version (z. B.`emr-6.4.0-latest`)
- Zielziele für die Protokollierung und Überwachung:
  - Amazon-CloudWatch-Protokollgruppen-Name und Protokollstream-Präfix
  - Amazon-S3-Standort zum Speichern von Ereignis- und Containerprotokollen

> **Important**
>
> Amazon EMR in EKS-Aufträge verwenden Amazon CloudWatch und Amazon S3 als Ziele für die Überwachung und Protokollierung. Sie können den Aufgabenfortschritt überwachen und Fehler beheben, indem Sie sich die an diese Ziele gesendeten Aufgabenprotokolle ansehen. Um die Protokollierung zu aktivieren, muss die IAM-Richtlinie, die der IAM-Rolle für die Aufgabenausführung zugeordnet ist, über die erforderlichen Berechtigungen für den Zugriff auf die Zielressourcen verfügen. Wenn die IAM-Richtlinie nicht über die erforderlichen Berechtigungen verfügt, müssen Sie die unter Die Vertrauensrichtlinie

der Auftragsausführungsrolle aktualisieren Konfiguration einer Aufgabenausführung zur Verwendung von Amazon-S3-Protokollen und Konfiguration einer Aufgabenausführung zur Verwendung von CloudWatch Logs beschriebenen Schritte ausführen, bevor Sie diese Beispielaufgabe ausführen.

Eine Spark-Anwendung ausführen

Führen Sie die folgenden Schritte aus, um eine einfache Spark-Anwendung bei Amazon EMR in EKS auszuführen. Die `entryPoint`-Anwendungsdatei für eine Spark-Python-Anwendung befindet sich unter `s3://`*`REGION`*`.elasticmapreduce/emr-containers/samples/wordcount/scripts/wordcount.py`. Die *`REGION`* ist die Region, in der sich Ihr virtueller Amazon EMR in EKS-Cluster befindet, z. B. *`us-east-1`*.

1. Aktualisieren Sie die IAM-Richtlinie für die Auftragausführungsrolle mit den erforderlichen Berechtigungen, wie die folgenden Richtlinienerklärungen zeigen.

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

```
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET-OUTPUT/*",
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET-LOGGING/*"
            ]
        },
        {
            "Sid": "DescribeAndCreateCloudwatchLogStream",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:*:*:*"
            ]
        },
        {
            "Sid": "WriteToCloudwatchLogs",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:*:*:log-group:my_log_group_name:log-
stream:my_log_stream_prefix/*"
            ]
        }
    ]
}
```

- Die erste Anweisung `ReadFromLoggingAndInputScriptBuckets` in dieser Richtlinie gewährt `ListBucket` und `GetObjects` Zugriff auf die folgenden Amazon-S3-Buckets:
  - *REGION*`.elasticmapreduce` – der Bucket, in dem sich die `entryPoint` Anwendungsdatei befindet.
  - *DOC-EXAMPLE-BUCKET-OUTPUT* – ein Bucket, das Sie für Ihre Ausgabedaten definieren.
  - *DOC-EXAMPLE-BUCKET-LOGGING* – ein Bucket, das Sie für Ihre Protokollierungs-Daten definieren.

- Die zweite Anweisung `WriteToLoggingAndOutputDataBuckets` in dieser Richtlinie erteilt dem Auftrag die Erlaubnis, Daten in Ihre Ausgabe- bzw. Protokollierungs-Buckets zu schreiben.
- Die dritte Anweisung `DescribeAndCreateCloudwatchLogStream` erteilt dem Auftrag die Erlaubnis, Amazon CloudWatch Logs zu beschreiben und zu erstellen.
- Die vierte Anweisung `WriteToCloudwatchLogs` gewährt Berechtigungen zum Schreiben von Protokollen in eine Amazon-CloudWatch-Protokollgruppe, die *my_log_group_name* unter einem Protokollstream namens *my_log_stream_prefix* ist.

2. Verwenden Sie den folgenden Befehl, um eine Spark-Python-Anwendung auszuführen. Ersetzen Sie alle ersetzbaren *roten kursiv geschriebenen* Werte durch entsprechende Werte. Die *REGION* ist die Region, in der sich Ihr virtueller Amazon EMR in EKS-Cluster befindet, z. B. *us-east-1*.

```
aws emr-containers start-job-run \
--virtual-cluster-id cluster_id \
--name sample-job-name \
--execution-role-arn execution-role-arn \
--release-label emr-6.4.0-latest \
--job-driver '{
  "sparkSubmitJobDriver": {
    "entryPoint": "s3://REGION.elasticmapreduce/emr-containers/samples/wordcount/
scripts/wordcount.py",
    "entryPointArguments": ["s3://DOC-EXAMPLE-BUCKET-OUTPUT/wordcount_output"],
    "sparkSubmitParameters": "--conf spark.executor.instances=2 --
conf spark.executor.memory=2G --conf spark.executor.cores=2 --conf
 spark.driver.cores=1"
  }
}' \
--configuration-overrides '{
  "monitoringConfiguration": {
    "cloudWatchMonitoringConfiguration": {
      "logGroupName": "my_log_group_name",
      "logStreamNamePrefix": "my_log_stream_prefix"
    },
    "s3MonitoringConfiguration": {
       "logUri": "s3://DOC-EXAMPLE-BUCKET-LOGGING"
    }
  }
}'
```

Die Ausgabedaten dieses Aufträge sind unter `s3://DOC-EXAMPLE-BUCKET-OUTPUT/wordcount_output` verfügbar.

Sie können auch eine JSON-Datei mit bestimmten Parametern für Ihre Auftragsausführung erstellen. Führen Sie dann den `start-job-run`-Befehl mit dem Pfad zur JSON-Datei aus. Weitere Informationen finden Sie unter Reichen Sie einen Auftrag ein, der ausgeführt wird mit `StartJobRun`. Weitere Informationen zur Konfiguration von Auftrag-Ausführungsparametern finden Sie unter Optionen für die Konfiguration einer Aufgabenausführung.

3. Verwenden Sie den folgenden Befehl, um eine Spark-SQL-Anwendung auszuführen. Ersetzen Sie alle *rot kursiv geschriebenen* Werte durch entsprechende Werte. Die *REGION* ist die Region, in der sich Ihr virtueller Amazon EMR in EKS-Cluster befindet, z. B. *us-east-1*.

```
aws emr-containers start-job-run \
--virtual-cluster-id cluster_id \
--name sample-job-name \
--execution-role-arn execution-role-arn \
--release-label emr-6.7.0-latest \
--job-driver '{
  "sparkSqlJobDriver": {
    "entryPoint": "s3://query-file.sql",
    "sparkSqlParameters": "--conf spark.executor.instances=2 --
conf spark.executor.memory=2G --conf spark.executor.cores=2 --conf
 spark.driver.cores=1"
  }
}' \
--configuration-overrides '{
  "monitoringConfiguration": {
    "cloudWatchMonitoringConfiguration": {
      "logGroupName": "my_log_group_name",
      "logStreamNamePrefix": "my_log_stream_prefix"
    },
    "s3MonitoringConfiguration": {
       "logUri": "s3://DOC-EXAMPLE-BUCKET-LOGGING"
    }
  }
}'
```

Eine Beispiel-SQL-Abfragedatei ist unten dargestellt. Sie benötigen einen externen Dateispeicher wie S3, in dem die Daten für die Tabellen gespeichert werden.

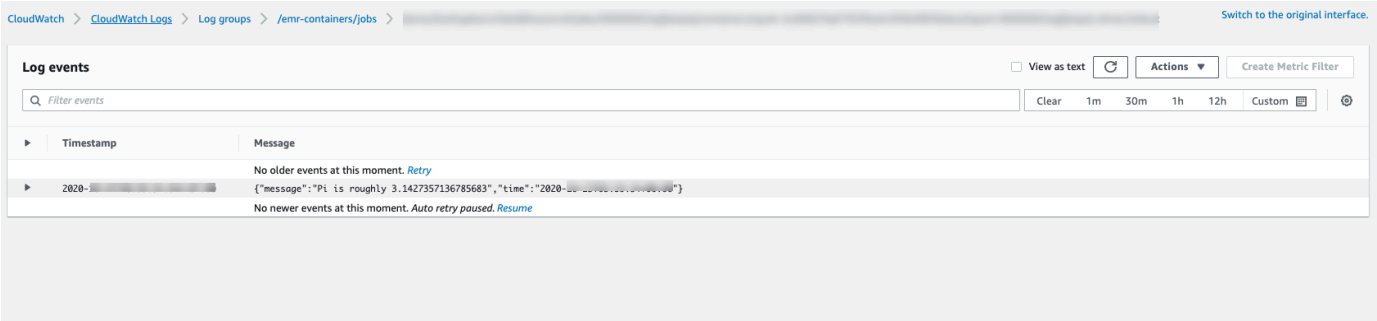
```
CREATE DATABASE demo;
CREATE EXTERNAL TABLE IF NOT EXISTS demo.amazonreview( marketplace string,
 customer_id string, review_id  string, product_id  string, product_parent  string,
 product_title  string, star_rating  integer, helpful_votes  integer, total_votes
  integer, vine  string, verified_purchase  string, review_headline  string,
 review_body  string, review_date  date, year  integer) STORED AS PARQUET LOCATION
 's3://URI to parquet files';
SELECT count(*) FROM demo.amazonreview;
SELECT count(*) FROM demo.amazonreview WHERE star_rating = 3;
```

Die Ausgabe für diesen Auftrag ist in den Standardprotokollen des Treibers in S3 oder CloudWatch verfügbar, je nachdem, was auf `monitoringConfiguration` konfiguriert ist.

4. Sie können auch eine JSON-Datei mit bestimmten Parametern für Ihre Auftragsausführung erstellen. Führen Sie dann den Befehl start-job-run mit einem Pfad zur JSON-Datei aus. Weitere Informationen finden Sie unter Ausführen einer Auftragsausführung. Weitere Informationen zur Konfiguration von Auftrag-Ausführungsparametern finden Sie unter Optionen für die Konfiguration einer Auftragsausführung.

   Um den Fortschritt des Aufträge zu überwachen oder Fehler zu debuggen, können Sie die in Amazon S3, CloudWatch Logs oder beides hochgeladenen Protokolle überprüfen. Weitere Informationen zum Protokollpfad in Amazon S3 finden Sie unter Aufgabenausführung für die Verwendung von S3-Protokollen konfigurieren und Cloudwatch-Protokolle unter Aufgabenausführung zur Verwendung von CloudWatch Logs konfigurieren. Befolgen Sie die Anweisungen unten, um Protokolle in CloudWatch Logs anzuzeigen.

   - Öffnen Sie die CloudWatch-Konsole unter https://console.aws.amazon.com/cloudwatch/.
   - Wählen Sie im Navigationsbereich Protokolle aus. Wählen Sie dann die Protokollgruppen aus.
   - Wählen Sie die Protokollgruppe für Amazon EMR in EKS aus und sehen Sie sich dann die hochgeladenen Protokollereignisse an.



Important

Aufträge haben eine Standardkonfigurierte Wiederholungsrichtlinie. Informationen zum Ändern oder Deaktivieren der Konfiguration finden Sie unter Richtlinien zur Aufgabenwiederholung verwenden.

Links zu den Leitfäden zu bewährten Methoden für Amazon EMR in EKS auf GitHub

Wir haben den Leitfaden für bewährte Methoden für Amazon EMR in EKS in Zusammenarbeit mit der Open-Source-Community erstellt, sodass wir schnell iterieren und Empfehlungen für eine Vielzahl von Anwendungsfällen geben können. Wir empfehlen Ihnen, für die Abschnitte den Leitfaden für bewährte Methoden für Amazon EMR in EKS zu verwenden. Wählen Sie die Links in jedem Abschnitt aus, um zur GitHub Website zu gelangen.

Sicherheit

> **Note**
>
> Weitere Informationen zur Sicherheit mit Amazon EMR in EKS finden Sie unter Bewährte Methoden für Sicherheit in Amazon EMR in EKS.

Bewährte Methoden zur Verschlüsselung: Verwendung der Verschlüsselung für Daten im Ruhezustand und bei der Übertragung.

Unter Netzwerksicherheit verwalten wird beschrieben, wie Sicherheitsgruppen für Pods für Amazon EMR in EKS konfiguriert werden, während Sie eine Verbindung zu Datenquellen herstellen, die in AWS-Services wie Amazon RDS und Amazon Redshift gehostet werden.

Verwenden von AWS Secrets Manager, um Secrets zu speichern.

Einreichung einer Pyspark-Aufgabe

Pyspark-Aufgaben-Einreichung: spezifiziert verschiedene Arten von Paketierungen für PySpark-Anwendungen unter Verwendung von Paketierungsformaten wie zip, egg, wheel und pex.

Speicher

Verwendung von EBS-Volumes: Verwendung von statischer und dynamischer Bereitstellung für Aufträge, die EBS-Volumes benötigen.

Verwendung von Volumes von Amazon FSx für Lustre: So verwenden Sie statische und dynamische Bereitstellung für Aufträge, die Volumes von Amazon FSx für Lustre benötigen.

Verwenden von Instance-Speicher-Volumes: So verwenden Sie Instance-Speicher-Volumes für die Aufgabenverarbeitung.

Metastore-Integration

Verwenden von Hive Metastore: bietet verschiedene Möglichkeiten, Hive Metastore zu verwenden.

Verwendung von AWS Glue: bietet verschiedene Möglichkeiten, den AWS Glue-Katalog zu konfigurieren.

Debugging

Verwendung von Spark-Debugging: So ändern Sie die Protokollebene.

Verbindung zur Spark-Benutzeroberfläche auf dem Treiber-Pod herstellen.

So verwenden Sie den selbst gehosteten Spark-Verlaufsserver mit Amazon EMR in EKS.

Beheben von Problemen in Amazon EMR in EKS

Fehlerbehebung.

Knotenplatzierung

Verwendung von Kubernetes-Knotenselektoren für `single-az` und andere Anwendungsfälle.

Verwenden der Fargate-Knotenplatzierung.

Leistung

Verwenden von Dynamic Resource Allocation (DRA).

Bewährte EKS-Methoden für das Amazon VPC Container Network Interface Plugin (CNI), Cluster Autoscaler und Core DNS.

Kostenoptimierung

Verwendung von Spot Instances: Bewährte Methoden für Amazon-EC2-Spot Instances und Verwendung des Features zur Außerbetriebnahme von Spark-Knoten.

Verwenden von AWS Outposts

Ausführen von Amazon EMR in EKS mithilfe von AWS Outposts

Anpassen von Docker-Images für Amazon EMR in EKS

Sie können benutzerdefinierte Docker-Images mit Amazon EMR in EKS verwenden. Das Anpassen des Amazon EMR in EKS-Runtime-Images bietet die folgenden Vorteile:

- Packen Sie Anwendungsabhängigkeiten und Laufzeitumgebung in einen einzigen unveränderlichen Container, der die Portabilität fördert und das Abhängigkeitsmanagement für jeden Workload vereinfacht.
- Installieren und konfigurieren Sie Pakete, die für Ihre Workloads optimiert sind. Diese Pakete sind in der öffentlichen Distribution von Amazon-EMR-Laufzeiten möglicherweise nicht allgemein verfügbar.
- Integrieren Sie Amazon EMR in EKS in die derzeit etablierten Erstellungs-, Test- und Bereitstellungsprozesse in Ihrem Unternehmen, einschließlich lokaler Entwicklung und Tests.
- Wenden Sie etablierte Sicherheitsprozesse an, wie z. B. das Scannen von Images, die die Compliance- und Governance-Anforderungen in Ihrem Unternehmen erfüllen.

Themen

Wie passen Sie Docker-Images an

Gehen Sie wie folgt vor, um Docker-Images für Amazon EMR in EKS anzupassen.

Hier sind weitere Optionen, die Sie bei der Anpassung von Docker-Images in Betracht ziehen sollten:

- Docker-Images für interaktive Endpunkte anpassen
- Arbeiten mit Images mit mehreren Architekturen

Voraussetzungen

- Führen Sie die folgenden Einrichten von Amazon EMR in EKS Schritte auf Amazon EMR in EKS aus.
- Installieren Sie Docker in Ihrer Umgebung. Weitere Informationen finden Sie unter Docker holen.

Schritt 1: Ein Basis-Image aus Amazon Elastic Container Registry (Amazon ECR) abrufen

Das Basis-Image enthält die Amazon-EMR-Laufzeit und Konnektoren, die für den Zugriff auf andere AWS -Services verwendet werden. Für Amazon EMR 6.9.0 und höher können Sie die Basis-Images aus der Amazon ECR Public Gallery abrufen. Durchsuchen Sie die Galerie nach dem Image-Link und laden Sie das Image in Ihren lokalen Workspace. Für die Version Amazon EMR 7.1.0 erhalten Sie beispielsweise mit dem folgenden `docker pull` Befehl das neueste Standard-Basis-Image. Sie können `emr-7.1.0:latest` durch `emr-7.1.0-spark-rapids:latest` ersetzen, um das Image abzurufen, das über den Nvidia-RAPIDS-Beschleuniger verfügt. Sie können auch `emr-7.1.0:latest` durch `emr-7.1.0-java11:latest` ersetzen, um das Image mit der Java-11-Laufzeit abzurufen.

```
docker pull public.ecr.aws/emr-on-eks/spark/emr-7.1.0:latest
```

Wenn Sie das Basis-Image für eine Amazon EMR 6.9.0 oder frühere Versionen abrufen möchten oder wenn Sie es lieber von Amazon-ECR-Registrierungskonten in jeder Region abrufen möchten, gehen Sie wie folgt vor:

1. Wählen Sie einen Basis-Image-URI aus. Der Image-URI folgt diesem Format, wie das folgende *ECR-registry-account*.dkr.ecr.*Region*.amazonaws.com/spark/*container-image-tag* Beispiel zeigt.

   ```
   895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.6.0:latest
   ```

   Informationen zur Auswahl eines Basis-Images in Ihrer Region finden Sie unter Wie wählt man einen Basis-Image-URI aus.

2. Melden Sie sich beim Amazon-ECR-Repository an, in dem das Basis-Image gespeichert ist. Ersetzen Sie *895885662937* und *us-west-2* durch das Amazon ECR-Registrierungskonto und die Region, die Sie ausgewählt haben. AWS

```
aws ecr get-login-password --region us-west-2 | docker login --username AWS --
password-stdin 895885662937.dkr.ecr.us-west-2.amazonaws.com
```

3. Ziehen Sie das Basis-Image in Ihren lokalen Workspace. Ersetzen Sie *emr-6.6.0:latest* durch das Container-Image-Tag, das Sie ausgewählt haben.

```
docker pull 895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.6.0:latest
```

Schritt 2: Ein Basis-Image anpassen

Gehen Sie wie folgt vor, um das Basis-Image, das Sie aus Amazon ECR abgerufen haben, anzupassen.

1. Erstellen Sie ein neues `Dockerfile` in Ihrem lokalen Workspace.
2. Bearbeiten Sie die soeben von Ihnen erstellte `Dockerfile` und fügen Sie die folgenden Inhalte hinzu. Dabei wird das Container-Image `Dockerfile` verwendet, das Sie aus `895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.6.0:latest` abgerufen haben.

```
FROM 895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.6.0:latest
USER root
### Add customization commands here ####
USER hadoop:hadoop
```

3. Fügen Sie Befehle in `Dockerfile` hinzu, um das Basis-Image anzupassen. Fügen Sie beispielsweise einen Befehl zur Installation von Python-Bibliotheken hinzu, wie im Folgenden `Dockerfile` gezeigt wird.

```
FROM 895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.6.0:latest
USER root
RUN pip3 install --upgrade boto3 pandas numpy // For python 3
USER hadoop:hadoop
```

4. Führen Sie in demselben Verzeichnis, in dem `Dockerfile` erstellt wurde, den folgenden Befehl aus, um das Docker-Image zu erstellen. Geben Sie einen Namen für das Docker-Image ein, z. B. *emr6.6_custom.*

```
docker build -t emr6.6_custom .
```

Schritt 3: (Optional, aber empfohlen) Ein benutzerdefiniertes Image validieren

Wir empfehlen Ihnen, die Kompatibilität Ihres benutzerdefinierten Images zu testen, bevor Sie es veröffentlichen. Sie können die benutzerdefinierte Image-CLI von Amazon EMR in EKS verwenden, um zu überprüfen, ob Ihr Image über die erforderlichen Dateistrukturen und die richtigen Konfigurationen für die Ausführung auf Amazon EMR in EKS verfügt.

> **Note**
>
> Die CLI für benutzerdefinierte Images von Amazon EMR in EKS kann nicht bestätigen, dass Ihr Image fehlerfrei ist. Seien Sie vorsichtig, wenn Sie Abhängigkeiten von den Basis-Images entfernen.

Führen Sie den folgenden Befehl aus, um Ihre Aufgaben zu löschen.

1. Laden Sie Amazon EMR auf der CLI für benutzerdefinierte EKS-Images herunter und installieren Sie es. Weitere Informationen finden Sie im CLI-Installationshandbuch für Amazon EMR in EKS Custom Image.
2. Führen Sie den folgenden Befehl aus, um die Installation zu testen.

```
emr-on-eks-custom-image --version
```

   Hier ein Beispiel für die Ausgabe.

```
Amazon EMR on EKS Custom Image CLI
Version: x.xx
```

3. Führen Sie den folgenden Befehl aus, um Ihre Ressourcen zu löschen.

```
emr-on-eks-custom-image validate-image -i image_name -r release_version [-
t image_type]
```

- `-i` gibt den lokalen Image-URI an, der validiert werden muss. Dies kann der Image-URI sein, ein beliebiger Name oder Tag, den Sie für Ihr Image definiert haben.
- `-r` gibt die genaue Release-Version für das Basis-Image an, zum Beispiel `emr-6.6.0-latest`.
- `-t` gibt den Imagetyp an. Wenn es sich um ein Image-Bild handelt, geben Sie `spark` ein. Der Standardwert ist `spark`. Die aktuelle CLI-Version von Amazon EMR in EKS für benutzerdefinierte Images unterstützt nur Spark-Laufzeit-Images.

Wenn Sie den Befehl erfolgreich ausführen und das benutzerdefinierte Image alle erforderlichen Konfigurationen und Dateistrukturen erfüllt, werden in der zurückgegebenen Ausgabe die Ergebnisse aller Tests angezeigt, wie das folgende Beispiel zeigt.

```
Amazon EMR on EKS Custom Image Test
Version: x.xx
... Checking if docker cli is installed
... Checking Image Manifest
[INFO] Image ID: xxx
[INFO] Created On: 2021-05-17T20:50:07.986662904Z
[INFO] Default User Set to hadoop:hadoop : PASS
[INFO] Working Directory Set to /home/hadoop : PASS
[INFO] Entrypoint Set to /usr/bin/entrypoint.sh : PASS
[INFO] SPARK_HOME is set with value: /usr/lib/spark : PASS
[INFO] JAVA_HOME is set with value: /etc/alternatives/jre : PASS
[INFO] File Structure Test for spark-jars in /usr/lib/spark/jars: 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 bin-files in /usr/bin: PASS
... Start Running Sample Spark Job
[INFO] Sample Spark Job Test with local:///usr/lib/spark/examples/jars/spark-
examples.jar : PASS
----------------------------------------------------------------
Overall Custom Image Validation Succeeded.
----------------------------------------------------------------
```

Wenn das benutzerdefinierte Image die erforderlichen Konfigurationen oder Dateistrukturen nicht erfüllt, treten Fehlermeldungen auf. Die zurückgegebene Ausgabe enthält Informationen zu den falschen Konfigurationen oder Dateistrukturen.

Schritt 4: Ein benutzerdefiniertes Image veröffentlichen

Führen Sie den folgenden Befehl aus, um Ihre Docker-Images zu löschen.

1. Führen Sie den folgenden Befehl aus, um ein Amazon-ECR-Repository zum Speichern Ihres Docker-Images zu erstellen. Geben Sie einen Namen für Ihr Repository ein, zum Beispiel *emr6.6_custom_repo.* Ersetzen Sie *us-west-2* durch Ihre Region.

   ```
   aws ecr create-repository \
       --repository-name emr6.6_custom_repo \
       --image-scanning-configuration scanOnPush=true \
       --region us-west-2
   ```

   Weitere Informationen finden Sie unter Erstellen eines Repositorys im Amazon-ECR-Benutzerhandbuch.

2. Führen Sie den folgenden Befehl aus, um Ihre Ressourcen zu löschen.

   ```
   aws ecr get-login-password --region us-west-2 | docker login --username AWS --
   password-stdin aws_account_id.dkr.ecr.us-west-2.amazonaws.com
   ```

   Weitere Informationen finden Sie unter Authentifizieren bei Ihrer Standardregistrierung im Amazon-ECR-Benutzerhandbuch.

3. Kennzeichnen und veröffentlichen Sie ein Image im von Ihnen erstellten Amazon-ECR-Repository.

   Markieren Sie das Image.

   ```
   docker tag emr6.6_custom aws_account_id.dkr.ecr.us-
   west-2.amazonaws.com/emr6.6_custom_repo
   ```

   Übertragen Sie das Image per Push.

```
docker push aws_account_id.dkr.ecr.us-west-2.amazonaws.com/emr6.6_custom_repo
```

Weitere Informationen finden Sie unter Übertragen eines Images an Amazon ECR im Amazon-ECR-Benutzerhandbuch.

Schritt 5: Einen Spark-Workload mit einem benutzerdefinierten Image in Amazon EMR einreichen

Nachdem ein benutzerdefiniertes Image erstellt und veröffentlicht wurde, können Sie mithilfe eines benutzerdefinierten Images einen Amazon-EMR-in-EKS-Auftrag einreichen.

Erstellen Sie zunächst eine start-job-run-request JSON-Datei und geben Sie den `spark.kubernetes.container.image` Parameter an, der auf das benutzerdefinierte Bild verweisen soll, wie die folgende JSON-Beispieldatei zeigt.

> **Note**
>
> Sie können das `local://`-Schema verwenden, um auf Dateien zu verweisen, die im benutzerdefinierten Image verfügbar sind, wie im folgenden JSON-Snippet mit dem Argument `entryPoint` gezeigt. Sie können das `local://`-Schema auch verwenden, um auf Anwendungsabhängigkeiten zu verweisen. Alle Dateien und Abhängigkeiten, auf die mithilfe des `local://`-Schemas verwiesen wird, müssen bereits im angegebenen Pfad im benutzerdefinierten Image vorhanden sein.

```
{
    "name": "spark-custom-image",
    "virtualClusterId": "virtual-cluster-id",
    "executionRoleArn": "execution-role-arn",
    "releaseLabel": "emr-6.6.0-latest",
    "jobDriver": {
      "sparkSubmitJobDriver": {
        "entryPoint": "local:///usr/lib/spark/examples/jars/spark-examples.jar",
        "entryPointArguments": [
                  "10"
              ],
```

```
        "sparkSubmitParameters": "--class org.apache.spark.examples.SparkPi --conf
 spark.kubernetes.container.image=123456789012.dkr.ecr.us-west-2.amazonaws.com/
emr6.6_custom_repo"
       }
    }
}
```

Sie können das benutzerdefinierte Image auch mit `applicationConfiguration`-Eigenschaften referenzieren, wie das folgende Beispiel zeigt.

```
{
    "name": "spark-custom-image",
    "virtualClusterId": "virtual-cluster-id",
    "executionRoleArn": "execution-role-arn",
    "releaseLabel": "emr-6.6.0-latest",
    "jobDriver": {
      "sparkSubmitJobDriver": {
        "entryPoint": "local:///usr/lib/spark/examples/jars/spark-examples.jar",
        "entryPointArguments": [
                  "10"
             ],
         "sparkSubmitParameters": "--class org.apache.spark.examples.SparkPi"
       }
    },
    "configurationOverrides": {
        "applicationConfiguration": [
            {
                "classification": "spark-defaults",
                "properties": {
                    "spark.kubernetes.container.image": "123456789012.dkr.ecr.us-
west-2.amazonaws.com/emr6.6_custom_repo"
                }
            }
        ]
    }
}
```

Führen Sie dann den `start-job-run`-Befehl aus, um den Auftrag zu senden.

```
aws emr-containers start-job-run --cli-input-json file://./start-job-run-request.json
```

Ersetzen Sie in den obigen JSON-Beispielen *emr-6.6.0-latest* durch Ihre Amazon-EMR-Release-Version. Wir empfehlen dringend, die `-latest` Release-Version zu verwenden, um sicherzustellen, dass die ausgewählte Version die neuesten Sicherheitsupdates enthält. Weitere Informationen zu Amazon-EMR-Releaseversionen und ihren Image-Tags finden Sie unter Wie wählt man einen Basis-Image-URI aus.

> **Note**
>
> Sie können `spark.kubernetes.driver.container.image` und `spark.kubernetes.executor.container.image` verwenden, um ein anderes Image für Treiber- und Executor-Pods anzugeben.

Docker-Images für interaktive Endpunkte anpassen

Sie können Docker-Images auch für interaktive Endpunkte anpassen, sodass Sie benutzerdefinierte Basis-Kernel-Images ausführen können. Dadurch können Sie sicherstellen, dass Sie über die Abhängigkeiten verfügen, die Sie benötigen, wenn Sie interaktive Workloads von EMR Studio ausführen.

1. Folgen Sie den oben beschriebenen Schritten 1–4, um ein Docker-Image anzupassen. Für Amazon-EMR-Versionen 6.9.0 und höher können Sie den Basis-Image-URI von der Amazon ECR Public Gallery abrufen. Für Versionen vor Amazon EMR 6.9.0 können Sie das Image in jedem Amazon-ECR-Registrierungskonto in AWS-Region abrufen, und der einzige Unterschied ist der Basis-Image-URI in Ihrer Docker-Datei. Die Basis-Image-URI folgt dem Format:

   ```
   ECR-registry-account.dkr.ecr.Region.amazonaws.com/notebook-spark/container-image-tag
   ```

   Sie müssen `notebook-spark` im Basis-Image den URI anstelle von `spark` verwenden. Das Basis-Image enthält die Spark-Laufzeit und die Notebook-Kernel, die damit ausgeführt werden. Weitere Informationen zur Auswahl von Regionen und Container-Image-Tags finden Sie unter Wie wählt man einen Basis-Image-URI aus.

> **Note**
>
> Derzeit werden nur Überschreibungen von Basis-Images unterstützt und die Einführung völlig neuer Kernel anderer Typen als die Basis-Images wird nicht AWS unterstützt.

2. Erstellen Sie einen interaktiven Endpunkt, der mit dem benutzerdefinierten Image verwendet werden kann.

   Erstellen Sie zunächst eine JSON-Datei mit dem Namen `custom-image-managed-endpoint.json` und den folgenden Inhalten.

```
{
    "name": "endpoint-name",
    "virtualClusterId": "virtual-cluster-id",
    "type": "JUPYTER_ENTERPRISE_GATEWAY",
    "releaseLabel": "emr-6.6.0-latest",
    "executionRoleArn": "execution-role-arn",
    "certificateArn": "certificate-arn",
    "configurationOverrides": {
        "applicationConfiguration": [
            {
                "classification": "jupyter-kernel-overrides",
                "configurations": [
                    {
                        "classification": "python3",
                        "properties": {
                            "container-image": "123456789012.dkr.ecr.us-
west-2.amazonaws.com/custom-notebook-python:latest"
                        }
                    },
                    {
                        "classification": "spark-python-kubernetes",
                        "properties": {
                            "container-image": "123456789012.dkr.ecr.us-
west-2.amazonaws.com/custom-notebook-spark:latest"
                        }
                    }
                ]
            }
        ]
    }
```

```
}
```

Erstellen Sie als Nächstes einen interaktiven Endpunkt mit den in der JSON-Datei angegebenen Konfigurationen, wie das folgende Beispiel zeigt.

```
aws emr-containers create-managed-endpoint --cli-input-json custom-image-managed-
endpoint.json
```

Weitere Informationen finden Sie unter Einen interaktiven Endpunkt für Ihren virtuellen Cluster erstellen.

3. Stellen Sie über EMR Studio eine Verbindung zum interaktiven Endpunkt her. Weitere Informationen finden Sie unter Herstellen einer Verbindung über Studio.

Arbeiten mit Images mit mehreren Architekturen

Amazon EMR in EKS unterstützt Multi-Architektur-Container-Images für Amazon Elastic Container Registry (Amazon ECR). Weitere Informationen finden Sie unter Einführung in Container-Images mit mehreren Architekturen für Amazon ECR.

Benutzerdefinierte Images von Amazon EMR auf EKS unterstützen sowohl AWS Graviton-basierte EC2-Instances als auch EC2-Instances, die nicht auf Graviton basieren. Die Graviton-basierten Images werden in denselben Image-Repositorys in Amazon ECR gespeichert wie Images, die nicht auf Graviton basieren.

Um beispielsweise die Docker-Manifestliste für 6.6.0-Images zu überprüfen, führen Sie den folgenden Befehl aus.

```
docker manifest inspect 895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/
emr-6.6.0:latest
```

Hier wird die Ausgabe gezeigt. Die `arm64`-Architektur ist für die Graviton-Instance vorgesehen. Das `amd64` ist für eine Nicht-Graviton-Instance.

```
{
   "schemaVersion": 2,
   "mediaType": "application/vnd.docker.distribution.manifest.list.v2+json",
   "manifests": [
      {
```

```
         "mediaType": "application/vnd.docker.distribution.manifest.v2+json",
         "size": 1805,
         "digest":
 "xxx123:6b971cb47d11011ab3d45fff925e9442914b4977ae0f9fbcdcf5cfa99a7593f0",
         "platform": {
            "architecture": "arm64",
            "os": "linux"
         }
      },
      {
         "mediaType": "application/vnd.docker.distribution.manifest.v2+json",
         "size": 1805,
         "digest":
 "xxx123:6f2375582c9c57fa9838c1d3a626f1b4fc281e287d2963a72dfe0bd81117e52f",
         "platform": {
            "architecture": "amd64",
            "os": "linux"
         }
      }
   ]
}
```

Führen Sie die folgenden Schritte aus, um Multi-Architektur-Images zu erstellen:

1. Erstellen Sie ein `Dockerfile` mit dem folgenden Inhalt, damit Sie das `arm64`-Image abrufen können.

   ```
   FROM --platform=arm64 895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/
   emr-6.6.0:latest
   USER root

   RUN pip3 install boto3 // install customizations here
   USER hadoop:hadoop
   ```

2. Befolgen Sie die Anweisungen unter Einführung von Multi-Architektur-Container-Images für Amazon ECR, um ein Multi-Architektur-Image zu erstellen.

   > **Note**
   >
   > Sie müssen `arm64` Images auf `arm64` Instances erstellen. In ähnlicher Weise müssen Sie `amd64` Images auf `amd64` Instances erstellen.

Mit dem Befehl Docker `buildx` können Sie auch Images mit mehreren Architekturen erstellen, ohne auf jedem spezifischen Instance-Typ aufbauen zu müssen. Weitere Informationen finden Sie unter Nutzung der Unterstützung mehrerer CPU-Architekturen.

3. Nachdem Sie das Multiarchitektur-Image erstellt haben, können Sie einen Auftrag mit demselben `spark.kubernetes.container.image` Parameter einreichen und ihn auf das Image verweisen. In einem heterogenen Cluster mit sowohl AWS Graviton-basierten als auch nicht-Graviton-basierten EC2-Instances bestimmt die Instance das richtige Architektur-Image auf der Grundlage der Instance-Architektur, die das Image abruft.

Wie wählt man einen Basis-Image-URI aus

> **Note**
>
> Für Amazon-EMR-Versionen 6.9.0 und höher können Sie das Basis-Image aus der Amazon ECR Public Gallery abrufen, sodass Sie den Basis-Image-URI nicht wie in den Anweisungen auf dieser Seite beschrieben erstellen müssen. Das Container-Image-Tag für Ihr Basis-Image finden Sie auf der Seite mit den Versionshinweisen für die entsprechende Version von Amazon EMR in EKS.

Die Basis-Docker-Images die Sie auswählen können werden in der Amazon Elastic Container Registry (Amazon ECR) gespeichert. Die Image-URI folgt diesem Format: *ECR-registry-account*`.dkr.ecr.`*Region*`.amazonaws.com/spark/`*container-image-tag*, wie das folgende Beispiel zeigt.

```
895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-7.1.0:latest
```

Der Image-URI für interaktive Endpunkte folgt diesem Format: *ECR-registry-account*`.dkr.ecr.`*Region*`.amazonaws.com/notebook-spark/`*container-image-tag*, wie das folgende Beispiel zeigt. Sie müssen `notebook-spark` im Basis-Image den URI anstelle von `spark` verwenden.

```
895885662937.dkr.ecr.us-west-2.amazonaws.com/notebook-spark/emr-7.1.0:latest
```

Ebenso lautet der Image-URI für Nicht-Spark-python3-Images für interaktive Endpunkte *ECR-registry-account*.dkr.ecr.*Region*.amazonaws.com/notebook-python/*container-image-tag*. Der folgende Beispiel-URI ist korrekt formatiert:

```
895885662937.dkr.ecr.us-west-2.amazonaws.com/notebook-python/emr-7.1.0:latest
```

Das Container-Image-Tag für Ihr Basis-Image finden Sie auf der Seite mit den Versionshinweisen für die entsprechende Version von Amazon EMR in EKS.

Amazon-ECR-Registrierungskonten nach Regionen

Um eine hohe Netzwerklatenz zu vermeiden, rufen Sie ein Basis-Image von Ihrem nächstgelegenen Computer ab. AWS-Region Wählen Sie anhand der folgenden Tabelle das Amazon-ECR-Registrierungskonto aus, das der Region entspricht, aus der Sie das Image abrufen.

Regionen	Amazon-ECR-Registrierungskonten	
ap-northeast-1	059004520145	
ap-northeast-2	996579266876	
ap-south-1	235914868574	
ap-southeast-1	671219180197	
ap-southeast-2	038297999601	
ca-central-1	351826393999	
eu-central-1	107292555468	
eu-north-1	830386416364	
eu-west-1	483788554619	
eu-west-2	118780647275	
eu-west-3	307523725174	

Regionen	Amazon-ECR-Registrierungskonten	
sa-east-1	052806832358	
us-east-1	755674844232	
us-east-2	711395599931	
us-west-1	608033475327	
us-west-2	895885662937	

Überlegungen

Wenn Sie Docker-Images anpassen, können Sie die genaue Laufzeit für Ihren Auftrag detailliert auswählen. Folgen Sie diesen bewährten Methoden bei der Verwendung dieses Features:

- Sicherheit ist eine gemeinsame Verantwortung zwischen Ihnen AWS und Ihnen. Sie sind dafür verantwortlich, die Binärdateien, die Sie dem Image hinzufügen, mit Sicherheitspatches zu aktualisieren. Befolgen Sie Bewährte Methoden für Sicherheit in Amazon EMR in EKS, insbesondere Die neuesten Sicherheitsupdates für benutzerdefinierte Images erhalten und Das Prinzip der geringsten Berechtigung anwenden.
- Wenn Sie ein Basis-Image anpassen, müssen Sie den Docker-Benutzer `hadoop:hadoop` so ändern, dass die Aufträge nicht mit dem Root-Benutzer ausgeführt werden.
- Amazon EMR in EKS mountet zur Laufzeit Dateien zusätzlich zu den Konfigurationen für das Image, z. B. `spark-defaults.conf`. Um diese Konfigurationsdateien zu überschreiben, empfehlen wir, den `applicationOverrides`-Parameter während der Auftragsübermittlung zu verwenden und die Dateien im benutzerdefinierten Image nicht direkt zu ändern.
- Amazon EMR in EKS mountet bestimmte Ordner zur Laufzeit. Alle Änderungen, die Sie an diesen Ordnern vornehmen, sind im Container nicht verfügbar. Wenn Sie eine Anwendung oder deren Abhängigkeiten für benutzerdefinierte Images hinzufügen möchten, empfehlen wir Ihnen, ein Verzeichnis zu wählen, das nicht Teil der folgenden vordefinierten Pfade ist:
  - `/var/log/fluentd`
  - `/var/log/spark/user`
  - `/var/log/spark/apps`

  - `/mnt`
  - `/tmp`
  - `/home/hadoop`
- Sie können Ihr benutzerdefiniertes Image in ein beliebiges Docker-kompatibles Repository wie Amazon ECR, Docker Hub oder ein privates Unternehmens-Repository hochladen. Weitere Informationen zur Konfiguration der Amazon-EKS-Cluster-Authentifizierung mit dem ausgewählten Docker-Repository finden Sie unter Abrufen eines Images aus einer privaten Registrierung.

Ausführen von Flink-Aufträgen mit Amazon EMR in EKS

Amazon-EMR-Versionen 6.13.0 und höher unterstützen Amazon EMR in EKS mit Apache Flink oder dem Flink-Kubernetes-Operator als Modell für die Einreichung von Aufträgen für Amazon EMR in EKS. Mit Amazon EMR in EKS mit Apache Flink können Sie Flink-Anwendungen mit der Amazon-EMR-Release-Laufzeit auf Ihren eigenen Amazon-EKS-Clustern bereitstellen und verwalten. Sobald Sie den Flink-Kubernetes-Operator in Ihrem Amazon-EKS-Cluster bereitgestellt haben, können Sie Flink-Anwendungen direkt beim Operator einreichen. Der Operator verwaltet den Lebenszyklus von Flink-Anwendungen.

Themen

Flink-Kubernetes-Operator

Auf den folgenden Seiten wird beschrieben, wie Sie den Flink-Kubernetes-Operator einrichten und verwenden, um Flink-Aufträge mit Amazon EMR in EKS auszuführen.

Themen

Flink-Kubernetes-Operator für Amazon EMR in EKS einrichten

Führen Sie zum Einrichten des Flink-Kubernetes-Operator auf Amazon EKS die folgenden Schritte aus. Wenn Sie bereits für Amazon Web Services (AWS) registriert sind und Amazon EKS schon verwendet haben, müssen Sie nur wenige Schritte ausführen, um Amazon EMR in EKS nutzen zu können. Führen Sie zum Einrichten von Flink-Operator die folgenden Schritte auf Amazon EKS aus. Wenn Sie bereits eine der Voraussetzungen erfüllt haben, können Sie diese überspringen und mit der nächsten fortfahren.

- Installieren Sie das AWS CLI— Wenn Sie das bereits installiert haben AWS CLI, vergewissern Sie sich, dass Sie die neueste Version haben.
- eksctl installieren – eksctl ist ein Befehlszeilentool, das Sie für die Kommunikation mit Amazon EKS verwenden.
- Helm installieren – Der Helm-Paketmanager für Kubernetes unterstützt Sie bei der Installation und Verwaltung von Anwendungen in Ihrem Kubernetes-Cluster.
- Einen Amazon-EKS-Cluster einrichten – Folgen Sie den Schritten, um einen neuen Kubernetes-Cluster mit Knoten in Amazon EKS zu erstellen.
- Wählen Sie ein Amazon-EMR-Versionskennung (Version 6.13.0 oder höher) – der Flink-Kubernetes-Operator wird mit Amazon-EMR-Versionen 6.13.0 und höher unterstützt.
- Aktivieren Sie IAM-Rollen für Servicekonten (IRSA) auf dem Amazon-EKS-Cluster.
- Erstellen einer Aufgaben-Ausführungsrolle.
- Aktualisieren Sie die Vertrauensrichtlinie der Auftragausführungsrolle.
- Erstellen Sie eine Operator-Ausführungsrolle. Dieser Schritt ist optional. Sie können die gleiche Rolle für Flink-Aufträge und -Operator verwenden. Wenn Sie eine andere IAM-Rolle für Ihren Operator haben möchten, können Sie eine separate Rolle erstellen.
- Aktualisieren Sie die Vertrauensrichtlinie der Operatorausführungsrolle. Sie müssen explizit einen Vertrauensrichtlinieneintrag für die Rollen hinzufügen, die Sie für das Flink-Kubernetes-Operator-Servicekonto von Amazon EMR verwenden möchten. Sie können diesem Beispielformat folgen:

```
{
   "Version": "2012-10-17",
   "Statement": [
       {
           "Effect": "Allow",
           "Principal": {
               "Federated": "arn:aws:iam::ACCOUNT_ID:oidc-provider/OIDC_PROVIDER"
```

```
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringLike": {
                    "OIDC_PROVIDER:sub": "system:serviceaccount:NAMESPACE:emr-
containers-sa-flink-operator"
                }
            }
        }
    ]
}
```

Erste Schritte mit dem Flink-Kubernetes-Operator für Amazon EMR in EKS

Dieses Thema hilft Ihnen beim Einstieg in die Verwendung des Flink-Kubernetes-Operators in Amazon EKS, indem Sie eine Flink-Bereitstellung bereitstellen.

Den Operator installieren

Gehen Sie wie folgt vor, um den Kubernetes-Operator für Apache Flink zu installieren.

1. Sofern noch nicht geschehen, führen die Schritte unter the section called “Einrichtung” aus.
2. Installieren Sie den *Cert-Manager* (einmal pro Amazon-EKS-Cluster), um das Hinzufügen der Webhook-Komponente zu ermöglichen.

   ```
   kubectl apply -f https://github.com/cert-manager/cert-manager/releases/download/
   v1.12.0/cert-manager.yaml
   ```

3. Installieren Sie das Helm-Chart.

   ```
   export VERSION=7.1.0 # The Amazon EMR release version
   export NAMESPACE=The Kubernetes namespace to deploy the operator

   helm install flink-kubernetes-operator-demo \
   oci://public.ecr.aws/emr-on-eks/flink-kubernetes-operator \
   --version $VERSION \
   --namespace $NAMESPACE
   ```

   Beispielausgabe:

```
NAME: flink-kubernetes-operator-demo
LAST DEPLOYED: Tue May 31 17:38:56 2022
NAMESPACE: $NAMESPACE
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

4. Warten Sie, bis die Bereitstellung abgeschlossen ist, und überprüfen Sie die Installation des Diagramms.

```
kubectl wait deployment flink-kubernetes-operator-demo --namespace $NAMESPACE --for
 condition=Available=True --timeout=30s
```

5. Die Ausgabe sollte folgendermaßen aussehen, wenn die Bereitstellung abgeschlossen ist.

```
deployment.apps/flink-kubernetes-operator-demo condition met
```

6. Verwenden Sie den folgenden Befehl, um den bereitgestellten Operator zu sehen.

```
helm list --namespace $NAMESPACE
```

Im Folgenden wird eine Beispielausgabe gezeigt, bei der die Anwendungsversion `x.y.z-amzn-n` der Flink-Operator-Version für Ihre Amazon EMR in EKS-Version entsprechen würde. Weitere Informationen finden Sie unter Unterstützte Versionen für Amazon EMR in EKS mit Apache Flink.

```
NAME                            NAMESPACE    REVISION    UPDATED
               STATUS      CHART                                  APP VERSION

flink-kubernetes-operator-demo    $NAMESPACE   1           2023-02-22
 16:43:45.24148 -0500 EST    deployed    flink-kubernetes-operator-emr-7.1.0
 x.y.z-amzn-n
```

Führen Sie eine Flink-Anwendung aus

Mit Amazon EMR 6.13.0 und höher können Sie eine Flink-Anwendung mit dem Flink-Kubernetes-Operator im Anwendungsmodus in Amazon EMR in EKS ausführen. Mit Amazon EMR 6.15.0 und höher können Sie eine Flink-Anwendung auch im Sitzungsmodus ausführen. Auf dieser Seite

werden beide Methoden beschrieben, mit denen Sie eine Flink-Anwendung mit Amazon EMR in EKS ausführen können.

> **Note**
>
> Sie müssen einen Amazon-S3-Bucket erstellt haben, um die Hochverfügbarkeitsmetadaten zu speichern, wenn Sie Ihren Flink-Auftrag einreichen. Wenn Sie dieses Feature nicht verwenden möchten, können Sie sie deaktivieren. Sie ist standardmäßig aktiviert.

Voraussetzung – Bevor Sie eine Flink-Anwendung mit dem Flink-Kubernetes-Operator ausführen können, führen Sie die Schritte in the section called “Einrichtung” und the section called “Den Operator installieren” aus.

Application mode

Mit Amazon EMR 6.13.0 und höher können Sie eine Flink-Anwendung mit dem Flink-Kubernetes-Operator im Anwendungsmodus in Amazon EMR in EKS ausführen.

1. Erstellen Sie eine `FlinkDeployment`-Aufgabendefinitions-Datei `basic-example-app-cluster.yaml` mit dem folgenden Beispielinhalt:

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: basic-example-app-cluster
spec:
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "2"
    state.checkpoints.dir: CHECKPOINT_S3_STORAGE_PATH
    state.savepoints.dir: SAVEPOINT_S3_STORAGE_PATH
  flinkVersion: v1_17
  executionRoleArn: JOB_EXECUTION_ROLE_ARN
  emrReleaseLabel: "emr-6.13.0-flink-latest" // 6.13 or higher
  jobManager:
    storageDir: HIGH_AVAILABILITY_STORAGE_PATH
    resource:
      memory: "2048m"
      cpu: 1
  taskManager:
    resource:
      memory: "2048m"
```

```
    cpu: 1
  job:
    # if you have your job jar in S3 bucket you can use that path as well
    jarURI: local:///opt/flink/examples/streaming/StateMachineExample.jar
    parallelism: 2
    upgradeMode: savepoint
    savepointTriggerNonce: 0
  monitoringConfiguration:
    cloudWatchMonitoringConfiguration:
       logGroupName: LOG_GROUP_NAME
```

2. Senden Sie die Flink-Bereitstellung mit dem folgenden Befehl. Dadurch wird auch ein `FlinkDeployment`-Objekt mit dem `basic-example-app-cluster`-Namen erstellt.

```
kubectl create -f example.yaml -n <NAMESPACE>
```

3. Greifen Sie auf die Flink-Benutzeroberfläche zu.

```
kubectl port-forward deployments/basic-example-app-cluster 8081 -n NAMESPACE
```

4. Öffnen Sie `localhost:8081`, um Ihre Flink-Aufträge lokal anzusehen.
5. Bereinigen Sie den Auftrag. Denken Sie daran, die S3-Artefakte zu bereinigen, die für diesen Job erstellt wurden, z. B. Checkpointing-, Hochverfügbarkeits-, Savepointing-Metadaten und Protokolle. CloudWatch

Weitere Informationen zum Einreichen von Bewerbungen über den Flink-Kubernetes-Operator an Flink finden Sie unter Beispiele für Flink-Kubernetes-Operatoren im Ordner unter. `apache/flink-kubernetes-operator` GitHub

Session mode

Mit Amazon EMR 6.15.0 und höher können Sie eine Flink-Anwendung mit dem Flink-Kubernetes-Operator im Sitzungsmodus in Amazon EMR in EKS ausführen.

1. Erstellen Sie eine `FlinkDeployment`-Aufgabendefinitions-Datei `basic-example-session-cluster.yaml` mit dem folgenden Beispielinhalt:

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
```

```
  name: basic-example-session-cluster
spec:
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "2"
    state.checkpoints.dir: CHECKPOINT_S3_STORAGE_PATH
    state.savepoints.dir: SAVEPOINT_S3_STORAGE_PATH
  flinkVersion: v1_17
  executionRoleArn: JOB_EXECUTION_ROLE_ARN
  emrReleaseLabel: "emr-6.15.0-flink-latest"
  jobManager:
    storageDir: HIGH_AVAILABILITY_S3_STORAGE_PATH
    resource:
      memory: "2048m"
      cpu: 1
  taskManager:
    resource:
      memory: "2048m"
      cpu: 1
  monitoringConfiguration:
    s3MonitoringConfiguration:
       logUri:
    cloudWatchMonitoringConfiguration:
       logGroupName: LOG_GROUP_NAME
```

2. Senden Sie die Flink-Bereitstellung mit dem folgenden Befehl. Dadurch wird auch ein `FlinkDeployment`-Objekt mit dem `basic-example-session-cluster`-Namen erstellt.

```
kubectl create -f example.yaml -n NAMESPACE
```

3. Verwenden Sie den folgenden Befehl, um zu bestätigen, dass der Sitzungscluster `LIFECYCLE STABLE` ist:

```
kubectl get flinkdeployments.flink.apache.org basic-example-session-cluster -
n NAMESPACE
```

Die Ausgabe sollte ähnlich wie im folgenden Beispiel aussehen:

```
NAME                              JOB STATUS   LIFECYCLE STATE
basic-example-session-cluster                          STABLE
```

4. Erstellen Sie eine `FlinkSessionJob`-Aufgabendefinitions-Datei `basic-session-job.yaml` mit dem folgenden Beispielinhalt:

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkSessionJob
metadata:
  name: basic-session-job
spec:
  deploymentName: basic-session-deployment
  job:
    # If you have your job jar in an S3 bucket you can use that path.
    # To use jar in S3 bucket, set
    # OPERATOR_EXECUTION_ROLE_ARN (--set emrContainers.operatorExecutionRoleArn=
$OPERATOR_EXECUTION_ROLE_ARN)
    # when you install Spark operator
    jarURI: https://repo1.maven.org/maven2/org/apache/flink/flink-examples-
streaming_2.12/1.16.1/flink-examples-streaming_2.12-1.16.1-TopSpeedWindowing.jar
    parallelism: 2
    upgradeMode: stateless
```

5. Reichen Sie den Flink-Sitzungsauftrag mit dem folgenden Befehl ein. Dadurch wird auch ein `FlinkSessionJob`-Objekt `basic-session-job` erstellt.

```
kubectl apply -f basic-session-job.yaml -n $NAMESPACE
```

6. Verwenden Sie den folgenden Befehl, um zu bestätigen, dass der Sitzungscluster `LIFECYCLE STABLE` und dass der `JOB STATUS RUNNING` ist:

```
kubectl get flinkdeployments.flink.apache.org basic-example-session-cluster -
n NAMESPACE
```

Die Ausgabe sollte ähnlich wie im folgenden Beispiel aussehen:

```
NAME                              JOB STATUS   LIFECYCLE STATE
basic-example-session-cluster     RUNNING      STABLE
```

7. Greifen Sie auf die Flink-Benutzeroberfläche zu.

```
kubectl port-forward deployments/basic-example-session-cluster 8081 -n NAMESPACE
```

8. Öffnen Sie `localhost:8081`, um Ihre Flink-Aufträge lokal anzusehen.

9. Bereinigen Sie den Auftrag. Denken Sie daran, die S3-Artefakte zu bereinigen, die für diesen Job erstellt wurden, wie Checkpointing-, Hochverfügbarkeits-, Savepointing-Metadaten und Logs. CloudWatch

Sicherheit

RBAC

Um den Operator bereitzustellen und Flink-Aufträge auszuführen, müssen wir zwei Kubernetes-Rollen erstellen: eine Operator- und eine Auftrag-Rolle. Amazon EMR erstellt die beiden Rollen standardmäßig, wenn Sie den Operator installieren.

Rolle des Operators

Wir verwenden die Operator-Rolle, um den Job und andere Ressourcen, wie Dienste, JobManager für jeden Flink-Job zu verwalten, `flinkdeployments` zu erstellen und zu verwalten.

Der Standardname der Operatorrolle lautet `emr-containers-sa-flink-operator` und erfordert die folgenden Berechtigungen.

```
rules:
- apiGroups:
  - ""
  resources:
  - pods
  - services
  - events
  - configmaps
  - secrets
  - serviceaccounts
  verbs:
  - '*'
- apiGroups:
  - rbac.authorization.k8s.io
  resources:
  - roles
  - rolebindings
  verbs:
  - '*'
- apiGroups:
  - apps
```

```
  resources:
  - deployments
  - deployments/finalizers
  - replicasets
  verbs:
  - '*'
- apiGroups:
  - extensions
  resources:
  - deployments
  - ingresses
  verbs:
  - '*'
- apiGroups:
  - flink.apache.org
  resources:
  - flinkdeployments
  - flinkdeployments/status
  - flinksessionjobs
  - flinksessionjobs/status
  verbs:
  - '*'
- apiGroups:
  - networking.k8s.io
  resources:
  - ingresses
  verbs:
  - '*'
- apiGroups:
  - coordination.k8s.io
  resources:
  - leases
  verbs:
  - '*'
```

Aufgabenrolle

Der JobManager verwendet die Job-Rolle zur Erstellung TaskManagers und Verwaltung sowie ConfigMaps für jeden Job.

```
rules:
- apiGroups:
  - ""
```

```
  resources:
  - pods
  - configmaps
  verbs:
  - '*'
- apiGroups:
  - apps
  resources:
  - deployments
  - deployments/finalizers
  verbs:
  - '*'
```

Flink-Kubernetes-Operator für Amazon EMR in EKS deinstallieren

Gehen Sie wie folgt vor, um den Flink-Kubernetes-Operator zu deinstallieren.

1. Löschen Sie den Operator.

```
helm uninstall flink-kubernetes-operator-demo -n <NAMESPACE>
```

2. Löschen Sie Kubernetes-Ressourcen, die Helm nicht deinstalliert.

```
kubectl delete serviceaccounts, roles, rolebindings -l emr-
containers.amazonaws.com/component=flink.operator --namespace <namespace>
kubectl delete crd flinkdeployments.flink.apache.org
 flinksessionjobs.flink.apache.org
```

3. (Optional) Löschen Sie den Cert-Manager.

```
kubectl delete -f https://github.com/jetstack/cert-manager/releases/download/
v1.12.0/cert-manager.yaml
```

Native Kubernetes

Amazon-EMR-Versionen 6.13.0 und höher unterstützen Flink Native Kubernetes als Befehlszeilentool, mit dem Sie Flink-Anwendungen an einen Amazon EMR in EKS-Cluster senden und ausführen können.

Themen

Einrichten von Flink Native Kubernetes für Amazon EMR in EKS

Führen Sie die folgenden Aufgaben aus, um eine Anwendung mit der Flink CLI in Amazon EMR in EKS ausführen zu können. Wenn Sie bereits für Amazon Web Services (AWS) registriert sind und Amazon EKS schon verwendet haben, müssen Sie nur wenige Schritte ausführen, um Amazon EMR in EKS nutzen zu können. Wenn Sie bereits eine der Voraussetzungen erfüllt haben, können Sie diese überspringen und mit der nächsten fortfahren.

- Installieren Sie das AWS CLI – Wenn Sie AWS CLI bereits installiert haben, vergewissern Sie sich, dass Sie über die neueste Version verfügen.
- Einen Amazon-EKS-Cluster einrichten – Folgen Sie den Schritten, um einen neuen Kubernetes-Cluster mit Knoten in Amazon EKS zu erstellen.
- Wählen Sie eine Amazon-EMR-Basis-Image-URI (Version 6.13.0 oder höher) aus – der Flink-Kubernetes-Befehl wird mit Amazon-EMR-Versionen 6.13.0 und höher unterstützt.
- Vergewissern Sie sich, dass das JobManager Servicekonto über die entsprechenden Berechtigungen zum Erstellen und Überwachen TaskManager von Pods verfügt. Weitere Informationen finden Sie unter Sicherheitsanforderungen für Flink- JobManager Servicekonten für native Kubernetes.
- Richten Sie Ihr lokales AWS-Anmeldeinformationsprofil ein.
- Erstellen oder aktualisieren Sie eine kubeconfig-Datei für einen Amazon-EKS-Cluster, auf dem Sie die Flink-Anwendungen ausführen möchten.

Erste Schritte mit Native Kubernetes für Amazon EMR in EKS

Eine Flink-Anwendung ausführen

Amazon EMR 6.13.0 und höher unterstützt Flink Native Kubernetes für die Ausführung von Flink-Anwendungen auf einem Amazon-EKS-Cluster. Gehen Sie folgendermaßen vor, um eine Flink-Anwendung auszuführen:

1. Bevor Sie eine Flink-Anwendung mit dem Befehl Flink Native Kubernetes ausführen können, führen Sie die Schritte unter the section called “Einrichtung” aus.

2. Laden Sie Flink herunter und installieren Sie es.
3. Legen Sie die Werte der folgenden Umgebungsvariablen fest.

```
#Export the FLINK_HOME environment variable to your local installation of Flink
export FLINK_HOME=/usr/local/bin/flink #Will vary depending on your installation
export NAMESPACE=flink
export CLUSTER_ID=flink-application-cluster
export IMAGE=<123456789012.dkr.ecr.sample-AWS-Region-.amazonaws.com/flink/
emr-6.13.0-flink:latest>
export FLINK_SERVICE_ACCOUNT=emr-containers-sa-flink
export FLINK_CLUSTER_ROLE_BINDING=emr-containers-crb-flink
```

4. Erstellen Sie ein Servicekonto, um Kubernetes-Ressourcen zu verwalten.

```
kubectl create serviceaccount $FLINK_SERVICE_ACCOUNT -n $NAMESPACE
kubectl create clusterrolebinding $FLINK_CLUSTER_ROLE_BINDING --clusterrole=edit --
serviceaccount=$NAMESPACE:$FLINK_SERVICE_ACCOUNT
```

5. Führen Sie den `run-application`-CLI-Befehl aus.

```
$FLINK_HOME/bin/flink run-application \
    --target kubernetes-application \
    -Dkubernetes.namespace=$NAMESPACE \
    -Dkubernetes.cluster-id=$CLUSTER_ID \
    -Dkubernetes.container.image.ref=$IMAGE \
    -Dkubernetes.service-account=$FLINK_SERVICE_ACCOUNT \
    local:///opt/flink/examples/streaming/Iteration.jar
2022-12-29 21:13:06,947 INFO  org.apache.flink.kubernetes.utils.KubernetesUtils
           [] - Kubernetes deployment requires a fixed port. Configuration
 blob.server.port will be set to 6124
2022-12-29 21:13:06,948 INFO  org.apache.flink.kubernetes.utils.KubernetesUtils
           [] - Kubernetes deployment requires a fixed port. Configuration
 taskmanager.rpc.port will be set to 6122
2022-12-29 21:13:07,861 WARN
 org.apache.flink.kubernetes.KubernetesClusterDescriptor      [] - Please note that
 Flink client operations(e.g. cancel, list, stop, savepoint, etc.) won't work from
 outside the Kubernetes cluster since 'kubernetes.rest-service.exposed.type' has
 been set to ClusterIP.
2022-12-29 21:13:07,868 INFO
 org.apache.flink.kubernetes.KubernetesClusterDescriptor      [] - Create flink
 application cluster flink-application-cluster successfully, JobManager Web
 Interface: http://flink-application-cluster-rest.flink:8081
```

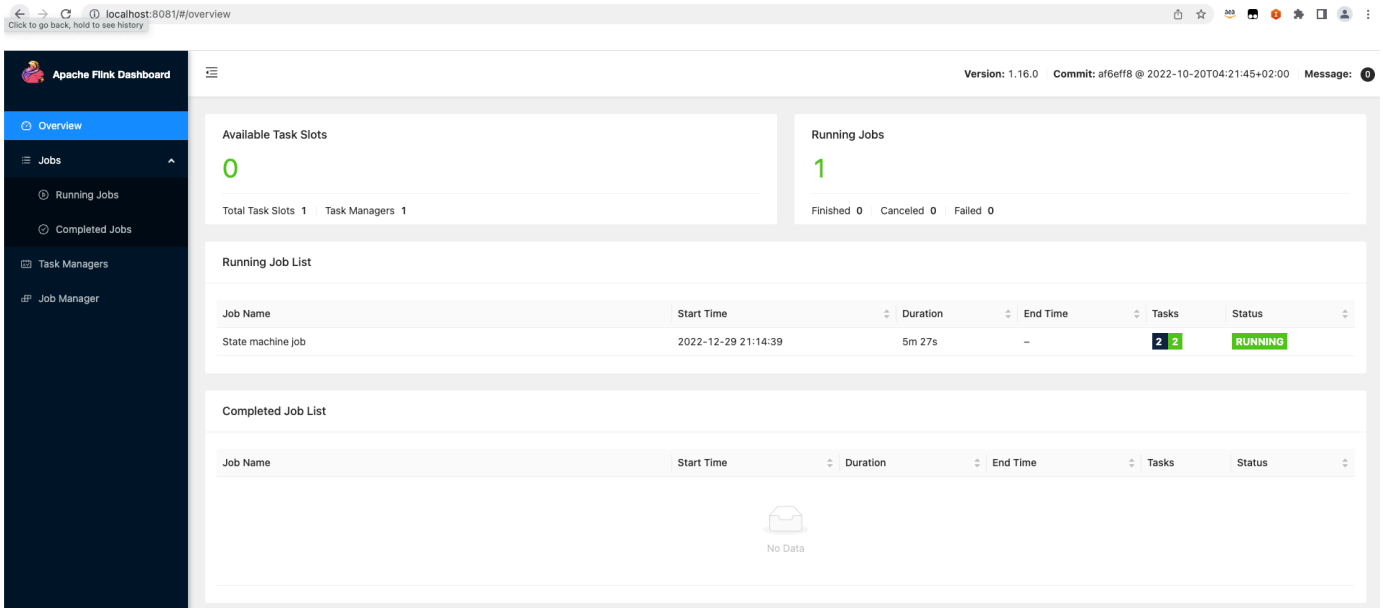
6. Untersuchen Sie die erstellten Kubernetes-Ressourcen.

```
kubectl get all -n <namespace>
NAME READY STATUS RESTARTS AGE
pod/flink-application-cluster-546687cb47-w2p2z 1/1 Running 0 3m37s
pod/flink-application-cluster-taskmanager-1-1 1/1 Running 0 3m24s

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
service/flink-application-cluster ClusterIP None <none> 6123/TCP,6124/TCP 3m38s
service/flink-application-cluster-rest ClusterIP 10.100.132.158 <none> 8081/TCP
 3m38s

NAME READY UP-TO-DATE AVAILABLE AGE
deployment.apps/flink-application-cluster 1/1 1 1 3m38s

NAME DESIRED CURRENT READY AGE
replicaset.apps/flink-application-cluster-546687cb47 1 1 1 3m38s
```

7. Portweiterleitung auf 8 081.

```
kubectl port-forward service/flink-application-cluster-rest 8081 -n <namespace>
Forwarding from 127.0.0.1:8081 -> 8081
```

8. Greifen Sie lokal auf die Flink-Benutzeroberfläche zu.



9. Löschen Sie die Flink-Anwendung.

```
kubectl delete deployment.apps/flink-application-cluster -n <namespace>
deployment.apps "flink-application-cluster" deleted
```

Weitere Informationen zum Senden von Anwendungen an Flink finden Sie unter Native Kubernetes in der Apache-Flink-Dokumentation.

Sicherheitsanforderungen für Flink- JobManager Servicekonten für native Kubernetes

Der Flink- JobManager Pod verwendet ein Kubernetes-Servicekonto, um auf den Kubernetes-API-Server zuzugreifen, um TaskManager Pods zu erstellen und zu beobachten. JobManager -Servicekonto muss über die entsprechenden Berechtigungen verfügen, um TaskManager Pods zu erstellen/zu löschen und dem -Standort TaskManager zu erlauben ConfigMaps , die Adresse von JobManager und ResourceManager in Ihrem Cluster abzurufen.

Es gelten die folgenden Regeln für dieses Servicekonto.

```
rules:
- apiGroups:
  - ""
  resources:
  - pods
  verbs:
  - "*"
- apiGroups:
  - ""
  resources:
  - services
  verbs:
  - "*"
- apiGroups:
  - ""
  resources:
  - configmaps
  verbs:
  - "*"
- apiGroups:
  - "apps"
```

```
  resources:
  - deployments
  verbs:
  - "*"
```

Anpassen von Docker-Images für Amazon EMR auf EKS mit Apache Flink

In den folgenden Abschnitten wird beschrieben, wie Sie Docker-Images für Amazon EMR auf EKS anpassen.

Themen

- Anpassen von Docker-Images für Flink und FluentD

Anpassen von Docker-Images für Flink und FluentD

Gehen Sie wie folgt vor, um Docker-Images für Amazon EMR auf EKS mit Apache Flink- oder FluentD-Images anzupassen.

Themen

- Voraussetzungen
- Schritt 1: Rufen Sie ein Basis-Image aus der Amazon Elastic Container Registry ab
- Schritt 2: Ein Basis-Image anpassen
- Schritt 3: Veröffentlichen Sie Ihr benutzerdefiniertes Bild
- Schritt 4: Senden Sie einen Flink-Workload in Amazon EMR mithilfe eines benutzerdefinierten Images

Voraussetzungen

Bevor Sie Ihr Docker-Image anpassen, stellen Sie sicher, dass Sie die folgenden Voraussetzungen erfüllt haben:

- Die Schritte zum Einrichten des Flink Kubernetes-Operators für Amazon EMR auf EKS wurden abgeschlossen.
- Docker in Ihrer Umgebung installiert. Weitere Informationen finden Sie unter Docker holen.

Schritt 1: Rufen Sie ein Basis-Image aus der Amazon Elastic Container Registry ab

Das Basis-Image enthält die Amazon EMR-Laufzeit und Konnektoren, die Sie für den Zugriff auf andere AWS-Services benötigen. Wenn Sie Amazon EMR auf EKS mit Flink Version 6.14.0 oder höher verwenden, können Sie die Basis-Images aus der Amazon ECR Public Gallery abrufen. Durchsuchen Sie die Galerie nach dem Image-Link und laden Sie das Image in Ihren lokalen Workspace. Für die Version Amazon EMR 6.14.0 gibt der folgende `docker pull` Befehl beispielsweise das neueste Standard-Basis-Image zurück. Ersetzen Sie es `emr-6.14.0:latest` durch die gewünschte Release-Version.

```
docker pull public.ecr.aws/emr-on-eks/flink/emr-6.14.0-flink:latest
```

Im Folgenden finden Sie Links zum Flink-Galeriebild und zum Fluentd-Galeriebild:

- emr-on-eks/flink/emr-6.14.0-flink
- emr-on-eks/fluentd/emr-6.14.0 (

Schritt 2: Ein Basis-Image anpassen

In den folgenden Schritten wird beschrieben, wie Sie das Basis-Image, das Sie aus Amazon ECR abgerufen haben, anpassen können.

1. Erstellen Sie ein neues `Dockerfile` in Ihrem lokalen Workspace.
2. Bearbeiten Sie den `Dockerfile` und fügen Sie den folgenden Inhalt hinzu. Dabei wird das Container-Image `Dockerfile` verwendet, aus dem Sie abgerufen habenpublic.ecr.aws/emr-on-eks/flink/emr-7.1.0-flink:latest.

   ```
   FROM public.ecr.aws/emr-on-eks/flink/emr-7.1.0-flink:latest
   USER root
   ### Add customization commands here ####
   USER hadoop:hadoop
   ```

   Verwenden Sie die folgende Konfiguration, wenn Sie verwendenFluentd.

   ```
   FROM public.ecr.aws/emr-on-eks/fluentd/emr-7.1.0:latest
   USER root
   ### Add customization commands here ####
   USER hadoop:hadoop
   ```

3. Fügen Sie Befehle in `Dockerfile` hinzu, um das Basis-Image anzupassen. Der folgende Befehl zeigt, wie Python-Bibliotheken installiert werden.

```
FROM public.ecr.aws/emr-on-eks/flink/emr-7.1.0-flink:latest
USER root
RUN pip3 install --upgrade boto3 pandas numpy // For python 3
USER hadoop:hadoop
```

4. Führen Sie in demselben Verzeichnis, in dem Sie es erstellt habenDockerFile, den folgenden Befehl aus, um das Docker-Image zu erstellen. Das Feld, das Sie hinter dem `-t` Flag angeben, ist Ihr benutzerdefinierter Name für das Image.

```
docker build -t <YOUR_ACCOUNT_ID>.dkr.ecr.<YOUR_ECR_REGION>.amazonaws.com/
<ECR_REPO>:<ECR_TAG>
```

Schritt 3: Veröffentlichen Sie Ihr benutzerdefiniertes Bild

Sie können das neue Docker-Image jetzt in Ihrer Amazon ECR-Registrierung veröffentlichen.

1. Führen Sie den folgenden Befehl aus, um ein Amazon ECR-Repository zum Speichern Ihres Docker-Images zu erstellen. Geben Sie einen Namen für Ihr Repository ein, z. B. `emr_custom_repo.` Weitere Informationen finden Sie unter Erstellen eines Repositorys im Amazon Elastic Container Registry User Guide.

```
aws ecr create-repository \
       --repository-name emr_custom_repo \
       --image-scanning-configuration scanOnPush=true \
       --region <AWS_REGION>
```

2. Führen Sie den folgenden Befehl aus, um Ihre Ressourcen zu löschen. Weitere Informationen finden Sie unter Authentifizieren Sie sich bei Ihrer Standardregistrierung im Amazon Elastic Container Registry User Guide.

```
aws ecr get-login-password --region <AWS_REGION> | docker login --username AWS --
password-stdin <AWS_ACCOUNT_ID>.dkr.ecr.<YOUR_ECR_REGION>.amazonaws.com
```

3. Übertragen Sie das Image per Push. Weitere Informationen finden Sie unter Ein Bild an Amazon ECR übertragen im Amazon Elastic Container Registry User Guide.

```
docker push <YOUR_ACCOUNT_ID>.dkr.ecr.<YOUR_ECR_REGION>.amazonaws.com/
<ECR_REPO>:<ECR_TAG>
```

Schritt 4: Senden Sie einen Flink-Workload in Amazon EMR mithilfe eines benutzerdefinierten Images

Nehmen Sie die folgenden Änderungen an Ihrer `FlinkDeployment` Spezifikation vor, um ein benutzerdefiniertes Bild zu verwenden. Geben Sie dazu Ihr eigenes Bild in die `spec.image` Zeile Ihrer Bereitstellungsspezifikation ein.

```
apiVersion: flink.apache.org/v1beta1
   kind: FlinkDeployment
   metadata:
     name: basic-example
   spec:
     flinkVersion: v1_18
     image: <YOUR_ACCOUNT_ID>.dkr.ecr.<YOUR_ECR_REGION>.amazonaws.com/
<ECR_REPO>:<ECR_TAG>
     imagePullPolicy: Always
     flinkConfiguration:
           taskmanager.numberOfTaskSlots: "1"
```

Um ein benutzerdefiniertes Image für Ihren Fluentd-Job zu verwenden, geben Sie Ihr eigenes Bild in die `monitoringConfiguration.image` Zeile Ihrer Bereitstellungsspezifikation ein.

```
  monitoringConfiguration:
       image: <YOUR_ACCOUNT_ID>.dkr.ecr.<YOUR_ECR_REGION>.amazonaws.com/
<ECR_REPO>:<ECR_TAG>
       cloudWatchMonitoringConfiguration:
         logGroupName: flink-log-group
         logStreamNamePrefix: custom-fluentd
```

Überwachung von Flink-Kubernetes-Operator- und Flink-Aufträgen

In diesem Abschnitt werden verschiedene Möglichkeiten beschrieben, wie Sie Ihre Flink-Aufträge mit Amazon EMR in EKS überwachen können.

Themen

- Verwenden von Amazon Managed Service für Prometheus zur Überwachung von Flink-Aufträge
- Verwenden der Flink-Benutzeroberfläche zur Überwachung von Flink-Aufträgen
- Verwendung der Überwachungskonfiguration zur Überwachung von Flink-Kubernetes-Operator- und Flink-Aufträge

Verwenden von Amazon Managed Service für Prometheus zur Überwachung von Flink-Aufträge

Sie können Apache Flink in Amazon Managed Service für Prometheus (Verwaltungsportal) integrieren. Amazon Managed Service für Prometheus unterstützt die Erfassung von Metriken von Amazon Managed Service for Prometheus Servern in Clustern, die auf Amazon EKS ausgeführt werden. Amazon Managed Service für Prometheus arbeitet mit einem Prometheus-Server zusammen, der bereits auf Ihrem Amazon-EKS-Cluster läuft. Beim Ausführen der Integration von Amazon Managed Service für Prometheus mit Amazon EMR wird der Flink-Operator automatisch einen Prometheus-Server für die Integration mit Amazon Managed Service für Prometheus bereitstellen und konfigurieren.

1. Einen Workspace von Amazon Managed Service für Prometheus erstellen. Dieser Workspace dient als Aufnahme-Endpunkt. Sie benötigen die Remote-Write-URL später.
2. IAM-Rollen für Servicekonten einrichten.

   Verwenden Sie für diese Onboarding-Methode IAM-Rollen für die Servicekonten im Amazon-EKS-Cluster, in dem der Prometheus-Server läuft. Diese Rollen werden als Servicerollen bezeichnet.

   Wenn Sie die Rollen noch nicht haben, richten Sie Servicerollen für die Erfassung von Metriken aus Amazon-EKS-Clustern ein.

   Bevor Sie fortfahren, erstellen Sie eine IAM-Rolle namens `amp-iamproxy-ingest-role`.
3. Installieren Sie Amazon-EMR-Flink-Operator mit Amazon Managed Service für Prometheus.

Da Sie nun über einen Workspace von Amazon Managed Service für Prometheus, eine dedizierte IAM-Rolle für Amazon Managed Service für Prometheus und die erforderlichen Berechtigungen verfügen, können Sie den Amazon-EMR-Flink-Operator installieren.

Erstellen Sie eine Datei `enable-amp.yaml`. Mit dieser Datei können Sie eine benutzerdefinierte Konfiguration verwenden, um die Einstellungen von Amazon Managed Service for Prometheus zu überschreiben. Stellen Sie sicher, dass Sie Ihre eigenen Rollen verwenden.

```
kube-prometheus-stack:
    prometheus:
    serviceAccount:
        create: true
        name: "amp-iamproxy-ingest-service-account"
        annotations:
            eks.amazonaws.com/role-arn: "arn:aws:iam::<AWS_ACCOUNT_ID>:role/amp-
iamproxy-ingest-role"
    remoteWrite:
        - url: <AMAZON_MANAGED_PROMETHEUS_REMOTE_WRITE_URL>
        sigv4:
            region: <AWS_REGION>
        queueConfig:
            maxSamplesPerSend: 1000
            maxShards: 200
            capacity: 2500
```

Verwenden Sie den Helm Install --set-Befehl, um Überschreibungen an das `flink-kubernetes-operator`-Diagramm zu übergeben.

```
helm upgrade -n <namespace> flink-kubernetes-operator \
   oci://public.ecr.aws/emr-on-eks/flink-kubernetes-operator \
   --set prometheus.enabled=true
   -f enable-amp.yaml
```

Dieser Befehl installiert automatisch einen Prometheus-Reporter im -Operator auf Port 9999. In Any Future wird `FlinkDeployment` auch ein `metrics` Port auf 9249 verfügbar gemacht.

- Die Metriken für Flink-Operatoren werden in Prometheus unter der Kennung `flink_k8soperator_` angezeigt.
- Die Metriken für Flink-Task-Manager werden in Prometheus unter der Kennung `flink_taskmanager_` angezeigt.
- Die Metriken für Flink-Aufgaben-Manager werden in Prometheus unter der Kennung `flink_jobmanager_` angezeigt.

Verwenden der Flink-Benutzeroberfläche zur Überwachung von Flink-Aufträgen

Verwenden Sie das Flink-Web-Dashboard, um den Zustand und die Leistung einer laufenden Flink-Anwendung zu überwachen. Dieses Dashboard enthält Informationen über den Status des Auftrags, die Anzahl der TaskManagerssowie die Metriken und Protokolle für den Auftrag. Außerdem können Sie damit die Konfiguration des Flink-Aufträge anzeigen und ändern und mit dem Flink-Cluster interagieren, um Aufträge einzureichen oder abzubrechen.

So greifen Sie auf das Flink-Web-Dashboard für eine laufende Flink-Anwendung auf Kubernetes zu:

1. Verwenden Sie den `kubectl port-forward` Befehl , um einen lokalen Port an den Port weiterzuleiten, auf dem das Flink-Web-Dashboard in den TaskManager Pods der Flink-Anwendung ausgeführt wird. Standardmäßig ist dies der Port 8081. Ersetzen Sie *deployment-name* durch den Namen der Flink-Anwendungsbereitstellung von oben.

   ```
   kubectl get deployments -n namespace
   ```

   Beispielausgabe:

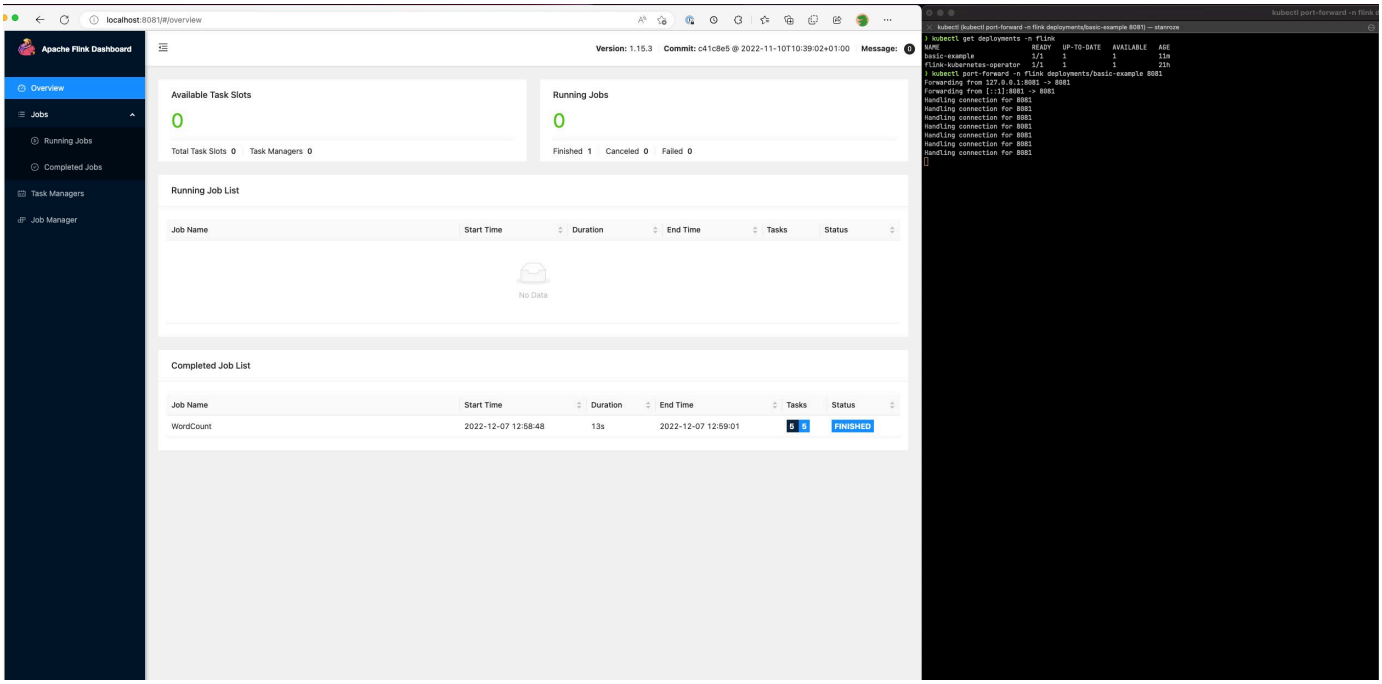
   ```
   kubectl get deployments -n flink-namespace
   NAME                        READY   UP-TO-DATE   AVAILABLE   AGE
   basic-example               1/1       1            1           11m
   flink-kubernetes-operator   1/1       1            1           21h
   ```

   ```
   kubectl port-forward deployments/deployment-name 8081 -n namespace
   ```

2. *Wenn Sie lokal einen anderen Port verwenden möchten, verwenden Sie den Parameter local-port:8081.*

   ```
   kubectl port-forward -n flink deployments/basic-example 8080:8081
   ```

3. Navigieren Sie in einem Webbrowser zu `http://localhost:8081` (oder `http://localhost:`*local-port* falls Sie einen benutzerdefinierten lokalen Port verwendet haben), um auf das Flink-Web-Dashboard zuzugreifen. Dieses Dashboard zeigt Informationen über die laufende Flink-Anwendung an, z. B. den Status des Auftrags TaskManagers, die Anzahl der sowie die Metriken und Protokolle für den Auftrag.



Verwendung der Überwachungskonfiguration zur Überwachung von Flink-Kubernetes-Operator- und Flink-Aufträge

Mit der Überwachungskonfiguration können Sie die Protokollarchivierung Ihrer Flink-Anwendung und -Operator-Protokolle auf einfache Weise in S3 und/oder einrichten CloudWatch (Sie können eine oder beide auswählen). Dadurch wird Ihren JobManager und TaskManager Pods ein FluentD-Sidecar hinzugefügt und anschließend die Protokolle dieser Komponenten an Ihre konfigurierten Senken weitergeleitet.

> **Note**
>
> Sie müssen IAM-Rollen für das Servicekonto für Ihren Flink-Operator und Ihren Flink-Auftrag (Servicekonten) einrichten, um dieses Feature nutzen zu können, da sie die Interaktion mit anderen AWS-Services erfordert. Sie müssen dies mithilfe von IRSA in Flink-Kubernetes-Operator für Amazon EMR in EKS einrichten einrichten.

Protokolle der Flink-Anwendung

Sie können diese Konfiguration folgendermaßen definieren.

```
apiVersion: flink.apache.org/v1beta1
```

```
kind: FlinkDeployment
metadata:
  name: basic-example
spec:
  image: FLINK IMAGE TAG
  imagePullPolicy: Always
  flinkVersion: v1_17
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "2"
  executionRoleArn: JOB EXECUTION ROLE
  jobManager:
    resource:
      memory: "2048m"
      cpu: 1
  taskManager:
    resource:
      memory: "2048m"
      cpu: 1
  job:
    jarURI: local:///opt/flink/examples/streaming/StateMachineExample.jar
  monitoringConfiguration:
    s3MonitoringConfiguration:
      logUri: S3 BUCKET
    cloudWatchMonitoringConfiguration:
      logGroupName: LOG GROUP NAME
      logStreamNamePrefix: LOG GROUP STREAM PREFIX
    sideCarResources:
      limits:
        cpuLimit: 500m
        memoryLimit: 250Mi
    containerLogRotationConfiguration:
        rotationSize: 2GB
        maxFilesToKeep: 10
```

Im Folgenden finden Sie Konfigurationsoptionen.

- `s3MonitoringConfiguration` – Konfigurationsschlüssel zum Einrichten der Weiterleitung an S3
  - `logUri` (erforderlich) – der S3-Bucket-Pfad, in dem Sie Ihre Protokolle speichern möchten.
  - Der Pfad auf S3 sieht nach dem Hochladen der Protokolle wie folgt aus.
    - Keine Protokollrotation aktiviert:

```
s3://${logUri}/${POD NAME}/STDOUT or STDERR.gz
```

- Die Protokollrotation ist aktiviert. Sie können sowohl eine rotierte Datei als auch eine aktuelle Datei (eine Datei ohne Datumsstempel) verwenden.

```
s3://${logUri}/${POD NAME}/STDOUT or STDERR.gz
```

Das folgende Format ist eine aufsteigende Zahl.

```
s3://${logUri}/${POD NAME}/stdout_YYYYMMDD_index.gz
```

- Die folgenden IAM-Berechtigungen sind erforderlich. um diese Weiterleitung zu nutzen.

```
{
    "Effect": "Allow",
    "Action": [
        "s3:PutObject"
    ],
    "Resource": [
       "S3_BUCKET_URI/*",
       "S3_BUCKET_URI"
    ]
}
```

- `cloudWatchMonitoringConfiguration` – Konfigurationsschlüssel zum Einrichten der Weiterleitung an CloudWatch.
  - `logGroupName` (erforderlich) – Name der CloudWatch Protokollgruppe, an die Sie Protokolle senden möchten (erstellt die Gruppe automatisch, falls sie nicht vorhanden ist).
  - `logStreamNamePrefix` (optional) – Name des Protokollstreams, an den Sie Protokolle senden möchten. Der Standardwert ist eine leere Zeichenfolge. Das Format lautet folgendermaßen:

```
${logStreamNamePrefix}/${POD NAME}/STDOUT or STDERR
```

  - Die folgenden IAM-Berechtigungen sind erforderlich. um diese Weiterleitung zu nutzen.

```
{
    "Effect": "Allow",
    "Action": [
        "logs:CreateLogStream",
```

```
            "logs:CreateLogGroup",
            "logs:PutLogEvents"
        ],
        "Resource": [
            "arn:aws:logs:REGION:ACCOUNT-ID:log-group:{YOUR_LOG_GROUP_NAME}:*",
            "arn:aws:logs:REGION:ACCOUNT-ID:log-group:{YOUR_LOG_GROUP_NAME}"
        ]
    }
```

- `sideCarResources` (optional) – der Konfigurationsschlüssel zum Festlegen von Ressourcenlimits für den gestarteten Fluentbit-Sidecar-Container.
  - `memoryLimit` (Optional) – Der Standardwert ist 512Mi. Passen Sie ihn entsprechend Ihren Anforderungen an.
  - `cpuLimit` (optional) – Diese Option hat keinen Standard. Passen Sie ihn entsprechend Ihren Anforderungen an.
- `containerLogRotationConfiguration` (optional) – steuert das Rotationsverhalten des Container-Protokolls. Sie ist standardmäßig aktiviert.
  - `rotationSize` (erforderlich) – gibt die Dateigröße für die Protokollrotation an. Der Bereich der möglichen Werte ist von 2 KB bis 2 GB. Die numerische Einheit des rotationSize-Parameters wird als Ganzzahl übergeben. Da Dezimalwerte nicht unterstützt werden, können Sie mit dem Wert 1 500 MB beispielsweise eine Rotationsgröße von 1,5 GB angeben. Der Standardwert ist 2 GB.
  - `maxFilesToKeep` (erforderlich) – gibt die maximale Anzahl von Dateien an, die nach der Rotation im Container aufbewahrt werden sollen. Der kleinste Wert ist 1 und der größte Wert ist 50. Der Standardwert ist 10.

Protokolle des Flink-Operators

Wir können auch die Protokollarchivierung für den Operator aktivieren, indem wir die folgenden Optionen in der `values.yaml`-Datei in Ihrer Helm-Chart-Installation verwenden. Sie können S3 CloudWatchoder beides aktivieren.

```
monitoringConfiguration:
  s3MonitoringConfiguration:
    logUri: "S3-BUCKET"
    totalFileSize: "1G"
    uploadTimeout: "1m"
  cloudWatchMonitoringConfiguration:
```

```
    logGroupName: "flink-log-group"
    logStreamNamePrefix: "example-job-prefix-test-2"
  sideCarResources:
    limits:
      cpuLimit: 1
      memoryLimit: 800Mi
  memoryBufferLimit: 700M
```

Im Folgenden sind die verfügbaren Konfigurationsoptionen unter `monitoringConfiguration` aufgeführt.

- `s3MonitoringConfiguration` – Stellen Sie diese Option ein, um auf S3 zu archivieren.
- `logUri` (erforderlich) – Der S3-Bucket-Pfad, in dem Sie Ihre Protokolle speichern möchten.
- Im Folgenden finden Sie Formate dafür, wie die S3-Bucket-Pfade nach dem Hochladen der Protokolle aussehen könnten.
  - Keine Protokollrotation aktiviert.

    ```
    s3://${logUri}/${POD NAME}/OPERATOR or WEBHOOK/STDOUT or STDERR.gz
    ```

  - Die Protokollrotation ist aktiviert. Sie können sowohl eine rotierte Datei als auch eine aktuelle Datei (eine Datei ohne Datumsstempel) verwenden.

    ```
    s3://${logUri}/${POD NAME}/OPERATOR or WEBHOOK/STDOUT or STDERR.gz
    ```

    Der folgende Formatindex ist eine aufsteigende Zahl.

    ```
    s3://${logUri}/${POD NAME}/OPERATOR or WEBHOOK/stdout_YYYYMMDD_index.gz
    ```

- `cloudWatchMonitoringConfiguration` – der Konfigurationsschlüssel, um die Weiterleitung an einzurichten CloudWatch.
  - `logGroupName` (erforderlich) – Name der CloudWatch Protokollgruppe, an die Sie Protokolle senden möchten. Die Gruppe wird automatisch erstellt, wenn sie noch nicht vorhanden ist.
  - `logStreamNamePrefix` (optional) – Name des Protokollstreams, an den Sie Protokolle senden möchten. Der Standardwert ist eine leere Zeichenfolge. Das Format in CloudWatch lautet wie folgt:

    ```
    ${logStreamNamePrefix}/${POD NAME}/STDOUT or STDERR
    ```

- `sideCarResources` (optional) – der Konfigurationsschlüssel zum Festlegen von Ressourcenlimits für den gestarteten Fluentbit-Sidecar-Container.
  - `memoryLimit` (optional) – das Speicherlimit. Passen Sie ihn entsprechend Ihren Anforderungen an. Der Standardwert ist 512 Mi.
  - `cpuLimit` – das CPU-Limit. Passen Sie ihn entsprechend Ihren Anforderungen an. Kein Standardwert.
- `containerLogRotationConfiguration` (optional): – steuert das Rotationsverhalten des Container-Protokolls. Sie ist standardmäßig aktiviert.
  - `rotationSize` (erforderlich) – gibt die Dateigröße für die Protokollrotation an. Der Bereich der möglichen Werte ist von 2 KB bis 2 GB. Die numerische Einheit des rotationSize-Parameters wird als Ganzzahl übergeben. Da Dezimalwerte nicht unterstützt werden, können Sie mit dem Wert 1 500 MB beispielsweise eine Rotationsgröße von 1,5 GB angeben. Der Standardwert ist 2 GB.
  - `maxFilesToKeep` (erforderlich) – gibt die maximale Anzahl von Dateien an, die nach der Rotation im Container aufbewahrt werden sollen. Der kleinste Wert ist 1 und der größte Wert ist 50. Der Standardwert ist 10.

Ausfallsicherheit des Auftrags

In den folgenden Abschnitten erfahren Sie, wie Ihre Flink-Aufträge zuverlässiger und hochverfügbar werden.

Themen

Verwenden der Hochverfügbarkeit (HA) für Flink-Operatoren und Flink-Anwendungen

Hochverfügbarkeit des Flink-Operators

Wir ermöglichen Hochverfügbarkeit für den Flink-Operator, sodass wir auf einen Standby-Flink-Operator umschalten können, um Ausfallzeiten im Regelkreis des Bedieners zu minimieren, falls

Ausfälle auftreten. Hochverfügbarkeit ist standardmäßig aktiviert, und die Standardanzahl der Startoperatorreplikate ist 2. Sie können das Feld Replicas in Ihrer `values.yaml`-Datei für das Helm-Chart konfigurieren.

Die folgenden Felder sind anpassbar:

- `replicas` (optional, Standardeinstellung ist 2): Wenn Sie diese Zahl auf einen Wert über 1 setzen, werden weitere Standby-Operatoren erstellt und Ihr Auftrag kann schneller wiederhergestellt werden.
- `highAvailabilityEnabled` (optional, der Standardwert ist true): Steuert, ob Sie HA aktivieren möchten. Wenn Sie diesen Parameter auf true angeben, wird die Unterstützung für Multi-AZ-Bereitstellungen aktiviert und die richtigen `flink-conf.yaml`-Parameter festgelegt.

Sie können HA für Ihren Betreiber deaktivieren, indem Sie die folgende Konfiguration in Ihrer `values.yaml`-Datei festlegen.

```
...
imagePullSecrets: []

replicas: 1

# set this to false if you don't want HA
highAvailabilityEnabled: false
...
```

Multi-AZ-Bereitstellung

Wir erstellen die Operator-Pods in mehreren Availability Zones. Dabei handelt es sich um eine weiche Einschränkung, und Ihre Operator-Pods werden in derselben AZ geplant, falls Sie nicht über genügend Ressourcen in einer anderen AZ verfügen.

Bestimmung des Leader-Replikats

Wenn HA aktiviert ist, ermitteln die Replikate anhand eines Leases, welcher der JMs der Leader ist, und verwenden einen K8s-Lease für die Auswahl des Leaders. Sie können den Lease beschreiben und anhand des Felds .Spec.Holder Identity den aktuellen Leader ermitteln

```
kubectl describe lease <Helm Install Release Name>-<NAMESPACE>-lease -n <NAMESPACE> |
 grep "Holder Identity"
```

Flink-S3-Interaktion

Konfigurieren von Zugriffs-Anmeldeinformationen

Bitte stellen Sie sicher, dass Sie IRSA mit den entsprechenden IAM-Berechtigungen für den Zugriff auf den S3-Bucket konfiguriert haben.

Auftrag-Jars werden aus dem S3-Anwendungsmodus abgerufen

Der Flink-Operator unterstützt auch das Abrufen von Anwendungs-Jars aus S3. Sie geben einfach den S3-Speicherort für den jarURI in Ihrer FlinkDeployment Spezifikation an.

Sie können diese Funktion auch verwenden, um andere Artefakte wie PyFlink Skripte herunterzuladen. Das resultierende Python-Skript wird unter dem Pfad `/opt/flink/usrlib/` abgelegt.

Das folgende Beispiel zeigt, wie Sie diese Funktion für einen PyFlink Auftrag verwenden. Beachten Sie die Felder jarURI und args.

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: python-example
spec:
  image: <YOUR CUSTOM PYFLINK IMAGE>
  emrReleaseLabel: "emr-6.12.0-flink-latest"
  flinkVersion: v1_16
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "1"
  serviceAccount: flink
  jobManager:
    highAvailabilityEnabled: false
    replicas: 1
    resource:
      memory: "2048m"
      cpu: 1
  taskManager:
    resource:
      memory: "2048m"
      cpu: 1
  job:
    jarURI: "s3://<S3-BUCKET>/scripts/pyflink.py" # Note, this will trigger the
 artifact download process
```

```
    entryClass: "org.apache.flink.client.python.PythonDriver"
    args: ["-pyclientexec", "/usr/local/bin/python3", "-py", "/opt/flink/usrlib/
pyflink.py"]
    parallelism: 1
    upgradeMode: stateless
```

Flink-S3-Konnektoren

Flink wird mit zwei S3-Konnektoren geliefert (unten aufgeführt). In den folgenden Abschnitten wird erläutert, wann welcher Anschluss verwendet werden sollte.

Checkpointing: Presto-S3-Konnektoren

- S3-Schema auf s3p://setzen
- Der empfohlene Konnektor für den Checkpoint zu s3.

FlinkDeployment Beispielspezifikation:

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: basic-example
spec:
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "2"
    state.checkpoints.dir: s3p://<UCKET-NAME>/flink-checkpoint/
```

- S3-Schema auf `s3://` oder (`s3a://`) setzen
- Der empfohlene Konnektor zum Lesen und Schreiben von Dateien aus S3 (einziger S3-Konnektor, der die Flinks-Dateisystemschnittstelle implementiert).
- Standardmäßig legen wir `fs.s3a.aws.credentials.provider` in der `flink-conf.yaml`-Datei fest, die `com.amazonaws.auth.WebIdentityTokenCredentialsProvider` ist. Wenn Sie die Standardeinstellung `flink-conf` vollständig überschreiben und mit S3 interagieren, stellen Sie sicher, dass Sie diesen Anbieter verwenden.

FlinkDeployment Beispielspezifikation

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
```

```
metadata:
  name: basic-example
spec:
  job:
    jarURI: local:///opt/flink/examples/streaming/WordCount.jar
    args: [ "--input", "s3a://<INPUT BUCKET>/PATH", "--output", "s3a://<OUTPUT BUCKET>/
PATH" ]
    parallelism: 2
    upgradeMode: stateless
```

Flink Job Manager

Hochverfügbarkeit (HA) für Flink-Bereitstellungen ermöglicht es Aufträgen, weiterhin Fortschritte zu machen, auch wenn ein vorübergehender Fehler auftritt und Ihre JobManager abstürzt. Die Aufträge werden neu gestartet, jedoch ab dem letzten erfolgreichen Checkpoint mit aktivierter HA. Ohne aktiviertes HA startet Kubernetes Ihr neu JobManager, aber Ihr Auftrag wird als neuer Auftrag gestartet und verliert seinen Fortschritt. Nach der Konfiguration von HA können wir Kubernetes anweisen, die HA-Metadaten in einem persistenten Speicher zu speichern, auf den im Falle eines vorübergehenden Ausfalls in der verwiesen werden soll, JobManager und dann unsere Aufträge ab dem letzten erfolgreichen Checkpoint fortzusetzen.

HA ist standardmäßig für Ihre Flink-Aufträge aktiviert (die Anzahl der Replikate ist auf 2 festgelegt, sodass Sie einen S3-Speicherort angeben müssen, damit HA-Metadaten bestehen bleiben).

HA-Konfigurationen

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: basic-example
spec:
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "2"
  executionRoleArn: "<JOB EXECUTION ROLE ARN>"
  emrReleaseLabel: "emr-6.13.0-flink-latest"
  jobManager:
    resource:
      memory: "2048m"
      cpu: 1
    replicas: 2
    highAvailabilityEnabled: true
    storageDir: "s3://<S3 PERSISTENT STORAGE DIR>"
```

```
  taskManager:
    resource:
      memory: "2048m"
      cpu: 1
```

Im Folgenden finden Sie Beschreibungen für die oben genannten HA-Konfigurationen in JobManager (definiert unter .spec.JobManager):

- `highAvailabilityEnabled` (optional, der Standardwert ist true): Setzen Sie diesen Wert auf `false` , wenn Sie HA nicht aktivieren und die bereitgestellten HA-Konfigurationen nicht verwenden möchten. Sie können das Feld „Replicas“ immer noch bearbeiten, um HA manuell zu konfigurieren.
- `replicas` (optional, Standard ist 2): Wenn Sie diese Zahl auf größer als 1 setzen, werden andere Standby-Vorgänge erstellt JobManagers und Ihr Auftrag wird schneller wiederhergestellt. Wenn Sie HA deaktivieren, müssen Sie die Anzahl der Replikate auf 1 setzen, sonst erhalten Sie weiterhin Validierungsfehler (nur 1 Replikat wird unterstützt, wenn HA nicht aktiviert ist).
- `storageDir` (erforderlich): Da wir die Anzahl der Replikate standardmäßig auf 2 setzen, müssen wir ein persistentes StorageDir bereitstellen. Derzeit akzeptiert dieses Feld nur S3-Pfade als Speicherort.

Pod-Lokalität

Wenn Sie HA aktivieren, versuchen wir auch, Pods in derselben AZ zusammenzufassen, was zu einer verbesserten Leistung führt (geringere Netzwerklatenz durch Pods in denselben AZs). Dabei handelt es sich um einen bestmöglichen Prozess. Wenn Sie also nicht über genügend Ressourcen in der AZ verfügen, in der die meisten Ihrer Pods geplant sind, werden die verbleibenden Pods zwar geplant, landen aber möglicherweise auf einem Knoten außerhalb dieser AZ.

Bestimmung des Leader-Replikats

Wenn HA aktiviert ist, ermitteln die Replikate anhand eines Leases, welches der JMs das Leader ist, und verwenden eine K8s-Configmap als Datenspeicher zum Speichern dieser Metadaten. Wenn Sie den Leader ermitteln möchten, können Sie sich den Inhalt der Configmap und den Schlüssel `org.apache.flink.k8s.leader.restserver` unter Daten ansehen, um den K8s-Pod mit der IP-Adresse zu finden. Sie können auch die folgenden Bash-Befehle verwenden.

```
ip=$(kubectl get configmap -n <NAMESPACE> <JOB-NAME>-cluster-config-map -o json | jq -
r ".data[\"org.apache.flink.k8s.leader.restserver\"]" | awk -F: '{print $2}' | awk -F
 '/' '{print $3}')
```

```
kubectl get pods -n NAMESPACE  -o json | jq -r ".items[] | select(.status.podIP ==
 \"$ip\") | .metadata.name"
```

Flink-Auftrag – natives Kubernetes

Amazon EMR 6.13.0 und höher unterstützt Flink Native Kubernetes für die Ausführung von Flink-Anwendungen im Hochverfügbarkeitsmodus auf einem Amazon-EKS-Cluster.

> **Note**
>
> Sie müssen einen Amazon-S3-Bucket erstellt haben, um die Hochverfügbarkeitsmetadaten zu speichern, wenn Sie Ihren Flink-Auftrag einreichen. Wenn Sie dieses Feature nicht verwenden möchten, können Sie sie deaktivieren. Sie ist standardmäßig aktiviert.

Um die Flink-Hochverfügbarkeitsfunktion zu aktivieren, geben Sie die folgenden Flink-Parameter an, wenn Sie den `run-application`-CLI-Befehl ausführen. Die Parameter sind unter dem Beispiel definiert.

```
-Dhigh-availability.type=kubernetes \
-Dhigh-availability.storageDir=S3://DOC-EXAMPLE-STORAGE-BUCKET \
-
Dfs.s3a.aws.credentials.provider="com.amazonaws.auth.WebIdentityTokenCredentialsProvider"
 \
-Dkubernetes.jobmanager.replicas=3 \
-Dkubernetes.cluster-id=example-cluster
```

- **`Dhigh-availability.storageDir`** – Ein S3-Bucket, in dem Sie die Ergebnisse dieser Anforderung speichern möchten.

  `Dkubernetes.jobmanager.replicas` – Die Anzahl der Job-Manager-Pods, die als Ganzzahl größer als 1 erstellt werden sollen.

  `Dkubernetes.cluster-id` – Eine eindeutige ID, die den Flink-Cluster identifiziert.

Optimierung der Neustartzeiten von Flink-Aufträgen für Aufgabenwiederherstellungs- und Skalierungsvorgänge mit Amazon EMR in EKS

Wenn eine Aufgabe fehlschlägt oder wenn ein Skalierungsvorgang stattfindet, versucht Flink, die Aufgabe vom letzten abgeschlossenen Prüfpunkt aus erneut auszuführen. Die Ausführung des Neustartvorgangs kann eine Minute oder länger dauern, abhängig von der Größe des Prüfpunktzustands und der Anzahl der parallelen Aufgaben. Während des Neustarts können sich Backlog-Aufgaben für den Auftrag ansammeln. Es gibt jedoch einige Möglichkeiten, wie Flink die Geschwindigkeit der Wiederherstellung und des Neustarts von Ausführungsdiagrammen optimiert, um die Auftragsstabilität zu verbessern.

Auf dieser Seite werden einige der Möglichkeiten beschrieben, mit denen Amazon EMR Flink die Zeit für den Neustart des Auftrags während der Aufgabenwiederherstellung oder -skalierung verbessern kann.

Themen

Aufgabenlokale Wiederherstellung

> **Note**
>
> Aufgabenlokale Wiederherstellung wird mit Flink in Amazon EMR 6.14.0 und höher unterstützt.

Mit Flink-Prüfpunkten erstellt jede Aufgabe einen Snapshot ihres Status, den Flink in verteilte Speicher wie Amazon S3 schreibt. Im Falle einer Wiederherstellung stellen die Aufgaben ihren Status aus dem verteilten Speicher wieder her. Der verteilte Speicher bietet Fehlertoleranz und kann den Status während der Neuskalierung neu verteilen, da er für alle Knoten zugänglich ist.

Ein verteilter Remote-Speicher hat jedoch auch einen Nachteil: Alle Aufgaben müssen ihren Status von einem entfernten Standort aus über das Netzwerk lesen. Dies kann bei der Aufgabenwiederherstellung oder bei Skalierungsvorgängen zu langen Wiederherstellungszeiten für große Zustände führen.

Dieses Problem der langen Wiederherstellungszeit wird durch eine aufgabenlokale Wiederherstellung gelöst. Aufgaben schreiben ihren Status am Prüfüunkt in einen sekundären Speicher, der sich lokal zur Aufgabe befindet, z. B. auf eine lokale Festplatte. Sie speichern ihren Status auch im Primärspeicher oder in unserem Fall in Amazon S3. Während der Wiederherstellung plant der Scheduler die Aufgaben in demselben Task-Manager, in dem die Aufgaben zuvor ausgeführt wurden, sodass sie aus dem lokalen Statusspeicher wiederhergestellt werden können, anstatt sie aus dem Remote-Statusspeicher zu lesen. Weitere Informationen finden Sie unter Aufgabenlokale Wiederherstellung in der Apache-Flink-Dokumentation.

Unsere Benchmark-Tests mit Beispielaufträgen haben gezeigt, dass die Wiederherstellungszeit bei aktivierter aufgabenlokaler Wiederherstellung von Minuten auf wenige Sekunden reduziert wurde.

Um die aufgabenlokale Wiederherstellung zu aktivieren, legen Sie die folgenden Konfigurationen in Ihrer `flink-conf.yaml`-Datei fest. Geben Sie den Wert für das Prüfpunkt-Intervall in Millisekunden an.

```
state.backend.local-recovery: true
state.backend: hasmap or rocksdb
state.checkpoints.dir: s3://STORAGE-BUCKET-PATH/checkpoint
execution.checkpointing.interval: 15000
```

Aufgabenlokale Wiederherstellung durch Amazon-EBS-Volume-Mount

> **Note**
>
> Die aufgabenlokale Wiederherstellung durch Amazon EBS wird mit Flink in Amazon EMR 6.15.0 und höher unterstützt.

Mit Flink in Amazon EMR in EKS können Sie automatisch Amazon-EBS-Volumes für die lokale Aufgabenwiederherstellung in den TaskManager-Pods bereitstellen. Der Standard-Overlay-Mount verfügt über ein Volumen von 10 GB, was für Aufträge mit einem niedrigeren Status ausreichend ist. Bei Aufträgen mit großem Status kann die Option automatisches EBS-Volume-Mount aktiviert

werden. Die TaskManager-Pods werden bei der Pod-Erstellung automatisch erstellt und bereitgestellt und beim Löschen des Pods entfernt.

Gehen Sie wie folgt vor, um das automatische EBS-Volume-Mount für Flink in Amazon EMR in EKS zu aktivieren:

1. Exportieren Sie die Werte für die folgenden Variablen, die Sie in den nächsten Schritten verwenden werden.

```
export AWS_REGION=aa-example-1
export FLINK_EKS_CLUSTER_NAME=my-cluster
export AWS_ACCOUNT_ID=111122223333
```

2. Erstellen oder aktualisieren Sie eine `kubeconfig`-YAML-Datei für Ihren Cluster.

```
aws eks update-kubeconfig --name $FLINK_EKS_CLUSTER_NAME --region $AWS_REGION
```

3. Erstellen Sie ein IAM-Servicekonto für den CSI (Container Storage Interface)-Treiber von Amazon EBS auf Ihrem Amazon-EKS-Cluster.

```
eksctl create iamserviceaccount \
   --name ebs-csi-controller-sa \
   --namespace kube-system \
   --region $AWS_REGION \
   --cluster $FLINK_EKS_CLUSTER_NAME\
   --role-name TLR_${AWS_REGION}_${FLINK_EKS_CLUSTER_NAME} \
   --role-only \
   --attach-policy-arn arn:aws:iam::aws:policy/service-role/
AmazonEBSCSIDriverPolicy \
   --approve
```

4. Erstellen Sie den CSI-Treiber von Amazon EBS mithilfe des folgenden Befehls:

```
eksctl create addon \
   --name aws-ebs-csi-driver \
   --region $AWS_REGION \
   --cluster $FLINK_EKS_CLUSTER_NAME \
   --service-account-role-arn arn:aws:iam::${AWS_ACCOUNT_ID}:role/TLR_
${AWS_REGION}_${FLINK_EKS_CLUSTER_NAME}
```

5. Erstellen Sie die Amazon-EBS-Speicherklasse mithilfe des folgenden Befehls:

```
cat # EOF # storage-class.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ebs-sc
provisioner: ebs.csi.aws.com
volumeBindingMode: WaitForFirstConsumer
EOF
```

Und wenden Sie dann die Klasse an:

```
kubectl apply -f storage-class.yaml
```

6. Helm installiert den Flink-Kubernetes-Operator von Amazon EMR mit Optionen zum Erstellen eines Servicekontos. Dadurch wird der `emr-containers-sa-flink` erstellt, der in der Flink-Bereitstellung verwendet werden soll.

```
helm install flink-kubernetes-operator flink-kubernetes-operator/ \
   --set jobServiceAccount.create=true \
   --set rbac.jobRole.create=true \
   --set rbac.jobRoleBinding.create=true
```

7. Um den Flink-Auftrag einzureichen und die automatische Bereitstellung von EBS-Volumes für die aufgabenlokale Wiederherstellung zu aktivieren, legen Sie die folgenden Konfigurationen in Ihrer `flink-conf.yaml`-Datei fest. Passen Sie die Größenbeschränkung an die Statusgröße des Auftrags an. Setzen Sie `serviceAccount` auf `emr-containers-sa-flink`. Geben Sie den Wert für das Prüfpunkt-Intervall in Millisekunden an. Und lassen Sie den `executionRoleArn` weg.

```
flinkConfiguration:
    task.local-recovery.ebs.enable: true
    kubernetes.taskmanager.local-recovery.persistentVolumeClaim.sizeLimit: 10Gi
    state.checkpoints.dir: s3://BUCKET-PATH/checkpoint
    state.backend.local-recovery: true
    state.backend: hasmap or rocksdb
    state.backend.incremental: "true"
    execution.checkpointing.interval: 15000
  serviceAccount: emr-containers-sa-flink
```

Wenn Sie bereit sind, das Plugin für den CSI-Treiber von Amazon EBS zu löschen, verwenden Sie die folgenden Befehle:

```
  # Detach Attached Policy
  aws iam detach-role-policy --role-name TLR_${$AWS_REGION}_${FLINK_EKS_CLUSTER_NAME}
 --policy-arn arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy
  # Delete the created Role
  aws iam delete-role --role-name TLR_${$AWS_REGION}_${FLINK_EKS_CLUSTER_NAME}
  # Delete the created service account
  eksctl delete iamserviceaccount --name ebs-csi-controller-sa --namespace kube-system
 --cluster $FLINK_EKS_CLUSTER_NAME --region $AWS_REGION
  # Delete Addon
  eksctl delete addon --name aws-ebs-csi-driver --cluster $FLINK_EKS_CLUSTER_NAME --
region $AWS_REGION
  # Delete the EBS storage class
  kubectl delete -f storage-class.yaml
```

Generischer protokollbasierter inkrementeller Prüfpunkt

> **Note**
>
> Generische protokollbasierte inkrementelle Prüfpunkte werden mit Flink in Amazon EMR in EKS 6.14.0 und höher unterstützt.

Generische protokollbasierte inkrementelle Prüfpunkte wurden in Flink 1.16 hinzugefügt, um die Geschwindigkeit von Prüfpunkten zu verbessern. Ein schnelleres Prüfpunktintervall führt häufig zu einer Reduzierung des Wiederherstellungsaufwands, da weniger Ereignisse nach der Wiederherstellung erneut verarbeitet werden müssen. Weitere Informationen finden Sie im Apache-Flink-Blog unter Verbesserung der Geschwindigkeit und Stabilität von Prüfpunkten mit generischen protokollbasierten inkrementellen Prüfpunkten.

Unsere Benchmark-Tests haben anhand von Beispielaufträgen gezeigt, dass sich die Prüfpunktzeit mit dem generischen protokollbasierten inkrementellen Prüfpunkt von Minuten auf wenige Sekunden reduziert hat.

Um generische protokollbasierte inkrementelle Prüfpunkte zu aktivieren, legen Sie die folgenden Konfigurationen in Ihrer Datei `flink-conf.yaml` fest. Geben Sie den Wert für das Prüfpunkt-Intervall in Millisekunden an.

```
    state.backend.changelog.enabled: true
    state.backend.changelog.storage: filesystem
    dstl.dfs.base-path: s3://bucket-path/changelog
    state.backend.local-recovery: true
    state.backend: rocksdb
    state.checkpoints.dir: s3://bucket-path/checkpoint
    execution.checkpointing.interval: 15000
```

Differenzierte Wiederherstellung

> **Note**
>
> Eine differenzierte Wiederherstellungsunterstützung für den Standard-Scheduler wird mit Flink in Amazon EMR in EKS 6.14.0 und höher unterstützt. Unterstützung für eine differenzierte Wiederherstellung im adaptiven Scheduler ist mit Flink in Amazon EMR in EKS 6.15.0 und höher verfügbar.

Wenn eine Aufgabe während der Ausführung fehlschlägt, setzt Flink das gesamte Ausführungsdiagramm zurück und löst eine vollständige Neuausführung ab dem letzten abgeschlossenen Prüfpunkt aus. Das ist teurer, als nur die fehlgeschlagenen Aufgaben erneut auszuführen. Bei einer differenzierten Wiederherstellung wird nur die mit der Pipeline verbundene Komponente der fehlgeschlagenen Aufgabe neu gestartet. Im folgenden Beispiel hat das Auftragsdiagramm 5 Scheitelpunkte (`A` bis `E`). Alle Verbindungen zwischen den Scheitelpunkten werden punktweise in Pipelines verlegt, und der Wert `parallelism.default` für den Auftrag ist auf 2 eingestellt.

```
A # B # C # D # E
```

In diesem Beispiel werden insgesamt 10 Aufgaben ausgeführt. Die erste Pipeline (`a1` bis `e1`) wird in einem TaskManager (`TM1`) ausgeführt, während die zweite Pipeline (`a2` bis `e2`) in einem anderen TaskManager (`TM2`) ausgeführt wird.

```
a1 # b1 # c1 # d1 # e1
a2 # b2 # c2 # d2 # e2
```

Es gibt zwei Komponenten, die über eine Pipeline miteinander verbunden sind: `a1 # e1` und `a2 # e2`. Wenn entweder `TM1` oder `TM2` fehlschlägt, wirkt sich der Fehler nur auf die 5 Aufgaben in der

Pipeline aus, in denen der TaskManager ausgeführt wurde. Bei der Neustartstrategie wird nur die betroffene Pipeline-Komponente gestartet.

Eine differenzierte Wiederherstellung funktioniert nur mit perfekt parallelen Flink-Aufträgen. Sie wird nicht mit `keyBy()`- oder `redistribute()`-Vorgängen unterstützt. Weitere Informationen finden Sie unter FLIP-1: Fine Grained Recovery from Task Failures (FLIP-1: Differenzierte Wiederherstellung nach Aufgabenfehlern) im Jira-Projekt Flink Improvement Proposal.

Um die differenzierte Wiederherstellung zu aktivieren, legen Sie die folgenden Konfigurationen in Ihrer `flink-conf.yaml`-Datei fest.

```
jobmanager.execution.failover-strategy: region
restart-strategy: exponential-delay or fixed-delay
```

Kombinierter Neustartmechanismus im adaptiven Scheduler

> **Note**
>
> Der kombinierte Neustartmechanismus im adaptiven Scheduler wird mit Flink in Amazon EMR in EKS 6.15.0 und höher unterstützt.

Der adaptive Scheduler kann die Parallelität des Auftrags auf der Grundlage der verfügbaren Slots anpassen. Er reduziert automatisch die Parallelität, wenn nicht genügend Slots für die konfigurierte Auftragsparallelität verfügbar sind. Wenn neue Slots verfügbar werden, wird der Auftrag wieder auf die konfigurierte Auftragsparallelität hochskaliert. Ein adaptiver Scheduler vermeidet Ausfallzeiten beim Auftrag, wenn nicht genügend Ressourcen verfügbar sind. Dies ist der unterstützte Scheduler für Flink Autoscaler. Aus diesen Gründen empfehlen wir den adaptiven Scheduler mit Amazon EMR Flink. Adaptive Scheduler können jedoch innerhalb kurzer Zeit mehrere Neustarts durchführen, und zwar einen Neustart für jede neu hinzugefügte Ressource. Dies könnte zu einem Leistungsabfall des Auftrags führen.

Mit Amazon EMR 6.15.0 und höher verfügt Flink über einen kombinierten Neustartmechanismus im adaptiven Scheduler, der ein Neustartfenster öffnet, wenn die erste Ressource hinzugefügt wird, und dann bis zum konfigurierten Fensterintervall von 1 Minute wartet. Er führt einen einzigen Neustart durch, wenn genügend Ressourcen zur Verfügung stehen, um den Auftrag mit konfigurierter Parallelität auszuführen, oder wenn das Intervall abgelaufen ist.

Unsere Benchmark-Tests haben anhand von Beispielaufträgen gezeigt, dass dieses Feature 10 % mehr Datensätze verarbeitet als das Standardverhalten, wenn Sie den adaptiven Scheduler und Flink Autoscaler verwenden.

Um den kombinierten Neustartmechanismus zu aktivieren, legen Sie die folgenden Konfigurationen in Ihrer Datei `flink-conf.yaml` fest.

```
jobmanager.adaptive-scheduler.combined-restart.enabled: true
jobmanager.adaptive-scheduler.combined-restart.window-interval: 1m
```

Ordnungsgemäße Stilllegung von Spot-Instances mit Flink in Amazon EMR in EKS

Flink mit Amazon EMR in EKS kann die Zeit für den Neustart des Auftrags während der Aufgabenwiederherstellung oder -skalierung verbessern.

Übersicht

Amazon EMR in EKS Versionen 6.15.0 und höher unterstützen die ordnungsgemäße Stilllegung von Task-Managern in Spot-Instances in Amazon EMR in EKS mit Apache Flink. Als Teil dieses Features bietet Amazon EMR in EKS mit Flink die folgenden Funktionen:

- J-ust-in-time Checkpointing – Flink-Streaming-Aufträge können auf Spot-Instance-Unterbrechung reagieren, just-in-time (JIT)-Checkpoint der laufenden Aufträge durchführen und die Planung zusätzlicher Aufgaben auf diesen Spot-Instances verhindern. Der JIT-Prüfpunkt wird mit dem standardmäßigen und dem adaptiven Scheduler unterstützt.
- Kombinierter Neustartmechanismus – Ein kombinierter Neustartmechanismus versucht nach besten Kräften, den Auftrag neu zu starten, nachdem er die Parallelität der Zielressource oder das Ende des aktuell konfigurierten Fensters erreicht hat. Dies verhindert auch aufeinanderfolgende Auftragsneustarts, die durch das Beenden mehrerer Spot-Instances verursacht werden könnten. Der kombinierte Neustartmechanismus ist nur mit dem adaptiven Scheduler verfügbar.

Diese Funktionen bieten die folgenden Vorteile:

- Sie können Spot-Instances nutzen, um Task-Manager auszuführen und die Cluster-Ausgaben zu reduzieren.
- Die verbesserte Verfügbarkeit des Spot-Instance-Task-Managers führt zu einer höheren Ausfallsicherheit und einer effizienteren Auftragsplanung.

- Ihre Flink-Aufträge haben eine höhere Verfügbarkeit, da nach der Beendigung der Spot-Instance weniger Neustarts erforderlich sind.

Funktionsweise

Nehmen wir das folgende Beispiel: Sie stellen einen Cluster von Amazon EMR in EKS bereit, auf dem Apache Flink ausgeführt wird, und Sie geben On-Demand-Knoten für Job-Manager und Spot-Instance-Knoten für Task-Manager an. Zwei Minuten vor der Beendigung erhält der Task-Manager eine Benachrichtigung über die Unterbrechung.

In diesem Szenario würde der Job Manager das Signal zur Unterbrechung der Spot-Instance verarbeiten, die Planung zusätzlicher Aufgaben auf der Spot-Instance blockieren und das JIT-Checkpointing für den Streaming-Auftrag initiieren.

Dann würde der Job-Manager das Auftragsdiagramm erst dann neu starten, wenn ausreichend neue Ressourcen verfügbar sind, um die aktuelle Auftragsparallelität im aktuellen Neustartintervallfenster zu erfüllen. Das Intervall für das Neustartfenster wird auf der Grundlage der Dauer des Austauschs der Spot-Instance, der Erstellung neuer Task-Manager-Pods und der Registrierung bei Job Manager festgelegt.

Voraussetzungen

Um eine ordnungsgemäße Dekomprimierung zu verwenden, erstellen Sie einen Streaming-Auftrag auf einem Amazon-EMR-in-EKS-Cluster, auf dem Apache Flink ausgeführt wird, und führen Sie ihn aus. Aktivieren Sie den adaptiven Scheduler und Task-Manager, die für mindestens eine Spot-Instance geplant sind, wie im folgenden Beispiel gezeigt. Sie sollten On-Demand-Knoten für den Job-Manager verwenden, und Sie können On-Demand-Knoten für den Task-Manager verwenden, sofern es auch mindestens eine Spot-Instance gibt.

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: deployment_name
spec:
  flinkVersion: v1_17
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "2"
    cluster.taskmanager.graceful-decommission.enabled: "true"
    execution.checkpointing.interval: "240s"
    jobmanager.adaptive-scheduler.combined-restart.enabled: "true"
```

```
    jobmanager.adaptive-scheduler.combined-restart.window-interval : "1m"
  serviceAccount: flink
  jobManager:
    resource:
      memory: "2048m"
      cpu: 1
    nodeSelector:
      'eks.amazonaws.com/capacityType': 'ON_DEMAND'
  taskManager:
    resource:
      memory: "2048m"
      cpu: 1
    nodeSelector:
      'eks.amazonaws.com/capacityType': 'SPOT'
  job:
    jarURI: flink_job_jar_path
```

Konfiguration

In diesem Abschnitt werden die meisten Konfigurationen behandelt, die Sie für Ihre Stilllegungsbedürfnisse angeben können.

Schlüssel	Beschreibung	Standardwert	Zulässige Werte
`cluster.taskmanager.graceful-decommission.enabled`	Aktivieren Sie die ordnungsgemäße Stilllegung des Task-Managers.	`true`	`true`, `false`
`jobmanager.adaptive-scheduler.combined-restart.enabled`	Aktivieren Sie den kombinierten Neustartmechanismus im adaptiven Scheduler.	`false`	`true`, `false`
`jobmanager.adaptiv`	Das kombinierte Neustart-Fensterintervall für die Durchführung	`1m`	Beispiele: `30`, `60s`, `3m`, `1h`

Schlüssel	Beschreibung	Standardwert	Zulässige Werte
`e-schedul er.combin ed-restar t.window- interval`	zusammengeführter Neustarts für den Auftrag. Eine Ganzzahl ohne Einheit wird als Millisekunden interpretiert.		

Verwenden von Autoscaler für Flink-Anwendungen

Der Operator-Autoscaler kann dazu beitragen, den Gegendruck zu verringern, indem er Messwerte von Flink-Aufträge sammelt und die Parallelität automatisch auf Auftrag-Scheitelpunktebene anpasst. Im Folgenden finden Sie ein Beispiel dafür, wie Ihre Konfiguration aussehen könnte:

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  ...
spec:
  ...
  flinkVersion: v1_17
  flinkConfiguration:
    kubernetes.operator.job.autoscaler.enabled: "true"
    kubernetes.operator.job.autoscaler.stabilization.interval: 1m
    kubernetes.operator.job.autoscaler.metrics.window: 5m
    kubernetes.operator.job.autoscaler.target.utilization: "0.6"
    kubernetes.operator.job.autoscaler.target.utilization.boundary: "0.2"
    kubernetes.operator.job.autoscaler.restart.time: 2m
    kubernetes.operator.job.autoscaler.catch-up.duration: 5m
    pipeline.max-parallelism: "720"
  ...
```

Im Folgenden finden Sie die Konfigurationsoptionen für den Autoscaler.

- `kubernetes.operator.job.autoscaler.scaling.enabled` – gibt an, ob die Autoscaler-Aktion aktiviert werden soll. Der Standardwert ist false, um einen passiven Modus oder Modus nur für Metriken zu unterstützen, in dem der Autoscaler nur skalierungsbezogene Leistungsmetriken sammelt und auswertet, aber keine Auftrag-Upgrades auslöst. Dies kann verwendet werden,

um Vertrauen in das Modul zu gewinnen, ohne dass dies Auswirkungen auf die laufenden Anwendungen hat.

- `kubernetes.operator.job.autoscaler.stabilization.interval` – der Stabilisierungszeitraum, in dem keine neue Skalierung durchgeführt wird. Standard ist 5 Minuten.
- `kubernetes.operator.job.autoscaler.metrics.window` – die Größe des Aggregationsfensters für Skalierungsmetriken. Je größer das Fenster, desto flüssiger und stabiler, aber der Autoscaler reagiert möglicherweise langsamer auf plötzliche Laständerungen. Standard ist 10 Minuten. Wir empfehlen Ihnen, zu experimentieren, indem Sie einen Wert zwischen 3 und 60 Minuten verwenden.
- `kubernetes.operator.job.autoscaler.target.utilization` – die Zielauslastung der Scheitelpunkte, um eine stabile Arbeitsleistung und einen gewissen Puffer für Lastschwankungen zu gewährleisten. In der Standardeinstellung wird eine `0.7` Auslastung/Last von 70 % für die Scheitelpunkte des Aufträge angestrebt.
- `kubernetes.operator.job.autoscaler.target.utilization.boundary` – die Auslastungsgrenze für den Zielscheitelpunkt, die als zusätzlicher Puffer dient, um eine sofortige Skalierung bei Lastschwankungen zu vermeiden. Die Standardeinstellung ist `0.4`, was bedeutet, dass eine Abweichung von 40 % von der Zielauslastung zulässig ist, bevor eine Skalierungsaktion ausgelöst wird.
- `kubernetes.operator.job.autoscaler.restart.time` – die erwartete Zeit für den Neustart der Anwendung. Standard ist 3 Minuten.
- `kubernetes.operator.job.autoscaler.catch-up.duration` – die erwartete Aufholzeit, d. h. die vollständige Bearbeitung von etwaigen Rückständen nach Abschluss eines Skalierungsvorgangs. Standard ist 5 Minuten. Durch die Verkürzung der Nachholdauer muss der Autoscaler mehr zusätzliche Kapazität für die Skalierungsaktionen reservieren.
- `pipeline.max-parallelism` – die maximale Parallelität, die der Autoscaler verwenden kann. Der Autoscaler ignoriert dieses Limit, wenn es höher ist als die maximale Parallelität, die in der Flink-Konfiguration oder direkt für jeden Operator konfiguriert wurde. Der Standardwert ist 200. Beachten Sie, dass der Autoscaler die Parallelität als Divisor der maximalen Parallelitätszahl berechnet. Es wird daher empfohlen, Einstellungen für maximale Parallelität zu wählen, die viele Teiler haben, anstatt sich auf die von Flink bereitgestellten Standardwerte zu verlassen. Wir empfehlen, für diese Konfiguration ein Vielfaches von 60 zu verwenden, z. B. 120, 180, 240, 360, 720 usw.

Eine detailliertere Referenzseite zur Konfiguration finden Sie unter Autoscaler-Konfiguration.

Autoscaler-Parameter-Autotuning

Der integrierte Open-Source-Flink Autoscaler verwendet zahlreiche Metriken, um die besten Skalierungsentscheidungen zu treffen. Die Standardwerte, die es für seine Berechnungen verwendet, sollen jedoch für die meisten Workloads gelten und für einen bestimmten Auftrag möglicherweise nicht optimal sein. Das Auto-Tuning-Feature, das der Version von Amazon EMR in EKS des Flink Operator hinzugefügt wurde, untersucht historische Trends, die bei bestimmten erfassten Metriken beobachtet wurden, und versucht dann entsprechend, den optimalen Wert zu berechnen, der auf den angegebenen Auftrag zugeschnitten ist.

Konfiguration	Erforderlich	Standard	Beschreibung
kubernetes.operator.job.autoscaler.autotune.enable	False	False	Gibt an, ob der Flink Autoscaler Konfigurationen im Laufe der Zeit automatisch anpassen soll, um die Skalierungsentscheidungen von Autoscalern zu optimieren. Derzeit kann der Autoscaler nur den Autoscaler-Parameter automatisch optimiere `nrestart.time` .
kubernetes.operator.job.autoscaler.autotune.metrics.history.max.count	False	3	Gibt an, wie viele historische Amazon-EMR-in-EKS-Metriken der Autoscaler in der Amazon-EMR-in-EKS-Metrik-Konfigurationskarte speichert.
kubernetes.operator.job.autoscaler.autotune.metrics.restart.count	False	3	Gibt an, wie viele Neustarts der Autoscaler durchführt, bevor er mit der Berechnung der durchschnittlichen Neustartzeit für einen bestimmten Auftrag beginnt.

Um die automatische Optimierung zu aktivieren, müssen Sie Folgendes abgeschlossen haben:

- Auf festlegen `kubernetes.operator.job.autoscaler.autotune.enable: true`
- Auf festlegen `metrics.job.status.enable: TOTAL_TIME`
- Die Einrichtung von Verwenden von Autoscaler für Flink-Anwendungen zum Aktivieren des Autotunings wurde befolgt

Im Folgenden finden Sie ein Beispiel für eine Bereitstellungsspezifikation, mit der Sie Autotuning ausprobieren können.

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: autoscaling-example
spec:
  flinkVersion: v1_18
  flinkConfiguration:

    # Autotuning parameters
    kubernetes.operator.job.autoscaler.autotune.enable: "true"
    kubernetes.operator.job.autoscaler.autotune.metrics.history.max.count: "2"
    kubernetes.operator.job.autoscaler.autotune.metrics.restart.count: "1"
    metrics.job.status.enable: TOTAL_TIME

    # Autoscaler parameters
    kubernetes.operator.job.autoscaler.enabled: "true"
    kubernetes.operator.job.autoscaler.scaling.enabled: "true"
    kubernetes.operator.job.autoscaler.stabilization.interval: "5s"
    kubernetes.operator.job.autoscaler.metrics.window: "1m"

    jobmanager.scheduler: adaptive

    taskmanager.numberOfTaskSlots: "1"
    state.savepoints.dir: s3://<S3_bucket>/autoscaling/savepoint/
    state.checkpoints.dir: s3://<S3_bucket>/flink/autoscaling/checkpoint/
    pipeline.max-parallelism: "4"

  executionRoleArn: <JOB ARN>
  emrReleaseLabel: emr-6.14.0-flink-latest
  jobManager:
    highAvailabilityEnabled: true
    storageDir: s3://<S3_bucket>/flink/autoscaling/ha/
    replicas: 1
```

```
    resource:
      memory: "1024m"
      cpu: 0.5
  taskManager:
    resource:
      memory: "1024m"
      cpu: 0.5
  job:
    jarURI: s3://<S3_bucket>/some-job-with-back-pressure
    parallelism: 1
    upgradeMode: last-state
```

Um Gegendruck zu simulieren, verwenden Sie die folgende Bereitstellungsspezifikation.

```
  job:
    jarURI: s3://<S3_bucket>/pyflink-script.py
    entryClass: "org.apache.flink.client.python.PythonDriver"
    args: ["-py", "/opt/flink/usrlib/pyflink-autotuning-script.py"]
    parallelism: 1
    upgradeMode: last-state
```

Laden Sie das folgende Python-Skript in Ihren S3-Bucket hoch.

```
import logging
import sys
import time
import random

from pyflink.datastream import StreamExecutionEnvironment
from pyflink.table import StreamTableEnvironment

TABLE_NAME="orders"
QUERY=f"""
CREATE TABLE {TABLE_NAME} (
  id INT,
  order_time AS CURRENT_TIMESTAMP,
  WATERMARK FOR order_time AS order_time - INTERVAL '5' SECONDS
)
WITH (
  'connector' = 'datagen',
  'rows-per-second'='10',
  'fields.id.kind'='random',
  'fields.id.min'='1',
```

```
  'fields.id.max'='100'
);
"""

def create_backpressure(i):
    time.sleep(2)
    return i

def autoscaling_demo():
    env = StreamExecutionEnvironment.get_execution_environment()
    t_env = StreamTableEnvironment.create(env)
    t_env.execute_sql(QUERY)
    res_table = t_env.from_path(TABLE_NAME)

    stream =  t_env.to_data_stream(res_table) \
      .shuffle().map(lambda x: create_backpressure(x))\
      .print()
    env.execute("Autoscaling demo")

if __name__ == '__main__':
    logging.basicConfig(stream=sys.stdout, level=logging.INFO, format="%(message)s")
    autoscaling_demo()
```

Verwenden Sie die folgenden Befehle, um zu überprüfen, ob Ihr Autotuner funktioniert. Beachten Sie, dass Sie Ihre eigenen Leader-Pod-Informationen für den Flink Operator verwenden müssen.

Zuerst der Name Ihres Leader-Pods.

```
ip=$(kubectl get configmap -n $NAMESPACE <job-name>-cluster-config-map -o json | jq -
r ".data[\"org.apache.flink.k8s.leader.restserver\"]" | awk -F: '{print $2}' | awk -F
 '/' '{print $3}')

kubectl get pods -n $NAMESPACE -o json | jq -r ".items[] | select(.status.podIP ==
 \"$ip\") | .metadata.name"
```

Sobald Sie den Namen Ihres Leader-Pods haben, können Sie den folgenden Befehl ausführen.

```
kubectl logs -n $NAMESPACE  -c flink-kubernetes-operator --follow <YOUR-FLINK-OPERATOR-
POD-NAME>  | grep -E 'EmrEks|autotun|calculating|restart|autoscaler'
```

Sie sollten Protokolle ähnlich den folgenden sehen.

```
[m[33m2023-09-13 20:10:35,941[m [36mc.a.c.f.k.o.a.EmrEksMetricsAutotuner[m
 [36m[DEBUG][flink/autoscaling-example] Using the latest
 Emr Eks Metric for calculating restart.time for autotuning:
 EmrEksMetrics(restartMetric=RestartMetric(restartingTime=65, numRestarts=1))

[m[33m2023-09-13 20:10:35,941[m [36mc.a.c.f.k.o.a.EmrEksMetricsAutotuner[m [32m[INFO ]
[flink/autoscaling-example] Calculated average restart.time metric via autotuning to
 be: PT0.065S
```

Wartung und Fehlerbehebung

In den folgenden Abschnitten wird beschrieben, wie Sie Ihre lang laufenden Flink-Aufträge verwalten können, und Sie erhalten Anleitungen zur Behebung einiger häufig auftretender Probleme.

Migrieren von Flink-Anwendungen

Flink-Anwendungen sind in der Regel so konzipiert, dass sie über lange Zeiträume wie Wochen, Monate oder sogar Jahre ausgeführt werden können. Wie alle Services mit langer Laufzeit müssen auch Flink-Streaming-Anwendungen gewartet werden. Dies beinhaltet Fehlerbehebungen, Verbesserungen und die Migration zu einem Flink-Cluster einer späteren Version.

Wenn sich die Spezifikation für `FlinkDeployment`- und `FlinkSessionJob`-Ressourcen ändert, müssen Sie die laufende Anwendung aktualisieren. Zu diesem Zweck stoppt der Operator den laufenden Auftrag (sofern er nicht bereits unterbrochen wurde) und stellt ihn erneut mit der neuesten Spezifikation und, bei statusbehafteten Anwendungen, mit dem Status der vorherigen Ausführung bereit.

Benutzer steuern mit der `upgradeMode`-Einstellung von `JobSpec`, wie der Status verwaltet werden soll, wenn statusbehaftete Anwendungen beendet und wiederhergestellt werden.

Upgrade-Modi

Optionale Einführung

Zustandslos

Zustandslose Anwendungen werden aus dem Status „Leer“ aktualisiert.

Letzter Status

Schnelle Upgrades in jedem Anwendungsstatus (auch bei fehlgeschlagenen Aufträgen) erfordern keinen fehlerfreien Auftrag, da immer der letzte erfolgreiche Prüfpunkt verwendet wird. Eine manuelle Wiederherstellung kann erforderlich sein, wenn HA-Metadaten verloren gehen. Um die Zeit zu begrenzen, in der der Auftrag möglicherweise auf den letzten Prüfpunkt zurückgreifen kann, können Sie `kubernetes.operator.job.upgrade.last-state.max.allowed.checkpoint.age` konfigurieren. Wenn der Prüfpunkt älter als der konfigurierte Wert ist, wird stattdessen ein Savepoint für fehlerfreie Aufträge verwendet. Dies wird im Sitzungsmodus nicht unterstützt.

Savepoint

Verwenden Sie Savepoint für das Upgrade, der maximale Sicherheit und die Möglichkeit bietet, als Backup-/Fork-Point zu dienen. Der Savepoint wird während des Upgrade-Prozesses erstellt. Beachten Sie, dass der Flink-Auftrag ausgeführt werden muss, damit der Savepoint erstellt werden kann. Wenn sich der Auftrag in einem fehlerhaften Zustand befindet, wird der letzte Prüfpunkt verwendet (es sei denn, kubernetes.operator.job.upgrade.last-state-fallback.enabled ist auf false gesetzt). Wenn der letzte Prüfpunkt nicht verfügbar ist, schlägt das Upgrade des Auftrags fehl.

Fehlerbehebung

In diesem Abschnitt wird beschrieben, wie Sie Probleme mit Amazon EMR in EKS beheben. Informationen zur Behebung allgemeiner Probleme mit Amazon EMR finden Sie unter Fehlerbehebung eines Clusters im Verwaltungshandbuch von Amazon EMR.

- Fehlerbehebung bei Aufträgen, die PersistentVolumeClaims (PVC) verwenden
- Fehlerbehebung von Amazon EMR im vertikalen Auto Scaling von EKS
- Fehlerbehebung beim Spark-Operator in Amazon EMR in EKS

Fehlerbehebung für Apache Flink auf Amazon EMR in EKS

Bei der Installation des Helm-Charts wurde keine Ressourcenzuweisung gefunden

Bei der Installation des Helm-Charts wird möglicherweise die folgende Fehlermeldung angezeigt.

```
Error: INSTALLATION FAILED: pulling from host 1234567890.dkr.ecr.us-
west-2.amazonaws.com failed with status code [manifests 6.13.0]: 403 Forbidden Error:
```

```
 INSTALLATION FAILED: unable to build kubernetes objects from release manifest:
 [resource mapping not found for name: "flink-operator-serving-cert" namespace: "<the
 namespace to install your operator>" from "": no matches for kind "Certificate" in
 version "cert-manager.io/v1"

ensure CRDs are installed first, resource mapping not found for name: "flink-operator-
selfsigned-issuer" namespace: "<the namespace to install your operator>" " from "": no
 matches for kind "Issuer" in version "cert-manager.io/v1"

ensure CRDs are installed first].
```

Um diesen Fehler zu beheben, installieren Sie cert-manager, um das Hinzufügen der Webhook-Komponente zu ermöglichen. Sie müssen cert-manager die Installation auf jedem Amazon-EKS-Cluster durchführen, den Sie verwenden.

```
kubectl apply -f https://github.com/cert-manager/cert-manager/releases/download/v1.12.0
```

AWS-Service-Zugriffsverweigerungsfehler

Wenn Sie einen access denied Fehler sehen, vergewissern Sie sich, dass die IAM-Rolle für `operatorExecutionRoleArn` in der `values.yaml`-Helm-Chart-Datei über die richtigen Berechtigungen verfügt. Stellen Sie außerdem sicher, dass die in Ihrer `FlinkDeployment`-Spezifikation unter `executionRoleArn` angegebene IAM-Rolle über die richtigen Berechtigungen verfügt.

`FlinkDeployment` steckt fest

Wenn Ihr `FlinkDeployment` im angehaltenen Zustand stehen bleibt, führen Sie die folgenden Schritte aus, um das Löschen der Bereitstellung zu erzwingen:

1. Bearbeiten Sie den Bereitstellungslauf.

   ```
   kubectl edit -n Flink Namespace flinkdeployments/App Name
   ```

2. Entfernen Sie diesen Finalizer.

   ```
   finalizers:
     - flinkdeployments.flink.apache.org/finalizer
   ```

3. Löschen Sie die Bereitstellung.

```
kubectl delete -n Flink Namespace flinkdeployments/App Name
```

Unterstützte Versionen für Amazon EMR in EKS mit Apache Flink

Apache Flink ist mit den folgenden Versionen von Amazon EMR in EKS verfügbar. Informationen zu allen verfügbaren Versionen finden Sie unter Versionen von Amazon EMR in EKS.

Versionsbezeichnung	Java	Flink	Flink-Operator
emr-7.0.0-flink-latest	11	1.18.0	-
emr-7.0.0-flink-k8s-operator-latest	11	1.18.0	1.6.1
emr-6.15.0-flink-latest	11	1.17.1 ist insgesamt möglich.	-
emr-6.15.0-flink-k8s-operator-latest	11	1.17.1 ist insgesamt möglich.	1.6.0
emr-6.14.0-flink-latest	11	1.17.1 ist insgesamt möglich.	-
emr-6.14.0-flink-k8s-operator-latest	11	1.17.1 ist insgesamt möglich.	1.6.0
emr-6.13.0-flink-latest	11	1.17.0	-
emr-6.13.0-flink-k8s-operator-latest	11	1.17.0	1.5.0

Ausführen von Aufträgen mit Amazon EMR in EKS

Eine Auftragsausführung ist eine Arbeitseinheit, z. B. eine Spark-Jar-, PySpark Skript- oder SparkSQL-Abfrage, die Sie an Amazon EMR on EKS senden. Dieses Thema bietet einen Überblick über die Verwaltung von Auftragsausführungen mit der AWS CLI, die Anzeige von Auftragsausführungen mit der Amazon EMR-Konsole und die Behebung häufiger Fehler bei der Auftragsausführung.

Beachten Sie, dass Sie keine IPv6-Spark-Jobs auf Amazon EMR auf EKS ausführen können

> **Note**
>
> Bevor Sie einen Auftrag einreichen, der mit Amazon EMR in EKS ausgeführt wird, müssen Sie die unter aufgeführten Schritte in Einrichten von Amazon EMR in EKS ausführen.

Themen

Spark-Aufträge ausführen mit `StartJobRun`

Themen

- Einrichten von Amazon EMR in EKS
- Reichen Sie einen Auftrag ein, der ausgeführt wird mit StartJobRun

Einrichten von Amazon EMR in EKS

Führen Sie zum Einrichten von Amazon EMR in EKS die folgenden Schritte aus. Wenn Sie bereits für Amazon Web Services (AWS) registriert sind und Amazon EKS schon verwendet haben, müssen Sie nur wenige Schritte ausführen, um Amazon EMR in EKS nutzen zu können. Überspringen Sie alle Aufgaben, die Sie bereits abgeschlossen haben.

> **Note**
>
> Sie können auch dem Amazon EMR in EKS-Workshop folgen, um alle Ressourcen einzurichten, die für die Ausführung von Spark-Jobs auf Amazon EMR in EKS erforderlich sind. Der Workshop bietet auch Automatisierung, indem CloudFormation Vorlagen verwendet werden, um die Ressourcen zu erstellen, die Sie für den Einstieg benötigen. Weitere Vorlagen und Best Practices finden Sie in unserem EMR Containers Best Practices Guide unter GitHub.

1. Installieren Sie das AWS CLI
2. Installieren `eksctl`
3. Einen Amazon-EKS-Cluster einrichten
4. Gewähren eines Clusterzugriff für Amazon EMR in EKS
5. Aktivieren Sie IAM-Rollen für Servicekonten (IRSA) auf dem EKS-Cluster
6. Erstellen einer Aufgabenausführungsrolle
7. Die Vertrauensrichtlinie der Auftragsausführungsrolle aktualisieren
8. Gewähren Sie Benutzern Zugriff auf Amazon EMR in EKS
9. Den Amazon-EKS-Cluster mit Amazon EMR registrieren

Installieren Sie das AWS CLI

Sie können die neueste Version von AWS CLI für macOS, Linux oder Windows installieren.

> **Important**
>
> Um Amazon EMR auf EKS einzurichten, muss die neueste Version von AWS CLI installiert sein.

Um das AWS CLI für macOS zu installieren oder zu aktualisieren

1. Wenn Sie das derzeit AWS CLI installiert haben, ermitteln Sie, welche Version Sie installiert haben.

   ```
   aws --version
   ```

2. Wenn Sie eine frühere Version von haben AWS CLI, verwenden Sie den folgenden Befehl, um die neueste AWS CLI Version 2 zu installieren. Weitere Installationsoptionen oder zum Aktualisieren Ihrer aktuell installierten Version 2 finden Sie unter Aktualisierung der AWS CLI Version 2 auf macOS.

   ```
   curl "https://awscli.amazonaws.com/AWSCLIV2.pkg" -o "AWSCLIV2.pkg"
   sudo installer -pkg AWSCLIV2.pkg -target /
   ```

   Wenn Sie AWS CLI Version 2 nicht verwenden können, stellen Sie mithilfe des folgenden Befehls sicher, dass Sie die neueste AWS CLI Version von Version 1 installiert haben.

   ```
   pip3 install awscli --upgrade --user
   ```

Um das AWS CLI für Linux zu installieren oder zu aktualisieren

1. Wenn Sie das derzeit AWS CLI installiert haben, ermitteln Sie, welche Version Sie installiert haben.

   ```
   aws --version
   ```

2. Wenn Sie eine frühere Version von haben AWS CLI, verwenden Sie den folgenden Befehl, um die neueste AWS CLI Version 2 zu installieren. Weitere Installationsoptionen oder zum Aktualisieren Ihrer aktuell installierten Version 2 finden Sie unter Aktualisieren der AWS CLI Version 2 unter Linux.

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
```

Wenn Sie AWS CLI Version 2 nicht verwenden können, stellen Sie mithilfe des folgenden Befehls sicher, dass Sie die neueste AWS CLI Version von Version 1 installiert haben.

```
pip3 install --upgrade --user awscli
```

Um das AWS CLI für Windows zu installieren oder zu aktualisieren

1. Wenn Sie das derzeit AWS CLI installiert haben, ermitteln Sie, welche Version Sie installiert haben.

   ```
   aws --version
   ```

2. Wenn Sie eine frühere Version von haben AWS CLI, verwenden Sie den folgenden Befehl, um die neueste AWS CLI Version 2 zu installieren. Weitere Installationsoptionen oder zum Aktualisieren Ihrer aktuell installierten Version 2 finden Sie unter Aktualisieren der AWS CLI Version 2 unter Windows.

   1. Laden Sie das AWS CLI MSI-Installationsprogramm für Windows (64-Bit) unter https://awscli.amazonaws.com/ AWSCLIV 2.msi herunter
   2. Führen Sie das heruntergeladene MSI-Installationsprogramm aus und folgen Sie den Anweisungen auf dem Bildschirm. Standardmäßig AWS CLI installiert das aufC:\Program Files\Amazon\AWSCLIV2.

   Wenn Sie AWS CLI Version 2 nicht verwenden können, stellen Sie mithilfe des folgenden Befehls sicher, dass Sie die neueste AWS CLI Version der Version 1 installiert haben.

   ```
   pip3 install --user --upgrade awscli
   ```

Konfigurieren Sie Ihre AWS CLI Anmeldedaten

Sowohl eksctl als auch the AWS CLI erfordern, dass Sie in Ihrer AWS Umgebung Anmeldeinformationen konfiguriert haben. Für den allgemeinen Gebrauch lässt sich Ihre AWS CLI - Installation am schnellsten über die `aws configure` einrichten.

```
$ aws configure
AWS Access Key ID [None]: <AKIAIOSFODNN7EXAMPLE>
AWS Secret Access Key [None]: <wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY>
Default region name [None]: <region-code>
Default output format [None]: <json>
```

Wenn Sie diesen Befehl eingeben, werden Sie zur Eingabe von vier Informationen AWS CLI aufgefordert: Zugriffsschlüssel, geheimer Zugriffsschlüssel, AWS Region und Ausgabeformat. Diese Informationen werden in einem Profil (eine Sammlung von Einstellungen) mit dem Namen `default` gespeichert. Dieses Profil wird verwendet, wenn Sie Befehle ausführen, es sei denn, Sie geben ein anderes Profil an. Weitere Informationen finden Sie unter Konfiguration von AWS CLI im AWS Command Line Interface Benutzerhandbuch.

Installieren **`eksctl`**

Installieren Sie die neueste Version des `eksctl`-Befehlszeilenprogramms unter macOS, Linux oder Windows. Weitere Informationen finden Sie unter https://eksctl.io/.

> **Important**
>
> Wir empfehlen, dass Sie die neueste Version von eksctl herunterladen, da für einige Funktionen in Amazon EMR auf EKS spätere Versionen erforderlich sind. Weitere Informationen finden Sie unter Installieren `eksctl`.

So installieren oder aktualisieren Sie eksctl mithilfe von Homebrew auf macOS

Am einfachsten gelingt der Einstieg in Amazon EKS und macOS mit einer Installation von eksctl mithilfe von Homebrew. Das eksctl-Homebrew-Rezept installiert eksctl und alle anderen Abhängigkeiten, die für Amazon EKS erforderlich sind, z. B. .kubectl. Das Rezept installiert auch das aws-iam-authenticator, was erforderlich ist, wenn Sie die AWS CLI Version 1.16.156 oder höher nicht installiert haben.

1. Wenn Sie Homebrew noch nicht auf macOS installiert haben, installieren Sie es mit dem folgenden Befehl.

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/
master/install.sh)"
```

2. Installieren Sie die Weaveworks-Homebrew-Registerkarte.

```
brew tap weaveworks/tap
```

3. 1. Installieren oder Aktualisieren von eksctl.
   - Installieren Sie `eksctl` mit dem folgenden Befehl:

     ```
     brew install weaveworks/tap/eksctl
     ```

   - Wenn `eksctl` bereits installiert ist, führen Sie mit folgenden Befehl ein Upgrade durch.

     ```
     brew upgrade eksctl & brew link --overwrite eksctl
     ```

   2. Überprüfen Sie mit folgendem Befehl, ob die Installation erfolgreich war. Sie müssen `eksctl` Version 0.34.0 oder höher verwenden.

      ```
      eksctl version
      ```

So installieren oder aktualisieren Sie **`eksctl`** unter Linux mithilfe von **`curl`**

1. Laden Sie die neueste Version von eksctl mit folgenden Befehl herunter.

```
curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/
download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp
```

2. Verschieben Sie die extrahierte Binärdatei zu /usr/local/bin.

```
sudo mv /tmp/eksctl /usr/local/bin
```

3. Überprüfen Sie mit folgendem Befehl, ob die Installation erfolgreich war. Sie müssen `eksctl` Version 0.34.0 oder höher verwenden.

```
eksctl version
```

So installieren oder aktualisieren Sie **`eksctl`** mit Chocolatey unter Windows

1. Wenn Sie Chocolatey noch nicht auf Ihrem Windows-System installiert haben, finden Sie Informationen unter Installieren von Chocolatey.
2. Installieren oder aktualisieren Sie `eksctl`.
   - Installieren Sie mit dem folgenden Befehl die Binärdateien.

     ```
     choco install -y eksctl
     ```

   - Wenn sie bereits installiert sind, führen Sie mit folgenden Befehl ein Upgrade durch:

     ```
     choco upgrade -y eksctl
     ```

3. Überprüfen Sie mit folgendem Befehl, ob die Installation erfolgreich war. Sie müssen `eksctl` Version 0.34.0 oder höher verwenden.

   ```
   eksctl version
   ```

Einen Amazon-EKS-Cluster einrichten

Amazon EKS ist ein verwalteter Service, mit dem Sie Kubernetes ganz einfach ausführen können, AWS ohne Ihre eigene Kubernetes-Steuerebene oder Knoten installieren, betreiben und warten zu müssen. Führen Sie die unten beschriebenen Schritte aus, um einen neuen Kubernetes-Cluster mit Knoten in Amazon EKS zu erstellen.

Voraussetzungen

> **Important**
>
> Bevor Sie einen Amazon-EKS-Cluster erstellen, sollten Sie die Amazon-EKS-VPC- und Subnetzanforderungen und Überlegungen im Amazon-EKS-Benutzerhandbuch abschließen, um sicherzustellen, dass Ihre Amazon-EKS-Cluster wie erwartet funktionieren und skalieren.

Sie müssen die folgenden Werkzeuge und Ressourcen installieren und konfigurieren, die Sie zum Erstellen und Verwalten eines Amazon-EKS-Clusters benötigen:

- Die neueste Version von. AWS CLI
- `kubectl` Version 1.20 oder höher.
- Die neueste Version von `eksctl`.

Weitere Informationen finden Sie unter Installieren Sie das AWS CLI, Installieren von `kubectl` und Installieren `eksctl`.

Erstellen Sie einen Amazon-EKS-Cluster mit **`eksctl`**

Führen Sie die folgenden Schritte aus, um einen Amazon-EKS-Cluster mit `eksctl` zu erstellen.

> **Important**
>
> Um schnell loszulegen, können Sie einen EKS-Cluster und die Knoten mit Standardeinstellungen erstellen. Für den Produktionseinsatz empfehlen wir Ihnen jedoch, die Einstellungen für den Cluster und die Knoten an Ihre spezifischen Anforderungen anzupassen. Führen Sie den Befehl `eksctl create cluster -h` aus, um eine Liste aller Einstellungen und Optionen anzuzeigen. Weitere Informationen zum Erstellen und Verwalten von Clustern finden Sie in der `eksctl`-Dokumentation.

1. Erstellen Sie ein Amazon-EC2-Schlüsselpaar.

   Wenn Sie über kein vorhandenes Schlüsselpaar verfügen, können Sie den folgenden Befehl ausführen, um ein neues Schlüsselpaar zu erstellen. Ersetzen Sie `us-west-2` durch die Region, in der Sie Ihren Cluster erstellen möchten.

   ```
   aws ec2 create-key-pair --region us-west-2 --key-name myKeyPair
   ```

   Speichern Sie die Rückgabeausgabe in einer Datei auf Ihrem lokalen Computer speichern. Weitere Informationen finden Sie unter Erstellen oder Importieren eines Schlüsselpaars im Amazon-EC2-Benutzerhandbuch für Linux-Instances.

   > **Note**
   >
   > Für die Erstellung eines EKS-Clusters ist kein Schlüsselpaar erforderlich. Durch Angabe des Schlüsselpaars, können Sie SSH-Verbindungen zu Knoten herstellen, sobald diese

erstellt wurden. Sie können ein Schlüsselpaar nur angeben, wenn Sie die Knotengruppe erstellen.

2. Erstellen Sie einen EKS-Cluster.

   Führen Sie den folgenden Befehl aus, um einen EKS-Cluster und einen Knoten zu erstellen. Ersetzen Sie *my-cluster* und *myKeyPair*durch Ihren eigenen Clusternamen und Schlüsselpaarnamen. Ersetzen Sie *us-west-2* mit der Region in der Sie Ihren Cluster bereitstellen möchten. Weitere Informationen zu den von Amazon EKS unterstützten Regionen finden Sie unter Serviceendpunkte und Kontingente für Amazon Elastic Kubernetes.

   ```
   eksctl create cluster \
   --name my-cluster \
   --region us-west-2 \
   --with-oidc \
   --ssh-access \
   --ssh-public-key myKeyPair \
   --instance-types=m5.xlarge \
   --managed
   ```

   Important

   Achten Sie beim Erstellen eines EKS-Clusters darauf, m5.xlarge als Instance-Typ oder einen anderen Instance-Typ mit einer höheren CPU und einem höheren Arbeitsspeicher zu verwenden. Die Verwendung eines Instance-Typs mit niedrigerer CPU oder weniger Arbeitsspeicher als m5.xlarge kann aufgrund unzureichender verfügbarer Ressourcen im Cluster zu Auftragsfehlern führen. Um alle erstellten Ressourcen anzuzeigen, sehen Sie sich den `eksctl-my-cluster-cluster` benannten Stack in der AWS -Cloud-Formation-Konsole an.

   Die Cluster- und Knotenerstellung dauert mehrere Minuten. Während der Erstellung der Cluster und Knoten werden Ihnen mehrere Ausgabezeilen angezeigt. Das folgende Beispiel illustriert die letzte Ausgabezeile.

   ```
   ...
   [#]  EKS cluster "my-cluster" in "us-west-2" region is ready
   ```

`eksctl` hat eine `kubectl`-config-Datei in `~/.kube` erstellt oder die Konfiguration des neuen Clusters innerhalb einer vorhandenen config-Datei in `~/.kube` hinzugefügt.

3. Ressourcen anzeigen und validieren

   Führen Sie den folgenden Befehl aus, um einen Cluster Knoten anzuzeigen.

   ```
   kubectl get nodes -o wide
   ```

   Das folgende Beispiel zeigt eine Ausgabe.

   ```
   Amazon EC2 node output

   NAME                                          STATUS   ROLES    AGE    VERSION
             INTERNAL-IP      EXTERNAL-IP     OS-IMAGE          KERNEL-VERSION
            CONTAINER-RUNTIME
   ip-192-168-12-49.us-west-2.compute.internal    Ready    none     6m7s
    v1.18.9-eks-d1db3c   192.168.12.49    52.35.116.65    Amazon Linux 2
    4.14.209-160.335.amzn2.x86_64   docker://19.3.6
   ip-192-168-72-129.us-west-2.compute.internal   Ready    none     6m4s
    v1.18.9-eks-d1db3c   192.168.72.129   44.242.140.21   Amazon Linux 2
    4.14.209-160.335.amzn2.x86_64   docker://19.3.6
   ```

   Weitere Informationen finden Sie unter Knoten anzeigen.

   Verwenden Sie den folgenden Befehl, um die Workloads anzuzeigen, die auf Ihrem Cluster ausgeführt werden.

   ```
   kubectl get pods --all-namespaces -o wide
   ```

   Das folgende Beispiel zeigt eine Ausgabe.

   ```
   Amazon EC2 output

   NAMESPACE     NAME                        READY   STATUS    RESTARTS   AGE     IP
                 NODE                                            NOMINATED NODE
    READINESS GATES
   kube-system   aws-node-6ctpm              1/1     Running   0          7m43s
    192.168.72.129   ip-192-168-72-129.us-west-2.compute.internal   none
    none
   ```

```
kube-system   aws-node-cbntg             1/1     Running   0          7m46s
 192.168.12.49    ip-192-168-12-49.us-west-2.compute.internal    none
 none
kube-system   coredns-559b5db75d-26t47   1/1     Running   0          14m
 192.168.78.81    ip-192-168-72-129.us-west-2.compute.internal   none
 none
kube-system   coredns-559b5db75d-9rvnk   1/1     Running   0          14m
 192.168.29.248   ip-192-168-12-49.us-west-2.compute.internal    none
 none
kube-system   kube-proxy-l8pbd           1/1     Running   0          7m46s
 192.168.12.49    ip-192-168-12-49.us-west-2.compute.internal    none
 none
kube-system   kube-proxy-zh85h           1/1     Running   0          7m43s
 192.168.72.129   ip-192-168-72-129.us-west-2.compute.internal   none
 none
```

Weitere Informationen zu dem, was Sie hier sehen, finden Sie unter Workloads anzeigen.

Erstellen Sie einen EKS-Cluster mit AWS Management Console und AWS CLI

Sie können auch AWS Management Console und verwenden AWS CLI , um einen EKS-Cluster zu erstellen. Folgen Sie den Schritten unter Erste Schritte mit Amazon EKS — AWS Management Console und AWS CLI. Auf diese Weise erhalten Sie Einblick in die Art und Weise, wie jede Ressource für den EKS-Cluster erstellt wird und wie die Ressourcen miteinander interagieren.

> **Important**
>
> Achten Sie beim Erstellen von Knoten für einen EKS-Cluster darauf, m5.xlarge als Instance-Typ oder einen anderen Instance-Typ mit einer höheren CPU und einem höheren Arbeitsspeicher zu verwenden.

Einen EKS-Cluster mit AWS Fargate erstellen

Sie können auch einen EKS-Cluster erstellen, auf dem Pods in AWS Fargate ausgeführt werden.

1. Um einen EKS-Cluster mit Pods zu erstellen, die auf Fargate ausgeführt werden, folgen Sie den Schritten unter Erste Schritte AWS Fargate mit Amazon EKS.

> **Note**
>
> Amazon EMR in EKS benötigt CoreDNS für die Ausführung von Aufträge auf dem EKS-Cluster. Wenn Sie Ihre Pods nur auf Fargate ausführen möchten, müssen Sie die Schritte unter CoreDNS aktualisieren befolgen.

2. Führen Sie den folgenden Befehl aus, um einen Cluster Knoten anzuzeigen.

```
kubectl get nodes -o wide
```

Das folgende Beispiel zeigt eine Fargate-Ausgabe.

```
Fargate node output

NAME                                                  STATUS   ROLES    AGE
   VERSION              INTERNAL-IP       EXTERNAL-IP   OS-IMAGE         KERNEL-
VERSION                   CONTAINER-RUNTIME
fargate-ip-192-168-141-147.us-west-2.compute.internal   Ready    none
 8m3s    v1.18.8-eks-7c9bda   192.168.141.147   none          Amazon Linux 2
 4.14.209-160.335.amzn2.x86_64   containerd://1.3.2
fargate-ip-192-168-164-53.us-west-2.compute.internal    Ready    none
 7m30s   v1.18.8-eks-7c9bda   192.168.164.53    none          Amazon Linux 2
 4.14.209-160.335.amzn2.x86_64   containerd://1.3.2
```

Weitere Informationen finden Sie unter Knoten anzeigen.

3. Führen Sie den folgenden Befehl aus, um die Workloads anzuzeigen, die auf Ihrem Cluster ausgeführt werden.

```
kubectl get pods --all-namespaces -o wide
```

Das folgende Beispiel zeigt eine Fargate-Ausgabe.

```
Fargate output

NAMESPACE     NAME                        READY   STATUS    RESTARTS   AGE   IP
           NODE                                                  NOMINATED NODE
 READINESS GATES
```

```
kube-system   coredns-69dfb8f894-9z95l   1/1     Running   0          18m
 192.168.164.53    fargate-ip-192-168-164-53.us-west-2.compute.internal    none
         none
kube-system   coredns-69dfb8f894-c8v66   1/1     Running   0          18m
 192.168.141.147   fargate-ip-192-168-141-147.us-west-2.compute.internal   none
         none
```

Weitere Informationen finden Sie unter Workloads anzeigen.

Gewähren eines Clusterzugriff für Amazon EMR in EKS

Sie müssen Amazon EMR in EKS Zugriff auf einen bestimmten Namespace in Ihrem Cluster gewähren, indem Sie die folgenden Aktionen ausführen: eine Kubernetes-Rolle erstellen, die Rolle an einen Kubernetes-Benutzer binden und den Kubernetes-Benutzer der serviceverknüpften Rolle `AWSServiceRoleForAmazonEMRContainers` zuordnen. Diese Aktionen werden in `eksctl` automatisiert, wenn der IAM-Befehl zur Identitätszuweisung zusammen mit `emr-containers` als Servicenamen verwendet wird. Sie können diese Vorgänge einfach durchführen, indem Sie den folgenden Befehl verwenden.

```
eksctl create iamidentitymapping \
    --cluster my_eks_cluster \
    --namespace kubernetes_namespace \
    --service-name "emr-containers"
```

Ersetzen Sie *`my_eks_cluster`* durch den Namen Ihres Amazon-EKS-Clusters und ersetzen Sie *`kubernetes_namespace`* durch den Kubernetes-Namespace, der für die Ausführung von Amazon-EMR-Workloads erstellt wurde.

> **Important**
>
> Sie müssen die neueste Version von eksctl mithilfe des vorherigen Schritts Installieren `eksctl` herunterladen, um diese Funktion nutzen zu können.

Manuelle Schritte zur Aktivierung des Clusterzugriffs für Amazon EMR in EKS

Sie können auch die folgenden manuellen Schritte verwenden, um den Clusterzugriff für Amazon EMR in EKS zu aktivieren.

1. Erstellen Sie eine Kubernetes-Rolle in einem bestimmten Namespace

   Amazon EKS 1.22 - 1.29

   Führen Sie mit Amazon EKS 1.22 — 1.29 den folgenden Befehl aus, um eine Kubernetes-Rolle in einem bestimmten Namespace zu erstellen. Diese Rolle gewährt Amazon EMR in EKS die erforderlichen RBAC-Berechtigungen.

```
namespace=my-namespace
cat - >>EOF | kubectl apply -f - >>namespace "${namespace}"
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: emr-containers
  namespace: ${namespace}
rules:
  - apiGroups: [""]
    resources: ["namespaces"]
    verbs: ["get"]
  - apiGroups: [""]
    resources: ["serviceaccounts", "services", "configmaps", "events", "pods",
 "pods/log"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "deletecollection", "annotate", "patch", "label"]
  - apiGroups: [""]
    resources: ["secrets"]
    verbs: ["create", "patch", "delete", "watch"]
  - apiGroups: ["apps"]
    resources: ["statefulsets", "deployments"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "annotate", "patch", "label"]
  - apiGroups: ["batch"]
    resources: ["jobs"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "annotate", "patch", "label"]
  - apiGroups: ["extensions", "networking.k8s.io"]
    resources: ["ingresses"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "annotate", "patch", "label"]
  - apiGroups: ["rbac.authorization.k8s.io"]
    resources: ["roles", "rolebindings"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "deletecollection", "annotate", "patch", "label"]
```

```
  - apiGroups: [""]
    resources: ["persistentvolumeclaims"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "deletecollection", "annotate", "patch", "label"]
EOF
```

Amazon EKS 1.21 and below

Führen Sie bei Amazon EKS 1.21 und niedriger den folgenden Befehl aus, um eine Kubernetes-Rolle in einem bestimmten Namespace zu erstellen. Diese Rolle gewährt Amazon EMR in EKS die erforderlichen RBAC-Berechtigungen.

```
namespace=my-namespace
cat - >>EOF | kubectl apply -f - >>namespace "${namespace}"
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: emr-containers
  namespace: ${namespace}
rules:
  - apiGroups: [""]
    resources: ["namespaces"]
    verbs: ["get"]
  - apiGroups: [""]
    resources: ["serviceaccounts", "services", "configmaps", "events", "pods",
 "pods/log"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "deletecollection", "annotate", "patch", "label"]
  - apiGroups: [""]
    resources: ["secrets"]
    verbs: ["create", "patch", "delete", "watch"]
  - apiGroups: ["apps"]
    resources: ["statefulsets", "deployments"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "annotate", "patch", "label"]
  - apiGroups: ["batch"]
    resources: ["jobs"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "annotate", "patch", "label"]
  - apiGroups: ["extensions"]
    resources: ["ingresses"]
```

```
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "annotate", "patch", "label"]
  - apiGroups: ["rbac.authorization.k8s.io"]
    resources: ["roles", "rolebindings"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "deletecollection", "annotate", "patch", "label"]
  - apiGroups: [""]
    resources: ["persistentvolumeclaims"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "deletecollection", "annotate", "patch", "label"]
EOF
```

2. Eine Kubernetes-Rollenbindung mit Gültigkeitsbereich für den Namespace erstellen

   Führen Sie den folgenden Befehl aus, um eine Kubernetes-Rollenbindung im angegebenen Namespace zu erstellen. Diese Rollenbindung gewährt einem Benutzer mit dem Namen `emr-containers` die in der im vorherigen Schritt erstellten Rolle definierten Berechtigungen. Dieser Benutzer identifiziert serviceverknüpfte Rollen für Amazon EMR in EKS und ermöglicht somit Amazon EMR in EKS, Aktionen auszuführen, die in der von Ihnen erstellten Rolle definiert sind.

```
namespace=my-namespace

cat - <<EOF | kubectl apply -f - --namespace "${namespace}"
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: emr-containers
  namespace: ${namespace}
subjects:
- kind: User
  name: emr-containers
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: emr-containers
  apiGroup: rbac.authorization.k8s.io
EOF
```

3. Aktualisieren Sie die **`aws-auth`**-Kubernetes-Konfigurationsübersicht

Sie können eine der folgenden Optionen verwenden, um die serviceverknüpfte Rolle Amazon EMR in EKS dem `emr-containers`-Benutzer zuzuordnen, der im vorherigen Schritt an die Kubernetes-Rolle gebunden war.

Option 1: Verwenden von **`eksctl`**

Führen Sie den folgenden `eksctl`-Befehl aus, um die serviceverknüpfte Rolle Amazon EMR in EKS dem `emr-containers`-Benutzer zuzuordnen.

```
eksctl create iamidentitymapping \
    --cluster my-cluster-name \
    --arn "arn:aws:iam::my-account-id:role/AWSServiceRoleForAmazonEMRContainers" \
    --username emr-containers
```

Option 2: Ohne eksctl zu verwenden

1. Führen Sie den folgenden Befehl aus, um die `aws-auth`-Konfigurationsübersicht im Texteditor zu öffnen.

   ```
   kubectl edit -n kube-system configmap/aws-auth
   ```

   > **Note**
   >
   > Wenn Sie eine Fehlermeldung erhalten`Error from server (NotFound): configmaps "aws-auth" not found`, lesen Sie die Schritte unter Benutzerrollen hinzufügen im Amazon EKS-Benutzerhandbuch, um den Bestand anzuwenden ConfigMap.

2. Fügen Sie die mit dem Service verknüpften Rollendetails von Amazon EMR in EKS zum `mapRoles`-Abschnitt von `ConfigMap` unter `data` hinzu. Fügen Sie diesen Abschnitt hinzu, wenn er nicht bereits in der Datei vorhanden sind. Der aktualisierte `mapRoles`-Abschnitt unter Daten sieht wie das folgende Beispiel aus.

   ```
   apiVersion: v1
   data:
     mapRoles: |
       - rolearn: arn:aws:iam::<your-account-id>:role/
   AWSServiceRoleForAmazonEMRContainers
   ```

```
      username: emr-containers
  - ... <other previously existing role entries, if there's any>.
```

3. Speichern Sie die Datei und beenden Sie den Text-Editor.

Automatisieren Sie die Aktivierung des Clusterzugriffs für Amazon EMR auf EKS

Amazon EMR ist in Amazon EKS Cluster Access Management (CAM) integriert, sodass Sie die Konfiguration der erforderlichen AuthN- und AuthZ-Richtlinien automatisieren können, um Amazon EMR Spark-Jobs in Namespaces von Amazon EKS-Clustern auszuführen. Wenn Sie einen virtuellen Cluster aus einem Amazon EKS-Cluster-Namespace erstellen, konfiguriert Amazon EMR automatisch alle erforderlichen Berechtigungen, sodass Sie Ihren aktuellen Workflows keine zusätzlichen Schritte hinzufügen müssen.

> **Note**
>
> Der Amazon EKS-Zugriffseintrag unterstützt maximal nur 100 Namespaces. Wenn Sie über mehr als 100 virtuelle Cluster verfügen, verwendet Amazon EMR die Access Entry APIs nicht, wenn Sie neue virtuelle Cluster erstellen. Sie können sehen, für welche Cluster die Access Entry Integration aktiviert ist, indem Sie den `eksAccessEntryIntegrated` Parameter auf true setzen, wenn Sie den `ListVirtualClusters` API-Vorgang oder den `list-virtual-clusters` CLI-Befehl ausführen. Der Befehl gibt die eindeutigen Bezeichner aller zutreffenden virtuellen Cluster zurück.

Voraussetzungen

- Stellen Sie sicher, dass Sie Version 2.15.3 oder höher von ausführen AWS CLI
- Ihr Amazon EKS-Cluster muss Version 1.23 oder höher haben.

Aufstellen

Um die Integration zwischen Amazon EMR und den AccessEntry API-Vorgängen von Amazon EKS einzurichten, stellen Sie sicher, dass Sie die folgenden Punkte abgeschlossen haben:

- Stellen Sie sicher, dass `authenticationMode` Ihr Amazon EKS-Cluster auf eingestellt ist`API_AND_CONFIG_MAP`.

```
aws eks describe-cluster --name <eks-cluster-name>
```

Falls dies noch nicht der Fall ist, stellen Sie es `authenticationMode` auf ein`API_AND_CONFIG_MAP`.

```
aws eks update-cluster-config
    --name <eks-cluster-name>
    --access-config authenticationMode=API_AND_CONFIG_MAP
```

Weitere Informationen zu Authentifizierungsmodi finden Sie unter Cluster-Authentifizierungsmodi.

- Stellen Sie sicher, dass die IAM-Rolle, die Sie zum Ausführen der `CreateVirtualCluster` und `DeleteVirtualCluster` API-Operationen verwenden, auch über die folgenden Berechtigungen verfügt:

```
{
        "Effect": "Allow",
        "Action": [
            "eks:DescribeAccessEntry",
            "eks:CreateAccessEntry",
            "eks:DeleteAccessEntry",
            "eks:ListAssociatedAccessPolicies",
            "eks:AssociateAccessPolicy",
            "eks:DisassociateAccessPolicy"
        ],
        "Resource": "*"
}
```

Konzepte und Terminologie

Im Folgenden finden Sie eine Liste von Terminologien und Konzepten im Zusammenhang mit Amazon EKS CAM.

- Virtueller Cluster (VC) — logische Darstellung des in Amazon EKS erstellten Namespaces. Es ist ein 1:1 -Link zu einem Amazon EKS-Cluster-Namespace. Sie können damit Amazon EMR-Workloads auf einem Amazon EKS-Cluster innerhalb des angegebenen Namespace ausführen.
- Namespace — Mechanismus zum Isolieren von Ressourcengruppen innerhalb eines einzelnen EKS-Clusters.

- Zugriffsrichtlinie — Berechtigungen, die einer IAM-Rolle innerhalb eines EKS-Clusters Zugriff und Aktionen gewähren.
- Zugriffseintrag — ein Eintrag, der mit einer Rolle arn erstellt wurde. Sie können den Zugriffseintrag mit einer Zugriffsrichtlinie verknüpfen, um bestimmte Berechtigungen im Amazon EKS-Cluster zuzuweisen.
- Integrierter virtueller EKS-Zugangscluster — der virtuelle Cluster, der mithilfe von Access Entry API-Operationen von Amazon EKS erstellt wurde.

Aktivieren Sie IAM-Rollen für Servicekonten (IRSA) auf dem EKS-Cluster

Das Feature IAM-Rollen für Servicekonten ist auf der neuen Amazon-EKS-Version 1.14 und höher und auf EKS Clustern verfügbar, die am oder nach dem 3. September 2019 auf die Version 1.13 oder höher aktualisiert wurden. Um dieses Feature zu nutzen, können Sie vorhandene EKS-Cluster auf Version 1.14 oder höher aktualisieren. Weitere Informationen finden Sie unter Aktualisieren einer Amazon-EKS-Cluster-Kubernetes-Version.

Wenn Ihr Cluster IAM-Rollen für Servicekonten unterstützt, ist ihm eine OpenID-Connect-Aussteller-URL zugeordnet. Sie können diese URL in der Amazon EKS-Konsole anzeigen oder den folgenden AWS CLI Befehl verwenden, um sie abzurufen.

> **Important**
>
> Sie müssen die neueste Version von verwenden AWS CLI , um die korrekte Ausgabe dieses Befehls zu erhalten.

```
aws eks describe-cluster --name cluster_name --query "cluster.identity.oidc.issuer" --
output text
```

Die erwartete Ausgabe sieht wie folgt aus.

```
https://oidc.eks.<region-code>.amazonaws.com/id/EXAMPLED539D4633E53DE1B716D3041E
```

Um IAM-Rollen für Servicekonten in Ihrem Cluster zu verwenden, müssen Sie einen OIDC-Identitätsanbieter entweder mit eksctl oder AWS Management Console erstellen.

So erstellen Sie einen IAM-OIDC-Identitätsanbieter für Ihren Cluster mit **`eksctl`**

Sie können Ihre `eksctl`-Version mit dem folgenden Befehl überprüfen. Bei diesem Verfahren wird davon ausgegangen, dass Sie `eksctl` installiert haben und dass Ihre `eksctl`-Version mindestens 0.32.0 oder höher ist.

```
eksctl version
```

Weitere Informationen über die Installation oder Aktualisierung von eksctl finden Sie unter Installieren oder Aktualisieren von eksctl.

Erstellen Sie Ihren OIDC-Identitätsanbieter für Ihren Cluster mit dem folgenden Befehl. Ersetzen Sie *`cluster_name`* durch Ihren eigenen Wert.

```
eksctl utils associate-iam-oidc-provider --cluster cluster_name --approve
```

Um einen IAM-OIDC-Identitätsanbieter für Ihren Cluster mit dem zu erstellen AWS Management Console

Rufen Sie die OIDC-Aussteller-URL aus der Amazon EKS-Konsolenbeschreibung Ihres Clusters ab, oder verwenden Sie den folgenden Befehl. AWS CLI

Verwenden Sie den folgenden Befehl, um die OIDC-Aussteller-URL aus der AWS CLI abzurufen.

```
aws eks describe-cluster --name <cluster_name> --query "cluster.identity.oidc.issuer"
 --output text
```

Führen Sie die folgenden Schritte aus, um die OIDC-Aussteller-URL von der Amazon-EKS-Konsole abzurufen.

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie im Navigationsbereich Identitätsanbieter und dann Anbieter erstellen aus.
    1. Wählen Sie für Provider Type (Anbietertyp) die Option Choose a provider type (Anbietertyp auswählen) und dann OpenID Connect aus.
    2. Fügen Sie unter Provider URL (Anbieter-URL) die OIDC-Aussteller-URL für Ihren Cluster ein.
    3. Geben Sie für Zielgruppe sts.amazonaws.com ein und wählen Sie Nächster Schritt aus.
3. Überprüfen Sie, ob die Anbieterinformationen korrekt sind, und wählen Sie dann Create (Erstellen) aus, um Ihren Identitätsanbieter zu erstellen.

Erstellen einer Aufgabenausführungsrolle

Um Workloads auf Amazon EMR in EKS auszuführen, müssen Sie eine IAM-Rolle erstellen. In dieser Dokumentation wird diese Rolle als Auftragsausführungsrolle bezeichnet. Weitere Informationen zum Erstellen von IAM-Rollen finden Sie unter Erstellen von IAM-Rollen im IAM-Benutzerhandbuch.

Sie müssen außerdem eine IAM-Richtlinie erstellen, die die Berechtigungen für die Auftragsausführungsrolle festlegt, und dann die IAM-Richtlinie an die Auftragsausführungsrolle anhängen.

Die folgende Richtlinie für die Jobausführungsrolle ermöglicht den Zugriff auf Ressourcenziele, Amazon S3 und CloudWatch. Diese Berechtigungen sind erforderlich, um Aufträge und Zugriffsprotokolle zu überwachen. Um den gleichen Prozess mit dem zu verfolgen AWS CLI, können Sie Ihre Rolle auch mithilfe der Schritte im Abschnitt Create IAM Role for Job Execution des Amazon EMR on EKS Workshops einrichten.

> **Note**
>
> Der Zugriff sollte angemessen begrenzt sein und nicht allen S3-Objekten in der Rolle Aufgabenausführung gewährt werden.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": "arn:aws:s3:::example-bucket"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents",
                "logs:CreateLogStream",
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams"
```

```
            ],
            "Resource": [
                "arn:aws:logs:*:*:*"
            ]
        }
    ]
}
```

Weitere Informationen finden Sie unter Verwenden von Rollen zur Auftragsausführung, Konfigurieren einer Auftragsausführung für die Verwendung von S3-Protokollen und Konfigurieren einer Auftragsausführung für die Verwendung CloudWatch von Protokollen.

Die Vertrauensrichtlinie der Auftragsausführungsrolle aktualisieren

Wenn Sie IAM-Rollen für Servicekonten (IRSA) verwenden, um Aufträge in einem Kubernetes-Namespace auszuführen, muss ein Administrator eine Vertrauensbeziehung zwischen der Auftragsausführungsrolle und der Identität des EMR-verwalteten Servicekontos einrichten. Die Vertrauensbeziehung kann hergestellt werden, indem die Vertrauensrichtlinie der Auftragsausführungsrolle aktualisiert wird. Beachten Sie, dass das EMR-verwaltete Servicekonto bei der Aufgabenübermittlung automatisch erstellt wird und auf den Namespace beschränkt ist, in dem die Aufgabe eingereicht wird.

Um die Vertrauensrichtlinie zu aktualisieren, führen Sie den folgenden Befehl aus.

```
aws emr-containers update-role-trust-policy \
      --cluster-name cluster \
      --namespace namespace \
      --role-name iam_role_name_for_job_execution
```

Weitere Informationen finden Sie unter Auftragausführungsrollen mit Amazon EMR in EKS verwenden.

> **Important**
>
> Der Operator, der den obigen Befehl ausführt, muss über die folgenden Berechtigungen verfügen: `eks:DescribeCluster` `iam:GetRole` `iam:UpdateAssumeRolePolicy`.

Gewähren Sie Benutzern Zugriff auf Amazon EMR in EKS

Für alle Aktionen, die Sie auf Amazon EMR in EKS ausführen, benötigen Sie eine entsprechende IAM-Berechtigung für diese Aktion. Sie müssen eine IAM-Richtlinie erstellen, die es Ihnen ermöglicht, Amazon EMR in EKS durchzuführen, und die Richtlinie an den IAM-Benutzer oder die IAM-Rolle anfügen, die Sie verwenden.

Dieses Thema enthält Schritte zum Erstellen einer neuen Richtlinie und zum Anhängen dieser an einen Benutzer. Es behandelt auch die grundlegenden Berechtigungen, die Sie für die Einrichtung Ihrer Umgebung von Amazon EMR in EKS benötigen. Wir empfehlen Ihnen, die Berechtigungen für bestimmte Ressourcen nach Möglichkeit an Ihre Geschäftsanforderungen anzupassen.

Eine neue IAM-Richtlinie erstellen und sie einem Benutzer in der IAM-Konsole zuordnen

Eine neue IAM-Richtlinie erstellen

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie im linken Navigationsbereich der IAM-Konsole Richtlinien aus.
3. Wählen Sie auf der Seite Policies (Richtlinien) die Option Create a policy (Richtlinie erstellen).
4. Navigieren Sie im Fenster Richtlinie erstellen zur Registerkarte JSON bearbeiten. Erstellen Sie ein Richtliniendokument mit einer oder mehreren JSON-Anweisungen, wie in den Beispielen nach diesem Verfahren gezeigt. Wählen Sie als Nächstes die Option Richtlinie überprüfen aus.
5. Geben Sie auf Bildschirm Review Policy (Richtlinie überprüfen) Ihren Policy Name (Richtlinienname) ein, z. B. `AmazonEMROnEKSPolicy`. Geben Sie eine optionale Beschreibung für Ihre Richtlinie ein und wählen Sie dann Richtlinie erstellen aus.

Die Richtlinie an einen Benutzer oder eine Rolle anhängen

1. Melden Sie sich bei der IAM-Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/iam/
2. Wählen Sie im Navigationsbereich Richtlinien.
3. Aktivieren Sie in der Liste der Richtlinien das Kontrollkästchen neben der im vorherigen Abschnitt erstellten Richtlinie. Über das Menü Filter und das Suchfeld können Sie die Richtlinienliste filtern.
4. Klicken Sie auf Policy actions und anschließend auf Attach.
5. Wählen Sie den Benutzer oder die Rolle aus, an den Sie die Richtlinie anfügen möchten. Über das Menü Filter und das Suchfeld können Sie die Liste der Prinzipal-Entitäten filtern. Nachdem

Sie den Benutzer oder die Rolle zum Anfügen der Richtlinie ausgewählt haben, klicken Sie auf Richtlinie anfügen.

Berechtigungen zum Verwalten virtueller Cluster

Um virtuelle Cluster in Ihrem AWS Konto zu verwalten, erstellen Sie eine IAM-Richtlinie mit den folgenden Berechtigungen. Mit diesen Berechtigungen können Sie virtuelle Cluster in Ihrem AWS Konto erstellen, auflisten, beschreiben und löschen.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "emr-containers.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "emr-containers:CreateVirtualCluster",
                "emr-containers:ListVirtualClusters",
                "emr-containers:DescribeVirtualCluster",
                "emr-containers:DeleteVirtualCluster"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon EMR ist in Amazon EKS Cluster Access Management (CAM) integriert, sodass Sie die Konfiguration der erforderlichen AuthN- und AuthZ-Richtlinien automatisieren können, um Amazon EMR Spark-Jobs in Namespaces von Amazon EKS-Clustern auszuführen. Dazu benötigen Sie die folgenden Berechtigungen:

```
{
        "Effect": "Allow",
        "Action": [
            "eks:DescribeAccessEntry",
            "eks:CreateAccessEntry",
            "eks:DeleteAccessEntry",
            "eks:ListAssociatedAccessPolicies",
            "eks:AssociateAccessPolicy",
            "eks:DisassociateAccessPolicy"
        ],
        "Resource": "*"
}
```

Weitere Informationen finden Sie unter Automatisieren der Aktivierung des Clusterzugriffs für Amazon EMR auf EKS.

Wenn der `CreateVirtualCluster` Vorgang zum ersten Mal von einem AWS Konto aus aufgerufen wird, benötigen Sie auch die `CreateServiceLinkedRole` Berechtigungen, um die serviceverknüpfte Rolle für Amazon EMR auf EKS zu erstellen. Weitere Informationen finden Sie unter Verwendung von serviceverknüpften Rollen für Amazon EMR in EKS.

Berechtigungen für das Einreichen von Aufträge

Um Jobs auf den virtuellen Clustern in Ihrem AWS Konto einzureichen, erstellen Sie eine IAM-Richtlinie mit den folgenden Berechtigungen. Mit diesen Berechtigungen können Sie Auftragsausführungen für alle virtuellen Cluster in Ihrem Konto starten, auflisten, beschreiben und abbrechen. Sie sollten in Betracht ziehen, Berechtigungen hinzuzufügen, um virtuelle Cluster aufzulisten oder zu beschreiben, sodass Sie den Status des virtuellen Clusters überprüfen können, bevor Sie Aufträge einreichen.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "emr-containers:StartJobRun",
                "emr-containers:ListJobRuns",
                "emr-containers:DescribeJobRun",
                "emr-containers:CancelJobRun"
            ],
```

```
            "Resource": "*"
        }
    ]
}
```

Berechtigungen für das Debuggen und Überwachen

Um Zugriff auf an Amazon S3 übertragene Protokolle zu erhalten oder um Anwendungsereignisprotokolle in der Amazon EMR-Konsole anzuzeigen, erstellen Sie eine IAM-Richtlinie mit den folgenden Berechtigungen. CloudWatch Wir empfehlen Ihnen, die Berechtigungen für bestimmte Ressourcen nach Möglichkeit an Ihre Geschäftsanforderungen anzupassen.

> **Important**
>
> Wenn Sie keinen Amazon-S3-Bucket erstellt haben, müssen Sie der Richtlinienerklärung `s3:CreateBucket`-Berechtigungen hinzufügen. Wenn Sie keine Protokollgruppe erstellt haben, müssen Sie `logs:CreateLogGroup` der Richtlinienerklärung hinzufügen.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "emr-containers:DescribeJobRun",
                "elasticmapreduce:CreatePersistentAppUI",
                "elasticmapreduce:DescribePersistentAppUI",
                "elasticmapreduce:GetPersistentAppUIPresignedURL"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": "*"
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "logs:Get*",
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams"
            ],
            "Resource": "*"
        }
    ]
}
```

Weitere Informationen zur Konfiguration einer Auftragsausführung für die Übertragung von Protokollen an Amazon S3 finden Sie unter Konfiguration einer Auftragsausführung zur Verwendung von S3-Protokollen und Konfiguration einer Auftragsausführung zur Verwendung von CloudWatch Protokollen. CloudWatch

Den Amazon-EKS-Cluster mit Amazon EMR registrieren

Die Registrierung Ihres Clusters ist der letzte erforderliche Schritt, um Amazon EMR in EKS für die Ausführung von Workloads einzurichten.

Verwenden Sie den folgenden Befehl, um einen virtuellen Cluster mit einem Namen Ihrer Wahl für den Amazon-EKS-Cluster und den Namespace zu erstellen, die Sie in den vorherigen Schritten eingerichtet haben.

> **Note**
>
> Jeder virtuelle Cluster muss in allen EKS-Clustern einen eindeutigen Namen haben. Wenn zwei virtuelle Cluster denselben Namen haben, schlägt der Bereitstellungsprozess fehl, auch wenn die beiden virtuellen Cluster zu unterschiedlichen EKS-Clustern gehören.

```
aws emr-containers create-virtual-cluster \
--name virtual_cluster_name \
--container-provider '{
    "id": "cluster_name",
    "type": "EKS",
    "info": {
        "eksInfo": {
            "namespace": "namespace_name"
        }
```

```
    }
}'
```

Alternativ können Sie eine JSON-Datei erstellen, die die erforderlichen Parameter für den virtuellen Cluster enthält, und dann den `create-virtual-cluster`-Befehl mit dem Pfad zur JSON-Datei ausführen. Weitere Informationen finden Sie unter Verwaltung virtueller Cluster.

> **Note**
>
> Um die erfolgreiche Erstellung eines virtuellen Clusters zu überprüfen, zeigen Sie den Status der virtuellen Cluster an, indem Sie den Vorgang `list-virtual-clusters` verwenden oder die Seite Virtuelle Cluster in der Amazon-EMR-Konsole aufrufen.

Reichen Sie einen Auftrag ein, der ausgeführt wird mit **`StartJobRun`**

Um einen Auftrag einzureichen, führen Sie ihn mit einer JSON-Datei mit angegebenen Parametern aus

1. Erstellen Sie eine `start-job-run-request.json`-Datei und geben Sie die erforderlichen Parameter für Ihre Auftragausführung an, wie die folgende JSON-Beispieldatei zeigt. Weitere Informationen zu den Parametern finden Sie unter Optionen für die Konfiguration einer Aufgabenausführung.

   ```
   {
     "name": "myjob",
     "virtualClusterId": "123456",
     "executionRoleArn": "iam_role_name_for_job_execution",
     "releaseLabel": "emr-6.2.0-latest",
     "jobDriver": {
       "sparkSubmitJobDriver": {
         "entryPoint": "entryPoint_location",
         "entryPointArguments": ["argument1", "argument2", ...],
          "sparkSubmitParameters": "--class <main_class> --conf
    spark.executor.instances=2 --conf spark.executor.memory=2G --conf
    spark.executor.cores=2 --conf spark.driver.cores=1"
       }
     },
     "configurationOverrides": {
       "applicationConfiguration": [
         {
   ```

```
        "classification": "spark-defaults",
        "properties": {
          "spark.driver.memory":"2G"
         }
      }
    ],
    "monitoringConfiguration": {
      "persistentAppUI": "ENABLED",
      "cloudWatchMonitoringConfiguration": {
        "logGroupName": "my_log_group",
        "logStreamNamePrefix": "log_stream_prefix"
      },
      "s3MonitoringConfiguration": {
        "logUri": "s3://my_s3_log_location"
      }
    }
  }
}
```

2. Verwenden Sie den `start-job-run`-Befehl mit einem Pfad zu der lokal gespeicherten `start-job-run-request.json`-Datei.

```
aws emr-containers start-job-run \
--cli-input-json file://./start-job-run-request.json
```

So starten Sie eine Auftragsausführung mithilfe des **`start-job-run`**-Befehls

1. Geben Sie alle angegebenen Parameter im `StartJobRun`-Befehl an, wie das folgende Beispiel zeigt.

```
aws emr-containers start-job-run \
--virtual-cluster-id 123456 \
--name myjob \
--execution-role-arn execution-role-arn \
--release-label emr-6.2.0-latest \
--job-driver '{"sparkSubmitJobDriver": {"entryPoint": "entryPoint_location",
 "entryPointArguments": ["argument1", "argument2", ...], "sparkSubmitParameters":
 "--class <main_class> --conf spark.executor.instances=2 --conf
 spark.executor.memory=2G --conf spark.executor.cores=2 --conf
 spark.driver.cores=1"}}' \
--configuration-overrides '{"applicationConfiguration": [{"classification":
 "spark-defaults", "properties": {"spark.driver.memory": "2G"}}],
```

```
 "monitoringConfiguration": {"cloudWatchMonitoringConfiguration":
 {"logGroupName": "log_group_name", "logStreamNamePrefix": "log_stream_prefix"},
 "persistentAppUI":"ENABLED",  "s3MonitoringConfiguration": {"logUri":
 "s3://my_s3_log_location" }}}'
```

2. Geben Sie für Spark SQL alle angegebenen Parameter im `StartJobRun`-Befehl an, wie das folgende Beispiel zeigt.

```
aws emr-containers start-job-run \
--virtual-cluster-id 123456 \
--name myjob \
--execution-role-arn execution-role-arn \
--release-label emr-6.7.0-latest \
--job-driver '{"sparkSqlJobDriver": {"entryPoint": "entryPoint_location",
 "sparkSqlParameters": "--conf spark.executor.instances=2 --conf
 spark.executor.memory=2G --conf spark.executor.cores=2 --conf
 spark.driver.cores=1"}}' \
--configuration-overrides '{"applicationConfiguration": [{"classification":
 "spark-defaults", "properties": {"spark.driver.memory": "2G"}}],
 "monitoringConfiguration": {"cloudWatchMonitoringConfiguration":
 {"logGroupName": "log_group_name", "logStreamNamePrefix": "log_stream_prefix"},
 "persistentAppUI":"ENABLED",  "s3MonitoringConfiguration": {"logUri":
 "s3://my_s3_log_location" }}}'
```

Spark-Aufträge mit dem Spark-Operator ausführen

Amazon-EMR-Versionen 6.10.0 und höher unterstützen den Kubernetes-Operator für Apache Spark oder den Spark-Operator als Aufgabeneinreichungsmodell für Amazon EMR in EKS. Mit dem Spark-Operator können Sie Spark-Anwendungen mit der Amazon-EMR-Release-Laufzeit auf Ihren eigenen Amazon-EKS-Clustern bereitstellen und verwalten. Sobald Sie den Spark Operator in Ihrem Amazon-EKS-Cluster bereitgestellt haben, können Sie Spark-Anwendungen direkt beim Operator einreichen. Der Operator verwaltet den Lebenszyklus von Spark-Anwendungen.

> **Note**
>
> Amazon EMR berechnet die Preise für Amazon EKS auf der Grundlage des vCPU- und Speicherverbrauchs. Diese Berechnung gilt für Treiber- und Executor-Pods. Diese Berechnung beginnt mit dem Herunterladen Ihres Amazon EMR-Anwendungsabbilds, bis der Amazon EKS-Pod endet und auf die nächste Sekunde gerundet wird.

Themen

Einrichten des Spark-Operators für Amazon EMR in EKS

Führen Sie zum Einrichten des Spark-Operator auf Amazon EKS die folgenden Schritte aus. Wenn Sie bereits für Amazon Web Services (AWS) registriert sind und Amazon EKS schon verwendet haben, müssen Sie nur wenige Schritte ausführen, um Amazon EMR in EKS nutzen zu können. Führen Sie zum Einrichten von Amazon Spark Operator die folgenden Schritte auf Amazon EKS aus. Wenn Sie bereits eine der Voraussetzungen erfüllt haben, können Sie diese überspringen und mit der nächsten fortfahren.

- Installieren Sie das AWS CLI – Wenn Sie AWS CLI bereits installiert haben, vergewissern Sie sich, dass Sie über die neueste Version verfügen.
- eksctl installieren – eksctl ist ein Befehlszeilentool, das Sie für die Kommunikation mit Amazon EKS verwenden.
- Helm installieren – Der Helm-Paketmanager für Kubernetes unterstützt Sie bei der Installation und Verwaltung von Anwendungen in Ihrem Kubernetes-Cluster.
- Einen Amazon-EKS-Cluster einrichten – Folgen Sie den Schritten, um einen neuen Kubernetes-Cluster mit Knoten in Amazon EKS zu erstellen.
- Eine Amazon-EMR-Basis-Image-URI auswählen (Version 6.10.0 oder höher) aus – der Spark-Operator wird von Amazon-EMR-Versionen 6.10.0 und höher unterstützt.

Erste Schritte mit dem Spark-Operator für Amazon EMR in EKS

Dieses Thema hilft Ihnen beim Einstieg in die Verwendung des Spark-Operators in Amazon EKS, indem Sie eine Spark-Anwendung und eine Schedule-Spark-Anwendung bereitstellen.

Den Spark-Operator installieren

Gehen Sie wie folgt vor, um den Kubernetes-Operator für Apache Spark zu installieren.

1. Sofern noch nicht geschehen, führen die Schritte unter Einrichten des Spark-Operators für Amazon EMR in EKS aus.
2. Authentifizieren Ihren Helm-Client in Ihrer Amazon-ECR-Registry. Ersetzen Sie im folgenden Befehl die *Region-ID-Werte* durch Ihre bevorzugten AWS-Region-Werte und den entsprechenden *ECR-Registry-Account-Wert* für die Region auf der Seite Amazon-ECR-Registrierungskonten nach Regionen.

   ```
   aws ecr get-login-password \
   --region region-id | helm registry login \
   --username AWS \
   --password-stdin ECR-registry-account.dkr.ecr.region-id.amazonaws.com
   ```

3. Installieren Sie den Spark-Operator mit dem folgenden Befehl.

   Verwenden Sie für den `--version`-Helm-Chart-Parameter Ihre Amazon-EMR-Versionskennung, wobei das `emr`-Präfix und das Datumssuffix entfernt wurden. Geben Sie beispielsweise `emr-6.12.0-java17-latest` bei der Version `6.12.0-java17` an. Das Beispiel im folgenden Befehl verwendet die `emr-7.1.0-latest` Version, also gibt sie `7.1.0` für das Helm-Chart `--version` an.

   ```
   helm install spark-operator-demo \
     oci://895885662937.dkr.ecr.region-id.amazonaws.com/spark-operator \
     --set emrContainers.awsRegion=region-id \
     --version 7.1.0 \
     --namespace spark-operator \
     --create-namespace
   ```

   Standardmäßig erstellt der Befehl ein Servicekonto `emr-containers-sa-spark-operator` für den Spark-Operator. Um ein anderes Servicekonto zu verwenden, geben Sie das Argument `serviceAccounts.sparkoperator.name` an. Beispielsweise:

   ```
   --set serviceAccounts.sparkoperator.name my-service-account-for-spark-operator
   ```

   Wenn Sie das vertikale Auto Scaling mit dem Spark-Operator verwenden möchten, fügen Sie dem Installationsbefehl die folgende Zeile hinzu, um Webhooks für den Operator zuzulassen:

   ```
   --set webhook.enable=true
   ```

4. Vergewissern Sie sich, dass Sie das Helm-Chart mit dem folgenden `helm list`-Befehl installiert haben:

```
helm list --namespace spark-operator -o yaml
```

Der `helm list`-Befehl sollte Ihre neu bereitgestellten Helm-Chart-Versionsinformationen zurückgeben:

```
app_version: v1beta2-1.3.8-3.1.1
chart: spark-operator-7.1.0
name: spark-operator-demo
namespace: spark-operator
revision: "1"
status: deployed
updated: 2023-03-14 18:20:02.721638196 +0000 UTC
```

5. Schließen Sie die Installation mit allen zusätzlichen Optionen ab, die Sie benötigen. Weitere Informationen finden Sie in der `spark-on-k8s-operator`Dokumentation unter. GitHub

Eine Spark-Anwendung ausführen

Der Spark-Operator wird mit Amazon EMR 6.10.0 oder höher unterstützt. Wenn Sie den Spark-Operator installieren, erstellt er standardmäßig das Servicekonto `emr-containers-sa-spark` für die Ausführung von Spark-Anwendungen. Gehen Sie wie folgt vor, um eine Spark-Anwendung mit dem Spark-Operator auf Amazon EMR in EKS 6.10.0 oder höher auszuführen.

1. Bevor Sie eine Spark-Anwendung mit dem Spark-Operator ausführen können, führen Sie die Schritte unter Einrichten des Spark-Operators für Amazon EMR in EKS und Den Spark-Operator installieren durch.
2. Erstellen Sie eine `SparkApplication`-Aufgabendefinitions-Datei `spark-pi.yaml` mit dem folgenden Beispielinhalt:

```
apiVersion: "sparkoperator.k8s.io/v1beta2"
kind: SparkApplication
metadata:
  name: spark-pi
  namespace: spark-operator
spec:
  type: Scala
```

```
  mode: cluster
  image: "895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.10.0:latest"
  imagePullPolicy: Always
  mainClass: org.apache.spark.examples.SparkPi
  mainApplicationFile: "local:///usr/lib/spark/examples/jars/spark-examples.jar"
  sparkVersion: "3.3.1"
  restartPolicy:
    type: Never
  volumes:
    - name: "test-volume"
      hostPath:
        path: "/tmp"
        type: Directory
  driver:
    cores: 1
    coreLimit: "1200m"
    memory: "512m"
    labels:
      version: 3.3.1
    serviceAccount: emr-containers-sa-spark
    volumeMounts:
      - name: "test-volume"
        mountPath: "/tmp"
  executor:
    cores: 1
    instances: 1
    memory: "512m"
    labels:
      version: 3.3.1
    volumeMounts:
      - name: "test-volume"
        mountPath: "/tmp"
```

3. Senden Sie jetzt die Spark-Anwendung mit dem folgenden Befehl. Dadurch wird auch ein `SparkApplication`-Objekt mit dem `spark-pi`-Namen erstellt:

```
kubectl apply -f spark-pi.yaml
```

4. Überprüfen Sie die Ereignisse für das `SparkApplication`-Objekt mit dem folgenden Befehl:

```
kubectl describe sparkapplication spark-pi --namespace spark-operator
```

Weitere Informationen zum Einreichen von Anwendungen an Spark über den Spark-Operator finden Sie unter Verwenden von a `SparkApplication` in der `spark-on-k8s-operator` Dokumentation zu GitHub.

Verwenden des vertikalen Auto Scalings mit dem Spark-Operator für Amazon EMR in EKS

Ab Amazon EMR 7.0 können Sie Amazon EMR auf der vertikalen automatischen Skalierung von EKS verwenden, um das Ressourcenmanagement zu vereinfachen. Die Speicher- und CPU-Ressourcen werden automatisch an die Anforderungen der Workload angepasst, die Sie für Amazon-EMR-Spark-Anwendungen bereitstellen. Weitere Informationen finden Sie unter Verwenden von vertikalem Auto Scaling mit Amazon-EMR-Spark-Aufträgen.

In diesem Abschnitt wird beschrieben, wie Sie den Spark-Operator für die Verwendung des vertikalen Auto Scalings konfigurieren.

Voraussetzungen

Bevor Sie fortfahren, müssen Sie die folgenden Einstellungen ausführen:

- Führen Sie die Schritte unter Einrichten des Spark-Operators für Amazon EMR in EKS aus.
- (Optional) Wenn Sie zuvor eine ältere Version des Spark-Operators installiert haben, löschen Sie die SparkApplication CRD/. ScheduledSparkApplication

```
kubectl delete crd sparkApplication
kubectl delete crd scheduledSparkApplication
```

- Führen Sie die Schritte unter Den Spark-Operator installieren aus. Fügen Sie in Schritt 3 dem Installationsbefehl die folgende Zeile hinzu, um Webhooks für den Operator zuzulassen:

```
--set webhook.enable=true
```

- Führen Sie die Schritte unter Einrichten des vertikalen Auto Scalings für Amazon EMR in EKS aus.
- Gewähren Sie Zugriff auf die Dateien an Ihrem Amazon S3 S3-Standort:

  1. Kommentieren Sie Ihr Fahrer- und Betreiber-Servicekonto mit dem`JobExecutionRole`, das über S3-Berechtigungen verfügt.

```
kubectl annotate serviceaccount -n spark-operator emr-containers-sa-spark
 eks.amazonaws.com/role-arn=JobExecutionRole
```

```
kubectl annotate serviceaccount -n spark-operator emr-containers-sa-spark-
operator eks.amazonaws.com/role-arn=JobExecutionRole
```

2. Aktualisieren Sie die Vertrauensrichtlinie Ihrer Jobausführungsrolle in diesem Namespace.

```
aws emr-containers update-role-trust-policy \
--cluster-name cluster \
--namespace ${Namespace}\
--role-name iam_role_name_for_job_execution
```

3. Bearbeiten Sie die IAM-Rollen-Vertrauensrichtlinie Ihrer Jobausführungsrolle und aktualisieren Sie das Feld `serviceaccount` von bis`emr-containers-sa-spark-*-*-xxxx`. `emr-containers-sa-*`

```
{
    "Effect": "Allow",
    "Principal": {
        "Federated": "OIDC-provider"
    },
    "Action": "sts:AssumeRoleWithWebIdentity",
    "Condition": {
        "StringLike": {
            "OIDC": "system:serviceaccount:${Namespace}:emr-containers-sa-*"
        }
    }
}
```

4. Wenn Sie Amazon S3 als Dateispeicher verwenden, fügen Sie Ihrer Yaml-Datei die folgenden Standardeinstellungen hinzu.

```
hadoopConf:
# EMRFS filesystem
  fs.s3.customAWSCredentialsProvider:
 com.amazonaws.auth.WebIdentityTokenCredentialsProvider
  fs.s3.impl: com.amazon.ws.emr.hadoop.fs.EmrFileSystem
  fs.AbstractFileSystem.s3.impl: org.apache.hadoop.fs.s3.EMRFSDelegate
  fs.s3.buffer.dir: /mnt/s3
  fs.s3.getObject.initialSocketTimeoutMilliseconds: "2000"

 mapreduce.fileoutputcommitter.algorithm.version.emr_internal_use_only.EmrFileSystem:
 "2"
  mapreduce.fileoutputcommitter.cleanup-
failures.ignored.emr_internal_use_only.EmrFileSystem: "true"
```

```
sparkConf:
 # Required for EMR Runtime
 spark.driver.extraClassPath: /usr/lib/hadoop-lzo/lib/*:/usr/lib/hadoop/hadoop-
aws.jar:/usr/share/aws/aws-java-sdk/*:/usr/share/aws/emr/emrfs/conf:/usr/share/
aws/emr/emrfs/lib/*:/usr/share/aws/emr/emrfs/auxlib/*:/usr/share/aws/emr/
security/conf:/usr/share/aws/emr/security/lib/*:/usr/share/aws/hmclient/lib/aws-
glue-datacatalog-spark-client.jar:/usr/share/java/Hive-JSON-Serde/hive-openx-
serde.jar:/usr/share/aws/sagemaker-spark-sdk/lib/sagemaker-spark-sdk.jar:/home/
hadoop/extrajars/*
 spark.driver.extraLibraryPath: /usr/lib/hadoop/lib/native:/usr/lib/hadoop-lzo/
lib/native:/docker/usr/lib/hadoop/lib/native:/docker/usr/lib/hadoop-lzo/lib/
native
 spark.executor.extraClassPath: /usr/lib/hadoop-lzo/lib/*:/usr/lib/hadoop/
hadoop-aws.jar:/usr/share/aws/aws-java-sdk/*:/usr/share/aws/emr/emrfs/conf:/usr/
share/aws/emr/emrfs/lib/*:/usr/share/aws/emr/emrfs/auxlib/*:/usr/share/aws/emr/
security/conf:/usr/share/aws/emr/security/lib/*:/usr/share/aws/hmclient/lib/aws-
glue-datacatalog-spark-client.jar:/usr/share/java/Hive-JSON-Serde/hive-openx-
serde.jar:/usr/share/aws/sagemaker-spark-sdk/lib/sagemaker-spark-sdk.jar:/home/
hadoop/extrajars/*
 spark.executor.extraLibraryPath: /usr/lib/hadoop/lib/native:/usr/lib/hadoop-
lzo/lib/native:/docker/usr/lib/hadoop/lib/native:/docker/usr/lib/hadoop-lzo/lib/
native
```

Einen Auftrag mit vertikalem Auto Scaling auf dem Spark-Operator ausführen

Bevor Sie eine Spark-Anwendung mit dem Spark-Operator ausführen können, führen Sie die Schritte unter Voraussetzungen durch.

Um vertikales Autoscaling mit dem Spark-Operator zu verwenden, fügen Sie dem Treiber für Ihre Spark-Anwendungsspezifikation die folgende Konfiguration hinzu, um das vertikale Autoscaling zu aktivieren:

```
dynamicSizing:
  mode: Off
  signature: "my-signature"
```

Diese Konfiguration ermöglicht vertikales Autoscaling und ist eine erforderliche Signaturkonfiguration, mit der Sie eine Signatur für Ihren Job auswählen können.

Weitere Informationen zu den Konfigurationen und Parameterwerten finden Sie unter Konfiguration der vertikalen Autoskalierung für Amazon EMR auf EKS. Standardmäßig wird Ihr Auftrag

beim vertikalen Auto Scaling im Modus Nur für Überwachung aus gesendet. In diesem Überwachungsstatus können Sie Ressourcenempfehlungen berechnen und anzeigen, ohne ein Auto Scaling durchführen zu müssen. Weitere Informationen finden Sie unter Vertikale Autoscaling-Modi.

Im Folgenden finden Sie eine `SparkApplication` Beispieldefinitionsdatei `spark-pi.yaml` mit dem Namen der erforderlichen Konfigurationen für die Verwendung der vertikalen Autoskalierung.

```
apiVersion: "sparkoperator.k8s.io/v1beta2"
kind: SparkApplication
metadata:
  name: spark-pi
  namespace: spark-operator
spec:
  type: Scala
  mode: cluster
  image: "895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-7.1.0:latest"
  imagePullPolicy: Always
  mainClass: org.apache.spark.examples.SparkPi
  mainApplicationFile: "local:///usr/lib/spark/examples/jars/spark-examples.jar"
  sparkVersion: "3.4.1"
  dynamicSizing:
    mode: Off
    signature: "my-signature"
  restartPolicy:
    type: Never
  volumes:
    - name: "test-volume"
      hostPath:
        path: "/tmp"
        type: Directory
  driver:
    cores: 1
    coreLimit: "1200m"
    memory: "512m"
    labels:
      version: 3.4.1
    serviceAccount: emr-containers-sa-spark
    volumeMounts:
      - name: "test-volume"
        mountPath: "/tmp"
  executor:
    cores: 1
    instances: 1
```

```
    memory: "512m"
    labels:
      version: 3.4.1
    volumeMounts:
      - name: "test-volume"
        mountPath: "/tmp"
```

Senden Sie jetzt die Spark-Anwendung mit dem folgenden Befehl. Dadurch wird auch ein `SparkApplication`-Objekt mit dem `spark-pi`-Namen erstellt:

```
kubectl apply -f spark-pi.yaml
```

Weitere Informationen zum Einreichen von Anwendungen an Spark über den Spark-Operator finden Sie unter Using a SparkApplication in der `spark-on-k8s-operator` Dokumentation zu GitHub.

Überprüfen der vertikalen Auto-Scaling-Funktionalität

Um zu überprüfen, ob das vertikale Auto Scaling für den eingereichten Auftrag korrekt funktioniert, rufen Sie mit kubectl die benutzerdefinierte `verticalpodautoscaler`-Ressource ab und sehen Sie sich Ihre Skalierungsempfehlungen an.

```
kubectl get verticalpodautoscalers --all-namespaces \
-l=emr-containers.amazonaws.com/dynamic.sizing.signature=my-signature
```

Die Ausgabe dieser Abfrage sollte wie folgt aussehen:

```
NAMESPACE        NAME                                                    MODE
 CPU   MEM         PROVIDED   AGE
spark-operator   ds-p73j6mkosvc4xeb3gr7x4xol2bfcw5evqimzqojrlysvj3giozuq-vpa   Off
     580026651   True       15m
```

Wenn Ihre Ausgabe nicht ähnlich aussieht oder einen Fehlercode enthält, finden Sie weitere Schritte zur Behebung des Problems in Fehlerbehebung von Amazon EMR im vertikalen Auto Scaling von EKS.

Führen Sie den folgenden Befehl aus, um die Pods und Anwendungen zu entfernen:

```
kubectl delete sparkapplication spark-pi
```

Deinstallieren des Spark-Operators für Amazon EMR in EKS

Gehen Sie wie folgt vor, um den Spark-Operator zu deinstallieren.

1. Löschen Sie den Spark-Operator mit dem richtigen Namespace. In diesem Beispiel lautet der Namespace `spark-operator-demo`.

```
helm uninstall spark-operator-demo -n spark-operator
```

2. Das Spark-Operator-Servicekonto löschen:

```
kubectl delete sa emr-containers-sa-spark-operator -n spark-operator
```

3. Den Spark-Operator `CustomResourceDefinitions` (CRDs) löschen:

```
kubectl delete crd sparkapplications.sparkoperator.k8s.io
kubectl delete crd scheduledsparkapplications.sparkoperator.k8s.io
```

Sicherheit und der Spark-Operator mit Amazon EMR in EKS

Themen

- Einrichten der Cluster-Zugriffskontrolle mit rollenbasierter Zugriffskontrolle (RBAC)
- Einrichten von Cluster-Zugriffsberechtigungen mit IAM-Rollen für Servicekonten (IRSA)

Einrichten der Cluster-Zugriffskontrolle mit rollenbasierter Zugriffskontrolle (RBAC)

Um den Spark-Operator bereitzustellen, erstellt Amazon EMR in EKS zwei Rollen und Servicekonten für den Spark-Operator und die Spark-Anwendungen.

Themen

- Servicekonto und Rolle des Operators
- Spark-Servicekonto und Rolle

Servicekonto und Rolle des Operators

Amazon EMR in EKS erstellt das Operator-Servicekonto und die Rolle, um `SparkApplications` für Spark-Aufträge und andere Ressourcen wie Services zu verwalten.

Der Standardname für dieses Servicekonto lautet `emr-containers-sa-spark-operator`.

Es gelten die folgenden Regeln für diese Servicerolle:

```
 rules:
- apiGroups:
  - ""
  resources:
  - pods
  verbs:
  - "*"
- apiGroups:
  - ""
  resources:
  - services
  - configmaps
  - secrets
  verbs:
  - create
  - get
  - delete
  - update
- apiGroups:
  - extensions
  - networking.k8s.io
  resources:
  - ingresses
  verbs:
  - create
  - get
  - delete
- apiGroups:
  - ""
  resources:
  - nodes
  verbs:
  - get
- apiGroups:
  - ""
  resources:
  - events
  verbs:
  - create
  - update
```

```
  - patch
- apiGroups:
  - ""
  resources:
  - resourcequotas
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - apiextensions.k8s.io
  resources:
  - customresourcedefinitions
  verbs:
  - create
  - get
  - update
  - delete
- apiGroups:
  - admissionregistration.k8s.io
  resources:
  - mutatingwebhookconfigurations
  - validatingwebhookconfigurations
  verbs:
  - create
  - get
  - update
  - delete
- apiGroups:
  - sparkoperator.k8s.io
  resources:
  - sparkapplications
  - sparkapplications/status
  - scheduledsparkapplications
  - scheduledsparkapplications/status
  verbs:
  - "*"
  {{- if .Values.batchScheduler.enable }}
  # required for the `volcano` batch scheduler
- apiGroups:
  - scheduling.incubator.k8s.io
  - scheduling.sigs.dev
  - scheduling.volcano.sh
  resources:
```

```
  - podgroups
  verbs:
  - "*"
  {{- end }}
  {{ if .Values.webhook.enable }}
- apiGroups:
  - batch
  resources:
  - jobs
  verbs:
  - delete
  {{- end }}
```

Spark-Servicekonto und Rolle

Ein Spark-Treiber-Pod benötigt ein Kubernetes-Servicekonto im selben Namespace wie der Pod. Dieses Servicekonto benötigt Berechtigungen zum Erstellen, Abrufen, Auflisten, Patchen und Löschen von Ausführer-Pods sowie zum Erstellen eines Kubernetes-Headless-Services für den Treiber. Der Treiber schlägt fehl und wird ohne das Servicekonto beendet, sofern das Standardservicekonto im Namespace des Pods nicht über die erforderlichen Berechtigungen verfügt.

Der Standardname für dieses Servicekonto lautet e`mr-containers-sa-spark`.

Es gelten die folgenden Regeln für diese Servicerolle:

```
 rules:
- apiGroups:
  - ""
  resources:
  - pods
  verbs:
  - "*"
- apiGroups:
  - ""
  resources:
  - services
  verbs:
  - "*"
- apiGroups:
  - ""
  resources:
  - configmaps
  verbs:
```

```
  - "*"
- apiGroups:
  - ""
  resources:
  - persistentvolumeclaims
  verbs:
  - "*"
```

Einrichten von Cluster-Zugriffsberechtigungen mit IAM-Rollen für Servicekonten (IRSA)

In diesem Abschnitt wird anhand eines Beispiels veranschaulicht, wie ein Kubernetes-Dienstkonto so konfiguriert wird, dass es eine AWS Identity and Access Management Rolle übernimmt. Pods, die das Dienstkonto verwenden, können dann auf jeden AWS Dienst zugreifen, für den die Rolle Zugriffsberechtigungen besitzt.

Im folgenden Beispiel wird eine Spark-Anwendung ausgeführt, um die Wörter aus einer Datei in Amazon S3 zu zählen. Zu diesem Zweck können Sie IAM-Rollen für Servicekonten (IRSA) einrichten, um Kubernetes-Servicekonten zu authentifizieren und zu autorisieren.

> **Note**
>
> In diesem Beispiel wird der Namespace „spark-operator“ für den Spark-Operator und für den Namespace verwendet, in den Sie die Spark-Anwendung einreichen.

Voraussetzungen

Führen Sie die folgenden Voraussetzungen aus, bevor Sie das Beispiel auf dieser Seite ausführen:

- Richten Sie den Spark-Operator ein.
- Den Spark-Operator installieren.
- Erstellen Sie einen Amazon-S3-Bucket.
- Speichern Sie Ihr Lieblingsgedicht in einer Textdatei mit dem Namen `poem.txt` und laden Sie die Datei in Ihren S3-Bucket hoch. Die Spark-Anwendung, die Sie auf dieser Seite erstellen, liest den Inhalt der Textdatei. Weiter Anleitungen zum Hinzufügen von Dateien in S3 finden Sie unter Upload eines Objekts zu Ihrem Bucket im Benutzerhandbuch zu Amazon Simple Storage Service.

Ein Kubernetes-Servicekonto zur Übernahme einer IAM-Rolle konfigurieren

Gehen Sie wie folgt vor, um ein Kubernetes-Dienstkonto so zu konfigurieren, dass es eine IAM-Rolle annimmt, mit der Pods auf AWS Dienste zugreifen können, für die die Rolle über Zugriffsberechtigungen verfügt.

1. Verwenden Sie nach Abschluss des dieVoraussetzungen, AWS Command Line Interface um eine `example-policy.json` Datei zu erstellen, die nur Lesezugriff auf die Datei ermöglicht, die Sie auf Amazon S3 hochgeladen haben:

```
cat >example-policy.json <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::my-pod-bucket",
                "arn:aws:s3:::my-pod-bucket/*"
            ]
        }
    ]
}
EOF
```

2. Dann erstellen Sie eine IAM-Richtlinie `example-policy`:

```
aws iam create-policy --policy-name example-policy --policy-document file://
example-policy.json
```

3. Erstellen Sie als Nächstes eine IAM-Rolle `example-role` und verknüpfen Sie sie mit einem Kubernetes-Servicekonto für den Spark-Treiber:

```
eksctl create iamserviceaccount --name driver-account-sa --namespace spark-operator
 \
--cluster my-cluster --role-name "example-role" \
--attach-policy-arn arn:aws:iam::111122223333:policy/example-policy --approve
```

4. Erstellen Sie eine Yaml-Datei mit den Cluster-Rollenbindungen, die für das Spark-Treiberservicekonto erforderlich sind:

```
cat >spark-rbac.yaml <<EOF
apiVersion: v1
kind: ServiceAccount
metadata:
  name: driver-account-sa
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: spark-role
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: edit
subjects:
  - kind: ServiceAccount
    name: driver-account-sa
    namespace: spark-operator
EOF
```

5. Die Konfigurationen für die Cluster-Rollenbindung anwenden:

```
kubectl apply -f spark-rbac.yaml
```

Der Befehl kubectl sollte die erfolgreiche Erstellung des Kontos bestätigen:

```
serviceaccount/driver-account-sa created
clusterrolebinding.rbac.authorization.k8s.io/spark-role configured
```

Ausführen einer Anwendung über den Spark-Operator

Nachdem Sie das Kubernetes-Servicekonto konfiguriert haben, können Sie eine Spark-Anwendung ausführen, die die Anzahl der Wörter in der Textdatei zählt, die Sie als Teil von Voraussetzungen hochgeladen haben.

1. Erstellen Sie eine neue Datei `word-count.yaml` mit einer `SparkApplication`-Definition für Ihre Anwendung zur Wortzählung.

```
cat >word-count.yaml <<EOF
apiVersion: "sparkoperator.k8s.io/v1beta2"
kind: SparkApplication
metadata:
  name: word-count
  namespace: spark-operator
spec:
  type: Java
  mode: cluster
  image: "895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.10.0:latest"
  imagePullPolicy: Always
  mainClass: org.apache.spark.examples.JavaWordCount
  mainApplicationFile: local:///usr/lib/spark/examples/jars/spark-examples.jar
  arguments:
    - s3://my-pod-bucket/poem.txt
  hadoopConf:
   # EMRFS filesystem
    fs.s3.customAWSCredentialsProvider:
 com.amazonaws.auth.WebIdentityTokenCredentialsProvider
    fs.s3.impl: com.amazon.ws.emr.hadoop.fs.EmrFileSystem
    fs.AbstractFileSystem.s3.impl: org.apache.hadoop.fs.s3.EMRFSDelegate
    fs.s3.buffer.dir: /mnt/s3
    fs.s3.getObject.initialSocketTimeoutMilliseconds: "2000"

 mapreduce.fileoutputcommitter.algorithm.version.emr_internal_use_only.EmrFileSystem:
 "2"
    mapreduce.fileoutputcommitter.cleanup-
failures.ignored.emr_internal_use_only.EmrFileSystem: "true"
  sparkConf:
    # Required for EMR Runtime
    spark.driver.extraClassPath: /usr/lib/hadoop-lzo/lib/*:/usr/lib/hadoop/
hadoop-aws.jar:/usr/share/aws/aws-java-sdk/*:/usr/share/aws/emr/emrfs/conf:/usr/
share/aws/emr/emrfs/lib/*:/usr/share/aws/emr/emrfs/auxlib/*:/usr/share/aws/emr/
security/conf:/usr/share/aws/emr/security/lib/*:/usr/share/aws/hmclient/lib/aws-
glue-datacatalog-spark-client.jar:/usr/share/java/Hive-JSON-Serde/hive-openx-
serde.jar:/usr/share/aws/sagemaker-spark-sdk/lib/sagemaker-spark-sdk.jar:/home/
hadoop/extrajars/*
    spark.driver.extraLibraryPath: /usr/lib/hadoop/lib/native:/usr/lib/hadoop-lzo/
lib/native:/docker/usr/lib/hadoop/lib/native:/docker/usr/lib/hadoop-lzo/lib/native
    spark.executor.extraClassPath: /usr/lib/hadoop-lzo/lib/*:/usr/lib/hadoop/
hadoop-aws.jar:/usr/share/aws/aws-java-sdk/*:/usr/share/aws/emr/emrfs/conf:/usr/
share/aws/emr/emrfs/lib/*:/usr/share/aws/emr/emrfs/auxlib/*:/usr/share/aws/emr/
security/conf:/usr/share/aws/emr/security/lib/*:/usr/share/aws/hmclient/lib/aws-
```

```
glue-datacatalog-spark-client.jar:/usr/share/java/Hive-JSON-Serde/hive-openx-
serde.jar:/usr/share/aws/sagemaker-spark-sdk/lib/sagemaker-spark-sdk.jar:/home/
hadoop/extrajars/*
    spark.executor.extraLibraryPath: /usr/lib/hadoop/lib/native:/usr/lib/hadoop-
lzo/lib/native:/docker/usr/lib/hadoop/lib/native:/docker/usr/lib/hadoop-lzo/lib/
native
  sparkVersion: "3.3.1"
  restartPolicy:
    type: Never
  driver:
    cores: 1
    coreLimit: "1200m"
    memory: "512m"
    labels:
      version: 3.3.1
    serviceAccount: my-spark-driver-sa
  executor:
    cores: 1
    instances: 1
    memory: "512m"
    labels:
      version: 3.3.1
EOF
```

2. Reichen Sie die Spark-Anwendung ein.

```
kubectl apply -f word-count.yaml
```

Der Befehl kubectl sollte eine Bestätigung zurückgeben, dass Sie erfolgreich ein `SparkApplication`-Objekt namens `word-count` erstellt haben.

```
sparkapplication.sparkoperator.k8s.io/word-count configured
```

3. Überprüfen Sie die Ereignisse für das `SparkApplication`-Objekt mit dem folgenden Befehl:

```
kubectl describe sparkapplication word-count -n spark-operator
```

Der Befehl kubectl sollte die Beschreibung des `SparkApplication` mit den Ereignissen zurückgeben:

```
Events:
```

```
  Type     Reason                                   Age                    From
    Message
  ----     ------                                   ----                   ----
    -------
  Normal   SparkApplicationSpecUpdateProcessed  3m2s (x2 over 17h)     spark-
operator  Successfully processed spec update for SparkApplication word-count
  Warning  SparkApplicationPendingRerun         3m2s (x2 over 17h)     spark-
operator  SparkApplication word-count is pending rerun
  Normal   SparkApplicationSubmitted            2m58s (x2 over 17h)    spark-
operator  SparkApplication word-count was submitted successfully
  Normal   SparkDriverRunning                   2m56s (x2 over 17h)    spark-
operator  Driver word-count-driver is running
  Normal   SparkExecutorPending                 2m50s                  spark-
operator  Executor [javawordcount-fdd1698807392c66-exec-1] is pending
  Normal   SparkExecutorRunning                 2m48s                  spark-
operator  Executor [javawordcount-fdd1698807392c66-exec-1] is running
  Normal   SparkDriverCompleted                 2m31s (x2 over 17h)    spark-
operator  Driver word-count-driver completed
  Normal   SparkApplicationCompleted            2m31s (x2 over 17h)    spark-
operator  SparkApplication word-count completed
  Normal   SparkExecutorCompleted               2m31s (x2 over 2m31s)  spark-
operator  Executor [javawordcount-fdd1698807392c66-exec-1] completed
```

Die Anwendung zählt jetzt die Wörter in Ihrer S3-Datei. Die Anzahl der Wörter finden Sie in den Protokolldateien Ihres Fahrers:

```
kubectl logs pod/word-count-driver -n spark-operator
```

Der Befehl kubectl sollte den Inhalt der Protokolldatei mit den Ergebnissen Ihrer Anwendung zur Wortzählung zurückgeben.

```
INFO DAGScheduler: Job 0 finished: collect at JavaWordCount.java:53, took 5.146519 s
               Software: 1
```

Weitere Informationen zum Einreichen von Bewerbungen über den Spark-Operator bei Spark finden Sie unter Using a SparkApplication in der Dokumentation zum Kubernetes-Operator für Apache Spark (spark-on-k8s-Operator) unter. GitHub

Ausführen von Spark-Aufträgen mit spark-submit

Amazon-EMR-Versionen 6.10.0 und höher unterstützen `spark-submit` als Befehlszeilentool, mit dem Sie Spark-Anwendungen an einen Amazon EMR in EKS-Cluster senden und ausführen können.

> **Note**
>
> Amazon EMR berechnet die Preise für Amazon EKS auf der Grundlage des vCPU- und Speicherverbrauchs. Diese Berechnung gilt für Treiber- und Executor-Pods. Diese Berechnung beginnt mit dem Herunterladen Ihres Amazon EMR-Anwendungsabbilds, bis der Amazon EKS-Pod endet und auf die nächste Sekunde gerundet wird.

Themen

Einrichten von spark-submit für Amazon EMR in EKS

Führen Sie die folgenden Aufgaben aus, um eine Anwendung mit spark-submit in Amazon EMR in EKS ausführen zu können. Wenn Sie bereits für Amazon Web Services (AWS) registriert sind und Amazon EKS schon verwendet haben, müssen Sie nur wenige Schritte ausführen, um Amazon EMR in EKS nutzen zu können. Wenn Sie bereits eine der Voraussetzungen erfüllt haben, können Sie diese überspringen und mit der nächsten fortfahren.

- Installieren Sie das AWS CLI – Wenn Sie AWS CLI bereits installiert haben, vergewissern Sie sich, dass Sie über die neueste Version verfügen.
- eksctl installieren – eksctl ist ein Befehlszeilentool, das Sie für die Kommunikation mit Amazon EKS verwenden.
- Einen Amazon-EKS-Cluster einrichten – Folgen Sie den Schritten, um einen neuen Kubernetes-Cluster mit Knoten in Amazon EKS zu erstellen.
- Eine EMR-Basis-Image-URI von Amazon auswählen (Version 6.10.0 oder höher) aus – der `spark-submit`-Befehl wird von Amazon-EMR-Versionen 6.10.0 und höher unterstützt.

- Bestätigen Sie, dass das Treiberservicekonto über die entsprechenden Berechtigungen zum Erstellen und Überwachen von Executor-Pods verfügt. Weitere Informationen finden Sie unter Sicherheitsanforderungen für Spark-Submit für das Spark-Servicekonto.
- Richten Sie Ihr lokales AWS -Anmeldeinformationsprofil ein.
- Wählen Sie in der Amazon EKS-Konsole Ihren EKS-Cluster aus und suchen Sie dann den EKS-Cluster-Endpunkt unter Übersicht, Details und dann API-Serverendpunkt.

Erste Schritte mit spark-submit für Amazon EMR in EKS

Eine Spark-Anwendung ausführen

Amazon EMR 6.10.0 und höher unterstützt Spark-Submit für die Ausführung von Spark-Anwendungen auf einem Amazon-EKS-Cluster. Führen Sie die folgenden Schritte aus, um die Spark-Anwendung auszuführen:

1. Bevor Sie eine Spark-Anwendung mit dem `spark-submit`-Befehl ausführen können, führen Sie die Schritte unter Einrichten von spark-submit für Amazon EMR in EKS durch.
2. Führen Sie einen Container mit einem Amazon EMR auf dem EKS-Basis-Image aus. Weitere Informationen finden Sie unter So wählen Sie einen Basis-Image-URI aus.

   ```
   kubectl run -it containerName --image=EMROnEKSImage --command -n namespace /bin/
   bash
   ```

3. Legen Sie die Werte der folgenden Umgebungsvariablen fest:

   ```
   export SPARK_HOME=spark-home
   export MASTER_URL=k8s://Amazon EKS-cluster-endpoint
   ```

4. Senden Sie jetzt die Spark-Anwendung mit dem folgenden Befehl:

   ```
   $SPARK_HOME/bin/spark-submit \
    --class org.apache.spark.examples.SparkPi \
    --master $MASTER_URL \
    --conf spark.kubernetes.container.image=895885662937.dkr.ecr.us-
   west-2.amazonaws.com/spark/emr-6.10.0:latest \
    --conf spark.kubernetes.authenticate.driver.serviceAccountName=spark \
    --deploy-mode cluster \
    --conf spark.kubernetes.namespace=spark-operator \
    local:///usr/lib/spark/examples/jars/spark-examples.jar 20
   ```

Weitere Informationen zum Senden von Anwendungen an Spark finden Sie im Thema Anwendungen senden in der Apache-Spark-Dokumentation.

> **Important**
>
> `spark-submit` unterstützt nur den Cluster-Modus als Einreichungsmechanismus.

Sicherheitsanforderungen für Spark-Submit für das Spark-Servicekonto

Der Spark-Treiber-Pod verwendet ein Kubernetes-Servicekonto, um auf den Kubernetes-API-Server zuzugreifen und Executor-Pods zu erstellen und zu überwachen. Das Treiber-Servicekonto muss über die entsprechenden Berechtigungen verfügen, um Pods in Ihrem Cluster aufzulisten, zu erstellen, zu bearbeiten, zu patchen und zu löschen. Sie können überprüfen, ob Sie diese Ressourcen auflisten können, indem Sie den folgenden Befehl ausführen:

```
kubectl auth can-i list|create|edit|delete|patch pods
```

Stellen Sie sicher, dass Sie über die erforderlichen Berechtigungen verfügen, indem Sie jeden Befehl ausführen.

```
kubectl auth can-i list pods
kubectl auth can-i create pods
kubectl auth can-i edit pods
kubectl auth can-i delete pods
kubectl auth can-i patch pods
```

Es gelten die folgenden Regeln für diese Servicerolle:

```
 rules:
- apiGroups:
  - ""
  resources:
  - pods
  verbs:
  - "*"
- apiGroups:
  - ""
  resources:
  - services
  verbs:
```

```
  - "*"
- apiGroups:
  - ""
  resources:
  - configmaps
  verbs:
  - "*"
- apiGroups:
  - ""
  resources:
  - persistentvolumeclaims
  verbs:
  - "*"
```

Einrichtung von IAM-Rollen für Dienstkonten (IRSA) für Spark-Submit

In den folgenden Abschnitten wird erklärt, wie Sie IAM-Rollen für Service Accounts (IRSA) einrichten, um Kubernetes-Dienstkonten zu authentifizieren und zu autorisieren, sodass Sie in Amazon S3 gespeicherte Spark-Anwendungen ausführen können.

Voraussetzungen

Bevor Sie eines der Beispiele in dieser Dokumentation ausprobieren, stellen Sie sicher, dass Sie die folgenden Voraussetzungen erfüllt haben:

- Die Einrichtung von Spark-Submit ist abgeschlossen
- Es wurde ein S3-Bucket erstellt und das Spark-Anwendungs-JAR hochgeladen

Konfiguration eines Kubernetes-Dienstkontos für die Übernahme einer IAM-Rolle

In den folgenden Schritten wird beschrieben, wie Sie ein Kubernetes-Dienstkonto so konfigurieren, dass es eine AWS Identity and Access Management (IAM-) Rolle annimmt. Nachdem Sie die Pods für die Verwendung des Dienstkontos konfiguriert haben, können sie auf alle Pods zugreifen, für die AWS-Service die Rolle über Zugriffsberechtigungen verfügt.

1. Erstellen Sie eine Richtliniendatei, um den schreibgeschützten Zugriff auf das Amazon S3 S3-Objekt zu ermöglichen, das Sie hochgeladen haben:

   ```
   cat >my-policy.json <<EOF
   {
       "Version": "2012-10-17",
   ```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::<my-spark-jar-bucket>",
                "arn:aws:s3:::<my-spark-jar-bucket>/*"
            ]
        }
    ]
}
EOF
```

2. Erstellen Sie die IAM-Richtlinie.

```
aws iam create-policy --policy-name my-policy --policy-document file://my-
policy.json
```

3. Erstellen Sie eine IAM-Rolle und verknüpfen Sie sie mit einem Kubernetes-Dienstkonto für den Spark-Treiber

```
eksctl create iamserviceaccount --name my-spark-driver-sa --namespace spark-
operator \
--cluster my-cluster --role-name "my-role" \
--attach-policy-arn arn:aws:iam::111122223333:policy/my-policy --approve
```

4. Erstellen Sie eine YAML-Datei mit den erforderlichen Berechtigungen für das Spark-Treiberdienstkonto:

```
cat >spark-rbac.yaml <<EOF
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: default
  name: emr-containers-role-spark
rules:
- apiGroups:
  - ""
  resources:
  - pods
```

```
  verbs:
  - "*"
- apiGroups:
  - ""
  resources:
  - services
  verbs:
  - "*"
- apiGroups:
  - ""
  resources:
  - configmaps
  verbs:
  - "*"
- apiGroups:
  - ""
  resources:
  - persistentvolumeclaims
  verbs:
  - "*"
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: spark-role-binding
  namespace: default
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: emr-containers-role-spark
subjects:
- kind: ServiceAccount
  name: emr-containers-sa-spark
  namespace: default
EOF
```

5. Wenden Sie die Bindungskonfigurationen für die Clusterrolle an.

```
kubectl apply -f spark-rbac.yaml
```

6. Der `kubectl` Befehl sollte die Bestätigung des erstellten Kontos zurückgeben.

```
serviceaccount/emr-containers-sa-spark created
```

```
clusterrolebinding.rbac.authorization.k8s.io/emr-containers-role-spark configured
```

Die Spark-Anwendung ausführen

Amazon EMR 6.10.0 und höher unterstützt Spark-Submit für die Ausführung von Spark-Anwendungen auf einem Amazon-EKS-Cluster. Führen Sie die folgenden Schritte aus, um die Spark-Anwendung auszuführen:

1. Stellen Sie sicher, dass Sie die Schritte unter Spark-Submit für Amazon EMR auf EKS einrichten abgeschlossen haben.
2. Legen Sie die Werte der folgenden Umgebungsvariablen fest:

```
export SPARK_HOME=spark-home
export MASTER_URL=k8s://Amazon EKS-cluster-endpoint
```

3. Senden Sie jetzt die Spark-Anwendung mit dem folgenden Befehl:

```
$SPARK_HOME/bin/spark-submit \
 --class org.apache.spark.examples.SparkPi \
 --master $MASTER_URL \
 --conf spark.kubernetes.container.image=895885662937.dkr.ecr.us-
west-2.amazonaws.com/spark/emr-6.15.0:latest \
 --conf spark.kubernetes.authenticate.driver.serviceAccountName=emr-containers-sa-
spark \
 --deploy-mode cluster \
 --conf spark.kubernetes.namespace=default \
 --conf "spark.driver.extraClassPath=/usr/lib/hadoop-lzo/lib/*:/usr/lib/hadoop/
hadoop-aws.jar:/usr/share/aws/aws-java-sdk/*:/usr/share/aws/emr/emrfs/conf:/usr/
share/aws/emr/emrfs/lib/*:/usr/share/aws/emr/emrfs/auxlib/*:/usr/share/aws/emr/
security/conf:/usr/share/aws/emr/security/lib/*:/usr/share/aws/hmclient/lib/aws-
glue-datacatalog-spark-client.jar:/usr/share/java/Hive-JSON-Serde/hive-openx-
serde.jar:/usr/share/aws/sagemaker-spark-sdk/lib/sagemaker-spark-sdk.jar:/home/
hadoop/extrajars/*" \
 --conf "spark.driver.extraLibraryPath=/usr/lib/hadoop/lib/native:/usr/lib/hadoop-
lzo/lib/native:/docker/usr/lib/hadoop/lib/native:/docker/usr/lib/hadoop-lzo/lib/
native" \
 --conf "spark.executor.extraClassPath=/usr/lib/hadoop-lzo/lib/*:/usr/lib/hadoop/
hadoop-aws.jar:/usr/share/aws/aws-java-sdk/*:/usr/share/aws/emr/emrfs/conf:/usr/
share/aws/emr/emrfs/lib/*:/usr/share/aws/emr/emrfs/auxlib/*:/usr/share/aws/emr/
security/conf:/usr/share/aws/emr/security/lib/*:/usr/share/aws/hmclient/lib/aws-
glue-datacatalog-spark-client.jar:/usr/share/java/Hive-JSON-Serde/hive-openx-
```

```
serde.jar:/usr/share/aws/sagemaker-spark-sdk/lib/sagemaker-spark-sdk.jar:/home/
hadoop/extrajars/*" \
 --conf "spark.executor.extraLibraryPath=/usr/lib/hadoop/lib/native:/usr/lib/
hadoop-lzo/lib/native:/docker/usr/lib/hadoop/lib/native:/docker/usr/lib/hadoop-lzo/
lib/native" \
 --conf
 spark.hadoop.fs.s3.customAWSCredentialsProvider=com.amazonaws.auth.WebIdentityTokenCredent
 \
 --conf spark.hadoop.fs.s3.impl=com.amazon.ws.emr.hadoop.fs.EmrFileSystem \
 --conf
 spark.hadoop.fs.AbstractFileSystem.s3.impl=org.apache.hadoop.fs.s3.EMRFSDelegate \
 --conf spark.hadoop.fs.s3.buffer.dir=/mnt/s3 \
 --conf spark.hadoop.fs.s3.getObject.initialSocketTimeoutMilliseconds="2000" \
 --conf
 spark.hadoop.mapreduce.fileoutputcommitter.algorithm.version.emr_internal_use_only.EmrFile
 \
 --conf spark.hadoop.mapreduce.fileoutputcommitter.cleanup-
failures.ignored.emr_internal_use_only.EmrFileSystem="true" \
 s3://my-pod-bucket/spark-examples.jar 20
```

4. Nachdem der Spark-Treiber den Spark-Job abgeschlossen hat, sollten Sie am Ende der Einreichung eine Protokollzeile sehen, die angibt, dass der Spark-Job abgeschlossen ist.

```
23/11/24 17:02:14 INFO LoggingPodStatusWatcherImpl: Application
 org.apache.spark.examples.SparkPi with submission ID default:org-apache-spark-
examples-sparkpi-4980808c03ff3115-driver finished
23/11/24 17:02:14 INFO ShutdownHookManager: Shutdown hook called
```

Bereinigen

Wenn Sie mit der Ausführung Ihrer Anwendungen fertig sind, können Sie die Bereinigung mit dem folgenden Befehl durchführen.

```
kubectl delete -f spark-rbac.yaml
```

Verwenden von Apache Livy mit Amazon EMR auf EKS

Mit Amazon EMR-Versionen 7.1.0 und höher können Sie Apache Livy verwenden, um Jobs auf Amazon EMR auf EKS einzureichen. Mit Apache Livy können Sie Ihren eigenen Apache Livy REST-Endpunkt einrichten und ihn verwenden, um Spark-Anwendungen auf Ihren Amazon EKS-Clustern

bereitzustellen und zu verwalten. Nachdem Sie Livy in Ihrem Amazon EKS-Cluster installiert haben, können Sie den Livy-Endpunkt verwenden, um Spark-Anwendungen an Ihren Livy-Server zu senden. Der Server verwaltet den Lebenszyklus der Spark-Anwendungen.

> **Note**
>
> Amazon EMR berechnet die Preise für Amazon EKS auf der Grundlage des vCPU- und Speicherverbrauchs. Diese Berechnung gilt für Treiber- und Executor-Pods. Diese Berechnung beginnt mit dem Herunterladen Ihres Amazon EMR-Anwendungsabbilds, bis der Amazon EKS-Pod endet und auf die nächste Sekunde gerundet wird.

Themen

Apache Livy für Amazon EMR auf EKS einrichten

Bevor Sie Apache Livy auf Ihrem Amazon EKS-Cluster installieren können, müssen Sie die folgenden Aufgaben ausführen.

- Installieren Sie das AWS CLI— Wenn Sie das bereits installiert haben, vergewissern Sie sich AWS CLI, dass Sie über die neueste Version verfügen.
- eksctl installieren – eksctl ist ein Befehlszeilentool, das Sie für die Kommunikation mit Amazon EKS verwenden.
- Helm installieren – Der Helm-Paketmanager für Kubernetes unterstützt Sie bei der Installation und Verwaltung von Anwendungen in Ihrem Kubernetes-Cluster.
- Einen Amazon-EKS-Cluster einrichten – Folgen Sie den Schritten, um einen neuen Kubernetes-Cluster mit Knoten in Amazon EKS zu erstellen.

- Wählen Sie ein Amazon EMR-Release-Label aus — Apache Livy wird von Amazon EMR-Versionen 7.1.0 und höher unterstützt.
- Installieren Sie den ALB-Controller — Der ALB-Controller verwaltet AWS Elastic Load Balancing für Kubernetes-Cluster. Es erstellt einen AWS Network Load Balancer (NLB), wenn Sie bei der Einrichtung von Apache Livy einen Kubernetes-Ingress erstellen.

Erste Schritte mit Apache Livy auf Amazon EMR auf EKS

Gehen Sie wie folgt vor, um Apache Livy zu installieren.

1. Falls Sie es noch nicht getan haben, richten Sie Apache Livy für Amazon EMR auf EKS ein.
2. Authentifizieren Ihren Helm-Client in Ihrer Amazon-ECR-Registry. Sie können den entsprechenden `ECR-registry-account` Wert für Ihre AWS-Region Amazon ECR-Registrierungskonten nach Regionen suchen.

   ```
   aws ecr get-login-password \--region <AWS_REGION> | helm registry login \
   --username AWS \
   --password-stdin <ECR-registry-account>.dkr.ecr.<region-id>.amazonaws.com
   ```

3. Durch die Einrichtung von Livy wird ein Dienstkonto für den Livy-Server und ein weiteres Konto für die Spark-Anwendung erstellt. Informationen zum Einrichten von IRSA für die Dienstkonten finden Sie unter Einrichten von Zugriffsberechtigungen mit IAM-Rollen für Dienstkonten (IRSA).
4. Erstellen Sie einen Namespace, um Ihre Spark-Workloads auszuführen.

   ```
   kubectl create ns <spark-ns>
   ```

5. Verwenden Sie den folgenden Befehl, um Livy zu installieren.

   Dieser Livy-Endpunkt ist nur intern für die VPC im EKS-Cluster verfügbar. Um den Zugriff über die VPC hinaus zu ermöglichen, geben Sie `—-set loadbalancer.internal=false` Ihren Helm-Installationsbefehl ein.

   > **Note**
   >
   > Standardmäßig ist SSL in diesem Livy-Endpunkt nicht aktiviert und der Endpunkt ist nur innerhalb der VPC des EKS-Clusters sichtbar. Wenn Sie `loadbalancer.internal=false` und festlegen`ssl.enabled=false`, setzen Sie einen unsicheren Endpunkt außerhalb Ihrer VPC offen. Informationen zum Einrichten

eines sicheren Livy-Endpunkts finden Sie unter Konfiguration eines sicheren Apache Livy-Endpunkts mit TLS/SSL.

```
helm install livy-demo \
  oci://895885662937.dkr.ecr.region-id.amazonaws.com/livy \
  --version 7.1.0 \
  --namespace livy-ns \
  --set image=ECR-registry-account.dkr.ecr.region-id.amazonaws.com/livy/
emr-7.1.0:latest \
  --set sparkNamespace=<spark-ns> \
  --create-namespace
```

Die Ausgabe sollte folgendermaßen aussehen.

```
NAME: livy-demo
LAST DEPLOYED: Mon Mar 18 09:23:23 2024
NAMESPACE: livy-ns
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
The Livy server has been installed.
Check installation status:
1. Check Livy Server pod is running
  kubectl --namespace livy-ns get pods -l "app.kubernetes.io/instance=livy-demo"
2. Verify created NLB is in Active state and it's target groups are healthy (if
 loadbalancer.enabled is true)

Access LIVY APIs:
    # Ensure your NLB is active and healthy
    # Get the Livy endpoint using command:
    LIVY_ENDPOINT=$(kubectl get svc -n livy-ns -l app.kubernetes.io/
instance=livy-demo,emr-containers.amazonaws.com/type=loadbalancer -o
 jsonpath='{.items[0].status.loadBalancer.ingress[0].hostname}' |  awk '{printf
 "%s:8998\n", $0}')
    # Access Livy APIs using http://$LIVY_ENDPOINT or https://$LIVY_ENDPOINT (if
 SSL is enabled)
    # Note: While uninstalling Livy, makes sure the ingress and NLB are deleted
 after running the helm command to avoid dangling resources
```

Die Standardnamen der Dienstkonten für den Livy-Server und die Spark-Sitzung lauten und. `emr-containers-sa-livy` `emr-containers-sa-spark-livy` Um benutzerdefinierte Namen zu verwenden, verwenden Sie die `sparkServiceAccount.name` Parameter `serviceAccounts.name` und.

```
--set serviceAccounts.name=my-service-account-for-livy
--set sparkServiceAccount.name=my-service-account-for-spark
```

6. Stellen Sie sicher, dass Sie das Helm-Diagramm installiert haben.

```
helm list -n livy-ns -o yaml
```

Der `helm list` Befehl sollte Informationen über Ihr neues Helm-Diagramm zurückgeben.

```
app_version: 0.7.1-incubating
chart: livy-emr-7.1.0
name: livy-demo
namespace: livy-ns
revision: "1"
status: deployed
updated: 2024-02-08 22:39:53.539243 -0800 PST
```

7. Stellen Sie sicher, dass der Network Load Balancer aktiv ist.

```
LIVY_NAMESPACE=<livy-ns>
LIVY_APP_NAME=<livy-app-name>
AWS_REGION=<AWS_REGION>

# Get the NLB Endpoint URL
NLB_ENDPOINT=$(kubectl --namespace $LIVY_NAMESPACE get svc -l "app.kubernetes.io/
instance=$LIVY_APP_NAME,emr-containers.amazonaws.com/type=loadbalancer" -o
 jsonpath='{.items[0].status.loadBalancer.ingress[0].hostname}')

# Get all the load balancers in the account's region
ELB_LIST=$(aws elbv2 describe-load-balancers --region $AWS_REGION)

# Get the status of the NLB that matching the endpoint from the Kubernetes service
NLB_STATUS=$(echo $ELB_LIST | grep -A 8 "\"DNSName\": \"$NLB_ENDPOINT\"" | awk '/
Code/{print $2}/}/' | tr -d '"},\n')
echo $NLB_STATUS
```

8. Stellen Sie nun sicher, dass die Zielgruppe im Network Load Balancer fehlerfrei ist.

```
LIVY_NAMESPACE=<livy-ns>
LIVY_APP_NAME=<livy-app-name>
AWS_REGION=<AWS_REGION>

# Get the NLB endpoint
NLB_ENDPOINT=$(kubectl --namespace $LIVY_NAMESPACE get svc -l "app.kubernetes.io/
instance=$LIVY_APP_NAME,emr-containers.amazonaws.com/type=loadbalancer" -o
 jsonpath='{.items[0].status.loadBalancer.ingress[0].hostname}')

# Get all the load balancers in the account's region
ELB_LIST=$(aws elbv2 describe-load-balancers --region $AWS_REGION)

# Get the NLB ARN from the NLB endpoint
NLB_ARN=$(echo $ELB_LIST | grep -B 1 "\"DNSName\": \"$NLB_ENDPOINT\"" | awk
 '/"LoadBalancerArn":/,/"/'| awk '/:/{print $2}' | tr -d \",)

# Get the target group from the NLB. Livy setup only deploys 1 target group
TARGET_GROUP_ARN=$(aws elbv2 describe-target-groups --load-balancer-arn $NLB_ARN
 --region $AWS_REGION | awk '/"TargetGroupArn":/,/"/'| awk '/:/{print $2}' | tr -d
 \",)

# Get health of target group
aws elbv2 describe-target-health --target-group-arn $TARGET_GROUP_ARN
```

Im Folgenden finden Sie eine Beispielausgabe, die den Status der Zielgruppe zeigt:

```
{
    "TargetHealthDescriptions": [
        {
            "Target": {
                "Id": "<target IP>",
                "Port": 8998,
                "AvailabilityZone": "us-west-2d"
            },
            "HealthCheckPort": "8998",
            "TargetHealth": {
                "State": "healthy"
            }
        }
    ]
```

```
}
```

Sobald der Status Ihrer NLB den Status `active` Ihrer Zielgruppe erreicht hat`healthy`, können Sie weitermachen. Dies kann ein paar Minuten dauern.

9. Rufen Sie den Livy-Endpunkt aus der Helm-Installation ab. Ob Ihr Livy-Endpunkt sicher ist oder nicht, hängt davon ab, ob Sie SSL aktiviert haben.

```
LIVY_NAMESPACE=<livy-ns>
 LIVY_APP_NAME=livy-app-name
 LIVY_ENDPOINT=$(kubectl get svc -n livy-ns -l app.kubernetes.io/
instance=livy-app-name,emr-containers.amazonaws.com/type=loadbalancer -o
 jsonpath='{.items[0].status.loadBalancer.ingress[0].hostname}' |  awk '{printf
 "%s:8998\n", $0}')
 echo "$LIVY_ENDPOINT"
```

10. Rufen Sie das Spark-Dienstkonto aus der Helm-Installation ab

```
SPARK_NAMESPACE=spark-ns
LIVY_APP_NAME=<livy-app-name>
SPARK_SERVICE_ACCOUNT=$(kubectl --namespace $SPARK_NAMESPACE
 get sa -l "app.kubernetes.io/instance=$LIVY_APP_NAME" -o
 jsonpath='{.items[0].metadata.name}')
echo "$SPARK_SERVICE_ACCOUNT"
```

Die Ausgabe sollte der folgenden Ausgabe ähneln:

```
emr-containers-sa-spark-livy
```

11. Wenn Sie `internalALB=true` den Zugriff von außerhalb Ihrer VPC aktivieren möchten, erstellen Sie eine Amazon EC2 EC2-Instance und stellen Sie sicher, dass der Network Load Balancer Netzwerkverkehr von der EC2-Instance zulässt. Sie müssen dies tun, damit die Instance Zugriff auf Ihren Livy-Endpunkt hat. Weitere Informationen zur sicheren Bereitstellung Ihres Endpunkts außerhalb Ihrer VPC finden Sie unter Einrichtung eines sicheren Apache Livy-Endpunkts mit TLS/SSL.
12. Durch die Installation von Livy wird das Dienstkonto für die Ausführung von Spark-Anwendungen erstellt. `emr-containers-sa-spark` Wenn Ihre Spark-Anwendung AWS Ressourcen wie S3 verwendet oder AWS API- oder CLI-Operationen aufruft, müssen Sie eine IAM-Rolle mit den

erforderlichen Berechtigungen mit Ihrem Spark-Dienstkonto verknüpfen. Weitere Informationen finden Sie unter Einrichten von Zugriffsberechtigungen mit IAM-Rollen für Dienstkonten (IRSA).

Apache Livy unterstützt zusätzliche Konfigurationen, die Sie bei der Installation von Livy verwenden können. Weitere Informationen finden Sie unter Installationseigenschaften für Apache Livy auf Amazon EMR auf EKS-Versionen.

Ausführen einer Spark-Anwendung mit Apache Livy für Amazon EMR auf EKS

Bevor Sie eine Spark-Anwendung mit Apache Livy ausführen können, stellen Sie sicher, dass Sie die Schritte unter Apache Livy für Amazon EMR auf EKS einrichten und Erste Schritte mit Apache Livy für Amazon EMR auf EKS ausgeführt haben.

Sie können Apache Livy verwenden, um zwei Arten von Anwendungen auszuführen:

- Batch-Sitzungen — eine Art von Livy-Workload zum Senden von Spark-Batch-Jobs.
- Interaktive Sitzungen — eine Art von Livy-Workload, der eine programmatische und visuelle Oberfläche zur Ausführung von Spark-Abfragen bietet.

> **Note**
>
> Treiber- und Executor-Pods aus verschiedenen Sitzungen können miteinander kommunizieren. Namespaces garantieren keine Sicherheit zwischen Pods. Kubernetes erlaubt keine selektiven Berechtigungen für eine Teilmenge von Pods innerhalb eines bestimmten Namespace.

Batch-Sitzungen ausführen

Verwenden Sie den folgenden Befehl, um einen Batch-Job zu senden.

```
curl -s -k -H 'Content-Type: application/json' -X POST \
      -d '{
            "name": "my-session",
            "file": "entryPoint_location (S3 or local)",
            "args": ["argument1", "argument2", ...],
            "conf": {
```

```
                "spark.kubernetes.namespace": "<spark-namespace>",
                "spark.kubernetes.container.image": "public.ecr.aws/emr-on-eks/spark/
emr-7.1.0:latest",
                "spark.kubernetes.authenticate.driver.serviceAccountName": "<spark-
service-account>"
            }
          }' <livy-endpoint>/batches
```

Verwenden Sie den folgenden Befehl, um Ihren Batch-Job zu überwachen.

```
curl -s -k -H 'Content-Type: application/json' -X GET <livy-endpoint>/batches/my-
session
```

Interaktive Sitzungen ausführen

Gehen Sie wie folgt vor, um interaktive Sitzungen mit Apache Livy auszuführen.

1. Stellen Sie sicher, dass Sie Zugriff auf ein selbst gehostetes oder ein verwaltetes Jupyter-Notebook haben, z. B. ein Jupyter-Notebook. SageMaker Auf Ihrem Jupyter-Notebook muss Sparkmagic installiert sein.
2. Erstellen Sie einen Bucket für die Spark-Konfiguration. `spark.kubernetes.file.upload.path` Stellen Sie sicher, dass das Spark-Dienstkonto Lese- und Schreibzugriff auf den Bucket hat. Weitere Informationen zur Konfiguration Ihres Spark-Dienstkontos finden Sie unter Einrichten von Zugriffsberechtigungen mit IAM-Rollen für Dienstkonten (IRSA)
3. Laden Sie sparkmagic mit dem Befehl in das Jupyter-Notebook. `%load_ext sparkmagic.magics`
4. Führen Sie den Befehl aus`%manage_spark`, um Ihren Livy-Endpunkt mit dem Jupyter-Notebook einzurichten. Wählen Sie die Registerkarte Endpunkte hinzufügen, wählen Sie den konfigurierten Authentifizierungstyp aus, fügen Sie den Livy-Endpunkt zum Notizbuch hinzu und wählen Sie dann Endpunkt hinzufügen.
5. Führen Sie den **`%manage_spark`** Vorgang erneut aus, um den Spark-Kontext zu erstellen, und wechseln Sie dann zur Sitzung erstellen. Wählen Sie den Livy-Endpunkt, geben Sie einen eindeutigen Sitzungsnamen an, wählen Sie eine Sprache und fügen Sie dann die folgenden Eigenschaften hinzu.

   ```
   {
     "conf": {
   ```

```
    "spark.kubernetes.namespace": "livy-namespace",
    "spark.kubernetes.container.image": "public.ecr.aws/emr-on-eks/spark/
emr-7.1.0:latest",
    "spark.kubernetes.authenticate.driver.serviceAccountName": "<spark-service-
account>",
    "spark.kubernetes.file.upload.path": "<URI_TO_S3_LOCATION_>"
  }
}
```

6. Reichen Sie die Anwendung ein und warten Sie, bis sie den Spark-Kontext erstellt hat.
7. Führen Sie den folgenden Befehl aus, um den Status der interaktiven Sitzung zu überwachen.

```
curl -s -k -H 'Content-Type: application/json' -X GET livy-endpoint/sessions/my-
interactive-session
```

Überwachung von Spark-Anwendungen

Verwenden Sie den Link`http://<livy-endpoint>/ui`, um den Fortschritt Ihrer Spark-Anwendungen mit der Livy-Benutzeroberfläche zu überwachen.

Deinstallation von Apache Livy mit Amazon EMR auf EKS

Gehen Sie wie folgt vor, um Apache Livy zu deinstallieren.

1. Löschen Sie das Livy-Setup mit den Namen Ihres Namespaces und des Anwendungsnamens. In diesem Beispiel lautet der Anwendungsname `livy-demo` und der Namespace ist. `livy-ns`

```
helm uninstall livy-demo -n livy-ns
```

2. Bei der Deinstallation löscht Amazon EMR on EKS den Kubernetes-Service in Livy, die AWS Load Balancer und die Zielgruppen, die Sie während der Installation erstellt haben. Das Löschen von Ressourcen kann einige Minuten dauern. Stellen Sie sicher, dass die Ressourcen gelöscht wurden, bevor Sie Livy erneut im Namespace installieren.
3. Löschen Sie den Spark-Namespace.

```
kubectl delete namespace spark-ns
```

Sicherheit für Apache Livy mit Amazon EMR auf EKS

Auf den folgenden Seiten erfahren Sie mehr über die Konfiguration der Sicherheit für Apache Livy mit Amazon EMR auf EKS

Themen

Einrichtung eines sicheren Apache Livy-Endpunkts mit TLS/SSL

In den folgenden Abschnitten erfahren Sie mehr über die Einrichtung von Apache Livy für Amazon EMR auf EKS mit end-to-end TLS- und SSL-Verschlüsselung.

Einrichtung der TLS- und SSL-Verschlüsselung

Gehen Sie folgendermaßen vor, um die SSL-Verschlüsselung auf Ihrem Apache Livy-Endpunkt einzurichten.

- Installieren Sie den Secrets Store CSI-Treiber und den AWS Secrets and Configuration Provider (ASCP) — der Secrets Store CSI-Treiber und ASCP speichern sicher die JKS-Zertifikate und Passwörter von Livy, die der Livy-Server-Pod benötigt, um SSL zu aktivieren. Sie können auch nur den Secrets Store CSI-Treiber installieren und jeden anderen unterstützten Secrets-Anbieter verwenden.
- Erstellen Sie ein ACM-Zertifikat — dieses Zertifikat ist erforderlich, um die Verbindung zwischen dem Client und dem ALB-Endpunkt zu sichern.
- Richten Sie ein JKS-Zertifikat, ein Schlüsselkennwort und ein Keystore-Passwort für ein AWS Secrets Manager — erforderlich, um die Verbindung zwischen dem ALB-Endpunkt und dem Livy-Server zu sichern.
- Fügen Sie dem Livy-Dienstkonto Berechtigungen zum Abrufen von Geheimnissen hinzu AWS Secrets Manager — der Livy-Server benötigt diese Berechtigungen, um Geheimnisse von ASCP abzurufen und die Livy-Konfigurationen hinzuzufügen, um den Livy-Server zu sichern. Informationen zum Hinzufügen von IAM-Berechtigungen zu einem Dienstkonto finden Sie unter Einrichten von Zugriffsberechtigungen mit IAM-Rollen für Dienstkonten (IRSA).

Einrichten eines JKS-Zertifikats mit einem Schlüssel und einem Keystore-Passwort für AWS Secrets Manager

Gehen Sie wie folgt vor, um ein JKS-Zertifikat mit einem Schlüssel und einem Keystore-Passwort einzurichten.

1. Generieren Sie eine Keystore-Datei für den Livy-Server.

```
keytool -genkey -alias <host> -keyalg RSA -keysize 2048 -dname
 CN=<host>,OU=hw,O=hw,L=<your_location>,ST=<state>,C=<country> -
keypass <keyPassword> -keystore <keystore_file> -storepass <storePassword> --
validity 3650
```

2. Erstellen Sie ein Zertifikat.

```
keytool -export -alias <host> -keystore mykeystore.jks -rfc -
file mycertificate.cert -storepass <storePassword>
```

3. Erstellen Sie eine Truststore-Datei.

```
keytool -import -noprompt -alias <host>-file <cert_file> -
keystore <truststore_file> -storepass <truststorePassword>
```

4. Speichern Sie das JKS-Zertifikat in. AWS Secrets Manager`livy-jks-secret`Ersetzen Sie es durch Ihr Geheimnis und `fileb://mykeystore.jks` durch den Pfad zu Ihrem Keystore-JKS-Zertifikat.

```
aws secretsmanager create-secret \
--name livy-jks-secret \
--description "My Livy keystore JKS secret" \
--secret-binary fileb://mykeystore.jks
```

5. Speichern Sie den Keystore und das Schlüsselkennwort in Secrets Manager. Stellen Sie sicher, dass Sie Ihre eigenen Parameter verwenden.

```
aws secretsmanager create-secret \
--name livy-jks-secret \
--description "My Livy key and keystore password secret" \
--secret-string "{\"keyPassword\":\"<test-key-password>\",\"keyStorePassword\":
\"<test-key-store-password>\"}"
```

6. Erstellen Sie mit dem folgenden Befehl einen Livy-Server-Namespace.

```
kubectl create ns <livy-ns>
```

7. Erstellen Sie das `ServiceProviderClass` Objekt für den Livy-Server, der über das JKS-Zertifikat und die Passwörter verfügt.

```
cat >livy-secret-provider-class.yaml << EOF
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
metadata:
  name: aws-secrets
spec:
  provider: aws
  parameters:
    objects: |
        - objectName: "livy-jks-secret"
          objectType: "secretsmanager"
        - objectName: "livy-passwords"
          objectType: "secretsmanager"

EOF
kubectl apply -f livy-secret-provider-class.yaml -n <livy-ns>
```

Erste Schritte mit dem SSL-fähigen Apache Livy

Nachdem Sie SSL auf Ihrem Livy-Server aktiviert haben, müssen Sie das einrichten, um Zugriff auf die und `serviceAccount` Secrets auf zu haben. `keyStore` `keyPasswords` AWS Secrets Manager

1. Erstellen Sie den Livy-Server-Namespace.

```
kubectl create namespace <livy-ns>
```

2. Richten Sie das Livy-Dienstkonto ein, um Zugriff auf die Geheimnisse im Secrets Manager zu haben. Weitere Informationen zur Einrichtung von IRSA finden Sie unter IRSA während der Installation von Apache Livy einrichten.

```
aws ecr get-login-password \--region region-id | helm registry login \
--username AWS \
--password-stdin ECR-registry-account.dkr.ecr.region-id.amazonaws.com
```

3. Installieren Sie Livy. Verwenden Sie für den Parameter Helm chart --version Ihr Amazon EMR-Release-Label, z. B. `7.1.0` Sie müssen auch die Amazon ECR-Registrierungskonto-ID und die Region-ID durch Ihre eigenen IDs ersetzen. Sie können den entsprechenden `ECR-registry-account` Wert für Ihre AWS-Region Amazon ECR-Registrierungskonten nach Regionen suchen.

```
helm install <livy-app-name> \
  oci://895885662937.dkr.ecr.region-id.amazonaws.com/livy \
  --version 7.1.0 \
  --namespace livy-namespace-name \
  --set image=<ECR-registry-account.dkr.ecr>.<region>.amazonaws.com/livy/
emr-7.1.0:latest \
  --set sparkNamespace=spark-namespace \
  --set ssl.enabled=true
  --set ssl.CertificateArn=livy-acm-certificate-arn
  --set ssl.secretProviderClassName=aws-secrets
  --set ssl.keyStoreObjectName=livy-jks-secret
  --set ssl.keyPasswordsObjectName=livy-passwords
  --create-namespace
```

4. Fahren Sie mit Schritt 5 der Installation von Apache Livy auf Amazon EMR auf EKS fort.

Einrichtung der Anwendungsberechtigungen für Apache Livy und Spark mit rollenbasierter Zugriffskontrolle (RBAC)

Um Livy bereitzustellen, erstellt Amazon EMR auf EKS ein Serverdienstkonto und eine Serverdienstrolle sowie ein Spark-Servicekonto und eine Rolle. Diese Rollen müssen über die erforderlichen RBAC-Berechtigungen verfügen, um die Einrichtung abzuschließen und Spark-Anwendungen auszuführen.

RBAC-Berechtigungen für das Serverdienstkonto und die Serverrolle

Amazon EMR on EKS erstellt das Livy-Serverdienstkonto und die Rolle zur Verwaltung von Livy-Sitzungen für Spark-Jobs und zur Weiterleitung von Datenverkehr zu und von den Eingangs- und anderen Ressourcen.

Der Standardname für dieses Servicekonto lautet `emr-containers-sa-livy`. Es muss über die folgenden Berechtigungen verfügen.

```
rules:
- apiGroups:
  - ""
```

```
  resources:
  - "namespaces"
  verbs:
  - "get"
- apiGroups:
  - ""
  resources:
  - "serviceaccounts"
    "services"
    "configmaps"
    "events"
    "pods"
    "pods/log"
  verbs:
  - "get"
    "list"
    "watch"
    "describe"
    "create"
    "edit"
    "delete"
    "deletecollection"
    "annotate"
    "patch"
    "label"
 - apiGroups:
    - ""
   resources:
   - "secrets"
   verbs:
   - "create"
     "patch"
     "delete"
     "watch"
 - apiGroups:
    - ""
   resources:
   - "persistentvolumeclaims"
   verbs:
   - "get"
     "list"
     "watch"
     "describe"
     "create"
```

```
    "edit"
    "delete"
    "annotate"
    "patch"
    "label"
```

RBAC-Berechtigungen für das Spark-Dienstkonto und die Spark-Rolle

Ein Spark-Treiber-Pod benötigt ein Kubernetes-Servicekonto im selben Namespace wie der Pod. Dieses Dienstkonto benötigt Berechtigungen zur Verwaltung von Executor-Pods und allen Ressourcen, die für den Treiber-Pod benötigt werden. Sofern das Standarddienstkonto im Namespace nicht über die erforderlichen Berechtigungen verfügt, schlägt der Treiber fehl und wird beendet. Die folgenden RBAC-Berechtigungen sind erforderlich.

```
rules:
- apiGroups:
  - ""
    "batch"
    "extensions"
    "apps"
  resources:
  - "configmaps"
    "serviceaccounts"
    "events"
    "pods"
    "pods/exec"
    "pods/log"
    "pods/portforward"
    "secrets"
    "services"
    "persistentvolumeclaims"
    "statefulsets"
  verbs:
  - "create"
    "delete"
    "get"
    "list"
    "patch"
    "update"
    "watch"
    "describe"
    "edit"
    "deletecollection"
```

```
    "patch"
    "label"
```

Einrichtung von Zugriffsberechtigungen mit IAM-Rollen für Dienstkonten (IRSA)

Standardmäßig haben der Livy-Server und die Treiber und Executoren der Spark-Anwendung keinen Zugriff auf Ressourcen. AWS Das Serverdienstkonto und das Spark-Dienstkonto steuern den Zugriff auf AWS Ressourcen für den Livy-Server und die Pods der Spark-Anwendung. Um Zugriff zu gewähren, müssen Sie die Dienstkonten einer IAM-Rolle zuordnen, die über die erforderlichen AWS Berechtigungen verfügt.

Sie können die IRSA-Zuordnung vor der Installation von Apache Livy, während der Installation oder nach Abschluss der Installation einrichten.

IRSA während der Installation von Apache Livy einrichten (für ein Serverdienstkonto)

> **Note**
>
> Diese Zuordnung wird nur für das Serverdienstkonto unterstützt.

1. Stellen Sie sicher, dass Sie die Einrichtung von Apache Livy für Amazon EMR auf EKS abgeschlossen haben und gerade dabei sind, Apache Livy mit Amazon EMR auf EKS zu installieren.
2. Erstellen Sie einen Kubernetes-Namespace für den Livy-Server. In diesem Beispiel lautet der Name des Namespaces. `livy-ns`
3. Erstellen Sie eine IAM-Richtlinie, die die Berechtigungen AWS-Services für die enthält, auf die Ihre Pods zugreifen sollen. Im folgenden Beispiel wird eine IAM-Richtlinie zum Abrufen von Amazon S3 S3-Ressourcen für den Spark-Einstiegspunkt erstellt.

```
cat >my-policy.json <<EOF{
"Version": "2012-10-17",
    "Statement": [
        {
"Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::my-spark-entrypoint-bucket"
        }
    ]
```

```
}
EOF

aws iam create-policy --policy-name my-policy --policy-document file://my-
policy.json
```

4. Verwenden Sie den folgenden Befehl, um Ihre AWS-Konto ID auf eine Variable festzulegen.

```
account_id=$(aws sts get-caller-identity --query "Account" --output text)
```

5. Setzen Sie den OpenID Connect (OIDC) -Identitätsanbieter Ihres Clusters auf eine Umgebungsvariable.

```
oidc_provider=$(aws eks describe-cluster --name my-cluster --region $AWS_REGION --
query "cluster.identity.oidc.issuer" --output text | sed -e "s/^https:\/\///")
```

6. Legen Sie Variablen für den Namespace und den Namen des Servicekontos fest. Achten Sie darauf, Ihre eigenen Werte zu verwenden.

```
export namespace=default
export service_account=my-service-account
```

7. Erstellen Sie mit dem folgenden Befehl eine Vertrauensrichtliniendatei. Wenn Sie allen Dienstkonten in einem Namespace Zugriff auf die Rolle gewähren möchten, kopieren Sie den folgenden Befehl und ersetzen Sie ihn durch `StringLike` und `StringEquals` `$service_account` ersetzen * Sie ihn durch.

```
cat >trust-relationship.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::$account_id:oidc-provider/$oidc_provider"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "$oidc_provider:aud": "sts.amazonaws.com",
          "$oidc_provider:sub": "system:serviceaccount:$namespace:$service_account"
        }
```

```
      }
    }
  ]
}
EOF
```

8. Erstellen Sie die -Rolle.

```
aws iam create-role --role-name my-role --assume-role-policy-document file://trust-
relationship.json --description "my-role-description"
```

9. Verwenden Sie den folgenden Helm-Installationsbefehl, um die IRSA `serviceAccount.executionRoleArn` auf die Zuordnung festzulegen. Das Folgende ist ein Beispiel für den Befehl Helm install. Sie können den entsprechenden `ECR-registry-account` Wert für Ihre AWS-Region Amazon ECR-Registrierungskonten nach Regionen suchen.

```
helm install livy-demo \
  oci://895885662937.dkr.ecr.us-west-2.amazonaws.com/livy \
  --version 7.1.0 \
  --namespace livy-ns \
  --set image=ECR-registry-account.dkr.ecr.region-id.amazonaws.com/livy/
emr-7.1.0:latest \
  --set sparkNamespace=spark-ns \
  --set serviceAccount.executionRoleArn=arn:aws:iam::123456789012:role/my-role
```

Zuordnung von IRSA zu einem Spark-Servicekonto

Bevor Sie IRSA einem Spark-Dienstkonto zuordnen, stellen Sie sicher, dass Sie die folgenden Schritte abgeschlossen haben:

- Stellen Sie sicher, dass Sie die Einrichtung von Apache Livy für Amazon EMR auf EKS abgeschlossen haben und gerade dabei sind, Apache Livy mit Amazon EMR auf EKS zu installieren.
- Sie müssen über einen vorhandenen IAM OpenID Connect (OIDC) -Anbieter für Ihren Cluster verfügen. Informationen darüber, ob Sie bereits einen haben oder wie Sie einen erstellen, finden Sie unter Erstellen eines IAM-OIDC-Anbieters für Ihren Cluster.
- Stellen Sie sicher, dass Sie Version 0.171.0 oder höher der `eksctl` CLI installiert haben oder. AWS CloudShell Informationen zur Installation oder Aktualisierung `eksctl` finden Sie unter Installation der `eksctl` Dokumentation.

Gehen Sie wie folgt vor, um IRSA Ihrem Spark-Dienstkonto zuzuordnen:

1. Verwenden Sie den folgenden Befehl, um das Spark-Dienstkonto abzurufen.

```
SPARK_NAMESPACE=<spark-ns>
LIVY_APP_NAME=<livy-app-name>
kubectl --namespace $SPARK_NAMESPACE describe sa -l "app.kubernetes.io/instance=
$LIVY_APP_NAME" | awk '/^Name:/ {print $2}'
```

2. Legen Sie Ihre Variablen für den Namespace und den Namen des Dienstkontos fest.

```
export namespace=default
export service_account=my-service-account
```

3. Verwenden Sie den folgenden Befehl, um eine Vertrauensrichtliniendatei für die IAM-Rolle zu erstellen. Das folgende Beispiel erteilt allen Dienstkonten innerhalb des Namespace die Erlaubnis, die Rolle zu verwenden. Ersetzen Sie dazu durch `StringLike` und `StringEquals` ersetzen Sie es `$service_account` durch *.

```
cat >trust-relationship.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::$account_id:oidc-provider/$oidc_provider"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "$oidc_provider:aud": "sts.amazonaws.com",
          "$oidc_provider:sub": "system:serviceaccount:$namespace:$service_account"
        }
      }
    }
  ]
}
EOF
```

4. Erstellen Sie die -Rolle.

```
aws iam create-role --role-name my-role --assume-role-policy-document file://trust-relationship.json --description "my-role-description"
```

5. Ordnen Sie das Server- oder Spark-Dienstkonto mit dem folgenden `eksctl` Befehl zu. Stellen Sie sicher, dass Sie Ihre eigenen Werte verwenden.

```
eksctl create iamserviceaccount --name spark-sa \
--namespace spark-namespace --cluster livy-eks-cluster \
--attach-role-arn arn:aws:iam::0123456789012:role/my-role \
--approve --override-existing-serviceaccounts
```

Installationseigenschaften für Apache Livy auf Amazon EMR auf EKS-Versionen

Bei der Installation von Apache Livy können Sie eine Version des Livy Helm-Diagramms auswählen. Das Helm-Diagramm bietet eine Vielzahl von Eigenschaften, mit denen Sie Ihr Installations- und Einrichtungserlebnis individuell anpassen können. Diese Eigenschaften werden für Amazon EMR auf EKS-Versionen 7.1.0 und höher unterstützt.

Themen

- Amazon EMR 7.1.0-Installationseigenschaften

Amazon EMR 7.1.0-Installationseigenschaften

In der folgenden Tabelle werden alle unterstützten Livy-Eigenschaften beschrieben. Bei der Installation von Apache Livy können Sie die Livy Helm-Diagrammversion wählen. Verwenden Sie den Befehl, um während der Installation eine Eigenschaft festzulegen. `--set <property>=<value>`

Property (Eigenschaft)	Description (Beschreibung)	Standard
Abbild	Die Amazon EMR-Release-URI des Livy-Servers. Dies ist eine erforderliche Konfiguration.	""

Property (Eigenschaft)	Description (Beschreibung)	Standard
Spark-Namespace	Namespace zum Ausführen von Livy Spark-Sitzungen. Geben Sie beispielsweise „Livy“ an. Dies ist eine erforderliche Konfiguration.	""
NameOverride	Geben Sie einen Namen anstelle von `livy` ein. Der Name wird als Bezeichnung für alle Livy-Ressourcen festgelegt	„Livy“
Vollständiger Name überschreiben	Geben Sie einen Namen an, der anstelle der vollständigen Namen der Ressourcen verwendet werden soll.	""
ssl.aktiviert	Aktiviert end-to-end SSL vom Livy-Endpunkt zum Livy-Server.	FALSE
SSL.CertificateN	Wenn SSL aktiviert ist, ist dies der vom Dienst erstellte ACM-Zertifikat-ARN für den NLB.	""
ssl. secretProviderClassName	Wenn SSL aktiviert ist, ist dies der geheime Anbieter-Klassenname zur Sicherung von NLB für die Livy-Serververbindung mit SSL.	""
ssl. keyStoreObjectName	Wenn SSL aktiviert ist, der Objektname für das Keystore-Zertifikat in der Secret-Provider-Klasse.	""

Property (Eigenschaft)	Description (Beschreibung)	Standard
ssl. keyPasswordsObject Name	Wenn SSL aktiviert ist, der Objektname für das Geheimnis, das den Keystore und das Schlüsselkennwort enthält.	""
rbac.create	Falls wahr, werden RBAC-Ressourcen erstellt.	FALSE
ServiceAccount.Create	Falls wahr, wird ein Livy-Dienstkonto erstellt.	TRUE
ServiceAccount.Name	Der Name des Dienstkontos, das für Livy verwendet werden soll. Wenn Sie diese Eigenschaft nicht festlegen und ein Servicekonto erstellen, generiert Amazon EMR auf EKS automatisch einen Namen mithilfe der `fullname` Override-Eigenschaft.	"emr-containers-sa-livy"
Servicekonto. execution RoleArn	Der ARN für die Ausführungsrolle des Livy-Dienstkontos.	""
sparkServiceAccount.erstellen	FALLS wahr, erstellt das Spark-Dienstkonto in `.Release.Namespace`	TRUE

Property (Eigenschaft)	Description (Beschreibung)	Standard
sparkServiceAccount.name	Der Name des Dienstkontos, das für Spark verwendet werden soll. Wenn Sie diese Eigenschaft nicht festlegen und ein Spark-Servicekonto erstellen, generiert Amazon EMR auf EKS automatisch einen Namen mit der `fullnameOverride` Eigenschaft mit `-spark-livy` Suffix.	„-livy“ emr-containers-sa-spark
Dienstname	Name des Livy-Dienstes	`"emr-containers-livy"`
service.annotations	Anmerkungen zum Livy-Service	`{}`
loadbalancer.enabled	Ob ein Load Balancer für den Livy-Service erstellt werden soll, der verwendet wird, um den Livy-Endpunkt außerhalb des Amazon EKS-Clusters verfügbar zu machen.	`FALSE`

Property (Eigenschaft)	Description (Beschreibung)	Standard
loadbalancer.internal	Ob der Livy-Endpunkt als intern in der VPC oder extern konfiguriert werden soll. Wenn Sie diese Eigenschaft auf setzen`FALSE`, wird der Endpunkt Quellen außerhalb der VPC zugänglich gemacht. Wir empfehlen, Ihren Endpunkt mit TLS/SSL zu sichern. Weitere Informationen finden Sie unter TLS- und SSL-Verschlüsselung einrichten.	`FALSE`
imagePullSecrets	Die Liste der `imagePullSecret` Namen, die verwendet werden sollen, um das Livy-Image aus privaten Repositorys abzurufen.	[]
Ressourcen	Die Ressourcenanforderungen und Grenzwerte für Livy-Container.	{}
NodeSelector	Die Knoten, für die Livy-Pods geplant werden sollen.	{}
Toleranzen	Eine Liste mit den zu definierenden Toleranzen für Livy-Pods.	[]
Affinität	Die Affinitätsregeln der Livy Pods.	{}

Property (Eigenschaft)	Description (Beschreibung)	Standard
Persistenz.aktiviert	Wenn der Wert true ist, wird die Persistenz für Sitzungsv erzeichnisse aktiviert.	FALSE
persistence.subPath	Der PVC-Unterpfad, der in die Sitzungsverzeichnisse eingebunden werden soll.	""
Persistence.ExistingClaim	Das PVC, das verwendet werden soll, anstatt ein neues zu erstellen.	{}
persistence.storageClass	Die zu verwendende Speicherklasse. Verwenden Sie das Format, um diesen Parameter zu definiere `nstorageClassName: <storageClass>` . Wenn Sie diesen Parameter auf einstellen, wird die dynamisch e Bereitstellung "-" deaktivie rt. Wenn Sie diesen Parameter auf Null setzen oder nichts angeben, legt Amazon EMR on EKS keinen Wert fest storageClassName und verwendet den Standard-Provisioner.	""
persistence.AccessMode	Der PVC-Zugriffsmodus.	`ReadWriteOnce`
persistence.size	Die PVC-Größe.	20 Gi
Persistenz. Anmerkungen	Zusätzliche Anmerkungen für das PVC.	{}

Property (Eigenschaft)	Description (Beschreibung)	Standard
Umgebung. *	Zusätzliche Umgebungen, die auf den Livy-Container gesetzt werden sollen. Weitere Informationen finden Sie unter Eingabe Ihrer eigenen Livy- und Spark-Konfigurationen während der Installation von Livy.	{}
Umgebung von. *	Zusätzliche Umgebungen, die aus einer Kubernetes-Konfigurationsmap oder einem Secret auf Livy gesetzt werden können.	[]
LivyConf. *	Zusätzliche livy.conf-Einträge , die aus einer bereitgestellten Kubernetes-Konfigurationsübersicht oder einem geheimen Schlüssel gesetzt werden können.	{}
sparkDefaultsConf.*	Zusätzliche `spark-defaults.conf` Einträge, die aus einer bereitgestellten Kubernetes-Konfigurationsübersicht oder einem geheimen Schlüssel gesetzt werden können.	{}

Fehlerbehebung

Eingabe Ihrer eigenen Livy- und Spark-Konfigurationen während der Installation von Livy

Sie können jede Apache Livy- oder Apache Spark-Umgebungsvariable mit der Helm-Eigenschaft konfigurieren. `env.*` Gehen Sie wie folgt vor, um die Beispielkonfiguration in ein unterstütztes Umgebungsvariablenformat `example.config.with-dash.withUppercase` zu konvertieren.

1. Ersetzen Sie Großbuchstaben durch eine 1 und einen Kleinbuchstaben des Buchstabens. Beispielsweise wird `example.config.with-dash.withUppercase` zu `example.config.with-dash.with1uppercase`.
2. Ersetzen Sie Bindestriche (-) durch 0. Zum Beispiel wird `example.config.with-dash.with1uppercase` `example.config.with0dash.with1uppercase`
3. Ersetzt Punkte (.) durch Unterstriche (_). Beispielsweise wird `example.config.with0dash.with1uppercase` zu `example_config_with0dash_with1uppercase`.
4. Ersetze alle Kleinbuchstaben durch Großbuchstaben.
5. Fügen Sie dem Variablennamen `LIVY_` das Präfix hinzu.
6. Verwenden Sie die Variable bei der Installation von Livy über das Helmchart im Format --set env. *`YOUR_VARIABLE_NAME .value= dein Wert`*

Um beispielsweise die Livy- und Spark-Konfigurationen festzulegen und diese Helm-Eigenschaften zu verwenden: `livy.server.recovery.state-store = filesystem` `spark.kubernetes.executor.podNamePrefix = my-prefix`

```
–set env.LIVY_LIVY_SERVER_RECOVERY_STATE0STORE.value=filesystem
–set env.LIVY_SPARK_KUBERNETES_EXECUTOR_POD0NAME0PREFIX.value=myprefix
```

Verwalten von Aufgabenausführungen von Amazon EMR in EKS

In den folgenden Abschnitten werden Themen behandelt, die Sie bei der Verwaltung Ihrer Aufgabenausführungen in Amazon EMR in EKS unterstützen.

Themen

- Verwalten von Aufgabenläufen mit dem AWS CLI
- Ausführen von Spark-SQL-Skripts über die StartJobRun-API
- Status von Aufgabenausführungen
- Aufträge in der Amazon-EMR-Konsole anzeigen

- Häufige Fehler beim Ausführen von Aufträgen

Verwalten von Aufgabenläufen mit dem AWS CLI

Auf dieser Seite wird beschrieben, wie Aufgabenausführungen mit dem AWS Command Line Interface (AWS CLI) verwaltet werden.

Optionen für die Konfiguration einer Aufgabenausführung

Verwenden Sie die folgenden Optionen, um die Aufgaben-Ausführungsparameter zu konfigurieren:

- `--execution-role-arn`: Sie müssen eine IAM-Rolle angeben, die für die Ausführung von Aufgaben verwendet wird. Weitere Informationen finden Sie unter Auftragausführungsrollen mit Amazon EMR in EKS verwenden.
- `--release-label`: Sie können Amazon EMR in EKS mit den Amazon-EMR-Versionen 5.32.0 und 6.2.0 und höher bereitstellen. Amazon EMR in EKS wird in früheren Amazon-EMR-Release-Versionen nicht unterstützt. Weitere Informationen finden Sie unter Versionen von Amazon EMR in EKS.
- `--job-driver`: Der Auftrag-Treiber wird verwendet, um Eingaben für die Hauptaufgabe bereitzustellen. Dies ist ein Feld vom Typ Union, in das Sie nur einen der Werte für den Aufgabentyp übergeben können, den Sie ausführen möchten. Unterstützte Aufgabentypen sind:
  - Spark-Submit-Aufgaben - Wird verwendet, um einen Befehl über Spark-Submit auszuführen. Sie können diesen Aufgabentyp verwenden, um Scala, PySpark, SparkR, SparkSQL und alle anderen unterstützten Aufgaben über Spark-Submit auszuführen. Dieser Aufgabentyp hat die folgenden Parameter:
    - Entrypoint – Dies ist der HCFS-Verweis (Hadoop Compatible File System) auf die Jar/PY-Hauptdatei, die Sie ausführen möchten.
    - EntryPointArguments - Dies ist eine Reihe von Argumenten, die Sie an Ihre Jar/PY-Hauptdatei übergeben möchten. Sie sollten das Lesen dieser Parameter mit Ihrem Einstiegspunkt-Code regeln. Jedes Argument im Array muss durch ein Komma getrennt werden. EntryPointArguments dürfen keine Klammern oder Klammern wie (), {} oder [] enthalten.
    - SparkSubmitParameters - Dies sind die zusätzlichen Spark-Parameter, die Sie an den Auftrag senden möchten. Verwenden Sie diesen Parameter, um Spark-Standardeigenschaften wie Treiberspeicher oder Anzahl der Ausführer wie -conf oder -class zu überschreiben. Weitere Informationen finden Sie unter Starten von Anwendungen mit Spark-Submit.

- Spark-SQL-Aufträge – Wird verwendet, um eine SQL-Abfragedatei über Spark SQL auszuführen. Sie können diesen Auftragtyp verwenden, um SparkSQL-Aufträge auszuführen. Dieser Aufgabentyp hat die folgenden Parameter:
  - Einstiegspunkt – Dies ist der HCFS-Verweis (Hadoop Compatible File System) auf die SQL-Abfragedatei, die Sie ausführen möchten.

    Eine Liste zusätzlicher Spark-Parameter, die Sie für einen Spark-SQL-Auftrag verwenden können, finden Sie unter Ausführen von Spark-SQL-Skripts über die StartJobRun-API.
- `--configuration-overrides`: Sie können die Standardkonfigurationen für die Anwendungen überschreiben, indem Sie ein Konfigurationsobjekt angeben. Sie können eine Syntax-Kurznotation verwenden, um die Konfiguration anzugeben oder Sie können auf die Konfiguration in einer JSON-Datei zu verweisen. Konfigurationsobjekte bestehen aus einer Klassifizierung, Eigenschaften und optionalen verschachtelten Konfigurationen. Eigenschaften bestehen aus den Einstellungen, die Sie in dieser Datei überschreiben möchten. Sie können mehrere Klassifizierungen für mehrere Anwendungen in einem einzigen JSON-Objekt angeben. Die verfügbaren Konfigurationsklassifizierungen variieren je nach Amazon-EMR-Version. Eine Liste der Konfigurationsklassifizierungen, die für jede Version von Amazon EMR verfügbar sind, finden Sie unter Versionen von Amazon EMR in EKS.

  Wenn Sie dieselbe Konfiguration in einer Anwendungsüberschreibung und in den Spark-Submit-Parametern übergeben, haben die Spark-Submit-Parameter Vorrang. Es folgt die vollständige Liste der Konfigurationsprioritäten, in der Reihenfolge von höchster Priorität bis niedrigster Priorität.

  - Konfiguration, die bei der Erstellung von `SparkSession` angegeben wurde.
  - Die Konfiguration wurde im Rahmen der `sparkSubmitParameters` unter `–conf`-Verwendung bereitgestellt.
  - Konfiguration, die im Rahmen von Anwendungsüberschreibungen bereitgestellt wird.
  - Optimierte Konfigurationen, die von Amazon EMR für die Veröffentlichung ausgewählt wurden.
  - Open-Source-Standardkonfigurationen für die Anwendung.

  Um Aufgabenausführungen mit Amazon CloudWatch oder Amazon S3 zu überwachen, müssen Sie die Konfigurationsdetails für CloudWatch angeben. Weitere Informationen finden Sie unter Eine Aufgabenausführung für die Verwendung von Amazon-S3-Protokollen konfigurieren und Einen Aufgabenlauf für die Verwendung von Amazon CloudWatch Logs konfigurieren. Wenn der S3-Bucket oder die CloudWatch-Protokollgruppe nicht existiert, erstellt Amazon EMR sie, bevor die Protokolle in den Bucket hochgeladen werden.

- Eine zusätzliche Liste der Kubernetes-Konfigurationsoptionen finden Sie unter Spark-Eigenschaften auf Kubernetes.

  Die folgenden Spark-Konfigurationen werden nicht unterstützt.

  - `spark.kubernetes.authenticate.driver.serviceAccountName`
  - `spark.kubernetes.authenticate.executor.serviceAccountName`
  - `spark.kubernetes.namespace`
  - `spark.kubernetes.driver.pod.name`
  - `spark.kubernetes.container.image.pullPolicy`
  - `spark.kubernetes.container.image`

  > **Note**
  >
  > Sie können `spark.kubernetes.container.image` für benutzerdefinierte Docker-Images verwenden. Weitere Informationen finden Sie unter Anpassen von Docker-Images für Amazon EMR in EKS.

Eine Aufgabenausführung für die Verwendung von Amazon-S3-Protokollen konfigurieren

Um den Aufgabenfortschritt überwachen und Fehler beheben zu können, müssen Sie Ihre Aufträge so konfigurieren, dass Protokollinformationen an Amazon S3, Amazon CloudWatch Logs oder beide gesendet werden. Dieses Thema hilft Ihnen bei den ersten Schritten beim Veröffentlichen von Anwendungsprotokollen in Amazon S3 für Ihre Aufträge, die mit Amazon EMR in EKS gestartet wurden.

S3 protokolliert die IAM-Richtlinie

Bevor Ihre Aufträge Protokolldaten an Amazon S3 senden können, müssen die folgenden Berechtigungen in der Berechtigungsrichtlinie für die Auftragausführungsrolle enthalten sein. Ersetzen Sie *DOC-EXAMPLE-BUCKET-LOGGING* durch den Namen ihres Protokollierungs-Buckets.

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

```
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET-LOGGING",
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET-LOGGING/*",
            ]
        }
    ]
}
```

> **Note**
>
> Amazon EMR in EKS kann auch einen Amazon-S3-Bucket erstellen. Wenn ein Amazon-S3-Bucket nicht verfügbar ist, nehmen Sie die `"s3:CreateBucket"`-Erlaubnis in die IAM-Richtlinie auf.

Nachdem Sie Ihrer Ausführungsrolle die entsprechenden Berechtigungen zum Senden von Protokollen an Amazon S3 erteilt haben, werden Ihre Protokolldaten an die folgenden Amazon-S3-Speicherorte gesendet, wenn `s3MonitoringConfiguration` im `monitoringConfiguration` Abschnitt einer `start-job-run`-Anforderung übergeben werden, wie unter Verwalten von Aufgabenläufen mit dem AWS CLI beschrieben.

- Controller-Protokolle – /*logUri*/*virtual-cluster-id*/jobs/*job-id*/containers/*pod-name/*(stderr.gz/stdout.gz)
- Treiberprotokolle – /*logUri*/*virtual-cluster-id*/jobs/*job-id*/containers/*spark-application-id*/spark-*job-id*-driver/(stderr.gz/stdout.gz)
- Ausführungsprotokolle – /*logUri*//*virtual-cluster-id*/jobs/ *job-id*/containers/*spark-application-id*/*executor-pod-name*/(stderr.gz/stdout.gz)

Einen Aufgabenlauf für die Verwendung von Amazon CloudWatch Logs konfigurieren

Um den Aufgabenfortschritt zu überwachen und Fehler zu beheben, müssen Sie Ihre Aufgaben so konfigurieren, dass Loginformationen an Amazon S3, Amazon CloudWatch Logs oder beide gesendet werden. Dieses Thema hilft Ihnen beim Einstieg in die Verwendung von CloudWatch

Logs für Ihre Aufträge, die mit Amazon EMR in EKS gestartet werden. Weitere Informationen zu CloudWatch Logs finden Sie unter Überwachen von Protokolldateien im Amazon-CloudWatch-Benutzerhandbuch.

CloudWatch-Logs-IAM-Richtlinie

Damit Ihre Aufträge Protokolldaten an CloudWatch Logs senden können, müssen die folgenden Berechtigungen in der Berechtigungsrichtlinie für die Aufgabenausführungsrolle enthalten sein. Ersetzen Sie *my_log_group_name* und *my_log_stream_prefix* durch die Namen Ihrer CloudWatch-Protokollgruppe bzw. Protokollstream-Namen. Amazon EMR in EKS erstellt die Protokollgruppe und den Protokollstream, falls sie nicht existieren, solange der ARN für die Ausführungsrolle über die entsprechenden Berechtigungen verfügt.

```
{
     "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:*:*:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:*:*:log-group:my_log_group_name:log-
stream:my_log_stream_prefix/*"
            ]
        }
    ]
}
```

> **Note**
>
> Amazon EMR in EKS kann auch einen Protokollstream erstellen. Wenn kein Protokollstream vorhanden ist, sollte die IAM-Richtlinie die `"logs:CreateLogGroup"` Genehmigung enthalten.

Nachdem Sie Ihrer Ausführungsrolle die entsprechenden Berechtigungen erteilt haben, sendet Ihre Anwendung ihre Protokolldaten an CloudWatch Logs, wenn sie `cloudWatchMonitoringConfiguration` im `monitoringConfiguration`-Abschnitt einer `start-job-run`-Anforderung übergeben werden, wie unter Verwalten von Aufgabenläufen mit dem AWS CLI gezeigt.

In der `StartJobRun`-API ist *`log_group_name der Protokollgruppenname`* für CloudWatch und *`log_stream_prefix ist das Protokollstream-Namenspräfix`* für CloudWatch. Sie können diese Protokolle in der AWS Management Console anzeigen und durchsuchen.

- Controller-Protokolle – *`LogGroup`*/*`LogStreamPrefix`*/*`virtuelle Cluster-ID`*/jobs/ *`job-id`*/containers/*`pod-name`*/(stderr/stdout)
- Treiberprotokolle – *`logGroup`*/*`logStreamPrefix`*/*`virtual-cluster-id`*/jobs/ *`job-id`*/containers/*`spark-application-id`*/spark-*`job-id`*-driver/ (stderrstdout)
- Ausführungsprotokolle – *`logGroup`*/*`logStreamPrefix`*/*`virtual-cluster-id`*/jobs/*`job-id`*/containers/*`spark-application-id`*/*`executor-pod-name`*/(stderr/stdout)

Aufgabenausführungen auflisten

Sie können `list-job-run` ausführen, um den Status der Aufgabenausführungen anzuzeigen, wie das folgende Beispiel zeigt.

```
aws emr-containers list-job-runs --virtual-cluster-id <cluster-id>
```

Eine Aufgabenausführung beschreiben

Sie können `describe-job-run` ausführen, um weitere Informationen über die Aufgabe abzurufen, z. B. den Status, die Statusdetails und den Aufgabennamen, wie das folgende Beispiel zeigt.

```
aws emr-containers describe-job-run --virtual-cluster-id cluster-id --id job-run-id
```

Eine Aufgabenausführung abbrechen

Sie können `cancel-job-run` ausführen, um laufende Aufträge abzubrechen, wie das folgende Beispiel zeigt.

```
aws emr-containers cancel-job-run --virtual-cluster-id cluster-id --id job-run-id
```

Ausführen von Spark-SQL-Skripts über die StartJobRun-API

Die Versionen 6.7.0 und höher von Amazon EMR in EKS enthalten einen Spark-SQL-Auftrag-Treiber, sodass Sie Spark-SQL-Skripts über die `StartJobRun`-API ausführen können. Sie können SQL-Einstiegspunktdateien bereitstellen, um Spark-SQL-Abfragen direkt auf Amazon EMR in EKS mit der `StartJobRun`-API auszuführen, ohne Änderungen an vorhandenen Spark-SQL-Skripts vornehmen zu müssen. In der folgenden Tabelle sind Spark-Parameter aufgeführt, die für die Spark-SQL-Aufträge über die StartJobRun-API unterstützt werden.

Sie können aus den folgenden Spark-Parametern wählen, um sie an einen Spark-SQL-Auftrag zu senden. Verwenden Sie diese Parameter, um die Spark-Standardeigenschaften zu überschreiben.

Option	Beschreibung
--Name NAME	Anwendungsname
--jars JARS	Durch Kommas getrennte Liste von JAR-Dateien, die in den Klassenpfad des Treibers und der Ausführung aufgenommen werden sollen.
--packages	Durch Kommas getrennte Liste der Maven-Koordinaten von JAR-Dateien, die in die Klassenpfade des Treibers und des Ausführers aufgenommen werden sollen.
--exclude-packages	Durch Kommas getrennte Liste von groupID:artifactId, die beim Auflösen der in –packages angegebenen Abhängigkeiten ausgeschlossen werden sollen, um Abhängigkeitskonflikte zu vermeiden.

Option	Beschreibung
--repositories	Kommagetrennte Liste zusätzlicher Remote-Repositorys, um nach den mit –packages angegebenen Maven-Koordinaten zu suchen.
--files FILES	Durch Kommas getrennte Liste von Dateien, die im Arbeitsverzeichnis jedes Ausführers abgelegt werden sollen.
--conf PROP=VALUE	Spark-Konfigurationseigenschaft.
--properties-file FILE	Pfad zu einer Datei, aus der zusätzliche Eigenschaften geladen werden sollen.
--driver-memory MEM	Arbeitsspeicher für den Treiber. Standard 1 024 MB.
--driver-java-options	Zusätzliche Java-Optionen, die an den Treiber übergeben werden.
--driver-library-path	Zusätzliche Bibliothekspfadeinträge, die an den Treiber übergeben werden.
--driver-class-path	Zusätzliche Klassenpfadeinträge, die an den Treiber übergeben werden.
--executor-memory MEM	Arbeitsspeicher pro Ausführer. Standard: 1 GB.
--driver-cores NUM	Anzahl der vom Treiber verwendeten Kerne.
--total-executor-cores NUM	Gesamtzahl der Kerne für alle Ausführer.
--executor-cores NUM	Anzahl der von jedem Ausführer verwendeten Kerne.
--num-executors NUM	Anzahl der zu startenden Ausführer.
-hivevar <key=value>	Variablenersetzung zur Anwendung auf Hive-Befehle, zum Beispiel `-hivevar A=B`

Option	Beschreibung
-hiveconf <property=value>	Wert, der für die angegebene Eigenschaft verwendet werden soll.

Erstellen Sie für einen Spark-SQL-Auftrag eine start-job-run-request.json-Datei und geben Sie die erforderlichen Parameter für Ihre Auftragsausführung an, wie im folgenden Beispiel:

```
{
  "name": "myjob",
  "virtualClusterId": "123456",
  "executionRoleArn": "iam_role_name_for_job_execution",
  "releaseLabel": "emr-6.7.0-latest",
  "jobDriver": {
    "sparkSqlJobDriver": {
      "entryPoint": "entryPoint_location",
       "sparkSqlParameters": "--conf spark.executor.instances=2 --conf
 spark.executor.memory=2G --conf spark.executor.cores=2 --conf spark.driver.cores=1"
    }
  },
  "configurationOverrides": {
    "applicationConfiguration": [
      {
        "classification": "spark-defaults",
        "properties": {
          "spark.driver.memory":"2G"
         }
      }
    ],
    "monitoringConfiguration": {
      "persistentAppUI": "ENABLED",
      "cloudWatchMonitoringConfiguration": {
        "logGroupName": "my_log_group",
        "logStreamNamePrefix": "log_stream_prefix"
      },
      "s3MonitoringConfiguration": {
        "logUri": "s3://my_s3_log_location"
      }
    }
  }
}
```

Status von Aufgabenausführungen

Wenn Sie eine Aufgabe an eine Aufgabenwarteschlange von Amazon EMR in EKS übergeben, geht die Aufgabe in den `PENDING`-Status über. Danach durchläuft er die folgenden Zustände, bis er erfolgreich ist (Abschluss mit Code `0`) oder fehlschlägt (Abschluss mit einem Code ungleich null).

Aufträge können die folgenden Status haben:

- `PENDING` – Der anfängliche Aufgabenstatus, wenn die Aufgabenausführung an Amazon EMR in EKS übermittelt wird. Der Auftrag wartet darauf, an den virtuellen Cluster übermittelt zu werden, und Amazon EMR in EKS arbeitet daran, diesen Auftrag zu senden.
- `SUBMITTED` – Eine Aufgabenausführung, die erfolgreich an den virtuellen Cluster übermittelt wurde. Der Cluster-Scheduler versucht dann, diesen Auftrag auf dem Cluster auszuführen.
- `RUNNING` – Ein Auftrag, der im virtuellen Cluster ausgeführt wird. In Spark-Anwendungen bedeutet dies, dass sich der Spark-Treiberprozess im `running`-Status befindet.
- `FAILED` – Eine Aufgabenausführung, die nicht an den virtuellen Cluster übermittelt werden konnte oder die erfolglos abgeschlossen wurde. Weitere Informationen zu diesem Auftragsfehler finden Sie unter StateDetails und FailureReason.
- `COMPLETED` – Eine Aufgabenausführung, die erfolgreich abgeschlossen wurde.
- `CANCEL_PENDING` – Ein Auftragslauf wurde zur Stornierung angefordert. Amazon EMR in EKS versucht, den Auftrag auf dem virtuellen Cluster abzubrechen.
- `CANCELLED` – Eine Aufgabenausführung, die erfolgreich abgebrochen wurde.

Aufträge in der Amazon-EMR-Konsole anzeigen

Gehen Sie wie folgt vor, um Aufträge in der Amazon-EMR-Konsole anzuzeigen.

1. Wählen Sie im linken Menü der Amazon-EMR-Konsole unter Amazon EMR in EKS die Option Virtuelle Cluster aus.
2. Wählen Sie in der Liste der virtuellen Cluster den virtuellen Cluster aus, für den Sie die Aufträge anzeigen möchten.
3. Wählen Sie in der Tabelle Aufgabenausführungen die Option Protokolle anzeigen aus, um die Details einer Aufgabenausführung anzuzeigen.

> **Note**
>
> Die Support für das Ein-Klick-Erlebnis ist standardmäßig aktiviert. Sie kann deaktiviert werden, indem `persistentAppUI` während der Aufgabenübermittlung in `DISABLED` auf `monitoringConfiguration` gesetzt wird. Weitere Informationen finden Sie unter Anzeigen von persistenten Anwendungsbenutzeroberflächen.

Häufige Fehler beim Ausführen von Aufträgen

Beim Ausführen der `StartJobRun`-API können folgende Fehler auftreten.

Fehlermeldung	Fehlerbedingung	Empfohlene nächste Schritte
Fehler: Argument – *`Argument`* ist erforderlich	Erforderliche Parameter fehlen.	Fügen Sie der API-Anfrage die fehlenden Argumente hinzu.
Beim Aufrufen des StartJobRun-Vorgangs ist ein Fehler aufgetreten (AccessDeniedException): Benutzer: *`ARN`* ist nicht berechtigt, Folgendes auszuführen: emr-containers:StartJobRun	Die Ausführungsrolle fehlt.	Siehe Verwenden von Auftragausführungsrollen mit Amazon EMR in EKS verwenden.
Beim Aufrufen des StartJobRun-Vorgangs ist ein Fehler aufgetreten (AccessDeniedException): Benutzer: *`ARN`* ist nicht berechtigt, Folgendes auszuführen: emr-containers:StartJobRun	Der Aufrufer hat keine Zugriffsrechte für die Ausführungsrolle [gültiges/nicht gültiges Format] über Bedingungsschlüssel.	Siehe Auftragausführungsrollen mit Amazon EMR in EKS verwenden.
Beim Aufrufen des StartJobRun-Vorgangs ist ein Fehler aufgetreten (AccessDeniedException): Benutzer: *`ARN`* ist nicht berechtigt, Folgendes	Der ARN für den Auftragsabsender und die Ausführungsrolle stammen von unterschiedlichen Konten.	Stellen Sie sicher, dass der Auftragsabsender und der ARN für die Ausführungsrolle aus demselben AWS-Konto stammen.

Fehlermeldung	Fehlerbedingung	Empfohlene nächste Schritte
auszuführen: emr-containers:StartJobRun		
1 Validierungsfehler erkannt: Die *Wertrolle* bei „executionRoleArn“ konnte das reguläre ARN-Ausdrucksmuster nicht erfüllen: ^arn:(aws[a-zA-Z0-9-]*):iam::(\d{12})?: (Rolle((\u002F)|(\u002F[\u0021-\u007F]+\u002F))[\w+=,.@-]+)	Der Aufrufer hat über Bedingungsschlüssel Berechtigungen für die Ausführungsrolle, aber die Rolle erfüllt nicht die Einschränkungen des ARN-Formats.	Geben Sie die Ausführungsrolle im ARN-Format an. Siehe Auftragausführungsrollen mit Amazon EMR in EKS verwenden.
Beim Aufrufen des Vorgangs StartJobRun ist ein Fehler aufgetreten (Resource NotFoundException): Die virtuelle Cluster-ID des *virtuellen Clusters* ist nicht vorhanden.	Die virtuelle Cluster-ID wurde nicht gefunden.	Geben Sie eine virtuelle Cluster-ID an, die bei Amazon EMR in EKS registriert ist.
Beim Aufrufen des Vorgangs StartAuftragRun ist ein Fehler aufgetreten (ValidationException): Der *Status* des virtuellen Clusters ist nicht gültig, um die Ressource JobRun zu erstellen.	Der virtuelle Cluster ist nicht bereit, den Auftrag auszuführen.	Siehe Status des virtuellen Clusters.
Beim Aufrufen der Operation StartJobRun ist ein Fehler aufgetreten (Resource NotFoundException): Release *RELEASE* ist nicht vorhanden.	Die in der Auftragsübermittlung angegebene Version ist falsch.	Siehe Versionen von Amazon EMR in EKS.

Fehlermeldung	Fehlerbedingung	Empfohlene nächste Schritte
Beim Aufrufen des StartJobRun-Vorgangs ist ein Fehler aufgetreten (AccessDeniedException): Benutzer: *ARN* ist nicht berechtigt, Folgendes auszuführen: emr-containers:StartJobRun für Ressource: *ARN* mit einer expliziten Ablehnung. Beim Aufrufen des StartJobRun-Vorgangs ist ein Fehler aufgetreten (AccessDeniedException): Benutzer: *ARN* ist nicht berechtigt, Folgendes auszuführen: emr-containers:StartJobRun für Ressource: *ARN*	Der Benutzer ist nicht autorisiert, StartJobRun aufzurufen.	Siehe Auftragausführungsrollen mit Amazon EMR in EKS verwenden.
Beim Aufrufen des Vorgangs StartJobRun ist ein Fehler aufgetreten (ValidationException): configurationOverrides.monitoringConfiguration.s3MonitoringConfiguration.logUri failed to satisfy constraint : %s	Die S3-Pfad-URI-Syntax ist nicht gültig.	logUri sollte das Format s3:// … haben

Die folgenden Fehler können auftreten, wenn Sie die `DescribeJobRun`-API ausführen, bevor der Auftrag ausgeführt wird.

Fehlermeldung	Fehlerbedingung	Empfohlene nächste Schritte
StateDetails: Die JobRun-Übermittlung ist fehlgeschlagen.	Die Parameter in StartJobRun sind nicht gültig.	Siehe Versionen von Amazon EMR in EKS.

Fehlermeldung	Fehlerbedingung	Empfohlene nächste Schritte
Die Klassifizierungsklassifizierung wird nicht unterstützt. failureReason: VALIDATION_ERROR state: FAILED.		
stateDetails: Die *Cluster-EKS-Cluster-ID* ist nicht vorhanden. failureReason: CLUSTER_UNAVAILABLE state: FAILED	Der EKS-Cluster ist nicht verfügbar.	Überprüfen Sie, ob der EKS-Cluster existiert und über die richtigen Berechtigungen verfügt. Weitere Informationen finden Sie unter Einrichten von Amazon EMR in EKS.
stateDetails: Die *Cluster-EKS-Cluster-ID* verfügt nicht über ausreichende Berechtigungen. failureReason: CLUSTER_UNAVAILABLE state: FAILED	Amazon EMR verfügt nicht über Berechtigungen für den Zugriff auf den EKS-Cluster.	Stellen Sie sicher, dass die Berechtigungen für Amazon EMR für den registrierten Namespace eingerichtet sind. Weitere Informationen finden Sie unter Einrichten von Amazon EMR in EKS.
stateDetails: Die *Cluster-EKS-Cluster-ID* ist derzeit nicht erreichbar. failureReason: CLUSTER_UNAVAILABLE state: FAILED	Der EKS-Cluster ist nicht erreichbar.	Überprüfen Sie, ob der EKS-Cluster existiert und über die richtigen Berechtigungen verfügt. Weitere Informationen finden Sie unter Einrichten von Amazon EMR in EKS.

Fehlermeldung	Fehlerbedingung	Empfohlene nächste Schritte
stateDetails: Die Aufgabenausführung ist aufgrund eines internen Fehlers fehlgeschlagen. failureReason: INTERNAL_ERROR state: FAILED	Es ist ein interner Fehler im EKS-Cluster aufgetreten.	–
stateDetails: Die *Cluster-EKS-Cluster-ID* verfügt nicht über ausreichende Ressourcen. failureReason: USER_ERROR state: FAILED	Im EKS-Cluster sind nicht genügend Ressourcen vorhanden, um den Auftrag auszuführen.	Fügen Sie der EKS-Knotengruppe mehr Kapazität hinzu oder richten Sie EKS Autoscaler ein. Weitere Informationen finden Sie unter Cluster-Autoscaler.

Die folgenden Fehler können auftreten, wenn Sie die `DescribeJobRun`-API ausführen, nachdem der Auftrag ausgeführt wurde.

Fehlermeldung	Fehlerbedingung	Empfohlene nächste Schritte
stateDetails: Probleme bei der Überwachung Ihres JobRun. Die *EKS-Cluster-ID* des Clusters ist nicht vorhanden. failureReason: CLUSTER_UNAVAILABLE state: FAILED	Der EKS-Cluster ist nicht vorhanden.	Überprüfen Sie, ob der EKS-Cluster existiert und über die richtigen Berechtigungen verfügt. Weitere Informationen finden Sie unter Einrichten von Amazon EMR in EKS.

Fehlermeldung	Fehlerbedingung	Empfohlene nächste Schritte
stateDetails: Probleme bei der Überwachung Ihres JobRun. Die *EKS-Cluster-ID* des Clusters verfügt nicht über ausreichende Berechtigungen. failureReason: CLUSTER_UNAVAILABLE state: FAILED	Amazon EMR verfügt nicht über Berechtigungen für den Zugriff auf den EKS-Cluster.	Stellen Sie sicher, dass die Berechtigungen für Amazon EMR für den registrierten Namespace eingerichtet sind. Weitere Informationen finden Sie unter Einrichten von Amazon EMR in EKS.
stateDetails: Probleme bei der Überwachung Ihres JobRun. Die *EKS-Cluster-ID* des Clusters ist derzeit nicht erreichbar. failureReason: CLUSTER_UNAVAILABLE state: FAILED	Der EKS-Cluster ist nicht erreichbar.	Überprüfen Sie, ob der EKS-Cluster existiert und über die richtigen Berechtigungen verfügt. Weitere Informationen finden Sie unter Einrichten von Amazon EMR in EKS.
stateDetails: Probleme beim Überwachen Ihrer Aufgabenausführung aufgrund eines internen Fehlers failureReason: INTERNAL_ERROR state: FAILED	Es ist ein interner Fehler aufgetreten, der die Überwachung von JobRun verhindert.	–

Der folgende Fehler kann auftreten, wenn ein Auftrag nicht gestartet werden kann und der Auftrag 15 Minuten im Status SUBMITTED wartet. Die Ursache können fehlende Clusterressourcen sein.

Fehlermeldung	Fehlerbedingung	Empfohlene nächste Schritte
Cluster-Timeout	Der Auftrag befindet sich seit mindestens 15 Minuten im Status SUBMITTED.	Sie können die Standarde instellung von 15 Minuten für diesen Parameter mit der unten gezeigten Konfigurationsüberschreibung überschreiben.

Verwenden Sie die folgende Konfiguration, um die Einstellung für das Cluster-Timeout auf 30 Minuten zu ändern. Beachten Sie, dass Sie den neuen `job-start-timeout`-Wert in Sekunden angeben:

```
{
"configurationOverrides": {
  "applicationConfiguration": [{
      "classification": "emr-containers-defaults",
      "properties": {
          "job-start-timeout":"1800"
      }
  }]
}
```

Verwenden der Auftragserteilung

Übersicht

Die Amazon EMR in EKS-`StartJobRun`-Anfrage erstellt einen Auftrag-Submitter-Pod (auch Auftrag-Runner-Pod genannt), um den Spark-Treiber zu starten. Mit der `emr-job-submitter`-Klassifizierung können Sie Knotenselektoren für Ihren Auftrag-Submitter-Pod konfigurieren.

Die folgende Einstellung ist unter der Klassifizierung verfügbar: `emr-job-submitter`

`jobsubmitter.node.selector.[`*`labelKey`*`]`

Fügt der Knotenauswahl des Pods, in dem der Auftrag eingereicht wurde, hinzu, wobei der Schlüssel *`labelKey`* und der Wert als Konfigurationswert für die Konfiguration verwendet werden. Sie können beispielsweise `jobsubmitter.node.selector.identifier` auf `myIdentifier` festlegen, dass der Pod des Auftragseinreichers über eine Knotenauswahl mit

dem Schlüsselkennungswert von `myIdentifier` verfügt. Um mehrere Knotenauswahltasten hinzuzufügen, legen Sie mehrere Konfigurationen mit diesem Präfix fest.

Als bewährte Methode empfehlen wir, dass Pods von Auftragseinreichern die Knoten-Platzierung auf On-Demand-Instances und nicht auf Spot Instances vornehmen. Das liegt daran, dass ein Auftrag fehlschlägt, wenn der Pod, der den Auftrag einreicht, Spot Instance-Unterbrechungen ausgesetzt ist. Sie können den Pod des Auftragseinreichers auch in einer einzigen Availability Zone platzieren oder beliebige Kubernetes-Labels verwenden, die auf die Knoten angewendet werden.

Beispiele für die Klassifizierung von Auftragseinreichern

In diesem Abschnitt

- StartJobRun-Anfrage mit On-Demand-Knotenplatzierung für den Pod des Auftrageinreichers
- StartJobRun-Anfrage mit Einzel-AZ-Knotenplatzierung für den Pod, der den Auftrag einreicht
- StartJobRun-Anfrage mit Platzierung der Instance-Typen Single-AZ und Amazon EC2 für den Auftrag-Submitter-Pod

`StartJobRun`-Anfrage mit On-Demand-Knotenplatzierung für den Pod des Auftrageinreichers

```
cat >spark-python-in-s3-nodeselector-job-submitter.json << EOF
{
  "name": "spark-python-in-s3-nodeselector",
  "virtualClusterId": "virtual-cluster-id",
  "executionRoleArn": "execution-role-arn",
  "releaseLabel": "emr-6.11.0-latest",
  "jobDriver": {
    "sparkSubmitJobDriver": {
      "entryPoint": "s3://S3-prefix/trip-count.py",
       "sparkSubmitParameters": "--conf spark.driver.cores=5  --conf
 spark.executor.memory=20G --conf spark.driver.memory=15G --conf
 spark.executor.cores=6"
    }
  },
  "configurationOverrides": {
    "applicationConfiguration": [
      {
        "classification": "spark-defaults",
        "properties": {
```

```
          "spark.dynamicAllocation.enabled":"false"
         }
      },
      {
        "classification": "emr-job-submitter",
        "properties": {
            "jobsubmitter.node.selector.eks.amazonaws.com/capacityType": "ON_DEMAND"
        }
      }
    ],
    "monitoringConfiguration": {
      "cloudWatchMonitoringConfiguration": {
        "logGroupName": "/emr-containers/jobs",
        "logStreamNamePrefix": "demo"
      },
      "s3MonitoringConfiguration": {
        "logUri": "s3://joblogs"
      }
    }
  }
}
EOF
aws emr-containers start-job-run --cli-input-json file:///spark-python-in-s3-
nodeselector-job-submitter.json
```

StartJobRun-Anfrage mit Einzel-AZ-Knotenplatzierung für den Pod, der den Auftrag einreicht

```
cat >spark-python-in-s3-nodeselector-job-submitter-az.json << EOF
{
  "name": "spark-python-in-s3-nodeselector",
  "virtualClusterId": "virtual-cluster-id",
  "executionRoleArn": "execution-role-arn",
  "releaseLabel": "emr-6.11.0-latest",
  "jobDriver": {
    "sparkSubmitJobDriver": {
      "entryPoint": "s3://S3-prefix/trip-count.py",
       "sparkSubmitParameters": "--conf spark.driver.cores=5  --conf
 spark.executor.memory=20G --conf spark.driver.memory=15G --conf
 spark.executor.cores=6"
    }
  },
  "configurationOverrides": {
```

```
    "applicationConfiguration": [
      {
        "classification": "spark-defaults",
        "properties": {
          "spark.dynamicAllocation.enabled":"false"
         }
      },
      {
        "classification": "emr-job-submitter",
        "properties": {
            "jobsubmitter.node.selector.topology.kubernetes.io/zone": "Availability
 Zone"
        }
      }
    ],
    "monitoringConfiguration": {
      "cloudWatchMonitoringConfiguration": {
        "logGroupName": "/emr-containers/jobs",
        "logStreamNamePrefix": "demo"
      },
      "s3MonitoringConfiguration": {
        "logUri": "s3://joblogs"
      }
    }
  }
}
EOF
aws emr-containers start-job-run --cli-input-json file:///spark-python-in-s3-
nodeselector-job-submitter-az.json
```

StartJobRun-Anfrage mit Platzierung der Instance-Typen Single-AZ und Amazon EC2 für den Auftrag-Submitter-Pod

```
{
  "name": "spark-python-in-s3-nodeselector",
  "virtualClusterId": "virtual-cluster-id",
  "executionRoleArn": "execution-role-arn",
  "releaseLabel": "emr-6.11.0-latest",
  "jobDriver": {
    "sparkSubmitJobDriver": {
      "entryPoint": "s3://S3-prefix/trip-count.py",
       "sparkSubmitParameters": "--conf spark.driver.cores=5  --conf
 spark.kubernetes.pyspark.pythonVersion=3 --conf spark.executor.memory=20G
```

```
  --conf spark.driver.memory=15G --conf spark.executor.cores=6 --conf
  spark.sql.shuffle.partitions=1000"
     }
   },
   "configurationOverrides": {
     "applicationConfiguration": [
       {
         "classification": "spark-defaults",
         "properties": {
           "spark.dynamicAllocation.enabled":"false",
          }
       },
       {
         "classification": "emr-job-submitter",
         "properties": {
             "jobsubmitter.node.selector.topology.kubernetes.io/zone": "Availability
  Zone",
             "jobsubmitter.node.selector.node.kubernetes.io/instance-type":"m5.4xlarge"
         }
       }
     ],
     "monitoringConfiguration": {
       "cloudWatchMonitoringConfiguration": {
         "logGroupName": "/emr-containers/jobs",
         "logStreamNamePrefix": "demo"
       },
       "s3MonitoringConfiguration": {
         "logUri": "s3://joblogs"
       }
     }
   }
}
```

Verwenden von Aufgabenvorlagen

In einer Auftragvorlage werden Werte gespeichert, die beim Starten eines Auftraglaufs von allen `StartJobRun`-API-Aufrufen gemeinsam genutzt werden können. Sie unterstützt zwei Anwendungsfälle:

- Um zu verhindern, dass sich wiederholende `StartJobRun`-API-Anforderungswerte auftreten.
- Um eine Regel durchzusetzen, nach der bestimmte Werte über `StartJobRun`-API-Anfragen bereitgestellt werden müssen.

Auftrag Auftragvorlagen können Sie eine wiederverwendbare Vorlage für Auftragläufe definieren, um zusätzliche Anpassungen vorzunehmen, zum Beispiel:

- Konfigurieren der Rechenkapazität für Executor und Treiber
- Einstellen von Sicherheits- und Governance-Eigenschaften wie IAM-Rollen
- Anpassen eines Docker-Images für die Verwendung in mehreren Anwendungen und Daten-Pipelines

Erstellen und Verwenden einer Aufgabenvorlage zum Starten einer Aufgabenausführung

In diesem Abschnitt wird beschrieben, wie Sie eine Auftragvorlage erstellen und die Vorlage verwenden, um einen Auftraglauf mit dem AWS Command Line Interface (AWS CLI) zu starten.

So erstellen Sie eine neue Aufgabenvorlage

1. Erstellen Sie eine `create-job-template-request.json`-Datei und geben Sie die erforderlichen Parameter für Ihre Aufgabenvorlage an, wie in der folgenden JSON-Beispieldatei gezeigt. Informationen zu allen verfügbaren Parametern finden Sie in der CreateJobTemplate-API.

   Die meisten Werte, die für die `StartJobRun`-API erforderlich sind, sind auch für `jobTemplateData` erforderlich. Wenn Sie Platzhalter für Parameter verwenden und Werte angeben möchten, wenn Sie StartJobRun mit einer Auftragvorlage aufrufen, lesen Sie bitte den nächsten Abschnitt über Auftragvorlagenparameter.

```
{
   "name": "mytemplate",
   "jobTemplateData": {
        "executionRoleArn": "iam_role_arn_for_job_execution",
        "releaseLabel": "emr-6.7.0-latest",
        "jobDriver": {
            "sparkSubmitJobDriver": {
                "entryPoint": "entryPoint_location",
                "entryPointArguments": [ "argument1","argument2",...],
                "sparkSubmitParameters": "--class <main_class> --conf
 spark.executor.instances=2 --conf spark.executor.memory=2G --conf
 spark.executor.cores=2 --conf spark.driver.cores=1"
            }
        },
```

```
        "configurationOverrides": {
            "applicationConfiguration": [
                {
                    "classification": "spark-defaults",
                    "properties": {
                         "spark.driver.memory":"2G"
                    }
                }
            ],
            "monitoringConfiguration": {
                "persistentAppUI": "ENABLED",
                "cloudWatchMonitoringConfiguration": {
                    "logGroupName": "my_log_group",
                    "logStreamNamePrefix": "log_stream_prefix"
                },
                "s3MonitoringConfiguration": {
                    "logUri": "s3://my_s3_log_location/"
                }
            }
        }
    }
}
```

2. Verwenden Sie den `create-job-template`-Befehl mit einem Pfad zu der lokal gespeicherten `create-job-template-request.json`-Datei.

```
aws emr-containers create-job-template \
--cli-input-json file://./create-job-template-request.json
```

So starten Sie eine Auftragsausführung mit einer Auftragsvorlage

Geben Sie die virtuelle Cluster-ID, die Aufgabenvorlagen-ID und den Aufgabennamen im `StartJobRun`-Befehl an, wie im folgenden Beispiel veranschaulicht.

```
aws emr-containers start-job-run \
--virtual-cluster-id 123456 \
--name myjob \
--job-template-id 1234abcd
```

Definieren von Aufgabenvorlagenparametern

Mit Auftragvorlagenparametern können Sie Variablen in der Auftragvorlage angeben. Werte für diese Parametervariablen müssen angegeben werden, wenn eine Aufgabenausführung mit dieser Aufgabenvorlage gestartet wird. Auftragsvorlagenparameter werden im `${parameterName}`-Format angegeben. Sie können einen beliebigen Wert in einem `jobTemplateData`-Feld als Aufgabenvorlagenparameter angeben. Geben Sie für jede der Parametervariablen der Aufgabenvorlage ihren Datentyp (`STRING` oder `NUMBER`) und optional einen Standardwert an. Das folgende Beispiel zeigt, wie Sie Aufgabenvorlagenparameter für Einstiegspunktposition, Hauptklasse und S3-Protokollspeicherort angeben können.

Um den Standort des Einstiegspunkts, die Hauptklasse und den Speicherort des Amazon-S3-Protokolls als Parameter für die Aufgabenvorlage anzugeben

1. Erstellen Sie eine `create-job-template-request.json`-Datei und geben Sie die erforderlichen Parameter für Ihre Aufgabenvorlage an, wie in der folgenden JSON-Beispieldatei gezeigt. Weitere Informationen zu den Parametern finden Sie unter CreateJobTemplate-API.

```
{
   "name": "mytemplate",
   "jobTemplateData": {
        "executionRoleArn": "iam_role_arn_for_job_execution",
        "releaseLabel": "emr-6.7.0-latest",
        "jobDriver": {
            "sparkSubmitJobDriver": {
                "entryPoint": "${EntryPointLocation}",
                "entryPointArguments": [ "argument1","argument2",...],
                "sparkSubmitParameters": "--class ${MainClass} --conf
 spark.executor.instances=2 --conf spark.executor.memory=2G --conf
 spark.executor.cores=2 --conf spark.driver.cores=1"
            }
        },
        "configurationOverrides": {
            "applicationConfiguration": [
                {
                    "classification": "spark-defaults",
                    "properties": {
                         "spark.driver.memory":"2G"
                    }
                }
            ],
```

```
            "monitoringConfiguration": {
                "persistentAppUI": "ENABLED",
                "cloudWatchMonitoringConfiguration": {
                    "logGroupName": "my_log_group",
                    "logStreamNamePrefix": "log_stream_prefix"
                },
                "s3MonitoringConfiguration": {
                    "logUri": "${LogS3BucketUri}"
                }
            }
        },
        "parameterConfiguration": {
            "EntryPointLocation": {
                "type": "STRING"
            },
            "MainClass": {
                "type": "STRING",
                "defaultValue":"Main"
            },
            "LogS3BucketUri": {
                "type": "STRING",
                "defaultValue":"s3://my_s3_log_location/"
            }
        }
    }
}
```

2. Verwenden Sie den `create-job-template`-Befehl mit einem Pfad zu der lokal gespeicherten `create-job-template-request.json`-Datei in Amazon S3.

```
aws emr-containers create-job-template \
--cli-input-json file://./create-job-template-request.json
```

Um eine Aufgabe zu starten, verwenden Sie eine Aufgabenvorlage mit Aufgabenvorlagenparametern

Um eine Aufgabenausführung mit einer Aufgabenvorlage zu starten, die Aufgabenvorlagenparameter enthält, geben Sie die Aufgabenvorlagen-ID sowie Werte für die Aufgabenvorlagenparameter in der `StartJobRun`-API-Anfrage an, wie unten gezeigt.

```
aws emr-containers start-job-run \
--virtual-cluster-id 123456 \
--name myjob \
```

```
--job-template-id 1234abcd \
--job-template-parameters '{"EntryPointLocation": "entry_point_location","MainClass":
 "ExampleMainClass","LogS3BucketUri": "s3://example_s3_bucket/"}'
```

Steuern des Zugriffs auf Aufgabenvorlagen

Mit der `StartJobRun`-Richtlinie können Sie erzwingen, dass ein Benutzer oder eine Rolle nur Aufträge mit von Ihnen angegebenen Aufgabenvorlagen ausführen kann und keine `StartJobRun`-Operationen ausführen kann, ohne die angegebenen Aufgabenvorlagen zu verwenden. Um dies zu erreichen, stellen Sie zunächst sicher, dass Sie dem Benutzer oder der Rolle eine Leseberechtigung für die angegebenen Aufgabenvorlagen erteilen, wie unten dargestellt.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "emr-containers:DescribeJobTemplate",
            "Resource": [
                "job_template_1_arn",
                "job_template_2_arn",
                ...
            ]
        }
    ]
}
```

Um sicherzustellen, dass ein Benutzer oder eine Rolle nur dann einen `StartJobRun`-Vorgang aufrufen kann, wenn er bestimmte Aufgabenvorlagen verwendet, können Sie einem bestimmten Benutzer oder einer bestimmten Rolle die folgende `StartJobRun`-Richtlinienberechtigung zuweisen.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "emr-containers:StartJobRun",
            "Resource": [
                "virtual_cluster_arn",
            ],
            "Condition": [
```

```
                "StringEquals": {
                    "emr-containers:JobTemplateArn": [
                        "job_template_1_arn",
                        "job_template_2_arn",
                        ...
                    ]
                }
            }
        }
    ]
}
```

Wenn die Aufgabenvorlage einen Aufgabenvorlagenparameter innerhalb des ARN-Felds der Ausführungsrolle angibt, kann der Benutzer einen Wert für diesen Parameter angeben und somit `StartJobRun` mit einer beliebigen Ausführungsrolle aufrufen. Informationen zur Einschränkung der Ausführungsrollen, die der Benutzer bereitstellen kann, finden Sie unter Steuern des Zugriffs auf die Ausführungsrolle in Auftragausführungsrollen mit Amazon EMR in EKS verwenden.

Wenn in der obigen `StartJobRun`-Aktionsrichtlinie für einen bestimmten Benutzer oder eine Rolle keine Bedingung angegeben ist, darf der Benutzer oder die Rolle mithilfe einer beliebigen Aufgabenvorlage, auf die er Lesezugriff hat, oder mithilfe einer beliebigen Ausführungsrolle `StartJobRun`-Aktionen auf dem angegebenen virtuellen Cluster aufrufen.

Verwenden von Pod-Vorlagen

Ab den Amazon-EMR-Versionen 5.33.0 oder 6.3.0 unterstützt Amazon EMR in EKS das Pod-Vorlagenfeature von Spark. Ein Pod ist eine Gruppe von einem oder mehreren Containern mit gemeinsam genutzten Speicher- und Netzwerkressourcen und einer Spezifikation für die Ausführung der Container. Pod-Vorlagen sind Spezifikationen, die bestimmen, wie jeder Pod ausgeführt wird. Sie können Pod-Vorlagendateien verwenden, um die Treiber- oder Executor-Pod-Konfigurationen zu definieren, die Spark-Konfigurationen nicht unterstützen. Weitere Informationen zum Pod-Vorlagenfeature von Spark finden Sie unter Pod-Vorlage.

> **Note**
>
> Das Pod-Vorlagenfeature funktioniert nur mit Treiber- und Executor-Pods. Sie können Auftrag-Controller-Pods nicht mithilfe der Pod-Vorlage konfigurieren.

Gängige Szenarien

Sie können definieren, wie Spark-Aufträge auf gemeinsam genutzten EKS-Clustern ausgeführt werden, indem Sie Pod-Vorlagen mit Amazon EMR in EKS verwenden. So können Sie Kosten sparen und die Ressourcennutzung und Leistung verbessern.

- Um die Kosten zu senken, können Sie Spark-Treiberaufgaben so planen, dass sie auf On-Demand-Instances von Amazon EC2 ausgeführt werden, und gleichzeitig Spark-Executor-Aufgaben für die Ausführung auf Amazon-EC2-Spot Instances planen.
- Um die Ressourcennutzung zu erhöhen, können Sie mehrere Teams dabei unterstützen, ihre Workloads auf demselben EKS-Cluster auszuführen. Jedes Team erhält eine bestimmte Amazon-EC2-Knotengruppe, auf der es seine Workloads ausführen kann. Sie können Pod-Vorlagen verwenden, um eine entsprechende Toleranz auf ihren Workload anzuwenden.
- Um die Überwachung zu verbessern, können Sie einen separaten Protokollierungs-Container ausführen, um Protokolle an Ihre bestehende Überwachungsanwendung weiterzuleiten.

Die folgende Pod-Vorlagendatei veranschaulicht beispielsweise ein gängiges Nutzungsszenario.

```
apiVersion: v1
kind: Pod
spec:
  volumes:
    - name: source-data-volume
      emptyDir: {}
    - name: metrics-files-volume
      emptyDir: {}
  nodeSelector:
    eks.amazonaws.com/nodegroup: emr-containers-nodegroup
  containers:
  - name: spark-kubernetes-driver # This will be interpreted as driver Spark main
 container
    env:
      - name: RANDOM
        value: "random"
    volumeMounts:
      - name: shared-volume
        mountPath: /var/data
      - name: metrics-files-volume
        mountPath: /var/metrics/data
  - name: custom-side-car-container # Sidecar container
```

```
    image: <side_car_container_image>
    env:
      - name: RANDOM_SIDECAR
        value: random
    volumeMounts:
      - name: metrics-files-volume
        mountPath: /var/metrics/data
    command:
      - /bin/sh
      - '-c'
      -  <command-to-upload-metrics-files>
  initContainers:
  - name: spark-init-container-driver # Init container
    image: <spark-pre-step-image>
    volumeMounts:
      - name: source-data-volume # Use EMR predefined volumes
        mountPath: /var/data
    command:
      - /bin/sh
      - '-c'
      -  <command-to-download-dependency-jars>
```

Die Pod-Vorlage führt die folgenden Aufgaben aus:

- Fügen Sie einen neuen Init-Container hinzu, der ausgeführt wird, bevor der Spark-Hauptcontainer gestartet wird. Der Init-Container teilt sich das aufgerufene EmptyDir-Volume namens `source-data-volume` mit dem Spark-Hauptcontainer. Sie können Ihren Init-Container Initialisierungsschritte wie das Herunterladen von Abhängigkeiten oder das Generieren von Eingabedaten ausführen lassen. Dann verbraucht der Spark-Hauptcontainer die Daten.
- Fügen Sie einen weiteren Sidecar-Container hinzu, der zusammen mit dem Spark-Hauptcontainer ausgeführt wird. Die beiden Container teilen sich ein weiteres `EmptyDir`-Volume namens `metrics-files-volume`. Ihr Spark-Auftrag kann Metriken wie Prometheus-Metriken generieren. Dann kann der Spark-Auftrag die Metriken in eine Datei schreiben und den Sidecar-Container die Dateien zur zukünfigen Analyse in Ihr eigenes BI-System hochladen lassen.
- Fügen Sie dem Spark-Hauptcontainer eine neue Umgebungsvariable hinzu. Sie können festlegen, dass Ihr Auftrag die Umgebungsvariable verwendet.
- Definieren Sie eine Knotenauswahl, sodass der Pod nur für die `emr-containers-nodegroup`-Knotengruppe geplant ist. Dies hilft dabei, Rechenressourcen für verschiedene Aufträge und Teams zu isolieren.

Aktivieren von Pod-Vorlagen mit Amazon EMR in EKS

Um das Pod-Vorlagenfeature mit Amazon EMR in EKS zu aktivieren, konfigurieren Sie die Spark-Eigenschaften `spark.kubernetes.driver.podTemplateFile` und `spark.kubernetes.executor.podTemplateFile` verweisen Sie auf die Pod-Vorlagendateien in Amazon S3. Spark lädt dann die Pod-Vorlagendatei herunter und verwendet sie, um Treiber- und Executor-Pods zu erstellen.

> **Note**
>
> Spark verwendet die Auftragausführungsrolle, um die Pod-Vorlage zu laden, sodass die Auftragausführungsrolle über Zugriffsberechtigungen für Amazon S3 verfügen muss, um die Pod-Vorlagen zu laden. Weitere Informationen finden Sie unter Erstellen einer Aufgabenausführungsrolle.

Sie können `SparkSubmitParameters` verwenden, um den Amazon-S3-Pfad zur Pod-Vorlage anzugeben, wie die folgende JSON-Datei mit der Auftragausführung zeigt.

```
{
  "name": "myjob",
  "virtualClusterId": "123456",
  "executionRoleArn": "iam_role_name_for_job_execution",
  "releaseLabel": "release_label",
  "jobDriver": {
    "sparkSubmitJobDriver": {
      "entryPoint": "entryPoint_location",
      "entryPointArguments": ["argument1", "argument2", ...],
       "sparkSubmitParameters": "--class <main_class> \
         --conf
 spark.kubernetes.driver.podTemplateFile=s3://path_to_driver_pod_template \
         --conf
 spark.kubernetes.executor.podTemplateFile=s3://path_to_executor_pod_template \
         --conf spark.executor.instances=2 \
         --conf spark.executor.memory=2G \
         --conf spark.executor.cores=2 \
         --conf spark.driver.cores=1"
    }
  }
}
```

Alternativ können Sie den `configurationOverrides` verwenden, um den Amazon-S3-Pfad zur Pod-Vorlage anzugeben, wie die folgende JSON-Datei mit der Auftragausführung zeigt.

```
{
  "name": "myjob",
  "virtualClusterId": "123456",
  "executionRoleArn": "iam_role_name_for_job_execution",
  "releaseLabel": "release_label",
  "jobDriver": {
    "sparkSubmitJobDriver": {
      "entryPoint": "entryPoint_location",
      "entryPointArguments": ["argument1", "argument2", ...],
       "sparkSubmitParameters": "--class <main_class> \
         --conf spark.executor.instances=2 \
         --conf spark.executor.memory=2G \
         --conf spark.executor.cores=2 \
         --conf spark.driver.cores=1"
    }
  },
  "configurationOverrides": {
    "applicationConfiguration": [
      {
        "classification": "spark-defaults",
        "properties": {
          "spark.driver.memory":"2G",
          "spark.kubernetes.driver.podTemplateFile":"s3://path_to_driver_pod_template",

 "spark.kubernetes.executor.podTemplateFile":"s3://path_to_executor_pod_template"
         }
      }
    ]
  }
}
```

> **Note**
>
> 1. Sie müssen die Sicherheitsrichtlinien befolgen, wenn Sie das Pod-Vorlagenfeature mit Amazon EMR in EKS verwenden, z. B. das Isolieren von nicht vertrauenswürdigem Anwendungscode. Weitere Informationen finden Sie unter Bewährte Methoden für Sicherheit in Amazon EMR in EKS.

2. Sie können die Namen der Spark-Hauptcontainer nicht mit `spark.kubernetes.driver.podTemplateContainerName` und `spark.kubernetes.executor.podTemplateContainerName` ändern, da diese Namen als `spark-kubernetes-driver` und `spark-kubernetes-executors` fest codiert sind. Wenn Sie den Spark-Hauptcontainer anpassen möchten, müssen Sie den Container in einer Pod-Vorlage mit diesen hartcodierten Namen angeben.

Pod-Vorlagenfelder

Beachten Sie die folgenden Feldeinschränkungen, wenn Sie eine Pod-Vorlage mit Amazon EMR in EKS konfigurieren.

- Amazon EMR in EKS erlaubt nur die folgenden Felder in einer Pod-Vorlage, um eine korrekte Aufgabenplanung zu ermöglichen.

  Dies sind die zulässigen Felder auf Pod-Ebene:

  - `apiVersion`
  - `kind`
  - `metadata`
  - `spec.activeDeadlineSeconds`
  - `spec.affinity`
  - `spec.containers`
  - `spec.enableServiceLinks`
  - `spec.ephemeralContainers`
  - `spec.hostAliases`
  - `spec.hostname`
  - `spec.imagePullSecrets`
  - `spec.initContainers`
  - `spec.nodeName`
  - `spec.nodeSelector`
  - `spec.overhead`
  - `spec.preemptionPolicy`
  - `spec.priority`

- spec.priorityClassName
- spec.readinessGates
- spec.runtimeClassName
- spec.schedulerName
- spec.subdomain
- spec.terminationGracePeriodSeconds
- spec.tolerations
- spec.topologySpreadConstraints
- spec.volumes

Dies sind die zulässigen Felder auf Spark-Hauptcontainerebene:

- env
- envFrom
- name
- lifecycle
- livenessProbe
- readinessProbe
- resources
- startupProbe
- stdin
- stdinOnce
- terminationMessagePath
- terminationMessagePolicy
- tty
- volumeDevices
- volumeMounts
- workingDir

Wenn Sie unzulässige Felder in der Pod-Vorlage verwenden, löst Spark eine Ausnahme aus und der Auftrag schlägt fehl. Das folgende Beispiel zeigt eine Fehlermeldung im Spark-Controller-Protokoll, die auf unzulässige Felder zurückzuführen ist.

```
Executor pod template validation failed.
Field container.command in Spark main container not allowed but specified.
```

- Amazon EMR in EKS definiert die folgenden Parameter in einer Pod-Vorlage vordefiniert. Die Felder, die Sie in einer Pod-Vorlage angeben, dürfen sich nicht mit diesen Feldern überschneiden.

  Dies sind die vordefinierten Volume-Namen:

  - `emr-container-communicate`
  - `config-volume`
  - `emr-container-application-log-dir`
  - `emr-container-event-log-dir`
  - `temp-data-dir`
  - `mnt-dir`
  - `home-dir`
  - `emr-container-s3`

  Dies sind die vordefinierten Volume-Mounts, die nur für den Spark-Hauptcontainer gelten:

  - Name: `emr-container-communicate`; MountPath: `/var/log/fluentd`
  - Name: `emr-container-application-log-dir`; MountPath: `/var/log/spark/user`
  - Name: `emr-container-event-log-dir`; MountPath: `/var/log/spark/apps`
  - Name: `mnt-dir`; MountPath: `/mnt`
  - Name: `temp-data-dir`; MountPath: `/tmp`
  - Name: `home-dir`; MountPath: `/home/hadoop`

  Dies sind die vordefinierten Umgebungsvariablen, die nur für den Spark-Hauptcontainer gelten:

  - `SPARK_CONTAINER_ID`
  - `K8S_SPARK_LOG_URL_STDERR`
  - `K8S_SPARK_LOG_URL_STDOUT`
  - `SIDECAR_SIGNAL_FILE`

  > **Note**
  >
  > Sie können diese vordefinierten Volumes weiterhin verwenden und sie in Ihre zusätzlichen Sidecar-Container einbinden. Sie können beispielsweise `emr-container-`

`application-log-dir` verwenden und in Ihrem eigenen Sidecar-Container bereitstellen, der in der Pod-Vorlage definiert ist.

Wenn die von Ihnen angegebenen Felder mit einem der vordefinierten Felder in der Pod-Vorlage in Konflikt stehen, löst Spark eine Ausnahme aus und der Auftrag schlägt fehl. Das folgende Beispiel zeigt eine Fehlermeldung im Spark-Anwendungsprotokoll aufgrund von Konflikten mit den vordefinierten Feldern.

```
Defined volume mount path on main container must not overlap with reserved mount
 paths: [<reserved-paths>]
```

Überlegungen zu Sidecarcontainern

Amazon EMR steuert den Lebenszyklus der Pods, die von Amazon EMR in EKS bereitgestellt werden. Die Sidecar-Container sollten dem gleichen Lebenszyklus folgen wie der Spark-Hauptcontainer. Wenn Sie zusätzliche Sidecar-Container in Ihre Pods einbauen, empfehlen wir Ihnen, das Pod-Lifecycle-Management, das Amazon EMR definiert, zu integrieren, sodass sich der Sidecar-Container selbst stoppen kann, wenn der Spark-Hauptcontainer verlassen wird.

Um die Kosten zu senken, empfehlen wir Ihnen, einen Prozess zu implementieren, der verhindert, dass Treiber-Pods mit Sidecar-Containern nach Abschluss Ihres Aufträge weiter ausgeführt werden. Der Spark-Treiber löscht Ausführer-Pods, wenn der Ausführer fertig ist. Wenn ein Treiberprogramm abgeschlossen ist, werden die zusätzlichen Sidecar-Container jedoch weiter ausgeführt. Der Pod wird in Rechnung gestellt, bis Amazon EMR in EKS den Treiber-Pod bereinigt hat, normalerweise weniger als eine Minute, nachdem der Treiber-Spark-Hauptcontainer abgeschlossen ist. Um die Kosten zu senken, können Sie Ihre zusätzlichen Sidecar-Container in den Lebenszyklus-Management-Mechanismus integrieren, den Amazon EMR in EKS sowohl für Treiber- als auch für Ausführer-Pods definiert, wie im folgenden Abschnitt beschrieben.

Der Spark-Hauptcontainer in den Treiber- und Ausführer-Pods sendet `heartbeat` alle zwei Sekunden an eine `/var/log/fluentd/main-container-terminated`-Datei. Indem Sie den vordefinierten `emr-container-communicate`-Amazon-EMR-Volume-Mount zu Ihrem Sidecar-Container hinzufügen, können Sie einen Unterprozess Ihres Sidecar-Containers definieren, der regelmäßig den Zeitpunkt der letzten Änderung für diese Datei verfolgt. Der Unterprozess stoppt sich dann selbst, wenn er feststellt, dass der Spark-Hauptcontainer den `heartbeat` für einen längeren Zeitraum stoppt.

Das folgende Beispiel zeigt einen Unterprozess, der die Heartbeat-Datei verfolgt und sich selbst stoppt. Ersetzen Sie *your_volume_mount* durch den Pfad, in dem Sie das vordefinierte Volume mounten. Das Skript ist in dem Image gebündelt, das vom Sidecar-Container verwendet wird. In einer Pod-Vorlagendatei können Sie einen Sidecar-Container mit den folgenden Befehlen `sub_process_script.sh` und `main_command` angeben.

```
MOUNT_PATH="your_volume_mount"
FILE_TO_WATCH="$MOUNT_PATH/main-container-terminated"
INITIAL_HEARTBEAT_TIMEOUT_THRESHOLD=60
HEARTBEAT_TIMEOUT_THRESHOLD=15
SLEEP_DURATION=10

function terminate_main_process() {
  # Stop main process
}

# Waiting for the first heartbeat sent by Spark main container
echo "Waiting for file $FILE_TO_WATCH to appear..."
start_wait=$(date +%s)
while ! [[ -f "$FILE_TO_WATCH" ]]; do
    elapsed_wait=$(expr $(date +%s) - $start_wait)
    if [ "$elapsed_wait" -gt "$INITIAL_HEARTBEAT_TIMEOUT_THRESHOLD" ]; then
        echo "File $FILE_TO_WATCH not found after $INITIAL_HEARTBEAT_TIMEOUT_THRESHOLD
 seconds; aborting"
        terminate_main_process
        exit 1
    fi
    sleep $SLEEP_DURATION;
done;
echo "Found file $FILE_TO_WATCH; watching for heartbeats..."

while [[ -f "$FILE_TO_WATCH" ]]; do
    LAST_HEARTBEAT=$(stat -c %Y $FILE_TO_WATCH)
    ELAPSED_TIME_SINCE_AFTER_HEARTBEAT=$(expr $(date +%s) - $LAST_HEARTBEAT)
    if [ "$ELAPSED_TIME_SINCE_AFTER_HEARTBEAT" -gt "$HEARTBEAT_TIMEOUT_THRESHOLD" ];
 then
        echo "Last heartbeat to file $FILE_TO_WATCH was more than
 $HEARTBEAT_TIMEOUT_THRESHOLD seconds ago at $LAST_HEARTBEAT; terminating"
        terminate_main_process
        exit 0
    fi
    sleep $SLEEP_DURATION;
done;
```

```
echo "Outside of loop, main-container-terminated file no longer exists"

# The file will be deleted once the fluentd container is terminated

echo "The file $FILE_TO_WATCH doesn't exist any more;"
terminate_main_process
exit 0
```

Verwenden von Richtlinien für die Wiederholung von Aufträgen

In Amazon EMR in EKS-Versionen 6.9.0 und höher können Sie eine Wiederholungsrichtlinie für Ihre Aufgabenausführungen festlegen. Richtlinien für Wiederholungen bewirken, dass ein Auftrags-Treiber-Pod automatisch neu gestartet wird, wenn er fehlschlägt oder gelöscht wird. Dies macht Spark-Streaming-Aufträge mit langer Laufzeit widerstandsfähiger gegenüber Ausfällen.

Festlegen einer Wiederholungsrichtlinie für einen Auftrag

Um eine Wiederholungsrichtlinie zu konfigurieren, stellen Sie mithilfe der StartJobRun-API ein `RetryPolicyConfiguration`-Feld bereit. Eine Beispiel-`retryPolicyConfiguration` wird hier gezeigt:

```
aws emr-containers start-job-run \
--virtual-cluster-id cluster_id \
--name sample-job-name \
--execution-role-arn execution-role-arn \
--release-label emr-6.9.0-latest \
--job-driver '{
  "sparkSubmitJobDriver": {
    "entryPoint": "local:///usr/lib/spark/examples/src/main/python/pi.py",
    "entryPointArguments": [ "2" ],
    "sparkSubmitParameters": "--conf spark.executor.instances=2 --conf
 spark.executor.memory=2G --conf spark.executor.cores=2 --conf spark.driver.cores=1"
  }
}' \
--retry-policy-configuration '{
    "maxAttempts": 5
  }' \
--configuration-overrides '{
  "monitoringConfiguration": {
    "cloudWatchMonitoringConfiguration": {
      "logGroupName": "my_log_group_name",
```

```
      "logStreamNamePrefix": "my_log_stream_prefix"
    },
    "s3MonitoringConfiguration": {
       "logUri": "s3://DOC-EXAMPLE-BUCKET-LOGGING"
    }
  }
}'
```

> **Note**
>
> `retryPolicyConfiguration` ist erst ab Version AWS CLI 1.27.68 verfügbar. Informationen zum Aktualisieren von AWS CLI auf die neueste Version finden Sie unter Installieren oder Aktualisieren der neuesten Version der AWS CLI

Konfigurieren Sie das `maxAttempts`-Feld so, wie oft der Aufgaben-Treiber-Pod maximal neu gestartet werden soll, wenn er ausfällt oder gelöscht wird. Das Ausführungsintervall zwischen zwei Wiederholungsversuchen des Auftrag-Treibers ist ein exponentielles Wiederholungsintervall von (10 Sekunden, 20 Sekunden, 40 Sekunden …), das auf 6 Minuten begrenzt ist, wie in der Kubernetes-Dokumentation beschrieben.

> **Note**
>
> Jede weitere Ausführung des Aufgabentreibers wird als weitere Aufgabenausführung in Rechnung gestellt und unterliegt den Preisen von Amazon EMR in EKS.

Wiederholungsrichtlinien-Konfigurationswerte

- Standard-Wiederholungsrichtlinie für einen Auftrag: `StartJobRun` beinhaltet eine Wiederholungsrichtlinie, die standardmäßig auf einen maximalen Versuch festgelegt ist. Sie können die Wiederholungs-Richtlinie wie gewünscht konfigurieren.

  > **Note**
  >
  > Wenn `maxAttempts` der `retryPolicyConfiguration` auf 1 gesetzt ist, bedeutet dies, dass bei einem Fehler keine erneuten Versuche unternommen werden, den Treiber-Pod aufzurufen.

- Deaktivieren der Wiederholungsrichtlinie für einen Auftrag: Um eine Wiederholungsrichtlinie zu deaktivieren, legen Sie den Wert für maximale Versuche in RetryPolicyConfiguration auf 1 fest.

```
"retryPolicyConfiguration": {
    "maxAttempts": 1
}
```

- Setzen Sie maxAttempts für einen Auftrag innerhalb des gültigen Bereichs: Der `StartJobRun`-Aufruf schlägt fehl, wenn der `maxAttempts`-Wert außerhalb des gültigen Bereichs liegt. Der gültige `maxAttempts`-Bereich liegt zwischen 1 und 2.147.483.647 (32-Bit-Ganzzahl), dem Bereich, der für die `backOffLimit`-Konfigurationseinstellung von Kubernetes unterstützt wird. Weitere Informationen finden Sie unter Pod-Backoff-Fehlerrichtlinie in der Kubernetes-Dokumentation. Falls der `maxAttempts`-Wert ungültig ist, wird folgende Fehlermeldung zurückgegeben:

```
{
 "message": "Retry policy configuration's parameter value of maxAttempts is invalid"
}
```

Abrufen eines Status einer Wiederholungsrichtlinie für einen Auftrag

Sie können den Status der Wiederholungsversuche für einen Auftrag mit den `ListJobRuns`- und `DescribeJobRun`-APIs anzeigen. Sobald Sie einen Auftrag mit aktivierter Konfiguration der Wiederholungsrichtlinie anfordern, enthalten die `ListJobRun`- und `DescribeJobRun`-Antworten und den Status der Wiederholungsrichtlinie im `RetryPolicyExecution`-Feld. Darüber hinaus enthält die `DescribeJobRun`-Antwort die Angaben `RetryPolicyConfiguration`, die in der `StartJobRun`-Anfrage für den Auftrag eingegeben wurden.

Beispielantworten

ListJobRuns response

```
{
  "jobRuns": [
    ...
    ...
    "retryPolicyExecution" : {
      "currentAttemptCount": 2
    }
```

```
    ...
    ...
  ]
}
```

DescribeJobRun response

```
{
  ...
  ...
  "retryPolicyConfiguration": {
    "maxAttempts": 5
   },
   "retryPolicyExecution" : {
    "currentAttemptCount": 2
  },
  ...
  ...
}
```

Diese Felder sind nicht sichtbar, wenn die Wiederholungsrichtlinie für den Auftrag deaktiviert ist, wie weiter unten in Wiederholungsrichtlinien-Konfigurationswerte beschrieben.

Überwachen eines Auftrags mit einer Wiederholungsrichtlinie

Wenn Sie eine Wiederholungsrichtlinie aktivieren, wird für jeden erstellten Auftragstreiber ein CloudWatch-Ereignis generiert. Um diese Ereignisse zu abonnieren, richten Sie mit dem folgenden Befehl eine CloudWatch-Ereignisregel ein:

```
aws events put-rule \
--name cwe-test \
--event-pattern '{"detail-type": ["EMR Job Run New Driver Attempt"]}'
```

Das Ereignis gibt Informationen zu den Auftragstreibern `newDriverPodName`, `newDriverCreatedAt` dem `previousDriverFailureMessage`-Zeitstempel und den `currentAttemptCount`-Auftragstreibern zurück. Diese Ereignisse werden nicht erzeugt, wenn die Wiederholungsrichtlinie deaktiviert ist.

Weitere Informationen zum Überwachen Ihrer Aufgabe mit CloudWatch-Ereignissen finden Sie unter Überwachen von Aufträgen mit Amazon CloudWatch Events.

Suchen nach Protokollen für Treiber und Ausführer

Die Namen der Treiber-Pods folgen dem Format `spark-<job id>-driver-<random-suffix>`. Dasselbe `random-suffix` wird zu den Namen der Ausführer-Pods hinzugefügt, die der Treiber erzeugt. Wenn Sie `random-suffix` verwenden, können Sie Protokolle für einen Treiber und die zugehörigen Ausführer finden. Der `random-suffix` ist nur vorhanden, wenn die Wiederholungsrichtlinie für den Auftrag aktiviert ist; andernfalls fehlt `random-suffix`.

Weitere Informationen zur Konfiguration von Aufträgen mit Überwachungskonfiguration für die Protokollierung finden Sie unter Eine Spark-Anwendung ausführen.

Verwenden der Rotation des Spark-Ereignisprotokolls

Mit Amazon EMR 6.3.0 und höher können Sie das Feature zur Rotation des Spark-Ereignisprotokolls für Amazon EMR in EKS aktivieren. Anstatt eine einzelne Ereignisprotokolldatei zu generieren, rotiert dieses Feature die Datei auf der Grundlage Ihres konfigurierten Zeitintervalls und entfernt die ältesten Ereignisprotokolldateien.

Durch das Rotieren von Spark-Ereignisprotokollen können Sie potenzielle Probleme mit einer großen Spark-Ereignisprotokolldatei vermeiden, die für Aufträge mit langer Laufzeit oder Streaming-Aufträge generiert wird. Sie starten beispielsweise einen Spark-Auftrag mit langer Laufzeit und einem mit dem `persistentAppUI`-Parameter aktivierten Ereignisprotokoll. Der Spark-Treiber generiert eine Ereignisprotokolldatei. Wenn der Auftrag stunden- oder tagelang ausgeführt wird und der Speicherplatz auf dem Kubernetes-Knoten begrenzt ist, kann die Ereignisprotokolldatei den gesamten verfügbaren Speicherplatz beanspruchen. Das Aktivieren des Spark-Features zur Rotation des Ereignisprotokolls löst das Problem, indem die Protokolldatei in mehrere Dateien aufgeteilt und die ältesten Dateien entfernt werden.

> **Note**
>
> Dieses Feature funktioniert nur mit Amazon EMR in EKS. Amazon EMR, das in Amazon EC2 ausgeführt wird, unterstützt die Rotation des Spark-Ereignisprotokolls nicht.

Um das Feature zur Rotation des Spark-Ereignisprotokolls zu aktivieren, konfigurieren Sie die folgenden Spark-Parameter:

- `spark.eventLog.rotation.enabled` – aktiviert die Protokoll-Rotation. Standardmäßig ist die Rückverfolgungsverwaltung in der Spark-Konfigurationsdatei deaktiviert. Stellen Sie diese auf true ein, um dieses Feature zu aktivieren.
- `spark.eventLog.rotation.interval` – gibt das Zeitintervall für die Protokollrotation an. Der Mindestwert beträgt 60 Sekunden. Der Standardwert beträgt 300 Sekunden.
- `spark.eventLog.rotation.minFileSize` – gibt eine Mindestdateigröße für die Rotation der Protokolldatei an. Der Mindest- und Standardwert beträgt 1 MB.
- `spark.eventLog.rotation.maxFilesToRetain` – gibt an, wie viele rotierte Protokolldateien während der Bereinigung beibehalten werden sollen. Der gültige Bereich ist 1 bis 10. Der Standardwert lautet 2.

Sie können diese Parameter im `sparkSubmitParameters`-Abschnitt der `StartJobRun`-API angeben, wie das folgende Beispiel zeigt.

```
"sparkSubmitParameters": "--class org.apache.spark.examples.SparkPi --conf
 spark.eventLog.rotation.enabled=true --conf spark.eventLog.rotation.interval=300 --
conf spark.eventLog.rotation.minFileSize=1m --conf
 spark.eventLog.rotation.maxFilesToRetain=2"
```

Verwenden der Spark-Container-Protokoll-Rotation

Mit Amazon EMR 6.11.0 und höher können Sie das Spark-Container-Protokoll-Rotationsfeature unktion für Amazon EMR in EKS aktivieren. Anstatt eine einzelne `stdout`- oder `stderr`-Protokolldatei zu generieren, rotiert dieses Feature die Datei auf der Grundlage Ihrer konfigurierten Rotationsgröße und entfernt die ältesten Protokolldateien aus dem Container.

Durch das Rotieren von Spark-Container-Protokolle können Sie potenzielle Probleme mit großen Spark-Protokolldateien vermeiden, die für lang laufende Aufträge oder Streaming-Aufträge generiert wurden. Sie könnten beispielsweise einen Spark-Auftrag mit langer Laufzeit starten und der Spark-Treiber generiert eine Container-Protokolldatei. Wenn der Auftrag stunden- oder tagelang ausgeführt wird und der Speicherplatz auf dem Kubernetes-Knoten begrenzt ist, kann die Container-Protokolldatei den gesamten verfügbaren Festplattenspeicher beanspruchen. Wenn Sie die Spark-Container-Protokollrotation aktivieren, teilen Sie die Protokolldatei in mehrere Dateien auf und entfernen die ältesten Dateien.

Um das Feature zur Rotation von Spark-Container-Protokollen zu aktivieren, konfigurieren Sie die folgenden Spark-Parameter:

`containerLogRotationConfiguration`

Fügen Sie diesen Parameter `monitoringConfiguration` hinzu, um die Protokoll-Rotation zu aktivieren. Standardmäßig ist die Rückverfolgung deaktiviert. Sie müssen `containerLogRotationConfiguration` zusätzlich zu `s3MonitoringConfiguration` verwenden.

`rotationSize`

Der `rotationSize`-Parameter gibt die Dateigröße für die Protokollrotation an. Der Bereich der möglichen Werte ist von `2KB` bis `2GB`. Die numerische Einheit des `rotationSize`-Parameters wird als Ganzzahl übergeben. Da Dezimalwerte nicht unterstützt werden, können Sie mit dem Wert `1500MB` beispielsweise eine Rotationsgröße von 1,5 GB angeben.

`maxFilesToKeep`

Der `maxFilesToKeep`-Parameter gibt die maximale Anzahl von Dateien an, die nach der Rotation im Container aufbewahrt werden sollen. Der kleinste Wert ist 1 und der größte Wert ist 50.

Sie können diese Parameter im `monitoringConfiguration`-Abschnitt der `StartJobRun`-API angeben, wie das folgende Beispiel zeigt. In diesem Beispiel rotiert Amazon EMR in EKS Ihre Protokolle mit `rotationSize = "10 MB"` und `maxFilesToKeep = 3` um 10 MB, generiert eine neue Protokolldatei und löscht dann die älteste Protokolldatei, sobald die Anzahl der Protokolldateien 3 erreicht.

```
{
  "name": "my-long-running-job",
  "virtualClusterId": "123456",
  "executionRoleArn": "iam_role_name_for_job_execution",
  "releaseLabel": "emr-6.11.0-latest",
  "jobDriver": {
    "sparkSubmitJobDriver": {
      "entryPoint": "entryPoint_location",
      "entryPointArguments": ["argument1", "argument2", ...],
       "sparkSubmitParameters": "--class main_class --conf spark.executor.instances=2
 --conf spark.executor.memory=2G --conf spark.executor.cores=2 --conf
 spark.driver.cores=1"
    }
  },
  "configurationOverrides": {
    "applicationConfiguration": [
```

```
      {
        "classification": "spark-defaults",
        "properties": {
          "spark.driver.memory":"2G"
         }
      }
    ],
    "monitoringConfiguration": {
      "persistentAppUI": "ENABLED",
      "cloudWatchMonitoringConfiguration": {
        "logGroupName": "my_log_group",
        "logStreamNamePrefix": "log_stream_prefix"
      },
      "s3MonitoringConfiguration": {
        "logUri": "s3://my_s3_log_location"
      },
      "containerLogRotationConfiguration": {
        "rotationSize":"10MB",
        "maxFilesToKeep":"3"
      }
    }
  }
}
```

Um eine Auftragsausführung mit der Spark-Container-Protokollrotation zu starten, geben Sie im Befehl einen Pfad zu der JSON-Datei an, die Sie mit diesen Parametern in `StartJobRun` konfiguriert haben.

```
aws emr-containers start-job-run \
--cli-input-json file://path-to-json-request-file
```

Verwenden von vertikalem Auto Scaling mit Amazon-EMR-Spark-Aufträgen

Das vertikale Auto Scaling von Amazon EMR in EKS passt die Speicher- und CPU-Ressourcen automatisch an die Anforderungen des Workloads an, die Sie für Amazon-EMR-Spark-Anwendungen bereitstellen. Dies vereinfacht das Ressourcenmanagement.

Vertikales Auto Scaling nutzt den Kubernetes Vertical Pod Autoscaler (VPA), um die Echtzeit- und historische Ressourcennutzung Ihrer Amazon-EMR-Spark-Anwendungen zu verfolgen. Die Funktion

des vertikalen Auto Scalings verwendet die von VPA gesammelten Daten, um die Speicher- und CPU-Ressourcen, die Ihren Spark-Anwendungen zugewiesen sind, automatisch zu optimieren. Dieser vereinfachte Prozess erhöht die Zuverlässigkeit und optimiert die Kosten.

Themen

Einrichten des vertikalen Auto Scalings für Amazon EMR in EKS

Dieses Thema hilft Ihnen dabei, Ihren Amazon-EKS-Cluster für die Übermittlung von Amazon-EMR-Spark-Aufträgen mit vertikalem Auto Scaling vorzubereiten. Für den Einrichtungsvorgang müssen Sie die Aufgaben in den folgenden Abschnitten bestätigen oder abschließen:

Themen

Voraussetzungen

Führen Sie die folgenden Aufgaben aus, bevor Sie den Kubernetes-Operator für vertikales Auto Scaling auf Ihrem Cluster installieren. Wenn Sie bereits eine der Voraussetzungen erfüllt haben, können Sie diese überspringen und mit der nächsten fortfahren.

- Installieren Sie das AWS CLI – Wenn Sie AWS CLI bereits installiert haben, vergewissern Sie sich, dass Sie über die neueste Version verfügen.
- kubectl installieren – kubectl ist ein Befehlszeilen-Tool, mit dem Sie mit dem API-Server kommunizieren. Sie benötigen kubectl, um Artefakte im Zusammenhang mit vertikalem Auto Scaling auf Ihrem Amazon-EKS-Cluster zu installieren und zu überwachen.

- Das Operator-SDK installieren – Amazon EMR in EKS verwendet das Operator-SDK als Paketmanager für die gesamte Lebensdauer des vertikalen Auto-Scaling-Operators, den Sie auf Ihrem Cluster installieren.
- Docker installieren – Sie benötigen Zugriff auf die Docker-CLI, um die Docker-Images für vertikales Auto Scaling zu authentifizieren und abzurufen, um sie auf Ihrem Amazon-EKS-Cluster zu installieren.
- Installieren des Kubernetes-Metrikservers – Sie müssen zuerst den Metrikserver installieren, damit der vertikale Pod-Autoscaler Metriken vom Kubernetes-API-Server abrufen kann.
- Einen Amazon-EKS-Cluster einrichten (Version 1.24 oder höher) – Vertikales Auto Scaling wird mit Amazon-EKS-Versionen 1.24 und höher unterstützt. Nachdem Sie den Cluster erstellt haben, registrieren Sie ihn für die Verwendung mit Amazon EMR.
- Eine Amazon-EMR-Basis-Image-URI auswählen (Version 6.10.0 oder höher) – Vertikales Auto Scaling wird mit Amazon-EMR-Versionen 6.10.0 und höher unterstützt.

Den Operator Lifecycle Manager (OLM) auf Ihrem Amazon-EKS-Cluster installieren

Verwenden Sie die Operator-SDK-CLI, um den Operator Lifecycle Manager (OLM) auf dem Amazon EMR in EKS-Cluster zu installieren, auf dem Sie vertikales Auto Scaling einrichten möchten, wie im folgenden Beispiel gezeigt. Sobald Sie es eingerichtet haben, können Sie OLM verwenden, um den Lebenszyklus des vertikalen Auto-Scaling-Operators von Amazon EMR zu installieren und zu verwalten.

```
operator-sdk olm install
```

Führen Sie den folgenden `olm status`-Befehl aus, um die Installation zu validieren:

```
operator-sdk olm status
```

Wenn der Befehl ausgeführt wird, gibt er eine Ausgabe ähnlich wie folgt zurück:

```
INFO[0007] Successfully got OLM status for version X.XX
```

Wenn Ihre Installation nicht erfolgreich ist, lesen Sie Fehlerbehebung von Amazon EMR im vertikalen Auto Scaling von EKS.

Den vertikalen Auto-Scaling-Operator für Amazon EMR in EKS installieren

Verwenden Sie die folgenden Schritte, um den vertikalen Auto-Scaling-Operator auf Ihrem Amazon-EKS-Cluster zu installieren:

1. Richten Sie die folgenden Umgebungsvariablen ein, die Sie verwenden werden, um die Installation abzuschließen:

   - **`$REGION`** verweist auf AWS-Region für Ihren Cluster. Beispiel: `us-west-2`
   - **`$ACCOUNT_ID`** verweist auf die Amazon-ECR-Konto-ID für Ihre Region. Weitere Informationen finden Sie unter Amazon-ECR-Registrierungskonten nach Regionen.
   - **`$RELEASE`** verweist auf die Amazon-EMR-Version, die Sie für Ihren Cluster verwenden möchten. Beim vertikalen Auto Scaling müssen Sie Amazon-EMR-Version 6.10.0 oder höher verwenden.

2. Als Nächstes müssen Sie Authentifizierungstoken für den Operator in der Amazon-ECR-Registrierung abrufen.

   ```
   aws ecr get-login-password \
    --region region-id | docker login \
    --username AWS \
    --password-stdin $ACCOUNT_ID.dkr.ecr.region-id.amazonaws.com
   ```

3. Installieren Sie den vertikalen Auto-Scaling-Operator von Amazon EMR in EKS mit dem folgenden Befehl:

   ```
   ECR_URL=$ACCOUNT_ID.dkr.ecr.$REGION.amazonaws.com && \
   REPO_DEST=dynamic-sizing-k8s-operator-olm-bundle && \
   BUNDLE_IMG=emr-$RELEASE-dynamic-sizing-k8s-operator && \
   operator-sdk run bundle \
   $ECR_URL/$REPO_DEST/$BUNDLE_IMG\:latest
   ```

   Dadurch wird eine Version des vertikalen Auto-Scaling-Operators im Standard-Namespace Ihres Amazon-EKS-Clusters erstellt. Verwenden Sie diesen Befehl, um in einem anderen Namespace zu installieren:

   ```
   operator-sdk run bundle \
   $ACCOUNT_ID.dkr.ecr.$REGION.amazonaws.com/dynamic-sizing-k8s-operator-olm-bundle/
   emr-$RELEASE-dynamic-sizing-k8s-operator:latest \
   -n operator-namespace
   ```

> **Note**
>
> Wenn der von Ihnen angegebene Namespace nicht vorhanden ist, installiert OLM den Operator nicht. Weitere Informationen finden Sie unter Kubernetes-Namespace nicht gefunden.

4. Stellen Sie sicher, dass Sie den Operator erfolgreich mit dem kubectl Kubernetes-Befehlszeilentool installiert haben.

```
kubectl get csv -n operator-namespace
```

Der **`kubectl`**-Befehl sollte Ihren neu bereitgestellten vertikalen Auto-Scaling-Operator mit dem Phasenstatus Erfolgreich zurückgeben. Falls Sie Probleme mit der Installation oder Einrichtung haben, finden Sie weitere Informationen unter Fehlerbehebung von Amazon EMR im vertikalen Auto Scaling von EKS.

Erste Schritte mit dem vertikalen Auto Scaling für Amazon EMR in EKS

Einreichen eines Spark-Auftrags mit vertikalem Auto Scaling

Wenn Sie einen Auftrag über die StartJobRun API einreichen, fügen Sie dem Treiber für Ihren Spark-Auftrag die folgenden beiden Konfigurationen hinzu, um das vertikale Auto Scaling zu aktivieren:

```
"spark.kubernetes.driver.label.emr-containers.amazonaws.com/dynamic.sizing":"true",
"spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/
dynamic.sizing.signature":"YOUR_JOB_SIGNATURE"
```

Im obigen Code aktiviert die erste Zeile die vertikale Auto-Scaling-Funktion. Die nächste Zeile ist eine erforderliche Signaturkonfiguration, mit der Sie eine Signatur für Ihren Auftrag auswählen können.

Weitere Informationen zu diesen Konfigurationen und akzeptablen Parameterwerten finden Sie unter Konfigurieren des vertikalen Auto Scalings für Amazon EMR in EKS. Standardmäßig wird Ihr Auftrag beim vertikalen Auto Scaling im Modus Nur für Überwachung aus gesendet. In diesem Überwachungsstatus können Sie Ressourcenempfehlungen berechnen und anzeigen, ohne ein Auto Scaling durchführen zu müssen. Weitere Informationen finden Sie unter Modi für vertikales Auto Scaling.

Das folgende Beispiel veranschaulicht, wie Sie einen `start-job-run`-Beispiel-Befehl mit vertikalem Auto Scaling abschließen:

```
aws emr-containers start-job-run \
--virtual-cluster-id $VIRTUAL_CLUSTER_ID \
--name $JOB_NAME \
--execution-role-arn $EMR_ROLE_ARN \
--release-label emr-6.10.0-latest \
--job-driver '{
  "sparkSubmitJobDriver": {
     "entryPoint": "local:///usr/lib/spark/examples/src/main/python/pi.py"
   }
 }' \
--configuration-overrides '{
    "applicationConfiguration": [{
        "classification": "spark-defaults",
        "properties": {
          "spark.kubernetes.driver.label.emr-containers.amazonaws.com/dynamic.sizing":
 "true",
          "spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/
dynamic.sizing.signature": "test-signature"
        }
    }]
  }'
```

Überprüfen der vertikalen Auto-Scaling-Funktionalität

Um zu überprüfen, ob das vertikale Auto Scaling für den eingereichten Auftrag korrekt funktioniert, rufen Sie mit kubectl die benutzerdefinierte `verticalpodautoscaler`-Ressource ab und sehen Sie sich Ihre Skalierungsempfehlungen an. Mit dem folgenden Befehl werden beispielsweise Empfehlungen für den Beispielauftrag aus dem Abschnitt Einreichen eines Spark-Auftrags mit vertikalem Auto Scaling abgefragt:

```
kubectl get verticalpodautoscalers --all-namespaces \
-l=emr-containers.amazonaws.com/dynamic.sizing.signature=test-signature
```

Die Ausgabe dieser Abfrage sollte wie folgt aussehen:

```
NAME                                                        MODE   CPU         MEM
 PROVIDED   AGE
```

```
ds-jceyefkxnhrvdzw6djum3naf2abm6o63a6dvjkkedqtkhlrf25eq-vpa   Off    3304504865  True
        87m
```

Wenn Ihre Ausgabe nicht ähnlich aussieht oder einen Fehlercode enthält, finden Sie weitere Schritte zur Behebung des Problems in Fehlerbehebung von Amazon EMR im vertikalen Auto Scaling von EKS.

Konfigurieren des vertikalen Auto Scalings für Amazon EMR in EKS

Sie können vertikales Auto Scaling konfigurieren, wenn Sie Amazon-EMR-Spark-Aufträge über die StartJobRun API senden. Stellen Sie die Konfigurationsparameter für Auto Scaling auf dem Spark-Treiber-Pod ein, wie im Beispiel unter Einreichen eines Spark-Auftrags mit vertikalem Auto Scaling gezeigt.

Der vertikale Auto-Scaling-Operator von Amazon EMR in EKS hört Treiber-Pods mit Auto Scaling ab und richtet dann die Integration mit dem vertikalen Kubernetes-Pod-Autoscaler (VPA) mit den Einstellungen auf dem Treiber-Pod ein. Dies erleichtert die Ressourcenverfolgung und Auto Scaling von Spark-Ausführer-Pods.

In den folgenden Abschnitten werden die Parameter beschrieben, die Sie bei der Konfiguration des vertikalen Auto Scalings für Ihren Amazon-EKS-Cluster verwenden können.

> **Note**
>
> Konfigurieren Sie den Feature-Toggle-Parameter als Label und konfigurieren Sie die übrigen Parameter als Anmerkungen auf dem Spark-Treiber-Pod. Die Auto-Scaling-Parameter gehören zur `emr-containers.amazonaws.com/`-Domain und haben das Präfix `dynamic.sizing`.

Erforderliche Parameter

Sie müssen die folgenden beiden Parameter in den Spark-Auftrag-Treiber aufnehmen, wenn Sie Ihren Auftrag einreichen:

Schlüssel	Beschreibung	Akzeptierte Werte	Standardwert	Typ	Spark-Parameter[1]
`dynamic.sizing`	Feature einschalten	`true`, `false`	nicht gesetzt	Bezeichnung	`spark.kubernetes.driver.label.emr-containers.amazonaws.com/dynamic.sizing`
`dynamic.sizing.signature`	Auftrag-Signatur	string	nicht gesetzt	Anmerkung	`spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/dynamic.sizing.signature`

[1] Verwenden Sie diesen Parameter als `SparkSubmitParameter` oder `ConfigurationOverride` in der `StartJobRun`-API.

- **`dynamic.sizing`** – Sie können das vertikale Auto Scaling mit dem `dynamic.sizing`-Label ein- und ausschalten. Um das vertikale Auto Scaling zu aktivieren, stellen Sie auf dem Spark-Treiber-Pod die Option `dynamic.sizing` auf `true` ein. Wenn Sie diese Bezeichnung weglassen oder sie auf einen anderen Wert als `true` festlegen, ist das vertikale Auto Scaling ausgeschaltet.
- **`dynamic.sizing.signature`** – Legen Sie die Aufgabensignatur mit der Anmerkung `dynamic.sizing.signature` auf dem Treiber-Pod fest. Vertikales Auto Scaling aggregiert Ihre Ressourcennutzungsdaten über verschiedene Ausführungen von Amazon-EMR-Spark-Aufträge hinweg, um Ressourcenempfehlungen abzuleiten. Sie geben die eindeutige Kennung an, um die Aufträge miteinander zu verknüpfen.

> **Note**
>
> Wenn sich Ihre Aufgabe in einem festen Intervall wiederholt, z. B. täglich oder wöchentlich, sollte Ihre Aufgabensignatur für jede neue Instance der Aufgabe gleich bleiben. Dadurch wird sichergestellt, dass mit vertikalem Auto Scaling Empfehlungen für verschiedene Ausführungen des Aufträge berechnet und aggregiert werden können.

[1] Verwenden Sie diesen Parameter als `SparkSubmitParameter` oder `ConfigurationOverride` in der `StartJobRun`-API.

Optionale Parameter

Vertikales Auto Scaling unterstützt auch die folgenden optionalen-Parameter. Legen Sie sie als Anmerkungen auf dem Treiber-Pod fest.

Schlüssel	Beschreibung	Akzeptierte Werte	Standardwert	Typ	Spark-Parameter[1]
dynamic.sizing.mode	Modus für vertikales Auto Scaling	`Off`, `Initial`, `Auto`	`Off`	Anmerkung	`spark.kubernetes.driver.label.emr-containers.amazonaws.com/dynamic.sizing.mode`
dynamic.sizing.scale.memory	Aktiviert die Speichers kalierung	*`true`*, *`false`*	`true`	Anmerkung	`spark.kubernetes.driver.label.emr-containers.amazonaws.com/dyna`

Schlüssel	Beschreibung	Akzeptierte Werte	Standardwert	Typ	Spark-Parameter[1]
					`mic.sizing.scale.memory`
`dynamic.sizing.scale.cpu`	CPU-Skalierung ein- oder ausschalten	*`true`*, *`false`*	`false`	Anmerkung	`spark.kubernetes.driver.label.emr-containers.amazonaws.com/dynamic.sizing.scale.cpu`
`dynamic.sizing.scale.memory.min`	Mindestlimit für die Speicherskalierung	Zeichenfolge, Ressourcenmenge K8s, z. B.: `1G`	nicht gesetzt	Anmerkung	`spark.kubernetes.driver.label.emr-containers.amazonaws.com/dynamic.sizing.scale.memory.min`

Schlüssel	Beschreibung	Akzeptierte Werte	Standardwert	Typ	Spark-Parameter[1]
dynamic.sizing.scale.memory.max	Höchstgrenze für die Speicherskalierung	Zeichenfolge, Ressourcenmenge K8s, z. B.: 4G	nicht gesetzt	Anmerkung	spark.kubernetes.driver.label.emr-containers.amazonaws.com/dynamic.sizing.scale.memory.max
dynamic.sizing.scale.cpu.min	Mindestlimit für die CPU-Skalierung	Zeichenfolge, Ressourcenmenge K8s, z. B.: 1	nicht gesetzt	Anmerkung	spark.kubernetes.driver.label.emr-containers.amazonaws.com/dynamic.sizing.scale.cpu.min
dynamic.sizing.scale.cpu.max	Höchstgrenze für die CPU-Skalierung	Zeichenfolge, Ressourcenmenge K8s, z. B.: 2	nicht gesetzt	Anmerkung	spark.kubernetes.driver.label.emr-containers.amazonaws.com/dynamic.sizing.scale.cpu.max

Modi für vertikales Auto Scaling

Der `mode`-Parameter ist den verschiedenen Auto-Scaling-Modi zugeordnet, die der VPA unterstützt. Verwenden Sie die `dynamic.sizing.mode`-Anmerkung auf dem Treiber-Pod, um den Modus einzustellen. Folgende Werte werden für diesen Parameter unterstützt:

- Aus – Ein Probelaufmodus, in dem Sie zwar Empfehlungen überwachen können, das Auto Scaling jedoch nicht durchgeführt wird. Dies ist der Standardmodus für das vertikale Auto Scaling. In diesem Modus berechnet die zugehörige vertikale Pod-Autoscaler-Ressource Empfehlungen, und Sie können die Empfehlungen mit Tools wie kubectl, Prometheus und Grafana überwachen.
- Anfänglich – In diesem Modus skaliert VPA Ressourcen beim Start des Aufträge automatisch, sofern Empfehlungen auf der Grundlage historischer Ausführungen des Aufträge verfügbar sind, z. B. im Fall eines wiederkehrenden Aufträge.
- Automatisch – In diesem Modus entfernt VPA die Spark-Ausführer-Pods und skaliert sie automatisch mit den empfohlenen Ressourceneinstellungen, wenn der Spark-Treiber-Pod sie neu startet. Manchmal entfernt die VPA laufende Spark-Ausführer-Pods, sodass es zu zusätzlicher Latenz kommen kann, wenn der unterbrochene Ausführer erneut versucht wird.

Skalierung von Ressourcen

Wenn Sie das vertikale Auto Scaling einrichten, können Sie auswählen, ob die CPU- und die Speicherressourcen skaliert werden sollen. Stellen Sie die Anmerkungen `dynamic.sizing.scale.cpu` und `dynamic.sizing.scale.memory` auf `true` oder `false` ein. Standardmäßig ist die CPU-Skalierung auf `false` und die Speicherskalierung auf `true` eingestellt.

Mindest- und Höchstwerte für Ressourcen (Grenzen)

Optional können Sie auch Grenzen für die CPU- und Speicherressourcen festlegen. Wählen Sie einen Mindest- und Höchstwert für diese Ressourcen mit den `dynamic.sizing.[memory/cpu].[min/max]`-Anmerkungen, wenn Sie das Auto Scaling aktivieren. In der Standardeinstellung haben die Ressourcen keine Einschränkungen. Legen Sie die Anmerkungen als Zeichenkettenwerte fest, die eine Kubernetes-Ressourcenmenge darstellen. Stellen Sie beispielsweise `dynamic.sizing.memory.max` auf `4G` ein, um 4 GB darzustellen.

Überwachen des vertikalen Auto Scalings für Amazon EMR in EKS

Sie können das Kubernetes-Befehlszeilentool kubectl verwenden, um die aktiven, vertikalen Auto-Scaling-Empfehlungen in Ihrem Cluster aufzulisten. Sie können auch Ihre verfolgten

Aufgabensignaturen einsehen und alle nicht benötigten Ressourcen löschen, die mit den Signaturen verknüpft sind.

Führen Sie die Empfehlungen für vertikales Auto Scaling für Ihren Cluster auf

Verwenden Sie kubectl, um die `verticalpodautoscaler`-Ressource abzurufen und den aktuellen Status und die Empfehlungen einzusehen. Die folgende Beispielabfrage gibt alle aktiven Ressourcen in Ihrem Amazon-EKS-Cluster zurück.

```
kubectl get verticalpodautoscalers \
-o custom-columns="NAME:.metadata.name,"\
"SIGNATURE:.metadata.labels.emr-containers\.amazonaws\.com/dynamic\.sizing
\.signature,"\
"MODE:.spec.updatePolicy.updateMode,"\
"MEM:.status.recommendation.containerRecommendations[0].target.memory" \
--all-namespaces
```

Die Ausgabe dieser Abfrage ähnelt der folgenden:

```
NAME                  SIGNATURE              MODE      MEM
ds-example-id-1-vpa   job-signature-1        Off       none
ds-example-id-2-vpa   job-signature-2        Initial   12936384283
```

Die Empfehlungen für vertikales Auto Scaling für Ihren Cluster abfragen und löschen

Wenn Sie eine Amazon-EMR-Ressource mit vertikaler Auto-Scaling-Auftragausführung löschen, wird automatisch das zugehörige VPA-Objekt gelöscht, das Empfehlungen verfolgt und speichert.

Im folgenden Beispiel wird kubectl verwendet, um Empfehlungen für einen Auftrag zu löschen, der durch eine Signatur identifiziert wird:

```
kubectl delete jobrun -n emr -l=emr-containers\.amazonaws\.com/dynamic\.sizing
\.signature=integ-test
jobrun.dynamicsizing.emr.services.k8s.aws "ds-job-signature" deleted
```

Wenn Sie die spezifische Aufgabensignatur nicht kennen oder alle Ressourcen im Cluster löschen möchten, können Sie in Ihrem Befehl `--all` oder `--all-namespaces` anstelle der eindeutigen Aufgaben-ID verwenden, wie im folgenden Beispiel gezeigt:

```
kubectl delete jobruns --all --all-namespaces
```

```
jobrun.dynamicsizing.emr.services.k8s.aws "ds-example-id" deleted
```

Deinstallieren Sie den vertikalen Auto-Scaling-Operator von Amazon EMR in EKS

Wenn Sie den vertikalen Auto-Scaling-Operator aus Ihrem Amazon-EKS-Cluster entfernen möchten, verwenden Sie den `cleanup`-Befehl mit der Operator-SDK-CLI, wie im folgenden Beispiel gezeigt. Dadurch werden auch Upstream-Abhängigkeiten gelöscht, die mit dem Operator installiert wurden, wie z. B. der vertikale Pod-Autoscaler.

```
operator-sdk cleanup emr-dynamic-sizing
```

Wenn Sie den Operator löschen, laufende Aufträge auf dem Cluster befinden, werden diese Aufträge ohne vertikales Auto Scaling weiter ausgeführt. Wenn Sie Aufträge auf dem Cluster einreichen, nachdem Sie den Operator gelöscht haben, ignoriert Amazon EMR in EKS alle vertikalen Auto-Scaling-Parameter, die Sie möglicherweise während der Konfiguration definiert haben.

Ausführen interaktiver Workloads auf Amazon EMR in EKS

Ein interaktiver Endpunkt ist ein Gateway, das Amazon EMR Studio mit Amazon EMR in EKS verbindet, sodass Sie interaktive Workloads ausführen können. Sie können interaktive Endpunkte mit EMR Studio verwenden, um interaktive Analysen mit Datensätzen in Datenspeichern wie Amazon S3 und Amazon DynamoDB durchzuführen.

Anwendungsfälle

- Erstellen Sie ein ETL-Skript mit der EMR-Studio-IDE-Erfahrung. Die IDE nimmt On-Premises-Daten auf und speichert sie nach Transformationen zur späteren Analyse in Amazon S3.
- Verwenden Sie Notebooks, um Datensätze zu untersuchen und ein Machine-Learning-Modell zu trainieren, um Anomalien in den Datensätzen zu erkennen.
- Erstellen Sie Skripts, die tägliche Berichte für Analyseanwendungen wie Geschäfts-Dashboards generieren.

Themen

Übersicht über interaktive Endpunkte

Ein interaktiver Endpunkt bietet interaktiven Clients wie Amazon EMR Studio die Möglichkeit, sich mit Amazon EMR in EKS-Clustern zu verbinden, um interaktive Workloads auszuführen. Der interaktive Endpunkt wird von einem Jupyter Enterprise Gateway unterstützt, das die Funktionen zur Verwaltung des Kernel-Lebenszyklus per Fernzugriff bereitstellt, die interaktive Clients benötigen. Kernel sind sprachspezifische Prozesse, die mit dem auf Jupyter basierenden Amazon-EMR-Studio-Client interagieren, um interaktive Workloads auszuführen.

Interaktive Endpunkte unterstützen die folgenden Kernel:

- Python 3
- PySpark auf Kubernetes
- Apache Spark mit Scala

> **Note**
>
> Die Preise für Amazon EMR in EKS gelten für die interaktiven Endpunkte und Kernel. Weitere Informationen finden Sie auf der Preisseite für Amazon EMR in EKS.

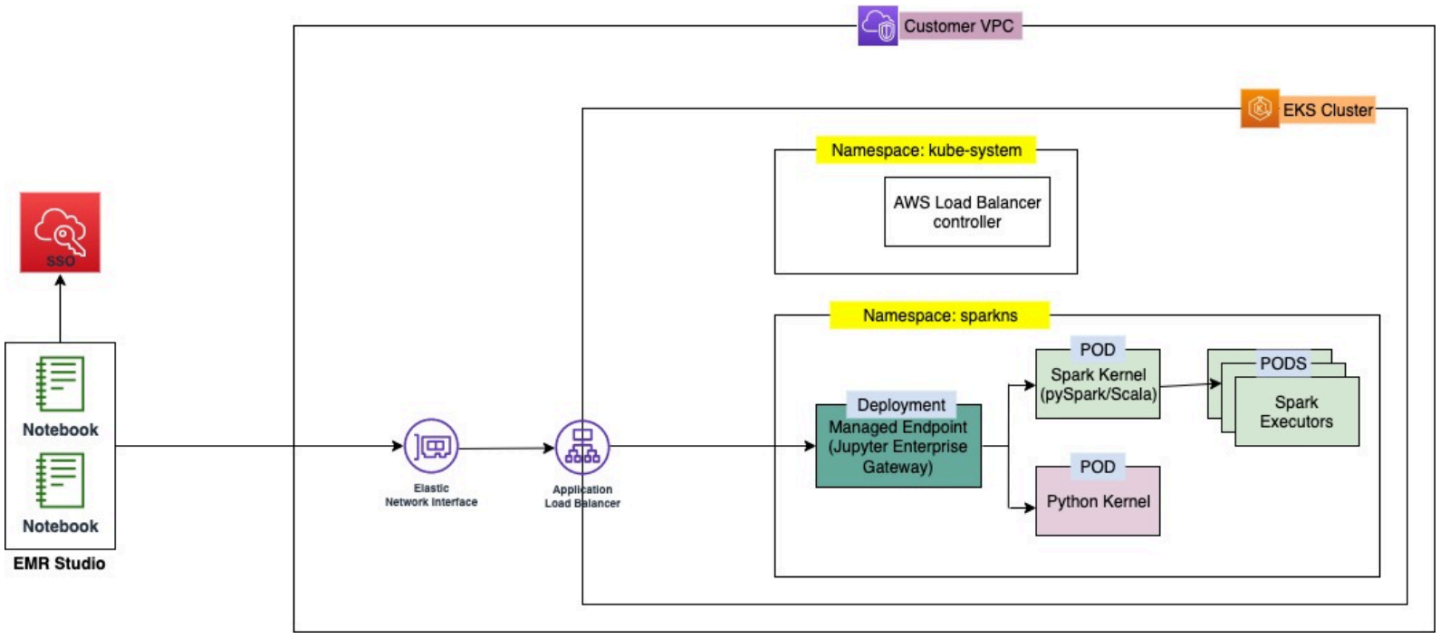
Die folgenden Entitäten sind erforderlich, damit EMR Studio eine Verbindung mit Amazon EMR in EKS herstellen kann.

- Virtueller Amazon EMR in EKS-Cluster – Ein virtueller Cluster ist ein Kubernetes-Namespace, bei dem Sie Amazon EMR registrieren. Amazon EMR verwendet virtuelle Cluster, um Aufträge auszuführen und Endpunkte zu hosten. Sie können mehrere virtuelle Cluster mit demselben physischen Cluster sichern. Jeder virtuelle Cluster ist jedoch einem Namespace auf einem Amazon-EKS-Cluster zugeordnet. Virtuelle Cluster erzeugen keine aktiven Ressourcen, die zu Ihrer Rechnung beitragen oder für die ein Lebenszyklus-Management außerhalb des Services erforderlich ist.
- Interaktiver Endpunkt von Amazon EMR in EKS – Ein interaktiver Endpunkt ist ein HTTPS-Endpunkt, mit dem EMR-Studio-Benutzer eine Verbindung zu einem Workspace herstellen können. Sie können nur von Ihrem EMR Studio aus auf die HTTPS-Endpunkte zugreifen und sie in einem privaten Subnetz der Amazon Virtual Private Cloud (Amazon VPC) für Ihren Amazon-EKS-Cluster erstellen.

  Die Python- PySpark, und Spark-Scala-Kernel verwenden die in Ihrer Amazon EMR on EKS-Jobausführungsrolle definierten Berechtigungen, um andere aufzurufen. AWS-Services Alle Kernel und Benutzer, die eine Verbindung zum interaktiven Endpunkt herstellen, verwenden die Rolle, die Sie bei der Erstellung des Endpunkts angegeben haben. Wir empfehlen, separate Endpoints für verschiedene Benutzer zu erstellen und dafür zu sorgen, dass die Benutzer unterschiedliche Rollen (IAM) haben. AWS Identity and Access Management
- AWS Application Load Balancer Balancer-Controller — Der AWS Application Load Balancer Balancer-Controller verwaltet Elastic Load Balancing für einen Amazon EKS Kubernetes-Cluster. Der Controller stellt einen Application Load Balancer (ALB) bereit, wenn Sie eine Kubernetes-Ingress-Ressource erstellen. Ein ALB macht einen Kubernetes-Service, z. B. einen interaktiven

Endpunkt, außerhalb des Amazon-EKS-Clusters, aber innerhalb derselben Amazon VPC verfügbar. Wenn Sie einen interaktiven Endpunkt erstellen, wird auch eine Ingress-Ressource bereitgestellt, die den interaktiven Endpunkt über die ALB verfügbar macht, damit interaktive Clients eine Verbindung herstellen können. Sie müssen nur einen AWS Application Load Balancer Balancer-Controller für jeden Amazon EKS-Cluster installieren.

Das folgende Diagramm zeigt die interaktive Endpunktarchitektur in Amazon EMR in EKS. Ein Amazon-EKS-Cluster umfasst die Rechenleistung zur Ausführung der analytischen Workloads und den interaktiven Endpunkt. Der Application-Load-Balancer-Controller wird im `kube-system`-Namespace ausgeführt. Die Workloads und interaktiven Endpunkte werden in dem Namespace ausgeführt, den Sie bei der Erstellung des virtuellen Clusters angeben. Wenn Sie einen interaktiven Endpunkt erstellen, erstellt die Amazon EMR in EKS-Kontrollebene die interaktive Endpunktbereitstellung im Amazon-EKS-Cluster. Darüber hinaus wird vom Load Balancer-Controller eine Instance des Application Load Balancer-Eingangs AWS erstellt. Der Application Load Balancer stellt die externe Schnittstelle für Clients wie EMR Studio bereit, um eine Verbindung zum Amazon-EMR-Cluster herzustellen und interaktive Workloads auszuführen.



Voraussetzungen für die Erstellung eines interaktiven Endpunkts auf Amazon EMR in EKS

In diesem Abschnitt werden die Voraussetzungen für die Einrichtung eines interaktiven Endpunkts beschrieben, mit dem EMR Studio eine Verbindung zu einem Amazon EMR in EKS-Cluster herstellen und interaktive Workloads ausführen kann.

AWS CLI

Folgen Sie den Schritten unterInstallieren Sie das AWS CLI, um die neueste Version von AWS Command Line Interface ()AWS CLI zu installieren.

Installation von eksctl

Befolgen Sie folgende Schritte unter Installieren `eksctl` um die neueste Version von eksctl zu installieren. Wenn Sie Kubernetes-Version 1.22 oder höher für Ihren Amazon-EKS-Cluster verwenden, verwenden Sie eine eksctl-Version größer als 0.117.0.

Amazon-EKS-Cluster

Erstellen Sie einen Amazon-EKS-Cluster. Im folgenden Beispiel wird gezeigt, wie Sie einen Amazon EMR in EKS-Cluster erstellen. Nachfolgend werden Anforderungen und Überlegungen für diesen Cluster aufgeführt:

- Der Cluster muss sich in derselben Amazon Virtual Private Cloud (VPC) befinden wie Ihr EMR Studio.

- Der Cluster muss über mindestens ein privates Subnetz verfügen, um interaktive Endpunkte zu aktivieren, Git-basierte Repositorys zu verknüpfen und den Application Load Balancer im privaten Modus zu starten.
- Es muss mindestens ein gemeinsames privates Subnetz zwischen Ihrem EMR Studio und dem Amazon-EKS-Cluster geben, den Sie für die Registrierung Ihres virtuellen Clusters verwenden. Dadurch wird sichergestellt, dass Ihr interaktiver Endpunkt als Option in Ihren Studio-Workspaces angezeigt wird, und die Konnektivität von Studio zum Application Load Balancer aktiviert.

  Sie können zwischen zwei Methoden wählen, um Ihr Studio und Ihren Amazon-EKS-Cluster zu verbinden:

  - Erstellen Sie einen Amazon-EKS-Cluster und verknüpfen Sie ihn mit den Subnetzen, die zu Ihrem EMR Studio gehören.
  - Alternativ können Sie ein EMR Studio erstellen und die privaten Subnetze für Ihren Amazon-EKS-Cluster angeben.
- Amazon-EKS-optimierte ARM-Amazon-Linux-AMIs werden für Amazon EMR auf interaktiven EKS-Endpunkten nicht unterstützt.
- Interaktive Endpunkte funktionieren mit Amazon EKS-Clustern, die Kubernetes-Versionen bis 1.28 verwenden.
- Es werden nur von Amazon EKS verwaltete Knotengruppen unterstützt.

Gewähren eines Clusterzugriff für Amazon EMR in EKS

Verwenden Sie die Schritte unter Clusterzugriff für Amazon EMR in EKS gewähren, um Amazon EMR in EKS Zugriff auf einen bestimmten Namespace in Ihrem Cluster zu gewähren.

IRSA im Amazon-EKS-Cluster aktivieren

Um IAM-Rollen für Servicekonten (IRSA) auf dem Amazon-EKS-Cluster zu aktivieren, folgen Sie den Schritten unter Aktivieren von IAM-Rollen für Servicekonten (IRSA).

Eine IAM-Auftragausführungsrolle erstellen

Sie müssen eine IAM-Rolle erstellen, um Workloads auf Amazon EMR auf interaktiven EKS-Endpunkten auszuführen. In dieser Dokumentation wird diese IAM-Rolle als Auftragsausausführungsrolle bezeichnet. Diese IAM-Rolle wird sowohl dem interaktiven Endpunkt-Container als auch den eigentlichen Ausführungscontainern zugewiesen, die erstellt werden, wenn Sie Aufträge mit EMR Studio einreichen. Sie benötigen den Amazon-Ressourcennamen (ARN) Ihrer Aufgabenausführungsrolle für Amazon EMR auf EKS. Dazu sind zwei Schritte erforderlich:

- Erstellen einer IAM-Rollen-Auftragsausführung.
- Aktualisieren Sie die Vertrauensrichtlinie der Auftragsausführungsrolle.

Gewähren Sie Benutzern Zugriff auf Amazon EMR in EKS

Die IAM-Entität (Benutzer oder Rolle), die die Anfrage zur Erstellung eines interaktiven Endpunkts stellt, muss außerdem über die folgenden Amazon-EC2- und `emr-containers`-Berechtigungen

verfügen. Folgen Sie den unter Gewähren Sie Benutzern Zugriff auf Amazon EMR in EKS beschriebenen Schritten, um diese Berechtigungen zu gewähren, die es Amazon EMR in EKS ermöglichen, die Sicherheitsgruppen zu erstellen, zu verwalten und zu löschen, die den eingehenden Datenverkehr auf den Load Balancer Ihres interaktiven Endpunkts beschränken.

Die folgenden `emr-containers`-Berechtigungen ermöglichen es dem Benutzer, grundlegende interaktive Endpunktoperationen durchzuführen:

```
"ec2:CreateSecurityGroup",
"ec2:DeleteSecurityGroup",
"ec2:AuthorizeSecurityGroupEgress",
"ec2:AuthorizeSecurityGroupIngress",
"ec2:RevokeSecurityGroupEgress",
"ec2:RevokeSecurityGroupIngress"

"emr-containers:CreateManagedEndpoint",
"emr-containers:ListManagedEndpoints",
"emr-containers:DescribeManagedEndpoint",
"emr-containers:DeleteManagedEndpoint"
```

Den Amazon-EKS-Cluster mit Amazon EMR registrieren

Richten Sie einen virtuellen Cluster ein und ordnen Sie ihn dem Namespace im Amazon-EKS-Cluster zu, in dem Sie Ihre Aufträge ausführen möchten. Verwenden Sie für AWS Fargate reine Cluster denselben Namespace sowohl für den virtuellen Amazon EMR on EKS-Cluster als auch für das Fargate-Profil.

Informationen zur Einrichtung eines virtuellen Amazon EMR auf einem EKS-Cluster finden Sie unter Den Amazon-EKS-Cluster mit Amazon EMR registrieren.

Stellen Sie den AWS Load Balancer Controller auf dem Amazon EKS-Cluster bereit

Für Ihren Amazon EKS-Cluster ist ein AWS Application Load Balancer erforderlich. Sie müssen nur einen Application-Load-Balancer-Controller pro Amazon-EKS-Cluster einrichten. Informationen zur Einrichtung des AWS Application Load Balancer Balancer-Controllers finden Sie unter Installation des Load AWS Balancer Controller-Add-ons im Amazon EKS-Benutzerhandbuch.

Einen interaktiven Endpunkt für Ihren virtuellen Cluster erstellen

Auf dieser Seite wird beschrieben, wie Sie mithilfe der AWS Befehlszeilenschnittstelle ()AWS CLI einen interaktiven Endpunkt erstellen.

Erstellen Sie einen interaktiven Endpunkt mit dem **`create-managed-endpoint`**-Befehl

Geben Sie die Parameter im `create-managed-endpoint`-Befehl wie folgt an. Amazon EMR in EKS unterstützt die Erstellung interaktiver Endpunkte mit Amazon-EMR-Versionen 6.7.0 und später.

```
aws emr-containers create-managed-endpoint \
--type JUPYTER_ENTERPRISE_GATEWAY \
--virtual-cluster-id 1234567890abcdef0xxxxxxxx \
--name example-endpoint-name \
--execution-role-arn arn:aws:iam::444455556666:role/JobExecutionRole \
--release-label emr-6.9.0-latest \
--configuration-overrides '{
    "applicationConfiguration": [{
        "classification": "spark-defaults",
        "properties": {
            "spark.driver.memory": "2G"
        }
    }],
    "monitoringConfiguration": {
        "cloudWatchMonitoringConfiguration": {
            "logGroupName": "log_group_name",
            "logStreamNamePrefix": "log_stream_prefix"
        },
        "persistentAppUI": "ENABLED",
        "s3MonitoringConfiguration": {
            "logUri": "s3://my_s3_log_location"
        }
    }
}'
```

Weitere Informationen finden Sie unter Parameter für die Erstellung eines interaktiven Endpunkts.

Erstellen Sie einen interaktiven Endpunkt mit angegebenen Parametern in einer JSON-Datei

1. Erstellen Sie eine `create-managed-endpoint-request.json`-Datei und geben Sie die erforderlichen Parameter für Ihren Endpunkt an, wie in der folgenden JSON-Datei gezeigt:

```
{
    "name": "MY_TEST_ENDPOINT",
    "virtualClusterId": "MY_CLUSTER_ID",
    "type": "JUPYTER_ENTERPRISE_GATEWAY",
    "releaseLabel": "emr-6.9.0-latest",
    "executionRoleArn": "arn:aws:iam::444455556666:role/JobExecutionRole",
    "configurationOverrides":
    {
        "applicationConfiguration":
        [
            {
                "classification": "spark-defaults",
                "properties":
                {
                    "spark.driver.memory": "8G"
                }
            }
        ],
        "monitoringConfiguration":
        {
            "persistentAppUI": "ENABLED",
            "cloudWatchMonitoringConfiguration":
            {
                "logGroupName": "my_log_group",
                "logStreamNamePrefix": "log_stream_prefix"
            },
            "s3MonitoringConfiguration":
            {
                "logUri": "s3://my_s3_log_location"
            }
        }
    }
}
```

2. Verwenden Sie den `create-managed-endpoint`-Befehl mit einem Pfad zu der lokal gespeicherten `create-managed-endpoint-request.json`-Datei in Amazon S3.

```
aws emr-containers create-managed-endpoint \
--cli-input-json  file://./create-managed-endpoint-request.json --region AWS-Region
```

Ausgabe von interaktivem Endpunkt erstellen

Die Ausgabe sollte im Terminal folgendermaßen aussehen. Die Ausgabe enthält den Namen und die Kennung Ihres neuen interaktiven Endpunkts:

```
{
    "id": "1234567890abcdef0",
    "name": "example-endpoint-name",
    "arn": "arn:aws:emr-containers:us-west-2:111122223333:/
virtualclusters/444455556666/endpoints/444455556666",
    "virtualClusterId": "111122223333xxxxxxxx"
}
```

Beim Ausführen von `aws emr-containers create-managed-endpoint` wird ein selbstsigniertes Zertifikat erstellt, das die HTTPS-Kommunikation zwischen EMR Studio und dem interaktiven Endpunktserver ermöglicht.

Wenn Sie `create-managed-endpoint` ausführen und die Voraussetzungen nicht erfüllt haben, gibt Amazon EMR eine Fehlermeldung mit den Maßnahmen zurück, die Sie ergreifen müssen, um fortzufahren.

Parameter für die Erstellung eines interaktiven Endpunkts

Themen

- Erforderliche Parameter für interaktive Endpunkte
- Optionale Parameter für interaktive Endpunkte

Erforderliche Parameter für interaktive Endpunkte

Sie müssen die folgenden Parameter angeben, wenn Sie einen interaktiven Endpunkt erstellen:

`--type`

Verwenden Sie `JUPYTER_ENTERPRISE_GATEWAY`. Dies ist der einzige unterstützte Typ.

`--virtual-cluster-id`

Die ID des virtuellen Clusters, den Sie bei Amazon EMR in EKS registriert haben.

`--name`

Ein aussagekräftiger Name für den interaktiven Endpunkt, der EMR-Studio-Benutzern hilft, ihn aus der Dropdownliste auszuwählen.

`--execution-role-arn`

Der Amazon-Ressourcenname (ARN) Ihrer IAM-Aufgabenausführungsrolle für Amazon EMR in EKS, die als Teil der Voraussetzungen erstellt wurde.

`--release-label`

Die Versionskennung der Amazon-EMR-Version, die für den Endpunkt verwendet werden soll. z. B. `emr-6.9.0-latest`. Amazon EMR in EKS unterstützt interaktive Endpunkte mit Amazon-EMR-Versionen 6.7.0 und höher.

Optionale Parameter für interaktive Endpunkte

Optional können Sie beim Erstellen eines interaktiven Endpunkts auch die folgenden Parameter angeben:

`--configuration-overrides`

Um die Standardkonfigurationen für Anwendungen zu überschreiben, geben Sie ein Konfigurationsobjekt an. Sie können eine Syntax-Kurznotation verwenden, um die Konfiguration anzugeben oder Sie können auf die Konfiguration in einer JSON-Datei zu verweisen.

Konfigurationsobjekte bestehen aus einer Klassifizierung, Eigenschaften und optionalen verschachtelten Konfigurationen. Eigenschaften bestehen aus den Einstellungen, die Sie in dieser Datei überschreiben möchten. Sie können mehrere Klassifizierungen für mehrere Anwendungen in einem einzigen JSON-Objekt angeben. Die verfügbaren Konfigurationsklassifizierungen variieren je nach Amazon EMR in EKS-Version. Eine Liste der Konfigurationsklassifizierungen, die für jede Version von Amazon EMR in EKS verfügbar sind, finden Sie unter Versionen von Amazon EMR in EKS. Zusätzlich zu den Konfigurationsklassifizierungen, die für jede Version aufgeführt sind, bieten interaktive Endpunkte zusätzliche `jeg-config`-Klassifizierungen. Weitere Informationen finden Sie unter Konfigurationsoptionen für Jupyter Enterprise Gateway (JEG).

Konfiguration von Einstellungen für interaktive Endpunkte

Überwachung von -Spark-Aufträgen

Damit Sie Fehler überwachen und beheben können, konfigurieren Sie Ihre interaktiven Endpunkte so, dass die mit dem Endpunkt initiierten Jobs Protokollinformationen an Amazon S3, Amazon CloudWatch Logs oder beide senden können. In den folgenden Abschnitten wird beschrieben, wie Sie Spark-Anwendungsprotokolle für die Spark-Aufträge, die Sie mit Amazon EMR auf interaktiven EKS-Endpunkten starten, an Amazon S3 senden.

IAM-Richtlinie für Amazon-S3-Protokolle konfigurieren

Bevor Ihre Kernel Protokolldaten an Amazon S3 senden können, muss die Berechtigungsrichtlinie für die Auftragausführungsrolle die folgenden Berechtigungen enthalten. Ersetzen Sie *DOC-EXAMPLE-BUCKET-LOGGING* durch den Namen ihres Protokollierungs-Buckets.

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

> **Note**
>
> Amazon EMR in EKS kann auch einen S3-Bucket erstellen. Wenn ein S3-Bucket nicht verfügbar ist, nehmen Sie die `s3:CreateBucket`-Erlaubnis in die IAM-Richtlinie auf.

Nachdem Sie Ihrer Ausführungsrolle die erforderlichen Berechtigungen zum Senden von Protokollen an den S3-Bucket erteilt haben, werden Ihre Protokolldaten an die folgenden Amazon-S3-Speicherorte gesendet. Dies geschieht, wenn `s3MonitoringConfiguration` im `monitoringConfiguration`-Abschnitt einer `create-managed-endpoint`-Anfrage übergeben wird.

- Treiberprotokolle – `logUri/virtual-cluster-id/endpoints/endpoint-id/containers/spark-application-id/spark-application-id-driver/(stderr.gz/stdout.gz)`
- Ausführungsprotokolle – `logUri/virtual-cluster-id/endpoints/endpoint-id/containers/spark-application-id/executor-pod-name-exec-<Number>/(stderr.gz/stdout.gz)`

> **Note**
>
> Amazon EMR in EKS lädt die Endpunktprotokolle nicht in Ihren S3-Bucket hoch.

Spezifizierung von benutzerdefinierten Pod-Vorlagen mit interaktiven Endpunkten

Sie können interaktive Endpunkte erstellen, in denen Sie benutzerdefinierte Pod-Vorlagen für Treiber und Ausführer angeben. Pod-Vorlagen sind Spezifikationen, die bestimmen, wie jeder Pod ausgeführt wird. Sie können Pod-Vorlagendateien verwenden, um die Konfigurationen von Treiber- oder Ausführer-Pods zu definieren, die von Spark-Konfigurationen nicht unterstützt werden. Pod-Vorlagen werden derzeit in Amazon-EMR-Versionen 6.3.0 und höher unterstützt.

Weitere Informationen zu Pod-Vorlagen finden Sie unter Verwenden von Pod-Vorlagen im Entwicklerhandbuch zu Amazon EMR in EKS.

Das folgende Beispiel zeigt, wie Sie einen interaktiven Endpunkt mit Pod-Vorlagen erstellen:

```
aws emr-containers create-managed-endpoint \
    --type JUPYTER_ENTERPRISE_GATEWAY \
    --virtual-cluster-id virtual-cluster-id \
    --name example-endpoint-name \
    --execution-role-arn arn:aws:iam::aws-account-id:role/EKSClusterRole \
    --release-label emr-6.9.0-latest \
    --configuration-overrides '{
        "applicationConfiguration": [
```

```
        {
            "classification": "spark-defaults",
            "properties": {
                "spark.kubernetes.driver.podTemplateFile": "path/to/driver/
template.yaml",
                "spark.kubernetes.executor.podTemplateFile": "path/to/executor/
template.yaml"
            }
        }]
    }'
```

Bereitstellen eines JEG-Pods in einer Knotengruppe

Die Platzierung von JEG-Pods (Jupyter Enterprise Gateway) ist ein Feature, mit dem Sie einen interaktiven Endpunkt auf einer bestimmten Knotengruppe bereitstellen können. Mit diesem Feature können Sie Einstellungen wie `instance type` für den interaktiven Endpunkt konfigurieren.

Zuordnen eines JEG-Pods zu einer verwalteten Knotengruppe

Mit der folgenden Konfigurationseigenschaft können Sie den Namen einer verwalteten Knotengruppe auf Ihrem Amazon-EKS-Cluster angeben, in der der JEG-Pod bereitgestellt wird.

```
//payload
--configuration-overrides '{
      "applicationConfiguration": [
            {
                "classification": "endpoint-configuration",
                "properties": {
                    "managed-nodegroup-name": NodeGroupName
                }
            }
        ]
    }'
```

Bei einer Knotengruppe muss die Kubernetes-Kennung `for-use-with-emr-containers-managed-endpoint-ng=`*`NodeGroupName`* an alle Knoten angehängt sein, die Teil der Knotengruppe sind. Um alle Knoten einer Knotengruppe aufzulisten, die dieses Tag haben, verwenden Sie den folgenden Befehl:

```
kubectl get nodes --show-labels | grep for-use-with-emr-containers-managed-endpoint-
ng=NodeGroupName
```

Wenn die Ausgabe des obigen Befehls keine Knoten zurückgibt, die Teil Ihrer verwalteten Knotengruppe sind, dann gibt es in der Knotengruppe keine Knoten, denen die `for-use-with-emr-containers-managed-endpoint-ng=`*`NodeGroupName`*-Kubernetes-Kennzeichnung zugewiesen ist. Führen Sie in diesem Fall die folgenden Schritte aus, um diese Kennzeichnung den Knoten in Ihrer Knotengruppe zuzuordnen.

1. Verwenden Sie den folgenden Befehl, um die `for-use-with-emr-containers-managed-endpoint-ng=`*`NodeGroupName`*-Kubernetes-Kennzeichnung allen Knoten in einer verwalteten Knotengruppe *`NodeGroupName`* hinzuzufügen:

   ```
   kubectl label nodes --selector eks:nodegroup-name=NodeGroupName for-use-with-emr-containers-managed-endpoint-ng=NodeGroupName
   ```

2. Um alle verfügbaren Befehle aufzulisten, führen Sie den folgenden Befehl aus:

   ```
   kubectl get nodes --show-labels | grep for-use-with-emr-containers-managed-endpoint-ng=NodeGroupName
   ```

Eine verwaltete Knotengruppe muss der Sicherheitsgruppe eines Amazon-EKS-Clusters zugeordnet sein. Dies ist normalerweise der Fall, wenn Sie Ihren Cluster und Ihre verwaltete Knotengruppe mithilfe von `eksctl` erstellt haben. Sie können dies in der AWS Konsole anhand der folgenden Schritte überprüfen.

1. Gehen Sie in der Amazon-EKS-Konsole zu Ihrem Cluster.
2. Gehen Sie zur Registerkarte Netzwerk Ihres Clusters und notieren Sie sich die Cluster-Sicherheitsgruppe.
3. Gehen Sie zur Registerkarte Compute Ihres Clusters und klicken Sie auf den Namen der verwalteten Knotengruppe.
4. Vergewissern Sie sich auf der Registerkarte Details der verwalteten Knotengruppe, dass die Cluster-Sicherheitsgruppe, die Sie zuvor notiert haben, unter Sicherheitsgruppen aufgeführt ist.

Wenn die verwaltete Knotengruppe nicht an die Amazon-EKS-Cluster-Sicherheitsgruppe angehängt ist, müssen Sie das `for-use-with-emr-containers-managed-endpoint-sg=`*`ClusterName`*`/`*`NodeGroupName`*-Tag an die Sicherheitsgruppe der Knotengruppe anhängen. Führen Sie die unten folgenden Schritte aus, um diesen Tag anzuhängen.

1. Gehen Sie zur Amazon-EC2-Konsole und klicken Sie im linken Navigationsbereich auf Sicherheitsgruppen.
2. Wählen Sie die Sicherheitsgruppe Ihrer verwalteten Knotengruppe aus, indem Sie auf das Kontrollkästchen klicken.
3. Fügen Sie auf der Registerkarte Tags das Tag `for-use-with-emr-containers-managed-endpoint-sg=`*`ClusterName`*`/`*`NodeGroupName`* mithilfe der Schaltfläche Tags verwalten hinzu.

Zuordnen eines JEG-Pods zu einer selbstverwalteten Knotengruppe

Mit der folgenden Konfigurationseigenschaft können Sie den Namen einer selbstverwalteten oder nicht verwalteten Knotengruppe auf dem Amazon-EKS-Cluster angeben, auf dem der JEG-Pod bereitgestellt wird.

```
//payload
--configuration-overrides '{
      "applicationConfiguration": [
           {
               "classification": "endpoint-configuration",
               "properties": {
                   "self-managed-nodegroup-name": NodeGroupName
               }
           }
        ]
    }'
```

Bei einer Knotengruppe muss die Kubernetes-Kennung `for-use-with-emr-containers-managed-endpoint-ng=`*`NodeGroupName`* an alle Knoten angehängt sein, die Teil der Knotengruppe sind. Um alle verfügbaren Parameter aufzulisten, verwenden Sie den folgenden Befehl:

```
kubectl get nodes --show-labels | grep for-use-with-emr-containers-managed-endpoint-
ng=NodeGroupName
```

Wenn die Ausgabe des obigen Befehls keine Knoten zurückgibt, die Teil Ihrer selbstverwalteten Knotengruppe sind, dann gibt es in der Knotengruppe keine Knoten, denen die `for-use-with-emr-containers-managed-endpoint-ng=`*`NodeGroupName`* Kubernetes-Kennzeichnung zugewiesen ist. Führen Sie in diesem Fall die folgenden Schritte aus, um diese Kennzeichnung den Knoten in Ihrer Knotengruppe zuzuordnen.

1. Wenn Sie die selbstverwaltete Knotengruppe mit `eksctl` erstellt haben, verwenden Sie den folgenden Befehl, um das `for-use-with-emr-containers-managed-endpoint-ng=`*`NodeGroupName`*-Kubernetes-Label allen Knoten in der selbstverwalteten Knotengruppe *`NodeGroupName`* auf einmal hinzuzufügen.

   ```
   kubectl label nodes --selector alpha.eksctl.io/nodegroup-name=NodeGroupName for-use-with-emr-containers-managed-endpoint-ng=NodeGroupName
   ```

   Wenn Sie `eksctl` nicht zum Erstellen der selbstverwalteten Knotengruppe verwendet haben, müssen Sie den Selektor im obigen Befehl durch eine andere Kubernetes-Kennzeichnung ersetzen, die allen Knoten der Knotengruppe zugeordnet ist.

2. Verwenden Sie den folgenden Befehl, um zu überprüfen, ob die Knoten korrekt beschriftet wurden:

   ```
   kubectl get nodes --show-labels | grep for-use-with-emr-containers-managed-endpoint-ng=NodeGroupName
   ```

Der Sicherheitsgruppe für die selbstverwaltete Knotengruppe muss das `for-use-with-emr-containers-managed-endpoint-sg=`*`ClusterName`*`/`*`NodeGroupName`*-Tag zugeordnet sein. Gehen Sie wie folgt vor, um das Tag an die Sicherheitsgruppe von AWS Management Console anzuhängen.

1. Navigieren Sie zur Amazon-EC2-Konsole. Wählen Sie im linken Navigationsbereich Sicherheitsgruppen.
2. Aktivieren Sie das Kontrollkästchen neben der Sicherheitsgruppe für Ihre selbstverwaltete Knotengruppe.
3. Verwenden Sie auf der Registerkarte Tags die Schaltfläche Tags verwalten, um das Tag `for-use-with-emr-containers-managed-endpoint-sg=`*`ClusterName`*`/`*`NodeGroupName`* hinzuzufügen. Ersetzen Sie *`ClusterName`* und *`NodeGroupName`* durch entsprechende Werte.

Zuordnen eines JEG-Pods zu einer verwalteten Knotengruppe mit On-Demand-Instances

Sie können auch zusätzliche Kennzeichnungen, sogenannte Kubernetes-Kennzeichnungsselektoren, definieren, um zusätzliche Einschränkungen oder Einschränkungen für die Ausführung eines interaktiven Endpunkts auf einem bestimmten Knoten oder einer bestimmten Knotengruppe

festzulegen. Das folgende Beispiel zeigt, wie On-Demand-Amazon-EC2-Instances für einen JEG-Pod verwendet werden.

```
--configuration-overrides '{
      "applicationConfiguration": [
            {
                "classification": "endpoint-configuration",
                "properties": {
                    "managed-nodegroup-name": NodeGroupName,
                    "node-labels": "eks.amazonaws.com/capacityType:ON_DEMAND"
                }
            }
        ]
    }'
```

> **Note**
>
> Sie können die Eigenschaft `node-labels` nur zusammen mit der Eigenschaft `managed-nodegroup-name` oder `self-managed-nodegroup-name` verwenden.

Konfigurationsoptionen für Jupyter Enterprise Gateway (JEG)

Amazon EMR in EKS verwendet Jupyter Enterprise Gateway (JEG), um interaktive Endpunkte zu aktivieren. Sie können die folgenden Werte für die JEG-Konfigurationen auf der Zulassungsliste festlegen, wenn Sie den Endpunkt erstellen.

- **`RemoteMappingKernelManager.cull_idle_timeout`** – Timeout in Sekunden (Ganzzahl), nach dessen Ablauf ein Kernel als inaktiv betrachtet wird und bereit ist, gelöscht zu werden. Werte von `0` oder niedriger deaktivieren das Culling. Kurze Timeouts können dazu führen, dass Kernel für Benutzer mit schlechten Netzwerkverbindungen entfernt werden.
- **`RemoteMappingKernelManager.cull_interval`** – Das Intervall in Sekunden (Ganzzahl), in dem nach Kerneln im Leerlauf gesucht werden soll, die den Wert für das Cull-Timeout überschreiten.

PySpark Sitzungsparameter ändern

Ab Amazon EMR auf EKS-Version 6.9.0 können Sie in Amazon EMR Studio die mit einer PySpark Sitzung verknüpfte Spark-Konfiguration anpassen, indem Sie den `%%configure` magischen Befehl in der EMR-Notebookzelle ausführen.

Das folgende Beispiel zeigt eine Beispielnutzlast, mit der Sie Speicher, Kerne und andere Eigenschaften für den Spark-Treiber und Executor ändern können. Für die `conf`-Einstellungen können Sie jede Spark-Konfiguration konfigurieren, die in der Apache-Spark-Konfigurationsdokumentation erwähnt wird.

```
%%configure -f
{
  "driverMemory": "16G",
  "driverCores" 4,
  "executorMemory" : "32G"
  "executorCores": 2,
  "conf": {
     "spark.dynamicAllocation.maxExecutors" : 10,
     "spark.dynamicAllocation.minExecutors": 1
  }
}
```

Das folgende Beispiel zeigt eine Beispielnutzlast, die Sie verwenden können, um Dateien, PyFiles und JAR-Abhängigkeiten zu einer Spark-Laufzeit hinzuzufügen.

```
%%configure -f
{
  "files": "s3://test-bucket-emr-eks/sample_file.txt",
  "pyFiles": : "path-to-python-files",
  "jars" : "path-to-jars
}
```

Benutzerdefiniertes Kernel-Image mit interaktivem Endpunkt

Um sicherzustellen, dass Sie die richtigen Abhängigkeiten für Ihre Anwendung haben, wenn Sie interaktive Workloads von Amazon EMR Studio aus ausführen, können Sie Docker-Images für interaktive Endpunkte anpassen und benutzerdefinierte Basis-Kernel-Images ausführen. Um einen interaktiven Endpunkt zu erstellen und ihn mit einem benutzerdefinierten Docker-Image zu verbinden, führen Sie die folgenden Schritte aus.

> **Note**
>
> Sie können nur Basis-Images überschreiben. Sie können keine neuen Kernel-Image-Typen hinzufügen.

1. Erstellen und veröffentlichen Sie ein benutzerdefiniertes Docker-Image. Das Basis-Image enthält die Spark-Laufzeit und die Notebook-Kernel, die damit ausgeführt werden. Um das Image zu erstellen, können Sie die Schritte 1 bis 4 unter Wie passen Sie Docker-Images an ausführen. In Schritt 1 muss der Basis-Image-URI in Ihrer Docker-Datei `notebook-spark` anstelle von `spark` verwendet werden.

   ```
   ECR-registry-account.dkr.ecr.Region.amazonaws.com/notebook-spark/container-image-tag
   ```

   Weitere Informationen zur Auswahl AWS-Regionen und Speicherung von Image-Tags finden Sie unter. Wie wählt man einen Basis-Image-URI aus
2. Erstellen Sie einen interaktiven Endpunkt, der mit dem benutzerdefinierten Image verwendet werden kann.

   a. Erstellen Sie eine JSON Datei mit dem Namen `custom-image-managed-endpoint.json` und dem folgenden Inhalt. In diesem Beispiel wird Amazon-EMR-Version 6.9.0 verwendet.

      Example

      ```
      {
          "name": "endpoint-name",
          "virtualClusterId": "virtual-cluster-id",
          "type": "JUPYTER_ENTERPRISE_GATEWAY",
          "releaseLabel": "emr-6.9.0-latest",
          "executionRoleArn": "execution-role-arn",
          "configurationOverrides": {
              "applicationConfiguration": [
                  {
                      "classification": "jupyter-kernel-overrides",
                      "configurations": [
                          {
                              "classification": "python3",
                              "properties": {
      ```

```
                            "container-image": "123456789012.dkr.ecr.us-
west-2.amazonaws.com/custom-notebook-python:latest"
                        }
                    },
                    {
                        "classification": "spark-python-kubernetes",
                        "properties": {
                            "container-image": "123456789012.dkr.ecr.us-
west-2.amazonaws.com/custom-notebook-spark:latest"
                        }
                    }
                ]
            }
        ]
    }
}
```

b. Erstellen Sie einen interaktiven Endpunkt mit den in der JSON-Datei angegebenen Konfigurationen, wie das folgende Beispiel zeigt. Weitere Informationen finden Sie unter Erstellen Sie einen interaktiven Endpunkt mit dem `create-managed-endpoint`-Befehl.

```
aws emr-containers create-managed-endpoint --cli-input-json custom-image-
managed-endpoint.json
```

3. Stellen Sie über EMR Studio eine Verbindung zum interaktiven Endpunkt her. Weitere Informationen und auszuführende Schritte finden Sie unter Von Studio aus verbinden im Abschnitt Amazon EMR on EKS der AWS Workshop Studio-Dokumente.

Überwachen interaktiver Endpunkte

Mit Amazon EMR auf EKS-Version 6.10 und höher senden interaktive Endpunkte CloudWatch Amazon-Metriken zur Überwachung und Fehlerbehebung von Kernel-Lebenszyklusvorgängen aus. Metriken werden durch interaktive Clients wie EMR Studio oder selbst gehostete Jupyter Notebooks ausgelöst. Jedem Vorgang, der von interaktiven Endpunkten unterstützt wird, sind Metriken zugeordnet. Die Vorgänge werden als Dimensionen für jede Metrik modelliert, wie in der folgenden Tabelle dargestellt. Von interaktiven Endpunkten ausgegebene Metriken sind in Ihrem Konto unter einem benutzerdefinierten Namespace, EMRContainers, sichtbar.

Metrik	Beschreibung	Einheit
RequestCount	Gesamtzahl der Anfragen eines Vorgangs, die vom interaktiven Endpunkt verarbeitet wurden.	Anzahl
RequestLatency	Der Zeitpunkt, zu dem eine Anfrage am interaktiven Endpunkt eingetroffen ist und eine Antwort vom interaktiven Endpunkt gesendet wurde.	Millisekunde
4XXError	Wird ausgegeben, wenn eine Anforderung für einen Vorgang bei der Verarbeitung zu einem 4xx-Fehler führt.	Anzahl
5XXError	Wird ausgegeben, wenn eine Anforderung für einen Vorgang zu einem serversei tigen 5XXX-Fehler führt.	Anzahl
KernelLaunchErfolgreich	Gilt nur für die CreateKer nel Operation. Es gibt die Gesamtzahl der Kernelsta rts an, die bis einschließlich dieser Anforderung erfolgreich waren.	Anzahl
KernelLaunchFehlschlag	Gilt nur für die CreateKer nel Operation. Es gibt die Gesamtzahl der Fehler beim Starten des Kernels bis einschließlich dieser Anforderu ng an.	Anzahl

Jeder interaktiven Endpunktmetrik sind die folgenden Dimensionen zugeordnet:

- **`ManagedEndpointId`** – Bezeichner für den interaktiven Endpunkt
- **`OperationName`** – Der vom interaktiven Client ausgelöste Vorgang

Mögliche Werte für die **`OperationName`**-Dimension werden in der folgenden Tabelle aufgeführt:

`operationName`	Beschreibung des Vorgangs
`CreateKernel`	Fordert an, dass der interaktive Endpunkt einen Kernel startet.
`ListKernels`	Fordern Sie an, dass der interaktive Endpunkt die Kernel auflistet, die zuvor mit demselben Sitzungstoken gestartet wurden.
`GetKernel`	Fordern Sie an, dass der interaktive Endpunkt Details zu einem bestimmten Kernel erhält, der zuvor gestartet wurde.
`ConnectKernel`	Fordert den interaktiven Endpunkt auf, die Konnektivität zwischen dem Notebook-Client und dem Kernel herzustellen.
`ConfigureKernel`	Veröffentlichen Sie `%%configure magic request` auf einem Pyspark-Kernel.
`ListKernelSpecs`	Fordern Sie an, dass der interaktive Endpunkt die verfügbaren Kernel-Spezifikationen auflistet .
`GetKernelSpec`	Fordern Sie an, dass der interaktive Endpunkt die Kernel-Spezifikationen eines Kernels erhält, der zuvor gestartet wurde.
`GetKernelSpecResource`	Fordern Sie an, dass der interaktive Endpunkt spezifische Ressourcen erhält, die mit den

operationName	Beschreibung des Vorgangs
	zuvor gestarteten Kernel-Spezifikationen verknüpft sind.

Beispiele

Um auf die Gesamtzahl der Kernel zuzugreifen, die an einem bestimmten Tag für einen interaktiven Endpunkt gestartet wurden, gehen Sie wie folgt vor:

1. Wählen Sie den benutzerdefinierten Namespace aus: `EMRContainers`
2. Wählen Sie `ManagedEndpointId`, `OperationName – CreateKernel`
3. `RequestCount`-Metrik mit der Statistik `SUM` und dem Zeitraum `1 day` liefert alle Kernel-Startanfragen der letzten 24 Stunden.
4. KernelLaunchSuccess Die Metrik mit Statistik `SUM` und Zeitraum `1 day` liefert alle erfolgreichen Kernel-Startanfragen der letzten 24 Stunden.

Um auf die Anzahl der Kernelfehler für einen interaktiven Endpunkt an einem bestimmten Tag zuzugreifen:

1. Wählen Sie den benutzerdefinierten Namespace: EMRContainers
2. Wählen Sie `ManagedEndpointId`, `OperationName – CreateKernel`
3. `KernelLaunchFailure`-Metrik mit der Statistik `SUM` und dem Zeitraum `1 day` liefert alle fehlgeschlagenen Kernel-Startanfragen der letzten 24 Stunden. Sie können auch die `4XXError`- und `5XXError`-Metrik auswählen, um zu erfahren, welche Art von Kernelstartfehler aufgetreten ist.

Verwenden von selbst gehosteten Jupyter Notebooks

Sie können Jupyter oder JupyterLab Notebooks auf einer Amazon EC2 EC2-Instance oder in Ihrem eigenen Amazon EKS-Cluster als selbst gehostetes Jupyter-Notebook hosten und verwalten. Anschließend können Sie interaktive Workloads mit Ihren selbstgehosteten Jupyter-Notebooks ausführen. In den folgenden Abschnitten wird der Prozess zur Einrichtung und Bereitstellung eines selbst gehosteten Jupyter Notebooks auf einem Amazon-EKS-Cluster beschrieben.

Erstellen eines selbstgehosteten Jupyter Notebooks auf einem EKS-Cluster

- Eine Sicherheitsgruppe erstellen
- Einen interaktiven Endpunkt für Amazon EMR in EKS erstellen
- Abrufen der Gateway-Server-URL Ihres interaktiven Endpunkts
- Rufen Sie ein Authentifizierungstoken ab, um eine Verbindung zum interaktiven Endpunkt herzustellen
- Beispiel: Stellen Sie ein Notebook bereit JupyterLab
- Löschen Sie ein selbst gehostetes Jupyter Notebook

Eine Sicherheitsgruppe erstellen

Bevor Sie einen interaktiven Endpunkt erstellen und einen selbst gehosteten Jupyter oder ein selbst gehostetes Notebook ausführen können, müssen Sie eine Sicherheitsgruppe erstellen, um den Verkehr zwischen Ihrem JupyterLab Notebook und dem interaktiven Endpunkt zu kontrollieren. Informationen zur Verwendung der Amazon EC2-Konsole oder des Amazon EC2-SDK zum Erstellen der Sicherheitsgruppe finden Sie in den Schritten unter Sicherheitsgruppe erstellen im Amazon EC2 EC2-Benutzerhandbuch. Sie sollten die Sicherheitsgruppe in der VPC erstellen, in der Sie Ihren Notebook-Server bereitstellen möchten.

Um dem Beispiel in diesem Handbuch zu folgen, verwenden Sie dieselbe VPC wie Ihr Amazon-EKS-Cluster. Wenn Sie Ihr Notebook in einer VPC hosten möchten, die sich von der VPC für Ihren Amazon-EKS-Cluster unterscheidet, müssen Sie möglicherweise eine Peering-Verbindung zwischen diesen beiden VPCs herstellen. Schritte zum Erstellen einer Peering-Verbindung zwischen zwei VPCs finden Sie unter Erstellen einer VPC-Peering-Verbindung im Handbuch zu ersten Schritten mit Amazon VPC.

Sie benötigen die ID für die Sicherheitsgruppe, um im nächsten Schritt einen interaktiven Endpunkt für Amazon EMR in EKS zu erstellen.

Einen interaktiven Endpunkt für Amazon EMR in EKS erstellen

Nachdem Sie die Sicherheitsgruppe für Ihr Notebook erstellt haben, gehen Sie wie unter Einen interaktiven Endpunkt für Ihren virtuellen Cluster erstellen beschrieben vor, um einen interaktiven Endpunkt zu erstellen. Sie müssen die Sicherheitsgruppen-ID angeben, die Sie für Ihr Notebook unter Eine Sicherheitsgruppe erstellen erstellt haben.

Geben Sie in den folgenden Einstellungen zur Änderung der Konfiguration die Sicherheits-ID anstelle von *your-notebook-security-group-id* ein:

```
--configuration-overrides '{
    "applicationConfiguration": [
        {
            "classification": "endpoint-configuration",
            "properties": {
                "notebook-security-group-id": "your-notebook-security-group-id"
            }
        }
    ],
    "monitoringConfiguration": {
    ...'
```

Abrufen der Gateway-Server-URL Ihres interaktiven Endpunkts

Nachdem Sie einen interaktiven Endpunkt erstellt haben, rufen Sie die Gateway-Server-URL mit dem `describe-managed-endpoint`-Befehl in AWS CLI ab. Sie benötigen diese URL, um Ihr Notebook mit dem Endpunkt zu verbinden. Die Gateway-Server-URL ist ein privater Endpunkt.

```
aws emr-containers describe-managed-endpoint \
--region region \
--virtual-cluster-id virtualClusterId \
--id endpointId
```

Der Endpunkt befindet sich zunächst im Status CREATING. Nach einigen Minuten wechselt er in den ACTIVE-Status. Wenn der Endpunkt ACTIVE ist, kann er verwendet werden.

Notieren Sie sich das `serverUrl`-Attribut, das der `aws emr-containers describe-managed-endpoint`-Befehl vom aktiven Endpunkt zurückgibt. Sie benötigen diese URL, um Ihr Notebook mit dem Endpunkt zu verbinden, wenn Sie Ihren selbst gehosteten Jupyter oder Ihr selbst gehostetes Notebook bereitstellen. JupyterLab

Rufen Sie ein Authentifizierungstoken ab, um eine Verbindung zum interaktiven Endpunkt herzustellen

Um von einem Jupyter oder JupyterLab Notebook aus eine Verbindung zu einem interaktiven Endpunkt herzustellen, müssen Sie mit der API ein Sitzungstoken generieren.

`GetManagedEndpointSessionCredentials` Das Token dient als Authentifizierungsnachweis für die Verbindung zum interaktiven Endpunktserver.

Der folgende Befehl wird anhand eines Ausgabebeispiels weiter unten ausführlicher erklärt.

```
aws emr-containers get-managed-endpoint-session-credentials \
--endpoint-identifier endpointArn \
--virtual-cluster-identifier virtualClusterArn \
--execution-role-arn executionRoleArn \
--credential-type "TOKEN" \
--duration-in-seconds durationInSeconds \
--region region
```

endpointArn

Der ARN Ihres Endpunkts. Sie finden den ARN im Ergebnis eines `describe-managed-endpoint`-Aufrufs.

virtualClusterArn

Die ARN des virtuellen Clusters.

executionRoleArn

Die ARN der Ausführungsrolle.

durationInSeconds

Die Dauer in Sekunden, für die das Token gültig ist. Die Standarddauer beträgt 15 Minuten (`900`) und die Höchstdauer 12 Stunden (`43200`).

region

Dieselbe Region wie Ihr Endpunkt.

Die Ausgabe sollte etwa wie das folgende Beispiel aussehen. Notieren Sie sich den *session-token* Wert, den Sie verwenden werden, wenn Sie Ihren selbst gehosteten Jupyter oder Ihr selbst gehostetes Notebook bereitstellen. JupyterLab

```
{
    "id": "credentialsId",
    "credentials": {
        "token": "session-token"
```

```
    },
    "expiresAt": "2022-07-05T17:49:38Z"
}
```

Beispiel: Stellen Sie ein Notebook bereit JupyterLab

Nachdem Sie die obigen Schritte abgeschlossen haben, können Sie dieses Beispielverfahren ausprobieren, um ein JupyterLab Notebook mit Ihrem interaktiven Endpunkt im Amazon EKS-Cluster bereitzustellen.

1. Erstellen Sie einen Namespace, um den Notebook-Server auszuführen.
2. Erstellen Sie lokal eine Datei `notebook.yaml` und den folgenden Inhalten. Der Inhalt der Datei wird im Folgenden beschrieben.

```
apiVersion: v1
kind: Pod
metadata:
  name: jupyter-notebook
  namespace: namespace
spec:
  containers:
  - name: minimal-notebook
    image: jupyter/all-spark-notebook:lab-3.1.4 # open source image
    ports:
    - containerPort: 8888
    command: ["start-notebook.sh"]
    args: ["--LabApp.token=''"]
    env:
    - name: JUPYTER_ENABLE_LAB
      value: "yes"
    - name: KERNEL_LAUNCH_TIMEOUT
      value: "400"
    - name: JUPYTER_GATEWAY_URL
      value: "serverUrl"
    - name: JUPYTER_GATEWAY_VALIDATE_CERT
      value: "false"
    - name: JUPYTER_GATEWAY_AUTH_TOKEN
      value: "session-token"
```

Wenn Sie das Jupyter-Notebook in einem reinen Fargate-Cluster bereitstellen, kennzeichnen Sie den Jupyter-Pod mit einer `role`-Kennzeichnung, wie im folgenden Beispiel gezeigt:

```
...
metadata:
  name: jupyter-notebook
  namespace: default
  labels:
    role: example-role-name-label
spec:
            ...
```

namespace

Der Kubernetes-Namespace, in dem das Notebook bereitgestellt wird.

serverUrl

Das `serverUrl`-Attribut, in dem der `describe-managed-endpoint`-Befehl in Abrufen der Gateway-Server-URL Ihres interaktiven Endpunkts zurückgegeben wurde.

session-token

Das `session-token`-Attribut, in dem der `get-managed-endpoint-session-credentials`-Befehl in Rufen Sie ein Authentifizierungstoken ab, um eine Verbindung zum interaktiven Endpunkt herzustellen zurückgegeben wurde.

KERNEL_LAUNCH_TIMEOUT

Die Zeit in Sekunden, die der interaktive Endpunkt darauf wartet, dass der Kernel den RUNNING-Status erreicht. Stellen Sie sicher, dass genügend Zeit bis zum Abschluss des Kernelstarts vergangen ist, indem Sie das Timeout für den Kernelstart auf einen geeigneten Wert setzen (maximal 400 Sekunden).

KERNEL_EXTRA_SPARK_OPTS

Optional können Sie zusätzliche Spark-Konfigurationen für die Spark-Kernel übergeben. Legen Sie diese Umgebungsvariable mit den Werten als Spark-Konfigurationseigenschaft fest, wie im folgenden Beispiel gezeigt:

```
- name: KERNEL_EXTRA_SPARK_OPTS
  value: "--conf spark.driver.cores=2
          --conf spark.driver.memory=2G
          --conf spark.executor.instances=2
          --conf spark.executor.cores=2
```

```
            --conf spark.executor.memory=2G
            --conf spark.dynamicAllocation.enabled=true
            --conf spark.dynamicAllocation.shuffleTracking.enabled=true
            --conf spark.dynamicAllocation.minExecutors=1
            --conf spark.dynamicAllocation.maxExecutors=5
            --conf spark.dynamicAllocation.initialExecutors=1
            "
```

3. Stellen Sie die Pod-Spezifikation in Ihrem Amazon-EKS-Cluster bereit:

```
kubectl apply -f notebook.yaml -n namespace
```

Dadurch wird ein minimales JupyterLab Notizbuch gestartet, das mit Ihrem interaktiven Endpunkt Amazon EMR auf EKS verbunden ist. Warten Sie, bis der Pod RUNNING ist. Sie können den Status mit dem folgenden Befehl überprüfen:

```
kubectl get pod jupyter-notebook -n namespace
```

Wenn der Pod bereit ist, gibt der `get pod`-Befehl eine Ausgabe zurück, die der folgenden ähnelt:

```
NAME              READY   STATUS    RESTARTS   AGE
jupyter-notebook  1/1     Running   0          46s
```

4. Ordnen Sie die Notebook-Sicherheitsgruppe dem Knoten zu, für den das Notebook geplant ist.

   a. Identifizieren Sie zunächst mit dem `describe pod`-Befehl den Knoten, für den der `jupyter-notebook`-Pod geplant ist.

   ```
   kubectl describe pod jupyter-notebook -n namespace
   ```

   b. Öffnen Sie die Amazon-EKS-Konsole unter https://console.aws.amazon.com/eks/home#/clusters.

   c. Navigieren Sie zur Registerkarte Datenverarbeitung für Ihren Amazon-EKS-Cluster und wählen Sie den durch den `describe pod`-Befehl identifizierten Knoten aus. Wählen Sie die Instance-ID für den Knoten aus.

   d. Wählen Sie im Menü Aktionen die Option Sicherheit > Sicherheitsgruppen ändern aus, um die Sicherheitsgruppe anzuhängen, die Sie in Eine Sicherheitsgruppe erstellen erstellt haben.

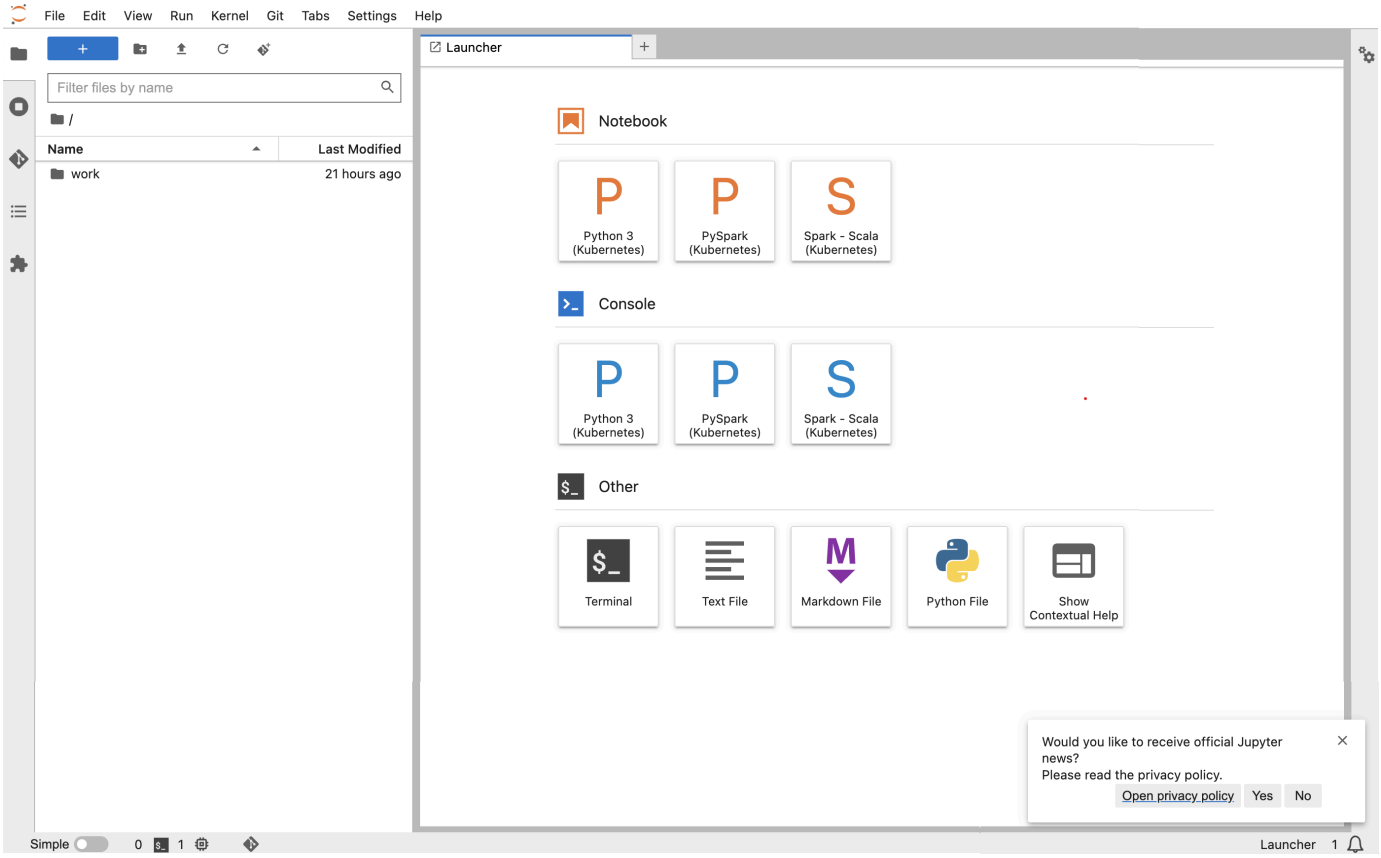
e. Wenn Sie den Jupyter-Notebook-Pod bereitstellen AWS Fargate, erstellen Sie einen SecurityGroupPolicy, der auf den Jupyter-Notebook-Pod angewendet werden soll, mit der Rollenbezeichnung:

```
cat >my-security-group-policy.yaml <<EOF
apiVersion: vpcresources.k8s.aws/v1beta1
kind: SecurityGroupPolicy
metadata:
  name: example-security-group-policy-name
  namespace: default
spec:
  podSelector:
    matchLabels:
      role: example-role-name-label
  securityGroups:
    groupIds:
      - your-notebook-security-group-id
EOF
```

5. Führen Sie nun einen Port-Forward durch, sodass Sie lokal auf die Schnittstelle zugreifen können: JupyterLab

```
kubectl port-forward jupyter-notebook 8888:8888 -n namespace
```

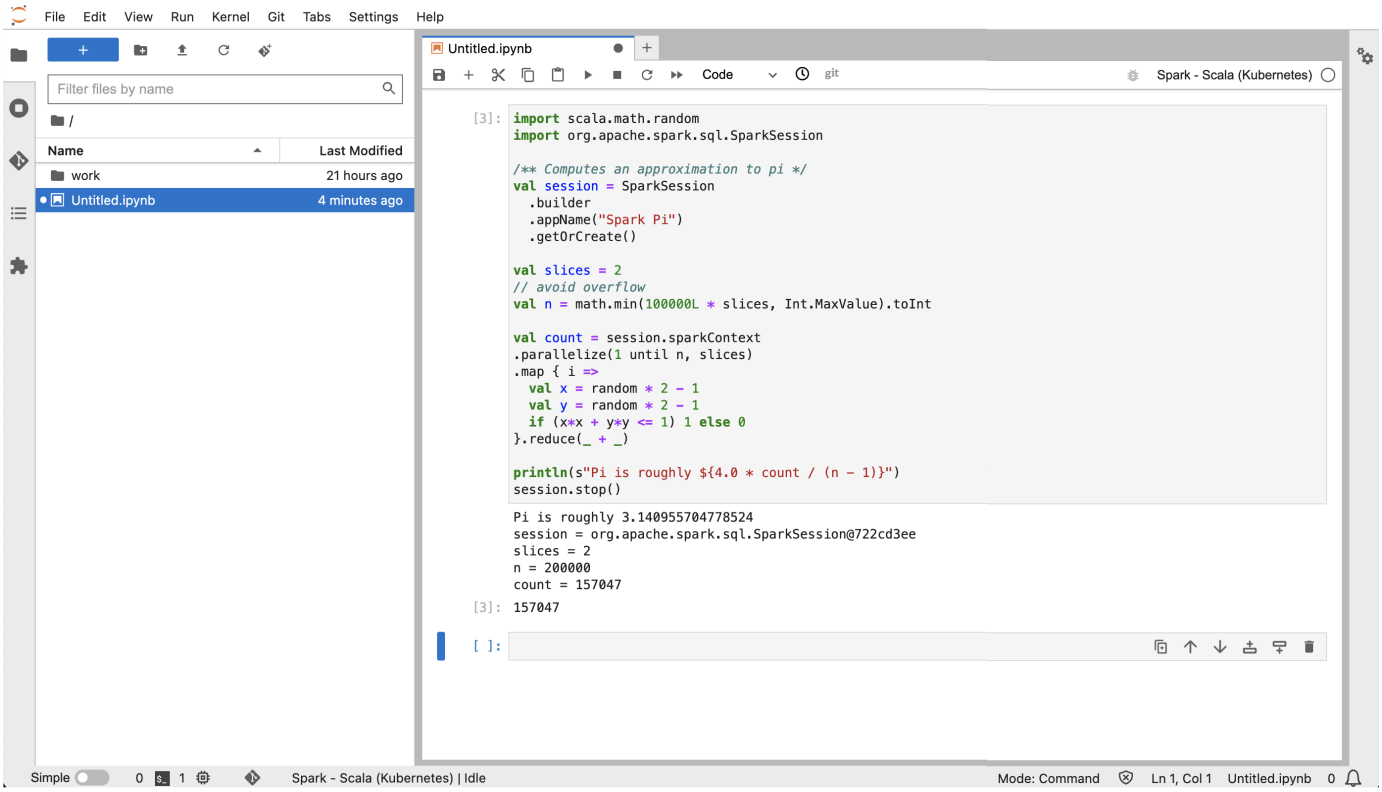
Sobald das läuft, navigieren Sie zu Ihrem lokalen Browser und besuchen `localhost:8888` Sie die JupyterLab Benutzeroberfläche:



6. Erstellen Sie von aus JupyterLab ein neues Scala-Notizbuch. Hier ist ein Beispielcodeausschnitt, den Sie ausführen können, um den Wert von Pi zu approximieren:

```
import scala.math.random
import org.apache.spark.sql.SparkSession

/** Computes an approximation to pi */
val session = SparkSession
  .builder
  .appName("Spark Pi")
  .getOrCreate()

val slices = 2
// avoid overflow
val n = math.min(100000L * slices, Int.MaxValue).toInt

val count = session.sparkContext
.parallelize(1 until n, slices)
.map { i =>
  val x = random * 2 - 1
  val y = random * 2 - 1
```

```
  if (x*x + y*y <= 1) 1 else 0
}.reduce(_ + _)

println(s"Pi is roughly ${4.0 * count / (n - 1)}")
session.stop()
```



Löschen Sie ein selbst gehostetes Jupyter Notebook

Wenn Sie bereit sind, Ihr selbst gehostetes Notebook zu löschen, können Sie auch den interaktiven Endpunkt und die Sicherheitsgruppe löschen. Führen Sie die Aktionen in der folgenden Reihenfolge aus:

1. Verwenden Sie den folgenden -Befehl, um den `jupyter-notebook` Pod zu löschen:

   ```
   kubectl delete pod jupyter-notebook -n namespace
   ```

2. Löschen Sie dann Ihren interaktiven Endpunkt mit dem `delete-managed-endpoint`-Befehl. Schritte zum Löschen eines interaktiven Endpunkts finden Sie unter Löschen eines interaktiven Endpunkts. Zu Beginn befindet sich Ihr Endpunkt im TERMINATING-Status. Sobald alle Ressourcen bereinigt wurden, wechselt er in den TERMINATED-Status.

3. Wenn Sie nicht vorhaben, die Notebook-Sicherheitsgruppe, in der Sie Eine Sicherheitsgruppe erstellen erstellt haben, für andere Jupyter-Notebook-Bereitstellungen zu verwenden, können Sie sie löschen. Weitere Informationen finden Sie unter Löschen einer Sicherheitsgruppe im Amazon-EC2-Benutzerhandbuch.

Andere Vorgänge auf einem interaktiven Endpunkt

Dieses Thema behandelt die unterstützten Operationen auf einem anderen interaktiven Endpunkt als `create-managed-endpoint`.

Rufen Sie interaktive Endpunktdetails ab

Nachdem Sie einen interaktiven Endpunkt erstellt haben, können Sie dessen Details mit dem Befehl abrufen. `describe-managed-endpoint` AWS CLI Fügen Sie Ihre eigenen Werte für *`managed-endpoint-id`*, *`virtual-cluster-id`* und *`region`* ein:

```
aws emr-containers describe-managed-endpoint --id managed-endpoint-id \
 --virtual-cluster-id virtual-cluster-id --region region
```

Die Ausgabe sieht mit dem angegebenen Endpunkt, wie ARN, ID und Name, in etwa wie folgt aus.

```
{
   "id": "as3ys2xxxxxxx",
   "name": "endpoint-name",
    "arn": "arn:aws:emr-containers:us-east-1:1828xxxxxxxx:/virtualclusters/
lbhl6kwwyoxxxxxxxxxxxxxxx/endpoints/as3ysxxxxxxxx",
    "virtualClusterId": "lbhl6kwwyoxxxxxxxxxxxxxxx",
    "type": "JUPYTER_ENTERPRISE_GATEWAY",
    "state": "ACTIVE",
    "releaseLabel": "emr-6.9.0-latest",
   "executionRoleArn": "arn:aws:iam::1828xxxxxxxx:role/RoleName",
    "certificateAuthority": {
        "certificateArn": "arn:aws:acm:us-east-1:1828xxxxxxxx:certificate/zzzzzzzz-
e59b-4ed0-aaaa-bbbbbbbbbbbb",
        "certificateData": "certificate-data"
    },
    "configurationOverrides": {
        "applicationConfiguration": [
            {
                "classification": "spark-defaults",
                "properties": {
```

```
                    "spark.driver.memory": "8G"
                }
            }
        ],
        "monitoringConfiguration": {
            "persistentAppUI": "ENABLED",
            "cloudWatchMonitoringConfiguration": {
                "logGroupName": "log-group-name",
                "logStreamNamePrefix": "log-stream-name-prefix"
            },
            "s3MonitoringConfiguration": {
                "logUri": "s3-bucket-name"
            }
        }
    },
   "serverUrl": "https://internal-k8s-namespace-ingressa-aaaaaaaaaa-
zzzzzzzzzz.us-east-1.elb.amazonaws.com:18888 (https://internal-k8s-nspluto-
ingressa-51e860abbd-1620715833.us-east-1.elb.amazonaws.com:18888/)",
    "createdAt": "2022-09-19T12:37:49+00:00",
    "securityGroup": "sg-aaaaaaaaaaaaaa",
    "subnetIds": [
        "subnet-11111111111",
        "subnet-22222222222",
        "subnet-33333333333"
    ],
    "stateDetails": "Endpoint created successfully. It took 3 Minutes 15 Seconds",
    "tags": {}
 }
```

Listen Sie alle interaktiven Endpunkte auf, die einem virtuellen Cluster zugeordnet sind

Verwenden Sie den `list-managed-endpoints` AWS CLI Befehl, um eine Liste aller interaktiven Endpunkte abzurufen, die einem bestimmten virtuellen Cluster zugeordnet sind. Ersetzen Sie `virtual-cluster-id` mit dem ID Ihres virtuellen Clusters.

```
aws emr-containers list-managed-endpoints --virtual-cluster-id virtual-cluster-id
```

Die Ausgabe des `list-managed-endpoint` Befehls ist unten dargestellt:

```
{
    "endpoints": [{
```

```
        "id": "as3ys2xxxxxxx",
        "name": "endpoint-name",
        "arn": "arn:aws:emr-containers:us-east-1:1828xxxxxxxx:/virtualclusters/
lbhl6kwwyoxxxxxxxxxxxxxx/endpoints/as3ysxxxxxxxx",
        "virtualClusterId": "lbhl6kwwyoxxxxxxxxxxxxxx",
        "type": "JUPYTER_ENTERPRISE_GATEWAY",
        "state": "ACTIVE",
        "releaseLabel": "emr-6.9.0-latest",
        "executionRoleArn": "arn:aws:iam::1828xxxxxxxx:role/RoleName",
        "certificateAuthority": {
            "certificateArn": "arn:aws:acm:us-east-1:1828xxxxxxxx:certificate/zzzzzzzz-
e59b-4ed0-aaaa-bbbbbbbbbbbb",
            "certificateData": "certificate-data"
        },
        "configurationOverrides": {
            "applicationConfiguration": [{
                "classification": "spark-defaults",
                "properties": {
                    "spark.driver.memory": "8G"
                }
            }],
            "monitoringConfiguration": {
                "persistentAppUI": "ENABLED",
                "cloudWatchMonitoringConfiguration": {
                    "logGroupName": "log-group-name",
                    "logStreamNamePrefix": "log-stream-name-prefix"
                },
                "s3MonitoringConfiguration": {
                    "logUri": "s3-bucket-name"
                }
            }
        },
        "serverUrl": "https://internal-k8s-namespace-ingressa-aaaaaaaaaa-
zzzzzzzzzz.us-east-1.elb.amazonaws.com:18888 (https://internal-k8s-nspluto-
ingressa-51e860abbd-1620715833.us-east-1.elb.amazonaws.com:18888/)",
        "createdAt": "2022-09-19T12:37:49+00:00",
        "securityGroup": "sg-aaaaaaaaaaaaa",
        "subnetIds": [
            "subnet-11111111111",
            "subnet-22222222222",
            "subnet-33333333333"
        ],
        "stateDetails": "Endpoint created successfully. It took 3 Minutes 15 Seconds",
        "tags": {}
```

```
    }]
}
```

Löschen eines interaktiven Endpunkts

Verwenden Sie den `delete-managed-endpoint` AWS CLI Befehl, um einen interaktiven Endpunkt zu löschen, der mit einem virtuellen Amazon EMR auf einem EKS-Cluster verknüpft ist. Wenn Sie einen interaktiven Endpunkt löschen, entfernt Amazon EMR in EKS die Standardsicherheitsgruppen, die für diesen Endpunkt erstellt wurden.

Geben Sie Werte für die folgenden Parameter für den Befehl an:

- --id: Die Kennung des interaktiven Endpunkts, den Sie löschen möchten.
- --virtual-cluster-id – Die ID des virtuellen Clusters, der dem interaktiven Endpunkt zugeordnet ist, den Sie löschen möchten. Dies ist dieselbe virtuelle Cluster-ID, die bei der Erstellung des interaktiven Endpunkts angegeben wurde.

```
aws emr-containers delete-managed-endpoint --id managed-endpoint-id --virtual-cluster-
id virtual-cluster-id
```

Der Befehl gibt eine Ausgabe ähnlich der folgenden zurück, um zu bestätigen, dass Sie den interaktiven Endpunkt gelöscht haben:

```
{
    "id":"8gai4l4exxxxx",
    "virtualClusterId":"0b0qvauoy3ch1nqodxxxxxxxx"
}
```

Überwachen von Aufträgen

Themen

Überwachen von Aufträgen mit Amazon CloudWatch Events

Amazon EMR in EKS gibt Ereignisse aus, wenn sich der Status einer Aufgabenausführung ändert. Jedes Ereignis enthält Informationen wie Datum und Uhrzeit des Auftretens des Ereignisses sowie weitere Details zu dem Ereignis, z. B. die virtuelle Cluster-ID und die ID der betroffenen Aufgabenausführung.

Sie können Ereignisse verwenden, um die Aktivität und den Zustand von Aufträge zu verfolgen, die Sie auf einem virtuellen Cluster ausführen. Sie können Amazon CloudWatch Events auch verwenden, um eine Aktion zu definieren, die ausgeführt werden soll, wenn eine Auftragsausführung ein Ereignis generiert, das einem von Ihnen angegebenen Muster entspricht. Ereignisse sind nützlich, um ein bestimmtes Ereignis während des Lebenszyklus einer Aufgabenausführung zu überwachen. Sie können beispielsweise überwachen, wann sich der Status einer ausgeführten Aufgabe von `submitted` zu `running` ändert. Weitere Informationen zu - CloudWatch Ereignissen finden Sie im Amazon- EventBridge Benutzerhandbuch.

Die folgende Tabelle enthält Ereignisse von Amazon EMR in EKS, zusammen mit den Änderungen hinsichtlich des Zustands oder des Status, auf den das Ereignis hinweist, sowie den Schweregrad des Ereignisses als auch Ereignismeldungen. Jedes Ereignis wird als JSON-Objekt dargestellt, das automatisch an den Ereignis-Stream gesendet wird. Das JSON-Objekt enthält weitere Details zum Ereignis. Das JSON-Objekt ist besonders wichtig, wenn Sie Regeln für die Ereignisverarbeitung mithilfe von CloudWatch Ereignissen einrichten, da Regeln versuchen, Muster im JSON-Objekt abzugleichen. Weitere Informationen finden Sie unter Amazon- EventBridge Ereignismuster und Amazon EMR in EKS Events im Amazon- EventBridge Benutzerhandbuch.

Aufgabenausführungsstatus-Änderungsereignisse

Status	Schweregrad	Fehlermeldung
SUBMITTED	INFO	Auftragsausführung *JobRunId* (*JobRunName*) wurde *VirtualClusterId* zum *Zeitpunkt* UTC erfolgreich an den virtuellen Cluster übermittelt.
AUSFÜHREN	INFO	Die Auftragsausführung *JobRunId* (*JobRunName*) im virtuellen Cluster *VirtualClusterId* begann um *Uhrzeit* .
COMPLETED	INFO	Auftragsausführung *jobRunId* (*JobRunName*) im virtuellen Cluster wurde zum *Zeitpunkt VirtualClusterId* abgeschlossen. Der Aufgabenlauf begann um *Zeit* mit der Ausführung und dauerte *Num* Minuten bis zum Abschluss.
CANCELLED	WARN	Die Beendigungsanforderung für die Auftragsausführung *JobRunId* (*JobRunName*) im virtuellen Cluster war *VirtualClusterId* zur *Zeit* erfolgreich und die Auftragsausführung wurde jetzt abgebrochen.
FEHLGESCHLAGEN	ERROR	Die Auftragsausführung *JobRunId* (*JobRunName*) im virtuellen Cluster *ist zum Zeitpunkt VirtualClusterId* fehlgeschlagen.

Automatisieren von Amazon EMR in EKS mit - CloudWatch Ereignissen

Sie können Amazon CloudWatch Events verwenden, um Ihre AWS Services zu automatisieren und auf Systemereignisse wie Probleme mit der Anwendungsverfügbarkeit oder Ressourcenänderungen

zu reagieren. Ereignisse von - AWS Services werden an CloudWatch Ereignisse nahezu in Echtzeit übermittelt. 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. Die folgenden Aktionen können beispielsweise automatisch ausgelöst werden:

- Aufrufen einer - AWS Lambda Funktion
- Aufrufen eines Amazon EC2 Run Command
- Weiterleiten des Ereignisses an Amazon Kinesis Data Streams
- Aktivieren eines - AWS Step Functions Zustandsautomaten
- Benachrichtigen eines Amazon Simple Notification Service (SNS)-Themas oder einer Amazon Simple Queue Service (SQS)-Warteschlange

Einige Beispiele für die Verwendung von - CloudWatch Ereignissen mit Amazon EMR in EKS sind die folgenden:

- Aktivieren einer Lambda-Funktion, wenn ein Aufgabenlauf erfolgreich ist
- Benachrichtigen eines Amazon-SNS-Themas, wenn ein Aufgabenlauf fehlschlägt

CloudWatch Ereignisse für „`detail-type:`“ „`EMR Job Run State Change`“ werden von Amazon EMR in EKS für `SUBMITTED`-, -`CANCELLED`, `RUNNING`- `FAILED` und -`COMPLETED`Statusänderungen generiert.

Beispiel: Einrichten einer Regel, die Lambda aufruft

Führen Sie die folgenden Schritte aus, um eine CloudWatch Ereignisregel einzurichten, die Lambda aufruft, wenn ein Ereignis „Änderung des Ausführungsstatus von EMR-Aufträgen“ vorliegt.

```
aws events put-rule \
--name cwe-test \
--event-pattern '{"detail-type": ["EMR Job Run State Change"]}'
```

Fügen Sie die Lambda-Funktion, die Sie besitzen, als neues Ziel hinzu und erteilen Sie CloudWatch Events die Berechtigung, die Lambda-Funktion wie folgt aufzurufen. Ersetzen Sie *123456789012* durch Ihre Konto-ID.

```
aws events put-targets \
--rule cwe-test \
--targets Id=1,Arn=arn:aws:lambda:us-east-1:123456789012:function:MyFunction
```

```
aws lambda add-permission \
--function-name MyFunction \
--statement-id MyId \
--action 'lambda:InvokeFunction' \
--principal events.amazonaws.com
```

> **Note**
>
> Sie können kein Programm schreiben, dass von der Reihenfolge oder dem Vorhandensein von Benachrichtigungsereignissen abhängig ist, da Benachrichtigungen möglicherweise nicht der Reihe nach erfolgen oder fehlen. Ereignisse werden auf bestmögliche Weise ausgegeben.

Überwachen des Treiber-Pods des Auftrags mit einer Wiederholungsrichtlinie mithilfe von Amazon CloudWatch Events

Mithilfe von CloudWatch Ereignissen können Sie Treiber-Pods überwachen, die in Aufträgen mit Wiederholungsrichtlinien erstellt wurden. Weitere Informationen finden Sie unter Überwachen eines Auftrags mit einer Wiederholungsrichtlinie in diesem Handbuch.

Verwaltung virtueller Cluster

Ein virtueller Cluster ist ein Kubernetes-Namespace, mit dem Amazon EMR registriert ist. Sie können virtuelle Cluster erstellen, beschreiben, auflisten und löschen. Cluster verbrauchen keine zusätzlichen Ressourcen in Ihrem System. Ein einzelner virtueller Cluster wird einem einzelnen Kubernetes-Namespace zugeordnet. Auf der Grundlage dieser Beziehung können Sie virtuelle Cluster auf die gleiche Weise modellieren wie Kubernetes-Namespaces, damit sie Ihren Anforderungen entsprechen. Mögliche Anwendungsfälle finden Sie in der Dokumentation Konzepte-Übersicht für Kubernetes.

Um Amazon EMR mit einem Kubernetes-Namespace auf einem Amazon-EKS-Cluster zu registrieren, benötigen Sie den Namen des EKS-Clusters und den Namespace, der für die Ausführung Ihres Workloads eingerichtet wurde. Diese registrierten Cluster in Amazon EMR werden als virtuelle Cluster bezeichnet, da sie keine physische Rechenleistung oder Speicherung verwalten, sondern auf einen Kubernetes-Namespace verweisen, in dem Ihr Workload geplant ist.

> **Note**
>
> Bevor Sie einen virtuellen Cluster erstellen, müssen Sie zunächst die Schritte 1-8 unter Einrichten von Amazon EMR in EKS ausführen.

Themen

Erstellen eines virtuellen Clusters

Führen Sie den folgenden Befehl aus, um einen virtuellen Cluster zu erstellen, indem Sie Amazon EMR mit einem Namespace auf einem EKS-Cluster registrieren. Ersetzen Sie *virtual_cluster_name* durch einen Namen, den Sie für Ihren virtuellen Cluster angeben.

Ersetzen Sie *eks_cluster_name* durch den Namen Ihres EKS Clusters. Ersetzen Sie *namespace_name* durch den Namespace, mit dem Sie Amazon EMR registrieren möchten.

```
aws emr-containers create-virtual-cluster \
--name virtual_cluster_name \
--container-provider '{
    "id": "eks_cluster_name",
    "type": "EKS",
    "info": {
        "eksInfo": {
            "namespace": "namespace_name"
        }
    }
}'
```

Alternativ können Sie eine JSON-Datei erstellen, die die erforderlichen Parameter für den virtuellen Cluster enthält, wie das folgende Beispiel zeigt.

```
{
    "name": "virtual_cluster_name",
    "containerProvider": {
        "type": "EKS",
        "id": "eks_cluster_name",
        "info": {
            "eksInfo": {
                "namespace": "namespace_name"
            }
        }
    }
}
```

Führen Sie dann den folgenden `create-virtual-cluster`-Befehl mit dem Pfad zur JSON-Datei aus.

```
aws emr-containers create-virtual-cluster \
--cli-input-json file://./create-virtual-cluster-request.json
```

> **Note**
>
> Um die erfolgreiche Erstellung eines virtuellen Clusters zu überprüfen, zeigen Sie den Status der virtuellen Cluster an, indem Sie den Befehl `list-virtual-clusters` ausführen oder die Seite Virtuelle Cluster in der Amazon-EMR-Konsole aufrufen.

Virtuelle Cluster auflisten

Führen Sie den folgenden Befehl aus, um den Status virtueller Cluster anzuzeigen.

```
aws emr-containers list-virtual-clusters
```

Einen virtuellen Cluster beschreiben

Führen Sie den folgenden Befehl aus, um weitere Informationen zu einem virtuellen Cluster wie Namespace, Status und Registrierungsdatum abzurufen. Ersetzen Sie *123456* durch Ihre virtuelle Cluster-ID.

```
aws emr-containers describe-virtual-cluster --id 123456
```

Einen virtuellen Cluster löschen

Führen Sie den folgenden Befehl aus, um einen virtuellen Cluster zu löschen. Ersetzen Sie *123456* durch Ihre virtuelle Cluster-ID.

```
aws emr-containers delete-virtual-cluster --id 123456
```

Status des virtuellen Clusters

Die folgende Tabelle beschreibt die vier möglichen Zustände eines virtuellen Clusters.

State	Beschreibung
RUNNING	Der virtuelle Cluster befindet sich im RUNNING Status.

State	Beschreibung
TERMINATING	Die angeforderte Beendigung des virtuellen Clusters ist im Gange.
TERMINATED	Die angeforderte Beendigung ist abgeschlossen.
ARRESTED	Die angeforderte Beendigung ist aufgrund unzureichender Berechtigungen fehlgeschlagen.

Tutorials für Amazon EMR in EKS

In diesem Abschnitt werden allgemeine Anwendungsfälle beschrieben, wenn Sie mit Anwendungen von Amazon EMR in EKS arbeiten.

Themen

Verwenden von Delta Lake mit Amazon EMR in EKS

So nutzen Sie Delta Lake mit Amazon EMR in EKS-Anwendungen

1. Wenn Sie einen Auftrag starten, der ausgeführt wird, um einen Spark-Auftrag in der Anwendungskonfiguration einzureichen, fügen Sie die Delta-Lake-JAR-Dateien hinzu:

```
--job-driver '{"sparkSubmitJobDriver" : {
      "sparkSubmitParameters" : "--jars local:///usr/share/aws/delta/lib/delta-
core.jar,local:///usr/share/aws/delta/lib/delta-storage.jar,local:///usr/share/aws/
delta/lib/delta-storage-s3-dynamodb.jar"}}'
```

2. Fügen Sie die zusätzliche Delta-Lake-Konfiguration hinzu und verwenden Sie AWS Glue Data Catalog als Ihren Metastore.

```
--configuration-overrides '{
        "applicationConfiguration": [
        {
```

```
          "classification" : "spark-defaults",
          "properties" : {
            "spark.sql.extensions" : "io.delta.sql.DeltaSparkSessionExtension",

 "spark.sql.catalog.spark_catalog":"org.apache.spark.sql.delta.catalog.DeltaCatalog",
"spark.hadoop.hive.metastore.client.factory.class":"com.amazonaws.glue.catalog.metastore.AW
           }
        }]}'
```

Apache Iceberg mit Amazon EMR in EKS verwenden

So verwenden Sie Apache Iceberg mit Amazon EMR in EKS-Anwendungen

1. Wenn Sie einen Auftrag starten, der ausgeführt wird, um einen Spark-Auftrag in der Anwendungskonfiguration einzureichen, fügen Sie die Iceberg-Spark-Laufzeit-JAR-Dateien hinzu:

   ```
   --job-driver '{"sparkSubmitJobDriver" : {"sparkSubmitParameters" : "--jars
    local:///usr/share/aws/iceberg/lib/iceberg-spark3-runtime.jar"}}'
   ```

2. Fügen Sie die zusätzliche Iceberg-Konfiguration hinzu:

   ```
   --configuration-overrides '{
       "applicationConfiguration": [
       "classification" : "spark-defaults",
       "properties" : {
           "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.sql.catalog.dev.io-impl": "org.apache.iceberg.aws.s3.S3FileIO"
           }
       ]
   }'
   ```

Weitere Informationen zu den Release-Versionen von EMR von Apache Iceberg finden Sie in der Iceberg-Versionshistorie.

Verwenden von PyFlink

Amazon EMR in EKS-Versionen 6.15.0 und höher unterstützen PyFlink. Wenn Sie bereits über ein PyFlink Skript verfügen, können Sie einen der folgenden Schritte ausführen:

- Erstellen Sie ein benutzerdefiniertes Image, in dem Ihr PyFlink Skript enthalten ist.
- Hochladen Ihres Skripts an einen Amazon S3-Speicherort

Wenn Sie noch kein Skript haben, können Sie das folgende Beispiel verwenden, um einen PyFlink Auftrag zu starten. In diesem Beispiel wird das Skript von S3 abgerufen. Wenn Sie ein benutzerdefiniertes Image verwenden, in dem Ihr Skript bereits im Image enthalten ist, müssen Sie den Skriptpfad auf den Speicherort des Skripts aktualisieren. Wenn sich das Skript an einem S3-Speicherort befindet, ruft Amazon EMR in EKS das Skript ab und platziert es im `/opt/flink/usrlib/` Verzeichnis im Flink-Container.

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: python-example
spec:
  flinkVersion: v1_17
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "1"
  executionRoleArn: job-execution-role
  emrReleaseLabel: "emr-6.15.0-flink-latest"
  jobManager:
    highAvailabilityEnabled: false
    replicas: 1
    resource:
      memory: "2048m"
      cpu: 1
  taskManager:
    resource:
      memory: "2048m"
      cpu: 1
  job:
    jarURI: s3://S3 bucket with your script/pyflink-script.py
    entryClass: "org.apache.flink.client.python.PythonDriver"
    args: ["-py", "/opt/flink/usrlib/pyflink-script.py"]
    parallelism: 1
```

```
    upgradeMode: stateless
```

Verwenden von AWS Glue mit Flink

Amazon EMR in EKS mit Apache Flink Version 6.15.0 und höher unterstützt die Verwendung des AWS Glue Data Catalog als Metadatenspeicher für Streaming- und Batch-SQL-Workflows.

Sie müssen zuerst eine AWS -Glue-Datenbank mit dem Namen erstellen`default`, die als Ihr Flink-SQL-Katalog dient. Dieser Flink-Katalog speichert Metadaten wie Datenbanken, Tabellen, Paritionen, Ansichten, Funktionen und andere Informationen, die für den Zugriff auf Daten in anderen externen Systemen erforderlich sind.

```
aws glue create-database \
    --database-input "{\"Name\":\"default\"}"
```

Um die AWS -Glue-Unterstützung zu aktivieren, verwenden Sie eine `FlinkDeployment` -Spezifikation. Diese Beispielspezifikation verwendet ein Python-Skript, um schnell einige Flink-SQL-Anweisungen für die Interaktion mit dem AWS -Glue-Katalog auszugeben.

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: python-example
spec:
  flinkVersion: v1_17
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "1"
    aws.glue.enabled: "true"
  executionRoleArn: job-execution-role-arn;
  emrReleaseLabel: "emr-6.15.0-flink-latest"
  jobManager:
    highAvailabilityEnabled: false
    replicas: 1
    resource:
      memory: "2048m"
      cpu: 1
  taskManager:
    resource:
      memory: "2048m"
      cpu: 1
  job:
```

```
    jarURI: s3://<S3_bucket_with_your_script/pyflink-glue-script.py
    entryClass: "org.apache.flink.client.python.PythonDriver"
    args: ["-py", "/opt/flink/usrlib/pyflink-glue-script.py"]
    parallelism: 1
    upgradeMode: stateless
```

Im Folgenden finden Sie ein Beispiel dafür, wie Ihr PyFlink Skript aussehen könnte.

```
import logging
import sys
from pyflink.datastream import StreamExecutionEnvironment
from pyflink.table import StreamTableEnvironment

def glue_demo():
    env = StreamExecutionEnvironment.get_execution_environment()
    t_env = StreamTableEnvironment.create(stream_execution_environment=env)
    t_env.execute_sql("""
          CREATE CATALOG glue_catalog WITH (
          'type' = 'hive',
          'default-database' = 'default',
          'hive-conf-dir' = '/glue/confs/hive/conf',
          'hadoop-conf-dir' = '/glue/confs/hadoop/conf'
          )
                      """)
    t_env.execute_sql("""
          USE CATALOG glue_catalog;
                      """)
    t_env.execute_sql("""
          DROP DATABASE IF EXISTS eks_flink_db CASCADE;
                      """)
    t_env.execute_sql("""
          CREATE DATABASE IF NOT EXISTS eks_flink_db WITH ('hive.database.location-
uri'= 's3a://S3-bucket-to-store-metadata/flink/flink-glue-for-hive/warehouse/');
                      """)
    t_env.execute_sql("""
          USE eks_flink_db;
                  """)
    t_env.execute_sql("""
          CREATE TABLE IF NOT EXISTS eksglueorders (
            order_number BIGINT,
            price        DECIMAL(32,2),
            buyer        RO first_name STRING, last_name STRING,
            order_time   TIMESTAMP(3)
```

```
          ) WITH (
            'connector' = 'datagen'
          );
                      """)
    t_env.execute_sql("""
          CREATE TABLE IF NOT EXISTS eksdestglueorders (
            order_number BIGINT,
            price        DECIMAL(32,2),
            buyer        ROW first_name STRING, last_name STRING,
            order_time   TIMESTAMP(3)
          ) WITH (
            'connector' = 'filesystem',
            'path' = 's3://S3-bucket-to-store-metadata/flink/flink-glue-for-hive/
warehouse/eksdestglueorders',
            'format' = 'json'
          );
                  """)
    t_env.execute_sql("""
          CREATE TABLE IF NOT EXISTS print_table (
            order_number BIGINT,
            price        DECIMAL(32,2),
            buyer        ROW first_name STRING, last_name STRING,
            order_time   TIMESTAMP(3)
          ) WITH (
            'connector' = 'print'
          );
                """)
    t_env.execute_sql("""
          EXECUTE STATEMENT SET
          BEGIN
          INSERT INTO eksdestglueorders SELECT * FROM  eksglueorders LIMIT 10;
          INSERT INTO print_table SELECT * FROM eksdestglueorders;
          END;
            """)


if __name__ == '__main__':
    logging.basicConfig(stream=sys.stdout, level=logging.INFO, format="%(message)s")
    glue_demo()
```

Verwendung von RAPIDS Accelerator für Apache Spark mit Amazon EMR in EKS

Mit Amazon EMR in EKS können Sie Aufträge für den Nvidia RAPIDS Accelerator für Apache Spark ausführen. In diesem Tutorial wird beschrieben, wie Spark-Aufträge mit RAPIDS auf EC2-GPU-Instance-Typen (Graphics Processing Unit) ausgeführt werden. Das Tutorial verwendet die folgenden Versionen:

- Amazon EMR in EKS veröffentlicht Version 6.9.0 und höher
- Apache Spark 3.x

Sie können Spark mit Amazon-EC2-GPU-Instance-Typen beschleunigen, indem Sie das Plugin Nvidia RAPIDS Accelerator für Apache Spark verwenden. Wenn Sie diese Technologien zusammen verwenden, beschleunigen Sie Ihre Datenwissenschafts-Pipelines, ohne Codeänderungen vornehmen zu müssen. Dies reduziert die für die Datenverarbeitung und das Modelltraining benötigte Laufzeit. Wenn Sie in kürzerer Zeit mehr erledigen, geben Sie weniger für die Infrastruktur aus.

Bevor Sie beginnen, stellen Sie sicher, dass Sie über folgenden Ressourcen verfügen.

- Amazon EMR in EKS in einem virtuellen Cluster
- Amazon-EKS-Cluster mit einer GPU-fähigen Knotengruppe

Ein virtueller Amazon-EKS-Cluster ist ein registriertes Handle für den Kubernetes-Namespace auf einem Amazon-EKS-Cluster und wird von Amazon EMR in EKS verwaltet. Das Handle ermöglicht Amazon EMR, den Kubernetes-Namespace als Ziel für die Ausführung von Aufträgen zu verwenden. Weitere Informationen über das Einrichten eines virtuellen Clusters finden Sie Einrichten von Amazon EMR in EKS in diesem Handbuch.

Sie müssen den virtuellen Amazon-EKS-Cluster mit einer Knotengruppe konfigurieren, die GPU-Instances enthält. Sie müssen die Knoten mit einem Nvidia-Geräte-Plugin konfigurieren. Weitere Informationen finden Sie unter Verwaltete Knotengruppen.

Gehen Sie wie folgt vor, um Ihren Amazon-EKS-Cluster so zu konfigurieren, dass GPU-fähige Knotengruppen hinzugefügt werden:

Um GPU-fähige Knotengruppen hinzuzufügen

1. Erstellen Sie eine GPU-fähige Knotengruppe mit dem folgenden Befehl create-nodegroup. Stellen Sie sicher, dass Sie Ihren Amazon-EKS-Cluster durch die richtigen Parameter ersetzen können. Verwenden Sie einen Instance-Typ, der Spark RAPIDS unterstützt, wie P4, P3, G5 oder G4dn.

```
aws eks create-nodegroup \
 --cluster-name EKS_CLUSTER_NAME \
 --nodegroup-name NODEGROUP_NAME \
 --scaling-config minSize=0,maxSize=5,desiredSize=2 CHOOSE_APPROPRIATELY \
 --ami-type AL2_x86_64_GPU \
 --node-role NODE_ROLE \
 --subnets SUBNETS_SPACE_DELIMITED  \
 --remote-access ec2SshKey= SSH_KEY \
 --instance-types GPU_INSTANCE_TYPE \
 --disk-size DISK_SIZE \
 --region AWS_REGION
```

2. Installieren Sie das Nvidia-Geräte-Plugin in Ihrem Cluster, um die Anzahl der GPUs auf jedem Knoten Ihres Clusters auszugeben und GPU-fähige Container in Ihrem Cluster auszuführen. Führen Sie den folgenden Code aus, um das Plugin zu installieren:

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/v0.9.0/
nvidia-device-plugin.yml
```

3. Führen Sie den folgenden Befehl aus, um zu überprüfen, wie viele GPUs auf jedem Knoten Ihres Clusters verfügbar sind:

```
kubectl get nodes  "-o=custom-
columns=NAME:.metadata.name,GPU:.status.allocatable.nvidia\.com/gpu"
```

So führen Sie einen Spark-RAPIDS-Auftrag aus

1. Senden Sie einen Spark-RAPIDS-Auftrag an Ihren Cluster von Amazon EMR in EKS. Der folgende Code zeigt ein Beispiel für einen Befehl zum Starten der Aufgabe. Wenn Sie den Auftrag zum ersten Mal ausführen, kann es einige Minuten dauern, bis das Bild heruntergeladen und auf dem Knoten zwischengespeichert ist.

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

```
--virtual-cluster-id VIRTUAL_CLUSTER_ID \
--execution-role-arn JOB_EXECUTION_ROLE \
--release-label emr-6.9.0-spark-rapids-latest \
--job-driver '{"sparkSubmitJobDriver": {"entryPoint": "local:///usr/lib/
spark/examples/jars/spark-examples.jar","entryPointArguments":  ["10000"],
 "sparkSubmitParameters":"--class org.apache.spark.examples.SparkPi "}}' \
---configuration-overrides '{"applicationConfiguration": [{"classification":
 "spark-defaults","properties": {"spark.executor.instances":
 "2","spark.executor.memory": "2G"}}],"monitoringConfiguration":
 {"cloudWatchMonitoringConfiguration": {"logGroupName": "LOG_GROUP
 _NAME"},"s3MonitoringConfiguration": {"logUri": "LOG_GROUP_STREAM"}}}'
```

2. Um zu überprüfen, ob der Spark RAPIDS Accelerator aktiviert ist, überprüfen Sie die Spark-Treiberprotokolle. Diese Protokolle werden entweder in CloudWatch oder an dem S3-Speicherort gespeichert, den Sie bei der Ausführung des `start-job-run`-Befehls angeben. Das folgende Beispiel zeigt im Allgemeinen, wie die Protokollzeilen aussehen:

```
22/11/15 00:12:44 INFO RapidsPluginUtils: RAPIDS Accelerator build:
 {version=22.08.0-amzn-0, user=release, url=, date=2022-11-03T03:32:45Z, revision=,
 cudf_version=22.08.0, branch=}
22/11/15 00:12:44 INFO RapidsPluginUtils: RAPIDS Accelerator JNI build:
 {version=22.08.0, user=, url=https://github.com/NVIDIA/spark-rapids-jni.git,
 date=2022-08-18T04:14:34Z, revision=a1b23cd_sample, branch=HEAD}
22/11/15 00:12:44 INFO RapidsPluginUtils: cudf build: {version=22.08.0,
 user=, url=https://github.com/rapidsai/cudf.git, date=2022-08-18T04:14:34Z,
 revision=a1b23ce_sample, branch=HEAD}
22/11/15 00:12:44 WARN RapidsPluginUtils: RAPIDS Accelerator 22.08.0-amzn-0 using
 cudf 22.08.0.
22/11/15 00:12:44 WARN RapidsPluginUtils:
 spark.rapids.sql.multiThreadedRead.numThreads is set to 20.
22/11/15 00:12:44 WARN RapidsPluginUtils: RAPIDS Accelerator is enabled, to disable
 GPU support set `spark.rapids.sql.enabled` to false.
22/11/15 00:12:44 WARN RapidsPluginUtils: spark.rapids.sql.explain is set to
 `NOT_ON_GPU`. Set it to 'NONE' to suppress the diagnostics logging about the query
 placement on the GPU.
```

3. Um zu sehen, welche Operationen auf einer GPU ausgeführt werden, führen Sie die folgenden Schritte durch, um die zusätzliche Protokollierung zu aktivieren. Beachten Sie die „`spark.rapids.sql.explain : ALL`“-Konfiguration.

```
aws emr-containers start-job-run \
--virtual-cluster-id VIRTUAL_CLUSTER_ID \
```

```
--execution-role-arn JOB_EXECUTION_ROLE \
--release-label emr-6.9.0-spark-rapids-latest \
--job-driver '{"sparkSubmitJobDriver": {"entryPoint": "local:///usr/lib/
spark/examples/jars/spark-examples.jar","entryPointArguments":  ["10000"],
 "sparkSubmitParameters":"--class org.apache.spark.examples.SparkPi "}}' \
---configuration-overrides '{"applicationConfiguration":
 [{"classification": "spark-defaults","properties":
 {"spark.rapids.sql.explain":"ALL","spark.executor.instances":
 "2","spark.executor.memory": "2G"}}],"monitoringConfiguration":
 {"cloudWatchMonitoringConfiguration": {"logGroupName":
 "LOG_GROUP_NAME"},"s3MonitoringConfiguration": {"logUri": "LOG_GROUP_STREAM"}}}'
```

Der vorherige Befehl ist ein Beispiel für einen Auftrag, der die GPU verwendet. Die Ausgabe würde ungefähr so aussehen wie im folgenden Beispiel. Anhand dieses Schlüssels können Sie die Ausgabe besser verstehen:

- * – markiert einen Vorgang, der auf einer GPU funktioniert
- ! – markiert einen Vorgang, der nicht auf einer GPU ausgeführt werden kann
- @ – markiert einen Vorgang, der auf einer GPU funktioniert, aber nicht ausgeführt werden kann, weil er Teil eines Plans ist, der auf einer GPU nicht ausgeführt werden kann

```
22/11/15 01:22:58 INFO GpuOverrides: Plan conversion to the GPU took 118.64 ms
22/11/15 01:22:58 INFO GpuOverrides: Plan conversion to the GPU took 4.20 ms
22/11/15 01:22:58 INFO GpuOverrides: GPU plan transition optimization took 8.37 ms
22/11/15 01:22:59 WARN GpuOverrides:
   *Exec <ProjectExec> will run on GPU
     *Expression <Alias> substring(cast(date#149 as string), 0, 7) AS month#310
will run on GPU
       *Expression <Substring> substring(cast(date#149 as string), 0, 7) will run
on GPU
         *Expression <Cast> cast(date#149 as string) will run on GPU
     *Exec <SortExec> will run on GPU
       *Expression <SortOrder> date#149 ASC NULLS FIRST will run on GPU
       *Exec <ShuffleExchangeExec> will run on GPU
         *Partitioning <RangePartitioning> will run on GPU
           *Expression <SortOrder> date#149 ASC NULLS FIRST will run on GPU
         *Exec <UnionExec> will run on GPU
           !Exec <ProjectExec> cannot run on GPU because not all expressions can
be replaced
             @Expression <AttributeReference> customerID#0 could run on GPU
```

```
              @Expression <Alias> Charge AS kind#126 could run on GPU
                @Expression <Literal> Charge could run on GPU
              @Expression <AttributeReference> value#129 could run on GPU
              @Expression <Alias> add_months(2022-11-15, cast(-(cast(_we0#142 as
 bigint) + last_month#128L) as int)) AS date#149 could run on GPU
                ! <AddMonths> add_months(2022-11-15, cast(-
(cast(_we0#142 as bigint) + last_month#128L) as int)) cannot run
 on GPU because GPU does not currently support the operator class
 org.apache.spark.sql.catalyst.expressions.AddMonths
                  @Expression <Literal> 2022-11-15 could run on GPU
                  @Expression <Cast> cast(-(cast(_we0#142 as bigint) +
 last_month#128L) as int) could run on GPU
                    @Expression <UnaryMinus> -(cast(_we0#142 as bigint) +
 last_month#128L) could run on GPU
                      @Expression <Add> (cast(_we0#142 as bigint) +
 last_month#128L) could run on GPU
                        @Expression <Cast> cast(_we0#142 as bigint) could run on
 GPU
                          @Expression <AttributeReference> _we0#142 could run on
 GPU
                        @Expression <AttributeReference> last_month#128L could run
 on GPU
```

Verwenden der Amazon-Redshift-Integration für Apache Spark auf Amazon EMR in EKS

Mit Amazon-EMR-Version 6.9.0 und höher enthält jedes Versions-Image einen Konnektor zwischen Apache Spark und Amazon Redshift. Auf diese Weise können Sie Spark on Amazon EMR in EKS verwenden, um in Amazon Redshift gespeicherte Daten zu verarbeiten. Die Integration basiert auf dem `spark-redshift`-Open-Source-Konnektor. Für Amazon EMR in EKS ist die Amazon-Redshift-Integration für Apache Spark als native Integration enthalten.

Themen

- Starten einer Spark-Anwendung mithilfe der Amazon-Redshift-Integration für Apache Spark
- Authentifizierung mit der Amazon-Redshift-Integration für Apache Spark
- Lesen und Schreiben von und zu Amazon Redshift
- Überlegungen und Einschränkungen bei der Verwendung des Spark-Connectors

Starten einer Spark-Anwendung mithilfe der Amazon-Redshift-Integration für Apache Spark

Um die Integration nutzen zu können, müssen Sie die erforderlichen Spark Redshift-Abhängigkeiten mit Ihrem Spark-Auftrag übergeben. Sie müssen `--jars` verwenden, um Redshift-Konnektor-bezogene Bibliotheken einzuschließen. 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`

Verwenden Sie den folgenden Beispielbefehl, um eine Spark-Anwendung mit der Amazon-Redshift-Integration für Apache Spark auf Amazon EMR in EKS-Version 6.9.0 oder höher zu starten. Beachten Sie, dass die mit der `--conf spark.jars`-Option aufgeführten Pfade die Standardpfade für die JAR-Dateien sind.

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

--virtual-cluster-id cluster_id \
--execution-role-arn arn \
--release-label emr-6.9.0-latest\
--job-driver '{
    "sparkSubmitJobDriver": {
        "entryPoint": "s3://script_path",
            "sparkSubmitParameters":
            "--conf spark.kubernetes.file.upload.path=s3://upload_path
             --conf spark.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"
                            }
            }'
```

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 Anmeldeinformationen in Secrets Manager speichern, um sich sicher bei Amazon Redshift zu authentifizieren. Sie können Ihren Spark-Auftrag die `GetSecretValue`-API aufrufen lassen, um die Anmeldeinformationen 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
```

Verwenden Sie die IAM-basierte Authentifizierung mit Amazon EMR in der EKS-Auftragsausführungsrolle

Ab Amazon EMR in EKS-Version 6.9.0 ist der Amazon-Redshift-JDBC-Treiber Version 2.1 oder höher in der Umgebung enthalten. Mit dem JDBC-Treiber 2.1 und höher können Sie die JDBC-URL angeben, ohne den unformatierten Benutzernamen und das Passwort anzugeben. Stattdessen können Sie ein `jdbc:redshift:iam://`-Schema angeben. Dadurch wird der JDBC-Treiber angewiesen, Ihre Amazon EMR in EKS-Auftragsausführungsrolle zu verwenden, um die Anmeldeinformationen automatisch abzurufen.

Weitere Informationen finden Sie unter Konfigurieren Sie eine JDBC- oder ODBC-Verbindung für die Verwendung von IAM-Anmeldeinformationen im Amazon-Redshift-Verwaltungshandbuch.

Die folgende Beispiel-URL verwendet ein `jdbc:redshift:iam://`-Schema.

```
jdbc:redshift:iam://examplecluster.abc123xyz789.us-west-2.redshift.amazonaws.com:5439/
dev
```

Die folgenden Berechtigungen sind für Ihre Auftragsausführungsrolle erforderlich, wenn sie die angegebenen Bedingungen erfüllt.

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-Cluster und AWS-Region in der JDBC-URL anstelle des Endpunkts angeben
`redshift-serverless:GetCredentials`	Erforderlich, damit der JDBC-Treiber die Anmeldeinformationen von Amazon Redshift Serverless abrufen kann
`redshift-serverless:GetWorkgroup`	Erforderlich, wenn Sie Amazon Redshift Serverless verwenden und die URL in Form von Arbeitsgruppenname und Region angeben

Ihre Auftragsausführungsrollenrichtlinie sollte über die folgenden Berechtigungen verfügen.

```
{
            "Effect": "Allow",
            "Action": [
                "redshift:GetClusterCredentials",
                "redshift:DescribeCluster",
                "redshift-serverless:GetCredentials",
                "redshift-serverless:GetWorkgroup"
            ],
            "Resource": [

 "arn:aws:redshift:AWS_REGION:ACCOUNT_ID:dbname:CLUSTER_NAME/DATABASE_NAME",
```

```
            "arn:aws:redshift:AWS_REGION:ACCOUNT_ID:dbuser:DATABASE_NAME/USER_NAME"
        ]
    }
```

Authentifizierung bei Amazon Redshift mit einem JDBC-Treiber

Geben Sie den Benutzernamen und das Passwort in der JDBC-URL ein

Um einen Spark-Auftrag bei einem Amazon-Redshift-Cluster zu authentifizieren, können Sie den Namen und das Passwort der Amazon-Redshift-Datenbank in der JDBC-URL angeben.

> **Note**
>
> Wenn Sie die Datenbankanmeldedaten in der URL übergeben, kann jeder, der Zugriff auf die URL hat, auch auf die Anmeldeinformationen zugreifen. Diese Methode wird im Allgemeinen nicht empfohlen, da sie keine sichere Option ist.

Wenn Sicherheit für Ihre Anwendung kein Problem darstellt, können Sie das folgende Format verwenden, um den Benutzernamen und das Passwort in der JDBC-URL festzulegen:

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

Lesen und Schreiben von und zu Amazon Redshift

Die folgenden Codebeispiele verwenden PySpark , um Beispieldaten aus und in eine Amazon-Redshift-Datenbank mit einer Datenquellen-API und mit SparkSQL zu lesen und zu schreiben.

Data source API

Verwenden Sie PySpark , um Beispieldaten aus und in eine Amazon-Redshift-Datenbank mit einer Datenquellen-API zu lesen und zu schreiben.

```
import boto3
from pyspark.sql import SQLContext

sc = # existing SparkContext
sql_context = SQLContext(sc)

url = "jdbc:redshift:iam://redshifthost:5439/database"
```

```
aws_iam_role_arn = "arn:aws:iam::accountID:role/roleName"

df = sql_context.read \
    .format("io.github.spark_redshift_community.spark.redshift") \
    .option("url", url) \
    .option("dbtable", "tableName") \
    .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", "tableName_copy") \
    .option("tempdir", "s3://path/for/temp/data") \
    .option("aws_iam_role", "aws_iam_role_arn") \
    .mode("error") \
    .save()
```

SparkSQL

Verwenden Sie PySpark , um Beispieldaten aus und in eine Amazon-Redshift-Datenbank mit SparkSQL 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::accountID:role/roleName"

bucket = "s3://path/for/temp/data"
tableName = "tableName" # Redshift table name

s = f"""CREATE TABLE IF NOT EXISTS {tableName} (country string, data string)
    USING io.github.spark_redshift_community.spark.redshift
```

```
    OPTIONS (dbtable '{tableName}', 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(tableName, overwrite=False)
df = spark.sql(f"SELECT * FROM {tableName}")
df.show()
```

Überlegungen und Einschränkungen bei der Verwendung des Spark-Connectors

- Wir empfehlen, Ihnen SSL für die JDBC-Verbindung von Spark auf Amazon EMR zu Amazon Redshift zu aktivieren.
- Als bewährte Methode empfehlen wir Ihnen, die Anmeldeinformationen für den Amazon-Redshift-Cluster in AWS Secrets Manager zu verwalten. Ein Beispiel finden Sie unter Verwenden von AWS Secrets Manager zum Abrufen von Anmeldeinformationen für die Verbindung mit Amazon Redshift.
- Wir empfehlen, dass Sie eine IAM-Rolle mit dem Parameter `aws_iam_role` für den Amazon-Redshift-Authentifizierungsparameter zu übergeben.
- Derzeit wird das Parquet-Format vom Parameter `tempformat` nicht unterstützt.
- Die `tempdir`-URI verweist auf einen Amazon-S3-Speicherort. Dieses temporäre Verzeichnis wird nicht automatisch bereinigt und kann zusätzliche Kosten verursachen.
- Beachten Sie die folgenden Empfehlungen für Amazon Redshift:
  - Wir empfehlen, den öffentlichen Zugriff auf den Amazon-Redshift-Cluster zu blockieren.
  - Wir empfehlen, die Amazon-Redshift-Auditprotokollierung zu aktivieren.
  - Wir empfehlen, die Amazon-Redshift-Verschlüsselung im Ruhezustand zu aktivieren.
- Beachten Sie die folgenden Empfehlungen für Amazon S3:
  - Wir empfehlen, den öffentlichen Zugriff auf Amazon-S3-Buckets zu blockieren.
  - Wir empfehlen die Verwendung der serverseitigen Amazon-S3-Verschlüsselung, um die verwendeten S3-Buckets zu verschlüsseln.

- Wir empfehlen, die Lebenszyklusrichtlinien für Amazon S3 zu verwenden, um die Aufbewahrungsregeln für den S3-Bucket zu definieren.
- Amazon EMR überprüft immer Code, der aus Open Source in das Image importiert wurde. Aus Sicherheitsgründen unterstützen wir die Verschlüsselung von AWS-Zugriffsschlüsseln in der `tempdir`-URI nicht als Authentifizierungsmethode von Spark zu Amazon S3.

Weitere Informationen zum Verwenden des Konnekctors 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

Verwendung von Volcano als benutzerdefiniertem Scheduler für Apache Spark auf Amazon EMR in EKS

Mit Amazon EMR in EKS können Sie Spark-Operator oder Spark-Submit verwenden, um Spark-Aufträge mit benutzerdefinierten Kubernetes-Planern auszuführen. In diesem Tutorial erfahren Sie, wie Sie Spark-Aufträge mit einem Volcano-Planer in einer benutzerdefinierten Warteschlange ausführen.

Übersicht

Volcano kann Sie bei der Verwaltung der Spark-Planung mit erweiterten Funktionen wie Warteschlangenplanung, Fair-Share-Planung und Ressourcenreservierung unterstützen. Weitere Informationen zu den Vorteilen von Volcano finden Sie in Warum Spark Volcano als integrierten Batch-Planer auf Kubernetes auswählt im CNCF-Blog der Linux Foundation.

Installieren und einrichten von Volcano

1. Wählen Sie einen der folgenden kubectl-Befehle, um Volcano je nach Ihrem architektonischen Bedarf zu installieren:

```
# x86_64
kubectl apply -f https://raw.githubusercontent.com/volcano-sh/volcano/v1.5.1/
installer/volcano-development.yaml
# arm64:
```

```
kubectl apply -f https://raw.githubusercontent.com/volcano-sh/volcano/v1.5.1/
installer/volcano-development-arm64.yaml
```

2. Bereiten Sie eine Volcano-Beispielwarteschlange vor. Eine Warteschlange ist eine Sammlung von PodGroups. Die Warteschlange verwendet FIFO und ist die Grundlage für die Aufteilung der Ressourcen.

```
cat << EOF > volcanoQ.yaml
apiVersion: scheduling.volcano.sh/v1beta1
kind: Queue
metadata:
  name: sparkqueue
spec:
  weight: 4
  reclaimable: false
  capability:
    cpu: 10
    memory: 20Gi
EOF

kubectl apply -f volcanoQ.yaml
```

3. Laden Sie ein PodGroup-Beispiel-Manifest in Amazon S3 hoch. PodGroup ist eine Gruppe von Pods mit starker Assoziation. Normalerweise verwenden Sie eine PodGroup für die Batch-Planung. Senden Sie die folgende Beispiel-PodGroup in die Warteschlange, die Sie im vorherigen Schritt definiert haben.

```
cat << EOF > podGroup.yaml
apiVersion: scheduling.volcano.sh/v1beta1
kind: PodGroup
spec:
  # Set minMember to 1 to make a driver pod
  minMember: 1
  # Specify minResources to support resource reservation.
  # Consider the driver pod resource and executors pod resource.
  # The available resources should meet the minimum requirements of the Spark job
  # to avoid a situation where drivers are scheduled, but they can't schedule
  # sufficient executors to progress.
  minResources:
    cpu: "1"
    memory: "1Gi"
```

```
  # Specify the queue. This defines the resource queue that the job should be
 submitted to.
  queue: sparkqueue
EOF

aws s3 mv podGroup.yaml s3://bucket-name
```

Führen Sie eine Spark-Anwendung mit Volcano Scheduler und dem Spark-Operator aus

1. Sofern noch nicht geschehen, stellen Sie sicher, dass Sie folgende Voraussetzungen erfüllen:

   a. Installieren und einrichten von Volcano

   b. Einrichten des Spark-Operators für Amazon EMR in EKS

   c. Den Spark-Operator installieren

   Geben Sie bei der Ausführung des `helm install spark-operator-demo`-Befehls die folgenden Argumente an:

   ```
   --set batchScheduler.enable=true
   --set webhook.enable=true
   ```

2. Erstellen Sie eine `SparkApplication`-Definitionsdatei `spark-pi.yaml` mit der `batchScheduler`-Konfiguration.

   ```
   apiVersion: "sparkoperator.k8s.io/v1beta2"
   kind: SparkApplication
   metadata:
     name: spark-pi
     namespace: spark-operator
   spec:
     type: Scala
     mode: cluster
     image: "895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.10.0:latest"
     imagePullPolicy: Always
     mainClass: org.apache.spark.examples.SparkPi
     mainApplicationFile: "local:///usr/lib/spark/examples/jars/spark-examples.jar"
     sparkVersion: "3.3.1"
     batchScheduler: "volcano"   #Note: You must specify the batch scheduler name as
    'volcano'
   ```

```
  restartPolicy:
    type: Never
  volumes:
    - name: "test-volume"
      hostPath:
        path: "/tmp"
        type: Directory
  driver:
    cores: 1
    coreLimit: "1200m"
    memory: "512m"
    labels:
      version: 3.3.1
    serviceAccount: emr-containers-sa-spark
    volumeMounts:
      - name: "test-volume"
        mountPath: "/tmp"
  executor:
    cores: 1
    instances: 1
    memory: "512m"
    labels:
      version: 3.3.1
    volumeMounts:
      - name: "test-volume"
        mountPath: "/tmp"
```

3. Senden Sie die Spark-Anwendung mit dem folgenden Befehl. Dadurch wird auch ein `SparkApplication`-Objekt mit dem Namen `spark-pi` erstellt:

```
kubectl apply -f spark-pi.yaml
```

4. Überprüfen Sie die Ereignisse für das `SparkApplication`-Objekt mit dem folgenden Befehl:

```
kubectl describe pods spark-pi-driver --namespace spark-operator
```

Das erste Pod-Ereignis zeigt, dass Volcano die Pods geplant hat:

```
Type    Reason     Age   From               Message
----    ------     ----  ----               -------
Normal  Scheduled  23s   volcano            Successfully assigned default/spark-
pi-driver to integration-worker2
```

Führen Sie eine Spark-Anwendung mit Volcano Scheduler mit `spark-submit` aus

1. Führen Sie zunächst die Schritte in diesem Einrichten von spark-submit für Amazon EMR in EKS-Abschnitt durch. Sie müssen Ihre `spark-submit` Distribution mit Volcano-Unterstützung erstellen. Weitere Informationen finden Sie im Abschnitt Erstellung unter Verwendung von Volcano als benutzerdefinierter Planer für Spark in Kubernetes in der Apache-Spark-Dokumentation.
2. Legen Sie die Werte der folgenden Umgebungsvariablen fest:

```
export SPARK_HOME=spark-home
export MASTER_URL=k8s://Amazon-EKS-cluster-endpoint
```

3. Senden Sie die Spark-Anwendung mit dem folgenden Befehl:

```
$SPARK_HOME/bin/spark-submit \
 --class org.apache.spark.examples.SparkPi \
 --master $MASTER_URL \
 --conf spark.kubernetes.container.image=895885662937.dkr.ecr.us-
west-2.amazonaws.com/spark/emr-6.10.0:latest \
 --conf spark.kubernetes.authenticate.driver.serviceAccountName=spark \
 --deploy-mode cluster \
 --conf spark.kubernetes.namespace=spark-operator \
 --conf spark.kubernetes.scheduler.name=volcano \
 --conf spark.kubernetes.scheduler.volcano.podGroupTemplateFile=/path/to/podgroup-
template.yaml \
 --conf
 spark.kubernetes.driver.pod.featureSteps=org.apache.spark.deploy.k8s.features.VolcanoFeatu
 \
 --conf
 spark.kubernetes.executor.pod.featureSteps=org.apache.spark.deploy.k8s.features.VolcanoFea
 \
 local:///usr/lib/spark/examples/jars/spark-examples.jar 20
```

4. Überprüfen Sie die Ereignisse für das `SparkApplication`-Objekt mit dem folgenden Befehl:

```
kubectl describe pod spark-pi --namespace spark-operator
```

Das erste Pod-Ereignis zeigt, dass Volcano die Pods geplant hat:

```
Type    Reason     Age   From                Message
----    ------     ----  ----                -------
Normal  Scheduled  23s   volcano             Successfully assigned default/spark-
pi-driver to integration-worker2
```

Verwendung von YuniKorn als benutzerdefiniertem Scheduler für Apache Spark in Amazon EMR in EKS

Mit Amazon EMR in EKS können Sie Spark-Operator oder Spark-Submit verwenden, um Spark-Aufträge mit benutzerdefinierten Kubernetes-Schedulern auszuführen. In diesem Tutorial erfahren Sie, wie Sie Spark-Aufträge mit einem YuniKorn-Planer in einer benutzerdefinierten Warteschlange und Gruppenplanung ausführen.

Übersicht

Apache YuniKorn kann Sie bei der Verwaltung der Spark-Planung mit einer anwendungsorientierten Planung unterstützen, sodass Sie eine genaue Kontrolle über Ressourcenkontingente und Prioritäten haben. Bei der Gruppenplanung plant YuniKorn eine Anwendung nur dann, wenn die minimale Ressourcenanforderung für die Anwendung erfüllt werden kann. Weitere Informationen finden Sie unter Was ist Gruppenplanung auf der Apache-YuniKorn-Dokumentationsseite.

Erstellen Sie Ihren Cluster und richten Sie ihn für YuniKorn ein

Führen Sie die folgenden Schritte aus, um einen Amazon-EKS-Cluster bereitzustellen. Sie können die AWS-Region (`region`) und Availability Zones (`availabilityZones`) ändern.

1. Definieren Sie den Amazon-EKS-Cluster:

```
cat <<EOF >eks-cluster.yaml
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: emr-eks-cluster
  region: eu-west-1

vpc:
```

```
  clusterEndpoints:
    publicAccess: true
    privateAccess: true

iam:
  withOIDC: true

nodeGroups:
  - name: spark-jobs
    labels: { app: spark }
    instanceType: m5.xlarge
    desiredCapacity: 2
    minSize: 2
    maxSize: 3
    availabilityZones: ["eu-west-1a"]
EOF
```

2. Erstellen Sie den Cluster:

```
eksctl create cluster -f eks-cluster.yaml
```

3. Erstellen Sie den Namespace `spark-job`, in dem Sie den Spark-Auftrag ausführen werden:

```
kubectl create namespace spark-job
```

4. Erstellen Sie als Nächstes eine Kubernetes-Rolle und eine Rollenbindung. Dies ist für das Servicekonto erforderlich, das der Spark-Auftraglauf verwendet.

   a. Definieren Sie das Servicekonto, die Rolle und die Rollenbindung für Spark-Aufträge.

```
cat <<EOF >emr-job-execution-rbac.yaml
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: spark-sa
  namespace: spark-job
automountServiceAccountToken: false
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: spark-role
```

```
  namespace: spark-job
rules:
  - apiGroups: ["", "batch","extensions"]
    resources: ["configmaps","serviceaccounts","events","pods","pods/
exec","pods/log","pods/
portforward","secrets","services","persistentvolumeclaims"]
    verbs: ["create","delete","get","list","patch","update","watch"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: spark-sa-rb
  namespace: spark-job
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: spark-role
subjects:
  - kind: ServiceAccount
    name: spark-sa
    namespace: spark-job
EOF
```

b. Wenden Sie die Kubernetes-Rollen- und Rollenbindungsdefinition mit dem folgenden Befehl an:

```
kubectl apply -f emr-job-execution-rbac.yaml
```

YuniKorn installieren und einrichten

1. Verwenden Sie den folgenden kubectl-Befehl, um einen `yunikorn`-Namespace für die Bereitstellung des Yunikorn-Planers zu erstellen:

```
kubectl create namespace yunikorn
```

2. Um den Planer zu installieren, führen Sie die folgenden Helm-Befehle aus:

```
helm repo add yunikorn https://apache.github.io/yunikorn-release
```

```
helm repo update
```

```
helm install yunikorn yunikorn/yunikorn --namespace yunikorn
```

Führen Sie eine Spark-Anwendung mit YuniKorn-Planer und dem Spark-Operator aus

1. Sofern noch nicht geschehen, stellen Sie sicher, dass Sie folgende Voraussetzungen erfüllen:

   a. Erstellen Sie Ihren Cluster und richten Sie ihn für YuniKorn ein

   b. YuniKorn installieren und einrichten

   c. Einrichten des Spark-Operators für Amazon EMR in EKS

   d. Den Spark-Operator installieren

      Geben Sie bei der Ausführung des `helm install spark-operator-demo`-Befehls die folgenden Argumente an:

      ```
      --set batchScheduler.enable=true
      --set webhook.enable=true
      ```

2. Erstellen Sie eine `SparkApplication`-Definitionsdatei `spark-pi.yaml`.

   Um YuniKorn als Planer für Ihre Aufträge zu verwenden, müssen Sie Ihrer Anwendungsdefinition bestimmte Anmerkungen und Labels hinzufügen. Die Anmerkungen und Beschriftungen spezifizieren die Warteschlange für Ihren Auftrag und die Planungsstrategie, die Sie verwenden möchten.

   Im folgenden Beispiel `schedulingPolicyParameters` richtet die Anmerkung die Gruppenplanung für die Anwendung ein. Anschließend werden im Beispiel Aufgabengruppen oder „Gruppen“ von Aufgaben erstellt, um die Mindestkapazität anzugeben, die verfügbar sein muss, bevor die Pods für den Start der Auftragsausführung geplant werden. Und schließlich gibt es in der Aufgabengruppendefinition an, dass Knotengruppen mit der `"app": "spark"`-Bezeichnung verwendet werden sollen, wie im Erstellen Sie Ihren Cluster und richten Sie ihn für YuniKorn ein-Abschnitt definiert.

   ```
   apiVersion: "sparkoperator.k8s.io/v1beta2"
   kind: SparkApplication
   metadata:
     name: spark-pi
   ```

```
  namespace: spark-job
spec:
  type: Scala
  mode: cluster
  image: "895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.10.0:latest"
  imagePullPolicy: Always
  mainClass: org.apache.spark.examples.SparkPi
  mainApplicationFile: "local:///usr/lib/spark/examples/jars/spark-examples.jar"
  sparkVersion: "3.3.1"
  restartPolicy:
    type: Never
  volumes:
    - name: "test-volume"
      hostPath:
        path: "/tmp"
        type: Directory
  driver:
    cores: 1
    coreLimit: "1200m"
    memory: "512m"
    labels:
      version: 3.3.1
    annotations:
      yunikorn.apache.org/schedulingPolicyParameters: "placeholderTimeoutSeconds=30
 gangSchedulingStyle=Hard"
      yunikorn.apache.org/task-group-name: "spark-driver"
      yunikorn.apache.org/task-groups: |-
        [{
            "name": "spark-driver",
            "minMember": 1,
            "minResource": {
              "cpu": "1200m",
              "memory": "1Gi"
            },
            "nodeSelector": {
              "app": "spark"
            }
          },
          {
            "name": "spark-executor",
            "minMember": 1,
            "minResource": {
              "cpu": "1200m",
              "memory": "1Gi"
```

```
            },
            "nodeSelector": {
              "app": "spark"
            }
        }]
    serviceAccount: spark-sa
    volumeMounts:
      - name: "test-volume"
        mountPath: "/tmp"
  executor:
    cores: 1
    instances: 1
    memory: "512m"
    labels:
      version: 3.3.1
    annotations:
      yunikorn.apache.org/task-group-name: "spark-executor"
    volumeMounts:
      - name: "test-volume"
        mountPath: "/tmp"
```

3. Senden Sie die Spark-Anwendung mit dem folgenden Befehl. Dadurch wird auch ein `SparkApplication`-Objekt mit dem Namen `spark-pi` erstellt:

```
kubectl apply -f spark-pi.yaml
```

4. Überprüfen Sie die Ereignisse für das `SparkApplication`-Objekt mit dem folgenden Befehl:

```
kubectl describe sparkapplication spark-pi --namespace spark-job
```

Das erste Pod-Ereignis zeigt, dass YuniKorn die Pods geplant hat:

```
Type    Reason            Age   From                         Message
----    ------            ----  ----                         -------
Normal Scheduling        3m12s yunikorn   spark-operator/org-apache-spark-examples-
sparkpi-2a777a88b98b8a95-driver is queued and waiting for allocation
Normal GangScheduling    3m12s yunikorn   Pod belongs to the taskGroup spark-
driver, it will be scheduled as a gang member
Normal Scheduled         3m10s yunikorn   Successfully assigned spark
Normal PodBindSuccessful 3m10s yunikorn   Pod spark-operator/
Normal TaskCompleted     2m3s  yunikorn   Task spark-operator/
Normal Pulling           3m10s kubelet    Pulling
```

Ausführung einer Spark-Anwendung mit dem YuniKorn-Planer mit `spark-submit`

1. Führen Sie zunächst die Schritte in diesem Einrichten von spark-submit für Amazon EMR in EKS-Abschnitt durch.
2. Legen Sie die Werte der folgenden Umgebungsvariablen fest:

```
export SPARK_HOME=spark-home
export MASTER_URL=k8s://Amazon-EKS-cluster-endpoint
```

3. Senden Sie die Spark-Anwendung mit dem folgenden Befehl:

   Im folgenden Beispiel `schedulingPolicyParameters` richtet die Anmerkung die Gruppenplanung für die Anwendung ein. Anschließend werden im Beispiel Aufgabengruppen oder „Gruppen“ von Aufgaben erstellt, um die Mindestkapazität anzugeben, die verfügbar sein muss, bevor die Pods für den Start der Auftragsausführung geplant werden. Und schließlich gibt es in der Aufgabengruppendefinition an, dass Knotengruppen mit der `"app": "spark"`-Bezeichnung verwendet werden sollen, wie im Erstellen Sie Ihren Cluster und richten Sie ihn für YuniKorn ein-Abschnitt definiert.

```
$SPARK_HOME/bin/spark-submit \
 --class org.apache.spark.examples.SparkPi \
 --master $MASTER_URL \
 --conf spark.kubernetes.container.image=895885662937.dkr.ecr.us-
west-2.amazonaws.com/spark/emr-6.10.0:latest \
 --conf spark.kubernetes.authenticate.driver.serviceAccountName=spark-sa \
 --deploy-mode cluster \
 --conf spark.kubernetes.namespace=spark-job \
 --conf spark.kubernetes.scheduler.name=yunikorn \
 --conf spark.kubernetes.driver.annotation.yunikorn.apache.org/
schedulingPolicyParameters="placeholderTimeoutSeconds=30 gangSchedulingStyle=Hard"
 \
 --conf spark.kubernetes.driver.annotation.yunikorn.apache.org/task-group-
name="spark-driver" \
 --conf spark.kubernetes.executor.annotation.yunikorn.apache.org/task-group-
name="spark-executor" \
 --conf spark.kubernetes.driver.annotation.yunikorn.apache.org/task-groups='[{
            "name": "spark-driver",
            "minMember": 1,
            "minResource": {
```

```
              "cpu": "1200m",
              "memory": "1Gi"
            },
            "nodeSelector": {
              "app": "spark"
            }
          },
          {
            "name": "spark-executor",
            "minMember": 1,
            "minResource": {
              "cpu": "1200m",
              "memory": "1Gi"
            },
            "nodeSelector": {
              "app": "spark"
            }
        }]' \
 local:///usr/lib/spark/examples/jars/spark-examples.jar 20
```

4. Überprüfen Sie die Ereignisse für das `SparkApplication`-Objekt mit dem folgenden Befehl:

```
kubectl describe pod spark-driver-pod --namespace spark-job
```

Das erste Pod-Ereignis zeigt, dass YuniKorn die Pods geplant hat:

```
Type    Reason           Age   From                          Message
----    ------           ----  ----                          -------
Normal Scheduling        3m12s yunikorn   spark-operator/org-apache-spark-examples-
sparkpi-2a777a88b98b8a95-driver is queued and waiting for allocation
Normal GangScheduling    3m12s yunikorn   Pod belongs to the taskGroup spark-
driver, it will be scheduled as a gang member
Normal Scheduled         3m10s yunikorn   Successfully assigned spark
Normal PodBindSuccessful 3m10s yunikorn   Pod spark-operator/
Normal TaskCompleted     2m3s  yunikorn   Task spark-operator/
Normal Pulling           3m10s kubelet    Pulling
```

Sicherheit in Amazon EMR in EKS

Die Sicherheit in der Cloud hat für AWS höchste Priorität. Als AWS-Kunde profitieren Sie von Rechenzentren und Netzwerkarchitekturen, die eingerichtet wurden, um die Anforderungen der anspruchsvollsten Organisationen in puncto Sicherheit zu erfüllen.

Sicherheit ist eine übergreifende Verantwortlichkeit zwischen AWS und Ihnen. Das Modell der geteilten Verantwortung beschreibt dies als Sicherheit der Cloud selbst und Sicherheit in der Cloud:

- Sicherheit der Cloud selbst – AWS ist dafür verantwortlich, die Infrastruktur zu schützen, mit der AWS-Services in der AWS Cloud ausgeführt werden. AWS stellt Ihnen außerdem Services bereit, die Sie sicher nutzen können. Auditoren von Drittanbietern testen und überprüfen die Effektivität unserer Sicherheitsmaßnahmen im Rahmen der AWS-Compliance-Programme regelmäßig. Informationen zu den Compliance-Programmen, die für Amazon EMR gelten, finden Sie unter Im Rahmen des Compliance-Programms zugelassene AWS-Services.
- Sicherheit in der Cloud – Ihr Verantwortungsumfang wird durch den AWS-Dienst bestimmt, den Sie verwenden. Sie sind auch für andere Faktoren verantwortlich, etwa für die Vertraulichkeit Ihrer Daten, für die Anforderungen Ihres Unternehmens und für die geltenden Gesetze und Vorschriften.

Diese Dokumentation zeigt Ihnen, wie Sie das Modell der übergreifenden Verantwortlichkeit bei der Verwendung von Amazon EMR in EKS einsetzen können. Die folgenden Themen zeigen Ihnen, wie Sie Amazon EMR in EKS konfigurieren, um Ihre Sicherheits- und Compliance-Ziele zu erreichen. Außerdem erfahren Sie, wie Sie andere AWS-Services verwenden, die Ihnen bei der Überwachung und Sicherung Ihrer Amazon EMR in EKS-Ressourcen helfen.

Themen

- Konfigurations- und Schwachstellenanalyse
- Herstellen einer Verbindung mit Amazon EMR in EKS über einen Schnittstellen-VPC-Endpunkt
- Kontoübergreifenden Zugriff für Amazon EMR in EKS einrichten

Bewährte Methoden für Sicherheit in Amazon EMR in EKS

Amazon EMR in EKS enthält eine Reihe von Sicherheitsfeatures, 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.

> **Note**
>
> Bewährte Methoden für die Sicherheit in finden Sie unter Bewährte Methoden für Sicherheit in Amazon EMR in EKS.

Das Prinzip der geringsten Berechtigung anwenden

Amazon EMR in EKS bietet eine detaillierte Zugriffsrichtlinie für Anwendungen, die IAM-Rollen verwenden, z. B. Ausführungsrollen. Diese Ausführungsrollen werden den Kubernetes-Servicekonten über die Vertrauensrichtlinie der IAM-Rolle zugeordnet. Amazon EMR in EKS erstellt Pods innerhalb eines registrierten Amazon-EKS-Namespace, die vom Benutzer bereitgestellten Anwendungscode ausführen. Die Auftrag-Pods, auf denen der Anwendungscode ausgeführt wird, übernehmen die Ausführungsrolle, wenn sie eine Verbindung zu anderen AWS-Diensten herstellen. 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.

Zugriffskontrollliste für Endgeräte

Verwaltete Endpunkte können nur für die EKS-Cluster erstellt werden, die so konfiguriert wurden, dass sie mindestens ein privates Subnetz in Ihrer VPC verwenden. Diese Konfiguration schränkt den Zugriff auf die Load Balancer ein, die von verwalteten Endpunkten erstellt wurden, sodass nur

von Ihrer VPC aus auf sie zugegriffen werden kann. Um die Sicherheit weiter zu erhöhen, empfehlen wir, Sicherheitsgruppen mit diesen Load Balancern zu konfigurieren, sodass sie den eingehenden Datenverkehr auf einen ausgewählten Satz von IP-Adressen beschränken können.

Die neuesten Sicherheitsupdates für benutzerdefinierte Images erhalten

Um benutzerdefinierte Images mit Amazon EMR in EKS zu verwenden, können Sie beliebige Binärdateien und Bibliotheken auf dem Image installieren. Sie sind dafür verantwortlich, die Binärdateien, die Sie dem Image hinzufügen, mit Sicherheitspatches zu aktualisieren. Amazon EMR in EKS-Images werden regelmäßig mit den neuesten Sicherheitspatches gepatcht. Um das neueste Image zu erhalten, müssen Sie die benutzerdefinierten Images immer dann neu erstellen, wenn es eine neue Basis-Image-Version der Amazon-EMR-Version gibt. Weitere Informationen finden Sie unter Versionen von Amazon EMR in EKS und Wie wählt man einen Basis-Image-URI aus.

Beschränken Sie den Zugriff auf Pod-Anmeldeinformationen

Kubernetes unterstützt mehrere Methoden, um einem Pod Anmeldeinformationen zuzuweisen. Die Bereitstellung mehrerer Anbieter von Anmeldeinformationen kann die Komplexität Ihres Sicherheitsmodells erhöhen. Amazon EMR in EKS hat die Verwendung von IAM-Rollen für Servicekonten (IRSA) als Standardanbieter für Anmeldeinformationen in einem registrierten EKS-Namespace eingeführt. Andere Methoden werden nicht unterstützt, darunter kube2iam, kiam und die Verwendung eines EC2-Instance-Profils der Instance, die auf dem Cluster ausgeführt wird.

Den Code nicht vertrauenswürdiger Anwendungen isolieren

Amazon EMR in EKS überprüft nicht die Integrität des Anwendungscodes, der von Benutzern des Systems eingereicht wurde. Wenn Sie einen virtuellen Cluster mit mehreren Mandanten ausführen, der mit mehreren Ausführungsrollen konfiguriert ist, die zum Senden von Aufträgen durch nicht vertrauenswürdige Mandanten verwendet werden können, die beliebigen Code ausführen, besteht die Gefahr, dass eine böswillige Anwendung ihre Rechte ausweitet. In diesem Fall sollten Sie erwägen, Ausführungsrollen mit ähnlichen Rechten in einem anderen virtuellen Cluster zu isolieren.

Rollenbasierte Zugriffskontrolle (RBAC)

Administratoren sollten die RBAC-Berechtigungen (rollenbasierte Zugriffskontrolle) für Amazon EMR auf von EKS verwalteten Namespaces strikt kontrollieren. Einreichern von Stellenangeboten in von Amazon EMR in EKS verwalteten Namespaces sollten mindestens die folgenden Berechtigungen nicht gewährt werden.

- Kubernetes-RBAC-Berechtigungen zum Ändern der Configmap – weil Amazon EMR in EKS Kubernetes-Configmaps verwendet, um verwaltete Pod-Vorlagen zu generieren, die den Namen des verwalteten Servicekontos haben. Dieses Attribut sollte nicht mutiert werden.
- Kubernetes-RBAC-Berechtigungen für die Ausführung in Amazon EMR in EKS-Pods – um zu verhindern, dass Zugriff auf verwaltete Pod-Vorlagen gewährt wird, die den verwalteten SA-Namen haben. Dieses Attribut sollte nicht mutiert werden. Diese Berechtigung kann auch Zugriff auf das im Pod montierte JWT-Token gewähren, das dann zum Abrufen der Anmeldeinformationen für die Ausführungsrolle verwendet werden kann.
- Kubernetes-RBAC-Berechtigungen zum Erstellen von Pods – um zu verhindern, dass Benutzer Pods mit einem Kubernetes ServiceAccount erstellen, der einer IAM-Rolle mit mehr AWS-Rechten als der Benutzer zugeordnet sein kann.
- Kubernetes-RBAC-Berechtigungen zur Bereitstellung mutierender Webhooks – um zu verhindern, dass Benutzer den mutierenden Webhook verwenden, um den Kubernetes ServiceAccount-Namen für Pods zu mutieren, die von Amazon EMR in EKS erstellt wurden.
- Kubernetes-RBAC-Berechtigungen zum Lesen von Kubernetes-Geheimnissen – um zu verhindern, dass Benutzer vertrauliche Daten lesen, die in diesen Geheimnissen gespeichert sind.

Den Zugriff auf Anmeldeinformationen für Knotengruppen, IAM-Rollen oder Instance-Profile beschränken

- Wir empfehlen, den IAM-Rollen von Nodegroup AWS Mindestberechtigungen zuzuweisen. Dies trägt dazu bei, eine Eskalation von Rechten durch Code zu vermeiden, der möglicherweise mit den Anmeldeinformationen für das Instance-Profil von EKS-Worker-Knoten ausgeführt wird.
- Um den Zugriff auf die Anmeldeinformationen des Instance-Profils für alle Pods, die in Amazon EMR auf von EKS verwalteten Namespaces ausgeführt werden, vollständig zu blockieren, empfehlen wir, `iptables` Befehle auf EKS-Knoten auszuführen. Weitere Informationen finden Sie unter Einschränken des Zugriffs auf Anmeldeinformationen des Amazon-EC2-Instance-Profils. Es ist jedoch wichtig, Ihre IAM-Servicekontorollen ordnungsgemäß zu definieren, damit Ihre Pods über alle erforderlichen Berechtigungen verfügen. Beispielsweise werden der IAM-Knotenrolle Berechtigungen zugewiesen, um Container-Abbilder von Amazon ECR abzurufen. Wenn einem Pod diese Berechtigungen nicht zugewiesen wurden, kann der Pod keine Container-Abbilder von Amazon ECR abrufen. Das VPC-CNI-Plugin muss ebenfalls aktualisiert werden. Weitere Informationen finden Sie unter Exemplarische Vorgehensweise: Aktualisieren des VPC-CNI-Plug-Ins zur Verwendung von IAM-Rollen für Servicekonten.

Datenschutz

Das AWS-Modell der geteilten Verantwortlichkeit wird auf den Datenschutz in Amazon EMR in EKS angewendet. Wie in diesem Modell beschrieben, ist AWS für den Schutz der globalen Infrastruktur verantwortlich, auf der die gesamte AWS-Cloud läuft. Sie sind dafür verantwortlich, die Kontrolle über Ihre in dieser Infrastruktur gehosteten Inhalte zu behalten. Dieser Inhalt enthält die Sicherheitskonfigurations- und Verwaltungsaufgaben für die von Ihnen verwendeten AWS-Services. Weitere Informationen zum Datenschutz finden Sie unter Häufig gestellte Fragen zum Datenschutz. Informationen zum Datenschutz in Europa finden Sie im Blog-Beitrag AWS-Modell der geteilten Verantwortung und die GDPR im Blog zur AWS-Sicherheit.

Aus Datenschutzgründen empfehlen wir Ihnen, die Anmeldeinformationen für AWS-Konten zu schützen und individuelle Konten mit AWS Identity and Access Management (IAM) einzurichten. So erhält jeder Benutzer nur die Berechtigungen, die zum Durchführen seiner Aufgaben erforderlich sind. Außerdem sollten Sie die Daten mit folgenden Methoden schützen:

- Verwenden Sie für jedes Konto die Multi-Faktor Authentifizierung (MFA).
- Verwenden Sie SSL/TLS für die Kommunikation mit AWS-Ressourcen. Wir empfehlen TLS 1.2 oder höher.
- Richten Sie die API und die Protokollierung von Benutzeraktivitäten mit ein AWS CloudTrail.
- Verwenden Sie AWS-Verschlüsselungslösungen zusammen mit allen Standardsicherheitskontrollen in AWS-Services.
- Verwenden Sie erweiterte verwaltete Sicherheitsservices wie Amazon Macie, die dabei helfen, in Amazon S3 gespeicherte persönliche Daten zu erkennen und zu sichern.
- Verwenden Sie Amazon EMR in EKS Verschlüsselungsoptionen für die Verschlüsselung von Daten im Ruhezustand bei der Speicherung und während der Übertragung.
- Wenn Sie für den Zugriff auf AWS über eine Befehlszeilenschnittstelle oder über eine API FIPS 140-2-validierte kryptografische Module benötigen, verwenden Sie einen FIPS-Endpunkt. Weitere Informationen über verfügbare FIPS-Endpunkte 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 in EKS oder anderen AWS-Services über die Konsole, API, AWS CLI oder AWS SDKs arbeiten. Alle Daten, die Sie in der Amazon EMR in EKS oder andere Services eingeben, können

in Diagnoseprotokolle aufgenommen werden. Wenn Sie eine URL für einen externen Server bereitstellen, schließen Sie keine Anmeldeinformationen zur Validierung Ihrer Anforderung an den betreffenden Server in die URL ein.

Verschlüsselung im Ruhezustand

Die Datenverschlüsselung verhindert, dass nicht autorisierte Benutzer Daten auf einem Cluster und in den dazugehörigen Datenspeichersystemen lesen können. Dies gilt für auf persistenten Medien gespeicherte Daten, auch als Daten im Ruhezustand bezeichnet, und für Daten, die während der Übertragung im Netzwerk möglicherweise abgefangen werden, auch als Daten während der Übertragung bezeichnet.

Die Datenverschlüsselung erfordert Aktivierungsschlüssel und Zertifikate. Sie können wählen zwischen verschiedenen Optionen, einschließlich Schlüsseln, die von AWS Key Management Service verwaltet werden, Schlüsseln, die von Amazon S3 verwaltet werden, sowie Schlüsseln und Zertifikaten, die von Anbietern bereitgestellt werden, die Sie angeben. Wenn Sie AWS KMS als Ihren Schlüsselanbieter auswählen, fallen für die Speicherung und Nutzung der Verschlüsselungsschlüssel Gebühren 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 von EMRFS-Daten in Amazon S3

Die Amazon-S3-Verschlüsselung funktioniert mit EMR File System (EMRFS)-Objekten, die gelesen werden und zu Amazon S3 geschrieben werden. Sie geben serverseitige Verschlüsselung (SSE) von Amazon S3 oder clientseitige Verschlüsselung (CSE) als Standardverschlüsselungsmodus an, 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 Amazon-S3-Verschlüsselung aktiviert ist, verschlüsselt Transport Layer Security (TLS) EMRFS-Objekte bei der Übertragung zwischen EMR-Cluster-Knoten und Amazon S3. Ausführliche Informationen zur Amazon-S3-Verschlüsselung finden Sie unter Schützen von Daten mithilfe von Verschlüsselung im Entwicklerhandbuch für Amazon Simple Storage Service.

> **Note**
>
> Wenn Sie AWS KMS auswählen, fallen für die Speicherung und Nutzung der Verschlüsselungsschlüssel 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 über SSE finden Sie unter Schützen von Daten mit serverseitiger Verschlüsselung im Entwicklerhandbuch zu Amazon Simple Storage Service.

Wenn Sie SSE in Amazon EMR in EKS einrichten, haben Sie die Wahl zwischen zwei verschiedenen Systemen für die Schlüsselverwaltung:

- SSE-S3 – Hierbei verwaltet Amazon S3 die Aktivierungsschlüssel für Sie.
- SSE-KMS – Sie verwenden eine AWS KMS key, um Richtlinien einzurichten, die für Amazon EMR in EKS geeignet sind.

SSE mit vom Kunden bereitgestellten Schlüsseln (SSE-C) ist für Amazon EMR in EKS; nicht verfügbar.

Clientseitige Verschlüsselung für Amazon S3

Mit Amazon S3 bei der clientseitigen Verschlüsselung erfolgt der Amazon-S3-Ver- und Entschlüsselungsvorgang im EMRFS-Client auf Ihrem EMR-Cluster. Objekte werden vor dem Hochladen nach Amazon S3 verschlüsselt und nach dem Herunterladen entschlüsselt. Der von Ihnen festgelegte Anbieter stellt den vom Client verwendeten Verschlüsselungsschlüssel bereit. Der Client kann vom AWS KMS bereitgestellte Schlüssel (CSE-KMS) oder eine benutzerdefinierte Java-Klasse verwenden, die den clientseitigen Root-Schlüssel (CSE-C) bereitstellt. Die Verschlüsselungseigenschaften unterscheiden sich geringfügig zwischen CSE-KMS und CSE-C, abhängig vom festgelegten Anbieter und von den Metadaten des Objekts, das entschlüsselt oder verschlüsselt werden soll. Weitere Informationen finden Sie unter Schützen von Daten mit clientseitiger Verschlüsselung im Entwicklerhandbuch von Amazon Simple Storage Service.

> **Note**
>
> Amazon S3 CSE stellt nur sicher, dass EMRFS-Daten, die mit Amazon S3 ausgetauscht werden, verschlüsselt sind. Nicht alle Daten auf den Cluster-Instance-Volumes werden verschlüsselt. Da Hue EMRFS nicht verwendet, werden darüber hinaus Objekte, die vom Hue-S3-Dateibrowser in Amazon S3 geschrieben werden, nicht verschlüsselt.

Verschlüsselung lokaler Datenträger

Apache Spark unterstützt die Verschlüsselung temporärer Daten, die auf lokale Festplatten geschrieben werden. Dies deckt Shuffle-Dateien, Shuffle-Spills und Datenblöcke ab, die sowohl für Caching- als auch für Broadcast-Variablen auf der Festplatte gespeichert sind. Es gilt nicht für die Verschlüsselung von Ausgabedaten, die von Anwendungen mit APIs wie `saveAsHadoopFile` oder `saveAsTable` generiert werden. Es gilt möglicherweise auch nicht für temporäre Dateien, die explizit vom Benutzer erstellt wurden. Weitere Informationen finden Sie unter Lokale Speicherverschlüsselung in der Spark-Dokumentation. Spark unterstützt keine verschlüsselten Daten auf der lokalen Festplatte, wie z. B. Zwischendaten, die von einem Executor-Prozess auf eine lokale Festplatte geschrieben werden, wenn die Daten nicht in den Arbeitsspeicher passen. Daten, die dauerhaft auf der Festplatte gespeichert werden, sind auf die Laufzeit des Aufträge beschränkt, und der Schlüssel, der zum Verschlüsseln der Daten verwendet wird, wird von Spark bei jeder Auftragausführung dynamisch generiert. Sobald der Spark-Auftrag beendet ist, kann kein anderer Prozess die Daten entschlüsseln.

Für den Treiber- und den Ausführer-Pod verschlüsseln Sie Daten im Ruhezustand, die auf dem bereitgestellten Volume gespeichert werden. Es gibt drei verschiedene AWS native Speicheroptionen, die Sie mit Kubernetes verwenden können: EBS, EFS und FSx für Lustre. Alle drei bieten Verschlüsselung im Ruhezustand mit einem vom Service verwalteten Schlüssel oder einem AWS KMS key. Weitere Informationen finden Sie unter EKS-Leitfaden für bewährte Methoden. Bei diesem Ansatz werden alle Daten, die auf dem bereitgestellten Volume gespeichert werden, verschlüsselt.

Schlüsselverwaltung

Sie können KMS so konfigurieren, dass Ihre KMS-Schlüssel automatisch rotiert werden. Dadurch werden Ihre Schlüssel einmal im Jahr rotiert, während alte Schlüssel auf unbestimmte Zeit gespeichert werden, sodass Ihre Daten weiterhin entschlüsselt werden können. Weitere Informationen finden Sie unter Rotieren von AWS KMS keys.

Verschlüsselung während der Übertragung

Bei der Verschlüsselung während der Übertragung sind mehrere Verschlüsselungsmechanismen aktiviert. Dabei handelt es sich um Open-Source-Features, die anwendungsspezifisch sind und je nach Amazon EMR in EKS Version variieren können. Die folgenden anwendungsspezifischen Verschlüsselungsfeatures können mit Amazon EMR in EKS aktiviert werden:

- Spark
  - Interne RPC-Kommunikationen zwischen Spark-Komponenten, z. B. dem Blocktransferdienst und dem externen Shuffle-Service, werden in Amazon-EMR-Version 5.9.0 und höher mit der AES-256-Bit-Verschlüsselung verschlüsselt. In früheren Versionen werden interne RPC-Kommunikationen mithilfe des Verschlüsselungsverfahrens SASL mit DIGEST-MD5 verschlüsselt.
  - HTTP-Protokollkommunikationen mit Benutzeroberflächen wie Spark History Server und HTTPS-fähigen Dateiservern werden mithilfe der SSL-Konfiguration von Spark verschlüsselt. Weitere Informationen finden Sie unter SSL Configuration in der Spark-Dokumentation.

  Weitere Informationen finden Sie unter Spark-Sicherheitseinstellungen.
- Sie sollten nur verschlüsselte Verbindungen über HTTPS (TLS) unter Anwendung der Bedingung aws:SecureTransport auf Amazon-S3-Bucket IAM-Richtlinien zulassen.
- Abfrageergebnisse, die zu JDBC- oder ODBC-Clients gestreamt werden, werden mit TLS verschlüsselt.

Identitäts- und Zugriffsverwaltung

AWS Identity and Access Management (IAM) ist ein AWS-Service, mit dem Administratoren den Zugriff auf AWS-Ressourcen sicher steuern können. IAM-Administratoren kontrollieren, wer authentifiziert (angemeldet) und autorisiert (mit Berechtigungen ausgestattet) werden kann, um Amazon-EMR-EKS-Ressourcen zu nutzen. IAM ist ein AWS-Service, den Sie ohne zusätzliche Kosten verwenden können.

Themen

- Zielgruppe
- Authentifizierung mit Identitäten
- Verwalten des Zugriffs mit Richtlinien
- Funktionsweise von Amazon EMR in EKS mit IAM

- Verwendung von serviceverknüpften Rollen für Amazon EMR in EKS
- Verwaltete Richtlinien für Amazon EMR in EKS
- Auftragausführungsrollen mit Amazon EMR in EKS verwenden
- Identitätsbasierte Beispiele für Amazon EMR in EKS-Richtlinien
- Richtlinien für Tag-basierte Zugriffskontrolle
- Fehlerbehebung für Identität und Zugriff in Amazon EMR in EKS

Zielgruppe

Wie Sie AWS Identity and Access Management (IAM) verwenden, unterscheidet sich je nach Ihrer Arbeit in Amazon EMR in EKS.

Service-Benutzer – Wenn Sie den Amazon EMR in EKS-Service zur Ausführung von Aufgaben verwenden, stellt Ihnen Ihr Administrator die Anmeldeinformationen und Berechtigungen bereit, die Sie benötigen. Je mehr Amazon-EMR-in-EKS-Features Sie für Ihre Arbeit nutzen, desto mehr Berechtigungen benötigen Sie möglicherweise. Wenn Sie die Featuresweise der Zugriffskontrolle nachvollziehen, wissen Sie bereits, welche Berechtigungen Sie von Ihrem Administrator anzufordern müssen. Unter Fehlerbehebung für Identität und Zugriff in Amazon EMR in EKS finden Sie nützliche Informationen für den Fall, dass Sie keinen Zugriff auf ein Feature in Amazon EMR in EKS haben.

Service-Administrator – Wenn Sie in Ihrem Unternehmen für die Ressourcen von Amazon EMR in EKS zuständig sind, haben Sie wahrscheinlich vollen Zugriff auf Amazon EKS. Ihre Aufgabe besteht darin, zu bestimmen, auf welche Features und Ressourcen von Amazon EMR in EKS Ihre Servicebenutzer zugreifen sollen. Sie müssen dann Anträge an Ihren IAM-Administrator stellen, um die Berechtigungen Ihrer Servicenutzer zu ändern. Lesen Sie die Informationen auf dieser Seite, um die Grundkonzepte von IAM nachzuvollziehen. Weitere Informationen dazu, wie Ihr Unternehmen IAM mit Amazon EMR in EKS verwenden kann, finden Sie unter Funktionsweise von Amazon EMR in EKS mit IAM.

IAM-Administrator - Wenn Sie ein IAM-Administrator sind, möchten Sie vielleicht Details darüber erfahren, wie Sie Richtlinien zur Verwaltung des Zugriffs auf Amazon EMR in EKS erstellen können. Beispiele für identitätsbasierte Richtlinien von Amazon EMR in EKS, die Sie in IAM verwenden können, finden Sie unter Identitätsbasierte Beispiele für Amazon EMR in EKS-Richtlinien.

Authentifizierung mit Identitäten

Authentifizierung ist die Art, wie Sie sich mit Ihren Anmeldeinformationen bei AWS anmelden. Die Authentifizierung (Anmeldung bei AWS) muss als Root-Benutzer des AWS-Kontos, als IAM-Benutzer oder durch Übernahme einer IAM-Rolle erfolgen.

Sie können sich bei AWS als Verbundidentität mit Anmeldeinformationen anmelden, die über eine Identitätsquelle bereitgestellt werden. Benutzer von AWS IAM Identity Center. (IAM Identity Center), die Single-Sign-on-Authentifizierung Ihres Unternehmens und Anmeldeinformationen für Google oder Facebook sind Beispiele für Verbundidentitäten. Wenn Sie sich als Verbundidentität anmelden, hat der Administrator vorher mithilfe von IAM-Rollen einen Identitätsverbund eingerichtet. Wenn Sie auf AWS mithilfe des Verbunds zugreifen, übernehmen Sie indirekt eine Rolle.

Je nachdem, welcher Benutzertyp Sie sind, können Sie sich bei der AWS Management Console oder beim AWS-Zugriffsportal anmelden. Weitere Informationen zum Anmelden bei AWS finden Sie unter So melden Sie sich bei Ihrem AWS-Konto an im Benutzerhandbuch von AWS-Anmeldung.

Bei programmgesteuerten Zugriff auf AWS bietet AWS ein Software Development Kit (SDK) und eine Command Line Interface (CLI, Befehlszeilenschnittstelle) zum kryptographischen Signieren Ihrer Anforderungen mit Ihren Anmeldeinformationen. Wenn Sie keine AWS-Tools verwenden, müssen Sie Anforderungen selbst signieren. Weitere Informationen zur Verwendung der empfohlenen Methode zum eigenen Signieren von Anforderungen finden Sie unter Signieren von AWS-API-Anforderungen im IAM-Benutzerhandbuch.

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

AWS-Konto-Root-Benutzer

Wenn Sie ein AWS-Konto neu erstellen, beginnen Sie mit einer Anmeldeidentität, die vollständigen Zugriff auf alle AWS-Services und Ressourcen des Kontos hat. Diese Identität wird als AWS-Konto-Root-Benutzer bezeichnet. Für den Zugriff auf den Root-Benutzer müssen Sie sich mit der E-Mail-Adresse und dem Passwort anmelden, die zur Erstellung des Kontos verwendet wurden. 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 unter Aufgaben, die Root-Benutzer-Anmeldeinformationen erfordern im IAM-Benutzerhandbuch.

Verbundidentität

Als bewährte Methode empfiehlt es sich, menschliche Benutzer, einschließlich Benutzer, die Administratorzugriff benötigen, aufzufordern, den Verbund mit einem Identitätsanbieter zu verwenden, um auf AWS-Services mit temporären Anmeldeinformationen zuzugreifen.

Eine Verbundidentität ist ein Benutzer aus dem Benutzerverzeichnis Ihres Unternehmens, ein Web Identity Provider, AWS Directory Service, das Identity-Center-Verzeichnis oder jeder Benutzer, der mit Anmeldeinformationen, die über eine Identitätsquelle bereitgestellt werden, auf AWS-Services zugreift. Wenn Verbundidentitäten auf AWS-Konten zugreifen, übernehmen sie Rollen und die Rollen stellen temporäre Anmeldeinformationen bereit.

Für die zentrale Zugriffsverwaltung empfehlen wir Ihnen, AWS IAM Identity Center zu verwenden. Sie können Benutzer und Gruppen im IAM Identity Center erstellen oder Sie können eine Verbindung mit einer Gruppe von Benutzern und Gruppen in Ihrer eigenen Identitätsquelle herstellen und synchronisieren, um sie in allen AWS-Konten und Anwendungen zu verwenden. Informationen zu IAM Identity Center finden Sie unter Was ist IAM Identity Center? im AWS IAM Identity Center-Benutzerhandbuch.

IAM-Benutzer und -Gruppen

Ein IAM-Benutzer ist eine Identität in Ihrem AWS-Konto mit bestimmten Berechtigungen für eine einzelne Person oder eine einzelne Anwendung. Wenn möglich, empfehlen wir, temporäre Anmeldeinformationen zu verwenden, anstatt IAM-Benutzer zu erstellen, die langfristige Anmeldeinformationen wie Passwörter und Zugriffsschlüssel haben. Bei speziellen Anwendungsfällen, die langfristige Anmeldeinformationen mit IAM-Benutzern erfordern, empfehlen wir jedoch, die Zugriffsschlüssel zu rotieren. Weitere Informationen finden Sie unter Regelmäßiges Rotieren von Zugriffsschlüsseln für Anwendungsfälle, die langfristige Anmeldeinformationen erfordern im IAM-Benutzerhandbuch.

Eine IAM-Gruppe 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 einer Gruppe mit dem Namen IAMAdmins Berechtigungen zum Verwalten 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 Erstellen eines IAM-Benutzers (anstatt einer Rolle) im IAM-Benutzerhandbuch.

IAM-Rollen

Eine IAM-Rolle ist eine Identität in Ihrem AWS-Konto mit spezifischen Berechtigungen. Sie ist einem IAM-Benutzer vergleichbar, ist aber nicht mit einer bestimmten Person verknüpft. Sie können vorübergehend eine IAM-Rolle in der AWS Management Console übernehmen, indem Sie Rollen wechseln. Sie können eine Rolle annehmen, indem Sie eine AWS CLI oder AWS-API-Operation aufrufen oder eine benutzerdefinierte URL verwenden. Weitere Informationen zu Methoden für die Verwendung von Rollen finden Sie unter Verwenden von IAM-Rollen im IAM-Benutzerhandbuch.

IAM-Rollen mit temporären Anmeldeinformationen sind in folgenden Situationen hilfreich:

- 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 unter Erstellen von Rollen für externe Identitätsanbieter im IAM-Benutzerhandbuch. Wenn Sie IAM Identity Center verwenden, konfigurieren Sie einen Berechtigungssatz. Wenn Sie steuern möchten, 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-Benutzerhandbuch.
- Temporäre IAM-Benutzerberechtigungen – Ein IAM-Benutzer oder eine -Rolle kann eine IAM-Rolle übernehmen, um vorübergehend andere Berechtigungen für eine bestimmte Aufgabe zu erhalten.
- Kontoübergreifender Zugriff – Sie können eine IAM-Rolle verwenden, um einem vertrauenswürdigen Prinzipal in einem anderen Konto den Zugriff auf Ressourcen in Ihrem Konto zu ermöglichen. Rollen stellen die primäre Möglichkeit dar, um kontoübergreifendem Zugriff zu gewähren. In einigen AWS-Services können Sie jedoch eine Richtlinie direkt an eine Ressource anfügen (anstatt eine Rolle als Proxy zu verwenden). Informationen zu den Unterschieden zwischen Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter So unterscheiden sich IAM-Rollen von ressourcenbasierten Richtlinien im IAM-Benutzerhandbuch.

- Serviceübergreifender Zugriff – Einige AWS-Services verwenden Features in anderen AWS-Services. Wenn Sie beispielsweise einen Aufruf in einem Service tätigen, führt dieser Service häufig Anwendungen in Amazon EC2 aus oder speichert Objekte in Amazon S3. Ein Dienst kann dies mit den Berechtigungen des aufrufenden Prinzipals mit einer Servicerolle oder mit einer serviceverknüpften Rolle tun.
  - Wenn Sie einen IAM-Benutzer oder eine IAM-Rolle zum Ausführen von Aktionen in Amazon Managed Service for Prometheus verwenden, gelten Sie als Prinzipal. Bei einigen Services könnte es Aktionen geben, die dann eine andere Aktion in einem anderen Service initiieren. FAS verwendet die Berechtigungen des Prinzipals, der einen AWS-Service aufruft, in Kombination mit der Anforderung an den AWS-Service, Anforderungen an nachgelagerte Services zu stellen. FAS-Anforderungen werden nur dann gestellt, wenn ein Dienst eine Anforderung erhält, die Interaktionen mit anderen AWS-Services oder Ressourcen erfordert, um abgeschlossen werden zu können. 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 IAM-Rolle, die ein Service übernimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM-Administrator kann eine Servicerolle innerhalb von IAM erstellen, ändern und löschen. Weitere Informationen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service im IAM-Benutzerhandbuch.
  - Serviceverknüpfte Rolle – Eine serviceverknüpfte Rolle ist ein Typ von Servicerolle, die mit einem AWS-Service verknüpft ist. Der Service kann die Rolle übernehmen, um eine Aktion in Ihrem Namen auszuführen. Serviceverknüpfte Rollen werden in Ihrem AWS-Konto angezeigt und gehören zum Service. Ein IAM-Administrator kann die Berechtigungen für serviceverbundene Rollen anzeigen, aber nicht bearbeiten.
- Anwendungen in Amazon EC2 – Sie können eine IAM-Rolle verwenden, um temporäre Anmeldeinformationen für Anwendungen zu verwalten, die auf einer EC2-Instance ausgeführt werden und AWS CLI- oder AWS-API-Anforderungen durchführen. Das ist eher zu empfehlen, als Zugriffsschlüssel innerhalb der EC2-Instance zu speichern. Erstellen Sie ein Instance-Profil, das an die Instance angefügt ist, um eine AWS-Rolle einer EC2-Instance zuzuweisen und die Rolle für sämtliche Anwendungen der Instance bereitzustellen. Ein Instance-Profil enthält die Rolle und ermöglicht, dass Programme, die in der EC2-Instance ausgeführt werden, temporäre Anmeldeinformationen erhalten. Weitere Informationen finden Sie unter Verwenden einer IAM-Rolle zum Erteilen von Berechtigungen für Anwendungen, die auf Amazon EC2-Instances ausgeführt werden im IAM-Benutzerhandbuch.

Informationen dazu, wann Sie IAM-Rollen oder IAM-Benutzer verwenden sollten, finden Sie unter Erstellen einer IAM-Rolle (anstatt eines Benutzers) im IAM-Benutzerhandbuch.

Verwalten des Zugriffs mit Richtlinien

Für die Zugriffssteuerung in AWS erstellen Sie Richtlinien und weisen diese den AWS-Identitäten oder -Ressourcen zu. Eine Richtlinie ist ein Objekt in AWS, das, wenn es einer Identität oder Ressource zugeordnet wird, deren Berechtigungen definiert. AWS wertet diese Richtlinien aus, wenn ein Prinzipal (Benutzer, Root-Benutzer oder Rollensitzung) eine Anforderung stellt. Berechtigungen in den Richtlinien bestimmen, ob die Anforderung zugelassen oder abgelehnt wird. Die meisten Richtlinien werden in AWS als JSON-Dokumente gespeichert. Weitere Informationen zu Struktur und Inhalten von JSON-Richtliniendokumenten finden Sie unter Übersicht über JSON-Richtlinien im IAM-Benutzerhandbuch.

Administratoren können mithilfe von AWS-JSON-Richtlinien festlegen, wer zum Zugriff auf was berechtigt ist. Das bedeutet, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Standardmäßig haben Benutzer, Gruppen und Rollen keine Berechtigungen. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern die Berechtigung erteilen, Aktionen für die Ressourcen auszuführen, die sie benötigen. Der Administrator kann dann die IAM-Richtlinien zu Rollen hinzufügen, und Benutzer können die Rollen annehmen.

IAM-Richtlinien definieren Berechtigungen für eine Aktion unabhängig von der Methode, die Sie zur Ausführung der Aktion verwenden. Angenommen, es gibt eine Richtlinie, die Berechtigungen für die `iam:GetRole`-Aktion erteilt. Ein Benutzer mit dieser Richtlinie kann Benutzerinformationen über die AWS Management Console, die AWS CLI oder die AWS -API abrufen.

Identitätsbasierte Richtlinien

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität anfügen können, wie z. B. IAM-Benutzern, -Benutzergruppen oder -Rollen. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Erstellen von IAM-Richtlinien im IAM-Benutzerhandbuch.

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 AWS-Konto anfügen können.

Verwaltete Richtlinien umfassen von AWS verwaltete und von Kunden verwaltete Richtlinien. Informationen dazu, wie Sie zwischen einer verwalteten Richtlinie und einer eingebundenen Richtlinie wählen, finden Sie unter Auswahl zwischen verwalteten und eingebundenen Richtlinien im IAM-Benutzerhandbuch.

Ressourcenbasierte Richtlinien

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele für ressourcenbasierte Richtlinien sind IAM-Rollen-Vertrauensrichtlinien und Amazon-S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Für die Ressource, an welche die Richtlinie angehängt ist, legt die Richtlinie fest, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für diese Ressource ausführen kann. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben. Prinzipale können Konten, Benutzer, Rollen, Verbundbenutzer oder AWS-Services umfassen.

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

Zugriffssteuerungslisten (ACLs)

Zugriffssteuerungslisten (ACLs) steuern, welche Prinzipale (Kontomitglieder, Benutzer oder Rollen) auf eine Ressource zugreifen können. ACLs sind ähnlich wie ressourcenbasierte Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

Amazon S3, AWS WAF und Amazon VPC sind Beispiele für Dienste, die ACLs unterstützen. Weitere Informationen“ zu ACLs finden Sie unter Zugriffskontrollliste (ACL) – Übersicht (Access Control List) im Amazon-Simple-Storage-Service-Entwicklerhandbuch.

Weitere Richtlinientypen

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

- Berechtigungsgrenzen – Eine Berechtigungsgrenze ist ein erweitertes Feature, mit der Sie die maximalen Berechtigungen festlegen können, die eine identitätsbasierte Richtlinie einer IAM-Entität (IAM-Benutzer oder -Rolle) erteilen 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. Ein ausdrückliches Ablehnen in einer dieser Richtlinien setzt das Zulassen außer Kraft. Weitere Informationen über Berechtigungsgrenzen finden Sie unter Berechtigungsgrenzen für IAM-Entitäten im IAM-Benutzerhandbuch.

- Service-Kontrollrichtlinien (SCPs) – SCPs sind JSON-Richtlinien, die die maximalen Berechtigungen für eine Organisation oder Organisationseinheit (OE) in AWS Organizations angeben. AWS Organizations ist ein Dienst für die Gruppierung und zentrale Verwaltung mehrerer AWS-Konten Ihres Unternehmens. Wenn Sie innerhalb einer Organisation alle Features aktivieren, können Sie Service-Kontrollrichtlinien (SCPs) auf alle oder einzelne Ihrer Konten anwenden. SCPs schränken Berechtigungen für Entitäten in Mitgliedskonten einschließlich des jeweiligen Root-Benutzer des AWS-Kontos ein. Weitere Informationen zu Organisationen und SCPs finden Sie unter Funktionsweise von SCPs im AWS Organizations-Benutzerhandbuch.
- 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 unter Sitzungsrichtlinien im IAM-Benutzerhandbuch.

Mehrere Richtlinientypen

Wenn mehrere auf eine Anforderung mehrere Richtlinientypen angewendet werden können, sind die entsprechenden Berechtigungen komplizierter. Informationen dazu, wie AWS die Zulässigkeit einer Anforderung ermittelt, wenn mehrere Richtlinientypen beteiligt sind, finden Sie unter Logik für die Richtlinienauswertung im IAM-Benutzerhandbuch.

Funktionsweise von Amazon EMR in EKS mit IAM

Bevor Sie mit IAM den Zugriff auf Amazon EMR in EKS verwalten können, sollten Sie sich darüber informieren, welche IAM-Features Sie mit Amazon EMR in EKS verwenden können.

IAM-Features, die Sie mit Amazon EMR in EKS verwenden können

IAM-Funktion	Unterstützung für Amazon EMR in EKS
Identitätsbasierte Richtlinien	Ja

IAM-Funktion	Unterstützung für Amazon EMR in EKS
Ressourcenbasierte Richtlinien	Nein
Richtlinienaktionen	Ja
Richtlinienressourcen	Ja
Bedingungsschlüssel für die Richtlinie	Ja
ACLs	Nein
ABAC (Tags in Richtlinien)	Ja
Temporäre Anmeldeinformationen	Ja
Hauptberechtigungen	Ja
Servicerollen	Nein
Serviceverknüpfte Rollen	Ja

Einen Überblick über das Zusammenwirken von Amazon EMR in EKS und anderen AWS-Services mit den meisten IAM-Features finden Sie unter AWS-Services, die mit IAM funktionieren im IAM-Benutzerhandbuch.

Identitätsbasierte Richtlinien für Amazon EMR in EKS

Unterstützt Richtlinien auf Identitätsbasis.	Ja

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität anfügen können, wie z. B. IAM-Benutzern, -Benutzergruppen oder -Rollen. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Erstellen von IAM-Richtlinien im IAM-Benutzerhandbuch.

Mit identitätsbasierten IAM-Richtlinien können Sie angeben, welche Aktionen und Ressourcen zugelassen oder abgelehnt werden. Darüber hinaus können Sie die Bedingungen festlegen,

unter denen Aktionen zugelassen oder abgelehnt werden. Sie können den Prinzipal nicht in einer identitätsbasierten Richtlinie angeben, da er für den Benutzer oder die Rolle gilt, dem er zugeordnet ist. Informationen zu sämtlichen Elementen, die Sie in einer JSON-Richtlinie verwenden, finden Sie in der IAM-Referenz für JSON-Richtlinienelemente im IAM-Benutzerhandbuch.

Identitätsbasierte Beispiele für Amazon EMR in EKS-Richtlinien

Beispiele für identitätsbasierte Richtlinien in Amazon EMR in EKS finden Sie unter Identitätsbasierte Beispiele für Amazon EMR in EKS-Richtlinien.

Ressourcenbasierte Richtlinien in Amazon EMR in EKS

Unterstützt ressourcenbasierte Richtlinien	Nein

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele für ressourcenbasierte Richtlinien sind IAM-Rollen-Vertrauensrichtlinien und Amazon-S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Für die Ressource, an welche die Richtlinie angehängt ist, legt die Richtlinie fest, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für diese Ressource ausführen kann. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben. Prinzipale können Konten, Benutzer, Rollen, Verbundbenutzer oder AWS-Services umfassen.

Um kontoübergreifenden Zugriff zu ermöglichen, können Sie ein gesamtes Konto oder IAM-Entitäten in einem anderen Konto als Prinzipal in einer ressourcenbasierten Richtlinie angeben. Durch das Hinzufügen eines kontoübergreifenden Auftraggebers zu einer ressourcenbasierten Richtlinie ist nur die halbe Vertrauensbeziehung eingerichtet. Wenn sich der Prinzipal und die Ressource in unterschiedlichen AWS-Konten befinden, muss ein IAM-Administrator im vertrauenswürdigen Konto auch der Prinzipalentität (Benutzer oder Rolle) die Berechtigung zum Zugriff auf die Ressource erteilen. 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 unter Wie sich IAM-Rollen von ressourcenbasierten Richtlinien unterscheiden im IAM-Benutzerhandbuch.

Richtlinienaktionen für Amazon EMR in EKS

Unterstützt Richtlinienaktionen	Ja

Administratoren können mithilfe von AWS-JSON-Richtlinien festlegen, wer zum Zugriff auf was berechtigt ist. Das heißt, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Das Element `Action` 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 zugehörige AWS-API-Operation. Es gibt einige Ausnahmen, z. B. Aktionen, die nur mit Genehmigung durchgeführt werden können und für die es keine passende API-Operation gibt. 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 von Aktionen in Amazon EMR in EKS finden Sie unter Aktionen, Ressourcen und Bedingungsschlüssel für Amazon EMR in EKS in der Service-Autorisierungs-Referenz.

Richtlinienaktionen in Amazon EMR in EKS verwenden das folgende Präfix vor der Aktion:

```
emr-containers
```

Um mehrere Aktionen in einer einzigen Anweisung anzugeben, trennen Sie sie mit Kommata:

```
"Action": [
      "emr-containers:action1",
      "emr-containers:action2"
         ]
```

Beispiele für identitätsbasierte Richtlinien in Amazon EMR in EKS finden Sie unter Identitätsbasierte Beispiele für Amazon EMR in EKS-Richtlinien.

Richtlinienressourcen für Amazon EMR in EKS

Unterstützt Richtlinienressourcen	Ja

Administratoren können mithilfe von AWS-JSON-Richtlinien festlegen, wer zum Zugriff auf was berechtigt ist. Das bedeutet die Festlegung, welcher Prinzipal Aktionen für welche Ressourcen unter welchen Bedingungen ausführen kann.

Das JSON-Richtlinienelement `Resource` gibt die Objekte an, auf welche die Aktion angewendet wird. Anweisungen müssen entweder ein – `Resource`oder ein `NotResource`-Element enthalten. Als bewährte Methode geben Sie eine Ressource mit dem zugehörigen Amazon-Ressourcennamen (ARN) an. 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 Ressourcentypen von Amazon EMR in EKS und ihrer ARNs finden Sie unter Von Amazon EMR in EKS definierte Ressourcen in der Service-Autorisierungs-Referenz. Informationen zu den Aktionen, mit denen Sie den ARN jeder Ressource angeben können, finden Sie unter Aktionen, Ressourcen und Bedingungsschlüssel für Amazon EMR in EKS.

Beispiele für identitätsbasierte Richtlinien in Amazon EMR in EKS finden Sie unter Identitätsbasierte Beispiele für Amazon EMR in EKS-Richtlinien.

Richtlinien-Bedingungsschlüssel für Amazon EMR in EKS

Unterstützt servicespezifische Richtlini enbedingungsschlüssel	Ja

Administratoren können mithilfe von AWS-JSON-Richtlinien festlegen, wer zum Zugriff auf was berechtigt ist. 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, wertet AWS diese mittels einer logischen `AND`-Operation aus. Wenn Sie mehrere Werte für einen einzelnen Bedingungsschlüssel angeben, wertet AWS die Bedingung mittels 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. Beispielsweise können Sie einem IAM-Benutzer die Berechtigung für den Zugriff auf eine Ressource nur dann gewähren, wenn sie mit dessen IAM-Benutzernamen gekennzeichnet ist. Weitere Informationen finden Sie unter IAM-Richtlinienelemente: Variablen und Tags im IAM-Benutzerhandbuch.

AWS unterstützt globale Bedingungsschlüssel und servicespezifische Bedingungsschlüssel. Eine Liste aller globalen AWS-Bedingungsschlüssel finden Sie unter Globale AWS-Bedingungskontextschlüssel im IAM-Benutzerhandbuch.

Eine Liste der Bedingungsschlüssel von Amazon EMR in EKS und Informationen darüber, welche Aktionen und Ressourcen Sie mit einem Bedingungsschlüssel verwenden können, finden Sie unter Aktionen, Ressourcen und Bedingungsschlüssel für Amazon EMR in EKS in der Service-Autorisierungs-Referenz.

Beispiele für identitätsbasierte Richtlinien in Amazon EMR in EKS finden Sie unter Identitätsbasierte Beispiele für Amazon EMR in EKS-Richtlinien.

Zugriffssteuerungslisten (ACLs) in Amazon EMR in EKS

Unterstützt ACLs	Nein

Zugriffssteuerungslisten (ACLs) steuern, welche Prinzipale (Kontomitglieder, Benutzer oder Rollen) auf eine Ressource zugreifen können. ACLs sind ähnlich wie ressourcenbasierte Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

Attributbasierte Zugriffskontrolle (ABAC) mit Amazon EMR in EKS

Unterstützt ABAC (Tags in Richtlinien)	Ja

Die attributbasierte Zugriffskontrolle (ABAC) ist eine Autorisierungsstrategie, bei der Berechtigungen basierend auf Attributen definiert werden. In AWS werden diese Attribute als Tags bezeichnet. Sie können Tags an IAM-Entitäten (Benutzer oder Rollen) und mehrere AWS-Ressourcen anfügen. Das Markieren von Entitäten und Ressourcen ist der erste Schritt von ABAC. Anschließend entwerfen Sie ABAC-Richtlinien, um Operationen zuzulassen, wenn das Tag des Prinzipals mit dem Tag der Ressource übereinstimmt, auf die sie zugreifen möchten.

ABAC ist in Umgebungen hilfreich, die schnell wachsen, und unterstützt Sie in Situationen, in denen die Richtlinienverwaltung mühsam 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 `aws:TagKeys`Bedingung verwenden.

Wenn ein Service alle drei Bedingungsschlüssel für jeden Ressourcentyp unterstützt, lautet der Wert für den Service Ja. Wenn ein Service alle drei Bedingungsschlüssel für nur einige Ressourcentypen unterstützt, lautet der Wert Teilweise.

Weitere Informationen zu ABAC finden Sie unter Was ist ABAC? im IAM-Benutzerhandbuch. Um ein Tutorial mit Schritten zur Einstellung von ABAC anzuzeigen, siehe Attributbasierte Zugriffskontrolle (ABAC) verwenden im IAM-Benutzerhandbuch.

Verwenden temporärer Anmeldeinformationen mit Amazon EMR in EKS

Unterstützt temporäre Anmeldeinformationen	Ja

Einige AWS-Services Featureieren nicht, wenn Sie sich mit temporären Anmeldeinformationen anmelden. Weitere Informationen, darunter welche AWS-Services mit temporären Anmeldeinformationen Featureieren, finden Sie unter AWS-Services, die mit IAM Featureieren im IAM-Benutzerhandbuch.

Sie verwenden temporäre Anmeldeinformationen, wenn Sie sich mit einer anderen Methode als einem Benutzernamen und einem Passwort bei der AWS Management Console anmelden. Wenn

Sie beispielsweise über den Single Sign-On (SSO)-Link Ihres Unternehmens auf AWS zugreifen, 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 Wechseln von Rollen finden Sie unter Wechseln zu einer Rolle (Konsole) im IAM-Benutzerhandbuch.

Sie können mithilfe der AWS CLI- oder AWS-API manuell temporäre Anmeldeinformationen erstellen. Sie können dann diese temporären Anmeldeinformationen verwenden, um auf AWS zuzugreifen. AWS empfiehlt, dass Sie temporäre Anmeldeinformationen dynamisch generieren, anstatt langfristige Zugriffsschlüssel zu verwenden. Weitere Informationen finden Sie unter Temporäre Sicherheitsanmeldeinformationen in IAM.

Serviceübergreifende Prinzipalberechtigungen für Amazon EMR in EKS

Unterstützt Forward Access Sessions (FAS)	Ja

Wenn Sie einen IAM-Benutzer oder eine IAM-Rolle zum Ausführen von Aktionen in AWS verwenden, gelten Sie als Prinzipal. Bei einigen Services könnte es Aktionen geben, die dann eine andere Aktion in einem anderen Service auslösen. FAS verwendet die Berechtigungen des Prinzipals, der einen AWS-Service aufruft, in Kombination mit der Anforderung an den AWS-Service, Anforderungen an nachgelagerte Services zu stellen. FAS-Anforderungen werden nur dann gestellt, wenn ein Dienst eine Anforderung erhält, die Interaktionen mit anderen AWS-Services oder Ressourcen erfordert, um abgeschlossen werden zu können. 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 Amazon EMR in EKS

Unterstützt Servicerollen	Nein

Serviceverknüpften Rollen für Amazon EMR in EKS

Unterstützt serviceverknüpfte Rollen	Ja

Details zum Erstellen oder Verwalten von serviceverknüpften Rollen finden Sie unter AWS-Services, 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.

Verwendung von serviceverknüpften Rollen für Amazon EMR in EKS

Amazon EMR in EKS verwendet AWS Identity and Access Management (IAM) serviceverknüpfte Rollen. Eine servicegebundene Rolle ist eine einzigartige Art von IAM-Rolle, die direkt mit Amazon EMR in EKS verknüpft ist. Serviceverknüpfte Rollen werden von Amazon EMR in EKS vordefiniert und schließen alle Berechtigungen ein, die der Service zum Aufrufen anderer AWS-Services in Ihrem Namen erfordert.

Eine serviceverknüpfte Rolle macht die Einrichtung von Amazon EMR in EKS einfacher, da Sie die erforderlichen Berechtigungen nicht manuell hinzufügen müssen. Amazon EMR in EKS definiert die Berechtigungen seiner mit dem Service verbundenen Rollen, und sofern nicht anders definiert, kann nur Amazon EMR in EKS seine Rollen übernehmen. Die definierten Berechtigungen umfassen die Vertrauens- und Berechtigungsrichtlinie. Diese Berechtigungsrichtlinie kann keinen anderen IAM-Entitäten zugewiesen werden.

Sie können eine serviceverknüpfte Rolle erst löschen, nachdem ihre verwandten Ressourcen gelöscht wurden. Dies schützt Ihre Amazon-EMR-in-EKS-Ressourcen, da Sie nicht versehentlich die Zugriffsberechtigung für die Ressourcen entfernen können.

Informationen zu anderen Services, die servicegebundene Rollen unterstützen, finden Sie unter AWS-Services, die mit IAM funktionieren. Suchen Sie nach den Services, für die Ja in der Spalte Servicegebundene Rolle angegeben ist. Wählen Sie über einen Link Ja aus, um die Dokumentation zu einer servicegebundenen Rolle für diesen Service anzuzeigen.

Serviceverknüpfte Rollenberechtigungen für Amazon EMR in EKS

Amazon EMR in EKS verwendet die serviceverknüpfte Rolle mit dem Namen AWSServiceRoleForAmazonEMRContainers.

Die serviceverknüpfte Rolle `AWSServiceRoleForAmazonEMRContainers` vertraut darauf, dass die folgenden Services die Rolle annehmen:

- `emr-containers.amazonaws.com`

Die Rollenberechtigungsrichtlinie `AmazonEMRContainersServiceRolePolicy` ermöglicht es Amazon EMR in EKS, eine Reihe von Aktionen für die angegebenen Ressourcen auszuführen, wie die folgende Richtlinienanweisung zeigt.

> **Note**
>
> Der Inhalt verwalteter Richtlinien kann sich ändern, sodass die hier gezeigte Richtlinie möglicherweise nicht mehr aktuell ist. Sie können die aktuelle AmazonEMRContainersServiceRolePolicy unter AWS Management Console anzeigen.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:DescribeCluster",
                "eks:ListNodeGroups",
                "eks:DescribeNodeGroup",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups",
                "elasticloadbalancing:DescribeInstanceHealth",
                "elasticloadbalancing:DescribeLoadBalancers",
                "elasticloadbalancing:DescribeTargetGroups",
                "elasticloadbalancing:DescribeTargetHealth"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "acm:ImportCertificate",
                "acm:AddTagsToCertificate"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/emr-container:endpoint:managed-certificate": "true"
                }
            }
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "acm:DeleteCertificate"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/emr-container:endpoint:managed-certificate":
 "true"
                }
            }
        }
    ]
}
```

Sie müssen Berechtigungen konfigurieren, damit eine juristische Stelle von IAM (z. B. Benutzer, Gruppe oder Rolle) eine servicegebundene Rolle erstellen, bearbeiten oder löschen kann. Weitere Informationen finden Sie unter serviceverknüpfte Rollenberechtigungen im IAM-Benutzerhandbuch.

Erstellen einer serviceverknüpften Rolle für Amazon EMR in EKS

Sie müssen eine serviceverknüpfte Rolle nicht manuell erstellen. Wenn Sie einen virtuellen Cluster anlegen, legt Amazon EMR in EKS die servicegebundene Rolle wieder für Sie an.

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 einen virtuellen Cluster anlegen, legt Amazon EMR in EKS die servicegebundene Rolle wieder für Sie an.

Sie können auch die IAM-Konsole verwenden, um eine serviceverknüpfte Rolle mit dem Anwendungsfall Amazon EMR in EKS zu erstellen. Erstellen Sie in der AWS CLI oder der AWS-API eine servicegebundene Rolle mit dem Servicenamen `emr-containers.amazonaws.com`. Weitere Informationen finden Sie unter Erstellen einer servicegebundenen Rolle im IAM-Leitfaden. Wenn Sie diese servicegebundene Rolle löschen, können Sie mit demselben Verfahren die Rolle erneut erstellen.

Bearbeiten einer serviceverknüpften Rolle für Amazon EMR in EKS

Amazon EMR in EKS verhindert die Bearbeitung der serviceverknüpften Rolle `AWSServiceRoleForAmazonEMRContainers`. Da möglicherweise verschiedene Entitäten auf

die Rolle verweisen, kann der Rollenname nach dem Erstellen einer serviceverknüpften Rolle nicht mehr geändert werden. Sie können jedoch die Beschreibung der Rolle mit IAM bearbeiten. Weitere Informationen finden Sie unter Bearbeiten einer serviceverknüpften Rolle im IAM-Benutzerhandbuch.

Löschen einer serviceverknüpften Rolle für Amazon EMR in EKS

Wenn Sie ein Feature oder einen Service, die bzw. der eine serviceverknüpften Rolle erfordert, nicht mehr benötigen, sollten Sie diese Rolle löschen. Auf diese Weise haben Sie keine ungenutzte juristische Stelle, die nicht aktiv überwacht oder verwaltet wird. Sie müssen jedoch die Ressourcen für Ihre serviceverknüpften Rolle zunächst bereinigen, bevor Sie sie manuell löschen können.

> **Note**
>
> Wenn der Amazon-EMR-in-EKS-Service die Rolle verwendet, wenn Sie versuchen, die Ressourcen zu löschen, schlägt der Löschvorgang möglicherweise fehl. Wenn dies passiert, warten Sie einige Minuten und versuchen Sie es erneut.

Löschen Sie Ressourcen von Amazon EMR in EKS, die von der **`AWSServiceRoleForAmazonEMRContainers`** verwendet werden, wie folgt

1. Öffnen Sie die Amazon-EMR-Konsole.
2. Wählen Sie einen virtuellen Cluster aus.
3. Klicken Sie auf der Seite `Virtual Cluster` auf Löschen.
4. Wiederholen Sie diesen Vorgang für alle anderen virtuellen Cluster in Ihrem Konto.

So löschen Sie die serviceverknüpfte Rolle mit IAM

Verwenden Sie die IAM-Konsole, AWS CLI- oder AWS-API, um die `AWSServiceRoleForAmazonEMRContainers` serviceverknüpfte Rolle zu löschen. Weitere Informationen finden Sie unter Löschen einer serviceverknüpften Rolle im IAM-Benutzerhandbuch.

Unterstützte Regionen für Amazon EMR in EKS serviceverknüpfte Rollen

Amazon EMR in EKS unterstützt die Verwendung von serviceverknüpften Rollen in allen Regionen, in denen der Service verfügbar ist. Weitere Informationen finden Sie unter Service-Endpunkte und Kontingente von Amazon EMR in EKS.

Verwaltete Richtlinien für Amazon EMR in EKS

Details zu Aktualisierungen von AWS verwalteten Richtlinien für Amazon EMR in EKS seit dem 1. März 2021 anzeigen.

Änderung	Beschreibung	Datum
`AmazonEMRContainersServiceRolePolicy` – Es wurden Berechtigungen zur Beschreibung und Auflistung von Amazon-EKS-Knotengruppen, zur Beschreibung von Load-Balancer-Zielgruppen und zur Beschreibung des Zustands des Load-Balancer-Ziels hinzugefügt.	Die folgenden Berechtigungen werden der Richtlinie hinzugefügt: `eks:ListNodeGroups` , `eks:DescribeNodeGroup` , `elasticloadbalancing:DescribeTargetGroups` , `elasticloadbalancing:DescribeTargetHealth` .	13. März 2023
`AmazonEMRContainersServiceRolePolicy` – Es wurden Berechtigungen zum Importieren und Löschen von Zertifikaten in AWS Certificate Manager hinzugefügt.	Die folgenden Berechtigungen werden der Richtlinie hinzugefügt: `acm:ImportCertificate` , `acm:AddTagsToCertificate` , `acm:DeleteCertificate` .	3. Dezember 2021
Amazon EMR in EKS hat mit der Nachverfolgung von Änderungen begonnen	Amazon EMR in EKS hat mit der Verfolgung von Änderungen für seine AWS-verwalteten Richtlinien begonnen.	1. März 2021

Auftragausführungsrollen mit Amazon EMR in EKS verwenden

Um den `StartJobRun` Befehl zum Senden eines auf einem EKS-Cluster ausgeführten Auftrags zu verwenden, müssen Sie zunächst eine Auftragausführungsrolle integrieren, die mit einem virtuellen Cluster verwendet werden soll. Weitere Informationen finden Sie unter Erstellen

einer Aufgabenausführungsrolle in Einrichten von Amazon EMR in EKS. Sie können auch den Anweisungen im Abschnitt IAM-Rolle für die Auftragsausführung erstellen des Amazon EMR in EKS-Workshops folgen.

Die folgenden Berechtigungen müssen in der Vertrauensrichtlinie für die Rolle „Auftragsausführung“ enthalten sein.

```
  {
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::AWS_ACCOUNT_ID:oidc-provider/OIDC_PROVIDER"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringLike": {
          "OIDC_PROVIDER:sub": "system:serviceaccount:NAMESPACE:emr-containers-sa-*-
*-AWS_ACCOUNT_ID-BASE36_ENCODED_ROLE_NAME"
        }
      }
    }
  ]
}
```

Die Vertrauensrichtlinie im vorherigen Beispiel gewährt Berechtigungen nur für ein von Amazon EMR verwaltetes Kubernetes-Servicekonto mit einem Namen, der dem Muster `emr-containers-sa-*-*-`*`AWS_ACCOUNT_ID`*`-`*`BASE36_ENCODED_ROLE_NAME`* entspricht. Servicekonten mit diesem Muster werden bei der Auftragsübermittlung automatisch erstellt und auf den Namespace beschränkt, in dem Sie den Auftrag einreichen. Diese Vertrauensrichtlinie ermöglicht es diesen Servicekonten, die Ausführungsrolle zu übernehmen und die temporären Anmeldeinformationen der Ausführungsrolle abzurufen. Servicekonten aus einem anderen Amazon-EKS-Cluster oder aus einem anderen Namespace innerhalb desselben EKS-Clusters dürfen die Ausführungsrolle nicht übernehmen.

Sie können den folgenden Befehl ausführen, um die Vertrauensrichtlinie im oben angegebenen Format automatisch zu aktualisieren.

```
aws emr-containers update-role-trust-policy \
```

```
        --cluster-name cluster \
        --namespace namespace \
        --role-name iam_role_name_for_job_execution
```

Steuern des Zugriffs auf die Ausführungsrolle

Ein Administrator für Ihren Amazon-EKS-Cluster kann einen virtuellen Amazon-EMR-in-EKS-Cluster mit mehreren Mandanten erstellen, dem ein IAM-Administrator mehrere Ausführungsrollen hinzufügen kann. Da nicht vertrauenswürdige Mandanten diese Ausführungsrollen verwenden können, um Aufträge einzureichen, die beliebigen Code ausführen, sollten Sie diese Mandanten möglicherweise einschränken, sodass sie keinen Code ausführen können, der die Berechtigungen erhält, die einer oder mehreren dieser Ausführungsrollen zugewiesen wurden. Um die mit einer IAM-Identität verknüpfte IAM-Richtlinie einzuschränken, kann der IAM-Administrator den optionalen Bedingungsschlüssel `emr-containers:ExecutionRoleArn` des Amazon-Ressourcennamen (ARN) verwenden. Diese Bedingung akzeptiert eine Liste von ARNs für Ausführungsrollen, die über Berechtigungen für den virtuellen Cluster verfügen, wie die folgende Berechtigungsrichtlinie zeigt.

```
  {
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "emr-containers:StartJobRun",
      "Resource": "arn:aws:emr-containers:REGION:AWS_ACCOUNT_ID:/
virtualclusters/VIRTUAL_CLUSTER_ID",
      "Condition": {
        "ArnEquals": {
          "emr-containers:ExecutionRoleArn": [
            "execution_role_arn_1",
            "execution_role_arn_2",
            ...
          ]
        }
      }
    }
  ]
}
```

Wenn Sie alle Ausführungsrollen zulassen möchten, die mit einem bestimmten Präfix beginnen, z. B. `MyRole`, können Sie den Bedingungsoperator `ArnEquals` durch den `ArnLike`-Operator ersetzen, und Sie können den `execution_role_arn`-Wert in der Bedingung durch ein Platzhalterzeichen * ersetzen. Zum Beispiel `arn:aws:iam::AWS_ACCOUNT_ID:role/MyRole*`. Alle anderen ARN-Bedingungsschlüssel werden ebenfalls unterstützt.

> **Note**
>
> Mit Amazon EMR in EKS können Sie Ausführungsrollen keine Berechtigungen auf der Grundlage von Tags oder Attributen gewähren. Amazon EMR in EKS unterstützt keine tagbasierte Zugriffskontrolle (TBAC) oder attributbasierte Zugriffskontrolle (ABAC) für Ausführungsrollen.

Identitätsbasierte Beispiele für Amazon EMR in EKS-Richtlinien

Benutzer und Rollen besitzen standardmäßig keine Berechtigungen zum Erstellen oder Ändern von Ressourcen von Amazon EMR in EKS. Sie können auch keine Aufgaben über die AWS Management Console, die AWS Command Line Interface (AWS CLI) oder die AWS-API ausführen. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern die Berechtigung erteilen, Aktionen für die Ressourcen auszuführen, die sie benötigen. Der Administrator kann dann die IAM-Richtlinien zu Rollen hinzufügen, und Benutzer können die Rollen annehmen.

Informationen dazu, wie Sie unter Verwendung dieser beispielhaften JSON-Richtliniendokumente eine identitätsbasierte IAM-Richtlinie erstellen, finden Sie unter Erstellen von IAM-Richtlinien im IAM-Benutzerhandbuch.

Einzelheiten zu den von ACM definierten Aktionen und Ressourcentypen Amazon EMR in EKS, einschließlich des Formats der ARNs für die einzelnen Ressourcentypen, finden Sie unter Aktionen, Ressourcen und Bedingungsschlüssel für Amazon EMR in EKS in der Service-Autorisierungs-Referenz.

Themen

- Bewährte Methoden für Richtlinien
- Verwenden der Amazon EMR in EKS-Konsole
- Gewähren der Berechtigung zur Anzeige der eigenen Berechtigungen für Benutzer

Bewährte Methoden für Richtlinien

Identitätsbasierte Richtlinien können festlegen, ob jemand Amazon EMR in EKS-Ressourcen in Ihrem Konto erstellen, darauf zugreifen oder daraus löschen kann. Dies kann zusätzliche Kosten für Ihr verursachen AWS-Konto. Befolgen Sie beim Erstellen oder Bearbeiten identitätsbasierter Richtlinien die folgenden Anleitungen und Empfehlungen:

- Erste Schritte mit AWS-verwaltete Richtlinien und Umstellung auf Berechtigungen mit den geringsten Berechtigungen – Um Ihren Benutzern und Workloads Berechtigungen zu gewähren, verwenden Sie die AWS-verwaltete Richtlinien die Berechtigungen für viele allgemeine Anwendungsfälle gewähren. Sie sind in Ihrem AWS-Konto verfügbar. Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie AWS-kundenverwaltete Richtlinien definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind. Weitere Informationen finden Sie unter AWS-verwaltete Richtlinien oder AWS-verwaltete Richtlinien für Auftragsfunktionen im IAM-Benutzerhandbuch.
- Anwendung von Berechtigungen mit den geringsten Rechten – Wenn Sie mit IAM-Richtlinien Berechtigungen festlegen, gewähren Sie nur die Berechtigungen, die für die Durchführung einer Aufgabe erforderlich sind. Sie tun dies, indem Sie die Aktionen definieren, die für bestimmte Ressourcen unter bestimmten Bedingungen durchgeführt werden können, auch bekannt als die geringsten Berechtigungen. Weitere Informationen zur Verwendung von IAM zum Anwenden von Berechtigungen finden Sie unter Richtlinien und Berechtigungen in IAM im IAM-Benutzerhandbuch.
- Verwenden von Bedingungen in IAM-Richtlinien zur weiteren Einschränkung des Zugriffs – Sie können Ihren Richtlinien eine Bedingung hinzufügen, um den Zugriff auf Aktionen und Ressourcen zu beschränken. Sie können beispielsweise eine Richtlinienbedingung schreiben, um festzulegen, dass alle Anforderungen mithilfe von SSL gesendet werden müssen. Sie können auch Bedingungen verwenden, um Zugriff auf Service-Aktionen zu gewähren, wenn diese durch ein bestimmtes AWS-Service, wie beispielsweise AWS CloudFormation, verwendet werden. Weitere Informationen finden Sie unter IAM-JSON-Richtlinienelemente: Bedingung im IAM-Benutzerhandbuch.
- Verwenden von IAM Access Analyzer zur Validierung Ihrer IAM-Richtlinien, um sichere und funktionale Berechtigungen zu gewährleisten – IAM Access Analyzer validiert neue und vorhandene Richtlinien, damit die Richtlinien der IAM-Richtliniensprache (JSON) und den bewährten IAM-Methoden entsprechen. IAM Access Analyzer stellt mehr als 100 Richtlinienprüfungen und umsetzbare Empfehlungen zur Verfügung, damit Sie sichere und funktionale Richtlinien erstellen können. Weitere Informationen finden Sie unter Richtlinienvalidierung zum IAM Access Analyzer im IAM-Benutzerhandbuch.

- Bedarf einer Multi-Faktor-Authentifizierung (MFA) – Wenn Sie ein Szenario haben, das IAM-Benutzer oder Root-Benutzer in Ihrem AWS-Konto erfordert, aktivieren Sie MFA für zusätzliche Sicherheit. Um MFA beim Aufrufen von API-Vorgängen anzufordern, fügen Sie Ihren Richtlinien MFA-Bedingungen hinzu. Weitere Informationen finden Sie unter Konfigurieren eines MFA-geschützten API-Zugriffs im IAM-Benutzerhandbuch.

Weitere Informationen zu bewährten Methoden in IAM finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.

Verwenden der Amazon EMR in EKS-Konsole

Um auf die Amazon EMR in EKS-Konsole zugreifen zu können, müssen Sie über einen Mindestsatz von Berechtigungen verfügen. Diese Berechtigungen müssen es Ihnen ermöglichen, Details über die Amazon EMR in EKS-Ressourcen in Ihrem AWS-Konto-Konto aufzulisten und anzuzeigen. Wenn Sie eine identitätsbasierte Richtlinie erstellen, die strenger ist als die mindestens erforderlichen Berechtigungen, funktioniert die Konsole nicht wie vorgesehen für Entitäten (Benutzer oder Rollen) mit dieser Richtlinie.

Für Benutzer, die nur Aufrufe an die AWS CLI oder AWS-API durchführen, müssen Sie keine Mindestberechtigungen in der Konsole erteilen. Stattdessen sollten Sie nur Zugriff auf die Aktionen zulassen, die der API-Operation entsprechen, die die Benutzer ausführen möchten.

Um sicherzustellen, dass Benutzer und Rollen weiterhin die Amazon-EMR in EKS-Konsole verwenden können, fügen Sie den Entitäten auch die von Amazon EMR in EKS AWS-verwaltete Richtlinie `ConsoleAccess` oder `ReadOnly` hinzu. Weitere Informationen finden Sie unter Hinzufügen von Berechtigungen zu einem Benutzer im IAM-Benutzerhandbuch.

Gewähren der Berechtigung zur Anzeige der eigenen Berechtigungen für Benutzer

In diesem Beispiel wird gezeigt, wie Sie eine Richtlinie erstellen, die IAM-Benutzern die Berechtigung zum Anzeigen der eingebundenen Richtlinien und verwalteten Richtlinien gewährt, die ihrer Benutzeridentität angefügt sind. Diese Richtlinie enthält Berechtigungen für die Ausführung dieser Aktion auf der Konsole oder für die programmgesteuerte Ausführung über die AWS CLI oder die 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": "*"
        }
    ]
}
```

Richtlinien für Tag-basierte Zugriffskontrolle

Sie können in Ihrer identitätsbasierten Richtlinie auf Tags basierende Bedingungen zum Steuern des Zugriffs auf virtuelle Cluster und Auftragsausführungen zu steueren. Weitere Informationen über das Markieren mit Tags finden Sie unter Kennzeichnen Ihrer Ressourcen von Amazon EMR in EKS.

Die folgenden Beispiele zeigen verschiedene Szenarien und Möglichkeiten zur Nutzung der Bedingungsoperatoren mit Bedingungskontextschlüsseln von Amazon EMR in EKS. Diese IAM-Richtlinienanweisungen dienen nur zu Demonstrationszwecken und sollten nicht in Produktionsumgebungen verwendet werden. Es gibt mehrere Möglichkeiten für die Kombination von Richtlinienanweisungen zum Gewähren oder Verweigern von Berechtigungen entsprechend Ihren

Anforderungen. Weitere Informationen zum Planen und Testen von IAM-Richtlinien finden Sie im IAM-Benutzerhandbuch.

> **Important**
>
> Das explizite Ablehnen von Berechtigungen für Markierungsaktionen stellt eine wichtige Überlegung dar. Dadurch wird verhindert, dass Benutzer eine Ressource markieren und sich dadurch selbst Berechtigungen erteilen, die Sie nicht gewähren wollten. Wenn Tagging-Aktionen für eine Ressource nicht verweigert werden, kann ein Benutzer Tags ändern und so die Absicht der tagbasierten Richtlinien umgehen. Ein Beispiel für eine Richtlinie, die Tagging-Aktionen verweigert, finden Sie unter Zugriff zum Hinzufügen und Entfernen von Tags verweigern.

Die folgenden Beispiele zeigen identitätsbasierte Berechtigungsrichtlinien, die verwendet werden, um die Aktionen zu steuern, die mit virtuellen Clustern von Amazon EMR in EKS zulässig sind.

Erlauben Sie Aktionen nur für Ressourcen mit bestimmten Tag-Werten

Im folgenden Richtlinienbeispiel versucht der StringEquals-Bedingungsoperator, dev mit dem Wert für das Tag department abzugleichen. Wenn das Tag department dem virtuellen Cluster nicht hinzugefügt wurde oder den Wert dev nicht enthält, ist die Richtlinie nicht anzuwenden und die Aktionen werden von dieser Richtlinie nicht zugelassen. Wenn keine anderen Richtlinienanweisungen die Aktionen zulassen, kann der Benutzer nur mit virtuellen Clustern arbeiten, die dieses Tag mit diesem Wert enthalten.

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

```
    }
  ]
}
```

Sie können auch mehrere Tag-Werte mithilfe eines Bedingungsoperators angeben. Um beispielsweise Aktionen in virtuellen Clustern zuzulassen, in denen das Tag `department` den Wert `dev` oder `test` enthält können Sie den Bedingungsblock im vorherigen Beispiel durch Folgendes ersetzen.

```
"Condition": {
        "StringEquals": {
          "aws:ResourceTag/department": ["dev", "test"]
        }
      }
```

Kennzeichnung bei der Erstellung einer Ressource vorschreiben

Im folgenden Beispiel muss das Tag bei der Erstellung des virtuellen Clusters angewendet werden.

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

Mit der folgenden Richtlinienanweisung kann ein Benutzer nur dann einen virtuellen Cluster erstellen, wenn der Cluster über ein `department`-Tag verfügt, das einen beliebigen Wert enthalten kann.

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-containers:CreateVirtualCluster"
      ],
      "Resource": "*",
      "Condition": {
        "Null": {
          "aws:RequestTag/department": "false"
        }
      }
    }
  ]
}
```

Zugriff zum Hinzufügen und Entfernen von Tags verweigern

Diese Richtlinie sieht vor, einem Benutzer die Berechtigung zum Hinzufügen oder Entfernen von Tags in virtuellen Clustern, die mit einem `department`-Tag mit dem Wert `dev` markiert sind, zu verweigern.

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

Fehlerbehebung für Identität und Zugriff in Amazon EMR in EKS

Diagnostizieren und beheben Sie mithilfe der folgenden Informationen gängige Probleme, die bei der Verwendung von Amazon EMR in EKS und IAM auftreten können.

Themen

Ich bin nicht autorisiert, eine Aktion in Amazon EMR in EKS auszuführen

Wenn die AWS Management Console Ihnen mitteilt, dass Sie nicht zur Ausführung einer Aktion autorisiert sind, 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-containers:`*`GetWidget`*-Berechtigungen verfügt.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform: emr-
containers: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-containers:`*`GetWidget`* zugreifen zu können.

Ich bin nicht zur Ausführung von iam:PassRole autorisiert

Wenn Sie die Fehlermeldung erhalten, dass Sie nicht zur Ausführung der Aktion `iam:PassRole` autorisiert sind, müssen Ihre Richtlinien aktualisiert werden, um eine Rolle an Amazon EMR in EKS übergeben zu können.

Einige AWS-Services erlauben die Übergabe einer vorhandenen Rolle an diesen Dienst, sodass keine neue Servicerolle oder serviceverknüpfte Rolle erstellt werden muss. 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 in EKS 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.

Wenden Sie sich an Ihren AWS-Administrator, falls Sie weitere Unterstützung benötigen. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich möchte Personen außerhalb meines AWS-Kontos Zugriff auf meine Ressourcen von Amazon EMR in EKS erteilen

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. Im Fall von Diensten, die ressourcenbasierte Richtlinien oder Zugriffskontrolllisten (Access Control Lists, ACLs) verwenden, können Sie diese Richtlinien verwenden, um Personen Zugriff auf Ihre Ressourcen zu gewähren.

Weitere Informationen dazu finden Sie hier:

- Informationen dazu, ob Amazon EMR in EKS diese Features unterstützt, finden Sie unter Funktionsweise von Amazon EMR in EKS mit IAM.
- Informationen zum Gewähren des Zugriffs auf Ihre Ressourcen für alle Ihre AWS-Konten finden Sie unter Gewähren des Zugriffs für einen IAM-Benutzer in einem anderen Ihrer AWS-Konto im IAM-Benutzerhandbuch.
- Informationen dazu, wie Sie AWS-Konten-Drittanbieter Zugriff auf Ihre Ressourcen bereitstellen, finden Sie unter Gewähren des Zugriffs auf AWS-Konten von externen Benutzern im IAM-Benutzerhandbuch.
- Informationen dazu, wie Sie über einen Identitätsverbund Zugriff gewähren, finden Sie unter Gewähren von Zugriff für extern authentifizierte Benutzer (Identitätsverbund) im IAM-Benutzerhandbuch.

- Informationen zum Unterschied zwischen der Verwendung von Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter So unterscheiden sich IAM-Rollen von ressourcenbasierten Richtlinien im IAM-Benutzerhandbuch.

Protokollierung und Überwachung

Um Vorfälle zu erkennen, Benachrichtigungen zu Vorfällen zu erhalten und um darauf zu reagieren, verwenden Sie diese Optionen mit Amazon EMR in EKS:

- Überwachen von Amazon EMR in EKS mit AWS CloudTrail – AWS CloudTrail bietet eine Aufzeichnung von Aktionen eines Benutzers, einer Rolle oder eines AWS-Services in Amazon EMR in EKS. Es erfasst Aufrufe über die Amazon-EMR-Konsole und Code-Aufrufe der API-Vorgänge von Amazon EMR in EKS als Ereignisse. Damit können Sie die an Amazon EMR in EKS ergangene Anfrage erkennen, die IP-Adresse, von der die Anfrage ausgegangen ist, wer die Anfrage getätigt hat, wann sie erfolgte, sowie weitere Einzelheiten. Weitere Informationen finden Sie unter Protokollieren von API-Aufrufen von Amazon EMR in EKS mit AWS CloudTrail.
- Verwenden Sie CloudWatch Events mit Amazon EMR in EKS – CloudWatch Events bietet einen nahezu in Echtzeit verfügbaren Stream von Systemereignissen, die Änderungen in AWS-Ressourcen beschreiben. CloudWatch Events erkennt betriebliche Änderungen, sobald sie auftreten, reagiert darauf und ergreift bei Bedarf Korrekturmaßnahmen, indem es Nachrichten sendet, um an die Umgebung zu reagieren, Funktionen zu aktivieren, Änderungen vorzunehmen und Zustandsinformationen zu erfassen. Um CloudWatch Events mit Amazon EMR in EKS zu verwenden, erstellen Sie eine Regel, die bei einem API-Aufruf von Amazon EMR in EKS über CloudTrail ausgelöst wird. Weitere Informationen finden Sie unter Überwachen von Aufträgen mit Amazon CloudWatch Events.

Protokollieren von API-Aufrufen von Amazon EMR in EKS mit AWS CloudTrail

Amazon EMR in EKS ist in AWS CloudTrail integriert, ein Service, der eine Aufzeichnung der von einem Benutzer, einer Rolle oder einem AWS-Service in Amazon EMR in EKS durchgeführten Aktionen bietet. CloudTrail erfasst alle API-Aufrufe für Amazon EMR in EKS als Ereignisse. Zu den erfassten Aufrufen gehören Aufrufe von der Konsole von Amazon EMR in EKS und Code-Aufrufe der API-Vorgänge von Amazon EMR in EKS. Wenn Sie einen Trail erstellen, können Sie die kontinuierliche Bereitstellung von CloudTrail-Ereignissen an einen Amazon-S3-Bucket, einschließlich Ereignissen für Amazon EMR in EKS, aktivieren. Wenn Sie keinen Trail konfigurieren, können Sie die

neuesten Ereignisse in der CloudTrail-Konsole trotzdem in Event history (Ereignisverlauf) anzeigen. Mit den von CloudTrail gesammelten Informationen können Sie die an Amazon EMR in EKS gestellte Anfrage, die IP-Adresse, von der die Anfrage gestellt wurde, den Initiator der Anfrage, den Zeitpunkt der Anfrage und zusätzliche Details bestimmen.

Weitere Informationen zu CloudTrail finden Sie im AWS CloudTrail-Benutzerhandbuch.

Informationen von Amazon EMR in EKS in CloudTrail

CloudTrail wird beim Erstellen Ihres AWS-Kontos für Sie aktiviert. Wenn in Amazon EMR in EKS eine Aktivität auftritt, wird sie in einem CloudTrail-Ereignis zusammen mit anderen AWS-Service-Ereignissen im Ereignisverlauf aufgezeichnet. Sie können die neusten Ereignisse in Ihr AWS-Konto herunterladen und dort suchen und anzeigen. Weitere Informationen finden Sie unter Anzeigen von Ereignissen mit dem CloudTrail-Ereignisverlauf.

Für eine fortlaufende Aufzeichnung der Ereignisse in Ihrem AWS-Konto, darunter Ereignisse für Amazon EMR in EKS, können Sie einen Trail erstellen. Ein Trail ermöglicht es CloudTrail, Protokolldateien in einem Amazon-S3-Bucket bereitzustellen. Wenn Sie einen Trail in der Konsole anlegen, gilt dieser für alle AWS-Regionen. Der Trail protokolliert Ereignisse aus allen Regionen in der AWS-Partition und stellt die Protokolldateien in dem von Ihnen angegebenen Amazon S3 Bucket bereit. Darüber hinaus können Sie andere AWS-Services konfigurieren, um die in den CloudTrail-Protokollen erfassten Ereignisdaten weiter zu analysieren und entsprechend zu agieren. Weitere Informationen finden Sie unter:

- Übersicht zum Erstellen eines Trails
- In CloudTrail unterstützte Services und Integrationen
- Konfigurieren von Amazon SNS-Benachrichtigungen für CloudTrail
- Empfangen von CloudTrail-Protokolldateien aus mehreren Regionen und Empfangen von CloudTrail-Protokolldateien von mehreren Konten.

Alle Aktionen von Amazon EMR in EKS werden von CloudTrail protokolliert und sind in der API-Dokumentation von Amazon EMR in EKS dokumentiert. Zum Beispiel generieren Aufrufe der Aktionen `CreateVirtualCluster`, `StartJobRun` und `ListJobRuns` Einträge in den CloudTrail-Protokolldateien.

Jeder Ereignis- oder Protokolleintrag enthält Informationen zu dem Benutzer, der die Anforderung generiert hat. Anhand der Identitätsinformationen zur Benutzeridentität können Sie Folgendes bestimmen:

- Ob die Anfrage mit Stammbenutzer- oder AWS Identity and Access Management (IAM)-Anmeldeinformationen ausgeführt wurde.
- Ob die Anforderung mit temporären Sicherheitsanmeldeinformationen für eine Rolle oder einen Verbundbenutzer ausgeführt wurde.
- Gibt an, ob die Anforderung aus einem anderen AWS-Service gesendet wurde

Weitere Informationen finden Sie unter CloudTrail-Element Benutzeridentität.

Erläuterungen der Protokolldateieinträge von Amazon EMR in EKS

Ein Trail ist eine Konfiguration, durch die Ereignisse als Protokolldateien an den von Ihnen angegebenen Amazon-S3-Bucket übermittelt werden. CloudTrail-Protokolldateien können einen oder mehrere Einträge enthalten. Ein Ereignis stellt eine einzelne Anfrage aus einer beliebigen Quelle dar und enthält unter anderem Informationen über die angeforderte Aktion, das Datum und die Uhrzeit der Aktion sowie über die Anfrageparameter. CloudTrail-Protokolleinträge sind kein geordnetes Stack-Trace der öffentlichen API-Aufrufe und erscheinen daher in keiner bestimmten Reihenfolge.

Das folgende Beispiel zeigt einen CloudTrail-Protokolleintrag, der die Aktion `ListJobRuns` demonstriert.

```
{
  "eventVersion": "1.05",
  "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": {
        "mfaAuthenticated": "false",
        "creationDate": "2020-11-04T21:49:36Z"
      }
```

```
    }
  },
  "eventTime": "2020-11-04T21:52:58Z",
  "eventSource": "emr-containers.amazonaws.com",
  "eventName": "ListJobRuns",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "203.0.113.1",
  "userAgent": "aws-cli/1.11.167 Python/2.7.10 Darwin/16.7.0 botocore/1.7.25",
  "requestParameters": {
    "virtualClusterId": "1K48XXXXXXHCB"
  },
  "responseElements": null,
  "requestID": "890b8639-e51f-11e7-b038-EXAMPLE",
  "eventID": "874f89fa-70fc-4798-bc00-EXAMPLE",
  "readOnly": true,
  "eventType": "AwsApiCall",
  "recipientAccountId": "012345678910"
}
```

Verwenden von Amazon S3 Access Grants mit Amazon EMR in EKS

Übersicht über S3 Access Grants für Amazon EMR in EKS

Mit den Amazon-EMR-Versionen 6.15.0 und höher bieten Amazon S3 Access Grants eine skalierbare Zugriffskontrolllösung, mit der Sie den Zugriff auf Ihre Amazon-S3-Daten von Amazon EMR in EKS aus erweitern können. 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-Daten über die Berechtigungen hinaus zu erweitern, die durch die Laufzeitrolle oder die IAM-Rollen gewährt werden, die den Identitäten mit Zugriff auf Ihren Cluster von Amazon EMR in EKS zugewiesen sind.

Weitere Informationen finden Sie unter Verwaltung des Zugriffs mit S3 Access Grants für Amazon EMR im Leitfaden zu Amazon EMR und Verwaltung des Zugriffs mit S3 Access Grants im Benutzerhandbuch zu Amazon Simple Storage Service.

Auf dieser Seite werden die Anforderungen für die Ausführung eines Spark-Auftrags in Amazon EMR in EKS mit S3-Access-Grants-Integration beschrieben. Mit Amazon EMR in EKS benötigt S3 Access

Grants eine zusätzliche IAM-Richtlinienanweisung in der Ausführungsrolle für Ihren Auftrag und eine zusätzliche Override-Konfiguration für die `StartJobRun`-API. Schritte zur Einrichtung von S3 Access Grants mit anderen Amazon-EMR-Bereitstellungen finden Sie in der folgenden Dokumentation:

- Verwenden von S3 Access Grants mit Amazon EMR
- Verwenden von S3 Access Grants mit EMR Serverless

Einen Cluster von Amazon EMR in EKS mit S3 Access Grants für die Datenverwaltung starten

Sie können S3 Access Grants in Amazon EMR in EKS aktivieren und einen Spark-Auftrag 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 Auftragsausführungsrolle für Ihren Cluster von Amazon EMR in EKS ein. Geben Sie die erforderlichen IAM-Berechtigungen an, die Sie für die Ausführung von Spark-Aufträgen 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 die IAM-Rollen, die Sie für die Auftragsausführung angeben, zusätzlichen Berechtigungen für den direkten Zugriff auf S3 enthalten, können Benutzer möglicherweise unabhängig von den Berechtigungen, die Sie in S3 Access Grants definieren, auf Daten zugreifen.

2. Senden Sie einen Auftrag an Ihren Cluster von Amazon EMR in EKS mit einer Amazon-EMR-Versionsbezeichnung von 6.15 oder höher und der `emrfs-site`-Klassifizierung, wie im folgenden Beispiel gezeigt. Ersetzen Sie die Werte in *`red text`* mit den passenden Werten für Ihr Nutzungsszenario.

```
{
  "name": "myjob",
  "virtualClusterId": "123456",
  "executionRoleArn": "iam_role_name_for_job_execution",
  "releaseLabel": "emr-7.1.0-latest",
  "jobDriver": {
    "sparkSubmitJobDriver": {
      "entryPoint": "entryPoint_location",
      "entryPointArguments": ["argument1", "argument2"],
       "sparkSubmitParameters": "--class main_class"
    }
  },
  "configurationOverrides": {
    "applicationConfiguration": [
      {
        "classification": "emrfs-site",
        "properties": {
          "fs.s3.s3AccessGrants.enabled": "true",
          "fs.s3.s3AccessGrants.fallbackToIAM": "false"
         }
      }
    ],
  }
}
```

Überlegungen zu S3 Access Grants mit Amazon EMR in EKS

Wichtige Informationen zu Support, Kompatibilität und Verhalten bei der Verwendung von Amazon S3 Access Grants mit Amazon EMR in EKS finden Sie unter Überlegungen zu S3 Access Grants mit Amazon EMR im Leitfaden zu Amazon EMR.

Compliance-Validierung für Amazon EMR in EKS

Externe Prüfer bewerten im Rahmen verschiedener AWS-Compliance-Programme die Sicherheit und Compliance von Amazon EMR in EKS. Hierzu zählen unter anderem SOC, PCI, FedRAMP und HIPAA.

Ausfallsicherheit bei Amazon EMR in EKS

Im Zentrum der globalen AWS Infrastruktur stehen die AWS-Regionen und Availability Zones (Verfügbarkeitszonen, AZs). 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 über AWS Regionen und Availability Zones finden Sie unter AWS Globale Infrastruktur.

Zusätzlich zur globalen AWS-Infrastruktur bietet Amazon EMR in EKS die Integration mit Amazon S3 über EMRFS, um Ihre Datenausfallsicherheit und Backup-Anforderungen zu unterstützen.

Infrastruktursicherheit in Amazon EMR in EKS

Als verwalteter Service ist Amazon EMR durch die AWS globale Netzwerksicherheit von geschützt. Informationen zu AWS-Sicherheitsservices und wie AWS die Infrastruktur schützt, finden Sie unter AWS-Cloud-Sicherheit. Informationen zum Entwerfen Ihrer AWS-Umgebung anhand der bewährten Methoden für die Infrastruktursicherheit finden Sie unter Infrastrukturschutz im Security Pillar AWS Well-Architected Framework.

Sie greifen mithilfe von AWS veröffentlichten API-Aufrufe über das Netzwerk auf Amazon EMR zu. Kunden müssen Folgendes unterstützen:

- Transport Layer Security (TLS). Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Verschlüsselungs-Suiten mit Perfect Forward Secrecy (PFS) wie DHE (Ephemeral Diffie-Hellman) oder ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Die meisten modernen Systeme wie Java 7 und höher unterstützen diese Modi.

Außerdem müssen Anforderungen mit einer Zugriffsschlüssel-ID und einem geheimen Zugriffsschlüssel signiert sein, der einem IAM-Prinzipal zugeordnet ist. Alternativ können Sie mit AWS Security Token Service (AWS STS) temporäre Sicherheitsanmeldeinformationen erstellen, um die Anforderungen zu signieren.

Konfigurations- und Schwachstellenanalyse

AWS übernimmt grundlegende Sicherheitsaufgaben wie Betriebssystemsicherheit und Datenbank-Patching, Firewall-Konfiguration und Notfallwiederherstellung. Diese Verfahren wurden von qualifizierten Dritten überprüft und zertifiziert. Weitere Informationen finden Sie in den folgenden Ressourcen:

- Compliance-Validierung für Amazon EMR in EKS
- Modell der übergreifenden Verantwortlichkeit
- Amazon Web Services: Übersicht über Sicherheitsverfahren (Whitepaper)

Herstellen einer Verbindung mit Amazon EMR in EKS über einen Schnittstellen-VPC-Endpunkt

Sie können über Schnittstellen-VPC-Endpunkte (AWS PrivateLink) in Ihrer Virtual Private Cloud (VPC) eine direkte Verbindung zu Amazon EMR in EKS herstellen, anstatt eine Verbindung über das Internet herzustellen. Wenn Sie einen Schnittstellen-VPC-Endpunkt verwenden, findet die Kommunikation zwischen Ihrer VPC und Amazon EMR in EKS vollständig innerhalb des AWS-Netzwerks statt. Jeder VPC-Endpunkt wird durch eine oder mehrere Elastic-Network-Schnittstellen (ENIs) mit privaten IP-Adressen in Ihren VPC-Subnetzen repräsentiert.

Der Schnittstellen-VPC-Endpunkt verbindet Ihre VPC direkt mit Amazon EMR in EKS ohne Internet-Gateway, NAT-Gerät, VPN-Verbindung oder AWS-Direct-Connect-Verbindung. Die Instances in Ihrer VPC benötigen für die Kommunikation mit der API von Amazon EMR in EKS keine öffentlichen IP-Adressen.

Sie können über die AWS Management Console oder AWS Command Line Interface (AWS CLI)-Befehle einen Schnittstellen-VPC-Endpunkt erstellen, um eine Verbindung zu Amazon EMR in EKS herzustellen. Weitere Informationen finden Sie unter Erstellen eines Schnittstellenendpunkts.

Wenn Sie nach dem Erstellen eines Schnittstellen-VPC-Endpunkts private DNS-Host-Namen für den Endpunkt aktivieren, wird der Standardendpunkt von Amazon EMR in EKS in den VPC-Endpunkt

aufgelöst. Der Service-Standardname für den Endpunkt für Amazon EMR in EKS hat das folgende Format.

```
emr-containers.Region.amazonaws.com
```

Wenn Sie keine privaten DNS-Hostnamen aktiviert haben, stellt Amazon VPC einen DNS-Endpunktnamen bereit, den Sie im folgenden Format verwenden können.

```
VPC_Endpoint_ID.emr-containers.Region.vpce.amazonaws.com
```

Weitere Informationen finden Sie unter Schnittstellen-VPC-Endpunkte (AWS-PrivateLink) im Amazon-VPC-Benutzerhandbuch. Amazon EMR in EKS unterstützt Aufrufe aller API-Aktionen innerhalb Ihrer VPC.

Sie können VPC-Endpunktrichtlinien an einen VPC-Endpunkt anfügen, um den Zugriff für IAM-Prinzipale zu steuern. Sie können einem VPC-Endpunkt auch Sicherheitsgruppen zuordnen, um den eingehenden und ausgehenden Zugriff basierend auf Ursprung und Ziel des Netzwerkdatenverkehrs zu steuern, z. B. mit einem IP-Adressbereich. Weitere Informationen finden Sie unter Steuern des Zugriffs auf Services mit VPC-Endpunkten.

Eine VPC-Endpunktrichtlinie für Amazon EMR in EKS erstellen

Sie können eine Richtlinie für Amazon-VPC-Endpunkte für Amazon EMR in EKS erstellen, in der Sie Folgendes angeben:

- Prinzipal, der Aktionen ausführen/nicht ausführen kann
- Aktionen, die ausgeführt werden können
- Ressourcen, für die Aktionen ausgeführt werden können

Weitere Informationen finden Sie unter Steuerung des Zugriffs auf Services mit VPC-Endpunkten im Amazon-VPC-Benutzerhandbuch.

Example VPC-Endpunktrichtlinie zum Verweigern des Zugriffs mit einem angegebenen AWS-Konto

Die folgende VPC-Endpunktrichtlinie verweigert dem AWS-Konto *123456789012* jeglichen Zugriff auf Ressourcen, die den Endpunkt verwenden.

```
{
```

```
    "Statement": [
        {
            "Action": "*",
            "Effect": "Allow",
            "Resource": "*",
            "Principal": "*"
        },
        {
            "Action": "*",
            "Effect": "Deny",
            "Resource": "*",
            "Principal": {
                "AWS": [
                    "123456789012"
                ]
            }
        }
    ]
}
```

Example VPC-Endpunktrichtlinie zum Gewähren des VPC-Zugriffs auf einen angegebenen IAM-Prinzipal (Benutzer)

Die folgende VPC-Endpunktrichtlinie gewährt nur dem IAM-Benutzer *lijuan* im AWS-Konto *123456789012* vollen Zugriff. Allen anderen IAM-Prinzipalen wird der Zugriff über den Endpunkt verweigert.

```
{
    "Statement": [
        {
            "Action": "*",
            "Effect": "Allow",
            "Resource": "*",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::123456789012:user/lijuan"
                ]
            }
        }
    ]
}
```

Example VPC-Endpunktrichtlinie zum Erlauben von Leseoperationen in Amazon EMR in EKS

Die folgende VPC-Endpunktrichtlinie erlaubt nur dem AWS-Konto *123456789012*, die angegebenen Aktionen für Amazon EMR in EKS auszuführen.

Die angegebenen Aktionen stellen das Äquivalent von schreibgeschütztem Zugriff für Amazon EMR in EKS dar. Alle anderen Aktionen in der VPC werden dem angegebenen Konto verweigert. Allen anderen Konten wird der Zugriff verweigert. Eine Liste der Aktionen in Amazon EMR in EKS finden Sie unter Aktionen, Ressourcen und Bedingungsschlüssel für Amazon EMR in EKS.

```
{
    "Statement": [
        {
            "Action": [
                "emr-containers:DescribeJobRun",
                "emr-containers:DescribeVirtualCluster",
                "emr-containers:ListJobRuns",
                "emr-containers:ListTagsForResource",
                "emr-containers:ListVirtualClusters"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Principal": {
                "AWS": [
                    "123456789012"
                ]
            }
        }
    ]
}
```

Example VPC-Endpunktrichtlinie, die den Zugriff auf einen angegebenen virtuellen Cluster verweigert

Die folgende VPC-Endpunktrichtlinie gewährt vollen Zugriff für alle Konten und Prinzipale, verweigert jedoch jeglichen Zugriff des AWS-Kontos *123456789012* auf Aktionen, die im virtuellen Cluster mit der Cluster-ID *A1B2CD34EF5G* ausgeführt werden. Andere Aktionen in Amazon EMR in EKS, die keine Berechtigungen auf Ressourcenebene für virtuelle Cluster unterstützen, sind weiterhin zulässig. Eine Liste der Aktionen von Amazon EMR in EKS und ihrer entsprechenden Ressourcentypen finden Sie unter Aktionen, Ressourcen und Bedingungsschlüssel für Amazon EMR in EKS im AWS Identity and Access Management-IAM-Benutzerhandbuch.

```
{
```

```
    "Statement": [
        {
            "Action": "*",
            "Effect": "Allow",
            "Resource": "*",
            "Principal": "*"
        },
        {
            "Action": "*",
            "Effect": "Deny",
            "Resource": "arn:aws:emr-containers:us-west-2:123456789012:/
virtualclusters/A1B2CD34EF5G",
            "Principal": {
                "AWS": [
                    "123456789012"
                ]
            }
        }
    ]
}
```

Kontoübergreifenden Zugriff für Amazon EMR in EKS einrichten

Sie können kontoübergreifenden Zugriff für Amazon EMR in EKS einrichten. Der kontoübergreifende Zugriff ermöglicht es Benutzern eines AWS-Kontos, Amazon EMR in EKS-Aufträge auszuführen und auf die zugrunde liegenden Daten zuzugreifen, die zu einem anderen AWS-Konto gehören.

Voraussetzungen

Um den kontoübergreifenden Zugriff für Amazon EMR in EKS einzurichten, erledigen Sie Aufgaben, während Sie bei den folgenden AWS-Konten angemeldet sind:

- `AccountA` – Ein AWS-Konto, bei dem Sie ein Amazon EMR auf einem virtuellen EKS-Cluster erstellt haben, indem Sie Amazon EMR mit einem Namespace auf einem EKS-Cluster registriert haben.
- `AccountB` – Ein AWS-Konto, das einen Amazon-S3-Bucket oder eine DynamoDB-Tabelle enthält, auf die Ihre Amazon EMR in EKS-Aufträge zugreifen sollen.

Sie müssen Folgendes in Ihren AWS-Konten bereithalten, bevor Sie den kontoübergreifenden Zugriff einrichten können:

- Ein virtueller Amazon EMR in EKS-Cluster in `AccountA`, in dem Sie Aufträge ausführen möchten.
- Eine Rolle zur Aufgabenausführung in `AccountA`, die über die erforderlichen Berechtigungen zum Ausführen von Aufgaben im virtuellen Cluster verfügt. Weitere Informationen finden Sie unter Erstellen einer Aufgabenausführungsrolle und Auftragausführungsrollen mit Amazon EMR in EKS verwenden.

So greifen Sie auf einen kontoübergreifenden Amazon-S3-Bucket oder eine DynamoDB-Tabelle zu

Gehen Sie wie folgt vor, um den kontoübergreifenden Zugriff für Amazon EMR in EKS einzurichten.

1. Erstellen Sie einen Amazon-S3-Bucket,`cross-account-bucket`, in `AccountB`. Weitere Informationen finden Sie unter Bucket erstellen. 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.
2. Erstellen Sie eine `Cross-Account-Role-B` IAM-Rolle in `AccountB`, die auf das `cross-account-bucket` zugreifen kann.
    1. Melden Sie sich an der IAM-Konsole an.
    2. Wählen Sie Rollen und anschließend Neue Rolle `Cross-Account-Role-`Berstellen aus. Weitere Informationen zum Erstellen von IAM-Rollen finden Sie unter Erstellen von IAM-Rollen im IAM-Benutzerhandbuch.
    3. Erstellen Sie eine IAM-Richtlinie, die die Berechtigungen für den `Cross-Account-Role-B` Zugriff auf den `cross-account-bucket` S3-Bucket festlegt, wie die folgende Richtlinienerklärung zeigt. Fügen Sie die IAM-Richtlinie an `Cross-Account-Role-B` an. Weitere Informationen finden Sie unter Erstellen einer neuen Richtlinie im IAM-Benutzerhandbuch.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::cross-account-bucket",
                "arn:aws:s3:::cross-account-bucket/*"
```

```
            ]
        }
    ]
}
```

Wenn DynamoDB-Zugriff erforderlich ist, erstellen Sie eine IAM-Richtlinie, die Berechtigungen für den Zugriff auf die kontoübergreifende DynamoDB-Tabelle festlegt. Fügen Sie die IAM-Richtlinie an `Cross-Account-Role-B` an. Weitere Informationen finden Sie unter Erstellen einer Rolle im IAM-Benutzerhandbuch.

Im Folgenden finden Sie eine Richtlinie für den Zugriff auf eine 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.

    1. Um die Vertrauensstellung für die Rolle zu konfigurieren, wählen Sie in der IAM-Konsole die Registerkarte Vertrauensbeziehungen für die in Schritt 2 erstellte Rolle aus: `Cross-Account-Role-B`.
    2. Wählen Sie Vertrauensbeziehungen bearbeiten aus.
    3. Fügen Sie das folgende Richtliniendokument hinzu, das es Ihnen ermöglicht `Job-Execution-Role-A` in `AccountA`, diese `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. Erteilen Sie `Job-Execution-Role-A` in `AccountA` mmit – STS Assume die Berechtigung, die Rolle `Cross-Account-Role-B` zu übernehmen.

   1. Wählen Sie in der IAM-Konsole für das AWS-Konto `AccountA` die Option `Job-Execution-Role-A` aus.
   2. 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"
        }
    ]
}
```

5. Stellen Sie für den Zugriff auf Amazon S3 die folgenden `spark-submit`-Parameter (`spark conf`) ein, während Sie den Auftrag an Amazon EMR in EKS senden.

> **Note**
>
> Standardmäßig verwendet EMRFS die Rolle Aufgabenausführung, um vom Auftrag aus auf den S3-Bucket zuzugreifen. Wenn `customAWSCredentialsProvider` jedoch auf `AssumeRoleAWSCredentialsProvider` gesetzt ist, verwendet EMRFS die entsprechende Rolle, die Sie mit `ASSUME_ROLE_CREDENTIALS_ROLE_ARN` angeben, und nicht `Job-Execution-Role-A` für den Amazon-S3-Zugriff.

- `--conf spark.hadoop.fs.s3.customAWSCredentialsProvider=com.amazonaws.emr.AssumeRol`

- `--conf spark.kubernetes.driverEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN=arn:aws:iam::Ac Cross-Account-Role-B \`
- `--conf spark.executorEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN=arn:aws:iam::AccountB:ro Cross-Account-Role-B \`

> **Note**
>
> Sie müssen in der `ASSUME_ROLE_CREDENTIALS_ROLE_ARN`-Auftrag-Spark-Konfiguration sowohl den Executor als auch den Treiber `env` angeben.

Für den kontoübergreifenden Zugriff von DynamoDB müssen Sie `--conf spark.dynamodb.customAWSCredentialsProvider=com.amazonaws.emr.AssumeRoleAWSCr` festlegen.

6. Führen Sie den Auftrag von Amazon EMR in EKS mit kontoübergreifendem Zugriff aus, wie das folgende Beispiel zeigt.

```
aws emr-containers start-job-run \
--virtual-cluster-id 123456 \
--name myjob \
--execution-role-arn execution-role-arn \
--release-label emr-6.2.0-latest \
--job-driver '{"sparkSubmitJobDriver": {"entryPoint": "entryPoint_location",
 "entryPointArguments": ["arguments_list"], "sparkSubmitParameters": "--class
 <main_class> --conf spark.executor.instances=2 --conf spark.executor.memory=2G
 --conf spark.executor.cores=2 --conf spark.driver.cores=1 --conf
 spark.hadoop.fs.s3.customAWSCredentialsProvider=com.amazonaws.emr.AssumeRoleAWSCredentials
 --conf
 spark.kubernetes.driverEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN=arn:aws:iam::AccountB:role/
Cross-Account-Role-B --conf
 spark.executorEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN=arn:aws:iam::AccountB:role/
Cross-Account-Role-B"}} ' \
--configuration-overrides '{"applicationConfiguration": [{"classification":
 "spark-defaults", "properties": {"spark.driver.memory": "2G"}}],
 "monitoringConfiguration": {"cloudWatchMonitoringConfiguration":
 {"logGroupName": "log_group_name", "logStreamNamePrefix": "log_stream_prefix"},
```

```
 "persistentAppUI":"ENABLED",  "s3MonitoringConfiguration": {"logUri": "s3://
my_s3_log_location" }}}'
```

Kennzeichnen Ihrer Ressourcen von Amazon EMR in EKS

Um Sie bei der Verwaltung Ihrer Ressourcen von Amazon EMR in EKS zu unterstützen, können Sie jeder Ressource eigene Metadaten in Form von Tags zuweisen. Dieses Thema bietet einen Überblick über die Tags-Funktion und zeigt, wie Sie Tags erstellen können.

Themen

Grundlagen zu Tags (Markierungen)

Ein Tag (Markierung) ist eine Markierung, die Sie einer AWS-Ressource zuordnen. Jeder Tag (Markierung) besteht aus einem Schlüssel und einem optionalen Wert, beides können Sie bestimmen.

Mithilfe von Tags können Sie Ihre AWS-Ressourcen nach Attributen kategorisieren, z. B. nach Zweck, Besitzer oder 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 Cluster in Amazon EMR in EKS definieren, um Ihnen dabei zu helfen, den Eigentümer und die Stack-Ebene jedes einzelnen Clusters nachzuverfolgen. Sie sollten für jeden Ressourcentyp einen konsistenten Satz von Tag-Schlüsseln entwickeln. Anschließend können Sie die Ressourcen basierend auf den hinzugefügten Tags suchen und filtern.

Tags werden nicht automatisch Ihren Ressourcen zugewiesen. Nachdem Sie ein Tag hinzugefügt haben, können Sie jederzeit Tag-Schlüssel und -Werte bearbeiten oder Tags aus einer Ressource entfernen. Wenn Sie eine Ressource löschen, werden alle Tags (Markierungen) der Ressource ebenfalls gelöscht.

Tags (Markierungen) haben keine semantische Bedeutung für Amazon EMR in EKS und werden ausschließlich als Zeichenfolgen interpretiert.

Ein Tag-Wert kann eine leere Zeichenfolge, aber nicht null sein. Ein Tag-Schlüssel kann keine leere Zeichenfolge sein. 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 AWS Identity and Access Management (IAM) verwenden, können Sie steuern, welche Benutzer in Ihrem AWS-Konto Tags (Markierungen) erstellen, bearbeiten oder löschen dürfen.

Beispiele für Tag-basierte Zugriffssteuerungsrichtlinien finden Sie unter Richtlinien für Tag-basierte Zugriffskontrolle.

Markieren Ihrer -Ressourcen mit Tags (Markierungen)

Sie können neue oder vorhandene virtuelle Cluster und Auftragsausführungen markieren, die sich im aktiven Status befinden. Zu den aktiven Status für Auftragsausführungen gehören: `PENDING`, `SUBMITTED`, `RUNNING`, und `CANCEL_PENDING`. Zu den aktiven Zuständen für virtuelle Cluster gehören: `RUNNING`, `TERMINATING` und `ARRESTED`. Weitere Informationen finden Sie unter Status von Aufgabenausführungen und Status des virtuellen Clusters.

Wenn ein virtueller Cluster beendet wird, werden die Tags bereinigt und es kann nicht mehr darauf zugegriffen werden.

Wenn Sie die API von Amazon EMR in EKS, die AWS CLI oder ein AWS-SDK verwenden, können Sie Tags mithilfe des Parameters auf der entsprechenden API-Aktion auf neue Ressourcen anwenden. Sie können auch über die `TagResource`-API Tags auf Ressourcen anwenden.

Sie können einige Aktionen zum Erstellen von Ressourcen verwenden, um Tags für eine Ressource anzugeben, wenn die Ressource erstellt wird. Wenn in diesem Fall die Tags nicht angewendet werden können, während die Ressource erstellt wird, kann die Ressource nicht erstellt werden. Auf diese Weise wird sichergestellt, dass Ressourcen, die Sie bei der Erstellung markieren möchten, entweder mit angegebenen Tags oder gar nicht erstellt werden. Wenn Sie Ressourcen zum Zeitpunkt der Erstellung markieren, müssen Sie nach der Ressourcenerstellung keine benutzerdefinierten Tagging-Skripts ausführen.

Die folgende Tabelle beschreibt die Ressourcen von Amazon EMR in EKS, die markiert werden können.

Ressource	Unterstützt Tags (Markierungen)	Unterstützt Tag-Propagierung	Unterstützt das Markieren bei Erstellung (API von Amazon EMR in EKS, AWS CLI und AWS-SDK)	API für die Erstellung (Tags können während der Erstellung hinzugefügt werden)
Virtueller Cluster	Ja	Nein. Mit einem virtuellen Cluster verknüpfte Tags werden nicht auf Auftragsausführungen übertragen, die an diesen virtuellen Cluster gesendet werden.	Ja	CreateVirtualCluster
Auftragsausführungen	Ja	Nein	Ja	StartJobRun

Tag (Markierung)-Einschränkungen

Die folgenden grundlegenden Einschränkungen gelten für Tags (Markierungen):

- Maximale Anzahl von Tags (Markierungen) pro Ressource: 50
- Jeder Tag (Markierung) muss für jede Ressource eindeutig sein. Jeder Tag (Markierung) kann nur einen Wert haben.
- Maximale Schlüssellänge: 128 Unicode-Zeichen in UTF-8
- Maximale Wertlänge: 256 Unicode-Zeichen in UTF-8
- Wenn Ihr Markierungsschema für mehrere AWS-Services und -Ressourcen verwendet wird, denken Sie daran, dass andere Services möglicherweise Einschränkungen für zulässige Zeichen

haben. Allgemein erlaubte Zeichen sind Buchstaben, Zahlen, Leerzeichen, die in UTF-8 darstellbar sind, sowie die folgenden Zeichen: + - = . _ : / @.

- Bei Tag-Schlüsseln und -Werten muss die Groß- und Kleinschreibung beachtet werden.
- Ein Tag-Wert kann eine leere Zeichenfolge, aber nicht null sein. Ein Tag-Schlüssel kann keine leere Zeichenfolge sein.
- Verwenden Sie weder `aws:` noch `AWS:` oder Kombinationen aus Groß- und Kleinbuchstaben von diesen als Präfix für Schlüssel oder Werte, da sie für die -Verwendung reserviert sind. Sie sind für die AWS Verwendung reserviert.

Arbeiten Sie mit Tags mit der AWS CLI und der API von Amazon EMR in EKS

Verwenden Sie die folgenden AWS CLI-Befehle oder API-Vorgänge von Amazon EMR in EKS, um die Tags für Ihre Ressourcen hinzuzufügen, zu aktualisieren, aufzulisten und zu löschen.

Aufgabe	AWS CLI	API-Aktion
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 man Tags an Ressourcen mithilfe der AWS CLI hinzufügt oder entfernt.

Beispiel 1: Markieren eines virtuellen Clusters

Der folgende Befehl markiert einen vorhandenen virtuellen Cluster.

```
aws emr-containers tag-resource --resource-arn resource_ARN --tags team=devs
```

Beispiel 2: Entfernen von Tags von einem bestehenden virtuellen Cluster

Der folgende Befehl löscht ein Tag von einem bestehenden virtuellen Cluster.

```
aws emr-containers untag-resource --resource-arn resource_ARN --tag-keys tag_key
```

Beispiel 3: Tags für eine Ressource auflisten

Der folgende Befehl listet die Tags auf, die einer vorhandenen Ressource zugeordnet sind.

```
aws emr-containers list-tags-for-resource --resource-arn resource_ARN
```

Beheben von Problemen in Amazon EMR in EKS

In diesem Abschnitt wird beschrieben, wie Sie Probleme mit Amazon EMR in EKS beheben. Informationen zur Behebung allgemeiner Probleme mit Amazon EMR finden Sie unter Problembehandlung bei Clustern im Verwaltungshandbuch für Amazon EMR.

Themen

Fehlerbehebung bei Aufträgen, die PersistentVolumeClaims (PVC) verwenden

Wenn Sie PersistentVolumeClaims (PVC) für einen Auftrag erstellen, auflisten oder löschen müssen, aber keine PVC-Berechtigungen zu den standardmäßigen Kubernetes-Rollen emr-containers hinzufügen, schlägt der Auftrag fehl, wenn Sie ihn einreichen. Ohne diese Berechtigungen kann die Rolle emr-containers keine erforderlichen Rollen für den Spark-Treiber oder den Spark-Client erstellen. Es reicht nicht aus, den Spark-Treiber oder Client-Rollen-Berechtigungen hinzuzufügen, wie es in den Fehlermeldungen angedeutet wird. Die primäre Rolle emr-container muss auch die erforderlichen Berechtigungen enthalten. In diesem Abschnitt wird erklärt, wie Sie der primären Rolle emr-containers die erforderlichen Berechtigungen hinzufügen.

Verifizierung

Um zu überprüfen, ob Ihre Rolle emr-containers über die erforderlichen Berechtigungen verfügt, setzen Sie die Variable NAMESPACE mit Ihrem eigenen Wert und führen Sie dann den folgenden Befehl aus:

```
export NAMESPACE=YOUR_VALUE
kubectl describe role emr-containers -n ${NAMESPACE}
```

Führen Sie außerdem den folgenden Befehl aus, um zu überprüfen, ob die Rollen Spark und Client über die erforderlichen Berechtigungen verfügen:

```
kubectl describe role emr-containers-role-spark-driver -n ${NAMESPACE}
```

```
kubectl describe role emr-containers-role-spark-client -n ${NAMESPACE}
```

Wenn die Berechtigungen nicht vorhanden sind, fahren Sie mit dem Patch wie folgt fort.

Patch

1. Wenn die Aufträge ohne die Berechtigungen gerade ausgeführt werden, beenden Sie diese Aufträge.
2. Erstellen Sie eine Datei mit dem Namen RBAC_Patch.py wie folgt:

```
import os
import subprocess as sp
import tempfile as temp
import json
import argparse
import uuid

def delete_if_exists(dictionary: dict, key: str):
    if dictionary.get(key, None) is not None:
        del dictionary[key]

def doTerminalCmd(cmd):
    with temp.TemporaryFile() as f:
        process = sp.Popen(cmd, stdout=f, stderr=f)
        process.wait()
        f.seek(0)
        msg = f.read().decode()
    return msg

def patchRole(roleName, namespace, extraRules, skipConfirmation=False):
    cmd = f"kubectl get role {roleName} -n {namespace} --output json".split(" ")
    msg = doTerminalCmd(cmd)
    if "(NotFound)" in msg and "Error" in msg:
        print(msg)
        return False
    role = json.loads(msg)
    rules = role["rules"]
    rulesToAssign = extraRules[::]
    passedRules = []
    for rule in rules:
        apiGroups = set(rule["apiGroups"])
        resources = set(rule["resources"])
```

```
        verbs = set(rule["verbs"])
        for extraRule in extraRules:
            passes = 0
            apiGroupsExtra = set(extraRule["apiGroups"])
            resourcesExtra = set(extraRule["resources"])
            verbsExtra = set(extraRule["verbs"])
            passes += len(apiGroupsExtra.intersection(apiGroups)) >=
len(apiGroupsExtra)
            passes += len(resourcesExtra.intersection(resources)) >=
len(resourcesExtra)
            passes += len(verbsExtra.intersection(verbs)) >= len(verbsExtra)
            if passes >= 3:
                if extraRule not in passedRules:
                    passedRules.append(extraRule)
                    if extraRule in rulesToAssign:
                        rulesToAssign.remove(extraRule)
                break
    prompt_text = "Apply Changes?"
    if len(rulesToAssign) == 0:
        print(f"The role {roleName} seems to already have the necessary
permissions!")
        prompt_text = "Proceed anyways?"
    for ruleToAssign in rulesToAssign:
        role["rules"].append(ruleToAssign)
    delete_if_exists(role, "creationTimestamp")
    delete_if_exists(role, "resourceVersion")
    delete_if_exists(role, "uid")
    new_role = json.dumps(role, indent=3)
    uid = uuid.uuid4()
    filename = f"Role-{roleName}-New_Permissions-{uid}-TemporaryFile.json"
    try:
        with open(filename, "w+") as f:
            f.write(new_role)
            f.flush()
        prompt = "y"
        if not skipConfirmation:
            prompt = input(
                doTerminalCmd(f"kubectl diff -f {filename}".split(" ")) +
f"\n{prompt_text} y/n: "
            ).lower().strip()
            while prompt != "y" and prompt != "n":
                prompt = input("Please make a valid selection. y/n:
").lower().strip()
        if prompt == "y":
```

```
            print(doTerminalCmd(f"kubectl apply -f {filename}".split(" ")))
    except Exception as e:
        print(e)
    os.remove(f"./{filename}")

if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument("-n", "--namespace",
                        help="Namespace of the Role. By default its the
 VirtualCluster's namespace",
                        required=True,
                        dest="namespace"
                        )

    parser.add_argument("-p", "--no-prompt",
                        help="Applies the patches without asking first",
                        dest="no_prompt",
                        default=False,
                        action="store_true"
                        )
    args = parser.parse_args()

    emrRoleRules = [
        {
            "apiGroups": [""],
            "resources": ["persistentvolumeclaims"],
            "verbs": ["list", "create", "delete"]
         }

    ]

    driverRoleRules = [
        {
            "apiGroups": [""],
            "resources": ["persistentvolumeclaims"],
            "verbs": ["list", "create", "delete"]
        },
        {
            "apiGroups": [""],
            "resources": ["services"],
            "verbs": ["get", "list", "describe", "create", "delete", "watch"]
        }
    ]
```

```
    clientRoleRules = [
        {
            "apiGroups": [""],
            "resources": ["persistentvolumeclaims"],
            "verbs": ["list", "create", "delete"]
        }
    ]

    patchRole("emr-containers", args.namespace, emrRoleRules, args.no_prompt)
    patchRole("emr-containers-role-spark-driver", args.namespace, driverRoleRules,
 args.no_prompt)
    patchRole("emr-containers-role-spark-client", args.namespace, clientRoleRules,
 args.no_prompt)
```

3. Führen Sie das Python-Skript aus:

```
python3 RBAC_Patch.py -n ${NAMESPACE}
```

4. Ein kubectl-Unterschied zwischen den neuen und den alten Berechtigungen wird angezeigt. Drücken Sie Y, um die Rolle zu patchen.
5. Überprüfen Sie die drei Rollen mit zusätzlichen Berechtigungen wie folgt:

```
kubectl describe role -n ${NAMESPACE}
```

6. Führen Sie das Python-Skript aus:

```
python3 RBAC_Patch.py -n ${NAMESPACE}
```

7. Nach dem Ausführen des Befehls wird ein kubectl-Unterschied zwischen den neuen und den alten Berechtigungen angezeigt. Drücken Sie Y, um die Rolle zu patchen.
8. Überprüfen Sie die drei Rollen mit zusätzlichen Berechtigungen:

```
kubectl describe role -n ${NAMESPACE}
```

9. Reichen Sie den Auftrag erneut ein.

Manuelles Patchen

Wenn die für Ihre Anwendung erforderliche Berechtigung für etwas anderes als die PVC-Regeln gilt, können Sie bei Bedarf manuell Kubernetes-Berechtigungen für Ihren virtuellen Amazon-EMR-Cluster hinzufügen.

> **Note**
>
> Die Rolle emr-containers ist eine primäre Rolle. Das bedeutet, dass sie alle erforderlichen Berechtigungen bereitstellen muss, bevor Sie Ihre zugrunde liegenden Treiber- oder Clientrollen ändern können.

1. Laden Sie die aktuellen Berechtigungen in Yaml-Dateien herunter, indem Sie die folgenden Befehle ausführen:

```
kubectl get role -n ${NAMESPACE} emr-containers -o yaml >> emr-containers-role-patch.yaml
kubectl get role -n ${NAMESPACE} emr-containers-role-spark-driver -o yaml >> driver-role-patch.yaml
kubectl get role -n ${NAMESPACE} emr-containers-role-spark-client -o yaml >> client-role-patch.yaml
```

2. Bearbeiten Sie jede Datei auf der Grundlage der für Ihre Anwendung erforderlichen Berechtigungen und fügen Sie zusätzliche Regeln hinzu, z. B. die folgenden:

   - emr-containers-role-patch.yaml

```
- apiGroups:
  - ""
  resources:
  - persistentvolumeclaims
  verbs:
  - list
  - create
  - delete
```

   - driver-role-patch.yaml

```
- apiGroups:
  - ""
```

```
  resources:
  - persistentvolumeclaims
  verbs:
  - list
  - create
  - delete
- apiGroups:
  - ""
  resources:
  - services
  verbs:
  - get
  - list
  - describe
  - create
  - delete
  - watch
```

- client-role-patch.yaml

```
- apiGroups:
  - ""
  resources:
  - persistentvolumeclaims
  verbs:
  - list
  - create
  - delete
```

3. Entfernen Sie die folgenden Attribute mit ihren Werten. Dies ist erforderlich, um das Update anzuwenden.
   - creationTimestamp
   - resourceVersion
   - uid
4. Führen Sie abschließend den Patch aus:

```
kubectl apply -f emr-containers-role-patch.yaml
kubectl apply -f driver-role-patch.yaml
kubectl apply -f client-role-patch.yaml
```

Fehlerbehebung von Amazon EMR im vertikalen Auto Scaling von EKS

Lesen Sie die folgenden Abschnitte, falls Sie bei der Einrichtung des vertikalen Auto-Scaling-Operators von Amazon EMR in EKS auf einem Amazon-EKS-Cluster mit Operator Lifecycle Manager auf Probleme stoßen. Weitere Informationen, einschließlich der Schritte zum Abschließen der Installation, finden Sie unter Verwenden von vertikalem Auto Scaling mit Amazon-EMR-Spark-Aufträgen.

Fehler 403 Forbidden

Wenn Sie die Schritte unter Den Operator Lifecycle Manager (OLM) auf Ihrem Amazon-EKS-Cluster installieren befolgt und den `olm status` Befehl ausgeführt haben und ein `403 Forbidden`-Fehler wie der folgende zurückgegeben wurde, haben Sie möglicherweise keine Authentifizierungstoken für das Amazon-ECR-Repository für den Operator abgerufen.

Um dieses Problem zu beheben, wiederholen Sie den Schritt unter Den vertikalen Auto-Scaling-Operator für Amazon EMR in EKS installieren, um die Token zu erhalten. Versuchen Sie anschließend erneut die Installation.

```
Error: FATA[0002] Failed to run bundle: pull bundle image: error pulling image IMAGE.
error resolving name : unexpected status code [manifests latest]: 403 Forbidden
```

Kubernetes-Namespace nicht gefunden

Wenn Sie den vertikalen Auto-Scaling-Operator von Amazon EMR in EKS auf einem Amazon-EKS-Cluster einrichten, erhalten Sie möglicherweise eine `namespaces not found`-Fehlermeldung wie die hier gezeigte:

```
FATA[0020] Failed to run bundle: create catalog: error creating catalog source:
 namespaces "NAME" not found.
```

Wenn der von Ihnen angegebene Namespace nicht existiert, installiert OLM den vertikalen Auto-Scaling-Operator nicht. Sie lösen dieses Problem, indem Sie den folgenden Befehl verwenden, um den Namespace zu erstellen. Versuchen Sie anschließend erneut die Installation.

```
kubectl create namespace NAME
```

Fehler beim Speichern der Docker-Anmeldeinformationen

Um vertikales Auto Scaling einzurichten, müssen Sie Ihr Amazon EMR auf den Docker-Images für vertikales Auto Scaling von EKS authentifizieren und abrufen. Wenn Sie dies tun, erhalten Sie möglicherweise eine Fehlermeldung wie die folgende, wenn Docker nicht läuft:

```
aws ecr get-login-password \
 --region $REGION | docker login \
 --username AWS \
 --password-stdin $ACCOUNT_ID.dkr.ecr.$REGION.amazonaws.com

Error saving credentials: error storing credentials - err: exit status 1
out: 'Post "http://ipc/registry/credstore-updated": dial unix backend.sock: connect: no
 such file or directory'
```

Um dieses Problem zu beheben, stellen Sie sicher, dass Docker läuft, oder öffnen Sie Docker Desktop. Versuchen Sie dann erneut, Ihre Anmeldeinformationen zu speichern.

Fehlerbehebung beim Spark-Operator in Amazon EMR in EKS

Lesen Sie die folgenden Abschnitte, falls Sie Probleme mit dem Spark-Operator von Amazon EMR in EKS haben. Weitere Informationen, einschließlich der Schritte zum Abschließen der Installation, finden Sie unter Spark-Aufträge mit dem Spark-Operator ausführen.

Fehler bei der Installation des Helm-Charts

Wenn Sie die Schritte unter Den Spark-Operator installieren befolgt haben und beim Versuch, das Helm-Chart zu installieren oder zu überprüfen, ein `INSTALLATION FAILED`-Fehler wie der folgende zurückgegeben wurde, haben Sie die Authentifizierungstoken möglicherweise nicht für das Amazon-ECR-Repository für den Operator abgerufen.

Sie lösen dieses Problem, indem Sie den Schritt unter Den Spark-Operator installieren wiederholen, um Ihren Helm-Client bei dem Amazon-ECR-Registrierung zu authentifizieren. Versuchen Sie dann den Installationsschritt erneut.

```
Error: INSTALLATION FAILED: Kubernetes cluster unreachable: the server has asked for
 the client to provide credentials
```

UnsupportedFileSystemException: Kein FileSystem für Schema „s3"

Möglicherweise tritt im Thread „main“ die folgende Ausnahme auf:

```
org.apache.hadoop.fs.UnsupportedFileSystemException: No FileSystem for scheme "s3"
```

Wenn dies der Fall ist, fügen Sie der `SparkApplication`-Spezifikation die folgenden Ausnahmen hinzu:

```
 hadoopConf:
   # EMRFS filesystem
   fs.s3.customAWSCredentialsProvider:
 com.amazonaws.auth.WebIdentityTokenCredentialsProvider
   fs.s3.impl: com.amazon.ws.emr.hadoop.fs.EmrFileSystem
   fs.AbstractFileSystem.s3.impl: org.apache.hadoop.fs.s3.EMRFSDelegate
   fs.s3.buffer.dir: /mnt/s3
   fs.s3.getObject.initialSocketTimeoutMilliseconds: "2000"
   mapreduce.fileoutputcommitter.algorithm.version.emr_internal_use_only.EmrFileSystem:
 "2"
   mapreduce.fileoutputcommitter.cleanup-
failures.ignored.emr_internal_use_only.EmrFileSystem: "true"
 sparkConf:
   # Required for EMR Runtime
   spark.driver.extraClassPath: /usr/lib/hadoop-lzo/lib/*:/usr/lib/hadoop/hadoop-
aws.jar:/usr/share/aws/aws-java-sdk/*:/usr/share/aws/emr/emrfs/conf:/usr/share/aws/
emr/emrfs/lib/*:/usr/share/aws/emr/emrfs/auxlib/*:/usr/share/aws/emr/security/conf:/
usr/share/aws/emr/security/lib/*:/usr/share/aws/hmclient/lib/aws-glue-datacatalog-
spark-client.jar:/usr/share/java/Hive-JSON-Serde/hive-openx-serde.jar:/usr/share/aws/
sagemaker-spark-sdk/lib/sagemaker-spark-sdk.jar:/home/hadoop/extrajars/*
   spark.driver.extraLibraryPath: /usr/lib/hadoop/lib/native:/usr/lib/hadoop-lzo/lib/
native:/docker/usr/lib/hadoop/lib/native:/docker/usr/lib/hadoop-lzo/lib/native
   spark.executor.extraClassPath: /usr/lib/hadoop-lzo/lib/*:/usr/lib/hadoop/hadoop-
aws.jar:/usr/share/aws/aws-java-sdk/*:/usr/share/aws/emr/emrfs/conf:/usr/share/aws/
emr/emrfs/lib/*:/usr/share/aws/emr/emrfs/auxlib/*:/usr/share/aws/emr/security/conf:/
usr/share/aws/emr/security/lib/*:/usr/share/aws/hmclient/lib/aws-glue-datacatalog-
spark-client.jar:/usr/share/java/Hive-JSON-Serde/hive-openx-serde.jar:/usr/share/aws/
sagemaker-spark-sdk/lib/sagemaker-spark-sdk.jar:/home/hadoop/extrajars/*
   spark.executor.extraLibraryPath: /usr/lib/hadoop/lib/native:/usr/lib/hadoop-lzo/lib/
native:/docker/usr/lib/hadoop/lib/native:/docker/usr/lib/hadoop-lzo/lib/native
```

Service-Endpunkte und Kontingente von Amazon EMR in EKS

Im Folgenden werden die Service-Endpunkte und Service Quotas für Amazon EMR in EKS beschrieben. Um programmgesteuert eine Verbindung zu einem AWS Dienst herzustellen, verwenden Sie einen Endpunkt. Zusätzlich zu den AWS Standardendpunkten bieten einige AWS Dienste FIPS-Endpunkte in ausgewählten Regionen. Weitere Informationen finden Sie unter AWS -Service-Endpunkte. Servicekontingente, auch als Limits bezeichnet, sind die maximale Anzahl von Serviceressourcen oder -vorgängen für Ihr AWS -Konto. Weitere Informationen finden Sie unter AWS -Servicekontingente.

Service-Endpunkte

AWS-Region Name	Code	Endpunkt	Protokoll
USA Ost (Nord-Virginia)	us-east-1	`emr-containers.us-east-1.amazonaws.com`	HTTPS
USA Ost (Ohio)	us-east-2	`emr-containers.us-east-2.amazonaws.com`	HTTPS
USA West (Nordkalifornien)	us-west-1	`emr-containers.us-west-1.amazonaws.com`	HTTPS
USA West (Oregon)	us-west-2	`emr-containers.us-west-2.amazonaws.com`	HTTPS
Asien-Pazifik (Tokio)	ap-northeast-1	`emr-containers.ap-northeast-1.amazonaws.com`	HTTPS
Asien-Pazifik (Seoul)	ap-northeast-2	`emr-containers.ap-northeast-2.amazonaws.com`	HTTPS
Asia Pacific (Mumbai)	ap-south-1	`emr-containers.ap-south-1.amazonaws.com`	HTTPS

AWS-Region Name	Code	Endpunkt	Protokoll
Asien-Pazifik (Singapur)	ap-southeast-1	emr-containers.ap-southeast-1.amazonaws.com	HTTPS
Asien-Pazifik (Sydney)	ap-southeast-2	emr-containers.ap-southeast-2.amazonaws.com	HTTPS
Asien-Pazifik (Hongkong)	ap-east-1	emr-containers.ap-east-1.amazonaws.com	HTTPS
Kanada (Zentral)	ca-central-1	emr-containers.ca-central-1.amazonaws.com	HTTPS
Europe (Frankfurt)	eu-central-1	emr-containers.eu-central-1.amazonaws.com	HTTPS
Europa (Irland)	eu-west-1	emr-containers.eu-west-1.amazonaws.com	HTTPS
Europe (London)	eu-west-2	emr-containers.eu-west-2.amazonaws.com	HTTPS
Europe (Stockholm)	eu-north-1	emr-containers.eu-north-1.amazonaws.com	HTTPS
Südamerika (São Paulo)	sa-east-1	emr-containers.sa-east-1.amazonaws.com	HTTPS
Naher Osten (Bahrain)	me-south-1	emr-containers.me-south-1.amazonaws.com	HTTPS
AWS GovCloud (US-Ost)	us-gov-east-1	emr-containers.us-gov-east-1.amazonaws.com	HTTPS
AWS GovCloud (US-West)	us-gov-west-1	emr-containers.us-gov-west-1.amazonaws.com	HTTPS

Servicekontingente

Amazon EMR on EKS drosselt die folgenden API-Anfragen für jedes AWS Konto pro Region. Weitere Informationen zur Anwendung der Drosselung finden Sie unter Drosselung von API-Anforderungen in der Amazon-EC2-API-Referenz. Sie können eine Erhöhung der API-Drosselungsquoten für Ihr Konto beantragen. AWS

API-Aktion	Maximale Kapazität des Buckets	Bucket-Auffüllrate (pro Sekunde)
CancelJobRun	25	1
CreateManagedEndpoint	25	1
CreateVirtualCluster	25	1
DeleteManagedEndpoint	25	1
DeleteVirtualCluster	25	1
DescribeJobRun	100	20
DescribeManagedEndpoint	100	5
DescribeVirtualCluster	100	5
ListJobRun	100	5
ListManagedEndpoint	25	1
ListVirtualCluster	100	5
StartJobRun	25	1
At the AWS account level, the bucket maximum capacity and refill rate for the sum of all API actions listed in this table	200	20

Versionen von Amazon EMR in EKS

Eine Amazon-EMR-Version ist eine Gruppe von Open-Source-Anwendungen aus dem Big-Data-Ökosystem. Jede Version besteht aus verschiedenen Big-Data-Anwendungen, Komponenten und Features, die Sie auswählen damit Amazon EMR in EKS bereitgestellt und konfiguriert wird, wenn Sie Ihren Auftrag ausführen.

Ab den Amazon-EMR-Versionen 5.32.0 und 6.2.0 können Sie Amazon EMR in EKS bereitstellen. Diese Bereitstellungsoption ist in früheren Amazon-EMR-Versionen nicht verfügbar. Sie müssen eine unterstützte Version angeben, wenn Sie Ihren Auftrag einreichen.

Amazon EMR in EKS verwendet die folgende Form einer Versionskennung: `emr-x.x.x-latest` oder `emr-x.x.x-yyyymmdd` mit einem bestimmten Veröffentlichungsdatum. Zum Beispiel `emr-7.1.0-latest` oder `emr-7.1.0-20210129`. Wenn Sie das `-latest`-Suffix verwenden, stellen Sie sicher, dass Ihre Amazon-EMR-Version immer die neuesten Sicherheitsupdates enthält.

> **Note**
>
> Einen Vergleich zwischen Amazon EMR auf EKS und Amazon EMR auf EC2 finden Sie in den häufig gestellten Fragen zu Amazon EMR auf der Website. AWS

Themen

Versionen von Amazon EMR auf EKS 7.1.0

Auf dieser Seite werden die neuen und aktualisierten Funktionen für Amazon EMR beschrieben, die spezifisch für die Bereitstellung von Amazon EMR in EKS sind. Einzelheiten zur Ausführung von Amazon EMR auf Amazon EC2 und zur Amazon EMR 7.1.0-Version im Allgemeinen finden Sie unter Amazon EMR 7.1.0 im Amazon EMR-Versionshandbuch.

Amazon EMR auf EKS 7.1-Versionen

Die folgenden Amazon EMR 7.1.0-Versionen sind für Amazon EMR auf EKS verfügbar. Wählen Sie eine bestimmte EMR-7.1.0-xxxx-Version aus, um weitere Details wie das zugehörige Container-Image-Tag anzuzeigen.

Flink releases

Die folgenden Amazon EMR 7.1.0-Versionen sind für Amazon EMR auf EKS verfügbar, wenn Sie Flink-Anwendungen ausführen.

- emr-7.1.0-flink-aktuell
- emr-7.1.0-flink-20240321

Spark releases

Die folgenden Amazon EMR 7.1.0-Versionen sind für Amazon EMR auf EKS verfügbar, wenn Sie Spark-Anwendungen ausführen.

- emr-7.1.0-aktuell

- emr-7.1.0-20240321
- emr-7.1.0-spark-rapids-latest
- emr-7.1.0-spark-rapids-20240321
- emr-7.1.0-java11-latest
- emr-7.1.0-java11-20240321
- emr-7.1.0-java8-latest
- emr-7.1.0-java8-20240321
- emr-7.1.0-spark-rapids-java8-latest
- emr-7.1.0-spark-rapids-java8-20240321
- notebook-spark/emr-7.1.0-latest
- notebook-spark/emr-7.1.0-20240321
- notebook-spark/emr-7.1.0-spark-rapids-latest
- notebook-spark/emr-7.1.0-spark-rapids-20240321
- notebook-spark/emr-7.1.0-java11-latest
- notebook-spark/emr-7.1.0-java11-20240321
- notebook-spark/emr-7.1.0-java8-latest
- notebook-spark/emr-7.1.0-java8-20240321
- notebook-spark/emr-7.1.0-spark-rapids-java8-latest
- notebook-spark/emr-7.1.0-spark-rapids-java8-20240321
- notebook-python/emr-7.1.0-latest
- notebook-python/emr-7.1.0-20240321
- notebook-python/emr-7.1.0-spark-rapids-latest
- notebook-python/emr-7.1.0-spark-rapids-20240321
- notebook-python/emr-7.1.0-java11-latest
- notebook-python/emr-7.1.0-java11-20240321
- notebook-python/emr-7.1.0-java8-latest
- notebook-python/emr-7.1.0-java8-20240321
- notebook-python/emr-7.1.0-spark-rapids-java8-latest
- notebook-python/emr-7.1.0-spark-rapids-java8-20240321
- livy/emr-7.1.0-latest

- livy/emr-7.1.0-20240321
- livy/emr-7.1.0-java11-latest
- livy/emr-7.1.0-java11-20240321
- livy/emr-7.1.0-java8-latest
- livy/emr-7.1.0-java8-20240321

Versionshinweise

Versionshinweise für Amazon EMR auf EKS 7.1.0

- Unterstützte Anwendungen – AWS SDK for Java 2.23.18 and 1.12.656, Apache Spark 3.5.0-amzn-1, Apache Hudi 0.14.1-amzn-0, Apache Iceberg 1.4.3-amzn-0, Delta 3.0.0, Apache Spark RAPIDS 23.10.0-amzn-1, Jupyter Enterprise Gateway 2.6.0, Apache Flink 1.18.1-amzn-0, Flink Operator 1.6.1-amzn-1
- Unterstützte Komponenten – `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg`, `spark-kubernetes`.
- Unterstützte Konfigurationsklassifizierungen

  Zur Verwendung mit StartJobRunund CreateManagedEndpointAPIs:

Klassifizierungen	Beschreibungen
`core-site`	Ändern Sie die Werte in der `core-site.xml` -Hadoop-Datei.
`emrfs-site`	Ändert die EMRFS-Einstellungen.
`spark-metrics`	Ändern Sie die Werte in der `metrics.properties` -Spark-Datei.
`spark-defaults`	Ändern Sie die Werte in der `spark-defaults.conf` -Spark-Datei.
`spark-env`	Ändert die Werte in der Spark-Umgebung.
`spark-hive-site`	Ändern Sie die Werte in der `hive-site.xml` -Spark-Datei.

Klassifizierungen	Beschreibungen
`spark-log4j2`	Ändern Sie die Werte in der `log4j2.properties` -Spark-Datei.
`emr-job-submitter`	Konfiguration für den Auftragsübermittler-Pod.

Speziell zur Verwendung mit `CreateManagedEndpoint`APIs:

Klassifizierungen	Beschreibungen
`jeg-config`	Ändern Sie die Werte in der Jupyter-Enterprise-Gateway-Datei `jupyter_enterprise_gateway_config.py` .
`jupyter-kernel-overrides`	Ändern Sie den Wert für das Kernel-Image in der Jupyter-Kernel-Spec-Datei.

Mithilfe von Konfigurationsklassifizierungen können Sie Anwendungen anpassen. Diese entsprechen häufig einer XML-Konfigurationsdatei für die Anwendung, z. B. `spark-hive-site.xml` Weitere Informationen finden Sie unter Konfigurieren von Anwendungen.

Bemerkenswerte Features

Die folgenden Funktionen sind in der Version 7.1.0 von Amazon EMR on EKS enthalten.

- Apache Livy-Unterstützung für Amazon EMR auf EKS — Mit Amazon EMR auf EKS-Versionen 7.1.0 und höher können Sie Apache Livy auf einem Amazon EKS-Cluster verwenden, um eine Apache Livy-REST-Schnittstelle zum Senden von Spark-Jobs oder Spark-Codefragmenten zu erstellen. Auf diese Weise können Sie Ergebnisse synchron und asynchron abrufen und gleichzeitig die Vorteile von Amazon EMR on EKS nutzen, wie z. B. die für Amazon EMR optimierte Spark-Laufzeit, SSL-fähige Livy-Endpunkte und eine programmatische Einrichtung.

Änderungen

Die folgenden Änderungen sind in der Version 7.1.0 von Amazon EMR on EKS enthalten.

- Mit Amazon EMR auf EKS 7.1.0 und höher verwendet Apache Flink jetzt standardmäßig Java 17-Laufzeit.

emr-7.1.0-aktuell

Versionshinweise: `emr-7.1.0-latest` verweist derzeit auf `emr-7.1.0-20240321`.

Regionen: `emr-7.1.0-latest` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-7.1.0:latest`

emr-7.1.0-20240321

Versionshinweise: `7.1.0-20240321` wurde im Dezember 2023 veröffentlicht. Dies ist die erste Version von Amazon EMR 7.1.0 (Spark).

Regionen: `emr-7.1.0-20240321` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-7.1.0:20240321`

emr-7.1.0-flink-aktuell

Versionshinweise: `emr-7.1.0-flink-latest` verweist derzeit auf `emr-7.1.0-flink-20240321`.

Regionen: `emr-7.1.0-flink-latest` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-7.1.0-flink:latest`

emr-7.1.0-flink-20240321

Versionshinweise: `7.1.0-flink-20240321` wurde im Dezember 2023 veröffentlicht. Dies ist die erste Version von Amazon EMR 7.1.0 (Flink).

Regionen: `emr-7.1.0-flink-20240321` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-7.1.0-flink:20240321`

Versionen von Amazon EMR in EKS 7.0.0

Auf dieser Seite werden die neuen und aktualisierten Funktionen für Amazon EMR beschrieben, die spezifisch für die Bereitstellung von Amazon EMR in EKS sind. Einzelheiten zur Ausführung von Amazon EMR in Amazon EC2 und zur Amazon-EMR-Version 7.0.0 im Allgemeinen finden Sie unter Amazon EMR 7.0.0 im Amazon-EMR-Versionshandbuch.

Amazon EMR in EKS 7.0.-Versionen

Die folgenden Versionen von Amazon EMR 7.0.0 sind für Amazon EMR in EKS verfügbar. Wählen Sie eine bestimmte emr-7.0.0-XXXX-Version aus, um weitere Details wie das zugehörige Container-Image-Tag anzuzeigen.

Flink releases

Die folgenden Versionen von Amazon EMR 7.0.0 sind für Amazon EMR in EKS verfügbar, wenn Sie Flink-Anwendungen ausführen.

- emr-7.0.0-flink-latest
- emr-7.0.0-flink-2024321
- emr-7.0.0-flink-20231211

Spark releases

Die folgenden Versionen von Amazon EMR 7.0.0 sind für Amazon EMR in EKS verfügbar, wenn Sie Spark-Anwendungen ausführen.

- emr-7.0.0-latest

- emr-7.0.0-20231211
- emr-7.0.0-spark-rapids-latest
- emr-7.0.0-spark-rapids-20231211
- emr-7.0.0-java11-latest
- emr-7.0.0-java11-20231211
- emr-7.0.0-java8-latest
- emr-7.0.0-java8-20231211
- emr-7.0.0-spark-rapids-java8-latest
- emr-7.0.0-spark-rapids-java8-20231211
- notebook-spark/emr-7.0.0-latest
- notebook-spark/emr-7.0.0-20231211
- notebook-spark/emr-7.0.0-spark-rapids-latest
- notebook-spark/emr-7.0.0-spark-rapids-20231211
- notebook-spark/emr-7.0.0-java11-latest
- notebook-spark/emr-7.0.0-java11-20231211
- notebook-spark/emr-7.0.0-java8-latest
- notebook-spark/emr-7.0.0-java8-20231211
- notebook-spark/emr-7.0.0-spark-rapids-java8-latest
- notebook-spark/emr-7.0.0-spark-rapids-java8-20231211
- notebook-python/emr-7.0.0-latest
- notebook-python/emr-7.0.0-20231211
- notebook-python/emr-7.0.0-spark-rapids-latest
- notebook-python/emr-7.0.0-spark-rapids-20231211
- notebook-python/emr-7.0.0-java11-latest
- notebook-python/emr-7.0.0-java11-20231211
- notebook-python/emr-7.0.0-java8-latest
- notebook-python/emr-7.0.0-java8-20231211
- notebook-python/emr-7.0.0-spark-rapids-java8-latest
- notebook-python/emr-7.0.0-spark-rapids-java8-20231211

Versionshinweise

Versionshinweise für Amazon EMR in EKS 7.0.0

- Unterstützte Anwendungen – AWS SDK for Java 2.20.160-amzn-0 and 1.12.595, Apache Spark 3.5.0-amzn-0, Apache Flink 1.18.0-amzn-0, Flink Operator 1.6.1, Apache Hudi 0.14.0-amzn-1, Apache Iceberg 1.4.2-amzn-0, Delta 3.0.0, Apache Spark RAPIDS 23.10.0-amzn-0, Jupyter Enterprise Gateway 2.6.0
- Unterstützte Komponenten – `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg`, `spark-kubernetes`.
- Unterstützte Konfigurationsklassifizierungen

  Zur Verwendung mit StartJobRunAPIs: CreateManagedEndpoint

Klassifizierungen	Beschreibungen
`core-site`	Ändern Sie die Werte in der `core-site.xml` -Hadoop-Datei.
`emrfs-site`	Ändert die EMRFS-Einstellungen.
`spark-metrics`	Ändern Sie die Werte in der `metrics.properties` -Spark-Datei.
`spark-defaults`	Ändern Sie die Werte in der `spark-defaults.conf` -Spark-Datei.
`spark-env`	Ändert die Werte in der Spark-Umgebung.
`spark-hive-site`	Ändern Sie die Werte in der `hive-site.xml` -Spark-Datei.
`spark-log4j`	Ändern Sie die Werte in der `log4j2.properties` -Spark-Datei.
`emr-job-submitter`	Konfiguration für den Auftragsübermittler-Pod.

  Speziell zur Verwendung mit CreateManagedEndpointAPIs:

Klassifizierungen	Beschreibungen
`jeg-config`	Ändern Sie die Werte in der Jupyter-Enterprise-Gateway-Datei `jupyter_enterprise_gateway_config.py` .
`jupyter-kernel-overrides`	Ändern Sie den Wert für das Kernel-Image in der Jupyter-Kernel-Spec-Datei.

Mithilfe von Konfigurationsklassifizierungen können Sie Anwendungen anpassen. Diese entsprechen häufig einer XML-Konfigurationsdatei für die Anwendung, z. B. `spark-hive-site.xml` Weitere Informationen finden Sie unter Konfigurieren von Anwendungen.

Bemerkenswerte Features

Die folgenden Features sind in der Version 7.0 von Amazon EMR in EKS enthalten.

- Anwendungs-Upgrades – Zu den Anwendungs-Upgrades von Amazon EMR in EKS 7.0.0 gehören Spark 3.5, Flink 1.18 und Flink Operator 1.6.1.
- Autotuning der Flink-Autoscaler-Parameter – Die Standardparameter, die Flink Autoscaler für seine Skalierungsberechnungen verwendet, sind für einen bestimmten Auftrag möglicherweise nicht der optimale Wert. Amazon EMR in EKS 7.0.0 verwendet historische Trends bestimmter erfasster Metriken, um den optimalen, auf den Auftrag zugeschnittenen Parameter zu berechnen.

Änderungen

Die folgenden Änderungen sind in der Version 7.0 von Amazon EMR in EKS enthalten.

- Amazon Linux 2023 – Mit Amazon EMR in EKS 7.0.0 und höher basieren alle Container-Images auf Amazon Linux 2023.
- Spark verwendet Java 17 als Standard-Laufzeit – Amazon EMR in EKS 7.0.0 Spark verwendet Java 17 als Standard-Laufzeit. Bei Bedarf können Sie zur Verwendung von Java 8 oder Java 11 mit dem entsprechenden Release-Label wechseln, wie in der Amazon EMR in EKS 7.0.-Versionen-Liste angegeben.

emr-7.0.0-latest

Versionshinweise: `emr-7.0.0-latest` verweist derzeit auf `emr-7.0.0-2024321`.

Regionen: `emr-7.0.0-latest` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-7.0.0:latest`

emr-7.0.0-2024321

Versionshinweise: `7.0.0-2024321` wurde am 11. März 2024 veröffentlicht. Im Vergleich zur vorherigen Version wurde diese Version mit den kürzlich aktualisierten Amazon Linux-Paketen und kritischen Korrekturen aktualisiert.

Regionen: `emr-7.0.0-2024321` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-7.0.0:2024321`

emr-7.0.0-20231211

Versionshinweise: `7.0.0-20231211` wurde im Dezember 2023 veröffentlicht. Dies ist die erste veröffentlichte Version von Amazon EMR 7.0.0 (Spark).

Regionen: `emr-7.0.0-20231211` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-7.0.0:20231211`

emr-7.0.0-flink-latest

Versionshinweise: `emr-7.0.0-flink-latest` verweist derzeit auf `emr-7.0.0-flink-2024321`.

Regionen: `emr-7.0.0-flink-latest` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-7.0.0-flink:latest`

emr-7.0.0-flink-2024321

Versionshinweise: wurde am 11. März 2024 veröffentlicht. `7.0.0-flink-2024321` Im Vergleich zur vorherigen Version wurde diese Version mit den kürzlich aktualisierten Amazon Linux-Paketen und kritischen Korrekturen aktualisiert.

Regionen: `emr-7.0.0-flink-2024321` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-7.0.0-flink:2024321`

emr-7.0.0-flink-20231211

Versionshinweise: `7.0.0-flink-20231211` wurde im Dezember 2023 veröffentlicht. Dies ist die erste veröffentlichte Version von Amazon EMR 7.0.0 (Flink).

Regionen: `emr-7.0.0-flink-20231211` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-7.0.0-flink:20231211`

Amazon EMR in EKS 6.15.0-Versionen

Auf dieser Seite werden die neuen und aktualisierten Funktionen für Amazon EMR beschrieben, die spezifisch für die Bereitstellung von Amazon EMR in EKS sind. Einzelheiten zur Ausführung von Amazon EMR in Amazon EC2 und zur Amazon-EMR-Version 6.15.0 im Allgemeinen finden Sie unter Amazon EMR 6.15.0 im Amazon-EMR-Versionshandbuch.

Amazon EMR in EKS 6.15-Versionen

Die folgenden Amazon-EMR-6.15.0-Versionen sind für Amazon EMR in EKS verfügbar. Wählen Sie eine bestimmte emr-6.15.0-XXXX-Version aus, um weitere Details wie das zugehörige Container-Image-Tag anzuzeigen.

Flink releases

Die folgenden Versionen von Amazon EMR 6.15.0 sind für Amazon EMR in EKS verfügbar, wenn Sie Flink-Anwendungen ausführen.

- emr-6.15.0-flink-latest
- emr-6.15.0-flink-20240105
- emr-6.15.0-flink-20231109

Spark releases

Die folgenden Versionen von Amazon EMR 6.15.0 sind für Amazon EMR in EKS verfügbar, wenn Sie Spark-Anwendungen ausführen.

- emr-6.15.0-latest
- emr-6.15.0-20231109
- emr-6.15.0-spark-rapids-latest
- emr-6.15.0-spark-rapids-20231109
- emr-6.15.0-java11-latest
- emr-6.15.0-java11-20231109
- emr-6.15.0-java17-latest
- emr-6.15.0-java17-20231109
- emr-6.15.0-java17-al2023-latest
- emr-6.15.0-java17-al2023-20231109
- emr-6.15.0-spark-rapids-java17-latest
- emr-6.15.0-spark-rapids-java17-20231109
- emr-6.15.0-spark-rapids-java17-al2023-latest
- emr-6.15.0-spark-rapids-java17-al2023-20231109
- notebook-spark/emr-6.15.0-latest
- notebook-spark/emr-6.15.0-20231109
- notebook-spark/emr-6.15.0-spark-rapids-latest
- notebook-spark/emr-6.15.0-spark-rapids-20231109
- notebook-spark/emr-6.15.0-java11-latest
- notebook-spark/emr-6.15.0-java11-20231109
- notebook-spark/emr-6.15.0-java17-latest
- notebook-spark/emr-6.15.0-java17-20231109
- notebook-spark/emr-6.15.0-java17-al2023-latest

- notebook-spark/emr-6.15.0-java17-al2023-20231109
- notebook-python/emr-6.15.0-latest
- notebook-python/emr-6.15.0-20231109
- notebook-python/emr-6.15.0-spark-rapids-latest
- notebook-python/emr-6.15.0-spark-rapids-20231109
- notebook-python/emr-6.15.0-java11-latest
- notebook-python/emr-6.15.0-java11-20231109
- notebook-python/emr-6.15.0-java17-latest
- notebook-python/emr-6.15.0-java17-20231109
- notebook-python/emr-6.15.0-java17-al2023-latest
- notebook-python/emr-6.15.0-java17-al2023-20231109

Versionshinweise

Versionshinweise für Amazon EMR in EKS 6.15.0

- Unterstützte Anwendungen – AWS SDK for Java 1.12.569, Apache Spark 3.4.1-amzn-2, Apache Flink 1.17.1-amzn-1, Apache Hudi 0.14.0-amzn-0, Apache Iceberg 1.4.0-amzn-0, Delta 2.4.0, Apache Spark RAPIDS 23.08.01-amzn-0, Jupyter Enterprise Gateway 2.6.0
- Unterstützte Komponenten – `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg`, `spark-kubernetes`.
- Unterstützte Konfigurationsklassifizierungen

  Zur Verwendung mit StartJobRun CreateManagedEndpointAPIs:

Klassifizierungen	Beschreibungen
`core-site`	Ändern Sie die Werte in der `core-site.xml` -Hadoop-Datei.
`emrfs-site`	Ändert die EMRFS-Einstellungen.
`spark-metrics`	Ändern Sie die Werte in der `metrics.properties` -Spark-Datei.

Klassifizierungen	Beschreibungen
`spark-defaults`	Ändern Sie die Werte in der `spark-defaults.conf` -Spark-Datei.
`spark-env`	Ändert die Werte in der Spark-Umgebung.
`spark-hive-site`	Ändern Sie die Werte in der `hive-site.xml` -Spark-Datei.
`spark-log4j`	Ändern Sie die Werte in der `log4j2.properties` -Spark-Datei.
`emr-job-submitter`	Konfiguration für den Auftragsübermittler-Pod.

Speziell zur Verwendung mit CreateManagedEndpointAPIs:

Klassifizierungen	Beschreibungen
`jeg-config`	Ändern Sie die Werte in der Jupyter-Enterprise-Gateway-Datei `jupyter_enterprise_gateway_config.py` .
`jupyter-kernel-overrides`	Ändern Sie den Wert für das Kernel-Image in der Jupyter-Kernel-Spec-Datei.

Mithilfe von Konfigurationsklassifizierungen können Sie Anwendungen anpassen. Diese entsprechen häufig einer XML-Konfigurationsdatei für die Anwendung, z. B. `spark-hive-site.xml` Weitere Informationen finden Sie unter Konfigurieren von Anwendungen.

Bemerkenswerte Features

Die folgenden Features sind in der Version 6.15 von Amazon EMR in EKS enthalten.

- Amazon EMR in EKS mit Apache Flink – Mit Amazon EMR in EKS 6.15.0 können Sie Ihre Apache-Flink-basierte Anwendung zusammen mit anderen Anwendungstypen auf demselben Amazon-EKS-Cluster ausführen. Dies trägt dazu bei, die Ressourcennutzung zu verbessern und das

Infrastrukturmanagement zu vereinfachen. Sie können Spot-Instances in einer Flink-Anwendung mit ordnungsgemäßer Stilllegung nutzen und mit Amazon EBS schnellere Neustartzeiten mit differenzierter Wiederherstellung und aufgabenlokaler Wiederherstellung erreichen. Zu den Funktionen für Barrierefreiheit und Überwachung gehören die Möglichkeit, eine Flink-Anwendung mit in Amazon S3 gespeicherten Jars zu starten, der Zugriff auf den AWS Glue-Datenkatalog, die Überwachungsintegration mit Amazon S3 und Amazon CloudWatch sowie die Rotation von Container-Protokollen.

emr-6.15.0-latest

Versionshinweise: `emr-6.15.0-latest` verweist derzeit auf `emr-6.15.0-20240105`.

Regionen: `emr-6.15.0-latest` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.15.0:latest`

emr-6.15.0-20240105

Versionshinweise: `6.15.0-20240105` wurde am 17. Januar 2024 veröffentlicht. Im Vergleich zur vorherigen Version wurde diese Version mit den kürzlich aktualisierten Amazon Linux-Paketen und kritischen Korrekturen aktualisiert.

Regionen: `emr-6.15.0-20240105` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.15.0:20240105`

emr-6.15.0-20231109

Versionshinweise: `6.15.0-20231109` wurde am 17. November 2023 veröffentlicht. Dies ist die erste veröffentlichte Version von Amazon EMR 6.15.0.

Regionen: `emr-6.15.0-20231109` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.15.0:20231109`

emr-6.15.0-flink-latest

Versionshinweise: `emr-6.15.0-flink-latest` verweist derzeit auf `emr-6.15.0-flink-20240105`.

Regionen: `emr-6.15.0-flink-latest` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.15.0-flink:latest`

emr-6.15.0-flink-20240105

Versionshinweise: wurde am 17. Januar 2024 veröffentlicht. `6.15.0-flink-20240105` Im Vergleich zur vorherigen Version wurde diese Version mit den kürzlich aktualisierten Amazon Linux-Paketen und kritischen Korrekturen aktualisiert.

Regionen: `emr-6.15.0-flink-20240105` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.15.0-flink:20240105`

emr-6.15.0-flink-20231109

Versionshinweise: `6.15.0-flink-20231109` wurde am 17. November 2023 veröffentlicht. Dies ist die erste veröffentlichte Version von Amazon EMR 6.15.0.

Regionen: `emr-6.15.0-flink-20231109` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.15.0-flink:20231109`

Versionen von Amazon EMR in EKS 6.14.0

Auf dieser Seite werden die neuen und aktualisierten Funktionen für Amazon EMR beschrieben, die spezifisch für die Bereitstellung von Amazon EMR in EKS sind. Einzelheiten zur Ausführung von Amazon EMR auf Amazon EC2 und zur Amazon-EMR-Version 6.14.0 im Allgemeinen finden Sie unter Amazon EMR 6.14.0 im Amazon-EMR-Versionshandbuch.

Versionen von Amazon EMR in EKS 6.14

Die folgenden Amazon-EMR-6.14.0-Versionen sind für Amazon EMR in EKS verfügbar. Wählen Sie eine bestimmte EMR-6.14.0-xxxx-Version aus, um weitere Details wie das zugehörige Container-Image-Tag anzuzeigen.

- emr-6.14.0-latest
- emr-6.14.0-20231005
- emr-6.14.0-spark-rapids-latest
- emr-6.14.0-spark-rapids-20231005
- emr-6.14.0-java11-latest
- emr-6.14.0-java11-20231005
- emr-6.14.0-java17-latest
- emr-6.14.0-java17-20231005
- emr-6.14.0-java17-al2023-latest
- emr-6.14.0-java17-al2023-20231005
- emr-6.14.0-spark-rapids-java17-latest
- emr-6.14.0-spark-rapids-java17-20231005
- emr-6.14.0-spark-rapids-java17-al2023-latest
- emr-6.14.0-spark-rapids-java17-al2023-20231005
- notebook-spark/emr-6.14.0-latest
- notebook-spark/emr-6.14.0-20231005
- notebook-spark/emr-6.14.0-spark-rapids-latest
- notebook-spark/emr-6.14.0-spark-rapids-20231005
- notebook-spark/emr-6.14.0-java11-latest
- notebook-spark/emr-6.14.0-java11-20231005
- notebook-spark/emr-6.14.0-java17-latest
- notebook-spark/emr-6.14.0-java17-20231005
- notebook-spark/emr-6.14.0-java17-al2023-latest
- notebook-spark/emr-6.14.0-java17-al2023-20231005
- notebook-python/emr-6.14.0-latest
- notebook-python/emr-6.14.0-20231005

- notebook-python/emr-6.14.0-spark-rapids-latest
- notebook-python/emr-6.14.0-spark-rapids-20231005
- notebook-python/emr-6.14.0-java11-latest
- notebook-python/emr-6.14.0-java11-20231005
- notebook-python/emr-6.14.0-java17-latest
- notebook-python/emr-6.14.0-java17-20231005
- notebook-python/emr-6.14.0-java17-al2023-latest
- notebook-python/emr-6.14.0-java17-al2023-20231005

Versionshinweise

Versionshinweise für Amazon EMR in EKS 6.14.0

- Unterstützte Anwendungen - AWS SDK for Java 1.12.543, Apache Spark 3.4.1-amzn-1, Apache Hudi 0.13.1-amzn-2, Apache Iceberg 1.3.0-amzn-0, Delta 2.4.0, Apache Spark RAPIDS 23.06.0-amzn-2, Jupyter Enterprise Gateway 2.7.0
- Unterstützte Komponenten – `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg`, `spark-kubernetes`.
- Unterstützte Konfigurationsklassifizierungen

  StartJobRunZur CreateManagedEndpointVerwendung mit und APIs:

Klassifizierungen	Beschreibungen
`core-site`	Ändern Sie die Werte in der `core-site.xml` -Hadoop-Datei.
`emrfs-site`	Ändert die EMRFS-Einstellungen.
`spark-metrics`	Ändern Sie die Werte in der `metrics.properties` -Spark-Datei.
`spark-defaults`	Ändern Sie die Werte in der `spark-defaults.conf` -Spark-Datei.
`spark-env`	Ändert die Werte in der Spark-Umgebung.

Klassifizierungen	Beschreibungen
`spark-hive-site`	Ändern Sie die Werte in der `hive-site.xml` -Spark-Datei.
`spark-log4j`	Ändern Sie die Werte in der `log4j2.properties` -Spark-Datei.
`emr-job-submitter`	Konfiguration für den Auftragsübermittler-Pod.

Speziell zur Verwendung mit CreateManagedEndpointAPIs:

Klassifizierungen	Beschreibungen
`jeg-config`	Ändern Sie die Werte in der Jupyter-Enterprise-Gateway-Datei `jupyter_enterprise_gateway_config.py` .
`jupyter-kernel-overrides`	Ändern Sie den Wert für das Kernel-Image in der Jupyter-Kernel-Spec-Datei.

Mithilfe von Konfigurationsklassifizierungen können Sie Anwendungen anpassen. Diese entsprechen häufig einer XML-Konfigurationsdatei für die Anwendung, z. B. `spark-hive-site.xml` Weitere Informationen finden Sie unter Konfigurieren von Anwendungen.

Bemerkenswerte Features

Die folgenden Features sind in der Version 6.14 von Amazon EMR in EKS enthalten.

- Apache Livy-Unterstützung – Amazon EMR in EKS unterstützt jetzt Apache Livy mit `spark-submit`.

emr-6.14.0-latest

Versionshinweise: `emr-6.14.0-latest` verweist derzeit auf `emr-6.14.0-20231005`.

Regionen: `emr-6.14.0-latest` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.14.0:latest`

emr-6.14.0-20231005

Versionshinweise: `6.14.0-20231005` wurde am 17. Oktober 2023 veröffentlicht. Dies ist die erste veröffentlichte Version von Amazon EMR 6.14.0.

Regionen: `emr-6.14.0-20231005` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.14.0:20231005`

Versionen von Amazon EMR in EKS 6.13.0

Auf dieser Seite werden die neuen und aktualisierten Funktionen für Amazon EMR beschrieben, die spezifisch für die Bereitstellung von Amazon EMR in EKS sind. Einzelheiten zur Ausführung von Amazon EMR auf Amazon EC2 und zur Amazon-EMR-Version 6.13.0 im Allgemeinen finden Sie unter Amazon EMR 6.13.0 im Amazon-EMR-Versionshandbuch.

Versionen von Amazon EMR in EKS 6.13

Die folgenden Versionen von Amazon EMR in EKS 6.13.0 sind für Amazon EMR in EKS verfügbar. Wählen Sie eine bestimmte EMR-6.13.0-xxxx-Version aus, um weitere Details wie das zugehörige Container-Image-Tag anzuzeigen.

- emr-6.13.0-latest
- emr-6.13.0-20230814
- emr-6.13.0-spark-rapids-latest
- emr-6.13.0-spark-rapids-20230814
- emr-6.13.0-java11-latest
- emr-6.13.0-java11-20230814
- emr-6.13.0-java17-latest

- emr-6.13.0-java17-20230814
- emr-6.13.0-java17-al2023-latest
- emr-6.13.0-java17-al2023-20230814
- emr-6.13.0-spark-rapids-java17-latest
- emr-6.13.0-spark-rapids-java17-20230814
- emr-6.13.0-spark-rapids-java17-al2023-latest
- emr-6.13.0-spark-rapids-java17-al2023-20230814
- notebook-spark/emr-6.13.0-latest
- notebook-spark/emr-6.13.0-20230814
- notebook-spark/emr-6.13.0-spark-rapids-latest
- notebook-spark/emr-6.13.0-spark-rapids-20230814
- notebook-spark/emr-6.13.0-java11-latest
- notebook-spark/emr-6.13.0-java11-20230814
- notebook-spark/emr-6.13.0-java17-latest
- notebook-spark/emr-6.13.0-java17-20230814
- notebook-spark/emr-6.13.0-java17-al2023-latest
- notebook-spark/emr-6.13.0-java17-al2023-20230814
- notebook-python/emr-6.13.0-latest
- notebook-python/emr-6.13.0-20230814
- notebook-python/emr-6.13.0-spark-rapids-latest
- notebook-python/emr-6.13.0-spark-rapids-20230814
- notebook-python/emr-6.13.0-java11-latest
- notebook-python/emr-6.13.0-java11-20230814
- notebook-python/emr-6.13.0-java17-latest
- notebook-python/emr-6.13.0-java17-20230814
- notebook-python/emr-6.13.0-java17-al2023-latest
- notebook-python/emr-6.13.0-java17-al2023-20230814

Versionshinweise

Versionshinweise für Amazon EMR in EKS 6.13.0

- Unterstützte Anwendungen - AWS SDK for Java 1.12.513, Apache Spark 3.4.1-amzn-0, Apache Hudi 0.13.1-amzn-0, Apache Iceberg 1.3.0-amzn-0, Delta 2.4.0, Apache Spark RAPIDS 23.06.0-amzn-1, Jupyter Enterprise Gateway 2.6.0.amzn
- Unterstützte Komponenten – `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg`, `spark-kubernetes`.
- Unterstützte Konfigurationsklassifizierungen

  StartJobRunZur CreateManagedEndpointVerwendung mit APIs:

Klassifizierungen	Beschreibungen
`core-site`	Ändern Sie die Werte in der `core-site.xml` -Hadoop-Datei.
`emrfs-site`	Ändert die EMRFS-Einstellungen.
`spark-metrics`	Ändern Sie die Werte in der `metrics.properties` -Spark-Datei.
`spark-defaults`	Ändern Sie die Werte in der `spark-defaults.conf` -Spark-Datei.
`spark-env`	Ändert die Werte in der Spark-Umgebung.
`spark-hive-site`	Ändern Sie die Werte in der `hive-site.xml` -Spark-Datei.
`spark-log4j`	Ändern Sie die Werte in der `log4j2.properties` -Spark-Datei.
`emr-job-submitter`	Konfiguration für den Auftragsübermittler-Pod.

  Speziell zur Verwendung mit CreateManagedEndpointAPIs:

Klassifizierungen	Beschreibungen
`jeg-config`	Ändern Sie die Werte in der Jupyter-Enterprise-Gateway-Datei `jupyter_enterprise_gateway_config.py` .
`jupyter-kernel-overrides`	Ändern Sie den Wert für das Kernel-Image in der Jupyter-Kernel-Spec-Datei.

Mithilfe von Konfigurationsklassifizierungen können Sie Anwendungen anpassen. Diese entsprechen häufig einer XML-Konfigurationsdatei für die Anwendung, z. B. `spark-hive-site.xml` Weitere Informationen finden Sie unter Konfigurieren von Anwendungen.

Bemerkenswerte Features

Die folgenden Features sind in der Version 6.13 von Amazon EMR in EKS enthalten.

- Amazon Linux 2023 – Mit Amazon EMR in EKS 6.13 und höher können Sie Spark mit AL2023 als Betriebssystem zusammen mit Java-17-Laufzeit starten. Verwenden Sie dazu die Versionskennung mit `al2023` im Namen. Zum Beispiel: `emr-6.13.0-java17-al2023-latest`. Wir empfehlen Ihnen, Leistungstests zu validieren und durchzuführen, bevor Sie Ihre Produktions-Workloads auf AL2023 und Java 17 umstellen.
- Amazon EMR in EKS mit Apache Flink (öffentliche Vorversion) – Amazon EMR in EKS-Versionen 6.13 und höher unterstützen Apache Flink, verfügbar als öffentliche Vorversion. Mit diesem Start können Sie Ihre Apache Flink-basierte Anwendung zusammen mit anderen Anwendungstypen auf demselben Amazon-EKS-Cluster ausführen. Dies trägt dazu bei, die Ressourcennutzung zu verbessern und das Infrastrukturmanagement zu vereinfachen. Wenn Sie bereits Big-Data-Frameworks auf Amazon EKS ausführen, können Sie jetzt Amazon EMR Ihre Bereitstellung und Verwaltung automatisieren lassen.

emr-6.13.0-latest

Versionshinweise: `emr-6.13.0-latest` verweist derzeit auf `emr-6.13.0-20230814`.

Regionen: `emr-6.13.0-latest` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.13.0:latest`

emr-6.13.0-20230814

Versionshinweise: `6.13.0-20230814` wurde am 7. September 2023 veröffentlicht. Dies ist die erste veröffentlichte Version von Amazon EMR 6.13.0.

Regionen: `emr-6.13.0-20230814` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.13.0:20230814`

Versionen von Amazon EMR in EKS 6.12.0

Auf dieser Seite werden die neuen und aktualisierten Funktionen für Amazon EMR beschrieben, die spezifisch für die Bereitstellung von Amazon EMR in EKS sind. Einzelheiten zur Ausführung von Amazon EMR auf Amazon EC2 und zur Amazon-EMR-Version 6.12.0 im Allgemeinen finden Sie unter Amazon EMR 6.12.0. im Amazon-EMR-Versionshandbuch.

Versionen von Amazon EMR in EKS 6.12

Die folgenden Amazon-EMR-6.12.0-Versionen sind für Amazon EMR in EKS verfügbar. Wählen Sie eine bestimmte EMR-6.12.0-xxxx-Version aus, um weitere Details wie das zugehörige Container-Image-Tag anzuzeigen.

- emr-6.12.0-latest
- emr-6.12.0-20240321
- emr-6.12.0-20230701
- emr-6.12.0- spark-rapids-latest
- emr-6.12.0-spark-rapids-20230701
- emr-6.12.0-java11-latest
- emr-6.12.0-java11-20230701
- emr-6.12.0-java17-latest

- emr-6.12.0-java17-20230701
- emr-6.12.0- 17 aktuell spark-rapids-java
- emr-6.12.0- spark-rapids-java 17-20230701
- notebook-spark/emr-6.12.0-latest
- notebook-spark/emr-6.12.0-20230701
- Notebook-Spark/EMR-6.12.0- spark-rapids-latest
- notebook-spark/emr-6.12.0-spark-rapids-20230701
- notebook-python/emr-6.12.0-latest
- notebook-python/emr-6.12.0-20230701
- Notebook-Python/EMR-6.12.0- spark-rapids-latest
- notebook-python/emr-6.12.0-spark-rapids-20230701

Versionshinweise

Versionshinweise für Amazon EMR in EKS 6.12.0

- Unterstützte Anwendungen - AWS SDK for Java 1.12.490, Apache Spark 3.4.0-amzn-0, Apache Hudi 0.13.1-amzn-0, Apache Iceberg 1.3.0-amzn-0, Delta 2.4.0, Apache Spark RAPIDS 23.06.0-amzn-0, Jupyter Enterprise Gateway 2.6.0
- Unterstützte Komponenten – `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg`, `spark-kubernetes`.
- Unterstützte Konfigurationsklassifizierungen

  StartJobRun CreateManagedEndpointZur Verwendung mit APIs:

Klassifizierungen	Beschreibungen
`core-site`	Ändern Sie die Werte in der `core-site.xml` -Hadoop-Datei.
`emrfs-site`	Ändert die EMRFS-Einstellungen.
`spark-metrics`	Ändern Sie die Werte in der `metrics.properties` -Spark-Datei.

Klassifizierungen	Beschreibungen
`spark-defaults`	Ändern Sie die Werte in der `spark-defaults.conf` -Spark-Datei.
`spark-env`	Ändert die Werte in der Spark-Umgebung.
`spark-hive-site`	Ändern Sie die Werte in der `hive-site.xml` -Spark-Datei.
`spark-log4j`	Ändern Sie die Werte in der `log4j2.properties` -Spark-Datei.
`emr-job-submitter`	Konfiguration für den Auftragsübermittler-Pod.

Speziell zur Verwendung mit CreateManagedEndpointAPIs:

Klassifizierungen	Beschreibungen
`jeg-config`	Ändern Sie die Werte in der Jupyter-Enterprise-Gateway-Datei `jupyter_enterprise_gateway_config.py` .
`jupyter-kernel-overrides`	Ändern Sie den Wert für das Kernel-Image in der Jupyter-Kernel-Spec-Datei.

Mithilfe von Konfigurationsklassifizierungen können Sie Anwendungen anpassen. Diese entsprechen häufig einer XML-Konfigurationsdatei für die Anwendung, z. B. `spark-hive-site.xml` Weitere Informationen finden Sie unter Konfigurieren von Anwendungen.

Bemerkenswerte Features

Die folgenden Features sind in der Version 6.12 von Amazon EMR in EKS enthalten.

- Java 17 – Mit Amazon EMR in EKS 6.12 und höher können Sie Spark mit Java-17-Laufzeit starten. Geben Sie dazu `emr-6.12.0-java17-latest` als Versionskennung ein. Wir empfehlen, dass

Sie Leistungstests validieren und ausführen, bevor Sie Ihre Produktionsworkloads von früheren Versionen des Java-Images auf das Java-17-Image verschieben.

emr-6.12.0-latest

Versionshinweise: `emr-6.12.0-latest` verweist derzeit auf `emr-6.12.0-20240321`.

Regionen: `emr-6.12.0-latest` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.12.0:latest`

emr-6.12.0-20240321

Versionshinweise: wurde am 11. März 2024 veröffentlicht. `6.12.0-20240321` Im Vergleich zur vorherigen Version wurde diese Version mit den kürzlich aktualisierten Amazon Linux-Paketen und kritischen Korrekturen aktualisiert.

Regionen: `emr-6.12.0-20240321` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.12.0:20240321`

emr-6.12.0-20230701

Versionshinweise: `6.12.0-20230701` wurde am 1. Juli 2023 veröffentlicht. Dies ist die erste veröffentlichte Version von Amazon EMR 6.12.0.

Regionen: `emr-6.12.0-20230701` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.12.0:20230701`

Versionen von Amazon EMR in EKS 6.11.0

Auf dieser Seite werden die neuen und aktualisierten Funktionen für Amazon EMR beschrieben, die spezifisch für die Bereitstellung von Amazon EMR in EKS sind. Einzelheiten zur Ausführung von

Amazon EMR auf Amazon EC2 und zur Amazon-EMR-Version 6.11.0 im Allgemeinen finden Sie unter Amazon EMR 6.11.0 im Amazon-EMR-Versionshandbuch.

Amazon EMR in EKS 6.11 Versionen

Die folgenden Amazon-EMR-6.11.0-Versionen sind für Amazon EMR in EKS verfügbar. Wählen Sie eine bestimmte EMR-6.11.0-xxxx-Version aus, um weitere Details wie das zugehörige Container-Image-Tag anzuzeigen.

- emr-6.11.0-latest
- emr-6.11.0-20230905
- emr-6.11.0-20230509

- emr-6.11.0- spark-rapids-latest
- emr-6.11.0-spark-rapids-20230509
- emr-6.11.0-java11-latest
- emr-6.11.0-java11-20230509
- notebook-spark/emr-6.11.0-latest
- notebook-spark/emr-6.11.0-20230509
- notebook-python/emr-6.11.0-latest
- notebook-python/emr-6.11.0-20230509

Versionshinweise

Versionshinweise für Amazon EMR in EKS 6.11.0

- Unterstützte Anwendungen - AWS SDK for Java 1.12.446, Apache Spark 3.3.2-amzn-0, Apache Hudi 0.13.0-amzn-0, Apache Iceberg 1.2.0-amzn-0, Delta 2.2.0, Apache Spark RAPIDS 23.02.0-amzn-0, Jupyter Enterprise Gateway 2.6.0
- Unterstützte Komponenten – `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg`, `spark-kubernetes`.
- Unterstützte Konfigurationsklassifizierungen

  StartJobRunZur CreateManagedEndpointVerwendung mit und APIs:

Klassifizierungen	Beschreibungen
`core-site`	Ändern Sie die Werte in der `core-site.xml` -Hadoop-Datei.
`emrfs-site`	Ändert die EMRFS-Einstellungen.
`spark-metrics`	Ändern Sie die Werte in der `metrics.properties` -Spark-Datei.
`spark-defaults`	Ändern Sie die Werte in der `spark-defaults.conf` -Spark-Datei.
`spark-env`	Ändert die Werte in der Spark-Umgebung.
`spark-hive-site`	Ändern Sie die Werte in der `hive-site.xml` -Spark-Datei.
`spark-log4j`	Ändern Sie die Werte in der `log4j.properties` -Spark-Datei.

Speziell zur Verwendung mit CreateManagedEndpointAPIs:

Klassifizierungen	Beschreibungen
`jeg-config`	Ändern Sie die Werte in der Jupyter-Enterprise-Gateway-Datei `jupyter_enterprise_gateway_config.py` .
`jupyter-kernel-overrides`	Ändern Sie den Wert für das Kernel-Image in der Jupyter-Kernel-Spec-Datei.

Mithilfe von Konfigurationsklassifizierungen können Sie Anwendungen anpassen. Diese entsprechen häufig einer XML-Konfigurationsdatei für die Anwendung, z. B. `spark-hive-site.xml` Weitere Informationen finden Sie unter Konfigurieren von Anwendungen.

Bemerkenswerte Features

Die folgenden Features sind in der Version 6.11 von Amazon EMR in EKS enthalten.

- Amazon EMR in EKS-Basis-Image in Amazon ECR Public Gallery – Wenn Sie die Funktion für benutzerdefinierte Images verwenden, bietet unser Basis-Image die wesentlichen Jars, Konfigurationen und Bibliotheken für die Interaktion mit Amazon EMR in EKS. Sie finden das Basis-Image jetzt in der Amazon ECR Public Gallery.
- Spark-Container-Log-Rotation – Amazon EMR in EKS 6.11 unterstützt die Spark-Container-Protokoll-Rotation. Sie können die Funktion `containerLogRotationConfiguration` innerhalb des `MonitoringConfiguration`-Betriebs der `StartJobRun`-API aktivieren. Sie können das `rotationSize` und konfigurieren`maxFilestoKeep`, um die Anzahl und Größe der Protokolldateien anzugeben, die Amazon EMR in EKS in den Spark-Treiber- und Ausführer-Pods speichern soll. Weitere Informationen finden Sie unter Verwenden der Spark-Container-Protokoll-Rotation.
- Volcano-Unterstützung in Spark-Operator und Spark-Submit – Amazon EMR in EKS 6.11 unterstützt die Ausführung von Spark-Aufträge mit Volcano als benutzerdefiniertem Kubernetes-Scheduler in Spark-Operator und Spark-Submit. Sie können Features wie Gruppenplanung, Warteschlangenverwaltung, Präemption und Fair-Share-Scheduling verwenden, um einen hohen Planungsdurchsatz und eine optimierte Kapazität zu erreichen. Weitere Informationen finden Sie unter Verwendung von Volcano als benutzerdefiniertem Scheduler für Apache Spark auf Amazon EMR in EKS.

emr-6.11.0-latest

Versionshinweise: `emr-6.11.0-latest` verweist derzeit auf `emr-20230905`.

Regionen: `emr-6.11.0-latest` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.11.0:latest`

emr-6.11.0-20230905

Versionshinweise: `6.11.0-20230905` wurde am 29. September 2023 veröffentlicht. Im Vergleich zur vorherigen Version wurde diese Version mit den kürzlich aktualisierten Amazon Linux-Paketen und kritischen Korrekturen aktualisiert.

Regionen: `emr-6.11.0-20230509` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.11.0:20230509`

emr-6.11.0-20230509

Versionshinweise: `6.11.0-20230509` wurde am 9. Mai 2023 veröffentlicht. Dies ist die erste veröffentlichte Version von Amazon EMR 6.11.0.

Regionen: `emr-6.11.0-20230509` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.11.0:20230509`

Versionen von Amazon EMR in EKS 6.10.0

Die folgenden Amazon-EMR-6.10.0-Versionen sind für Amazon EMR in EKS verfügbar. Wählen Sie eine bestimmte EMR-6.10.0-xxxx-Version aus, um weitere Details wie das zugehörige Container-Image-Tag anzuzeigen.

- emr-6.10.0-latest
- emr-6.10.0-20230905
- emr-6.10.0-20230624
- emr-6.10.0-20230421
- emr-6.10.0-20230403
- emr-6.10.0-20230220
- emr-6.10.0- spark-rapids-latest
- emr-6.10.0-spark-rapids-20230624
- emr-6.10.0-spark-rapids-20230220
- emr-6.10.0-java11-latest
- emr-6.10.0-java11-20230624
- emr-6.10.0-java11-20230220

- notebook-spark/emr-6.10.0-latest
- notebook-spark/emr-6.10.0-20230624
- notebook-spark/emr-6.10.0-20230220
- notebook-python/emr-6.10.0-latest
- notebook-python/emr-6.10.0-20230624
- notebook-python/emr-6.10.0-20230220

Versionshinweise für Amazon EMR 6.10.0

- Unterstützte Anwendungen - AWS SDK for Java 1.12.397, Spark 3.3.1-amzn-0, Hudi 0.12.2-amzn-0, Iceberg 1.1.0-amzn-0, Delta 2.2.0.
- Unterstützte Komponenten – `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg`, `spark-kubernetes`.
- Unterstützte Konfigurationsklassifizierungen:

  StartJobRunZur CreateManagedEndpointVerwendung mit und APIs:

Klassifizierungen	Beschreibungen
`core-site`	Ändern Sie die Werte in der `core-site.xml` -Hadoop-Datei.
`emrfs-site`	Ändert die EMRFS-Einstellungen.
`spark-metrics`	Ändern Sie die Werte in der `metrics.properties` -Spark-Datei.
`spark-defaults`	Ändern Sie die Werte in der `spark-defaults.conf` -Spark-Datei.
`spark-env`	Ändert die Werte in der Spark-Umgebung.
`spark-hive-site`	Ändern Sie die Werte in der `hive-site.xml` -Spark-Datei.
`spark-log4j`	Ändern Sie die Werte in der `log4j.properties` -Spark-Datei.

Speziell zur Verwendung mit CreateManagedEndpointAPIs:

Klassifizierungen	Beschreibungen
`jeg-config`	Ändern Sie die Werte in der Jupyter-Enterprise-Gateway-Datei `jupyter_enterprise_gateway_config.py` .
`jupyter-kernel-overrides`	Ändern Sie den Wert für das Kernel-Image in der Jupyter-Kernel-Spec-Datei.

Mithilfe von Konfigurationsklassifizierungen können Sie Anwendungen anpassen. Diese entsprechen häufig einer XML-Konfigurationsdatei für die Anwendung, z. B. `spark-hive-site.xml` Weitere Informationen finden Sie unter Konfigurieren von Anwendungen.

Bemerkenswerte Features

- Spark-Operator – Mit Amazon EMR in EKS 6.10.0 und höher können Sie den Kubernetes-Operator für Apache Spark oder den Spark-Operator verwenden, um Spark-Anwendungen mit der Amazon-EMR-Versions-Laufzeit auf Ihren eigenen Amazon-EKS-Clustern bereitzustellen und zu verwalten. Weitere Informationen finden Sie unter Spark-Aufträge mit dem Spark-Operator ausführen.
- Java 11 – Mit Amazon EMR in EKS 6.10 und höher können Sie Spark mit Java-11-Laufzeit starten. Geben Sie dazu `emr-6.10.0-java11-latest` als Versionskennung ein. Wir empfehlen, dass Sie Leistungstests validieren und ausführen, bevor Sie Ihre Produktionsworkloads vom Java-8-Image auf das Java-11-Image verschieben.
- Für die Amazon-Redshift-Integration für Apache Spark entfernt Amazon EMR in EKS 6.10.0 die Abhängigkeit von `minimal-json.jar` und fügt die erforderlichen `spark-redshift` zugehörigen JAR-Dateien automatisch zum Ausführer-Klassenpfad für Spark hinzu: `spark-redshift.jar`, `spark-avro.jar` und `RedshiftJDBC.jar`.

Änderungen

- Der für EMRFS S3 optimierte Committer ist jetzt standardmäßig für Parquet-, ORC- und textbasierte Formate (einschließlich CSV und JSON) aktiviert.

emr-6.10.0-latest

Versionshinweise: `emr-6.10.0-latest` verweist derzeit auf `emr-6.10.0-20230905`.

Regionen: `emr-6.10.0-latest` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.10.0:latest`

emr-6.10.0-20230905

Versionshinweise: `6.10.0-20230905` wurde am 29. September 2023 veröffentlicht. Im Vergleich zur vorherigen Version wurde diese Version mit kürzlich aktualisierten Amazon-Linux-Paketen und kritischen Korrekturen aktualisiert.

Regionen: `emr-6.10.0-20230905` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.10.0:20230905`

emr-6.10.0-20230624

Versionshinweise: `6.10.0-20230624` wurde am 7. Juli 2023 veröffentlicht. Im Vergleich zur vorherigen Version wurde diese Version mit kürzlich aktualisierten Amazon-Linux-Paketen und kritischen Korrekturen aktualisiert.

Regionen: `emr-6.10.0-20230624` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.10.0:20230624`

emr-6.10.0-20230421

Versionshinweise: `6.10.0-20230421` wurde am 28. April 2023 veröffentlicht. Im Vergleich zur vorherigen Version wurde diese Version mit kürzlich aktualisierten Amazon-Linux-Paketen und kritischen Korrekturen aktualisiert.

Regionen: `emr-6.10.0-20230421` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.10.0:20230421`

emr-6.10.0-20230403

Versionshinweise: `6.10.0-20230403` wurde am 12. April 2023 veröffentlicht. Im Vergleich zur vorherigen Version wurde diese Version mit kürzlich aktualisierten Amazon-Linux-Paketen und kritischen Korrekturen aktualisiert.

Regionen: `emr-6.10.0-20230403` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.10.0:20230403`

emr-6.10.0-20230220

Versionshinweise: `emr-6.10.0-20230220` wurde am 20. Februar 2023 veröffentlicht. Dies ist die erste veröffentlichte Version von Amazon EMR 6.10.0.

Regionen: `emr-6.10.0-20230220` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.10.0:20230220`

Versionen von Amazon EMR in EKS 6.9.0

Die folgenden Amazon-EMR-6.9.0-Versionen sind für Amazon EMR in EKS verfügbar. Wählen Sie eine bestimmte EMR-6.9.0-xxxx-Version aus, um weitere Details wie das zugehörige Container-Image-Tag anzuzeigen.

- emr-6.9.0-latest
- ???
- emr-6.9.0-20230624
- emr-6.9.0-20221108
- emr-6.9.0- spark-rapids-latest

- emr-6.9.0-spark-rapids-20230624
- emr-6.9.0-spark-rapids-20221108
- notebook-spark/emr-6.9.0-latest
- notebook-spark/emr-6.9.0-20230624
- notebook-spark/emr-6.9.0-20221108
- notebook-python/emr-6.9.0-latest
- notebook-python/emr-6.9.0-20230624
- notebook-python/emr-6.9.0-20221108

Versionshinweise für Amazon EMR 6.9.0

- Unterstützte Anwendungen - AWS SDK for Java 1.12.331, Spark 3.3.0-amzn-1, Hudi 0.12.1-amzn-0, Iceberg 0.14.1-amzn-0, Delta 2.1.0.
- Unterstützte Komponenten – `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg`, `spark-kubernetes`.
- Unterstützte Konfigurationsklassifizierungen:

  StartJobRunZur Verwendung CreateManagedEndpointmit und APIs:

Klassifizierungen	Beschreibungen
`core-site`	Ändern Sie die Werte in der core-site.xml-Datei in Hadoop.
`emrfs-site`	Ändert die EMRFS-Einstellungen.
`spark-metrics`	Ändert die Werte in der metrics.properties-Datei in Spark.
`spark-defaults`	Ändert die Werte in der spark-defaults.conf-Datei in Spark.
`spark-env`	Ändert die Werte in der Spark-Umgebung.
`spark-hive-site`	Ändern Sie die Werte in der hive-site.xml-Datei in Spark.

Klassifizierungen	Beschreibungen
`spark-log4j`	Ändert die Werte in der log4j.properties-Datei in Spark.

Speziell zur Verwendung mit CreateManagedEndpointAPIs:

Klassifizierungen	Beschreibungen
`jeg-config`	Ändern Sie die Werte in der Jupyter-Enterprise-Gateway-Datei `jupyter_enterprise_gateway_config.py` .
`jupyter-kernel-overrides`	Ändern Sie den Wert für das Kernel-Image in der Jupyter-Kernel-Spec-Datei.

Mithilfe von Konfigurationsklassifizierungen können Sie Anwendungen anpassen. Diese entsprechen häufig einer XML-Konfigurationsdatei für die Anwendung, z. B. `spark-hive-site.xml` Weitere Informationen finden Sie unter Konfigurieren von Anwendungen.

Bemerkenswerte Features

- Nvidia RAPIDS Accelerator für Apache Spark – Amazon EMR in EKS zur Beschleunigung von Spark mithilfe von EC2-GPU-Instance-Typen (Graphics Processing Unit). Um das Spark-Image mit RAPIDS Accelerator zu verwenden, geben Sie das Release-Label als emr-6.9.0- an. spark-rapids-latest Besuchen Sie die Dokumentationsseite, um mehr zu erfahren.
- Spark-Redshift-Konnektor Die Amazon-Redshift-Integration für Apache Spark ist in den Amazon-EMR-Versionen 6.9.0 und höher enthalten. Die native Integration war bisher ein Open-Source-Tool und ist ein Spark-Konnektor, mit dem Sie Apache-Spark-Anwendungen erstellen können, die Daten in Amazon Redshift und Amazon Redshift Serverless lesen und in diese schreiben. Weitere Informationen finden Sie unter Verwenden der Amazon-Redshift-Integration für Apache Spark auf Amazon EMR in EKS.
- Delta Lake – Delta Lake ist ein Open-Source-Speicherformat, das den Aufbau von Data Lakes mit Transaktionskonsistenz, konsistenter Definition von Datensätzen, Änderungen der

Schemaentwicklung und Unterstützung von Datenmutationen ermöglicht. Weitere Informationen finden Sie unter Verwenden von Delta Lake.

- PySpark Parameter ändern — Interaktive Endpunkte unterstützen jetzt das Ändern von Spark-Parametern, die mit PySpark Sitzungen im EMR Studio Jupyter Notebook verknüpft sind. Weitere Informationen finden Sie unter PySpark Sitzungsparameter ändern.

Gelöste Probleme

- Wenn Sie den DynamoDB-Konnektor mit Spark auf den Amazon-EMR-Versionen 6.6.0, 6.7.0 und 6.8.0 verwenden, geben alle Lesevorgänge aus Ihrer Tabelle ein leeres Ergebnis zurück, obwohl der Eingabe-Split auf nicht leere Daten verweist. Amazon EMR Version 6.9.0 behebt dieses Problem.
- Amazon EMR in EKS 6.8.0 füllt den Build-Hash fälschlicherweise in die Metadaten von Parquet-Dateien ein, die mit Apache Spark generiert wurden. Dieses Problem kann dazu führen, dass Tools, die die Metadaten-Versionszeichenfolge aus Parquet-Dateien analysieren, die von Amazon EMR in EKS 6.8.0 generiert wurden, fehlschlagen.

Bekanntes Problem

- 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-6.9.0-latest

Versionshinweise: `emr-6.9.0-latest` verweist derzeit auf `emr-6.9.0-20230905`.

Regionen: `emr-6.9.0-latest` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.9.0:latest`

emr-6.9.0-20230905

Versionshinweise: `emr-6.9.0-20230905`. Im Vergleich zur vorherigen Version wurde diese Version mit den kürzlich aktualisierten Amazon Linux-Paketen und kritischen Korrekturen aktualisiert.

Regionen: `emr-6.9.0-20230905` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.9.0:20230905`

emr-6.9.0-20230624

Versionshinweise: `emr-6.9.0-20230624` wurde am 7. Juli 2023 veröffentlicht.

Regionen: `emr-6.9.0-20230624` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.9.0:20230624`

emr-6.9.0-20221108

Versionshinweise: `emr-6.9.0-20221108` wurde am 08. Dezember 2022 veröffentlicht. Dies ist die erste veröffentlichte Version von Amazon EMR 6.9.0.

Regionen: `emr-6.9.0-20221108` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.9.0:20221108`

Versionen von Amazon EMR in EKS 6.8.0

Die folgenden Amazon-EMR-6.8.0-Versionen sind für Amazon EMR in EKS verfügbar. Wählen Sie eine bestimmte emr-6.8.0-XXXX-Version aus, um weitere Details wie das zugehörige Container-Image-Tag anzuzeigen.

- emr-6.8.0-latest
- emr-6.8.0-20230905
- emr-6.8.0-20230624

- emr-6.8.0-20221219
- emr-6.8.0-20220802

Versionshinweise für Amazon EMR 6.8.0

- Unterstützte Anwendungen - AWS SDK for Java 1.12.170, Spark 3.3.0-amzn-0, Hudi 0.11.1-amzn-0, Iceberg 0.14.0-amzn-0.
- Unterstützte Komponenten – `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg`, `spark-kubernetes`.
- Unterstützte Konfigurationsklassifizierungen:

Klassifizierungen	Beschreibungen
`core-site`	Ändern Sie die Werte in der core-site.xml-Datei in Hadoop.
`emrfs-site`	Ändert die EMRFS-Einstellungen.
`spark-metrics`	Ändert die Werte in der metrics.properties-Datei in Spark.
`spark-defaults`	Ändert die Werte in der spark-defaults.conf-Datei in Spark.
`spark-env`	Ändert die Werte in der Spark-Umgebung.
`spark-hive-site`	Ändern Sie die Werte in der hive-site.xml-Datei in Spark.
`spark-log4j`	Ändert die Werte in der log4j.properties-Datei in Spark.

  Mithilfe von Konfigurationsklassifizierungen können Sie Anwendungen anpassen. Diese entsprechen häufig einer XML-Konfigurationsdatei für die Anwendung, z. B. `spark-hive-site.xml` Weitere Informationen finden Sie unter Konfigurieren von Anwendungen.

Bemerkenswerte Features

- Spark 3.3.0 – Amazon EMR in EKS 6.8 enthält Spark 3.3.0, das die Verwendung separater Node-Selector-Labels für Spark-Treiber-Ausführer-Pods unterstützt. Mit diesen neuen Bezeichnungen können Sie die Knotentypen für die Treiber- und Executor-Pods separat in der API definieren, ohne Pod-Vorlagen verwenden zu müssen. StartJobRun
  - Eigenschaft zur Treiberknotenauswahl: spark.kubernetes.driver.node.selector. [LabelKey]
  - Eigenschaft zur Auswahl des Ausführer-Knotens: spark.kubernetes.executor.node.selector. [LabelKey]
- Verbesserte Meldung bei Aufgabenfehlern – In dieser Version werden die Konfiguration `spark.stage.extraDetailsOnFetchFailures.enabled` und `spark.stage.extraDetailsOnFetchFailures.maxFailuresToInclude` die Nachverfolgung von Fehlern bei Aufgaben eingeführt, die auf Benutzercode zurückzuführen sind. Diese Informationen werden verwendet, um die Fehlermeldung zu verbessern, die im Treiberprotokoll angezeigt wird, wenn eine Phase aufgrund eines Fehlers beim Zufallsabruf abgebrochen wird.

Name der Eigenschaft	Standardwert	Bedeutung	Seit Version
`spark.stage.extraDetailsOnFetchFailures.enabled`	false	Wenn diese Eigenschaft auf `true` gesetzt ist, wird sie verwendet, um die Meldung eines Aufgabenfehlers zu verbessern, die im Treiberprotokoll angezeigt wird, wenn eine Phase aufgrund von Shuffle Fetch Failures abgebrochen wird. Standardmäßig werden die letzten fünf durch Benutzercode verursachten Taskfehler protokolliert, und	emr-6.8

Name der Eigenschaft	Standardwert	Bedeutung	Seit Version
		die Fehlermeldung wird an die Treiberprotokolle angehängt. Informationen zur Erhöhung der Anzahl von Aufgabenfehlern, bei denen Benutzerausnahmen nachzuverfolgen sind, finden Sie in der Konfiguration `spark.stage.extraDetailsOnFetchFailures.maxFailuresToInclude` .	

Name der Eigenschaft	Standardwert	Bedeutung	Seit Version
`spark.stage.extraDetailsOnFetchFailures.maxFailuresToInclude`	5	Anzahl der Aufgabenfehler, die pro Phase und Versuch nachverfolgt werden müssen. Diese Eigenschaft wird verwendet, um die Meldung eines Aufgabenfehlers um Benutzerausnahmen zu erweitern, die im Treiberprotokoll angezeigt werden, wenn eine Phase aufgrund von Shuffle Fetch Failures abgebrochen wird. Diese Eigenschaft funktioniert nur bei Config spark.stage.extraDetailsOnFetchFailures.enabled ist auf true gesetzt.	emr-6.8

Weiter Informationen finden Sie unter Konfiguration in der Apache-Spark-Dokumentation.

Bekanntes Problem

- Amazon EMR in EKS 6.8.0 füllt den Build-Hash fälschlicherweise in die Metadaten von Parquet-Dateien ein, die mit Apache Spark generiert wurden. Dieses Problem kann dazu führen, dass Tools, die die Metadaten-Versionszeichenfolge aus Parquet-Dateien analysieren, die von Amazon EMR in EKS 6.8.0 generiert wurden, fehlschlagen. Kunden, die die Versionszeichenfolge anhand

der Parquet-Metadaten analysieren und vom Build-Hash abhängig sind, sollten zu einer anderen Amazon-EMR-Version wechseln und die Datei neu schreiben.

Gelöste Probleme

- Kernelfähigkeit für PySpark-Kernel unterbrechen – In Bearbeitung befindliche interaktive Workloads, die durch die Ausführung von Zellen in einem Notebook ausgelöst werden, können mithilfe dieser Funktion `Interrupt Kernel` gestoppt werden. Es wurde ein Fix eingeführt, sodass diese Funktionalität für PySpark-Kernel funktioniert. Dies ist auch als Open Source unter Changes for handling interrupts for PySpark Kubernetes Kernel #1115 verfügbar.

emr-6.8.0-latest

Versionshinweise: `emr-6.8.0-latest` verweist derzeit auf `emr-6.8.0-20230624`.

Regionen: `emr-6.8.0-latest` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.8.0:latest`

emr-6.8.0-20230905

Versionshinweise: `emr-6.8.0-20230905` wurde am 29. September 2023 veröffentlicht. Im Vergleich zur vorherigen Version wurde diese Version mit den kürzlich aktualisierten Amazon Linux-Paketen und kritischen Korrekturen aktualisiert.

Regionen: `emr-6.8.0-20230905` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.8.0:20230905`

emr-6.8.0-20230624

Versionshinweise: `emr-6.8.0-20230624` wurde am 7. Juli 2023 veröffentlicht. Im Vergleich zur vorherigen Version wurde diese Version mit den kürzlich aktualisierten Amazon Linux-Paketen und kritischen Korrekturen aktualisiert.

Regionen: `emr-6.8.0-20230624` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.8.0:20230624`

emr-6.8.0-20221219

Versionshinweise: `emr-6.8.0-20221219` wurde am 19. Januar 2023 veröffentlicht. Im Vergleich zur vorherigen Version wurde diese Version mit den kürzlich aktualisierten Amazon Linux-Paketen und kritischen Korrekturen aktualisiert.

Regionen: `emr-6.8.0-20221219` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.8.0:20221219`

emr-6.8.0-20220802

Versionshinweise: `emr-6.8.0-20220802` wurde am 27. September 2022 veröffentlicht. Dies ist die erste veröffentlichte Version von Amazon EMR 6.8.0.

Regionen: `emr-6.8.0-20220802` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.8.0:20220802`

Versionen von Amazon EMR in EKS 6.7.0

Die folgenden Amazon-EMR-6.7.0-Versionen sind für Amazon EMR in EKS verfügbar. Wählen Sie eine bestimmte emr-6.7.0-XXXX-Version aus, um weitere Details wie das zugehörige Container-Image-Tag anzuzeigen.

- emr-6.7.0-latest
- emr-6.7.0-20240321
- emr-6.7.0-20230624
- emr-6.7.0-20221219

- emr-6.7.0-20220630

Versionshinweise für Amazon EMR 6.7.0

- Unterstützte Anwendungen – Spark 3.2.1-amzn-0, Jupyter Enterprise Gateway 2.6, Hudi 0.11-amzn-0, Iceberg 0.13.1.
- Unterstützte Komponenten – `aws-hm-client` (Glue-Konnektor), `aws-sagemaker-spark-sdk`, `emr-s3-select`, `emrfs`, `emr-ddb`, `hudi-spark`.
- Mit dem Upgrade auf JEG 2.6 ist die Kernelverwaltung jetzt asynchron, was bedeutet, dass JEG keine Transaktionen blockiert, wenn ein Kernelstart im Gange ist. Dies verbessert die Benutzererfahrung erheblich, da Folgendes bereitgestellt wird:
  - Fähigkeit, Befehle in aktuell laufenden Notebooks auszuführen, wenn andere Kernelstarts im Gange sind
  - Fähigkeit, mehrere Kernel gleichzeitig zu starten, ohne dass sich dies auf bereits laufende Kernel auswirkt
- Unterstützte Konfigurationsklassifizierungen:

Klassifizierungen	Beschreibungen
`core-site`	Ändern Sie die Werte in der `core-site.xml` -Hadoop-Datei.
`emrfs-site`	Ändert die EMRFS-Einstellungen.
`spark-metrics`	Ändern Sie die Werte in der `metrics.properties` -Spark-Datei.
`spark-defaults`	Ändern Sie die Werte in der `spark-defaults.conf` -Spark-Datei.
`spark-env`	Ändert die Werte in der Spark-Umgebung.
`spark-hive-site`	Ändern Sie die Werte in der `hive-site.xml` -Spark-Datei.
`spark-log4j`	Ändern Sie die Werte in der `log4j.properties` -Spark-Datei.

Mithilfe von Konfigurationsklassifizierungen können Sie Anwendungen anpassen. Diese entsprechen häufig einer XML-Konfigurationsdatei für die Anwendung, z. B. `spark-hive-site.xml` Weitere Informationen finden Sie unter Configuring Applications.

Gelöste Probleme

- Amazon EMR in EKS 6.7 behebt ein Problem in 6.6 bei der Verwendung der Pod-Vorlagenfunktion von Apache Spark mit interaktiven Endpunkten. Das Problem trat in den Amazon EMR in EKS-Versionen 6.4, 6.5 und 6.6 auf. Sie können jetzt Pod-Vorlagen verwenden, um zu definieren, wie Ihre Spark-Treiber- und Ausführer-Pods starten, wenn Sie interaktive Endpunkte zur Ausführung interaktiver Analysen verwenden.
- In früheren Versionen von Amazon EMR in EKS blockierte Jupyter Enterprise-Gateway-Transaktionen, wenn der Kernel gestartet wurde, was die Ausführung der aktuell laufenden Notebook-Sitzungen behinderte. Sie können jetzt Befehle in aktuell laufenden Notebooks ausführen, wenn andere Kernelstarts im Gange sind. Sie können auch mehrere Kernel gleichzeitig starten, ohne das Risiko einzugehen, die Konnektivität zu Kerneln zu verlieren, die bereits laufen.

emr-6.7.0-latest

Versionshinweise: `emr-6.7.0-latest` verweist derzeit auf `emr-6.7.0-20240321`.

Regionen: `emr-6.7.0-latest` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.7.0:latest`

emr-6.7.0-20240321

Versionshinweise: `emr-6.7.0-20240321` wurde am 11. März 2024 veröffentlicht. Im Vergleich zur vorherigen Version wurde diese Version mit den kürzlich aktualisierten Amazon Linux-Paketen und kritischen Korrekturen aktualisiert.

Regionen: `emr-6.7.0-20240321` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.7.0:20240321`

emr-6.7.0-20230624

Versionshinweise: `emr-6.7.0-20230624` wurde am 7. Juli 2023 veröffentlicht. Im Vergleich zur Vorgängerversion wurde diese Version mit den kürzlich aktualisierten Amazon Linux-Paketen und kritischen Korrekturen aktualisiert.

Regionen: `emr-6.7.0-20230624` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.7.0:20230624`

emr-6.7.0-20221219

Versionshinweise: `emr-6.7.0-20221219` wurde am 19. Januar 2023 veröffentlicht. Im Vergleich zur Vorgängerversion wurde diese Version mit den kürzlich aktualisierten Amazon Linux-Paketen und kritischen Korrekturen aktualisiert.

Regionen: `emr-6.7.0-20221219` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.7.0:20221219`

emr-6.7.0-20220630

Versionshinweise: `emr-6.7.0-20220630` wurde am 12. Juli 2022 veröffentlicht. Dies ist die erste veröffentlichte Version von Amazon EMR 6.7.0.

Regionen: `emr-6.7.0-20220630` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.7.0:20220630`

Versionen von Amazon EMR in EKS 6.6.0

Die folgenden Amazon-EMR-6.6.0-Versionen sind für Amazon EMR in EKS verfügbar. Wählen Sie eine bestimmte emr-6.6.0-XXXX-Version aus, um weitere Details wie das zugehörige Container-Image-Tag anzuzeigen.

- emr-6.6.0-latest
- emr-6.6.0-20240321
- emr-6.6.0-20230624
- emr-6.6.0-20221219
- emr-6.6.0-20220411

Versionshinweise für Amazon EMR 6.6.0

- Unterstützte Anwendungen – Spark 3.2.0-amzn-0, Jupyter Enterprise Gateway (Endpunkte, öffentliche Vorversion), Hudi 0.10.1-amzn-0, Iceberg 0.13.1.
- Unterstützte Komponenten – `aws-hm-client` (Glue-Konnektor), `aws-sagemaker-spark-sdk`, `emr-s3-select`, `emrfs`, `emr-ddb`, `hudi-spark`.
- Unterstützte Konfigurationsklassifizierungen:

Klassifizierungen	Beschreibungen
`core-site`	Ändern Sie die Werte in der core-site.xml-Datei in Hadoop.
`emrfs-site`	Ändert die EMRFS-Einstellungen.
`spark-metrics`	Ändert die Werte in der metrics.properties-Datei in Spark.
`spark-defaults`	Ändert die Werte in der spark-defaults.conf-Datei in Spark.
`spark-env`	Ändert die Werte in der Spark-Umgebung.
`spark-hive-site`	Ändern Sie die Werte in der hive-site.xml-Datei in Spark.

Klassifizierungen	Beschreibungen
`spark-log4j`	Ändert die Werte in der log4j.properties-Datei in Spark.

Mithilfe von Konfigurationsklassifizierungen können Sie Anwendungen anpassen. Diese entsprechen häufig einer XML-Konfigurationsdatei für die Anwendung, z. B. spark-hive-site .xml. Weitere Informationen finden Sie unter Configuring Applications.

Bekanntes Problem

- Die Spark-Pod-Vorlagenfunktion mit interaktiven Endpunkten funktioniert in Amazon EMR in EKS-Versionen 6.4, 6.5 und 6.6 nicht.

Gelöste Probleme

- Interaktive Endpunktprotokolle werden auf Cloudwatch und S3 hochgeladen.

emr-6.6.0-latest

Versionshinweise: `emr-6.6.0-latest` verweist derzeit auf `emr-6.6.0-20240321`.

Regionen: `emr-6.6.0-latest` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.6.0:latest`

emr-6.6.0-20240321

Versionshinweise: `emr-6.6.0-20240321` wurde am 11. März 2024 veröffentlicht. Im Vergleich zur vorherigen Version wurde diese Version mit den kürzlich aktualisierten Amazon Linux-Paketen und kritischen Korrekturen aktualisiert.

Regionen: `emr-6.6.0-20240321` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.6.0:20240321`

emr-6.6.0-20230624

Versionshinweise: `emr-6.6.0-20230624` wurde am 27. Januar 2023 veröffentlicht. Im Vergleich zur Vorgängerversion wurde diese Version mit den kürzlich aktualisierten Amazon Linux-Paketen und kritischen Korrekturen aktualisiert.

Regionen: `emr-6.6.0-20230624` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.6.0:20230624`

emr-6.6.0-20221219

Versionshinweise: `emr-6.6.0-20221219` wurde am 27. Januar 2023 veröffentlicht. Im Vergleich zur Vorgängerversion wurde diese Version mit den kürzlich aktualisierten Amazon Linux-Paketen und kritischen Korrekturen aktualisiert.

Regionen: `emr-6.6.0-20221219` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.6.0:20221219`

emr-6.6.0-20220411

Versionshinweise: `emr-6.6.0-20220411` wurde am 20. Mai 2022 veröffentlicht. Dies ist die erste veröffentlichte Version von Amazon EMR 6.6.0.

Regionen: `emr-6.6.0-20220411` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.6.0:20220411`

Versionen von Amazon EMR in EKS 6.5.0

Die folgenden Amazon EMR 6.5.0 Versionen sind für Amazon EMR in EKS verfügbar. Wählen Sie eine bestimmte emr-6.5.0-XXXX-Version aus, um weitere Details wie das zugehörige Container-Image-Tag anzuzeigen.

- emr-6.5.0-latest
- emr-6.5.0-20240321
- emr-6.5.0-20221219
- emr-6.5.0-20220802
- emr-6.5.0-20211119

Versionshinweise für Amazon EMR 6.5.0

- Unterstützte Anwendungen – Spark 3.1.2-amzn-1, Jupyter Enterprise Gateway (Endpunkte, öffentliche Vorschauversion).
- Unterstützte Komponenten – `aws-hm-client` (Glue-Konnektor), `aws-sagemaker-spark-sdk`, `emr-s3-select`, `emrfs`, `emr-ddb`, `hudi-spark`.
- Unterstützte Konfigurationsklassifizierungen:

Klassifizierungen	Beschreibungen
`core-site`	Ändern Sie die Werte in der core-site.xml-Datei in Hadoop.
`emrfs-site`	Ändert die EMRFS-Einstellungen.
`spark-metrics`	Ändert die Werte in der metrics.properties-Datei in Spark.
`spark-defaults`	Ändert die Werte in der spark-defaults.conf-Datei in Spark.
`spark-env`	Ändert die Werte in der Spark-Umgebung.
`spark-hive-site`	Ändern Sie die Werte in der hive-site.xml-Datei in Spark.

Klassifizierungen	Beschreibungen
`spark-log4j`	Ändert die Werte in der log4j.properties-Datei in Spark.

Mithilfe von Konfigurationsklassifizierungen können Sie Anwendungen anpassen. Diese entsprechen häufig einer XML-Konfigurationsdatei für die Anwendung, z. B. spark-hive-site .xml. Weitere Informationen finden Sie unter Configuring Applications.

Bekanntes Problem

- Die Spark-Pod-Vorlagenfunktion mit interaktiven Endpunkten funktioniert in den Amazon EMR in EKS-Versionen 6.4 und 6.5 nicht.

emr-6.5.0-latest

Versionshinweise: `emr-6.5.0-latest` verweist derzeit auf `emr-6.5.0-20240321`.

Regionen: `emr-6.5.0-latest` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.5.0:latest`

emr-6.5.0-20240321

Versionshinweise: `emr-6.5.0-20240321` wurde am 11. März 2024 veröffentlicht. Im Vergleich zur Vorgängerversion wurde diese Version mit den kürzlich aktualisierten Amazon Linux-Paketen und kritischen Korrekturen aktualisiert.

Regionen: `emr-6.5.0-20240321` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.5.0:20240321`

emr-6.5.0-20221219

Versionshinweise: `emr-6.5.0-20221219` wurde am 19. Januar 2023 veröffentlicht. Im Vergleich zur Vorgängerversion wurde diese Version mit den kürzlich aktualisierten Amazon Linux-Paketen und kritischen Korrekturen aktualisiert.

Regionen: `emr-6.5.0-20221219` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.5.0:20221219`

emr-6.5.0-20220802

Versionshinweise: `emr-6.5.0-20220802` wurde am 24. August 2022 veröffentlicht. Im Vergleich zur Vorgängerversion wurde diese Version mit den kürzlich aktualisierten Amazon Linux-Paketen aktualisiert.

Regionen: `emr-6.5.0-20220802` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.5.0:20220802`

emr-6.5.0-20211119

Versionshinweise: `emr-6.5.0-20211119` wurde am 20. Januar 2022 veröffentlicht. Dies ist die erste veröffentlichte Version von Amazon EMR 6.5.0.

Regionen: `emr-6.5.0-20211119` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.5.0:20211119`

Versionen von Amazon EMR in EKS 6.4.0

Die folgenden Amazon-EMR-6.4.0-Versionen sind für Amazon EMR in EKS verfügbar. Wählen Sie eine bestimmte emr-6.4.0-XXXX-Version aus, um weitere Details wie das zugehörige Container-Image-Tag anzuzeigen.

- emr-6.4.0-latest
- emr-6.4.0-20240321
- emr-6.4.0-20221219
- emr-6.4.0-20210830

Versionshinweise für Amazon EMR 6.4.0

- Unterstützte Anwendungen – Spark 3.1.2-amzn-0, Jupyter Enterprise Gateway (Endpunkte, öffentliche Vorschauversion).
- Unterstützte Komponenten – `aws-hm-client` (Glue-Konnektor), `aws-sagemaker-spark-sdk`, `emr-s3-select`, `emrfs`, `emr-ddb`, `hudi-spark`.
- Unterstützte Konfigurationsklassifizierungen:

Klassifizierungen	Beschreibungen
`core-site`	Ändern Sie die Werte in der core-site.xml-Datei in Hadoop.
`emrfs-site`	Ändert die EMRFS-Einstellungen.
`spark-metrics`	Ändert die Werte in der metrics.properties-Datei in Spark.
`spark-defaults`	Ändert die Werte in der spark-defaults.conf-Datei in Spark.
`spark-env`	Ändert die Werte in der Spark-Umgebung.
`spark-hive-site`	Ändern Sie die Werte in der hive-site.xml-Datei in Spark.
`spark-log4j`	Ändert die Werte in der log4j.properties-Datei in Spark.

Mithilfe von Konfigurationsklassifizierungen können Sie Anwendungen anpassen. Diese entsprechen häufig einer XML-Konfigurationsdatei für die Anwendung, z. B. spark-hive-site .xml. Weitere Informationen finden Sie unter Configuring Applications.

Bekanntes Problem

- Die Spark-Pod-Vorlagenfunktion mit interaktiven Endpunkten funktioniert in Amazon EMR in EKS-Version 6.4 nicht.

emr-6.4.0-latest

Versionshinweise: `emr-6.4.0-latest` verweist derzeit auf `emr-6.4.0-20240321`.

Regionen: `emr-6.4.0-latest` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.4.0:latest`

emr-6.4.0-20240321

Versionshinweise: `emr-6.4.0-20240321` wurde am 11. März 2024 veröffentlicht. Im Vergleich zur vorherigen Version wurde diese Version mit den kürzlich aktualisierten Amazon Linux-Paketen und kritischen Korrekturen aktualisiert.

Regionen: `emr-6.4.0-20240321` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.4.0:20240321`

emr-6.4.0-20221219

Versionshinweise: `emr-6.4.0-20221219` wurde am 27. Januar 2023 veröffentlicht. Im Vergleich zur Vorgängerversion wurde diese Version mit den kürzlich hinzugefügten Amazon Linux-Paketen aktualisiert.

Regionen: `emr-6.4.0-20221219` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.4.0:20221219`

emr-6.4.0-20210830

Versionshinweise: `emr-6.4.0-20210830` wurde am 9. Dezember 2021 veröffentlicht. Dies ist die erste veröffentlichte Version von Amazon EMR 6.4.0.

Regionen: `emr-6.4.0-20210830` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.4.0:20210830`

Versionen von Amazon EMR in EKS 6.3.0

Die folgenden Amazon-EMR-6.3.0-Versionen sind für Amazon EMR in EKS verfügbar. Wählen Sie eine bestimmte emr-6.3.0-XXXX-Version aus, um weitere Details wie das zugehörige Container-Image-Tag anzuzeigen.

- emr-6.3.0-latest
- emr-6.3.0-20240321
- emr-6.3.0-20220802
- emr-6.3.0-20211008
- emr-6.3.0-20210802
- emr-6.3.0-20210429

Versionshinweise für Amazon EMR 6.3.0

- Neue Feature – Ab Amazon EMR 6.3.0 in der 6.x-Versions-Serie unterstützt Amazon EMR in EKS das Pod-Vorlagenfeature von Spark. Sie können auch das Spark-Feature zur Rotation des Ereignisprotokolls für Amazon EMR in EKS aktivieren. Weitere Informationen finden Sie unter Verwenden von Pod-Vorlagen und Verwenden der Rotation des Spark-Ereignisprotokolls.
- Unterstützte Anwendungen – Spark 3.1.1-amzn-0, Jupyter Enterprise Gateway (Endpunkte, öffentliche Vorschauversion).

- Unterstützte Komponenten – `aws-hm-client` (Glue-Konnektor), `aws-sagemaker-spark-sdk`, `emr-s3-select`, `emrfs`, `emr-ddb`, `hudi-spark`.
- Unterstützte Konfigurationsklassifizierungen:

Klassifizierungen	Beschreibungen
`core-site`	Ändern Sie die Werte in der core-site.xml-Datei in Hadoop.
`emrfs-site`	Ändert die EMRFS-Einstellungen.
`spark-metrics`	Ändert die Werte in der metrics.properties-Datei in Spark.
`spark-defaults`	Ändert die Werte in der spark-defaults.conf-Datei in Spark.
`spark-env`	Ändert die Werte in der Spark-Umgebung.
`spark-hive-site`	Ändern Sie die Werte in der hive-site.xml-Datei in Spark.
`spark-log4j`	Ändert die Werte in der log4j.properties-Datei in Spark.

Mithilfe von Konfigurationsklassifizierungen können Sie Anwendungen anpassen. Diese entsprechen häufig einer XML-Konfigurationsdatei für die Anwendung, z. B. spark-hive-site .xml. Weitere Informationen finden Sie unter Configuring Applications.

emr-6.3.0-latest

Versionshinweise: `emr-6.3.0-latest` verweist derzeit auf `emr-6.3.0-20240321`.

Regionen: `emr-6.3.0-latest` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.3.0:latest`

emr-6.3.0-20240321

Versionshinweise: `emr-6.3.0-20240321` wurde am 11. März 2024 veröffentlicht. Im Vergleich zur vorherigen Version wurde diese Version mit den kürzlich aktualisierten Amazon Linux-Paketen und kritischen Korrekturen aktualisiert.

Regionen: `emr-6.3.0-20240321` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.3.0:20240321`

emr-6.3.0-20220802

Versionshinweise: `emr-6.3.0-20220802` wurde am 27. September 2022 veröffentlicht. Im Vergleich zur Vorgängerversion wurde diese Version mit den kürzlich aktualisierten Amazon Linux-Paketen aktualisiert.

Regionen: `emr-6.3.0-20220802` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.3.0:20220802`

emr-6.3.0-20211008

Versionshinweise: `emr-6.3.0-20211008` wurde am 9. Dezember 2021 veröffentlicht. Im Vergleich zur Vorgängerversion enthält diese Version Problembehebungen und Sicherheitsupdates.

Regionen: `emr-6.3.0-20211008` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.3.0:20211008`

emr-6.3.0-20210802

Versionshinweise: `emr-6.3.0-20210802` wurde am 2. August 2021 veröffentlicht. Im Vergleich zur Vorgängerversion enthält diese Version Problembehebungen und Sicherheitsupdates.

Regionen: `emr-6.3.0-20210802` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.3.0:20210802`

emr-6.3.0-20210429

Versionshinweise: `emr-6.3.0-20210429` wurde am 29. April 2021 veröffentlicht. Dies ist die erste veröffentlichte Version von Amazon EMR 6.3.0.

Regionen: `emr-6.3.0-20210429` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.3.0:20210429`

Versionen von Amazon EMR in EKS 6.2.0

Die folgenden Amazon-EMR-6.2.0-Versionen sind für Amazon EMR in EKS verfügbar. Wählen Sie eine bestimmte emr-6.2.0-XXXX-Version aus, um weitere Details wie das zugehörige Container-Image-Tag anzuzeigen.

- emr-6.2.0-latest
- emr-6.2.0-20240321
- emr-6.2.0-20220802
- emr-6.2.0-20211008
- emr-6.2.0-20210802
- emr-6.2.0-20210615
- emr-6.2.0-20210129
- emr-6.2.0-20201218
- emr-6.2.0-20201201

Versionshinweise für Amazon EMR 6.2.0

- Unterstützte Anwendungen – Spark 3.0.1-amzn-0, Jupyter Enterprise Gateway (Endpunkte, öffentliche Vorschauversion).

- Unterstützte Komponenten – `aws-hm-client` (Glue-Konnektor), `aws-sagemaker-spark-sdk`, `emr-s3-select`, `emrfs`, `emr-ddb`, `hudi-spark`.
- Unterstützte Konfigurationsklassifizierungen:

Klassifizierungen	Beschreibungen
`core-site`	Ändern Sie die Werte in der core-site.xml-Datei in Hadoop.
`emrfs-site`	Ändert die EMRFS-Einstellungen.
`spark-metrics`	Ändert die Werte in der metrics.properties-Datei in Spark.
`spark-defaults`	Ändert die Werte in der spark-defaults.conf-Datei in Spark.
`spark-env`	Ändert die Werte in der Spark-Umgebung.
`spark-hive-site`	Ändern Sie die Werte in der hive-site.xml-Datei in Spark.
`spark-log4j`	Ändert die Werte in der log4j.properties-Datei in Spark.

Mithilfe von Konfigurationsklassifizierungen können Sie Anwendungen anpassen. Diese entsprechen häufig einer XML-Konfigurationsdatei für die Anwendung, z. B. spark-hive-site .xml. Weitere Informationen finden Sie unter Configuring Applications.

emr-6.2.0-latest

Versionshinweise: `emr-6.2.0-latest` verweist derzeit auf `emr-6.2.0-20240321`.

Regionen: `emr-6.2.0-latest` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.2.0:20240321`

emr-6.2.0-20240321

Versionshinweise: `emr-6.2.0-20240321` wurde am 11. März 2024 veröffentlicht. Im Vergleich zur vorherigen Version wurde diese Version mit den kürzlich aktualisierten Amazon Linux-Paketen und kritischen Korrekturen aktualisiert.

Regionen: `emr-6.2.0-20240321` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.2.0:20240321`

emr-6.2.0-20220802

Versionshinweise: `emr-6.2.0-20220802` wurde am 27. September 2022 veröffentlicht. Im Vergleich zur Vorgängerversion wurde diese Version mit den kürzlich aktualisierten Amazon Linux-Paketen aktualisiert.

Regionen: `emr-6.2.0-20220802` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-6.2.0:20220802`

emr-6.2.0-20211008

Versionshinweise: `emr-6.2.0-20211008` wurde am 9. Dezember 2021 veröffentlicht. Im Vergleich zur Vorgängerversion enthält diese Version Problembehebungen und Sicherheitsupdates.

Regionen: `emr-6.2.0-20211008` ist in den folgenden Regionen verfügbar: USA Ost (Nord-Virginia), USA West (Oregon), Asien-Pazifik (Tokio), Europa (Irland), Südamerika (São Paulo).

Container-Image-Tag: `emr-6.2.0:20211008`

emr-6.2.0-20210802

Versionshinweise: `emr-6.2.0-20210802` wurde am 2. August 2021 veröffentlicht. Im Vergleich zur Vorgängerversion enthält diese Version Problembehebungen und Sicherheitsupdates.

Regionen: `emr-6.2.0-20210802` ist in den folgenden Regionen verfügbar: USA Ost (Nord-Virginia), USA West (Oregon), Asien-Pazifik (Tokio), Europa (Irland), Südamerika (São Paulo).

Container-Image-Tag: `emr-6.2.0:20210802`

emr-6.2.0-20210615

Versionshinweise: `emr-6.2.0-20210615` wurde am 15. Juni 2021 veröffentlicht. Im Vergleich zur Vorgängerversion enthält diese Version Problembehebungen und Sicherheitsupdates.

Regionen: `emr-6.2.0-20210615` ist in den folgenden Regionen verfügbar: USA Ost (Nord-Virginia), USA West (Oregon), Asien-Pazifik (Tokio), Europa (Irland), Südamerika (São Paulo).

Container-Image-Tag: `emr-6.2.0:20210615`

emr-6.2.0-20210129

Versionshinweise: `emr-6.2.0-20210129` wurde am 29. Januar 2021 veröffentlicht. Im Vergleich zu `emr-6.2.0-20201218` enthält diese Version Problembehebungen und Sicherheitsupdates.

Regionen: `emr-6.2.0-20210129` ist in den folgenden Regionen verfügbar: USA Ost (Nord-Virginia), USA West (Oregon), Asien-Pazifik (Tokio), Europa (Irland), Südamerika (São Paulo).

Container-Image-Tag: `emr-6.2.0-20210129`

emr-6.2.0-20201218

Versionshinweise: `emr-6.2.0-20201218` wurde am 18. Dezember 2020 veröffentlicht. Im Vergleich zu `emr-6.2.0-20201201` enthält diese Version Problembehebungen und Sicherheitsupdates.

Regionen: `emr-6.2.0-20201218` ist in den folgenden Regionen verfügbar: USA Ost (Nord-Virginia), USA West (Oregon), Asien-Pazifik (Tokio), Europa (Irland), Südamerika (São Paulo).

Container-Image-Tag: `emr-6.2.0-20201218`

emr-6.2.0-20201201

Versionshinweise: `emr-6.2.0-20201201` wurde am 1. Dezember 2020 veröffentlicht. Dies ist die erste veröffentlichte Version von Amazon EMR 6.2.0.

Regionen: `emr-6.2.0-20201201` ist in den folgenden Regionen verfügbar: USA Ost (Nord-Virginia), USA West (Oregon), Asien-Pazifik (Tokio), Europa (Irland), Südamerika (São Paulo).

Container-Image-Tag: `emr-6.2.0-20201201`

Versionen von Amazon EMR in EKS 5.36.0

Die folgenden Versionen von Amazon EMR 5.36.0 sind für Amazon EMR in EKS verfügbar. Wählen Sie eine bestimmte emr-5.36.0-XXXX-Version aus, um weitere Details wie das zugehörige Container-Image-Tag anzuzeigen.

- emr-5.36.0-latest
- emr-5.36.0-20240321
- emr-5.36.0-20221219
- emr-5.36.0-20220620
- emr-5.36.0-20220525

Versionshinweise für Amazon EMR 5.36.0

- Log4j2-Sicherheitsprobleme wurden behoben.
- Unterstützte Anwendungen – Spark 2.4.8-amzn-2, Jupyter Enterprise Gateway (Endpunkte, öffentliche Vorschau; Scala-Kernel wird nicht unterstützt), livy-0.7.1, fluentd-4.0.0.
- Unterstützte Komponenten - aws-hm-client,, emr-ddb aws-sagemaker-spark-sdk, emr-goodies, emr-kinesis, kerberos-Server.
- Unterstützte Konfigurationsklassifizierungen:

Klassifizierungen	Beschreibungen
`core-site`	Ändern Sie die Werte in der core-site.xml-Datei in Hadoop.
`emrfs-site`	Ändert die EMRFS-Einstellungen.
`spark-metrics`	Ändert die Werte in der metrics.properties-Datei in Spark.
`spark-defaults`	Ändert die Werte in der spark-defaults.conf-Datei in Spark.

Klassifizierungen	Beschreibungen
`spark-env`	Ändert die Werte in der Spark-Umgebung.
`spark-hive-site`	Ändern Sie die Werte in der hive-site.xml-Date i in Spark.
`spark-log4j`	Ändert die Werte in der log4j.properties-Datei in Spark.

Mithilfe von Konfigurationsklassifizierungen können Sie Anwendungen anpassen. Diese entsprechen häufig einer XML-Konfigurationsdatei für die Anwendung, z. B. .xml. spark-hive-site Weitere Informationen finden Sie unter Configuring Applications.

emr-5.36.0-latest

Versionshinweise: `emr-5.36.0-latest` verweist derzeit auf `emr-5.36.0-20240321`.

Regionen: `emr-5.36.0-latest` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-5.36.0:latest`

emr-5.36.0-20240321

Versionshinweise: `emr-5.36.0-20240321` wurde am 11. März 2024 veröffentlicht. Im Vergleich zur vorherigen Version wurde diese Version mit den kürzlich aktualisierten Amazon Linux-Paketen und kritischen Korrekturen aktualisiert.

Regionen: `emr-5.36.0-20240321` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-5.36.0:20240321`

emr-5.36.0-20221219

Versionshinweise: `emr-5.36.0-20221219` wurde am 27. Januar 2023 veröffentlicht. Im Vergleich zur Vorgängerversion wurde diese Version mit den kürzlich aktualisierten Amazon Linux-Paketen aktualisiert.

Regionen: `emr-5.36.0-20221219` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-5.36.0:20221219`

emr-5.36.0-20220620

Versionshinweise: `emr-5.36.0-20220620` wurde am 27. Juli 2022 veröffentlicht. Im Vergleich zur Vorgängerversion wurde diese Version mit den kürzlich aktualisierten Amazon Linux-Paketen aktualisiert.

Regionen: `emr-5.36.0-20220620` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-5.36.0:20220620`

emr-5.36.0-20220525

Versionshinweise: `emr-5.36.0-20220525` wurde am 16. Juni 2022 veröffentlicht. Dies ist die erste veröffentlichte Version von Amazon EMR 5.36.0.

Regionen: `emr-5.36.0-20220525` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-5.36.0:20220525`

Versionen von Amazon EMR in EKS 5.35.0

Die folgenden Versionen von Amazon EMR 5.35.0 sind für Amazon EMR in EKS verfügbar. Wählen Sie eine bestimmte emr-5.35.0-XXXX-Version aus, um weitere Details wie das zugehörige Container-Image-Tag anzuzeigen.

- emr-5.35.0-latest
- emr-5.35.0-20240321
- emr-5.35.0-20221219
- emr-5.35.0-20220802
- emr-5.35.0-20220307

Versionshinweise für Amazon EMR 5.35.0

- Log4j2-Sicherheitsprobleme wurden behoben.
- Unterstützte Anwendungen – Spark 2.4.8-amzn-1, Hudi 0.9.0-amzn-2, Jupyter Enterprise Gateway (Endpunkte, öffentliche Vorschau; Scala-Kernel wird nicht unterstützt).
- Unterstützte Komponenten - aws-hm-client (Glue-Anschluss), emr-s3-select aws-sagemaker-spark-sdk, emrfs, emr-ddb, hudi-spark.
- Unterstützte Konfigurationsklassifizierungen:

Klassifizierungen	Beschreibungen
`core-site`	Ändern Sie die Werte in der core-site.xml-Datei in Hadoop.
`emrfs-site`	Ändert die EMRFS-Einstellungen.
`spark-metrics`	Ändert die Werte in der metrics.properties-Datei in Spark.
`spark-defaults`	Ändert die Werte in der spark-defaults.conf-Datei in Spark.
`spark-env`	Ändert die Werte in der Spark-Umgebung.
`spark-hive-site`	Ändern Sie die Werte in der hive-site.xml-Datei in Spark.
`spark-log4j`	Ändert die Werte in der log4j.properties-Datei in Spark.

Mithilfe von Konfigurationsklassifizierungen können Sie Anwendungen anpassen. Diese entsprechen häufig einer XML-Konfigurationsdatei für die Anwendung, z. B. .xml. spark-hive-site Weitere Informationen finden Sie unter Configuring Applications.

emr-5.35.0-latest

Versionshinweise: `emr-5.35.0-latest` verweist derzeit auf `emr-5.35.0-20240321`.

Regionen: `emr-5.35.0-latest` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-5.35.0:latest`

emr-5.35.0-20240321

Versionshinweise: `emr-5.35.0-20240321` wurde am 11. März 2024 veröffentlicht. Im Vergleich zur vorherigen Version wurde diese Version mit den kürzlich aktualisierten Amazon Linux-Paketen und kritischen Korrekturen aktualisiert.

Regionen: `emr-5.35.0-20240321` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-5.35.0:20240321`

emr-5.35.0-20221219

Versionshinweise: `emr-5.35.0-20221219` wurde am 27. Januar 2023 veröffentlicht. Im Vergleich zur Vorgängerversion wurde diese Version mit den kürzlich aktualisierten Amazon Linux-Paketen aktualisiert.

Regionen: `emr-5.35.0-20221219` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-5.35.0:20221219`

emr-5.35.0-20220802

Versionshinweise: `emr-5.35.0-20220802` wurde am 27. September 2022 veröffentlicht. Im Vergleich zur Vorgängerversion wurde diese Version mit den kürzlich aktualisierten Amazon Linux-Paketen aktualisiert.

Regionen: `emr-5.35.0-20220802` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-5.35.0:20220802`

emr-5.35.0-20220307

Versionshinweise: `emr-5.35.0-20220307` wurde am 30. März 2022 veröffentlicht. Im Vergleich zur Vorgängerversion wurde diese Version mit den kürzlich aktualisierten Amazon Linux-Paketen aktualisiert.

Regionen: `emr-5.35.0-20220307` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-5.35.0:20220307`

Versionen von Amazon EMR in EKS 5.34.0

Die folgenden Versionen von Amazon EMR 5.34.0 sind für Amazon EMR in EKS verfügbar. Wählen Sie eine bestimmte emr-5.34.0-XXXX-Version aus, um weitere Details wie das zugehörige Container-Image-Tag anzuzeigen.

- emr-5.34.0-latest
- emr-5.34.0-20240321
- emr-5.34.0-20220802

Versionshinweise für Amazon EMR 5.34.0

- Unterstützte Anwendungen – Spark 2.4.8-amzn-0, Jupyter Enterprise Gateway (Endpunkte, öffentliche Vorschau; Scala-Kernel wird nicht unterstützt).

- Unterstützte Komponenten – `aws-hm-client` (Glue-Konnektor), `aws-sagemaker-spark-sdk`, `emr-s3-select`, `emrfs`, `emr-ddb`, `hudi-spark`.
- Unterstützte Konfigurationsklassifizierungen:

Klassifizierungen	Beschreibungen
`core-site`	Ändern Sie die Werte in der core-site.xml-Datei in Hadoop.
`emrfs-site`	Ändert die EMRFS-Einstellungen.
`spark-metrics`	Ändert die Werte in der metrics.properties-Datei in Spark.
`spark-defaults`	Ändert die Werte in der spark-defaults.conf-Datei in Spark.
`spark-env`	Ändert die Werte in der Spark-Umgebung.
`spark-hive-site`	Ändern Sie die Werte in der hive-site.xml-Datei in Spark.
`spark-log4j`	Ändert die Werte in der log4j.properties-Datei in Spark.

Mithilfe von Konfigurationsklassifizierungen können Sie Anwendungen anpassen. Diese entsprechen häufig einer XML-Konfigurationsdatei für die Anwendung, z. B. spark-hive-site .xml. Weitere Informationen finden Sie unter Configuring Applications.

emr-5.34.0-latest

Versionshinweise: `emr-5.34.0-latest` verweist derzeit auf `emr-5.34.0-20220802`.

Regionen: `emr-5.34.0-latest` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-5.34.0:latest`

emr-5.34.0-20240321

Versionshinweise: `emr-5.34.0-20240321` wurde am 11. März 2024 veröffentlicht. Im Vergleich zur vorherigen Version wurde diese Version mit den kürzlich aktualisierten Amazon Linux-Paketen und kritischen Korrekturen aktualisiert.

Regionen: `emr-5.34.0-20240321` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-5.34.0:20240321`

emr-5.34.0-20220802

Versionshinweise: `emr-5.34.0-20220802` wurde am 24. August 2022 veröffentlicht. Im Vergleich zur Vorgängerversion wurde diese Version mit den kürzlich aktualisierten Amazon Linux-Paketen aktualisiert.

Regionen: `emr-5.34.0-20220802` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-5.34.0:20220802`

emr-5.34.0-20211208

Versionshinweise: `emr-5.34.0-20211208` wurde am 20. Januar 2022 veröffentlicht. Im Vergleich zur Vorgängerversion wurde diese Version mit den kürzlich aktualisierten Amazon Linux-Paketen aktualisiert.

Regionen: `emr-5.34.0-20211208` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-5.34.0:20211208`

Versionen von Amazon EMR in EKS 5.33.0

Die folgenden Versionen von Amazon EMR 5.33.0 sind für Amazon EMR in EKS verfügbar. Wählen Sie eine bestimmte emr-5.33.0-XXXX-Version aus, um weitere Details wie das zugehörige Container-Image-Tag anzuzeigen.

- emr-5.33.0-latest
- emr-5.33.0-20240321
- emr-5.33.0-20221219
- emr-5.33.0-20220802
- emr-5.33.0-20211008
- emr-5.33.0-20210802
- emr-5.33.0-20210615
- emr-5.33.0-20210323

Versionshinweise für Amazon EMR 5.33.0

- Neues Feature – Ab Amazon EMR 5.33.0 in der 5.x-Versions-Serie unterstützt Amazon EMR in EKS das Pod-Vorlagenfeature von Spark. Weitere Informationen finden Sie unter Verwenden von Pod-Vorlagen.
- Unterstützte Anwendungen – Spark 2.4.7-amzn-1, Jupyter Enterprise Gateway (Endpunkte, öffentliche Vorschau; Scala-Kernel wird nicht unterstützt).
- Unterstützte Komponenten – `aws-hm-client` (Glue-Konnektor), `aws-sagemaker-spark-sdk`, `emr-s3-select`, `emrfs`, `emr-ddb`, `hudi-spark`.
- Unterstützte Konfigurationsklassifizierungen:

Klassifizierungen	Beschreibungen
`core-site`	Ändern Sie die Werte in der core-site.xml-Datei in Hadoop.
`emrfs-site`	Ändert die EMRFS-Einstellungen.
`spark-metrics`	Ändert die Werte in der metrics.properties-Datei in Spark.
`spark-defaults`	Ändert die Werte in der spark-defaults.conf-Datei in Spark.
`spark-env`	Ändert die Werte in der Spark-Umgebung.

Klassifizierungen	Beschreibungen
`spark-hive-site`	Ändern Sie die Werte in der hive-site.xml-Date i in Spark.
`spark-log4j`	Ändert die Werte in der log4j.properties-Datei in Spark.

Mithilfe von Konfigurationsklassifizierungen können Sie Anwendungen anpassen. Diese entsprechen häufig einer XML-Konfigurationsdatei für die Anwendung, z. B. spark-hive-site .xml. Weitere Informationen finden Sie unter Configuring Applications.

emr-5.33.0-latest

Versionshinweise: `emr-5.33.0-latest` verweist derzeit auf `emr-5.33.0-20240321`.

Regionen: `emr-5.33.0-latest` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-5.33.0:latest`

emr-5.33.0-20240321

Versionshinweise: `emr-5.33.0-20240321` wurde am 11. März 2024 veröffentlicht. Im Vergleich zur vorherigen Version wurde diese Version mit den kürzlich aktualisierten Amazon Linux-Paketen und kritischen Korrekturen aktualisiert.

Regionen: `emr-5.33.0-20240321` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-5.33.0:20240321`

emr-5.33.0-20221219

Versionshinweise: `emr-5.33.0-20221219` wurde am 19. Januar 2023 veröffentlicht. Im Vergleich zur Vorgängerversion wurde diese Version mit den kürzlich aktualisierten Amazon Linux-Paketen und kritischen Korrekturen aktualisiert.

Regionen: `emr-5.33.0-20221219` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-5.33.0:20221219`

emr-5.33.0-20220802

Versionshinweise: `emr-5.33.0-20220802` wurde am 24. August 2022 veröffentlicht. Im Vergleich zur Vorgängerversion wurde diese Version mit den kürzlich aktualisierten Amazon Linux-Paketen aktualisiert.

Regionen: `emr-5.33.0-20220802` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-5.33.0:20220802`

emr-5.33.0-20211008

Versionshinweise: `emr-5.33.0-20211008` wurde am 9. Dezember 2021 veröffentlicht. Im Vergleich zur Vorgängerversion enthält diese Version Problembehebungen und Sicherheitsupdates.

Regionen: `emr-5.33.0-20211008` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-5.33.0:20211008`

emr-5.33.0-20210802

Versionshinweise: `emr-5.33.0-20210802` wurde am 2. August 2021 veröffentlicht. Im Vergleich zur Vorgängerversion enthält diese Version Problembehebungen und Sicherheitsupdates.

Regionen: `emr-5.33.0-20210802` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-5.33.0:20210802`

emr-5.33.0-20210615

Versionshinweise: `emr-5.33.0-20210615` wurde am 15. Juni 2021 veröffentlicht. Im Vergleich zur Vorgängerversion enthält diese Version Problembehebungen und Sicherheitsupdates.

Regionen: `emr-5.33.0-20210615` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-5.33.0:20210615`

emr-5.33.0-20210323

Versionshinweise: `emr-5.33.0-20210323` wurde am 23. März 2021 veröffentlicht. Dies ist die erste veröffentlichte Version von Amazon EMR 5.33.0.

Regionen: `emr-5.33.0-20210323` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-5.33.0-20210323`

Versionen von Amazon EMR in EKS 5.32.0

Die folgenden Versionen von Amazon EMR 5.32.0 sind für Amazon EMR in EKS verfügbar. Wählen Sie eine bestimmte emr-5.32.0-XXXX-Version aus, um weitere Details wie das zugehörige Container-Image-Tag anzuzeigen.

- emr-5.32.0-latest
- emr-5.32.0-20240321
- emr-5.32.0-20220802
- emr-5.32.0-20211008
- emr-5.32.0-20210802
- emr-5.32.0-20210615
- emr-5.32.0-20210129
- emr-5.32.0-20201218
- emr-5.32.0-20201201

Versionshinweise für Amazon EMR 5.32.0

- Unterstützte Anwendungen – Spark 2.4.7-amzn-0, Jupyter Enterprise Gateway (Endpunkte, öffentliche Vorschau; Scala-Kernel wird nicht unterstützt).
- Unterstützte Komponenten – `aws-hm-client` (Glue-Konnektor), `aws-sagemaker-spark-sdk`, `emr-s3-select`, `emrfs`, `emr-ddb`, `hudi-spark`.
- Unterstützte Konfigurationsklassifizierungen:

Klassifizierungen	Beschreibungen
`core-site`	Ändern Sie die Werte in der core-site.xml-Datei in Hadoop.
`emrfs-site`	Ändert die EMRFS-Einstellungen.
`spark-metrics`	Ändert die Werte in der metrics.properties-Datei in Spark.
`spark-defaults`	Ändert die Werte in der spark-defaults.conf-Datei in Spark.
`spark-env`	Ändert die Werte in der Spark-Umgebung.
`spark-hive-site`	Ändern Sie die Werte in der hive-site.xml-Datei in Spark.
`spark-log4j`	Ändert die Werte in der log4j.properties-Datei in Spark.

Mithilfe von Konfigurationsklassifizierungen können Sie Anwendungen anpassen. Diese entsprechen häufig einer XML-Konfigurationsdatei für die Anwendung, z. B. spark-hive-site .xml. Weitere Informationen finden Sie unter Configuring Applications.

emr-5.32.0-latest

Versionshinweise: `emr-5.32.0-latest` verweist derzeit auf `emr-5.32.0-20240321`.

Regionen: `emr-5.32.0-latest` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-5.32.0:latest`

emr-5.32.0-20240321

Versionshinweise: `emr-5.32.0-20240321` wurde am 11. März 2024 veröffentlicht. Im Vergleich zur vorherigen Version wurde diese Version mit den kürzlich aktualisierten Amazon Linux-Paketen und kritischen Korrekturen aktualisiert.

Regionen: `emr-5.32.0-20240321` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-5.32.0:20240321`

emr-5.32.0-20220802

Versionshinweise: `emr-5.32.0-20220802` wurde am 24. August 2022 veröffentlicht. Im Vergleich zur Vorgängerversion wurde diese Version mit den kürzlich aktualisierten Amazon Linux-Paketen aktualisiert.

Regionen: `emr-5.32.0-20220802` ist in allen Regionen verfügbar, die von Amazon EMR in EKS unterstützt werden. Weitere Informationen finden Sie unter Service-Endpunkte von Amazon EMR in EKS.

Container-Image-Tag: `emr-5.32.0:20220802`

emr-5.32.0-20211008

Versionshinweise: `emr-5.32.0-20211008` wurde am 9. Dezember 2021 veröffentlicht. Im Vergleich zur Vorgängerversion enthält diese Version Problembehebungen und Sicherheitsupdates.

Regionen: `emr-5.32.0-20211008` ist in den folgenden Regionen verfügbar: USA Ost (Nord-Virginia), USA West (Oregon), Asien-Pazifik (Tokio), Europa (Irland), Südamerika (São Paulo).

Container-Image-Tag: `emr-5.32.0:20211008`

emr-5.32.0-20210802

Versionshinweise: `emr-5.32.0-20210802` wurde am 2. August 2021 veröffentlicht. Im Vergleich zur Vorgängerversion enthält diese Version Problembehebungen und Sicherheitsupdates.

Regionen: `emr-5.32.0-20210802` ist in den folgenden Regionen verfügbar: USA Ost (Nord-Virginia), USA West (Oregon), Asien-Pazifik (Tokio), Europa (Irland), Südamerika (São Paulo).

Container-Image-Tag: `emr-5.32.0:20210802`

emr-5.32.0-20210615

Versionshinweise: `emr-5.32.0-20210615` wurde am 15. Juni 2021 veröffentlicht. Im Vergleich zur Vorgängerversion enthält diese Version Problembehebungen und Sicherheitsupdates.

Regionen: `emr-5.32.0-20210615` ist in den folgenden Regionen verfügbar: USA Ost (Nord-Virginia), USA West (Oregon), Asien-Pazifik (Tokio), Europa (Irland), Südamerika (São Paulo).

Container-Image-Tag: `emr-5.32.0:20210615`

emr-5.32.0-20210129

Versionshinweise: `emr-5.32.0-20210129` wurde am 29. Januar 2021 veröffentlicht. Im Vergleich zu `emr-5.32.0-20201218` enthält diese Version Problembehebungen und Sicherheitsupdates.

Regionen: `emr-5.32.0-20210129` ist in den folgenden Regionen verfügbar: USA Ost (Nord-Virginia), USA West (Oregon), Asien-Pazifik (Tokio), Europa (Irland), Südamerika (São Paulo).

Container-Image-Tag: `emr-5.32.0-20210129`

emr-5.32.0-20201218

Versionshinweise: `5.32.0-20201218` wurde am 18. Dezember 2020 veröffentlicht. Im Vergleich zu `5.32.0-20201201` enthält diese Version Problembehebungen und Sicherheitsupdates.

Regionen: `emr-5.32.0-20201218` ist in den folgenden Regionen verfügbar: USA Ost (Nord-Virginia), USA West (Oregon), Asien-Pazifik (Tokio), Europa (Irland), Südamerika (São Paulo).

Container-Image-Tag: `emr-5.32.0-20201218`

emr-5.32.0-20201201

Versionshinweise: `5.32.0-20201201` wurde am 1. Dezember 2020 veröffentlicht. Dies ist die erste veröffentlichte Version von Amazon EMR 5.32.0.

Regionen: `5.32.0-20201201` ist in den folgenden Regionen verfügbar: USA Ost (Nord-Virginia), USA West (Oregon), Asien-Pazifik (Tokio), Europa (Irland), Südamerika (São Paulo).

Container-Image-Tag: `emr-5.32.0-20201201`

Dokumentverlauf

Die folgende Tabelle beschreibt die wichtigsten Änderungen in der Dokumentation seit der letzten Version von Amazon EMR in EKS. Um mehr Informationen über Aktualisierungen dieser Dokumentation zu erhalten, können Sie einen RSS-Feed abonnieren.

Änderung	Beschreibung	Datum
Neue Version	Versionen von Amazon EMR auf EKS 7.1.0	17. April 2024
Neue Version	Versionen von Amazon EMR in EKS 7.0.0	22. Dezember 2023
Neue Version	Amazon EMR in EKS 6.15.0-Versionen	17. November 2023
Neue Version	Versionen von Amazon EMR in EKS 6.14.0	17. Oktober 2023
Inhalt aktualisieren	Benennen Sie „verwaltete Endpunkte“ in interaktive Endpunkte um. Interaktive Endpunkte sind allgemein verfügbar	29. September 2023
Neue Version	Versionen von Amazon EMR in EKS 6.13.0 und öffentliche Vorschaud okumente für Ausführen von Flink-Auf trägen mit Amazon EMR in EKS	12. September 2023
Neue Version	Versionen von Amazon EMR in EKS 6.12.0	21. Juli 2023
Neuer Inhalt	Verwendung von Volcano als benutzerd efiniertem Scheduler für Apache Spark auf Amazon EMR in EKS hinzugefügt	13. Juni 2023
Neuer Inhalt	Verwendung von Volcano als benutzerd efiniertem Scheduler für Apache Spark auf Amazon EMR in EKS hinzugefügt	13. Juni 2023

Änderung	Beschreibung	Datum
Neuer Inhalt	Verwenden der Spark-Container-Protokoll-Rotation hinzugefügt	12. Juni 2023
Inhalt aktualisieren	Die benutzerdefinierte Image-Dokumentation für die Suche nach Basis-Image-informationen in der Amazon ECR Public Gallery wurde aktualisiert.	08. Juni 2023
Neue Version	Versionen von Amazon EMR in EKS 6.11.0	08. Juni 2023
Neuer Inhalt	Spark-Aufträge mit dem Spark-Operator ausführen hinzugefügt und die Abschnitte Auftragsausführungen wurden unter Ausführen von Aufträgen mit Amazon EMR in EKS neu organisiert.	5. Juni 2023
Neuer Inhalt	Es wurden zwei Abschnitte: Verwenden von vertikalem Auto Scaling mit Amazon-EMR-Spark-Aufträgen und Verwenden von selbst gehosteten Jupyter Notebooks hinzugefügt	4. Mai 2023
Dokumentverlaufsseite	Erstellt eine Dokumentverlaufsseite für Amazon EMR in EKS.	13. März 2023
Seite Verwaltete Richtlinien	Erstellt eine Seite mit verwalteten Richtlinien für Amazon EMR in EKS.	13. März 2023

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.